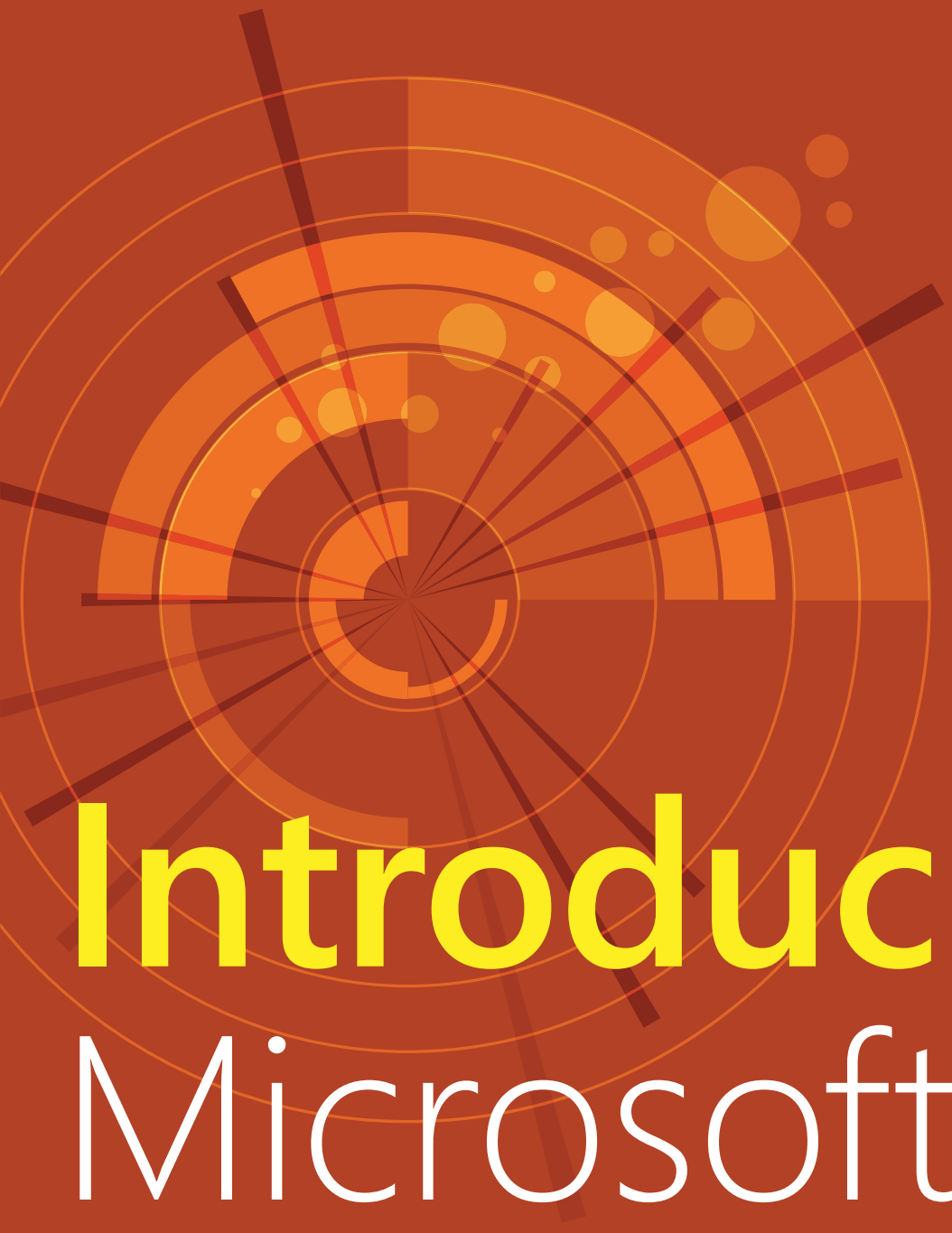


Microsoft

Introducing Microsoft Power BI

Alberto Ferrari and Marco Russo

PUBLISHED BY
Microsoft Press
A division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

ISBN: 978-1-5093-0228-4

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Support at mspinput@microsoft.com. Please tell us what you think of this book at http://aka.ms/tellpress.

This book is provided "as-is" and expresses the author's views and opinions. The views, opinions and information expressed in this book, including URL and other Internet website references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

Acquisitions and Developmental Editor: Rosemary Caperton
Editorial Production: Dianne Russell, Octal Publishing, Inc.
Copyeditor: Bob Russell, Octal Publishing, Inc.
Technical Reviewer: Ed Price; Technical Review services provided by Content Master, a member of CM Group, Ltd.
Cover: Twist Creative • Seattle

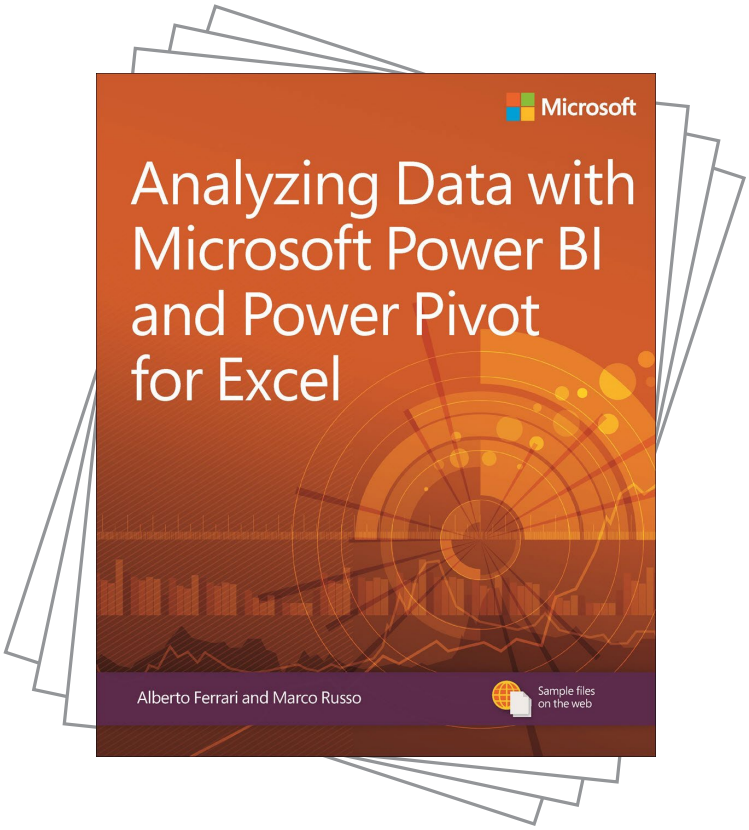
Visit us today at

microsoftpressstore.com

- **Hundreds of titles available** – Books, eBooks, and online resources from industry experts
- **Free U.S. shipping**
- **eBooks in multiple formats** – Read on your computer, tablet, mobile device, or e-reader
- **Print & eBook Best Value Packs**
- **eBook Deal of the Week** – Save up to 60% on featured titles
- **Newsletter and special offers** – Be the first to hear about new releases, specials, and more
- **Register your book** – Get additional benefits

Microsoft

Contents



Exclusive offer for *Introducing Microsoft Power BI* readers!

Available this fall! Be the first to receive an **exclusive pre-order discount** for *Analyzing Data with Microsoft Power BI and Power Pivot for Excel* by expert trainers Alberto Ferrari and Marco Russo. It's easy:

- Sign up to receive **special offers from Microsoft Press**
- Watch your inbox for an exclusive email offer during the first week of September

Visit **MicrosoftPressStore.com/PowerBI** to get started. Even better? You'll receive a code* to save 35% on your next purchase at the Microsoft Press Store.

Learn more about Power BI at **powerbi.microsoft.com**.

Discount code valid on single book or eBook purchase from microsoftpressstore.com. Code cannot be combined with eBook Deal of the Day, Official Microsoft Practice Tests fulfilled by MeasureUp, or any other offer. Offer expires November 31, 2016.

Microsoft

Introduction

Microsoft introduced the idea of Self-Service Business Intelligence (BI) back in 2009, announcing Power Pivot for Microsoft Excel 2010. Strangely, at that time, it did not make big announcements, hold conferences, or undertake a big marketing campaign for it. Everything started slowly, with some enthusiastic users adopting the new technology, but the vast majority of people did not even know about its existence. As part of the community of BI professionals, we were very surprised by that approach. At the time, we could clearly see the advantages for users to begin adopting Power Pivot as a tool for gathering insights from data, so this complete lack of marketing was somewhat disappointing.

Thus, for several years we (as a community) kept asking Microsoft what they were waiting for; what was the delay in promoting Self-Service BI to the greater audience of data analysts, data scientists, decision makers, and BI enthusiasts all over the planet. We asked for the ability to share reports with a team, and the answer was to use SharePoint, either on-premises or the online version, with the first release of Power BI—an experience that was still not completely satisfactory. While we were waiting for Microsoft to fix the issues with the previous versions and to begin advertising the current products, it was doing something different that, with the benefit of hindsight, looks to have been the perfect choice. Microsoft collected the feedback of users, carefully considered what was missing in the world of end-user BI, and then crafted the version of Power BI that's available to you today.

Power BI is an evolution of the add-ins previously available in Excel: Power Pivot, Power Query, and Power View. You can use Power BI with or without Excel—you no longer are dependent on the version of Microsoft Office installed at your company. People did not like to share reports by using only SharePoint, and Microsoft moved away from it. Users wanted a mobile experience, and the development team created it. Data analysts wanted power, simplicity, new visualizations, and all of this is now available in Power BI. In addition, a lot of effort went into the creation of a seamless experience in loading data from many different cloud sources and building the infrastructure needed to provide all BI enthusiasts with a framework with which they can grow their reports, share them with their teams, and refresh the data in a simple yet effective way.

To make a long story short, Microsoft heard the feedback of users and built a great set of tools for the adoption of Self-Service BI. And, now—only now—it has begun marketing it.

Suddenly, in the last few months, Power BI has become the hottest topic at conferences, webinars, talks, and courses. As expected, people like you gathered interest in Power BI and began to search for resources to learn it. This book is one of these resources and its goal is to provide you with an effective introduction to the features available in the new Power BI.

We wanted to write an introduction to Power BI that covers the basics of the tool and, at the same time, shows you what the main capabilities of Power BI are. Thus, it is fair to say that the content of the book is somewhat unbalanced. At the beginning, we go for an easy introduction of the concepts along with an educational approach that lets you follow on your PC the same steps we show in the book. However, if we continued with that same mindset for the entire book, its size would quickly

become intimidating. Thus, after the first chapters, we begin to run a bit faster, knowing that we are no longer guiding you step by step. Instead, we show you available features; if you want to learn the details, you will need to read and study more.

This book is targeted to a variety of readers. There are information workers and people who are totally new to the BI world. For those readers, the book acts as a simple introduction to the concepts that are the foundation of BI. Yet, another category of we wanted to target is that of IT professionals and database administrators who might need to drive the decisions of the company in adopting Power BI, because their users are asking for it. If this is you, this book acts as both a simple introduction to the basic concepts, to help you understand why users are so interested in Power BI, and as an overview of the capabilities and tools available in Power BI, so that you can make educated choices in adopting it. Power BI is not just a tool: it is an ecosystem that can integrate existing corporate BI with Self-Service BI. The last chapter of the book gives you an overview of these capabilities.

We hope you enjoy reading the book as much as we enjoyed writing it. Keep in mind that this is probably your first step in the fascinating world of Self-Service BI, the first step of a long journey in gathering insights from your data.

Downloads

All of the chapters in this book include workbooks and databases that let you interactively try out new material learned in the main text. All sample content can be downloaded from the following page:

http://aka.ms/IntroPowerBI/downloads

Follow the instructions to download the IntroPowerBI_302284_CompanionContent.zip file.

Installing the companion content

Follow these steps to install the companion content on your computer so that you can use them with the exercises in this book.

1. Unzip the IntroPowerBI_302284_CompanionContent.zip file that you downloaded from the book's website (name a specific directory along with directions to create it, if necessary).
2. If prompted, review the displayed end user license agreement. If you accept the terms, select the accept option, and then click Next.

Acknowledgments

As usual with projects of this sort, there are too many people to thank, and a complete list of everyone who contributed to this book would be impossible to write.

Nevertheless, there are certain people we must mention personally, because of their particular contributions.

We want to thank Microsoft Press and all the people there who worked on the project. Rosemary Caperton has been a great editor who helped us immeasurably with the process of book writing. Many others behind the scenes guided us through the complexity of authoring a book—thanks to you all.

Finally, a special mention goes to our technical reviewer, Ed Price. He double-checked all the content of our original text, searching for errors and sentences that were not clear; giving us invaluable suggestions on how to improve the book. Without his meticulous work, the book would have been much harder to read and would contain more mistakes!

If the book contains fewer errors than our original manuscript, it is only because of them. If it still contains errors, it is our fault, of course.

Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

http://aka.ms/mspressfree

Check back often to see what is new!

Errata, updates, & book support

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

http://aka.ms/IntroPowerBI/errata

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at mspinput@microsoft.com.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to http://support.microsoft.com.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://aka.ms/tellpress

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: http://twitter.com/MicrosoftPress.

CHAPTER 1

Introducing Power BI

David is the manager of budgeting at Contoso, a company that sells electronic products worldwide through several retail shops and a website. Around the globe, country/region managers are responsible for producing figures for next year's budget for their respective countries/regions, which David then aggregates to produce the big picture to show to his boss.

Our scenario begins in October 2015, when David commences working on the budget for 2016. As always, David has a Microsoft Excel workbook containing the relevant information to produce the budget. Based on the results of the workbook, he would typically create a Microsoft PowerPoint presentation to share the results during internal meetings. This year, however, David wants to take advantage of the new Power BI service provided by Microsoft.

This entire book is a journey that we'll take along with David as he discovers how Power BI can help to build a rather sophisticated reporting solution; in this case, based on a budgeting system. But because this book is about Power BI, not budgeting, we will not focus on the complexity of building a budget. Instead, we will keep the budgeting considerations fairly basic, focusing on the complexity of teamwork, data modeling, and reporting.

We provided all the workbooks and databases that we used to build the demonstrations in the companion content for this book. If you are interested in learning the basics of Power BI, you can replicate David's activities on your computer so that you can augment your learning experience by following the examples. Be aware, though, that the results you obtain by running the demonstrations might be slightly different, and the appearance of webpages and the user interface might not be identical, either. Power BI is evolving very quickly, and we tried our best to show examples that will

last some time. Nevertheless, differences might occur; thus, you should concentrate on learning the features of Power BI, not the demonstrations. So, even if the numbers end up being different, what's important is to absorb how to do something, not just replicate what you read in the book.

Moreover, we strongly encourage you to test Power BI using your own data. You can perform the same operations on your personal information that we describe in the book, thus reaping the combined benefits of learning the Power BI tools while simultaneously gaining insights into your data.

Getting started with Power BI

Any journey begins with the first step, so let's take that step together.

David obtained from IT an Excel report that contains the sales for the past three years, divided by country/region, brand, and month. Sales in Contoso are strongly brand-oriented, and some brands are prone to seasonal effects that David wants to take into account. For this reason, he uses data grouped by month. Figure 1-1 shows a small portion of the resulting data, which he stores in an Excel file. If you would like to become more familiar with David's data, you can open 2015 Sales.xlsx from the book's companion content.

CountryRegion	Brand	Month	Sale 2013	Sale 2014	Sale 2015
China	Contoso	January	1,271.55	3,236.31	9,945.86
China	Contoso	February	7,372.34	5,995.55	1,303.01
China	Contoso	March	11,364.07	2,767.32	3,437.26
China	Contoso	April	13,590.94	17,530.47	25,007.74
China	Contoso	May	11,780.08	13,501.47	5,270.05
China	Contoso	June	8,562.00	8,338.87	7,357.26
China	Contoso	July	7,001.78	13,874.09	31,571.42
China	Contoso	August	605.89	3,340.02	1,386.20
China	Contoso	September	3,958.70	1,709.68	2,224.36
China	Contoso	October	8,785.89	5,787.57	10,792.05
China	Contoso	November	7,631.64	2,412.91	
China	Contoso	December	21,968.86	5,657.70	
China	Fabrikam	January	5,794.38	9,244.97	
China	Fabrikam	February		6,230.00	
China	Fabrikam	March	3,415.96	1,044.89	
China	Fabrikam	April			5,155.78
China	Fabrikam	May	3,657.95	2,954.00	20,177.94

Figure 1-1: An excerpt from the initial Excel workbook for David's budget plan.

Every year, David makes some considerations on these numbers and then he shares his findings with the country/region managers, who then send back to him workbooks with their numbers for the next year. Figure 1-1 shows some data from China, but there are several other countries/regions, as well. During the process of computing those numbers, there are many meetings and discussions in which the managers bring their experience and knowledge to bear on the process, adding their own versions of the original workbooks, each displaying various charts and calculations, which must all be explained to others. This is a daunting task, to be certain, and one that David would like to streamline.

Fortunately, David heard about an interesting tool called Power BI that Microsoft created in 2015 that might be helpful toward creating a collaborative environment in which any stakeholder of the budgeting process can share his findings with others, working together on the goal. But, at this point, the name and maybe a marketing video is all that David knows about Power BI.

Driven by curiosity, he navigates to www.powerbi.com and starts down his learning path. Figure 1-2 depicts the welcome page of the Power BI website.

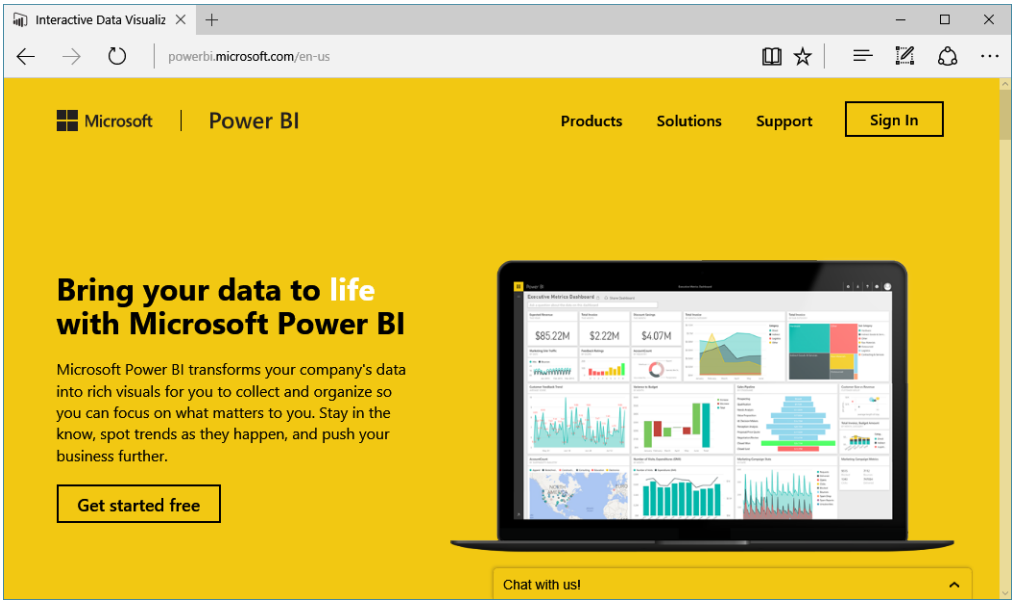


Figure 1-2: The welcome page of Power BI, the starting point of David's journey.

To begin, David clicks the Get Started Free button. He is then offered a choice as to which experience he would prefer to use: he can choose either Power BI Desktop For Windows or just Power BI, as shown in Figure 1-3.

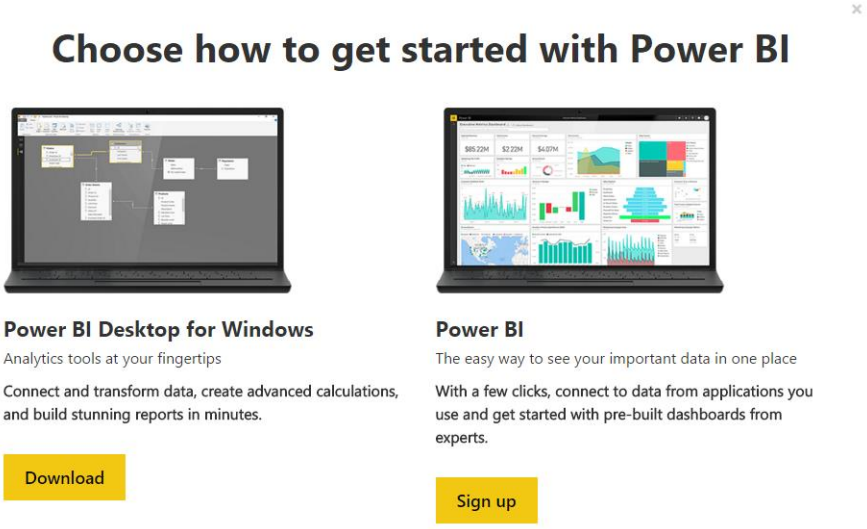


Figure 1-3: You can start with Power BI by using either of the two main experiences.

Actually, there is very little difference between the two. In fact, Power BI Desktop and Power BI are two sides of the same coin: Power BI Desktop is a Windows application running on your PC, whereas Power BI is a cloud service that you use through the web browser. In both cases, you will be able to perform the same operations, albeit with some subtle differences. Moreover, the two tools complement each other, and you are likely to use both to build your dashboards.

After reading the descriptions, David correctly concludes that Power BI Desktop is designed for more advanced tasks. Given that he's just beginning to learn about it, he opts for plain, vanilla Power BI.

When David clicks the Sign Up button, the screen shown in Figure 1-4 appears. Power BI is a web service to which you can upload data and build insightful dashboards and charts. As with any web service, you need to sign in, but Power BI does not require much in the way of credentials: to get started, all you need is a valid email address, which David provides.

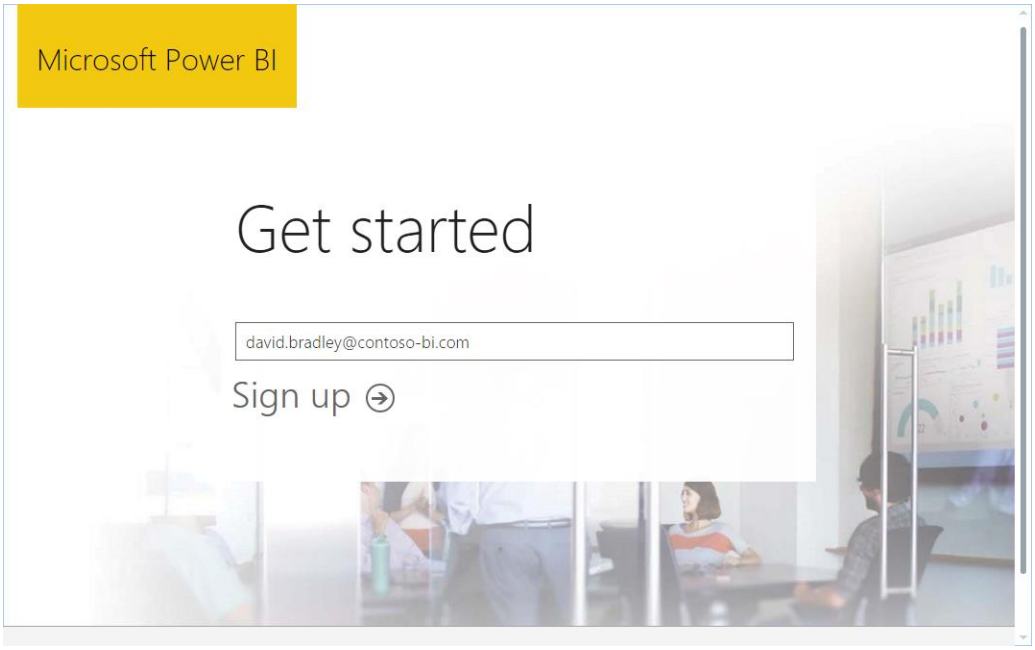


Figure 1-4: You need only a valid email address to gain access to Power BI.

After clicking Sign Up, Power BI informs David that he already has a subscription to Microsoft Office 365; those credentials are sufficient to gain access to Power BI.

> **Note** If you do not have an Office 365 account, Power BI will send you an email with a link to complete the registration process. (Be aware that you cannot use a personal email service such as Hotmail, Yahoo, or Gmail.) This is to ensure that you actually own the email address. Following the link directs you to the registration page, where you provide some basic details such as first name, last name, and so on. In both cases, no credit card or any other form of payment is required, because most of the features of Power BI are totally free.

On the same No Need To Sign Up page, David could click OK, Got It to sign in without any additional steps. Rather than do that, he goes back to the sign-in page, but instead of clicking Sign Up, he signs in to the portal by using the Sign In button (see Figure 1-2) and then provides his Office 365 credentials. The system takes a few seconds to prepare his account, after which David gets his first glimpse at the Power BI portal, as shown in Figure 1-5.

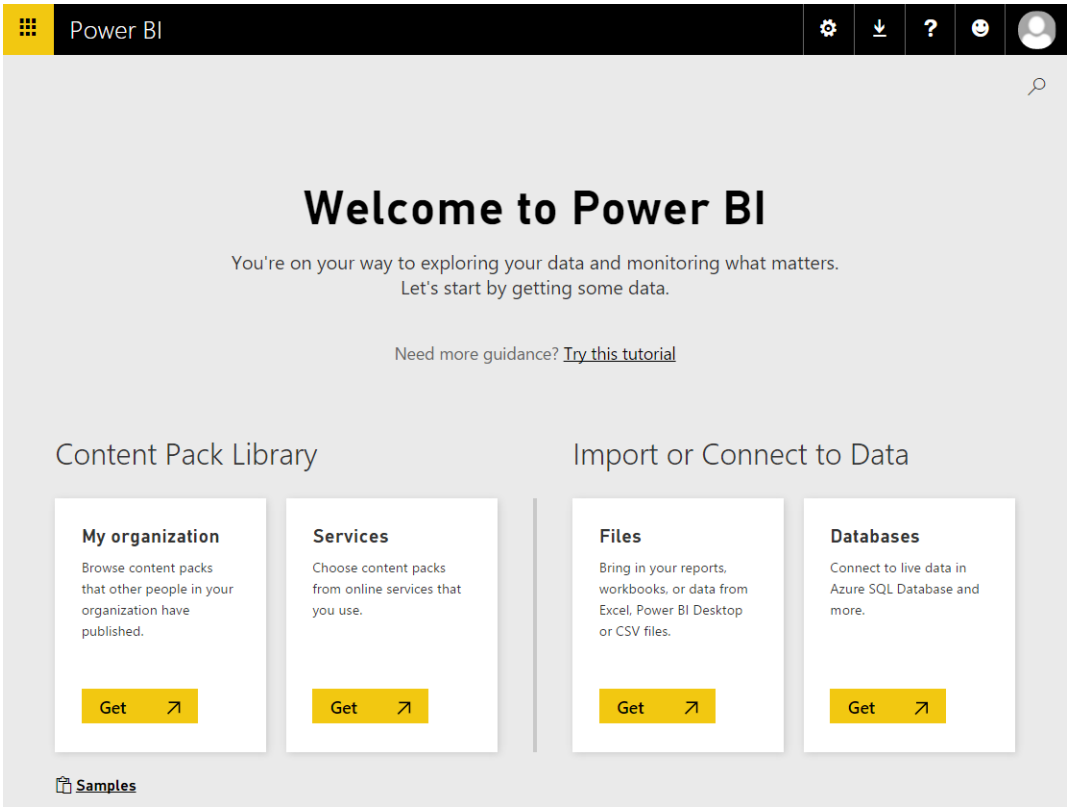


Figure 1-5: The introduction page of the Power BI portal.

Uploading data to Power BI

David has an Excel workbook that he wants to upload to Power BI to see what it has to offer. Because the data is stored in a local file on his laptop, he clicks the Get button on the Files tile (see Figure 1-5). This displays the screen in Figure 1-6, where he can then choose from among several upload options.

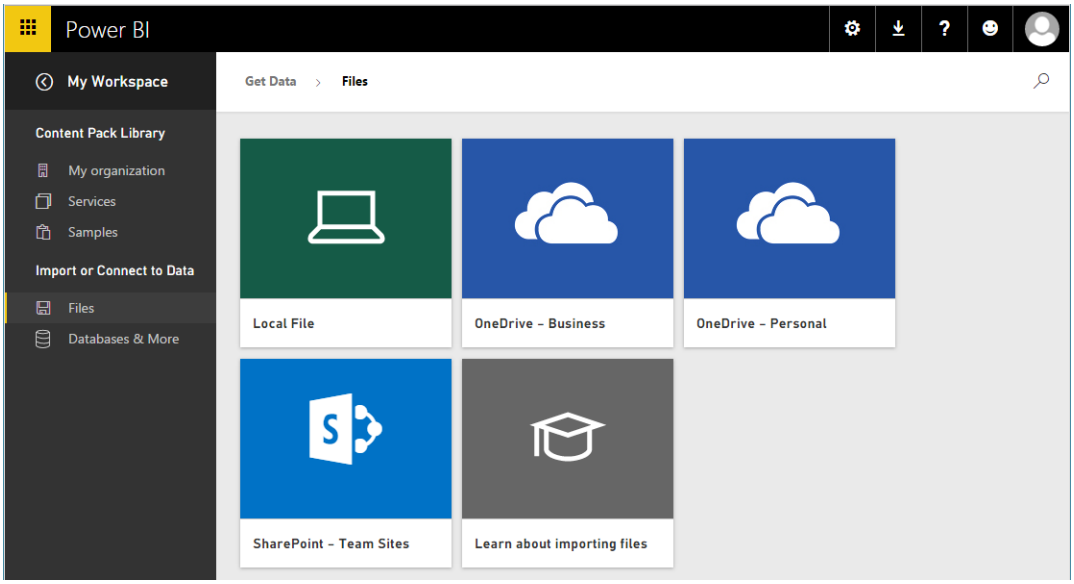


Figure 1-6: Some of the file uploading options in Power BI.

We will explore these options at greater length in the chapters that follow. For now, David chooses Local File, navigates to a file on his laptop named 2015 Sales.xlsx, and then clicks Open to upload the workbook to Power BI. After a few seconds, the Power BI dashboard displays the screen depicted in Figure 1-7.

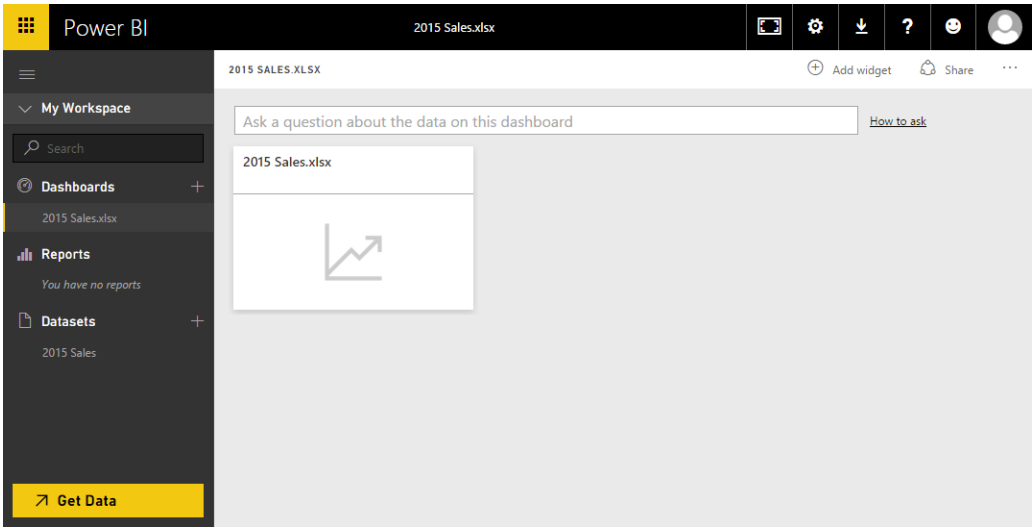


Figure 1-7: This is how the Power BI service looks after you load an Excel workbook.

Before going any further, we want to take a few moments to explain how the Power BI portal is organized. On the left side of the screen, in the pane labeled My Workspace, there are several items. Let's take a look at them:

- **Dashboards** This lists all of the dashboards you have created. After loading a single workbook, Power BI creates a dashboard for you, using the same name as that of the original workbook.
- **Reports** Here, you will see the reports based on your data. In Figure 1-7, there is no default report, but we'll follow along as David creates one very soon.
- **Datasets** This lists all of the data sources that you connected to Power BI. In our narrative thus far, the only workbook David loaded is 2015 Sales.

The Power BI experience is all about gaining insights from data. You begin with a dataset (2015 Sales, in this example), you then build reports on the data, and, finally, you organize visualizations of the reports into dashboards. You will learn how to perform all of these operations in detail in this book. For the moment, we want only for you to become acquainted with the basic operations.

Referring back to Figure 1-7, the central pane is positioned on the 2015 Sales dashboard and, because David has loaded the file but has not yet performed any analysis on the data it contains, the dashboard is essentially empty, showing only the Ask A Question box and the 2015 Sales.xlsx tile, which indicates that the dashboard is indeed connected to his Excel workbook.

Introducing natural-language queries

With Power BI, you have the ability to carry out analysis of your data by asking it questions, in plain English—no special code or syntax is required. This feature is called *natural-language queries*, and with it, you can ask Power BI to perform tasks in much the same way you would ask one of your colleagues. Let's take a look at an example of how David uses natural-language queries in Power BI.

In the central pane, in the question box, David types a simple query: "Show sales 2015 by brand." Power BI understands the query and presents a bar chart (see Figure 1-8) in which the brands are displayed alphabetically and the length of the bars is proportional to the corresponding sales for each brand in 2015.

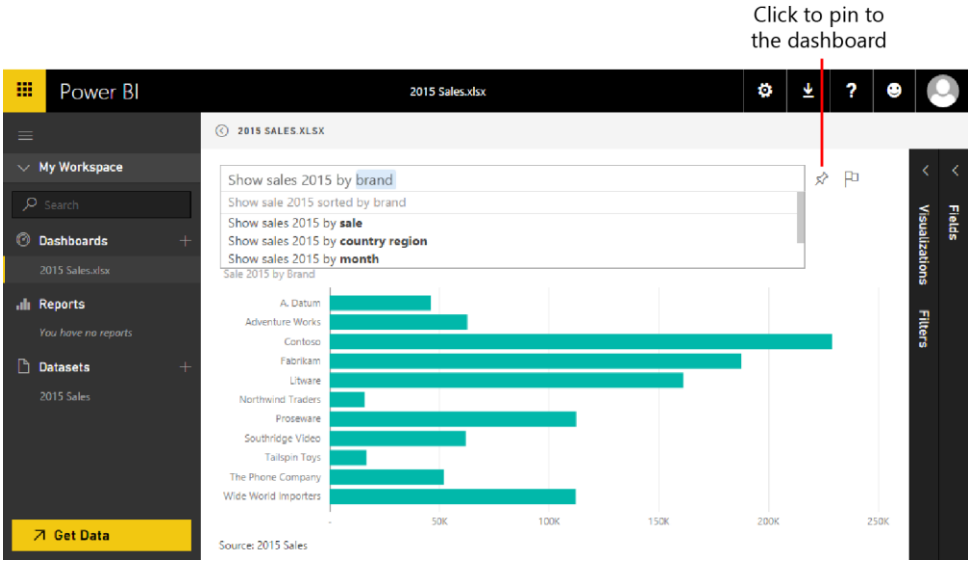


Figure 1-8: Power BI understands queries in natural language and displays the data you request.

Not only did Power BI understand David's query, but, after performing an analysis of his dataset, it also suggests other meaningful queries in a list that appeared when he began to type the query. For David's data, that analysis revealed that he might also be interested in viewing sales in 2015 by country/region or by month, so Power BI suggests those as alternate queries.

Also in Figure 1-8, notice the highlighted pushpin icon to the right of the question box. You can click this to "pin" the currently displayed visualization to the dashboard; this way, you can easily see it when you connect to Power BI. When you click the pushpin button, Power BI opens the Pin To Dashboard dialog box shown in Figure 1-9.

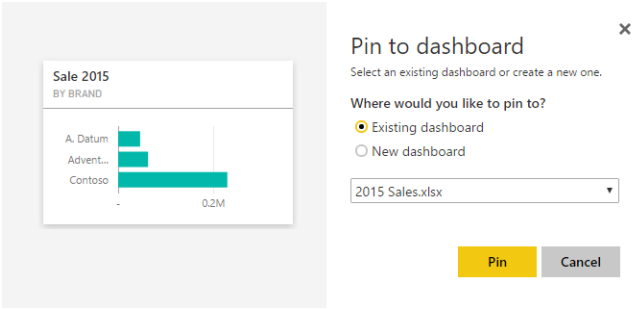


Figure 1-9: Using the Pin To Dashboard dialog box, you can choose to pin a visualization to an existing or a new dashboard.

To save the newly created bar chart to the dashboard, click Pin. Figure 1-10 shows how Power BI presents the dashboard with the pinned bar chart. (You need to go back to the dashboard to see it.)

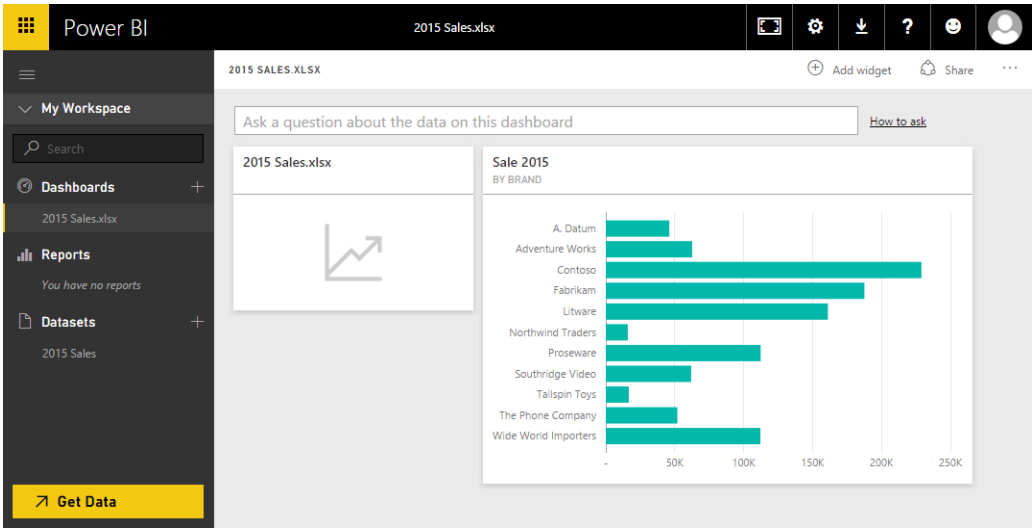


Figure 1-10: The dashboard is a container for visualizations created on top of datasets.

Using natural-language queries is quite impressive, but it is only one of the many ways in which Power BI can analyze your data.

Introducing Quick Insights

Another feature that is worth learning as soon as you begin using Power BI is *Quick Insights*. With this feature, Power BI can search a dataset for interesting patterns and provide you with a list of charts that help you to better understand your data.

To activate Quick Insights, click the ellipsis to the right of the dataset (see Figure 1-11) on which you want to perform the analysis: in David's case, that's "2015 Sales". This opens the dataset menu; here you choose Quick Insights. When David clicks Quick Insights, the button changes to View Insights.

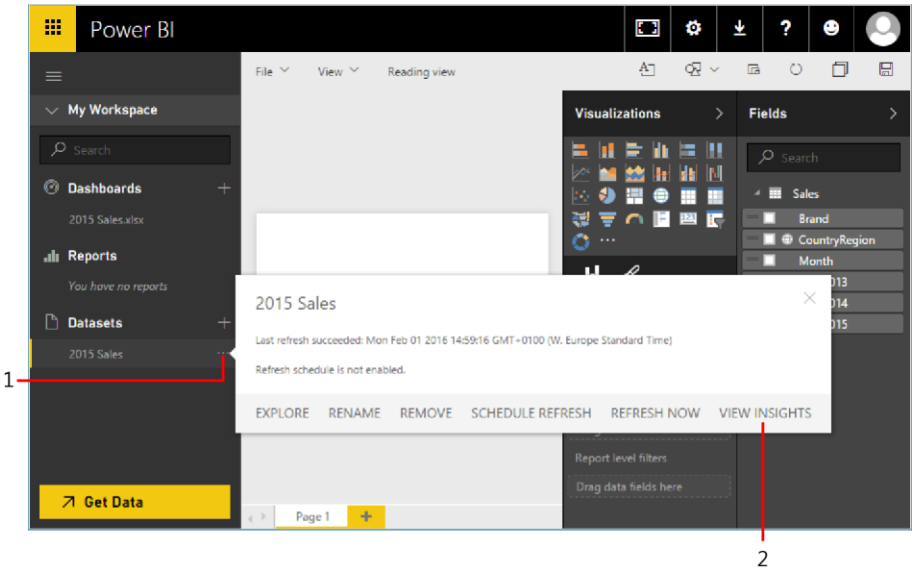


Figure 1-11: You can activate Quick Insights from the dataset menu by clicking Quick Insights.

The first time you run Quick Insights on a dataset, Power BI schedules an analysis of that dataset. This might last for some seconds or minutes, depending on the size of the data. When the search for insights is complete, Power BI notifies you. Of course, whenever you update your dataset, this search operation will need to be repeated. However, as long as the dataset remains unchanged, the insights will be immediately available.

But, what are these insights?

The basic idea is that Power BI can use artificial intelligence to analyze your data, searching for some useful or interesting patterns. It uses very sophisticated algorithms whose speed depends on the size and complexity of the dataset. Obviously, on a small dataset such as the one David uploaded, finding insights takes no more than a few seconds. As soon as the search is complete, you can access it. On the Insights Are Ready dialog box, David clicks View Insights. Figure 1-12 presents the first two insights that Power BI found on David's file. Many others are within the list, so many, in fact, that they would not fit on the page in this book. David scrolls down to view them all.

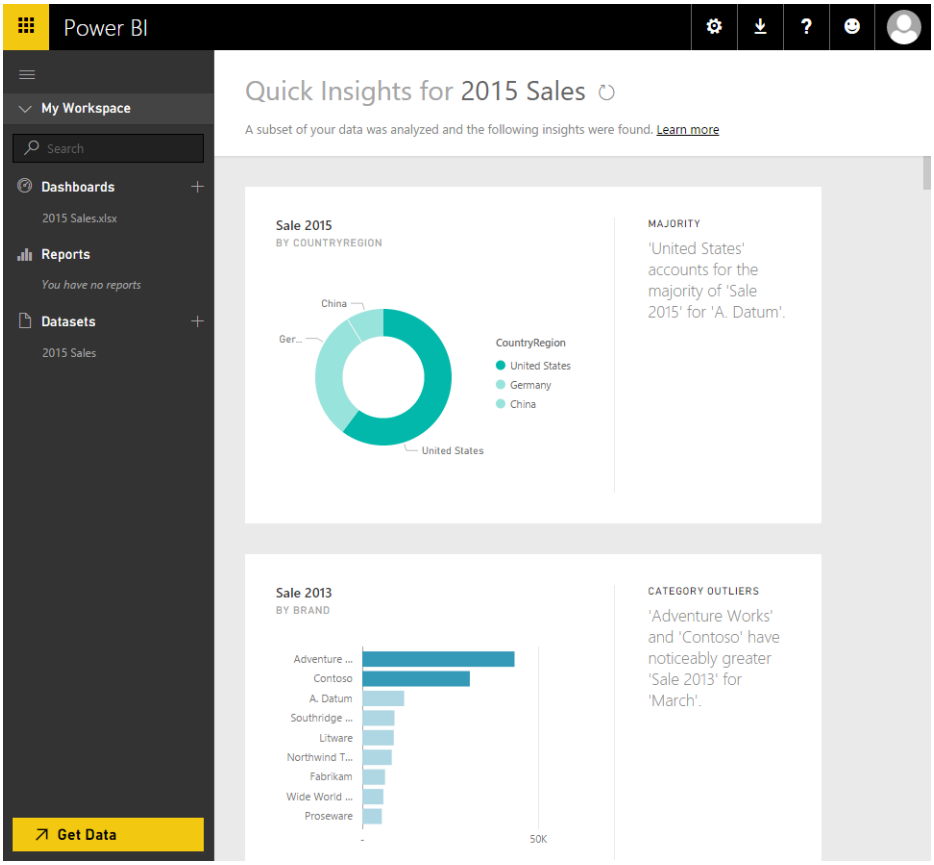


Figure 1-12: Quick Insights are a powerful analytical tool to glean information from your data.

The first insight shows that the United States accounts for most of the sales of the A. Datum brand, compared with China and Germany. The second insight reveals a substantial seasonal-effect increase in sales for the month of March for Adventure Works and Contoso. If you run Quick Insights on the data, you will likely get different insights, which Power BI chooses to display at the top.

Of course, insights are gathered by Power BI without it having any knowledge of your business or the economic scenario as a whole, so there might be many different reasons that explain the data and findings. Power BI cannot replace your brain when it comes to interpreting the numbers, but it can prove extremely useful because it can easily find some points of interests in your data by using the brute force of algorithms.

The best way you can use Quick Insights is to browse through them, looking for the confirmation of what you already know about your data and, at the same time, for fresh ideas. It might be the case that some of the insights are not really meaningful, but, with the sheer number of insights that Power BI finds for you, it's likely that there are some real hidden gems that might improve your knowledge of your numbers.

> **More info** You can find a more complete description of the algorithms used by Power BI and the types of insights that it can reveal by going to https://powerbi.microsoft.com/en-US/documentation/powerbi-service-auto-insights-types/. Of course, with newer versions of the analytics engine, the numbers and the quality of insights might change and improve.

You can click any insight to enlarge it. If you hover over one, the same pushpin button as that of the natural-language query appears so that you can pin the insight to the dashboard if you want. David clicks the Category Outliers Insight from Figure 1-12 to enlarge it, clicks the Pin icon, and then in the Pin To Dashboard dialog box, he leaves Existing Dashboard selected and clicks Pin to pin it to his dashboard.

Pinning one of the insights to the dashboard makes it more interesting. Moreover, by doing that, you will learn that you can move and resize visualizations pinned to a dashboard by using a convenient grid, making them more aesthetically appealing. David returns to look at his expanded 2015 Sales.xlsx dashboard, and he moves the new Sale 2013 By Brand visualization below the others. Figure 1-13 demonstrates David's dashboard, which now contains two visualizations.

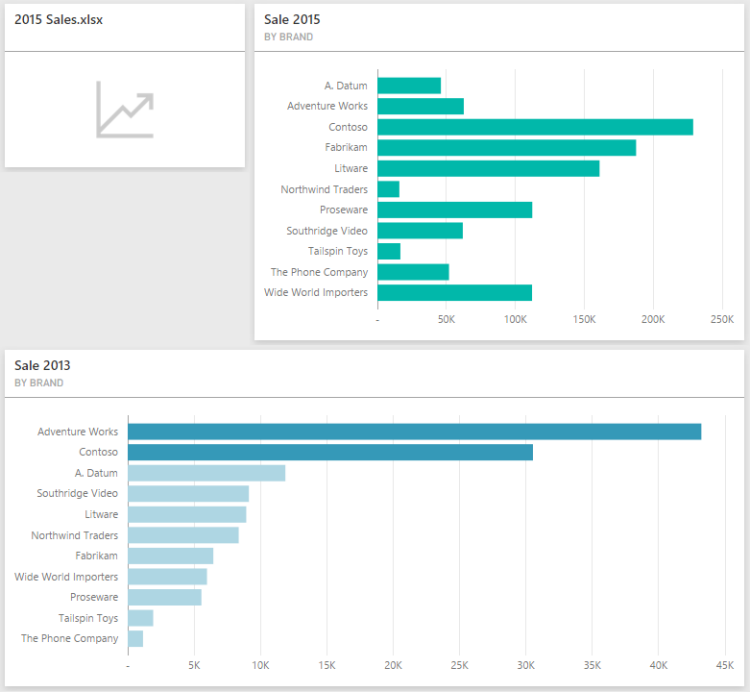


Figure 1-13: A dashboard can contain multiple visualizations organized in a grid, individually moved and resized.

Introduction to reports

So far, David has used only automated report building, using a natural-language query as well as the Quick Insights feature. As you might imagine, he only scratched the surface of Power BI's reporting capabilities. In fact, he can build reports manually, unleashing the full potential of Power BI visualizations.

To create a new report, in the Datasets section of the navigation pane, click a dataset. David clicks 2015 Sales. Power BI opens an empty report based on that dataset, as illustrated in Figure 1-14.

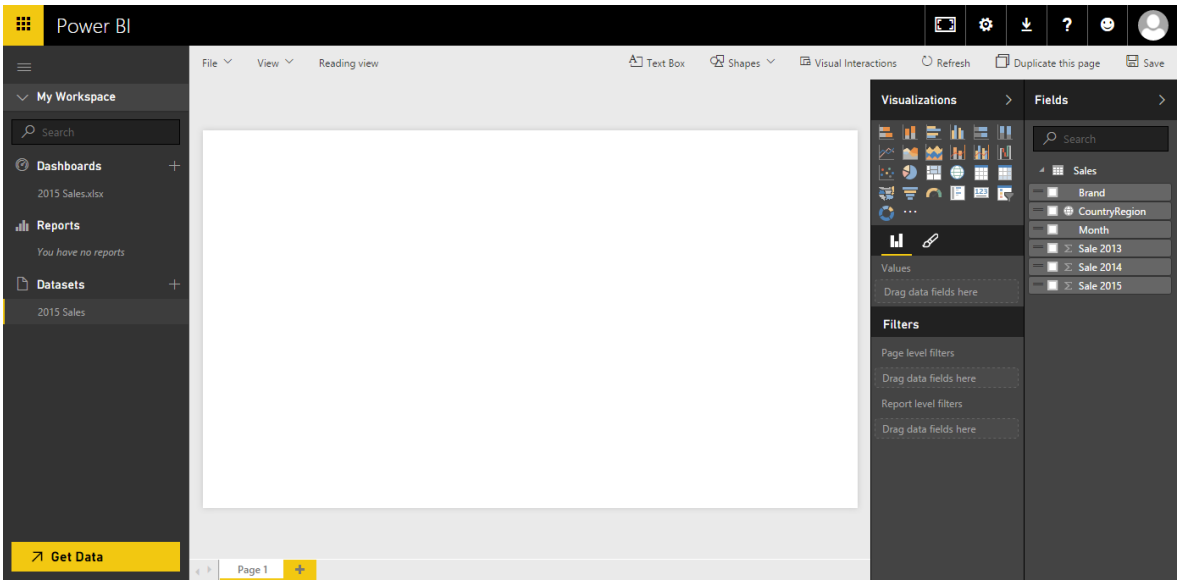


Figure 1-14: Clicking a dataset creates an empty report based on that dataset.

The user interface of a report is very powerful because it combines many different features in a single window. On the far left is the standard Power BI navigation pane. The central pane is the canvas on which you can build a report by adding visualizations. Here, you can also configure the properties of each visualization. On the right are two panes: Visualizations and Fields. The Visualizations pane offers the entire set of available visualizations at the top; the bottom section presents filtering options. The Fields pane contains the list of all the fields of your dataset. In David's case, you can see how the Fields pane lists the columns of the Excel table he uploaded to Power BI.

Figure 1-15 shows an enlarged view of the Fields pane. If you focus your attention on the individual columns there, you can see that some of them have a small icon beside their names. This icon identifies the main usage of the field. For example, the fields Sale 2013, Sale 2014, and Sale 2015 each have a summarization icon (a Greek sigma), indicating that the total for each column will be displayed if used in a report. The CountryRegion field shows a small globe, indicating that this field contains geographical data, and it will be used to draw data on maps.

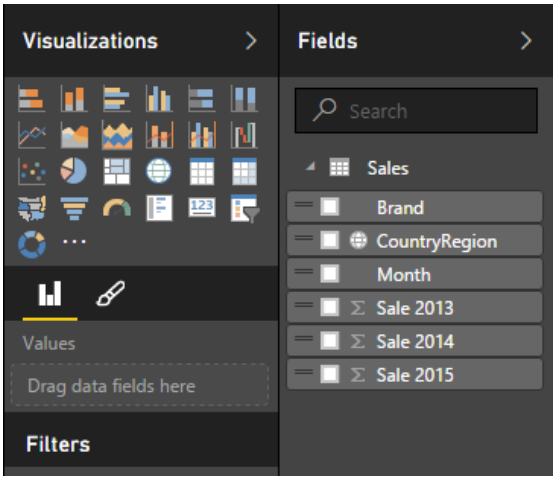
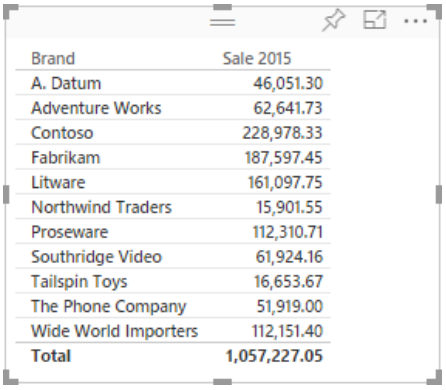


Figure 1-15: Many columns in the Fields list display a small icon. The icon indicates the default aggregation it uses.

To create a report, select the fields that you want to appear in the report. For example, referring back to Figure 1-14, in the Fields pane, David clicks Brand and then Sale 2015. Because Brand has no summarization icon, it is used to slice data, whereas Sale 2015, which displays a sigma, will present the sum for that column, generating the report shown in Figure 1-16.



Brand	Sale 2015
A. Datum	46,051.30
Adventure Works	62,641.73
Contoso	228,978.33
Fabrikam	187,597.45
Litware	161,097.75
Northwind Traders	15,901.55
Proseware	112,310.71
Southridge Video	61,924.16
Tailspin Toys	16,653.67
The Phone Company	51,919.00
Wide World Importers	112,151.40
Total	**1,057,227.05**

Figure 1-16: A first visualization based on Brand and Sale 2015.

What David just created is the default visualization; that is, it's a grid with the brands and the sum of Sale 2015 on the rows, showing raw numbers. Numbers are very interesting, but they do not give a clear idea of the relationship among them. In fact, at first glance it's not evident which brand is the most important one, which ones are the smallest, and what the relative importance of the numbers is. Charts, on the other hand, can give viewers a much quicker understanding of the data.

You can modify the visualization of a tile by choosing one of the many available types of charts in the Visualizations pane. For example, you can use a column chart by first selecting the visualization and then clicking the column chart icon, which is among the many highlighted in Figure 1-17.

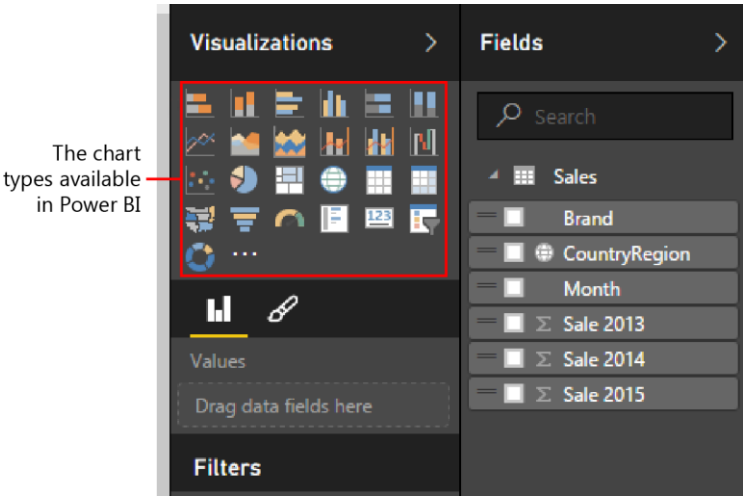


Figure 1-17: The Visualizations pane offers many different visualizations to use in your reports.

> **Note** If you click a visualization type but do not have a specific tile selected, Power BI inserts a new, empty visualization. If this happens to you, do not worry: just select the empty chart and delete it by pressing the Delete key. Then, select the tile that you want to change and try again.

As Figure 1-18 so clearly demonstrates, the same numbers—Sale 2015 by Brand—shown in a column chart are much easier to understand.

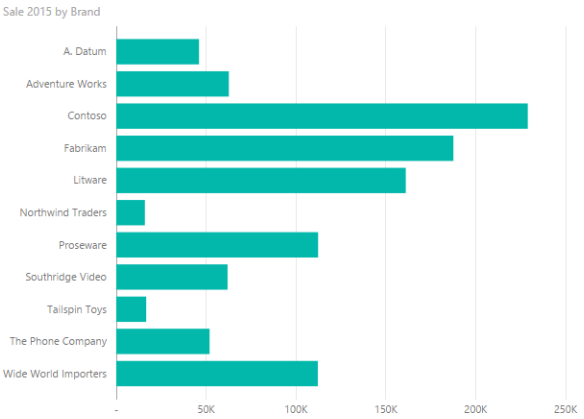


Figure 1-18: With the correct visualization, numbers are much more meaningful.

> **Note** Before proceeding further, feel free to experiment by using different visualizations for the same data. As you will discover, each visualization offers a different insight from the same numbers. With Power BI you can use different visualizations to find the best way to tell a story about your data, using the same numbers.

So far, you've learned how to create an individual chart. But one chart alone is not yet a full report. If you click an empty area of the central canvas and repeat the aforementioned procedure, but, adding the CountryRegion and Sale 2015 fields, you will generate a new tile, this time displaying a map with sales in the three countries/regions contained in the demonstration dataset, as shown in Figure 1-19.

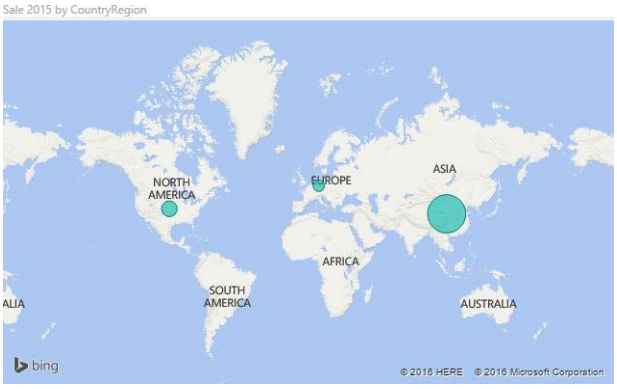


Figure 1-19: A map of the world showing sales in different countries/regions, highlighted as bubbles.

Now, maps are powerful charting tools, but, as Figure 1-19 demonstrates, by having only three values they look dispersive. There are too many details in the map, whereas the goal is to show only the relative size of three areas. In this case, a column chart does this job well. You can transform the map into a column chart and then move the two visualizations so that they look like those shown in Figure 1-20.

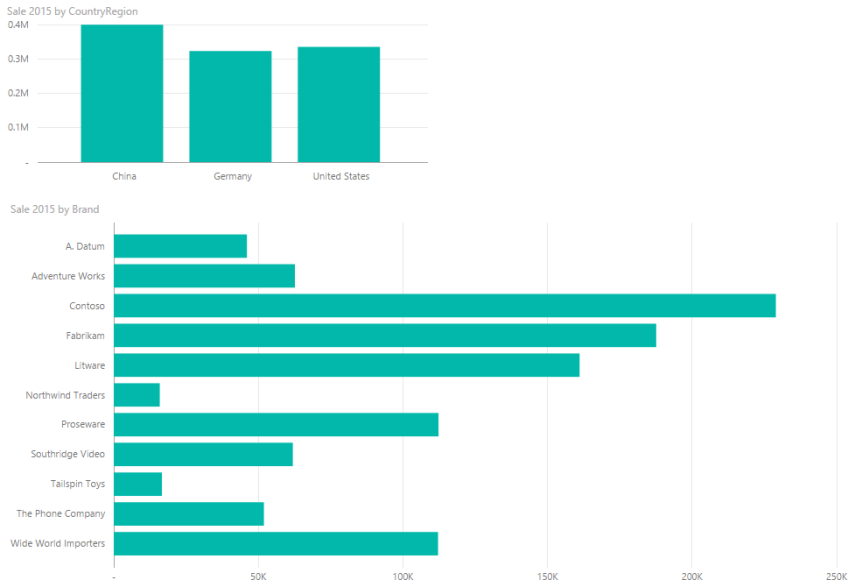


Figure 1-20: A report can contain multiple visualizations.

As you have seen, a report is a collection of visualizations organized in such a way as to communicate insights about the data. In Figure 1-20, a reader has an immediate feeling that sales in China, Germany, and the United States are nearly the same. Also it is clearly evident that there are only a few brands that make up most of the sales (Contoso, Fabrikam, and Litware), whereas others (Northwind Traders and Tailspin Toys) produce only a relatively tiny amount of sales.

Introducing Visual Interactions

This feature is very similar to what David could have achieved by using Excel and a couple of pivot tables on top of the table containing sales, yet there are some important differences between a report created in Excel and the same report done by using Power BI. We will look at those as you proceed through the book, but, for the moment, let's look at the interactive nature of Power BI reports.

In the top chart from Figure 1-20, click the column for Germany. As soon as you click an element within the chart, the entire report is filtered showing the contribution of Germany to sales of different brands, by means of coloring with two shades the Sale 2015 By Brand visualization, as depicted in Figure 1-21.

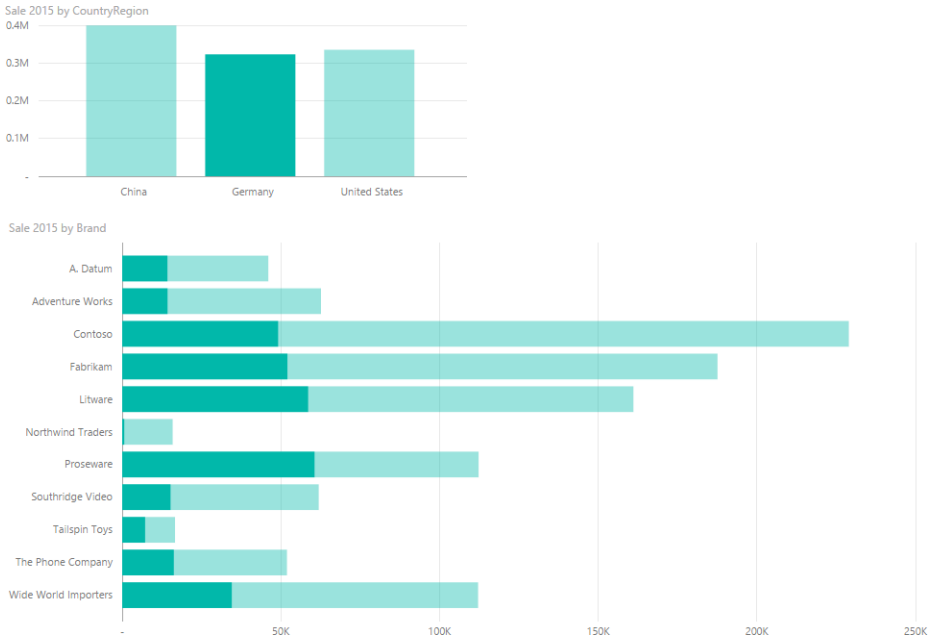


Figure 1-21: Clicking on one column in the column chart filters the bar chart, highlighting the contribution against the total.

By doing this simple operation, David notes that sales of Northwind Traders in Germany are tiny when compared with China and the United States. Clearly, that brand is not popular in Germany, and David is curious to see whether it is sold in equal volumes in China and United States or whether one of those countries/regions has much more sales than the other one.

To perform this analysis, he clicks the bar for Northwind Traders. By doing so, the filter will move from the country/region to the brand and, as it happened before, the country/region chart will highlight the contribution of Northwind Traders to the total sales, as shown in Figure 1-22.

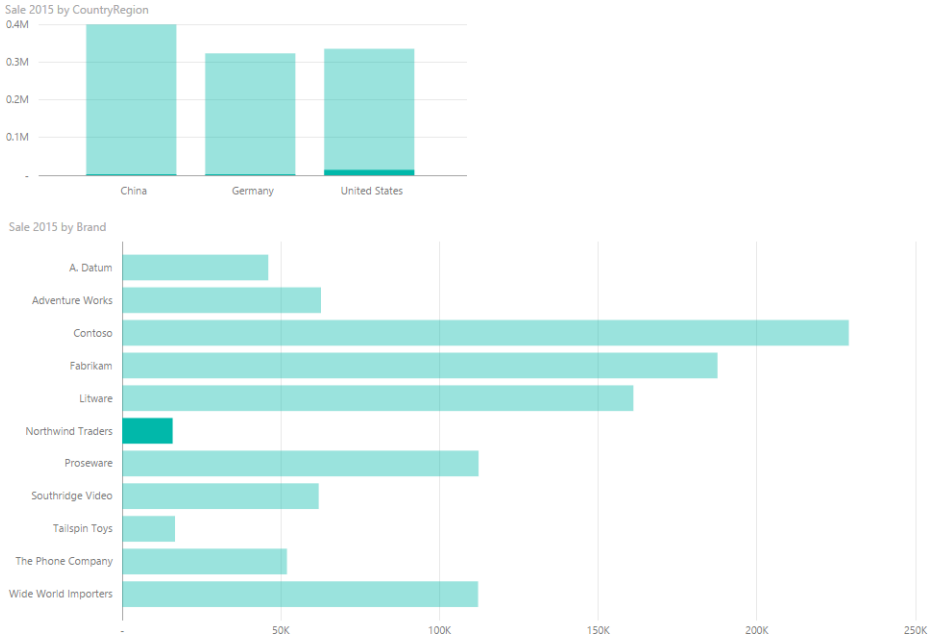


Figure 1-22: Filtering one brand shows the contribution of the brand against the total of sales by country/region.

The chart with sales by country/region already shows that a majority of sales are in United States, but because David is analyzing a very small brand, the chart is not clear in terms of relative importance of sales in different countries/regions.

Before we move on, we now need to be a bit more accurate in describing what we are seeing. Any chart produces graphical visualizations of the underlying numbers. Any of those visualizations can behave as a filter, and such a filter is activated by simply clicking the chart. So far, you have seen that a filter—when applied to other charts—highlights the relative contribution of the filtered item against the grand total by using two colors. This behavior is known as *visual interaction*, and it is extremely interesting. Yet, there are scenarios, like the one David is experimenting with, for which it would be better to compare the differences between countries/regions more than the overall contribution of a brand against the other brands.

You can configure visual interactions in a highly precise way. Namely, you can configure how the filtering on a chart behaves with respect to all of the other ones. The scenario we are looking at—with only two charts—is perfect for experimenting because it is very simple. To configure visual interactions, on the top menu bar of the report, click the Visual Interactions button, which you can see highlighted on the right in Figure 1-23.

