Pro Power BI Desktop

Interactive data analysis and visualization for the desktop

—

Second Edition

—

Adam Aspin

Apress®

Pro Power BI Desktop

Second Edition

Adam Aspin

Apress®

Pro Power BI Desktop

Adam Aspin
Stoke-on-Trent, Staffordshire,
United Kingdom

ISBN-13 (pbk): 978-1-4842-3209-5 ISBN-13 (electronic): 978-1-4842-3210-1
https://doi.org/10.1007/978-1-4842-3210-1

Library of Congress Control Number: 2017962307

Cover image designed by Freepik

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Technical Reviewer: Chad Schuessler
Coordinating Editor: Jill Balzano
Copy Editor: Bill McManus
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail `rights@apress.com`, or visit `http://www.apress.com/rights-permissions`.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at `http://www.apress.com/bulk-sales`.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at `www.apress.com/9781484232095`. For more detailed information, please visit `http://www.apress.com/source-code`.

Printed on acid-free paper

To the memories of my mother and grandmother.

Contents

About the Author

Adam Aspin is an independent Business Intelligence consultant based in the United Kingdom. He has worked with SQL Server for over 20 years. During this time, he has developed several dozen reporting and analytical systems based on the Microsoft BI product suite.

A graduate of Oxford University, Adam began his career in publishing before moving into IT. Databases soon became a passion, and his experience in this arena ranges from Access to Oracle and MySQL, with occasional sorties into the world of DB2. He is, however, most at home in the Microsoft universe when using SQL Server Analysis Services, SQL Server Reporting Services, SQL Server Integration Services, SharePoint, and Power BI.

Business Intelligence has been his principal focus for the last 15 years. He has applied his skills for a range of clients in industry sectors from Finance to Utilities, Telecoms to Insurance, and Manufacturing to Banking.

Adam is a frequent contributor to SQLServerCentral.com and Simple-Talk. He has written numerous articles for various French IT publications. A fluent French speaker, Adam has worked in France and Switzerland for many years. He speaks regularly at events such as SQLBits, SQL Saturdays, and local SQL Server user groups.

He is the author of *SQL Server 2012 Data Integration Recipes* (Apress, 2012), *High Impact Data Visualization in Excel with Power View, 3D Maps, Get & Transform and Power BI* (Apress, 2016), and *Business Intelligence with SQL Server Reporting Services* (Apress, 2015).

About the Technical Reviewer

Chad Schuessler is an experienced database consultant specializing in Business Intelligence. In addition to many programming languages, Chad has worked with a wide range of database technologies over the past 22 years including Informix, Oracle, DB2, MySQL, and Sybase. Microsoft SQL Server, however, has been the primary platform he has used since 2000, implementing solutions across Manufacturing, Health Care, Retail, Financial Services, Automotive, and Energy.

Being a dual American-British citizen, Chad has been able to gain international experience working extensively in both America and England.

Acknowledgments

Writing a technical book can be a lonely occupation. So I am all the more grateful for all the help and encouragement that I have received from so many fabulous friends and colleagues.

First, my considerable thanks go to Jonathan Gennick, the commissioning editor of this book. Throughout the publication process Jonathan has been both a tower of strength and an exemplary mentor. He has always been available to share his vast experience selflessly and courteously.

Heartfelt thanks go to Jill Balzano, the Apress coordinating editor, for managing this book through the production process. She succeeded—once again—in the well-nigh impossible task of making a potentially stress-filled trek into a pleasant journey filled with light and humor. Her team also deserves much praise for their calm under pressure.

I owe a deep debt of gratitude to Chad Schuessler, who reviewed this book from a technical standpoint. He took far too many hours away from his young family to go through all the detail in the book and provided a plethora of valuable insights and recommendations. His vast experience of Power BI was invaluable. Thanks Chad!

My thanks also go to Bill McManus for his tireless and subtle work editing and polishing the prose.

Once again, my deepest gratitude is reserved for the two people who have given the most to this book. They are my wife and son. Timothy has put up with a mentally absent father for months while nonetheless providing continual encouragement to persevere. Karine has not only given me the support and encouragement to persevere, but also the love without which nothing would be worth any effort. I am a very lucky man to have both of them.

Introduction

Business Intelligence has become one of the buzzwords that defines an age. Its younger sibling, self-service BI, has attained the status of a holy grail for businesses. Managers want their staff to deliver insight in seconds, and users just want to do their jobs quickly and to produce clear, telling, and accurate analyses with tools that are intuitive and easy to use.

Microsoft recognized these trends and needs a short time ago when it produced the first version of Power BI Desktop. Moreover, Microsoft made this amazing toolkit absolutely free.

Since its launch, the product has grown beyond recognition. It now allows any user to take data from virtually any source and use it to produce stunning dashboards and compelling reports that will seize their audience's attention. Using this rapidly evolving tool, any user can slice and dice the data with remarkable ease and then add metrics, instant analyses, and KPIs to project the insights that create a real competitive advantage.

This book shows how to deliver eye-catching Business Intelligence with Microsoft Power BI Desktop. It teaches you how to make raw data into clear, accurate, and interactive information. The aim of this book is to help you to push your BI delivery to the next level. In these chapters you will learn to create great-looking visualizations and let your audience have fun by interacting with the elegant and visually arresting output that you can now deliver. You will see how to choose from a wide range of built-in and third-party visualization types so that your message is always enhanced. You'll discover ways to deliver those results on the PC, on tablets, and on smartphones, as well as share results via the cloud. Finally, this book helps you save time by preparing the underlying data correctly without needing an IT department to prepare it for you. Power BI Desktop will let your analyses speak for themselves. This book will help you learn how to unleash its vast resources.

If you wish, you can read this book from start to finish, as it is designed to be a progressive tutorial that will help you to learn Power BI Desktop. However, as Power BI Desktop is composed of several interdependent BI functions, this book is broken down into groups of chapters that focus on the various areas of the product.

- Chapter 1 introduces new users to the product. You can consider it a kind of "executive summary" of Power BI Desktop.
- Chapters 2 through 5 show you how to connect to a range of varied data sources and bring this data into Power BI Desktop.
- Chapters 6 through 9 explain how to transform and clean data so that you can use it for analysis.
- Chapter 10 introduces you to the art of creating a data model on which you can base your interactive dashboards and reports.
- Chapters 11 through 13 are an introduction to DAX: the Power BI Desktop language that you use to advance and tailor your analysis. You will use this language to create the metrics for your business intelligence.

- Chapters 14 through 22 take you through the myriad possibilities that Power BI Desktop offers to help you create stunning reports and dashboards. This covers tables, charts, maps, KPIs, and many other types of visuals.
- Chapter 23 explains how you can share your insights using PowerBI.com. This is the cloud-based service available from Microsoft that lets colleagues and friends see and interact with your reports and dashboards.

This is the second edition of *Pro Power BI Desktop*. In the course of the year and a half since the first edition, Power BI Desktop has evolved magisterially. So a new edition of the book was necessary to update the original with all the new and exciting changes and extensions that have been added to the product. Indeed, the monthly release cycle that Microsoft has maintained—and is still maintaining—is proof of the company's dedication to improving an already outstanding product. This book includes the latest updates added in September 2017.

This book comes with a small set of sample data that is used to create the examples that are presented throughout the book. I realize that it may seem paradoxical to use a tiny data set for a product that can handle tens of millions of rows of data, but I prefer to use a comprehensible set of source data so that the reader can concentrate on what is being learned, rather than the data itself.

It is inevitable that not every question can be anticipated and answered in one book. Nonetheless, I hope that I have answered many of the self-service BI questions that you might encounter and-more importantly-have given you the approaches and the confidence to resolve most of the Power BI Desktop challenges that you might encounter when applying this product to solve real-world problems.

I wish you good luck in using Power BI Desktop and I sincerely hope that you have as much fun with it as I did in writing this book.

CHAPTER 1

Introduction to Power BI Desktop

If you are reading this book, it is probably because you need to use data. More specifically, you want to take a journey from data through to insight where quantities of facts and figures need to be shaped into comprehensible information and given clear and visual meaning. Once you have correlated and isolated your analyses, you probably want to share them with colleagues or friends.

This book is all about that journey. It covers the many ways that you can transform raw data into high-impact analyses using Power BI Desktop—the self-service business intelligence (BI) and analytics application that Microsoft is making freely available. This fresh approach to self-service BI presumes minimal central IT intervention or dependency. It is based on giving the user the ability to use simple yet powerful tools to handle industrial-strength quantities of data and to share stunning output in the shortest possible timescales.

The following are keywords in this universe:

- Speed
- Delivery
- Empowerment
- Decentralization
- Disintermediation

Once you have mastered the tools and techniques described in this book, you will be able to

- Discover, structure, and load your data from a wide range of sources
- Add all the calculations you need to enhance information and extract accurate analysis
- Create stylish interactive presentations
- Share your insights with your colleagues and clients

It follows that this book is written from the perspective of the user. Essentially, it is all about giving users the tools and knowledge to define their own requirements and satisfy their own needs simply and efficiently through developing new and existing skills.

This chapter assumes that you have no prior knowledge of Power BI Desktop or even business intelligence. Consequently, it starts from the very beginning by explaining what exactly self-service business intelligence really is. Then it takes you through the necessary steps to download and install Power BI Desktop. Finally, it takes you on a whirlwind tour of Power BI Desktop, where you see just how quickly and easily you can go from raw data to polished insight using this amazing tool.

If you intend to follow the dashboard example that you find in this chapter (and I hope that you will), then you need to download the source material for this book from the Apress web site at www.apress.com/9781484232095. Appendix A describes how to do this so that you can prepare the terrain for your upcoming adventure with Power BI Desktop.

The Microsoft Self-Service Business Intelligence Solution

It is important to understand from the start that Microsoft's self-service business intelligence solution, Power BI, is a constantly evolving process. Indeed, it is in a continuous state of flux. Fortunately, this perpetual motion is now at a peak of readiness, and while it is still undergoing some enhancements and revisions, it is ready for immediate use.

So what exactly is Power BI? At its heart, it is a cloud-based service that lets you store and share essential business data in the form of dashboards and reports. However, before you can share dashboards, you need to create them—and this is where Power BI Desktop comes in. This easy-to-use tool is completely free. It is used to find, cleanse, and mash up data so that you can then develop telling metrics and deliver them in the form of stylish visualizations. Once your tables, charts, and maps are assembled into reports, you can then share them with a selected audience in Azure—the Microsoft Cloud (should you want to, of course). Yet the good news does not stop there. Your public can view your insights on just about any Windows, iOS, or Android device using the free Power BI apps that Microsoft has made available for these platforms.

So all that you have to do is to create dashboards with Power BI Desktop, share the output in Azure, and then view and interact with the results using the Power BI apps. It really is that simple. Moreover, up to a certain limit on file sizes, it is completely free. In a large corporate environment, you may even install a Power BI Report Server to make Power BI reports and dashboards available to employees without using the Web to share them.

There is more—much more—in the Power BI universe, but this short description will suffice to get you started. In any case, Chapter 23 provides more detail on the way that Power BI Desktop fits into the Microsoft's self-service business intelligence solution. In the meantime, let's move the focus back to Power BI Desktop. To begin, what exactly will you be using this application for? There are three answers:

- Import data
- Model data
- Create reports and dashboards

Let's take a quick look at some of the things that these may entail.

Importing Data from Diverse Sources

The first step on the path to delivering concrete business intelligence is to find and import all the data that you need for your analysis. Power BI Desktop lets you

- Import data from a wide variety of sources. This covers corporate databases to desktop files, social media to big data.
- Merge data from multiple sources and shape it into a coherent structure.
- Cleanse your data to make it reliable and easy to use.
- Break down the data into the rows and columns that suit your requirements.

There was a time when these tasks required dedicated teams of IT specialists. In fact, it was considered so complex that it earned its own acronym, ETL, short for **E**xtract, **T**ransform, and **L**oad. Well, this process no longer needs specialists. With Power BI Desktop, you can mash up your own data so that it is ready to use as part of your self-service BI solution.

Importing and connecting to data is discussed in Chapters 2 through 5. Mashing up (cleansing, joining, and transforming) data is examined in Chapters 6 through 9.

Modeling Your Data

Power BI Desktop is not just a data store for your information. It also lets you extend and develop the cleansed data. More specifically, it allows you to

- Create a data model by joining tables to develop a coherent data structure from multiple separate sources of data. This data model is then used in dashboards.
- Enrich the data model by applying coherent names and data types.
- Create calculations and prepare the core metrics that you want to use in your analyses and presentations.

It is worth noting that you can load data into Power BI Desktop directly without going through the data cleansing and modeling stage. If the source data is already in good shape, then you can start using it straightaway.

Modeling data and adding calculations is discussed in Chapters 10 through 13.

Creating Reports and Dashboards

I think of creating reports and dashboards as the “jewel in the crown” of self-service business intelligence. A truly dynamic analysis and presentation approach lets you deliver business intelligence composed of

- Tables
- Matrixes
- Charts
- Maps
- Gauges
- Images
- KPIs

and many other types of visualization with Power BI Desktop.

Not only that, but it is incredibly fast and highly intuitive. It provides advanced interactivity so that you and your users can “slice and dice” the data “on the fly” in real time using

- Slicers
- Filters

Creating dashboards and reports is discussed in Chapters 14 through 22.

Power BI Desktop Files

Power BI Desktop lets you create multiple pages in a single file. Each collection of pages that is based on the same underlying data is called a *report*. A Power BI Desktop file therefore contains all the dashboards and all the data that is needed by each element (called a *visualization*) on each page. So, a Power BI Desktop file is completely self-contained.

Power BI Desktop is built to handle vast quantities of data. Fortunately, however, it compresses the data that you load in an extremely efficient way. This means that Power BI Desktop files often take up only a fraction of the space that they would if they contained only the raw source data. Indeed, when connecting to certain data sources, you do not even have to load and compress the data. You can connect directly to the data and use it immediately.

This compression also applies to the data that Power BI Desktop uses when you are modeling data and creating dashboards. This is because Power BI Desktop loads all the data that you are using into memory, where it is compressed to make the most of the available memory. This means that Power BI Desktop is extremely fast to use and normally shows you the results of any changes that you make or any filters that you apply in fractions of a second. This instantaneous interactivity also applies to dashboards that you display in Windows, iOS, or Android apps.

The Power BI Universe

Power BI Desktop is one part of an integrated collection of products and services that are generally known as "Power BI." This universe is constantly changing, but for the moment it is composed of the following:

- *PowerBI.Com*: An Azure-based service where you can create and share (depending on your subscription level) data and dashboards.
- *Power BI Desktop*: The core tool that you use to create Power BI reports. This can involve connecting to multiple data sources and modeling and cleansing the source data.
- *Power BI Report Server*: An on-premises Power BI report server that allows you to distribute and deliver reports inside a corporate firewall.
- *Power BI Mobile Apps*: Apps that allow you to view and interact intuitively with Power BI reports and dashboards on Windows, iOS, and Android devices.
- *Power BI Apps*: Apps that provide a method of collecting and deploying purpose-built dashboards and reports for tailored groups of users in the cloud.
- *Power BI Embedded*: An Azure service that enables application developers to add interactive Power BI reports into their own applications.
- *Power BI Gateway*: Allows you to connect the Azure-based Power BI service to on-premises data sources and automate data refresh.
- *Third-party visuals*: A collection of visual elements, often created by third parties. You can add these visuals to your Power BI reports and dashboards.

It would take an entire book to describe all the elements that make up the Power BI ecosystem-and this is not that book. So, while I will provide a brief overview of third-party visuals in Chapter 18, and an introduction to the Power BI Service, the Power BI Gateway, and Power BI Apps in Chapter 23, I can only suggest that you keep a close eye on the Microsoft Power BI web site (`https://powerbi.microsoft.com`) to keep abreast of the latest developments in this fast-changing product universe.

Installing Power BI Desktop

The first thing that you have to do to create dashboards (or reports or pages of visualizations) is download and install Power BI Desktop. Although this process is really easy, you will save time if you ensure that the computer where you want to install Power BI Desktop has the capability to run this application. Currently, the minimum requirements are as follows:

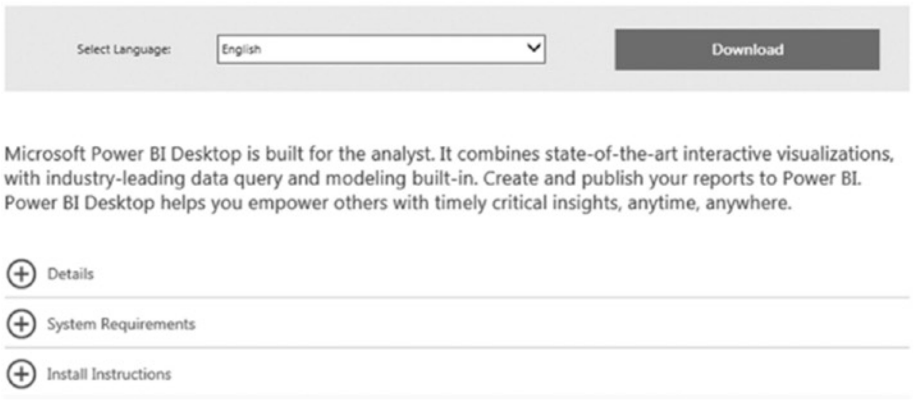
- Windows 7, Windows Server 2008 R2 or later
- Internet Explorer 9 or greater
- .Net 4.5
- At least 1GB of available RAM

■ **Note** Power BI Desktop Designer works on both 32-bit and 64-bit computers. However, if you intend to analyze large datasets, a 64-bit workstation with several gigabytes of memory could very well prove necessary.

Microsoft specifies 1GB as a minimum memory requirement, but you need to be aware that although the application itself is not a memory hog, it can let you load huge amounts of data. Given that all of this data will in most cases be loaded into memory, you need to ensure that you have enough available memory if you intend to analyze large amounts of data. The exception to this principle is when you will be using Power BI Desktop as a "front end" to certain databases or data warehouses, and establishing a direct connection to these sources without loading any data.

So, if you are sure that your PC or laptop is ready for Power BI Desktop, you can install it by following these steps:

1. Go to the Power BI Desktop download page on the Microsoft web site. This is currently `https://www.microsoft.com/en-us/download/details.aspx?id=45331`. You can easily find the right page by entering **Power BI Desktop download** in your favorite search engine. You should see a web page containing something like the information shown in Figure 1-1.

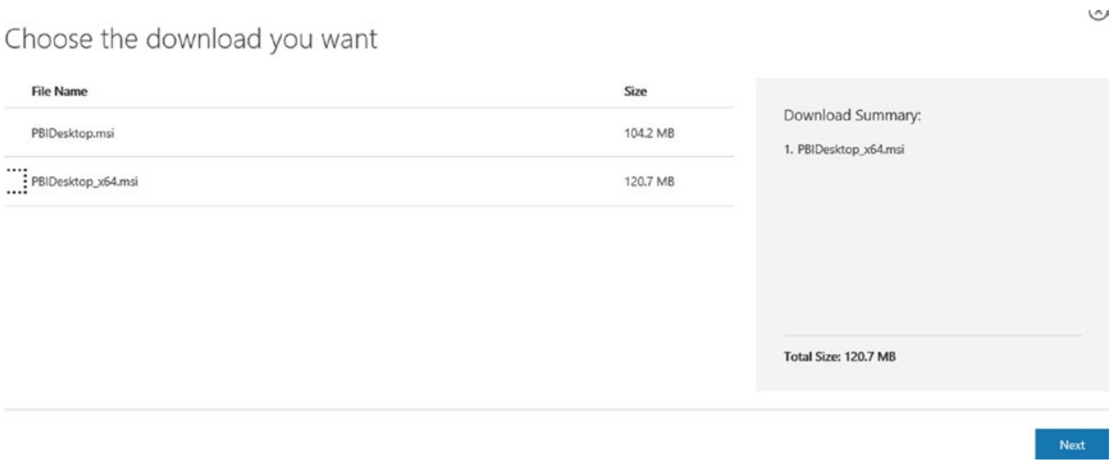


Microsoft Power BI Desktop is built for the analyst. It combines state-of-the-art interactive visualizations, with industry-leading data query and modeling built-in. Create and publish your reports to Power BI. Power BI Desktop helps you empower others with timely critical insights, anytime, anywhere.

Figure 1-1. *The Power BI Desktop download page*

2. Select the language.

3. Click the Download button. You will be taken to the next page, where you should choose the type of download (32-bit or 64-bit), as shown in Figure 1-2.


Choose the download you want

File Name	Size
PBIDesktop.msi	104.2 MB
PBIDesktop_x64.msi	120.7 MB

Download Summary:

1. PBIDesktop_x64.msi

Total Size: 120.7 MB

Figure 1-2. *The download selection page*

4. Click Next (this button only appears once you have selected the type of download that you want). The final download page is displayed. A pop-up appears, as shown in Figure 1-3.

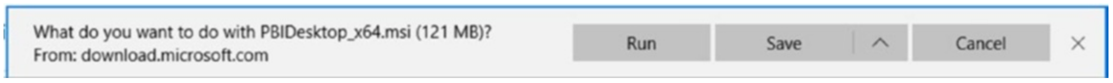
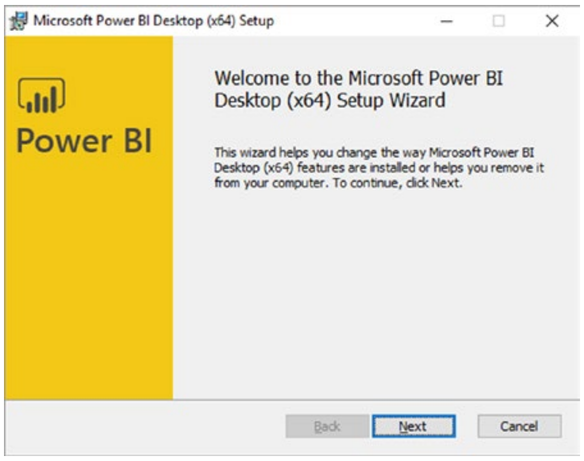

What do you want to do with PBIDesktop_x64.msi (121 MB)?
From: download.microsoft.com

Figure 1-3. *The save or run download popup*

5. Click Run. The Power BI Desktop installation package will download (probably in under a minute) and the initial setup dialog is displayed, as shown in Figure 1-4. If you do not see this dialog once the download has completed, then click the toolbar icon (this will have appeared in the toolbar to make it show on top of any other open windows).

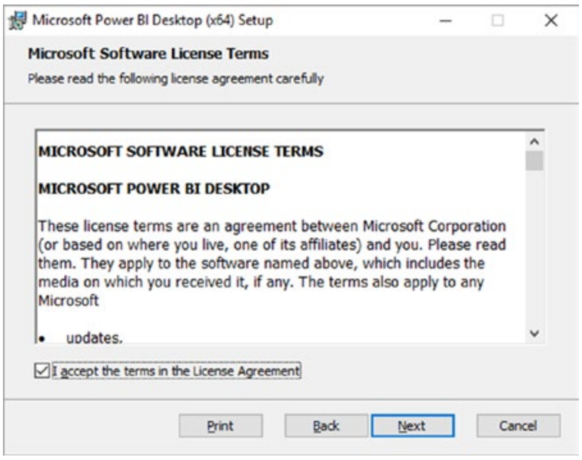

Microsoft Power BI Desktop (x64) Setup

Welcome to the Microsoft Power BI Desktop (x64) Setup Wizard

This wizard helps you change the way Microsoft Power BI Desktop (x64) features are installed or helps you remove it from your computer. To continue, click Next.

Figure 1-4. *The initial Power BI Desktop setup dialog*

6. Click Next. The setup license agreement dialog will appear, as shown in Figure 1-5.


Microsoft Power BI Desktop (x64) Setup
Microsoft Software License Terms
Please read the following license agreement carefully

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT POWER BI DESKTOP

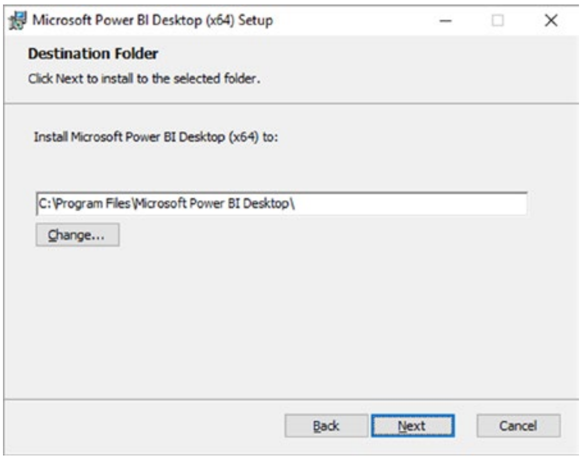
These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft

- updates,

☑ I accept the terms in the License Agreement

Figure 1-5. *The setup license agreement dialog*

7. Ensure that the check box accepting the license agreement is checked and click Next. The setup destination dialog will appear, as shown in Figure 1-6. If you prefer to install the Power BI Desktop files in a different directory, then you can enter it here (or click the Change button and browse to select it).


Microsoft Power BI Desktop (x64) Setup
Destination Folder
Click Next to install to the selected folder.

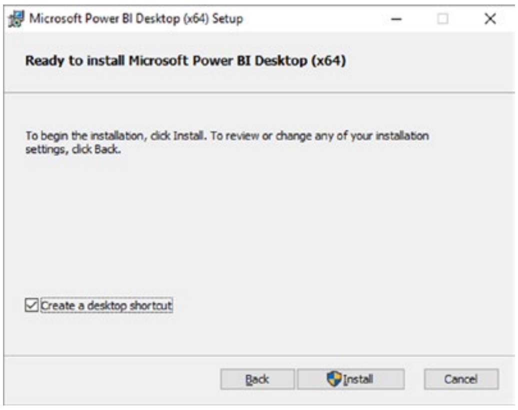
Install Microsoft Power BI Desktop (x64) to:

C:\Program Files\Microsoft Power BI Desktop\

Change...
Back
Next
Cancel

Figure 1-6. *The setup destination dialog*

8. Click Next. The final confirmation dialog will appear, as shown in Figure 1-7.



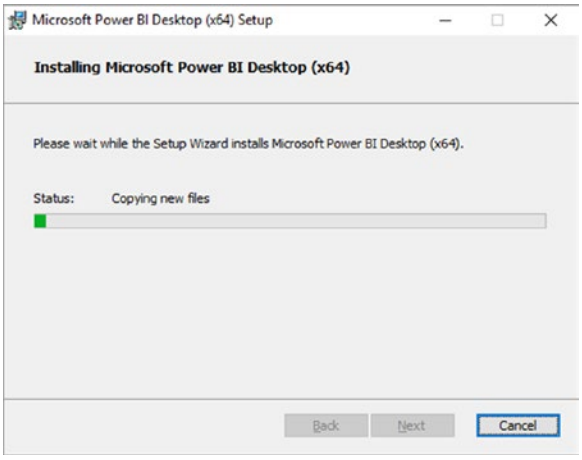
Ready to install Microsoft Power BI Desktop (x64)

To begin the installation, click Install. To review or change any of your installation settings, click Back.

☑ Create a desktop shortcut

Figure 1-7. *The setup final confirmation dialog*

9. Click Install. The Power BI Desktop installation package will run and will complete the installation in a few seconds. You may have to confirm that you allow this application to make changes to your system. You will see a progress dialog, as shown in Figure 1-8.



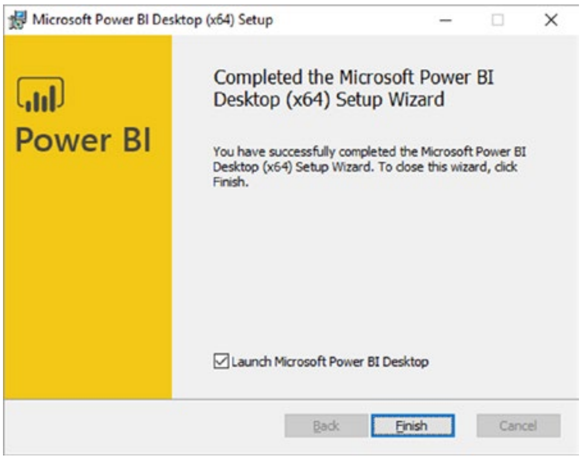
Installing Microsoft Power BI Desktop (x64)

Please wait while the Setup Wizard installs Microsoft Power BI Desktop (x64).

Status: Copying new files

Figure 1-8. *The installation progress dialog*

10. Once the installation process has finished successfully, you will see the completion dialog, as shown in Figure 1-9.



Completed the Microsoft Power BI Desktop (x64) Setup Wizard

You have successfully completed the Microsoft Power BI Desktop (x64) Setup Wizard. To close this wizard, click Finish.

☑ Launch Microsoft Power BI Desktop

Figure 1-9. *The final Power BI Desktop installation dialog*

11. If you want to run Power BI Desktop immediately, then leave the Launch Microsoft Power BI Desktop check box ticked; otherwise, uncheck it and click Finish. The dialog will close and Power BI Desktop is now installed on your computer.

Removing Power BI Desktop

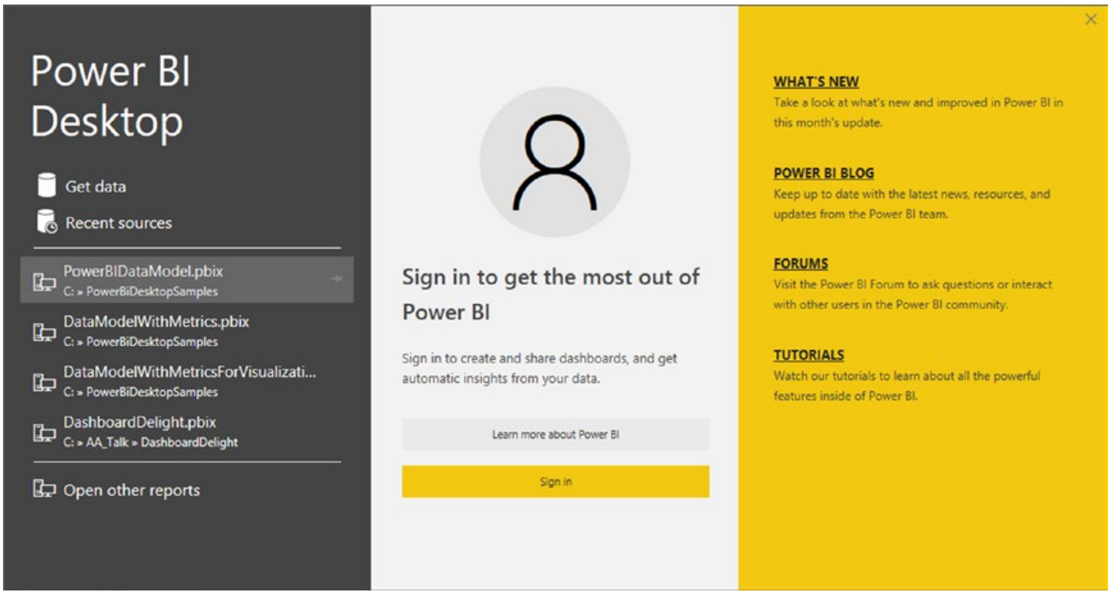
Should you ever want to remove Power BI Desktop from a computer where it has been installed, you have a couple of choices:

- Run the web-based installation process as described earlier. At step 4, you see a dialog asking you if you want to repair or remove Power BI Desktop from your computer. Click Remove and follow the process that is indicated to delete the application from your machine.
- Open the Windows Control Panel. In the Programs section, click Uninstall a program. Select Power BI Desktop from the list of currently installed programs to uninstall it.

Running Power BI Desktop

Once you have installed Power BI Desktop successfully, you are ready to start creating dashboards and analyzing your data. You can start your Power BI Desktop experience as follows:

1. Click the Power BI Desktop icon that was created on the Desktop as part of the installation process. You will see the Power BI Desktop splash screen, as shown in Figure 1-10.


Power BI Desktop
Get data
Recent sources
PowerBIDataModel.pbix
C: » PowerBiDesktopSamples
DataModelWithMetrics.pbix
C: » PowerBiDesktopSamples
DataModelWithMetricsForVisualizati...
C: » PowerBiDesktopSamples
DashboardDelight.pbix
C: » AA_Talk » DashboardDelight
Open other reports
Learn more about Power BI
Sign in
WHAT'S NEW
POWER BI BLOG
FORUMS
TUTORIALS

Sign in to get the most out of Power BI

Sign in to create and share dashboards, and get automatic insights from your data.

Take a look at what's new and improved in Power BI in this month's update.

Keep up to date with the latest news, resources, and updates from the Power BI team.

Visit the Power BI Forum to ask questions or interact with other users in the Power BI community.

Watch our tutorials to learn about all the powerful features inside of Power BI.

Figure 1-10. *The Power BI Desktop splash screen*

For the moment, however, it is time to stop and draw breath. You have successfully installed Power BI Desktop and you are ready to create your first dashboard with this exciting and revolutionary application.

A First Power BI Desktop Dashboard

This book takes you through an immense amount of detail that explains how to import, cleanse, and shape data from a multitude of different sources. You then learn how to carry out a variety of calculations that will help you to tease out the meaning from the data that you are analyzing. Finally, you see how to transform this analysis into telling visuals that make your insights intuitively comprehensible to your audience.

Yet before delving into all of this detail, it is perhaps more important to appreciate the really fundamental qualities of this amazing application. Despite the depth and reach of this piece of software, there are other qualities that make it stand out and that are possibly even more fundamental, including

- *Simplicity*: Anyone can learn to create stunning visualizations and carry out in-depth analysis of data without having to endure a steep or arduous learning curve.
- *Power*: Data from virtually any source can be loaded, manipulated, and combined with other data elements, and then modeled and extended without needing advanced knowledge of IT systems or data management.

Consequently, it is important to see just how easy it is to use the Power BI Desktop dashboard. Indeed, the fastest way to get you "hooked" on this particular tool is to let you see for yourself how fast you can go from zero to hero in delivering compelling dashboards. So let's see just how quick and easy it can be to take a data source (an Excel file in this instance) and transform it into a Power BI Desktop dashboard.

Loading the Source Data

Once you have launched Power BI Desktop, you are faced with the startup screen that you saw in Figure 1-10. Given that you are working with an application that lives and breathes data, it is not really surprising that the first step in a new analytical challenge is to find and load some data. So the following explains what you have to do (assuming that you have downloaded the sample data that accompanies this book from the Apress web site-this is explained in Appendix A).

1. Click Get Data in the startup screen. The Get Data dialog will appear, as shown in Figure 1-11.

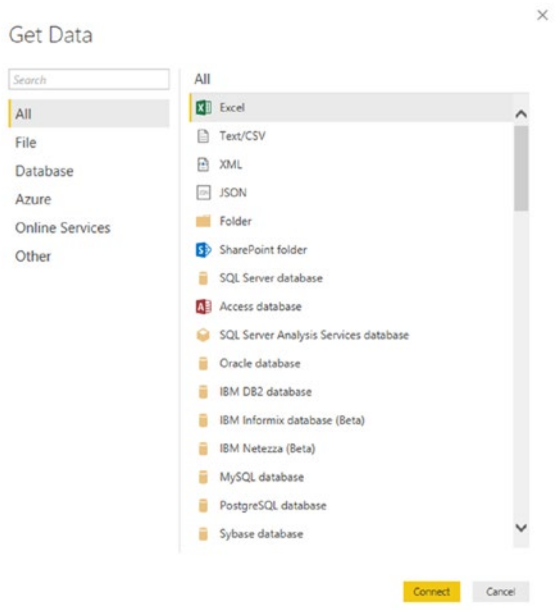


Figure 1-11. *The Get Data dialog*

2. In the list of all the possible data sources on the right of this dialog, click Excel, and then click Connect. The Windows Open File dialog will appear.
3. Click the file C:\PowerBiDesktopSamples\CH01\CarSales.Xlsx. The Windows Open dialog will look like the one in Figure 1-12.

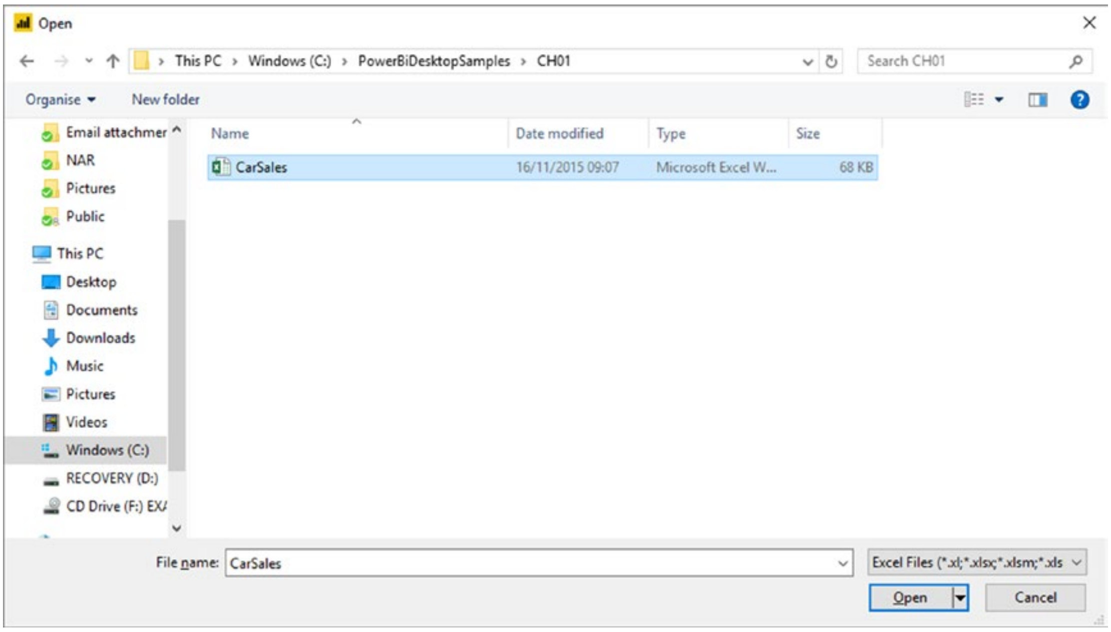
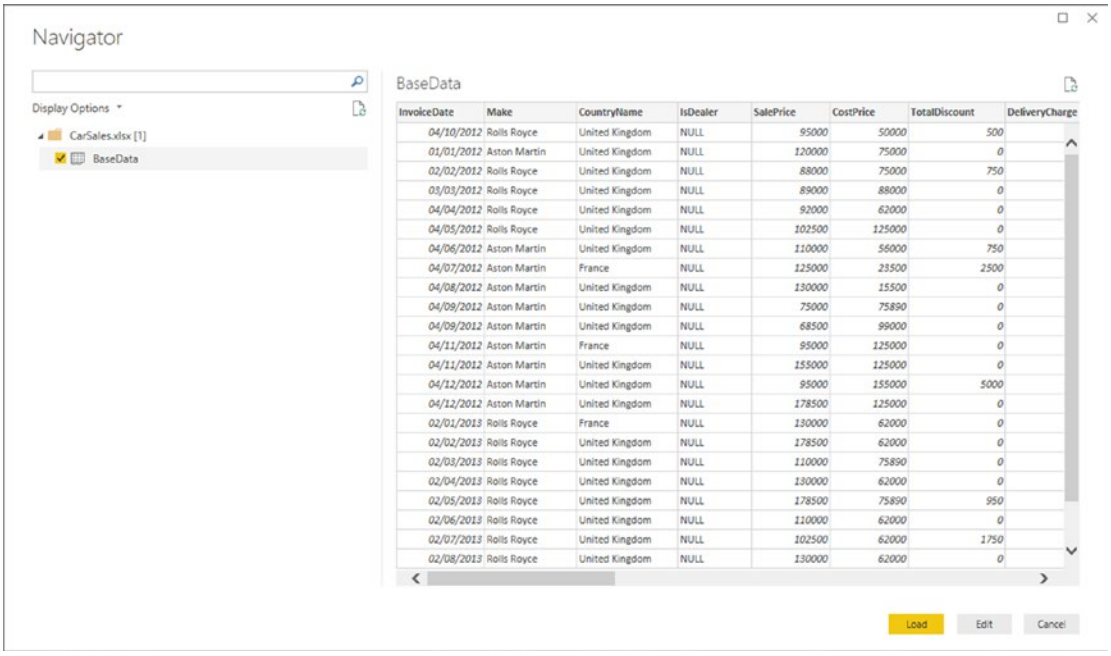


Figure 1-12. *The Windows Open File dialog when loading data from a file source*

4. Click the Open button. The Connecting dialog will appear for a second or two and then the Navigator dialog will appear.

5. You will see that the CarSales.xlsx file appears on the left of the Navigator dialog and that any workbooks, named ranges, or data tables that it contains are also listed. Click the BaseData worksheet name that is on the left. The contents of this workbook will appear in the data pane on the right of the Navigator dialog.

6. Click the check box for the BaseData worksheet on the left. The Load and Edit buttons will be activated. The Navigator dialog should look like Figure 1-13.



BaseData

InvoiceDate	Make	CountryName	IsDealer	SalePrice	CostPrice	TotalDiscount	DeliveryCharge
04/10/2012	Rolls Royce	United Kingdom	NULL	95000	50000	500	
01/01/2012	Aston Martin	United Kingdom	NULL	120000	75000	0	
02/02/2012	Rolls Royce	United Kingdom	NULL	88000	75000	750	
03/03/2012	Rolls Royce	United Kingdom	NULL	89000	88000	0	
04/04/2012	Rolls Royce	United Kingdom	NULL	92000	62000	0	
04/05/2012	Rolls Royce	United Kingdom	NULL	102500	125000	0	
04/06/2012	Aston Martin	United Kingdom	NULL	110000	56000	750	
04/07/2012	Aston Martin	France	NULL	125000	23500	2500	
04/08/2012	Aston Martin	United Kingdom	NULL	130000	15500	0	
04/09/2012	Aston Martin	United Kingdom	NULL	75000	75890	0	
04/09/2012	Aston Martin	United Kingdom	NULL	68500	99000	0	
04/11/2012	Aston Martin	France	NULL	95000	125000	0	
04/11/2012	Aston Martin	United Kingdom	NULL	155000	125000	0	
04/12/2012	Aston Martin	United Kingdom	NULL	95000	155000	5000	
04/12/2012	Aston Martin	United Kingdom	NULL	178500	125000	0	
02/01/2013	Rolls Royce	France	NULL	130000	62000	0	
02/02/2013	Rolls Royce	United Kingdom	NULL	178500	62000	0	
02/03/2013	Rolls Royce	United Kingdom	NULL	110000	75890	0	
02/04/2013	Rolls Royce	United Kingdom	NULL	130000	62000	0	
02/05/2013	Rolls Royce	United Kingdom	NULL	178500	75890	950	
02/06/2013	Rolls Royce	United Kingdom	NULL	110000	62000	0	
02/07/2013	Rolls Royce	United Kingdom	NULL	102500	62000	1750	
02/08/2013	Rolls Royce	United Kingdom	NULL	130000	62000	0	

Figure 1-13. *The Navigator dialog with data selected*

7. Click Load. The data will be loaded from the Excel file into Power BI Desktop. You will see the Power BI Desktop report window, like the one shown in Figure 1-14.

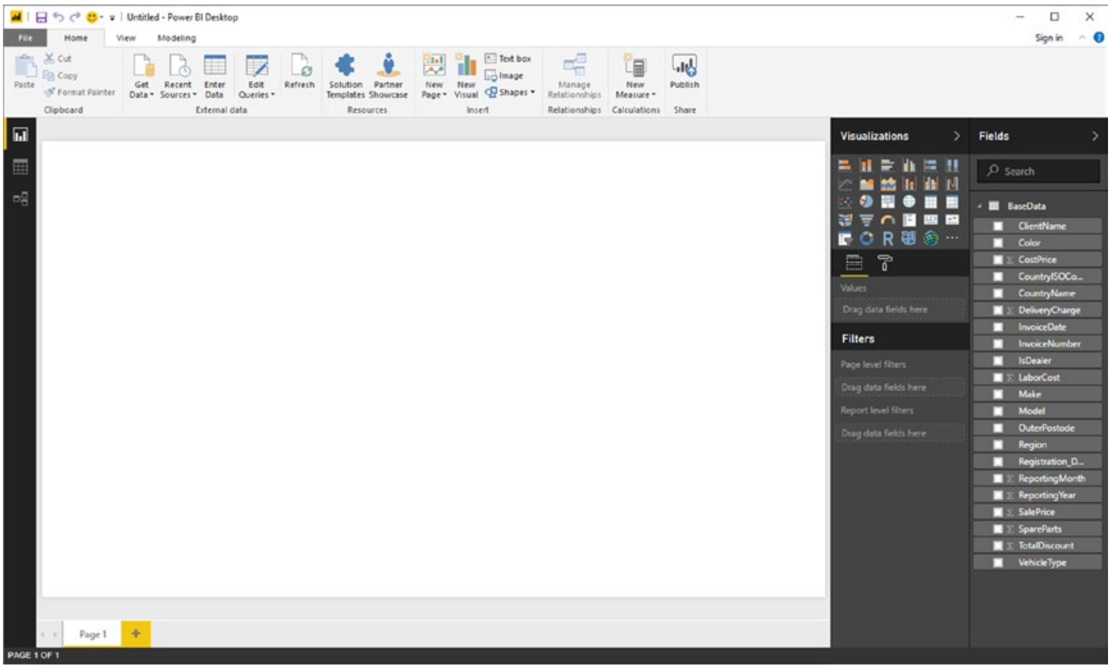


Figure 1-14. *The Power BI Desktop report window*

I imagine that loading this data took under a minute. Yet you now have a fully operational data model in Power BI Desktop that is ready for action. However, before moving forward and creating a dashboard, I would like to pause for an instant and explain exactly what you have seen so far. Of course, if you are itching to race ahead and actually create a couple of tables and charts, then you can always jump ahead to the "Your First Visualizations" section.

The Data Load Process

What you have seen so far is an extremely rapid dash through a Power BI Desktop data load scenario. In reality, this process can range from the blindingly simple (as you just saw) to the more complex where you join, filter, and modify multiple datasets from different sources (as you will discover in Chapters 2 through 9). However, loading data will always be the first step in any data analysis scenario when you are using Power BI Desktop.

In this short example, you nonetheless saw many of the key elements of the data load process. These included

- Accessing data that is available in any of the source formats that Power BI Desktop can read
- Taking a first look at the data before loading it into Power BI Desktop

What you did not see here is how Power BI Desktop can add an intermediate step to the data load process and edit the source data in Power BI Desktop Query Editor. This aspect of data manipulation is covered extensively in the following few chapters.

The Navigator Window

One key aspect of the data load process is using the Navigator window correctly. The Navigator window appears when connecting to many, but not all, data sources. It is there to let you

- Take a quick look at the available data tables in the data source
- Filter multiple data elements that are available in a single data source
- Look at the data in individual tables
- Select one or more data tables to load into Power BI Desktop

Depending on the data source to which you have connected, you might see only a few data tables in the Navigator window, or hundreds of them. In any case, what you can see are the structured datasets that Power BI Desktop can recognize and is confident that it can import. Equally dependent on the data source is the level of complexity of what you will see in the Navigator window. If you are looking at a database server, for instance, then you may start out with a list of databases and you may need to dig deeper into the arborescence of the data by expanding databases to list the available data tables and views.

You will see much, much more of the Navigator window in the following three chapters.

The Navigator Data Preview

The Navigator Data Preview pane (on the right) is, as its name implies, a preview of the data in a data source. It provides

- A brief overview of the top few records in any of the datasets that you want to look at. Given that the data you are previewing could be hundreds of columns wide and millions of rows long, there could be scroll bars for the data table visible inside the Navigator Data Preview.
- A list of the available columns in the data table. These are shown at the bottom of the Navigator Data Preview.

Power BI Desktop can preview and load data from several different sources. Indeed (as you can see from the list of possible data sources in the Get Data dialog in Figure 1-11), it can read most of the commonly available enterprise data sources as well as many, many others. What is important to appreciate is that Power BI Desktop applies a common interface to the art and science of loading data, whatever the source. So whether you are examining a SQL Server or an Oracle database, an XML file or a text file, a web page or a big data source, you will always be using a standardized approach to examining and loading the data. This makes the Power BI Desktop data experience infinitely simpler—and extremely reassuring. It means that you spend less time worrying about technical aspects of data sources and you are free to focus on the data itself.

■ **Note** The Navigator Data Preview is a brilliant data discovery tool. Without having to load any data, you can take a quick look at the data source and any data that it contains that can probably be loaded by Power BI Desktop. You can then decide if it is worth loading, so that you do not waste time on a data load that could be superfluous to your needs.

Modifying Data

Once you have one or more queries in Power BI Desktop that can connect to data sources and bring the data into this environment, you can start thinking about the next step—transforming the data so that it is ready for use. Depending on the number of data sources that you are handling and the extent of any modifications that are required, this could vary from the simple to the complex. To give a process some structure, I advise that you try to break down any steps into the following main threads:

- *Shape the dataset*: This covers filtering out records to reduce the size of the dataset, as well as removing any extraneous columns. It may also involve adding columns that you create by splitting existing columns, creating calculated columns, or even joining queries.
- *Cleanse and modify the data*: This is also known as *data transformation* (the *T* in ETL). It encompasses the process of converting text data to uppercase and lowercase, as well as (for instance) removing nonprinting characters. Rounding numbers and extracting date parts from date data are also possible (among many other eventual transformations).

For the moment, however, it is only important to understand that Power BI Desktop can do all of this if you need it to. Transforming data is explained in detail in Chapters 6 through 8.

The Power BI Desktop Window

Before we go any further, I would like to explain the Power BI Desktop window, since it is something that you will use a lot in this chapter from this point onward. The Power BI Desktop window contains the elements that are outlined in Figure 1-15.

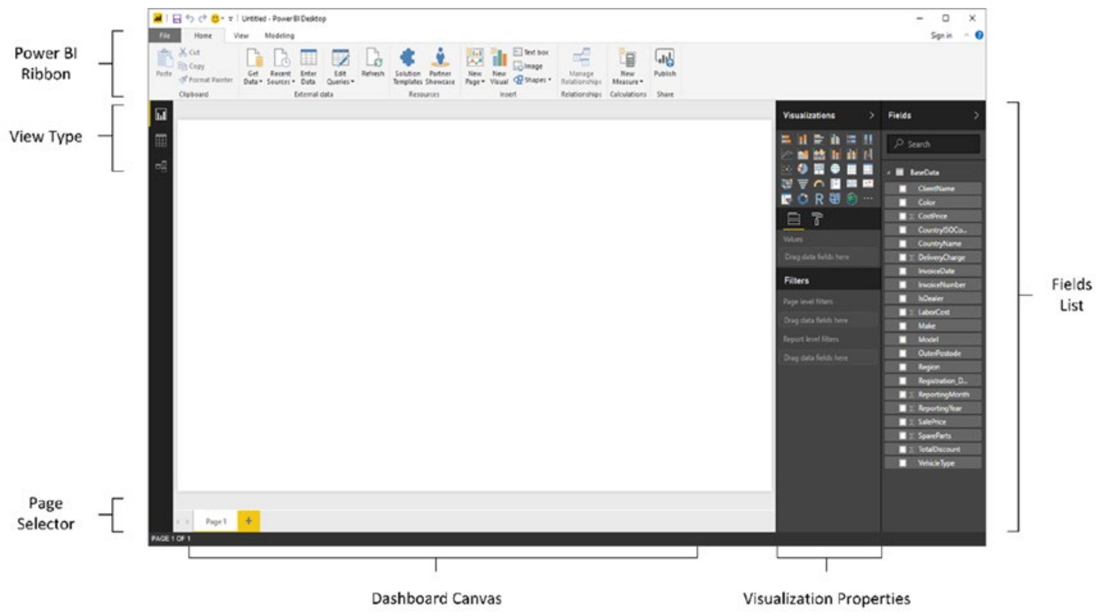


Figure 1-15. *The Power BI Desktop*

As you can see, the Power BI Desktop screen is simple and uncluttered. The various elements that it contains are explained in Table 1-1.

Table 1-1. *Power BI Desktop Options*

Option	Description
Power BI Ribbon	This contains the principal options that are available to you when developing dashboards with Power BI Desktop.
View Type	These three icons let you flip between Dashboard view (where you create dashboards and reports), Data view (where you add calculations), and Relationships view (where you join data from different sources).
Dashboard Canvas	This is the main area, where you add visualizations and design your dashboards.
Visualization pane	This area of the application is specific to each type of visualization and lets you set the specific attributes of each element on a dashboard. It also allows you to filter dashboards, pages, and individual visualizations. You can also format visualizations using this pane.
Fields List	Here you can see all the available fields from the source data that you can use to build your visualizations.
Visualization Palette	This area contains all the currently available types of visualization that you can add to a dashboard.
Page Selector	These are tabs that let you switch from page to page in a report.

Power BI Desktop—like most Microsoft applications—has several available ribbons. These are explained in the course of this book.

Your First Visualizations

With your data safely in place inside Power BI Desktop, you can now begin to create the tables, charts, maps, and other elements that you want to add to a dashboard, which you can use to present your first insights into Brilliant British Cars. As this is a first "taster" exercise, I am not looking at explaining all that can be done using Power BI Desktop. All I want to do is to show you how easy it is to create dashboards in minutes rather than hours. Indeed, I only hope that this first simple dashboard will leave you hungry to learn more—and so to move on to the rest of this book.

Before creating a few simple visualizations, let me clarify some of the terms that you will meet when working with Power BI Desktop:

- *Visualizations*: Also known as *visuals*, these are the individual presentation elements that you create based on the underlying data. A visual can be a table, a chart, a gauge, a map, or many things indeed. I use these terms interchangeably in this book.
- *Dashboard*: A Power BI Desktop dashboard is a collection of visualizations on a single page. Indeed, I tend to use the terms *page* and *dashboard* interchangeably.
- *Report*: This is a collection of pages (or dashboards) in a single file, all using the same dataset.

Displaying Available Fields

One of the first things to do is make sure that you can see the data that you will be working with in dashboards and reports. If you look at the right of the Power BI Desktop Report view, you see a vertical pane with the label *Fields* at the top. This is the Fields pane (or Fields List). It is from here that you access all the data that you will use in your visualizations and dashboards.

For the moment, however, all that you can see is probably the name of the dataset that you imported previously—the BaseData dataset from Excel. Do the following to see all the fields that this table contains:

1. Click the small triangle to the left of the table name. The table will expand to reveal all the available fields that it contains. Alternatively, if the fields are already visible, they will disappear from view, leaving only the data table name visible. You can see what this looks like in Figure 1-16.

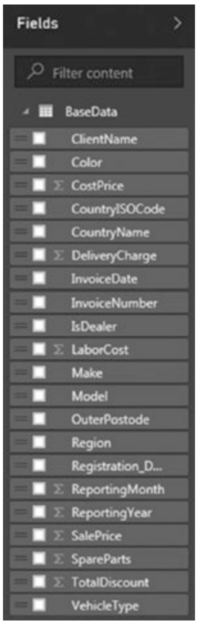


Figure 1-16. *The Power BI Desktop Fields list*

You can see that some of the fields have a sigma (∑) icon to their left. This indicates that the data in the field is numeric. As you progress through this book, you will see that there are other icons that Power BI Desktop uses to flag different types of fields.

Adding a Matrix of Sales per Country by Year

It is now time to draw on the blank canvas that is your first dashboard. To begin, let's start with a simple matrix of sales per country for each year that Brilliant British Cars has been trading.

1. In the Visualizations pane, click the matrix icon, as highlighted in Figure 1-17. A blank matrix will appear on the dashboard canvas.

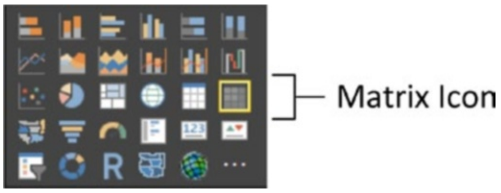


Figure 1-17. *The matrix icon in the Visualizations pane*

2. Leaving the freshly created matrix selected, click the check box to the left of the CountryName field in the Fields list. The list of countries where cars have been sold will appear as the left-hand column of the matrix.

3. Drag the ReportingYear field into the Visualizations pane over the Columns fields area (this is called the *field well*). Figure 1-18 shows how to do this. This adds the model years as column headers in the matrix.

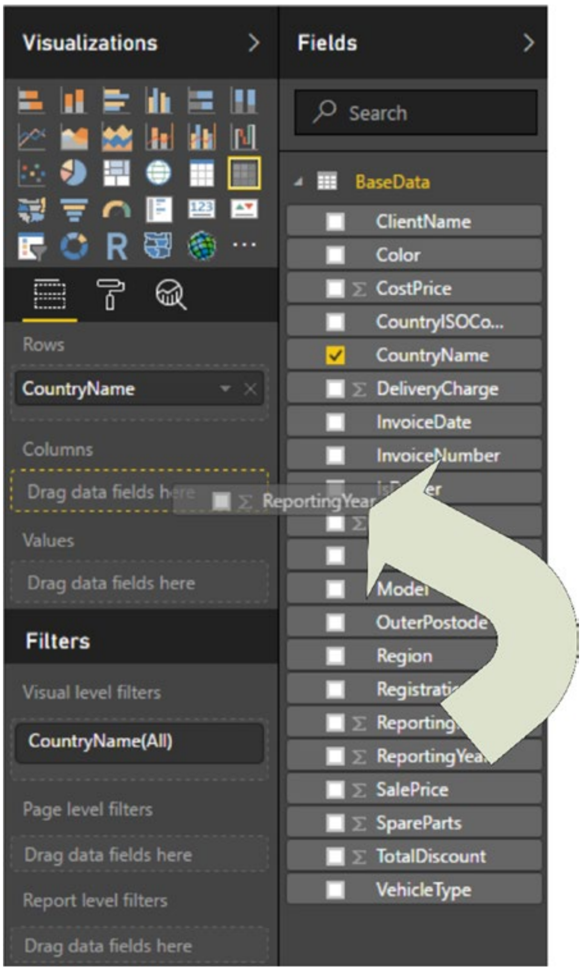
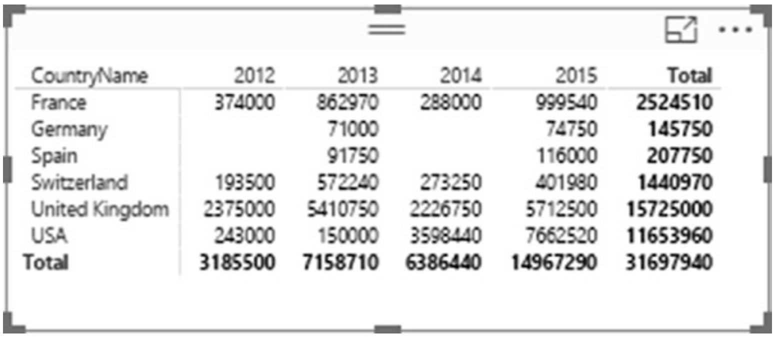


Figure 1-18. *Adding the Columns fields to a matrix*

4. Leaving the matrix selected, click the check box to the left of the SalePrice field in the Fields list. The aggregated sale price for all vehicles sold by country and by year will appear in the matrix.

5. Drag the corner handle of the matrix to resize it so that there is no spare whitespace inside the matrix itself. It will look like Figure 1-19.



CountryName	2012	2013	2014	2015	Total
France	374000	862970	288000	999540	**2524510**
Germany		71000		74750	**145750**
Spain		91750		116000	**207750**
Switzerland	193500	572240	273250	401980	**1440970**
United Kingdom	2375000	5410750	2226750	5712500	**15725000**
USA	243000	150000	3598440	7662520	**11653960**
Total	**3185500**	**7158710**	**6386440**	**14967290**	**31697940**

Figure 1-19. *A matrix of sales per country*

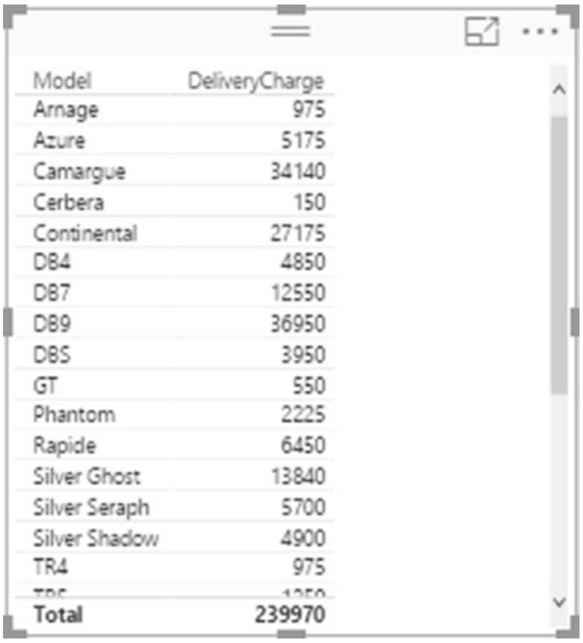
It would be hard to make this any simpler. Within seconds, you have created a matrix of sales by year and country and the totals have been added automatically. Of course, there are many ways of extending and developing a matrix in Power BI Desktop—and you can discover them all in Chapter 15—but for now, it is time to press on and add a chart to your fledgling dashboard.

■ **Note** In this short exercise, you saw that you can both select fields from the Fields list or drag them to the field well to add them to a selected visual. An alternative is to drag a field from the Fields list onto the visualization itself.

Adding a Column Chart of Delivery Charge by Model

Now that you have seen how easy it is to create a matrix in Power BI Desktop, the time has come to add some visual impact to your analysis. This time, you will use the available data to display the total delivery charge for each model of car sold.

1. Click an empty area of the dashboard canvas to unselect any visualizations.
2. Drag the Model field onto an empty area of the dashboard canvas. Power BI Desktop automatically creates a table displaying all the vehicle models sold.
3. Drag the DeliveryCharge field from the Fields pane onto the table that you just created. Power BI Desktop will calculate the total DeliveryCharge for each available make. The table will look like Figure 1-20.



Model	DeliveryCharge
Arnage	975
Azure	5175
Camargue	34140
Cerbera	150
Continental	27175
DB4	4850
DB7	12550
DB9	36950
DBS	3950
GT	550
Phantom	2225
Rapide	6450
Silver Ghost	13840
Silver Seraph	5700
Silver Shadow	4900
TR4	975
Total	**239970**

Figure 1-20. *A table of aggregated delivery charge per make*

4. Leaving the table selected, click the clustered column chart icon in the Visualizations pane. This is the second icon on the left on the upper row of the selection of visualizations. Power BI Desktop will switch the table to a chart.

5. Drag the corner handle of the chart to resize it so that all the makes are visible on the bottom axis. The chart will look like Figure 1-21.

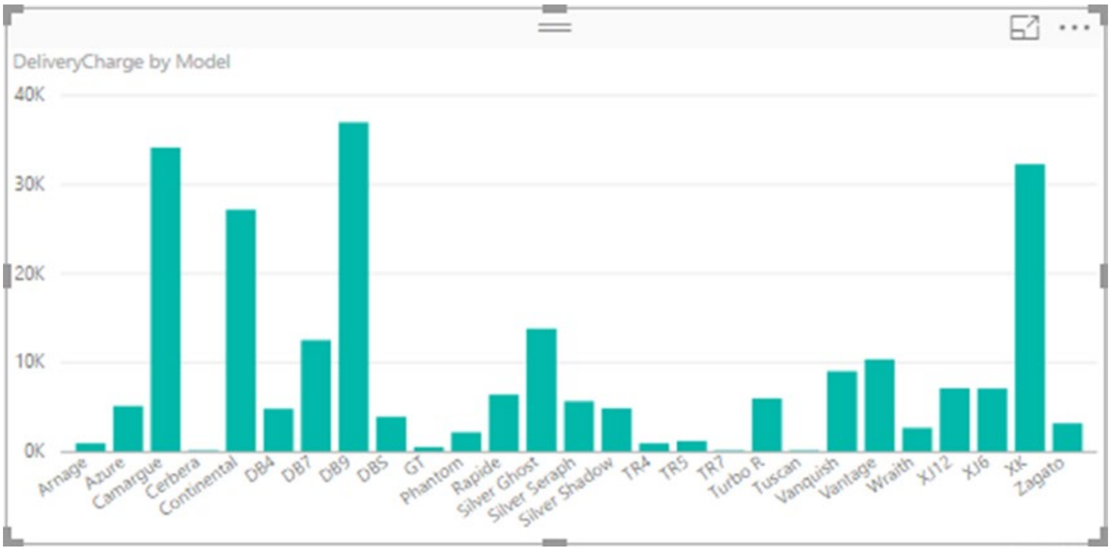


Figure 1-21. *A column chart of delivery charge by model*

Equally simple, don't you think? Yet, believe me, the fun has only just begun. While you will see lots more about how to create and enhance charts in Chapters 16 and 17, you can always try a few basic tweaks now. For instance, if you select the chart and then click any of the other charting icons in the Visualizations pane, you can change the type of chart instantaneously. Moreover, if you are not sure which icon does what, then all you have to do is hover the mouse pointer over an icon in the Visualizations pane to display a tooltip that will guide you further.

Adding a Map of Labor Cost by Country

Tables and charts are all very well, but nothing beats a good picture when it comes to illustrating a point or highlighting an insight. So, as we have a dataset that includes information for a range of countries, why not display some of our analysis as a map?

1. Click any empty part of the dashboard canvas to unselect any visualizations.
2. Click the filled map icon in the Visualizations pane. You can see this icon in Figure 1-22.

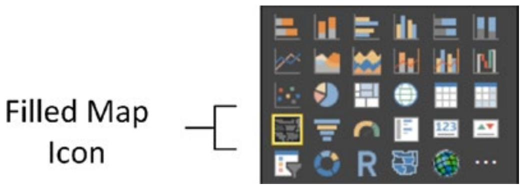


Figure 1-22. *Adding a filled map visualization to a dashboard*

3. Leaving the empty map visualization selected, click the check box to the left of the CountryName field in the Fields list. This will display a map of the world.
4. Leaving the map selected, drag the LaborCost field onto the map. This will highlight any countries where there are labor costs relating to vehicles sold.
5. Drag the colored European countries to the center of the map.
6. Using the mouse wheel, zoom in to the colored European countries. The finished map will look like Figure 1-23.

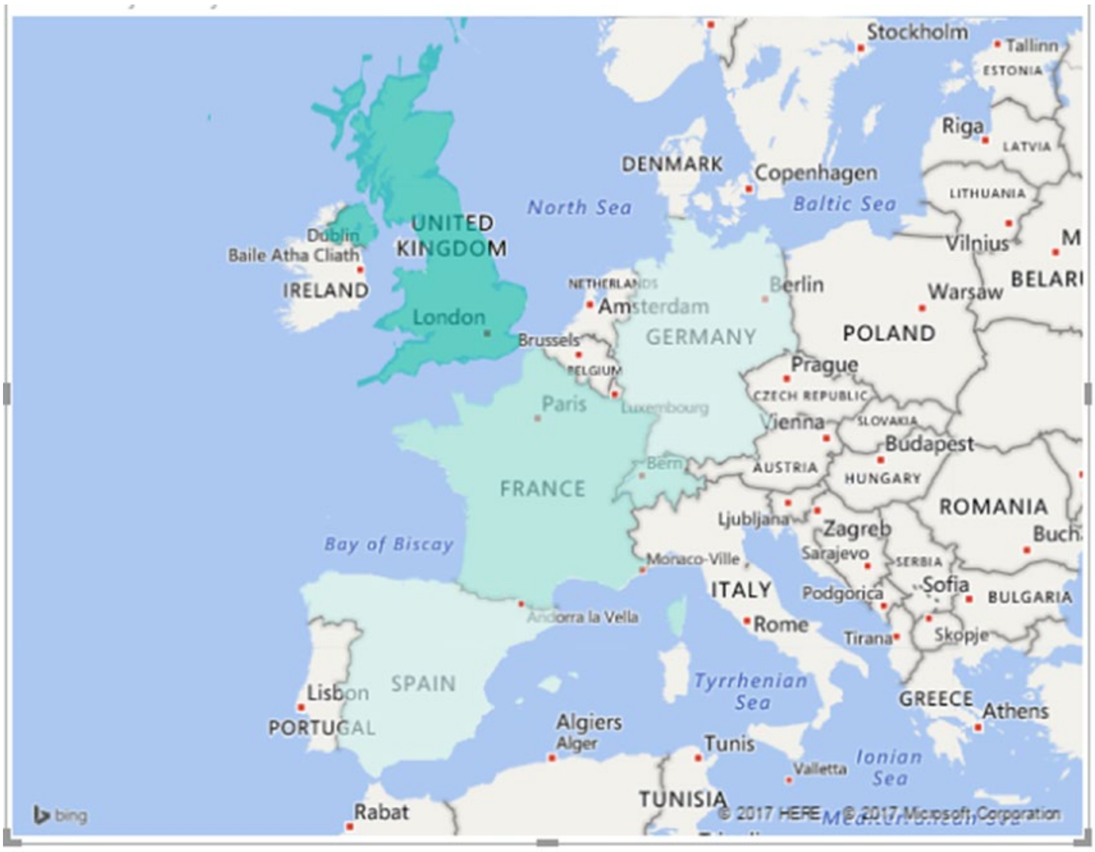


Figure 1-23. *A map of labor cost by country*

This time, and in only a few clicks, you have used your data to create a map that clearly illustrates the geography of your sales. Once again, this is only a rapid overview of all that Power BI Desktop can do when it comes to displaying mapping data. You will learn more about creating and modifying maps in Power BI Desktop in Chapter 19.

Adding a Card Showing the Total Cost of Spare Parts

Sometimes you do not want to show a large amount of data but quite the opposite. You want to highlight a single figure to give it prominence on the dashboard. Power BI Desktop has a really effective way of doing just this. It consists of adding visualizations called *cards*, which are what you will now add to your dashboard.

1. Click the dashboard canvas to unselect any visualizations.
2. Click the card icon in the Visualizations pane, as shown in Figure 1-24.

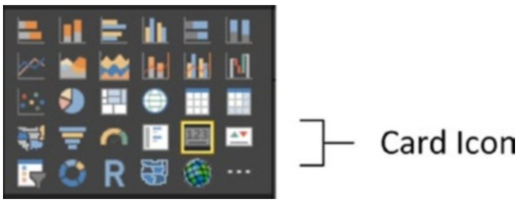


Figure 1-24. *Adding a card visualization to a dashboard*

3. Leaving the (slightly clunky) empty card visualization selected, click the check box to the left of the SpareParts field in the Fields list. This displays the spare parts total in the source data.

4. Drag the corner handle of the matrix to resize it so that there is no spare whitespace inside the matrix itself. It will look like Figure 1-25.

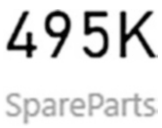


Figure 1-25. *A card showing the total cost of spare parts*

That is all you have to do. Three or four clicks and you have a clear visualization of a key metric for your audience. This is not the only way that you can create this particular visualization, but you have to wait for Chapter 15 to get all the details on adding cards to Power BI Desktop dashboards.

Adding a Slicer by Make

As a final tweak, you will add some interactivity to the dashboard that you are building. You will add a slicer (an interactive selection tool) that will let you—or any user of this dashboard—filter by any or all car models sold. Here is how you can do this:

1. Drag the Make field to a blank area on the dashboard canvas. Power BI Desktop will create a list of vehicle models.

2. Click the slicer icon in the Visualizations pane, as shown in Figure 1-26.

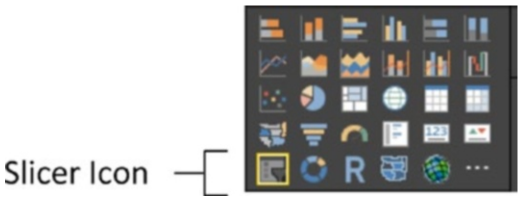


Figure 1-26. *Adding a slicer to a dashboard*

3. Drag the corner handle of the slicer to resize it so that there is no spare whitespace inside the slicer. It will look like Figure 1-27.

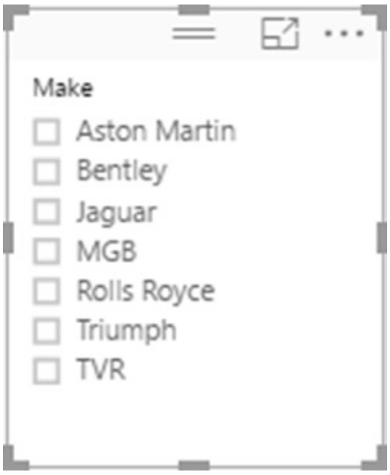


Figure 1-27. *A slicer on the model of vehicle*

You can now test the slicer by selecting—or deselecting—any car model that is listed in the slicer. The other visualizations on the dashboard will instantly be updated to reflect the choice of models. You will soon get a first look at how this slicer can be used to filter data.

Arranging the Dashboard

Now that you have created a few visuals, it is time to coordinate them on the page so that you can deliver a meaningful dashboard that adds power to your insights.

Moving a Visualization

Moving a visualization is impressively easy:

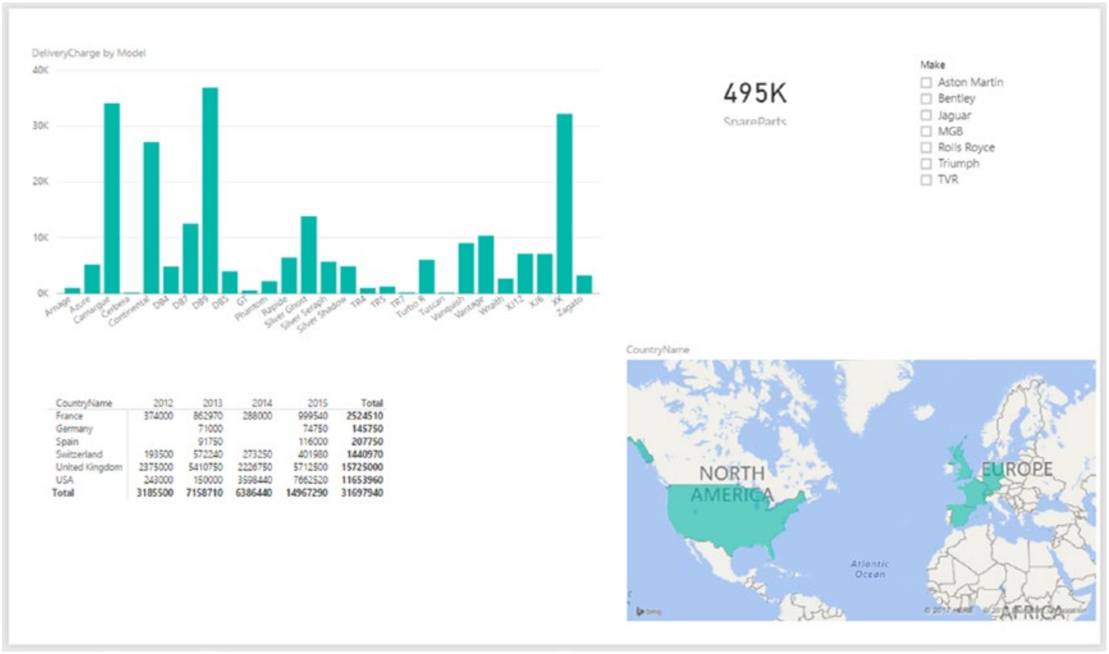
1. Click the visualization that you want to move.
2. Drag the visualization elsewhere on the dashboard canvas.

Resizing a Visualization

Resizing a visualization is also extremely easy:

1. Click the visualization that you want to resize.
2. Move the mouse pointer over any of the corner or side handles of the visualization. The pointer will become a double-headed arrow.
3. Drag the edge of the visualization to increase or decrease its current size.

After a little effort, your dashboard could look like the one in Figure 1-28.



CountryName	2012	2013	2014	2015	Total
France	374000	862970	288000	999540	**2524510**
Germany		71000		74750	**145750**
Spain		91750		116000	**207750**
Switzerland	193500	572240	273250	401980	**1440970**
United Kingdom	2375000	5410750	2226750	5712500	**15725000**
USA	243000	150000	3598440	7662520	**11653960**
Total	**3185500**	**7158710**	**6386440**	**14967290**	**31697940**

Figure 1-28. *The final version of your first dashboard*

So here you have your first Power BI dashboard. I have to admit that this first stab at self-service business intelligence was not concentrating overmuch on the aesthetics of the output. I preferred to let you appreciate the speed and simplicity with which you have created an entire dashboard, from scratch and with no prior knowledge of the tool that you have used.

How long did it take you to build this dashboard, do you think? Fifteen minutes? Thirty minutes? Indeed, however long it took, you have also learned the basics of dashboarding with Power BI Desktop. You can now build on this knowledge as you progress through this book.

In fact, extending a dashboard by adding further visualizations is so intuitive that it is too easy to miss the salient points of what you have seen so far. So, to resume, what you have just learned is that

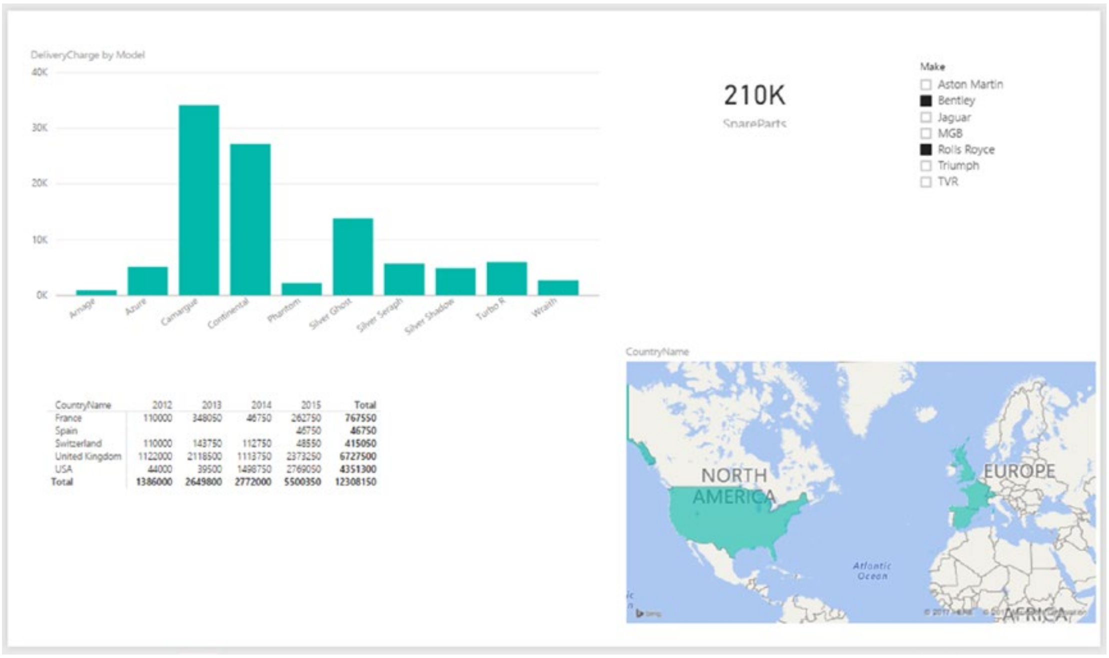
- You can place any visualization anywhere on the dashboard canvas.
- You can resize an element quickly and easily.
- You can convert any type of visualization to any other type in a single click.

Interactivity in Dashboards

Building a dashboard was only the start, as far as Power BI Desktop is concerned. For a Power BI dashboard is never set in stone. In fact, quite the opposite is true, because every dashboard that you create is instantly and intuitively interactive. This means that you can use it to highlight salient points and drill down to expose the key insights that your analysis has led you to.

Even a simple dashboard like the one that you just created is immediately interactive. As an example, suppose that you want to use this dashboard to display data for only a couple of the makes that the company has sold. The following explains how to do it:

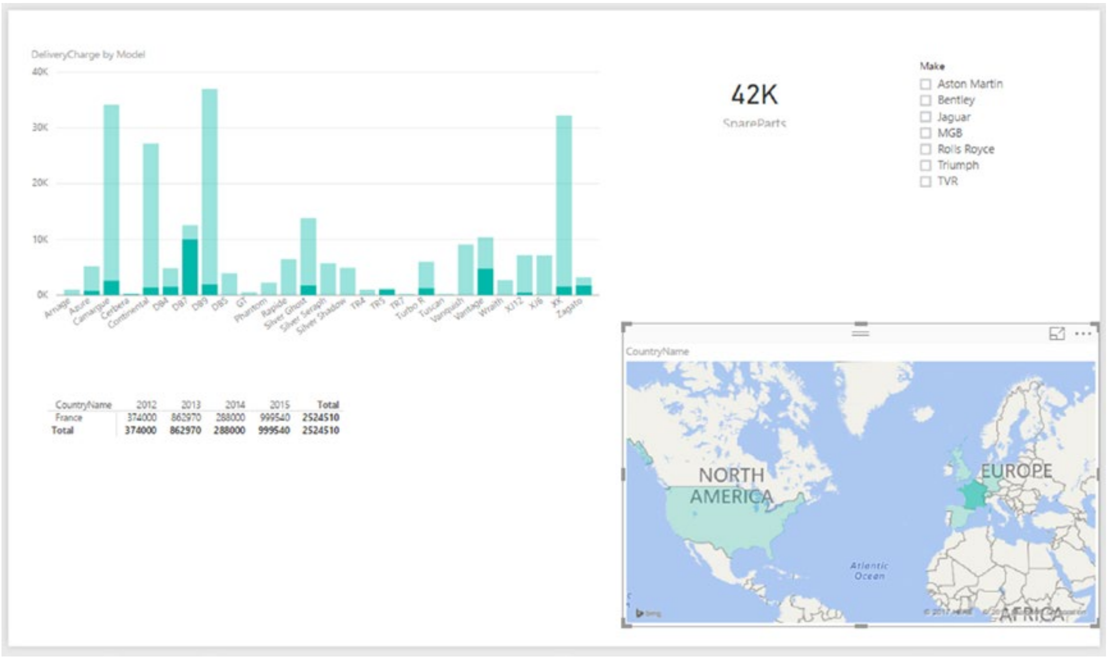
1. In the slicer (on the top right of the dashboard), Ctrl-click Bentley and Rolls-Royce. The dashboard will instantly update to show only data for these car models, as shown in Figure 1-29.



CountryName	2012	2013	2014	2015	Total
France	110000	348050	46750	262750	**767550**
Spain				46750	**46750**
Switzerland	110000	143750	112750	48550	**415050**
United Kingdom	1122000	2118500	1113750	2373250	**6727500**
USA	44000	39500	1498750	2769050	**4351300**
Total	**1386000**	**2649800**	**2772000**	**5500350**	**12308150**

Figure 1-29. *Interactively filtering a dashboard using a slicer*

2. In the slicer, click any element twice to clear the filter.
3. In the map, click France. The dashboard will instantly update to show only data for the selected country, as shown in Figure 1-30.



CountryName	2012	2013	2014	2015	Total
France	374000	862970	288000	999540	2524510
Total	374000	862970	288000	999540	2524510

Figure 1-30. *Interactively filtering a dashboard using a map*

Now it is your turn. Rather than explain here all that you can do to filter and view your data, I suggest that you try clicking parts of the map or the column chart and see what happens! In any case, all the detail concerning filters and slicers is explained in Chapters 20 and 21.

Formatting Reports

Power BI Desktop allows you to create reports with multiple visuals in record time. Out of the box you can produce stylish levels of presentation in a few clicks. Yet with a little effort, you can enhance the standard Power BI Desktop formatting and truly leave your audience impressed with your presentation skills.

This section will introduce you to a few of the techniques that you can apply to transform the look and feel of your reports. As was the case with the previous sections in this chapter, the aim here is not to give you an exhaustive tour of all the formatting possibilities that Power BI Desktop has to offer (Chapters 14 through 22 will do this). Instead, I want only to show you how easily and intuitively you can take the allure of your reporting to the next level. Fortunately, Power BI Desktop lets you format nearly all visuals in similar ways. So you only have to learn a few basic techniques to enhance all your reports in record time.

If you want to try formatting a report, but have not built the report that you have seen so far in this chapter, you can always load the Power BI Desktop file named C:\PowerBiDesktopSamples\CH01\CH01Example.pbix-assuming, of course, that you have downloaded the sample data as described in Appendix A.

The Format Pane

All formatting is defined in the Format Pane. You switch to the Format Pane as follows:

1. Select an existing visual.
2. Click the "paint roller" icon in the Visualizations pane. This is the Format icon. You can see this in Figure 1-31.

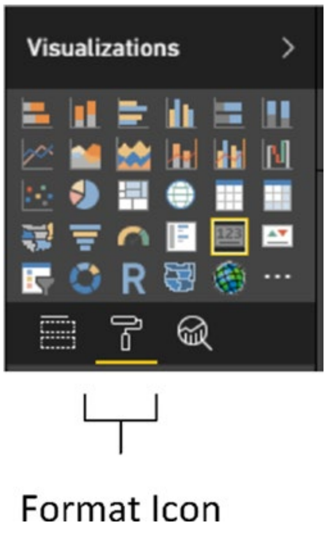


Figure 1-31. *The Format icon*

Clicking the Format icon will display the Format Pane containing the formatting options that are available for the type of visual that you have selected. The range of available formatting options will vary for each type of visual.

■ **Note** You can only format multiple visuals at the same time if *all* the selected visuals are of the same type.

Borders

Nearly all visuals allow you to add a border. To enhance a visual with a border:

1. Select the visual that you want to format (I will select the card in the report that you have built in this chapter).
2. Click the Format icon.
3. Click the Border button to add a border.
4. Expand the Border section of the Format Pane.
5. Click the color palette and select a color for the border. The Format Pane will look like Figure 1-32.

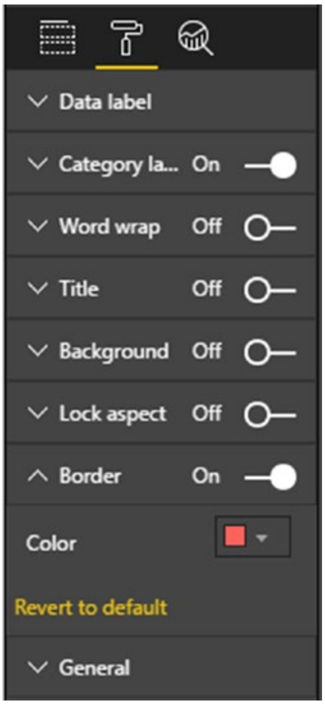

Data label
Category la... On
Word wrap Off
Title Off
Background Off
Lock aspect Off
Border On
Color
Revert to default
General

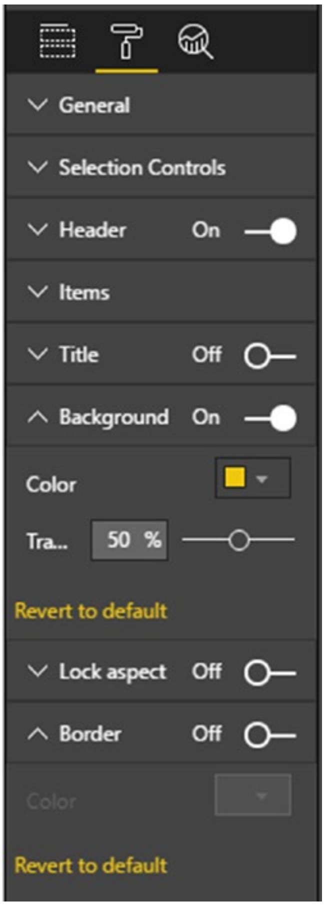
Figure 1-32. *The Format Pane for card visuals*

That is all you have to do. The border is now added to the selected visual. Fortunately, this technique will work in the same way for most Power BI Desktop report visuals, whatever they may be.

Background Color

Some visuals look better with a different background color. I like to distinguish slicers from other visuals, so let's add a different background color to the slicer in the sample report.

1. Select the visual that you want to format (I will select the slicer in the report that you have built in this chapter).
2. Click the Format icon, unless the Format Pane is already displayed.
3. Expand the Background section of the Format Pane.
4. Click the Background button to add a background color. The Format pane will display "Background On" and the background button will now be a filled circle.
5. Click the color palette and select a color for the background color. The Format Pane will look like Figure 1-33.



Header On
Title Off
Background On
Tra... 50 %
Lock aspect Off
Border Off

Figure 1-33. *The Format Pane for slicers*

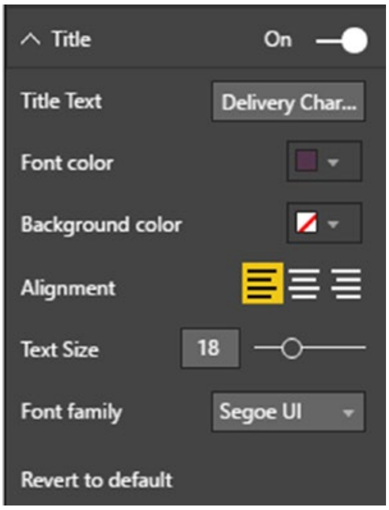
As you can see, there are other background color options. I will explain these in Chapter 15.

Titles

Certain visuals need titles, either to make a point or to differentiate a specific visual. Adding or modifying a title is easy:

1. Select the visual whose title you want to modify (I will select the chart in the report that you have built in this chapter).
2. Click the Format icon, unless the Format Pane is already displayed.
3. Expand the Title section of the Format Pane.
4. Ensure that the Title button is on. This should already be the case for a chart.
5. Change the title text to Delivery Charges.
6. Click the color palette and select a color for the font color.

7. Drag the slider for the Text Size option to the right to increase the font size. The Format Pane will look like Figure 1-34.



Title On
Title Text: Delivery Char...
Font color
Background color
Alignment
Text Size: 18
Font family: Segoe UI
Revert to default

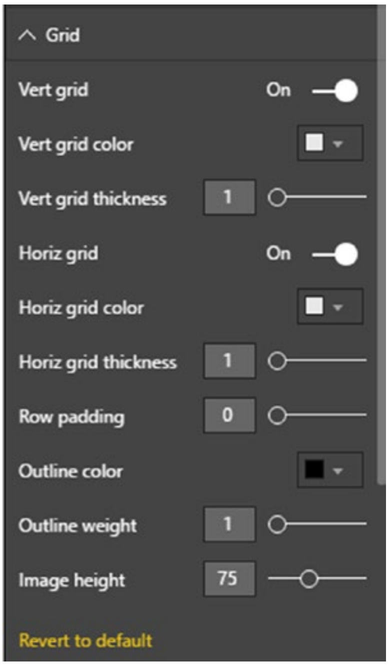
Figure 1-34. *The Format Pane for chart visuals*

As you can see, a chart has many more available formatting options than the other visuals that you saw so far. These options will be explained in more depth in Chapter 17.

Table Gridlines

Tables and matrices can be enhanced with your choice of gridlines. Adding these is as easy as

1. Select the visual that you want to format (I will select the table in the report that you have built in this chapter).
2. Click the Format icon, unless the Format Pane is already displayed.
3. Expand the Grid section of the Format Pane.
4. Switch on the vertical and horizontal grids.
5. Select an Outline Color from the Outline Color palette. The Format Pane will look like Figure 1-35.



^ Grid

Vert grid	On	
Vert grid color	⬜	
Vert grid thickness	1	
Horiz grid	On	
Horiz grid color	⬜	
Horiz grid thickness	1	
Row padding	0	
Outline color	⬛	
Outline weight	1	
Image height	75	

Revert to default

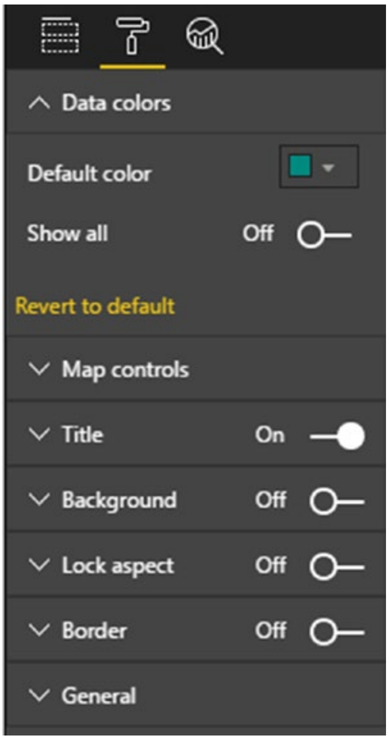
Figure 1-35. *The Format Pane for table visuals*

Yet again, there are many more options that you can adjust to add extremely precise formatting to tables. These techniques are explained in Chapter 14.

Data Colors

As a final quick example of how formatting can be applied to your visuals, suppose that you want to change the color of the data representation in the map that you created. To do this:

1. Select the map that you want to format.
2. Click the Format icon.
3. Expand the Data colors section of the Format Pane.
4. Click the color palette and select a color for the border. The Format Pane will look like Figure 1-36.



Data colors

Default color

Show all Off

Revert to default

Map controls

Title On

Background Off

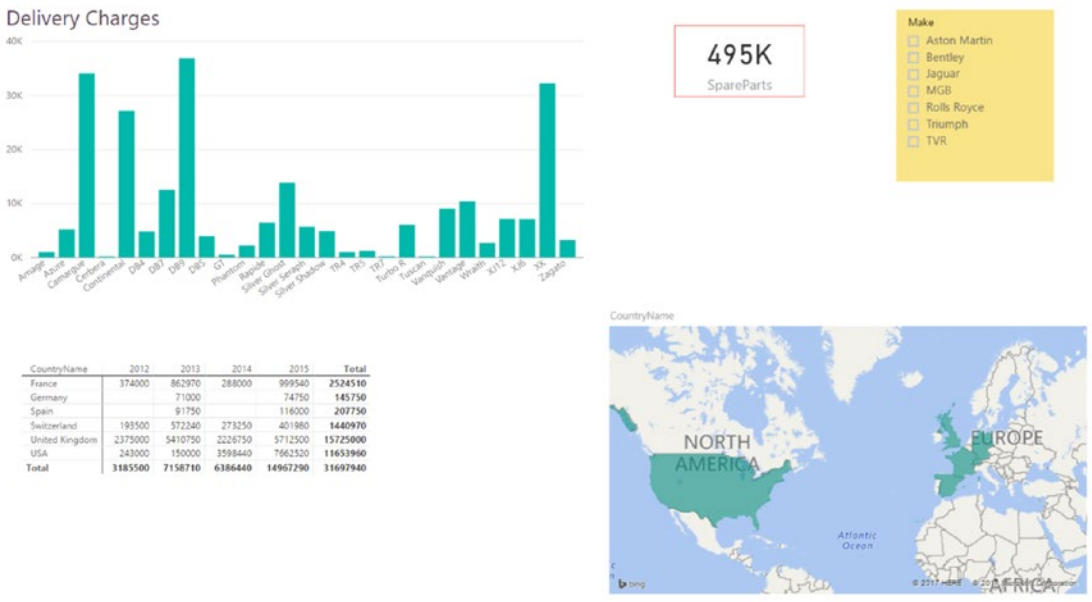
Lock aspect Off

Border Off

General

Figure 1-36. *The Format Pane for maps*

Now that you have tweaked the presentation of your starter report, it should look something like the one shown in Figure 1-37.



CountryName	2012	2013	2014	2015	Total
France	374000	862970	288000	999540	**2524510**
Germany		71000		74750	**145750**
Spain		91750		116000	**207750**
Switzerland	193500	572240	273250	401980	**1440970**
United Kingdom	2375000	5410750	2226750	5712500	**15725000**
USA	243000	150000	3598440	7662520	**11653960**
Total	**3185500**	**7158710**	**6386440**	**14967290**	**31697940**

Figure 1-37. *The original report with initial formatting added*

Admittedly, these are only a few quick modifications to introduce you to some of the formatting possibilities that are currently available in Power BI Desktop. There is much, much more that you can do once you are at ease with Power BI Desktop and have understood the basic approach to formatting visuals. With a little practice, you should be able to produce high-quality reports and dashboards that enable your analyses to leap out at the audience and convey the information that you want to deliver. If you want to take a look at the final, formatted report, you can open the Power BI Desktop file C:\PowerBiDesktopSamples\CH01\CH01Example_Formatted.pbix.

■ **Note** Deleting a visual is as simple as selecting it and pressing the Delete key. Removing a visual will not affect the underlying data in any way.

Creating and Modifying Reports

So far in this chapter, we have treated the Power BI Desktop file as if it consisted of only a single page. In practice, you are likely to need to base several pages of analysis and information on the same underlying dataset. Consequently, Power BI Desktop makes it easy to add, copy, and delete the pages in your original file so that you can create complex data stories that all use the same dataset.

If (as I presume is probably the case for many Power BI users) you are used to using Excel, then you will likely find the way that pages are handled in self-service BI incredibly simple, because in Power BI Desktop, each page is very similar to an Excel worksheet. To make matters clearer, look at Figure 1-38, where you can see the page tabs of Power BI Desktop.

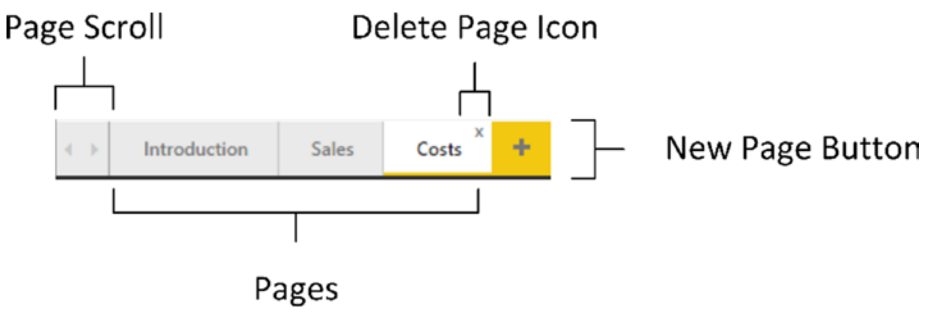


Figure 1-38. *The Power BI Desktop page tabs*

Adding Pages

When you open a new Power BI Desktop file, it always defaults to having a single page, thoughtfully named Page 1. You can add a new page as follows:

In the Home ribbon, click the New Page button. A new blank page named Page *n* will be added to the existing collection of pages in the report.

■ **Tip** As an alternative to the New Page button, you can always click the small plus-sign tab at the bottom of the screen, as seen in Figure 1-38.

Renaming Pages

You can rename pages, as follows:

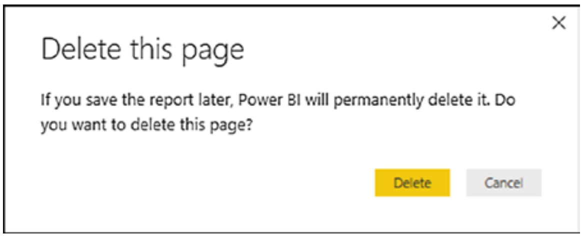
1. Double-click the tab of the page that you want to rename. The existing name will be highlighted.
2. Enter a new name for the page.
3. Press Enter. Click inside the dashboard canvas for the page or click another tab to confirm your changes.

Deleting Pages

If a page is no longer any use to you, then you can delete it, of course.

1. Hover the mouse pointer over the tab for the page that you want to delete. A small cross appears at the top right of the page name, as you can see in Figure 1-38.

2. Click the cross. A warning dialog will appear, as shown in Figure 1-39.


Delete this page

If you save the report later, Power BI will permanently delete it. Do you want to delete this page?

Figure 1-39. *The page delete dialog*

3. Click Delete. The page will be deleted and *all* visuals on the page are removed from the file.

Moving Pages

To alter the sequencing of the pages in your report, do the following:

1. Click the tab corresponding to the page that you want to move.
2. Drag the page tab left or right to a new position in the set of pages.

Duplicating Pages

If a page contains a set of elements that you want to reuse (it may be a template page containing a logo and the background for a series of pages in a report, for instance), then you can make duplicates of pages, as follows:

1. Hover the mouse pointer over the tab for the page that you want to delete.
2. Right-click the tab. A pop-up menu will appear.
3. Select Duplicate Page.

An identical copy of the page will appear to the right of any existing pages. There is also a Duplicate Page option in the popup menu for the New Page button if you prefer.

Scrolling Through Collections of Pages

If your report contains dozens of pages, then it can get very wearing to trawl through the set of pages one at a time. Instead, you can click the page scroll buttons (see Figure 1-38) to scroll through the set of pages in a Power BI Desktop file.

Conclusion

Welcome to Power BI Desktop. In a short chapter, you have seen just how amazingly simple and intuitive it is to use this free, self-service business intelligence tool from Microsoft. You have seen how to load data from an external source. This chapter has also given you an idea of the wealth of potential sources of data that Power BI Desktop can handle. You saw how to take data and use it to create tables, charts, maps, and slicers in an interactive dashboard that you can now share with co-workers and friends, if you want to. Once you have created your reports, you can then enhance them visually through a wide range of formatting options.

Yet all of this merely scratched the surface of the vast potential of this amazing application. As you will discover as you progress through this book, you are on the cusp of discovering a veritable treasure trove of analytical possibilities and stunning visualizations that will help you drive your data analysis and presentation skills to the next level.

CHAPTER 2

Discovering and Loading File-Based Data with Power BI Desktop

Before you can present any analysis or insight, you need data. Your sources could be in many places and in many formats. Nonetheless, you need to access them, look at them, select them, and quite possibly restructure them or clean them up to some extent. You may also need to join many separate data sources before you shape the data into a coherent model that you can use as the foundation for your dashboards and reports. The amazing thing is that you can do all of this using Power BI Desktop without needing any other tools or utilities.

Discovering, loading, cleaning, and modifying source data is one of the areas where Power BI Desktop really shines. It allows you to accomplish the following:

- *Data discovery*: Find and connect to a myriad of data sources containing potentially useful data. This can be from both public and private data sources. This is the subject of Chapters 2 through 5.
- *Data loading*: Select the data you have examined and load it into Power BI Desktop for shaping. You saw this briefly in Chapter 1.
- *Data modification*: Modify the structure of each dataset that you have imported, then filter and clean the data itself (we will look at this in detail in Chapters 6 through 8).
- *Data shaping*: Join datasets to create a clear, unified, and accessible data model. You will learn how to do this in Chapter 10.

Although I have outlined these four steps as if they are completely separate and sequential, the reality is that they often blend into a single process. Indeed, there could be many occasions when you will examine the data *after* it has been loaded into Power BI Desktop—or clean datasets *before* you load them. The core objective will, however, always remain the same: find some data and then load it into Power BI Desktop where you can tweak, clean, and shape it.

This process could be described simplistically as "First, catch your data." In the world of data warehousing, the specialists call it ETL, which is short for **E**xtract, **T**ransform, and **L**oad. Despite the reassuring confidence that the acronym brings, this process is rarely a smooth, logical progression through a clear-cut series of steps. The reality is often far messier than that. You may often find yourself importing some data, cleaning it, importing some more data from another source, combining the second dataset with the first one, removing some rows and columns, and then repeating many of these operations several times over.

In this and the following few chapters I will try to show you how the process can work in practice using Power BI Desktop. I hope that this will make the various steps that comprise an ETL process clearer. All I am asking is that you remain aware that the range of options that Power BI Desktop includes make it a multifaceted and tremendously capable tool. The science is to know *which* options to use. The art is to know *when* to use them.

The Power BI Desktop Query Editor

This chapter extends the data load process that you saw briefly in Chapter 1. In the previous chapter, you loaded data directly into Power BI Desktop—or more precisely, into the Power BI Desktop data model. In this chapter, you extend this approach with an additional step. You will, for some data sources, see how to load data into the Power BI Desktop Query Editor *before* adding it to the data model. This "detour" is the part of the process that allows you to cleanse and transform the data before it is added to the data model. Of course, if your data is perfect, then you can add it straight into the data model and start building reports. Indeed, if you are connecting to cleansed and structured corporate data, you may want to jump straight to Chapter 10 and learn how to create a data model. However, if your data needs any adjustment at all, then the Power BI Desktop Query Editor will likely soon become a trusted tool. Consequently, it is probably worth reading Chapters 2 through 9 that describe how to load data from a range of possible sources and then shape, modify, and structure your data so that it becomes a clear source of new and valuable insights.

As the first part of your journey through the data mashup process, this chapter will show you how to find and load data from a variety of file-based sources. These kinds of data are typically those that you can either locate on a shared network drive, download from the Internet, receive as an e-mail attachment, or copy to your computer's local drive. The files that are used in the examples in this chapter are available on the Apress web site. If you have followed the download instructions in Appendix A, then these files will be in the C:\PowerBIDesktopSamples\CH02 folder.

Data Sources

In the first chapter, you saw how quickly and easily you can load data into Power BI Desktop and create stunning dashboards. It is now time to take a wider look at the *types* of file-based data that Power BI Desktop can ingest and manipulate.

As the sheer wealth of possible data sources can seem overwhelming at first, Power BI Desktop groups potential data sources into the following categories:

- *File*: Includes Excel files, CSV (comma-separated values) files, text files, JSON files, and XML files. Power BI Desktop can even load entire folders full of files.
- *Database*: A comprehensive collection of relational databases that are currently in the workplace and in the cloud, including (among others) MS Access, SQL Server, and Oracle. The full list of those available when this book went to press is given in the following chapter.
- *Azure*: This option lets you see an immense range of data types that is hosted in the Microsoft Cloud. This covers data formats from SQL Server through to big data sources. You can see how a few of these are used with Power BI Desktop in Chapter 5.
- *Online services*: These sources range from SharePoint lists to SalesForce, Dynamics 365 to Facebook—and many, many others. Some of these are examined in Chapter 5.
- *Other*: A considerable and ever-growing range of data sources, from Facebook to Microsoft Exchange. Some of these will be touched on in the course of the four chapters that cover accessing data in Power BI Desktop.

The list of possible data sources is changing all the time and you need to be aware that you have to look at the version of Power BI Desktop that you are using if you want an exhaustive list of the all available data sources that you can use. Indeed, I expect that several more will have been added by the time that you read this book.

You can also list the contents of folders on any available local disk, network share, or even in the cloud and then leverage this to import several files at once. Similarly (if you have the necessary permissions), you can list the databases and data available on the database servers you connect to. This way, Power BI Desktop can provide not only the data, but also the *metadata*—or data about data—that can help you to take a quick look at potential sources of data and only choose those that you really need.

Unfortunately, the sheer range of data sources from which Power BI Desktop can read data is such that we do not have space in a few chapters to examine the minutiae of every one. Consequently, we will take a rapid tour of *some* of the most frequently used data sources in this and the next few chapters. Fortunately, most of the data sources that Power BI Desktop can read are used in a similar way. This is because the Power BI Desktop interface does a wonderful job of making the arcane connection details as unobtrusive as possible. So even if you are faced with a data source that is not described in these chapters, you will nonetheless see a variety of techniques that can be applied to virtually any of the data sources that Power BI Desktop can connect to.

■ **Note** The list of data sources that Power BI Desktop can access is growing all the time. Consequently, when you read this book you will probably find even more sources than those described in this and the next three chapters.

File Sources

Sending files across networks and over the Internet or via e-mail has become second nature to most of us. As long as the files that you have obtained conform to some of the widely recognized standards currently in use (of which you will learn more later), you should have little difficulty loading them into Power BI Desktop.

The file sources that Power BI Desktop can currently read and from which it can load data are given in Table 2-1.

Table 2-1. *File Sources*

File Source	Comments
Excel	Allows you to read Microsoft Excel files (versions 97 to 2016) and load worksheets, named ranges, and tables.
CSV	Lets you load text files that conform to the CSV (comma-separated values) format.
XML	Allows you to load data from XML files.
Text	Lets you load text files using a variety of separators.
folder	Lets you load the information about all the files in a folder.
SharePoint folder	Allows you to list the files in a SharePoint folder.
Access database	Lets you connect to a Microsoft Access file on your network and load queries and tables.
JSON	Allows you to load data from JSON files.

■ **Note** I realize that Power BI Desktop considers MS Access to be a database and not a "file" data type. While I completely agree with this classification, I prefer nonetheless to treat Access as if it were a file-based data source, given that all the data resides in a single file that can be copied and e-mailed, and not in a database on a distant server. For this reason, we will look at MS Access in this chapter, and not the next one that deals with corporate data sources.

Loading Data

It is time to start looking at the heavy-lifting aspect of Power BI Desktop and how you can use it to load data from a variety of different sources. I will begin on the bunny slopes with a simple example of "scraping" data from a web page. Then, given the plethora of available data sources, and to give the process a clearer structure, we will load data from several of the ubiquitous file-based data sources that are found in most workplaces. These data sources are the basis of the data that you will learn to tweak and "mash up" in Chapters 6 through 8. This data could also become the basis of many of the dashboards that you will create in Chapters 14 to 22. These sources are as follows:

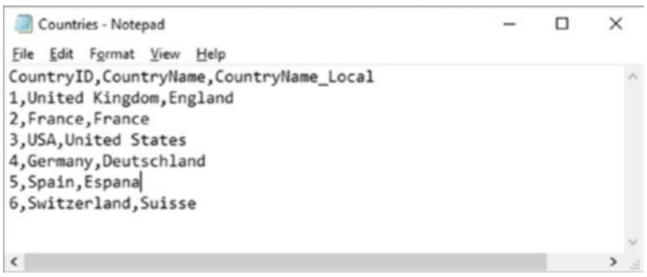
- CSV
- Text
- XML
- Excel
- Access
- JSON

You will see how to load multiple text or CSV files at once. To conclude the chapter, you will see how to store only the details about the files in a folder rather than the files themselves.

CSV Files

The scenario is as follows: you have been given a CSV file containing a list of data. You now want to load this into Power BI Desktop so that you can look at the data and consider what needs to be done (if anything) to make it useable.

First, you need an idea of the data that you want to load. If you open the source file C:\PowerBIDesktopSamples\CH02\Countries.csv with a text editor, such as Notepad, you can view its contents. This is what you can see in Figure 2-1.



```
CountryID,CountryName,CountryName_Local
1,United Kingdom,England
2,France,France
3,USA,United States
4,Germany,Deutschland
5,Spain,Espana
6,Switzerland,Suisse
```

Figure 2-1. *The contents of the Countries.csv file*

The following steps explain what you have to do to load the contents of this file into Power BI Desktop:

1. Open Power BI Desktop and close the splash screen.
2. In the Power BI Desktop Home ribbon, click the Get Data button (and not the small triangle that displays menu options).
3. Click File on the left. You will see something like Figure 2-2 (the Get Data dialog).

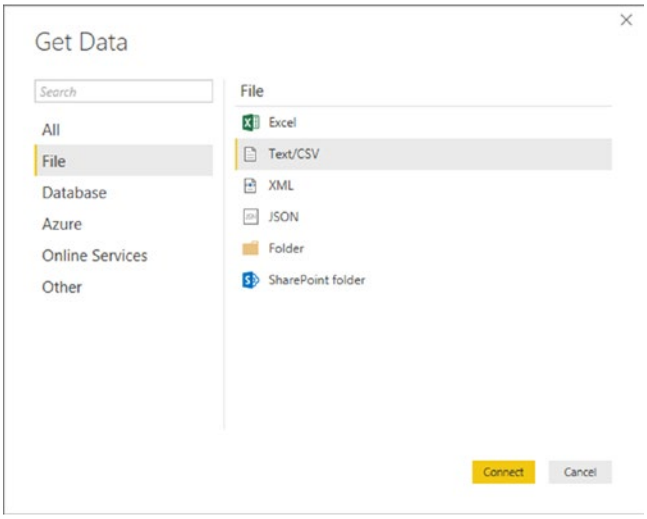
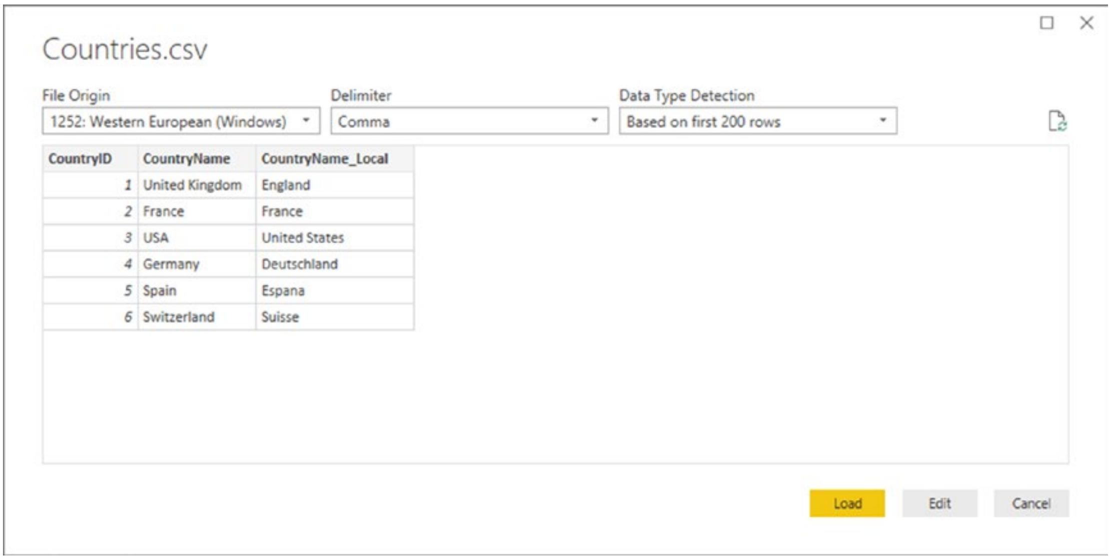


Figure 2-2. *The contents of the Countries.csv file*

4. Click Text/CSV on the right of the dialog.
5. Click Connect. The Open dialog will appear.
6. Navigate to the folder containing the file that you want to load and select it (C:\PowerBIDesktopSamples\CH02\Countries.csv, in this example).
7. Click Open. A dialog will display the initial contents of the file, as shown in Figure 2-3.

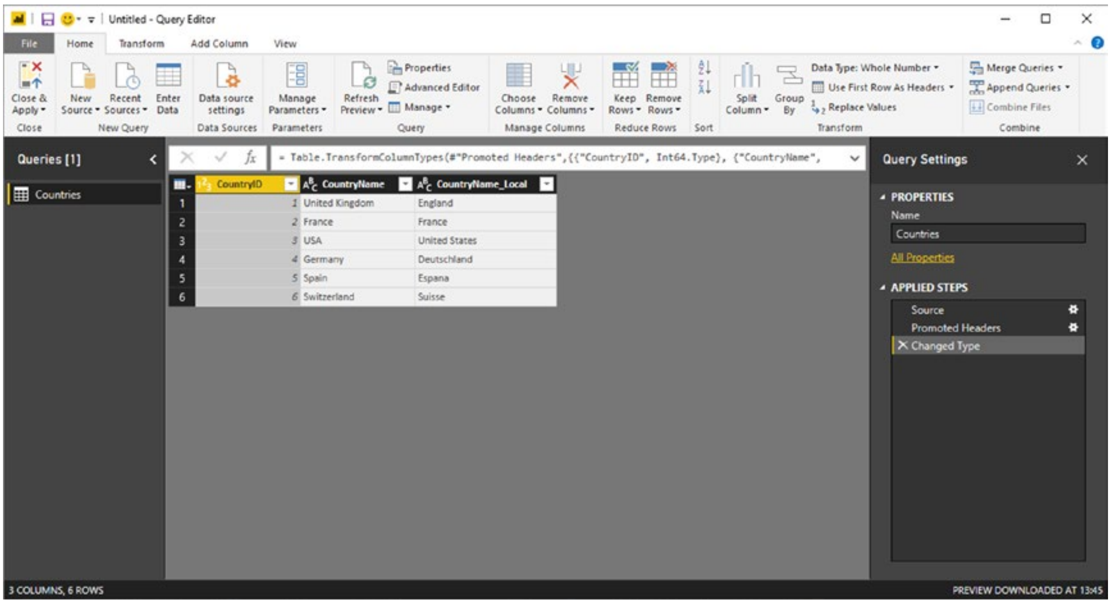

Countries.csv

File Origin: 1252: Western European (Windows)
Delimiter: Comma
Data Type Detection: Based on first 200 rows

CountryID	CountryName	CountryName_Local
1	United Kingdom	England
2	France	France
3	USA	United States
4	Germany	Deutschland
5	Spain	Espana
6	Switzerland	Suisse

Figure 2-3. *The Power BI Desktop file dialog*

8. Click the Edit button. The Power BI Desktop Query window appears; it contains a sample of the contents of the CSV file—or possibly the entire file if it is not too large. You can see this in Figure 2-4.


Untitled - Query Editor

```
= Table.TransformColumnTypes(#"Promoted Headers",{{"CountryID", Int64.Type}, {"CountryName",
```

	CountryID	CountryName	CountryName_Local
1	1	United Kingdom	England
2	2	France	France
3	3	USA	United States
4	4	Germany	Deutschland
5	5	Spain	Espana
6	6	Switzerland	Suisse

Query Settings
PROPERTIES
Name: Countries
APPLIED STEPS: Source, Promoted Headers, Changed Type

3 COLUMNS, 6 ROWS
PREVIEW DOWNLOADED AT 13:45

Figure 2-4. *The Power BI Desktop Query window with the contents of a CSV file loaded*

9. Click the Close & Apply button in the Power BI Desktop Query window (you can see this at the top left in Figure 2-4). The Power BI Desktop Query Editor will close and return the focus to the Power BI Desktop window, where you can see that the Countries dataset appears in the Fields list on the right of the screen, as shown in Figure 2-5.

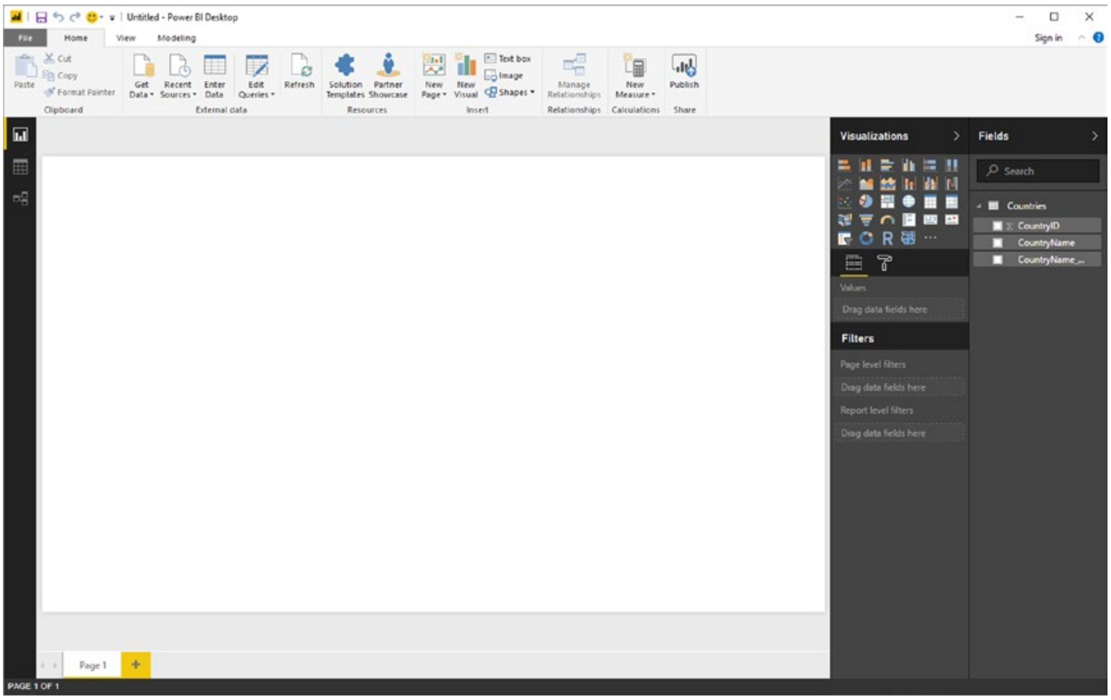


Figure 2-5. *The Power BI Desktop Query window with a file loaded*

And that, for the moment, is that. You have loaded the file into Power BI Desktop in a matter of a few clicks and it is ready for use in dashboards and reports. If necessary, you can expand the Countries dataset and then see the fields that this dataset contains—just as they were in the dialog shown in Figure 2-3.

In later chapters, you will learn how to shape this data. For the moment, however, let's continue looking at some other file-based data sources.

What Is a CSV File?

Before we move on to other file types, there are a few comments I need to make about CSV files. There is a technical specification of what a "true" CSV file is, but I won't bore you with that. What's more, many programs that generate CSV files do not always follow the definition exactly. What matters is that Power BI Desktop can handle text files that

- Have a .csv extension (it uses this by default to apply the right kind of processing).
- Use a comma to separate the elements in a row. This, too, is a default that can be overridden by selecting a delimiter from those in the dialog shown in Figure 2-3.

- End with a line feed, carriage return, or line feed/carriage return.
- Can, optionally, contain double quotes to encapsulate fields. These will be stripped out as part of the data load process. If there are double quotes, they do not have to appear for every field, nor even for every record in a field that can have occasionally inconsistent double quotes.
- Can contain "irregular" records; that is, rows that do not have every element found in a standard record. However, the first row (whether or not it contains titles) must cover every element found in all the remaining records in the list. Put simply, any other record can be shorter than the first one but cannot be longer.
- Do not contain anything other than the data itself. If the file contains header rows or footer rows that are not part of the data, then Power BI Desktop cannot load the dataset without further work. There are workarounds to this all-too-frequent problem; one is given in Chapter 6.

■ **Note** Another way of accessing CSV files is to click Get Data ➤ File and select Text/CSV in the Get Data dialog.

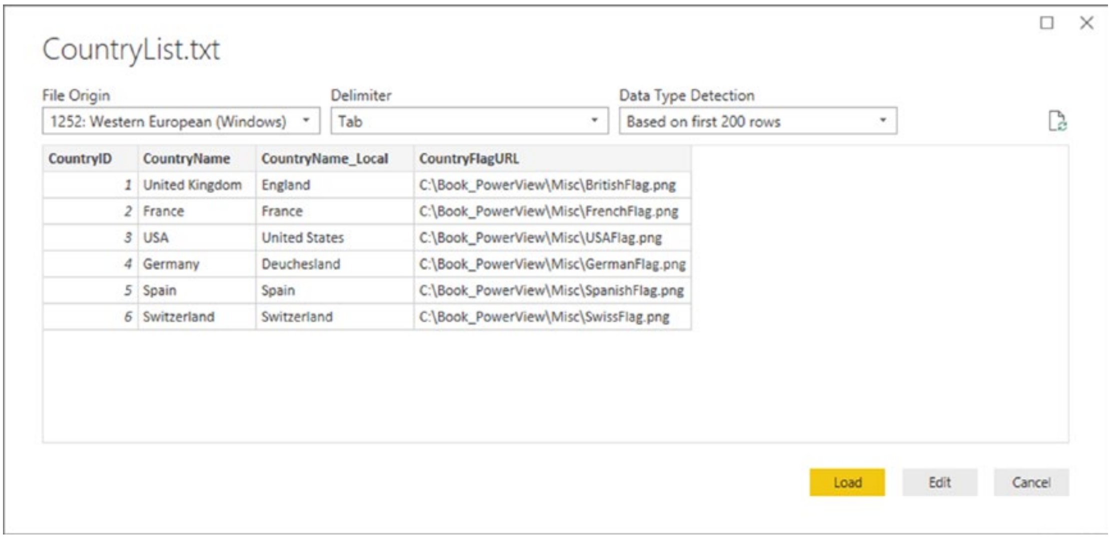
Text Files

If you followed the process for loading a CSV file in the previous section, then you will find that loading a text file is virtually identical. This is not surprising. Both are text files and both should contain a single list of data. The following are the core differences:

- A text file can have something *other* than a comma to separate the elements in a list. You can specify the delimiter when defining the load step.
- A text file should normally have the extension .txt (though this, too, can be overridden).
- A text file *must* be perfectly formed; that is, every record (row) must have the same number of elements as every other record.
- A text file, too, *must not* contain anything other than the dataset if you want a flawless data load the first time.
- If a text file encounters difficulties, it should import the data as a single column that you can then try and split up into multiple columns, as described in Chapter 8.

Here, then, is how to load a text file into Power BI Desktop:

1. In the Power BI Desktop ribbon, click Get Data ➤ Text/CSV. The Open dialog will be displayed.
2. Navigate to the folder containing the file and select the file (C:\PowerBIDesktopSamples\CH02\CountryList.txt, in this example).
3. Click Open. A dialog will display the initial contents of the file. You can, of course, double-click the file name rather than click Open. The dialog should look like the one in Figure 2-6.


CountryList.txt

File Origin: 1252: Western European (Windows)
Delimiter: Tab
Data Type Detection: Based on first 200 rows

CountryID	CountryName	CountryName_Local	CountryFlagURL
1	United Kingdom	England	C:\Book_PowerView\Misc\BritishFlag.png
2	France	France	C:\Book_PowerView\Misc\FrenchFlag.png
3	USA	United States	C:\Book_PowerView\Misc\USAFlag.png
4	Germany	Deuchesland	C:\Book_PowerView\Misc\GermanFlag.png
5	Spain	Spain	C:\Book_PowerView\Misc\SpanishFlag.png
6	Switzerland	Switzerland	C:\Book_PowerView\Misc\SwissFlag.png

Figure 2-6. *The contents of a text file ready for loading into Power BI Desktop*

4. Click the Cancel button (because after a quick look at the contents of the file, you have decided that you do not really need it).

■ **Note** As text-based files (which include CSV files) are such a frequent source of data, you will nearly always see the Text/CSV option directly accessible in the popup menu that you access by clicking the small triangle in the Get Data button in the Home ribbon. If this option is not visible, you can instead select Get Data ➤ File and select Text/CSV, as you did previously.

Where Power BI Desktop is really clever is that it can make a very educated guess as to how the text file is structured; that is, it can nearly always guess the field separator (the character that isolates each element in a list from the other elements). And so not only will it break the list into columns, but it will also avoid importing the column separator. If it does not guess correctly, then don't despair. You will learn how to correct this in Chapter 8.

Looking at the contents of a file and then deciding not to use it is part and parcel of the *data discovery* process that you will find yourself using when you work with Power BI Desktop. The point of this exercise is to show you how easy it is to glance inside potential data sources and then decide whether to import them into the data model or not. Moreover, it can be easier to see the first few rows of large text or CSV files directly in the Load dialog of Power BI Desktop than it is to open the whole file in a text editor.

■ **Tip** At the risk of stating the obvious, you can press Enter to accept a default choice in a dialog and press Esc to cancel out of the dialog.

Text and CSV Options

You can see in Figure 2-6 that there are few options available that you can tweak when loading text or CSV files. Most of the time Power BI Desktop will guess the correct settings for you. However, there could be times when you will need to adjust these parameters slightly. The potential options that you can modify are

- File Origin
- Delimiter
- Data Type Detection

File Origin

This option defines the character encoding in which the file is stored. Different character sets can handle differing ranges of characters, such as accents. Normally this information is correctly interpreted by Power BI Desktop, and you should only need to select a different character set (file origin) on very rare occasions.

Delimiter

Power BI Desktop will try and guess the special character that is used in a text or CSV file to separate the "columns" of data. Should you wish to override the chosen delimiter, you have the choice of:

- Colon
- Comma
- Equals sign
- Semicolon
- Space
- Tab character

You can also decide to enter a custom delimiter such as the pipe (|) character, or even specify that every field has a fixed width.

Data Type Detection

Power BI Desktop will make an educated guess at the data encoding that is used in a text or CSV file. By default, to save time, it will only read the first 200 records. However, you can choose from any of the following three options:

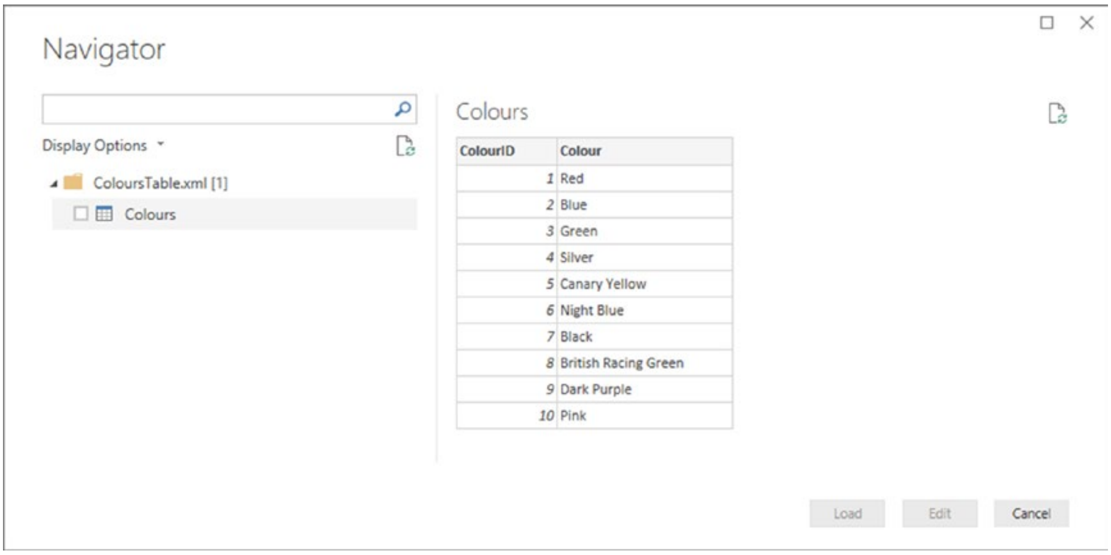
- Read the first 200 rows
- Read the entire file
- No data type detection

■ **Note** Be warned that reading a large file in its entirety can take quite a while. However, without accurate data type detection, you risk seeing some weird characters in the data that you load.

XML Files

XML, or Extensible Markup Language, is a standard means of sending data between IT systems. Consequently, you likely will load an XML file one day. Although an XML file is just text, it is text that has been formatted in a very specific way, as you can see if you ever open an XML file in a text editor such as Notepad. Do the following to load an XML file:

1. In the Power BI Desktop ribbon, click the small triangle on the Get Data button, and then click More in the menu that appears. Next, in the Get Data dialog, select File and XML.
2. Click Connect. The Open dialog will appear.
3. Navigate to the folder containing the file and select the file (C:\PowerBIDesktopSamples\CH02\ColoursTable.xml, in this example).
4. Click Open. The Navigator dialog will open.
5. Click the Colours dataset in the left-hand pane of the Navigator dialog. The contents of this part of the XML file will be displayed on the right of the Navigator dialog, as shown in Figure 2-7.



Colours

ColourID	Colour
1	Red
2	Blue
3	Green
4	Silver
5	Canary Yellow
6	Night Blue
7	Black
8	British Racing Green
9	Dark Purple
10	Pink

Figure 2-7. *The Navigator dialog before loading an XML file*

6. Click the check box to the left of the Colors dataset on the left. The Load and Edit buttons will be enabled.
7. Click the Edit button. The Power BI Desktop Data window will display the contents of the XML file.
8. Click the Close and Apply button in the Power BI Desktop Data window. You will see that the Colors dataset appears in the Fields list on the right of the screen.

The actual internal format of an XML file can get extremely complex. Sometimes an XML file will contain only one dataset; sometimes it will contain many separate datasets. On other occasions, it will contain one dataset whose records contain nested levels of data that you need to handle by expanding a hierarchy of elements. You will see how the Navigator dialog handles nested hierarchies of data in the following chapter in the context of database sources.

■ **Note** Certain types of data source allow you to load multiple sets of data simultaneously. XML files (unlike CSV and text files) can contain multiple independent datasets. You can load several datasets simultaneously by selecting the check box to the left of each dataset that you want to load from the XML file.

Excel Files

You are probably already a major Excel user and have many, many spreadsheets full of data that you want to rationalize and use for analysis and presentation in Power BI Desktop. So, let's see how to load the contents of an Excel file.

1. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click Excel. The Open dialog will appear.
2. Navigate to the directory containing the file that you want to look at (C:\PowerBIDesktopSamples\CH02, in this example).
3. Select the source file (InvoicesAndInvoiceLines.xlsx, in this example) and click OK. The Navigator dialog will appear, showing the worksheets, tables, and ranges in the workbook file, as shown in Figure 2-8.

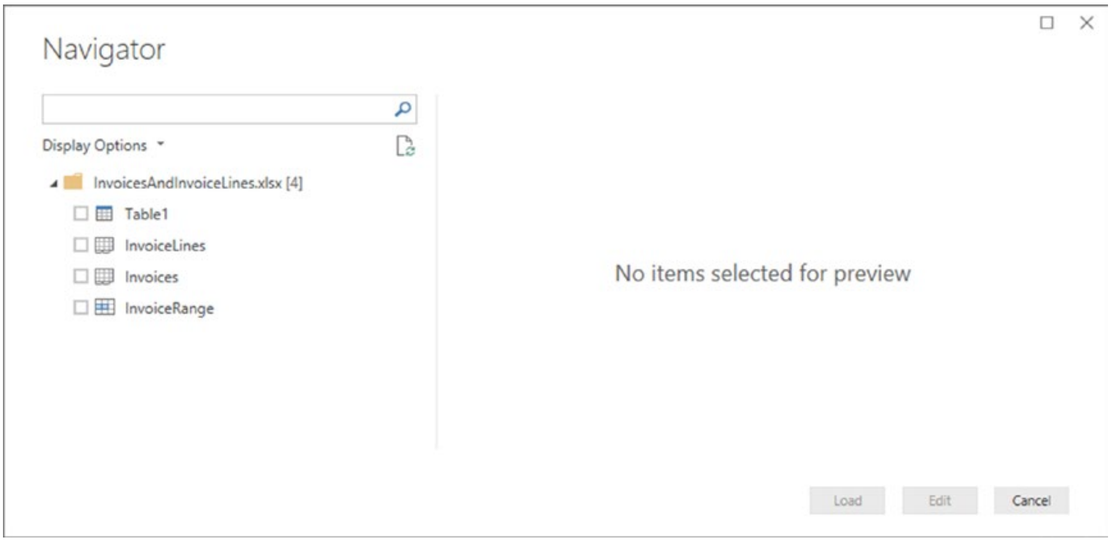


Figure 2-8. The Navigator dialog before loading data from an Excel workbook

4. Click one of the datasets listed on the left of the Navigator dialog. The top few rows of the selected spreadsheet will appear on the right of the dialog to show you what the data in the chosen dataset looks like.

5. Click the check boxes to the left of the Invoices and InvoiceLines datasets on the left.

6. Click Load. The selected worksheets will be loaded into the Power BI Desktop data model and will appear in the Fields list in the Report window.

As you can see from this simple example, having Power BI Desktop read Excel data is really not difficult. You could have edited this data in Power BI Desktop Query Editor before loading it, but as the data seemed clean and ready to use, I preferred to load it straight into Power BI Desktop (or rather the Power BI Desktop data model). As well, you saw that Power BI Desktop can load multiple datasets at the same time from a single data source. However, you might still be wondering about a couple of things that you saw during this process, so here are some anticipatory comments:

The Navigator dialog displays

- Worksheets (Invoices and InvoiceLines in Figure 2-8)
- Named ranges (InvoiceRange in Figure 2-8)
- Named tables (Table1 in Figure 2-8)

Each of these elements is represented by a different icon in the Navigator dialog. Sometimes these can, in effect, be duplicate references to the same data, so you should really use the most precise data source that you can. For instance, I advise using a named table or a range name rather than a worksheet source, as the latter could easily end up containing “noise” data (that is, data from outside the rows and columns that interest you), which would make the load process more complex than it really needs to be. Indeed, unless a worksheet is prepared and structured, ready for loading into Power BI Desktop, you could end up with superfluous data in your data model.

■ **Note** Power BI Desktop will list and use data connections to external data sources (such as SQL Server, Oracle, or SQL Server Analysis Services) in a source Excel workbook *if* the data connection is active and has returned data to the workbook. Once a link to Power BI Desktop has been established, you can delete the data table itself in the source Excel workbook—and still load the data over the data connection in the source workbook into Power BI Desktop.

Power BI Desktop will *not* take into account any data filters on an Excel data table, but will load all the data that is in the source table. Consequently, you will have to reapply any filters (of which you’ll learn more in chapter 6) in Power BI Desktop if you want to subset the source data.

Importing Excel and Power View Items

Power BI Desktop is not the first incarnation of Power BI; the data model that it uses has been around for some years now. So, you may already be an accomplished Power View expert using Power View for Excel—or you may have advanced data models that you have built using Power Pivot in Excel that you want to transfer into Power BI Desktop.

Fortunately, the team at Microsoft has thought of this, and the result is that you can transfer all your effort from Excel (Power View dashboards, Power Pivot data models, and DAX metrics) into Power BI Desktop with remarkable ease. Here is how:

1. Open a new, blank Power BI Desktop file.
2. In the File menu, select Import ➤ Excel Workbook Contents, as shown in Figure 2-9.

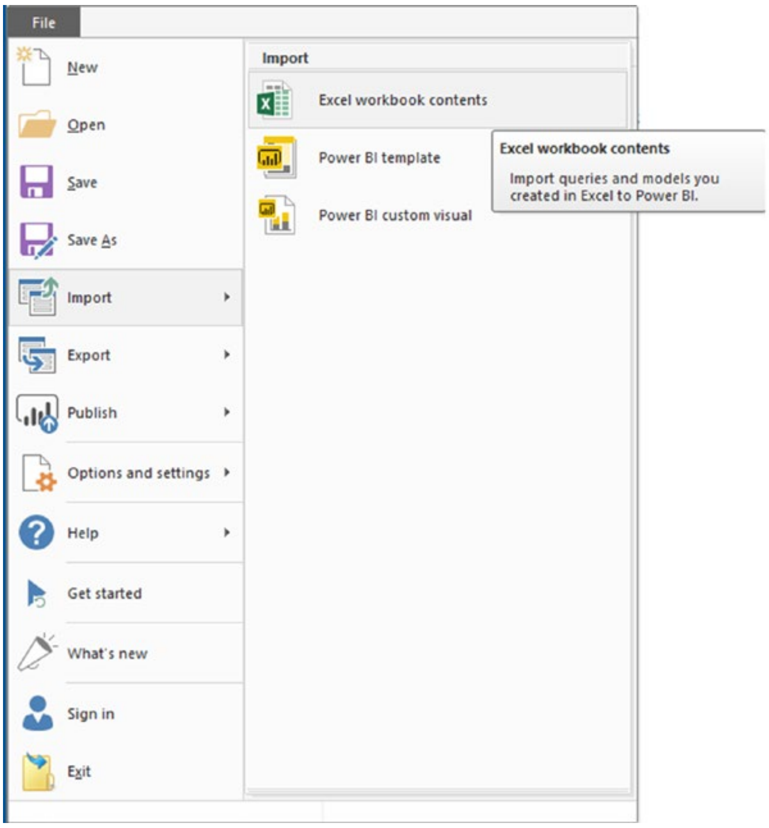
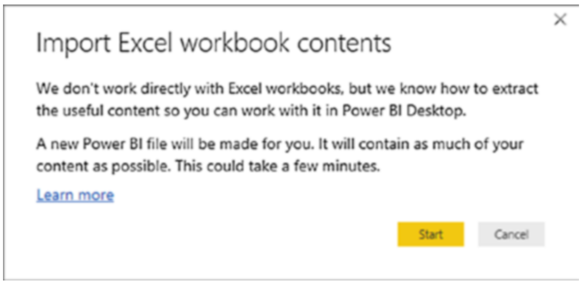


***Figure 2-9.** Importing existing Power View or Power Pivot items from Excel*

3. The Windows Open dialog appears, from which you can select an existing Excel file containing Power View or Power Pivot items. In this example, you can use the file CarSalesForPowerBI.xlsx from the sample files for this chapter.
4. Click Open.
5. Power BI Desktop will import any compatible items and display the import screen (as shown in Figure 2-10) during the import process.


Import Excel workbook contents

We don't work directly with Excel workbooks, but we know how to extract the useful content so you can work with it in Power BI Desktop.

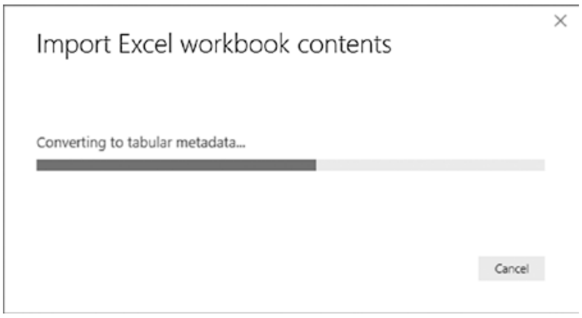
A new Power BI file will be made for you. It will contain as much of your content as possible. This could take a few minutes.

Learn more

Start Cancel

Figure 2-10. *Importing Power View and Power Pivot elements from Excel*

6. Click Start. Power BI Desktop will begin to load and convert data and elements from Excel. Indeed, you could see further specific questions. Then the import will continue, showing the progress dialog that you see in Figure 2-11.

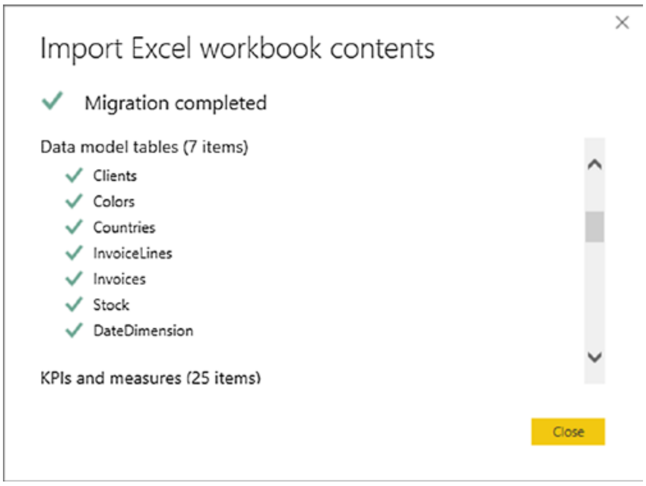

Import Excel workbook contents

Converting to tabular metadata...

Cancel

Figure 2-11. *The import progress dialog*

7. Once the import process has successfully finished, Power BI Desktop will display the summary dialog that you see in Figure 2-12.


Import Excel workbook contents

✓ Migration completed

Data model tables (7 items)

- ✓ Clients
- ✓ Colors
- ✓ Countries
- ✓ InvoiceLines
- ✓ Invoices
- ✓ Stock
- ✓ DateDimension

KPIs and measures (25 items)

Close

Figure 2-12. *The import summary dialog*

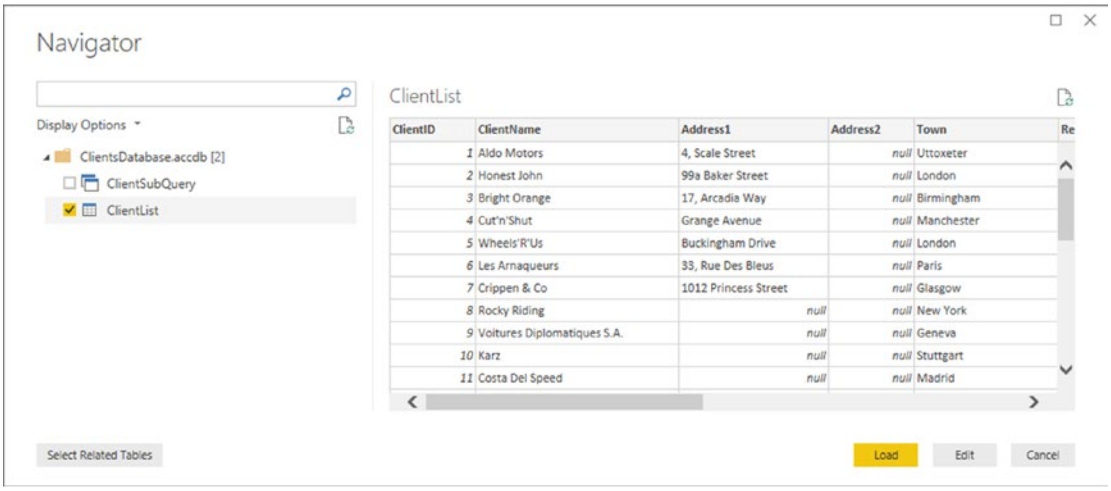
8. Click Close. The items become a Power BI Desktop report.

As this book went to press, there were a few aspects of some Power View visualizations that were not imported perfectly into the Power BI Desktop data model. However, as this technology is currently developing at a rapid pace, you could well find that these minor limitations have been resolved by the time that you read this book. In any case, I advise you to consult the Power BI web site for up-to-date details on any remaining limitations concerning the conversion of Excel objects to Power BI Desktop reports.

Microsoft Access Databases

Another well-used data repository that proliferates in many corporations today is Microsoft Access. It is a powerful desktop relational database and can contain hundreds of tables, each containing millions of records. So we need to see how to load data from this particular source. Moreover, Power BI Desktop can be particularly useful when handling Access data because it allows you to see the contents of Access databases without even having to install Access itself.

1. In the Power BI Desktop ribbon, click Get Data ➤ More ➤ Database and select Access Database in the Get Data dialog.
2. Click Connect and navigate to the MS Access database containing the data that you want to load (C:\PowerBIDesktopSamples\CH02\ClientsDatabase.accdb, in this example).
3. Select the Access file and click OK. The Navigator dialog appears; it lists all the tables and queries in the Access database.
4. Check the check box for the ClientList dataset. This displays the contents of the table, as you can see in Figure 2-13.



ClientList

ClientID	ClientName	Address1	Address2	Town	Re
1	Aldo Motors	4, Scale Street	null	Uttoxeter	
2	Honest John	99a Baker Street	null	London	
3	Bright Orange	17, Arcadia Way	null	Birmingham	
4	Cut'n'Shut	Grange Avenue	null	Manchester	
5	Wheels'R'Us	Buckingham Drive	null	London	
6	Les Arnaqueurs	33, Rue Des Bleus	null	Paris	
7	Crippen & Co	1012 Princess Street	null	Glasgow	
8	Rocky Riding	null	null	New York	
9	Voitures Diplomatiques S.A.	null	null	Geneva	
10	Karz	null	null	Stuttgart	
11	Costa Del Speed	null	null	Madrid	

Figure 2-13. *The Navigator dialog before loading data from an Access database*

5. Click Load. The Power BI Desktop window opens and displays the table in the Fields list in the Report window.

If you look closely at the left of the Navigator dialog in Figure 2-13, you can see that it displays two different icons for Access objects:

- A table for Access data tables
- Two small windows for Access queries

This can help you to understand the type of data that you are looking at inside the Access database.

■ **Note** Power BI Desktop cannot see linked tables in Access, only imported tables or tables that are actually in the Access database. It can, however, read queries overlaid upon native, linked, or imported data.

JSON Files

More and more data is now being exchanged in a relatively new format called JSON. This stands for JavaScript Object Notation, and it is considered an efficient and lightweight way of transferring potentially large amounts of data.

Now, while Power BI Desktop can connect to JSON data files, these are not always instantly comprehensible. So be warned that, while this section will teach you how to connect to JSON data using Power BI Desktop, you will have to wait for Chapter 8 to see how this connection can be tweaked to convert it into meaningful information.

To connect to a JSON file:

1. In the Power BI Desktop ribbon, click Get Data ➤ File and select JSON in the list of file sources on the right of the Get Data dialog. The dialog should look something like the one shown in Figure 2-14.

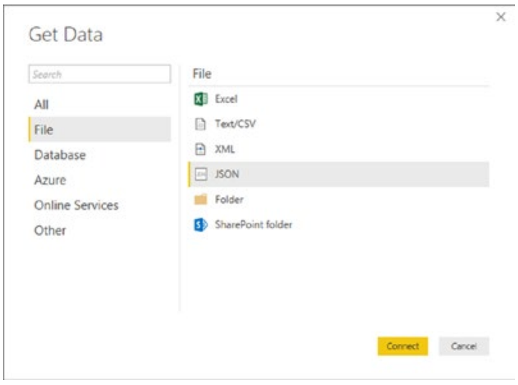
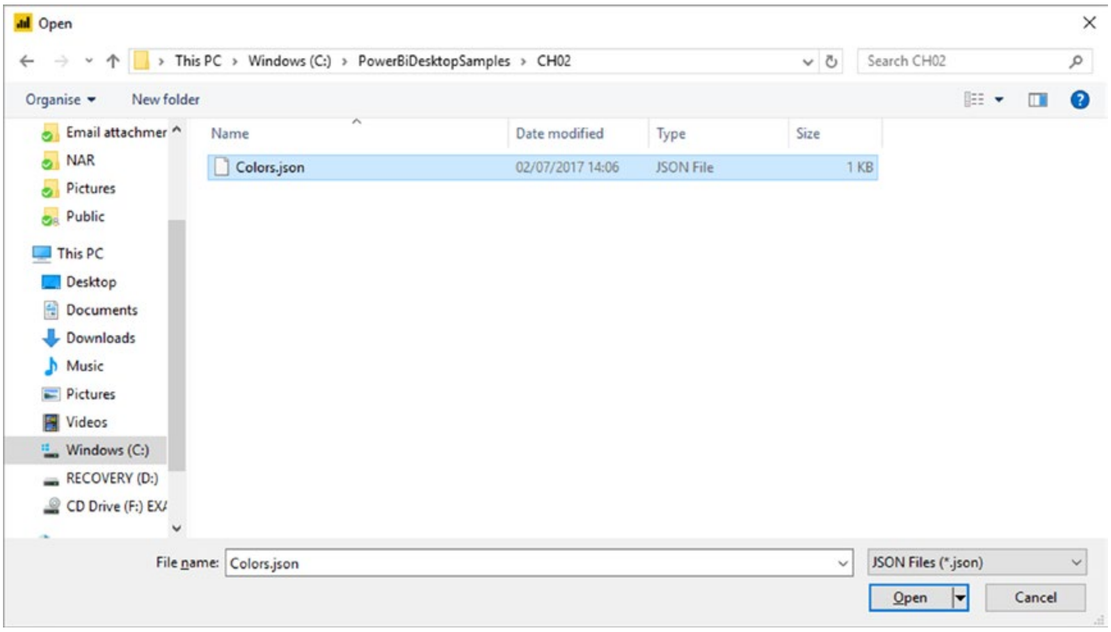


Figure 2-14. *Establishing a JSON connection*

2. Click Connect and navigate to the folder containing the JSON file that you want to load (C:\PowerBIDesktopSamples\CH02\Colors.json, in this example). You can see this in Figure 2-15.



Name	Date modified	Type	Size
Colors.json	02/07/2017 14:06	JSON File	1 KB

File name: Colors.json
JSON Files (*.json)

Figure 2-15. *Opening a JSON file*

3. Click Open. The dialog will close and you will return to the Power BI Desktop window.
4. Click the Apply Changes button at the top of the Power BI Desktop window.

You will not yet see the data in Power BI Desktop. However, you will have established a connection to the JSON file that you can later convert into the underlying data. You will learn the next steps that are required to load JSON data in Chapter 8.

Loading Multiple Files from a Directory

On many occasions you could find yourself faced with a set of identical files that have to be loaded to make up a complete dataset. They could be a file for each day's data, for instance, exported from an older system. As no one wants to load 365 files individually, the Power BI Desktop development team has come up with an answer to this kind of challenge. Provided that you gather all the source files into a single directory, you can load all the files at once.

■ **Note** This approach requires that all the files in the source folder share an identical format. If this is not the case, you will probably not succeed in loading the data.

1. Open a new Power BI Desktop file.
2. Click Get Data ➤ File, and select Folder from the options on the right of the dialog. The Get Data dialog should look like the one shown in Figure 2-16.

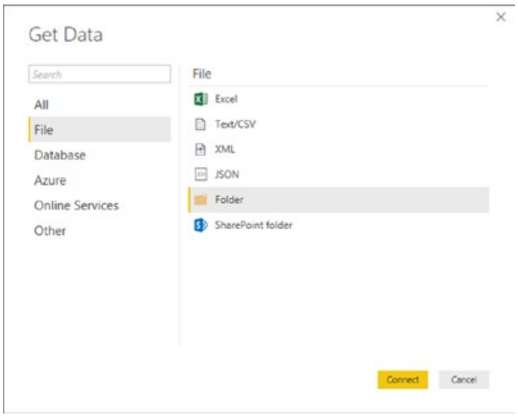
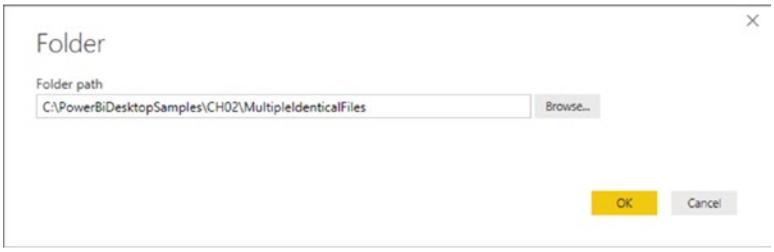


***Figure 2-16.** The Get Data dialog used to select multiple files from a folder*

3. Click Connect. The Folder dialog will appear.
4. Click the Browse button and navigate to the folder containing several identically structured text files (C:\PowerBIDesktopSamples\CH02\MultipleIdenticalFiles, in this example). The Folder dialog will look like the one shown in Figure 2-17.

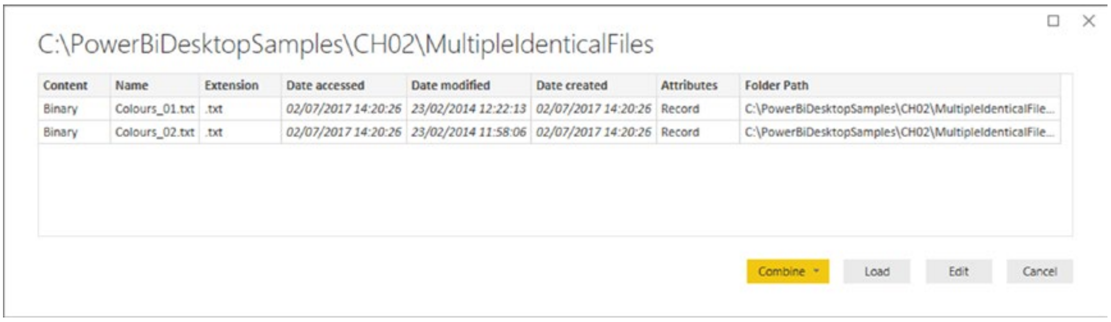

Folder

Folder path

C:\PowerBiDesktopSamples\CH02\MultipleIdenticalFiles

***Figure 2-17.** Selecting a folder*

5. Click OK. The contents of the folder will be displayed as you can see in Figure 2-18.

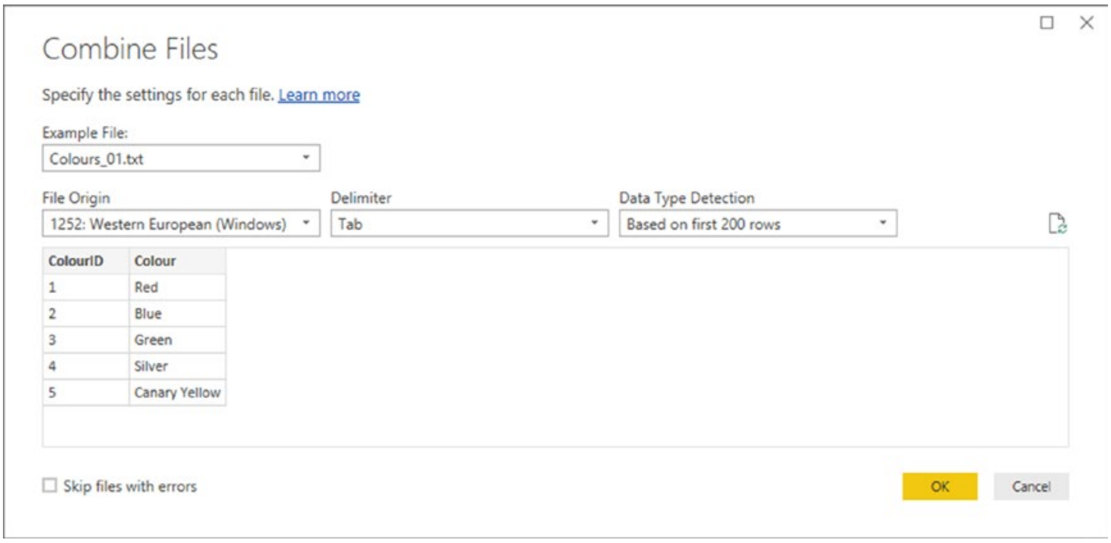

C:\PowerBiDesktopSamples\CH02\MultipleIdenticalFiles

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
Binary	Colours_01.txt	.txt	*02/07/2017 14:20:26*	*23/02/2014 12:22:13*	*02/07/2017 14:20:26*	Record	C:\PowerBiDesktopSamples\CH02\MultipleIdenticalFile...
Binary	Colours_02.txt	.txt	*02/07/2017 14:20:26*	*23/02/2014 11:58:06*	*02/07/2017 14:20:26*	Record	C:\PowerBiDesktopSamples\CH02\MultipleIdenticalFile...

***Figure 2-18.** The contents of the selected folder*

6. Click Combine ➤ Combine and Load. The Combine Files dialog will appear.

7. Select one of the source files to serve as a model for the structure of all the files to load. The dialog should look like the one shown in Figure 2-19.


Combine Files

Specify the settings for each file. Learn more

Example File: Colours_01.txt

File Origin: 1252: Western European (Windows)

Delimiter: Tab

Data Type Detection: Based on first 200 rows

ColourID	Colour
1	Red
2	Blue
3	Green
4	Silver
5	Canary Yellow

☐ Skip files with errors

Figure 2-19. *The Combine Files dialog*

8. Click OK. The data from all the files in the folder will be loaded into Power BI Desktop.

If you carry out this operation, you will see that an extra column will be added to the Fields list containing the name of the source file for each record.

■ **Note** This technique presumes that all the files in the source folder will share an identical format. If you choose, you can check the Skip Files With Errors check box in step 7. This will exclude any files that do not share the same format as the file that you specified as being the example file.

The Combine Files dialog gives you a few options that are similar to those that you saw previously when loading text/CSV files. They include

- Specifying the delimiter (separator) for fields in the dataset
- Specifying how much, if any, of the model file is used to detect the file structure
- Defining the character set (the encoding) for the source files

Loading the Contents of a Folder

Rather than loading all the files in a folder, you can choose to connect to a folder and load only the essential information about the available files. You can use this connection to load the files themselves later, should you want to. To do this:

1. Open a new Power BI Desktop file.
2. Click Get Data ➤ File, and select Folder from the options on the right of the dialog.
3. Click Connect. The Folder dialog will appear.
4. Navigate to the folder containing several identically structured text files (C:\PowerBIDesktopSamples\CH02\MultipleIdenticalFiles, in this example).
5. Click OK. The contents of the folder will be displayed.
6. Click Load. A series of predefined fields will appear in the Fields list, as shown in Figure 2-20.

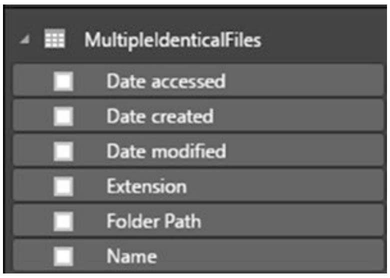


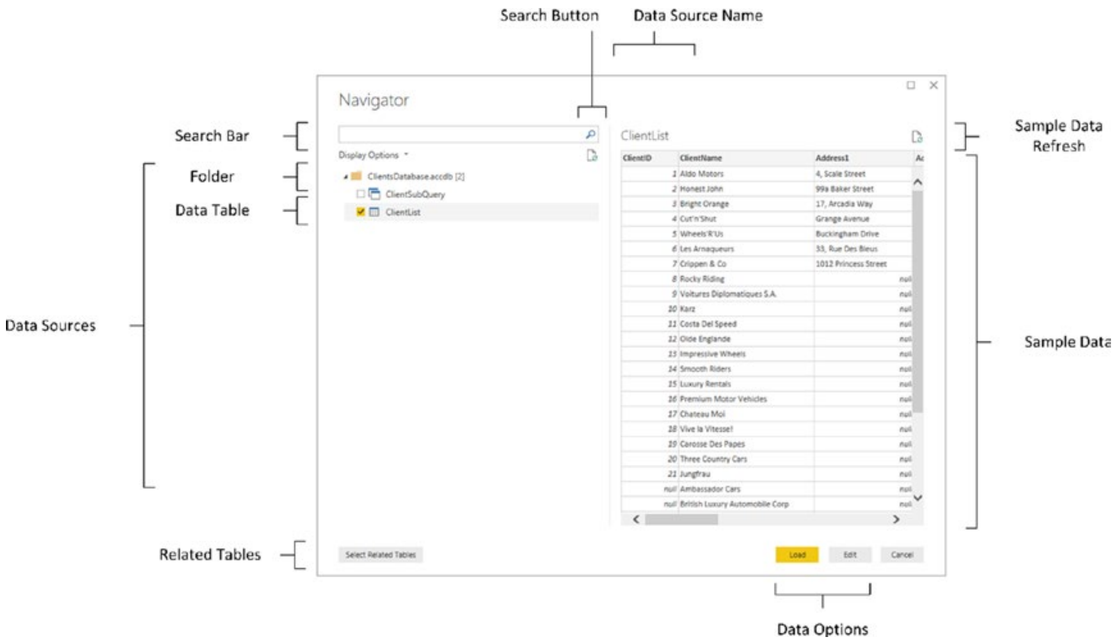
Figure 2-20. *Data for folder contents in Power BI Desktop*

You can now use this information either to display the contents of the folder in Power BI Desktop or (and this is where things get more fun) to select the files in the folder and load any or all of them. You will be seeing this technique in Chapter 8.

The Navigator Dialog

The more you work with Power BI Desktop, the more you will use the Navigator dialog. So it seems appropriate to explain at this early juncture some of the tricks and techniques that you can apply to make your life easier when delving into data sources.

Let's start by taking a closer look at the available options. These are highlighted in Figure 2-21.



ClientList

ClientID	ClientName	Address1
1	Aldo Motors	4, Scale Street
2	Honest John	99a Baker Street
3	Bright Orange	17, Arcadia Way
4	Cut'n'Shut	Grange Avenue
5	Wheels'R'Us	Buckingham Drive
6	Les Arnaqueurs	33, Rue Des Bleus
7	Crippen & Co	1012 Princess Street
8	Rocky Riding	null
9	Voitures Diplomatiques S.A.	null
10	Karz	null
11	Costa Del Speed	null
12	Olde Englande	null
13	Impressive Wheels	null
14	Smooth Riders	null
15	Luxury Rentals	null
16	Premium Motor Vehicles	null
17	Chateau Moi	null
18	Vive la Vitesse!	null
19	Carosse Des Papes	null
20	Three Country Cars	null
21	Jungfrau	null
null	Ambassador Cars	null
null	British Luxury Automobile Corp	null

Figure 2-21. *The Navigator dialog*

The Navigator dialog is essentially in two parts:

- On the left: The hierarchy of available data sources. These can consist of a single dataset or multiple datasets, possibly organized into one or many folders.
- On the right: A preview of the data in the selected element.

Searching for Datasets

There will, inevitably, be cases where the data source that you are connecting to will contain hundreds of datasets. This is especially true for databases. Fortunately, Power BI Desktop lets you filter the datasets that are displayed extremely easily.

1. In the Navigator dialog, click inside the Search box.
2. Enter a part of a dataset name that you want to isolate.
3. Click the magnifying glass icon at the right of the Search box. The list of datasets will be filtered to show only datasets containing the text that you entered. You can see this in Figure 2-22.

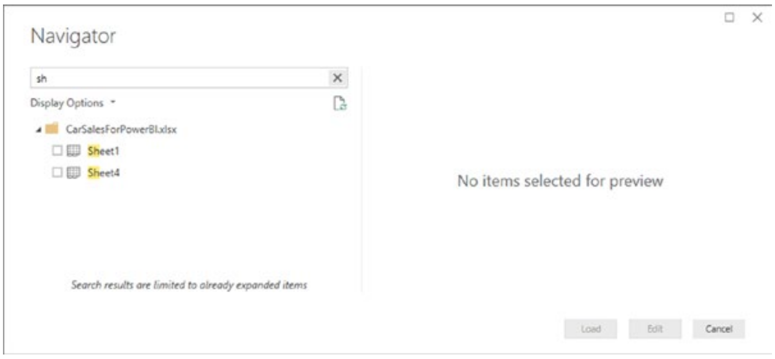

Navigator
sh
Display Options
CarSalesForPowerBI.xlsx
Sheet1
Sheet4
No items selected for preview
Search results are limited to already expanded items
Load
Edit
Cancel

Figure 2-22. *Dataset search in the Navigator dialog*

Once you have previewed and selected the datasets that you want to use, simply click the cross at the right of the Search box. Navigator will clear the filter and display all the datasets in the data source.

Display Options

Clicking Display Options will show a popup menu with two options:

- Only selected items
- Enable data previews

Only Selected Items

Selecting this option will prevent any datasets that you have not selected from appearing in the data source pane.

Enable Data Previews

Selecting this option will show a small subset of the data available in the selected dataset. You could choose to disable data previews if the connection to the source data is slow.

Refresh

If you need to, you can refresh either or both of the following:

- The source data
- The data preview

Source Data Refresh

Clicking the preview button under the search bar will refresh the source data in the source data pane.

Data Preview Refresh

Clicking the preview button on the top right of the Navigator dialog will refresh the preview data visible on the right.

Select Related Tables

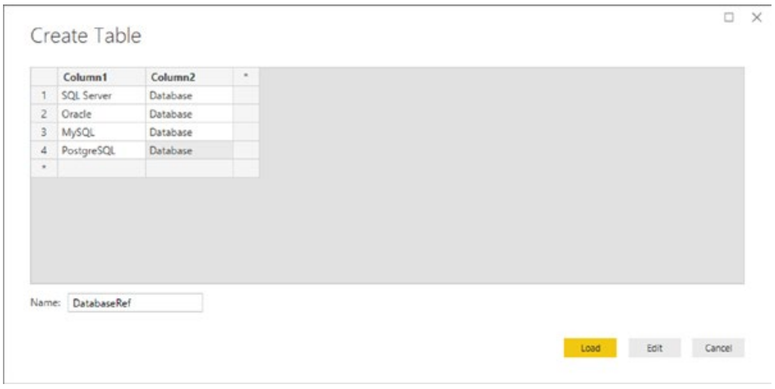
Clicking the Select Related Tables button is only valid for database sources, such as Microsoft Access or Oracle, for instance. If the source database has been designed correctly to include joins between tables, then this option will automatically select all tables that are linked to any tables that you have already selected.

Adding Your Own Data

All the data you need may not be always available. You might find yourself needing to add a list of products, a group of people, or, indeed, any kind of data to the datasets that you have loaded into Power BI Desktop.

The development team at Microsoft has recognized this need, and offers a simple solution: you can create your own tables of data to complete the collection of datasets in a Power BI in-memory data model. Then you can enter any extra data that you need, on the fly.

1. In the Power BI Desktop Home ribbon, click Enter Data. The Create Table dialog will appear.
2. Click the asterisk to the right of Column1 to add a column.
3. Enter the data that you need. The dialog will look like the one shown in Figure 2-23.


Create Table

	Column1	Column2	*
1	SQL Server	Database	
2	Oracle	Database	
3	MySQL	Database	
4	PostgreSQL	Database	
*			

Name: DatabaseRef

Figure 2-23. *The Create Table dialog*

4. Enter a name for the table in the Name field at the bottom of the dialog.
5. Click Load to load the data into the Power BI in-memory data model.

Editing facilities in the Create Table dialog are extremely simplistic. You can delete, cut, copy, and paste data and columns, but that is about all that you can do. However, this option can, nonetheless, be extremely useful when you need to add some last-minute data to a model.

■ **Note** This is an extremely simple process that is designed for small amounts of data. If you need more than just a handful of rows and columns, you could be better served by creating the data in Excel and then loading it into Power BI Desktop.

Conclusion

In this chapter, you have seen how this powerful addition to the Microsoft business intelligence toolset, Power BI Desktop, can help you find and load data from a variety of File-based data sources. These sources can be Access, Excel, CSV, XML, JSON, or text files—or they could come from entire folders of identically structured text/CSV files.

I am sure that you can see a pattern emerging in the course of this chapter. Indeed, this pattern will continue as you progress to loading tables from relational databases in Chapter 3. The process is nearly always

1. Know the type of source data that you want to look at.
2. Find the file that lets you access the data.
3. Examine the data and select the elements that you want to load.

You have seen that Power BI Desktop will let you see a sample of the contents of the data sources that it can read without needing any other application. This makes it a superb tool for peeking into data sources and deciding if a file actually contains the data that you need. Indeed, Power BI Desktop's Navigator can help you filter multiple datasets in XML files or Access databases, preview each dataset, and only select the ones that you want to load. Of course, it can also load dozens of datasets at once if they all are stored in the same source.

However, file-based data sources are only a small part of the picture. Power BI Desktop can also load data from a wide range of relational databases and data warehouses. We will take a look at some of these in the next chapter.

CHAPTER 3

Loading Data from Databases and Data Warehouses

Much of the world's corporate data currently resides in relational databases, data warehouses, and data warehouse appliances. Power BI Desktop can connect to many (if not most) of the world's leading databases and data warehouses. Not only that, but it can also connect to many of the lesser-known or more niche data sources that are currently available. This chapter will show you how to extract data from several of these data sources to power your analytics using Power BI Desktop. Indeed, you will discover that once you have learned how to connect to one or two databases, you have learned how to use nearly all of them, thanks to the standardized approach that Power BI Desktop brings to data extraction.

Once again, in this chapter I will be using a set of example files that you can find on the Apress web site. If you have followed the instructions in Appendix A, then these files will be in the C:\PowerBIDesktopSamples\CH03 folder. Be aware, however, that not all the examples in this chapter use sample data that is available on the Apress web site. In some cases, such as Oracle and FileMaker Pro, I will let you load your own sample data, or use the sample data that can be installed with the databases themselves.

Relational Databases

Being able to access the data stored in relational databases is essential for much of today's business intelligence. As enterprise-grade relational databases still hold much of the world's data, you really need to know how to tap into the vast mines of information that they contain. The bad news is that there are many, many databases out there, each with its own intricacies and quirks. The good news is that once you have learned to load data from *one* of them, you can reasonably expect to be able to use *any* of them.

In the real world, connecting to corporate data could require you to have a logon name and possibly a password that will let you connect (unless the database can recognize your Windows login). I imagine that you will also require permissions to read the tables and views that contain the data. So the techniques described here are probably the easy bit. The hard part is convincing the guardians of corporate data that you actually *need* the data and you should be allowed to see it.

The databases that Power BI Desktop can currently connect to, and can preview and load data from, are given in Table 3-1.

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_3

Table 3-1. *Database Sources*

Database	Comments
SQL Server database	Lets you connect to a Microsoft SQL Server on-premises database and import records from all the data tables and views that you are authorized to access.
Access database	Lets you connect to a Microsoft Access file on your network and load queries and tables.
SQL Server Analysis Services database	Lets you connect to a SQL Server Analysis Services (SSAS) data warehouse. This can be either an online analytical processing (OLAP) cube or an in-memory tabular data warehouse.
Oracle database	Lets you connect to an Oracle database and import records from all the data tables and views that you are authorized to access.
IBM DB2 database	Lets you connect to an IBM DB2 database and import records from all the data tables and views that you are authorized to access.
IBM Informix database	Lets you connect to an IBM Informix database and import records from all the data tables and views that you are authorized to access.
IBM Netezza	Lets you connect to an IBM Netezza data warehouse appliance and import records from all the data tables that you are authorized to access.
MySQL database	Lets you connect to a MySQL database and import records from all the data tables and views that you are authorized to access.
PostgreSQL database	Lets you connect to a PostgreSQL database and import records from all the data tables and views that you are authorized to access.
Sybase database	Lets you connect to a Sybase database and import records from all the data tables and views that you are authorized to access.
Teradata database	Lets you connect to a Teradata database and import records from all the data tables and views that you are authorized to access.
SAP HANA database	Lets you connect to a SAP HANA in-memory database and import records from all the objects that you have permission to access.
Amazon Redshift	Lets you connect to an Amazon Redshift database and import records from all the data tables and views that you are authorized to access.
Impala	Lets you connect to an Impala database and import records from all the data tables and views that you are authorized to access.

■ **Note** As the list of database and data warehouse sources that you can connect to from Power BI Desktop continues to evolve, this list could see several new items by the time that you read this book.

As well as connections for specific databases, Power BI Desktop contains generic connectors that can help you to read data from databases that are not specifically in the list of available databases. These generic connectors are explained in Table 3-2.

Table 3-2. *Generic Database Access*

Source	Comments
ODBC data source	Lets you connect over Open Database Connectivity to a database or data source.
OLE DB data source	Lets you connect over Object Linking and Embedding, Database to a database or data source.

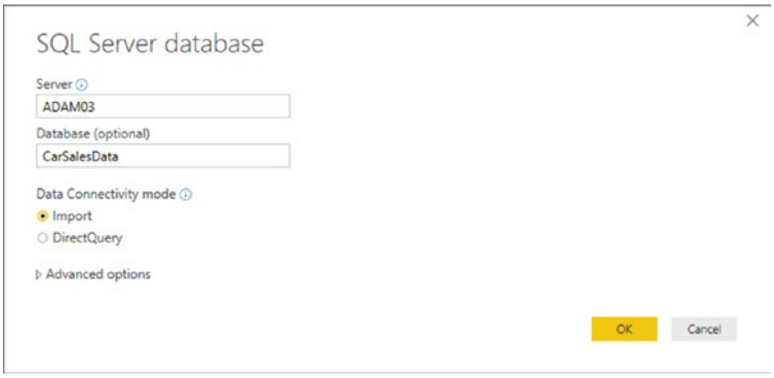
Be warned that these generic connectors will not work with any database. However, they should work with a database for which you have procured, installed, and configured a valid ODBC or OLE DB driver.

■ **Note** Although Power BI Desktop classifies Microsoft Access as a relational database, I prefer to handle it as a file-based source. For this reason, MS Access data was discussed in the previous chapter.

SQL Server

Here I will use the Microsoft enterprise relational database—SQL Server—as an example to show you how to load data from a database into Power BI Desktop. The first advantage of this setup is that you probably do not need to install any software to enable access to SQL Server (although this is not always the case, so talk this through with your IT department). A second advantage is that the techniques are pretty similar to those used and applied by Oracle, DB2, and the other databases to which Power BI Desktop can connect. Furthermore, you can load multiple tables or views from a database at once. To see this in action (and presuming that you have created the database CarSalesData as described in Appendix A), take the following steps:

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
3. Enter the server name in the Server text box. This will be the name of your SQL Server or one of the SQL Server resources used by your organization.
4. Enter the database name; if you are using the sample data, it will be CarSalesData.
5. Click the Import button. The dialog will look like Figure 3-1.


SQL Server database

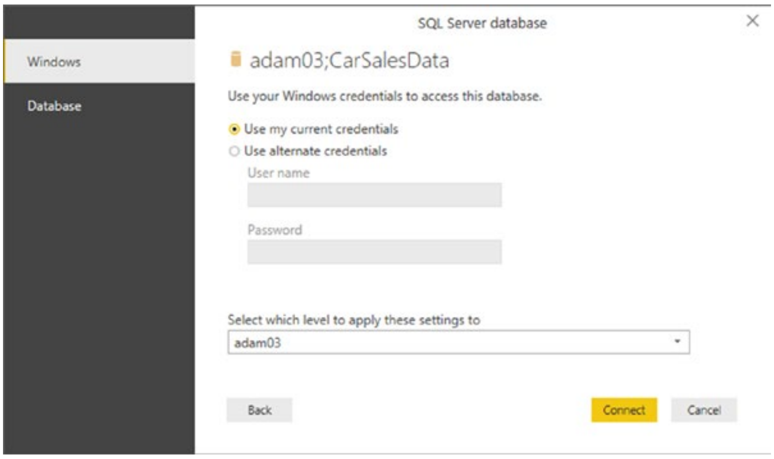
Server: ADAM03

Database (optional): CarSalesData

Data Connectivity mode: [✓ Import] DirectQuery

Figure 3-1. *The Microsoft SQL Server Database dialog*

6. Click OK. The SQL Server Database dialog will appear. Assuming that you are authorized to use your Windows login to connect to the database, leave "Use my current credentials" selected, as shown in Figure 3-2.


SQL Server database

adam03;CarSalesData

Use your Windows credentials to access this database.

[✓ Use my current credentials] Use alternate credentials

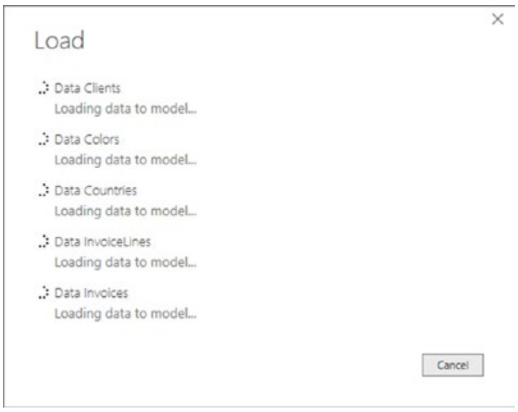
User name:

Password:

Select which level to apply these settings to: adam03

Figure 3-2. *The credentials Database dialog*

7. Click Connect. While the data is being loaded, Power BI Desktop will display the Load dialog and show the load progress for each selected table. You can see this in Figure 3-3.


Load

Data Clients
Loading data to model...

Data Colors
Loading data to model...

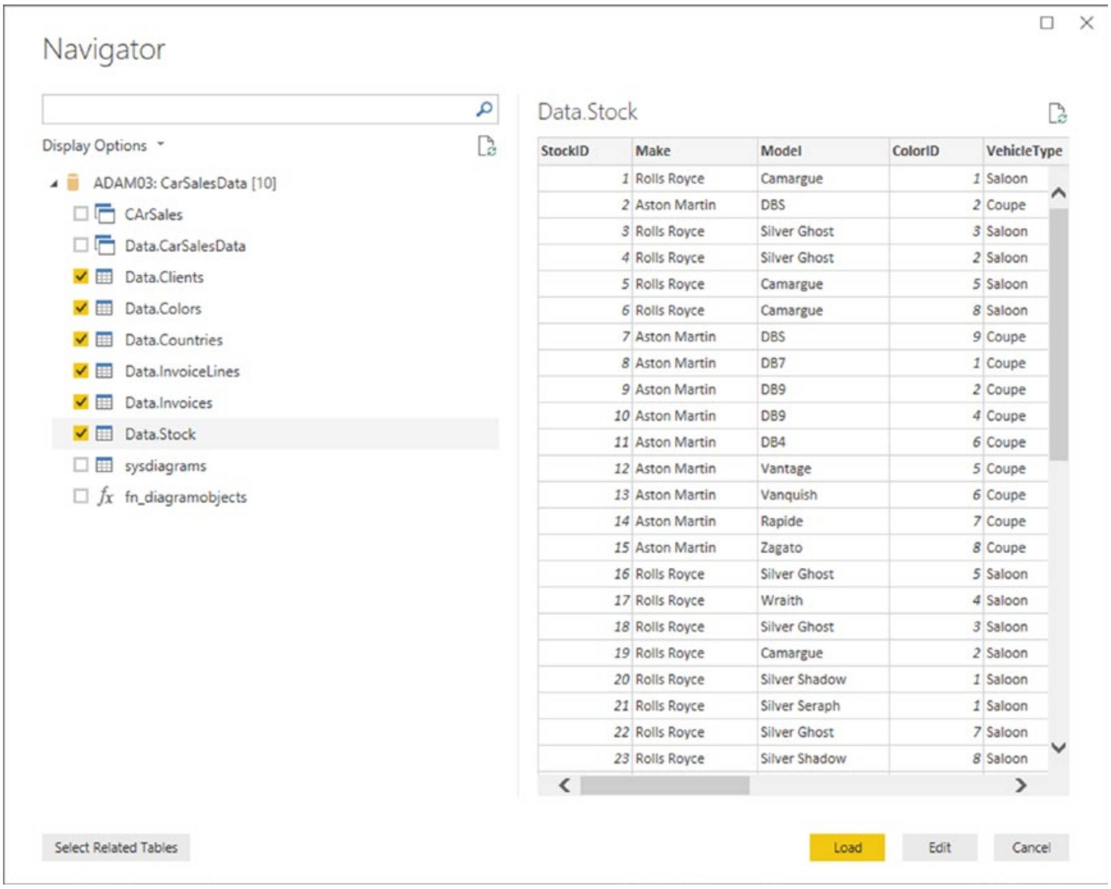
Data Countries
Loading data to model...

Data InvoiceLines
Loading data to model...

Data Invoices
Loading data to model...

Figure 3-3. *The Load dialog displaying data load progress*

8. Power BI Desktop will connect to the server and display the Navigator dialog containing all the tables and views in the database that you have permission to see on the server you selected. In some cases, you could see a dialog saying that the data source does not support encryption. If you feel happy with an unencrypted connection, then click the OK button for this dialog.
9. Click the check boxes for the Clients, Colors, Countries, Invoices, InvoiceLines, and Stock tables. The data for the most recently selected dataset appears on the right of the Navigator dialog, as shown in Figure 3-4.



ADAM03: CarSalesData [10]

- ☐ CArSales
- ☐ Data.CarSalesData
- ✓ Data.Clients
- ✓ Data.Colors
- ✓ Data.Countries
- ✓ Data.InvoiceLines
- ✓ Data.Invoices
- ✓ Data.Stock
- ☐ sysdiagrams
- ☐ fn_diagramobjects

Data.Stock

StockID	Make	Model	ColorID	VehicleType
1	Rolls Royce	Camargue	1	Saloon
2	Aston Martin	DBS	2	Coupe
3	Rolls Royce	Silver Ghost	3	Saloon
4	Rolls Royce	Silver Ghost	2	Saloon
5	Rolls Royce	Camargue	5	Saloon
6	Rolls Royce	Camargue	8	Saloon
7	Aston Martin	DBS	9	Coupe
8	Aston Martin	DB7	1	Coupe
9	Aston Martin	DB9	2	Coupe
10	Aston Martin	DB9	4	Coupe
11	Aston Martin	DB4	6	Coupe
12	Aston Martin	Vantage	5	Coupe
13	Aston Martin	Vanquish	6	Coupe
14	Aston Martin	Rapide	7	Coupe
15	Aston Martin	Zagato	8	Coupe
16	Rolls Royce	Silver Ghost	5	Saloon
17	Rolls Royce	Wraith	4	Saloon
18	Rolls Royce	Silver Ghost	3	Saloon
19	Rolls Royce	Camargue	2	Saloon
20	Rolls Royce	Silver Shadow	1	Saloon
21	Rolls Royce	Silver Seraph	1	Saloon
22	Rolls Royce	Silver Ghost	7	Saloon
23	Rolls Royce	Silver Shadow	8	Saloon

Figure 3-4. *The Navigator dialog when selecting multiple items*

10. Click Load.

11. The Power BI Desktop window will open and display the tables that you selected in the Fields list in the Report window when you click OK.

Since this is very similar to the way in which you loaded data from Access in the previous chapter, I imagine that you are getting the hang of how to use database sources by now. Once again the Navigator dialog is a simple and efficient way to select the datasets that you want to use in your reports and dashboards.

■ **Tip** When selecting multiple tables or views, you will only ever see the contents of a single data source in the Navigator dialog. However, you can preview the contents of any of the selected data sources (or even any that are not selected) simply by clicking the table or view name. This will not affect the choice of selected tables and views that you want to load into Power BI Desktop.

■ **Note** You can enter the server IP address instead of the server name if you prefer. If there are several SQL Server instances on the same server, you will need to add a backslash and the instance name. This kind of detailed information can be obtained from corporate database administrators.

Automatically Loading Related Tables

Relational databases are nearly always intricate structures composed of many interdependent tables. Indeed, you will frequently need to load several tables to obtain all the data that you need.

Knowing which tables to select is not always easy. Power BI Desktop tries to help you by automatically detecting the links that exist in the source database between tables; this way, you can rapidly isolate the collections of tables that have been designed to work together.

Do the following to see a related group of tables:

1. Connect to the source database as described in the previous section.
2. In the Navigator dialog, click a table that contains data that you need.
3. Click the "Select related tables" button.

Any tables that are linked in the database are selected. You can deselect any tables that you do not want, of course. More importantly, you can click the names of the selected tables to see their contents.

■ **Note** Sometimes you have to select several tables in turn and click "Select related tables" to ensure that Power BI Desktop will select all the tables that are necessary to underpin your analysis.

Database Options

The world of relational databases is—fortunately or unfortunately—a little more complex than the world of files or MS Access. Consequently, there are a few comments to make about using databases as a data source; specifically, how to connect to them.

First, let's cover the initial connection to the server. The options are explained in Table 3-3.

Table 3-3. *Database Connection Options*

Option	Comments
Server	You cannot browse to find the server and you need to type or paste the server name. If the server has an instance name, you need to enter the server and the instance. Your IT department will be able to supply this if you are working in a corporate environment.
Database	If you know the database, then you can enter (or paste) it here. This restricts the number of available tables in the Navigator dialog and makes finding the correct table or view easier.
SQL statement	You can enter a valid snippet of T-SQL that returns data from the database.

These options probably require a little more explanation. So let's look at each one in turn.

Server Connection

It is fundamental that you know the exact connection string for the database that you want to connect to. This could be the following:

- The database server name.
- The database server name, a backslash, and an instance name (if there is one).
- The database server IP address.
- The database server IP address, a backslash, and an instance name (if there is one).
- If the SQL Server instance is using a custom port, you must end the server name with a comma followed by the port number. This is, inevitably, a question for corporate DBAs.
- If you are running a single SQL Server instance on your own PC, then you can use the name *localhost* to refer to the server.

■ **Note** A database instance is a separate SQL Server service running alongside others on the same physical or virtual server. You will always need both the server and this instance name (if there is one) to successfully connect. You can also specify a timeout period if you wish.

Most SQL Server instances host many, many databases. Sometimes these can number in the hundreds. Sometimes, inevitably, you cannot remember which database you want to connect to. Fortunately, Power BI Desktop can let you browse the databases on a server. To do this, do the following:

1. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
2. Enter the server name in the Server text box and click OK. Do not enter a database name. The Navigator window opens and displays all the available databases, as shown in Figure 3-5. Of course, the actual contents depend on the server that you are connecting to.

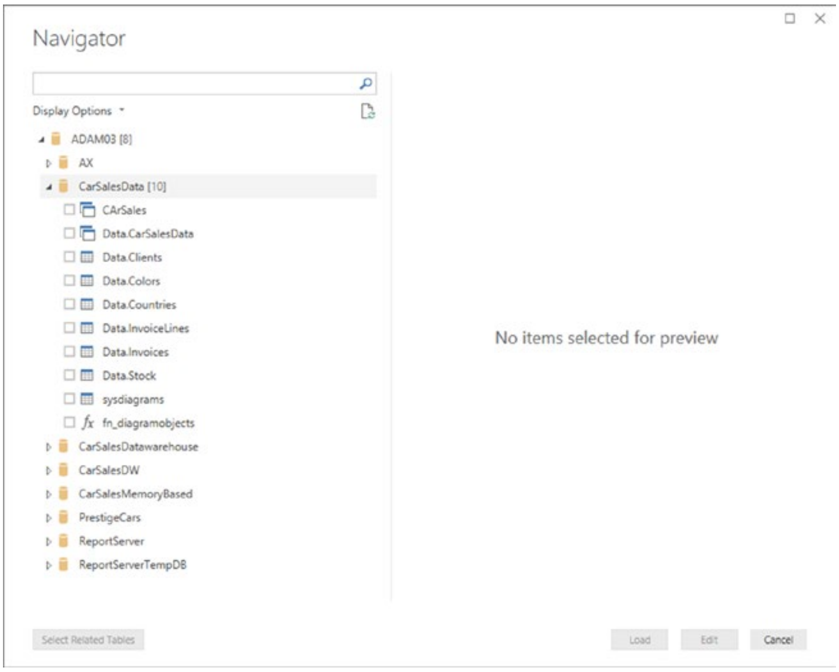


Figure 3-5. *The Navigator dialog when selecting databases*

You can see from Figure 3-5 that if you click the small triangle to the left of a database, then you are able to see all the tables and views that are accessible to you in this database. Although this can mean an overabundance of possible choices when looking for the table(s) or view(s) that you want, it is nonetheless a convenient way of reminding you of the name of dataset that you require.

■ **Tip** The actual databases that you will be able to see on a corporate server will depend on the permissions that you have been given. If you cannot see a database, then you will have to talk to the database administrators to sort out any permissions issues.

Searching for Databases, Tables, and Views in Navigator

If you are overwhelmed by the sheer volume of table(s) and view(s) that appear in the left panel of the Navigator dialog, then you can use Navigator's built-in search facility to help you to narrow down the set of potential data sources.

Searching for Databases

To isolate specific databases, do the following:

1. Carry out steps 1 and 2 in the earlier "SQL Server" section to connect to a SQL Server instance *without* specifying a database.

2. In the Search box of the Navigator dialog, enter a few characters that you know are contained in the name of the table or view that you are looking for. Entering, for example, **data** on my server gives the result that you see in Figure 3-6.

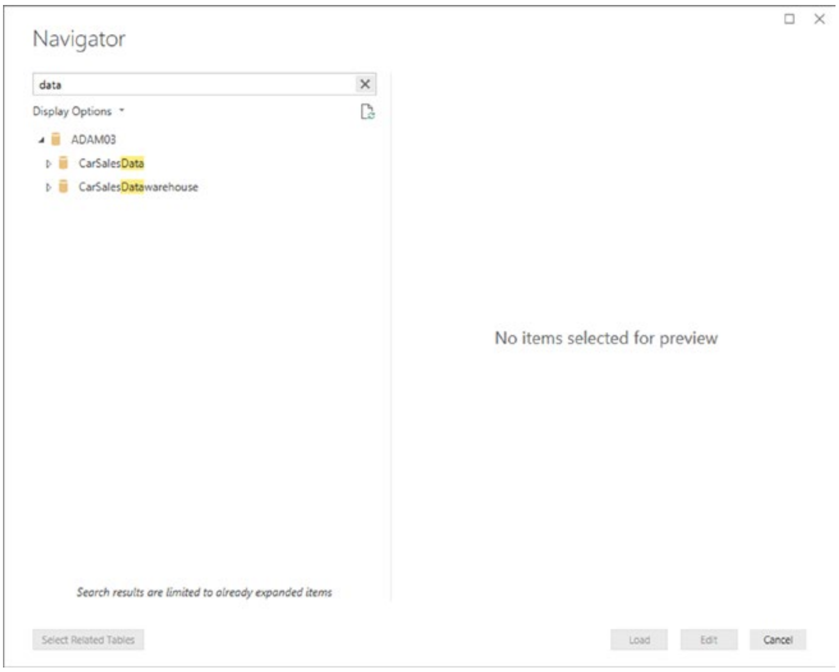


Figure 3-6. *Using Search with Navigator to find databases*

Searching for Tables

If you are searching for tables, do the following:

1. Expand any databases that you want to search for specific tables.

2. In the Search box of the Navigator dialog, enter a few characters that you know are contained in the name of the table or view that you are looking for. Entering, for example, **cli** on my server gives the result that you see in Figure 3-7.

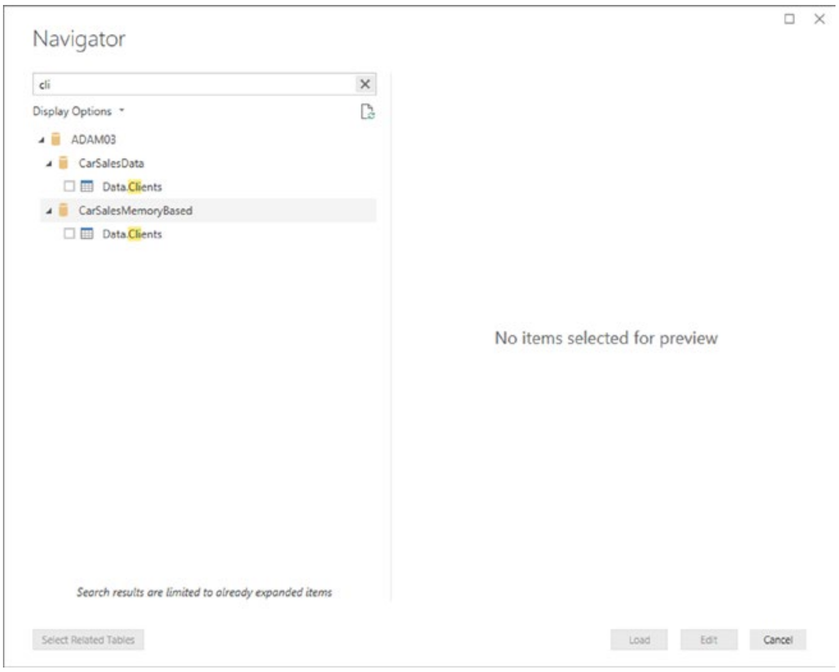


Figure 3-7. *Using Search with Navigator to find tables*

When searching for objects, you can enter the text in uppercase or lowercase, and the text can appear anywhere in the names of the tables or views—not just at the start of the name. With every character that you type, the list of potential matches gets shorter and shorter. Once you have found the dataset that you are looking for, simply proceed as described earlier to load the data into Power BI Desktop.

If your search does not return the subset of tables in any views that you were expecting, all you have to do is click the cross at the right of the Search box. This cancels the search and displays all the available tables, as well as clears the Search box.

If you are not convinced that you are seeing all the tables and views that are in the database, then click the small icon at the bottom right of the Search box (it looks like a small page with two green circular arrows). This is the Refresh button, which refreshes the connection to the database and displays all the tables and views that you have permission to see. Finally, it is worth noting that filtering database tables is very similar to searching for Excel objects—a technique that you saw in the previous chapter.

■ **Note** SQL Server databases can also be accessed using the DirectQuery option. This technique is explained in the next chapter.

Database Security

Remember that databases are designed to be extremely secure. Consequently, you only see servers, databases, tables, and views if you are authorized to access them. You might have to talk to your IT department to ensure that you have the required permissions; otherwise, the table that you are looking for could be in the database, but remain invisible to you.

■ **Tip** If you experience a connection error when first attempting to connect to SQL Server, simply click the Edit button to return to the Microsoft SQL Database dialog and correct any mistakes. This avoids having to start over.

Using a SQL Statement

If there is a downside to using a relational database such as SQL Server as a data source, it is that the sheer amount of data that the database stores—even in a single table—can be dauntingly huge. Fortunately, all the resources of SQL Server can be used to filter the data that is used by Power BI Desktop before you even load the data. This way, you do not have to load entire tables of data at the risk of drowning in information before you have even started to analyze it.

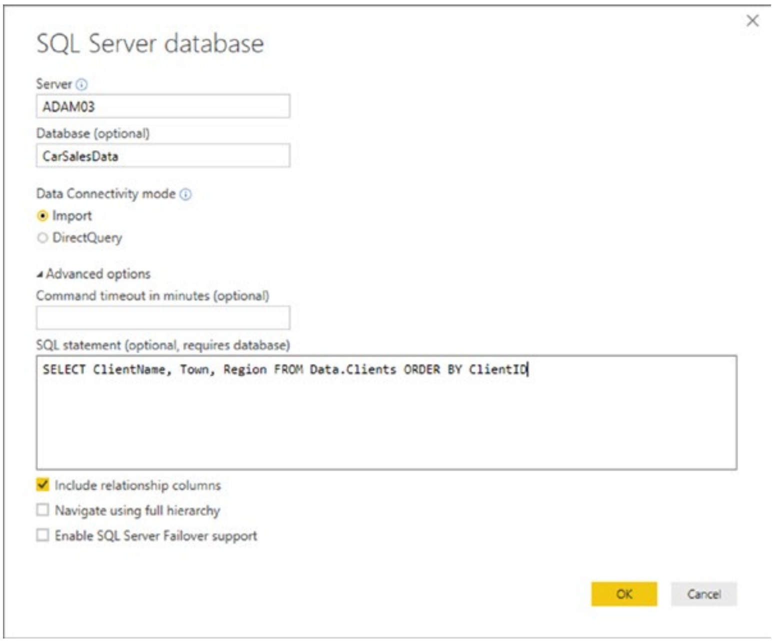
The following are SQL Server techniques that you can use to extend the partnership between SQL Server and Power BI Desktop:

- SQL SELECT statements
- Stored procedures
- Table-valued functions

These are, admittedly, fairly technical solutions. Indeed, if you are not a database specialist, you could well require the services of your IT department to use these options to access data in the server. Nonetheless, it is worth taking a quick look at these techniques in case they are useful one day.

Any of these options can be applied from the SQL Server Database dialog. Here is an example of how to filter data from a database table using a SELECT statement:

1. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
2. Enter the server name and the database.
3. Click the triangle to the left of SQL Statement (optional). This opens a box where you can enter a SQL command.
4. Enter the SQL command that you want to apply. In this case, it is `SELECT ClientName, Town, Region FROM Data.Clients ORDER BY ClientID`. The dialog will look like Figure 3-8.


SQL Server database

Server: ADAM03

Database (optional): CarSalesData

Data Connectivity mode: ● Import ○ DirectQuery

Advanced options

Command timeout in minutes (optional):

SQL statement (optional, requires database):

```
SELECT ClientName, Town, Region FROM Data.Clients ORDER BY ClientID
```

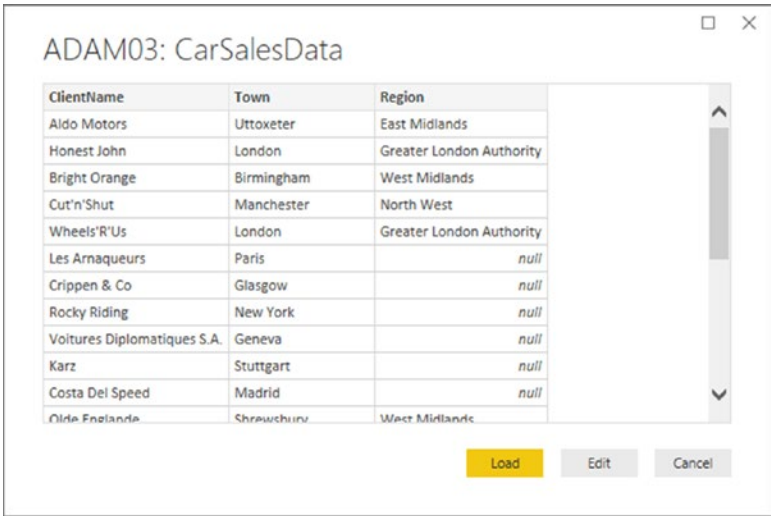
☑ Include relationship columns

☐ Navigate using full hierarchy

☐ Enable SQL Server Failover support

Figure 3-8. *Using SQL to select database data*

5. Click OK. A sample of the corresponding data is displayed in a dialog like the one shown in Figure 3-9.


ADAM03: CarSalesData

ClientName	Town	Region
Aldo Motors	Uttoxeter	East Midlands
Honest John	London	Greater London Authority
Bright Orange	Birmingham	West Midlands
Cut'n'Shut	Manchester	North West
Wheels'R'Us	London	Greater London Authority
Les Arnaqueurs	Paris	*null*
Crippen & Co	Glasgow	*null*
Rocky Riding	New York	*null*
Voitures Diplomatiques S.A.	Geneva	*null*
Karz	Stuttgart	*null*
Costa Del Speed	Madrid	*null*
Olde Englande	Shrewsbury	West Midlands

Figure 3-9. *Database data selected using the SQL Statement option*

6. Click Load or Edit to continue with the data load process. Alternatively, you can click Cancel and start a different data load.

■ **Tip** When entering custom SQL (or when using stored procedures, as is explained in the following section) you should, preferably, specify the database name in step 3. If you do not give the database name, you will have to use three-part notation in your SQL query. That is, you must add the database name and a period before the schema and table name of *every* table name used in the query.

Stored Procedures in SQL Server

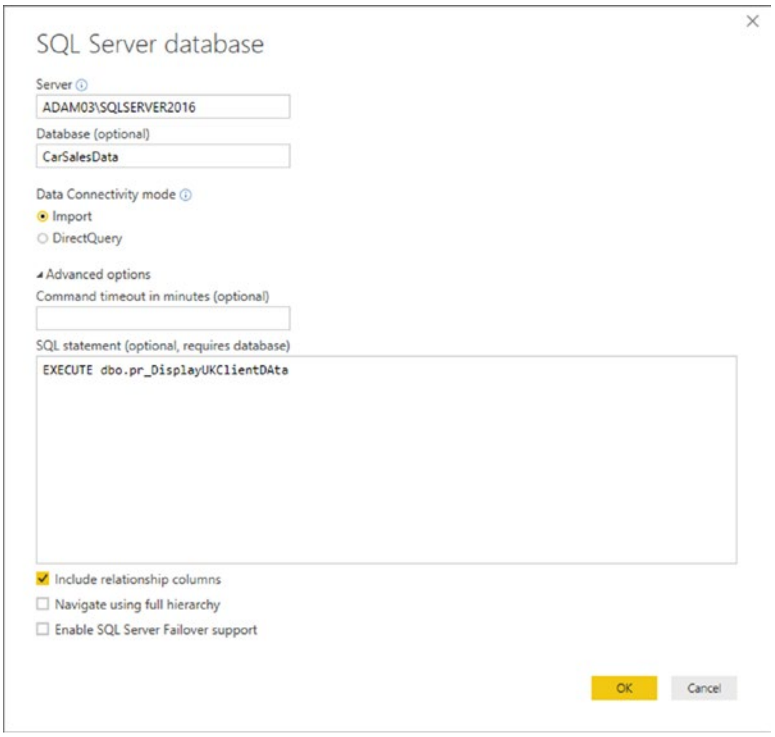
The same principles apply when using stored procedures of functions to return data from SQL Server. You will always use the SQL Statement option to enter the command that will return the data. Just remember that to call a SQL Server stored procedure or function, you would enter the following elements into the Microsoft SQL Database dialog:

- Server: <your server name>
- Database: <the database name>
- SQL Statement: EXECUTE (or EXEC) <enter the schema (if there is one, followed by a period) and the stored procedure name, followed by any parameters>

This way, either you or your IT department can create complex and secure ways to allow data from the corporate databases to be read into Power BI Desktop from enterprise databases.

To see this in practice, you can use a SQL Server stored procedure that is included in the sample SQL Server database to return only a subset of the available data. The stored procedure is called pr_DisplayUKClientData, and you apply it like this:

1. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
2. Enter the server name and the database.
3. Click the triangle to the left of SQL Statement (optional). This opens a box where you can enter a SQL command.
4. Enter the SQL command that you want to apply. In this case, it is `EXECUTE dbo.pr_DisplayUKClientData`. The dialog will look like Figure 3-10.


SQL Server database

Server: ADAM03\SQLSERVER2016

Database (optional): CarSalesData

Data Connectivity mode
- [✓ Import]
- DirectQuery

Advanced options

Command timeout in minutes (optional):

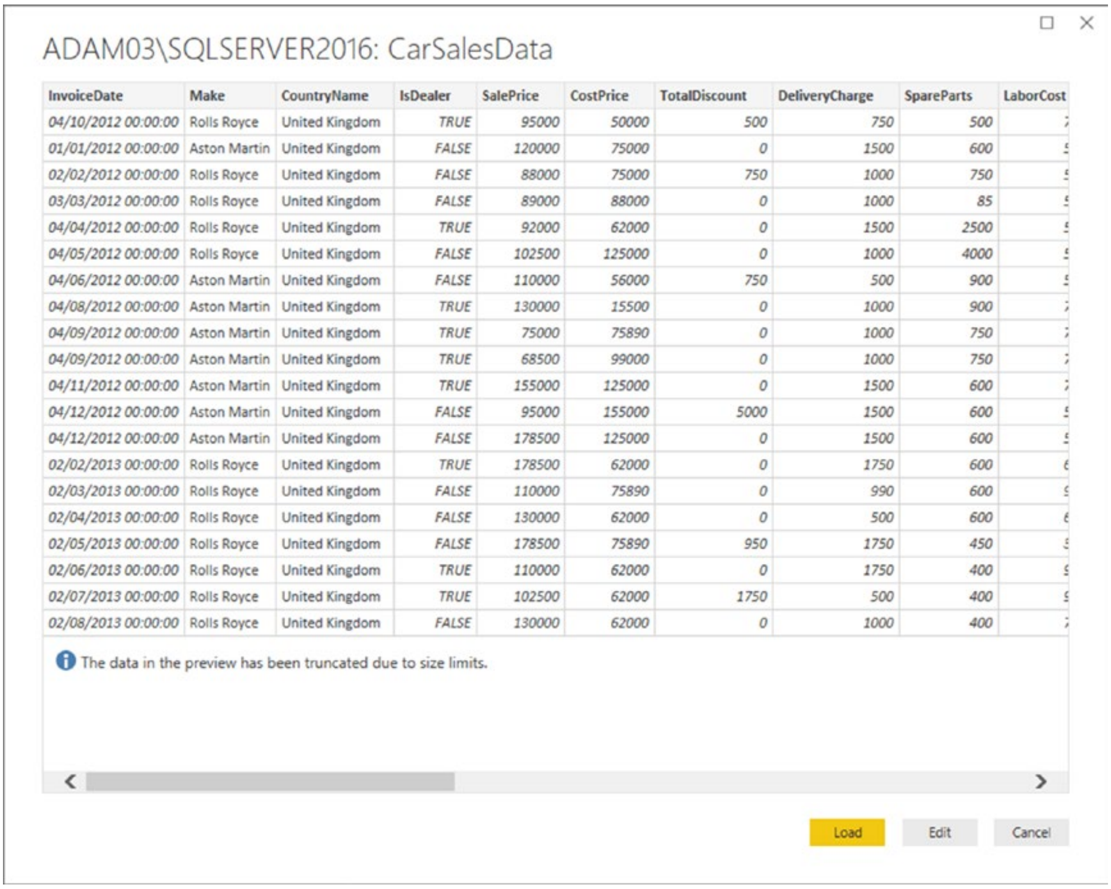
SQL statement (optional, requires database):

```
EXECUTE dbo.pr_DisplayUKClientDAta
```

- [✓] Include relationship columns
- [ ] Navigate using full hierarchy
- [ ] Enable SQL Server Failover support

Figure 3-10. *Using SQL to select database data*

5. Click OK. A sample of the corresponding data is displayed in a dialog like the one shown in Figure 3-11.



ADAM03\SQLSERVER2016: CarSalesData

InvoiceDate	Make	CountryName	IsDealer	SalePrice	CostPrice	TotalDiscount	DeliveryCharge	SpareParts	LaborCost
04/10/2012 00:00:00	Rolls Royce	United Kingdom	TRUE	95000	50000	500	750	500	[illegible]
01/01/2012 00:00:00	Aston Martin	United Kingdom	FALSE	120000	75000	0	1500	600	[illegible]
02/02/2012 00:00:00	Rolls Royce	United Kingdom	FALSE	88000	75000	750	1000	750	[illegible]
03/03/2012 00:00:00	Rolls Royce	United Kingdom	FALSE	89000	88000	0	1000	85	[illegible]
04/04/2012 00:00:00	Rolls Royce	United Kingdom	TRUE	92000	62000	0	1500	2500	[illegible]
04/05/2012 00:00:00	Rolls Royce	United Kingdom	FALSE	102500	125000	0	1000	4000	[illegible]
04/06/2012 00:00:00	Aston Martin	United Kingdom	FALSE	110000	56000	750	500	900	[illegible]
04/08/2012 00:00:00	Aston Martin	United Kingdom	TRUE	130000	15500	0	1000	900	[illegible]
04/09/2012 00:00:00	Aston Martin	United Kingdom	TRUE	75000	75890	0	1000	750	[illegible]
04/09/2012 00:00:00	Aston Martin	United Kingdom	TRUE	68500	99000	0	1000	750	[illegible]
04/11/2012 00:00:00	Aston Martin	United Kingdom	TRUE	155000	125000	0	1500	600	[illegible]
04/12/2012 00:00:00	Aston Martin	United Kingdom	FALSE	95000	155000	5000	1500	600	[illegible]
04/12/2012 00:00:00	Aston Martin	United Kingdom	FALSE	178500	125000	0	1500	600	[illegible]
02/02/2013 00:00:00	Rolls Royce	United Kingdom	TRUE	178500	62000	0	1750	600	[illegible]
02/03/2013 00:00:00	Rolls Royce	United Kingdom	FALSE	110000	75890	0	990	600	[illegible]
02/04/2013 00:00:00	Rolls Royce	United Kingdom	FALSE	130000	62000	0	500	600	[illegible]
02/05/2013 00:00:00	Rolls Royce	United Kingdom	FALSE	178500	75890	950	1750	450	[illegible]
02/06/2013 00:00:00	Rolls Royce	United Kingdom	TRUE	110000	62000	0	1750	400	[illegible]
02/07/2013 00:00:00	Rolls Royce	United Kingdom	TRUE	102500	62000	1750	500	400	[illegible]
02/08/2013 00:00:00	Rolls Royce	United Kingdom	FALSE	130000	62000	0	1000	400	[illegible]

The data in the preview has been truncated due to size limits.

Figure 3-11. *Database data selected using the SQL Statement option*

6. Click Load or Edit to continue with the data load process. Alternatively, you can click Cancel and start a different data load.

The data that is returned in this example is only a subset of the available data that has been selected by the stored procedure. You need to be aware that stored procedures can perform a multitude of tasks on the source data. These can include selecting, sorting, and cleansing the data.

■ **Note** A SQL statement or stored procedure will only return data as a *single table*. Admittedly, this table could contain data from several tables or views, but filtering the source data will prevent Power BI Desktop from loading data from several tables as separate queries. Consequently, you could have to create multiple queries rather than a single load query to get data from a coherent set of tables in the data source.

Oracle Databases

There are many, many database vendors active in the corporate marketplace today. Arguably the most dominant of them is Oracle. So while I have used Microsoft data sources to begin the journey into an understanding of how to use databases with Power BI Desktop, it would be remiss of me not to explain how to access databases from other suppliers.

So now is the time to show you just how open-minded Power BI Desktop really is. It does not limit you to Microsoft data sources-far from it. Indeed, it is every bit as easy to use databases from other vendors as the source of your analytical reports. As an example of this, let's take a look at loading Oracle data into Power BI Desktop.

Installing and configuring an Oracle database is a non-trivial task. Consequently, I am not providing an Oracle sample database, but will leave you either to discover a corporate database that you can connect to or, preferably, consult the many excellent resources available that do an excellent job of explaining how to set up your own Oracle database and install the sample data that is available.

Be aware that connecting to Oracle will require installing Oracle client software on the computer where you are running Power BI Desktop. This, too, can be complex to set up. So you might need some help from a corporate resource if you are planning to use Oracle data with Power BI Desktop.

Should you be feeling brave, you can use the following URLs to find the Oracle client software. For 32-bit versions of Power BI Desktop, use the following link to download and install the 32-bit Oracle client:

```
www.oracle.com/technetwork/topics/dotnet/utilsoft-086879.html
```

For 64-bit versions of Power BI Desktop, use the following link to download and install the 64-bit Oracle client:

```
www.oracle.com/technetwork/database/windows/downloads/index-090165.html
```

If you need to check which version of Power BI Desktop you are using (32 bit or 64 bit), click File ➤ Help ➤ About. You will see a dialog that tells you which version you are using.

So, assuming that you have an Oracle database available (and that you know the server name or SID as well as a valid user name and password), the following steps show how you can load data from this particular source into Power BI Desktop. I will be using standard Oracle sample data that is often installed with sample databases in this example.

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click Database.
3. In the Get Data dialog, click Oracle database on the right. The dialog will look like Figure 3-12.

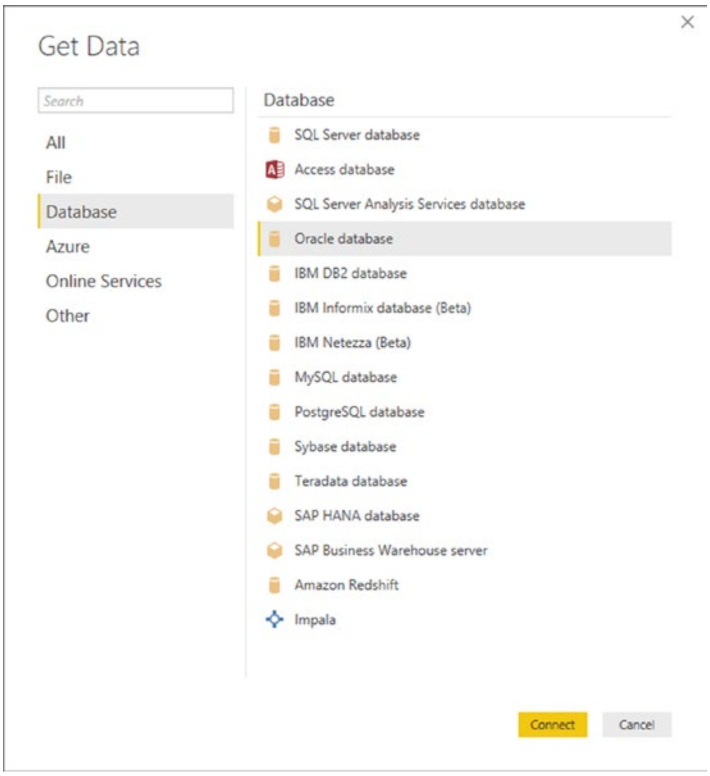
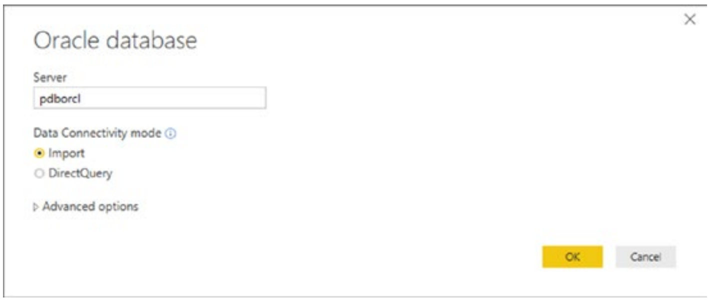


Figure 3-12. *Connecting to an Oracle database*

4. Click Connect. The Oracle Database dialog will appear.
5. Enter the server name in the Server text box. This will be the name of your Oracle server or one of the Oracle Server resources used by your organization.
6. Click the Import button. The dialog will look like Figure 3-13.


Oracle database

Server

pdborcl

Data Connectivity mode

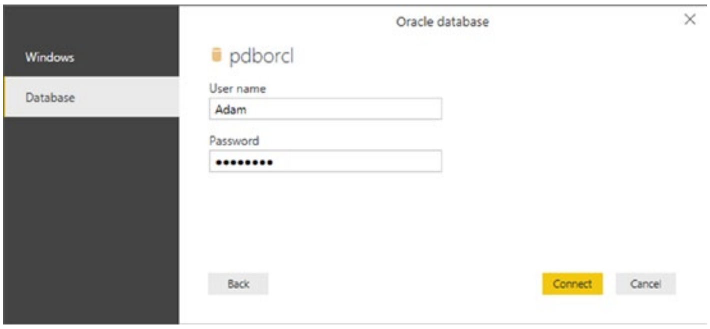
[✓ Import]

DirectQuery

Figure 3-13. *The Oracle Database dialog*

7. Click OK. The Oracle Database security dialog will appear. Assuming that you are not authorized to use your Windows login to connect to the database, click Database on the left of the dialog.

8. Enter the user name and password that allow you to log into Oracle, as shown in Figure 3-14.

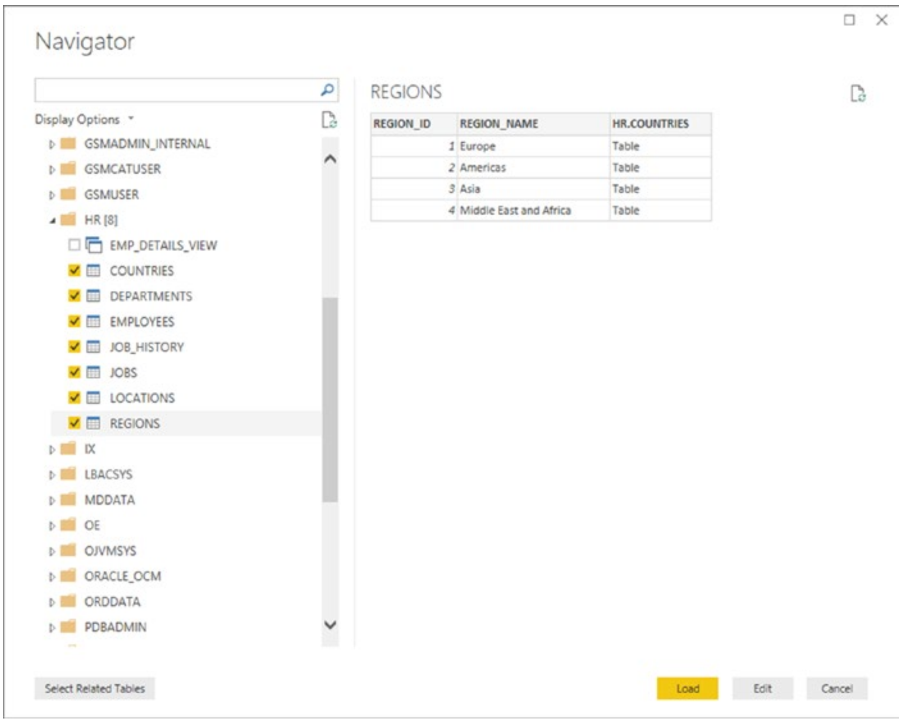


User name
Adam
Password
••••••••

Figure 3-14. *The Oracle Database security dialog*

9. Click Connect. Power BI Desktop will connect to the server and display the Navigator dialog containing all the tables and views in the database that you have permission to see on the server you selected. In some cases, you could see a dialog saying that the data source does not support encryption. If you feel happy with an unencrypted connection, then click the OK button for this dialog.

10. Expand the HR folder. This is a standard Oracle sample schema that could be installed on your Oracle instance. If not, you will have to choose another schema. Click the check boxes for the tables that interest you. The data for the most recently selected data appears on the right of the Navigator dialog, as shown in Figure 3-15.



REGIONS

REGION_ID	REGION_NAME	HR.COUNTRIES
1	Europe	Table
2	Americas	Table
3	Asia	Table
4	Middle East and Africa	Table

Figure 3-15. *The Navigator dialog using Oracle data*

11. Click Load. The Power BI Desktop window will open and display the tables that you selected in the Fields list in the Report window when you click OK. You can see the data in Power BI Desktop in Figure 3-16.

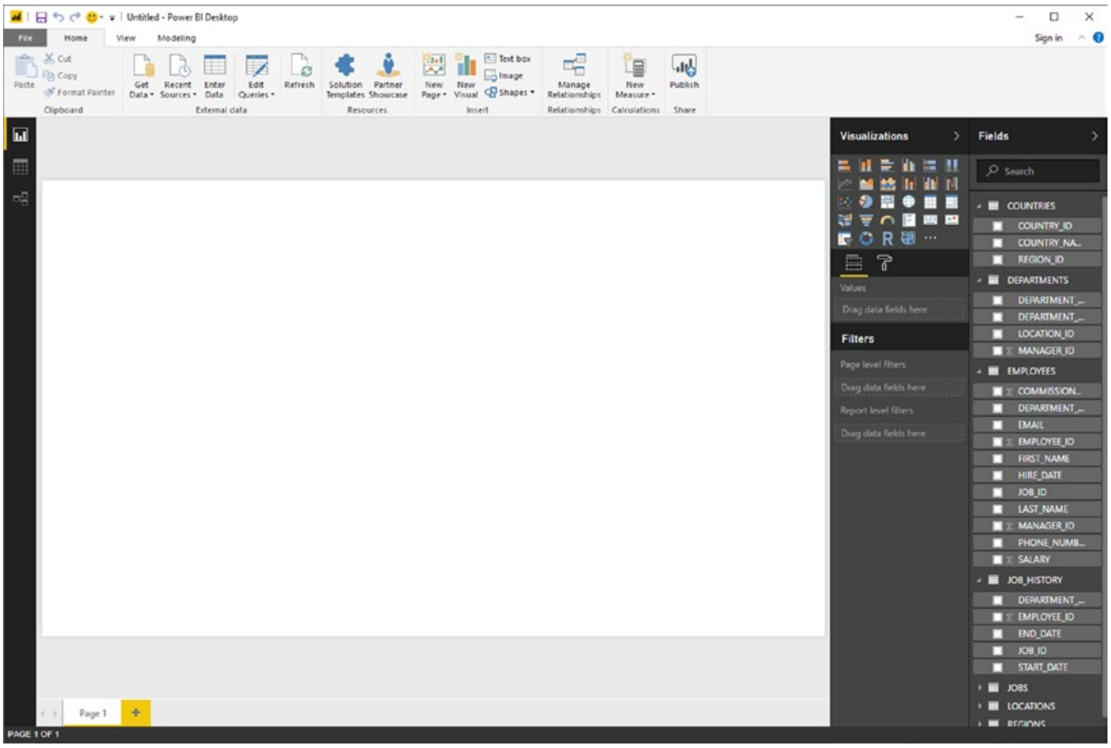


Figure 3-16. *Oracle data loaded into Power BI Desktop*

If you have already followed the example earlier in this chapter to load data from SQL Server, you will probably appreciate how much the two techniques have in common. Indeed, one of the great advantages of using Power BI Desktop is that loading data from different data sources follows a largely similar approach and uses many of the same steps and dialogs. This is especially true of databases, where the steps are virtually identical-whatever the database.

Of course, no two databases are alike. Consequently, you connect to an Oracle instance (or server) but cannot choose a database as you can in SQL Server (or Sybase, for instance). Similarly, where Oracle has Schemas to segregate and organize data tables, SQL Server has databases. Nonetheless, the Power BI Desktop Navigator will always organize data into a hierarchy of folders so that you can visualize the data structures in a clear, simple, and intuitive manner, whatever the underlying database.

■ **Note** Oracle databases can also be accessed using the DirectQuery option. This technique is explained in the next chapter.

Other Relational Databases

Table 3-1 at the start of this chapter contains the list of relational databases that Power BI Desktop could connect to as this book went to press. I imagine that the list has grown since this book was published. However, the good news is that you probably do not need much more information to connect to any of the databases that are available for you to use as data sources. Simply put-if you know how to connect to one of them, you can probably connect to any of them.

So I am not going to fill out reams of pages with virtually identical explanations of how to get data from a dozen or more relational databases. Instead I suggest that you simply try to connect, using the techniques that you have learned in this chapter for Oracle and SQL Server.

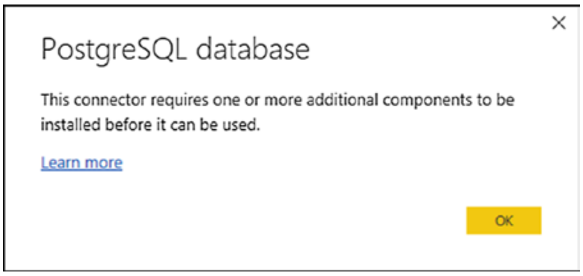
Be warned, though, that to connect to a relational database you will inevitably need to know the following details:

- The server name
- A database name (possibly)
- A valid user name (depending on the security that has been implemented)
- A valid password for the user that you are connecting as (this, too, will depend on the security in place)

However, if you have these elements, then nothing should stop you from using a range of corporate data sources as the basis for your analysis with Power BI Desktop. You will, of course, need all the necessary permissions to access the database and the data that it contains.

It is also worth knowing that connecting to DB2, MySQL, PostgreSQL, Sybase, IBM Informix, IBM Netezza, SAP HANA, or Teradata can require not only that the database administrator has given you the necessary permissions, but also that connection software (known as *drivers* or *providers*) has been installed on your PC. Given the "corporate" nature of the requirements, it may help if you talk directly to your IT department to get this set up in your enterprise IT landscape.

One way to find out if the software that is required to enable a connection to a specific database has been installed is to select the database from the list available in the Get Data dialog. If the drivers have not been installed, you will see a warning similar to the one in Figure 3-17.


PostgreSQL database

This connector requires one or more additional components to be installed before it can be used.

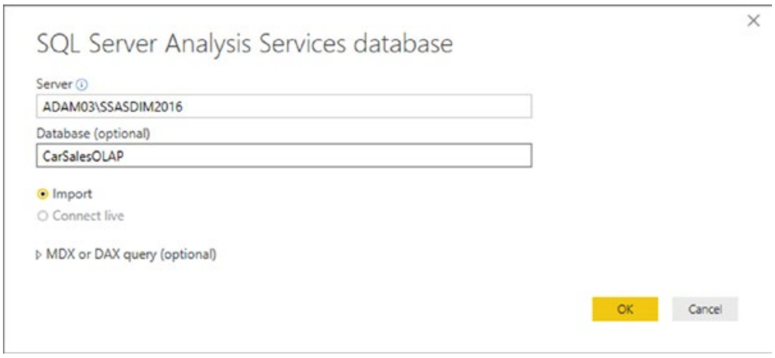
Figure 3-17. *The missing driver alert*

Clicking the "Learn more" link will take you to the download page for the missing drivers. Be warned, however, that configuring data providers can, in some cases, require a little specialist knowledge as well as access rights on the computer where the drivers have to be installed.

Microsoft SQL Server Analysis Services Data Sources

An Analysis Services database is a data warehouse technology that can contain vast amounts of data that has been optimized to enable decision making. *SSAS cubes* (as these databases are also called) are composed of facts (measures or values) and dimensions (descriptive attributes). If your enterprise uses Analysis Services databases, you can access them by doing the following steps:

1. In the Power BI Desktop ribbon, click Get Data ➤ More➤ Database and select SQL Server Analysis Services Database in the Get Data dialog.
2. Click Connect. The SQL Server Analysis Services Database dialog will appear.
3. Enter the Analysis Services server name and the database (or "cube") name, if you know it. If you are using the sample data from the Apress site for this book, the database is CarSalesOLAP; otherwise, you have to specify your own SSAS database name. In any case you will need to use the name of your own SSAS server. The dialog will look like Figure 3-18.


SQL Server Analysis Services database

Server: ADAM03\SSASDIM2016

Database (optional): CarSalesOLAP

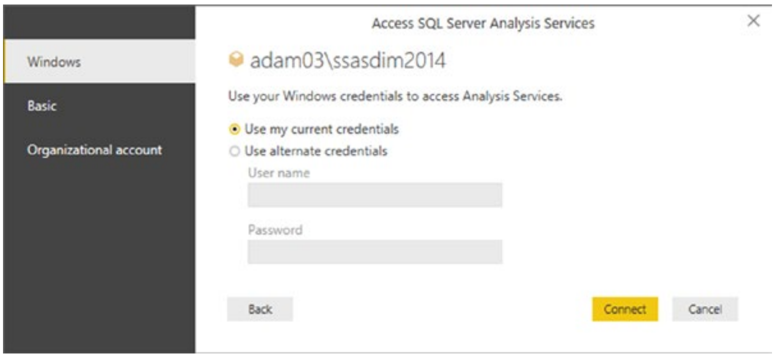
● Import
○ Connect live

▹ MDX or DAX query (optional)

OK | Cancel

Figure 3-18. *Connecting to an SSAS (multidimensional) database*

4. Click OK. If this is the first time that you are connecting to the cube, then the Access SQL Server Analysis Service dialog will appear so that you can define the credentials that you are using to connect to the Analysis Services database, as shown in Figure 3-19.


Access SQL Server Analysis Services

Windows | Basic | Organizational account

adam03\ssasdim2014

Use your Windows credentials to access Analysis Services.

● Use my current credentials
○ Use alternate credentials

User name

Password

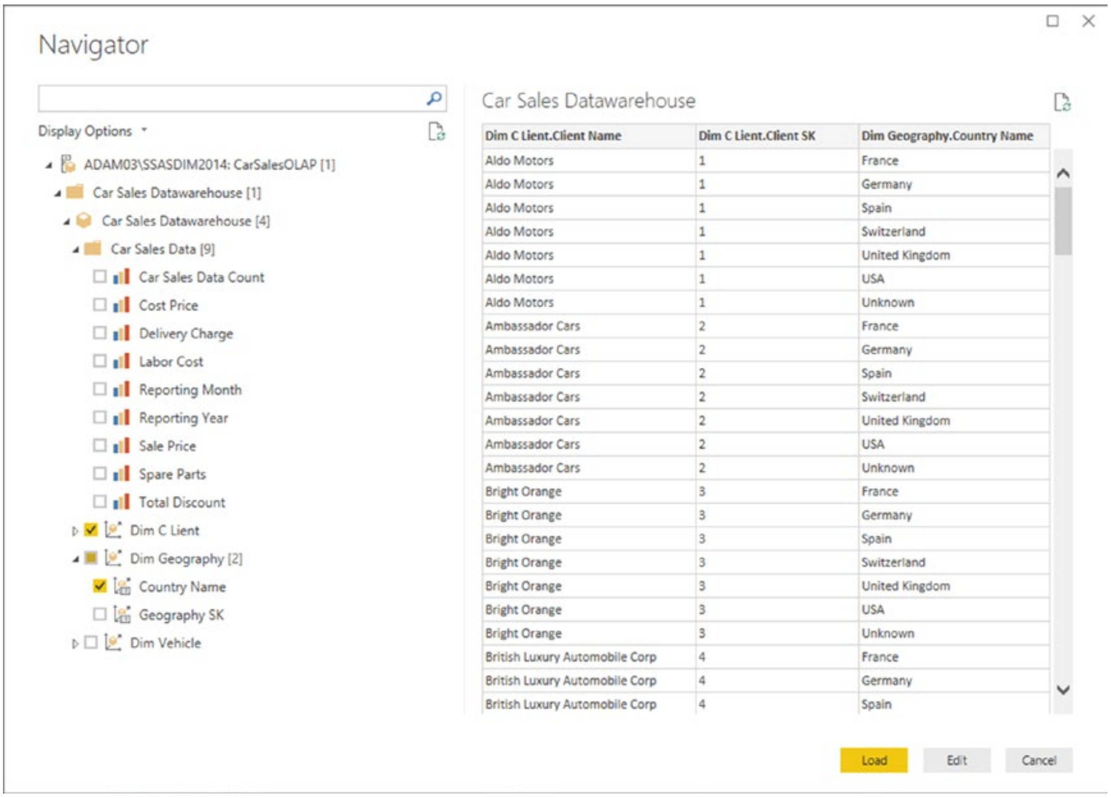
Back | Connect | Cancel

Figure 3-19. *SQL Server Analysis Services Credentials dialog*

5. Accept or alter the credentials and click Connect. The Navigator dialog will appear.

6. Expand the folders in the left pane of the dialog. This way, you can see all the fact tables and dimensions contained in the data warehouse.

7. Select the fact tables, dimensions, or even only the dimension elements and measures that you want to load. The dialog will look something like Figure 3-20.



ADAM03\SSASDIM2014: CarSalesOLAP [1]
Car Sales Datawarehouse [1]
Car Sales Datawarehouse [4]
Car Sales Data [9]
☐ Car Sales Data Count
☐ Cost Price
☐ Delivery Charge
☐ Labor Cost
☐ Reporting Month
☐ Reporting Year
☐ Sale Price
☐ Spare Parts
☐ Total Discount
☑ Dim C Lient
■ Dim Geography [2]
☑ Country Name
☐ Geography SK
☐ Dim Vehicle

Car Sales Datawarehouse

Dim C Lient.Client Name	Dim C Lient.Client SK	Dim Geography.Country Name
Aldo Motors	1	France
Aldo Motors	1	Germany
Aldo Motors	1	Spain
Aldo Motors	1	Switzerland
Aldo Motors	1	United Kingdom
Aldo Motors	1	USA
Aldo Motors	1	Unknown
Ambassador Cars	2	France
Ambassador Cars	2	Germany
Ambassador Cars	2	Spain
Ambassador Cars	2	Switzerland
Ambassador Cars	2	United Kingdom
Ambassador Cars	2	USA
Ambassador Cars	2	Unknown
Bright Orange	3	France
Bright Orange	3	Germany
Bright Orange	3	Spain
Bright Orange	3	Switzerland
Bright Orange	3	United Kingdom
Bright Orange	3	USA
Bright Orange	3	Unknown
British Luxury Automobile Corp	4	France
British Luxury Automobile Corp	4	Germany
British Luxury Automobile Corp	4	Spain

Figure 3-20. *Selecting attributes and measures from an SSAS cube*

8. Click Load. The Power BI Desktop window will open and display the measures and attributes that you selected in the Fields list in the Report window.

■ **Note** If you did not enter the cube (database) name in step 3, then the Navigator dialog will display all the available cubes on the SSAS server.

SSAS cubes are potentially huge. They can contain dozens of dimensions, many fact tables, and literally hundreds of measures and attributes. Understanding multidimensional cubes and how they work is beyond the scope of this book. Nonetheless, it is important to understand that for Power BI Desktop, a cube is just another data source. This means that you can be extremely selective as to the cube elements that you load into Power BI Desktop, and only load the elements that you need for your analysis. You can load entire dimensions or just a few attributes, just like you can load whole fact tables or just a selection of measures.

■ **Note** You can filter the data that is loaded from an SSAS cube by expanding the MDX or DAX query (optional) item in the SQL Server Analysis Services Database dialog. Then you can enter an MDX query in the box that appears before clicking OK. Be warned that SSAS cubes use queries written in MDX—a specialist language that is considered not always easy to learn. The good news is that if an Analysis Services expert has set up a cube correctly, you can see SSAS display folders in PowerBI Desktop Query (which is another term that people use to describe the Query Editor).

Analysis Services Cube Tools

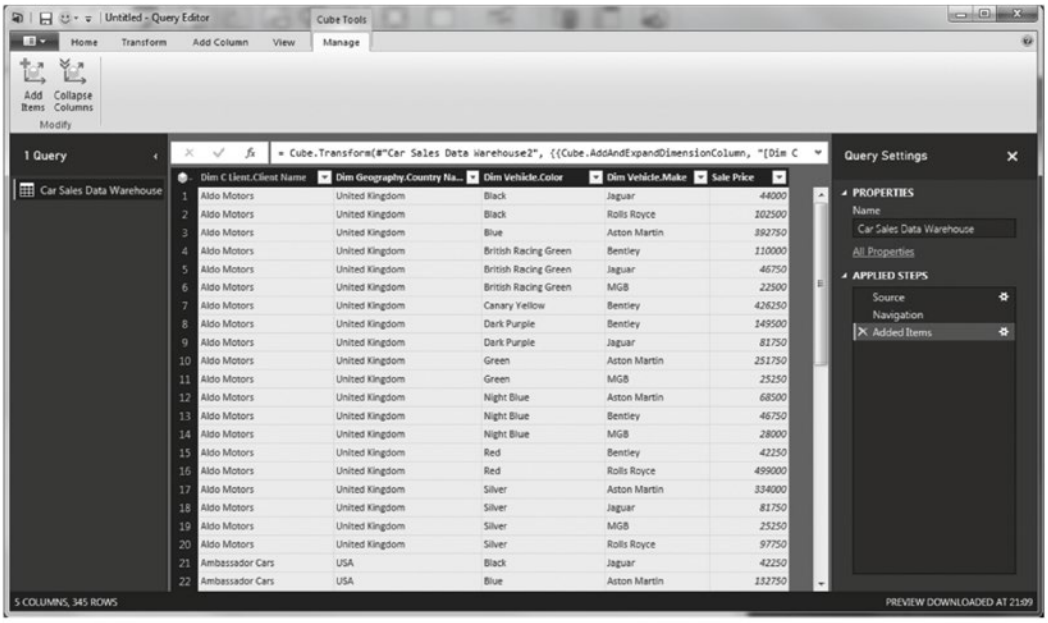
Analysis Services data sources allow you to tweak the selection of source elements in a way that is not available with other data sources. Essentially, you have two extra options:

- Add Items
- Collapse Columns

Add Items

When using an SSAS data source, you can at any time add any attributes or measures that you either forgot or thought that you would not need when setting up the initial connection.

1. In Power BI Desktop, click the Edit Queries button. The Power BI Desktop Query Editor window will be displayed. Assuming that there is only one query, the Manage ribbon will appear, as shown in Figure 3-21. Otherwise, click the SSAS query that you have previously established.



```
= Cube.Transform(#"Car Sales Data Warehouse2", {{Cube.AddAndExpandDimensionColumn, "[Dim C
```

	Dim C Lient.Client Name	Dim Geography.Country Na...	Dim Vehicle.Color	Dim Vehicle.Make	Sale Price
1	Aldo Motors	United Kingdom	Black	Jaguar	44000
2	Aldo Motors	United Kingdom	Black	Rolls Royce	102500
3	Aldo Motors	United Kingdom	Blue	Aston Martin	392750
4	Aldo Motors	United Kingdom	British Racing Green	Bentley	110000
5	Aldo Motors	United Kingdom	British Racing Green	Jaguar	46750
6	Aldo Motors	United Kingdom	British Racing Green	MGB	22500
7	Aldo Motors	United Kingdom	Canary Yellow	Bentley	426250
8	Aldo Motors	United Kingdom	Dark Purple	Bentley	149500
9	Aldo Motors	United Kingdom	Dark Purple	Jaguar	81750
10	Aldo Motors	United Kingdom	Green	Aston Martin	251750
11	Aldo Motors	United Kingdom	Green	MGB	25250
12	Aldo Motors	United Kingdom	Night Blue	Aston Martin	68500
13	Aldo Motors	United Kingdom	Night Blue	Bentley	46750
14	Aldo Motors	United Kingdom	Night Blue	MGB	28000
15	Aldo Motors	United Kingdom	Red	Bentley	42250
16	Aldo Motors	United Kingdom	Red	Rolls Royce	499000
17	Aldo Motors	United Kingdom	Silver	Aston Martin	334000
18	Aldo Motors	United Kingdom	Silver	Jaguar	81750
19	Aldo Motors	United Kingdom	Silver	MGB	25250
20	Aldo Motors	United Kingdom	Silver	Rolls Royce	97750
21	Ambassador Cars	USA	Black	Jaguar	42250
22	Ambassador Cars	USA	Blue	Aston Martin	132750

Figure 3-21. *Cube Tools*

2. In the Manage ribbon, click the Add Items button. The Add Items dialog will appear, as shown in Figure 3-22.

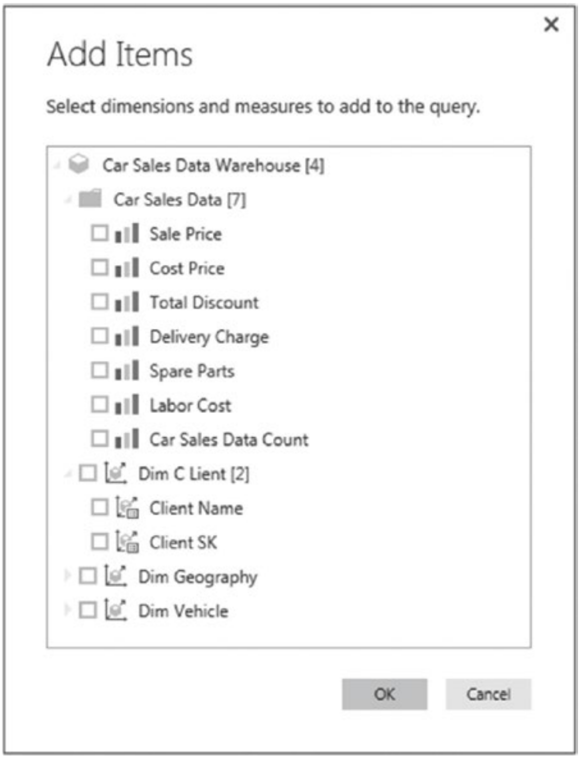


Figure 3-22. *The Add Items dialog*

3. Expand any measure groups and select all the measures and attributes that you want to add.
4. Click OK.
5. In the Power BI Desktop Query Editor activate the Home ribbon and click Close and Apply.