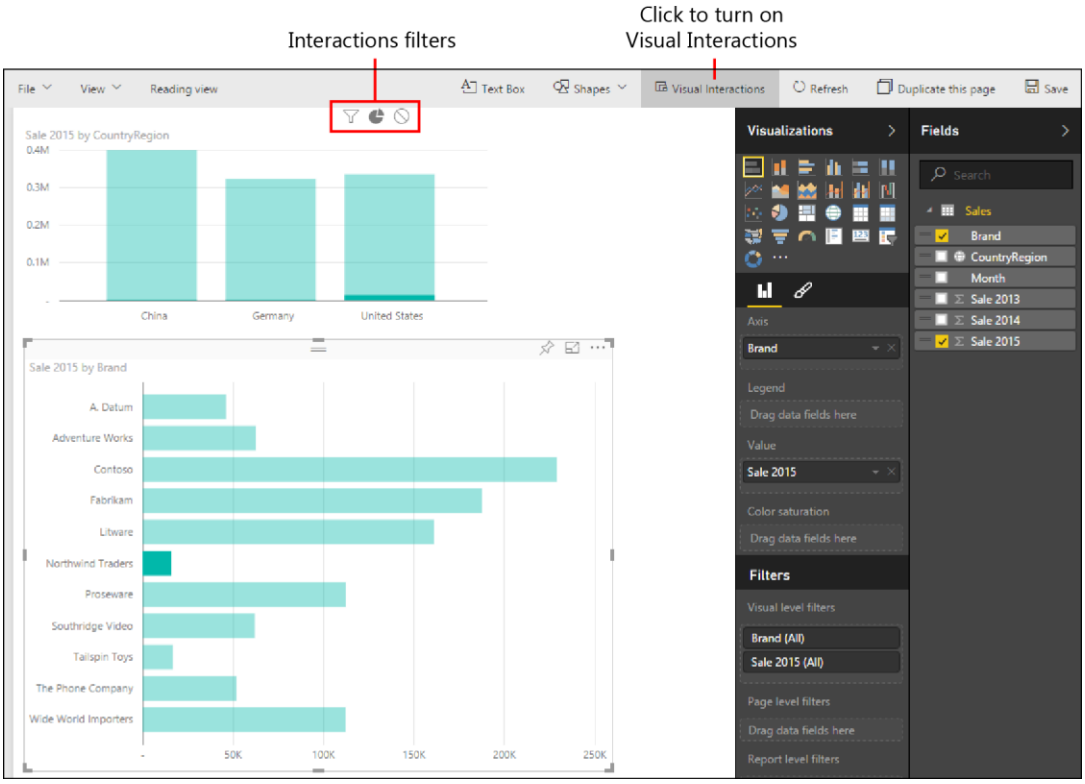


Figure 1-23: When you turn on visual interactions, you can configure how a chart interacts with other charts.

When you turn on visual interactions, each chart shows a different set of icons. The one you select (in this example, Sale 2015 By Brand) shows the standard selection icon, whereas all of the others (Sale 2015 By CountryRegion) show the three different kinds of interactions you can choose:

- The first is the filtering interaction (the funnel icon). When you click this, filtering the selected chart will place the very same filter on the destination chart. In such a case, you will not see the contribution of the selection to the total. Instead, you will see only the selection in the chart, excluding the values (and corresponding areas) related to unselected items.

- The second interaction is a pie chart (the pie icon); that is, the relative contribution. This is the default filtering behavior, where the filtering on one chart shows, on the destination chart, the relative contribution of the selection against the total.
- The third is the no filtering interaction. When this is the selected behavior, filtering the selected chart has no effect on the target chart.

> **Note** Visual interactions are much easier to use than to explain in a book that, by nature, contains static figures. If you are still not clear on the behavior of filtering, try it yourself; you will understand it in a much easier way.

For example, you can select the filtering interaction from the sales by brand to the sales by country/region. By doing so, when you select Northwind Traders, the resulting report will show a different result, as shown in Figure 1-24.

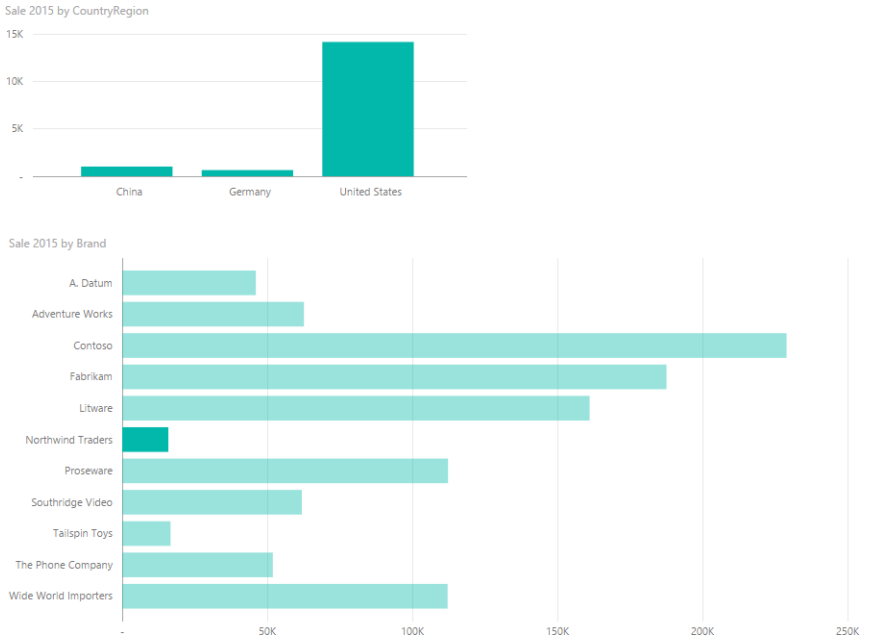


Figure 1-24: By using the filtering behavior, the relative size of the bars in the Sale 2015 By CountryRegion chart is more meaningful.

Now, when you browse the report, you can quickly click a brand and see which country/region sold more than the others. Because of automatic determination of the scale, the insights are much clearer.

> **Note** You can configure the filtering behavior for any two pairs of visualizations. To perform this, you first select the source (that is, the tile from which you want to filter) and then choose the proper action on the destination visualization. Needless to say, you need to pay attention because mixing different filtering behaviors on the same page can result in some complexity and confusion when using the report.

Decorating the report

In the previous sections, David performed some analysis on the data, and now he thinks that his first report, although simple, contains some findings that are worth sharing. He can obviously make a screenshot and attach it to an email with some description, but Power BI offers some tools that make it possible for him to annotate a report with remarks.

David can add text to the report and decorate it with shapes. For example, he can add a colored arrow to Northwind Traders and a text box with some remarks about what he found. When David does this, the report is easier to read, as demonstrated in Figure 1-25.

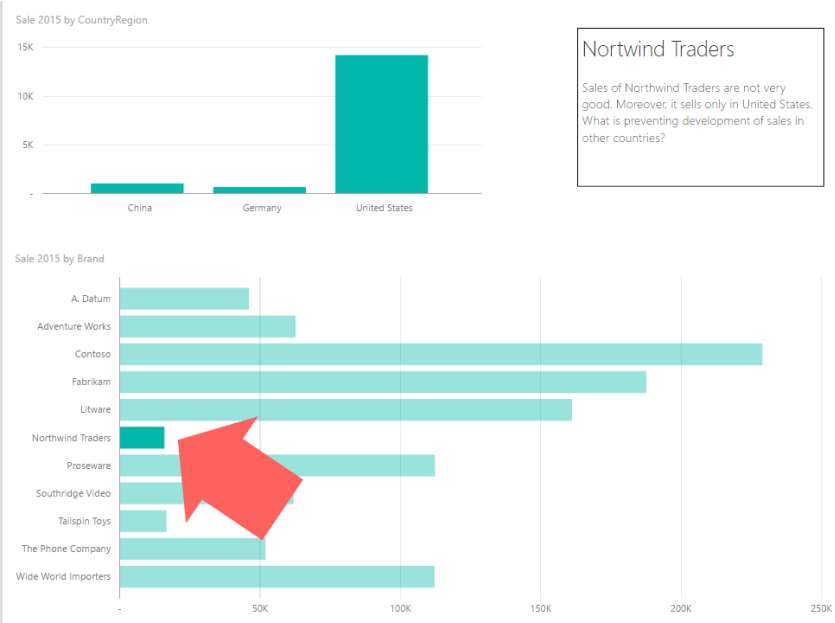


Figure 1-25: Decorating the report makes it easier for the reader to immediately understand the insights.

To add the text, above the central pane, David clicks Text Box. When the text box appears, he types and formats his text. To add the arrow, again above the central pane, David clicks Shapes and then Arrow. He then moves the arrow into position and resizes it.

When you add the text box or the arrow, you need to set some properties for these objects. In fact, the appearance of each object (either a decoration or a full chart) in a Power BI report is controlled by a set of properties that you can access by clicking the object in the central pane. The properties that you can set appear in the Visualizations pane, as shown in Figure 1-26.

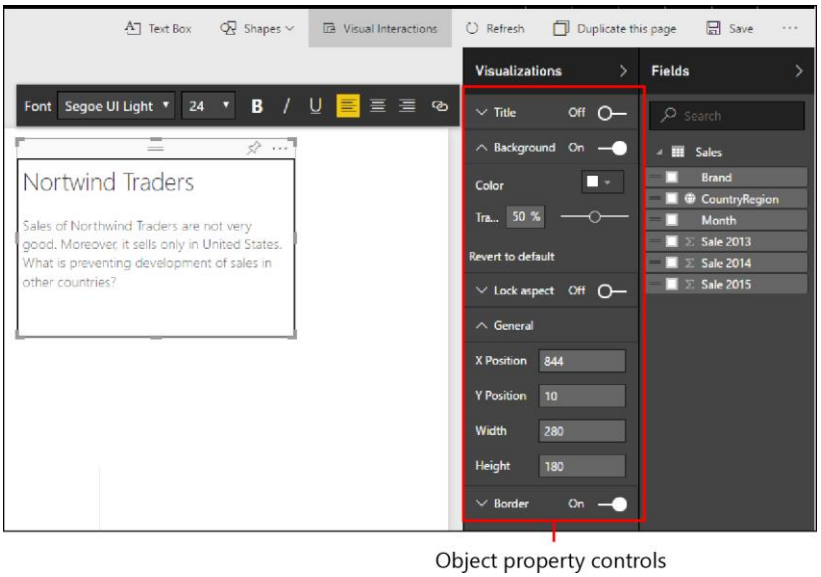


Figure 1-26: Each visualization has a set of properties that you can adjust to customize it.

For example, to rotate the arrow, David selects it and then, in the Format Shape pane on the right, he clicks Rotation and drags the slider. Similarly, he also changes the fill color.

Finally, keep in mind that visual filters on reports (that is, filters that you set by clicking a chart item) are not saved as part of the report. Thus, when you look at the report again, the arrow is useful to tell you where to apply the filter to see the data. Later in this chapter, you will learn how to place a permanent filter on a report.

Saving the report

At this point, David can save his report so that he can continue working on it later. To save a report, go to File, click Save (see Figure 1-27), and then provide a name for the report. Here, David saves his report with the name Northwind Traders.

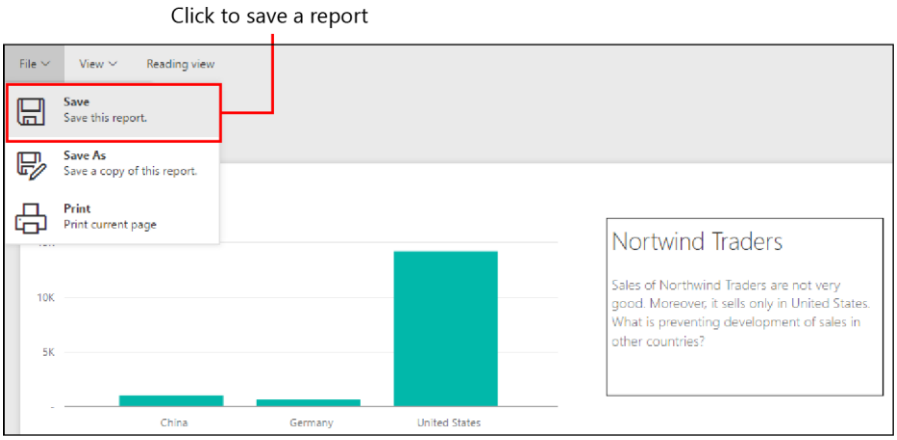


Figure 1-27: When you finish editing a report, saving it is always a good idea.

After you save the report, it appears in the My Workspace pane, in the Reports section. You can now access it any time you sign in to Power BI.

When you select a saved report, it opens and remains in read-only mode until you explicitly activate it for editing by clicking the Edit Report button highlighted in Figure 1-28.

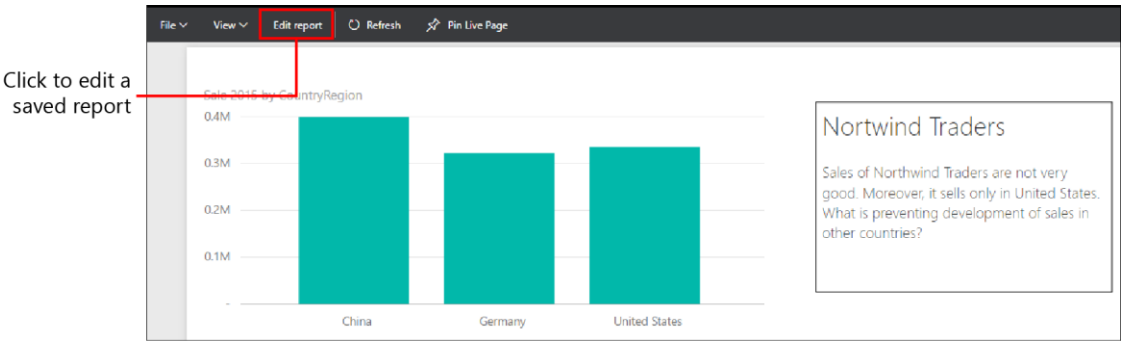


Figure 1-28: You need to click the Edit Report button to bring a saved report in edit mode.

This behavior is useful to avoid unintentional editing of the report. A saved report can be easily viewed, and it always reflects the latest data. If you need further filtering or you want to perform a different analysis on the same report, you need to turn on edit mode.

Pinning a report

When you open a report in read-only mode, the menu bar at the top of the screen offers you several actions: you can choose to save a copy of the report under another name, edit or print it, and apply different visualizations. All of these operations are seamless and need no further explanation.

But, one of the menu bar items is worth a few moments of our attention: Pin Live Page (see Figure 1-29).

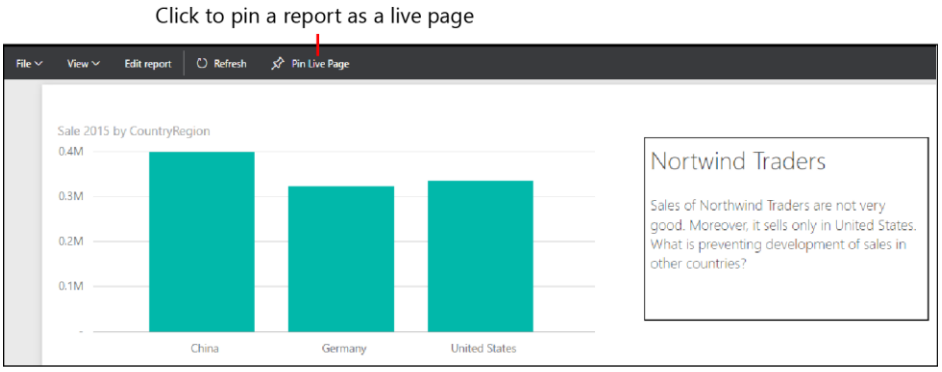


Figure 1-29: Visualizations aren't the only things that you can you can pin to the dashboard: You can pin reports, too.

What is the difference between pinning a visualization and pinning a full report? When you pin a visualization, Power BI saves it as it is, but the visualization is disconnected from any others in the same dashboard. Thus, any visualization in a dashboard does not include the visual interactions of other visualizations. This is usually good, because a dashboard is not intended for interaction. If you need interaction or further analysis, you can always click a visualization from the dashboard to open the source report.

Nevertheless, sometimes you want to keep visual interactions between some components of your dashboard. If this is your goal, you need to build the report and pin it in its entirety as a live page. Visualizations belonging to the same live page will maintain the behavior of visual interactions, albeit they will be limited to the visuals in the report. In other words, visualizations belonging to the same report can interact among themselves, whereas filtering them has no effects on other visualizations in the same dashboard. For example, after David adds his Northwind Traders report to his dashboard, the two charts are mutually interactive, but they don't affect the other visualizations in the dashboard, and those other visualizations don't affect the two brought over in the report.

Refreshing the budget workbook

So far, David has learned the basics of Power BI and ended up with some useful findings that he will want to share with the country/region managers. Nevertheless, before continuing, David is worried about how he will refresh his data when new figures become available. In fact, you might remember that he began building the budget in October. Thus, new data will be arriving over time for sales, and the country/region managers will provide new forecasts that David will need to add to his workbook. How will he upload new data to the service to refresh the existing data?

If fact, there are many ways by which you can refresh data in Power BI.

David receives figures from the country/region managers in a very simple form: They each send him a workbook with forecasts based on the brand, with no monthly details. Figure 1-30 shows what the forecasts look like for China.

CountryRegion	Brand	Budget
China	A. Datum	5,000.00
China	Adventure Works	36,500.00
China	Contoso	94,500.00
China	Fabrikam	90,800.00
China	Litware	65,000.00
China	Northwind Traders	12,000.00
China	Proseware	20,000.00
China	Southridge Video	22,500.00
China	Tailspin Toys	8,000.00
China	The Phone Company	30,000.00
China	Wide World Importers	45,000.00

Figure 1-30: An example of forecasts received from the China manager.

Because the forecasts are at the year level, but David set them up at the month level, he opts for a simple solution: divide the yearly sales by 12 and copy the result to his own workbook in a new column called Budget.

The workbook with the new Budget column now looks like Figure 1-31. Looking at the numbers, it is clear that David should have used a different allocation, because the numbers do not reflect the seasonal nature of sales and, more important, they are not correct. We will fix this and use a better technique later in the book. Right now let's focus on Power BI. Remember, this is not a book about budgeting techniques.

CountryRegion	Brand	Month	Sale 2013	Sale 2014	Sale 2015	Budget
China	A. Datum	January	3,234.00		1,935.00	416.67
China	A. Datum	February	6,270.00	7,059.00		416.67
China	A. Datum	March	4,352.00			416.67
China	A. Datum	April	3,814.00			416.67
China	A. Datum	May	6,234.00			416.67
China	A. Datum	June	5,571.00	3,216.00		416.67
China	A. Datum	July	7,424.00			416.67
China	A. Datum	August			800.00	416.67
China	A. Datum	September	1,254.00	1,617.00	396.00	416.67
China	A. Datum	October	1,881.00	3,042.00	936.00	416.67
China	A. Datum	November		3,653.00		416.67
China	A. Datum	December	6,135.00	2,810.00		416.67
China	Adventure Works	January	12,418.26	5,735.48	1,559.87	3,041.67
China	Adventure Works	February	31,770.26		2,937.90	3,041.67
China	Adventure Works	March	3,689.85	5,489.23	11,163.94	3,041.67

Figure 1-31: The 2015 Sales workbook now contains monthly sales in the last column, named Budget.

Now, David faces this scenario: the workbook on his laptop has different numbers and a different structure (the Budget column is new), whereas the workbook he uploaded into Power BI still retains the old values and model.

The simplest way that comes to mind to refresh the workbook is to upload it again to the Power BI service. David follows the same upload procedure he used the first time, but when it is about to complete the upload, Power BI issues the warning shown in Figure 1-32.

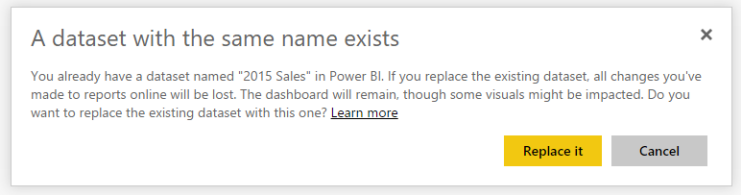


Figure 1-32: If you upload the same workbook twice, Power BI issues a warning about possible data loss.

The error message is not totally clear. It states that David is going to lose changes to reports online, but he did not make any changes. He created several reports, without modifying them. But, David is nothing if not a brave and cavalier sort, so he clicks Replace It to see what happens.

> **Note** Even if it might be obvious, it is worth remembering that Power BI is an online service. When it comes to datasets, you cannot use the standard technique of "making a copy of the workbook before replacing it" that you probably use on your PC.

After uploading the file, the old workbook is replaced with the new one and all the reports and the dashboard look identical. David did not lose anything. In reality, the warning pertains to Power View reports that might have been automatically created within the Excel data model, a feature that you did not learn yet. So, David breathes a sigh of relief and pushes onward.

> **Note** Be aware that there might be a delay of several minutes from when you upload a new version of a dataset until the new columns and tables appear in Power BI. The exact timing depends on whether there is a recent release of Power BI and is subject to change in the future. If, for any reason, you do not see updated information after a new upload, just wait a few minutes and try it again; Power BI is being refreshed and nothing is going wrong.

Now, the Power BI model contains the new Budget column, which David can use to build more interesting reports. It is worth noting that a report can contain multiple pages. Thus, he can add different visualizations to the 2015 Sales report. For example, he created the report page depicted in Figure 1-33.

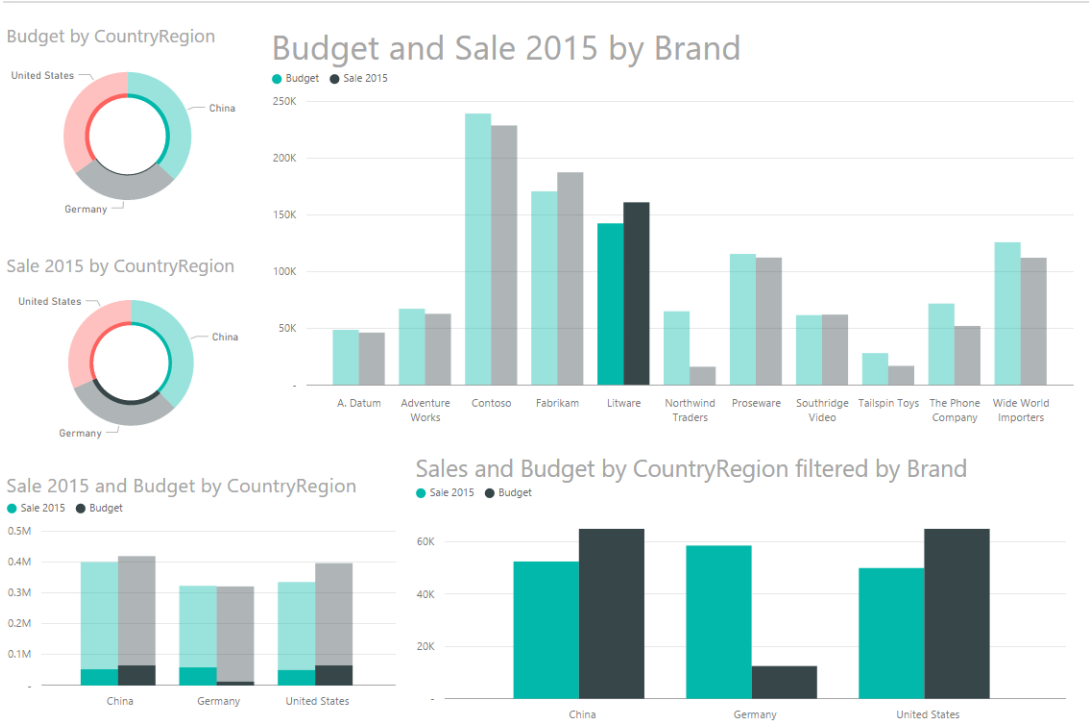


Figure 1-33: With the Budget in the model, reports are richer and provide better information.

Observe that the two visualizations at the bottom use a different visual interaction method. The chart on the left shows the contribution of a brand to the overall sales and budget, whereas the one on the right is more useful to perform a comparison of budget and sales in different countries/regions. Both are useful and provide different insights. The technique of adding multiple copies of the same visualization with different visual interactions is common, and we encourage you to learn to use it.

You might have noticed that to make it more evident, David uses different titles for the two visualizations (and different font sizes, too). You can manage these visualization details by using the brush icon highlighted in Figure 1-34, which shows the many options to configure a visual. In the example, we used a custom title and changed the font size.

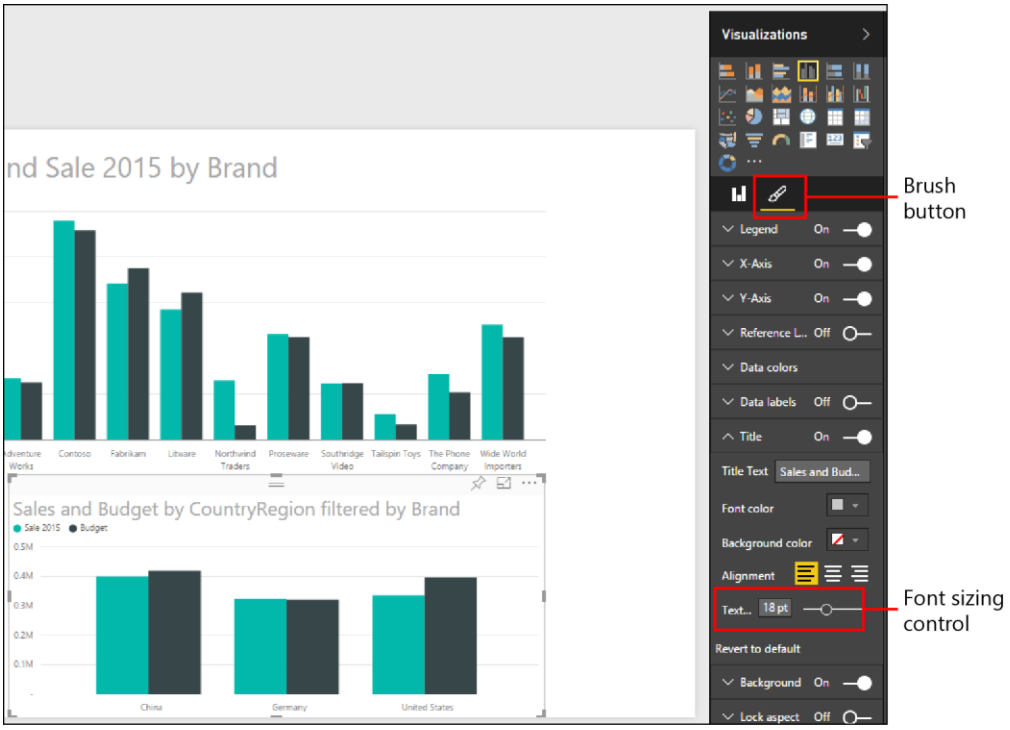


Figure 1-34: You can configure many aspects of a visualization by using the formatting options.

You will learn many more details about visualization formatting as the book progresses. However, for an introductory chapter, this is enough. It's time now to draw some conclusions.

Filtering a report

You already learned about the visual interactions feature, which makes filtering a report a breeze. Visual interactions are useful, but they come with some limitations:

- The filter is not saved as part of the report. Whenever you open a report, you can begin to play with visual filters but there is no way to store the filter in the saved report.
- The filter is always visible. Sometimes you want a filter for the entire report, but you do not want any visual indication of the filter being applied. In other words, you want something like a hidden filter working in the background on the full page or report.

Power BI offers you a different way of filtering data. They are the standard filters (as opposed to visual filters), and they can be applied to three different layers:

- **Visual-level filters** Visual-level filters work on only an individual visualization, reducing the amount of data that the visualization can see. Moreover, visual-level filters can filter both data and calculations.
- **Page-level filters** Page-level filters work at the report-page level. Different pages in the same report can have different page-level filters.

- **Report-level filters** A report-level filter works on the entire report, filtering all pages and visualizations included in the report.

You can set all of the filters in the Filters section in the Visualizations pane. Figure 1-35 illustrates that for David's report, there are three kinds of filters.

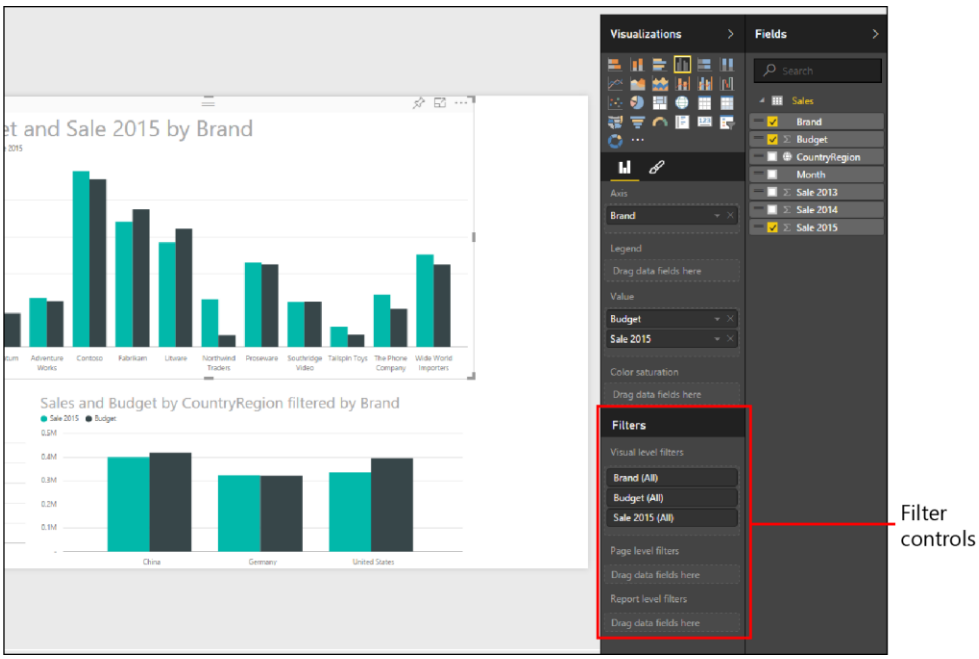


Figure 1-35: You can configure filters in the same place, in the Filters section of the Visualizations pane.

You can drag columns from the Fields in any filter and, when there, you can click them to apply a filter, by simply selecting some values from the list.

For example, Figure 1-36 shows the result if you add a page-level filter to the report, selecting only China and Germany.

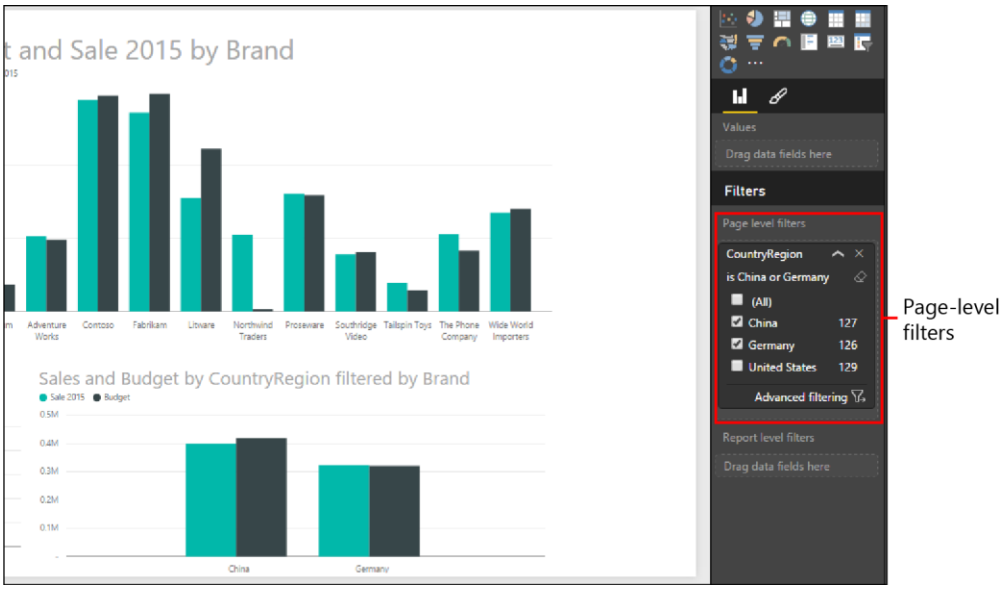


Figure 1-36: The same report as the report presented in Figure 1-33, this time filtered using only China and Germany.

Filters at the report and page level behave the same way. Filters on a visualization, on the other hand, have an additional feature: they can filter both data (as was the case for the country/region) or the metric associated with the chart.

For example, you can filter the upper-right chart to include only values for which the budget is greater than 50,000. Figure 1-37 presents the result.

Figure 1-37: A visual-level filter can filter the measure used to draw the chart.

Notice in Figure 1-37 that the number of brands is much less than that of Figure 1-36. This is because the latter report shows only brands that have a Sale 2015 measure greater than 50,000.

All of these filters are saved as part of the report, and they are not shown in any visual way. For this reason, if you prepare a report that, for example, filters only 2015, it is always useful to add a description of the filter as part of the report title—"Sales in 2015" instead of "Sales."

Conclusions

After this first tour in Power BI, it's now time to take a breath and describe what we've learned so far.

- Power BI is a cloud service that provides tools to perform analysis of data and gain insights from your numbers.
- To build a dashboard, you need a dataset, a report, and, finally, the dashboard. The dataset is the source of data, reports are useful to create visualizations that might be connected through visual interactions, and a dashboard is a collection of visualizations and/or reports.
- You can create visualizations by using natural-language queries, Quick Insights, or full reports.
- You can decorate a report by using text boxes, shapes, and pictures.
- Visualizations in a dashboard are not connected through visual interactions, which work only among visualizations in a report. If needed, you can pin a report as a live page in a dashboard to maintain the interaction capability.

- You can load data in Power BI from many different sources. So far, David has used only an Excel workbook, but there are many other sources that he will learn to use before becoming a Power BI expert (and that you will learn about later in this book).
- You can refresh the content of your workbook by uploading a new version of it. But, as Chapter 2 describes, there are better ways to refresh data.
- You can apply filters by using visual filters, which produce highly interactive reports, or you can use static filters, which you can apply at the visual level, page level, and report level. Static filters are saved as part of the report, whereas visual filters are not.

Sharing the dashboard

In Chapter 1, David, our manager of budgeting at Contoso, created his first dashboard with analysis of sales. But that dashboard is only the starting point for creating the budget for 2016. David will involve his colleagues in this process, so the first thing he needs to do is share the work he has done thus far with these colleagues. After that, he must collect feedback and numbers from other managers in order to complete the entire budget. Also, he needs to choose how to share the resulting data and reports between members of the team.

In this chapter, you will see how David can use the features of Microsoft Power BI, along with other services, to achieve his goals.

Inviting a user to see a dashboard

David wants to share the dashboard he created with his colleague Wendy, the country/region manager for Germany. To do that, he opens the dashboard in its own Power BI account and then clicks the Share button located in the dashboard's upper-right corner (see Figure 2-1).

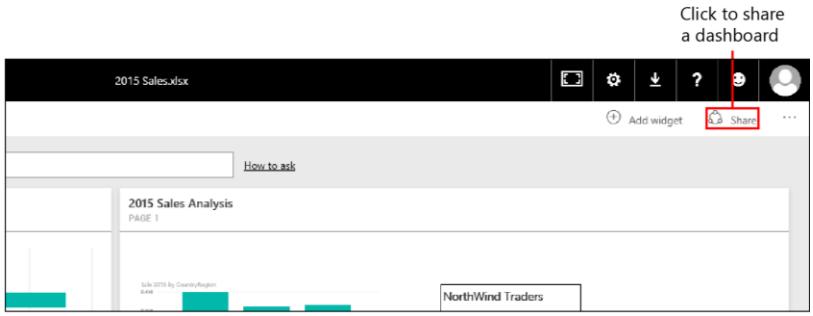


Figure 2-1: To share a dashboard, click the Share button in its upper-right corner.

This opens the Share Dashboard dialog box in which David can send Wendy an invitation. Before we continue along with David, let's take a closer look at this dialog box. The Share Dashboard dialog box has two tabs: Invite and Shared With. The Invite tab (see Figure 2-2) is where you provide the email address (or addresses) of the people with whom you want to share a dashboard. You also can include an optional message that you would like your invitees to receive from the Power BI service. At the bottom of the tab are the Allow Recipients To Share Your Dashboard check box and the Send Email Notification To Recipients check box, which are self-explanatory. If you decide to send an email as a notification to recipients, Power BI will automatically include a link to open the dashboard in the email that is sent.

Notice in Figure 2-2 that when David types Wendy's name in the name box, Power BI suggests her email address. Because Wendy is a coworker in the same organization as David, Power BI is able to offer these suggestions (more details on this later). However, you also can type an email address in this box if the email address of the person with whom you are sharing does not have the same domain name as your address (for more details about this, read the section "Inviting users outside your organization," later in this chapter). You can add more than one person if required.

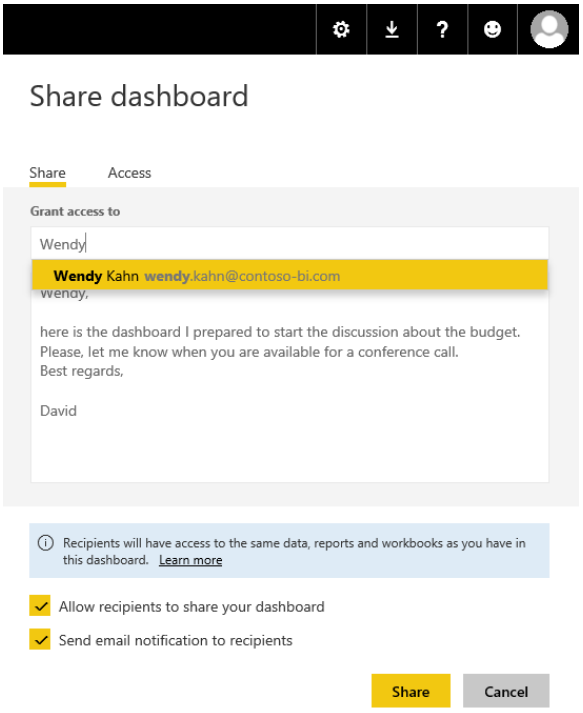


Figure 2-2: In the Share Dashboard dialog box, you specify the list of people invited and, optionally, include a message for them.

If you clear the Send Email Notification To Recipients check box (see Figure 2-3), you will need to go on the Shared With tab to copy the link to your dashboard that you will then send to recipients after you click the Share button. The email addresses you provide will give those accounts access to your dashboard, but those individuals will not receive an email notification.

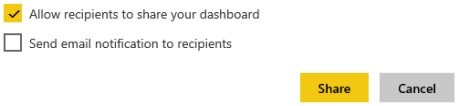


Figure 2-3: You can share a dashboard by copying a link instead of sending an email from the Power BI service.

On the Access tab (see Figure 2-4), the Dashboard Link box contains the URL for the dashboard. If you choose to not send a message via the Power BI service to your recipients or you simply prefer to use your own email account to do so, you will need to copy this URL and send it to your invitees. Using your own account can be helpful when you want to ensure that your recipients recognize that the incoming email is from you, as opposed to seeing "no-reply@powerbi.com," which they, or their email client, might filter out as spam. Also on the Access tab is a list of users with whom you have shared the dashboard, along with their assigned privileges.

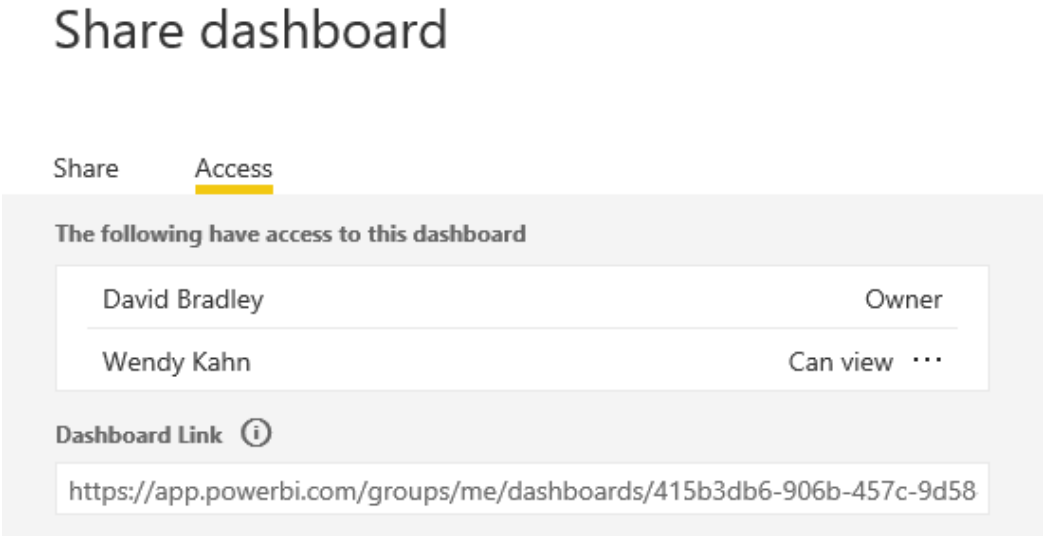


Figure 2-4: The Access tab shows the users who have access to the dashboard and provides a link to share it.

The email message that Wendy receives looks similar to that shown in Figure 2-5. If she has not previously used Power BI, when she clicks the link to open 2015 Sales.xlsx, she will be directed to the Power BI website where she will need to register with the service, just as David had to do when he first used Power BI (see Chapter 1 for a refresher on getting started with Power BI). If she is already enrolled as a Power BI user, she will go straight to the dashboard that David is sharing with her.

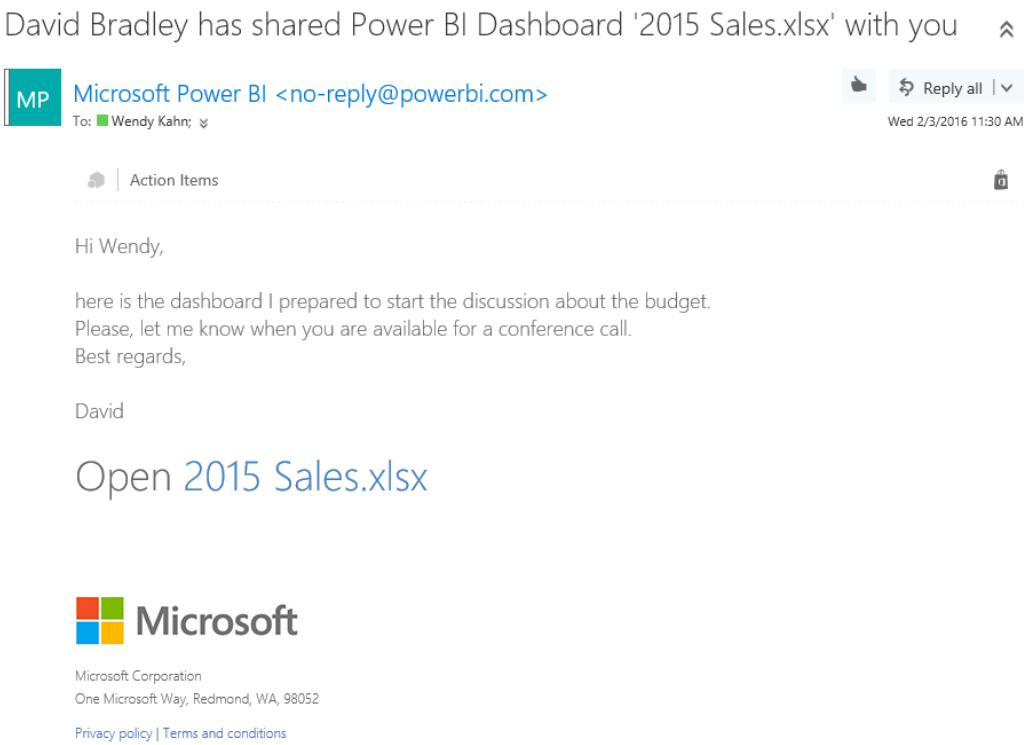


Figure 2-5: Power BI sends an email message to users invited to share a dashboard.

Now, Wendy is looking at the same dashboard as David, including the reports underlying the visualizations pinned to the dashboard. However, Wendy cannot modify either the dashboard layout itself or any of the single reports; at this point, she has read-only permission. For David to make it possible for other users to edit his dashboards and reports, he needs to create a group workspace, which we will cover a bit later in the chapter.

After David invites Wendy, she can open the dashboard in its own Power BI session. Dashboards shared by another user appear on a guest user's Workspace pane with a "shared" icon adjacent to the dashboard name, as shown in Figure 2-6. This indicates that the dashboard is read-only. So in this case, Wendy cannot change or modify the content of 2015 Sales.xlsx, but she can interact with reports pinned to the dashboard and can open the reports underlying each visualization by simply clicking a visualization. Even if such reports are not listed in the workspace, Wendy can open them through the shared dashboard, but she cannot modify their content (the Edit Report feature is turned off in these cases).

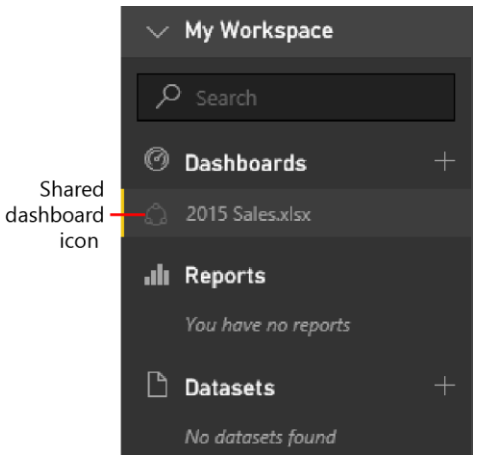


Figure 2-6: Shared dashboards display a "shared" icon before the name in the list of dashboards.

Sharing via content packs

An additional technique to share reports and dashboards within an organization is through a *content pack*, which is a set of datasets, reports, and dashboards that a user can copy within his personal workspace. David might consider using this feature to deploy a report to other users, but he does not use this system in our scenario because it is a technique that is better suited to distribute a set of predefined reports and dashboards that other users can customize in their personal copy. Content packs are not designed to share reports between users in an active way, as David needs at the moment. Chapter 5 shows you how to create and consume content packs from the public gallery and within an organization.

Inviting users outside your organization

Thus far, David has invited Wendy to view his dashboard; she works in the same company and has an email with the same domain (@contoso-bi.com). But what happens when David wants to invite someone who is not part of the same company? Answering this question requires some explanation.

Although Power BI is designed for you to share a dashboard with users who are within the same organization, you can also share dashboards with people from other organizations. The way Power BI identifies "an organization" can be described as follows:

- Every user requires an email address within the domain of the company.
- Power BI does not accept generic email domains such as hotmail.com, gmail.com, and so on. Your company needs a unique domain name, and all of the users must have an email address within that domain. All of the users having an email within the same domain are considered part of the same organization.
- If you use Microsoft Office 365 and/or Microsoft Azure Active Directory, you might have different domains belonging to the same organization. This is the only case for which users having email with a different domain name belong to the same organization for Power BI.

Note If you are not sure whether your organization already uses Office 365 and Azure Active Directory, ask your IT administrator, and if he would like to read more technical details about authentication in Power BI, refer him to the following documents:

https://powerbi.microsoft.com/documentation/powerbi-admin-power-bi-security and http://go.microsoft.com/fwlink/?LinkId=619090 (which downloads the Power BI Security white paper)

On the surface, that seems rather restrictive, but in reality, you also can share a dashboard with users in other organizations, using the same method as that described in the previous section. However, when you specify an email address with a domain other than that of your organization, you will see a message similar to one shown in Figure 2-7, which David receives when he tries to share a dashboard with a vendor.

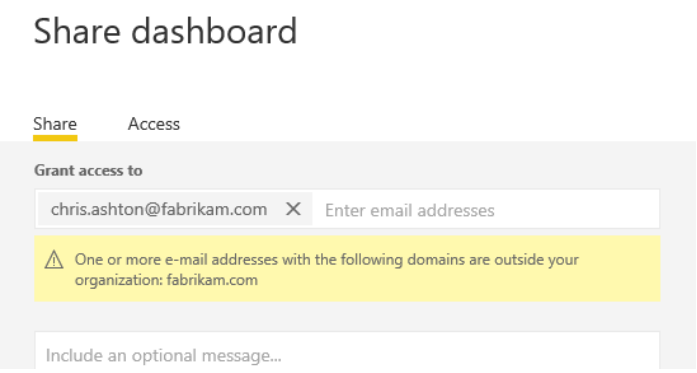


Figure 2-7: The message that displays when you try to share a dashboard with someone outside of your organization.

It is important to understand the difference between a user within your organization (internal users) and outside of it (external users):

- **Internal users** You can invite internal users to share a dashboard by email or by sending them the URL of the dashboard. In the latter case, users must be authorized. If a user does not have authorization, she can ask for permission when she clicks the dashboard URL.
- **External users** You can share a dashboard with external users only by inviting them by email. When an external user receives the email, she must sign in to Power BI using the same email account used in the invitation. If she never previously used Power BI, she can create a free account the first time she signs in.

Finally, you can publish a report (but not a dashboard) on the web. To do so, select the report, click the File menu, and then click Publish To Web, as depicted in Figure 2-8. In the Embed In A Public Website (Preview) dialog box, click Create Embed Code. This creates a public webpage that anyone can visit. Keep in mind, though, that you cannot control who can see such a report, meaning anyone who has the URL can view your data. For this reason, you should use this technique only when you want to publish information intended for public consumption; for example, a report embedded in the public website of your company.

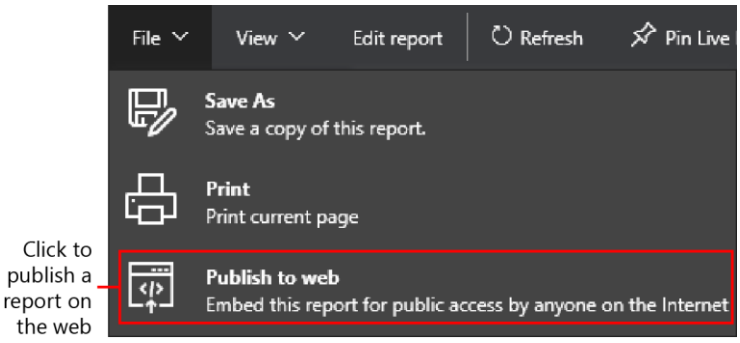


Figure 2-8: The File menu includes the Publish To Web command, which makes a report available on the Internet.

The Publish To Web feature guides you in creating a public webpage, getting a URL that you can send in an email, or the HTML code required to embed the report in a page of a website you own. For more technical information about publishing a report to the web and to get a detailed step-by-step guide, go to https://powerbi.microsoft.com/documentation/powerbi-service-publish-to-web/.

Creating a group workspace in Power BI

Let's return to David and Wendy. After David invited Wendy, he realizes that he will need to repeat the same share operation for every dashboard he creates. Moreover, as soon as other people become involved in the budgeting process, he will need to send them the invitation for all of the dashboards he shares with the group. Fortunately, David discovers that he can create a group of users with whom he can automatically share all of his dashboards, and also provide editing rights to certain users within that group. By creating groups of users in Power BI, you increase the level of collaboration among them.

The only caveat is that you must have a Power BI Pro license to have access to the group feature; you cannot create groups by using the free version of Power BI. However, you can try Power BI Pro for 60 days free of charge, giving you an opportunity to evaluate this feature and determine whether it is good for your company.

Assuming that you—and David's organization—opt to purchase a Power BI Pro license, to create a group, in the My Workspace pane, immediately below the list of workspaces, click the "+" button to the right of Create A Group, as demonstrated in Figure 2-9. (You might need to click My Workspace to open that pane.)

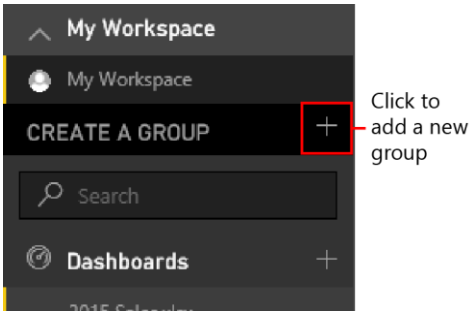


Figure 2-9: You can create a group by clicking the plus symbol (+) to the right of Create A Group.

David clicks the "+" button, which opens the Create A Group dialog box in which he creates a group named Budget 2016. This group will initially include himself as administrator and Wendy as a member. In the Privacy section, shown in Figure 2-10, you can define the privacy levels of the group.

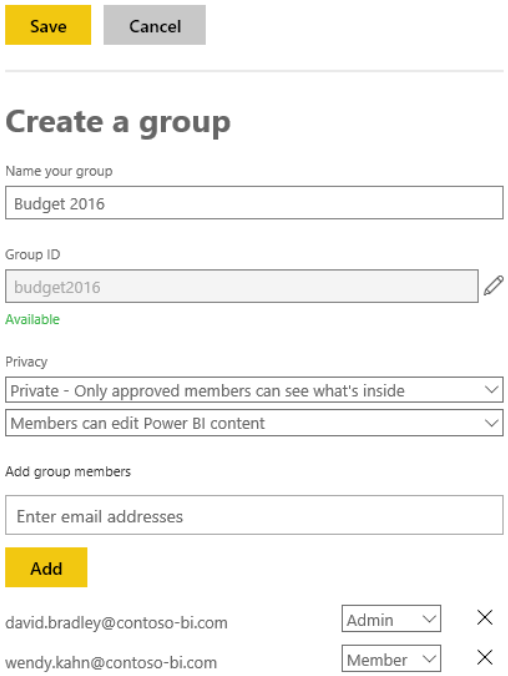


Figure 2-10: The group includes a list of members and privacy settings that group administrators can change later.

Every group has two privacy settings. The first determines whether the group is visible only to its members or also by other users within the organization who are not members of the group. Here are the possible choices:

- **Private** Only approved members can see the results of the group's activities.
- **Public** Anyone can see what the group is doing.

The second setting specifies whether all the group members can modify the contents of reports and dashboards. There are two choices:

- Members can edit Power BI content.
- Members can only view Power BI content.

If you select view-only for this second setting, only group administrators can edit dashboards.

With David having configured the group as shown in Figure 2-10, Wendy will be able to edit the content of dashboards and reports included in the group. David and Wendy will see the public group in their list of group workspaces, as illustrated in Figure 2-11.

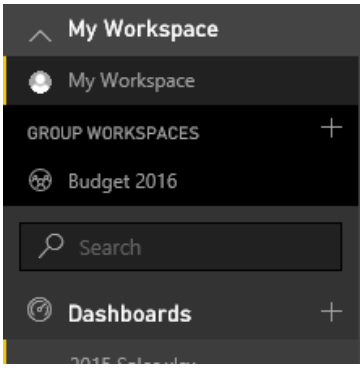


Figure 2-11: The list of group workspaces includes all the groups of which the user is a member.

Now that David has created a group, he can create reports and dashboards in the group that will be immediately visible by Wendy. However, he must import the data for these reports in the same group; he cannot move into the group what he already created in his personal workspace. Importing data and creating the reports will require some time at this point, repeating the same operation he has already done. Thus, when you know you will work with a team, it is a good idea to create the group at the outset, and save yourself a lot of redundant work later on.

Turning on sharing with Microsoft OneDrive for Business

Before moving forward, David wonders whether he will be able to share the data sources, not just the results (reports and dashboards). In particular, he wants to allow other colleagues to enter their data in Microsoft Excel files, so that he will be able to create a budget using data from workbooks modified by a number of other people.

In Chapter 1, David was forced to copy the budget data from Excel files received by country/region managers, and then he split those values by month in the table used to feed his Power BI report. Now, he wants to give other users the ability to modify the content of that Excel file directly so that he does not need to do all that work himself. For this reason, David creates a single Budget table in another worksheet of the same Excel file and copies to it the budget values by Country/Region and Brand, as depicted in Figure 2-12.

CountryRegion	Brand	Budget 2016
China	A. Datum	5000
China	Adventure Works	36500
China	Contoso	94500
China	Fabrikam	60000
China	Fabrikam	30000
China	Litware	60000
China	Litware	5500
China	Northwind Traders	9000
China	Proseware	20000
China	Southridge Video	22500
China	Tailspin Toys	8000
China	The Phone Company	30000
China	Wide World Importers	37500
Germany	A. Datum	13500
Germany	Adventure Works	15000
Germany	Contoso	50000

Figure 2-12: David's budget table now contains at least one row for each Country/Region and Brand.

David created a formula in the table used by Power BI that allocates the budget value over 12 months. At this point, David wants to share this workbook so that his colleagues will be able to directly modify the content of the budget table, and this should automatically apply the new values to reports and dashboards published in Power BI.

Using OneDrive for Business is the best way to share his source Excel workbook with other colleagues. You might already know OneDrive, which is the personal service with which you can store files in the cloud. But, even though you can share files on OneDrive, there are limitations when using it as a data source for Power BI, especially when it entails automatic data refresh. OneDrive for Business removes those limitations and provides more control, as well. Moreover, OneDrive for Business is integrated with Office 365 and directly supports groups, making it easier to share documents across your organization. To access OneDrive for Business, in the upper-left corner of the Power BI site, click the yellow button with the nine small squares in it (see Figure 2-13), and then click the OneDrive tile.

Note OneDrive for Business is a feature included in Office 365. If you do not have an Office 365 plan, you can subscribe to OneDrive for Business separately, without activating Office 365. If you are interested in using this feature, contact your IT administrator to determine which licensing option better fits your requirements.

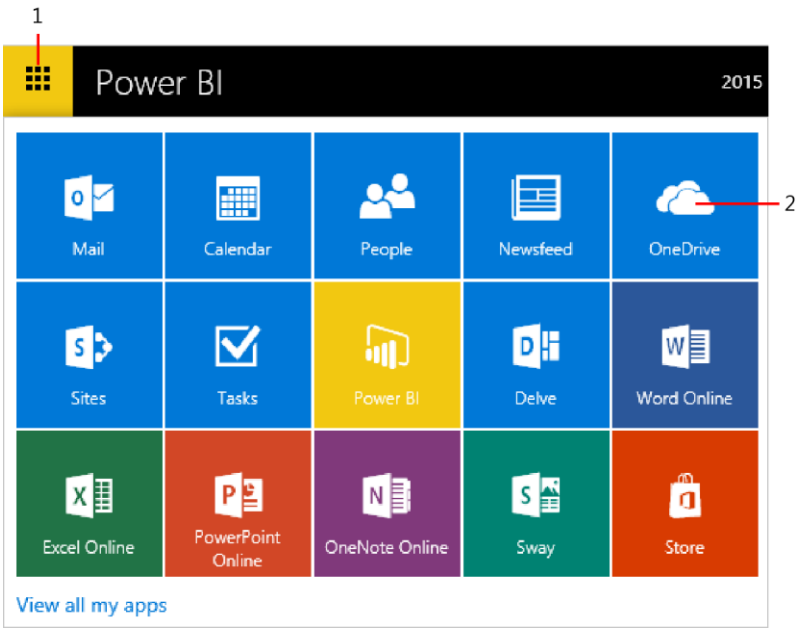


Figure 2-13: You can access OneDrive by clicking the button in the upper-left corner of the Power BI website and then clicking the OneDrive tile.

On the OneDrive webpage that opens, David uploads the workbook to the Budget 2016 group he previously created to share reports and dashboards. A group defined in Power BI corresponds to a group in Office 365, so you have an associated OneDrive for Business folder where you can place files to share. Figure 2-14 shows the sequence of steps that David needs to do to upload the document.

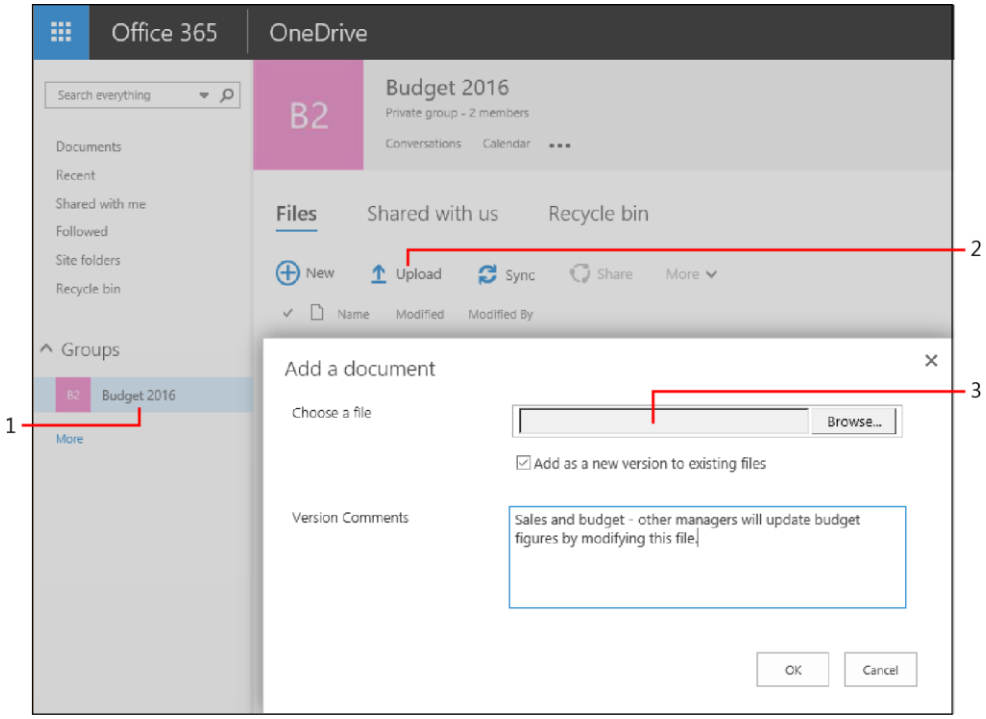


Figure 2-14: The actions required to upload a document in a group folder in OneDrive for Business.

When David finishes uploading the file, he can see it in the list of the files of the Budget 2016 folder, as depicted in Figure 2-15. Also in the figure, there is a Sync button which you can use to get information about how to synchronize the folder with a local computer, so that you can edit the file in a local folder of your PC and automatically upload any update to the shared folder on OneDrive. Thus, all of the files available in this OneDrive folder can now be shared among members of the Budget 2016 group, and the files will be available as a possible data source for reports created in the corresponding Power BI workspace.

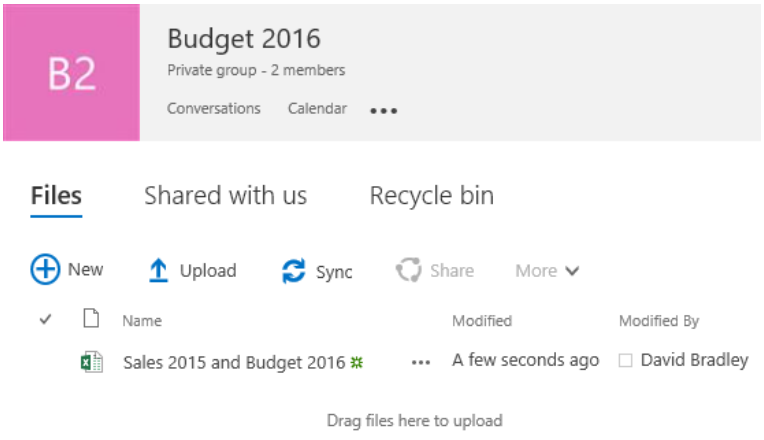


Figure 2-15: After upload, the “Sales 2015 and Budget 2016” file is available and listed in the Budget 2016 group folder.

David has now shared a file to the Budget 2016 group. Later, he will ask other group members to update their budget data themselves. Before doing that, David wants to prepare a report that will display the aggregated total of the budget for every country/region, comparing it with the sales made in previous years.

Going back to Power BI, David opens the list of workspaces available (refer to Figure 2-11) and selects Budget 2016. Power BI displays dashboards, reports, and datasets for that group. Of course, initially all of these lists are empty, as shown in Figure 2-16.

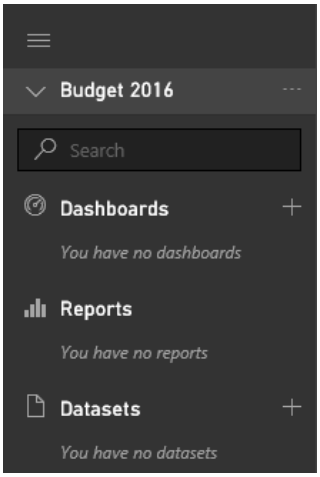


Figure 2-16: The initial list of dashboards, reports, and datasets is empty for a new workspace group.

David wants to create a dashboard and a report based on data stored in the Excel file that now resides in the group folder on OneDrive for Business. The approach is similar to what he did when he made his first foray into Power BI (see Chapter 1), but instead of uploading a file from his own local computer (Local File), this time David selects the OneDrive tile to specify his data source, as shown in Figure 2-17. Notice, though, that the name on the tile is "OneDrive – Budget 2016." Because David is using the Budget 2016 workspace, the associated OneDrive folder is automatically proposed as a possible data source. From a Power BI perspective, the biggest difference between a local file (one stored on your computer) and a file on OneDrive (stored in the cloud) is that the former cannot automatically update a report based on it, whereas the latter can propagate changes to data to Power BI reports without user intervention.

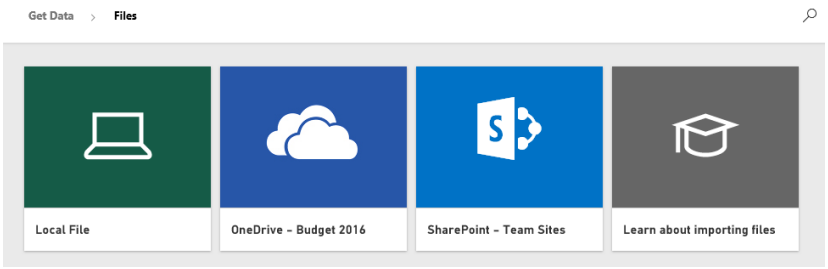


Figure 2-17: Possible sources of files for Power BI include local files and OneDrive.

After David selects OneDrive – Budget 2016, a message asks him to select the file to which he wants to connect to Power BI. There is only one file in this folder, so David clicks it and then clicks Connect, as shown in Figure 2-18. The file selected is highlighted in a different color so that it is recognizable when there are multiple files available.

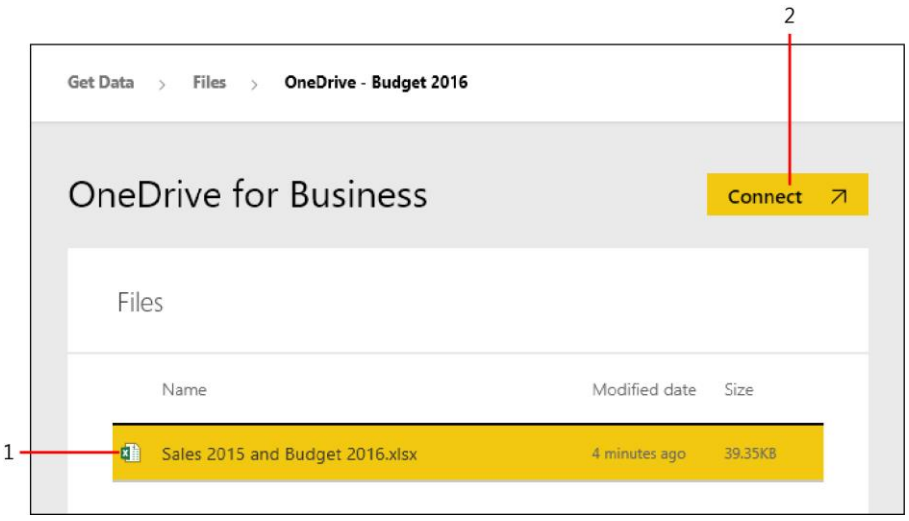


Figure 2-18: The list of files available in OneDrive for Business.

After choosing the Excel file on OneDrive for Business, David must then decide how he wants to use that file. There are two options (see Figure 2-19): Import Excel Data Into Power BI, and Connect, Manage And View Excel In Power BI. If you want to use the Excel file just as a "raw" data source for your reports, select Import Excel Data Into Power BI. Or, you might prefer to copy an existing Excel file as is, using both the data model (if you have one in Power Pivot) and all of the Excel features, such as PivotTables, PivotCharts, and other visualizations available in Excel. If this is the case, choose the Connect, Manage And View Excel In Power BI option. You will see practical examples of this choice later in this book.

David selects Import Excel Data Into Power BI by clicking the Import button in that section.

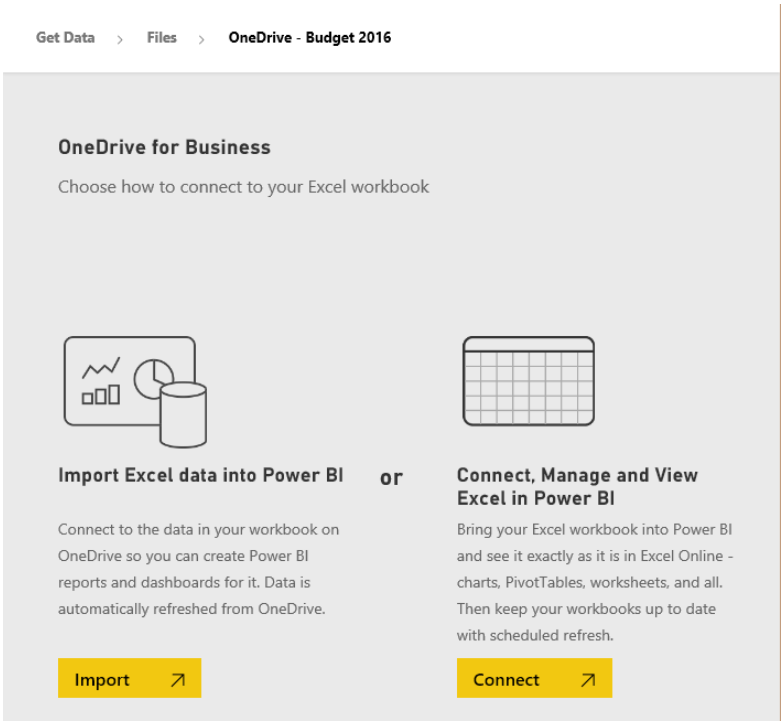


Figure 2-19: Click the Import button to bring in the content of an Excel file stored in OneDrive for Business.

David now has a dataset available, named Sales 2015 And Budget 2016. The dataset contains two tables, Sales and Budget2016, because the Excel file imported has two worksheets with one table each. The Budget2016 table (refer to Figure 2-12) is the one that needs to be modified by other managers, inserting updated numbers for their individual budgets. The Sales table is still the same as it was in Chapter 1 (refer to Figure 1-29), in which David has updated only the Budget column using a formula that searches the corresponding value in the Budget2016 table and allocates it by month. Thus, when a manager updates a row in the Budget2016 table, Excel automatically updates the Sales table.

Using the new dataset, David creates a report by dragging available fields to the report's central pane. The goal of this report is to show a quick recap of the overall budget divided by brand and country/region. For this reason, David chooses a matrix visualization in which he inserts fields from the Sales table (Brand on rows, CountryRegion on columns, and Budget on values, as shown in Figure 2-20). While doing this, he realizes that the two tables have identical fields, and this could be confusing. Because the budget is allocated in the Sales table, it would be nice to hide the Budget2016 table from the Fields list.

When you import an Excel file, all of the tables become part of the Power BI data model and are visible. Later on in the book, you will see how to control the Power BI data model in more detail. For the moment, David just wants to create a first report, and he knows that using the fields from the Sales table is the right choice because he can also create a clustered column chart, below the matrix, that compares the budget with the sales of previous years.

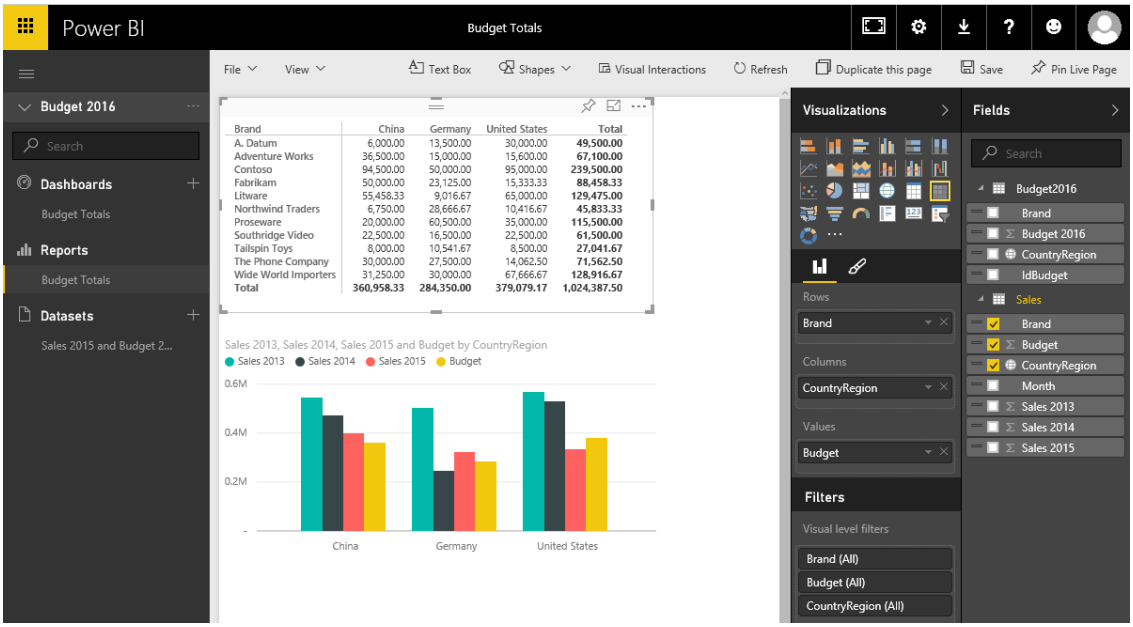


Figure 2-20: The Budget Totals report has both matrix and clustered column chart visualizations.

David saves the new report as Budget Totals and he also pins both visuals to a new dashboard with the same name. David already included Wendy in the group, so he sends her an email asking for a review and edit of budget numbers for Germany (Wendy is the country/region manager for Germany).

Wendy receives David's email, signs in to Power BI, and sees the dashboard. She chooses the Budget 2016 workspace, which she can see because David added her as a member of the group. She opens OneDrive and clicks the Sales 2015 And Budget 2016 file name. Now, she can see the content of the workbook in her browser, and she chooses the Budget worksheet. Next, on the menu bar, Wendy

clicks Edit Workbook and then clicks Edit In Excel Online to edit the file in her browser, as illustrated in Figure 2-21.

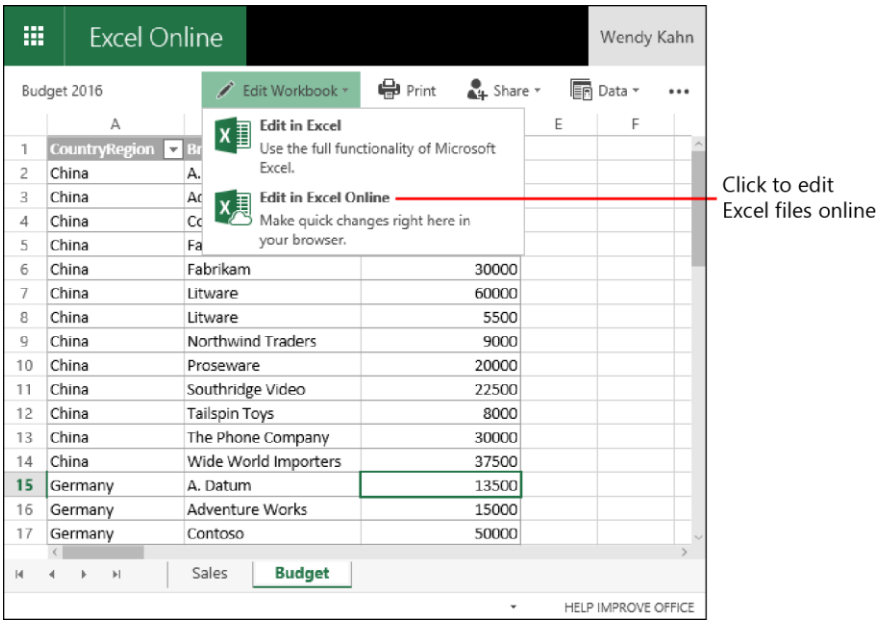


Figure 2-21: Wendy can edit the Excel workbook stored in OneDrive for Business in her browser.

> **Note** It is worth remembering that Excel Online is available on many platforms. This means that Wendy can edit the workbook directly on her iPad.

Wendy corrects the budget for Germany for the A. Datum and Contoso brands (see Figure 2-22), because she knows that different marketing conditions and product lifecycles will affect the previous estimation.

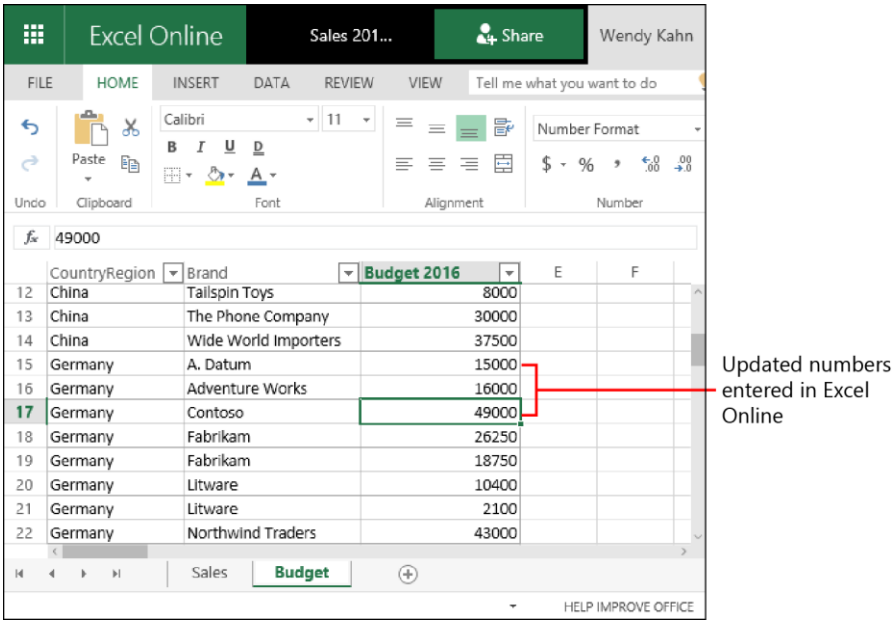


Figure 2-22: The numbers entered in Excel Online are immediately stored, updating the underlying workbook in OneDrive.

Wendy wants to see whether the new values have been correctly allocated, so she clicks the Sales worksheet and sees that the new value (15,000) for the A. Datum brand in Germany is divided into 12 identical parts (1,250 each) for each month, as shown in Figure 2-23. She makes a mental note that such an allocation does not correctly represent the seasonality of certain products, so she will discuss this issue with David during the next meeting.

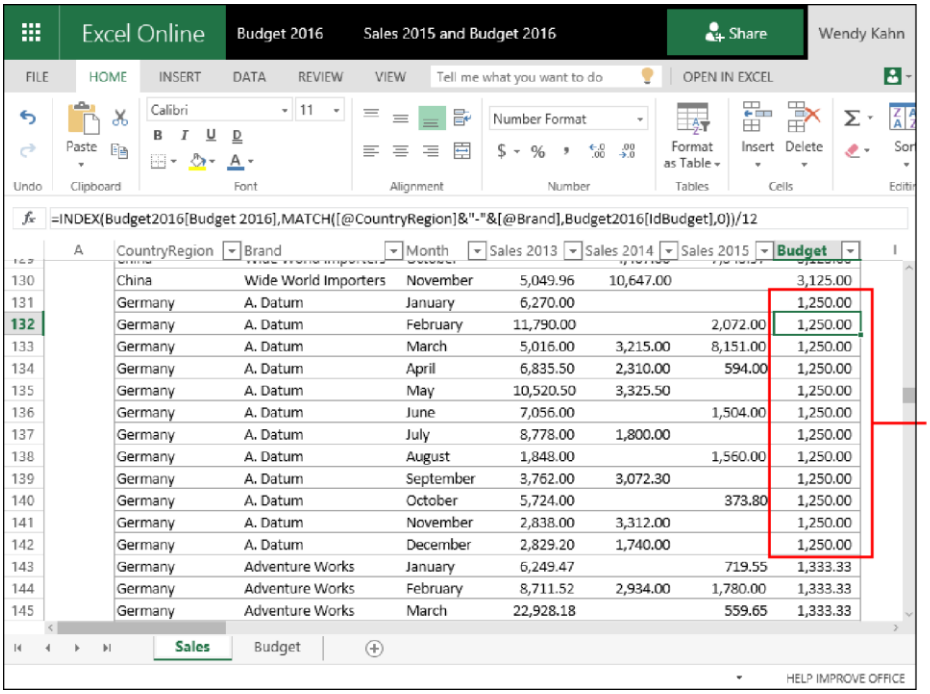


Figure 2-23: The Sales table allocates the budget by month in the Budget column.

Now, Wendy goes back to the Budget Totals dashboard in Power BI and can see the updated data (see Figure 2-24) correctly represented in the matrix. She also can see that all of the other totals are updated, too. Wendy is pleased and feels confident that the reports built by David will be accurate.

Note The Power BI report and dashboard might exhibit some latency when displaying data that was updated in files residing on OneDrive. For performance reasons, the reports are not refreshed every second; thus, if you try to do the same operation, you might not see updated numbers immediately when you go back to Power BI from Excel Online. However, if you wait a few minutes, you will likely see updated numbers. You also might need to refresh the page in the browser to see those updated numbers.

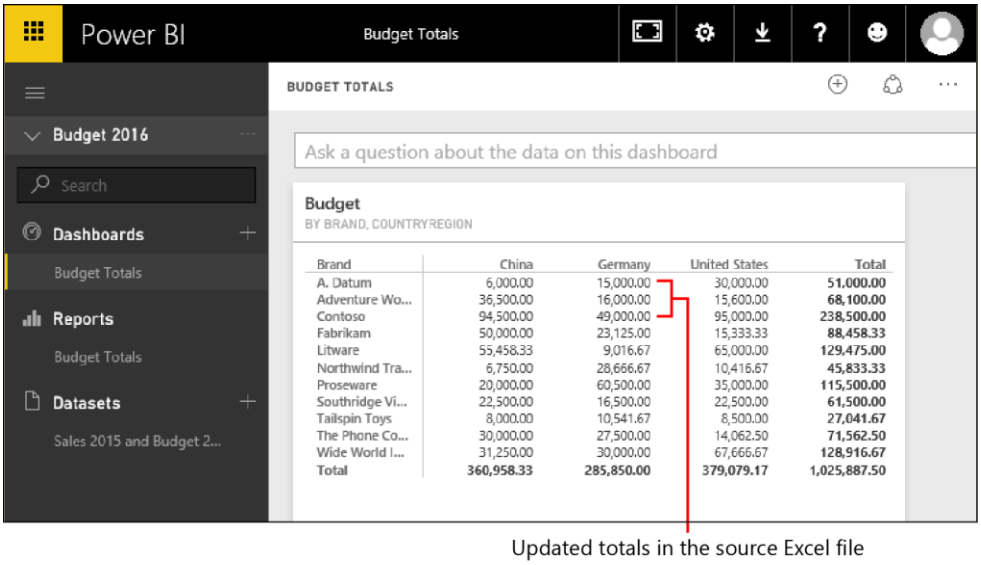


Figure 2-24: The dashboard in Power BI automatically updates numbers when someone modifies the underlying Excel workbook stored in OneDrive.

You have seen how David created a collaborative environment in Power BI by using groups and OneDrive for Business. Such a collaboration requires licenses for Power BI Pro and OneDrive for Business (remember that an Office 365 subscription usually includes OneDrive for Business, too).

Using groups, you can share files in OneDrive for Business only with group members. If you want to share a file with someone outside the group, you must use the personal workspace in Power BI and the personal folder in OneDrive for Business. Remember: only group members can see everything you store in a group.

If you do not have OneDrive for Business available, you can still use the personal edition of OneDrive (which is available at no charge), but the scenario described in this chapter will be slightly different. You can share an Excel workbook in OneDrive, and other people can edit its content if you share the workbook with them. However, you can import such a workbook only to the personal workspace in Power BI, not to a group workspace. Thus, it is possible to do the following:

- Create an Excel workbook on OneDrive.
- Share the Excel file with other users, even those outside your organization (there are several limitations when doing so with OneDrive for Business).
- Import the Excel file stored on OneDrive in a Power BI dataset.
- Use the dataset in your reports and dashboards. However, you can share them only within your organization, not external users.

You can use the personal OneDrive to create a very similar scenario, but you will not gain the benefit of automatic visibility of files to group members as you do by using OneDrive for Business.

Viewing reports and dashboards on mobile devices

All users who can access Power BI can see all of the reports and dashboards, even on native applications for mobile devices. There are native Power BI apps for Windows (available at the Windows store), iOS (available at the Apple store), and for Android (available at Google Play). These

native apps also have additional features, such as annotations. They are updated with new features regularly.

Getting back to David, he might, for example, want to show the dashboard on his Windows 10 tablet device to a colleague during a meeting. Because the meeting room does not have good Internet connectivity, he takes advantage of the offline availability of dashboards by using the mobile app. This feature also makes it possible to view reports offline, even if you have no interactive capabilities. Thus, David can see the data available with the last refresh of the report, and this is enough for the meeting he planned. Figure 2-25 shows that the rendering of the Budget Totals report is pretty similar to the one available in the browser.

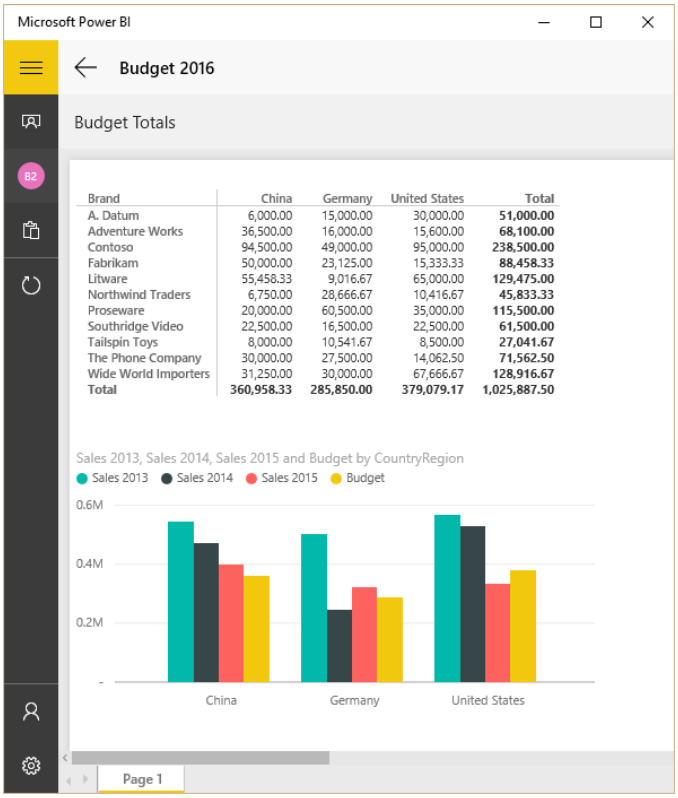


Figure 2-25: Rendering of a report in the Power BI app for Windows 10.

You might also experience different visual presentations in the Power BI app on tablets and smartphones. For example, Figure 2-26 demonstrates how an iPad displays the dashboard created by David in Chapter 1. You can zoom in to each visualization and navigate with the benefit of larger graphics, making it easier to read numbers, as illustrated in Figure 2-27.

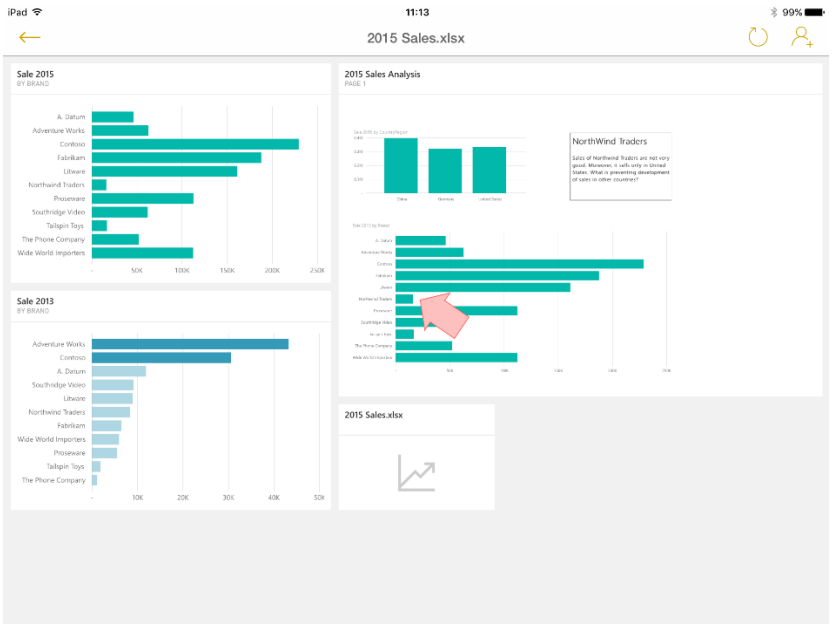


Figure 2-26: Rendering of a dashboard in the Power BI app for iPad.

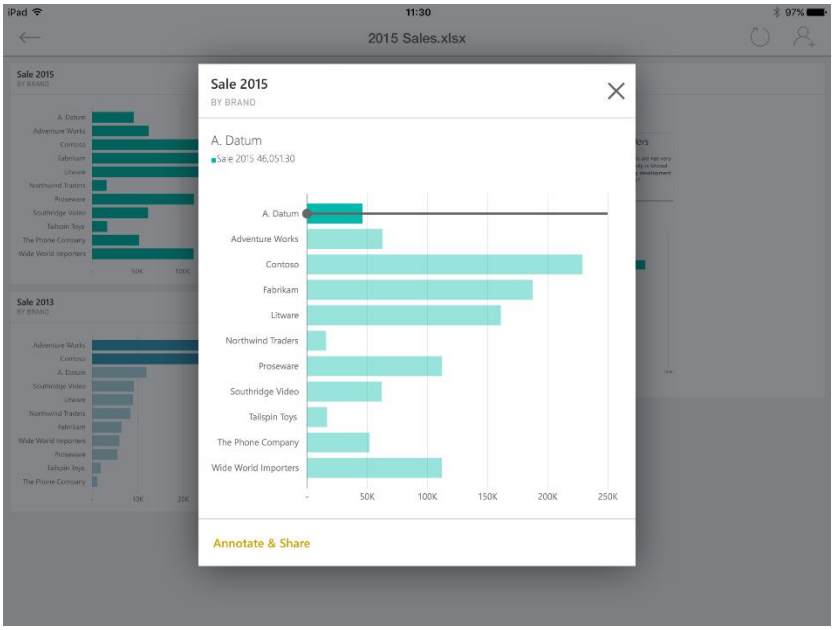


Figure 2-27: An enlarged view of a dashboard's visualization in the Power BI app for iPad.

That same dashboard is rendered differently on the smaller screen of an Android smartphone, as depicted in Figure 2-28. Visualizations are organized in a vertical column, and you can zoom in to each one, as shown in Figure 2-29.

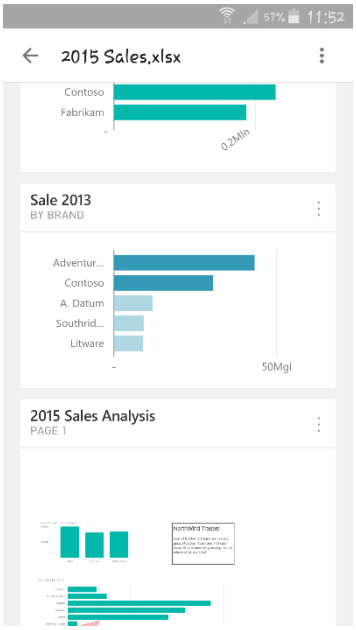


Figure 2-28: Rendering of a dashboard in the Power BI app for an Android smartphone.

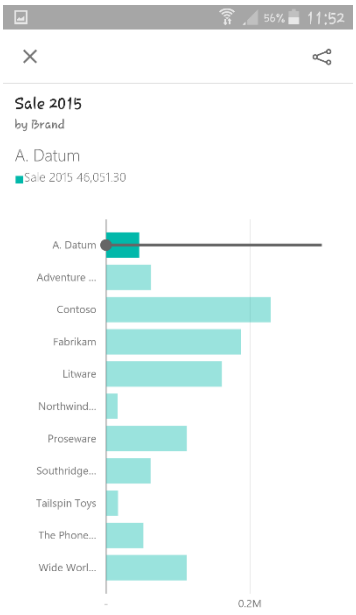


Figure 2-29: An enlarged dashboard visualization in the Power BI app for an Android smartphone.

Users who need to enter budget data in Excel files can also choose to use the Excel app on mobile devices. This app provides a user interface that is optimized for the device, which offers a better user experience than the generic web user interface of Excel Online. Figure 2-30 and Figure 2-31 show how the two worksheets of the Sales 2015 And Budget 2016 workbook display on an Android smartphone.

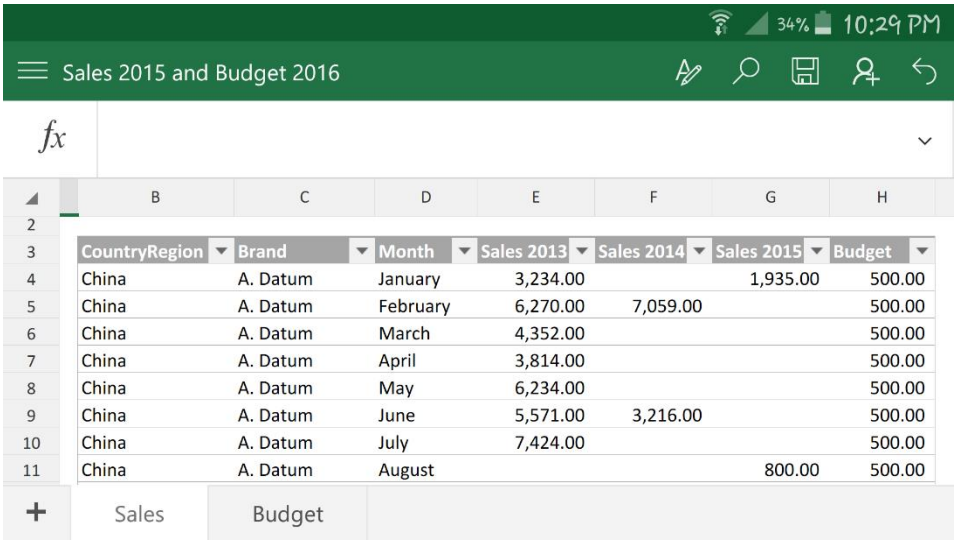


Figure 2-30: The Excel app on an Android smartphone, displaying the Sales worksheet.

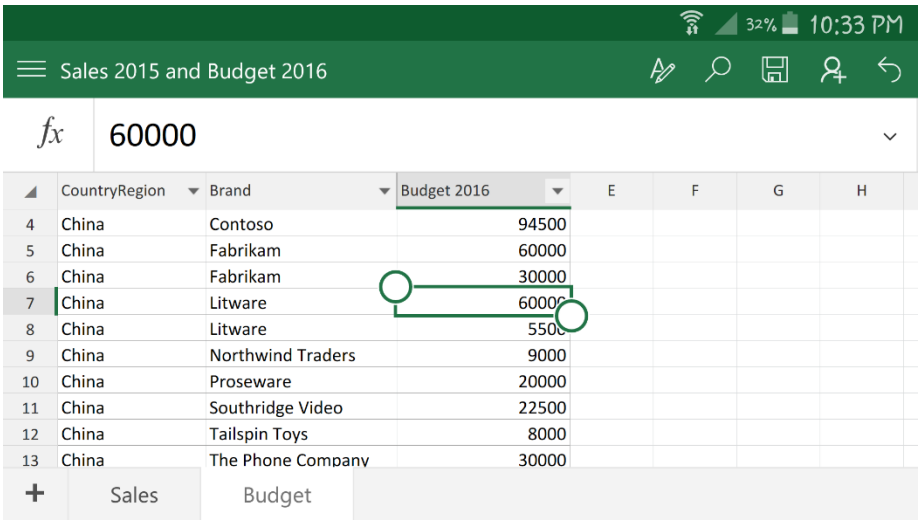


Figure 2-31: The Excel app on an Android smartphone, displaying the Budget worksheet. Users can enter data here.

Different operating systems and devices might provide a slightly different user interface. The goal here is just to inform you of the options available for tools that complement the Power BI service and app. We suggest that you try these applications for yourself, with your own data, so that you can evaluate which of your data for Power BI can be displayed/edited using a mobile device.

Conclusions

In this chapter, you saw how to share dashboards, reports, and raw data with people within an organization.

- You can invite a user within your organization to see a dashboard. That person can also see the reports underlying the dashboard.
- To share a dashboard with users outside of your organization, you must provide them an account within the same tenant.

- You can publish a report on a webpage; however, anyone can view that report. You should consider this option only for data intended for public consumption; you should not use it for sensitive, company-private data.
- You can create groups of users within your organization and share datasets, reports, and dashboards.
- You can use the same groups with OneDrive for Business to make it possible for other users to directly enter data in shared reports, which are then automatically updated after each change.
- You can display dashboards and reports on mobile devices by using native apps. These apps typically provide a better user experience if you are using smaller displays and have limited offline access to dashboards.

Get the latest news from Microsoft Press sent to your inbox.

- New and upcoming books
- Special offers
- Free eBooks
- How-to articles

Sign up today at MicrosoftPressStore.com/Newsletters

Microsoft

Understanding data refresh

Let's continue following David, the manager of budgeting at Contoso. While he explores Microsoft Power BI, he now has begun sharing his dashboards and reports with Contoso's country/region managers around the globe. Everybody likes the idea of being able to view a report on any device and share considerations while looking at the same figures. Nevertheless, the managers are concerned that they are making decisions based on sales as of October 2015, and it is now mid-December. The figures are no longer the best data upon which to forecast sales, a fact that is even more pertinent for the products that show a clear seasonality.

Thus, David needs to retrieve the latest sales data set and refresh the Microsoft Excel model. Urged on by the country/region managers, he ends up doing this each and every morning until he begins to wonder whether this process can be automated in some way. In fact, Power BI offers an option with which he can schedule refreshes.

Introducing data refresh

In Chapter 1 and Chapter 2, you learned the basics of Power BI: how to upload a workbook containing some data, build reports and dashboards, and how to share the content with other users in your organization or outside of it. Those chapters also touched upon the basics of data refresh: uploading a new version of a workbook containing data, and using Microsoft OneDrive for Business to automate the uploading process.

For both of these scenarios, updating data meant that David needed to refresh the content of the Excel file manually and then upload it to Power BI, either via another web service, such as OneDrive for Business, or by using the Power BI user interface (UI). When it comes to refreshing your data, Power BI offers much more control. Learning it requires some attention to details. We suggest that you read this chapter in its entirety because there are some small but important details that you need to know before making any decision about your future data refresh strategy.

More important, in the first two chapters, we kept the presentation deliberately simple, trying to show only the basic concepts. In this chapter, however, we begin to delve into the details of working with Power BI. Hereinafter, the details become important.

Anyway, first, we need to focus on what "refresh" really means. In the context of this chapter, refreshing data does not mean to manually update the Excel workbook and save it as another version of the same file. Instead, we want the workbook to automatically update its content by using a connection to the source database from which we originally query the data.

Let's review the steps in David's simple data-processing system:

1. Data is retrieved from the database by IT. IT then gives David an Excel file containing the latest figures of sales.
2. David manually copies the information to his own version of the Sales 2015 workbook.
3. He saves it so that OneDrive uploads the content to the cloud.
4. Power BI loads the content of the file from OneDrive and updates its own internal data model.

As part of step 3, the Excel file automatically computes the forecasts generated by the country/region managers by using formulas. These results are saved in OneDrive by the managers, so the results are immediately available to Power BI, too.

The data refresh mechanisms David learned so far are useful to automate step 4. But now, he wants to automate steps 1 and 2. This requires some more understanding of how Power BI works internally.

Introducing the Power BI refresh architecture

What happens when you upload a workbook to Power BI? Let's consider the workbook that David uploaded to look more closely at the process. Recall that his workbook contained a table. Figure 3-1 shows the flow of data.

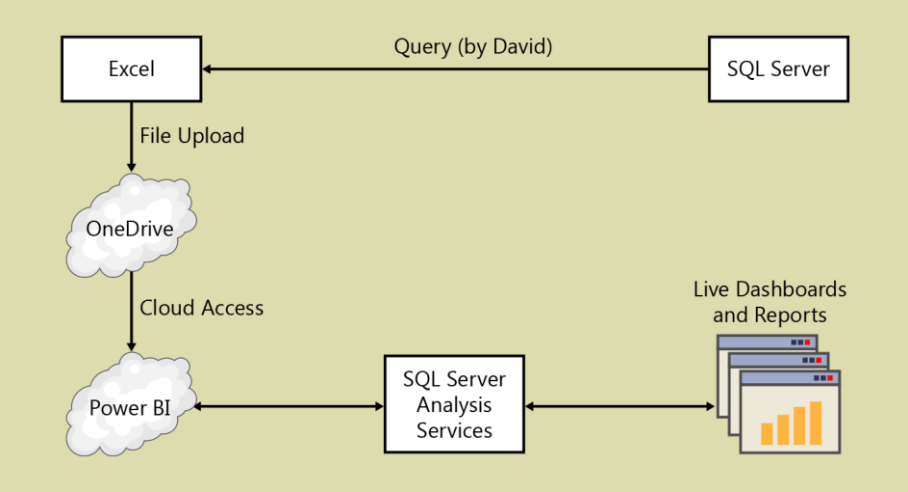


Figure 3-1: The data flow, from the original database up to Power BI.

David's data moves from SQL Server into Excel (David does that). Next, the Excel file is uploaded to OneDrive where Power BI reads it. After reading the file, Power BI generates a SQL Server Analysis Services (SSAS) database that ultimately computes, through the Power BI UI, the dashboards and reports. Sounds complicated, right? In fact, it is, but luckily, Power BI hides all of this complexity, making it easy for you to generate reports from Excel files. Nevertheless, to understand how data refresh works, you need to have a clear picture of the complete flow of information.

To use data refresh, you need a way to pull data from the data source (SQL Server, in this example) and push it directly into the SSAS model generated by Power BI. In other words, you want to create a data flow that circumvents Excel and OneDrive (both operations are done outside of Power BI) to make it flow as shown in Figure 3-2, in which the steps that we want to remove appear in a blue box.

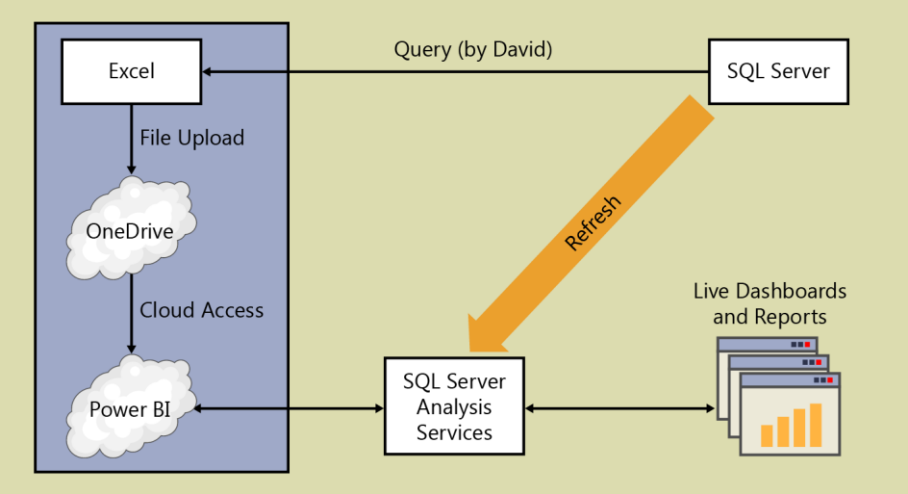


Figure 3-2: We want to remove the steps enclosed in the blue box to make data refresh work more efficiently.

Here is what you need to do to make this happen:

- The data set cannot be a plain Excel table, because Power BI needs to know how to query the source database (SQL Server, in this example) in order to refresh the data. Obviously, it cannot rely on asking you or asking IT. The method must be formalized and use a language that Power BI can understand.

- The SSAS engine running in Power BI needs a way to access the source database. Such a database is usually located within your company (or on your laptop). Thus, you will need software that implements the connection with Power BI.

Be sure that you understand these architectural requirements well before moving on with the rest of the chapter. As you will see by reading the next sections, we stripped away most of the technical complexities that comprise data refresh, but you need to keep in mind that the aim is to create the scenario depicted in Figure 3-2.

Introducing Power BI Desktop

You might remember from Chapter 1 that Power BI offers you two ways of interacting with it: direct access to the web service, or by using Power BI Desktop. In this section, we are going to show you how to use Power BI Desktop. Power BI Desktop is an application that runs locally on your computer but offers you all of the features available on the web, plus many more options for building a data model.

By using Power BI Desktop, you do not rely on the web service to create the data model for you. Instead, you have the full modeling capabilities of Power BI available at your disposal and control.

Why should you worry about Power BI Desktop at all if the web service is capable of building a model for you? There are several reason for this, but for now, let's concentrate on one of those reasons: you can describe the details of your dataset, which accomplishes the first requirement of our data-refresh scenarios from the preceding section. But, before we can get to that, first you need to download Power BI Desktop from the Power BI website and install it.

On the Power BI website, on the right side of the menu bar at the top, click the download button (see step 1 in Figure 3-3), and then click Power BI Desktop (see step 2).

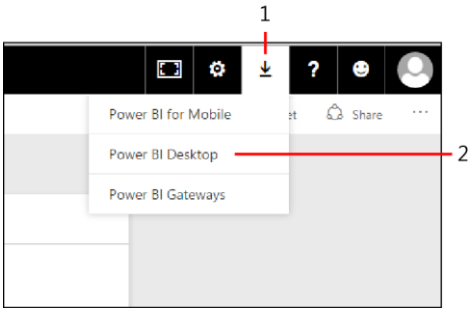


Figure 3-3: Downloads in Power BI are available in the download list, on the right side of the menu bar.

After you have downloaded Power BI Desktop, install it by following the instructions provided in the Microsoft Power BI Desktop Setup Wizard, and then start the application. A welcome screen greets you, and then you see the main Power BI Desktop window, as shown in Figure 3-4.

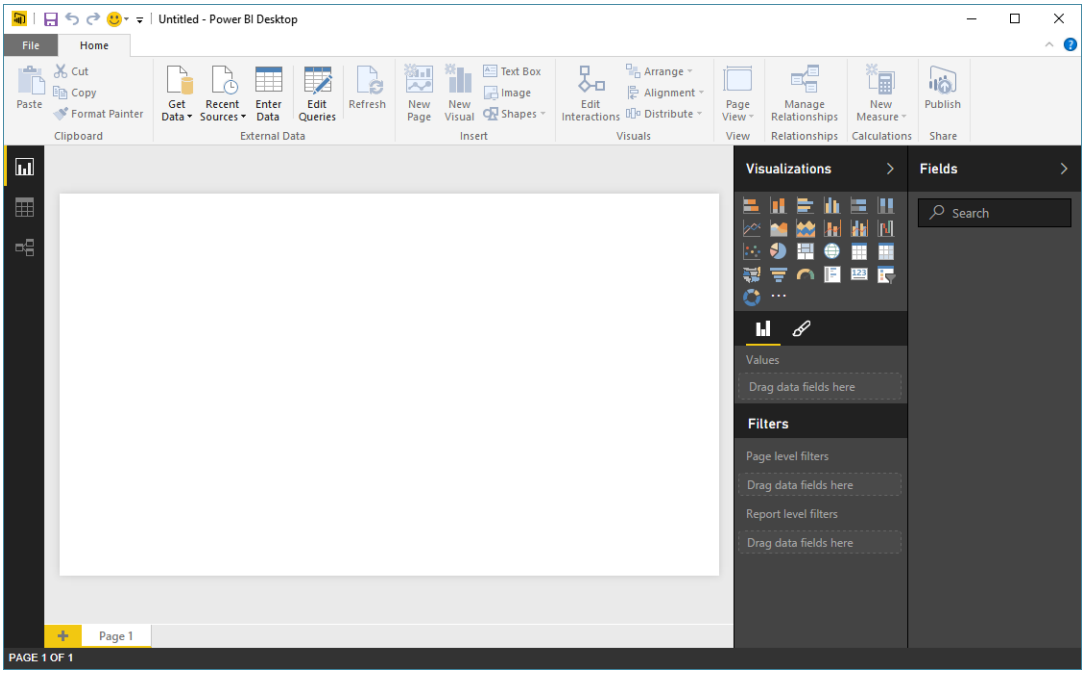


Figure 3-4: The Power BI Desktop UI resembles the Power BI website.

The first thing you will probably notice is that the UI of Power BI Desktop is very similar to that of Power BI. Nevertheless, as you will learn, there is more to Power BI Desktop, and it is a bit more complex to use than its web-based counterpart, but it exploits the full power of the Power BI engine.

As with Power BI, the first step is to provide some data to Power BI Desktop. Because the original data is in Excel, you can begin practicing with Power BI Desktop by using the same Excel file you used earlier for the website. On the ribbon, in the Data group, click Get Data, and then click Excel (see Figure 3-5).

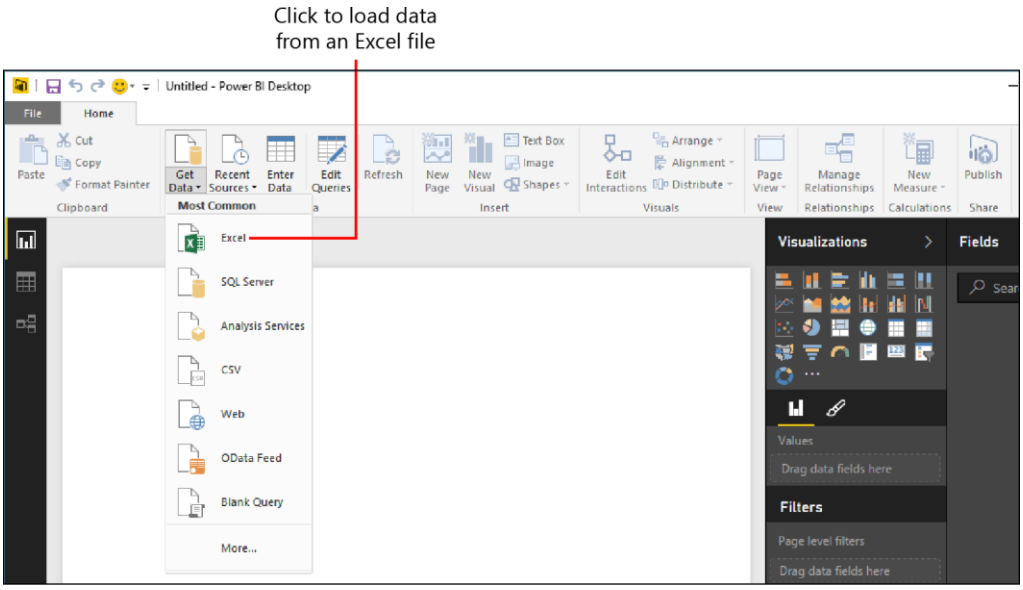


Figure 3-5: Your first step when using Power BI Desktop is to load some data; in this example it's from Excel.

In the Open dialog box, select an Excel file and then click Open. The Navigator dialog box opens, in which you select the source file. When you load data from Excel, you can choose to load from tables

or from worksheets. Figure 3-6 shows the two sample items available. Note that for this exercise we renamed the worksheet "Sales" to "Sales Worksheet" to make the figure clearer. In your models, it is likely they will have the same name, and only the icons adjacent to the names will differentiate tables from worksheets. For this example, let's work with the table.

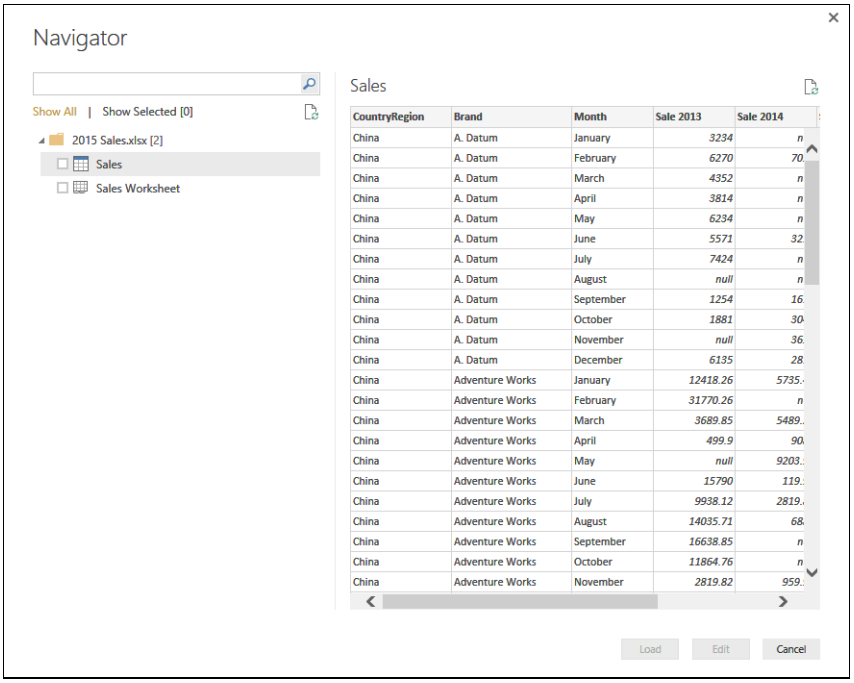


Figure 3-6: The Power BI Navigator helps you to choose the source for importing data in Power BI Desktop.

Select Sales, click Load to import the table into Power BI Desktop, and then close the Navigator dialog box. At this point, you will likely feel at home and in familiar surroundings. In fact, by using the Fields and Visualizations panes, you can build a report in Power BI Desktop in the very same way you built the report earlier, on the website.

For example, Figure 3-7 shows that you can build a report similar to the one you built on the website. The main difference is that, now, you are doing the work directly on your PC instead of interacting with a cloud service.

> **Note** We suggest that you become familiar with the UI of Power BI Desktop by experimenting with some reports. In this chapter, we are not showing you a step-by-step guide to Power BI Desktop. Instead, we focus on the new features available in Power BI Desktop that are not in the cloud service.

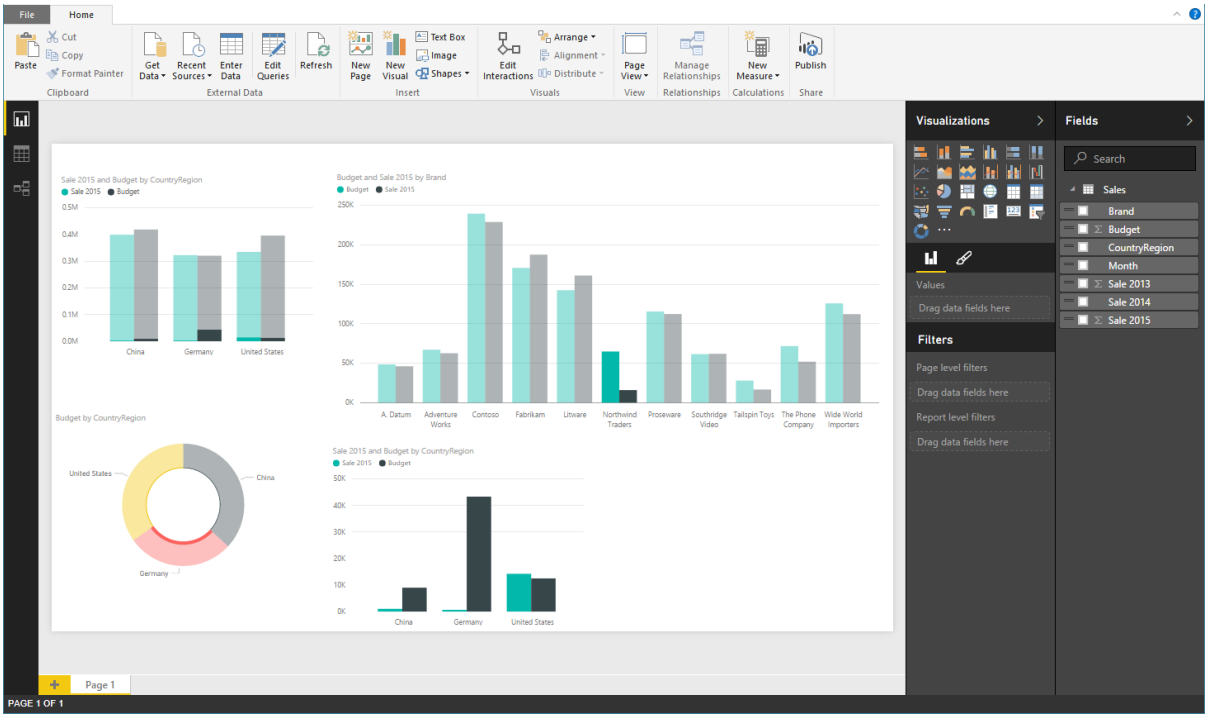


Figure 3-7: This is a sample report that you can build by using Power BI Desktop.

Building the report, you can appreciate how user-friendly the Power BI Desktop environment is, with features such as copy-and-paste available. So, for example, if you need a visualization similar to one you have already created, copy it, paste it, et voilà: the job is done. Of course, there is more to it than that, but Power BI Desktop offers little things like this that make life much better.

Publishing to Power BI

Let's get back to our exercise. When the model is ready, save it on your computer using the name **Sales PBD**. Of course, at this point, the report is local to your PC, and no one else can view it. However, you ultimately want to publish this model on the Power BI cloud service, to take advantage of all the features of Power BI, including sharing and viewing on a mobile device.

To do this, go to the Power BI Desktop ribbon, and then, on the Home tab, in the Share group, click Publish. Power BI Desktop then asks you to sign in to the Power BI service (and might ask if you want to modify your changes). As previously mentioned, Power BI Desktop is an application that runs locally on your PC, and it can work without a Power BI account, but as soon as you want to publish your data, you need to have an account (or create one) and sign in. After you sign in, Power BI Desktop shows the message depicted in Figure 3-8, confirming the operation and providing a link to the published report.

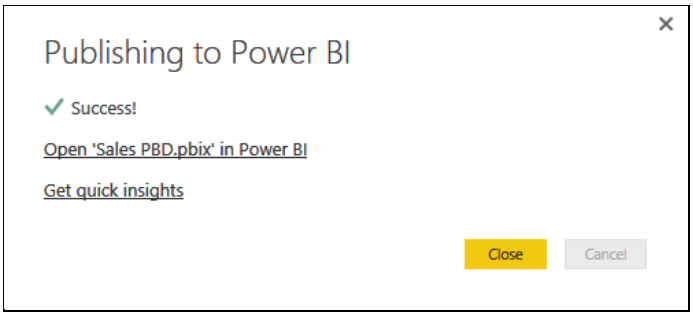


Figure 3-8: When the model is published, you can immediately see it using the appropriate link.

If you open the file in the Power BI website, you will find a dataset named Sales PBD and a report with the same name. By using Power BI Desktop, you created both a model and a report, and, when publishing it to Power BI, it created both objects.

> **Note** The model is copied from your local file to Power BI. When the model is in Power BI, you can further enhance it, but the two versions are disconnected. Changes that you apply to the published model from within your browser are not be applied to your local version on your PC, and any subsequent publish operation that you initiate from Power BI Desktop will overwrite the changes that you made via the web browser. Thus, when you begin building models with Power BI Desktop, it is a good practice to continue updating them locally and then republish them. Do not modify the online version. You might want to use different version names for your local files in order to publish different, varied reports in the Power BI cloud service.

After you get used to this method of developing reports, you will find it extremely convenient. In fact, working with Power BI Desktop is more productive because you do not need an Internet connection and you have the full power of a Windows application. When the model is ready, you publish it, overwriting any previous version that you might already have done.

Let's recap what we have seen so far:

- Power BI Desktop is a Windows application that offers the same features of the cloud service, but it runs on your local PC.
- You can build a model with Power BI Desktop and save it to your PC.
- You can publish a Power BI Desktop model to Power BI, but you'll need to have or create an account and sign in first.

You might remember that this chapter is about data refresh. Why is Power BI Desktop relevant to data refresh? A Power BI Desktop file contains all the information needed to refresh the model. In fact, in the Power BI Desktop file, we created a link between the original Excel file containing figures for the budget and the Power BI Desktop model.

In Chapter 1, when we uploaded the file to Power BI, we simply copied it. However, using Power BI Desktop, we create a link between the Excel file and the Power BI Desktop file by writing a query. In reality, we do not actually author any query, but as you will learn, Power BI Desktop created the query for us, making the task transparent.

All that is missing at this point is to provide a way for Power BI to access the Excel file that is the ultimate source of our data. In fact, the original Excel file is stored on the local PC, and the Power BI cloud service cannot access it. Solving this problem is the topic of the next section.

Installing the Power BI Personal Gateway

The Power BI Personal Gateway is another piece of software that can connect with the Power BI cloud service and carry out the queries stored in the Power BI Desktop file. You can download it from the same webpage from which you downloaded the Power BI Desktop application, but this time select Power BI Gateways after you click the download button. Power BI then asks you to choose between the two Power BI gateways:

- **Personal Gateway** This version is intended for use with personal datasets. It is simple to use and install but offers limited features regarding monitoring and security for multiple users.
- **Enterprise Gateway** This version offers more functionality but, at the same time, involves more complexity in its setup and usage and usually requires involving your IT department.

> **Note** In the last chapter of this book, we briefly outline the differences between the personal and the enterprise versions of the gateway. For the purposes of this discussion on how data refresh works, those differences are negligible.

Let's go back and visit David to see how he is progressing.

David is still experimenting with Power BI, so he has no intention of installing a complex system. For this reason, he chooses the Personal Gateway. Before jumping into the setup of the Personal Gateway, though, he needs to understand how it will be implemented.

If David runs the setup with administrative privileges (that is, he installs it as an administrator), the gateway will run as a service. On the other hand, if he installs it as a standard user, the gateway runs as a normal program. What is the difference? When the gateway runs as a service, it runs even when no user (or another user) is signed in to his PC. Conversely, if it runs as a standard program, the gateway will run only if David is signed in to his PC. This might be relevant if at some point he wants to refresh his data while at home, accessing the Power BI cloud service, but his laptop is still in the office, turned on but without anybody using it. In such a scenario, if the gateway runs as a service, the refresh operation will succeed, whereas if it runs as a normal program, it will fail.

The choice of which to run is up to you. Figure 3-9 shows David choosing to run the installer as an administrator.

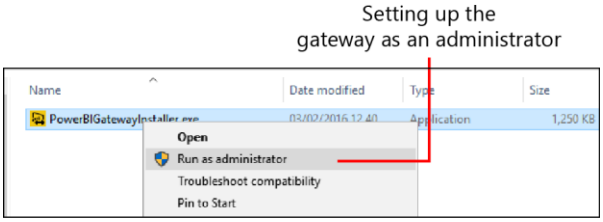


Figure 3-9: Right-click the downloaded installer to install the gateway as an administrator.

After you install the gateway, you must start it to complete the configuration. In fact, when you start it, it requires the credentials to access Power BI. The gateway requires them because it must contact the Power BI cloud service and begin answering queries coming from the service.

After David provides the correct sign-in, the Power BI Gateway – Personal dialog box opens, in which he must provide another set of credentials, as illustrated in Figure 3-10.

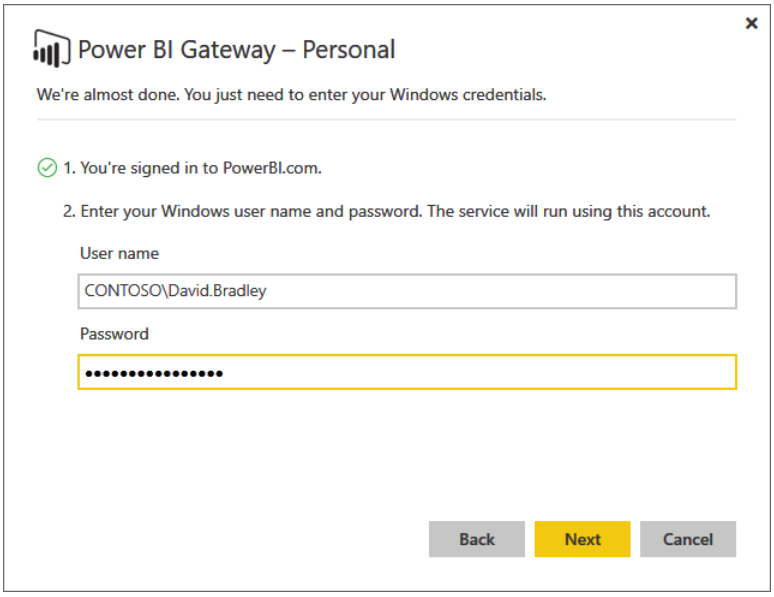


Figure 3-10: As part of the configuration, the gateway asks for the credentials to use when running as a service.

Why does the gateway require these credentials? Because David wants to run it as a service. Being a service, it will run even if no one is connected to the PC, and it requires credentials to access files and connections from David's PC. In other words, the gateway will have the same permissions that David has on his PC, and those are required to let it access his files and data sets as if it were him.

When everything is complete, the gateway informs you that one operation still remains: you need to go to powerbi.com to complete the setup of the data sources. When you go there, you will reenter your credentials.

> **Note** At first sight, it might seem cumbersome to be required to enter the credentials so many times. Indeed, it is important that you follow the procedure in the correct way. The gateway connects the Power BI website to your PC, and it will be able to access all of your files. Security is always important, and Microsoft takes it very seriously.

David is now near the end of the gateway configuration. The last step is to visit the Power BI cloud service. To complete the setup, in Power BI, on the menu bar, he clicks the configuration button (the small gear icon; step 1 in Figure 3-11) and then chooses Settings (step 2). This opens the Settings page on which he can configure the settings for each data set, as demonstrated in Figure 3-11.

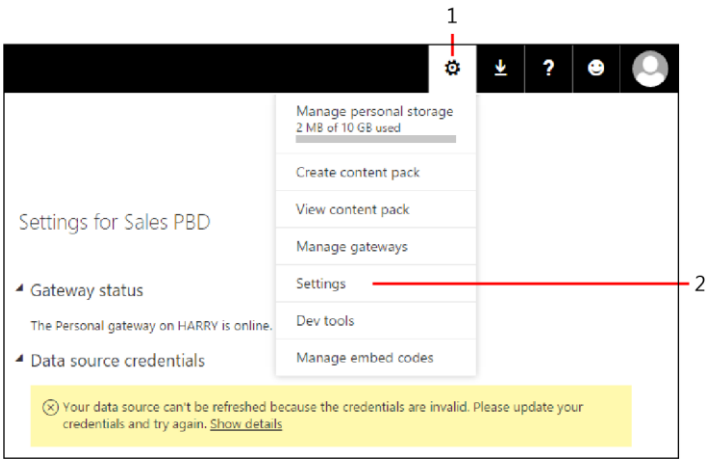


Figure 3-11: The final step in the configuration of the gateway is to grant permission for a Power BI dataset to access it.

On the Settings page, David sees an alert with two important pieces of information:

- The gateway is online and running (in this case) on a PC named HARRY (which you can see in Figure 3-11, in the line below the Gateway Status section header).
- This data set is not yet ready. In fact, even if the gateway is set up, you need to specify the credentials specific for the single data source.

Why do you need to enter credentials again for each data source? The reason is that every data source might require different user credentials. In our example, the Sales PBD file is stored on David's local PC, so it is automatically accessible by using the Windows credentials that David stored in the Personal Gateway, too. In this case, when David clicks Edit Credentials, he is asked to choose an authentication method, as illustrated in Figure 3-12. The Windows authentication method is the only option, and when David clicks Sign In, he will not need to provide his user name and password again. However, for other data sources not using Windows authentication, clicking Sign In will require a user name and password to make it possible for the Personal Gateway to connect to that data source during a refresh operation.

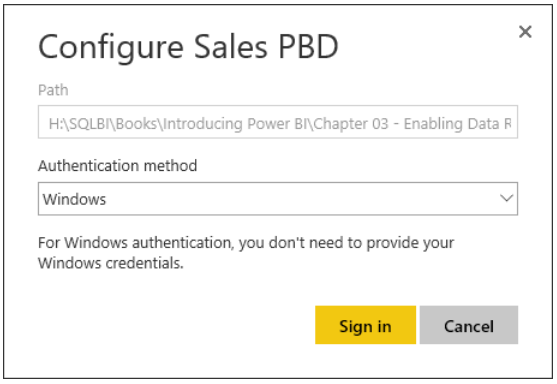


Figure 3-12: You have a choice of authentication methods for each data source.

Configuring automatic refresh

After you set the credentials, the data source is ready to be refreshed. Power BI now has all of the information it needs to refresh the data, both on demand and on a scheduled basis. Expanding the Schedule Refresh section, you can define when Power BI attempts to refresh the dataset. Figure 3-13 shows an example of a data refresh scheduled twice daily, at 9:00 AM and 4:00 PM.

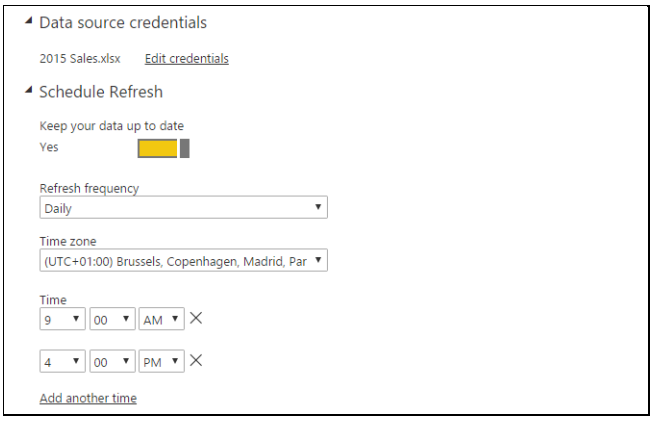


Figure 3-13: You can configure automatic data refresh to run daily or weekly and at different times.

> **Note** You need a license for Power BI Pro to schedule more than one refresh per day. Using the free Power BI license, you can schedule only one daily refresh.

In case the refresh fails, you have the option to receive an email alert so that you can take the needed remedial actions.

At this point, you can either wait for the refresh to happen or, if you want to be sure that everything is set up correctly, you can force an immediate refresh. To perform this operation, in the navigation pane on the left, in the Datasets section, click the ellipsis to the right of the data source (Sales PBD, in our example; step 1 in Figure 3-14), and then click Refresh Now, shown as step 2 in Figure 3-14.

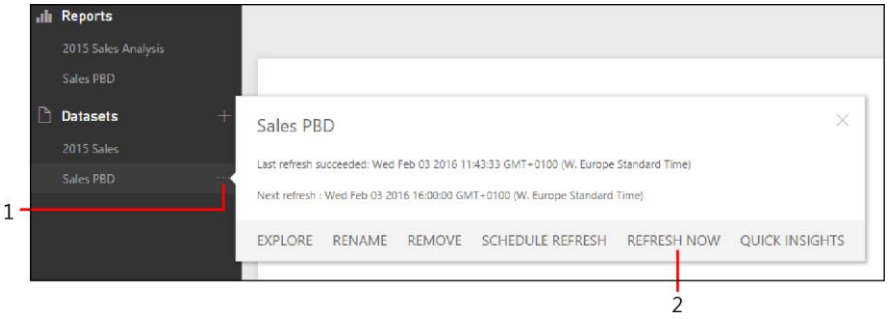


Figure 3-14: You can refresh a dataset immediately by clicking Refresh Now.

When you ask for a refresh, Power BI prepares for the data refresh and then starts it. Depending on the size of the dataset and the speed of the Internet connection, the time to carry out the refresh can range from a few seconds, as in the case of David, to a much longer duration. You can see when a dataset was last refreshed by looking in the same window where you asked for the immediate refresh (click the ellipsis to the right of the data set).

Note Although it might be obvious, it is useful to state an important fact: the data refresh happened on the model in the cloud, not in the model on David's PC. In fact, if David opens Sales PBD by using Power BI Desktop, he will see the data as he saved it. Power BI does not change any file on your PC, it only uses the gateway as a medium to access the data sets upon which the model is based.

Conclusions

In this chapter, you learned the basics of data refresh. Let's recap them briefly:

- You can upload simple data models, based on an Excel file, to Power BI and refresh them by using the Personal Gateway, which makes it possible for Power BI to access your local datasets.
- For data refresh to work on more complex models—for example those that load data from SQL Server—you need to use Power BI Desktop.
- With Power BI Desktop, you build models containing the needed information to let the cloud service connect to the Personal Gateway and retrieve the dataset.
- You can refresh your data daily with the free license, whereas you need a professional license if you need to refresh your model multiple times each day.

Power BI Desktop offers many more features and will let you move to the next level in the learning path of data modeling. This is the topic for Chapter 4.

CHAPTER 4

Using Power BI Desktop

In Chapter 3, we introduced the concept of data refresh. Contoso's manager of budgeting, David, was able to refresh a Power BI model, based on a Microsoft Excel workbook that contains sales for the last three years and forecasts for the next year. To perform that, he had to learn the basics of Power BI Desktop, a Windows application that brings the full modeling power of Power BI to your desktop.

In this chapter, David will move a few steps further in building his Power BI model. But, the solution he has built so far still depends on IT providing him with the sales figures for the last three years. Fortunately, David discovers that Power BI Desktop can load the numbers directly from the Contoso data warehouse via Microsoft SQL Server. Now, when Power BI carries out data refresh, it will automatically get the latest sales data, updating the entire model.

Let's take a brief look at what David needs to do:

- Load sales figures from the data warehouse instead of using his Excel file. This requires accessing the corporate database, but the IT department can give him access to the data he needs.
- Load forecasts for the next year from the Excel file that the country/region managers update every day.

Power BI Desktop offers all of the functionalities that David requires. Now, let's go deeper and learn how to take advantage of this extremely useful application.

Connecting to a database

David already knows how to load data into Power BI Desktop from Excel; he always did it using the Excel file that he received from IT. Now he needs to work from a database. To access the database, he will ask the IT department to provide him the credentials to query the database and read the information he needs. Karin, the database administrator at Contoso, grants him read access to a view that returns the same dataset that she has been sending him every day.

Karin is happy to do so because she will no longer need to fill the Excel workbook. If David can load the data and manage to grab insights by himself, her daily list of chores will become lighter. Thus, Karin tells David, "You can access the view named Sales2015 using your Windows credentials on the server ContosoDbServer." David's account can only read that view, so there is no potential danger. Database security will guarantee that nothing bad can happen to the database.