Any changes that you made are reflected in the data and the selected measures and attributes are added as new columns at the right of the dataset.

■ **Note** Power BI Desktop Query will not detect if any new measures and attributes that you add are already in the dataset. So if you add an element a second time, it will appear *twice* in the query.

Collapse Columns

Do the following to remove any columns that you no longer require from the data source (which can accelerate data refresh).

1. In the Query pane, click the SSAS query that you have previously established. (The sample data for this connection will be displayed in the center of the Query window-this is yet another term for the Query Editor window.) The Manage ribbon will appear.
2. In the Manage ribbon, click the Collapse Columns button.

The columns are removed from the connection to the SSAS cube and, consequently, from the Fields list at the right of the Power BI Desktop window. They are also removed from any visualizations that use them.

■ **Note** Removing columns from Power BI Desktop Query can have a serious domino effect on reports and dashboards. Consequently, you need to be very careful when removing them.

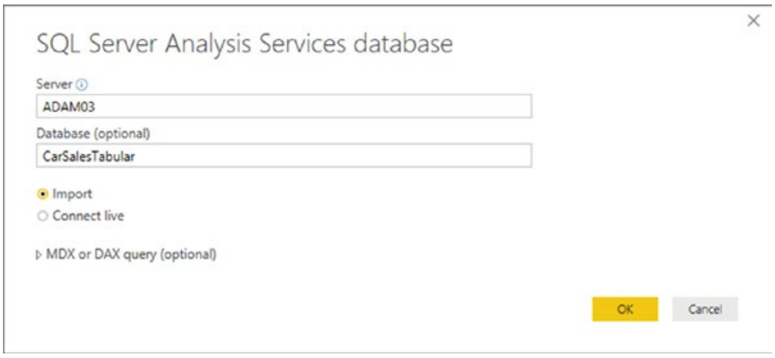
SSAS Tabular Data Warehouses

The previous section showed you how to connect to a SQL Server Analysis Services cube. However, there are now two types of SQL Server Analysis Services data warehouses:

- The “traditional” cube
- The “newer” tabular warehouse

As more and more data warehouses (at least the ones that are based on Microsoft technologies) are being built using the newer, tabular technology, it is probably worth your while to see how quickly and easily you can use these data sources with Power BI Desktop. Indeed, the steps that you follow to connect to either of these data warehouse sources are virtually identical. However, as Power BI is rapidly becoming the tool of choice to query tabular data warehouses, it is certainly worth a few minutes to learn how to connect to SSAS Tabular (as it is often called, for short).

1. In the Power BI Desktop ribbon, click Get Data ➤ More ➤ Database and select SQL Server Analysis Services Database in the Get Data dialog.
2. Click Connect. The SQL Server Analysis Services Database dialog will appear.
3. Enter the Analysis Services server name and the tabular database name (we don’t tend to call these cubes), if you know it. If you are using the sample data from the Apress site for this book, the database is CarSalesTabular; otherwise, you have to specify your own tabular database name. In any case you will need to use the name of your own SSAS server.
4. Click Import.
5. The dialog will look like Figure 3-23.



SQL Server Analysis Services database

Server: ADAM03

Database (optional): CarSalesTabular

[✓ Import] Connect live

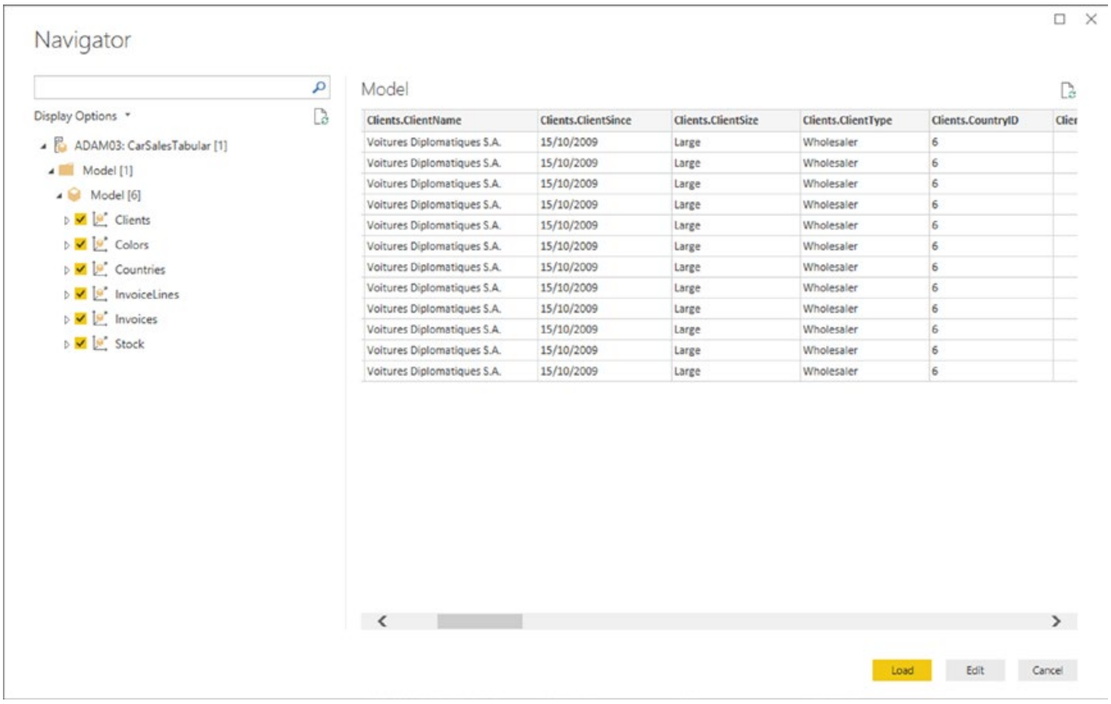
MDX or DAX query (optional)

Figure 3-23. *Connecting to an SSAS (multidimensional) database*

6. Click OK. If this is the first time that you are connecting to the tabular warehouse, then the Access SQL Server Analysis Service dialog will appear so that you can define the credentials that you are using to connect to the Analysis Services database, where you will have to accept or alter the credentials and click Connect. The Navigator dialog will appear.

7. Expand the folders in the left pane of the dialog. This way, you can see all the tables contained in the data warehouse. These may—or may not—be structured as facts and dimensions as was the case with a “classic” SSAS data warehouse.

8. Select the tables that you want to load. The dialog will look something like Figure 3-24.



Navigator

- ADAM03: CarSalesTabular [1]
  - Model [1]
    - Model [6]
      - ✓ Clients
      - ✓ Colors
      - ✓ Countries
      - ✓ InvoiceLines
      - ✓ Invoices
      - ✓ Stock

Model

Clients.ClientName	Clients.ClientSince	Clients.ClientSize	Clients.ClientType	Clients.CountryID	Clier
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	
Voitures Diplomatiques S.A.	15/10/2009	Large	Wholesaler	6	

Figure 3-24. *Selecting attributes and measures from an SSAS tabular data source*

9. Click Load. The Power BI Desktop window will open and display the measures and attributes that you selected in the Fields list in the Report window.

■ **Tip** You can filter the data that is loaded from an SSAS tabular database by expanding the MDX or DAX query (optional) item in the SQL Server Analysis Services Database dialog. Then you can enter a DAX query in the box that appears before clicking OK. SSAS tabular databases use queries written in DAX, which is the language the Power BI itself uses (and which you will start to learn in Chapters 11 through 13).

Import or Connect Live

So far in this chapter I have suggested that you use the Import option when sourcing data from Microsoft SQL Server databases and data warehouses. This is because the alternative, Connect Live (which is also known as DirectQuery), is such an important part of Power BI Desktop that I have preferred to make it the subject of a whole separate chapter. You will discover how to use this far-reaching and impressive technique in the next chapter.

ODBC Sources

As you have seen in this chapter and the preceding one, Power BI Desktop can connect to a wide range of data sources. However, there will always be database applications for which there is no specific connector built in to Power BI Desktop.

This is where a generic solution called Open Database Connectivity (or ODBC) comes into play. ODBC is a standard way to connect to data sources, most of which are databases or structured like databases. Simply put, if an ODBC driver exists for the application that you want to connect to, then you can load data from it into Power BI Desktop.

Hundreds of ODBC drivers have been written. Some are freely available, others require you to purchase a license. They exist for a wide spectrum of applications ranging from those found on most PCs to niche products.

Although ODBC is designed as a standard way of accessing data in applications, each ODBC driver is slightly different from every other ODBC driver. Consequently, you might have to spend a little time learning the quirks of the interface for the driver that comes with the application that you want to connect to.

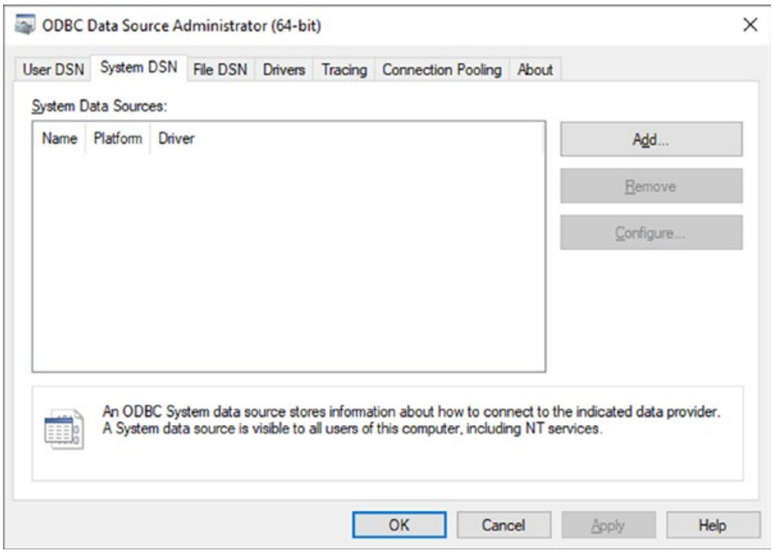
In this section we will use FileMaker Pro as a data source. This product is a desktop and server database system that has been around for quite some time. However, there is currently no specific Power BI Desktop connector for it. The good news is that FileMaker Pro *does* have an ODBC driver. So we will use ODBC to connect to FileMaker Pro from Power BI Desktop.

I have to add that I am not expecting you to install a copy (even if it is only a trial copy) of FileMaker Pro and its companion ODBC driver to carry out this exercise. What I do want to explain, however, is how you can use ODBC to connect to a wide range of data sources where an ODBC driver is available. So feel free to download and install FileMaker Pro and its ODBC driver if you wish, but you will have to refer to the FileMaker Pro documentation for an explanation of how to do this.

Assuming that you have an ODBC-compliant data source and a working ODBC driver for this data source, here is how to load data into Power BI Desktop using ODBC:

1. Run the ODBC Data Source Administrator app. This is normally in the folder C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools. Be sure to use the 64-bit version if you are using 64-bit Power BI Desktop or the 32-bit version if you are using 32-bit Power BI Desktop.

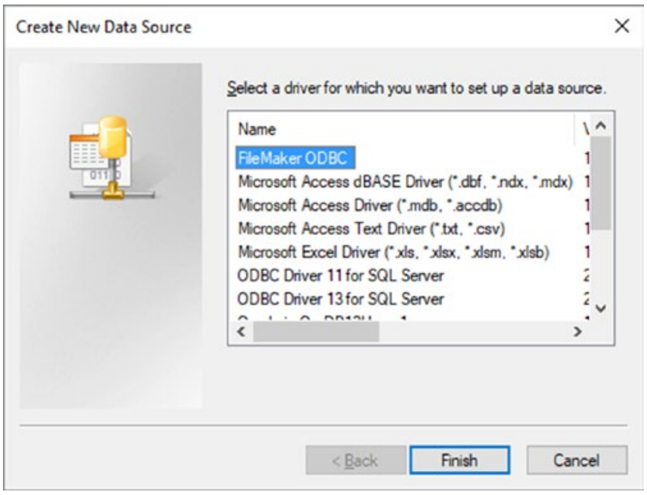
2. Click the System DSN tab. You should see the dialog shown in Figure 3-25.



An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users of this computer, including NT services.

Figure 3-25. *The ODBC Data Source Administrator*

3. Click Add. You will see the list of all currently installed ODBC drivers on your computer. This should look something like the dialog shown in Figure 3-26.



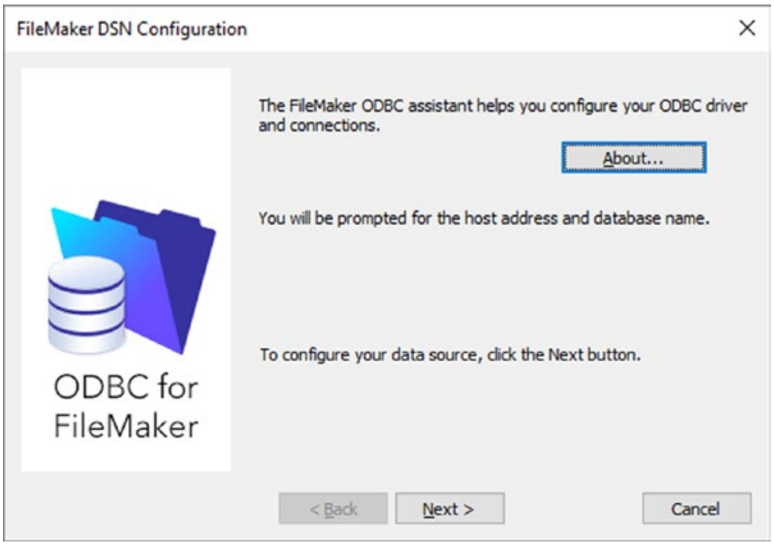
Select a driver for which you want to set up a data source.

FileMaker ODBC
Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
Microsoft Access Driver (*.mdb, *.accdb)
Microsoft Access Text Driver (*.txt, *.csv)
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
ODBC Driver 11 for SQL Server
ODBC Driver 13 for SQL Server

Figure 3-26. *The list of installed ODBC drivers*

4. Select the appropriate ODBC driver corresponding to the data source that you want to connect to (FileMaker ODBC in this example). If you cannot see the ODBC driver, you need to install—or reinstall—the driver.

5. Click Finish. The configuration dialog for the specific ODBC driver that you have selected will appear. If you are using FileMaker Pro, the dialog will look like Figure 3-27.


FileMaker DSN Configuration

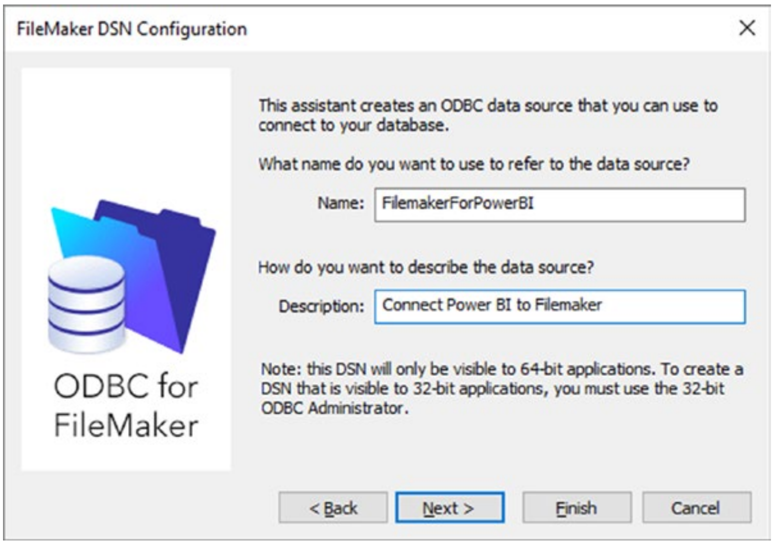
The FileMaker ODBC assistant helps you configure your ODBC driver and connections.

You will be prompted for the host address and database name.

To configure your data source, click the Next button.

Figure 3-27. *The FileMaker Pro ODBC configuration assistant*

6. Click Next, and enter a name and a description for this particular ODBC connection. This should look something like the dialog shown in Figure 3-28.


FileMaker DSN Configuration

This assistant creates an ODBC data source that you can use to connect to your database.

What name do you want to use to refer to the data source?

Name: FilemakerForPowerBI

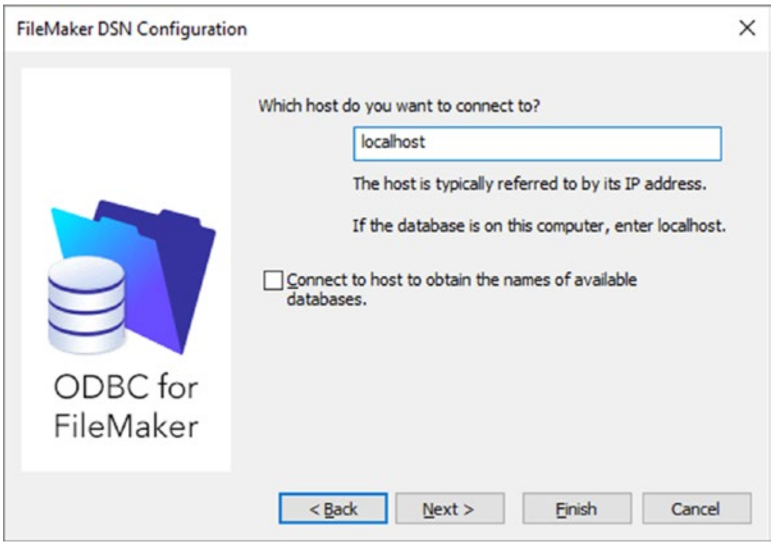
How do you want to describe the data source?

Description: Connect Power BI to Filemaker

Note: this DSN will only be visible to 64-bit applications. To create a DSN that is visible to 32-bit applications, you must use the 32-bit ODBC Administrator.

Figure 3-28. *Naming the ODBC connection for FileMaker Pro*

7. Click Next and enter **localhost** as the hostname if you are using a FileMaker trial version on your local computer. Otherwise, enter the IP address of the FileMaker server. You should see the dialog shown in Figure 3-29.


FileMaker DSN Configuration

Which host do you want to connect to?

localhost

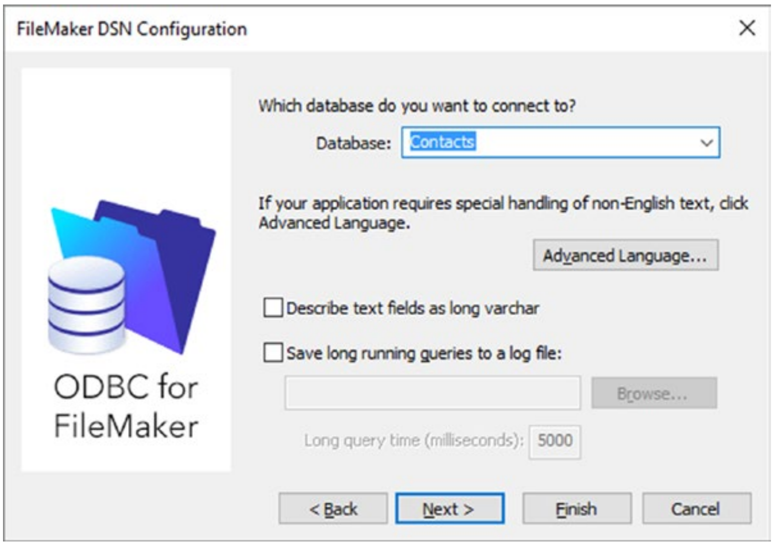
The host is typically referred to by its IP address.

If the database is on this computer, enter localhost.

☐ Connect to host to obtain the names of available databases.

Figure 3-29. *Specifying the host for the ODBC data*

8. Click Next and select the database in FileMaker Pro that you want to connect to. You will see the dialog shown in Figure 3-30 (if you are *not* using FileMaker Pro—remember that these dialogs can vary depending on the specific ODBC driver).


FileMaker DSN Configuration

Which database do you want to connect to?

Database: Contacts

If your application requires special handling of non-English text, click Advanced Language.

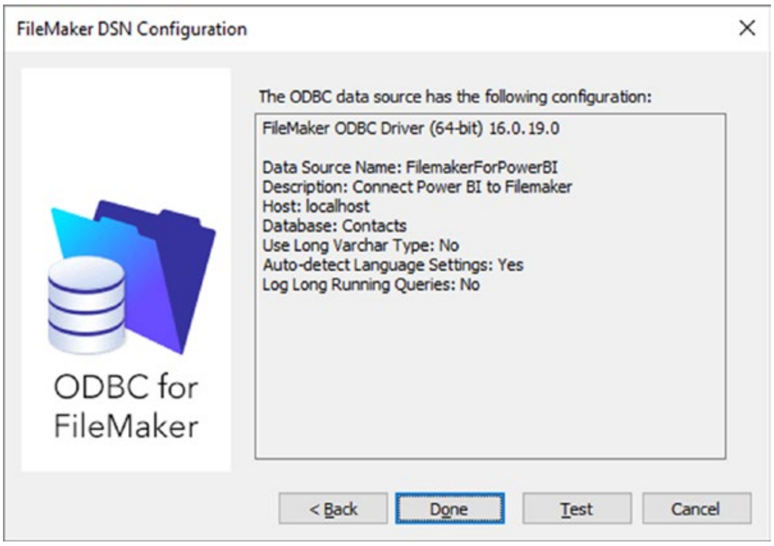
☐ Describe text fields as long varchar

☐ Save long running queries to a log file:

Long query time (milliseconds): 5000

Figure 3-30. *Specifying the database for the ODBC data*

9. Click Next. The ODBC configuration dialog will resume the specifications for the connection. This could look something like the one shown in Figure 3-31.



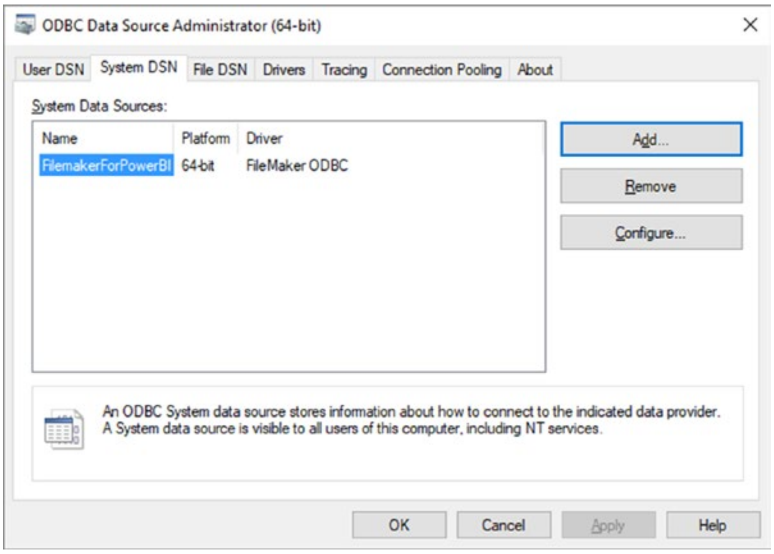
The ODBC data source has the following configuration:

FileMaker ODBC Driver (64-bit) 16.0.19.0

Data Source Name: FilemakerForPowerBI
Description: Connect Power BI to Filemaker
Host: localhost
Database: Contacts
Use Long Varchar Type: No
Auto-detect Language Settings: Yes
Log Long Running Queries: No

Figure 3-31. *The ODBC connection confirmation dialog*

10. Click Done. You will return to the ODBC Data Source Administrator, where you will see the System DSN that you just created. The ODBC Data Source Administrator dialog should look something like the one shown in Figure 3-32.

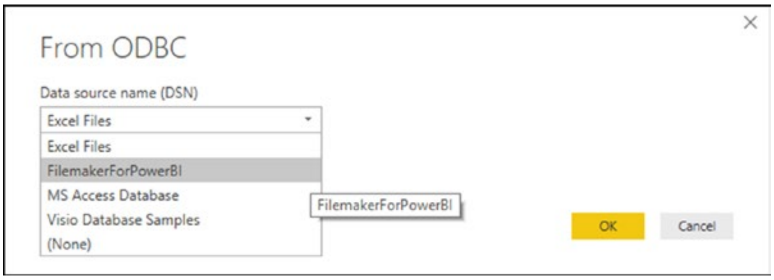


Name	Platform	Driver
FilemakerForPowerBI	64-bit	FileMaker ODBC

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users of this computer, including NT services.

Figure 3-32. *The ODBC Data Source Administrator dialog with an ODBC driver configured*

11. Click OK. This will close the ODBC Data Source Administrator dialog.
12. Launch Power BI Desktop.
13. Click Get Data ➤ Other.
14. Select ODBC from the available data sources on the right.
15. Click Connect. The From ODBC dialog will appear.
16. Expand the list of available DSNs. The From ODBC dialog will look something like the one in Figure 3-33.


From ODBC

Data source name (DSN)

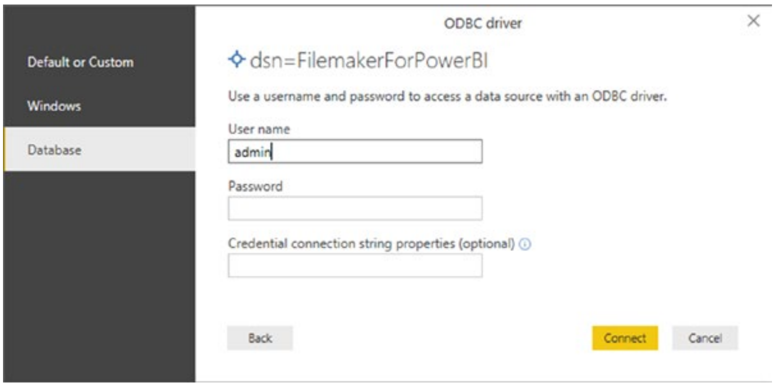
Excel Files

Excel Files
FilemakerForPowerBI
MS Access Database
Visio Database Samples
(None)

FilemakerForPowerBI

Figure 3-33. *The Power BI Desktop From ODBC dialog to select an ODBC data source*

17. Select the DSN that you created previously (FileMakerForPowerBI in this example).
18. Click OK. The Credentials dialog will appear.
19. Choose Windows integrated security or click Database on the left and enter the user name that has permissions to connect using the ODBC driver. The Credentials dialog will look something like the one in Figure 3-34.


ODBC driver

dsn=FilemakerForPowerBI

Use a username and password to access a data source with an ODBC driver.

User name: admin

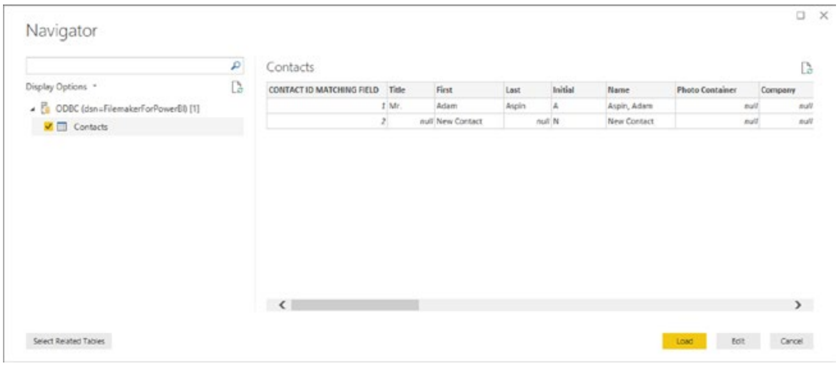
Password

Credential connection string properties (optional)

Figure 3-34. *The ODBC driver security dialog*

20. Click Connect. You will see the data that is available in the ODBC data source in the Navigator window.

21. Select the table(s) that you want to load into Power BI Desktop. You can see the data contained in the selected table in Figure 3-35.



Contacts

CONTACT ID MATCHING FIELD	Title	First	Last	Initial	Name	Photo Container	Company
1	Mr.	Adam	Aspin	A	Aspin, Adam	null	null
2	null	New Contact	null	N	New Contact	null	null

Figure 3-35. The Navigator dialog when using an ODBC source dialog

22. Click Load to load the data from the ODBC source into Power BI Desktop.

I realize that this process may seem a little laborious at first. Yet you have to remember that you will, in all probability, only set up the ODBC connection once. After that you can use it to connect to the source data as often as you want.

You need to be aware that each and every ODBC driver is different. So the appearance of the dialogs in steps 5 to 10 will vary slightly with each different ODBC driver that you configure. The key elements will, nonetheless, always be the same. They are

- Name the DSN.
- Specify the host computer for the data.
- Define the data repository (or database).

There is much more that could be written about creating and using ODBC connections to load data into Power BI Desktop—or indeed into any number of destination applications. However, I will have to refer you to the wealth of available resources both in print and online if you need to learn more about this particular technology. A good starting point is the Microsoft documentation that explains the difference between System, User, and File DSNs and describes many of the key elements that you might need to know.

■ **Note** FileMaker Pro must be open and/or running for an ODBC connection to work. Other ODBC sources could have their own specific quirks.

As a final , I can only urge you to procure all the relevant documentation for the ODBC driver that you intend to use with Power BI Desktop. Indeed, if you are using an enterprise data source that uses ODBC drivers, you may have corporate resources who can configure ODBC for you.

OLE DB Data Sources

OLE DB (short for Object Linking and Embedding, Database) is technically what is known as an application programming interface (API). Less technically it is a technique for connecting to database sources in a generic manner.

So, in a somewhat similar fashion to ODBC, you can use OLE DB to connect to data sources (which are often databases, although they can be other sources of data). Indeed, you may find that OLE DB is a useful way to connect to a database even if another method exists.

So, whatever the use that you find for OLE DB, it is well worth getting to know how it works. In this example I will use OLE DB to connect to SQL Server and the sample database that you first saw at the start of this chapter.

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the Get Data button.
3. Click Other on the left then click OLE DB on the right. The dialog will look like Figure 3-36.

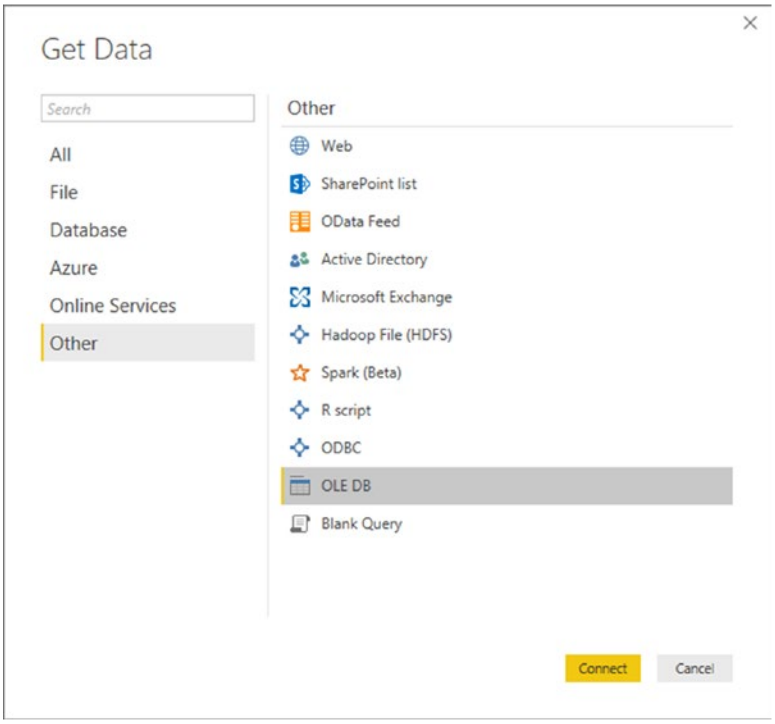


Figure 3-36. *Starting an OLE DB connection*

4. Click Connect. The From OLE DB dialog will appear. It should look like Figure 3-37.

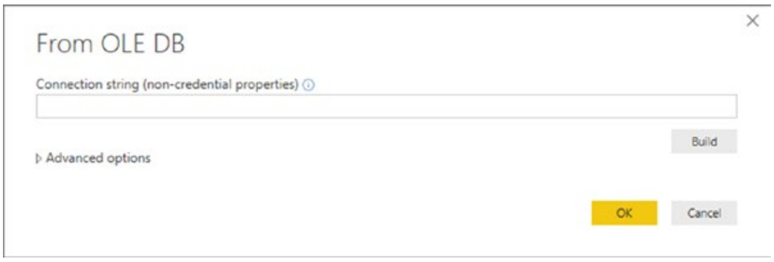
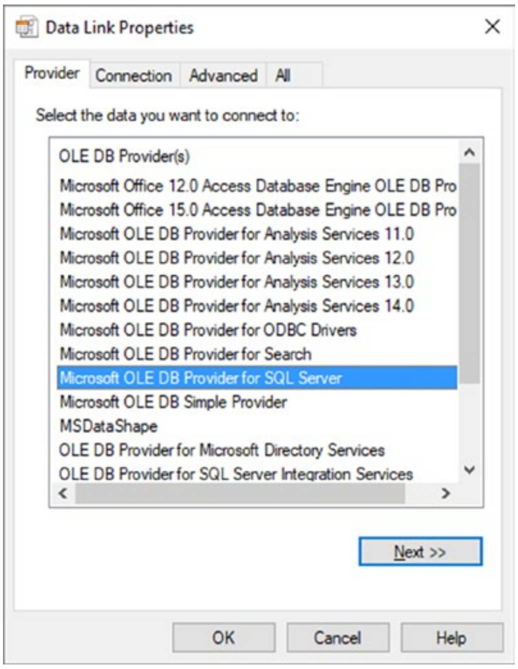


Figure 3-37. *The From OLE DB dialog*

5. If you have a fully working connection string, enter it in the Connection String text box.

6. If you do not have a connection string, click the Build button. The OLE DB Data Link Properties dialog will be displayed, as shown in Figure 3-38.



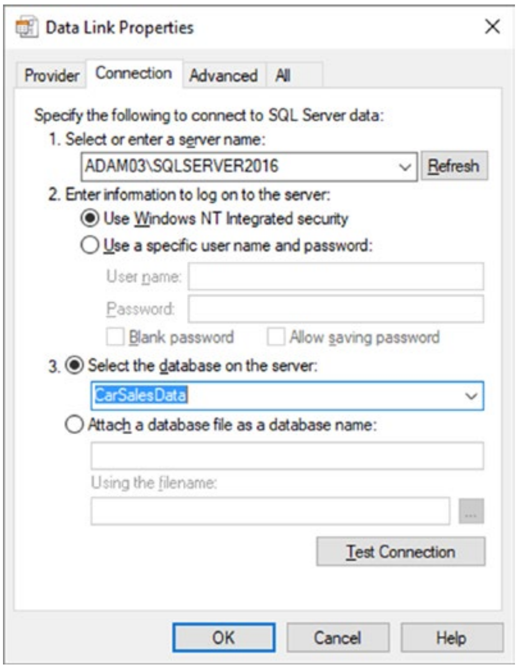
Select the data you want to connect to:

OLE DB Provider(s)
Microsoft Office 12.0 Access Database Engine OLE DB Pro
Microsoft Office 15.0 Access Database Engine OLE DB Pro
Microsoft OLE DB Provider for Analysis Services 11.0
Microsoft OLE DB Provider for Analysis Services 12.0
Microsoft OLE DB Provider for Analysis Services 13.0
Microsoft OLE DB Provider for Analysis Services 14.0
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for Search
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MSDataShape
OLE DB Provider for Microsoft Directory Services
OLE DB Provider for SQL Server Integration Services

Figure 3-38. *The OLE DB Data Link Properties dialog*

7. Select the OLE DB data provider that you want to use. In this example, it will be Microsoft OLE DB Provider for SQL Server.

8. Click Next. The Connection properties pane of the OLE DB Data Link Properties dialog will appear.
9. Select an available SQL Server (or enter its name) from the "Select or enter a server name" popup.
10. Select the type of security, and enter a user name and password if you have selected to use a specific user name instead of using Windows NT Integrated security.
11. Select the source database from the "Select the database on the server" popup. The dialog will look something like the one shown in Figure 3-39.



Specify the following to connect to SQL Server data:
1. Select or enter a server name:
ADAM03\SQLSERVER2016

2. Enter information to log on to the server:
[✓ Use Windows NT Integrated security]
Use a specific user name and password:
User name:
Password:
Blank password
Allow saving password
3. [✓ Select the database on the server:]
CarSalesData
Attach a database file as a database name:
Using the filename:

Figure 3-39. *The Connection properties of the OLE DB Data Link Properties dialog*

12. Click the Test Connection button to ensure that the connection is valid. You should see the message in Figure 3-40.

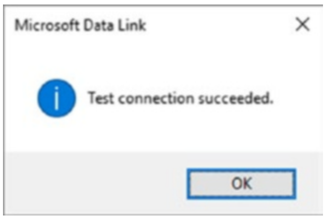
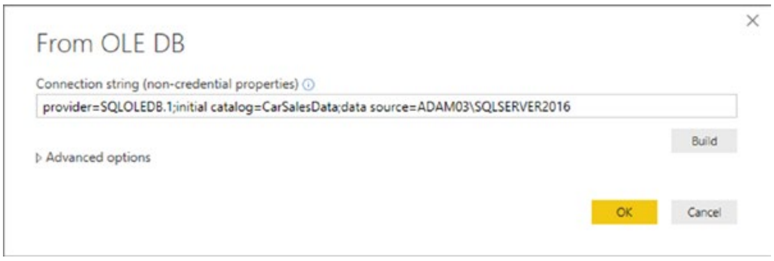

Microsoft Data Link
Test connection succeeded.

Figure 3-40. *The test connection alert*

13. Click OK. Power BI Desktop will build the connection string and insert it into the From OLE DB dialog, as shown in Figure 3-41.

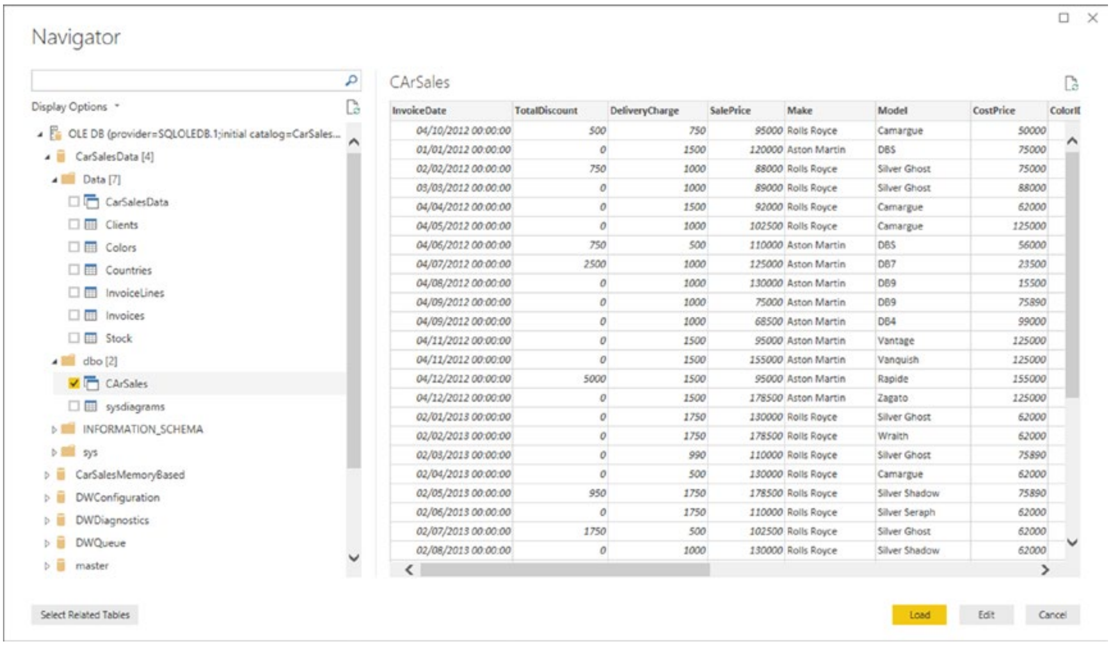

From OLE DB

Connection string (non-credential properties)

provider=SQLOLEDB.1;initial catalog=CarSalesData;data source=ADAM03\SQLSERVER2016

Figure 3-41. *The From OLE DB dialog with a valid connection string*

14. Click OK. The Navigator window will be displayed, as shown in Figure 3-42.


Navigator

Display Options

- OLE DB (provider=SQLOLEDB.1;initial catalog=CarSales...
  - CarSalesData [4]
    - Data [7]
      - CarSalesData
      - Clients
      - Colors
      - Countries
      - InvoiceLines
      - Invoices
      - Stock
    - dbo [2]
      - ✓ CArSales
      - sysdiagrams
    - INFORMATION_SCHEMA
    - sys
  - CarSalesMemoryBased
  - DWConfiguration
  - DWDiagnostics
  - DWQueue
  - master

CArSales

InvoiceDate	TotalDiscount	DeliveryCharge	SalePrice	Make	Model	CostPrice	ColorI
04/10/2012 00:00:00	500	750	95000	Rolls Royce	Camargue	50000	
01/01/2012 00:00:00	0	1500	120000	Aston Martin	DBS	75000	
02/02/2012 00:00:00	750	1000	88000	Rolls Royce	Silver Ghost	75000	
03/03/2012 00:00:00	0	1000	89000	Rolls Royce	Silver Ghost	88000	
04/04/2012 00:00:00	0	1500	92000	Rolls Royce	Camargue	62000	
04/05/2012 00:00:00	0	1000	102500	Rolls Royce	Camargue	125000	
04/06/2012 00:00:00	750	500	110000	Aston Martin	DBS	56000	
04/07/2012 00:00:00	2500	1000	125000	Aston Martin	DB7	23500	
04/08/2012 00:00:00	0	1000	130000	Aston Martin	DB9	15500	
04/09/2012 00:00:00	0	1000	75000	Aston Martin	DB9	75890	
04/09/2012 00:00:00	0	1000	68500	Aston Martin	DB4	99000	
04/11/2012 00:00:00	0	1500	95000	Aston Martin	Vantage	125000	
04/11/2012 00:00:00	0	1500	155000	Aston Martin	Vanquish	125000	
04/12/2012 00:00:00	5000	1500	95000	Aston Martin	Rapide	155000	
04/12/2012 00:00:00	0	1500	178500	Aston Martin	Zagato	125000	
02/01/2013 00:00:00	0	1750	130000	Rolls Royce	Silver Ghost	62000	
02/02/2013 00:00:00	0	1750	178500	Rolls Royce	Wraith	62000	
02/03/2013 00:00:00	0	990	110000	Rolls Royce	Silver Ghost	75890	
02/04/2013 00:00:00	0	500	130000	Rolls Royce	Camargue	62000	
02/05/2013 00:00:00	950	1750	178500	Rolls Royce	Silver Shadow	75890	
02/06/2013 00:00:00	0	1750	110000	Rolls Royce	Silver Seraph	62000	
02/07/2013 00:00:00	1750	500	102500	Rolls Royce	Silver Ghost	62000	
02/08/2013 00:00:00	0	1000	130000	Rolls Royce	Silver Shadow	62000	

Figure 3-42. *The Navigator window for an OLE DB connection*

15. Click Load to load the data into Power BI Desktop.

■ **Note** If this is a first connection to an OLE DB source, you may be asked for a user name and password, as was the case with earlier examples in this chapter.

You need to be aware that an OLE DB connection requires that the OLE DB driver (or "provider") is installed on the computer where you are running Power BI Desktop. However, what is really interesting is that an OLE DB connection can be reduced to a simple connection string. So if you need to share the connection with other users, you can simply e-mail the connection string to them in many cases. Your colleagues can then simply paste the connection string into the From OLE DB dialog in Power BI Desktop. In other words (and using this example as a model), you can simply send the following text to a co-worker:

```
provider=SQLOLEDB.1;initial catalog=CarSalesData;data source=ADAM03\SQLSERVER2016
```

They can use this string to connect to a specified database by pasting it into the From OLE DB dialog.

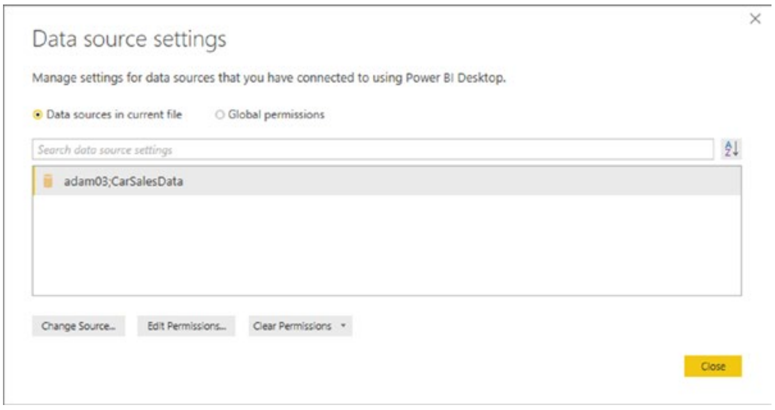
There are other advantages to using OLE DB connections too. Specifically, you (or your IT department) can provide a high level of configuration in the connection string to speed up or otherwise ameliorate the access to the data. This could be by specifying a mirrored server that is to be used for reporting to relieve the pressure on a main server, for instance. At this level the technical ramifications will depend on the OLE DB data source as well as the driver used, and consequently are outside the scope of this book.

Modifying Connections

If you are working in a structured development environment-or even if you are testing dashboards on a dataset that is either an old version or possibly on a non-live server-you could want at some point to switch from a current data source to another source. Power BI Desktop will let you do this. However, switching data sources will *only* work if the structures of the source and the destination data are *identical*. Practically, this means that the server and database can be named differently, but the tables and fields must have the same names.

To see this in action, you can do the following:

1. In an existing Power BI Desktop file, click the small triangle at the bottom right of the Edit Queries button in the Home ribbon, and select Data Source Settings. The Data Source Settings dialog will be displayed. Figure 3-43 shows you this dialog for the SQL Server connection that you saw at the start of this chapter.


Data source settings

Manage settings for data sources that you have connected to using Power BI Desktop.

● Data sources in current file ○ Global permissions

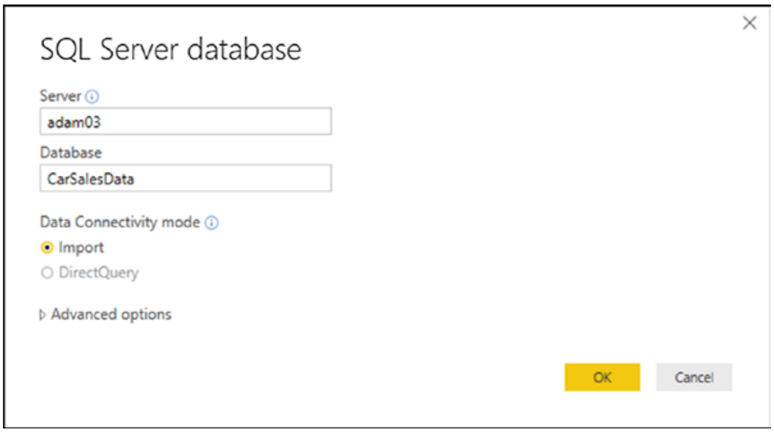
Search data source settings

adam03;CarSalesData

Figure 3-43. *The Data Source Settings dialog*

2. Click the connection that you want to modify.

3. Click Change Source. The dialog that originally allowed you to specify the data source (in this example that is the SQL Server Database connection dialog) will appear. You can see this in Figure 3-44.



SQL Server database

Server: adam03

Database: CarSalesData

Data Connectivity mode: [✓ Import] DirectQuery

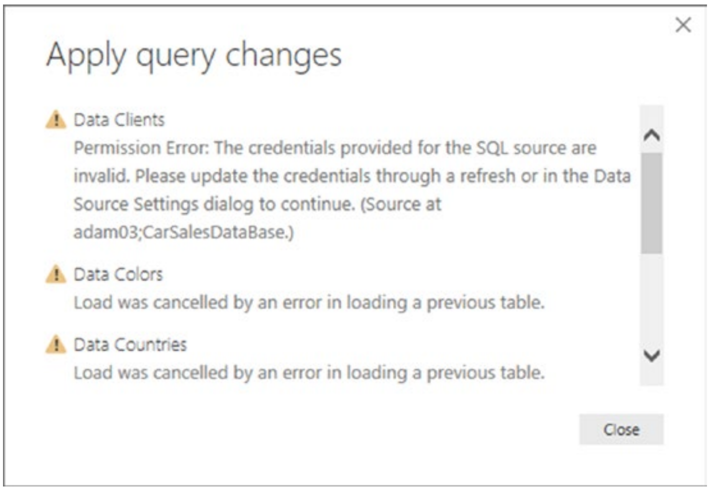
Figure 3-44. *The connection dialog*

4. Specify a different server and/or database as the data source.
5. Click OK. You will return to the Data Source Settings dialog.
6. Click Close.
7. Click the Apply Changes button that appears under the Power BI Desktop ribbon at the top of the screen.

■ **Note** If this is the first time that you are establishing a connection to this new server and database, you will have to specify the credentials to use.

Assuming that the new data source contains the same table names and structures, the existing data will be replaced with the data from the new source that you specified.

I have to stress again that this technique will only work if the underlying database metadata is identical across the two servers. If the data structures are not the same, you will see an error dialog similar to that shown in Figure 3-45.


Apply query changes

Data Clients
Permission Error: The credentials provided for the SQL source are invalid. Please update the credentials through a refresh or in the Data Source Settings dialog to continue. (Source at adam03;CarSalesDataBase.)

Data Colors
Load was cancelled by an error in loading a previous table.

Data Countries
Load was cancelled by an error in loading a previous table.

Figure 3-45. *Modifying a database connection*

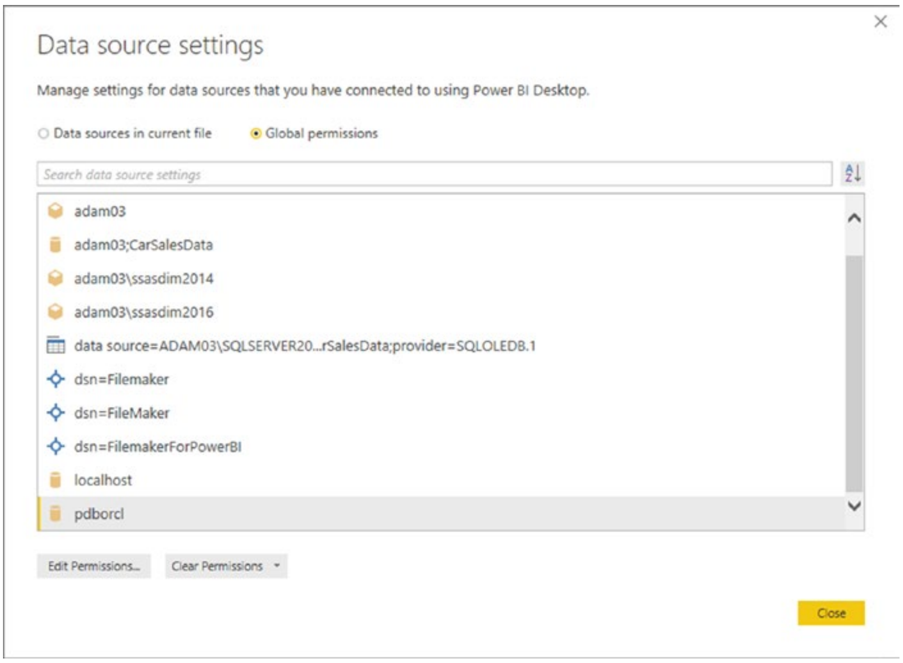
In cases like this, you may well have to rebuild a new Power BI Desktop file using the new data source. This could mean re-creating or copying any data mashups and formulas (as well as actual visualizations) from the old version to the new file.

Changing Permissions

It is all too frequent when working with databases and data warehouses to encounter permissions problems. It could be that you set up a connection to a database which required you to change the password at a later date. Meanwhile, the password stored in Power BI Desktop is the old version. So when you try to update your dashboard, you hit a blocker.

Fortunately (assuming, at least, that you know the new password), you can update your stored credentials in Power BI Desktop. As an example, let's suppose that you want to update your Oracle password. Here is how:

1. In an existing Power BI Desktop file, click the small triangle at the bottom right of the Edit Queries button in the Home ribbon, and select Data Source Settings.
2. Click Global Permissions. The Data Source Settings dialog will be displayed. Figure 3-46 shows you this dialog for the current Power BI Desktop connections on my PC.


Data source settings

Manage settings for data sources that you have connected to using Power BI Desktop.

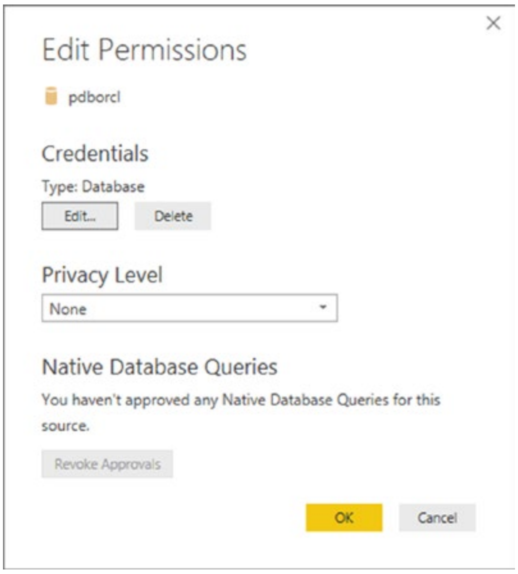
○ Data sources in current file ● Global permissions

Search data source settings

adam03
adam03;CarSalesData
adam03\ssasdim2014
adam03\ssasdim2016
data source=ADAM03\SQLSERVER20...rSalesData;provider=SQLOLEDB.1
dsn=Filemaker
dsn=FileMaker
dsn=FilemakerForPowerBI
localhost
pdborcl

Figure 3-46. *Permissions for current connections*

3. Click the name of the connection that you want to change (pdborcl in this example).
4. Click the Edit Permissions button. The Edit Permissions dialog will be displayed, as shown in Figure 3-47.


Edit Permissions

pdborcl

Credentials

Type: Database

Privacy Level

None

Native Database Queries

You haven't approved any Native Database Queries for this source.

Figure 3-47. *The Edit Permissions dialog*

5. Click Edit. The connection permissions dialog will be displayed (shown earlier in Figure 3-14 in the section that explains how to connect to an Oracle database).
6. Modify the password (or even the user name if this is required).
7. Click Save. This returns you to the Edit Permissions dialog.
8. Click OK. You will return to the Data Source Settings dialog.
9. Click Close to return to Power BI Desktop.

You can now refresh the data using the new permissions that you have just entered.

Refreshing Data from Databases and Data Warehouses

Loading data from databases and data warehouses only means that a snapshot of the source data is copied into Power BI Desktop. If the source data is updated, extended, or deleted, then you will need to get the latest version of the data if you want your analyses to reflect the current state of the data.

Essentially you have two options to do this:

- Refresh all the source data from all the data sources that you have defined.
- Refresh one or more tables individually.

Refreshing the Entire Data in the Power BI Desktop In-Memory Model

There is only one way to be certain that all your data is up to date. Refreshing the entire data may take longer, but you will be sure that your Power BI Desktop file contains the latest available data from all the sources that you have connected to.

To carry out a complete refresh:

1. In the Home ribbon, click the Refresh button. The Refresh dialog will appear showing all the data sources that are currently being refreshed. This dialog will look like the one in Figure 3-48.

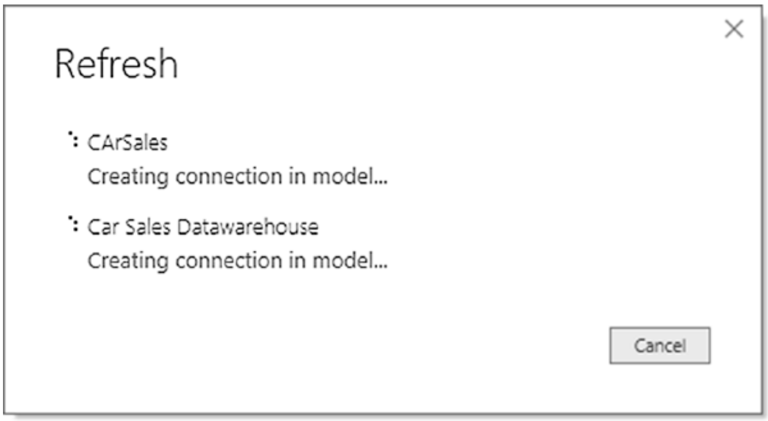

Refresh
CArSales
Creating connection in model...
Car Sales Datawarehouse
Creating connection in model...
Cancel

Figure 3-48. *The Refresh dialog*

■ **Note** A full data refresh can take quite a while if the source data is voluminous or if the network connection is slow.

Refreshing an Individual Table

If you are certain that only one or more tables need to be refreshed in your Power BI Desktop data model, then you can choose to refresh tables individually. To do this:

1. In the Fields pane, right-click the table that you want to refresh.
2. Select Refresh Data in the context menu. This is illustrated in Figure 3-49.

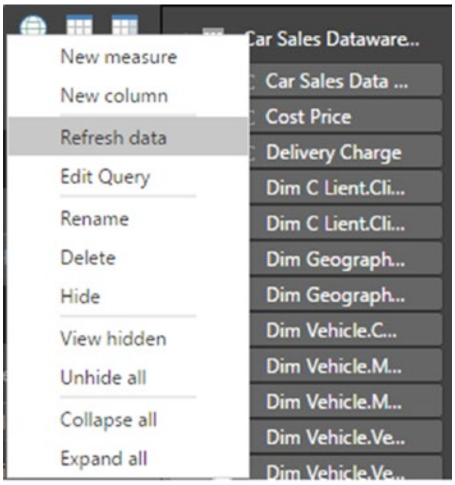


Figure 3-49. *Refreshing a single table*

The refresh dialog will appear (possibly only briefly) and the existing data for this table will be replaced with the latest data.

Conclusion

In this chapter, you have seen how to connect Power BI Desktop to some of the plethora of databases and data warehouses that currently exist. Moreover, you have seen that Power BI Desktop comes equipped "out of the box" with connections to most of the databases that currently exist in a corporate environment.

Power BI Desktop does not limit you to a predefined set of available data sources. Provided that your source database comes complete with one of the generic data providers-ODBC or OLE DB-then Power BI Desktop can, in all probability, access these sources too.

Moreover, corporate data sources can evolve and change. You also saw how Power BI Desktop allows you to update the permissions that you originally specified for a connection with the latest access details. You even saw how to switch between data sources and update the data both selectively and globally.

Despite their usefulness in storing and structuring large quantities of information, corporate databases can present one small drawback: the time that it can take to load the data from the database into Power BI Desktop's in-memory data model. The development team at Microsoft is clearly aware of this potential shortcoming. The team has come up with a solution, called DirectQuery (or Connect Live), which you can discover how to implement in the next chapter.

CHAPTER 4

DirectQuery and Connect Live

The previous chapter showed you how to access data from a range of database and data warehouse sources. This process is both simple and efficient, as you saw. However there is one stage in the process of fetching data that can take a little time, especially if you are dealing with large datasets. This is the "load" phase where the data from the source system is transferred into the Power BI Desktop in-memory model and compressed.

The Power BI developers have clearly looked hard at this question, and have come up with a potentially far-reaching solution: connect directly to the data source and avoid having to download the data. This technique is called DirectQuery. It is worth noting that Microsoft now calls the direct data connection to SQL Server Analysis Services "Connect Live." However, I will consider this to be, nevertheless, part of the DirectQuery technological approach.

In this chapter we will take a look at when you can use DirectQuery in Power BI Desktop, and what the advantages (and, of course, any drawbacks) are to using these data connection methods.

DirectQuery and Connect Live

To begin with, you need to know that DirectQuery (and Connect Live) currently only works with a few of the available data sources that Power BI Desktop can connect to. At the time of writing, these are

- SQL Server Database
- SQL Server Analysis Services ("classic" SSAS and tabular)
- Oracle Database
- Teradata Database
- SAP HANA
- Azure SQL Database
- Azure SQL Data Warehouse

DirectQuery is different from the more traditional data load methods for the following reasons:

- You do not load the data into Power BI Desktop. Instead, you use the data directly from the database server.
- Because you are not loading a copy of the data into Power BI Desktop, you *cannot* work offline. You need to be able to connect to the source database or data warehouse to use the data.

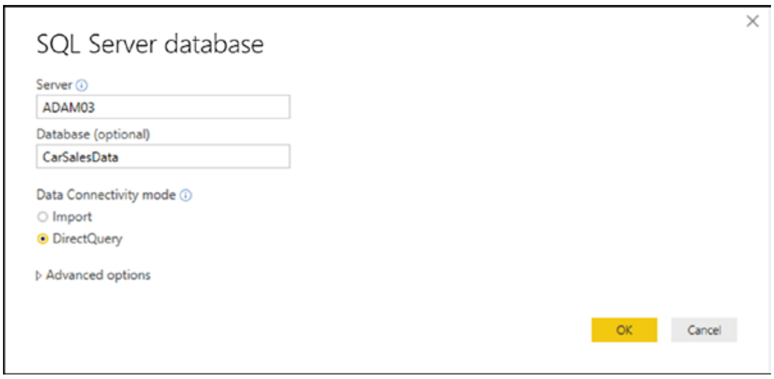
A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_4

- The connection to the source database or data warehouse—and the consequent availability of the data for analysis—is usually extremely fast, if not instantaneous.
- The data is fetched specifically for the requirements of each new visual that you create.
- It follows that you *do not* need to refresh the data source if ever the data is updated in the source database or data warehouse. The data that is available in Power BI Desktop is always the latest version of the data.
- You have all the data that is in the source database or data warehouse available.
- Data is refreshed every time you apply a slicer or a filter.
- You cannot connect to any other data source if you are using DirectQuery to connect to a database or data warehouse (without loading the data into the in-memory model). So the data source has to contain all the data that you need for your analysis if you want to use DirectQuery.

Microsoft SQL Server Data

As a first example of DirectQuery at work, I will use a Microsoft SQL Server database as the data source. The steps are as follows:

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
3. Enter the server name in the Server text box. This will be the name of your SQL Server or one of the SQL Server resources used by your organization.
4. Enter the database name; if you are using the sample data, it will be CarSalesData.
5. Select the DirectQuery button. The dialog will look like Figure 4-1.


SQL Server database

Server: ADAM03

Database (optional): CarSalesData

Data Connectivity mode: Import [✓ DirectQuery]

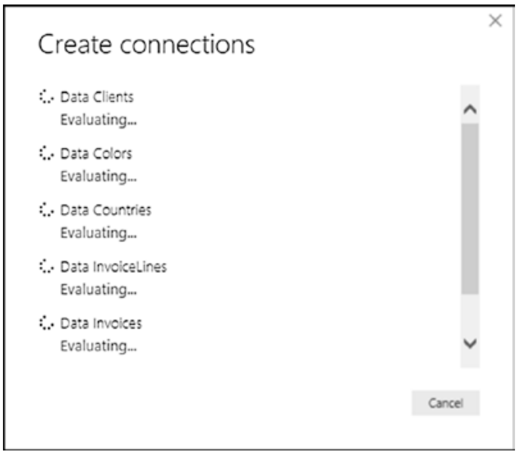
Figure 4-1. *The Microsoft SQL Server Database dialog*

6. Click OK. The Credentials dialog will appear. Define the type of credentials that you want to use, as you did in Chapter 3.

7. Click Connect. Power BI Desktop will connect to the server and display the Navigator dialog containing all the tables and views in the database that you have permission to see on the server you selected. In some cases, you could see a dialog saying that the data source does not support encryption. If you feel happy with an unencrypted connection, then click the OK button for this dialog.

8. Click the check boxes for the Clients, Colors, Countries, Invoices, InvoiceLines, and Stock tables. The data for the most recently selected data appears on the right of the Navigator dialog.

9. Click Load. Power BI Desktop will display the list of source tables for which it is establishing a connection. You can see this in Figure 4-2.


Create connections

Data Clients
Evaluating...

Data Colors
Evaluating...

Data Countries
Evaluating...

Data InvoiceLines
Evaluating...

Data Invoices
Evaluating...

Figure 4-2. *Creating DirectQuery connections*

10. The Power BI Desktop window will open and display the tables that you selected in the Fields list in the Report window. You can see this in Figure 4-3.

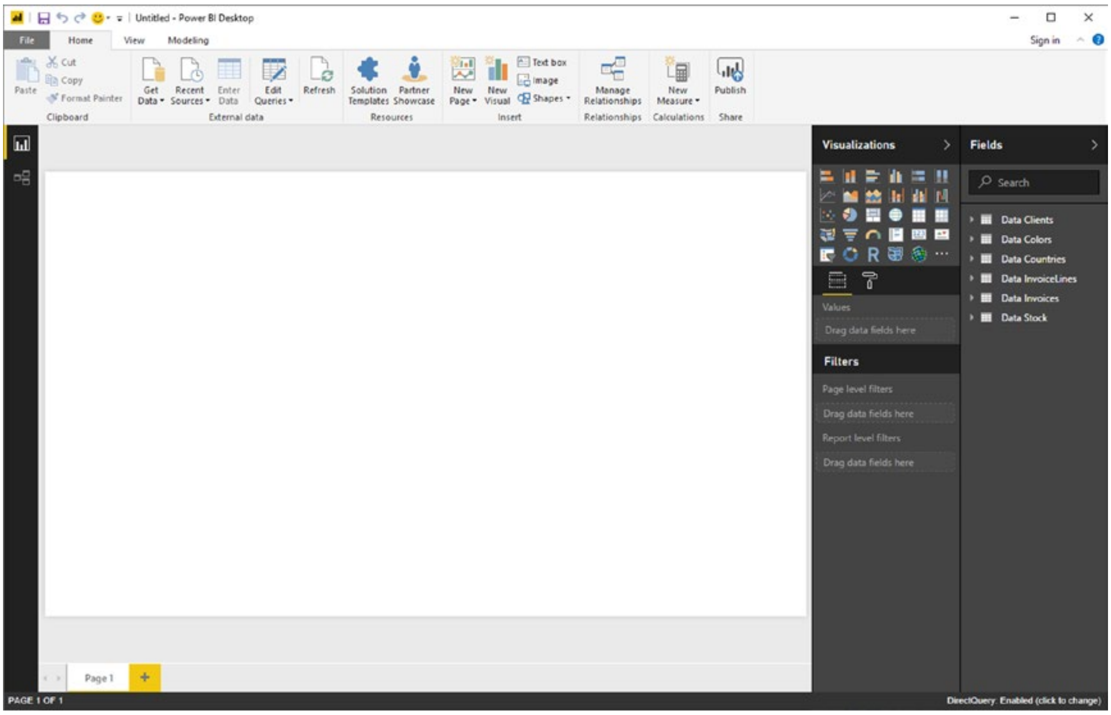


Figure 4-3. *Power BI Desktop using Direct Connection*

This is so similar to the process that you saw as the first section of the previous chapter that you can be forgiven for asking, “So what’s the difference?” Well there are a few differences, but they are so subtle as to be nearly invisible:

- There was no data load phase. When you clicked Load in step 9, the Power BI Desktop window appeared almost instantaneously. This is why the dialog that appeared briefly in step 9 says “Create Connections” instead of “Load Data.”
- The Data icon is no longer available at the top left of the Power BI Desktop window.
- When you create Power BI Desktop visuals, they take longer to populate with data and display.
- To be slightly technical, what Power BI Desktop has done here is to query only the *metadata* from the source system. As metadata is nothing more than the description of the data and the data structures (or “data about data”), the process is extremely rapid as very little information is sent back from the server to Power BI Desktop. This is why establishing a DirectQuery connection is so fast, initially.

By deciding to use DirectQuery, you have adopted a different logic to how the data is stored. Instead of copying all the selected source data into Power BI Desktop, you are leaving the data where it is (in SQL Server in this example) and only importing the data that describes the data—the metadata. So instead of needing all the data in Power BI Desktop before you can do any analysis, you can access only the data that you need, as and when you need it. Nevertheless, you can use this data as a basis for the data mashup and modeling that are described in Chapters 6–8 and Chapter 11. So you can add calculations and tweak the data (with a few limitations, at least) just as you can if you have loaded all the source data into Power BI Desktop before you begin your analysis.

In deciding to use DirectQuery you have, in essence, accepted a trade-off. Using DirectQuery implies that:

- You will gain time through *not* loading the data into the Power BI Desktop in-memory model.
- You will query the data source every time that you create, modify, or filter a Power BI Desktop visual—but *only* for the subset of the data that is required for the specific visual that you are creating or modifying.
- When you refresh the data, you will *not* reload all the data into memory as is the case when importing data. Power BI Desktop will only query the database for the *data that is actually required* to display the visuals that you have created.
- Slicing data can take longer when using DirectQuery because all the Power BI Desktop visuals are re-queried.
- Some of the data mashup techniques that you will discover in Chapters 6–8 cannot be used. I will highlight some of these when relevant in the appropriate chapters.

However, if you have a largely clean and coherent set of source data—such as data from a corporate data warehouse, where the "heavy lifting" required to make the data reliable and useable has already been carried out—then DirectQuery can really accelerate your data analysis. And you are absolutely certain to be seeing the current data, as well.

■ **Note** You can equally well use a T-SQL query or a SQL Server stored procedure to return data over a DirectQuery connection. Simply expand the Advanced Options section of the connection dialog in step 5 (see Figure 4-1) and enter or copy the SQL text to execute as described in the previous chapter for loading data from SQL Server.

To give a balanced picture (and despite a heartfelt appreciation of the usefulness of DirectQuery), I have to admit that there are a few drawbacks to this connection type that you have to be aware of:

- You cannot (for the moment at least) specify a stored procedure in the Advanced Options as the data source. Power BI Desktop will eventually return an error if you attempt this. A potential workaround is to export that data from the stored procedure in SQL Server and then connect Power BI Desktop to the table that contains the results from the stored procedure. If you need this data to be kept reasonably up to date, you can always set up a SQL Agent job to rerun the stored procedure at regular intervals. This technique might require assistance from the DBAs at your organization.
- You can only return a maximum of one million records to Power BI Desktop when using DirectQuery. Fortunately this threshold is a row limit, and not a limit on the source data. So, if you are aggregating a billion-record data source but only returning 999,999 summary records, then the query will work.
- All the source tables used in the Power BI Desktop file must come from a *single* source database. If you add a second SQL Server connection, Power BI Desktop will switch to a "classic" data load connection for the existing DirectQuery connection. In other words, you can only have a single database connection if you want to use DirectQuery.

- Relationship filtering (you will see this in Chapter 10) will only work in a single direction.
- Really complex DAX queries (you can learn about DAX in Chapters 11–13) simply will not work when using DirectQuery. The only solution to force complex queries to work is to switch back to loading the source data into the in-memory data model.
- Selecting File ➤ Options and Settings ➤ Options ➤ DirectQuery, then “Allow unrestricted measures in DirectQuery mode” will prevent Power BI Desktop applying built-in limitations to DAX expressions. However, this can make some queries extremely slow, as the conversion from DAX to SQL is not always efficient.
- Time Intelligence (explained in Chapter 13) is not available with DirectQuery.

To end on a positive note, DirectQuery does come with the following advantages:

- Reports created using DirectQuery can, of course, be published to the Power BI Service that you will meet in Chapter 23.
- The 1GB limit on the dataset size in Power BI Desktop does not apply to DirectQuery connections.

■ **Note** If Power BI Desktop has recently requested the data from the server that is required for a visualization, then it will use the existing data that has been cached to avoid placing undue stress on the source server as well as to enhance the user experience. Consequently, you need to refresh the data if you want to be sure that you are looking at the most up-to-date information and you suspect that the data in the source has been updated recently.

SQL Server Analysis Services Dimensional Data

Another data source that can use DirectQuery (which uses the variant that Microsoft calls “Connect Live”) is the SQL Server Analysis Services dimensional data warehouse-or “classic SSAS” as it is also known. Although a live connection to a classic SSAS dimensional data warehouse is very similar to loading data from SSAS into Power BI Desktop, there are a few differences that might make you prefer this method.

■ **Note** Live Connection to a tabular data warehouse will only work if you are using SQL Server 2012 SP1 CU4 or greater. In this case you have to have an Enterprise or Business Intelligence Edition unless you are using SQL Server 2016, in which case standard edition may be used.

Setting up a live connection to classic SSAS requires the following steps:

1. In the Power BI Desktop ribbon, click Get Data ➤ More ➤ Database and select SQL Server Analysis Services Database in the Get Data dialog.
2. Click Connect. The SQL Server Analysis Services Database dialog will appear.

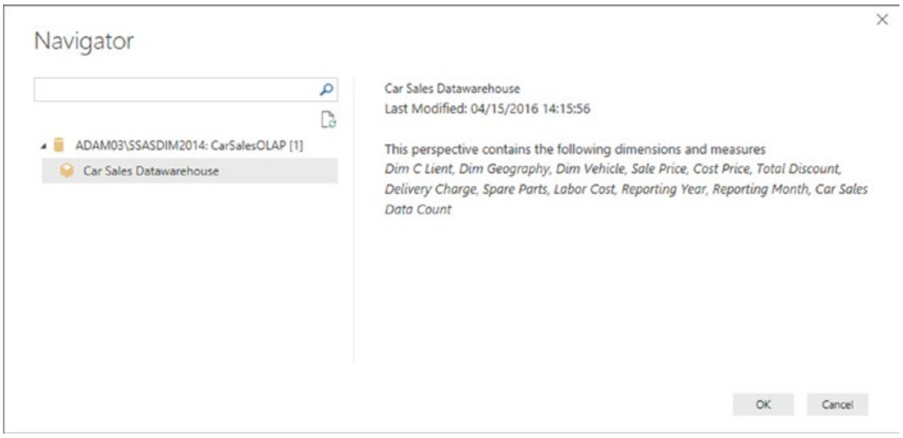
3. Enter the Analysis Services server name and the database (or “cube”) name, if you know it. If you are using the sample data from the Apress site for this book, the database is CarSalesOLAP; otherwise, you have to specify your own SSAS database name. In any case, you will need to use the name of your own SSAS server.

4. Select the Connect Live button.

5. Click OK. If this is the first time that you are connecting to the cube, then the Access SQL Server Analysis Service dialog will appear so that you can define the credentials that you are using to connect to the Analysis Services database.

6. Accept or alter the credentials and click Connect. The Navigator dialog will appear.

7. Click the cube that you want to connect to from the SSAS database. The dialog will look something like Figure 4-4.



Car Sales Datawarehouse
Last Modified: 04/15/2016 14:15:56

This perspective contains the following dimensions and measures
Dim C Lient, Dim Geography, Dim Vehicle, Sale Price, Cost Price, Total Discount, Delivery Charge, Spare Parts, Labor Cost, Reporting Year, Reporting Month, Car Sales Data Count

Figure 4-4. *Live Connection to a tabular database*

8. Click OK. The Power BI Desktop window will open and display the fact tables and dimensions that you selected in the Fields list in the Report window. It could look something like Figure 4-5.

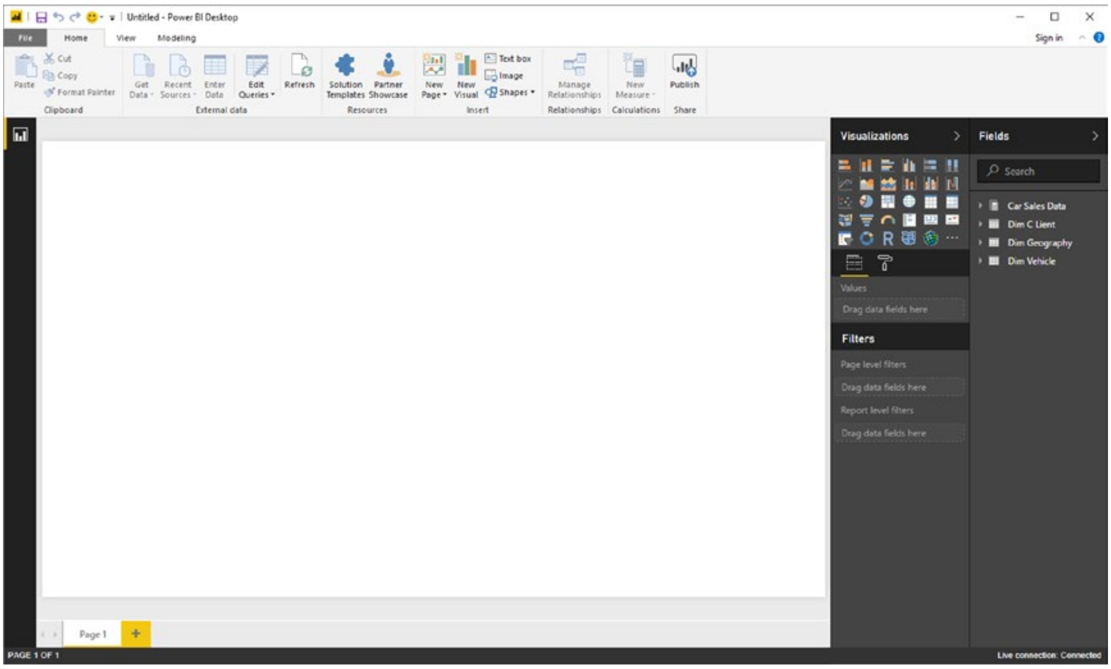


Figure 4-5. *Power BI Desktop using a Live Connection*

So, although generally similar to the process for loading data from classic SSAS into Power BI Desktop that you saw in the previous chapter, this approach does, nonetheless, manifest some differences:

- The Relationships icon is no longer available on the top left of the Power BI Desktop window.
- You were not able to select the tables to use from the tabular data source. However, the underlying structure of the Analysis Services cube is visible just as it would be from, say, Excel. This includes visibility of the folder hierarchy for data that is present in the SSAS cube.

So, overall, a live connection implies that the source data *must* be ready to use and correctly structured for you to base your Power BI Desktop analytical reports on it. You *cannot* make changes to the data or the data structures or add any calculations in Power BI Desktop.

There are a couple of other points that you might need to take into account if you are hesitating between a live connection and loading data into Power BI Desktop:

- None of the data mashup possibilities that you will learn in Chapters 6–8 are available.
- You cannot use MDX to select the data that you want to use in Power BI Desktop. So a live connection is an “all or nothing” option.
- Certain features in your cube are either not supported fully or not supported at all with Live Connections from Power BI Desktop.

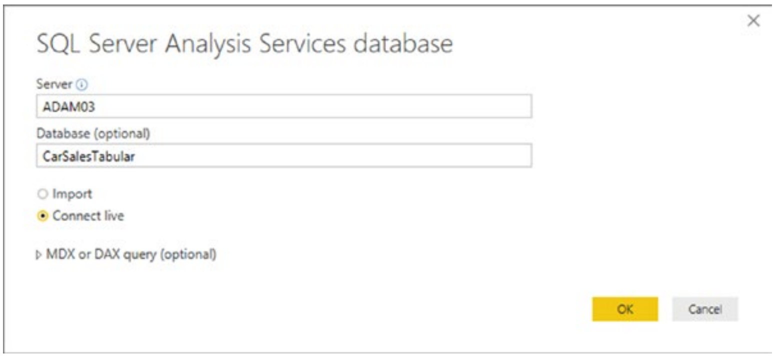
Yet, once again, a direct connection brings one crucial factor into the mix, and that is sheer speed. As SSAS data warehouses can be huge, the fact that you are not loading massive amounts of data into Power BI Desktop can save an immense amount of time. Not only that, a data warehouse that was too large to load into Power BI Desktop could now become accessible over a direct connection. Moreover, your Power BI Desktop visuals will only return the exact data that they need from the SSAS data source, as was the case for the SQL Server database in the previous section. All the hard work is carried out by the server, leaving Power BI Desktop (and you) free to concentrate on analysis and presentation.

Microsoft SQL Server Analysis Services Tabular Data Sources

Now let's see how to use a Microsoft SQL Server Analysis Services tabular data warehouse as the data source for a live connection. A SQL Server Analysis Services tabular database is another technology that is used for data warehousing. It is different from the more traditional dimensional data warehouse in that it is entirely stored in the server memory and, consequently, is usually very much faster to use.

To establish a live connection to an SSAS tabular data source:

1. In the Power BI Desktop ribbon, click Get Data ➤ More ➤ Database and select SQL Server Analysis Services Database in the Get Data dialog.
2. Click Connect. The SQL Server Analysis Services Database dialog will appear.
3. Enter the Analysis Services server name and the tabular database name (I don't tend to call these cubes), if you know it. If you are using the sample data from the Apress site for this book, the database is CarSalesTabular; otherwise, you have to specify your own tabular database name. In any case you will need to use the name of your own SSAS server.
4. Select the Connect Live button. The dialog will look like Figure 4-6.


SQL Server Analysis Services database

Server: ADAM03

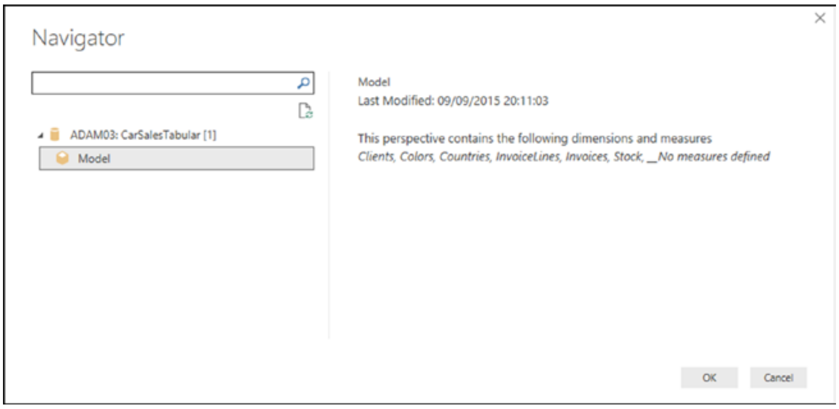
Database (optional): CarSalesTabular

Import [✓ Connect live]

Figure 4-6. *Connecting to an SSAS (multidimensional) database*

5. Click OK. If this is the first time that you are connecting to the tabular data source, then the Access SQL Server Analysis Service dialog will appear so that you can define the credentials that you are using to connect to the Analysis Services database, where you will have to accept or alter the credentials and click Connect. The Navigator dialog will appear.

6. Click the perspective (Model in this example) that you wish to connect to. The dialog will look something like Figure 4-7.



Model
Last Modified: 09/09/2015 20:11:03

This perspective contains the following dimensions and measures
Clients, Colors, Countries, InvoiceLines, Invoices, Stock, _No measures defined

Figure 4-7. *Selecting attributes and measures from an SSAS tabular source*

7. Click OK. The Power BI Desktop window will open and display the tables that you selected in the Fields list in the Report window. You can see this in Figure 4-8.

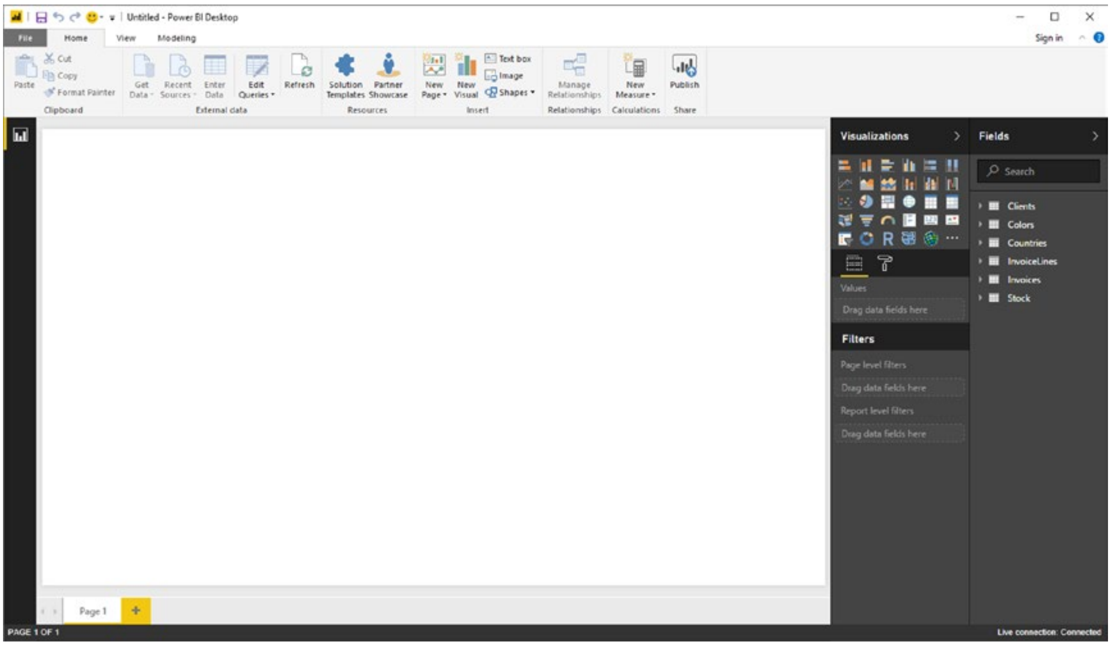


Figure 4-8. *A DirectQuery connection to an SSAS tabular data warehouse*

This process was fairly similar to the DirectQuery connection that you established in the previous section. There are, nonetheless, a couple of further differences:

- The Relationships icon is no longer available on the top left of the Power BI Desktop window.
- None of the data mashup possibilities that you will learn in Chapters 6–8 are available.
- You were not able to select the tables to use from the tabular data source.

These three points actually imply the even deeper trade-off that you have accepted when you clicked Connect Live in step 4. What you have agreed to is using the data source as the *complete and final model* for the data. You are, in effect, using Power BI Desktop as the front end for the data "as is."

This trade-off, however, is bursting with positives for you, the data analyst. You can now

- See all the data in the data warehouse (or at least the data that you are authorized to see) without a complex process where you have to select dozens of tables (and risk overlooking a few vital sources of data).
- Access the data model as it has been designed by an expert with all the relationships between the tables defined at source.
- Get the latest data—what you see in Power BI Desktop is the current data in the tabular data warehouse. You only need to refresh Power BI Desktop if the data is refreshed at source.
- Use predefined hierarchies and calculations from the tabular data warehouse.
- *Access the data at lightning speed, because the source data is stored in the host server's memory*, and not on disk (as is the case with classic relational databases).

This last point is the one that really needs emphasizing. Not only are you gaining time through not loading a copy of the data into the Power BI Desktop in-memory data model; you are accessing in-memory data on the server itself, avoiding the need for slow searches for data on disk. The overall result is unbelievably fast access to huge amounts of clean, structured, and aggregated data. All in all, the combination of a SQL Server tabular data warehouse and Power BI Desktop is designed to make data analysis considerably faster and easier.

It is worth noting that there is no real Power BI Desktop Query Editor access for a live connection to an Analysis Services tabular data warehouse. If you click the Edit Queries button, all you will see is the connection dialog for the Analysis Services database.

■ **Note** When you set up a live connection to a tabular data source, you *cannot* filter the data that is loaded from an SSAS cube by expanding the MDX or DAX query (optional) item in the SQL Server Analysis Services database dialog and entering a DAX query.

DirectQuery with Non-Microsoft Databases

So far in this chapter we have focused on using Microsoft data sources to establish DirectQuery connections to the source data. Fortunately, Microsoft has extended this technology to several other sources of corporate data.

Equally fortunate is the fact that connecting to (say) an Oracle database or an SAP HANA data warehouse using DirectQuery is every bit as simple as loading the data into Power BI Desktop from these (or indeed other) database and data warehouse sources. So I will not waste pages here in re-explaining the technique. If the source is a database, then you select DirectQuery; if the source is a data warehouse, then you can select Live Connection. A DirectQuery will allow you to specify a SQL statement to select data, and a Live Connection will display all the objects in the data warehouse. It really is that simple.

DirectQuery and In-Memory Tables

A recent addition to the Microsoft SQL Server database has been in-memory tables. These are perceived by tools like Power BI Desktop as standard tables, yet they exist in the server's memory (as opposed to storing the data on disk). This makes accessing data from in-memory tables extremely fast. When you add to this the fact that these tables can, in many cases, also use the compression technology that SSAS tabular data warehouses use, then you have pretty nearly the best of both worlds as far as analytics is concerned:

- Data stored in classic relational structures that is updated instantly
- Optimized structures for analytics (data is stored by column, rather than by row, and is both compressed and in-memory)

Here is not the place to expose all that this technology can bring to the table. However, as it is a potential game-changer, it is worth showing you how you can use the latest versions of SQL Server (that is, 2016 and up) as a direct source of analytical data. As this is essentially a rerun of the initial section of this chapter, I will not show here, again, the same screenshots of the process.

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
3. Enter the server name in the Server text box. This will be the name of your SQL Server or one of the SQL Server resources used by your organization.
4. Enter the database name; if you are using the sample data, it will be CarSalesMemoryBased.
5. Select the DirectQuery button.
6. Click OK. The Access a SQL Server Database dialog will appear. Define the type of credentials that you want to use, as you did in the previous chapter.
7. Click Connect. Power BI Desktop will connect to the server and display the Navigator dialog containing all the tables and views in the database that you have permission to see on the server you selected.
8. Click the check boxes for the Clients, Colors, Countries, Invoices, InvoiceLines, and Stock tables. The data for the most recently selected data appears on the right of the Navigator dialog.
9. Click Load. The Power BI Desktop window will open and display the tables that you selected in the Fields list in the Report window.

Yes, the process is identical to a "normal" DirectQuery connection. You will only return data when creating or modifying visuals. Yet here too, the data is stored in memory on the server, and all the heavy lifting is carried out by the server. However (and unlike when using a Live Connection to a tabular data warehouse), you can extend the data model, hide or add further columns, and create complex calculations.

DirectQuery and Refreshing the Data

You saw how to refresh data from databases and data warehouses in the previous chapter. If you are using DirectQuery, refreshing the data will likely only be necessary if the source data has changed. This is because every time that you create or modify a visual, or when you filter reports, pages, or visuals (or if you apply slicers), the data for any visual affected by the change is refreshed.

Consequently, you only have to click the Refresh button if you know that the source data has been updated. It could be, for instance, that the underlying database or data warehouse is reprocessed overnight—or even on an hourly basis. In cases like these, you should click the Refresh button once you know that the data has been updated on the server so that you can be absolutely sure that you are using the latest available data in your reports.

In the case of a DirectQuery connection to a database, you could wish only to refresh selected tables. In this case, all you have to do is to right-click the table name in the Fields list and select Refresh for each table that you want to update.

Conclusion

This short chapter extended your abilities to use databases and data warehouses as the data source for your Power BI analysis. You saw that you can choose not to load data into the Power BI Desktop in-memory data model, but can, instead, connect directly to the source data repository. Moreover, you saw that the approach to setting up a direct connection is virtually identical to the way that you set up a standard data load into Power BI Desktop.

DirectQuery allows you to avoid a potentially massive data transfer. This can save you a large amount of time and also guarantees that you are looking at the latest data. So, while it can take a little longer to design and filter your visuals, you are only using the precise data that you need on each and every occasion. Sometimes this can be the only feasible way to analyze data from terabyte-sized data warehouses.

When combined with in-memory data storage on the server (in in-memory tables or tabular data warehouses), this technique can make Power BI Desktop into a near-perfect front end for the analysis of huge and complex corporate data sources.

However, corporate databases and data warehouses are not the only sources of large-scale data that exist. More and more data is now stored in the cloud. Accessing cloud and web-based data will be the subject of the next chapter.

CHAPTER 5

Loading Data from the Web and the Cloud

In this chapter we will take a look at a subset of the fast-growing and wide-ranging set of data sources available over the Internet that you can use as a source of analytical data for Power BI Desktop. While the data sources that you will see in the following pages may be extremely diverse, they all have one thing in common: they are stored outside the enterprise and are available using an Internet connection.

The data sources that are available are available from a multitude of suppliers. Looking at all the available sources would take up an entire book, so I will show you how to access several of the mainstream services that are currently available. Once you have learned how to access a few of them, you should be able to extend the basic techniques to access just about any of the web and cloud services that can currently be used by Power BI Desktop.

Power BI Desktop is now firmly entrenched as a fundamental part of the Microsoft universe. As PowerBI.com (the cloud service that you can use to store and share dashboards) is part of Microsoft Azure, it is perhaps inevitable that the Power BI Desktop developers have gone out of their way to ensure that Power BI has become the analytical tool of choice for solutions that are hosted in Azure-the Microsoft Cloud. For this reason, I will explain quite a few of the core services that host data in MS Azure.

Nearly all of the data connections outlined in this chapter require access to a specific online source. Most of these sources are industrial-strength—and not free. However, if your enterprise is not a subscriber to these services, and you wish, nevertheless, to experiment with them, it could be worth taking a look at the free trial offers available from many (if not all) of the service providers whose offerings are outlined in this chapter.

Web and Cloud Services

Before delving into the details of some of the web and cloud services that are available, let's take an initial high-level look at what these really are. These data sources include (among many others):

- Web pages
- Online services, such as Google Analytics, Salesforce, or MS Dynamics 365
- Microsoft Azure, which covers hosting files in Azure Blob services, storing data in an Azure SQL Database, or storing data in an Azure SQL Data Warehouse (or even reading big data in Azure HD Insight)
- OData, the generic method of accessing data on the Internet

Web Pages

If you need to collect some data that you can see as a table in a web browser, you can use Power BI Desktop to connect to the URL for the page in question and then load all the data from any table on the page.

Online Services

Online services is a catch-all phrase used to describe data that you can access using the Internet. Most of the online services available to Power BI Desktop are what are called “platforms.” These are (often huge) software and data resources that either are only available online or were once housed in corporate systems but are now available as services on the Internet. There are currently dozens of online services that are available to connect to using Power BI Desktop. Indeed, the number of available services is growing at a startling pace. Some of the more frequently used include those listed in Table 5-1.

Table 5-1. *Some Online Services Available to Power BI Desktop*

Source	Comments
Salesforce Objects	Lets you access data in Salesforce.
Salesforce Reports	Lets you access the pre-structured data that underlies built-in Salesforce reports.
Google Analytics	Lets you access the data managed by Google to track web site traffic.
Facebook	Accesses Facebook data.
SharePoint Online	Connects to the Cloud version of Microsoft SharePoint.
Microsoft Exchange Online	Connects to the Cloud version of Microsoft Exchange
Dynamics 365 Online	Connects to the Cloud version of Microsoft Dynamics 365—the MS CRM and ERP solution.
OData	Although OData is not, technically, an online platform, it is certainly an online source of data. This is a standardized method for connecting to different data structures using a URL as a starting point.

■ **Note** As the number of available online services is increasing at an ever-increasing rate, you will probably find many more than those that I have listed here by the time that this book is published. Moreover, there are currently a number of online services that are available in beta. This means that you can test them, but they are not yet finalized.

Microsoft Azure

Azure is the Microsoft Cloud. The Azure data sources that Power BI Desktop can currently connect to, and can preview and load data from, are given in Table 5-2.

Table 5-2. *Azure Sources*

Source	Comments
Microsoft Azure SQL Database	Lets you connect to a Microsoft SQL Server cloud-based database and import records from all the data tables and views that you are authorized to access.
Microsoft Azure SQL data warehouse	Lets you connect to Microsoft's cloud-based, elastic, enterprise data warehouse.
Microsoft Azure Marketplace	Lets you load data that you are authorized to access on the Microsoft Azure Marketplace. It requires a Microsoft Azure Marketplace subscription.
Microsoft Azure HDInsight	Reads cloud-based Hadoop files in the Microsoft Azure environment.
Microsoft Azure Blob Storage	Reads from a cloud-based unstructured data store.
Microsoft Azure Table Storage	Reads from Microsoft Azure tables.
Azure HDInsight Spark	Lets you connect to Microsoft's parallel-processing framework in Microsoft Cloud.
Microsoft Azure DocumentDB (now called CosmosDB)	Lets you connect to Microsoft's NoSQL database.
Microsoft Azure Data Lake Store	Lets you connect to Microsoft's raw data cloud storage.

Obviously, more Azure connection options are being added to Power BI Desktop by Microsoft as the Azure offering is extended.

Web Pages

As a first and extremely simple example, let's grab some data from a web page. Since I want to concentrate on the method rather than the data, I will use a web page that has nothing to do with the sample data in the book. I will not be using this other than as a simple introduction to the process of loading data from web pages using Power BI Desktop.

Assuming that you have launched Power BI Desktop and closed the splash screen...

1. Click the small triangle at the bottom of the Get Data button in the Home ribbon.
2. Select Web from the menu that appears, as shown in Figure 5-1.

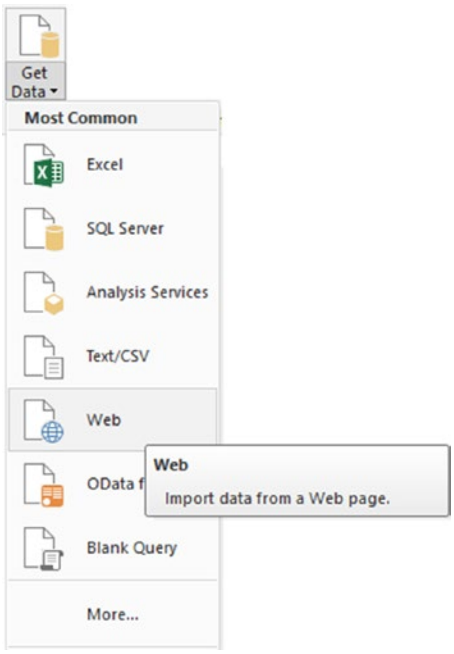


Figure 5-1. *The Get Data menu*

3. Enter the following URL (it is a Microsoft help page for Power BI Desktop that contains a few tables of data): `http://office.microsoft.com/en-gb/excel-help/guide-to-the-power-query-ribbon-HA103993930.aspx`. I am, of course, hoping that it is still available when you read this book. Of course, if you have a URL that you want to try out, then feel free! The dialog will look something like Figure 5-2.

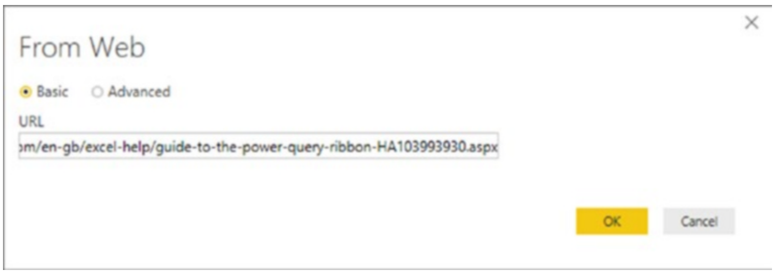

From Web
Basic
Advanced
URL
›m/en-gb/excel-help/guide-to-the-power-query-ribbon-HA103993930.aspx
OK
Cancel

Figure 5-2. *The From Web dialog*

4. Click OK. The Navigator dialog will appear. After a few seconds, during which Power BI Desktop is connecting to the web page, the list of available tables of data in the web page will be displayed.

5. Click one of the table names on the left of the Navigator dialog. The contents of the table will appear on the right of the Navigator dialog to show you what the data in the chosen table looks like, as shown in Figure 5-3.

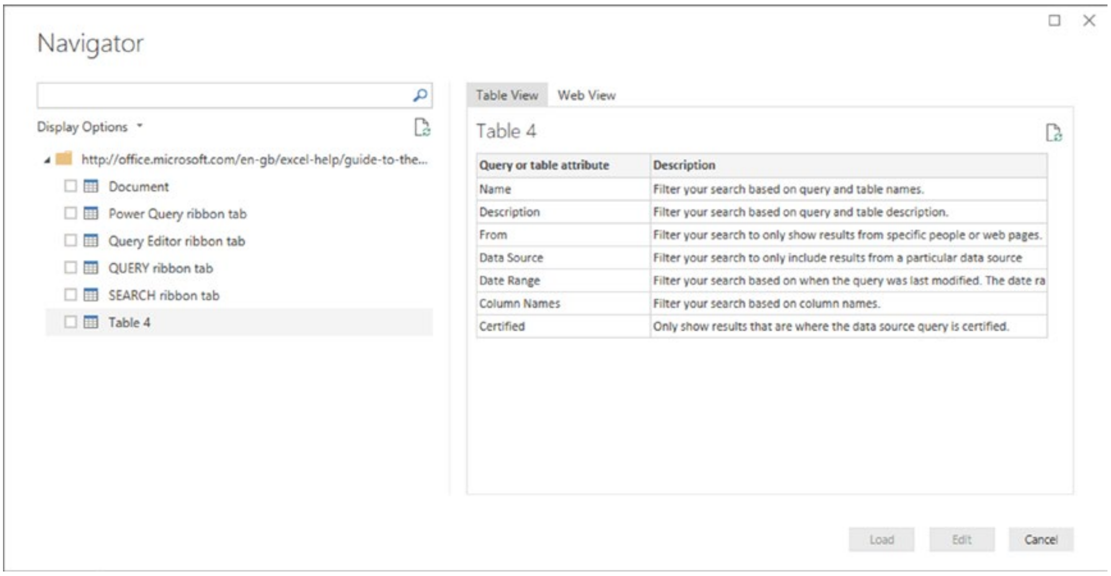
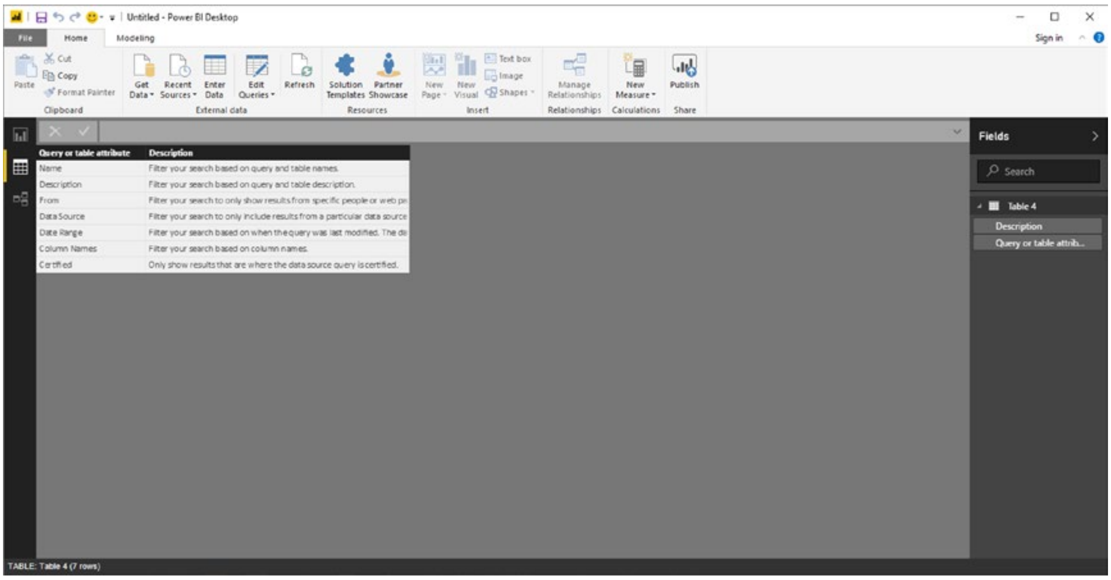


Table 4

Query or table attribute	Description
Name	Filter your search based on query and table names.
Description	Filter your search based on query and table description.
From	Filter your search to only show results from specific people or web pages.
Data Source	Filter your search to only include results from a particular data source
Date Range	Filter your search based on when the query was last modified. The date ra
Column Names	Filter your search based on column names.
Certified	Only show results that are where the data source query is certified.

Figure 5-3. *The Navigator dialog previewing the contents of a table on a web page*

6. Select the check box in the Navigator dialog (shown to the left of Table 4 in Figure 5-3).
7. Click Load at the bottom of the window (or double-click the table name).
8. Click the Data icon on the top left of the Power BI Desktop window to display the table of data. It should look like Figure 5-4.



Query or table attribute	Description
Name	Filter your search based on query and table names.
Description	Filter your search based on query and table description.
From	Filter your search to only show results from specific people or web pr
Data Source	Filter your search to only include results from a particular data source
Date Range	Filter your search based on when the query was last modified. The da
Column Names	Filter your search based on column names.
Certified	Only show results that are where the data source query is certified.

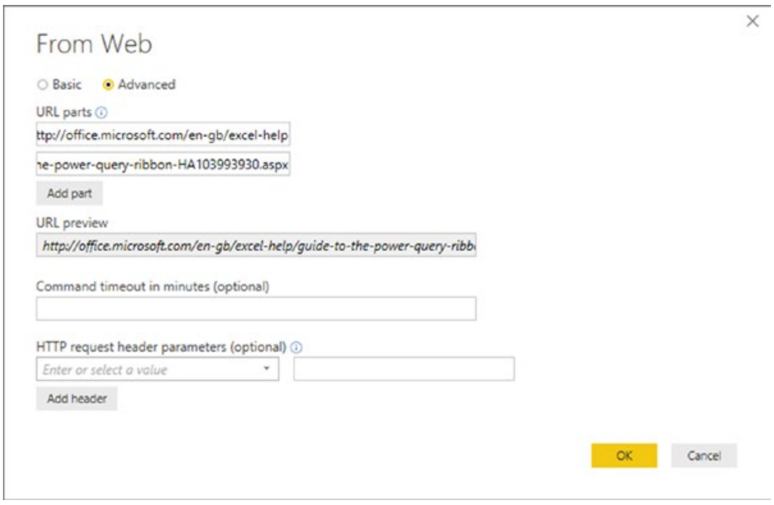
Figure 5-4. *The Power BI Desktop Query window*

■ **Tip** Another way of accessing web pages is to click Get Data ➤ Other. You can then select Web in the list on the right of the Get Data dialog.

This simple example showed how you can load data from a supported data source and load it into Power BI Desktop Query.

Advanced Web Options

In step 3 of the previous example, you could have selected the Advanced button. Had you done this, the From Web dialog would have expanded to allow you to build complex URLs by adding URL parts. You can see an example of this in Figure 5-5.


From Web

○ Basic ● Advanced

URL parts ⓘ

ttp://office.microsoft.com/en-gb/excel-help

ne-power-query-ribbon-HA103993930.aspx

Add part

URL preview

http://office.microsoft.com/en-gb/excel-help/guide-to-the-power-query-ribb

Command timeout in minutes (optional)

HTTP request header parameters (optional) ⓘ

Enter or select a value

Add header

OK Cancel

Figure 5-5. *The Advanced options in the From Web dialog*

Clicking the Add Part button allows you to define multiple URL parts.

If necessary, you can also specify HTTP request header parameters that will be used when submitting the URL. These could be required by certain web pages. A discussion of these is outside the scope of this book.

Table View or Web View

Looking at the tables that a web page contains is not always the most natural way of finding the right data. This is because you are looking at the data tables out of context. By this I mean that you cannot see where they are on the web page. After all, the Web is a very visual medium.

To help you find the correct data table on a web page, the Query Editor lets you switch between two views of the web source:

- Web view (which you saw in Figure 5-3)
- Table view (which you can see in Figure 5-6)

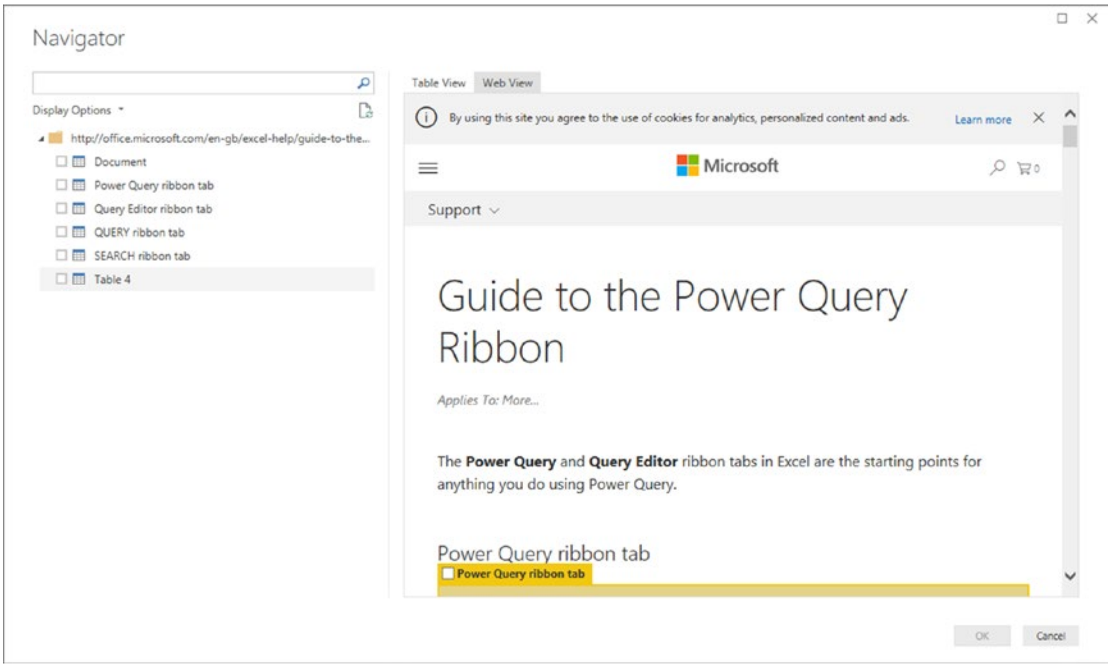

Navigator
Display Options
http://office.microsoft.com/en-gb/excel-help/guide-to-the...
Document
Power Query ribbon tab
Query Editor ribbon tab
QUERY ribbon tab
SEARCH ribbon tab
Table 4
Table View
Web View
By using this site you agree to the use of cookies for analytics, personalized content and ads.
Learn more
Microsoft
Support
Guide to the Power Query Ribbon
Applies To: More...
The Power Query and Query Editor ribbon tabs in Excel are the starting points for anything you do using Power Query.
Power Query ribbon tab
OK
Cancel

Figure 5-6. *Web View in the Navigator dialog*

You alternate between these ways of visualizing the web page by clicking the Table View and Web View buttons that are at the top center of the Navigator dialog. The same web page that you saw previously looks like Figure 5-6 when you switch to Web View.

Salesforce

One of the pioneers in the online services space—and now, indisputably, one of the leaders—is Salesforce. So it is perhaps inevitable that Power BI Desktop will allow you to connect to Salesforce and load any data that you have permission to view using your Salesforce account.

Indeed, Salesforce is such a wide-ranging and complete service that you have two possible methods of accessing your data:

- Objects
- Reports

Briefly, Salesforce objects are the underlying data tables that contain the information that you want to access. Salesforce reports are the data that has been collated from the data tables into a more accessible form of output.

■ **Tip** If you do not have a corporate Salesforce account but want, nevertheless, to see how to use Power BI Desktop to connect to Salesforce data, you can always set up a free 30-day trial account. The URL for this is `https://www.salesforce.com/form/signup/freetrial-sales.jsp`.

Loading Data from Salesforce Objects

Assuming, then, that you have a valid Salesforce account, here is how you can load data from Salesforce objects into Power BI Desktop:

1. In the Power BI Desktop Home ribbon, click the Get Data button.
2. Click Online Services on the left, and then select Salesforce Objects on the right. The Get Data dialog will look like Figure 5-7.

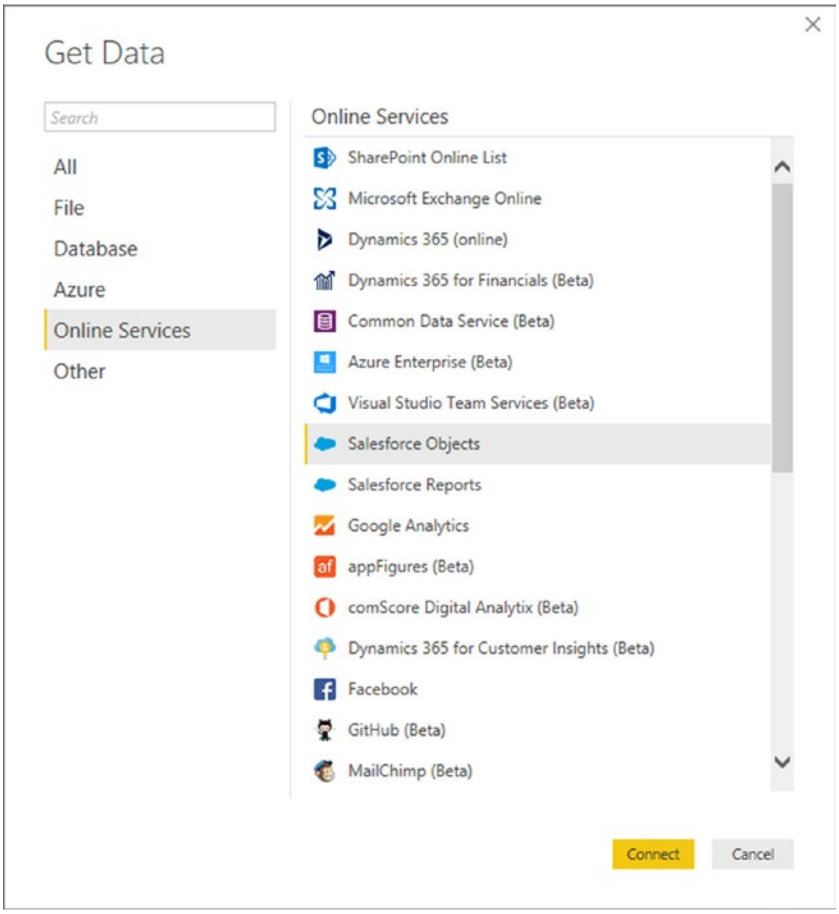


Figure 5-7. *The Get Data dialog for online services*

3. Click Connect. The Salesforce Objects dialog will appear. It should look like the one shown in Figure 5-8.

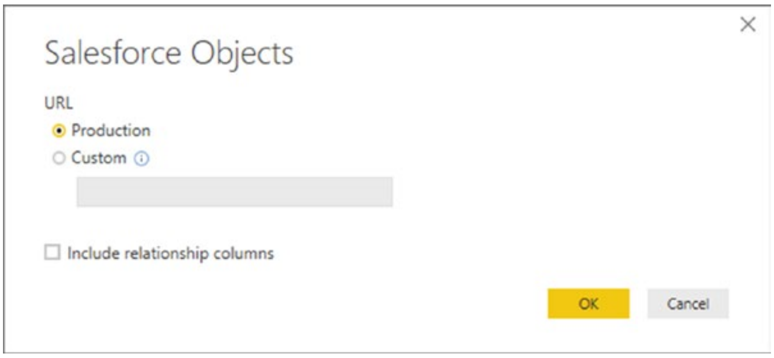


Figure 5-8. *The Salesforce Objects dialog*

4. Select the Production button and click OK. The Access Salesforce login dialog will appear. It should look like the one shown in Figure 5-9.

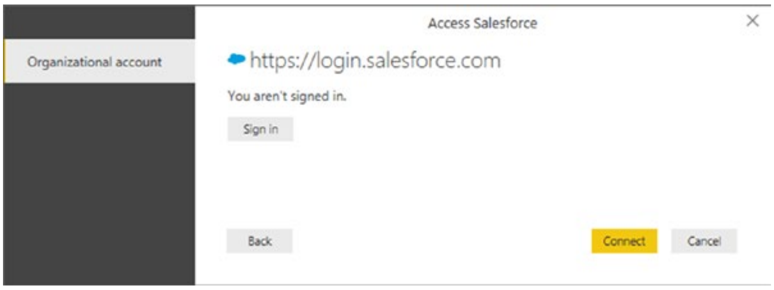


Figure 5-9. *The Access Salesforce login dialog*

5. Unless you are already signed in, click Sign in. The Salesforce sign-in dialog will appear.
6. Enter your Salesforce login and password. The dialog should look something like the one shown in Figure 5-10.

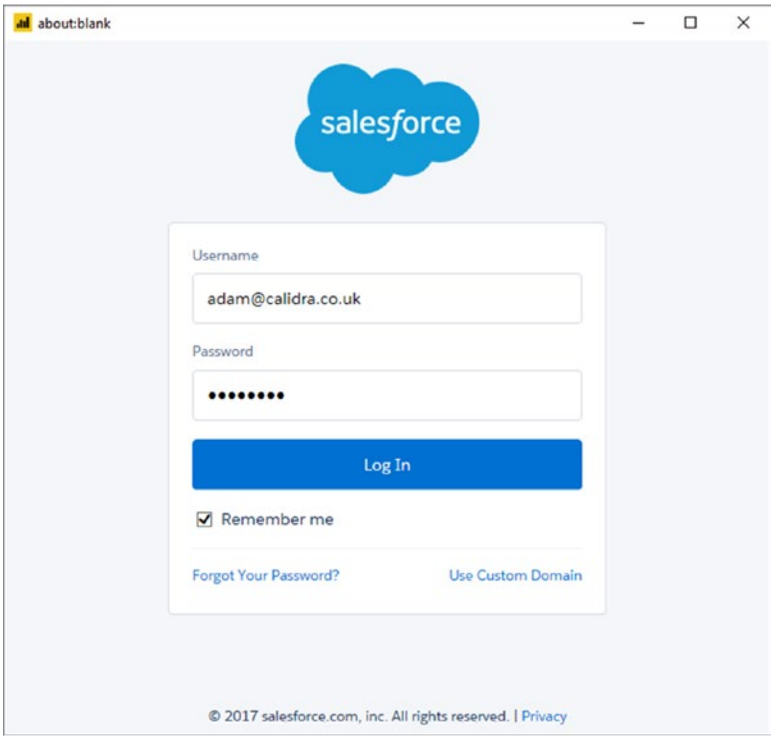
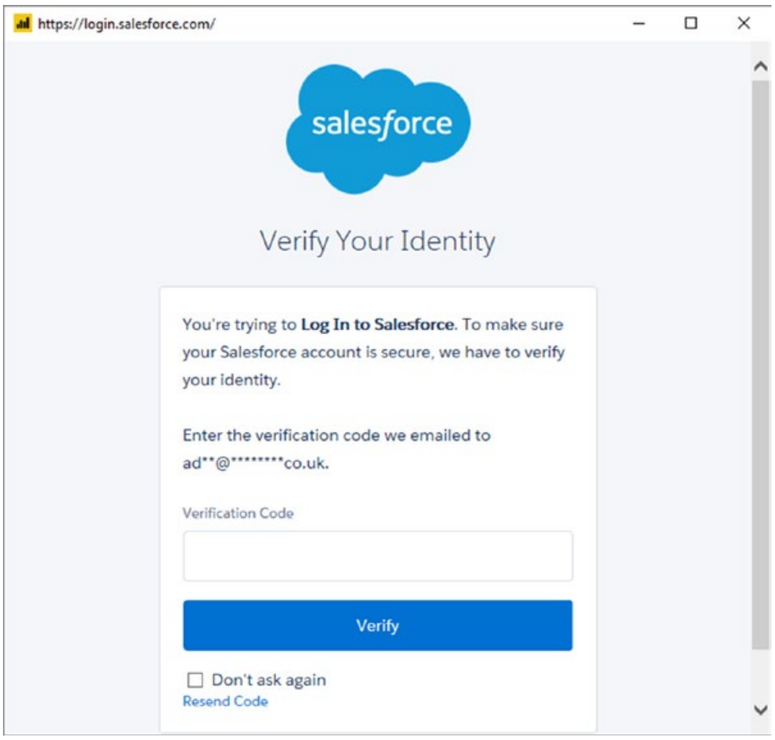


Figure 5-10. *The Salesforce sign-in dialog*

7. If this is the first time that you are connecting to Salesforce from Power BI Desktop (or if you have requested that Salesforce request confirmation each time that you log in), you will be asked to verify your identity. The Salesforce Verify Your Identity dialog will appear, as shown in Figure 5-11.


https://login.salesforce.com/

Verify Your Identity

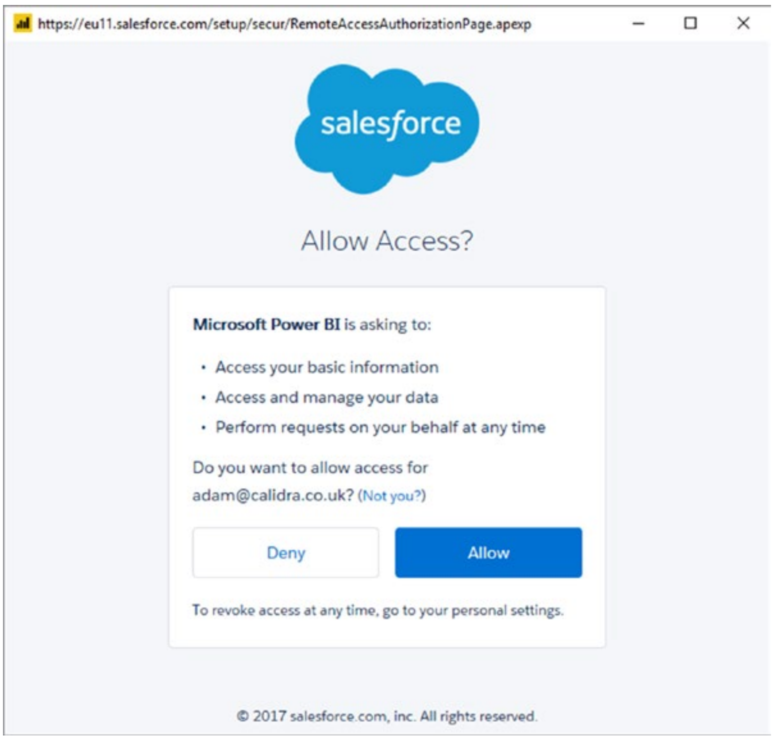
You're trying to **Log In to Salesforce**. To make sure your Salesforce account is secure, we have to verify your identity.

Enter the verification code we emailed to ad**@********co.uk.

Verification Code

Figure 5-11. *The Salesforce Verify Your Identity dialog*

8. Click Verify. Salesforce will send a verification code to the e-mail account that you are using to log in to Salesforce.
9. Enter the code in the Verification Code field and click OK. You will see the Allow Access dialog, as in Figure 5-12.


https://eu11.salesforce.com/setup/secur/RemoteAccessAuthorizationPage.apexp
salesforce
Deny
Allow

Allow Access?

Microsoft Power BI is asking to:

- Access your basic information
- Access and manage your data
- Perform requests on your behalf at any time

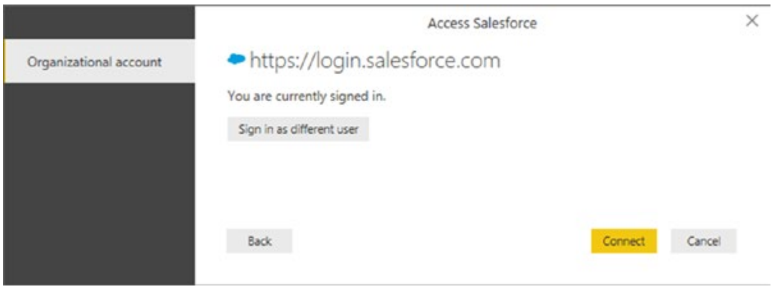
Do you want to allow access for adam@calidra.co.uk? (Not you?)

To revoke access at any time, go to your personal settings.

© 2017 salesforce.com, inc. All rights reserved.

Figure 5-12. *The Salesforce Allow Access dialog*

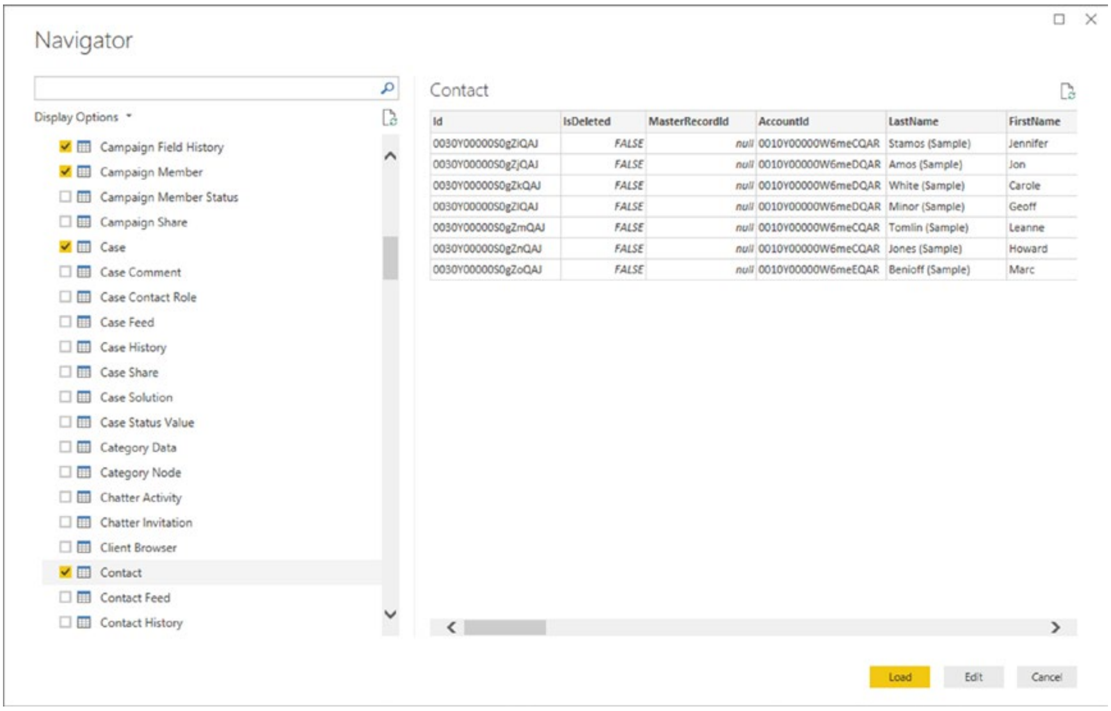
10. Click Allow. You will return to the Access Salesforce dialog, only now you are logged in. You can see this in Figure 5-13.



You are currently signed in.

Figure 5-13. *The Access Salesforce dialog*

11. Click Connect. The Navigator will appear, showing the Salesforce objects that you have permissions to access. You can see some of the objects that are available if you are using a trial account in Figure 5-14.



Contact

Id	IsDeleted	MasterRecordId	AccountId	LastName	FirstName
0030Y00000S0gZIQAJ	FALSE	null	0010Y00000W6meCQAR	Stamos (Sample)	Jennifer
0030Y00000S0gZjQAJ	FALSE	null	0010Y00000W6meDQAR	Amos (Sample)	Jon
0030Y00000S0gZkQAJ	FALSE	null	0010Y00000W6meDQAR	White (Sample)	Carole
0030Y00000S0gZlQAJ	FALSE	null	0010Y00000W6meDQAR	Minor (Sample)	Geoff
0030Y00000S0gZmQAJ	FALSE	null	0010Y00000W6meCQAR	Tomlin (Sample)	Leanne
0030Y00000S0gZnQAJ	FALSE	null	0010Y00000W6meCQAR	Jones (Sample)	Howard
0030Y00000S0gZoQAJ	FALSE	null	0010Y00000W6meEQAR	Benioff (Sample)	Marc

Figure 5-14. *The Navigator dialog showing Salesforce Objects*

12. Select the objects whose data you wish to load into Power BI Desktop and click Load. The data will be loaded into Power BI Desktop ready for you to create dashboards and reports based on your Salesforce data.

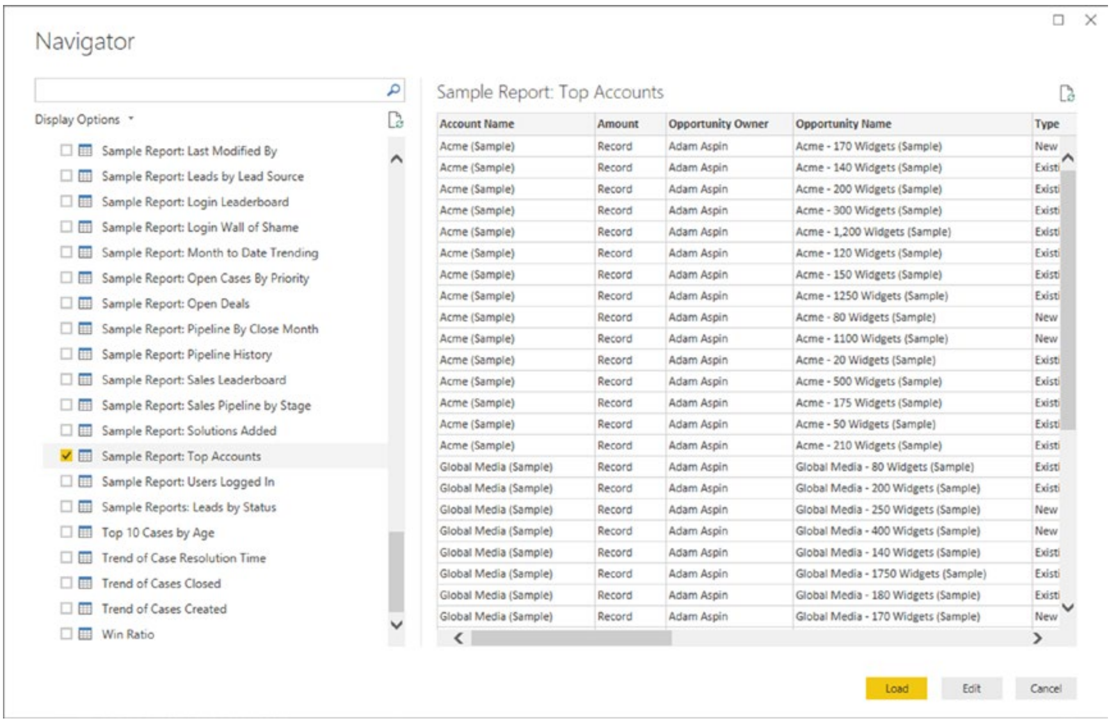
■ **Tip** To avoid having to confirm your identity to Salesforce every time that you create a new suite of Power BI Desktop reports using Salesforce data, you can check “Remember me” in the Salesforce sign-in dialog and “Don’t ask again” in the Salesforce Verify Your Identity dialog.

Salesforce objects contain a vast amount of data. However, you are, in effect, accessing a database structure. This means that you have to have some understanding of how the underlying data is stored. Should you wish to learn about the way that Salesforce data is structured, then I suggest that you start with the Salesforce documentation currently available at `https://trailhead.salesforce.com/en/modules/data_modeling/units/objects_intro`.

Salesforce Reports

If you find that you are simply submerged by the amount of data that is available in Salesforce, you can, instead, go directly to the data that underlies standard Salesforce reports. This will avoid your having to learn about the underlying data structures. The downside is that you cannot easily extend these datasets.

To access Salesforce report data, simply follow the steps outlined in the previous section. However, instead of choosing Salesforce Objects in step 2, select Salesforce Reports instead. The Navigator dialog will, in this case, look something like the one shown in Figure 5-15.



- ☐ Sample Report: Last Modified By
- ☐ Sample Report: Leads by Lead Source
- ☐ Sample Report: Login Leaderboard
- ☐ Sample Report: Login Wall of Shame
- ☐ Sample Report: Month to Date Trending
- ☐ Sample Report: Open Cases By Priority
- ☐ Sample Report: Open Deals
- ☐ Sample Report: Pipeline By Close Month
- ☐ Sample Report: Pipeline History
- ☐ Sample Report: Sales Leaderboard
- ☐ Sample Report: Sales Pipeline by Stage
- ☐ Sample Report: Solutions Added
- ☑ Sample Report: Top Accounts
- ☐ Sample Report: Users Logged In
- ☐ Sample Reports: Leads by Status
- ☐ Top 10 Cases by Age
- ☐ Trend of Case Resolution Time
- ☐ Trend of Cases Closed
- ☐ Trend of Cases Created
- ☐ Win Ratio

Sample Report: Top Accounts

Account Name	Amount	Opportunity Owner	Opportunity Name	Type
Acme (Sample)	Record	Adam Aspin	Acme - 170 Widgets (Sample)	New
Acme (Sample)	Record	Adam Aspin	Acme - 140 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 200 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 300 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 1,200 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 120 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 150 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 1250 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 80 Widgets (Sample)	New
Acme (Sample)	Record	Adam Aspin	Acme - 1100 Widgets (Sample)	New
Acme (Sample)	Record	Adam Aspin	Acme - 20 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 500 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 175 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 50 Widgets (Sample)	Existi
Acme (Sample)	Record	Adam Aspin	Acme - 210 Widgets (Sample)	Existi
Global Media (Sample)	Record	Adam Aspin	Global Media - 80 Widgets (Sample)	Existi
Global Media (Sample)	Record	Adam Aspin	Global Media - 200 Widgets (Sample)	Existi
Global Media (Sample)	Record	Adam Aspin	Global Media - 250 Widgets (Sample)	New
Global Media (Sample)	Record	Adam Aspin	Global Media - 400 Widgets (Sample)	New
Global Media (Sample)	Record	Adam Aspin	Global Media - 140 Widgets (Sample)	Existi
Global Media (Sample)	Record	Adam Aspin	Global Media - 1750 Widgets (Sample)	Existi
Global Media (Sample)	Record	Adam Aspin	Global Media - 180 Widgets (Sample)	Existi
Global Media (Sample)	Record	Adam Aspin	Global Media - 170 Widgets (Sample)	New

Figure 5-15. *The Navigator dialog showing the data for Salesforce Reports*

From here you can select and load the reports data from Salesforce that you want to use to create your own dashboards.

Microsoft Dynamics 365

Another online service that contains much valuable enterprise data is Microsoft Dynamics 365. As you would probably expect, Power BI Desktop can connect easily to Microsoft online sources such as Dynamics. Here is how to do this:

■ **Tip** If you do not have a corporate Microsoft Dynamics 365 online account but want, nevertheless, to see how to use Power BI Desktop to connect to Microsoft Dynamics 365 data, you can always set up a free 30-day trial account. The URL for this is `https://trials.dynamics.com/CustomerEngagement/ChangeSignup/`. Indeed, this example is from using a free 30-day trial account (that will likely have expired long before this book is in print).

---

1. In the Power BI Desktop Home ribbon, click the Get Data button.
2. Click Online Services on the left, and then select Dynamics 365 (online) on the right. The Get Data dialog will look like Figure 5-16.

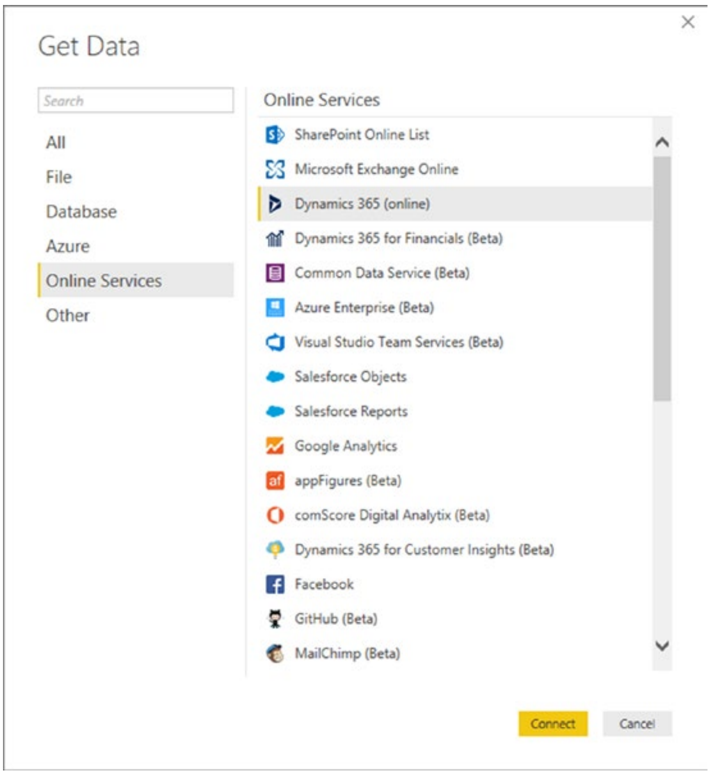
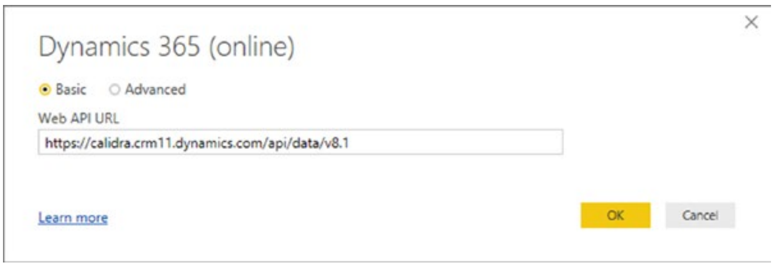


Figure 5-16. *The Get Data dialog for Dynamics 365*

3. Click Connect. The Dynamics 365 (online) dialog will appear.
4. Enter the URL that you use to connect to Dynamics 365 and add `/api/data/v8.1` (at least, this was the case as this book went to press). It could look like the one shown in Figure 5-17. Note, however, that this URL will vary depending on where you are in the world.



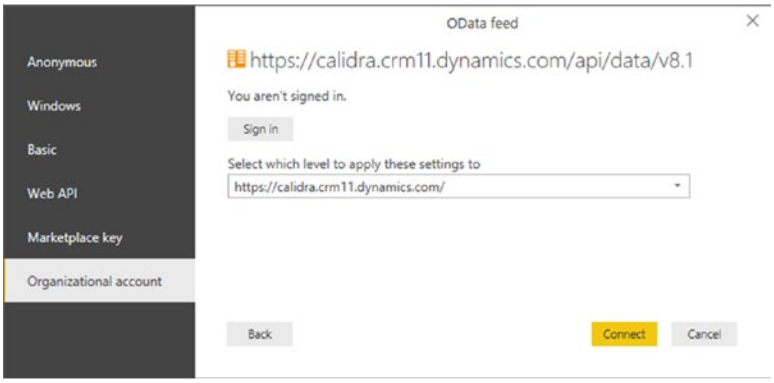
Web API URL

https://calidra.crm11.dynamics.com/api/data/v8.1

Figure 5-17. *The Dynamics 365 (online) dialog*

5. Click OK. The OData Feed dialog will appear.

6. Select Organizational Account as the security access method. The OData Feed dialog will look like Figure 5-18.



https://calidra.crm11.dynamics.com/api/data/v8.1

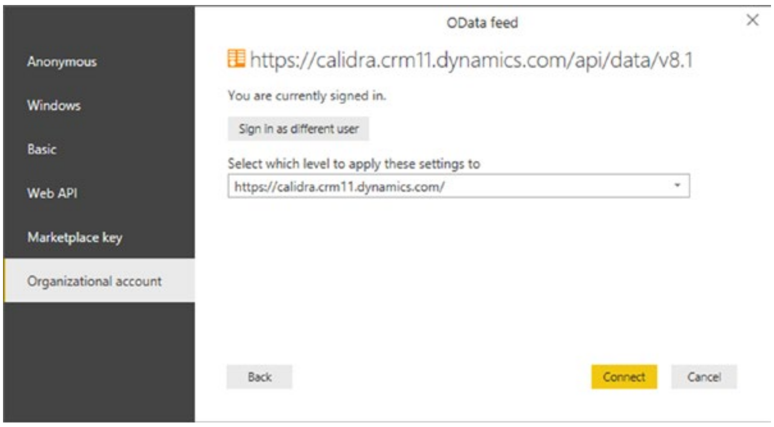
You aren't signed in.

Select which level to apply these settings to

https://calidra.crm11.dynamics.com/

Figure 5-18. *The OData Feed dialog*

7. Click Sign In to sign in to your Dynamics 365 account and follow the Microsoft sign-in process. Once completed, the OData Feed dialog will look something like the one in Figure 5-19.



https://calidra.crm11.dynamics.com/api/data/v8.1

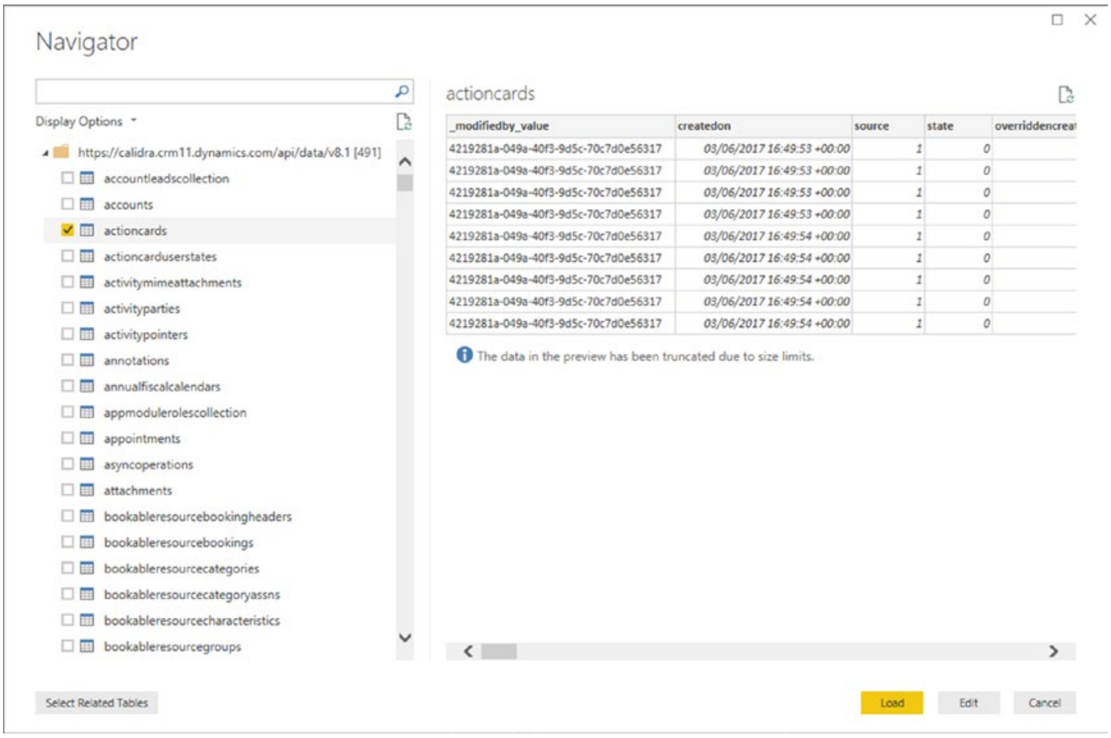
You are currently signed in.

Select which level to apply these settings to

https://calidra.crm11.dynamics.com/

Figure 5-19. *The OData Feed dialog after sign-in*

8. Click Connect. The Navigator dialog will appear showing all the Dynamics objects that you have permissions to connect to. You can see an example of this in Figure 5-20.


Navigator

Display Options

https://calidra.crm11.dynamics.com/api/data/v8.1 [491]

- [ ] accountleadscollection
- [ ] accounts
- [x] actioncards
- [ ] actioncarduserstates
- [ ] activitymimeattachments
- [ ] activityparties
- [ ] activitypointers
- [ ] annotations
- [ ] annualfiscalcalendars
- [ ] appmodulerolescollection
- [ ] appointments
- [ ] asyncoperations
- [ ] attachments
- [ ] bookableresourcebookingheaders
- [ ] bookableresourcebookings
- [ ] bookableresourcecategories
- [ ] bookableresourcecategoryassns
- [ ] bookableresourcecharacteristics
- [ ] bookableresourcegroups

actioncards

_modifiedby_value	createdon	source	state	overriddencreat
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:53 +00:00	1	0	
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:53 +00:00	1	0	
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:53 +00:00	1	0	
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:53 +00:00	1	0	
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:54 +00:00	1	0	
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:54 +00:00	1	0	
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:54 +00:00	1	0	
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:54 +00:00	1	0	
4219281a-049a-40f3-9d5c-70c7d0e56317	03/06/2017 16:49:54 +00:00	1	0	

The data in the preview has been truncated due to size limits.

Figure 5-20. *The Navigator dialog for Dynamics 365*

■ **Note** In step 6 you saw that an MS Dynamics 365 connection is really an OData connection. OData is explained in more detail in a subsequent section of this chapter.

There are a huge number of Dynamics 365 tables—and this number will vary depending on the subscription that your organization has taken out. However, you are, in reality, accessing a database structure. This means that you have to have some understanding of how the underlying data is stored. Should you wish to learn about Dynamics 365 tables, then I suggest that you start with the Microsoft online help at `https://docs.microsoft.com/en-us/dynamics365/unified-operations/dev-itpro/data-entities/data-entities`.

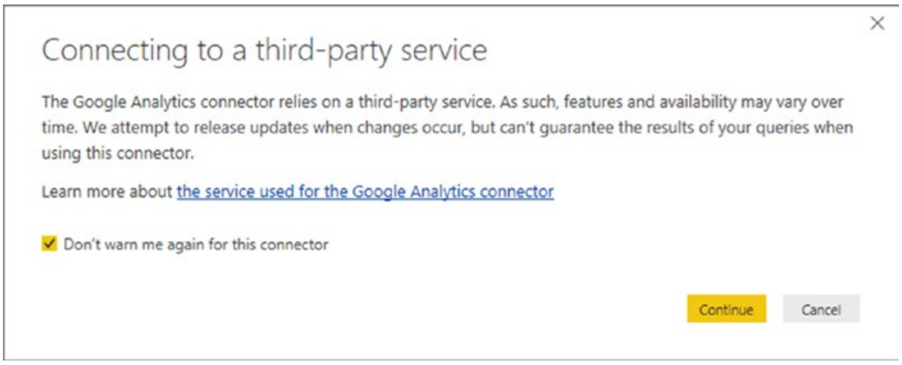
Google Analytics

Assuming that you have a valid Google Analytics account set up, you can use Power BI Desktop to connect to the Google Analytics data that you have permissions to access. For this example to work, you will need a valid and functioning Google Analytics account.

■ **Note** To sign up for a Google Analytics account that you can use to test Power BI Desktop, go to `https://www.google.com/analytics`.

---

1. In the Power BI Desktop Home ribbon, click the Get Data button.
2. Click Online Services on the left, and then select Google Analytics on the right. The Connecting to a third part service dialog will look like Figure 5-21.



Connecting to a third-party service

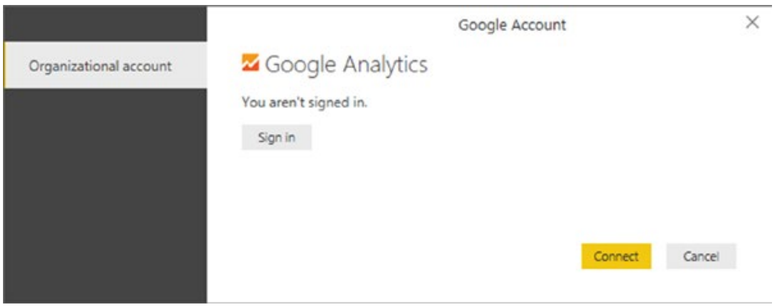
The Google Analytics connector relies on a third-party service. As such, features and availability may vary over time. We attempt to release updates when changes occur, but can't guarantee the results of your queries when using this connector.

Learn more about the service used for the Google Analytics connector

✓ Don't warn me again for this connector

Figure 5-21. The third-party service connector alert

3. Click Continue. The Google Account dialog will appear. This currently looks like the one in Figure 5-22.



You aren't signed in.

Figure 5-22. The Google Analytics connection dialog

4. Click Sign In. The Google Choose an Account dialog will be displayed. This currently looks like the image in Figure 5-23.

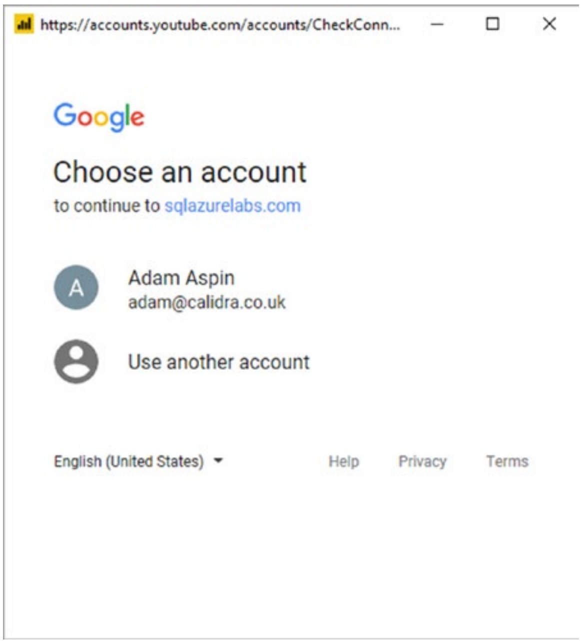
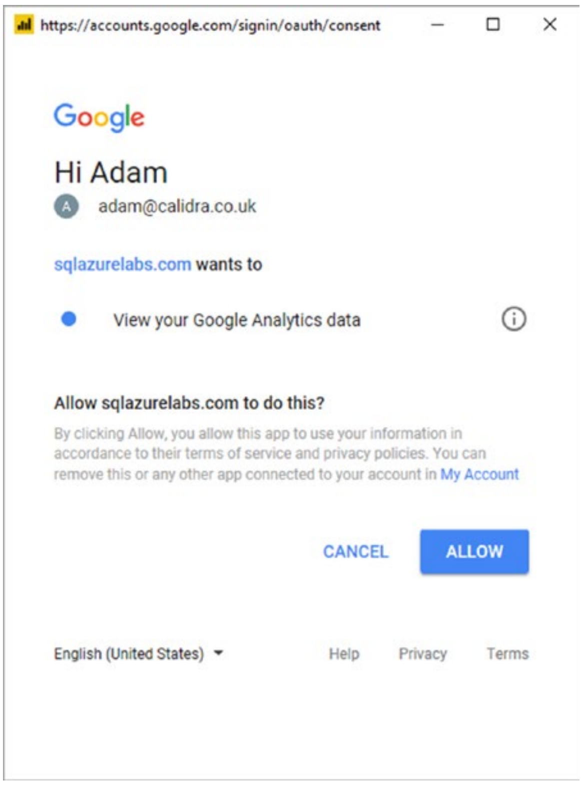


Figure 5-23. *The Google Analytics login dialog*

5. Click the existing account to use for Google Analytics. The Google Analytics permissions dialog will appear. This currently looks like the one in Figure 5-24.


https://accounts.google.com/signin/oauth/consent

Google

Hi Adam

adam@calidra.co.uk

sqlazurelabs.com wants to

View your Google Analytics data

Allow sqlazurelabs.com to do this?

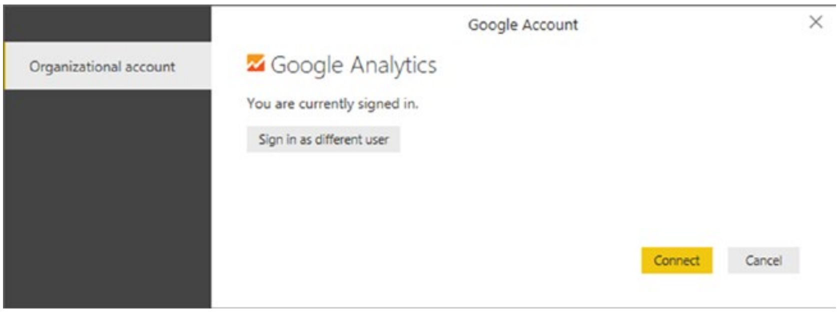
By clicking Allow, you allow this app to use your information in accordance to their terms of service and privacy policies. You can remove this or any other app connected to your account in My Account

CANCEL ALLOW

English (United States) Help Privacy Terms

Figure 5-24. *The Google Analytics permissions dialog*

6. Click Allow. You will return to the Google Account dialog, but logged in this time. You can see this in Figure 5-25.


Google Account

Organizational account

Google Analytics

You are currently signed in.

Sign in as different user

Connect Cancel

Figure 5-25. *The Google Analytics dialog when signed in*

7. Click Connect. The Navigator dialog will appear displaying the data tables that you can connect to in Google Analytics.

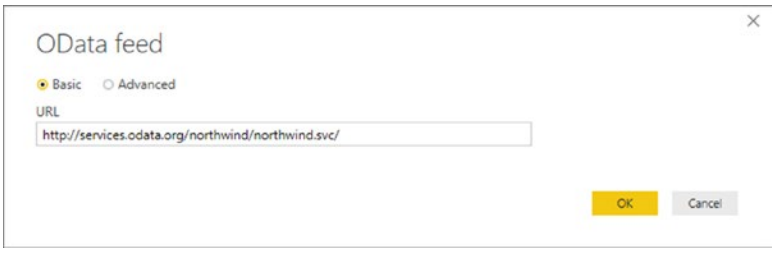
OData Feeds

OData is a short way of referring to the Open Data Protocol. This protocol allows web clients to publish and edit resources, identified as URLs. The data that you connect to using OData can be in a tabular format or indeed in different structures.

OData is something of a generic method of connecting to web-based data. Consequently each OData source could differ from others that you may have used previously. Indeed, you have already seen OData when connecting to Dynamics 365.

However, there are a multitude of OData sources that are available. Some are public, some are only accessible if you have appropriate permissions. However, the access method will always be broadly similar. Here, then, is an example of how to connect to an OData sample source that Microsoft has made freely available:

1. In the Power BI Desktop Home ribbon, click the small triangle at the bottom of the Get Data button.
2. Select OData Feed from the menu. The OData Feed dialog will appear.
3. Enter the URL that you are using to connect to the OData source. In this example I will use a Microsoft sample OData feed that you can find at `http://services.odata.org/northwind/northwind.svc`. The dialog should look like Figure 5-26.

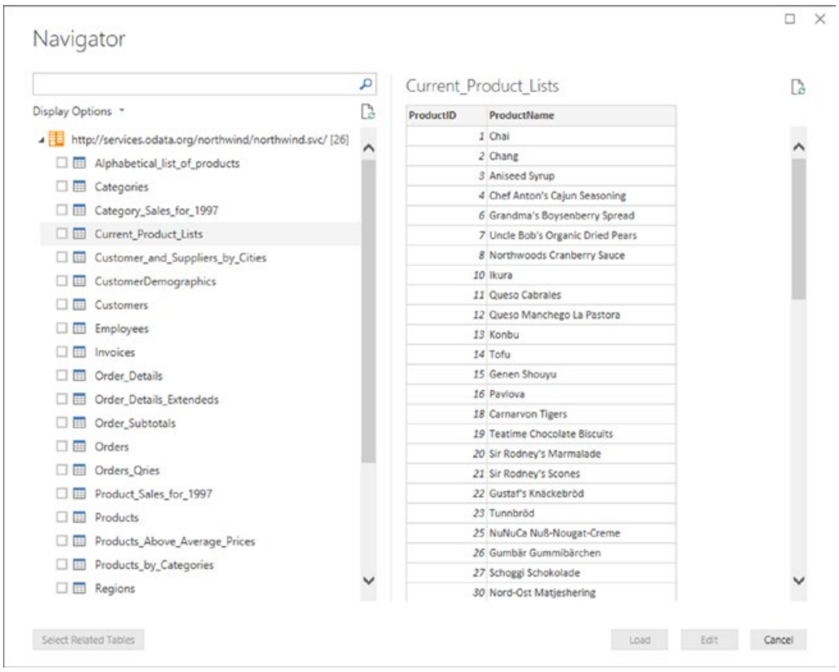


● Basic ○ Advanced

URL: http://services.odata.org/northwind/northwind.svc/

***Figure 5-26.** The OData Feed dialog*

4. Click OK. The Navigator dialog will be displayed and will show the data available using the specified URL. An example is given in Figure 5-27.


Navigator

Display Options ▾

http://services.odata.org/northwind/northwind.svc/ [26]
- ☐ Alphabetical_list_of_products
- ☐ Categories
- ☐ Category_Sales_for_1997
- ☐ Current_Product_Lists
- ☐ Customer_and_Suppliers_by_Cities
- ☐ CustomerDemographics
- ☐ Customers
- ☐ Employees
- ☐ Invoices
- ☐ Order_Details
- ☐ Order_Details_Extendeds
- ☐ Order_Subtotals
- ☐ Orders
- ☐ Orders_Qries
- ☐ Product_Sales_for_1997
- ☐ Products
- ☐ Products_Above_Average_Prices
- ☐ Products_by_Categories
- ☐ Regions

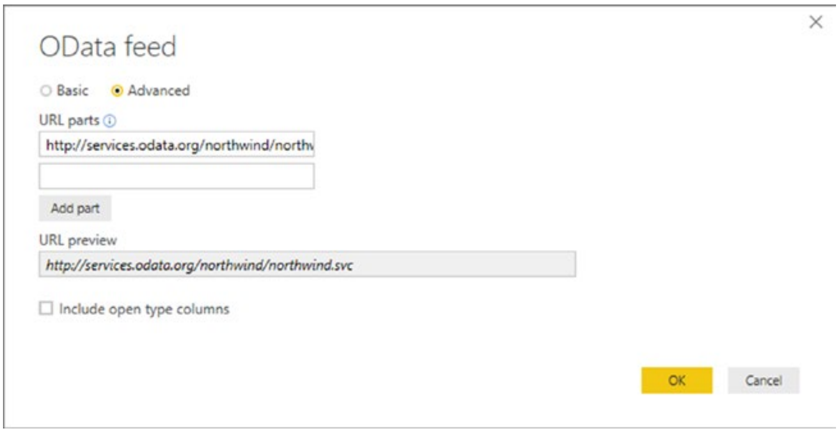
Current_Product_Lists

ProductID	ProductName
1	Chai
2	Chang
3	Aniseed Syrup
4	Chef Anton's Cajun Seasoning
6	Grandma's Boysenberry Spread
7	Uncle Bob's Organic Dried Pears
8	Northwoods Cranberry Sauce
10	Ikura
11	Queso Cabrales
12	Queso Manchego La Pastora
13	Konbu
14	Tofu
15	Genen Shouyu
16	Pavlova
18	Carnarvon Tigers
19	Teatime Chocolate Biscuits
20	Sir Rodney's Marmalade
21	Sir Rodney's Scones
22	Gustaf's Knäckebröd
23	Tunnbröd
25	NuNuCa Nuß-Nougat-Creme
26	Gumbär Gummibärchen
27	Schoggi Schokolade
30	Nord-Ost Matjeshering

Figure 5-27. *The Navigator dialog using an OData source*

OData Options

The OData Feed dialog (rather like the From Web dialog) also contains an Advanced button. Selecting this will expand the dialog to allow you to add one or more URL parts to the URL. You can see this in Figure 5-28.


OData feed

Basic [✓ Advanced]

URL parts

http://services.odata.org/northwind/northw

URL preview

http://services.odata.org/northwind/northwind.svc

☐ Include open type columns

Figure 5-28. *The OData Feed dialog Advanced options*

■ **Note** URL parts can be parameterized in the Power BI Desktop Query Editor. I will explain parameterization in Chapter 9.

Azure SQL Database

SQL Server does not only exist as an on-premises database. It is also available as a "Platform as a Service" (also known as PaaS). Simply put, this lets you apply a pay-as-you-go model to your database requirements where you can fire up a database server in the cloud in a few minutes and then scale it to suit your requirements, rather than buying hardware and software and having to maintain them.

Connecting to Microsoft's PaaS offering, called Azure SQL Database, is truly simple. If you have the details of a corporate Azure SQL Database, you can use this to connect to. If you do not, and nonetheless want to experiment with connecting Power BI Desktop to Azure SQL Database, you can always request a free trial account from Microsoft and set up an Azure SQL Database database in a few minutes. If this is the path that you are taking, then you can find instructions on how to do this (including loading the sample data that you will connect to later in this section) at the following URL: `https://docs.microsoft.com/en-gb/azure/sql-database/sql-database-get-started-portal`.

■ **Tip** When you are creating the Azure SQL database, be sure to define the source to be *Sample*. This will ensure that the MS sample data is loaded into your test database.

■ **Note** If you are setting up an Azure SQL database, make sure that you include firewall rules to allow connection from the computer where you are running Power BI Desktop to the Azure SQL database.

To connect from Power BI Desktop to an Azure SQL database:

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click More.
3. Click Azure in the list on the left, and then Azure SQL Database on the right. The Get Data dialog will look like the one in Figure 5-29.

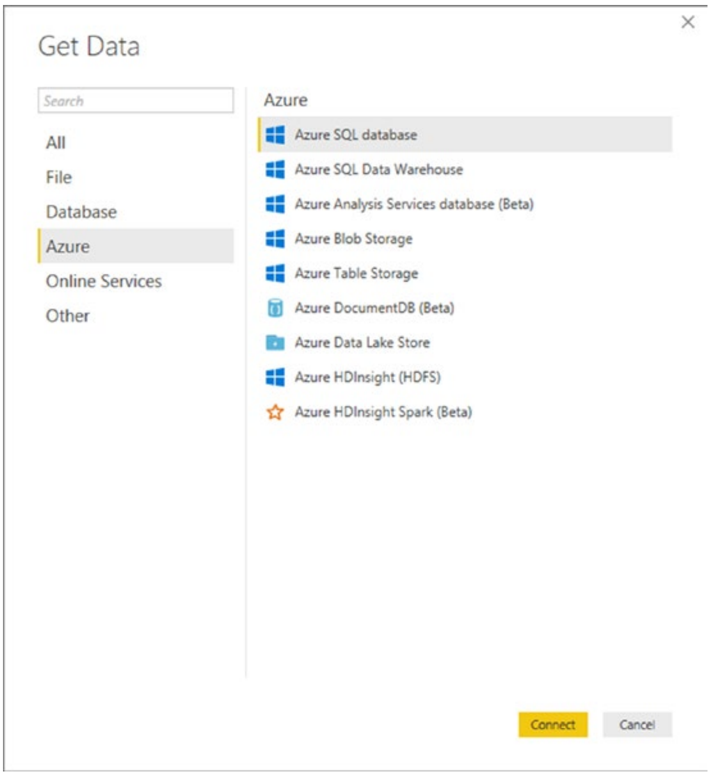
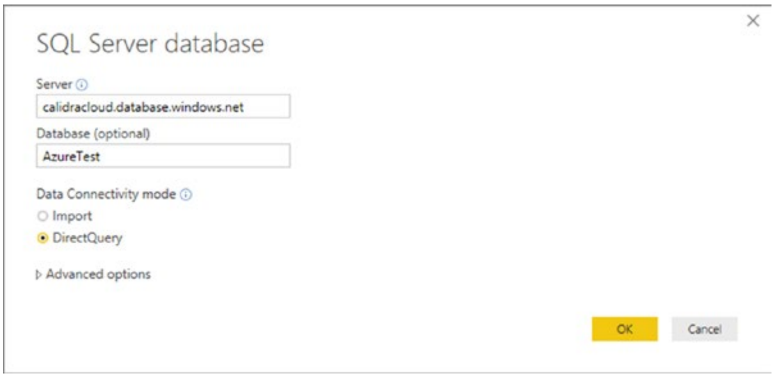


Figure 5-29. *Azure data sources in the Get Data dialog*

4. Click Connect. The SQL Server Database dialog will appear (after all, an Azure SQL database is a SQL Server database—but in the cloud).
5. Enter the Azure SQL Database server name that you obtained from the Microsoft Azure management portal (or that was given to you by a corporate DBA). The SQL Server Database dialog will look like the one shown in Figure 5-30.


SQL Server database

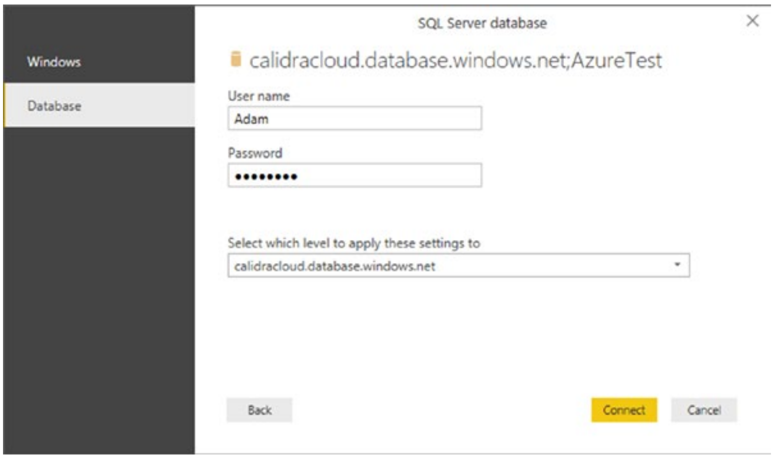
Server: calidracloud.database.windows.net

Database (optional): AzureTest

Data Connectivity mode: ○ Import [✓ ● DirectQuery]

Figure 5-30. *The SQL Server Database dialog for an Azure SQL database connection*

6. Click OK. The credentials dialog will appear.
7. Click Database on the left and enter a valid user name and password. The credentials dialog will look like the one shown in Figure 5-31.



calidracloud.database.windows.net;AzureTest

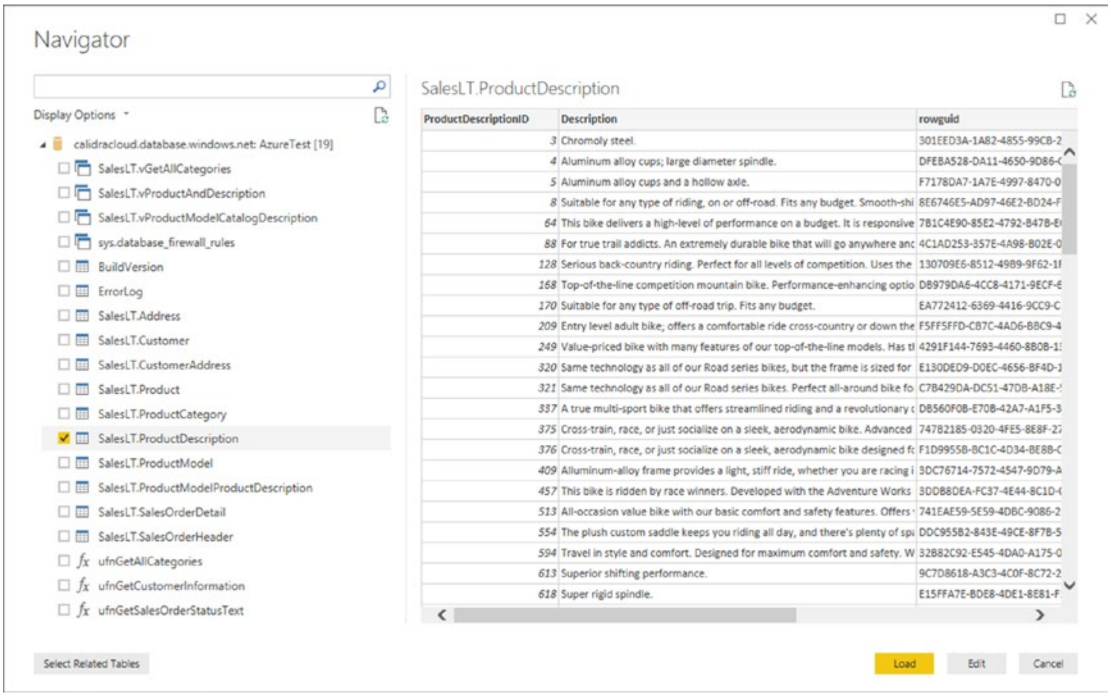
User name: Adam

Password: ••••••••

Select which level to apply these settings to: calidracloud.database.windows.net

***Figure 5-31.** The SQL Server credentials dialog for an Azure SQL database connection*

8. Click Connect. The Navigator dialog will appear showing the database(s) that you have permission to access in the Azure SQL Server database. This dialog will look like the one shown in Figure 5-32 if you are using the test data supplied by Microsoft for a default Azure SQL database.


Navigator

Display Options

calidracloud.database.windows.net: AzureTest [19]

- ☐ SalesLT.vGetAllCategories
- ☐ SalesLT.vProductAndDescription
- ☐ SalesLT.vProductModelCatalogDescription
- ☐ sys.database_firewall_rules
- ☐ BuildVersion
- ☐ ErrorLog
- ☐ SalesLT.Address
- ☐ SalesLT.Customer
- ☐ SalesLT.CustomerAddress
- ☐ SalesLT.Product
- ☐ SalesLT.ProductCategory
- ✓ SalesLT.ProductDescription
- ☐ SalesLT.ProductModel
- ☐ SalesLT.ProductModelProductDescription
- ☐ SalesLT.SalesOrderDetail
- ☐ SalesLT.SalesOrderHeader
- ☐ ufnGetAllCategories
- ☐ ufnGetCustomerInformation
- ☐ ufnGetSalesOrderStatusText

SalesLT.ProductDescription

ProductDescriptionID	Description	rowguid
3	Chromoly steel.	301EED3A-1A82-4855-99CB-2
4	Aluminum alloy cups; large diameter spindle.	DFEBA528-DA11-4650-9D86-(
5	Aluminum alloy cups and a hollow axle.	F7178DA7-1A7E-4997-8470-0
8	Suitable for any type of riding, on or off-road. Fits any budget. Smooth-shi	8E6746E5-AD97-46E2-BD24-F
64	This bike delivers a high-level of performance on a budget. It is responsive	7B1C4E90-85E2-4792-B47B-E
88	For true trail addicts. An extremely durable bike that will go anywhere anc	4C1AD253-357E-4A98-B02E-0
128	Serious back-country riding. Perfect for all levels of competition. Uses the	130709E6-8512-49B9-9F62-1I
168	Top-of-the-line competition mountain bike. Performance-enhancing optio	DB979DA6-4CC8-4171-9ECF-6
170	Suitable for any type of off-road trip. Fits any budget.	EA772412-6369-4416-9CC9-C
209	Entry level adult bike; offers a comfortable ride cross-country or down the	F5FF5FFD-CB7C-4AD6-BBC9-4
249	Value-priced bike with many features of our top-of-the-line models. Has tl	4291F144-7693-4460-8B0B-1:
320	Same technology as all of our Road series bikes, but the frame is sized for	E130DED9-D0EC-4656-BF4D-1
321	Same technology as all of our Road series bikes. Perfect all-around bike fo	C7B429DA-DC51-47DB-A18E-'
337	A true multi-sport bike that offers streamlined riding and a revolutionary c	DB560F0B-E70B-42A7-A1F5-3
375	Cross-train, race, or just socialize on a sleek, aerodynamic bike. Advanced	747B2185-0320-4FE5-8E8F-27
376	Cross-train, race, or just socialize on a sleek, aerodynamic bike designed fc	F1D9955B-BC1C-4D34-BE8B-(
409	Alluminum-alloy frame provides a light, stiff ride, whether you are racing i	3DC76714-7572-4547-9D79-A
457	This bike is ridden by race winners. Developed with the Adventure Works	3DDB8DEA-FC37-4E44-8C1D-(
513	All-occasion value bike with our basic comfort and safety features. Offers	741EAE59-5E59-4DBC-9086-2
554	The plush custom saddle keeps you riding all day, and there's plenty of spi	DDC955B2-843E-49CE-8F7B-5
594	Travel in style and comfort. Designed for maximum comfort and safety. W	32B82C92-E545-4DA0-A175-0
613	Superior shifting performance.	9C7D8618-A3C3-4C0F-8C72-2
618	Super rigid spindle.	E15FFA7E-BDE8-4DE1-8E81-F

Figure 5-32. *The Navigator dialog for an Azure SQL database connection showing sample data*

If you followed the steps to connect to an on-premises SQL Server database in Chapter 3, then you are probably feeling that the approach used here is virtually identical. Fortunately, the Power BI development team has worked hard to make the two processes as similar as possible. This extends to

- Ensuring that the DataSource settings are stored by Power BI Desktop and can be updated just as you can for an on-premises database connection
- Allowing you either to use DirectQuery or to import data into the Power BI in-memory data model
- Using the same Advanced options (writing your own SELECT queries or using stored procedures) that you can use with an on-premises SQL Server

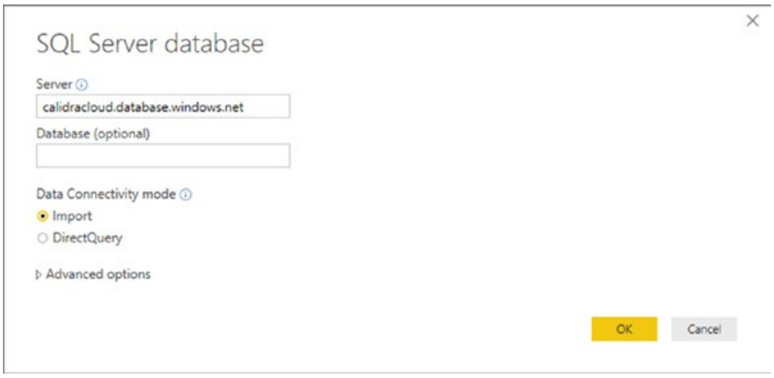
Azure SQL Data Warehouse

Azure has many available platforms to store data. One that is particularly well adapted to Power BI Desktop is the Azure SQL Data Warehouse. This is a tabular data warehouse that is hosted in the cloud.

Once again I will presume that, unless you have a corporate Azure SQL Data Warehouse at hand, you will be using a trial Azure account and that you have provisioned an Azure SQL Data Warehouse using the sample data that Microsoft provides. Setting up a test data warehouse is very similar to preparing a database, as described at the URL at the start of the previous section. Here, too, you need firewall rules to be set up correctly (although this may not be strictly necessary if you have previously set up firewall rules for, say, Azure SQL Database).

■ **Note** When you are creating the Azure SQL Data Warehouse, be sure to define the source to be *Sample*. This will ensure that the MS sample data is loaded into your test data warehouse and you will not be querying an empty data warehouse.

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click More.
3. Click Azure in the list on the left, and then Azure SQL Data Warehouse on the right.
4. Click Connect. The SQL Server Database dialog will appear.
5. Enter the Azure SQL Data Warehouse server name that you obtained from the Microsoft Azure management portal (or that was given to you by a corporate DBA). The SQL Server Database dialog will look like the one shown in Figure 5-33.


SQL Server database

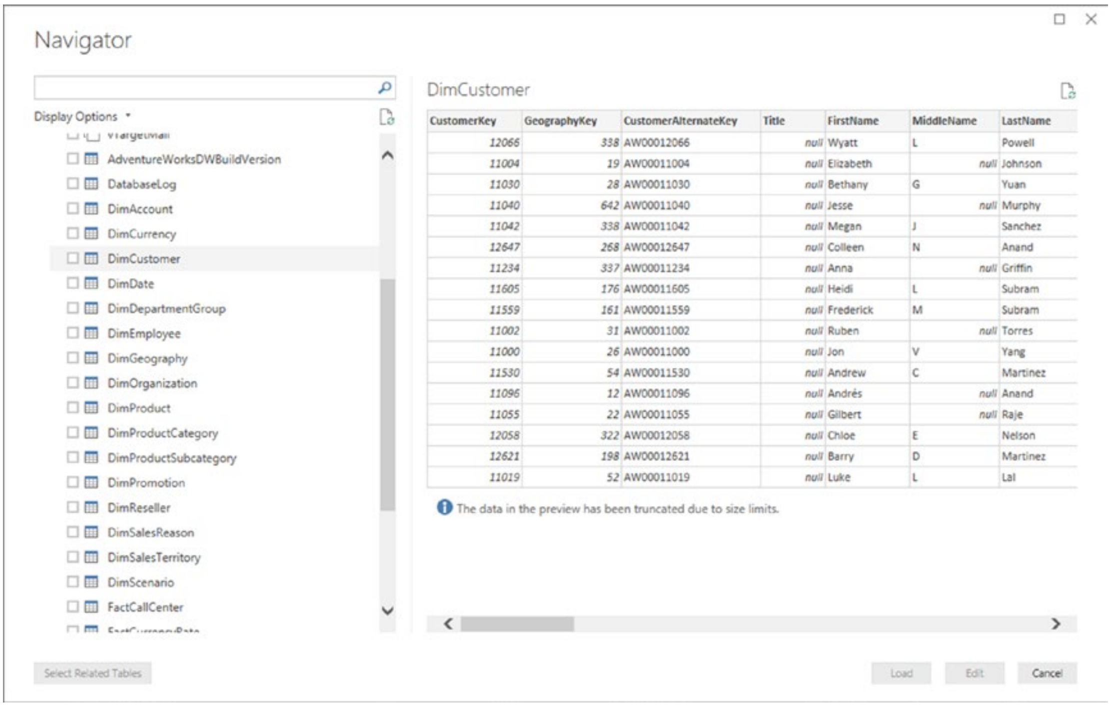
Server: calidracloud.database.windows.net

Database (optional):

Data Connectivity mode: [✓ Import] DirectQuery

Figure 5-33. *The SQL Server Database dialog for an Azure SQL Data Warehouse connection*

6. Select the Import button and then click OK. The credentials dialog will appear.
7. Click Database on the left and enter a valid user name and password.
8. Click Connect. The Navigator dialog will appear showing the database(s) that you have permission to access in the Azure SQL Server database. This dialog will look like the one shown in Figure 5-34 if you are using the test data supplied by Microsoft.



DimCustomer

CustomerKey	GeographyKey	CustomerAlternateKey	Title	FirstName	MiddleName	LastName
12066	338	AW00012066	null	Wyatt	L	Powell
11004	19	AW00011004	null	Elizabeth	null	Johnson
11030	28	AW00011030	null	Bethany	G	Yuan
11040	642	AW00011040	null	Jesse	null	Murphy
11042	338	AW00011042	null	Megan	J	Sanchez
12647	268	AW00012647	null	Colleen	N	Anand
11234	337	AW00011234	null	Anna	null	Griffin
11605	176	AW00011605	null	Heidi	L	Subram
11559	161	AW00011559	null	Frederick	M	Subram
11002	31	AW00011002	null	Ruben	null	Torres
11000	26	AW00011000	null	Jon	V	Yang
11530	54	AW00011530	null	Andrew	C	Martinez
11096	12	AW00011096	null	Andrés	null	Anand
11055	22	AW00011055	null	Gilbert	null	Raje
12058	322	AW00012058	null	Chloe	E	Nelson
12621	198	AW00012621	null	Barry	D	Martinez
11019	52	AW00011019	null	Luke	L	Lal

The data in the preview has been truncated due to size limits.

Figure 5-34. *The Navigator dialog for an Azure SQL Data Warehouse connection showing sample data*

9. Select the tables that you need and click Load or Edit to return to Power BI Desktop and begin adding visuals to your report.

■ **Note** Do not be phased by the fact that the title for the dialog where you specify the server and database says "SQL Server Database." This will connect you to the Azure Data Warehouse correctly.

As was the case for on on-premises connection, you can choose a Live Connection (even if the dialog calls it DirectQuery) and can expand the Advanced options field to enter a specific DAX query if you are loading data.

Connecting to SQL Server on an Azure Virtual Machine

More and more databases are now hosted outside a corporate environment by cloud services providers. With a provider such as Amazon (with RDS for SQL Server) or Microsoft (who offers virtual machines—or VMs—for SQL Server in Azure), you can now site your databases outside the enterprise and access them from virtually anywhere in the world.

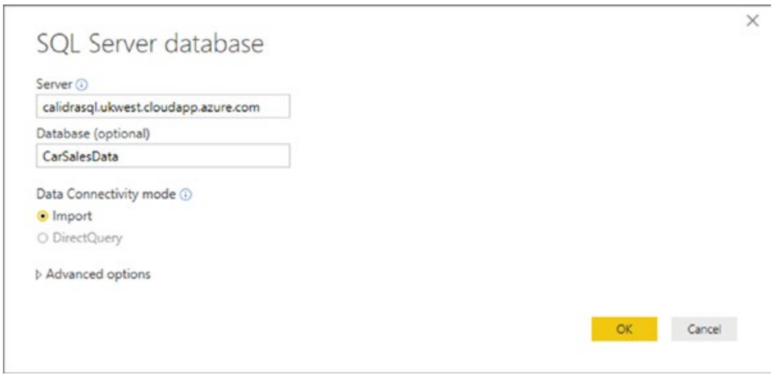
So, to extend the panoply of data sources available to Power BI Desktop, we will now see, briefly, how to connect to SQL Server on an Azure Virtual Machine . Admittedly, connecting to SQL Server on an Azure Virtual Machine is nearly the same as connecting to SQL Server in a corporate environment. However, it is worth a short detour to explain, briefly, how to return data to Power BI Desktop from a SQL Server instance in the cloud.

Once again, if you do not have a SQL Server instance that is hosted on an Azure Virtual Machine in your corporate environment, then you can always test this process using an Azure trial account. I cannot, however, explain here how to set up a SQL Server instance on a VM, as this is outside the scope of this book. There are, however, many resources available that can explain how to do this should you need them.

■ **Note** If you are creating your own virtual machine, then you can connect to this in SQL Server Management Studio using the same connection string that you use in step 5 below and create the sample database manually. You can then load the sample data as described in Appendix B. I will not explain this process in detail as I am presuming a basic level of familiarity with Windows Server and SQL Server if you are setting up your own VM.

To connect to SQL Server on a Virtual Machine:

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
3. Enter the full string that describes the server in the Server text box. Either this will be given to you by a corporate DBA or, if you are using your own Azure account, you can find it in the Azure Management Portal.
4. Enter the database name; if you have loaded the sample data that accompanies this book into a SQL Server instance in a VM, it will be CarSalesData. The dialog will look like Figure 5-35.


SQL Server database

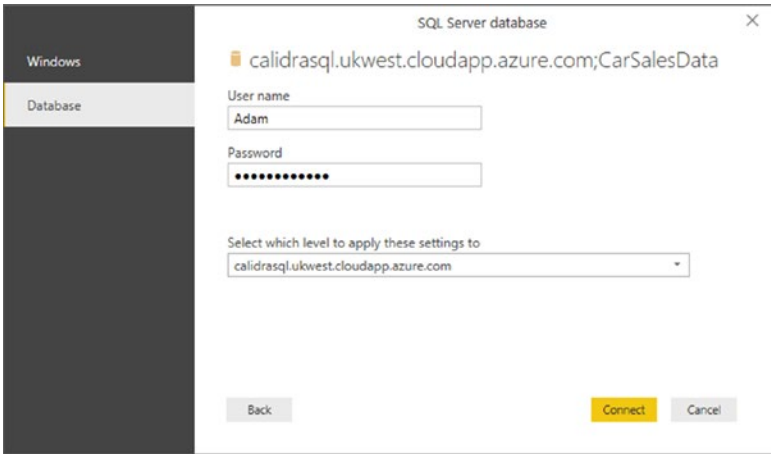
Server: calidrasql.ukwest.cloudapp.azure.com

Database (optional): CarSalesData

Data Connectivity mode: [✓ Import] DirectQuery

Figure 5-35. *The Microsoft SQL Server Database dialog*

5. Click OK. The Access a SQL Server Database dialog will appear. Select Database as the security mode and enter the user name and password, as shown in Figure 5-36. If you are using your own Azure account, these can be the user name and password that you specified when setting up the virtual machine.



calidrasql.ukwest.cloudapp.azure.com;CarSalesData

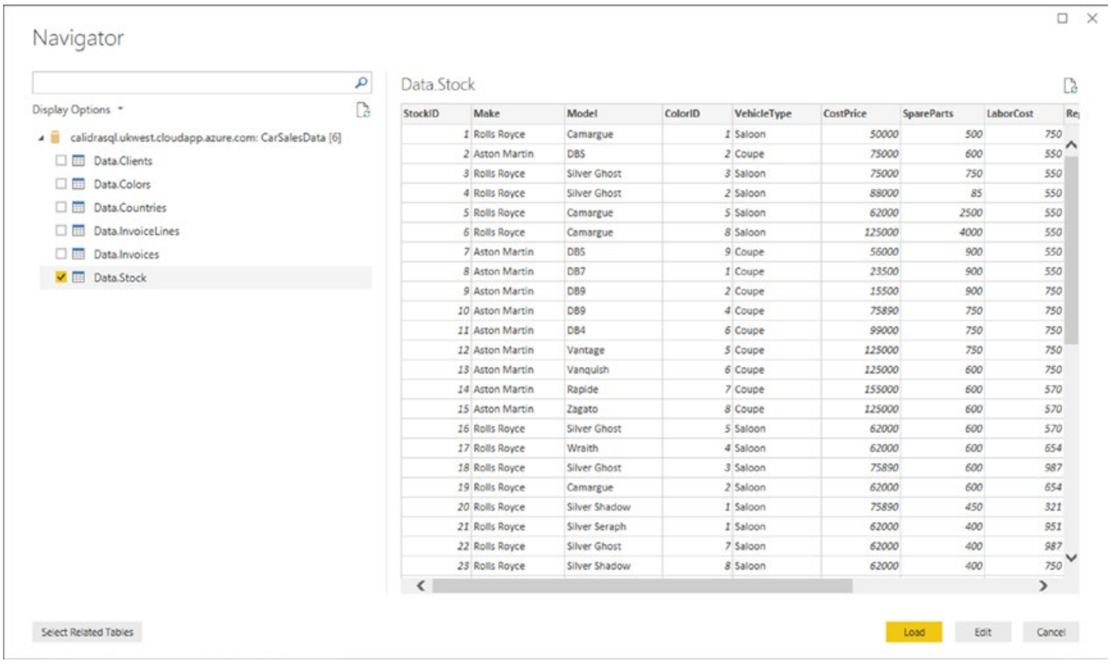
User name: Adam

Password: ••••••••••••

Select which level to apply these settings to: calidrasql.ukwest.cloudapp.azure.com

Figure 5-36. *The SQL Server Database dialog when connecting to a virtual machine*

6. If you see the encryption support dialog, click OK. The Navigator dialog will appear as shown in Figure 5-37, listing all the tables that you have permissions to see on the SQL Server hosted by the virtual machine.



calidrasql.ukwest.cloudapp.azure.com: CarSalesData [6]

- ☐ Data.Clients
- ☐ Data.Colors
- ☐ Data.Countries
- ☐ Data.InvoiceLines
- ☐ Data.Invoices
- ✓ Data.Stock

Data.Stock

StockID	Make	Model	ColorID	VehicleType	CostPrice	SpareParts	LaborCost	Re
1	Rolls Royce	Camargue	1	Saloon	50000	500	750	
2	Aston Martin	DBS	2	Coupe	75000	600	550	
3	Rolls Royce	Silver Ghost	3	Saloon	75000	750	550	
4	Rolls Royce	Silver Ghost	2	Saloon	88000	85	550	
5	Rolls Royce	Camargue	5	Saloon	62000	2500	550	
6	Rolls Royce	Camargue	8	Saloon	125000	4000	550	
7	Aston Martin	DBS	9	Coupe	56000	900	550	
8	Aston Martin	DB7	1	Coupe	23500	900	550	
9	Aston Martin	DB9	2	Coupe	15500	900	750	
10	Aston Martin	DB9	4	Coupe	75890	750	750	
11	Aston Martin	DB4	6	Coupe	99000	750	750	
12	Aston Martin	Vantage	5	Coupe	125000	750	750	
13	Aston Martin	Vanquish	6	Coupe	125000	600	750	
14	Aston Martin	Rapide	7	Coupe	155000	600	570	
15	Aston Martin	Zagato	8	Coupe	125000	600	570	
16	Rolls Royce	Silver Ghost	5	Saloon	62000	600	570	
17	Rolls Royce	Wraith	4	Saloon	62000	600	654	
18	Rolls Royce	Silver Ghost	3	Saloon	75890	600	987	
19	Rolls Royce	Camargue	2	Saloon	62000	600	654	
20	Rolls Royce	Silver Shadow	1	Saloon	75890	450	321	
21	Rolls Royce	Silver Seraph	1	Saloon	62000	400	951	
22	Rolls Royce	Silver Ghost	7	Saloon	62000	400	987	
23	Rolls Royce	Silver Shadow	8	Saloon	62000	400	750	

Figure 5-37. *The Navigator dialog when connecting to a virtual machine*

As you can see, the process is virtually identical to the one that you followed to connect to SQL Server in Chapter 2. I have, nonetheless, a few points that I need to bring to your attention:

- You use the Azure VM multipart name as the server name.
- As was the case when connecting to an on-premises SQL Server instance, you can select the database if required.
- You can use the server's IP address as the database name if the VM has specified a public IP address.
- Security is a big and separate question. In a corporate environment, you *might* be able to use Windows security to connect. You will almost certainly have to use database security for a test VM.
- As is always the case in Azure, firewalls must be set up correctly.
- DirectQuery is not available when connecting to SQL Server on an Azure Virtual Machine.

Azure Blob Storage

The final Azure data source that I want to introduce you to in this chapter is Azure Blob Storage. To all intents and purposes you can consider this, as far as Power BI Desktop is concerned, as a file share in the cloud. So if you need to access data that is stored as files, you can connect to them via Azure Blob Storage.

Once again, you will need either corporate access to Azure Blob Storage or an Azure trial account. In either case you need to copy the two sample files that are in the folder C:\PowerBiDesktopSamples\CH05 into a container in your Azure Blob Storage. Downloading the sample files is explained in Appendix A.

Once the source data is available in Azure Blob Storage, you can carry out the following steps:

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click More.
3. Click Azure in the list on the left, and then Azure Blob Storage on the right. The Azure Blob Storage connection dialog will be displayed.
4. Enter the account name that you are using to connect to Azure Blob Storage. The Azure Blob Storage dialog will look like the one shown in Figure 5-38. If you are using a corporate Azure Blob Storage account, then your system administrator will provide this. In a test scenario you can find this in the Azure Management Portal by opening the Storage Account blade and copying the Blob Service Endpoint.

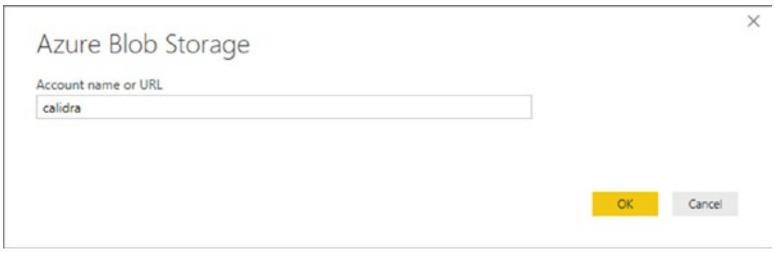

Azure Blob Storage
Account name or URL
calidra

Figure 5-38. *The Azure Blob Storage connection dialog*

5. Click OK. The Azure Blob Storage Account Key dialog will appear.

6. In the Azure Management Portal, copy an account key. These can be found in the Azure Management Portal by clicking the Storage Account blade and then clicking Access Keys. If you have been sent an account key by a system administrator, then use that instead.

7. Paste the account key into the Azure Blob Storage Account Key dialog. The dialog will look like the one in Figure 5-39.

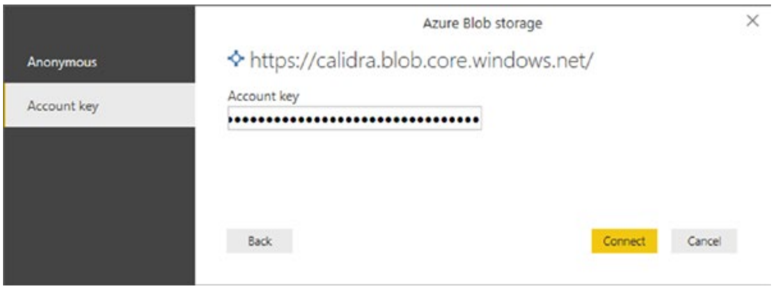
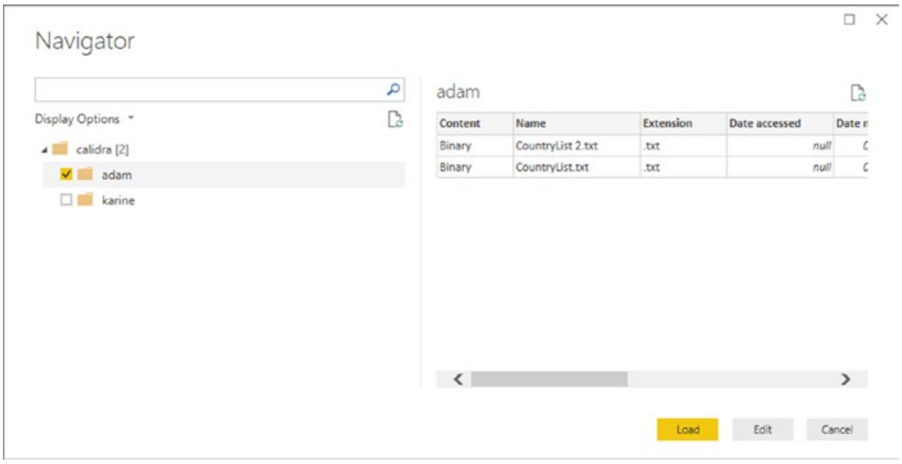


Figure 5-39. *The Azure Blob Storage Account Key dialog*

8. Click Connect. The Navigator will appear, showing the list of files in the selected container. You can see an example of this in Figure 5-40.



Content	Name	Extension	Date accessed	Date n
Binary	CountryList 2.txt	.txt	null	C
Binary	CountryList.txt	.txt	null	C

Figure 5-40. *The Navigator dialog showing available containers in Azure Blob Storage*

9. Click Load. The list of files stored in Azure will appear in Power BI Desktop.

■ **Note** It is important to note that, for the moment at least, what you have returned from Azure is a list of available files. Chapter 8 explains how to select and load data from some or all of the available files into Power BI Desktop, where they can be used as a basis for analytics.

Azure Security

All cloud service providers take security extremely seriously. As you have seen in this chapter, you will always be obliged to enter some form of security token and/or specify a valid user name and password to connect to cloud-based data.

All the security information that you entered is stored in the Power BI Desktop Storage Settings. This can be removed or modified in the same way that you learned to update or remove database security information in Chapter 3.

Conclusion

In this chapter you saw, briefly, how to retrieve data that you access using the Internet. This can range from a table of data on a web page to a massive Azure SQL data warehouse. Alternatively, perhaps you need to create dashboards based in your Salesforce, Google Analytics, or MS Dynamics 365 data. Maybe your organization has decided to move its datacenters to the cloud, and is using SQL Server in Azure or Amazon Redshift. In any case, Power BI Desktop can connect and access the data available in these services and repositories. It can even access big data in Hadoop.

Given the vast number of online sources, this chapter could only scratch the surface of this huge range of potential data repositories. However, as Power BI Desktop is rigorous about standardizing access to data, you should be able to apply the approaches you have learned in this chapter to many other data services, both current and future.

This chapter concludes the set of four chapters that introduced you to some of the many and varied data sources that you can use with Power BI Desktop. In the course of four chapters, you have seen how to load data from a selection of the more frequently used available sources. The good news is that Power BI Desktop can read data from dozens more sources. The bad news is that it would take a whole book to go into all of them in detail.

So I will not be describing any other data sources in this book. This is because now that you have come to appreciate the core techniques that make up the extremely standardized approach that Power BI Desktop takes to loading data, you can probably load any possible data type without needing much more information from me. Should you need any specific information on other data sources, then your best port of call is the Microsoft Power BI web site. This is currently at `http://support.powerbi.com`.

Now that you can find, access, and load the data you need into Power BI Desktop, it is time to move on to the next step. This means cleansing and restructuring the datasets so that they suit your analytical requirements. Handling these challenges is the subject of the next three chapters.

CHAPTER 6

Dealing with Datasets

In the previous four chapters you saw some of the ways in which you can find and load data into the Power BI Desktop data model. Inevitably, this is the first part of any process that you follow to extract, transform, and load data. Yet it is quite definitely only a first step. Once the data is in Power BI Desktop, you need to know how to adapt it to suit your requirements in a multitude of ways. This is because not all data is ready to be used immediately. Quite often, you have to do some initial work on the data to make it more easily useable in Power BI Desktop. Tweaking source data is generally referred to as *data transformation*, which is the subject of this chapter as well as the next two.

The range of transformations that Power BI Desktop offers is extensive and varied. Learning to apply the techniques that Power BI Desktop makes available enables you to take data as you find it, then cleanse it and push it back into the Power BI Desktop data model as a series of coherent and structured data tables. Only then is it ready to be used to create compelling dashboards and reports.

As it is all too easy to be overwhelmed (at least initially) by the extent of the data transformation options that Power Bi Desktop has to offer, I have grouped the possible modifications into four categories. These categories are my own, and are merely a suggestion to facilitate understanding:

- *Data transformation*: This includes adding and removing columns and rows, renaming columns, as well as filtering data.
- *Data modification*: This covers altering the actual data in the rows and columns of a dataset.
- *Extending datasets*: This encompasses adding further columns, possibly expanding existing columns into more columns or rows, and adding calculations.
- *Joining datasets*: This involves combining multiple separate datasets—possibly from different data sources—into a single dataset.

This chapter introduces you to the core techniques that you can apply to shape each individual dataset that you have loaded. These transformations include

- Renaming, removing, and reordering columns
- Removing groups or sets of rows
- Deduplicating datasets
- Sorting the data
- Excluding records by filtering the data
- Grouping records

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_6

In the next chapter, you learn how to cleanse and modify data. In Chapter 8, you see how to subset columns to extract part of the available data in a column, calculate columns, merge data from separate queries, and add further columns containing different types of calculations, and you learn about pivoting and unpivoting data. So, if you cannot find what you are looking for in this chapter, there is a good chance that the answer is in the following two chapters.

In this chapter, I will also use a set of example files that you can find on the Apress web site. If you have followed the instructions in Appendix A, then these files are in the C:\PowerBiDesktopSamples\CH06 folder.

Power BI Desktop Queries

In Chapter 1, you saw how to load source data directly into Power BI Desktop and use it immediately to create dashboards. Clearly, this approach presumes that the data that you are using is perfectly structured, clean, and error-free. Source data is nearly always correct and ready to use in reports and dashboards when it comes from "corporate" data sources such as data warehouses (held in relational, dimensional, or tabular databases). This is not always the case when you are faced with multiple disparate sources of data that have not been precleansed and prepared by an IT department. The everyday reality is that you could have to cleanse and transform much of the source data that you will use for your Power BI Desktop dashboards.

The really good news is that the kind of data transformation that used to require expensive servers and industrial-strength software is now available for free. Yes, Power BI Desktop comes with an awesome ETL (Extract, Transform, and Load) tool that can rival many applications that cost hundreds of thousands of dollars.

Power BI Desktop data transformation is carried out using *queries*. As you saw in Chapter 1, you do not have to modify source data. You can load it directly if it is ready for use. Yet if you need to cleanse the data, you add an intermediate step between connecting the data and loading it into the Power BI Desktop data model. This intermediate step uses the Power BI Desktop Query Editor to tweak the source data.

So how do you apply queries to transform your data? You have two choices:

- Load the data first from one or more sources, and then transform it later.
- Edit each source data element in a query before loading it.

Power BI Desktop is extremely forgiving. It does not force you to select one or the other method and then lock you in to the consequences of your decision. You can load data first and then realize that it needs some adjustment, switch to the Query Editor and make changes, and then return to creating your dashboard. Or you can first focus on the data and try to get it as polished and perfect as possible before you start building reports. The choice is entirely up to you.

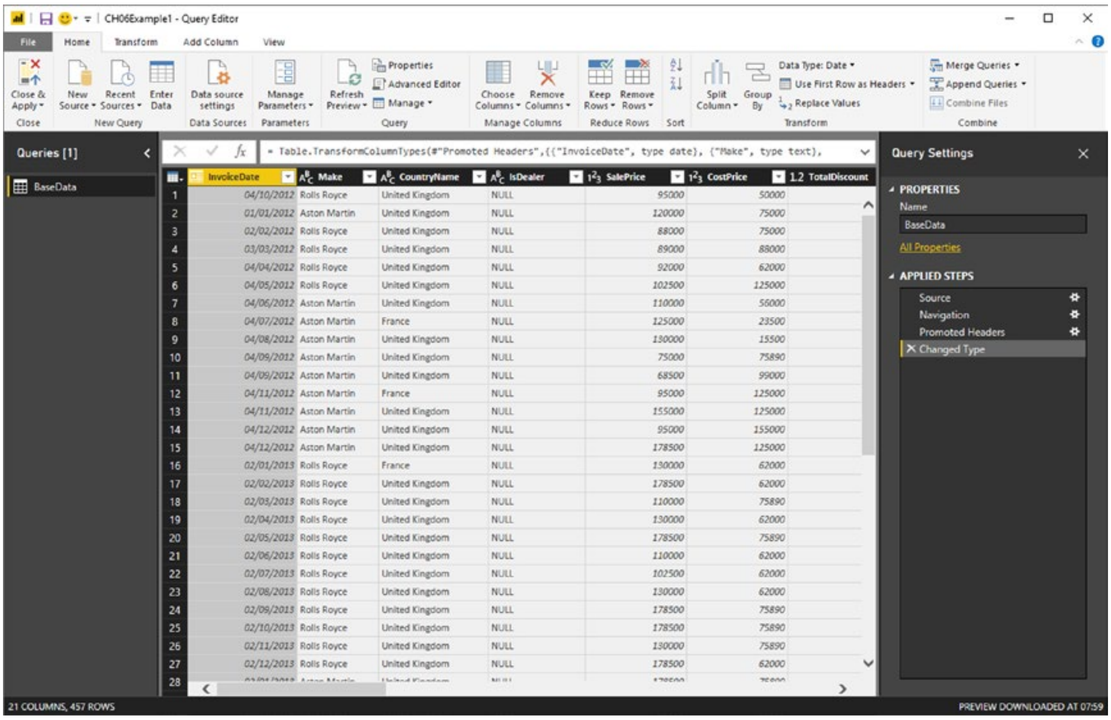
To make this point, let's take a look at both of these ways of working.

■ **Note** At risk of being pedantic and old-fashioned, I would advise you to make notes when creating really complex transformations, because going back to a solution and trying to make adjustments later can be painful when they are not documented at all.

Editing Data After a Data Load

In Chapter 1, you saw how to load the Excel workbook CarSales.xlsx directly into the Power BI Desktop data model and use it to create a starter dashboard. Now let's presume that you want to make some changes to the data structure of the data that you have already loaded. Specifically, you want to rename the CostPrice column. The file that you want to modify is CH06Example1.pbix file in the C:\PowerBiDesktopSamples\CH06 directory.

1. Launch Power BI Desktop.
2. Open the sample file C:\PowerBiDesktopSamples\CH06\CH06Example1.pbix. Take a look at the Fields list and note that there is a field named CostPrice.
3. In the Power BI Desktop Home ribbon, click the Edit Queries button. The Power BI Desktop Query Editor will open and display the source data as a table. The window will look like Figure 6-1.



```
= Table.TransformColumnTypes(#"Promoted Headers",{{"InvoiceDate", type date}, {"Make", type text},
```

	InvoiceDate	Make	CountryName	IsDealer	SalePrice	CostPrice	TotalDiscount
1	04/10/2012	Rolls Royce	United Kingdom	NULL	95000	50000	
2	01/01/2012	Aston Martin	United Kingdom	NULL	120000	75000	
3	02/02/2012	Rolls Royce	United Kingdom	NULL	88000	75000	
4	03/03/2012	Rolls Royce	United Kingdom	NULL	89000	88000	
5	04/04/2012	Rolls Royce	United Kingdom	NULL	92000	62000	
6	04/05/2012	Rolls Royce	United Kingdom	NULL	102500	125000	
7	04/06/2012	Aston Martin	United Kingdom	NULL	110000	56000	
8	04/07/2012	Aston Martin	France	NULL	125000	23500	
9	04/08/2012	Aston Martin	United Kingdom	NULL	130000	15500	
10	04/09/2012	Aston Martin	United Kingdom	NULL	75000	75890	
11	04/09/2012	Aston Martin	United Kingdom	NULL	68500	99000	
12	04/11/2012	Aston Martin	France	NULL	95000	125000	
13	04/11/2012	Aston Martin	United Kingdom	NULL	155000	125000	
14	04/12/2012	Aston Martin	United Kingdom	NULL	95000	155000	
15	04/12/2012	Aston Martin	United Kingdom	NULL	178500	125000	
16	02/01/2013	Rolls Royce	France	NULL	130000	62000	
17	02/02/2013	Rolls Royce	United Kingdom	NULL	178500	62000	
18	02/03/2013	Rolls Royce	United Kingdom	NULL	110000	75890	
19	02/04/2013	Rolls Royce	United Kingdom	NULL	130000	62000	
20	02/05/2013	Rolls Royce	United Kingdom	NULL	178500	75890	
21	02/06/2013	Rolls Royce	United Kingdom	NULL	110000	62000	
22	02/07/2013	Rolls Royce	United Kingdom	NULL	102500	62000	
23	02/08/2013	Rolls Royce	United Kingdom	NULL	130000	62000	
24	02/09/2013	Rolls Royce	United Kingdom	NULL	178500	75890	
25	02/10/2013	Rolls Royce	United Kingdom	NULL	178500	75890	
26	02/11/2013	Rolls Royce	United Kingdom	NULL	130000	75890	
27	02/12/2013	Rolls Royce	United Kingdom	NULL	178500	62000	
28	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	

Figure 6-1. *The Power BI Desktop Query Editor*

4. Right-click the title of the CostPrice column. The column will be selected and the title will appear in yellow.
5. Select Rename from the context menu.
6. Type **VehicleCost** and press Enter. The column title will change to VehicleCost.
7. In the Power BI Desktop Query Editor Home ribbon, click the Close & Apply button. The Power BI Desktop Query Editor will close and return you to the Power BI Desktop window. VehicleCost has replaced CostPrice anywhere that it was used in the dashboard. This is immediately visible in the Fields list.

I hope that this simple example makes it clear that transforming the source data is a quick and painless process. The technique that you applied—renaming a column—is only one of many dozens of possible techniques that you can apply to transform your data. However, it is not the specific transformation that is the core idea to take away here. What you need to remember is that the data that underpins your dashboard is always present and it is only a single click away. At any time, you can "flip" to the data and make changes, simply by clicking the Edit Queries button in the Power BI Desktop window. Any changes that you make and confirm will update your dashboards and reports instantaneously.

■ **Note** Alternatively, if you want you can edit the query behind any table that is visible in the Report or Data views simply by right-clicking in the table and selecting Edit Query from the context menu.

Transforming Data Before Loading

On some occasions, you might prefer to juggle with your data before you load it. This is a variation on the approach that you have used in Chapter 1 when creating a simple dashboard. Do the following to transform your data *before* it appears in the Power BI Desktop window:

1. Open a new Power BI Desktop window.
2. In the Power BI Desktop Home ribbon, click the tiny triangle in the Get Data button.
3. Select Excel in the menu and open the Excel file C:\PowerBiDesktopSamples\CH06\CarSales.xlsx.
4. In the Navigator window, select the BaseData worksheet.
5. Click the Edit button (*not* the Load button).
6. The Power BI Desktop Query Editor will open and display the source data as a table.
7. Carry out steps 4 through 6 from the previous example to rename the CostPrice column.
8. In the Power BI Desktop Query Editor Home ribbon, click the Close & Apply button. The Power BI Desktop Query Editor will close and return you to the Power BI Desktop window. You will see the Apply Query Changes dialog while the data is loaded, like the one that you can see in Figure 6-2.

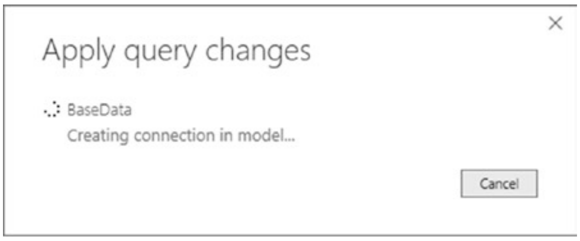


Figure 6-2. *The Apply Query Changes dialog*

This time, you have made a simple modification to the data *before* loading the dataset into the Power BI Desktop data model. The data modification technique was exactly the same. The only difference between loading the data directly and taking a detour via the Query window was clicking Edit instead of Load in the Navigator dialog. This means that the data was only loaded once you had finished making any modifications to the source data in the Power BI Desktop Query Editor.

Query or Load?

Power BI Desktop always gives you the choice of loading data directly into its data model or taking a constructive detour via Power BI Desktop Query. The path that you follow is entirely up to you and clearly depends on each set of circumstances. Nonetheless, you might want to consider the following basic principles when faced with a new dashboarding challenge using unfamiliar data:

- Are you convinced that the data is ready to use? That is, is it clean and well structured? If so, then you can try loading it directly into the Power BI Desktop data model.
- Are you faced with multiple data sources that need to be combined and molded into a coherent structure? If this is the case, then you really need to transform the data using Power BI Desktop Query.
- Does the data come from an enterprise data warehouse? This could be held in a relational database, a SQL Server Analysis Services cube, or even an in-memory tabular data warehouse. As these data sources are nearly always the result of many hundreds—or even thousands—of hours of work cleansing, preparing, and structuring the data, you can probably load these straight into the data model.
- Does the data need to be preaggregated and filtered? Think Power BI Desktop Query.
- Are you likely to need to change the field names to make the data more manageable? It could be simpler to load the data directly into the data model and change them there.
- Are you faced with lots of lookup tables that need to be added to a "core" data table? Then Power BI Desktop Query is your friend.
- Does the data contain many superfluous or erroneous elements? Then use Power BI Desktop Query to remove these as a first step.
- Does the data need to be rationalized and standardized to make it easier to handle? In this case, the path to success is via Power BI Desktop Query.
- Is the data source enormous? If this is the case, you could save time by editing the data first in the Query Editor. This is because the Query Editor only loads a *sample* of the data for you to tweak. The entire dataset will only be loaded when you confirm all your modifications and close the Query Editor.

These kinds of questions are only rough guidelines. Yet they can help to point you in the right direction when you are working with Power BI Desktop. Inevitably, the more that you work with this application, the more you will develop the reflexes and intuition that will help you make the correct decisions. Remember, however, that Power BI Desktop is there to help, and that even a directly loaded dataset is based on a query. So you can always load data and then decide to tweak the query structure later if you need to. Alternatively, editing data in a Query window can be a great opportunity to take a closer look at your data before loading it into the data model—and it only adds a couple of clicks.

So feel free to adopt a way of working that you feel happy with. Power BI Desktop will adapt to your style easily and almost invisibly, letting you switch from data to dashboards so fluidly that it will likely become second nature.

The remainder of this chapter will take you through some of the core techniques that you need to know to cleanse and shape your data. However, before getting into all the detail, let's take a quick, high-level look at the Power BI Desktop Query Editor and the way that it is laid out.

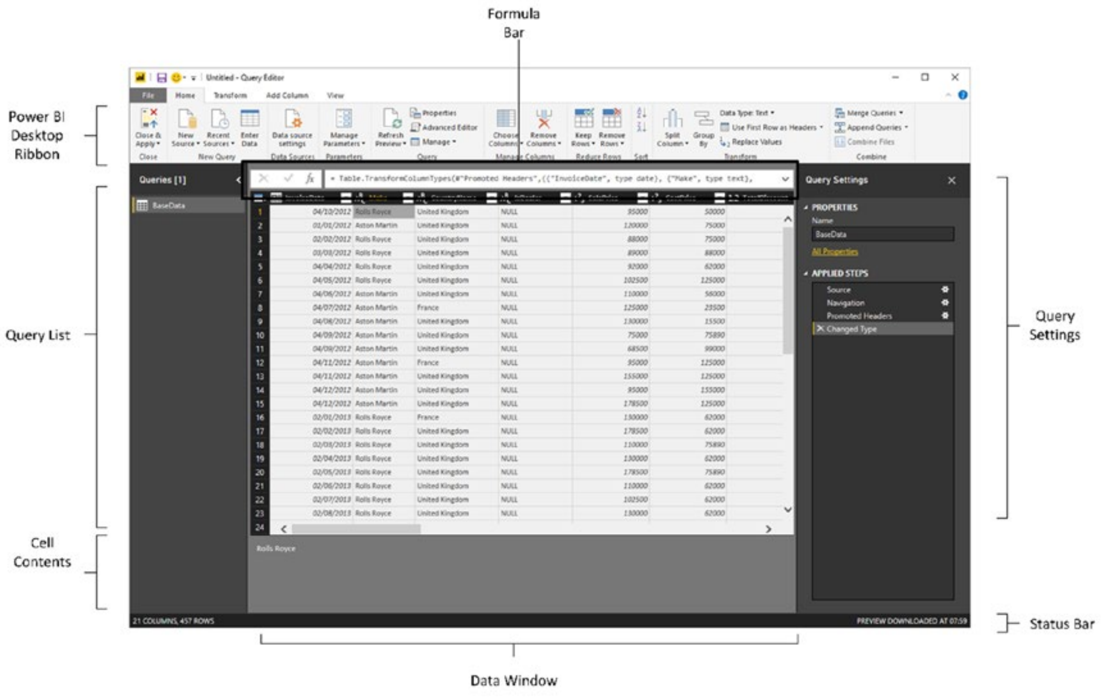
The Power BI Desktop Query Editor

All of your data transformation will take place in the Power BI Desktop Query Editor. It is a separate window from the one where you create your dashboards and it has a slightly different layout.

The Power BI Desktop Query Editor consists of six main elements:

- The four ribbons: Home, Transform, Add Column, and View
- The Query list pane containing all the queries that have been added to a Power BI Desktop file
- The Data window, where you can see a sample of the data for a selected query
- The Query Settings pane that contains the list of steps used to transform data
- The formula bar above the data that shows the code (written in the Power BI "M" language) that performs the selected transformation step
- The status bar (at the bottom of the window) that indicates useful information, such as the number of rows and columns in a query table, and the date when the dataset was downloaded

The callouts for these elements are shown in Figure 6-3.



```
= Table.TransformColumnTypes(#"Promoted Headers",{{"InvoiceDate", type date}, {"Make", type text},
```

1	04/10/2012	Rolls Royce	United Kingdom	NULL	95000	50000
2	01/01/2012	Aston Martin	United Kingdom	NULL	120000	75000
3	02/02/2012	Rolls Royce	United Kingdom	NULL	88000	75000
4	03/03/2012	Rolls Royce	United Kingdom	NULL	89000	88000
5	04/04/2012	Rolls Royce	United Kingdom	NULL	92000	62000
6	04/05/2012	Rolls Royce	United Kingdom	NULL	102500	125000
7	04/06/2012	Aston Martin	United Kingdom	NULL	110000	56000
8	04/07/2012	Aston Martin	France	NULL	125000	23500
9	04/08/2012	Aston Martin	United Kingdom	NULL	130000	15500
10	04/09/2012	Aston Martin	United Kingdom	NULL	75000	75890
11	04/09/2012	Aston Martin	United Kingdom	NULL	68500	99000
12	04/11/2012	Aston Martin	France	NULL	95000	125000
13	04/11/2012	Aston Martin	United Kingdom	NULL	155000	125000
14	04/12/2012	Aston Martin	United Kingdom	NULL	95000	155000
15	04/12/2012	Aston Martin	United Kingdom	NULL	178500	125000
16	02/01/2013	Rolls Royce	France	NULL	130000	62000
17	02/02/2013	Rolls Royce	United Kingdom	NULL	178500	62000
18	02/03/2013	Rolls Royce	United Kingdom	NULL	110000	75890
19	02/04/2013	Rolls Royce	United Kingdom	NULL	130000	62000
20	02/05/2013	Rolls Royce	United Kingdom	NULL	178500	75890
21	02/06/2013	Rolls Royce	United Kingdom	NULL	110000	62000
22	02/07/2013	Rolls Royce	United Kingdom	NULL	102500	62000
23	02/08/2013	Rolls Royce	United Kingdom	NULL	130000	62000
24						

Rolls Royce

21 COLUMNS, 457 ROWS

PREVIEW DOWNLOADED AT 07:59

Figure 6-3. *The Power BI Desktop Query Editor, explained*

■ **Note** Occasionally you will see other ribbons appear in specific circumstances. You can see an example of this for dealing with lists in Chapters 8 and 9.

The Applied Steps List

Data transformation is by its very nature a sequential process. So the Query window stores each modification that you make when you are cleansing and shaping source data. The various elements that make up a data transformation process are listed in the Applied Steps list of the Query Settings pane in the Query Editor.

The Power BI Desktop Query Editor does not number the steps in a data transformation process, but it certainly remembers each one. They start at the top of the Applied Steps list (nearly always with the Source step) and can extend to dozens of individual steps that trace the evolution of your data until you load it into the data model. You can, if you want, consider the Query Editor as a kind of "macro recorder."

Moreover, as you click each step in the Applied Steps list, the data in the Data window changes to reflect the results of each transformation, giving you a complete and visible trail of all the modifications that you have applied to the dataset.

The Applied Steps list gives a distinct name to the step for each and every data modification option that you cover in this chapter and the next. As it can be important to understand exactly what each function actually achieves, I will always draw to your attention the standard name that Power BI Desktop Query applies.

The Power BI Desktop Query Editor Ribbons

Power BI Desktop Query Editor uses (in the August 2017 version, at least) four ribbons. They are fundamental to what you learn in the course of this chapter. They are as follows:

- The Home ribbon
- The Transform ribbon
- The Add Column ribbon
- The View ribbon

I am not suggesting for a second that you need to memorize what all the buttons in these ribbons do. What I hope is that you are able to use the following brief descriptions of the Query Editor ribbon buttons to get an idea of the amazing power of Power BI Desktop in the field of data transformation. So if you have an initial dataset that is not quite as you need it, you can take a look at the resources that Power BI Desktop has to offer and how they can help. Once you find the function that does what you are looking for, you can jump to the relevant section for the full details on how to apply it.

The Home Ribbon

Since we will be making intense use of the Power BI Desktop Query Editor Home ribbon to transform data, it is important to have an idea of what it can do. I explain the various options in Figure 6-4 and in Table 6-1.

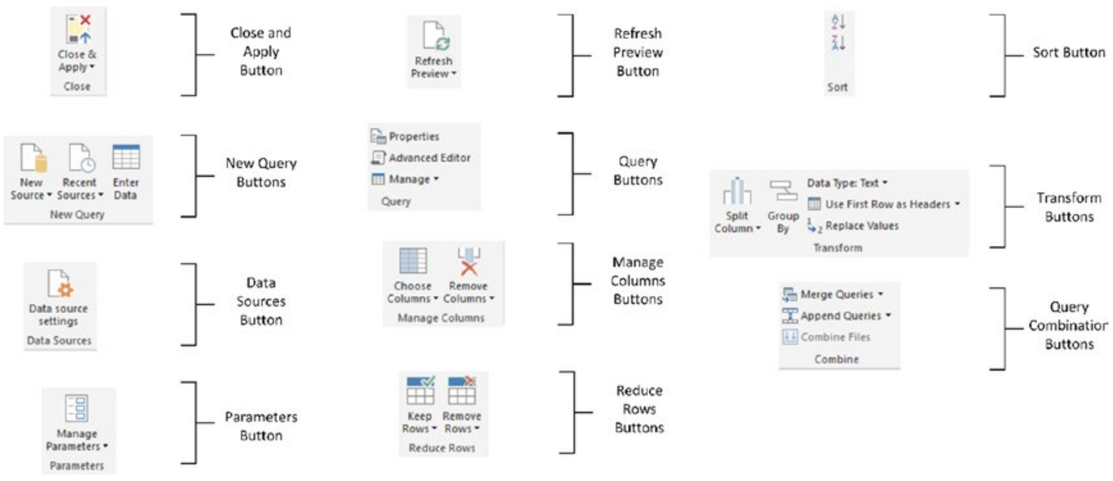


Figure 6-4. *The Query Editor Home ribbon*

Table 6-1. *Query Editor Home Ribbon Options*

Option	Description
Close & Apply	Finishes the processing steps; saves and closes the query.
New Source	Lets you discover and add a new data source to the set of queries.
Recent Sources	Lists all the recent data sources that you have used.
Enter Data	Lets you add your own specific data in a custom table.
Data Source Settings	Allows you to manage settings for data sources that you have already connected to.
Manage Parameters	Lets you view and modify any parameters defined for this Power BI Desktop file.
Refresh Preview	Refreshes the preview data.
Properties	Displays the core query properties.
Advanced Editor	Displays the “M” language editor.
Manage	Lets you delete, duplicate, or reference a query.
Choose Columns	Lets you select the columns to retain from all the columns available in the source data.
Remove Columns	Lets you remove one or more columns.
Keep Rows	Keeps the specified number of rows at the top of the table.
Remove Rows	Removes a specified number of rows from the top of the data table.
Sort	Sorts the table using the selected column as the sort key.
Split Column	Splits a column into one or many columns at a specified delimiter or after a specified number of characters.

(*continued*)

Table 6-1. *(continued)*

Option	Description
Group By	Groups the table using a specified set of columns and aggregates any numeric columns for this grouping.
Data Type	Applies the chosen data type to the column.
Use First Row As Headers	Uses the first row as the column titles.
Replace Values	Carries out a search-and-replace operation on the data in a column or columns. This only affects the complete data in a column.
Merge Queries	Joins a second query table to the current query results and aggregates or adds data from the second to the first.
Append Queries	Adds the data from another query to the current query in the current Power BI Desktop file.
Combine Files	Adds the data from a series of similarly structured files into a single table.

The Transform Ribbon

The Transform ribbon, as its name implies, contains a wealth of functions that can help you to transform your data. The various options it contains are explained in Figure 6-5 and Table 6-2.

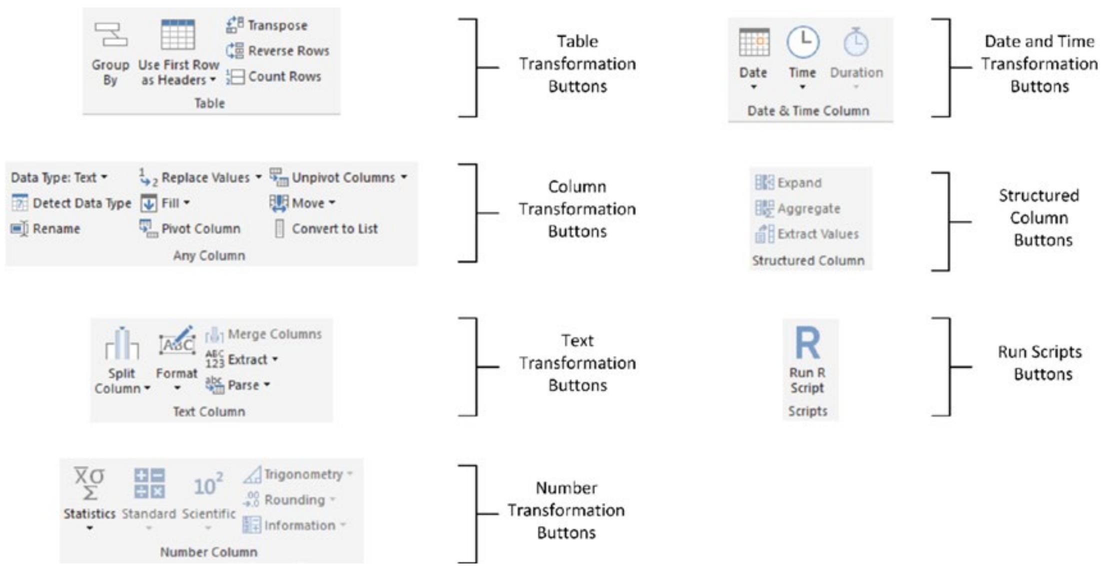


Figure 6-5. *The Query Editor Transform ribbon*

Table 6-2. *Query Editor Transform Ribbon Options*

Option	Description
Group By	Groups the table using a specified set of columns; aggregates any numeric columns for this grouping.
Use First Row As Headers	Uses the first row as the column titles.
Transpose	Transforms the columns into rows and the rows into columns.
Reverse Rows	Displays the source data in reverse order, showing the final rows at the top of the window.
Count Rows	Counts the rows in the table and replaces the data with the row count.
Data Type	Applies the chosen data type to the column.
Detect Data Type	Detects the correct data type to apply to multiple columns.
Rename	Renames a column.
Replace Values	Carries out a search-and-replace operation inside a column, replacing a specified value with another value.
Fill	Copies the data from cells above or below into empty cells in the column.
Pivot Column	Creates a new set of columns using the data in the selected column as the column titles.
Unpivot Columns	Takes the values in a set of columns and unpivots the data, creating two new columns using the column headers as the descriptive elements.
Move	Moves a column.
Convert to List	Converts the contents of a column to a list.
Split Column	Splits a column into one or many columns at a specified delimiter or after a specified number of characters.
Format	Modifies the text format of data in a column (uppercase, lowercase, capitalization) or removes trailing spaces.
Merge Columns	Takes the data from several columns and places it in a single column, adding an optional separator character.
Extract	Replaces the data in a column using a defined subset of the current data. You can specify a number of characters to keep from the start or end of the column, set a range of characters beginning at a specified character, or even list the number of characters in the column.
Parse	Creates an XML or JSON document from the contents of each cell in a column.
Statistics	Returns the Sum, Average, Maximum, Minimum, Median, Standard Deviation, Count, or Distinct Value Count for all the values in the column.
Standard	Carries out a basic mathematical calculation (add, subtract, divide, multiply, integer-divide, or return the remainder) using a value that you specify applied to each cell in the column.
Scientific	Carries out a basic scientific calculation (square, cube, power of n, square root, exponent, logarithm, or factorial) for each cell in the column.

(*continued*)

Table 6-2. (*continued*)

Option	Description
Trigonometry	Carries out a basic trigonometric calculation (Sine, Cosine, Tangent, ArcSine, ArcCosine, or ArcTangent) using a value that you specify applied to each cell in the column.
Rounding	Rounds the values in the column either to the next integer (up or down) or to a specified factor.
Information	Replaces the value in the column with simple information: Is Odd, Is Even, or Positive/Negative.
Date	Isolates an element (day, month, year, etc.) from a date value in a column.
Time	Isolates an element (hour, minute, second, etc.) from a date/time or time value in a column.
Duration	Calculates the duration from a value that can be interpreted as a duration in days, hours, minutes, and so forth.
Expand	Adds the (identically structured) data from another query to the current query.
Aggregate	Calculates the sum or product of numeric columns from another query and adds the result to the current query.
Extract Values	Extracts the values of the contents of a column as a single text value.
Scripts	Runs scripts from languages such as "R."

The Add Column Ribbon

The Add Column ribbon does a lot more than just add columns. It also contains functions to break columns down into multiple columns, and to add columns containing dates and calculations based on existing columns. The various options it contains are explained in Figure 6-6 and Table 6-3.

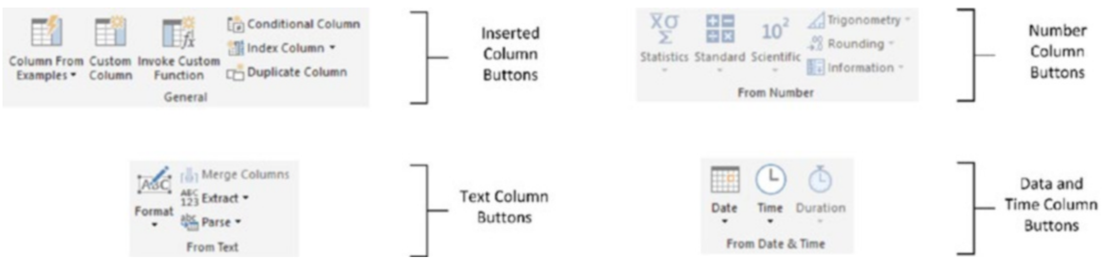


Figure 6-6. *The Query Editor Add Column ribbon*

Table 6-3. *Query Editor Add Column Ribbon Options*

Option	Description
Column From Examples	Lets you use one or more columns as examples to create a new column.
Custom Column	Adds a new column using a formula to create the column's contents.
Invoke Custom Function	Applies an "M" language function to every row.
Conditional Column	Adds a new column that conditionally adds the values from the selected column.
Index Column	Adds a sequential number in a new column to uniquely identify each row.
Duplicate Column	Creates a copy of the current column.
Format	Modifies the text format of data in a column (uppercase, lowercase, capitalization) or removes trailing spaces.
Merge Columns	Takes the data from several columns and places it in a single column, adding an optional separator character.
Extract	Replaces the data in a column using a defined subset of the current data. You can specify a number of characters to keep from the start or end of the column, set a range of characters beginning at a specified character, or even list the number of characters in the column.
Parse	Creates an XML or JSON document from the contents of each cell in a column.
Statistics	Returns the Sum, Average, Maximum, Minimum, Median, Standard Deviation, Count, or Distinct Value Count for all the values in the column.
Standard	Carries out a basic mathematical calculation (add, subtract, divide, multiply, integer-divide, or return the remainder) using a value that you specify applied to each cell in the column.
Scientific	Carries out a basic scientific calculation (square, cube, power of n, square root, exponent, logarithm, or factorial) for each cell in the column.
Trigonometry	Carries out a basic trigonometric calculation (Sine, Cosine, Tangent, ArcSine, ArcCosine, or ArcTangent) using a value that you specify applied to each cell in the column.
Rounding	Rounds the values in the column either to the next integer (up or down) or to a specified factor.
Information	Replaces the value in the column with simple information: Is Odd, Is Even, or Positive/Negative.
Date	Isolates an element (day, month, year, etc.) from a date value in a column.
Time	Isolates an element (hour, minute, second, etc.) from a date/time or time value in a column.
Duration	Calculates the duration from a value that can be interpreted as a duration in days, hours, minutes, and seconds.

The View Ribbon

The View ribbon lets you alter some of the Query Editor settings and see the underlying data transformation code. The various options that it contains are explained in the next chapter.

Dataset Shaping

So you are now looking at a data table that you have loaded into Power BI Desktop. For argument's sake, let's assume that it is the C:\PowerBiDesktopSamples\CH06\CH06Example1.pbix file from the sample data directory, and that you have clicked the Edit Queries button to display the Power BI Desktop Query Editor. What can you do to the BaseData dataset that is now visible? It is time to take a look at some of the core techniques that you can apply to shape the initial dataset. These include the following:

- Renaming columns
- Reordering columns
- Removing columns
- Merging columns
- Removing records
- Removing duplicate records
- Filtering the dataset

I have grouped these techniques together as they affect the initial size and shape of the data. Also, it is generally not only good practice, but also easier for you, the data modeler, if you begin by excluding any rows and columns that you do not need. I also find it easier to understand datasets if the columns are logically laid out and given comprehensible names from the start. All in all, this makes working with the data easier in the long run.

Renaming Columns

Although we took a quick look at renaming columns in the first pages of this chapter, let's look at this technique again in more detail. I admit that renaming columns is not actually modifying the form of the data table. However, when dealing with data, I consider it vital to have all data clearly identifiable. This implies meaningful column names being applied to each column. Consequently, I consider this modification to be fundamental to the shape of the data and also as an essential best practice when importing source data.

To rename a column:

1. Click inside the column that you want to rename.
2. Click Transform to activate the Transform ribbon.
3. Click the Rename button. The column name will be highlighted.
4. Enter the new name or edit the existing name.
5. Press Enter or click outside the column title.

The column will now have a new title. The Applied Steps list on the right will now contain another element, Renamed Columns. This step will be highlighted.

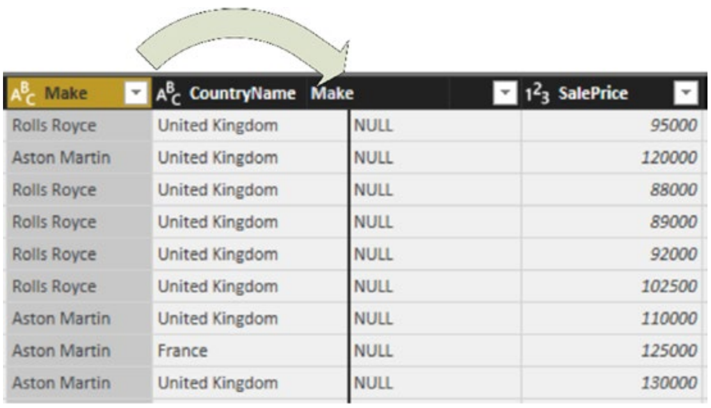
■ **Note** As an alternative to using the Transform ribbon, you can right-click the column title and select Rename.

Reordering Columns

Power BI Desktop will load data as it is defined in the data source. Consequently, the column sequence will be entirely dependent on the source data (or by a SQL query if you used a source database, as described in Chapter 2). This column order need not be definitive, however, and you can reorder the columns if that helps you understand and deal with the data. Do the following to change column order:

1. Click the header of the column you want to move.
2. Drag the column left or right to its new position. You will see the column title slide laterally through the column titles as you do this, and a thicker gray line will indicate where the column will be placed once you release the mouse button. Reordered Columns will appear in the Applied Steps list.

Figure 6-7 shows this operation.



Make	CountryName	Make	SalePrice
Rolls Royce	United Kingdom	NULL	95000
Aston Martin	United Kingdom	NULL	120000
Rolls Royce	United Kingdom	NULL	88000
Rolls Royce	United Kingdom	NULL	89000
Rolls Royce	United Kingdom	NULL	92000
Rolls Royce	United Kingdom	NULL	102500
Aston Martin	United Kingdom	NULL	110000
Aston Martin	France	NULL	125000
Aston Martin	United Kingdom	NULL	130000

Figure 6-7. *Reordering columns*

If your query contains dozens—or even hundreds—of columns, you may find that dragging a column around can be slow and laborious. Equally, if columns are extremely wide, it can be difficult to "nudge" a column left or right. Power BI Desktop can come to your aid in these circumstances with the Move button in the Transform ribbon. Clicking this button gives you the menu options that are outlined in Table 6-4.

Table 6-4. *Move Button Options*

Option	Description
Left	Moves the currently selected column to the left of the column on its immediate left.
Right	Moves the currently selected column to the right of the column on its immediate right.
To Beginning	Moves the currently selected column to the left of all the columns in the query.
To End	Moves the currently selected column to the right of all the columns in the query.

The Move command also works on a set of columns that you have selected by Ctrl-clicking and/or Shift-clicking. Indeed, you can move a selection of columns that is not contiguous if you need to.

■ **Note** You need to select a column (or a set of columns) before clicking the Move button. If you do not, then the first time that you use Move, Power BI Desktop Query selects the column(s) but does not move it.

Removing Columns

So how do you delete a column or series of columns? Like this:

1. Click inside the column you want to delete, or if you want to delete several columns at once, Ctrl-click the titles of the columns that you want to delete.
2. Click the Remove Columns button in the Home ribbon. The column(s) will be deleted and Removed Columns will be the latest element in the Applied Steps list.

When working with imported datasets over which you have had no control, you may frequently find that you only need a few columns of a large data table. If this is the case, you will soon get tired of Ctrl-clicking numerous columns to select those you want to remove. Power BI Desktop has an alternative method. Just select the columns you want to keep and delete the others. To do this:

1. Ctrl-click the titles of the columns that you want to keep.
2. Click the small triangle in the Remove Columns button in the Home ribbon. Select Remove Other Columns from the menu. All unselected columns will be deleted and Removed Other Columns will be added to the Applied Steps list.

When selecting a contiguous range of columns to remove or keep, you can use the standard Windows Shift-click technique to select from the first to the last column in the block of columns that you want to select.

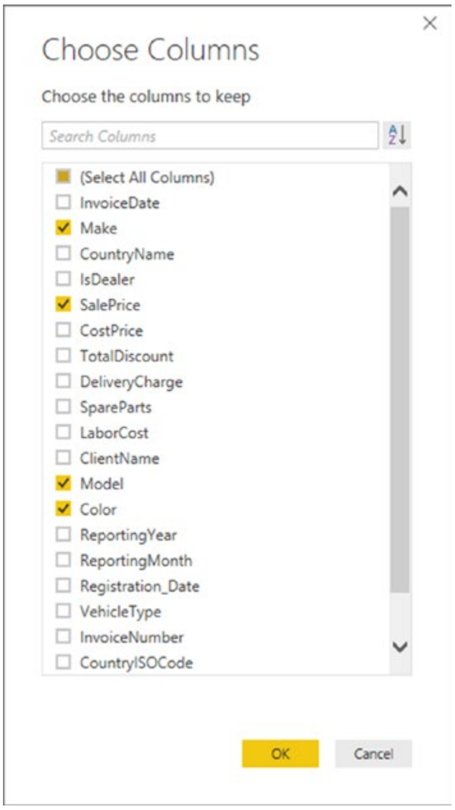
■ **Note** Both of these options for removing columns are also available from the context menu, if you prefer. It shows Remove (or Remove Columns, if there are several columns selected) when deleting columns, as well as Remove Other Columns if you right-click a column title.

Choosing Columns

If you prefer not to scroll through a wide dataset, yet still need to select a subset of columns as the basis for your reports, then there is another way to define the collection of fields that you want to use. You can choose the columns that you want to keep (and, by definition, those that you want to exclude) like this:

1. Open the sample file CH06Example1.pbix in the folder C:\PowerBiDesktopSamples\CH06 unless it is already open.
2. In the Home ribbon, click the Edit Queries button. The Query Editor will open.
3. In the Home ribbon of the Query Editor, click the Choose Columns button.
4. Click (Select All Columns) to deselect the entire collection of columns in the dataset.

5. Select the columns Make, Model, Color, and SalePrice. The Choose Columns dialog will look like the one in Figure 6-8.


Choose Columns

Choose the columns to keep

Search Columns

- (Select All Columns)
- [ ] InvoiceDate
- [✓] Make
- [ ] CountryName
- [ ] IsDealer
- [✓] SalePrice
- [ ] CostPrice
- [ ] TotalDiscount
- [ ] DeliveryCharge
- [ ] SpareParts
- [ ] LaborCost
- [ ] ClientName
- [✓] Model
- [✓] Color
- [ ] ReportingYear
- [ ] ReportingMonth
- [ ] Registration_Date
- [ ] VehicleType
- [ ] InvoiceNumber
- [ ] CountryISOCode

Figure 6-8. *The Choose Columns dialog*

6. Click OK. The Query Editor will only display the columns that you selected.

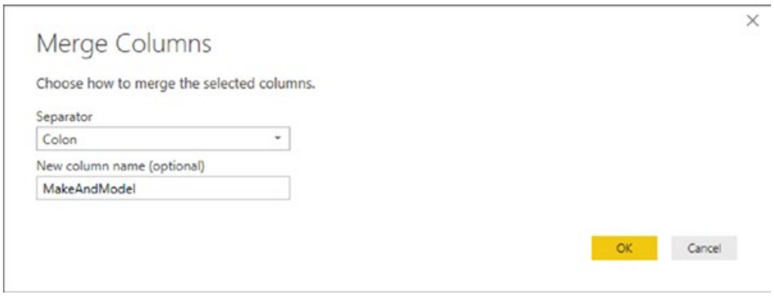
The Choose Columns dialog comes with a couple of extra functions that you might find useful when choosing the set of columns that you want to work with:

- You can sort the column list in alphabetical order (or, indeed, revert to the original order) by clicking the Sort icon (the small A-Z) at the top right of the Choose Columns dialog and selecting the required option.
- You can filter the list of columns that is displayed simply by entering a few characters in the Search Columns field at the top of the dialog and then pressing Enter.
- The (Select All Columns) option switches between selecting and deselecting all the columns in the list.

Merging Columns

Source data is not always exactly as you wish it could be (and that is sometimes a massive understatement). Certain data sources could have data spread over many columns that could equally well be merged into a single column. So it probably comes as no surprise to discover that Power BI Desktop Query can carry out this kind of operation too. Here is how to do it:

1. Ctrl-click the headers of the columns that you want to merge (Make and Model in the BaseData dataset in this example).
2. In the Transform ribbon, click the Merge Columns button. The Merge Columns dialog will be displayed.
3. From the Separator pop-up menu, select one of the available separator elements. I chose Colon in this example.
4. Enter a name for the column that will be created from the two original columns (I am calling it MakeAndModel). The dialog should look like Figure 6-9.



Merge Columns

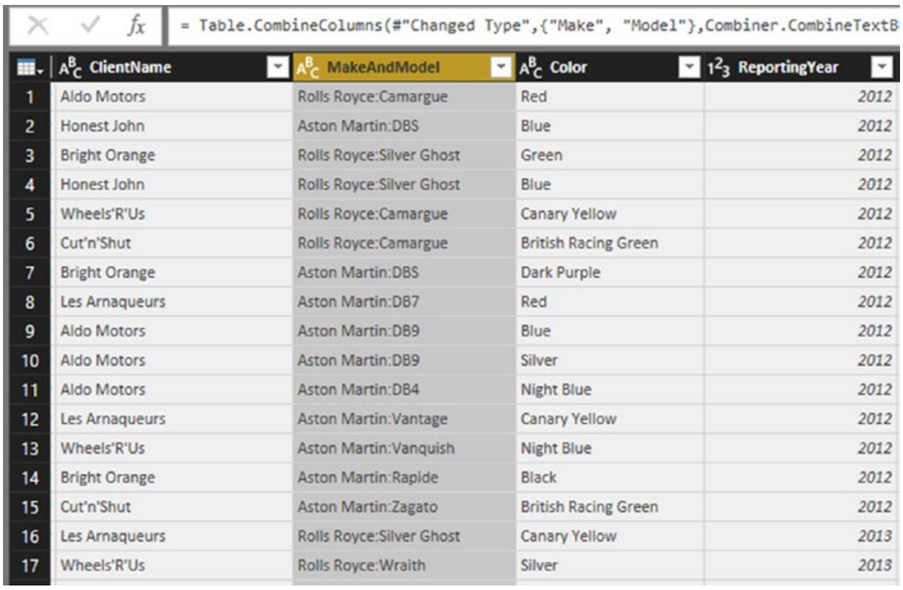
Choose how to merge the selected columns.

Separator: Colon

New column name (optional): MakeAndModel

Figure 6-9. *The Merge Columns dialog*

5. Click OK. The columns that you selected will be replaced by the data from all the columns, as shown in Figure 6-10.



```
= Table.CombineColumns(#"Changed Type",{"Make", "Model"},Combiner.CombineTextB
```

	ClientName	MakeAndModel	Color	ReportingYear
1	Aldo Motors	Rolls Royce:Camargue	Red	2012
2	Honest John	Aston Martin:DBS	Blue	2012
3	Bright Orange	Rolls Royce:Silver Ghost	Green	2012
4	Honest John	Rolls Royce:Silver Ghost	Blue	2012
5	Wheels'R'Us	Rolls Royce:Camargue	Canary Yellow	2012
6	Cut'n'Shut	Rolls Royce:Camargue	British Racing Green	2012
7	Bright Orange	Aston Martin:DBS	Dark Purple	2012
8	Les Arnaqueurs	Aston Martin:DB7	Red	2012
9	Aldo Motors	Aston Martin:DB9	Blue	2012
10	Aldo Motors	Aston Martin:DB9	Silver	2012
11	Aldo Motors	Aston Martin:DB4	Night Blue	2012
12	Les Arnaqueurs	Aston Martin:Vantage	Canary Yellow	2012
13	Wheels'R'Us	Aston Martin:Vanquish	Night Blue	2012
14	Bright Orange	Aston Martin:Rapide	Black	2012
15	Cut'n'Shut	Aston Martin:Zagato	British Racing Green	2012
16	Les Arnaqueurs	Rolls Royce:Silver Ghost	Canary Yellow	2013
17	Wheels'R'Us	Rolls Royce:Wraith	Silver	2013

Figure 6-10. *The result of merging columns*

I need to make a few comments about this process:

- You can select as many columns as you want when merging columns.
- If you do not give the resulting column a name in the Merge Columns dialog, it will simply be renamed Merged. You can always rename it later if you want.
- The order in which you select the columns affects the way that the data is merged. So, always begin by selecting the column whose data must appear at the left of the merged column, then the column whose data should be next, and so forth. You do not have to select columns in the order that they initially appeared in the dataset.
- If you do not want to use any of the standard separators that Power BI Desktop Query suggests, you can always define your own. Just select --Custom-- in the pop-up menu in the Merge Columns dialog. A new box will appear in the dialog, in which you can enter your choice of separator. This can be composed of several characters if you really want.
- Merging columns from the Transform ribbon removes all the selected columns and replaces them with a single column. The same option is also available from the Add Column ribbon—only in this case, this operation adds a new column and leaves the original columns in the dataset.

■ **Note** This option is also available from the context menu if you right-click a column title.

The available merge separators are described in Table 6-5.

Table 6-5. *Merge Separators*

Option	Description
Colon	Uses the colon (:) as the separator.
Comma	Uses the comma (,) as the separator.
Equals Sign	Uses the equals sign (=) as the separator.
Semi-Colon	Uses the semicolon (;) as the separator.
Space	Uses the space ( ) as the separator.
Tab	Uses the tab character as the separator.
Custom	Lets you enter a custom separator.

■ **Tip** You can split, remove, and duplicate columns using the context menu if you prefer. Just remember to right-click the column title to display the correct context menu.

Going to a Specific Column

Power BI Desktop can load datasets that contain hundreds of columns. As scrolling left and right across dozens of columns can be more than a little frustrating, you can always jump to a specific column at any time.

1. In the Home ribbon of the Query Editor, click the small triangle at the bottom of the Choose Columns button. The Go to Column dialog will appear.
2. Select the column you want to move to. The dialog will look like Figure 6-11.

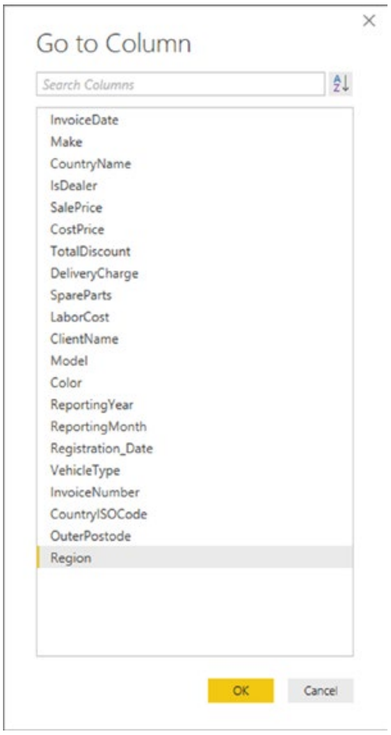


Figure 6-11. *The Go to Column dialog*

3. Click OK. Power BI Desktop will select the chosen column.

■ **Tip** If you prefer, you can double-click a column name in the Go to Column dialog to move to the chosen column.

Removing Records

You may not always need *all* the data that you have loaded into a Power BI Desktop query. There could be several possible reasons for this:

- You are taking a first look at the data and you only need a sample to get an idea of what the data is like.
- The data contains records that you clearly do not need and that you can easily identify from the start.
- You are testing data cleansing and you want a smaller dataset to really speed up the development of a complex data extractions and transformation process.
- You want to analyze a reduced dataset to extrapolate theses and inferences, and to save analysis on a full dataset for later, or even using a more industrial-strength toolset such as SQL Server Integration Services.

To allow you to reduce the size of the dataset, Power BI Desktop proposes two basic approaches out of the box:

- Keep certain rows
- Remove certain rows

Inevitably, the technique that you adopt will depend on the circumstances. If it is easier to specify the rows to sample by inclusion, then the keep-certain-rows approach is the best option to take. Inversely, if you want to proceed by exclusion, then the remove-certain-rows technique is best. Let's look at each of these in turn.

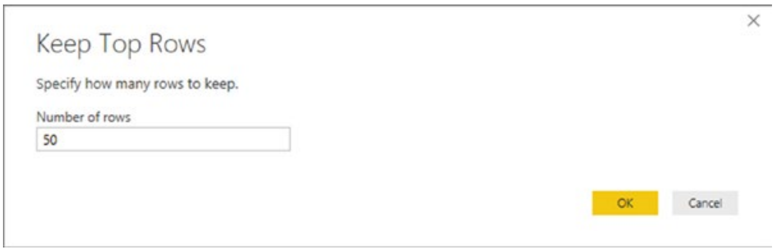
Keeping Rows

This approach lets you specify the rows that you want to continue using. It is based on the application of one of the following three choices:

- Keep the top *n* records.
- Keep the bottom *n* records.
- Keep a specified range of records—that is, keep *n* records every *y* records.

Most of these techniques are very similar, so let's start by imagining that you want to keep the top 50 records in the sample C:\PowerBiDesktopSamples\CH06\CH06Example1.pbix file.

1. In the Home ribbon of the Power BI Query Editor, click the Keep Rows button. The menu will appear.
2. Select Keep Top Rows. The Keep Top Rows dialog will appear.
3. Enter **50** in the "Number of rows" box, as shown in Figure 6-12.


Keep Top Rows

Specify how many rows to keep.

Number of rows

50

***Figure 6-12.** The Keep Top Rows dialog*

4. Click OK. All but the first 50 records are deleted and Kept First Rows is added to the Applied Steps list.

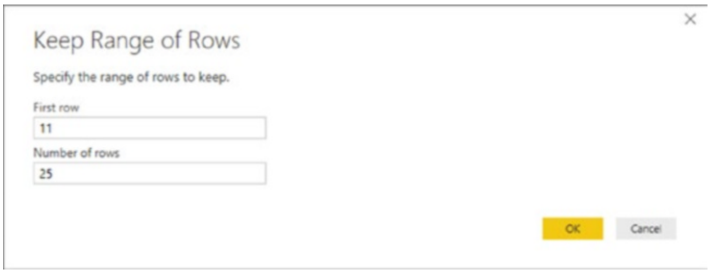
To keep the bottom *n* rows, the technique is virtually identical. Follow the steps in the previous example, but select Keep Bottom Rows in step 2. In this case, the Applied Steps list displays Kept Last Rows.

To keep a range of records, you need to specify a starting record and the number of records to keep from then on. For instance, suppose that you wish to lose the first 10 records but keep the following 25. This is how to go about it:

1. In the Home ribbon, click the Keep Rows button.
2. Select Keep Range of Rows. The Keep Range of Rows dialog will appear.

3. Enter **11** in the “First row” box.

4. Enter **25** in the “Number of rows” box, as shown in Figure 6-13.


Keep Range of Rows

Specify the range of rows to keep.

First row: 11

Number of rows: 25

Figure 6-13. *The Keep Range of Rows dialog*

5. Click OK. All but records 1–10 and 36 to the end are deleted and Kept Range Of Rows is added to the Applied Steps list.

Removing Rows

Removing rows is a nearly identical process to the one you just used to keep rows. As removing the top or bottom *n* rows is highly similar, I will not go through it in detail. All you have to do is click the Remove Rows button in the Home ribbon and follow the process as if you were keeping rows. The Applied Steps list will read Removed Top Rows or Removed Bottom Rows in this case, and rows will be removed instead of being kept in the dataset, of course.

The remove rows approach does have one very useful option that can be applied as a sampling technique. It allows you to remove one or more records every few records to produce a subset of the source data. To do this, you need to do the following:

1. Click the Remove Rows button in the Query window Home ribbon. The menu will appear.

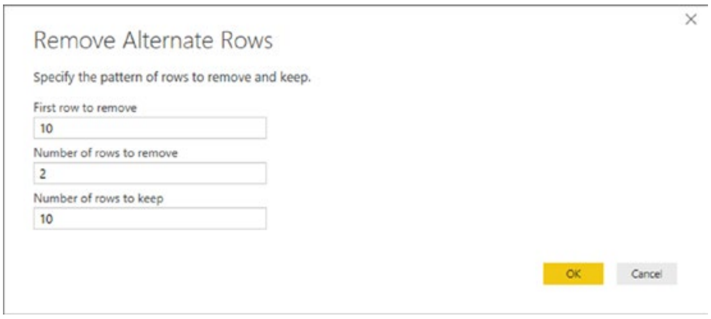
2. Select Remove Alternate Rows. The Remove Alternate Rows dialog will appear.

3. Enter **10** as the First row to remove.

4. Enter **2** as the Number of rows to remove.

5. Enter **10** as the Number of rows to keep.

The dialog will look like Figure 6-14.


Remove Alternate Rows

Specify the pattern of rows to remove and keep.

First row to remove: 10

Number of rows to remove: 2

Number of rows to keep: 10

Figure 6-14. *The Remove Alternate Rows dialog*

6. Click OK. All but the records matching the pattern you entered in the dialog are removed. Removed Alternate Rows is then added to the Applied Steps list.

■ **Note** If you are really determined to extract a sample that you consider to be representative of the key data, then you can always filter the data before subsetting it to exclude any outliers. Filtering data is explained later in this chapter.

Removing Blank Rows

If your source data contains completely blank (empty) rows, you can delete these as follows:

1. Click the Remove Rows button in the Query window Home ribbon. The menu will appear.
2. Select Remove Blank Rows.

This results in empty rows being deleted. Removed Blank Rows is then added to the Applied Steps list.

Removing Duplicate Records

An external source of data might not be quite as perfect as you might hope. One of the most annoying features of poor data is the presence of duplicates. These are insidious since they falsify results and are not always visible. If you suspect that the data table contains *strict* duplicates (that is, where every field is identical in two or more records), then you can remove the duplicates like this:

1. Click the Remove Duplicates in the popup menu for the table (this is at the top left of the table grid). All duplicate records are deleted and Removed Duplicates is added to the Applied Steps list.

■ **Note** I must stress that this approach will only remove *completely* identical records. If two records have just one different character or a number but everything else is identical, then they are *not* considered duplicates by the Power BI Desktop Query Editor. Alternatively, if you want to isolate and examine the duplicate records, then you can display only completely identical records by selecting Keep Duplicates from the popup menu for the table.

So if you suspect or are sure that the data table you are dealing with contains duplicates, what are the practical solutions? This can be a real conundrum, but there are some basic techniques that you can apply:

- Remove all columns that you are sure you will not be using later in the data-handling process. This way, Power BI Desktop will only be asked to compare essential data across potentially duplicate records.
- Group the data on the core columns (this is explained later in this chapter).

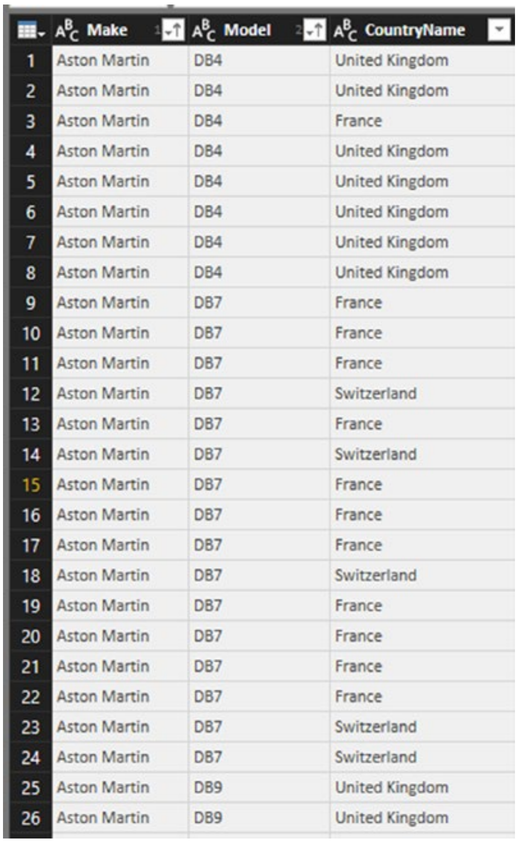
■ **Note** As you have seen, Power BI Desktop Query can help you to home in on the essential elements in a dataset in just a few clicks. If anything, you need to be careful that you are *not removing* valuable data—and consequently skewing your analysis—when excluding data from the query.

Sorting Data

Although not strictly a data modification step, sorting an imported table will probably be something that you want to do at some stage, if only to get a clearer idea of the data that you are dealing with. Do the following to sort the data:

1. Close the file CH06Example1.pbix (if, indeed, you have been using it to test the techniques explained so far in this chapter) without saving it.
2. Open the file C:\PowerBiDesktopSamples\CH06\CH06Example2.pbix.
3. Click Edit Queries to open the Query Editor.
4. Click inside the column you wish to sort by.
5. Click Sort Ascending (the A/Z icon) or Sort Descending (the Z/A icon) in the Home ribbon.

The data is sorted in either alphabetical (smallest to largest) or reverse alphabetical (largest to smallest) order. If you want to carry out a complex sort operation (that is, first by one column and then by another if the first column contains the same element over several rows), you do this simply by sorting the columns one after another. Power BI Desktop Query Editor adds a tiny 1, 2, 3, and so on to the right of the column title to indicate the sort sequence. You can see this in Figure 6-15, where I sorted first on the column Make, and finally on the column Model.



	Make 1	Model 2	CountryName
1	Aston Martin	DB4	United Kingdom
2	Aston Martin	DB4	United Kingdom
3	Aston Martin	DB4	France
4	Aston Martin	DB4	United Kingdom
5	Aston Martin	DB4	United Kingdom
6	Aston Martin	DB4	United Kingdom
7	Aston Martin	DB4	United Kingdom
8	Aston Martin	DB4	United Kingdom
9	Aston Martin	DB7	France
10	Aston Martin	DB7	France
11	Aston Martin	DB7	France
12	Aston Martin	DB7	Switzerland
13	Aston Martin	DB7	France
14	Aston Martin	DB7	Switzerland
15	Aston Martin	DB7	France
16	Aston Martin	DB7	France
17	Aston Martin	DB7	France
18	Aston Martin	DB7	Switzerland
19	Aston Martin	DB7	France
20	Aston Martin	DB7	France
21	Aston Martin	DB7	France
22	Aston Martin	DB7	France
23	Aston Martin	DB7	Switzerland
24	Aston Martin	DB7	Switzerland
25	Aston Martin	DB9	United Kingdom
26	Aston Martin	DB9	United Kingdom

Figure 6-15. *Sorting multiple columns*

As sorting data is considered part of the data modification process, it also appears in the Applied Steps list as Sorted Rows.

■ **Note** An alternative technique for sorting data is to click the pop-up menu for a column (the downward-facing triangle at the right of a column title) and select Sort Ascending or Sort Descending from the pop-up menu.

Reversing the Row Order

If you find that the data that you are looking at seems upside-down (that is, with the bottom rows at the top and vice versa), you can reverse the row order in a single click, if you want. To do this, do the following:

In the Transform ribbon, click the Reverse Rows button.

The entire dataset will be reversed and the bottom row will now be the top row.

Undoing a Sort Operation

If you subsequently decide that you do not want to keep your data sorted, you can undo the sort operation at any time, as follows:

1. Click the sort icon at the right of the name of the column that you used as the basis for the sort operation. The context menu will appear, as you can see in Figure 6-16.

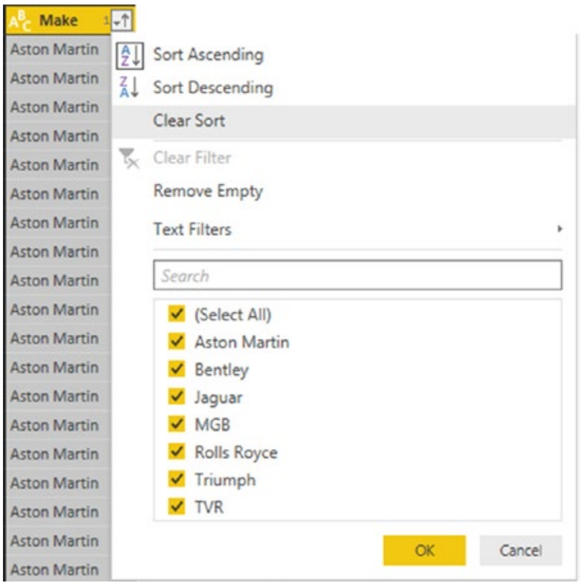


Figure 6-16. *Removing a sort operation*

2. Click Clear Sort.

The sort order that you applied will be removed, and the data will revert to its original row order.

■ **Note** If you sorted the dataset on several columns, you can choose either to remove all the sort order that you applied by clicking the *first* column that you used to sort the data. This will remove the sort order that you applied to *all* the columns that you used to define the sort criteria. If all you want to do is undo the sort on the final column in a set of columns used to sort the recordset, then you can clear only the sort operation on this column.

Filtering Data

The most frequently used way of limiting a dataset is, in my experience, the use of filters on the table that you have loaded. Now, I realize that you may be coming to Power BI Desktop after years with Excel, or after some time using Power Pivot, and that the filtering techniques that you are about to see probably look much like the ones you have used in those two tools. However, because it is fundamental to include and exclude appropriate records when loading source data, I will thoroughly handle Power BI Desktop filters, even if this means that certain readers will experience a strong sense of déjà vu.

Here are two basic approaches for filtering data in Power BI Desktop:

- Select one or more specific values from the unique list of elements in the chosen column.
- Define a range of data to include or exclude.

The first option is common to all data types, whether they are text, number, or data/time. The second approach varies according to the data type of the column that you are using to filter data.

Selecting Specific Values

Selecting one or more values present in a column of data is as easy as this (assuming that you are still using the Power BI Desktop file CH06Example2.pbix and are in the Query Editor):

1. Click a column's pop-up menu. (I used Make in the sample dataset in this example.) The filter menu appears.
2. Check all elements that you want to retain and uncheck all elements that you wish to exclude. In this example, I kept Bentley and Rolls-Royce, as shown in Figure 6-17.

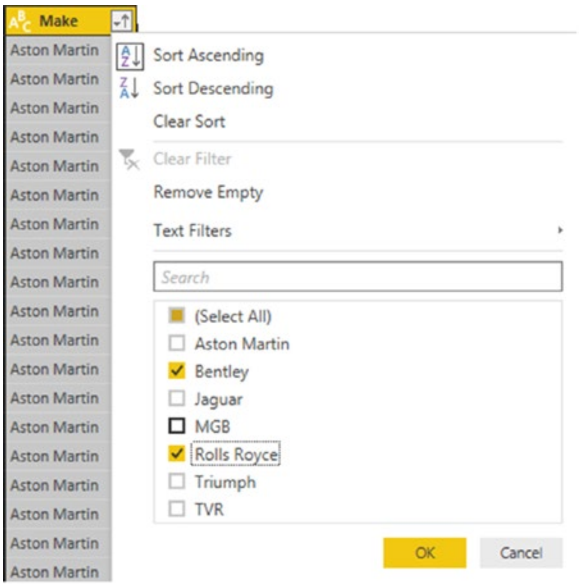


Figure 6-17. *A filter menu*

3. Click OK. The Applied Steps list adds Filtered Rows.

■ **Note** You can deselect all items by clicking the (Select All) check box; reselect all the items by selecting this box again. It follows that, if you want to keep only a few elements, it may be faster to unselect all of them first and then only select the ones that you want to keep. If you want to exclude any records without a value in the column that you are filtering on, then select Remove Empty from the filter menu.

Finding Elements in the Filter List

Scrolling up and down in a filter list can get extremely laborious. A fast way of limiting the list to a subset of available elements is to do the following:

1. Click the pop-up menu for a column. (I use Model in the sample dataset in this example.) The filter menu appears.

2. Enter a letter or a few letters in the Search box. The list shortens with every letter or number that you enter. If you enter **ar**, then the filter popup will look like Figure 6-18.

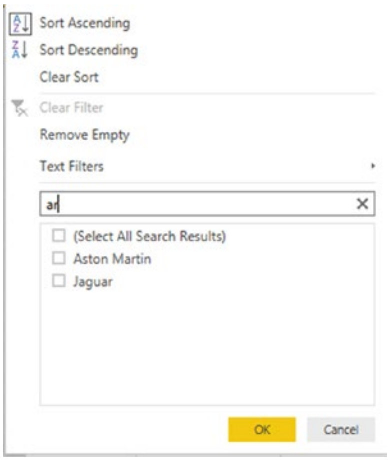


Figure 6-18. *Searching the filter menu*

To remove a filter, all that you have to do is click the cross that appears at the right of the Search box.

Filtering Text Ranges

If a column contains text, then you can apply specific options to filter the data. These elements are found in the filter popup of any text-based column in the Text Filters submenu. The choices are given in Table 6-6.

Table 6-6. *Text Filter Options*

Filter Option	Description
Equals	Sets the text that must match the cell contents.
Does Not Equal	Sets the text that must not match the cell contents.
Begins With	Sets the text at the left of the cell contents.
Does Not Begin With	Sets the text that must not appear at the left of the cell contents.
Ends With	Sets the text at the right of the cell contents.
Does Not End With	Sets the text that must *not* appear at the right of the cell contents.
Contains	Lets you enter a text that will be part of the cell contents.
Does Not Contain	Lets you enter a text that will *not* be part of the cell contents.

Filtering Numeric Ranges

If a column contains numbers, then there are also specific options that you can apply to filter the data. You'll find these elements in the filter popup of any text-based column in the Number Filters submenu. The choices are given in Table 6-7.

Table 6-7. *Numeric Filter Options*

Filter Option	Description
Equals	Sets the number that must match the cell contents.
Does Not Equal	Sets the number that must not match the cell contents.
Greater Than	Cell contents must be greater than this number.
Greater Than Or Equal To	Cell contents must be greater than or equal to this number.
Lesser Than	Cell contents must be less than this number.
Lesser Than Or Equal To	Cell contents must be less than or equal to this number.
Between	Cell contents must be between the two numbers.

Filtering Date and Time Ranges

If a column contains dates or times (or both), then specific options can also be applied to filter the data. These elements are found in the filter popup of any text-based column in the Date/Time Filters submenu. The choices are given in Table 6-8.

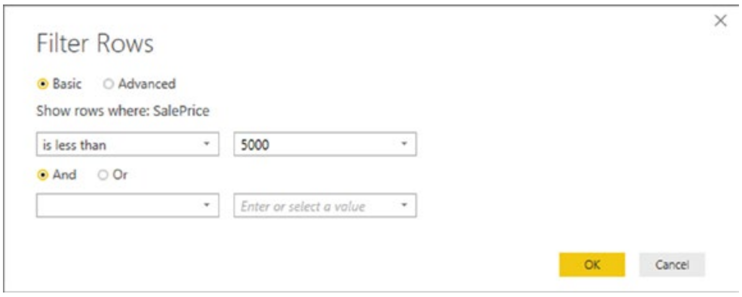
Table 6-8. *Date and Time Filter Options*

Filter Element	Description
Equals	Filters data to include only records for the selected date.
Before	Filters data to include only records up to the selected date.
After	Filters data to include only records after the selected date.
Between	Lets you set an upper and a lower date limit to exclude records outside that range.
In the Next	Lets you specify a number of days, weeks, months, quarters, or years to come.
In the Previous	Lets you specify a number of days, weeks, months, quarters, or years up to the date.
Is Earliest	Filters data to include only records for the earliest date.
Is Latest	Filters data to include only records for the latest date.
Is Not Earliest	Filters data to include only records for dates not including the earliest date.
Is Not Latest	Filters data to include only records for dates not including the latest date.
Day ➤ Tomorrow	Filters data to include only records for the day after the current system date.
Day ➤ Today	Filters data to include only records for the current system date.
Day ➤ Yesterday	Filters data to include only records for the day before the current system date.
Week ➤ Next Week	Filters data to include only records for the next calendar week.
Week ➤ This Week	Filters data to include only records for the current calendar week.
Week ➤ Last Week	Filters data to include only records for the previous calendar week.
Month ➤ Next Month	Filters data to include only records for the next calendar month.
Month ➤ This Month	Filters data to include only records for the current calendar month.
Month ➤ Last Month	Filters data to include only records for the previous calendar month.
Month ➤ Month Name	Filters data to include only records for the specified calendar month.
Quarter ➤ Next Quarter	Filters data to include only records for the next quarter.
Quarter ➤ This Quarter	Filters data to include only records for the current quarter.
Quarter ➤ Last Quarter	Filters data to include only records for the previous quarter.
Quarter ➤ Quarter Name	Filters data to include only records for the specified quarter.
Year ➤ Next Year	Filters data to include only records for the next year.
Year ➤ This Year	Filters data to include only records for the current year.
Year ➤ Last Year	Filters data to include only records for the previous year.
Year ➤ Year To Date	Filters data to include only records for the calendar year to date.
Custom Filter	Lets you set up a specific filter for two possible date ranges.

Filtering Data

Filtering data uses a very similar approach, whatever the type of filter that is applied. As a simple example, here is how to apply a number filter to the sale price to find vehicles that sold for less than £5,000.00:

1. Click the popup menu for the SalePrice column.
2. Click Number Filters. The submenu will appear.
3. Select Less Than. The Filter Rows dialog will be displayed.
4. Enter **5000** in the box next to the "is less than" box, as shown in Figure 6-19.



Filter Rows

[✓ Basic] Advanced

Show rows where: SalePrice

is less than | 5000

[✓ And] Or

Figure 6-19. *The Filter Rows dialog*

5. Click OK. The dataset only displays rows that conform to the filter that you have defined.

Although extremely simple to apply, filters do require a few comments:

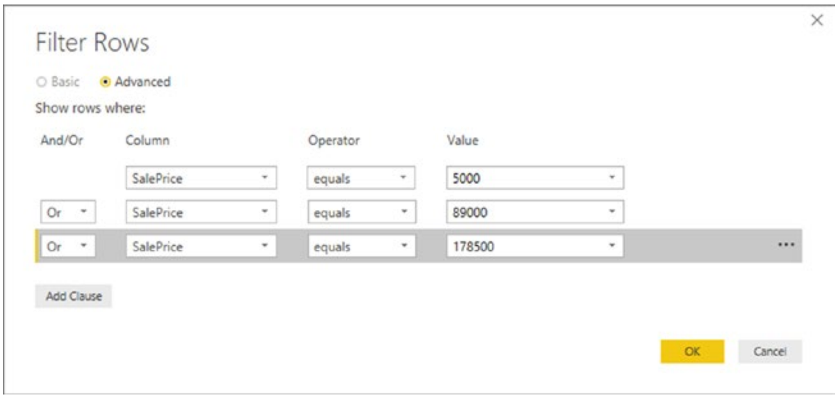
- You can combine up to two elements in a basic filter. These can be mutually inclusive (an AND filter) or they can be an alternative (an OR filter).
- You can combine several elements in an advanced filter—as you will learn in the next section.
- You should not apply any formatting when entering numbers.
- Any text that you filter on is not case-sensitive.
- If you choose the wrong filter, you do not have to cancel and start over. Simply select the correct filter type from the popup in the left-hand boxes in the Filter Rows dialog.

■ **Tip** If you set a filter value that excludes all the records in the table, Power BI Desktop displays an empty table except for the words "This table is empty." You can always remove the filter by clicking the cross to the left of Filtered Rows in the Applied Steps list. This will remove the step and revert the data to its previous state.

Applying Advanced Filters

Should you ever need to be extremely specific when filtering data, you can always use Power BI Desktop's advanced filters. These let you extend the filter elements so that you can include or exclude records to a high level of detail. Here is the procedure:

1. Click the popup menu for the SalePrice column.
2. Click Number Filters. The submenu will appear.
3. Select Equals. The Filter Rows dialog will be displayed.
4. Click Advanced.
5. Enter **5000** as the value for the first filter element in the dialog.
6. Select Or from the popup as the filter type for the second filter element.
7. Select Equals as the operator.
8. Enter **89000** as the value for the second filter in the dialog.
9. Click Add clause. A new filter element will be added to the dialog under the existing elements.
10. Select Equals as the operator.
11. Enter **178500** as the value for the third filter element in the dialog. The Filter dialog will look like the one shown in Figure 6-20.



Filter Rows

○ Basic ● Advanced

Show rows where:

And/Or	Column	Operator	Value
	SalePrice	equals	5000
Or	SalePrice	equals	89000
Or	SalePrice	equals	178500

Figure 6-20. *Advanced filters*

12. Click OK. Only records containing the figures that you entered in the Filter dialog will be displayed in the Power BI Desktop Query Editor.

I would like to finish on the subject of filters with a few comments:

- In the Advanced Filter dialog you can "mix and match" columns and operators to achieve the filter result that you are looking for.

- You can also order the sequence of filters if you ever need to. To do this, simply click inside a filter row and it will appear with a gray background (like the third filter in Figure 6-20). Then click the ellipses at the right of the filter row and select Move Up or Move Down from the popup menu.
- To delete a filter, click the ellipses at the right of the filter row and select Delete.

■ **Note** When you are dealing with really large datasets, you may find that a filter does not always show all the available values from the source data. This is because Power BI Query Editor has loaded only a sample subset of the data. In cases like these you will see an alert in the filter popup menu and a "Load more" link. Clicking this link will force Power BI to reload a larger sample set of data. However, memory restrictions may prevent it loading all the data that you need. In cases like this you should consider modifying the source query if possible so that it brings back a representative dataset that can fit into memory.

Grouping Records

At times, you will need to transform your original data in an extreme way—by grouping the data. This is very different from filtering data, removing duplicates, or cleansing the contents of columns. When you group data, you are altering the structure of the dataset to "roll up" records where you do the following:

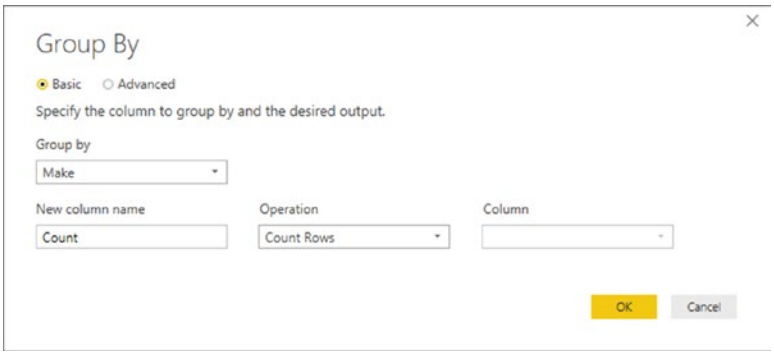
- Define the attribute columns that will become the unique elements in the grouped data table
- Specify which aggregations are applied to any numeric columns included in the grouped table

Grouping is frequently an extremely selective operation. This is inevitable, since the more attribute (that is, non-numeric) columns you choose to group on, the more records you are likely to include in the grouped table. However, this will always depend on the particular dataset you are dealing with, and grouping data efficiently is always a matter of flair, practice, and good, old-fashioned trial and error.

Simple Groups

To understand how grouping works—and how it can radically alter the structure of your dataset—let's see a simple example of row grouping in action:

1. In the sample file CH06Example1.pbix, click inside the Make column.
2. In either the Home ribbon or Transform ribbon, click the Group By button. The Group By dialog will appear, looking like the one in Figure 6-21.


Group By

Basic [✓] Advanced

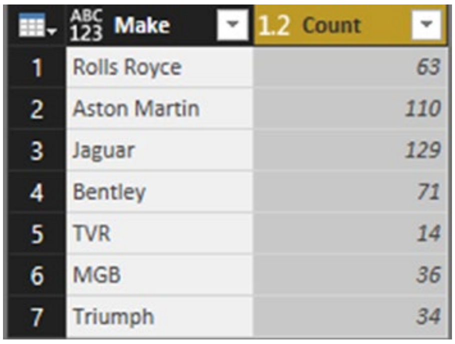
Specify the column to group by and the desired output.

Group by: Make

New column name	Operation	Column
Count	Count Rows	

Figure 6-21. *Simple grouping*

3. Click OK. The dataset will now only contain the list of makes of vehicle and the number of records for each make. You can see this in Figure 6-22.



	Make	Count
1	Rolls Royce	63
2	Aston Martin	110
3	Jaguar	129
4	Bentley	71
5	TVR	14
6	MGB	36
7	Triumph	34

Figure 6-22. *Simple grouping output*

Power BI Desktop will add a step named Grouped Rows to the Applied Steps list when you apply grouping to a dataset.

■ **Note** The best way to cancel a grouping operation is to delete the Grouped Rows step in the Applied Steps list.

Although Power BI Desktop defaults to counting rows, there are several other operations that you can apply when grouping data. These are outlined in Table 6-9.

Table 6-9. *Aggregation Operations when Grouping*

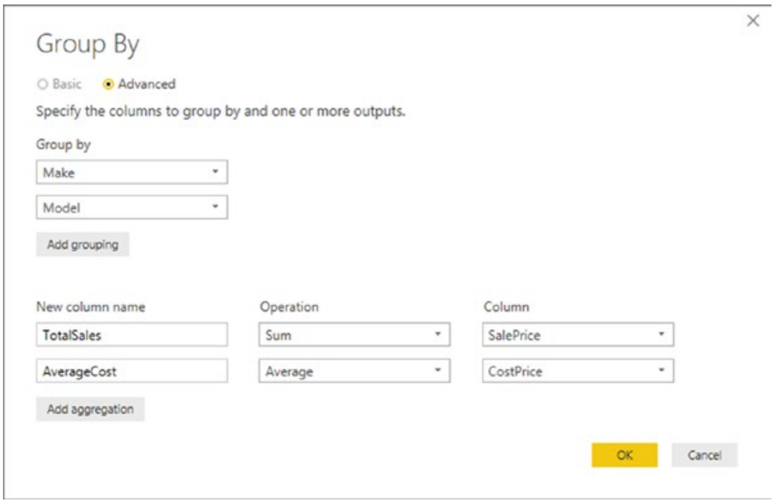
Aggregation Operation	Description
Count Rows	Counts the number of records.
Count Distinct Rows	Counts the number of unique records.
Sum	Returns the total for a numeric column.
Average	Returns the average for a numeric column.
Median	Returns the median value of a numeric column.
Min	Returns the minimum value of a numeric column.
Max	Returns the maximum value of a numeric column.
All Rows	Creates a table of records for each grouped element.

Complex Groups

Power BI Desktop can help you shape your datasets in more advanced ways by creating more complex data groupings. As an example, you could try out the following to group by make and model and add columns showing the total sales value and the average cost:

1. Open the sample file C:\PowerBiDesktopSamples\CH06\CH06Example1.pbix.
2. Click Edit Queries to open the Query Editor.
3. Select the following columns (by Ctrl-clicking the column headers):
    a. Make
    b. Model
4. In either the Home ribbon or Transform ribbon, click Group By.
5. In the New Column Name box, enter **TotalSales**.
6. Select Sum as the operation.
7. Choose SalePrice as the source column in the Column popup list.
8. Click the Add Aggregation button and repeat the operation, only this time, use the following:
    a. New Column Name: **AverageCost**
    b. Operation: Average
    c. Column: CostPrice

The Group By dialog should look like the one in Figure 6-23.



Group By

○ Basic ● Advanced

Specify the columns to group by and one or more outputs.

Group by

Make

Model

Add grouping

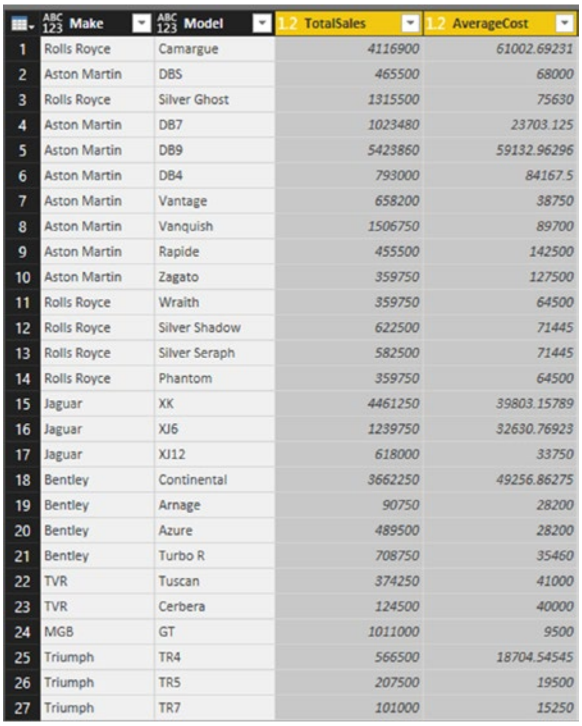
New column name	Operation	Column
TotalSales	Sum	SalePrice
AverageCost	Average	CostPrice

Add aggregation

OK Cancel

Figure 6-23. *The Group By dialog*

9. Click OK. All columns, other than those that you specified in the Group By dialog, are removed, and the table is grouped and aggregated, as shown in Figure 6-24. Grouped Rows will be added to the Applied Steps list. I have also sorted the table by the Make and Model columns to make the grouping easier to comprehend.



	Make	Model	TotalSales	AverageCost
1	Rolls Royce	Camargue	4116900	61002.69231
2	Aston Martin	DBS	465500	68000
3	Rolls Royce	Silver Ghost	1315500	75630
4	Aston Martin	DB7	1023480	23703.125
5	Aston Martin	DB9	5423860	59132.96296
6	Aston Martin	DB4	793000	84167.5
7	Aston Martin	Vantage	658200	38750
8	Aston Martin	Vanquish	1506750	89700
9	Aston Martin	Rapide	455500	142500
10	Aston Martin	Zagato	359750	127500
11	Rolls Royce	Wraith	359750	64500
12	Rolls Royce	Silver Shadow	622500	71445
13	Rolls Royce	Silver Seraph	582500	71445
14	Rolls Royce	Phantom	359750	64500
15	Jaguar	XK	4461250	39803.15789
16	Jaguar	XJ6	1239750	32630.76923
17	Jaguar	XJ12	618000	33750
18	Bentley	Continental	3662250	49256.86275
19	Bentley	Arnage	90750	28200
20	Bentley	Azure	489500	28200
21	Bentley	Turbo R	708750	35460
22	TVR	Tuscan	374250	41000
23	TVR	Cerbera	124500	40000
24	MGB	GT	1011000	9500
25	Triumph	TR4	566500	18704.54545
26	Triumph	TR5	207500	19500
27	Triumph	TR7	101000	15250

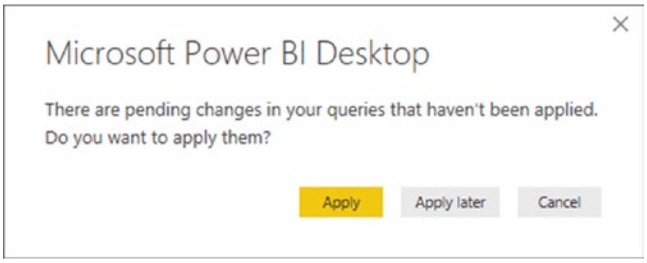
Figure 6-24. *Grouping a dataset*

If you have created a really complex group and then realized that you need to change the order of the columns, all is not lost. You can alter the order of the columns in the output by clicking the ellipses to the right of each column definition in the Group By dialog and selecting Move Up or Move Down. The order of the columns in the Group By dialog will be the order of the columns (left to right) in the resulting dataset.

■ **Note** You do not have to Ctrl-click to select the grouping columns. You can add them one by one to the Group By dialog by clicking the Add Aggregation button. Equally, you can remove grouping columns (or added and aggregated columns) by clicking the ellipses to the right of a column name and selecting Delete from the popup menu.

Saving Changes in the Query Editor

As you would expect, you can save any changes that you have made when using the Power BI Desktop Query Editor at any time. However, you need to be aware that when you click the Save icon above the ribbon (or if you click File ➤ Save), you will be presented with a choice in the dialog that you can see in Figure 6-25.



Microsoft Power BI Desktop

There are pending changes in your queries that haven't been applied. Do you want to apply them?

Figure 6-25. *Applying pending changes before saving in the Query Editor*

At this point you have to decide if you want only to save the work that you have done using the Query Editor, or if you want not only to save your work, but also update the data in the data model with the latest version of the data that results from the data transformation that you have carried out.

Consequently, you have to choose between:

- *Apply*: Apply the changes and load the data to the data model.
- *Apply Later*: Save your modifications but leave the data as it is currently.

Exiting the Query Editor

In a similar vein to the Save options just described, you can choose how to exit the Query Editor and return to your reports and dashboards in Power BI Desktop. The default option (when you click the Close & Apply button) is to apply all the changes that you have made to the data, update the data model with the new data, and return to Power BI Desktop.

However, you have two other options that may prove useful. These appear in the menu for the Close & Apply button:

- *Apply*: Apply the changes that you have made to the data model. This may involve changes to the fields in the Fields list.
- *Close*: Close the Query Editor but do not apply any changes.

If you choose not to apply the changes that you have made, then you will see the alert that is displayed in Figure 6-26. At some point you will have to update the data model to ensure that you are using the correct data for your reports by clicking the Apply Changes button in the alert that appears above the report canvas.

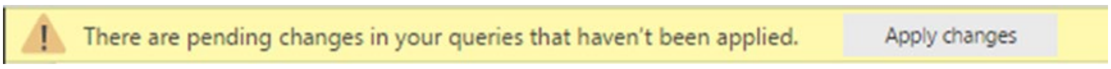

There are pending changes in your queries that haven't been applied.

Figure 6-26. *The pending changes alert*

Conclusion

This chapter started you on the road to transforming datasets with Power BI Desktop. You saw how to trim datasets by removing rows and columns. You also learned how to subset a sample of data from a data source by selecting alternating groups of rows.

You also saw how to choose the columns that you want to use in reports, how to move columns around in the dataset, and how to rename columns so that your data is easily comprehensible when you use it later in dashboards and reports. Then, you saw how to filter and sort data, as well as how to remove duplicates to ensure that your dataset only contains the precise rows that you need for your upcoming visualizations. Finally, you learned how to group and aggregate data.

It has to be admitted, nonetheless, that preparing raw data for use in dashboards and reports is not always easy and can take a while to get right. However, Power BI Desktop Query can make this task really easy with a little practice. So now that you have grasped the basics, it is time to move on and discover some further data transformation techniques. Specifically, you will see how to transform and potentially cleanse the data that you have imported. This is the subject of the next chapter.

CHAPTER 7

Data Transformation

Once a dataset has been shaped and filtered (as covered in the previous chapter), it probably still needs a good few modifications to make it ready for consumption. Many of these modifications are, at their heart, a selection of fairly simple yet necessary techniques that you apply to make the data cleaner and more standardized. I have chosen to group these approaches under the heading *data transformation*.

The sort of things that you may be looking to do before finally loading source data into the data model normally cover a range of processes that *cleanse* the data. They can include the following:

- Change the data type for a column—by telling Power BI Desktop that the column contains numbers, for example
- Ensure that the first row is used as headers (if this is required)
- Remove part of a column's contents
- Replace the values in a cell with other values
- Transform the column contents—by making the text uppercase, for instance, or by removing decimals from numbers
- Fill data down or up over empty cells to ensure that records are complete
- Apply math or statistical (or even trigonometric) functions to columns of numbers
- Convert date or time data into date elements such as days, months, quarters, years, hours, or minutes

Transforming data does not only consist of reducing it. Sometimes you may have to *extend* the data to make it useable. This normally means adding further columns to a data table. The techniques to do this include

- Duplicating column and possibly altering the format of the data in the copied column
- Extracting part of the data in a column into a new column
- Separating all the data in a column so that each data element appears in a separate column
- Merging columns into a new column
- Adding custom columns that possibly contain calculations or extract part of a column's data into a new column, or even concatenate columns
- Adding "index" columns to ensure uniqueness or memorize a sort order

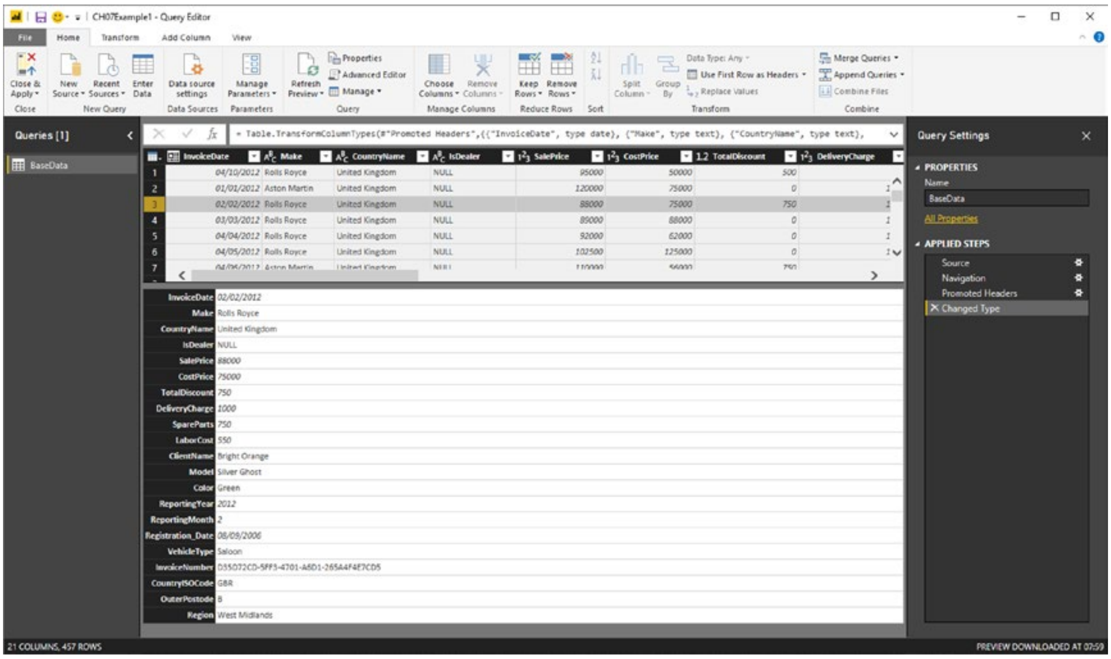
This chapter will take you on a tour of these kinds of essential data transformations. Once you have finished reading it, you should be confident that you can take a rough and ready data source as a starting point and convert it into a polished and coherent data table that is ready to become a pivotal part of your Power BI Desktop data model. Not only that, but you will have carried out really heavy lifting much faster and more easily than you could have done using enterprise-level tools.

The sample data that you will need to follow the exercises in this chapter is in the folder C:\PowerBiDesktopSamples\CH07.

Viewing a Full Record

Before even starting to cleanse data, you probably need to take a good look at it. While the Power BI Desktop Query Editor is great for scrolling up and down columns to see how data compares for a single field, it is often less easy to appreciate the entire contents of a single record.

So to avoid having to scroll frenetically left and right across rows of data, the Query Editor has another brilliantly simple solution. If you click a row (or more specifically, on the number of a row in the grid on the left), the Power BI Desktop Query Editor will display the contents of an entire record in a single window under the dataset. You can see an example of this in Figure 7-1.



```
= Table.TransformColumnTypes(#"Promoted Headers",{{"InvoiceDate", type date}, {"Make", type text}, {"CountryName", type text},
```

	InvoiceDate	Make	CountryName	IsDealer	SalePrice	CostPrice	TotalDiscount	DeliveryCharge
1	04/10/2012	Rolls Royce	United Kingdom	NULL	95000	50000	500	
2	01/01/2012	Aston Martin	United Kingdom	NULL	120000	75000	0	1
3	02/02/2012	Rolls Royce	United Kingdom	NULL	88000	75000	750	1
4	03/03/2012	Rolls Royce	United Kingdom	NULL	89000	88000	0	1
5	04/04/2012	Rolls Royce	United Kingdom	NULL	92000	62000	0	1
6	04/05/2012	Rolls Royce	United Kingdom	NULL	102500	125000	0	1
7	04/06/2012	Aston Martin	United Kingdom	NULL	110000	56000	750	

InvoiceDate	02/02/2012
Make	Rolls Royce
CountryName	United Kingdom
IsDealer	NULL
SalePrice	88000
CostPrice	75000
TotalDiscount	750
DeliveryCharge	1000
SpareParts	750
LaborCost	550
ClientName	Bright Orange
Model	Silver Ghost
Color	Green
ReportingYear	2012
ReportingMonth	2
Registration_Date	08/09/2006
VehicleType	Saloon
InvoiceNumber	D35072CD-5FF3-4701-A8D1-265A4F4E7CD5
CountryISOCode	GBR
OuterPostode	B
Region	West Midlands

Figure 7-1. *Viewing a full record*

■ **Note** You can alter the relative height of the recordset and dataset windows simply by dragging the gray separator line between the upper and lower windows up or down.

Power BI Desktop Query Editor Context Menus

As is normal for Windows programs, Power BI Desktop Query Editor makes full use of context (or "right-click") menus as an alternative to using the ribbons. When transforming datasets, there are three main context menus that you will probably find yourself using:

- *Table menu*: This menu appears when you right-click the top corner of the grid containing the data.
- *Column menu*: This menu appears when you right-click a column title.
- *Cell menu*: This menu appears when you right-click a data cell.

While I have referred copiously to the context menus when explaining how to transform data, it is probably easier to take a quick look at them now so that you can see the various options. Figure 7-2 gives you a quick overview of these three context menus.

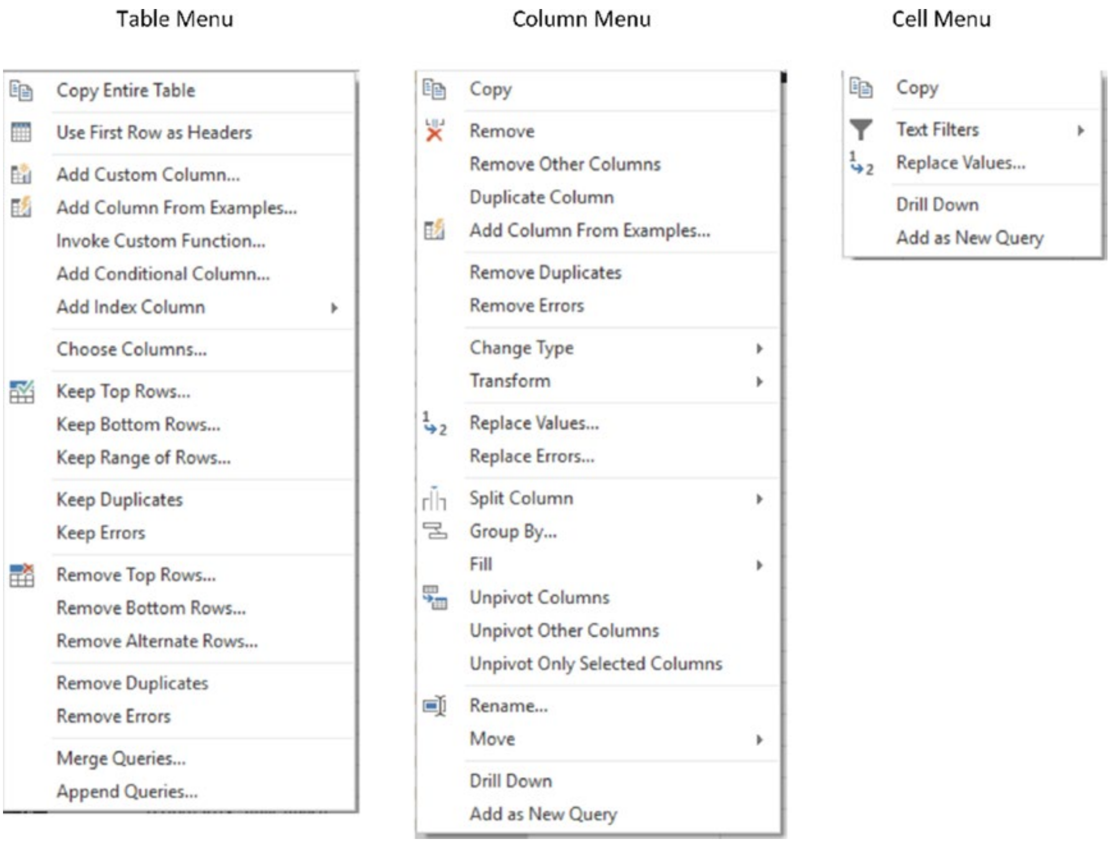


Figure 7-2. *The Power BI Desktop Query Editor context menus*

Because the options that are available in the context menus are explained throughout this, the previous, and the following chapters, I will not explain them all in detail here.

Using the First Row As Headers

Power BI Desktop is very good at guessing if it needs to take the first record of a source dataset and have it function as the column headers. This is fundamental for two reasons:

- You avoid leaving the columns named Column1, Column2, and so on. Leaving them named generically like this would make it needlessly difficult for a user (or even yourself) to understand the data.
- You avoid having a text element (which should be the column title) in a column of figures, which can cause problems later on. This is because a whole column needs to have the same data type for another data type to be applied. Having a header text in the first row prevents this for numeric and data/time data types, for instance. This could be because the header is a text whereas the remainder of the column contains numbers or dates.

Yet there could be—albeit rare—occasions when Power BI Desktop guesses incorrectly and assumes that the first record in a dataset is data when it is really the header information. So instead of headers, you have a set of generic column titles such as Column1, Column2, and so forth. Fortunately, correcting this and using the first row as headers is a simple task:

Click Use First Row As Headers in the Transform ribbon of the Power BI Desktop Query window.

After a few seconds, the first record disappears and the column titles become the elements that were in the first record. The Applied Steps list on the right now contains a Promoted Headers element, indicating which process has taken place. This step is highlighted.

■ **Note** Power BI Desktop is often able to apply this step automatically when the source is a database. It can often correctly guess when the source is a file. However, it cannot always guess accurately, so sometimes you have to intervene. You can see if Power BI Desktop has had to guess this if it has added a Promoted Headers step to the Applied Steps list.

In the rare event that Power BI Desktop gets this operation wrong and presumes that a first row is column titles when it is not, you can reset the titles to be the first row by clicking the tiny triangle to the right of the Use First Row As Headers button. This displays a short menu where you can click the Use Headers As First Row option. The Applied Steps list on the right now contains a Demoted Headers element and the column titles are Column1, Column2, and so forth. You can subsequently rename the columns as you see fit.

Changing Data Type

A truly fundamental aspect of data modification is ensuring that the data is of the appropriate type; that is, if you have a column of numbers that are to be calculated at some point, then the column should be a numeric column. If it contains dates, then it should be set to one of the date or time data types. I realize that this can seem arduous and even superfluous; however, *if you want to be sure that your data can be sliced and diced correctly further down the line*, then setting the right data types at the outset is *vital*. An added bonus is that if you validate the data types early on in the process of loading data, you can see from the start if the data has any potential issues—dates that cannot be read as dates, for instance. This allows you to decide what to do with poor or unreliable data early in your work with a dataset.

The good news here is that for many data sources, Power BI Desktop applies an appropriate data type. Specifically, if you have loaded data from a database, then Power BI Desktop will recognize the data type for each column and apply a suitable native data type. Unfortunately, things can get a little more painful with file sources, specifically CSV, text, and (occasionally) Excel files, as well as some XML files. In the case of these file types, Power BI Desktop often tries to guess the data type, but there are times when it does not succeed. If it has made a stab at deducing data types, then you see a Changed Type step in the Applied Steps list. Consequently, if you are obtaining your data from these sources, then you could well be obliged to apply data types to many of the columns manually.

■ **Note** In some cases, numbers are not meant to be interpreted as numerical data. For instance, a French postal code is five numbers, but it will never be calculated in any way. So it is good practice to let Power BI Desktop know this by changing the data type to Text.

Do the following to change data type for a column or a group of columns:

1. Open the sample file C:\PowerBiDesktopSamples\CH07\CH07Example1.pbix.
2. Click the Edit Queries button in the Home ribbon. The Query Editor will open.
3. Click inside the column whose data type you wish to change. If you want to modify several columns, then Ctrl-click the requisite column titles. In this example, you could select the CostPrice and TotalDiscount columns.
4. Click the Data Type button in the Transform ribbon. A pop-up menu of potential data types will appear.
5. Select an appropriate data type. If you have selected the CostPrice and TotalDiscount columns, then Whole Number is the type to choose.

After a few seconds, the data type will be applied. Changed Type will appear in the Applied Steps list. The data types that you can apply are outlined in Table 7-1.

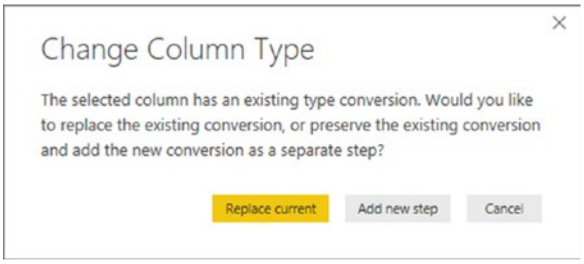
Table 7-1. *Data Types in Power BI Desktop*

Data Type	Description
Decimal Number	Converts the data to a decimal number.
Fixed Decimal Number	Converts the data to a decimal number with a fixed number of decimals.
Whole Number	Converts the data to a whole (integer) number.
Date/Time	Converts to a date and time data type.
Date	Converts to a date data type.
Time	Converts to a time data type.
Duration	Sets the data as being a duration. These are used for date and time calculations.
Text	Sets to a text data type.
True/False	Sets the data type to Boolean (True or False).
Binary	Defines the data as binary, and consequently, it is not directly visible.

■ **Note** The Data Type button is also available in the Home ribbon. Equally, you can right-click a column header and select Change Type to select a different data type.

Inevitably, there will be times when you try to apply a data type that simply cannot be used with a certain column of data. Converting a text column (such as Make in this sample data table) into dates will simply not work. If you do this, then Power BI Desktop will replace the column contents with Error. This is not definitive or dangerous, and all you have to do to return the data to its previous state is to delete the Changed Type step in the Applied Steps list using the technique described in the previous chapter.

Sometimes you could try and change a data type when the data type has already been changed. In this case you will get an alert like the one shown in Figure 7-3.


Change Column Type

The selected column has an existing type conversion. Would you like to replace the existing conversion, or preserve the existing conversion and add the new conversion as a separate step?

Figure 7-3. *The Change Column Type alert*

If this occurs, you can do one of two things:

- Let Power BI Desktop update the existing conversion step with the data type that you just selected.
- Add a new conversion step.

Your choice will depend on exactly what type of transformation you are applying to the underlying dataset.

It can help to alter data types at the same time for a *set* of columns where you think that this operation is necessary. There are a couple of good reasons for this approach:

- You can concentrate on getting data types right, and if you are working methodically, you are less likely to forget to set a data type.
- Applying data types for many columns (even if you are doing this in several operations, to single or multiple columns) will only add a single step to the Applied Steps list.

■ **Note** Don't look for any data formatting options in Power BI Desktop Query; there aren't any. This is deliberate since this tool is designed to structure, load, and cleanse data, but not to present it. You carry out the formatting in the Power BI Desktop Data View, as you will see in Chapter 10.

Detecting Data Types

Applying the correct data type to dozens of columns can be more than a little time-consuming. Fortunately, Power BI Desktop now contains an option to apply data types automatically to a whole table:

1. In the Transform ribbon, click the Detect Data Type button.

2. Changed Type will appear in the Applied Steps list. Most of the columns will have the correct data type applied.

This technique does not always give perfect results, and there will be times when you want to override the choice of data type that Power BI Desktop has applied. Yet it is nonetheless a welcome addition to the data preparation toolset that can save you considerable time when preparing a dataset.

Data Type Indicators

It would be singularly unproductive to have to guess which column was set to which data type. So Power BI Desktop comes to your aid by indicating, visually, the corresponding data type for each column. If you look closely to the left of each individual column header, you will see a tiny icon. Each icon specifies the column's data type. The meaning of each icon is given in Table 7-2.

Table 7-2. *Data Type Icons in Power BI Desktop Query Editor*

Data Type Icon	Description
ABC 123	Any data type from among the possible data types
123	Whole Number
1.2	Decimal Number
$	Fixed Decimal Number
%	Percentage
ABC	Text
✓	True/False
📅🕒	Date/Time
📅	Date
🕒	Time
🌐	Date/Time/Timezone
⏱	Duration
▤	Binary

Switching Data Types

Another quick way to alter the data type for a column is to click the data type icon to the left of the column title and select the required data type from the context menu that you can see in Figure 7-4.

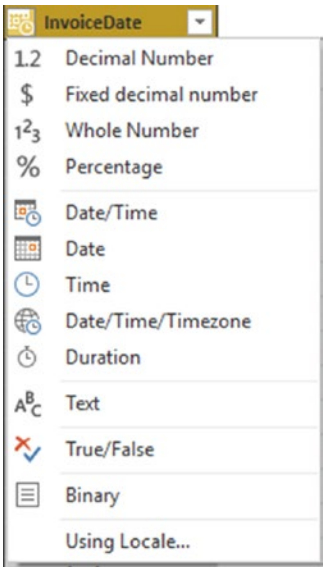
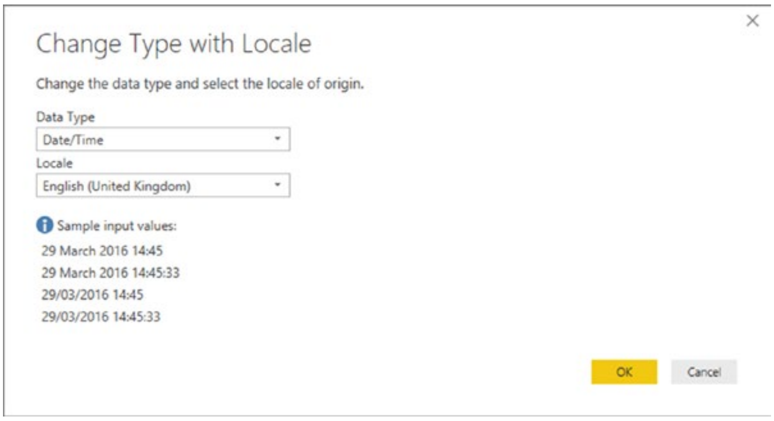


Figure 7-4. *The data type context menu*

Data Type Using Locale

When you are converting data types, you can also choose to use the current locale to specify date, time, and number formats. This means that users opening the Power BI Desktop file in another country will see date, time, and number formats adapted to the local formatting conventions. To do this:

1. Open the Query Editor (unless it is already open).
2. Click the data type icon to the left of the column title.
3. Select Using Locale from the popup menu. The Change Type with Locale dialog will appear.
4. Choose the new data type to apply from the list of available data types.
5. Select the required locale from the list of worldwide locales. The dialog will look like Figure 7-5.


Change Type with Locale

Change the data type and select the locale of origin.

Data Type: Date/Time

Locale: English (United Kingdom)

Sample input values:
29 March 2016 14:45
29 March 2016 14:45:33
29/03/2016 14:45
29/03/2016 14:45:33

Figure 7-5. *The Change Type with Locale dialog*

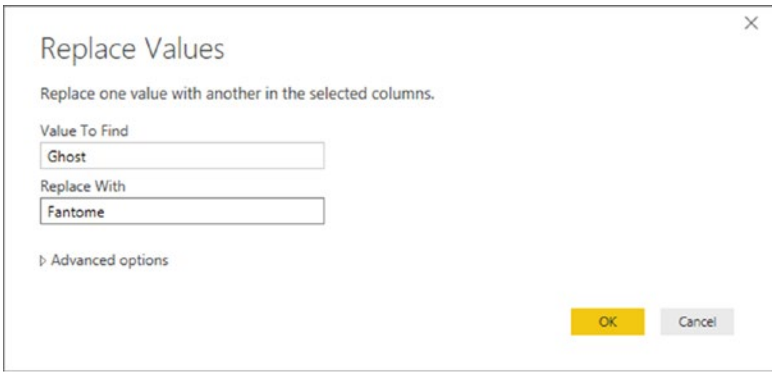
6. Click OK.

The data type will be converted to the selected locale. The Applied Steps list will contain a step entitled Changed Type with Locale.

Replacing Values

Some data that you load will need certain values to be replaced by others in a kind of global search-and-replace operation—just as you would in a document. For instance, perhaps you need to standardize spellings where a make of car (to use the current sample dataset as an example) has been entered incorrectly. To carry out this particular data cleansing operation, do the following:

1. Click the title of the column that contains the data that you want to replace. The column will become selected. In this example, I used the Model column as an example.
2. In the Home ribbon, click the Replace Values button. The Replace Values dialog will appear.
3. In the Value To Find box, enter the text or number that you want to replace. I used Ghost in this example.
4. In the Replace With box, enter the text or number that you want to replace. I used Fantôme in this example, as shown in Figure 7-6.


Replace Values

Replace one value with another in the selected columns.

Value To Find: Ghost

Replace With: Fantome

Figure 7-6. *The Replace Values dialog*

5. Click OK. The data is replaced in the entire column. Replaced Values is added to the Applied Steps list.

I only have a few comments about this technique:

- The Replace Values process searches for every occurrence of the text that you are looking for in each record of the selected columns. It does not look for the entire contents of the cell unless you specifically request this by checking the Match Entire Cell Contents check box in the advanced options.
- If you click a cell containing the contents that you want to replace (rather than the column title, as we just did), before starting the process, Power BI Desktop automatically places the cell contents in the Replace Values dialog as the value to find.
- You can only replace text in columns that contain text elements. This does not work with columns that are set as a numeric or date data type.
- If you really have to replace parts of a date or figures in a numeric column with other dates or numbers, then you can
  - Convert the column to a text data type
  - Carry out the replace operation
  - Convert the column back to the original data type

The Replace Values dialog also has a few advanced options that you can apply. You can see these if you expand the “Advanced options” item by clicking the triangle to its left. These options are explained in Table 7-3.

Table 7-3. *Advanced Replace Options*

Option	Description
Match Entire Cell Contents	Only replaces the search value if it makes up the entire contents of the column for a row.
Replace Using Special Characters	Replaces the search value with a nonprinting character.
Tab	Replaces the search value with a tab character.
Carriage Return	Replaces the search value with a carriage return character.
Line Feed	Replaces the search value with a line feed character.
Carriage Return and Line Feed	Replaces the search value with a carriage return and line feed.

■ **Note** Replacing words that are subsets of other words are dangerous. When replacing any data, make sure that you don't damage elements other than the one you intend to change.

As a final and purely spurious comment, I must add that I would never suggest rebranding a Rolls-Royce, as it would be close to automotive sacrilege.

Transforming Column Contents

Power BI Desktop has a powerful toolbox of automated data transformations that allow you to standardize the contents of a column in several ways. These include

- Setting the capitalization of text columns
- Rounding numeric data or applying math functions
- Extracting date elements such as the year, month, or day (among others) from a date column

Power BI Desktop is very strict about applying transformations to appropriate types of data. This is because transforms are totally dependent on the data type of the selected column. This is yet another confirmation that applying the requisite data type is an operation that should be carried out early in any data transformation process—and certainly *before* transforming the column contents. Remember, you will only be able to select a numeric transformation if the column is a numeric data type, and you will only be able to select a date transformation if the column is a date data type. Equally, the text-based transformations can only be applied to columns that are of the text data type.

Text Transformation

Let's look at a simple transformation operation in action. As an example, I will get Power BI Desktop to convert the Make column into uppercase characters.

1. Still using the file CH07Example1.pbix, click anywhere in the column whose contents you wish to transform (Make, in this case).
2. In the Transform ribbon, click the Format button. A popup menu will appear.
3. Select UPPERCASE, as shown in Figure 7-7.

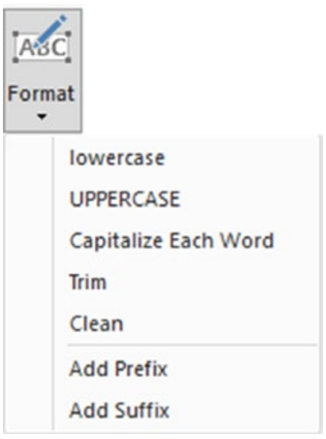


***Figure 7-7.** The Format menu*

The contents of the entire column will be converted to uppercase. Uppercased Text will be added to the Applied Steps list.

As you can see from the menu for the Format button, you have five possible options when formatting (or transforming) text. These options are explained in Table 7-4.

***Table 7-4.** Text Transformations*

Transformation	Description	Applied Steps Definition
Lowercase	Converts all the text to lowercase.	Lowercased Text
Uppercase	Converts all the text to uppercase.	Uppercased Text
Capitalize Each Word	Converts the first letter of each word to a capital.	Capitalized Each Word
Trim	Removes all spaces before and after the text.	Trimmed Text
Clean	Removes any nonprintable characters.	Cleaned Text
Add Prefix	Adds text at the start of the column contents.	Added Prefix
Add Suffix	Adds text at the end of the column contents.	Added Suffix

■ **Note** I realize that Power BI Desktop Query calls text transformations Formatting. Nonetheless, these options are part of the overall data transformation options.

Adding a Prefix or a Suffix

You can also add a prefix or a suffix to all the data in a column. This is as easy as:

1. Click inside the column where you want to add a prefix.
2. In the Transform ribbon, select Format ➤ Add Prefix. The Prefix dialog will be displayed, as you can see in Figure 7-8.

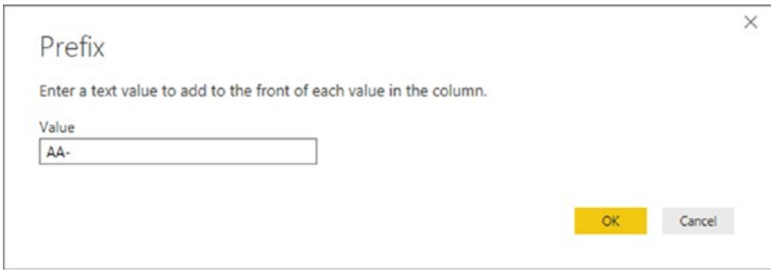

Prefix
Enter a text value to add to the front of each value in the column.
Value
AA-
OK
Cancel

Figure 7-8. *Adding a prefix to a text*

3. Enter the prefix to add in the Value field.
4. Click OK.

The prefix that you designated will be placed at the start of every record in the dataset for the selected field.

■ **Note** If you add a prefix or a suffix to a numeric or date/time column, then the column data type will automatically be converted to text.

Removing Leading and Trailing Spaces

There will inevitably be occasions when you inherit data that has extra spaces before, after, or before *and* after the data itself. This can be insidious, as it can cause

- Data duplication, because a value with a trailing space is *not* considered identical to the same text without the spaces that follow
- Sort issues, because a leading space causes an element to appear at the *top* of a sorted list
- Grouping errors, because elements with spaces are not part of the same group as elements without spaces

Fortunately, Power BI Desktop Query has a ruthlessly efficient solution to this problem.

1. Click anywhere in the column whose contents you wish to transform (Make, in this case).
2. In the Transform ribbon, click the Format button. A popup menu will appear.
3. Select Trim from the menu.

All superfluous leading and trailing spaces will be removed from the data in the column. This should help with sorting, grouping, and deduplicating records.

Removing Non-Printing Characters

Some source data can contain somewhat insidious elements called non-printing characters. These can, even if they are nearly always invisible to humans, cause problems when you print reports and dashboards.

If you suspect that your source data contains non-printing characters, you can remove them simply like this:

1. Click inside the column (or select the columns) that you know to contain (or that you suspect contain) non-printing characters.
2. Click Format ➤ Clean.

Power BI Desktop will add Cleaned Text to the list of Applied Steps.

Number Transformations

Just as you can transform the contents of text-based columns, you can also apply transformations to numeric values. As an example, suppose that you want to round up all the figures in a column to the nearest whole number.

1. Click anywhere in the column whose contents you wish to transform (TotalDiscount, in this case).
2. In the Transform ribbon, click the Rounding button. A popup menu will appear.
3. Select Round Up.

The values in the entire column will be rounded up to the nearest whole number. Rounded Up will be added to the Applied Steps list.

The other possible numeric transformations that are available are described in Table 7-5. Because these numeric transformations use several buttons in the Transform ribbon, I have indicated which button to use to get the desired result.

Table 7-5. Number Transformations

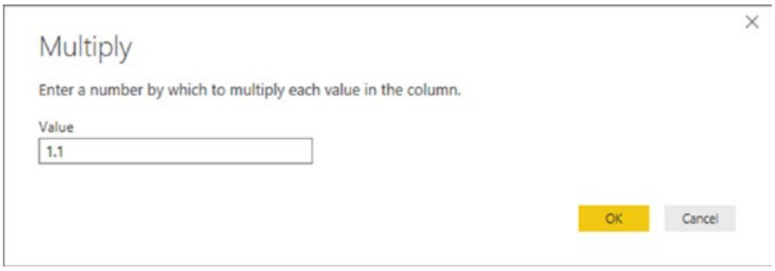
Transformation	Description	Applied Steps Definition
Rounding ➤ Round Up	Rounds each number to the specified number of decimal places.	Rounded Up
Rounding ➤ Round Down	Rounds each number down.	Rounded Down
Round...	Rounds each number to the number of decimals that you specify. If you specify a negative number, you round to a given decimal.	Rounded Off
Scientific ➤ Absolute Value	Makes the number absolute (positive).	
Scientific ➤ Power ➤ Square	Returns the square of the number in each cell.	Calculated Square
Scientific ➤ Power ➤ Cube	Returns the cube of the number in each cell.	Calculated Cube Value
Scientific ➤ Power ➤ Power	Raises each number to the power that you specify.	Calculated Power
Scientific ➤ Square Root	Returns the square root of the number in each cell.	Square Root
Scientific ➤ Exponent	Returns the exponent of the number in each cell.	Calculated Exponent
Scientific ➤ Logarithm ➤ Base 10	Returns the base 10 logarithm of the number in each cell.	Calculated Base 10 Logarithm
Scientific ➤ Logarithm ➤ Natural	Returns the natural logarithm of the number in each cell.	Calculated Natural Logarithm
Scientific ➤ Factorial	Gives the factorial of numbers in the column.	Calculated Factorial
Trigonometry ➤ Sine	Gives the sine of the numbers in the column.	Calculated Sine
Trigonometry ➤ Cosine	Gives the cosine of the numbers in the column.	Calculated Cosine
Trigonometry ➤Tangent	Gives the tangent of the numbers in the column.	Calculated Tangent
Trigonometry ➤ ArcSine	Gives the arcsine of the numbers in the column.	Calculated ArcSine
Trigonometry ➤ ArcCosine	Gives the arccosine of the numbers in the column.	Calculated ArcCosine
Trigonometry ➤ ArcTangent	Gives the arctangent of the numbers in the column.	Calculated ArcTangent

■ **Note** Power BI Desktop Query will not even let you try to apply numeric transformation to texts or dates. The relevant buttons remain grayed out if you click inside a column of letters or dates.

Calculating Numbers

Power BI Desktop Query can also apply simple arithmetic to the figures in a column. Suppose, for instance, that you want to multiply all the sale prices by 110% as part of your forecasts. This is how you can do just that:

1. Click inside any column of numbers. In this example, I used the column SalePrice.
2. Click the Standard button in the Transform ribbon. The menu will appear.
3. Click Multiply. The Multiply dialog will appear.
4. Enter **1.1** in the Value box. The dialog will look like the one shown in Figure 7-9.


Multiply

Enter a number by which to multiply each value in the column.

Value: 1.1

Figure 7-9. *Applying a calculation to a column*

5. Click OK.

All the numbers in the selected column will be multiplied by 1.1. In other words, they are now 110% of the original value. Table 7-6 describes the possible math operations that you can carry out in Power BI Desktop Query.

Table 7-6. *Applying Basic Calculations*

Transformation	Description	Applied Steps Definition
Add	Adds a selected value to the numbers in a column.	Added to Column
Multiply	Multiplies the numbers in a column by a selected value.	Multiplied Column
Subtract	Subtracts a selected value from the numbers in a column.	Subtracted from Column
Divide	Divides the numbers in a column by a selected value.	Divided Column
Integer-Divide	Divides the numbers in a column by a selected value and removes any remainder.	Integer-Divided Column
Modulo	Divides the numbers in a column by a selected value and leaves only the remainder.	Calculated Modulo
Percentage	Applies the selected percentage to the column.	Calculated Percentage
Percent Of	Expresses the value in the column as a percent of the value that you enter.	Calculated Percent Of

---

■ **Note** You can also carry out many types of calculations in Power BI Desktop Data View and avoid carrying out calculations in the Query Editor. Indeed, many Power BI Desktop purists seem to prefer that anything resembling a calculation should take place inside the data model rather than at the Query stage. As ever, I will let you decide which approach you prefer. Yet I would advise you to read Chapters 11 through 13 to get a clearer understanding of how to add calculated elements to Power BI Desktop using DAX. This is because some transformations need to adjust to the situation (the context) in which they are used, and consequently need to be done using DAX. Be aware that some heavy transforms can slow the reports down if calculated at run time, whereas others can only be effective as part of a well-thought-out calculation process.

Finally, it is important to remember that you are altering the data when you carry out this kind of operation. In the real world, you might be safer duplicating a column before profoundly altering the data it contains. This allows you to keep the initial data available, albeit at the cost of increasing both the load time and the size of the Power BI Desktop file.

Date Transformations

Transforming dates follows similar principles to transforming text and numbers. As an example, here is how to isolate the month from a date:

1. Click inside the InvoiceDate column.
2. In the Transform ribbon, click the Date button. The menu will appear.
3. Click Year. The submenu will appear.
4. Select Year. The year part of the date will replace all the dates in the InvoiceDate column.

The other possible date transformations that are possible are given in Table 7-7.

Table 7-7. *Date Transformations*

Transformation	Description	Applied Steps Definition
Age	Calculates the date and time difference (in days and hours) between the original date and the current local time.	Calculated Age
Date Only	Converts the data to a date without the time element.	Calculated Date
Year ➤ Year	Extracts the year from the date.	Calculated Year
Year ➤ Start of Year	Returns the first day of the year for the date.	Calculated Start of Year
Year ➤ End of Year	Returns the last day of the year for the date.	Calculated End of Year
Month ➤ Month	Extracts the number of the month from the date.	Calculated Month
Month ➤ Start of Month	Returns the first day of the month for the date.	Calculated Start of Month
Month ➤ End of Month	Returns the last day of the month for the date.	Calculated End of Month
Month ➤ Days in Month	Returns the number of days in the month for the date.	Calculated Days in Month
Month ➤ Name of Month	Returns the name of the month for the date.	Calculated Name of Month
Day ➤ Day	Extracts the day from the date.	Calculated Day
Day ➤ Day of Week	Returns the weekday as a number (Monday is 1, Tuesday is 2, etc.).	Calculated Day of Week
Day ➤ Day of Year	Calculates the number of days since the start of the year for the date.	Calculated Day of Year
Day ➤ Start of Day	Transforms the value to the start of the day for a date and time.	Calculated Start of Day
Day ➤ End of Day	Transforms the value to the end of the day for a date and time.	Calculated End of Day
Day ➤ Name of Day	Returns the weekday as a day of week.	Calculated Name of Day
Quarter ➤ Quarter	Returns the calendar quarter of the year for the date.	Calculated Quarter
Quarter ➤ Start of Quarter	Returns the first date of the calendar quarter of the year for the date.	Calculated Start of Quarter
Quarter ➤ End of Quarter	Returns the last date of the calendar quarter of the year for the date.	Calculated End of Quarter
Week ➤ Week of Year	Calculates the number of weeks since the start of the year for the date.	Calculated Week of Year
Week ➤ Week of Month	Calculates the number of weeks since the start of the month for the date.	Calculated Week of Month
Week ➤ Start of Week	Returns the date for the first day of the week (Monday) for the date.	Calculated Start of Week
Week ➤ End of Week	Returns the date for the last day of the week (Monday) for the date.	Calculated End of Week

Time Transformations

You can also transform date/time or time values into their component parts using Power BI Desktop Query. This is extremely similar to how you apply date transformations, but in the interest of completeness, the following explains how to do this:

1. Click inside the InvoiceDate column.
2. In the Transform ribbon, click the Time button. The menu will appear.
3. Click Hour. The hour part of the time will replace all the values in the InvoiceDate column.

■ **Note** Time transformations can only be applied to columns of the date/time or time data types.

The range of Time transformations is given in Table 7-8.

Table 7-8. *Time Transformations*

Transformation	Description	Applied Steps Definition
Time Only	Isolates the time part of a date and time.	Extracted Time
Local Time	Converts the date/time to local time from date/time and timezone values.	Extracted Local Time
Parse	Extracts the date and/or date/time elements from a text.	Parsed DateTime
Hour ➤ Hour	Isolates the hour from a date/time or date value.	Extracted Hour
Hour ➤ Start of Hour	Returns the start of the hour from a date/time or time value.	Calculated Start of Hour
Hour ➤ End of Hour	Returns the end of the hour from a date/time or time value.	Calculated End of Hour
Minute	Isolates the minute from a date/time or time value.	Extracted Minute
Second	Isolates the second from a date/time or time value.	Extracted Second
Earliest	Returns the earliest time from a date/time or time value.	Calculated Earliest
Latest	Returns the latest time from a date/time or time value.	Calculated Latest

■ **Note** In the real world, you could well want to leave a source column intact and apply number or date transformations to a copy of the column. To do this, simply apply the same transformation technique, only use the buttons in the Add Column ribbon instead of those in the Transform ribbon.

Duration

If you have values in a column that can be interpreted as a duration (in days, hours, minutes, and seconds), then Power BI Desktop Query can extract the component parts of the duration as a data transformation. For this to work, however, the column *must* be set to the duration data type. This means that the contents of the column have to be interpreted as a duration by Power BI Desktop. Any values that are incompatible with this data type will be set to error values.

If you have duration data, you can extract its component parts like this:

1. Click inside the column.
2. In the Transform ribbon, click the Duration button. The menu will appear.
3. Click Hour. The hour part of the time will replace all the values in the InvoiceDate column.

The range of duration transformations is given in Table 7-9.

Table 7-9. *Duration Transformations*

Transformation	Description	Applied Steps Definition
Days	Isolates the day element from a duration value.	Extracted Days
Hours	Isolates the hour element from a duration value.	Extracted Hours
Minutes	Isolates the minutes element from a duration value.	Extracted Minutes
Seconds	Isolates the seconds element from a duration value.	Extracted Seconds
Total Days	Displays the duration value as the number of days and a fraction representing hours, minutes, and seconds.	Calculated Total Days
Total Hours	Displays the duration value as the number of hours and a fraction representing minutes and seconds.	Calculated Total Hours
Total Minutes	Displays the duration value as the number of minutes and a fraction representing seconds.	Calculated Total Minutes
Total Seconds	Displays the duration value as the number of seconds and a fraction representing milliseconds.	Calculated Total Seconds
Multiply	Multiplies the duration (and all its component parts) by a value that you enter.	Multiplied Column
Divide	Divides the duration (and all its component parts) by a value that you enter.	Divided Column
Statistics ➤ Sum	Returns the total for all the duration elements in the column.	Calculated Sum
Statistics ➤ Minimum	Returns the minimum value of all the duration elements in the column.	Calculated Minimum

(*continued*)

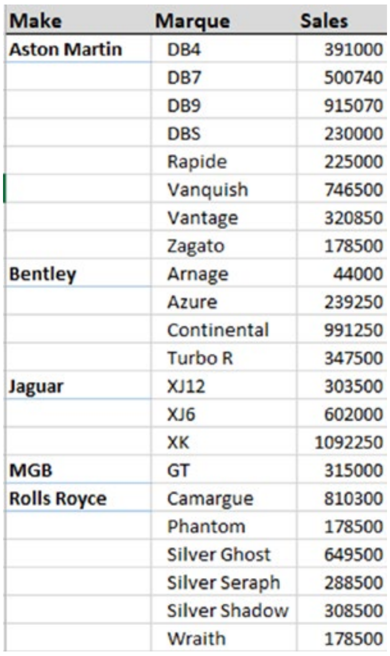
Table 7-9. *(continued)*

Transformation	Description	Applied Steps Definition
Statistics ➤ Maximum	Returns the maximum value of all the duration elements in the column.	Calculated Maximum
Statistics ➤ Median	Returns the median value for all the duration elements in the column.	Calculated Median
Statistics ➤ Average	Returns the average for all the duration elements in the column.	Calculated Average

■ **Note** If you multiply or divide a duration, Power BI Desktop Query displays a dialog so that you can enter the value to multiply or divide the duration by.

Filling Down Empty Cells

Imagine a data source where the data has come into Power BI Desktop from a matrix-style structure. The result is that some columns only contain a single example of an element and then a series of empty cells until the next element in the list. If this is difficult to imagine, then take a look at the sample file CarMakeAndModelMatrix.xlsx shown in Figure 7-10.



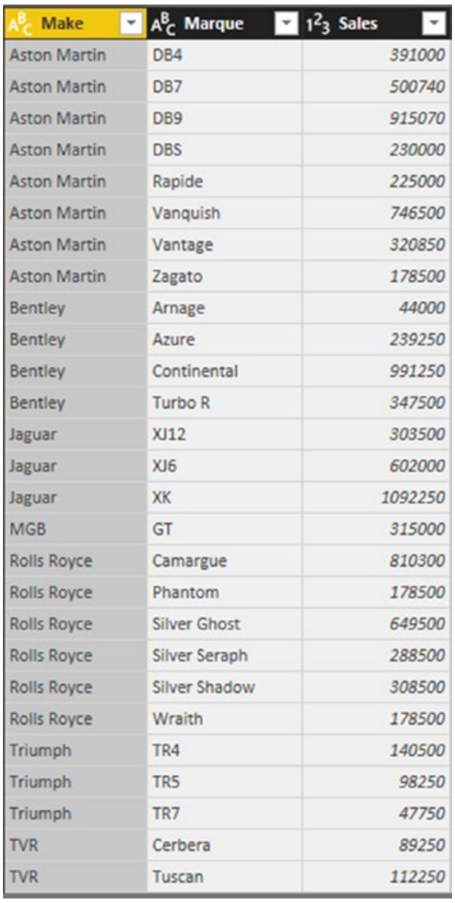
Make	Marque	Sales
Aston Martin	DB4	391000
	DB7	500740
	DB9	915070
	DBS	230000
	Rapide	225000
	Vanquish	746500
	Vantage	320850
	Zagato	178500
Bentley	Arnage	44000
	Azure	239250
	Continental	991250
	Turbo R	347500
Jaguar	XJ12	303500
	XJ6	602000
	XK	1092250
MGB	GT	315000
Rolls Royce	Camargue	810300
	Phantom	178500
	Silver Ghost	649500
	Silver Seraph	288500
	Silver Shadow	308500
	Wraith	178500

Figure 7-10. *A matrix data table in Excel*

All these blank cells are a problem since we need a full data table—or rather, they would be, if Power BI Desktop did not have a really cool way of overcoming this particular difficulty. Do the following to solve this problem:

1. Open a new Power BI Desktop file.
2. In the splash screen, click Get Data.
3. In the Get Data dialog, select Excel. Then Click Connect and navigate to C:\PowerBiDesktopSamples\CH07\CarMakeAndModelMatrix.xlsx.
4. Click Open, select Sheet 1, and click Edit.
5. Click Edit Queries. This will take you directly to the Query Editor.
6. Click in the column that contains the empty cells; make sure that you click where you want to replace the empty cells with the contents of the first non-empty cell above.
7. In the Transform ribbon, click Fill. The menu will appear.
8. Select Down. The blank cells will be replaced by the value in the first non-empty cell above. Filled Down will be added to the Applied Steps list.

The table will now look like Figure 7-11.



Make	Marque	Sales
Aston Martin	DB4	391000
Aston Martin	DB7	500740
Aston Martin	DB9	915070
Aston Martin	DBS	230000
Aston Martin	Rapide	225000
Aston Martin	Vanquish	746500
Aston Martin	Vantage	320850
Aston Martin	Zagato	178500
Bentley	Arnage	44000
Bentley	Azure	239250
Bentley	Continental	991250
Bentley	Turbo R	347500
Jaguar	XJ12	303500
Jaguar	XJ6	602000
Jaguar	XK	1092250
MGB	GT	315000
Rolls Royce	Camargue	810300
Rolls Royce	Phantom	178500
Rolls Royce	Silver Ghost	649500
Rolls Royce	Silver Seraph	288500
Rolls Royce	Silver Shadow	308500
Rolls Royce	Wraith	178500
Triumph	TR4	140500
Triumph	TR5	98250
Triumph	TR7	47750
TVR	Cerbera	89250
TVR	Tuscan	112250

Figure 7-11. *A data table with empty cells replaced by the correct data*

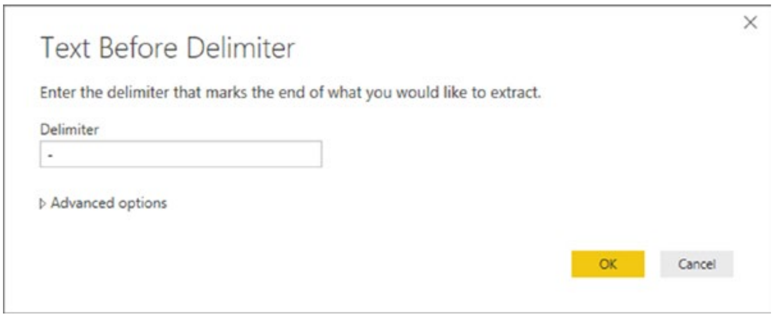
■ **Note** This technique is built to handle a fairly specific problem and only really works if the imported data is grouped by the column containing the missing elements.

Although rare, you can also use this technique to fill empty cells with the value from below. If you need to do this, just select Fill ➤ Up from the Transform ribbon. In either case, you need to be aware that the technique is applied to the entire column.

Extracting Part of a Column's Contents

There could well be times when the contents of a source column contain more data than you actually need. In cases like this, Power BI Desktop can help you by extracting only part of a column. This technique works like this:

1. Load the C:\PowerBiDesktopSamples\CH07\CH07Example1.pbix sample file and click Edit Queries.
2. Click inside the InvoiceNumber column.
3. In the Transform ribbon, click Extract ➤ Text Before Delimiter. The Text Before Delimiter dialog will be displayed.
4. Enter a hyphen (or a minus sign) in the Delimiter field. The dialog will look like Figure 7-12.


Text Before Delimiter

Enter the delimiter that marks the end of what you would like to extract.

Delimiter: -

Figure 7-12. *The Text Before Delimiter dialog*

5. Click OK. The contents of the field will be replaced by the characters before the hyphen. A step named Extracted Text Before Delimiter will be added to the Applied Steps list.

The Extract function allows you to choose from a variety of ways in which you can extract a subset of data from a column. The currently available options are explained in Table 7-10.

Table 7-10. *Extract Transformations*

Transformation	Description	Applied Steps Definition
Length	Displays the length in characters of the contents of the field.	Extracted Length
First Characters	Displays a specified number of characters from the left of the field.	Extracted First Characters
Last Characters	Displays a specified number of characters from the right of the field.	Extracted Last Characters
Range	Displays a specified number of characters between a specified start and end position (in characters, from the left of the field).	Extracted Range
Text Before Delimiter	Displays all the text occurring before a specified character.	Extracted Text Before Delimiter
Text After Delimiter	Displays all the text occurring after a specified character.	Extracted Text After Delimiter
Text Between Delimiters	Displays all the text occurring between two specified characters.	Extracted Text Between Delimiters

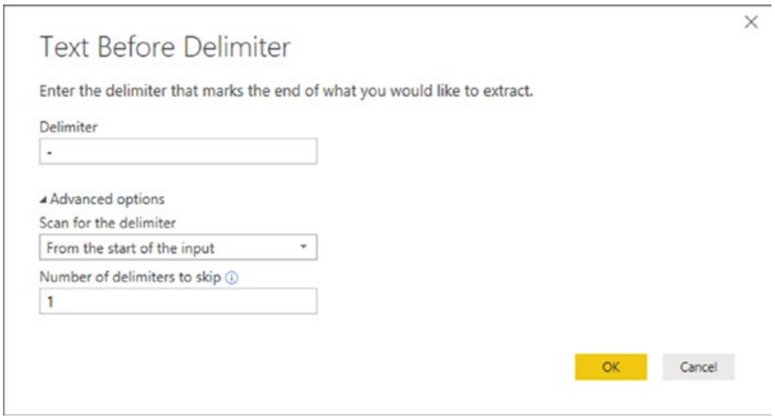
Advanced Extract Options

Three of the Extract options (Text Before Delimiter, Text After Delimiter, and Text Between Delimiters) let you apply some advanced options that allow you to push the envelope even further when extracting data from a column. These techniques are explained in the following two sections.

Text Before and After Delimiter

If you are extracting part of the contents of a column and you are using a delimiter to isolate the text you want to keep, then you have a couple of additional options available.

You can access these options from the dialog that you saw in Figure 7-8 by clicking Advanced Options. The dialog will then look like the one shown in Figure 7-13.


Text Before Delimiter

Enter the delimiter that marks the end of what you would like to extract.

Delimiter: -

Advanced options

Scan for the delimiter: From the start of the input

Number of delimiters to skip: 1

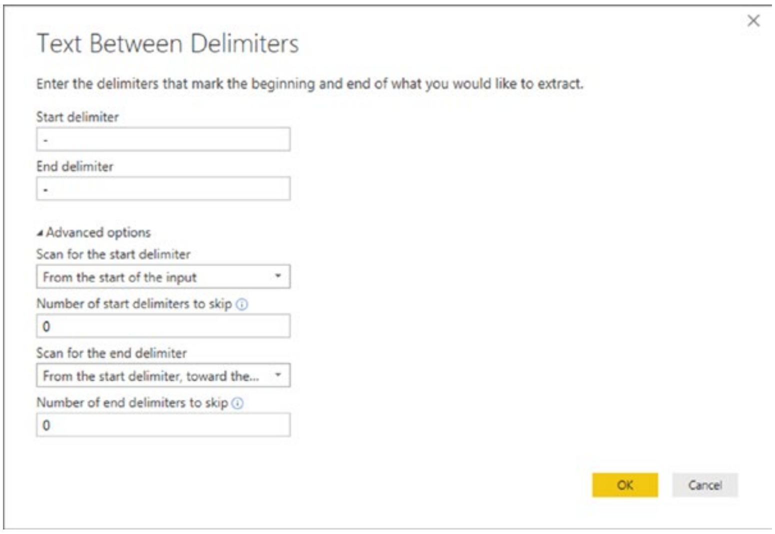
Figure 7-13. *The Advanced Options of the Text Before and Text After Delimiter dialogs*

The two options that you now have are:

- *Scan for the delimiter*: This option lets you choose between working forward from the start of the contents of the column or working backward from the end of the contents of the column to locate the delimiter you are searching for.
- *Number of delimiters to skip*: Here you can specify that it is the *n*th occurrence of a delimiter that interests you.

Text Between Delimiters

The Advanced Options of the Text Between Delimiters dialog essentially lets you apply the same options that you saw previously, only for both the initial delimiter and the final delimiter. In Figure 7-14 you can see this in the Text Between Delimiters dialog.



Text Between Delimiters

Enter the delimiters that mark the beginning and end of what you would like to extract.

Start delimiter: -

End delimiter: -

Advanced options

Scan for the start delimiter: From the start of the input

Number of start delimiters to skip: 0

Scan for the end delimiter: From the start delimiter, toward the...

Number of end delimiters to skip: 0

Figure 7-14. *The Advanced Options of the Text Between Delimiters dialog*

■ **Note** The Extract button can be found in both the Transform and New Column ribbons. If you carry out this operation from the Transform ribbon, then the contents of the existing column will be replaced. If you use the button in the Add Column ribbon, then a new column containing the extracted text will be added at the right of any existing columns.

Duplicating Columns

Sometimes you just need a simple copy of a column, with nothing added and nothing taken away. This is where the Duplicate Column button comes into play.

1. Load the C:\PowerBiDesktopSamples\CH07\CH07Example1.pbix sample file.
2. Open the Power BI Desktop Query Editor.

3. Click inside (or on the title of) the column that you want to duplicate. I will use the Make column in this example.
4. In the Add Column ribbon, click the Duplicate Column button. After a few seconds, a copy of the column is created at the right of the existing table. Duplicated Column will appear in the Applied Steps list.
5. Scroll to the right of the table and rename the existing column; it is currently named Make-Copy.

■ **Note** The duplicate column is named Original Column Name-Copy. I find that it helps to rename copies of columns sooner rather than later in a data mashup process.

Splitting Columns

Sometimes a source column contains data that you really need to break up into smaller pieces across two or more columns. The following are classic cases where this happens:

- A column contains a list of elements, separated by a specific character (known as a *delimiter*).
- A column contains a list of elements, but the elements can be divided at specific places in the column.
- A column contains a concatenated text that needs to be split into its composite elements (a bank account number or a Social Security number are examples of this).

The following short sections explain how to handle such eventualities.

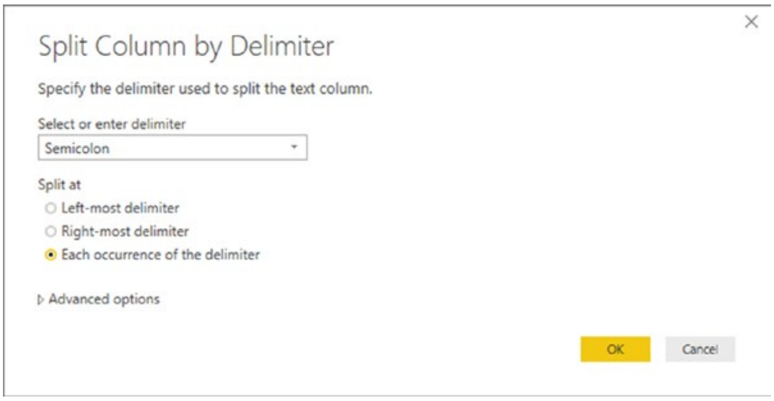
Splitting Column by a Delimiter

Here is another requirement that you may encounter occasionally. The data that has been imported has a column that needs to be further split into multiple columns. Imagine a text file where columns are separated by semicolons, and these subdivisions each contain a column that holds a comma-separated list of elements. Once you have imported the file, you then need to further separate the contents of this column that uses a different delimiter.

Here is what you can do to split the data from one column over several columns:

1. Edit the C:\PowerBiDesktopSamples\CH07\DataToParse.xlsx sample file in the Query Editor.
2. In the Transform ribbon, click Use First Row As Headers.
3. Click inside the ClientList column. You can see that this column contains several data elements, each separated by a semicolon.
4. In the Transform ribbon, click Split Column ➤ By Delimiter. The Split Column by Delimiter dialog appears.
5. Select Semicolon from the list of available options in the “Select or enter delimiter” popup (although the Query Editor could well have detected this already).

6. Click "Each occurrence of the delimiter" as the location to split the text column. The dialog should look like Figure 7-15.



Split Column by Delimiter

Specify the delimiter used to split the text column.

Select or enter delimiter: Semicolon

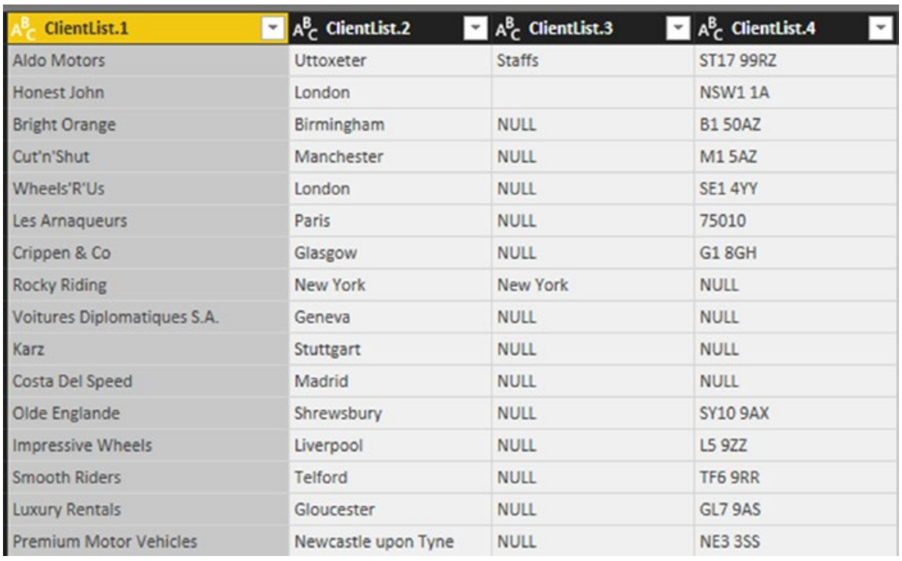
Split at
- ○ Left-most delimiter
- ○ Right-most delimiter
- ● Each occurrence of the delimiter

▷ Advanced options

Figure 7-15. *Splitting a column using a delimiter*

7. Click OK. Split Column by Delimiter will appear in the Applied Steps list.

The initial column is replaced and all the new columns are named InvoiceNumber.1, InvoiceNumber.2, and so forth. As many additional columns as there are delimiters are created; each is named (*Column.n*) and is sequentially numbered. The result of this operation looks like Figure 7-16.



ClientList.1	ClientList.2	ClientList.3	ClientList.4
Aldo Motors	Uttoxeter	Staffs	ST17 99RZ
Honest John	London		NSW1 1A
Bright Orange	Birmingham	NULL	B1 50AZ
Cut'n'Shut	Manchester	NULL	M1 5AZ
Wheels'R'Us	London	NULL	SE1 4YY
Les Arnaqueurs	Paris	NULL	75010
Crippen & Co	Glasgow	NULL	G1 8GH
Rocky Riding	New York	New York	NULL
Voitures Diplomatiques S.A.	Geneva	NULL	NULL
Karz	Stuttgart	NULL	NULL
Costa Del Speed	Madrid	NULL	NULL
Olde Englande	Shrewsbury	NULL	SY10 9AX
Impressive Wheels	Liverpool	NULL	L5 9ZZ
Smooth Riders	Telford	NULL	TF6 9RR
Luxury Rentals	Gloucester	NULL	GL7 9AS
Premium Motor Vehicles	Newcastle upon Tyne	NULL	NE3 3SS

Figure 7-16. *The results of splitting a column*

This particular process has several options, and their consequences can be fairly far-reaching as far as the data is concerned. Table 7-11 contains a description of the available options.

***Table 7-11.** Delimiter Split Options*

Option	Description
Colon	Uses the colon (:) as the delimiter.
Comma	Uses the comma (,) as the delimiter.
Equals Sign	Uses the equals sign (=) as the delimiter.
Semi-Colon	Uses the semicolon (;) as the delimiter.
Space	Uses the space ( ) as the delimiter.
Tab	Uses the tab character as the delimiter.
Custom	Lets you enter a custom delimiter.
At the Left-Most Delimiter	Splits the column once only at the first occurrence of the delimiter.
At the Right-Most Delimiter	Splits the column once only at the last occurrence of the delimiter.
At Each Occurrence of the Delimiter	Splits the column into as many columns as there are delimiters.
Split into Columns	This leaves the number of rows as it is in the dataset and creates new columns for each new element resulting from the split operation.
Split into Rows	Creates a new row for each new element resulting from the split operation and duplicates the existing record as many times as there are split elements.
Advanced Options ➤ Number of Columns to Split Into	Allows you to set a maximum number of columns into which the data is split in chunks of the given number of characters. Any extra columns are placed in the rightmost column.
Advanced Options ➤ Quote Character	Separators inside a text that is contained in double quotes are not used to split the text into columns. Setting this option to "none" will split elements inside quotes.
Split using special characters	Enables the Insert Special Character button. You can then click this button and select the special character to split data on. The choice is between: Tab, Carriage Return, Line-Feed, Carriage Return, and Line-Feed or non-breaking space.

Splitting Columns by Number of Characters

Another variant on this theme is when text in each column is a fixed number of characters and needs to be broken down into constituent parts at specific intervals. Suppose, for instance, that you have a field where each group of (a certain number of) characters has a specific meaning, and you want to break it into multiple columns. Alternatively, suppose you want to extract the leftmost or rightmost *n* characters and leave the rest. A bank account or Social Security number are examples of this. This is where splitting a column by the number of characters can come in useful. As the principle is very similar to the process that we just saw, I will not repeat the whole thing again. All you have to do is choose the "By number of characters" menu option at step 5 in the previous exercise. Options for this type of operation are given in Table 7-12.

Table 7-12. *Options When Splitting a Column by Number of Characters*

Option	Description
Number of Characters	Lets you define the number of characters of data before splitting the column.
Once, As Far Left As Possible	Splits the column once only at the given number of characters in from the left.
Once, As Far Right As Possible	Splits the column once only at the given number of characters in from the right.
Repeatedly	Splits the column as many times as necessary to cut it into segments every defined number of characters.
Advanced Options ➤ Number of Columns to Split Into	Allows you to set a maximum number of columns into which the data is split in chunks of the given number of characters. Any extra columns are placed in the rightmost column.
Split into Columns	This leaves the number of rows as it is in the dataset and crates new columns for each new element resulting from the split operation.
Split into Rows	Creates a new row for each new element resulting from the split operation and duplicates the existing record as many times as there are split elements.

There are a couple of things to note when splitting columns:

- When splitting by a delimiter, Power BI Desktop makes a good attempt at guessing the maximum number of columns into which the source column must be split. If it gets this wrong (and you can see what its guesstimate is if you expand the Advanced Options box), you can override the number here.
- If you select a Custom Delimiter, Power BI Desktop displays a new box in the dialog where you can enter a specific delimiter.
- Not every record has to have the same number of delimiters. Power BI Desktop simply leave the rightmost column(s) blank if there are fewer split elements for a row.

■ **Note** You can only split columns if they are text data. The Split Column button remains grayed out if your intention is to try to split a date or numeric column.

Merging Columns

You may be feeling a certain sense of déjà vu when you read the title of this section. After all, we saw how to merge columns (that is, how to fuse the data from several columns into a single, wider column) in a previous chapter, did we not?

Yes, we did indeed. However, this is not the only time in this chapter that you will see something that you have tried previously. This is because Power BI Desktop Query repeats several of the options that are in the Transform Ribbon in the Add Column Ribbon. While these functions all work in much the same way, there is one essential difference. If you select an option from the Transform Ribbon, then the column(s) that you selected is *modified*. If you select a similar option from the Add Column Ribbon, then the original column(s) will not be altered, but a *new column* is added containing the results of the data transformation.

Merging columns is a case in point. Now, as I went into detail as to how to execute this kind of data transformation in the previous chapter, I will not describe it all over again here. Suffice it to say, if you Ctrl-click the headings of two or more columns and then click Merge Columns in the Add Column ribbon, you will still see the data from the selected columns concatenated into a single column. However, this time the original columns *remain* in the dataset. The new column is named Merged, exactly as was the case for the first of the columns that you selected when merging columns using the Transform ribbon.

The following are other functions that can either overwrite the data in existing columns *or* display the result as a new column:

- *Format*: Trims or changes the capitalization of text.
- *Extract*: Takes part of a column and creates another column from this data.
- *Parse*: Adds a column containing the source column data as JSON or XML strings.
- *Statistics*: Creates a new column of aggregated numeric values.
- *Standard*: Creates a new column of calculated numeric values.
- *Scientific*: Creates a new column by applying certain kinds of math operations to the values in a column.
- *Trigonometry*: Creates a new column by applying certain kinds of trigonometric operations to the values in a column.
- *Rounding*: Creates a new column by rounding the values in a column.
- *Information*: Creates a new column Indicating arithmetical information about the values in a column.
- *Date*: Creates a new column by extracting date elements from the values in a date column.
- *Time*: Creates a new column by extracting time elements from the values in a time or date/time column.
- *Duration*: Creates a new column by calculating the duration between two dates or date/times.

When transforming data, the art is to decide whether you want or need to keep the original column before applying one of these functions. Yet, once again, it is not really fundamental if you later decide that you made an incorrect decision, as you can always backtrack. Alternatively, you can always decide to insert new columns as a matter of principle and delete any columns that you really do not need at a later stage in the data transformation process.

Custom Columns

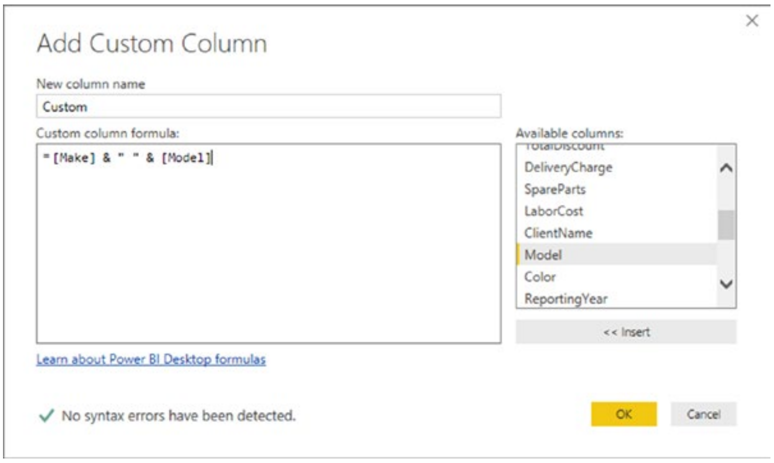
Another way to extend the original data table is to add more columns. Although these are known as *custom columns* in Power BI Desktop, they are also known more generically as *derived columns* or *calculated columns*. Although they can do many things, their essential role is to

- Concatenate (or join, if you prefer) existing columns.
- Add calculations to the data table.
- Extract a specific part of a column.
- Add flags to the table based on existing data.

The best way to understand these columns is probably to see them in action. You can then extend these principles in your own processes.

Initially, let's perform a column join and create a column named Vehicle, which concatenates the Make and Model columns with a space in between.

1. Load the C:\PowerBiDesktopSamples\CH07\CH07Example1.pbix sample file.
2. Click Edit Queries.
3. In the Add Column ribbon, click Custom Column. The Add Custom Column dialog is displayed.
4. Click the Make column in the column list on the right, then click the Insert button; `=[Make]` will appear in the Custom Column Formula box at the left of the dialog.
5. Enter & " " & in the Custom Column Formula box after `=[Make]`.
6. Click the Model column in the column list on the right, and then click the Insert button.
7. Click inside the New Column Name box and enter a name for the column. I call it CarType. The dialog will look like Figure 7-17.


Add Custom Column

New column name: Custom

Custom column formula:

```
= [Make] & " " & [Model]
```

Available columns: TotalDiscount, DeliveryCharge, SpareParts, LaborCost, ClientName, Model, Color, ReportingYear

✓ No syntax errors have been detected.

***Figure 7-17.** The Add Custom Column dialog*

8. Click OK. The new column is added to the right of the data table; it contains the results of the formula. Inserted Column appears in the Applied Steps list.

You can always double-click a column to insert it into the Custom Column Formula box if you prefer. To remove a column, simply delete the column name (including the square brackets) in the Custom Column Formula box.

■ **Note** You must always enclose a column name in square brackets.

Rather than take you step by step through other examples, I prefer to show you some of the formulas that you can use to calculate columns and extract data into a new column. These code snippets are given in Table 7-13.

Table 7-13. *Custom Column Code Examples*

Output	Code Snippet	Description
Column Calculations	`= [SalePrice]-[CostPrice]`	Subtracts the Cost Price from the Sale Price to give the Gross Margin
Column Arithmetic	`=[SalePrice] * 1.2`	Adds the UK sales tax (20%) to the Net Sale Price
Left	`Text.Start([Make],3)`	Returns the first three characters from the Make column
Right	`Text.End([Make],3)`	Returns the last three characters from the Make column
Up to a specific character	`Text.Start([Make],Text.PositionOf([Make]," "))`	Returns the leftmost characters up to the first space

■ **Note** If you are an Excel user, you can probably see a distinct similarity with how you build formulas in Excel and Power Pivot, except that here (as in Power Pivot) you use column names rather than cell references.

If you look ahead to Chapters 12-14, then you are probably wondering why you carry out operations like this in Power BI Desktop Query when you can do virtually the same thing in the data model. Well, it is true that there is some overlap; so you have the choice of which to use. You can perform certain operations at multiple stages in the data preparation and analysis process. It all depends on how you are using the data and with what tool you are carrying out the analyses.

The last three examples in Table 7-13 probably seem a little abstruse for the moment. This is because they are examples of how to use the Power BI data transformation language. This language is normally referred to as "M." Plumbing the depths of this language is outside the scope of this book, unfortunately.

Creating Columns from Examples

Creating your own columns can be a little scary if you have not had much experience with Excel or Power Pivot formulas, so the Power BI Desktop development team has tried to make your life easier by adding another way to create custom columns. Instead of referring to columns by the column name (and having to handle square brackets and other peculiar characters), you can build a new column by using the actual data in a row.

The following steps show an example of how to do this:

1. Load the C:\PowerBiDesktopSamples\CH07\CH07Example1.pbix sample file.
2. Click Edit Queries.
3. In the Add Column ribbon, click Column From Examples. A new kind of formula bar will appear above the data. It will look like Figure 7-18. At the same time, a new, empty column will be created at the right of the existing data.

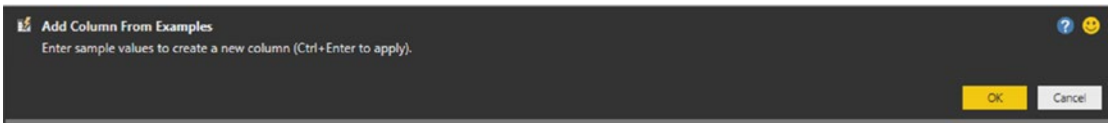
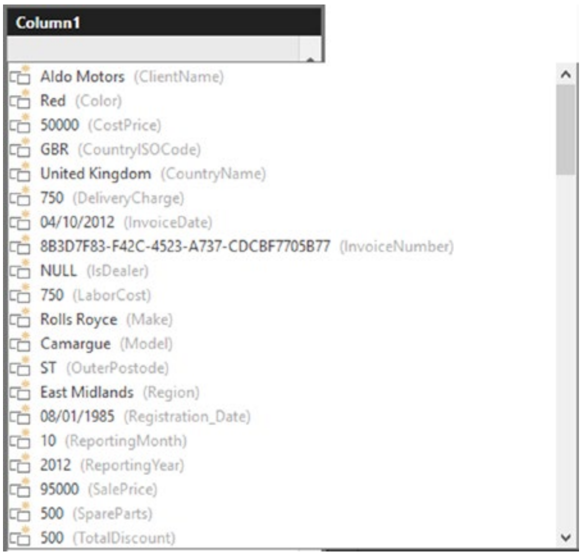

Add Column From Examples
Enter sample values to create a new column (Ctrl+Enter to apply).
OK
Cancel

Figure 7-18. Creating a column from examples

4. Double-click in the new column on the right. A list of data from each field will be displayed, as shown in Figure 7-19.



Aldo Motors (ClientName)
Red (Color)
50000 (CostPrice)
GBR (CountryISOCode)
United Kingdom (CountryName)
750 (DeliveryCharge)
04/10/2012 (InvoiceDate)
8B3D7F83-F42C-4523-A737-CDCBF7705B77 (InvoiceNumber)
NULL (IsDealer)
750 (LaborCost)
Rolls Royce (Make)
Camargue (Model)
ST (OuterPostode)
East Midlands (Region)
08/01/1985 (Registration_Date)
10 (ReportingMonth)
2012 (ReportingYear)
95000 (SalePrice)
500 (SpareParts)
500 (TotalDiscount)

Figure 7-19. Displaying the data from a row when creating a column from examples

5. Double-click Red to select the data from the Color column.
6. Enter a space, a hyphen, and a space, then type **Camargue** (this is the name of the model for this row).
7. Click OK in the formula bar at the top.

Power BI Desktop will add a new column containing the color, a separator, and the Model. Inserted Merged Column will be added as a new step in the Applied Steps list.

As you can see from this short example, creating columns by example lets you use the data from a column rather than the column name. It also removes the need for double quotes and ampersand characters that you had to use when writing the code to create a new column in the previous section.

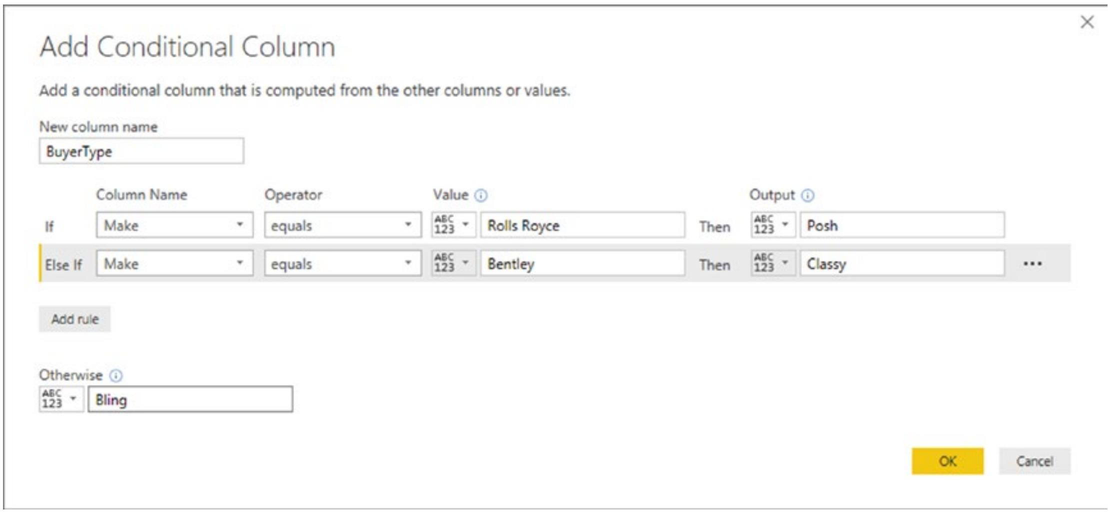
■ **Tip** If you select Column From Examples ➤ From Selection, then you will only see data from the selected columns when you double-click inside the new column to see samples of data as you did in step 6 of this example.

Adding Conditional Columns

Not all additional columns are a simple extraction or concatenation of existing data. There will be times when you will want to apply some simple conditions that define the contents of a new column. This is where Power BI Desktop's Conditional Column function comes into its own.

Conditional Columns are probably best understood with the aid of a practical example. So let's suppose that you want to add a column that contains a comment on the type of buyer for Brilliant British Cars's products. Here is how you can do this:

1. Load the C:\PowerBiDesktopSamples\CH07\CH07Example1.pbix sample file.
2. Click Edit Queries.
3. In the Add Column ribbon, click Conditional Column. The Add Conditional Column dialog will appear.
4. Enter **BuyerType** in the "New column name" field.
5. Select Make as the column name.
6. Leave Equals as the operator.
7. Enter **Rolls Royce** as the value.
8. Enter **Posh** as the output.
9. Click Add Rule.
10. Select Make as the column name, leave Equals as the operator, enter **Bentley** as the value, and add **Classy** as the output.
11. Enter **Bling** in the Otherwise field. The dialog will look like Figure 7-20.


Add Conditional Column

Add a conditional column that is computed from the other columns or values.

New column name: BuyerType

	Column Name	Operator	Value		Output
If	Make	equals	Rolls Royce	Then	Posh
Else If	Make	equals	Bentley	Then	Classy

Otherwise: Bling

***Figure 7-20.** The Add Conditional Column dialog*

12. Click OK. The new column will be added containing either Posh, Classy, or Bling, depending on the make for each record. Added Conditional Columns will appear as the new step in the Applied Steps list.

As you can see from the Add Conditional Column dialog, it has a range of options that you can tweak when defining the logic for the data matching. These options are outlined in Table 7-14.

Table 7-14. *Custom Column Operators*

Operator	Description
Equals	Sets the text that must match the contents of the selected field for the output to be applied.
Does Not Equal	Sets the text that must not match the contents of the selected field for the output to be applied.
Begins With	Sets the text at the left of the selected field for the output to be applied.
Does Not Begin With	Sets the text that must not appear at the left of the selected field for the output to be applied.
Ends With	Sets the text at the right of the selected field for the output to be applied.
Does Not End With	Sets the text that must not appear at the right of the selected field for the output to be applied.
Contains	Sets the text that can appear anywhere in the selected field for the output to be applied.
Does Not Contain	Sets the text that cannot appear anywhere in the selected field for the output to be applied.

It is also worth noting that the comparison value, the output, and the alternative output can be values (as was the case in this example), columns, or parameters (which you will learn about in Chapter 9). If you want to remove a rule, simply click the ellipses at the right of the required rule and select Delete.

■ **Tip** Should you wish to alter the order of the rules in the Add Conditional Column dialog, all you have to do is click the ellipses at the right of the selected rule and select Move Up or Move Down from the popup menu.

Index Columns

An index column is a new column that numbers every record in the table sequentially. This numbering scheme applies to the table, because it is currently sorted and begins at zero. There are many situations where an index column can be useful. The following are some examples:

- Reapply a previous sort order.
- Create a unique reference for every record.
- Prepare a recordset for use as a dimension table in a Power BI Desktop data model. In cases like this, the index column becomes what dimensional modelers call a *surrogate key*.

This list is not intended to be exhaustive in any way; you will almost certainly find other uses as you work with Power BI Desktop. Whatever the need, here is how to add an index column:

1. In the Add Column ribbon, click Index Column. The new, sequentially numbered column is added at the right of the table, and Added Index is added to the Applied Steps list.
2. Scroll to the right of the table and rename the index column; it is currently named Index.

You have a fairly free hand when it comes to deciding how to begin numbering an index column. The choices are as follows:

- Start at 0 and increment by a value of 1 for each row.
- Start at 1 and increment by a value of 1 for each row.
- Start at any number and increase by any number.

As you saw in step 1, the default is for Power BI Desktop Query to begin numbering rows at 0. However, you can choose another option by clicking the small triangle to the right of the Add Index Column button. This displays a menu with the three options outlined.

Selecting the third option, Custom, displays the dialog that you see in Figure 7-21.

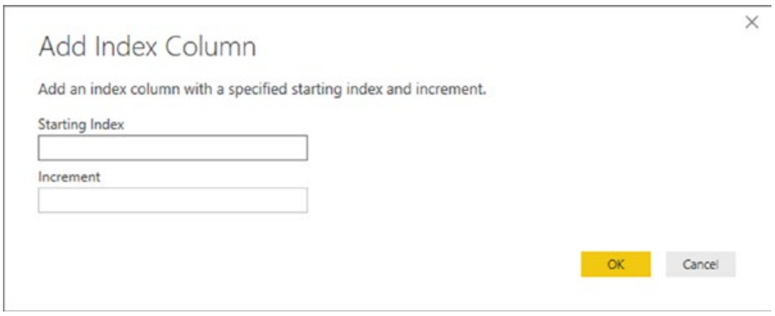

Add Index Column
Add an index column with a specified starting index and increment.
Starting Index
Increment
OK
Cancel

Figure 7-21. *The Add Index Column dialog*

This dialog lets you specify the start number for the first row in the dataset as well as the increment that is added for each record.

Conclusion

In this chapter, you learned some essential techniques that you can use to cleanse and extend datasets. You saw how to round numbers up and down, how to deliver conformed text presentation, and how to remove extraneous spaces and non-printing characters from columns of data.

You also saw how to replace values inside columns, as well as ways of applying mathematical, statistical, and trigonometric functions to numbers. Other techniques covered extracting date, time, and duration elements from date/time and duration columns.

Finally, you saw a series of techniques that help you to add new columns based on the data in existing columns. These range from simple copies of an entire column or combining columns to extracting parts of a column's data or even deducing different data that is added to a new column using simple logic.

It is now time to see how you can join hitherto separate datasets into single queries, and parse complex data types to add them to a dataset. You will even learn how to load multiple files in a single query and how to pivot and unpivot data. All of his will be the subject of the next chapter.

CHAPTER 8

Data Mashup

In the previous two chapters, you saw how to hone your dataset so that you defined only the rows and columns of data that you really need. Then you learned how to cleanse and complete the data that they contain. In this chapter, you will learn how to build on these foundations to deliver data that is ready to be molded into a structured and useable data model.

The generic term for this kind of data preparation in Power BI Desktop is *data mashup*. It covers the following:

- *Joining datasets* (or queries, if you prefer): This involves taking two queries and linking them so that you display the data from both sources as a single dataset. You will learn how to extend a query with multiple columns from a second query as well as how to aggregate the data from a second query and add this to the initial dataset. You will also see how to create complex joins when merging queries.
- *Assembling data from the files in a folder*: Here you will build on the knowledge acquired previously to use the Query Editor to filter the file types from a source folder and only amalgamate the ones that you choose.
- *Pivoting and unpivoting data*: If you need to switch data in rows to display as columns—or vice versa—then you can get the Power BI Desktop Query Editor to help you do exactly this. This means that you can guarantee that the data in all the tables that you are using conforms to a standardized tabular structure.
- *Parsing column contents*: This can be required to break down the contents of a column into data structured as a data table.

I want to be clear that separating data preparation into these two apparently contradictory approaches—reducing datasets only to augment them later—is not necessarily the way that you will work in practice. Given the range of features available in Power BI Desktop Query, I have simply tried to apply some structure to the way that they are explained. Hopefully, this will make the data-handling toolkit that is the Power BI Desktop Query Editor a little easier to understand. I imagine that when you are delving into data with Power BI Desktop, you will mix and alternate many of the techniques that are outlined in Chapters 6 through 8 in any order. After all, one of the great strengths of Power BI Desktop Query is that it does not impose any strict way of working and lets you experiment freely. So remember that you are at liberty to take any approach you want when transforming source data. The only thing that matters is that it gives you the result that you want.

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_8

The Power BI Desktop Query Editor View Ribbon

Until now, we have concentrated our attention on the Power BI Desktop Query Editor Home, Transform, and Add Column ribbons. This is for the good and simple reason that these ribbons are where nearly all the action takes place. There is, however, a fourth essential Power BI Desktop Query Editor ribbon—the View ribbon. The buttons that it contains are shown in Figure 8-1 and the options are explained in Table 8-1.

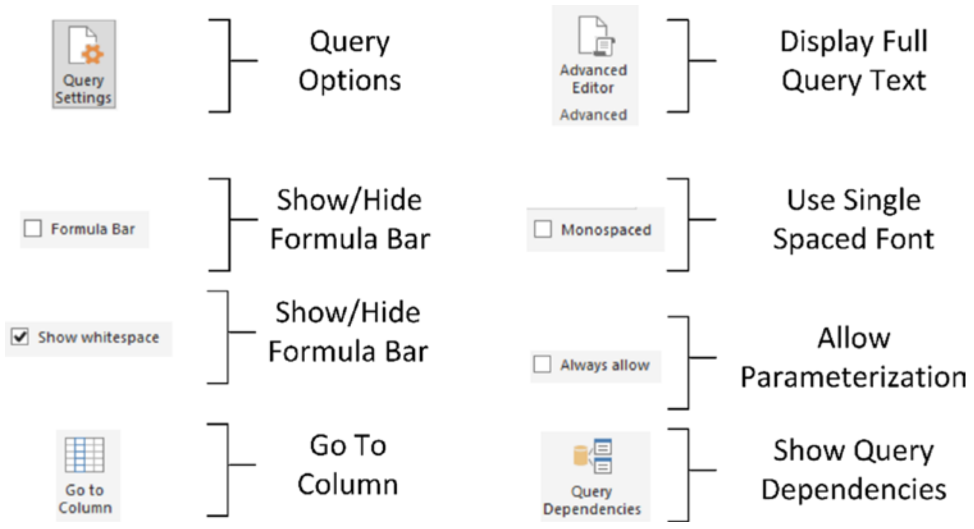


Figure 8-1. *The Power BI Desktop View ribbon*

Table 8-1. *Power BI Desktop View Ribbon Options*

Option	Description
Query Settings	Displays or hides the Query Settings pane at the right of the Power BI Desktop window. This includes the Applied Steps list.
Formula Bar	Shows or hides the formula bar containing the M language code for a transformation step.
Monospaced	Displays data in a monospaced (Courier) font.
Show Whitespace	Shows whitespace and newline characters.
Go to Column	Allows you to move to a column selected from those available.
Always Allow	Allows parameterization.
Advanced Editor	Displays the Advanced Editor dialog containing all the code for the steps in the query.
Query Dependencies	Displays the sequence of query links and dependencies.

Possibly the only option that is not immediately self-explanatory is the Advanced Editor button. It displays the code for all the transformations in the query as a single block of "M" language script.

■ **Tip** Personally, I find that the Query Settings pane and the formula bar are too vital to be removed from the Power BI Desktop Query window when transforming data. Consequently, I tend to leave them visible. If you need the screen real estate, however, then you can always hide them for a while.

Merging Data

Until now, we have treated each individual query as if it existed in isolation. The reality, of course, is that you will frequently be required to use the output of one query in conjunction with the output of another to join data from different tables in various ways. Assuming that the results of one query share a common field (or fields) with another query, you can "join" queries into a single data table. Power BI Desktop calls this a merge operation, and it enables you, among other things, to

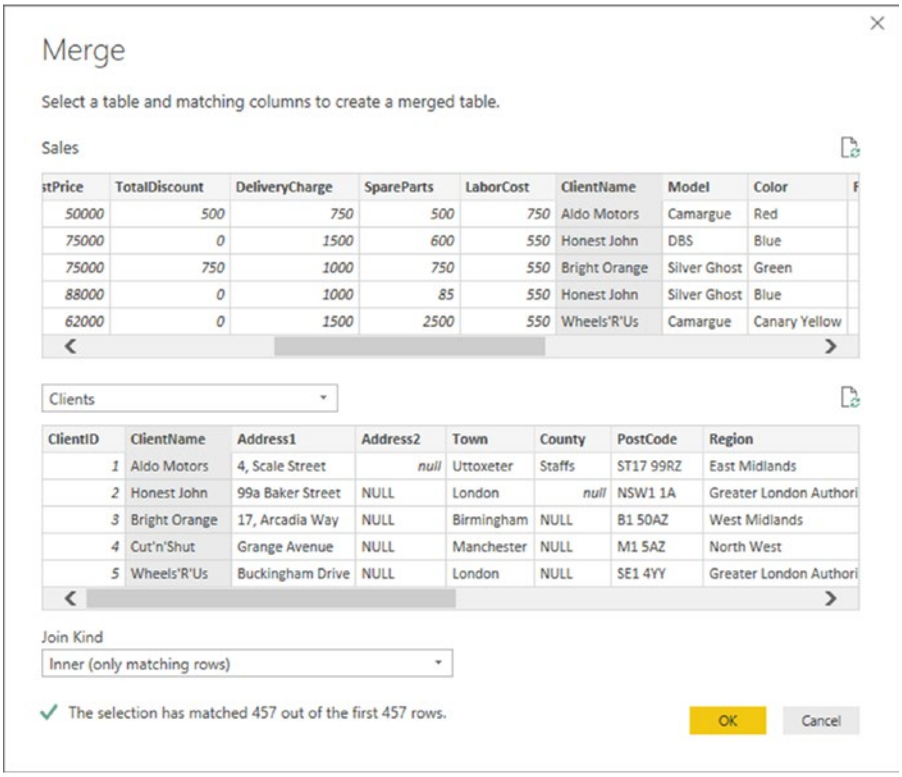
- Look up data elements in another "reference" table to add lookup data. For example, you may want to add a client name where only the client code exists in your main table.
- Aggregate data from a "detail" table (such as invoice lines) and include the totals in a higher-grained table, such as a table of invoices.

Here, again, the process is not difficult. The only fundamental factor is that the two tables, or queries, that you are merging must have a shared field or fields that enable the two tables to match records coherently. Let's look at a couple of examples.

Adding Data

First, let's try looking up extra data that we will add to a query:

1. In a new, empty Power BI Desktop file, load both the worksheets in the C:\PowerBiDesktopSamples\CH08\SalesData.xlsx Excel file.
2. Click the Edit Queries button in the Home ribbon.
3. Click the query named Sales in the Queries pane of the Power BI Desktop Query window.
4. Click the Merge Queries button in the Home ribbon. The Merge dialog will appear.
5. In the upper part of the dialog—where an overview of the output from the current query is displayed—scroll to the right and click the ClientName column title. This column is highlighted.
6. In the popup under the upper table, select the Clients query. The output from this query will appear in the lower part of the dialog.
7. In the lower table, select the column title for the column—the join column—that maps to the column that you selected in step 4. This will also be the ClientName column. This column is then selected in the lower table. You may be asked to set privacy levels for the data sources. If this is the case, set them to Public.
8. Select Inner (only matching rows) from the Join Kind popup menu. The dialog will look like Figure 8-2.


Merge

Select a table and matching columns to create a merged table.

Sales

stPrice	TotalDiscount	DeliveryCharge	SpareParts	LaborCost	ClientName	Model	Color	F
50000	500	750	500	750	Aldo Motors	Camargue	Red	
75000	0	1500	600	550	Honest John	DBS	Blue	
75000	750	1000	750	550	Bright Orange	Silver Ghost	Green	
88000	0	1000	85	550	Honest John	Silver Ghost	Blue	
62000	0	1500	2500	550	Wheels'R'Us	Camargue	Canary Yellow	

Clients

ClientID	ClientName	Address1	Address2	Town	County	PostCode	Region
1	Aldo Motors	4, Scale Street	null	Uttoxeter	Staffs	ST17 99RZ	East Midlands
2	Honest John	99a Baker Street	NULL	London	null	NSW1 1A	Greater London Authori
3	Bright Orange	17, Arcadia Way	NULL	Birmingham	NULL	B1 50AZ	West Midlands
4	Cut'n'Shut	Grange Avenue	NULL	Manchester	NULL	M1 5AZ	North West
5	Wheels'R'Us	Buckingham Drive	NULL	London	NULL	SE1 4YY	Greater London Authori

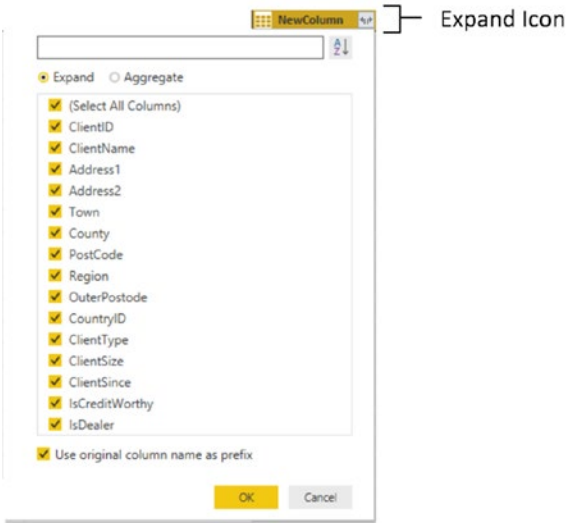
Join Kind

Inner (only matching rows)

✓ The selection has matched 457 out of the first 457 rows.

Figure 8-2. *The Merge dialog*

9. Click OK. A new column is added to the right of the existing data table.
10. Scroll to the right of the existing data table and click the Expand icon to the right of the column name (it has probably been named New Column and every row contains the word *Table*). The popup list of all the available fields in this data table (or query, if you prefer) is displayed, as shown in Figure 8-3.


NewColumn

Expand Icon

[✓ Expand] Aggregate

- [x] (Select All Columns)
- [x] ClientID
- [x] ClientName
- [x] Address1
- [x] Address2
- [x] Town
- [x] County
- [x] PostCode
- [x] Region
- [x] OuterPostode
- [x] CountryID
- [x] ClientType
- [x] ClientSize
- [x] ClientSince
- [x] IsCreditWorthy
- [x] IsDealer

- [x] Use original column name as prefix

Figure 8-3. *The fields available in a joined query*

11. Ensure that the Expand radio button is selected.
12. Clear the selection of all the columns by unchecking the (Select All Columns) check box.
13. Select the following columns:
    a. ClientName
    b. ClientSize
    c. ClientSince
14. Click OK. The selected columns from the linked table are merged into the main table, and the link to the reference table (New Column) is removed.
15. Rename the columns that have been added, and apply and close the query.

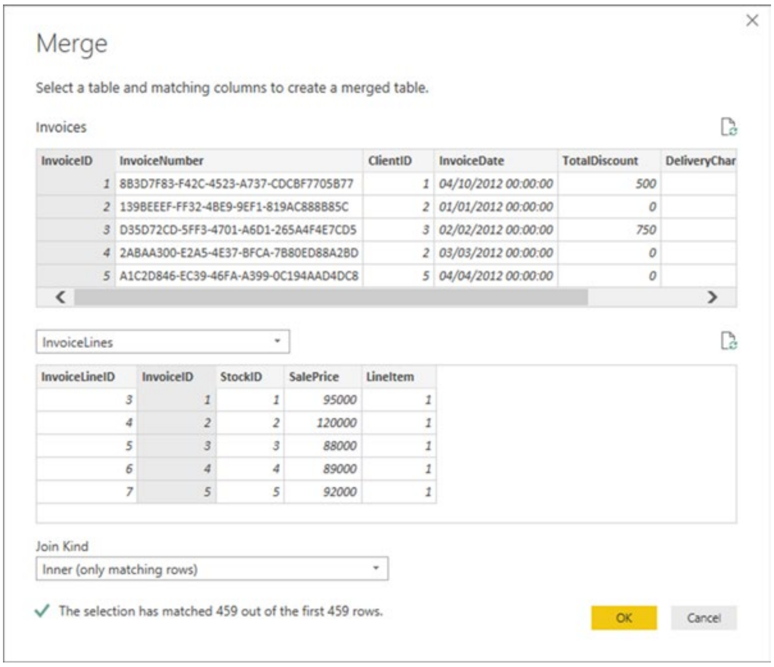
You now have a single table of data that contains data from two linked data sources. Reprocessing the Sales query will also reprocess the dependent *clients* query and result in the latest version of the data being reloaded. Moreover, each of the columns that has been added has the title "NewColumn", followed by the column name from the merged data source.

■ **Note** You probably noticed that the Merge dialog indicated how many matching records there were in the two queries. This can be a useful indication that you have selected the correct column(s) to join the two queries.

Aggregating Data During a Merge Operation

If you are not just looking up reference data but need to aggregate data from a separate table and then add the results to the current query, then the process is largely similar. This second approach, however, is designed to suit another completely different requirement. Previously, you saw the case where the current query had many records that mapped to a *single* record in the lookup table. This second approach is for when your current (or main) query has a single record where there are *multiple* linked records in the second query. Consequently, you need to aggregate the data in the second table to bring the data across into the first table. Here is a simple example, using some of the sample data from the C:\PowerBiDesktopSamples folder:

1. Find the InvoicesAndInvoiceLines.xlsx Excel source file in the C:\PowerBiDesktopSamples\CH08 folder. Edit the two worksheets it contains (Invoices and InvoiceLines) into the Power BI Desktop Query Editor. This will create two queries.
2. Click the query named Invoices in the Queries pane on the left.
3. In the Home ribbon, click the Merge Queries button. The Merge dialog will open. You will see some of the data from the Invoices dataset in the upper part of the dialog.
4. Click anywhere inside the InvoiceID column. This column is selected.
5. In the popup, select the InvoiceLines query. You will see some of the data from the Invoices dataset in the lower part of the dialog.
6. Click anywhere inside the InvoiceID column for the lower table. This column is selected.
7. Select Inner (only matching rows) from the Join Kind popup menu. The dialog will look like Figure 8-4.


Merge

Select a table and matching columns to create a merged table.

Invoices

InvoiceID	InvoiceNumber	ClientID	InvoiceDate	TotalDiscount	DeliveryChar
1	8B3D7F83-F42C-4523-A737-CDCBF7705B77	1	04/10/2012 00:00:00	500	
2	139BEEEF-FF32-4BE9-9EF1-819AC888B85C	2	01/01/2012 00:00:00	0	
3	D35D72CD-5FF3-4701-A6D1-265A4F4E7CD5	3	02/02/2012 00:00:00	750	
4	2ABAA300-E2A5-4E37-BFCA-7B80ED88A2BD	2	03/03/2012 00:00:00	0	
5	A1C2D846-EC39-46FA-A399-0C194AAD4DC8	5	04/04/2012 00:00:00	0	

InvoiceLines

InvoiceLineID	InvoiceID	StockID	SalePrice	LineItem
3	1	1	95000	1
4	2	2	120000	1
5	3	3	88000	1
6	4	4	89000	1
7	5	5	92000	1

Join Kind
Inner (only matching rows)

✓ The selection has matched 459 out of the first 459 rows.

Figure 8-4. *The Merge dialog when aggregating data*

8. Click OK.

9. Scroll to the right of the existing data table. You will see a new column (named NewColumn) that contains the word *Table* in every cell. This column will look something like Figure 8-5.

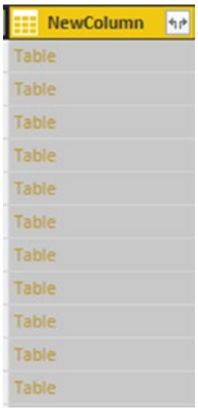


Figure 8-5. *A merged column*

10. Click the Expand icon to the right of the new column title (the two arrows facing left and right). The popup list of all the available fields in the InvoiceLines query is displayed.

11. Select the Aggregate radio button.

12. Select the Sum Of SalePrice field.

13. Uncheck the “Use original column name as prefix” check box. The dialog will look like Figure 8-6.

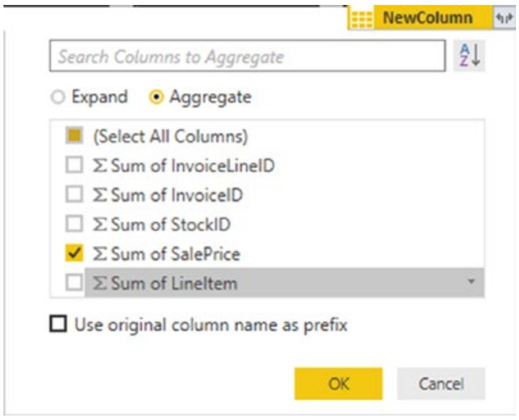


Figure 8-6. *The available fields from a merged dataset*

14. Click OK.

Power BI Desktop will add up the total sale price for each invoice and add this as a new column. Naturally, you can choose the type of aggregation that you wish to apply (before clicking OK), if the sum is not what you want. To do this, place the cursor over the column that you want to aggregate (see step 11 in the preceding exercise) and click the popup menu at the right of the field name. Power BI Desktop will suggest a set of options. The available aggregation options are explained in Table 8-2.

Table 8-2. *Merge Aggregation Options*

Option	Description
Sum	Returns the total value of the field.
Average	Returns the average value of the field.
Median	Returns the median value of the field.
Minimum	Returns the minimum value of the field.
Maximum	Returns the maximum value of the field.
Count (All)	Counts all records in the dataset.
Count (Not Blank)	Counts all records in the dataset that are not empty.

■ **Tip** If you loaded the data instead of editing the query in step 1, simply Click the Edit Queries button in the Home ribbon to switch to the Query Editor.

The merge process that you have just seen, while not complex in itself, suddenly opens up many new horizons. It means that you can now create multiple separate queries that you can then use together to expand your data in ways that allow you to prepare quite complex datasets.

Here are a couple of comments I need to make about the merge operation:

- Only queries that have been previously created in the Power BI Desktop Query window can be used when merging datasets. So remember to connect to all the datasets that you require before attempting a merge operation.
- Refreshing a query will cause any other queries that are merged into this query to be refreshed also. This way you will always get the most up-to-date data from all the queries in the process.

■ **Tip** You do not have to load a dataset to merge it with another query. You can simply edit it. This will establish a connection to the source data.

Types of Join

When merging queries—either to join data or to aggregate values—you are faced with a choice when it comes to how to link the two queries. The choice of join can have a profound effect on the resulting dataset. Consequently, it is important to understand the six join types that are available. These are described in Table 8-3.

Table 8-3. *Join Types*

Join Type	Explanation
Left Outer	Keeps all records in the upper dataset in the Merge dialog (the dataset that was active when you began the merge operation). Any matching rows (those that share common values in the join columns) from the second dataset are kept. All other rows from the second dataset are discarded.
Right Outer	Keeps all records in the lower dataset in the Merge dialog (the dataset that was not active when you began the merge operation). Any matching rows (those that share common values in the join columns) from the upper dataset are kept. All other rows from the upper dataset (the dataset that was active when you began the merge operation) are discarded.
Full Outer	All rows from both queries are retained in the resulting dataset. Any records that do not share common values in the join field(s) contain blanks in certain columns.
Inner	Only joins queries where there is an exact match on the column(s) that are selected for the join. Any rows from either query that do not share common values in the join column(s) are discarded.
Left Anti	Keeps only rows from the upper (first) query.
Right Anti	Keeps only rows from the lower (second) query.

■ **Note** When you use any of the *outer* joins, you are keeping records that do not have any corresponding records in the second query. Consequently, the resulting dataset contains empty values for some of the columns.

When you are expanding the column that is the link to a merged dataset, you have a couple of useful options that are worth knowing about:

- Use original column name as prefix
- Search columns to expand

Use the Original Column Name As the Prefix

You will probably find that some columns from joined queries can have the same names in both source datasets. It follows that you need to identify which column came from which dataset. If you leave the check box selected for the "Use original column name as prefix" merge option (which is the default), any merged columns will include the source query name to help you identify the data more accurately.

If you find that these longer column names only get in the way, you can unselect this check box. This will leave the added columns from the second query with their original names. However, because Power BI Desktop cannot accept duplicate column names, any new columns will have .1, .2, and so forth, added to the column name.

Search Columns to Expand

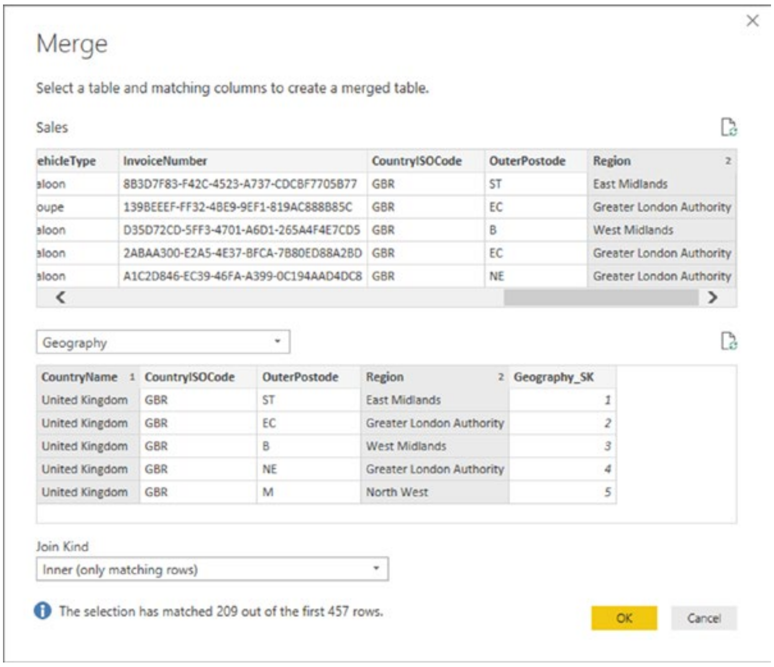
If you are merging a query with a second query that contains a large number of columns, then it can be laborious to search for the columns that you want to include. To narrow your search you can enter a few characters from the column that you are looking for. The more characters you type, the fewer matching columns are displayed in the Expansion popup dialog.

Joining on Multiple Columns

In the examples so far, you only joined queries on a single column. While this may be possible if you are looking at data that comes from a clearly structured source (such as a relational database), you may need to extend the principle when joining queries from diverse sources. Fortunately, Power BI Desktop allows you to join queries on multiple columns when the need arises.

As an example of this, the sample data contains a file that I have prepared as an example of how to join queries on more than one column. This sample file contains data from the sources that you saw in previous chapters. However, they have been modeled as a data warehouse star schema. To complete the model, you need to join a dimension named Geography to a fact table named Sales so that you can add the field GeographySK to the fact table. However, the Sales table and the Geography table share three fields (Country, Region, and Town) that must correspond for the queries to be joined. The following explains how to perform a join using multiple fields:

1. Click Get Data, select Excel as the source, and in the C:\PowerBiDesktopSamples\CH08\StarSchema.xlsx Excel file select the two worksheets.
2. Click Edit Queries to open the Query Editor.
3. Select the Sales query from the list of existing queries from the Queries pane on the left of the Power BI Desktop Query window.
4. In the Home ribbon, click the Merge Queries button. The Merge dialog will appear.
5. In the popup list of queries, select Geography as the second query to join to the first (upper) query.
6. Select Inner (only matching rows) from the Join Kind popup.
7. In the upper list of fields (taken from the Sales table), Ctrl-click the fields CountryName and Region, in this order. A small number will appear to the right of each column header indicating the order that you selected the columns.
8. In the lower list of fields (taken from the Sales table) Ctrl-click in the fields CountryName and Region, in this order. A small number will appear to the right of each column header indicating the order that you selected the columns.
9. Verify that you have a reasonable number of matching rows in the information message at the bottom of the dialog. The dialog will look like Figure 8-7.



Merge

Select a table and matching columns to create a merged table.

Sales

ehicleType	InvoiceNumber	CountryISOCode	OuterPostode	Region 2
aloon	8B3D7F83-F42C-4523-A737-CDCBF7705B77	GBR	ST	East Midlands
oupe	139BEEEF-FF32-4BE9-9EF1-819AC888B85C	GBR	EC	Greater London Authority
aloon	D35D72CD-5FF3-4701-A6D1-265A4F4E7CD5	GBR	B	West Midlands
aloon	2ABAA300-E2A5-4E37-BFCA-7B80ED88A2BD	GBR	EC	Greater London Authority
aloon	A1C2D846-EC39-46FA-A399-0C194AAD4DC8	GBR	NE	Greater London Authority

Geography

CountryName 1	CountryISOCode	OuterPostode	Region 2	Geography_SK
United Kingdom	GBR	ST	East Midlands	1
United Kingdom	GBR	EC	Greater London Authority	2
United Kingdom	GBR	B	West Midlands	3
United Kingdom	GBR	NE	Greater London Authority	4
United Kingdom	GBR	M	North West	5

Join Kind

Inner (only matching rows)

The selection has matched 209 out of the first 457 rows.

***Figure 8-7.** Joining queries using multiple columns*

10. Click OK.

You can then continue with the data mashup. In this example, that would be adding the GeographySK field to the Sales query and then removing the Country, Region, and Town fields from the Sales query.

There is no real limit to the number of columns that can be used when joining queries. It will depend entirely on the shape of the source data. However, each column used to define the join must exist in both datasets and each pair of columns must be of the same (or a similar) data type.

Preparing Datasets for Joins

You could have to carry out a little preparatory work on real-world datasets before joining queries. More specifically, any columns that you join have to be the same basic data type. Put simply, you need to join text-based columns to other text-based columns, number columns to number columns, and date columns to date columns. If the columns are *not* the same data type, you receive a warning message when you try to join the columns in the Merge dialog.

Consequently, it is nearly always a good idea to take a look at the columns that you will use to join queries *before* you start the merge operation itself. Remember that data types do not have to be identical, just similar. So a decimal number type can map to a whole number, for instance.

You might also have to cleanse the data in the columns that are used for joins before attempting to merge queries. This could involve the following:

- Removing trailing or leading spaces in text-based columns
- Isolating part of a column (either in the original column or as a new column) to use in a join
- Verifying that appropriate data types are used in join columns

Correct and Incorrect Joins

Merging queries is the one data mashup operation that is often easier in theory than in practice, unfortunately. If the source queries were based on tables in a relational or even dimensional database, then joining them could be relatively easy, as a data architect will (hopefully) have designed the database tables to allow for them to be joined. However, if you are joining two completely independent queries, then you could face several major issues:

- The columns do not map.
- The columns map, but the result is a massive table with duplicate records.

Let's take a look at these possible problems.

The Columns Do Not Map

If the columns do not map (that is, you have joined the data but get no resulting records), then you need to take a close look at the data in the columns that you are using to establish the join. The questions you need to ask are as follows:

- Are the values in the two queries the same data type?
- Do the values really map—or are they different?
- Are you using the correct columns?
- Are you using too many columns and so specifying data that is not in both queries?

The Columns Map, but the Result Is a Massive Table with Duplicate Records

Joining queries depends on isolating *unique* data in both source queries. Sometimes a single column does not contain enough information to establish a unique reference that can uniquely identify a row in the query.

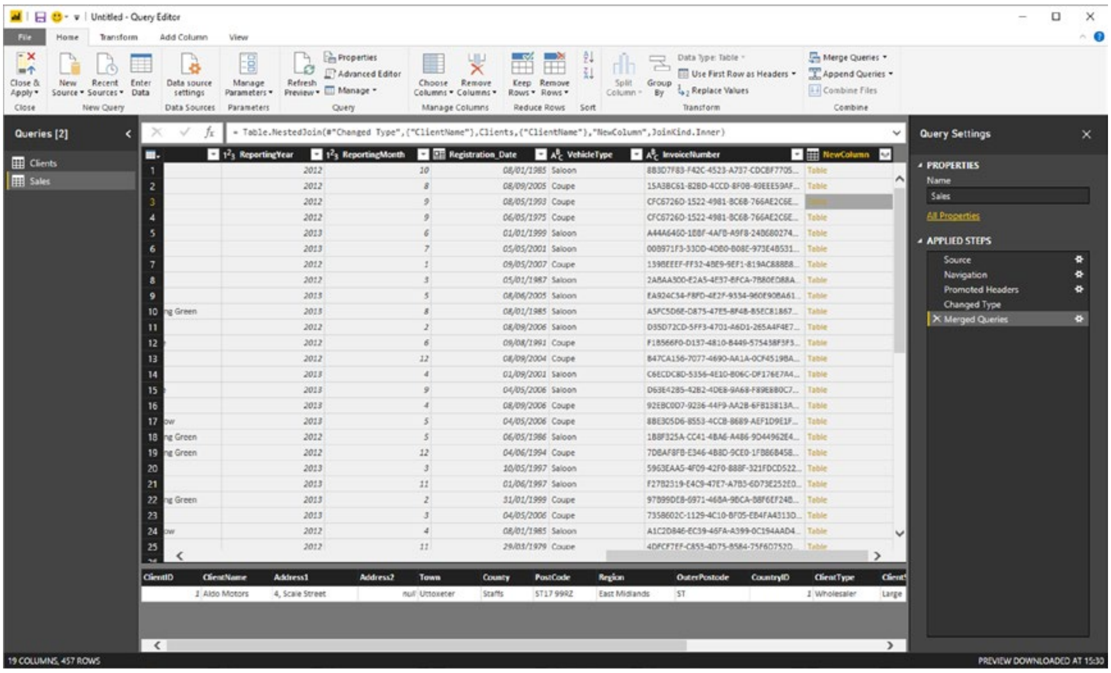
In these cases, you need to use two or more columns to join queries—or else rows will be duplicated in the result. Therefore, once again, you need to look carefully at the data and decide on the minimum number of columns that you can use to join queries correctly.

Examining Joined Data

Joining data tables is not always easy. Neither is deciding if the outcome of a merge operation will produce the result that you expect. So Power BI Desktop Query includes a solution to these kinds of dilemma. It can help you more clearly see what a join has done. More specifically, it can show you for each record in the first query exactly which rows are joined from the second query.

Do the following to see this in action:

1. Carry out steps 1 through 8 in the first example in the "Merging Data" section (in the "Adding Data" subsection).
2. Scroll to the right in the data table. You will see the new column named NewColumn (as shown in Figure 8-8).



```
= Table.NestedJoin(#"Changed Type",{"ClientName"},Clients,{"ClientName"},"NewColumn",JoinKind.Inner)
```

		ReportingYear	ReportingMonth	Registration_Date	VehicleType	InvoiceNumber	NewColumn
1		2012	10	08/01/1985	Saloon	8B3D7F83-F42C-4523-A737-CDCBF7705...	Table
2		2012	8	08/09/2005	Coupe	15A3BC61-82BD-4CCD-8F0B-49EEE59AF...	Table
3		2012	9	08/05/1993	Coupe	CFC6726D-1522-4981-8C68-766AE2C6E...	Table
4		2012	9	06/05/1975	Coupe	CFC6726D-1522-4981-8C68-766AE2C6E...	Table
5		2013	6	01/01/1999	Saloon	A44A6460-1E8F-4AFB-A9F8-24B680274...	Table
6		2013	7	05/05/2001	Saloon	00B971F3-33DD-4D80-B08E-973E48531...	Table
7		2012	1	09/05/2007	Coupe	139BEEEF-FF32-4BE9-9EF1-819AC88888...	Table
8		2012	3	05/01/1987	Saloon	2ABAA300-E2A5-4E37-8FCA-7B80ED88A...	Table
9		2013	5	08/06/2005	Saloon	EA924C34-F8FD-4E2F-9334-960E908A61...	Table
10	ng Green	2013	8	08/01/1985	Saloon	A5FC5D6E-D875-47E5-8F48-85EC81867...	Table
11		2012	2	08/09/2006	Saloon	D35D72CD-5FF3-4701-A6D1-265A4F4E7...	Table
12		2012	6	09/08/1991	Coupe	F18566F0-D137-4810-B449-575438F3F3...	Table
13		2012	12	08/09/2004	Coupe	847CA156-7077-4690-AA1A-0CF4519BA...	Table
14		2013	4	01/09/2001	Saloon	C6ECDC8D-5356-4E10-806C-DF176E7A4...	Table
15		2013	9	04/05/2006	Saloon	D63E4285-42B2-4DE8-9A68-F89E880C7...	Table
16		2013	4	08/09/2006	Coupe	92EBC0D7-9236-44F9-AA2B-6FB13813A...	Table
17	ow	2013	5	04/05/2006	Coupe	8BE305D6-8553-4CCB-8689-AEF1D9E1F...	Table
18	ng Green	2012	5	06/05/1986	Saloon	1B8F325A-CC41-4BA6-A486-9D44962E4...	Table
19	ng Green	2012	12	04/06/1994	Coupe	7DBAF8FB-E346-4B8D-9CE0-1FB86B458...	Table
20		2013	3	10/05/1997	Saloon	5963EAA5-4F09-42F0-888F-321FDCD522...	Table
21		2013	11	01/06/1997	Saloon	F27B2319-E4C9-47E7-A7B3-6D73E252E0...	Table
22	ng Green	2013	2	31/01/1999	Coupe	97B99DE8-6971-468A-9BCA-88F6EF248...	Table
23		2013	3	04/05/2006	Coupe	7358602C-1129-4C10-BF05-EB4FA4313D...	Table
24	ow	2012	4	08/01/1985	Saloon	A1C2D846-EC39-46FA-A399-0C194AAD4...	Table
25		2012	11	29/03/1979	Coupe	4DFCF7EF-C853-4D75-8584-75F6D752D...	Table

ClientID	ClientName	Address1	Address2	Town	County	PostCode	Region	OuterPostode	CountryID	ClientType	Client
1	Aldo Motors	4, Scale Street	null	Uttoxeter	Staffs	ST17 99RZ	East Midlands	ST	1	Wholesaler	Large

19 COLUMNS, 457 ROWS

PREVIEW DOWNLOADED AT 15:30

Figure 8-8. *Joined data*

3. Click to the *right* of the word *Table* in the row where you want to see the joined data. Note that you must *not* click the word *Table*. A second table will appear under the main query's data table containing the data from the second query that is joined for this particular row. Figure 8-8 shows an example of this.

This technique is as simple as it is useful. There are nonetheless a few comments that I need to make:

- You can resize the lower table (and consequently display more or less data from the second joined table) by dragging the bottom border of the top data table up or down.
- Clicking to the right of the word *Table* in the NewColumn column will enable the Expand and Aggregate buttons in the Transform ribbon.
- Clicking the word *Table* in the NewColumn column adds a new step to the query that replaces the source data with the linked data. You can also do this by right-clicking inside the NewColumn column and selecting Drill Down.

■ **Note** Drilling down into the merged table in effect limits the query to the row(s) of the subtable. Consequently, you have to delete this step if you want to access all the data in the merged tables.

The Expand and Aggregate Buttons

Power BI Desktop Query offers an alternative to using the NewColumn popup menu to expand or aggregate data when merging queries. If you have clicked to the right of the word *Table* in the NewColumn, you can then click the newly activated Expand button or Aggregate button in the Transform ribbon to display the Expand dialog or Aggregate dialog.

The only real difference between the dialogs and the popups is that the Expand dialog also has an option where you can add a new column prefix that you choose for the additional columns from the second query. You can see this in Figure 8-9.

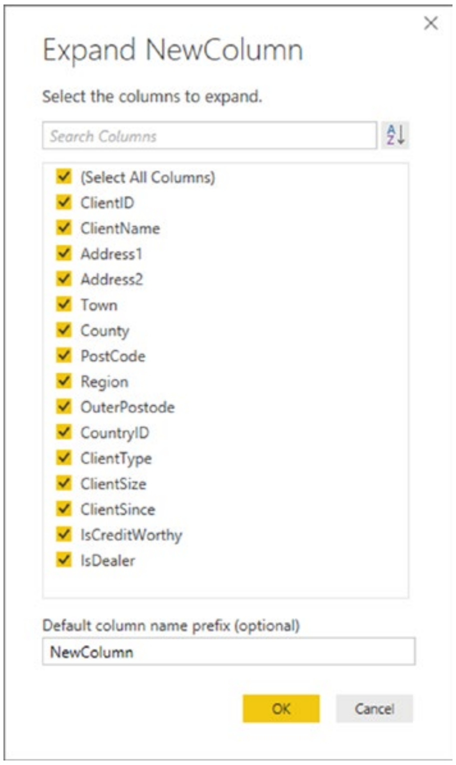


Figure 8-9. *The Expand New Column dialog*

Appending Data

Not all source data is delivered in its entirety in a single file or as a single database table. You may be given access to two or more tables or files that have to be loaded into a single table in Excel or Power Pivot. In some cases, you might find yourself faced with hundreds of files—all text, CSV, or Excel format—and the requirement to load them all into a single table that you will use as a basis for your analysis. Well, Power BI Desktop can handle these eventualities, too.

Adding the Contents of One Query to Another

In the simplest case, you could have two data sources that are structurally identical (that is, they have the same columns in the same order), and all that you have to do is add one to another to end up with a query that outputs the amalgamated content of the two sources. This is called *appending data*, and it is easy, provided that the two data sources have *identical* structures; this means

- They have the same number of columns.
- The columns are in the same order.
- The data types are identical for each column.
- The columns have the same names.

As long as all these conditions are met, you can append the output of queries (which Power BI Desktop also calls *Tables* and many people, including me, refer to as datasets) one into another. The queries do not have to have data that comes from identical source types, so you can append the output from a CSV file to data that comes from an Oracle database, for instance. As an example, we will take two text files and use them to create one single output:

1. Create queries to load each of the following text files into Power BI Desktop—without the final load step, which would output them to Excel or the Excel data model. Both files are in the C:\PowerBiDesktopSamples \CH08\MultipleIdenticalFiles folder:
   a. Colors_01.txt
   b. Colors_02.txt
2. Name the queries **Colors_01** and **Colors_02**.
3. Open one of the queries (I use Colors_01, but either will do).
4. Click the Append Queries button in the Power BI Desktop Query Editor Home ribbon. The Append dialog will appear.
5. From the Select Table To Append popup, choose the query Colors_02.
6. Click OK.

The data from the two output tables is placed in the current query. You can now continue with any modifications that you need to apply. You will notice that the column names are not repeated as part of the data when the tables are appended one to the other.

Adding Multiple Files from a Source Folder

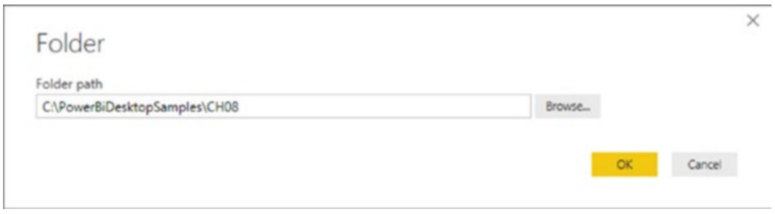
Now let's consider another possibility. You have been sent a load of files, possibly downloaded from an FTP site or received by e-mail, and you have placed them all into a specific directory. However, you do not want to have to carry out the process that you just saw and load files one by one if there are several hundred files. Another use for this approach is when you have to load files stored in multiple separate subdirectories. So here is a way to get Power BI Desktop to do the work of trawling through the directory and *only* loading files that correspond to a file name specification you have indicated. This approach extends the technique that you saw in Chapter 2, where you learned to load all the files in a directory.

1. Create a new Power BI Desktop file.

2. In the Power BI Desktop Home ribbon, click Get Data ➤ File in the Get Data dialog. Then select Folder to the right of the dialog.

3. Click Connect. The Folder dialog is displayed.

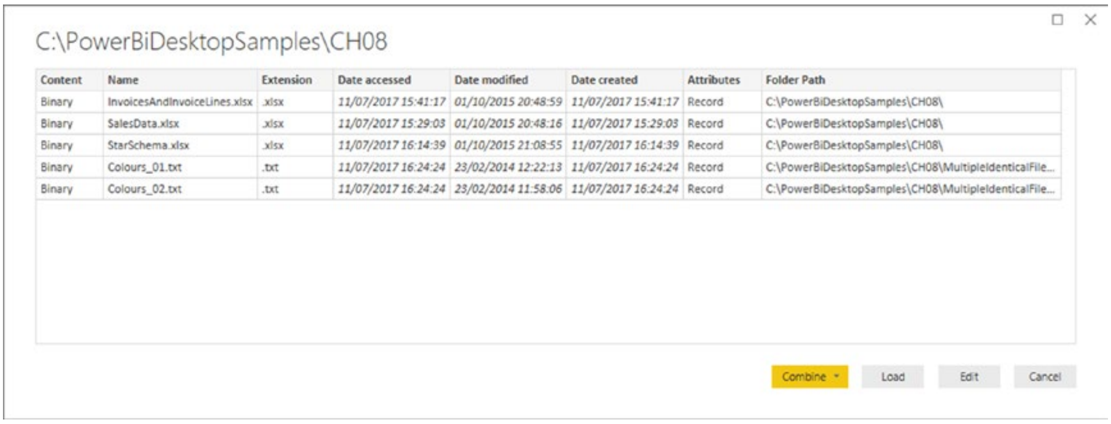
4. Click the Browse button and navigate to the folder that contains the files to load. In this example, it is C:\PowerBiDesktopSamples\CH08. You can also paste in, or enter, the folder path if you prefer. The Folder dialog will look like Figure 8-10.


Folder

Folder path: C:\PowerBiDesktopSamples\CH08

***Figure 8-10.** The Folder dialog*

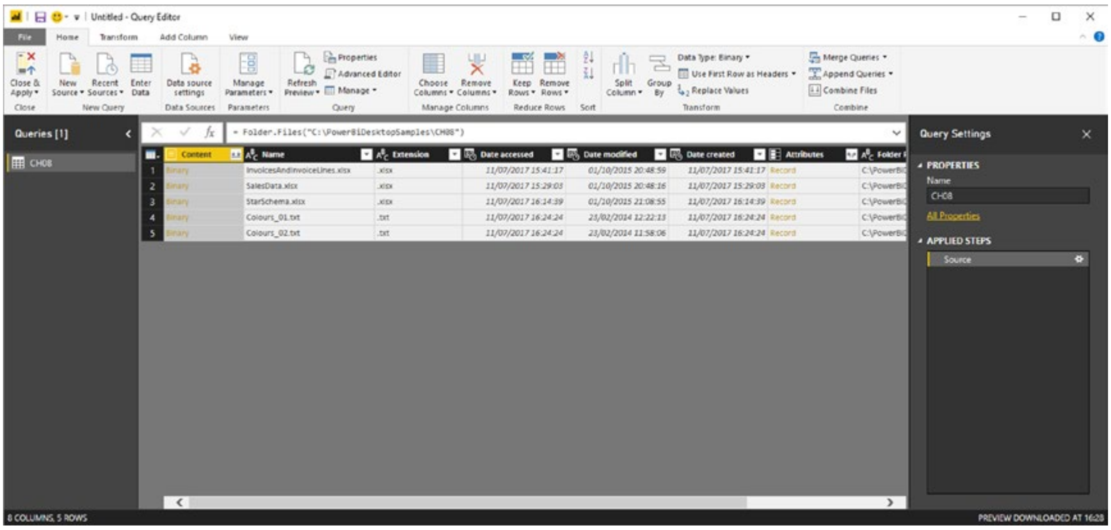
5. Click OK. The Power BI Desktop window opens. The contents of the folder and all subfolders are listed as a table, as shown in Figure 8-11.


C:\PowerBiDesktopSamples\CH08

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
Binary	InvoicesAndInvoiceLines.xlsx	.xlsx	11/07/2017 15:41:17	01/10/2015 20:48:59	11/07/2017 15:41:17	Record	C:\PowerBiDesktopSamples\CH08\
Binary	SalesData.xlsx	.xlsx	11/07/2017 15:29:03	01/10/2015 20:48:16	11/07/2017 15:29:03	Record	C:\PowerBiDesktopSamples\CH08\
Binary	StarSchema.xlsx	.xlsx	11/07/2017 16:14:39	01/10/2015 21:08:55	11/07/2017 16:14:39	Record	C:\PowerBiDesktopSamples\CH08\
Binary	Colours_01.txt	.txt	11/07/2017 16:24:24	23/02/2014 12:22:13	11/07/2017 16:24:24	Record	C:\PowerBiDesktopSamples\CH08\MultipleIdenticalFile...
Binary	Colours_02.txt	.txt	11/07/2017 16:24:24	23/02/2014 11:58:06	11/07/2017 16:24:24	Record	C:\PowerBiDesktopSamples\CH08\MultipleIdenticalFile...

***Figure 8-11.** The folder contents in Power BI Desktop*

6. Click Edit. The Power BI Desktop Query Editor will display the list of files. This is shown in Figure 8-12.



```
= Folder.Files("C:\PowerBiDesktopSamples\CH08")
```

	Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder
1	Binary	InvoicesAndInvoiceLines.xlsx	.xlsx	11/07/2017 15:41:17	01/10/2015 20:48:59	11/07/2017 15:41:17	Record	C:\PowerBI
2	Binary	SalesData.xlsx	.xlsx	11/07/2017 15:29:03	01/10/2015 20:48:16	11/07/2017 15:29:03	Record	C:\PowerBI
3	Binary	StarSchema.xlsx	.xlsx	11/07/2017 16:14:39	01/10/2015 21:08:55	11/07/2017 16:14:39	Record	C:\PowerBI
4	Binary	Colours_01.txt	.txt	11/07/2017 16:24:24	23/02/2014 12:22:13	11/07/2017 16:24:24	Record	C:\PowerBI
5	Binary	Colours_02.txt	.txt	11/07/2017 16:24:24	23/02/2014 11:58:06	11/07/2017 16:24:24	Record	C:\PowerBI

Figure 8-12. *Files listed in the Power BI Desktop Query Editor*

7. As you want to load only text files, and avoid files of any other type, click the filter popup menu for the column title Extension and uncheck all elements *except* .txt. This is shown in Figure 8-13.

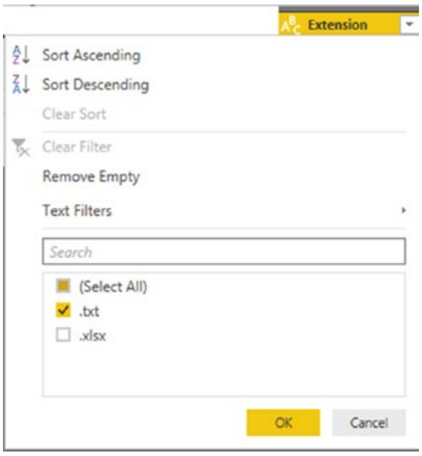
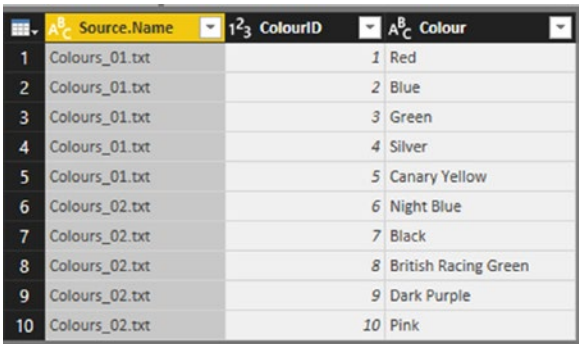


Figure 8-13. *Filtering file types when loading multiple identical files*

8. Click OK. You will now only see the text files in the Query Editor.
9. Click the Expand icon (two downward-facing arrows) to the right of the first column title; this column is called Content and every row in the column contains the word *Binary*. Power BI will display the Combine Files dialog that you saw previously in Chapter 2.

10. Click OK. Power BI Desktop Query Editor will load all the files and display the result, as shown in Figure 8-14.



	Source.Name	ColourID	Colour
1	Colours_01.txt	1	Red
2	Colours_01.txt	2	Blue
3	Colours_01.txt	3	Green
4	Colours_01.txt	4	Silver
5	Colours_01.txt	5	Canary Yellow
6	Colours_02.txt	6	Night Blue
7	Colours_02.txt	7	Black
8	Colours_02.txt	8	British Racing Green
9	Colours_02.txt	9	Dark Purple
10	Colours_02.txt	10	Pink

Figure 8-14. *All files loaded from a folder*

The contents of all the source files are now loaded into the Power BI Desktop Query Editor and can be transformed and used like any other dataset. This might involve removing superfluous header rows (as described in the next section). What is more, if ever you add more files to the source directory, and then click Refresh in the Home ribbon, *all* the source files are reloaded, including any new files added to the directory.

■ **Note** You can also filter on directories, dates, or any of the file information that is displayed. Simply apply the filtering techniques that you learned in Chapter 6.

Removing Header Rows After Multiple File Loads

If the source files contained header rows, here is a practical way to remove them—fast—from the data:

1. If (but only if) each file contains header rows, then scroll down through the resulting table until you find a title element. In this example, it is the word *ColourID* in the ColourID column.

2. Right-click ColourID and select Text Filters ➤ Does Not Equal. All rows containing superfluous column titles are removed.

■ **Note** If your source directory only contains the files that you want to load, then step 4 is unnecessary. Nonetheless, I always add steps like this in case files of the "wrong" type are added later, which would cause any subsequent process runs to fail. Equally, you can set filters on the file name to restrict the files that are loaded.

Changing the Data Structure

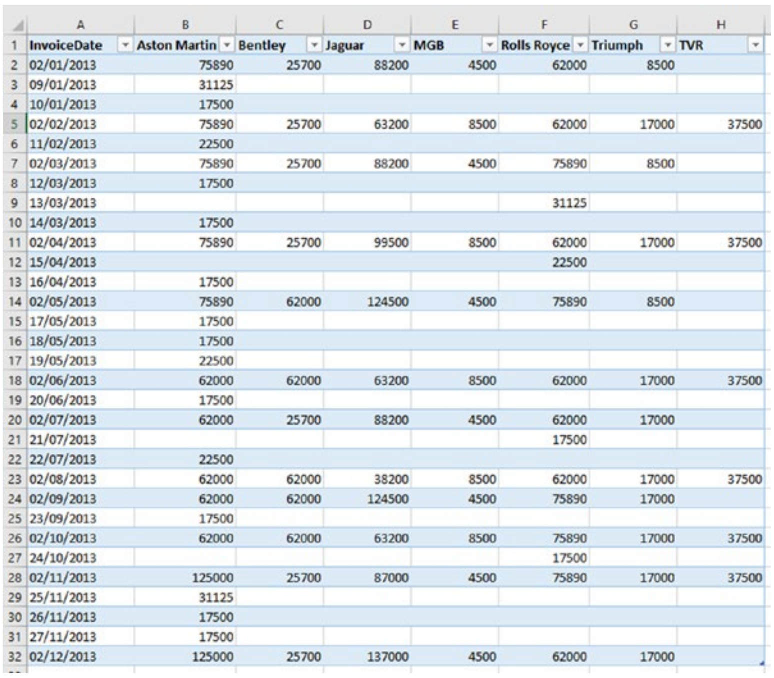
Sometimes your requirements go beyond the techniques that we have seen so far when discussing data cleansing and transformation. Some data structures need more radical reworking, given the shape of the data that you have acquired. I include in this category the following:

- Unpivoting data
- Pivoting data
- Transforming rows and columns

Each of these techniques is designed to meet a specific, yet frequent, need in data loading, and all are described in the next few pages.

Unpivoting Tables

From time to time, you may need to analyze data that has been delivered in a "pivoted" or "denormalized" format. Essentially, this means that information that really should be in a single column has been broken down and placed across several columns. An example of the first few rows of a pivoted dataset is given in Figure 8-15 and can be found in the C:\PowerBiDesktopSamples\CH08\PivotedDataSet.xlsx sample file.



	A	B	C	D	E	F	G	H
1	InvoiceDate	Aston Martin	Bentley	Jaguar	MGB	Rolls Royce	Triumph	TVR
2	02/01/2013	75890	25700	88200	4500	62000	8500	
3	09/01/2013	31125						
4	10/01/2013	17500						
5	02/02/2013	75890	25700	63200	8500	62000	17000	37500
6	11/02/2013	22500						
7	02/03/2013	75890	25700	88200	4500	75890	8500	
8	12/03/2013	17500						
9	13/03/2013					31125		
10	14/03/2013	17500						
11	02/04/2013	75890	25700	99500	8500	62000	17000	37500
12	15/04/2013					22500		
13	16/04/2013	17500						
14	02/05/2013	75890	62000	124500	4500	75890	8500	
15	17/05/2013	17500						
16	18/05/2013	17500						
17	19/05/2013	22500						
18	02/06/2013	62000	62000	63200	8500	62000	17000	37500
19	20/06/2013	17500						
20	02/07/2013	62000	25700	88200	4500	62000	17000	
21	21/07/2013					17500		
22	22/07/2013	22500						
23	02/08/2013	62000	62000	38200	8500	62000	17000	37500
24	02/09/2013	62000	62000	124500	4500	75890	17000	
25	23/09/2013	17500						
26	02/10/2013	62000	62000	63200	8500	75890	17000	37500
27	24/10/2013					17500		
28	02/11/2013	125000	25700	87000	4500	75890	17000	37500
29	25/11/2013	31125						
30	26/11/2013	17500						
31	27/11/2013	17500						
32	02/12/2013	125000	25700	137000	4500	62000	17000	

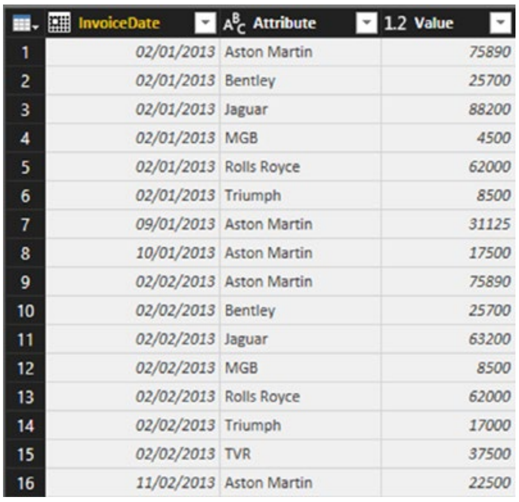
Figure 8-15. *A pivoted dataset*

To analyze this data correctly, we really need the makes of the cars to be switched from being column titles to becoming the contents of a specific column. Fortunately, this is not hard at all:

1. In a new Power BI Desktop file, load the table PivotedCosts from the C:\PowerBiDesktopSamples\CH08\PivotedDataSet.xlsx file into Power BI Desktop. Ensure that the first row is set to be the table headers.

2. Switch to the Query Editor (unless it is already open) and select all the columns that you want to unpivot. In this example, this means all columns except the first one.

3. In the Transform ribbon, click the Unpivot Columns button (or right-click with the columns selected and choose Unpivot Columns from the context menu). The table is reorganized and the first few records look as they do in Figure 8-16. Unpivoted Columns is added to the Applied Steps list.



	InvoiceDate	Attribute	1.2 Value
1	02/01/2013	Aston Martin	75890
2	02/01/2013	Bentley	25700
3	02/01/2013	Jaguar	88200
4	02/01/2013	MGB	4500
5	02/01/2013	Rolls Royce	62000
6	02/01/2013	Triumph	8500
7	09/01/2013	Aston Martin	31125
8	10/01/2013	Aston Martin	17500
9	02/02/2013	Aston Martin	75890
10	02/02/2013	Bentley	25700
11	02/02/2013	Jaguar	63200
12	02/02/2013	MGB	8500
13	02/02/2013	Rolls Royce	62000
14	02/02/2013	Triumph	17000
15	02/02/2013	TVR	37500
16	11/02/2013	Aston Martin	22500

***Figure 8-16.** An unpivoted dataset*

4. Rename the columns that Power BI Desktop Query has named Attribute and Value.

The data is now presented in a standard tabular way, and so it can be used to create a data model and then produce reports and dashboards.

Unpivot Options

There are a couple of available options when you unpivot data using the Unpivot Columns button popup in the Transform ribbon:

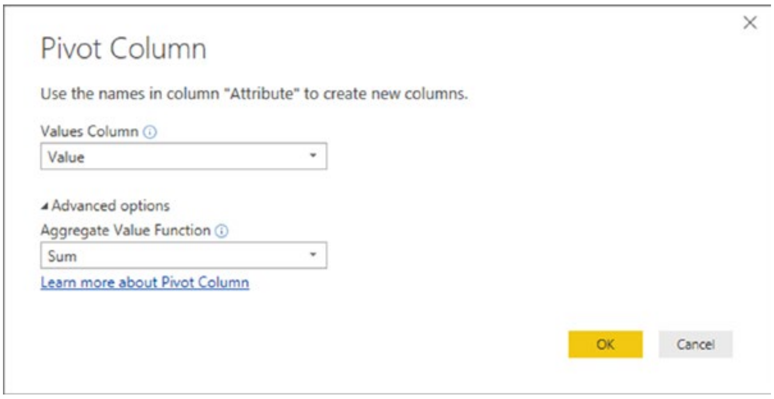
- *Unpivot Other Columns*: This will add the contents of all the other columns to the unpivoted output.

- *Unpivot Only Selected Columns*: This will only add the contents of any preselected columns to the unpivoted output.

Pivoting Tables

On some occasions, you may have to switch data from columns to rows so that you can use it efficiently. This kind of operation is called *pivoting data*. It is—perhaps unsurprisingly—very similar to the unpivot process that you saw in the previous section.

1. Follow steps 1 through 3 of the previous section so that you end up with the table of data that you can see in Figure 8-16.
2. Click inside the column Attribute.
3. In the Transform ribbon, click the Pivot Column button. The Pivot Column dialog will appear.
4. Select Value (the column of figures) as the values column that is aggregated by the pivot transformation. The Pivot Column dialog will look like Figure 8-17.


Pivot Column

Use the names in column "Attribute" to create new columns.

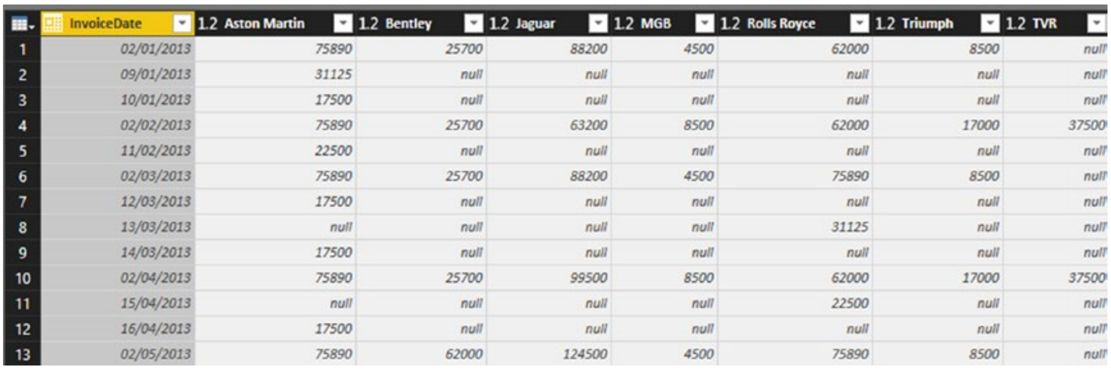
Values Column: Value

Advanced options

Aggregate Value Function: Sum

Figure 8-17. *The Pivot Column dialog*

5. Click OK. The table is pivoted and looks like Figure 8-18. Pivoted Column is added to the Applied Steps list.



	InvoiceDate	1.2 Aston Martin	1.2 Bentley	1.2 Jaguar	1.2 MGB	1.2 Rolls Royce	1.2 Triumph	1.2 TVR
1	02/01/2013	75890	25700	88200	4500	62000	8500	null
2	09/01/2013	31125	null	null	null	null	null	null
3	10/01/2013	17500	null	null	null	null	null	null
4	02/02/2013	75890	25700	63200	8500	62000	17000	37500
5	11/02/2013	22500	null	null	null	null	null	null
6	02/03/2013	75890	25700	88200	4500	75890	8500	null
7	12/03/2013	17500	null	null	null	null	null	null
8	13/03/2013	null	null	null	null	31125	null	null
9	14/03/2013	17500	null	null	null	null	null	null
10	02/04/2013	75890	25700	99500	8500	62000	17000	37500
11	15/04/2013	null	null	null	null	22500	null	null
12	16/04/2013	17500	null	null	null	null	null	null
13	02/05/2013	75890	62000	124500	4500	75890	8500	null

Figure 8-18. *Pivoted data*

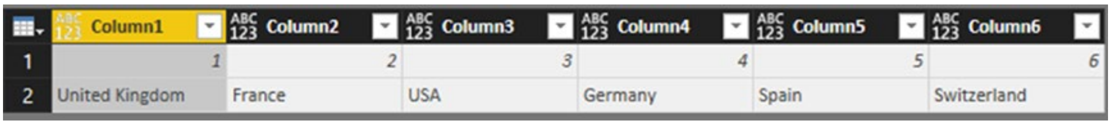
■ **Note** The Advanced Options section of the Pivot Column dialog lets you choose the aggregation operation that is applied to the values in the pivoted table.

The Unpivot button contains another menu option that is displayed if you click the small triangle to the right of the Unpivot button. This is the Unpivot Other Columns option that will switch the contents of columns into rows for all the columns that are not selected when you run the transformation.

Transposing Rows and Columns

On some occasions, you may have a source table where the columns need to become rows and the rows columns. Fortunately, this is a one-click transformation for Power BI Desktop. Here is how to do it:

1. Edit the C:\PowerBiDesktopSamples\CH08\DataToTranspose.xlsx Excel file in the Power BI Desktop Query Editor. You will need to select Sheet1. You will see a data table like the one in Figure 8-19.



	Column1	Column2	Column3	Column4	Column5	Column6
1	1	2	3	4	5	6
2	United Kingdom	France	USA	Germany	Spain	Switzerland

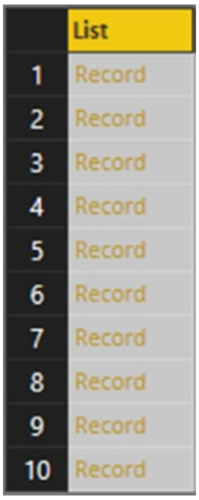
Figure 8-19. *A dataset needing to be transposed*

2. In the Transform ribbon, click the Transpose button. The data is transposed and appears as two columns, just like the CountryList.txt file that you saw in Chapter 2.
3. Rename the columns.

Parsing JSON Files

In Chapter 2 you saw how to import JSON files. Previously, however, you only saw how to load the data into Power BI Desktop, and the file contents remained pretty much unusable. So now is the time to explain how you can use the Power BI Desktop Query Editor to convert the contents of a JSON file to a state that you can use in your data mashup.

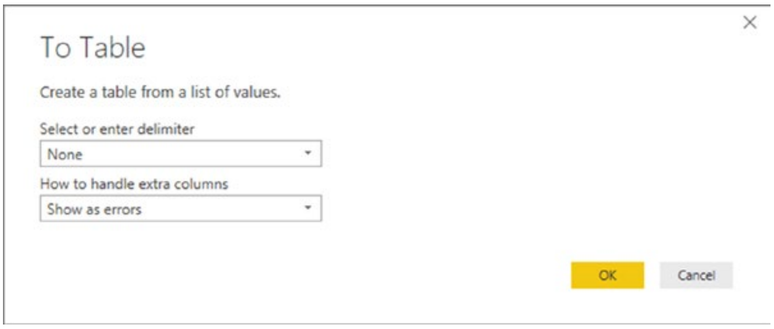
1. In the Home ribbon, click Get Data ➤ File ➤ JSON, then click Connect.
2. Select the file C:\PowerBiDesktopSamples\CH08\Colors.json, and click Open. You will see a list of records like the one shown in Figure 8-20.



	List
1	Record
2	Record
3	Record
4	Record
5	Record
6	Record
7	Record
8	Record
9	Record
10	Record

Figure 8-20. *A JSON file after initial import*

3. You will see that the Query Editor has added the List Tools Transform ribbon. Click the To Table button in this ribbon. The To Table dialog will appear, as shown in Figure 8-21.



To Table

Create a table from a list of values.

Select or enter delimiter

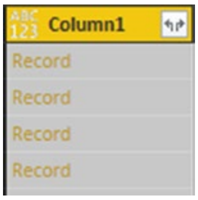
None

How to handle extra columns

Show as errors

Figure 8-21. *The To Table dialog*

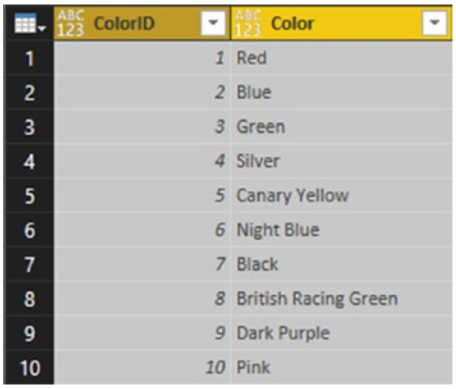
4. Click OK. The list of data will be converted to a table. This means that it now shows the Expand icon at the right of the column title, as you can see in Figure 8-22.



Column1
Record
Record
Record
Record

Figure 8-22. *A JSON file converted to a table*

5. Click the Expand icon to the right of the column title, and in the popup dialog uncheck "Use original column name as prefix."
6. Click OK. The contents of the JSON file now appear as a standard dataset, as you can see in Figure 8-23.



	ColorID	Color
1	1	Red
2	2	Blue
3	3	Green
4	4	Silver
5	5	Canary Yellow
6	6	Night Blue
7	7	Black
8	8	British Racing Green
9	9	Dark Purple
10	10	Pink

Figure 8-23. *A JSON file transformed into a dataset*

Although not particularly difficult, this process may seem a little counterintuitive. However, it certainly works, and you can use it to process complex JSON files so that you can use the data they contain in Power BI Desktop.

The List Tools Transform Ribbon

Power BI Desktop considers some data to be lists, not tables of data. It handles lists slightly differently, and displays a specific ribbon to modify list data. The List Tools Transform ribbon is explained in Figure 8-24 and Table 8-4.

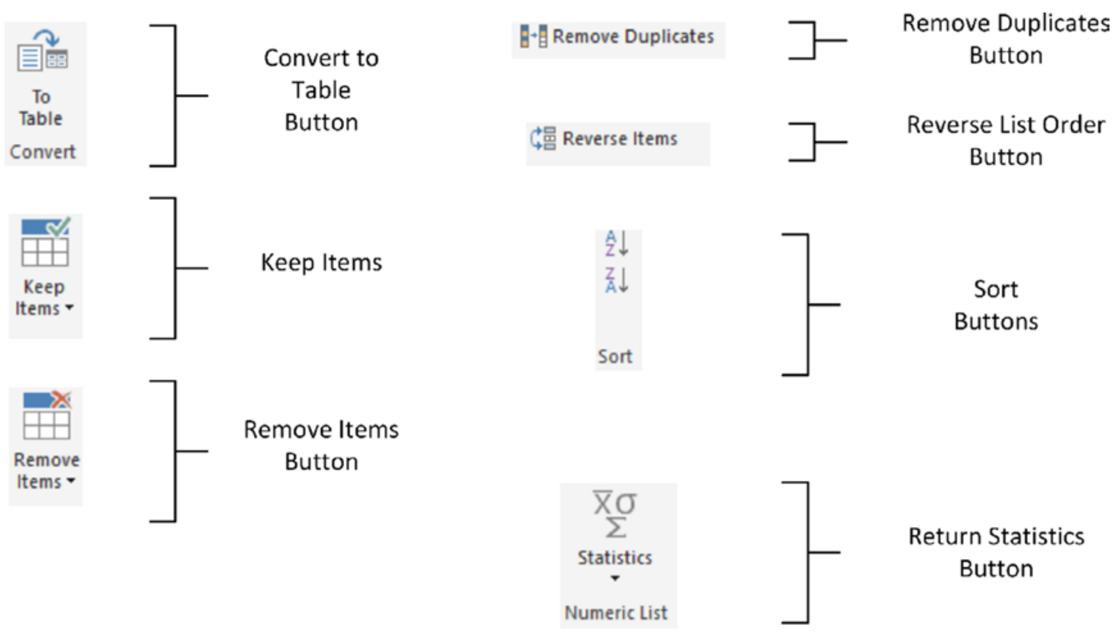


Figure 8-24. *The List Tools Transform ribbon*

Table 8-4. *The List Tools Transform Ribbon Options*

Option	Description
To Table	Converts the list to a table structure.
Keep Items	Allows you to keep a number of items from the top or bottom of the list, or a range of items from the list.
Remove Items	Allows you to remove a number of items from the top or bottom of the list, or a range of items from the list.
Remove Duplicates	Removes any duplicates from the list.
Reverse Items	Reverses the list order.
Sort	Sorts the list lowest to highest or highest to lowest.
Statistics	Returns calculated statistics about the elements in the list.

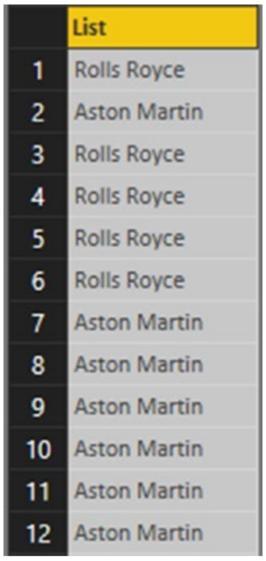
Convert a Column to a List

Sometimes you will need to use data in a list format. Fortunately, Power BI Desktop lets you convert a column to a list really easily:

1. Open the Power BI Desktop file C:\PowerBiDesktopSamples\CH08\CH08Example1.pbix.
2. In the Home ribbon, click Edit Queries to open the Query Editor.

3. Select a column to convert to a list by clicking the column header. I will use the column Make in this example.

4. In the Transform ribbon, click Convert to List. The Query Editor will show the resulting list, as you can see in Figure 8-25.



	List
1	Rolls Royce
2	Aston Martin
3	Rolls Royce
4	Rolls Royce
5	Rolls Royce
6	Rolls Royce
7	Aston Martin
8	Aston Martin
9	Aston Martin
10	Aston Martin
11	Aston Martin
12	Aston Martin

Figure 8-25. *The list resulting from a conversion-to-list operation*

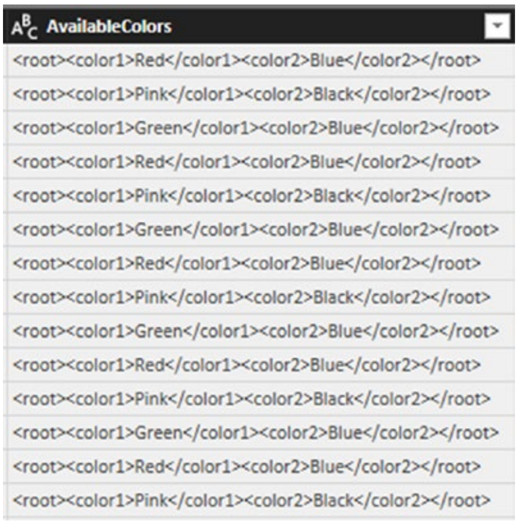
Parsing XML Data from a Column

Some data sources, particularly database sources, include XML data actually inside a field. The problem here is that XML data is interpreted as plain text by Power BI Desktop when the data is loaded. If you look at Figure 8-26 you can see that this is not particularly useful.

So once again, Power BI Desktop has a solution to this kind of issue. To demonstrate how to convert this kind of text into usable data, you will find a sample Excel file (C:\PowerBiDesktopSamples\CH08\XMLInColumn.xlsx) that contains XML data as a column. Proceed as follows:

1. Edit the Excel file C:\PowerBiDesktopSamples\CH08\XMLInColumn.xlsx in the Query Editor. Select the only worksheet in this file: Sales.

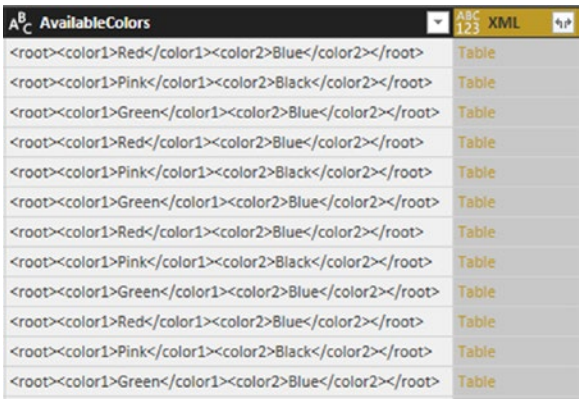
2. Scroll to the right of the dataset and select the last column: AvailableColors. It looks like Figure 8-26.



AvailableColors
<root><color1>Red</color1><color2>Blue</color2></root>
<root><color1>Pink</color1><color2>Black</color2></root>
<root><color1>Green</color1><color2>Blue</color2></root>
<root><color1>Red</color1><color2>Blue</color2></root>
<root><color1>Pink</color1><color2>Black</color2></root>
<root><color1>Green</color1><color2>Blue</color2></root>
<root><color1>Red</color1><color2>Blue</color2></root>
<root><color1>Pink</color1><color2>Black</color2></root>
<root><color1>Green</color1><color2>Blue</color2></root>
<root><color1>Red</color1><color2>Blue</color2></root>
<root><color1>Pink</color1><color2>Black</color2></root>
<root><color1>Green</color1><color2>Blue</color2></root>
<root><color1>Red</color1><color2>Blue</color2></root>
<root><color1>Pink</color1><color2>Black</color2></root>

Figure 8-26. *A column containing XML*

3. In the Add Column ribbon, click Parse ➤ XML. A new column will be added to the right. It will look like Figure 8-27, and will have the title XML.

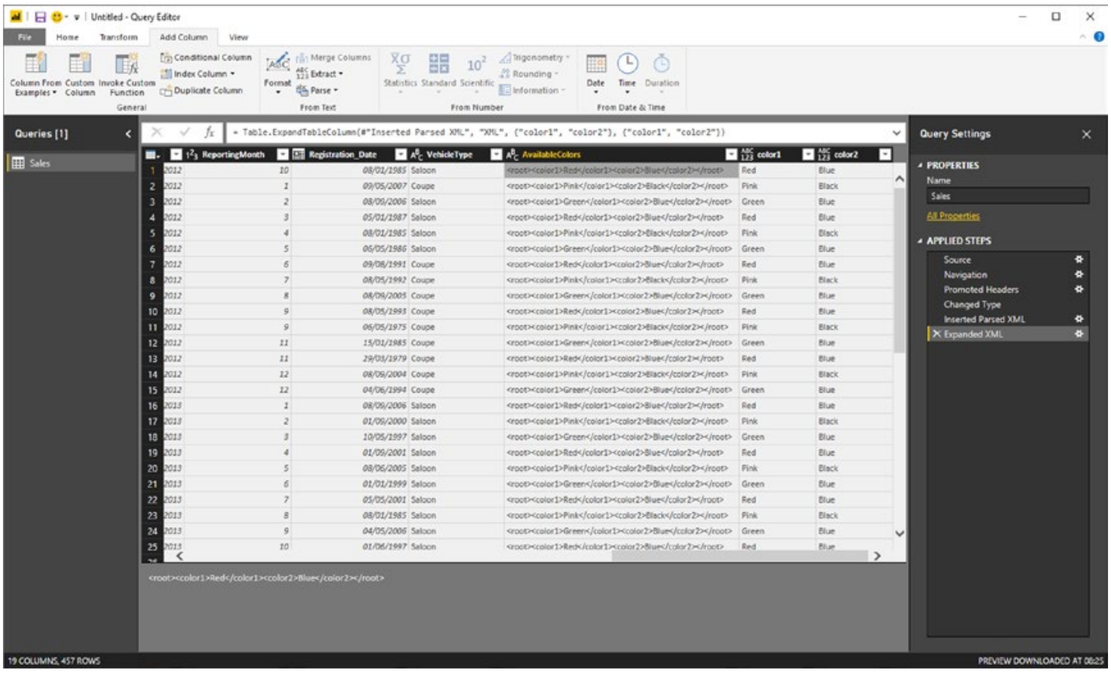


AvailableColors	XML
<root><color1>Red</color1><color2>Blue</color2></root>	Table
<root><color1>Pink</color1><color2>Black</color2></root>	Table
<root><color1>Green</color1><color2>Blue</color2></root>	Table
<root><color1>Red</color1><color2>Blue</color2></root>	Table
<root><color1>Pink</color1><color2>Black</color2></root>	Table
<root><color1>Green</color1><color2>Blue</color2></root>	Table
<root><color1>Red</color1><color2>Blue</color2></root>	Table
<root><color1>Pink</color1><color2>Black</color2></root>	Table
<root><color1>Green</color1><color2>Blue</color2></root>	Table
<root><color1>Red</color1><color2>Blue</color2></root>	Table
<root><color1>Pink</color1><color2>Black</color2></root>	Table
<root><color1>Green</color1><color2>Blue</color2></root>	Table

Figure 8-27. *An XML column converted to a table column*

4. Click the Expand icon to the right of the XML column title and uncheck “Use original column name as prefix” in the popup dialog. Ensure that all the columns are selected and click OK. Two new columns (or, indeed, as many new columns as there are XML data elements) will appear at the right of the dataset. The Query Editor will look like Figure 8-28.

5. Delete the column containing the initial XML data.



```
= Table.ExpandTableColumn(#"Inserted Parsed XML", "XML", {"color1", "color2"}, {"color1", "color2"})
```

		ReportingMonth	Registration_Date	VehicleType	AvailableColors	color1	color2
1	2012	10	08/01/1985	Saloon	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue
2	2012	1	09/05/2007	Coupe	<root><color1>Pink</color1><color2>Black</color2></root>	Pink	Black
3	2012	2	08/09/2006	Saloon	<root><color1>Green</color1><color2>Blue</color2></root>	Green	Blue
4	2012	3	05/01/1987	Saloon	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue
5	2012	4	08/01/1985	Saloon	<root><color1>Pink</color1><color2>Black</color2></root>	Pink	Black
6	2012	5	06/05/1986	Saloon	<root><color1>Green</color1><color2>Blue</color2></root>	Green	Blue
7	2012	6	09/08/1991	Coupe	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue
8	2012	7	08/05/1992	Coupe	<root><color1>Pink</color1><color2>Black</color2></root>	Pink	Black
9	2012	8	08/09/2005	Coupe	<root><color1>Green</color1><color2>Blue</color2></root>	Green	Blue
10	2012	9	08/05/1993	Coupe	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue
11	2012	9	06/05/1975	Coupe	<root><color1>Pink</color1><color2>Black</color2></root>	Pink	Black
12	2012	11	15/01/1985	Coupe	<root><color1>Green</color1><color2>Blue</color2></root>	Green	Blue
13	2012	11	29/03/1979	Coupe	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue
14	2012	12	08/09/2004	Coupe	<root><color1>Pink</color1><color2>Black</color2></root>	Pink	Black
15	2012	12	04/06/1994	Coupe	<root><color1>Green</color1><color2>Blue</color2></root>	Green	Blue
16	2013	1	08/09/2006	Saloon	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue
17	2013	2	01/09/2000	Saloon	<root><color1>Pink</color1><color2>Black</color2></root>	Pink	Black
18	2013	3	10/05/1997	Saloon	<root><color1>Green</color1><color2>Blue</color2></root>	Green	Blue
19	2013	4	01/09/2001	Saloon	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue
20	2013	5	08/06/2005	Saloon	<root><color1>Pink</color1><color2>Black</color2></root>	Pink	Black
21	2013	6	01/01/1999	Saloon	<root><color1>Green</color1><color2>Blue</color2></root>	Green	Blue
22	2013	7	05/05/2001	Saloon	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue
23	2013	8	08/01/1985	Saloon	<root><color1>Pink</color1><color2>Black</color2></root>	Pink	Black
24	2013	9	04/05/2006	Saloon	<root><color1>Green</color1><color2>Blue</color2></root>	Green	Blue
25	2013	10	01/06/1997	Saloon	<root><color1>Red</color1><color2>Blue</color2></root>	Red	Blue

```
<root><color1>Red</color1><color2>Blue</color2></root>
```

Name: Sales

Applied Steps:
- Source
- Navigation
- Promoted Headers
- Changed Type
- Inserted Parsed XML
- Expanded XML

19 COLUMNS, 457 ROWS

PREVIEW DOWNLOADED AT 08:25

***Figure 8-28.** XML data expanded into new columns*

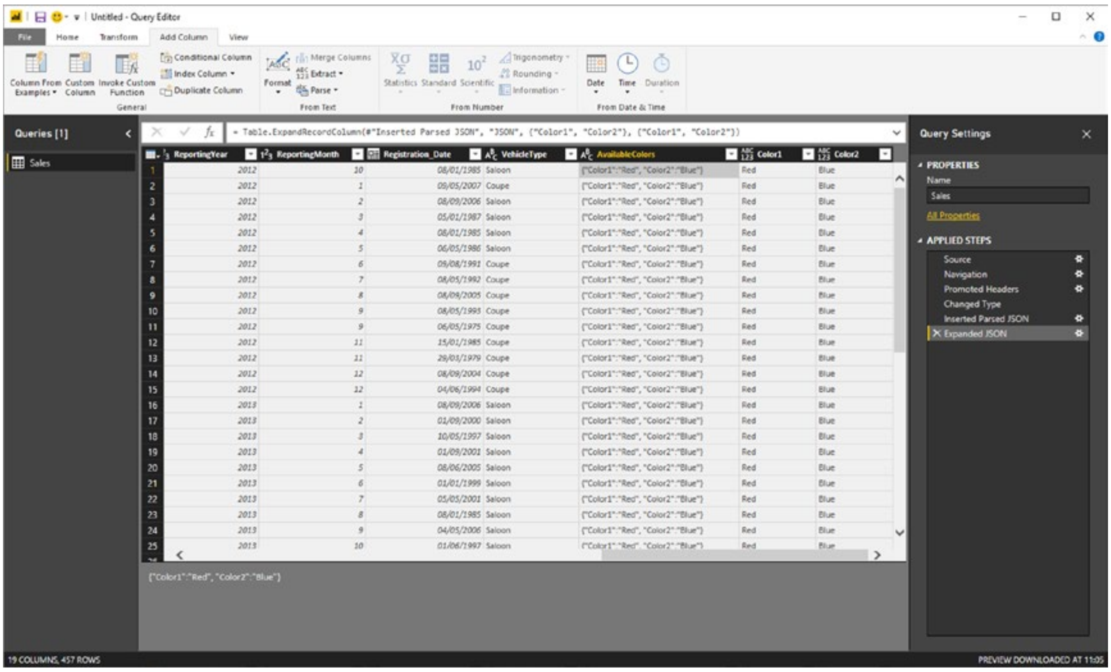
Using this technique, you can now extract the XML data that is embedded (or “nested” if you prefer) in source datasets and use it to extend the original source data.

Parsing JSON Data from a Column

Sometimes you may encounter data containing JSON in a field, too. The technique to extract this data from the field inside the dataset and convert it to columns is virtually identical to the approach that you saw in the previous section for XML data.

Given that the approach is so similar, I will only provide a screenshot for the final result of the process. Here you will be able to see the source JSON as well as the columns of data that were extracted from the JSON and added to the dataset.

1. Edit the Excel file C:\PowerBiDesktopSamples\CH08\JSONInColumn.xlsx in the Query Editor. Select the only worksheet in this file: Sales.
2. Scroll to the right of the dataset and select the last column: AvailableColors.
3. In the Add Column ribbon, click Parse ➤ JSON. A new column will be added to the right and will have the title JSON.
4. Click the Expand icon to the right of the JSON column title and uncheck "Use original column name as prefix" in the popup dialog. Ensure that all the columns are selected and click OK. Two new columns (or, indeed, as many new columns as there are JSON data elements) will appear at the right of the dataset. The Query Editor will look like Figure 8-29.



```
= Table.ExpandRecordColumn(#"Inserted Parsed JSON", "JSON", {"Color1", "Color2"}, {"Color1", "Color2"})
```

	ReportingYear	ReportingMonth	Registration_Date	VehicleType	AvailableColors	Color1	Color2
1	2012	10	08/01/1985	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
2	2012	1	09/05/2007	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
3	2012	2	08/09/2006	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
4	2012	3	05/01/1987	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
5	2012	4	08/01/1985	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
6	2012	5	06/05/1986	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
7	2012	6	09/08/1991	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
8	2012	7	08/05/1992	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
9	2012	8	08/09/2005	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
10	2012	9	08/05/1993	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
11	2012	9	06/05/1975	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
12	2012	11	15/01/1985	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
13	2012	11	29/03/1979	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
14	2012	12	08/09/2004	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
15	2012	12	04/06/1994	Coupe	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
16	2013	1	08/09/2006	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
17	2013	2	01/09/2000	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
18	2013	3	10/05/1997	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
19	2013	4	01/09/2001	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
20	2013	5	08/06/2005	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
21	2013	6	01/01/1999	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
22	2013	7	05/05/2001	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
23	2013	8	08/01/1985	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
24	2013	9	04/05/2006	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue
25	2013	10	01/06/1997	Saloon	{"Color1":"Red", "Color2":"Blue"}	Red	Blue

{"Color1":"Red", "Color2":"Blue"}

Figure 8-29. *JSON data expanded into new columns*

5. Delete the column containing the initial JSON data.

Admittedly, the structure of the JSON data in this example is extremely simple. Real-world JSON data could be much more complex. However, you now have a starting point upon which you can build when parsing JSON data that is stored in a column of a dataset.

Conclusion

This chapter showed you how to structure your source data into a valid data table from one or more potential sources. You saw how to pivot and unpivot data, to fill rows up and down with data, as well as how to create new columns of data from columns containing XML or JSON data.

Possibly the most important thing that you have learned is how to join individual queries so that you can add the data from one query into another. This can involve looking up data from a separate query or carrying the aggregated results from one query into another.

In this chapter and the six previous chapters, you have seen essentially a three-stage process: first, you find the data, then you load it into Power BI Desktop Query, and from there, you cleanse and modify it. The techniques that you can use are simple but powerful and can range from changing a data type to merging multiple data tables. Now that your data is prepared and ready for use, you can add it to the Power BI Desktop data model and start creating your Power BI Desktop dashboards.

CHAPTER 9

Structuring, Managing, and Parameterizing Queries

Producing an efficient data query is not just about finding the appropriate load and transform functions and placing them in the correct sequence. It is also about extending, adjusting, and maintaining the process. This can be either to correct an error once the query is being tested or to adapt a query to new requirements. This chapter will introduce you to some of the techniques that you can apply to manage and extend existing queries.

Creating an eye-catching dashboard can mean sourcing data from a large range and variety of queries. It may also imply that these queries have to be linked together to create a cascade of data transformations that prepares the core elements of a practical and usable data model collated from multiple sources. It follows that you will therefore need to know how to manage the queries that you create to use them efficiently.

However, not all data flows are rigid and predictable. There will, inevitably, be cases where you want to shape the data ingestion process depending on aspects of the source data. This can mean parameterizing your queries to allow user interaction or adjusting the data flow dynamically.

To give you an idea of some of the techniques that you might need to keep your queries under control in real-world situations, we will end this chapter with a short overview of query management. The final flourish will consist of showing you how you can *export* data from Power BI Desktop Query Editor.

Managing the Transformation Process

Pretty nearly all the transformation steps that we have applied so far have been individual elements that can be applied to just about any data table. However, when you are carrying out even a simple data load and transform process, you are likely to want to step through several transformations in order to shape, cleanse, and filter the data to get the result you want. This is where the Power BI Desktop approach is so clever, because you can apply most data transformation steps to just about any data table. The art consists of placing them in a sequence that can then be reused any time that the data changes to reprocess the new source data and deliver an up-to-date output.

The key to appreciating and managing this process is to get well acquainted with the Applied Steps list in the Query Settings pane. This list contains the details of every step that you applied, in the order in which you applied it. Each step retains the name that Power BI Desktop gave it when it was created, and each can be altered in the following ways:

- Renamed
- Deleted
- Moved (in certain cases)

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_9

The even better news is that, in many case, steps can be modified. This way you are not stuck with the choices that you made initially, but have the opportunity of tweaking and improving individual steps in a process. This can avoid your having to rebuild an entire sequence of steps in an ETL routine simply by replacing one element in the ETL process.

In order to experiment with the various ways that you can modify queries, you are going to need some initial data. So, to start with, I suggest that you create a query that loads data from the following Excel source file: C:\PowerBiDesktopSamples\CH09\CarSalesDataForQueries.xlsx. From this source file, select the following tables:

- Clients
- Colors
- Countries
- Invoices
- InvoiceLines
- Stock

Once you have loaded the data, switch to the Query Editor window.

Modifying a Step

How you alter a step will depend on how the original transformation was applied. This becomes second nature after a little practice and will always involve first clicking the step that you wish to modify and then applying a different modification. If you invoke a ribbon option, such as altering the data type, for instance, then you change the data type by simply applying another data type directly from the ribbon. If you used an option that displayed a dialog (such as splitting a column, among others), then you can right-click the step in the Applied Steps list and select Edit Settings from the context menu. Alternatively, and if you prefer, you can click the “gear” icon to the right of any step that was created using a dialog to display the dialog for modification. This will cause the original dialog to reappear showing all the options and settings that you applied initially; in it, you can make any modifications that you consider necessary.

A final possibility that makes it easy to alter the settings for a process is to edit the formula that appears in the formula bar each time you click a step. This, however, involves understanding all the complexities of each piece of the code that underpins the data transformation process. As a full explanation of all the subtleties of the language that Power BI Desktop uses is outside the scope of this book, I will only provide a cursory overview of code modification in this chapter.

■ **Tip** If you can force yourself to organize the process that you are writing with Power BI Desktop, then a little forethought and planning can reap major dividends. For instance, certain tasks, such as setting data types, can be carried out in a single operation. Not just that, but if you need to alter a data type for a column at a later stage, I suggest that you click the ChangedType step before you make any further alterations. This way, you extend the original step, rather than creating other steps, which can make the process more confusing and needlessly voluminous.

Renaming a Step

Power BI Desktop names steps using the name of the transformation that was applied. This means that if another similar step is applied later, Power BI Desktop uses the same name with a numeric increment. As this is not always comprehensible when reviewing a sequence of transformation steps, you may prefer to give more user-friendly names to individual steps. This is done as follows:

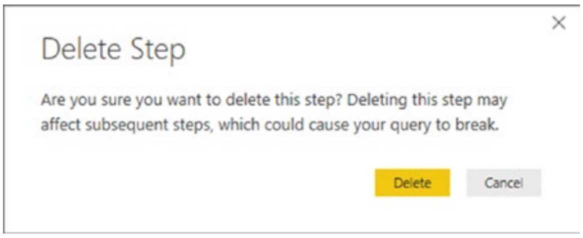
1. Select the query (or source table or worksheet, if you prefer). I will use the Clients query in this example.
2. Right-click the step that you want to rename, Changed Type for instance.
3. Select Rename from the context menu.
4. Type in the new name. I will use **NewDataTypes**.
5. Press Enter.

The step is renamed and the new name will appear in the Applied Steps list in the Query Settings pane. This way you can ensure that when you come back to a data transformation process days, weeks, or months later, you are able to understand more intuitively the process that you defined, as well as why you shaped the data like you did.

Deleting a Step or a Series of Steps

Deleting a step is all too easy, but doing so can have serious consequences. This is because an ETL process is often an extremely tightly coupled series of events, where each event depends intimately on the preceding one. So deleting a step can make every subsequent step fail. Knowing which events you can delete without drastic consequences will depend on the types of process that you are developing as well as your experience with Power BI Desktop. In any case, this is what you should do if you need to delete a step:

1. Place the pointer over the process step that you want to delete.
2. Click the cross (×) icon that appears.
3. Select Delete. The Delete Step dialog *might* appear, as shown in Figure 9-1.


Delete Step

Are you sure you want to delete this step? Deleting this step may affect subsequent steps, which could cause your query to break.

Figure 9-1. *The Delete Step dialog*

4. Confirm by clicking the Delete button. The step is deleted.

If—and it is highly possible—deleting this step causes issues for the rest of the process, you will see that the data table is replaced by an error message. This message will vary depending on the type of error that Power BI Desktop has encountered.

When describing this technique, I was careful to state that you *might* see the Delete Step dialog. If you are deleting the final step in a sequence of steps, then you will probably not see it, since there should not be any potentially horrendous consequences; at worst, you will have to re-create the step. If you are deleting a step in the middle of a process, then you might want to think seriously about doing so before you cause a potentially vast number of problems. Consequently, you are asked to confirm the deletion in these cases.

■ **Note** If you realize at this point that you have just destroyed hours of work, then (after drawing a deep breath) click the File menu in the Power BI Desktop window (the downward-facing triangle at the top left) and select Discard And Close. You will lose all work up until the last time you clicked Apply And Close, however. Don't count on using an undo function as you can in other desktop applications. To lower your blood pressure, you may prefer to save a copy of a file containing an intricate data transformation process *before* deleting any steps.

An alternative technique is to right-click the step that you want to delete and select Delete. You may still have to confirm the deletion.

If you realize that an error in a process step has invalidated all your work up until the end of the process, rather than deleting multiple elements one by one, click Delete Until End from the context menu at step 2 in the preceding exercise.

Modifying an Existing Step

Power BI Desktop does not try and lock you into a rigid sequence of events when you create a series of applied steps to create and transform a data flow. This really becomes obvious when you discover that you need to alter a step in a process.

Suppose, for instance, that you discover that you have loaded a wrong Excel worksheet when you selected the initial data from Excel. You do not want to repeat the process when you can simply substitute one worksheet name for another.

1. Select the query that you want to modify (Clients in this example)
2. Click the step to modify (in this case it will be Navigation)
3. Click the gear (or cog) icon to the right of the step name. The appropriate dialog will appear. In this case it will be the Navigation dialog that you can see in Figure 9-2.

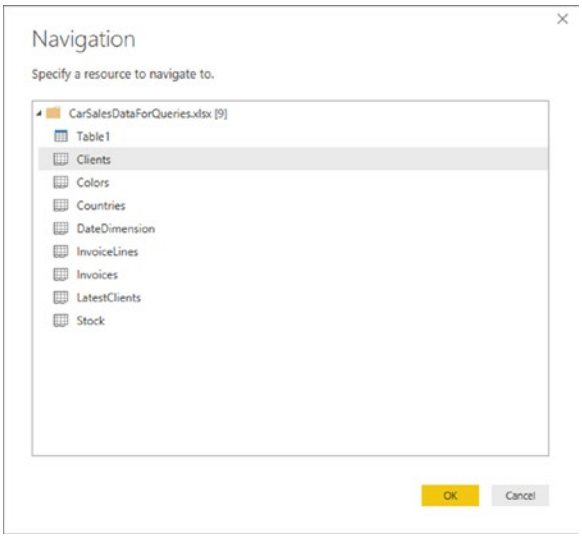


Figure 9-2. *The Navigation dialog displayed for step modification*

4. Click the table or worksheet that you want to use instead of the current dataset (LatestClients in this example).
5. Click OK.

The Query Editor will replace one source dataset with another. It might also add extra steps to ensure that the data is adapted for use in the query.

As you saw in the previous six chapters, Power BI Desktop offers a vast range of data ingestion and modification possibilities. So I cannot, here, describe every possible option as far as modifying an Applied Step is concerned. Nonetheless, the principle is simple:

- If the Query Editor can modify a step, the gear icon will be displayed to the right of the step name.
- Clicking the modification (the gear) icon will display the dialog that was used to create the step (even if the step was created automatically by Power BI Desktop).

Certain steps do not display the modification icon. This is because the step cannot be modified, only removed (at least, using the Query Editor interface). As an example of this, add the following step:

1. Select the query that you want to modify (Clients in this example).
2. Click the last step.
3. Right-click the Address2 column and select Remove.

A new step will appear in the Applied Steps list, named Removed Columns. This step does not have the modification icon. So, for the moment, you can remove it, but *not* modify it.

■ **Note** Modifying existing steps is not a "magic bullet." This is because a series of data transformations can be highly dependent on a tailored logic that has been developed for a specific data structure. It follows, for instance, that you can only replace a data source with another one that has a nearly identical structure. However, modifying a step can avoid your having to rewrite an entire data flow sequence in many cases.

Adding a Step

You can add a step anywhere in the sequence. All you have to do is click the step that *precedes* the new step that you want to insert *before* clicking the icon in any of the ribbons that corresponds to the new step. As is the case when you delete a step, Power BI Desktop will display an alert warning you that this action *could* cause problems with the process from this new step on.

Altering Process Step Sequencing

It is possible—technically—to resequence steps in a process. However, in my experience, this is not always practical, since changing the order of steps in a process can cause as much damage as deleting a step. Nonetheless, you can always try it like this:

1. Right-click the step that you want to resequence.
2. Select Move Up or Move Down from the context menu.

I remain pessimistic that this can work miracles, but it is good to know that it is there.

■ **Tip** Remember that before tweaking the order in which the process is applied, clicking any process step causes the table in the Power BI Desktop window to refresh to show you the state of the data up to and including the selected step. This is a very clear visual guide to the process and how the ETL is carried out.

An Approach to Sequencing

Given the array of available data transformation options, you may well be wondering how best to approach a new ETL project using Power BI Desktop. I realize that all projects are different, but as a rough and ready guide, I suggest attempting to order your project like this:

1. Load the data into Power BI Desktop.
2. Promote or add correct column headers. For example, you really do not want to be looking at step 47 of a process and wondering what Column29 is, when it could read (for instance) ClientName.
3. Remove any columns that you do not need. The smaller the dataset, the faster the processing. What is more, you will find it easier to concentrate on, and understand, the data if you are only looking at information that you really need. Any columns that have been removed can be returned to the dataset simply be deleting or editing the step that removed them.

4. Alter the data types for every column in the table. Correct data types are fundamental for many transformation steps, and are essential for filtering, so it's best to get them sorted out early on.
5. Filter out any records that you do not need. Once again, the smaller the dataset, the faster the processing. This includes deduplication.
6. Carry out any necessary data cleansing and transforms.
7. Carry out any necessary column splits or adding custom columns.
8. Add any calculations or logical transformations of data.

Once again, I must stress that this is not a definitive guide. I hope, however, that it will help you to see "the wood for the trees" when you are creating data load and transformation processes using Power BI Desktop.

Error Records

Some data transformation operations will cause errors. This can be a fact of life when mashing up source data. For instance, you could have a few rows in a large dataset where a date column contains a few records that are texts or numbers. If you convert the column to a date data type, then any values that cannot be converted will appear as error values.

Removing Errors

Assuming that you do not need records that Power BI Desktop has flagged as containing an error, you can remove all such records in a single operation:

1. Click inside the column containing errors; or if you want to remove errors from several columns at once, Ctrl-click the titles of the columns that contain the errors.
2. Click Remove Errors in the Home ribbon. Any records with errors flagged in the selected columns are deleted. Removed Errors is added to the Applied Steps list.

You have to be very careful here not to remove valid data. Only you can judge, once you have taken a look at the data, if an error in a column means that the data can be discarded safely. In all other cases, you would be best advised to look at cleansing the data or simply leaving records that contain errors in place. The range and variety of potential errors is as vast as the data itself.

Modifying the Code for a Step

Data ingestion and modification is not only interface-driven in Power BI Desktop. In fact, the entire process is underpinned and powered by a highly specific programming language. Called, informally, "M," this language underlies everything that you have learned to do in the last seven chapters.

Before introducing you to this concept, I need to add a few caveats:

- The "M" language that underpins Power BI Desktop queries is not for the faint of heart. The language can seem abstruse at first sight.
- The documentation is extremely technical and not wildly comprehensible for the uninitiated.

- The learning curve can be steep, even for experienced programmers.
- The "M" language is very different from VBA, which many Excel power-users know well.
- Tweaking a step manually can cause havoc to a carefully wrought data load and transform process.

Moreover, the "M" language is so vast that it requires an entire book, so I have deliberately chosen to provide only the most superficial of introductions here. For greater detail I suggest that you consult the Microsoft documentation. This is currently available at the following URLs:

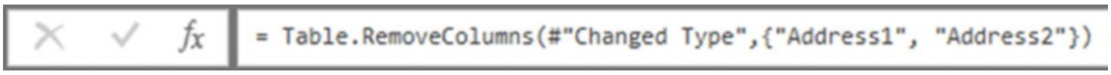
- `https://msdn.microsoft.com/en-us/library/mt779182.aspx`
- `https://msdn.microsoft.com/en-us/library/mt807488.aspx`

Modifying a Single Step

If you feel that you want to delve into the inner reaches of Power BI Desktop, you can also modify steps in a query by editing the code that is created automatically every time that you add or modify a query step.

To get a quick idea of what can be done:

1. Click the Remove Columns step in the Clients table created previously. You will see the "M" code in the formula bar. It will look like that shown in Figure 9-3.



```
= Table.RemoveColumns(#"Changed Type",{"Address1", "Address2"})
```

Figure 9-3. *"M" code for an applied step*

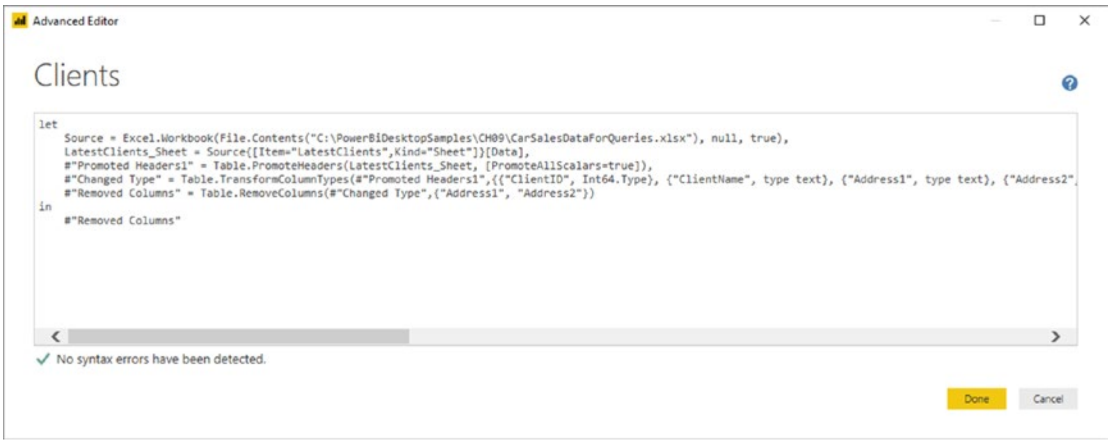
2. Edit the code.
3. Press Enter or click the tick icon (check mark) in the formula bar.

The step and subsequent data will be updated to reflect your changes.

Modifying a Query in "M"

You are not limited to modifying single steps in "M." If you prefer, you can alter an entire query.

1. In the View ribbon, click Advanced Editor. The Advanced Editor dialog will appear, looking like the one shown in Figure 9-4.



```
let
    Source = Excel.Workbook(File.Contents("C:\PowerBiDesktopSamples\CH09\CarSalesDataForQueries.xlsx"), null, true),
    LatestClients_Sheet = Source{[Item="LatestClients",Kind="Sheet"]}[Data],
    #"Promoted Headers1" = Table.PromoteHeaders(LatestClients_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers1",{{"ClientID", Int64.Type}, {"ClientName", type text}, {"Address1", type text}, {"Address2",
    #"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"Address1", "Address2"})
in
    #"Removed Columns"
```

✓ No syntax errors have been detected.

Figure 9-4. *The Advanced Editor for a query*

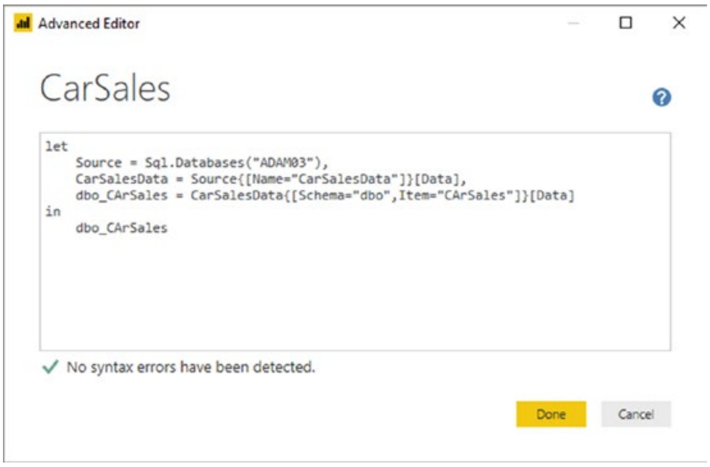
2. Edit the code.
3. Press Enter or click Done.

However, I am not going to do anything other than point you in the direction of the “M” language in this book—and wish you good luck and hope that you have fun if you choose to use it in your work with Power BI Desktop.

Modifying Data Source Settings in the Query Editor

You saw in Chapter 3 how to modify existing connections to data sources. You can also do this directly in the “M” language. This assumes that you know and understand the database that you are working with!

1. Add a new query that connects to the SQL Server database CarSalesData.
2. Select this query in the Queries list on the left.
3. In the Home ribbon, click the Advanced Editor button. The Advanced Editor dialog will appear, as shown in Figure 9-5.



```
let
    Source = Sql.Databases("ADAM03"),
    CarSalesData = Source{[Name="CarSalesData"]}[Data],
    dbo_CArSales = CarSalesData{[Schema="dbo",Item="CArSales"]}[Data]
in
    dbo_CArSales
```

Figure 9-5. *The Advanced Editor dialog to alter a database connection*

4. Alter any of the following elements:
   a. The server name in the Source line (currently "ADAM03").
   b. The database name in the second line (currently Name="CarSalesData").
   c. The schema name in the third line (currently Schema="dbo").
   d. The table name in the third line (currently Item="CarSales").
5. Click Done.

This approach really is working without a safety net, and I am showing you more to raise awareness than anything else. However, it does open the door to some far-reaching possibilities if you wish to continue learning all about the “M” language.

Managing Queries

Once you have used Power BI Desktop for any length of time, you will probably become addicted to creating more and deeper analyses based on wider-ranging data sources. Inevitably, this will mean learning to manage the data sources that feed into your data models efficiently and productively.

Fortunately, Power BI Desktop Query comes replete with a small arsenal of query management tools to help you. These include

- Organizing queries
- Grouping queries
- Duplicating queries
- Referencing queries
- Documenting queries
- Adding a column as a new query

- Enabling data load
- Enabling report refresh

Let's take a look at these functions, one by one.

Organizing Queries

When you have a dozen or more queries that you are using in the Power BI Desktop Query Editor, you may want to exercise some control over how they are organized. To begin with, you can modify the order in which queries appear in the Queries pane on the left of the Power BI Desktop Query window. This lets you override the default order, which is that the most recently added data source appears at the bottom of the list.

Do the following to change the position of a query in the list:

1. Right-click the query that you want to move.
2. Select Move Up (or Move Down) from the context menu.

You have to carry out this operation a number of times to move a query up or down a number of places.

Grouping Queries

You can also create custom groups to better organize the queries that you are using in a Power BI Desktop file. This will not have any effect on how the queries work. Grouping queries is simply an organizational technique, and it will not change in any way the data tables that you see in report mode in Power BI Desktop.

Creating a New Group

The following explains how to create a new group:

1. Right-click the query that you want to add to a new group. I will use the Colors query.
2. Select Move To Group ➤ New Group from the context menu. The New Group dialog will appear.
3. Enter a name for the group and (optionally) a description. I will name the group **ReferenceData**. The dialog will look something like Figure 9-6.

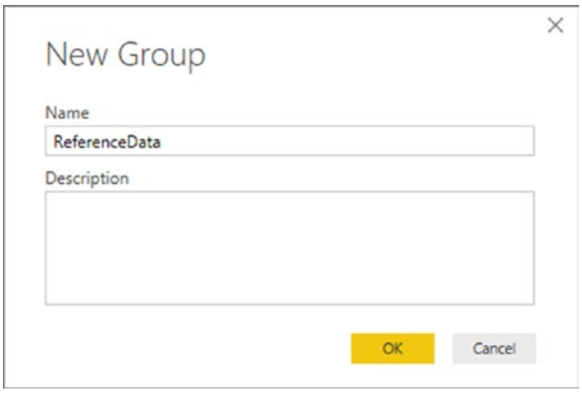


***Figure 9-6.** The New Group dialog*

4. Click OK.

The new group is created and the selected query will appear in the group. The Queries pane will look something like Figure 9-7.

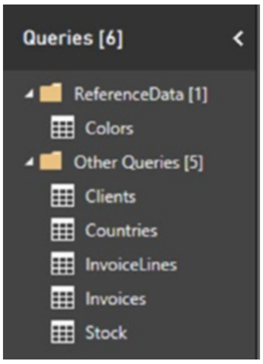


Figure 9-7. *The Queries pane with a new group added*

■ **Note** By default, all other queries are added to a group named Other Queries.

Renaming Groups

You can rename any groups that you have added.

1. Right-click the query that you want to rename.
2. Select Rename from the context menu.
3. Edit or replace the name.
4. Press Enter.

■ **Note** The Other Queries group cannot be renamed or deleted. By default, all new queries will be added to this group.

Adding a Query to a Group

To move a query from its current group to another group, you can carry out the following steps:

1. Right-click the query that you want to add to another existing group.
2. Select Move To Group ➤ *Destination Group Name* from the context menu.

The selected query is moved to the chosen group.

Duplicating Queries

If you have done a lot of work transforming data, you could well want to keep a copy of the original query before trying out any potentially risky alterations to your work. Fortunately, this is extremely simple.

1. Right-click the query that you want to copy.
2. Select Duplicate from the context menu.

The query is copied and the duplicate appears in the list of queries inside the same group as the source query. It has the same name as the original query, with a number in parentheses appended. You can always rename it in the Query Settings pane or in the Queries pane on the left of the Query Editor window.

■ **Note** You can copy and paste queries if you prefer. The advantage of this technique is that you can choose the destination group for the copied query simply by clicking the folder icon for the required group *before* pasting the copy of the query.