In Power BI Desktop, on the Home tab, David uses the Get Data function, this time using the SQL Server option. In the dialog box that opens, he provides the connection information Karin gave him (see Figure 4-1) and then clicks OK.

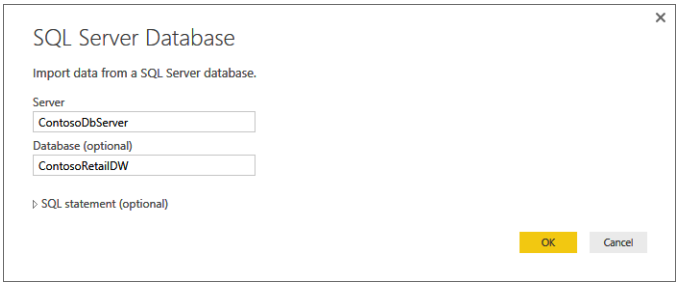


Figure 4-1: To connect to SQL Server, you need to provide the server location.

Power BI Desktop connects to the database and presents a list of data tables, which includes the one that Karin created, Sales2015. When David clicks it, Power BI Desktop shows a preview of the content, as depicted in Figure 4-2.

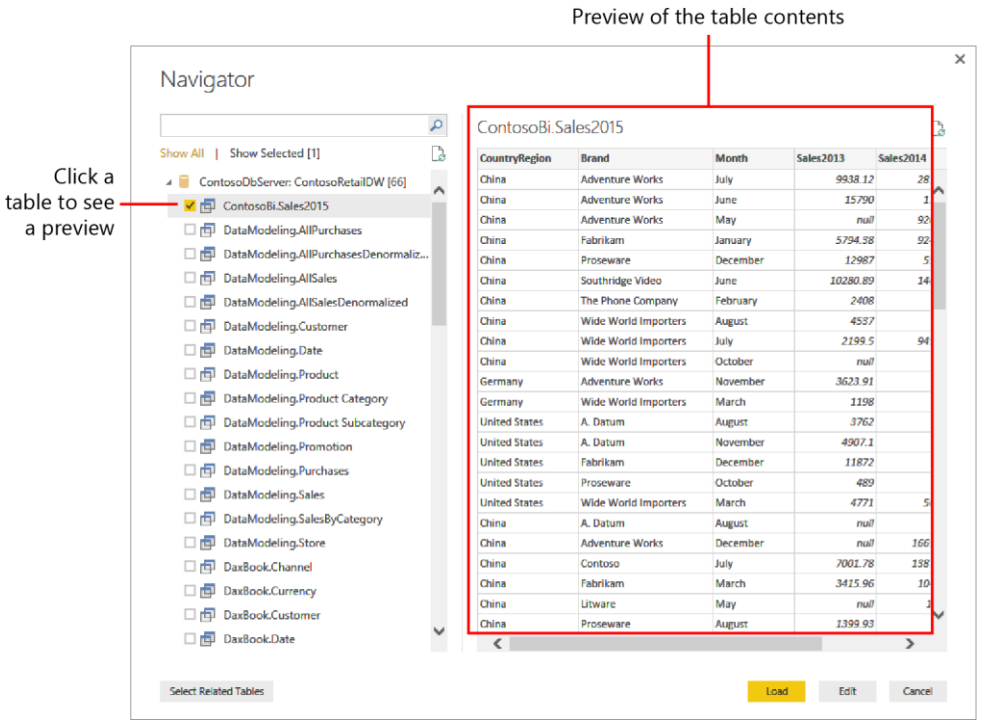


Figure 4-2: Use the Navigator to choose from among the tables available on the database.

After David makes his selection and clicks Load. Power BI Desktop asks how he wants to interact with the data. He selects Import and then clicks OK, as illustrated in Figure 4-3.

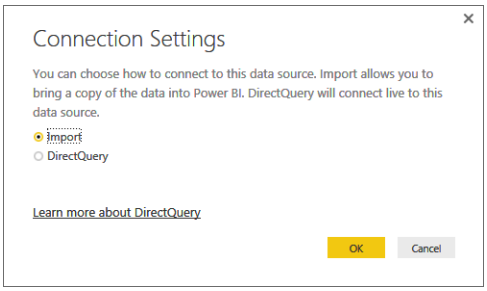


Figure 4-3: Before importing data from an SQL database, you need to choose the loading method.

Let's take a moment to learn about this connection option because it is an important one and will help shed more light on how Power BI connections work.

When you choose Import, Power BI Desktop connects to the database, loads the information, and stores it within its internal data model. You can then work on your data in Power BI Desktop without being connected to the database. You will only need a connection when you want to refresh the data.

With DirectQuery, Power BI Desktop does not load the data into its internal database. Instead, it runs a query to the original database every time it needs to draw a chart or, in general, run a query. Thus, the connection between Power BI Desktop and the database will be permanent.

The contrast in the query timings reflects a key difference: when you use Import, you are working with data that is only as current as the latest refresh, whereas with DirectQuery you always see the latest information available when you create the report.

At first glance, it looks like DirectQuery is the most convenient method for loading data, but this is not totally true. If the data is updated frequently, it is very likely that one minute you will see a report with

a set of figures, but when you open it again a few minutes later, the numbers might no longer be the same. This is frustrating if you are analyzing information over the span of an entire year (which is what David is doing). Numbers that change too frequently can become disturbing. Also, although real-time data might sometimes be useful, it comes at the cost of query speed; DirectQuery by its very nature is much slower than working with data that is resident on your device and directly accessible by Power BI Desktop.

As a final note, keep in mind that DirectQuery works fine when you use Power BI Desktop on your laptop, but when you publish the model to Power BI, the cloud service needs a way to communicate with the internal database server. This is accomplished by using the Enterprise Gateway, which is the advanced version of the Personal Gateway, to which you were introduced in Chapter 3.

> **Note** Because this is an introductory book, we do not discuss DirectQuery further, as it entails some deeper technical details. Yet, we consider it important, so we wanted to provide you with a fundamental understanding of the choice and its implications as well as the additional options that are available when you use DirectQuery.

As mentioned just a moment ago, David wants to analyze data over the span of a year, so he does not need his figure to be updated constantly; thus, he chooses Import and, after a few seconds, the loading process finishes. He notices that the name of the table is too long. In fact, Power BI Desktop used the full name of the table (including the schema name), but you can easily give it a new name by right-clicking it and then, on the shortcut menu that opens, select Rename, as demonstrated in Figure 4-4.

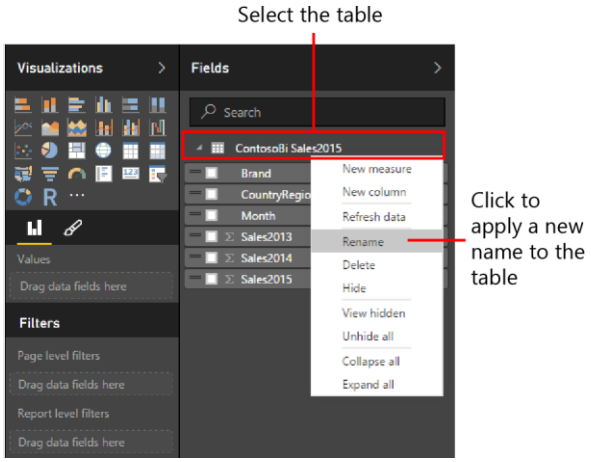


Figure 4-4: You can rename a table by using the Rename command.

After he renames the table, there seems to be no difference between this model and the previous ones that he created by loading data from Excel. In reality, there is a big difference: now, the Power BI model is linked to the original source of data, which is the SQL Server database. When Power BI Desktop refreshes the information, it does not need the Excel file (which if you recall was manually updated by David and Karin). Instead, by connecting directly to the database, it always gathers the latest information available at the moment of refresh. In other words, David eliminated Excel as a middle step, saving the time and effort required to prepare that file.

Loading from multiple sources

Working directly with a database looks great, but, after further investigation, David experienced an unpleasant surprise: by using Excel, he was able to integrate into the same table both the sales, which came from SQL Server, and the budget forecasts, which came from an Excel file. However, the Excel

file containing the forecast cannot be gathered from the SQL Server database, because the country/region managers update the Excel file whenever they want to share some new figures for 2016.

To solve this problem, we need to dive a bit more into the internal structure of the Power BI Desktop model. In Chapter 3, we said that a Power BI Desktop model contains an internal query, created by Power BI Desktop, for each dataset. This internal query is not visible if you perform basic operations, such as loading data from an Excel file or from a SQL Server database. Yet, it is there, and if you need to modify it, you can.

The query language of Power BI Desktop is used by Query Editor, and discussion of that language alone would fill an entire book of several hundred pages. As you might imagine, we cannot cover it in a mere few pages here. Instead, we want to show you some basic features of Query Editor so that you better understand its capabilities.

> **More info** If you are interested in learning more, we suggest that you to read one of the many good books about Query Editor. You can find them by searching for the M language or Power Query (Power Query was the previous name of the Power BI Query Editor).

To modify a Query Editor script, on the Power BI Desktop ribbon, on the Home tab, click Edit Queries, as shown in Figure 4-5.

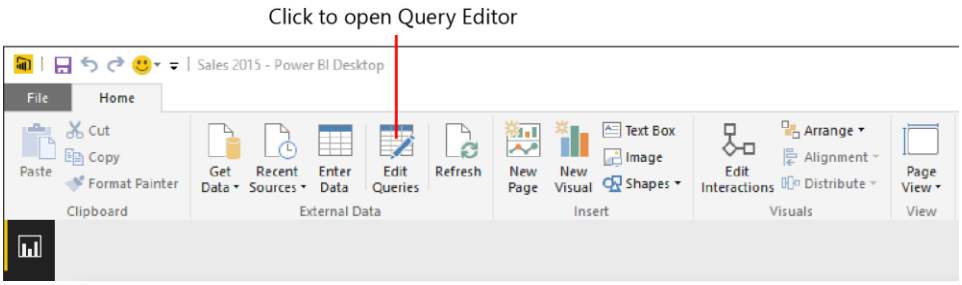


Figure 4-5: Click Edit Queries to access the Query Editor window.

Query Editor opens in a new window, presenting myriad options, as depicted in Figure 4-6.

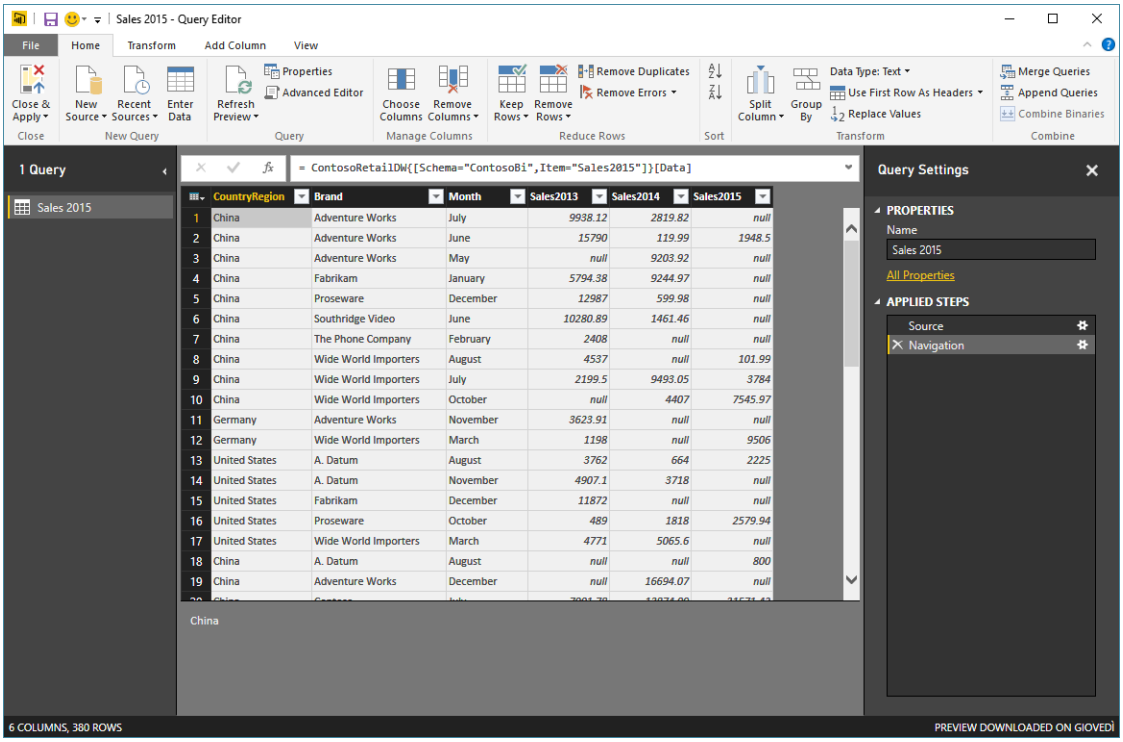


Figure 4-6: Power BI Desktop's Query Editor is a complete development environment in and of itself.

Let's take a quick tour of the Query Editor window. Along the top is the ribbon, which has four tabs: Home, Transform, Add Column, and View. Below the ribbon, on the left side, is the Query pane, which displays a list of all the queries for the model. The middle pane shows the result of the query. The Query Settings pane on the right displays the query properties.

In David's scenario, he is already accessing the 2015 sales data from the Contoso database, so the objective now is to create a new query that also retrieves the budget forecast information from the Excel workbook.

To load data from a new dataset, on the ribbon, in the New Query group of the Home tab, click New Source, and then specify to load the data from the Budget table. (The process is nearly identical to what you already learned in Chapter 3.) This results in two tables in the Queries pane on the left side of the Query Editor window, as demonstrated in Figure 4-7.

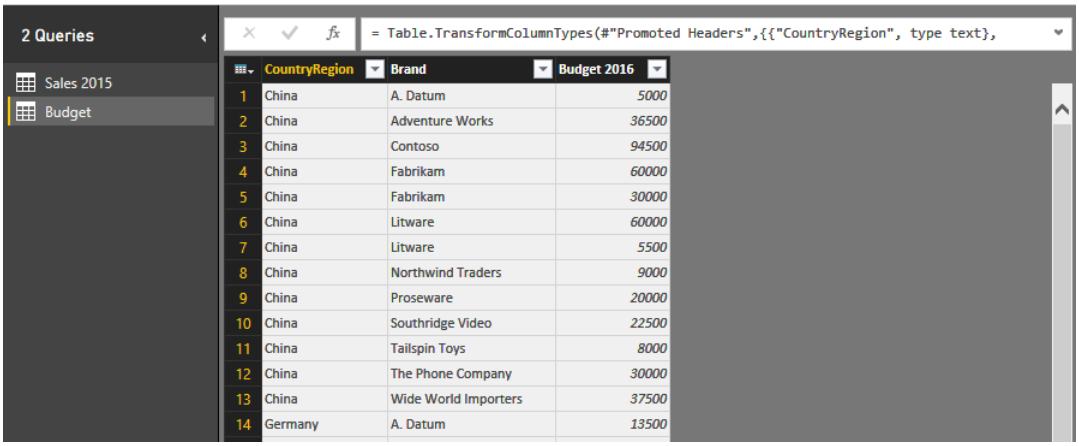


Figure 4-7: Query Editor can create multiple datasets, as you can see in this figure.

When you are finished editing, on the ribbon, on the Home tab, click Close & Apply to load the data into Power BI Desktop. When this is done, the Power BI Desktop Fields pane shows the two sources: the table in Excel, and the SQL Server table in the Contoso database, as depicted in Figure 4-8.

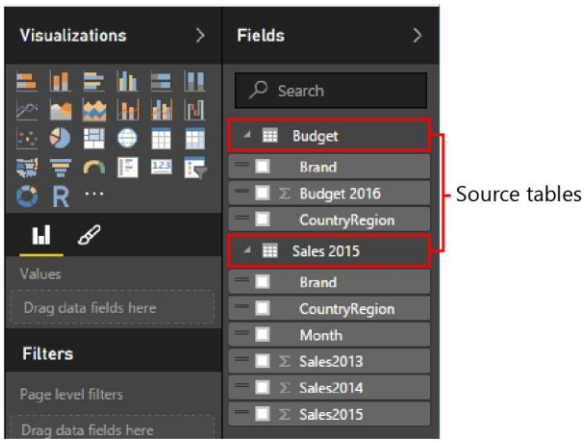


Figure 4-8: The Fields pane lists all of the tables (and columns) that the Power BI Desktop model is using.

Using Query Editor

At this point, David can build a report containing both the Budget and Sales 2015 tables sliced by the Brand column, for example. But, as shown in Figure 4-9, a bad surprise is awaiting him: the value of the budget is the same for all the columns.

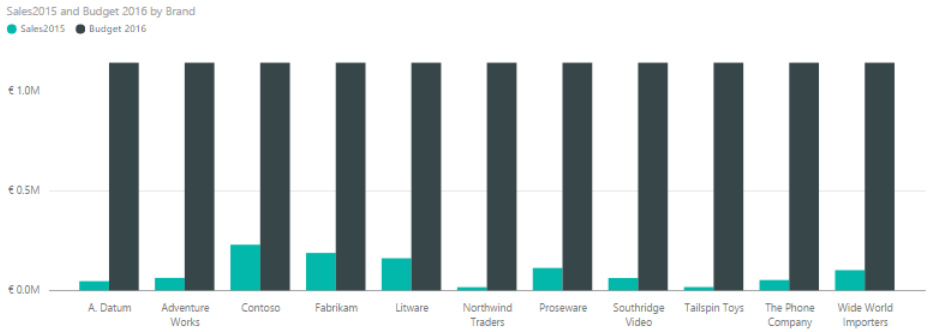


Figure 4-9: In this chart, the value for Budget 2016 is the same for all the columns, and it is too high.

The chart is confusing, at the very least. David used the Brand and Sales2015 attributes from the Sales2015 table, and the Budget 2016 column from the Budget table. Nevertheless, the values shown for the budget are always the same. Moreover, the value looks to be too high for each brand.

The problem here is that if you use the Brand column from the Sales2015 table, despite having the same name, it is not the same as using the Brand column from the Budget table. The two columns have the same values and the same name, but they are not the same column. In fact, if you were to try replacing Brand in the chart with the Brand column from the Budget table, the result would be similar, but with the opposite behavior. Figure 4-10 shows that by using the Brand column from the Budget table, the budget values are now correctly sliced, but the sales are not.

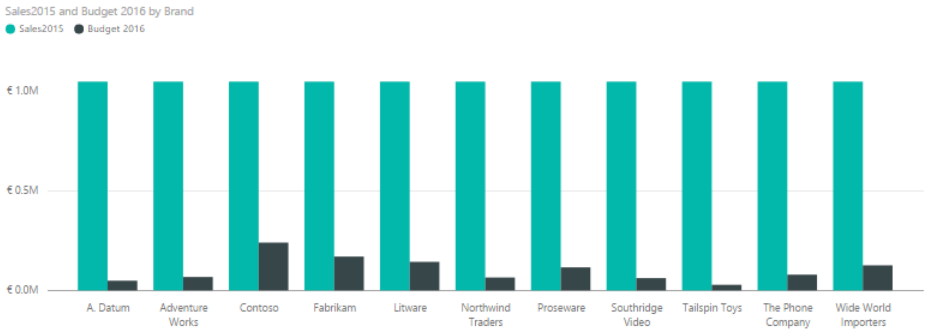


Figure 4-10: When slicing by Brand in Budget, the behavior is the opposite: sales are not sliced, whereas Budget is.

This would be a good time to digress and discuss what the correct data model to represent David's dataset is. It would be useful, but somewhat pedantic and not very relevant to the objectives of this book. The important point here is that you cannot slice numbers coming from two tables using columns from only one of them, unless the two tables share some kind of relationship.

> **More info** In this specific case, the two tables have no relationships. Moreover, a relationship cannot be created in an easy way: you would need to use a third table—in the middle—that can slice both of them. If you would like to learn more about this, read our book, *Data Modeling with Excel 2016 and Power BI* (2016, O'Reilly Media).

To solve the problem, you need to bring the Budget 2016 column from the Excel Budget table into the Sales2015 table, exactly where it was in the original Excel file. Using the technical terms, we say that you need to join the two tables together, copying the Budget column for the given country/region and brand. It turns out that Power BI Desktop's Query Editor is the perfect tool to perform such an operation.

In fact, with Query Editor you can load tables in an easy way, but, as we said earlier, you also have the option of editing the autogenerated query to make it behave differently. Let's catch up with David to see how he does this.

David goes back to the Power BI Desktop Query Editor window and modifies the Sales 2015 query. He selects the Sales 2015 query and then, on the ribbon, on the Home tab, he clicks Merge Queries (see Figure 4-11).

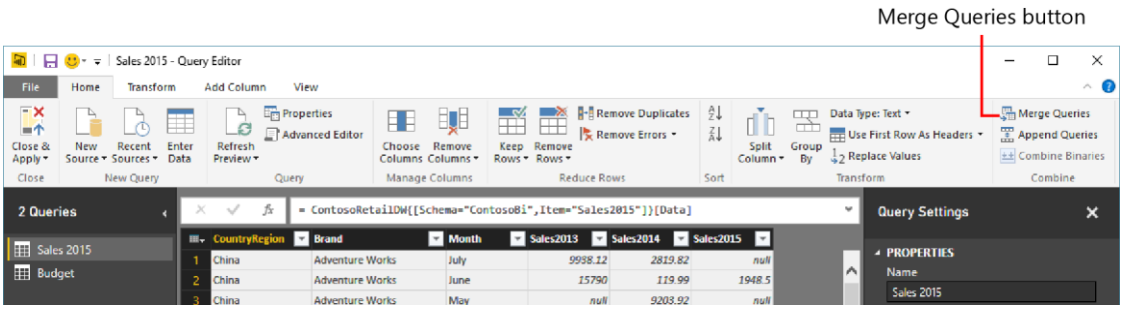


Figure 4-11: Click Merge Queries to join multiple queries into a single one.

This opens the Merge dialog box, in which you need to specify the destination table (the source is the one selected) and which columns to use to join them together. In David's case, the columns to use are CountryRegion and Brand, in both tables, as illustrated in Figure 4-12.

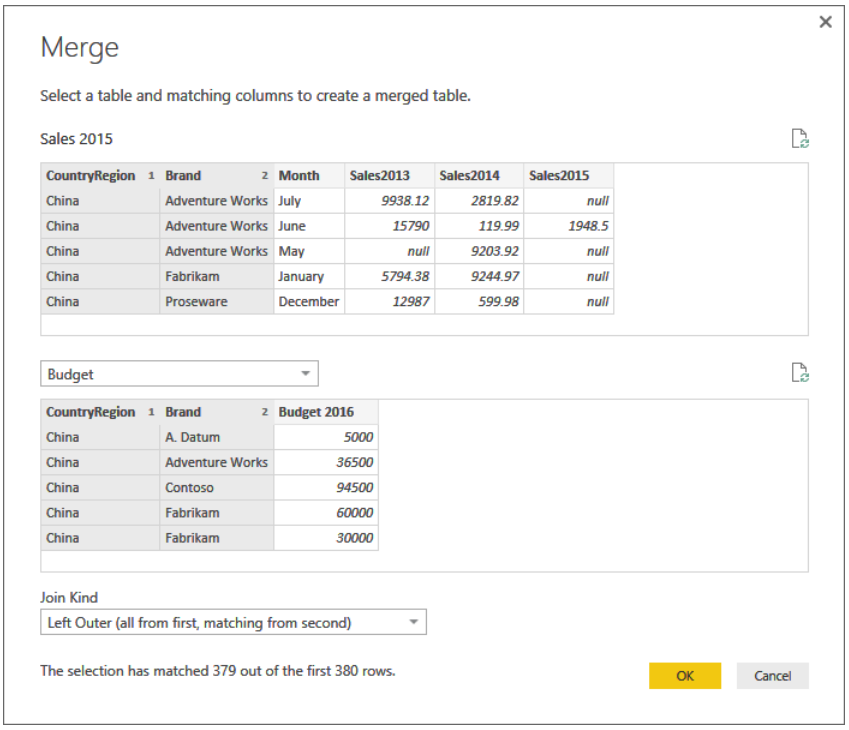


Figure 4-12: In the Merge dialog box, you can select which columns to use when merging two tables.

When you are selecting the columns to merge, Query Editor might display a dialog box similar to that shown in Figure 4-13, asking you for the privacy level of the data sources.

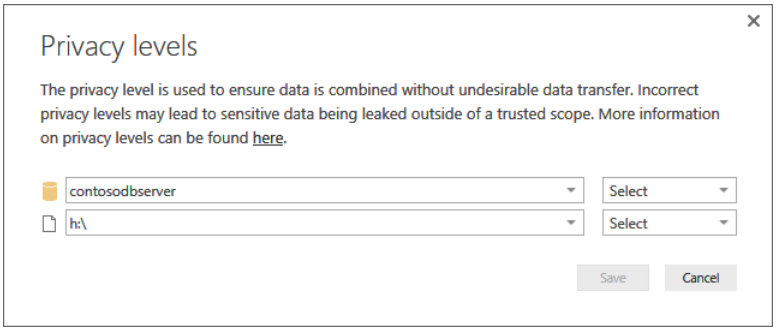


Figure 4-13: Query Editor needs to know the privacy levels of your data sources in order to merge them.

Privacy levels are used to ensure that you do not send private information to data sources outside of your secure area. Setting the wrong privacy level might expose sensitive data to untrusted sources or might affect the performance of the query. In David's case, he sets both sources to Private because both sources are within its network. If you are loading data from the web, for example, you should mark that data source as Public; this will avoid sending information to the web that is coming from one private source.

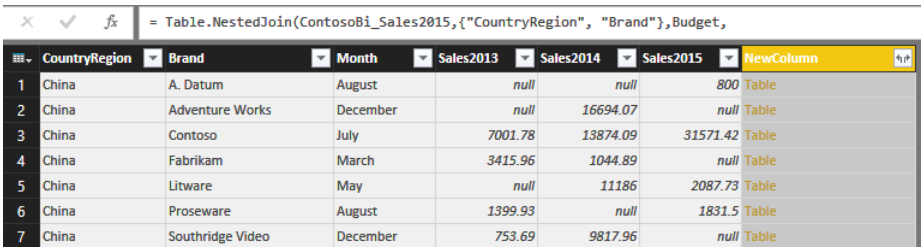
> **More info** A complete discussion of privacy levels is beyond the scope of this book. If you are interested in learning more about them, go to http://aka.ms/privacylevelspowerquery.

The last option available in the Merge dialog box is Join Kind. You use this to choose what happens with rows in one table that have no corresponding rows in the other table. For example, if there are sales for a country/region but there is no budget for it, should that country/region be included in the resulting dataset? The most typical kind of join is the default: a Left Outer join, which includes all the

rows from the source table and only matching rows from the merged one. In other words, in David's case, it retrieves all the sales, plus the budget for the countries/regions and brands with sales.

> **Note** In case there were new countries/regions or new brands, David should have used a Full Outer join so as to include both countries/regions and brands with no sales but with budget data, plus countries/regions and brands with no budget but with sales data. There are many kinds of joins, but the most useful ones are left outer and full outer. The remaining ones are somewhat exotic, interesting only for very technical people.

After David clicks OK, the table shows a new column named NewColumn, whose content is from a table, as shown in Figure 4-14.



```
= Table.NestedJoin(ContosoBi_Sales2015,{"CountryRegion", "Brand"},Budget,
```

	CountryRegion	Brand	Month	Sales2013	Sales2014	Sales2015	NewColumn
1	China	A. Datum	August	null	null	800	Table
2	China	Adventure Works	December	null	16694.07	null	Table
3	China	Contoso	July	7001.78	13874.09	31571.42	Table
4	China	Fabrikam	March	3415.96	1044.89	null	Table
5	China	Litware	May	null	11186	2087.73	Table
6	China	Proseware	August	1399.93	null	1831.5	Table
7	China	Southridge Video	December	753.69	9817.96	null	Table

Figure 4-14: When you merge two tables, the result is the original column, plus a new column of type Table.

In fact, when you merge two tables, the result is the original column, plus a new column of type Table. This new column contains all of the rows that are related with the current row in the original table. In David's case, the table contains a single row, but, for more complex joins, it might contain many rows.

David is not interested in retrieving the full table. Instead, he wants only the Budget column. To do this, he needs to expand the NewColumn table so that it includes only the columns he wants. He can do this easily by clicking the two arrows adjacent to the column name, in the yellow box around the column. This opens the expand column dialog box, as illustrated in Figure 4-15.

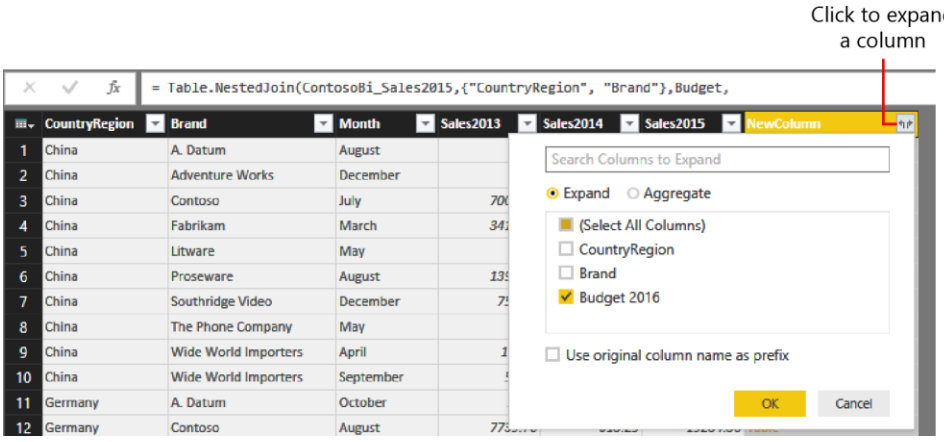
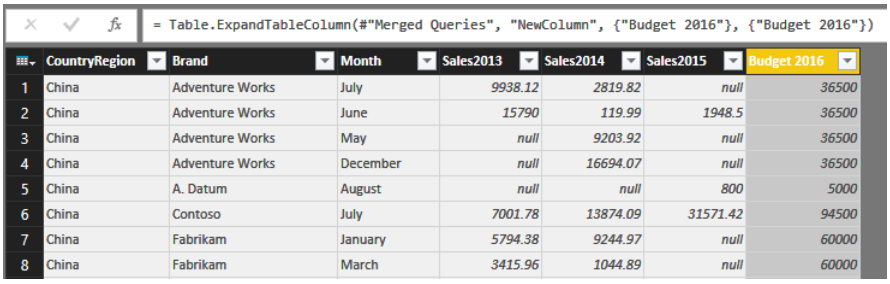


Figure 4-15: You can choose which columns to include in the result dataset in the expand column dialog box.

In the example, David selected only the Budget 2016 column. The result of this is that instead of NewColumn, you now have the Budget 2016 column for each row of the Sales table, as depicted in Figure 4-16.



	CountryRegion	Brand	Month	Sales2013	Sales2014	Sales2015	Budget 2016
1	China	Adventure Works	July	9938.12	2819.82	null	36500
2	China	Adventure Works	June	15790	119.99	1948.5	36500
3	China	Adventure Works	May	null	9203.92	null	36500
4	China	Adventure Works	December	null	16694.07	null	36500
5	China	A. Datum	August	null	null	800	5000
6	China	Contoso	July	7001.78	13874.09	31571.42	94500
7	China	Fabrikam	January	5794.38	9244.97	null	60000
8	China	Fabrikam	March	3415.96	1044.89	null	60000

Figure 4-16: When you expand a column, it is replaced with the columns you chose.

David is nearly done. The last step is that the column, as it is, shows the full yearly budget, whereas the Sales table should contain only the monthly budget. In Excel, David divided the budget value by 12; hence, he is doing the same here. To perform this, on the ribbon, on the Add Column tab, he clicks Add Custom Column (in the General group) to create a new column containing the value of Budget 2016 divided by 12, as demonstrated in Figure 4-17.

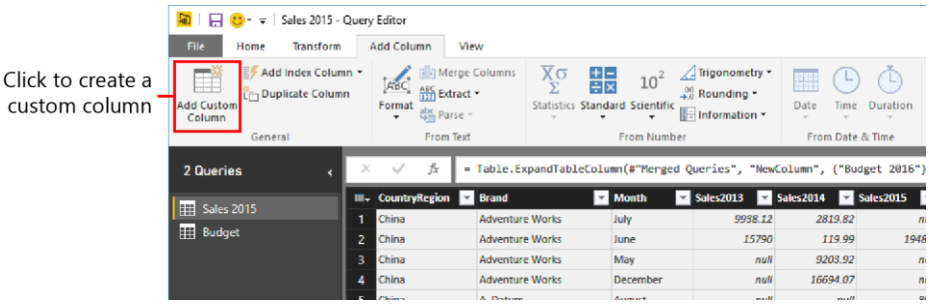


Figure 4-17: You can add new calculated columns to your query by using custom columns in Query Editor.

To create a new column, David must provide the expression that computes it. He can do that in the Add Custom Column dialog box that opens, as shown in Figure 4-18.

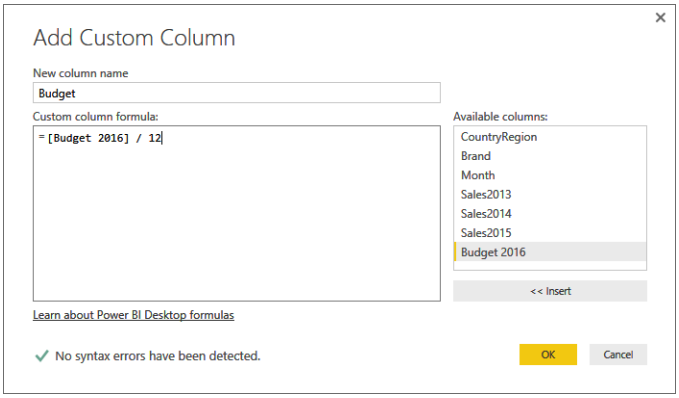
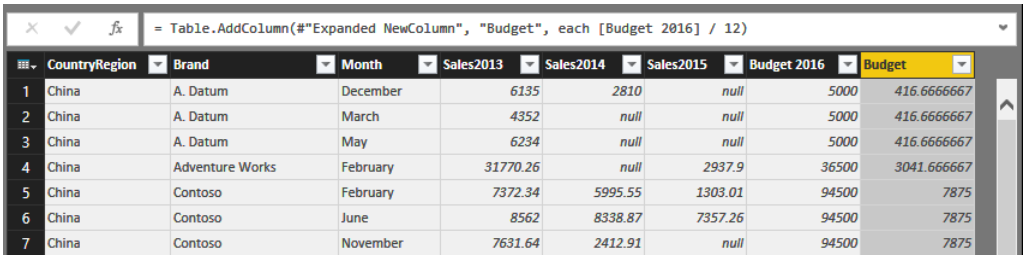


Figure 4-18: When you create a new column, you enter the formula in the Custom Column Formula box.

Figure 4-19 presents the result of all these steps, in which you can see the newly computed Budget column.



	CountryRegion	Brand	Month	Sales2013	Sales2014	Sales2015	Budget 2016	Budget
1	China	A. Datum	December	6135	2810	null	5000	416.6666667
2	China	A. Datum	March	4352	null	null	5000	416.6666667
3	China	A. Datum	May	6234	null	null	5000	416.6666667
4	China	Adventure Works	February	31770.26	null	2937.9	36500	3041.666667
5	China	Contoso	February	7372.34	5995.55	1303.01	94500	7875
6	China	Contoso	June	8562	8338.87	7357.26	94500	7875
7	China	Contoso	November	7631.64	2412.91	null	94500	7875

Figure 4-19: The new Budget custom column appears at the far right of the table.

As a final step, David right-clicks and removes the Budget 2016 column, which is no longer useful.

> **Note** You can delete columns in Query Editor even if they are used in other calculations. Unlike Excel, which saves formulas, Query Editor saves the steps of a calculation, and it will run them similarly to what you did by using the user interface.

Before saving the query, David needs to perform a final step: he defines the data type of the column. By default, custom columns are of the Any data type, meaning that the data type is not defined. But, because he wants to use it to aggregate values (which are, numbers), he must change the data type to Decimal Number, as shown in Figure 4-20.

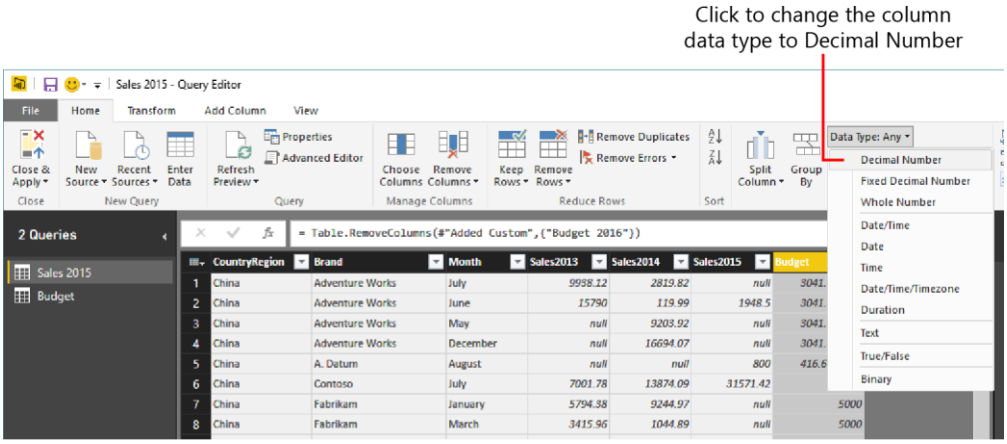


Figure 4-20: Changing the data type for the Budget custom column.

When this work is done, David ends up with a Sales2015 table that looks identical to that of its Excel counterpart. The big difference now is that the value of sales is computed from the SQL Server database and, when the model is refreshed, it will retrieve the latest figures in the Sales2015 table, with no manual intervention.

Hiding or removing tables

There is a last, small issue with this model. Using Query Editor, David moved the budget figures to the Sales2015 table to create a single table with all the columns required for his report. But, the Fields pane continues to display the Budget table. This might be confusing for Wendy and other people looking at the report.

You can resolve the issue in either of two ways: hide the Budget table from the Fields list, or avoid loading it altogether. To hide a table, in the Fields pane, right-click the table name and then, on the shortcut menu that opens, click Hide, as depicted in Figure 4-21.

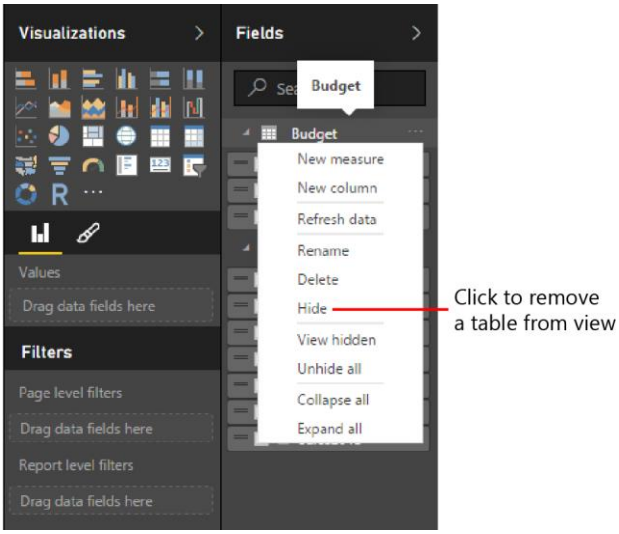


Figure 4-21: You can hide a table by right-clicking the table name and then selecting Hide on the shortcut menu.

A hidden table, as its name implies, is no longer visible in the Fields pane. You can always make it visible again by choosing View Hidden from the context menu of any table of the field pane and then clearing the Hide check mark. Keep in mind that hiding a table does not mean it is at all secure. A hidden table is only marked as not visible, but any user can see it by simply using the user interface; hiding a table simply makes the model easier to browse and less error-prone.

In David's case, he would prefer to avoid loading the table altogether. In fact, all of the information needed to build the reports is now stored in the Sales 2015 table. The Budget table is used by Query Editor to merge the budget (divided by 12) into Sales 2015. After the budget information is stored in Sales 2015, the Budget table is redundant and does not need to be loaded.

To avoid loading a table, in Query Editor, in the Queries pane, right-click the Budget query that you want to remove and then, on the shortcut menu, clear the check mark beside Enable Load, as shown in Figure 4-22.

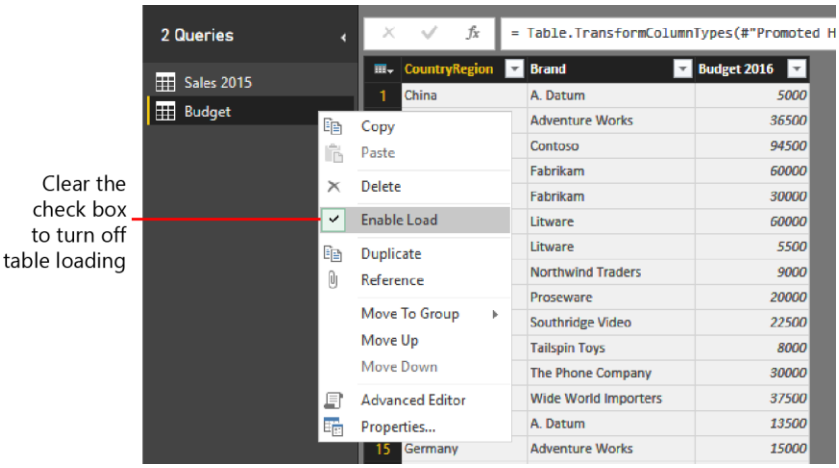


Figure 4-22: You can turn on or turn off loading for a table by using Query Editor.

When you turn off loading for a table, Query Editor warns you with the message shown in Figure 4-23.

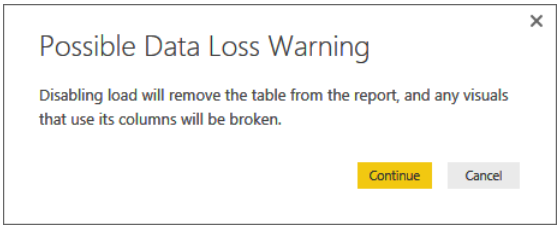


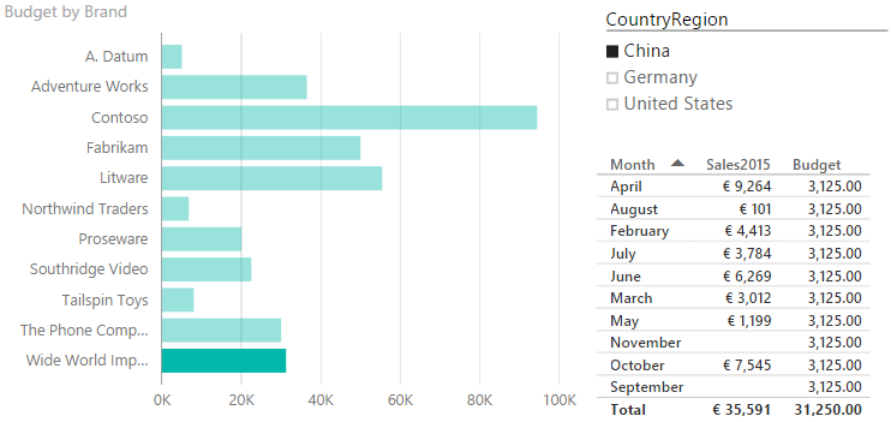
Figure 4-23: Before removing a table, Query Editor warns you about possible data loss.

If you continue, the table will be removed from the model, which becomes a single-table model again, with Sales 2015 containing all the relevant information.

Handling seasonality and sorting months

Recall from Chapter 2 that Wendy had some notes about seasonality. In fact, David splits the budget figures by 12, but, in reality, many brands show some seasonal effect that is not taken into account while computing the budget. Moreover, because some brands do not have sales at all in some months, the final report does not contain all the months.

For example, you can spot the problem easily by looking at the budget data report for Wide World Importers in China, as depicted in Figure 4-24.



Month	Sales2015	Budget
April	€ 9,264	3,125.00
August	€ 101	3,125.00
February	€ 4,413	3,125.00
July	€ 3,784	3,125.00
June	€ 6,269	3,125.00
March	€ 3,012	3,125.00
May	€ 1,199	3,125.00
November		3,125.00
October	€ 7,545	3,125.00
September		3,125.00
Total	**€ 35,591**	**31,250.00**

Figure 4-24: Some brands in China, for example, have no sales in November and September.

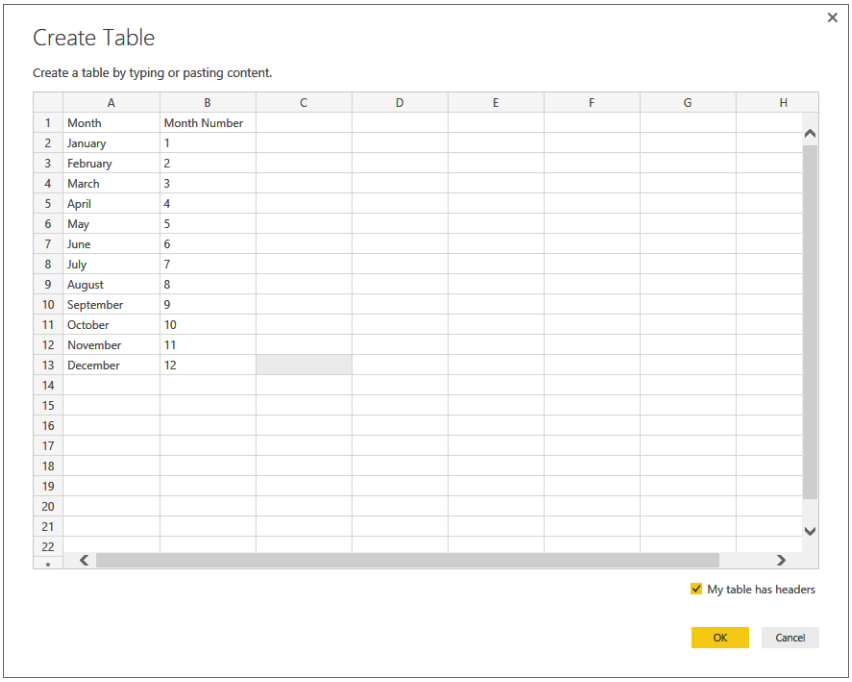
There are two issues with the report in Figure 4-24:

- Sales for January and December are missing from the tabular data. This is because there are no sales in January and December, so the corresponding rows are missing. This is a big issue, because the budget is computed as the total budget divided by 12, but only 10 rows are accounted for in the final figures. As a result, the budget values of the reports are wrong. In fact, the budget for World Wide Importers in 2016 was 37,500, whereas the report shows a total of only 31,250.
- While searching for the missing months, you might have noticed that months are not sorted in sequential order. In fact, by default, Power BI sorts each column alphabetically, which, of course, is not the correct way to sort months.

Note When you have a date column in the data and you use it in Power BI Desktop, the month name proposed is already sorted alphabetically. However, in this case, the data source contains the month name and not a date, so you need to correct the sorting order.

David is determined to solve these two issues, beginning with the last one, which is somewhat easier. To sort the months by sequential order, he needs a new column in the Sales table containing a number, ranging from 1 to 12, which contains the sort order of the month. The problem is there is no such column available, and there is no predefined functionality to achieve this goal.

If David's data were still in Excel, he could easily add the column to the table manually, but now data is coming from SQL Server, in the Contoso database, and he cannot modify the content of the SQL Server view to show such a month. Fortunately, Power BI Desktop offers you a great feature when you have some data to add to an existing model: you simply enter it. To do this, on the Query Editor ribbon, on the Home tab, in the New Query group, click Enter Data, and Query Editor shows you a grid in the Create Table dialog box, in which you can type (or paste) the data you want to add to the model. Figure 4-25 shows how David used this feature to create a table containing the month names and numbers.



	A	B	C	D	E	F	G	H
1	Month	Month Number						
2	January	1						
3	February	2						
4	March	3						
5	April	4						
6	May	5						
7	June	6						
8	July	7						
9	August	8						
10	September	9						
11	October	10						
12	November	11						
13	December	12						
14								
15								
16								
17								
18								
19								
20								
21								
22								

Figure 4-25: Using the Enter Data functionality, you can type or paste new datasets.

The next step, after saving and renaming the table as Month Numbers, is to bring the Month Number column from this table into Sales 2015. The technique you use is very similar to what David already did with the budget: join the Sales 2015 table with Month Numbers. This time, the relationship is based on the month name. Figure 4-26 presents the Merge dialog box already prepared.

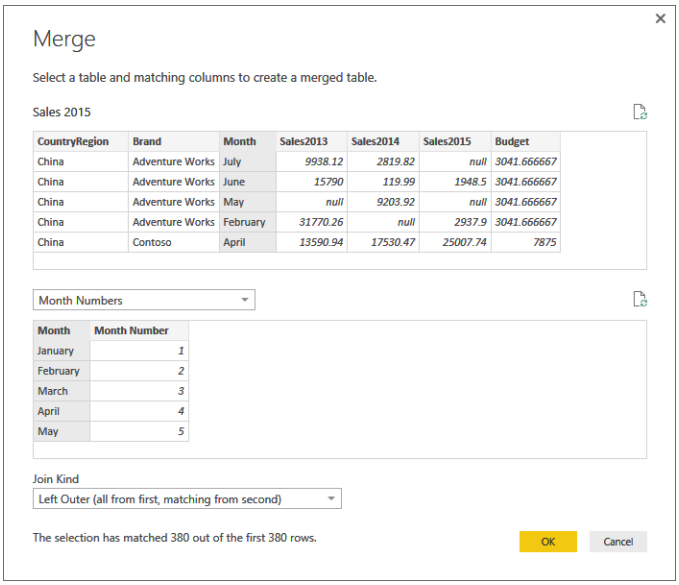


Figure 4-26: Merging the Sales 2015 table with the Month Numbers table.

After the merge, you still need to expand the Month Number column and load the content in the Power BI Desktop data model, similar to what you did in back in Figure 4-15.

Now that the month number belongs to the table, you need to instruct Power BI to sort the month names by month number. In the Fields pane, click the month name. Note that the ribbon displays a new tab: Modeling. On the Modeling tab, click Sort By Column (number 1 in Figure 4-27), and then select Month Number (number 2 in Figure 4-27).

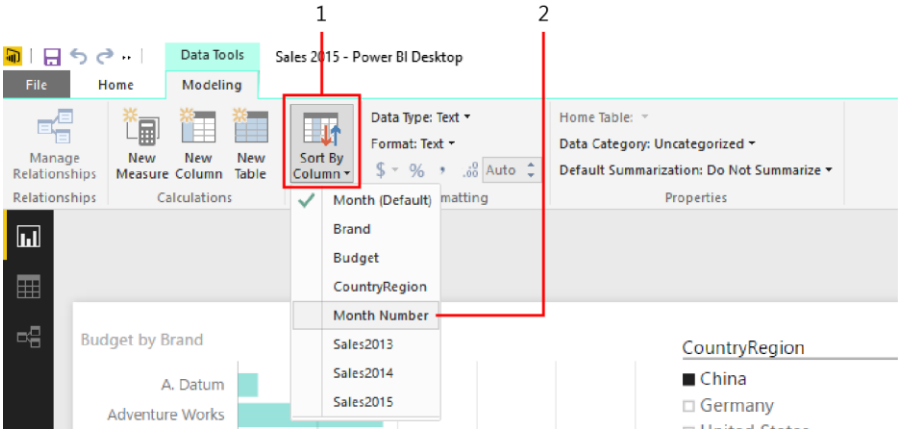


Figure 4-27: You can sort month names by numbers by using the Sort By Column feature.

After you select it, the report changes and shows the months correctly sorted, as demonstrated in Figure 4-28.

> **Note** It is a good practice to hide the column that you use to sort another visible column.

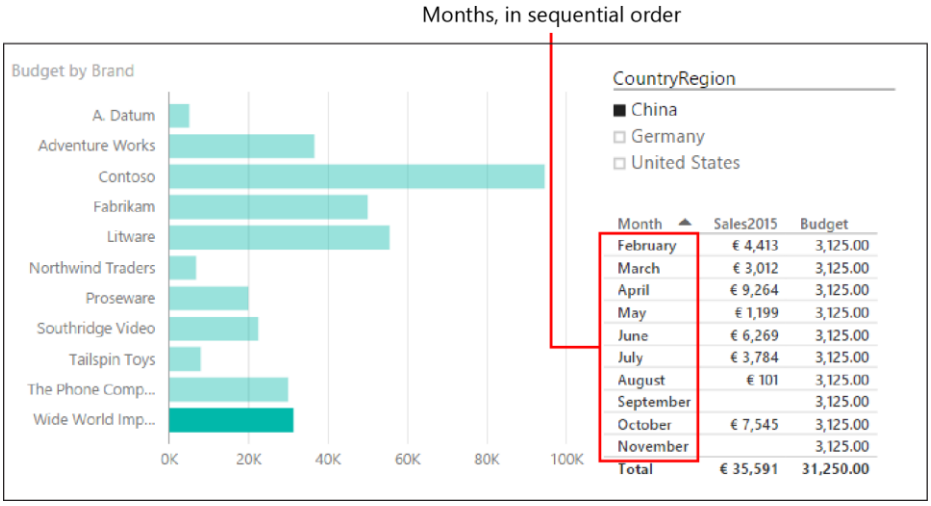


Figure 4-28: In this report, the months are now sorted properly.

Now that David has the months displaying correctly, it is much clearer that January and December are missing, so he turns his attention to fix this. Because this requires a bit more work, there needs to be a bit of fore planning.

First, David needs to decide which year to use as a basis for demonstrating seasonality. In fact, Wide World Importers might have sales in January 2014 and no sales in January 2015. Should he consider 2014 or 2015 to decide what to allocate in January? David goes for 2015 because it shows the best figures. You might make different decisions here, but as we have cautioned several times already, keep in mind that this is a book about Power BI, not a budgeting tutorial. So, please be patient; we are very naïve regarding choices like this one.

Now that David has made his decision, he needs to build a table containing, for each country/region and brand, the number of months for which there are sales in 2015.

This requires several steps in Query Editor:

1. Start from Sales 2015, and then remove all the unwanted columns, to keep only CountryRegion, Brand, Month, and Sales2015.
2. Remove all the rows in Sales 2015 that are empty.
3. Remove the Sales2015 column.
4. For each brand, count the number of months.

The first part of step 1 is easy: in Query Editor, right-click Sales 2015, and then, on the shortcut menu, click Duplicate to make a copy of the table. Name the copy Months Count.

The second part of step 1 is also easy: using the small delete icon that appears in the applied steps of the Query Settings panel (see Figure 4-29), remove all of the steps, keeping only the first two (Source and Navigation), so as to return to the original query.

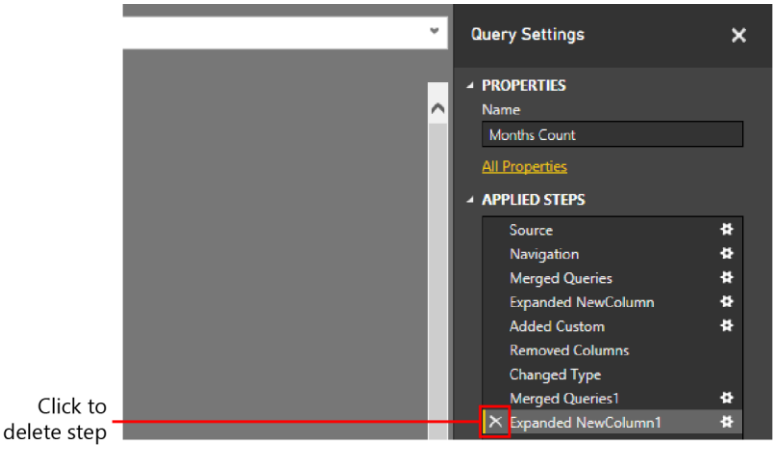


Figure 4-29: Use the small delete icon to remove unwanted steps.

Remember, we are working on a copy of Sales 2015, so we are free to update it as needed. The original table remains untouched.

At this point, you can delete the two Sales2013 and Sales2014 columns, which are not needed for the purpose of this query. To do this, right-click the column header for each column and then, on the shortcut menu, click Remove Columns. Step 1 is done!

Moving on to step 2: David notices that the first row (China, Adventure Works, July) contains a null value for Sales2015. He right-clicks the value to open its shortcut menu, where he chooses Number Filters and then Does Not Equal, meaning that he wants to filter only the rows for which Sales2015 is not null, as shown in Figure 4-30.

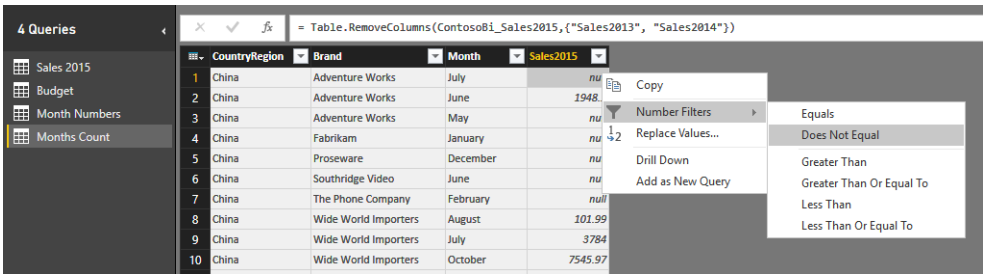


Figure 4-30: Right-click a cell value to filter rows using several criteria.

Step 3: At this point, the column Sales2015 is no longer useful; David can remove it as he did with the other two years. Steps 2 and 3 are now also done, in just a few clicks.

Finally, step 4: Group the current dataset by CountryRegion and Brand, then count, for each combination, the number of months. Because this is a very common operation on datasets, Query Editor offers a specific functionality: first, select the columns to group by, and then choose Group By, as illustrated in Figure 4-31.

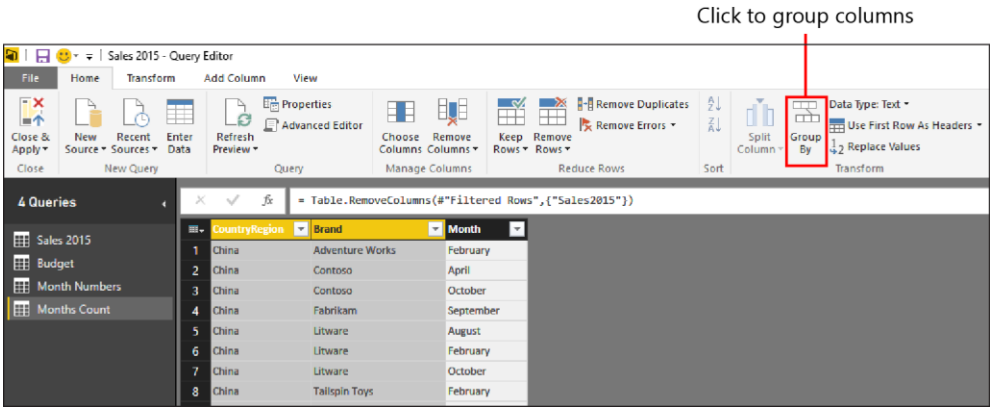


Figure 4-31: First, select the columns to group by, and then click Group By to open the Group By dialog box.

You use the Group By dialog box to choose the columns to group and the operation to perform on other columns. In this case, the default options are good (see Figure 4-32): David wants to group by CountryRegion and Brand, and then count the number of rows (which is, months) for each combination.

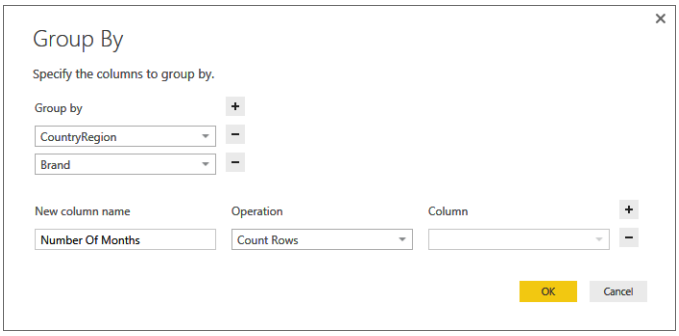


Figure 4-32: In the Group By dialog box, you choose the grouping parameters.

After he clicks OK to confirm this dialog box, David sees the dataset shown in Figure 4-33.

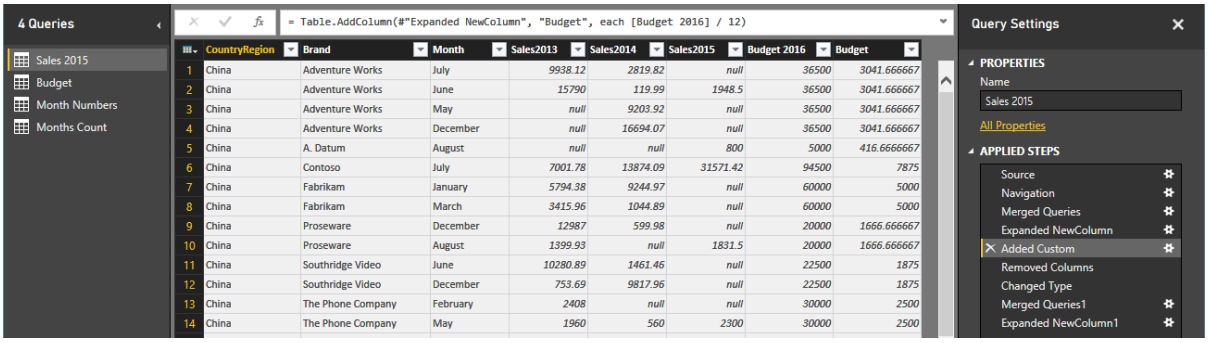
	CountryRegion	Brand	Number Of Months
1	China	A. Datum	4
2	Germany	A. Datum	6
3	United States	A. Datum	7
4	China	Adventure Works	7
5	Germany	Adventure Works	6
6	United States	Adventure Works	7
7	China	Contoso	10
8	Germany	Contoso	10
9	United States	Contoso	10

Figure 4-33: The final dataset contains the columns CountryRegion, Brand, and Number Of Months.

Now, the new dataset indicates how many months are present for each brand and country/region. You need to use this number, instead of 12, in the division of the budget to obtain the correct value to use for each month. Of course, you do not want this table in the model, so you turn off loading for it. This table is considered a *helper* table: you will use it with a join operation with Sales 2015. The table contains information that is useful only during the join operation, but not beyond that.

Thus, the last step is to modify Sales 2015 to take this number into account. This time, you do not need to add further steps to an existing query; however, you need to replace some of them, and this requires a bit more attention.

If you reopen the Sales 2015 query in Query Editor and begin navigating through the Applied Steps panel, you notice that what is displayed in the results pane reflects what the query looks like after having applied the selected step. For example, in Figure 4-34 you can see that when you select the Added Custom step, the query shows the Budget 2016 column, which will be removed by the next step.



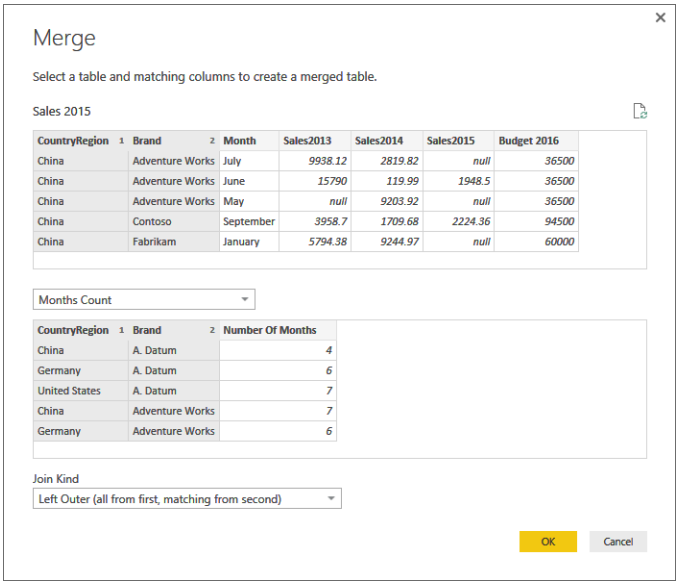
	CountryRegion	Brand	Month	Sales2013	Sales2014	Sales2015	Budget 2016	Budget
1	China	Adventure Works	July	9938.12	2819.82	null	36500	3041.666667
2	China	Adventure Works	June	15790	119.99	1948.5	36500	3041.666667
3	China	Adventure Works	May	null	9203.92	null	36500	3041.666667
4	China	Adventure Works	December	null	16694.07	null	36500	3041.666667
5	China	A. Datum	August	null	null	800	5000	416.6666667
6	China	Contoso	July	7001.78	13874.09	31571.42	94500	7875
7	China	Fabrikam	January	5794.38	9244.97	null	60000	5000
8	China	Fabrikam	March	3415.96	1044.89	null	60000	5000
9	China	Proseware	December	12987	599.98	null	20000	1666.666667
10	China	Proseware	August	1399.93	null	1831.5	20000	1666.666667
11	China	Southridge Video	June	10280.89	1461.46	null	22500	1875
12	China	Southridge Video	December	753.69	9817.96	null	22500	1875
13	China	The Phone Company	February	2408	null	null	30000	2500
14	China	The Phone Company	May	1960	560	2300	30000	2500

Figure 4-34: Navigating through the Applied Steps area of the Query Settings pane, you can view partial results of the final query.

You need to add a few steps before the Added Custom step (which computes the Budget, divided by 12) and then modify the calculation of the budget itself. When you choose an operation from the toolbar, the step is added right after the currently selected one. Thus, you select the fourth step (Expanded NewColumn) and, there, you add the merge of Sales 2015 with Months Count, basing the relationship on CountryRegion and Brand.

> **Note** When you insert a step into a query, Query Editor warns you about possible issues with the query. In this case, we do not have to worry, because we are not modifying the query behavior; we are only adding a new column that is coming from another query. There are scenarios, however, for which this warning makes sense. If you remove or rename a column that is used later, the query might break because of your changes.

Figure 4-35 presents the Merge dialog box for the inclusion of the Number Of Months column.



CountryRegion	Brand	Month	Sales2013	Sales2014	Sales2015	Budget 2016
China	Adventure Works	July	9938.12	2819.82	null	36500
China	Adventure Works	June	15790	119.99	1948.5	36500
China	Adventure Works	May	null	9203.92	null	36500
China	Contoso	September	3958.7	1709.68	2224.36	94500
China	Fabrikam	January	5794.38	9244.97	null	60000

CountryRegion	Brand	Number Of Months
China	A. Datum	4
Germany	A. Datum	6
United States	A. Datum	7
China	Adventure Works	7
Germany	Adventure Works	6

Figure 4-35: The parameters to bring the Number Of Months column into Sales 2015.

After you add the column to the view, you must replace the expression for the Budget 2016 column with a different one. In fact, you want to compute the Budget 2016 column divided by the number of months, for only the months for which there are sales in 2015. To perform this operation, in the Applied Steps area of the Query Settings pane, click the settings button (the small "gear" icon) to the right of the Added Custom step, and then change the expression accordingly, so that it appears like that shown in Figure 4-36.