Referencing Queries

If you are building a complex ETL (Extract, Transform, Load) routine, you might conceivably organize your work in stages to better manage the process. To help you with this, the Power BI Desktop Query Editor allows you to use the output from one query as the source for another query. This enables you to break down different parts of the process (structure, filters, then cleansing, for example) into separate queries so that you can concentrate on different aspects of the transformation in different queries.

To use the output of one query as the source data for another, you need to *reference* a query. The following explains how to do it:

1. Right-click the query that you want to use as the source data for a new query.
2. Select Reference from the context menu. A new query is created in the list of queries in the Queries pane.
3. Right-click the new query, select Rename, and give it a meaningful name.

Unless you rename the query, the new query has the same name as the original query, with a number in parentheses appended. If you click the new query, you see exactly the same data in the referenced query as you can see if you click the final step in the source query.

From now on, any modifications that you make in the referenced (source) query produces an effect on the data that is used as the source for the second query.

In practice, I suspect, you will not want to use two copies of the same query to create reports. Indeed, if a query is being used as an “intermediate” query, the data that it contains might not even be fully usable. So you could want to make the intermediate query unavailable when creating reports and dashboards in Power BI Desktop. To do this:

1. Right-click the original (source) query.
2. Unselect Enable Load from the context menu. The check mark to the left of the menu item will disappear.

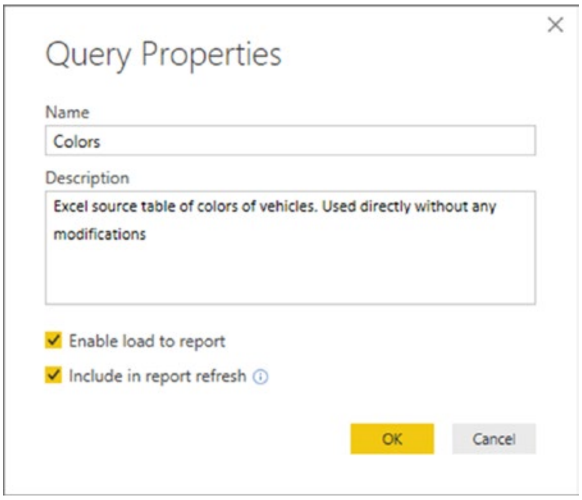
The source query will no longer appear in Report View, and so cannot be used in visuals. It is worth noting that any queries that are “intermediate” queries (that is, queries that you use to modify data but that do not show in Data View or Report View) are in italics.

■ **Note** Ensure that any existing reports do not use a query that you subsequently make unavailable in this way, or you will end up with broken visuals in your reports.

Documenting Queries

In a complex ETL process, it is easy to get confused—or simply forget—which query does what. So I always advise documenting queries by adding a meaningful description.

1. Right-click the query that you want to annotate.
2. Select Properties from the context menu. The Query Properties dialog will appear.
3. Add a description. The result could be like the dialog shown in Figure 9-8.


Query Properties

Name: Colors

Description: Excel source table of colors of vehicles. Used directly without any modifications

✓ Enable load to report

✓ Include in report refresh

Figure 9-8. *Adding a description to a query*

4. Click OK.

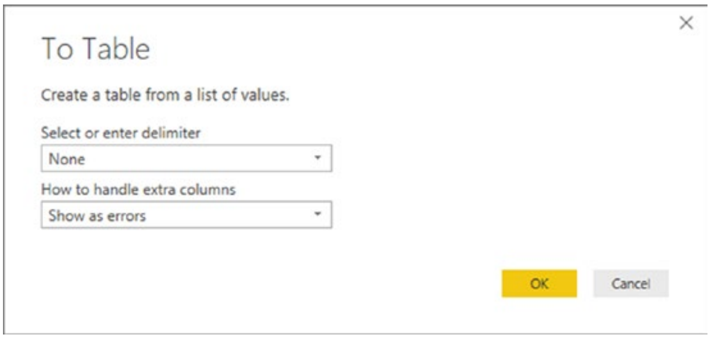
The description that you added is now visible as a tooltip if you hover the cursor over the query name in the list of queries in the Query Editor.

Adding a Column As a New Query

There are occasions when you might want to extract a column of data and use it as a separate query. It could be that you need the data that it contains as reference data for another query, for example. The following steps explain how you can do this:

1. In the Queries list on the left, select the query containing the column that you want to isolate as a new query.

2. Right-click the title of the column containing the data that you want to isolate.
3. Select Add As New Query from the context menu. A new query is created. It is named after the original query and the source column.
4. In the Transform ribbon, click To Table. The To Table dialog will appear, as you can see in Figure 9-9.



To Table

Create a table from a list of values.

Select or enter delimiter: None

How to handle extra columns: Show as errors

Figure 9-9. *The To Table dialog*

5. Click OK. The new query will become a table of data and will have the name of the column that you selected.
6. Rename the query, if you judge this necessary.

You can now use this query in your data model and as part of a linked set of query processes.

■ **Note** A query created in this way is completely disconnected from the source query from where the data was taken. Put another way, any refresh of the source data will have *no* effect on the new query that you created from a column.

Enabling Data Load

You may have gained the impression that any query that you create will always become part of the Power BI Desktop data model. This is emphatically *not* the case. You can create queries that are in effect only "staging" queries that are part of a more complex sequence of transformations, or queries that contain only lookup data that is added to another table but not needed in the data model, for instance. In cases like these, you certainly do not want these tables adding clutter to the data model.

To prevent a query being added to the data model, do the following:

1. In the Queries list on the left, select the query that you want to keep in the Query Editor—but not in the data model.
2. Right-click the query and uncheck Enable Load.

The query will no longer be loaded into the data model with the other queries.

To reset a query as a candidate for loading into the data model, merely carry out the same operation and ensure that Enable Load is checked. You can see from the check mark in the context menu if the query is due to be loaded or not. As an alternative, you can right-click the query and display the query properties, where you can check (or uncheck) the “Enable load to report” check box.

■ **Note** You can also set this property in the Query Properties dialog that you saw in Figure 9-8.

Enabling Report Refresh

By default, all queries can be refreshed. This lets you gather the very latest data from the source into the Power BI Desktop Query Editor and then into the data model.

There could be times when you do not want to refresh a query. Perhaps the data source is unobtainable, or is slow, or you want to return later for the latest data. Power BI Desktop Query Editor lets you set the refresh option for each query like this:

1. In the Queries list on the left, right-click the query whose refresh status you want to modify.
2. Uncheck “Include in report refresh.”

■ **Note** Only queries that are enabled for load can have the refresh property modified.

Pending Changes

When you are dealing with data sources and switch from the Query Editor to the Power BI Desktop view, normally, you then want to load the full data from the source into the data model.

The downside to this approach is that a huge set of source data can take a long time to load, or reload, when you move back to creating and modifying visualizations. This is why Power BI Desktop will let you select Close from the Close & Apply button in the Query Editor Home ribbon. Doing this will return you instantly to the Data View, but will not apply any changes that you have made. As a reminder, you will see an alert like the one shown in Figure 9-10 at the top of the Data View.

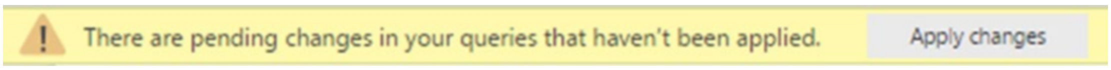

There are pending changes in your queries that haven't been applied.

Figure 9-10. *The pending changes alert*

You can continue to work in Data View as long as you like. Click the Apply Changes button when you have time to reload the modified source data into the data model. You need to be aware, however, that you will not see all the latest changes to the structure of the tables and fields in the Fields list until you apply the changes.

Reusing Data Sources

Over the course of Chapters 2 through 5, you have seen how to access data from a wide variety of sources to build a series of queries across a range of reports. The reality will probably be that you will frequently want to point to the same sources of data over and over again. In anticipation of this, the Power BI development team has found a way to make your life easier.

Power BI Desktop remembers the most recent data sources that you have used, and lets you reuse them quickly and easily in any report. Here is how:

1. In the Home ribbon, click the Recent Sources button. The list of the dozen or so most recently used data sources will appear. You can see this in Figure 9-11.

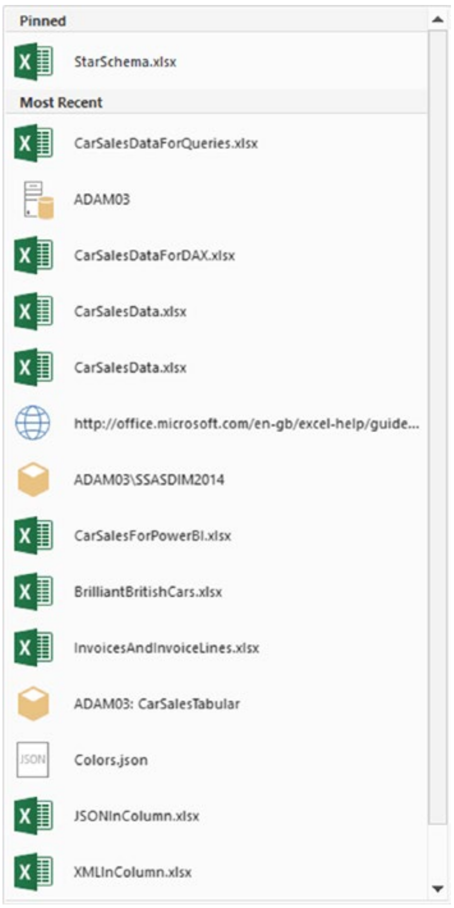
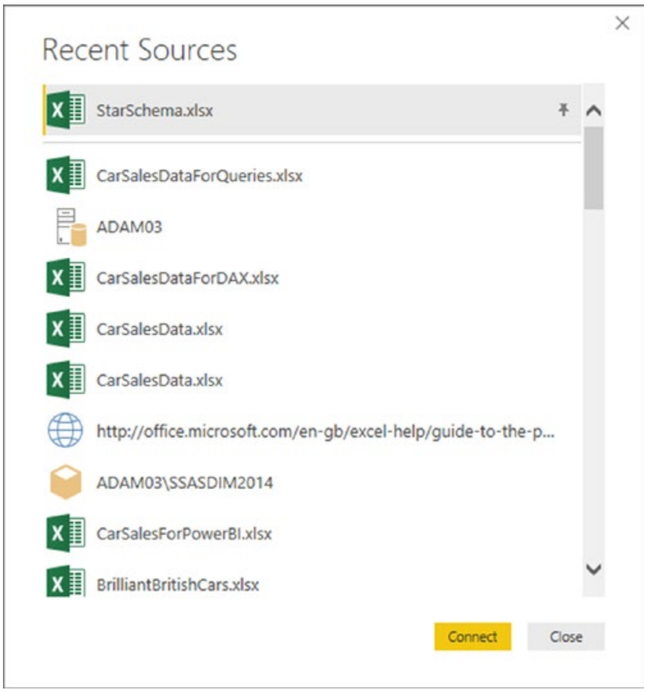

Pinned
StarSchema.xlsx
Most Recent
CarSalesDataForQueries.xlsx
ADAM03
CarSalesDataForDAX.xlsx
CarSalesData.xlsx
CarSalesData.xlsx
http://office.microsoft.com/en-gb/excel-help/guide...
ADAM03\SSASDIM2014
CarSalesForPowerBI.xlsx
BrilliantBritishCars.xlsx
InvoicesAndInvoiceLines.xlsx
ADAM03: CarSalesTabular
Colors.json
JSONInColumn.xlsx
XMLInColumn.xlsx

Figure 9-11. *Recently used sources*

2. Click the source that you want to reconnect to, and continue with the data load or connection.

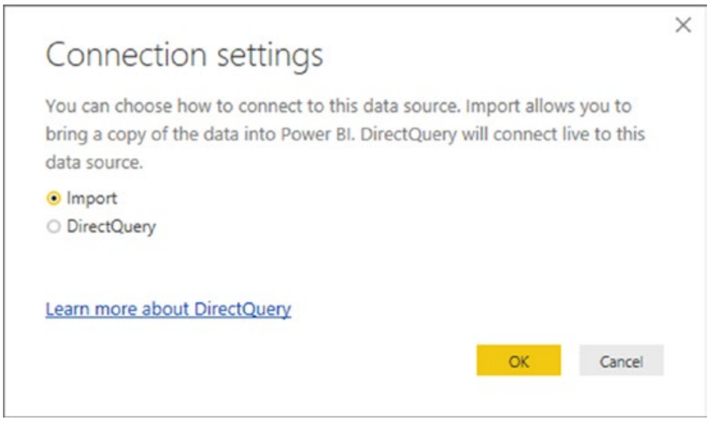
If you cannot see the data source that you want, and you are sure that you have used it recently, then you can scroll to the bottom of this list and click More. Power BI Desktop will display the complete list of recent sources in the Recent Sources dialog that you can see in Figure 9-12.


Recent Sources

StarSchema.xlsx
CarSalesDataForQueries.xlsx
ADAM03
CarSalesDataForDAX.xlsx
CarSalesData.xlsx
CarSalesData.xlsx
http://office.microsoft.com/en-gb/excel-help/guide-to-the-p...
ADAM03\SSASDIM2014
CarSalesForPowerBI.xlsx
BrilliantBritishCars.xlsx

Figure 9-12. *The Recent Sources dialog*

If you are connecting to any of the database or data warehouse sources that allow DirectQuery or Live Connection, you will see a dialog like the one in Figure 9-13.


Connection settings

You can choose how to connect to this data source. Import allows you to bring a copy of the data into Power BI. DirectQuery will connect live to this data source.

● Import
○ DirectQuery

Learn more about DirectQuery

Figure 9-13. *Defining connections settings when reusing an existing connection*

Here you can decide whether to load data or use a direct connection, this time with the chosen data source.

Pinning a Data Source

If you look closely at Figures 9-11 and 9-12, you see that the Excel file StarSchema.xlsx is pinned to the top of both the Recent Sources menu and the Recent Sources dialog. This allows you to make sure that certain data sources are always kept on hand and ready to reuse.

Do the following to pin a data source that you have recently used to the menu and dialog of recent sources:

1. Click the Recent Sources button in the Power BI Desktop Home ribbon.
2. Scroll down to the bottom of the menu and click More. The Recent Sources dialog will appear.
3. Hover the mouse over a recently used data source. A pin icon will appear at the right of the data source name.
4. Click the pin icon. The data source is pinned to the top of both the Recent Sources menu and the Recent Sources dialog. A small pin icon remains visible at the right of the data source name.

■ **Note** To unpin a data source from the Recent Sources menu and the Recent Sources dialog, all you have to do is click the pin icon for a pinned data source. This unpins it and it reappears in the list of recently used data sources.

Parameterizing Queries

Parameters in Power BI Desktop enable you to define and apply specific criteria to certain aspects of queries. At their heart, they are a technique that enables you to

- Store a value that can then be used in multiple queries.
- Restrict a selection of potential values to a specific list of options.

There are currently three basic ways of creating parameters:

- A single value that you enter
- A selection of a value from a list of possible values that you enter manually
- A selection of a value from a list of possible values that you create using existing queries

It follows that using parameters is a two-step process:

- Create a parameter.
- Apply it to a query.

A parameter is really nothing more than a specialized type of query. As it is a query, you can

- Load it into the data model (although this is rarely required).

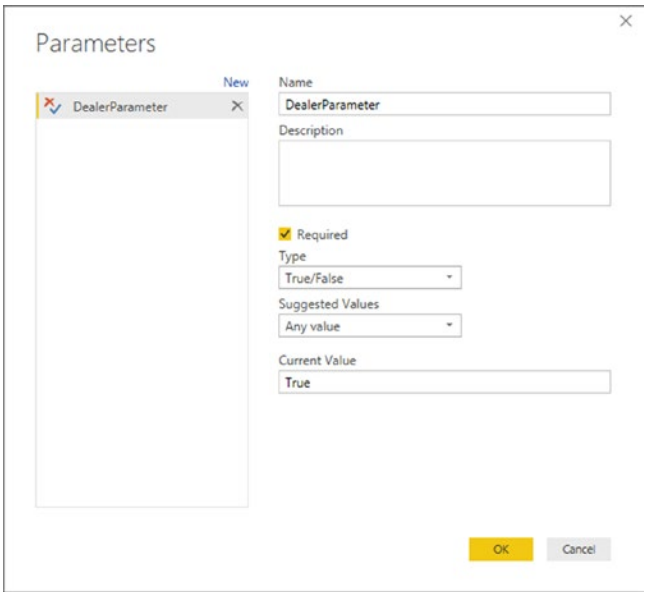
- Reference it from another query.
- Use it in DAX formulas.

The next three short subsections will explain how you can create parameters. I will then show you some of the ways that you can apply parameters in the Query Editor to filter or transform the data.

Creating a Simple Parameter

At its simplest, a parameter is a value that you store so that you can use it later to assist you in your data transformation. Here is how you can store a parameter containing a "True" value ready for use in filtering subsequent datasets:

1. Load the data contained in the Excel file C:\PowerBiDesktopSamples\CH09\CarSalesOverview.xlsx (using the worksheet that is also called CarSalesOverview) into Power BI Desktop.
2. In the Home ribbon, click Edit Queries. The Query Editor will open.
3. In the Query Editor Home ribbon, click the small triangle at the bottom of the Manage Parameters button, then select New Parameter from the available menu options. The Parameters dialog will appear.
4. Enter **DealerParameter** as the parameter name.
5. Ensure that the Required check box is selected.
6. Choose True/False from the popup list of types.
7. Enter True as the current value from the popup list. The dialog will look like the one in Figure 9-14.



Name: DealerParameter
Description:
Required ✓
Type: True/False
Suggested Values: Any value
Current Value: True

Figure 9-14. *The Parameters dialog*

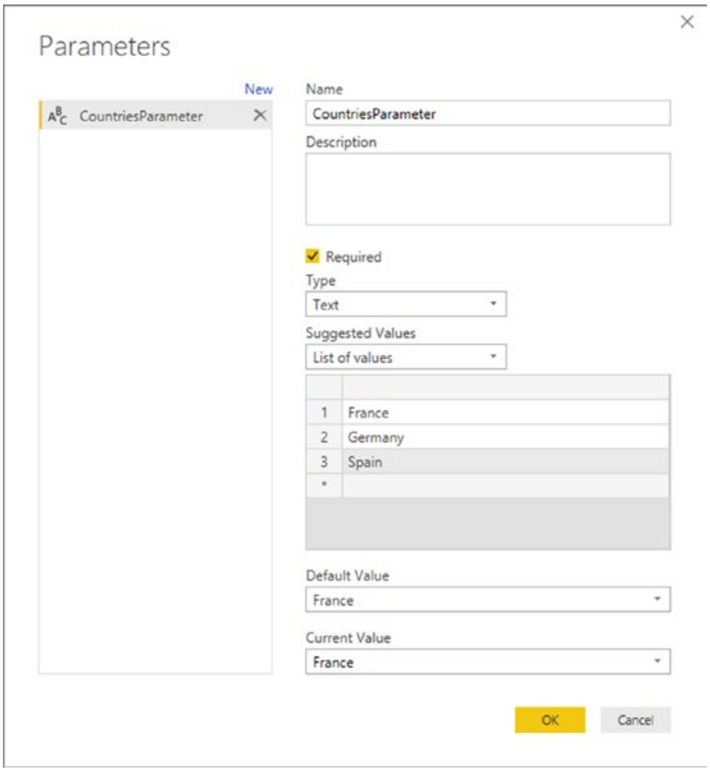
8. Click OK. The new parameter will appear in the Queries list on the left.

For the moment, all you have done is create a parameter and store a value in it. You will see how to use this parameter in a few pages' time.

Creating a Set of Parameter Values

While a single parameter can always be useful, in reality you are likely to need lists of potential parameters. This will allow you to choose a parameter value from a predefined list in certain circumstances. Here is an example of creating a parameter containing a subset of the available country names used in the sample data:

1. Using the Power BI Desktop file that you created in the previous section (the one based on the Excel file C:\PowerBiDesktopSamples\CH09\CarSalesOverview.xlsx), open the Query Editor.
2. In the Home ribbon, click Edit Queries, unless the Query Editor is already open.
3. In the Query Editor Home ribbon, click the small triangle at the bottom of the Manage Parameters button, then select New Parameter from the available menu options. The Parameters dialog will be displayed,
4. Enter **CountriesParameter** as the parameter name.
5. Ensure that the Required check box is selected.
6. Choose Text from the popup list of types.
7. In the Suggested Values popup list, select List of Values.
8. Enter the following three values in the grid that has now appeared:
    a. France
    b. Germany
    c. Spain
9. Select France as the Default Value from the popup list.
10. Select France as the Current Value from the popup list. The dialog will look like the one shown in Figure 9-15.



Name: CountriesParameter

Description:

✓ Required

Type: Text

Suggested Values: List of values

1	France
2	Germany
3	Spain
*	

Default Value: France

Current Value: France

Figure 9-15. *The Parameters dialog*

11. Click OK. The new parameter will appear in the Queries list on the left.

Once again, all you have done is create the parameter. You will see how it can be applied in a couple of pages' time.

■ **Note** As you can see, any current value that you have chosen will appear in the Queries pane in parentheses to the right of the parameter name. This is to help you remember which value is current-and is possibly being used to filter data.

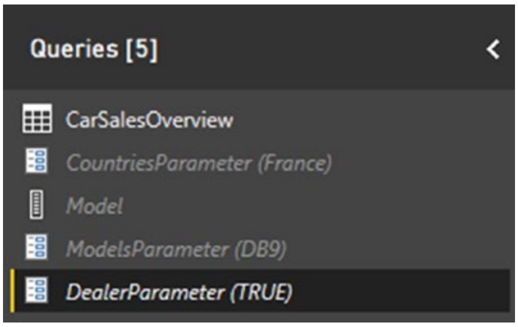
Creating a Query-Based Parameter

Typing lists of values that you can use to choose a parameter is not only laborious, it is also potentially error-prone. So you can use the data from existing queries to create the series of available elements that you use in a parameter instead of typing lists of values. As an example of this, suppose that you want a parameter that contains all the available makes of car that the company sells:

1. Using the Power BI Desktop file that you created in the previous section (the one based on the Excel file C:\PowerBiDesktopSamples\CH09\CarSalesOverview.xlsx), open the Query Editor.

2. Select the query CarSalesOverview in the Queries list.
3. Right-click the title of the column named Model, and select Add As New Query. A new query named Model will appear in the Queries list.
4. Right-click the Model query and uncheck Enable Load.
5. In the newly created query, click Remove Duplicates in the Transform ribbon. The List column will only display unique values.
6. In the Query Editor Home ribbon, click the small triangle at the bottom of the Manage Parameters button, then select New Parameter from the available menu options. The Parameters dialog will be displayed.
7. Enter **ModelsParameter** as the parameter name.
8. Ensure that the Required check box is selected.
9. Choose Text from the popup list of types.
10. In the Suggested Values popup list, select Query.
11. Select Model as the query containing a list of values to use from the popup list of available lists.
12. Enter DB9 as the Current Value.
13. Click OK. The new parameter will appear in the Queries list on the left.

You should now be able to see all three parameters that you have created in the Queries pane, as shown in Figure 9-16.


Queries [5]

CarSalesOverview

CountriesParameter (France)

Model

ModelsParameter (DB9)

DealerParameter (TRUE)

Figure 9-16. *Parameters in the Queries list*

■ **Note** It is normal to prevent a list that is being used to provide a series of values for a parameter from being loaded into the data model (as you did in step 4). This is because this query (or list, if you prefer) is not required in the data model, and you do not want to confuse users by having it appear in the Fields list.

Once your parameters have been created, you can quit the Query Editor by clicking the Close & Apply button. You should see no change in the Power BI Desktop Report View. This is because parameters are used to shape a data flow process, but *not* as the data in the data model.

■ **Tip** It is also possible to create parameters "on the fly" (that is directly from inside a dialog that uses a parameter) when you want to use them. However, I find it better practice—and more practical—to prepare parameters beforehand. This forces you to think through the reasons for the parameter as well as the potential range of its use. It can also avoid your making errors when trying to do two different things at once.

Modifying a Parameter

Fortunately, parameters are not set in stone once they are created. You can easily modify

- The structure of a parameter
- The selected parameter element (the current value)

Modifying the Structure of a Parameter

Should you need to modify the way that a parameter is constructed, one way is to do the following:

1. In the Query Editor Home ribbon, click Manage Parameters. The Parameters dialog will be displayed.
2. In the left pane of the dialog, click the parameter that you want to modify. The parameter definition will appear on the right.
3. Carry out any required modifications.
4. Click OK.

Alternatively, you can do this:

1. Click the parameter in the Queries pane on the left of the Query Editor. The Query Editor will look like the one shown in Figure 9-17.

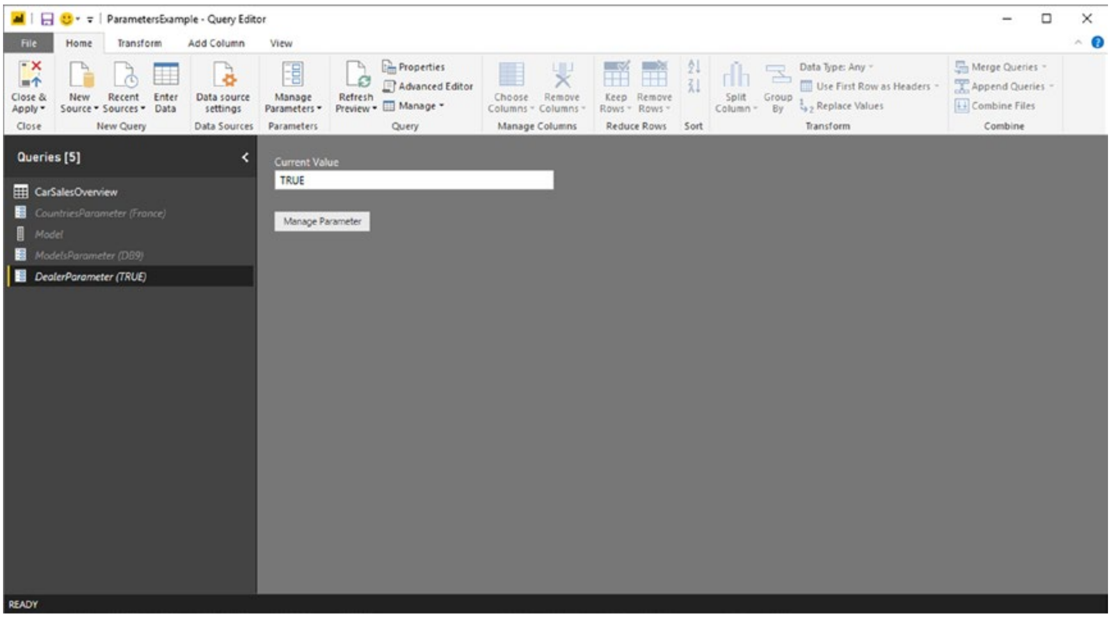


Figure 9-17. *Managing parameters*

2. Click the Manage Parameter button. The Parameters dialog will appear.
3. Carry out any required modifications.
4. Click OK.

You can also, if you prefer, right-click a parameter in the Queries pane and select Manage from the popup menu to display the Parameters dialog.

Applying a Parameter when Filtering Records

Now that you have seen how parameters are created, it is time to see them in action. As a first example of applying a parameter, you will see how to use a parameter to filter a query:

1. Open the file C:\PowerBiDesktopSamples\CH09\ParametersExample.pbix. This file contains the three parameters created previously.
2. Open the Power BI Desktop Query Editor.
3. Click the CarSalesOverview query in the Queries list. A dataset of car sales information will appear.
4. Click the popup menu for the CountryName column.
5. Select Text Filters ➤ Equals. The Filter Rows dialog will appear.
6. Leave Equals as the first choice.
7. Click the second popup and select Parameter from the list. You can see this in Figure 9-18.

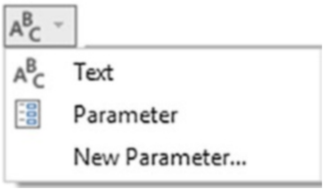
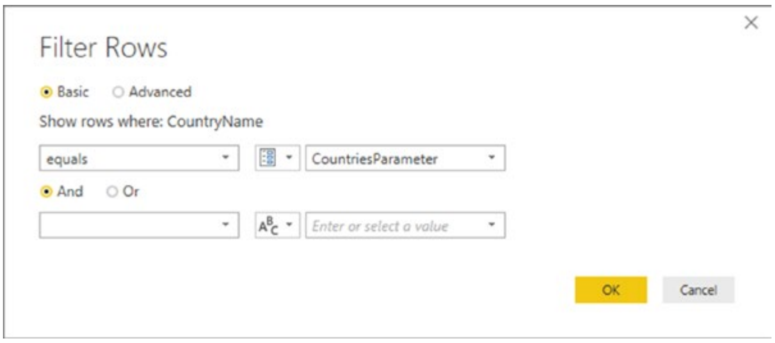


Figure 9-18. *Selecting a parameter for a filter*

8. Select CountriesParameter for the third popup. The dialog will look like the one shown in Figure 9-19.


Filter Rows

Basic [✓] Advanced

Show rows where: CountryName

equals | CountriesParameter

And [✓] Or

Enter or select a value

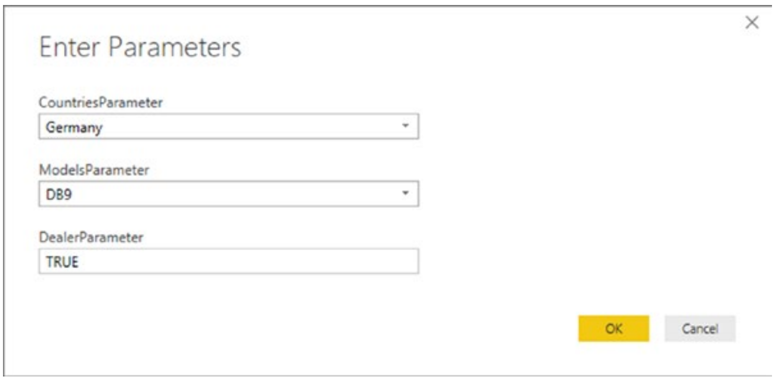
Figure 9-19. *Applying a parameter for a filter*

9. Click OK. The current parameter value (the country that you selected) will be applied and the dataset will be filtered using the current parameter value.

Modifying the Current Value of a Parameter

You could be forgiven for wondering if it is worth setting up a parameter merely to filter a dataset. However, this whole approach becomes more interesting if you modify the current parameter value and then refresh the data to apply the new parameter. Here is an example of this:

1. In the Query Editor Home ribbon, click the small triangle to display the menu for the Manage Parameters button.
2. Select Edit Parameters. The Enter Parameters dialog will appear.
3. From the popup list of values for the CountriesParameter, select one of the available values (and not the value that was previously selected). The dialog should look like the one shown in Figure 9-20.



CountriesParameter: Germany

ModelsParameter: DB9

DealerParameter: TRUE

Figure 9-20. *Modifying the current value of a parameter*

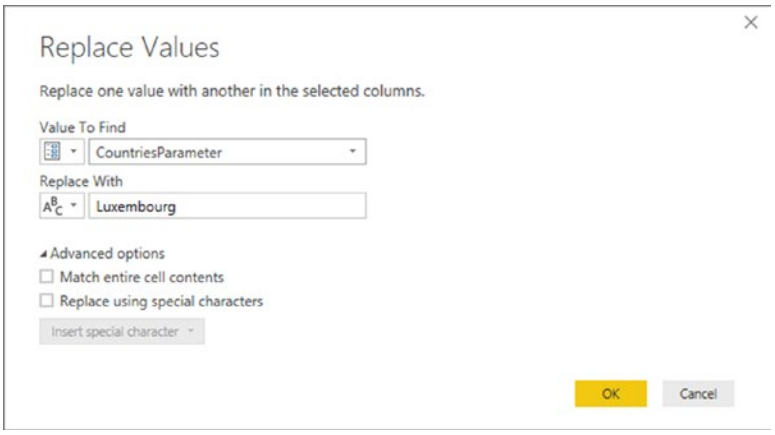
4. Click OK.
5. In the Query Editor Home ribbon, click Refresh Preview. The data will be refreshed and the new parameter value applied to the filter.

This approach becomes particularly useful if you have many combinations of filter values to test. In essence, you can apply a series of filters to several columns (or create complex filters) using several parameters and then test the results of different combinations of parameters on a dataset using the Enter Parameters dialog. This technique avoids having to alter multiple filters manually-and repeatedly. As an added bonus, you can restrict the user (or yourself) to specific lists of parameter choices by defining the lists of available parameter options. You can see this for the popup lists that appear when you select the CountriesParameter popup or the ModelsParameter popup.

Applying a Parameter in a Search and Replace

Another use for parameters is to apply them as either the search value and/or the replacement value in a search and replace operation. You can see this in the following example:

1. Open the Power BI Desktop file C:\PowerBiDesktopSamples\CH09\ParametersExample.pbix.
2. Open the Power BI Desktop Query Editor.
3. Click the CarSalesOverview query in the Queries pane.
4. Click inside the CountryName column.
5. In the Transform ribbon, click Replace Values. The Replace Values dialog will be displayed.
6. Click the popup list to the left under Value To Find, and select Parameter.
7. Choose CountriesParameter as the parameter to apply.
8. Enter Luxemburg (for instance) as the replacement value. The dialog will look like the one in Figure 9-21.


Replace Values

Replace one value with another in the selected columns.

Value To Find: CountriesParameter

Replace With: Luxembourg

Advanced options

☐ Match entire cell contents

☐ Replace using special characters

Figure 9-21. *Using a parameter in search and replace*

9. Click OK. The parameter's current value will be replaced in the dataset for the selected column.

Applying a Parameter to a Datasource

In some corporate environments, there are many database servers that are available, and possibly even more databases. You may find it difficult to remember all of these-and so may the users that you are preparing PowerBI Desktop reports for.

One solution that can make a corporate environment easier to navigate is to prepare parameters that contain the lists of available servers and databases. These parameters can then be used—and updated—to guide users in their choice of SQL Server, Oracle, or other database data sources.

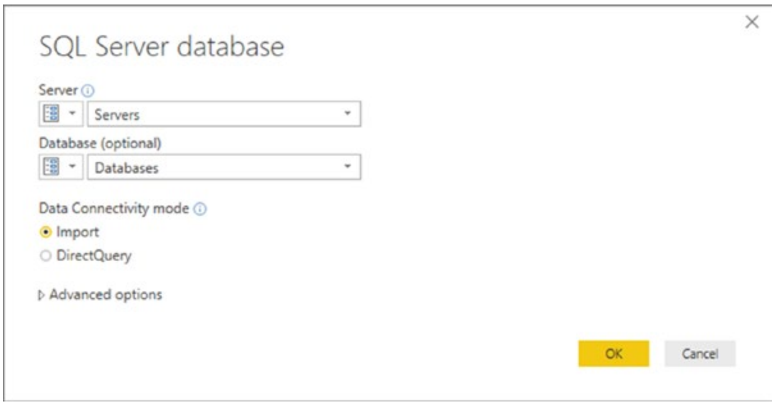
To see this in action, you will first have to prepare two parameters:

- A list of servers
- A list of databases

You can then see how to use these parameters to connect to data sources. Of course, you will have to replace the example server and database names that I use here in steps 3.d and 4.d with names from your own environment.

1. Open a new Power BI Desktop file and close the splash screen.
2. Click Edit Queries to open the Query Editor.
3. Create a new parameter using the following elements:
    a. Name: Servers
    b. Type: Text
    c. Suggested Values: List of Values
    d. Values in the list: ADAM03 and ADAM03\SQLServer2016 (or your database server)
    e. Default Value: ADAM03\SQLServer2016 (or your database server)
    f. CurrentValue: ADAM03\SQLServer2016 (or your database server)

4. Create a new parameter using the following elements:
    a. Name: Databases
    b. Type: Text
    c. Suggested Values: List of Values
    d. Values in the list: CarSalesData and CarSalesMemoryBased
    e. Default Value: CarSalesData
    f. CurrentValue: CarSalesData
5. Click Close & Apply to close the Query Editor.
6. In the Power BI Desktop Report screen, click Get Data ➤ SQL Server Database.
7. On the Server line, click the popup for the server and choose Parameters. Select the Servers parameter.
8. On the Database line, click the popup for the database and choose Parameters. Select the Databases parameter. The SQL Server dialog will look like the one shown in Figure 9-22.


SQL Server database

Server: Servers

Database (optional): Databases

Data Connectivity mode: [✓ Import] DirectQuery

Figure 9-22. *Using a parameter to select the server and database*

9. Choose the data connectivity mode and any advanced options that you want to set.
10. Click OK. The server connection process and dialogs will appear and you will then see the Navigator dialog displaying the tables and views for the current server and database values in the two parameters.

■ **Note** Preparing parameters for data connection is particularly useful if you then save the file as a template that can be used as the basis for multiple different Power BI Desktop report files. You will learn how to create templates in a few pages.

Other Uses for Parameters

These examples only cover a few of the cases where parameters can be applied in Power BI Desktop. Indeed, the range of circumstances where a parameter can be applied is increasing with each release of the product. So look out for all the dialogs that give you the option of using a parameter!

Applying a Parameter to a SQL Query

If you are using a relational database, such as Oracle or SQL Server, as a data source (and if you are reasonably up to speed with the flavor of SQL that the source database uses), you can query a database using SQL and then apply Power BI Desktop parameters to the source query.

Let's see this in action:

1. Open a new Power BI Desktop file, open the Query Editor, and create the parameter named CountriesParameter that you saw a few pages ago.
2. Click Close & Apply to close the Query Editor.
3. In the Power BI Desktop Report screen, click Get Data ➤ SQL Server Database.
4. Enter the server and database that you are using. (If you are using the examples from the Apress web site, then it will be your server and the database CarSalesData.)
5. Click Advanced Options and enter the following SQL statement:

```
SELECT  *
FROM    CarSalesData.Data.CarSalesData
WHERE   CountryName = 'Germany'
```

6. Click OK and confirm any dialogs about data access and permissions.
7. Click Edit to connect to the data and open the Query Editor.
8. There should only be one Applied Step for the data connection. Expand the formula bar and tweak the formula so that it looks like this:

```
= Sql.Database("ADAM03\SQLSERVER2016", "CarSalesData", [Query="SELECT  *#(lf)
FROM    CarSalesData.Data.CarSalesData#(lf)WHERE   CountryName = '"&
CountriesParameter &"'"])
```

9. Click the tick icon in the formula bar to confirm your changes. The data will change to display the data for France (the current parameter value) rather than Germany (the initial value in the SQL.)

You can now alter the parameter value and refresh the data. This will place the current parameter inside the SQL WHERE clause and only get the data for the current parameter.

In case this seems a little succinct, let's look at the code used by Power BI Desktop *before* you made the change in step 8. The M language read:

```
= Sql.Database("ADAM03\SQLSERVER2016", "CarSAlesDAta", [Query="SELECT  *#(lf)
FROM    CarSalesData.Data.CarSalesData#(lf)WHERE   CountryName = 'France'#(lf)"])
```

The change was to replace:

France

with:

“& CountriesParameter &”

What you did was to replace the hard-coded criterion “France” with the parameter reference. Indeed, much as you would in Excel, you added double quotes and ampersands to the formula to allow the code to include an extraneous element.

This was an extremely simple example, but I hope that it opens the door to some fairly advanced use of parameters in database connections.

■ **Note** Updating data once a parameter has changed might require accepting data changes and new permissions.

Query Icons

As you could see previously in Figure 9-16, there are three query icons. These are explained in Table 9-1.

Table 9-1. *Query Icons*

Icon	Query Type	Description
▦	Query	The icon for a standard query.
▤	List	The icon for a list.
☰	Parameter	The icon for a parameter.

Power BI Templates

If you have created a Power BI model that contains powerful data transformation routines, complex calculations (of which you’ll learn more in later chapters), or simply a set of data connection options-like the ones that you saw in the previous pages-you could want to use any or all of these as the common basis for multiple reports.

The solution to this is extremely simple. You just save a Power BI Desktop file containing all the code, ETL, and other resources that you will need in future reports as a template. You can then open the template and a new Power BI Desktop file will open using all the template’s queries, parameters, and code. You then save the new file perfectly normally as a “standard” Power BI Desktop file.

To save a file as a template:

1. Create the model file containing the queries, parameters, and all other elements that make it the basis for future reports.
2. In the Home ribbon, click File ➤ Save As.

3. In the Save As dialog, select Power BI Template File (*.pbit) as the Save As type.
4. Enter a file name and click OK.

You can now open the template file just as you would any other Power BI Desktop file. Power BI Desktop will open a copy of the template, leaving the template itself untouched.

Copying Data from Power BI Desktop Query Editor

Power BI Desktop is designed as a data destination. It does not have any data export functionality as such. You can manually copy data from the Power BI Desktop Query Editor, however. More precisely, you can copy any of the following:

- The data in the query
- A column of data
- A single cell

In all cases, the process is the same:

1. Click the element to copy. This can be
    a. The top-left square of the data grid
    b. A column title
    c. A single cell
2. Right-click and select Copy from the context menu.

You can then paste the data from the clipboard into the destination application.

■ **Note** This process is somewhat limited because you cannot select a range of cells. And you must remember that you are only looking at sample data in the Query Editor.

Conclusion

In this chapter you saw how to manage and extend the contents of the queries that you can create using Power BI Desktop. Specifically, you saw how to modify individual steps in a data load and transformation process. This ranged from renaming steps to changing the order of steps in a process—or even altering the specification of what a step actually does.

Then you saw how to manage whole queries. You learned how to rename and group queries as well as how to chain queries so that the output from one query became the source of data for another query.

Finally, you learned how to add parameters to queries and how to interact with queries in a controlled fashion. This lets you make queries—and so the entire ETL process—more flexible and interactive.

This chapter closes the section on loading and transforming source data. You will now move on to structuring an entire data model using the output from queries. This is the subject of the next chapter.

CHAPTER 10

Creating a Data Model

You need only one thing to create stunning visualizations and that is good data. Specifically, you need clean and accurate data loaded into the Power BI Desktop data model. There you can hone tens of millions of rows from multiple data sources into a coherent and powerful framework on which you can build your analyses and dashboards.

Finding, editing, and loading the source data using queries (as you learned in Chapters 2 through 9) is fundamental to preparing your data for the data model. However, the process does not stop there. Once you have made accessible the data that you need, you still have a few more tasks to carry out. To complete the process, you need to assemble these queries into a coherent structure that is clear and comprehensible to anyone who uses it.

To guarantee that you are at ease when creating a data model in Power BI Desktop for later analysis in your dashboards, this chapter introduces you to some of the techniques that you need to apply to model your data. You will discover how to take the data tables that you loaded and convert them into a structured dataset in the Power BI Desktop data model. This framework enables you to deliver information, insight, and analysis from the data in the tables. This learning curve covers

- Manipulating tables
- Specifying data types
- Formatting data in the data model
- Establishing relationships between the tables
- Defining hierarchies
- Categorizing data
- Adding “sort by” columns that ensure the correct sort order in dashboard elements
- Creating groups based on column values

The fourth point in this list is probably the most fundamental aspect of data modeling. To report on data accurately and precisely, you must allow Power BI Desktop to understand how the data in one table is linked to the data contained in another table. Chaining one table to another will let you use the data to deliver accurate and cogent results.

Once again, all the sample files used in this chapter are available on the Apress web site. Once downloaded, they should be in the C:\PowerBiDesktopSamples\CH10 folder.

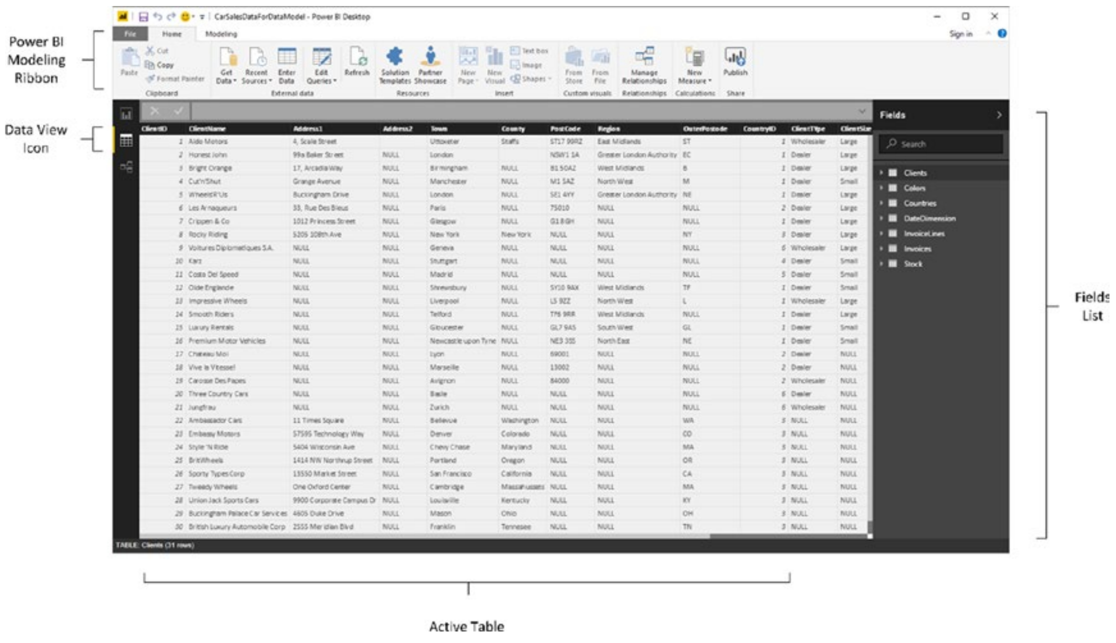
Data Modeling in the Power BI Desktop Environment

Before leaping into the detail of what you can do to create and enhance a data model, I think that it is best if you first familiarize yourself with the tool itself so that you can feel at home in this new environment.

The Power BI Desktop Data View

To start using Power BI Desktop for data modeling, you need to follow these steps:

1. Run Power BI Desktop and load any queries that you need to access your data. To follow the examples in this chapter, load the C:\PowerBiDesktopSamples\CH10\CarSalesDataFromDataModel.pbix file.
2. If you are using your own data, make sure that you have closed the Power BI Desktop Query window, applying any changes that you have made, and reverted to the Power BI Desktop environment.
3. Click the Data View icon on the left. You will switch to the Data View, where you can see all the available tables in the Fields list on the right. One of the tables will be selected and its data will be visible in the Data pane, as shown in Figure 10-1.



ClientID	ClientName	Address1	Address2	Town	County	PostCode	Region	OuterPostcode	CountryID	ClientType	ClientSiz
1	Aldo Motors	4, Scale Street		Uttoxeter	Staffs	ST17 99RZ	East Midlands	ST	1	Wholesaler	Large
2	Honest John	99a Baker Street	NULL	London		NSW1 1A	Greater London Authority	EC	1	Dealer	Large
3	Bright Orange	17, Arcadia Way	NULL	Birmingham	NULL	B1 5OAZ	West Midlands	B	1	Dealer	Large
4	Cut'n'Shut	Grange Avenue	NULL	Manchester	NULL	M1 5AZ	North West	M	1	Dealer	Small
5	WheelsR'Us	Buckingham Drive	NULL	London	NULL	SE1 4YY	Greater London Authority	NE	1	Dealer	Large
6	Les Arnaqueurs	33, Rue Des Bleus	NULL	Paris	NULL	75010	NULL	NULL	2	Dealer	Large
7	Crippen & Co	1012 Princess Street	NULL	Glasgow	NULL	G1 8GH	NULL	NULL	1	Dealer	Large
8	Rocky Riding	5205 108th Ave	NULL	New York	New York	NULL	NULL	NY	3	Dealer	Large
9	Voitures Diplomatiques S.A.	NULL	NULL	Geneva	NULL	NULL	NULL	NULL	6	Wholesaler	Large
10	Karz	NULL	NULL	Stuttgart	NULL	NULL	NULL	NULL	4	Dealer	Small
11	Costa Del Speed	NULL	NULL	Madrid	NULL	NULL	NULL	NULL	5	Dealer	Small
12	Olde Englande	NULL	NULL	Shrewsbury	NULL	SY10 9AX	West Midlands	TF	1	Dealer	Small
13	Impressive Wheels	NULL	NULL	Liverpool	NULL	L5 9ZZ	North West	L	1	Wholesaler	Large
14	Smooth Riders	NULL	NULL	Telford	NULL	TF6 9RR	West Midlands	NULL	1	Dealer	Large
15	Luxury Rentals	NULL	NULL	Gloucester	NULL	GL7 9AS	South West	GL	1	Dealer	Small
16	Premium Motor Vehicles	NULL	NULL	Newcastle upon Tyne	NULL	NE3 3SS	North East	NE	1	Dealer	Small
17	Chateau Moi	NULL	NULL	Lyon	NULL	69001	NULL	NULL	2	Dealer	NULL
18	Vive la Vitesse!	NULL	NULL	Marseille	NULL	13002	NULL	NULL	2	Dealer	NULL
19	Carosse Des Papes	NULL	NULL	Avignon	NULL	84000	NULL	NULL	2	Wholesaler	NULL
20	Three Country Cars	NULL	NULL	Basle	NULL	NULL	NULL	NULL	6	Dealer	NULL
21	Jungfrau	NULL	NULL	Zurich	NULL	NULL	NULL	NULL	6	Wholesaler	NULL
22	Ambassador Cars	11 Times Square	NULL	Bellevue	Washington	NULL	NULL	WA	3	NULL	NULL
23	Embassy Motors	57595 Technology Way	NULL	Denver	Colorado	NULL	NULL	CO	3	NULL	NULL
24	Style 'N Ride	5404 Wisconsin Ave	NULL	Chevy Chase	Maryland	NULL	NULL	MA	3	NULL	NULL
25	BritWheels	1414 NW Northrup Street	NULL	Portland	Oregon	NULL	NULL	OR	3	NULL	NULL
26	Sporty Types Corp	13550 Market Street	NULL	San Francisco	California	NULL	NULL	CA	3	NULL	NULL
27	Tweedy Wheels	One Oxford Center	NULL	Cambridge	Massachussets	NULL	NULL	MA	3	NULL	NULL
28	Union Jack Sports Cars	9900 Corporate Campus Dr	NULL	Louisville	Kentucky	NULL	NULL	KY	3	NULL	NULL
29	Buckingham Palace Car Services	4605 Duke Drive	NULL	Mason	Ohio	NULL	NULL	OH	3	NULL	NULL
30	British Luxury Automobile Corp	2555 Meridian Blvd	NULL	Franklin	Tennesee	NULL	NULL	TN	3	NULL	NULL

TABLE: Clients (31 rows)

Figure 10-1. *Data View*

■ **Note** Modeling data is only possible once data has been loaded into the Power BI Desktop data model. By this I mean that editing data in the Query Editor and returning to the Power BI Desktop window without applying your changes will hinder your data modeling—or at the very least will prevent you from returning accurate results.

Data Model or Query?

You could well be thinking that Data View does not look like very much at all yet. If anything, it looks like an extension of the Power BI Desktop Query window. However, be reassured, you will soon see what you can do in Data View, and exactly how powerful a tool Power BI Desktop really is when it comes to data modeling. For the moment, it is essential to remember that you have (so to speak) opened a door into the engine room of Power BI Desktop. Although this new world is part of Power BI Desktop (and you can return to Power BI visualizations instantaneously just by clicking the Report View icon above the Data View icon), it is best if you consider it a kind of parallel universe for the moment. This universe has its own ribbons and buttons and is separate from the dashboard View so that you can concentrate on enhancing the data and data structure without getting distracted.

The Power BI Desktop Data View is also different from the Power BI Desktop Query Editor. For instance, one major difference between the Power BI Desktop Query window and the Data View is that in Data View you are looking at *all* your data. The Query window only ever shows a *sample* of data. Once you load the data into the data model, you finally have access to the *entire* dataset (all the data in all the tables), and this is what you can see in the Data View.

A second difference is that once you have left the Query window, you are always working with the entire dataset and only filtering it as required by specific dashboards or visualizations. So you need to be aware that any filters that you apply in the Query window limits the data that can be displayed in Power BI Desktop. By contrast (and as you will see later in this book), you can apply subsequent filters to entire dashboards or specific visualizations once you have defined your core data and created the data model.

There are some areas where Power BI Desktop Query can do some of the things that can also be done in the Data View or Relationships View. For instance, you can create calculated columns in both (you will see how to do this in Data View in the following chapter). My advice is to try to remember that Power BI Desktop Query is for finding, filtering, and mashing up data, whereas Data View is for refining and calculating the metrics in your data model. However, each user can take the approach that they prefer and carry out any necessary calculations in either the Query Editor or the Data View. After all, it is the result that counts.

■ **Note** *Always* make sure that you have clicked Close & Apply when using the Query Editor *before* modeling the data. This guarantees that you are working on the most up-to-date version of the data transformations that you have made, as well as the most recent data.

The Power BI Desktop Data View Ribbons

So what, exactly, are you looking at when you start Power BI Desktop? Essentially, you can see two ribbons, which are all devoted to data management:

- The Home ribbon
- The Modeling ribbon

Since the Home ribbon is principally used when creating reports and dashboards, I will explain it in Chapter 14. I prefer to explain the Power BI Desktop Modeling ribbon as we start out in this chapter, because you will use it extensively in the next few pages. This should make understanding the Power BI Desktop environment easier. However, if you prefer to skip this section (or possibly use it as reference later), then feel free to jump ahead.

The Modeling Ribbon

The Modeling ribbon is used to categorize and organize data and tables as well as to add calculated columns and additional metrics. The buttons that it contains are shown in Figure 10-2 and explained in Table 10-1.

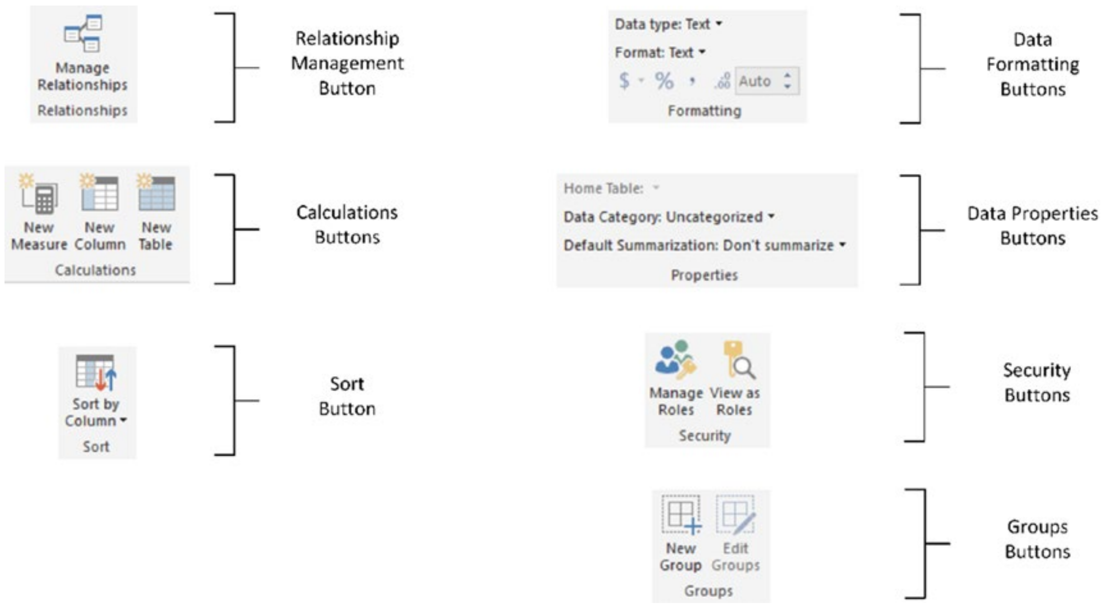


Figure 10-2. *Buttons in the Modeling ribbon*

Table 10-1. *The Modeling Ribbon Buttons*

Button	Description
Manage Relationships	Lets you join tables as well as delete these joins (called *relationships*) and modify them.
New Measure	Used to add a new value or calculation to a table.
New Column	Adds a new calculated column to a table.
New Table	Adds a new table
Sort By Column	Specifies that the data in one column be used as the basis for sorting data in another column.
Data Type	Lets you define the data type for a column.
Format	Lets you specify how numbers are formatted.
Home Table	Lets you select the table that will contain a measure.
Data Category	Allows you to define that a certain column is of a specific category. It is mostly used with geographical data for mapping.
Default Summarization	Lets you define the default aggregation that is applied in dashboard visualizations.
Manage Roles	Allows you to restrict data access for given users.
View As Roles	Lets you see the data available for a given role or roles.
New Group	Allows you to create groups from the values in a column.
Edit Groups	Allows you to modify groups from the values in a column.

Managing Power BI Desktop Data

Assuming that all has gone well, you now have a series of tables from various sources successfully added to your Power BI Desktop data model. If you have opened the file C:\PowerBiDesktopSamples\CH10\CarSalesDataFromDataModel.pbix you can see a set of data tables in the Fields list at the right of the Power BI Desktop window. Clicking a table will display the data from that table in the central area of the Data View window. It will soon be time to see what you can do with this data, but first, to complete the roundup of overall data management, you need to know how to do the following:

- Rename tables
- Delete tables
- Move tables
- Move around a table
- Rename a column
- Delete columns
- Set column width

■ **Note** When loading and editing source data, I generally use the term dataset to describe the source data. Now that the data is finally loaded, I will use the term (data) table to describe the data entities that Power BI Desktop uses when shaping and extending the data model. This is because a table is, virtually always, the output resulting from all transformations produced by a query.

Manipulating Tables

Let's begin by seeing how you can tweak the tables that you have successfully imported and transformed.

Renaming Tables

Suppose that you wish to rename a table that you previously imported using Power BI Desktop Query. These are the steps to follow:

1. Right-click the table name in the Fields list at the right of the Power BI Desktop window.
2. Select Rename. The current name will be highlighted in the Fields list.
3. Enter the new name or modify the existing name.
4. Press Enter.

This also renames the query on which the table is based. In essence, Power BI will let you rename datasets either as queries in the Power BI Desktop Query window or in the Data window. Because the query *is* the table, renaming one renames the other.

Deleting a Table

Deleting a table is virtually identical to the process of renaming one—you just right-click the table name tab and select Delete instead of Rename. As this is a potentially far-reaching operation, Power BI Desktop will demand confirmation.

■ **Note** When you delete a table, you are removing it from Power BI Desktop completely. This means that it is *also* removed from the set of queries that you may have used to transform the data. So you need to be careful when deleting tables, because you could lose all your carefully wrought transformation steps as well.

Selecting a Column from the List of Available Column Names

If you want to leap straight to a column of data (and presuming you know which column contains the data that interests you), all you have to do is

1. In the Fields list, click the expand triangle to the left of the table containing the field that you want to jump to. A list of the fields in the table appears, as shown in Figure 10-3.

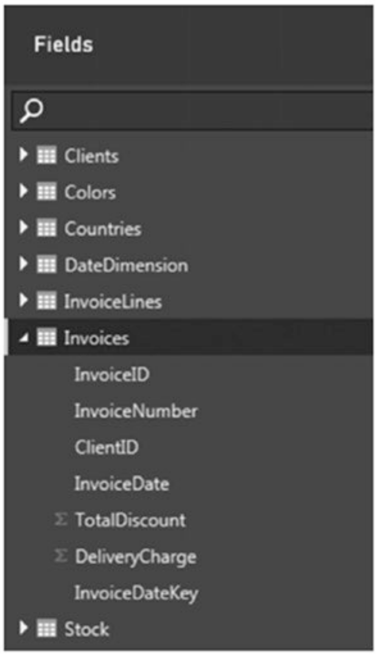


Figure 10-3. *The list of columns in a table*

2. Click the name of the field that contains the data that you want to study. The field will be selected in the table.

Manipulating Columns

Now let's see how to perform similar actions—but this time inside a table—to the columns of data that make up the table.

Renaming a Column

Renaming a column is pretty straightforward. All you have to do is

1. In the Fields list, right-click the title of the column that you wish to rename. In this example, it is the field ClientType in the Data Clients table. The column will be selected and the context menu will appear. This is shown in Figure 10-4.

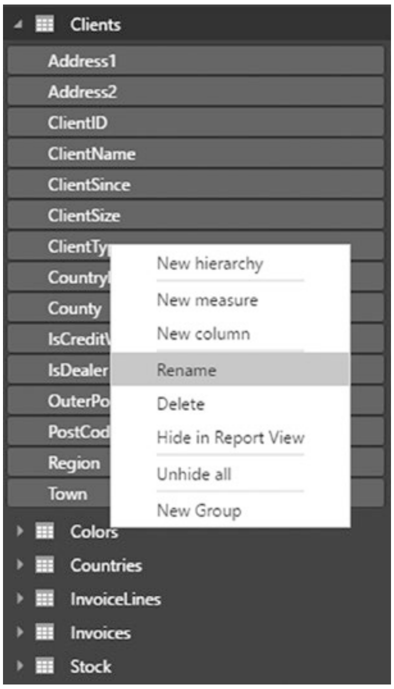
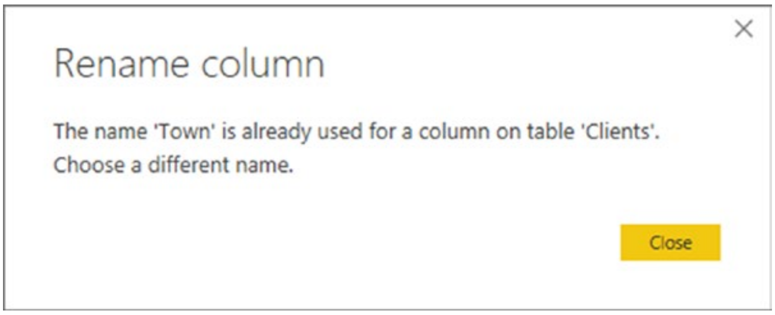


***Figure 10-4.** The Power BI Desktop context menu*

2. Select Rename from the context menu. The current column name will be highlighted.
3. Type in the new name for the column (**TypeOfClient**) and press Enter.

And that is it. Your column has been renamed; it is also selected. You cannot use the name of an existing column in the same table, however. If you try, Power BI Desktop will display the dialog that you can see in Figure 10-5.



Rename column

The name 'Town' is already used for a column on table 'Clients'.
Choose a different name.

Figure 10-5. *The Rename column dialog*

If this happens, just click Close and start over using a different new column name.

Although renaming a column may seem trivial, it can be important. Consider other users first; they need columns to have instantly understandable names that mean something to them. Then there is the PowerBI.com Q&A natural language feature. This only works well if your columns have the sort of names that are used in the queries—or ones that are recognizable synonyms. Finally, you cannot rename columns individually in separate visualizations, so you really need to give your columns the names that you are happy seeing standardized across all the dashboards that are based on this dataset. The upside to this is that the same data will always have the same name, thus removing potential incomprehension or confusion on the part of the audience-and yourself.

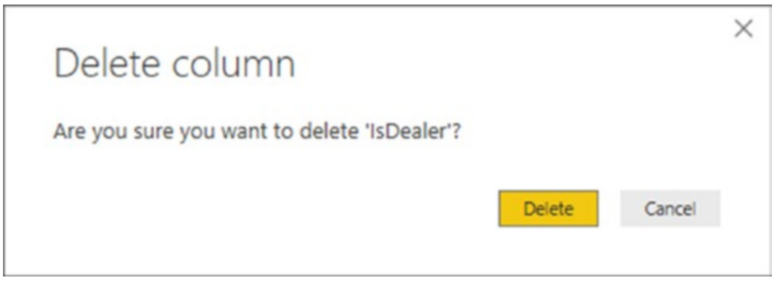
You can also rename a column by right-clicking the column title that interests you and selecting Rename from the context menu.

■ **Tip** Power BI Desktop is very forgiving when it comes to renaming columns (or, indeed, tables and calculations too). You can rename *most* elements in the Query window, the Data View, the Report View, or the Relationships View (depending where they were added) and the changes will ripple through the entire Power BI Desktop data model. Better still, renaming columns, calculations, and tables generally does not cause Power BI Desktop any difficulties if these elements have already been applied to dashboards, as these are also updated to reflect the change of name. Just be aware that if you customize a lot of names and don't document the columns for reference, you will find that reverse engineering a report can be quite a task. In practice, it can save you a lot of effort if you always keep a reference of alias columns back to source in the query.

Deleting Columns

Deleting a column is equally easy. You will probably find yourself doing this when you bring in a column that you did not mean to import or when you find that you no longer need a column. So, to delete a column, you need to do the following:

1. Right-click the title of the column that you wish to delete. Alternatively, right-click the field name in the Fields list. The column will be selected and the context menu will appear.
2. Select Delete Columns from the context menu. Unless the column is empty, the confirmation dialog will appear as shown in Figure 10-6.


Delete column

Are you sure you want to delete 'IsDealer'?

Figure 10-6. *Deleting a column*

3. Click OK. The column will be deleted from the table.

Deleting unused columns is good practice because you will

- Reduce the memory required for the dataset.
- Speed up data refresh operations.
- Reduce the size of the Power BI Desktop file.

■ **Note** Deleting a column really is permanent. You cannot use the undo function to recover it. Indeed, refreshing the data will not add the column back into the table either. If you have deleted a column by accident, you can choose to close the Power BI Desktop file without saving and reopen it, thus reverting to the previous version. Otherwise, you can return to the source query and add the column name back into the query.

Moving Columns

Once in the data model, you *cannot* move columns around. So if you want to change the column order for any reason, you have to switch to the Power BI Desktop Query Editor and move the column there. Once you save and apply your changes, the modified column order is visible in the data model. However, columns will always appear in alphabetical order in the Fields list in Report View.

Setting Column Widths

One final thing that you may want to do to make your data more readable—and consequently easier to understand—is to adjust the column width. I realize that as an Excel or Word user, you may find this old hat, but in the interests of completeness, this is how you do it:

1. Place the mouse pointer over the right-hand limit of the column title in the column whose width you want to alter. The cursor will become a two-headed arrow.
2. Drag the cursor left or right.

■ **Note** You cannot select several adjacent columns before widening (or narrowing) one of them to set them all to the width of the column that you are adjusting. You can double-click the right-hand limit of the column title in the column whose width you want to alter so that Power BI Desktop can set the width to that of the longest element in the column.

Power BI Desktop Data Types

When you are importing data from an external source, Power BI Desktop tries to convert it to one of the nine data types that it uses. These data types are described in Table 10-2.

Table 10-2. *Power BI Desktop Data Types*

Data Type	Description
Decimal Number	Stores the data as a real number with a maximum of 15 significant decimal digits. Negative values range from –1.79E +308 to –2.23E –308. Positive values range from 2.23E –308 to 1.79E + 308.
Fixed Decimal Number	Stores the data as a number with a specified number of decimals.
Whole Number	Stores the data as integers that can be positive or negative but are whole numbers between 9,223,372,036,854,775,808 (–2^63) and 9,223,372,036,854,775,807 (2^63–1).
Date/Time	Stores the data as a date and time in the format of the host computer. Only dates on or after the 1st of January 1900 are valid.
Date	Stores the data as a date in the format of the host computer.
Time	Stores the data as a time element in the format of the host computer.
Text	Stores the data as a Unicode string of 536,870,912 bytes at most.
True/False	Stores the data as Boolean—true or false.
Binary	Stores the data as binary (machine-readable) data.

Formatting Power BI Desktop Data

Power BI Desktop allows you to apply formatting to the data in the tables that it contains. When you format the data in the Power BI Desktop data model, you are defining the format that will be used in *all visualizations in all the dashboards* that you create using this metric. So it is probably worth learning to format data for the following reasons:

- You will save time and multiple repetitive operations when creating reports and presentations by defining a format once and for all in Data View. The data will then appear using the format that you applied in multiple visualizations in this Power BI Desktop file.
- It can help you understand your data more intuitively if you can see the figures in a format that has intrinsic meaning.

This explains how to format a column (of figures in this example):

1. Assuming that you are working in the Power BI Desktop Data View, click the table name in the Fields list that contains the metric you wish to format.
2. Click inside the column that you want to format (SalePrice from the InvoiceLines table in the CarSalesDataFromDataModel.pbix sample file).
3. In the Modeling ribbon, click the Thousands Separator icon (the comma in the Formatting section). All the figures in the column will be formatted with a thousands separator and two decimals.

The various formatting options available are described in Table 10-3.

Table 10-3. *Currency Format Options*

Format Option	Icon	Description	Example
General		Leaves the data unformatted.	100000.01
Currency	$	Adds a thousands separator and two decimals as well as the current monetary symbol.	$100,000.01
Date/Time		Formats a date and/or time value in one of a selection of date and time formats.	
Decimal Number	,	Adds a thousands separator and two decimals.	100,000.01
Whole Number		Adds a thousands separator and truncates any decimals.	100,000
Percentage	%	Multiplies by 100, adds two decimals, and prefixes with the percentage symbol.	28.78%
Scientific		Displays the numbers in scientific format.	1.00E+05
Text		Formats the column contents as text.	
Binary		Leaves the column contents as a binary representation of the data.	
True/False		Formats the column contents as True/False.	
Decimal Point	.00 0	Increases or reduces the number of decimals.	

If you wish to return to “plain vanilla” data, then you can do this by selecting the General format. Remember that you are not in Excel, and you cannot format only a range of figures—it is the whole column or nothing. Also, there is no way to format nonadjacent columns by Ctrl-clicking to perform a noncontiguous selection. And, you cannot select multiple adjacent columns and format them in a single operation.

By now, you have probably realized that Power BI Desktop operates on a “Format once/apply everywhere” principle. However, this does not mean that you have to prepare the data exhaustively before creating dashboards and reports. You can flip between the data model and the Report View at any time to select another format, secure in the knowledge that the format that you just applied is used throughout all of your dashboards wherever the relevant metric is used.

■ **Note** Numeric formats are not available for selection if the data in a field is of text or data/time data type. Similarly, date and time formats are only available if the column contains data that can be interpreted as dates or times. Only Boolean (True/False) data types can be formatted as True/False. Finally, only columns containing binary data can be formatted as binary.

Currency Formats

Power BI Desktop will propose a wide range of currency formats. To choose the currency that you want:

1. Click the popup (the downward-facing triangle) to the right of the currency format icon. This will display a list of available formats.
2. Select the currency symbol that you want, or scroll through the list to view all the available currency formats, as shown in Figure 10-7, then click OK.

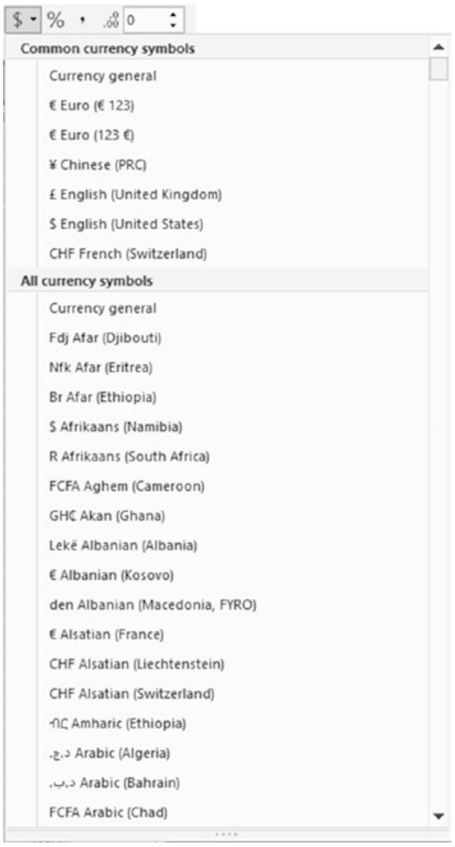


Figure 10-7. *The currency format popup list*

■ **Note** The thousands separator that is applied, as well as the decimal separator, depends on the settings of the PC on which the formatting is applied.

Preparing Data for Dashboards

Corralling data into a structure that can power your dashboards necessitates a good few tweaks above and beyond specifying data types and formats for final presentation. As part of the groundwork for your dashboards, you could also have to

- Categorize data.
- Apply a default summarization.
- Define Sort By columns.
- Hide fields.

These ideas probably seem somewhat abstruse at first sight, so let's see them in action to make it clear why you need to add these touches to your data model.

Categorizing Data

Power BI dashboards are not just made up of facts and figures. They can also contain geographical data or hyperlinks to web sites or documents. While we humans can recognize a URL pretty easily and we can guess that a column with postcodes contains, well, postcodes, such intuitions may not be quite as self-evident for a computer.

So, if you want Power BI to be able to add maps or hyperlinks (for instance), you will make life easier for both you and the application if you categorize any columns that contain the types of data that are used for maps and links.

For instance, suppose that you want to prepare the Countries table as a potential data source for a dashboard map (and assuming that you have loaded the CarSalesDataFromDataModel.pbix sample file):

1. In the Data View, select the Countries table in the Fields list. The data in the Countries table will appear in the center of the Power BI Desktop window.
2. Click inside the CountryName column. The column will be highlighted.
3. In the Modeling ribbon, click the popup to the right of the Data Category button. You can see this context menu in Figure 10-8.

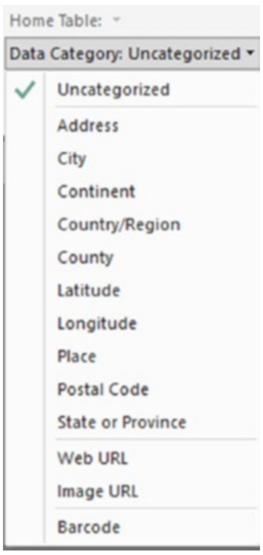


Figure 10-8. *The Data Category options*

4. Select Country/Region from the menu.

The ribbon will show Data Category: Country/Region. This means that Power BI Desktop now knows to use the contents of this field as a country when generating maps in dashboards.

The Data Category options that are available are described in Table 10-4.

Table 10-4. *Data Category Options*

Data Category Option	Description
Uncategorized	Applies to data that is not used for hyperlinks or creating maps.
Address	Specifies an address for mapping.
City	Specifies a city for mapping.
Continent	Specifies a continent for mapping.
Country/Region	Specifies a country or region for mapping.
County	Specifies a county for mapping.
Latitude	Specifies a latitude for mapping.
Longitude	Specifies a longitude for mapping.
Place	Specifies a location or place for mapping.
Postal Code	Specifies a postal (ZIP) code for mapping.
State or Province	Specifies a state or province for mapping.
Web URL	Indicates a URL for a hyperlink.
Image URL	Indicates a URL for the source of an image in a dashboard.
Barcode	Specifies that the field contains a barcode.

■ **Note** Not specifying a data category does not mean that Power BI Desktop cannot create maps in dashboards or recognize URLs. However, the results cannot be guaranteed of a reasonable chance of success unless you have indicated to the application that a column contains a certain type of data.

Applying a Default Summarization

When you are creating dashboards, you are often aggregating numeric data. Most times, this means adding up the figures to return the column total. However, there could be columns of data where you want another aggregation applied. Do the following to set your own default aggregation (assuming that you have loaded the CarSalesDataFromDataModel.pbix sample file):

1. In the Data View, select the Colors table in the Fields list. The Colors data will appear in the center of the Power BI Desktop window.
2. Click inside the Color column. The column will be highlighted.
3. In the Modeling ribbon, click the popup to the right of the Default Summarization button. The Default Summarization popup menu appears, as shown in Figure 10-9.

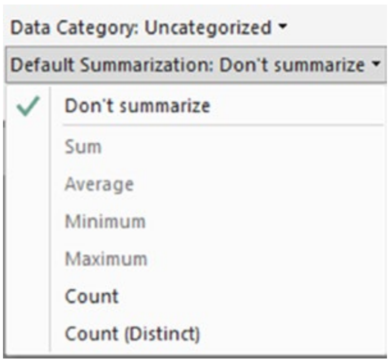

Data Category: Uncategorized
Default Summarization: Don't summarize
Don't summarize
Sum
Average
Minimum
Maximum
Count
Count (Distinct)

Figure 10-9. *The Default Summarization popup menu*

4. Select Count from the context menu.

The ribbon will show Default Summarization: Count.
The Default Summarization options that are available are described in Table 10-5.

Table 10-5. *Default Summarization Options*

Default Summarization Option	Description
Don't Summarize	The data in this column is not summarized.
Sum	The data in this column is added (summed).
Average	The average value for the data in this column is returned.
Minimum	The minimum value for the data in this column is returned.
Maximum	The maximum value for the data in this column is returned.
Count	The number of elements in this column is returned.
Count (Distinct)	The number of individual (distinct) elements in this column is returned.

Obviously, you can only apply mathematical aggregations to numeric values. However, you can apply counts to any type of data.

■ **Note** Specifying a default aggregation does not prevent you from overriding the default in dashboards. It merely sets a default that is applied as a standard when aggregating data from a column. In the real world, this can be really useful because it reduces the time you spend building dashboards.

Defining Sort By Columns

Sometimes you will want to sort data in a dashboard visualization based not on the contents of the selected column, but by the contents of another column. As an example, imagine that you have a table of data that contains the month for a sale. If you sort by month, you probably do not want to see the months in alphabetical order, starting with April. In cases such as this, you can tell Power BI Desktop that you want to sort the month *name* element by the month *number* that is contained in another column.

1. Load the C:\PowerBiDesktopSamples\CH10\CarSalesDataFromDataModel.pbix sample file (unless you have already loaded it, of course).
2. In the Data View, select the DateDimension table in the Fields list. The date data will appear in the center of the Power BI Desktop window.
3. Click inside the MonthFull column. The column will be highlighted.
4. In the Modeling ribbon, click the Sort By Column button and select MonthNum from the list of available columns.

Now if you sort by MonthFull in a visualization, you see the months in the order that you probably expect—from January to December. Had you *not* applied a Sort By column, then calendar months would have been sorted in alphabetical order, which is from April to September! Once again, this choice applies to *any* visualization that you create in a Power BI Desktop dashboard that is based on this data model. So remember to add numeric sort columns alongside textual columns, such as dates and so forth, at the source.

To remove the Sort By column, leave the column selected and simply click the Sort By Column button again and select the original column name from the list of available columns.

■ **Tip** If you want to see which column has been set as a Sort By column, simply click inside the column to be sorted and then click the Sort By Column button. The popup list of columns shows a tick to the left of the column that is being used to sort the selected column.

The sample file for this chapter (CarSalesDataFromDataModel.pbix) already contains columns that you can use as Sort By columns. In the real world, your source data might not always be this instantly useable. So remember that you can always switch to the Power BI Desktop Query Editor and enhance source tables with extra columns that you can then use to sort data in Data View.

■ **Note** In Chapter 13 I will explain ways of creating date tables like the one you just saw.

Hiding Tables and Fields from the User

A data model can conceivably contain many more columns than those that you need to create reports and dashboards. These could include (among others):

- Sort By columns
- Columns that already appear in hierarchies (as you will see in a couple of pages' time)
- Columns that contain intermediate calculations (these are explained in the next chapter)
- Columns that contain data that you have loaded "just in case" but don't need yet

It follows that, in the interests of clarity, you may want to hide columns (or even entire tables) that are not strictly relevant to your analysis. This not only reduces clutter, it can avoid confusion when new users start to work with a data model that you have created. It also guarantees that only essential data will be available.

Hiding Columns and Tables

As a first step, you need to flag columns (or fields, if you prefer) and tables as hidden. Here is how:

1. Click the column or table that you want to hide.
2. Select Hide in Report View from the popup menu.

Hidden columns and tables will still appear in the Fields list for the moment, but will be grayed out. Hidden columns and the columns in hidden tables can, nonetheless, still be used in visualizations.

To redisplay a hidden column or table, simply right-click the requisite column or table and uncheck the Hide in Report View option in the popup menu.

Removing Hidden Columns and Tables from View

If your objective is to "declutter" the Fields list, you can then remove hidden columns and tables from view completely, as follows:

Click any table or field in the Fields list and uncheck the View Hidden option in the popup menu.

All the hidden columns and tables will disappear from the Fields list. To redisplay all the hidden columns and tables, simply check the View Hidden option in the popup menu.

Sorting Data in Power BI Desktop Tables

A Power BI Desktop table could contain millions of rows, so the last thing that you want to have to do is to scroll down through a random dataset. Fortunately, ordering data in a table is simple:

1. Right-click inside the column you want to order the data by. I will choose the Make column in the Stock table.
2. Click the Sort Ascending option in the context menu to sort this column in ascending (alphabetical) order.

The table is sorted using the selected column as the sort key, and even a large dataset appears correctly ordered in a very short time. If you want to sort a table in descending order (reverse alphabetical or largest to smallest order), click the Sort Descending option in the context menu.

At this juncture, you need to remember that the data model is not really designed for interactive data analysis. That is what dashboards are for. Consequently, you should not expect to use the data tables in Power BI Desktop as if they were vast Excel spreadsheets.

■ **Tip** If you need a visual indication that a column is sorted, look at the right of the column name. You will see a small arrow that faces upward to indicate a descending sort or downward to indicate an ascending sort.

If you want to remove the sort operation that you applied and return to the initial dataset as it was imported, all you have to do is click the Clear Sort icon in the context menu for the column.

■ **Note** You cannot perform complex sort operations; that is, you cannot sort first on one column, then carry out a secondary sort in another column (if there are identical elements in the first column). You also cannot perform multiple sort operations sorting on the least important column and then progressing up to the most important column to sort on to get the effect of a complex sort. This is because Power BI Desktop always sorts the data based on the dataset as it was initially loaded. Remember that you can add index columns (as you saw in Chapter 7) in Power BI Desktop Query and then sort on these if you want to reapply an initial sort order.

Adding Hierarchies

Organizing data can be fundamental when you want to "see the wood for the trees." Consequently Power BI Desktop lets you create any hierarchy on the fly so that you can better appreciate the structure of the information that you are presenting.

To create a hierarchy:

1. Switch to Report View.
2. Select the Stock table.

3. Right-click the field that will become the top-level element in the new hierarchy (Make in this example).

4. Select New Hierarchy. A new hierarchy named Make Hierarchy will appear in the Stock table, containing the Make field as the top-level element in the hierarchy.

5. Drag the Model field onto the new hierarchy title. The Model field will be added to the hierarchy under the highest-level element. You can see this in Figure 10-10.

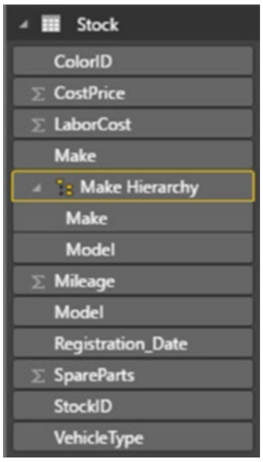


Figure 10-10. *An added hierarchy*

This is all that you have to do! You can now drag the hierarchy on to the report canvas to create a table that is based on the multiple elements that make up the hierarchy. The data can be used anywhere—and in any visual—where the multiple fields that make up the hierarchy would be used. They are particularly useful as the basis for matrices and drill-down charts, which you will discover in Chapters 15 and 16.

You can, of course, rename or delete the hierarchy or any level in a hierarchy just as you would any standard data field.

■ **Tip** Adding a field to a hierarchy means that the field now exists twice in the table-once as a "stand-alone" field and once inside a hierarchy. In most cases it is best to hide the original version of the field so that users (and you) don't see the field twice in the data model.

Creating and Modifying Groups

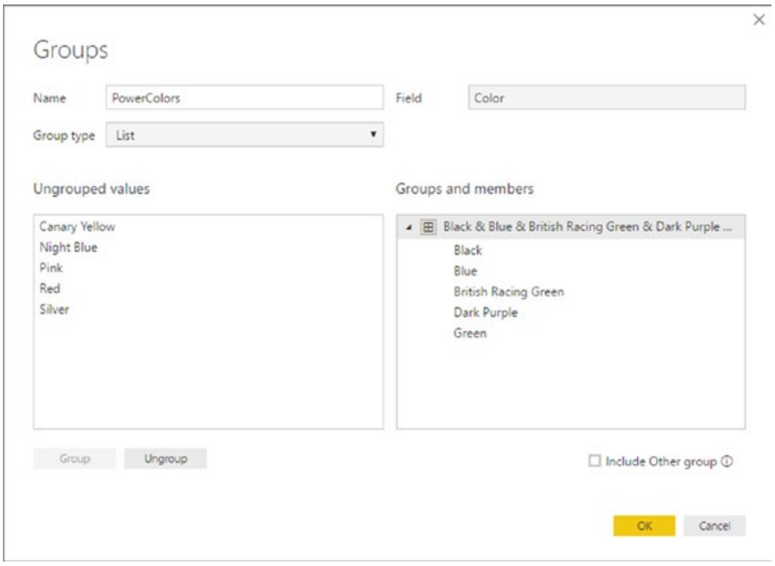
There could be times when you have multiple individual items in a list and you wish to gather separate sets of values into groups. This could be to get a higher-level view of separate subdomains or simply to isolate certain values from a recordset.

■ **Tip** If you like, you can consider that groups are to "horizontal" organization what hierarchies are to "vertical" selection.

Creating a Group

Suppose that you will frequently be using a set of colors as the basis for tables and charts in your dashboards. Here is how to create a group containing a subset of the available colors:

1. Click the Data icon on the top left of the Power BI Desktop window.
2. Select the Colors table.
3. Click inside the Color column.
4. In the Modeling ribbon, click the New Group icon. The Groups dialog will appear.
5. In the Name field, enter a name for the group. I will use PowerColors in this example.
6. In the list of ungrouped values on the left, Control-click the following five elements:
    a. Black
    b. Blue
    c. British Racing Green
    d. Dark Purple
    e. Green
7. Click Group. The dialog will look like the one shown in Figure 10-11.


Groups

Name: PowerColors
Field: Color
Group type: List

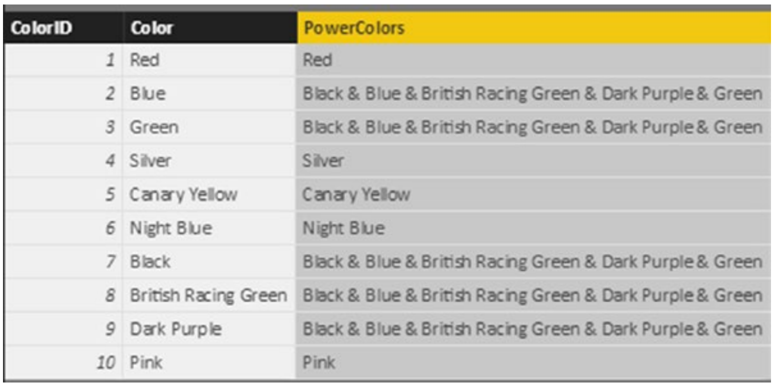
Ungrouped values:
Canary Yellow
Night Blue
Pink
Red
Silver

Groups and members:
Black & Blue & British Racing Green & Dark Purple ...
Black
Blue
British Racing Green
Dark Purple
Green

Include Other group

Figure 10-11. *The Groups dialog*

8. Click OK. A new column will be created containing the group definitions, as you can see in Figure 10-12. The name of the group becomes the new column name.



ColorID	Color	PowerColors
1	Red	Red
2	Blue	Black & Blue & British Racing Green & Dark Purple & Green
3	Green	Black & Blue & British Racing Green & Dark Purple & Green
4	Silver	Silver
5	Canary Yellow	Canary Yellow
6	Night Blue	Night Blue
7	Black	Black & Blue & British Racing Green & Dark Purple & Green
8	British Racing Green	Black & Blue & British Racing Green & Dark Purple & Green
9	Dark Purple	Black & Blue & British Racing Green & Dark Purple & Green
10	Pink	Pink

Figure 10-12. *A group column*

You can now use this column in visuals to group elements. You will see this in greater detail in Chapter 16. There a couple of final points to note when creating groups:

- Clicking the Include Other Group check box will take all remaining items in the original column and use them to create an "Other" group.
- You can create multiple groups. Simple carry out steps 6 and 7 as many times as you want separate groups.
- No value from the original column can appear more than once in a group.

Modifying a Group

Any existing group can be altered easily, simply by doing the following:

1. Click inside the column containing the group that you want to modify.
2. In the Modeling ribbon, click the Edit Groups icon. The Groups dialog will appear.
3. Remove any items from the group by clicking the value in the Groups and Members list on the right. Then click Ungroup.
4. Add any new values by selecting the item on the left and, once you have selected the appropriate group on the right, clicking the Group button.
5. Click OK. The modified group will be applied to the selected column.

Deleting a Group

As a group is, to all intents and purposes, a column, you delete a group exactly as you would delete a column:

1. In the Fields list, expand the table containing the new column. You will see that group columns have a small icon with two empty squares to the left of the field name.
2. Right-click the group column and select Delete.
3. Confirm the deletion.

■ **Note** You can "mix and match" groups in a table. You can, for instance, allocate all the separate values in a table to different groups, or you can create multiple groups to handle certain values and nonetheless leave other values outside a group.

Designing a Power BI Desktop Data Model

Congratulations! You are well on the way to developing a high-performance data model for self-service business intelligence (BI). You have imported data from one or even from several sources into the Power BI Desktop data model. You have taken a good look at your data using the Power BI Desktop data model window and you can carry out essential operations to rename tables and columns. The final step to ensure that your dataset is ready for initial use as a self-service BI data repository is to create and manage relationships between tables. This is a fundamental part of designing a coherent and useable dataset in Power BI Desktop.

Before leaping into the technicalities of table relationships, we first need to answer a couple of simple questions:

- What are relationships between tables?
- Why do we need them?

Table relationships are links between tables of data that allow columns in one table to be used meaningfully in another table. If you have opened the Power BI Desktop example file CarSalesDataFromDataModel.pbix, then you can see that there is a Stock table of stock data that contains a ColorID column, but not the actual color itself. As a complement to this, there is a reference table of colors, named Colors. It follows that, if we want to say what color a car was when it was bought, we need to be able to link the tables so that the Stock table can "look up" the actual color of the car that was bought. This requires some commonality between the two tables and, fortunately, both contain a column named ColorID. So if we are able to join the two tables using this field, we can see which color is represented by the color ID for each car in stock.

You can see another example of linking tables if you take a look at the Invoices and InvoiceLines tables. These two tables have been designed using a technique called relational modeling. Essentially this means that two tables have been created to avoid pointless data duplication. So any data that is used to describe an invoice (such as the invoice date or invoice number) is stored in the Invoices table, whereas all the details concerning the vehicles sold are held in a separate table named InvoiceLines. The two tables then share a field that allows them to be joined so that users can see the data from the two tables together if they need to. This means that any elements that would otherwise be repeated are stored in a "header" table such as the Invoices table and nonrepeating data is stored in a detail table (the InvoiceLines table in this example).

It is possible to store the data from these two tables as one table. However, this would mean repeating elements such as the invoice date or invoice number each time that an invoice contained more than one item. This would entail repeated data elements and vastly increased file sizes.

Clearly, these examples are extremely simple. However, they are not unduly contrived. They represent the way many relational databases store data. So there is every chance that you will see potential links or relationships like this in the real-world data that you import from corporate databases. In any case, if you want to use data from multiple sources in your data analysis, you will have to find a way to link the tables using a common field, like the ColorID field that I just mentioned. The reality may be messier (the fields may not have the same name in the two tables, for instance), but the principle always applies.

■ **Tip** If you have the necessary permissions as well as the SQL knowledge, then you can join tables directly in the source database using a query. This way you can create fewer "flattened" tables in Power BI Desktop from the start and avoid having to create a spider's web of new table relationships in Power BI Desktop.

Data View and Relationship View

Power BI Desktop lets you see your data model in two different yet complementary ways. When you need a high-level overview of all the data tables, you should use Relationship View, as this allows you to step back from the detail and look at the dataset as a whole. The following explains how to do this:

1. Click the Relationship View button on the left of the Power BI Desktop window. Power BI Desktop will display the tables in Relationship View, as shown in Figure 10-13.

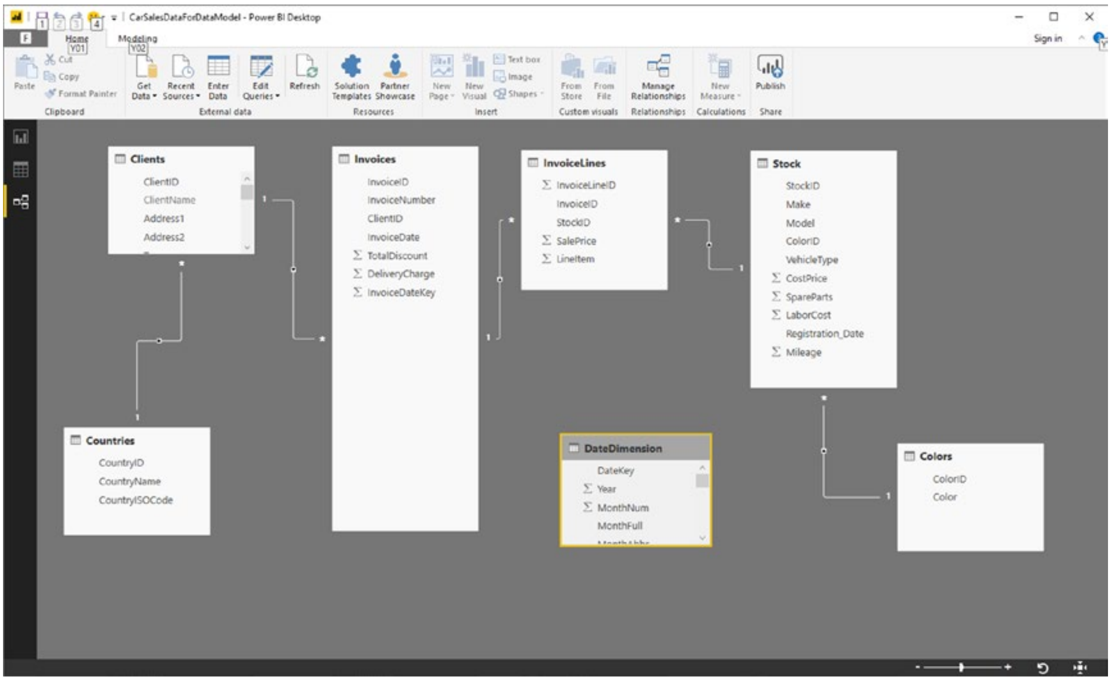


Figure 10-13. Relationship View

As you can see from Figure 10-13, you are now looking at most or all of your tables, and although you can see the table and column names, you cannot see the data. Not only that, but you can also see that some tables are already joined—though not all of them. This is because Power BI Desktop always attempts to guess any possible relationships between tables and creates relationships automatically, if it can, to save you time and effort.

This is exactly what we need, because now it is time to think in terms of overall structures rather than the nitty-gritty. You can use this view to move and resize the tables. Moving a table is as easy as dragging the table's title bar. Resizing a table means placing the pointer over a table edge or corner and dragging the mouse.

■ **Tip** Although repositioning tables can be considered pure aesthetics, I find that doing so is really useful. A well-laid-out dataset design helps you understand the relationships between the tables and the inherent structure of the data.

Relationship View Display Options

The whole point of Relationship View is to let you get a good look at the entire dataset and, if necessary, modify the layout in order to see the relationships between tables more clearly.

Maximizing a Table

If you have many, many fields in a table, then you may occasionally need to take a closer look at a single table. Fortunately, the creators of Power BI Desktop have thought of this. So, to zoom in on a specific table:

1. Click the table that you wish to examine more closely. In this example, it will be the Invoices table.
2. Click the Maximize button to the right of the table name. The table will expand to give you a clearer view of the fields in the table, whereas the rest of the data model will be grayed out. You can see an example of this in Figure 10-14.

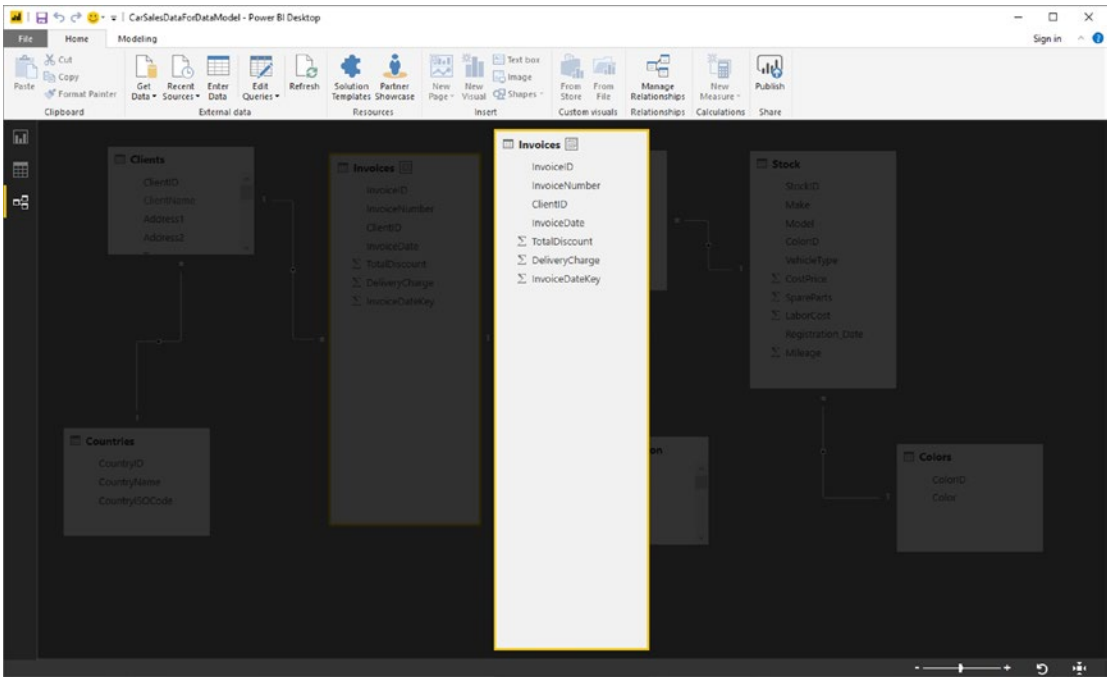


Figure 10-14. Maximizing a table

Minimizing a Table

To reset the table to its previous size, click the same icon—now called the Restore button—at the right of the table name.

Creating Relationships

Creating relationships is easy once you know which fields are common between tables. Since we already agreed that we need to join the Colors table to the Stock table, let's look at how to do this using the file C:\PowerBiDesktopSamples\CH10\CarSalesDataForLinkingTables.pbix. This file contains the data tables that you saw up until now in this chapter, but without any of the relationships between the tables.

1. Open the Power BI Desktop file CarSalesDataForLinkingTables.pbix in the folder C:\PowerBiDesktopSamples\CH10.
2. Drag the ColorID field from the Stock table over the ColorID field in the Colors table, as shown in Figure 10-15.

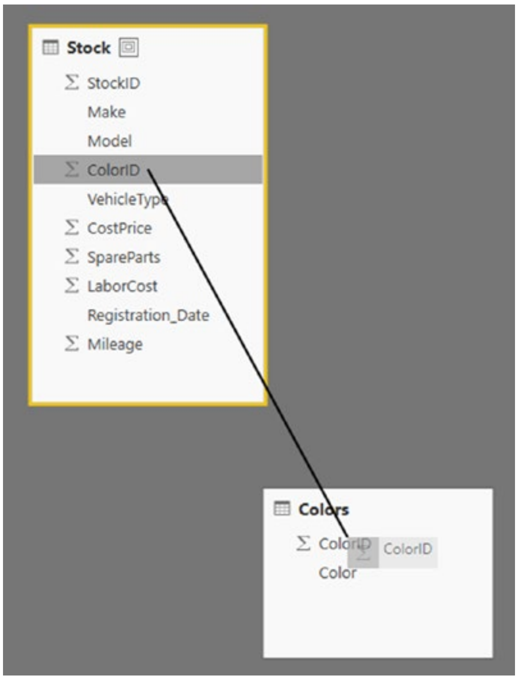


Figure 10-15. *A table relationship*

This is all that you have to do. The two tables are now joined and the data from both tables can be used meaningfully in reports and dashboards.

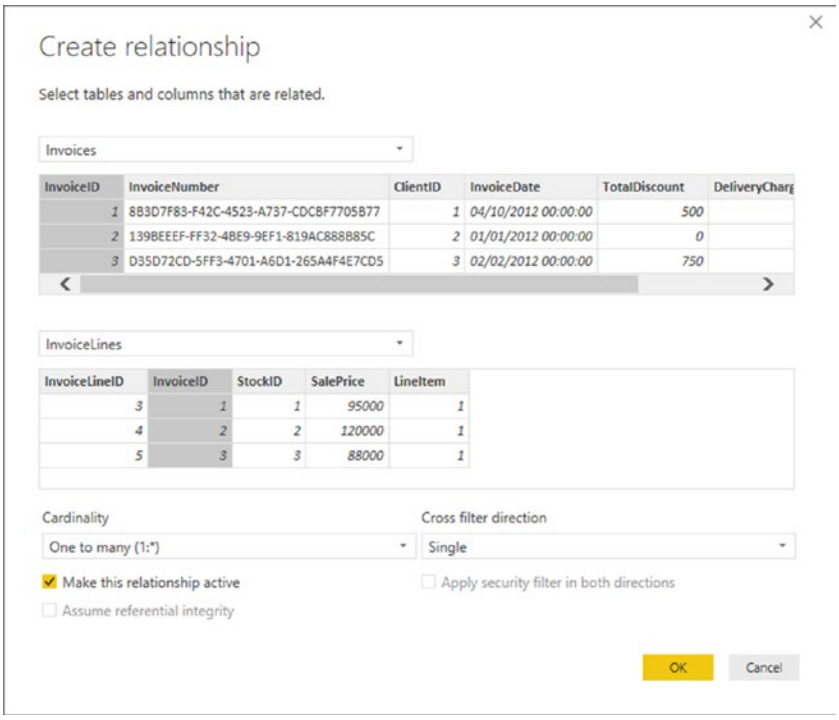
■ **Note** Currently, in the Power BI Desktop data model, you can only join tables on a single field. You may need to take this into account when preparing queries in the Power BI Desktop Query Editor for later use in the data model.

Creating Relationships Manually

You do not have to drag and drop field names to create relationships. If you prefer, you can specify the tables and fields that will be used to create a relationship between tables. What is more, you do not have to be in Relationship View to do this. So, just to make a point and to show you how flexible Power BI Desktop can be, in this example, you will join the Invoices and InvoiceLines tables on their common InvoiceID field:

1. Select a table that you want to make appear in a relationship. I will use the Invoices table for this example.
2. Click the Manage Relationships button in the Home ribbon. The Manage Relationships dialog will appear.
3. Click New. The Create Relationship dialog will appear.
4. In the upper part of the dialog, select the Invoices table from the popup list of tables.

5. In the lower part of the dialog, select the InvoiceLines table as the related lookup table. The InvoiceID field should appear automatically as the field to join on (it will be selected in both tables). If Power BI Desktop has guessed the field, it will appear selected. If it does not, or if it has guessed incorrectly, you can always select the correct field for both tables. The Create Relationship dialog should look like Figure 10-16.


Create relationship

Select tables and columns that are related.

Invoices

InvoiceID	InvoiceNumber	ClientID	InvoiceDate	TotalDiscount	DeliveryCharg
1	8B3D7F83-F42C-4523-A737-CDCBF7705B77	1	04/10/2012 00:00:00	500	
2	139BEEEF-FF32-4BE9-9EF1-819AC888B85C	2	01/01/2012 00:00:00	0	
3	D35D72CD-5FF3-4701-A6D1-265A4F4E7CD5	3	02/02/2012 00:00:00	750	

InvoiceLines

InvoiceLineID	InvoiceID	StockID	SalePrice	LineItem
3	1	1	95000	1
4	2	2	120000	1
5	3	3	88000	1

Cardinality: One to many (1:*)

Cross filter direction: Single

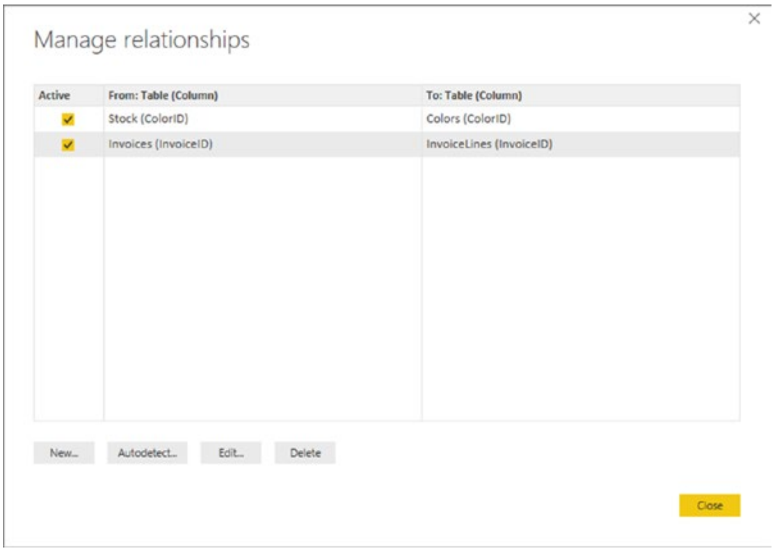
[✓] Make this relationship active

[ ] Apply security filter in both directions

[ ] Assume referential integrity

Figure 10-16. *The Create Relationship dialog*

6. Click OK. The Create Relationship dialog will close and return you to the Manage Relationships dialog. It should look like Figure 10-17 at the moment. You can see that it indicates which column in which table is used to join to which other column in which other table.



Active	From: Table (Column)	To: Table (Column)
✓	Stock (ColorID)	Colors (ColorID)
✓	Invoices (InvoiceID)	InvoiceLines (InvoiceID)

Figure 10-17. *The Manage Relationships dialog*

7. Click Close. The Manage Relationships dialog will close and the relationship will be created.

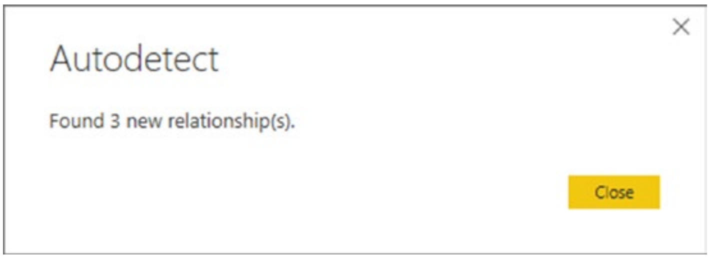
Creating Relationships Automatically

Whatever the data source, you can have Power BI Desktop detect the relationships automatically. This approach has a couple of advantages:

- You avoid a lot of manual work.
- You reduce the risk of error (that is, creating relationships between tables on the wrong fields, or even creating relationships between tables that are not related).

This technique is unbelievably easy. All you do is the following:

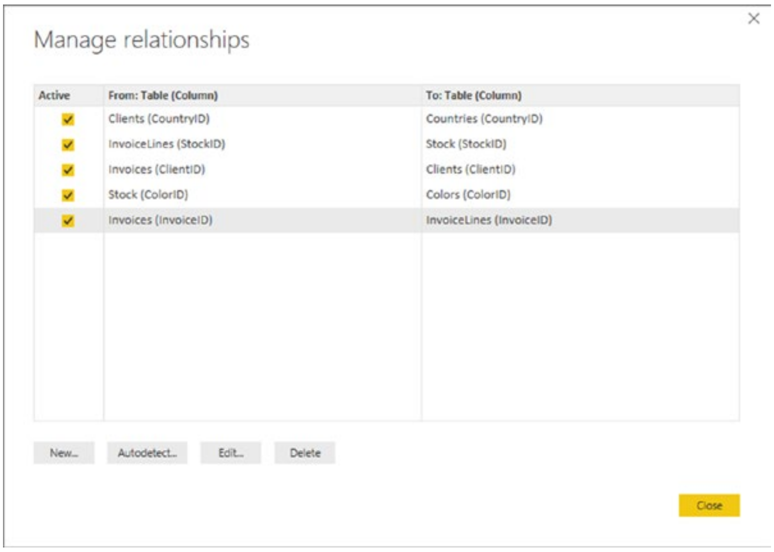
1. Click the Manage Relationships button in the Home ribbon. The Manage Relationships dialog will appear.
2. Click Autodetect. After a few seconds the dialog shown in Figure 10-18 will appear.


Autodetect

Found 3 new relationship(s).

Figure 10-18. *The Autodetect dialog*

3. Click Close. The Manage Relationships dialog will list all the relationships in the data model—both pre-existing relationships and those discovered by the autodetection process. The Manage Relationships dialog will now look like the one in Figure 10-19.


Manage relationships

Active	From: Table (Column)	To: Table (Column)
✓	Clients (CountryID)	Countries (CountryID)
✓	InvoiceLines (StockID)	Stock (StockID)
✓	Invoices (ClientID)	Clients (ClientID)
✓	Stock (ColorID)	Colors (ColorID)
✓	Invoices (InvoiceID)	InvoiceLines (InvoiceID)

Figure 10-19. *The Manage Relationships dialog after automatically detecting relationships*

4. Click Close to return to the Relationships View.

You will see that the tables you just imported already have the relationships generated in Power BI Desktop.

Deleting Relationships

In addition to creating relationships, you will inevitably want to remove them at some point. This is both visual and intuitive.

1. Click the Design View button in the Home ribbon. Power BI Desktop will display the tables in Relationship View.

2. Select the relationship that you want to delete. The arrow joining the two tables will become a double link, and the two tables will be highlighted.

3. Right-click and choose Delete (or press the Delete key). The Confirmation dialog will appear, as shown in Figure 10-20.

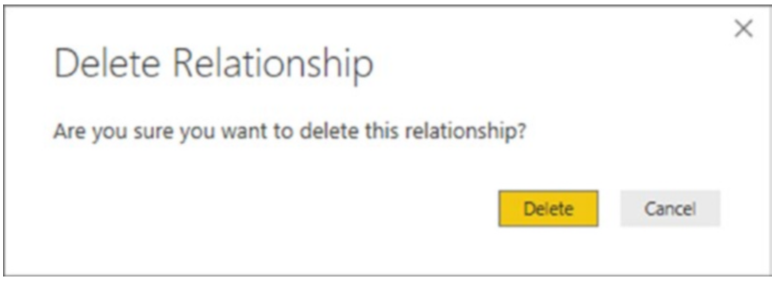


Figure 10-20. *The Delete Relationship dialog*

4. Click Delete.

The relationship will be deleted. However, the tables will remain in the data model.

Managing Relationships

If you wish to change the field in a table that serves as the basis for a relationship, then you have another option. You can use the Manage Relationships dialog. This approach may also be useful if you want to create or delete several relationships at once. If you want to use this dialog:

1. In the Design tab, of the Home ribbon click Manage Relationships. The Manage Relationships dialog appears, as was shown previously in Figure 10-17.
2. Click the relationship you wish to modify.
3. Click Edit (or double-click the relationship). The Edit Relationship dialog appears (it is virtually identical to the Create Relationship dialog shown in Figure 10-16).
4. Continue modifying the relationship as described previously.

As you can see from this dialog, you also have the option of creating or deleting relationships. Since the processes here are identical to those I have already described, I will not repeat them.

The techniques used to create and manage relationships are not, in themselves, very difficult to apply. It is nonetheless *absolutely fundamental* to establish the correct relationships between the tables in the dataset. Put simply, if you try to use data from unconnected tables in a single Power BI Desktop dashboard, not only will you get an alert warning you that relationships need to be created, you will also get visibly inaccurate results. Basically, all of your analyses will be false. So it is well worth it to spend a few minutes upfront designing a clean, accurate, and logically coherent dataset.

■ **Tip** Double-clicking a relationship in the Relationship View will also display the Edit Relationship dialog.

Deactivating Relationships

If you no longer need a relationship between tables but do not want to delete it, you also have the option of deactivating the relationship. This means that the relationship no longer functions, but that you can reactivate it quickly and easily.

Deactivating—or reactivating—a relationship is as simple as selecting or unselecting the box in the Active column of the Manage Relationships dialog (shown earlier in Figure 10-17).

Advanced Relationship Options

The Edit Relationship dialog contains a few advanced options that you could find useful on occasion. These include a couple of fundamental options that you have to take into consideration when defining the structure of a data model:

- Cardinality
- Cross filter direction

These are the subject of the next two subsections.

Cardinality

Cardinality defines how many columns in one table relate to the matching column in the linked table. You can see the available choices in the cardinality popup in Figure 10-21.

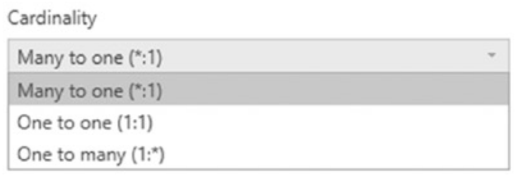


Figure 10-21. *Cardinality in table relationships*

These various options are described in Table 10-6.

Table 10-6. *Power BI Desktop Relationship Types*

Relationship Option	Description
Many to One	Specifies that there are many records in one table for a single record that maps in the table that is joined.
One to One	Specifies that there is a single record in one table for a single record that maps in the table that is joined.
One to Many	Specifies that there is a single record in one table for many records that map in the table that is joined.

Power BI Desktop will nearly always detect the correct cardinality for a relationship. However, if you want to override the choice made by the software because you suspect that another type of cardinality will be required in the future, for instance, then you simply select the cardinality that you want from the popup list.

Cardinality Issues

Data is not always perfect, unfortunately. So there will be times when you will select the cardinality that you want to use to join two tables and will get an error message like the one shown in Figure 10-22.

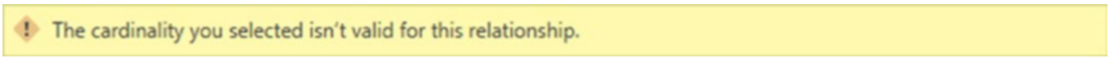

The cardinality you selected isn't valid for this relationship.

Figure 10-22. *Invalid cardinality*

It is impossible to cover all the multiple reasons that can provoke this error. So, if you see this message, here are a few pointers to try and help you sort out this issue, should you encounter it:

- For a one-to-many cardinality, ensure that the values for the field on the "one" side of the relationship exist for *all* the corresponding values on the "many" side.
- For a one-to-one cardinality, check that there are *no duplicate values* for the fields on either side of the table relationship.
- For a one-to-one cardinality, make sure that there are no empty or null values for the fields on either side of the table relationship.
- For a one-to-many cardinality, ensure that the values for the field on the "one" side of the relationship do not contain empty or null values.

Managing Relationships Between Tables

Managing relationships in Power BI Desktop is often the key to creating an efficient data model. It is, however, outside the scope of this book to provide a complete course in data modeling. Nonetheless, here are a few tips to bear in mind when creating your initial data models:

- It can help to think in terms of *main* tables and *lookup* tables. In the data model that you have looked at in this chapter, you can consider certain tables as lookup tables for the main data, such as the Colors, Countries, and Clients tables.
- Lookup tables generally contain a series of values that are not repeated in the table (a list of countries, for instance). These values are called the "one" side of a relationship and are linked to another table where they are referred to on many occasions. Hence the table that contains the multiple references is called the "many" side of the relationship. This is also called the *cardinality* of a relationship by database and data warehouse designers.
- If you take a look at Figure 10-10 (the Power BI Desktop Relationship View) you can see that the different types of relationship are indicated in the join between related tables. A small 1 at the end of a relationship indicates the "one" side of a relationship, and an asterisk indicates the "many" side of the relationship.
- If your source data contains many lookup tables that cascade down through a series of relationships (a classic case is the Category ➤ Subcategory ➤ Product hierarchy that you find in many retail environments), to avoid overcomplicating the data model, you may prefer to merge multiple tables into a single table using Power BI Desktop Query *before* developing the data model in Power BI Desktop itself.

- Sometimes—and this can be the case when importing data from relational databases—you need to join tables on more than one field. This is not possible in Power BI Desktop. However, you can often find workarounds to this, again using Power BI Desktop Query before modeling the data. In cases like this, you can merge tables by creating joins using multiple columns (as you saw in Chapter 8), for instance.
- Data imported from data warehouses can have a built-in structure of facts (main tables containing metrics) and dimensions (containing lookup elements). However, you may want to "flatten" complex hierarchies of dimensions and create single-level tables of lookup elements here, too, using Power BI Desktop Query.
- Sometimes you may want to use the same table twice in different contexts. For instance, a date table may be useful to join to a sale date and a purchase date. In cases like this, you can reimport the date table a second time (and give it another name) and then create two separate joins from a lookup table to the two different lookup tables. This allows you to filter and aggregate data by separate date criteria. A lookup table like this is often called a *role-playing dimension*.
- It is possible to create multiple relations between one table and another, and to specify that only one of them is active. You can then force a calculation to apply a nonactive relationship using the `USERELATIONSHIP()` function in DAX. However, we are already starting to reach more complex levels of data modeling, so I will only mention the possibility here.

Cross Filter Direction

You can see the two cross-filtering options in Figure 10-23. These allow you to specify whether filters will apply to all tables in a joined set of tables or only in the table where an aggregation is being carried out.

Figure 10-23. *Cross filter direction in table relationships*

The two options are described in Table 10-7.

Table 10-7. *Cross Filter Direction Options*

Cross Filter Direction Option	Description
Single	Filters on the linked table will apply to the table where the data aggregations take place, but not the reverse.
Both	For filtering purposes, both tables are considered as if they are a single, larger, table. All data in all tables will be filtered.

Other Relationship Options

The fundamental remaining options for relationships are

- Make this relationship active
- Assume referential integrity

Make this Relationship Active

In some data models, there can be multiple relationships (on different fields) between tables. However, only one of these relationships can be active at any one time. Clicking the "Make this relationship" active check box tells Power BI Desktop that the selected relationship is the active one. Other relationships can, however, be used in DAX code, as you will see in the upcoming chapters.

Assume Referential Integrity

This option is only available when using DirectQuery (which you learned to use in Chapter 4). By definition, this option applies only to a relational database. Enabling this lets Power BI Desktop send more efficient queries to the underlying data source. This means that data is updated faster. For this to work, there are a couple of basic requirements:

- Data in the From column in the relationship can never be Null or blank.
- For each data element in the "from" column, there is a corresponding value in the "to" column.
- The "from" column is on the "many" side in a one-to-many relationship, or the relationship is a one-to-one type of join.

■ **Note** You need to be aware that setting the "Assume referential integrity" option when the underlying data does not fulfill the preceding requirements will not prevent the option from being applied, or even data being returned. However, the results might be erroneous.

Reimporting Related Tables

There is one fairly important point to make to conclude this chapter. This is that if you delete a set of related tables and subsequently reimport them without importing the relationships, then Power BI Desktop will *not* remember the relationships that existed previously. Consequently, you will have to re-create any relationships manually. The same is true if you delete and reimport any table that you linked to an existing table in Power BI Desktop—once a relationship has been removed through the process of deleting a table, you will have to re-create it.

Conclusion

This chapter was all about taking the clean data that you had prepared using Power BI Desktop Query and molding it into a structured and coherent data model that will be the basis for your dashboards and reports.

To begin with, you saw how to look at the whole dataset that was now available in the data model. This included sorting the data and adjusting column widths.

Then you learned how to ensure that each column was defined as having the appropriate data type. After that you applied any number formats that would be required in future dashboard elements directly to the data model. You also saw how to prepare certain types of column for use in maps or to provide hyperlinks.

Most importantly, you then learned how to create table relationships that pull all the disparate data sources together in a joined-up data model that has now become the basis for some in-depth analytics. This will, in most circumstances, be the singular most important aspect of the data model. Simply put, a good, clean, and well-structured data model will allow you to get the most out of your data.

All that you have to do now is add any calculated metrics that your reports need. This is the subject of the next three chapters.

CHAPTER 11

Extending the Data Model with Calculated Columns

This chapter further develops the data model that you created in the previous chapter. It explains how to augment the existing tables by adding new columns containing calculations to the tables that you have imported. You can then apply these additional metrics to the dashboards that you create using Power BI Desktop.

Admittedly, not every data model in Power BI Desktop needs extensive calculations. Frequently, the data can speak for itself without much polishing. Yet business intelligence is, at its heart, based on figures. Consequently, sooner or later, you need to apply simple math, calculate percentages, or compare figures over time. You may even want to develop more complex formulas that enable you to extend your analyses and illustrate your insights. Fortunately, Power BI Desktop makes these—and many, many other calculations—amazingly easy. What's more, if you are an Excel user, you will probably find most of the techniques explained in this chapter to be totally intuitive.

In some cases, you only need to cherry-pick techniques from the range of available options to finalize a dataset. So, it probably helps to know what Power BI Desktop can do and when to use the techniques outlined in this chapter. Therefore, I leave it to you to decide what is fundamental and what is useful. The objective in this chapter and the next two is to present a tried and tested suite of calculation solutions so you are empowered to deal with a range of the potential challenges that you may encounter in your data analyses.

All calculations in the Power BI Desktop data model are written using a simple language named DAX. This stands for **D**ata **A**nalysis e**X**pressions. As you will see, DAX is not in any way a complex programming language. Indeed, it is known as a *formula language* because it is a set of nearly 300 formulas that you can use and combine to extend data models and to create metrics to underpin the visualizations in your dashboards. Fortunately, DAX formulas are loosely based on the formulas in Excel (indeed, a good third of them are identical) so the learning curve for an Excel user is really quite short.

Given the vast horizons that DAX opens up to Power BI Desktop users, a single chapter could never be enough to give you a decent idea of the practical uses of this formula language. So the introduction to DAX in this book is spread over three chapters. To apply some structure to a potentially huge and amorphous area, I have broken down DAX into the following areas:

- This chapter covers *column-based calculations* (where the formula appears as a new column in a table).
- Chapter 12 describes *measures* (calculations that are added to a table but that do not add a column of calculated data).
- Chapter 13 describes *time calculations* (measures that are used to aggregate data over time periods or to compare data over time).

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_11

If you want to continue enhancing the data based on the kinds of data transformations that you saw in the previous few chapters, download the CarSalesDataWithDataModel.pbix file from the Apress web site. This file lets you follow the examples as they appear in this chapter.

Types of Calculations

If you are lucky, then the data that you have imported contains everything that you need to create all the visualizations you can dream up in Power BI Desktop. Reality is frequently more brutal than that, however, and it necessitates adding further metrics to one or more tables. These calculated metrics will extend the data available for visualization. This is fundamental when you are using tools such as Power BI Desktop that do not allow you to add calculated elements to the output, but insist that all metrics—whether they are source data or calculated metrics—exist in the dataset. This is less of a constraint and more of a nod toward good design practice, because it forces you to develop calculations once and to place them in a single central repository. It also reduces the risk of error, because users cannot develop their own (possibly erroneous) metrics and calculations and so distort the truth behind the data.

When creating DAX metrics, you are defining elements that are of practical use for your dashboard visualizations. This can include

- Creating derived metrics that will appear in visualizations.
- Adding elements that you use to filter pages or visualizations.
- Creating elements that you use to segment or classify data. This can include creating your own groupings.
- Defining new metrics based on existing metrics.
- Adding your own specific calculations (such as accounting or financial formulas).
- Adding weightings to values.
- Ranking and ordering data.

And many, many more…

Adding New Columns

Adding new columns is one of the two ways in which you can extend a dataset with derived metrics that you can use in Power BI Desktop dashboards. There are multiple reasons why you may need further columns, including:

- Concatenating data from two existing columns into one new column
- Performing basic calculations for every row in the table, such as adding or subtracting the data in two or more columns
- Extracting date elements such as the month or year from a date column and adding them as a new column
- Extracting part of the data in a column into another column
- Replacing part of the data in a column with data from another column

- Creating the column needed to apply a visually coherent sort order to an existing column
- Showing a value from a column in a linked table inside the source table

Indeed, the list could go on. However, I am sure that you get the idea from what you have read so far.

Before you start wondering exactly what you are getting yourself into, I want to add a few words of reassurance about the ways in which a data model can be extended.

- First, extending a table with added columns is designed to be extremely similar to what you would do in Excel. Consequently, you are in all probability building on your existing knowledge as an Excel power user.
- Second, the functions that you will be using are, wherever possible, similar to existing Excel functions. This does not mean that you have to be an Excel super user to add a column, but that knowledge gained using Excel will help with Power BI Desktop and vice versa.
- Finally, most of the basic table extension techniques follow similar patterns and are not complex. So the more you work at adding columns, the easier it will become as you reuse and extend techniques and formulas.

Creating columns is a bit like creating a formula in Excel that you then copy down over the entire column. They are even closer to the derived columns that you can add to queries in Access. The key thing to note is that any formula will be applied to the *entire* column.

It is worth noting from the start that a formula that you add to a new column is calculated and applied to a column when it is created. It is only recalculated if you recalculate the entire table or file or if you refresh the source data.

■ **Note** In this chapter and the next, I frequently emphasize that you need to prepare your metrics *before* building visualizations for dashboards and reports. In practice, Power BI Desktop is extremely forgiving and immensely supple. It lets you switch from Dashboard View to Data View at any time so that you can add any new columns or missing metrics. Indeed, you can even add measures from the Fields list in Dashboard View. However, it can be more constructive to think through all your data requirements before rushing into the fun part that is creating dashboards. This approach can save you creating duplicate measures with different names and can help you to adopt a clear and coherent approach to naming metrics as well.

Naming Columns

If you create new columns, then you need to give them names. Inevitably, there are a few minor limitations on the names that you can apply. So, rather than have Power BI Desktop cause problems, I prefer to explain the overall guidelines on the Power BI Desktop naming conventions earlier rather than later in the course of this chapter.

The first thing to remember is that column names have to be *unique* inside each table. Therefore, you *cannot* have two columns with the same name inside the *same* table. You *can*, however, have two columns that share a name if they are in separate tables. However, I generally advise that you try to keep column names unique across all the tables in a Power BI Desktop file if you possibly can. This can make building visualizations easier and safer because you do not run the risk of using a column from the "wrong" table in a chart, for instance, and getting entirely inappropriate results as a consequence.

The essential point of note is that columns cannot contain any of the following:

- Spaces (unless the spaces are enclosed by brackets or single apostrophes)
- The characters: `.,;':/\*|?&%$!+=()[]{}<>`

Fortunately, column names are *not* case sensitive.

■ **Note** All the restrictions that concern column names also apply to measures (which you will learn about in the next chapter).

Concatenating Column Contents

As an initial example of a new column, I will presume that, when working with Power BI Desktop to create a dashboard for Brilliant British Cars, you have met a need for a single column of data that contains both the make and model of every car sold. Because the data we imported contains this information as separate columns, we need to add a new column that takes the data from the columns Make and Model, and joins them together (or concatenates them, if you prefer) in a new column. The following explains how to do this:

1. Open the file C:\PowerBiDesktopSamples\CH11\CarSalesDataWithDataModel.pbix.
2. In the Power BI Desktop window, make sure that you are in Data View.
3. Click the Stock table to select it.
4. In the Modeling ribbon, click the New Column button. A new empty column will appear to the right of the final column of data. This column is currently entitled Column and is highlighted.
5. The formula bar above the table of data will display `Column =`.
6. In the formula bar, click to the right of the equals sign. Press **[**. A list of the fields in the current table will appear, as shown in Figure 11-1.

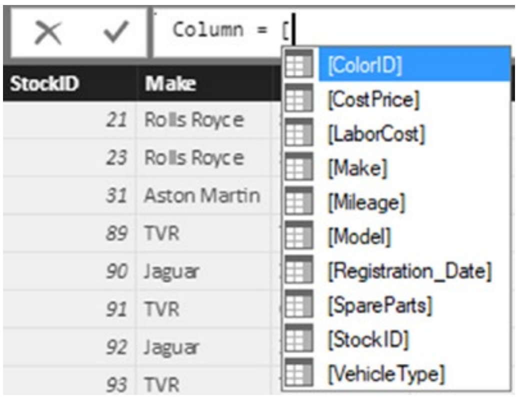


Figure 11-1. *Selecting a field for a formula*

7. Double-click the [Make] field in the popup list of columns (or use the cursor keys to highlight the [Make] field and press the Tab key). The formula bar now reads `=[Make]`.

8. In the formula bar, add **& " " &**. The formula bar now reads `=[Make] & " " &`.

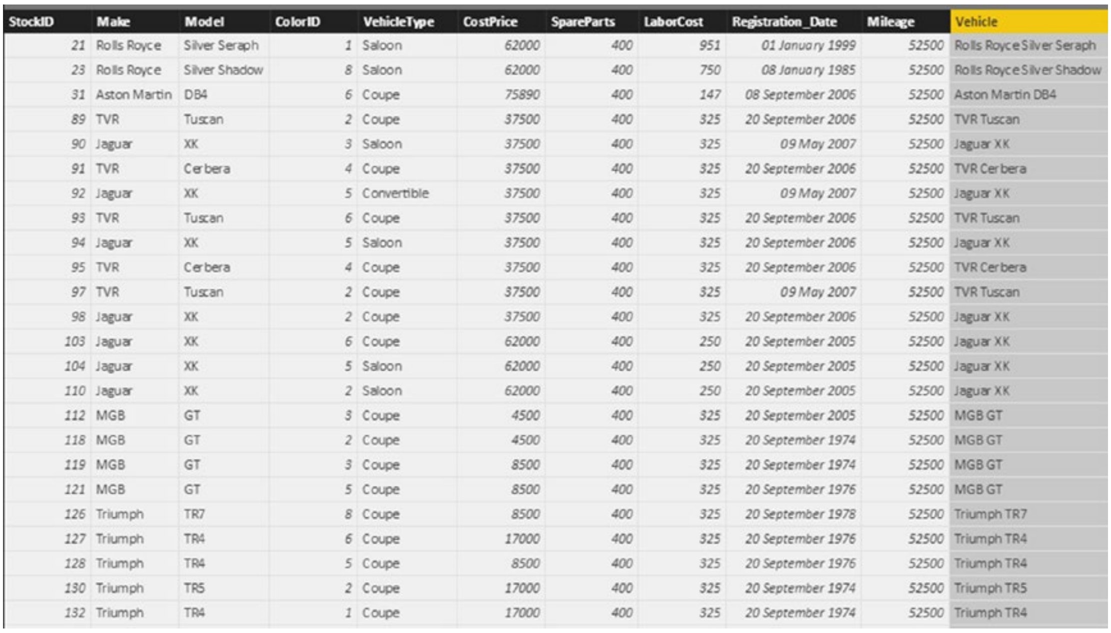
9. Press **[**. Select [Model] from the list of fields. The formula bar now reads `=[Make] & " " & [Model]`.

10. Press Enter (or click the tick icon in the formula bar). The column is automatically filled with the result of the formula and it shows the make and model of each car sold.

11. Right-click the column header for the new column and select Rename.

12. Type the word **Vehicle** and press Enter.

The table will now look something like Figure 11-2. Moreover, the new column has been added as a field to the Stock table in the Fields list.



StockID	Make	Model	ColorID	VehicleType	CostPrice	SpareParts	LaborCost	Registration_Date	Mileage	Vehicle
21	Rolls Royce	Silver Seraph	1	Saloon	62000	400	951	01 January 1999	52500	Rolls Royce Silver Seraph
23	Rolls Royce	Silver Shadow	8	Saloon	62000	400	750	08 January 1985	52500	Rolls Royce Silver Shadow
31	Aston Martin	DB4	6	Coupe	75890	400	147	08 September 2006	52500	Aston Martin DB4
89	TVR	Tuscan	2	Coupe	37500	400	325	20 September 2006	52500	TVR Tuscan
90	Jaguar	XK	3	Saloon	37500	400	325	09 May 2007	52500	Jaguar XK
91	TVR	Cerbera	4	Coupe	37500	400	325	20 September 2006	52500	TVR Cerbera
92	Jaguar	XK	5	Convertible	37500	400	325	09 May 2007	52500	Jaguar XK
93	TVR	Tuscan	6	Coupe	37500	400	325	20 September 2006	52500	TVR Tuscan
94	Jaguar	XK	5	Saloon	37500	400	325	20 September 2006	52500	Jaguar XK
95	TVR	Cerbera	4	Coupe	37500	400	325	20 September 2006	52500	TVR Cerbera
97	TVR	Tuscan	2	Coupe	37500	400	325	09 May 2007	52500	TVR Tuscan
98	Jaguar	XK	2	Coupe	37500	400	325	20 September 2006	52500	Jaguar XK
103	Jaguar	XK	6	Coupe	62000	400	250	20 September 2005	52500	Jaguar XK
104	Jaguar	XK	5	Saloon	62000	400	250	20 September 2005	52500	Jaguar XK
110	Jaguar	XK	2	Saloon	62000	400	250	20 September 2005	52500	Jaguar XK
112	MGB	GT	3	Coupe	4500	400	325	20 September 2005	52500	MGB GT
118	MGB	GT	2	Coupe	4500	400	325	20 September 1974	52500	MGB GT
119	MGB	GT	3	Coupe	8500	400	325	20 September 1974	52500	MGB GT
121	MGB	GT	5	Coupe	8500	400	325	20 September 1976	52500	MGB GT
126	Triumph	TR7	8	Coupe	8500	400	325	20 September 1978	52500	Triumph TR7
127	Triumph	TR4	6	Coupe	17000	400	325	20 September 1976	52500	Triumph TR4
128	Triumph	TR4	5	Coupe	8500	400	325	20 September 1976	52500	Triumph TR4
130	Triumph	TR5	2	Coupe	17000	400	325	20 September 1974	52500	Triumph TR5
132	Triumph	TR4	1	Coupe	17000	400	325	20 September 1974	52500	Triumph TR4

Figure 11-2. *An initial calculated column*

I imagine that if you have been using Excel for any length of time, then you might have a strong sense of déjà vu after seeing this. After all, what you just did is virtually what you would have done in Excel. All you have to remember is that

- Any additional columns are added to the *right* of the existing columns. You *cannot* move them elsewhere in the table once they have been created.
- All functions begin with the equals sign.
- Any function can be developed and edited in the formula bar at the top of the table.

- Reference is always made to *columns,* not to cells (as you would in Excel).
- Column names are always enclosed in square brackets.
- You can nest calculations in parentheses to force inner calculations before outer calculations—again, just as you would in Excel.

Once a new column has been created, it remains at the right of any imported columns in the table where you added it. It is not possible to move the new column elsewhere in the table. The field that it represents is always added to the bottom of the collection of fields for this table in the Fields list. This way the available fields will appear in the order that they were created. However, fields always appear in alphabetical order in the Fields list in Dashboard View.

If you look closely at the field that was added, you will notice that there is a tiny Fx icon to its left. This is how you can distinguish new columns from other fields such as numeric fields (which have a sigma (∑) icon to their left) or measures (which you will meet in the next chapter) that have a small calculator icon to their left.

■ **Note** In this example, you selected columns from the popup list of the available columns in the table. You can enter the column name in the formula bar if you prefer, but if you do then you *must* enclose the column name in square brackets. You must also enter it *exactly* as it appears in the Fields list and column title.

Tweaking Text

In Chapter 7, you learned many techniques that you can apply to text-based columns. If you remember, these included changing the capitalization and removing extra spaces (among other things).

DAX also lets you clean up and modify the text in the tables that you have imported into your data model. Indeed, it offers a wide range of functions that you can apply to standardize and cleanse text in tables. As an example, let's imagine that you want to create a column in the Clients table that contains a shortened version of each town. In fact, what you want to do is extract a three-letter acronym from the first letters of the town name, which you can use later in charts.

1. In the Power BI Desktop window, make sure that you are in Data View.
2. Click the Stock table in the Fields list.
3. In the Modeling ribbon, click the New Column button. A new empty column will appear to the right of the final column of data. This column is currently entitled Column and is highlighted.
4. The formula bar above the table of data will display `Column =`.
5. In the formula bar, click to the right of the equals sign.
6. Type **LEFT(**. Once the function appears in the popup menu, you can select it if you prefer.
7. Click inside the Town column of the Clients table. `Clients[Town]` will appear in the formula bar. Alternatively, you can type a left square bracket and select the [Town] field.
8. Enter a comma.

9. Enter the number **3**. This indicates to the LEFT() function that it is the *three* characters on the left that you want to isolate for each row in this column.

10. Add a right parenthesis, **)**.

11. Still inside the formula bar, replace Column with **TownAbbreviation**. The formula bar will read

```
TownAbbreviation = LEFT(Clients[Town],3)
```

12. Press Enter (or click the tick icon in the formula bar). The column is automatically filled with the result of the formula and it shows the first three letters of every town's name.

As you can see, the LEFT() function takes two parameters:

- First, the field from which you want to extract the leftmost characters
- Second, the number of characters to extract

And that is all that you have to do. By applying a simple text formula, you have prepared a column of text for effective use in a visualization.

DAX contains a couple of dozen functions that you can apply to the text in columns. Most of them follow the same principle as the LEFT() function in that they take at least two parameters, the first of which is the column that you want to take as the basis for your new column and the second (or even third) parameters provide information about how the modification is to be applied. Since I do not have space to explain each and every one of these functions, Table 11-1 contains a succinct overview of a selection of some of the most useful text functions. This table does not explain all the subtleties of every function, but is designed to be both a brief introduction and a starting point for your DAX formulas that rework the text elements of your data tables.

Table 11-1. *Core Power BI Desktop Text Functions*

Function	Description	Example
LEFT()	Extracts a specified number of characters from the left of a column.	LEFT(Clients[Town], 3)
RIGHT()	Extracts a specified number of characters from the right of a column.	RIGHT(Invoices[InvoiceNumber], 12)
MID()	Extracts a specified number of characters (the second parameter) from a specified position defined by the number of characters from the left (the first parameter) inside a column.	RIGHT(Invoices[InvoiceNumber], 10, 4)
UPPER()	Converts the data to uppercase. This function takes no parameters.	UPPER(Clients[ClientName])
LOWER()	Converts the data to lowercase. This function takes no parameters.	LOWER(Clients[ClientName])

(*continued*)

Table 11-1. (*continued*)

Function	Description	Example
`TRIM()`	Removes any extra spaces (trailing or leading) from the text inside a column. This function takes no parameters.	`TRIM(Clients[Address1])`
`LEN()`	Counts the number of characters in a column. This is often used with the `MID()` function. This function takes no parameters.	`LEN(Clients[Address1])`
`FIND()`	Gives the starting point (as a number of characters) of a string inside a column. This function is case sensitive. Interestingly, the first parameter is the text to find and the second is the column.	`FIND('Car',Clients[ClientName])`
`SEARCH()`	Gives the starting point (as a number of characters) of a string inside a column. This function is not case sensitive and disregards accents. Interestingly, the first parameter is the text to find and the second is the column.	`SEARCH('Car',Clients[ClientName])`
`SUBSTITUTE()`	Replaces one text with another inside the column. This is a bit like the search and replace function in a word processor.	`SUBSTITUTE(Clients[ClientName], 'Car', 'Vehicle')`
`VALUE()`	Converts a figure in a text column to a numeric data type. This function takes no parameters.	`VALUE(Colors[ColorID])`
`FIXED()`	Takes a number and rounds it to a specified number of decimals then converts it to a text. The second parameter indicates the number of decimals to apply.	`FIXED(Stock[LaborCost], 2)`

There are a couple of points to note now that you have seen how to use DAX formulas in Power BI Desktop:

- Functions need not apply to entire columns; they can be applied to specific texts as part of a more complex formula.
- You can enter functions in uppercase or lowercase.

Simple Calculations

To extend the basic principle and to show a couple of variations on a theme, let's now add a calculation to the Stock table. More precisely, I assume that our Power BI Desktop visualizations frequently need to display the figure for the direct costs relating to all vehicles purchased, which I define as being the purchase price plus any related costs. You obtain this by applying a variation on a technique that you have used before:

1. In Data View, select the Stock table.
2. Click New Column in the Modeling ribbon.

3. In the formula bar, replace the word Column with the word **Direct Costs**. (Notice that there is a space, because column names can contain spaces.)
4. To the right of the equals sign, enter a left square bracket: **[**. The list of the fields available in the Stock table will appear in the formula bar.
5. Type the first few characters of the column that you want to reference—**CostPrice** in this example. The more characters you type, the fewer columns are displayed in the list.
6. Click the column name. It will appear in the formula bar (including the right bracket).
7. Enter the minus sign.
8. Enter a left parenthesis.
9. Enter a left square bracket and select the SpareParts column.
10. Enter a plus sign.
11. Enter a left square bracket and select the LaborCost column.
12. Enter a right parenthesis. This corresponds to the left parenthesis before the SpareParts field. The formula should read

```
Direct Costs = [CostPrice] - ([SpareParts] + [LaborCost])
```

13. Click the tick box in the formula bar (or press Enter). The new column will be created. It will also appear as a new field in the Fields list for this table.

As you can see, using arithmetic in calculated columns in Power BI Desktop is almost the same as using calculating cells in Excel. If anything, it is easier because you do not have to copy the formula over hundreds or even thousands of rows as the formula is automatically applied to every row in the table.

When selecting fields in steps 5, 9, and 11, you can (if you prefer) click the field name in the Fields list rather than enter a left bracket and scroll through a list of fields. Of course, this assumes that you have not hidden the Fields list and that you have expanded the table name so that you can see the fields that it contains.

If you are a Power Pivot user, then the sense of déjà vu is probably so total as to be overwhelming. In fact, most Power Pivot users will probably only need to skim through this chapter as the Data View of Power BI Desktop is virtually identical to the Power Pivot window in Excel. Except for the tables that now appear on the right (instead of tabs at the bottom of the window) and the absence of a formula button, there are few differences. So feel free to jump over any sections (in this chapter and the next) that you already know by heart if you are a Power Pivot expert.

■ **Note** You can give a new column an appropriate name either when you create the formula initially (by replacing the word Column with the new column name) or by renaming the column once the formula is correct and confirmed. You can include spaces in column names if you want. After all, this is how the name will appear in your dashboards.

Math Operators

For the sake of completeness, and in case there are any newcomers to the world of Microsoft products out there, I prefer to recapitulate the core math functions that are available in Power BI Desktop. These are given in Table 11-2.

Table 11-2. *Core Power BI Desktop Math Operators*

Operator	Description	Example
+	Adds two elements.	`[SpareParts] + [LaborCost]`
-	Subtracts one element from another.	`[CostPrice] - [SpareParts ]`
/	Divides one element by another.	`[CostPrice] / [SpareParts ]`
*	Multiplies one element by another.	`[CostPrice] * 1.5`
^	Raises one element to the power of another.	`[CostPrice] ^ 2`

If you are working in BI, then you are certainly able to perform basic math operations. Consequently, I will not explain things you most likely already know. Just use the same arithmetical operators as you would use in Excel and, after a little practice, you should be able to produce calculated columns with ease. Remember that you have to enclose in parentheses any part of a formula that you want to have calculated before the remainder of the formula. This way, you will avoid any unexpected results in your dashboards.

Rounding Values

You already saw in Chapter 6 that Power BI Desktop Query can round and truncate values when you are preparing data ready for loading into a data model. In practice, of course, you might not yet be aware that you need to tweak your data at this stage. Fortunately, DAX also contains a range of functions that can be used to round values up and down, or even to the nearest hundred, thousand, or million, if need be.

As an example of this, take the column Direct Costs that you created previously and round it to the nearest integer. This way, you also learn how to modify a formula in DAX.

1. In Data View, select the Stock table.
2. Click inside the Direct Costs column. The column will be selected and the formula that you created previously will appear in the formula bar.
3. Click inside the formula bar to the right of the equals sign.
4. Enter **Round(**. You can also select the formula from the popup if you prefer.
5. Click at the right of the formula in the formula bar.
6. Enter a comma.
7. Enter a **0**.
8. Add a right parenthesis to complete the `ROUND()` function. You will see that the corresponding left parenthesis is highlighted in the formula bar to help you track which pair of parentheses is which.

9. Click the tick box in the formula bar (or press Enter). The formula will be modified and any decimals removed from the data in the column. The formula will read

```
= ROUND([Direct Costs], 0)
```

This example introduced the ROUND() function. It will round a value (whether calculated or loaded from a data source) to the number of decimals specified as the second parameter of the function, which is zero in this example.

ROUND() is only one of the functions that you can choose when truncating or rounding values. The DAX functions that carry out rounding and truncation are given in Table 11-3.

Table 11-3. *DAX Rounding and Truncation Functions*

Function	Description	Example
ROUND()	Rounds the value to 0 if the second parameter is zero. If the second parameter is greater than zero, the function rounds the value to the number of decimals indicated by the second parameter. If the second parameter is less than zero, the figure to the left of the decimal is rounded to the nearest 10 (for a second parameter of –1, 100 (for a second parameter of –2, and so forth).	ROUND([CostPrice], 2)
ROUNDDOWN()	Rounds the value down to 0 if the second parameter is zero. If the second parameter is greater than zero, the function rounds the value down to the number of decimals indicated by the second parameter. If the second parameter is less than zero, the figure to the left of the decimal is rounded down to the nearest 10 (for a second parameter of –1, 100 (for a second parameter of –2, and so forth). The value is always rounded down; never up.	ROUNDDOWN([CostPrice], 2)
ROUNDUP()	Rounds the value up to 0 if the second parameter is zero. If the second parameter is greater than zero, the function rounds the value up to the number of decimals indicated by the second parameter. If the second parameter is less than zero, the figure to the left of the decimal is rounded up to the nearest 10 (for a second parameter of –1, 100 (for a second parameter of –2, and so forth). The value is always rounded up; never down.	ROUNDUP([CostPrice], 2)
MROUND()	Rounds the value to the nearest multiple of the second parameter.	MROUND(([CostPrice], 2)

(*continued*)

Table 11-3. (*continued*)

Function	Description	Example
`TRUNC()`	Removes the decimals from a value.	`TRUNC([CostPrice])`
`INT()`	Rounds down (or up, if the number is negative) to the nearest integer.	`INT([CostPrice])`
`FLOOR()`	Rounds down to the nearest multiple of the second parameter.	`FLOOR([CostPrice], .2)`
`CEILING()`	Rounds up to the nearest multiple of the second parameter.	`CEILING([CostPrice], .2)`
`FIXED()`	Rounds a value to the number of decimals indicated by the second parameter and converts the result to text.	`FIXED([CostPrice], 2)`
`EVEN()`	Rounds a value up to the next even number.	`EVEN([CostPrice])`
`ODD()`	Rounds a value up to the next odd number.	`ODD([CostPrice])`
`CURRENCY()`	Converts a value to the currency data type (with four decimals).	`CURRENCY([CostPrice])`

■ **Note** Remember that using the `ROUND()` function *modifies* the data, whereas formatting numbers only changes their appearance.

Calculating Across Tables

If your data model is not complex (particularly if it consists of a single table), then most calculations should be simple. All you have to do is follow the principle of building math expressions using column names and arithmetic operators.

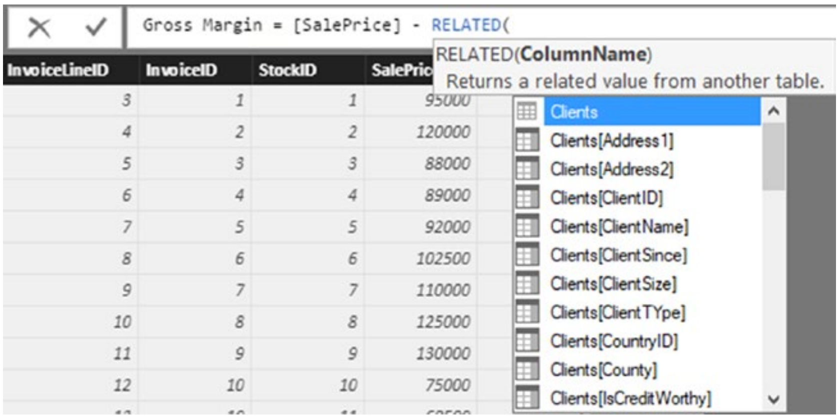
The real world of data analysis is rarely this uncomplicated. In most cases, you have metrics on one table that you need to apply in a calculation in a completely different table. Power BI Desktop makes these “cross-table” calculations really easy, *if* you have defined a coherent data model. (This can be done even if tables are not joined in a data model, as you will discover in the next chapter.)

As an example, let’s look at how to subtract the Stock table’s Direct Costs column from the InvoiceLines table’s SalePrice column to calculate the margin on sales. To do this, we will add a new column, called Gross Margin, to the InvoiceLines table.

1. In Data View, select the InvoiceLines table.
2. Click New Column in the Modeling ribbon.
3. In the formula bar, replace the word Column with the words **Gross Margin**.
4. To the right of the equals sign, enter a left square bracket: **[**. The list of the fields available in the InvoiceLines table will appear in the formula bar.
5. Select the SalePrice field. (Remember that you can type the first few characters to limit the popup list of fields to those most closely resembling the field that you are looking for).

6. Enter a minus sign (you can add spaces before and/or after it, if you want).

7. Start typing the keyword **RELATED**, and select this function once you have limited the selection of functions in the popup list. (Alternatively, you can type the whole word and a left parenthesis.) The popup list will list all the fields of all the tables that can be joined to the current table in the data model. The popup list will look like Figure 11-3.



```
Gross Margin = [SalePrice] - RELATED(
```

RELATED(**ColumnName**)
Returns a related value from another table.

InvoiceLineID	InvoiceID	StockID	SalePric
3	1	1	95000
4	2	2	120000
5	3	3	88000
6	4	4	89000
7	5	5	92000
8	6	6	102500
9	7	7	110000
10	8	8	125000
11	9	9	130000
12	10	10	75000

Figure 11-3. The popup list of related tables and fields

8. Scroll down through the list and select the field Direct Costs in the Stock table. You can jump directly to the Stock table by typing the first few characters of the table name.

9. Enter a right parenthesis. The formula bar should contain the following formula:

```
Gross Margin = [SalePrice]-RELATED(Stock[DirectCosts)
```

10. Click the check icon in the formula bar or press Enter to complete the definition of the calculated column.

You can now see a new column added to the right of the InvoiceLines table. This column contains the gross margin for every vehicle sold, even if the sale price is on one table and the sum of the costs is in a separate table. This is all thanks to the `RELATED()` function, which links fields from different tables using the joins that you defined in the data model.

If you are an Excel user who has spent hours, or even days, wrestling with the Excel `LOOKUP()` function, then you are probably feeling an immense sense of relief. For it really is this easy to look up values in another table in Power BI Desktop. Once again (and at risk of laboring the point), if you have a coherent data model, then you are building the foundations for simple and efficient data analyses further down the line using DAX.

Choosing the Correct Table for Linked Calculations

The nature and structure of a Power BI Desktop data model controls where you can add new columns from another table. In essence, you can only bring data into a table if it is from another table that

- Contains reference data
- Contains many records that are "sub" elements of the current table

So tables such as Countries, Clients, and Colors cannot pull back data from another table using the `RELATED()` function. This is because they are lookup tables and contain reference data that appears only once, but is used many times in other tables. In database terms, these tables are the "one" side of a relationship; whereas tables such as InvoiceLines are on the "many" side of a relationship. In Power BI Desktop (as is the case in a relational database), you can only look up data from the "many" side.

Equally, you can add data to the InvoiceLines table from the Invoices table, because the latter is considered a "parent" to the former. However, you cannot pull data into the Invoices table from the InvoiceLines table. Quite simply, when an invoice contains many lines, Power BI Desktop does not know which row to select and return to the destination table.

In a data model like the sample Brilliant British Cars example, some tables can return data from several linked tables even if this means traversing multiple links. For instance, the Stock table can reach into the InvoiceLines table (because there is a single record in the Stock table for each record in the InvoiceLines table). Since the InvoiceLines table is a child of the Invoices table, the Stock table can also reach "through" the InvoiceLines table into the Invoices table. Indeed, as the Colors table is a lookup table for the Stock table, and the Invoices table looks up data from the Clients table, these are also accessible to the Stock table.

So essentially, the table where you add a new column has the potential to reach through most, if not all, of the data model and return data from many other tables—providing that the data model has been constructed in a coherent manner, of course.

Cascading Column Calculations

New columns can refer to previously created new columns. This apparently anodyne phrase hides one of the most powerful features of Power BI Desktop: the ability to create spreadsheet-like links between columns where a change in one column ripples through the whole data model.

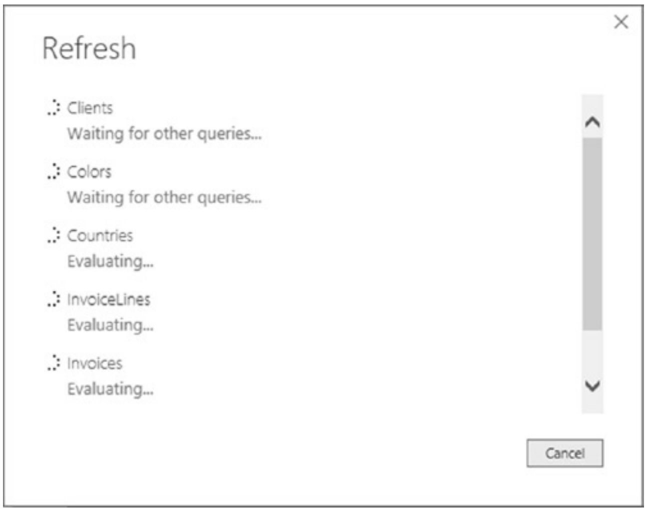
This implies that you help yourself if you build the columns in a logical sequence, so that you always proceed step by step and do not find yourself trying to create a calculation that requires a column that you have not created yet. Another really helpful aspect of new columns is that if you rename a column, Power BI Desktop automatically updates all formulas that used the previous column name, and uses the new name in any visualizations that you have already created. This makes Power BI Desktop a truly pliant and forgiving tool to work with.

So if we take the DAX formulas that you have created so far, you have the Gross Margin column that depends on the data for the sale price and the calculation of the Direct Costs column, which itself is based on the data for the cost price, spare parts, and labor cost. As a spreadsheet user, you probably won't be surprised to see that any change to the source data for the four elements causes both the Direct Costs and Gross Margin columns to be recalculated.

Refreshing Data

Do the following to force Power BI Desktop to recalculate the data model:

1. Activate the Home ribbon in Data View (or in Dashboard View).
2. Click the Refresh button. The Refresh dialog will appear, looking something like Figure 11-4.



Refresh

Clients
Waiting for other queries...

Colors
Waiting for other queries...

Countries
Evaluating...

InvoiceLines
Evaluating...

Invoices
Evaluating...

Figure 11-4. *The Refresh dialog*

After a short while (depending on the amount of data that has to be reloaded from the source(s) into the data model), the dialog closes and the data reappears with all calculated columns updated.

Using Functions in New Columns

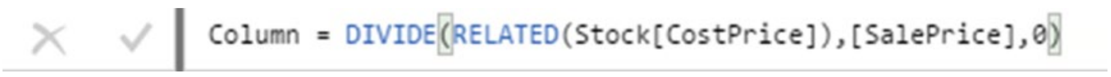
You have seen just how easy it is to extend a data model with some essential metrics in Power BI Desktop. Yet we have only performed simple arithmetic to achieve our ends. Power BI Desktop can do much more than just carry out simple sums, of course.

Safe Division

I imagine that if you are an Excel user, you have seen your fair share of DIV/0 (divide by zero) errors in spreadsheets. Fortunately, the Power BI Desktop team shares your antipathy to this particular issue and they have endowed Power BI Desktop with a particularly elegant solution to the problem. This solution also serves as a simple introduction to the world of DAX functions in Power BI Desktop.

Suppose that you want to add a new column that divides the contents of one column by the contents of another. Still using the CarSalesDataWithDataModel.pbix sample file, you can implement safe division like this:

1. With the InvoiceLines table selected, activate the Modeling ribbon and click the New Column button.
2. To the right of the equals sign, type **DIVIDE** followed by a left parenthesis.
3. Type the formula **RELATED** and a left parenthesis.
4. Select the Stock[CostPrice] field from the list. This will be the numerator (the value that will be divided by another value).
5. Add a right parenthesis (this is to finish the `RELATED()` function).
6. Enter a comma.
7. Enter a left square bracket: **[**.
8. Select the SalePrice field from the list. This will be the denominator (the value that divides the first value).
9. Enter a comma.
10. Enter a **0**. This is the figure that will appear if there is a division by zero error.
11. Add a right parenthesis to end the `DIVIDE()` function. The formula bar will look like Figure 11-5.



```
Column = DIVIDE(RELATED(Stock[CostPrice]),[SalePrice],0)
```

Figure 11-5. *The DIVIDE() function*

12. To the left of the equals sign, replace Column with **SalePriceToSalesCostsRatio**.
13. Click the tick icon in the formula bar (or press the Enter key). The ratio of sales cost to the sale price will be calculated for every row in the table. The column will fill with zeros and will remain highlighted.
14. In the Modeling ribbon, click the percent button, and then add a couple of decimals by using the decimals button. As this metric is a ratio, it is best presented as a percentage to be more easily comprehensible, not only in the current table but also in any visualizations that it appears in.

In this example, you have used a function that required multiple parameters.

- The *numerator*: The number that is divided by another number.
- The *denominator*: The number that is used to divide the first value.
- The *error value*: The number that is used if a divide by zero error is encountered.

I imagine that by now you are feeling that DAX formulas are not only relatively easy, but also made easier by their close relationship to Excel formulas. So let's move on to see a few more.

■ **Note** I realize that explaining each step in a formula might seem like overkill, especially to Excel or Power Pivot gurus. Nonetheless, I prefer to take this approach as it allows me to draw your attention to the various reasons for each part of an operation. This way I can also explain the different shortcut operations that are available as you build up a formula. If you prefer to type in a formula directly, then feel free to jump to the step that contains the formula ready for use toward the end of the sequence of steps that builds the formula.

Counting Reference Elements

Data models are often assembled to make the best use of reference elements. The Brilliant British Cars data model has a couple of lookup tables (Clients and Colors) that contain essential information that you could need to analyze the underlying data. So as an example of how the data model can be put to good use, let's look at how DAX can calculate the number of clients per country.

This challenge introduces two new DAX elements:

- The `COUNTROWS()` function
- The `RELATEDTABLE()` function

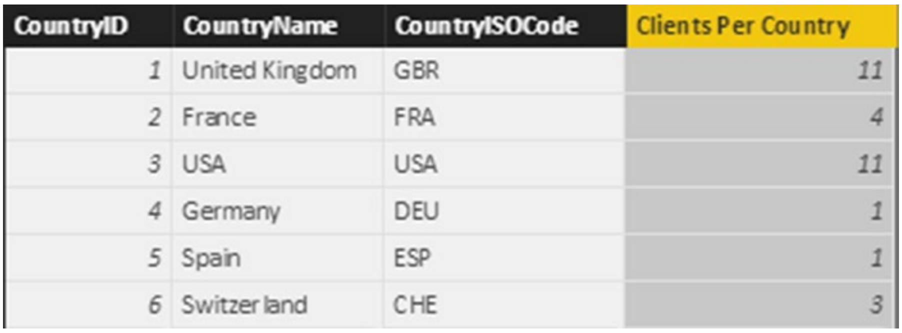
As its name implies, the `COUNTROWS()` function counts a number of rows in a table. While you can use it simply to return the number of records in the current table, it is particularly useful when used with the `RELATEDTABLE()` function. Once again, because the data model has been set up coherently, and the Countries table is joined to the Clients table, using the `COUNTROWS()` and `RELATEDTABLE()` functions together does not just return the number of records in a table, but also calculates the number of records for each element in the table where it is applied. This means that any elements from the Countries table that exist in the Clients table can be identified because the Clients table is using the Countries table as a lookup table.

Here is an example of how you can use these two functions to count reference elements:

1. In Data View, select the Countries table.
2. Click New Column in the Modeling ribbon.
3. In the formula bar, replace the word Column with the words **Clients Per Country**.
4. To the right of the equals sign, type **COUNTROWS(**. Once the keyword appears in the popup list, you can select it to save time (and keystrokes), if you want.
5. Enter (and/or select) **RELATEDTABLE(**.
6. The list of the tables that are related to the Countries table will appear in the formula bar.
7. Select Clients.
8. Add *two* right parentheses. One will close the `RELATEDTABLE()` function and the other will end the `COUNTROWS()` function.
9. The formula bar will contain the following:

```
Clients Per Country = COUNTROWS(RELATEDTABLE(Clients))
```

10. Click the tick icon in the formula bar (or press the Enter key). The number of customers for each country will appear in the new column named Clients Per Country. The Countries table now looks like what's shown in Figure 11-6.



CountryID	CountryName	CountryISOCode	Clients Per Country
1	United Kingdom	GBR	11
2	France	FRA	4
3	USA	USA	11
4	Germany	DEU	1
5	Spain	ESP	1
6	Switzerland	CHE	3

***Figure 11-6.** Using the COUNTROWS() function to calculate the number of clients per country*

Statistical Functions

As an intrinsic part of the Microsoft BI offering, DAX can calculate aggregates. After all, analyzing totals, averages, minima, and maxima (among others) is a core aspect of much business intelligence.

However, I do not want just to show you how to create a column containing the average sale price of all vehicles in the dataset. This would hardly be instructive. So instead of this, let's begin with a slightly more interesting requirement. Suppose that, for each vehicle, you want to see how the net profit compares to the average net profit.

1. Select the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Column button.
3. In the formula bar to the left of the equals sign, replace Column with **DeltaToAvgNetProfit**.
4. Click to the right of the equals sign.
5. Enter a left square bracket.
6. Select the Gross Margin field.
7. Enter a minus sign.
8. Start typing the word **Average**. Power BI Desktop will display the list of available fields. When enough of the function name Average appears in the list of functions, select the `AVERAGE()` function.
9. Select the field Gross Margin.
10. Add a right parenthesis. The formula will look like this:

    ```
    DeltaToAvgNetProfit = [Gross Margin]-AVERAGE([Gross Margin])
    ```

11. Confirm the formula by pressing Enter or clicking the tick icon in the formula bar. The new column will display the difference between the net margin for each row and the average net margin.

Once again, if you are an Excel or Microsoft Access user, you are probably feeling quite at ease with this way of working. Even if you are not a spreadsheet or database expert, you must surely be feeling reassured that creating calculations that apply instantly to an entire column is truly easy.

Now that you have seen the basic principles, look at some of the more common available aggregation functions described in Table 11-4.

Table 11-4. *DAX Statistical Functions*

Function	Description	Example
`AVERAGE()`	Calculates the average (the arithmetic mean) of the values in a column. Any non-numeric values are ignored.	`AVERAGE([Mileage])`
`AVERAGEA()`	Calculates the average (the arithmetic mean) of the values in a column. Empty text, non-numeric values, and FALSE values count as 0. TRUE values count as 1.	`AVERAGEA([Mileage])`
`COUNT()`	Counts the number of cells in a column that contain numeric values.	`COUNT([Mileage])`
`COUNTA()`	Counts the number of cells in a column that contain any values.	`COUNTA([Mileage])`
`COUNTBLANK()`	Counts the number of blank cells in a column.	`COUNTBLANK([Mileage])`
`COUNTROWS()`	Counts the number of rows in a table.	`COUNTROWS(Stock)`
`DISTINCTCOUNT()`	Counts the number of unique values in a table.	`DISTINCTCOUNT([Vehicle])`
`MAX()`	Returns the largest numeric value in a column.	`MAX([Mileage])`
`MAXA()`	Returns the largest value in a column. Dates and logical values are also included.	`MAXA([Mileage])`
`MEDIAN()`	Returns the median numeric value in a column.	`MEDIAN([Mileage])`
`MIN()`	Returns the smallest numeric value in a column.	`MIN([Mileage])`
`MINA()`	Returns the smallest numeric value in a column. Dates and logical values are also included.	`MINA([Mileage])`

There are many more statistical functions in DAX, and you can take a deeper look at them in the Power BI online documentation. However, for the moment the intention is not to blind you with science, but to introduce you more gently to the amazing power of DAX. For the moment, then, rest reassured that all your favorite Excel functions are present when it comes to calculating aggregate values in Power BI Desktop.

Applying a Specific Format to a Calculation

Sometimes you will want to display a number in a particular way. You may need to do this to fit more information along the axis of a chart, for instance. In cases like these, you can, in effect, duplicate a column and reformat the data so that you can use it for specific visualizations. As an example of this, and to show how functions can be added to formulas that contain math, you will convert the cost of vehicles and any spare parts from pounds sterling to US dollars and then format the result in dollars.

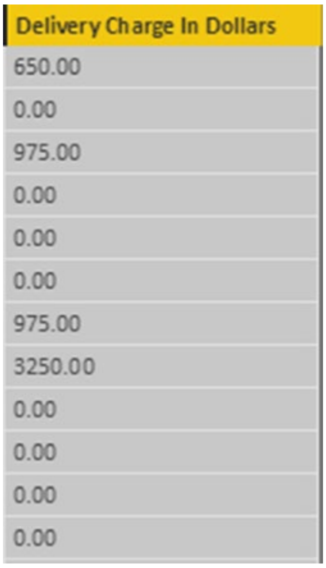
1. In the Power BI Desktop window, make sure that you are in Data View.
2. Click the Invoices table.
3. In the Modeling ribbon, click the New Column button. A new empty column will appear to the right of the final column of data. This column is currently highlighted and titled Column.

4. The formula bar above the table of data will display `Column =`.
5. In the formula bar, click to the right of the equals sign.
6. Type **FORMAT(**. Once the function appears in the popup menu, you can select it if you prefer.
7. Click inside the DeliveryCharge column. Invoices[DeliveryCharge] will appear in the formula bar. Alternatively, you can type a left square bracket and select the [DeliveryCharge] field.
8. Enter * **1.6**.
9. Enter a comma.
10. Enter the text **"Fixed"** (include the double quotes).
11. Add a right parenthesis.
12. Still inside the formula bar, replace Column with **Delivery Charge In Dollars**.
13. To the right of the equals sign, enter **"$ "** & (include the double quotes and the space after the dollar sign).

    The formula bar will read as follows:

    ```
    Delivery Charge In Dollars = "$ " & FORMAT([DeliveryCharge] * 1.6, "Fixed")
    ```

14. Press Enter (or click the tick icon in the formula bar). The column will contain the number that is in the DeliveryCharge column multiplied by 30% (this is a very approximate exchange rate). However, it is formatted as a text and preceded by a dollar sign. Figure 11-7 shows a few records from this new column.



Delivery Charge In Dollars
650.00
0.00
975.00
0.00
0.00
0.00
975.00
3250.00
0.00
0.00
0.00
0.00

Figure 11-7. *Applying a custom numeric format*

The `FORMAT()` function can be applied equally well to dates and times as to numeric columns, as you will see later in this chapter. Indeed, it offers a wealth of possibilities. So many in fact, that rather than illustrate all of them, Tables 11-5 and 11-6 contain the essential predefined formats that you can apply to columns of numbers.

Table 11-5. *Predefined Currency Formats*

Format Code	Description	Example	Comments
Currency	Currency	`FORMAT(Stock[CostPrice], "Currency")`	The currency indicator will depend on the PC's settings and language used.
Scientific	Exponential or scientific notation	`FORMAT(Stock[CostPrice], "Scientific")`	This is also called the *scientific format.*
Fixed	Fixed number of decimals	`FORMAT(Stock[CostPrice], "Fixed")`	Displays at least one figure to the left of the decimal (even if it is a zero) and two decimals.
General Number	No format	`FORMAT(Stock[CostPrice], "General Number")`	Displays the number with no thousand separators.
Percent	Percentage (and divided by 100)	`FORMAT(Stock[SalePriceToSalesCostsRatio], "Percent")`	Displays the number as a percentage.

Table 11-6. *Custom Number Formats*

Format Code	Description	Comments
`0`	The zero placeholder.	Adds a zero even if no number is present.
`#`	The digit placeholder.	Represents a number if one is present.
`.`	The decimal character.	Sets the character that is used before the decimals.
`,`	The thousands separator.	Defines the thousands separator.
`%`	The percentage symbol.	Adds a percentage symbol.

Should you wish to create your own highly specific date and number formats, you can assemble them using the format code elements found in Tables 11-5 and 11-6.

Using the custom number formats is not difficult; but rather than explain all the permutations laboriously, here are a couple of examples to help you to see how they work:

- `FORMAT([Cost Plus Spares], "#,#.00")` gives you 44,500.00 (and any figure less than 1 will have a nothing to the left of the decimal).
- `FORMAT([Cost Plus Spares], "0.0")` gives you 12250.0 (and any figure less than 1 will have a 0 to the left of the decimal).

Remember that if you want to abandon a formula while you are creating it, all you have to do is click the cross icon in the formula bar or press Escape.

■ **Note** The `FORMAT()` function actually converts a number to text. Consequently, you might not be able to use a calculated column that is the result of a `FORMAT()` operation in further calculations.

Simple Logic: The IF( ) Function

Having data available is always a prerequisite for analysis; however, the raw data may not always lend itself to being used in dashboard visualizations in an ideal way.

DAX can help you to see "the wood for the trees" in the thicket of data that underlies your data model. Let's begin by looking at a series of practical examples that extend your data in ways that use the resources of Power BI Desktop to do the heavy lifting and let you focus on items that need your attention.

Exception Indicators

As a first example of how to use the `IF()` function, suppose that you want to highlight any records where the cost of spare parts is over £2.000.00. This means comparing the contents of the column PartsCost to a fixed value (3500). If this test turns out to be true (that is, the parts cost is over the threshold that you have set), then you want to display the words *Too Much!*

The following explains how to add a column that applies this test to the data. I will use the Stock table here.

1. Click the New Column button in the Modeling ribbon. A new column named Column will appear at the right of any existing columns.
2. To the right of the equals sign, enter **IF(**. You will see that as you enter the first few characters, the list of functions will list all available functions beginning with these characters.
3. Press the **[** key. The list of available fields will appear.
4. Scroll down through the list of fields and click the SpareParts field.
5. Enter the greater than symbol: >.
6. Enter **2000**.
7. Enter a comma.
8. Enter the following text (including the double quotes): **"Too Much!"**.
9. Enter a closing parenthesis: **)**. The code in the formula bar will look like this:

   ```
   Column = IF([SpareParts]>2000,"Too Much!")
   ```

10. Press Enter or click the tick icon in the formula bar. The new column will display Too Much! in any rows where the cost of spares is over £2,000.00.
11. Rename the column **Excessive Parts Cost**.

■ **Note** You might not see the results of a logical test like this one in the first few records of a table. So remember to scroll down the dataset and check that the formula has worked as you intended.

Like the `DIVIDE()` function, the `IF()` function can take up to three arguments (as the separate elements that you enter between the parentheses are called). The first two are compulsory:

- A *test*, in this case comparing the contents of a column to a fixed value
- The outcome if the test is *positive* (or TRUE in programming terms)

The `IF()` function can also have a third argument, although this is optional, as you can see in the example in this section:

- The outcome if the test is *negative* (or FALSE in programming terms)

This was a simple test to help you to isolate certain records. Later, when building dashboards, you can use the contents of this new column as the basis for tables, charts, and indeed just about any Power BI Desktop visualization.

Creating Alerts

When using the `IF()` function, the major focus is nearly always on the first argument—the test. After all, this is where you can apply the real force of Power BI Desktop. So here is another example of an `IF()` function being used, only this time it is to create an alert based on a slightly more complex calculation. This time, the objective is to detect records where the selling price of the vehicle is less than half the average sale price for all cars.

1. Click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon. A new column named Column will appear at the right of any existing columns.
3. To the left of the equals sign, replace the word Column with **Price Check**.
4. To the right of the equals sign, enter **IF(**. You will see that as you enter the first few characters, the list of functions will list all available functions beginning with these characters.
5. Press the **[** key. The list of available fields will appear.
6. Scroll down through the list of fields and click the CostPrice field.
7. Enter >= (it represents “greater than or equals to”).
8. Start typing the word **Average**. Power BI Desktop will display the list of available fields. When enough of the function name Average appears in the list of functions, select the `AVERAGE()` function.
9. Enter a left square bracket.
10. Start typing the name of the CostPrice field.
11. When the [CostPrice] field is visible in the popup list of fields, select it.
12. Add a right parenthesis. This ends the `AVERAGE()` function.
13. Enter ***2**.
14. Enter a comma.
15. Enter the following text (including the double quotes before and after the text): **“Price too high”**.

16. Enter a comma.
17. Enter **"Price OK"** (including the pair of double quotes).
18. Enter a closing parenthesis. This ends the `IF()` function. The code in the formula bar will look like this:

    ```
    PriceCheck = IF([CostPrice] >= AVERAGE([CostPrice]) *2,"Price too high", "Price OK")
    ```

19. Press Enter or click the tick icon in the formula bar. The new column will display Price too High or Price OK, depending on whether the cost price is more than half the average cost price or not.

You can then use the results of the cost price test as the basis for a visualization to compare the expensive purchases with the others.

Comparison Operators

When carrying out tests like the preceding example, you need to compare values. You may be familiar with the standard comparison operators that many programs and languages use (such as Excel), but for the sake of completeness, Table 11-7 provides a list of the most frequently used operators.

Table 11-7. *DAX Comparison Operators*

Operator	Description
=	Equals (exactly!)
<>	Not equals to
<	Less than
>	Greater than
<=	Less than or equals to
>=	Greater than or equals to

Flagging Data

As a practical example of how you might use an `IF()` function to validate data, imagine that Brilliant British Cars is embarking on a "know your customer" program and you envisage a chart that compares the clients that have reliable postcodes with those that do not. This way you can make a business case for cleansing the data and potentially rooting out certain clients.

For the moment, the Clients table either has or does not have a postcode (ZIP code) for each customer. What you want is a clear extra column that contains either HasPostCode or NoPostCode to indicate whether there is a postcode present.

In this example, you will not only test numeric values, but also look at whether a record contains a value for a row. This means introducing a new DAX function. This is the `ISBLANK()` function. It allows you to see if a column contains any data or not. Technically, this DAX function returns TRUE if the column is empty, and FALSE if it contains data. So you can nest it inside an `IF()` function to detect the presence of data, rather than looking at the data itself.

The following explains how to create a clear indicator of the presence or absence of a postcode:

1. Click the Clients table in the Fields list.
2. Click the New Column button in the Modeling ribbon.
3. To the left of the equals sign, replace Column with **IsPostCode**.
4. To the right of the equals sign, enter **IF(**. You will see that as you enter the first few characters, the list of functions lists all available functions beginning with these characters.
5. Type **IsB**. The list of functions will show ISBLANK().
6. Click ISBLANK() or press the Tab key to select this function. Power BI Desktop will place the function in the formula bar and add the left parenthesis automatically.
7. Press the **[** key. The list of available fields will appear.
8. Select [PostCode].
9. Enter a right parenthesis. (This finishes the `ISBLANK()` function.)
10. Enter a comma and then type **"NoPostCode", "HasPostCode"**. These are the outputs that the `IF()` function will return, depending on whether the column is blank or not.
11. Enter a final right parenthesis. The formula bar will display this:

    ```
    IsPostCode = IF(ISBLANK([PostCode]),"NoPostCode","HasPostCode")
    ```

12. Press Enter or click the tick icon in the formula bar. The new column will display either NoPostCode or HasPostCode for every Client.

You can now use this new field to filter data in dashboards or in tables and charts to separate the clients that do or do not have postcodes.

Nested IF() Functions

A frequent requirement in data analysis is to categorize records by ranges of values. Suppose, for instance, that you want to break down the stock of cars into low-, medium-, and high-mileage models. This requires more than a simple `IF()` function. However, it is not very difficult, as all that is needed is to "nest" one `IF()` function inside another, thereby extending the test that is applied to cover three possible outcomes.

Here, then, is how to create a simple nested `IF()` function:

1. Click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon.
3. To the left of the equals sign, replace Column with **Mileage Range**.
4. To the right of the equals sign, enter **IF(**. You see that as you enter the first few characters, the list of functions lists all available functions beginning with these characters.
5. Press the **[** key. The list of available fields will appear.

6. Select [Mileage]. You can type it fully if you prefer, but remember to add the right square bracket if you do.
7. Enter <= **50000**.
8. Enter a comma.
9. Enter **"Low"** (including the double quotes).
10. Enter a comma.
11. Enter **IF(**.
12. Select (or enter) the Mileage column.
13. Enter < **100000**.
14. Enter a comma.
15. Enter **"Medium", "High"**. This must include the double quotes and the comma separating the two words.
16. Enter two right parentheses, one for each of the `IF()` functions.

    The formula bar will display

    ```
    Mileage Range = IF([Mileage] <= 50000, "Low", IF([Mileage] < 100000,
    "Medium","High"))
    ```

17. Press Enter or click the tick icon in the formula bar. The new column will display Low, Medium, or High for every vehicle in the new Mileage Range column.

You have now categorized all the cars in stock by their mileage, and can use the category flag in the Mileage Range column to create, for instance, a chart that shows the number of vehicles corresponding to each mileage category.

A nested `IF()` function works like this:

1. You set up a first test. In this example, the test is to flag all cars that have less than 50,000 miles "on the clock."
2. You specify what the outcome is if this initial test is positive. In this example, the word *Low* appears in the column.
3. You then add a second test. By definition, this will *only* apply to cars that have traveled more than 50,000 miles; otherwise, the formula returns the word *Low*. So you add a higher threshold for the second test, which is 10,0000 miles in this example.
4. If the record passes the test and the vehicle has traveled less than 100,000 miles, then the word *Medium* will appear in the column. In all other cases (that is, for all mileage over 100,000 miles), the word *High* is displayed in the column.

When writing nested `IF()` statements, the essential trick is to use a sequence of tests that follows a logical order, from lowest to highest (or in some cases, from highest to lowest). This way, the succession of `IF()` statements acts like a series of hoops that catch the values and return an appropriate result.

You can nest up to 64 `IF()` statements in a single DAX expression. In fact, you can nest a maximum of 64 DAX expressions, whatever they are. However, more than half a dozen can be painful to write correctly, and getting the correct number of right parentheses in place can be tricky. However, there may be many occasions when you need to segment your data for your visualizations and dashboards, even if it means grappling with complex nested `IF()` statements. So let's take a look at one of these to whet your appetite.

Creating Custom Groups Using Multiple Nested IF() Statements

To give you another example of a slightly more complex DAX function (but one that can be very necessary), consider the following requirement. Our data now has the car age, but we want to group the cars by age segments (or buckets, if you prefer). So we will use a nested IF() function to do this. Then, to allow us to sort the column in a more coherent way, we will create a Sort By column for the new Vehicle Age Category column that we will create. If you remember, you saw how to create and use Sort By columns in the previous chapter.

In this example, I will not explain every step, as you have seen how to select functions and fields from popups in the previous examples. Instead, I prefer to concentrate on the logic itself and explain how complex IF() statements can be built.

The code for the Vehicle Age Category column is as follows:

```
Vehicle Age Category=IF(
  [VehicleAgeInYears] <=5,"Under 5",
  IF(AND([VehicleAgeInYears]>=6,[VehicleAgeInYears]<=10),"6-10",
    IF(AND([VehicleAgeInYears]>= 11,[VehicleAgeInYears]<=15),"11-15",
     IF(AND([VehicleAgeInYears]>=16,[VehicleAgeInYears]<=20),"16-20",
       IF(AND([VehicleAgeInYears]>=21,[VehicleAgeInYears]<=25),"21-25",
        IF(AND([VehicleAgeInYears]>=26,[VehicleAgeInYears]<=30),"26-30",
          ">30"
         )
        )
      )
     )
    )
  )
```

The only slight problem with a great technique for segmenting data is that if you sort the Vehicle Age Category column, you will find that the category that corresponds to the highest age appears at the top of the list. So you need to add a second column that can be used as a sort order column for the new column that you just created. The following is the code for this Vehicle Age Category Sort column:

```
Vehicle Age Category Sort=IF([VehicleAgeInYears]<=5, "1",
  IF(AND([VehicleAgeInYears]>=6, [VehicleAgeInYears]<=10),"2",
    IF(AND([VehicleAgeInYears]>= 11, [VehicleAgeInYears]<=15), "3",
     IF(AND([VehicleAgeInYears]>=16, [VehicleAgeInYears]<=20), "4" ,
       IF(AND([VehicleAgeInYears]>=21 , [VehicleAgeInYears]<=25), "5",
        IF(AND([VehicleAgeInYears]>=26, [VehicleAgeInYears]<=30),"6","7"
         )
        )
      )
     )
   )
  )
```

These formulas could have come straight from an Excel spreadsheet. Indeed, some 80 of the DAX functions are nearly identical to their Excel cousins. So experience and imagination combined have shown me that you have many ways to extend the data you imported by adding calculated columns. Even better, all calculated columns are updated when you refresh the data from the source. The only major caveat is that when you are tweaking the data connection, you must be careful *not* to delete any source columns on which a calculated column depends, or else you will get errors in the Power BI Desktop table.

■ **Tip** You could have created the formula in this example without creating the `VehicleAgeInYears` column first, as you could have used the formula that calculates the age of the car each time that you need the vehicle age. However, as you can imagine, it is easier to create a column that contains the vehicle age first and then refer to this in the Vehicle Age Category formula. This makes the more complex formula easier to read. It also helps you to break down the analytical requirement into successive steps, which is good DAX development practice. Moreover, you can always hide any "intermediate" columns if you do not need them in dashboards and reports. Indeed, you could even do this kind of calculation in Power BI Desktop Query.

Multiline Formulas

By default, all formulas that you create in Power BI Desktop will be on a single line that overflows onto the next line when there is no more room in the formula bar. This can become an extremely tedious way of working, so it is worth knowing that you can tweak long formulas to force them to display over more than one line. All you have to do is force a line return inside the formula bar by pressing Shift+Enter where you want to force a new line. My experience is that Power BI Desktop will not let you create line breaks everywhere in a formula. Nonetheless, with a bit of trial and error, a more complicated formula, such as the `VehicleAgeInYears` column that you created, can look like the multiline formula in Figure 11-8.

Just in case you were wondering, you do *not* have to write formulas over multiple lines as I did just. Indeed, the two formulas used to create complex nested `IF()` statements could be written as follows:

```
Vehicle Age Category=IF([VehicleAgeInYears] <=5,"Under 5",
IF(AND([VehicleAgeInYears]>=6,[VehicleAgeInYears]<=10),"6-10",
IF(AND([VehicleAgeInYears]>= 11,[VehicleAgeInYears]<=15),"11-15",
IF(AND([VehicleAgeInYears]>=16,[VehicleAgeInYears]<=20),"16-20",
IF(AND([VehicleAgeInYears]>=21,[VehicleAgeInYears]<=25),"21-25",
IF(AND([VehicleAgeInYears]>=26,[VehicleAgeInYears]<=30),"26-30","Over 30"))))))
```

And

```
Vehicle Age Category Sort=IF([VehicleAgeInYears] <=5,"1",
IF(AND([VehicleAgeInYears]>=6,[VehicleAgeInYears]<=10),"2",
IF(AND([VehicleAgeInYears]>= 11,[VehicleAgeInYears]<=15),"3",
IF(AND([VehicleAgeInYears]>=16,[VehicleAgeInYears]<=20),"4",
IF(AND([VehicleAgeInYears]>=21,[VehicleAgeInYears]<=25),"5",
IF(AND([VehicleAgeInYears]>=26,[VehicleAgeInYears]<=30),"6","7"))))))
```

I chose to write the formulas over multiple lines, hoping that by doing so, I'd make the nested logic clearer. You can write your formulas in any way that suits you and that does not cause Power BI Desktop a problem.

Complex Logic

Categorizing data can sometimes involve applying logic that is more complex than a single simple comparison. You could need to apply two or more conditions when evaluating a record, and this more intricate logic could require you to test the contents of more than one column.

Once again, DAX can help you in circumstances like these. To explain by example, consider the following analytical challenge. You want to flag any vehicle that is a red or blue coupe. This example will show the basics of applying complex logic to data analysis with DAX.

1. In the CarSalesDataWithDataModel.pbix file, click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon.
3. To the left of the equals sign, replace Column with **Special Sales**.
4. To the right of the equals sign, enter **IF()**. Since we are dealing with multiple parentheses in this formula, I prefer to enter both the opening and the closing parentheses for each function when adding the function.
5. Click inside the parentheses.
6. Type the **AND()** function and then click inside the parentheses. This function ensures that multiple logical conditions are applied and that all must be satisfied for the test to be successful.
7. Type **[VehicleType] = "Coupe",** (including the comma). This is one of the conditions that has to be true for the test to be successful.
8. After the comma, type **OR()** and then click inside the parentheses. This is a second condition (it is still part of the `AND()` function), but it can be one of many different tests.
9. Type (or use the popup menus to select as well as partially typing) **RELATED(Colors[Color]) = "Red", RELATED(Colors[Color]) = "Blue"**. This is the second part of the test. However, because the color is in another table, you have to use the `RELATED()` function to find the color of the vehicle because it is not in the Stock table. Also there are two alternative conditions inside the parentheses for the `OR()` expression.
10. Click just inside the final parenthesis at the right of the DAX expression (this is the one that ends the `IF()` function) and type **,"Special" ,"Normal"**. Be sure to include the commas. The formula should read

    ```
    Special Sales = IF(AND([VehicleType] = "Coupe", OR(RELATED(Colors[Color]) =
    "Red", RELATED(Colors[Color]) = "Blue")),"Special","Normal")
    ```
11. Press Enter or click the tick icon in the formula bar. The new column will display either Special or Normal for every vehicle in the new Special Sales column

I realize that a formula like this can seem daunting at first sight. So let's take another look at this DAX expression formatted a little differently:

```
Special Sales = IF(
                   AND(
                         [VehicleType] = "Coupe",
                         OR(RELATED(Colors[Color]) = "Red",
                         RELATED(Colors[Color]) = "Blue")
                      )
                   ,"Special"
                   ,"Normal"
                   )
```

As you can see, at its heart, the expression is an `IF()` expression. As such, it consists of three parts:

- A *test* (the car is a coupe that is either red or blue)
- An *outcome for a positive result* (displays "Special")
- An *outcome for a negative result* (displays "Normal")

The only tricky bit now is the test itself. Since it is built on logic that is more complex, it requires a little explanation:

- First, you have stated that the test is in several parts, all of which must be true for a record to pass the test. This is done by using the `AND()` function and then separating each individual test (of which there are two in this example—the vehicle type and the color).
- Second, you have told DAX that the second test (on the color) can be any of several possibilities (two in this example). You did this using the `OR()` function and separating each individual test by a comma.

Although I prefer to build complex functions from the inside out (that is, by adding all the required parentheses first and adding what goes inside them second), this is not an obligation. You are free to build DAX formulas in any way that works.

■ **Note** This example only showed two alternatives for the `AND()` and `OR()` functions. This is because these functions are limited to only two parameters. What is important to remember is that you will have *to repeat the field* (and possibly the table name if it is not the current table) for *each comparison*, just as you did here when testing the colors of the cars. If you need more than two alternatives for an AND or an OR operation, then you will have to use the logical *operators* (`&&`, `||`, and `!`) that are described in the upcoming "Logical Operators" section.

Armed with this knowledge, you can now build extremely complex logical tests on your data. If you are an Excel or Access power user, then the learning curve should be quite short as the principles and functions are similar to those that you are using already. If you have come from the world of programming, then the concepts are probably familiar. If you are just starting out, then just be prepared to spend a little time practicing, and above all, analyze the question that you want DAX to answer *before* starting to write the statement.

DAX Logical and Information Functions

So far in this chapter, you have seen three of the DAX logical functions. In practice, you may need to build formulas that use some of the other functions that DAX provides to apply logic and to test the state and type of information in columns. Table 11-8 describes the essential functions for creating complex data models.

Table 11-8. *DAX Logical Functions*

Operator	Description	Example
`IF()`	Tests a condition and applies a result if the test is true, and possibly a result if the test is false.	`IF([PartsCost]> 500, "Check Parts", "OK")`
`AND()`	Extends the logic to include several conditions *all* of which must be met.	`IF(AND([PartsCost]> 500,[LaborCost]>1000), "Repair Cost Excessive", "OK")`
`OR()`	Extends the logic to include several conditions *any* of which must be met.	`IF(OR([PartsCost]> 500,[LaborCost]>1000), "Repair or Labor Cost issue", "OK")`
`NOT()`	Extends the logic to include several conditions *none* of which must be met.	`IF(NOT([PartsCost]> 500,[LaborCost]>1000), "No Repair or Labor Cost issue", "")`
`ISERROR()`	Tests a value and returns TRUE if there is an error value.	`IF(ISERROR([PartsCost]), "Check parts", "")`
`TRUE()`	Returns TRUE.	`IF(Stock[Mileage] > 100000, TRUE(), FALSE())`
`FALSE()`	Returns FALSE.	`IF(Stock[Mileage] > 100000, TRUE(), FALSE())`
`ISNUMBER()`	Detects if a column value is numeric.	`IF(ISNUMBER([PartsCost]), "", "Data Error")`
`ISTEXT()`	Detects if a column value is a text.	`IF(ISTEXT([PartsCost]), "Data Error", "")`
`ISNONTEXT()`	Detects if a column value is not a text and is not a blank.	`IF(ISNONTEXT([PartsCost]), "", "Data Error")`
`ISODD()`	Detects if a value is an odd number.	`IF(ISODD([PartsCost]), "Data Error", "")`
`ISEVEN()`	Detects if a value is an even number.	`IF(ISEVEN([PartsCost]), "", "Data Error")`
`ISLOGICAL()`	Detects if a column value is a true or false.	`IF(ISLOGICAL([PartsCost]), "Data Error", "")`

Logical Operators

If you are writing more complex logical statements when specifying intricate conditions that must be met, then DAX has an alternative to the `AND()`, `OR()`, and `NOT()` functions. These are called *logical operators*, which are explained in Table 11-9.

Table 11-9. *DAX Logical Operators*

Operator	Description	Example
`&&`	AND	`[Color] = "Red" && [VehicleType] = "Coupe"`
`||`	OR	`[Color] = "Red" || [Color] = "Blue"`
`!`	NOT	`[Color] = "Red" !  [VehicleType] = "Coupe"`

As a simple example, you could want a choice of three possible colors in a logical operation. The code for this would read

```
[Color] = "Red" || [Color] = "Blue" Color] || [Color] = "Green"
```

Formatting Logical Results

Sometimes a logical function might exist only to return a simple true or false. For instance, you could want to test a value and indicate if it is over a certain threshold, using a formula like the following (added to the Stock table):

```
High Mileage = IF(Stock[Mileage] > 100000, TRUE(), FALSE())
```

This formula simply tests the mileage figure for each record and returns `TRUE()` if the mileage is greater than 100,000 miles; it returns `FALSE()` in all other cases.

However, you might not want to display simply TRUE or FALSE in the column. So DAX also lets you format logical output, whether it is calculated like it is here or imported as a TRUE or FALSE value from a data source. The formula that you just saw can be formatted like this:

```
High Mileage = FORMAT(IF(Stock[Mileage] > 100000, TRUE(), FALSE()),"Yes/No")
```

There are only three logical formats available in DAX. These are explained in Table 11-10.

Table 11-10. *DAX Logical Formats*

Format Code	Description
Yes/No	Formats the output as either Yes or No.
True/False	Formats the output as either True or False.
On/Off	Formats the output as either Yes or No.

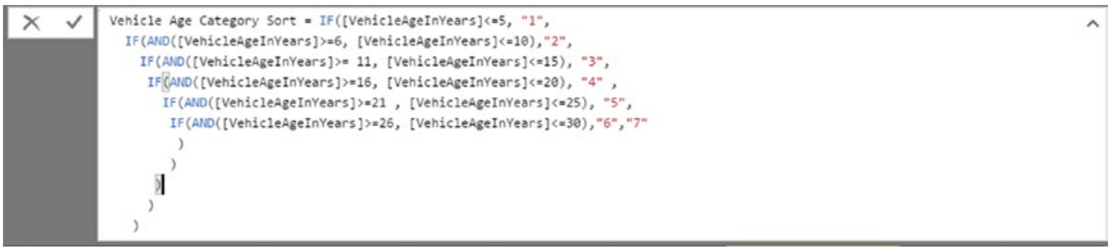
■ **Note** Different data sources represent True in different ways; however, nearly all represent False as a zero. Consequently, DAX interprets a logical column as a number, any zeros as a False, and anything else as a True.

Making Good Use of the Formula Bar

If you only ever enter simple formulas, then not only will you be extremely lucky, but you can also content yourself with a single line in the formula bar. I doubt that this is likely to be the case, however, because you will want to do great things with Power BI Desktop. It follows that you may soon be tired of creating long and complex DAX formulas in a limited space. So here is how to expand the formula bar—pretty much as you would in Excel:

Click the Expand icon at the right of the formula bar (the downward-facing chevron).

The formula bar increases in height to allow you to type and see several lines of text. To reduce the height of the formula bar and reset it to a single line, just click the Reduce icon at the right of the formula bar (which has now become an upward-facing chevron). You can see this icon in Figure 11-8.



```
Vehicle Age Category Sort = IF([VehicleAgeInYears]<=5, "1",
  IF(AND([VehicleAgeInYears]>=6, [VehicleAgeInYears]<=10),"2",
    IF(AND([VehicleAgeInYears]>= 11, [VehicleAgeInYears]<=15), "3",
     IF(AND([VehicleAgeInYears]>=16, [VehicleAgeInYears]<=20), "4" ,
       IF(AND([VehicleAgeInYears]>=21 , [VehicleAgeInYears]<=25), "5",
        IF(AND([VehicleAgeInYears]>=26, [VehicleAgeInYears]<=30),"6","7"
         )
        )
      )
     )
   )
```

Figure 11-8. *Multiline formulas*

Conclusion

This chapter introduced you to some of the core techniques that you can apply to extend a Power BI Desktop data model with further metrics. These additional elements were in the form of new columns that you added to many of the data tables that you had previously loaded, cleansed, and assembled into a structured data model.

All the added columns were based on DAX, the Power BI Desktop formula language. As you saw, this language is not especially difficult and it is fairly close to the Excel formula language.

You also learned how to concatenate fields and how to perform basic arithmetic. Then you practiced carrying out calculations that involve multiple tables. Finally, you learned how to segment records using logical functions.

This brief introduction is nonetheless only a quick foretaste of the power of DAX. There is much, much more that can be accomplished to prepare the quantitative analyses that you are likely to need to produce telling visuals with Power BI Desktop. It is time to move on to the next chapter and take a look at the next feature of DAX: creating measures.

CHAPTER 12

Adding Measures to the Data Model

Adding new columns can provide much of the extra data that you want to output in tools like Power BI Desktop. It is unlikely, however, that this approach can deliver *all* the analyses that you need. Specifically, calculated columns can *only* work on a row-by-row basis; they cannot contain formulas that have to apply to all or part of the records in a table. For instance, counting the number of cars sold for a year, a quarter, or a month has nothing to do with the data in a single row in the Stock table. It does, however concern the table as a whole.

Generally, you need to add a second type of formula to your tables when you have to look at subsets of the data. These formulas are called, simply, *measures*. These calculations (or measures or metrics—call them what you will) also use DAX. They are applied differently, though, and they can produce some extremely powerful results to help you analyze your data. This is because measures do things that calculated columns simply cannot do. So if you need to work with aggregate values and not on a row-by-row basis, then you will have to create measures to achieve the correct result.

As with so many aspects of Power BI Desktop and self-service business intelligence in general, measures are probably best introduced through a few examples. Unfortunately, it is impossible to do anything other than scratch the surface of measures in only a few pages, because they are arguably the most powerful element in Power BI Desktop—one that deserves an entire book to itself. Nonetheless, I hope that this short introduction will whet your appetite, and that you will then continue to learn all about DAX and its more advanced application from the many excellent resources currently available.

In this chapter, we continue to develop the file that you began in the previous chapter. The file with all the columns that were added in the previous chapter is available for download as C:\PowerBiDesktopSamples\CH12\CarSalesDataWithNewColumns.pbix.

A First Measure: Number of Cars Sold

Suppose that you want to be able to display the number of cars sold in Power BI Desktop. Not only that, but you want this figure to adjust when it is filtered or sliced by another criterion, such as country or color. Put simply, you want this metric to be infinitely sensitive to how it is displayed, yet always give the right answer.

So how are we going to achieve this? The following explains how:

1. Open the Power BI Desktop file C:\PowerBiDesktopSamples\CH12\CarSalesDataWithNewColumns.pbix.

2. Ensure that you are in Data View (the middle of the view icons on the left below the ribbon).

3. Select the table to which you wish to add a measure. I chose Stock here.

4. In the Modeling ribbon, click the New Measure button. The formula bar will look like Figure 12-1.

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_12

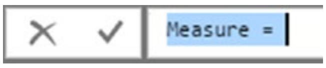

Measure =

Figure 12-1. *The formula bar when creating a measure*

5. Add the following formula to the formula bar:

   ```
   NumberOfCarsSold=COUNTROWS(
   ```

   You will see that the popup will then suggest a list of DAX formulas interspersed with the names of tables in the current data model. If you scroll down the list, it will look like Figure 12-2.

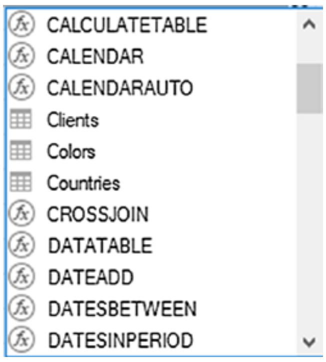


Figure 12-2. *The popup menu showing functions and tables*

6. Select the table Stock and add the right parenthesis. The formula will look like this:

   ```
   NumberOfCarsSold = COUNTROWS(Stock)
   ```

7. Confirm the creation of the formula by pressing Enter or by clicking the check mark icon in the formula bar. A new field will appear in the Fields list after any existing fields for the Stock table.

Assuming that you have just read Chapter 11, the first thing that will strike you in comparison with creating a new column is that no column is created for this measure. The only indication that it exists is its presence in the Fields list once you expand the Stock table. If you look closely at the field (NumberOfCarsSold) that you have just created, you will also see that there is a tiny icon of a calculator to the left of the field name. This allows you to distinguish measures from new columns (which have a small Fx icon to the left of the field name).

Not difficult, I am sure you will agree. Yet the best is yet to come. Suppose that you now use this field in a Power BI Desktop card (you saw these briefly in Chapter 1, and can see them in more detail in Chapter 15). The result is filtered so that only the number of sales for 2014 is displayed. In other words, the formula is completely separate from the data in a column, but applies any filters that are selected. You can see this applied to a visualization in Figure 12-3.

***Figure 12-3.** Using a measure in a card visualization*

The key thing to take away is that a correctly applied measure can be used in a Power BI Desktop table, chart, or indeed any type of visualization, and always shows the correct result of any and all filters and slicers that you have applied. Also, the figures are correct for each intersection of rows and columns in tables. All in all, it is well worth ensuring that you have all the measures that you need for your analytical output in place and that they are working correctly in Power BI Desktop, because you can then rely on these calculations in the dataset in many different visualizations across a series of reports.

■ **Tip** You can rename measures by either right-clicking the name of the measure in the Fields list and selecting Rename, or by clicking the name of the measure in the Fields list and altering the name in the formula bar.

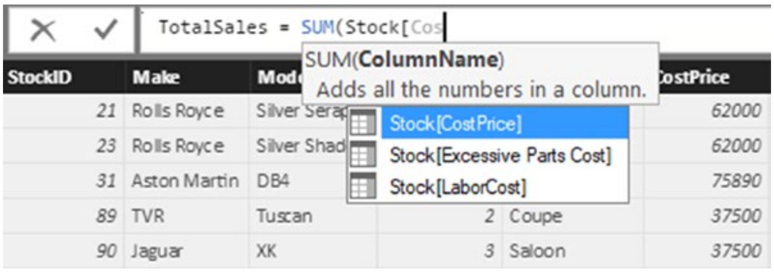
When you start out creating measures, it can be a little disconcerting at first not to see the results of a formula immediately, as you can when adding new columns. If this worries you (or if you want to test the result of a new measure), then one approach is to create a table in Dashboard View and add the new measure plus any other useful measures that allow you to verify that everything works as you expected. This technique was outlined briefly in Chapter 1 and is explained in detail in Chapter 14.

Basic Aggregations in Measures

Measures are DAX formulas, so in learning to use measures you will have to become familiar with some more DAX functions. My intention here, though, is definitely not to take you through all that DAX can offer. Instead, I would like to show you a few basic formulas that can be useful in real-world dashboards and give you some initial DAX recipes that should prove practical.

So, as a second example, let's calculate total costs of vehicles purchased. Although you can just type in a simple DAX formula, I prefer to show you how you can extend the knowledge that you gained when creating calculated columns and apply many of the same techniques to creating measures.

1. In Power BI Desktop, ensure that you are in Data View.
2. Select the Stock table and click the New Measure button in the Modeling ribbon.
3. Replace Measure with the name that you want to use (**Total Sales**).
4. Click to the right of the equals sign (=).
5. Enter **SUM** as the function, followed by a left parenthesis. A list of all the tables and fields in the data model will appear (including any columns and measures that you have added).
6. Enter a left bracket to restrict the popup list to fields in the current table.
7. Start typing the field name (**CostPrice** in this example). After a couple of characters, any tables or fields with these characters will be listed, as shown in Figure 12-4.



```
TotalSales = SUM(Stock[Cos
SUM(ColumnName)
Adds all the numbers in a column.
Stock[CostPrice]
Stock[Excessive Parts Cost]
Stock[LaborCost]
```

StockID	Make	Mod			ostPrice
21	Rolls Royce	Silver Serap			62000
23	Rolls Royce	Silver Shad			62000
31	Aston Martin	DB4			75890
89	TVR	Tuscan	2	Coupe	37500
90	Jaguar	XK	3	Saloon	37500

Figure 12-4. *Creating a measure containing an aggregation*

8. Scroll down and select the [CostPrice] field.
9. Add a right parenthesis. The formula should read:

   ```
   TotalSales=SUM(Stock[CostPrice])
   ```

10. Press Enter (or click the check mark icon in the formula bar).
11. Format the new column appropriately (I have used pounds sterling).

The measure is created and it appears in the Fields list. This particular function gives you the total of the SalePrice column. However, when you use it in Power BI Desktop, it is filtered and applied (or sliced and diced if you prefer) to take into account how the data is subset.

One important thing to note when creating measures is that you should use the table name as well as the field name if there are fields that have the same name in several tables. This is so that Power BI Desktop is certain that it is using the right field from the right table. What's more, if a table name contains spaces, then the table name needs to be in single quotes. In all cases, the field name has to be enclosed in square brackets.

To practice a little (and to prepare the ground for some eye-catching visualizations in Chapters 14–22), try creating the average, maximum, and minimum sale price using the formulas in Table 12-1.

Table 12-1. *A Few Elementary DAX Measures*

Name	DAX Code
Average Cost Price	`AVERAGE(Stock[CostPrice])`
Maximum Sale Price	`MAX(InvoiceLines[SalePrice])`
Minimum Sale Price	`MIN(InvoiceLines[SalePrice])`

Using Multiple Measures

As you can well imagine, not all metrics are likely to be as simple as those that you just saw. You can also create measures that are the result of combining several DAX functions.

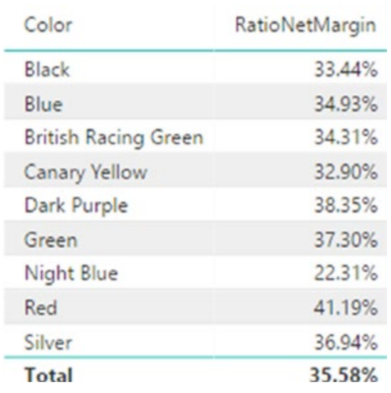
For a little practice, you could try adding the ratio of gross margin to sale price to the data model. This measure will then be used in the upcoming chapters on creating dashboards with Power BI Desktop. If you do not want to create this measure step by step, then you can jump to step 13 and type or paste the formula into the formula bar.

1. In Power BI Desktop, ensure that you are in Data View.
2. Select the InvoiceLines table and click the New Measure button in the Modeling ribbon.
3. Replace Measure with the name that you want to use (**RatioNetMargin**).
4. Click to the right of the equals sign (=).
5. Enter **SUM** as the function, followed by a left parenthesis.
6. Enter a left bracket to limit the list to fields in the current table.
7. Select the [Gross Margin] field.
8. Enter a right parenthesis.
9. Enter a forward slash (the divide-by operator).
10. Enter **SUM** as the function, followed by a left parenthesis.
11. Enter a left bracket to limit the list to fields in the current table.
12. Select the [SalePrice] field.
13. Enter a right parenthesis. The formula should read

    ```
    RatioNetMargin = SUM([Gross Margin])/SUM([SalePrice])
    ```

14. Press Enter (or click the check mark icon in the formula bar).
15. Power BI Desktop should guess that this is a ratio, and format the column as a percentage accordingly. If it does not, in the Modeling ribbon, click the percentage button to apply a percentage format.

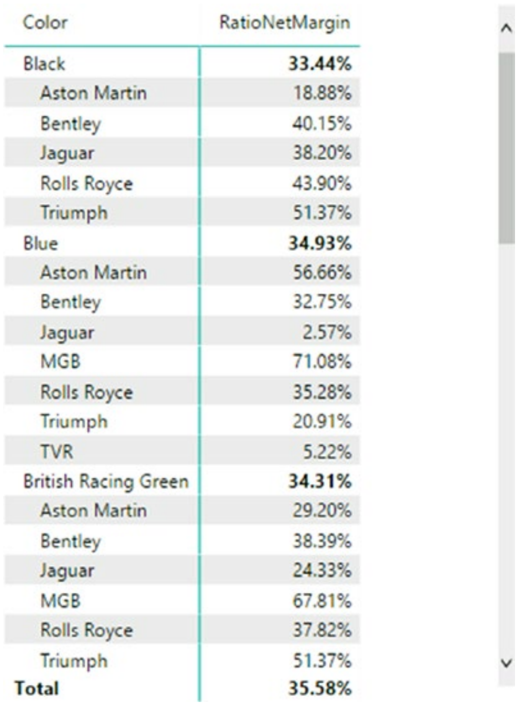
The easiest way to see the output of a measure like this is to create a table that uses the measure and another attribute so that you can see how the measure works in practice. Figure 12-5 shows a simple example of this, but for all years' data.



Color	RatioNetMargin
Black	33.44%
Blue	34.93%
British Racing Green	34.31%
Canary Yellow	32.90%
Dark Purple	38.35%
Green	37.30%
Night Blue	22.31%
Red	41.19%
Silver	36.94%
Total	**35.58%**

Figure 12-5. *Applying a measure in a table*

This is an extremely simple example of a composite DAX function, of course. Indeed, you can probably see a distinct resemblance to an Excel formula. However, the key point to take away is that once you have created the measure, it will work in just about any Power BI Desktop visualization and using most, if not all, of the attributes from the data model. Power BI Desktop can also apply the measure intelligently to hierarchies of data. So, for instance, if you add the Make field to Table 12-1 and switch the visualization to a matrix, you will instantly see the calculations that are shown in Figure 12-6. Here, Power BI Desktop has automatically calculated the net margin ratio for each vehicle sold without your having to alter the formula in any way.



Color	RatioNetMargin
Black	**33.44%**
Aston Martin	18.88%
Bentley	40.15%
Jaguar	38.20%
Rolls Royce	43.90%
Triumph	51.37%
Blue	**34.93%**
Aston Martin	56.66%
Bentley	32.75%
Jaguar	2.57%
MGB	71.08%
Rolls Royce	35.28%
Triumph	20.91%
TVR	5.22%
British Racing Green	**34.31%**
Aston Martin	29.20%
Bentley	38.39%
Jaguar	24.33%
MGB	67.81%
Rolls Royce	37.82%
Triumph	51.37%
Total	**35.58%**

Figure 12-6. *A hierarchy using a measure*

When you use measures like this one in visualizations, you may well find that some calculations are displayed to many decimal places. If you find this distracting, then you can format measures in the same way that you format Power BI Desktop columns. Any formats that you apply are used in Power BI Desktop by default whenever you use this measure.

A final point. When you insert a table or a field from the popup list shown in Figure 12-4, you see three types of icons to the left of the table or field: the icon with a table outline denotes a Power BI Desktop table; the table with a selected column icon indicates a column of data or a column that you have added; a calculator icon indicates an existing measure.

Cross-Table Measures

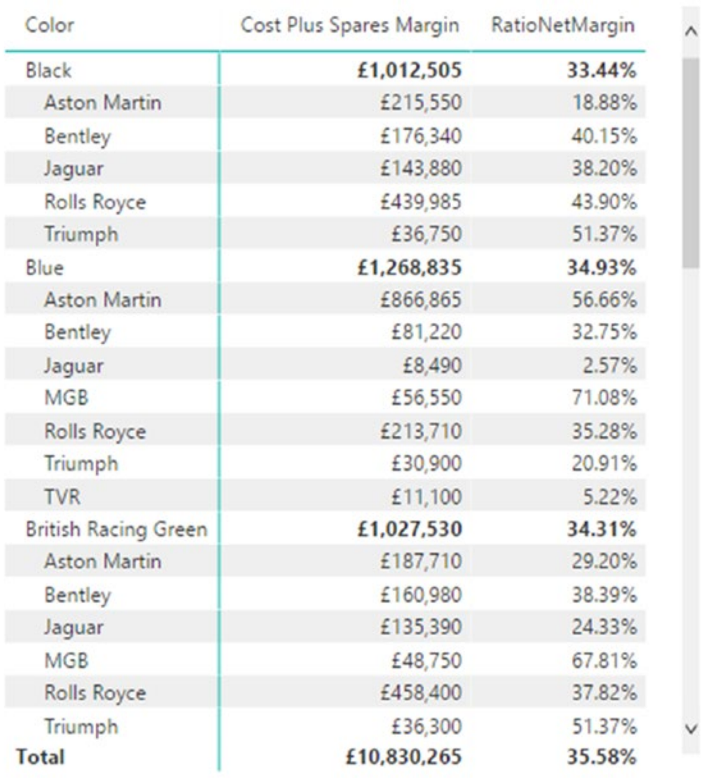
You are not limited to creating measures that refer to the fields in a single table. If anything, measures are designed to apply across *all* the fields in a data model. To start out with a simple example, suppose that you want to create a custom measure that displays the margin for each vehicle once the cost price and any cost of spares have been deducted. If you just want to type in the formula, then jump directly to step 12.

1. In Power BI Desktop, ensure that you are in Data View (the middle of the view icons on the left below the ribbon).
2. Select the InvoiceLines table and click the New Measure button in the Modeling ribbon.
3. Replace Measure with the name that you want to use (**Cost Plus Spares Margin**).
4. Click to the right of the equals sign (=).
5. Enter **SUM** as the function, followed by a left parenthesis. A list of all the tables and fields in the data model will appear (including any columns and measures that you have added).
6. Select InvoiceLines[SalePrice].
7. Add a minus sign after the formula (you can add spaces before and afterward if you want).
8. Enter **SUM** as the function, followed by a left parenthesis. A list of all the tables and fields in the data model will appear (including any columns and measures that you have added).
9. Select InvoiceLines[SalePrice].
10. Add a minus sign after the formula (you can add spaces before and afterward if you want).
11. Enter **SUM(** and then expand the Stock table in the Fields list and click the CostPrice field.
12. Add a right parenthesis. The formula will look like this:

```
Cost Plus Spares Margin = SUM(InvoiceLines[SalePrice]) - SUM(Stock[CostPrice]) -
SUM(Stock[SpareParts])
```

13. Press Enter (or click the check mark icon in the formula bar).

You could then add this new measure to the matrix that you saw previously. If you do, you see something like Figure 12-7.



Color	Cost Plus Spares Margin	RatioNetMargin
Black	**£1,012,505**	**33.44%**
Aston Martin	£215,550	18.88%
Bentley	£176,340	40.15%
Jaguar	£143,880	38.20%
Rolls Royce	£439,985	43.90%
Triumph	£36,750	51.37%
Blue	**£1,268,835**	**34.93%**
Aston Martin	£866,865	56.66%
Bentley	£81,220	32.75%
Jaguar	£8,490	2.57%
MGB	£56,550	71.08%
Rolls Royce	£213,710	35.28%
Triumph	£30,900	20.91%
TVR	£11,100	5.22%
British Racing Green	**£1,027,530**	**34.31%**
Aston Martin	£187,710	29.20%
Bentley	£160,980	38.39%
Jaguar	£135,390	24.33%
MGB	£48,750	67.81%
Rolls Royce	£458,400	37.82%
Triumph	£36,300	51.37%
Total	**£10,830,265**	**35.58%**

Figure 12-7. *Cross-table measures*

Creating cross-table measures in DAX is easier than creating new columns that use values from more than one table. From this example, you can see the following:

- You do not need to use the `RELATED()` function.
- You only have to specify the table name before entering the field name (or select the combination of table and field from the popup).
- You can use the Fields list to refer to fields in other tables (or even in the same table).
- You *must* use aggregation functions on numeric fields. If you do not, you will get an error message.

■ **Note** A measure is attached to a table so that it appears as a field in the specific table. The measure does not have to use any of the fields in the table that "hosts" it. This means that you can attach measures to any table in your data model, which allows for a considerable organizational freedom when extending the model with further metrics. Moreover, you can move measures between tables if you want.

■ **Tip** You can even create an otherwise empty table (using the technique that you saw in Chapter 2) to serve as a container for measures.

More Advanced Aggregations

Now that you have seen how to create basic measures, it is time to move on to some more advanced concepts. More precisely, I want to outline a couple of ways to aggregate data on a row-by-row basis, yet return the result with any filters and slicers applied. There are many cases where this *cannot* be done using a calculated column and then returning the aggregate of the column data. After all, you need to return the *ratio of the sum* of any values and *not* the *sum of the ratio*. Think of calculating a ratio for each row and then averaging the results to get the average ratio; it is arithmetically false.

Fortunately, Power BI Desktop has some simple yet powerful solutions to this kind of conundrum. One principal tool is the use of the "X" functions—`AVERAGEX()`, `COUNTX()`, `SUMX()`, `MAXX()`, and `MINX()`, among others. These functions allow you to specify

- The table in which the calculations apply
- The row-by-row calculation that is to be applied

As an example, consider the requirement for the ratio of the cost of any parts compared to the purchase price of each vehicle. Not only do we need this potentially at the finest level of granularity—the individual record—but we may need it sliced and diced by any number of criteria. To extend your knowledge, we will also create this measure directly in the Report View, without stepping sideways into the Data View. The following explains how to create the formula that you could use in Power BI Desktop reports (and step 15 contains the final formula if you only want this):

1. In Power BI Desktop, ensure that you are in Report View (the topmost of the view icons on the left below the ribbon).
2. Select the Stock table in the Fields list and click the New Measure button in the Modeling ribbon.
3. Replace Measure with the name that you want to use (**Average Parts Cost Ratio**).
4. Click to the right of the equals sign (=).
5. Enter **AVERAGEX** as the function, followed by a left parenthesis. A list of all the tables and fields in the data model will appear (including any columns and measures that you have added).
6. Select the Stock table.
7. Enter a comma.
8. Add a left parenthesis. This is to ensure that the subtraction is carried out before the division.
9. Enter a left bracket and select the [CostPrice] field or enter the field name, including the square brackets.
10. Add a minus sign after the field name (you can add spaces before and after if you want).
11. Enter a left bracket and select the SpareParts field or enter the field name including the square brackets.
12. Enter a right parenthesis. This matches the opening left parenthesis in step 7.
13. Add a forward slash (the divide-by operator).
14. Enter a left bracket and select the [SpareParts] field or enter the field name, including the square brackets.

15. Enter a right parenthesis. This finishes the `AVERAGEX()` function. The formula will look like this:

```
Average Parts Cost Ratio = AVERAGEX(Stock,([CostPrice]-[SpareParts]) /
[SpareParts])
```

16. Press Enter (or click the check mark icon in the formula bar).

Just creating the formula is pretty meaningless. So take a look at the chart in Figure 12-8, where you can see how this ratio instantly shows you which models are the most costly as far as spare parts are concerned.

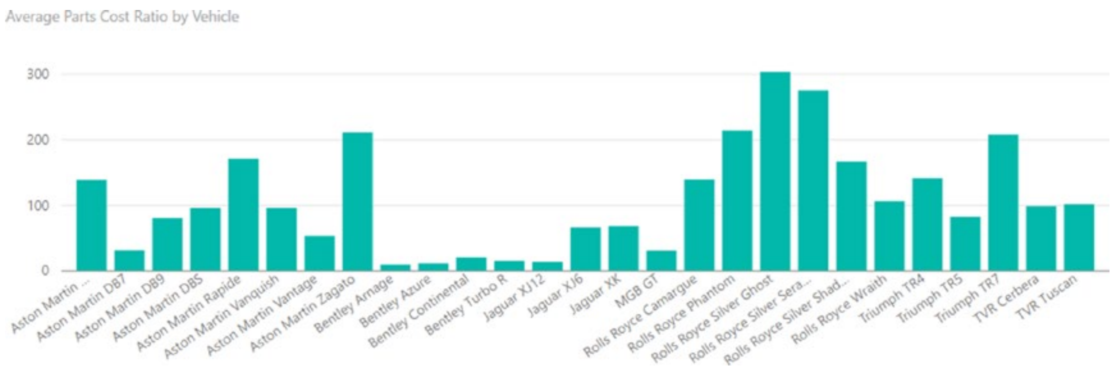


Figure 12-8. *Using the AVERAGEX() function in DAX*

As you can see from this example—and unlike the `AVERAGE()` function—`AVERAGEX()` takes two inputs (or parameters as they are technically known):

- The *table* to which the formula is applied
- The *formula* to use, which is just as you would apply it to a calculated column

This formula deducts the CostPrice from the SalePrice for every row in the table, and then returns the average dependent on the filters and selections currently applied. This way, you always get the mathematically accurate result in your visualizations.

The following are the essential points to take away from this example:

- It is essential to wrap any field references in an aggregate function, such as `SUM()`, `AVERAGE()`, or `COUNT()`, for an aggregated result to work. This is because the calculation (depending on the filters used) is not applied to only one record, but potentially several records, so data must be aggregated. Hence, the use of the `SUM()` function in this example.
- You can, and indeed, must, nest calculations inside parentheses to force Power BI Desktop to calculate elements in the correct order. This functions exactly as it does in Excel, so I will not labor the point here.

There are many more of these "X" functions (which are generally known as *iterative functions*, as they iterate over an entire table) in Power BI. Table 12-2 outlines those that are currently available.

Table 12-2. *DAX Iterative Functions*

Formula	Description	Example
`MINX()`	Calculates a value for each row in a table and displays the minimum value.	`MINX(Stock, [PartsCost]+[LaborCost])`
`MAXX()`	Calculates a value for each row in a table and displays the maximum value.	`MAXX(Stock, [PartsCost]+[LaborCost])`
`SUMX()`	Calculates a value for each row in a table and displays the total.	`SUMX(Stock, [PartsCost]+[LaborCost])`
`AVERAGEX()`	Calculates a value for each row in a table and displays the average of these values.	`AVERAGEX(Stock, [PartsCost]+[LaborCost])`
`COUNTX()`	Calculates a value for each row in a table and counts the resulting rows, including non-blank results of the calculation.	`COUNTX(Stock, [PartsCost]+[LaborCost])`
`COUNTAX()`	Calculates a value for each row in a table and counts the resulting rows, not including non-blank results of the calculation.	`COUNTAX (Stock, [PartsCost]+[LaborCost])`
`GEOMEANX()`	Calculates a value for each row in a table and displays the geometric mean of these values.	`GEOMEANX(Stock, [PartsCost]+[LaborCost])`
`MEDIANX()`	Calculates a value for each row in a table and displays the median value of these values.	`MEDIANX(Stock, [PartsCost]+[LaborCost])`
`PERCENTILEX.EXC()`	Returns the percentile of a record relative to the dataset.	`PERCENTILEX.EXC(Stock, [PartsCost]+[LaborCost])`
`PERCENTILEX.INC()`	Returns the percentile of a record relative to the dataset.	`PERCENTILEX.INC(Stock, [PartsCost]+[LaborCost])`
`RANKX()`	Orders the rows by progressive rank.	`RANKX(ALL(Stock[Make]), SUMX (RELATEDTABLE(InvoiceLines), [SalePrice]))`
`STDEVX.P()`	Calculates a value for each row in a table and displays the standard deviation of the entire population of these values.	`STDEVX.P(Stock, [PartsCost]+[LaborCost])`
`STDEVX.S()`	Calculates a value for each row in a table and displays the standard deviation of a sample population of these values.	`STDEVX.S(Stock, [PartsCost]+[LaborCost])`
`VARX.S()`	Calculates a value for each row in a table and displays the variance of the entire population of these values.	`VARX.S(Stock, [PartsCost]+[LaborCost])`
`VARX.P()`	Calculates a value for each row in a table and displays the variance of the entire population of these values.	`VARX.P(Stock, [PartsCost]+[LaborCost])`
`PRODUCTX()`	Returns the product of an expression evaluated for each row in a table.	`AVERAGEX(Stock, [PartsCost], 0.1)`

Filter Context

When working with DAX (at least if you want to develop any complex formulas), you need to understand *filter context*. This is the basis of the dynamic data analysis using Power BI Desktop. It is the basis of the approach where the results of a formula can change to reflect the current row or cell selection and any related filters.

Filter context can become an extremely complicated subject. However, since this is not a book on DAX, I am deliberately simplifying some of the ideas explained later in this chapter. After all, the aim is to get you started with DAX, not to scare you off right at the start.

The following are three key elements that you need to understand:

- Row context
- Query context
- Filter context

Let's take a brief look at these in turn.

Row Context

Row context is essentially the values from the current row. You saw this when creating new columns. This means that any fields that you used in a calculation always used other fields from the same record—or from a linked table. Therefore, this is largely automatic and typically handled by DAX without any intervention on your part.

Query Context

Query context is the combination of the following factors that produce a calculated result:

- Report-level filters (including roles)
- Page-level filters
- Visualization-level filters
- Slicers
- Interactive selection
- Row and column filters

The first three are cumulative (report-level filters, page-level filters, and visualization-level filters), and reduce the available data that a visualization can show. They are explained in detail in Chapter 20. For the moment, just consider them as a set of filters that only allows certain data to be used.

Slicers are any interactive filtering that you add for a dashboard. These restrict even further the dataset resulting from any report-level filters, page-level filters, and visualization-level filters to reduce even further the data that can be displayed. Interactive selection is a method of filtering data by selecting an element in another visualization. Both of these techniques are explained in detail in Chapter 21.

Row and column filters are best thought of as the row and column headers in a pivot table (or a cross-tab, if you prefer). These define the intersections of data that can be shown.

Query context is the cumulative effect of any filters that you apply when creating Power BI reports using the Power BI Desktop interface.

Filter Context

Filter context is added when you specify filter constraints on the set of values allowed in a column or table, by using formulas that expand or reduce the dataset that is used to obtain a result. Filter context applies on top of other contexts, such as row context or query context. This is the focus of the next few pages.

Filtering Data in Measures

Inevitably, there will be times when the filters that you apply using the Power BI Desktop user interface (the query-level filters that were described earlier) are not quite what you are looking for. There could be several reasons for this, including the following:

- You want to apply a highly specific filter to a single metric.
- You want to override the natural result of the query-level filter.
- You are creating a highly complex formula and it has to be tailored to a specific use.

Any of these reasons (and there are many others that you will discover as you progress with DAX) could require you to filter the data in a measure. Let's look at a few circumstances where this could prove necessary. Given the wide-ranging possibilities of DAX, I do not intend to explain anything more than the basics of DAX filtering using a few simple examples that I hope you find practical when building your own dashboards.

Simple Filters

There will probably be many occasions in your career when you need to home in on a specific subset of data. Maybe you need to compare and contrast one sales stream with another. Perhaps you need to highlight one cost compared to a total. Whatever the actual requirement, you need to apply a specific filter to a metric.

There are dozens—if not hundreds—of ways of applying different filters when building measures. However, one function is definitely an essential part of your DAX toolbox: the `CALCULATE()` function. This function lets you apply a range of filters to a measure that you can then apply in the visualizations that you build into your dashboards.

Text Filters

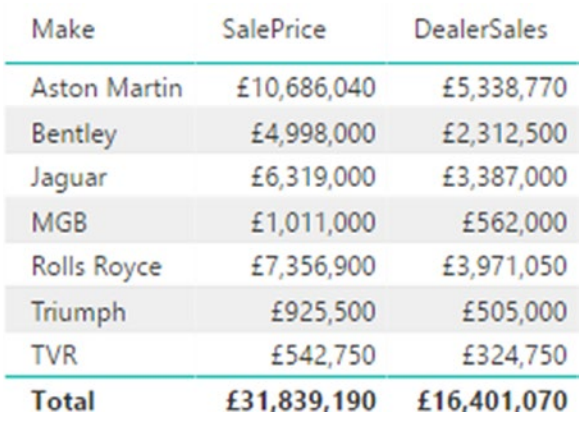
To begin, let's look at a fairly simple filter requirement. Brilliant British Cars sells to two types of clients: dealers and wholesalers. As part of your ongoing sales analysis, you want to isolate the dealer sales stream. The following explains how you can do this. Take a look at step 12 if all you want is the final formula.

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **DealerSales**.
4. To the right of the equals sign, enter (or select) **CALCULATE(**.
5. Enter (or select) **SUM(**.
6. Select the InvoiceLines[SalePrice] field.
7. Add a right parenthesis. This will terminate the `SUM()` function.

8. Enter a comma. This tells the `CALCULATE()` function that you are about to add the filters.
9. Select the Clients[ClientType] field.
10. Enter an equals sign.
11. Add the word **"Dealer"** (include the double quotes).
12. Add a right parenthesis. This will terminate the `CALCULATE()` function. The formula should now read as follows:

```
DealerSales = CALCULATE(SUM(InvoiceLines[SalePrice]),Clients[ClientType]="Dealer")
```

You could then add this new measure to a simple table of sales by make. If you do, you will see something like Figure 12-9.



Make	SalePrice	DealerSales
Aston Martin	£10,686,040	£5,338,770
Bentley	£4,998,000	£2,312,500
Jaguar	£6,319,000	£3,387,000
MGB	£1,011,000	£562,000
Rolls Royce	£7,356,900	£3,971,050
Triumph	£925,500	£505,000
TVR	£542,750	£324,750
Total	**£31,839,190**	**£16,401,070**

Figure 12-9. *Using a simple filter*

You can see from this table that the SalePrice column is not filtered in any way. However, the DealerSales column always shows a smaller figure for the sales per make, as it is displaying only the filtered subset of data that you requested. The new measure that you created can be applied to any visualization and can be filtered, sliced, and diced like any other data column, calculated column, or metric.

Now let me explain. Here we are using a function in DAX called `CALCULATE()`. This function does what its name implies: it calculates an aggregation. However, the calculation is nearly always a *filter* operation. This is because of the way in which its two parameters work:

- The first parameter defines the *function* to use (`SUM()` and `AVERAGE()` here), and the table and column that is aggregated; it could have been potentially a much more complex formula.
- The second parameter is a *filter* to force DAX to show only a subset of the data. In this specific case, it returns the sum of sales *only* when the client is a car dealership.

The filter that is applied here comes from another column. Indeed, it comes from another table altogether. When using the `CALCULATE()` function, you can use just about any column (either an original data column or a calculated column) as the source for a filter.

■ **Note** When you are filtering on a text (such as the type of dealer in this example), you *must* always enclose the text that you are searching for in double quotes.

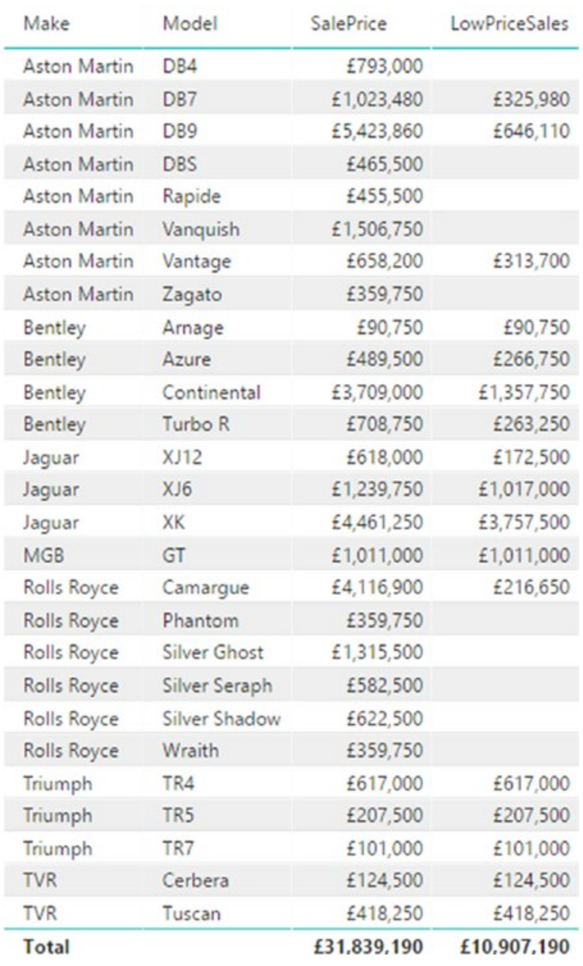
Numeric Filters

You are not restricted to filtering on text-based data only in Power BI Desktop. You can also subset data by numeric values. As an example, suppose that you want to see totals for sales for lower-priced models so that you can target the higher end of the market. The following explains how you could do this. Take a glance at step 12 if all you want is the final formula.

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **LowPriceSales**.
4. To the right of the equals sign, enter (or select) **CALCULATE(**.
5. Enter (or select) **SUM(**.
6. Select the InvoiceLines[SalePrice] field.
7. Add a right parenthesis. This will terminate the `SUM()` function.
8. Enter a comma. This tells the `CALCULATE()` function that you are about to add the filters.
9. Select the InvoiceLines[SalePrice] field again.
10. Enter a less-than sign: <.
11. Enter **50000**.
12. Add a right parenthesis. This will terminate the `CALCULATE()` function. The formula should now read

    ```
    LowPriceSales = CALCULATE(SUM(InvoiceLines[SalePrice]),
    InvoiceLines[SalePrice] < 50000)
    ```

Comparing the low-price sales with the unfiltered sales per 2015 make and model produces a table like the one in Figure 12-10.



Make	Model	SalePrice	LowPriceSales
Aston Martin	DB4	£793,000	
Aston Martin	DB7	£1,023,480	£325,980
Aston Martin	DB9	£5,423,860	£646,110
Aston Martin	DBS	£465,500	
Aston Martin	Rapide	£455,500	
Aston Martin	Vanquish	£1,506,750	
Aston Martin	Vantage	£658,200	£313,700
Aston Martin	Zagato	£359,750	
Bentley	Arnage	£90,750	£90,750
Bentley	Azure	£489,500	£266,750
Bentley	Continental	£3,709,000	£1,357,750
Bentley	Turbo R	£708,750	£263,250
Jaguar	XJ12	£618,000	£172,500
Jaguar	XJ6	£1,239,750	£1,017,000
Jaguar	XK	£4,461,250	£3,757,500
MGB	GT	£1,011,000	£1,011,000
Rolls Royce	Camargue	£4,116,900	£216,650
Rolls Royce	Phantom	£359,750	
Rolls Royce	Silver Ghost	£1,315,500	
Rolls Royce	Silver Seraph	£582,500	
Rolls Royce	Silver Shadow	£622,500	
Rolls Royce	Wraith	£359,750	
Triumph	TR4	£617,000	£617,000
Triumph	TR5	£207,500	£207,500
Triumph	TR7	£101,000	£101,000
TVR	Cerbera	£124,500	£124,500
TVR	Tuscan	£418,250	£418,250
Total		**£31,839,190**	**£10,907,190**

Figure 12-10. *Using a numeric filter*

In this simple example you saw how to use the less than (<) comparison operator. You can use any of the standard logical comparison operators (=, <, >, <=, >=, <>) that you saw in the previous chapter.

■ **Note** When you are filtering on a number, you must *not* enclose the number that you are searching for in quotes. Neither must you format the number in any way.

More Complex Filters

In the two previous examples, you saw the basics of creating filtered measures using either a text or a number to subset the data returned by the `CALCULATE()` function. In the real world of data analysis, filters can get a lot more complex. Indeed, allowing you to create specific and complex filters for metrics is one of the ways that DAX can help you tease out real insight from your data. So without attempting to get overly complicated, next are a few examples of the ways that you can define more complex filtered metrics in your data models.

Multiple Criteria in Filters

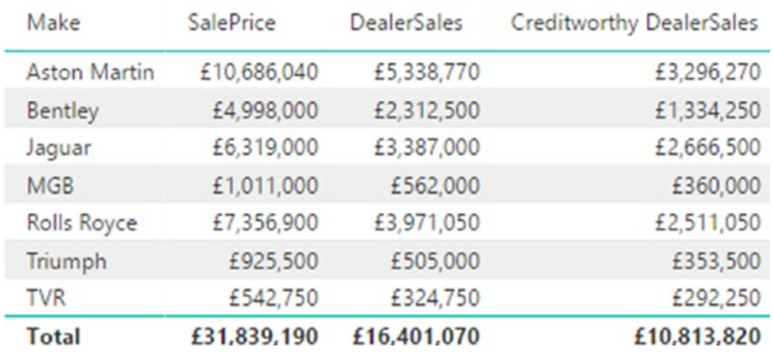
The `CALCULATE()` function is not limited to a single filter. Far from it. You can add multiple filters to the second part of this function, each separated by a comma.

For example, imagine that you not only want to see dealer sales when you look at your data, but also want to see the (slightly lower) figure for dealer sales where the client has a good credit status. This means combining two filter criteria.

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **Creditworthy DealerSales**.
4. To the right of the equals sign, enter (or select) **CALCULATE(**.
5. Enter (or select) **SUM(**.
6. Select the InvoiceLines[SalePrice] field.
7. Add a right parenthesis. This will terminate the `SUM()` function.
8. Enter a comma. This tells the `CALCULATE()` function that you are about to add the filters.
9. Select the Clients[ClientType] field.
10. Enter an equals sign.
11. Add the word **“Dealer”** (include the double quotes).
12. Enter a comma. This indicates that you are adding another filter criterion.
13. Select the Clients[IsCreditWorthy] field.
14. Enter an equals sign.
15. Add the word **TRUE()**. This is a logical value; it does not need to be enclosed in quotes and it has an empty parenthesis added.
16. Add a right parenthesis. This will terminate the `CALCULATE()` function. The formula should now read

    ```
    Creditworthy DealerSales = CALCULATE(SUM(InvoiceLines[SalePrice]),
    Clients[ClientType]="Dealer",Clients[IsCreditWorthy]=TRUE())
    ```

If this column is added to the table that you saw in Figure **12-9**, you will see a result something like the one in Figure 12-11.



Make	SalePrice	DealerSales	Creditworthy DealerSales
Aston Martin	£10,686,040	£5,338,770	£3,296,270
Bentley	£4,998,000	£2,312,500	£1,334,250
Jaguar	£6,319,000	£3,387,000	£2,666,500
MGB	£1,011,000	£562,000	£360,000
Rolls Royce	£7,356,900	£3,971,050	£2,511,050
Triumph	£925,500	£505,000	£353,500
TVR	£542,750	£324,750	£292,250
Total	**£31,839,190**	**£16,401,070**	**£10,813,820**

Figure 12-11. *Using a more complex filter*

If anything, this was a simple example. The filters that you add to any measure that uses the `CALCULATE()` function can contain multiple elements. Also, you can create a series of filters where each filter element compares data from different tables and mixes both text-based and numeric filters.

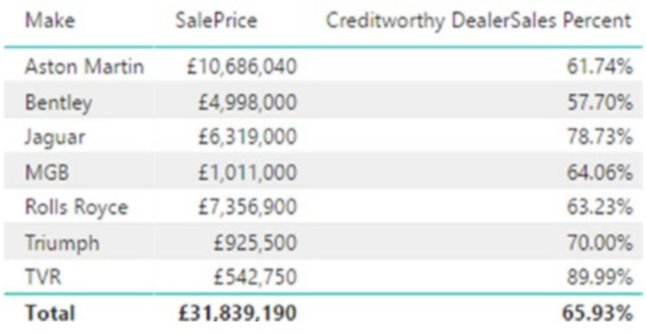
Using Multiple Filters

For a final filter example, imagine that you want to isolate the percentage of creditworthy dealer sales relative to dealer sales. You can do this by using the two measure calculations (DealerSales and CreditworthyDealerSales) in a single measure to obtain the desired result.

Since you just saw how to create these calculations, I will only show you the formula here:

```
Creditworthy DealerSales Percent = CALCULATE(SUM(InvoiceLines[SalePrice]),
Clients[ClientType]="Dealer",Clients[IsCreditWorthy]=TRUE()) /
CALCULATE(SUM(InvoiceLines[SalePrice]),Clients[ClientType]="Dealer")
```

As you can see (as you would in Excel), you can combine functions—even complex filtered functions—in a single metric to deliver powerful analysis. Using this measure in a simple table displaying sales by make produces the results shown in Figure 12-12.



Make	SalePrice	Creditworthy DealerSales Percent
Aston Martin	£10,686,040	61.74%
Bentley	£4,998,000	57.70%
Jaguar	£6,319,000	78.73%
MGB	£1,011,000	64.06%
Rolls Royce	£7,356,900	63.23%
Triumph	£925,500	70.00%
TVR	£542,750	89.99%
Total	**£31,839,190**	**65.93%**

Figure 12-12. *Using multiple filters in a measure*

Now that you have learned how to create filtered measures, you have mastered the building blocks of an extremely powerful technique that you can adapt and extend in your own data models.

Calculating Percentages of Totals

The filters that you have applied up until now in this chapter merely delivered subsets of data. Sometimes you need filters to do the opposite, and apply a calculation to an entire dataset. In other words, you need filters that *remove* filters. This is often because calculating a total means telling DAX to aggregate a column without applying any of the filtering by row that would normally be applied. In other words, you need to *prevent* the automatic filters that have proved so useful thus far.

A Simple Percentage

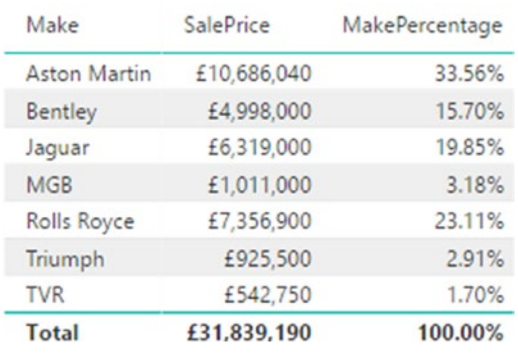
Imagine a table where you want to calculate the percentage of a total that each row represents. This could be the total of sales by make, for instance. Here you need to simply divide the sales by the total sales. The complete formula is in step 18, should you need it.

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **MakePercentage**.
4. To the right of the equals sign, enter (or select) **DIVIDE(**.
5. Enter (or select) **SUM(**.
6. Select the InvoiceLines[SalePrice] field.
7. Add a right parenthesis. This will terminate the `SUM()` function.
8. Enter a comma.
9. Enter (or select) **CALCULATE(**.
10. Enter (or select) **SUM(**.
11. Select the InvoiceLines[SalePrice] field.
12. Add a right parenthesis. This will terminate the `SUM()` function.
13. Enter a comma. This tells the `CALCULATE()` function that you are about to add the filters.
14. Enter (or select) **ALL(**.
15. Select the Stock[Make] field.
16. Add a right parenthesis. This will terminate the `ALL()` function.
17. Add a right parenthesis. This will terminate the `CALCULATE()` function.
18. Add a right parenthesis. This will terminate the `DIVIDE()` function. The formula should now read as follows:

```
MakePercentage = DIVIDE(SUM(InvoiceLines[SalePrice]), CALCULATE(SUM
(InvoiceLines[SalePrice]), ALL(Stock[Make])))
```

19. Format the new column as a percentage.

If you create a simple table of sales per make and add this new measure, it should look like Figure 12-13.

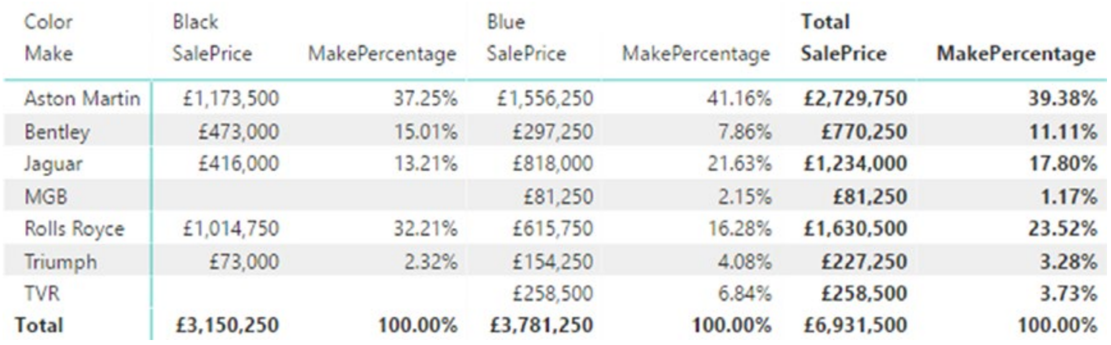


Make	SalePrice	MakePercentage
Aston Martin	£10,686,040	33.56%
Bentley	£4,998,000	15.70%
Jaguar	£6,319,000	19.85%
MGB	£1,011,000	3.18%
Rolls Royce	£7,356,900	23.11%
Triumph	£925,500	2.91%
TVR	£542,750	1.70%
Total	**£31,839,190**	**100.00%**

Figure 12-13. *Using the ALL() function to calculate a percentage per attribute*

This formula and the concept behind it probably seem a little peculiar. So let me explain the `ALL()` function in greater detail. In essence, the `ALL()` function says, "Remove all the filters concerning any specified fields." Consequently, in this example, the make is *not* filtered when calculating the total sales. This means that the unfiltered total can now be calculated—and so can the percentage of each make relative to this grand total.

It is important to note that the `ALL()` function only removes filters for the fields that you have specified. For instance, look at Figure 12-14, which shows a matrix for the sales and the percentage by make for two colors.



Color	Black		Blue		**Total**	
Make	SalePrice	MakePercentage	SalePrice	MakePercentage	**SalePrice**	**MakePercentage**
Aston Martin	£1,173,500	37.25%	£1,556,250	41.16%	**£2,729,750**	**39.38%**
Bentley	£473,000	15.01%	£297,250	7.86%	**£770,250**	**11.11%**
Jaguar	£416,000	13.21%	£818,000	21.63%	**£1,234,000**	**17.80%**
MGB			£81,250	2.15%	**£81,250**	**1.17%**
Rolls Royce	£1,014,750	32.21%	£615,750	16.28%	**£1,630,500**	**23.52%**
Triumph	£73,000	2.32%	£154,250	4.08%	**£227,250**	**3.28%**
TVR			£258,500	6.84%	**£258,500**	**3.73%**
Total	**£3,150,250**	**100.00%**	**£3,781,250**	**100.00%**	**£6,931,500**	**100.00%**

Figure 12-14. *The ALL() function lets all other filters be applied*

You can see here that the measure MakePercentage is correctly applied independently from sales for 2014, 2015, and the grand total. This is because all other filters (the year in this case) are applied as you have come to expect with Power BI Desktop; *only* the make is not filtered when calculating the total sales.

Removing Multiple Filter Elements

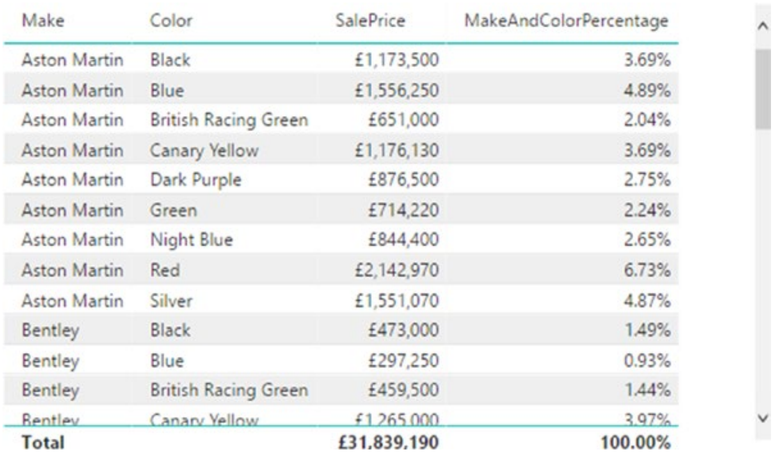
If your visualization is more complex than the simple example that you just saw, then you have to craft your measures appropriately to handle any complexity. For instance, take the case where you want to see sales by make and color, and display the percentage of each row compared to the total. You need to calculate a total that discards the filters for make *and* color so that DAX can arrive at the correct figure for the overall total. Here is the formula that can do this:

```
MakeAndColorPercentage = DIVIDE(SUM(InvoiceLines[SalePrice]), CALCULATE(SUM(InvoiceLines
[SalePrice]), ALL(Stock[Make]), ALL(Colors[Color])))
```

Since this code snippet is only an extension of the previous one, I have not explained how to construct it in detail. All you have to do is follow the steps from the previous example and add a second filter to the second `CALCULATE()` function (the one that returns the grand total of sales). As you saw earlier in this chapter, the `CALCULATE()` function can take multiple filter parameters. It follows that it can also take multiple "unfilter" parameters, as it is doing here. The key is in the following part of the measure:

```
ALL(Stock[Make]), ALL(Colors[Color])
```

This piece of DAX is simply saying, "Don't apply any make or color filters when calculating." The consequence is that this measure now calculates the total sales whatever the make or color. This then becomes the basis for the percentage calculation, as you can see in Figure 12-15.



Make	Color	SalePrice	MakeAndColorPercentage
Aston Martin	Black	£1,173,500	3.69%
Aston Martin	Blue	£1,556,250	4.89%
Aston Martin	British Racing Green	£651,000	2.04%
Aston Martin	Canary Yellow	£1,176,130	3.69%
Aston Martin	Dark Purple	£876,500	2.75%
Aston Martin	Green	£714,220	2.24%
Aston Martin	Night Blue	£844,400	2.65%
Aston Martin	Red	£2,142,970	6.73%
Aston Martin	Silver	£1,551,070	4.87%
Bentley	Black	£473,000	1.49%
Bentley	Blue	£297,250	0.93%
Bentley	British Racing Green	£459,500	1.44%
Bentley	Canary Yellow	£1,265,000	3.97%
Total		**£31,839,190**	**100.00%**

Figure 12-15. *Using the ALL() function to calculate a percentage per attribute*

Admittedly, preparing highly specific measures like those that you have seen in the last few pages can take a few minutes. Clearly, these measures are tightly linked to the data that you are displaying in the visuals that use them. However, the ability to compose highly focused metrics like these is often key to using your dashboards to highlight the insights that you want to deliver.

Visual Totals

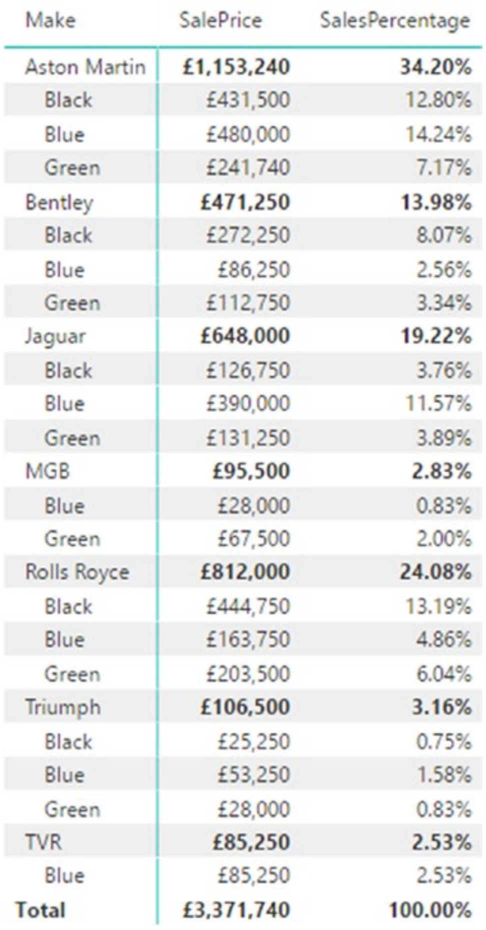
Users (meaning the target audience for your reports and dashboards) do not like anomalies or apparent contradictions. So you have to be sure that the data that they see is visually coherent. This is especially true when displaying tables and matrices with subtotals and grand totals where you want percentage totals to reflect the figures shown and not include any records that have been removed by the filter.

One technique that can help you here is the `ALLSELECTED()` function. This only applies any filters that have been added (either at the report, page, or visualization level, or as slicers or cross-filters from other visuals) *without* your having to specify the fields that you do not want to filter, as you did in the previous examples.

SalesPercentage is a measure that uses the `ALLSELECTED()` function as the filter for the `CALCULATE()` function that returns the total much like you did earlier:

```
SalesPercentage = DIVIDE(SUM(InvoiceLines[SalePrice]), CALCULATE(SUM(InvoiceLines
[SalePrice]), ALLSELECTED()))
```

If you look at Figure 12-16, you see that the subtotals for the sales percentage (and indeed the grand total) are accurate, despite the fact that the country (USA) is selected in a slicer and three colors in a filter.



Make	SalePrice	SalesPercentage
Aston Martin	**£1,153,240**	**34.20%**
Black	£431,500	12.80%
Blue	£480,000	14.24%
Green	£241,740	7.17%
Bentley	**£471,250**	**13.98%**
Black	£272,250	8.07%
Blue	£86,250	2.56%
Green	£112,750	3.34%
Jaguar	**£648,000**	**19.22%**
Black	£126,750	3.76%
Blue	£390,000	11.57%
Green	£131,250	3.89%
MGB	**£95,500**	**2.83%**
Blue	£28,000	0.83%
Green	£67,500	2.00%
Rolls Royce	**£812,000**	**24.08%**
Black	£444,750	13.19%
Blue	£163,750	4.86%
Green	£203,500	6.04%
Triumph	**£106,500**	**3.16%**
Black	£25,250	0.75%
Blue	£53,250	1.58%
Green	£28,000	0.83%
TVR	**£85,250**	**2.53%**
Blue	£85,250	2.53%
Total	**£3,371,740**	**100.00%**

Figure 12-16. *Using the ALLSELECTED() function to calculate a percentage per attribute per group*

The ALLSELECTED() function says to DAX, "Don't filter on any filters applied by the user, whatever the technique used to apply them." This can make creating percentage totals much easier, as this function removes the need to create highly specific measures that are tied to specific levels of totals and subtototals.

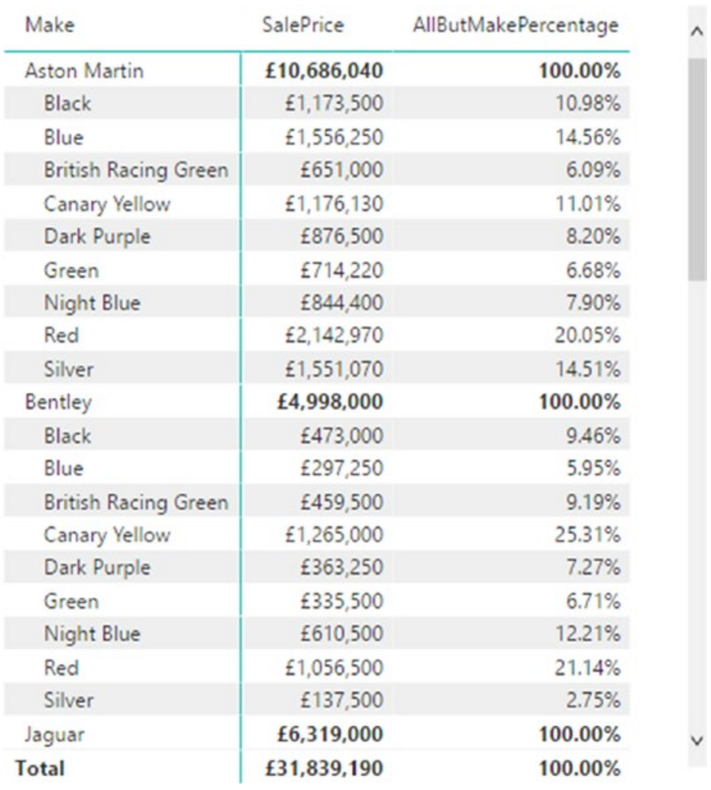
The ALLEXCEPT() Function

In practice, you could find yourself having to write extremely targeted measures that need to remove filters from all the elements in a calculation except one or two. So, to save you from having to write long lists of ALL() functions, you can say "All but" a field using the ALLEXCEPT() function.

As an example of this (although it is extremely simple), suppose that you want to see the percentages of sales grouped by a subclassification. You know that you want to have Make as the main grouping element, but then you might want to use Color, Client, or even Model as the subgroup. So to save you from having to write a measure specifically for each of these combinations, you can write the following:

```
AllButMakePercentage = DIVIDE(SUM(InvoiceLines[SalePrice]), CALCULATE(SUM(InvoiceLines
[SalePrice]), ALLEXCEPT(Stock, Stock[Make])))
```

If you use this measure in a matrix where Make is the leftmost column, you can then add subgroups using any other field to get the kind of output that is shown in Figure 12-17.



Make	SalePrice	AllButMakePercentage
Aston Martin	**£10,686,040**	**100.00%**
Black	£1,173,500	10.98%
Blue	£1,556,250	14.56%
British Racing Green	£651,000	6.09%
Canary Yellow	£1,176,130	11.01%
Dark Purple	£876,500	8.20%
Green	£714,220	6.68%
Night Blue	£844,400	7.90%
Red	£2,142,970	20.05%
Silver	£1,551,070	14.51%
Bentley	**£4,998,000**	**100.00%**
Black	£473,000	9.46%
Blue	£297,250	5.95%
British Racing Green	£459,500	9.19%
Canary Yellow	£1,265,000	25.31%
Dark Purple	£363,250	7.27%
Green	£335,500	6.71%
Night Blue	£610,500	12.21%
Red	£1,056,500	21.14%
Silver	£137,500	2.75%
Jaguar	**£6,319,000**	**100.00%**
Total	**£31,839,190**	**100.00%**

Figure 12-17. *Using the ALLEXCEPT() function to calculate a percentage per attribute per group*

In this example, other filters (color here) are applied, but not make. So you are displaying the percentage for each color compared to the aggregate total for the make.

■ **Note** `ALLEXCEPT()` does what it says and removes all filters except the one that you specify. This can have the effect of preventing other filters from working as you expect.

Filtering on Measures

The `CALCULATE()` function is without a doubt one of the most powerful functions that you will use in DAX. However, there are a few things that it cannot do. One of these is to filter data by comparing to a *measure* rather than to a column. If you cast your mind back to the examples where `CALCULATE()` was applied, you will remember that a data column or a calculated column was used every time that a comparison (text-based or numeric) was invoked. Indeed, if you try to use `CALCULATE()` with a measure rather than a column, you will get an error.

Fortunately, DAX has a solution to this conundrum, which is to use the `FILTER()` function. You may well wonder what the differences are between `FILTER()` and `CALCULATE()`. Well, at its simplest, `FILTER()` can use *measures* as part of a comparison, whereas `CALCULATE()` must use columns—or calculated columns. Also, `FILTER()` *must* use an iterator function (such as `SUMX()`) rather than a simple aggregation function to produce a correct result.

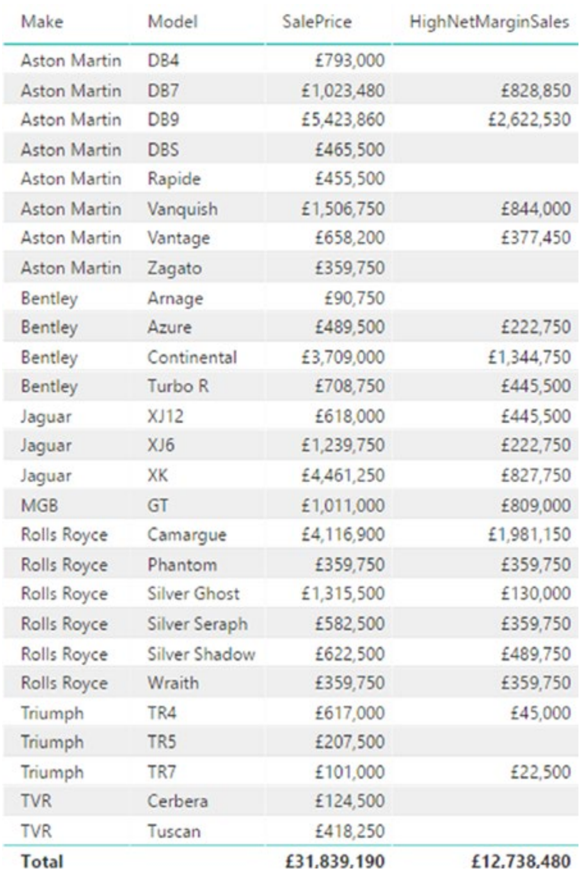
Let's see this in action. Suppose that you want to isolate sales where the ratio of net margin is over 50%. Fortunately, you have a measure—RatioNetMargin—that calculates the percentage. The following explains how you can use this measure in a filter so that you can display these lucrative sales. The complete formula is in step 16.

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **HighNetMarginSales**.
4. To the right of the equals sign, enter (or select) **CALCULATE(**.
5. Enter (or select) **SUM(**.
6. Select the InvoiceLines[SalePrice] field.
7. Add a right parenthesis. This will terminate the `SUM()` function.
8. Enter a comma. This tells the `CALCULATE()` function that you are about to add the filters.
9. Enter (or select) **FILTER(**.
10. Select the InvoiceLines table. This is the table to filter.
11. Enter a comma. This tells the `FILTER()` function that you are about to enter the filter criteria.
12. Enter (or select) the column **[RatioNetMargin]**.
13. Enter the greater than symbol: >.
14. Enter the figure **0.5**.
15. Add a right parenthesis. This will terminate the `FILTER()` function.

16. Add a right parenthesis. This will terminate the `CALCULATE()` function. The formula should now read

```
HighNetMarginSales = CALCULATE(SUM(InvoiceLines[SalePrice]),FILTER
(InvoiceLines, [RatioNetMargin]>0.5))
```

If you use this measure in a table of sales by make and model, you should see something like Figure 12-18.



Make	Model	SalePrice	HighNetMarginSales
Aston Martin	DB4	£793,000	
Aston Martin	DB7	£1,023,480	£828,850
Aston Martin	DB9	£5,423,860	£2,622,530
Aston Martin	DBS	£465,500	
Aston Martin	Rapide	£455,500	
Aston Martin	Vanquish	£1,506,750	£844,000
Aston Martin	Vantage	£658,200	£377,450
Aston Martin	Zagato	£359,750	
Bentley	Arnage	£90,750	
Bentley	Azure	£489,500	£222,750
Bentley	Continental	£3,709,000	£1,344,750
Bentley	Turbo R	£708,750	£445,500
Jaguar	XJ12	£618,000	£445,500
Jaguar	XJ6	£1,239,750	£222,750
Jaguar	XK	£4,461,250	£827,750
MGB	GT	£1,011,000	£809,000
Rolls Royce	Camargue	£4,116,900	£1,981,150
Rolls Royce	Phantom	£359,750	£359,750
Rolls Royce	Silver Ghost	£1,315,500	£130,000
Rolls Royce	Silver Seraph	£582,500	£359,750
Rolls Royce	Silver Shadow	£622,500	£489,750
Rolls Royce	Wraith	£359,750	£359,750
Triumph	TR4	£617,000	£45,000
Triumph	TR5	£207,500	
Triumph	TR7	£101,000	£22,500
TVR	Cerbera	£124,500	
TVR	Tuscan	£418,250	
Total		**£31.839.190**	**£12.738.480**

Figure 12-18. *Applying a filter to a measure*

In this example, you filtered data on a measure (RatioNetMargin) rather than a column. Be aware, however, that the `FILTER()` function can be slow when applied to large datasets.

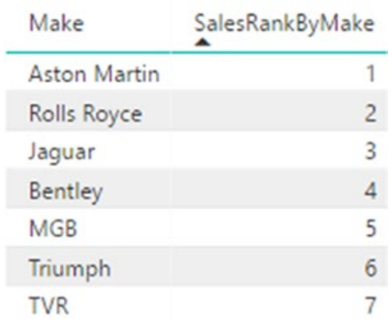
Displaying Rank

DAX can do so much when it comes to preparing metrics for BI delivery that it is hard to know exactly what you need and when. The final example in this short tour of DAX measures explains how to rank sales by make. I realize that you can do this just by sorting records, but should you need a clear and unequivocal indicator of ranking, then here is how it can be done-and the formula is directly available in step 16:

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **SalesRankByMake**.
4. To the right of the equals sign, enter (or select) **RANKX(**.
5. Enter (or select) **ALL(**.
6. Select the Stock[Make] field.
7. Add a right parenthesis. This will terminate the `ALL()` function.
8. Enter a comma. This tells the `RANK()` function that you are going to enter the calculation of how to order the data.
9. Enter (or select) **SUMX(**.
10. Enter (or select) **RELATEDTABLE(**.
11. Select the InvoiceLines table. This is the table where the data is to be sourced.
12. Add a right parenthesis. This will terminate the `RELATEDTABLE()` function.
13. Enter a comma. This tells the `RELATEDTABLE()` function that you are about to enter the field to use.
14. Enter (or select) **[SalePrice]**.
15. Add a right parenthesis. This will terminate the `SUMX()` function.
16. Add a right parenthesis. This will terminate the `RANKX()` function. The formula should now read

```
SalesRankByMake = RANKX(ALL(Stock[Make]),SUMX(RELATEDTABLE (InvoiceLines), [SalePrice]))
```

If you apply this measure to a simple table that lists the makes sold (in 2014, for instance) and then sort by the SalesRankByMake field, you will see something like Figure 12-19.



Make	SalesRankByMake
Aston Martin	1
Rolls Royce	2
Jaguar	3
Bentley	4
MGB	5
Triumph	6
TVR	7

Figure 12-19. *Using the RANKX() function to classify data*

As its name implies, RANKX() ranks the first field using the order returned by the descending output of the second field.

A Few Comments and Notes on Using Measures

Measures are an immense subject. The breadth and depth of the calculations that can be delivered using DAX are little short of astounding. Consequently, it is impossible in an introductory chapter on measures to do anything other than give you a taste of what can be done and provide a few useful starter functions for you to adapt to your own requirements.

As you move on with DAX, a few things might help you on your way. The first concerns the use of calculated columns. Sometimes they are such an easy solution that it is a shame not to create them. However, they are stored in the table and do take up space. This means more space on disk and more space in memory. This is particularly true for a table containing tens of millions of rows. Measures, on the other hand, are only calculated at run time, and so they take up virtually no space. So, if you are considering creating many calculated columns, perhaps some of them could become measures instead.

Calculation Options

I imagine that you have not had to worry about recalculation of Power BI Desktop workbooks if you have been using relatively small datasets like the sample data for this book. If you are using vast amounts of data (after all, this is what Power BI Desktop was designed for), however, then recalculation could become a subject that you need to master.

By default, Power BI Desktop recalculates all calculated columns and measures when there is a change in the dataset. These are the main operations that can trigger a recalculation:

- Data from an external data source (of any kind) has been updated.
- Data from an external data source has been filtered.
- You have changed the name of a table or column.
- You have added, modified, or deleted relationships between tables.
- You have altered any formula for a calculated column or a measure.
- You have added new calculated columns or measures.

More generally, if you want to be sure that your data is up to date, you should probably update the data. You can do this by clicking the Refresh button in the Home ribbon.

Conclusion

In this chapter, you took a first look at one of the most powerful features in DAX: measures. These let you develop custom calculations for the Power BI Desktop data model. You then use these metrics in your visuals to deliver specific insights based on your data.

First, you saw how to apply iterator functions so that you can apply a calculation to a set of rows and return an aggregation, be it a sum, average, or any other available aggregate function. Then you saw how to apply specific filters to your calculations. Finally, you saw how to prevent filters from being applied so that you can display percentages and calculate advanced ratios.

This chapter was only a brief introduction to measures in DAX. Yet I hope that it has whetted your appetite and that you will now feel empowered to continue, and consequently to develop the analyses and metrics that you need for your own data.

It will soon be time to start applying these measures to a range of visualizations in Power BI Desktop. However, before then let's take a look at one final fundamental element of the data model that you will need to set up before you can extract real value from your data. The remaining piece of the puzzle is adding time intelligence, and this is the subject of the next chapter.

CHAPTER 13

Analyzing Data over Time

A considerable amount of data analysis—and probably most business intelligence—involves looking at how metrics evolve over time. You may need to aggregate sales by month, week, or year, for instance. Perhaps you want to compare figures for a previous month, quarter, or year with the figures for a current period. Whatever the exact requirement, handling time (by which we nearly always mean dates) is essential in Power BI Desktop.

Initially, using time functions in Power BI Desktop may only be limited to extracting time intervals from the available data and grouping results by units of time, such as days, weeks, months, quarters, and years. As you will find out in the first part of this chapter, DAX makes this kind of analysis really simple.

However, Power BI Desktop can also add what is called *time intelligence* to data models. This massively useful capability can take your analysis to a fundamentally higher level. This approach involves adding a separate table (called a *date* or *time dimension*) to the data model and then adding DAX functions that enable you to see how data evolves over time. Consequently, this chapter also includes an introduction to time intelligence using a wide range of DAX formulas that are available to help you create time-based calculations quickly and easily.

In this chapter, we continue to develop the file that you extended in the two previous chapters. Should you need it, a complete version of this file (CarSalesDataWithNewColumnsAndMeasures.pbix) is available on the Apress web site, including the extensions added in the previous chapter, in the folder C:\PowerBiDesktopSamples\CH13.

Simple Date Calculations

Data analysis often involves looking at how key metrics evolve over time. Power BI Desktop can help you group and isolate date and time elements in your data. Since dates (and time) are often a continuous stream of dates in a dataset, it can be useful to isolate the years, months, weeks, and days in a table alongside the date that a row contains so that you can create tables or visuals that group and aggregate records into these more comprehensible "buckets." You can then display and compare data over years and months, for instance, in order to tease out the real insights that your raw data contains.

For the first example of how DAX can help you to categorize records using date-based criteria, let's imagine that you envisage creating a couple of charts. First, you need one that lets you track sales over the years that Brilliant British Cars has traded. Then you want a second graphic that looks at sales for each month over the years. Unfortunately (for the moment at least), your data model does not contain columns that show the year or the month of a sale, and Power BI Desktop does not let you create metrics as part of a visualization. You have to have the metric available in the data model if you plan to use it in a dashboard element. Moreover, as you will see in Chapters 14 through 22, it is best if you have all the metrics in place before you create any visualizations. Indeed, this is precisely the reason that you are learning how to extend the data model with new columns using DAX. The following explains how to create these two new columns in the Invoices table:

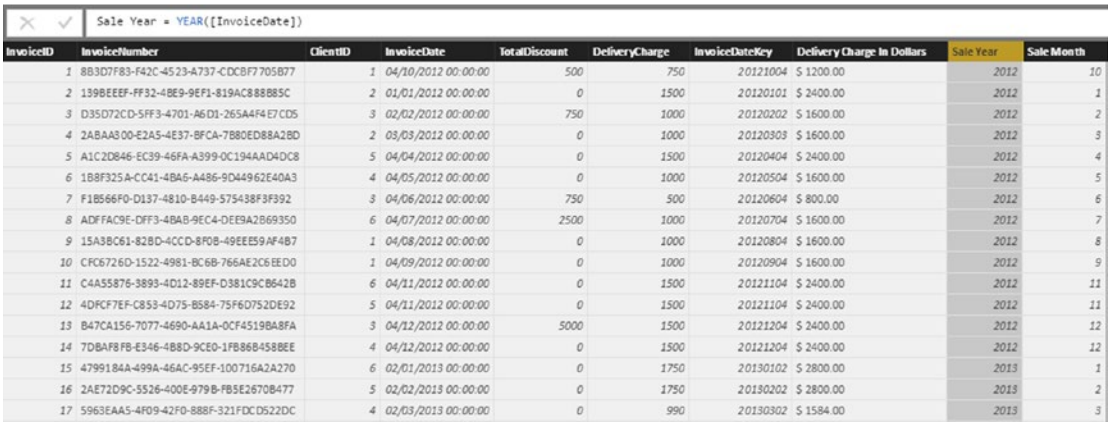
A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_13

1. Open the Power BI Desktop file C:\PowerBiDesktopSamples\CH13\CarSalesDataWithNewColumnsAndMeasures.pbix.
2. Click the Data icon.
3. Select the Invoices table in the Fields list.
4. In the Modeling ribbon, click the New Column button.
5. In the formula bar to the left of the equals sign, replace Column with **Sale Year**.
6. Click to the right of the equals sign.
7. Enter (or start typing) and then select YEAR(.
8. Enter a left square bracket: **[**.
9. Select the InvoiceDate field.
10. Add a final right parenthesis to complete the `YEAR()` function.
11. Confirm the formula by pressing Enter or clicking the tick icon in the formula bar. The new column will display the year of each sale.
12. Repeat steps 2 through 9, only use the DAX formula `MONTH()` instead of `YEAR()`, and name the new column Sale Month.

The formula for the two new columns will be

```
Sale Month = MONTH([InvoiceDate])
Sale Year = YEAR([InvoiceDate])
```

Figure 13-1 shows what the Invoices table looks like with these two new columns added.



```
Sale Year = YEAR([InvoiceDate])
```

InvoiceID	InvoiceNumber	ClientID	InvoiceDate	TotalDiscount	DeliveryCharge	InvoiceDateKey	Delivery Charge In Dollars	Sale Year	Sale Month
1	8B3D7F83-F42C-4523-A737-CDCBF7705B77	1	04/10/2012 00:00:00	500	750	20121004	$ 1200.00	2012	10
2	139BEEEF-FF32-4BE9-9EF1-819AC888B85C	2	01/01/2012 00:00:00	0	1500	20120101	$ 2400.00	2012	1
3	D35D72CD-5FF3-4701-A6D1-265A4F4E7CD5	3	02/02/2012 00:00:00	750	1000	20120202	$ 1600.00	2012	2
4	2ABAA300-E2A5-4E37-BFCA-7B80ED88A2BD	2	03/03/2012 00:00:00	0	1000	20120303	$ 1600.00	2012	3
5	A1C2D846-EC39-46FA-A399-0C194AAD4DC8	5	04/04/2012 00:00:00	0	1500	20120404	$ 2400.00	2012	4
6	1B8F325A-CC41-4BA6-A486-9D44962E40A3	4	04/05/2012 00:00:00	0	1000	20120504	$ 1600.00	2012	5
7	F1B566F0-D137-4810-B449-575438F3F392	3	04/06/2012 00:00:00	750	500	20120604	$ 800.00	2012	6
8	ADFFAC9E-DFF3-4BAB-9EC4-DEE9A2B69350	6	04/07/2012 00:00:00	2500	1000	20120704	$ 1600.00	2012	7
9	15A3BC61-828D-4CCD-8F0B-49EEE59AF4B7	1	04/08/2012 00:00:00	0	1000	20120804	$ 1600.00	2012	8
10	CFC6726D-1522-4981-BC6B-766AE2C6EED0	1	04/09/2012 00:00:00	0	1000	20120904	$ 1600.00	2012	9
11	C4A55876-3893-4D12-89EF-D381C9CB642B	6	04/11/2012 00:00:00	0	1500	20121104	$ 2400.00	2012	11
12	4DFCF7EF-C853-4D75-B584-75F6D752DE92	5	04/11/2012 00:00:00	0	1500	20121104	$ 2400.00	2012	11
13	B47CA156-7077-4690-AA1A-0CF4519BA8FA	3	04/12/2012 00:00:00	5000	1500	20121204	$ 2400.00	2012	12
14	7DBAF8FB-E346-4B8D-9CE0-1FB86B458BEE	4	04/12/2012 00:00:00	0	1500	20121204	$ 2400.00	2012	12
15	4799184A-499A-46AC-95EF-100716A2A270	6	02/01/2013 00:00:00	0	1750	20130102	$ 2800.00	2013	1
16	2AE72D9C-5526-400E-979B-FB5E2670B477	5	02/02/2013 00:00:00	0	1750	20130202	$ 2800.00	2013	2
17	5963EAA5-4F09-42F0-888F-321FDCD522DC	4	02/03/2013 00:00:00	0	990	20130302	$ 1584.00	2013	3

Figure 13-1. *The YEAR() and MONTH() DAX functions*

In Chapters 15 through 20, you see how to use metrics like these to create visualizations that explore the way that sales have evolved over time.

■ **Note** To extract part of a date like this, the column that you are using for the original data must be of the date data type or be capable of being interpreted as a date by Power BI Desktop.

This example illustrated two of the DAX date and time functions. Inevitably, there are many other functions that you can apply to extract a date or time element from a date field. Since they all follow the same principles as those that you have just seen (with the exception of the `NOW()` and `TODAY()` functions that are explained in a couple of pages' time), it is easier to list them in Table 13-1 rather than provide a set of nearly identical examples.

Table 13-1. *DAX Date and Time Functions*

Function	Description	Example
`YEAR()`	Extracts the year element from a date.	`YEAR([InvoiceDate])`
`MONTH()`	Extracts the month number from a date.	`MONTH([InvoiceDate])`
`DAY()`	Extracts the day number from a date.	`DAY([InvoiceDate])`
`WEEKDAY()`	Extracts the weekday from a date. Sunday is 1, Monday is 2, and so forth.	`WEEKDAY([InvoiceDate])`
`WEEKNUM()`	Extracts the number of the week in the year from a date.	`WEEKNUM([InvoiceDate])`
`HOUR()`	Extracts the hour from a time or datetime column.	`HOUR([InvoiceDate])`
`MINUTE()`	Extracts the minutes from a time or datetime column.	`MINUTE([InvoiceDate])`
`SECOND()`	Extracts the seconds from a time or datetime column.	`SECOND([InvoiceDate])`
`EOMONTH()`	Returns the last day of the month from a date.	`EOMONTH([InvoiceDate])`
`NOW()`	Returns the current date and time.	`NOW()`
`TODAY()`	Returns the current date.	`TODAY()`
`DATE()`	Lets you enter a date as year, month, and day.	`DATE(2015, 07, 25)`
`TIME()`	Lets you enter a time as hours and minutes.	`TIME(19, 57)`
`DATEDIFF()`	Calculates the difference between two dates and/or times expressed as a number of specified periods.	`DATEDIFF([InvoiceDate], DATE(2025, 07, 25), YEAR)`
`STARTOFMONTH()`	Selects the first day of the month for a date.	`STARTOFMONTH(Invoices[SaleDate])`
`STARTOFQUARTER()`	Selects the first day of the quarter for a date.	`STARTOFQUARTER(Invoices[SaleDate])`
`STARTOFYEAR()`	Selects the first day of the year for a date.	`STARTOFYEAR(Invoices[SaleDate])`

(*continued*)

Table 13-1. (*continued*)

Function	Description	Example
`ENDOFMONTH()`	Selects the last day of the month for a date.	`ENDOFMONTH(Invoices[SaleDate])`
`ENDOFQUARTER()`	Selects the last day of the quarter for a date.	`ENDOFQUARTER(Invoices[SaleDate])`
`ENDOFYEAR()`	Selects the last day of the year for a date.	`ENDOFYEAR(Invoices[SaleDate])`
`YEARFRAC()`	Calculates the fraction of the year represented by the number of whole days between two dates.	`YEARFRAC(Stock[PurchaseDate],Invoices[SaleDate])`

■ **Note** If you define a field as having a date or datetime data type, then Power BI Desktop always creates a hierarchy of Year ➤ Quarter ➤ Month ➤ Day in the visual that you are creating. This can become annoying when all you need is the day. So it is well worth creating a well-structured date dimension table, as described later in this chapter, so that you only use the exact date element that you want, rather than have Power BI Desktop make assumptions for you.

Date and Time Formatting

When dealing with dates and times in Power BI Desktop, you may not need to go to the lengths of extracting a part of a date field, but may simply need to display a date in a different way. Rather like Excel, DAX can help you to do this quickly and easily using the `FORMAT()` function.

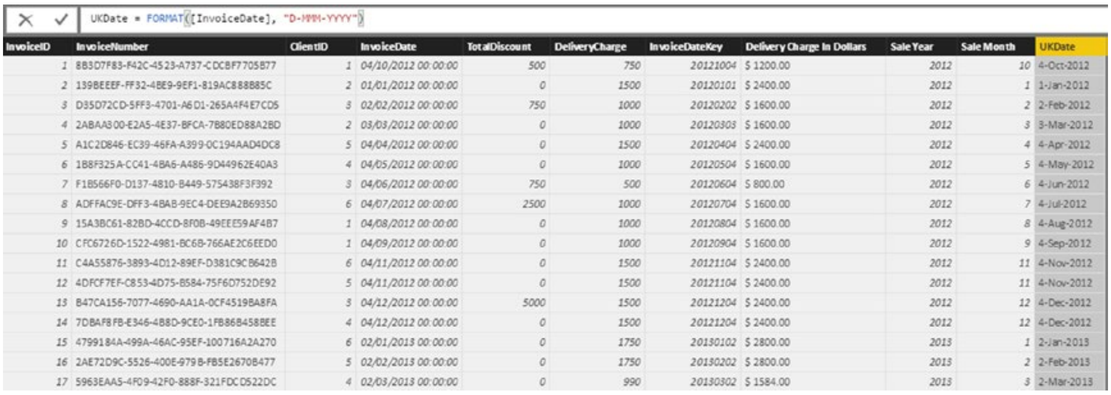
Suppose that you want to have the InvoiceDate field displayed in a specific date format for use in certain visualizations. The following explains how you can do this. The finished formula is in step 11 if you prefer to use this.

1. Select the Invoices table in the Fields list.
2. In the Modeling ribbon, click the New Column button.
3. In the formula bar to the left of the equals sign, replace Column with **UKDate**.
4. Click to the right of the equals sign.
5. Enter (or start typing and then select) **FORMAT(**.
6. Enter a left square bracket: **[**.
7. Select the InvoiceDate field.
8. Add a comma.
9. Enter the format code **"D-MMM-YYYY"** (include the double quotes).
10. Add a final right parenthesis to complete the `FORMAT()` function.
11. Replace the word Column with **UKDate**. The formula bar will display the following code:

```
UKDate = FORMAT([InvoiceDate], "D-MMM-YYYY")
```

12. Confirm the formula by pressing Enter or clicking the tick icon in the formula bar. The new column will display the sale date in a different format.

If you look at Figure 13-2, you see what the newly formatted invoice data field looks like in the new column.



```
UKDate = FORMAT([InvoiceDate], "D-MMM-YYYY")
```

InvoiceID	InvoiceNumber	ClientID	InvoiceDate	TotalDiscount	DeliveryCharge	InvoiceDateKey	Delivery Charge In Dollars	Sale Year	Sale Month	UKDate
1	8B3D7F83-F42C-4523-A737-CDCBF7705B77	1	04/10/2012 00:00:00	500	750	20121004	$ 1200.00	2012	10	4-Oct-2012
2	139BEEEF-FF32-4BE9-9EF1-819AC888B85C	2	01/01/2012 00:00:00	0	1500	20120101	$ 2400.00	2012	1	1-Jan-2012
3	D35D72CD-5FF3-4701-A6D1-265A4F4E7CD5	3	02/02/2012 00:00:00	750	1000	20120202	$ 1600.00	2012	2	2-Feb-2012
4	2ABAA300-E2A5-4E37-BFCA-7B80ED88A2BD	2	03/03/2012 00:00:00	0	1000	20120303	$ 1600.00	2012	3	3-Mar-2012
5	A1C2D846-EC39-46FA-A399-0C194AAD4DC8	5	04/04/2012 00:00:00	0	1500	20120404	$ 2400.00	2012	4	4-Apr-2012
6	1B8F325A-CC41-4BA6-A486-9D44962E40A3	4	04/05/2012 00:00:00	0	1000	20120504	$ 1600.00	2012	5	4-May-2012
7	F1B566F0-D137-4810-B449-575438F3F392	3	04/06/2012 00:00:00	750	500	20120604	$ 800.00	2012	6	4-Jun-2012
8	ADFFAC9E-DFF3-4BAB-9EC4-DEE9A2B69350	6	04/07/2012 00:00:00	2500	1000	20120704	$ 1600.00	2012	7	4-Jul-2012
9	15A3BC61-82BD-4CCD-8F0B-49EEE59AF4B7	1	04/08/2012 00:00:00	0	1000	20120804	$ 1600.00	2012	8	4-Aug-2012
10	CFC6726D-1522-4981-BC6B-766AE2C6EED0	1	04/09/2012 00:00:00	0	1000	20120904	$ 1600.00	2012	9	4-Sep-2012
11	C4A55876-3893-4D12-89EF-D381C9CB642B	6	04/11/2012 00:00:00	0	1500	20121104	$ 2400.00	2012	11	4-Nov-2012
12	4DFCF7EF-C853-4D75-B584-75F6D752DE92	5	04/11/2012 00:00:00	0	1500	20121104	$ 2400.00	2012	11	4-Nov-2012
13	B47CA156-7077-4690-AA1A-0CF4519BA8FA	3	04/12/2012 00:00:00	5000	1500	20121204	$ 2400.00	2012	12	4-Dec-2012
14	7DBAF8FB-E346-4B8D-9CE0-1FB86B458BEE	4	04/12/2012 00:00:00	0	1500	20121204	$ 2400.00	2012	12	4-Dec-2012
15	479918 4A-499A-46AC-95EF-100716A2A270	6	02/01/2013 00:00:00	0	1750	20130102	$ 2800.00	2013	1	2-Jan-2013
16	2AE72D9C-5526-400E-979B-FB5E2670B477	5	02/02/2013 00:00:00	0	1750	20130202	$ 2800.00	2013	2	2-Feb-2013
17	5963EAA5-4F09-42F0-888F-321FDCD522DC	4	02/03/2013 00:00:00	0	990	20130302	$ 1584.00	2013	3	2-Mar-2013

Figure 13-2. *Applying the FORMAT() function*

■ **Note** To reformat a date like this, the column that you are using for the original data must either be of the date data type or be capable of being interpreted as a date by Power BI Desktop.

The date format that you applied in this example was not predefined by DAX in any way. In fact, it was assembled from a set of day, month, and year codes that you can combine to create the date format that you want. Table 13-2 explains the codes that are available.

Table 13-2. *Custom Date Formats*

Format Code	Description	Example
`d`	The day of the month.	`"d MMM yyyy"` produces 2 Jan 2016
`dd`	The day of the month with a leading zero when necessary.	`"dd MMM yyyy"` produces 02 Jan 2016
`ddd`	The three-letter abbreviation for the day of the week.	`"ddd d MMM yyyy"` produces Sat 2 Jan 2016
`dddd`	The day of the week in full.	`"ddd dd MMM yyyy"` produces Saturday 02 Jan 2016
`M`	The number of the month.	`"dd M yyyy"` produces 02 1 2016
`MM`	The number of the month with a leading zero when necessary.	`"dd MM yyyy"` produces 02 01 2016
`MMM`	The three-letter abbreviation for the month.	`"dd MMM yyyy"` produces 02 Jan 2016
`MMMM`	The full month.	`"dd MMMM yyyy"` produces 02 January 2016
`yy`	The year as two digits.	`"d MMM yy"` produces 2 Jan 16
`yyyy`	The full year.	`"MMMM yyyy"` produces January 2016

If you do not want to build your own date formats, then you can choose from the seven predefined date and time formats that Power BI Desktop has available. These are explained in Table 13-3.

Table 13-3. *Predefined Date Formats*

Format Code	Description	Example	Comments
Short Date	The short date as defined in the PC's settings.	`FORMAT([InvoiceDate], "Short Date")`	Formats the date using figures only.
Medium Date	The medium date as defined in the PC's settings.	`FORMAT([InvoiceDate], "Medium Date")`	Formats the date with the month as a text and the day of the week.
Long Date	The long date as defined in the PC's settings.	`FORMAT([InvoiceDate], "Long Date")`	Formats the date with the month as a text and the day of the week.
General Date	The date defined by the local PC's culture settings.	`FORMAT([InvoiceDate], "General Date")`	Formats the date defined by the local PC's culture settings.
Long Time	The long time as defined in the PC's settings.	`FORMAT([InvoiceDate], "Long Time")`	Formats the datetime or time column with the hour and minutes of the day.
Medium Time	The medium time as defined in the PC's settings.	`FORMAT([InvoiceDate], "Short Time")`	Formats the datetime or time column with the hour and minutes of the day.
Short Time	The short time as defined in the PC's settings.	`FORMAT([InvoiceDate], "Short Time")`	Formats the datetime or time column with the hour and minutes and seconds of the day.

■ **Note** As I explained in Chapter 11, the `FORMAT()` function converts a field to a text, so you need to be aware that this has the potential to restrict its use if further calculations are applied to the result produced by this formula. In my opinion, it is essentially useful when applied to modify the way that dates are displayed.

Calculating the Age of Cars Sold

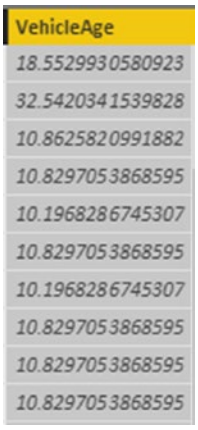
To continue with elementary DAX formulas—and as an admittedly extremely simple example—I will presume that we need to calculate the age of every car sold relative to the current date. As the source data contains the registration date for each vehicle, this will not be difficult. So, you have to create the formula that you can see finished in step 8:

1. With the Stock table selected, activate the Modeling ribbon and click the New Column button.
2. Enter a left parenthesis to the right of the equals sign.
3. Type **NOW()**. Make sure that you add the left and right parentheses even if these remain empty.

4. Enter a minus sign and then enter (or select) **[Registration_Date]**.
5. Enter a right parenthesis. This corresponds to the left parenthesis before the `NOW()` function.
6. Enter a forward slash (the division symbol).
7. Enter **365**.
8. Replace Column with **VehicleAge**. The formula should look like this:

```
VehicleAge=(NOW()-[Registration_Date])/365
```

9. Press Enter or click the tick box in the formula bar. The formula will be added to the entire new column. Figure 13-3 shows you a small sample of the result of this operation.



VehicleAge
18.5529930580923
32.5420341539828
10.8625820991882
10.8297053868595
10.1968286745307
10.8297053868595
10.1968286745307
10.8297053868595
10.8297053868595
10.8297053868595

Figure 13-3. *Calculated output using the NOW() function*

Let me be clear, this is not the only way to calculate a time difference using DAX. It is probably not even the best one. It is merely a simple yet (I hope) comprehensible introduction to a DAX Date/Time calculation formula and it reminds you just how close a cousin DAX is to the Excel formula language. It also shows you an easy way to see exactly what DAX functions are available and what they do, since each function displays a brief explanation when you hover the mouse pointer over it.

Adding Time Intelligence to a Data Model

I want now to explain the vital set of functions that concern time, or rather, the dates used in analyzing data. Power BI Desktop calls this time intelligence (even though it nearly always refers to the use of date ranges). Applying this kind of temporal analysis can be a fundamental aspect of data presentation in business intelligence. After all, what enterprise does not need to know how this year's figures compare to last year's and what kind of progress is being made?

Time intelligence always requires a valid date table, which is one of the reasons why we will now spend a certain amount of time creating this core pillar of a successful data model. Then the date table has to be joined to the table containing the data that you want to compare over time on a date field. The good news is that once you have a valid date table, and have acquainted yourself with a handful of data and time functions in DAX, you can deliver some extremely impressive results. These kinds of calculations can cover (among other things) the following:

- YearToDate, QuarterToDate, and MonthToDate calculations
- Comparisons with previous years, quarters, or months
- Rolling aggregations over a period of time, such as the sum for the last three months
- Comparison with a parallel period in time, such as the same month in the previous year

An introduction to time intelligence in Power BI Desktop gives you a taste of some of the DAX functions that you are likely to use when analyzing data over time. To begin with, you will learn how to create a date table. Then you will look at some of the different types of calculations that you can create to analyze data over time.

Creating and Applying a Date Table

For time intelligence to work, you need a table that contains an *uninterrupted* range of dates that begins at least at the *earliest* date in your data, and that ends with a date at least equal to the *final* date in your data. In practice, this will nearly always mean creating a date table that begins on the 1st of January of the earliest date in your data and that ends on the 31st of December of the last year for which you have data. Once you have your date table, you can join it to one of the tables in your data model and then begin to exploit all the time-related analytical functions of Power BI Desktop.

The good news is that you can use Power BI Desktop itself to create a date table. It is also possible to import a contiguous range of dates from other applications such as Excel. Because I prefer you to come to appreciate the breadth and depth of DAX, I will explain how to implement a date table directly in Power BI Desktop.

Creating the Date Table

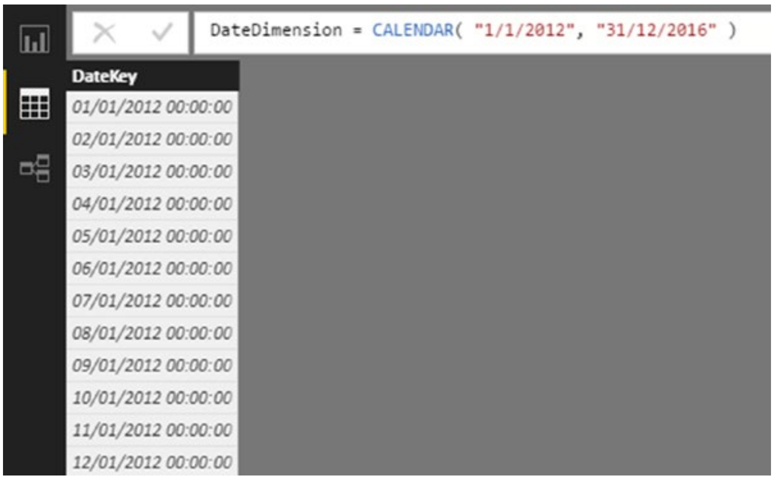
The first requirement before starting to apply time intelligence is to have a valid date table. The following steps explain one way to create a date table using and extending your newly acquired DAX skills:

1. In Data View, activate the Modeling ribbon and click the New Table button. The expression `Table =` will appear in the formula bar.
2. Replace the word Table with **DateDimension**.
3. Click to the right of the equals sign and enter **CALENDAR(**.
4. Enter **"1/1/2012"** (include the double quotes). This is the starting date for a table of dates. I am assuming here that your computer is configured for the UK or European date format. If this is not the case, then enter the date as you would normally using your local date format.
5. Enter a comma.
6. Enter **"31/12/2016"** (or the equivalent date format that represents the 31st of December 2016 in your local date format). This is the end date for a table of dates.
7. Enter a right parenthesis. This corresponds to the left parenthesis of the `CALENDAR()` function. The formula bar will display this:

```
DateDimension = CALENDAR( "1/1/2012", "31/12/2016" )
```

8. Press Enter or click the tick icon in the formula bar. Power BI Desktop will create a table containing a single column of dates from the 1st of January 2012 until the 31st of December 2016.

9. In the Fields list, right-click the Date field in the DateDimension table and select Rename. Rename the Date field to **DateKey**. The date table will look like Figure 13-4.



```
DateDimension = CALENDAR( "1/1/2012", "31/12/2016" )
```

DateKey
01/01/2012 00:00:00
02/01/2012 00:00:00
03/01/2012 00:00:00
04/01/2012 00:00:00
05/01/2012 00:00:00
06/01/2012 00:00:00
07/01/2012 00:00:00
08/01/2012 00:00:00
09/01/2012 00:00:00
10/01/2012 00:00:00
11/01/2012 00:00:00
12/01/2012 00:00:00

Figure 13-4. *An initial date table for a time dimension*

10. Add 24 new columns containing the formulas explained in Table 13-4. Because Chapter 12 provided an exhaustive explanation covering the techniques that you need to apply when adding new columns, I will not repeat the process in detail here.

Table 13-4. *DAX Formulas to Extend a Date Table*

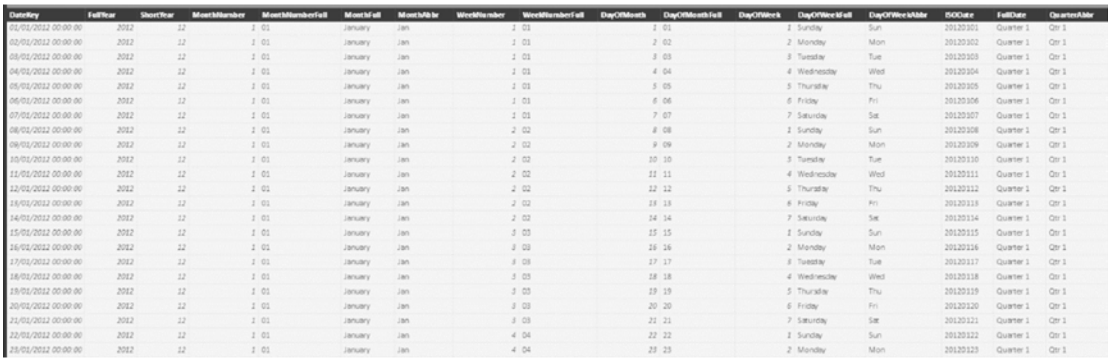
Column Title	Formula	Comments
FullYear	`YEAR([DateKey])`	Isolates the year as a four-digit number.
ShortYear	`VALUE(Right(Year([DateKey]),2))`	Isolates the year as a two-digit number.
MonthNumber	`MONTH([DateKey])`	Isolates the number of the month in the year as one or two digits.
MonthNumberFull	`FORMAT([DateKey], "MM")`	Isolates the number of the month in the year as two digits, with a leading zero for the first nine months.
MonthFull	`FORMAT([DateKey], "MMMM")`	Displays the full name of the month.
MonthAbbr	`FORMAT([DateKey], "MMM")`	Displays the name of the month as a three-letter abbreviation.
WeekNumber	`WEEKNUM([DateKey])`	Shows the number of the week in the year.

(*continued*)

Table 13-4. *(continued)*

Column Title	Formula	Comments
WeekNumberFull	`FORMAT(Weeknum([DateKey]), "00")`	Shows the number of the week in the year with a leading zero for the first nine weeks.
DayOfMonth	`DAY([DateKey])`	Displays the number of the day of the month.
DayOfMonthFull	`FORMAT(Day([DateKey]),"00")`	Displays the number of the day of the month with a leading zero for the first nine days.
DayOfWeek	`WEEKDAY([DateKey])`	Displays the number of the day of the week.
DayOfWeekFull	`FORMAT([DateKey],"dddd")`	Displays the name of the weekday.
DayOfWeekAbbr	`FORMAT([DateKey],"ddd")`	Displays the name of the weekday as a three-letter abbreviation.
ISODate	`[FullYear] & [MonthNumberFull] & [DayOfMonthFull]`	Displays the date in the ISO (internationally recognized) format of YYYYMMDD.
FullDate	`[DayOfMonth] & " " & [MonthFull] & " " & [FullYear]`	Displays the full date with spaces.
QuarterFull	`"Quarter " & ROUNDDOWN(MONTH ([DateKey])/4,0)+1`	Displays the current quarter.
QuarterAbbr	`"Qtr " &ROUNDDOWN(MONTH([Date Key])/4,0)+1`	Displays the current quarter as a three-letter abbreviation plus the quarter number.
Quarter	`"Q" &ROUNDDOWN(MONTH ([DateKey])/4,0)+1`	Displays the current quarter in short form.
QuarterNumber	`ROUNDDOWN(MONTH([Date Key])/4,0)+1`	Displays the number of the current quarter. This is essentially used as a Sort By column.
QuarterAndYear	`DateDimension[Quarter] & " " & DateDimension[FullYear]`	Shows the quarter and the year.
MonthAndYearAbbr	`DateDimension[MonthAbbr] & " " & [FullYear]`	Shows the abbreviated month and year.
QuarterAndYearNumber	`[FullYear] & [QuarterNumber]`	Shows the year and quarter numbers. This is essentially used as a Sort By column.
YearAndWeek	`VALUE([FullYear] &[WeekNumberFull])`	Indicates the year and week. The `VALUE()` function ensures that the figure is considered as numeric by Power BI Desktop.
YearAndMonthNumber	`VALUE(DateDimension[FullYear] & DateDimension[MonthNumberFull])`	A numeric value for the year and month. This is essentially used as a Sort By column.

The first few columns of the DateDimension table should now look like Figure 13-5.



DateKey	FullYear	ShortYear	MonthNumber	MonthNumberFull	MonthFull	MonthAbbr	WeekNumber	WeekNumberFull	DayOfMonth	DayOfMonthFull	DayOfWeek	DayOfWeekFull	DayOfWeekAbbr	ISODate	FullDate	QuarterAbbr
01/01/2012 00:00:00	2012	12	1	01	January	Jan	1	01	1	01	1	Sunday	Sun	20120101	Quarter 1	Qtr 1
02/01/2012 00:00:00	2012	12	1	01	January	Jan	1	01	2	02	2	Monday	Mon	20120102	Quarter 1	Qtr 1
03/01/2012 00:00:00	2012	12	1	01	January	Jan	1	01	3	03	3	Tuesday	Tue	20120103	Quarter 1	Qtr 1
04/01/2012 00:00:00	2012	12	1	01	January	Jan	1	01	4	04	4	Wednesday	Wed	20120104	Quarter 1	Qtr 1
05/01/2012 00:00:00	2012	12	1	01	January	Jan	1	01	5	05	5	Thursday	Thu	20120105	Quarter 1	Qtr 1
06/01/2012 00:00:00	2012	12	1	01	January	Jan	1	01	6	06	6	Friday	Fri	20120106	Quarter 1	Qtr 1
07/01/2012 00:00:00	2012	12	1	01	January	Jan	1	01	7	07	7	Saturday	Sat	20120107	Quarter 1	Qtr 1
08/01/2012 00:00:00	2012	12	1	01	January	Jan	2	02	8	08	1	Sunday	Sun	20120108	Quarter 1	Qtr 1
09/01/2012 00:00:00	2012	12	1	01	January	Jan	2	02	9	09	2	Monday	Mon	20120109	Quarter 1	Qtr 1
10/01/2012 00:00:00	2012	12	1	01	January	Jan	2	02	10	10	3	Tuesday	Tue	20120110	Quarter 1	Qtr 1
11/01/2012 00:00:00	2012	12	1	01	January	Jan	2	02	11	11	4	Wednesday	Wed	20120111	Quarter 1	Qtr 1
12/01/2012 00:00:00	2012	12	1	01	January	Jan	2	02	12	12	5	Thursday	Thu	20120112	Quarter 1	Qtr 1
13/01/2012 00:00:00	2012	12	1	01	January	Jan	2	02	13	13	6	Friday	Fri	20120113	Quarter 1	Qtr 1
14/01/2012 00:00:00	2012	12	1	01	January	Jan	2	02	14	14	7	Saturday	Sat	20120114	Quarter 1	Qtr 1
15/01/2012 00:00:00	2012	12	1	01	January	Jan	3	03	15	15	1	Sunday	Sun	20120115	Quarter 1	Qtr 1
16/01/2012 00:00:00	2012	12	1	01	January	Jan	3	03	16	16	2	Monday	Mon	20120116	Quarter 1	Qtr 1
17/01/2012 00:00:00	2012	12	1	01	January	Jan	3	03	17	17	3	Tuesday	Tue	20120117	Quarter 1	Qtr 1
18/01/2012 00:00:00	2012	12	1	01	January	Jan	3	03	18	18	4	Wednesday	Wed	20120118	Quarter 1	Qtr 1
19/01/2012 00:00:00	2012	12	1	01	January	Jan	3	03	19	19	5	Thursday	Thu	20120119	Quarter 1	Qtr 1
20/01/2012 00:00:00	2012	12	1	01	January	Jan	3	03	20	20	6	Friday	Fri	20120120	Quarter 1	Qtr 1
21/01/2012 00:00:00	2012	12	1	01	January	Jan	3	03	21	21	7	Saturday	Sat	20120121	Quarter 1	Qtr 1
22/01/2012 00:00:00	2012	12	1	01	January	Jan	4	04	22	22	1	Sunday	Sun	20120122	Quarter 1	Qtr 1
23/01/2012 00:00:00	2012	12	1	01	January	Jan	4	04	23	23	2	Monday	Mon	20120123	Quarter 1	Qtr 1

Figure 13-5. *The completed DateDimension table*

The point behind creating all these ways of expressing dates and parts of dates is that you can now use them in your tables, charts, and gauges to aggregate and display data over time. Any record that has a date element can now be expressed visually; not just as the date itself, but shown as and aggregated as years, quarters, months, or weeks. The trick is to prepare all the time groupings that you are likely to need in the date table of your dashboards. However, you do not need to worry if you find yourself later needing an extra column or two further down the line, because you can always add columns that contain other date elements.

Adding Sort By Columns to the Date Table

In Chapter 11, you saw how to sort a column using the data in another column to provide the sort order. This technique is essential when dealing with date tables, as you want to be sure that any visualizations that contain date elements appear in the right order. The classic example is months. As things stand, if you were to use the MonthFull column or MonthAbbr column in a chart or table, then you would see the month names appearing on an axis or in a column in alphabetical order.

To avoid this, you have to add one final tweak to the date table and apply a Sort By column to certain of the date elements in other columns. Since you saw how this is done in Chapter 11, now I will only provide the list of columns that need this extra tweak, rather than reiterating all the details. Table 13-5 gives you the required information to extend the data table so that all date elements are sorted correctly.

Table 13-5. *The Sort By Columns Needed for the Date Table*

Column	Sort By Column
FullDate	DateKey
MonthFull	MonthNumber
MonthAbbr	MonthNumber
DayOfWeekFull	DayOfWeek
DayOfWeekAbbr	DayOfWeek
Quarter And Year	QuarterAndYearNumber
MonthAndYearAbbr	YearAndMonthNumber
MonthAndYear	YearAndMonthNumber

Date Table Techniques

When using date tables to invoke time intelligence in DAX, there are two fundamental principles that must always be applied. I realize that I mention them elsewhere, but they are so essential that they bear repetition.

- The date range must be *continuous*; that is, there must not be any dates missing in the column that contains the list of calendar days in the table of dates.
- The date range must encompass *all the dates* that you are using in other tables in the data model.

The first requirement is covered by the use of the `CALENDAR()` function to create a date table. The second can require that you discover the lower and upper date thresholds in one or more tables of dates. Since this can be a little laborious (not to mention error-prone), it is probably easier to get DAX to find these dates for you and apply them to the `CALENDAR()` function.

Consequently, once you know where the lowest and highest dates are in your data model (even if they are in separate tables), you can use these dates as the lower and upper boundaries of the `CALENDAR()` function. In the case of the Brilliant British Cars data, the formula could read as follows:

```
DateDimension = CALENDAR(MIN('Stock'[PurchaseDate]), MAX('Invoices'[InvoiceDate]))
```

This formula shows that you can apply two functions that you have seen already—`MIN()` and `MAX()`—with date and datetime data types. They simply tell DAX to find the earliest and latest dates in a column.

Alternatively, and as a nod to best practice, you may prefer to ensure that the date dimension always covers *entire years* of dates. In this case, the formula to create the table would be as follows:

```
DateDimension = CALENDAR(STARTOFYEAR(MIN('Stock'[PurchaseDate])), ENDOFYEAR(MAX('Invoices'
[InvoiceDate])))
```

This formula extends the DAX calculation by using two functions that were explained in Table 13-1:

- `STARTOFYEAR()`: Deduces the first day of the year.
- `ENDOFYEAR()`: Deduces the last day of the year.

This formula also shows you that you can define a date range using different columns, or even different tables when you are specifying a date range. This is particularly useful when defining a date table.

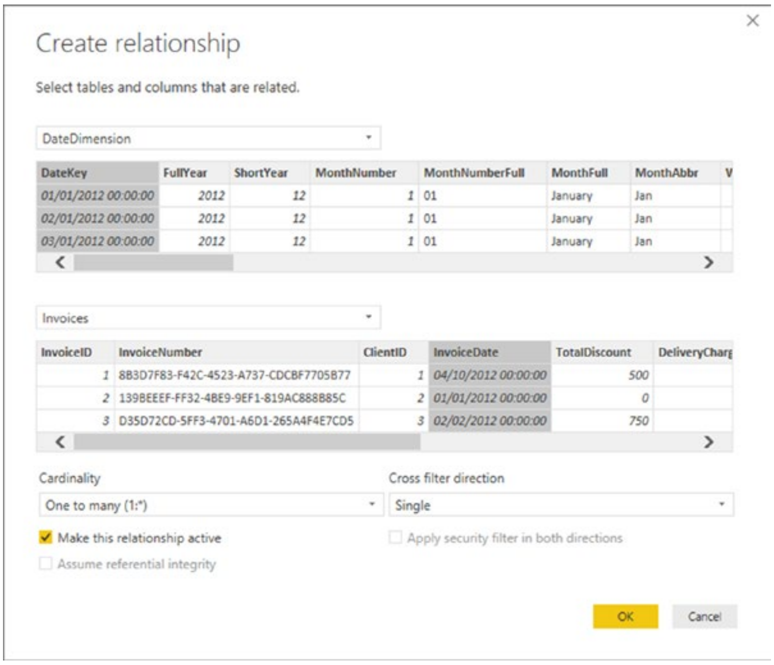
■ **Note** The principal advantage of looking up the date boundaries like this is that they update automatically as the source data changes. So if the Stock table or Invoices table is extended in the data source to contain further rows with dates outside the existing date range, then the DateDimension table will also grow to encompass these new dates.

Creating a data table can be fun, but nonetheless takes a few minutes. So here's a tip that I can give you: create a Power BI Desktop file that contains nothing but a date dimension table (using a manually defined start date and end date, just as you saw at the beginning of this example) with all the other columns added. You can then make copies of this "template" file and use them as the basis for any new data models that you create. This can include replacing the fixed threshold dates with references to the data in tables, as mentioned previously.

Adding the Date Table to the Data Model

Now that you have a date table, you can integrate it with your data model so that you can start to apply some of the time intelligence that DAX makes possible.

1. Click the Relationships View icon to display the tables in the data model.
2. In the Home ribbon, click the Manage Relationships button. The Manage Relationships dialog will appear.
3. Click the New button. The Create Relationship dialog will appear.
4. At the top of the dialog, select the DateDimension table from the popup list.
5. Once the sample data from the DateDimension table is displayed, click inside the DateKey column to select it.
6. Under the DateDimension table sample data, select the Invoices table from the popup list.
7. Once the sample data from the Invoices table is displayed, click inside the InvoiceDate column to select it. The dialog will look like Figure 13-6.


Create relationship

Select tables and columns that are related.

DateDimension

DateKey	FullYear	ShortYear	MonthNumber	MonthNumberFull	MonthFull	MonthAbbr	V
01/01/2012 00:00:00	2012	12	1	01	January	Jan	
02/01/2012 00:00:00	2012	12	1	01	January	Jan	
03/01/2012 00:00:00	2012	12	1	01	January	Jan	

Invoices

InvoiceID	InvoiceNumber	ClientID	InvoiceDate	TotalDiscount	DeliveryCharg
1	8B3D7F83-F42C-4523-A737-CDCBF7705B77	1	04/10/2012 00:00:00	500	
2	139BEEEF-FF32-4BE9-9EF1-819AC888B85C	2	01/01/2012 00:00:00	0	
3	D35D72CD-5FF3-4701-A6D1-265A4F4E7CD5	3	02/02/2012 00:00:00	750	

Cardinality: One to many (1:*)

Cross filter direction: Single

☑ Make this relationship active

☐ Apply security filter in both directions

☐ Assume referential integrity

Figure 13-6. *Adding a date dimension to the data model*

8. Click OK. The relationship will appear in the Manage Relationships dialog.
9. Click Close. The DateDimension table will appear in the data model joined to the Invoices table. You can see this in Figure 13-7.

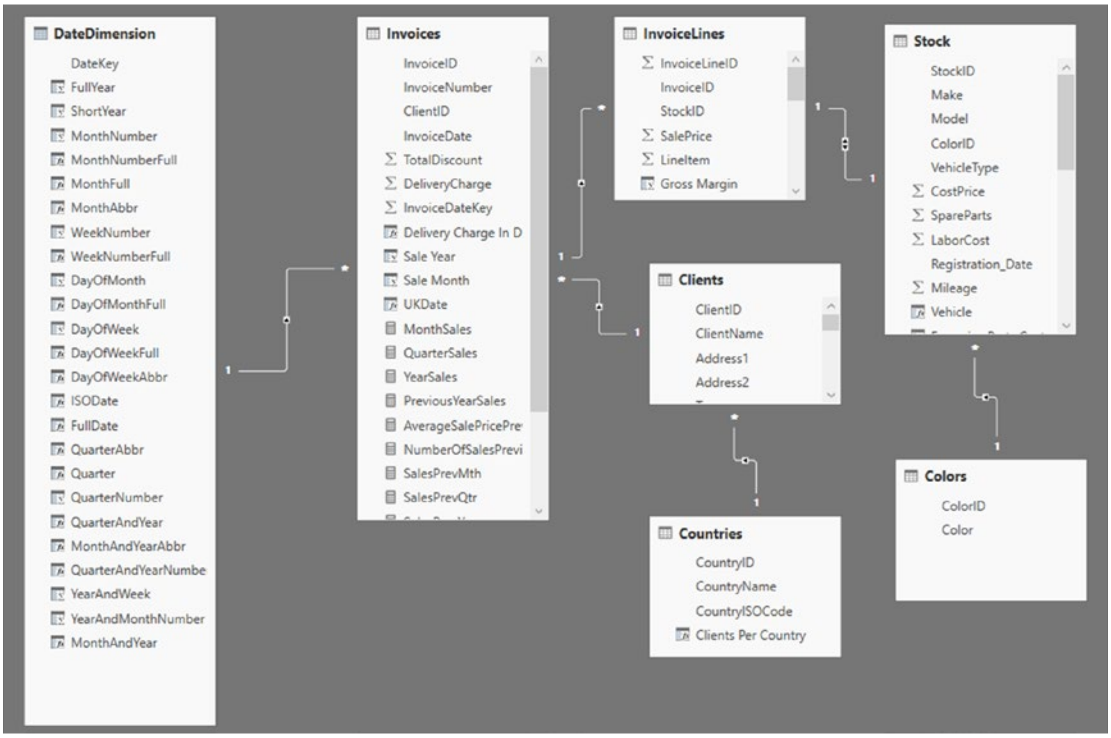


Figure 13-7. A data model with a data table added

■ **Note** For time intelligence in Power BI Desktop to work correctly, the fields used to join a date table and a data table must both be set to the *date* or *datetime* data type.

Calculating the Difference Between Two Dates

Now that you have a date dimension in the data model, you can create some more powerful date and time calculations. For instance, and as you would probably expect, DAX can do more than just specify a number of days. As befits a formula language that is designed to aid in business analysis, it can deduce the time between two dates expressed as the following:

- Years
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

This can be extremely useful when you want to classify records according to a duration, and can be calculated using the `DATEDIFF()` function. The `DATEDIFF()` function expects you to apply three parameters when calculating an interval:

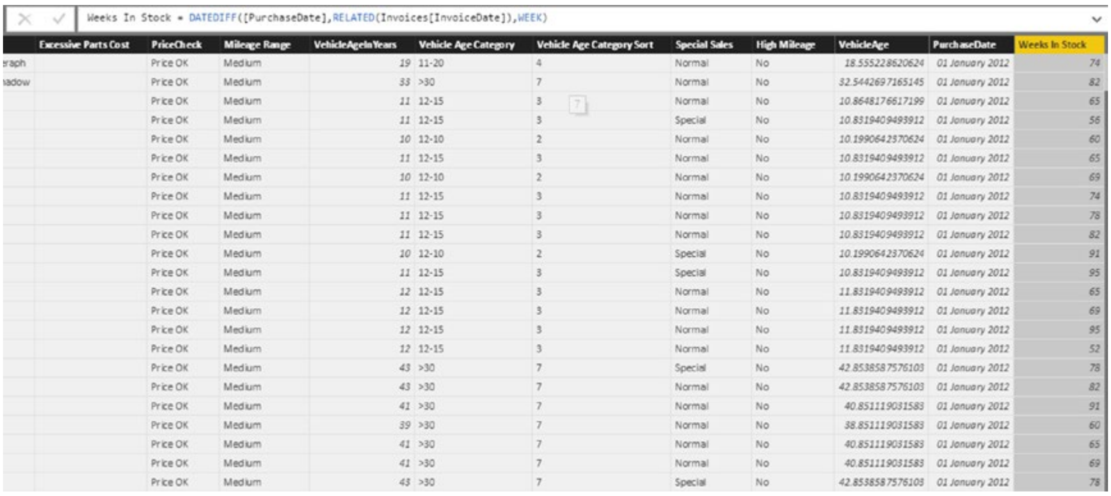
- The start date for the calculation of the interval.
- The end date up until when the interval will be calculated.
- The interval to calculate. This could be in years, days, or minutes, for example.

As an example, imagine that you want to display the number of weeks that each vehicle remained in stock, as this will help you to determine the fastest-selling models and consequently optimize the company's cash flow. As the data contains both the purchase date for each vehicle and its sale date (even if the two are in different tables), this can be done using the DAX `DATEDIFF()` function (as you can see in the finished formula in step 12) as follows:

1. In the CarSalesDataWithNewColumnsAndMeasures.pbix file, click the Data View icon and then click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon. A new column named Column will appear at the right of any existing columns.
3. To the right of the equals sign, enter **DATEDIFF(**. You will see that as you enter the first few characters, the list of functions will list all available functions beginning with these characters. You can click the function name to have it appear in the formula bar.
4. Press the **[** key. The list of available fields from the current table will appear.
5. Select the [PurchaseDate] field.
6. Enter a comma.
7. Type **RELATED(**. The popup will display all the fields that can be linked via the data model to this table.
8. Scroll down the list and select Invoices[InvoiceDate].
9. Add a right parenthesis to close the `RELATED()` function.
10. Enter a comma. A popup list of available intervals will appear.
11. Select WEEK.
12. Add a final right parenthesis. The formula will look like this:

    ```
    Weeks In Stock = DATEDIFF([PurchaseDate],RELATED(Invoices[InvoiceDate]),WEEK)
    ```

13. Press Enter or click the tick box in the formula bar. The formula will be added to the entire new column. You can now rename it **Weeks In Stock**. Figure 13-8 shows you a small sample of the result of this operation.



```
Weeks In Stock = DATEDIFF([PurchaseDate],RELATED(Invoices[InvoiceDate]),WEEK)
```

	Excessive Parts Cost	PriceCheck	Mileage Range	VehicleAgeInYears	Vehicle Age Category	Vehicle Age Category Sort	Special Sales	High Mileage	VehicleAge	PurchaseDate	Weeks In Stock
raph		Price OK	Medium	19	11-20	4	Normal	No	18.555228620624	01 January 2012	74
adow		Price OK	Medium	33	>30	7	Normal	No	32.5442697165145	01 January 2012	82
		Price OK	Medium	11	12-15	3	Normal	No	10.8648176617199	01 January 2012	65
		Price OK	Medium	11	12-15	3	Special	No	10.8319409493912	01 January 2012	56
		Price OK	Medium	10	12-10	2	Normal	No	10.1990642370624	01 January 2012	60
		Price OK	Medium	11	12-15	3	Normal	No	10.8319409493912	01 January 2012	65
		Price OK	Medium	10	12-10	2	Normal	No	10.1990642370624	01 January 2012	69
		Price OK	Medium	11	12-15	3	Normal	No	10.8319409493912	01 January 2012	74
		Price OK	Medium	11	12-15	3	Normal	No	10.8319409493912	01 January 2012	78
		Price OK	Medium	11	12-15	3	Normal	No	10.8319409493912	01 January 2012	82
		Price OK	Medium	10	12-10	2	Special	No	10.1990642370624	01 January 2012	91
		Price OK	Medium	11	12-15	3	Special	No	10.8319409493912	01 January 2012	95
		Price OK	Medium	12	12-15	3	Normal	No	11.8319409493912	01 January 2012	65
		Price OK	Medium	12	12-15	3	Normal	No	11.8319409493912	01 January 2012	69
		Price OK	Medium	12	12-15	3	Normal	No	11.8319409493912	01 January 2012	95
		Price OK	Medium	12	12-15	3	Normal	No	11.8319409493912	01 January 2012	52
		Price OK	Medium	43	>30	7	Special	No	42.8538587576103	01 January 2012	78
		Price OK	Medium	43	>30	7	Normal	No	42.8538587576103	01 January 2012	82
		Price OK	Medium	41	>30	7	Normal	No	40.851119031583	01 January 2012	91
		Price OK	Medium	39	>30	7	Normal	No	38.851119031583	01 January 2012	60
		Price OK	Medium	41	>30	7	Normal	No	40.851119031583	01 January 2012	65
		Price OK	Medium	41	>30	7	Normal	No	40.851119031583	01 January 2012	69
		Price OK	Medium	43	>30	7	Special	No	42.8538587576103	01 January 2012	78

Figure 13-8. *The results of a DATEDIFF() function*

You can now see the number of weeks that each vehicle was in stock before being sold and use this figure in Power BI Desktop dashboards. Only *complete* intervals are displayed. In other words, if you have selected YEAR as the interval, then the difference between the two dates must be fractionally more than one year for the function to return 1. Notice that you used the `RELATED()` function again to compare elements from separate tables. Once again, Power BI Desktop only listed the tables that were correctly linked to the destination table when you applied this function.

■ **Note** When you use the `DATEDIFF()` function, you must always add the lower date (the Purchase date in this example) as the *first date* that the function uses. If you do not, you will get an error message.

As you saw in the popup in step 10, the `DATEDIFF()` function lets you choose from a range of available intervals. These are explained in Table 13-6.

Table 13-6. *Date Difference Intervals*

Function	Description
`YEAR`	Returns the time difference in complete years.
`QUARTER`	Returns the time difference in complete quarters.
`MONTH`	Returns the time difference in complete months.
`WEEK`	Returns the time difference in complete weeks.
`DAY`	Returns the time difference in complete days.
`HOUR`	Returns the time difference in complete hours.
`MINUTE`	Returns the time difference in complete minutes.
`SECOND`	Returns the time difference in complete seconds.

■ **Note** DAX cannot currently handle negative date differences, so you need to ensure that your data has been correctly prepared in Power BI Desktop Query *before* you calculate date and time differences.

Applying Time Intelligence

Now that all the preparations have been completed, it is (finally) time to see just how DAX can make your life easier when it comes to calculating metrics over time. This is possible only because you have a date table in place and it is connected to the requisite date field in the table that contains the data you want to aggregate. However, with the foundations in place, you can now start to deliver some really interesting and persuasive output.

YearToDate, QuarterToDate, and MonthToDate Calculations

To begin, let's resolve a simple but frequent requirement: calculating month-to-date, quarter-to-date, and year-to-date sales figures.

The three functions that you will see in this example are extremely similar. Consequently, I will only explain the first one (a month-to-date calculation that you can see in step 9), and then will let you create the next two in a couple of copy, paste, and tweak operations.

1. In the Data View, select the Invoices table, and in the Modeling ribbon, click New Measure.
2. In the formula bar, replace Measure with **MonthSales**.
3. To the right of the equals sign, enter (or type and select) **TOTALMTD(**. This will apply the month-to-date aggregation for a field.
4. Enter (or type and select) **SUM(**. This specifies the actual aggregation that you want to apply.
5. Select the InvoiceLines[SalePrice] field from the popup list of available fields.
6. Enter a right parenthesis to end the `SUM()` function.
7. Enter a comma.
8. Type the first few characters of the date table (**DateDimension** in this example) and select the date key field (DateKey in this example).
9. Enter a right parenthesis to end the `TOTALMTD()` function. The formula should read as follows:

   ```
   MonthSales = TOTALMTD(SUM(InvoiceLines[SalePrice]),DateDimension[DateKey])
   ```

10. Press Enter or click the tick mark icon to complete the measure definition.
11. Format the measure just as you would format a column (I am applying the pounds sterling format).

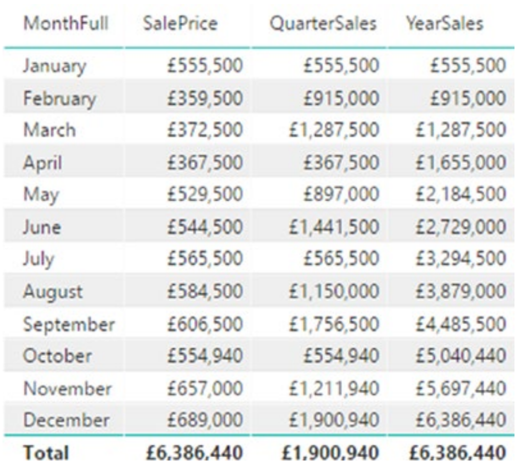
Now copy the formula that you just created and use it as the basis for two new measures. These will be quarterly sales to date and annual sales to date, as follows:

```
QuarterSales = TOTALQTD(SUM(InvoiceLines[SalePrice]),DateDimension[DateKey])
YearSales = TOTALYTD(SUM(InvoiceLines[SalePrice]),DateDimension[DateKey])
```

The three formulas that you have used are TOTALMTD() for the month-to-date aggregation, TOTALQTD() for the quarter-to-date aggregation, and TOTALYTD() for the year-to-date aggregation. All three functions take the following two parameters:

- The *aggregate function*. Depends on the actual metric that you want to deliver and the table and column that is aggregated. (SUM() was used here, although it could have been AVERAGE(), or COUNT(), or any other of the aggregate functions that you saw in Chapter 12.)
- The *key field of the date table*. Since the Invoices table is linked to the date table in the data model using the InvoiceDate field, DAX can apply the correct calculation if—and only if—you have added a date table to the data model and then specified the key field of the date table.

Since you have created these measures, I imagine that you would like to see them in action. Figure 13-9 shows the quarter-to-date and year-to-date sales for 2014 (along with the aggregated sales from the initial SalePrice column in the InvoiceLines table) in a simple Power BI Desktop table like the one that you created in Chapter 1 (you will see a lot more in the next chapter). To obtain this result, you have to drag the FullYear field from the DateDimension table onto the Page Level Filters area and then select only the check box for 2014. This restricts the dates to the year 2014. I realize that I am anticipating the content of Chapter 20 here; however, filtering at this level is so intuitive that it is worth taking a quick look now.



MonthFull	SalePrice	QuarterSales	YearSales
January	£555,500	£555,500	£555,500
February	£359,500	£915,000	£915,000
March	£372,500	£1,287,500	£1,287,500
April	£367,500	£367,500	£1,655,000
May	£529,500	£897,000	£2,184,500
June	£544,500	£1,441,500	£2,729,000
July	£565,500	£565,500	£3,294,500
August	£584,500	£1,150,000	£3,879,000
September	£606,500	£1,756,500	£4,485,500
October	£554,940	£554,940	£5,040,440
November	£657,000	£1,211,940	£5,697,440
December	£689,000	£1,900,940	£6,386,440
Total	**£6,386,440**	**£1,900,940**	**£6,386,440**

Figure 13-9. *The quarter- and year-to-date functions in DAX*

As you can see, you have each month's sales figures along with the cumulative sales for each quarter to date (and restarting each quarter). The final column shows you the yearly sales total for each month to date. Also, note that the months appear in calendar order, because the MonthFull field has used the MonthNumber field as its Sort By column.

Conceptually, the three functions that are outlined—`TOTALMTD()`, `TOTALQTD()`, and `TOTALYTD()`—can be considered as `CALCULATE()` functions that have been extended to deliver a specific result for a time frame. You can always calculate aggregations for a period to date using the `CALCULATE()` function if you want to. However, since this "shorthand" version is so practical and easy to use, I see no reason to try anything more complicated when there is no real need.

■ **Note** In this example, I suggest starting the process of adding a new measure with the Invoices table selected, merely because this table seems a good place to store the metric. You can create the metric in virtually any table in practice—provided that you have a coherent data model to build on.

Analyzing Data As a Ratio over Time

Looking at how data aggregates over time is only one of the ways that time intelligence can enable you to deliver time-based analysis. On occasion, you may well want to see how a day's sales relate to the total sales for a period. So in this example, we will calculate the daily percentage of sales for Brilliant British Cars relative to the total sales for the year 2014. The completed code for this slightly complex function is in step 21, should you prefer to type or copy and paste it.

1. In the Data View, select the InvoiceLines table and click New Measure.
2. Replace Measure with **PercentOfYear**.
3. To the right of the equals sign, enter (or type and select) **SUM(**.
4. Select the [SalePrice] field from the popup list of available fields.
5. Insert a closing parenthesis for the `SUM()` function.
6. Enter a forward slash to indicate division.
7. Enter (or type and select) **CALCULATE(SUM(**. This tells DAX that you want a filtered calculation and that the specific aggregation that you want to apply is a `SUM()` function.
8. Select the [SalePrice] field from the popup list of available fields.
9. Enter a right parenthesis to end the `SUM()` function.
10. Enter a comma. This tells the `CALCULATE()` function that you have chosen the aggregation that you want and will now apply a filter.
11. Enter (or type and select) **DATESBETWEEN(**.
12. Start typing the name of the date table (**DateDimension**) and then select the DateKey field from the popup. This tells the `DATESBETWEEN()` function which field in the time dimension should be used as its first parameter.
13. Add a comma. This ends the first parameter for the `DATESBETWEEN()` function.
14. Enter (or type and select) **STARTOFYEAR(**.
15. Start typing the name of the date table (**DateDimension**) and then select the DateKey field from the popup. This will be the lower boundary of the time span that will be used to filter the `CALCULATE()` function.

16. Add a right parenthesis to close the `STARTOFYEAR()` function.
17. Add a comma. This ends the second parameter for the `DATESBETWEEN()` function.
18. Enter (or type and select) **ENDOFYEAR(**.
19. Once again, start typing the name of the date table (**DateDimension**) and then select the DateKey field from the popup. This will be the upper boundary of the time span that will be used to filter the `CALCULATE()` function.
20. Add a right parenthesis to close the `ENDOFYEAR()` function.
21. Add two right parentheses. The first one ends the `DATESBETWEEN()` function, and the final parenthesis closes the `CALCULATE()` function. The formula should look like this:

```
PercentOfYear = Sum([SalePrice]) / CALCULATE(SUM([SalePrice]),
DATESBETWEEN(DateDimension[DateKey],STARTOFYEAR(DateDimension
[DateKey]),ENDOFYEAR(DateDimension[DateKey])))
```

22. Press Enter or click the tick mark icon to complete the measure definition.
23. In the Modeling ribbon, click the percentage icon to format this measure as a percentage.

This formula was a little more intricate than those that you have seen so far in this chapter. So let me explain it in a bit more detail. At its heart, this formula consists of two main elements:

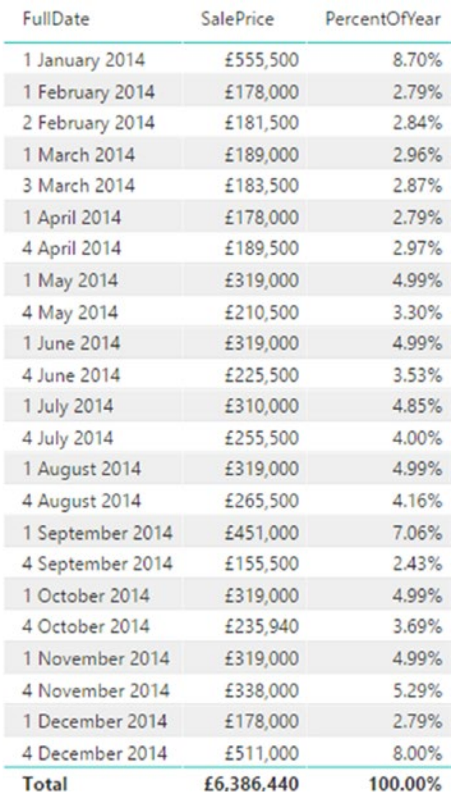
- The *sales total for the time element* that will be used in a visualization. This could be the day, week, month, or quarter, for instance.
- The *total sales for the entire year* that will be used in a visualization. This has to be calculated independently of the actual date element that will be displayed. Consequently, the `CALCULATE()` function is used to extend the aggregation—the `SUM()` in this case—to the whole year. This is done by setting a range of dates as the filter for the aggregation. The date range is set using the `DATESBETWEEN()` function. This requires a lower threshold (defined by the `STARTOFYEAR()` function) and a higher threshold (defined by the `ENDOFYEAR()` function). This way, the date range runs from the first to the last days of the year.

Finally, the calculation divides the specified time span's total by the total for the year; the percentage for that sales period is displayed.

So that you can see the outcome of your formula, you could create a table that contains the following three fields:

- FullDate
- SalePrice
- PercentOfYear

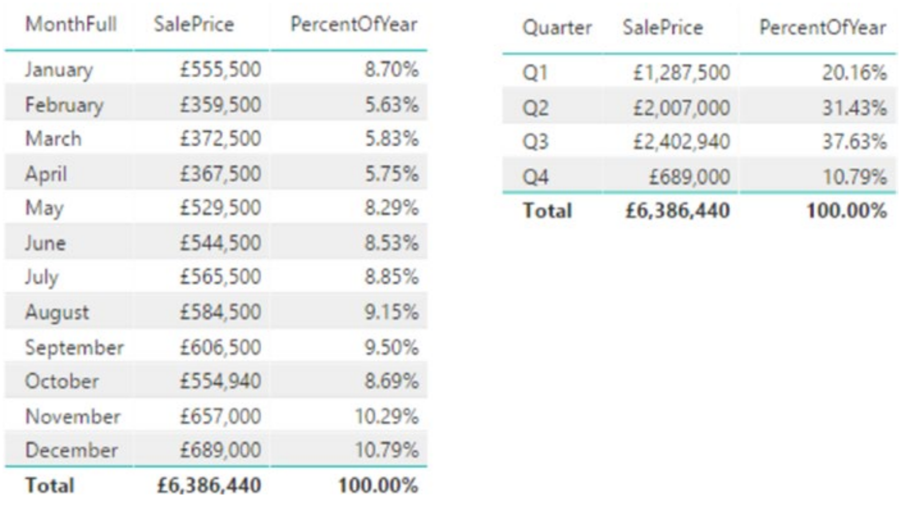
You should see something like Figure 13-10 if you have applied a page-level filter to restrict the FullYear field to 2014, as described in the previous example.



FullDate	SalePrice	PercentOfYear
1 January 2014	£555,500	8.70%
1 February 2014	£178,000	2.79%
2 February 2014	£181,500	2.84%
1 March 2014	£189,000	2.96%
3 March 2014	£183,500	2.87%
1 April 2014	£178,000	2.79%
4 April 2014	£189,500	2.97%
1 May 2014	£319,000	4.99%
4 May 2014	£210,500	3.30%
1 June 2014	£319,000	4.99%
4 June 2014	£225,500	3.53%
1 July 2014	£310,000	4.85%
4 July 2014	£255,500	4.00%
1 August 2014	£319,000	4.99%
4 August 2014	£265,500	4.16%
1 September 2014	£451,000	7.06%
4 September 2014	£155,500	2.43%
1 October 2014	£319,000	4.99%
4 October 2014	£235,940	3.69%
1 November 2014	£319,000	4.99%
4 November 2014	£338,000	5.29%
1 December 2014	£178,000	2.79%
4 December 2014	£511,000	8.00%
Total	**£6,386,440**	**100.00%**

Figure 13-10. *Displaying the sales per day as a percentage of the yearly sales*

You may be thinking that this was a lot of work just to get a percentage figure. Well, perhaps it is at first. Yet you can now capitalize on your effort and see how time intelligence is worth the effort. For instance, if you replace the DateKey field with the MonthFull field in the table shown in Figure 13-9, you instantly see the sales percentages for each month. The same applies if you replace MonthFull with Quarter. You can see both these outputs in Figure 13-11. This is because you have created an extremely supple and fluid formula that can adapt to any time segment. The formula says to “Take the time segment (day, month, quarter, or year) and then find the date range for the whole year. Use this to calculate the total for the year, and then divide the figure for the time span by this total to display the percentage.”



MonthFull	SalePrice	PercentOfYear
January	£555,500	8.70%
February	£359,500	5.63%
March	£372,500	5.83%
April	£367,500	5.75%
May	£529,500	8.29%
June	£544,500	8.53%
July	£565,500	8.85%
August	£584,500	9.15%
September	£606,500	9.50%
October	£554,940	8.69%
November	£657,000	10.29%
December	£689,000	10.79%
Total	**£6,386,440**	**100.00%**

Quarter	SalePrice	PercentOfYear
Q1	£1,287,500	20.16%
Q2	£2,007,000	31.43%
Q3	£2,402,940	37.63%
Q4	£689,000	10.79%
Total	**£6,386,440**	**100.00%**

Figure 13-11. *Reusing a DAX time formula*

Comparing a Metric with the Result from a Range of Dates

Let's push the time-based data analysis a little further and imagine that you need to look at how figures have evolved compared to a specific time interval in the past. By this I mean that you want to see how sales for the current month have fluctuated compared to, say, the previous month. The following formula (that you can see in full in Step 14) shows you how to do just this. Once you understand the principles that this technique can be extended to, compare the data over many different time spans.

1. In the Data View, select the InvoiceLines table and click New Measure.
2. Replace Measure with **PercentOfPreviousMonth**.
3. To the right of the equals sign, type or select **SUM(**.
4. Type a left bracket and select the [SalePrice] field from the popup list of available fields.
5. Enter a right parenthesis to end the `SUM()` function.
6. Enter a forward slash to indicate division.
7. Enter (or type and select) **CALCULATE(SUM(**.
8. Select the [SalePrice] field from the popup list of available fields.
9. Enter a right parenthesis to end the `SUM()` function.
10. Enter a comma. This tells the calculate function that you have chosen the aggregation that you want; it will now apply a filter.
11. Enter (or type and select) **PREVIOUSMONTH(**.
12. Start typing the name of the date table (**DateDimension**) and then select the DateKey field from the popup. This tells the `PREVIOUSMONTH()` function the time dimension field that should be used as its parameter.
13. Enter a right parenthesis to end the `PREVIOUSMONTH()` function.

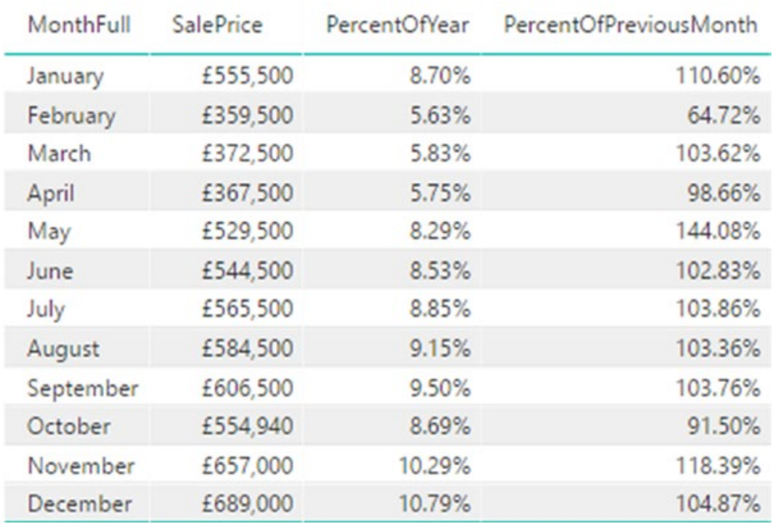
14. Enter a right parenthesis to end the `CALCULATE()` function. The formula should look like this:

```
PercentOfPreviousMonth = SUM([SalePrice]) / CALCULATE(SUM([SalePrice]),
PREVIOUSMONTH(DateDimension[DateKey]))
```

15. Press Enter or click the tick mark icon to complete the measure definition.

16. Format the measure as a percentage.

Figure 13-12 shows you the output that can be obtained by using the MonthFull field to give the month of a year (2014 in this example) alongside the `PREVIOUSMONTH()` function. To stress that time intelligence can adapt to different circumstances, you can also see the `PERCENTOFYEAR` function from the previous example—only it has automatically returned the percentage for the month this time.



MonthFull	SalePrice	PercentOfYear	PercentOfPreviousMonth
January	£555,500	8.70%	110.60%
February	£359,500	5.63%	64.72%
March	£372,500	5.83%	103.62%
April	£367,500	5.75%	98.66%
May	£529,500	8.29%	144.08%
June	£544,500	8.53%	102.83%
July	£565,500	8.85%	103.86%
August	£584,500	9.15%	103.36%
September	£606,500	9.50%	103.76%
October	£554,940	8.69%	91.50%
November	£657,000	10.29%	118.39%
December	£689,000	10.79%	104.87%

***Figure 13-12.** Comparing values with a previous period of time*

Once again, the "magic" in the formula was the use of the `CALCULATE()` function. Yet again, this function required you to enter two elements:

- An aggregation to carry out (the sum of the Sale Price in this case).
- A filter to apply (the previous month in this example). Once again, the function uses the date key from the date table to apply the time intelligence.

With these two parameters in place, DAX was able to take the time element used in the visualization (the month) and say, "Give me the aggregation for the previous month."

Comparisons with Previous Time Periods

While the various DAX functions that return a "previous" time span are extremely useful, it could be argued that they are a little rigid for some types of calculation. After all, comparisons to the previous month typically require you to display data by month. So DAX has alternative methods of comparing data over time that do not require you to specify exactly which time element (day, month, quarter, or year) you want to compare with. Instead, you can merely say that you want to go back a defined period in time, and then depending on the choice of time element that you use in a visualization, DAX automatically calculates the correct figure for comparison.

Calculating Sales for the Previous Year

As a first example, let's imagine that you want to compare current sales with sales for the current period—be it a day, week, month, quarter, or year. There might be several ways of doing this, but there is one fairly simple approach that returns the total car sales for the same time span for the previous year. Once the principle is clear, I will show you how to extend this to calculate the following:

- The average car sales price for the previous year
- The number of cars sold in the previous quarter

Initially, you should carry out the following steps to calculate sales for the previous year (see step 14 for the complete formula):

1. In the Data View, select the Invoices table and click New Measure.
2. Replace Measure with **PreviousYearSales**.
3. To the right of the equals sign, enter (or type and select) **CALCULATE(**.
4. Enter (or select) **SUM(**.
5. Select the InvoiceLines[SalePrice] field.
6. Enter a right parenthesis to close the `SUM()` function.
7. Enter a comma.
8. Enter (or select) **DATEADD(**.
9. Select the DateDimension[DateKey] field. This indicates to DAX the correct date table and date key field.
10. Enter a comma.
11. Enter **-1**. This will cause the time comparison to apply to a *previous* period of time.
12. Enter a comma.
13. Enter (or select) **YEAR**. This indicates to the `DATEADD()` function that it wants to compare figures with the previous year.

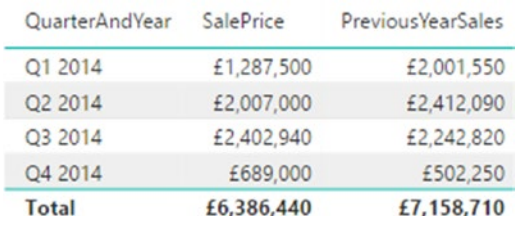
14. Enter two right parentheses. The first one closes the `DATEADD()` function; the second one closes the `CALCULATE()` function. The formula will look like this:

```
PreviousYearSales = CALCULATE(SUM(InvoiceLines[SalePrice]),
DATEADD(DateDimension[DateKey],-1,YEAR))
```

15. Press Enter or click the tick icon in the formula bar to finish the measure.

16. Format the measure appropriately.

Figure 13-13 shows how the sale price (at the quarter level in this instance) is compared to last year's sale price when you add the PreviousYearSales measure to a table. In this example, the table shows only sales for 2014. Consequently, the previous year's sales are those for 2013. What is so impressive is that if you filter the table on another year (2015 for example), you will see that the previous year's figures are now those for 2014.



QuarterAndYear	SalePrice	PreviousYearSales
Q1 2014	£1,287,500	£2,001,550
Q2 2014	£2,007,000	£2,412,090
Q3 2014	£2,402,940	£2,242,820
Q4 2014	£689,000	£502,250
Total	**£6,386,440**	**£7,158,710**

Figure 13-13. *Calculating metrics for a previous year*

The formula sums or averages the data in a column, but only for the previous year, compared to the date field for each row. (This is the InvoiceDate in the sample data, because the date field is linked to the date table in the sample data model.) Note that you do not use the InvoiceDate field in these formulas. This is because it is the field that is linked to the DateKey field of the date table. Power BI Desktop knows which field to use in the Stock table as the basis for time comparisons. To labor the point, it was essential to create a coherent and complete data model to make time intelligence work perfectly.

In this example, you set the "time shift" to a negative value, so that DAX would go back one year in time. You can also use positive numbers if you are looking at data from a past viewpoint and want to compare with data from later dates.

■ **Tip** The `DATEADD()` function lets you replace `YEAR` with `MONTH` or `DAY` if you need to compare with data from days or months previously.

Once you have mastered this technique, you can extend and enhance a formula such as this to provide a multitude of metrics that will adapt to the time-based filters on tables and charts. As a second example, try creating the following measure. It will calculate the average sale price for the selected period(s) in the preceding year. I hope that by now you have become used to writing DAX formulas, so I will not explain how to create this formula, step by step. Instead, I will let you create it unaided. This should provide you with some good practice in creating DAX formulas by yourself.

```
AverageSalePricePreviousYear = CALCULATE(
                                          AVERAGE(InvoiceLines[SalePrice]),
                                          DATEADD(DateDimension[DateKey],-1,YEAR)
                                         )
```

■ **Note** I have formatted the code in this example for greater readability on the page and hopefully to make the logic of the functions more comprehensible. You might not be able to use the code formatted like this in Power BI Desktop without simplifying the presentation.

Alternatively, perhaps you want to see the number of sales for the previous quarter relative to the date that is used to filter a visualization. This formula would be as follows:

```
NumberOfSalesPreviousQtr = CALCULATE(
                                         COUNT(InvoiceLines[InvoiceID]),
                                         DATEADD(DateDimension[DateKey],
                                                   -1,QUARTER)
                                        )
```

Now that you have seen the principle, you are free to adapt it to your specific requirements. You can use any of the DAX aggregation functions that were described in the previous chapter. You can mix these with the four interval types (Year, Quarter, Month, and Day) that the DATEADD() function uses to deliver a truly wide-ranging set of time comparison metrics that automatically adapt to the time span of your Power BI Desktop visualization.

Comparison with a Parallel Period in Time

Looking at metrics from the past can be a key indicator of how a business is progressing. Clearly identifying the extent (or lack of) progress is even more telling. There are several ways that you can perform these types of calculation in DAX. In this section, you will see a couple of techniques that you may find useful.

Comparing Data from Previous Years

The following two measures (YearOnYearDelta and YearOnYearDeltaPercent) calculate the increase or decrease in sales compared to a previous year and also that change expressed as a percentage. These measures extend the logic of the last few formulas using functions that you have already met. I will presume that after two chapters on DAX, you do not really need a step-by-step explanation on how to enter a formula. So I will only present and then explain the code from now on. The following is the code to add YearOnYearDelta and YearOnYearDeltaPercent as new measures to the Invoices table:

```
YearOnYearDelta=IF(
                    ISBLANK(
                            SUM(InvoiceLines[SalePrice])
                           ),
                    BLANK(),
                    IF(
                       ISBLANK(
                               CALCULATE(
                                 SUM(InvoiceLines[SalePrice]),
                                 DATEADD(DateDimension[DateKey],
                                          -1, YEAR)
                                )
                       ),
                  BLANK(),
                  SUM(InvoiceLines [SalePrice])
                  - CALCULATE(
                              SUM(InvoiceLines[SalePrice]),
                              DATEADD(DateDimension[DateKey], -1, YEAR)
                             )
                  )
                )
```

```
YearOnYearDeltaPercent=IF(
                              ISBLANK(
                                      SUM(InvoiceLines[SalePrice])
                                      ),
                              BLANK(),
                              IF(
                                 ISBLANK(
                                         CALCULATE(
                                                   SUM(InvoiceLines[SalePrice]),
                                                  DATEADD(DateDimension[DateKey],
                                                           -1, YEAR)
                                                  )
                                        ),
                                         BLANK(),
                                         (
                                          SUM(InvoiceLines[SalePrice])
                                          - CALCULATE(
                                            SUM(InvoiceLines[SalePrice]),
                                                DATEADD(DateDimension[DateKey],
                                                         -1, YEAR)
                                                   )
                                         )
                                         /CALCULATE(
                                                   SUM(InvoiceLines[SalePrice]),
                                                 DATEADD(DateDimension[DateKey],
                                                          -1, YEAR)
                                                  )
                                         )
                                 )
```

These two formulas are a lot easier than they look, believe me.

The formula for YearOnYearDelta is this:

```
SUM(Stock[SalePrice])
- CALCULATE(SUM(InvoiceLines[SalePrice]), DATEADD(DateDimension[DateKey], -1, YEAR))
```

All the code says is "Subtract last year's sales from this year's sales." Everything else is wrapper code to prevent a calculation if either this year's or last year's data is zero.

Equally, this is the core code for the YearOnYearDeltaPercent formula:

```
SUM(InvoiceLines[SalePrice]) - CALCULATE(SUM(InvoiceLines[SalePrice]),DATEADD(DateDimension
[DateKey], -1, YEAR)) / CALCULATE(SUM(InvoiceLines[SalePrice]),DATEADD(DateDimension[DateKey],
-1, YEAR))
```

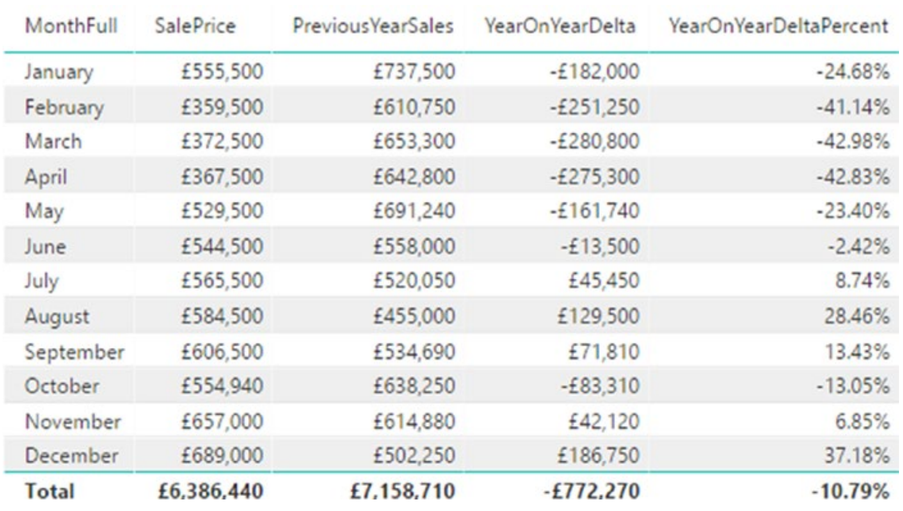
In other words, "Subtract last year's sales from this year's sales and divide by last year's sales." Everything else in the complete formula that is given in full earlier exists to prevent divide-by-zero errors or unwanted results for the first year where there is no previous year's data!

The logic "wrapper" around the core formula uses two new functions:

- `ISBLANK()`: This function tests if a calculation returns nothing and allows you to specify what to do if this happens. This is a bit like an `IF()` function that only tests for blank data.
- `BLANK()`: Returns a blank (or Null). This is useful for overriding unwanted results and replacing them with a blank.

Using these functions lets you handle the case where there is no data for a previous period of time. Because you are using the `ISBLANK()` function to test for inexistent data, you are able to replace any missing data with a `BLANK()`—rather than letting Power BI Desktop display an unsightly error.

We have seen a couple of fairly complex formulas in a short section. So I think that it is a good idea to see how they look when you apply them. In this case, I will use a Power BI Desktop table to show the results, as shown in Figure 13-14. As you can see, the appropriate formats have been applied to each metric to enhance readability.



MonthFull	SalePrice	PreviousYearSales	YearOnYearDelta	YearOnYearDeltaPercent
January	£555,500	£737,500	-£182,000	-24.68%
February	£359,500	£610,750	-£251,250	-41.14%
March	£372,500	£653,300	-£280,800	-42.98%
April	£367,500	£642,800	-£275,300	-42.83%
May	£529,500	£691,240	-£161,740	-23.40%
June	£544,500	£558,000	-£13,500	-2.42%
July	£565,500	£520,050	£45,450	8.74%
August	£584,500	£455,000	£129,500	28.46%
September	£606,500	£534,690	£71,810	13.43%
October	£554,940	£638,250	-£83,310	-13.05%
November	£657,000	£614,880	£42,120	6.85%
December	£689,000	£502,250	£186,750	37.18%
Total	**£6,386,440**	**£7,158,710**	**-£772,270**	**-10.79%**

Figure 13-14. *Power BI Desktop output for year-on-year comparisons*

Once again, these formulas only scratch the surface of the myriad possibilities that DAX has on offer. However, you can adapt them to create comparisons by quarter, month, or day if you prefer simply by changing the specified time interval from `YEAR` to `QUARTER`, `MONTH`, or `DAY`.

Comparing with the Same Date Period from a Different Quarter, Month, or Year

In the last couple of sections, you saw various ways to compare data from a prior year or month. DAX offers one alternative method for these kinds of calculations that can be both easy to implement and extremely powerful. Moreover, it can serve as the basis for comparison with years, quarters, or months. This is the `PARALLELPERIOD()` function. Suppose that you want to use it to find average sales for the previous quarter.

Because this function is fairly similar to the `DATEADD()` function that you saw previously, I will not explain it step by step; instead, I prefer to give you three examples of measures that you can add to the InvoiceLines table directly.

Here are the sales for the preceding month:

```
SalesPrevMth = CALCULATE(SUM(InvoiceLines[SalePrice]), PARALLELPERIOD(DateDimension
[DateKey],-1,MONTH))
```

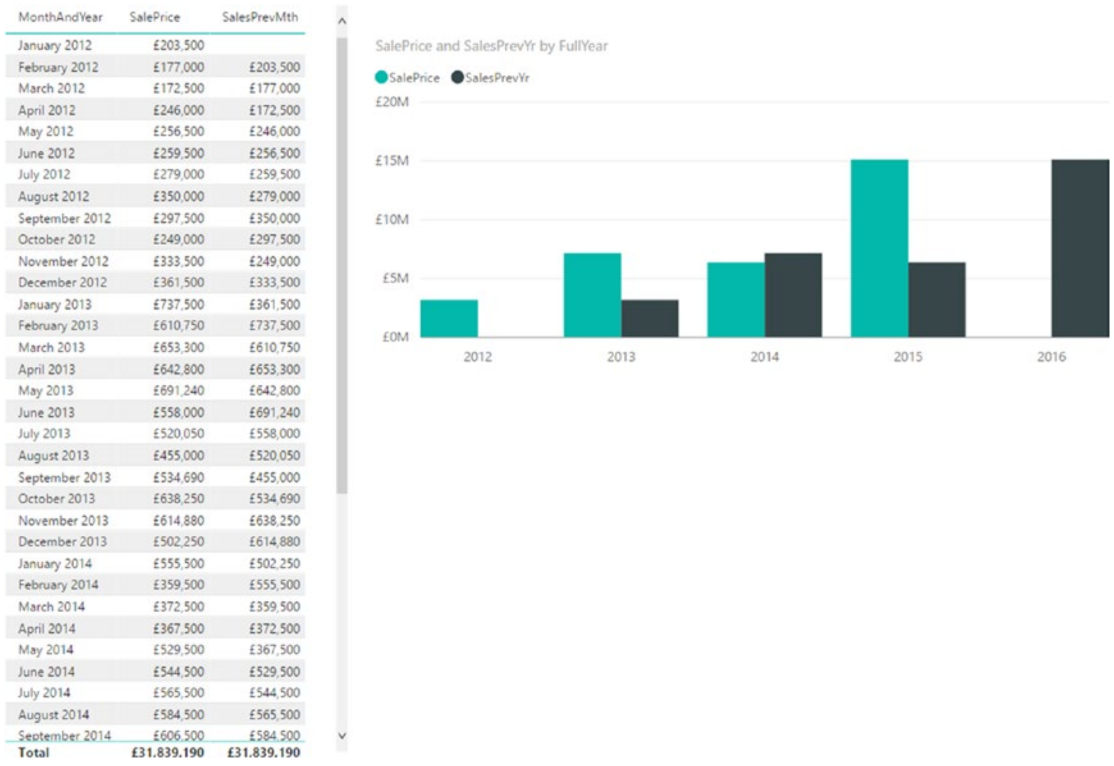
Here are the sales for the preceding quarter:

```
SalesPrevQtr = CALCULATE(SUM(InvoiceLines[SalePrice]), PARALLELPERIOD(DateDimension
[DateKey],-1,QUARTER))
```

Here are the sales for the preceding year:

```
SalesPrevYr = CALCULATE(SUM(InvoiceLines[SalePrice]), PARALLELPERIOD(DateDimension
[DateKey],-1,YEAR))
```

You see two examples of these functions displayed as simple visualizations in Figure 13-15.



MonthAndYear	SalePrice	SalesPrevMth
January 2012	£203,500	
February 2012	£177,000	£203,500
March 2012	£172,500	£177,000
April 2012	£246,000	£172,500
May 2012	£256,500	£246,000
June 2012	£259,500	£256,500
July 2012	£279,000	£259,500
August 2012	£350,000	£279,000
September 2012	£297,500	£350,000
October 2012	£249,000	£297,500
November 2012	£333,500	£249,000
December 2012	£361,500	£333,500
January 2013	£737,500	£361,500
February 2013	£610,750	£737,500
March 2013	£653,300	£610,750
April 2013	£642,800	£653,300
May 2013	£691,240	£642,800
June 2013	£558,000	£691,240
July 2013	£520,050	£558,000
August 2013	£455,000	£520,050
September 2013	£534,690	£455,000
October 2013	£638,250	£534,690
November 2013	£614,880	£638,250
December 2013	£502,250	£614,880
January 2014	£555,500	£502,250
February 2014	£359,500	£555,500
March 2014	£372,500	£359,500
April 2014	£367,500	£372,500
May 2014	£529,500	£367,500
June 2014	£544,500	£529,500
July 2014	£565,500	£544,500
August 2014	£584,500	£565,500
September 2014	£606,500	£584,500
Total	**£31,839,190**	**£31,839,190**

SalePrice and SalesPrevYr by FullYear
● SalePrice ● SalesPrevYr
£20M
£15M
£10M
£5M
£0M
2012
2013
2014
2015
2016

Figure 13-15. *Using the PARALLELPERIOD() function*

■ **Note** You need to be aware that the `PARALLELPERIOD()` function compares the current data, not just up to the same data in the previous period (be it a year, a quarter, or a month), but for the entire previous time period.

There are further DAX functions that you can also use when comparing data over time. These are outlined in Table 13-7.

Table 13-7. *DAX Date and Time Formulas to Compare Values over Time*

Formula	Description	Example
`PARALLELPERIOD()`	Finds dates from a "parallel" timeframe defined by a certain number of set intervals. The first parameter is the source data; the second is the number of years, quarters, months, or days; and the third is the definition of the interval: years, quarters, months, or days. A positive number of intervals looks forward in time and a negative number goes backward in time.	`CALCULATE(SUM(InvoiceLines[Sale Price]), PARALLELPERIOD(Date Dimension[DateKey],-1,MONTH))`
`SAMEPERIODLASTYEAR()`	Finds the date(s) for the same time range one year before.	`CALCULATE(SUM(InvoiceLines[Sale Price]), SAMEPERIODLASTYEAR (DateDimension[DateKey))`
`DATEADD()`	Used to return data from a past or future period in time compared to a specified date. The date difference can be in years, quarters, months, or days.	`CALCULATE(SUM(InvoiceLines [SalePrice]), DATEADD (DateDimension[DateKey], -1, YEAR))`
`DATESBETWEEN()`	Calculates a list of dates between two dates. The first parameter is the start date and the second parameter is the end date.	`DATESBETWEEN(Stock[PurchaseDate], Invoices[SaleDate])`
`DATESINPERIOD()`	Calculates a list of dates beginning with a start date for a specified period.	`DATESINPERIOD(DateDimension [DateKey], DATE(2015,06,30),-90,day))`

Rolling Aggregations over a Period of Time

We are now getting into the arena of more complex DAX formulas. So, since returning the rolling sum (or average) of a specified period to date necessitates several DAX functions and some in-depth nesting of these functions, I will take this as an example of a more complicated DAX formula. I will begin by outlining some of the functions that are used to deliver a result that is reliable and efficient:

- `DATESBETWEEN()`: Lets you select a range of dates. The three parameters are the date key field from the date table, then the starting date, then the ending date.
- `FIRSTDATE()`: Allows you to get the first date from a range. Since we are using this momentarily to go back a defined number of months, it will get the first day of the month.
- `LASTDATE()`: Allows you to get the last date from a range. Since we are using this momentarily to go back a defined number of months, it will get the last day of the month.

You can now create two measures (3MonthsToDate and Previous3Months) using some fairly sophisticated logic to ensure that only blank cells are returned if there is no previous year's data using the following formulas:

```
3MonthsToDate=IF(
                  ISBLANK(SUM(InvoiceLines[SalePrice])),
                          BLANK(),
                          CALCULATE(
                                   SUM(InvoiceLines[SalePrice]),
                                   DATESINPERIOD(
                                                 DateDimension[DateKey],
                                                 LASTDATE(DateDimension[DateKey]),-3,MONTH
                                                )
                                   )
                  )

Previous3Months=IF(
                    ISBLANK(
                           CALCULATE(
                                     SUM(InvoiceLines[SalePrice]),
                                     DATEADD(DateDimension[DateKey],-1,MONTH)
                                     )
                           ),
                           BLANK(),
                           CALCULATE(
                                     SUM(InvoiceLines[SalePrice]),
                                     DATESBETWEEN(
                                                DateDimension[DateKey],
                                                FIRSTDATE(DATEADD(DateDimension[DateKey],
                                                  -4,MONTH)),
                                                LASTDATE(DATEADD(DateDimension[DateKey],
                                                  -1,MONTH))
                                     )
                           )
                    )
```

The 3MonthsToDate formula essentially evaluates the code that is in boldface. This says, "Add up the sales for a time period ranging from three months ago to now," using the InvoiceDate field as the date to evaluate. The `IF()` function detects if there are sales for the current date, and if there are none (`ISBLANK`), then the calculation is not attempted, and a `BLANK` is returned.

The Previous3Months formula is pretty similar, except that the time span uses the `DATESBETWEEN()` function to set a range of dates—from the first day of the month four months ago to the last date in the preceding month.

DAX contains a couple of functions that you can use to return a specific date when aggregating data over time. These are shown briefly in Table 13-8.

Table 13-8. *DAX Date and Time Formulas to Return a Date*

Formula	Description	Example
`FIRSTDATE()`	Finds the first date that an event took place.	`FIRSTDATE(Invoices[InvoiceDate])`
`LASTDATE()`	Finds the last date that an event took place.	`LASTDATE(Invoices[InvoiceDate])`

Conclusion

This chapter has taken you on a concise tour of some of the ways that you can use DAX in Power BI Desktop to extract meaning from your data by analyzing its evolution over time.

First, you saw how to extract date and time elements from columns that contain dates. The new columns that you create based on dates can then be used to filter or group data in your visualizations. This way you can provide a daily, weekly, monthly, quarterly, and yearly breakdown of your source data. These date elements can also help you to filter the datasets that you are using by dates, date elements, and date ranges.

Then you saw how to prepare the dataset for time intelligence by adding a date table and joining this table to the other tables in the data model. Finally, you saw how to start adding formulas to the dataset to prepare all the time-based metrics that your Power BI Desktop reports could need. This can include analyzing sales to date or comparing data from previous time periods with current data.

Explaining all the possibilities of DAX would take an entire book, so all I wanted to do in this and the previous two chapters was explain how you can use core DAX functions in a handful of useful calculations. I sincerely hope that this brief overview helps you on your road to mastery of DAX and that you are able to apply these formulas to deliver stunning visualizations using Power BI Desktop.

Your data is now ready for output. It can be used as the basis for multiple dashboards and reports. This is the subject of the next six chapters.

CHAPTER 14

Table Visuals

You are now entering the final straight on your race to deliver clear, powerful, and visually compelling analytics. The time has come to transform data into attention-grabbing dashboards that capture the imagination of your audience. In this chapter, you will start learning how to use Power BI Desktop to

- Delve deeply into data and produce valuable information from the mass of facts and figures available.
- Create interactive views of your insights, where you can test your analyses quickly and easily.
- Enhance the presentation of your results to grab your audience's attention.

This chapter and the next one take you through the process of creating text-based visualizations (or visuals if you want to call them that) in Power BI Desktop. In these two chapters you will learn how to create and enhance

- Tables
- Matrices
- Cards
- Multirow cards

What these types of visuals all have in common is that they are designed to use *text* to convey information rather than using the more graphical display types, which you will discover in Chapters 16 through 19. Text-based visuals are essential when it comes to

- Presenting lists of information
- Displaying cross-sections of key data
- Focusing the audience's attention on a single essential figure
- Delivering a clear overview of a few key metrics

As you are on the first step of the learning curve, this chapter will focus solely on all that you can do with *tables* of data in Power BI Desktop. This will allow you to familiarize yourself with the core techniques that you can then apply to other text-based visuals—as well as many, if not all, of the vast range of graphical and geographical visualizations that Power BI Desktop lets you use to deliver your analyses.

The techniques that you will learn in this chapter use a Power BI Desktop file that results from the application of much of what you saw in the last four chapters. This file is called C:\PowerBiDesktopSamples\CH14\CarSalesDataForReports.pbix. If you need a ready-prepared version of this file, then it is available on the Apress web site as part of the downloadable material that accompanies this book.

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_14

Power BI Desktop Dashboards

So far in this book, you have used two of the three core interface modes of Power BI Desktop: Data View and Relationship View. Now that you are moving on to create reports and dashboards, you will use the final part of the trilogy: Report View.

You always use Report View to build your dashboards (or reports, if you prefer). It is here that you add and configure visuals that use the data that you have already loaded, cleansed, structured, and enhanced in the data model.

Switching to Report View

Although Report View is the default mode when you open Power BI Desktop, you need to switch back to it if you have spent any time working on the underlying data model. The following explains how to activate Report View:

1. Click the Report View icon at the top left of the Power BI Desktop screen. This icon is shown in Figure 14-1.

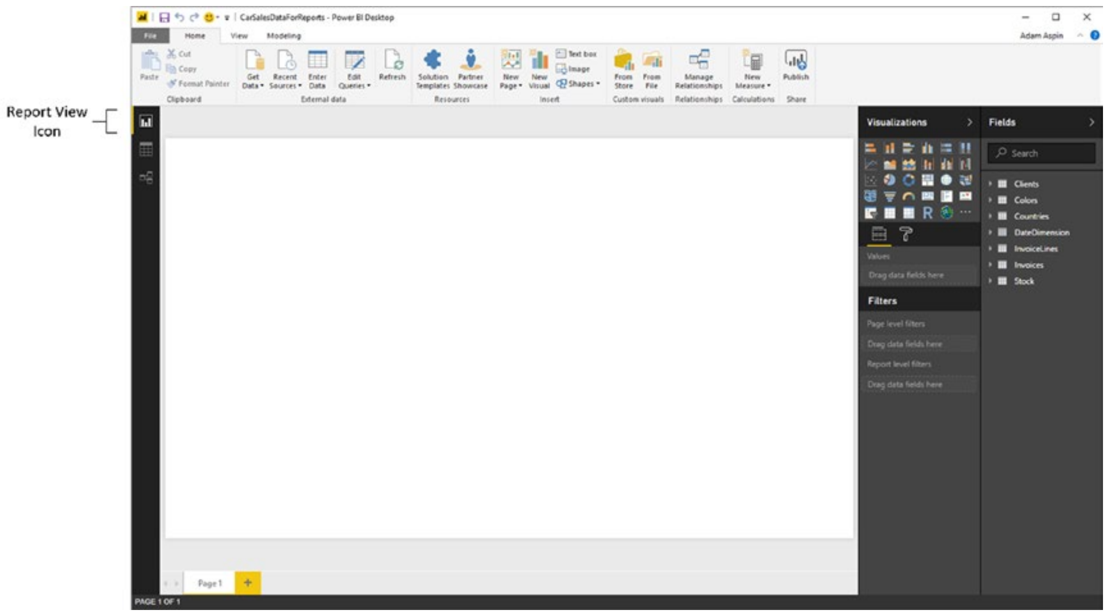


Figure 14-1. The Power BI Desktop Report View icon

■ **Note** You learn how to create and modify complex multipage reports in Chapter 22.

Working with Tables

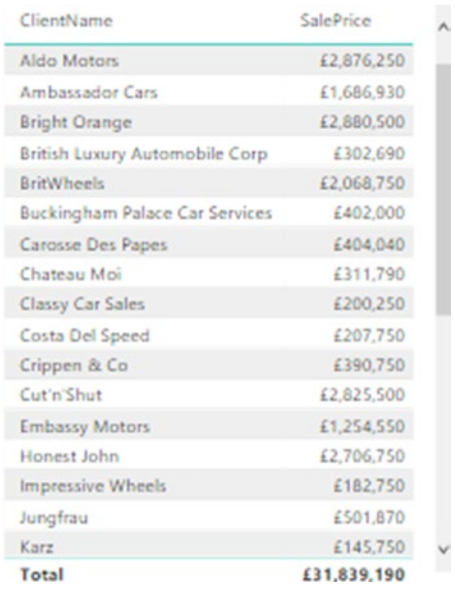
Tables are probably the simplest and most elementary way of displaying data. However, this simplicity should not detract from their usefulness. Indeed, for Power BI Desktop, the humble table is the default visualization. Moreover, and as you will see in this chapter, the table is the default presentation style for non-geographical data.

I realize, of course, that it may seem contradictory to spend time on a subject that is generally described as intuitive. In answer to this I can only say that, while getting up and running is easy, attaining an in-depth understanding of all of the potential of this powerful tool does require some explanation. Consequently, the approach that I am taking is to go through all the possibilities of each item as thoroughly as possible. So feel free to jump ahead (and back) if you don't need all the detail just yet-or if you want to skim over the intricacies in order to get a high-level overview of the subject.

Creating a Basic Table

The following example introduces you to tables in Power BI Desktop by creating an initial table that will display the list of clients and their total sales:

1. Open the C:\PowerBiDesktopSamples\CH14\CarSalesDataForReports.pbix file from the downloadable samples.
2. In the Fields list, expand the Clients table.
3. Drag the ClientName field on to the dashboard canvas. A table displaying all the clients of Brilliant British Cars will appear.
4. Leave this new table selected and, in the Fields list, expand the InvoiceLines table.
5. Click the check box to the left of the SalePrice field. This will add the cumulative sales per client to the table. It should look like Figure 14-2.



ClientName	SalePrice
Aldo Motors	£2,876,250
Ambassador Cars	£1,686,930
Bright Orange	£2,880,500
British Luxury Automobile Corp	£302,690
BritWheels	£2,068,750
Buckingham Palace Car Services	£402,000
Carosse Des Papes	£404,040
Chateau Moi	£311,790
Classy Car Sales	£200,250
Costa Del Speed	£207,750
Crippen & Co	£390,750
Cut'n'Shut	£2,825,500
Embassy Motors	£1,254,550
Honest John	£2,706,750
Impressive Wheels	£182,750
Jungfrau	£501,870
Karz	£145,750
Total	**£31,839,190**

Figure 14-2. *A basic table of sales per client*

This is a very tiny table. In the real world, you could be looking at tables that contain thousands, or tens of thousands (or even millions), of records. Power BI Desktop accelerates the display of large datasets by only loading the data that is required as you scroll down through a list, and starts with the initial few records in the data. So you might see the scroll bar advance somewhat slowly as you progress downward through a large table.

If a table contains fields that you have used in a visual, then the table name appears in yellow in the Fields list when you select the visualization. You can always see which fields have been selected for a table by expanding the table in the Fields list. The fields used are instantly displayed in both the Fields list (as selected fields) and in the field well of the Visualizations pane. To get you used to this idea, see Figure 14-3, which shows both these elements for the table that you just created.

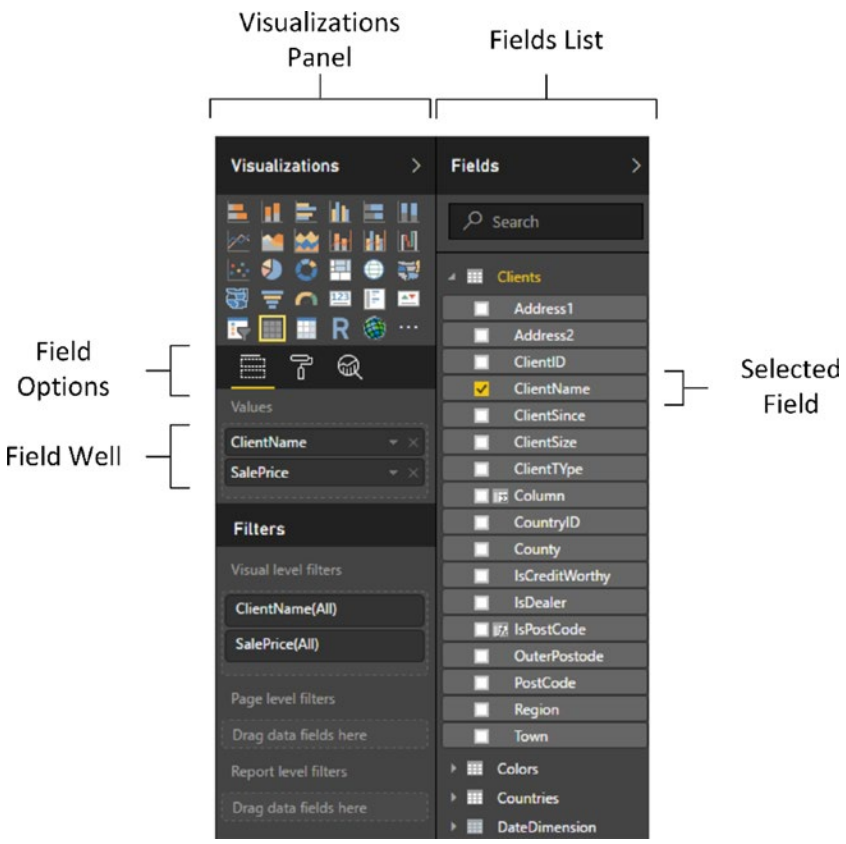


Figure 14-3. *The fields used in a visual*

As befits such a polished product, Power BI Desktop does not limit you to just one way of adding fields to a table. The following are ways in which you can add fields to a table:

- By dragging the field name into the field well in the Visualizations pane
- By selecting a field (which means checking the check box to the left of the field) in the Fields list—assuming that the table is already selected
- By dragging a field on to an existing visualization

You can add further fields to an existing table at any time. The key thing to remember (if you are using any of the three techniques described) is that you must *select the table that you want to modify first*. This is as simple as clicking inside it. After you click, you instantly see that the table is active because tiny handles appear at the corners of the table as well as in the middle of each side of the table.

■ **Note** If you do not select an existing table before adding a field, Power BI Desktop will create a new table using the field that you are attempting to add to a table.

To create another table, all you have to do is click outside any existing visuals in the Power BI Desktop report canvas and begin selecting fields as described earlier. A new table is created as a result. Power BI Desktop always tries to create new tables in an empty part of the canvas. You will see how to rearrange this default presentation shortly.

There is another way to create a table. This is to click the report canvas to ensure that you have deselected any existing visuals, and then click the table icon in the palette of visualizations at the top of the Visualizations pane. This will create a blank table on the report canvas. You can see this icon in Figure 14-4, along with a sample empty table.

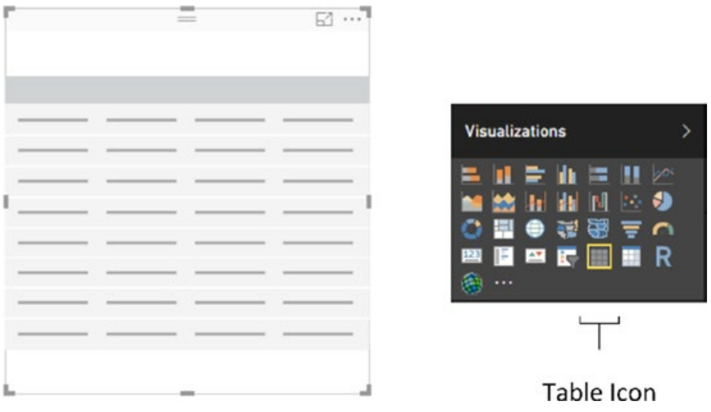


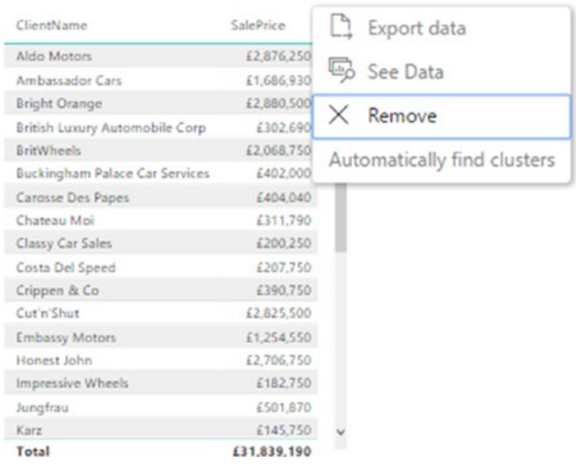
Figure 14-4. *The table icon*

Deleting a Table

Suppose that you no longer need a table in a Power BI Desktop report. Well, deleting it is simple:

1. Select the table. You can do this by clicking anywhere inside the table. The table borders will appear, even if you move the mouse pointer away from the table.
2. Press the Delete key.

Another way to select a table is to click the options button (the ellipses) that appears at the top right of any visual once it is selected. The options menu for the visual (the table in this example) will appear. Clicking the Remove option will delete the table. You can see the context menu in Figure 14-5.



ClientName	SalePrice
Aldo Motors	£2,876,250
Ambassador Cars	£1,686,930
Bright Orange	£2,880,500
British Luxury Automobile Corp	£302,690
BritWheels	£2,068,750
Buckingham Palace Car Services	£402,000
Carosse Des Papes	£404,040
Chateau Moi	£311,790
Classy Car Sales	£200,250
Costa Del Speed	£207,750
Crippen & Co	£390,750
Cut'n'Shut	£2,825,500
Embassy Motors	£1,254,550
Honest John	£2,706,750
Impressive Wheels	£182,750
Jungfrau	£501,870
Karz	£145,750
Total	**£31,839,190**

Figure 14-5. *The table visualization options menu*

Deleting a table is so easy that you can do it by mistake, so remember that you can restore an accidentally deleted table by pressing Ctrl+Z or using the Undo arrow at the top left of the Power BI Designer window.

Copying a Table

You need to copy tables on many occasions. There could be several reasons for this:

- You are creating a new visual on the Power BI Desktop report and need the table as a basis for the new element, such as a chart, for instance.
- You are copying visuals between reports.
- You want to keep an example of a table and try some fancy tricks on the copy, but you want to keep the old version as a failsafe option.

In any case, all you have to do is

1. Select the table (as described previously).
2. In the Home ribbon, select Copy (or press Ctrl+C).

To paste a copy, click outside any visual in a current or new Power BI Desktop report, and select Paste from the Home ribbon (or press Ctrl+V).

Changing the Table Size and Position

A table can be resized and moved (just like any other visual in a Power BI Desktop report) using the techniques that you saw, briefly, in Chapter 1. Although you have already seen these possibilities, I will explain them here in full for the sake of completeness.

Resizing a Table

Resizing a table is mercifully easy. All you have to do is to click any of the table handles and drag the mouse. Lateral handles will alter the table width; top and bottom handles will change the table's height; corner handles modify both height and width by changing the size of the two sides that touch on the selected corner.

Power BI also has one formatting option that you may find useful when resizing tables. You can lock a table's aspect ratio; that is, the ratio of the height to width. Doing this means that you cannot alter the relative height and width of a table when resizing it.

To lock the aspect ratio:

1. Click the table to select it.
2. In the Visualizations pane, click the Format icon under the collection of available visuals (the small paint roller). You can see this icon in Figure 14-3.
3. In the Lock Aspect section, click Off to turn the lock on, or alternatively, click to the right of the empty circle indicating that the lock is off so that it slides to the right and becomes a filled circle.

Now, when you resize a table using the corner handles, it will keep its height-to-width ratio.

Moving a Table

Moving a table is as easy as placing the pointer over the table so that the edges appear and, once the cursor changes to the hand shape, dragging the table to its new position.

You may prefer to use the keyboard when fine-tuning the position of a visual on the dashboard. The keyboard options for doing this are shown in Table 14-1.

Table 14-1. *Keyboard Shortcuts for Moving Visuals*

Option	Description
Nudge left	Left cursor key
Nudge right	Right cursor key
Nudge up	Up cursor key
Nudge down	Down cursor key
Jump left	Ctrl-Left cursor key
Jump right	Ctrl-Right cursor key
Jump up	Ctrl-Up cursor key
Jump down	Ctrl-Down cursor key

Changing Column Order

If you have built a Power BI Desktop table, you are eventually going to want to modify the order in which the columns appear from left to right. The following explains how to do this:

1. Activate the Fields list (unless it is already displayed).
2. In the Visualizations pane, ensure that the Fields options are displayed (by clicking the small bar chart icon under the collection of available visuals).

3. Click the name of the field that you wish to move.
4. Drag the field vertically to its new position. This can be between existing fields, at the top or at the bottom of the Fields list. A thick yellow line indicates where the field will be positioned.

Figure 14-6 shows how to drag a field from one position to another.

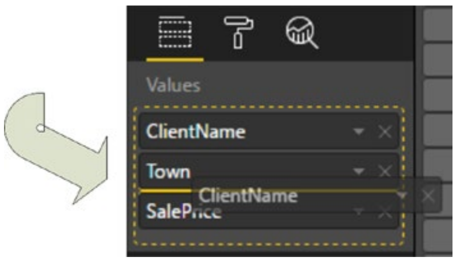


Figure 14-6. *Changing column order by moving fields*

■ **Note** You cannot change the position of a column in a table by dragging it sideways inside the table itself.

Renaming Fields

A recent extension to Power BI Desktop is the ability to rename fields in visuals without having to rename the underlying column in the data table. While this can cause a certain disconnect between the data and its display, it is a technique that can be extremely useful in certain circumstances. To do this:

1. Select the visual containing the data element that you want to rename (I will use the table that you just created in this example).
2. Right-click the field to rename in the field well of the Visualizations pane.
3. Select Rename from the popup menu.
4. Edit the field name—or delete the current name and enter a new name.
5. Press Enter.

The field is now renamed. The new name will appear only in the selected visual, and the data element will keep its original name.

Removing Columns from a Table

Another everyday task in Power BI Desktop is removing columns from a table when necessary. As is the case when rearranging the order of columns, this is not done directly in the table but is carried out using the Fields list. There are, in fact, several ways of removing columns from a table, so I will begin with the way that I think is the fastest and then describe the others.

1. Display the Fields list—unless it is already displayed.
2. Uncheck the field name in the Fields list.

Assuming that a visual is selected, the following are other ways to remove a field:

- In the Visualizations pane, click the cross icon at the right of the field name.
- In the Visualizations pane, click the small triangle at the right of the name in the field well (remember that these are the fields used in the visualization). When the popup menu appears, select Remove Field.

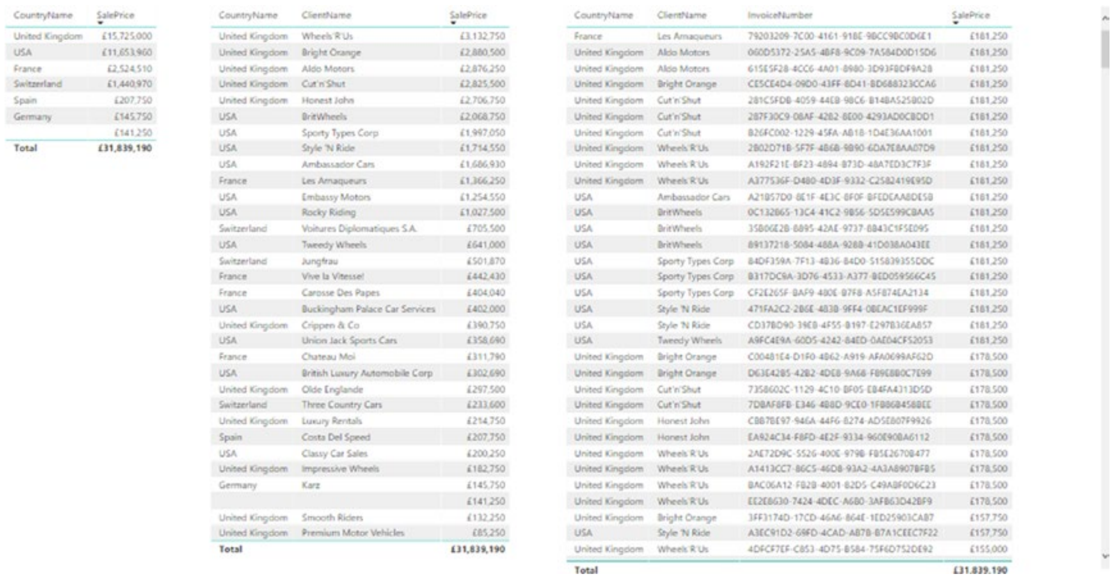
■ **Note** Do *not* click the popup menu icon (the ellipses at the right of the field name) and select Delete in the *Fields list* to remove a field from the table. Doing this deletes the field from the data model.

Table Granularity

A Power BI Desktop table will automatically aggregate data to the lowest available level of grain. Put simply, this means that it is important to select data at the lowest useful level of detail but not to add pointlessly detailed elements.

This is probably easier to understand if I use an example. Suppose you start with a high level of aggregation—the country for instance. If you create a table with CountryName and SalePrice columns, it will give you the total sales by country. If you use the sample data given in the examples for this book (the CarSalesDataForReports.pbix file), this table only contains half a dozen or so lines.

Then add the ClientName column after the CountryName column. When you do this, you obtain a more finely grained set of results, with the aggregate sales for *each client in each country*. If you (finally) add the InvoiceNumber, you get a very detailed level of data. Indeed, adding such a fine level of grain to your table could produce an extremely large number of records-as indicated by the appearance of a vertical scroll bar in this table. These progressive levels of granularity are shown in Figure 14-7.



CountryName	SalePrice
United Kingdom	£15,725,000
USA	£11,653,960
France	£2,524,510
Switzerland	£1,440,970
Spain	£207,750
Germany	£145,750
	£141,250
Total	**£31,839,190**

CountryName	ClientName	SalePrice
United Kingdom	Wheels'R'Us	£3,132,750
United Kingdom	Bright Orange	£2,880,500
United Kingdom	Aldo Motors	£2,876,250
United Kingdom	Cut'n'Shut	£2,825,500
United Kingdom	Honest John	£2,706,750
USA	BritWheels	£2,068,750
USA	Sporty Types Corp	£1,997,050
USA	Style 'N Ride	£1,714,550
USA	Ambassador Cars	£1,686,930
France	Les Amaqueurs	£1,366,250
USA	Embassy Motors	£1,254,550
USA	Rocky Riding	£1,027,500
Switzerland	Voitures Diplomatiques S.A.	£705,500
USA	Tweedy Wheels	£641,000
Switzerland	Jungfrau	£501,870
France	Vive la Vitesse!	£442,430
France	Carosse Des Papes	£404,040
USA	Buckingham Palace Car Services	£402,000
United Kingdom	Crippen & Co	£390,750
USA	Union Jack Sports Cars	£358,690
France	Chateau Moi	£311,790
USA	British Luxury Automobile Corp	£302,690
United Kingdom	Olde Englande	£297,500
Switzerland	Three Country Cars	£233,600
United Kingdom	Luxury Rentals	£214,750
Spain	Costa Del Speed	£207,750
USA	Classy Car Sales	£200,250
United Kingdom	Impressive Wheels	£182,750
Germany	Karz	£145,750
		£141,250
United Kingdom	Smooth Riders	£132,250
United Kingdom	Premium Motor Vehicles	£85,250
Total		**£31,839,190**

CountryName	ClientName	InvoiceNumber	SalePrice
France	Les Amaqueurs	79203209-7C00-4161-918E-9BCC9BC0D6E1	£181,250
United Kingdom	Aldo Motors	060D5372-25A5-4BF8-9C09-7A584D0D15D6	£181,250
United Kingdom	Aldo Motors	615E5F28-4CC6-4A01-8980-3D93FBDF9A28	£181,250
United Kingdom	Bright Orange	CE5CE4D4-09D0-43FF-8D41-BD688323CCA6	£181,250
United Kingdom	Cut'n'Shut	281C5FDB-4059-44EB-98C6-B14BA525B02D	£181,250
United Kingdom	Cut'n'Shut	287F30C9-08AF-4282-8E00-4293AD0CBDD1	£181,250
United Kingdom	Cut'n'Shut	B26FC002-1229-45FA-AB18-1D4E36AA1001	£181,250
United Kingdom	Wheels'R'Us	2B02D71B-5F7F-4B6B-9B90-6DA7E8AA07D9	£181,250
United Kingdom	Wheels'R'Us	A192F21E-BF23-4894-B73D-48A7ED3C7F3F	£181,250
United Kingdom	Wheels'R'Us	A377536F-D480-4D3F-9332-C2582419E95D	£181,250
USA	Ambassador Cars	A21857D0-8E1F-4E3C-8F0F-BFEDEAA8DE58	£181,250
USA	BritWheels	0C132865-13C4-41C2-9B56-5D5E599CBAA5	£181,250
USA	BritWheels	35B06E2B-8895-42AE-9737-8B43C1F5E095	£181,250
USA	BritWheels	89137218-5084-488A-9288-41D038A043EE	£181,250
USA	Sporty Types Corp	84DF359A-7F13-4B36-84D0-515839355DDC	£181,250
USA	Sporty Types Corp	B317DC9A-3D76-4533-A377-BED059566C45	£181,250
USA	Sporty Types Corp	CF2E265F-BAF9-480E-87F8-A5F874EA2134	£181,250
USA	Style 'N Ride	471FA2C2-2B6E-483B-9FF4-0BEAC1EF999F	£181,250
USA	Style 'N Ride	CD37BD90-39EB-4F55-B197-E297B36EA857	£181,250
USA	Tweedy Wheels	A9FC4E9A-60D5-4242-84ED-0AE04CF52053	£181,250
United Kingdom	Bright Orange	C00481E4-D1F0-4B62-A919-AFA0699AF62D	£178,500
United Kingdom	Bright Orange	D63E4285-42B2-4DE8-9A68-FB9EBB0C7E99	£178,500
United Kingdom	Cut'n'Shut	7358602C-1129-4C10-BF05-EB4FA4313D5D	£178,500
United Kingdom	Cut'n'Shut	7DBAF8FB-E346-488D-9CE0-1FB86B458BEE	£178,500
United Kingdom	Honest John	CBB7BE97-946A-44F6-B274-AD5E807F9926	£178,500
United Kingdom	Honest John	EA924C34-F8FD-4E2F-9334-960E90BA6112	£178,500
United Kingdom	Wheels'R'Us	2AE72D9C-5526-400E-979B-FB5E2670B477	£178,500
United Kingdom	Wheels'R'Us	A1413CC7-86C5-46D8-93A2-4A3A8907BFB5	£178,500
United Kingdom	Wheels'R'Us	BAC06A12-FB28-4001-82D5-C49ABF0D6C23	£178,500
United Kingdom	Wheels'R'Us	EE2E8630-7424-4DEC-A680-3AFB63D42BF9	£178,500
United Kingdom	Bright Orange	3FF3174D-17CD-46A6-864E-1ED25903CAB7	£157,750
USA	Style 'N Ride	A3EC91D2-69FD-4CAD-AB7B-B7A1CEEC7F22	£157,750
United Kingdom	Wheels'R'Us	4DFCF7EF-C853-4D75-B584-75F6D752DE92	£155,000
Total			**£31,839,190**

***Figure 14-7.** Progressive table granularity*

Power BI Desktop always attempts to display the data using the information available to it in the underlying data model. Exactly how this can be optimized for the best possible results is described in Chapters 10 through 13.

Types of Data

Not all data is created equal (or at least identical), and the data model that underlies Power BI Desktop will provide you with different types of data. The two core data types are

- Descriptive (non-numeric) attributes
- Values (or numeric measures)

Power BI Desktop indicates the precise data type by using a descriptive icon beside many of the fields, which you can see when you expand a data table in the Fields list. These data types are described in Table 14-2.

Table 14-2. *Data Types*

Data Type	Icon	Description
Attribute	None	A descriptive element and is non-numeric. It can be counted but not summed or averaged.
Aggregates	Σ	A numeric field whose aggregation type can be changed.
Calculated Column	▦	The result of a formula applied to create a new, calculated column.
Calculation	🖩	A numeric field whose aggregation type cannot be changed as it is the result of a specific calculation.
Geography	🌐	This field can potentially be used in a map to provide geographical references.
Binary Data		This field contains data, such as images.

Numeric fields are not the only ones that can be added as aggregates. If you add an attribute field to a table and then display the popup menu for this field by clicking the small triangle to the right of the field name in the Visualizations pane, you can select Count. You will display the *number* of elements for this attribute in the column of the table instead of the text of the element. Figure 14-8 shows you a sample popup field menu for a text field.

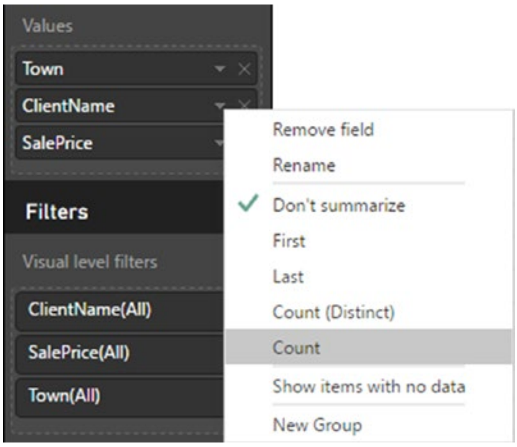


Figure 14-8. *The popup menu for a text field*

Enhancing Tables

So you have a basic table set up and it has the columns you want in the correct order. Quite naturally, the next step is to want to spice up the presentation of the table a little. So let's see what Power BI Desktop has to offer here. Specifically, we will look at

- Adding and removing totals
- Adjusting text sizes in tables
- Changing columns widths
- Sorting rows by the data in a specific column

Row Totals

Row totals are added automatically to all numeric fields. You may wish to remove the totals, however. Conversely, you could want to add totals that were removed previously. In any case, to remove all the totals from a table:

1. Select the table. In this example, I use the table that you saw in Figure 14-2.
2. In the Visualizations pane, click the Format icon (the small paint roller beneath the palette of available visual types). The formatting options for the selected table will be displayed.
3. Expand the Total section by clicking the downward-facing chevron to the left of the word *Total*.
4. Drag the full circle button (to the right of Totals) to the left (or click to the left of the button or even on the button itself). It will become an empty circle and On will become Off in the formatting options. The Visualizations pane will look what is shown in Figure 14-9.

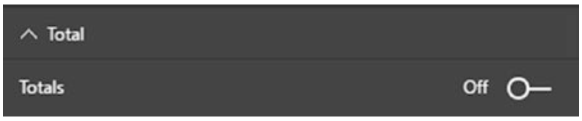
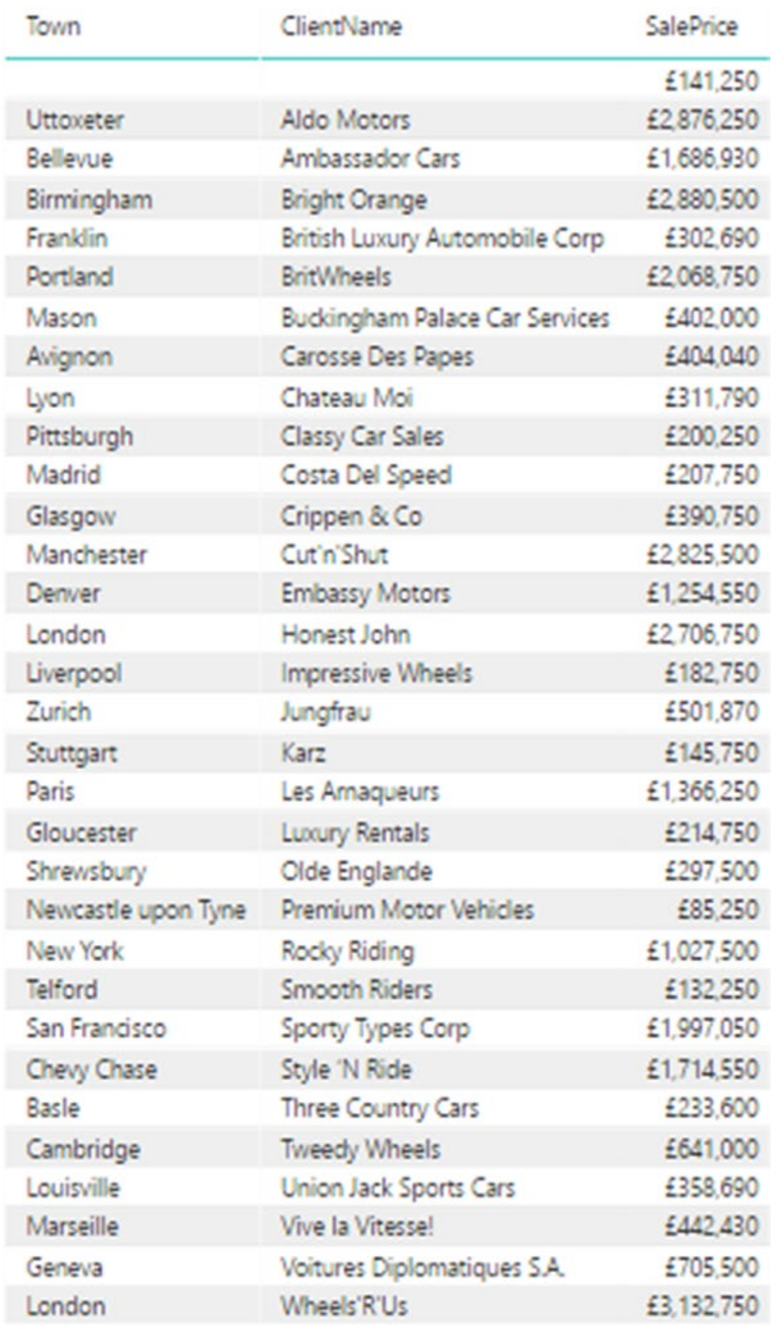


Figure 14-9. *Formatting options for tables in the Visualizations pane*

To add totals where there are none, follow the process described (with the table selected) and at step 4, drag the Off button to the right for Totals to set the totals to On. You can see the table you created previously—without totals—in Figure 14-10.



Town	ClientName	SalePrice
		£141,250
Uttoxeter	Aldo Motors	£2,876,250
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500
Franklin	British Luxury Automobile Corp	£302,690
Portland	BritWheels	£2,068,750
Mason	Buckingham Palace Car Services	£402,000
Avignon	Carosse Des Papes	£404,040
Lyon	Chateau Moi	£311,790
Pittsburgh	Classy Car Sales	£200,250
Madrid	Costa Del Speed	£207,750
Glasgow	Crippen & Co	£390,750
Manchester	Cut'n'Shut	£2,825,500
Denver	Embassy Motors	£1,254,550
London	Honest John	£2,706,750
Liverpool	Impressive Wheels	£182,750
Zurich	Jungfrau	£501,870
Stuttgart	Karz	£145,750
Paris	Les Amaqueurs	£1,366,250
Gloucester	Luxury Rentals	£214,750
Shrewsbury	Olde Englande	£297,500
Newcastle upon Tyne	Premium Motor Vehicles	£85,250
New York	Rocky Riding	£1,027,500
Telford	Smooth Riders	£132,250
San Francisco	Sporty Types Corp	£1,997,050
Chevy Chase	Style 'N Ride	£1,714,550
Basle	Three Country Cars	£233,600
Cambridge	Tweedy Wheels	£641,000
Louisville	Union Jack Sports Cars	£358,690
Marseille	Vive la Vitesse!	£442,430
Geneva	Voitures Diplomatiques S.A.	£705,500
London	Wheels'R'Us	£3,132,750

Figure 14-10. *The initial table without totals*

■ **Note** You can only add or remove totals if a table displays multiple records. If a table is displaying the highest level of aggregation for a value, then no totals can be displayed, as you are looking at the grand total already. In this case, the Totals button is grayed out.

Formatting Numbers in Reports

You cannot format numbers directly in tables in a Power BI Desktop report. This is because all number formatting is centralized in the data model. So if you want to apply a different numeric format from the one that appears when you add a value to a table, you will have to switch to Data View and apply the formatting there. If you need to remind yourself exactly how this is done, then just flip back to Chapter 10 for a quick refresher course on how to format numbers in Power BI Desktop.

Font Sizes in Tables

You may prefer to alter the default font size that Power BI Desktop applies when a table is first created. This is easy to do:

1. Select the table.
2. In the Visualizations pane, click the Format icon.
3. Expand the Values section by clicking the downward-facing chevron to the left of the word Values.
4. Drag the Text Size button to the right (or click to the right of the button). The new text size in points will be displayed in the Text Size box and the actual size of the text in the table will increase.

Here are a few points regarding font sizes in tables:

- As you probably have guessed, dragging the Text Size slider button to the left will decrease the font size.
- You can enter a font size in the Text Size box to specify an exact size, if you prefer.
- Altering the font size will *not* cause the table to grow or shrink, as Power BI Desktop will continue to display the same number of characters per column as were visible using the previous font size. So you may end up having to alter the column widths by setting Auto-size to On (this is described in the following section) or adjusting the table size (as described previously) to make your table look exactly the way you want it.
- This formatting option only affects the size of the *records* in the table, not the column headings. To change the size of the header font, expand the Column Headers section of the Formatting pane and use exactly the same techniques to set the font size. This time it will be the header font that is altered.

Changing Column Widths

Power BI Desktop will automatically set the width of a column so that all the data is visible when you create an initial table.

However—and as you would expect—you can also manually adjust the size of individual columns in a table. You do this more or less as you would in Excel:

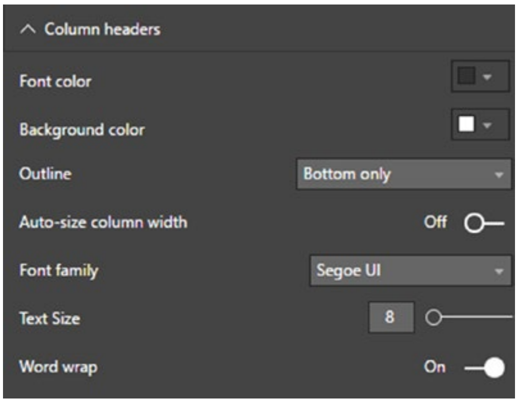
1. In the column header row, place the mouse pointer over the rightmost edge of the column.
2. Drag left or right to adjust the column width.

■ **Note** As you would probably expect from a product that aims at Excel users, you can double-click the column separator at the right of a column's header to have Power BI Desktop adjust the width of the column automatically to fit the widest data element.

Inhibiting Automatic Adjustment of Column Width and Enabling Word Wrap

If you want to, you can prevent Power BI Desktop from resetting column widths automatically, and you can also activate word wrap to make sure that text in a column is not truncated, but flows on to the following line. Here is how:

1. Select the table. In this example, I use the table that you saw in Figure 14-10.
2. In the Visualizations pane, click the Format icon (the small paint roller beneath the palette of available visual types). The formatting options for the selected table will be displayed.
3. Expand the Column Headers section by clicking the downward-facing chevron to the left of the word Column Headers.
4. Drag the full circle button to the right of "Auto-size column width" to the left. It will become an empty circle and On will become Off in the formatting options.
5. Set Word Wrap to On. The Visualizations pane will look like Figure 14-11.


^ Column headers

Font color

Background color

Outline: Bottom only

Auto-size column width: Off

Font family: Segoe UI

Text Size: 8

Word wrap: On

Figure 14-11. *Inhibiting automatic column resizing for tables*

You can now alter the size of the font used in the table without causing column widths to grow or shrink. Inversely, resetting the Auto-size option to On causes the column width to be readjusted automatically to display the widest element in the column.

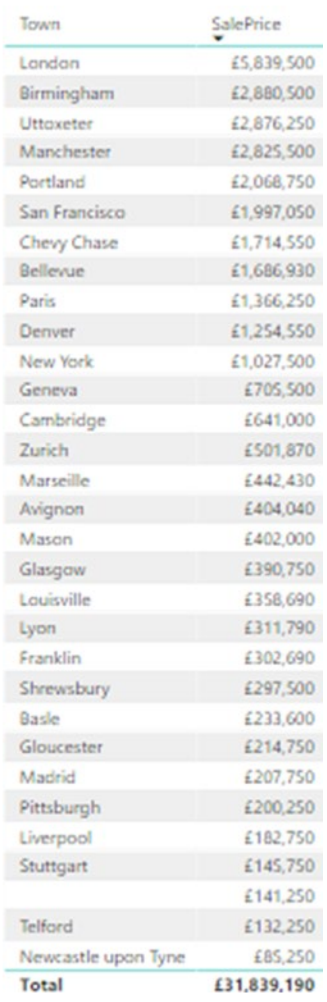
■ **Note** If you are formatting a table and you start to get a little confused as to which options are active and how best to start over, you can always click "Revert to default" to reapply the standard "factory settings" for tables—or for any visual.

Sorting by Column

Any column can be used as the sort criterion for a table, whatever the column data type. To sort the table, merely click the column header. The rows in the table are sorted according to the elements in the selected column. The sort order is A to Z (or lowest to highest for numeric values) the first time that you sort a column.

Once a table has been sorted, you cannot unsort it. You can use another column to resort the data, however. Alternatively, you can click the column title again to sort in the opposing order—Z to A (or highest to lowest for numeric values).

For an example of sorting a column, look at Figure 14-12 (once again, I will use the table we created at the very beginning of this chapter). You can see that the data is sorted by Sale Price from highest to lowest. Moreover, you can see a small down-facing triangle below the title of the SalePrice column. This indicates that the column is sorted from the highest to the lowest value.



Town	SalePrice
London	£5,839,500
Birmingham	£2,880,500
Uttoxeter	£2,876,250
Manchester	£2,825,500
Portland	£2,068,750
San Francisco	£1,997,050
Chevy Chase	£1,714,550
Bellevue	£1,686,930
Paris	£1,366,250
Denver	£1,254,550
New York	£1,027,500
Geneva	£705,500
Cambridge	£641,000
Zurich	£501,870
Marseille	£442,430
Avignon	£404,040
Mason	£402,000
Glasgow	£390,750
Louisville	£358,690
Lyon	£311,790
Franklin	£302,690
Shrewsbury	£297,500
Basle	£233,600
Gloucester	£214,750
Madrid	£207,750
Pittsburgh	£200,250
Liverpool	£182,750
Stuttgart	£145,750
	£141,250
Telford	£132,250
Newcastle upon Tyne	£85,250
Total	**£31,839,190**

Figure 14-12. *Sorting a table by a numeric column*

Sometimes you are sorting a column on one field (as was the case in all the examples so far), but the actual sort uses another column as the basis for the sort operation. For example, you could sort by month name but see the result by the month number (so that you are not sorting months alphabetically, but numerically). You saw how this is configured in Chapter 13.

Formatting Tables

As befits such a polished product, Power BI Desktop can enhance the presentation of even a humble table so that it stands out from the crowd, and delivers that attention-grabbing effect that your audience expects. The currently available formatting options include

- Adding automatic table styles
- Formatting titles
- Modifying the table background
- Adding and removing table borders
- Formatting rows—including alternating rows
- Adding and removing a table grid
- Formatting column headers
- Column formatting
- Formatting totals
- Conditional formatting—including color scales and data bars

I confidently expect that these options will continue to be developed as Power BI Desktop matures. Consequently, you could see further presentation enhancements by the time that you read this book. However, for the moment, let's take a look at the currently available options.

■ **Note** There are no specific options for formatting the figures in tables. You format the data by clicking on the data icon on the left of the Power BI Desktop window, select the column to format and, in the Modeling ribbon, select a number format. This format will then apply to the selected field in the same way in every visual where it is used.

Table Style

Power BI Desktop allows you to choose from nine different ways of instantly formatting a table in a few clicks. This includes the option of removing all table formatting. To format an entire table:

1. Click inside the table to format.
2. In the Visualizations pane, click the Format icon.
3. Expand the Table Style section. Select a table style from the available options.

You can see the range of available table styles in Table 14-3.

Table 14-3. *The Available Table Styles*

Matrix Style	Description
Default	Adds a gray background to alternating rows.
None	Removes all formatting.
Minimal	Adds a light spacing between records.
Bold Header	Adds a dark background to the title row.
Alternating Rows	Adds a gray background to alternating rows and a dark background to the title and totals rows.
Contrast Alternating Rows	Alternates the row background between light and darker gray. Adds a dark background to the title and totals rows.
Flashy Rows	Alternates the row background between light and darker green.
Bold Header Flashy Rows	Alternates the row background between light and darker green. Adds a dark background to the title and totals rows.
Sparse	Adds a dark background to the title and totals rows and removes the light line separating rows.
Condensed	Adds a dark background to the title and totals rows and adds vertical and horizontal lines between rows and columns.

The table styles that you can add are really nothing more than preset values. If you want to be precise in the definition of how the records in a table appear, you can apply the various options that are explained in the next few sections.

■ **Note** To remove a table style and revert to a "plain vanilla" presentation, simply select None from the popup list of available table styles.

Adding and Formatting Titles

Like most visualizations in Power BI Desktop, tables can have titles. Once you have added a title, you can set its

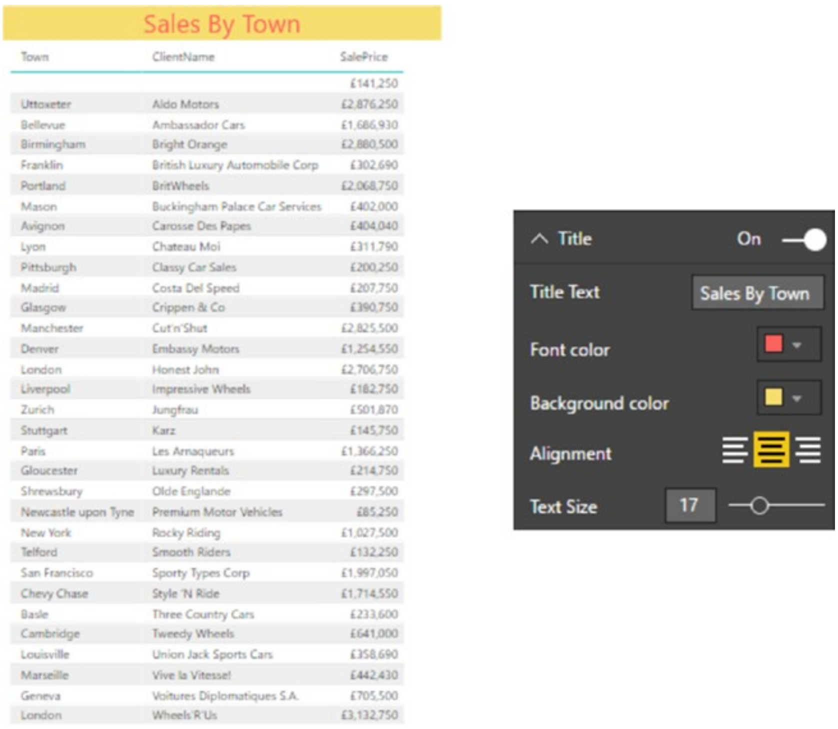
- Font color
- Background color
- Text alignment

As simple as this is, I prefer to explain all the options for the sake of completeness.

1. Select the table. In this example, I use the table that you saw in Figure 14-1.
2. In the Visualizations pane, click the Format icon (the small paint roller beneath the palette of available visualization types). The formatting options for the selected table will be displayed.

3. Expand the Title section by clicking the downward-facing chevron to the left of the word Title.
4. Click to the right of the empty circle to the right of the word Off. This will activate a title for the selected table.
5. In the empty box to the right of Title Text, enter a title for the table. I suggest **Sales by Town**.
6. Click the popup menu triangle to the right of Font Color and select a color from the palette of available hues.
7. Click the popup menu triangle to the right of Background Color and select a color from the palette of available tones.
8. Click the middle icon to the right of Alignment. This will center the text relative to the width of the table.
9. Drag the Text Size button to the right to increase the size of the title font.

The formatting options and the table will look like they do Figure 14-13.



Sales By Town

Town	ClientName	SalePrice
		£141,250
Uttoxeter	Aldo Motors	£2,876,250
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500
Franklin	British Luxury Automobile Corp	£302,690
Portland	BritWheels	£2,068,750
Mason	Buckingham Palace Car Services	£402,000
Avignon	Carosse Des Papes	£404,040
Lyon	Chateau Moi	£311,790
Pittsburgh	Classy Car Sales	£200,250
Madrid	Costa Del Speed	£207,750
Glasgow	Crippen & Co	£390,750
Manchester	Cut'n'Shut	£2,825,500
Denver	Embassy Motors	£1,254,550
London	Honest John	£2,706,750
Liverpool	Impressive Wheels	£182,750
Zurich	Jungfrau	£501,870
Stuttgart	Karz	£145,750
Paris	Les Arnaqueurs	£1,366,250
Gloucester	Luxury Rentals	£214,750
Shrewsbury	Olde Englande	£297,500
Newcastle upon Tyne	Premium Motor Vehicles	£85,250
New York	Rocky Riding	£1,027,500
Telford	Smooth Riders	£132,250
San Francisco	Sporty Types Corp	£1,997,050
Chevy Chase	Style 'N Ride	£1,714,550
Basle	Three Country Cars	£233,600
Cambridge	Tweedy Wheels	£641,000
Louisville	Union Jack Sports Cars	£358,690
Marseille	Vive la Vitesse!	£442,430
Geneva	Voitures Diplomatiques S.A.	£705,500
London	Wheels'R'Us	£3,132,750

Title: On
Title Text: Sales By Town
Font color: 🟥
Background color: 🟨
Alignment
Text Size: 17

Figure 14-13. Options for titles in tables

■ **Note** You can, if you prefer, enter the desired font size in the Text Size field.

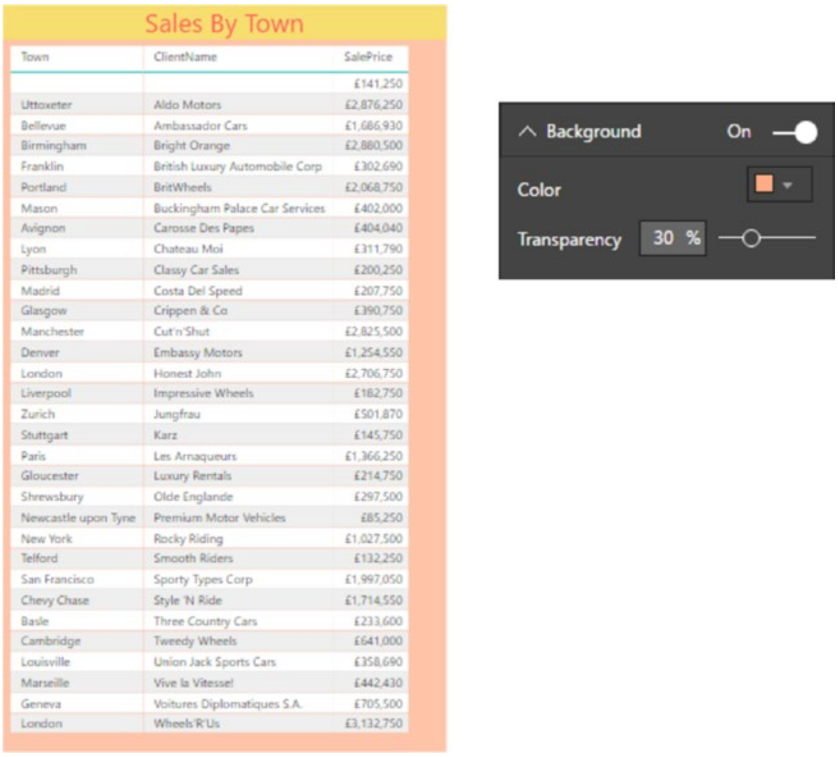
Modifying the Table Background

The table title is not the only aspect of a table that you can modify for visual effect. You can also change a couple of elements of the table itself to add pizzazz to your dashboards. The following are the two things that you can modify:

- The background color
- The table transparency

The following example illustrates both of these possibilities:

1. Select the table. In this example, I use the table that you saw in Figure 14-12.
2. In the Visualizations pane, click the Format icon (the small paint roller beneath the palette of available visual types). The formatting options for the selected table will be displayed.
3. Expand the Background section by clicking the downward-facing chevron to the left of the word Background.
4. Ensure that the switch to the right of the word Background is set to On (if it is not, then click to the right of the empty circle to change this).
5. Click the popup menu triangle to the right of Color and select a color from the palette of available colors.
6. Click the Transparency slider switch and slide it to the left to intensify the background color by reducing the percentage of transparency. The formatting options and the table will look like Figure 14-14.



Sales By Town

Town	ClientName	SalePrice
		£141,250
Uttoxeter	Aldo Motors	£2,876,250
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500
Franklin	British Luxury Automobile Corp	£302,690
Portland	BritWheels	£2,068,750
Mason	Buckingham Palace Car Services	£402,000
Avignon	Carosse Des Papes	£404,040
Lyon	Chateau Moi	£311,790
Pittsburgh	Classy Car Sales	£200,250
Madrid	Costa Del Speed	£207,750
Glasgow	Crippen & Co	£390,750
Manchester	Cut'n'Shut	£2,825,500
Denver	Embassy Motors	£1,254,550
London	Honest John	£2,706,750
Liverpool	Impressive Wheels	£182,750
Zurich	Jungfrau	£501,870
Stuttgart	Karz	£145,750
Paris	Les Arnaqueurs	£1,366,250
Gloucester	Luxury Rentals	£214,750
Shrewsbury	Olde Englande	£297,500
Newcastle upon Tyne	Premium Motor Vehicles	£85,250
New York	Rocky Riding	£1,027,500
Telford	Smooth Riders	£132,250
San Francisco	Sporty Types Corp	£1,997,050
Chevy Chase	Style 'N Ride	£1,714,550
Basle	Three Country Cars	£233,600
Cambridge	Tweedy Wheels	£641,000
Louisville	Union Jack Sports Cars	£358,690
Marseille	Vive la Vitesse!	£442,430
Geneva	Voitures Diplomatiques S.A.	£705,500
London	Wheels'R'Us	£3,132,750

Background: On
Color: 🟧
Transparency: 30 %

Figure 14-14. Setting the background color for a table

■ **Note** You can see that, using this technique, you have only formatted the background of the table visual, not the rows and columns that make up the table itself. You will learn how to format the rows and columns in a couple of pages' time.

Table Borders

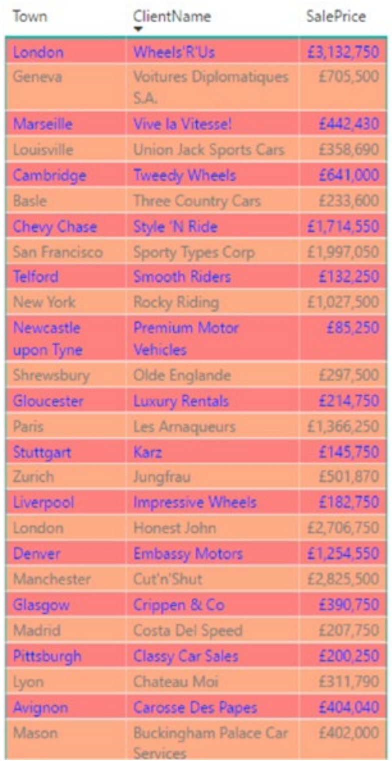
To add another flourish, you can add an outside border to any table. The following explains how to do this:

1. Select the table. In this example, I use the table that you saw in Figure 14-14.
2. In the Visualizations pane, click the Format icon. The formatting options for the selected table will be displayed.
3. Slide the Border button to the right. This will apply a border to the table.
4. Expand the Border section.
5. Select a border color from those available in the popup palette.

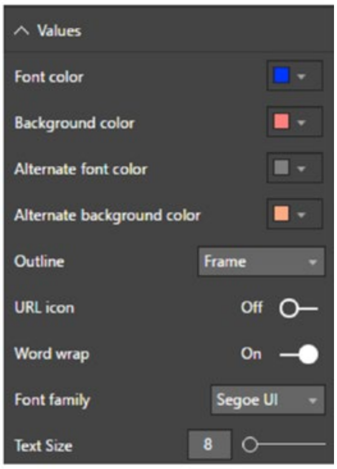
Row Formatting

In an earlier section you saw how to change the font size for the rows in a table. This is only one of the available formatting options that you can apply to the rows in a table. Here are some of the other possibilities:

1. Select the table that you want to enhance. In this example, I use the table that you saw in Figure 14-10.
2. In the Visualizations pane, click the Format icon.
3. Expand the Values section.
4. Select a Background Color as well as an Alternate Background Color.
5. Select a Font Color as well as an Alternate Font Color.
6. Select Outline from the available outline styles.
7. Set Word Wrap to On.
8. Reduce the width of the left two columns. The table should look something like the one shown in Figure 14-15.



Town	ClientName	SalePrice
London	Wheels'R'Us	£3,132,750
Geneva	Voitures Diplomatiques S.A.	£705,500
Marseille	Vive la Vitesse!	£442,430
Louisville	Union Jack Sports Cars	£358,690
Cambridge	Tweedy Wheels	£641,000
Basle	Three Country Cars	£233,600
Chevy Chase	Style 'N Ride	£1,714,550
San Francisco	Sporty Types Corp	£1,997,050
Telford	Smooth Riders	£132,250
New York	Rocky Riding	£1,027,500
Newcastle upon Tyne	Premium Motor Vehicles	£85,250
Shrewsbury	Olde Englande	£297,500
Gloucester	Luxury Rentals	£214,750
Paris	Les Arnaqueurs	£1,366,250
Stuttgart	Karz	£145,750
Zurich	Jungfrau	£501,870
Liverpool	Impressive Wheels	£182,750
London	Honest John	£2,706,750
Denver	Embassy Motors	£1,254,550
Manchester	Cut'n'Shut	£2,825,500
Glasgow	Crippen & Co	£390,750
Madrid	Costa Del Speed	£207,750
Pittsburgh	Classy Car Sales	£200,250
Lyon	Chateau Moi	£311,790
Avignon	Carosse Des Papes	£404,040
Mason	Buckingham Palace Car Services	£402,000



Values

Font color	🟦
Background color	🟥
Alternate font color	⬜
Alternate background color	🟧
Outline	Frame
URL icon	Off
Word wrap	On
Font family	Segoe UI
Text Size	8

Figure 14-15. *Formatting alternate rows in a table*

There are currently seven outline styles available, plus the option of removing all outline styles. These are explained in Table 14-4.

Table 14-4. *The Available Outline Styles*

Outline Style	Description
None	Removes borders from the title.
Bottom only	Adds a bottom border to the title.
Top only	Adds a top border to the title.
Left only	Adds a left border to the title.
Right only	Adds a right border to the title.
Top + bottom	Adds both top and bottom borders to the title.
Left + right	Adds both left and right borders to the title.
Frame	Adds a frame around the title.

I have a couple of minor explanations to add at this juncture:

- Selecting an outline style will add a border to the entire table *inside* the visual, as opposed to the table border that you added previously that applies to the outer edge of the entire visual.
- Enabling Word Wrap will enable text to flow to the next line in all columns. This will increase the height of any affected rows.
- Enabling the URL Icon option in the Values section will convert a URL text to an icon.

Table Grid

You can extend the presentation of a table with much greater precision if you use the Table Grid section of the formatting options in the Visualizations pane.

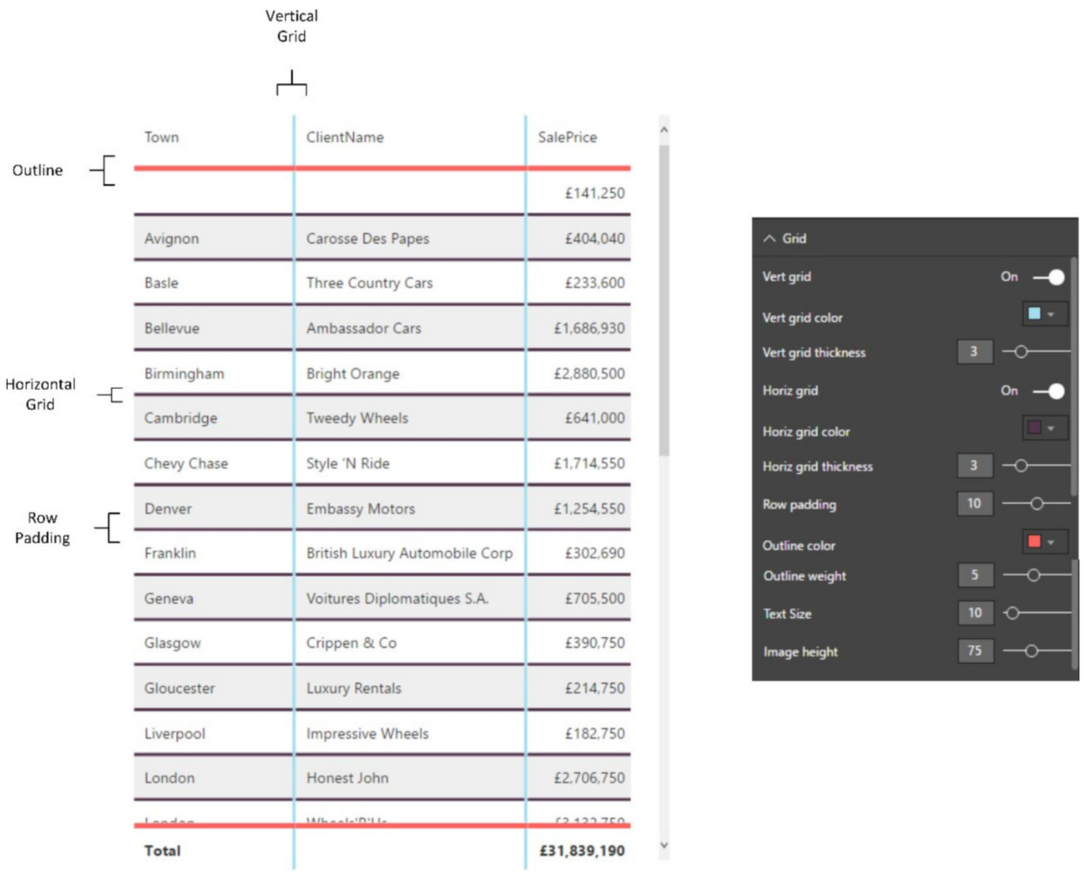
Let's see how this works, taking the table from Figure 14-10, and with the default style:

1. Select the table to format, and in the Visualizations pane, click the Format icon.
2. Expand the Grid section.
3. Display the vertical grid by switching Vert Grid to On.
4. Select a color from the palette of available colors for the vertical grid.
5. Enter **3** in the field for Vert Grid Thickness.
6. Select a color from the palette of available colors for the horizontal grid (this should be on by default; if not, simply slide the button for Horiz Grid to the right).
7. Slide the Horiz grid button to the right to set a different thickness for the horizontal grid.
8. Enter **10** in the field for Row Padding.
9. Select a color from the palette of available colors for the outline color.

10. Enter **5** in the field for Outline Weight.

11. Enter **10** in the field for Text Size.

You can see the results of these changes in Figure 14-16. I am in no way pretending that there is any profound aesthetic value to these enhancements-but you can, at least, see how Power BI Desktop can be used to format your data.



Town	ClientName	SalePrice
		£141,250
Avignon	Carosse Des Papes	£404,040
Basle	Three Country Cars	£233,600
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500
Cambridge	Tweedy Wheels	£641,000
Chevy Chase	Style 'N Ride	£1,714,550
Denver	Embassy Motors	£1,254,550
Franklin	British Luxury Automobile Corp	£302,690
Geneva	Voitures Diplomatiques S.A.	£705,500
Glasgow	Crippen & Co	£390,750
Gloucester	Luxury Rentals	£214,750
Liverpool	Impressive Wheels	£182,750
London	Honest John	£2,706,750
[illegible]	[illegible]	[illegible]
Total		**£31,839,190**

Grid	
Vert grid	On
Vert grid color	🟦
Vert grid thickness	3
Horiz grid	On
Horiz grid color	🟪
Horiz grid thickness	3
Row padding	10
Outline color	🟥
Outline weight	5
Text Size	10
Image height	75

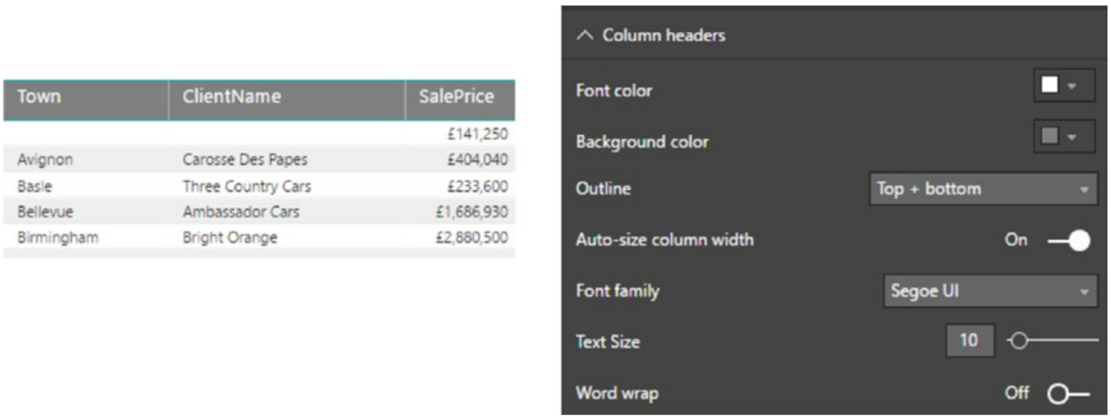
Figure 14-16. *Table grid settings*

■ **Note** Depending on the resolution of your monitor, you might have to scroll down inside the Grid section of the Formatting pane. You do this using the gray vertical bar at the right of the Grid section.

Column Headers

Power BI Desktop also allows you to modify the formatting of column headers. You could carry out the following tweaks to see this in action:

1. Re-create the initial table from Figure 14-1, with the default table style.
2. Expand the Column Headers section.
3. Set a font and background color using the appropriate palettes for each.
4. Slide the Text Size button to the right to increase the font size for the title.
5. In the Outline popup, select Top + Bottom. These styles are the same as those that you previously applied to row values, and are described in Table 14-4. You can see the resulting format in Figure 14-17.



Town	ClientName	SalePrice
		£141,250
Avignon	Carosse Des Papes	£404,040
Basle	Three Country Cars	£233,600
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500

Column headers

Font color	⬜
Background color	◼
Outline	Top + bottom
Auto-size column width	On
Font family	Segoe UI
Text Size	10
Word wrap	Off

Figure 14-17. *Formatting column headers*

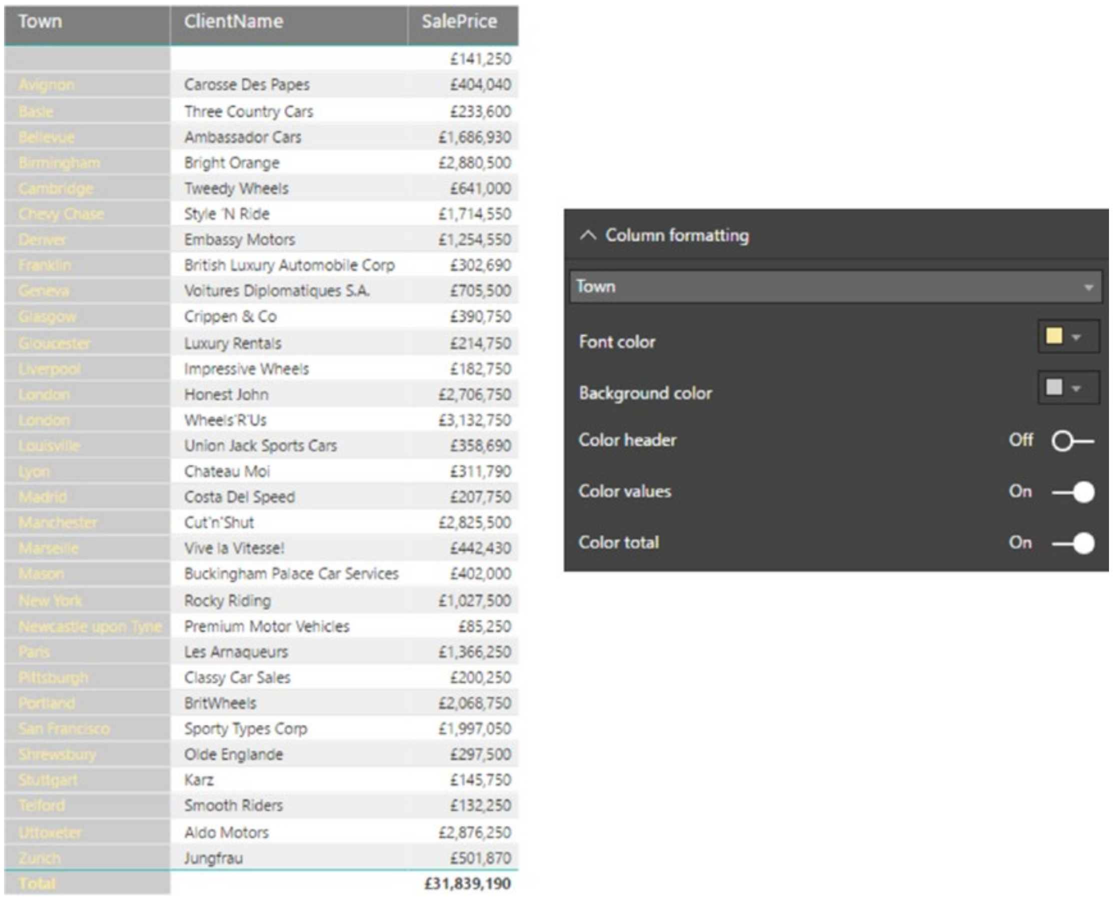
■ **Note** If you choose to enter a text size in the Text Size field, be aware that there are limits that you cannot exceed. For instance, if you enter a figure less than 8, Power BI Desktop will refuse to apply the requested font size and will add a red outline to the Text Size field. You will then have to enter a valid number.

Column Formatting

Each individual column can be formatted separately. You need to be aware that, in practice, overuse of this technique can easily lead to total illegibility. Nonetheless, it is worth learning how to format an individual column.

1. Re-create the table from Figure 14-10, and with the default matrix style.
2. Expand the Column Formatting section.
3. Select the column header for the column that you wish to format from the popup list at the top of the Column Formatting section. I will choose Town in this example.

4. Set a font and background color using the appropriate palettes for each.
5. Set Color Total to On. You can see the output in Figure 14-18.



Town	ClientName	SalePrice
		£141,250
Avignon	Carosse Des Papes	£404,040
Basle	Three Country Cars	£233,600
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500
Cambridge	Tweedy Wheels	£641,000
Chevy Chase	Style 'N Ride	£1,714,550
Denver	Embassy Motors	£1,254,550
Franklin	British Luxury Automobile Corp	£302,690
Geneva	Voitures Diplomatiques S.A.	£705,500
Glasgow	Crippen & Co	£390,750
Gloucester	Luxury Rentals	£214,750
Liverpool	Impressive Wheels	£182,750
London	Honest John	£2,706,750
London	Wheels'R'Us	£3,132,750
Louisville	Union Jack Sports Cars	£358,690
Lyon	Chateau Moi	£311,790
Madrid	Costa Del Speed	£207,750
Manchester	Cut'n'Shut	£2,825,500
Marseille	Vive la Vitesse!	£442,430
Mason	Buckingham Palace Car Services	£402,000
New York	Rocky Riding	£1,027,500
Newcastle upon Tyne	Premium Motor Vehicles	£85,250
Paris	Les Arnaqueurs	£1,366,250
Pittsburgh	Classy Car Sales	£200,250
Portland	BritWheels	£2,068,750
San Francisco	Sporty Types Corp	£1,997,050
Shrewsbury	Olde Englande	£297,500
Stuttgart	Karz	£145,750
Telford	Smooth Riders	£132,250
Uttoxeter	Aldo Motors	£2,876,250
Zurich	Jungfrau	£501,870
Total		£31,839,190

Column formatting

Town

Font color

Background color

Color header Off

Color values On

Color total On

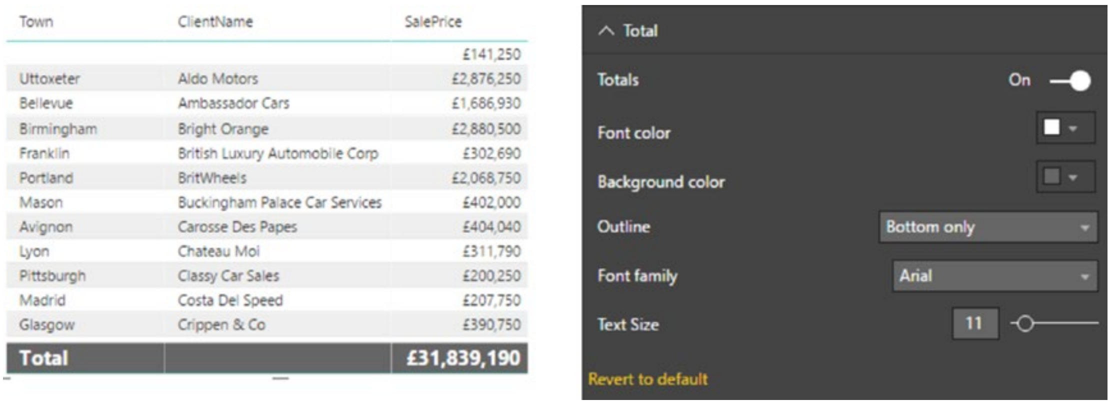
Figure 14-18. *Adding column formatting*

Formatting Totals

Power BI Desktop lets you do more to totals in tables than just switch them on and off. You can format the total to enhance the visibility (or, inversely, try and divert attention from poor figures), like this:

1. Re-create the table that you can see in Figure 14-9.
2. In the Visualizations pane (and, of course, leaving the table selected), activate the Format panel and expand the Total section.
3. Ensure that the Totals option is set to On.
4. Choose a font color from those in the palette.
5. Choose a background color from those in the palette.

6. Enter of adjust the slider to obtain a larger font size. You can see the result of these actions as well as the Total section of the Formatting pane in Figure 14-19.



Town	ClientName	SalePrice
		£141,250
Uttoxeter	Aldo Motors	£2,876,250
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500
Franklin	British Luxury Automobile Corp	£302,690
Portland	BritWheels	£2,068,750
Mason	Buckingham Palace Car Services	£402,000
Avignon	Carosse Des Papes	£404,040
Lyon	Chateau Moi	£311,790
Pittsburgh	Classy Car Sales	£200,250
Madrid	Costa Del Speed	£207,750
Glasgow	Crippen & Co	£390,750
Total		**£31,839,190**

Total

Totals On

Font color

Background color

Outline Bottom only

Font family Arial

Text Size 11

Revert to default

Figure 14-19. Formatting totals

Conditional Formatting

Tables of data may convey vast amounts of information, but they rarely enable instant understanding of the meaning of the figures. This is where conditional formatting can be applied to convert data into instantly useful information. Indeed, you may already be used to applying these kinds of enhancements to Excel spreadsheets.

In Power BI Desktop there are two types of conditional formatting that you can apply to add meaning to your figures:

- *Color scales*: Add shades of color to the entire background of a table cell.
- *Data bars*: Add a background bar to a table cell. Here the size of the bar represents the value in the cell.

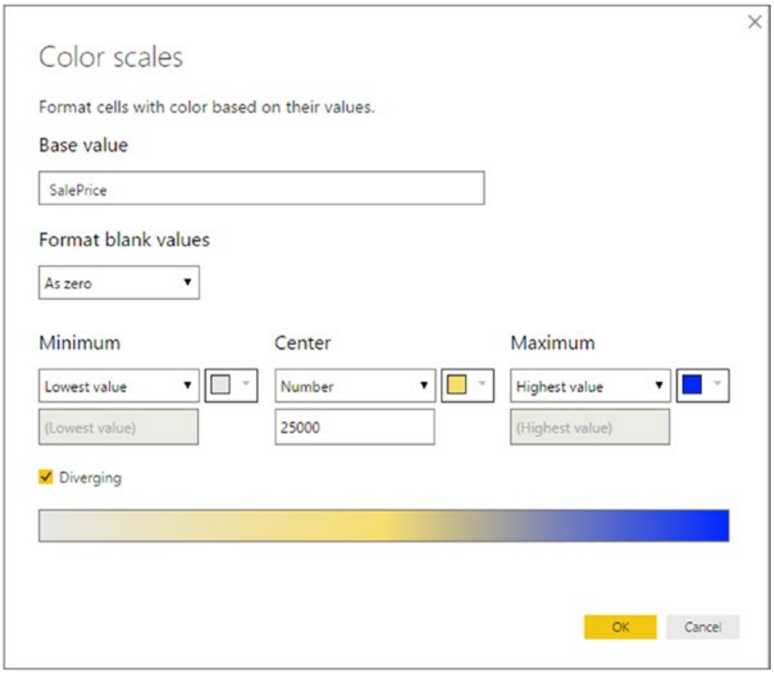
■ **Note** Conditional formatting can only be applied to numeric values, not to text elements.

Color Scales

Color scales are a background color that you apply to each cell in a column. The actual color will vary according to the value of each row. As an example of how color scales work, try the following:

1. Re-create the table from Figure 14-10.
2. Click the Format icon in the Visualizations pane and expand the Conditional Formatting section.
3. Select the name of the column to format from the popup list of the columns in the table (in this example there is only the SalePrice column).
4. Set the Color Scales button to On.

5. Click the Advanced Controls link. The Color Scales dialog will be displayed.
6. In the “Format blank values” popup, select As Zero.
7. Set the color for the minimum to a light color from the popup palette.
8. Click the Diverging check box.
9. Set the color for the center to a medium color from the popup palette.
10. In the popup for Center, select Number.
11. Enter **25000** in the number field that is now enabled.
12. Set the color for the maximum to a dark color from the popup palette. The Color Scales dialog will look like the one shown in Figure 14-20.


Color scales

Format cells with color based on their values.

Base value: SalePrice

Format blank values: As zero

Minimum: Lowest value (Lowest value)

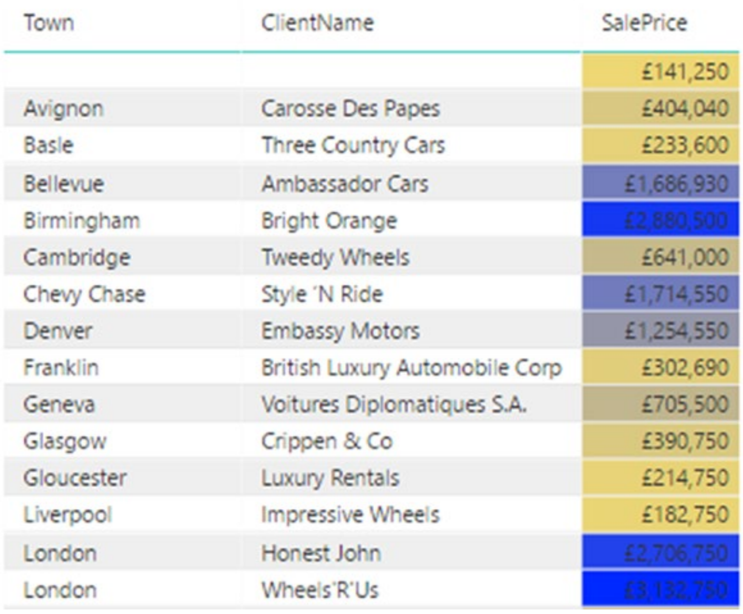
Center: Number 25000

Maximum: Highest value (Highest value)

✓ Diverging

Figure 14-20. The Color Scales dialog

13. Click OK. The table will look like the one in Figure 14-21.



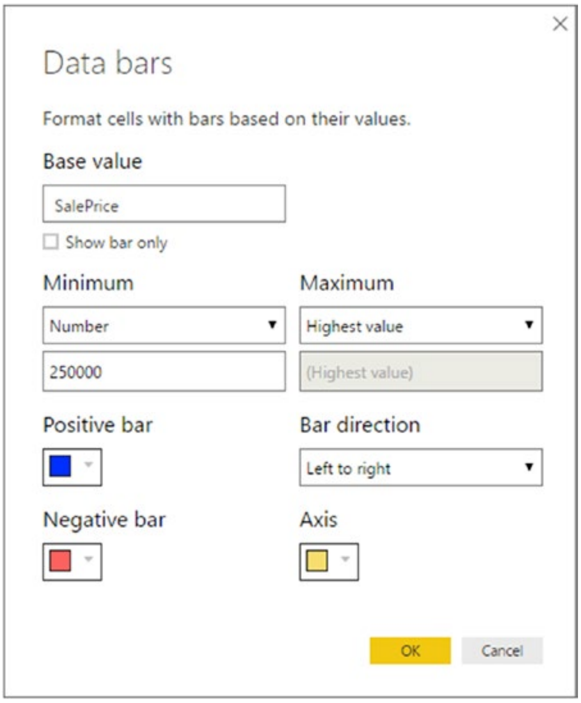
Town	ClientName	SalePrice
		£141,250
Avignon	Carosse Des Papes	£404,040
Basle	Three Country Cars	£233,600
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500
Cambridge	Tweedy Wheels	£641,000
Chevy Chase	Style 'N Ride	£1,714,550
Denver	Embassy Motors	£1,254,550
Franklin	British Luxury Automobile Corp	£302,690
Geneva	Voitures Diplomatiques S.A.	£705,500
Glasgow	Crippen & Co	£390,750
Gloucester	Luxury Rentals	£214,750
Liverpool	Impressive Wheels	£182,750
London	Honest John	£2,706,750
London	Wheels'R'Us	£3,132,750

Figure 14-21. *A table with color scales applied*

Data Bars

Data bars are a presentation technique that represents the figure as a colored bar behind the actual value in a table cell. You can add data bars using the following steps:

1. Re-create the initial table from Figure 14-10 (and with the default table style applied, if necessary).
2. Click the Format icon in the Visualizations pane and expand the Conditional Formatting section.
3. Select the name of the column to format from the popup list of the columns in the table (in this example there is only the SalePrice column).
4. Set the Data Bars button to On.
5. Click the Advanced Controls link. The Data Bars dialog will be displayed.
6. In the Minimum popup, select Number.
7. Enter **250000** as the value in the number field that is now enabled.
8. Choose colors for the positive bar and negative bar. The dialog should look like the one in Figure 14-22.



Data bars

Format cells with bars based on their values.

Base value: SalePrice

☐ Show bar only

Minimum	Maximum
Number	Highest value
250000	(Highest value)

Positive bar: 🟦

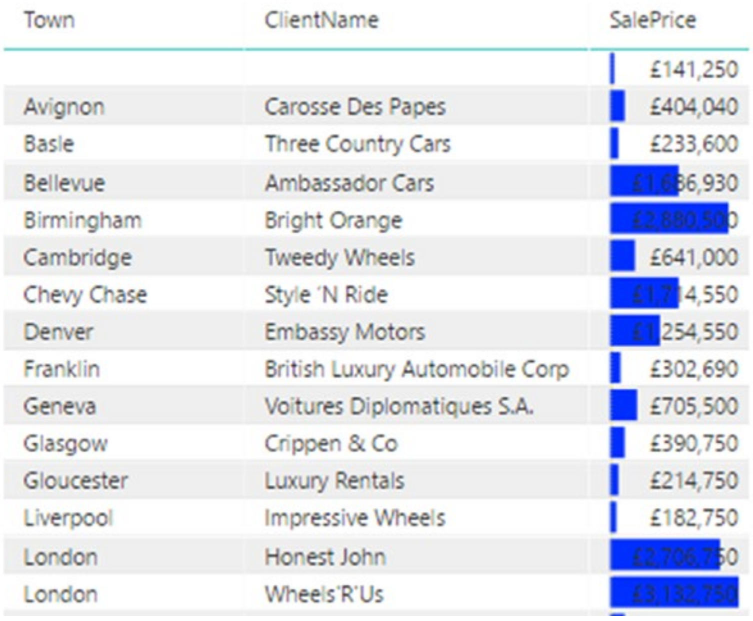
Bar direction: Left to right

Negative bar: 🟥

Axis: 🟨

***Figure 14-22.** The Data Bars dialog*

9. Click OK. The table will look like the one in Figure 14-23.



Town	ClientName	SalePrice
		£141,250
Avignon	Carosse Des Papes	£404,040
Basle	Three Country Cars	£233,600
Bellevue	Ambassador Cars	£1,686,930
Birmingham	Bright Orange	£2,880,500
Cambridge	Tweedy Wheels	£641,000
Chevy Chase	Style 'N Ride	£1,714,550
Denver	Embassy Motors	£1,254,550
Franklin	British Luxury Automobile Corp	£302,690
Geneva	Voitures Diplomatiques S.A.	£705,500
Glasgow	Crippen & Co	£390,750
Gloucester	Luxury Rentals	£214,750
Liverpool	Impressive Wheels	£182,750
London	Honest John	£2,706,750
London	Wheels'R'Us	£3,132,750

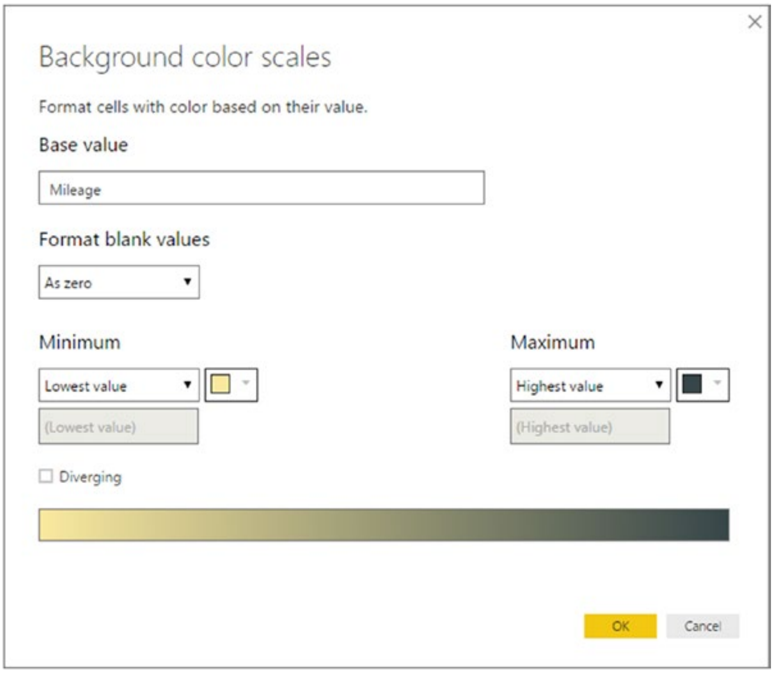
***Figure 14-23.** Data bars applied to a table*

■ **Note** You can choose to add data bars and color scales to the same column at the same time. However, this may detract from the visual rather than enabling the reader to understand the meaning of the data.

Conditional Text and Background

A final conditional formatting option is to set the color of the text or the cell background for each value using conditional formatting. This is, if anything, even simpler than applying data bars or color scales. Indeed, in this example I will use the same settings for both text and background to create a simplistic heatmap in a table.

1. Create a table using the following data elements:
   a. Make
   b. Mileage
2. In the Visualizations pane, click the Format icon.
3. Expand the Conditional Formatting section.
4. Set Background Color Scales to On.
5. Set Font Color Scales to On.
6. Click Advanced Controls for Background Color Scales. The Background Color Scales dialog will appear.
7. Select a color for the minimum from the palette.
8. Select a color for the maximum from the palette. The dialog will look like the one shown in Figure 14-24.


Background color scales

Format cells with color based on their value.

Base value: Mileage

Format blank values: As zero

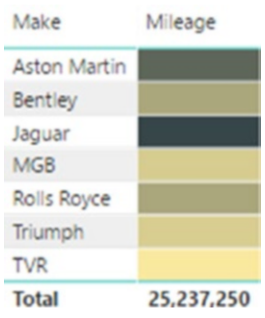
Minimum: Lowest value (Lowest value)

Maximum: Highest value (Highest value)

☐ Diverging

Figure 14-24. *The Background Color Scales dialog*

9. Click OK.
10. Click Advanced Controls for Font Color Scales. The Font Color Scales dialog will appear.
11. Select the same color for the minimum from the palette that you chose for the background.
12. Select the same color for the maximum from the palette that you chose for the background.
13. Click OK. The table will look like it does in Figure 14-25.



Make	Mileage
Aston Martin	⬛
Bentley	🟫
Jaguar	⬛
MGB	🟨
Rolls Royce	🟫
Triumph	🟨
TVR	🟨
Total	**25,237,250**

Figure 14-25. *Applying conditional text and background to a table*

■ **Note** If you prefer, you can set the text and background to different colors to obtain a completely different effect from the application of conditional text.

Conclusion

I hope that you are now comfortable with the Power BI Desktop reports interface and are relaxed about using it to present your data as tables. You learned not just how to create tables, but also how to change their presentation in a myriad of subtle (and not so subtle) ways to enhance their impact. This ranges from setting font attributes to conditional formatting via adjusting the appearance of both titles and data.

Equally, I hope that you are at ease sorting the data in your tables using the various possible techniques.

This chapter was just a taster of the many ways in which Power BI Desktop can help you analyze and display the information that you want your audience to appreciate. So, as tables are the basis for just about every other form of visual, it is well worth mastering the techniques and tricks of table creation. This way, you are well on the way to a fluent mastery of Power BI Desktop, which lays the foundations for some truly impressive presentations.

Now that you have mastered the basics, it is time to move on to the next level of dashboard creation where you will discover how to create and enhance matrices and cards. These are explained in the next chapter.

CHAPTER 15

Matrix and Card Visuals

This chapter takes you through the process of creating further text-based visualizations in Power BI Desktop. You will learn how to further develop the skills that you acquired in the previous chapter to create and enhance

- Matrices
- Cards
- Multirow cards

The techniques that you will learn in this chapter use a Power BI Desktop file that results from the application of much of what you saw in the last four chapters. This file is called C:\PowerBiDesktopSamples\CH15\CarSalesDataForReports.pbix. It is available on the Apress web site as part of the downloadable material that accompanies this book.

Creating a Matrix

In the previous chapter you saw how tables can be used to display the information as full columns of lists. Lists do not, however, always give an intuitive feeling for how data should be grouped at various levels. Presenting information in a neat hierarchy with multiple grouped levels is the task of a matrix visual.

Creating a Row Matrix

When creating a matrix, I find that it helps to think in terms of a hierarchy of information and to visualize this information flowing from left to right. For instance, suppose that we want to create a matrix with the country name as the highest level in the hierarchy (and consequently the leftmost item). Then we want the make of car to be the second level, and the next element in from the left. Finally, we want the color of car sold, followed by all the numeric fields that interest us. You can see this progression illustrated in Figure 15-1.

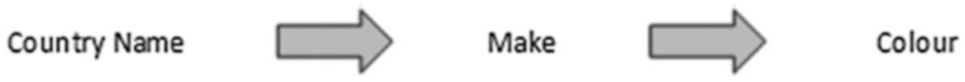


Figure 15-1. *An information hierarchy*

When creating a matrix, it is important to have the Visualizations pane reflect the hierarchy. Put another way, you must ensure that the order of the fields that you select for the matrix and that you place in the field well follows the display hierarchy that you want. Consequently, to create a matrix like the one just described, you need to do the following:

1. Use the C:\PowerBiDesktopSamples\CH15\CarSalesDataForReports.pbix file as your source of data.
2. Click the Matrix icon in the Visualizations pane. This is shown in Figure 15-2. An empty matrix will appear on the report canvas.

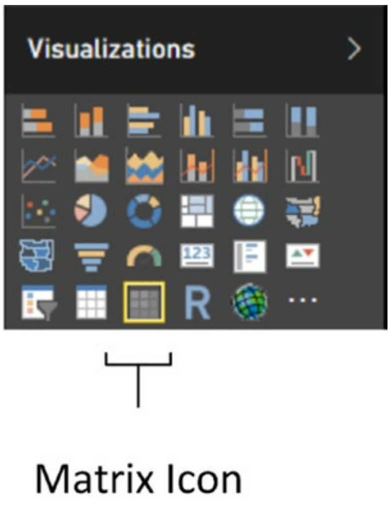
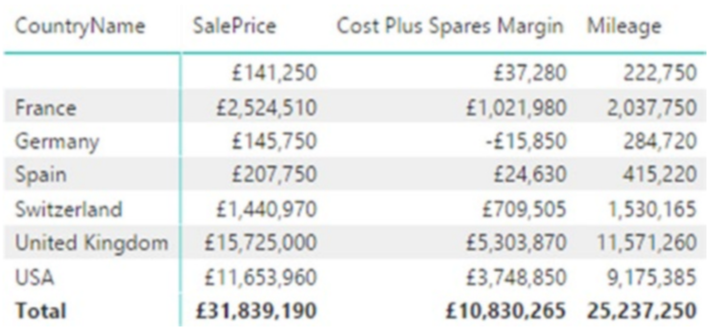


Figure 15-2. *A matrix table*

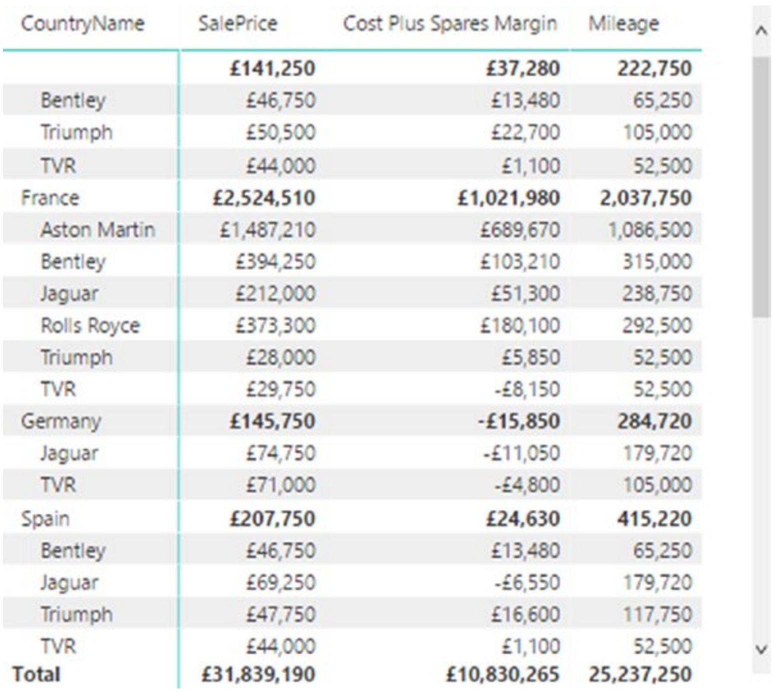
1. Drag the fields CountryName (from the Countries table), Make (from the Stock table), and Color (from the Colors table) in this order to the Rows well in the Visualizations pane. Then add the fields SalePrice and Cost Plus Spares Margin (from the InvoiceLines table) and Mileage (from the Stock table). The matrix should look something like Figure 15-3.



CountryName	SalePrice	Cost Plus Spares Margin	Mileage
	£141,250	£37,280	222,750
France	£2,524,510	£1,021,980	2,037,750
Germany	£145,750	-£15,850	284,720
Spain	£207,750	£24,630	415,220
Switzerland	£1,440,970	£709,505	1,530,165
United Kingdom	£15,725,000	£5,303,870	11,571,260
USA	£11,653,960	£3,748,850	9,175,385
Total	**£31,839,190**	**£10,830,265**	**25,237,250**

Figure 15-3. *A matrix showing the top level of data*

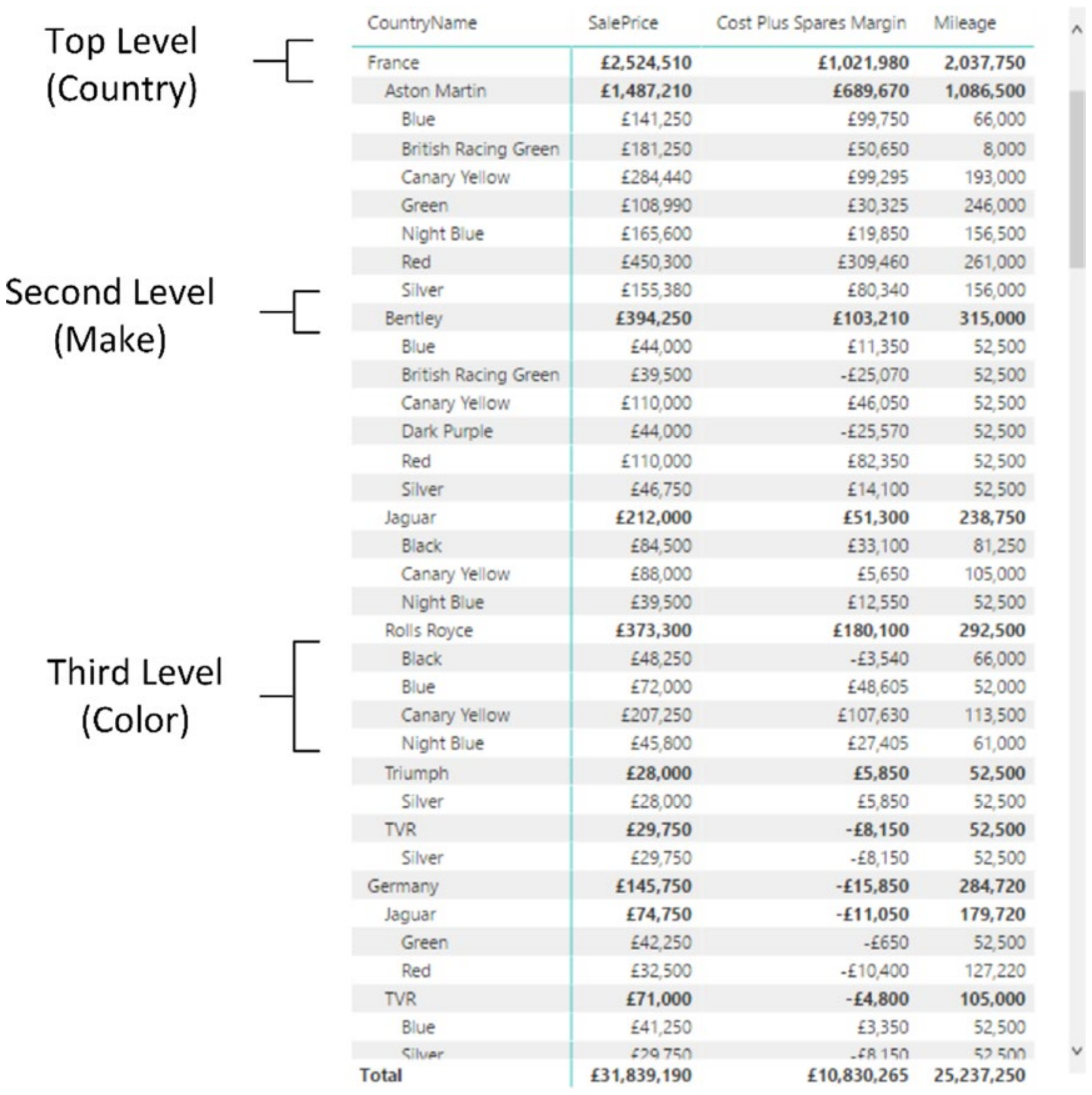
2. Click the “Expand all down one level in the hierarchy” icon at the top of the matrix. This is the third icon from the left that looks like a small pitchfork (you can see this icon in Figure 15-11 if you want to jump ahead slightly). The matrix should look something like Figure 15-4 (note that it is perfectly normal that there is no country name for the first set of rows).



CountryName	SalePrice	Cost Plus Spares Margin	Mileage
	£141,250	**£37,280**	**222,750**
Bentley	£46,750	£13,480	65,250
Triumph	£50,500	£22,700	105,000
TVR	£44,000	£1,100	52,500
France	**£2,524,510**	**£1,021,980**	**2,037,750**
Aston Martin	£1,487,210	£689,670	1,086,500
Bentley	£394,250	£103,210	315,000
Jaguar	£212,000	£51,300	238,750
Rolls Royce	£373,300	£180,100	292,500
Triumph	£28,000	£5,850	52,500
TVR	£29,750	-£8,150	52,500
Germany	**£145,750**	**-£15,850**	**284,720**
Jaguar	£74,750	-£11,050	179,720
TVR	£71,000	-£4,800	105,000
Spain	**£207,750**	**£24,630**	**415,220**
Bentley	£46,750	£13,480	65,250
Jaguar	£69,250	-£6,550	179,720
Triumph	£47,750	£16,600	117,750
TVR	£44,000	£1,100	52,500
Total	**£31,839,190**	**£10,830,265**	**25,237,250**

Figure 15-4. *A two-level matrix*

3. Click the “Expand all down one level in the hierarchy” icon again. The matrix should look something like Figure 15-5.



CountryName	SalePrice	Cost Plus Spares Margin	Mileage
France	**£2,524,510**	**£1,021,980**	**2,037,750**
Aston Martin	**£1,487,210**	**£689,670**	**1,086,500**
Blue	£141,250	£99,750	66,000
British Racing Green	£181,250	£50,650	8,000
Canary Yellow	£284,440	£99,295	193,000
Green	£108,990	£30,325	246,000
Night Blue	£165,600	£19,850	156,500
Red	£450,300	£309,460	261,000
Silver	£155,380	£80,340	156,000
Bentley	**£394,250**	**£103,210**	**315,000**
Blue	£44,000	£11,350	52,500
British Racing Green	£39,500	-£25,070	52,500
Canary Yellow	£110,000	£46,050	52,500
Dark Purple	£44,000	-£25,570	52,500
Red	£110,000	£82,350	52,500
Silver	£46,750	£14,100	52,500
Jaguar	**£212,000**	**£51,300**	**238,750**
Black	£84,500	£33,100	81,250
Canary Yellow	£88,000	£5,650	105,000
Night Blue	£39,500	£12,550	52,500
Rolls Royce	**£373,300**	**£180,100**	**292,500**
Black	£48,250	-£3,540	66,000
Blue	£72,000	£48,605	52,000
Canary Yellow	£207,250	£107,630	113,500
Night Blue	£45,800	£27,405	61,000
Triumph	**£28,000**	**£5,850**	**52,500**
Silver	£28,000	£5,850	52,500
TVR	**£29,750**	**-£8,150**	**52,500**
Silver	£29,750	-£8,150	52,500
Germany	**£145,750**	**-£15,850**	**284,720**
Jaguar	**£74,750**	**-£11,050**	**179,720**
Green	£42,250	-£650	52,500
Red	£32,500	-£10,400	127,220
TVR	**£71,000**	**-£4,800**	**105,000**
Blue	£41,250	£3,350	52,500
Silver	£29,750	-£8,150	52,500
Total	**£31,839,190**	**£10,830,265**	**25,237,250**

Figure 15-5. *A three-level matrix*

As you can see, a matrix display not only makes data easier to digest, but it automatically groups records by each element in the hierarchy and adds totals for each level as well. What is more, each level in the hierarchy is sorted in ascending order. You can expand all the levels that are available in the data that you added to the matrix definition. This final level will display all available detail at that level.

You can also add fields directly to a table by dragging them onto the table. When you do this, Power BI Desktop always adds a text field to the *right* of existing fields. In a matrix, this means that any aggregate/numeric field is added to the right of existing aggregate fields (and appear in the Values box), whereas any text or date/time fields are added to the right of any existing hierarchy fields (and appear in the Rows box). However, it is always a simple matter to reorganize them by dragging the required fields up and down in the Rows and Values boxes to define the correct hierarchy and the overall type of display that you want to achieve. You can see how the Visualizations pane reflects the matrix hierarchy in Figure 15-6.

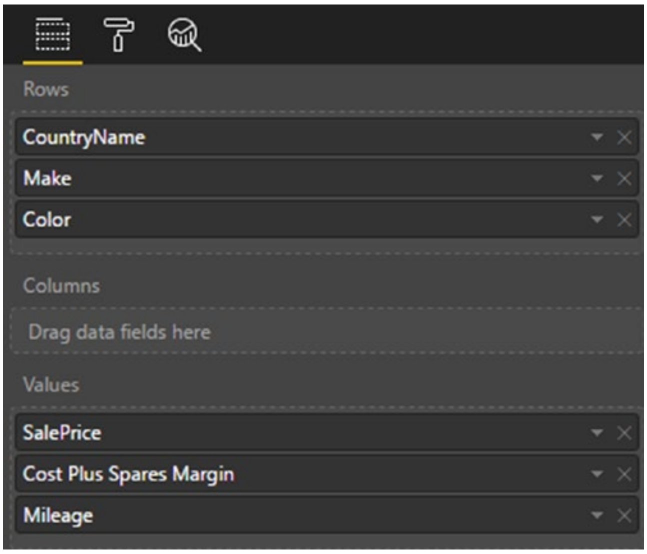


Figure 15-6. *The Visualizations pane showing the matrix hierarchy*

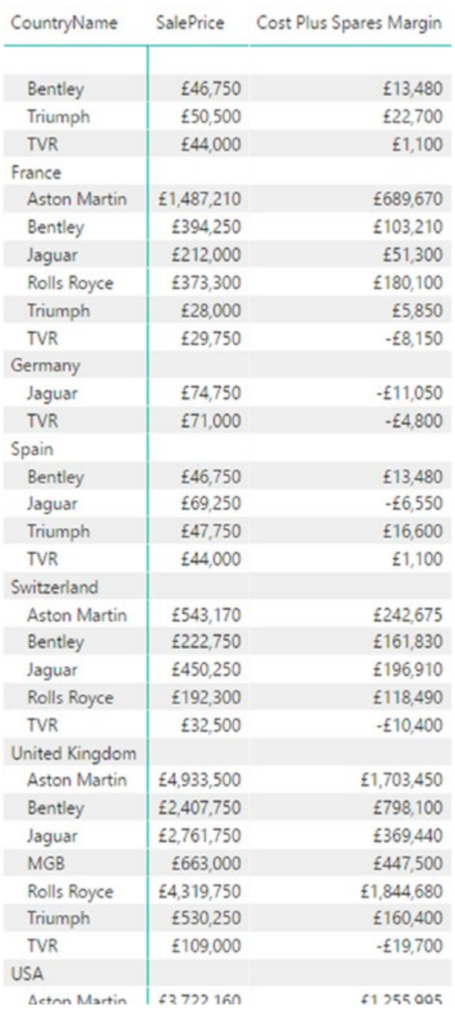
When creating matrices, my personal preference is to drag the fields that constitute the hierarchy of non-numeric values into the Rows box, which means I am placing them accurately above, below, or between any existing elements. This ensures that your matrix looks right the first time, which can help you avoid some very disconcerting double takes! Also as Power BI Desktop will add additional fields to the Columns well by default, it helps to be precise when building a matrix. This means carefully placing the fields for the row headers in the Rows well.

Adding or Removing Subtotals in a Matrix

By default, a new matrix will include the subtotals for every level of the data. Should you prefer to remove the subtotals from a matrix:

1. Select the matrix visual that you used in the previous section.
2. In the Visualizations pane, click the Format icon.
3. Expand the Subtotals section.

4. Click the Row Subtotals button to turn the row subtotals off. This will remove all the totals from the matrix, as you can see in Figure 15-7.



CountryName	SalePrice	Cost Plus Spares Margin
Bentley	£46,750	£13,480
Triumph	£50,500	£22,700
TVR	£44,000	£1,100
France		
Aston Martin	£1,487,210	£689,670
Bentley	£394,250	£103,210
Jaguar	£212,000	£51,300
Rolls Royce	£373,300	£180,100
Triumph	£28,000	£5,850
TVR	£29,750	-£8,150
Germany		
Jaguar	£74,750	-£11,050
TVR	£71,000	-£4,800
Spain		
Bentley	£46,750	£13,480
Jaguar	£69,250	-£6,550
Triumph	£47,750	£16,600
TVR	£44,000	£1,100
Switzerland		
Aston Martin	£543,170	£242,675
Bentley	£222,750	£161,830
Jaguar	£450,250	£196,910
Rolls Royce	£192,300	£118,490
TVR	£32,500	-£10,400
United Kingdom		
Aston Martin	£4,933,500	£1,703,450
Bentley	£2,407,750	£798,100
Jaguar	£2,761,750	£369,440
MGB	£663,000	£447,500
Rolls Royce	£4,319,750	£1,844,680
Triumph	£530,250	£160,400
TVR	£109,000	-£19,700
USA		
Aston Martin	£3,722,160	£1,255,995

Figure 15-7. *A matrix without subtotals*

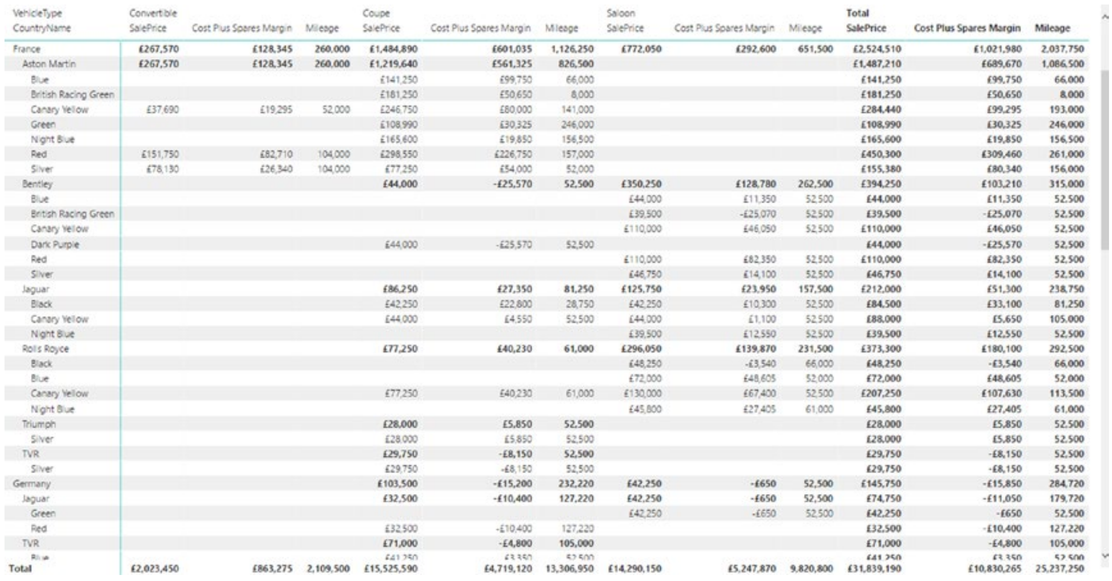
Column Matrix

Power BI Desktop does not limit you to adding row-level hierarchies; you can also create column-level hierarchies, or mix the two. Suppose that we want to get a clear idea of sales and gross margin by country, make, and vehicle type, and how they impact one another. To achieve this, I suggest extending the matrix that you created previously by adding a VehicleType level as a column hierarchy.

Here is how you can do this:

1. Click inside the matrix that you created previously to select it. The Visualizations pane will display the fields that are used for this table in the Rows and Values boxes.

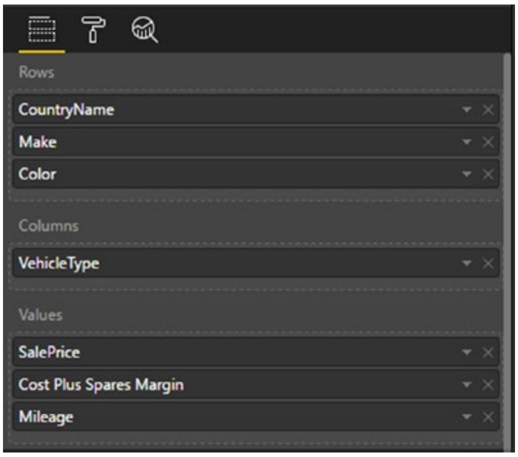
2. Drag the VehicleType field from the Stock table down into the Columns well in the Visualizations pane. This will add a hierarchy to the columns in the table. The table will look like Figure 15-8.



VehicleType	Convertible			Coupe			Saloon			Total		
CountryName	SalePrice	Cost Plus Spares Margin	Mileage	SalePrice	Cost Plus Spares Margin	Mileage	SalePrice	Cost Plus Spares Margin	Mileage	SalePrice	Cost Plus Spares Margin	Mileage
France	£267,570	£128,345	260,000	£1,484,890	£601,035	1,126,250	£772,050	£292,600	651,500	£2,524,510	£1,021,980	2,037,750
Aston Martin	£267,570	£128,345	260,000	£1,219,640	£561,325	826,500				£1,487,210	£689,670	1,086,500
Blue				£141,250	£99,750	66,000				£141,250	£99,750	66,000
British Racing Green				£181,250	£50,650	8,000				£181,250	£50,650	8,000
Canary Yellow	£37,690	£19,295	52,000	£246,750	£80,000	141,000				£284,440	£99,295	193,000
Green				£108,990	£30,325	246,000				£108,990	£30,325	246,000
Night Blue				£165,600	£19,850	156,500				£165,600	£19,850	156,500
Red	£151,750	£82,710	104,000	£298,550	£226,750	157,000				£450,300	£309,460	261,000
Silver	£78,130	£26,340	104,000	£77,250	£54,000	52,000				£155,380	£80,340	156,000
Bentley				£44,000	-£25,570	52,500	£350,250	£128,780	262,500	£394,250	£103,210	315,000
Blue							£44,000	£11,350	52,500	£44,000	£11,350	52,500
British Racing Green							£39,500	-£25,070	52,500	£39,500	-£25,070	52,500
Canary Yellow							£110,000	£46,050	52,500	£110,000	£46,050	52,500
Dark Purple				£44,000	-£25,570	52,500				£44,000	-£25,570	52,500
Red							£110,000	£82,350	52,500	£110,000	£82,350	52,500
Silver							£46,750	£14,100	52,500	£46,750	£14,100	52,500
Jaguar				£86,250	£27,350	81,250	£125,750	£23,950	157,500	£212,000	£51,300	238,750
Black				£42,250	£22,800	28,750	£42,250	£10,300	52,500	£84,500	£33,100	81,250
Canary Yellow				£44,000	£4,550	52,500	£44,000	£1,100	52,500	£88,000	£5,650	105,000
Night Blue							£39,500	£12,550	52,500	£39,500	£12,550	52,500
Rolls Royce				£77,250	£40,230	61,000	£296,050	£139,870	231,500	£373,300	£180,100	292,500
Black							£48,250	-£3,540	66,000	£48,250	-£3,540	66,000
Blue							£72,000	£48,605	52,000	£72,000	£48,605	52,000
Canary Yellow				£77,250	£40,230	61,000	£130,000	£67,400	52,500	£207,250	£107,630	113,500
Night Blue							£45,800	£27,405	61,000	£45,800	£27,405	61,000
Triumph				£28,000	£5,850	52,500				£28,000	£5,850	52,500
Silver				£28,000	£5,850	52,500				£28,000	£5,850	52,500
TVR				£29,750	-£8,150	52,500				£29,750	-£8,150	52,500
Silver				£29,750	-£8,150	52,500				£29,750	-£8,150	52,500
Germany				£103,500	-£15,200	232,220	£42,250	-£650	52,500	£145,750	-£15,850	284,720
Jaguar				£32,500	-£10,400	127,220	£42,250	-£650	52,500	£74,750	-£11,050	179,720
Green							£42,250	-£650	52,500	£42,250	-£650	52,500
Red				£32,500	-£10,400	127,220				£32,500	-£10,400	127,220
TVR				£71,000	-£4,800	105,000				£71,000	-£4,800	105,000
Blue				£41,250	£3,350	52,500				£41,250	£3,350	52,500
Total	£2,023,450	£863,275	2,109,500	£15,525,590	£4,719,120	13,306,950	£14,290,150	£5,247,870	9,820,800	£31,839,190	£10,830,265	25,237,250

Figure 15-8. *A row and column matrix table*

The Visualizations pane now looks like the snippet shown in Figure 15-9.



Rows
CountryName
Make
Color

Columns
VehicleType

Values
SalePrice
Cost Plus Spares Margin
Mileage

Figure 15-9. *The Visualizations pane for a row and column matrix table*

As you can see, you now have the sales and gross margin by country name, color, make, and vehicle type, but it is in a cross-matrix, where the data is broken down by both rows and columns.

To conclude the section on creating matrices, there are a few things that you might like to note:

- If you add totals, then *every level* of the hierarchy has totals.
- Adding non-numeric data to the aggregated data makes Power BI Desktop display the Count aggregation.
- Matrices can get very wide, especially if you have a multilevel hierarchy. Power BI Desktop matrices reflect this in the way in which horizontal scrolling works. A matrix table freezes the non-aggregated data columns on the left and allows you to scroll to the right to display aggregated (numeric) data.
- Moving the fields in the Columns and Rows wells of the Visualizations pane (using drag-and-drop, as described previously), reorders the aggregated data columns in the table.

Expanding and Drilling Down and Up

When creating the original matrix, you saw how to expand the matrix to include sublevels. However, Power BI Desktop offers a wide range of possibilities when it comes to navigating through layers of data in a matrix. These cover:

- Displaying one or all sublevels of data in a hierarchy
- Displaying only a deeper level of data
- Drilling down to display only lower levels of a specific item
- Drilling up to show the only previous level of a hierarchy

Navigating between levels of data is done using a combination of three interface elements:

- The Data/Drill ribbon
- The icons at the top of every matrix
- The context menu for a data element

It will probably help to get an overview of all three before learning to move through levels of data.

The Data/Drill Ribbon

One important tool when drilling through hierarchical data is the Data/Drill ribbon. The buttons that this ribbon contains are shown in Figure 15-10, and explained in Table 15-1.