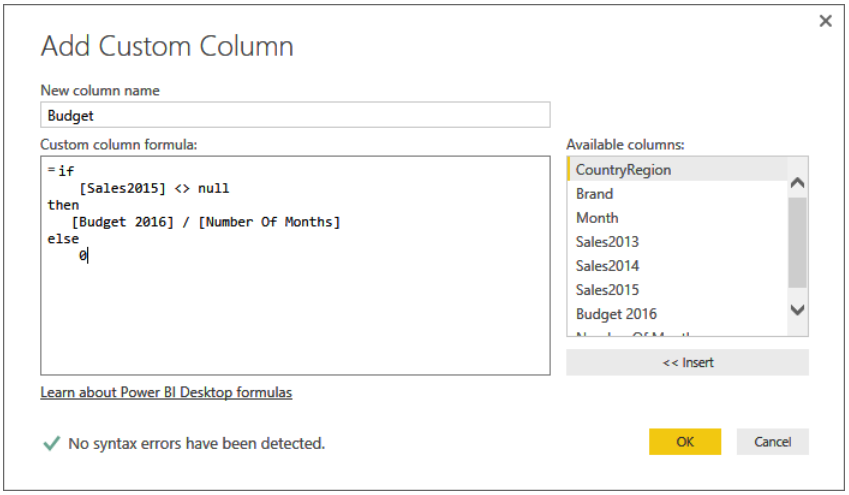
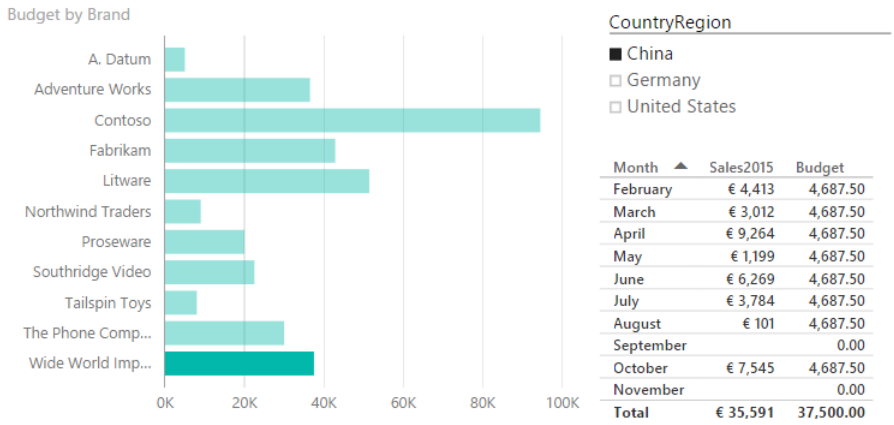


Figure 4-36: The new expression for the Budget column tests Sales2015 and divides Budget 2016 by Number Of Months.

With the new query in place, the report is updated as soon as you click OK, and now it shows correct figures, as demonstrated in Figure 4-37.



Month	Sales2015	Budget
February	€ 4,413	4,687.50
March	€ 3,012	4,687.50
April	€ 9,264	4,687.50
May	€ 1,199	4,687.50
June	€ 6,269	4,687.50
July	€ 3,784	4,687.50
August	€ 101	4,687.50
September		0.00
October	€ 7,545	4,687.50
November		0.00
Total	**€ 35,591**	**37,500.00**

Figure 4-37: The report now shows the correct value of 37,500.00 for the budget.

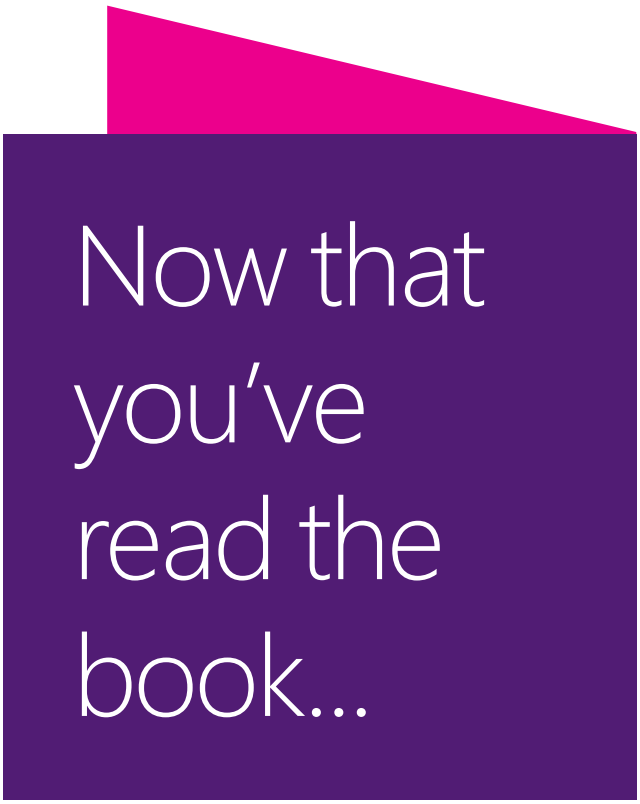
Conclusions

In this chapter, you learned the basics of Power BI Desktop, which is a desktop application that brings the full strength of Power BI to your PC. This was a basic introduction, and we will explore more features later in this book. In fact, there is a lot more to learn about Power BI Desktop, but that would be beyond the scope of this book.

Here are the most relevant features:

- Power BI Desktop can load data from any database. In the example in this chapter, we used Microsoft SQL Server.
- Using Power BI Desktop, you can load data from multiple sources. In the example, we mixed data from Excel with data residing in a SQL Server database.
- Power BI Desktop uses Query Editor to load data. Query Editor offers many powerful features. We highlighted in particular the capability of merging different queries and adding calculations to the query.
- Some queries are loaded into the model; others are useful only to compute values in the main query. You can mark queries that should not go into the model as "do not load" so that they are used only in Query Editor.
- You can upload a Power BI Desktop model to the Power BI online service, and it retains the same refresh features: by using the Personal Gateway, you can refresh a Power BI model in the cloud, letting it access data on your PC.

At the beginning, it might look complex, but after you get used to it, Query Editor offers you a lot of power to build your models. Having successfully finished this chapter, you can call yourself a data modeler. Later in the book, we will introduce some more features of Power BI Desktop to further enhance your model with more advanced calculations.



Tell us what you think!

Was it useful?
Did it teach you what you wanted to learn?
Was there room for improvement?

Let us know at http://aka.ms/tellpress

Your feedback goes directly to the staff at Microsoft Press,
and we read every one of your responses. Thanks in advance!

Microsoft

CHAPTER 5

Getting data from services and content packs

Working on the forecast for the next year, David realizes that it would be useful to consider the statistics for the pages visited on the company website. Using these insights, he might anticipate which products will gain more traction in the upcoming months.

Microsoft Power BI provides many connectors and content packs that make it possible for David to easily access the data generated from different cloud services. Content packs are also a useful tool to deploy and share predefined models and reports within a company.

In this chapter, you will see how David imports data from Google Analytics into Power BI, using different techniques. You can use these same techniques to access many web services, the list of which is growing rapidly, week by week.

Note In the following examples in this chapter, we will show data gathered by using Google Analytics from the website www.daxformatter.com, which we created to help users format DAX expressions and queries. We will use this data as part of the scenario we create for David. Our assumption is that the data would make sense in the sales generated by Contoso, too. But, also, we would not have enough data to show if we created a fictitious website for these examples. For this reason, when you continue reading, assume that visitors from this website are meaningful for sales of Contoso products, even if we know this is not true. If you want to replicate the same example, you can use Google Analytics data for a website to which you have access.

Consuming a service content pack

David wants to analyze data regarding customer visits to the website in an effort to glean some early indicators of a potential growth in certain countries/regions. If the number of visitors to the website increases in certain countries/regions, there also could be a growth in sales for the same country/region. Looking at this information for the previous two years might help in the budgeting process by providing data that will help David define the sales target for each country/region. Comparing website visitors and sales by origin in the same report is an important step in the analysis that he wants to accomplish.

David knows that his company's website is monitored by Google Analytics, and he wonders whether Power BI supports it. Reading the Power BI documentation, he finds that Google Analytics is indeed supported as a service content pack in the Power BI service, and as a connector in Power BI Desktop. With that, he begins using the service content pack in the Power BI service.

David starts Power BI and then, in the lower left corner of the window, he clicks the Get button. He is then greeted by the Get Data page, as depicted in Figure 5-1.

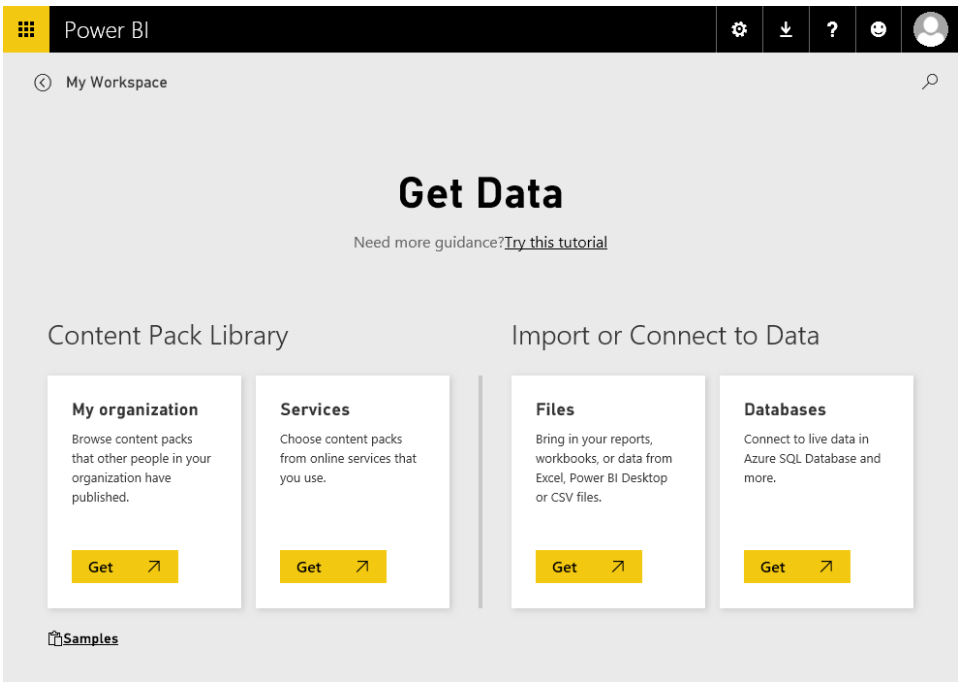


Figure 5-1: The choices available to you to get data in Power BI.

When you load data from a service, Power BI automatically creates for you a data source, a report, and a dashboard connected to that service, using predefined templates for each of these elements. You can modify these objects later if you want. Obviously, Power BI will use your credentials to access the service. Thus, even if the report is the same, the numbers will represent your own data.

What is a content pack?

A content pack can include the following parts, which have dependencies among them:

Dataset One or more datasets provide data to reports and dashboards included in the content pack library.

Report Each report in a content pack connects to a dataset included in the same content pack. The datasets required by reports are always part of the pack.

Dashboard Each dashboard of a content pack includes visualizations of one or more reports included in the same content pack. The reports used by a dashboard are always part of the pack.

David wants to use a content pack for a service; so, in the Content Pack Library section of the Get Data page, on the Services tile, he clicks the Get button. This displays a list of the services that are available, as shown in Figure 5-2.

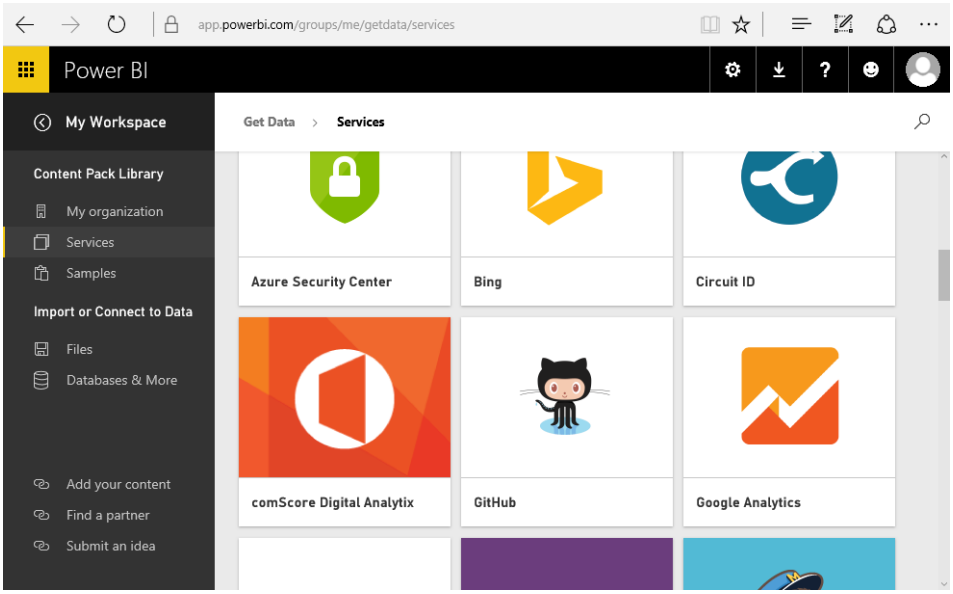


Figure 5-2: A partial list of the services that are available in Content Pack Library.

He clicks the Google Analytics tile and sees a message containing a description of the service, as illustrated in Figure 5-3.

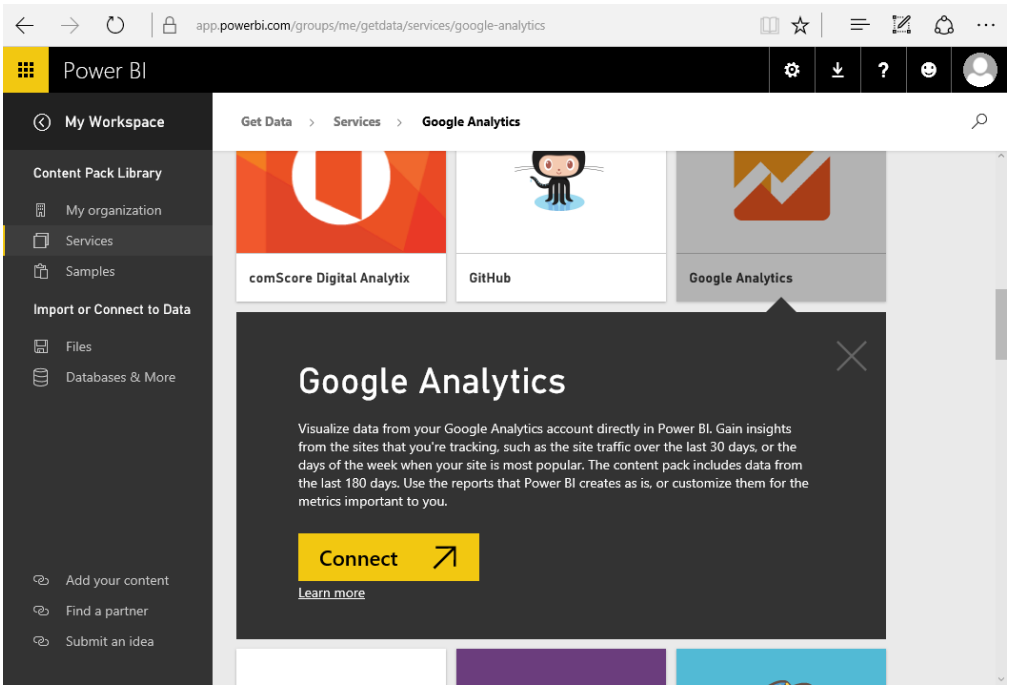


Figure 5-3: The description of the selected Google Analytics content pack.

> **Note** For all available services, Power BI presents a similar description along with a Connect button. However, the steps that follow after you click Connect might differ from those for Google Analytics, depending on the security model and the implementation of the authentication of the user for the selected process. So, keep in mind that if you select another service, the experience might be altogether different from what we describe in this chapter.

David clicks Connect and is then prompted to choose the authentication method to use to connect to Google Analytics. The only choice available to him is oAuth2, as depicted in Figure 5-4.

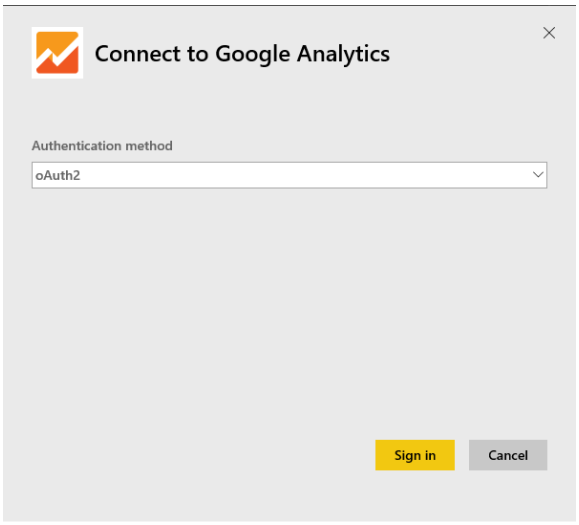


Figure 5-4: You must choose an authentication method when connecting to Google Analytics.

Be aware that after you click Sign In, you might be requested to sign in to Google Analytics (see Figure 5-5), unless your credentials are already stored because of previous access.

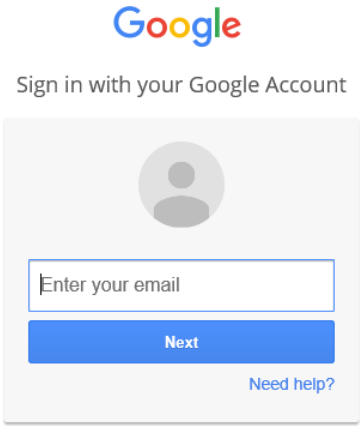


Figure 5-5: The Google sign-in page.

When David completes the sign-in, he is asked to provide offline access to the Power BI application, as shown in Figure 5-6. To authorize Power BI to retrieve data from the Google Analytics service on his behalf, he clicks Allow.

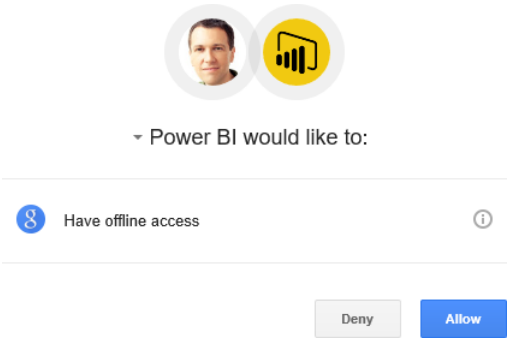


Figure 5-6: Confirming offline access to Google Analytics for Power BI.

> **Note** By allowing "offline access" you are indicating that you want to allow the Power BI Desktop application to interact with Google Analytics even when you are not interacting with the Google Analytics service directly. For example, when you navigate in the data offered by Google Analytics in a web browser, you are engaging in an "online access," because you directly interact with the service. However, when Power BI Desktop requests data from Google Analytics, it will act on your behalf, and you will not see the details of every request. This is true not only when you use Power BI Desktop interactively, but also whenever you schedule an automatic refresh of the dataset in Power BI.

On the next page of the Connect To Google Analytics dialog box, David can choose which part of the data available in Google Analytics that he wants to bring into the content pack. There are three options (see Figure 5-7):

- **Account** The account name for Google Analytics (a single user might have access rights to multiple accounts). Select from the list of available account names for the current user.
- **Property** The name of the property (which is a Google Analytics concept) within the data owned by the selected account.
- **View** The view name within the property. Oftentimes, the view corresponds to the property, unless you handle multiple websites within the same account.

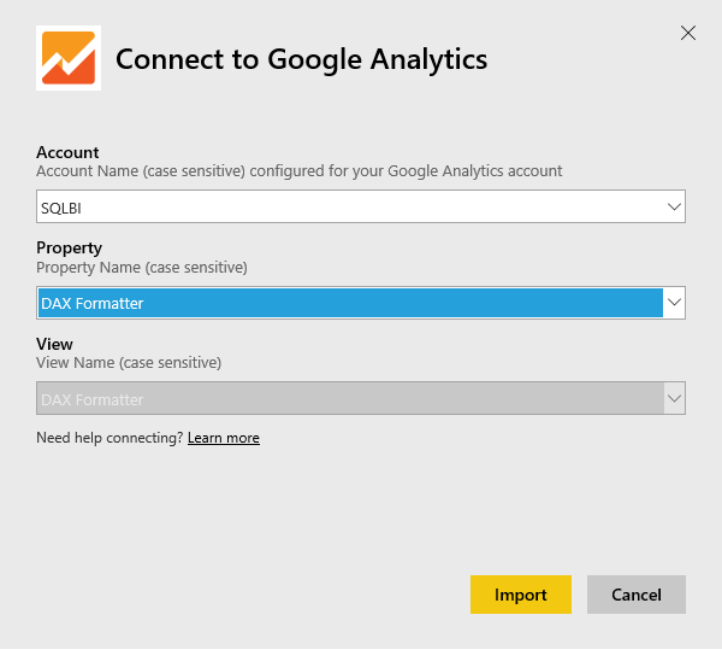


Figure 5-7: Choosing what part of the data to import from Google Analytics.

> **Note** In our example, for the DAX Formatter property, the View option is unavailable because there is only one view from which to choose.

When David clicks Import, the Power BI service copies the content library into the currently selected workspace, and then it updates the connection with the information provided in the previous steps. Finally, it populates the workspace with data read from the Google Analytics service. As a result, he obtains one dashboard, one report, and one dataset named Google Analytics, which he can rename if necessary (you should do that in case you import multiple copies of the same content pack in the same workspace—in this case, they would look the same, but they would contain different data).

The Google Analytics dashboard includes information about the traffic received in the past 30 days, as demonstrated in Figure 5-8.

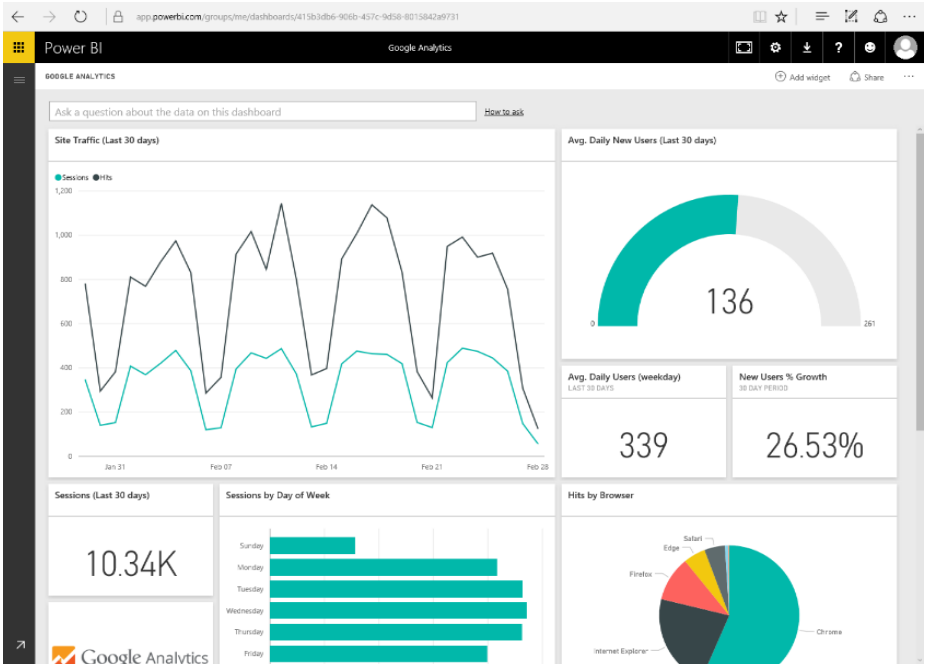


Figure 5-8: A dashboard created by the Google Analytics content pack.

Let's take a moment to discuss this dashboard. If you click one of the dashboard's visualizations, you are moved to the underlying data in a corresponding report. All of the visualizations come from the same report, which has pages that filter the data from the past 30 days (such as Total Users), the past 90 days (such as Site Traffic), or the past 180 days (such as System Usage, Page Performance, and Top Pages). Figure 5-9 presents the System Usage page for David's report.

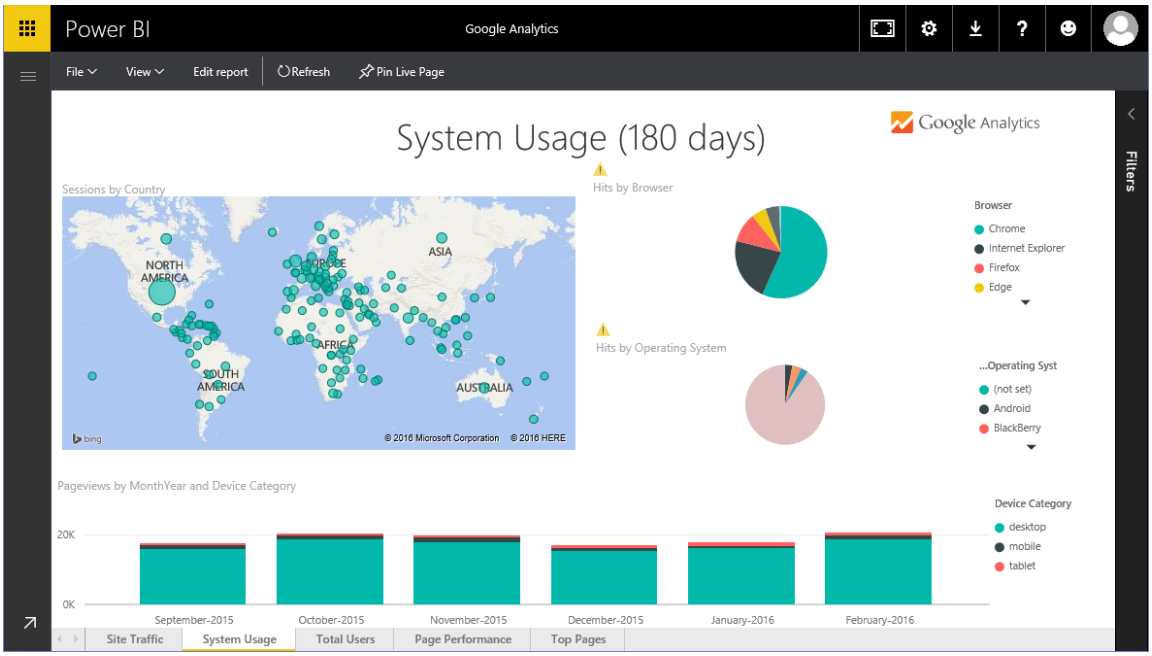


Figure 5-9: A report created by the Google Analytics content pack.

Back to David; he can now edit the report created by the content pack, or he can create a new report based on the same dataset used by the existing report. In both cases, one of the issues is that the dataset provided in this content pack does not include all the possible measures and slicers available

in the Google Analytics server. Also the available historical depth is limited to a maximum of 180 days. Figure 5-10 demonstrates that the number of tables, attributes, and measures available is just a fraction of the measures available in Google Analytics.

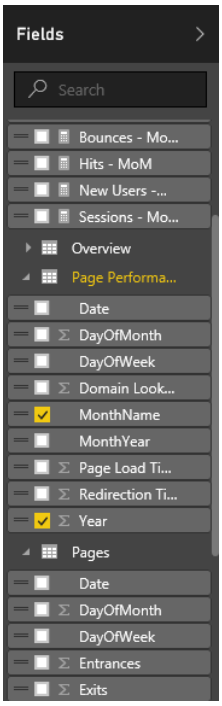


Figure 5-10: The tables, attributes, and measures available in the dataset of the Google Analytics content pack.

The reason for this is that the engineers who created this content pack tried to include the minimum amount of information required to create the desired reports. In this way, the size of the resulting Power BI file (.pbix) is kept to a minimum, improving the performance of many related operations. However, the level of details available might not satisfy your requirements, depending on what you are trying to accomplish.

The Google Analytics content pack has been useful for an initial overview of the data available, but it does not meet David's requirements. First, he needs greater historical depth to analyze the trends, and the limit of 180 days provided in the content pack is not sufficient for his requirements. Second, he would like to create a single report showing the relationship between data from Google Analytics and other data, such as past sales and forecast. This requires a single dataset with multiple connections, so getting data from the Google Analytics content pack is not very helpful, because you cannot change or customize the data model of a dataset copied from a content pack, whereas you can customize the reports and dashboard imported from a content pack.

For these reasons, David will create a new dataset using only the connection to Google Analytics in Power BI Desktop, without relying on the content pack he has used thus far.

Creating a custom dataset from a service

David wants to show in the same report some measures from Google Analytics related only to China, Germany, and the United States, which are the countries/regions interested in the budget process. He wants a better result than the one he can obtain by using the Google Analytics content pack on Power BI. Also, he needs greater historical depth than what is available through the service he tried earlier.

Instead of using the predefined (and read-only) content pack, David will use Power BI Desktop to connect to Google Analytics, and then include the Google dataset in the budget model he is developing. Of course, this requires some more effort, but it provides him with the advantage of flexibility in the definition of measures, calculations, new tables, and relationships in the data model.

Thus, David begins with the model he created by using Power BI Desktop in Chapter 4, which, to refresh your memory, has one table that contains the sales and budget in different columns. Such a table will be useful to analyze historical trends in sales. However, to import other tables from Google Analytics, he needs to use a special connector that imports data from Google Analytics directly into the model of Power BI Desktop.

David starts Power BI Desktop. On the ribbon, on the Home tab, he clicks Get Data and then selects the More option. In the Get Data dialog box that opens, in the pane on the left, he clicks the Other category, and then, in the pane on the right, he clicks Google Analytics, as shown in Figure 5-11.

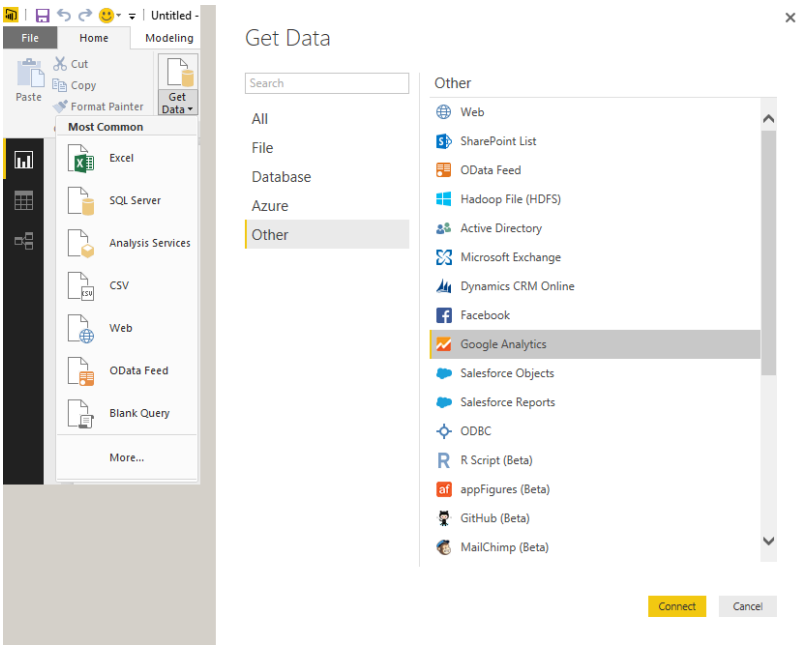


Figure 5-11: The Google Analytics connector is available in the Get Data dialog box.

When he clicks the Connect button, he is asked to sign in to his Google Account (Figure 5-12) so that Power BI Desktop will be able to access information in Google Analytics to which his Google Account has permission.

> **Note** You might see the Connecting To A Third-Party Service message box, warning you that the features, updates, and availability change often. Click Continue to close the message and move on to the next step.

When you sign in for the first time, you must provide your user name and password. (The sign-in process might use 2-factor authentication if required.) After you complete the sign-in process, you can click the Connect button to move forward.

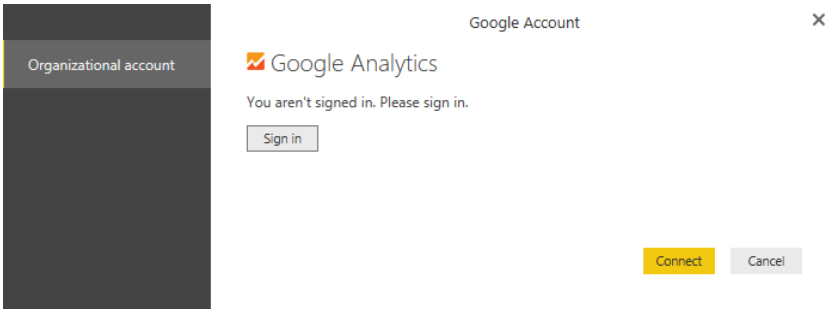


Figure 5-12: Connecting to Google Analytics from Power BI Desktop requires that you sign in to a related Google Account.

When David completes the sign-in process, he then needs to select the attributes and the measures to import in the data model. The Navigator dialog box provides a list of all the services monitored by the Google Account that he is using. After he selects the service, he is presented with a list of folders containing attributes and measures that he can select and import. The attributes are information collected by Google Analytics, such as date-related columns (year, month, day, etc.), demographic and geographical information about the visitors (such as age, gender, country/region, and city), and many other pieces of information that can be used to group and filter data. The measures are numeric information describing the frequency or the size of an event; for example, the number of users, the number of visits, the average time to load a page, and so on.

Querying a service such as Google Analytics is similar to querying a data model by using a pivot table. You select certain attributes, and measures are automatically aggregated at the granularity defined by the attributes included within the same report. For example, Figure 5-13 depicts the preview of the result obtained by selecting the Country/Region attribute from the Geo Network folder, the Year attribute from the Time folder, the Sessions measure from the Session folder, and the New Users and Users measures from the User folder.

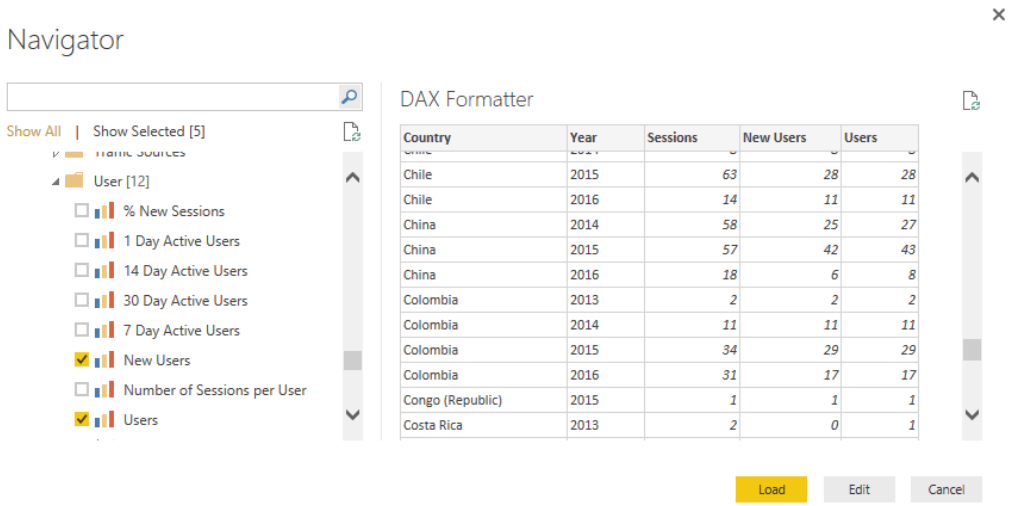


Figure 5-13: Connecting to Google Analytics, you select the measures and the attributes to import in the data model.

Choosing the right granularity

The approach for getting data from a service such as Google Analytics is different from the one you use when you collect data from a relational database (such as SQL Server) or from an external Excel file. In these cases, you always see the data at the maximum level of detail (also known as "fine granularity"). The selection of the attributes defines the granularity, and you can still change the query later in Query Editor, but you must add other attributes to the query in order to increase the granularity. You have a similar user interface in Query Editor when you import data from an external rich semantic model, such as the one provided by Analysis Services and SAP HANA.

In Power BI Desktop, on the Home tab, David clicks Edit Queries to open the Query Editor window. Because David wants to analyze a limited number of countries/regions, he needs to apply a filter in Query Editor to the Country/Region column, restricting the selection to China, Germany, and the United States. This obtains the result illustrated in Figure 5-14.

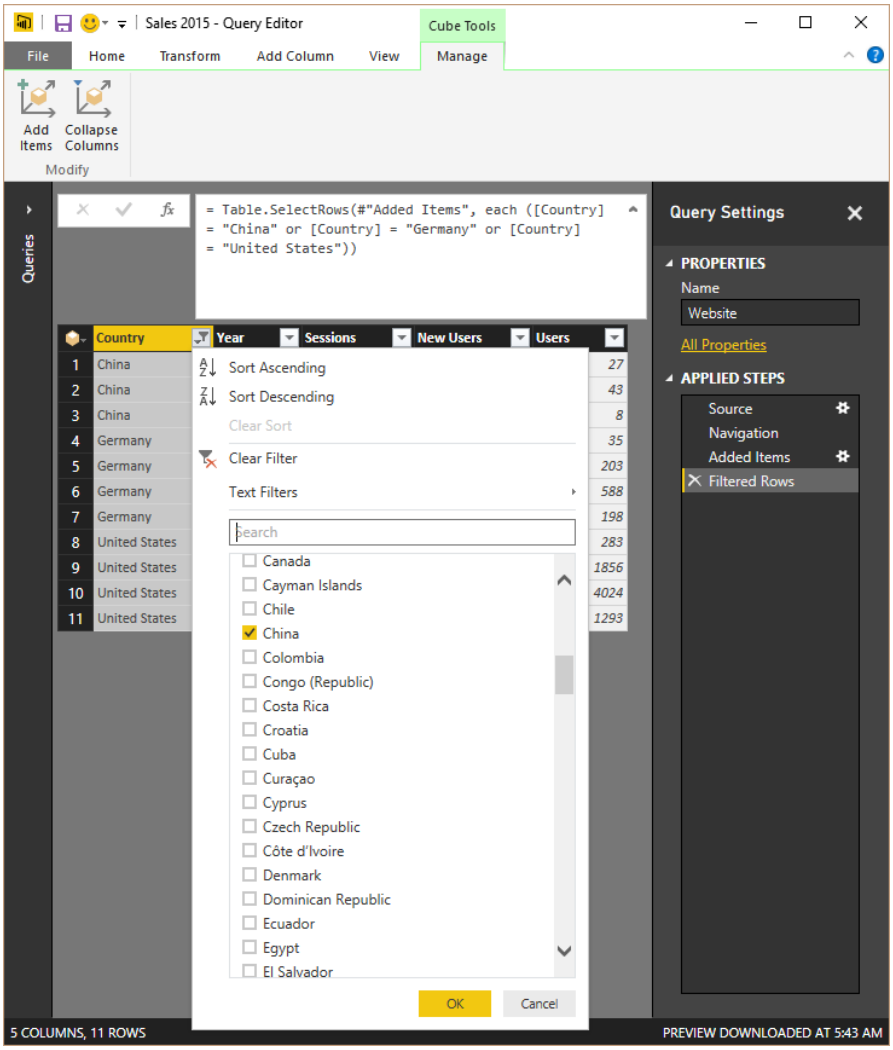


Figure 5-14: Using Query Editor to refine the query for getting data from Google Analytics.

Also note in Figure 5-14 that the ribbon includes a special tab: Cube Tools | Manage. The name "cube" references an external rich semantic model, and as we said earlier, it is the same approach available for Analysis Services and SAP Hana. When you click Add Items, you can select additional attributes and measures that will modify the query made to Google Analytics, enhancing the granularity (with

attributes) or more information (with measures). Figure 5-15 depicts the different graphical representation of attributes (in the Audience folder) and measures (in the DoubleClick Campaign Manager folder).

Note You should use the Collapse Columns button whenever you want to remove an attribute and obtain a corresponding granularity without the attribute you removed. If you just remove an attribute by removing the corresponding column in Query Editor, you do not modify the original query, and the cardinality will still include the attribute you removed. A detailed tutorial about how to use Query Editor with each data source is beyond the scope of this book, but you should be aware of this important difference compared to other types of data sources.

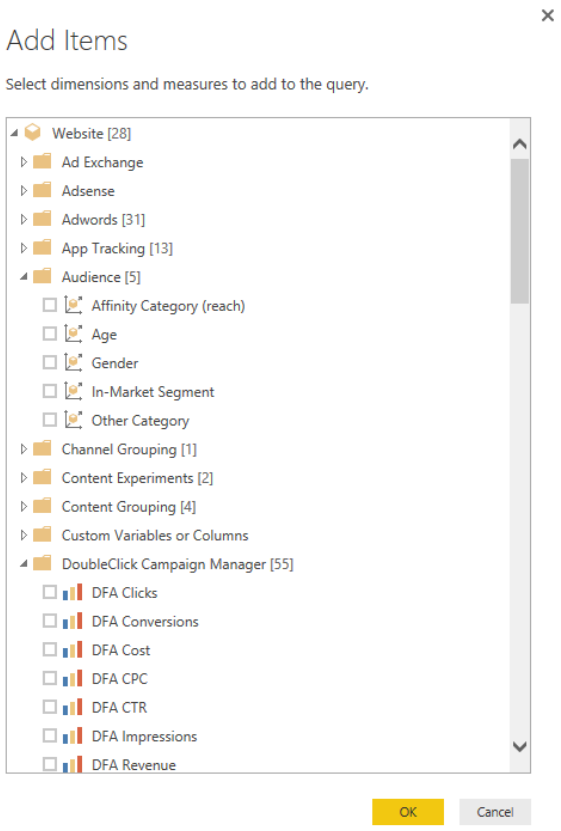
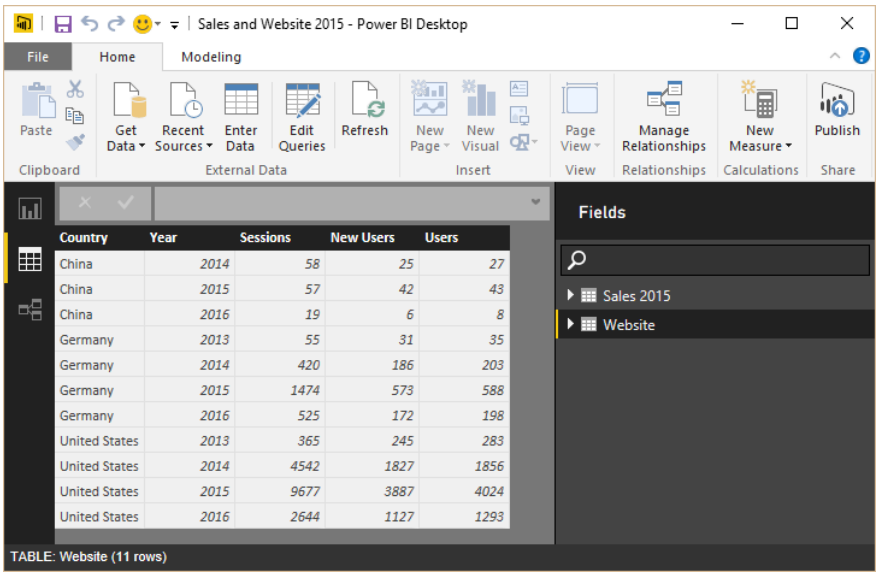


Figure 5-15: The Add Items dialog box in which you can add measures and attributes from a Google Analytics connection.

After David confirms the items to import, Power BI Desktop sends the query to Google Analytics and imports the result in a new table, as shown in Figure 5-16, in which you can see the content of the table named Website, containing the results of the query to Google Analytics defined so far.



Country	Year	Sessions	New Users	Users
China	2014	58	25	27
China	2015	57	42	43
China	2016	19	6	8
Germany	2013	55	31	35
Germany	2014	420	186	203
Germany	2015	1474	573	588
Germany	2016	525	172	198
United States	2013	365	245	283
United States	2014	4542	1827	1856
United States	2015	9677	3887	4024
United States	2016	2644	1127	1293

Figure 5-16: The table Website is the result of the query to Google Analytics.

> **Note** For the purposes of this scenario, our intrepid budgeting manager, David, does not create relationships between the table with data from Google Analytics and the other table in the same data model. However, it is common to create relationships between tables in order to simplify data navigation. You will see more complex data models in Chapter 6.

The key metric that David wants to obtain is the growth in new users that each country/region experienced in 2015. The data available in the Google Analytics service content pack did not have the historical depth required for this analysis; David requires data for both 2014 and 2015. The data imported in Power BI Desktop does not have such a limitation, so it is possible to import such historical depth. However, to create the calculation of the growth percentage, David has to learn the language used by Power BI, which is called DAX.

DAX, which stands for Data Analysis Expressions, was introduced in Power Pivot for Excel in 2010 and is based on the Excel formula language. If you have experience with Excel, you will find many functions that have the same name and syntax as those in your favorite spreadsheet. But, there are also several new concepts and functions that would require a separate book to cover fully. Fortunately, these books exist, such as *The Definitive Guide to DAX*, published by Microsoft Press. You will find a very basic discussion about DAX "measures" (the DAX term used to refer to scripts) in Chapter 6 of this book.

Because David used Power Pivot for Excel in the past, he already knows how to write the measure he needs. So, on the ribbon, on the Home tab, he clicks New Measure and inserts the following DAX measure in the formula bar:

```
New Users Growth =
IF (
    HASONEVALUE ( Website[Year] ),
    DIVIDE (
        SUM ( Website[New Users] ),
        CALCULATE (
            SUM ( Website[New Users] ),
            Website[Year] = VALUES ( Website[Year] ) - 1
        )
    )
)
```

Then, he displays this measure in a separate visualization, under the New Users metric, grouped by country/region and year, as illustrated in Figure 5-17.

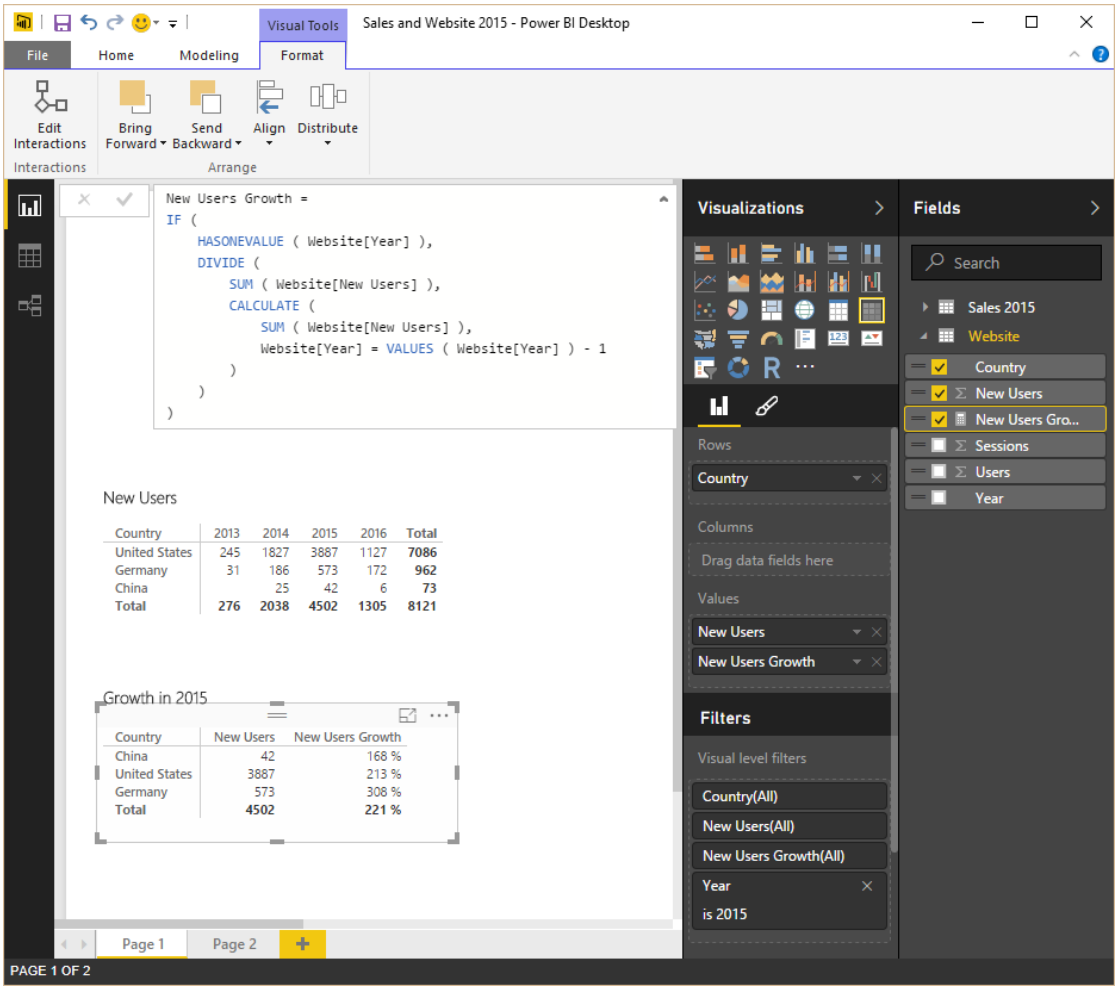


Figure 5-17: Two visualizations displaying the New Users measure and its growth, using data from Google Analytics.

At this point, David has the data he needed from Google Analytics, and he can consolidate that in a single report, together with data coming from other data sources. Chapter 4 shows you how to load data from different data sources, and Chapter 6 shows you how you can combine this data in a single model by using the DAX language. This helps to improve the browsing experience by providing a unique filter for each entity (such as the Country/Region in this report) instead of having similar columns in different tables that filter only the specific table to which they belong.

David can now publish the report he created in Power BI Desktop by using Power BI. By doing that, he is able to pin report content to a dashboard. Figure 5-18 depicts a dashboard built by pinning the two visualizations of the report he created. At this point, David has a dashboard and a report published on Power BI that are based on a custom dataset he created in Power BI Desktop, which gets a particular selection of data from Google Analytics.

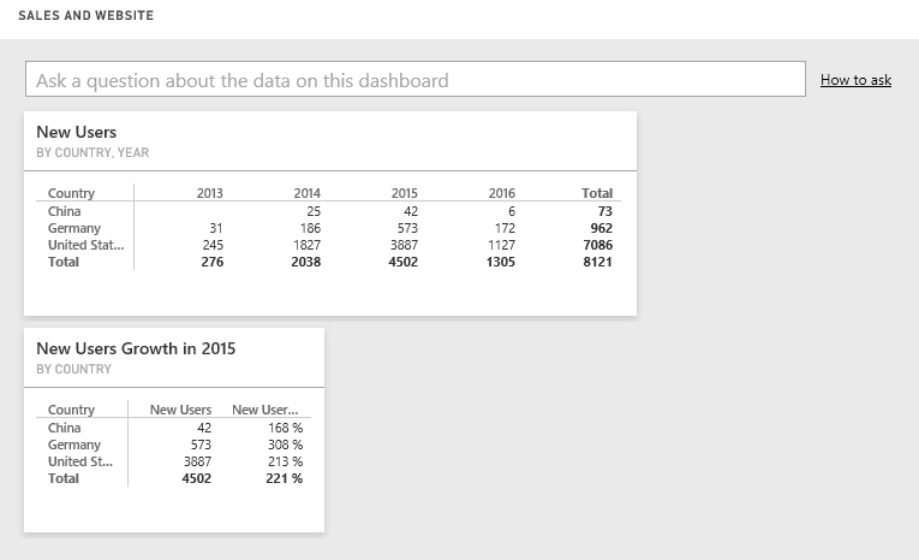


Figure 5-18: A dashboard that includes two visualizations displaying data from Google Analytics.

Creating a content pack for your organization

After David created a report in Power BI Desktop using the Google Analytics connector, he wants to share the result of his work with other colleagues in the company, and he wonders what the best tool is in Power BI to do that.

David realizes that the report he created would be a good starting point for deeper insights created by his colleagues. By using the share feature, he is able to share only a dashboard and its underlying reports, but he would like to publish the report based on Google Analytics as a template for reports created by other colleagues. The sharing options you have seen so far do not satisfy David's requirements. He wants other users to be able to customize the reports he made and create new reports based on his work. Sharing a dashboard does not provide such flexibility. Using the group workspace, there would be a single copy of reports and dashboards shared among the group's users, which would not be visible to users who are external to the group. However, David wants to share the results of analysis based on Google Analytics with other users outside the budgeting group. Thus, sharing a dashboard and creating a group workspace are not viable options.

The content pack for an organization is a good solution for David's requirements. This content pack can contain datasets, reports, and dashboards. Users receive a copy of these objects that are automatically synchronized in case a new version of the same content pack is published. If users customize one of these objects, they will work on their own copy of the reports, which will no longer be synchronized with the original one.

To create a content pack, in the upper-right corner, click the Settings button (the small gear icon), and then, on the menu that opens, select Create Content Pack, as shown in Figure 5-19.

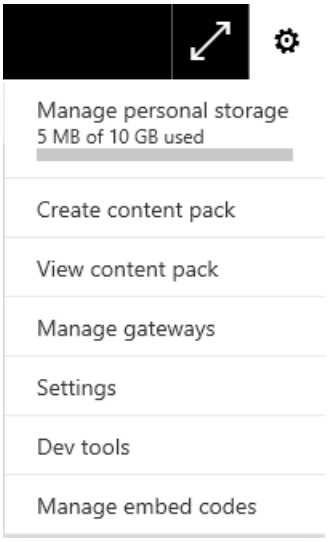


Figure 5-19: The Create Content Pack command on the Settings menu.

The Create Content Pack dialog box opens (see Figure 5-20). Here, David provides data about the new content pack. He can choose whether the content pack should be visible to any user in the organization or just to users belonging to specific groups. In this case, David selects the My Entire Organization option. He defines a title for the content pack (Sales And Website 2015, in this example) and a description of its content.

It is important to provide a clear description because this is what other users will read before importing a content pack in their own workspace. It is also possible to customize the content pack by using a specific image, which could be just the company logo or a more customized graphic. Selecting an image is optional; if you do not do that, a default one will be used, instead.

The more important part of the Create Content Pack dialog box is the area where you select the item to publish. There are three lists for of all the dashboards, reports, and the datasets you have in your personal workspace. You can select any number of the available entities, even if there are a few constraints. If you select a dashboard, all of the reports and datasets used in the visualizations of that dashboard will be automatically included, too. For example, in Figure 5-20 you can see that the dashboard Sales And Website is selected, and because of that, the report and the dataset named Sales And Website 2015 are automatically selected, as well. You cannot remove the selection of a dataset or a report if it is used in a dashboard. The same is true when you select a report: the underlying dataset is automatically selected, too.

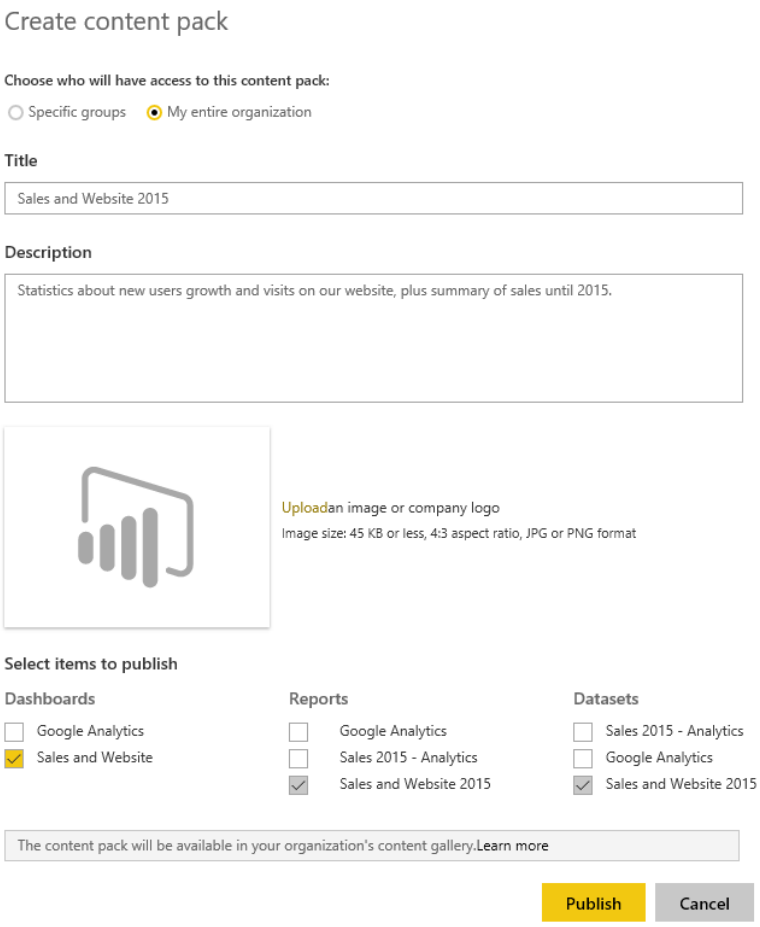


Figure 5-20: The Create Content Pack dialog box requires you to select objects to publish in a new content pack.

When you click the Publish button, the content pack is published and displayed in the list of the content packs that you can obtain by selecting the View Content Pack item in the Settings menu (refer to Figure 5-19).

Figure 5-21 presents David's list of content packs.

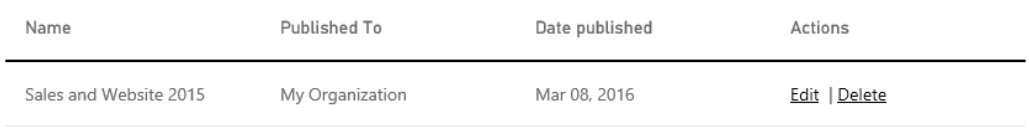


Figure 5-21: List of content packs published by the current user.

From this list, David can edit or delete each content pack. If he were to select Edit, he would return to the same configuration window shown in Figure 5-20; however, this time the window would be titled Update Content Pack instead of Create Content Pack.

At this point, other users in his organization are able to consume the content pack that David created. In the next section, you will see how this works and what the difference is between consuming an existing content pack as is and creating a personal copy that can be modified.

Consuming an organizational content pack

In this section, you will see how Wendy can use the content pack that David created.

When she clicks Get Data, Power BI opens the familiar Get Data page, shown in Figure 5-22.

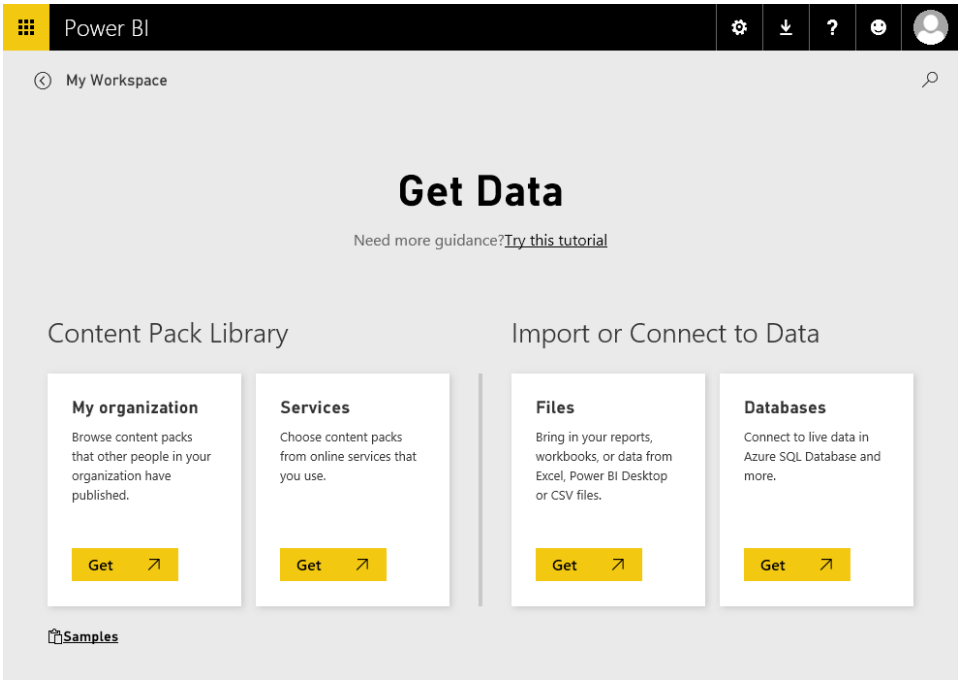


Figure 5-22: The Get Data page in Power BI.

In the Content Pack Library section in Figure 5-22, Wendy clicks Get on the My Organization tile. This opens the list of content packs she can use, along with a tile to create a new content pack, as demonstrated in Figure 5-23. In this scenario, the only content pack available to Wendy is the one David published, Sales And Website 2015.

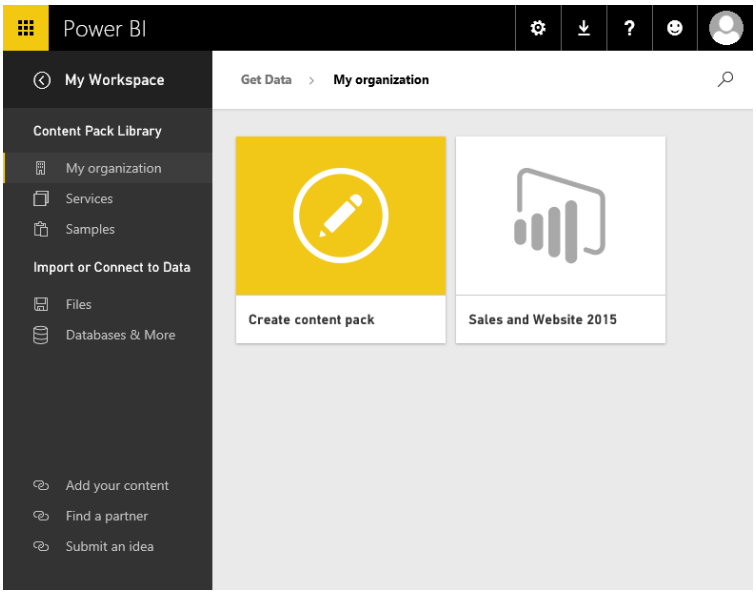


Figure 5-23: The list of content packs available in My Organization.

Wendy clicks the Sales And Website 2015 tile and then sees the information that David previously included when he published the content pack (refer back to Figure 5-20). This information includes a description of the content pack, the name of its publisher, and the time since it was last published, as shown in Figure 5-24.

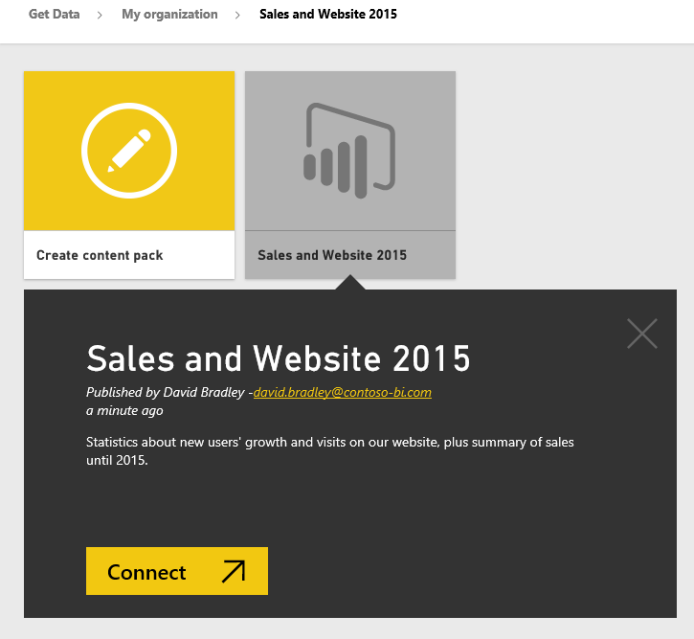


Figure 5-24: Details about the selected content pack.

When Wendy clicks the Connect button, Power BI imports into her personal workspace the entities included in the content pack. She will find a new dashboard (Sales And Website) has been added to the left pane, as well as a new report and a new dataset (both named Sales And Website 2015), as illustrated in Figure 5-25.

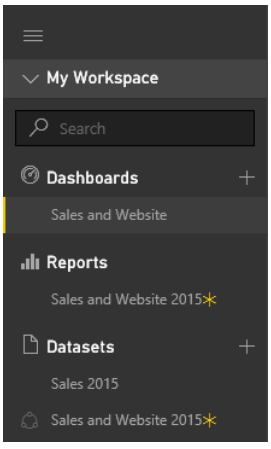


Figure 5-25: Entities included in the content pack that are imported into a personal workspace.

Wendy can navigate in the dashboards and in the reports in view mode without any issue. If the content pack were to be updated by David, the new version of datasets, reports, and dashboards included in the content pack would automatically replace those available to Wendy. However, if She tries to pin something more in the Sales And Website dashboard, or edit the report Sales And Website 2015, or if she clicks on the Sales and Website 2015 dataset to create a new report based on such a dataset, a message will appear (Figure 5-26), asking if she wants to personalize the content pack.

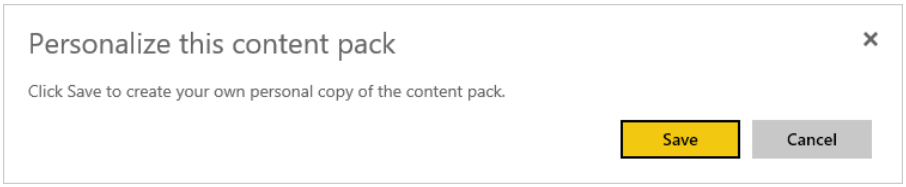


Figure 5-26: The request to personalize a content pack.

The same option is also available as the Personalize button when she clicks the option of a dashboard, report, or dataset that she obtained from a content pack, as demonstrated in Figure 5-27. These options also include the ability to remove or to open the object.

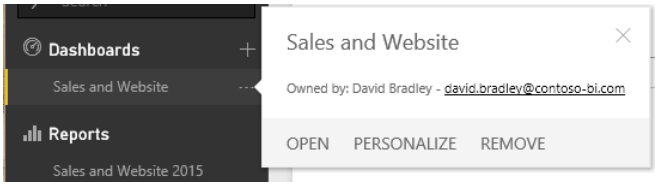


Figure 5-27: The options available in a dashboard obtained from a content pack.

If Wendy removes any object belonging to a content pack, all of the other entities from the same content pack are removed, as well. If she proceeds with the personalization and clicks Save in response to the message shown in Figure 5-26, she creates a copy of the objects in the content pack, and she is able to modify or delete them as she pleases.

In other words, if Wendy does not create a personal copy of the content pack, she will automatically receive any update to that same content pack that David publishes. If she creates a personal copy, any future updates by David will not propagate to her workspace. She will be able to modify the dashboards and the reports in her personal copy of the content pack; however, she will not be able to modify the dataset, because a dataset obtained from a content pack is always a shared dataset, and only the package owner can modify and refresh its content.

To recap, we have this following possible behavior for each object type:

- **Dashboards and Reports** These can be shared from the content pack, or they can be copied from the content pack in a personal copy. All of the dashboards and reports of a content pack are copied to the personal workspace if the user wants a personal copy of the content pack, but those dashboards and reports no longer receive updates from the content pack publisher.
- **Datasets** These are always owned by the content pack publisher, who is the only person who can schedule refresh operations and change other definitions in the dataset.