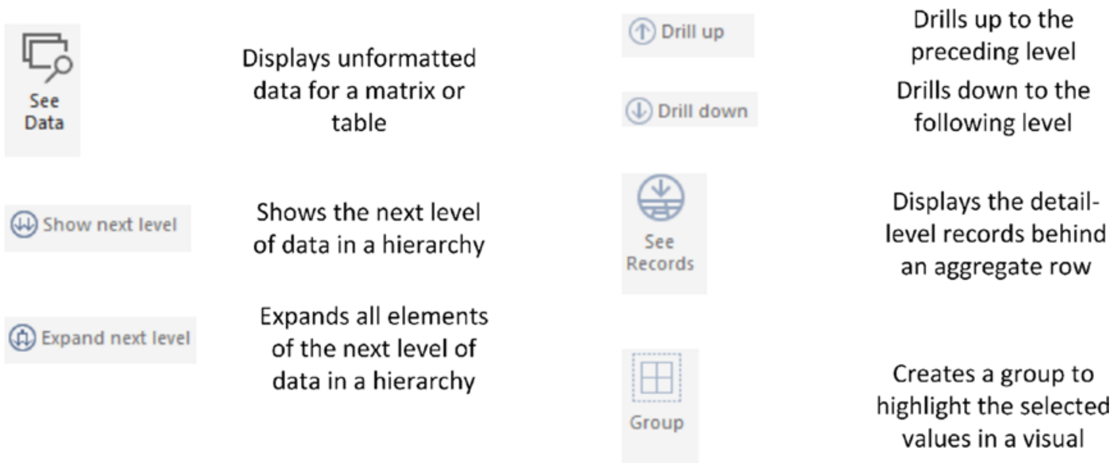

See Data
Displays unformatted data for a matrix or table
Show next level
Shows the next level of data in a hierarchy
Expand next level
Expands all elements of the next level of data in a hierarchy
Drill up
Drills up to the preceding level
Drill down
Drills down to the following level
See Records
Displays the detail-level records behind an aggregate row
Group
Creates a group to highlight the selected values in a visual

Figure 15-10. *The Data/Drill ribbon*

Table 15-1. *The Available Matrix Styles*

Button	Description
See Data	Shows the data, unformatted, in a separate pane.
Show Next Level	Displays the data segmented by the data at the next level in the hierarchy.
Expand Next Level	Shows all data items with the next level of data.
Drill Up	Returns to the previous data level (expanded or drilled down).
Drill Down	Shows the selected data item and any sublevel of data only for this item.
See Records	Displays the data for the records making up an aggregate row.
Group	Creates a group to highlight the selected values in a visual.

The Matrix Navigation Icons

Any matrix has a series of icons above the data. You can see these in Figure 15-11. I suggest that you familiarize yourself with these as they are fundamental when drilling through hierarchies of information. This applies not only to matrices, but to certain types of chart, too.

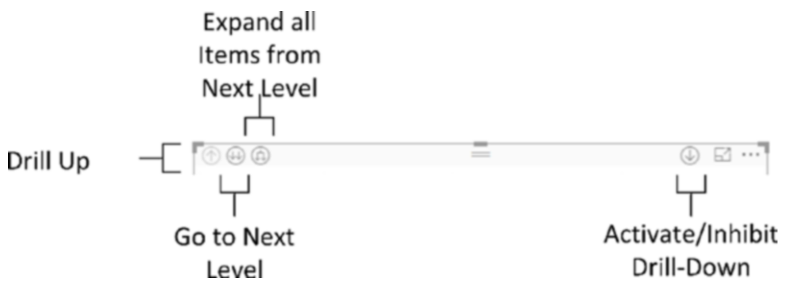


Figure 15-11. *The drill and expand icons for a matrix*

The Context Menu for Matrix Items

The final approach that you might need when navigating through hierarchies of data in matrices is the popup menu that appears when you right-click any record in a matrix. You can see this in Figure 15-12.

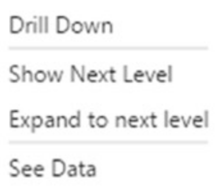


Figure 15-12. *The context menu for matrix items*

Displaying Data at the Previous Level

A few pages back you saw how to expand all the sublevels when creating a matrix, so I do not need to repeat this here. However, what if you are currently showing, say, three levels of a data hierarchy and want to return to displaying only two levels?

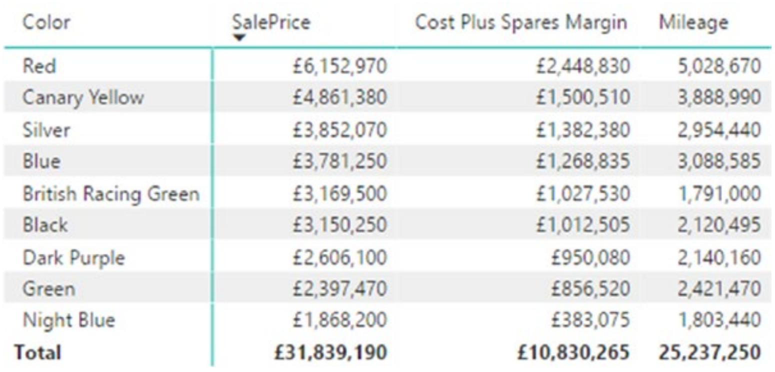
1. Take the matrix with three levels of data displayed that you can see in Figure 15-5.
2. In the Data/Drill ribbon, click Drill Up.

The matrix will show only two levels of data and look, once again, like the image in Figure 15-4.

Displaying Data for a Sublevel

A matrix does not oblige you to display all levels of data at once. You can, if you prefer, display only the data for a lower level of the matrix hierarchy, as follows:

1. Create the matrix that you began with in Figure 15-3, showing only the top-level data.
2. Click the "Go to the next level in the hierarchy" icon at the top of the matrix. Only the second level of data (color) will be displayed in the matrix. You can see an example of this in Figure 15-13.



Color	SalePrice	Cost Plus Spares Margin	Mileage
Red	£6,152,970	£2,448,830	5,028,670
Canary Yellow	£4,861,380	£1,500,510	3,888,990
Silver	£3,852,070	£1,382,380	2,954,440
Blue	£3,781,250	£1,268,835	3,088,585
British Racing Green	£3,169,500	£1,027,530	1,791,000
Black	£3,150,250	£1,012,505	2,120,495
Dark Purple	£2,606,100	£950,080	2,140,160
Green	£2,397,470	£856,520	2,421,470
Night Blue	£1,868,200	£383,075	1,803,440
Total	**£31,839,190**	**£10,830,265**	**25,237,250**

Figure 15-13. *Displaying a different level of data from a matrix*

As you can see, the totals are identical. Only the data segmentation has changed.

To return to the previous level in the data hierarchy, simply click the Drill Up button in the Data/Drill ribbon—or the drill up icon at the top left of the matrix.

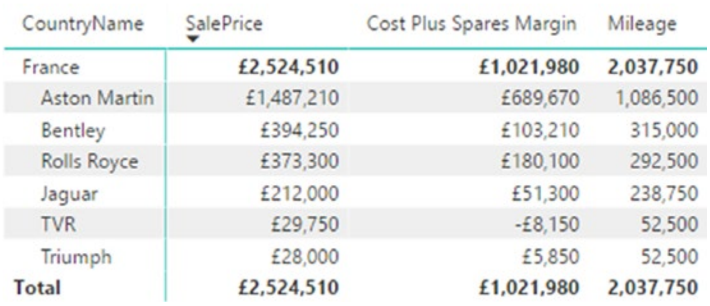
Drilling Down at Row Level

Another completely different option when navigating hierarchies of data is to drill up and down into a specific element of a hierarchy. The key difference between expanding and drilling are as follows:

- Expanding a hierarchy will display the next level of data for *all* top-level elements.
- Drilling down will only display the next sublevel for the *selected* data element.

To see this in practice, try the following:

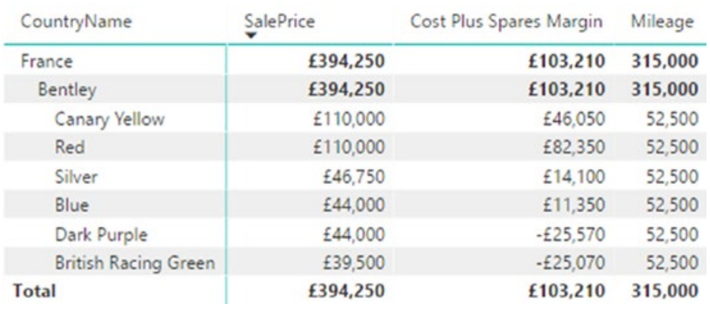
1. Right-click any top-level data element in a matrix (I will take France in the matrix that you can see in Figure 15-5).
2. Select Drill Down from the context menu. The matrix will now look like the one in Figure 15-14.



CountryName	SalePrice	Cost Plus Spares Margin	Mileage
France	**£2,524,510**	**£1,021,980**	**2,037,750**
Aston Martin	£1,487,210	£689,670	1,086,500
Bentley	£394,250	£103,210	315,000
Rolls Royce	£373,300	£180,100	292,500
Jaguar	£212,000	£51,300	238,750
TVR	£29,750	-£8,150	52,500
Triumph	£28,000	£5,850	52,500
Total	**£2,524,510**	**£1,021,980**	**2,037,750**

Figure 15-14. *Drilling down in a matrix*

3. You can continue to drill down into further levels of data simply by right-clicking any data element at the second level and selecting Drill Down from the context menu (Bentley, for instance, in this example). The matrix will now look like the one in Figure 15-15.



CountryName	SalePrice	Cost Plus Spares Margin	Mileage
France	**£394,250**	**£103,210**	**315,000**
Bentley	**£394,250**	**£103,210**	**315,000**
Canary Yellow	£110,000	£46,050	52,500
Red	£110,000	£82,350	52,500
Silver	£46,750	£14,100	52,500
Blue	£44,000	£11,350	52,500
Dark Purple	£44,000	-£25,570	52,500
British Racing Green	£39,500	-£25,070	52,500
Total	**£394,250**	**£103,210**	**315,000**

Figure 15-15. *Drilling down further in a matrix*

To return to a previous level, all you have to do is click the Drill Up button in the Data/Drill ribbon—or the drill up icon at the top left of the matrix. Alternatively, you can right-click a data item and select Drill Up from the context menu.

Drill Down Using Click-Through

If you find it a little wearing to right-click and select the Drill Down option, then you have an alternative solution. You can switch a matrix to drill down with a simple click on a data item. Here is how you can do this:

1. At the top right of a matrix, click the "Click to turn on drill down" icon. This will become a light arrow on a dark background.
2. Click any data element (a country, for instance). You will drill down to the next level.

You can turn off this option at any time by re-clicking the "Click to turn on drill down" icon. It will return to being a light arrow on a clear background, indicating that this form of drill down is deactivated.

Drilling Down at Column Level

The same drill-down and drill-up logic applies to columns as to rows. Take the column matrix that you created previously (and that you can see in Figure 15-8):

1. Add the Model (from the Stock table) to the data for the matrix to the Columns well, under the VehicleType field. A popup appears at the top left of the matrix containing the choice between rows and columns. You can see this in Figure 15-16.

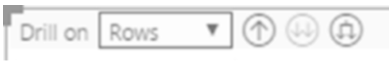
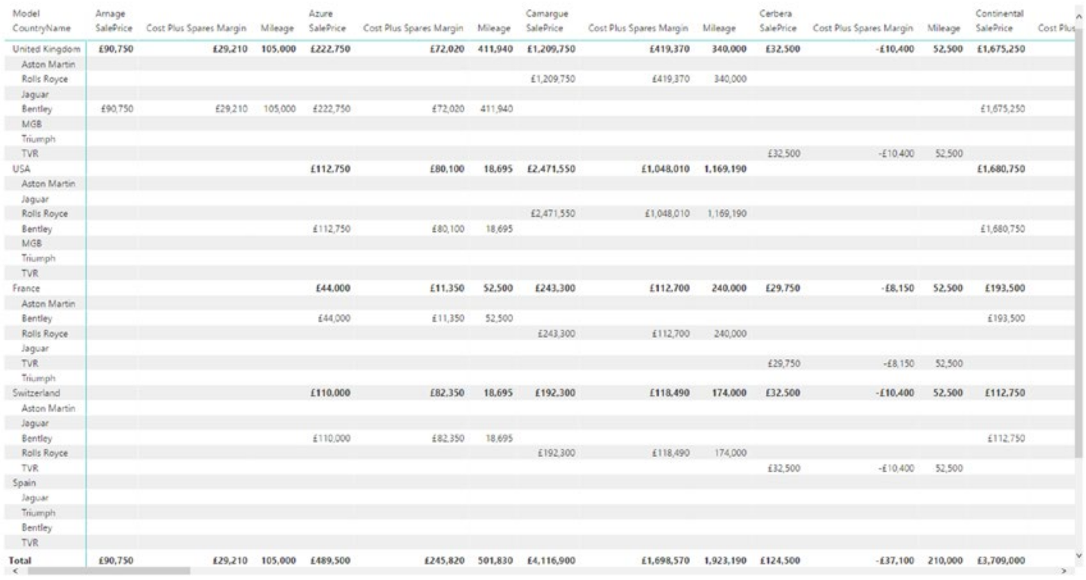


Figure 15-16. *The drill-down option in complex matrices*

2. Click the popup and select Columns.

3. Click the "Go to the next level in the hierarchy" icon. You will see something like Figure 15-17, where the country has been replaced by make as the column header.



Model	Arnage			Azure			Camargue			Cerbera			Continental	
CountryName	SalePrice	Cost Plus Spares Margin	Mileage	SalePrice	Cost Plus Spares Margin	Mileage	SalePrice	Cost Plus Spares Margin	Mileage	SalePrice	Cost Plus Spares Margin	Mileage	SalePrice	Cost Plus
United Kingdom	£90,750	£29,210	105,000	£222,750	£72,020	411,940	£1,209,750	£419,370	340,000	£32,500	-£10,400	52,500	£1,675,250	
Aston Martin														
Rolls Royce							£1,209,750	£419,370	340,000					
Jaguar														
Bentley	£90,750	£29,210	105,000	£222,750	£72,020	411,940							£1,675,250	
MGB														
Triumph														
TVR										£32,500	-£10,400	52,500		
USA				£112,750	£80,100	18,695	£2,471,550	£1,048,010	1,169,190				£1,680,750	
Aston Martin														
Jaguar														
Rolls Royce							£2,471,550	£1,048,010	1,169,190					
Bentley				£112,750	£80,100	18,695							£1,680,750	
MGB														
Triumph														
TVR														
France				£44,000	£11,350	52,500	£243,300	£112,700	240,000	£29,750	-£8,150	52,500	£193,500	
Aston Martin														
Bentley				£44,000	£11,350	52,500							£193,500	
Rolls Royce							£243,300	£112,700	240,000					
Jaguar														
TVR										£29,750	-£8,150	52,500		
Triumph														
Switzerland				£110,000	£82,350	18,695	£192,300	£118,490	174,000	£32,500	-£10,400	52,500	£112,750	
Aston Martin														
Jaguar														
Bentley				£110,000	£82,350	18,695							£112,750	
Rolls Royce							£192,300	£118,490	174,000					
TVR										£32,500	-£10,400	52,500		
Spain														
Jaguar														
Triumph														
Bentley														
TVR														
Total	£90,750	£29,210	105,000	£489,500	£245,820	501,830	£4,116,900	£1,698,570	1,923,190	£124,500	-£37,100	210,000	£3,709,000	

Figure 15-17. *Drilling down in a multilevel column hierarchy*

As you can see, a hierarchy can be applied equally to columns and rows. This boils down to

- Adding the field to the correct well in the Visualizations pane (Rows or Columns).
- Selecting the appropriate drill-down mode from the popup at the top of the matrix (Columns or Rows).

Visualize Source Data

If a formatted table or matrix becomes, paradoxically, too large or complex, you can opt to look at the underlying data without any formatting applied. This often lets you see more information than you can in a carefully formatted visualization.

1. Select the table or matrix whose data you want to display without formatting.
2. In the Data/Drill ribbon, click the See Data button. The unformatted version of the data for the selected table or matrix will be displayed in a second window. You can see this in Figure 15-18.

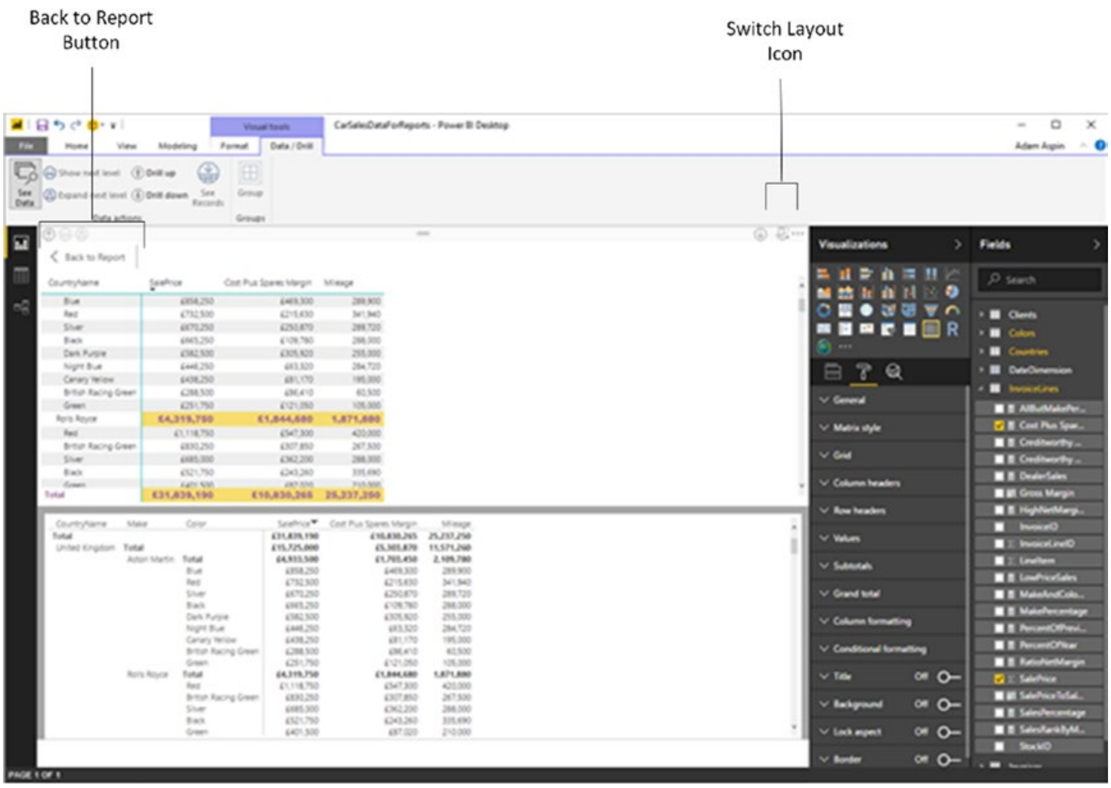


Figure 15-18. *Visualizing source data*

3. Click the Back to Report button above the formatted table (or click again the See Data button in the Data/Drill ribbon) to return to the normal Power BI Desktop Report View.

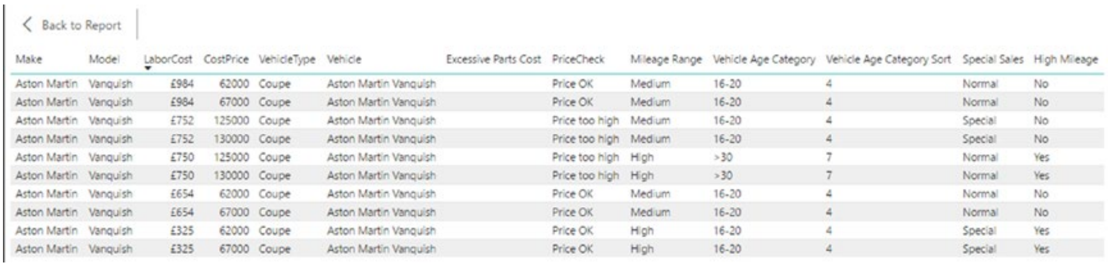
■ **Note** You cannot currently resize the two windows when viewing unformatted data. You can, however, scroll up and down in either window. If you prefer to see the data to the right of the visual (rather than underneath it), then click the Switch Layout icon at the top right of the window.

Viewing Records

When dealing with aggregated data, you can sometimes lose sight of valuable details. You might miss a large outlier that is skewing an average value, for instance. Or, perhaps, an excessive value for a single record is making a total higher than is normal—or healthy.

To help you follow your intuitions and weight aggregated values against detailed numbers, Power BI Desktop lets you view the underlying records "behind" a total at any time in a matrix. Here's an example:

1. Create a matrix based on the following fields (all of them are from the Stock table):
   a. Make
   b. Model
   c. LaborCost
   d. CostPrice
2. Click Expand Next Level to see the complete hierarchy of makes and models.
3. Click the figure for the labor cost for Aston Martin Vanquish. You will see all the detail records that make up this total figure. This is shown in Figure 15-19.



Make	Model	LaborCost	CostPrice	VehicleType	Vehicle	Excessive Parts Cost	PriceCheck	Mileage Range	Vehicle Age Category	Vehicle Age Category Sort	Special Sales	High Mileage
Aston Martin	Vanquish	£984	62000	Coupe	Aston Martin Vanquish		Price OK	Medium	16-20	4	Normal	No
Aston Martin	Vanquish	£984	67000	Coupe	Aston Martin Vanquish		Price OK	Medium	16-20	4	Normal	No
Aston Martin	Vanquish	£752	125000	Coupe	Aston Martin Vanquish		Price too high	Medium	16-20	4	Special	No
Aston Martin	Vanquish	£752	130000	Coupe	Aston Martin Vanquish		Price too high	Medium	16-20	4	Special	No
Aston Martin	Vanquish	£750	125000	Coupe	Aston Martin Vanquish		Price too high	High	>30	7	Normal	Yes
Aston Martin	Vanquish	£750	130000	Coupe	Aston Martin Vanquish		Price too high	High	>30	7	Normal	Yes
Aston Martin	Vanquish	£654	62000	Coupe	Aston Martin Vanquish		Price OK	Medium	16-20	4	Normal	No
Aston Martin	Vanquish	£654	67000	Coupe	Aston Martin Vanquish		Price OK	Medium	16-20	4	Normal	No
Aston Martin	Vanquish	£325	62000	Coupe	Aston Martin Vanquish		Price OK	High	16-20	4	Special	Yes
Aston Martin	Vanquish	£325	67000	Coupe	Aston Martin Vanquish		Price OK	High	16-20	4	Special	Yes

Figure 15-19. *Displaying data records from a matrix*

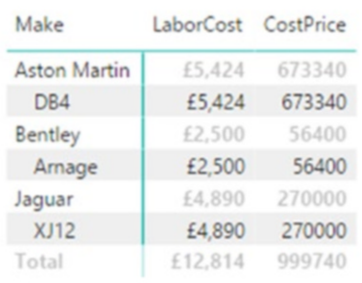
4. Click the Back to Report button above the records that are now displayed to return to the normal Power BI Desktop Report View.

■ **Note** Alternatively, you can right-click any value in a matrix and select See Records to drill through to the detailed records that give the selected aggregate figure.

Including and Excluding Matrix Elements

Some matrices can swamp you in data. So you might need to focus on a few selected elements. This is really simple in Power BI Desktop:

1. Re-create the matrix from the previous section (Make, Model, LaborCost, and CostPrice).
2. Click Expand Next Level to see the complete hierarchy of makes and models.
3. Control-click any element (make, model, or a value) for:
    a. Aston Martin DB4
    b. Bentley Arnage
    c. Jaguar XJ12
4. Right-click one of the selected elements and select Include from the popup menu. All other elements will be removed from the matrix. The result is shown in Figure 15-20.



Make	LaborCost	CostPrice
Aston Martin	£5,424	673340
DB4	£5,424	673340
Bentley	£2,500	56400
Arnage	£2,500	56400
Jaguar	£4,890	270000
XJ12	£4,890	270000
Total	£12,814	999740

Figure 15-20. *Including selected elements from a matrix*

■ **Note** As an alternative to including certain elements, you can select all the elements that you wish to *exclude*, and choose Exclude from the popup menu.

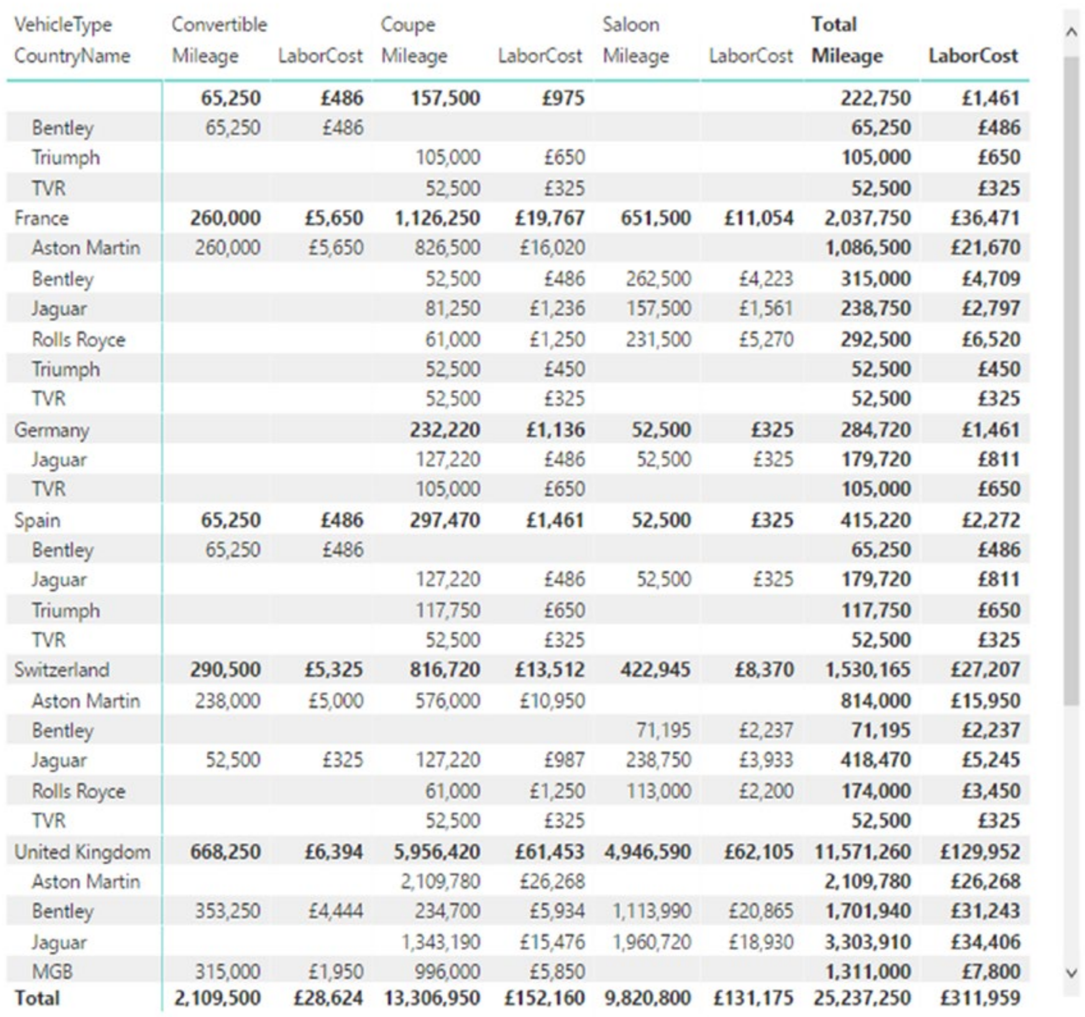
For the moment, the only way to return to a complete matrix containing all the rows of data seems to be to press Ctrl-Z and undo this operation.

Displaying Multiple Values As Rows

A feature that has been added to Power BI Desktop fairly recently is the possibility of creating complex matrices where the columns of data can now be displayed as rows. That is, instead of seeing possibly multiple data values side by side for a complex column hierarchy, you can see all the numeric data displayed as rows.

This is possibly best appreciated with the help of an example:

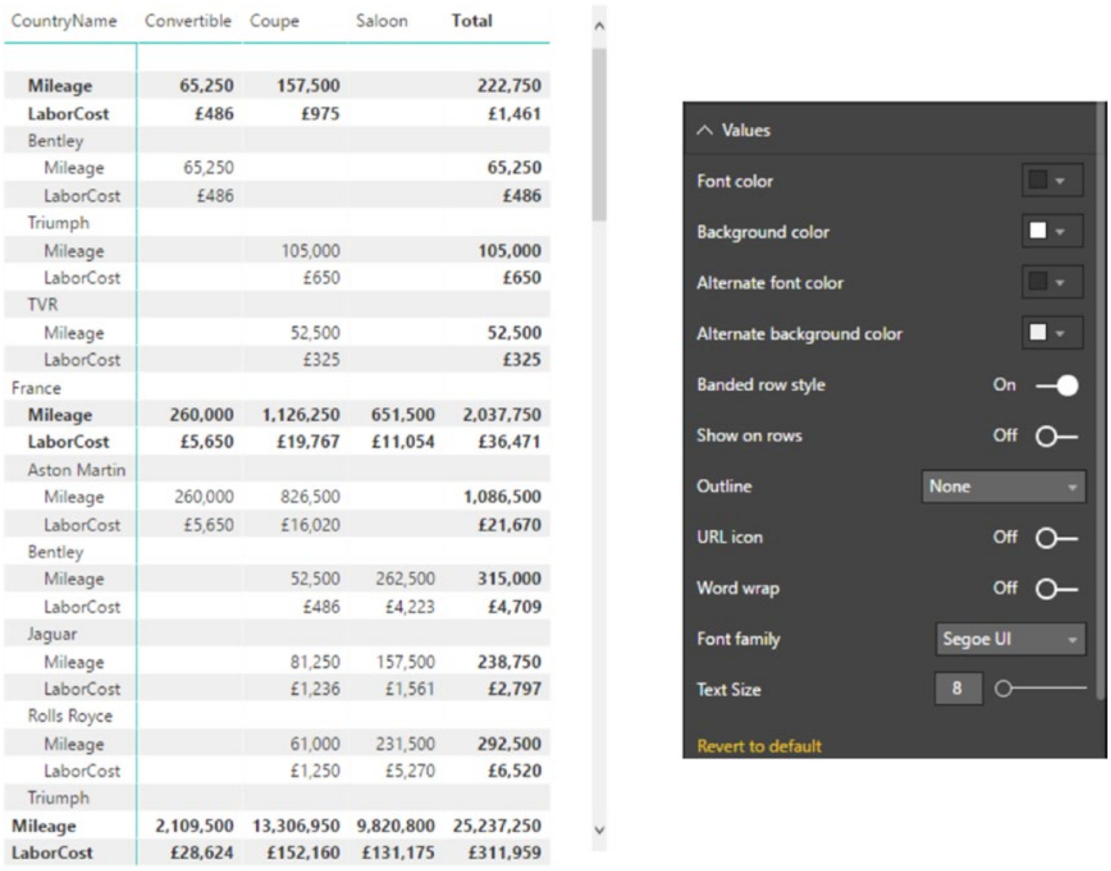
1. Create a matrix using the following elements:
   a. Rows: CountryName from the Countries table and Make from the Stock table (in this order).
   b. Column: VehicleType (from the Stock table).
   c. Values: Mileage and LaborCost (in this order). Both are from the Stock table.
2. Expand all down one level. The matrix will look like the one shown in Figure 15-21.



VehicleType	Convertible		Coupe		Saloon		**Total**	
CountryName	Mileage	LaborCost	Mileage	LaborCost	Mileage	LaborCost	**Mileage**	**LaborCost**
	65,250	**£486**	**157,500**	**£975**			**222,750**	**£1,461**
Bentley	65,250	£486					**65,250**	**£486**
Triumph			105,000	£650			**105,000**	**£650**
TVR			52,500	£325			**52,500**	**£325**
France	**260,000**	**£5,650**	**1,126,250**	**£19,767**	**651,500**	**£11,054**	**2,037,750**	**£36,471**
Aston Martin	260,000	£5,650	826,500	£16,020			**1,086,500**	**£21,670**
Bentley			52,500	£486	262,500	£4,223	**315,000**	**£4,709**
Jaguar			81,250	£1,236	157,500	£1,561	**238,750**	**£2,797**
Rolls Royce			61,000	£1,250	231,500	£5,270	**292,500**	**£6,520**
Triumph			52,500	£450			**52,500**	**£450**
TVR			52,500	£325			**52,500**	**£325**
Germany			**232,220**	**£1,136**	**52,500**	**£325**	**284,720**	**£1,461**
Jaguar			127,220	£486	52,500	£325	**179,720**	**£811**
TVR			105,000	£650			**105,000**	**£650**
Spain	**65,250**	**£486**	**297,470**	**£1,461**	**52,500**	**£325**	**415,220**	**£2,272**
Bentley	65,250	£486					**65,250**	**£486**
Jaguar			127,220	£486	52,500	£325	**179,720**	**£811**
Triumph			117,750	£650			**117,750**	**£650**
TVR			52,500	£325			**52,500**	**£325**
Switzerland	**290,500**	**£5,325**	**816,720**	**£13,512**	**422,945**	**£8,370**	**1,530,165**	**£27,207**
Aston Martin	238,000	£5,000	576,000	£10,950			**814,000**	**£15,950**
Bentley					71,195	£2,237	**71,195**	**£2,237**
Jaguar	52,500	£325	127,220	£987	238,750	£3,933	**418,470**	**£5,245**
Rolls Royce			61,000	£1,250	113,000	£2,200	**174,000**	**£3,450**
TVR			52,500	£325			**52,500**	**£325**
United Kingdom	**668,250**	**£6,394**	**5,956,420**	**£61,453**	**4,946,590**	**£62,105**	**11,571,260**	**£129,952**
Aston Martin			2,109,780	£26,268			**2,109,780**	**£26,268**
Bentley	353,250	£4,444	234,700	£5,934	1,113,990	£20,865	**1,701,940**	**£31,243**
Jaguar			1,343,190	£15,476	1,960,720	£18,930	**3,303,910**	**£34,406**
MGB	315,000	£1,950	996,000	£5,850			**1,311,000**	**£7,800**
Total	**2,109,500**	**£28,624**	**13,306,950**	**£152,160**	**9,820,800**	**£131,175**	**25,237,250**	**£311,959**

Figure 15-21. *A standard matrix with values as columns*

3. In the Visualizations pane, click the Format icon and expand the Values section.
4. Set Show on Rows to On. The matrix will now look like the one shown in Figure 15-22, where the multiple columns of data have become rows of data.



CountryName	Convertible	Coupe	Saloon	Total
Mileage	65,250	157,500		222,750
LaborCost	£486	£975		£1,461
Bentley				
Mileage	65,250			65,250
LaborCost	£486			£486
Triumph				
Mileage		105,000		105,000
LaborCost		£650		£650
TVR				
Mileage		52,500		52,500
LaborCost		£325		£325
France				
Mileage	260,000	1,126,250	651,500	2,037,750
LaborCost	£5,650	£19,767	£11,054	£36,471
Aston Martin				
Mileage	260,000	826,500		1,086,500
LaborCost	£5,650	£16,020		£21,670
Bentley				
Mileage		52,500	262,500	315,000
LaborCost		£486	£4,223	£4,709
Jaguar				
Mileage		81,250	157,500	238,750
LaborCost		£1,236	£1,561	£2,797
Rolls Royce				
Mileage		61,000	231,500	292,500
LaborCost		£1,250	£5,270	£6,520
Triumph				
Mileage	2,109,500	13,306,950	9,820,800	25,237,250
LaborCost	£28,624	£152,160	£131,175	£311,959

Values

Font color	
Background color	
Alternate font color	
Alternate background color	
Banded row style	On
Show on rows	Off
Outline	None
URL icon	Off
Word wrap	Off
Font family	Segoe UI
Text Size	8

Revert to default

Figure 15-22. *A matrix with values as rows*

Simply switching Show on Rows to Off will revert the matrix to its initial format.

Formatting a Matrix

Fortunately, the techniques that you apply to format a matrix are largely identical to those you applied to tables. So I will not cover, again, the principles that you saw in the last few pages to change the presentation of:

- Matrix style (this is the table style, but applied to a matrix)
- Grid
- Column headers
- Row headers
- Values
- Columns
- Title
- Background
- Conditional formatting

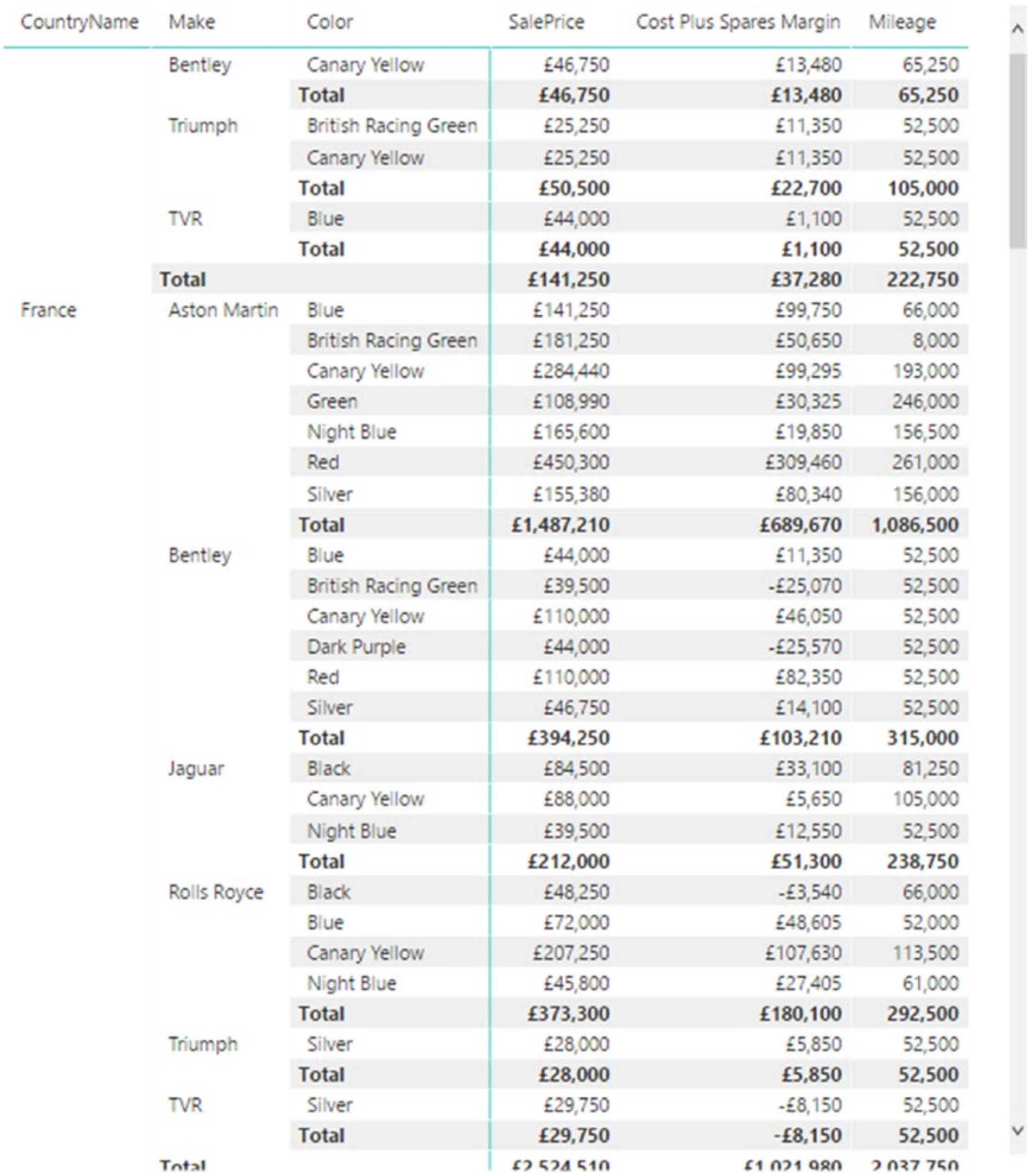
There are, nonetheless, a couple of extra formatting tweaks that you can add to matrices. These are stepped layout and subtotals. I will explain these in the following two sections.

Stepped Layout

In Figure 15-5 you can see the multiple levels of a matrix. These are shown using levels of indentation. However, you may prefer to display each level in a separate column. Here is how you can do this:

1. Select (or re-create) the matrix containing three visible levels that you can see in Figure 15-5.
2. In the Visualizations pane, click the Format icon.
3. Expand the Row Headers section.
4. Set Stepped Layout to Off.

5. Adjust the width of the matrix if necessary. The matrix will look like the one in Figure 15-23.



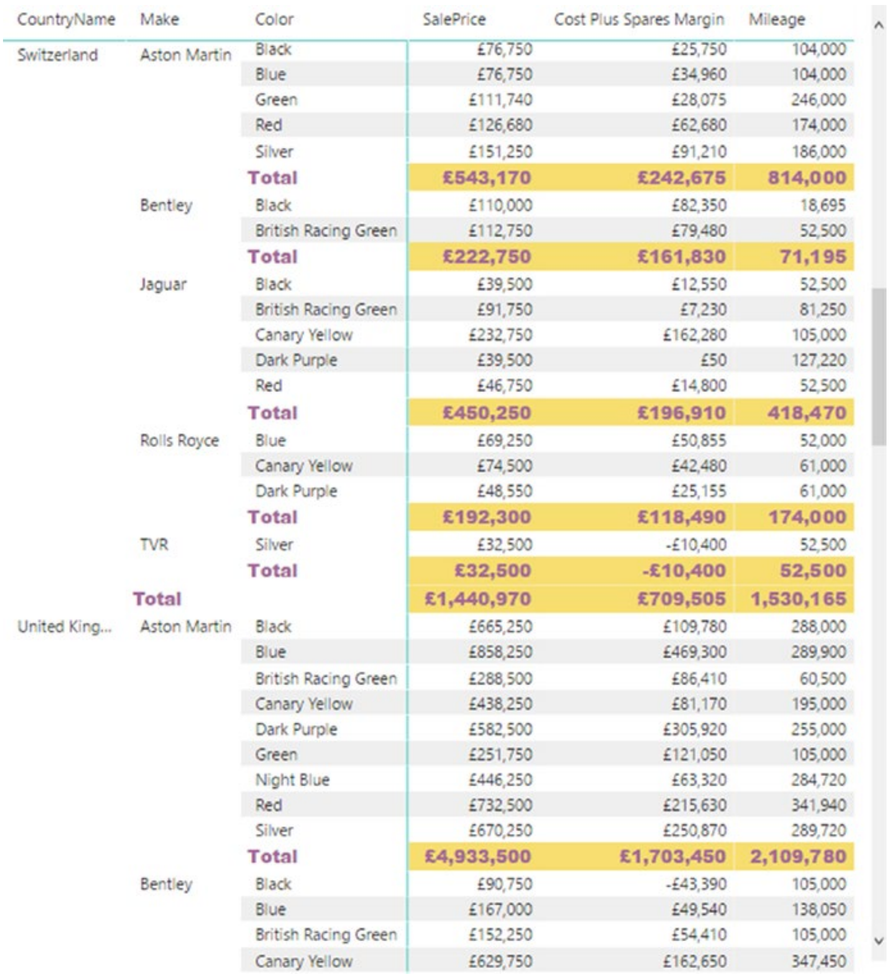
CountryName	Make	Color	SalePrice	Cost Plus Spares Margin	Mileage
	Bentley	Canary Yellow	£46,750	£13,480	65,250
		Total	**£46,750**	**£13,480**	**65,250**
	Triumph	British Racing Green	£25,250	£11,350	52,500
		Canary Yellow	£25,250	£11,350	52,500
		Total	**£50,500**	**£22,700**	**105,000**
	TVR	Blue	£44,000	£1,100	52,500
		Total	**£44,000**	**£1,100**	**52,500**
	Total		**£141,250**	**£37,280**	**222,750**
France	Aston Martin	Blue	£141,250	£99,750	66,000
		British Racing Green	£181,250	£50,650	8,000
		Canary Yellow	£284,440	£99,295	193,000
		Green	£108,990	£30,325	246,000
		Night Blue	£165,600	£19,850	156,500
		Red	£450,300	£309,460	261,000
		Silver	£155,380	£80,340	156,000
		Total	**£1,487,210**	**£689,670**	**1,086,500**
	Bentley	Blue	£44,000	£11,350	52,500
		British Racing Green	£39,500	-£25,070	52,500
		Canary Yellow	£110,000	£46,050	52,500
		Dark Purple	£44,000	-£25,570	52,500
		Red	£110,000	£82,350	52,500
		Silver	£46,750	£14,100	52,500
		Total	**£394,250**	**£103,210**	**315,000**
	Jaguar	Black	£84,500	£33,100	81,250
		Canary Yellow	£88,000	£5,650	105,000
		Night Blue	£39,500	£12,550	52,500
		Total	**£212,000**	**£51,300**	**238,750**
	Rolls Royce	Black	£48,250	-£3,540	66,000
		Blue	£72,000	£48,605	52,000
		Canary Yellow	£207,250	£107,630	113,500
		Night Blue	£45,800	£27,405	61,000
		Total	**£373,300**	**£180,100**	**292,500**
	Triumph	Silver	£28,000	£5,850	52,500
		Total	**£28,000**	**£5,850**	**52,500**
	TVR	Silver	£29,750	-£8,150	52,500
		Total	**£29,750**	**-£8,150**	**52,500**
	Total		**£2,524,510**	**£1,021,980**	**2,037,750**

Figure 15-23. *A matrix without stepped layout*

Subtotals

Whereas a table only has a grand total (assuming that you want to display it), a matrix can have subtotals for each level of data displayed. Although formatting subtotals is similar to formatting the total in a table, I prefer, for the sake of completeness, to describe the possibilities here:

1. Select the matrix visual that you used in the previous section.
2. In the Visualizations pane, click the Format icon.
3. Expand the Subtotals section.
4. Choose a different font color from the Font Color palette.
5. Select a different font from the Font Family popup list.
6. Enter a number in the Text Size field (or adjust the slider) to set a larger font size.
7. Set Apply to Labels to On—this will ensure that the subtotal labels as well as numbers are formatted. The matrix should look like the one in Figure 15-24.



CountryName	Make	Color	SalePrice	Cost Plus Spares Margin	Mileage
Switzerland	Aston Martin	Black	£76,750	£25,750	104,000
		Blue	£76,750	£34,960	104,000
		Green	£111,740	£28,075	246,000
		Red	£126,680	£62,680	174,000
		Silver	£151,250	£91,210	186,000
		Total	**£543,170**	**£242,675**	**814,000**
	Bentley	Black	£110,000	£82,350	18,695
		British Racing Green	£112,750	£79,480	52,500
		Total	**£222,750**	**£161,830**	**71,195**
	Jaguar	Black	£39,500	£12,550	52,500
		British Racing Green	£91,750	£7,230	81,250
		Canary Yellow	£232,750	£162,280	105,000
		Dark Purple	£39,500	£50	127,220
		Red	£46,750	£14,800	52,500
		Total	**£450,250**	**£196,910**	**418,470**
	Rolls Royce	Blue	£69,250	£50,855	52,000
		Canary Yellow	£74,500	£42,480	61,000
		Dark Purple	£48,550	£25,155	61,000
		Total	**£192,300**	**£118,490**	**174,000**
	TVR	Silver	£32,500	-£10,400	52,500
		Total	**£32,500**	**-£10,400**	**52,500**
	Total		**£1,440,970**	**£709,505**	**1,530,165**
United King...	Aston Martin	Black	£665,250	£109,780	288,000
		Blue	£858,250	£469,300	289,900
		British Racing Green	£288,500	£86,410	60,500
		Canary Yellow	£438,250	£81,170	195,000
		Dark Purple	£582,500	£305,920	255,000
		Green	£251,750	£121,050	105,000
		Night Blue	£446,250	£63,320	284,720
		Red	£732,500	£215,630	341,940
		Silver	£670,250	£250,870	289,720
		Total	**£4,933,500**	**£1,703,450**	**2,109,780**
	Bentley	Black	£90,750	-£43,390	105,000
		Blue	£167,000	£49,540	138,050
		British Racing Green	£152,250	£54,410	105,000
		Canary Yellow	£629,750	£162,650	347,450

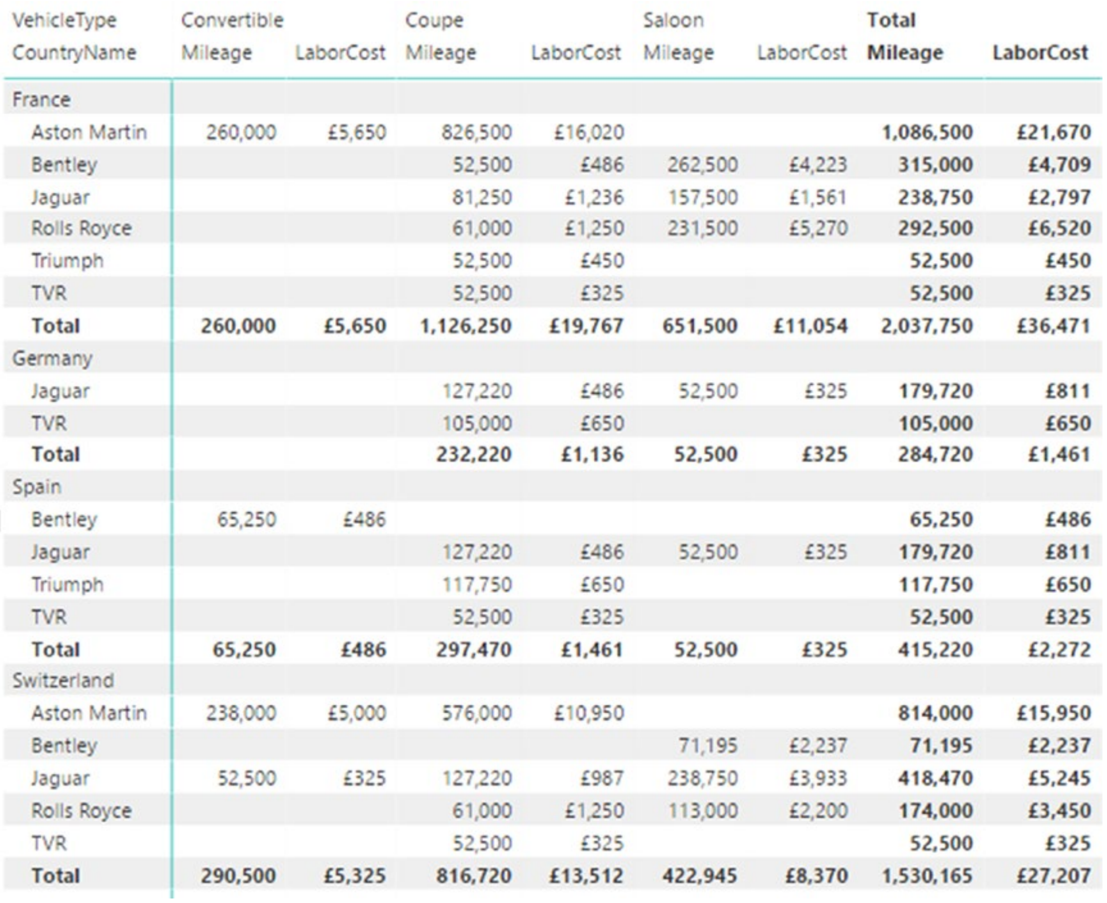
***Figure 15-24.** Formatting subtotals in a matrix*

■ **Note** As the grand total is essentially formatted in the same way as the Total for a table, I will not re-explain how to do this here. If you need to refresh your memory, then just flip back to the previous chapter.

Placing Subtotals

You can also decide whether the subtotals are placed above a group of elements, alongside the element header, or under a group, on a separate row. Simply do the following to place subtotals:

1. Re-create the matrix that you used in Figure 15-21 (following only steps 1 and 2).
2. In the Visualizations pane, click the Format icon and expand the Subtotals section.
3. In the popup for Row Subtotal Position, select Bottom. The matrix should now look like the one in Figure 15-25.



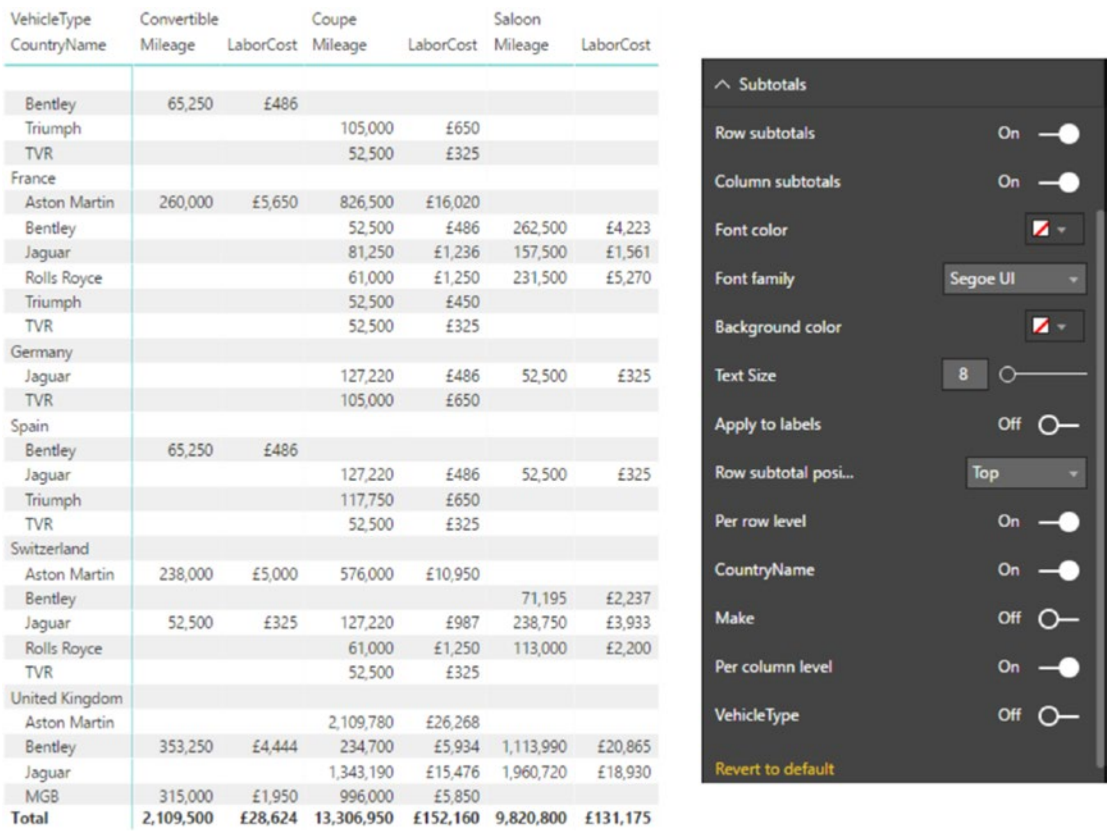
VehicleType	Convertible		Coupe		Saloon		**Total**	
CountryName	Mileage	LaborCost	Mileage	LaborCost	Mileage	LaborCost	**Mileage**	**LaborCost**
France								
Aston Martin	260,000	£5,650	826,500	£16,020			**1,086,500**	**£21,670**
Bentley			52,500	£486	262,500	£4,223	**315,000**	**£4,709**
Jaguar			81,250	£1,236	157,500	£1,561	**238,750**	**£2,797**
Rolls Royce			61,000	£1,250	231,500	£5,270	**292,500**	**£6,520**
Triumph			52,500	£450			**52,500**	**£450**
TVR			52,500	£325			**52,500**	**£325**
Total	**260,000**	**£5,650**	**1,126,250**	**£19,767**	**651,500**	**£11,054**	**2,037,750**	**£36,471**
Germany								
Jaguar			127,220	£486	52,500	£325	**179,720**	**£811**
TVR			105,000	£650			**105,000**	**£650**
Total			**232,220**	**£1,136**	**52,500**	**£325**	**284,720**	**£1,461**
Spain								
Bentley	65,250	£486					**65,250**	**£486**
Jaguar			127,220	£486	52,500	£325	**179,720**	**£811**
Triumph			117,750	£650			**117,750**	**£650**
TVR			52,500	£325			**52,500**	**£325**
Total	**65,250**	**£486**	**297,470**	**£1,461**	**52,500**	**£325**	**415,220**	**£2,272**
Switzerland								
Aston Martin	238,000	£5,000	576,000	£10,950			**814,000**	**£15,950**
Bentley					71,195	£2,237	**71,195**	**£2,237**
Jaguar	52,500	£325	127,220	£987	238,750	£3,933	**418,470**	**£5,245**
Rolls Royce			61,000	£1,250	113,000	£2,200	**174,000**	**£3,450**
TVR			52,500	£325			**52,500**	**£325**
Total	**290,500**	**£5,325**	**816,720**	**£13,512**	**422,945**	**£8,370**	**1,530,165**	**£27,207**

Figure 15-25. *Placing subtotals below groups*

Custom Subtotal Settings per Level of Matrix

Power BI Desktop now lets you define whether subtotals are displayed for each level of the matrix hierarchy for both rows and columns. To see this in action:

1. Re-create the matrix that you used in Figure 15-21 (following only steps 1 and 2).
2. In the Visualizations pane, click the Format icon and expand the Subtotals section.
3. Set Per Row Level to On. This will add the list of all the row data elements to the Visualizations pane.
4. Set Per Column Level to On. This will add the list of all the column data elements to the Visualizations pane.
5. Set the button to Off for both Make and VehicleType. As you can see in Figure 15-26, the subtotals have been removed for these elements.



VehicleType	Convertible		Coupe		Saloon	
CountryName	Mileage	LaborCost	Mileage	LaborCost	Mileage	LaborCost
Bentley	65,250	£486				
Triumph			105,000	£650		
TVR			52,500	£325		
France						
Aston Martin	260,000	£5,650	826,500	£16,020		
Bentley			52,500	£486	262,500	£4,223
Jaguar			81,250	£1,236	157,500	£1,561
Rolls Royce			61,000	£1,250	231,500	£5,270
Triumph			52,500	£450		
TVR			52,500	£325		
Germany						
Jaguar			127,220	£486	52,500	£325
TVR			105,000	£650		
Spain						
Bentley	65,250	£486				
Jaguar			127,220	£486	52,500	£325
Triumph			117,750	£650		
TVR			52,500	£325		
Switzerland						
Aston Martin	238,000	£5,000	576,000	£10,950		
Bentley					71,195	£2,237
Jaguar	52,500	£325	127,220	£987	238,750	£3,933
Rolls Royce			61,000	£1,250	113,000	£2,200
TVR			52,500	£325		
United Kingdom						
Aston Martin			2,109,780	£26,268		
Bentley	353,250	£4,444	234,700	£5,934	1,113,990	£20,865
Jaguar			1,343,190	£15,476	1,960,720	£18,930
MGB	315,000	£1,950	996,000	£5,850		
Total	**2,109,500**	**£28,624**	**13,306,950**	**£152,160**	**9,820,800**	**£131,175**

Subtotals

Row subtotals	On
Column subtotals	On
Font color	
Font family	Segoe UI
Background color	
Text Size	8
Apply to labels	Off
Row subtotal posi...	Top
Per row level	On
CountryName	On
Make	Off
Per column level	On
VehicleType	Off

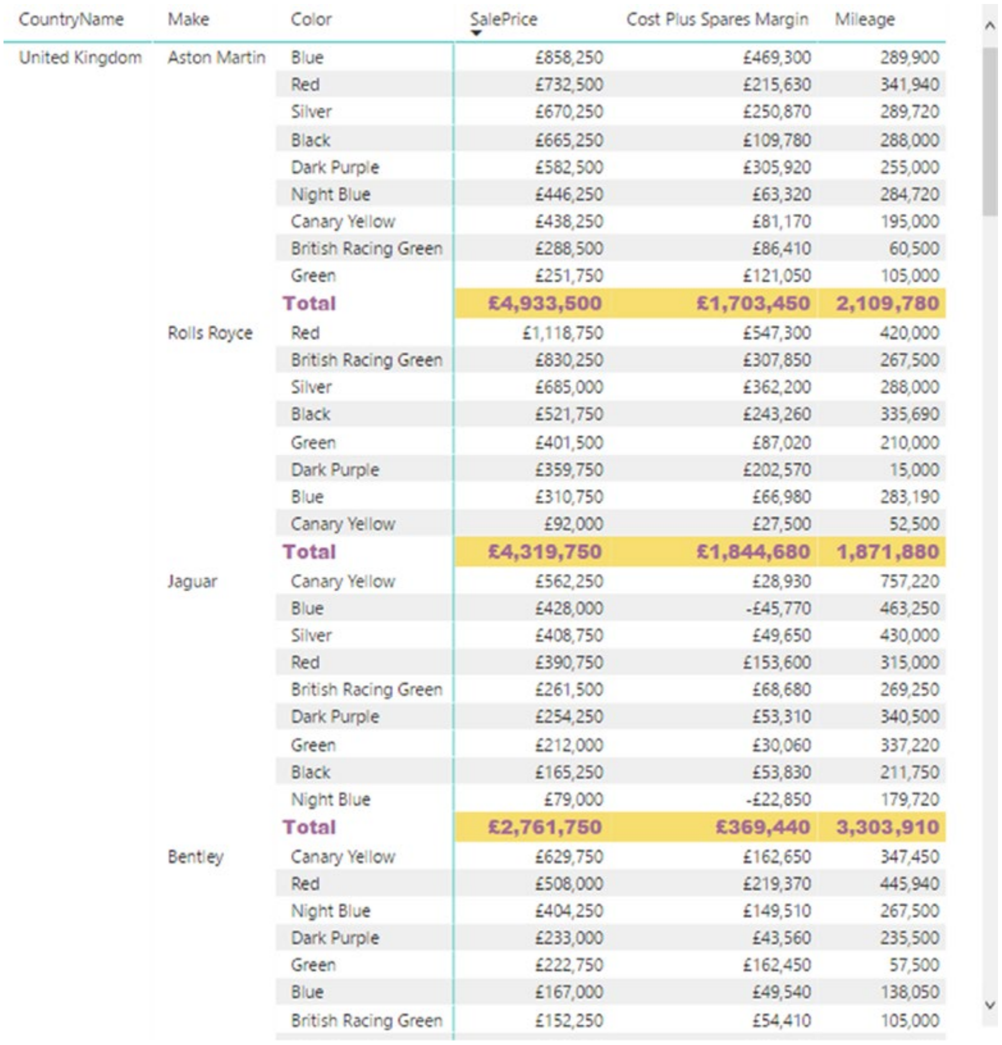
Revert to default

Figure 15-26. Selective subtotals

Sorting Data in Matrices

When you sort data in a matrix table, the sort order will respect the matrix hierarchy. This means that if you sort on the second element in a hierarchy (Make, in the example table we just created), then the primary element in the hierarchy (CountryName, the leftmost column) will switch back to the initial (alphabetical) sort order, as will any lower levels in the hierarchy of row elements.

If you sort by any value in a matrix, then the total for the highest level of the hierarchy is used to reorder the whole table. You can see this in Figure 15-27, where the matrix from the previous figure has been sorted on the total gross margin in descending order. This has made the country with the most sales move to the top of the table. As well, if you have a column matrix (as in this example), then you will end up sorting on the grand total of the columns (the two rightmost columns in this example) to make the matrix sort by numeric values, albeit in ascending order.



CountryName	Make	Color	SalePrice	Cost Plus Spares Margin	Mileage
United Kingdom	Aston Martin	Blue	£858,250	£469,300	289,900
		Red	£732,500	£215,630	341,940
		Silver	£670,250	£250,870	289,720
		Black	£665,250	£109,780	288,000
		Dark Purple	£582,500	£305,920	255,000
		Night Blue	£446,250	£63,320	284,720
		Canary Yellow	£438,250	£81,170	195,000
		British Racing Green	£288,500	£86,410	60,500
		Green	£251,750	£121,050	105,000
		Total	**£4,933,500**	**£1,703,450**	**2,109,780**
	Rolls Royce	Red	£1,118,750	£547,300	420,000
		British Racing Green	£830,250	£307,850	267,500
		Silver	£685,000	£362,200	288,000
		Black	£521,750	£243,260	335,690
		Green	£401,500	£87,020	210,000
		Dark Purple	£359,750	£202,570	15,000
		Blue	£310,750	£66,980	283,190
		Canary Yellow	£92,000	£27,500	52,500
		Total	**£4,319,750**	**£1,844,680**	**1,871,880**
	Jaguar	Canary Yellow	£562,250	£28,930	757,220
		Blue	£428,000	-£45,770	463,250
		Silver	£408,750	£49,650	430,000
		Red	£390,750	£153,600	315,000
		British Racing Green	£261,500	£68,680	269,250
		Dark Purple	£254,250	£53,310	340,500
		Green	£212,000	£30,060	337,220
		Black	£165,250	£53,830	211,750
		Night Blue	£79,000	-£22,850	179,720
		Total	**£2,761,750**	**£369,440**	**3,303,910**
	Bentley	Canary Yellow	£629,750	£162,650	347,450
		Red	£508,000	£219,370	445,940
		Night Blue	£404,250	£149,510	267,500
		Dark Purple	£233,000	£43,560	235,500
		Green	£222,750	£162,450	57,500
		Blue	£167,000	£49,540	138,050
		British Racing Green	£152,250	£54,410	105,000

Figure 15-27. *Sorting a matrix on values*

■ **Note** As was the case with tables, there are no specific options for formatting the figures in matrices. You format the data by clicking the Data View icon on the left of the Power BI Desktop window, select the column to format, and, in the Modeling ribbon, select a number format.

Cards

On some occasions, you will want to display a single figure more prominently than others. Perhaps you need to attract the viewer's attention to your new sales record, or you would like the boss to appreciate the customer satisfaction ratings that you have achieved.

Whatever the motivation, cards are the solution. Power BI Desktop cards are a simple and powerful way to isolate and emphasize a single figure. Suppose that you want to display all vehicle sales to date in a dashboard, for instance; the following explains how you can do it:

1. Using the C:\PowerBiDesktopSamples\CH15\CarSalesDataForReports.pbix file as your source of data, click outside any existing visualizations.
2. In the gallery of available visuals, click the Card icon. You can see this icon in Figure 15-28. An empty card will appear on the dashboard canvas.

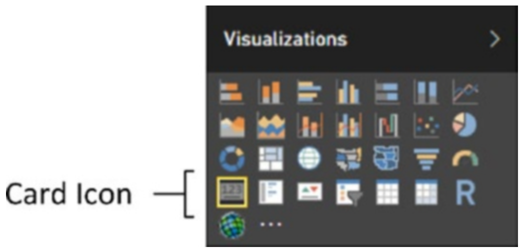


***Figure 15-28.** The Card icon in the gallery of available visuals*

3. Expand the InvoiceLines table fields in the Fields list and select the SalePrice field. The total for all sales will appear in the card.
4. Resize the card so that the figure fits inside the card borders without any wasted space. The final card will look like Figure 15-29.

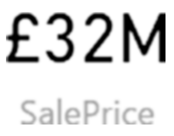


***Figure 15-29.** A card visual*

Formatting Cards

As you just saw, cards are an extremely effective way of focusing your audience's attention on key data. Yet there is a lot more that you can do to extend this emphasis, including the following:

- Change the units, number of decimals, color, and text size of the data that is displayed
- Remove the label that is displayed automatically, as well as alter its text size
- Add a title that you can format independently
- Change the background color of the card

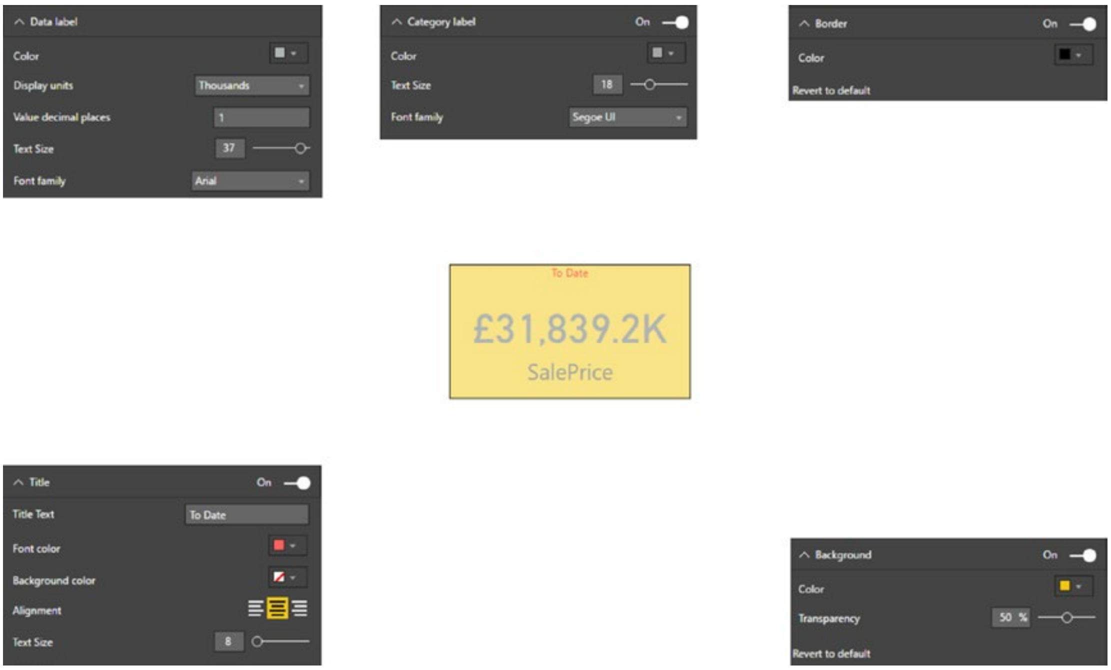
Let's see how to tweak the card that you just created:

1. Select the card. In this example, I use the card that you saw in Figure 15-25.
2. In the Visualizations pane, click the Format icon (the small paint roller beneath the palette of available visualization types). The formatting options for the selected card will be displayed.
3. Expand the Background section by clicking the downward-facing chevron to the left of the word Background.
4. Make sure that the switch to the right of the word Background is set to On (if it is not, then click to the right of the empty circle to change this).
5. Click the popup menu triangle to the right of Color and select a color from the palette of available colors.
6. Click the Transparency slider switch and slide it to the left to intensify the background color by reducing the percentage of transparency.
7. Expand the Category Label section by clicking the downward-facing chevron to the left of the words Category Label.
8. Adjust the text size by dragging the slider to the right. You can change the color as well if you want to.
9. Expand the Title section by clicking the downward-facing chevron to the left of the word Title.
10. Set the title to On by clicking immediately to the right of the On/Off button to the right of Title.
11. Enter **To Date** as the Title Text.
12. Click the popup menu triangle to the right of Font Color and select a color for the font from the palette of available colors.
13. Click the popup menu triangle to the right of Background Color and select a background color from the palette of available colors.
14. Click the middle icon of the Alignment icons to center the card title.
15. Expand the Data Label section by clicking the downward-facing chevron to the left.
16. Click the popup menu triangle to the right of Color and select a data label color from the palette of available colors.
17. Adjust the text size by dragging the Text Size slider to the right or left.

18. Select a display unit for the figure in the card by clicking the popup list of Display Units.

19. Select the number of decimals to be used for the card data by clicking the up and down triangles to the right of the Precision box. You can also enter the number of decimals directly, if you prefer.

20. Expand the Border section and switch the border on. Change the border color if you want to. The card (along with the formatting palette) will now look something like the elements that you can see in Figure 15-30.



Data label
Color
Display units: Thousands
Value decimal places: 1
Text Size: 37
Font family: Arial

Category label: On
Color
Text Size: 18
Font family: Segoe UI

Border: On
Color
Revert to default

To Date
£31,839.2K
SalePrice

Title: On
Title Text: To Date
Font color
Background color
Alignment
Text Size: 8

Background: On
Color
Transparency: 50 %
Revert to default

Figure 15-30. *A formatted card*

The available display units for formatting the value in a card are described in Table 15-2.

Table 15-2. *Display Units*

Display Unit	Description
Auto	Chooses an appropriate display unit from those available depending on the size of the figure.
None	Displays the actual figure.
Thousands	Displays the figure in thousands followed by a K.
Millions	Displays the figure in millions followed by an M.
Billions	Displays the figure in billions followed by a bn.
Trillions	Displays the figure in trillions followed by a T.

Multirow Cards

Tabular data can also be displayed in an extremely innovative way using the Power BI Desktop card style of output. As is the case with matrices, you begin by choosing the fields that you want to display as a basic table and then you convert this to another type of visual. Here is an example of how this can be done:

1. Using the C:\PowerBiDesktopSamples\CH15\CarSalesDataForReports.pbix file as your source of data, click outside any existing visuals.
2. In the Visualizations pane, click the Card icon. You can see this in Figure 15-31. An empty multirow card will appear on the dashboard canvas.

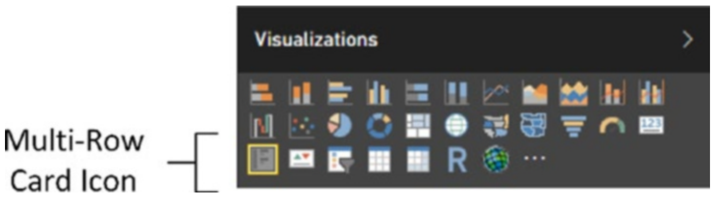
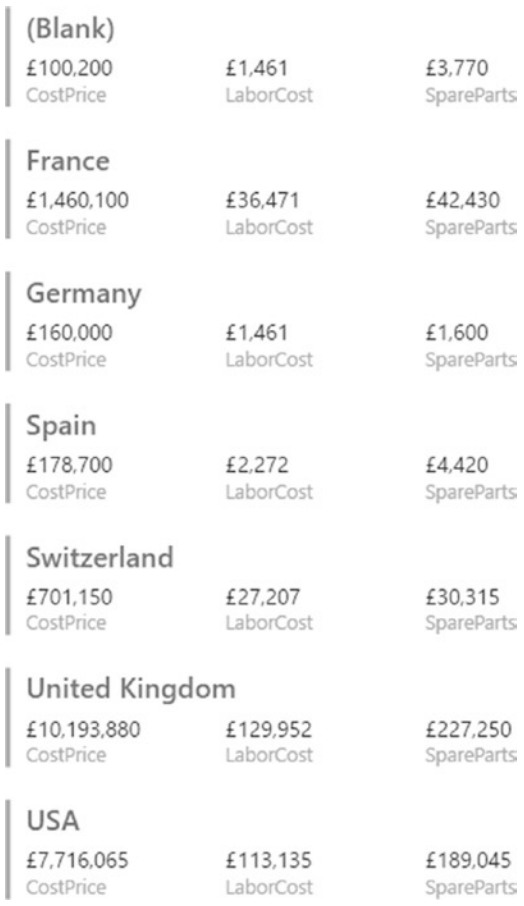


Figure 15-31. *The multirow card icon in the Visualizations pane*

3. Add the following fields, in this order:
   a. CountryName (from the Countries table)
   b. CostPrice (from the Stock table)
   c. LaborCost (from the Stock table)
   d. SpareParts (from the Stock table)

4. Resize the multirow card to display all the countries, with all of the figures on a single row per country. The visual will look like the one in Figure 15-32. Don't worry about the (Blank) costs; these correspond to vehicles in stock that are not yet sold, and so do not have a client country yet.



(Blank)

£100,200	£1,461	£3,770
CostPrice	LaborCost	SpareParts

France

£1,460,100	£36,471	£42,430
CostPrice	LaborCost	SpareParts

Germany

£160,000	£1,461	£1,600
CostPrice	LaborCost	SpareParts

Spain

£178,700	£2,272	£4,420
CostPrice	LaborCost	SpareParts

Switzerland

£701,150	£27,207	£30,315
CostPrice	LaborCost	SpareParts

United Kingdom

£10,193,880	£129,952	£227,250
CostPrice	LaborCost	SpareParts

USA

£7,716,065	£113,135	£189,045
CostPrice	LaborCost	SpareParts

Figure 15-32. *A card visualization*

Card-type tables will display the selected fields in the order in which they appear in the field well in the Visualizations pane, and it is here that they can be reordered, as with any table. This makes each card into a data record. The fields will flow left to right and then on to the following line in each card. What is interesting here is that adjusting the size of the table can change the appearance of the table quite radically. A very narrow table will list the fields vertically, one above the other. If you can fit all the fields onto a single row, then you will get a highly original multiple-record display.

Formatting Multirow Cards

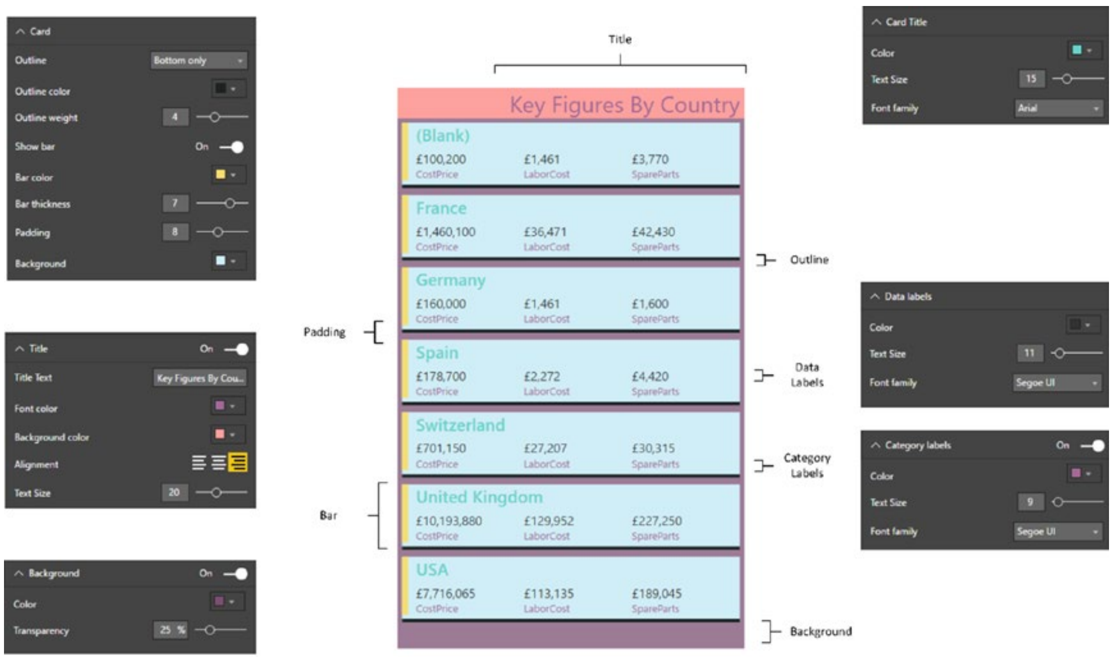
There are a few visual aspects of multirow cards that you can tweak for added effect. These include adding a title and changing the background color. Let's try both of these:

1. Select the multirow card visualization. In this example, I use the card that you saw in Figure 15-28.
2. In the Visualizations pane, click the Format icon (the small paint roller beneath the palette of available visual types). The formatting options for the selected multirow card are displayed.
3. Expand the Background section by clicking the downward-facing chevron to the left of the word Background.
4. Make sure that the switch to the right of the word Background is set to On (if it is not, then click to the right of the empty circle to change this).
5. Click the popup menu triangle to the right of Color and select a color from the palette of available tones.
6. Click the Transparency slider switch and slide it to the left to intensify the background color by reducing the percentage of transparency.
7. Expand the Title section by clicking the downward-facing chevron to the left of the word Title.
8. Set the title to On by clicking immediately to the right of the On/Off button to the right of Title.
9. Enter **Key Figures By Country** as the Title Text.
10. Click the popup menu triangle to the right of Font Color and select a color for the font from the palette of available colors.
11. Click the popup menu triangle to the right of Background Color and select a background color from the palette of available colors.
12. Slide the Text Size button to the right to set a larger font for the title.
13. Click the icon on the right for the Alignment. This will right align the title.
14. Click the popup menu triangle to the right of Card and select Bottom Only as the outline style.
15. Click inside the Outline Weight field and enter **4**.
16. Select a different bar color from the palette of available bar colors.
17. Slide the Bar Thickness button to the right to increase the width of the bar.
18. Slide the Padding button to the left to reduce the padding between individual elements in the card.
19. Choose a background color from the palette of colors available for the Background.
20. Expand the Data Labels section and slide the Text Size button to the right to increase the size of the font.

21. Expand the Card Title section and increase the text size by sliding the Text Size button to the right.

22. Choose a text color from the palette of available colors.

23. Expand the Category Labels section and select a text color from the palette of available colors. The multirow card visual will now look like the one in Figure 15-33. You might have to tweak its size to get the correct effect.



Card
Outline: Bottom only
Outline color
Outline weight: 4
Show bar: On
Bar color
Bar thickness: 7
Padding: 8
Background

Title: On
Title Text: Key Figures By Cou...
Font color
Background color
Alignment
Text Size: 20

Background: On
Color
Transparency: 25 %

Card Title
Color
Text Size: 15
Font family: Arial

Data labels
Color
Text Size: 11
Font family: Segoe UI

Category labels: On
Color
Text Size: 9
Font family: Segoe UI

Key Figures By Country

(Blank)		
£100,200 CostPrice	£1,461 LaborCost	£3,770 SpareParts
France		
£1,460,100 CostPrice	£36,471 LaborCost	£42,430 SpareParts
Germany		
£160,000 CostPrice	£1,461 LaborCost	£1,600 SpareParts
Spain		
£178,700 CostPrice	£2,272 LaborCost	£4,420 SpareParts
Switzerland		
£701,150 CostPrice	£27,207 LaborCost	£30,315 SpareParts
United Kingdom		
£10,193,880 CostPrice	£129,952 LaborCost	£227,250 SpareParts
USA		
£7,716,065 CostPrice	£113,135 LaborCost	£189,045 SpareParts

Figure 15-33. *A formatted multirow card*

■ **Note** You can also fix the aspect ratio of a multirow card visual just as you did for the initial table at the start of this chapter. Equally, you can add a border as you saw previously for a card visual.

Sorting Multirow Cards

Like tables and matrices, multirow cards can be sorted. However, there is a specific way to apply a sort order to this kind of visual:

1. Click the ellipses at the top right of the multirow card visual.

2. Select the column name to sort by from the list of columns in the popup menu.

The elements in the multirow card will now be ordered according to the values in the chosen column. You can see an example of the popup menu in Figure 15-34. This example uses the multirow card that you created previously.

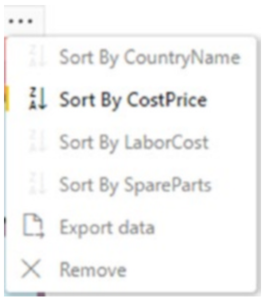


Figure 15-34. *Sorting a multirow card*

■ **Note** Sorting on the category in a multirow card (the country in this example) will initially sort in reverse alphabetical order (as a multirow card is initially ordered in alphabetical order). Re-selecting a sort on the category will reset the sort order to alphabetical. Sorting on a value will always order from highest to lowest—and re-clicking the value in the popup menu will remove the sort order and revert to the initial order.

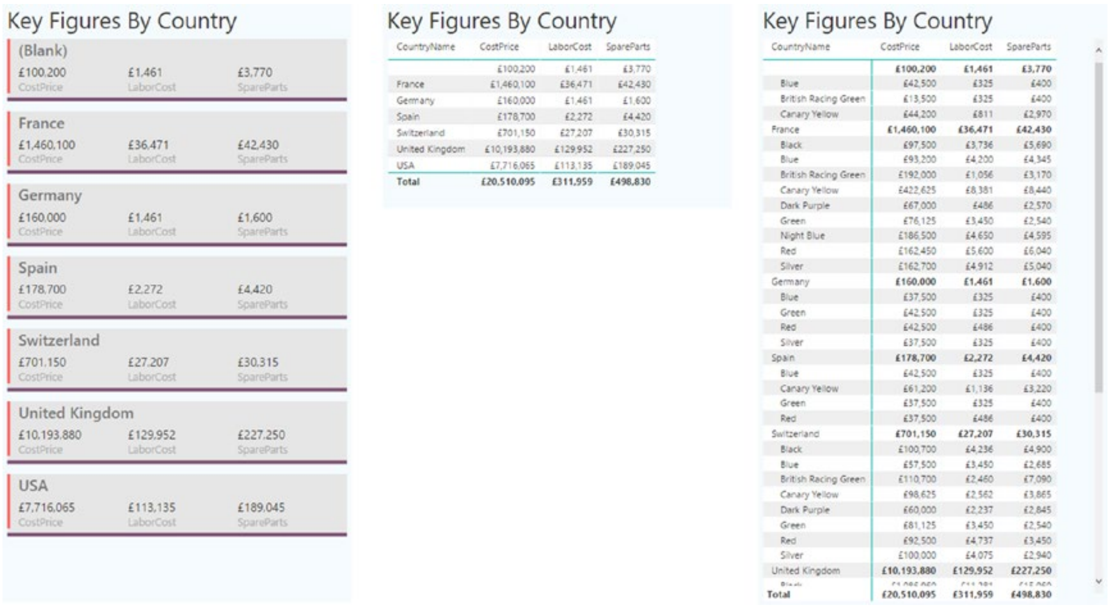
Switching Between Table Types

One of the fabulous things about Power BI Desktop is that it is designed from the ground up to let you test ideas and experiment with ways of displaying your data quickly and easily. So, quite naturally, you can switch table types easily to see which style of presentation is best suited to your ideas and the message that you want to convey. To switch table types, all you have to do is select the current text-based visualization (table, multirow card, or matrix) and select one of these options from the gallery of available visuals:

- Table
- Matrix
- Multirow card

What is even more reassuring is that Power BI Desktop remembers the attributes of the previous table type you used. So, for instance, if you set up a matrix with a carefully crafted hierarchy and then switch to a card-type table, Power BI Desktop remembers how you set up the matrix should you want to switch back to it.

To see an example of how this works in practice, take a look at Figure 15-35. This shows the same initial visual that has been copied and then switched between the text-based core types of visual. I have added the Color field to the matrix and expanded the next level to make the difference between the table and matrix clearer.



Key Figures By Country

(Blank)
£100,200 CostPrice | £1,461 LaborCost | £3,770 SpareParts

France
£1,460,100 CostPrice | £36,471 LaborCost | £42,430 SpareParts

Germany
£160,000 CostPrice | £1,461 LaborCost | £1,600 SpareParts

Spain
£178,700 CostPrice | £2,272 LaborCost | £4,420 SpareParts

Switzerland
£701,150 CostPrice | £27,207 LaborCost | £30,315 SpareParts

United Kingdom
£10,193,880 CostPrice | £129,952 LaborCost | £227,250 SpareParts

USA
£7,716,065 CostPrice | £113,135 LaborCost | £189,045 SpareParts

Key Figures By Country

CountryName	CostPrice	LaborCost	SpareParts
	£100,200	£1,461	£3,770
France	£1,460,100	£36,471	£42,430
Germany	£160,000	£1,461	£1,600
Spain	£178,700	£2,272	£4,420
Switzerland	£701,150	£27,207	£30,315
United Kingdom	£10,193,880	£129,952	£227,250
USA	£7,716,065	£113,135	£189,045
Total	**£20,510,095**	**£311,959**	**£498,830**

Key Figures By Country

CountryName	CostPrice	LaborCost	SpareParts
	£100,200	**£1,461**	**£3,770**
Blue	£42,500	£325	£400
British Racing Green	£13,500	£325	£400
Canary Yellow	£44,200	£811	£2,970
France	**£1,460,100**	**£36,471**	**£42,430**
Black	£97,500	£3,736	£5,690
Blue	£93,200	£4,200	£4,345
British Racing Green	£192,000	£1,056	£3,170
Canary Yellow	£422,625	£8,381	£8,440
Dark Purple	£67,000	£486	£2,570
Green	£76,125	£3,450	£2,540
Night Blue	£186,500	£4,650	£4,595
Red	£162,450	£5,600	£6,040
Silver	£162,700	£4,912	£5,040
Germany	**£160,000**	**£1,461**	**£1,600**
Blue	£37,500	£325	£400
Green	£42,500	£325	£400
Red	£42,500	£486	£400
Silver	£37,500	£325	£400
Spain	**£178,700**	**£2,272**	**£4,420**
Blue	£42,500	£325	£400
Canary Yellow	£61,200	£1,136	£3,220
Green	£37,500	£325	£400
Red	£37,500	£486	£400
Switzerland	**£701,150**	**£27,207**	**£30,315**
Black	£100,700	£4,236	£4,900
Blue	£57,500	£3,450	£2,685
British Racing Green	£110,700	£2,460	£7,090
Canary Yellow	£98,625	£2,562	£3,865
Dark Purple	£60,000	£2,237	£2,845
Green	£81,125	£3,450	£2,540
Red	£92,500	£4,737	£3,450
Silver	£100,000	£4,075	£2,940
United Kingdom	**£10,193,880**	**£129,952**	**£227,250**
Total	**£20,510,095**	**£311,959**	**£498,830**

Figure 15-35. *Switching visuals using the same data and formatting*

The following are the main things to note here:

- The visual remains the same size when you switch types. Consequently, you will probably have to resize it.
- All formatting that can be retained is retained. This can be seen in the titles and background of all three visuals in Figure 15-35.

Conclusion

This chapter extended the knowledge that you acquired in the previous chapter to show you how to create matrices. You then learned how to drill down into any matrix to display a hierarchy of information—or only a sublevel in the data hierarchy.

You then learned how to create card visuals and Power BI Desktop's unique and effective multirow cards.

Now that you have mastered the basics of creating visuals, it is time to move on to the next level of dashboard creation. In the next two chapters you will discover how to create and enhance charts. This will allow you to give freer rein to your imagination and help you let your data speak for itself.

CHAPTER 16

Charts in Power BI Desktop

It is one thing to have a game-changing insight that can fundamentally alter the way your business works. It is quite another to be able to convince your colleagues of your vision. So what better way to show them—intuitively and instantaneously—that you are right than with a chart that irrefutably makes your point?

Power BI Desktop is predicated on the concept that a picture is worth many thousands of words. Its charting tools let you create clear and convincing visuals that tell your audience far more than a profusion of figures ever could. This chapter, therefore, will show you how simple it can be not just to make your data explain your analysis, but to make it seem to leap off the screen. You will see over the next few pages how a powerful chart can persuade your peers and bosses that your ideas and insights are the ones to follow.

A little more prosaically, Power BI Desktop lets you make a suitable dataset into

- Pie charts
- Bar charts
- Column charts
- Line charts
- Area charts
- Scatter charts
- Bubble charts
- Funnel charts
- Waterfall charts
- Donut charts
- Ribbon charts

In this chapter, we get up and running by looking at all these types of charts, and then extend some of them to create stacked bar, stacked column, and stacked area charts, 100% stacked bar and stacked column charts, as well as dual-axis charts. Once you have decided upon the most appropriate chart type, you can then enhance your visualization with a title, data labels, and legends, where they are appropriate. Finally, you can format all of these elements to give your charts the wow factor that they deserve.

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_16

A First Chart

It is generally easier to appreciate the simplicity and power of Power BI Desktop by doing rather than talking. So I suggest leaping straight into creating a first chart. In this section, we will look only at "starter" charts that all share a common thread—they are based on a single column of data values and a single column of descriptive elements. The data will include

- A list of clients
- Car sales for a given year

So, let's get charting! In this chapter, you will use the C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix Power BI Desktop file that is available on the Apress web site for download.

Creating a First Chart

Any Power BI Desktop chart begins as a dataset. So, let me introduce you to the world of charts by showing you how to make a bar chart in a few clicks; the following explains how to begin:

1. Open the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix from the downloadable samples.
2. Click the Stacked Bar Chart icon at the top left of the Visualizations pane. This icon is illustrated in Figure 16-1. An empty bar chart visual will appear on the dashboard canvas.

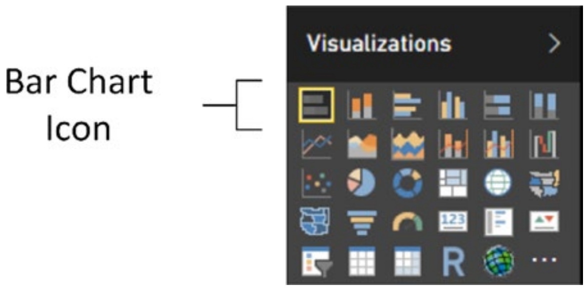


Figure 16-1. *The Bar Chart icon in the Visualizations pane*

3. Leave the empty bar chart visual selected and expand the Clients table in the Fields list.
4. Select the box to the left of the ClientName field.
5. Leaving this new bar chart visual selected, expand the InvoiceLines table In the Fields list.
6. Click the check box to the left of the SalePrice field. This will add the cumulative sales per client to the chart.
7. Resize the chart (I suggest widening it and increasing the height) until the axis labels are clearly visible, as shown in Figure 16-2.

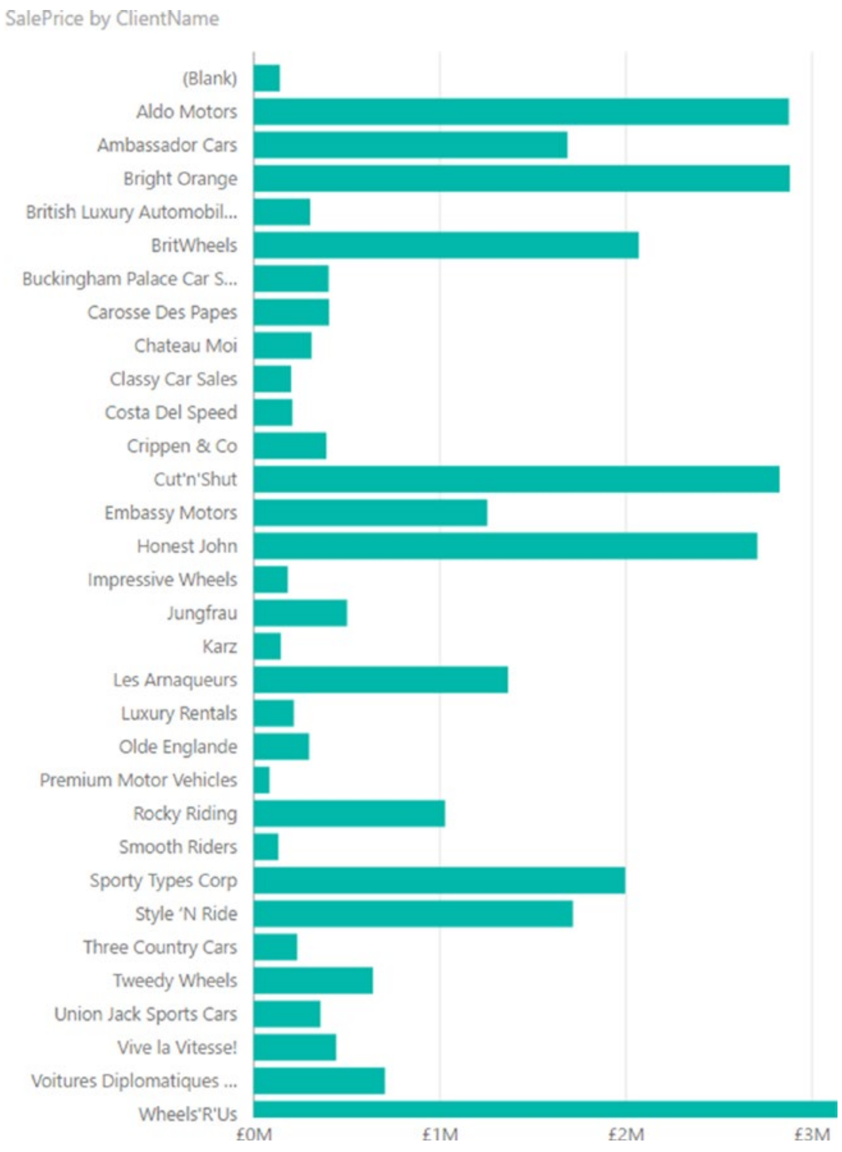


Figure 16-2. *A bar chart after resizing*

And that is all that there is to creating a simple starter chart. This process might only take a few seconds, and once it is complete, it is ready to show to your audience, or be remodeled to suit your requirements.

Nonetheless, a few comments are necessary to clarify the basics of chart creation in Power BI Desktop:

- When creating the chart, you can use any of the techniques described in Chapters 14 and 15 to add fields to visuals. You can drag fields into the Fields Well of the Visualizations pane or onto the visual directly if you prefer.
- When using only a single dataset, you can choose either clustered or stacked as the chart type for a bar or column chart; the result is the same in either case. As you will see as we progress, this will not be the case when the chart is based on multiple data fields.
- Power BI Desktop will add a title at the top left of the chart explaining what data the chart is based on. You can see an example of this in Figure 16-2.
- Creating a chart is very much a first step. You can do so much more to enhance a chart and accentuate the insights that it can bring. All of this follows further on in this chapter and in the next one.

Converting a Table into a Chart

Another way to create a chart is to create a table first (by dragging the data fields onto the Power BI desktop canvas without first clicking a chart icon, for instance). Then select the table and click a chart icon in the Visualizations pane to convert the table into a chart.

However, if you create a table first, then transform a table into a chart (by selecting a table and clicking the Stacked Bar chart icon), the field well of the Visualizations pane changes to reflect the options available when creating or modifying a chart. If you select the chart that you just created, you will see that the ClientName field has been placed in the Axis box, and the SalePrice field has been placed in the Value box. Neither of these boxes exists if the visual is a table. This can be seen in Figure 16-3.

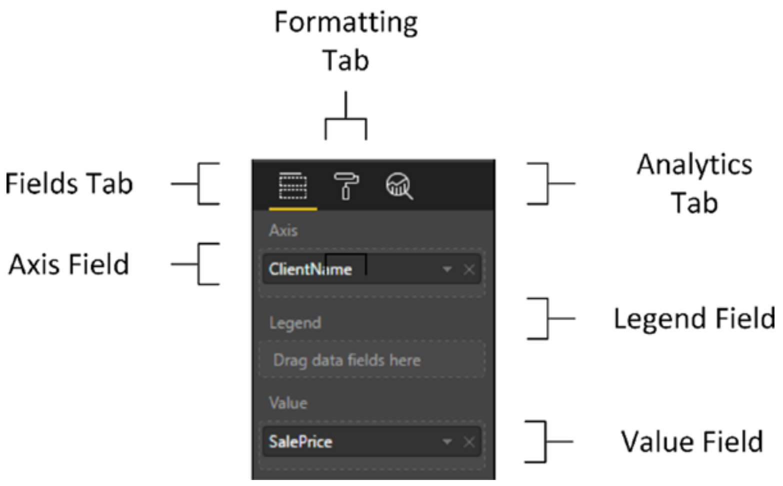


***Figure 16-3.** The Fields list for a bar chart*

Deleting a Chart

Deleting a chart is as simple as deleting a table, a matrix, or a card. All that you have to do is

1. Click inside the chart.
2. Press the Delete key.

If you remove all the fields from the Layout section of the field well of the Visualizations pane (with the chart selected), then you will also delete the chart.

Basic Chart Modification

So you have an initial chart. Suppose, however, that you want to change the fields on which the chart is based. Well, all you have to do to change both the axis elements (the client names and the values represented) is

1. Click the bar chart that you created previously. Avoid clicking any of the bars in the chart for the moment.
2. In the field well of the Visualizations pane, click the small cross at the right of SalePrice in the Values well (or click the popup menu for SalePrice, and select Remove Field). The bars will disappear from the chart.
3. Drag the field LaborCost from the Stock table in the Fields list into the Values well.
4. In the field well of the Visualizations pane, click the popup menu for ClientName in the Axis box, and select Remove Field. The client names will disappear from the chart and a single bar will appear.
5. In the Fields list, expand the Colors table and drag the Color field from the Colors table into the Axis box. The list of colors will replace the list of clients on the axis, and a series of bars will replace the single bar. Look at Figure 16-4 to see the difference.

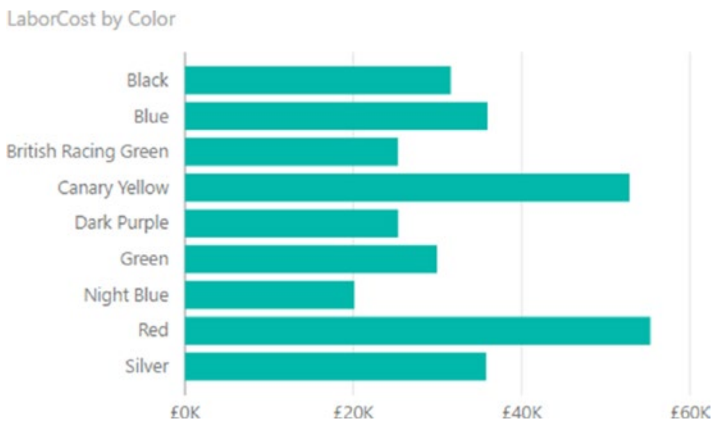


Figure 16-4. *A simple bar chart with the corresponding Layout section*

That is it. You have changed the chart completely without rebuilding it. Power BI Desktop has updated the data in the chart and the chart title to reflect your changes.

Basic Chart Types

When dealing with a single set of values, you will probably be using the following six core chart types to represent data:

- Bar chart
- Column chart
- Line chart
- Pie chart
- Donut chart
- Funnel chart

Let's see how we can try out these types of chart using the current dataset—the colors and Gross Margin that you applied in the previous chapter.

Column Charts

A column chart is, to all intents and purposes, a bar chart where the bars are vertical rather than horizontal. So, do the following to switch your bar chart to a column chart:

1. Click the bar chart that you previously created and modified (the one shown in Figure 16-4). Avoid clicking any of the bars in the chart.
2. In the Visualizations pane, click the Stacked Column Chart icon.
3. Resize the chart as required. Your chart should look like Figure 16-5.

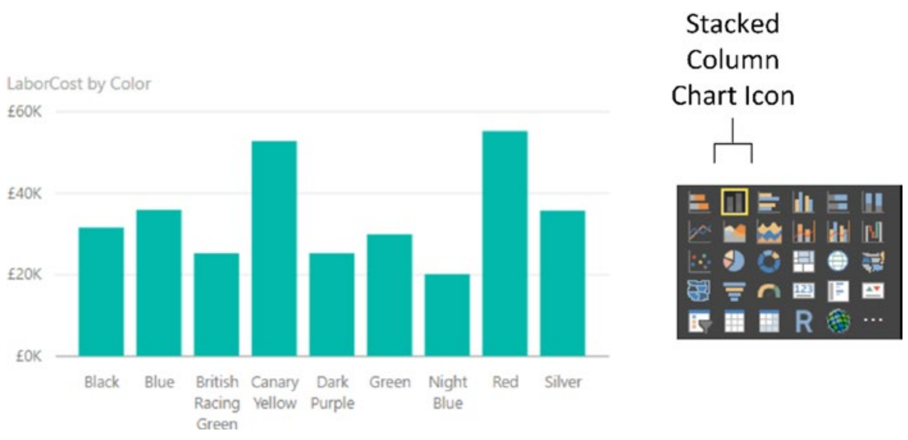


Figure 16-5. *An elementary column chart*

Line Charts

A line chart displays the data as a set of points joined by a line. Do the following to switch your column chart to a line chart:

1. Click the column chart that you created previously. Avoid clicking any of the columns in the chart for the moment.
2. In the Visualizations pane, click the Line Chart icon. Your chart should look like Figure 16-6.

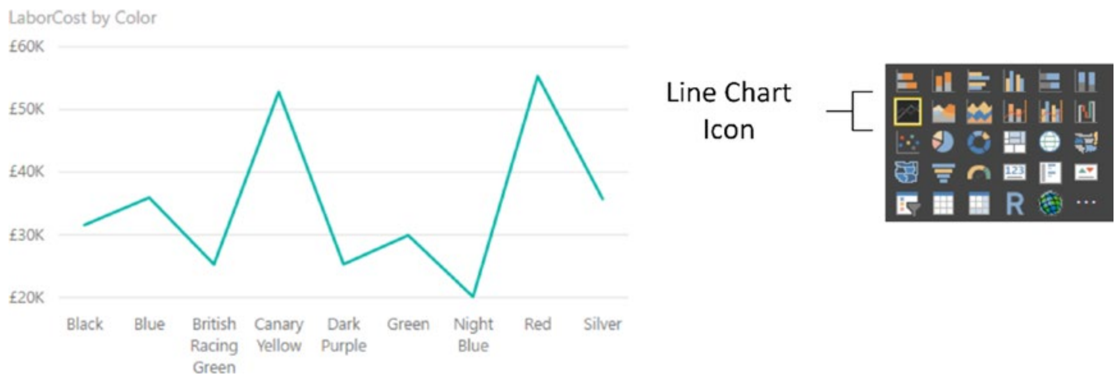


Figure 16-6. *A simple line chart*

Pie Charts

Pie charts can be superb at displaying a limited set of data for a single series, like we have in this example. To switch the visual to a pie chart:

1. Click the line chart that you created previously. Avoid clicking any of the lines in the chart for the moment.
2. In the Visualizations pane, click the Pie Chart icon. The line chart will become a pie chart.
3. Resize the pie chart, if necessary, to display the text for all the colors correctly. Your chart should look like Figure 16-7. You will notice that the Layout section has changed slightly for a pie chart, and the Axis box has been replaced by a Legend box.

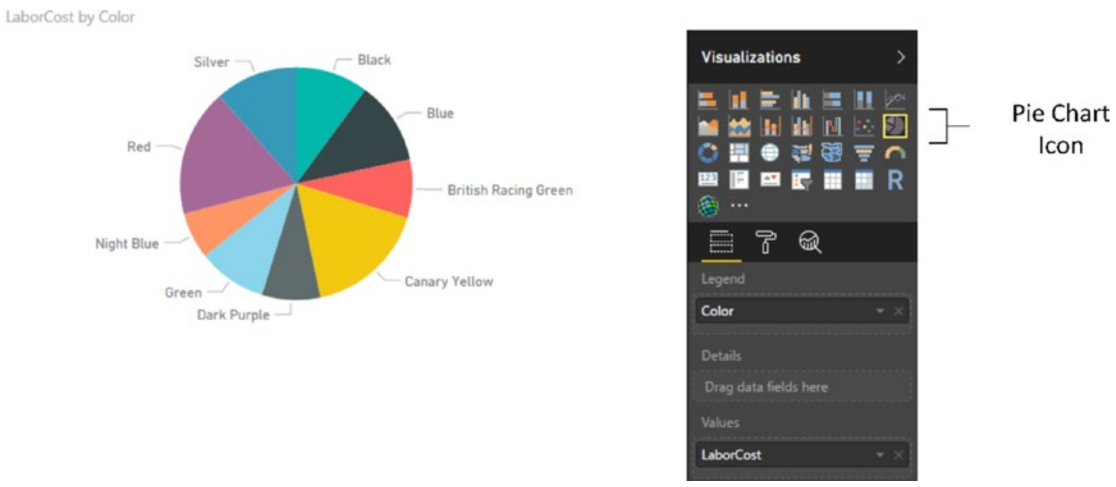


Figure 16-7. *A basic pie chart*

A pie chart is distorted if it includes negative values at the same time as it contains positive values. Power BI Desktop will not display the negative values. If your dataset contains a mix of positive and negative data, then Power BI Desktop displays an alert above the chart warning "Too many values. Not showing all data." What this nearly always really means is that the pie chart contains positive *and* negative values, and that the negative values are not displayed. This also applies to donut charts. Negative values can make funnel charts appear a little peculiar, too. If you want to see this for yourself, try creating a table of makes and gross margin. You will see that the table contains seven rows, but the corresponding pie or donut chart only displays six sections. This is because the make with a negative gross margin (TVR) does not appear in the pie chart.

In practice, you may prefer not to use pie or donut charts when your data contains negative values, or you may want to separate out the positive and negative values into two datasets and display two charts, using filters (this is explained in Chapter 20).

■ **Note** Juggling chart size and font size to fit in all the elements and axis and/or legend labels can be tricky. One useful trick is to prepare "abbreviated" data fields in the source data, as has been done in the case of the QuarterAbbr field in the Date table that contains Q1, Q2, and so on, rather than Quarter 1, Quarter 2, and so on, to save space in the chart. Techniques for this sort of data preparation are given in Chapters 11 and 13.

Essential Chart Adjustments

I hope you will agree that creating a chart in Power BI Desktop is extremely simple. Yet the process of producing a telling visualization does not stop when you take a dataset and display it as a chart. At the very least, you want to make the following tweaks to your new chart:

- Resize the chart
- Reposition the chart
- Sort the elements in the chart
- Alter the size of the fonts in the chart

None of these tasks is at all difficult. Indeed, it can take only a few seconds to transform your initial chart into a compelling visual argument—when you know the techniques to apply.

Resizing Charts

A chart is like any other visual on the Power BI Desktop dashboard canvas and can be resized to suit your requirements. The following explains how to resize a chart:

1. Click inside the chart (but not on any of the bars, columns, lines, or pie segments).
2. Place the mouse pointer over any of the eight handles that appear at the corners and in the middle of the edges of the chart that you wish to adjust. The pointer becomes a two-headed arrow.
3. Drag the mouse pointer to resize the chart.

■ **Note** Remember that the lateral handles let you resize the chart only horizontally or vertically, and that the corner handles allow you to resize both horizontally and vertically.

Resizing a chart can have a dramatic effect on the text that appears on an axis. Power BI Desktop always tries to keep the space available for the text on an axis proportionate to the size of the whole chart.

For column, line, and bar charts, this can mean that the text can be truncated, with an ellipsis (three dots) indicating that not all the text is visible. With column and line charts, the text may even be angled at 45 degrees or possibly swiveled to appear vertically.

If you reduce the height (for a bar chart) or the width (for a column or a line chart) below a certain threshold, Power BI Desktop will stop trying to show all the elements on the non-numeric axis. Instead, it only shows a few elements and adds a scroll bar to allow you to scroll through the remaining data. What is more, if axis labels cannot be displayed in their entirety, they will be truncated (and ellipses added). You can see an example of these outcomes for a bar chart in Figure 16-8.

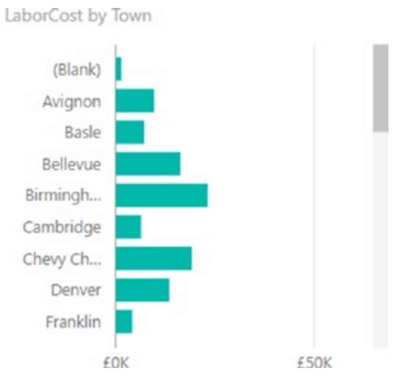


Figure 16-8. *A chart with a scroll bar visible*

All that this means is you might have to tweak the size and height-to-width ratio of your chart until you get the best result. If you are in a hurry to get this right, I advise using the handle in the bottom-right corner to resize a chart, as dragging this up, down, left, and right will quickly show you the available display options.

Repositioning Charts

You can move a chart anywhere inside the Power BI Desktop report, as follows:

1. Place the mouse pointer over the border of the chart. The pointer changes into a hand icon.
2. Drag the mouse pointer.

Sorting Chart Elements

Sometimes you can really make a point about data by changing the order in which you have it appear in a chart. Up until this point you have probably noticed that when you create a chart, the elements on the axis (and this is true for a bar chart, column chart, line chart, or pie chart) are in alphabetical order by default. If you want to confirm this, then just look at Figures 16-5 to 16-8.

Suppose now, for instance, you want to show the way that the gross margin is affected by the color of the vehicle. In this case, you want to sort the data in a chart from highest to lowest so that you can see the way in which the figures fall, or rise, in a clear order. The following steps explain how to do this:

1. Select the bar chart type for Color and LaborCost, as described earlier (and shown in Figure 16-4).
2. Place the mouse pointer over the chart. You will see that the chart border and options button (the ellipses at the top right) appear.
3. Click the ellipses to display the chart menu and then click the chevron to the right of Sort By Color (or the appropriate field to order the data). This is shown in Figure 16-9, where the columns are sorted in reverse alphabetical order.

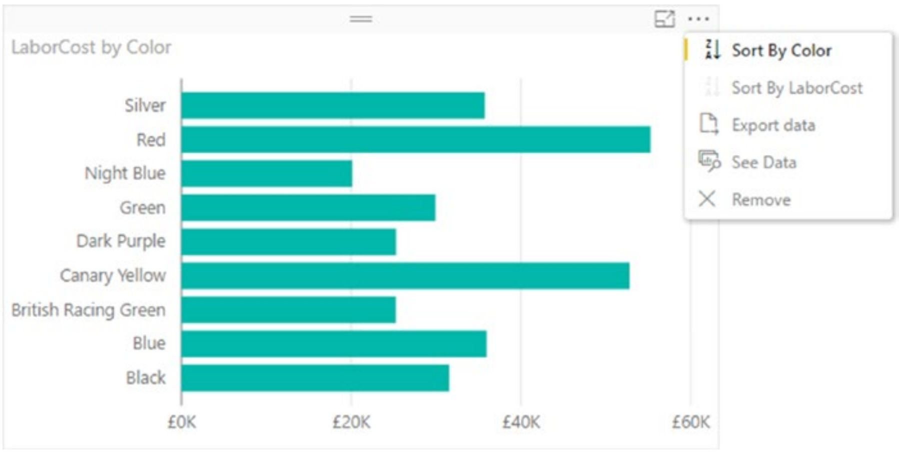


***Figure 16-9.** Using the popup menu in a chart to sort data*

4. In the popup menu, click the option Sort By LaborCost. This will change the sort order of the elements in the chart. You will now see a chart looking like the one in Figure 16-10.

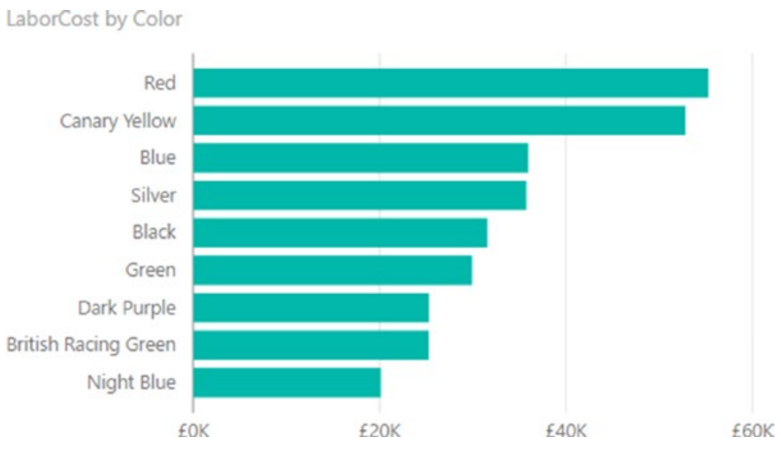


Figure 16-10. *Sorting data in a bar chart*

As you can see, the initial sort is in descending order when you sort by a numeric value. Let's suppose now that you want to see the sales by color in *ascending* order. All you have to do is repeat the operation that you just carried out and the chart will be resorted. Only this time, it is sorted in ascending order. Equally, the *first* time that you sort on a *text* element the chart is sorted in *ascending* order and the *second* time it is sorted in *descending* order.

I should add just a short remark about sorting pie charts. When you sort a pie chart by a numeric field, the pie chart is sorted clockwise, starting at the top of the chart. So if you are sorting colors by LaborCost in descending order, the top selling color is at the top of the pie chart (at 12 o'clock), with the second bestselling color to its immediate right (2 o'clock, for example), and so on. An example of this is shown in Figure 16-11.

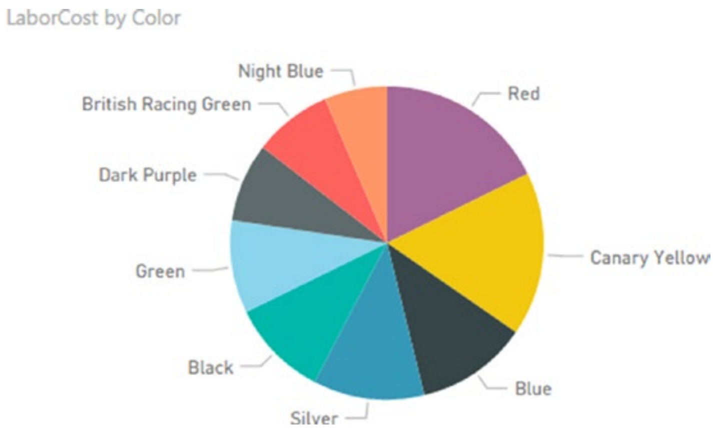


Figure 16-11. *Sorting data in a pie chart*

Donut Charts

Donut charts are essentially a variation of a pie chart. However, they can make a welcome presentational change from their overexposed older sibling. Fortunately, they are equally easy to create. In this example, you are going to see how to visualize the parts cost incurred for each make of vehicle purchased. To vary the approach, in this example you will first create a table and then convert it to a donut chart.

1. Continue using (or open) the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix.
2. Ensure that no current visual is selected.
3. Expand the Stock table in the Fields list.
4. Select the box to the left of the Make field. A table containing the list of makes purchased will appear in the dashboard canvas.
5. Leave this new table selected and click the check box to the left of the SpareParts field. This will add the cumulative cost of spare parts per make to the table.
6. Click the Donut Chart icon in the Visualizations pane. This will convert the table to a donut chart.
7. Resize the chart if necessary to show the names of the makes clearly, as shown in Figure 16-12.

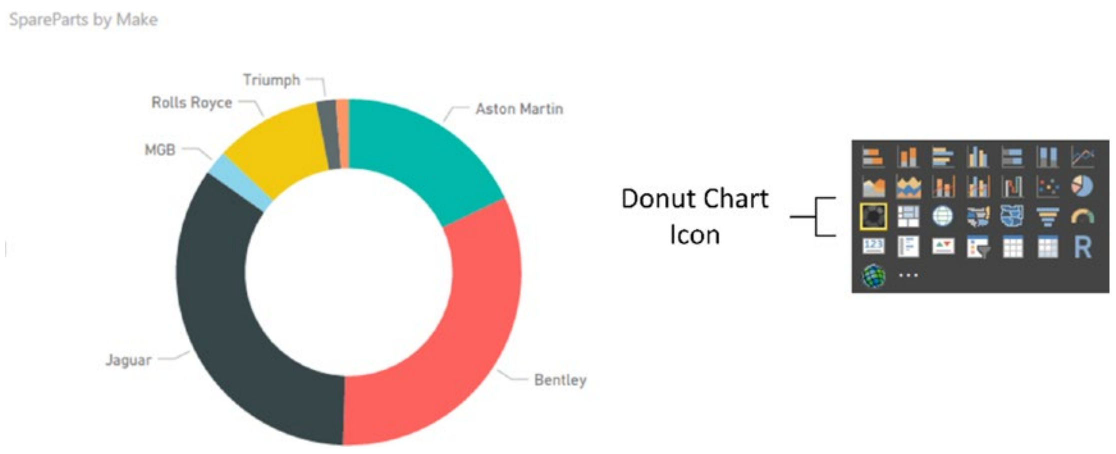


Figure 16-12. *A donut chart*

Funnel Charts

Funnel charts are excellent when it comes to comparing the relative values of a single series of figures. As you are now well-versed in the art of creating charts with Power BI Desktop, designing a funnel chart that displays parts cost per color should not be a problem for you.

1. Carry on using the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix.
2. Expand the Colors table in the Fields list.

3. Drag the Color field onto an empty part of the dashboard canvas. A table containing the list of vehicle colors will appear.

4. Leave this new table selected and click the check box to the left of the SpareParts field in the Stock table. This will add the cumulative cost of parts per make to the table.

5. Click the Funnel Chart icon in the Visualizations pane. This will convert the table to a funnel chart.

6. Place the mouse pointer over the chart. You will see that the chart border and options button (the ellipses at the top right) appear.

7. Click the ellipses to display the chart menu and then click Sort By SpareParts. The funnel chart will display the color with the highest value for the cost of spare parts at the top of the visual.

8. Resize the chart if necessary to show the names of the colors clearly as well as making the relative weights for all the colors more clearly comprehensible, as shown in Figure 16-13.

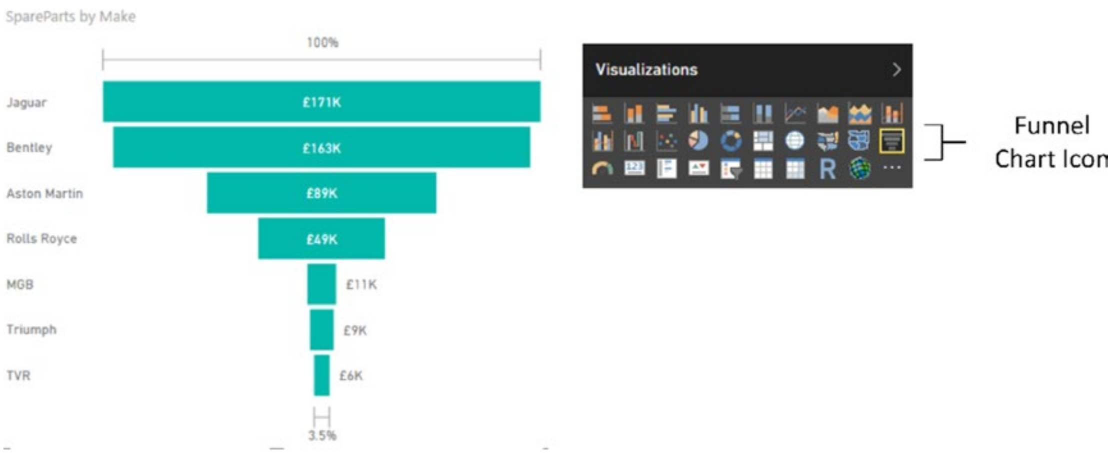


Figure 16-13. *A funnel chart*

Multiple Data Values in Charts

So far in this chapter, we have seen simple charts that display a single value. Life is, unfortunately, rarely that simple, and so it is time to move on to slightly more complex, but possibly more realistic, scenarios where you need to compare and contrast multiple data elements.

For this set of examples, I will presume that we need to take an in-depth look at the following indirect cost elements of our car sales to date:

- Parts
- Labor

Consequently, to begin with a fairly simple comparison of these indirect costs, let's start with a clustered column chart:

1. Open the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix from the downloadable samples.
2. Starting with a clean Power BI Desktop report, create a table that displays the following fields:
   a. ClientName (from the Clients table)
   b. SpareParts (from the Stock table)
   c. LaborCost (from the Stock table)
3. Leaving the table selected, click the Clustered Column icon in the Visualization pane.
4. Resize the chart to make it clear and comprehensible, as shown in Figure 16-14. (I have included the fields from the Visualizations pane so that you can see this, too.)

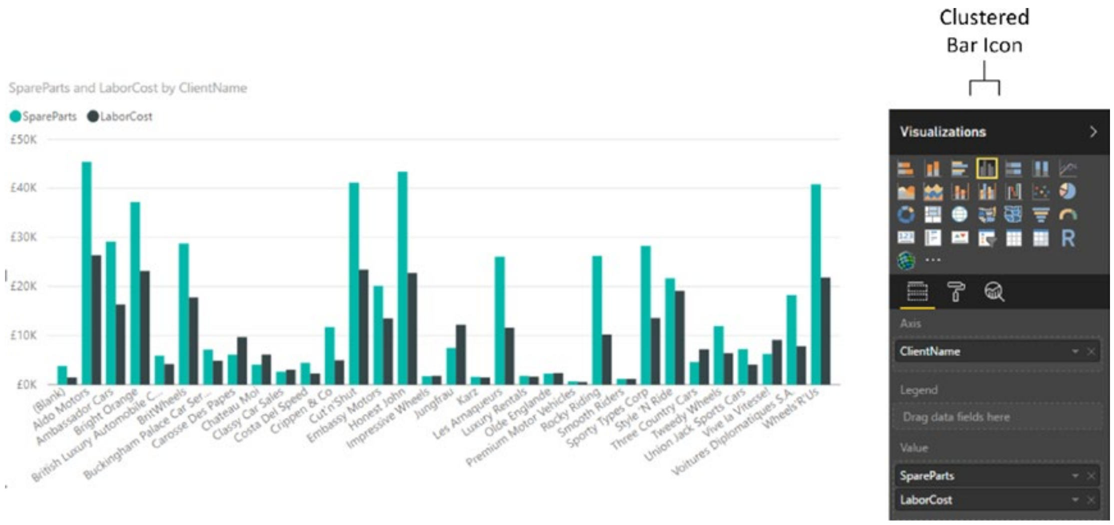


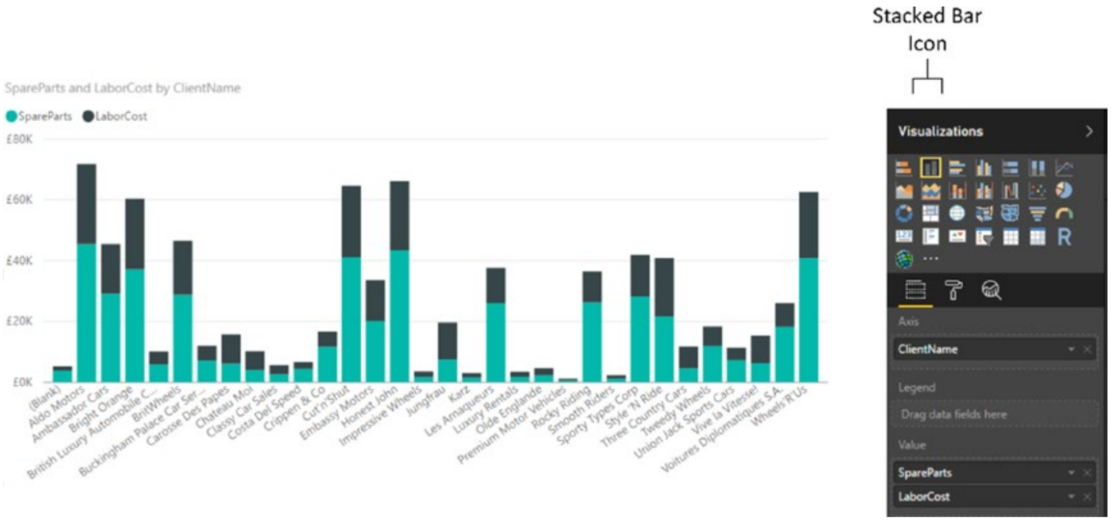
Figure 16-14. *Multiple data values in charts—a clustered column chart with the Layout section shown*

You will notice that a chart with multiple datasets has a legend by default, and that the automatic chart title now says SpareParts and LaborCost By ClientName.

The same dataset can be used as a basis for other charts that can effectively display multiple data values:

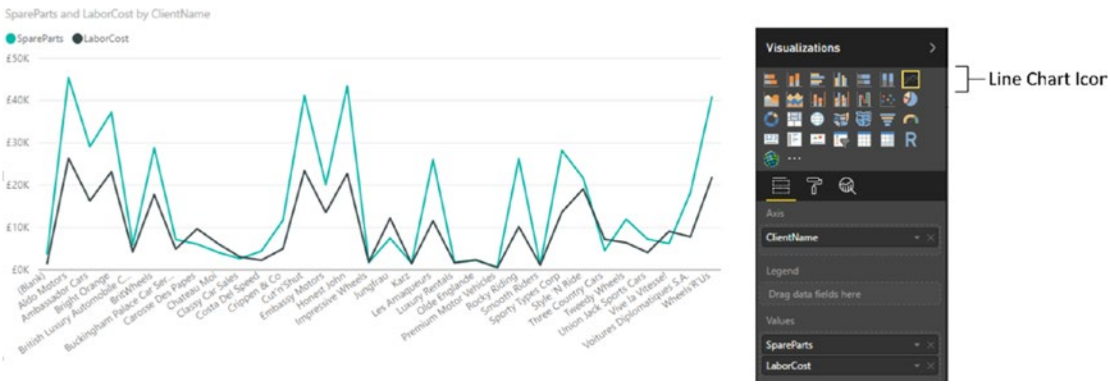
- Stacked charts
- Clustered column and stacked column
- Line charts
- Area charts
- Stacked area charts

Since column charts are essentially bar charts pivoted 90 degrees, I will not show examples. However, in Figures 16-15, 16-16, 16-17, and 16-18, you will see examples of a stacked column chart, a line chart, an area chart, and a stacked area chart-all created from the same data. You also see that when creating these types of visualization, the Layout section of the field well of the Visualizations pane remains the same for all of these charts.



Axis: ClientName
Legend
Value: SparePart, LaborCost

Figure 16-15. A simple stacked bar chart



Axis: ClientName
Legend
Values: SpareParts, LaborCost

Figure 16-16. A line chart that displays multiple values

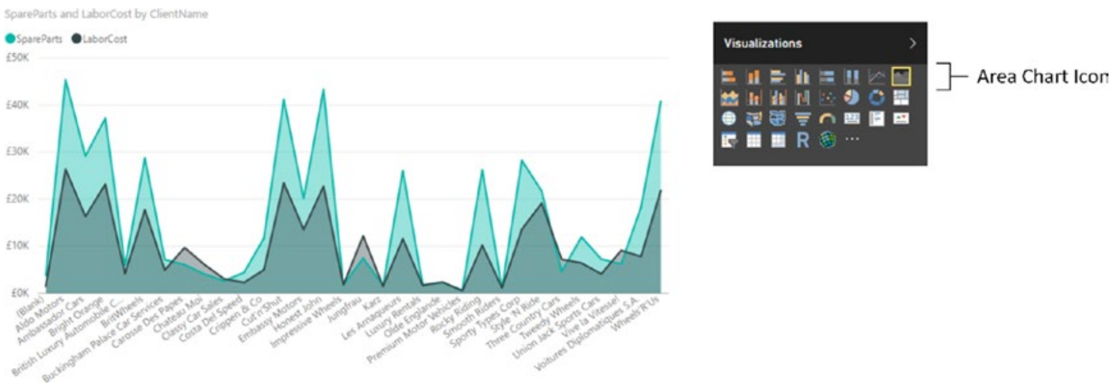


Figure 16-17. *An area chart displaying multiple values*

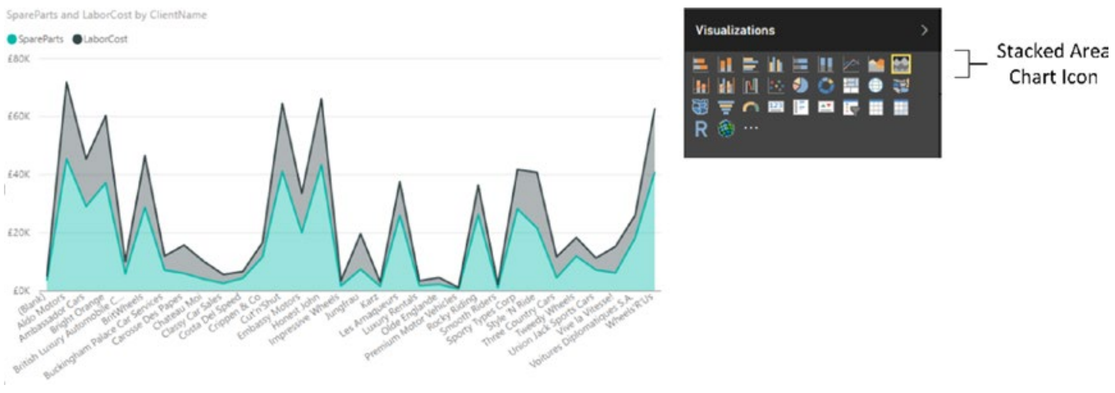


Figure 16-18. *A stacked area chart based on multiple values*

These four charts all display the same data in different ways. Knowing this, you can choose the type of chart that best conveys the information and draws your reader's attention to the point that you are trying to make.

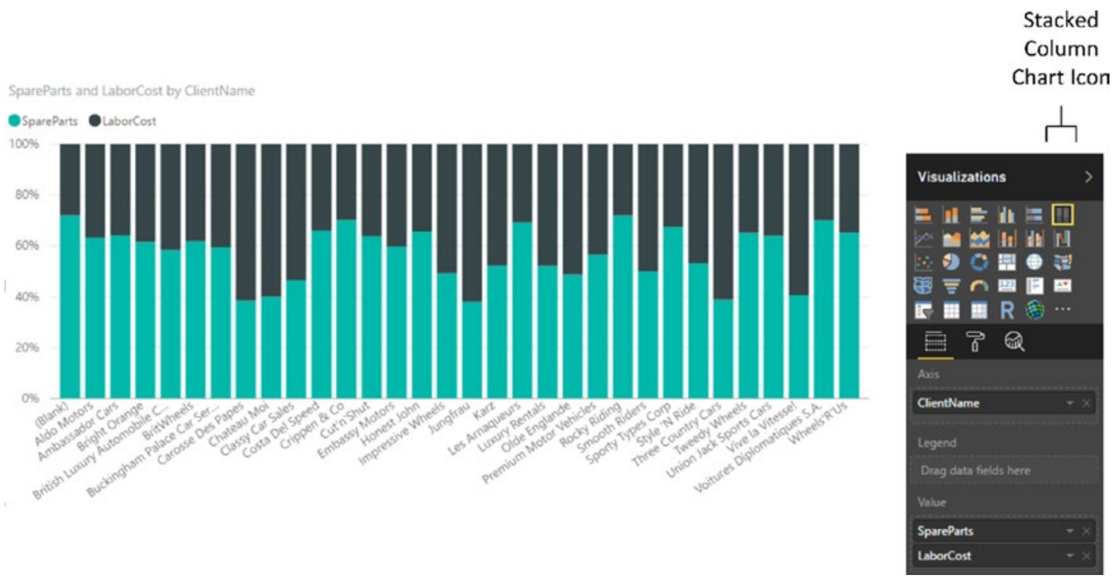
If you are not sure which chart best conveys your message, then try them all, one after another. All it takes is a single click of the relevant icon in the visuals gallery to convert a chart to another chart type. Indeed, sometimes the differences can be extremely subtle. For instance, the essential difference between the area chart and the stacked area chart is the X (vertical) axis. In the case of Figure 16-18, this is a cumulative value. Figure 16-17, in contrast, shows two values directly compared one to the other with the higher value placed behind the lower value.

■ **Note** Any of these chart types can be used to display a single data series if you want to. It all depends on the effect and the clarity of the insights that you are projecting using the chart.

100% Stacked Column and Bar Charts

One way to compare data from multiple datasets is to present each individual data series as a percentage of the total. Power BI Desktop includes two chart types that can do this “out of the box.” They are the 100% stacked column and 100% stacked bar charts.

Since you now know how to create charts that use multiple series of data, I will not explain how to produce these two chart types in detail; all you have to do is select the correct chart icon from the Visualizations pane. So instead, I suggest that you look at Figures 16-19 and 16-20, which show an example of each of these charts using the same data that you used to create the clustered column chart shown in Figure 16-14.



Axis
ClientName
Legend
Drag data fields here
Value
SpareParts
LaborCost

Figure 16-19. *A 100% stacked column chart*

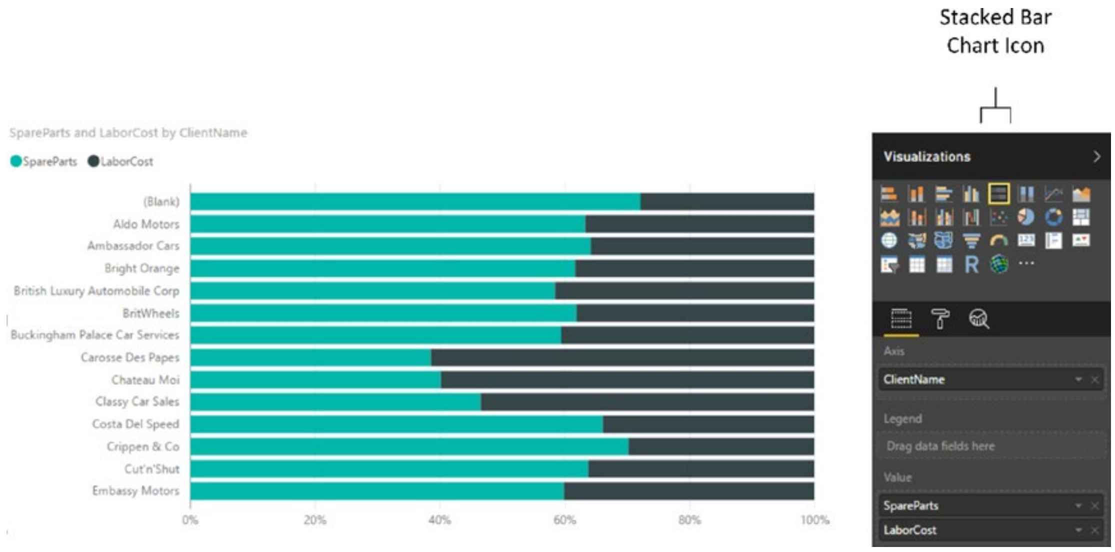


Figure 16-20. *A 100% stacked bar chart*

Scatter Charts

A scatter chart is a plot of data values against two numeric axes, and so by definition, you need two sets of numeric data to create a scatter chart. To appreciate the use of these charts, let’s imagine that you want to see the sales and margin for all the makes and models of car you sold overall. Hopefully, this allows you to see where you really made money. The following explains how to do it:

1. Using the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix, delete any existing visualizations.

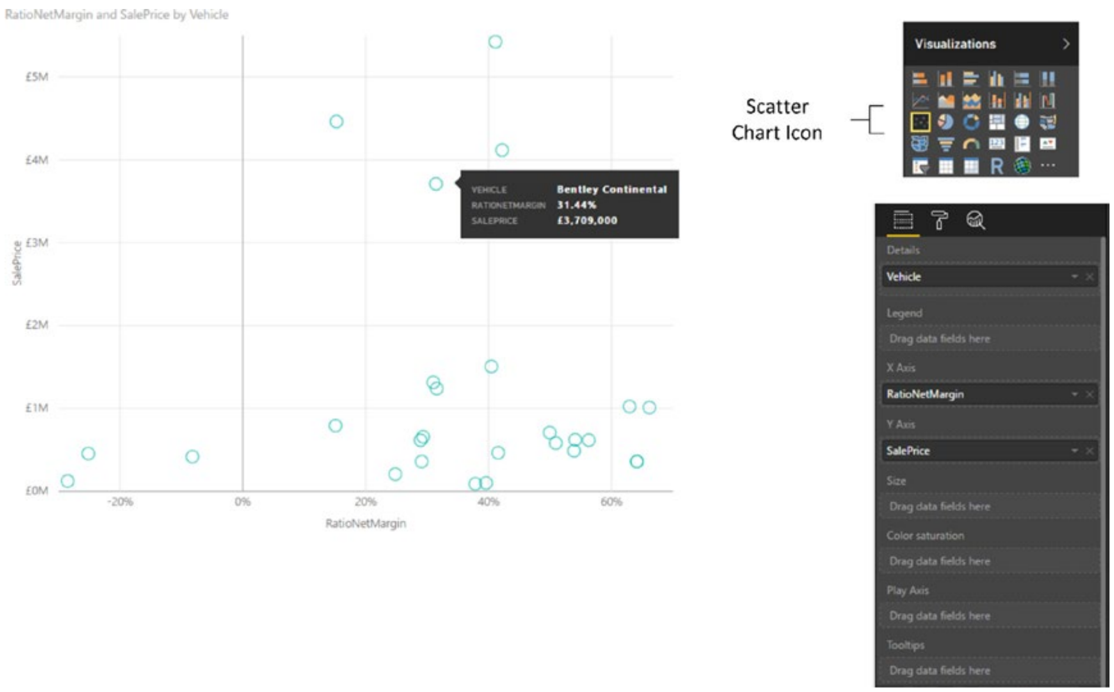
2. Create a table with the following fields in this order:

    a. Vehicle (from the Stock table)

    b. RatioNetMargin (from the InvoiceLines table)

    c. SalePrice (from the InvoiceLines table)

3. Convert the table to a scatter chart by clicking the Scatter Chart icon in the Visualizations pane. Power BI Desktop will display a scatter chart that looks like the one shown in Figure 16-21. Resize the chart to suit your taste.



VEHICLE Bentley Continental
RATIONETMARGIN 31.44%
SALEPRICE £3,709,000

Details: Vehicle
Legend: Drag data fields here
X Axis: RatioNetMargin
Y Axis: SalePrice
Size: Drag data fields here
Color saturation: Drag data fields here
Play Axis: Drag data fields here
Tooltips: Drag data fields here

Figure 16-21. *A scatter chart*

If you look at the Fields area of the Visualizations pane (which is also shown in Figure 16-21), you will see that Power BI Desktop has used the fields that you selected like this:

- Vehicle: Placed in the Details box.
- RatioNetMargin: Placed in the X Axis box. This is the vertical axis.
- SalePrice: Placed in the Y Axis box. This is the horizontal axis.

If you hover the mouse pointer over one of the points in the scatter chart, you see the data for the specific car model. Figure 16-21 shows this.

■ **Note** By definition, a scatter chart requires numeric values for both the X and Y axes. So if you add a non-numeric value to either the X Value or Y Value boxes, then Power BI Desktop converts the data to a count aggregation.

We made this chart by adding all the required fields to the initial table first. We also made sure that we added them in the right order so the scatter chart would display correctly the first time. In the real world of interactive data visualization, things may not be quite this coherent, so it is good to know that Power BI

Desktop is very forgiving. And, it lets you build a scatter chart (just like any other chart) step by step if you prefer. In practice, this means that you can start with a table containing just two of the three fields that are required at a minimum for a scatter chart, convert the table to a scatter chart, and then add the remaining data field. Power BI Desktop always attributes numeric or time fields to the X and Y axes (in the order in which they appear in the Fields box) and places the first descriptive field into the Details box.

Once a scatter chart has been created, you can swap the fields around and replace existing fields with other fields from the tables in the data to your heart's content.

Bubble Charts

A variant of the scatter chart is the bubble chart. This is one of my favorite chart types, though of course you cannot overuse it without losing some of its power. Essentially, a bubble chart is a scatter chart with a third piece of data included. So whereas a scatter chart shows you two pieces of data (one on the X axis, one on the Y axis), a bubble chart lets you add a third piece of information, which becomes the *size* of the point. Consequently, each point becomes a bubble.

The best way to appreciate a bubble chart is to create one. So here we assume that you want to look at the following for all makes of car sold in a single chart:

- The total sales
- The net margin ratio
- The gross margin

This explains how a bubble chart can do this for you:

1. Using the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix, delete any existing visualizations.
2. Create a table with the following fields from the SalesData table, in this order:
   a. Make (from the Stock table)
   b. SalePrice (from the InvoiceLines table)
   c. RatioNetMargin (from the InvoiceLines table)
   d. Gross Margin (from the InvoiceLines table)
3. Convert the table to a scatter chart by clicking the Scatter Chart icon. Yes, a bubble chart is a scatter chart, with a fresh tweak added.
4. Drag the ClientSize field from the Clients table into the Legend well of the Visualizations pane. Power BI Desktop will display a bubble chart that looks like that shown in Figure 16-22.

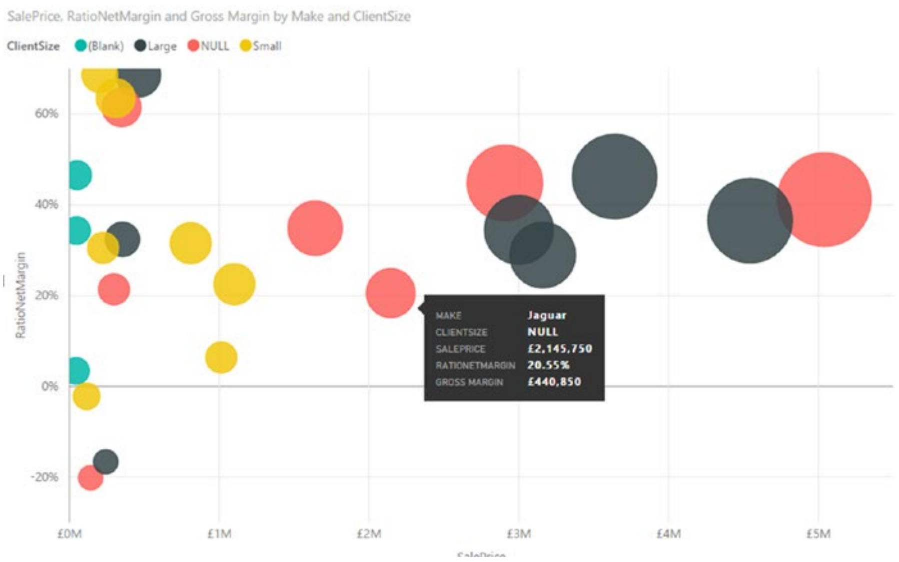


Figure 16-22. *An initial bubble chart*

5. Resize the chart if you need to.

If you look at the field well in the Visualizations pane (shown in Figure 16-23), you will see that Power BI Desktop has used the fields that you selected like this:

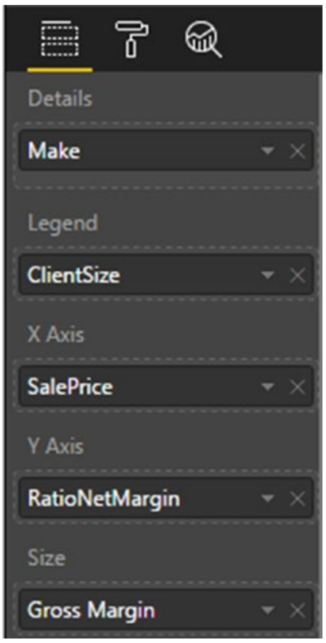

Details
Make
Legend
ClientSize
X Axis
SalePrice
Y Axis
RatioNetMargin
Size
Gross Margin

Figure 16-23. *The Fields well of the Visualizations pane for a bubble chart*

- *Make*: Placed in the Details box. This defines the core bubbles.
- *ClientSize*: Added to the Legend box. This creates bubbles for each combination of the detail element (make) and legend item (client size).
- *SalePrice*: Placed in the X Axis box. This is the vertical axis.
- *RatioNetMargin*: Placed in the Y Axis box. This is the horizontal axis. Each bubble is placed at the intersection of the values on the vertical and horizontal axes.
- *Gross Margin*: Placed in the Size box. This defines the size of the points, which have consequently become bubbles of different sizes.

Hover the mouse pointer over one of the points in the bubble chart. You will see all the data that you placed in the Fields list Layout section for each make, including the Gross Margin. You can see this in Figure 16-23.

If a bubble chart containing this many data points is simply too overloaded to be truly useful, then you have an alternative way of presenting the data. Continuing with the chart that you just created:

1. Remove the ClientSize field from the Legend well in the Visualizations pane.
2. Drag the SpareParts field from the Stock table into the Color Saturation well of the Visualizations pane. The chart will now look like the one shown in Figure 16-24, where the relative intensity of the color of each bubble is defined by the spare parts value.

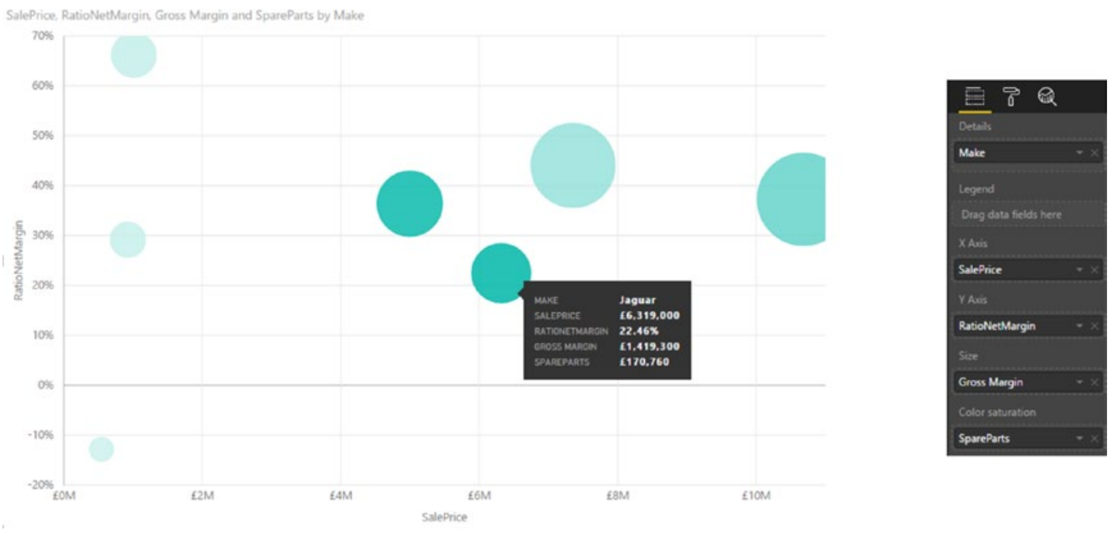


Figure 16-24. *A bubble chart using color saturation*

Waterfall Charts

We are very near the end of our tour of Power BI Desktop chart types. What I want to look at now is the antepenultimate chart type Power BI Desktop offers—the waterfall chart. This chart type is excellent at displaying the component parts of a final figure; in this example, it is used to show how the various makes sold make up the total sales figure.

1. Open the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix and delete any existing visualizations.
2. Click the Waterfall Chart icon You can see this in Figure 16-25. An empty waterfall chart will appear on the dashboard canvas.
3. Expand the Stock table and check the following fields:
   a. Make
   b. CostPrice
4. Resize the chart to suit your aesthetic requirements. It should look something like Figure 16-25.

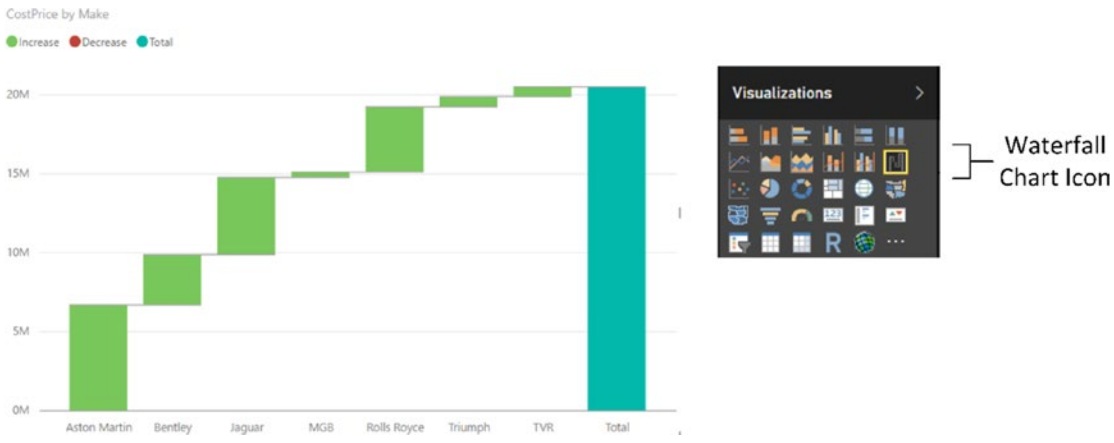


Figure 16-25. *A waterfall chart*

Waterfall charts can be extremely useful when you are analyzing the constituent elements of a whole. For instance, you could try using one to break down all the cost elements of a sale.

Ribbon Charts

A fairly recent addition to the range of chart options that Power BI Desktop has on offer is the ribbon chart. Ribbon charts virtually always use a time element as the X axis, and are designed to show evolution over time. More specifically, they always place the highest value as the upper ribbon of the chart.

This is probably best understood with the help of an example. So I suggest that you try out the following:

1. Click the Ribbon Chart icon in the Visualizations pane (this is shown in Figure 16-26).

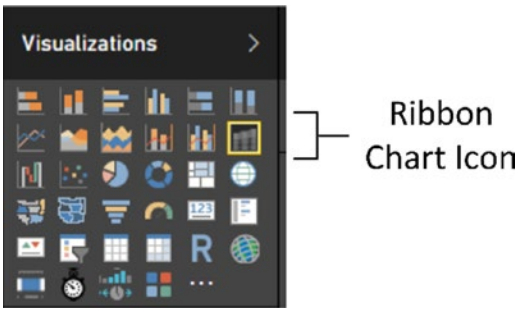


Figure 16-26. *The Ribbon Chart icon*

2. In the Fields list, click the following fields:
   a. MonthAndYear (in the DateDimension table)
   b. SalePrice (in the InvoiceLines table)
   c. Make (in the Stock table)
3. Resize the chart until it looks like the one shown in Figure 16-27.

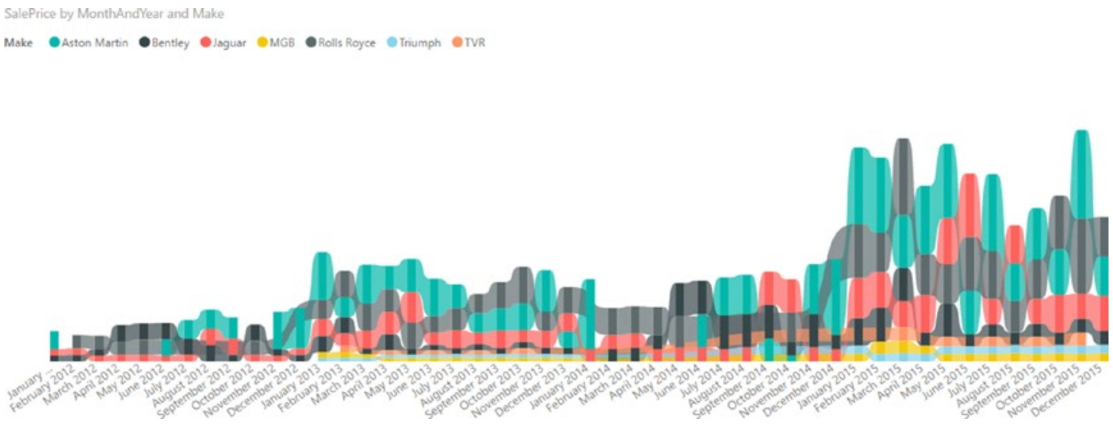


Figure 16-27. *A ribbon chart*

As you can see, the ribbon chart shows the ebb and flow of sales of different makes of vehicle over time. Indeed—and at the risk of anticipating the contents of Chapter 21—I suggest that you click one of the makes in the chart legend. This will highlight the make and show its evolution over time.

Dual-Axis Charts

To conclude our tour of chart types, let's take a look at a couple of charts that combine two of the basic chart types that you have seen previously in this chapter:

- Line and clustered column chart
- Line and stacked column chart

Let's discuss each of these in turn.

Line and Clustered Column Chart

Suppose that the CEO of Brilliant British Cars has decided to embark on an analysis of vehicle purchases. He wants to isolate any interesting correlations that could influence purchases in order to maximize profits. So, determined to satisfy his request (or possibly to humor him) you have in turn decided to take a look at indirect costs and mileage to see if there are any correlations.

1. Open the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix and delete any existing visual.
2. Click the Line and Clustered Column Chart icon. An empty line and clustered column chart will appear on the dashboard canvas.
3. Expand the Stock table and check the following fields:
   a. Make (this will be added to the Shared Axis box)
   b. LaborCost (this will be added to the Column Values box)
   c. SpareParts (this will be added to the Column Values box)
4. Also from the Stock table, drag the Mileage field into the Line Values box in the field well of the Visualizations pane.
5. Resize the chart to suit your aesthetic requirements. It should look something like Figure 16-28.

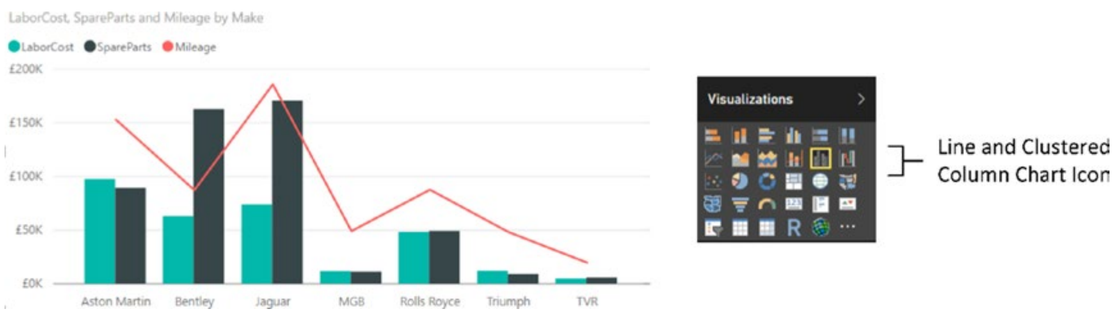


***Figure 16-28.** A line and clustered column chart*

As you can see, this chart combines a clustered column chart that displays the labor and parts costs using the left axis with a line chart that displays the mileage using the right axis. Both of these charts share the X (or horizontal) axis.

Moreover, this kind of analysis makes it immediately clear that there is one make where low mileage does not necessarily mean lower repair costs. So the boss should be happy that you have isolated this unexpected correlation.

Line and Stacked Column Chart

One of the key advantages of dual-axis charts is that the two X axes can have vastly different scales. So, for instance, when you want to see how the parts and labor cost relate to the purchase cost (which is orders of magnitude higher than the other two costs), a line and stacked column chart can be really useful to get a clearer view of the data.

1. Open the file C:\PowerBiDesktopSamples\CH16\CarSalesDataForReports.pbix and delete any existing visualizations.
2. Expand the Stock table and check the following fields to display a table on the dashboard canvas:
    a. Make
    b. LaborCost
    c. SpareParts
3. Click the Line and Stacked Column Chart icon. The table will be converted to a line and stacked column chart.
4. Also from the Stock table, drag the CostPrice field into the Field well of the Visualizations pane. Place it in the Line Values box.
5. Resize the chart to suit your aesthetic requirements. It should look something like Figure 16-29.

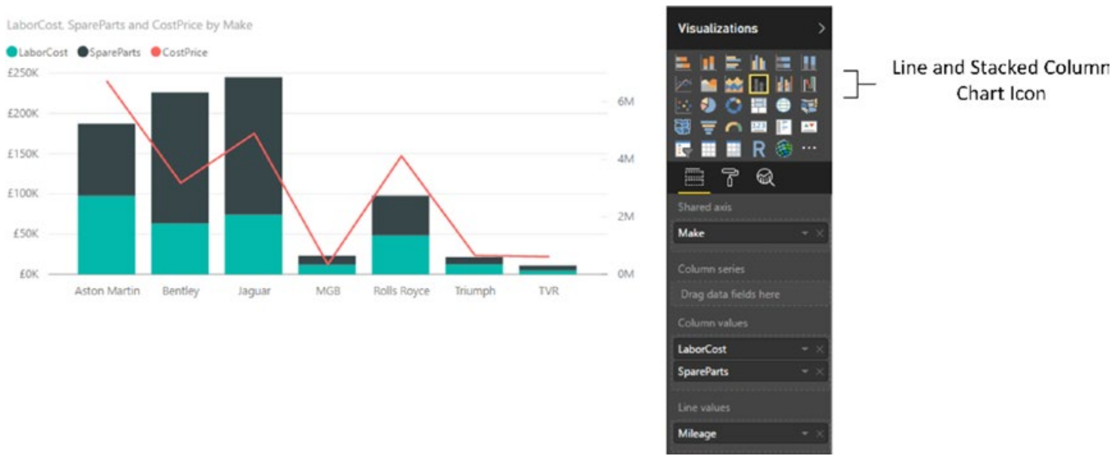


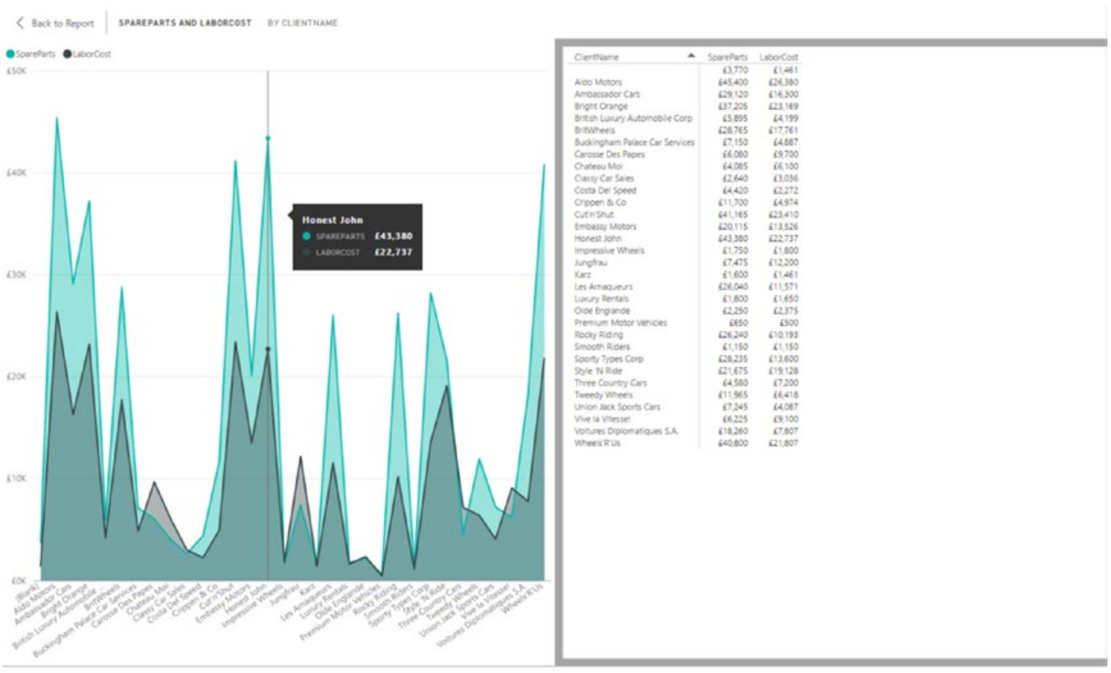
Figure 16-29. *A line and stacked column chart*

This way you can compare values where the use of a single chart type would lead to one value (the cost price in this example) dwarfing the other values to the point of making them unreadable.

Data Details

To conclude our tour of chart types, I just want to make a few comments:

- You can always see exactly what the figures behind a bar, column, line, point, or pie segment are just by hovering the mouse pointer over the bar (or column, or line, or pie segment). This works whether the chart is its normal size or has been popped out to cover the Power BI Desktop report area.
- However much work you have done to a chart, you can always switch it back to a table if you want. Simply select the chart, and select the required table type from the Table button in the Design ribbon. If you do this, you see that the table attempts to mimic the design tweaks that you applied to the chart, keeping the font sizes the same as in the chart, and the size of the table identical to that of the chart. Should you subsequently switch back to the chart, then you should find virtually all of the design choices that you applied are still present—unless, of course, you made any changes to the table before switching back to the chart visualization.
- You can always juxtapose the raw data that powers a chart under the chart itself, as you could with text-based visuals. Simply click the ellipses at the top right of a chart and select See Data to display the source data. You can see an example of this in Figure 16-30 (where you can also see how a data point is displayed in line, area, and stacked area charts).



ClientName	SpareParts	LaborCost
	£3,770	£1,461
Aldo Motors	£45,400	£26,380
Ambassador Cars	£29,120	£16,300
Bright Orange	£37,205	£23,169
British Luxury Automobile Corp	£5,895	£4,199
BritWheels	£28,765	£17,761
Buckingham Palace Car Services	£7,150	£4,887
Carosse Des Papes	£6,080	£9,700
Chateau Moi	£4,085	£6,100
Classy Car Sales	£2,640	£3,036
Costa Del Speed	£4,420	£2,272
Crippen & Co	£11,700	£4,974
Cut'n'Shut	£41,165	£23,410
Embassy Motors	£20,115	£13,526
Honest John	£43,380	£22,737
Impressive Wheels	£1,750	£1,800
Jungfrau	£7,475	£12,200
Karz	£1,600	£1,461
Les Arnaqueurs	£26,040	£11,571
Luxury Rentals	£1,800	£1,650
Olde Englande	£2,250	£2,375
Premium Motor Vehicles	£650	£500
Rocky Riding	£26,240	£10,193
Smooth Riders	£1,150	£1,150
Sporty Types Corp	£28,235	£13,600
Style 'N Ride	£21,675	£19,128
Three Country Cars	£4,580	£7,200
Tweedy Wheels	£11,965	£6,418
Union Jack Sports Cars	£7,245	£4,087
Vive la Vitesse!	£6,225	£9,100
Voitures Diplomatiques S.A.	£18,260	£7,807
Wheels'R'Us	£40,800	£21,807

Figure 16-30. *Displaying pop-out data for a chart*

To switch back to the dashboard canvas, simply click Back to Report at the top left of the chart.

Drilling into and Expanding Chart Data Hierarchies

An extremely useful aspect of Power BI Desktop charts is that you can use them as an interactive presentation tool. One of the more effective ways to both discover what your data reveals and deliver the message to your public is to drill into, or expand, layers of data. All the standard Power BI chart types let you do just this.

Drill down works with all the standard chart types. This means that you can drill into:

- Stacked bar charts
- Stacked column charts
- Clustered bar charts
- Clustered column charts
- 100 % stacked bar charts
- 100 % stacked column charts
- Line charts
- Area charts
- Stacked area charts
- Line and stacked column charts
- Line and clustered column charts
- Waterfall charts
- Scatter charts
- Bubble charts
- Pie charts
- Funnel charts
- Donut charts
- Tree map charts

The key point to remember when you are creating drill-down charts is that you *must* place the fields that you wish to drill into in the Axis area (the Shared Axis area for double-axis charts) of the field well, *not* in the Legend area (and not in the Details area for a pie chart).

Drill Down

Fortunately, you use the same techniques that you saw in the previous chapter when navigating data hierarchies to drill down into charts. So extending this approach from tables and matrices to charts is really easy. Here, then, is a simple example of how you can analyze the hierarchy of makes, models, and colors of vehicles sold in a drill-down chart:

1. Click the Clustered Column Chart icon in the Visualizations pane. An empty chart visual will be created on the dashboard canvas.
2. Drag the following fields into the Axis area of the field well in this order:
    a. Make (from the Stock table)
    b. Model (from the Stock table)
    c. Color (from the Colors table)
3. Drag the SalePrice (from the InvoiceLines table) and the Mileage (from the Stock table) fields onto the chart. The chart will look like the one in Figure 16-31. As you can see, it only displays the top-level element from the Axis field well—the make of vehicle.

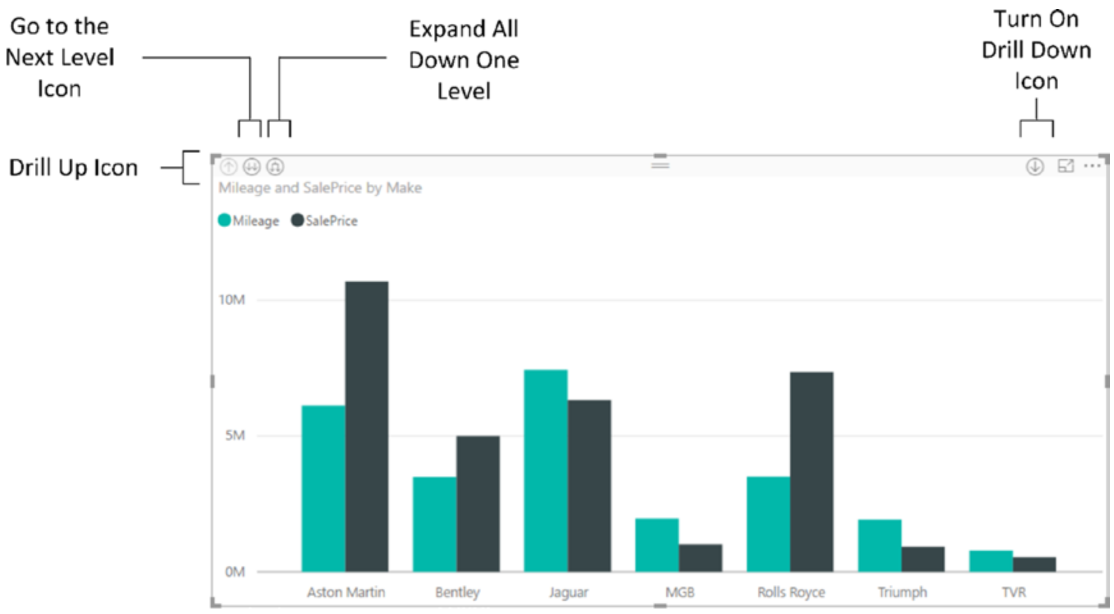


Figure 16-31. *A column chart displaying the top level of data in a hierarchy*

4. Click the Turn on Drill Down icon at the top right of the chart. This icon will become a white arrow on a dark background.

5. Click the column for Aston Martin. The chart will now display the models of Aston Martin sold, as you can see in Figure 16-32.

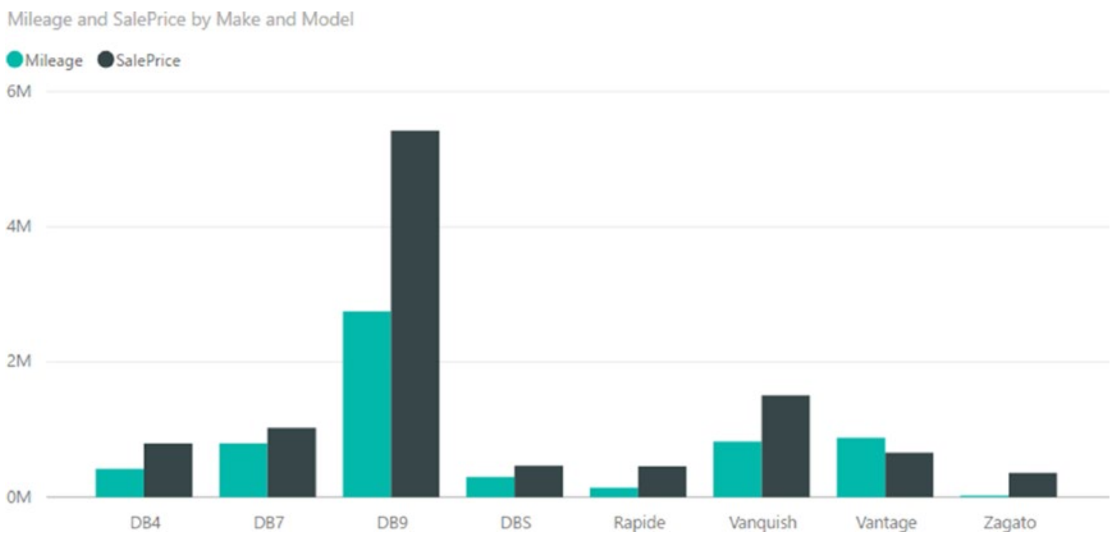


Figure 16-32. *A column chart displaying the second level of data in a hierarchy*

6. Click the column for DB9. The chart will now display the colors of Aston Martin DB9 sold, as you can see in Figure 16-33.

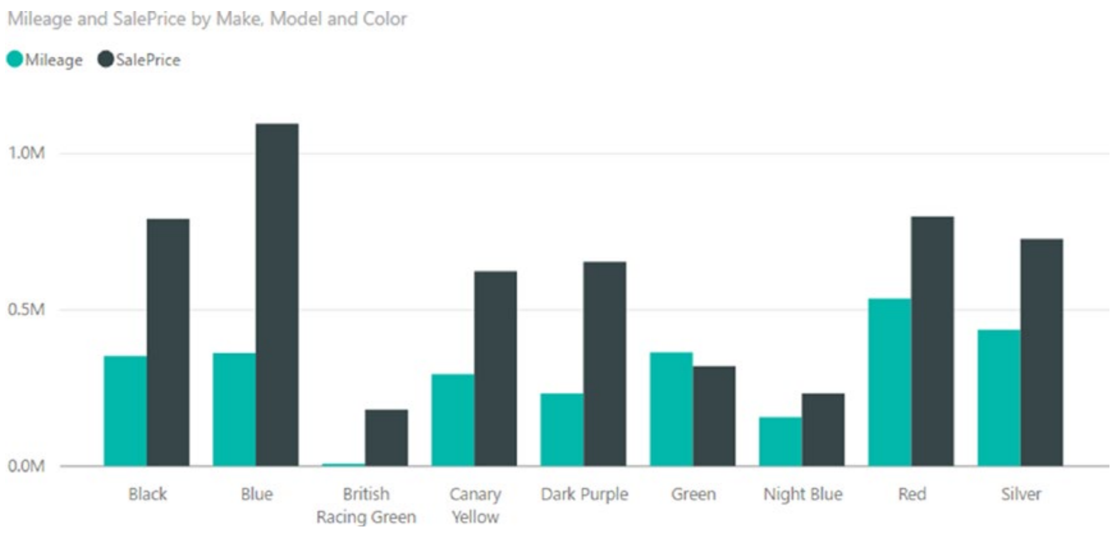


Figure 16-33. *A column chart displaying the lowest level of data in a hierarchy*

Once you have drilled down to a lower level in a chart, you can always return to the previous level (and from there continue drilling up until you reach the top level) by clicking the Drill Up icon at the top left of the chart.

Expand All Down One Level

The Expand All Down One Level icon produces a different result to drilling down by clicking a bar in the chart. If you click the Expand All Down One Level icon, you see *all* the elements at the next level down, not just the elements in the hierarchy that are a subgroup. To see this, try out the following steps:

1. Re-create the chart from the previous section (follow steps 1 through 3) until you can once again see the chart from Figure 16-31.
2. Click the Expand All Down One Level icon.
3. Resize the chart to enhance its visibility. You will see a chart that looks like the one shown in Figure 16-34.

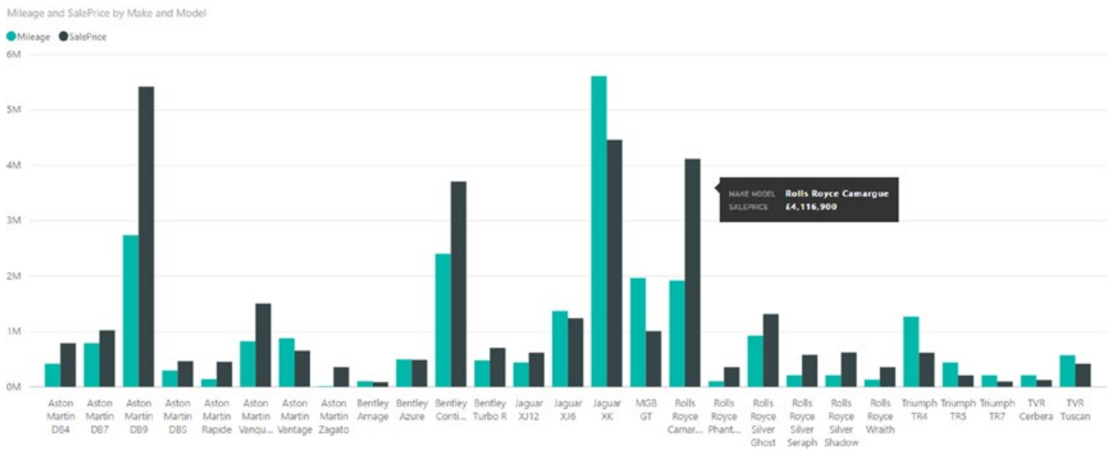


Figure 16-34. *Expanding a chart down one level*

4. Click the Expand All Down One Level icon a second time.

5. Resize the chart if necessary. This time the chart looks like the one shown in Figure 16-35.

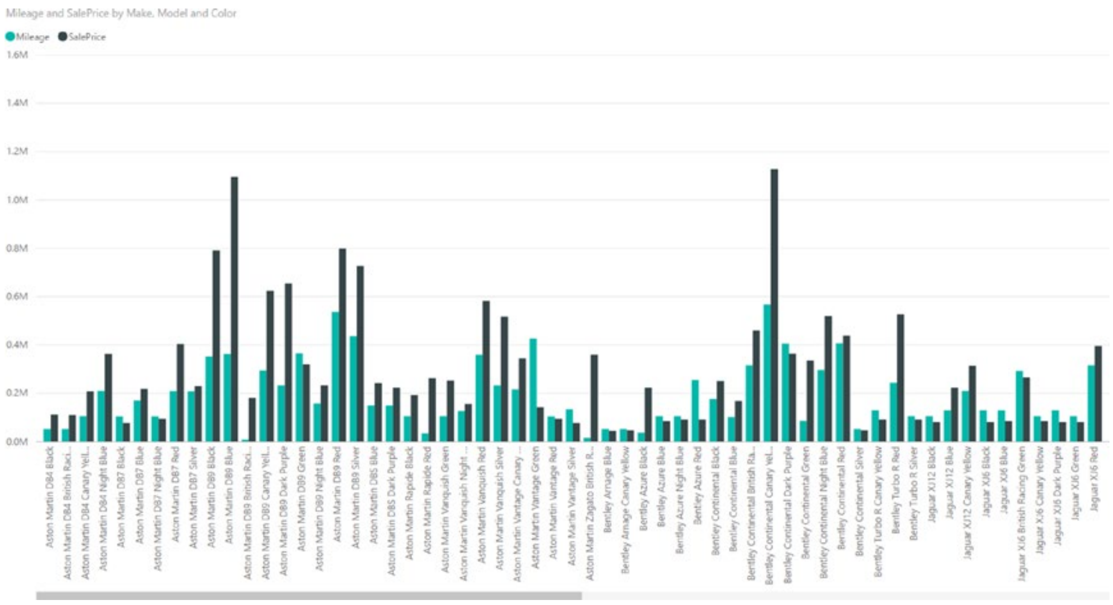


Figure 16-35. *Expanding a chart down one more level*

Go to the Next Level

The final hierarchy navigation technique that you can apply is to go to the next level. To test this, try out the following steps:

1. Re-create the chart from the last but one section (follow steps 1 through 3).
2. Click the Go to the Next Level icon.

3. Resize the chart to enhance its visibility. You should see a chart like the one shown in Figure 16-36.

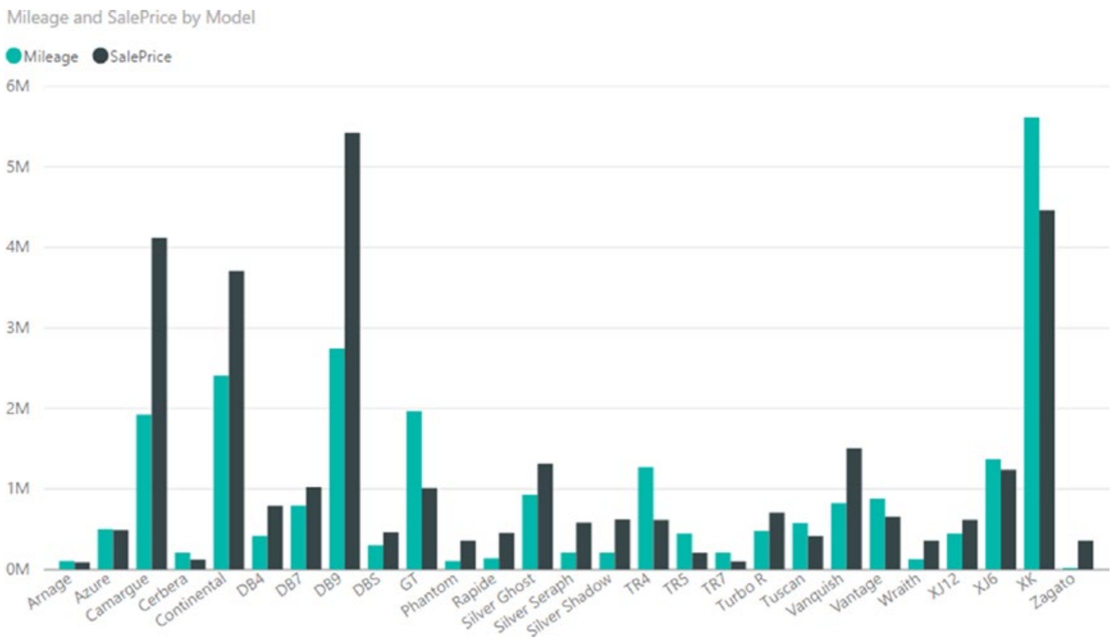


Figure 16-36. *Going to the next level in a chart*

4. Click the Go to the Next Level icon one more time. You will see a chart that looks like the one shown in Figure 16-37.

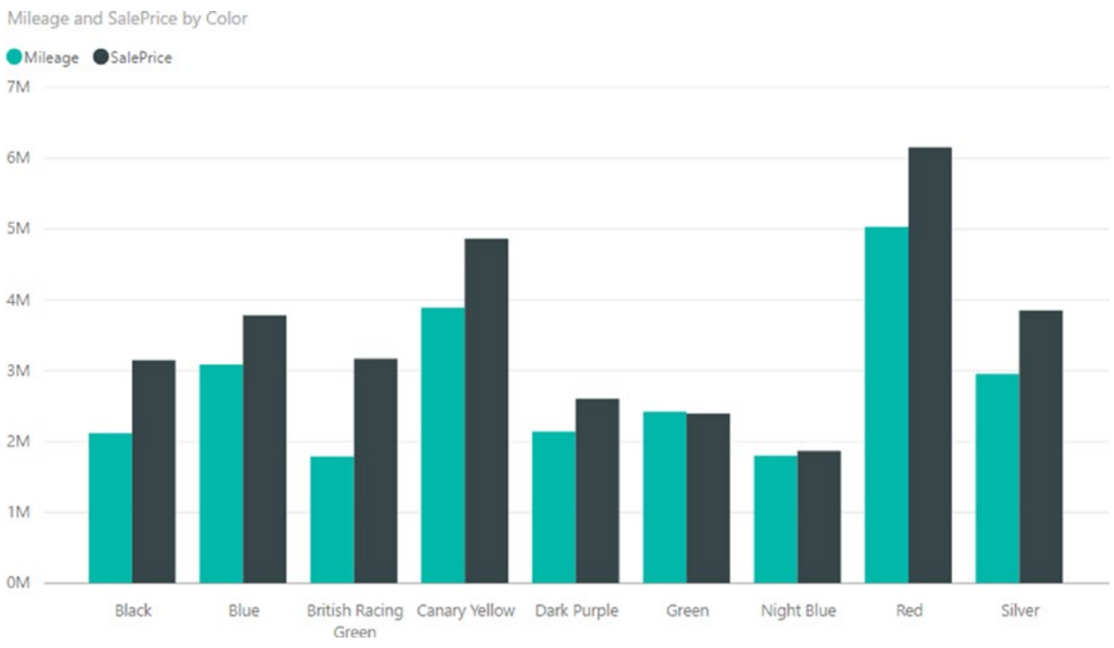


Figure 16-37. *Going to a deeper level in a chart*

To resume, the three options that you can use to navigate a chart hierarchy are

- *Drill Down*: This will only show the data at the next level for the *specific element* that you clicked.
- *Expand All Down One Level*: This will show the data at the next level for *all* elements.
- *Go to the Next Level*: This will display data *only for the lower level* in the data hierarchy.

■ **Note** Whichever technique you used to display a lower level of data, you can always go up to the previous level by clicking the Drill Up icon.

Including and Excluding Data Points

You saw in Chapter 15 that you can include and exclude records from a matrix. Well, you can apply this technique to charts as well. This can help you to remove clutter, discard outliers, or simply focus on a selected set of data points. Here's an example of how to exclude data points:

1. Create a clustered column chart like the one shown in Figure 16-38 using the following data elements:

    a. Axis: Make (from the Stock table)

    b. Legend: Model (from the Stock table)

    c. Value: SalePrice (from the InvoiceLines table)

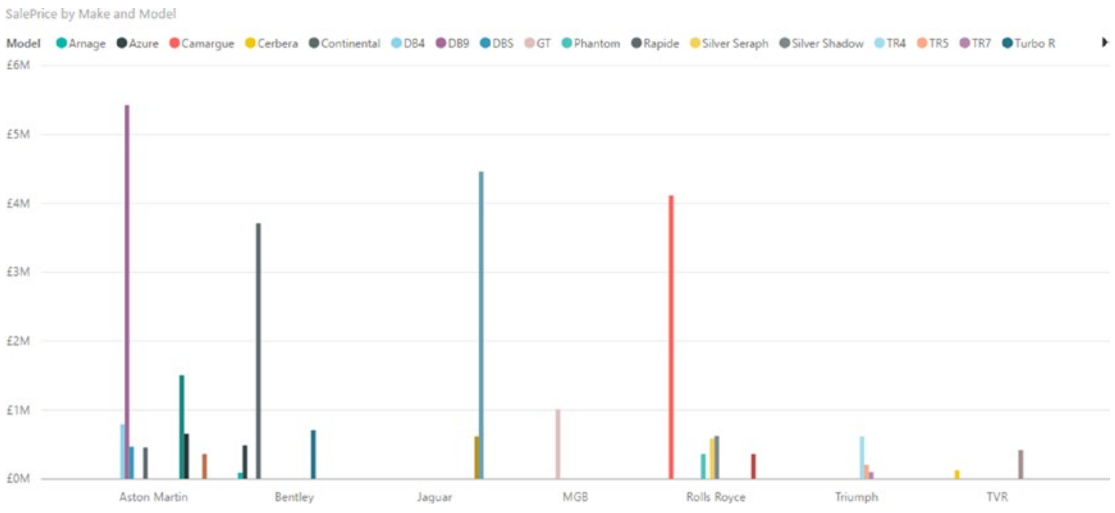


Figure 16-38. *A column chart before excluding data points*

2. Ctrl-click to select the four tallest columns.

3. Right-click any of the selected columns and choose Exclude from the popup menu. The chart will look like the one shown in Figure 16-39.

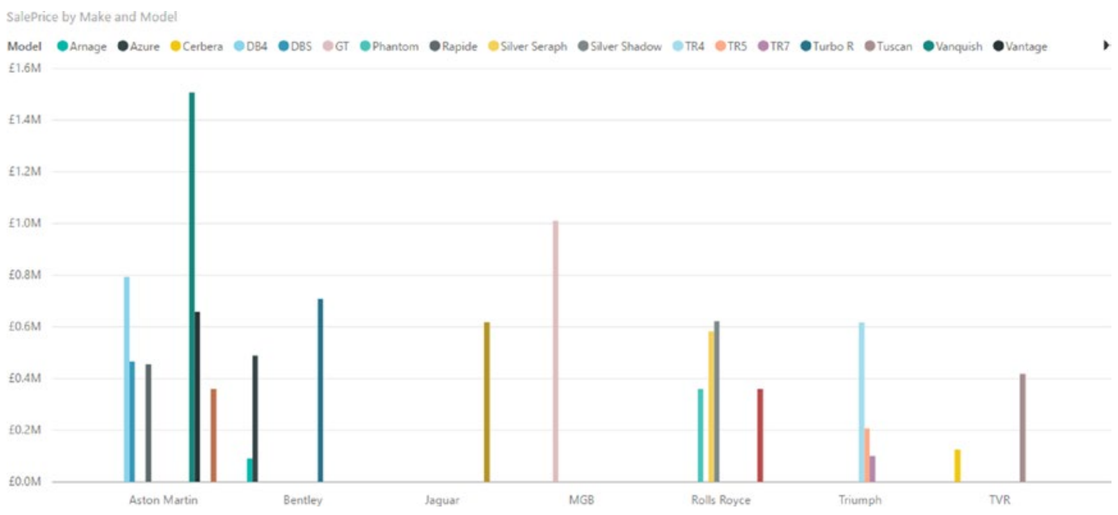


Figure 16-39. *A column chart after excluding data points*

The remaining data is exactly the same as before, but removing the very tall columns allows you to see more clearly any differences among the remaining data elements.

As a final point, you can see from Figure 16-39 that a chart whose legend contains too many elements to view comfortably has a scroll arrow to allow you to view the remaining items in the legend.

Conclusion

This chapter took you on a tour of the available chart types that you can use in your Power BI Desktop reports and dashboards. These extend from the classic pie, line, column, and bar charts to the less common funnel, donut, area, scatter, bubble, and waterfall charts. You saw that there are charts to suit a single data series and others that can handle multiple series of data. To add extra effect, you saw how to create mixed chart types, create 100% bar and column charts, and tweak axis elements.

However, these charts can be presented at a higher level if you decide to add some compelling formatting. This is what you will learn to do in the next chapter.

CHAPTER 17

Formatting Charts in Power BI Desktop

Now that you have mastered basic charts, it is time to move on to the next step and learn how to tweak your charts to the greatest effect. The next few sections are devoted to the various techniques available in Power BI Desktop to give your charts real clarity and power. Some of these enhancements apply to all chart types whereas others are specific to a single type of chart—or even one or two chart types.

Sometimes you change the configuration of a chart—and consequently enhance it—by altering the mapping of the data to the chart elements. However, on most occasions, you enhance a chart by modifying the various formatting options that are displayed when you select a chart, and then clicking the paint roller icon (the Format icon) in the Visualizations pane. I refer to this as the “format section” for ease of reference from now on.

As well as looking at formatting charts in this chapter, we will also look at a really interesting aspect of Power BI Desktop. This will involve a tour of the built-in analytics functions that are available. These can help you to isolate and highlight key trends and information in the charts that you create.

In this chapter you will use the C:\PowerBiDesktopSamples\CH17\CarSalesDataForReports.pbix Power BI Desktop file (that is available on the Apress web site for download) as the basis for any charts that you will create.

Multiple Chart Formatting

It is worth noting from the outset that you can apply identical modifications to a set of charts on the same dashboard page by Ctrl-clicking to select multiple charts before you make any formatting changes.

However, simultaneous modification of several charts only works if *all* the selected charts are the same type.

Chart Legends

If you have a chart with more than one field that provides the values on which the chart is based (or if you are creating a pie or donut chart), then you see a legend appear automatically.

You can format a legend by modifying any of the following options:

- Legend display
- Position
- Legend title

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_17

Let's look at each of these in turn.

■ **Note** A legend can only be displayed if there is more than one field added to the Value well (except in the case of pie and donut charts).

Legend Display

Power BI Desktop may automatically display legends in some cases, but you have the final decision as to whether a legend is required. To turn a legend off, do the following:

1. Select the chart whose legend you want to hide.
2. In the Visualizations pane, click the Format icon to display the format section.
3. Drag the Legend button to the left (or click to its left) so that it changes from a full circle to an empty circle.

The legend will disappear from the selected chart.

Legend Position

The default for the legend is for it to be placed at the top left of the chart. However, you can choose where to place the legend inside the chart area. Follow these steps:

1. Select the chart whose legend you want to modify.
2. In the Visualizations pane, click the Format icon.
3. Expand the Legend section.
4. Select one of the available legend positions from the Position popup.

The available options are given in Table 17-1.

Table 17-1. *Legend Position Options*

Legend Option	Description
Top	The legend is displayed above the chart on the left.
Bottom	The legend is displayed below the chart on the left.
Left	The legend is displayed at the left of the chart at the top.
Right	The legend is displayed at the right of the chart at the top.
Top Center	The legend is displayed above the chart on the left in the center.
Bottom Center	The legend is displayed below the chart on the left in the center.
Left Center	The legend is displayed at the left of the chart in the middle.
Right Center	The legend is displayed at the right of the chart in the middle.

If one of the legend options is grayed out, it is the option that is currently active.

Legends can require a little juggling until they display their contents in a readable way. This is because the text of the legend is often truncated when it is initially displayed. If this is the case, the only real option is to modify the chart size or the legend font size.

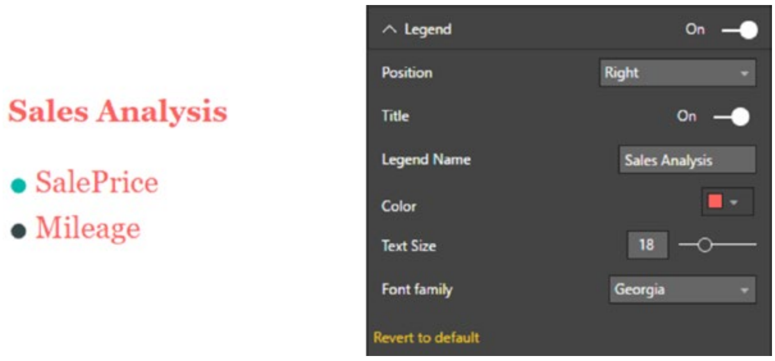
■ **Note** You cannot add a legend to an area, column, bar, or line chart that only has a *single* data series.

Legend Title

You can add a title to a legend if you want.

1. Create a bar chart using the following fields (I am assuming that you are, by now, familiar enough with the source data tables to locate these fields):
   a. Make
   b. SalePrice
   c. Mileage
2. In the Visualizations pane, click the Format icon.
3. Expand the Legend section.
4. Switch the Title button to the On position.
5. Select a position from the popup list of available options.
6. Enter the legend title in the Legend Name box.
7. Select a color for the text in the legend from the color palette.
8. Select a font family and font size.

An example of a legend with a title is shown in Figure 17-1.



Legend On
Position Right
Title On
Legend Name Sales Analysis
Color 🟥
Text Size 18
Font family Georgia

Figure 17-1. A legend with a title

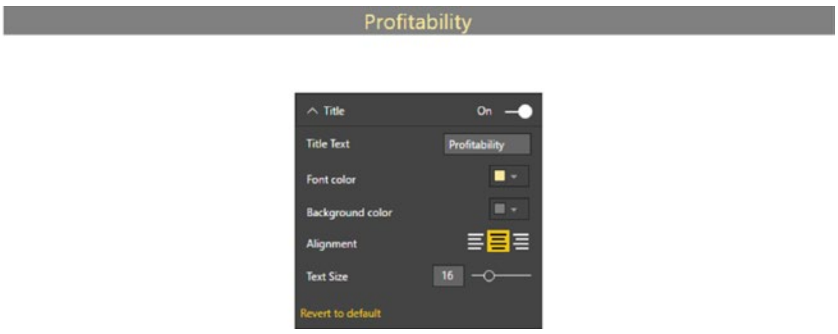
Chart Title

Each chart is created with a title explaining what the chart is displaying; that is, the fields on which it is based. The available options are fairly simple:

- Hide or display the title
- Modify the default text of the title
- Change the font color
- Change the background color for the title
- Alter the alignment

Let's see how to apply all of these options to the title of the chart that you created earlier:

1. Select the chart containing the title that you want to modify.
2. In the Visualizations pane, click the Format icon.
3. Expand the Title section.
4. Ensure the Title button is in the On position. Alternatively, switch to the Off position to hide the title.
5. Modify the title text in the Title Text box. In this example, I set it to **Profitability**.
6. Click the popup menu triangle to the right of Font Color and select a color from the palette of available colors.
7. Click the popup menu triangle to the right of Background Color and select a color from the palette of available hues.
8. Click the middle icon to the right of Alignment. This will center the text relative to the width of the table.
9. Adjust the Text Size slider to set a larger font size. You can see the result of these kinds of adjustment in Figure 17-2.



Title On
Title Text Profitability
Font color
Background color
Alignment
Text Size 16
Revert to default

Figure 17-2. *Chart title adjustments*

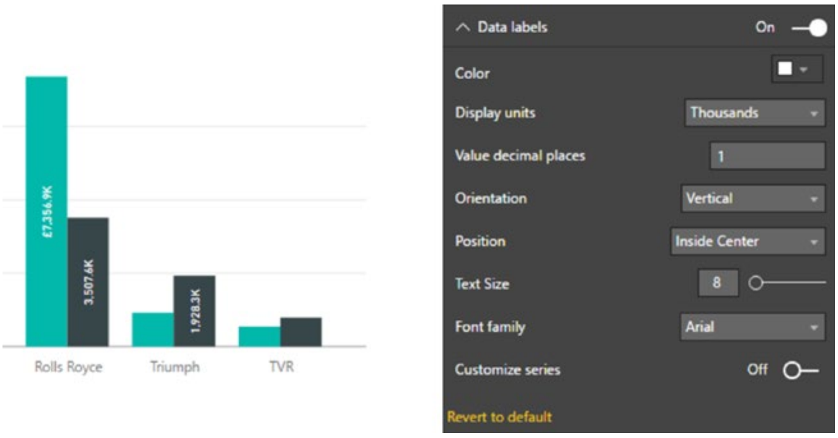
You can always add further annotations to a chart using free-form text boxes. This technique is described in Chapter 22.

Chart Data Labels

As you have seen already, you can display the exact data behind a column, bar, or point in a line chart simply by hovering the mouse pointer over the data that interests you. Yet there could be times when you want to display the values behind the chart permanently on the visualization. This is where data labels come into play.

To add data labels to a chart (in this example, I continue using the chart that you created at the start of this chapter), all you have to do is this:

1. Select the chart to which you wish to add data labels.
2. In the Visualizations pane, click the Format icon.
3. Expand the Data Labels section.
4. Set the Data Labels button to On.
5. Select a display unit for the figure in the card by clicking the popup list of display units. I will apply thousands as the unit in this example.
6. Select a number of decimals to be used for the card data by clicking the up and down triangles to the right of the Precision box. You can also enter the number of decimals directly if you prefer.
7. Select a color for the data labels from the popup palette of colors.
8. Choose an alignment style (vertical or horizontal) from the Orientation popup list.
9. Select a position for the label relative to the data (this only applies to bar and column charts).
10. Tweak the text size and font if you wish. You can see a sample of the output from this in Figure 17-3.



Data labels On
Color
Display units: Thousands
Value decimal places: 1
Orientation: Vertical
Position: Inside Center
Text Size: 8
Font family: Arial
Customize series: Off
Revert to default

Figure 17-3. *Data labels in a column chart*

■ **Note** When applying data labels to column, bar, and line charts, you notice that sometimes Power BI Desktop cannot physically place all the data labels exactly where the option that you have selected implies that they should appear. This is because sometimes there is simply not enough space inside a bar or column to fit the figures. In these cases, Power BI Desktop places the data outside the bar or column. On other occasions, the data cannot fit outside a line, column, or bar without being placed above the upper end of the axis. Here again, Power BI Desktop tweaks the presentation to get as close as possible to the effect that you asked for. This can mean that data labels are not displayed at all.

The various options for positioning the data labels are given in Table 17-2.

Table 17-2. *Data Label Position Options*

Data Label Position	Description
Auto	Power BI Desktop places the data label for the best effect.
Inside End	The data label is placed inside the bar at the right or inside the column at the top.
Outside End	The data label is placed outside the bar at the right or outside the column at the top.
Inside Center	The data label is placed inside the center of the bar or column.
Inside Base	The data label is placed inside the bar at the left or inside the column at the bottom.

There are a few final points to note on the subject of data labels:

- Pie charts do not display data labels like other charts, but instead add the figure to the callout.
- Scatter charts and balloon charts do not display data labels.
- You can, if you prefer, show and tweak data labels for one or all of the data series in a chart. Simply click the Customize Series button to set it to On and then select the data element to format from the popup list. You can then format this selected series in the same way that you formatted the data labels for the entire chart.

Chart Background

If you need to alter the aesthetics of a chart, or should the need arise to make one chart stand out from the others in a dashboard, you can also modify the background color of the entire chart. So, continuing with the chart that you created previously, this is what you can do:

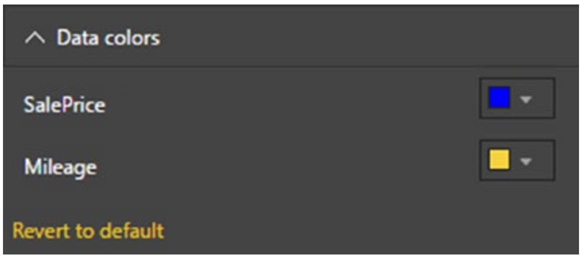
1. Select the chart whose background you wish to modify.
2. In the Visualizations pane, click the Format icon.
3. Expand the Background section.
4. Ensure that the Background On/Off switch is set to On (a full circle on the right).
5. Click the popup menu triangle to the right of Background Color and select a color from the palette of available colors.
6. Move the Transparency slider to the left or right to increase or decrease the transparency (and consequently the intensity) of the background color.

Data Colors

All charts allow you to specify the color of each and every data series. Of course, the amount of effort involved will depend on the number of data series that you want to alter. The way to do it is as follows:

1. Select the chart for which you wish to modify the data colors.
2. In the Visualizations pane, click the Format icon.
3. Expand the Data Colors section. The list of chart data series (or data elements for certain chart types) will be displayed.
4. Select a color for each data element or series from the popup palette of available colors.

Figure 17-4 shows modified data colors for the chart that you are currently working on.



SalePrice 🟦

Mileage 🟨

Figure 17-4. *Specifying data colors*

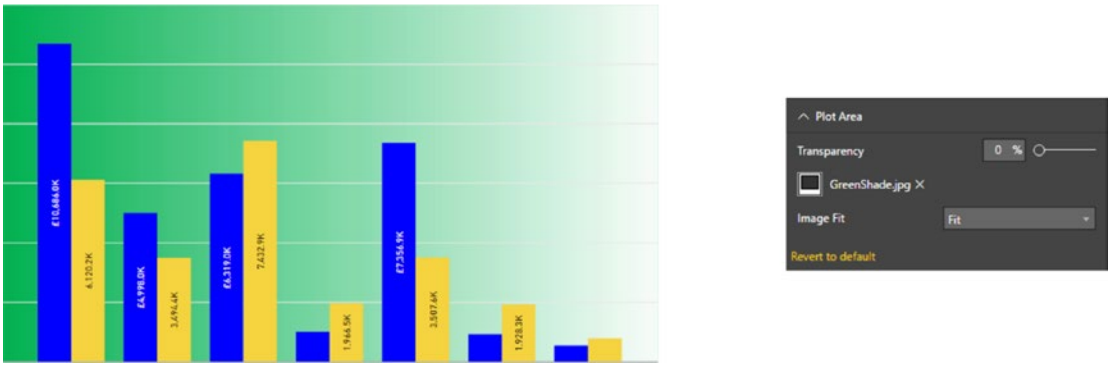
■ **Note** If a chart only contains a single data series (this is always the case for pie and donut charts, and can be the case for column, line, bar, and area charts), you have the option to set a color for *each data point* of a data series.

Plot Area

For all except pie, donut, and funnel charts, you can set an image as the background for the plot area of the selected chart. Here is an example of how to do this:

1. Select the chart whose plot area you want to alter (I am still using the chart that you first saw at the beginning of this chapter).
2. In the Visualizations pane, click the Format icon.
3. Expand the Plot Area section.
4. Click Add Image. The standard Windows Open dialog will appear.
5. Navigate to the image file that you want to add. I will use C:\PowerBiDesktopSamples\Images\GreenShade.png.
6. Click Open.

7. In the Image Fit popup, select Fit. The image will expand to fill all the plot area. You can see an example of an image applied to a chart in Figure 17-5.



Plot Area

Transparency 0 %

GreenShade.jpg ✕

Image Fit: Fit

Revert to default

Figure 17-5. *Adding an image to a chart's plot area*

■ **Note** To remove the image that you added, click the small cross to the right of the image file name in the Plot Area section of the Format area of the Visualizations pane.

Axis Modification

Most charts—except pie, donut, and funnel charts—allow you to tweak the axis attributes. This means modifying

- The X (horizontal) axis
- The Y (vertical) axis

We will take a quick look at these two separately.

Modifying the X Axis

There are several aspects of the X axis that you can alter:

- The axis display
- The axis title and presentation
- The font attributes of the axis labels
- The category width

A simple example explains how to adjust all of these aspects of the X axis:

1. Select the chart with an X axis that you wish to modify. I will continue with the chart that you first saw at the start of this chapter.
2. In the Visualizations pane, click the Format icon.
3. Expand the X-Axis section.

4. Ensure that the X Axis switch is set to On (this displays the axis and descriptive elements).
5. Select a font, font size, and color for the axis labels.
6. Set the Title to On (this adds the axis title).
7. Scroll down inside the Visualizations pane (if necessary) and select a font, font size, and color for the axis title.
8. Set a larger minimum category width (this is the width of each bar in the chart). You can see the Format area of the Visualizations pane for these attributes in Figure 17-6. The final, formatted, chart is shown in Figure 17-7.

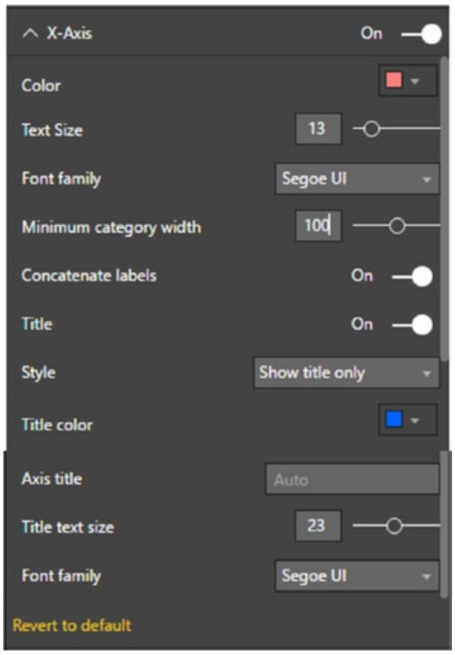

X-Axis On
Color
Text Size 13
Font family Segoe UI
Minimum category width 100
Concatenate labels On
Title On
Style Show title only
Title color
Axis title Auto
Title text size 23
Font family Segoe UI
Revert to default

Figure 17-6. *Formatting the X axis*

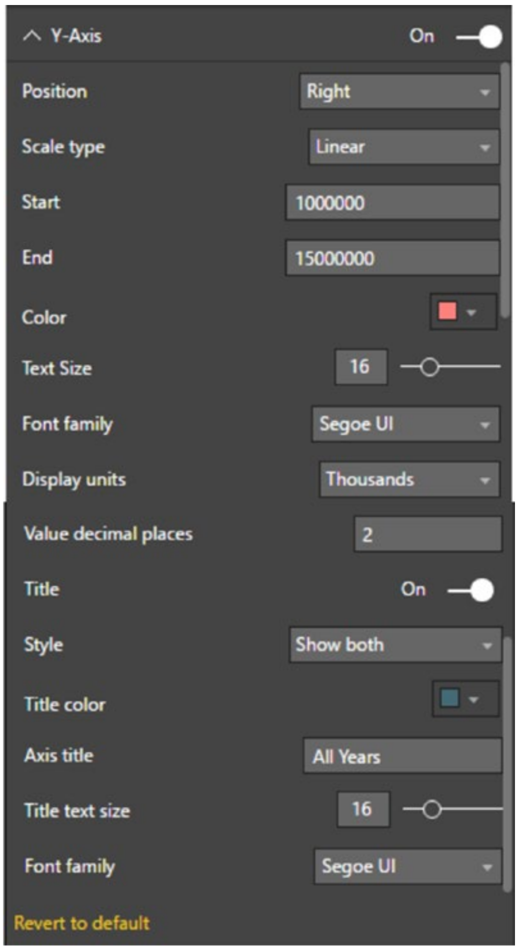
Modifying the Y Axis

If your chart has a Y axis (or in the case of a dual-axis chart, two Y axes), then you can modify the following elements:

- The axis position
- The axis scale type
- The lower axis value
- The upper axis value
- The title display
- The display units
- The numeric precision

Once again, it is probably easiest to see how these various formatting options can be applied using a single example. I continue using the chart that first saw at the start of this chapter.

1. Select the chart to which you wish to alter the axis attributes.
2. In the Visualizations pane, click the Format icon.
3. Expand the Y-Axis section.
4. From the Position popup, select Right.
5. Ensure that the Scale Type is set to Linear—or set to Log if you need a logarithmic scale.
6. Enter a Start value to set the starting figure for the axis. This can be negative, and must be a full, unformatted figure including thousands or millions.
7. Enter an End value to set the upper figure for the axis. This, too, must be a full, unformatted figure including thousands or millions.
8. Set the font, font size, and color for the Y-Axis elements.
9. Switch Title to On to display the Y axis title.
10. Set the Style to Show Both. This way the axis title will display both numbers and units.
11. Enter the number of decimals to be displayed on the axis in the Value Decimal Places field.
12. Enter a title in the Axis Title field.
13. Set the axis title font, font size, and color.
14. You can see the Format area of the Visualizations pane for the Y-Axis section in Figure 17-7. The final chart is shown in Figure 17-8.



Y-Axis On

Position	Right
Scale type	Linear
Start	1000000
End	15000000
Color	🟥
Text Size	16
Font family	Segoe UI
Display units	Thousands
Value decimal places	2
Title	On
Style	Show both
Title color	🟦
Axis title	All Years
Title text size	16
Font family	Segoe UI

Revert to default

Figure 17-7. *Formatting the Y axis*

If you are modifying a chart that only has a single axis, you can move the axis from the left side of the chart to the right by selecting Right from the Position popup menu. In the case of a chart with two axes, you can swap the axes around using this setting.

If your chart contains *two* Y axes (one on the left and one on the right), then you can repeat these settings for the other axis. You may have to scroll down in the Visualizations pane to see the settings for the second axis.

■ **Note** Remember that you can define a specific value as a measure in the data model. This can be extremely useful if you need to set reusable maximum and minimum values for the Y axis in charts.

Chart Borders

You can add a border to a chart just like you did to tables in the previous chapter. Simply click the Format icon and set the Border button to On. You can always choose a border color by expanding the Border section and selecting a color from the popup menu of available colors.

General Attributes

As was the case for the text-based visuals that you saw in the previous chapter, you can specify the exact position of a chart on the dashboard canvas in the General section of the Format area of the Visualizations pane.

Chart Aspect Ratio

As a final comment on chart formatting, it is worth remembering that you can lock the aspect ratio of a chart. This ensures that when you resize a chart, the width and height maintain their original proportions relative to each other.

To set the aspect ratio:

1. Click the chart to select it.
2. In the Visualizations pane, click the Format icon.
3. In the Lock Aspect section, click Off to turn the lock on, or alternatively click to the right of the empty circle indicating that the lock is off so that it slides to the right and becomes a filled circle.

Now, when you resize a table using the corner handles, it will keep its height-to-width ratio.

Finally, in Figure 17-8 you can see the chart with all your modifications applied. Before rolling your eyes at my (lack of) taste, please note that I am not suggesting that you should present charts in this way (indeed, you should probably take this as an example of what *not* to do). I am merely showing the result of applying many of the available chart formatting options.

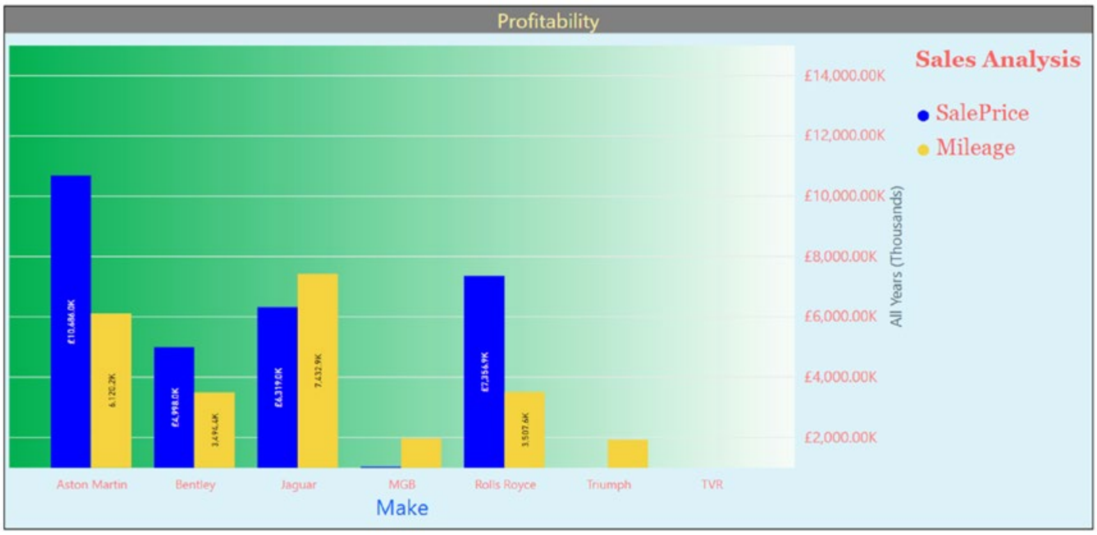


Figure 17-8. *A fully formatted chart*

Tooltips

To avoid overloading charts with too many data points, you may prefer to add useful data to the tooltip that appears when you hover the pointer over a data point. You can do this by adding extra fields to the Fields area of the Visualizations pane in this way:

1. Create a chart (I will make an area chart) based on the following fields:
   a. Axis: Color (from the Colors table).
   b. Values: SalePrice (from the InvoiceLines table).

2. Drag the following fields from the Stock table into the Tooltips well of the Fields area of the Visualizations pane:
   a. LaborCost
   b. SpareParts

The chart should look like the one that is shown in Figure 17-9. You can see that the data for the labor cost and spare parts is only visible in the tooltip for a single data point.

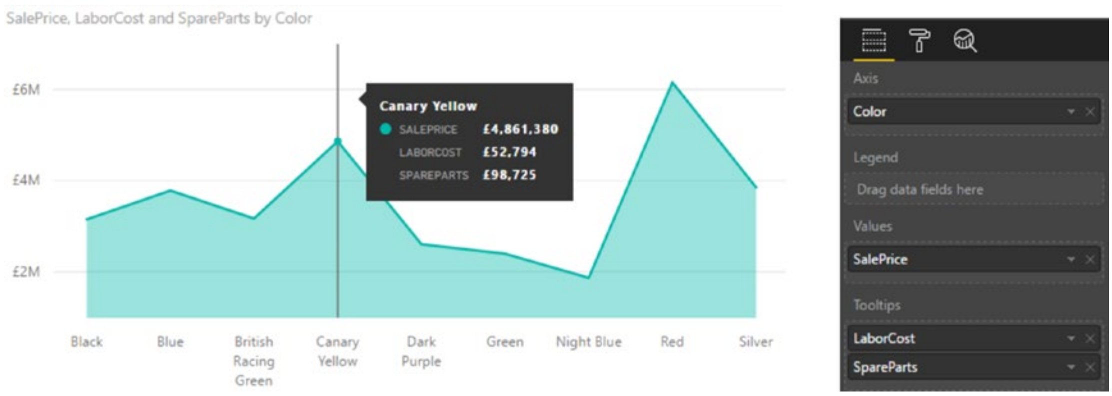


Figure 17-9. *Extending tooltips*

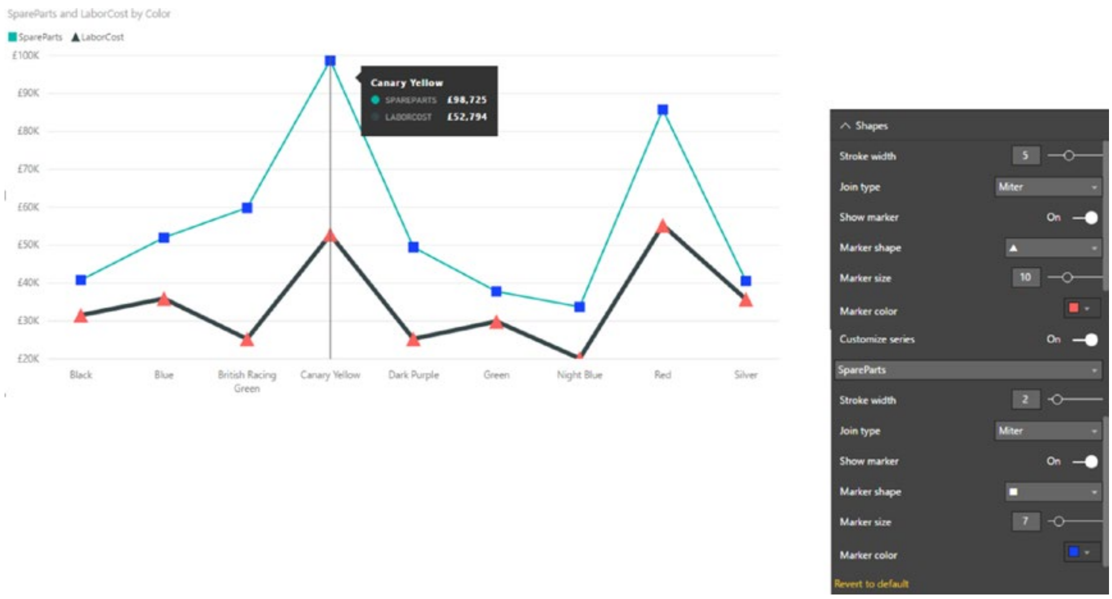
Specific Chart Formatting

Although most charts share many of the common formatting options that you have seen so far in this chapter, others present different formatting possibilities. This means that, while bar, stacked bar, column, and stacked column charts can all be formatted using the techniques that you have recently learned, other chart types require discovering a few new tweaks.

Line, Area, and Stacked Area Charts

One group of charts that all share a new formatting option are line charts, area charts, and stacked area charts. More specifically, they all allow you to adjust the line and point aspects of the chart. Here is an example to show you this in action:

1. Create a new line chart using the following fields:
    a. Color (from the Colors table)
    b. SpareParts (from the Stock table)
    c. LaborCost (from the Stock table)
2. In the Visualizations pane, click the Format icon.
3. Expand the Shapes section.
4. Set the Stroke Width to 5.
5. Set Show Marker to On.
6. Select a marker shape from the Marker Shape popup list.
7. Increase the marker size and change its color.
8. Set Customize Series to On.
9. Select SpareParts as the data series to modify from the popup list.
10. Set the Stroke Width to 2.
11. Set Show Marker to On.
12. Select a different marker shape from the Marker Shape popup list.
13. Decrease the marker size and change its color. You should see a chart like the one in Figure 17-10.



Shapes
Stroke width 5
Join type Miter
Show marker On
Marker shape ▲
Marker size 10
Marker color 🟥
Customize series On
SpareParts
Stroke width 2
Join type Miter
Show marker On
Marker shape ■
Marker size 7
Marker color 🟦
Revert to default

Figure 17-10. Shape options in a line chart

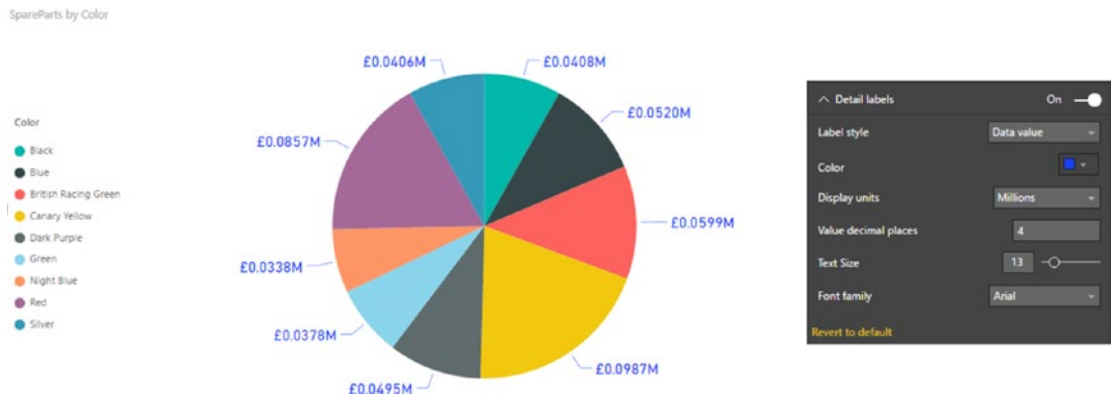
■ **Note** If you are not displaying the marker, you can select a join type (the intersection where each line alters direction) from the Join Type popup in the Shapes section.

Pie and Donut Charts

Pie and donut charts also have specific formatting possibilities. These consist of the ways that you can alter the detail labels for a pie or donut chart. Try out the following:

1. Create a new line chart using the following fields:
   a. Color (from the Colors table)
   b. SpareParts (from the Stock table)
2. In the Visualizations pane, click the Format icon.
3. Set the Legend to On and place it at the right in the center.
4. Expand the Detail Labels section.
5. Select Data Value from the Label Style popup list.
6. Choose a color from the color palette.
7. Enter, or select, a larger text size.
8. Select Millions as the display units from the Display Units popup list.

9. Enter **4** as the Value Decimal Places setting. You will have a chart like the one in Figure 17-11.



Detail labels On
Label style: Data value
Color
Display units: Millions
Value decimal places: 4
Text Size: 13
Font family: Arial
Revert to default

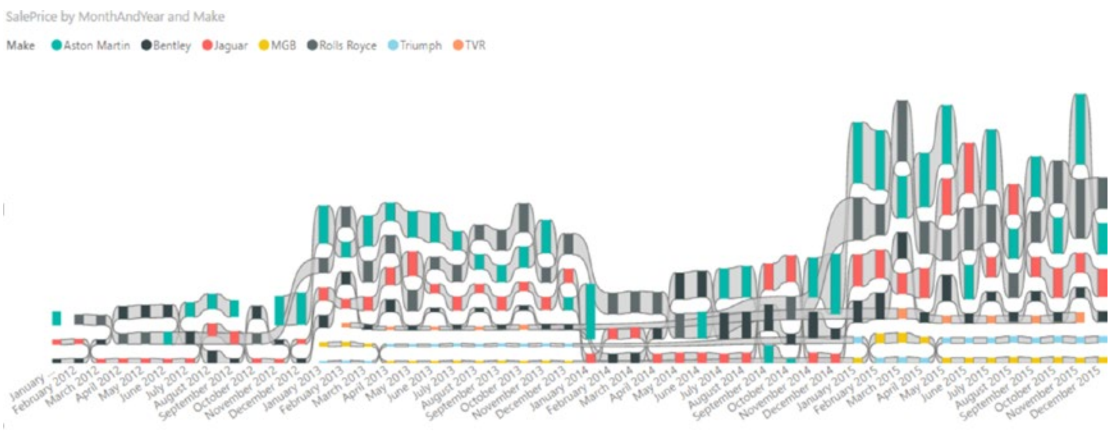
***Figure 17-11.** Shape options in a pie chart*

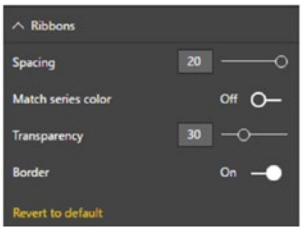
Ribbon Charts

Ribbon charts contain all the standard formatting options that you have seen applied to stacked column charts, so I will not reiterate all these possibilities here. There is, however, one formatting option that is specific to this chart type. This is the ribbon formatting.

To see this in action:

1. Re-create the ribbon chart that you made in Chapter 16.
2. Click on the Format icon in the Visualizations pane.
3. Expand the Ribbons section.
4. Set the Spacing to 20. This sets each ribbon apart from the others.
5. Set Match Series Color to Off. This sets the ribbon color to gray, rather than the color of the bar for each data point.
6. Set the Border to On. The result (which you can compare to Figure 16-24) is shown in Figure 17-12.





Spacing 20
Match series color Off
Transparency 30
Border On

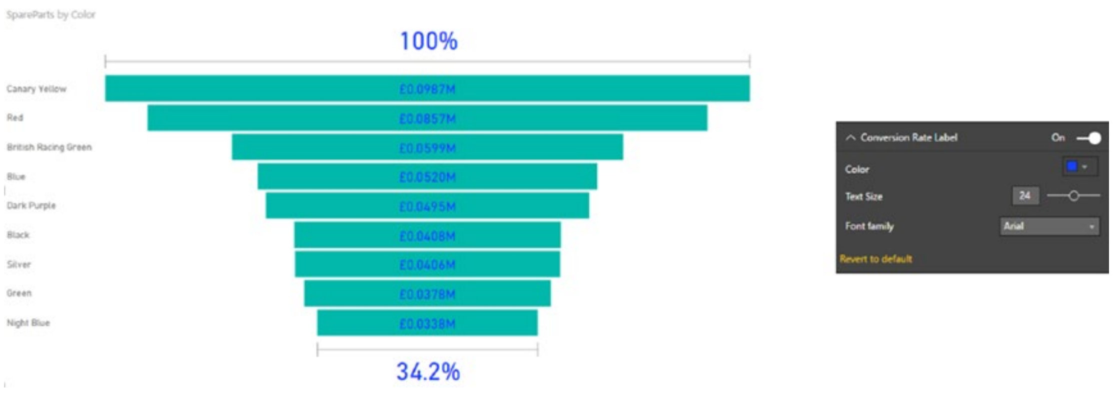
***Figure 17-12.** A ribbon chart*

Funnel Charts

Funnel charts allow you to display (or not) and format one element that is specific to this kind of chart—the conversion rate. These are the percentage figures that appear above and below the funnel. This short walk-through explains how you can do this:

1. Take the pie chart that you created in the previous section and click the Funnel Chart icon in the Visualizations pane.
2. Sort by SpareParts.
3. Expand the Conversion Rate section of the Format area of the Visualizations pane.
4. Choose a color from the color palette.

5. Enter, or select, a larger text size. You will have a chart like the one in Figure 17-13.



Conversion Rate Label On
Color
Text Size 24
Font family Arial

Figure 17-13. *Adjusting the conversion rate in a funnel chart*

■ **Note** To prevent the conversion rate from being displayed, all you have to do is click the Conversion Rate Label button.

Scatter and Bubble Charts

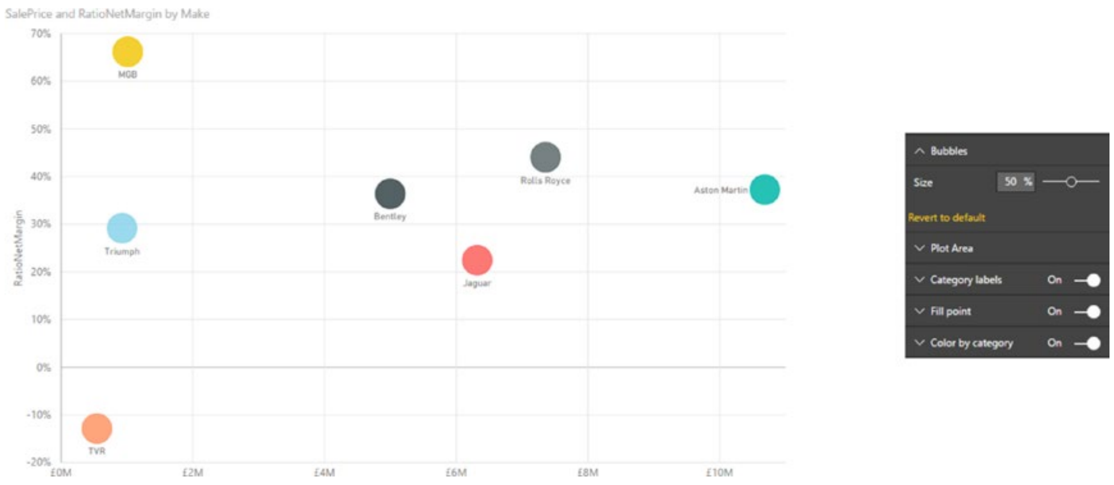
The final set of charts that allow you to apply some particular formatting are scatter and bubble charts. When you are creating these chart types, you can alter the following:

- Fill point
- Category labels
- Color by category
- Color border
- Bubbles

To see these effects, we will create a new chart and then try out some of the available options. We will start with a scatter chart that we will then extend to make into a bubble chart:

1. Create a scatter chart using the following data elements:
   a. Details: Make (from the Stock table)
   b. X Axis: SalePrice (from the InvoiceLines table)
   c. Y Axis: RatioNetMargin (from the InvoiceLines table)
2. Click the Format icon of the Visualizations pane.
3. Switch the Fill Point button to On.
4. Set Color by Category also to On.

5. Expand the Bubbles section (even if this is still a scatter chart) and set the Size to 50%. The chart will now look like the one shown in Figure 17-14.



Bubbles
Size 50 %
Revert to default
Plot Area
Category labels On
Fill point On
Color by category On

Figure 17-14. *A formatted scatter chart*

6. Add the following two data elements to the Fields area of the Visualizations pane:

   a. Size: GrossMargin (from the InvoiceLines table)

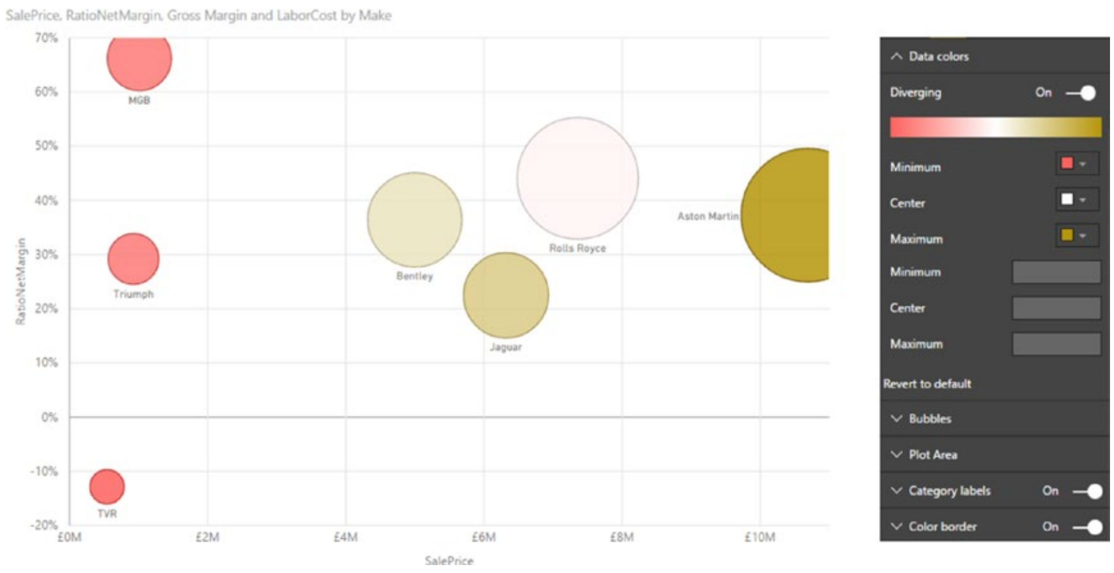
   b. ColorSaturation: Mileage (from the Stock table)

7. In the Format area of the Visualizations pane, reduce the bubble size.

8. Set Color Border to On.

9. Expand the Data Colors section.

10. Choose three different colors for the Minimum, Center, and Maximum. The chart will now look like the one in Figure 17-15.



Data colors
Diverging On
Minimum 🟥
Center ⬜
Maximum 🟨
Minimum
Center
Maximum
Revert to default
Bubbles
Plot Area
Category labels On
Color border On

Figure 17-15. *Formatting a bubble chart*

There are a couple of points that need to be made at this juncture:

- Adding a field to express the color saturation will deactivate the Color by Category formatting option, and will prevent you from attributing specific data colors.
- For a bubble chart, you can specify the actual values used to define the color divergence in the bubbles in the Data Colors section (assuming that a value for color saturation has been added).

Bubble Chart Play Axis

So far in this chapter, you have seen various ways of presenting data as charts, and how to select and compare the data using a variety of chart types. A final trick with Power BI Desktop (one that can be extremely effective at riveting your audience) is to apply a play axis to the visualization. This animates the chart and, ideally, is suited to showing how data evolves over time. Unfortunately, it is harder to get the "wow" effect using these printed pages, so this really is a technique that you have to try yourself.

You need to know that a play axis can only be applied to scatter or bubble charts. Similarly, adding a play axis will not suit or enhance all types of data. However, if you have a time-dependent element that can be added to your chart as the Y axis (such as sales to date, for instance), then you can produce some powerful and revelatory effects.

The following explains how to create a bubble chart that shows the net margin ratio for colors of car sold against the sales for the year to date:

1. Open the file C:\PowerBiDesktopSamples\CH17\CarSalesDataForReports.pbix and delete any existing visuals.
2. Create a bubble chart with the following fields using the following data:
    a. Legend: Make (from the Stock table)
    b. X Axis: YearSales (from the Invoices table)
    c. Y Axis: RatioNetMargin (from the InvoiceLines table)
    d. Size: CostPrice (from the Stock table)
    e. Play Axis: MonthAndYearAbbr (from the DateDimension table)
3. Adjust the presentation (size, legend placement, data labels, etc.) to obtain the best effect.

The visual will look like that shown in Figure 17-16.

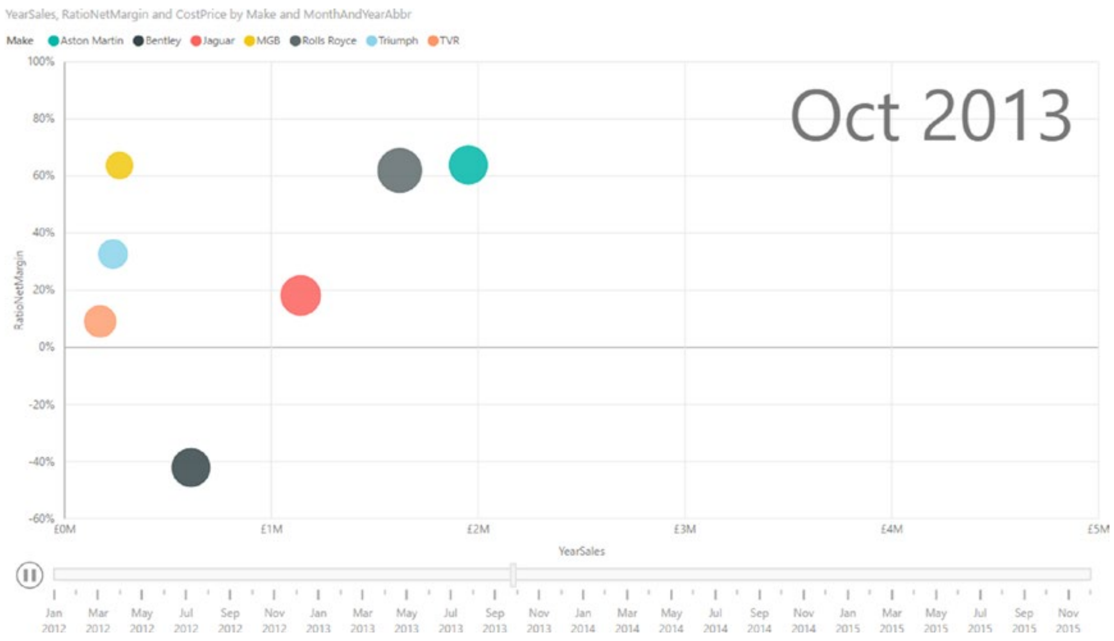


Figure 17-16. *The play axis on a bubble chart*

Click the Play icon to the left of the play axis. You will see the bubbles reveal how sales progress throughout the year.

There are a few points worth noting about the play axis while we are discussing it:

- You can pause the automated display by clicking the Pause icon, which the Play icon has become, while the animation is progressing. You can stop and start as often as you like.
- You can click any month (or any element) in the play axis to display the data just for that element, without playing the data before that point. This essentially means that you can use the play axis as a filter for your data.
- A play axis need not be time-based. However, it can be harder to see any coherence or progression in the data if time is not used as a basis for a play axis.
- Using cumulative data (such as the YearSales figure in this example, which is the cumulative year-to-date sales) is particularly effective with a time-based play axis as it lets you appreciate the growth of sales for each data item over time.
- You can use a play axis as another interactive filter for your data, but doing this makes you miss out on a fabulous animation technique!

Chart Analytics

Yet another way to add extra emphasis to the data in a chart is to add analytical enhancements. These are added elements that draw the reader's attention to a value that is calculated automatically. All these items are added as lines to a chart independently of the existing data. They include

- *Constants*: You may wish to add a specific value as an item in a chart.
- *Minimum values*: This allows you to draw attention to a lower threshold.
- *Maximum values*: This allows you to draw attention to an upper threshold.
- *Averages*: You can have Power BI Desktop calculate the average of a set of values and display this as a line.
- *Median values*: Power BI Desktop can also calculate and display a median value.
- *Percentiles*: You can specify a percentile to be displayed.
- *Weighted average*: You can specify a weighted average to be displayed.

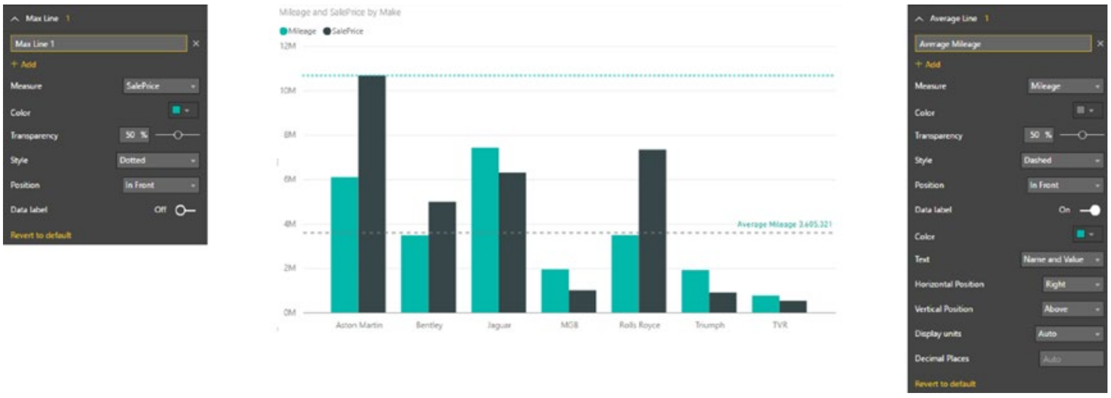
To see this a little closer, suppose that you want to add the following lines to a chart containing the selling price and mileage of all cars by make:

- The average mileage
- The maximum selling price

Here is how you can do this:

1. Click the Clustered Column Chart icon in the Visualizations pane. An empty chart visual will be created on the dashboard canvas.
2. Drag the following fields into the Axis area of the field well in this order:
    a. Make (from the Stock table)
    b. Model (from the Stock table)
    c. Color (from the Colors table)

3. Drag the following fields into the Value area of the field well:
    a. SalePrice (from the InvoiceLines table)
    b. Mileage (from the Stock table)
4. Click the Analytics icon (to the right of the Format icon in the Visualizations pane).
5. Expand the Average Line section.
6. Click Add.
7. Select Mileage from the Measure popup list.
8. Choose a color from the palette of available colors.
9. Set the Style as Dashed.
10. Click the Data Label button to activate the data label.
11. Set the Text to Name and Value.
12. Set the Horizontal Position to Right.
13. Set the Vertical Position to Above.
14. Double-click in the field containing the text "Average Line 1" and enter **Average Mileage**.
15. Close the Average Line section.
16. Expand the Max Line section.
17. Click Add.
18. Select SalePrice from the Measure popup list.
19. Set the Style to Dotted. The chart should look like the one shown in Figure 17-17.



Max Line 1

Max Line 1

+ Add

Measure	SalePrice
Color	🟩
Transparency	50 %
Style	Dotted
Position	In Front
Data label	Off

Revert to default

Average Line 1

Average Mileage

+ Add

Measure	Mileage
Color	⬛
Transparency	50 %
Style	Dashed
Position	In Front
Data label	On
Color	🟩
Text	Name and Value
Horizontal Position	Right
Vertical Position	Above
Display units	Auto
Decimal Places	Auto

Revert to default

Figure 17-17. *Adding chart analytics*

It is worth noting that you can add several lines for an analytical element. That is, you can add as many averages, for instance, as there are data elements in the chart. To do this, simply click Add once more in the section where you wish to add another line and then define the settings to achieve the result that you want. This is, nonetheless, a technique that can detract from the visibility of a chart as easily as it can add to the data story that you are telling. So it is probably best used sparingly.

If you are adding a percentile line, then you have one extra element that you will probably need to alter. This is the actual percentile value of the maximum value of a data element, which you can alter using the Percentile slider.

Removing an analytical line is as simple as this:

1. Click the Analytics icon.
2. Expand the section corresponding to the line that you wish to remove.
3. Click the cross to the right of the line that you wish to remove.

■ **Note** You cannot actually select an analytical line in a chart.

Scatter Chart Symmetry Shading and Ratio Line

Two new features of Power BI desktop can help you "see the wood for the trees" in scatter charts:

- Symmetry shading
- Ratio lines

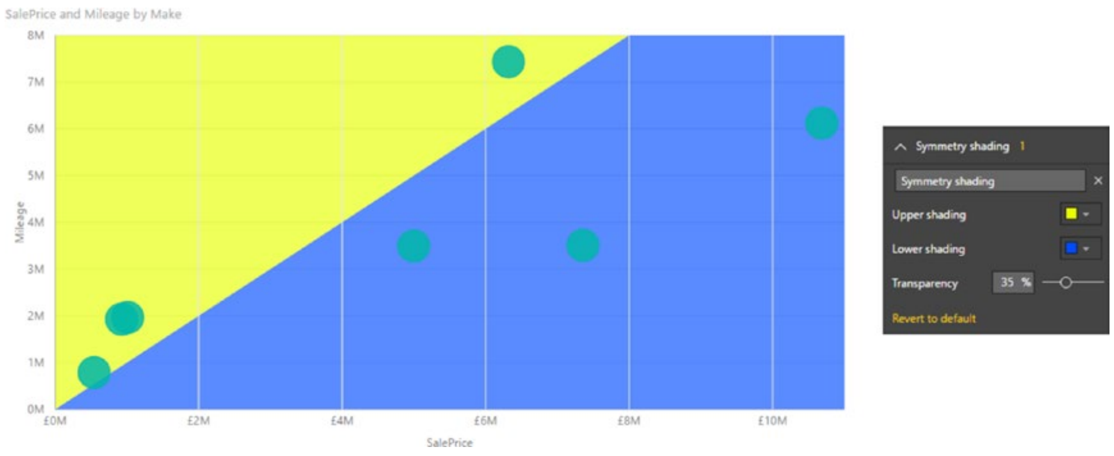
They are equally simple to apply, but can really help you to see beyond a mass of data points and isolate valuable trends and salient items of data.

Symmetry Shading

This feature allows you to see which points have a higher value on the X axis compared to the Y axis (and vice versa). Like so much of Power BI Desktop, it is best understood using a practical example:

1. Create a scatter chart using the following elements:
    a. Details: Make (from the Stock table)
    b. X Axis: SalePrice (from the InvoiceLines table)
    c. Y Axis: Mileage: (from the Stock table)
2. With the scatter chart selected, click the Analytics icon (the magnifying glass) in the Visualizations pane.
3. Expand the Symmetry Shading section.
4. Click Add.
5. Select a color for the upper shading from the palette of available colors.
6. Select a color for the lower shading from the palette of available colors.

7. Adjust the Transparency slider to set the transparency to 35%. The chart will look like the one shown in Figure 17-18.



Symmetry shading 1
Symmetry shading ×
Upper shading
Lower shading
Transparency 35 %
Revert to default

Figure 17-18. *Adding symmetry shading to a scatter chart*

You can remove symmetry shading by clicking the small cross at the right of the Symmetry Shading box in the Symmetry Shading section.

■ **Note** Symmetry shading will only really be of use if the values on the X and Y axes are comparable in extent.

Ratio Line

One final instant analysis function is the ratio line. This shows you how to apply it:

1. Follow steps 1 and 2 of the previous section to create a scatter chart.
2. Expand the Ratio Line section.
3. Click Add.
4. Select a color for the ratio line from the palette of available colors.

5. Select a line type from the popup list of line styles. The chart will look like the one shown in Figure 17-19.

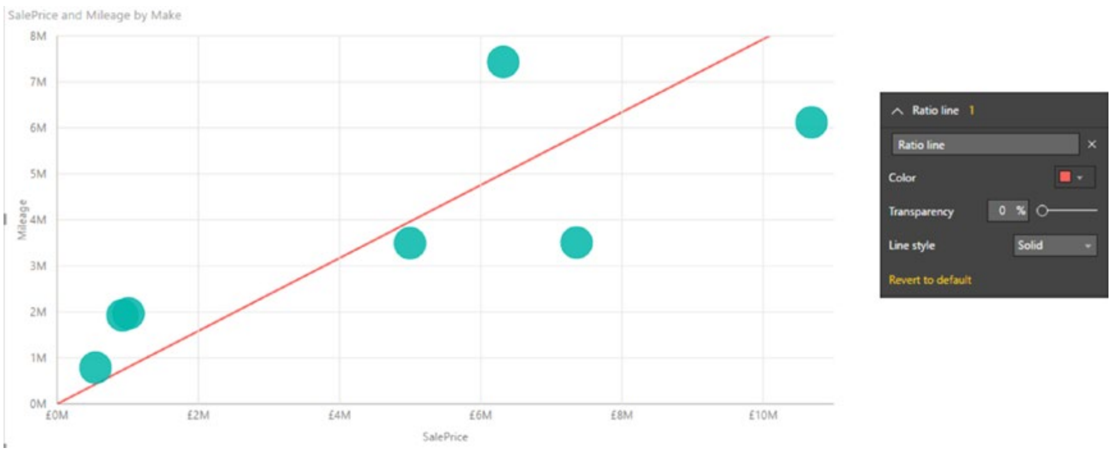


Figure 17-19. *Adding a ratio line to a scatter chart*

AS was the case for symmetry shading, you can remove a ratio line by clicking the small cross at the right of the Ratio Line box in the Ratio Line section.

Conclusion

In this chapter you learned how to enhance Power BI Desktop chart types by formatting them for added effect. This included modifying the colors of data series and data points as well as adding or removing titles. You also saw how to add legends and data labels to certain types of chart.

You then saw how to enhance charts with automatic analytics. These included averages and selected maximum values as well as ratio lines.

As well, you saw how certain types of chart (bubble charts specifically) can even be animated so that you can add time as a descriptive factor to help your audience understand how data evolves.

You are now well on the way to becoming a Power BI Desktop dashboard maestro. All you need to look at now are a few remaining visualization types and you will have attained a complete mastery of high-impact presentations. So now, on to the next two chapters to finish your apprenticeship.

CHAPTER 18

Other Types of Visuals

While text-based visuals and charts can often make your point, there are times when you need to deliver your insights in ways that go beyond the more traditional data displays. This is where Power BI Desktop really comes to your aid. With the right data—and only a few clicks—you can revitalize your dashboards with

- Tree maps
- Gauges
- KPIs
- R visuals

Most of these visualization types are as simple to create as the text-based visuals and charts that you saw in the preceding four chapters. Yet their very ease of use should not distract you from the clarity and power that they can bring to your reports and presentations.

Yet this is only the starting point. As a freely extensible platform, Power BI Desktop also hosts a wide and growing variety of visuals developed by both Microsoft and third parties. This gallery of visual extensions is continually growing and incredibly easy to access and use. The final part of this chapter shows you some of the current visuals and how to find, add, and use them in your Power BI Desktop dashboards. This chapter will use the file C:\PowerBiDesktopSamples\CH18\CarSalesDataForReports.pbix as the basis for the visuals that you will create.

Tree Maps

One type of visual that can really help your audience to see the way that individual values relate to a total is the tree map. While this is not a map in any geographical sense, it can certainly assist users in finding their way around a set of figures.

As, yet again, seeing is believing in the world of visuals, let's take a look at a tree map showing how the labor costs stack up for each make and model of car purchased:

1. Open the C:\PowerBiDesktopSamples\CH18\CarSalesDataForReports.pbix file, or click an empty part of the dashboard canvas.
2. Click the Tree Map icon in the Visualizations pane. A blank tree map will appear.
3. Expand the Stock table and drag the Make field on to the Group box in the field well.
4. Drag the Model field on to the Details box in the field well.

A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_18

5. Drag the LaborCost field on to the Values box in the field well. The tree map should look like the one in Figure 18-1.

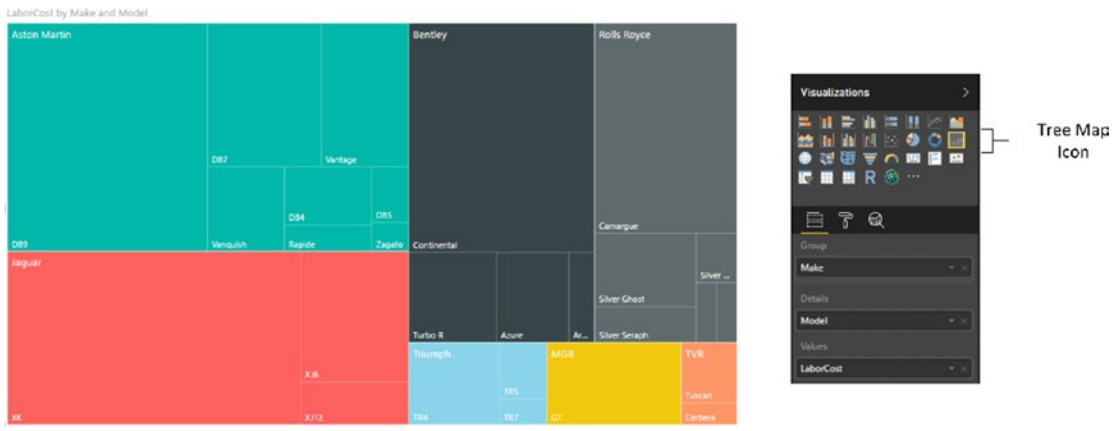


Figure 18-1. *A tree map showing labor cost for each make of car*

As you can see in Figure 18-1, the tree map has grouped each model of car by make, and then displayed the labor cost as the relative size of each box in the tree map. This way, your viewers can get a rough idea of

- How the labor cost for each model compares to the total labor cost for the make.
- How the labor cost for each make compares to the total labor cost.

As you are probably expecting by now, you can hover the mouse over any box in the tree map and see a popup tooltip of the exact data.

Power BI Desktop will decide how best to organize the tree map. However, the final appearance will depend on how wide and how tall the visual is. So don't hesitate to resize this particular kind of visualization and see how it changes as you adjust the relative height and width.

There could be cases when a tree map cannot display all the available data, possibly because there are too many data points, or the range of values is too wide to be shown properly, or (as the case with the pie charts) because there are negative values in the data. In these circumstances, the tree map shows a warning triangle in the top left of the visual. If you hover the mouse pointer over this icon, you will see a message like the one shown in Figure 18-2.

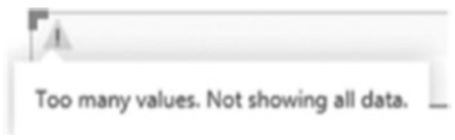

Too many values. Not showing all data.

Figure 18-2. *The tree map data alert*

If you see this warning, then you could be advised to apply filters to the data (as described in Chapter 20) to filter out the data that is causing the problem.

Drill Down and Tree Maps

The same drill-down and expand lower-level logic that you have already seen for matrices, charts, and maps also applies to tree maps. As you are used to the concept by now, I will only show a short example of drill down using a tree map:

1. Select the tree map that you just created.
2. Drag the Color field from the Colors table to the Group well of the Visualizations pane. Place this *under* the Make field.
3. Activate Drill Mode by clicking the drill mode icon at the top right of the visual.
4. Click any of the elements for Aston Martin. You will drill down to the next level (color) and the visual will look like the one shown in Figure 18-3.

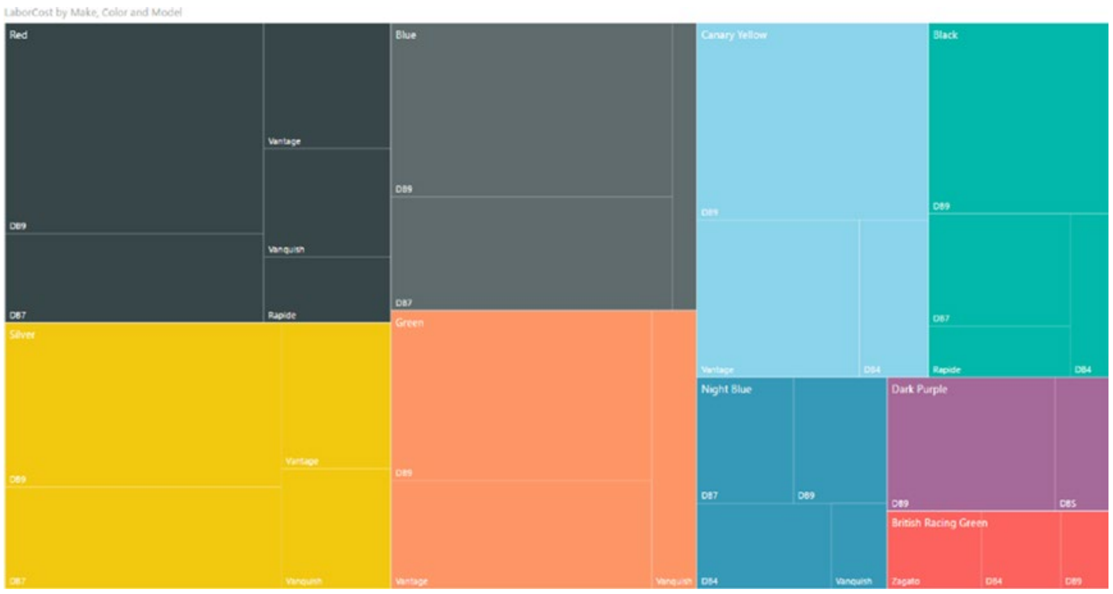


Figure 18-3. *Drilling down into a tree map visual*

Formatting Tree Maps

Tree maps can be formatted just like any other visual. The following are the available options in the Format icon in the Visualizations pane:

- Legend
- Data Colors
- Data Labels
- Category Labels
- Title
- Background
- Lock Aspect
- Border

Since applying all of these options was explained (for other types of visual) in the four previous chapters, I will not explain them in detail again, but I will provide a few comments about the available options.

Legend

In my opinion, a legend is superfluous as it duplicates the information about the grouping data that is already displayed in the tree map. You may find a legend necessary if you choose not to display the category labels, however.

Data Colors

This palette lets you choose a specific color for each grouping element.

Data Labels

You can display (or hide) the actual figures that are behind each segment of a tree map if you wish.
This entails switching the Data Labels button to On or Off. If you expand the Data Labels section of the Formatting pane, you can also modify the color, units, number of decimals, and text size and font of the data labels.

Category Labels

This option lets you decide whether or not to display the grouping elements and the detail element information inside the tree map. If your tree map contains dozens (or hundreds) of data points, then you might want to hide the category labels. If you expand the Category Labels section of the Formatting pane you can also modify the color, font, and size of the category labels.

Title

As is the case with most visuals, you can choose to show or hide the title, as well as alter its text, font size, and background color.

Background

You can apply a colored background to the kind of visuals that you can see in this chapter, too.

Lock Aspect

If you set Lock Aspect to On, then you can resize the visual while keeping it proportionally sized.

Border

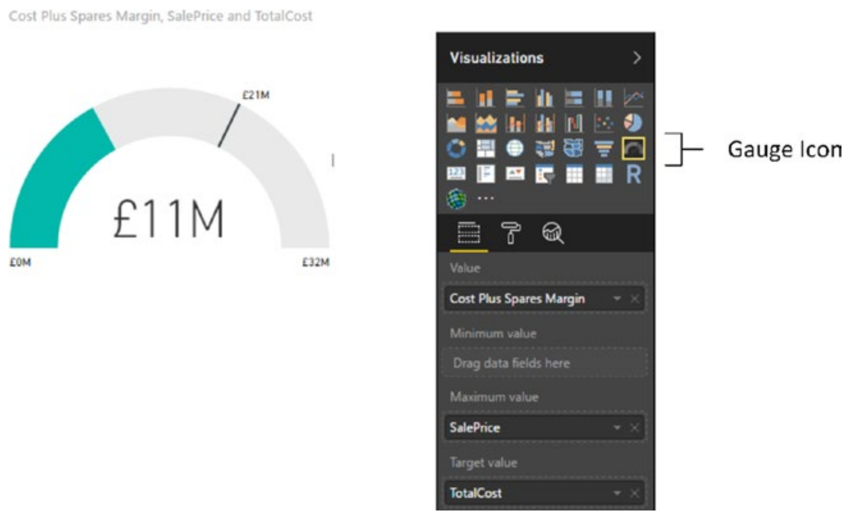
AS is the case with just about any visual, you can add or remove a border to tree maps, too.

Gauges

After nearly five chapters of delving into the range of visuals that Power BI Desktop has to offer, there is one final built-in visual to look at. This is the Power BI dashboard gauge. These kinds of visuals are particularly effective when you want to compare actuals to targets or see how a metric compares to a key performance indicator (KPI), for instance. In this example, you will use a gauge to see how the total cost of purchases and spares compares to the threshold set by the finance director:

1. Open the C:\PowerBiDesktopSamples\CH18\CarSalesDataForReports.pbix file, or click an empty part of the dashboard canvas.
2. Click the Gauge icon in the Visualizations pane. A blank gauge visual will appear.
3. Expand the InvoiceLines table and drag the Cost Plus Spares Margin field into the Value box in the field well. Alternatively, drag this field directly onto the gauge that you just created.
4. Drag the SalePrice field into the Maximum Value box in the field well.
5. Drag the TotalCost field from the Stock table into the Target Value box in the field well.

The gauge will look like Figure 18-4.



Value
Cost Plus Spares Margin
Minimum value
Drag data fields here
Maximum value
SalePrice
Target value
TotalCost

Figure 18-4. *A gauge in Power BI Desktop*

Gauges are designed to show progress against a target or a total. Consequently, you can set the elements described in Table 18-1 for a gauge.

Table 18-1. *The Specific Data Requirements for Gauges*

Element	Description
Value	The element that appears as a colored bar in the gauge.
Minimum Value	The minimum value to alter the perceptual effect of the gauge.
Maximum Value	The maximum value to alter the perceptual effect of the gauge.
Target Value	A metric that is defined as something to be attained.

In fact, all that a gauge needs is a value. It will work without any of the other three elements. However, the real visual power of a gauge comes from the way that it uses the viewer's presumption of a target or value to be achieved. Consequently, it is usually best to apply a target at least, and a maximum value to convey an idea of success or failure for the reader.

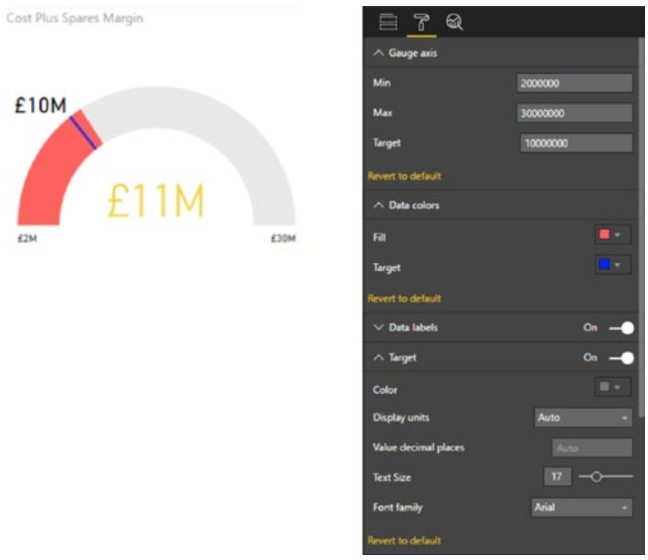
■ **Note** Remember that you can define a specific value as a measure in the data model. This can be extremely useful if you need to set the maximum, minimum, or target values for gauges. This way you can modify a threshold once without having to change the attributes of multiple visuals independently.

Formatting Gauges

Many of the gauge formatting options are highly specific to gauge visuals. So here is a quick trip through the available presentation techniques that you need to know to enhance this type of visual:

1. Click the gauge that you created previously.
2. In the Visualizations pane, remove the Target value (the CostPlusSpares field).
3. Remove the Maximum value (the SalePrice field).
4. In the Visualizations pane, click the Format icon.
5. Expand the Gauge Axis section.
6. Enter **2000000** as the minimum value.
7. Enter **30000000** as the maximum value.
8. Enter **10000000** as the target.
9. Expand the Data Labels section.
10. Ensure that the Data Labels switch is set to On.
11. Choose a color from the popup palette for colors for the data label fill.
12. Choose a color from the popup palette for colors for the target.
13. Expand the Callout Value section.

14. Ensure that the Callout Value switch is set to On.
15. Choose a color from the popup palette for colors for the callout value.
16. Expand the Data Colors section and select a color for the fill and the target.
17. Expand the Target section and increase the text size. The gauge will look something like Figure 18-5.



Gauge axis
Min 2000000
Max 30000000
Target 10000000
Revert to default

Data colors
Fill 🟥
Target 🟦
Revert to default

Data labels On

Target On
Color
Display units Auto
Value decimal places Auto
Text Size 17
Font family Arial
Revert to default

Figure 18-5. *A formatted gauge*

If you wish, you can also alter the background data labels, border, and title just as you did previously for other visuals. You can also choose not to display the data labels or the callout value (the value of the gauge) if you prefer. All you have to do is to set their respective switches to the Off position.

■ **Note** When entering figures in the Formatting pane (referenced earlier in the chapter) of the Visualizations pane, you must not add any numeric formatting, such as thousands separators. If you do, the formatting will not work and Power BI will wait until you have entered the numbers in their raw form.

Gauges are a simple yet effective visual. A set of gauges can also be a truly effective way of displaying high-level key metrics on a dashboard.

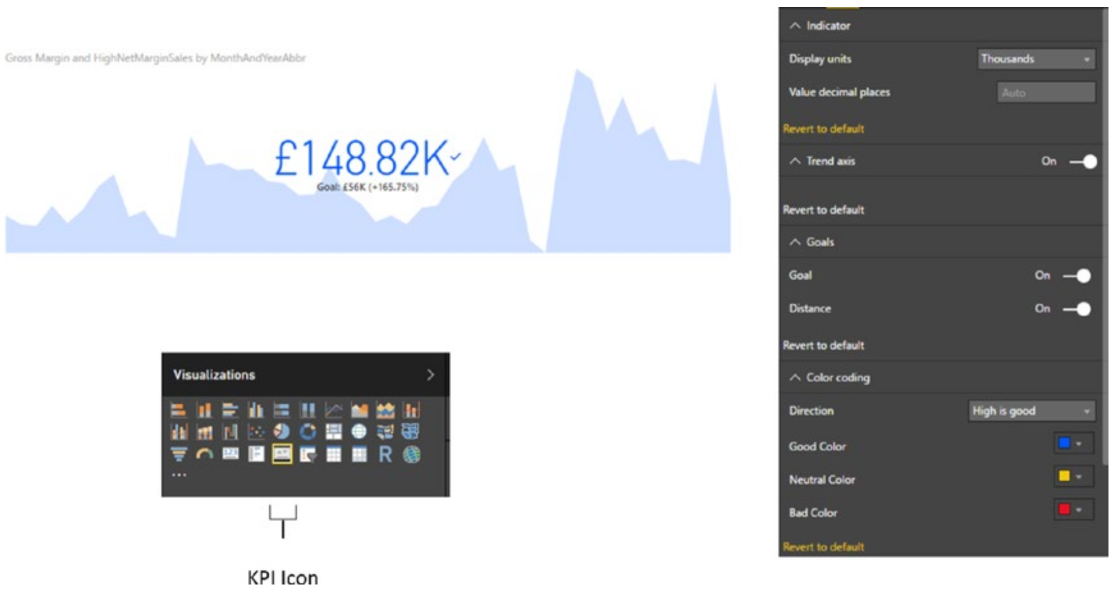
The remaining gauge formatting options are common to most of the visuals that you have already seen:

- Title
- Background
- Lock Aspect
- Border

KPIs

Power BI Desktop can also create key performance indicator visuals that illustrate how a data element is trending compared to a target value. Here is one example of this:

1. Create a clustered column chart visual.
2. Add the SalesPrice field (from the InvoiceLines table) as well as the MonthAndYear field (from the DateDimension table).
3. Click the popup menu at the top right of the chart and select "Sort the chart by month and year."
4. Click the KPI icon to convert the chart to a KPI. You can see this in Figure 18-6.
5. Add the HighNetMarginSales field from the InvoiceLines table to the Target Goals well of the Visualizations pane.
6. Click the Format icon in the Visualizations pane.
7. Expand the Indicator section and select Thousands as the display unit.
8. Expand Color Coding and select another color for the Good Color. The KPI will look like the one shown in Figure 18-6.



Indicator
Display units: Thousands
Value decimal places: Auto
Revert to default
Trend axis: On
Revert to default
Goals
Goal: On
Distance: On
Revert to default
Color coding
Direction: High is good
Good Color: 🟦
Neutral Color: 🟨
Bad Color: 🟥
Revert to default

Figure 18-6. *A KPI visual*

■ **Note** You cannot currently sort a KPI visual. So you need to be sure that you have sorted the chart visual *before* you convert it to a KPI.

R Visuals

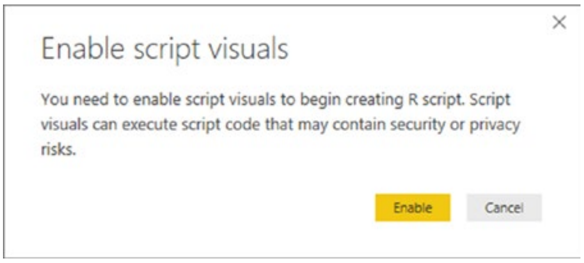
Should you find that even the wide range of built-in visuals that Power BI Desktop has to offer are just not enough to make your point, then you can take visuals to another, higher level using the R language. "R" as it is known is a language that is used to analyze data and create visualizations of statistical data. It can be used inside Power BI Desktop reports and dashboards to extend the analysis that you are delivering using the data that you have prepared for your existing visuals.

R is not an intuitive language, but it is extremely powerful. So here is an example of how you can add an R visual to a dashboard using the CarSales data that you have already loaded:

■ **Note** Creating R visuals implies installing an R engine on the PC that hosts Power BI Desktop. You can find one of the available R downloads at `https://mran.revolutionanalytics.com/download/`.

---

1. Open the Power BI Desktop file C:\PowerBiDesktopSamples\CH18\CarSalesDataForReports.pbix.
2. Click the R visuals icon in the Visualizations pane, the Enable Script Visuals dialog will be displayed, like the one shown in Figure 18-7.


Enable script visuals

You need to enable script visuals to begin creating R script. Script visuals can execute script code that may contain security or privacy risks.

Figure 18-7. *The Enable Script Visuals dialog*

3. Click Enable. An empty R visual will be added to the desktop canvas and a pane containing the R script editor will appear under the report. You can see this in Figure 18-8.

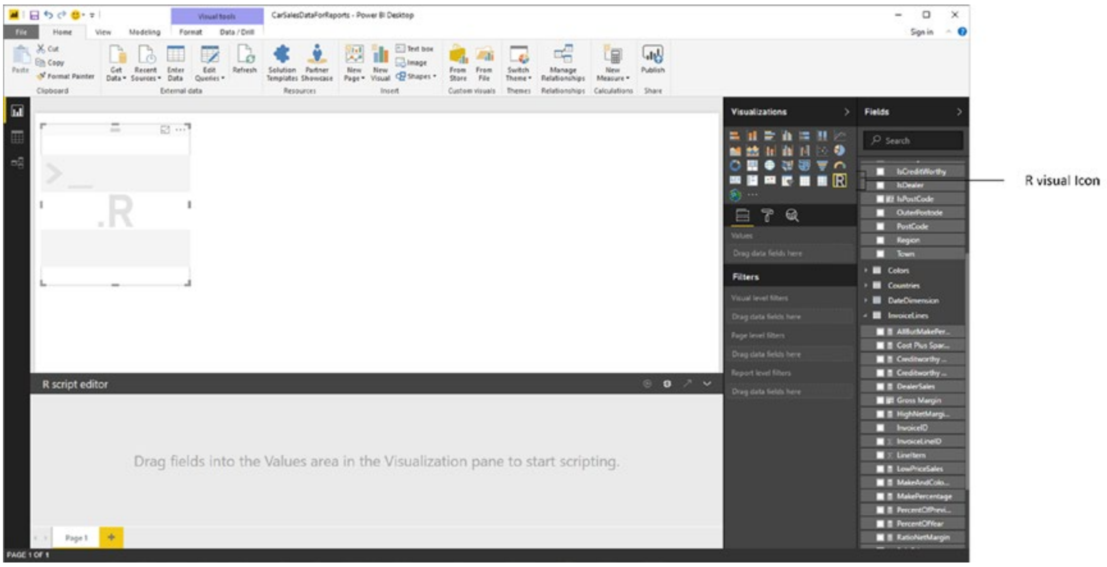

R visual Icon
R script editor
Drag fields into the Values area in the Visualization pane to start scripting.
Visualizations
Fields
Filters

Figure 18-8. Adding an R visual

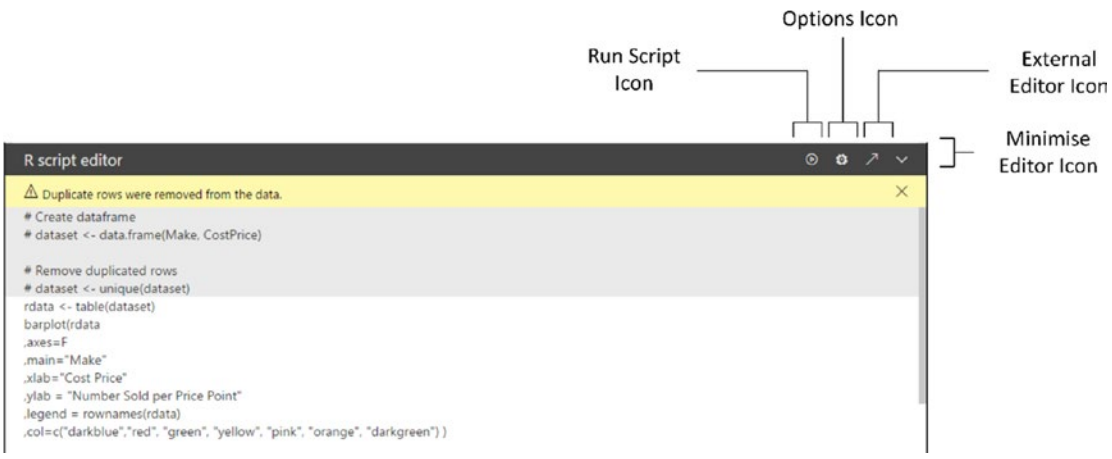
4. Drag the following fields onto the R visual (both are from the Stock table):

   a. Make

   b. CostPrice

5. Add the following script under the header lines in the R script editor (this script is available in the file C:\PowerBiDesktopSamples\CH18\Rscript.txt). Once you have done this the script editor will look like Figure 18-9.

```
rdata <- table(dataset)
barplot(rdata
,axes=F
,main="Make"
,xlab="Cost Price"
,ylab = "Number Sold per Price Point"
,legend = rownames(rdata)
,col=c("darkblue","red", "green", "yellow", "pink", "orange", "darkgreen") )
```



```
# Create dataframe
# dataset <- data.frame(Make, CostPrice)

# Remove duplicated rows
# dataset <- unique(dataset)
rdata <- table(dataset)
barplot(rdata
,axes=F
,main="Make"
,xlab="Cost Price"
,ylab = "Number Sold per Price Point"
,legend = rownames(rdata)
,col=c("darkblue","red", "green", "yellow", "pink", "orange", "darkgreen") )
```

Figure 18-9. *The R script editor with a functioning script*

6. Click the Run Script icon at the top right of the script editor.
7. The R visual will appear in Power BI Desktop.
8. Resize the R visual as necessary. The visual could end up looking like the one shown in Figure 18-10.

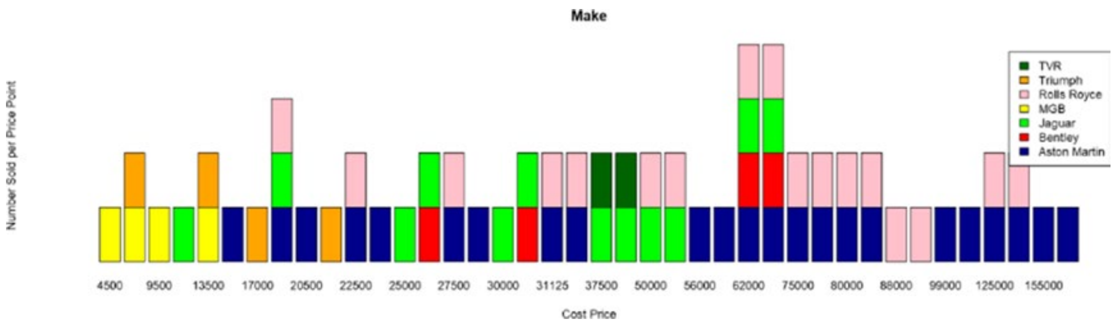


Figure 18-10. *An R visual*

You can, of course, only create R visuals if you are prepared to learn the R language. As this is the subject of entire tomes, I will not be describing the language here. Instead I suggest that you refer to the many excellent resources that are already available on this subject.

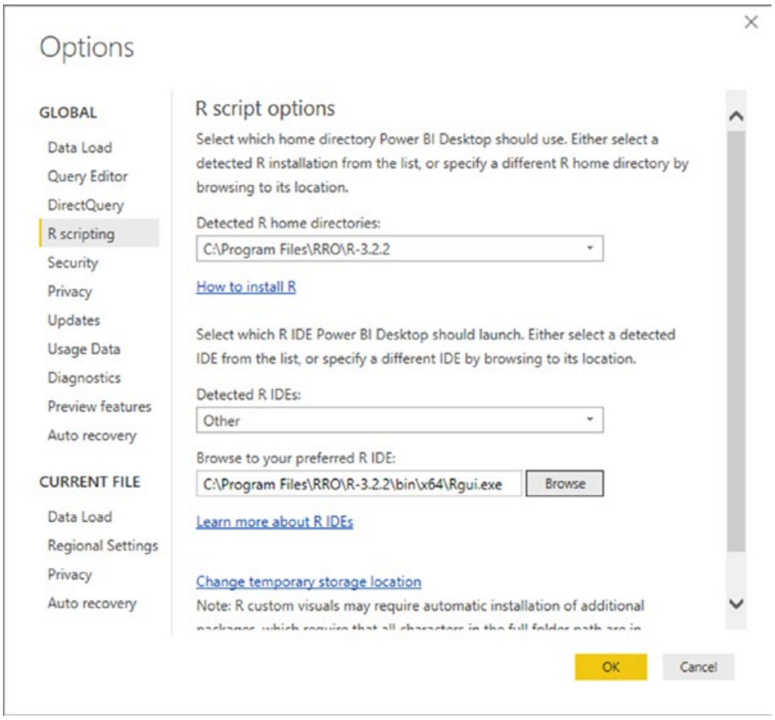
The R script editor icons are described in Table 18-2.

***Table 18-2.** The R Script Editor Icons*

Icons	Description
Run Script	Runs-or reruns-the R script and re-creates the R visual in Power BI Desktop.
R Script Options	Displays the dialog with available R script options.
Edit script in external R IDE	Runs an external R editor, using the script from the Power BI Desktop R visual.
Minimize the script pane	Hides (or redisplays) the R script pane.

R Options

There are a couple of R options that you can adjust if this proves really necessary. You can find these by choosing File ➤ Options and Settings ➤ Options and then clicking R scripting in the left-hand pane of the Options dialog that you can see in Figure 18-11.



R script options

Select which home directory Power BI Desktop should use. Either select a detected R installation from the list, or specify a different R home directory by browsing to its location.

Detected R home directories:
C:\Program Files\RRO\R-3.2.2

How to install R

Select which R IDE Power BI Desktop should launch. Either select a detected IDE from the list, or specify a different IDE by browsing to its location.

Detected R IDEs:
Other

Browse to your preferred R IDE:
C:\Program Files\RRO\R-3.2.2\bin\x64\Rgui.exe

Learn more about R IDEs

Change temporary storage location

Note: R custom visuals may require automatic installation of additional

***Figure 18-11.** R options*

There are, essentially, only a couple of possible tweaks that you could have to make:

- *R Scripting Engine*: Power BI Desktop should detect the R scripting engine that you have previously installed. Should it not do this, then you can browse to the directory containing the R scripting engine. Simply select Other in the popup list and click the Browse button.
- *IDE*: If you prefer to use a different external R integrated development interface (IDE), then simply click the Browse button to select a preferred R IDE. This will then be invoked the next time that you click the External Editor icon at the top right of the R script editor.

■ **Note** The vast majority of formatting for an R visual is done inside the R script itself.

Additional Visuals

By now, you must surely have come to appreciate the sheer range of visual possibilities that Power BI Desktop has to offer. From a range of chart types to gauges, tables, and cards, it delivers a wealth of easy-to-use ways of delivering insight into your data clearly and effectively.

Yet Power BI Desktop does not stop with the built-in visualizations that you have seen so far. It has been designed to be a completely open and extensible business intelligence application that will allow third parties (and even Microsoft) to add other visuals. This means that the core elements that you have met so far are only a starting point. There are many other chart, text, and map visuals that you can add to Power BI Desktop in a few clicks-and then literally stun your audience with an eye-catching variety of dashboard elements.

Adding and using these enhancements is both quick and easy. This is because all Power BI visualizations use the same interface and they are based on the underlying data model in exactly the same way. Consequently, learning to use a new visual will most likely only take a couple of minutes, as you are always building on the knowledge and techniques that you have already acquired.

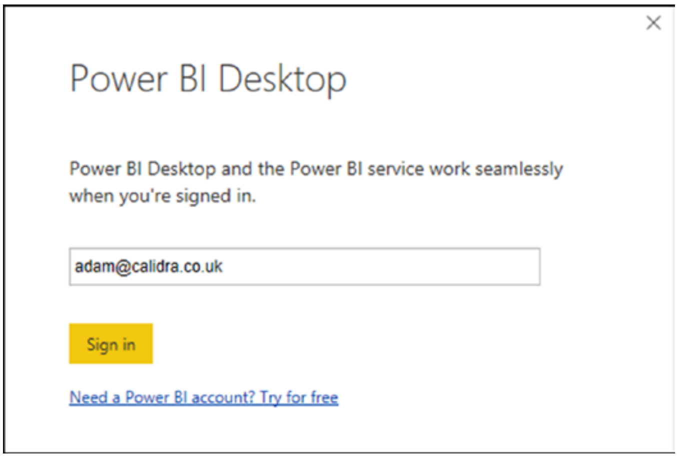
■ **Note** You need to be aware that not all the custom visuals that are available are entirely bug-free. Also, they might not have all the formatting attributes or interactive capabilities available that you might wish for. However, this does not make them any less useful—or any less fun to use. In any case, we can reasonably expect them to be enhanced and improved over time.

The Power BI Visuals Gallery

The first thing to do is become acquainted with the Power BI visuals gallery. This is a central hub where you can find a range of free visuals, developed either by Microsoft or by third parties, that are ready to be added to your Power BI dashboards.

The following explains how to connect to the Power BI visuals gallery:

1. In the Home ribbon, click the From Store button. If you are not logged in to a Power BI account (more on this in Chapter 23), you will see a dialog like the one shown in Figure 18-12, where you can log in.



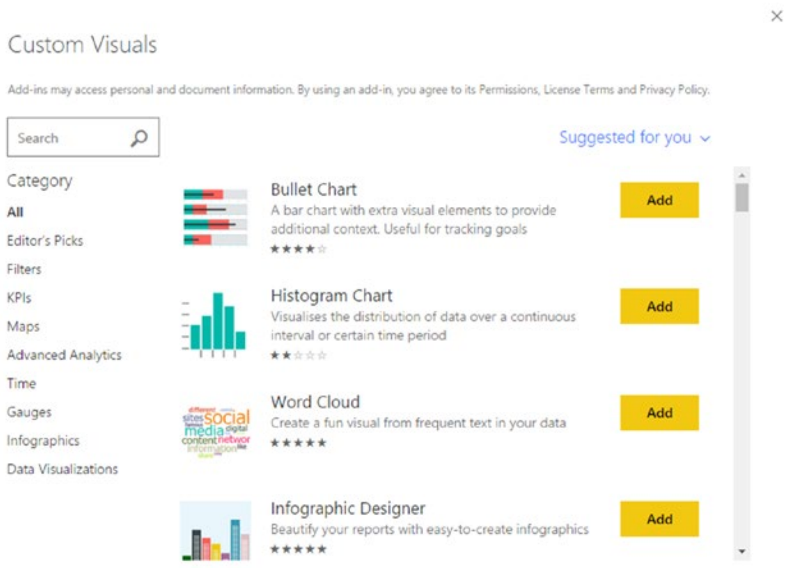
Power BI Desktop

Power BI Desktop and the Power BI service work seamlessly when you're signed in.

adam@calidra.co.uk

Figure 18-12. *The Power BI login dialog*

2. Enter your password and sign in. You will see the dialog from Figure 18-13.



Custom Visuals

Add-ins may access personal and document information. By using an add-in, you agree to its Permissions, License Terms and Privacy Policy.

Bullet Chart
A bar chart with extra visual elements to provide additional context. Useful for tracking goals

Histogram Chart
Visualises the distribution of data over a continuous interval or certain time period

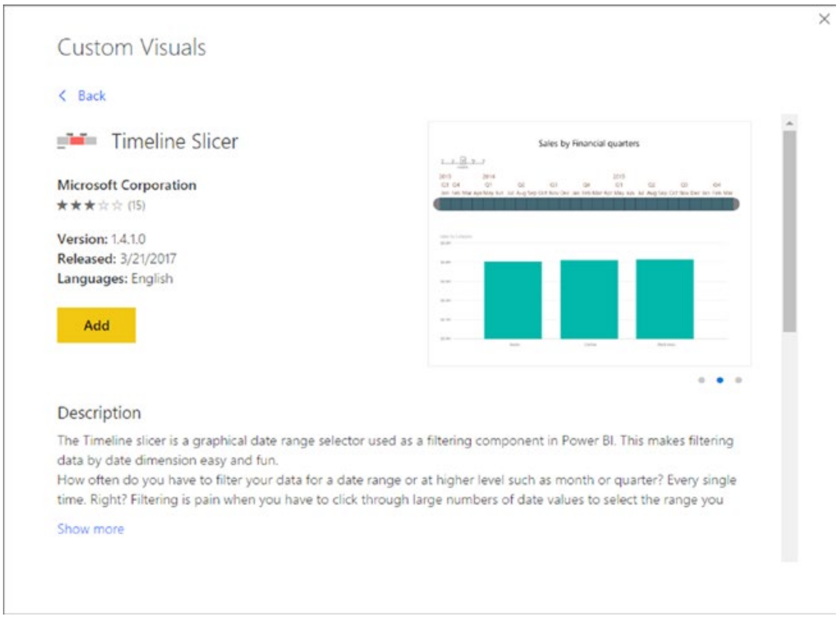
Word Cloud
Create a fun visual from frequent text in your data

Infographic Designer
Beautify your reports with easy-to-create infographics

Figure 18-13. *The Custom Visuals gallery*

3. Select a visual from among those available (you can search for a specific visual by entering a search term in the Search box or see them grouped and filtered by type if you click a type of visual on the left of the dialog).

4. Click Add. You will see a dialog like the one in Figure 18-14.



Custom Visuals

< Back

Timeline Slicer

Microsoft Corporation
★★★☆☆ (15)

Version: 1.4.1.0
Released: 3/21/2017
Languages: English

Add

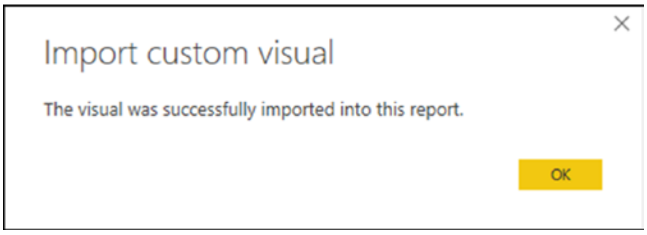
Description

The Timeline slicer is a graphical date range selector used as a filtering component in Power BI. This makes filtering data by date dimension easy and fun.
How often do you have to filter your data for a date range or at higher level such as month or quarter? Every single time. Right? Filtering is pain when you have to click through large numbers of date values to select the range you

Show more

Figure 18-14. *Selecting a third-party visual*

5. Click Add. The visual will be added to the palette of available visualizations and the dialog that you can see in Figure 18-15 will be displayed.



Import custom visual

The visual was successfully imported into this report.

OK

Figure 18-15. *Confirmation dialog for visual import*

6. Click OK. You can now use the visual in the current Power BI Desktop report.

This selection of third-party visuals has undoubtedly evolved considerably since this book went to print, and so I imagine that what you are looking at now is very different. Hopefully, though, you should be excited to see lots of new and amazing types of data visualization that will allow you to add unparalleled pizzazz to your dashboards.

Loading Custom Visuals

Another way to access this treasure trove of extra presentation elements for your reports is to download them to your PC and then add them to Power BI Desktop. Here is how to do this:

1. In your web browser, navigate to `https://app.powerbi.com/visuals`. You will see the Custom Visuals for Power BI web page. As things stand, it looks like Figure 18-16.

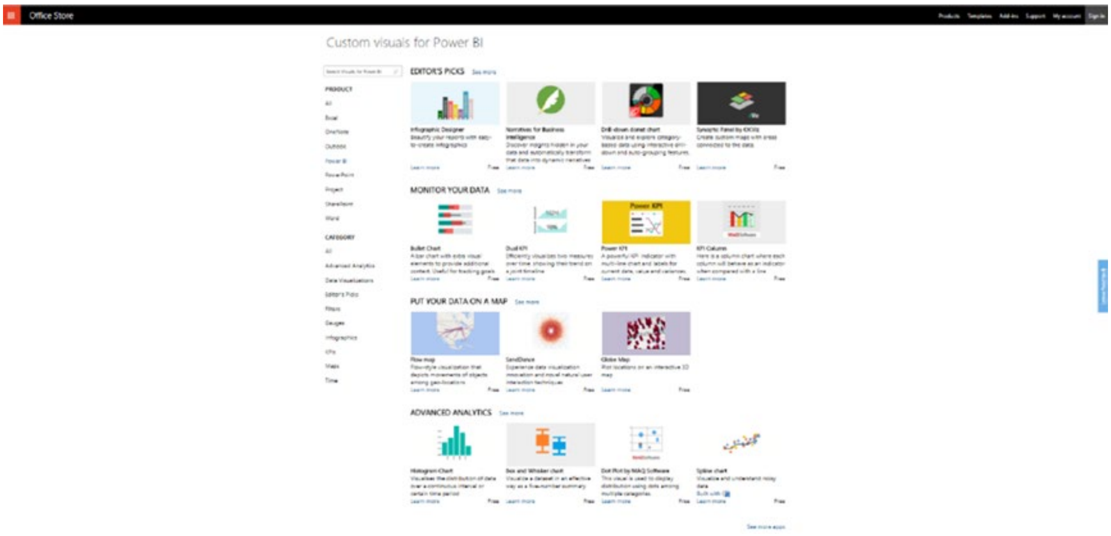
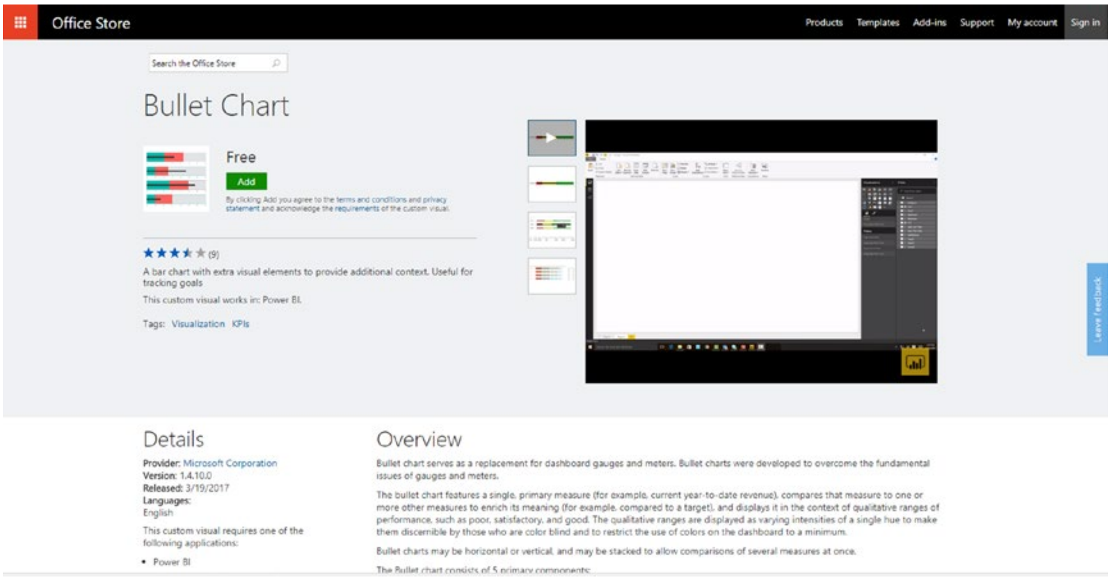


Figure 18-16. *The Custom Visuals for Power BI web page*

2. Click the visual that you want to download. A second web page devoted to this specific visual will be displayed. It will look something like Figure 18-17.



Bullet Chart

Free

Add

By clicking Add you agree to the terms and conditions and privacy statement and acknowledge the requirements of the custom visual.

★★★★☆ (9)

A bar chart with extra visual elements to provide additional context. Useful for tracking goals

This custom visual works in: Power BI.

Tags: Visualization KPIs

Details

Provider: Microsoft Corporation
Version: 1.4.10.0
Released: 3/19/2017
Languages:
English

This custom visual requires one of the following applications:

- Power BI

Overview

Bullet chart serves as a replacement for dashboard gauges and meters. Bullet charts were developed to overcome the fundamental issues of gauges and meters.

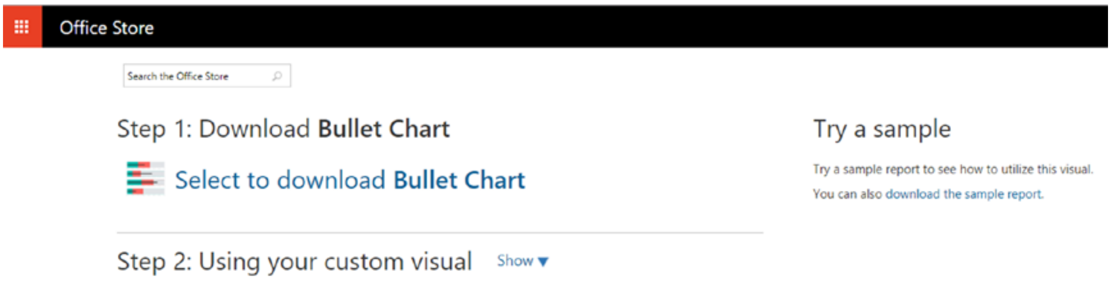
The bullet chart features a single, primary measure (for example, current year-to-date revenue), compares that measure to one or more other measures to enrich its meaning (for example, compared to a target), and displays it in the context of qualitative ranges of performance, such as poor, satisfactory, and good. The qualitative ranges are displayed as varying intensities of a single hue to make them discernible by those who are color blind and to restrict the use of colors on the dashboard to a minimum.

Bullet charts may be horizontal or vertical, and may be stacked to allow comparisons of several measures at once.

The Bullet chart consists of 5 primary components:

Figure 18-17. *The visuals download dialog*

3. Click Add. The web page shown in Figure 18-18 will be displayed.



Step 1: Download **Bullet Chart**

Select to download **Bullet Chart**

Step 2: Using your custom visual Show ▾

Try a sample

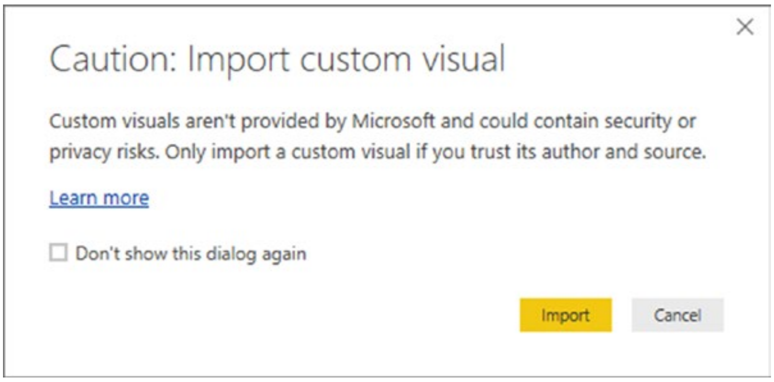
Try a sample report to see how to utilize this visual.

You can also download the sample report.

Figure 18-18. *The visuals community agreement dialog*

4. Click Select to Download. The visual will be downloaded. Depending on your version of Windows (and any machine-specific configuration), it will be placed in a specific folder or give you the choice of a folder.

5. In Power BI Desktop, click From File in the Home ribbon. A cautionary dialog like the one in Figure 18-19 will appear.



Caution: Import custom visual

Custom visuals aren't provided by Microsoft and could contain security or privacy risks. Only import a custom visual if you trust its author and source.

Learn more

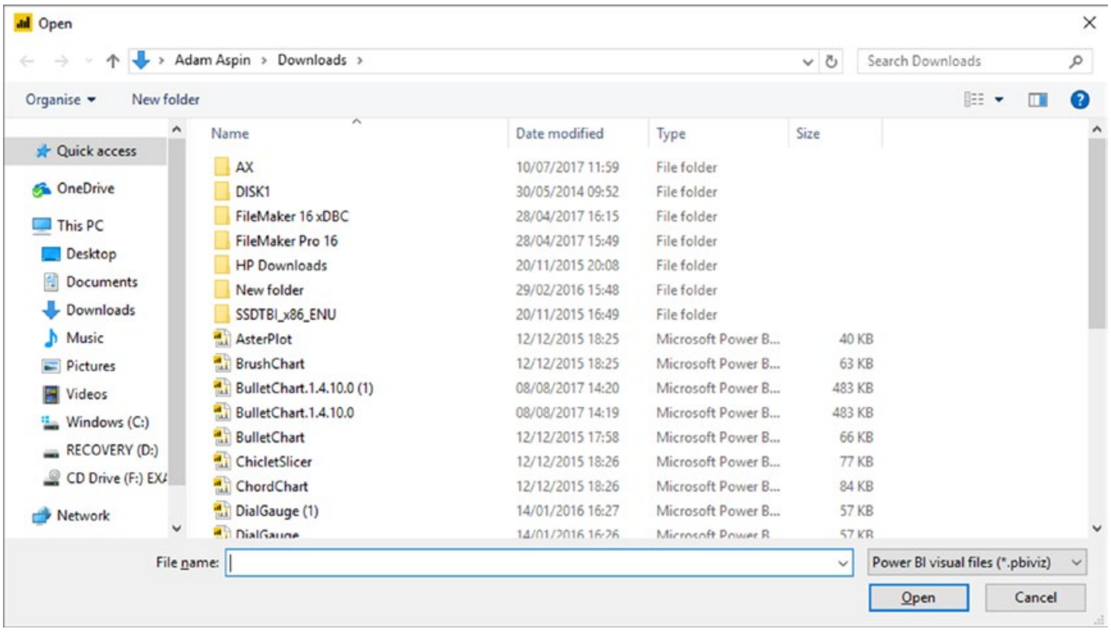
☐ Don't show this dialog again

Import Cancel

Figure 18-19. *The import visuals dialog*

6. Click Import. The Windows File Open dialog will appear.

7. Navigate to the folder where you downloaded the visual file in step 3. Click the visual file. You can see an example of this in Figure 18-20.



Open

Adam Aspin > Downloads >

Search Downloads

Organise ▾ New folder

Quick access
OneDrive
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
RECOVERY (D:)
CD Drive (F:) EX
Network

Name	Date modified	Type	Size
AX	10/07/2017 11:59	File folder	
DISK1	30/05/2014 09:52	File folder	
FileMaker 16 xDBC	28/04/2017 16:15	File folder	
FileMaker Pro 16	28/04/2017 15:49	File folder	
HP Downloads	20/11/2015 20:08	File folder	
New folder	29/02/2016 15:48	File folder	
SSDTBI_x86_ENU	20/11/2015 16:49	File folder	
AsterPlot	12/12/2015 18:25	Microsoft Power B...	40 KB
BrushChart	12/12/2015 18:25	Microsoft Power B...	63 KB
BulletChart.1.4.10.0 (1)	08/08/2017 14:20	Microsoft Power B...	483 KB
BulletChart.1.4.10.0	08/08/2017 14:19	Microsoft Power B...	483 KB
BulletChart	12/12/2015 17:58	Microsoft Power B...	66 KB
ChicletSlicer	12/12/2015 18:26	Microsoft Power B...	77 KB
ChordChart	12/12/2015 18:26	Microsoft Power B...	84 KB
DialGauge (1)	14/01/2016 16:27	Microsoft Power B...	57 KB
DialGauge	14/01/2016 16:26	Microsoft Power B	57 KB

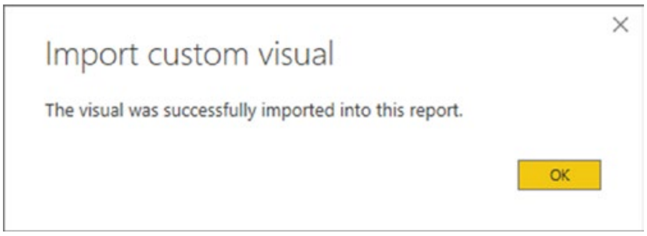
File name:

Power BI visual files (*.pbiviz)

Open Cancel

Figure 18-20. *Loading a visual from file*

8. Click Open. The confirmation dialog will appear, as shown in Figure 18-21.


Import custom visual

The visual was successfully imported into this report.

Figure 18-21. *The visual import confirmation dialog*

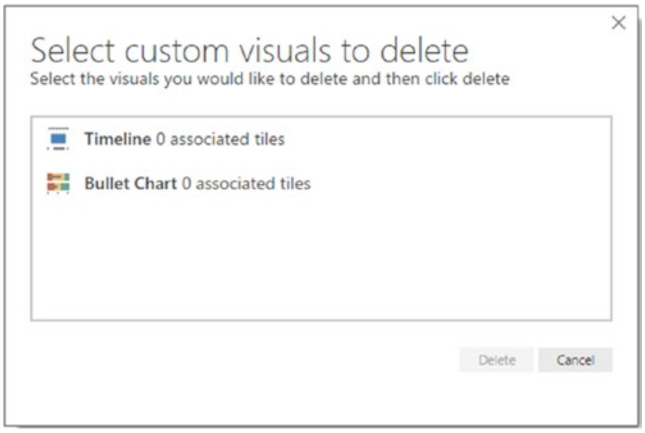
The icon for the visual will appear at the bottom of the visuals gallery in the Visualizations pane. You can then use this visual just like any other Power BI visual.

■ **Tip** Visuals are added as required to each individual Power BI Desktop file. So while you only download the visuals once, you have to load them into each Power BI Desktop (.pbix) file that you want to use them in. If you find yourself frequently using the same third-party visuals, you should create a Power BI Desktop file that contains all the visuals that you use, and then save it as a template for your future dashboard development. This was explained in Chapter 9.

Another way to import custom visuals that you have already downloaded is to click the ellipses in the gallery of visuals in the Visualizations pane. This will display a popup menu with the option to import visuals-either from the Power BI Store or from your collection of downloaded visuals.

If you wish to remove a third-party visual from the Visualizations pane:

1. Click the ellipses in the gallery of visuals in the Visualizations pane
2. Select Delete a Custom Visual. The dialog that you can see in Figure 18-22 will appear.


Select custom visuals to delete

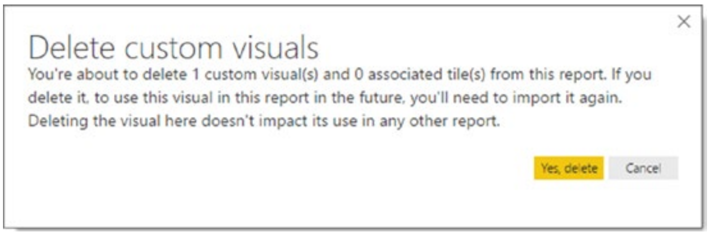
Select the visuals you would like to delete and then click delete

Timeline 0 associated tiles

Bullet Chart 0 associated tiles

Figure 18-22. *Deleting a custom visual*

3. Select the visual to remove from the current report.
4. Click Delete. The dialog shown in Figure 18-23 will be displayed.


Delete custom visuals

You're about to delete 1 custom visual(s) and 0 associated tile(s) from this report. If you delete it, to use this visual in this report in the future, you'll need to import it again. Deleting the visual here doesn't impact its use in any other report.

Figure 18-23. *The custom visuals delete dialog*

5. Click Yes, Delete.

■ **Note** This will not remove any files that you have previously downloaded. You delete those as you would any standard file.

A Rapid Overview of a Selection of Custom Visuals

As the gallery of custom visuals is in a state of permanent flux, it is impossible to discuss all the currently available extensions to Power BI. However, to give you an idea of some of the possible enhancements that are available, the next few pages show some of the added visuals made available by Microsoft. Since there is no guarantee that these visuals will still be available in their current state by the time that you are reading this book, I do not explain how to create them, but merely show some examples of other chart types using the data available in the sample data that you have been using in this book.

Aster Plots

An *aster plot* is derived from the standard donut chart. However, it uses an extra data value to provide the "sweep" that you see in Figure 18-24.

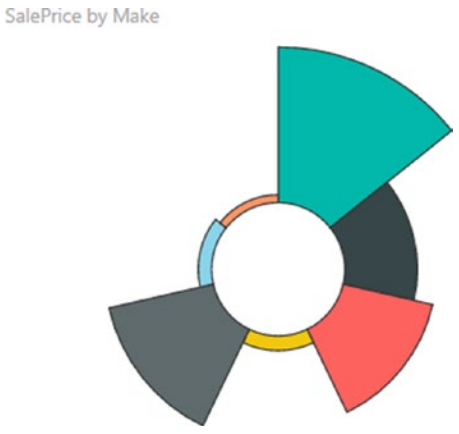


Figure 18-24. *An aster plot*

Radar Charts

Radar charts are often used in performance analyses to display metrics. They allow you to show multiple values relative to a shared axis, as you can see in Figure 18-25.

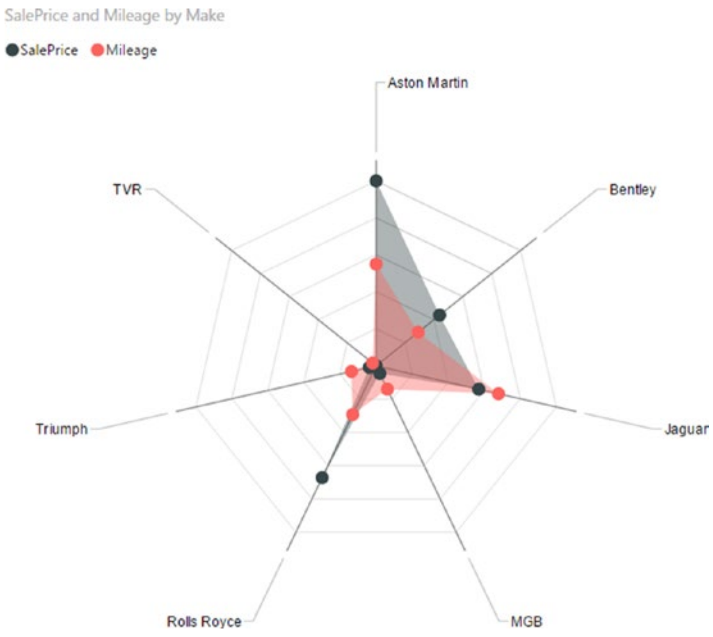


Figure 18-25. *A radar chart*

Bullet Charts

Bullet charts are extremely useful for tracking KPIs against targets. They frequently require you to add data that provides the thresholds against which performance is measured, either as absolute values or as percentages. However, they can be extremely effective at displaying how results compare to targets, as Figure 18-26 shows.

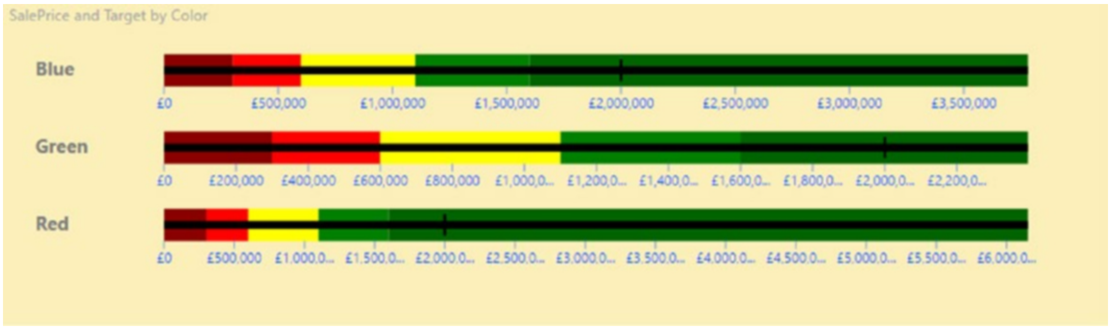


Figure 18-26. *A bullet chart*

Word Clouds

A *word cloud* shows the number of times that words in a dataset appear relative to each other. It can be a uniquely visual way to show relative weights of values, as you can see in Figure 18-27.

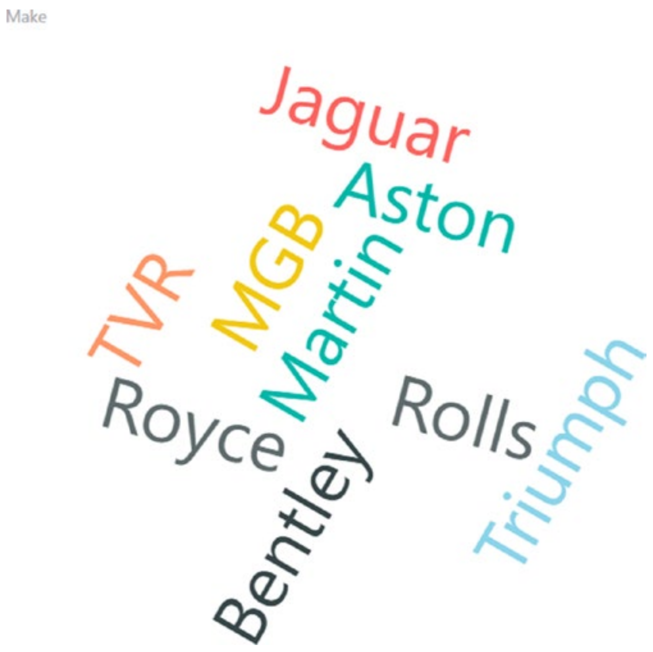


Figure 18-27. *A word cloud*

Sunburst Charts

Sunburst charts are essentially multilevel donut charts. However, as you can see in Figure 18-28, they are good at showing hierarchies of data. The inner ring represents each car make and the corresponding segments of the outer ring show the sales for each model compared to the sales by make and the whole.

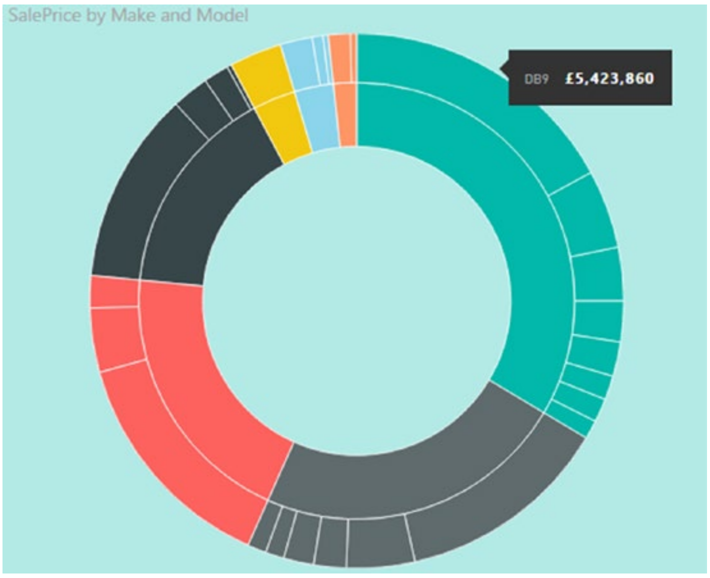


Figure 18-28. *A sunburst chart*

Streamgraphs

A *streamgraph* is a smoothed, stacked, area chart. As you can see in Figure 18-29, a streamgraph is good at showing how values change over time.

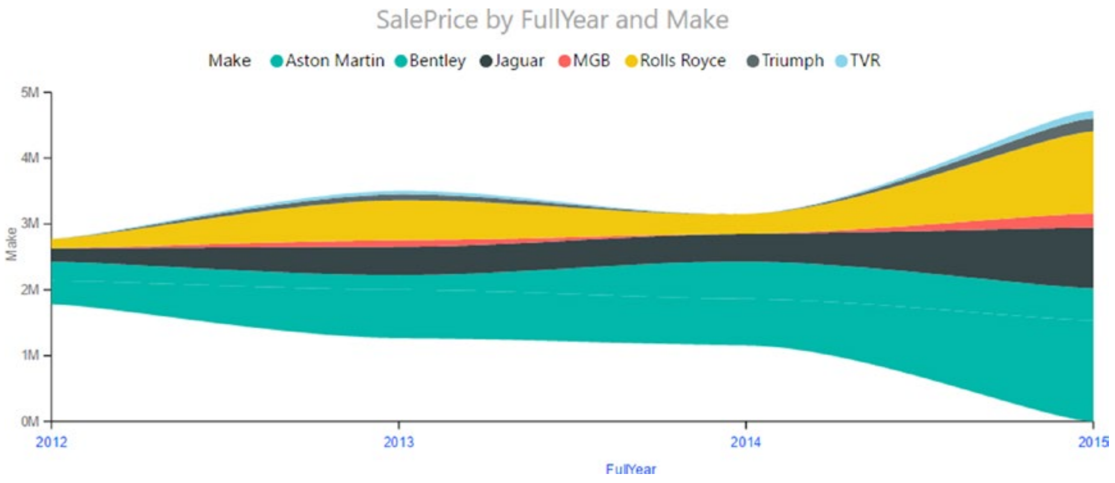


Figure 18-29. *A streamgraph*

Interestingly, a streamgraph has many more formatting options available than is the case for some other imported visuals, as you can see in this example.

Tornado Charts

A *tornado chart* is a variation of a bar chart, only the separation between two sets of values is made clearer by the vertical separation, as you can see in Figure 18-30. As was the case with bar charts, you can sort tornado charts by any of the data elements in the chart.

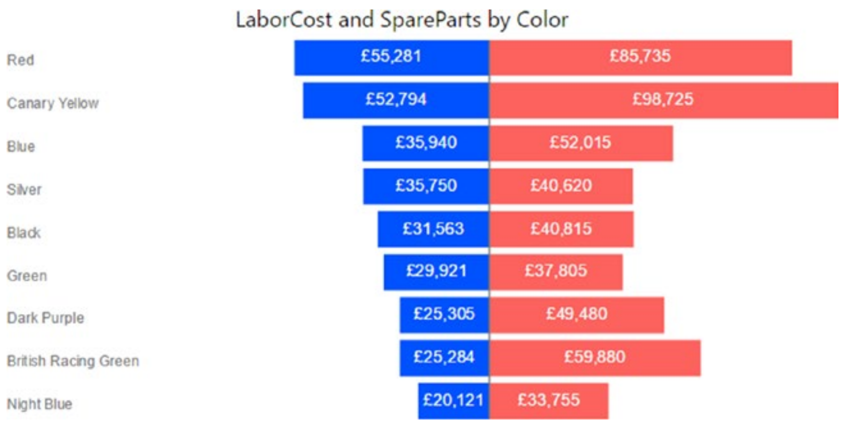


Figure 18-30. *A tornado chart*

Chord Charts

A *chord chart* is an interesting—and somewhat less traditional—way of displaying the relationship between values in a matrix structure. Figure 18-31 shows how colors and countries relate by sales.

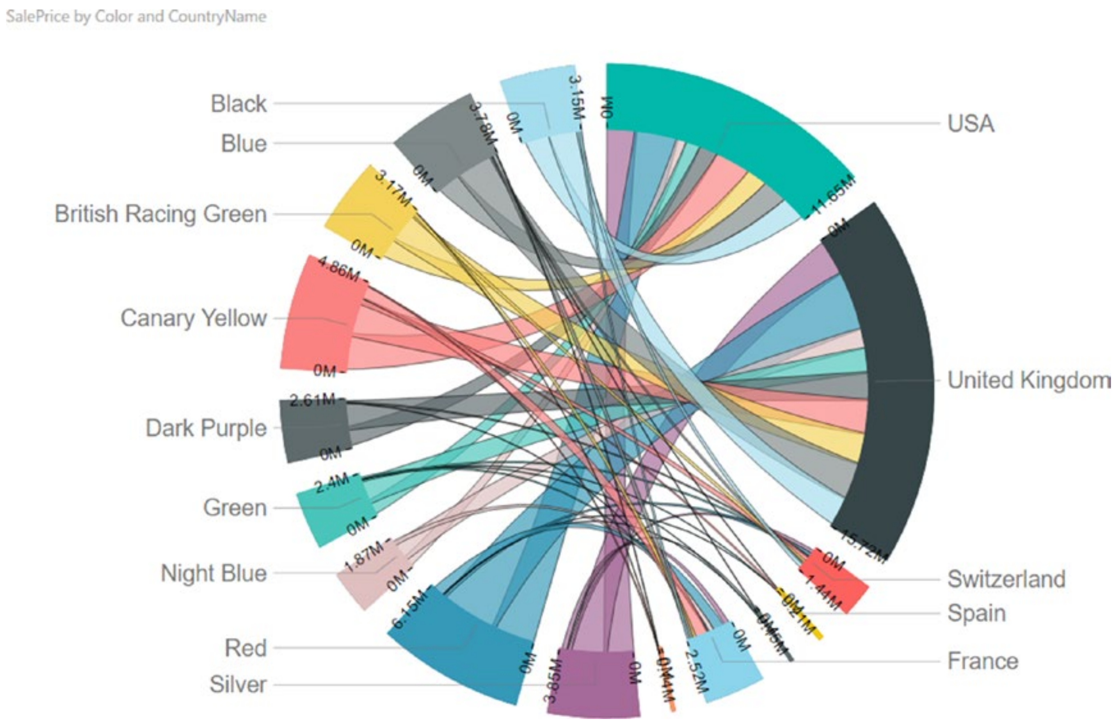


***Figure 18-31.** A chord chart*

Sankey Diagrams

The *Sankey diagram* in Figure 18-32 also displays how colors and country sales relate. This chart has been filtered so that it only displays data for four countries.

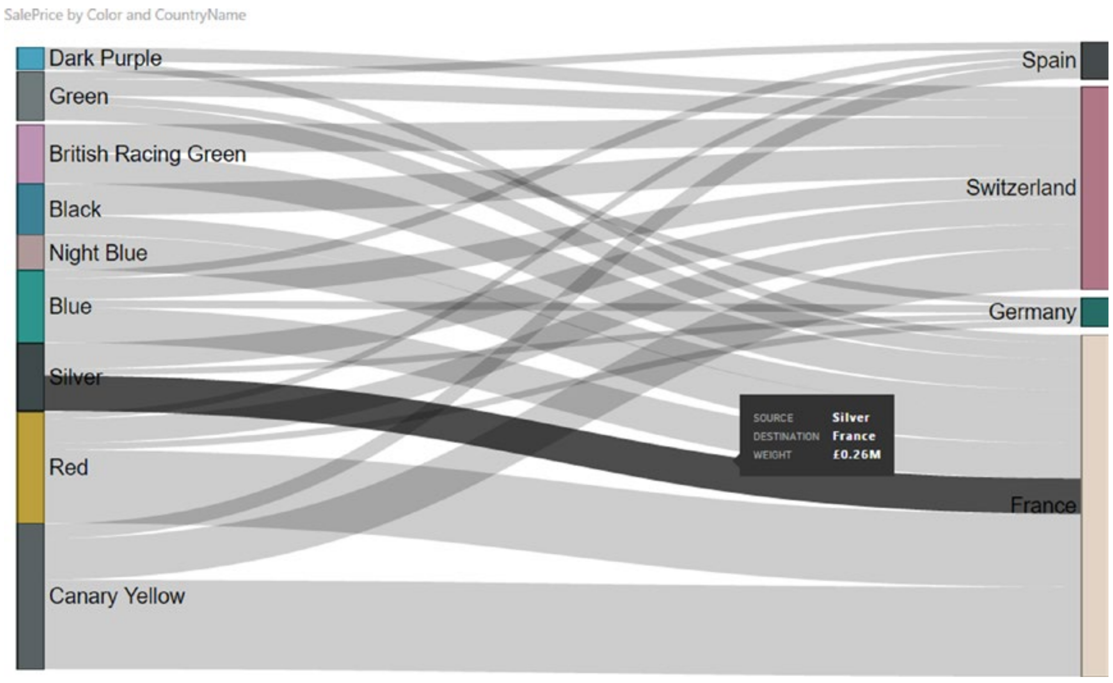


Figure 18-32. *A Sankey diagram*

Correlation Plots

The correlation plot in Figure 18-33 shows how spare parts and labor costs relate.

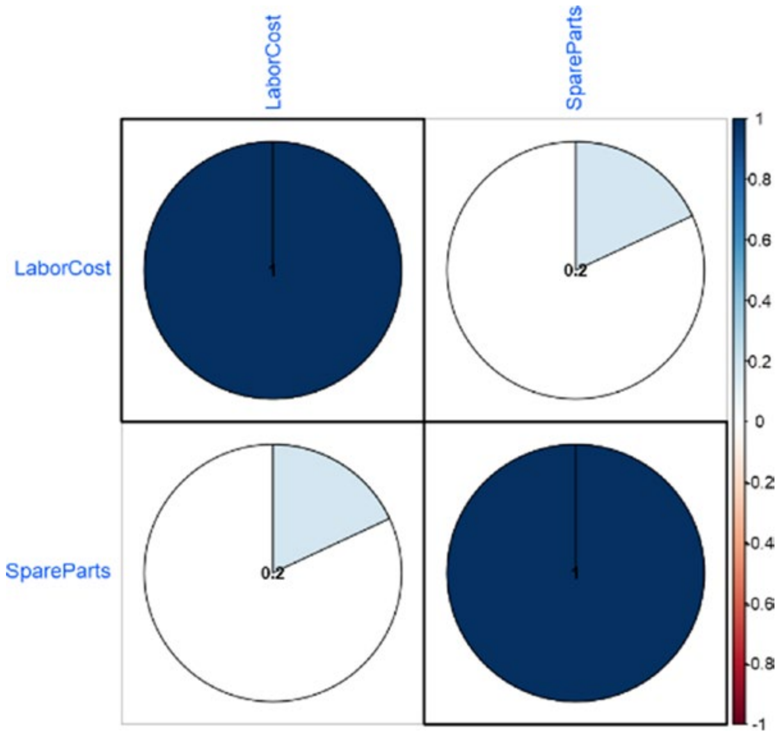
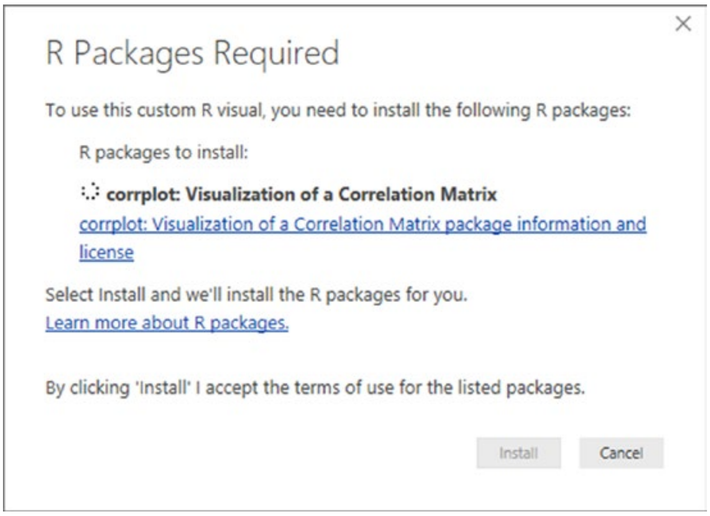


Figure 18-33. *A correlation plot*

It is worth remarking that this visual (like many of the third-party visuals available) uses the R language. This kind of visual will require you to have one or more R packages installed on your computer. If these packages are not already installed, you will see a dialog like the one shown in Figure 18-34.


R Packages Required

To use this custom R visual, you need to install the following R packages:

R packages to install:

corrplot: Visualization of a Correlation Matrix

corrplot: Visualization of a Correlation Matrix package information and license

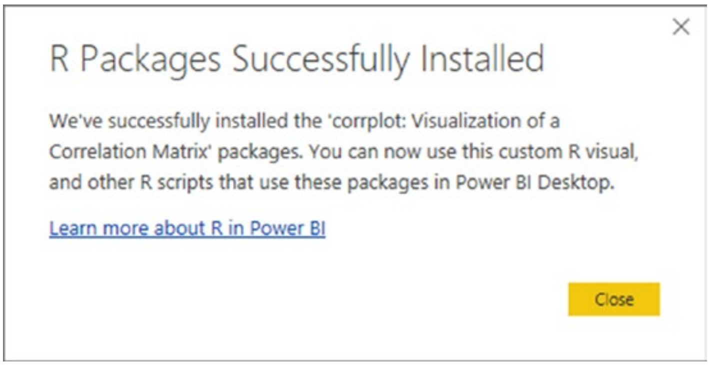
Select Install and we'll install the R packages for you.

Learn more about R packages.

By clicking 'Install' I accept the terms of use for the listed packages.

Figure 18-34. *R packages that need installing locally*

Clicking Install will install the R packages, and then cause the dialog that you can see in Figure 18-35 to be displayed.


R Packages Successfully Installed

We've successfully installed the 'corrplot: Visualization of a Correlation Matrix' packages. You can now use this custom R visual, and other R scripts that use these packages in Power BI Desktop.

Learn more about R in Power BI

Figure 18-35. *Successfully installed R packages for a custom visual*

Countdown

The countdown shows the number of days, hours, minutes, and seconds in a “clock” that decrements in real time in any report. You can see an example in Figure 18-36.

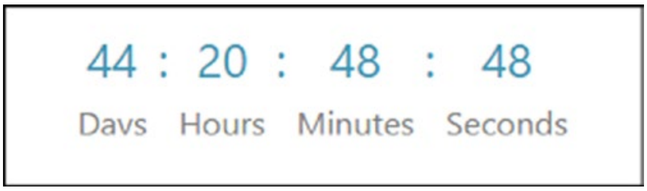


Figure 18-36. *The countdown*

Custom Slicers

There are several excellent custom slicers available in the Microsoft App Store. I will introduce a few of these in Chapter 21.

Conclusion

Over the course of the previous five chapters, you have met a wide range of visuals that let you express the trends hidden in your data. This chapter built on the previous chapters, by adding tree maps, gauges and R-based visuals to the mix.

Then, just as you thought that it could not get any better, you discovered the jewel in the crown of Power BI Desktop: an extensive range of freely available extensions that you can add to your dashboards to deliver your insights in ways that leave other analytics tools in the dust.

CHAPTER 19

Maps in Power BI Desktop

Another powerful technique that you can use to both analyze and present your insights is to display the data in map form. All that this requires is that your source data contains information that can be used for geographical representation. So if you have country, state, town, postal (or ZIP) code, or even latitude and longitude in the dataset, then you can get Power BI Desktop to add a map to your report and show the selected data using the map as a background.

Better yet, a Power BI Desktop map behaves just like any other visual. This means that you can filter the data that is displayed in a map, as well as highlighting it, just as you can do for charts, tables, and matrices. Not only that, but a map is an integral part of a Power BI Desktop report. So if you highlight data in a chart, a map in the same report will also be highlighted. However, you will have to wait for the next chapter to discover all this in detail.

Power BI Desktop provides you with four types of map visuals:

- Maps
- Filled maps
- Shape maps
- ArcGIS maps

The first three map types are simple and expressive. The fourth type (ARCGis maps) requires a little more effort to master, but is capable of delivering some truly stunning output.

The aim of this chapter is to show you some of the ways in which you can add real spice to your reports by using maps. Then, when presenting your analyses, you can interact with the maps and *really* impress your audience.

This chapter will use the file C:\PowerBiDesktopSamples\CH19\CarSalesDataForReports.pbix as the basis for the visuals that you will create.

Working with Bing Maps

Before adding the first map, I want to explain how mapping works in Power BI Desktop. The geographical component is based on Bing Maps. So, in order to add a map, you need to be able to connect to the Internet and use the Bing Maps service. Secondly, the underlying dataset must contain fields that are recognized by Bing Maps as geographical data. In other words, you need country, state, town, or other information that Bing Maps can use to generate the plot of the map. Fortunately, Power BI Desktop will indicate if it recognizes a field as containing data that it can use (hopefully) to create a map, as it will display a tiny icon of a globe in the field well for every field in the underlying dataset that apparently contains geographical data.

© Adam Aspin 2018
A. Aspin, *Pro Power BI Desktop*, https://doi.org/10.1007/978-1-4842-3210-1_19

To avoid the risk of misinterpreting data, you can add metadata to the underlying data model, which defines geographical field types. By applying data categories to fields, Power BI Desktop maps will then use these categories to interpret geographical data for mapping. Preparing data so that any fields used by Bing Maps are not only recognizable as containing geographic data, but also uniquely recognizable, is vital. You must help Bing Maps so that if you are mapping data for a city named Paris, Bing can see whether you mean Paris, France, or Paris, Texas. Chapter 10 explains some of the ways in which you can prepare your data for use by Bing Maps, and consequently, use it to add map visuals to Power BI Desktop.

■ **Note** There are some areas of the world that cannot use Bing Maps. So if you attempt to use Power BI Desktop mapping in these geographical zones, you will not see any map appear when you attempt to create a map.

Creating Maps in Power BI Desktop

Let's begin by creating a map of Sales by Country. Fortunately, the sample dataset contains the country where the sale was made. This means that we can use this data to make Power BI Desktop display a map of our worldwide sales. Here is how to create an initial map:

1. Open the C:\PowerBiDesktopSamples\CH19\CarSalesDataForReports.pbix file.
2. Click the Map icon in the Visualizations pane (this is the largely empty globe that you can see in Figure 19-1). A blank map will appear (even if it looks a little peculiar, it is meant to be a map).
3. Expand the Countries table and drag the CountryName field on to the map visual. The visual will display a map showing points in all the countries for which there are values.
4. Expand the InvoiceLines table and drag the SalePrice field on to the map visual. The visual will change the size of each data point to reflect the sales value.
5. Place the cursor over the map. The pointer icon will become a hand icon. Click and drag Europe to the center of the map.
6. Using the mouse wheel, zoom in to fit the countries with sales in the center of the visual. It will look something like Figure 19-1.

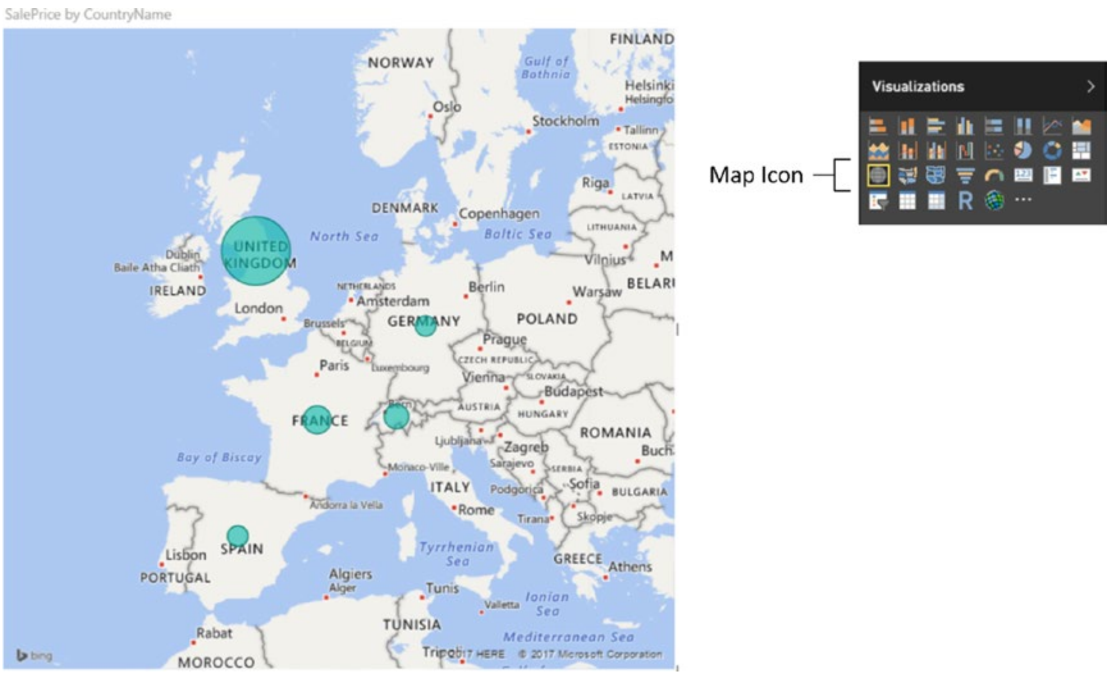


Figure 19-1. *A map of sales*

It is probably worth clarifying a few points about maps in Power BI Desktop before we go any further. The following are the essential points to note:

- A map is a visual like any other. You can resize and move it anywhere on the Power BI Desktop canvas.
- The map will apply any filters that have been set for the report (you will learn more about filters in the next chapter).
- Each data point (or bubble) in a map is proportional to the relative size of the underlying data.

You can hover the mouse pointer over a data "bubble" to display a popup showing the exact data that is represented. An example of this is shown in Figure 19-2.

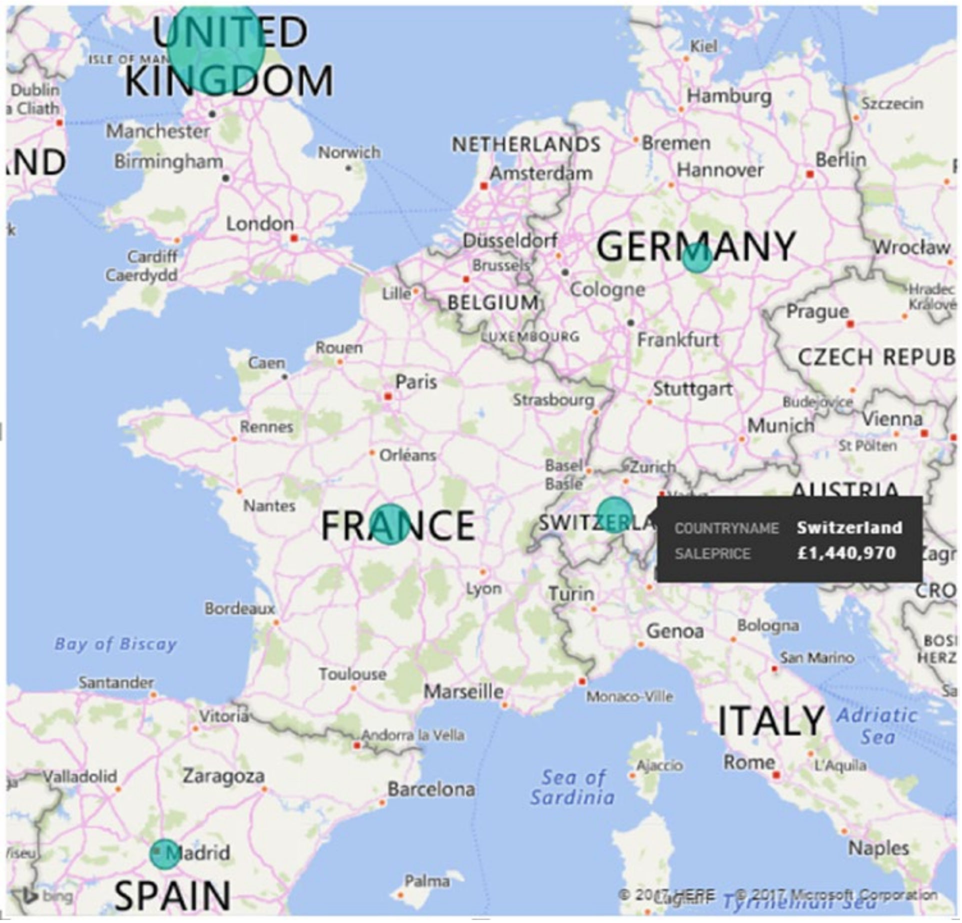


Figure 19-2. *Displaying the exact data in a Power BI Desktop map*

Using Geographical Data

While it might seem obvious, you need geographical data if Power BI Desktop is to display your insights overlaid upon a map. So you really, absolutely must ensure that your underlying dataset contains columns of data that can be interpreted as geographical information. In the examples so far, you have seen how Power BI Desktop is capable of interpreting country names and using these to display data in maps. However, it is not limited to displaying only countries. You can use any of the following to generate map visuals:

- Country
- Zip (postal) code
- State
- County
- Town
- Latitude and longitude

As an example, take a look at Figure 19-3, where the Town field in the Clients table was used in the Location box in the field well. The map was then adjusted to display only the United Kingdom using the techniques described in steps 5 and 6 of the initial example.

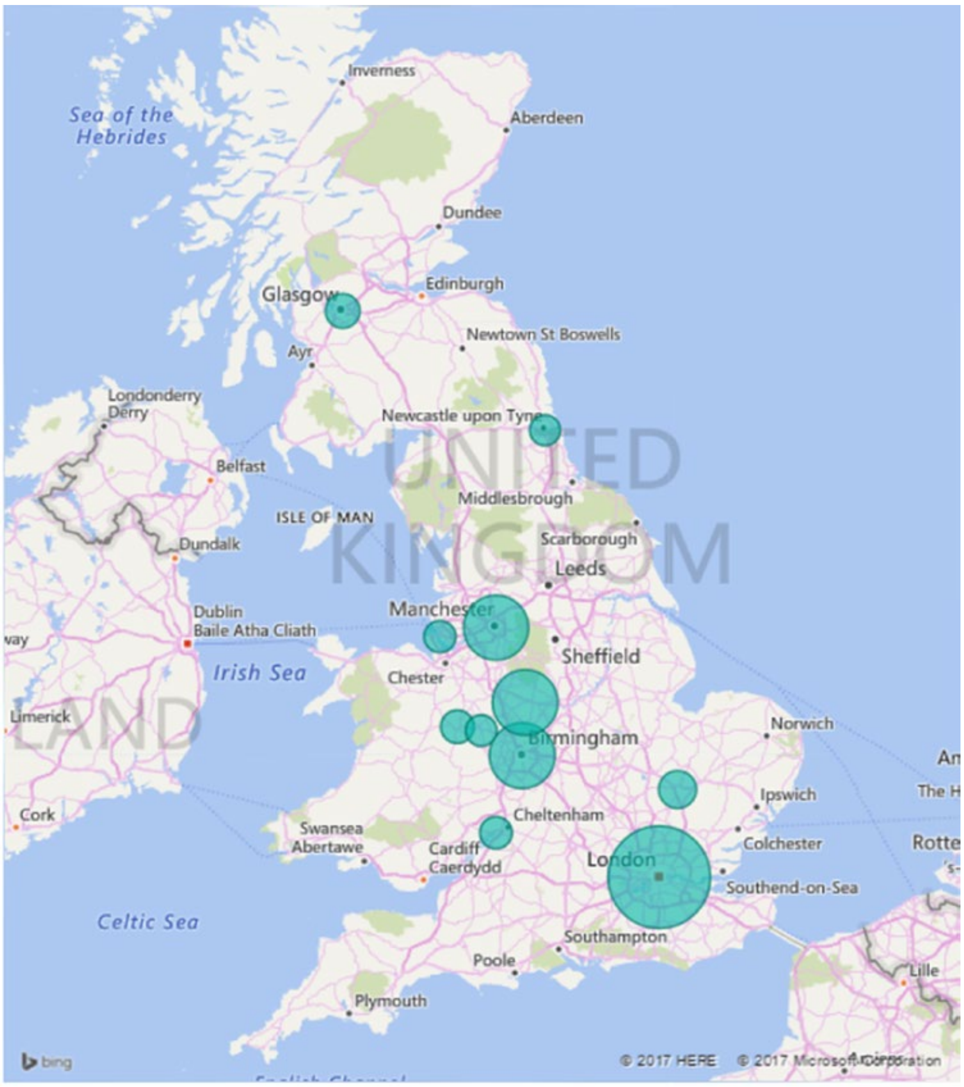


Figure 19-3. *Displaying relative sales per town*

Geographical Data Types

Power BI Desktop really is exceptionally helpful and forgiving when it comes to creating maps. If the data that you have can be interpreted geographically, then Power BI Desktop will do its best to display a map. However, you will almost certainly have to mold the dataset into a coherent data model before you start using Power BI Desktop.

In the first Power BI Desktop example, we added a single geographical data field. What is more, this field was recognized instantly for what it was—country names. In the real world of mapping data, however, you may have to not only add several fields but also specify which type of geographical data each field represents. Put simply, Power BI Desktop needs to know what the data you are supplying represents. Not only that, it needs to know what it is looking at without ambiguity. Consequently, it is up to you to define the source data as clearly and unambiguously as possible. This can involve one or more of several possible approaches.

Define the Data Category in the Data Model

As you saw in Chapter 11, you can define a data category for each column of data in Power Pivot. Although this is not an absolute prerequisite for accurate mapping with Power BI Desktop, it can help reduce the number of potential anomalies.

Add Multiple Levels of Geographical Information

The Power BI Desktop data model lets you add several levels of geographical information to a table. For instance, you can add not just a country, but also a county and a town to a record in a table. The advantage of adding as many relevant source data fields as possible is that by working in this way, you are helping Power BI Desktop dispel possibly ambiguous references. For instance, if you only had a Town field, Power BI Desktop might not know if you are referring to Birmingham, Alabama, or England's second city. If the data source has a Country field *and* a Town field, however, then Power BI Desktop has a much better chance of detecting the correct geographical location. This principle can be extended to adding states, counties, and other geographical references.

Drilling Down in Maps

Power BI Desktop lets you drill into geographical data just as you can drill into other data hierarchies in other types of visualization. To see this, you can carry out the following steps:

1. Create a new map visual, using the following fields (and in this order):
    a. CountryName (from the Countries table)
    b. Town (from the Clients table)
    c. SalePrice (from the InvoiceLines table)
2. Click the Drill Mode icon at the top right of the map visual to activate drill-down.
3. Click the bubble for the United Kingdom. The map visual will display data at the next level (town) for the selected country. You can see this in Figure 19-4, which shows both levels of data.

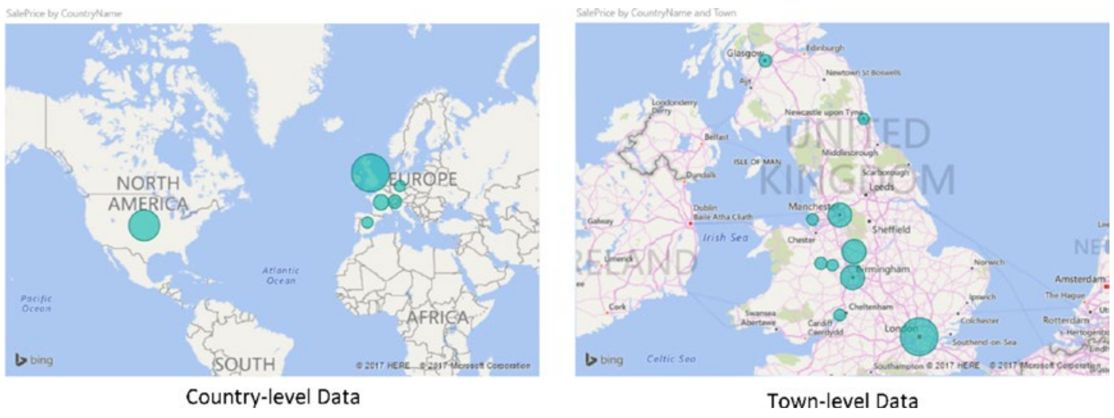


Figure 19-4. *Drilling down into a geographical hierarchy*

Fortunately, the techniques that you use to drill down and up in maps are identical to those that you have already seen for matrices and charts. To resume, briefly:

- Click the Drill Up icon at the top left to go up one level.
- Click the "Go to the next level in the hierarchy" icon to see data from the next level down.
- Click the "Go to the next level in the hierarchy" icon to see all data from the next level down.

■ **Note** As you might expect, Power BI Desktop will zoom into a country to show the detail at town level if you are drilling down from the country. The map will stay at its original resolution if you decide to show all data from a lower level.

Adjusting the Map Display in Power BI Desktop

As you have seen, creating a map is extremely easy. However, the initial map is not necessarily the finalized version that you wish to show to your audience. You may wish to

- Position the map elements more precisely inside the visual.
- Zoom in or out of the map.

In the next few sections, we will look at the various modifications that you can make to Power BI Desktop maps. Hopefully, you will find these tweaks both intuitive and easy to implement. In any case, with a little practice you should find that these modifications take only a few seconds to accomplish.

Positioning the Map Elements

If the area displayed in a map is not quite as perfect as you would prefer, then you can alter the area (whether it is a country or a region) that appears in the map visual. You can do this as follows:

1. Place the mouse pointer over the map. The pointer icon will become a hand icon.
2. Click and drag the pointer around to move the map elements.

Zooming In or Out

It is conceivable that the map that is displayed is not at a scale, which you would prefer. Fortunately, this is extremely easy to fix. All you have to do is the following:

1. Place the mouse pointer over the map.
2. Roll the mouse scroll wheel forward to zoom in to see part of the map in greater detail. Alternatively, roll the mouse scroll wheel backward to zoom out of the map.

■ **Note** There are currently no keyboard shortcuts for moving around in maps, or zooming in and out of maps.

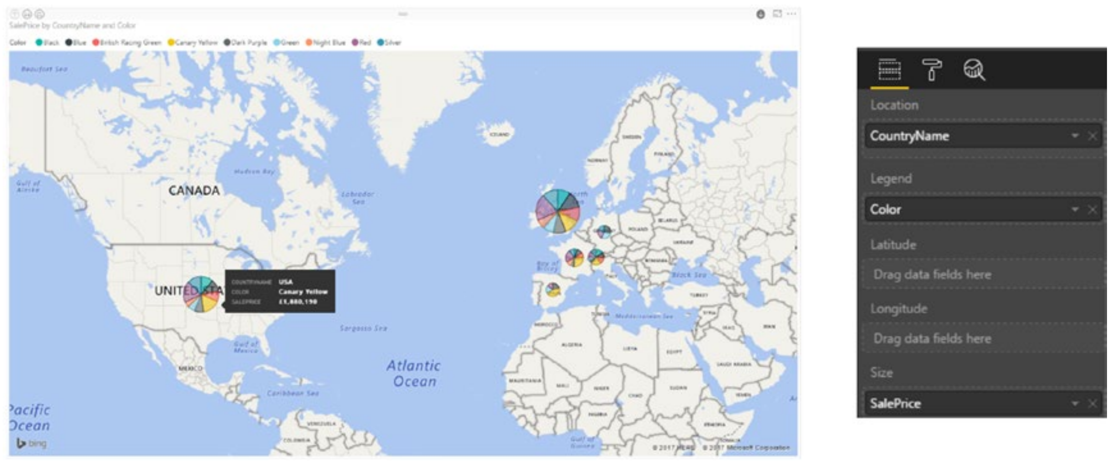
Multivalue Series

So far, you have seen how you can add a single data series to a map and have the data represented as a data point. Power BI Desktop can extend this paradigm by allowing you to display the data bubble as a pie that contains a second data series—and consequently display the data broken down by a specific dataset per geographical entity.

As an example of this (and to revise some of the map creation techniques that we have seen so far), let's try to analyze European car sales by age range:

1. Keep the map that you created in the previous section (or create it if you have not yet done so).
2. Expand the Colors table and drag the Color field into the field well onto the Legend box.
3. Resize the map visual.

The map now contains a legend for the colors of vehicles sold. Each bubble is now a pie of data. The overall size of the pie represents, proportionally, the sum total data for each country compared to the other pies. The map should now look like Figure 19-5.


SalePrice by CountryName and Color
Color ● Black ● Blue ● British Racing Green ● Canary Yellow ● Dark Purple ● Green ● Night Blue ● Red ● Silver
CANADA
UNITED STA
Atlantic Ocean
Pacific Ocean
Gulf of Mexico
Caribbean Sea
Sargasso Sea
Labrador Sea
Hudson Bay
Beaufort Sea
bing

COUNTRYNAME USA
COLOR Canary Yellow
SALEPRICE £1,880,190

Location
CountryName
Legend
Color
Latitude
Drag data fields here
Longitude
Drag data fields here
Size
SalePrice

Figure 19-5. *Displaying pie charts in a Power BI Desktop map*

It is worth remarking that if you hover the mouse pointer over the data representation (the pie) for a country, as you pass the pointer over each pie segment, you will see a tooltip giving the details of the data, including which car color it refers to.

■ **Note** When displaying maps, you nearly always need to filter data in some way. You will discover all the details about filtering data in the next chapter.

Color Saturation

In a similar way to what you already saw when creating bubble charts, you can set the color saturation of bubbles in a map. To do this:

1. Create a new map visual, using the following fields (and in this order):
   a. Town (from the Countries table)
   b. SalePrice (from the InvoiceLines table)
2. Drag the Gross Margin field from the InvoiceLines table into the Color Saturation well. The map will look like the one in Figure 19-6 (after centering on the UK).

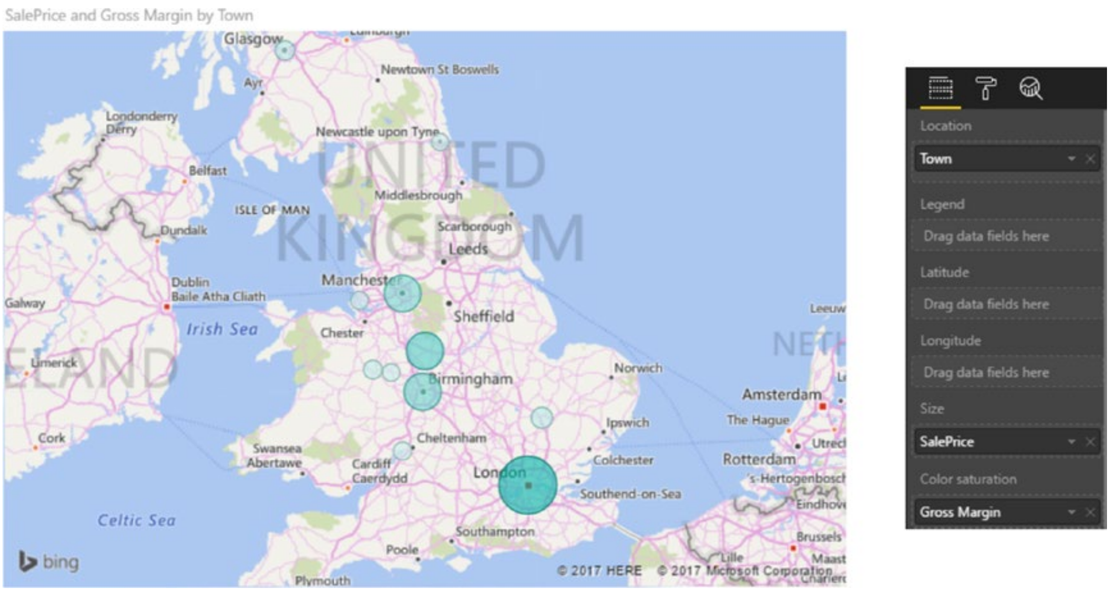


Figure 19-6. *Applying color saturation to a map*

■ **Note** You can apply color saturation or a legend to a map-but not both at once.

Highlighting Map Data

If you have added data to the Legend box of the field well, and a legend is displayed, you can highlight segments of data in a map. This allows you to draw the audience's attention to specific trends in your data.

Using the map that you created and can see in Figure 19-5, click Blue (the second element) in the legend. The pie segments that correspond to blue cars sold are highlighted, as shown in Figure 19-7.

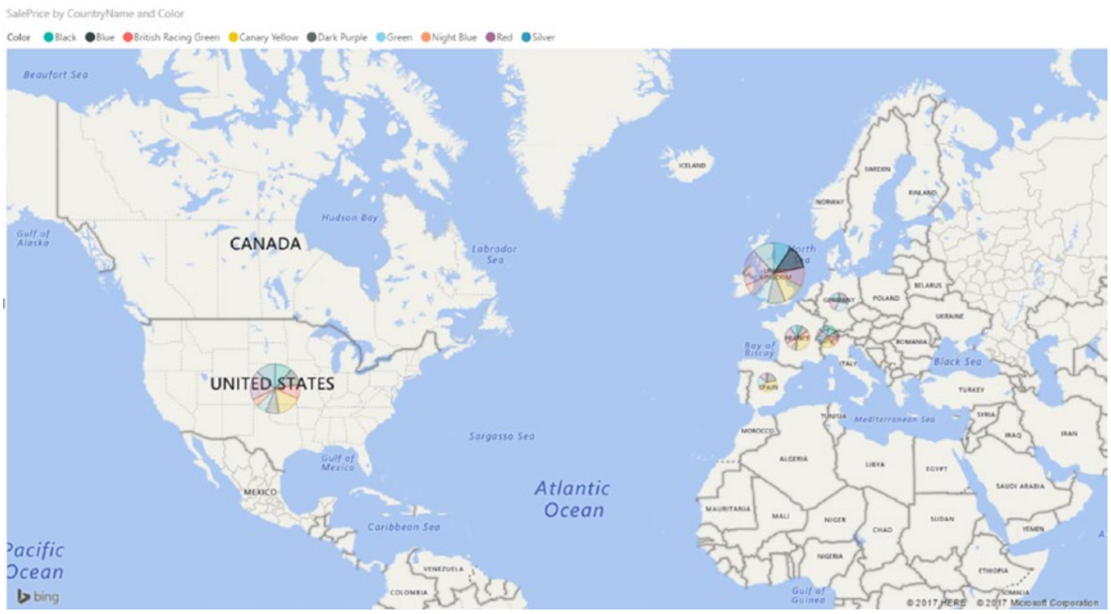

SalePrice by CountryName and Color
Color ● Black ● Blue ● British Racing Green ● Canary Yellow ● Dark Purple ● Green ● Night Blue ● Red ● Silver
CANADA
UNITED STATES
Atlantic Ocean
Pacific Ocean
bing

Figure 19-7. *Highlighting map data*

To remove highlighting from a map, all you have to do is click the legend element that you are using to highlight data, or simply click the title of the legend. Alternatively, click any of the highlighted elements on the map.

This is just a preview of how Power BI Desktop applies highlighting to visuals. You will learn more about this technique in Chapter 21.

Filled Maps

Another way to display data is to fill a geographical area with color rather than to display a bubble. Power BI Desktop lets you do this, too.

1. Click in an empty part of the dashboard canvas in the C:\PowerBiDesktopSamples\CH19\CarSalesDataForReports.pbix file.
2. Click the Filled Map icon in the Visualizations pane. A blank map will appear.
3. Expand the Countries table and drag the CountryName field on to the map visual. The visual will display a map shading in the countries for which there are values.
4. Expand the Stock table and drag the CostPrice field on to the Color Saturation field well. The map will change the color of the shading for each country to reflect the sales value. A darker shade indicates a higher value. The map will now look like Figure 19-8.

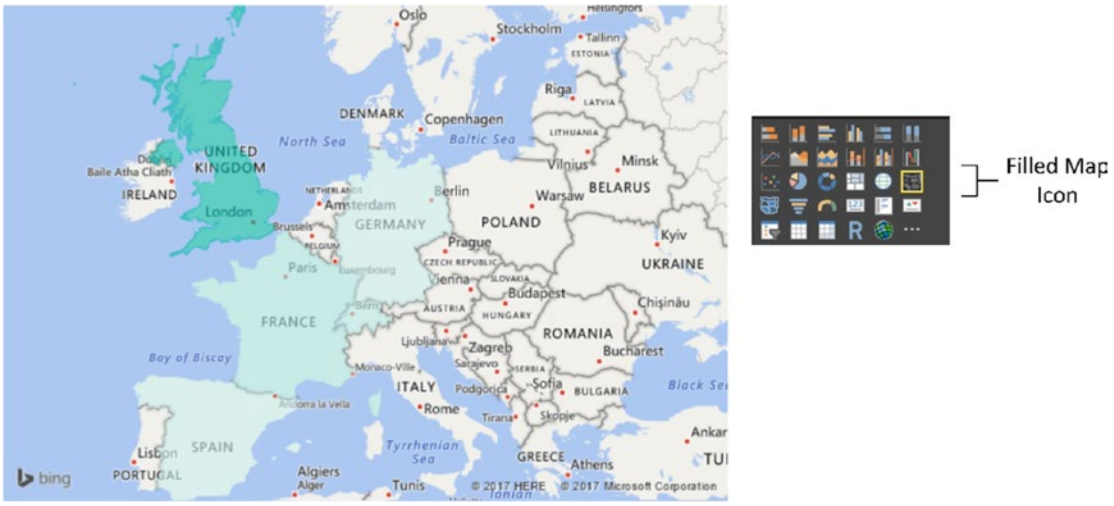


Figure 19-8. *A filled map of costs per country*

Shape Maps

Another way to visualize geographical data is by state or region. This option implies that the source data contains the standard codes that are used to define states and regions. To create a shape map:

1. Click the Shape Map icon in the Visualizations pane.
2. Drag the OuterPostcode field into the Location well (or on to the map).
3. Drag the SalePrice field (from the InvoiceLines table) into the Color Saturation well (or on to the map). The shape map will look like the one in Figure 19-9.

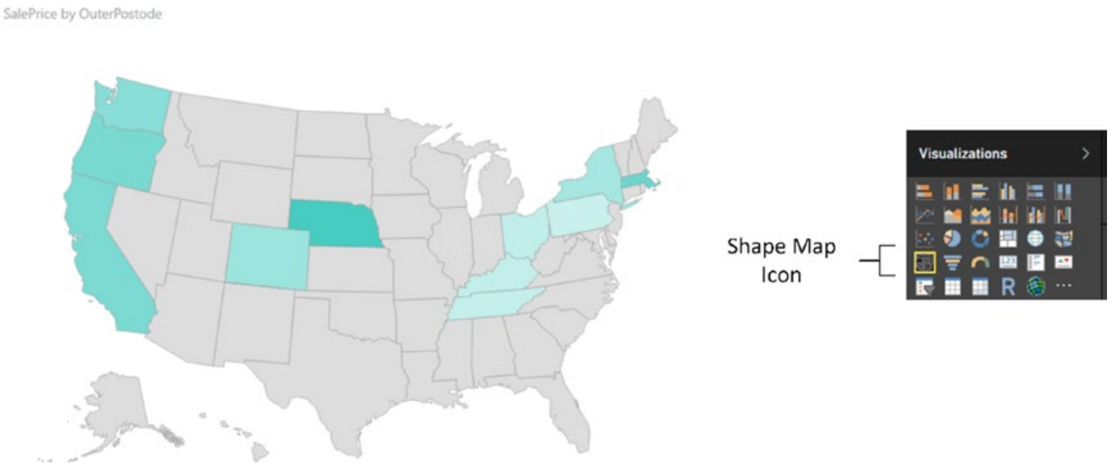
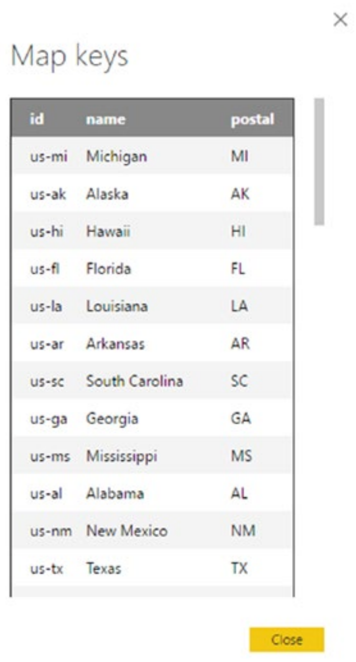


Figure 19-9. *A shape map*

Map Keys

You will probably need to find the map keys (the region codes) that Power BI Desktop uses to display data in a shape map. This way you can add them to your source data or tweak the source data to include the appropriate geographical codes.

1. Select the shape map that you created in Figure 19-9.
2. Click the Format icon in the Visualizations pane.
3. Expand the Shape section.
4. Select the map (the country) whose codes you want to display.
5. Click View Map Keys. A dialog containing the map keys will be displayed, as you can see in Figure 19-10.



id	name	postal
us-mi	Michigan	MI
us-ak	Alaska	AK
us-hi	Hawaii	HI
us-fl	Florida	FL
us-la	Louisiana	LA
us-ar	Arkansas	AR
us-sc	South Carolina	SC
us-ga	Georgia	GA
us-ms	Mississippi	MS
us-al	Alabama	AL
us-nm	New Mexico	NM
us-tx	Texas	TX

Figure 19-10. *Map keys for a specific country*

This is the list of the data elements that Power BI Desktop will use to allocate data to a geographical area. It follows that you will need to have this data—or to be prepared to add it—in your source data. So at least one column in the source data must correspond to one of the values shown in Figure 19-10.

Adding Shape Maps

If you have specific map shapes (and this can include everything from towns to floor plans), then you can load these by clicking the Add Map button in the Shape section of the Formatting pane for shape maps.

Shape map definitions must be created in a specific .json format. Unfortunately, looking at creating these is outside the scope of this book.

Formatting Maps

As you would probably expect, there are numerous formatting options that you can apply to maps. Indeed, the way that you can format maps is virtually identical to the way that you apply formatting to other visuals. Given that the five previous chapters covered many of these options in detail, I will only explain the possibilities in this section, and refer you back to Chapters 14 through 18 if you want all the details.

As far as maps and filled maps are concerned, you can modify the following:

- *Title*: This includes displaying or hiding the title, as well as setting the title text, font color, and background color.
- *Background*: This covers applying a background, setting its color, and defining its transparency.
- *Lock Aspect*: If you set Lock Aspect to On, then you can resize the map while keeping it sized proportionally.
- *Border*: You can add or remove a border for a map just as you can with just about any Power BI Desktop visual.

All map types allow you to adjust the data colors. However, the way that the colors are modified is slightly different, depending on the map type. We will now look at the individual formatting options by map type.

Maps

Map visuals allow you to adjust the following aspects of their formatting:

- Data Colors
- Category Labels
- Bubbles
- Map Controls
- Map Styles

Some of these elements are similar to formatting options that you have seen before, so I will not re-explain them here. However, other elements of formatting are handled either in completely new ways or in a manner so different from how a similar format option is dealt with in other types of visual that they need a fresh explanation.

Data Colors

Selecting a data color for a map allows you to select the color that will be used for the data bubble.

■ **Note** You cannot apply different colors to the various data points in a map.

Category Labels

If you need to supply the name of the geographical element that is currently displayed in a map (the town or the country, say), then you can set the Category Labels button to On. If you expand the Category Labels section, then you can also choose the font family, size, and color for the category labels.

Bubbles

Expanding the Bubbles section allows you to adjust the Size slider to set the relative size of the bubbles. You can see an example of this in Figure 19-11.

Map Controls

There is currently only one map control that can be tweaked, and that is the auto zoom function. The default (Auto Zoom set to On) means that filtering the map data will cause only relevant geographical areas to appear in the map. If you switch this option to Off, then the map will display the same area until you alter this manually.

Map Styles

There are only two map styles in the version of Power BI Desktop that was available as this book went to press:

- Road (the default)
- Aerial

You can see the difference that selecting an aerial view makes in Figure 19-11 (where you can also see the effect of tweaking the bubble size and adding category labels).

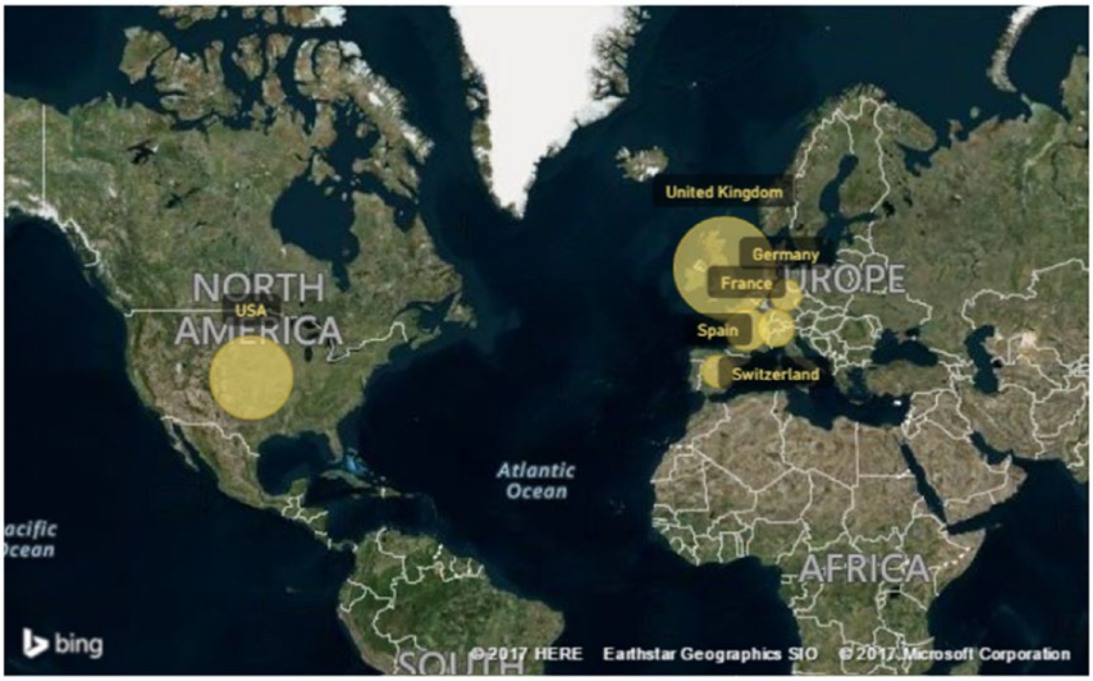


Figure 19-11. *A formatted map*

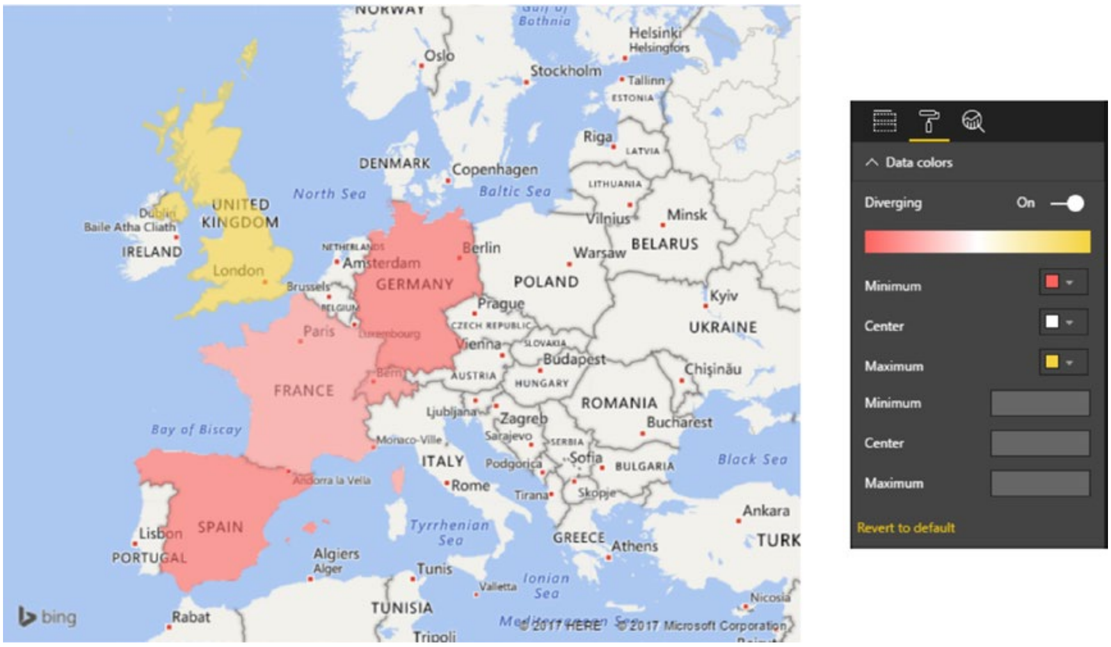
Filled Maps

Filled maps offer two formatting options over and above those that you have already seen for other visuals:

- Data Colors
- Map Controls

As the Map Controls section only gives you the Auto Zoom selection that I described in the "Maps" section, the option that interests us is Data Colors. This can help you to differentiate geographical areas by expanding or compressing the range of colors that are applied to a filled map. This, too, is best understood with the aid of an example:

1. Re-create the map from the start of this chapter.
2. Click the Format icon in the Visualizations pane.
3. Expand the Data Colors section.
4. Set Diverging to On.
5. Select a color for the Minimum, Center, and Maximum. You can see the result of this tweak in Figure 19-12.



Data colors

Diverging On

Minimum 🟥

Center ⬜

Maximum 🟨

Minimum

Center

Maximum

Revert to default

Figure 19-12. *Formatting a filled map*

■ **Note** You do not have to set the minimum and maximum values for the shading that is applied, but doing this lets you control how the range of shading is applied to the data.

Tooltips

Just as was the case for charts, you can add extra data fields to the Tooltips well in the Visualizations pane. Then, when you hover the pointer over a map data point (which could be a country in a shaded map like the one in Figure 19-12), you will see the extra data information for the map point in the tooltip.

Shape Maps

The essential thing that you can change in a shape map is the color—or colors—of the actual shapes. The current version of Power BI Desktop makes this a little confusing, but hopefully things will have been made easier by the time that you read this. Essentially you have two options:

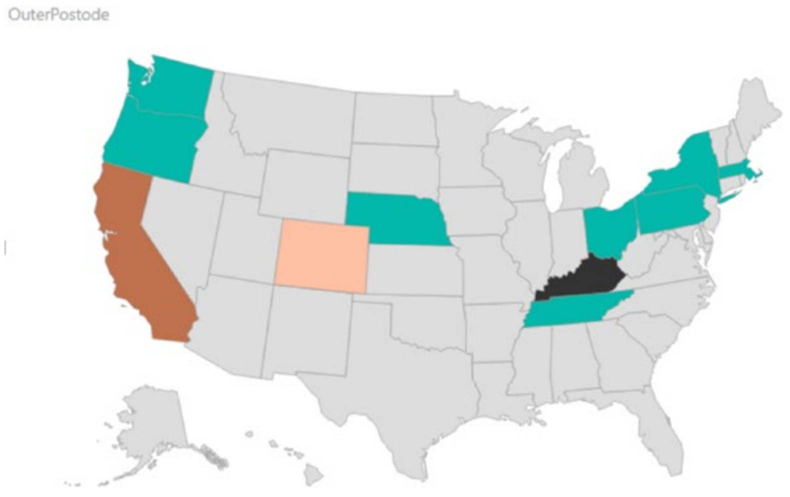
- Select a specific color for each geographical area in a shape map. This is used for illustration, and does not require anything other than geographical references for the map.
- Set a color range for each geographical area for which there is data.

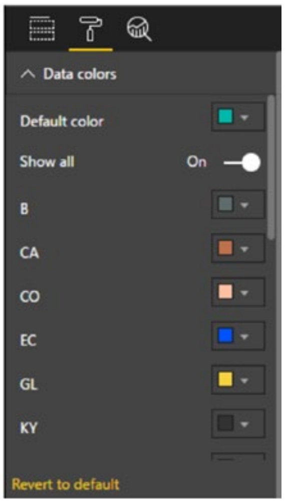
As well, you can select a color for any parts of the map for which there is no data.

Specific Data Colors

To set the data for each specific geographical area for which there is data in a shape map:

1. Create a new shape map.
2. Add the Outer Postcode field from the Clients table to the Location well in the Visualizations pane.
3. Click the Format icon and expand the Data Colors section at the top.
4. Set Show All to On.
5. Choose a color for each of the data areas. The map should look like the one in Figure 19-13.




Data colors

Default color 🟩

Show all On

B ⬛

CA 🟫

CO 🟧

EC 🟦

GL 🟨

KY ⬛

Revert to default

Figure 19-13. *A shape map will geographical shading*

Defining a Color Range for Data

If a shape map contains numeric data, then you can set the colors for each geographical zone using a diverging color range—much as you did in the case of a filled map. Here is how:

1. Create a shape map by adding Outer Postcode field (from the Clients table) to the Location well in the Visualizations pane, and the SalePrice field (from the InvoiceLines table) to the Color Saturation well.
2. Click the Format icon and expand the Data Colors section (in the middle of the Formatting pane).
3. Select a color for the Minimum, Center, and Maximum.
4. Expand the Default Color section.
5. Set Show to On.
6. Select a default color from the Color palette.

7. Select a border color from the Border Color palette. The map will now look like the one in Figure 19-14.

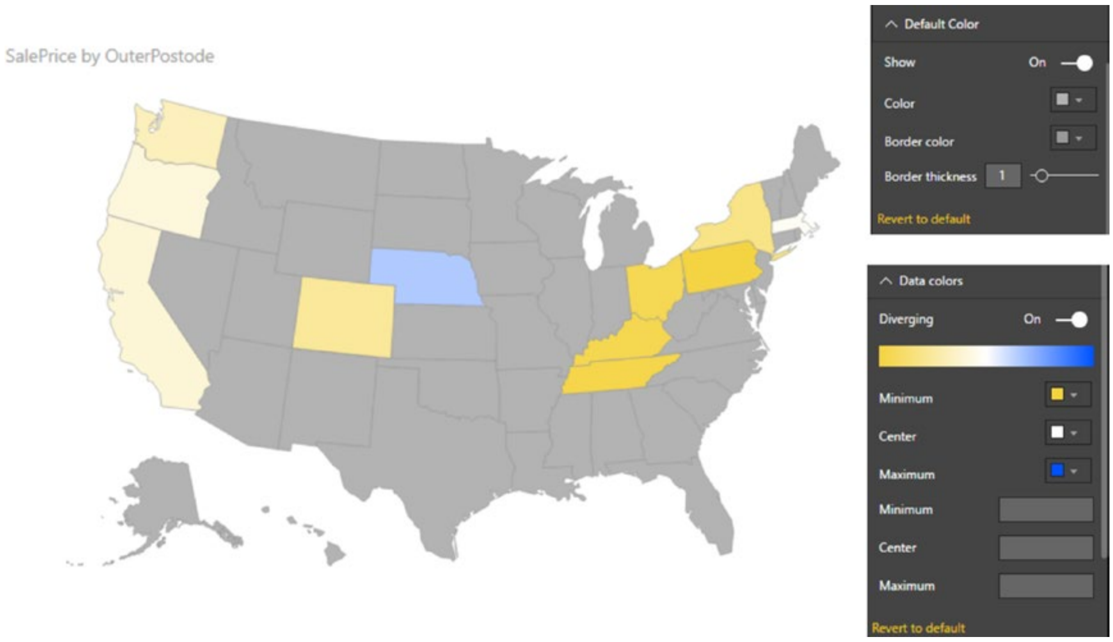


Figure 19-14. *A shaded filled map*

ARCGis Maps

Microsoft has collaborated with ESRI (an international supplier of geographic information system [GIS] software) to include ARCGis mapping in Power BI Desktop. Quite simply, this feature allows you to take geospatial representation of data to a whole new level. Using ARCGis maps, you can

- Perform spatial analysis by adding heat maps and aggregating data.
- Add demographic and reference data.
- Make any map look great.

Creating an ARCGis Map

In this section you will see a couple of examples of how using ARCGis maps can add real pizzazz to your dashboards.

1. Open the file C:\PowerBiDesktopSamples\CH19\CarSalesDataForReports.pbix.
2. In the Visualizations pane, click the ArcGIS maps for Power BI icon. A blank ARCGis map will be added to the report canvas.
3. Add the following fields to the ARCGis map:
    a. Town (from the Clients table)
    b. SalePrice (from the InvoiceLines table)
4. Enlarge the visual to get a clearer view. The ARCGis map will look like the one in Figure 19-15.

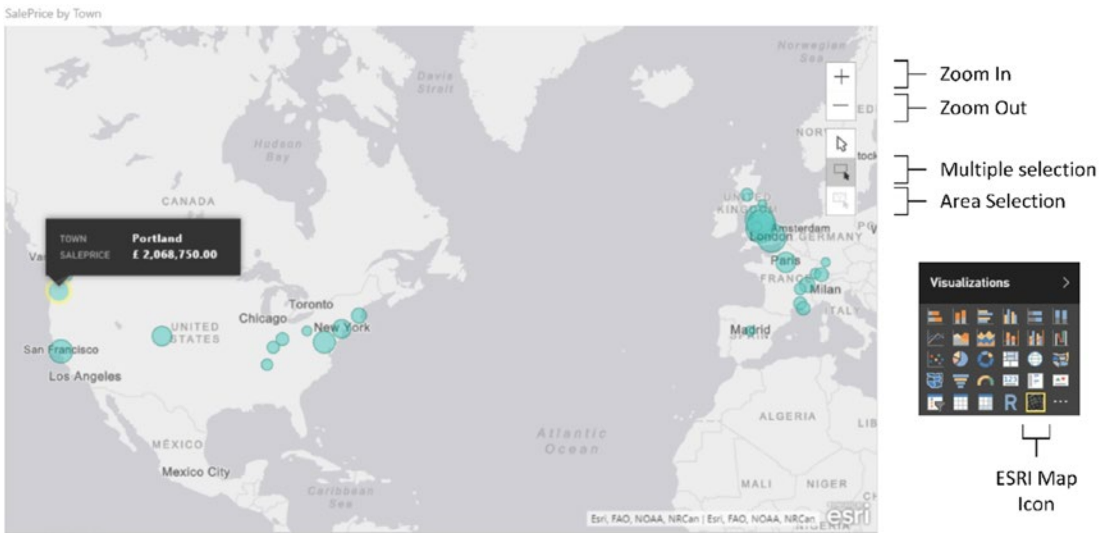


Figure 19-15. *An initial ARCGis map*

So far so easy-and in all honesty, this is not so very different from what you have done already using the traditional Power BI Desktop mapping tools. However, things are about to get much more interesting as you learn to use the added options in ARCGis maps.

■ **Note** You can move the display area in an ARCGis map or zoom in and out just as you can for any other map in Power BI Desktop.

Selecting a Basemap Type

Once you have created an ARCGis map, you can extend it with some powerful effects. One option is to change the appearance of the map itself:

1. In the ARCGis map, click the popup menu at the top right (the ellipses).
2. Select Edit from the context menu. The map will expand to full screen and add the mapping toolbar at the top of the map.
3. Click Basemap at the left of the toolbar.
4. Select a basemap from the palette of available types. The map display will change to reflect your choice, as you can see in Figure 19-16.

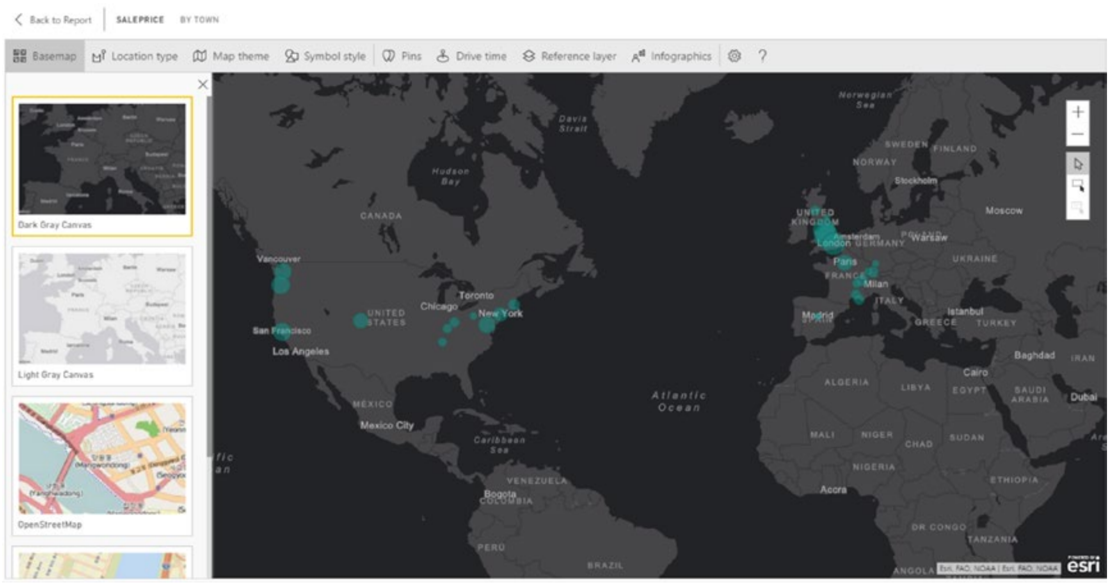


Figure 19-16. *Changing the Basemap type of an ARCGis map*

5. Click Back to Report at the top left of the ARCGis map to return to the report canvas.

■ **Note** To close the palette of basemaps and display only the map itself, simply click the small cross at the top right of the basemap type palette. This will, of course, only apply to ARCGis editing mode.

Selecting a Location Type

You can choose how map data is displayed using the Location Type option. This lets you decide between displaying:

- Data as points
- Data as boundaries (areas)

If you select that locations will be represented as boundaries, you can then select exactly which boundaries will be displayed. The type of boundary will depend on the selected country. Table 19-1 shows a few of the available boundary definitions that are available.

Table 19-1. *Boundary Types*

Location Type	Country	Description
Entire Country	USA	Takes the entire USA as the map boundary.
States	USA	Uses each state as a geographical element.
DMAs	USA	Uses each DMA as a geographical element.
Congressional Districts	USA	Uses each congressional district as a geographical element.
CBSAs	USA	Uses each core-based statistical area as a geographical element.
Counties	USA	Uses each county as a geographical element.
County Subdivisions	USA	Uses each part of a county as a geographical element.
ZIP Codes	USA	Uses each ZIP (postal) code as a geographical point.
Cities and Towns	USA	Uses individual towns and cities as a geographical point.
Census Tracts	USA	Uses each census area as a geographical element.
Block Groups	USA	Uses each Census Block Group as a geographical boundary.
Country	UK	Uses each of the four countries of the UK as a geographical boundary.
Regions	UK	Uses each region as a geographical element.
Postcode Areas	UK	Uses each postcode area as a geographical element.
Counties	UK	Uses each county as a geographical boundary.
Districts	UK	Uses each district as a geographical element.
Postcode districts	UK	Uses each postcode district (outer postcode) as a geographical element.
Postcode sectors	UK	Uses each postcode sector (inner postcode) as a geographical element.
Census Areas	UK	Uses each census area as a geographical element.

■ **Note** For boundary display to be effective, the underlying data must contain the data that allows ARCGis maps to identify the boundary.

Adding a Map Theme

Map themes let you change the way that geographical data is displayed to powerful effect. As an example, suppose that you want to display the sales data as a heat map:

1. Edit the ARCGis map that you created previously.
2. Select Map Theme from the menu buttons. A palette of available options appears on the left.
3. Select Heat Map.

You can see the results of selecting this option in Figure 19-17.

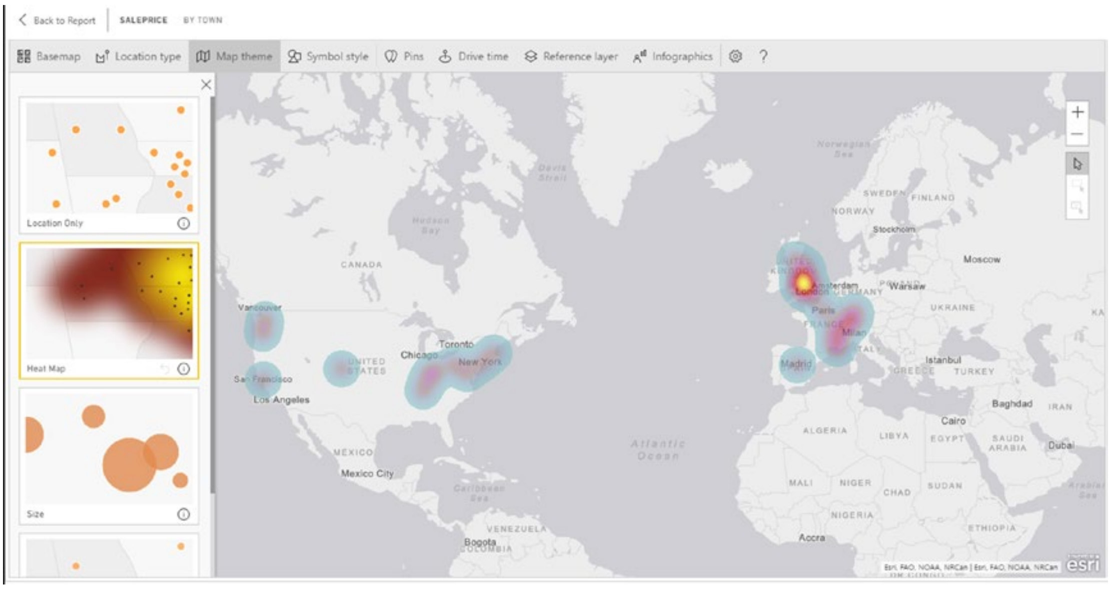


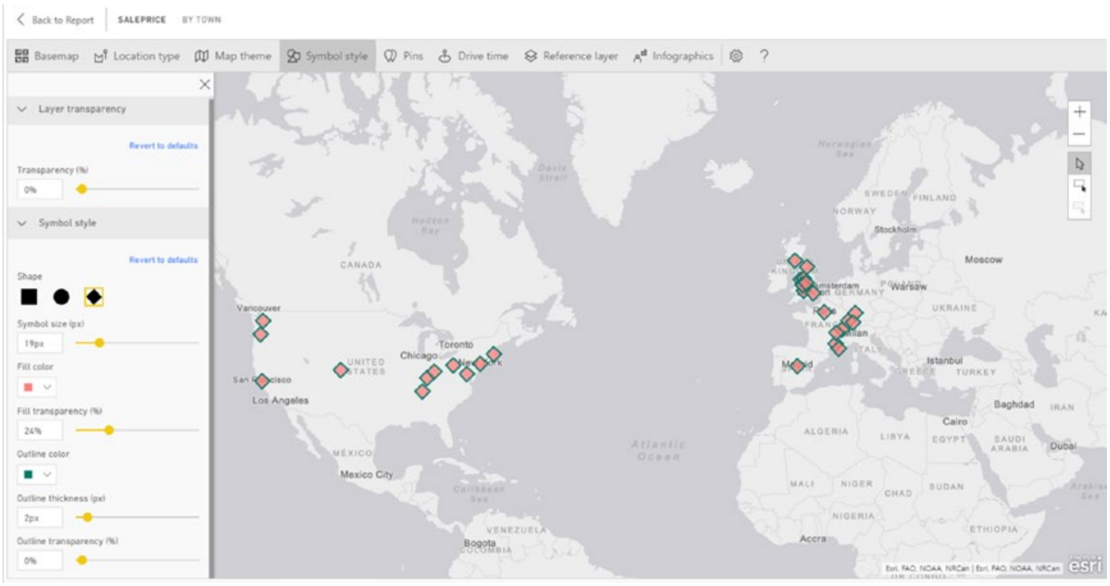
Figure 19-17. A heat map applied to an ARCGis map

Choosing a Symbol Type

ARCGis maps also give you a variety of ways of displaying data points on a map. You can see this in the following example:

1. Re-create the ARCGis map that you first saw in Figure 19-15 at the start of this section.
2. Edit the map.
3. Select Symbol Style from the menu buttons.
4. Under Symbol Style on the left, select the diamond symbol.
5. Increase the symbol size by dragging the Symbol Size slider to the right.
6. Select a symbol fill color from the palette of available colors.

7. Tweak the outline color and thickness (and the various transparency settings) if you wish. You will see something like the map shown in Figure 19-18.



Layer transparency

Revert to defaults

Transparency (%) 0%

Symbol style

Revert to defaults

Shape ■ ● ◆

Symbol size (px) 19px

Fill color 🟥

Fill transparency (%) 24%

Outline color 🟩

Outline thickness (px) 2px

Outline transparency (%) 0%

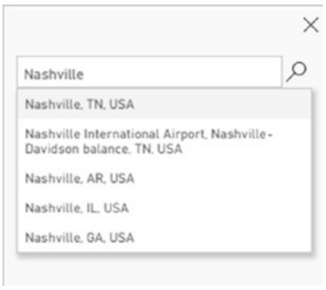
Figure 19-18. *An ARCGis map with formatted symbols*

Adding Pins

An original way to highlight specific points on a map is to add a pin. This is used to highlight a specific geographical location independently of the underlying data. You do this using the Pins menu button of the ARCGis Maps toolbar as follows:

1. In an existing ARCGis map visual, click the Pins menu button of the ARCGis Maps toolbar. The locations pane will appear on the left.

2. Enter the location to search for in the search field. A list of possible locations will appear, as shown in Figure 19-19.



Nashville

Nashville, TN, USA

Nashville International Airport, Nashville-Davidson balance, TN, USA

Nashville, AR, USA

Nashville, IL, USA

Nashville, GA, USA

Figure 19-19. *Selecting a pin location*

3. Click the precise location that you wish to highlight. An example of the output is shown in Figure 19-20.

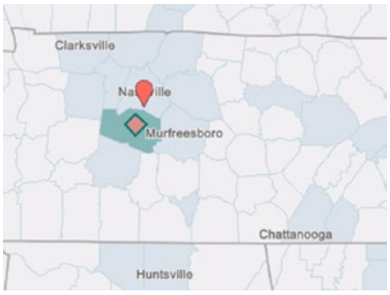


Figure 19-20. *A pin added to an ARCGis map*

To remove a pin, simply click Delete for the pin that you want to remove from the map in the locations pane.

■ **Tip** You can add multiple individual pins to an ARCGis map.

Adding a Reference Layer

ARCGis maps allow you to access certain statistical geographical datasets and add them to maps. You could, for instance, add the median American home value data to a map and see how (if at all) that relates to car sales:

1. Re-create the initial ARCGis map that you used at the start of this section.
2. Edit the ARCGis map.
3. Click the Reference Layer button on the ARCGis Maps toolbar.

4. Select the 2016 USA Median Home Value reference layer from the reference layer pane. You can see this in Figure 19-21.

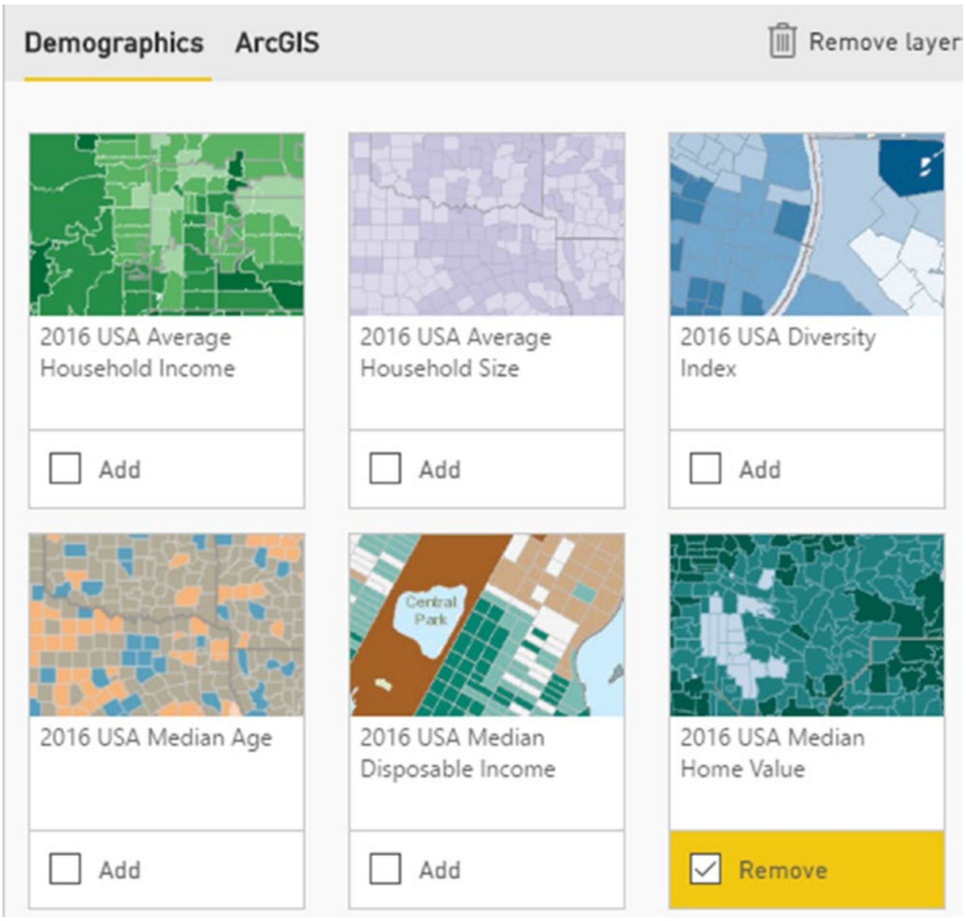

Demographics
ArcGIS
Remove layer
2016 USA Average Household Income
Add
2016 USA Average Household Size
Add
2016 USA Diversity Index
Add
2016 USA Median Age
Add
Central Park
2016 USA Median Disposable Income
Add
2016 USA Median Home Value
Remove

Figure 19-21. *Some of the available reference layers in ARCGis maps*

5. Adjust the map display to zoom in on the United States. You can see the resulting map with reference data in Figure 19-22.

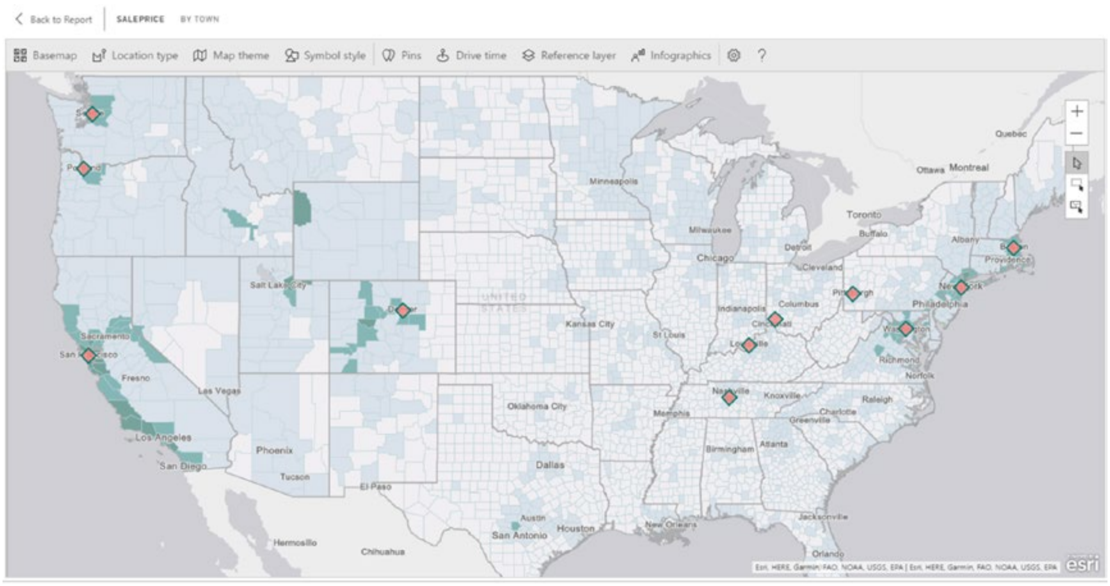


***Figure 19-22.** A USA map with reference data added*

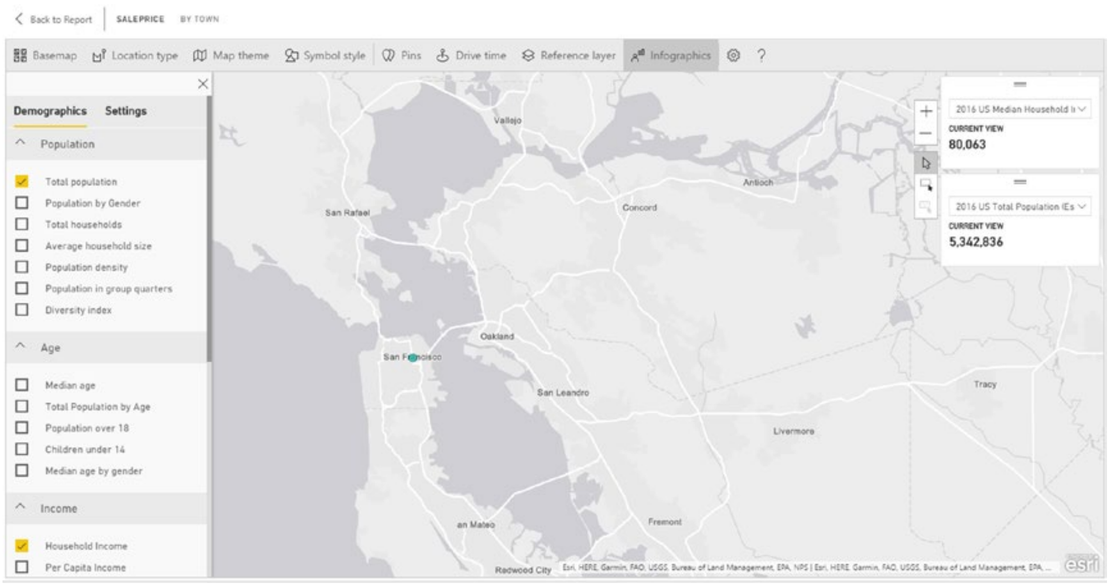
■ **Tip** You can only add one layer of reference data at a time.

Adding Infographics

ARCGis maps also come with access to data relating to demographics. This data is limited to North America for the moment. You can add reference data like this:

1. Re-create and then edit the initial ARCGis map that you used at the start of this section.
2. Click the Infographics button on the ARCGis Maps toolbar.
3. Select the following options:
   a. Total Population
   b. Household Income

4. Zoom in to the San Francisco area on the map. The reference data is adjusted to correspond to the area that is displayed. You can see this in Figure 19-23.



Demographics Settings

Population

- [✓] Total population
- [ ] Population by Gender
- [ ] Total households
- [ ] Average household size
- [ ] Population density
- [ ] Population in group quarters
- [ ] Diversity index

Age

- [ ] Median age
- [ ] Total Population by Age
- [ ] Population over 18
- [ ] Children under 14
- [ ] Median age by gender

Income

- [✓] Household Income
- [ ] Per Capita Income

2016 US Median Household I
CURRENT VIEW
80,063

2016 US Total Population (Es
CURRENT VIEW
5,342,836

Figure 19-23. *ARCGis reference data*

Conclusion

In this chapter, you saw how to display geographical data in a variety of map types, from filled maps, where the data is represented by the intensity of an area's shading, to bubble maps, where the size of data points on a map lets the data speak for itself. You also saw how to add truly expressive ARCGis map visuals to your reports, where you were able to add extra geographical data to enhance your analysis.

Over the course of six chapters, you have met a wide range of visuals that let you express the trends hidden in your data. This chapter built on the five previous chapters, where you learned all about text-based visuals, charts, tree maps, gauges, and custom visuals, by adding maps to the mix.

CHAPTER 20

Filtering Data

Power BI Desktop is built from the ground up to enable you, the user, to sift through mounds of facts and figures so that you can deliver meaningful insights. Consequently, what matters is being able to delve into data and highlight the information it contains quickly and accurately. This way, you can always explore a new idea or simply follow your intuitions without needing either to apply complex processes or to struggle with an impenetrable interface. After all, Power BI Desktop is there to help you come up with new analyses that could give your business an edge on the competition.

Power BI Desktop provides two main approaches to assist you in focusing on the key elements of your data:

- *Filters*: Restrict the data displayed in a report, page, or visual. This is the subject of the current chapter.
- *Interactive data selection*: Allows you to highlight key information instantly and visually for an audience. We will look at these aspects in the next chapter.

The people who developed Power BI Desktop recognize that your data is the key to delivering accurate analysis. This is why you can filter on any field or set of fields in the underlying data model to extract its real value. This approach is not only intuitive and easy, it is also extremely fast, which ensures that you almost never have to wait for results to be returned.

You can add filters before, after, or during the creation of a Power BI Desktop report. If you add filters before creating a table, then your table will only display the data that the filter allows through. If you add a filter to an existing report, then the data visualization will alter before your eyes to reflect the new filter. If you modify a filter when you have visualizations on a Power BI Desktop report, then (as you probably guessed by now) all the visualizations affected by the filter will also be updated to reflect the new filter criteria—instantaneously.

You can filter any type of data:

- Text
- Numeric values
- Dates
- Logical values

Each data type has its own ways of selecting elements and setting (where possible) ranges of values that can be included—or excluded. This chapter will explain the various techniques for isolating only the data that you want to display. You will then be able to create Power BI Desktop reports based only on the data that you want them to show.

We will see all how these approaches work in detail in the rest of this chapter. In any case—and as is so often the case with Power BI Desktop—it is easier to grasp these ideas by seeing them in practice than by talking about them, so let's see how tiles, slicers, and highlighting work. This chapter uses the C:\PowerBiDesktopSamples\CH20\CarSalesDataForReports.pbix sample file as the basis for all the filters that you will learn to apply.

Filters

Subsetting data in Power BI Desktop is based on the correct application of filters. Consequently, the first thing that you need to know about filters is that they work at three levels. You have

- Report-level filters
- Page-level filters
- Visualization-level filters

The characteristics of these three kinds of filter are described in Table 20-1.

Table 20-1. *Power BI Desktop Filters*

Filter Type	Application	Comments
Report-level	Applies to every visualization in the current report.	Filters data for every visual in the current file.
Page-level	Applies to every visualization in the active page.	Filters data for every visual in the current page.
Visual-level	Only applies to the selected visualization.	Applies only to the selected visual (table, chart, etc.).

Filters are always applied in exactly the same way. What matters is the extent that they affect the visuals in a Power BI Desktop presentation. In practice, this makes your life much easier because you only have to learn how to apply a filter once and then you can use it in the same way at different levels in separate files.

Let's now look at how to use filters, beginning at the lowest level of the filter hierarchy: visual-level filters.

Visual-Level Filters

Saying that there are three types of filter available in Power BI Desktop is a purely descriptive distinction. For Power BI Desktop, any filter is a filter, and all filters work in the same way. However, as there is a clear hierarchy in their application, I will begin with visual-level filters and then move on to their ascendants—page-level filters and report-level filters. Given the general similarity between the three, it is probably worth noting that it is important you ensure that you are creating or modifying the appropriate filter. As this is not always obvious, not least when you are starting out with Power BI Desktop, I will try to make it clear as we proceed how exactly you can distinguish at what level you are applying a filter, as the effects can have wide-ranging consequences for the message that you are trying to convey.

The Filters Well

The first thing to note is that all filters are applied in the Filters well. This is in the lower part of the Visualizations pane, and looks (for a selected sample visual) like Figure 20-1.

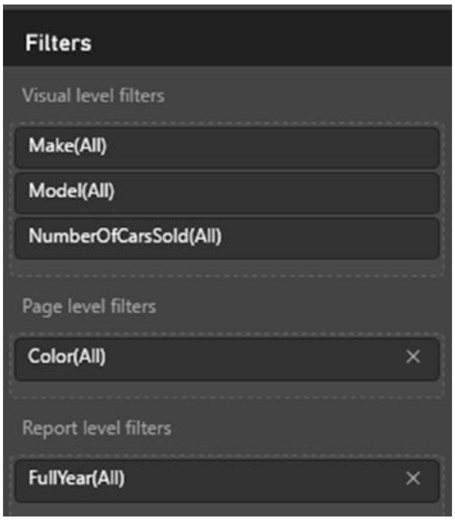


Figure 20-1. *The Filters well*

As you can see in this figure, different filters have been applied at all the possible levels of the filter hierarchy. If you look closely, you can also see that each filter gives an indication of how (if at all) the filter is applied. Since no filtering has been applied here, each filter shows All as the current selection. This tells you that all data is allowed through for the filter. You will learn how to add and adjust filters in the following few pages.

■ **Note** The Filters well only shows visual-level filters if a visual is selected.

Adding Filters

The Filters well automatically adds any fields that you use as the basis for a visualization. To see this, create the following visual:

1. Open the C:\PowerBiDesktopSamplesCH20\CarSalesDataForReports.pbix file.
2. Create a clustered bar chart using these fields:
   a. Make (from the Stock table)
   b. SalePrice (from the InvoiceLines table)

The Filters well will look like Figure 20-2.

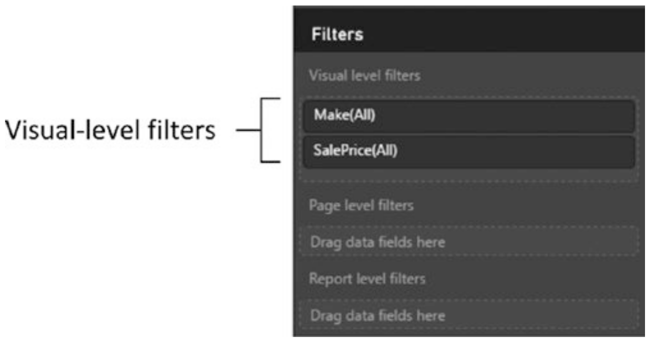


Figure 20-2. *Automatic creation of visual-level filters*

Figure 20-2 shows that adding a data field to a visual automatically adds the same field to the Visual Level Filters well.

Applying Filters

To see the filter working, let's limit the chart to displaying only a few makes of vehicle:

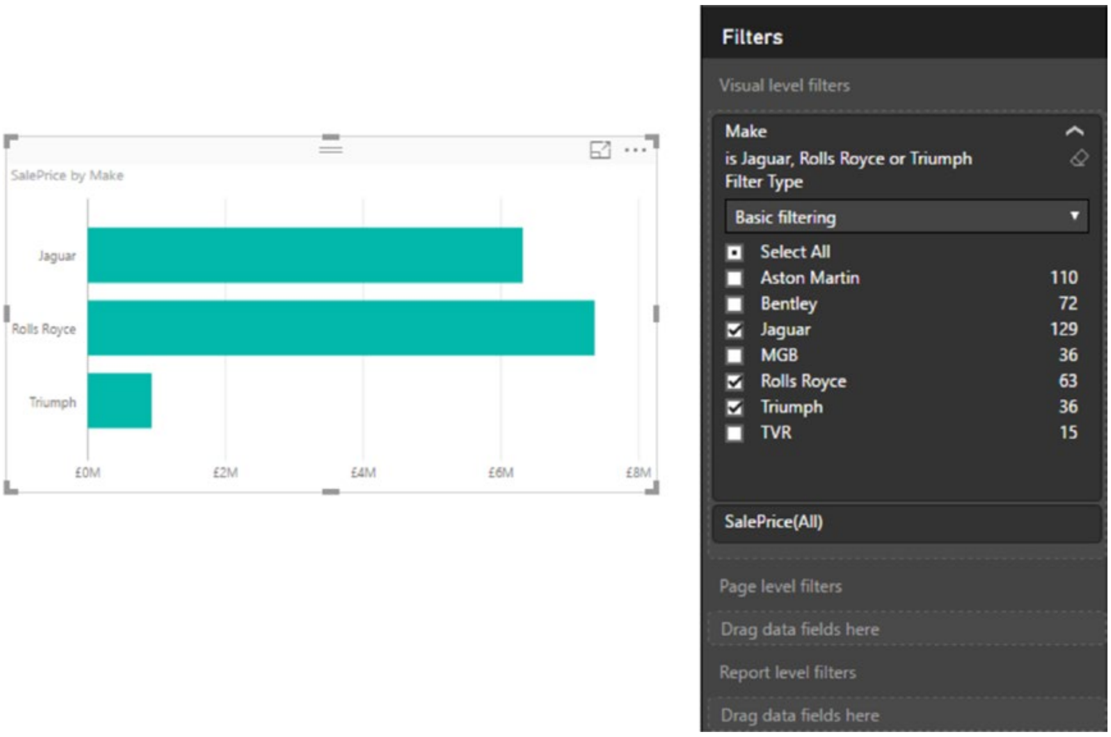
1. Select the clustered bar chart that you created previously.
2. In the Visualizations pane, hover the pointer over the Make filter in the Filters well. A downward-facing chevron will appear at the right of the field that is being used as a filter. This will be a visual-level filter.
3. Expand the Make filter by clicking the chevron. The Filters well will display all the makes that appear in the visual.
4. Select the following makes in the Filters well by selecting the check box to the left of each of the following elements:

    a. Jaguar

    b. Rolls-Royce

    c. Triumph

You will see that data is only displayed for the makes of car that were selected in the filter. The resulting chart should look like Figure 20-3.



Make
is Jaguar, Rolls Royce or Triumph
Filter Type
Basic filtering

■	Select All	
☐	Aston Martin	110
☐	Bentley	72
✓	Jaguar	129
☐	MGB	36
✓	Rolls Royce	63
✓	Triumph	36
☐	TVR	15

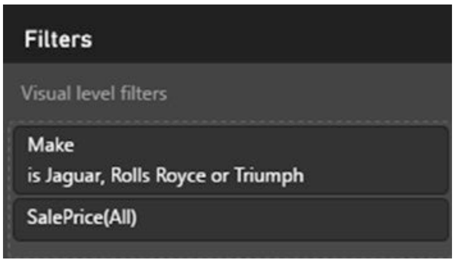
SalePrice(All)

Figure 20-3. *A simple filtered chart*

You will have noticed that when the filter was first applied, every check box was empty, including the Select All check box. The default is (fairly logically) to set up a filter ready for tweaking, but not actually to filter any data until the user has decided what filters to apply. Once you start adding filter elements, they will be displayed in the Filters well just below the name of the field that is being used to filter data.

You modify filters the same way you apply them. All you have to do to remove a selected filter element is to click the check box with a check mark to clear it. Conversely, to add a supplementary filter element, just click a blank check box.

One final thing to note is that if you subsequently minimize the filter for a field (by clicking the upward-facing chevron that has replaced the downward-facing chevron to the right of the field name), you will now see not only the field name in the Filters well, but also a succinct description of the filter that has been applied. You can see an example of this in Figure 20-4.


Filters

Visual level filters

Make
is Jaguar, Rolls Royce or Triumph

SalePrice(All)

Figure 20-4. *Filter description*

■ **Note** You can also add fields to the Filters well before adding data fields to the visual. Simply create a blank visual and then leave it selected while you add the filter fields to the Filters well.

The Select All Filter

The only subtlety concerning simple filters is that you also have the Select All check box. This acts as a global on/off switch to select, or deselect, all the available filter elements for a given filter field. The Select All filter field has three states:

- *Blank*: No filters are selected for this field.
- *Checked*: All filters are selected for this field.
- *Dotted*: Some filters are selected for this field.

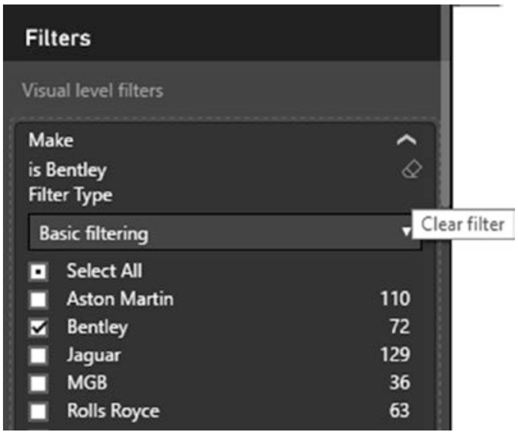
Checking or unchecking the Select All filter field will select or deselect all filter elements for this field, in effect rendering the filter inactive. The Select All filter field is particularly useful when you want not only to remove multiple filter selections in order to start over but also want to select all elements in order to deselect certain elements individually (and avoid manually selecting reams of elements).

■ **Note** When selecting multiple elements in filter lists, you may be tempted to apply the classic Windows keyboard shortcuts that you may be in the habit of using in, for instance, Excel or other Windows applications. Unfortunately, Ctrl- or Shift-clicking to select a subset of elements does not work. Neither can you select and deselect a check box using the space bar. It is not possible to use the cursor keys to pass from one element to another in a filter list either.

Clearing Filters

Setting up a finely honed filter so that you are drilling through the noise in your data to the core information can take some practice. Fortunately, the virtually instantaneous application of filters means that you can see almost immediately if you are heading down the right path in your analysis. However, there are frequent occasions when you want to start over and remove any settings for a particular filter. This can be done in one of two ways. The first is a single step:

1. Click the small eraser icon to the right of the filter, as shown in Figure 20-5.



■	Select All	
☐	Aston Martin	110
☑	Bentley	72
☐	Jaguar	129
☐	MGB	36
☐	Rolls Royce	63

Figure 20-5. *The Clear Filter icon*

Alternatively, you can do the following, but *only* for basic filters:

1. Expand the filter in the Filters well by clicking the downward-facing chevron.
2. Click the Select All option to remove all filter selections.
3. Minimize the filter.

Once a filter has been cleared, the only way to get it back to its previous state is to press Ctrl+Z immediately. Otherwise, you will have to reapply the requisite criteria.

Filtering Different Data Types

So far, you have only seen how Power BI Desktop can filter text elements. Although text-based elements are a major part of many data filters, they are far from the only available type of data. There are also

- Numeric data
- Date and time data
- Logical (true/false) data

Filtering Numeric Data

You can filter on numeric elements just as you can filter on text-based elements in Power BI Desktop. However, when filtering on numbers you likely want to select ranges of numbers as precise figures.

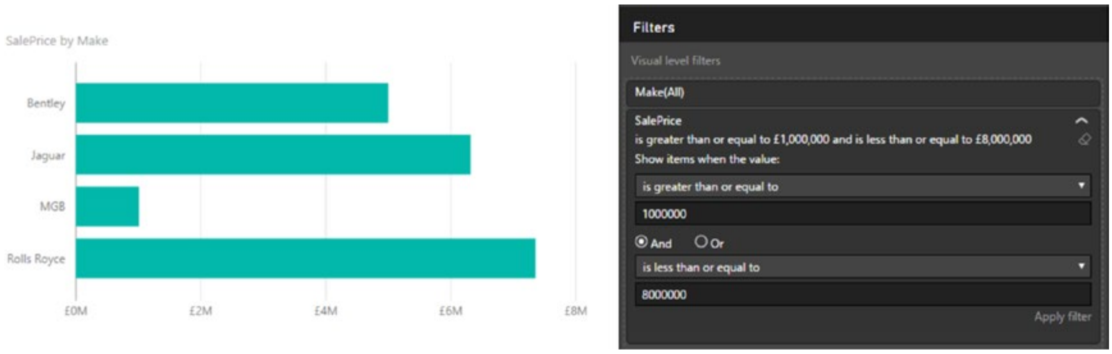
Range Filter Mode

The first trick worth knowing is that, when filtering on numeric data, the default option is to use a *threshold selector*, which is the only filter for numeric filters. The threshold selector allows you to set the lower and/or upper limits of the range of numbers that you want to display in a Power BI Desktop report, page, or visual.

The following explains how to set the range of figures for which data is displayed:

1. Select the chart that you created previously.
2. Clear any existing filters.
3. In the Visualizations pane, hover the pointer over the SalePrice filter in the Filters well. A downward-facing chevron will appear at the right of the field that is being used as a filter.
4. Expand the SalesPrice filter by clicking the chevron. The Filters well will display two popups that will enable you to set thresholds.
5. Click the upper popup and select the "Is greater than or equal to" option.
6. In the box under the upper popup, enter the value **1000000** (without any thousands separators or currency units).
7. Ensure that the And radio button is selected under the popup.
8. Click the lower popup and select the "Is less than or equal to" option.
9. In the box under the lower popup, enter the value **8000000** (without any thousands separators or currency units).
10. Click Apply Filter. The chart will change to show only values in the range that you set in the filter.

That is it. You have set a range for all data in the Power BI Desktop report corresponding to the selected field. It should look like Figure 20-6. It is worth noting that in this figure I have widened the Filters well so that you can see the entire text that Power BI Desktop adds to a range filter to explain what the filter does. If you leave the Visualizations pane at its default width, you will only see an abbreviated version of the filter definition.



SalePrice
is greater than or equal to £1,000,000 and is less than or equal to £8,000,000
Show items when the value:
is greater than or equal to
1000000
[✓ And] Or
is less than or equal to
8000000

Figure 20-6. *A filter range*

When selecting a range of numeric data, you do not have to set both upper and lower bounds. You may set either one or both.

■ **Note** Should you want to select a precise number from those available in a dataset, you can switch a numeric filter's filter type to basic filtering. This will show all the individual numbers in the selected field, as well as the number of times that each value occurs.

Numeric Filter Options

Numbers cannot be filtered exactly the same ways as text; the filtering options are slightly different. These filter options are described in Table 20-2.

Table 20-2. *Numeric Filter Options*

Filter Option	Description
Is Less Than	The selected field is less than the number you are searching for.
Is Less Than Or Equal To	The selected field is less than or equal to the number you are searching for.
Is Greater Than	The selected field is greater than the number you are searching for.
Is Greater Than Or Equal To	The selected field is greater than or equal to the number you are searching for.
Is	The selected field matches exactly the number you are searching for.
Is Not	The selected field does not exactly match the number you are searching for.
Is Blank	The selected field is blank.
Is Not Blank	The selected field is not blank.

If you are using both threshold levels to define a range of values to include or exclude, or even specific values to include or exclude, then you need to apply one of the logical filter options. These are shown in Table 20-3.

Table 20-3. *Logical Filter Options*

Filter Type	Description
And	Applies both filter elements to *reduce* the amount of data allowed through the filter.
Or	Applies either of the filter elements separately to *increase* the amount of data allowed through the filter.

When applying a numeric filter, you must—not altogether surprisingly—enter a numeric value. If you enter text by mistake, you will see a yellow lozenge appear at the right of the box to alert you to the fact that you entered a text by mistake, and the Apply filter text will remain grayed out.

In this case, you have to delete the characters that you entered and enter a numeric value in the place of the text.

Filtering Date and Time Data

At its simplest, date and time data is merely a list of elements or a range of numeric data. Consequently, dragging a field from a date dimension into the Filters well allows you to select one or more elements (such as years or months) or to define a range (of weeks, for instance). Take the following steps to see this in action:

1. Select the chart that you created earlier using the Make and SalePrice fields.
2. Clear any existing filters.
3. Expand the DateDimension table in the Fields list.
4. Drag the FullYear field to the Visual Level Filters well. Since Power BI Desktop assumes that the years are numbers, it switches to advanced filtering.
5. Select Basic Filtering from the Filter Type popup list to revert to a list of years.
6. Click one or two years in the filter list. Figure 20-7 demonstrates this. The chart will be updated to show only data for the chosen years.

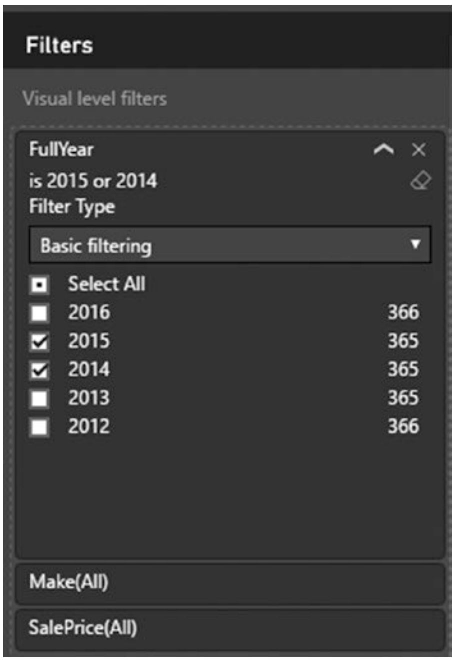


Figure 20-7. *A date filter*

One of the reasons that I created a date dimension is to allow you to filter on date elements this simply. So you can now use any of the fields in the DateDimension table to restrict the data that is in a visualization. In my opinion, the following are very useful fields in this dimension:

- FullYear
- MonthAndYearAbbr
- QuarterAndYear
- YearAndWeek

However, you need to remember that you can combine multiple elements in a filter to get the correct result. So there is nothing to stop you from filtering on a specific year, and then adding the day of the week and calculating all the sales for the Saturdays in the year, for instance. By combining different filter elements from a properly constructed data dimension, you can look at how data varies over time incredibly easily.

This means that, when filtering by dates, you could need to apply multiple filters, where you can select elements from each of the different filters: Year, Quarter, and/or Month. Alternatively, if you will be filtering on successive elements in a date hierarchy (Year, followed by Month, for instance) you may find it more intuitive to drag the filter elements from the date hierarchy to the Filters well in the temporal order in which you will be using them (that is, Year followed by Month). This way, you can proceed in a logical manner, from top to bottom in the Filters well, to apply the date criteria that interest you.

■ **Note** Whatever the data type, you cannot add hierarchies to a filter. Instead, you must expand the hierarchy in the Fields list, and drag each element of a hierarchy into the required filters well.

Date and Time Filters

If you are filtering on a Date or DateTime field, then you quickly notice that Power BI Desktop adapts the filter to help you select dates and times more easily. In essence, Power BI Desktop lets you select from four ways of filtering date and time data:

- Select one or more exact dates or times (basic filtering)
- Define a range of dates or times (advanced filtering)
- Specify a range of dates relative to the current day (relative date filtering)
- Select a top few dates (top N filtering)

Moreover, the filter well for a date adds:

- A calendar popup that lets you click a day of the month (and scroll through the months of the year, forward and backward)
- A time series scroll filter that lets you select times to every minute throughout the day

To see this in action, imagine that you want to see all sales for a range of dates:

1. Select the chart that you created previously.
2. Clear all filters from it, as described earlier.
3. Leaving the chart selected, expand the DateDimension table and drag the DateKey field into the Visual Level Filters well. You will see a list of all the dates in the DateDimension table in the filter.
4. Click Advanced Filtering.
5. From the "Show items when the value" popup list, select "is on or after."
6. Click the calendar icon beneath the popup. The calendar popup is shown in Figure 20-8.

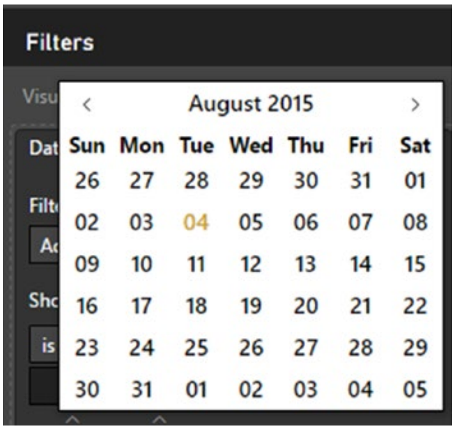


Figure 20-8. *The calendar popup*

7. Select a date from the calendar. This date will appear in the box under the calendar icon.
8. Click the And radio button.
9. In the second popup, select "is on or before."
10. Click the calendar icon beneath the popup and select a date from the calendar. This date will appear in the box under the second popup.

11. Click Apply Filter. The Filters well will look like Figure 20-9.

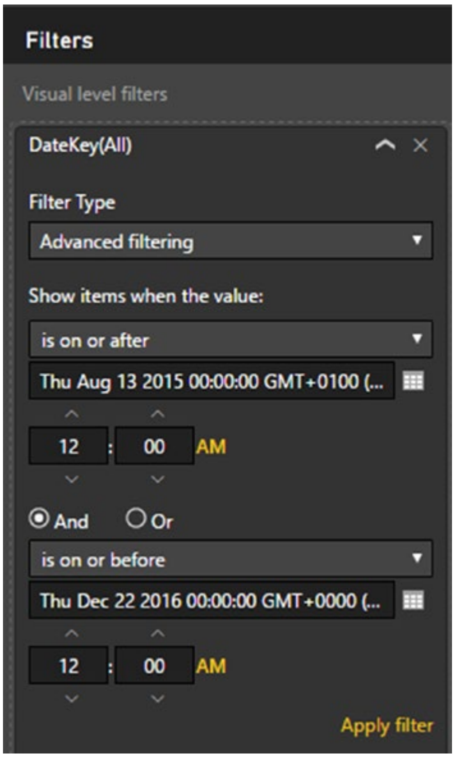

Filters
Visual level filters
DateKey(All)
Filter Type
Advanced filtering
Show items when the value:
is on or after
Thu Aug 13 2015 00:00:00 GMT+0100 (...
12 : 00 AM
(●) And ( ) Or
is on or before
Thu Dec 22 2016 00:00:00 GMT+0000 (...
12 : 00 AM
Apply filter

Figure 20-9. *A date range filter*

There are a couple of tricks that may save you time when you are selecting dates from the calendar popup (you may be familiar with these techniques in other desktop packages):

- When using the calendar popup, clicking the right-facing chevron to the right of the month and year displays the following month.
- When using the calendar popup, clicking the left-facing chevron to the left of the month and year displays the previous month.

- When using the calendar popup, clicking the month and year displays a Year popup, in which you can click the right-facing chevron to the right of the year to display the following year (or click the left-facing chevron to the left of the year to display the previous year), and then you can select the month from those displayed. You can see this in Figure 20-10.

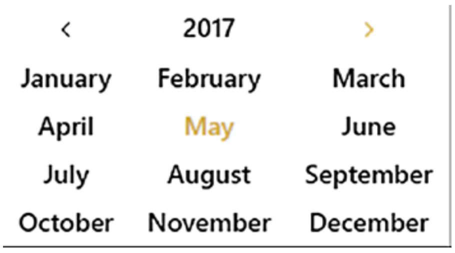


Figure 20-10. *The calendar popup for months*

- If you have already clicked the month and year and are looking at the months for a year, you can click the year to see ranges of years. You can use the left- and right-facing chevrons to scroll through previous and successive ranges of years, then select the year that you want to filter on. You can see this in Figure 20-11.

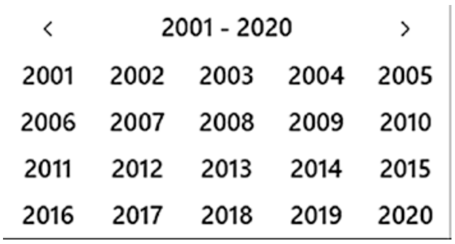


Figure 20-11. *The calendar popup for years*

- When using the time popup, clicking inside any constituent part of the time (hour, minute, or second) and then clicking the up and down scroll triangles above and below the time field allows you to scroll rapidly through the available options.
- Clicking AM or PM to the right of the time box lets you switch from AM to PM.

If you do not want to select a date using the calendar popup, then you can enter a date directly in the date box of the advanced filter for a Date (or DateTime) field. Just remember that you must enter the date in the date format corresponding to the environment that you are using and that can be understood by Power BI Desktop.

■ **Note** If you enter a date where the format does not correspond to the system format, or if the date is purely and simply invalid (the 30th of February, for instance), then Power BI Desktop will not let you apply the filter. To correct this, merely select a correct date using the calendar popup. Similarly, if you enter a nonexistent time, then Power BI Desktop will refuse to accept it and will revert to the previous (acceptable) time that was chosen.

Date Filter Options

Dates cannot be filtered in exactly the same ways as text or numbers. Consequently, the advanced filtering options for date filters are slightly different from those used when filtering other data types. They are described in Table 20-4.

Table 20-4. *Advanced Date Filter Options*

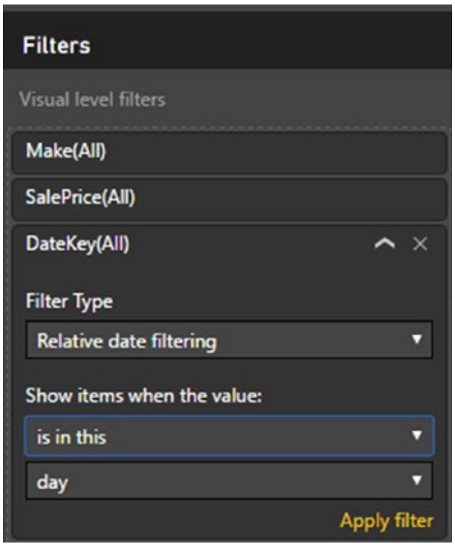
Filter Option	Description
Is	The selected field contains the date that you are searching for.
Is Not	The selected field does not contain the date that you are searching for.
Is After	The selected field contains dates after the date that you entered; that is, later dates that do not include the date you entered.
Is On Or After	The selected field contains dates beginning with the date that you entered or later.
Is Before	The selected field contains dates before the date that you entered; that is, earlier dates, not including the date you entered.
Is On Or Before	The selected field contains dates on or before the date that you entered; that is, earlier dates, up to and including the date you entered.
Is Blank	The selected field is blank.
Is Not Blank	The selected field is not blank.

Relative Date Filtering

Another way of finding data for a range of dates applies specifically to the current date. Power BI Desktop can help you find, for instance, sales in the past days, weeks, months, or even years. Here is how:

1. Select the chart that you created previously.
2. Clear all filters from it, as described earlier.
3. Leaving the chart selected, expand the DateDimension table and drag the DateKey field into the Visual Level Filters well.
4. In the Filter Type popup list, select Relative Date Filtering.
5. In the popup for “Show items when the value,” select “is in this.”

6. In the second popup for “Show items when the value,” select Year. The Filters well will look like the one in Figure 20-12.



Filters

Visual level filters

Make(All)

SalePrice(All)

DateKey(All)

Filter Type: Relative date filtering

Show items when the value: is in this day

Apply filter

Figure 20-12. *Applying a relative date filter*

7. Click Apply Filter to filter the data in the visual.

There are several options available for relative date filtering. These are described in Table 20-5.

Table 20-5. *Relative Date Filter Options*

Filter Option	Filter Definition	Description
Is In The Last	Relative positioning	Takes a range of dates before the current date.
Is In This	Relative positioning	Specifies the current time period to be used.
Is In The Next	Relative positioning	Takes a range of dates after the current date.
Days	Time element	Sets days as the number of time periods.
Weeks	Time element	Sets weeks (rolling seven-day periods) as the number of time periods.
Calendar Weeks	Time element	Sets full calendar weeks as the number of time periods.
Months	Time element	Sets months (rolling periods from date to date) as the number of time periods.
Calendar Months	Time element	Sets full months as the number of time periods.
Years	Time element	Sets years (rolling periods from date to date) as the number of time periods.
Calendar Years	Time element	Sets full years as the number of time periods.

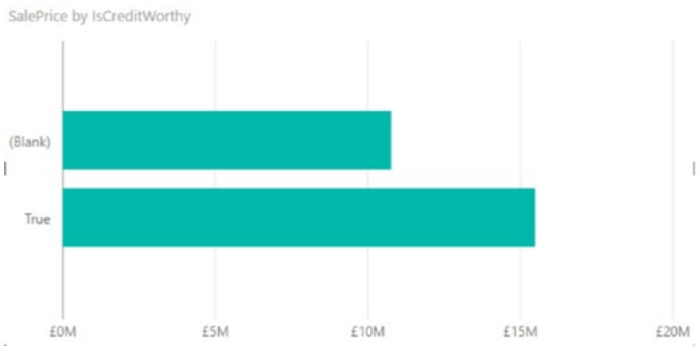
■ **Note** Selecting "Is in the last" or "Is in the next" relative date filter options will display an additional field in the Filters well where you can enter the number of days, weeks, months, etc. to be used to filter the data.

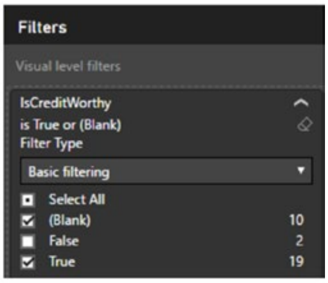
Filtering True or False Data

There are other data types in the source data that you are likely to be handling. You might have Boolean (True or False) data, for instance. However, for Power BI Desktop, this is considered, for all intents and purposes, to be a text-based filter. On the other hand, there are data types that you cannot filter on and that do not ever appear in the Filters well. Binary data (such as images) is a case in point.

So if you filter on Boolean data, Power BI Desktop displays True and False in the expanded filter for this data type. The following explains how to see this:

1. Create a chart using the following fields:
    a. IsCreditWorthy (from the Clients table)
    b. SalePrice (from the InvoiceLines table)
2. Expand the IsCreditWorthy field in the Filters well.
3. Select True and Blank. The chart and Filters well will look like they do in Figure 20-13.





IsCreditWorthy
is True or (Blank)
Filter Type

Basic filtering

☐	Select All	
✓	(Blank)	10
☐	False	2
✓	True	19

Figure 20-13. *Applying a Boolean filter*

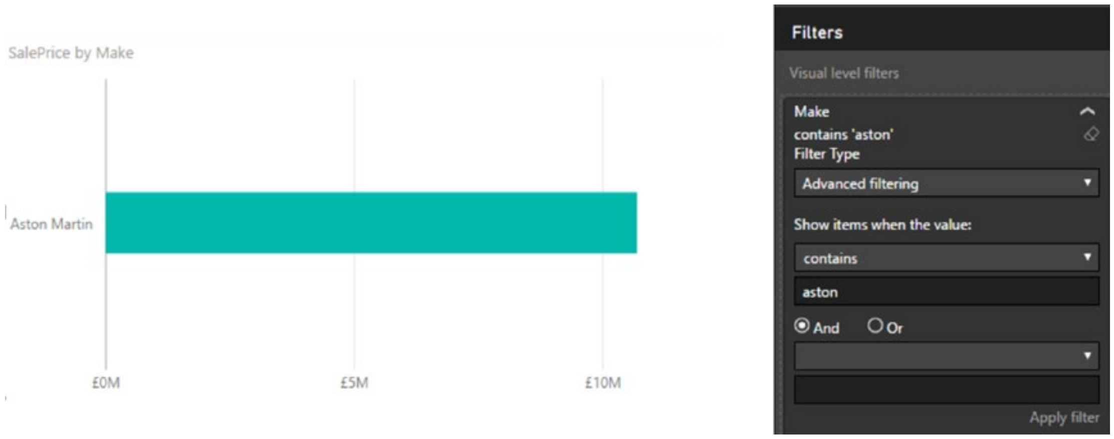
Advanced Text Filters

In many cases, when you are delving into your data, merely selecting a "simple" filter will be enough to highlight the information that interests both you and your audience. There will inevitably be cases when you need to filter your data more finely in order to return the kinds of results that sort the wheat from the chaff. This is where Power BI Desktop's advanced filtering capabilities come to the fore. Advanced filtering lets you search inside field data with much greater precision, and it is of particular use when you need to include, or exclude, data based on parts of a field if it is text.

Applying an Advanced Text Filter

Let's begin with a simple example of how to apply an advanced filter to a text field:

1. Click the visual that you want to filter. Once again, we will use the chart that you created earlier in this chapter using Make and SalePrice as the data fields.
2. Clear any existing filters.
3. Expand the Make field in the Filters well (unless it has already been done).
4. Select Advanced Filtering from the Filter Type popup. The body of the filter switches to show the advanced filter popups and boxes, and the text under the filter title now reads "Show items when the value."
5. Select Contains from the popup.
6. Click inside the filter text box (under the box displaying Contains) and enter the text to filter on (**aston** in this example).
7. Click Apply Filter or press the Enter key. All objects in the Power BI Desktop report will only display data where the client contains the text *aston*. The result is shown in Figure 20-14; the advanced filter used to produce it is also shown.



Filters
Visual level filters
Make
contains 'aston'
Filter Type
Advanced filtering
Show items when the value:
contains
aston
● And ○ Or
Apply filter

***Figure 20-14.** The results of applying an advanced filter*

Here are several comments that it is important to make at this stage:

- Advanced filtering is *not* case sensitive. You can enter uppercase or lowercase characters in the filter box; the result is the same.
- Spaces and punctuation are important, as they are taken literally. For instance, if you enter *A* (with a space after the A), then you only find elements containing an A (uppercase or lowercase) followed by a space.
- Advanced filters, just like standard filters, are cumulative in their effect. So, if you have applied a filter and do not get the results you were expecting, be sure to check that no other filter at another level is active that might be narrowing the data returned beyond what you want.

- If your filter excludes all data from the result set, then any tables in the Power BI Desktop report displays This Table Contains No Rows.
- Similarly, if your filter excludes all data, charts will be empty.

In any case, if you end up displaying no data, or data that does not correspond to what you wanted to show, just clear the filter and start over.

Clearing an Advanced Filter

Inevitably, you will also need to know how to remove an advanced filter. The process is the same as for a standard filter; all you have to do is click the Clear Filter icon at the top of the filter for this field (just under the chevron for the field in the field well). The filter elements are removed for this filter.

Reverting to Basic Filtering

If you decide that you no longer wish to define and use a complex filter, but you wish to revert to basic filtering, then all you have to do is select Basic Filtering from the Filter Type popup list for the selected filter. The Filters well will switch to basic filtering for the selected field.

Text Filter Options

When filtering on the text contained in a data field, you can apply the string you are filtering on to the underlying data in several ways. These are the same for both the upper and lower of the two advanced filter options for a text field. They are described in Table 20-6.

Table 20-6. *Advanced Text Filter Options*

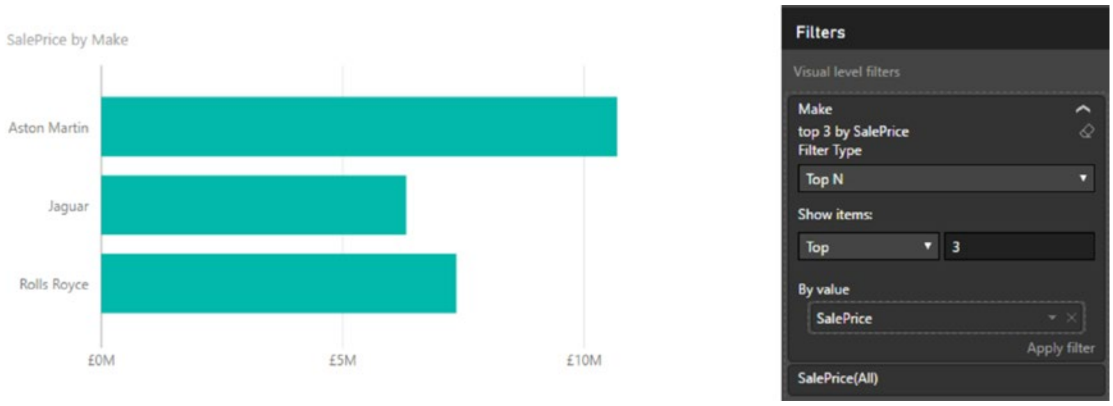
Filter Option	Description
Contains	The selected field contains the search text anywhere in the field data.
Does Not Contain	The selected field does not contain the search text anywhere in the field data.
Starts With	The selected field begins with the search text, followed by any data.
Does Not Start With	The selected field does not begin with the search text, followed by any data.
Is	The selected field matches the search text exactly.
Is Not	The selected field does not match the search text exactly.
Is Blank	The selected field is blank.
Is Not Blank	The selected field is not blank.

You are not limited to setting a single advanced text filter. Just as was the case for numeric values, you can set two filters and apply either of them (by setting the logical operator to Or). Alternatively, you can set two complementary text filters by setting the logical operator to And.

Top N Filtering

One of the reasons to filter data is to extract meaning from the mass of available information. One easy way to deliver meaningful analysis to isolate the best-performing (or worst-performing) elements. Suppose, for instance, that you want to see the top three best-selling makes of car:

1. Click the visual that you want to filter. Once again, we will use the chart that you created earlier in this chapter using Make and SalePrice as the data fields.
2. In the Visualizations pane, expand the Make section of the Visual Level Filters well.
3. Select Top N from the Filter Type popup list.
4. Ensure that Top is selected from the available elements in the Show Items popup list.
5. Enter **3** in the field to the right of the Show Items popup list.
6. Drag the SalePrice field into the By Value field.
7. Click Apply Filter. You will see the chart and filter looking like they do in Figure 20-15.



Make
top 3 by SalePrice
Filter Type: Top N
Show items: Top 3
By value: SalePrice
SalePrice(All)

***Figure 20-15.** Applying a top N filter*

What has happened here is that the SalePrice field has been aggregated for each make in the background. Then the highest three values have been used to filter the data.

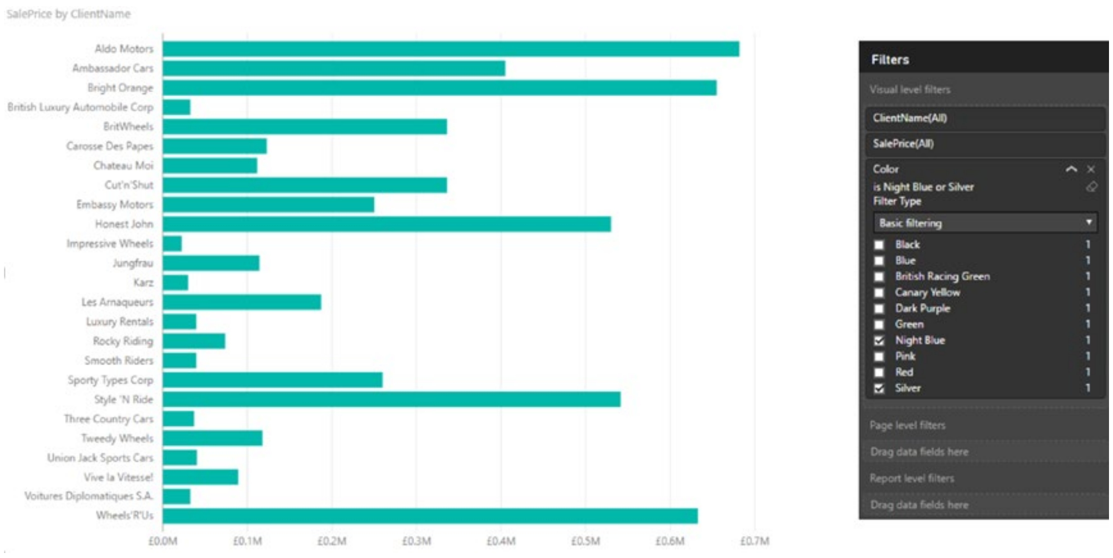
■ **Note** Filtering on the bottom N elements merely means selecting Bottom in the Show Items list.

Specific Visualization-Level Filters

So far in this chapter, we have looked at filters where the fields that were used to filter a visual were *also* visible in the actual visual—be it a chart, map, or table. Inevitably, there will be times when you will want to filter on a field that is *not* displayed in a visual. Although we touched on this earlier, it is worth explaining the concept in greater detail.

The following example explains how to apply a visualization-level filter:

1. Create a clustered bar chart of SalePrice by ClientName.
2. Display the Filters well (unless it is already visible).
3. Expand the Colors table and drag the Color field into the Filters well for the visual-level filters. The list of colors will be expanded automatically in the Filters well. The Color field will *not* be displayed in the visual, however.
4. Select a couple of colors, such as Night Blue and Silver. The result is shown in Figure 20-16.



Filters

Visual level filters

ClientName(All)

SalePrice(All)

Color
is Night Blue or Silver
Filter Type
Basic filtering

☐	Black	1
☐	Blue	1
☐	British Racing Green	1
☐	Canary Yellow	1
☐	Dark Purple	1
☐	Green	1
☑	Night Blue	1
☐	Pink	1
☐	Red	1
☑	Silver	1

Page level filters
Drag data fields here
Report level filters
Drag data fields here

Figure 20-16. *A visualization-level filter without using the data field*

You will notice right away that the filters that you have applied only affect the selected visualization (the chart in this example). When you create complex reports that contain several visualizations, you will see that no other visualizations in the report have their underlying data modified in any way.

You can clear any filter at the visualization level by clicking the Clear icon at the top right of the filter name. Moreover, you can add multiple fields to the Filters well for a visual *and* you can add them in any order.

■ **Note** Removing a field from the field well will not remove this field from the Filters well if the filter is active. This can be deceptive because the field is no longer displayed in the visual; however, its effects can still be seen.

Multiple Filters

So far, we have treated filters as if only one was ever going to be applied at a time. Believe me, when dealing with large and intricate datasets, it is unlikely that this will be the case. As a matter of course, Power BI Desktop will let you add multiple filters to a report. This entails some careful consideration of the following possible repercussions:

- All filters are active at once (unless you have cleared a filter) and their effect is cumulative. That is, data will only be returned if the data matches *all* the criteria set by all the active filters. So, for example, if you have requested data between a specified date range and above a certain sales figure, you will *not* get any data in which the sales figure is lower than the figure that you specified or with a sales date before or after the dates that you set.
- It is easy to forget that filters can be active. Remember that all active filters in the Filters well remain operational whether the Filters well itself is expanded or collapsed. If you are going to collapse filters to make better use of the available space on the screen, then it is worth getting into the habit of looking at the second line below any filter title that will give you a description of the current filter state. It will display something like Contains Rolls. Of course, the exact text varies according to the filter that you have applied.

Page-Level Filters

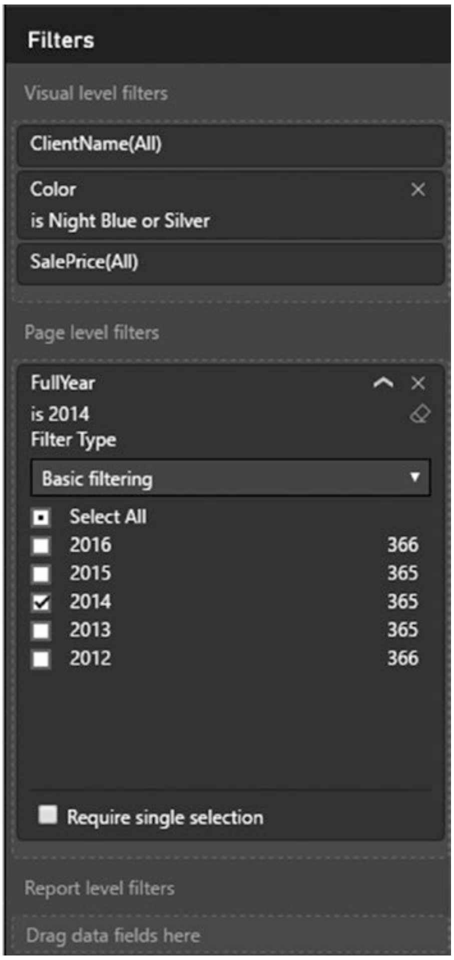
Now that you have seen what filters are and how you can apply them to visuals, it is time to extend the concept and see how filters can be applied to multiple visuals.

The good news is that all filters are configured in exactly the same way whatever the level at which they are applied. Consequently, applying filters at page level or report level is simply a question of choosing where in the Filters well to place a filter.

As an example, suppose that you want to filter all the visuals on a page to display data for a specific year:

1. In an open Power BI Desktop file (such as the one containing the chart that you can see in Figure 20-17), click the dashboard canvas outside any existing visuals.
2. Expand the DateDimension table.
3. Drag the FullYear field onto the field well into the Page Level Filters box. The advanced filtering options for this field will be displayed.
4. Select Basic Filtering as the filter type.
5. Select 2014 from the list of available years. All visuals on the current page will be filtered to display only data for this year.

Figure 20-17 shows you the Filters well for this operation. You can see that a page-level filter looks identical to a visual-level filter. The only difference (apart from the effect that it produces) is the position in the Filters well.



☐	2016	366
☐	2015	365
✓	2014	365
☐	2013	365
☐	2012	366

Figure 20-17. *Applying a page-level filter*

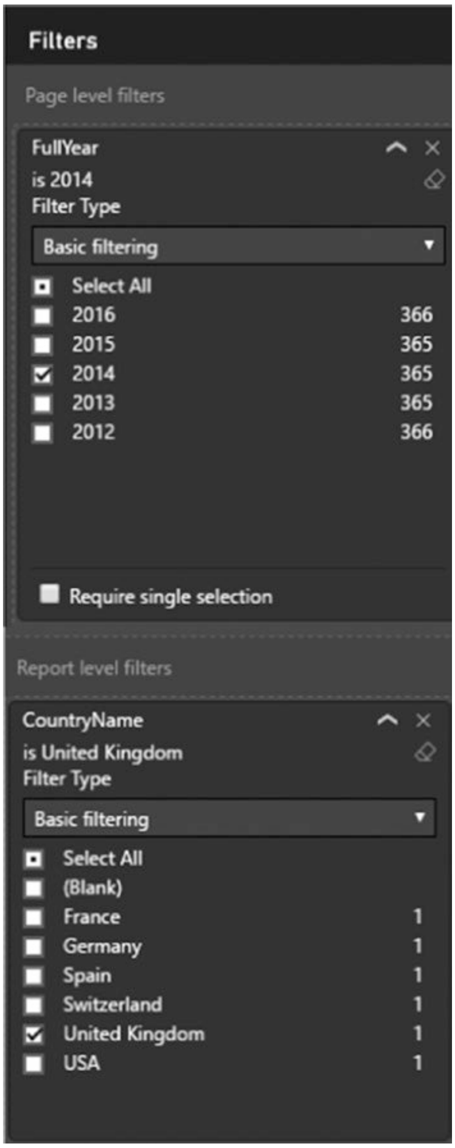
You can add multiple page-level filters if you wish simply by dragging further fields into the field well and into the Page Level Filters box. The order in which the fields are added is unimportant. You will notice that the page-level filters remain visible whether you have selected a visual or not.

Report-Level Filters

The highest level of filtering is applied at report level. This means that any filter set here will apply to every page (or dashboard) in a report and consequently also to every visual in the file. Applying report-level filters is virtually identical to the application of page-level filters. So, let's suppose that your entire report covers a single country. Here is how to set a filter that will apply to every page in the Power BI Desktop report:

1. Take the report that you created with SalePrice by ClientName, and then filtered by color at visual level and FullYear at page level.
2. Click the dashboard canvas outside any existing visuals.
3. Expand the Countries table.
4. Drag the CountryName field onto the field well into the Report Level Filters box. The advanced filtering options for this field will be displayed.
5. Ensure that Basic Filtering is the selected filter type for this filter.
6. Select United Kingdom from the list of available countries. All visuals in the current report (on every page) will be filtered to display only data for this year.

Figure 20-18 shows you the Filters well for this operation.


Filters

Page level filters

FullYear
is 2014
Filter Type
Basic filtering

■	Select All	
☐	2016	366
☐	2015	365
✓	2014	365
☐	2013	365
☐	2012	366

☐ Require single selection

Report level filters

CountryName
is United Kingdom
Filter Type
Basic filtering

■	Select All	
☐	(Blank)	
☐	France	1
☐	Germany	1
☐	Spain	1
☐	Switzerland	1
✓	United Kingdom	1
☐	USA	1

Figure 20-18. *Applying a report-level filter*

You can add multiple report-level filters if you wish simply by dragging further fields into the field well and into the Report Level Filters well. Once again, the order in which the fields are added is irrelevant. You will notice that the report-level filters remain visible whether you have selected a visual or not.

Removing Filters

When working with filters, at times you may want to clear the decks and start over. The fastest way to do this is to delete a filter; once a filter is deleted, it no longer has any effect on the data in the Power BI Desktop report. This can be done as follows:

1. Click the Remove Filter icon to the right of the selected filter. This icon is shown in Figure 20-19.

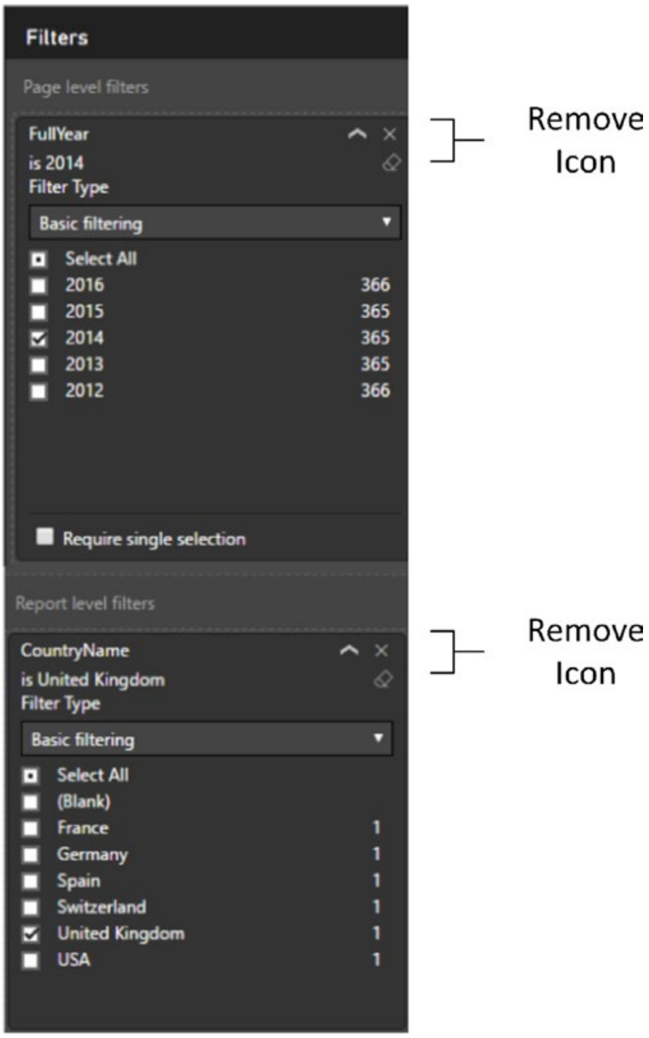


Figure 20-19. *Removing a filter*

The art here is to ensure that you have selected the correct filter to remove. The technique, however, is the same for all report- and page-level filters, as well as for visual-level filters that are not used to display data.

Once a filter has been removed, the only way to get it back is to click Undo (or press Ctrl+Z) immediately; otherwise, you will have to rebuild it from scratch. Interestingly, although you can add filters by dragging elements into the Filters well, you cannot drag them out of the Filters well to remove them.

However, you can remove visualization-level fields (that are not used as data for the visual) from the Filters well by clicking the cross at the right of the field name.

■ **Note** Visual-level filters cannot be removed, only reset to empty.

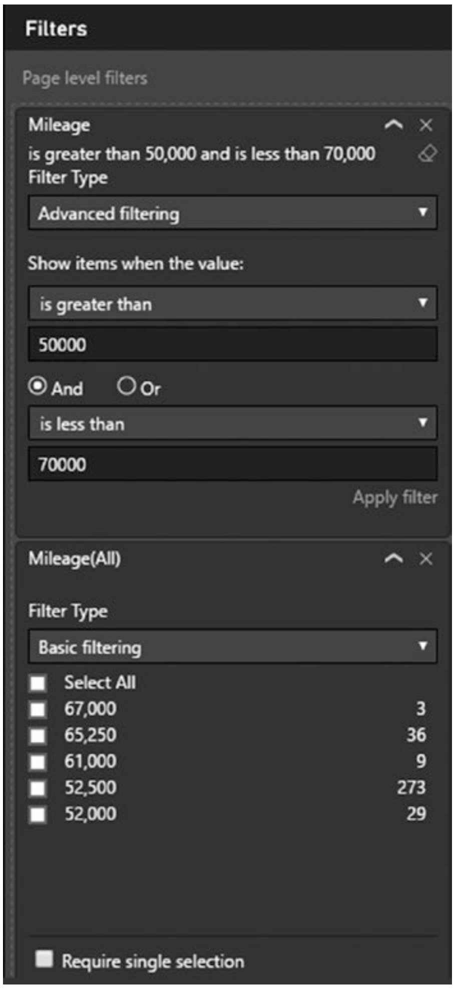
Filter Field Reuse

Although it may seem counterintuitive, you can reuse the same field at the same filter level to assist you in certain cases.

As an example of this, imagine that you want to see all the mileage for vehicles between clearly defined thresholds. Take a look at the following example:

1. Take the report that you created with SalePrice by ClientName, and then filtered.
2. Remove all filters.
3. Expand the Stock table.
4. Drag the Mileage field into the field well and place it in the Page box.
5. Select “is greater than” from the first popup.
6. Enter **50000** in the box for the first threshold.
7. Select “is less than” from the second popup.
8. Enter **70000** in the box for the first threshold.
9. Click Apply Filter.
10. Drag the Mileage field into the field well and place it in the Page box under the filter that you just created.

11. Ensure that Basic Filtering is selected for this filter. A list of available mileage for vehicles in stock (as well as the number of cars for each mileage figure) will appear. You can see this in Figure 20-20.


Filters

Page level filters

Mileage
is greater than 50,000 and is less than 70,000
Filter Type: Advanced filtering
Show items when the value:
is greater than
50000
[✓ And] Or
is less than
70000
Apply filter

Mileage(All)
Filter Type: Basic filtering

Select All	
67,000	3
65,250	36
61,000	9
52,500	273
52,000	29

Require single selection

Figure 20-20. Combining multiple iterations of the same field in a filter

You can now use the more detailed filter to filter the visuals on the current page.

The interesting thing to note is that a hierarchy of filters is applied, even inside a filter box in the Filters well. Put simply, a filter that is placed above another filter will filter the available elements in the lower filter. This only applies, however, to reuse of the same filter.

■ **Note** This example was set to filter at page level. It could equally well have been applied at report level.

Requiring Single Selection

Some filter choices will allow you to select multiple options from a list. You may want to force either yourself or an end user to select only one element from this list.

In such cases, all you have to do is check the Require Single Selection check box under the list. This will deselect any existing selected elements from the list both once this option is active and (possibly more importantly) when a new selection is made in the filter.

■ **Note** Choosing the Require Single Selection option will also hide the Filter Type popup list. To switch to a different filter type, you will have to de-activate the Require Single Selection option by unchecking the check box.

Using the Filter Hierarchy

Given the multiple levels of filters that can be applied, a hierarchy of filters is applied in Power BI Desktop:

- First, at the data level, any selections or choices you apply to the underlying data restrict the dataset that Power BI Desktop can use to visualize your information.
- Second, at the report level, any report-level filters that you apply affect all visualizations in the report using the (possibly limited) available source data.
- Third, at the page level, any page-level filters that you apply will affect all visualizations on the view, using the (possibly limited) available source data filtered by any report-level filters.
- Finally, for each visualization, any visualization-level filters that you apply will further limit the data that is allowed through the report- and page-level filters—but only for the selected visualization.

It is worth noting the following points:

- You have no way to apply a completely different selection to a visualization filter if it has been filtered out at a higher (report or page) level. Clicking Select All will *only* select from the subset of previously filtered elements.
- If you apply a filter at visualization level and then reapply the same filter at report or page level, but with different elements selected, you will still be excluding all nonselected elements from the filter at visualization level. I stress this because Power BI Desktop will remember the previously selected elements at visualization level, and leave them visible even if they cannot be used in a filter, because they have already been excluded from the visualization-level filter by being ruled out at view level. In my opinion, this adds a certain visual confusion, even if the hierarchical selection logic is applied.

Hopefully, this shows you that Power BI Desktop is rigorous in applying its hierarchy of filters. Should you need to apply a filter at visualization level when the filter choice is excluded at report or page level, you have no choice but to remove the filter at the higher level and then reapply visualization-level filters to all necessary visualizations.

Filtering Tips

Power BI Desktop makes it incredibly easy to filter data and to exclude any data that you feel is not helpful in your data analysis. However, like many powerful tools, this ability to apply filters so quickly and easily can be something of a double-edged sword. So here are a few words of advice and caution when applying filters to your data.

Don't Filter Too Soon

As an initial point, I would say that a key ground rule is "Don't filter too soon." By this, I mean that if you are examining data for trends, anomalies, and insights, you have to be careful not to exclude data that could contain the very insights that can be game changing.

The problem is that when you first delve into a haystack of data in search of needles of informational value, you have no idea what you could be looking for. So I can only suggest the following approaches:

- Begin with no filters at all and see what the data has to say in its most elemental form.
- Apply filters one at a time and remember to delete a filter before trying out another one.
- Try to think in terms of "layers" of filters. So, once you have defined an initial set of filters, add further filters one by one.
- Go slowly. The temptation is to reach a discovery in order to shout about it from the rooftops. This can lead to excessive filtering and unreliable data.
- Always remove any filters that are not absolutely necessary.
- Be careful if you hide the Filters well. It is too easy to forget that there are active filters if they are not visible in some way.
- Remember that you can have filters specific to a visualization that might not be immediately visible in the Filters well without scrolling. So always check if any visualization filters are active for each table and chart in a report.

Annotate, Annotate, Annotate

If you are presenting a key finding based on a dataset, then it can save a lot of embarrassment if you make it clear in every case what the data does and does not contain. For example, you could be so pleased with the revelatory sales trend that you have discovered that you forget to note an important exclusion in the underlying data. Now, no one is suggesting that you are doing anything other than making a point, but your audience needs to know what has been excluded and why—just in case it makes a difference. After all, you don't want a workplace rival pointing this out to invalidate your findings in the middle of a vital meeting, do you?

Annotation techniques are described in Chapter 22 if you need to jump ahead to check this out now.

Avoid Complex Filters

Power BI Desktop filters are designed to be intuitive and easy to use. A consequence of this is that they can prove to be a little limited when you need to apply very complex filters—be they text, numeric, or date filters.

If you need to create a complex filter, it's probably best *not* to create one in the report. Instead, consider trying to filter source data using Power BI Desktop Query. Should you need a more interactive way of switching between complex filter settings in a report, then use DAX to define columns that display a text as the result of a filter (as described in Chapter 11), then use the result of the filter as a selection. As a very simple example of this, consider the Mileage Range column that you defined in Chapter 11. In effect, this groups vehicle mileage into certain bandings. It follows that selecting one or more bandings in a filter will restrict the display to data that matches the predefined data ranges. With a little practice, you can extend this technique to create quite complex filters in DAX at the data level.

Conclusion

This chapter has shown you how to apply and fine-tune a series of techniques to enable you to select the data that will appear in your Power BI Desktop reports. The main thing to take away is that you can filter data at three levels: the overall report, each page, and each visualization on a page.

You have also seen a variety of selection techniques that allow you to subset data. These range from the avowedly simple selection of a few elements to the specification of a more complex spread of dates or values. Finally, it is worth remembering that you can filter data using any of the fields in the underlying dataset, whether the field is displayed in a visual in a report or not.

CHAPTER 21

Using Slicers

With your filters in place, you now have some extremely powerful and insightful dashboards ready to be paraded in front of your colleagues, bosses, and clients. Yet static illustrations can only tell a story in a certain way. What you need to clinch the deal or convince an audience is some truly telling interaction with your facts and figures. Once again, Power BI Desktop is the tool of choice, as it highlights the key metrics in your presentation with a single click—and makes your point, simply and elegantly.

Put less breathlessly, you can interact with your filtered data in Power BI Desktop reports to subset or isolate metrics. These elements have the following characteristics:

- Always visible in the Power BI Desktop report
- Instantly accessible
- Interactive
- Clearly indicate which selections are being applied

So what are the effects that you can add to a Power BI Desktop report to select and project your data? Essentially, they boil down to two main approaches

- Slicers
- Highlighting

These interactive elements can be considered to function as a supplementary level of filtering. That is, they take the current filters that are set in the Filters well (at any level) and then provide further fine-grained *interactive* selection on top of the dataset that has been allowed through the existing filters. Each approach has its advantages and limitations, but used appropriately, each gives you the ability not only to discover the essence of your data, but also to make your point clearly and effectively.

You will use the C:\PowerBiDesktopSamples\CH21\CarSalesDataForReports.pbix sample file as the basis for all the slicers that you create in this chapter.

Slicers

A key form of interactive filter in Power BI Desktop is the *slicer*. This is, to all intents and purposes, a standard multiselect filter, where you can choose one or more elements to filter data in a report. The essential difference is that a slicer remains visible on the Power BI Desktop report, whereas a filter is normally hidden. So this is an overt rather than a covert approach to data selection that makes the selection criteria immediately visible. Moreover, you can add multiple different slicers to a Power BI Desktop report and consequently slice and dice the data instantaneously and interactively using multiple, cumulative, criteria. Slicers can be text-based, or indeed, they can be simple charts, as you will soon see.

Adding a Slicer

To appreciate all that slicers can do, we need to see one in action. This means having at least one standard visual in a page so that you can see the result of applying a slicer. To test a slicer

1. Create a table (to show the effect of using a slicer) using the following fields:
    a. CountryName (from the Countries table)
    b. SalePrice (from the InvoiceLines table)
2. Resize the table so that it fits the data.
3. In the Fields list, expand the Colors table.
4. Drag the Color field to an empty part of the dashboard canvas. It will become a single-column table.
5. Click the Slicer icon in the Visualizations pane. The table of colors will become a slicer. The Slicer icon is shown in Figure 21-1.

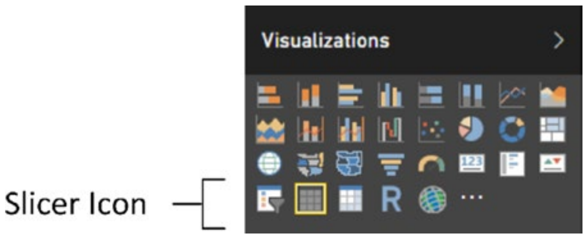
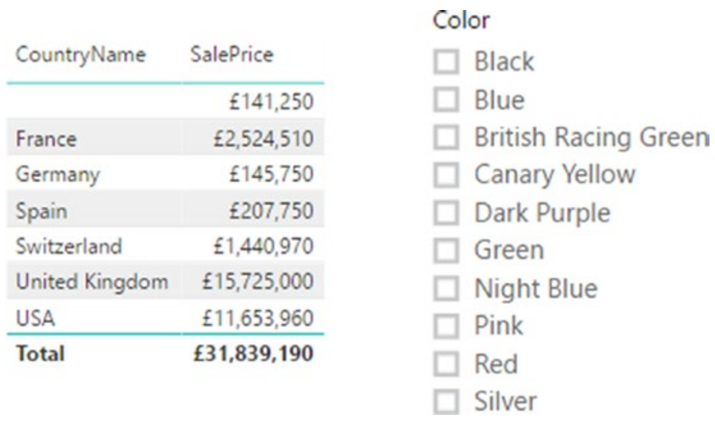


Figure 21-1. *The Slicer icon*

6. Adjust the size of the slicer to suit your requirements using the corner or lateral handles. If the slicer contains many elements, Power BI Desktop will add a vertical scroll bar to indicate that there are further elements available.

You can recognize a slicer by the small squares to the left of each element in the list. This way you know that it is not just a single-column table. Figure 21-2 shows a slicer using the Color field.



CountryName	SalePrice
	£141,250
France	£2,524,510
Germany	£145,750
Spain	£207,750
Switzerland	£1,440,970
United Kingdom	£15,725,000
USA	£11,653,960
Total	**£31,839,190**

Color
- ☐ Black
- ☐ Blue
- ☐ British Racing Green
- ☐ Canary Yellow
- ☐ Dark Purple
- ☐ Green
- ☐ Night Blue
- ☐ Pink
- ☐ Red
- ☐ Silver

Figure 21-2. *A slicer*

■ **Tip** If the Slicer icon is grayed out, then ensure that the table that you are trying to convert to a slicer has only *one* column (that is, one field in the Values box of the field well for this visual).

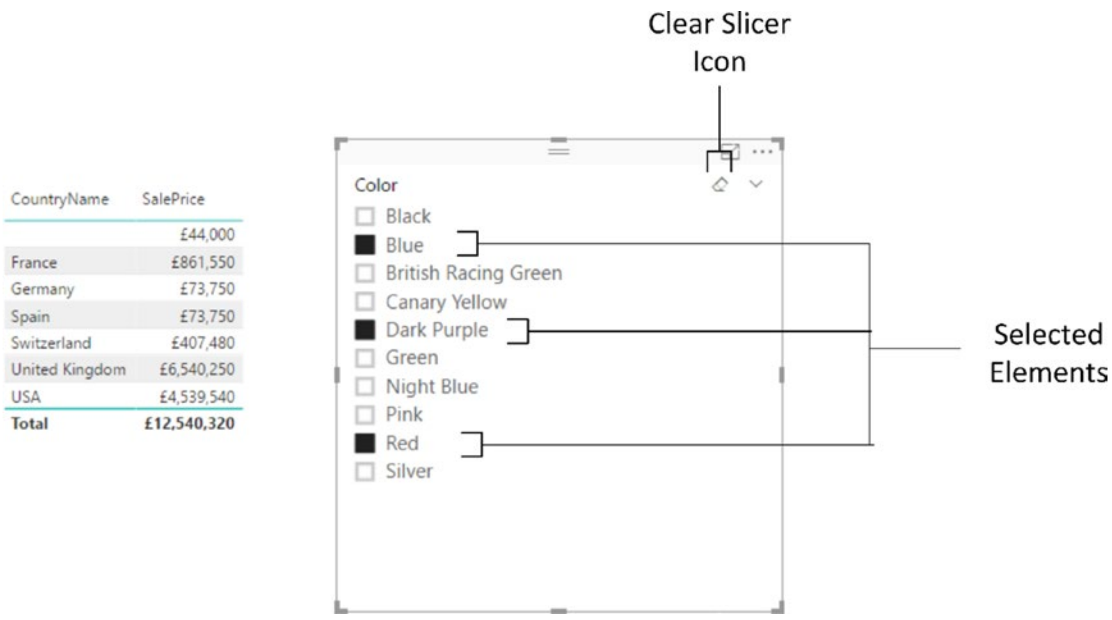
You can create *multiple* slicers for each page. All you have to do is repeat steps 3 through 6 for adding a slicer using a different field as the data for the new slicer.

Applying Slicers

To apply a slicer and use it to filter data on a page, click a single element in the slicer or Ctrl-click multiple elements.

All the objects in a Power BI Desktop page are filtered to reflect the currently selected slicer list. In addition, each element in the slicer list that is active (and consequently used to filter data by that element) now has a small rectangle to its left, indicating that this element is selected.

Figure 21-3 shows what happens when the slicer defined previously is applied to the visualization shown in Figure 21-2.



CountryName	SalePrice
	£44,000
France	£861,550
Germany	£73,750
Spain	£73,750
Switzerland	£407,480
United Kingdom	£6,540,250
USA	£4,539,540
Total	**£12,540,320**

Color
- ☐ Black
- ■ Blue
- ☐ British Racing Green
- ☐ Canary Yellow
- ■ Dark Purple
- ☐ Green
- ☐ Night Blue
- ☐ Pink
- ■ Red
- ☐ Silver

Figure 21-3. *Applying a slicer*

When you apply a slicer, think filter. That is, if you select a couple of elements from a slicer based on the CountryName field, as well as three elements in another slicer based on the Color fields, you are forcing the two slicers (filters) to limit all the data displayed in the page to two countries that have any of the three colors that you selected. The core difference between a slicer and a filter is that a slicer is always visible—and that you have to select or unselect elements, not ranges of values.

If you experiment, you will also see that you cannot create a slicer from numeric fields in the source data. A slicer has to be based on a text field. If you need slicers based on ranges of data, then you will need to prepare these ranges in the data model. The CarAgeBucket field is an example of this. Chapters 11 and 12 explain how to add these sorts of fields to a data model.

■ **Tip** If you Ctrl-click all the selected elements in a slicer, you can unselect all the data it represents. This will not clear the Power BI Desktop report, however. Unselecting everything is the same as selecting everything—despite the fact that the selection squares are no longer visible to the left of each element in the slicer. Using Shift-click is no different from clicking a slicer element, however.

Clearing a Slicer

To clear a slicer and stop filtering on the selected data elements in a view, click the Clear Slicer icon at the top right of the slicer. This icon is pointed out in Figure 21-3.

Any filters applied by the slicer to the view are now removed. You will see that each element in the slicer list now has a small empty rectangle to its left, indicating that this element is not selected. No data is now filtered out of the report.

■ **Tip** Another technique used to completely clear a slicer is to click (or Ctrl-click) the last remaining active element in a slicer. This leaves all elements inactive. Additionally, you can Ctrl-click to select every item. So, in effect, removing all slicer elements is the same as activating them all.

Deleting a Slicer

To delete a slicer and remove all filters that it applies for a view, select the slicer and press the Delete key. Alternatively, click the context menu button (the ellipses) that appears at the top right when you hover the mouse pointer over a slicer, and select Remove.

Any filters applied by the slicer to the view, as well as the slicer itself, are now removed.

You can even copy and paste slicers if you wish. This is very useful when you are copying slicers across different Power BI Desktop reports or between report pages.

Converting a Slicer to Another Visual Type

If you intend to use the field that was the basis for a slicer in a table or chart, you do not need to delete the slicer and re-create a table based on the same underlying field. You can merely

1. Select the slicer.
2. Click the Table button in the Design ribbon, and select the type of table (table, matrix, or card) to which you want to convert the slicer.

The instant that a slicer becomes a table, it ceases to subset the data in the Power BI Desktop report.

Modifying a Slicer

If all you want to do is replace the field that is used in a slicer with another field, then it is probably simplest to do this:

1. Select the slicer that you want to modify.
2. Drag the new field over the existing field in the field well.

The current slicer field is replaced by the new field and the slicer updates to display the contents of the field that you added. Alternatively, you can delete the slicer and re-create it.

■ **Note** You cannot add more than one data field to a slicer, nor can you drag a new field onto the slicer to replace the existing field.

When you start applying slicers to your Power BI Desktop reports, you rapidly notice one important aspect of the Power BI Desktop filter hierarchy. A slicer can only display data that is not specifically excluded by a report- or page-level filter. For instance, if you add a Color filter at page level and select only certain colors in this filter, you are only able to create a slicer that also displays this subset of colors. The slicer is dynamic and reflects the elements that can be displayed once any report and page filters have been applied—just like any other visual. Consequently, adding or removing elements in a filter causes these elements to appear (or disappear) in a slicer that is based on the filtered field.

If you wish, you can apply a filter specifically to a slicer. This allows you to restrict the elements that appear in a slicer. If you want to do this, then you must apply the filter after step 3 in the previous example and *before* you convert the table to a slicer. However, be aware that the slicer itself does *not* give you any indication that it is being filtered; that is, there is no visual-level filter displayed in the Filters well.

Conversely, to remove a filter from a slicer, you need to switch it back to a table and remove the filter, and then switch back to a slicer.

■ **Tip** When you save a Power BI Desktop file containing Power BI Desktop reports with active slicers, the slicer is reopened in the state in which it was saved.

Date Slicers

Selecting ranges of dates can extend the appearance and usefulness of slicers. As this is probably easier to appreciate if you see it in action, I suggest that you try out the following:

1. Open the sample file C:\PowerBiDesktopSamples\CH21\CarSalesDataForReports.pbix.
2. In the Visualizations pane, click the Slicer icon. An empty slicer will be added to the report canvas.
3. Expand the DateDimension table and check the FullYear field.
4. Drag the left-hand (lower) threshold to the right until the left year field shows 2013.

5. Drag the right-hand (upper) threshold to the left until the right year field shows 2015. The slicer will look like the one in Figure 21-4. Any data in the page will be filtered to exclude dates outside this range of years.

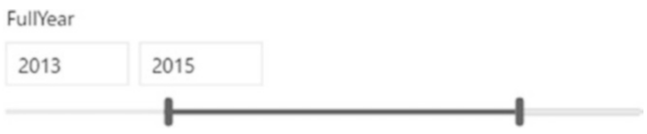


Figure 21-4. *Slicing on years*

You can, if you prefer, enter a year directly in one of the year fields in a slicer like this one. However, I feel that the usefulness of a slicer like this is precisely in its ability to alter the threshold values quickly and easily using the slider.

If you use a data field that is set as a date type, then the slicer can help you set dates quickly and easily, but in a slightly different way. Here's how:

1. In the Visualizations pane, click the Slicer icon. An empty slicer will be added to the report canvas.
2. Expand the DateDimension table and check the DateKey field.
3. Click inside one of the date fields. The slicer will look like the one in Figure 21-5.

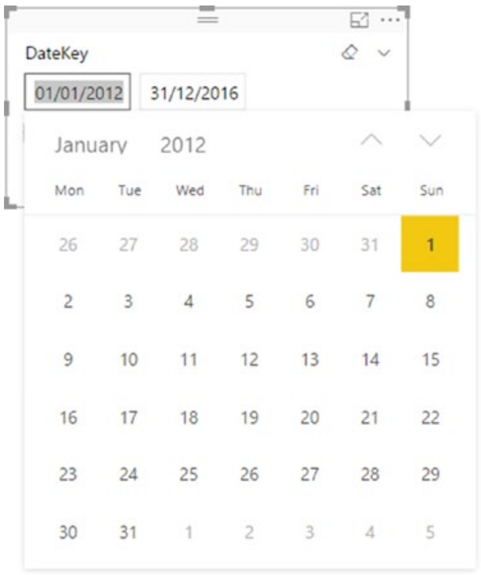


Figure 21-5. *A date slicer*

4. Select the date that you want to set for the date threshold.

To make the most of this calendar, you need to know a few tricks:

- Clicking the up and down chevrons displays the following or preceding month.
- Clicking the year in the calendar popup displays a list of years. From here you can select the required year.
- Clicking the month in the calendar popup displays a list of months. From here you can select the required month.
- You cannot select an upper date that is less than the lower date.
- The calendar *will* allow you to select dates that are not included in the underlying table of dates. This, however, can cause some unexpected results to appear in your reports.

■ **Note** Certain fields in a date dimension (sometimes this can mean most of them) are considered as being simple selection elements. So, for instance, using the MonthFull field in a slicer will result in a list of months, without sliders. The best thing to do is to experiment with the various fields that are available in the Date Dimension table and use those that best suit your purposes.

Formatting Slicers

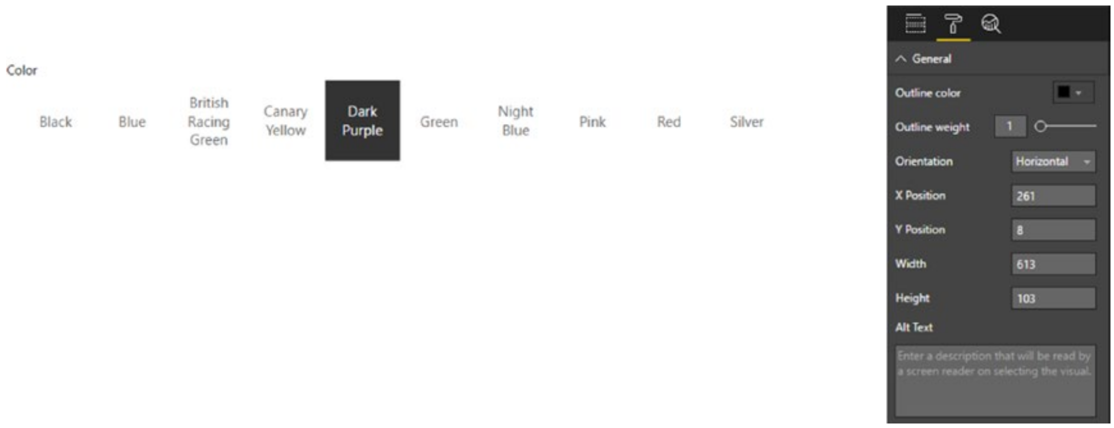
Slicers can be formatted just like any other Power BI Desktop visual. Indeed, many of the techniques that you use to format slicers are identical to those that you have already seen when formatting tables and matrices. Consequently, to avoid pointless repetition, this section concentrates on any formatting attributes that are slicer-specific, and only refers in passing to formatting approaches that you have already covered in previous chapters.

Slicer Orientation

To help you to make the best possible use of report real estate, slicers can be configured to appear vertically (as in Figure 21-2) or horizontally. To switch a slicer's orientation:

1. Select the slicer that you want to modify. (I selected the colors slicer that you created previously.)
2. In the Visualizations pane, click the Format icon.
3. Expand the General section.
4. In the Orientation popup, select Horizontal.

5. Resize the slicer to suit your dashboard. A horizontal slicer will look something like the one in Figure 21-6.



Outline color
Outline weight 1
Orientation Horizontal
X Position 261
Y Position 8
Width 613
Height 103
Alt Text
Enter a description that will be read by a screen reader on selecting the visual.

Figure 21-6. *A horizontal slicer*

As you can see in Figure 21-6, a horizontal slicer does not have the small check boxes that a vertical slicer does. Consequently, when a horizontal slicer element is selected, the entire element is in "reverse video."

You probably also noticed that Power BI Desktop alters the number of rows of text as well as the width of text elements when you resize the slicer. It follows that you are best advised to experiment when altering slicer height and width in order to get the effect that suits you best.

Modifying the Outline

For Power BI Desktop, the outline is the line that separates the title from the items in a slicer. The following discusses how you can format this particular element.

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier.)
2. In the Visualizations pane, click the Format icon.
3. Expand the General section.
4. Use the Outline Weight slider to set the weight of the outline to 3 points.
5. Click the Outline Color popup and choose a color. The separator line will change to reflect the modifications that you have made.

Adjusting Selection Controls

If you are not happy with the way that Power BI Desktop lets you select items in a slicer, then you can adjust the way that you interact with this particular visual. While the Power BI default mode of interaction is probably sufficient in most cases, it certainly does no harm to know that there are other ways of selecting elements in slicers.

Adding or Removing the Select All Box

You can add the Select All box to slicer as follows:

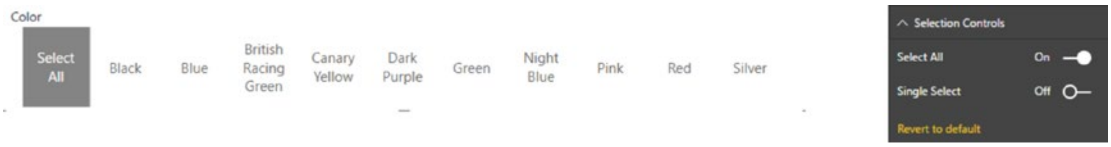
1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier).
2. In the Visualizations pane, click the Format icon.
3. Expand the Selection Controls section.
4. Slide the Select All switch to the On position. This will add a Select All item to the top (or left) of the slicer.

Should you *not* want a Select All item in a slicer, all you have to do is ensure that the Select All switch is set to Off in step 4.

Enabling Single Select

A slicer's default mode is to select only a single item, unless the user Ctrl-clicks several items. If you prefer to add items one at a time—and deactivate them individually too—you can enable Single Select to do just this. Simply follow these steps:

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier).
2. In the Visualizations pane, click the Format icon.
3. Expand the Selection Controls section.
4. Slide the Single Select all switch to the Off position. This will set the slicer interaction to single select. You can see the result of this operation—and what happens when you add a Select All element—in Figure 21-7.