Updating an organizational content pack

David is the owner of the content pack named Sales And Website 2015. The moment he publishes the content pack on the Power BI service, any ensuing changes to any object included in the content pack on the Power BI service will generate a special notification. For example, if David changes the dashboard by moving the visualizations to different positions, he receives the warning message depicted in Figure 5-28.

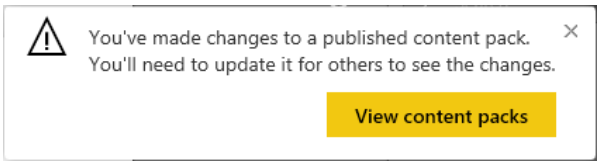


Figure 5-28: A warning message after changes are made to an object included in a published content pack.

> **Note** If you edit and then publish a content pack for a data model created with Power BI Desktop, subsequent changes made to the local data model (the .pbix file) do not automatically propagate to the Power BI service and to the published content pack. You do not receive any update in Power BI Desktop. It is up to you to remember that you must publish the .pbix file on the Power BI service to update the data model in the cloud.

If David clicks View Content Packs, in the list of the content packs he published, he sees which ones are affected by one or more changes he made to the reports and dashboards. Figure 5-29 shows that the only content pack he published, named Sales And Website 2015, displays a warning icon next to the name. When David points to the icon, he can see the content of the warning message, specifying that the content pack must be updated in order to show the changes to other users.

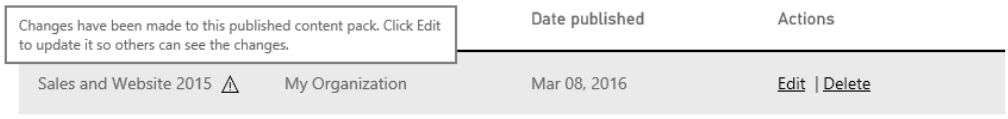


Figure 5-29: A content pack is marked with a warning icon when it includes unpublished changes.

In practice, any changes made to objects that are a part of a content pack are not automatically published in a new version of the content pack until the owner specifically performs such an update. When David clicks the Edit action, he can publish a new version of the content pack, which will include the current version of the objects (dashboards, reports, and datasets), replacing the version previously published with the same name. Figure 5-30 shows that the Update Content Pack dialog box is identical to that of Create Content Pack, the only difference being that the Publish button is now labeled Update.

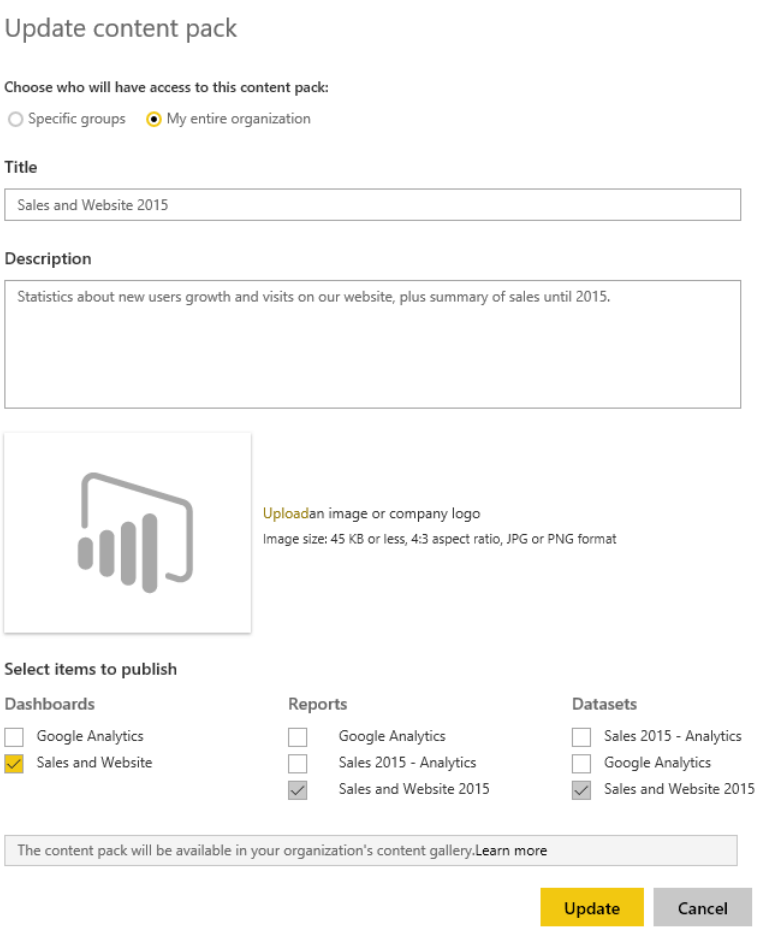


Figure 5-30: The Update Content Pack dialog box is the same as the Create Content Pack dialog box, except the Publish button is now labeled Update.

When David clicks Update, he generates a new version of the content pack that overrides the previous one. This action automatically updates all of the objects within the content pack in all the workspaces of any users who have consumed the same content pack without any customizations. This type of consumption of a content pack corresponds to the share feature of a dashboard, which provides a read-only copy of the dashboard that other users can see but not modify. However, using the content pack, this capability is extended to reports and datasets, which can be a part of a content pack regardless of whether they're used in a dashboard.

If a user who consumed the content pack created her own copy of it, a warning message will display in her workspace, notifying her that a new version of the content pack is available, but that is all that will happen, because changes in the consumed content pack are no longer automatically published to the corresponding entities in her content pack. For example, earlier we saw that Wendy created her own copy of the content pack, so when David publishes the new version of the content pack, she does not see any changes applied to the Sales And Website dashboard. However, because she still uses some objects that were originally created from a content pack that now has been updated, she receives the message shown in Figure 5-31 in her Power BI window.

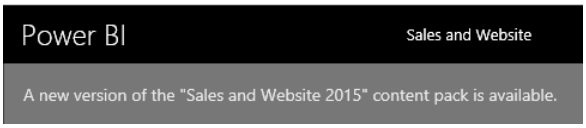


Figure 5-31: The warning message when there is an update to a content pack that has been copied to a personal copy.

Knowing that a new version of the content pack is available, Wendy can decide whether to get data from the same content pack again. If she decides to do so, a new copy of all the objects will be imported into her workspace. Because these objects will have the same name as the ones she previously copied, it would be a good idea to rename objects imported from a content pack when you decide to create your own personal copy; this way, you would not confuse them with the original copy in case you import the same content pack again in the future.

Conclusions

In this chapter, you learned how to consume and create content packs in Power BI. There are different types of content packs, each one with different behaviors available to the user. Moreover, you have also seen that you can create custom datasets from a service when the corresponding service content pack does not provide the data model that you need.

Here are the most important features you learned:

- A content pack contains a set of dashboards, reports, and datasets that a Power BI user can quickly import into his own personal workspace. He can also customize dashboards and reports imported from a content pack, but not a dataset.
- Content packs are available in the Power BI service only.
- A service content pack is published by Microsoft that you can use to connect to a service importing data that populates a set of predefined dashboards and reports. You must provide the credentials required to connect to the service from which you want to extract data.
- You can create a custom dataset in Power BI Desktop only, using a connector corresponding to the service content pack that you cannot customize. Be aware that not all service packs have a corresponding connector in Power BI Desktop, but there are several (such as Google Analytics) that have both.
- Any user can publish an organizational content pack. These contain predefined connections to data sources that cannot be changed by the user who consumes a content pack; only the content pack publisher can modify those connections.
- A user can consume an organizational content pack by just reading its content or by creating a personal copy that she then can modify.
- An organizational content pack publisher can update data. Such changes are automatically propagated to users who consume the content pack in a read-only mode.

Content packs are an important tool to quickly create a set of predefined reports and dashboards based on data coming from an existing external service, or from a dataset created within the organization.

CHAPTER 6

Building a data model

In this chapter, David moves to the next level in Microsoft Power BI usage. We are probably cheating a bit now, but we wanted this book to show you what Power BI can do for you when you master it, not just demonstrate its basic features. To do that, we presume that David—encouraged by the good results so far—spent some time learning the basics of data modeling and the DAX language. Having learned more details about Power BI, he begins again building the budgeting solution, but this time he can trust his better knowledge of the tools.

David loads the sales in the previous years, but unlike the last time, he does not use the view that Karin, the database administrator at Contoso, created for him. Instead, he uses more basic views, on top of the Contoso data warehouse, that provide data in a more fragmented way. There is a table containing information on the stores, one with the sales data, one for the date, and, lastly, a table of the products themselves. Using this information, he builds a first sales analysis project. Finally, he adds the budget information from the Microsoft Excel workbook and writes some DAX code to prepare the dashboards.

We will not discuss in detail all of the formulas and intricacies of the code and the data model; it's not realistic to expect you to learn how to perform these operations by reading one chapter. Our goal is to build the full project together (remember, you can replicate it by using the companion content). If you like the final project, you will probably be better motivated to proceed further with your study and follow David's path in learning data modeling and DAX.

Loading individual tables

Recall from Chapter 3 that David needed to speak with Karin to gain access to the Microsoft SQL Server database containing a view that returns sales for the past three years. David learned that he can perform an analysis on sales in a better way if—instead of using Karin's view—he loads the data from the original tables where Karin stores Contoso information. So, he arranges a meeting with Karin to gather more information about the internal structure of the Contoso data warehouse.

Karin explains to him that the database is organized in tables that he can access by using individual views (one per table). There is a table for each business entity of Contoso's business:

- **Products** This table contains information about the products sold by Contoso.
- **Sales** This one contains detailed sales, one row for each individual sale.
- **Stores** This table has information about the stores where the sales were transacted.
- **Date** This is a helper table that contains the calendar. David learned in a Business Intelligence class that such a table is of paramount importance when building a good data model.

Karin gives David access to the views so that he can load the granular information. David decides to begin again from scratch, so he opens Power BI Desktop and loads these tables into a new model, following the same procedure he did to load the Sales2015 view. The only difference is this time he loads four tables at once, as shown in Figure 6-1.

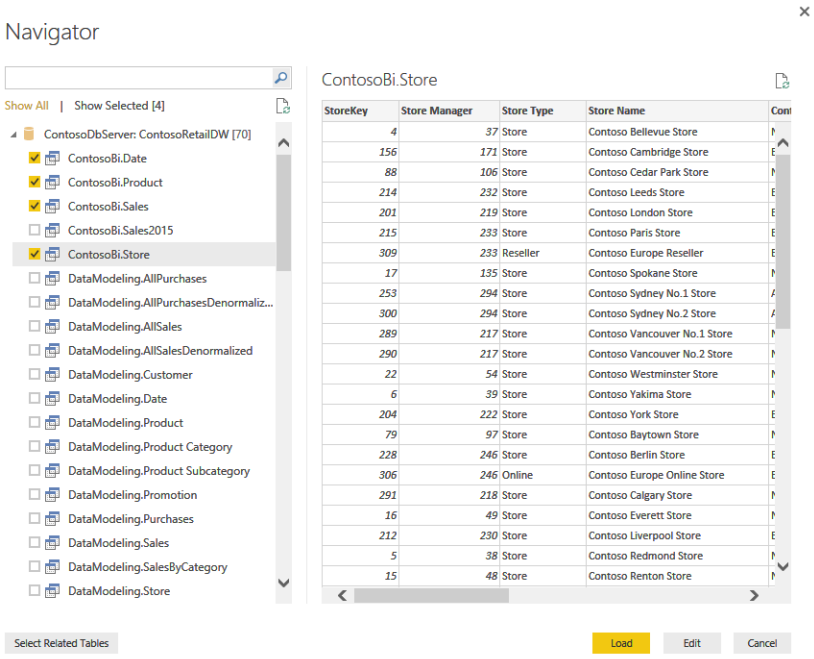


Figure 6-1: Using the Navigator dialog box, you can load multiple tables at once.

Instead of loading the tables directly from the Navigator dialog box, you'll find it more convenient to click the Edit Queries button (on the Home tab of the ribbon, in the External Data group) to open Query Editor and then change the names of the tables, removing the ContosoBi prefix. In fact, as you might remember, Query Editor names them ContosoBi.Sales instead of the more readable Sales.

After you close Query Editor, Power BI Desktop loads the table in the model and automatically creates some relationships among them. The Power BI Desktop algorithm that detects relationships is not perfect, and, in fact, it did not detect all of the relationships between the tables.

To follow along and catch up to this point, open Power BI Desktop and then open the companion content file for Chapter 6: Budget – Start.pbix. On the navigation bar, David clicks the Relationship View icon. He then sees the model illustrated in Figure 6-2 and notices that a relationship wasn't created between the Date and Sales tables.

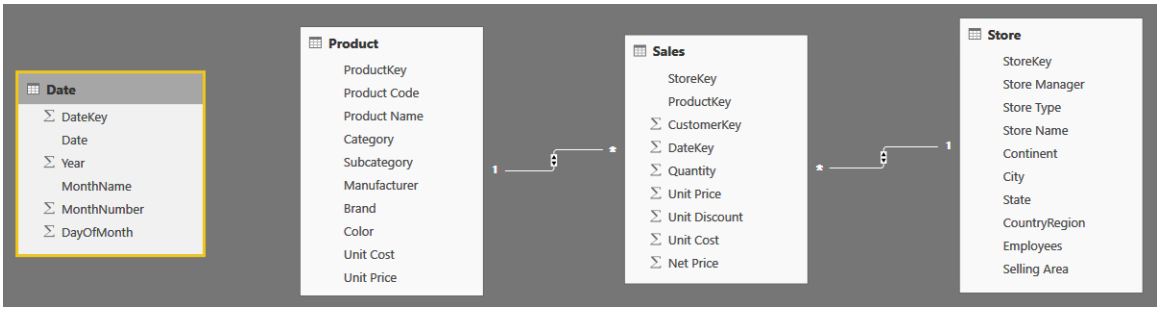


Figure 6-2: The Power BI Desktop relationship detector did not find the relationship between Sales and Date.

This is not an issue; you can easily create the relationship between Date and Sales by dragging DateKey from the Sales table to DateKey in the Date table to link the two tables with the correct relationship. The final data model is presented in Figure 6-3. (Note that your view of the data models might be different; for example, the Date table might be to the left of Sales, not below it.)

Figure 6-3: The data model structure is a very simple schema, with Sales in the middle and the other tables around it.

Implementing measures

The model, as it is, still requires some adjustments. First, David hides all of the columns that should not be visible when creating reports. He does this by going to Report View, selecting the table columns, right-clicking one, and then clicking Hide. David hides all of the keys and the columns that would be misleading if they were summed straight.

For example, the Sales table contains Quantity and Net Price. By default, Power BI offers to summarize Net Price by summing the values. In reality, this would be wrong because summing the price would not take into consideration the quantity sold.

> **Note** The Sales table in the Budget – Start.pbix file is hidden by default because all of its columns are hidden. To make it visible, in Data View, right-click the table, and then, on the shortcut menu that opens, click Unhide All.

The default summarization used by Power BI works perfectly well when you have a simple data model. But, as soon as you begin loading data from relational databases for which numbers are not stored in such a way as to be used in Excel workbooks, you need to stop using default summarization and begin writing DAX *measures*, instead. Measures, in DAX parlance, are scripts that you write using DAX-specific syntax. By using measures, you can author your own code and produce much more powerful data models.

David creates a simple measure to compute the Sales Amount. In Report View, in the Fields pane, David right-clicks the Sales table and then clicks New Measure. In the formula bar above the canvas in the middle pane, he replaces "Measure =" with the following code:

```
Sales Amount = SUMX ( Sales, Sales[Quantity] * Sales[Unit Price] )
```

This measure alone is already extremely powerful. In fact, because David is now loading data directly from the data warehouse, he can slice sales by using any of the available columns, not just those that were present in the single view he was using before. For example, now the Product table contains Category and Subcategory, which are useful for performing an analysis of sales in different countries/regions, with a report such as the one depicted in Figure 6-4.

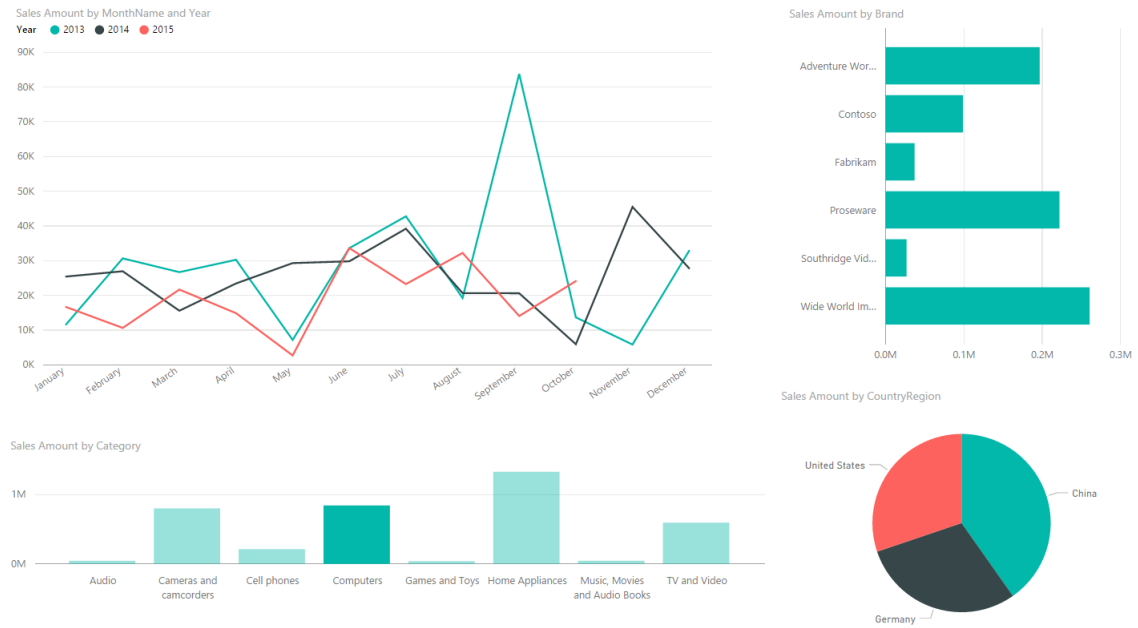


Figure 6-4: Using new columns available in the data model, the reports become more powerful.

Analysis of sales in previous years becomes more interesting as soon as you have more columns available. The report shows the relative contribution of different brands to the Computers category (notice how the Computers category is selected in the lower-left bar chart in Figure 6-4), whereas the line chart shows the behavior of sales over time. Different years are highlighted by different line colors. Using this tool, David can find an answer to different questions, like what is the reason for the peak in September 2013.

Creating calculated columns

Having more power typically raises the requirements of the data model. As an example, consider the line chart: having the sales of the three years with different lines might be useful for a comparison of different years; however, if you want to analyze the behavior of sales over the three years, it would be much better to show a single line that spans all of the years.

The problem is that the Date table contains the month name, and you can easily use it as we did in Figure 6-4, but if you remove the year from the legend, you get sales divided by month, not by month and year, as shown in Figure 6-5.

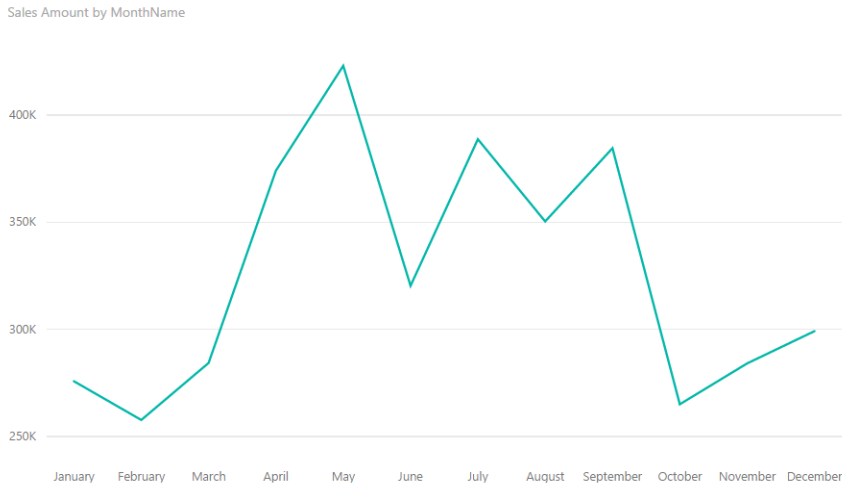


Figure 6-5: Slicing sales by month shows the total sold over all years for each month.

David needs a column containing both year and month at the same time. Such a column is not available in the original database. Fortunately, he has two different options to create this column: he can use Query Editor to add the column to the Date query, or he can create a *calculated column*.

In Chapter 4, you learned how to use Query Editor to generate new columns; let's use this as an opportunity to learn how to use calculated columns. To create a new column in a table, on the Power BI Desktop ribbon, on the Modeling tab, click New Column, as indicated in Figure 6-6.

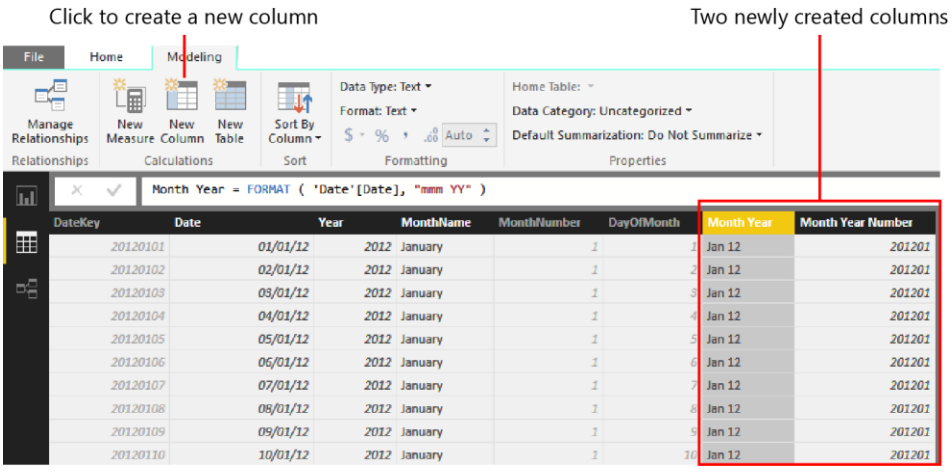


Figure 6-6: You can add new columns to a table by using the New Column button.

You can add these two columns to the Date table by typing the following measures in the formula bar above the table:

```
Month Year = FORMAT ( 'Date'[Date], "mmm YY" )
```

```
Month Year Number = 'Date'[Year] * 100 + MONTH ( 'Date'[Date] )
```

The first column contains a shortened version of month and year (we keep it short, to make it suitable for the line chart), whereas the second column is used to sort the first one, using the Sort By Column feature we already discussed.

If you now replace the Month Name with the Month Year as the axis of the line chart, the visualization is exactly what you want, showing the behavior of sales over three years, as you can see in Figure 6-7.

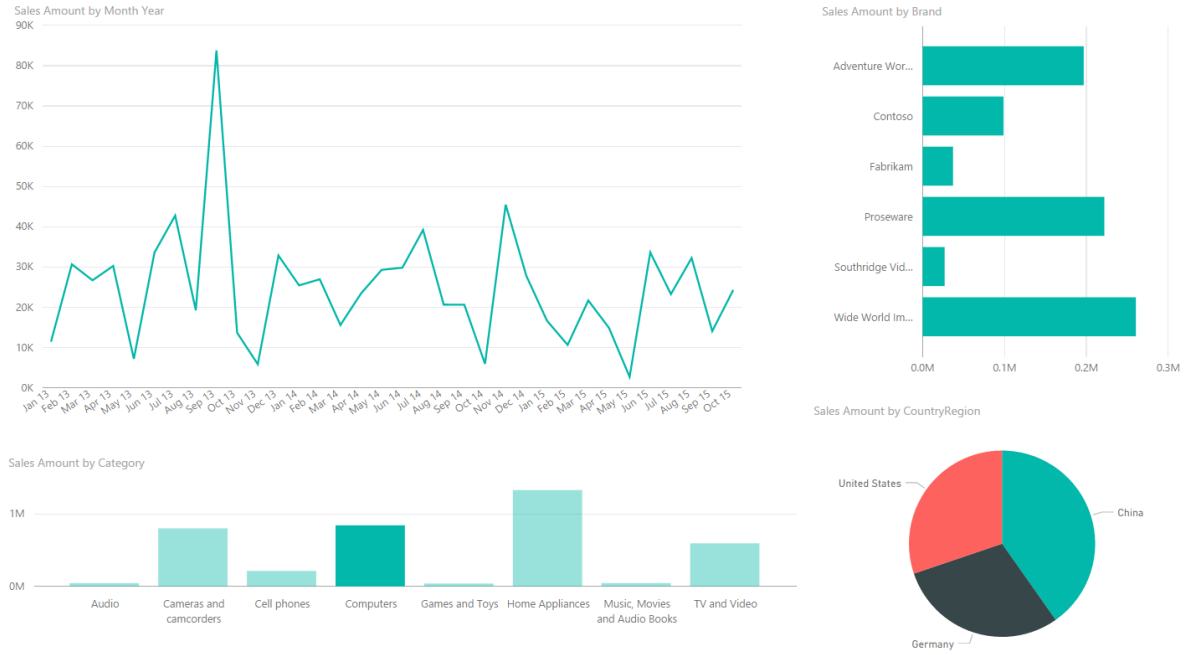


Figure 6-7: Using a calculated column for the axis of the line chart leads to the desired visualization.

When building reports, you will typically need a calculated column to make the visualization look perfect. Sometimes the descriptions are too large. In other cases, such as this one, you need a column representing a specific behavior. Power BI is an environment in which you model the data while having the visualization in mind as the final goal.

Improving the report by using measures

When you use calculated columns and measures to perform analyses, you're limited only by your imagination. For example, with a few calculations you can easily build a report like the one shown in Figure 6-8, which shows a bubble chart with the number of products versus the margin divided by category, where the size of each bubble is the amount sold.

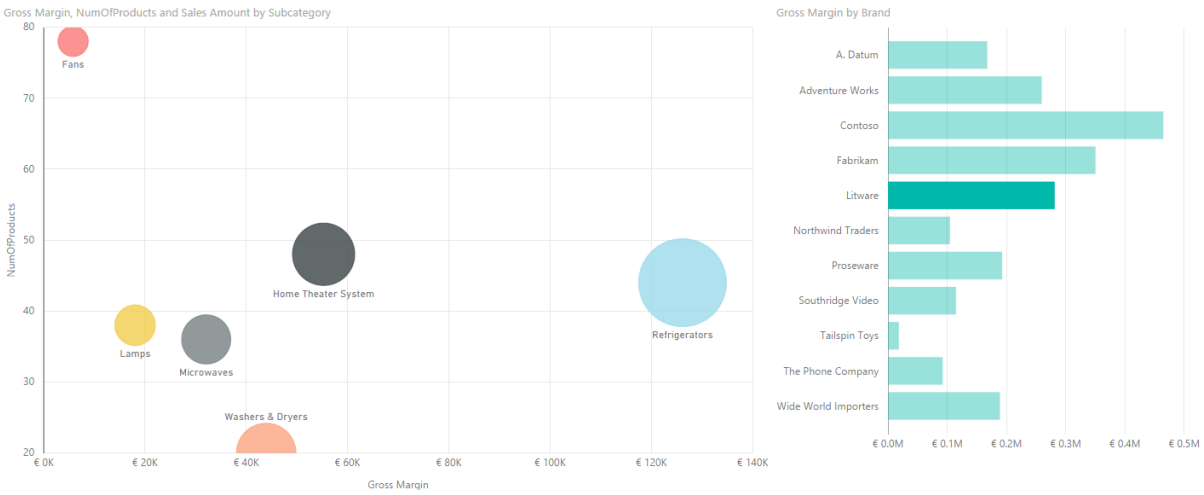


Figure 6-8: A bubble chart shows a large amount of information in a single chart, and they are gorgeous when used with visual interaction and automatic filtering.

The measures needed to build the report are very simple:

```
NumOfProducts = COUNTROWS ( 'Product' )

Gross Margin = SUMX ( Sales, Sales[Quantity] * ( Sales[Unit Price] - Sales[Unit Cost] ) )
```

NumOfProducts simply counts the number of products and gives an idea of how many articles are in the portfolio, whereas Gross Margin computes the gross margin of sales by subtracting the cost from the unit price before multiplying that value by the quantity.

> **Note** As you have seen so far, we are not explaining how to write the DAX code. The goal of this chapter is not to teach DAX; a short chapter in a short book would not properly address the complexity of the language. Our goal here is to show you what you can do as soon as you begin learning the basics of data modeling and DAX coding. Incredible analytical power awaits you when you complete your journey learning DAX, so hurry and start learning it today. In the meantime, we will move on with more complex DAX code and present some more scenarios that you can solve by writing simple formulas.

Integrating budget information

So far, David is excited about the power of his analytical tool; so much so, in fact, that he's forgotten that the task is about budgeting, not sales analysis. This is one of the major drawbacks of using Power BI: it is so much fun to dive into data and analyze it that you might become lost in evocative reports. Now it's time to get back to business and integrate the budget information.

Loading the budget information from Excel is straightforward, and David has already used the technique. But a problem arises as soon as he looks at the data model. The new table containing the budget does not have any relationship with the other tables, as you can see in Figure 6-9.

Figure 6-9: The Budget table does not have any relationships with the other tables in the model.

This time, it's not an issue of Power BI failing to detect the relationship, which earlier David was able to correct by using a simple drag-and-drop technique to establish a relationship. In this instance, the relationship cannot be created this way. In fact, when he tries to drag CountryRegion from the Budget table to the Store table, he sees the error shown in Figure 6-10.

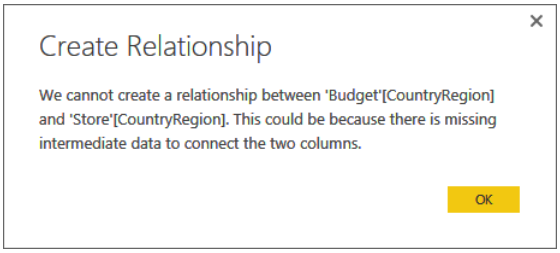


Figure 6-10: Trying to create a relationship between the Budget and Store tables leads to this error.

The error message suggests that an intermediate table might help solve the problem. But, before solving the issue, it's worth taking a few moments to understand it better.

You can create a relationship between two tables if the column you use to create the relationship is a key in the destination table. You can create a relationship between the Sales and Date tables based on the DateKey column because DateKey has a different value for each row in Date. Having a different value for each row is the requisite for a column to be a key. In fact, when you have a given date, you can uniquely identify the entire row in Date. In the model with Budget, CountryRegion is neither a key in the Budget table, nor in Store. Thus, you cannot create such a relationship.

There are multiple ways to solve this problem, based on the model or on an advanced usage of the DAX language. The solution based on the model is somewhat easier to learn and, by the way, it is the way suggested by the error message.

If you create a table containing all the possible values of CountryRegion, CountryRegion becomes a key for that table. At that point, you are able to create the relationships between Budget and Store.

You can build a table containing the possible values of CountryRegion by using Query Editor, as you did in Chapter 4, or by adopting a new technique: a *calculated table*. Calculated tables are tables computed using the DAX language that you can store in the model and use as any other table. To

create a calculated table, on the Power BI ribbon, on the Modeling tab, click New Table, and then, in the formula bar, type the following DAX expression:

```
CountryRegions =

SUMMARIZE (
    UNION (
        DISTINCT ( Budget[CountryRegion] ),
        DISTINCT ( Store[CountryRegion] )
    ),
    [CountryRegion]
)
```

Figure 6-11 shows the resulting table.

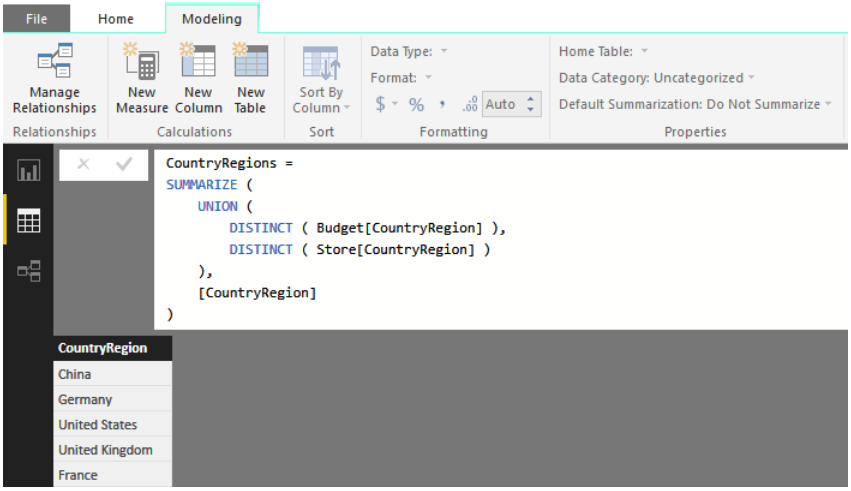


Figure 6-11: You can populate a calculated table with the distinct values of CountryRegion.

Let's return to see how David is faring. To create the table, he takes the distinct values of both the Budget[CountryRegion] and Store[CountryRegion] columns, forms a union with the partial results, and finally the Summarize function returns a summary table of the CountryRegion column. In this way, all the possible values will be represented in the resulting table, either referenced by the Budget or Store tables.

Now, David has the intermediate table that the error message in Figure 6-10 suggested to him. He can create one relationship between Store and CountryRegions, and another one between Budget and CountryRegions, completing the model. The table is a technical table, which is useful only to propagate the filter from Store to the Budget. Figure 6-12 presents the data model with the new table in place.

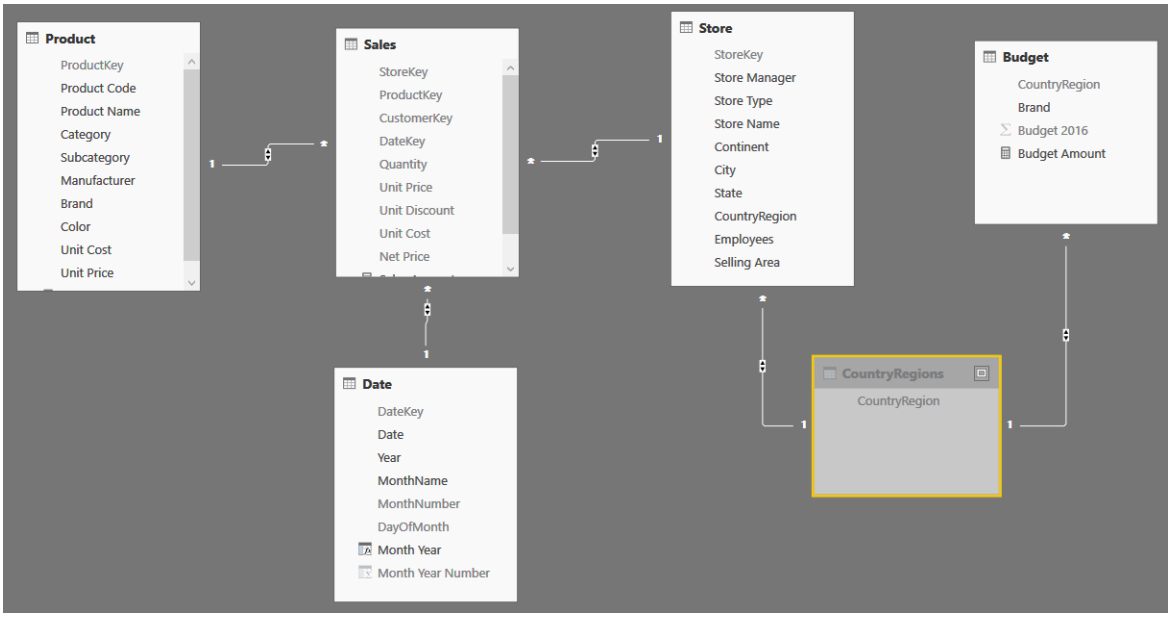


Figure 6-12: The CountryRegions table acts as an intermediate between Store and Budget.

From a technical point of view, David created a *many-to-many* relationship between the Budget and Store tables, using CountryRegions as the bridge table between them. To test that the model works well, he creates the following simple measure that returns the sum of the budget:

```
Budget Amount = SUM ( Budget[Budget 2016] )
```

You can project this measure on a simple report containing CountryRegion, the value of budget, and the value of sales. Now, CountryRegion correctly slices both Sales Amount and Budget Amount, as illustrated in Figure 6-13.

CountryRegion	Budget Amount	Sales Amount
China	418,500.00	1,416,712.00
United States	395,350.00	1,419,755.00
Germany	327,500.00	1,070,649.00
Total	**1,141,350.00**	**3,907,116.00**

Figure 6-13: With the correct data model, now the numbers are sliced correctly.

The first problem, here, is that the report is not showing meaningful numbers. In fact, because there is no filter on the year, it is accumulating sales over all available years and comparing them with the budget, which contains forecasts for 2016 only. You can easily solve the issue by creating the following measure that computes the sales amount for only 2015:

```
Sales 2015 = CALCULATE ( [Sales Amount], 'Date'[Year] = 2015 )
```

By replacing Sales Amount with Sales 2015 in the report, the numbers can be compared, as demonstrated in Figure 6-14.

CountryRegion	Budget Amount	Sales 2015
China	418,500.00	399,388.00
United States	395,350.00	324,186.00
Germany	327,500.00	322,863.00
Total	**1,141,350.00**	**1,046,439.00**

Figure 6-14: Using Sales 2015 in the report makes the numbers comparable.

You also need to apply the same technique to the other column available in budget, which is Brand. The DAX code is very similar to the previous example; you have only to change the column names to obtain the list of all the possible brands:

```
Brands =
SUMMARIZE (
    UNION (
        DISTINCT ( Budget[Brand] ),
        DISTINCT ( Product[Brand] )
    ),
    [Brand]
)
```

So far, so good. The next step hides a problem that—again—requires a bit of theory to be explained.

When you create the relationships between Product, Budget, and Brands, you end up with a model like the one depicted in Figure 6-15, wherein the relationship between Budget and Brands has been created but remained inactive (this is because it was the last one we created—by following a different order in the creation of relationships, you might obtain the relationship between Product and Brands as an inactive one).

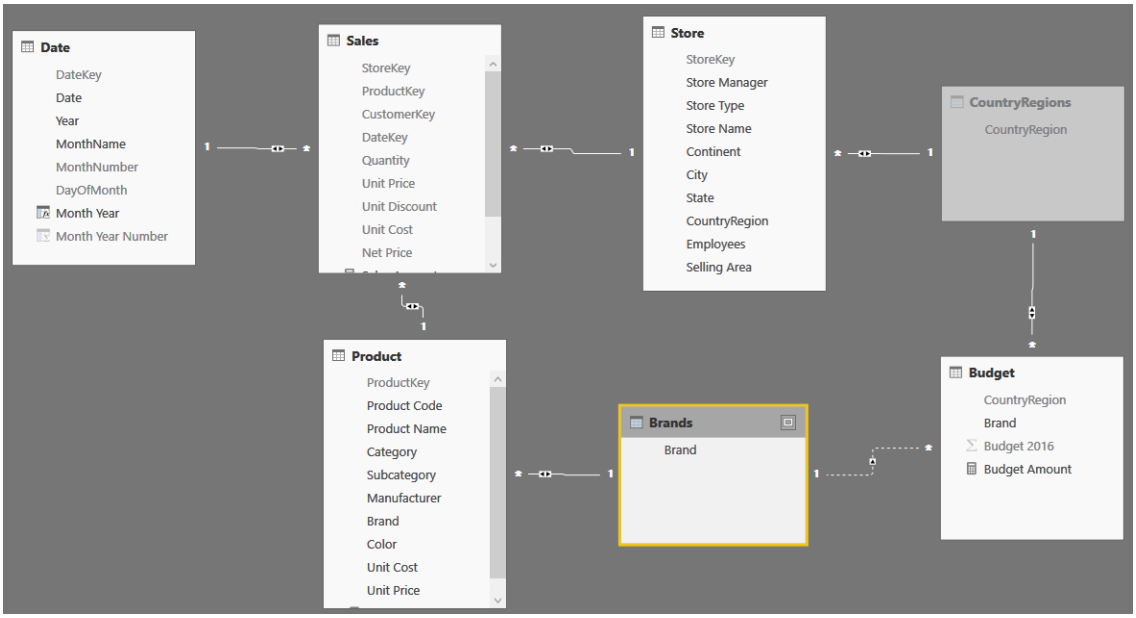


Figure 6-15: Among the many relationships, the one between Brands and Budget is inactive, signified by the dashed connector line between them.

What is an inactive relationship? It is a relationship that is present in the model but not used in the automatic filtering of values. Why did Power BI Desktop deactivate the last relationship we created? Because if it did not deactivate it, we would end up with an ambiguous model.

With the inactive relationship, filtering does not happen in the correct way. In fact, if you build a report using countries/regions and brands, the result is wrong, as demonstrated in Figure 6-16.

Brand	China	Germany	United States	Total
A. Datum	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Adventure Works	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Contoso	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Fabrikam	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Litware	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Northwind Traders	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Proseware	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Southridge Video	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Tailspin Toys	418.500,00	327.500,00	395.350,00	**1.141.350,00**
The Phone Company	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Wide World Importers	418.500,00	327.500,00	395.350,00	**1.141.350,00**
Total	**418.500,00**	**327.500,00**	**395.350,00**	1.141.350,00

Figure 6-16: Slicing by Brand does not produce a meaningful result (all of the values are repeated) because the underlying relationship is inactive.

An ambiguous model is a data model within which there are multiple paths linking two tables because all of the relationships are set as bidirectional (that is, the filter applies in both directions). (Note that you can tell when a relationship is bidirectional because there are two small arrows on the connector line, facing both directions.) So, where is the ambiguity? There are many here. For example, if you start from Product (see Figure 6-15), you can reach Budget following the bottom chain Product/Brands/Budget or the upper chain Product/Sales/Store/CountryRegion/Budget. If all of the relationships remained active, both would be legal paths, and you would end up with ambiguity.

You can solve ambiguity in most cases by just preventing the path from being traversed, but still maintaining the model features. For example, in the model examined, there is no need to make Sales filter Store and Sales filter Product. It is enough that the opposite direction works in both cases; that is, you can have both the Store and Product tables filter Sales. To perform this, you can double-click a relationship (the connector line between Sales and Store, for example), which opens the Edit Relationship dialog box, as shown in Figure 6-17.

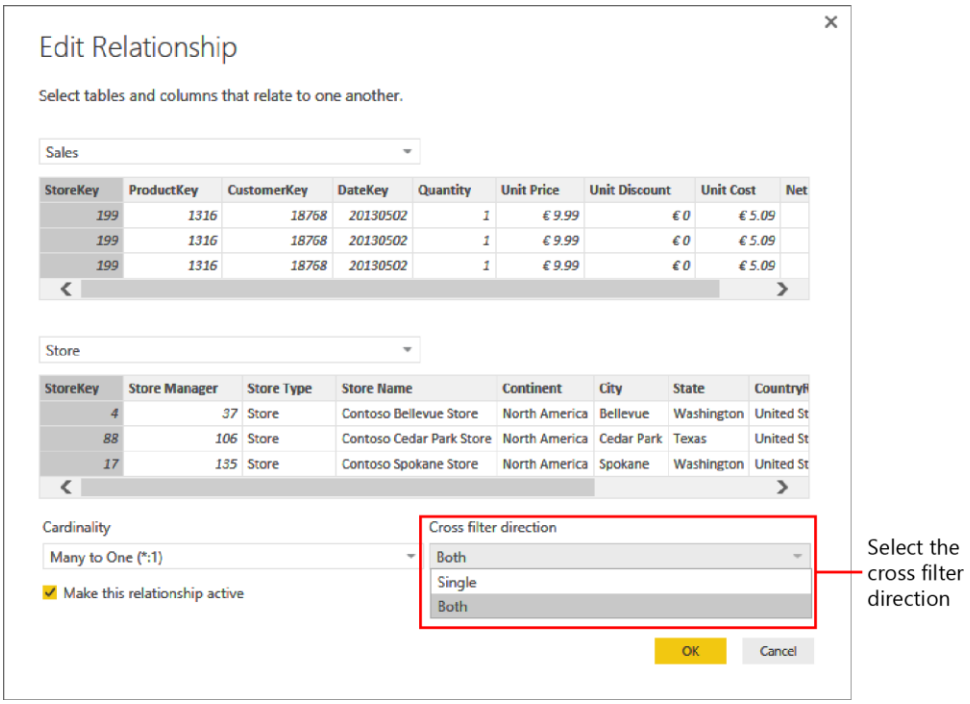


Figure 6-17: In the Edit Relationship dialog box, you can configure many properties of a relationship.

To disable bidirectional filtering, you must set the Cross Filter Direction to Single. You should do this to several relationships: the one between Sales and Store, the one between Sales and Product, and to

the two linking Budget with CountryRegions and Brands. The final model is represented in Figure 6-18.

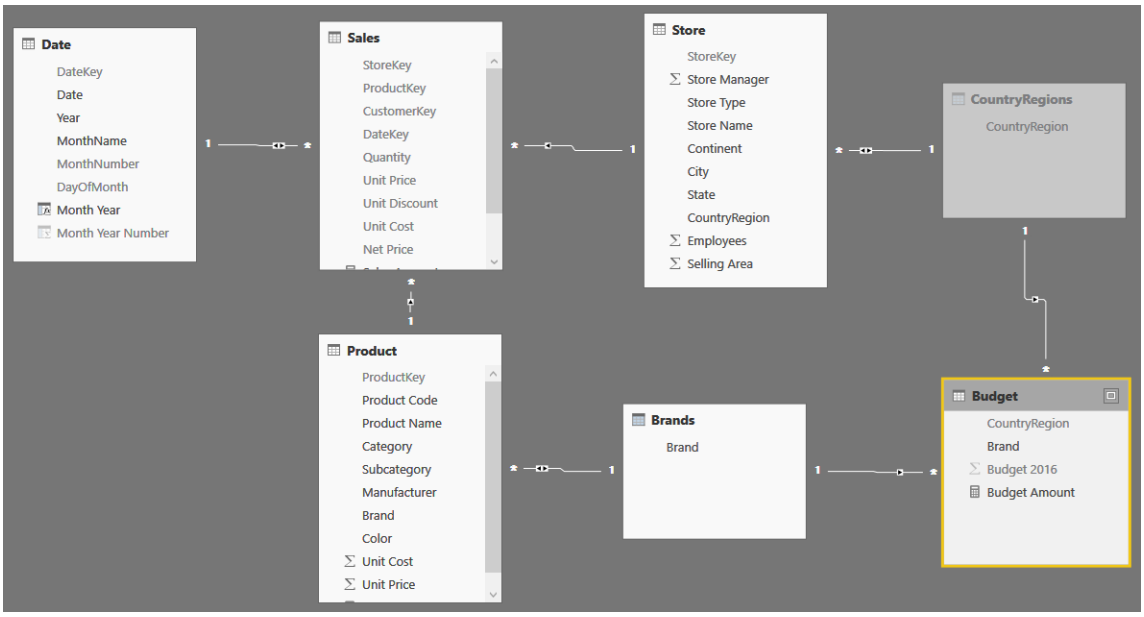


Figure 6-18: Disabling bidirectional filtering on most relationships removes the ambiguity.

After you remove ambiguity from the model and activate the correct set of relationships, the model works fine. You can test it by building a simple matrix that shows both Brand and CountryRegion now filtering the budget correctly, as illustrated in Figure 6-19.

Brand	China	Germany	United States	Total
Contoso	94,500.00	50,000.00	95,000.00	**239,500.00**
Fabrikam	90,000.00	45,000.00	34,500.00	**169,500.00**
Litware	65,500.00	12,500.00	65,000.00	**143,000.00**
Wide World Importers	37,500.00	30,000.00	58,000.00	**125,500.00**
Proseware	20,000.00	60,500.00	35,000.00	**115,500.00**
The Phone Company	30,000.00	30,000.00	18,750.00	**78,750.00**
Adventure Works	36,500.00	15,000.00	15,600.00	**67,100.00**
Northwind Traders	9,000.00	43,000.00	12,500.00	**64,500.00**
Southridge Video	22,500.00	16,500.00	22,500.00	**61,500.00**
A. Datum	5,000.00	13,500.00	30,000.00	**48,500.00**
Tailspin Toys	8,000.00	11,500.00	8,500.00	**28,000.00**
Total	**418,500.00**	**327,500.00**	**395,350.00**	**1,141,350.00**

Figure 6-19: With the correct model, slicing is performed the right way.

Reallocating the budget

Budget numbers are correct as long as you slice by brand or by country/region, which is the granularity at which the budget is defined. However, if you add a column that is not a part of the Budget table (for example, you can slice by Country/Region and Color), the result will be wrong. Figure 6-20 shows that values for Black and Silver colors in United States have the same value as the grand total of all the colors in United States.

Color	China	Germany	United States	Total
Black	418,500.00	327,500.00	395,350.00	**1,141,350.00**
Silver	418,500.00	327,500.00	395,350.00	**1,141,350.00**
White	413,500.00	314,000.00	365,350.00	**1,092,850.00**
Grey	409,500.00	284,500.00	382,850.00	**1,076,850.00**
Blue	388,500.00	297,500.00	376,600.00	**1,062,600.00**
Red	383,500.00	284,000.00	346,600.00	**1,014,100.00**
Pink	362,000.00	252,000.00	344,750.00	**958,750.00**
Green	329,500.00	266,000.00	338,500.00	**934,000.00**
Brown	346,500.00	169,000.00	290,600.00	**806,100.00**
Orange	292,500.00	151,000.00	282,500.00	**726,000.00**
Yellow	237,000.00	163,500.00	261,500.00	**662,000.00**
Gold	250,000.00	166,500.00	209,250.00	**625,750.00**
Purple	214,500.00	147,000.00	239,000.00	**600,500.00**
Silver Grey	189,500.00	108,500.00	159,500.00	**457,500.00**
Transparent	94,500.00	50,000.00	95,000.00	**239,500.00**
Azure	5,000.00	13,500.00	30,000.00	**48,500.00**
Total	**418,500.00**	**327,500.00**	**395,350.00**	**1,141,350.00**

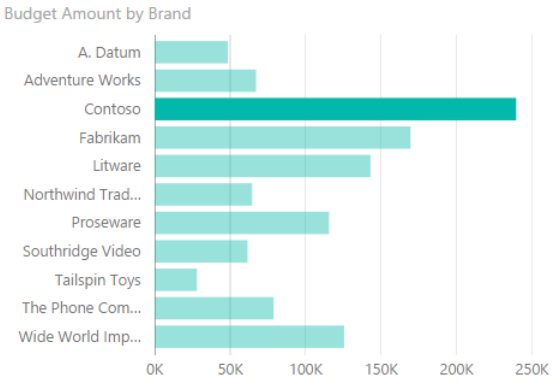
Figure 6-20: Slicing at an excessive granularity leads to incorrect numbers.

In reality, it is not that the number is wrong; it's just that it is very difficult to understand what it is computing. For example, the value in the cell at the intersection of United States and Black shows the sum of the budget in the United States for all the brands that have at least one product of the color black. Because some colors, like Black and Silver, are present for every brand, these rows show the same value of the grand total.

The number shown is clearly not what you would like to see. Intuitively, you would like to see the budget related to only Black products in the United States. However, with the data model we have so far, this is not what you obtain.

The problem is that the budget for black products (or any other color, for instance) is not available in the source workbook. There, you only have the budget for all the products of the same brand. Nevertheless, even if the number is not there, you can compute it by using a technique that is similar to the easy one David used at the beginning of this book (you might remember that David sliced the budget by month and then simply divided that value by 12).

To better understand the technique, let's begin with the report presented in Figure 6-21, which shows budget and sales in a matrix, and a chart used to filter and show only Contoso's data.



Color	Budget Amount	Sales 2015
Black	239,500.00	49,592.00
Blue	239,500.00	2,131.00
Brown	239,500.00	10,122.00
Gold	239,500.00	232.00
Green	239,500.00	28,464.00
Grey	239,500.00	23,469.00
Orange	239,500.00	
Pink	239,500.00	12,352.00
Purple	239,500.00	28.00
Red	239,500.00	13,834.00
Silver	239,500.00	54,875.00
Silver Grey	239,500.00	
Transparent	239,500.00	167.00
White	239,500.00	33,708.00
Yellow	239,500.00	
Total	**239,500.00**	**228,978.00**

Figure 6-21: The report shows Budget and Sales 2015 sliced by color for the individual brand, Contoso.

What is the budget value for Black? You can take the grand total (which is 239,500.00) and multiply it by an allocation factor computed by dividing sales of Black in 2015 (49,592.00) by the grand total of sales (228,978.00). Thus, the correction factor is 0.2165, and the value to display is 51,781.

Using this technique, you allocate the budget based on sales in the previous year. This time you take into account the correct seasonality and any other factors that made higher or lower sales for a specific color, category, or subcategory.

The budget has a granularity of brand and country/region. If you focus only on the Brand at the moment, you can compute the allocation factor by using the following measure:

```
AllocationFactor =
DIVIDE (
    [Sales 2015],
    CALCULATE ( [Sales 2015], ALLEXCEPT ( 'Product', 'Product'[Brand] ) )
)
```

Figure 6-22 shows the value of such a measure formatted as a percentage.

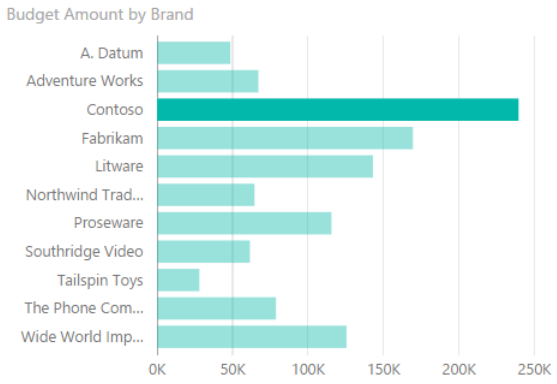
Color	Budget Amount	Sales 2015	AllocationFactor
Black	239,500.00	49,592.00	21.66%
Blue	239,500.00	2,131.00	0.93%
Brown	239,500.00	10,122.00	4.42%
Gold	239,500.00	232.00	0.10%
Green	239,500.00	28,464.00	12.43%
Grey	239,500.00	23,469.00	10.25%
Orange	239,500.00		
Pink	239,500.00	12,352.00	5.39%
Purple	239,500.00	28.00	0.01%
Red	239,500.00	13,834.00	6.04%
Silver	239,500.00	54,875.00	23.97%
Silver Grey	239,500.00		
Transparent	239,500.00	167.00	0.07%
White	239,500.00	33,708.00	14.72%
Yellow	239,500.00		
Total	**239,500.00**	**228,978.00**	**100.00%**

Figure 6-22: The allocation factor is the percentage to compute Budget Amount for when you analyze the budget at a lower granularity.

At this point, you can modify the code of Budget Amount, taking into account the allocation factor.

```
Budget Amount = SUM ( Budget[Budget 2016] ) * [AllocationFactor]
```

The result, which is shown in Figure 6-23, shows that now the budget is correctly sliced by Color, even if the original budget was not.



Color	Budget Amount	Sales 2015	AllocationFactor
Black	51,870.88	49,592.00	21.66%
Blue	2,229.84	2,131.00	0.93%
Brown	10,587.11	10,122.00	4.42%
Gold	243.18	232.00	0.10%
Green	29,772.14	28,464.00	12.43%
Grey	24,547.63	23,469.00	10.25%
Pink	12,919.69	12,352.00	5.39%
Purple	29.29	28.00	0.01%
Red	14,470.15	13,834.00	6.04%
Silver	57,396.92	54,875.00	23.97%
Transparent	175.28	167.00	0.07%
White	35,257.88	33,708.00	14.72%
Total	**239,500.00**	**228,978.00**	**100.00%**

Figure 6-23: AllocationFactor is now included in the formula of Budget Amount.

So far, we have focused only on the Brand, which is an attribute of the Product table. Alas, the budget also is defined at the CountryRegion level, and we need to take this into account. You need to

consider the CountryRegion when computing the allocation factor, similarly to what you did for the brand before. These are the final versions of the formulas to use:

```
AllocationFactor =
DIVIDE (
    [Sales 2015],
    CALCULATE (
        [Sales 2015],
        ALLEXCEPT ( 'Product', 'Product'[Brand] ),
        ALLEXCEPT( Store, Store[CountryRegion] ),
        ALL ( Date )
    )
)

Budget Amount = SUM ( Budget[Budget 2016] ) * [AllocationFactor]
```

Figure 6-24 demonstrates that the allocation is performed against 2015. Notice in the same chart on the right, the measures Budget Amount, Sales 2015, and Sales 2014. Budget follows the same distribution of Sales 2015 and ignores Sales 2014, which has a different distribution of numbers.

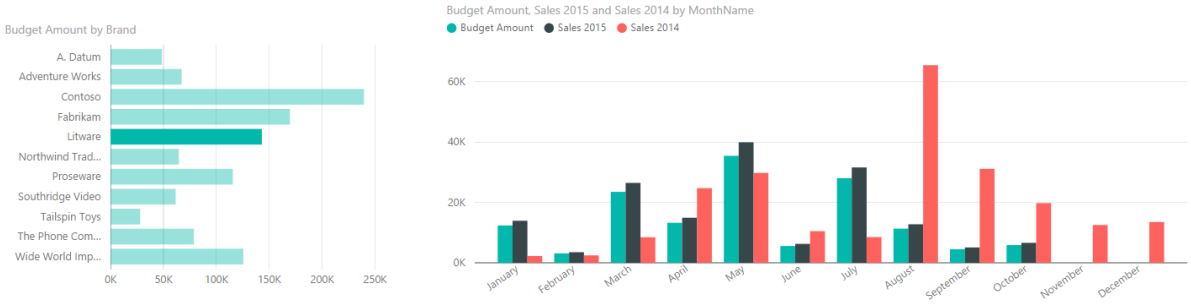


Figure 6-24: Distribution of Budget Amount is identical to Sales 2015 and different from Sales 2014.

Conclusions

As we said in the introduction, in this chapter we cheated a bit. We did not want to show you another step-by-step guide to implement another dashboard. Instead, we wanted to give you a sneak preview of the capabilities of Power BI Desktop when you uncover the most advanced tools, namely:

- **Building a model** When you begin loading the raw tables from the SQL Server database, instead of predefined queries with aggregated values, you can perform much more powerful analyses. At the same time, you are responsible for handling the data model by yourself. Power BI Desktop offers you all the tools required to build a complex data model.
- **The DAX language** DAX is your best friend in the process of analyzing data. In this chapter, we used it to create calculated columns, measures, and calculated tables. This book is not the proper venue for explaining how DAX works; that would fill an entire book by itself. If you are interested in learning more about DAX, check out our book *The Definitive Guide to DAX* (Microsoft Press, 2015).
- **Building columns for specific charts** Sometimes, you need a column for an individual chart. There is nothing wrong with doing this; you can just build it.

By using some basic skills, you can take Power BI from a simple reporting tool to what it really is: an extremely powerful modeling tool with which you can build gorgeous analyses on top of your data.

CHAPTER 7

Improving Power BI reports

The previous chapters introduce many features of Power BI that are related to data modeling, publishing, and sharing. The focus in those chapters is on the data and the numbers, not on the presentation. Our friend David took advantage of these features to create a solution that uses Power BI to support a collaborative effort in the yearly budgeting process for his company, Contoso. Those chapters intentionally do not focus on the presentation layer or the visual options of Power BI, but now it is time to improve the reports and dashboards in your models.

This chapter is dedicated to the visual presentation capabilities of Power BI. We will use different dashboards with different datasets and requirements to show you the different scenarios and tools available. For this reason, we're going to look beyond David's requirements for his solution—we need a broader scope now. So, you will learn how to choose between the built-in visualizations available in Power BI, how to use custom visuals, how to use DAX to solve common reporting issues, and, finally, what the correct approach is for designing high-density reports.

This chapter is neither a step-by-step tutorial for using the visual editor in Power BI, nor a complete visual design patterns guide covering all the possible types of standard and custom visualizations. Its

goal is to show you the available options in Power BI and act as an initial guide to choosing visualization solutions, depending on your requirements. We will do that by providing different practical examples that are available in the companion content, so that you can open the files and analyze the data in more detail, reviewing all the properties we used. In these pages, we provide commentary, explaining the reason for certain design choices; you should be able to apply the same principles when designing your own reports.

Choosing the right visualizations

You have several standard visualizations available in Power BI. In addition, you will see that you can extend this set by using custom visualizations. But, before doing that, you need to know what you can and cannot do using the standard components, which you can see in Figure 7-1.

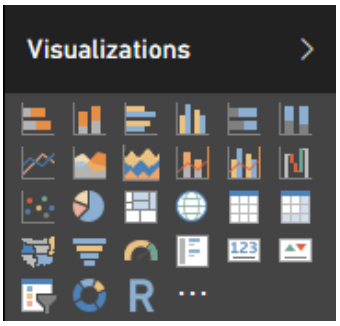


Figure 7-1: The standard visualizations available in Power BI.

The list of standard visuals includes 27 components in Power BI that are available as of this writing, but this might expand in future versions. Here are descriptions of the standard components:

- **Stacked bar chart** Use this when you want to compare different values of the same measure, side by side, or when you need to display different measures that are a part of the same whole. The bars are horizontally oriented rows.
- **Stacked column chart** The same as a stacked bar chart, but vertically oriented.
- **Clustered bar chart** Similar to the stacked bar chart, but instead of comparing different measures within the same bar, with a clustered bar chart you can compare different measures side by side.
- **Clustered column chart** The same as a clustered bar chart, but vertically oriented.
- **100% Stacked bar chart** Similar to the stacked bar chart, but with each measure using a slice of each bar, which always corresponds to the entire width available (100%).
- **100% Stacked column chart** The same as the 100% stacked column chart, but vertically oriented.
- **Line chart** Use this to display the trend of some measures over time. Usually the y-axis has a range that does not include zero.
- **Area chart** Similar to the line chart, use this when you want to display cumulative data rather than sequences of points. Usually, the y-axis range begins at zero, and there is only one measure. This looks like a line chart wherein the areas are filled with layers of colors.
- **Stacked area chart** Similar to the area chart, but with each measure cumulated to the others.

- **Line and stacked column chart** Use this when you need to display measures with different scales, such as currency and percentage, or different value ranges.
- **Line and clustered column chart** The same as the line and stacked column chart, but using clustered columns instead of stacked columns.
- **Waterfall chart** Use this to display cumulative data, highlighting for each value its positive or negative value. The initial and final value columns usually start on the horizontal access, with color-coded floating columns between them, making it look like a waterfall or bridge.
- **Scatter chart** Use this when you want to show possible correlations between two measures.
- **Pie chart** Use this to display the distribution of values of one or more measures. The values appear as pieces of the pie, with the larger values taking up larger slices. However, using pie charts is not a best practice.
- **Treemap** Similar to a pie chart, but using a rather different graphical representation, wherein the values are represented by colored rectangles on a page. It could be an alternative to a pie chart, but it is equally unreadable when you have many elements in it.
- **Map** Use this to display geographical data with variable-sized circular shapes on Bing maps.
- **Table** Use this to display data in a textual form as a simple table, where every attribute and every measure is a single column in the result.
- **Matrix** This extends the table, making it possible to group the measures by rows and columns.
- **Filled map** Similar to the map, but the data is represented by colored overlay areas.
- **Funnel** Similar to the stacked bar chart, but with a single measure and a different graphical representation, wherein the rows are stacked in order, which makes the chart look like a funnel.
- **Gauge** Use this to display a single measure against a goal. This chart resembles a gauge style that is common in cars.
- **Multi-row card** Use this to display different measures and attributes for each instance of an entity, each placed on a different colored and graphical card.
- **Card** Use this to display a single numerical value of a measure textually, placed on a colored and graphical card.
- **KPI** Use this to display a single value with a trend line chart in the background, highlighting its performance with colors.
- **Slicer** Use this to filter one or more charts by selecting values of an attribute.
- **Donut chart** Similar to the pie chart, but with a donut or tire-like graphical representation. However, using donut charts is not a best practice.
- **R script visual** Use this to display charts generated by R-language code.

The first design principle is simple: just because you have many components, there is no reason for you to use all of them. In a single report, the presence of many different types of components can be confusing. Thus, do not put too many different visualizations in the same report without a good reason. Moreover, the default properties of a component are not necessarily the right choice for your report, so you should consider modifying their value to better display your data. You will see several examples of these principles applied to our first report example.

For instance, consider a dashboard displaying Sales for Contoso. By choosing the right visualization and setting the right colors, you can obtain a good presentation of your data, focusing your viewer's attention on the data instead of the visualization itself. You can see a first version of this dashboard in Figure 7-2, in which sales amount, margin, and target values are sliced by date, brand, subcategory, and class. This dashboard is available on the page "Sales 2015" of the Sample-Sales.pbix file included in the companion content.

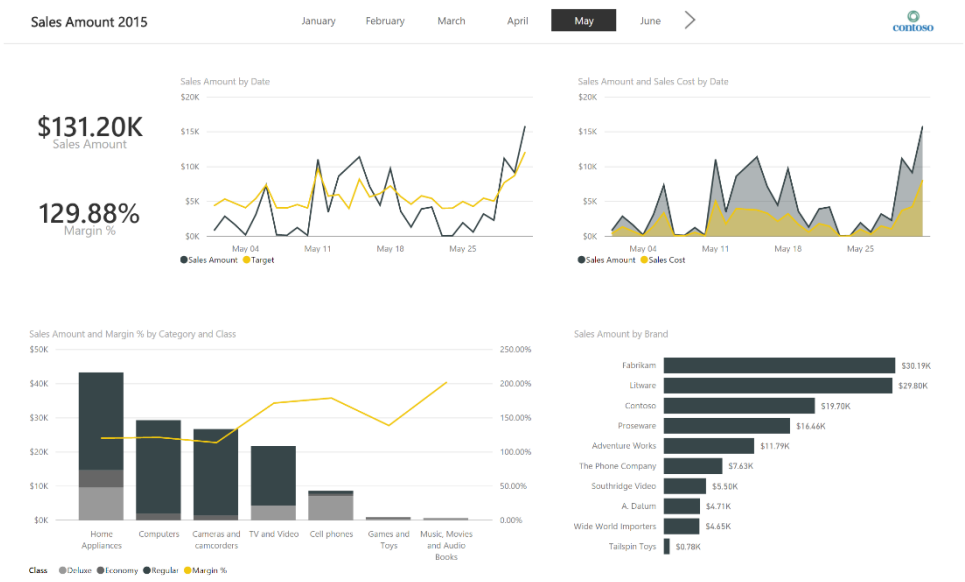


Figure 7-2: First version of a report displaying Contoso's sales in 2015 using standard visualizations.

The first thing to notice in Figure 7-2 is that there are only two colors used in the report: black and yellow. Items in black identify the sales amount measure, whereas items in yellow are for other comparison measures (target, sales cost, and margin percent, depending on the visualization).

The four charts in the report use a limited number of common visualizations: line chart, bar chart, and columns chart. The rationale behind each visualization choice is described in the next section, but in general it is better to use simple, well-known visualizations when that is sufficient to produce the display of useful and understandable information.

Choosing between standard visuals

The first chart in Figure 7-2 compares Sales Amount and the Target values using a simple line chart. Figure 7-3 shows this in more detail.