Selection Controls
Select All On
Single Select Off
Revert to default

Figure 21-7. *Altering a slicer's selection controls*

Setting the Exact Size and X and Y coordinates of a Slicer

If you want to place a slicer with total accuracy on a dashboard canvas, then you can set the X and Y (horizontal and vertical) coordinates for the slicer. You can also specify its exact height and width. The following explains how:

1. Select the slicer that you want to modify.
2. In the Visualizations pane, click the Format icon.
3. Expand the General section.
4. Replace the X Position, Y Position, Width, and Height values with the pixel values that define the size and position of the slicer that you wish to apply.

Formatting the Slicer Header

Should you want to add some visual pizzazz to a slicer, you can tweak the display of the slicer header. Here is an example of how to do this:

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier).
2. In the Visualizations pane, click the Format icon.
3. Expand the Header Controls section.
4. Ensure that the Header switch is in the On position.
5. Click the Font Color popup and choose a color for the header text.
6. Click the Background Color popup and choose a color for the header background.
7. Adjust the Text Size slider to tweak the size of the header text.

Formatting Slicer Items

Slicer items are the individual elements that make up the list of data that appear in a slicer, based on the underlying field. These, too, can be formatted to focus the attention of the reader. Here's an example:

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier.)
2. In the Visualizations pane, click the Format icon.
3. Expand the Items section.
4. Click the Font Color popup and choose a color for the item text.
5. Click the Background Color popup and choose a color for the background of each item.
6. Adjust the Text Size slider to tweak the size of the text of each item.
7. Click the Outline popup and select Frame.

You can also format the following aspects of a slicer:

- Background
- Title
- Lock aspect ratio

However, since all of these are identical to formatting attributes that apply to tables and charts (and that you have seen in previous chapters), I will not repeat the description of them.

Finally, Figure 21-8 shows you a slicer with many of the formatting options that you just saw.

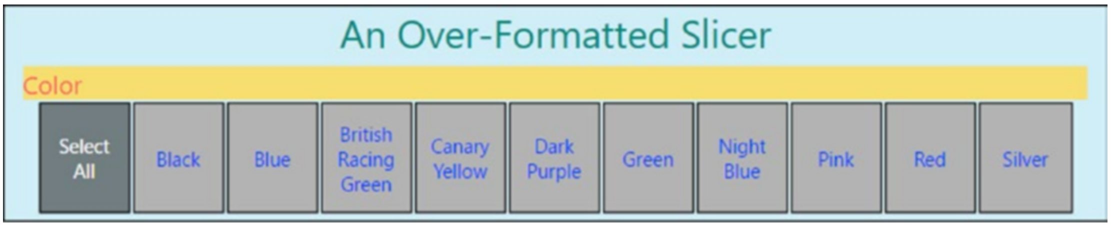


***Figure 21-8.** A formatted slicer*

Sorting Slicer Elements

The elements in a slicer will be displayed in alphabetical order. If you wish, you can reverse this sort order, as follows:

1. Click the ellipses at the top right of the slicer.
2. Select Sort By Color (or whatever the field name is).

The slicer elements will now be displayed in reverse alphabetical order.

■ **Tip** You can add a Sort By column to any data field that you wish to display in another sort order.

Switching to Dropdown Slicers

Slicers are amazingly useful, but they do have one drawback: they can take up a lot of valuable screen space. So the development team at Microsoft has come up with an elegant solution. This is to use dropdown slicers, where the slicer elements only appear when you click a popup menu.

To convert an existing slicer to a dropdown slicer:

1. Select an existing slicer (or hover the mouse pointer over it). I will use the colors slicer that you have used so far in this chapter.

2. Click the chevron on the top right next to the Clear Selections icon. A popup menu will appear, as you can see in Figure 21-9.

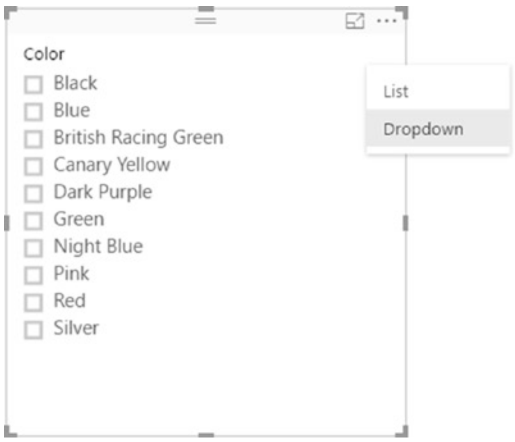


Figure 21-9. *Switching to dropdown slicer*

3. Select Dropdown. The slicer will convert to a dropdown slicer. You can see what this now looks like in Figure 21-10.

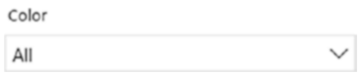


Figure 21-10. *A dropdown slicer*

4. Click the chevron for the slicer popup list and select the element that you want to slice the data on.

■ **Tip** Dropdown slicers can be particularly useful when designing Power BI reports for mobile phones.

Exporting Slicer Data

Slicer data can be exported just like the data in any other visual:

1. Click the ellipses at the top right of the slicer.
2. Select Export Data.
3. Browse to the desired directory and enter a file name.
4. Click Save.

Using Charts As Slicers

You have seen how a table can become a slicer, which is a kind of filter. Well, charts can also be used as slicers. Knowing how charts can affect the data in a Power BI Desktop report can even influence the type of chart that you create, or your decision to use a chart to filter data rather than a standard slicer. Charts can be wonderful tools to grab and hold your audience's attention—as I am sure you will agree once you have seen the effects that they can produce.

Charts As Slicers

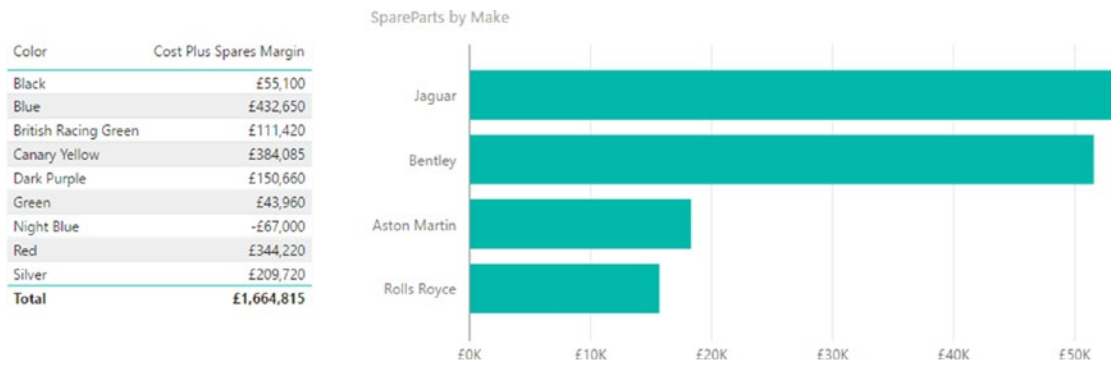
To begin with, let's see how a chart can be used to act as a slicer to filter data interactively for any or all of the visualizations in a Power BI Desktop report. Initially, let's assume that we are aiming to produce a report using two objects:

- A Cost Plus Spares by Color table
- A Spare Parts by Make column chart

Let's create a Net Margin by Color table. It is principally used to show the effect that using a chart as a slicer in a Power BI Desktop report has on other objects. As an added extra, you will apply a filter to the page to demonstrate that filters and slicers work together, as described.

1. Open the C:\PowerBiDesktopSamples\CH21\CH21\CarSalesDataForReports.pbix file and delete any existing visuals. You will need an entire uncluttered report for this example.
2. Expand the DateDimension table and drag the FullYear field into the Page Level Filters box in the Filters well.
3. Set the Filter Type popup to Basic Filtering.
4. Check the box for the year 2014. This way, you have filtered the page to display only data for 2014.
5. Add a table based on the following fields:
    a. Color (from the Colors table)
    b. CostPlusSpares (from the InvoiceLines table)
6. Add a bar chart based on the following fields:
    a. Make (from the Stock table)
    b. SpareParts (also from the Stock table)
7. Click the popup menu for the bar chart (the ellipses that appear at the top right of the visual when you place the mouse pointer over the chart).
8. Click the ellipses at the top right of the chart and then select Sort By SpareParts. This will sort the bar chart in descending order.

9. Adjust the layout of the two visualizations so that it looks something like Figure 21-11.

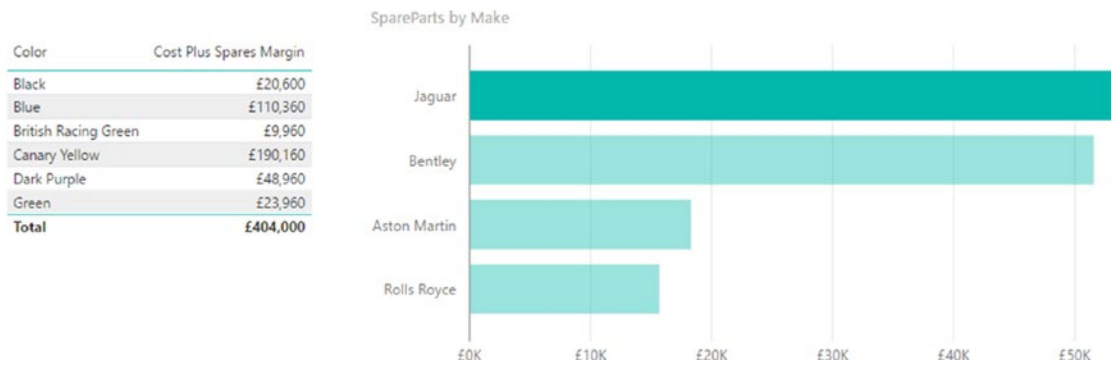


Color	Cost Plus Spares Margin
Black	£55,100
Blue	£432,650
British Racing Green	£111,420
Canary Yellow	£384,085
Dark Purple	£150,660
Green	£43,960
Night Blue	-£67,000
Red	£344,220
Silver	£209,720
Total	**£1,664,815**

Figure 21-11. *Preparing a chart for use as a slicer*

To see how to use a chart as a slicer, click any column in the chart of parts costs by make. I will choose Jaguar in this example.

The Power BI Desktop report will look something like Figure 21-12.



Color	Cost Plus Spares Margin
Black	£20,600
Blue	£110,360
British Racing Green	£9,960
Canary Yellow	£190,160
Dark Purple	£48,960
Green	£23,960
Total	**£404,000**

Figure 21-12. *Slicing data using a chart*

You will see that not only is the make that you selected highlighted in the chart (the bars for other makes are dimmed), but that the figures in the table also change. They, too, only display the cost plus spares (for each color) for the selected make.

To slice on another make, merely click the corresponding column in the column chart. To cancel the effect of the chart acting as a slicer, all you have to do is click for a *second time* on the highlighted column.

Any bar chart, pie chart, or column chart can act like a slicer in this way, as can funnel charts, treemaps, scatter charts, bubble charts, and maps. The core factor is that for a simple slice effect, you need to use a chart that contains only one axis; that is, there will only be a single axis in the source data and no color or legend. What happens when you use more evolved charts to slice, filter, and highlight data is explained next.

■ **Tip** It is perfectly possible to select multiple bars in a chart to highlight data in the same way that you can select multiple elements in a slicer. Simply Ctrl-click to select multiple items.

Highlighting Chart Data

So far, we have seen how a chart can become a slicer for all the visualizations in a dashboard page. However, you can also use another aspect of Power BI Desktop interactivity to make data series in charts stand out from the crowd when you are presenting your findings. This particular aspect of data presentation is called *highlighting*.

Once again, highlighting is probably best appreciated with a practical example. First, we will create a stacked bar chart of parts and labor costs by CountryName; and then we will use it to highlight the various costs inside the chart.

1. In a blank Power BI Desktop report (so you do not get distracted), create a clustered column chart based on the following fields:
   a. CountryName (from the Countries table)
   b. SpareParts (from the Stock table)
   c. LaborCost (from the Stock table)
2. Click SpareParts in the *legend*. All the sales costs will be highlighted (that is, remain the original color) in the column for each country, whereas the other cost will be grayed out.

After highlighting has been applied, the chart will look like Figure 21-13.

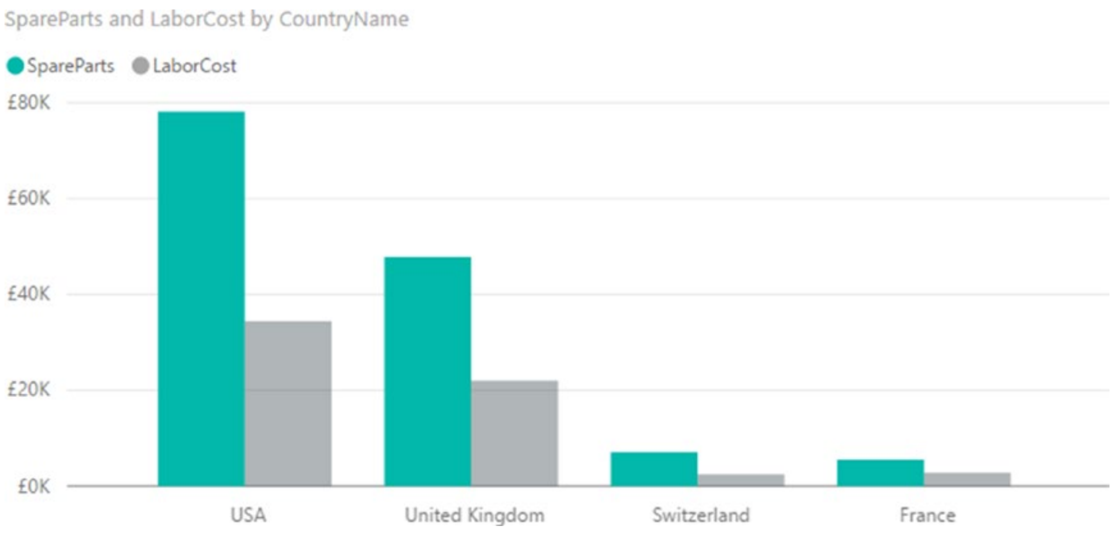


Figure 21-13. *Highlighting data inside a chart*

To remove the highlighting, all you have to do is click a second time on the same element in the legend. Or, if you prefer, you can click another legend element to highlight this aspect of the visualization instead. Yet another way to remove highlighting is to click inside the chart, but not on any data element.

Highlighting data in this way should suit any type of bar or column chart as well as line charts. It can also be useful in pie charts where you have added data to both the Axis and Legend boxes, which, after all, means you have multiple elements in the chart just as you can have with bar, column, and line charts. You might find it less useful with scatter charts.

Cross-Chart Highlighting

Cross-chart highlighting adds an interesting extra aspect to chart highlighting and filtering. If you use one chart as a filter, the other chart is updated to reflect the effect of selecting this new filter not only by excluding any elements (slices, bars, or columns) that are filtered out, but also by showing the proportion of data excluded by the filter.

As an example of this, create a pie chart of cost price by color and a column chart of direct costs by vehicle type. We will then cross-filter the two charts and see the results. The steps to follow are

1. Create a pie chart using the following fields:
   a. Color (from the Colors table)
   b. CostPrice (from the Stock table)
2. Create a (clustered) column chart using the following fields:
   a. Vehicle (from the Stock table)
   b. CostPrice (from the Stock table)

For charts that are this simple, Power BI Desktop automatically attributes the fields to the correct boxes in the Fields list once the source tables are converted into charts. The result is shown in Figure 21-14.

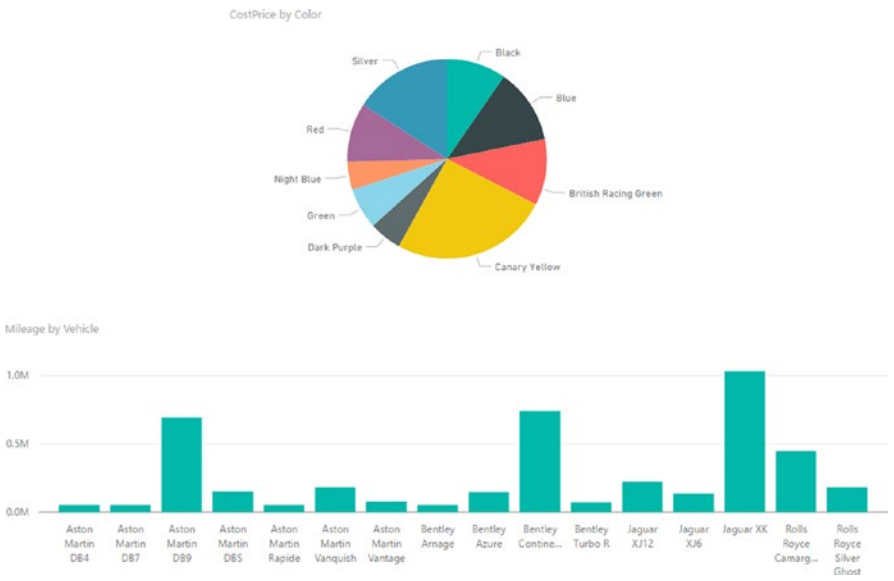


Figure 21-14. *Preparing charts for cross-chart highlighting*

3. Now click the largest slice in the pie chart (Canary Yellow). You should see the result given in Figure 21-15. Not only have all the other segments of the pie chart been dimmed, but the bars in the bar chart have been highlighted to show the proportion of the selected color of the total sales cost per vehicle cost.

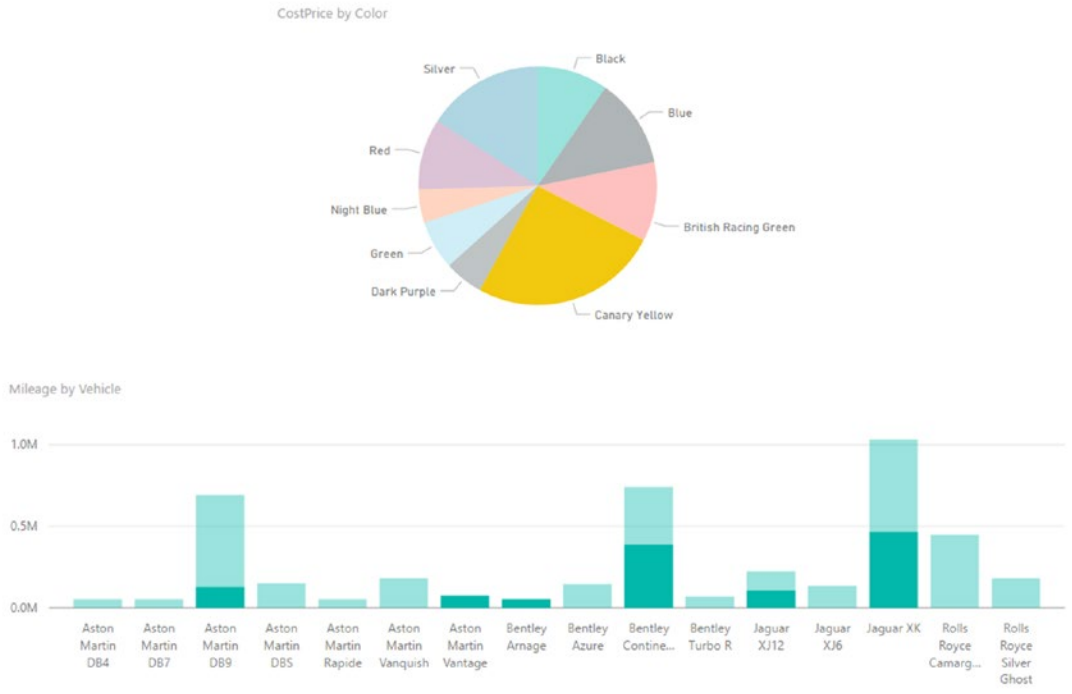


Figure 21-15. *Cross-chart highlighting*

4. Now click the bar in the bar chart representing Aston Martin DB9. You are now using the bar chart as a slicer. As you can see in Figure 21-16, the pie chart displays the proportion of Aston Martin DB9 sales for each color.

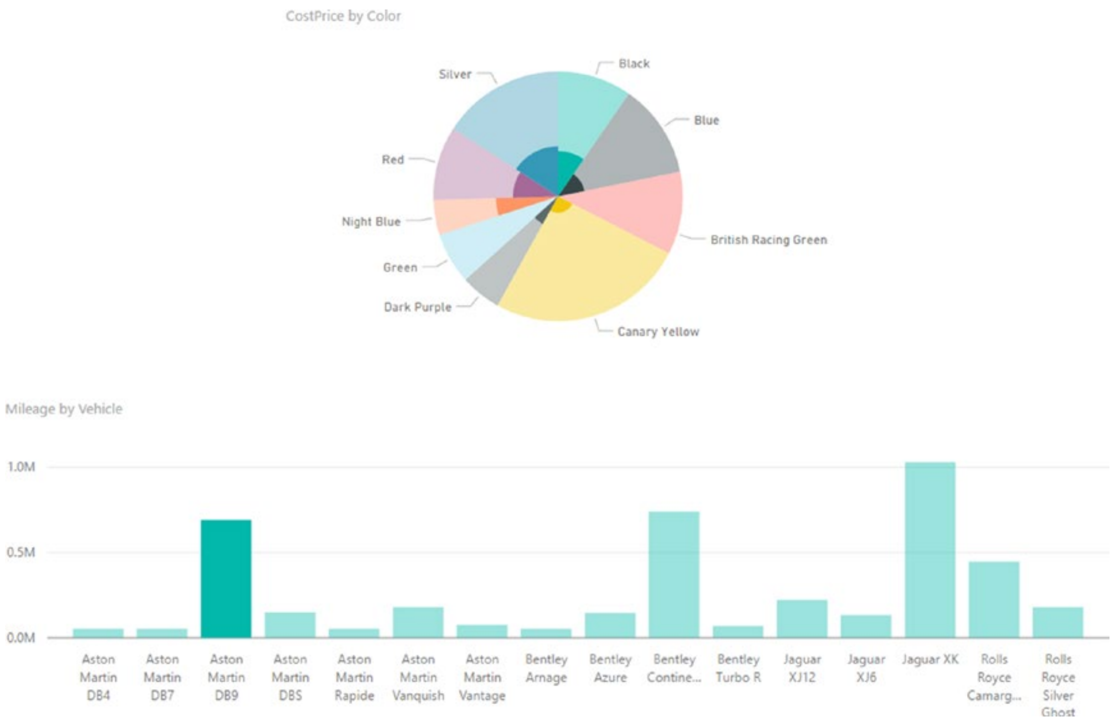


Figure 21-16. *Cross-chart highlighting applied to a pie chart*

■ **Note** When you use a filter, you do not highlight a chart but actually filter the data that feeds into it; consequently, you remove elements from the chart. Highlighting leaves elements in a chart but accentuates certain aspects of the data relative to others.

Highlighting Data in Bubble Charts

Often when developing a visualization whose main objective, after all, is to help you to see through the fog of data into the sunlit highlands of comprehension, profit, or indeed, whatever is the focus of your analysis, you may feel that you cannot see the forest for the trees. This is where Power BI Desktop's ability to highlight data in a chart visualization can be so effective.

Let's take a visualization that contains a lot of information; in this example, it is a bubble chart of vehicle types. Indeed, in this example, an audience might think that there is so much data that it is difficult to see the bubbles for specific makes of car, and so analyze the uniqueness for sales data by make. Power BI Desktop has a solution to isolate a data series in such a chart. To see this in action and to make the details clearer, you need to do as follows:

1. Create a bubble chart (remember that this is a scatter chart, really) using the following elements:
    a. *X Axis*: Average Parts Cost Ratio (from the Stock table)
    b. *Y Axis*: SalePrice (from the InvoiceLines table)
    c. *Size*: GrossMargin (from the InvoiceLines table)
    d. *Details*: Color (from the Colors table)
    e. *Legend*: VehicleType (from the Stock table)
2. In the legend for the chart, click a vehicle type. I used saloon (the British word for sedan in the US) in this example. The data for this vehicle type is highlighted in the chart, and the data for all the other vehicle types are dimmed, making one set of information stand out. This is shown in Figure 21-17.

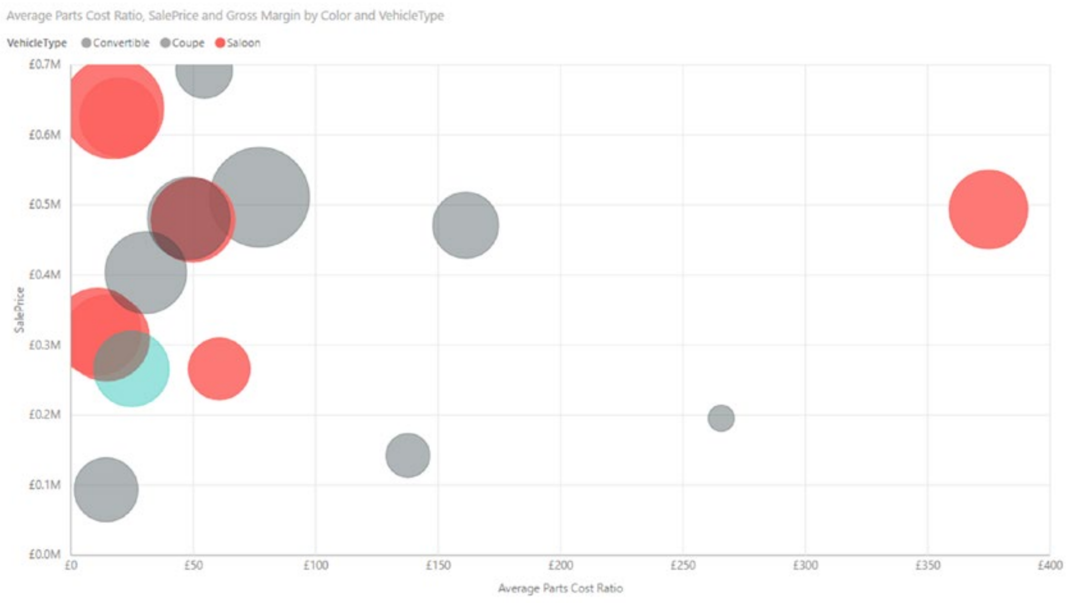


***Figure 21-17.** Highlighting data in bubble charts*

This technique needs a few comments:

- To highlight another dataset, merely click another element in the legend.
- To revert to displaying all the data, click the selected element in the legend again.
- Highlighting data in this way also filters data in the entire page, as described previously.

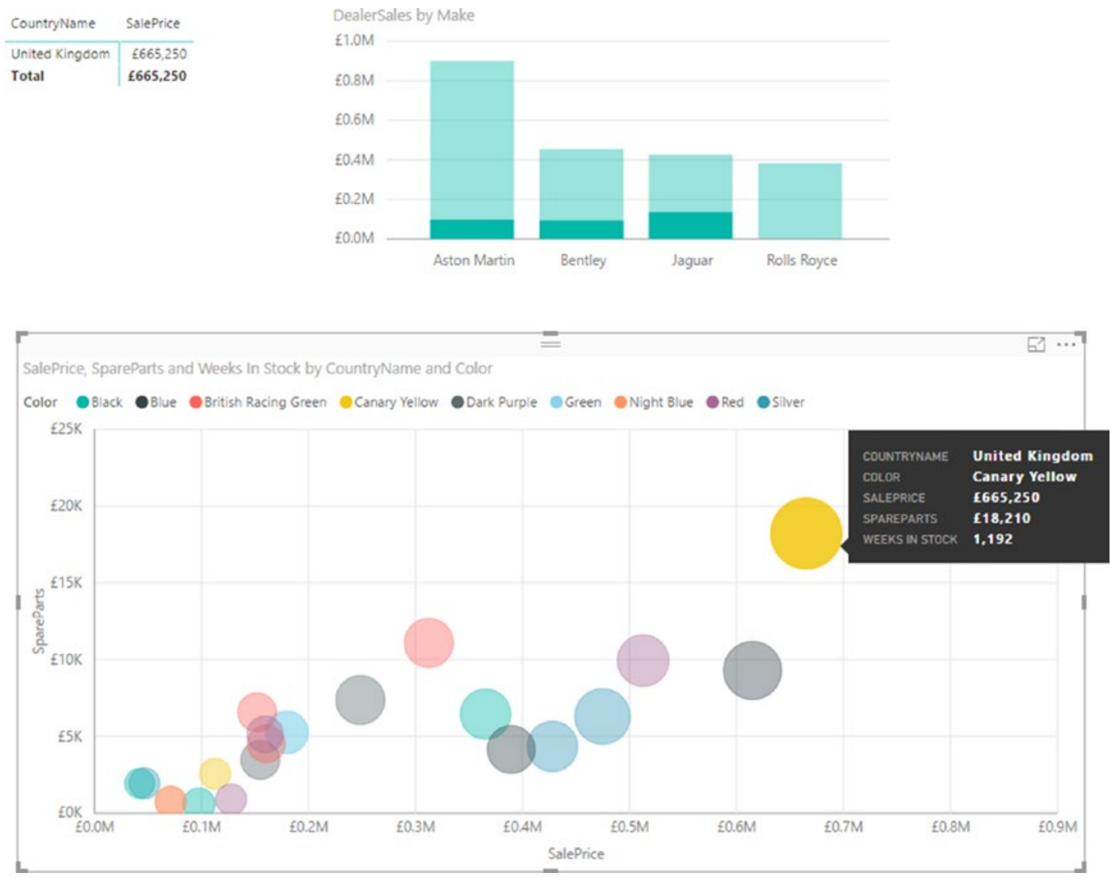
■ **Tip** You can add drill down to charts and still use chart highlighting in exactly the same way as you would use it normally, provided that you have disabled drill down using a chart element, as described in Chapter 16. In this case, you have to drill down (as well as up) using the Drill icons. The chart highlights an element at a drill-down level or sublevel, as well as applying filtering to the Power BI Desktop report.

Charts As Complex Slicers

Now that you have seen how charts can be used as slicers, let's take things one step further and see them used as more complex slicer elements. To show this, I build on the principles shown in the previous example, but add a bubble chart that will slice data on two elements at once.

Follow these steps for the purposes of this example:

1. Build a Power BI Desktop report with the following:
   a. A SalePrice by CountryName and Color matrix
   b. A net sales by Make clustered column chart
2. Create a bubble chart using the following data:
   a. *X Axis*: SalePrice (from the InvoiceLines table)
   b. *Y Axis*: SpareParts (from the Stock table)
   c. *Size*: Weeks In Stock (from the Stock table)
   d. *Details*: CountryName (from the Countries table)
   e. *Legend*: Color (from the Colors table)
3. Resize and tweak the bubble chart so that it is displayed under the existing column chart and table.
4. Click the bubble next to the top-right bubble in the bubble chart. The Power BI Desktop report should look like Figure 21-18. The tooltip indicates which bubble has been used to filter the other visuals.



CountryName	SalePrice
United Kingdom	£665,250
Total	**£665,250**

COUNTRYNAME	United Kingdom
COLOR	Canary Yellow
SALEPRICE	£665,250
SPAREPARTS	£18,210
WEEKS IN STOCK	1,192

***Figure 21-18.** Highlighting and filtering using a chart*

You can see that the other visualizations are filtered so that both the elements that make up the individual bubble (CountryName and Color) are used as filters (or double-slicers if you prefer to think of them like that). This means that

- The table only shows colors where there are sales for this country and this color.
- The chart highlights data for this country and color only.

As was the case with simple chart slicers, you can cancel the filter effect merely by clicking for a second time on the selected bubble. Or you can switch filters by clicking another bubble in the bubble chart. You will also see the chart itself has data highlighted, but this is explained a little further on.

Clearly, you do not have to display the fields on which you are filtering and highlighting in all the visualizations in a report. I chose to do it in this example to make the outcome clearer. In the real world, all other visualizations in a report are filtered on the elements in the Details and Legend boxes of the bubble chart.

Bubble charts are not the only chart type that lets you apply two simultaneous filters, however. All chart types that display multiple fields allow this. However, I am of the opinion that some charts are better suited than others to this particular technique. Specifically, I am not convinced that line charts are always suited to being used as filters for a Power BI Desktop report. Scatter charts may work—visually, that is—if you use cross-filtering, but it is just as likely that they will not. Stacked bar and stacked column charts can be ideal for cross-filtering, as you will see in the following sections.

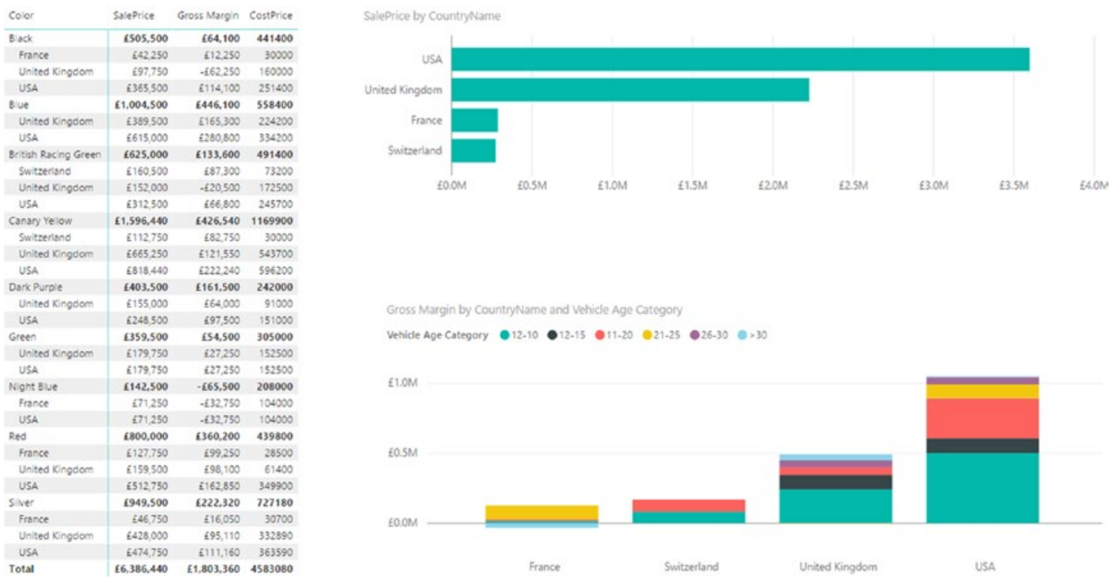
Column and Bar Charts As Filters

Column charts and bar charts can also be used to filter a Power BI Desktop report on two elements simultaneously. The only limitation is that you can only have one set of numeric data as the values for the chart. If the bar or column chart is a stacked bar, then you can click any of the sections in the stacked bar. In addition, if the chart is a clustered bar or column, you can click any of the columns in a group to slice by the elements represented in that section.

If this limitation is not a problem, then this is how you can use bar or column charts (whether they are clustered, stacked, or 100% stacked) to apply double filters to a report:

1. Create a Power BI Desktop dashboard using the C:\PowerBiDesktopSamples\CH21\CarSalesDataForReports.pbix file with the following two elements:
    a. A matrix based on color, country name, sales price, gross margin, and cost price. Once the matrix is created, click the Expand all down one level icon in the hierarchy to see all the levels of data.
    b. A clustered bar chart of sales price by country name.
2. Then create a stacked column chart using the following data:
    a. *Values*: GrossMargin (from the InvoiceLines table)
    b. *Axis*: CountryName (from the Countries table)
    c. *Legend*: VehicleAgeCategory (from the Stock table)

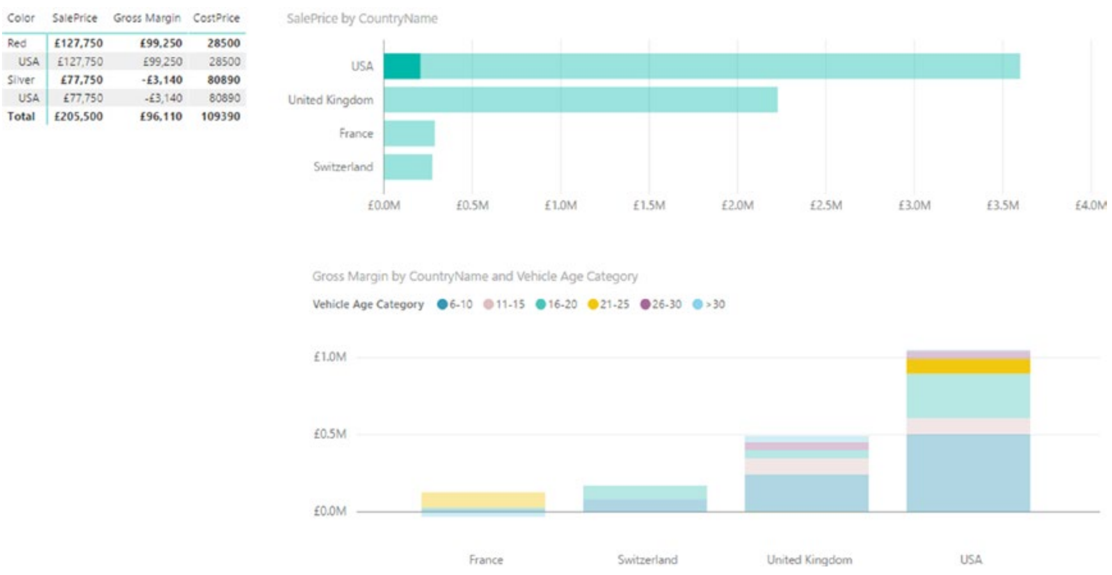
Once tweaked to clarify the appearance of the chart, the net result should look like Figure 21-19.



Color	SalePrice	Gross Margin	CostPrice
Black	**£505,500**	**£64,100**	**441400**
France	£42,250	£12,250	30000
United Kingdom	£97,750	-£62,250	160000
USA	£365,500	£114,100	251400
Blue	**£1,004,500**	**£446,100**	**558400**
United Kingdom	£389,500	£165,300	224200
USA	£615,000	£280,800	334200
British Racing Green	**£625,000**	**£133,600**	**491400**
Switzerland	£160,500	£87,300	73200
United Kingdom	£152,000	-£20,500	172500
USA	£312,500	£66,800	245700
Canary Yellow	**£1,596,440**	**£426,540**	**1169900**
Switzerland	£112,750	£82,750	30000
United Kingdom	£665,250	£121,550	543700
USA	£818,440	£222,240	596200
Dark Purple	**£403,500**	**£161,500**	**242000**
United Kingdom	£155,000	£64,000	91000
USA	£248,500	£97,500	151000
Green	**£359,500**	**£54,500**	**305000**
United Kingdom	£179,750	£27,250	152500
USA	£179,750	£27,250	152500
Night Blue	**£142,500**	**-£65,500**	**208000**
France	£71,250	-£32,750	104000
USA	£71,250	-£32,750	104000
Red	**£800,000**	**£360,200**	**439800**
France	£127,750	£99,250	28500
United Kingdom	£159,500	£98,100	61400
USA	£512,750	£162,850	349900
Silver	**£949,500**	**£222,320**	**727180**
France	£46,750	£16,050	30700
United Kingdom	£428,000	£95,110	332890
USA	£474,750	£111,160	363590
Total	**£6,386,440**	**£1,803,360**	**4583080**

Figure 21-19. *A report ready for chart-based filtering and highlighting*

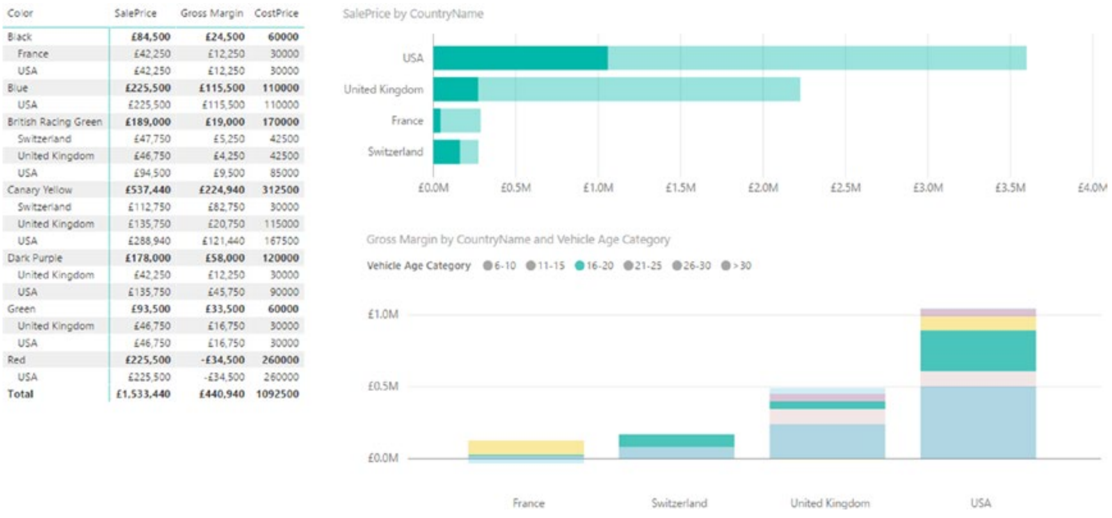
Clicking any segment of a bar filters and highlights other visualizations on the same report for that country and car age range. An example of this is given in Figure 21-20, where the car age range of 21–25 has been selected for the USA bar.



Color	SalePrice	Gross Margin	CostPrice
Red	**£127,750**	**£99,250**	**28500**
USA	£127,750	£99,250	28500
Silver	**£77,750**	**-£3,140**	**80890**
USA	£77,750	-£3,140	80890
Total	**£205,500**	**£96,110**	**109390**

Figure 21-20. *Applying filters and highlights*

Clicking any car age range in the legend will filter by car age range only. You see this in Figure 21-21, where the legend item for 16-20 has been selected.



Color	SalePrice	Gross Margin	CostPrice
Black	**£84,500**	**£24,500**	**60000**
France	£42,250	£12,250	30000
USA	£42,250	£12,250	30000
Blue	**£225,500**	**£115,500**	**110000**
USA	£225,500	£115,500	110000
British Racing Green	**£189,000**	**£19,000**	**170000**
Switzerland	£47,750	£5,250	42500
United Kingdom	£46,750	£4,250	42500
USA	£94,500	£9,500	85000
Canary Yellow	**£537,440**	**£224,940**	**312500**
Switzerland	£112,750	£82,750	30000
United Kingdom	£135,750	£20,750	115000
USA	£288,940	£121,440	167500
Dark Purple	**£178,000**	**£58,000**	**120000**
United Kingdom	£42,250	£12,250	30000
USA	£135,750	£45,750	90000
Green	**£93,500**	**£33,500**	**60000**
United Kingdom	£46,750	£16,750	30000
USA	£46,750	£16,750	30000
Red	**£225,500**	**-£34,500**	**260000**
USA	£225,500	-£34,500	260000
Total	**£1,533,440**	**£440,940**	**1092500**

Figure 21-21. *Filtering using a legend element*

So in fact, you can choose to filter on a single element or multiple elements, depending on whether you use the chart or the legend as the filter source.

■ **Note** A line chart will not produce the same effect when you click only a data point. If you click a series in a line chart, you are highlighting that series, which is numeric data, and so it cannot be used as a slicer. Similarly, if you click an element in the legend of a column or bar chart, you are selecting a data series, and this, too, cannot serve as a slicer (even though it highlights the series in the chart).

Specifying Visual Interactions

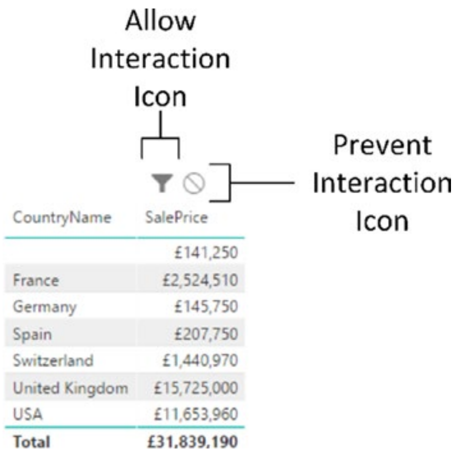
In the previous few sections, you saw that the effects of selecting one or more items in a slicer are applied automatically to every visual on a page, as will the effects of using a chart as a slicer. This is indeed true, and is the default behavior of Power BI Desktop unless you specifically configure a visual *not* to react when a chart or slicer on the page has items selected.

In effect, this means that you can attain a tremendous degree of subtlety in your dashboards, because you can define which visuals are to remain interactive—and which must not change when a slicer, chart, or multiple slicers and charts are used.

The following explains how to alter the default setting, and remove interactivity from a visual. You need to be aware that you need at least two visuals in a report to carry out this modification.

1. Select the visual that you want to prevent having a highlighting effect on other visuals.

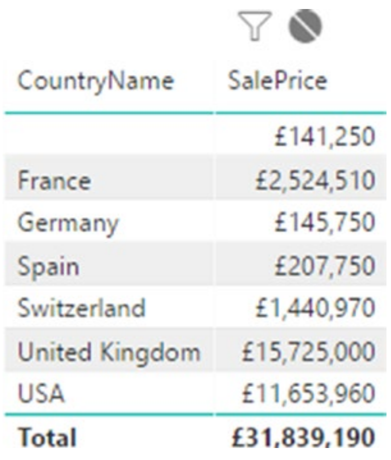
2. In the Home ribbon, click the Edit Interactions button. All the other visuals on the page will display the interaction icons that you can see in Figure 21-22 (a funnel and a "no entry" sign).



CountryName	SalePrice
	£141,250
France	£2,524,510
Germany	£145,750
Spain	£207,750
Switzerland	£1,440,970
United Kingdom	£15,725,000
USA	£11,653,960
Total	**£31,839,190**

Figure 21-22. *The visual interaction icons for visuals*

3. Click the Stop Interaction icon (the "no entry" sign) in another visualization. I used the table of sales by country that you created previously. This icon will appear filled in, as shown in Figure 21-23.



CountryName	SalePrice
	£141,250
France	£2,524,510
Germany	£145,750
Spain	£207,750
Switzerland	£1,440,970
United Kingdom	£15,725,000
USA	£11,653,960
Total	**£31,839,190**

Figure 21-23. *The visual interaction icons set to prevent interaction*

4. Repeat steps 2 through 5 for all visuals that you want to "disconnect" from the selected visual.

5. In the Home ribbon, click the Edit Interactions button to stop configuration of visual interaction.

You can then iterate through all the charts and slicers on a page to set their dependency on another element.

■ **Note** A slicer can also be linked to or dependent on another slicer on the page. Consequently, you can set the interaction for a slicer just as you can for any other visual.

What-If Slicers

A really interesting (and fairly new) feature in Power BI Desktop is the ability to interact with data using what-if slicers. These allow you to define variable values that are then applied to the data in the data model for any fields that you choose. You can then adjust the what-if value in a slider on screen and watch the calculated values change in real time.

This feature is best appreciated if you experiment with it. So here is one example of a what-if slicer. Let's suppose that you want to see what happens to gross margin if suppliers increase the cost of spare parts.

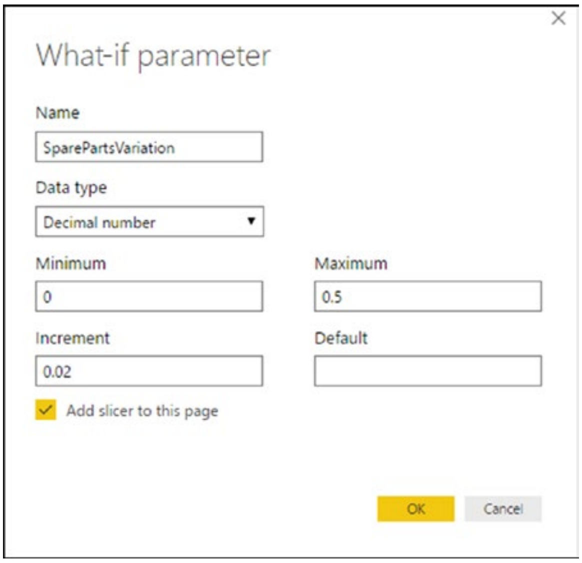
1. In the Modeling ribbon, click the New Parameter button.

2. Enter **SparePartsVariation** in the Name box.

3. Select Decimal Number as the data type from the Data Type popup list.

4. Leave the Minimum value as 0, and enter **0.5** (50 percent) as the Maximum value.

5. Set the increment to **0.02** (two percent). The dialog will look like the one in Figure 21-24.



What-if parameter

Name: SparePartsVariation

Data type: Decimal number

Minimum: 0

Maximum: 0.5

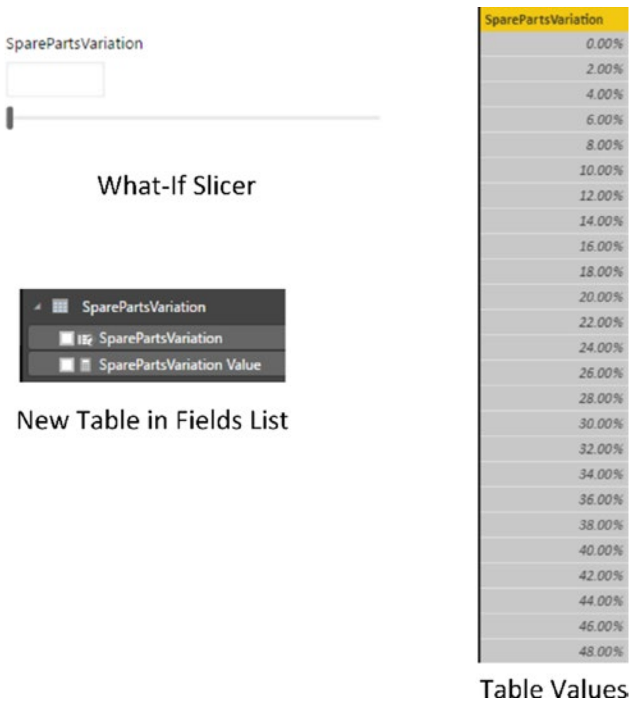
Increment: 0.02

Default:

✓ Add slicer to this page

Figure 21-24. *The What-If Parameter dialog*

6. Click OK. Power BI Desktop will add a new table to the data model named SparePartsVariation as well as a slicer to the desktop canvas. After resizing the slicer, you can see these, as well as the data in the new table, in Figure 21-25.



SparePartsVariation
0.00%
2.00%
4.00%
6.00%
8.00%
10.00%
12.00%
14.00%
16.00%
18.00%
20.00%
22.00%
24.00%
26.00%
28.00%
30.00%
32.00%
34.00%
36.00%
38.00%
40.00%
42.00%
44.00%
46.00%
48.00%

Figure 21-25. *A what-if slicer*

7. Click the SparePartsVariation table in the Fields list and then click New Column in the Modeling ribbon.

8. Enter the following formula in the Formula bar (this will calculate a new spare parts cost using the multiplier from the what-if slicer):

```
SparePartsNew = SUM(Stock[SpareParts]) + (SUM(Stock[SpareParts])
* 'SparePartsVariation'[SparePartsVariation Value])
```

9. Create a clustered column chart using the following fields:

   a. Make (from the Stock table)

   b. SpareParts (from the Stock table)

   c. SparePartsNew (from the SparePartsVariation table)

10. Adjust the slider to apply a new percentage increase in the cost of spare parts. The resulting chart and slicer will look like those in Figure 21-26.

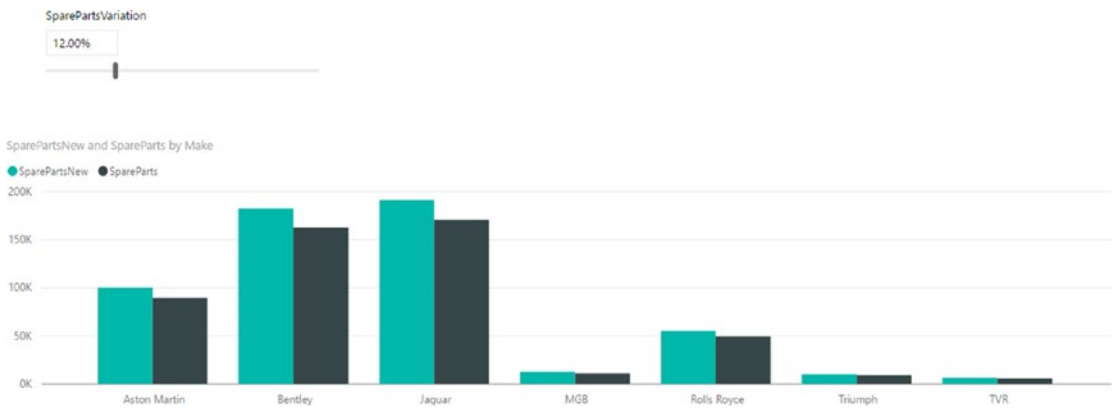


Figure 21-26. *A what-if slicer applied to a chart*

The values that you add in the table that underlies the what-if slicer can be fixed values, percentages—anything you like. Moreover, you can create as many new measures based on the what-if value as you want to.

Custom Visuals As Slicers

There are many excellent third-party visuals available that are designed to be slicers. This section provides a few pointers to some of the currently available third-party slicers. These are only a few of the slicers that you can find. Once again I have limited the selection to third-party visuals developed by Microsoft.

Timeline Slicer

The timeline slicer lets you drag the upper and lower limits of a date range. You can also define the date element (year, quarter, month, week, or day) to use interactively. You can see this in Figure 21-27.

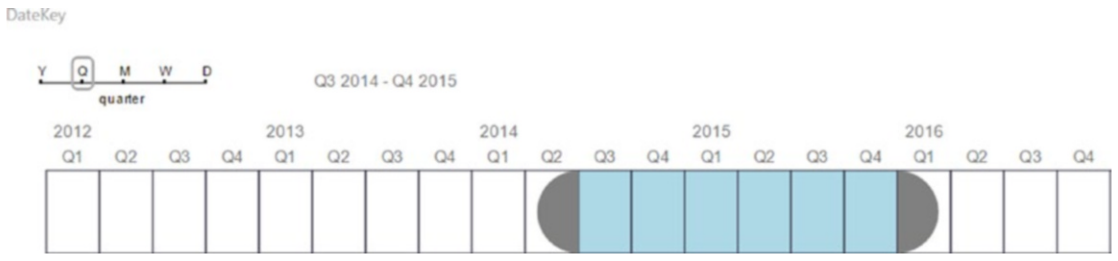


Figure 21-27. *The timeline slicer*

Timebrush Slicer

A timebrush slicer allows you to highlight a section of a time-ordered dataset. You can see this in Figure 21-28.

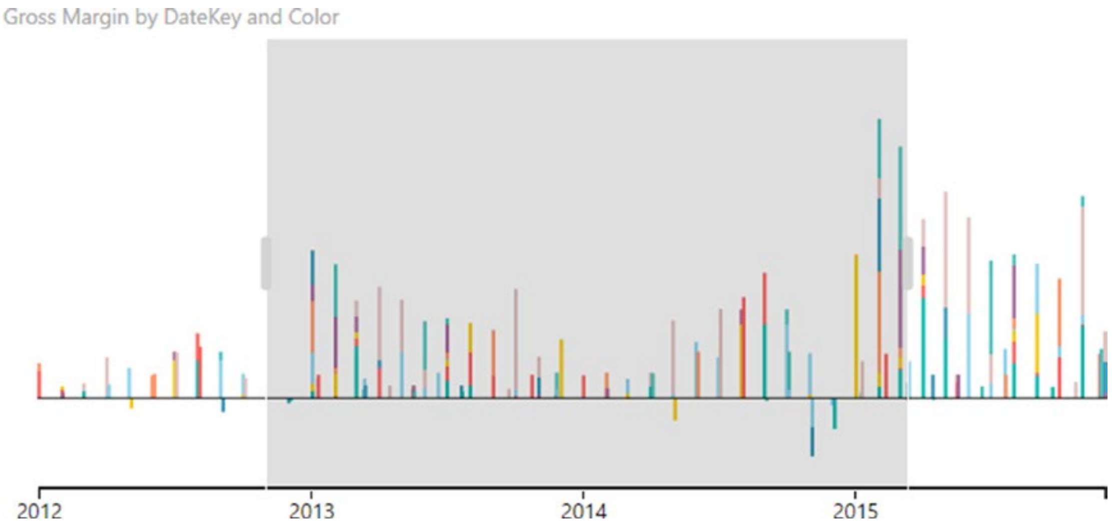


Figure 21-28. *The timebrush slicer*

Chicklet Slicer

A chicklet slicer is a slicer that you can arrange and format in a multitude of ways. You can see a simple example in Figure 21-29.

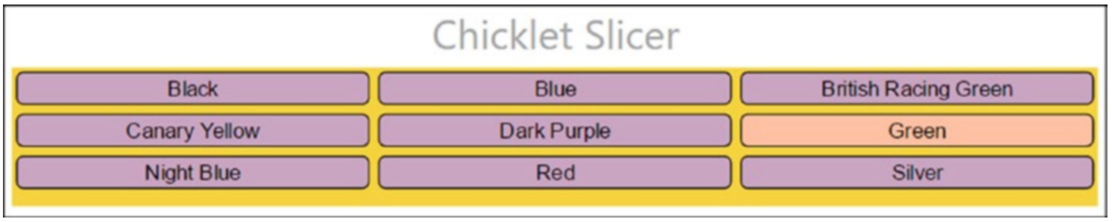


Figure 21-29. *The chicklet slicer*

This particular slicer really does add a multitude of options. I strongly recommend that you take a look at all that it can do.

Choosing the Correct Approach to Interactive Data Selection

Now that you have taken a tour of the interactive options that Power BI Desktop offers, it is worth remembering that there is a fundamental difference between slicers and chart filters:

- Slicers and chart filters apply to all visuals on the Power BI Desktop page. Unless you have tweaked the visual interactions
- Highlighting only applies to the selected chart, although it filters data in other tables and highlights the percentage of this element in other charts.

Conclusion

In this chapter, you have seen how to use the interactive potential of Power BI Desktop to enhance the delivery of information to your audience. You saw how to add slicers to a report, and then how to use them to filter out data from the visualizations it contains. Then, you learned how to highlight data in charts. Next, you saw how to use charts as interactive slicers to isolate specific elements in a presentation.

Finally, you saw how to apply what-if slicers and took a peek at some of the third-party slicers that are available.

These techniques are powerful tools that can dramatically enhance the way that you present data to an audience. Used carefully, they will help your dashboards become more powerful and even more memorable. So all that remains is for you to start applying these techniques using your own data. Then you can see how you can impress your audiences using all the interactive possibilities of Power BI Desktop.

CHAPTER 22

Enhancing Dashboards

After spending a little time working with Power BI Desktop, I can assume that you have analyzed your data. In fact, I imagine that you have been able to tease out a few extremely interesting trends and telling facts from your deep dive into the figures—and you have created the tables, charts, maps, and gauges to prove your point. To finish the job, you now want to add the final tweaks to the look and feel of your work so that it will come across to your audience as polished and professional.

Fortunately, Power BI Desktop is on hand to help with all of these final touches, too. It can propel your effort onto a higher level of presentation—without you needing to be a graphic artist—so that your audience is captivated. With a few clicks, you can

- Align and distribute objects on the report canvas.
- Add free-form text.
- Add images to a report.
- Apply a report background.
- Add basic shapes to enhance your visuals.
- Superpose objects and define how they are placed one on top of another.
- Prepare reports ready for smartphone use.

So, while not attempting to rival expensive drawing applications, Power BI Desktop can certainly add the necessary design flourishes that will help you to seize your audience's attention and convince them of the value of your analysis.

This chapter takes you through these various techniques and explains how to use them to add real pizzazz to your analysis. We will use the C:\PowerBiDesktopSamples\CH22\CarSalesDataForReports.pbix file as the source data and as an example of all the dashboards in this chapter. You will also have to download the image files to the C:\PowerBiDesktopSamples\Images folder from the Apress web site, as described in Appendix A.

Formatting Ribbons

When enhancing dashboards, you will need to use two further ribbons:

- The View ribbon
- The Format ribbon

These two ribbons are described in the following sections.

The View Ribbon

The View ribbon options are illustrated in Figure 22-1 and explained in Table 22-1.

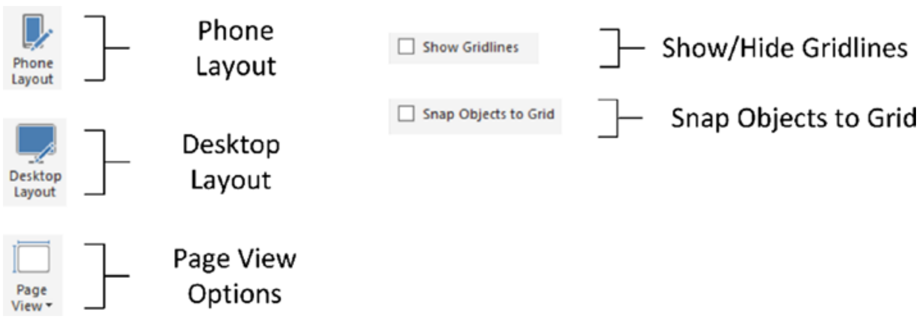


Figure 22-1. *The View ribbon*

Table 22-1. *View Ribbon Options*

Option	Description
Phone Layout	Switches from desktop layout to phone layout.
Desktop Layout	Switches from phone layout to desktop layout.
Page View: Fit to Page	Scales the display to fit the available screen dimensions.
Page View: Fit to Width	Scales the display to fit the width of the screen.
Page View: Actual Size	Displays the page at its actual size.
Show or Hide Gridlines	Displays or hides the underlying grid.
Snap Objects to Grid	Forces visuals to align to the grid (hidden or visible) or leaves them free-floating.

The Format Ribbon

The Format ribbon options are illustrated in Figure 22-2 and explained in Table 22-2.

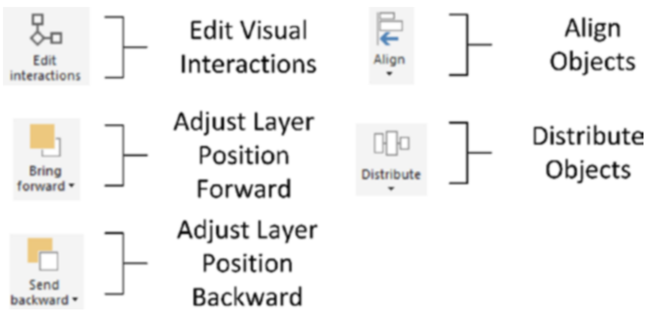


Figure 22-2. *The Format ribbon*

Table 22-2. *Format Ribbon Options*

Option	Description
Edit Interactions	Changes how visuals react when data points in other visuals are selected.
Bring Forward	Brings a visual toward the top of a layer of objects.
Send Backward	Sends a visual toward the bottom of a layer of objects.
Align Objects	Aligns two or more visuals.
Distribute Objects	Distributes three or more visuals.

Formatting the Page

Before you spend a certain amount of time and effort finalizing a presentation, it is vital to define one fundamental aspect of the output—the page format. Power BI Desktop lets you select from among the following output device formats:

- 16:9
- 4:3
- Cortana
- Letter
- Custom size

To change the page size, follow these steps:

1. Click anywhere on the report canvas (and *not* on a visual).
2. In the Visualizations pane, click the Format icon.
3. Expand the Page Size section.
4. Select one of the page size presets from the Type popup list.

Should you wish to specify your own page size, simply select Custom from the Type popup list and then enter the required page height or width in pixels. You can use the scroll arrows to set the required figure if you prefer. You can see the results of this in Figure 22-3.

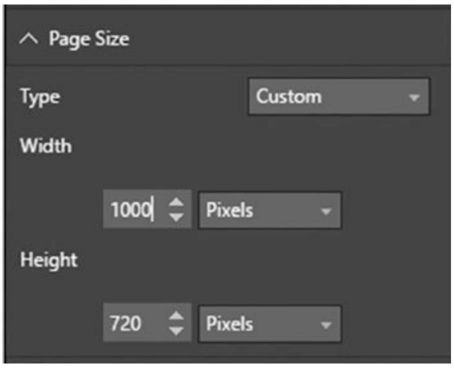

Page Size
Type: Custom
Width: 1000 Pixels
Height: 720 Pixels

Figure 22-3. *Custom page size settings*

■ **Note** Currently, the only available unit of measure is pixels.

Aligning and Distributing Visuals

There is one quick and easy tweak that is capable of adding a polished look to any dashboard. This is the simple decision to align visuals flawlessly so that you avoid giving an impression of ragged positioning. Like it or not, well-aligned visuals will help convince your audience that your analysis is valid.

To be clear, when I say *align visuals*, I include distributing visuals in the concept. So here is how you can apply both techniques to give a patina of professionalism to your dashboards.

Aligning Visuals

A set of neatly aligned elements on a page will always please an audience. What is more, it literally only takes a couple of seconds to take a set of existing visuals and to present them harmoniously. The following explains what you do:

1. Select the visuals that you want to align (Ctrl-click each one).
2. In the Format ribbon, click the Align button to display the popup menu. It will look like Figure 22-4.

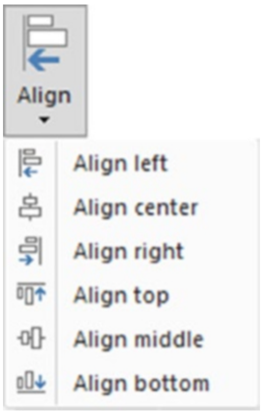


Figure 22-4. *Alignment options*

3. Select the required option.

The selected elements will be aligned along their tops, bottoms, left or right sides, or centered, depending on the choice that you made. The available alignment options are outlined in Table 22-3.

Table 22-3. *Alignment Options*

Option	Description
Align Left	Aligns all the selected visuals along their left sides.
Align Center	Centers all the selected visuals.
Align Right	Aligns all the selected visuals along their right sides.
Align Top	Aligns all the selected visuals along their top edges.
Align Middle	Aligns all the selected visuals in the middle of the elements.
Align Bottom	Aligns all the selected visuals along their bottom edges.

Distributing Visuals

One way to add a final touch that implies "attention to detail" to a dashboard is to make sure that all visuals are neatly distributed horizontally and/or vertically. This will simply guarantee that there is always the same amount of space between each element.

1. Select the visuals that you want to distribute (Ctrl-click each one).
2. In the Format ribbon, click the Distribute button to display the popup menu. It will look like Figure 22-5.

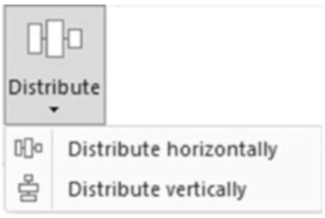


Figure 22-5. *Distribution options*

3. Select the required option.
4. The selected elements will be distributed horizontally or vertically, depending on the choice that you made. The available distribution options are outlined in Table 22-4.

Table 22-4. *Distribution Options*

Option	Description
Distribute Horizontally	Distributes a selected series of visuals along a horizontal plane.
Distribute Vertically	Distributes a selected series of visuals along a vertical axis.

■ **Note** You may need to distribute a collection of visuals both horizontally and vertically to get the best effect. You will need to carry out the two alignment operations successively to obtain this result.

Aligning to the Grid

The Power BI Desktop canvas includes a grid (like a sheet of squared paper) that you can use to align visuals and other objects. This can help you create more polished-looking presentations. To align to the grid, simply

1. Activate the View menu.
2. Select Snap Objects to Grid.

Now, whenever you move or resize an object, it will align to the grid or increase/decrease in size by one grid point (or pixel).

Displaying the Grid

If you prefer you can also display the grid to help you align objects:

1. Activate the View menu.
2. Select Show Gridlines.

You will now see a pattern of dots (one every 10 pixels) representing the hidden grid. You can see a page with a grid displayed in Figure 22-9, later in the chapter.

■ **Note** The number of pixels in the Power BI Desktop report canvas will depend on the selected page size.

Specifying the Exact Position of a Visual

A final option that you can apply when placing visuals is to specify their exact position, height, and width. This feature is particularly useful when you want to make a series of visuals the same size. Follow these steps:

1. Select the visual that you want to resize and/or reposition.
2. In the Visualizations pane, click the Format icon.
3. Expand the General section.
4. Set the X (horizontal starting) position, Y (vertical starting) position, height, and width.

Power BI Desktop uses pixels as the unit of placement as it does for the underlying grid.

Adding Text Boxes to Annotate a Report

Let's begin at the start. You have spent quite a while digging into data and have found effective ways of drawing your audience's attention to the valuable information that it contains. However, you need the one, final, cherry on the cake—a title for the report. As a title is nothing other than a text box, this is your introduction to adding, formatting, and modifying text boxes.

A text box is a floating text entity that you can place anywhere on the Power BI Desktop report canvas. They are especially useful for annotating specific parts of a dashboard.

Adding a Text Box

Adding a text box is so easy that it takes longer to describe than to do, but nonetheless, this is how you do it:

1. In the Home ribbon, click the Text Box button. An empty text box will appear on the dashboard canvas.
2. Type in the text that you want to add; it will be a title. I entered **Sales for 2017**.
3. Place the mouse pointer over either the corner or lateral central indicators of the title box and drag the mouse to resize the title box.
4. Click the text box header bar (the gray area at the top of the text box) and drag the text box to the top center of the page.
5. Click outside the title anywhere in the blank report canvas.

Figure 22-6 shows you a text box a report. Moreover, should you want to modify a text, it is as easy as clicking inside the text box and altering the existing text just as you would in, say, PowerPoint.

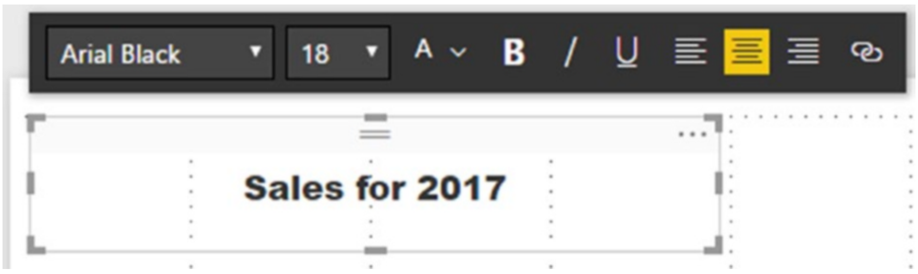


Figure 22-6. *Adding a text box to a report*

■ **Note** If a text box is too small to hold the entire text, then scroll bars are added to the text box when you select it.

Moving Text Boxes

Text boxes are a Power BI Desktop visualization like any other, and consequently, they can be moved and resized just as if they were a table or a chart. So all you have to do to move a title is to

1. Hover the mouse pointer over the text box. A container will appear indicating the text box shape.
2. Click the top bar of the text box and drag the text box elsewhere on the dashboard canvas.

Alternatively, you can select the text box and then place the mouse pointer over the edges of the text box (but not on the corner or side handles) and drag the text box to a new position.

Formatting a Text Box

A text box can be formatted specifically so that you can give it the weight and power that you want. Even though this is completely intuitive, for the sake of completeness here is a short example of what you can do:

1. Select the text box that you created previously. The Format palette will be displayed under or above the text box.
2. Select the text that you wish to modify.
3. Click the Font popup and select the font that you want to apply to the selected text.
4. Click the Font Size popup and select the size (in points) that you want to apply.
5. Apply any font attributes that you require (bold, underline, italic) by clicking the relevant icons.
6. Align the text by clicking one of the three alignment icons. This will apply to all the text in the box.

■ **Tip** Remember that you must highlight the text to format it. If you select the text box itself, then you can only alter the alignment of the text in the text box.

The formatting options for text boxes are explained in Table 22-5.

Table 22-5. *Formatting Options for Text Boxes*

Element	Icon	Description
Font	Font Segoe UI Light ▾	Lets you choose a font from those installed on the computer.
Font Size	14 ▾	Lets you choose a font size.
Theme Colors	A ⌄	Lets you choose a color.
Bold	**B**	Makes the text appear in boldface.
Italic	*I*	Makes the text appear in italics.
Underline	U	Underlines the text.
Left	≡	Places the text on the left of the text box.
Center	≡	Centers the text in the text box.
Right	≡	Places the text on the right of the text box.
Hyperlink	🔗	Adds a hyperlink for the selected text.

Adding a Hyperlink

Power BI Desktop dashboards certainly do not exist in isolation. It follows that you can use them as a starting point to link to other documents or web pages. Remember, however, that a hyperlink will *only* work once a Power BI Desktop file has been deployed to the Power BI Service in the cloud.

To add a hyperlink:

1. Select the text box that you created previously.
2. Select the text that will be the hyperlink.
3. Click the Hyperlink icon. The Format palette will expand to display the hyperlink box.
4. Enter or paste in a URL.
5. Click the Done button in the Format palette.

Removing a Hyperlink

To remove a hyperlink that you have already added:

1. Select the text box containing a hyperlink that you created previously.
2. Click inside the hyperlink. The Format palette will expand.
3. Click the Remove button.

If you prefer, you can always simply remove the text that is the link.

Deleting Text Boxes

If you want to delete a text box, then be sure to

1. Select the text box.
2. Click the popup menu for the text box (the ellipses at the top right of the text box).
3. Select Remove. The text box will be deleted.

An alternative way to delete a text box is to select the text box and press the Delete key.

■ **Note** Merely selecting and deleting the text inside the text box will not remove the text box itself; so to be sure that you do not leave any unnecessary clutter in a report, delete any unwanted and empty text boxes.

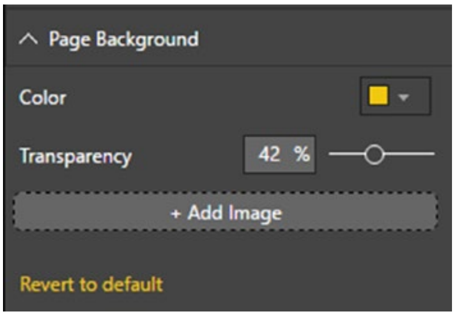
Modifying the Page Background Color

Power BI Desktop does not condemn you to presenting every report with a white background. To avoid monotony you can add a different color background to each page in a report individually with a couple of clicks.

To apply a background to a report, all you have to do is

1. Click inside the dashboard canvas, but not on any existing visuals.
2. In the Visualizations pane, click the Format icon.

3. Expand the Page Background section. The available options will look like they do in Figure 22-7.



^ Page Background
Color
Transparency 42 %

Figure 22-7. *Page background options*

4. Select a color from the palette of available colors.
5. Slide the Transparency button left or right to select a level of intensity for the chosen color.

■ **Note** You have to format every page in a report individually. You cannot Control-click to select multiple pages simultaneously.

Images

We all know what a picture is worth. Well, so does Power BI Desktop. Consequently, you can add pictures, or images, as they are generically known, to a Power BI Desktop report to replace words and enhance your presentation. The images that you insert into a Power BI Desktop report can come from the Web or from a file on a disk—either local or on an available network share. Once an image has been inserted, it is *not* linked to the source file. So if the source image changes, you will have to reinsert it to keep it up to date.

The following are some of the uses for images in Power BI Desktop:

- As a background image for a report.
- Images in tables instead of text. An example could be to use product images.
- Images in slicers. These could be flags of countries, for instance.
- Independent images—a logo, for instance, or a complement to draw the viewer's attention to a specific point.
- Images as a chart background.

Once we have looked at the types of image formats available, we will see how images can be used in all these contexts.

Image Sources

There are multitudes of image formats. Power BI Desktop accepts all of the following industry-standard image types:

- *JPEG*: This is a venerable standard image file format.
- *PNG*: This is a standard file format for Internet images.
- *BMP*: This is a standard image type produced by MS Paint, for instance.
- *GIF*: This is a venerable image format frequently used in web pages.
- *TIFF*: This is a standard format for scanned images.
- *DIB:* The Device Independent Bitmap format.
- *WEBP*: A "lossy" image file format.
- *SVGZ*: A Scalable Vector Graphics file.
- *ICO*: An image format mostly used for Windows icons.
- *SVG*: The Scalable Vector Graphics format, developed by the World Wide Web Consortium (W3C).
- *XBM*: The X-windows bitmap format.
- *PJPEG*: The progressive JPEG bitmap format.
- *PJP*: The JPEG/MIME image format.

All of these formats (and their various descendant formats that Power BI Desktop can handle) can deliver reasonable quality images that should certainly suffice for Power BI Desktop reports. However, if you attempt to insert an image that is not in a format that Power BI Desktop can handle, you will get an alert and the image will *not* be inserted.

■ **Note** When you attempt to insert an image from a file, the Open dialog filters the files so that only files with one of the acceptable extensions are visible. You can force the dialog to display other file formats, but Power BI Desktop may not be able to load them.

Adding an Image

You may still want to add completely free-form floating images to a report. However, before getting carried away with all that can be done with images, remember that Power BI Desktop is not designed as a high-end presentation package. If anything, it is there to help you analyze and present information quickly and cleanly. Inevitably, you will find that there are things that you cannot do in Power BI Desktop that you are used to doing in, say, PowerPoint. Consequently, there are many presentation tricks and techniques that you may be tempted to achieve in Power BI Desktop using images to get similar results. Indeed, you can achieve many things in a Power BI Desktop report by adding images. Yet the question that you must ask yourself is "Am I adding value to my report?" I am a firm believer that less is more in a good presentation. Consequently, although I will show you a few tricks using images, many of them go against the grain of fast and efficient Power BI Desktop report creation and can involve considerable adjustment whenever the data in a visualization changes. So I advise you not to go overboard using images to enhance your presentations unless it is really necessary.

Despite these caveats, let's add a floating, independent image to a Power BI Desktop report. In this example it is a company logo—that of Brilliant British Cars, the company whose metrics we are analyzing throughout the course of this book.

Adding an image is really simple. All you have to do is

1. In the Power BI Desktop Home ribbon, click the Image button. A classic Windows dialog will appear; it lets you choose the source image, as shown in Figure 22-8.

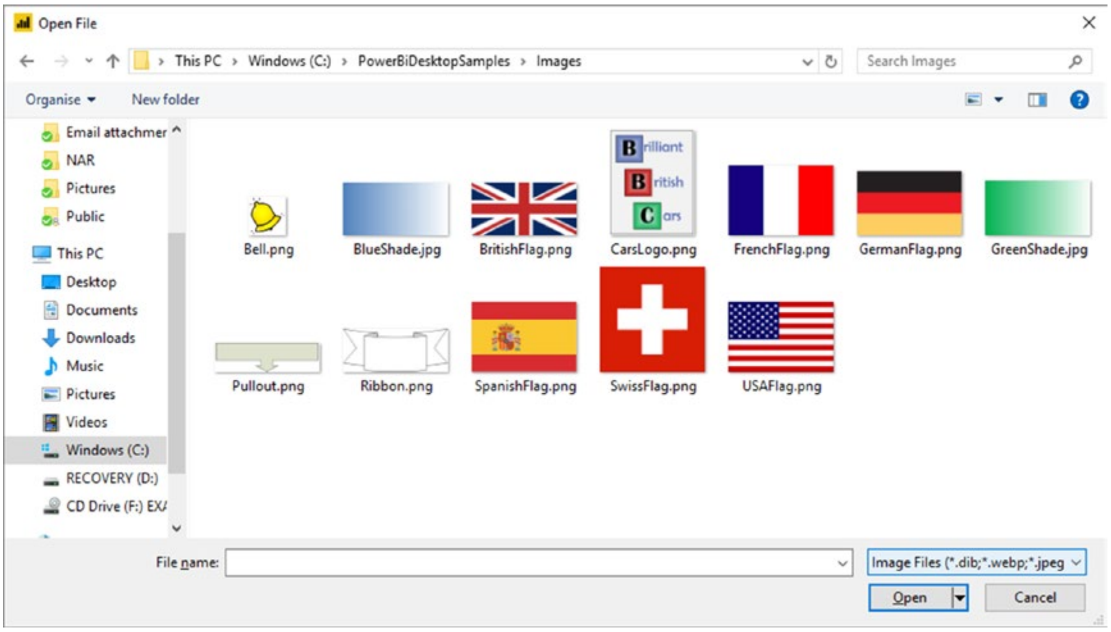


Figure 22-8. *Navigating to an image file*

2. Navigate to the directory containing the image that you wish to insert. There are several sample images in the C:\PowerBiDesktopSamples\Images folder, which you can install as described in Appendix A.
3. Click the image file. I use the example image CarsLogo.png in this example.
4. Click Open. The selected image will be loaded into the page.

5. Drag the image to the top left of the page. The dashboard will look like it does in Figure 22-9.

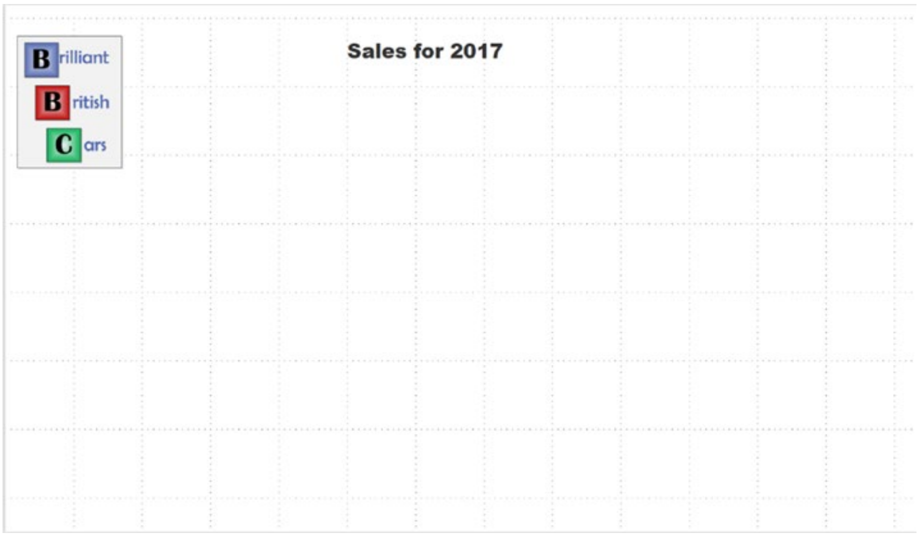


Figure 22-9. *An image added to a dashboard*

Removing an Image

To remove an image, all you have to do is

1. Select the image.
2. Click the popup menu for the image (the ellipses at the top right of the text box).
3. Select Remove. The image will be removed from the report. The original image file, of course, will not be affected in any way.

An alternative way to delete an image is to select the image and press the Delete key. The selected image will disappear from the report.

Resizing Images

An image is just like any other visual in a Power BI Desktop dashboard in that any of these elements can be moved and resized in exactly the same way. Simply do the following:

1. Select the image that you want to resize.
2. Place the mouse pointer over either the corner or lateral central indicators of the image and drag the mouse to resize the image.

Formatting Images

Once you have added an image, it can be tweaked to some extent in Power BI Desktop. The following are the parameters that you can modify:

- Image scaling
- Image title
- Image background
- Image border
- Aspect ratio
- Exact position and size

All of these modifications except for the first are common to most of the visualizations in Power BI Desktop. Indeed, you have come across some of them several times already. Consequently, I will not waste your time repeating things that you already know, or can find elsewhere in this book.

Image *scaling* is completely new, however. Here is how you can alter the way that an image is altered if you resize it:

1. Select the image whose scaling you want to modify.
2. In the Visualizations pane (which has switched automatically to show only formatting options), expand the Scaling section.
3. In the Scaling popup, select Fill.

The image will adjust to take up all the available space in the image placeholder.

The three available image scaling options are explained in Table 22-6.

Table 22-6. *Image Scaling Options*

Option	Description
Normal	The image maintains its height-to-width ratio (whatever the Lock Aspect setting) and resizes to fill the image placeholder as well as it can.
Fill	The image fills the image placeholder but distort the image.
Fit	The image fills the image placeholder but cuts off parts of the image rather than distort the image.

Background Images

One major, and frequently very striking, use of images is as a background to a report—and possibly even to a whole series of reports. So, let's take a look at how to use images for report backgrounds.

Adding a Background Image

Before anything else, you need to add a background image. This is, once again, extremely simple:

1. Click the dashboard canvas for the page where you want to add a background image. Make sure that no visuals are selected.
2. In the Visualizations pane, click the Format icon.
3. Expand the Page Background section.
4. Click Add Image. A classic Windows dialog will appear; it lets you choose the source image.
5. Navigate to the directory containing the image that you wish to insert. In this example, it will be C:\PowerBiDesktopSamples\Images.
6. Click the image file. I will use the example image GreenShade.jpg in this example.
7. Click Open. The selected image will be loaded into the page.
8. In the Image Fit popup list, select Fit.

The selected image will cover the entire page behind any visuals.

There are three possible ways of adjusting the size of an image. These are given in Table 22-7.

Table 22-7. *Background Image Sizing Options*

Option	Description
Normal	The image stays the size it was created.
Fit	The image expands (and may be deformed) to cover the entire dashboard area.
Fill	The image is expanded proportionally to cover as much of the dashboard area as possible.

Moreover, you can alter a background image's intensity once it has been added to a page. To do this:

1. With the Page Background section of the Format panel expanded, slide the Transparency slider left or right.

■ **Note** There is nothing that you can do to resize an image manually in Power BI Desktop. If you need an image to be a certain size, you have to create it at the exact required size in an image creation application.

Some Uses for Independent Images

The limits of what images can do to a report are only those of your imagination, so it is impossible to give a comprehensive list of suggestions. Nonetheless, the following are a few uses that I have found for free-form images:

- *Company logos*, as we have just seen.
- *Images added for a purely decorative effect*. I would hesitate before doing this at all, however, as it can distract from the analysis rather than enhance it. Nonetheless, at times, this may be precisely what you want to do (to turn attention away from some catastrophic sales figures, for instance). So add decoration if you must, but please use sparingly!
- *To enhance the text in a text box* by providing shading that is in clear relief to the underlying image or background.
- *As a background* to a specific column in a table. Be warned, however, that the image cannot be made to move with a column if it is resized.

Adding Shapes

Sometimes your figures may need just a little help to stand out from the crowd. Maybe a small set of visuals (gauges or cards perhaps) are best grouped together. Whatever the need, Power BI Desktop can add a few final touches to your dashboards by adding one or more shapes to a page.

The following are built-in shapes that you can choose from:

- Rectangle (which means squares, too)
- Oval (including circles)
- Line
- Triangle
- Arrow

Let's suppose that you want to add a decorative arrow to a dashboard to draw the reader's attention to a specific figure. The following explains how you can do this:

1. In the Home ribbon, click the Shapes button to see a popup list of available shapes. You can see this in Figure 22-10.

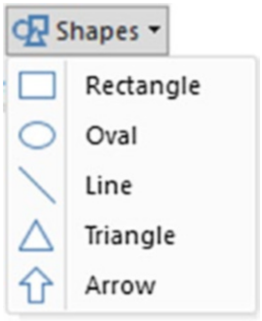


Figure 22-10. *The Shapes popup*

2. Select Rectangle. A square will be added to the dashboard canvas.
3. Resize the square in the same way that you would resize a chart or an image. If you use the top, bottom, or side handles, then the square will become a rectangle.

As the other available shapes can be inserted in exactly the same way, I will not explain them individually, but will let you have some fun by adding different shapes to a page to see how they can enhance a dashboard.

Formatting Shapes

You can tweak the following aspects of shapes:

- Lines
- Fill
- Rotation
- Title
- Background
- Aspect ratio
- Exact position and size (X and Y coordinates)

As it is only the first three that are new as far as formatting visuals is concerned, let's see them in action.

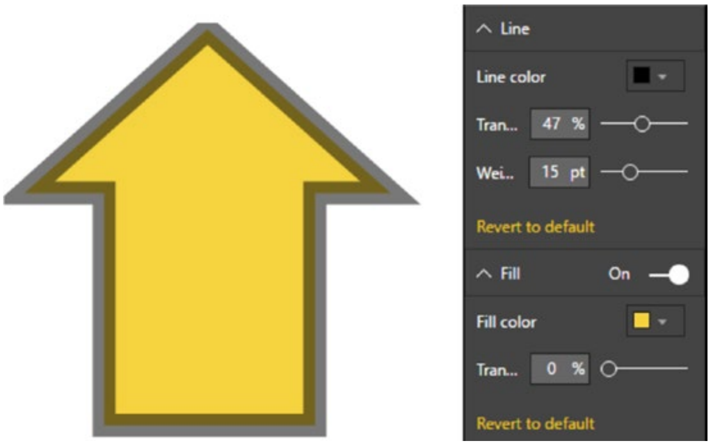
Lines and Fill Color in Shapes

When we say “lines” in shapes, we are really talking about the exterior boundary of the shape. Power BI Desktop lets you alter its

- Color
- Thickness (or weight, as it is known)
- Transparency

As well as you can set the fill color of the shape. Here is how you can alter basic characteristics for any of the available shapes:

1. Select the shape to format.
2. In the Visualizations pane (that now displays only the Format Shape options), expand the Line section.
3. Select a color for the line from the popup palette of available colors. The exterior line of the shape will change color.
4. Move the Weight slider to the right to adjust the line thickness. The exterior line of the shape will grow thicker.
5. Move the Transparency slider to the left to adjust the intensity of the color.
6. Expand the Fill section.
7. Select a color for the fill from the popup palette of available colors. The interior of the shape will change color.
8. Move the Transparency slider to the right to adjust the intensity of the color. The Visualizations pane and the shape will look like they do in Figure 22-11.



Line
Line color ■
Tran... 47 %
Wei... 15 pt
Revert to default
Fill On
Fill color ■
Tran... 0 %
Revert to default

Figure 22-11. *Formatting a shape*

Shape Rotation

Some shapes need to be rotated to point in the right direction for the effect that you are trying to produce. The shapes that generally need adjusting in this way are arrows and triangles-though this technique can be applied to any shape. To rotate a shape:

1. Select the shape to rotate.
2. In the Visualizations pane (that now displays only the Format Shape options), expand the Rotation section.
3. Drag the Rotation slider left and/or right to set the arrow to point in the direction that you want. Alternatively, you can enter the exact rotation value in the Rotation box to the left of the slider. The shape and Visualizations pane will look like they do in Figure 22-12.

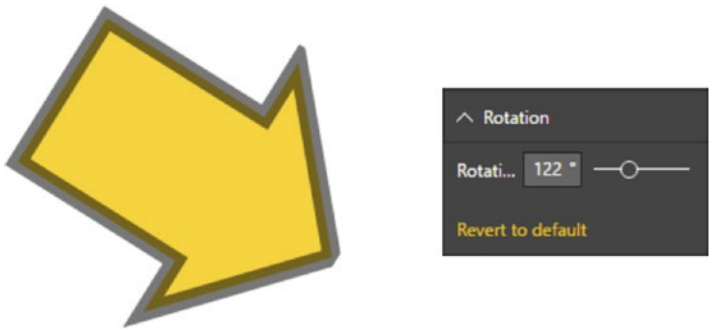


Figure 22-12. *Rotating a shape*

It is worth noting that you can enter a precise rotation value in the Rotation box if you prefer. This is particularly useful when swiveling a shape through multiples of 45 or 90 degrees.

■ **Note** When adding an ellipse or a rectangle, you will discover that Power BI Desktop begins by creating these as, respectively, circles and squares. To ensure that they stay perfectly formed (if that is what you want), ensure that the Lock Aspect formatting option is set to On before you resize the shape.

Removing Shapes

To remove a shape, all you have to do is

1. Select the shape to remove.
2. Click the popup menu for the shape (the ellipses at the top right of the text box).
3. Select Remove. The shape will be deleted.

An alternative way to delete a shape is to select the shape and press the Delete key. The selected shape will disappear from the report.

Standardizing Shapes

If you are adding the final decorative (and hopefully also illustrative touches) to a dashboard, you might want to make a series of shapes all look identical. After all, you never want a dashboard to look like a patchwork quilt. A trusted tool from MS Office is available to help you rationalize dashboards in this way-the Format Painter.

1. Select the shape that will serve as the model for other shapes.
2. Click the Format Painter button in the Home ribbon.
3. Click the shape that you want to see formatted identically to the first shape.

■ **Note** You cannot double-click the Format Painter to apply the same format several times to different shapes as you can when formatting in MS Word or Excel.

Organizing Visuals on the Page

A complex dashboard can consist of many elements (though hopefully not so many that you end up confusing your public). This is where some elementary polishing of the final appearance can help. Put simply…

- An audience gives more credence to a slickly organized dashboard.
- Clarity of layout is interpreted as clarity of thought—and so adds to the credibility of the facts and figures that you are presenting.
- Good aesthetics add to, rather than detract from, the points that you are making.

So to finish our tour of the ways that you can finalize and perfect your dashboards, you need to learn how to adjust the way that visuals (whether they are tables, charts, gauges, images, or shapes) relate to one another on the page; this essentially means layering visuals vertically.

Layering Visuals

As a report gets more complex, you will inevitably need to arrange the elements that it contains not only side by side, but also one on top of the other. Power BI Desktop lets you do this simply and efficiently.

As an example of this, let’s create a chart with another chart superposed on it:

1. Create a donut chart using the following two fields:
    a. CostPlusSpares (from the InvoiceLines table)
    b. CountryName (from the Countries table)
2. Create a (clustered) column chart using the following two fields:
    a. Make (from the Stock table)
    b. RatioNetMargin (from the InvoiceLines table)

3. Order the column chart by RatioNetMargin, in descending order.
4. In the donut chart, add a legend and set it to appear on the right. Also, set the Detail Labels option to Off. Set the Donut Background option to Off.
5. Place the donut chart in the top right-hand corner of the bar chart.
6. With the donut chart selected, choose Bring Forward ➤ Bring To Front from the Power BI Desktop Format ribbon.

Your composite chart should look like Figure 22-13. Bringing the donut chart to the front means that the gridlines for the column chart are now under the donut chart.

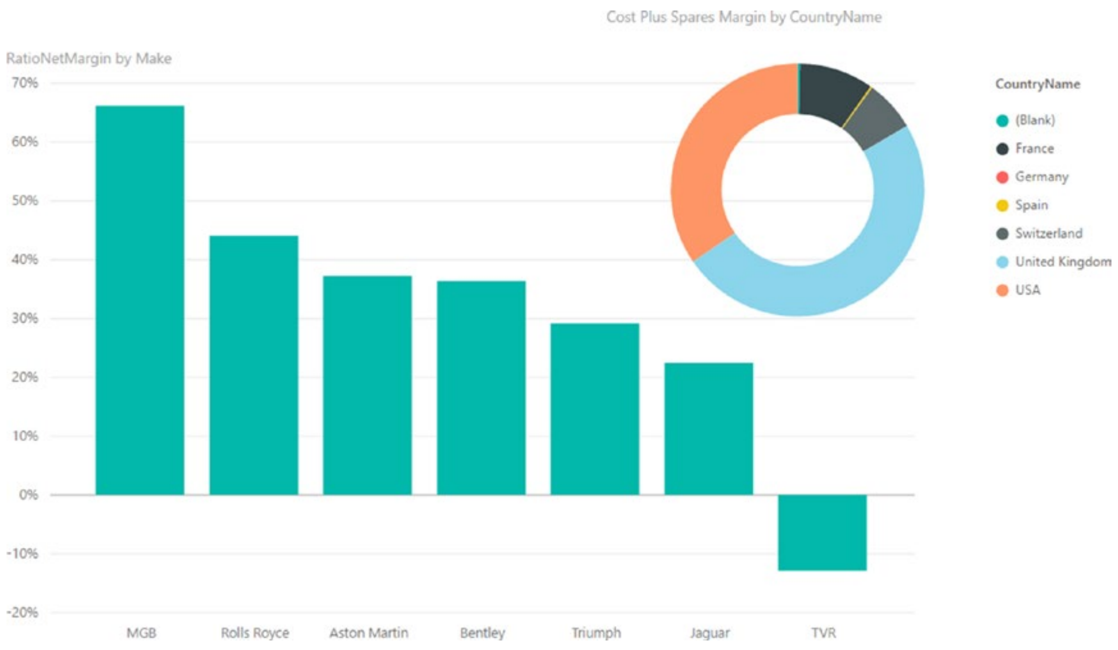


Figure 22-13. *Layering charts*

This technique is particularly useful when you are adding independent images as was described in the previous section. It is also handy when you are combining elements such as images and text boxes, as you will see in the next section.

Drill-Through

A new function that has recently been added to Power BI Desktop is the ability to drill through to another page based on a filter criteria. This enables you to click an item and request detailed information about one aspect of the data.

Drill-through is a function that requires at least two pages in a report:

- The source page that contains data about many elements
- A destination (or detail) page that is focused on a single element

Let's see this in action. To avoid pages of instructions, I will only ask that you create two very simple report pages. In practice, of course, your pages could be much more complicated than these.

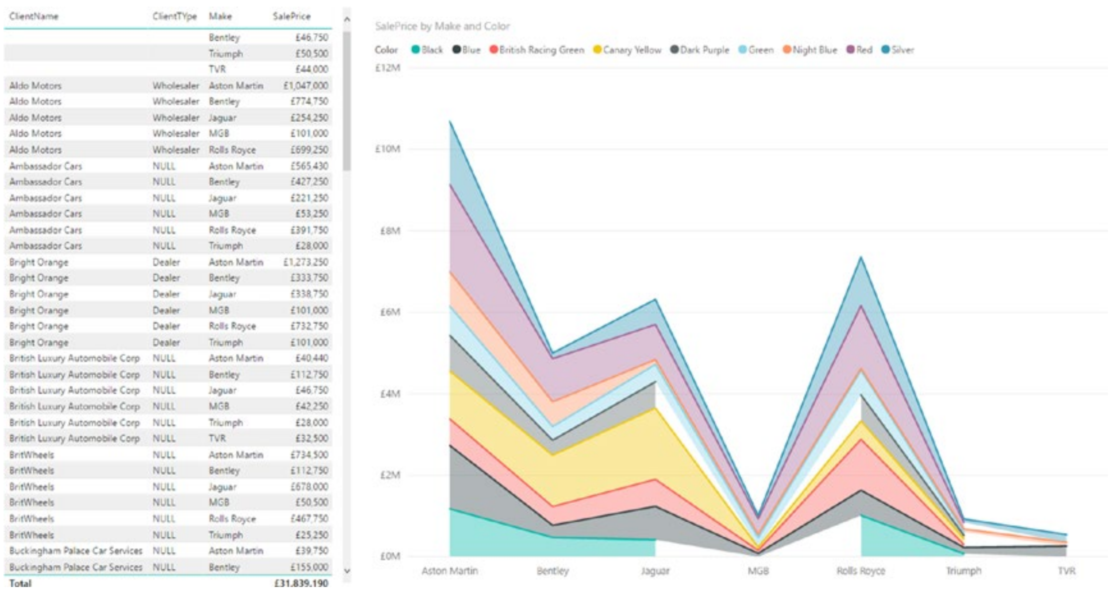
■ **Note** If you merely want to test drill-through, then the sample file C:\PowerBiDesktopSamples\CH22\DrillThroughExample.pbix contains the necessary elements for you to do this.

The Source Page

This is a perfectly ordinary Power BI Desktop report. For argument's sake, let me specify it like this:

1. Open the Power BI Desktop file C:\PowerBiDesktopSamples\CH22\CarSalesDataForReports.pbix.
2. Rename the page **Make Details**.
3. On this page, create a table using the following fields, in this order:
    a. Client Name
    b. ClientType
    c. Make
    d. SalePrice
4. Create a stacked area chart using the following fields:
    a. Make
    b. Color
    c. SalePrice

5. Adjust the size and position of these two visuals until you have something like the report shown in Figure 22-14.



ClientName	ClientTYpe	Make	SalePrice
		Bentley	£46,750
		Triumph	£50,500
		TVR	£44,000
Aldo Motors	Wholesaler	Aston Martin	£1,047,000
Aldo Motors	Wholesaler	Bentley	£774,750
Aldo Motors	Wholesaler	Jaguar	£254,250
Aldo Motors	Wholesaler	MGB	£101,000
Aldo Motors	Wholesaler	Rolls Royce	£699,250
Ambassador Cars	NULL	Aston Martin	£565,430
Ambassador Cars	NULL	Bentley	£427,250
Ambassador Cars	NULL	Jaguar	£221,250
Ambassador Cars	NULL	MGB	£53,250
Ambassador Cars	NULL	Rolls Royce	£391,750
Ambassador Cars	NULL	Triumph	£28,000
Bright Orange	Dealer	Aston Martin	£1,273,250
Bright Orange	Dealer	Bentley	£333,750
Bright Orange	Dealer	Jaguar	£338,750
Bright Orange	Dealer	MGB	£101,000
Bright Orange	Dealer	Rolls Royce	£732,750
Bright Orange	Dealer	Triumph	£101,000
British Luxury Automobile Corp	NULL	Aston Martin	£40,440
British Luxury Automobile Corp	NULL	Bentley	£112,750
British Luxury Automobile Corp	NULL	Jaguar	£46,750
British Luxury Automobile Corp	NULL	MGB	£42,250
British Luxury Automobile Corp	NULL	Triumph	£28,000
British Luxury Automobile Corp	NULL	TVR	£32,500
BritWheels	NULL	Aston Martin	£734,500
BritWheels	NULL	Bentley	£112,750
BritWheels	NULL	Jaguar	£678,000
BritWheels	NULL	MGB	£50,500
BritWheels	NULL	Rolls Royce	£467,750
BritWheels	NULL	Triumph	£25,250
Buckingham Palace Car Services	NULL	Aston Martin	£39,750
Buckingham Palace Car Services	NULL	Bentley	£155,000
Total			£31,839,190

SalePrice by Make and Color
Color ● Black ● Blue ● British Racing Green ● Canary Yellow ● Dark Purple ● Green ● Night Blue ● Red ● Silver
£12M
£10M
£8M
£6M
£4M
£2M
£0M
Aston Martin
Bentley
Jaguar
MGB
Rolls Royce
Triumph
TVR

Figure 22-14. *A drill-through source page*

The Destination Page

This page will have a couple of specific tweaks added-as you will see. However, it must be designed to show data that is relevant to the single element that you want it to display. In this example that means information about a *make* of vehicle.

1. Add a new page to the existing report.
2. Rename this page **Make Details**.
3. Add a card using the field Make.
4. In the Formatting pane for the card, set the Category Label to Off.
5. Add an area chart using the following fields:
    a. Color
    b. TotalCost
    c. SalePrice
6. Add a filled map using the following fields:
    a. CountryName
    b. SalePrice

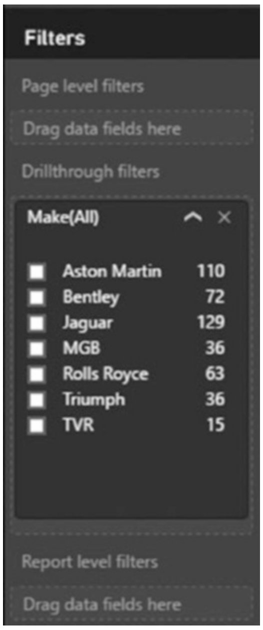
7. Add a table map using the following fields:

    a. ClientName

    b. CostPrice

    c. Gross Margin

8. Click the report canvas (outside any visual) and drag the Make field into the Filters area of the Visualizations pane. Place this field in the Drillthrough Filters well. You can see the result in Figure 22-15.

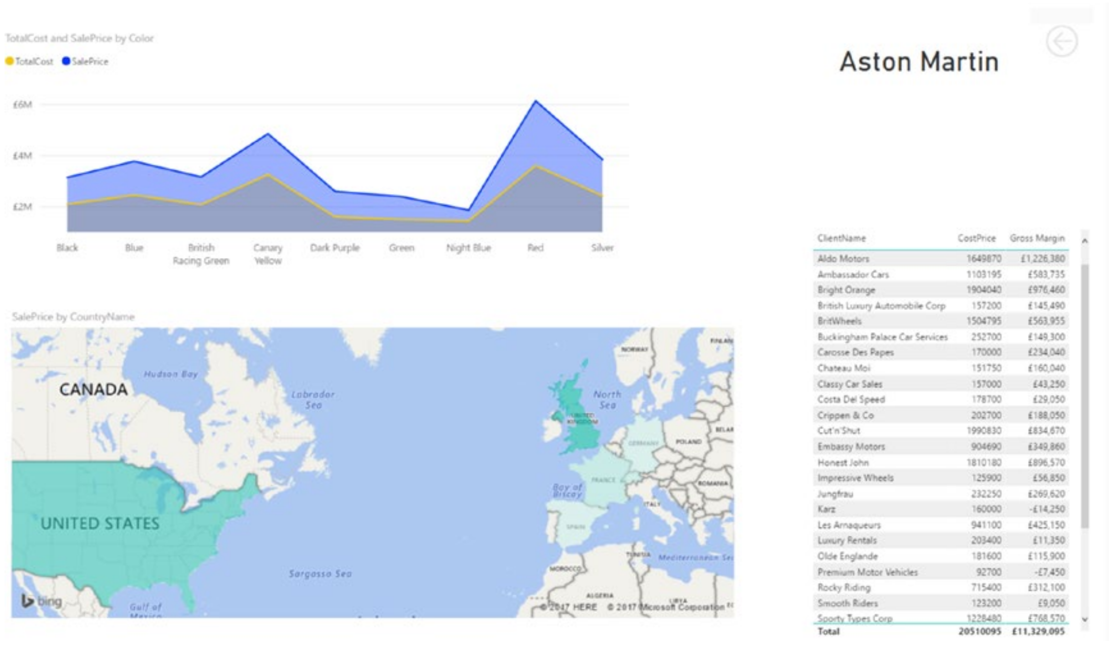


Aston Martin	110
Bentley	72
Jaguar	129
MGB	36
Rolls Royce	63
Triumph	36
TVR	15

Figure 22-15. *The Drillthrough Filters well*

9. A “Back” button will be added to the report. Format it as you think fit using the techniques described for shapes earlier in this chapter.

10. Adjust the size and position of all the visuals until you have something like the report shown in Figure 22-16.



ClientName	CostPrice	Gross Margin
Aldo Motors	1649870	£1,226,380
Ambassador Cars	1103195	£583,735
Bright Orange	1904040	£976,460
British Luxury Automobile Corp	157200	£145,490
BritWheels	1504795	£563,955
Buckingham Palace Car Services	252700	£149,300
Carosse Des Papes	170000	£234,040
Chateau Moi	151750	£160,040
Classy Car Sales	157000	£43,250
Costa Del Speed	178700	£29,050
Crippen & Co	202700	£188,050
Cut'n'Shut	1990830	£834,670
Embassy Motors	904690	£349,860
Honest John	1810180	£896,570
Impressive Wheels	125900	£56,850
Jungfrau	232250	£269,620
Karz	160000	-£14,250
Les Arnaqueurs	941100	£425,150
Luxury Rentals	203400	£11,350
Olde Englande	181600	£115,900
Premium Motor Vehicles	92700	-£7,450
Rocky Riding	715400	£312,100
Smooth Riders	123200	£9,050
Sporty Types Corp	1228480	£768,570
Total	**20510095**	**£11,329,095**

Figure 22-16. *A drill-through destination page*

Applying Drillthrough

You can now use your drillthrough report.

1. Go to the Sales page.
2. Right-click any line in the table or any data point in the chart. You will see the popup menu that is shown in Figure 22-17.

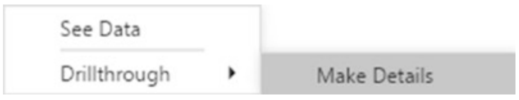


Figure 22-17. *The drill-through context menu*

3. Select Drillthrough ➤ Make Details.
4. Control-click the back button to return to the source report.

The Make Details report will be displayed and will filter the data to display the make of the row or data point that you right-clicked.

The Back Button

Power BI Desktop created a back button automatically on the drill-through report (the "destination" report page). However, you do not have to use this button if you do not want to. In fact, any shape or image can become a back button. Here is how you set this:

1. Add an image or shape to a drill-through "destination" report.
2. Select the image or shape.
3. In the Visualizations pane, set the Back Button option to On. You can see this in Figure 22-18.

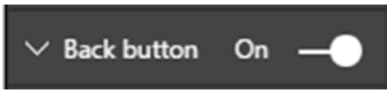


Figure 22-18. *Specifying that a visual acts as a back button*

■ **Note** Once a report is deployed to PowerBI.com (that you will discover in the next chapter), simply clicking the back button will return the focus to the "source" report.

Phone Layout

More and more information workers need their data on the move. This means consulting Power BI dashboards on their mobile phones. To make this easier, the Power BI team at Microsoft has added a valuable feature to Power BI Desktop—phone layout.

This technique lets you take an existing report and define how each of the visuals that make up the dashboard are displayed on a smartphone. There are several advantages to this approach:

- You do not have to create duplicate reports, one for a computer screen and another for a phone, and then update them both.
- You do not have to include all the visuals in a dashboard on the phone version.
- Any changes in a visual for the dashboard are reflected in the corresponding phone layout.

As a simple example of this, here is how to create a phone report from the dashboard containing the superposed charts that you just created:

1. Open, or create, a Power BI Desktop dashboard.
2. In the View ribbon, click the Phone Layout button. You will see a screen like the one shown in Figure 22-19.

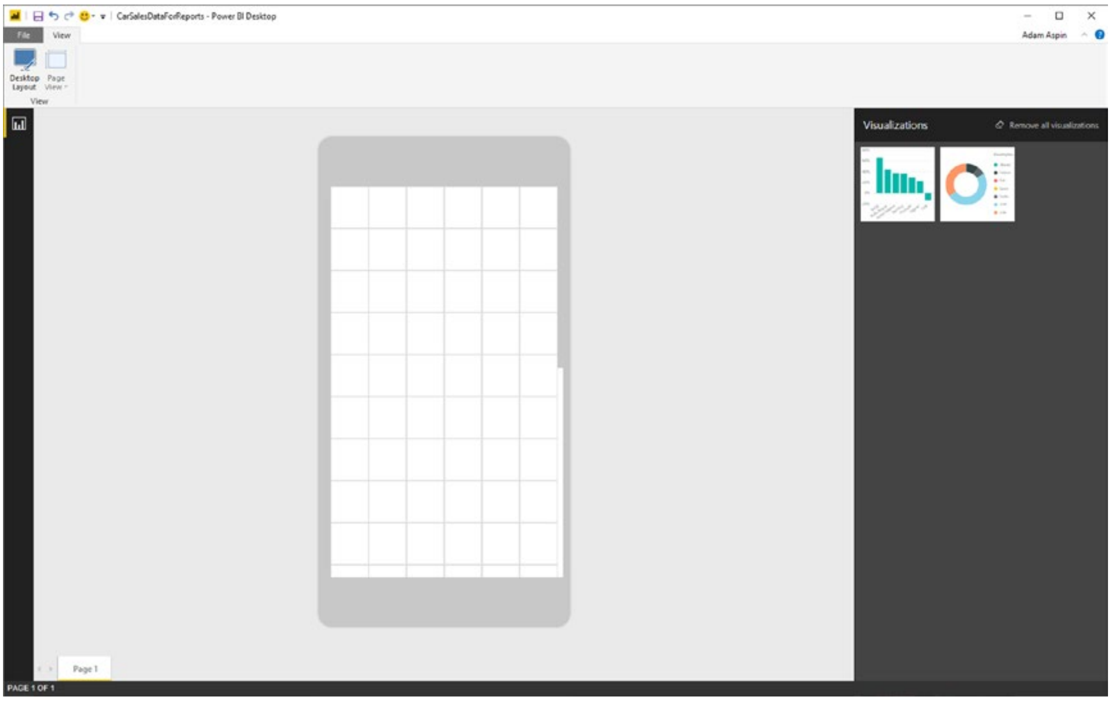


Figure 22-19. *Phone layout*

3. Drag the first visual that you want to use in a phone version of this report to the phone image.
4. Resize the visual so that it covers the required phone screen real estate.

5. Add any other visuals that you wish to use and position them to suit your requirements. The phone layout screen could look like Figure 22-20.

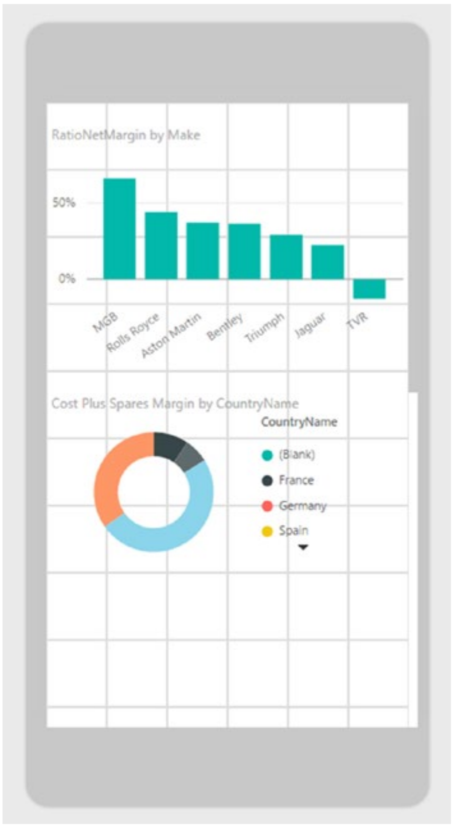


Figure 22-20. *A completed phone layout*

6. Switch back to desktop layout by clicking the Desktop Layout button.

Now, once this report is published to the Power BI Service (or an on-premises Power BI Report Server), any mobile users will see the phone layout on their smartphones.

■ **Note** If a visual is not completely visible in phone layout, you will see ellipses and a small down-facing triangle in the visual to indicate that visibility may be an issue. You can see this in the donut visual in Figure 22-20.

Report Themes

You can define your own set of colors that will be applied automatically to tables, matrices, and data in charts if you so wish. This involves creating a simple file in a specific format that you then load into Power BI Desktop. This file must contain the following elements:

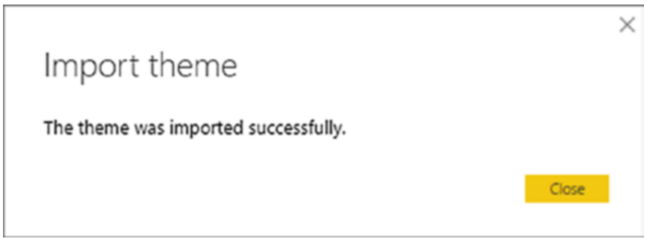
- A name for the theme
- A set of data colors enclosed in square brackets and comma-separated
- A background element
- A foreground element
- An accent element for tables

You can see a sample file in step 1 of the following example:

1. Create a text file named BrilliantBritishCars.json that contains the following code:

```
{
    "name": "Brilliant British Cars",
    "dataColors": ["#000000", "#0000CC", "#006600", "#FFFF00", "#F90FF9",
                           "#5BD078", "#000099", "#FF0000", "#C0C0C0"],
    "background":"#99CCFF",
    "foreground": "#0000FF",
    "tableAccent": "#FF0000"
}
```

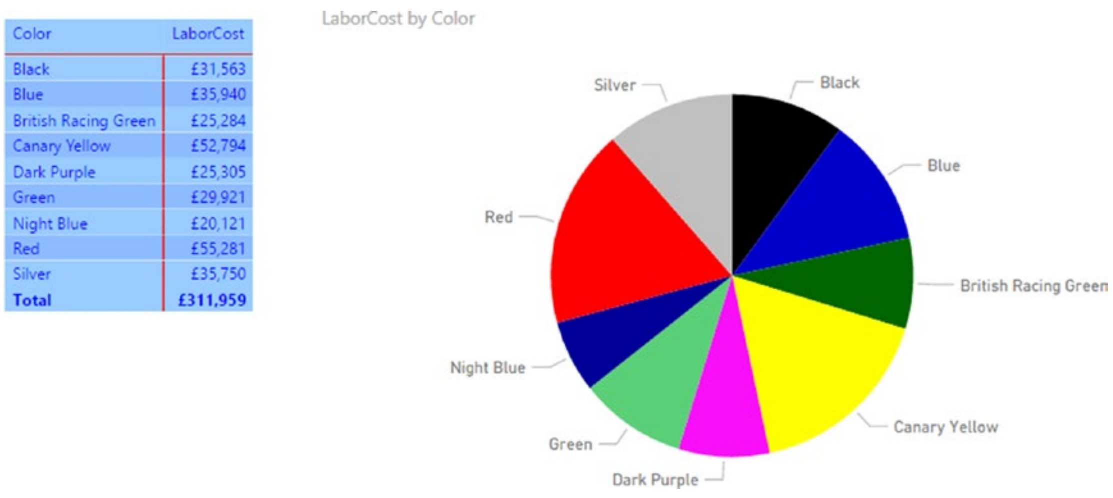
2. In the Home ribbon, click Switch Theme ➤ Import Theme and browse to the file that you just created.
3. Click Open. A dialog like the one in Figure 22-21 will confirm that the theme file has been loaded successfully.


Import theme

The theme was imported successfully.

Figure 22-21. *The Import Theme dialog*

The background, foreground, and table accent colors will now apply to every table and matrix in the report. Moreover, the data colors will, by default, be those specified in the themes file. You can see this in Figure 22-22.



Color	LaborCost
Black	£31,563
Blue	£35,940
British Racing Green	£25,284
Canary Yellow	£52,794
Dark Purple	£25,305
Green	£29,921
Night Blue	£20,121
Red	£55,281
Silver	£35,750
Total	**£311,959**

Figure 22-22. *Applying a theme to a report*

This theme file is also available in the folder C:\PowerBiDesktopSamples\CH22 (assuming that you have downloaded the sample files from the Apress web site).

■ **Note** To remove the theme that you have just loaded and revert to the default theme, click Switch Theme ➤ Default Theme in the Home ribbon.

Report themes are very much a work in progress in Power BI Desktop, and are currently in a constant state of flux. I imagine that the structure of the JSON file will have evolved considerably by the time that this book is published. So I cannot, with any certainty, delve any deeper into the question of report branding in this edition of this book.

Exporting Data from a Visualization

Finally, it is worth noting that you can export the data that underlies a visualization. This applies to any visual, from tables to charts. All you have to do is

1. Click the menu for the visual—the ellipses at the top right of the chart, table, or map. Then select Export Data from the popup menu.
2. Select a destination directory and file name
3. Click OK and the relevant data will be exported. The only current export format is a CSV file.

■ **Note** Any existing filters are applied when the data is exported. This means that, potentially, only a subset of the data will be exported to the CSV file.

Conclusion

In this chapter, you saw how to push the envelope when using Power BI Desktop to deliver particularly compelling presentations. You saw how adding images can turbocharge the impression that your analysis gives when you add graphic elements to tables and slicers. And, used sparingly, images, shapes, and free-form text elements can draw your audience's attention to the most salient features of your presentation. So now it is up to you to use these powerful Power BI Desktop features to deliver some really compelling interactive analyses to your audience.

In this chapter you also saw how to take existing reports and prepare them for display on a smartphone. With very little extra effort you were able to ensure that a complex report becomes perfectly adapted for a mobile device.

Finally, you learned how to create your own "branding" for Power BI Desktop reports. This lets you create a standardized look and feel across a range of reports automatically.

So, now that you have finished your tour of report creation with Power BI Desktop, it only remains to discover how to share your work with colleagues and friends. You will find this in the next-and final-chapter.

CHAPTER 23

PowerBI.com

You are now approaching the end of your journey into the world of self-service business intelligence with Power BI. Up until now in this book, you have seen how to use Power BI Desktop to prepare and visualize your data. Now, all that remains is to learn how to share your insights with your colleagues. This is where PowerBI.com comes into the frame.

PowerBI.com is a cloud-based online service from Microsoft that lets you…

- Share your Power BI Desktop BI files in the cloud. This will allow your co-workers to view and interact in real time with Power BI Desktop reports in a browser window.
- Create new reports in the cloud using the data from Power BI Desktop files that you have already loaded into PowerBI.com.
- Configure any Power BI Desktop files that you have loaded into PowerBI.com so that they connect to on-premises data and so they refresh the data that they contain from onsite data sources at regular intervals. This way you can be sure that your colleagues are always using the most recent available data.
- Use the new Power BI app for mobile devices to view and interact with Power BI Desktop reports on tablets and mobile phones.

However, the truly amazing thing about PowerBI.com is that it is absolutely *free* (for up to 1GB of data). If this threshold is too low (and to access many other possibilities, especially where group collaboration are concerned) you can upgrade to the PowerBI.com Enterprise service and raise the limit to 10GB of data (as well as obtaining other advantages) for $9.99 per month. At least these were the prices when this book went to press.

I have to add that PowerBI.com is a vast product in its own right, and worthy of a book in itself. So I have to warn you that in this chapter, you are only taking a quick look at some of its features; you will not examine everything that this service can do in detail.

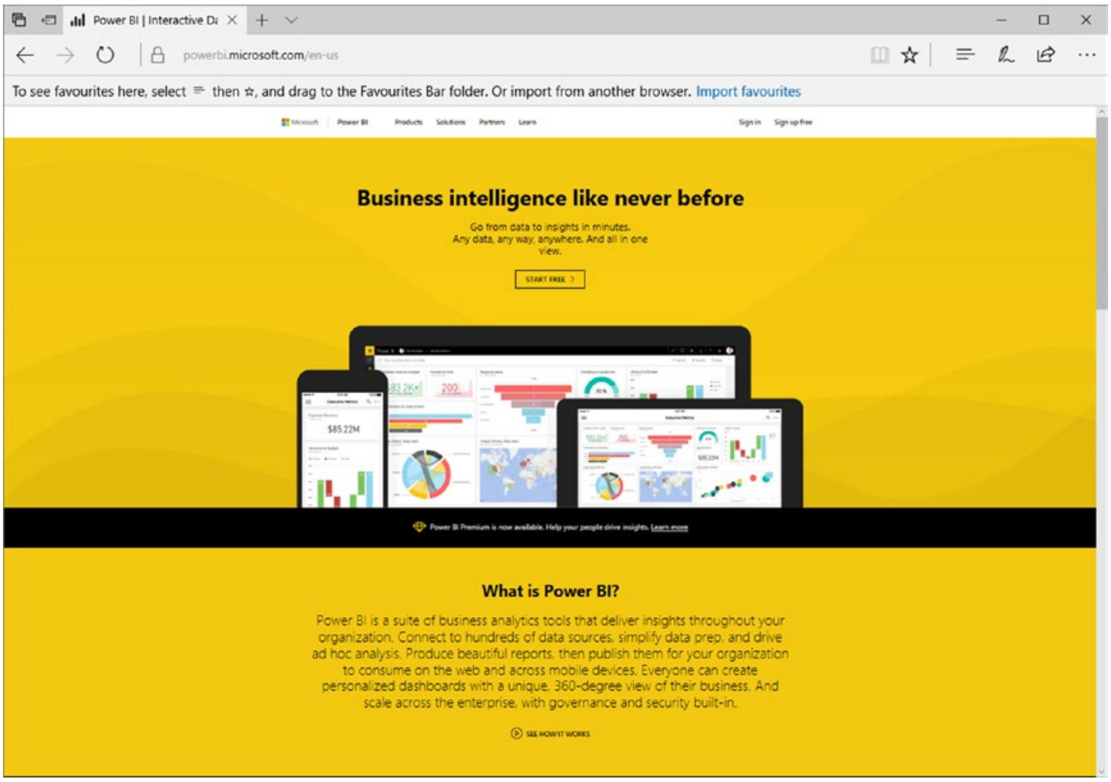
Publishing Reports to PowerBI.com

The best way to appreciate PowerBI.com is to see it in action. The following sections explain how to create a PowerBI.com account and upload a report created with Power BI Desktop.

Creating a Power BI Account

Before you can use PowerBI.com, you need to create a Power BI account, as follows:

1. In your browser, navigate to `https://powerbi.microsoft.com`. You should see a web page like the one shown in Figure 23-1.



To see favourites here, select ≡ then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites

Business intelligence like never before

Go from data to insights in minutes.
Any data, any way, anywhere. And all in one view.

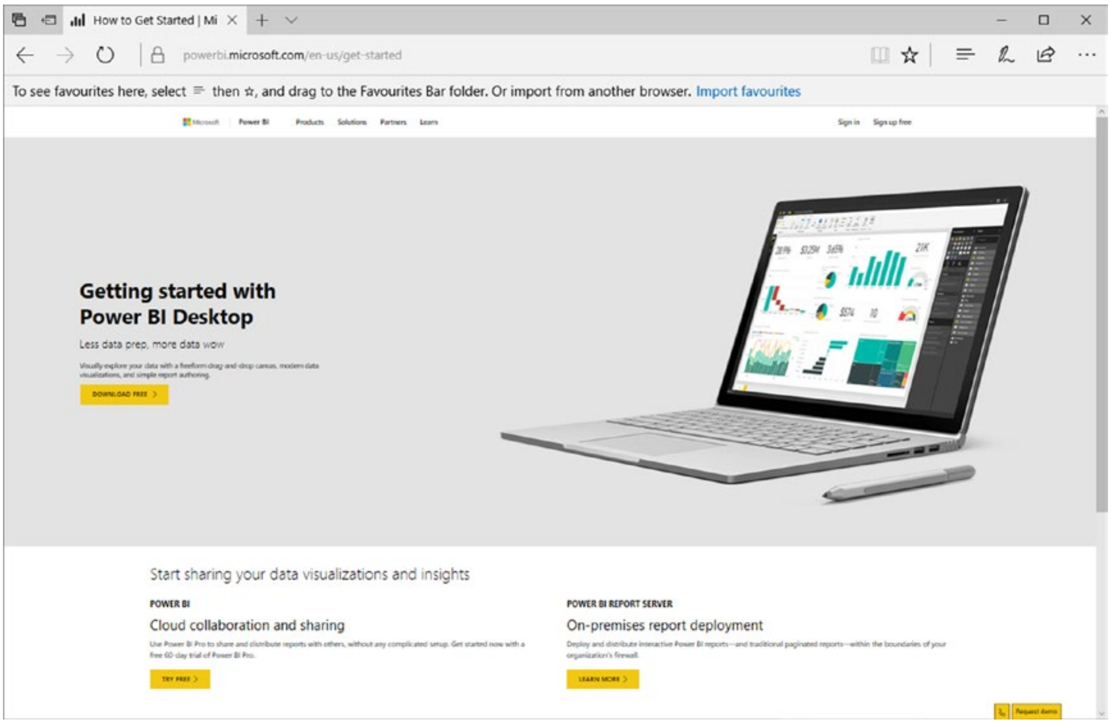
Power BI Premium is now available. Help your people drive insights. Learn more

What is Power BI?

Power BI is a suite of business analytics tools that deliver insights throughout your organization. Connect to hundreds of data sources, simplify data prep, and drive ad hoc analysis. Produce beautiful reports, then publish them for your organization to consume on the web and across mobile devices. Everyone can create personalized dashboards with a unique, 360-degree view of their business. And scale across the enterprise, with governance and security built-in.

***Figure 23-1.** The PowerBI.com connection page*

2. Click Start Free. You will see the How to Get Started page, as shown in Figure 23-2.



To see favourites here, select ≡ then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites

Getting started with
Power BI Desktop

Less data prep, more data wow

Visually explore your data with a freeform drag-and-drop canvas, modern data visualizations, and simple report authoring.

Start sharing your data visualizations and insights

POWER BI

Cloud collaboration and sharing

Use Power BI Pro to share and distribute reports with others, without any complicated setup. Get started now with a free 60-day trial of Power BI Pro.

POWER BI REPORT SERVER

On-premises report deployment

Deploy and distribute interactive Power BI reports—and traditional paginated reports—within the boundaries of your organization's firewall.

Figure 23-2. *The How to Get Started page*

3. Click Try Free. You will see the Sign Up page.

4. Enter your work e-mail address. The page will look like the one displayed in Figure 23-3.

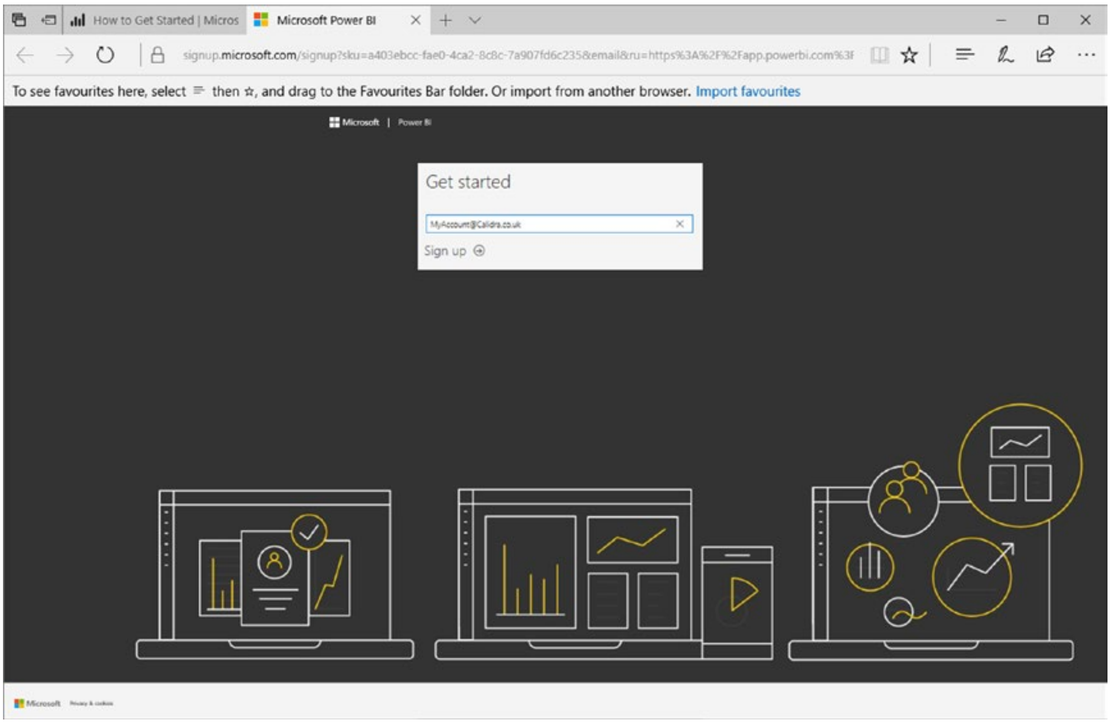
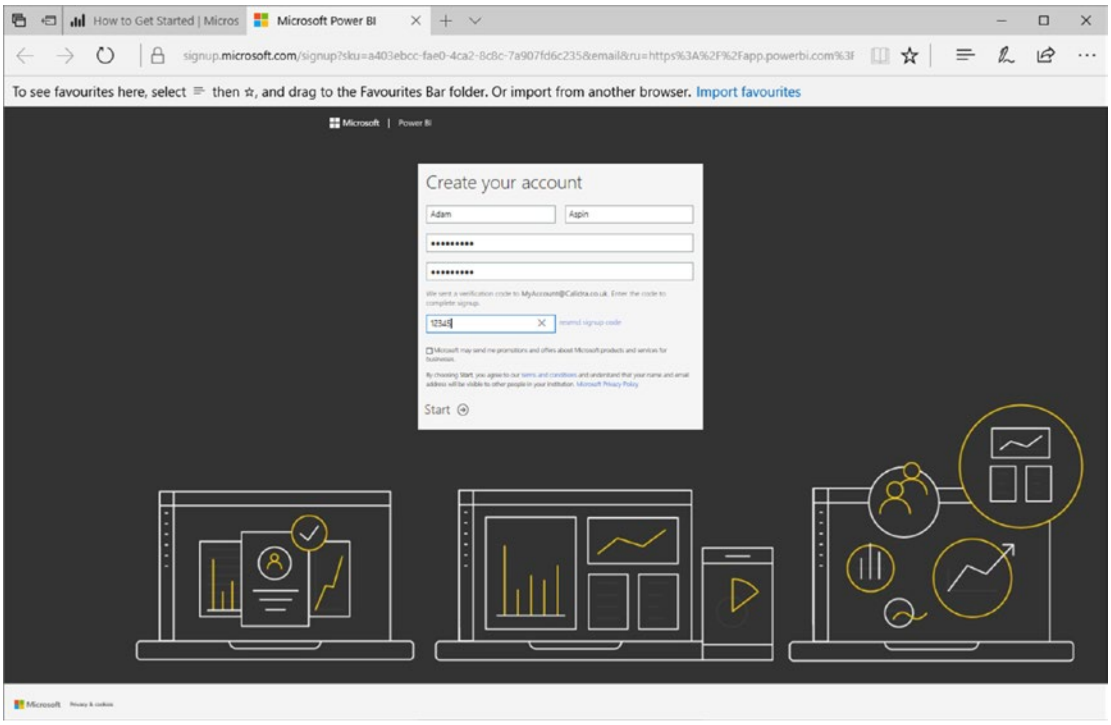

How to Get Started | Micros
Microsoft Power BI
signup.microsoft.com/signup?sku=a403ebcc-fae0-4ca2-8c8c-7a907fd6c235&email&ru=https%3A%2F%2Fapp.powerbi.com%3F
To see favourites here, select ≡ then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites
Microsoft | Power BI
Get started
MyAccount@Calidra.co.uk
Sign up

Figure 23-3. *The PowerBI.com Get Started dialog*

5. Click the arrow. PowerBI.com will send you an e-mail confirming the creation of your account.

6. Go to your e-mail and open the mail from Power BI. Type or paste the verification code into the Create Your Account dialog, which should look like what's shown in Figure 23-4.


How to Get Started | Micros
Microsoft Power BI
signup.microsoft.com/signup?sku=a403ebcc-fae0-4ca2-8c8c-7a907fd6c235&email&ru=https%3A%2F%2Fapp.powerbi.com%3F
To see favourites here, select ≡ then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites

Microsoft | Power BI

Create your account

Adam
Aspin
••••••••••
••••••••••

We sent a verification code to MyAccount@Calidra.co.uk. Enter the code to complete signup.

12345
resend signup code

☐ Microsoft may send me promotions and offers about Microsoft products and services for business.

By choosing Start, you agree to our terms and conditions and understand that your name and email address will be visible to other people in your institution. Microsoft Privacy Policy

Start

Microsoft Privacy & cookies

Figure 23-4. *Entering the e-mail verification code from PowerBI.com*

7. Click Start. You will be connected to the PowerBI.com service and see a web page like the one in Figure 23-5.

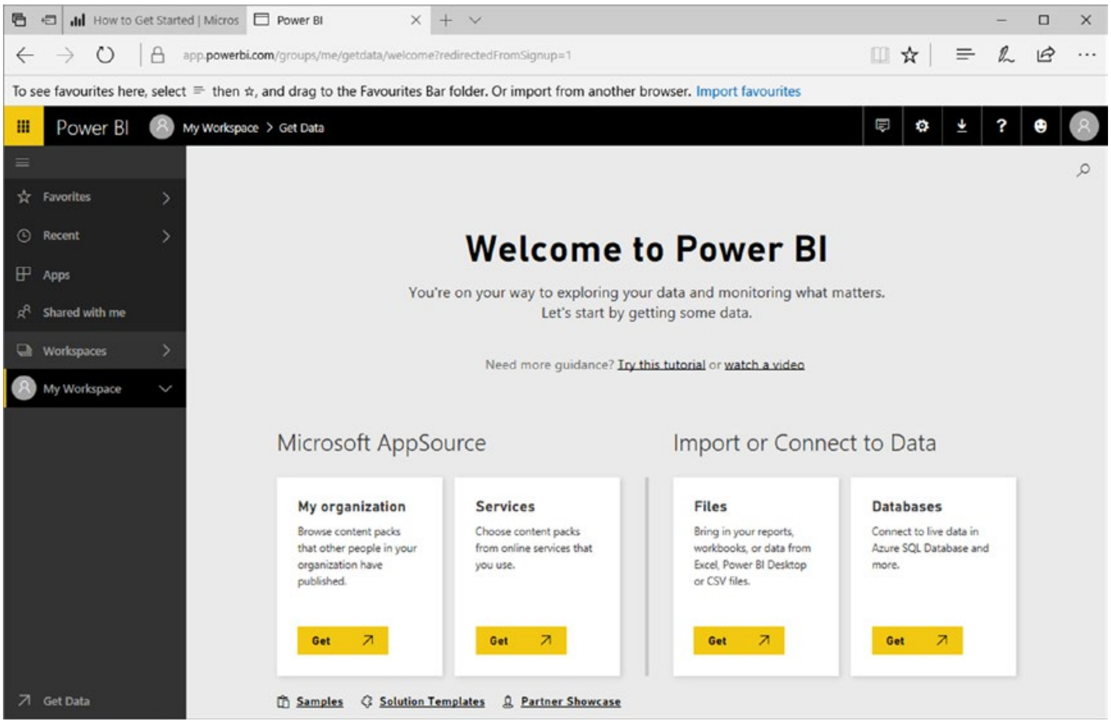

How to Get Started | Micros
Power BI
app.powerbi.com/groups/me/getdata/welcome?redirectedFromSignup=1
To see favourites here, select ≡ then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites
Power BI
My Workspace > Get Data
Favorites
Recent
Apps
Shared with me
Workspaces
My Workspace
Get Data
Welcome to Power BI
You're on your way to exploring your data and monitoring what matters.
Let's start by getting some data.
Need more guidance? Try this tutorial or watch a video
Microsoft AppSource
Import or Connect to Data
My organization
Browse content packs that other people in your organization have published.
Services
Choose content packs from online services that you use.
Files
Bring in your reports, workbooks, or data from Excel, Power BI Desktop or CSV files.
Databases
Connect to live data in Azure SQL Database and more.
Get
Samples
Solution Templates
Partner Showcase

Figure 23-5. *The PowerBI.com web page*

That is all that you have to do. You now have a PowerBI.com account and can start sharing your insights with colleagues and friends.

Using Power BI Desktop Files in PowerBI.com

It is all very well and good if you actually have a PowerBI.com account, but what is it and what can you use it for? PowerBI.com is essentially a collaboration environment optimized for self-service business intelligence. Given the whole self-service ethos that underlies Power BI, the easiest way to understand self-service BI is to see the steps in the life cycle of a Power BI Desktop file. This will show the process, from initial load, through sharing and interaction, to deletion. Hopefully, this will give you an idea of why Power BI is so different, so visual, and so tremendously useful in practice.

Logging On to PowerBI.com

1. Enter the PowerBI.com URL in your browser to connect to the PowerBI.com start page that you saw in Figure 23-1.
2. Click Sign In. The Microsoft sign-in page will appear, as shown in Figure 23-6.

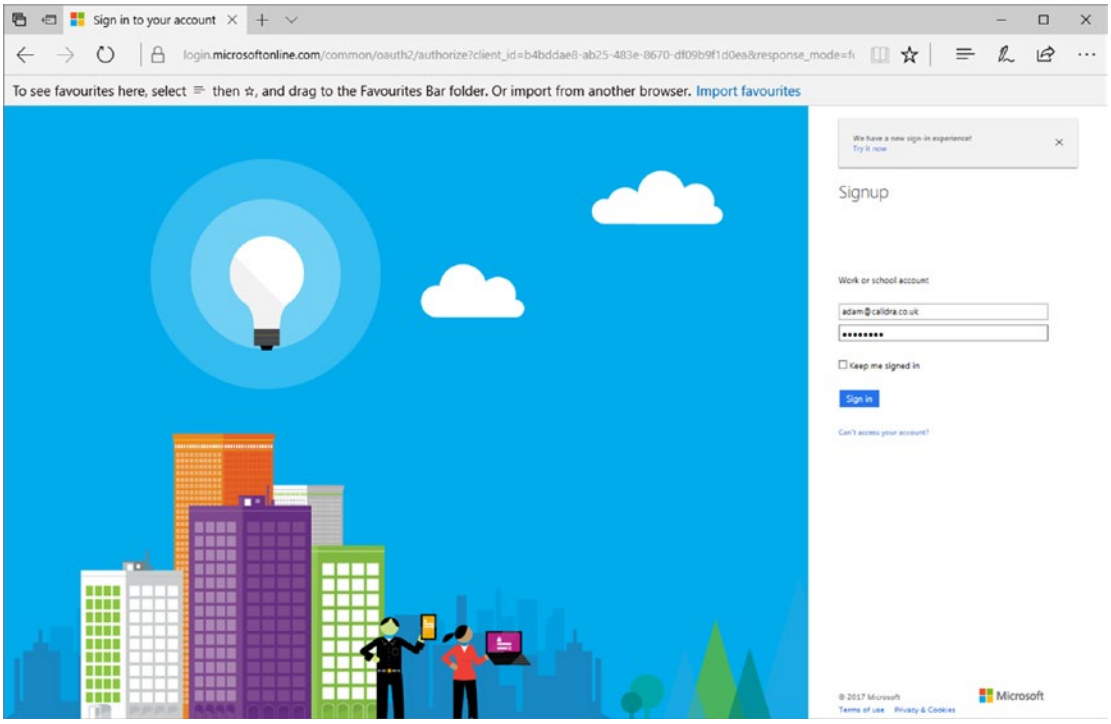

Sign in to your account
login.microsoftonline.com/common/oauth2/authorize?client_id=b4bddae8-ab25-483e-8670-df09b9f1d0ea&response_mode=f
To see favourites here, select ≡ then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites
We have a new sign-in experience!
Try it now
Signup
Work or school account
adam@calidra.co.uk
•••••••••
Keep me signed in
Sign in
Can't access your account?
© 2017 Microsoft
Terms of use
Privacy & Cookies
Microsoft

Figure 23-6. *The Microsoft sign-in page*

3. Click the account that you created previously. You see the PowerBI.com welcome page, as shown in Figure 23-7.

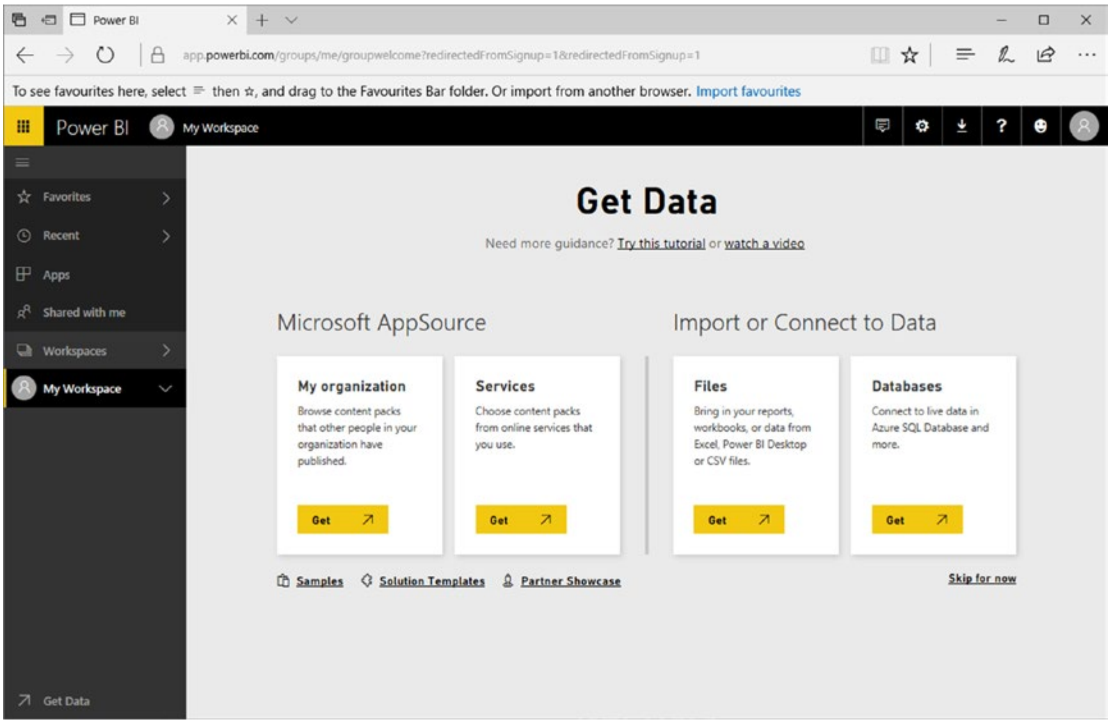

Power BI
app.powerbi.com/groups/me/groupwelcome?redirectedFromSignup=1&redirectedFromSignup=1
To see favourites here, select ≡ then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites
Power BI
My Workspace
Favorites
Recent
Apps
Shared with me
Workspaces
My Workspace
Get Data
Get Data
Need more guidance? Try this tutorial or watch a video
Microsoft AppSource
Import or Connect to Data
My organization
Browse content packs that other people in your organization have published.
Get
Services
Choose content packs from online services that you use.
Get
Files
Bring in your reports, workbooks, or data from Excel, Power BI Desktop or CSV files.
Get
Databases
Connect to live data in Azure SQL Database and more.
Get
Samples
Solution Templates
Partner Showcase
Skip for now

Figure 23-7. *The PowerBI.com My Workspace page*

You are now logged into PowerBI.com and ready to upload files created with Power BI Desktop.

Adding a Power BI Desktop File

As an (admittedly very simple) example, let's add a sample Power BI Desktop file directly to PowerBI.com:

1. In the Files section, click Get. You will switch to the Import or Connect to Data page that you can see in Figure 23-8.

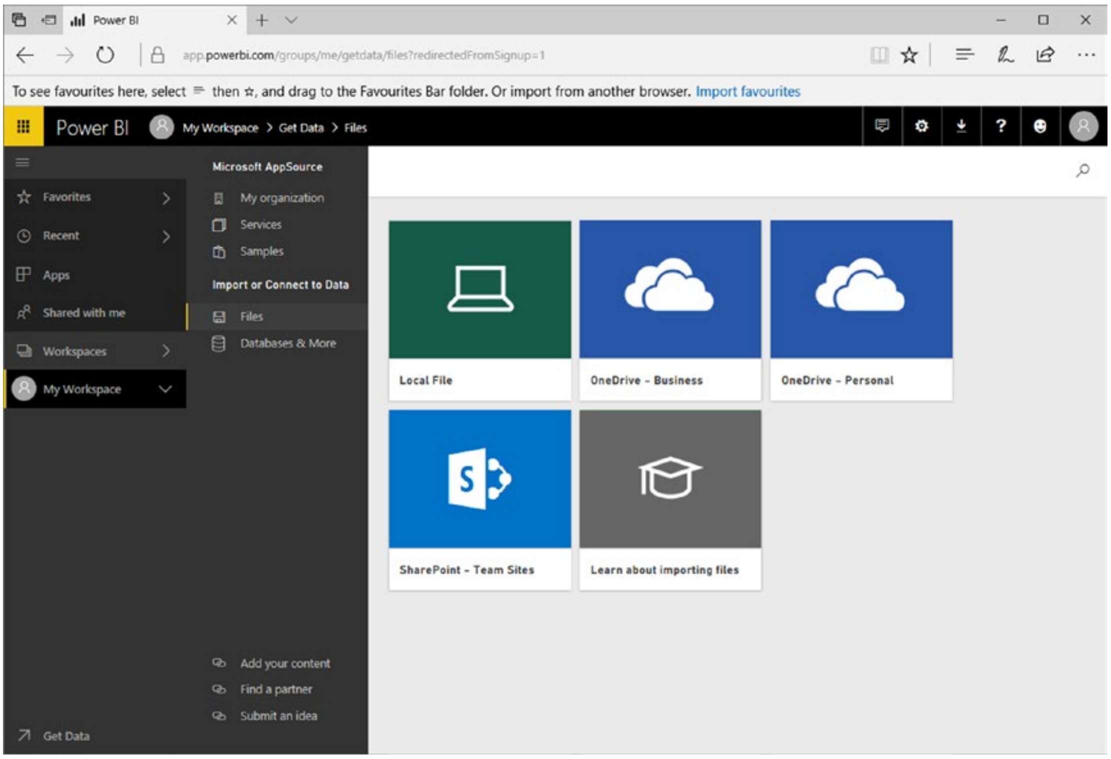
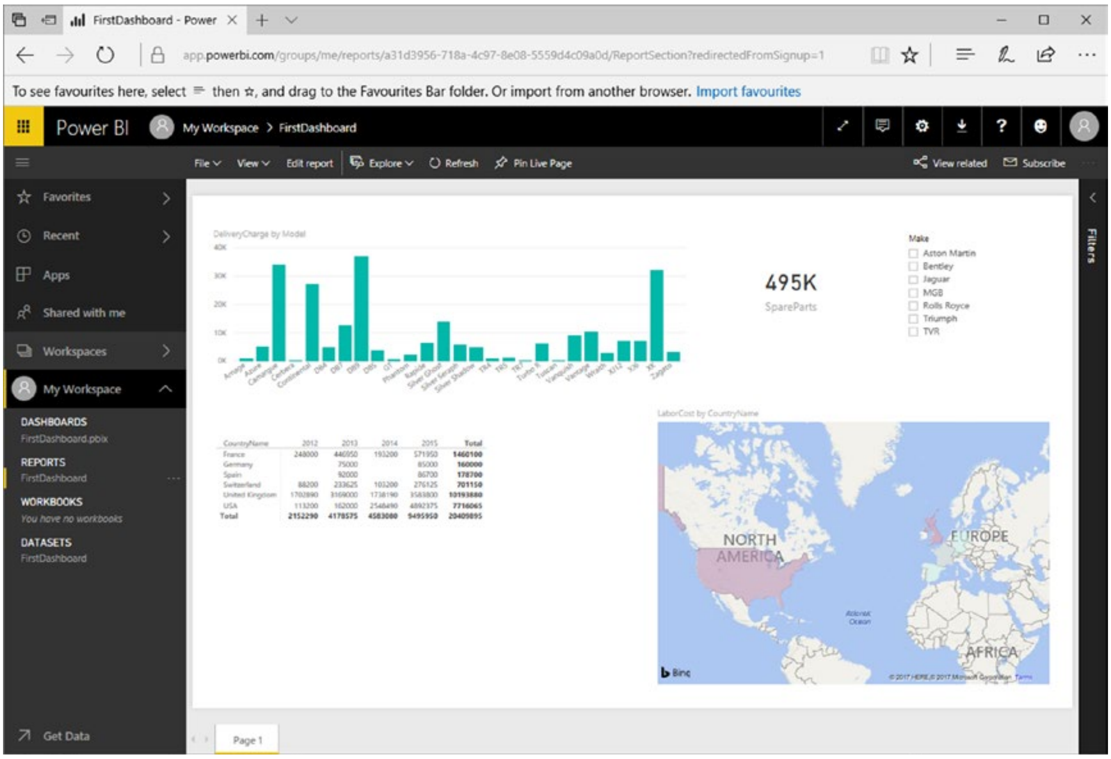

Power BI
app.powerbi.com/groups/me/getdata/files?redirectedFromSignup=1
To see favourites here, select then, and drag to the Favourites Bar folder. Or import from another browser. Import favourites
Power BI
My Workspace > Get Data > Files
Favorites
Recent
Apps
Shared with me
Workspaces
My Workspace
Get Data
Microsoft AppSource
My organization
Services
Samples
Import or Connect to Data
Files
Databases & More
Add your content
Find a partner
Submit an idea
Local File
OneDrive - Business
OneDrive - Personal
SharePoint - Team Sites
Learn about importing files

Figure 23-8. *The Power BI Import or Connect to Data page*

2. Click Local File. The Open dialog will be displayed.
3. Navigate to the Power BI Desktop file that you want to add to PowerBI.com. I used the C:\PowerBiDesktopSamples\CH23\FirstDashboard.pbix file from the sample data files that are available for download in this example.
4. Click OK. The file is added to PowerBI.com and appears in the Reports, and Datasets sections of the My Workspace section.
5. Expand the My Workspace section.

6. In the Reports section, click the FirstDashboard report. You will see the report contents exactly as they were created in Power BI Desktop. You can see this in Figure 23-9.



CountryName	2012	2013	2014	2015	Total
France	248000	446950	193200	571950	**1460100**
Germany		75000		85000	**160000**
Spain		92000		86700	**178700**
Switzerland	88200	233625	103200	276125	**701150**
United Kingdom	1702890	3169000	1738190	3583800	**10193880**
USA	113200	162000	2548490	4892375	**7716065**
Total	**2152290**	**4178575**	**4583080**	**9495950**	**20409895**

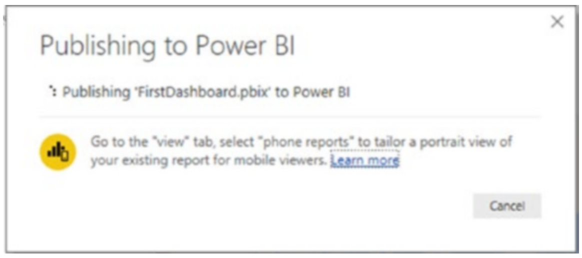
Figure 23-9. *PowerBI.com with a Power BI Desktop file added*

Publishing a Power BI Desktop File

Another way to add a Power BI Desktop file to PowerBI.com is from inside Power BI Desktop itself. This is unbelievably simple:

1. Open the Power BI Desktop file that you want to upload. I suggest using the sample file C:\PowerBiDesktopSamples\CH23\DatabaseCarSales.pbix.

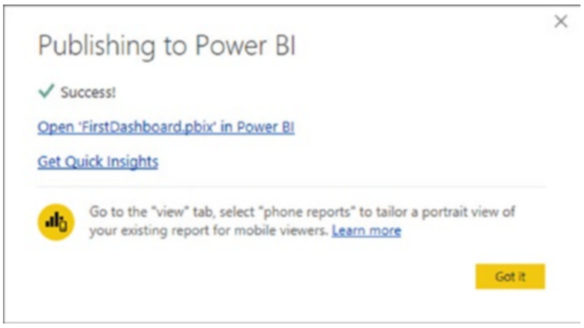
2. In the Home ribbon, click the Publish button. Power BI Desktop will display a dialog while the file is uploaded (and ask for confirmation if this file is replacing an existing version). You can see this in Figure 23-10. Once published, the success dialog will be displayed, as shown in Figure 23-11.


Publishing to Power BI

Publishing 'FirstDashboard.pbix' to Power BI

Go to the "view" tab, select "phone reports" to tailor a portrait view of your existing report for mobile viewers. Learn more

Figure 23-10. *The Publishing to Power BI dialog*


Publishing to Power BI

✓ Success!

Open 'FirstDashboard.pbix' in Power BI

Get Quick Insights

Go to the "view" tab, select "phone reports" to tailor a portrait view of your existing report for mobile viewers. Learn more

Figure 23-11. *The report successfully published dialog*

3. Click Got It.

You can now connect to your PowerBI.com account and see the report and dataset that you just loaded.

Dashboards, Reports, and Datasets

Adding a Power BI Desktop file to PowerBI.com has, in fact, added three separate elements:

- The report itself, which you can see in Figure 24-9.
- An element added to a new dashboard (you will learn more about this in a few pages' time).
- The dataset that drives the report.

All three elements are visible in the My Workspace section on the left of the PowerBI.com page.

Interacting with a Report on PowerBI.com

Now that you have uploaded a report to PowerBI.com, it is worth asking what you can do with reports now that you are no longer in Power BI Desktop. The answer is quite simple: you can do nearly everything that you can do in Power BI Desktop. Specifically, you can do the following:

- *Filter data* in the filter pane on the right of the PowerBI.com report, just as you would if using Power BI Desktop. Note that you cannot add further filters, but that any *existing* report, page, or visualization filters are accessible, and you can switch between basic filtering and advanced filtering just as you can in Power BI Desktop.
- *Highlight data* by clicking chart elements, for instance (again, just as you can in Power BI Desktop).
- *Switch between pages* using the tabs at the bottom of the report.
    - *Zoom in* to a specific visual by clicking the Focus icon in the top-right corner of a visual (yes, once again, just as you can in Power BI Desktop).
    - *Sort data* in a visual in the same way that you can in Power BI Desktop.
    - *Pan and zoom* in map visuals.
    - *Use any custom visuals* that you have downloaded from the PowerBI.com site.

I imagine that by now you understand that the reports that you have loaded from Power BI Desktop are virtually identical to the originals. The only major difference is that you cannot extend an existing report by adding new data or visuals. Indeed, if you have read the chapters explaining how to filter, sort, and highlight data, then you will find that PowerBI.com uses almost exactly the same techniques as Power BI Desktop, so you do not need any further explanation to help you on your way to interacting with your Power BI reports in the cloud.

Printing PowerBI.com Reports

Although PowerBI.com is built for sharing information interactively using the Web, it will let you print reports directly from a PowerBI.com site. To print a report, you simply do the following:

1. In the Report menu, click File. The File menu will appear.
2. Select Print. The Print Options dialog of your PC or tablet will appear, enabling you to select a printer and print the report.

Working with PowerBI.com Reports

PowerBI.com is not designed to be a replacement for Power BI Desktop. The capabilities that it offers as far as report modification are concerned are considerably more limited than those of the tool that you have learned to use in this book. Nonetheless, PowerBI.com does let you tweak existing reports to a certain extent. As most—if not all—of the report editing functions that are available also exist in the desktop version, I will only provide a cursory overview in this section.

The PowerBI.com Report Menu

The first thing that you need to know is that report modification is not carried out using ribbons, as is the case with Power BI Desktop. To make changes, you need to use the report menu, which is found above the report. You can see this in Figure 23-12.

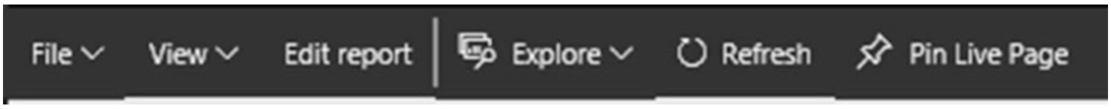


Figure 23-12. *The PowerBI.com report menu*

Report Editing

To modify a report (other than filtering the report or interacting with the visuals), you need to switch to Edit mode. To do this simply click the Edit Report menu button. A set of secondary menu items will appear. You can see these in Figure 23-13. These elements are explained in Table 23-1.

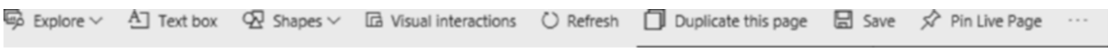


Figure 23-13. *Edit Report menu items*

Table 23-1. *Edit Report Menu Options*

Option	Description
Explore	Lets you drill up, down, and through the data in visuals. You can also see the data behind a visual.
Text Box	Adds a text box that you can then format exactly as described in the previous chapter.
Shapes	Adds shape objects to the report.
Visual Interactions	Allows you to specify how visuals interact when elements are selected.
Refresh	Refreshes the visuals from the source data.
Duplicate This Page	Creates a copy of the current page in the report.
Save	Saves your modifications to the cloud-based report.

In Edit mode you can add new visuals exactly as you can in Power BI Desktop. However, you are limited to using the data that is already in the report.

Another, final couple of options that you might find useful are in the File menu:

- Save the report as a Power BI Desktop file.
- Export the report as a PowerPoint file.

Exiting Edit Mode

Once you have finished your additions and modifications for a report, simply click the Reading view menu item. This will return the report to interactive mode.

Adding Datasets to PowerBI.com

Nearly this entire book has been devoted to explaining how you can create powerful, interactive, analytical reports using Power BI Desktop. Moreover, as I mentioned at the start of this chapter, PowerBI.com then comes into the frame to let you share your reports. However, a large part of the knowledge that you have acquired so far in this book can also be used directly in PowerBI.com to create reports directly in the cloud.

Reports that you create in PowerBI.com need data, so the first thing that you must do, before starting to create a report, is to obtain some data upon which you can base the report. Currently only two data sources (apart from Power BI Desktop files) can be used:

- Excel files (.xlsx and .xlsm)
- Text (.CSV) files

A couple of comments are necessary here:

- Any Excel file that you load *must* contain named tables, or the Excel file will not load. Remember that creating a table from a set of rows and column is as easy as clicking Insert ➤ Table in Excel and then, eventually, renaming the table.
- Any CSV file must be a correctly structured comma-separated file, as described in Chapter 2.

To see this in action:

1. Click Get Data at the bottom left of the PowerBI.com page.

2. Click Get ➤ Local File and select the file C:\PowerBiDesktopSamples\CH23\CarSales.xlsx. You will see the page shown in Figure 23-14.

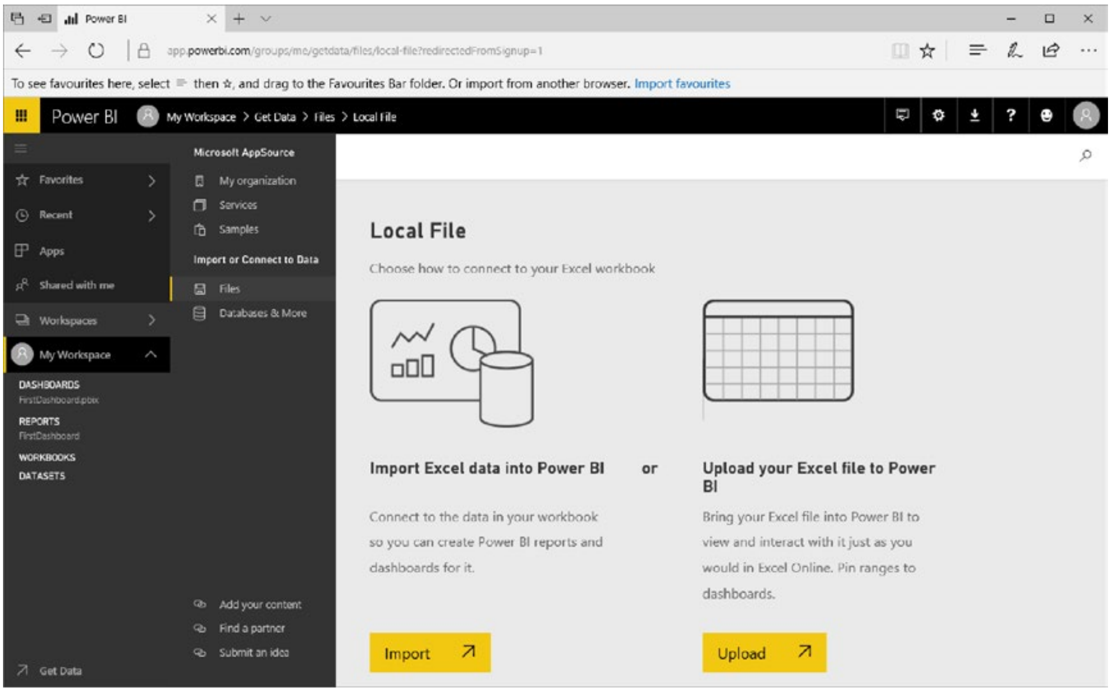

Power BI
app.powerbi.com/groups/me/getdata/files/local-file?redirectedFromSignup=1
To see favourites here, select then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites
Power BI
My Workspace > Get Data > Files > Local File
Favorites
Recent
Apps
Shared with me
Workspaces
My Workspace
DASHBOARDS
FirstDashboard.pbix
REPORTS
FirstDashboard
WORKBOOKS
DATASETS
Get Data
Microsoft AppSource
My organization
Services
Samples
Import or Connect to Data
Files
Databases & More
Add your content
Find a partner
Submit an idea
Local File
Choose how to connect to your Excel workbook
Import Excel data into Power BI
or
Upload your Excel file to Power BI
Connect to the data in your workbook so you can create Power BI reports and dashboards for it.
Bring your Excel file into Power BI to view and interact with it just as you would in Excel Online. Pin ranges to dashboards.
Import
Upload

Figure 23-14. *Loading data into PowerBI.com*

3. Click Import. After a few seconds the data will be loaded and you will see the page shown in Figure 23-15.

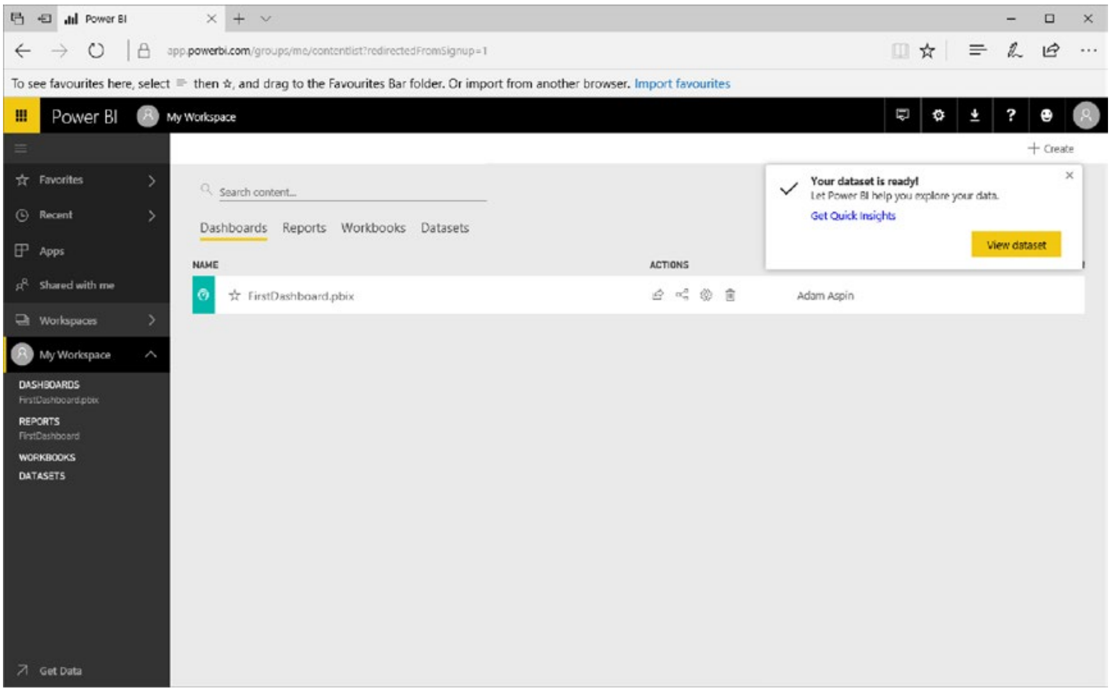


Figure 23-15.

4. Click View Dataset. PowerBI.com will display a blank report based on this dataset.

You can now build a PowerBI.com report using this data.

■ **Note** Clearly, you are extremely limited as to the data sources that you can use when creating reports from scratch in PowerBI.com. This leads inescapably to the conclusion that it is really Power BI Desktop that you should be using to create reports. Not only is the user experience immensely more gratifying, but the range of available data sources is vast.

Creating New Reports in PowerBI.com

Clearly, Power BI Desktop is the preferred solution when it comes to creating Power BI reports. Should you need to, however, you can also use existing cloud-based data or Excel, Power BI Desktop, or CSV files as data sources to create new reports. As a simple example, the following explains how to use the data in the Power BI Desktop file that you loaded earlier in this chapter to create a new report:

1. In the My Workspace tab, click the create report icon. A new, blank report canvas very similar to those that you have been using in Power BI Desktop will be displayed. You can see this in Figure 23-16.

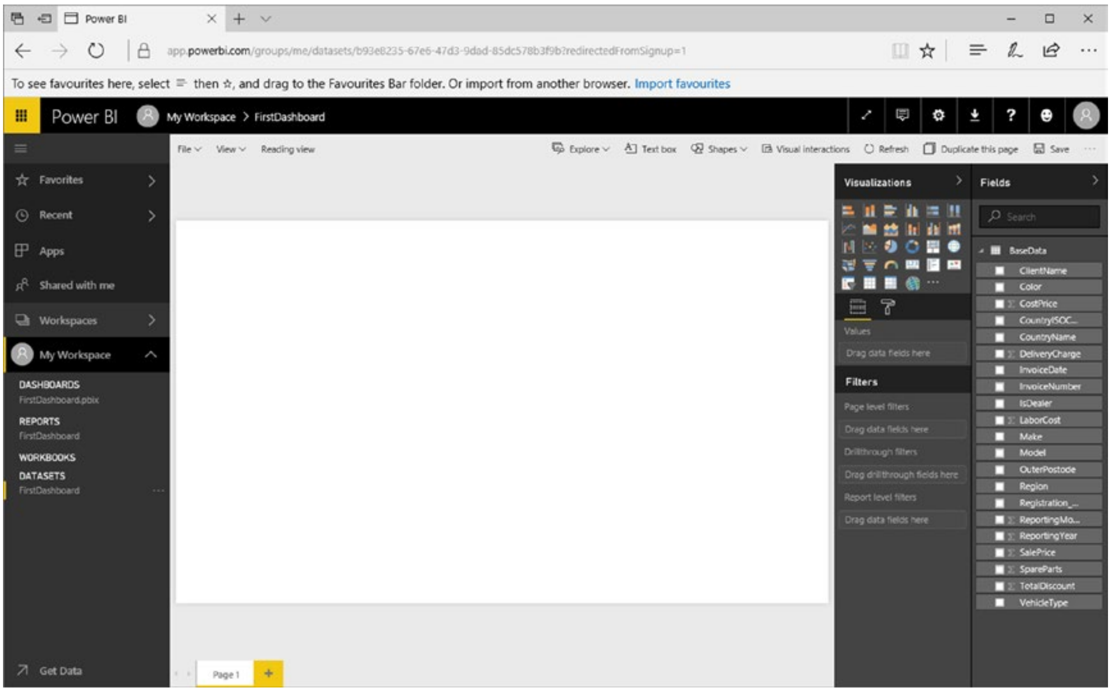


Figure 23-16. *The new report canvas*

2. Build your report just as you would in Power BI Desktop! This includes creating any of the visualizations that you explored in earlier chapters. You can format the visuals just as you would in Power BI Desktop, as well as add filters and shapes and define visual interactions much as you saw previously in this book.

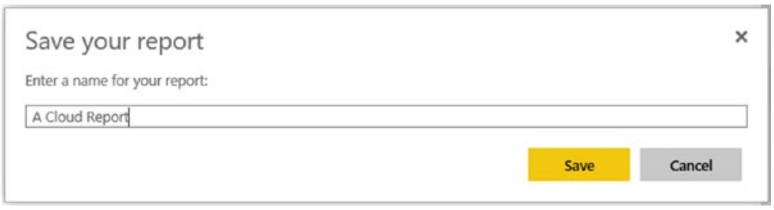
Well, I did warn you that it was almost identical to what you have learned so far. However, it is probably worth outlining the essential differences between creating reports directly in PowerBI.com and developing reports in Power BI Desktop:

- *Data*: This is currently limited to Excel, Power BI Desktop, or CSV files, or data that is accessible from a cloud-based source such as
  - Azure SQL database
  - Azure SQL Data Warehouse
  - SQL Server Analysis Services
- *Calculated columns and measures*: It is not currently possible to extend the dataset with calculated columns and measures. So you will have to ensure that all the metrics that you need are in the source data.
- *Custom visuals*: You can import any of the custom visuals that are available on the Power BI site, just as you can when using Power BI Desktop.

There are a few other minor differences, but the preceding are the key points that you need to remember if you are tempted to create reports in PowerBI.com.

One final thing that you need to know is how to save the report that you have made. In reality, you create a new report by saving the canvas in the dataset:

1. At the top right of the Datasets page, click Save.
2. Enter a name for the report. I chose PowerBIReport for this example. The save dialog will look like the one in Figure 23-17.


Save your report

Enter a name for your report:

A Cloud Report

Figure 23-17. *The Save Your Report dialog*

3. Click Save. The report will be displayed in the Workspace pane in the Reports tab.

PowerBI.com Dashboards

In PowerBI.com, the term *dashboard* has a slightly different meaning from the one that we have been using so far in this book. In PowerBI.com, a dashboard is a central point of focus where you can do this:

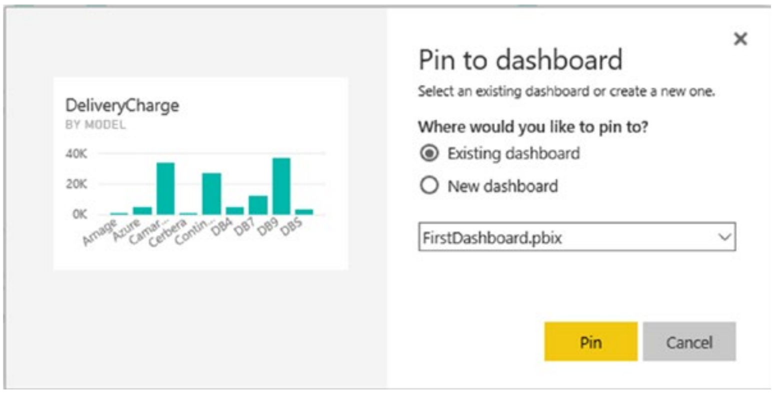
- Add visuals from any report that you have loaded into PowerBI.com. These visuals are called *tiles*.
- Jump straight from a dashboard to the report hosting a visual in a dashboard.
- Enhance dashboards with specific annotations.

Let's look at how this works by taking the dashboard that was created when you loaded the FirstDashboard report and adding a visual to it from this same report.

Adding Tiles to PowerBI.com Dashboards

Now that you have a dashboard ready and waiting, the following explains how to add some content from an existing report:

1. In the Navigation pane, click the report that you have already loaded. This will be FirstDashboard, assuming that you followed the earlier example. The report will appear, as shown in Figure 23-9.
2. Hover the mouse pointer over the chart (with the title DeliveryCharge by Model) on the top left of the report. Three small icons will appear at the top right of the chart.
3. Click the Pin icon. This will display the Pin to Dashboard dialog that you can see in Figure 23-18.



Pin to dashboard

Select an existing dashboard or create a new one.

Where would you like to pin to?

● Existing dashboard

○ New dashboard

FirstDashboard.pbix

Pin Cancel

Figure 23-18. *The Pin to Dashboard dialog*

4. Ensure that the Existing Dashboard radio button is selected and that the Car Sales dashboard is selected in the popup list of available dashboards.
5. Click Pin.

The chart is added to the Car Sales dashboard. If you click the name of this dashboard in the Navigation pane, you will see that the dashboard looks like Figure 23-19.

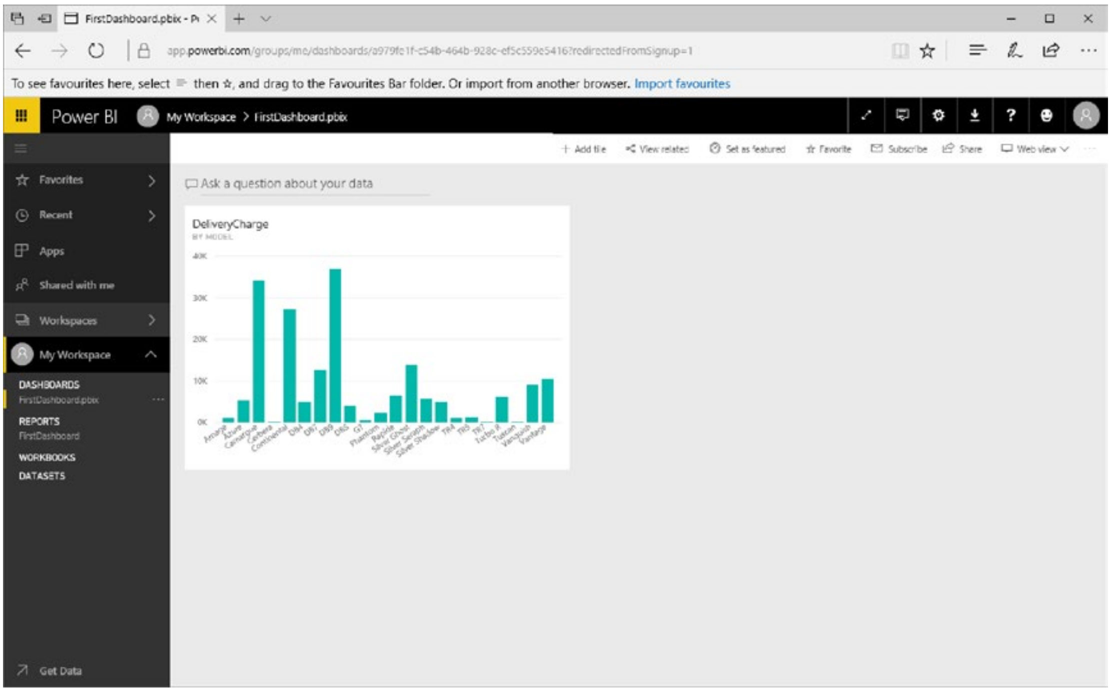

FirstDashboard.pbix - Pi
app.powerbi.com/groups/me/dashboards/a979fe1f-c54b-464b-928c-ef5c559e5416?redirectedFromSignup=1
To see favourites here, select then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites
Power BI
My Workspace > FirstDashboard.pbix
Add tile
View related
Set as featured
Favorite
Subscribe
Share
Web view
Favorites
Recent
Apps
Shared with me
Workspaces
My Workspace
DASHBOARDS
FirstDashboard.pbix
REPORTS
FirstDashboard
WORKBOOKS
DATASETS
Get Data
Ask a question about your data
DeliveryCharge
BY MODEL
40K
30K
20K
10K
0K

Figure 23-19. *A dashboard with a visual added*

If you click the chart in the dashboard, you immediately jump to the source report and page for this visual.

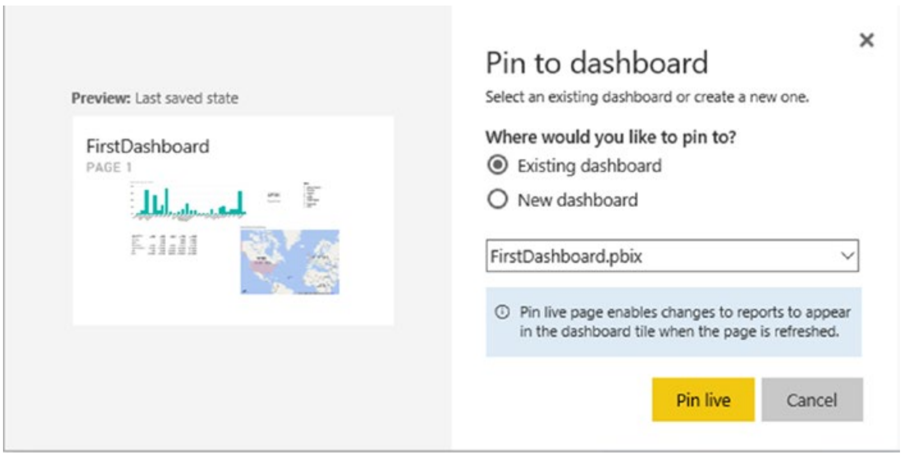
■ **Note** You can also create a new dashboard when you pin a visual by selecting New Dashboard as the destination in the Pin to Dashboard dialog.

Pinning Entire Reports

If you need a dashboard tile that is, in reality, an entire report, then you can choose to pin an entire report:

1. Expand My Workspace, expand the Reports section, and click the report that you wish to pin to a dashboard.

2. Click Pin Live Page on the Report menu. The Pin to Dashboard dialog will appear, as shown in Figure 23-20.



Pin to dashboard

Select an existing dashboard or create a new one.

Where would you like to pin to?

[✓ Existing dashboard]

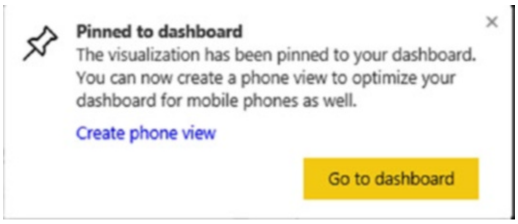
New dashboard

FirstDashboard.pbix

Pin live page enables changes to reports to appear in the dashboard tile when the page is refreshed.

Figure 23-20. *The Pin to Dashboard dialog*

3. Click Pin Live. The Pinned to Dashboard alert appears at the top right to inform you that the report has been pinned. You can see this in Figure 23-21.



Pinned to dashboard

The visualization has been pinned to your dashboard. You can now create a phone view to optimize your dashboard for mobile phones as well.

Figure 23-21. *The Pinned to Dashboard alert*

4. Click Go to Dashboard to view the dashboard with the report. You can see the result of this approach in Figure 23-22.

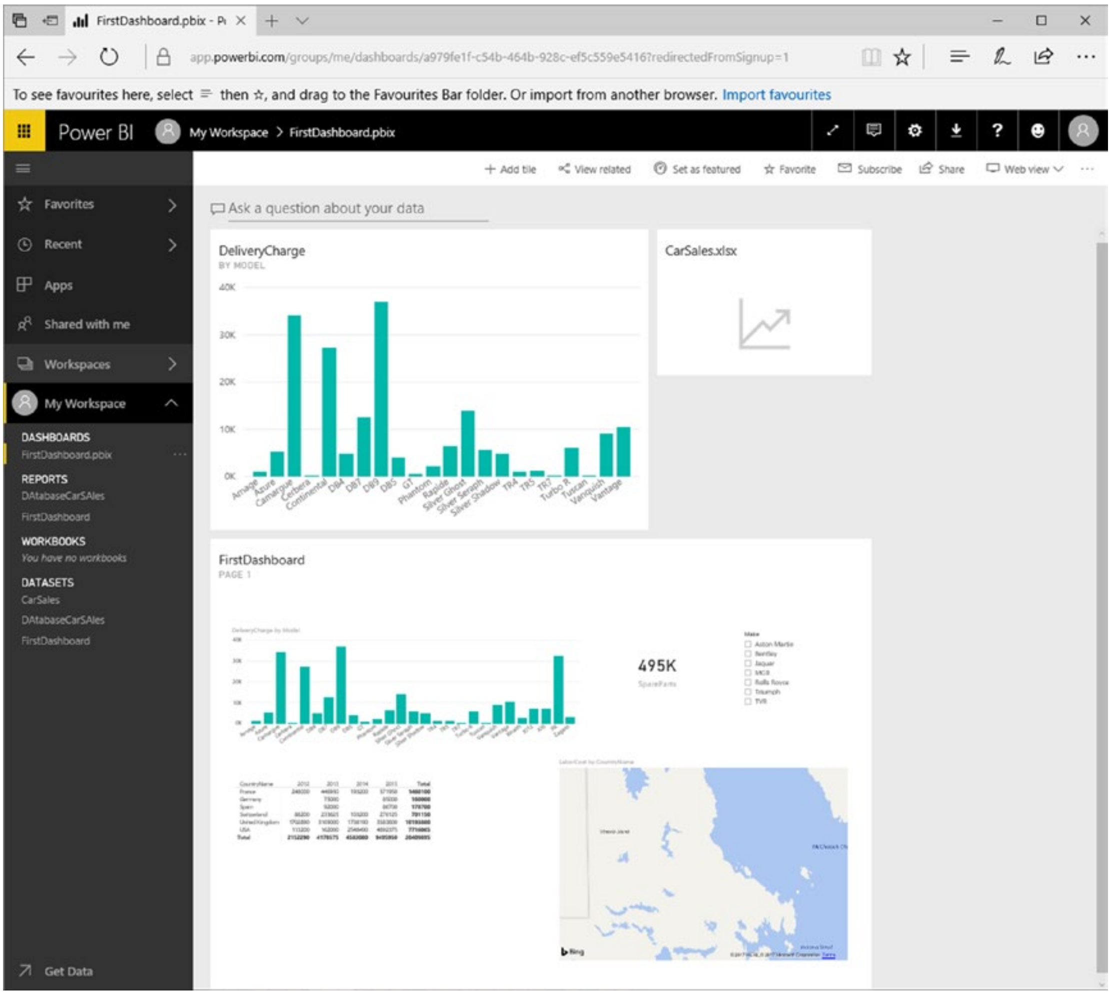


Figure 23-22. A dashboard with a report pinned to it

■ **Note** Pinning an entire report means that any changes made to the report will update the dashboard in real time.

Editing Dashboard Tiles

There are a few interesting things that you can do to tiles once you have added them to dashboards. The available options include

- Deleting tiles
- Modifying tile details
- Exporting the data behind the tile
- Pinning the tile to another dashboard

Let's look at each of these in turn.

Deleting Tiles

The following explains how to delete a tile from a dashboard (while leaving the source visual intact on the original report):

1. Click the tile menu icon (the ellipses at the top right of the tile). The tile options will appear, overlaying the tile data, as you can see in Figure 23-23.

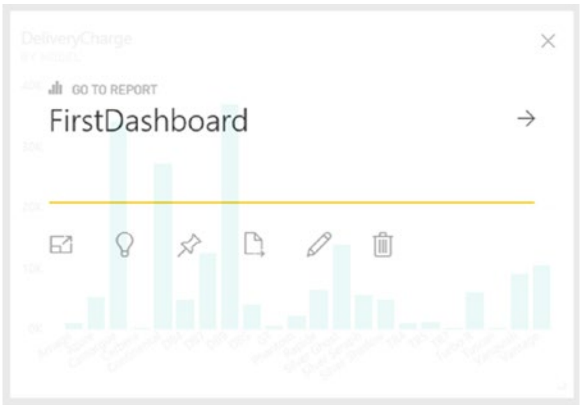


Figure 23-23. *The tile menu*

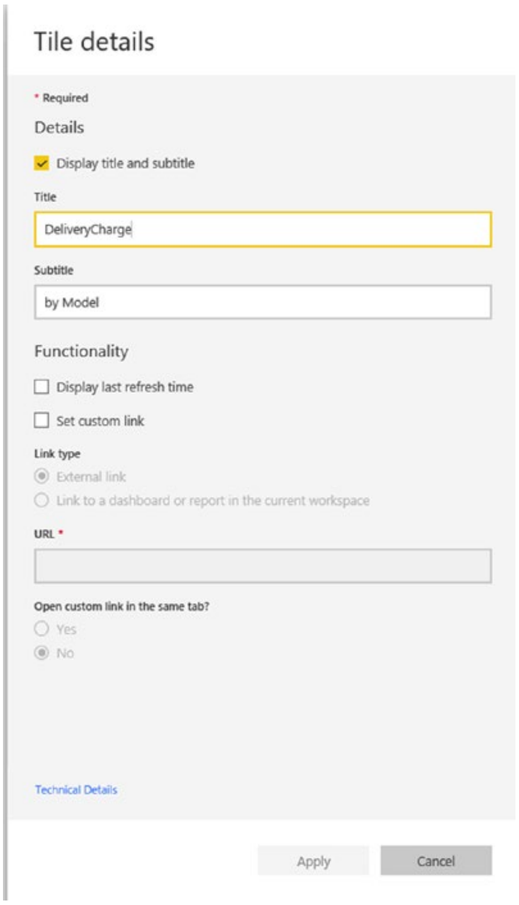
2. Click the waste bin icon on the right. The tile will be removed from the dashboard.

■ **Note** If you delete a tile by mistake, you cannot undo the action. However, since the original visual is still in the source report, you can add it back easily enough.

Modifying Tile Details

There are a few details that you can modify when adding tiles to dashboards:

1. Click the tile menu icon (the ellipses at the top right) to display the tile menu options.
2. Click the pencil icon. The Tile Details pane will appear on the right of the screen, looking rather like the one in Figure 23-24.


Tile details

* Required

Details

[✓] Display title and subtitle

Title: DeliveryCharge

Subtitle: by Model

Functionality

[ ] Display last refresh time

[ ] Set custom link

Link type: [● External link] ○ Link to a dashboard or report in the current workspace

URL *

Open custom link in the same tab? ○ Yes [● No]

Figure 23-24. *The Tile Details pane*

The available options are largely self-explanatory, so I will not take you through all the possibilities, but simply explain what can be done to tweak a tile in a dashboard.

- *Title* and *Subtitle*: If you edit the elements in these fields, you can change the title and subtitle that were inherited from the source visualization.
- *Display last refresh time*: Checking this option adds the latest refresh date and time to the tile title.

- *Set custom link*: Adding an URL to this field defines a hyperlink to a completely different web page (rather than the original report containing the source visual) when you click the tile in PowerBI.com.

Exporting the Data Behind the Tile

You can export the data that a visual is based on at any time. It is exported as a CSV file to your Downloads folder. To do this, you need to do the following:

1. Click the tile menu icon to display the tile menu options.
2. Click the page icon. The data export toolbar will appear at the bottom of the web page. You can see this in Figure 23-25.

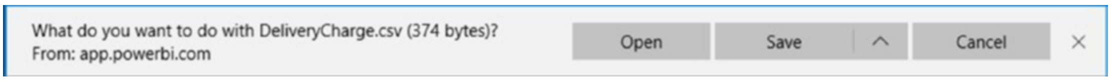

What do you want to do with DeliveryCharge.csv (374 bytes)?
From: app.powerbi.com

Figure 23-25. *The data export toolbar*

3. Export the data as you would when downloading data from just about any web page.

■ **Note** PowerBI.com does not export the entire dataset that a report is based on. It only exports the subset of data that is used in a visualization.

Pinning the Tile to Another Dashboard

Just as you pinned a visual to a dashboard, you can extend the process by pinning a tile to another dashboard.

1. Click the tile menu icon to display the tile menu options.
2. Click the Pin icon. This will display the Pin to Dashboard dialog that you saw previously in Figure 23-18.
3. Select a new or existing dashboard and click Pin. The tile will be added to the selected dashboard.

Modifying Dashboards

Apart from adding tiles, there other things that you can do with dashboards:

- Resize tiles
- Add other tiles (which can be images, text boxes, or other web content)
- Display dashboards in full screen mode
- Print dashboards

Let’s take a quick look at each of these in turn.

Resizing Tiles

As you might expect, tiles can be resized (and moved) on dashboards. All you have to do to move a tile is to click inside the tile and drag it elsewhere on the dashboard pane. To resize a tile, you click the bottom-right corner of the tile and drag the mouse up and left (to shrink the tile) or down and right (to enlarge a tile).

Adding Other Tiles (Images, Text Boxes, Video, or Other Web Content)

Dashboards are not limited to displaying only visuals from existing reports. To enhance your message, you can add other web content or existing images. You can even add text boxes to make a specific point. Adding tiles is this simple:

1. At the top right of the PowerBI.com screen, click Add Tile. The Add Tile dialog will appear, as you can see in Figure 23-26.

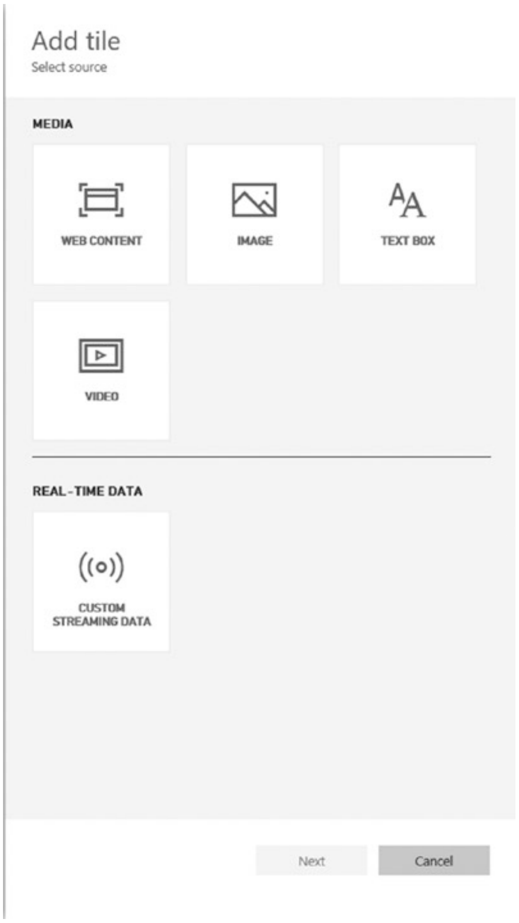


Figure 23-26. *Dashboard tiles*

2. Select one of the available tile options. The appropriate Tile Details pane will appear at the right of the web page, where you can enter any necessary information.

The currently available options for tiles are outlined in Table 23-2.

Table 23-2. *PowerBI.com Tile Types*

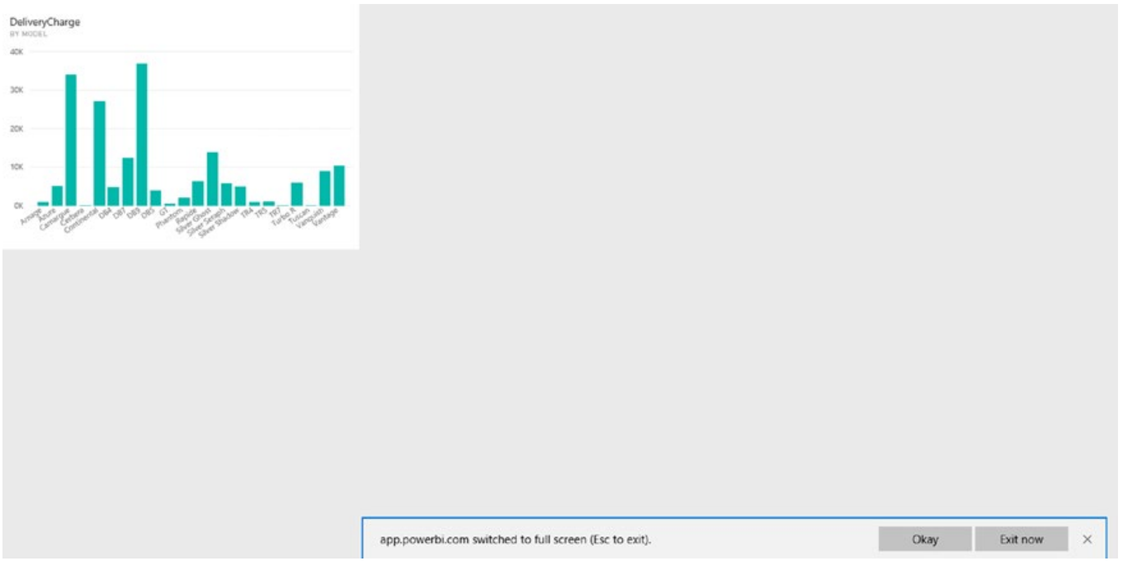
Tile Type	Description
Image	Allows you to set a URL for the image that you want to display. You cannot add images from a local disk.
Video	Allows you to set a URL for the YouTube video that you want to display. Only YouTube videos can be added to PowerBI.com dashboards as of the time of writing.
Web Content	Displays the Tile Details pane, where you can add any code that you want to embed in the dashboard page.
Text Box	Displays the Text Box pane, where you can enter and format the text that you want to display in a dashboard.
Real-Time Data	Lets you add streaming datasets to your PowerBI.com dashboard.

■ **Note** If you have begun to add a tile and want to abandon it before you have finished, then simply click the close icon at the top right of the Tile Details pane.

Displaying Dashboards in Full Screen Mode

If you are using PowerBI.com as a presentation tool, or if you simply want to see the contents of a dashboard without any of the other panes or menus, you can always display the dashboard in full screen mode. It's a single step:

Click the Enter Full Screen Mode icon (the diagonal two-headed arrow) in the PowerBI.com title bar. The dashboard will appear in full screen mode, as shown in Figure 23-27.



app.powerbi.com switched to full screen (Esc to exit).

Figure 23-27. *Displaying a dashboard in full screen mode*

To exit full screen mode, simply press Escape or click the Exit Now button in the toolbar at the bottom of the web page.

Printing Dashboards

Although PowerBI.com is built for sharing and interactivity on the Web, it also lets you make paper copies of your dashboards. The following explains how to print a dashboard:

1. Click the More Options button (the ellipses at the top right of the web page). The options menu will appear as shown in Figure 23-28.

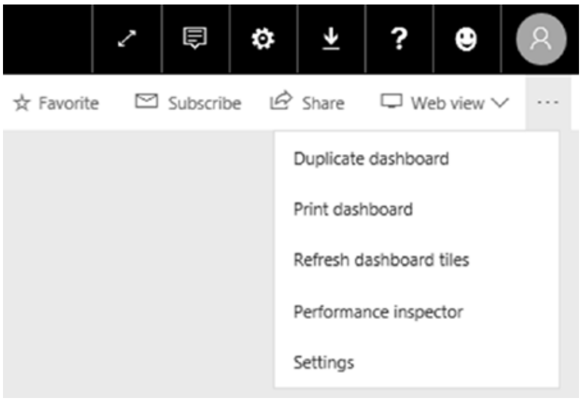


Figure 23-28. *The More Options menu*

2. Click Print Dashboard. The Print Options dialog of your PC or tablet will appear, from where you can select a printer and print the dashboard.

PowerBI Pro

Power BI Pro is the Enterprise version of the free PowerBI.com offering. It has a cost (currently $9.99 per user per month) and has considerably greater capacities when it comes to sharing content and integrating with Power BI Premium—Microsoft's Enterprise BI offering.

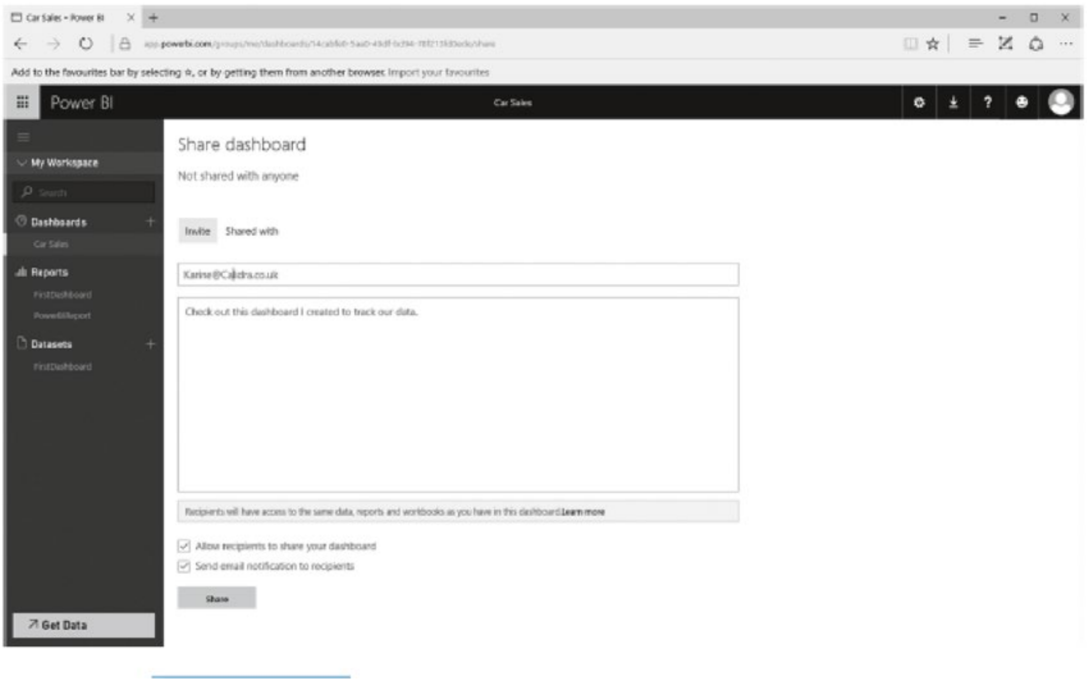
As the commercial landscape for Power BI is subject to frequent change, I suggest that you look at the Microsoft web site for the latest details.

Sharing Dashboards

Creating insightful dashboards may be fun, but it is a truly meaningful activity only if the information can be shared with colleagues. Perhaps inevitably, PowerBI.com not only allows your co-workers to share your insights, but also actively helps you to disseminate the information. Here's an example:

1. Click the Share icon at the top right of the dashboard. The Share Dashboard page will be displayed.
2. Enter in the upper field the e-mail addresses of the people who you want to share your dashboard with.

3. Add a message for the recipients in the lower field. The web page will look something like Figure 23-29.


Share dashboard

Not shared with anyone

Karine@Calidra.co.uk

Check out this dashboard I created to track our data.

Recipients will have access to the same data, reports and workbooks as you have in this dashboard. Learn more

[✓] Allow recipients to share your dashboard

[✓] Send email notification to recipients

Figure 23-29. *Sharing dashboards*

4. Click Share.

Your colleagues will receive an e-mail containing the link to the dashboard that you have shared. Clicking this link will display the dashboard.

If you do not want to send an e-mail with a link, you can click the Shared With tab in the Share Dashboard page, and simply copy and paste the link into a file or a Microsoft Lync message (for instance) and enable your co-workers to access your dashboard in this way.

In any case, the recipient with whom you are sharing the dashboard needs a PowerBI.com account.

PowerBI.com Gateways

The ability to share your analyses on a platform as powerful as PowerBI.com needs only one more thing to make it an essential business tool. All that is missing is the ability to ensure that the data that you are sharing is up to date. Fortunately, the Power BI team has thought of this, too. Their solution is to let you create gateways from your on-premises data to the PowerBI.com site. These gateways allow you to refresh the data in your Power BI Desktop files from the data sources on which they are built. Not only that, but you can refresh the data manually or automatically on a schedule that you define.

Currently, there are two kinds of Power BI gateways:

- Personal
- Enterprise

The essential differences are that the Enterprise gateway is designed for multiple users, and consequently has fine-grained access control, monitoring, and auditing. As an added bonus, it allows for DirectQuery connections (which you saw in Chapter 2) to SQL Server.

PowerBI.com gateways are a large subject. Consequently, I will only give a brief introduction to the personal gateway here. However, once you understand how PowerBI.com gateways work, you should have no difficulties in extending their use to suit your specific requirements.

Downloading a Gateway

A PowerBI.com gateway has to be installed on a local computer first. This is as easy as downloading the gateway application and then carrying out basic configuration:

1. In PowerBI.com, click the downloads icon at the top right of the page, as seen in Figure 23-30.

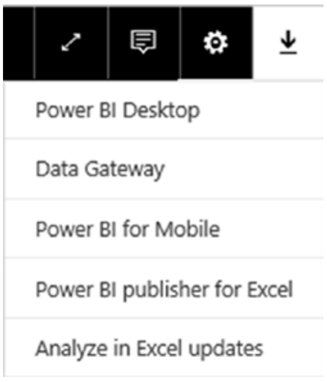
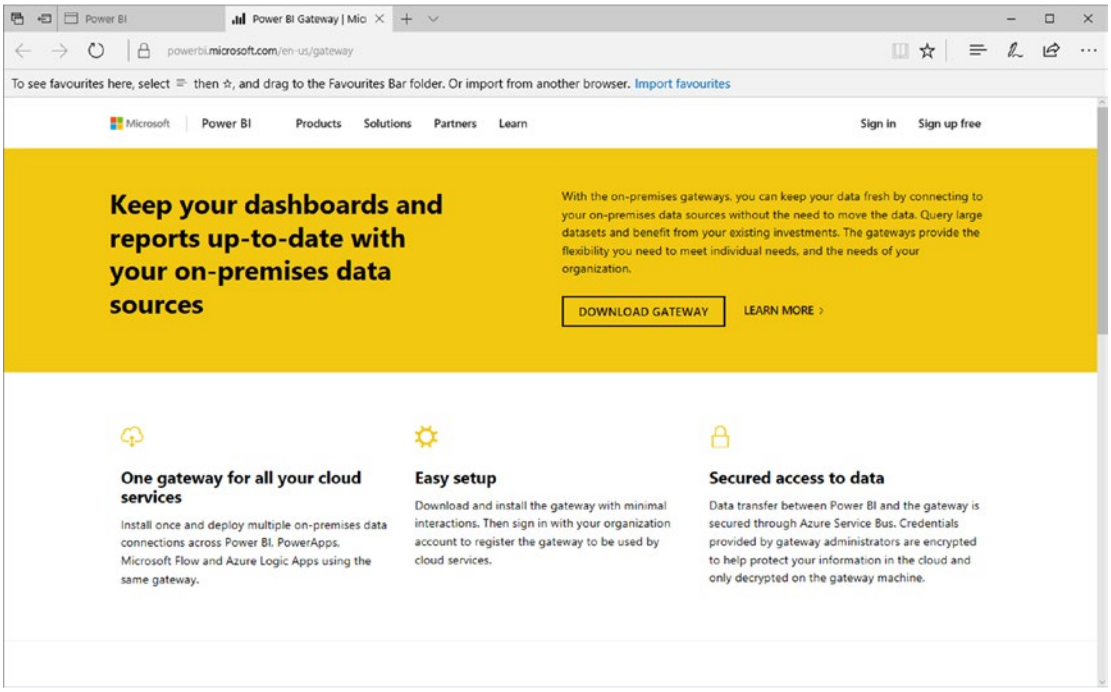


Figure 23-30. *The downloads icon and menu*

2. Click Data Gateway. A new browser tab will open, displaying the Power BI Gateway splash screen. This is shown in Figure 23-31.


Power BI
Power BI Gateway | Mic
powerbi.microsoft.com/en-us/gateway
To see favourites here, select ≡ then ☆, and drag to the Favourites Bar folder. Or import from another browser. Import favourites
Microsoft
Power BI
Products
Solutions
Partners
Learn
Sign in
Sign up free
DOWNLOAD GATEWAY
LEARN MORE ›

Keep your dashboards and reports up-to-date with your on-premises data sources

With the on-premises gateways, you can keep your data fresh by connecting to your on-premises data sources without the need to move the data. Query large datasets and benefit from your existing investments. The gateways provide the flexibility you need to meet individual needs, and the needs of your organization.

One gateway for all your cloud services

Install once and deploy multiple on-premises data connections across Power BI, PowerApps, Microsoft Flow and Azure Logic Apps using the same gateway.

Easy setup

Download and install the gateway with minimal interactions. Then sign in with your organization account to register the gateway to be used by cloud services.

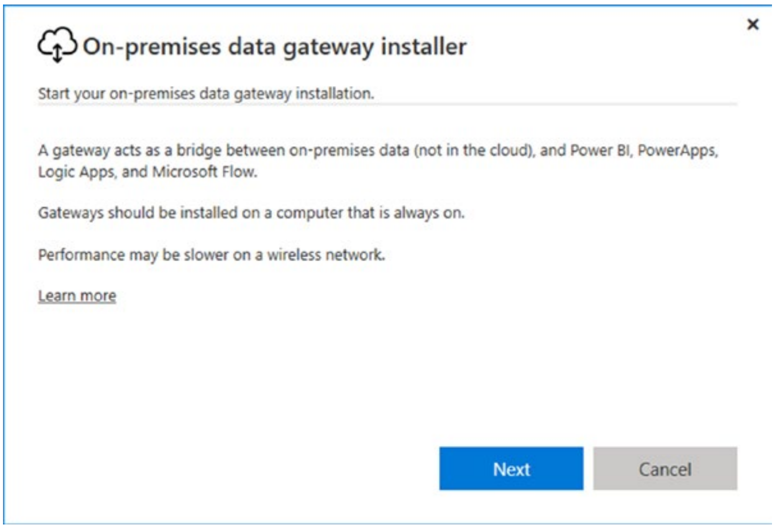
Secured access to data

Data transfer between Power BI and the gateway is secured through Azure Service Bus. Credentials provided by gateway administrators are encrypted to help protect your information in the cloud and only decrypted on the gateway machine.

Figure 23-31.

3. Click Download Gateway. After a short time the web browser will display the PowerBIGatewayInstaller.exe button at the bottom of the screen.

4. Click the PowerBIGatewayInstaller.exe button. The On-premises Data Gateway Installer dialog will appear. You can see this in Figure 23-32.



On-premises data gateway installer

Start your on-premises data gateway installation.

A gateway acts as a bridge between on-premises data (not in the cloud), and Power BI, PowerApps, Logic Apps, and Microsoft Flow.

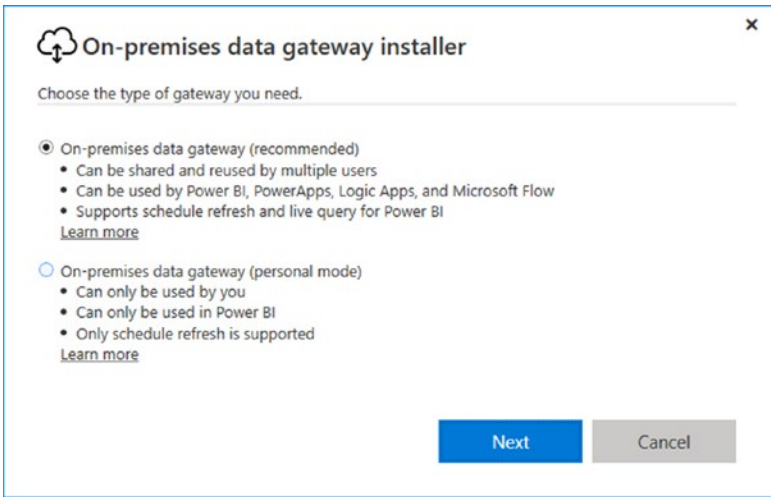
Gateways should be installed on a computer that is always on.

Performance may be slower on a wireless network.

Learn more

Figure 23-32. *The On-premises Data Gateway Installer dialog*

5. Click Next, then choose the gateway that you wish to install (the recommended on-premises data gateway in this example). The dialog will look like the one shown in Figure 23-33.



On-premises data gateway installer

Choose the type of gateway you need.

● On-premises data gateway (recommended)
- Can be shared and reused by multiple users
- Can be used by Power BI, PowerApps, Logic Apps, and Microsoft Flow
- Supports schedule refresh and live query for Power BI

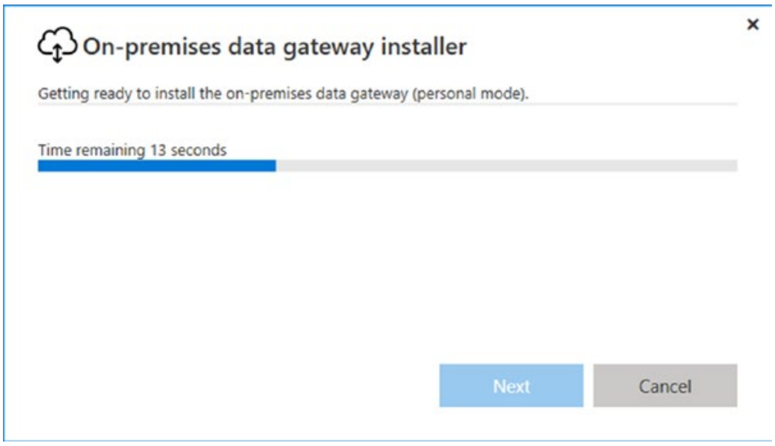
Learn more

○ On-premises data gateway (personal mode)
- Can only be used by you
- Can only be used in Power BI
- Only schedule refresh is supported

Learn more

Figure 23-33. *The On-Premises Gateway installer dialog*

6. Click Next. The gateway installation will start, and the gateway installer will display the installation progress, as you can see in Figure 23-34.

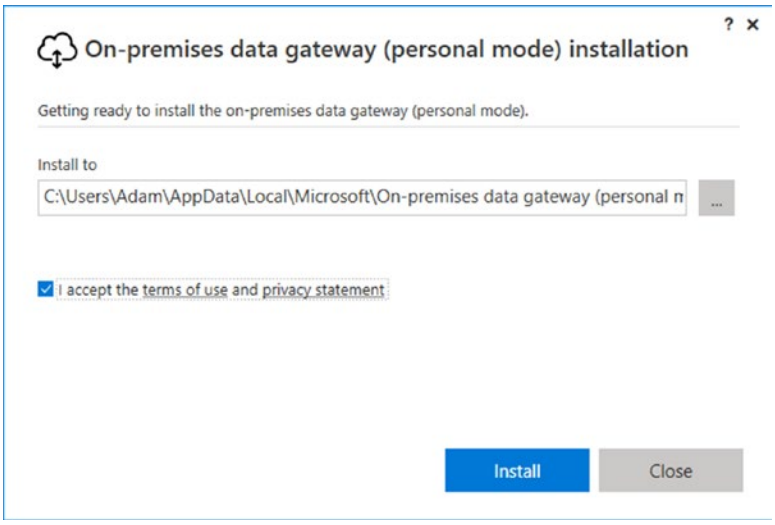

On-premises data gateway installer

Getting ready to install the on-premises data gateway (personal mode).

Time remaining 13 seconds

Figure 23-34. *Installation progress of the gateway*

7. Accept the terms of use and click Install, and choose the installation folder (or accept the default). You can see this in Figure 23-35.


On-premises data gateway (personal mode) installation

Getting ready to install the on-premises data gateway (personal mode).

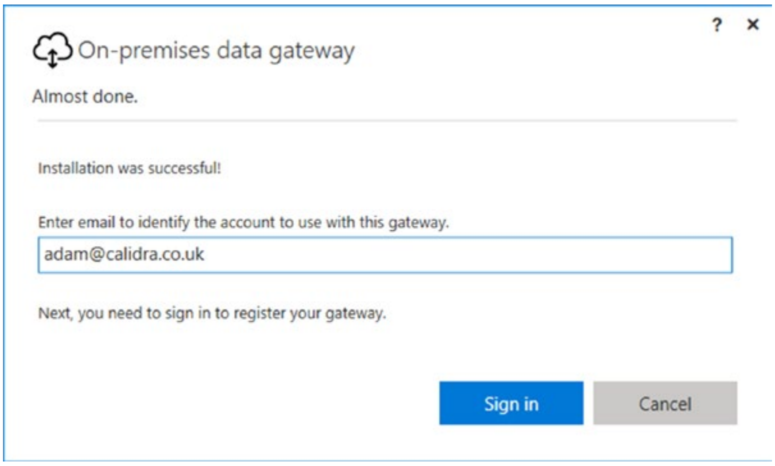
Install to

C:\Users\Adam\AppData\Local\Microsoft\On-premises data gateway (personal n

☑ I accept the terms of use and privacy statement

Figure 23-35. *Gateway location*

8. Click Install. Once the installation process has finished, you will see the gateway installation sign-in dialog, like the one in Figure 23-36. Enter the e-mail address of the account that you will be using.



Almost done.

Installation was successful!

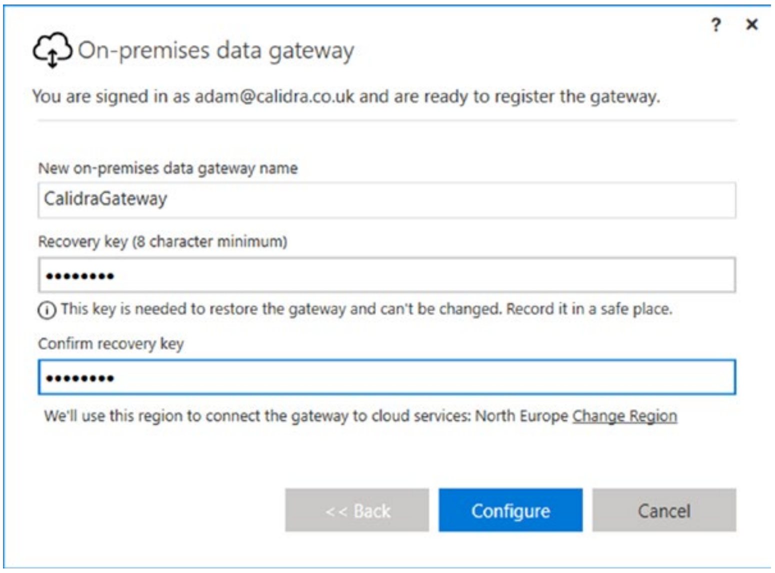
Enter email to identify the account to use with this gateway.

adam@calidra.co.uk

Next, you need to sign in to register your gateway.

Figure 23-36. *Gateway successfully installed dialog*

9. Click the Sign In button.
10. Enter your password and confirm. You will see the dialog for entering the gateway name and recovery key.
11. Enter a gateway name and a recovery key, as shown in Figure 23-37.



You are signed in as adam@calidra.co.uk and are ready to register the gateway.

New on-premises data gateway name

CalidraGateway

Recovery key (8 character minimum)

••••••••

ⓘ This key is needed to restore the gateway and can't be changed. Record it in a safe place.

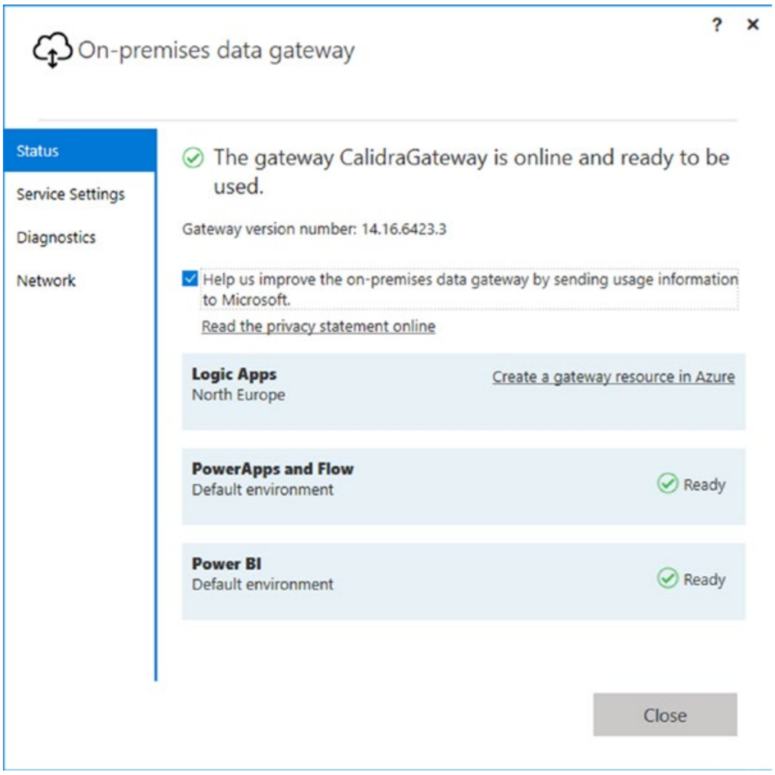
Confirm recovery key

••••••••

We'll use this region to connect the gateway to cloud services: North Europe Change Region

Figure 23-37. *The gateway name and recovery key dialog*

12. Click Configure. Once you are connected to your PowerBI.com account, you will see a dialog confirming a successful connection. You can see this in Figure 23-38.



The gateway CalidraGateway is online and ready to be used.

Gateway version number: 14.16.6423.3

☑ Help us improve the on-premises data gateway by sending usage information to Microsoft.

Read the privacy statement online

Logic Apps
North Europe
Create a gateway resource in Azure

PowerApps and Flow
Default environment
✓ Ready

Power BI
Default environment
✓ Ready

Figure 23-38. *Gateway online dialog*

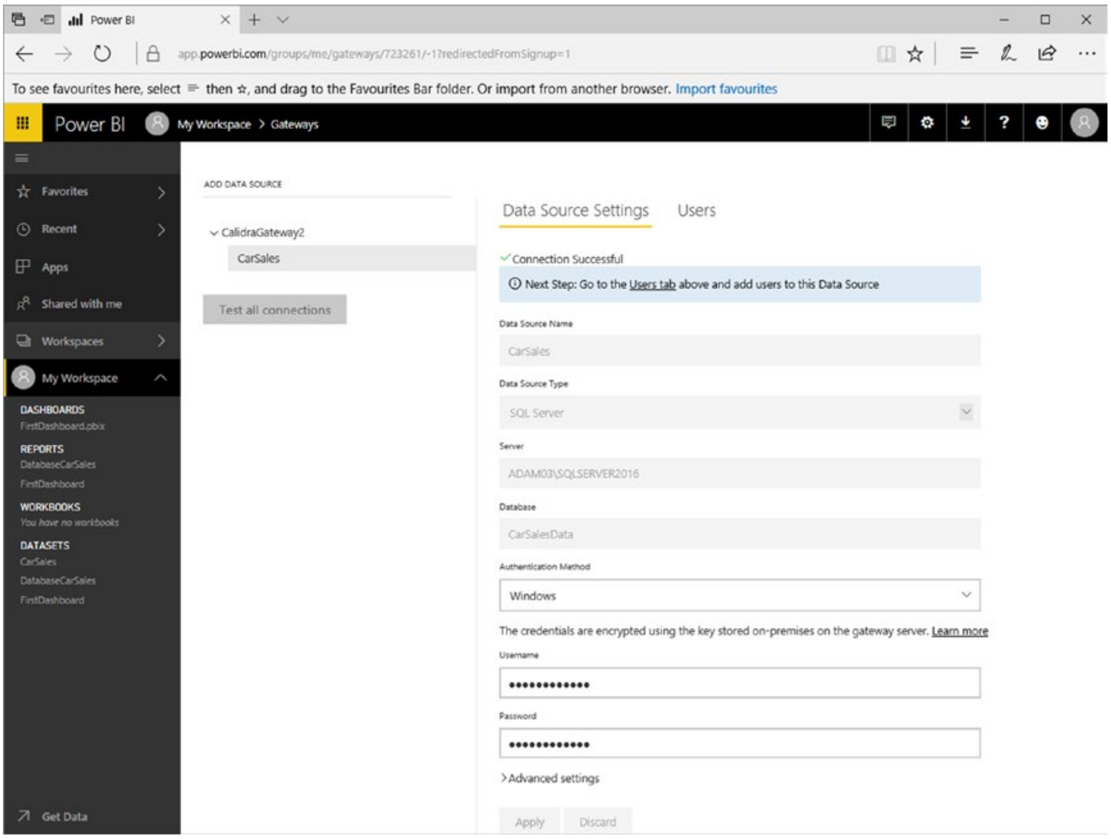
13. Click Close.

■ **Note** PowerBI.com Gateway is an app on your PC. So you can always find it (depending on your version of Windows) among the installed applications on your computer.

Configuring Data Sources

Once a gateway is installed, you will need to configure it to access on-premises data sources.

You can see this in Figure 23-39.



Connection Successful

Next Step: Go to the Users tab above and add users to this Data Source

Data Source Name: CarSales

Data Source Type: SQL Server

Server: ADAM03\SQLSERVER2016

Database: CarSalesData

Authentication Method: Windows

The credentials are encrypted using the key stored on-premises on the gateway server. Learn more

Username: ●●●●●●●●●●●●

Password: ●●●●●●●●●●●●

> Advanced settings

Figure 23-39. *Configuring a data source*

Once you have configured the data source, you must add users, as you can see in Figure 23-40.

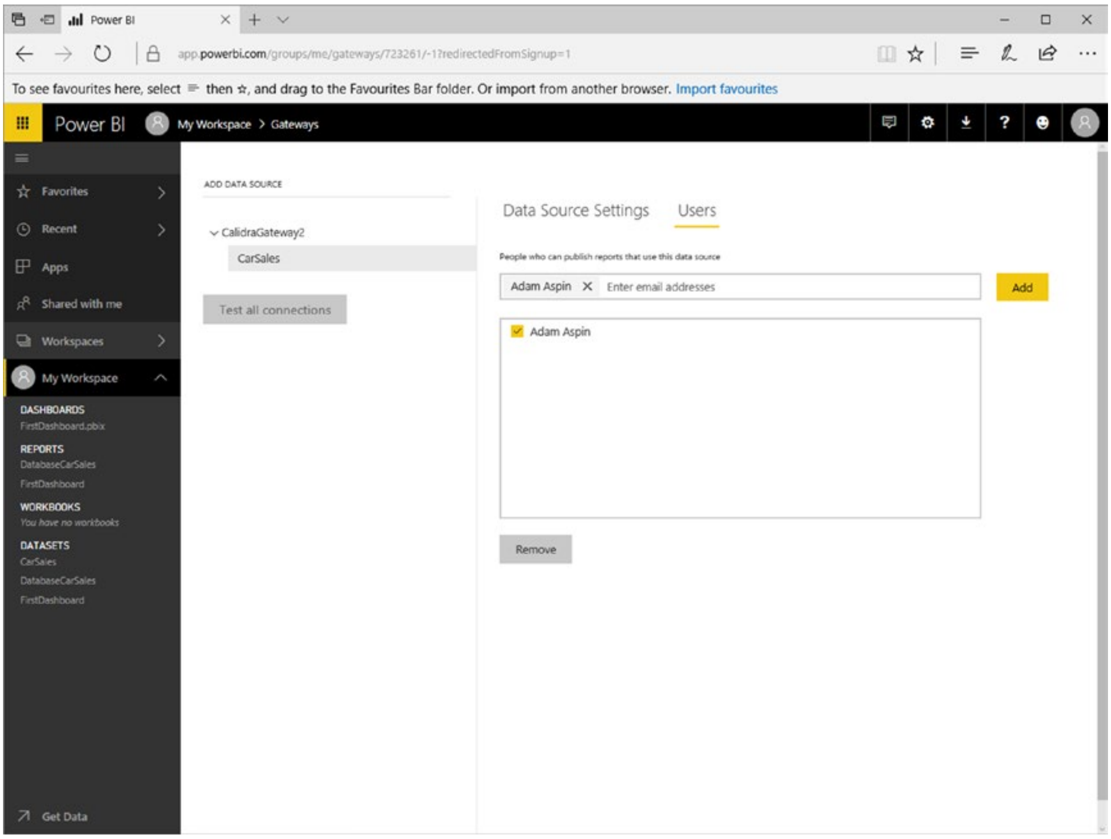


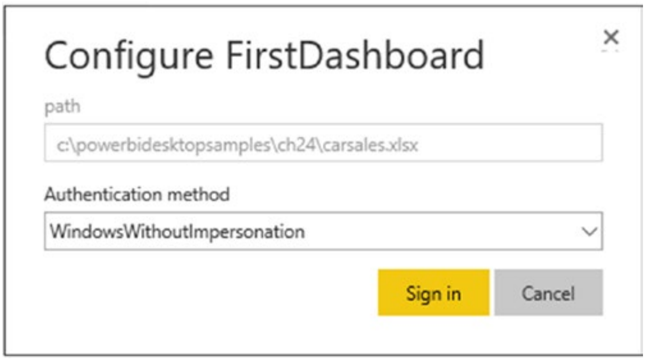
Figure 23-40. *Adding users for a data source*

Applying Dataset Settings

Finally, you need to apply dataset settings for individual datasets:

1. On the left of the PowerBI.com web page, expand My Workspace, and then expand the Datasets section.
2. Click the menu (the ellipses) for an existing dataset.
3. Select Dataset Settings from the popup menu.
4. Select the dataset that you loaded earlier (FirstDashboard).

5. Expand Data Source Credentials on the main page, and click Edit Credentials. The Configure dialog for this dataset will appear, as shown in Figure 23-41.


Configure FirstDashboard

path

c:\powerbidesktopsamples\ch24\carsales.xlsx

Authentication method

WindowsWithoutImpersonation

Figure 23-41. *The Configure dialog for a dataset*

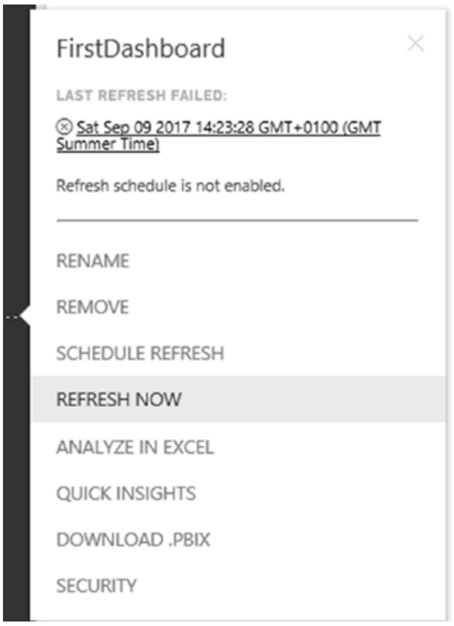
6. Click the Sign In button. An alert will confirm that the data source connection has been updated.

You can now refresh the data in PowerBI.com from the source data.

Manual Data Refresh

As an example of how to use a gateway that you have installed on your PC, let's apply a manual refresh to the file that you loaded at the start of this chapter:

1. In PowerBI.com, click the options menu (the ellipses) for the dataset that you want to refresh (FirstDashboard in this example). The options menu will be displayed, as shown in Figure 23-42.


FirstDashboard

LAST REFRESH FAILED:

Sat Sep 09 2017 14:23:28 GMT+0100 (GMT Summer Time)

Refresh schedule is not enabled.

Figure 23-42. *The dataset options menu*

2. Click Refresh Now.

The dataset will be refreshed with the latest data from the source. This may take a minute or two. Of course, all your visuals will be updated to reflect any changes in the source data.

■ **Note** Alternatively, you can carry out ad hoc data refresh by clicking the Refresh button in the menu that appears above each report that you select in PowerBI.com.

Scheduled Data Refresh

Once a gateway is correctly installed and configured, you can specify which data source(s) you want to refresh according to a schedule, as follows. This enables you to ensure that your data is always up to date.

1. In PowerBI.com, click the options menu (the ellipses) for the dataset that you want to refresh (FirstDashboard in this example). The options menu will be displayed.
2. Select Schedule Refresh. The Datasets Settings page will be displayed.
3. Expand Schedule Refresh.
4. Set the "Keep your data up to date" switch to Yes.
5. Choose a refresh frequency (daily or weekly).
6. Click Apply.

The selected dataset will now be refreshed according to the schedule that you have chosen. If you have downloaded and installed the Enterprise gateway, then you can refresh your data hourly instead of daily.

You can also refresh larger amounts of data.

PowerBI.com on Tablets and Smartphones

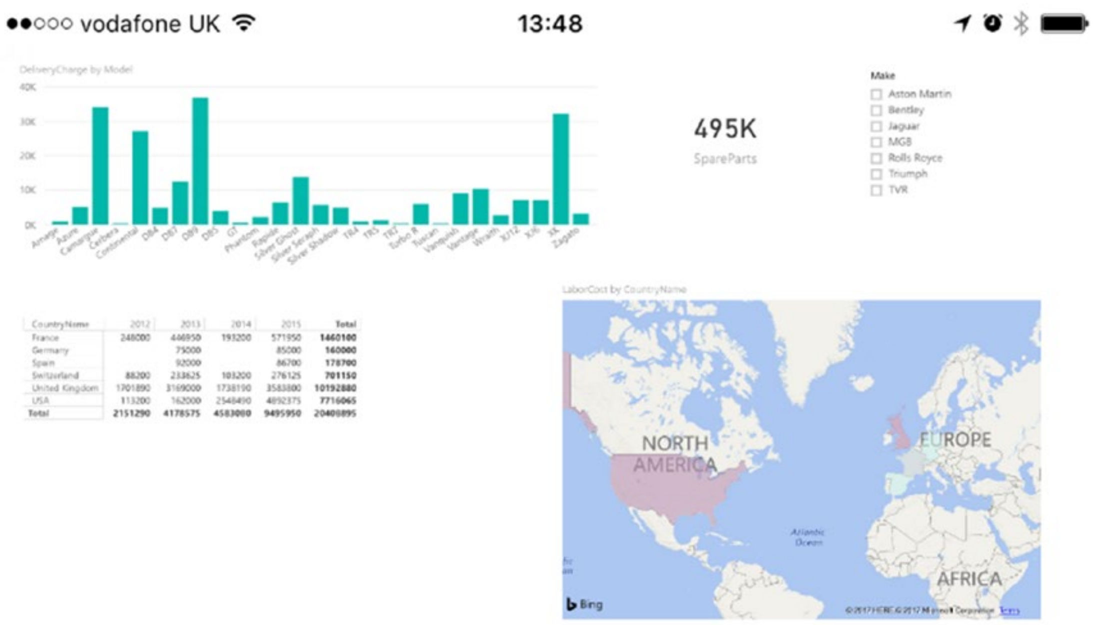
Power BI is also available as apps for the following devices:

- IOS tablets and smartphones
- Android tablets and smartphones
- Windows tablets and smartphones

The first step, inevitably, is to load the app on your device. The easiest way is to go to the Apple App store if you have an iPhone or iPad, or Google Play if you are using an Android device. If you have a Windows tablet, you can use the URL: `https://www.microsoft.com/en-gb/store/p/microsoft-power-bi/9nblgggzlxn1?rtc=1`.

Once the app is installed, all you have to do is log in and you can access any Power BI reports, data, and dashboards that you have loaded from Power BI Desktop.

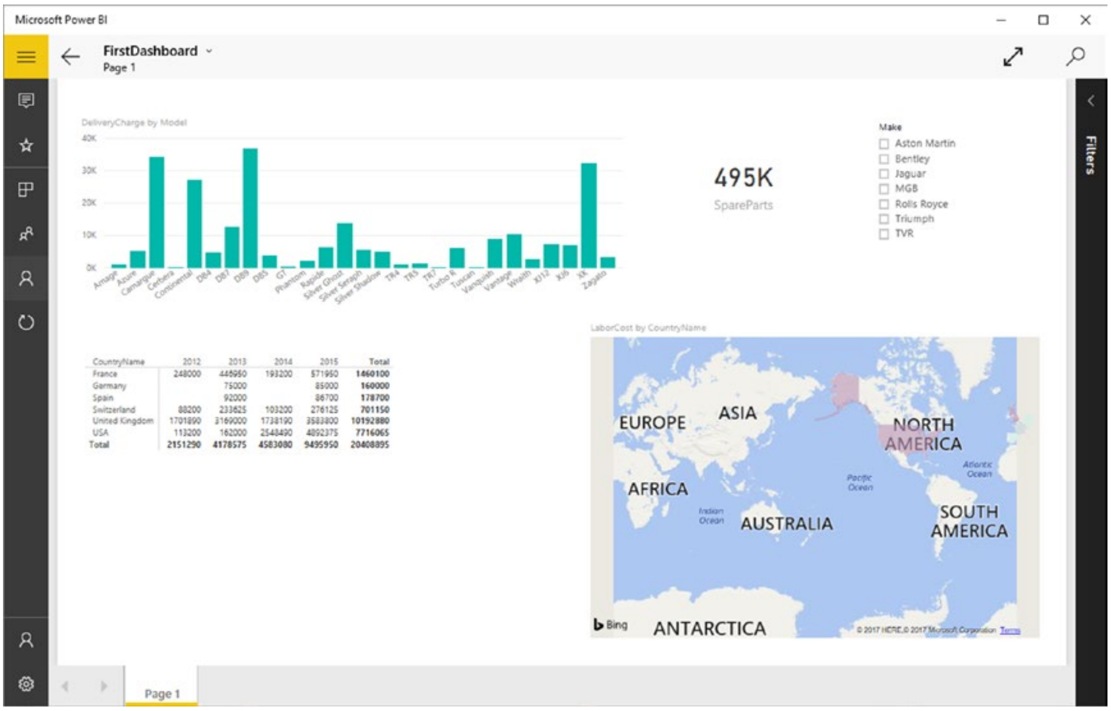
Figure 23-43 shows you the dashboard that you loaded earlier displayed on an iPhone.



CountryName	2012	2013	2014	2015	Total
France	248000	446950	193200	571950	**1460100**
Germany		75000		85000	**160000**
Spain		92000		86700	**178700**
Switzerland	88200	233625	103200	276125	**701150**
United Kingdom	1701890	3169000	1738190	3583800	**10192880**
USA	113200	162000	2548490	4892375	**7716065**
Total	**2151290**	**4178575**	**4583080**	**9495950**	**20408895**

Figure 23-43. *A dashboard on an iPhone*

Figure 23-44 shows you the same dashboard on a Windows tablet in the Power BI App.



CountryName	2012	2013	2014	2015	Total
France	248000	446950	193200	571950	**1460100**
Germany		75000		85000	**160000**
Spain		92000		86700	**178700**
Switzerland	88200	233625	103200	276125	**701150**
United Kingdom	1701890	3169000	1738190	3583800	**10192880**
USA	113200	162000	2548490	4892375	**7716065**
Total	**2151290**	**4178575**	**4583080**	**9495950**	**20408895**

Figure 23-44. *A dashboard on a Windows tablet*

Conclusion

Over the course of this book, you have seen how to develop the data discovery, modeling, and visualization capabilities of Power BI Desktop. As the culmination of your journey into self-service BI, this chapter has shown you the new ways you can share discoveries and collaborate from anywhere using PowerBI.com. This has included adding files to a PowerBI.com, managing the connection to on-premises source data, configuring automated data updates, and allowing controlled access to corporate data if you are using the Enterprise gateway.

Once you master and implement these techniques and technologies, PowerBI.com could really become a dynamic online hub for insight and collaboration, data reuse, and interaction among your colleagues. I sincerely hope that you will have fun using Power BI and develop some awesome uses for this amazing technology.

Sample Data

Sample Data

If you wish to follow the examples used in this book—and I hope you will—you will need some sample data to work with. All the files referenced in this book are available for download and can easily be installed on your local PC. This appendix explains where to obtain the sample files, how to install them, and what they are used for.

Downloading the Sample Data

The sample files used in this book are currently available on the Apress site. You can access them as follows:

1. In your web browser, navigate to the following URL `www.apress.com/9781484232095`.
2. Scroll down the page and click the tab Source Code/Downloads.
3. Click the link Download Now, and choose a directory where you will save the file PowerBIDesktopSamples.zip.

You will then need to extract the files and directories from the zip file. How you do this will depend on which software you are using to handle zipped files. If you are not using any third-party software, then one way to do this is

1. Create a directory named C:\PowerBIDesktopSamples.
2. In the Windows Explorer navigation pane, click the file PowerBIDesktopSamples.zip.
3. Select all the files and folders that it contains.
4. Copy them to the folder that you created in step 1.

Images

The images used in various chapters can be found in the directory C:\PowerBIDesktopSamples\Images.

Sample Databases

If you wish to load data from a SQL Server 2014 database or an Analysis Services database, you will need to restore the sample SQL Server and Analysis Services databases that are in the C:\DataVisualisationInExcel2016\Database directory.

The CarSalesData Database

This database is available in the sample data as the file CarSalesData.bak in the directory C:\PowerBIDesktopSamples\DatabaseBackups. You will also need to create a directory for the database files. In the following code, this is C:\PowerBIDesktopSamples\Databases.

Before you can load this database, you will need access to a functioning SQL Server database instance. If you need to, you can download and install the free SQL Server 2014 Express version. It is currently available at the following URL:

https://www.microsoft.com/en-us/download/details.aspx?id=42299&WT.mc_id=rss_alldownloads_devresources

Once installed, you will need to restore the database backup:

1. Open SQL Server Management Studio Express.
2. Open a new query window by clicking New Query in the toolbar.
3. Run the following script:

```
USE [master] RESTORE DATABASE [CarSalesData]
FROM  DISK = N'C:\PowerBIDesktopSamples\Database\CarSalesData.bak'
WITH  FILE = 1
,  NOUNLOAD,  STATS = 5
,MOVE 'CarSalesData' TO 'C:\PowerBIDesktopSamples\Database\CarSalesData_Data.mdf'
,MOVE 'CarSalesData_Log' TO  C:\PowerBIDesktopSamples\DatabaseCarSalesData_Log.ldf'
GO
```

The database will be restored, and can be used in the examples.

The CarSalesMemoryBased Database

This database is available in the sample data as the file CarSalesMemoryBased.bak in the directory C:\PowerBIDesktopSamples\DatabaseBackups. You will also need to create a directory for the database files. In the following code, this is C:\PowerBIDesktopSamples\Databases.

1. Open SQL Server Management Studio Express.
2. Open a new query window by clicking New Query in the toolbar.

3. Run the following script:

```
USE [master] RESTORE DATABASE [CarSalesData]
FROM  DISK = N'C:\PowerBIDesktopSamples\Database\CarSalesMemoryBased.bak'
WITH  FILE = 1,  NOUNLOAD,  STATS = 5
,MOVE 'CarSalesMemoryBased' TO 'C:\PowerBIDesktopSamples\Database\CarSalesMemoryBased_Data.mdf'
,MOVE ' CarSalesMemoryBased _Log'
TO 'C:\PowerBIDesktopSamples\CarSalesMemoryBased_Log.ldf'
GO
```

The database will be restored, and can be used in the examples.

The Analysis Services Database

To restore the Analysis Services database, you will first need a functioning SSAS instance. You can then restore the file CarSalesOLAP,abf using the standard SSAS database restoration techniques.

The SSAS Tabular Database

To restore the Analysis Services database, you will first need a functioning SSAS tabular instance. You can then restore the file CarSalesTabular,abf using the standard SSAS database restoration techniques.

Index