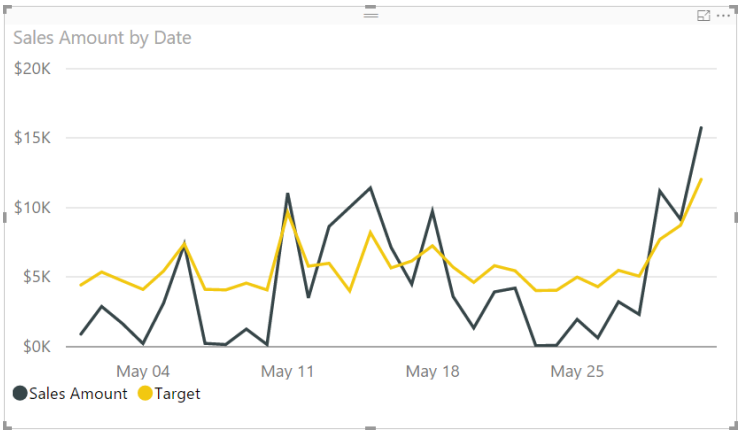


Figure 7-3: A line chart of sales amount by date.

The line chart is the primary choice when you display a measure over a date range or time, using the x-axis for the temporal dimension. You can select the colors of the different measures by using the Data Colors properties, as demonstrated in Figure 7-4. Here you can select the color of each measure included in the line chart.

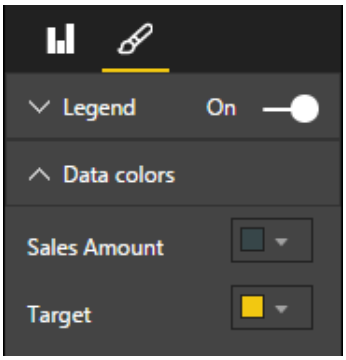


Figure 7-4: The Data Colors properties for a line chart visualization.

You can use a slight variation of the line chart when a measure that you want to display is part of another measure. For example, consider the Sales Amount and Sales Cost measures. Hopefully, Sales Cost is always lower than Sales Amount, and the graphical distance between them represents this margin. With the line chart, the gap between the two measures might not be clear, so you might want to "paint" the area below the line by using the values of the measures along time. The area chart does exactly this, as illustrated in Figure 7-5.

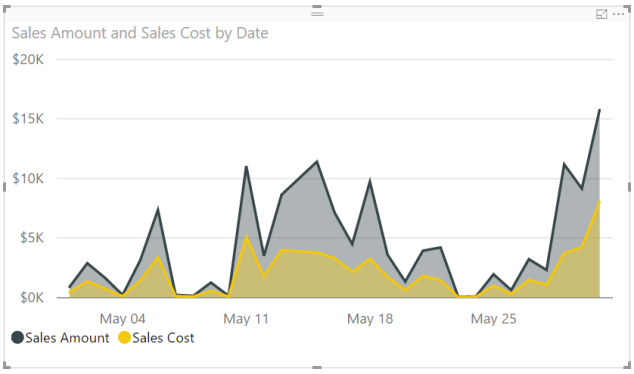


Figure 7-5: An area chart of sales amount versus sales cost by date.

The y-axis must begin at zero; otherwise, the area would not be fully representative of the two values. The visible gray area expresses the delta between the two measures in a graphical way and corresponds to the margin. You should not use an area chart when you have several intersections between different lines. You should consider it only when measures do not intersect often. The example in Figure 7-5 demonstrates one of the few cases for which you can consider using it.

> **Note** For the sample reports in this chapter, for the most part we do not use pie charts and donut charts. This is because they are not considered a best practice, with an exception that you will see in the last section. The human brain can more easily make comparisons between lengths (as in a bar chart) than between angles (as in pie and donut charts).

Comparing measures with different scales requires particular visualizations. You need to display two y-axes, and you need a way to easily associate the axis corresponding to each measure. For example, Figure 7-6 shows a line and stacked column chart that yields more details by displaying the sales amount measure divided by category and class, compared with the margin percent divided by category.

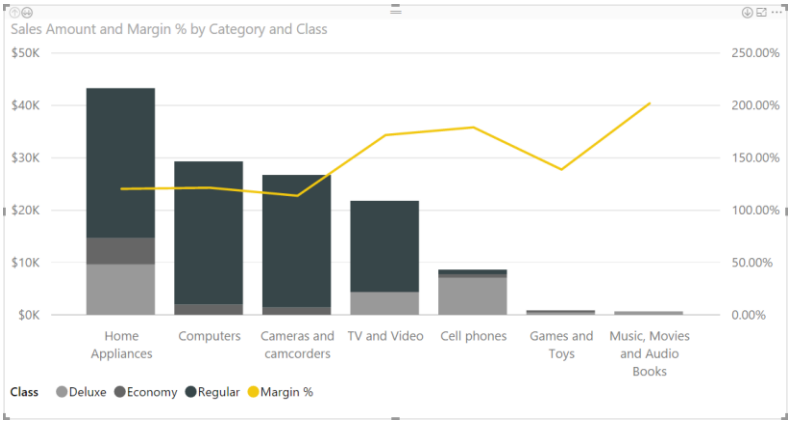


Figure 7-6: A line and stacked column chart of sales amount and margin percent by category and class.

The scale of the main measure (sales amount) is represented on the left y-axis, and the other measure (margin percent) is on the right y-axis. The x-axis shows the name of the category corresponding to each column, which is also divided by class using different shades of gray. Figure 7-7 shows the properties of this component used to bind data. The x-axis is called the *shared axis*, and it can include more than one attribute. In this case, we used two product attributes: Category and Subcategory. This makes it possible to perform an interactive drill-down of data in Power BI.

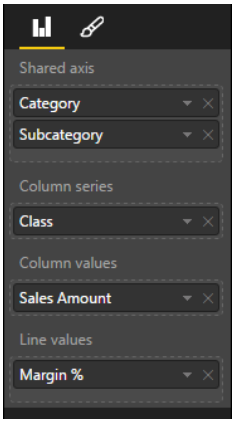


Figure 7-7: Data binding properties for a line and stacked column chart.

You can activate the drill-down feature for the selected column chart by clicking the drill down button (the down-arrow icon) located in the upper-right corner of the visualization. When drill-down mode is turned on, the drill-down button changes to display a black background, as depicted in Figure 7-8.

Figure 7-8: The drill-down button in a visualization. The black background signifies that drill-down mode is turned on.

With drill-down mode activated, when you click a column within the chart, you can drill down to that column's respective subcategories. Figure 7-9 shows the resulting chart when you click the Computers column in Figure 7-6. Note all of the subcategories that are related to the Computers category.

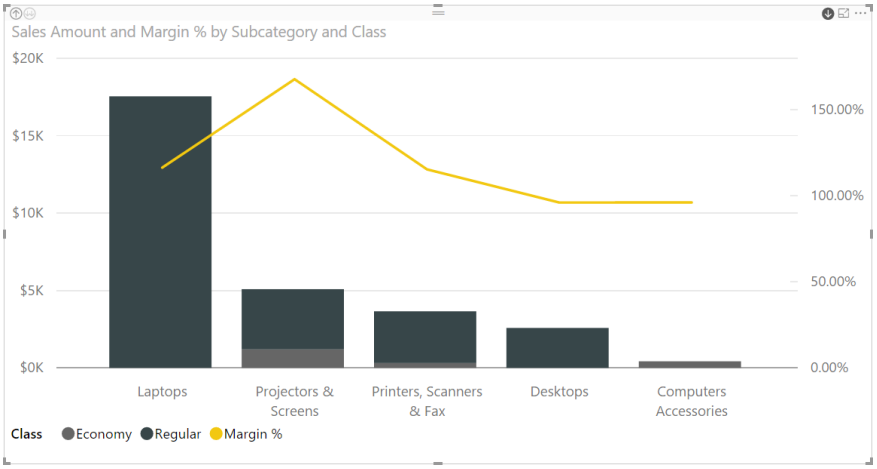


Figure 7-9: A line and stacked column chart of sales amount and margin percent by subcategory and class for the Computers category.

To drill up, in the upper-left corner of the visualization, click the drill-up button (the up-arrow icon), as highlighted in Figure 7-10.

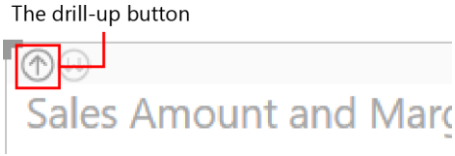


Figure 7-10: The drill-up button in a visualization.

After you move up to the product category again, you can drill down to all of the subcategories for every category by selecting the double-arrow button in the upper-left corner of the visualization, as illustrated in Figure 7-11.

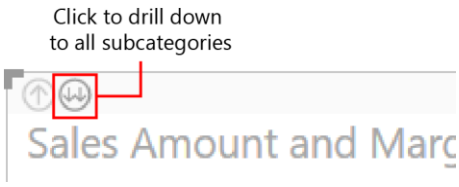


Figure 7-11: The drill-down button in a visualization.

When you drill down to the subcategory level for all the categories, you obtain a chart similar to that shown in Figure 7-12. Both the Sales Amount and Margin percentage measures have the subcategories granularity in the chart, so you can still compare them.

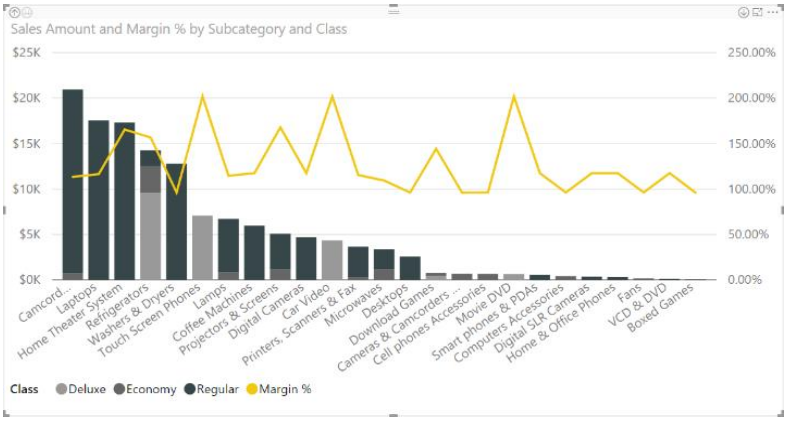


Figure 7-12: A line and stacked column chart of sales amount and margin percent by subcategory and class, for all the categories.

More info You can find an animated guide on how to use the drill-down feature in Power BI at https://powerbi.microsoft.com/documentation/powerbi-service-drill-down-in-a-visualization/.

Using custom visualizations

Power BI provides a number of embedded visualizations that are ready to use. By changing properties of the existing components, you can create solid presentations of your data. However, sometimes you might want to present data in a way that is not possible using the standard components. Power BI provides you with a gallery of custom visualizations that have been created by members of the Power BI community. You can download and install one or more of these custom visualizations in your report. To view a gallery of custom visualizations, go to https://app.powerbi.com/visuals/.

In this section, we will improve Power BI reports using features available in selected custom visualizations. The goal is to show you how to include custom visualizations in your report and what improvement you can achieve by using them. We suggest that you visit the gallery of available components because we cannot cover all the custom visualizations that are available, and new custom visualizations are published weekly.

First steps with custom visualizations

The first improvement we want to apply to the dashboard in the Sample-Sales.pbix file is coloring Sales Amount and Margin % according to their value. Using the standard card visualization, these values have a fixed color, as depicted in Figure 7-13.

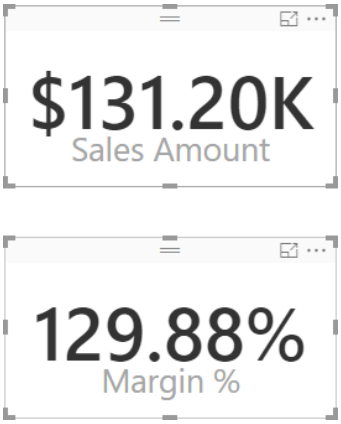


Figure 7-13: Sales Amount and Margin % displayed in a standard card visualization.

You can change the color in the Data Label properties, but the choice would be static. We would like to dynamically change the color so that the Sales Amount is green when it increases more than 20 percent compared to the value of the previous year, and the margin percentage is green when it is higher than 130 percent; yellow when it is between 100 percent and 130 percent; and red when it is lower than 100 percent (you might want to use a red color only for negative margins in other scenarios; we use ranges that adapt to this specific example).

To achieve this goal, you need a custom visualization that dynamically changes the color of the displayed value, based on a particular state. In the custom visual gallery, you can choose the Card With States By SQLBI visualization, as illustrated in Figure 7-14.

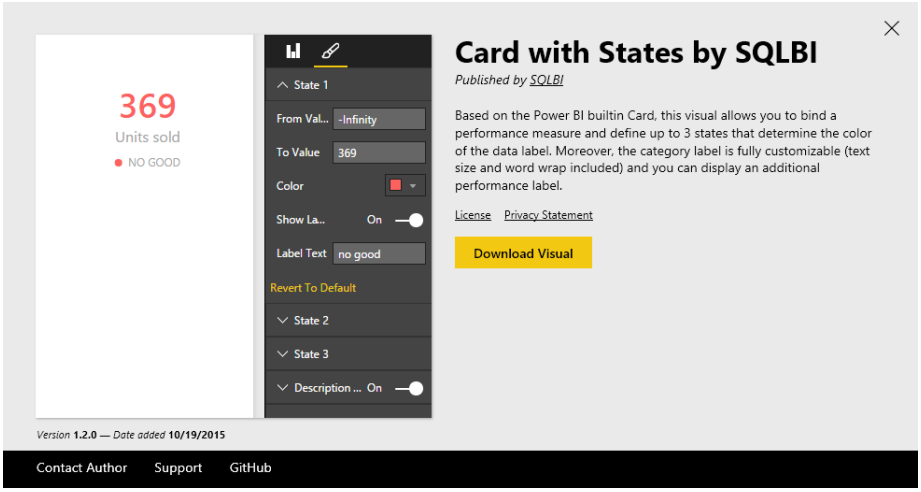


Figure 7-14: A description of the custom visualization Card With States By SQLBI.

After you download the visualization, you save the file using the .pbiviz extension (Power BI Visual). Then, in the Visualizations pane, you can import that visualization in your report by clicking the Import From File button (the ellipsis), which is highlighted in Figure 7-15.

More info You can find a more detailed guide describing how to import a custom visualization at https://powerbi.microsoft.com/documentation/powerbi-custom-visuals-use/.

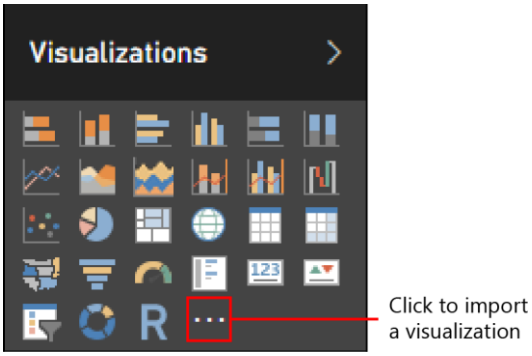


Figure 7-15: The Import From File button (ellipsis) in the Visualizations pane.

> **Note** The file Sample-Sales.pbix already includes the custom visualizations used in this example. You should already see the corresponding icons without having to install them.

If you select the card visualization displaying Sales Amount, you can change it to Card With States By SQLBI by clicking its button in the Visualizations pane, which became available after you imported the custom visual. Figure 7-16 shows the new button.

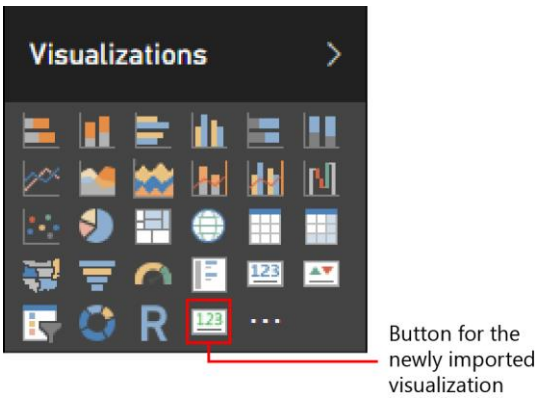


Figure 7-16: The button for the imported Card With States By SQLBI visualization.

In the Fields pane of the component (left side, Figure 7-17), you set State Value to YOY %, which is a measure defined in the data model that provides the year-over-year growth as a percentage. We want a red color if the growth is negative, yellow if the growth is positive and less than or equal to 20 percent, and green if the growth is greater than 20 percent. Set the To Value property (shown in the right pane in Figure 7-17) of the State 2 section to 0.2 (which corresponds to a 20 percent increase), and the From Value property of the State 3 section to 0.2, as well.

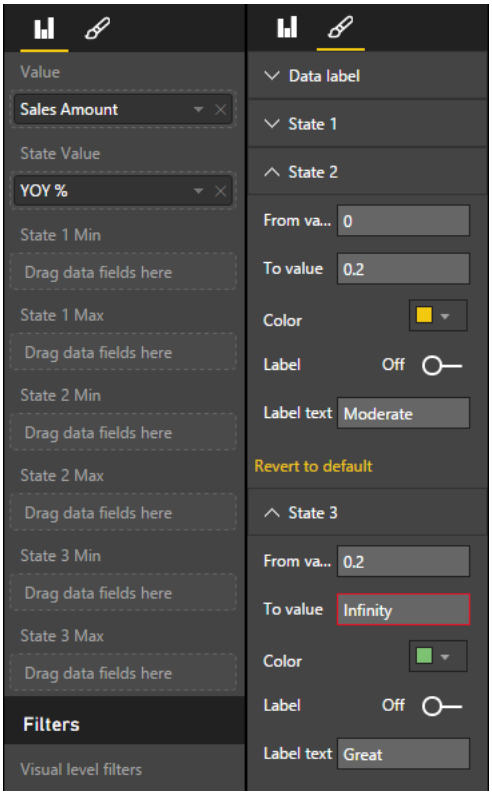


Figure 7-17: The Fields and Format panes for the Card With States By SQLBI visualization.

You repeat the same operation for the card visualization displaying Margin %: changing it to Card With States By SQLBI, setting the State Value field to the Margin % measure, and setting To Value in the State 2 properties to 1.3, and also changing From Value in State 3 to 1.3. In this way, you will display the Margin % using the same value displayed, whereas the Sales Amount will be colored, based on a year-over-year growth percentage. Figure 7-18 shows the final result. By changing the selection of months, categories, and brands, you might see different colors.

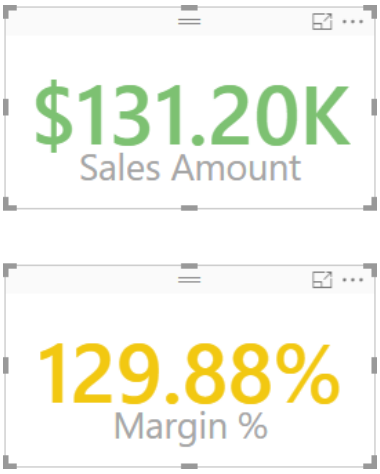


Figure 7-18: The Sales Amount and Margin percentage using Card With States By SQLBI.

We used this simple example to explain the process of importing and using custom visualizations in your report. In the following sections, we will focus more on considerations about when a custom visualization is useful or even required.

Improving reports by using custom visualizations

One of the charts available in the initial dashboard (the clustered bar chart in the lower-right corner of Figure 7-2) shows the sales amount divided by brand. This chart, as with all of the others in the dashboard, is dynamically updated whenever you select a different month or an element in other charts in the same dashboard. For this reason, the values displayed correspond to the sales filtered by the current selections in the charts and slicers in the dashboard. However, you might want to compare the sales amount of each brand with the corresponding sales made one year before as well as with the goal defined for the same selection. This initial chart displays only the sales amount, as shown in Figure 7-19.

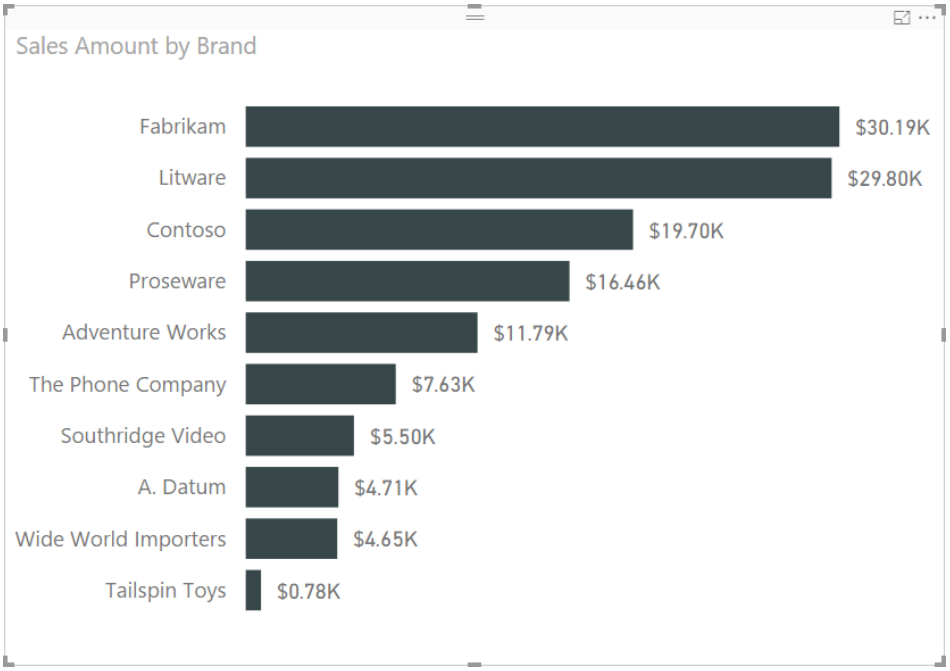


Figure 7-19: A clustered bar chart of sales amount by brand.

If you want to display other measures in the same chart, you need to add them. Each measure will have a different bar, and you should set different colors to recognize each measure. For example, Figure 7-20 shows the goal measure and the sales amount measures for the years 2014 and 2015 (previously, we were displaying the sales amount only for the year 2015).

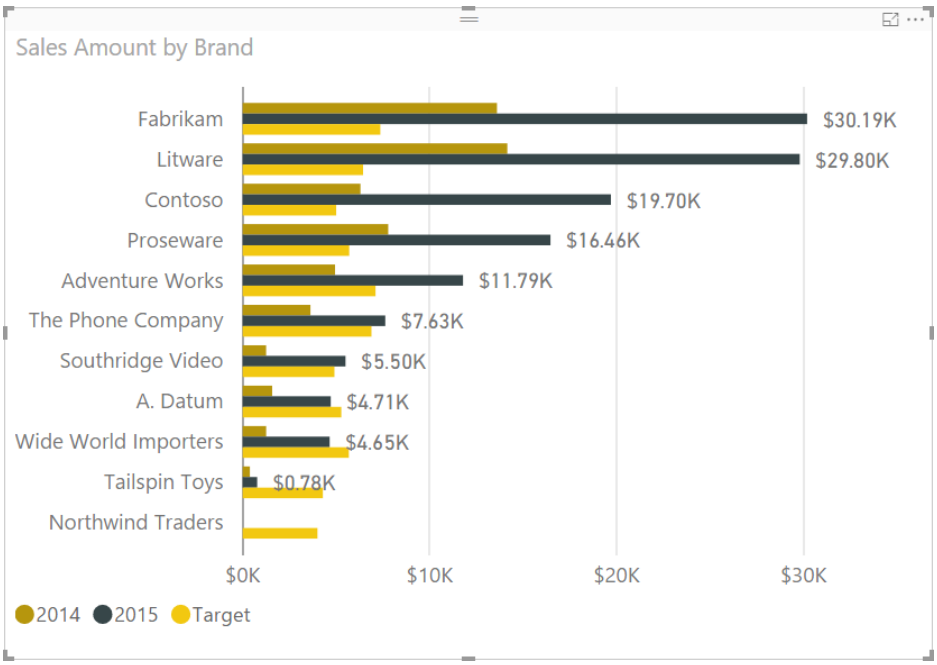


Figure 7-20: A clustered bar chart of sales amount and target by brand.

> **Note** You might wonder how we displayed two values of the same measure (sales amount) in the same chart with two different names. In Power BI Desktop, you can create new measures in the data model, so you can simply assign an existing measure to a new measure with a different name and then assign the new measure to the visualization. You will find other considerations about using DAX to improve reports in the section "Using DAX in data models," later in this chapter.

You might consider using a different visualization to improve the readability of this chart. For example, by using the custom Bullet Chart By SQLBI, you can obtain the results shown in Figure 7-21. The value of the sales amount for 2015 is a bar in the middle, surrounded by a shorter bar (overlapping on top of it) that has the sales amount for 2014, along with a short, black vertical line that acts as a marker for the goal value. The different graphics simplify the way the reader recognizes the more important value and the terms of comparison.

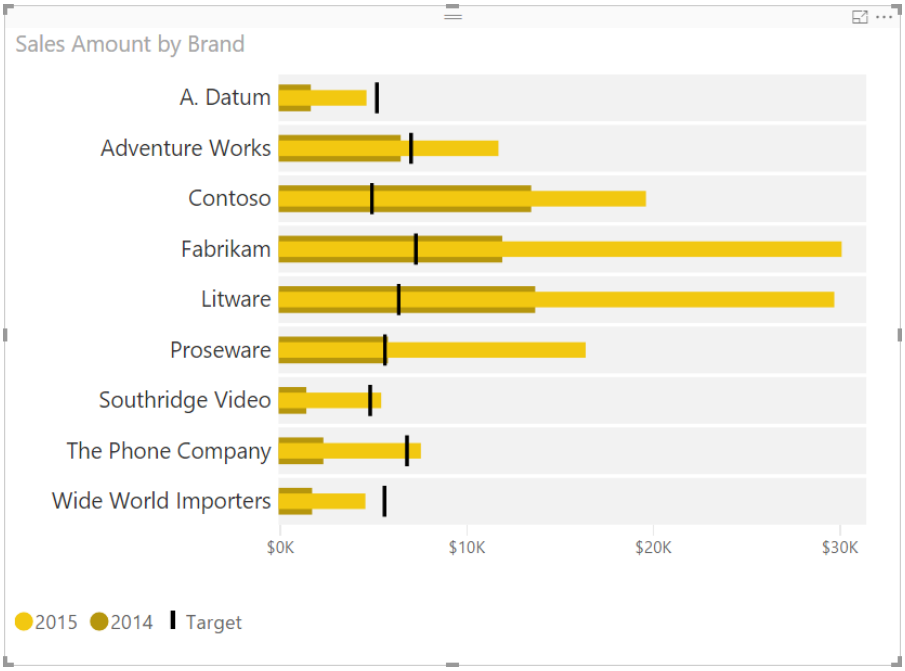


Figure 7-21: The Bullet Chart By SQLBI visualization, which displays actual sales amounts and goals by brand.

The only issue in this visualization is that we used black for the Target marker and yellow for the 2015 sales amount, which inverts the choice made in other charts of the same report. The reason for this is if we applied this color choice to other charts, it would have produced a barely readable marker for the goal value. In this case, the custom visualization can slightly improve the visualization, but it is not strictly necessary. Figure 7-22 demonstrates the final result of the Sample-Sales.pbix report, using the two custom visualizations that we've used thus far.

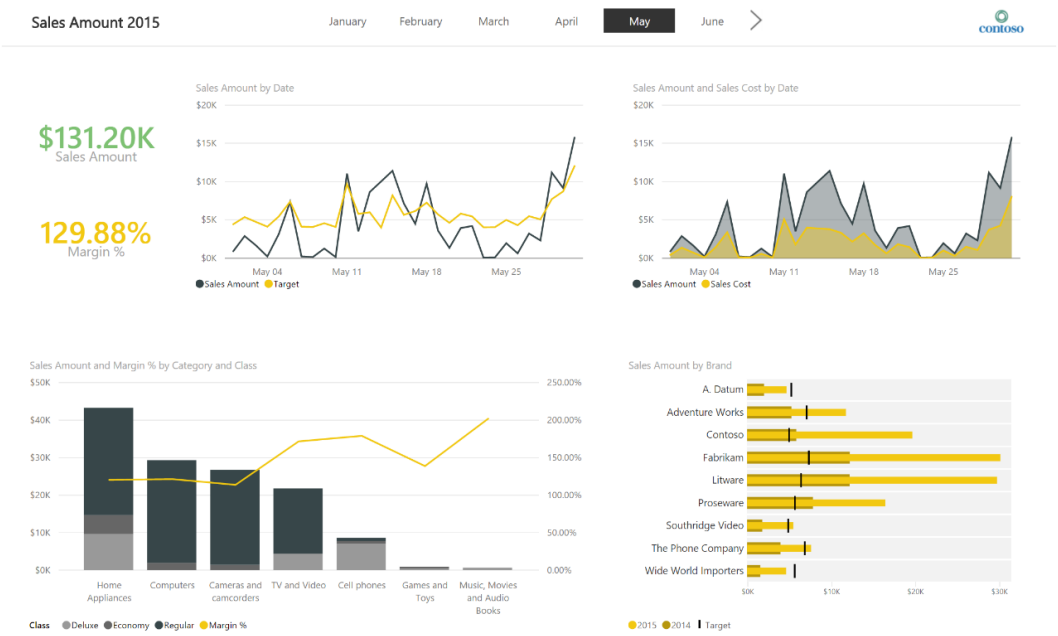


Figure 7-22: The final version of the report, displaying Contoso's sales in May, 2015. This version uses standard and custom visualizations.

Let's look at another example to see how using a custom visualization can improve a report. Figure 7-23 shows a report that represents the density of the population in each state within the United States, using the standard filled-map visualization. Darker shades correspond to higher-density values; lighter shades correspond to lower-density values.

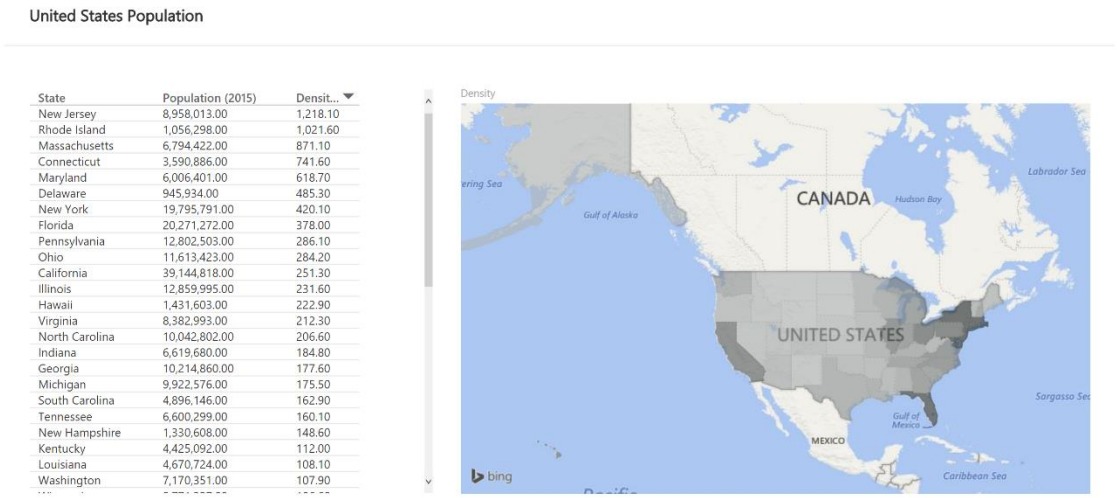


Figure 7-23: A report displaying population density using a filled-map visualization.

This particular visualization (on the right of Figure 7-23) does not include the name of each state, because more than 95 percent of the states are condensed in less than 20 percent of the available real estate in the chart, so there simply is not enough room.

You can represent the same information by using a custom map and moving Alaska and Hawaii in a different position, changing also their size and making the map more readable. You can do this by using the Synoptic Panel By SQLBI visualization component that you can find in the visualization gallery. With this component, you can draw custom areas over any map image. (There is a gallery of maps ready to use at http://synoptic.design.) Figure 7-24 shows where you can find and download a map of the United States that includes the state names.

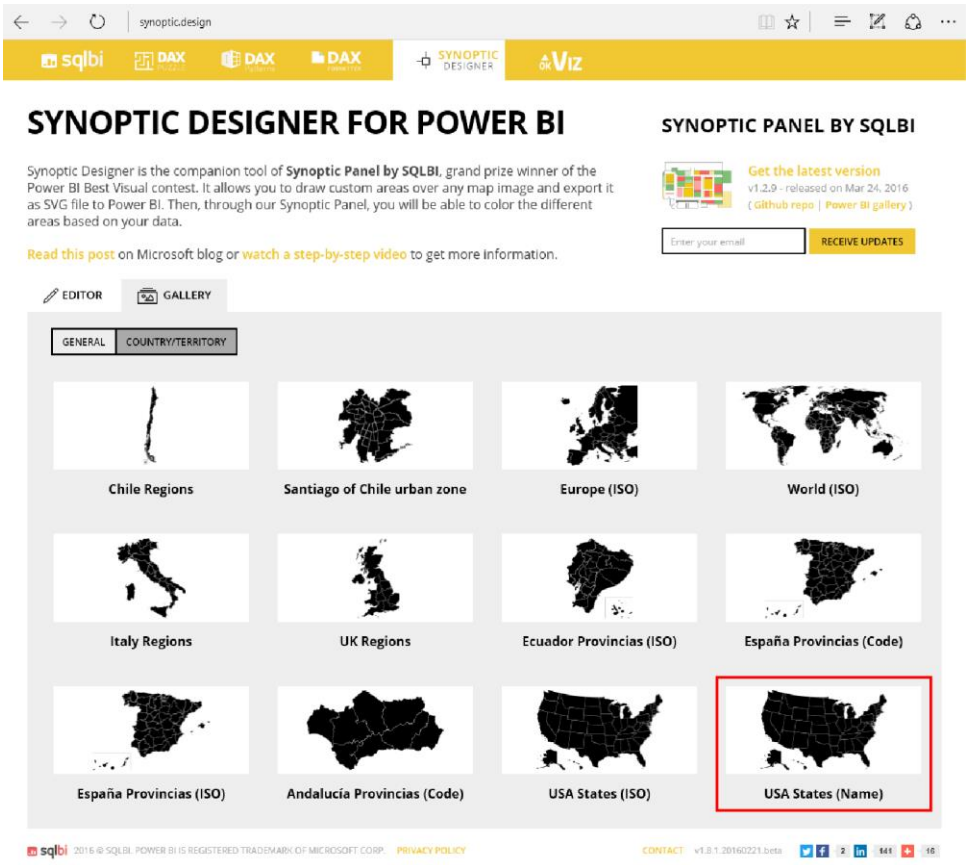


Figure 7-24: The gallery of country/region maps that are available on the Synoptic Designer website.

By using the Synoptic Design panel with the customized map of the United States, you can create the map shown in Figure 7-25, in which each state displays its name as well as its respective shade of gray. Moreover, the Synoptic Design panel also can work offline, whereas displaying the standard map component requires an active Internet connection to query the Bing service.

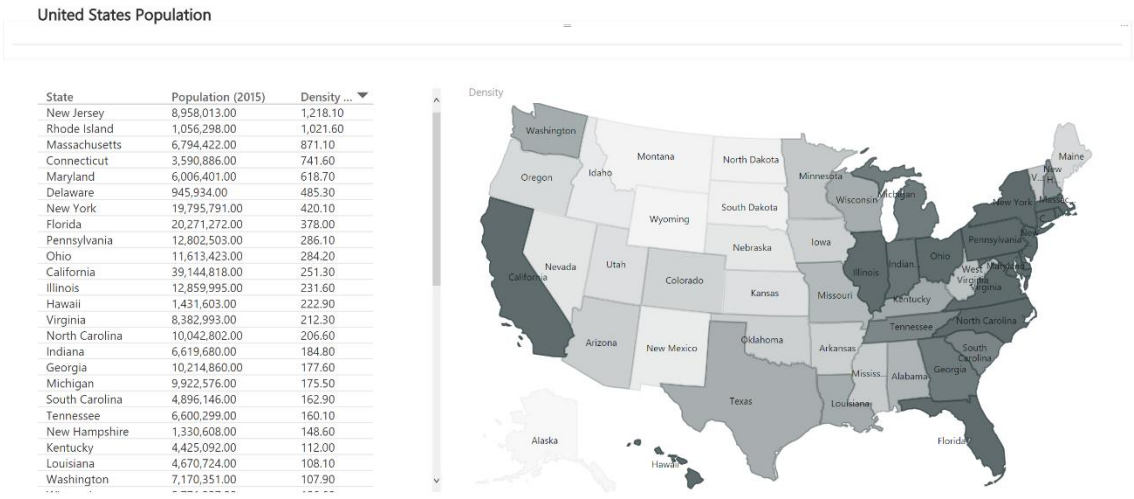


Figure 7-25: The same report displaying population density, but this time using a panel from Synoptic Design with a custom map of the United States.

Here again, you have seen how to improve a report by using a custom visualization, but until now we never had the need for a custom visualization to show the desired data. The standard components always offered an alternative way for you to display the same data that, even if not ideal, you could use if custom visualizations are not available. In the next section, you will see some examples for which a custom visualization is required to achieve a minimum goal.

Identifying conditions when custom visualizations are required

In this section, we use a different report to show conditions for which different visualization choices can produce more- or less-meaningful results. The report we want to consider shows the status of a stocks portfolio, using historical prices of stocks to display charts describing the behaviors of different shares and of the entire portfolio. Figure 7-26 depicts the initial version of this report.

Note You can find this report on the page Stocks in the Sample-Stocks.pbix file.

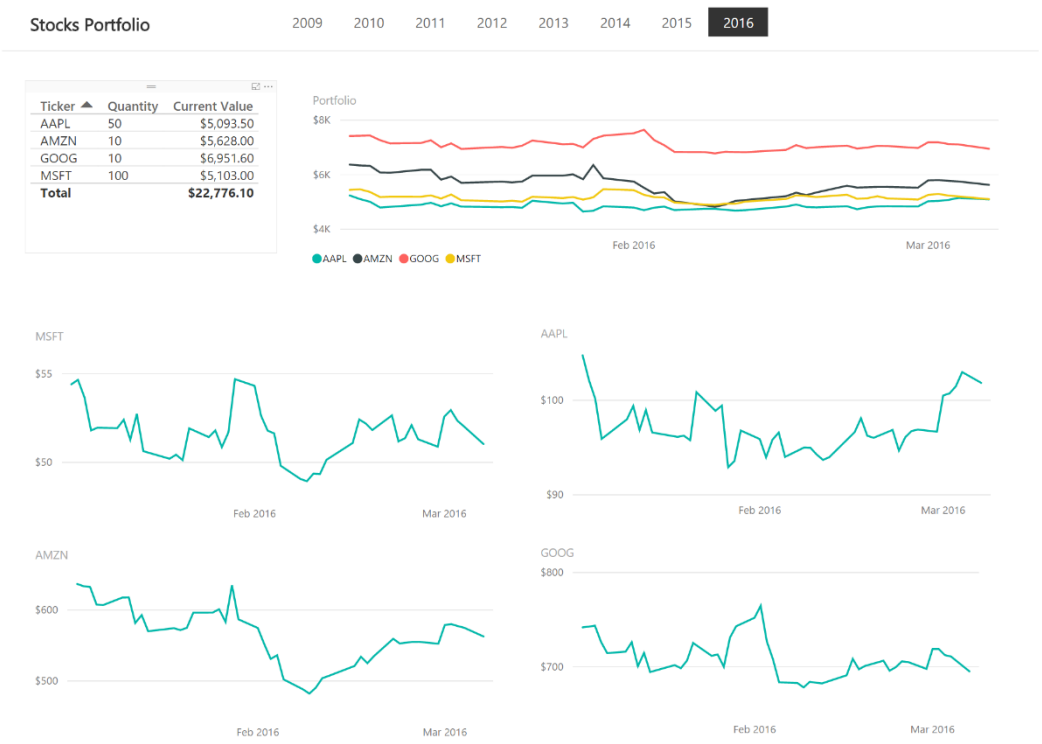


Figure 7-26: A report displaying stock portfolio performance over time using standard visualizations.

The report includes a line chart for each share, representing the closing price of the corresponding ticker. The details of the portfolio are included in a simple table in the upper-left corner, which displays for each share the quantity owned in the portfolio and its corresponding value at current prices. There is also a line chart for the entire portfolio, in which each share is represented by a single line of a different color. The line represents the total value of such a share in the portfolio over time. This chart can be useful to identify which stock has the largest value in the portfolio over time, but it does not provide in any way an indication of the total value of the portfolio over time. However, you can change this visualization to the stacked area chart depicted in Figure 7-27 to obtain a better representation of the portfolio's value.

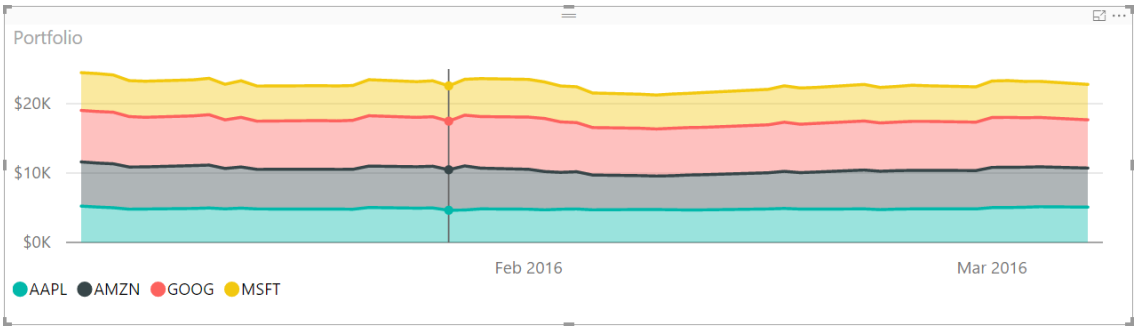


Figure 7-27: The same portfolio value shown in Figure 7-26, but now shown over time, divided by share name.

Every color represents a different share, so you also have a rough estimation of the weight of each stock in the portfolio. The main difference between a stacked area chart and a line chart is that the former cumulates the value of each share name, whereas the latter displays each share name individually. In this case, an accurate choice of the visualization between the existing ones can satisfy the presentation requirements. However, in the other charts, a standard Power BI visualization is not very useful.

The four charts displaying the stock price for each day do not provide complete information. The data model actually contains different price values for each day: open price, minimum price, maximum price, and close price. It is very common to display these four values for each period considered (a day, in this example) by using a candlestick chart. This chart type is not available in the standard Power BI visualizations, so you need to install a custom visualization for that. For example, the Candlestick by SQLBI visualization provides a basic visualization that includes four measures for each period: Open, Close, High, and Low. Figure 7-28 presents the final result of the report after applying the two changes described in this section.

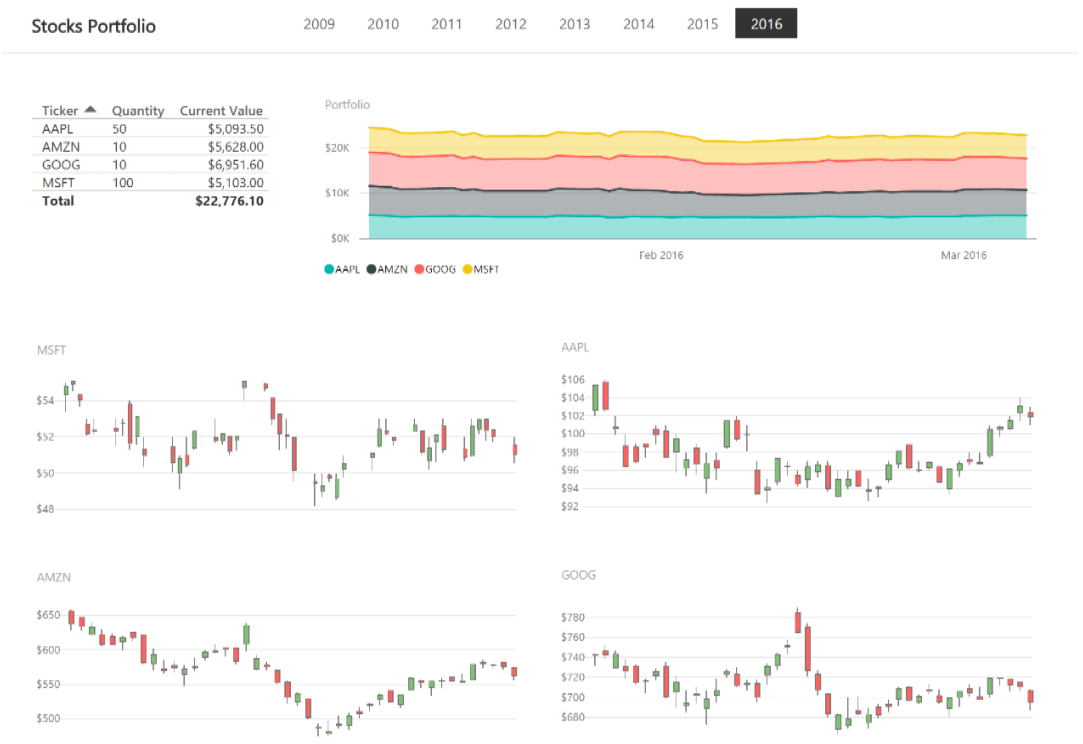


Figure 7-28: The same stock portfolio report displaying performance over time using custom visualizations.

You have seen how custom visualizations can improve the report, and sometimes they are necessary to achieve the desired graphical result. From time to time, though, you might still need to massage the data model to present measures and attributes to visualizations in the expected way, using the right granularity, and the expected formatting. The DAX language is your best friend here, as you will see in the next section.

Using DAX in data models

So far in this chapter, we have presented several examples of visualizations. We've demonstrated how you can improve your reports by choosing the right visualization, setting the properties in an appropriate way, and even installing custom visualizations when required. However, there are a number of improvements that you can achieve that do not require a direct action on the visualizations. You can instead create *measures* or calculated columns in DAX. Usually, you use DAX to obtain a certain numeric result, but sometimes you can take advantage of a DAX expression to control the report layout.

Our first example concerns the measures used in the candlestick charts of the report shown in Figure 7-28. Every time period displayed involves four measures: Open, Close, High, and Low. The data recorded in the data model has four corresponding columns for each day and stock. However, you might use the candlestick chart to display data by week or by month instead of by day. The aggregation required for each measure depends on the measure itself. The Open measure's price must be the DayOpen value of the first day in the period, the Close price must be the DayClose value of the last day in the period, the High price must be the maximum value of DayHigh in the period, and

the Low price must be the minimum value of DayLow in the period. You can write these four measures by using the following DAX expressions:

```
Open =
IF (
    HASONEVALUE( StocksPrices[Date] ),
    VALUES ( StocksPrices[DayOpen] ),
    CALCULATE ( VALUES ( StocksPrices[DayOpen] ), FIRSTDATE ( StocksPrices[Date] ) )
)

Close =
IF (
    HASONEVALUE( StocksPrices[Date] ),
    VALUES ( StocksPrices[DayClose] ),
    CALCULATE ( VALUES ( StocksPrices[DayClose] ), LASTDATE ( StocksPrices[Date] ) )
)

High = MAX ( StocksPrices[DayHigh] )

Low = MIN ( StocksPrices[DayLow] )
```

As mentioned earlier in this chapter, another useful tip is to create a measure just to display a measure with a different name. The reason is that many visualization components use the measure name in a legend or other description, and you do not have a way to rename that by using visualization properties. For instance, if you have two measures, Current and Previous, but you want to display the exact year in a particular visualization, you might create two measures with the exact year that you want to display in the legend, as in the following example:

```
[2014] = [Previous]

[2015] = [Current]
```

In a report that you will see in the next section, we will use data extracted from Google Analytics. The Website table contains the columns Users and Country ISO Code. In the dashboard, it will be interesting to show on a map the ratio between the number of users visiting a website and the population of the country/region where users come from, but such information is not available directly from Google Analytics. You can import the information about countries'/regions' populations in a separate Countries/Regions table and link it to the Website table using the Country ISO Code column, as illustrated in Figure 7-29.

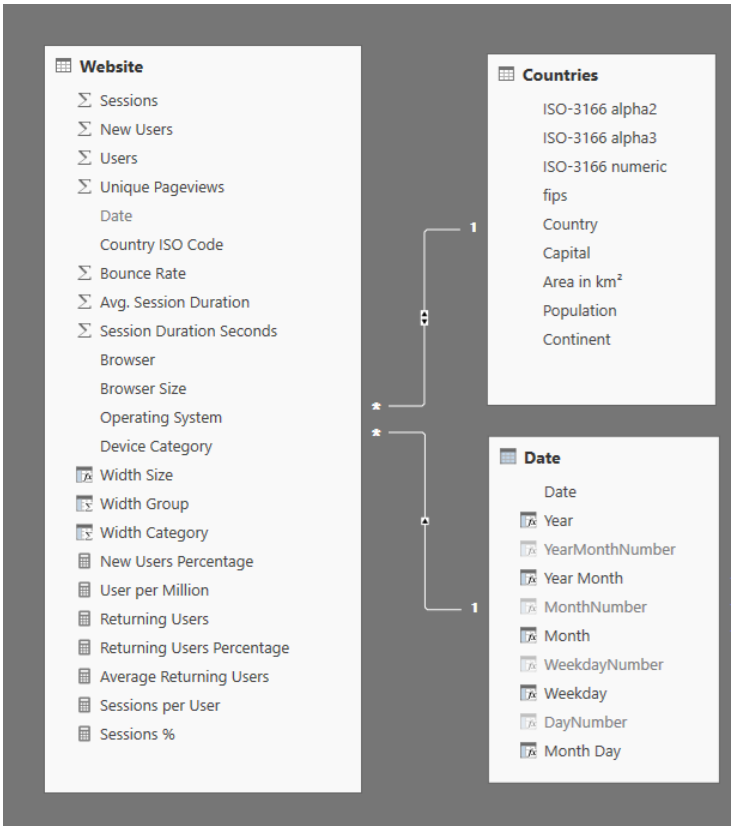


Figure 7-29: A schema of tables and relationships of a data model that extends Google Analytics website data.

Because the ratio would be a decimal number, you can create a metric called Users Per Million by using the following measure definition in DAX:

```
Users per Million =
DIVIDE (
    SUM ( 'Website'[Users] ),
    SUM ( 'Countries'[Population] )
) * 1000000
```

You should not expect that a visualization component would directly do a calculation, such as a ratio or a difference. It is always better to define the corresponding calculation in a DAX measure, and then you bind that measure to the visualization.

Finally, you should consider using a calculated column when you need a classification that groups existing data with a high granularity to a lower number of unique values that are easier to display in a chart. For example, Figure 7-30 shows the existing values in the Browser Size column that is provided by Google Analytics. There are more than 5,000 unique combinations of width and height, and this fragmentation of values makes the analysis harder. Moreover, each resolution is a single string, and the reports should group different resolutions by width, ignoring the height.

Browser Size
1260x680
1350x670
1520x770
1660x920
1010x670
1590x800
1350x650
1420x750

Figure 7-30: A partial list of values included in the Browser Size column.

You can split the problem into two steps. First, you extract the width size from the string, converting the digits before the "x" character in an integer number. Then, you compare this number with a list of predefined values representing the interesting range of resolutions you want to analyze, such as 1024, 1280, 1440, 1920, and 2560. You can see the two calculations implemented in the following two measures, Width Size and Width Category, respectively:

```
Width Size =
VAR xPos = FIND ( "x", Website[Browser Size], , 0 )
VAR widthString = IF ( xPos > 1, LEFT ( Website[Browser Size] , xPos - 1 ), "" )
RETURN IF ( widthString <> "", INT ( widthString ) )

Width Category =
SWITCH (
    TRUE(),
    Website[Width Size] <= 1024, 1024,
    Website[Width Size] <= 1280, 1280,
    Website[Width Size] <= 1440, 1440,
    Website[Width Size] <= 1920, 1920,
    Website[Width Size] <= 2560, 2560,
    CALCULATE ( MAX ( Website[Width Size] ), ALL ( Website ) )
)
```

Figure 7-31 presents the results of the two calculated columns. We will use the Width Category in one of the visualizations described in the next section.

Browser Size	Width Size	Width Category
1260x680	*1260*	*1280*
1350x670	*1350*	*1440*
1520x770	*1520*	*1920*
1660x920	*1660*	*1920*
1010x670	*1010*	*1024*
1590x800	*1590*	*1920*
1350x650	*1350*	*1440*
1420x750	*1420*	*1440*

Figure 7-31: A partial list of values included in the Browser Size, Width Size, and Width Category columns.

Creating high-density reports

In the last section of this chapter, we want to discuss the challenges that you face when creating high-density reports. When you include many visualizations in a single report, you need to carefully balance the amount of information provided in each of those visualizations, removing all the unnecessary elements that would reduce the attention of the user. You want your user to focus only on the data.

As you will see, having an idea of the overall structure and tuning properties of each visualization is much more important than using particular custom visualizations, which are usually the icing on the cake.

Figure 7-32 depicts a first version of a report showing website data from Google Analytics for the DAX Formatter website, which is available in the companion content, in the file Sample-DAXFormatter-Analytics.pbix file. The report contains 28 visualizations with data, plus one slicer and eight components without data that are there only for aesthetic reasons (title, logo, pictures, and separators). The 28 visualizations only use 7 different visualization types, and some of them are simple variations of the same concept, such as stacked/clustered bar/column charts. You can obtain a complete and complex report using only a few component types.

Figure 7-32: A high-density report based on Google Analytics data.

The entire report is organized in three zones: left, center, and right. The left zone contains metrics regarding the number of users, the center zone shows data about the sessions, and the right zone includes technical information about average page load time, device type, operating system, browser, and resolution used by website visitors.

If you were to try to create a similar report starting from scratch, you would need to apply the following guidelines:

- **Reduce text** Include only the minimum necessary of textual elements, avoiding repetitive or verbose descriptions.
- **Remove legends** Whenever possible, avoid including a legend, especially whenever there is only one measure displayed in the chart.

- **Remove axes** In a compact visualization for which you already included the data labels (setting the Data Labels property to On), you can remove corresponding axes. All the clustered bar charts in the report are formatted in this way.
- **Use images to explain concepts** Use an icon or a meaningful image related to the data you show. Remember, a picture is worth a thousand words. In the example in Figure 7-32, at the top of the report is one different image for each of the three zones (Users, Sessions, and Average Page Load Time).

In the report, we used a donut chart. Previously in this chapter, we mentioned that using a pie chart or a donut chart is not a good idea, so you should avoid doing that. The exception we wanted to include in this report is when you compare only two values. In this example, we used a donut chart for the search engines distribution, showing the percentage of sessions coming from Bing searches versus Google searches. It is clear that Google has a clear leadership for directing visitors to this website, with Bing producing only a marginal contribution. For this difference, looking at the exact number or percentage is not relevant.

There are three visualizations that we wanted to improve, starting from this initial example. The sparklines used at the top of the report are simple line charts without any axes, legend, data labels, or border. Figure 7-33 illustrates that we set to Off most of the format properties of those line charts.

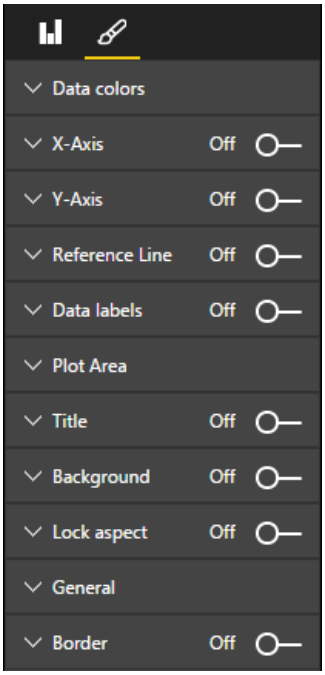


Figure 7-33: The Properties pane of a line chart used as a sparkline.

However, you cannot change the line width of the line chart. When you use the line chart in a small area, this produces a result that is difficult to read. You can replace the line chart with the Sparkline custom visualization that you can download from the Power BI visualization gallery. Figure 7-34 shows a side-by-side comparison of the same chart displayed by using a line chart (on the left) and a sparkline custom visualization (on the right). The custom visualization draws a line with a smaller width, generating a final result that is more readable.

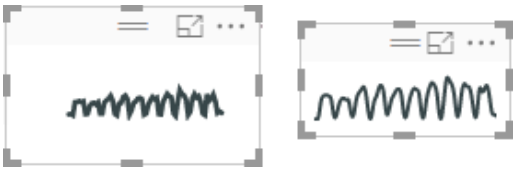


Figure 7-34: A couple of examples of the same chart, using a standard line chart and a sparkline custom visualization.

Another visualization to improve is the countries/regions penetration displayed in the lower-left corner of the report. Instead of using a standard map, which shows a pie for each country/region with a size depending on the population density, you can use the Synoptic Designs panel, loading the world map from the gallery shown back in Figure 7-24. The result of displaying the measure Users Per Million in a Synoptic Designs pane is visible in Figure 7-35.

Figure 7-35: Synoptic Designs panel with a world map that displays the ratio between users and population.

The last visualization to improve is the one in the lower-right corner, showing the number of sessions by browser resolution. In the original data, we have a very fragmented number of different resolutions, so the bar chart only displays the first values, but the number of sessions of the most common resolution is just five percent of the total number of sessions, so there are a wide number of different resolutions considered, most of them not visible in the report. We solved this problem in two steps. First, we created a column containing the width category, which classifies the width extracted from the resolution string, as you have seen in the previous section about DAX. Then, we changed the clustered bar chart to a waterfall chart, using the Sessions % measure instead of the Sessions measure, as shown in Figure 7-36.

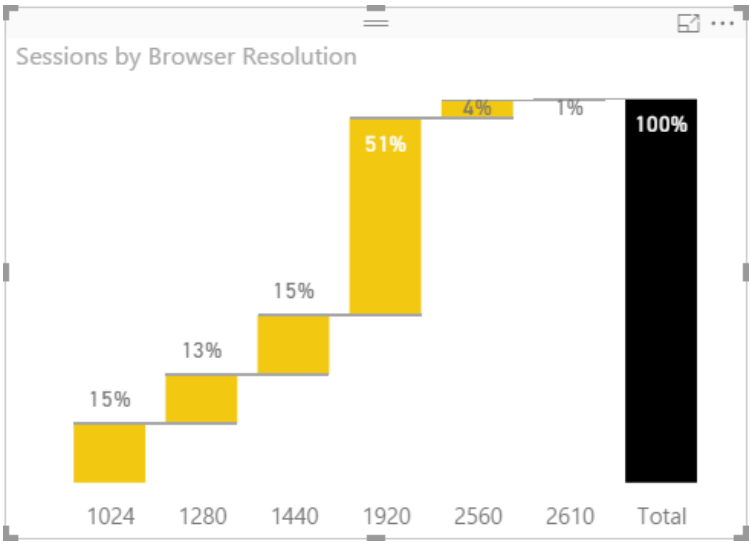


Figure 7-36: A waterfall chart with the distribution of sessions by browser resolution.

We do not use any decreasing step in the waterfall chart, but the final result clearly shows the distribution of the resolution in a meaningful way. The Sessions % measure is a DAX expression created just for this report, using the following definition:

```
Sessions % =
DIVIDE (
    SUM ( Website[Sessions] ),
    CALCULATE (
        SUM ( Website[Sessions] ),
        ALL ( Website[Width Category] )
    )
)
```

Figure 7-37 presents the final result, after these improvements have been incorporated. As you can see, the overall difference is an incremental improvement and not a substantial change from the result we got initially by using standard visualization components.

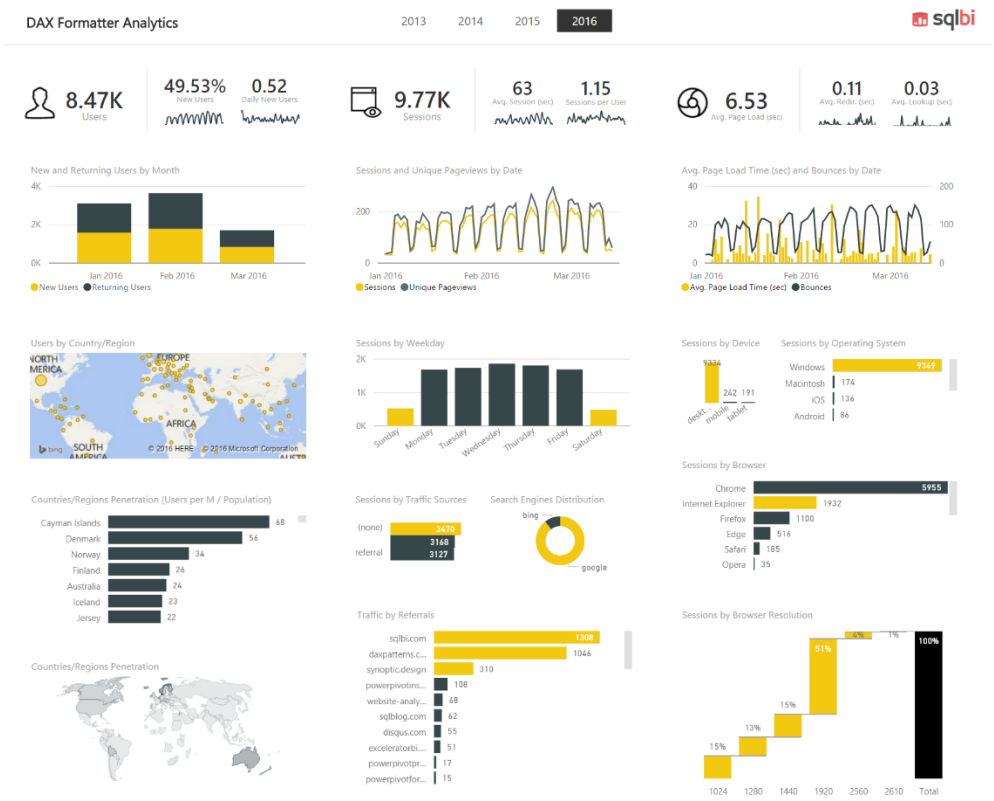


Figure 7-37: A high-density report based on Google Analytics data.

Especially in a high-density report, you should focus on the overall quality and readability, reducing the number of distractions for the reader. There is already an overwhelming amount of information, so the focus of the reader should be entirely on data, not decorations or visualizations that are too complex and do not provide any added value.

Conclusions

In this chapter, you have seen a number of techniques to improve Power BI reports by choosing the best built-in visualizations and adding custom visualizations. Here are the main steps in this process:

- **Choose the right visualization type** You can choose between many visualization types, but usually you do not need too many of them in the same report. Do not be afraid of using the same visualization type many times if it is the one that best displays your data.
- **Customize visualization properties** You can customize every visualization by using format properties. Using a consistent color scheme is one of the most important aspects of a good report.
- **Consider custom visualizations when necessary** The Power BI custom visualizations gallery provides you with many visualizations that extend the set of standard ones available in Power BI. You should consider using them when there is a concrete advantage over the standard visualizations.

- **Use DAX to create measures and calculated columns** You also can use DAX expressions to achieve the desired visualization. Even a simple transformation, such as renaming a measure in a legend, is not always possible within the properties of a visualization, but you can overcome this by creating new measures and calculated columns.
- **Remove unnecessary elements in high density reports** In a high-density report, you need to remove any graphical element that is not necessary to communicate information to the users; you do not want to distract them by including details that do not provide any useful information.

These guidelines are just a starting point in your journey to create clear and useful reports. Your experience, the feedback from users of your reports, and the analysis of reports created by other people are the other important steps along this road.

CHAPTER 8

Using Microsoft Power BI in your company

With Power BI, you can create a dashboard using some data in a Microsoft Excel workbook, or you can connect to existing structured databases in your company. The previous chapters in this book demonstrate how you can create data models, reports, and dashboards, gathering information from different data sources. They also show you how you can consume these results on different devices. This chapter shows you techniques to obtain a deeper integration with existing systems and applications.

You can get data from many existing data sources in your company. You can embed a dashboard visualization in Microsoft Office with minimal effort. If you have developer skills, you also can take advantage of the REST API to automate operations in Power BI by implementing real-time updates of dashboards, with no more latency than just a few seconds.

Power BI is a platform that exposes several services through a REST API that is easy to use by any application, including standard web-based applications and those running on mobile devices. In this chapter, we will look at some example results that will help you to understand why the presence of this extensibility option is so important. The goal is not to describe in detail how to use these APIs from a developer's point of view, but to make you aware of what you can obtain by using them. If you are a developer, you will find some links and references to additional material to understand how to use the existing API. If you are a business user or a BI architect, you will gain a fuller understanding and know what you can and cannot ask a developer. However, keep in mind that the API for Power BI

is undergoing constant evolution. If something is not possible as of this writing, it might become possible in a future release. When in doubt, check what new features are available in the API.

Getting data from existing systems

Any company has a number of existing data sources that you can use in Power BI. You have seen that you can create a data model in Power BI by copying the content of tables that exist in other databases or files. You also have the option of refreshing this content dynamically, or you can directly query the data source whenever you access a report. By querying directly, you avoid the need to create a copy of the data that you must then synchronize periodically. In this section, you will see the available options with which you can connect Power BI to either your on-premises database or a database in the cloud.

Before looking at the details, here are a few terms with which you should be familiar:

- **On-premises** If you get data from a database that is physically stored in a server managed by your company, we say that the database is on-premises (often shortened to on-prem).
- **Cloud** If you get data from a Microsoft Azure service, you are using data in the cloud. Cloud computing accesses and uses shared compute and storage resources on the Internet.
- **Relational database** This is a database that stores data using tables that have relationships with one another. Typically, you query this by using the SQL language. Examples of on-premises relational databases that Power BI supports are Microsoft SQL Server, Microsoft Access, Oracle, IBM DB2, MySQL, PostgreSQL, Sybase, and Teradata. Cloud-based relational databases that Power BI supports include Azure SQL Database and Azure SQL Data Warehouse.
- **Rich semantic model** This is a database that stores both data and metadata, simplifying navigation by using tools such as Excel PivotTables and Power BI reports. A typical example is Microsoft SQL Server Analysis Services. Other supported providers are SAP HANA and SAP Business Warehouse.
- **Power BI Personal Gateway** This is a component installed on the user's computer that makes it possible to perform data refreshes on models published using the Power BI service. (Chapter 3 explains how to install this.) A Personal Gateway serves only one user, and only when the user's computer is turned on.
- **Power BI Enterprise Gateway** This is a component similar to the Personal Gateway that a system administrator installs on a server in your company. A single Enterprise Gateway can serve all the users of a company, and it is also available as soon as the server is turned on (servers are usually active 24/7). You can find more technical details about how to install it at https://powerbi.microsoft.com/documentation/powerbi-gateway-enterprise/.

There is also another concept to clarify before moving forward, which is the difference between a model requiring data refresh and a live connection.

Understanding differences between data refresh and live connections

When you navigate in data via Power BI, you can read a copy of the data stored in Power BI (either the Power BI service or the Power BI Desktop application), or you can have a live connection that sends a real-time query to the data source without creating a copy of the data.

Chapter 4 shows that when you connect Power BI to a SQL Server database, you have two connection settings from which to choose: Import and DirectQuery. The DirectQuery option does not create a

copy of the data in Power BI, and it translates any user action made navigating on a report into one or more queries to SQL Server. In a more general classification, we can say that you can either import data in Power BI or connect to the data source via a "live connection." With relational databases, DirectQuery is the tool used to obtain a live connection to the data source. As you will see later in this chapter, in Power BI you have similar options when you connect to a SQL Server Analysis Services data source: Connect Live and Import Data.

Regardless of the underlying database, when you create a model in Power BI by importing data, you have full access to the features of Power BI. However, you need to run or schedule a data refresh to keep data updated on Power BI.

On the other hand, when you use a live connection, your Power BI model can have only one data source, so a single Power BI report cannot mix visualizations connected to data coming from different data sources. To do that, you must import data into Power BI. In a dashboard, however, you can always include visualizations from different reports; thus, you can combine visualizations from different live connections in a dashboard only.

In the following sections, you will see in more detail how to use live and imported data sources using existing databases and models in your company.

Using relational databases on-premises

When you create a data model in Power BI Desktop, you often get data from an on-premises relational database. For example, Chapter 4 demonstrates how to create a new Power BI Desktop data model that gets data from a Microsoft SQL Server database. In this case, you use a database on-premises, and you can choose between the Import and DirectQuery connection types. The former creates a copy of the data in the Power BI Desktop model, which requires a refresh in order to synchronize the content of the Power BI data model with the source database. The latter does not create a copy of the data; instead, Power BI generates queries to SQL Server every time you navigate in a report.

In both cases, after you publish the Power BI Desktop file to the Power BI service, the refresh operation requires either a Personal Gateway or an Enterprise Gateway. If you use the Personal Gateway, you can only refresh datasets created by importing data, but you cannot use the DirectQuery option. To publish a Power BI Desktop data model created by using DirectQuery, you need the Power BI Enterprise Gateway.

Figure 8-1 illustrates what happens when you publish a Power BI data model connected to on-premises data sources via the Import connection type. A copy of the data and the description of the data model are stored in the Microsoft cloud. The data is always available to queries sent by any client, which sees data updated on the last data refresh. You need a Power BI gateway installed on-premises to complete the data refresh: either a Personal Gateway or an Enterprise Gateway, in this scenario.

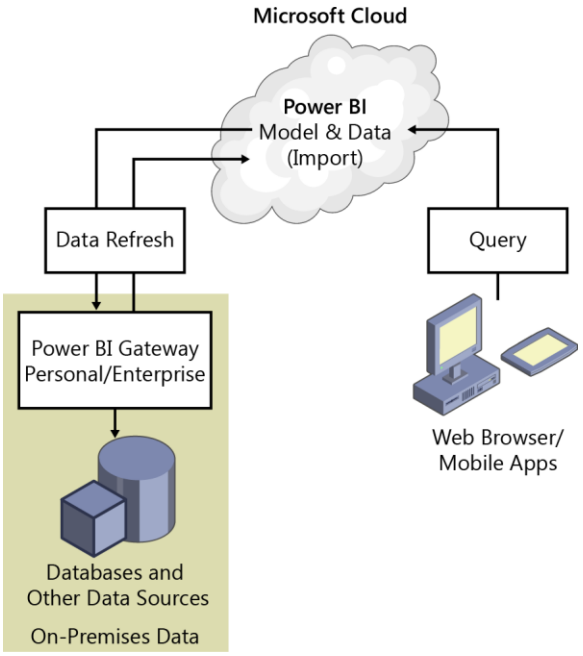


Figure 8-1: A Power BI gateway is required to refresh a Power BI model, getting data from on-premises databases.

Figure 8-2 shows what happens when you publish a Power BI model created by using the DirectQuery option. The Power BI service does not store a copy of data in the Microsoft cloud; it has only a semantic description of the data model, with the information required to retrieve data from the original source database. Every time the Power BI service receives a query, it generates one or more queries in SQL language and sends these requests to the relational data source through the Power BI Enterprise Gateway.

More info The Microsoft cloud service does not preserve any data received from the relational databases on-premises; it might only keep a transient data cache on a volatile device in order to improve the performance of other queries sent by the same user. You can find more information in the Power BI Security whitepaper published by Microsoft at http://download.microsoft.com/download/4/8/C/48CFCF8A-2025-4B97-B249-7B505E26E7ED/Power%20BI%20Security%20Whitepaper.docx.

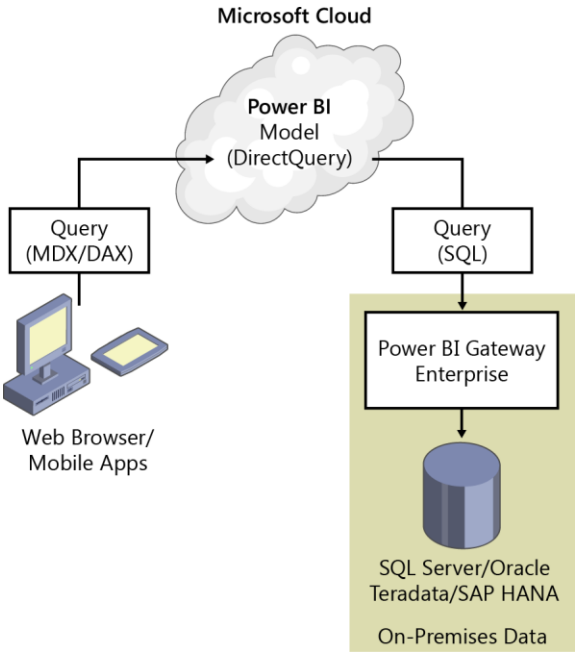


Figure 8-2: For on-premises data sources, you must have the Power BI Enterprise Gateway for a model using the DirectQuery connection setting.

As of April 2016, you can take advantage of DirectQuery on SQL Server, Oracle, or Teradata relational databases, which are all supported in the Power BI Enterprise Gateway.

Using relational databases in the cloud

If you create a Power BI model that uses a relational database stored in the cloud, you might not need the Power BI Gateway to refresh data. Power BI supports direct connection to Azure SQL Database and Azure SQL Data Warehouse data sources, so you can schedule a data refresh or you can use DirectQuery without the need to install and configure any gateway.

You will still have a different architecture, depending on which connection setting you use, Import or DirectQuery. Figure 8-3 illustrates that by using the Import setting you still have a copy of data owned by the Power BI service, but you can refresh that copy without any gateway if the data source is Azure SQL Database or Azure SQL Data Warehouse.

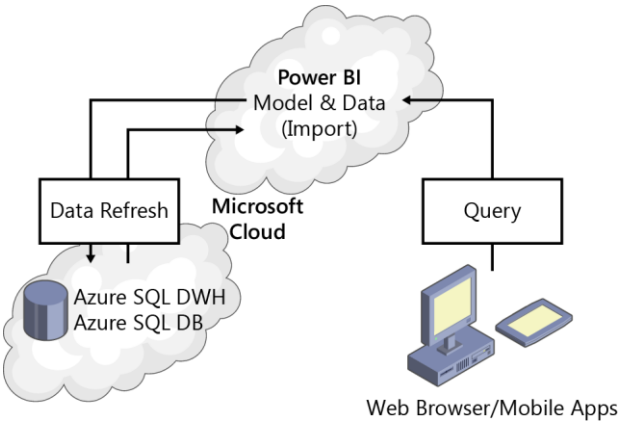


Figure 8-3: Power BI can connect directly to Azure SQL Database and Azure SQL Data Warehouse to refresh a Power BI model.

Figure 8-4 illustrates the behavior of Power BI using DirectQuery connected to Azure SQL Database or Azure SQL Data Warehouse. As with any DirectQuery connection, the Power BI service has only a semantic description of the data model, along with the information required to retrieve data from the original source database. It does not store a copy of data in the Microsoft cloud. Every time the Power BI service receives a query, Power BI generates one or more SQL queries and sends these requests to the relational data source, with no gateway required.

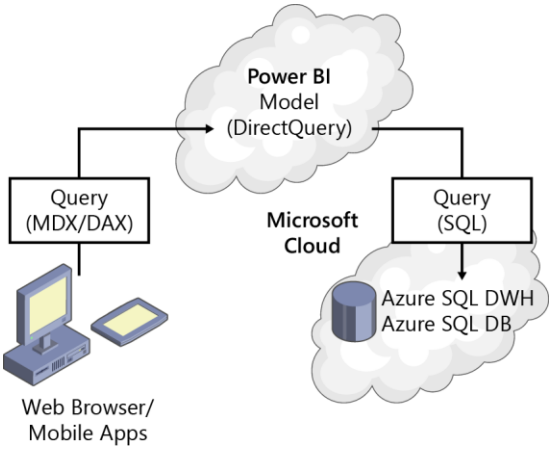


Figure 8-4: In DirectQuery mode, Power BI connects directly to Azure SQL Database and Azure SQL Data Warehouse.

If you have a cloud-based relational database, other than Azure SQL Database and Azure SQL Data Warehouse, you must use the architecture for on-premises data, and you need to install a gateway to complete the refresh operation or to implement DirectQuery.

> **Note** Multiple requests on different servers in different locations might increase the latency of requests, so you might want to consider installing the Enterprise Gateway on a server hosted in Azure Virtual Machines to improve the performance.

Using live connections to Analysis Services

When you create a live connection to Analysis Services in Power BI Desktop, you do not create a data model, and you do not have a copy of the data in the PBIX file. Thus, when you publish the model on the Power BI service, the PBIX file contains only the definition of the reports, but the entities are defined in the Analysis Services file. When you edit a report in Power BI that is tied to a live connection to Analysis Services, all of the operations requested by the client are redirected to Analysis Services through the Power BI Enterprise Gateway, as shown in Figure 8-5.

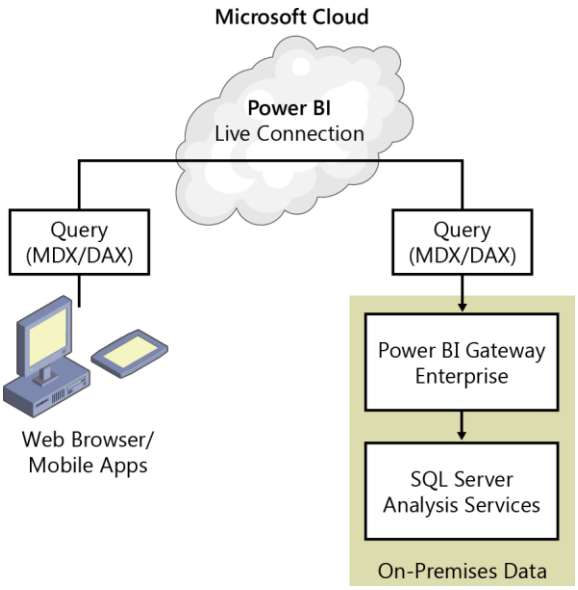


Figure 8-5: When a model has a live connection, Power BI redirects a query to Analysis Services on-premises.

The Power BI service contains neither data nor metadata for the data model. Any change made to the data model in Analysis Services is automatically reflected in Power BI, without requiring a data refresh operation in Power BI. However, keep in mind that usually Analysis Services has a copy of the data read from one or more data sources (unless you create a data model in Analysis Services by using DirectQuery), so you need to refresh the model in Analysis Services to keep it up to date.

Integrating Power BI with Office

Showing dashboards and reports created in Power BI using one of the options available (web browser, dedicated apps on mobile devices, and Power BI Desktop) is very useful. However, perhaps you want to create a particular report in Excel or a presentation in Microsoft PowerPoint, which could benefit from a tighter integration with Power BI. As you will see in this section, there are several features in Power BI that can take advantage of such a service in certain applications of Office.

Publish Excel data models in Power BI

When you publish a Power BI Desktop file to Power BI, you are copying to the cloud a file containing a data model, a copy of the data, the query to import and refresh the data, and all the reports you created. If you have an Excel file with a data model, you have a similar file, and you can publish such a file on Power BI, as well. In fact, the following correspondence exists between features in Power BI and Excel:

- Power BI data model → Excel Data Model (also known as Power Pivot data model)
- Power BI Query Editor → Workbook Queries (formerly known as Power Query in Excel 2010 and Excel 2013)
- Power BI report → Power View

You can load into Power BI an XLSX file containing a data model instead of a .pbix file. In doing this, you keep all of the existing features and reports in your Excel file, but you also can then use the same data model in Power BI. All the existing reports in Power View are converted in equivalent reports in Power BI whenever possible (certain features of Power View might not have a corresponding

visualization or feature in Power BI). In this scenario, the PivotTables and PivotCharts you have in Excel continue to work with the Power Pivot data model. Figure 8-6 presents the Publish To Power BI feature that is available in Excel 2016, which guides you in publishing a Power Pivot data model to Power BI without even opening the Power BI website, similar to how you would publish within Power BI Desktop.

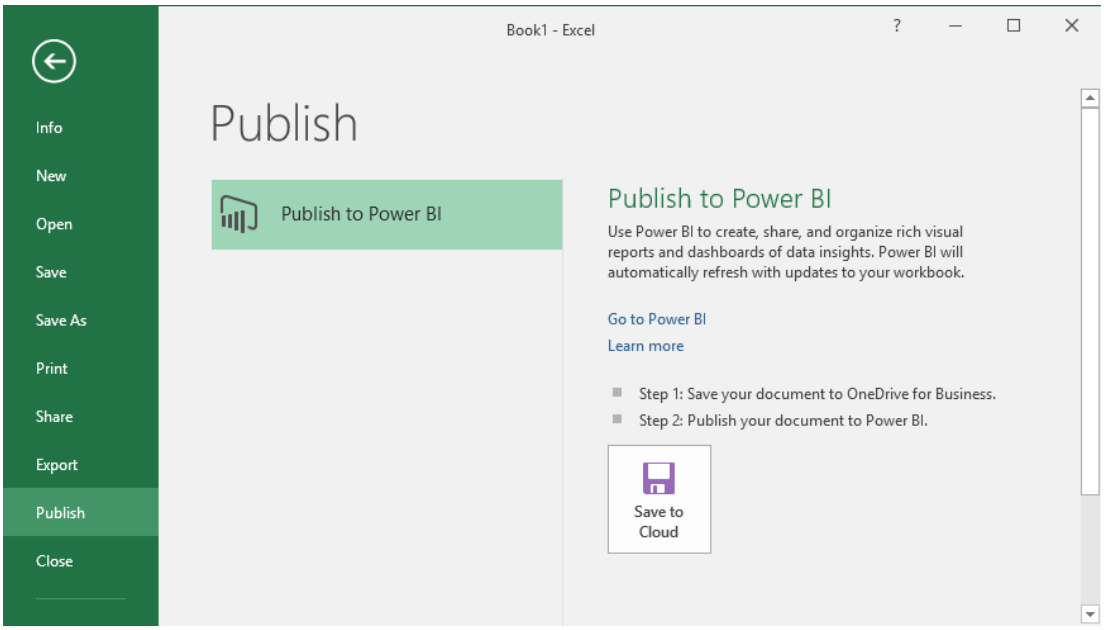


Figure 8-6: The Publish To Power BI feature in Excel 2016.

It is important to consider that, if you import the Power Pivot data model in Power BI Desktop, you will have a .pbix file instead of an .xlsx file. In this way, you can work locally with Power BI, but you will have two copies of the same data model and the same data. You will use Excel to navigate data with PivotTables and PivotCharts, and you will use Power BI to navigate using dashboards and reports. However, if you publish a .pbix file, you also can consume your data in Excel using the Analyze In Excel feature, which is described in the next section.

Consume Power BI content from Excel

In this book, you have seen how to import data from Excel to Power BI by using different techniques. However, you might want to move data in the opposite direction, consuming in Excel data that is published on Power BI. This is indeed possible, and you can do it by using the Analyze In Excel feature that is available in the Power BI service. Figure 8-7 depicts the Analyze In Excel action that is available for datasets and reports.

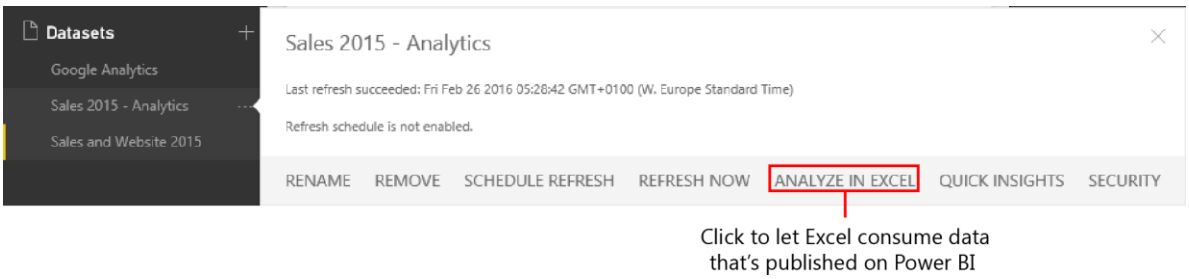


Figure 8-7: The Analyze In Excel feature in Power BI.

When you click Analyze In Excel, you might be prompted to install an updated driver for Excel; if this happens, follow the instructions to install the suggested driver. In any case, this action downloads a small file on your computer (with an .odc extension) that Excel uses to open a connection to the corresponding Power BI model, creating a PivotTable on top of the model. Figure 8-8 shows the result if you request Analyze In Excel on the dataset Google Analytics.

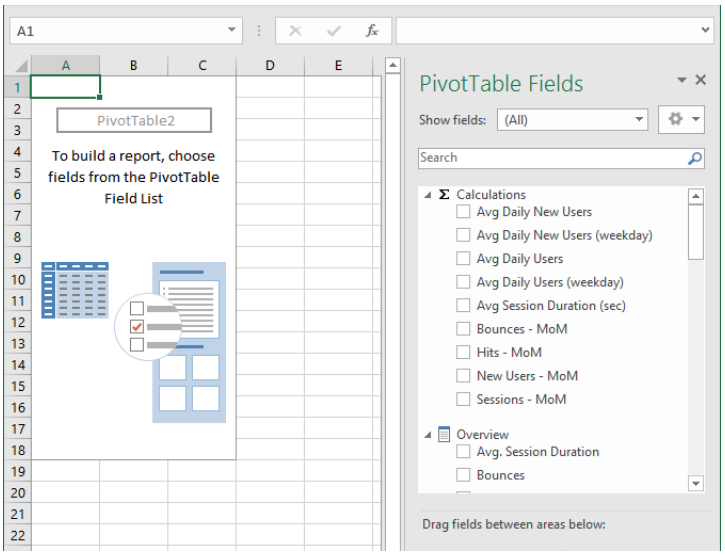


Figure 8-8: A PivotTable in Excel that is connected to the Google Analytics model in Power BI by using Analyze In Excel.

> **Note** By establishing a connection using Analyze In Excel, you are consuming your Power BI model as if it were an external analytical database. This is the same behavior you have in Excel when you connect to an Analysis Services database, or you connect Excel to a Power Pivot model published on SharePoint by using the Excel document URL as the server name. The Analyze In Excel feature uses Excel only as a client, without storing the data model within Excel, as you would do when you are using Power Pivot for Excel.

After you establish the connection with a PivotTable, you have all the tables, columns, and measures of the Power BI data model listed in the PivotTable fields. You can use only the measures that have been explicitly defined in the data model, so you cannot create measures during the navigation as you can do in Power BI. For this reason, it is important that you create all the measures that could be necessary to an Excel user, without assuming that any numeric column can be aggregated or that a measure can be created upon request. For example, the Sales 2015 – Analytics model that David created in Chapter 5 does not contain any explicit measures, and this makes it difficult to use in Excel. Figure 8-9 illustrates how the data of such a model is presented in the Excel PivotTable Fields pane.

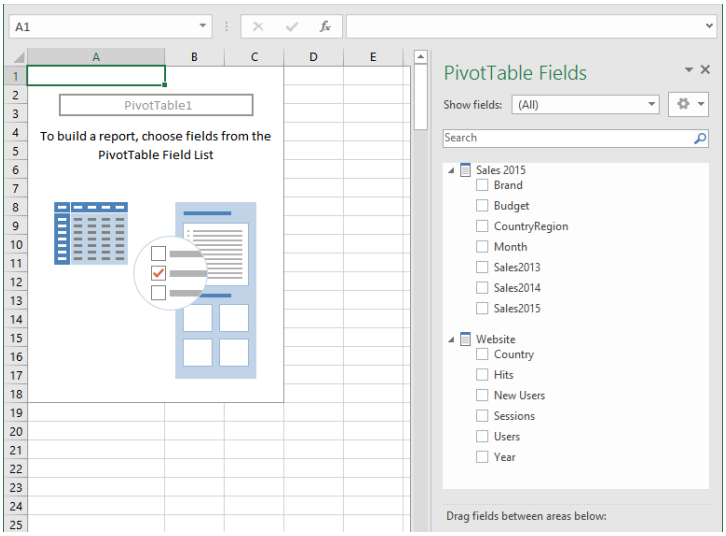


Figure 8-9: A PivotTable in Excel that is connected to the Sales 2015 - Analytics model in Power BI by using Analyze In Excel.

The model without explicit measures is not very useful in Excel, because you do not have any calculations to put in the Values area of the PivotTable. For example, Figure 8-10 shows the result of selecting the columns CountryRegion, Sales2013, and Sales2014 from the PivotTable Fields pane. These columns are placed in the Rows area of the PivotTable, and you cannot move them in the Values area. Thus, the result is a list of all the unique values of these columns, grouped by CountryRegion, Sales2013, and Sales2014.

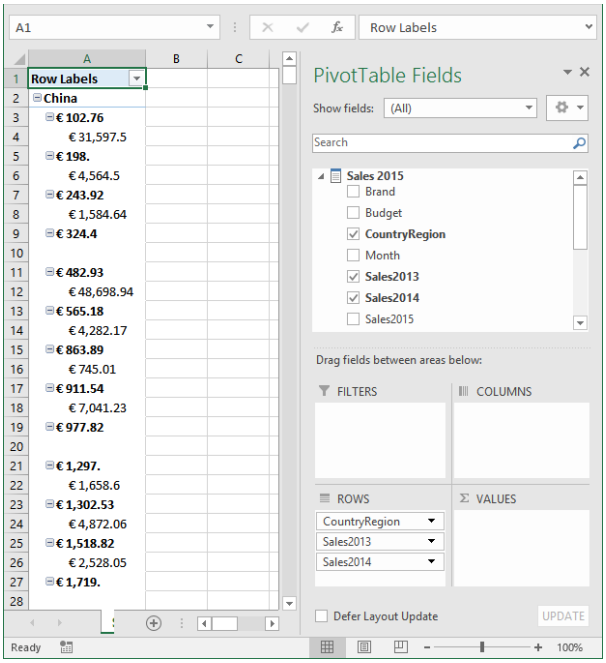


Figure 8-10: A PivotTable in Excel that is connected to the Sales 2015 - Analytics model in Power BI by using Analyze In Excel.

Excel ignores that the nature of the sales columns is that of numbers that can be aggregated. This information must be provided by using explicit measures, such as those demonstrated in Chapter 6. For example, Figure 8-11 shows a PivotTable obtained by using Analyze In Excel on the Budget data

model created in Chapter 6 and published on Power BI. In this case, the measure Sales Amount is aggregated by Brand and Year.

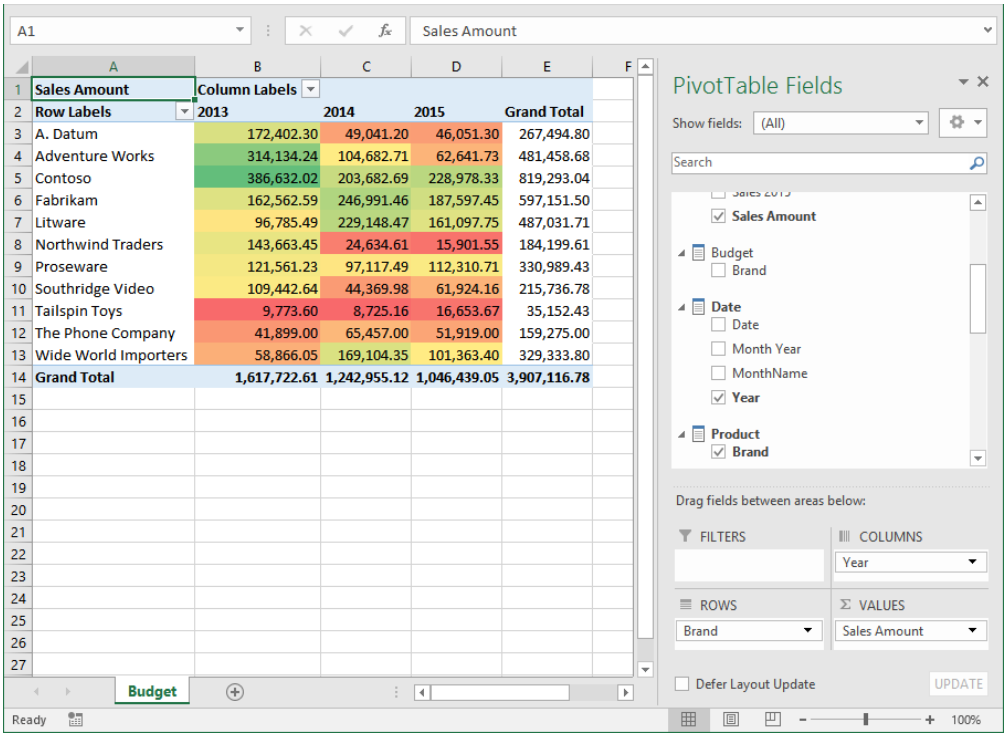


Figure 8-11: A PivotTable in Excel that is connected to the Budget model in Power BI by using Analyze In Excel.

One of the reasons to analyze data in Excel is to take advantage of specific Excel features. For example, in Figure 8-11 we applied a conditional formatting rule, so that higher values have a green background color, and smaller values have a red background color.

You might have many other reasons to use Excel to analyze a Power BI model. In general, Excel is a flexible application with which you can easily integrate data coming from different sources with data and/or calculations you have in the Excel file. Moreover, as of this writing, many of the features of a PivotTable in Excel are not yet available in the Power BI visualizations. Thanks to Analyze In Excel, you have the maximum flexibility to combine different clients (Power BI and Excel, for example) to analyze the data of the same data model.

Note The authentication used to connect Excel to the Power BI model requires a new version of the OLE DB driver that is used to establish a connection to an external Analysis Services database (OLE DB for OLAP). For this reason, you might be prompted to install such a driver the first time you use Analyze In Excel on a computer. The connection to Power BI uses claims-based authentication, which is a different technology than the Windows integrated security that you might be using to connect to Analysis Services. You might be prompted to provide a user name and password to connect to Power BI the first time you open such a connection. If you need to connect to different Power BI models using different users, you might need to modify the connections string manually, even if this behavior might change in the future, because the Analyze In Excel feature is in preview mode as of this writing. For updated documentation, go to https://powerbi.microsoft.com/documentation/powerbi-service-analyze-in-excel/.

Using Power BI Tiles from Office Store

You can create an Office document in Excel and PowerPoint in which you embed one or more Power BI visualizations. You can create a PowerPoint presentation in which you show live data from Power BI. In a similar way, you can create an Excel workbook in which you embed some visualizations from Power BI on the same page where you also have other data presented with standard Excel tools. This is possible thanks to a free third-party add-in called Power BI Tiles, which takes advantage of the Power BI APIs (these APIs will be explained in more detail later in this chapter). If you are not a developer, you still might be interested in the technical details; you just want to use the existing tool.

You can download Power BI Tiles from the Office Store. It is compatible with Microsoft Office 2013 Service Pack 1, or any following version, including Office 2016. If you have a subscription to Office 365 that includes the licensing of desktop applications, you should already have a compatible version of Office installed on your computer.

> **Note** Power BI Tiles is a free add-in created by DevScope; it is not a Microsoft product, but as of this writing, there are no corresponding solutions produced by Microsoft. You do not need administrative rights to install Power BI Tiles.

Let's visit with David once again, and consider how he can create a PowerPoint presentation that embeds some of the visualizations he created while working on the budget. In PowerPoint, on the ribbon, on the Insert tab, David clicks the Store button, as shown in Figure 8-12.

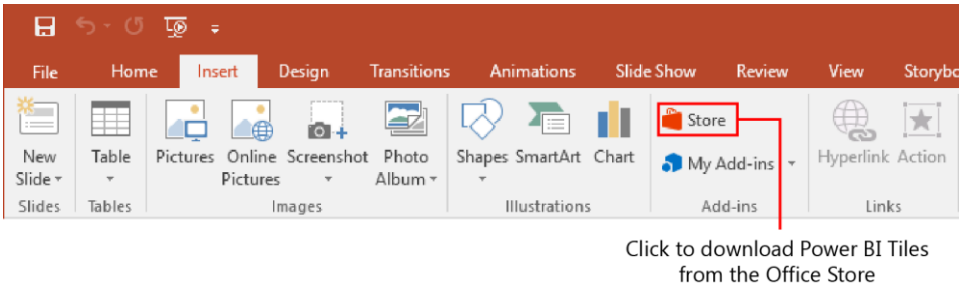


Figure 8-12: The Store button on the PowerPoint ribbon.

In the Office Add-Ins dialog box, in the search box, David types "power bi tiles," which presents him with the result depicted in Figure 8-13.

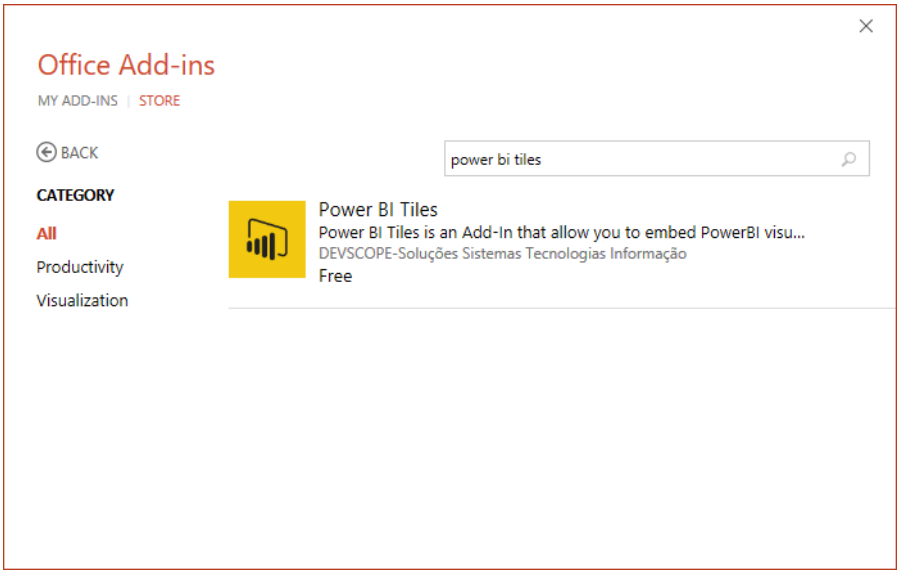


Figure 8-13: A list of available Office add-ins, filtered by the string "power bi tiles" in the search box.

> **Note** If you do not have PowerPoint installed locally and you want to use Office online, you can still use the add-ins available in the Office Store by using the online version of the Office application.

After David installs the add-in, Power BI Tiles becomes available on the PowerPoint ribbon, on the Insert tab, in the My Add-Ins list, as demonstrated in Figure 8-14.

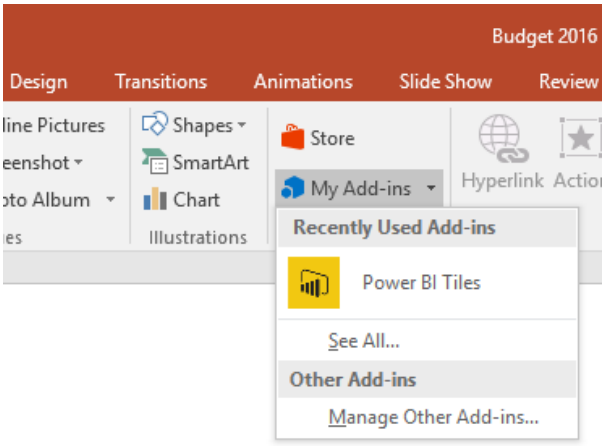


Figure 8-14: The Power BI Tiles add-in on the PowerPoint ribbon in the My Add-Ins list.

When David clicks Power BI Tiles in the list of My Add-Ins, PowerPoint inserts a new rectangular object in the current slide that will display the content of a report or a dashboard. Within this area, you are prompted to choose between connecting to your Power BI account or displaying a public report (Chapter 2 explains how to publish a report to a webpage for public access), as shown in Figure 8-15.

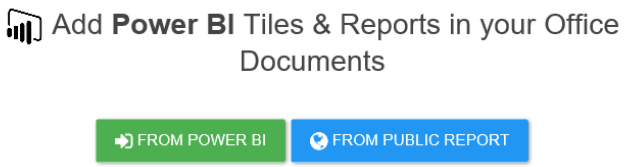


Figure 8-15: You can choose between connecting to a Power BI account or public reports for Power BI Tiles.

The first time David connects to Power BI, he is asked to grant authorization for Power BI Tiles to access certain Power BI features, as shown in Figure 8-16.

Figure 8-16: Power BI authorization request for Power BI Tiles access.

David clicks Accept, which authorizes Power BI to accept requests coming from Power BI Tiles. He now can use dashboards and reports available in any Power BI workspaces to which he has access. David can select either a dashboard or a report, and he needs to consider that there are a few differences in the visualizations and user interactions between the two. For example, Figure 8-17 depicts the list of reports available to David in the Power BI Tiles visualization in a PowerPoint slide. In this example, he chose the Budget 2016 group workspace, using the middle button of the three in the upper-right corner of the area used by the Power BI add-in. The list of reports displays the only report published in the workspace: Budget Totals. If you want to see the list of available dashboards in the same workspace, click the Dashboards button, located directly to the left of Reports, above the report list.

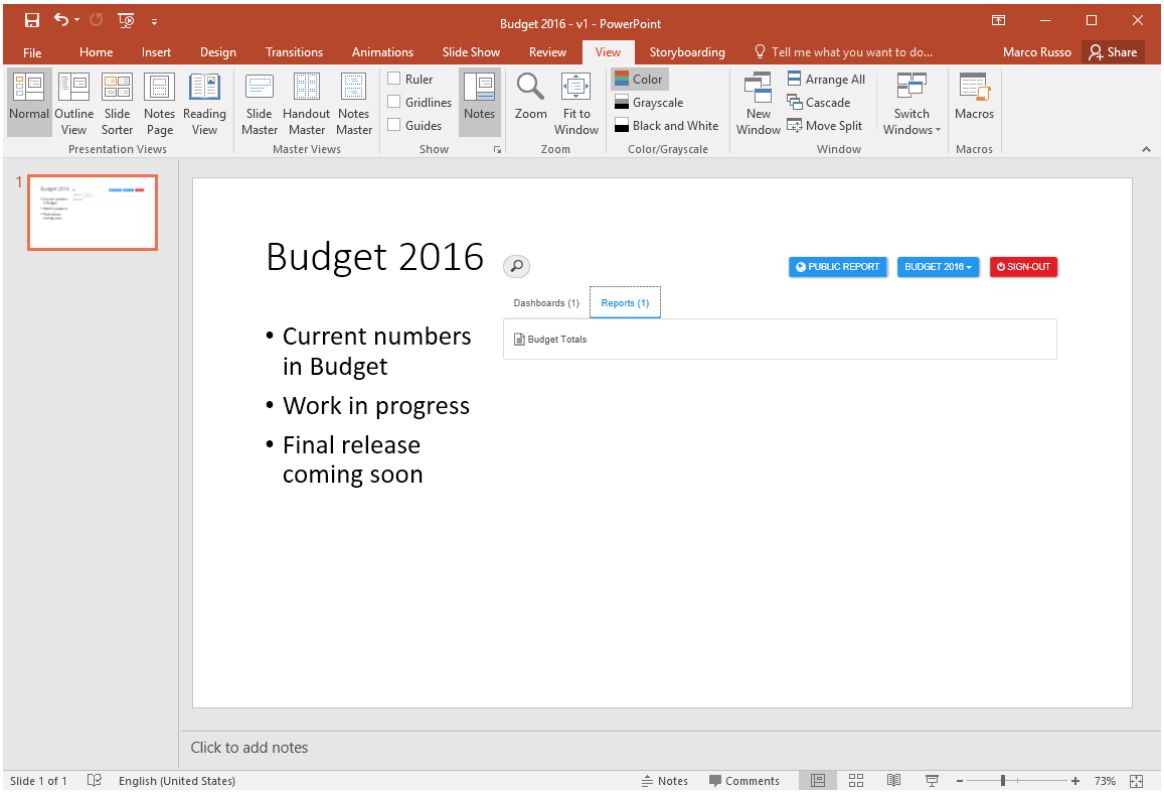


Figure 8-17: A Power BI Tiles object inserted in a PowerPoint slide lists the available reports to display.

After David selects a report, it is rendered within the workspace of the PowerPoint slide. If the report size is larger than the available space, scrollbars will appear, as shown in Figure 8-18. The report embedded by Power BI Tiles is fully functional and interactive, so David can use filters and zoom single visualizations of the report, exactly as he can do on the Power BI website. He can even utilize this interaction in Slide Show mode.

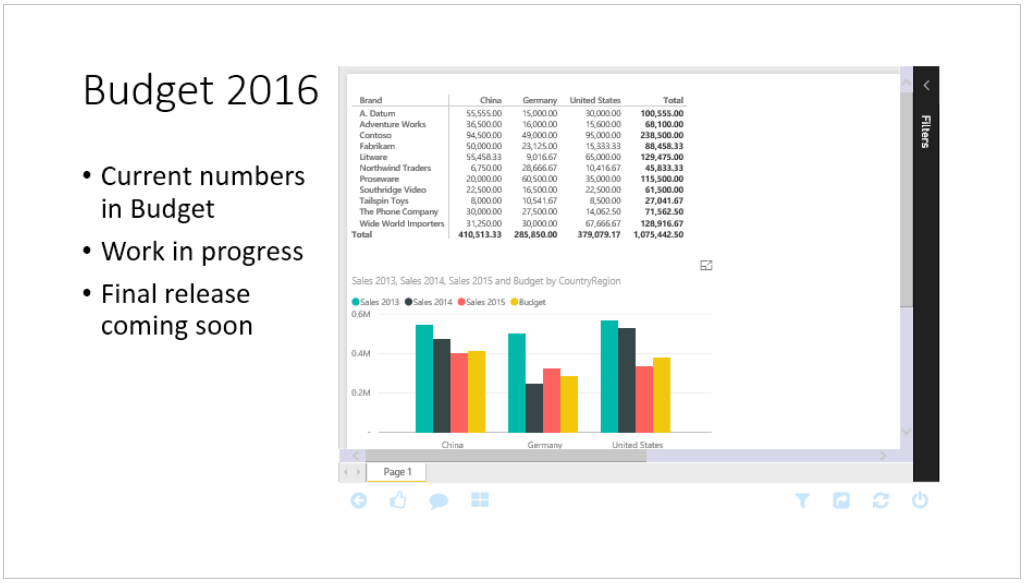


Figure 8-18: A slide in PowerPoint that embeds the content of the Budget Totals report.

You can change the objects displayed in the Power BI Tiles add-in by going back to the list of dashboards and reports (refer back to Figure 8-17). To do that, below the report, click the Back button (the left arrow), which is the one farthest left.

If you select a dashboard, you obtain a slightly different behavior: the Power BI Tiles add-in displays only one visualization from a dashboard at a time. If the dashboard contains two or more visualizations, arrows will appear on the left and right side of each one, which you can click to scroll through the visualizations, as illustrated in Figure 8-19. In this example, only one of the two available visualizations in the Budget Totals dashboard (which, by the way, are the same visualizations used in the report) is visible on the slide. Each visualization takes up the entire amount of space available, which makes them easier to view, but you do not have any interaction with the charts.

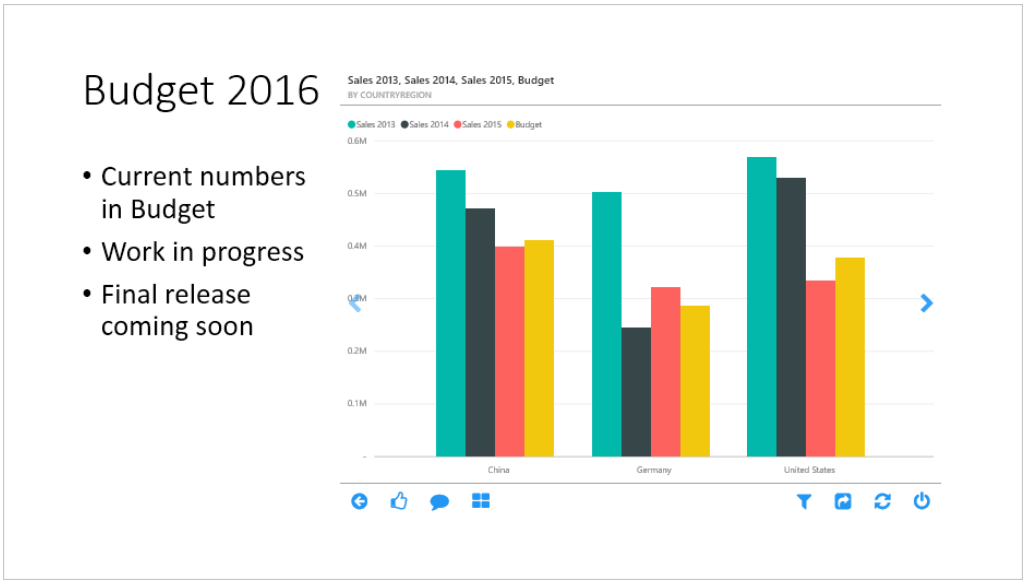


Figure 8-19: A slide in PowerPoint with embedded visualizations of the Budget Totals dashboard.

Whether you choose to embed a dashboard or a report depends on the type of presentation you are creating. If you want to interact with the data, you can modify either choice during the presentation, navigating in the list of dashboards and reports available. However, you should choose the visualization that is more readable and effective for your presentation.

If you want to refresh the data during the presentation, click the Refresh button, which is the second one from the right, below the Power BI Tiles add-in (see Figure 8-19).

You also can use the Power BI Tiles add-in in Excel, creating a worksheet that displays a visualization from Power BI next to data rendered in Excel; for example, using a PivotTable. The features of this add-in work well in Office online, too.

Managing security to access data

The previous chapters of this book demonstrated several ways to share data with other users, within and outside of your organization. In this section, you will review the features available in Power BI to share data with other users and to control access to data in a more granular way, up to the row level in each table.

The following list explains the visibility options available to a Power BI user (other options through APIs are described later in this chapter):

- **My Workspace** All of the datasets, reports, and dashboards you save in My Workspace are visible only to you, unless you explicitly share them by using one of the sharing features that follow.
- **Group Workspace** All of the datasets, reports, and dashboards saved in a group workspace are visible to all the members of the group. The group in Power BI corresponds to a group in Office 365, so you can administer the group from both administrative user interfaces. A new member in the group automatically has access to the data available to the group.
- **Share dashboard** When you share a dashboard, you send a personal invitation to a single individual identified by an email address. This email must correspond to a Power BI account. If the invited user connects to Power BI for the first time, he can create a Power BI sign-in to access data. You can control who has access to a dashboard, adding and removing users from the allowed list at any time. Optionally, you can choose to delegate another user, who can then share the dashboard to other users. When you share a dashboard with a user, you also provide him read-only access to all the underlying reports (from pinned tiles) and datasets used in the dashboard. In fact, users can freely navigate in data sources by starting with a request in the Q&A question box and then customizing filters and visuals, selecting different slices of data and changing the measures displayed and attributes analyzed.
- **Create content pack** You can publish an organizational content pack, sharing datasets, reports, and dashboards, at the granularity that you prefer. When you include a report in a content pack, you also automatically share the underlying datasets. When you include a dashboard in a content pack, you also share the underlying reports and datasets. Those who use an organizational content pack have access to all of the data included in the content pack and can freely navigate to them, creating new reports and dashboards based on that data.
- **Publish to the web** You can publish a report to the web, embedding it within a custom webpage on a website you can edit, or by simply providing a URL containing just the published report. Any user can access that report through this URL, and she can interact with the report using only the visualizations displayed in the report itself. The user cannot gain access to the underlying data source, and cannot modify measures, slicers, filters, and visualizations used in the report. You cannot control who accesses the report, because the URL can be freely shared with anyone; in fact, Microsoft can publish the report in a public gallery. You should not use this feature to distribute reports that contain sensitive data.
- **Row-level security** You can restrict data access for specific users by defining filters at the row level in one or more tables of a dataset. This is an additional security level applied to users who already have access to the data because you shared a dashboard with them. For example, you might share the same dashboard with five different managers, one for each country/region, letting them see only the data of the country/region that they manage.

When you choose the method by which you want to share data with other users, you need to evaluate the visibility you want to provide to reports and datasets, and the type of restrictions you want to apply. In the previous chapters, we describe each of the aforementioned features except the last one, which is the topic of the next section.

Using row-level security

When you want to restrict the rows visible to a single user, you must apply a security rule to the dataset, so that regardless of the report displayed or edited, the user cannot access data that he is not

allowed to see. For example, the managers of China or Europe should see only data relevant to their area, even if all of them use the same report. This type of security is known as *row-level security*.

If you are using a live connection to Analysis Services or you created a model by using DirectQuery, you must implement row-level security on the source database, and you cannot modify its behavior in Power BI.

If you have a model that imported data in the Power BI service, you can apply row-level security to the dataset. You can manage row-level security by selecting the Security action available for datasets, as you can see in Figure 8-20.

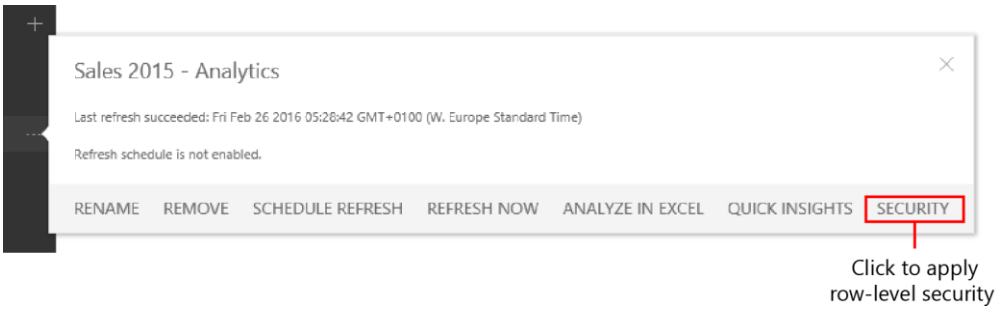


Figure 8-20: Selecting Security to activate row-level security on a dataset.

In the row-level security configuration, you create one or more security roles, which define the rows a user can see in each table. You can find a step-by-step guide to configuring row-level security at https://powerbi.microsoft.com/documentation/powerbi-admin-rls/.

> **Note** As of this writing, this feature is in preview, and details might change very quickly. You might want to refer to the online documentation to see if there have been any updates to the user interface.

Figure 8-21 shows the final result of a security role (named China) providing access to only those rows corresponding to sales made in China.

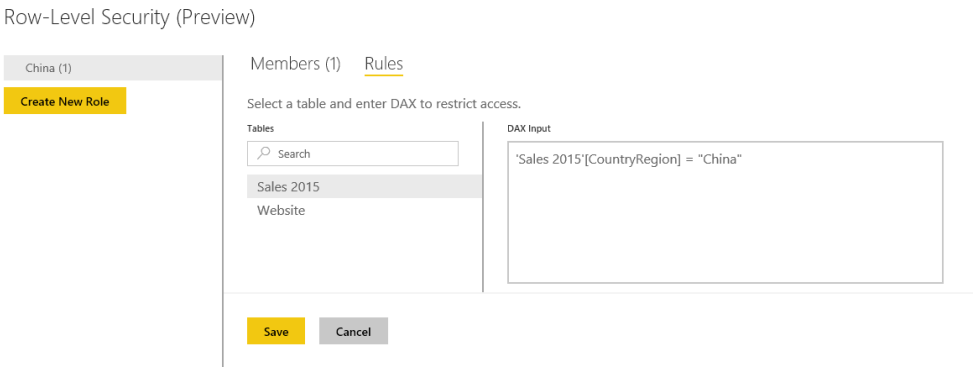


Figure 8-21: The configuration for row-level security for limiting access to sales made only in China.

Each role contains one or more members; these are the users who can access the model through the role. The rules defined for the role are in the form of a DAX expression for each table. A row in a table is visible if the condition, for that row, is true. If a user belongs to more than one role, he will have access to all the rows that are visible in at least one of his roles. However, if a user does not belong to any security role and the dataset has row-level security active, he will not see any data for that

dataset, regardless of whether he can access the dashboard containing data from that dataset because it has been shared by another user.

When you restrict access to a table that has a one-to-many relationship with other tables, you restrict access to the related tables, too. Consider a model with two tables: Customers and Sales. Applying a security rule to Customers also restricts Sales, showing only the sales related to a visible customer.

Let's take a look at what happens when David creates the row-level security rule shown in Figure 8-21, assigning Wendy Kahn as a member of the role, and then he shares with her the dashboard depicted in Figure 8-22.

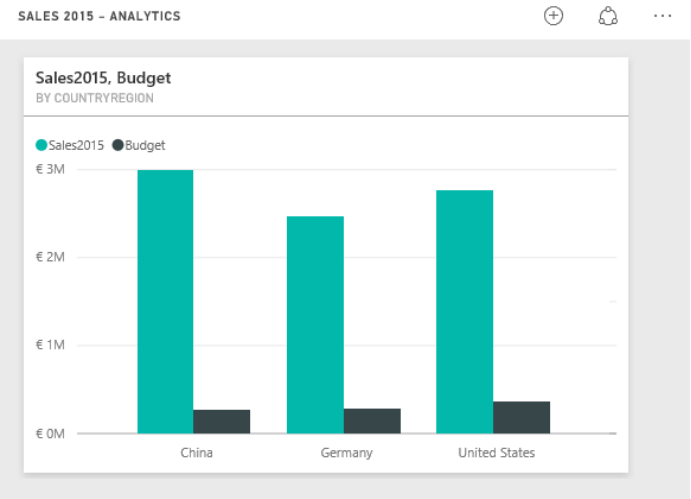


Figure 8-22: A dashboard shared by David, showing China, Germany, and the United States.

When Wendy opens the same dashboard shared by David, she sees only China; Germany and the United States are not visible, as illustrated in Figure 8-23.

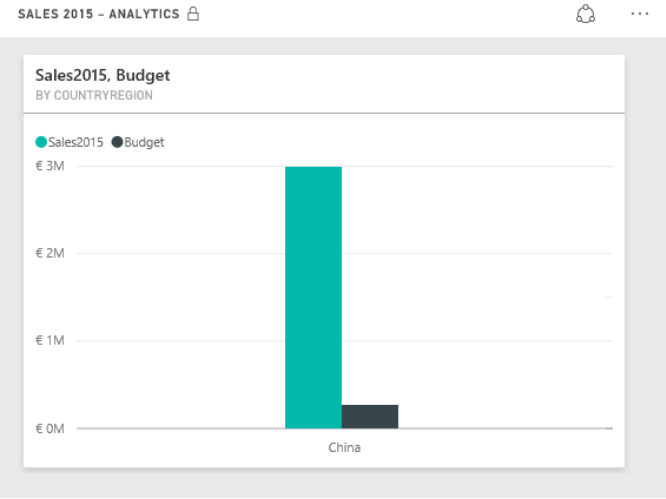


Figure 8-23: A dashboard displayed to Wendy, showing only China.

The security restrictions applied to the dashboard Wendy received from David are also applied to any other visualization shown to Wendy and are based on the same dataset. By applying security at the row level, you can easily customize the aggregations visible to each user. However, keep in mind that

you cannot prevent a user from viewing a particular table, column, or measure. The row-level security filters rows, not columns or other entities of a certain dataset. When you share a dashboard and its reports, in theory the user consuming the data cannot gain access to other entities (measures, columns) that were not published in the report, but this is not guaranteed by the row-level security filter and cannot be enforced at the dataset level. For this reason, if you need to ensure that certain measures are not visible to a group of users, you should consider creating a separate model for that, including only columns and measures that can be made visible to all of the users who can access a certain dataset.

Note As of this writing, the row-level security feature is in preview. It has a number of restrictions that might be lifted or removed in subsequent releases. As of now, you can apply row-level security only in datasets included in My Workspace, but not in group workspaces and not in datasets published in content packs. You can add only single users as role members, not user groups or distribution lists. You can apply it only to datasets created by using Power BI Desktop, not to datasets created with Power Pivot for Excel (but you can import such a model in Power BI Desktop and then publish the Power BI Desktop file). However, when you publish a new version of the Power BI Desktop file, all the existing security roles are removed completely. DirectQuery is not supported for row-level security. Q&A and Cortana are not supported by row-level security, so Q&A input is not visible if role-level security is active for all the models related to a shared dashboard.

Extending and customizing Power BI

Power BI is not only a service that you can activate and use. Likewise, it is not only a product (Power BI Desktop) that you can download and install. You can extend and customize Power BI in many ways, because Power BI offers a number of extensibility points to developers who are interested in adding features, customizing the experience, or integrating existing applications with Power BI.

You can find a more detailed introduction oriented to developers at https://powerbi.microsoft.com/documentation/powerbi-developer-overview-of-power-bi-rest-api/. The goal of this section is to introduce you to what is possible and what the current limitations are. In this way, you will have a better understanding of the platform before looking for developers who might help you in this effort, if you do not have the required skills but are interested in achieving the results.

Creating custom visualizations for Power BI

Chapter 7 describes how to add custom visualizations in a Power BI report. As a user, you are likely interested in using existing custom visualizations, and you can find a public gallery of them at https://visuals.powerbi.com. If you need specific visualizations that are not available in the gallery, you can (or ask a programmer to) create a new visualization, following the guide available at https://powerbi.microsoft.com/documentation/powerbi-custom-visuals/. The main skills required to create a custom visualization are TypeScript (a typed version of JavaScript) and CSS. Thus, if you have programming skills in JavaScript and CSS, you will have a short learning curve to become proficient in writing code for a custom visualization. To become inspired to create new visualizations in Power BI, take a look at the examples of custom visualizations in Chapter 7.

Introducing the Power BI REST API

Previously in this chapter, we showed you how to use the Power BI Tiles add-in. Recall that this component interacts with the data and services provided by Power BI using a programming interface called REST API. In this book, we do not want to go into the details of a REST API or how to use it, but

you can find these details in the online documentation. Most of the information there is useful to developers who want to integrate Power BI services in their applications. The goal here is to explain the importance of this API and why it is the foundation of an ecosystem that makes the integration between Power BI and other applications and services possible, going beyond the features currently available to Power BI end users.

Using the REST API for Power BI, a developer can create a new application or extend an existing one so that it can publish or consume data, reports, and dashboards in Power BI. REST stands for *Representational State Transfer*, which is a protocol that allows any existing programming language to interact with the API, and it is widely adopted in modern programming platforms. REST facilitates interoperability across different languages and operating systems. There are particular specifications to handle authentication and authorization, and you can find all of the details and many examples at https://msdn.microsoft.com/library/dn877544.aspx.

Besides the REST protocol, the API exposes several features that can manipulate the following entities:

- **Dataset** You can create new datasets and read existing ones.
- **Table** You can create new tables and modify the schema of existing ones that you created before using the same API. You cannot modify tables that are a part of a data model that imports data from external data sources.
- **Row** You can add and delete rows in tables that you created in a dataset. The delete operation removes all the rows, and you cannot specify any filter, whereas the add operation works incrementally, adding new rows to the existing ones.
- **Group** You can access a particular group to create a dataset in a group instead of a personal workspace.
- **Import** You can import a Power BI Desktop model (.pbix file) or a Power Pivot for Excel data model (.xlsx file) in Power BI.
- **Dashboard** You can retrieve dashboards and tiles from dashboards from a particular workspace to which you have access. The Power BI Tiles add-in, for example, uses this API to retrieve the selected visualization from a dashboard.
- **Report** You can retrieve reports from a particular workspace to which you have access.

You can integrate the visualizations and report objects that you can access through the API in an existing application (this is called "embedding an IFrame"), which is actually what the Power BI add-in does. In the following sections, you will see two examples of applications that are possible thanks to the Power BI REST API.

Limitations

It is worth mentioning the current limitations, considering that the API will evolve and new features will be added, hopefully also to cover some of the scenarios that are not available today. In general, you cannot manipulate the content of a single object. For example, you cannot alter a published dataset, dashboard, or report. You cannot create a report or a dashboard programmatically. You can publish a Power BI Desktop file (.pbix), which can include particular reports, but you do not have an API to create a report or a .pbix file programmatically. This is a current limitation for the Power BI embedded scenario that is described in the next section; but keep in mind that it is still in preview as of this writing, and new API features certainly will be added in the future. Thus, you'll probably want to

look at the updated documentation to check whether new features have been added to overcome the limitations described here.

By using the Power BI REST API, you can extend existing applications, integrating features available in the Power BI service. This API also opens myriad possibilities to third-party vendors to create components, applications, and services that extend the features available in Power BI.

Pushing real-time data to Power BI dashboards

One of the features available by using the Power BI API is the ability to "push" data in a dashboard in real time. By using this feature, the numbers and visualizations included in a dashboard are automatically updated almost every second, reflecting the changes received in the data. However, the datasets underlying these dashboards are designed in a particular way, and in this section, we want to give you an overall view of the features and limitations of this technique. If you're a developer who is interested in creating these dashboards, a complete walkthrough is available at https://powerbi.microsoft.com/documentation/powerbi-developer-walkthrough-push-data/.

To obtain a real-time dashboard, you first need to programmatically create a dataset. Then, you need to build reports using this dataset, and finally you can pin report visualizations and Q&A visualizations in the dashboard. These tiles will be automatically updated as soon as the underlying dataset receives new data.

The first step also defines the biggest limitation that exists for real-time dashboards: You must first create a dataset programmatically, and this dataset can have tables and columns that will be filled with data sent by an application. The data refresh is not possible in these datasets. We call this a *push mode*, where an application sends data to Power BI, instead of having Power BI ask for data from the data source (which is the classic *pull mode* used by data refresh). You cannot create the data model by using Power BI Desktop, and you cannot add relationships and measures to the data model. You can obtain only standard aggregations for measures, such as sum, average, count distinct, and other predefined ones, but you cannot create either calculated columns or measures using custom DAX expressions. For this reason, it is difficult to display percentages and variations obtained by aggregating existing data. The dataset can be created in a personal workspace or in a group workspace.

The second step, which still requires that you or an application developer write custom code, is to insert rows in the tables of the dataset using the Power BI REST API. Every table has a limit of 5,000,000 rows, or 200,000 rows if you choose a storage model (also known as FIFO dataset) that automatically removes the older rows. The amount of rows written and the frequency of update depends entirely on the application that "pushes" data into the dataset in Power BI. A few limitations exist, based on the Power BI plan used (10,000 rows per hour for the free service plan, and 1,000,000 rows per hour for the paid service plan, Power BI Pro). You can write this code specifically for a single dataset, or you can take advantage of the Azure Stream Analytics service, which simplifies using the same stream of data in different datasets, as described at https://azure.microsoft.com/documentation/articles/stream-analytics-power-bi-dashboard/.

After a dataset is created and populated with data, you can create a report using Power BI online. You cannot use Power BI Desktop for such an operation. You can use all the visualizations and filters available in a report to create your visualization. A report is not updated in real time; you must always manually refresh it to display the updated data. You can pin every visualization used in the report to a dashboard visualization, and when you establish the connection between the dashboard and the

dataset (by pinning the first visualization), you can also begin using Q&A to navigate the data and to obtain other visuals that you can pin to the dashboard. All the tiles in the dashboard that are connected to a dataset that receive data in push mode, automatically refresh their content as soon as new data is received. You might observe a refresh almost every second, and the latency between updates of data and visualizations is typically only a few seconds.

Figure 8-24 presents an example of the real-time dashboard we created to monitor the usage of the DAX Formatter service, which is available at www.daxformatter.com. Every time a user formats a DAX query using this service, the application updates the dataset on Power BI, providing the date and time of the request and a flag that specifies whether the request was formatted correctly (increasing the Formatted counter) or contained a syntax error (increasing the Errors counter).

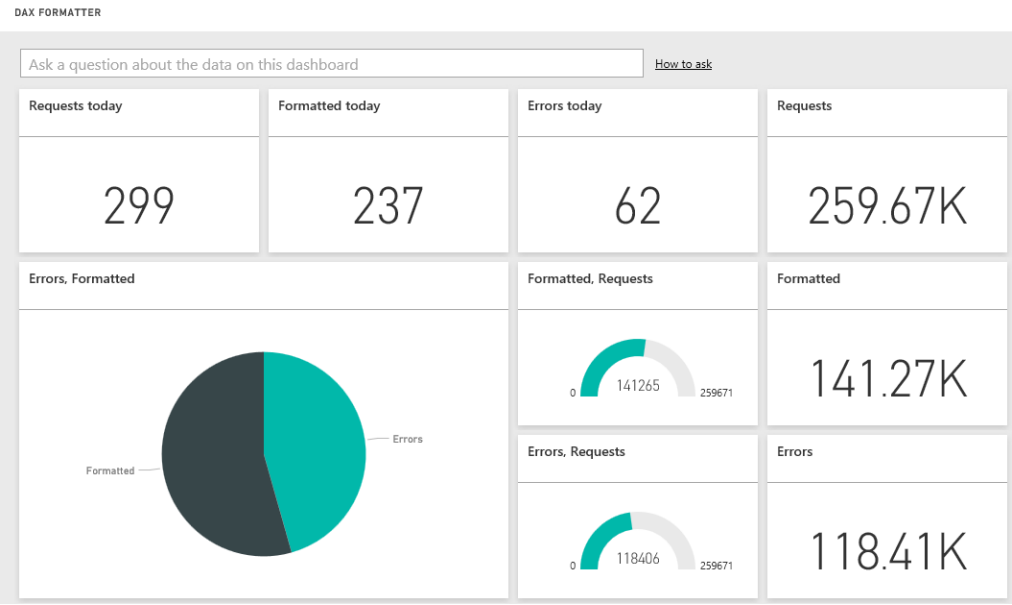


Figure 8-24: A dashboard updated in real time with data from www.daxformatter.com.

It is interesting to note that using Q&A is important to obtain certain dynamic filters, which are impossible to build in a report because you cannot create custom measures in DAX in this type of dataset. For example, the number of requests made today can be obtained by asking the question through Q&A, as illustrated in Figure 8-25. The word "today" is converted to the current date and is applied as a filter to the date column, returning the number of requests made in the current day. Note also that the number of requests in Figure 8-25 increased from Figure 8-24 because data is continuously updated in the dataset!

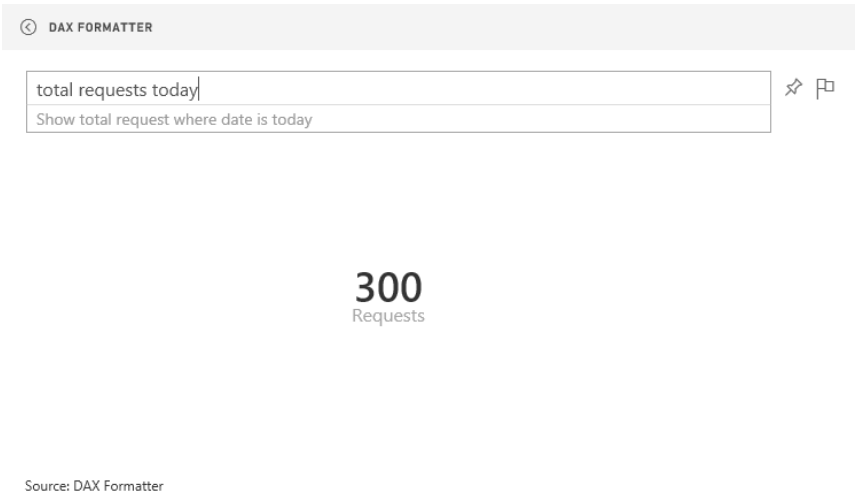


Figure 8-25: Request made through Q&A for the number of requests made today.

Power BI embedded in applications

Another extension available through the Power BI REST API and specific libraries to manage authentication for custom application is Power BI Embedded, which is an Azure service with which you can set up integration between an application and Power BI services.

For example, consider a service that collects data about personal bicycle trips. This service has a web application to manage the configuration and manual upload of data, even if most of the information will be sent by specific devices and/or apps. When it comes time to analyze data, it would be nice for this application to use Power BI services. One way to obtain this integration is to export data to Power BI and create custom data models and reports. For the company that provides this service, it could be a good idea to create a service content pack to make it easy for existing Power BI users to import their data. However, this approach requires all users to create their own Power BI account (even if it is a free one), whereas the application instead should have some embedded solution to display standard reports containing personal data. By using Power BI Embedded, the developers who build the web application can design these reports and publish them within their application in a seamless way. In this way, the web application can show a report containing data of the user who signed in within the same webpage, and without requiring additional authentication to the user itself. The process is completely transparent to the user.

More info You can find documentation for Power BI Embedded and its pricing details at https://azure.microsoft.com/services/power-bi-embedded/.

Note It is worth to mention that we showed you another type of web publishing in Chapter 2, the Publish To Web feature. The main difference between Power BI Embedded and Publish To Web is that the former controls the authentication of the user and can display customized content within the same report, whereas the latter only shows the same content to anonymous users.

Conclusions

In this chapter, you learned how you can use Power BI in your company, what the architectural implications of the many options available for data refresh are, how to manage security, and what the options are to customize Power BI and/or integrate it with existing applications.

Here are the most important features you learned:

- Available options to update data with scheduled refresh or live connections
- Integration of Power BI with Microsoft Office
- Control data access for specific users with row-level security
- Possible extensibility options using the Power BI REST API

Power BI is an open ecosystem that is constantly growing, thanks to the features added by Microsoft and those additional options provided by third-party groups, which use the same API you can use to customize and extend Power BI according to your specific needs.

About the authors

Marco Russo and **Alberto Ferrari** are the founders of sqlbi.com, where they regularly publish articles about Microsoft Power BI, Power Pivot, DAX, and SQL Server Analysis Services. They have worked with DAX since the first beta version of Power Pivot in 2009, and, during these years, sqlbi.com became one of the major sources for DAX articles and tutorials.

They both provide consultancy and mentoring on business intelligence (BI), with a particular specialization in the Microsoft technologies related to BI. They have written several books and papers about Power Pivot, DAX, and Analysis Services. They also wrote popular white papers such as "The Many-to-Many Revolution" (about modeling patterns using many-to-many relationships) and "Using Tabular Models in a Large-Scale Commercial Solution" (a case study of Analysis Services adoption published by Microsoft). Marco and Alberto are also regular speakers at major international conferences, including Microsoft Ignite, PASS Summit, and SQLBits.

You can contact Marco at marco.russo@sqlbi.com, and Alberto at alberto.ferrari@sqlbi.com.

From technical overviews to drilldowns on special topics, get *free* ebooks from Microsoft Press at:

www.microsoftvirtualacademy.com/ebooks

Download your free ebooks in PDF, EPUB, and/or Mobi for Kindle formats.

Look for other great resources at Microsoft Virtual Academy, where you can learn new skills and help advance your career with free Microsoft training delivered by experts.

Microsoft Press