Microsoft

Analyzing and Visualizing Data with Microsoft Excel

Exam Ref 70-779

Chris Sorensen

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Microsoft

Exam Ref 70-779 Analyzing and Visualizing Data with Microsoft Excel

Chris Sorensen

Exam Ref 70-779 Analyzing and Visualizing Data with Microsoft Excel

Published with the authorization of Microsoft Corporation by:
Pearson Education, Inc.

ISBN-13: 978-1-5093-0804-0
ISBN-10: 1-5093-0804-0

Library of Congress Control Number: 2018943933
1 18

Editor-in-Chief	Brett Bartow
Senior Editor	Trina MacDonald
Development Editor	Rick Kughen
Managing Editor	Sandra Schroeder
Senior Project Editor	Tracey Croom
Editorial Production	Backstop Media
Copy Editor	Liv Bainbridge
Indexer	Julie Grady
Proofreader	Katje Richstatter
Technical Editor	Daniil Maslyuk
Cover Designer	Twist Creative, Seattle

I dedicate this book to my wife, Joely, my daughter Camryn, and my son Murphy. Their love, support and encouragement had no bounds through what was one of the most challenging projects I have ever worked on.

—Chris Sorensen

Contents at a glance

Contents

Acknowledgements

I would like to thank the first two members of the Iteration Insights team, Jane Wood and Emily Gu, for their enthusiasm and contributions as we worked on this project as a newly formed company. They helped make this book possible as we worked days, nights, and weekends for months to bring it all together. I would also like to thank Trina MacDonald for her encouragement, support, and belief in our team through the process of writing this book. Writing a book of this magnitude would not be possible without the contribution of the editing staff of Daniil Maslyuk, Troy Mott, Rick Kughen, and Christina Rudloff, who sifted through the many versions of our materials to ensure technical and grammatical accuracy. Their insights and contributions were appreciated.

About the author

CHRIS SORENSEN, MCSE (Data Management and Analytics) and MCT, is the Founder and President of Iteration Insights Ltd. He is a consultant, architect, educator, and coach who has been working in the Analytics space for nearly 20 years. Over his career, he has provided strategic and architectural advisory services to many clients and most recently he has been involved with leading numerous Power BI and Excel PowerPivot projects. He has evangelized both Excel and Power BI with Microsoft since July 2015. Follow him on both Linkedin and Twitter as @wjdataguy.

Introduction

The 70-779 exam is designed for both Business Intelligence developers and Business power users that have used Excel for many years to help support Analytics in an organization. The book is an even split between the skills needed to Consume and Transform Data, Model Data, and then Visualize Data.

In the Consume and Transform Data chapter the focus is on the sources that Excel can connect with for data, and then how to transform data using the Power Query Editor. It is important to understand the M language that underlies the Power Query Editor and is generated when performing transform tasks using the GUI.

The chapter on Modeling Data turns the focus to the Excel Data Model to build the necessary relationships that glue the data model together, and the how to optimize it for reporting. Next, we look at how to extend the model and make it easy to consume for users by utilizing DAX, KPIs, and Hierarchies.

When Visualizing data, we spend time considering PivotTables and PivotCharts as the two primary methods in Excel for presenting data. Lastly, you will spend time investigating how to interact with Power BI as an additional means for distributing content to users.

This book covers every major topic area found on the exam, but it does not cover every exam question. Only the Microsoft exam team has access to the exam questions, and Microsoft regularly adds new questions to the exam, making it impossible to cover specific questions. You should consider this book a supplement to your relevant real-world experience and other study materials. If you encounter a topic in this book that you do not feel completely comfortable with, use the "Need more review?" links you'll find in the text to find more information and take the time to research and study the topic. Great information is available on MSDN, TechNet, and in blogs and forums.

Organization of this book

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list is available for each exam on the Microsoft Learning website: *http://aka.ms/examlist*. Each chapter in this book corresponds to a major topic area in the list, and the technical tasks in each topic area determine a chapter's organization. If an exam covers six major topic areas, for example, the book will contain six chapters.

Microsoft certifications

Microsoft certifications distinguish you by proving your command of a broad set of skills and experience with current Microsoft products and technologies. The exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop, or implement and support, solutions with Microsoft products and technologies both on-premises and in the cloud. Certification brings a variety of benefits to the individual and to employers and organizations.

MORE INFO **ALL MICROSOFT CERTIFICATIONS**

For information about Microsoft certifications, including a full list of available certifications, go to *http://www.microsoft.com/learning*.

Check back often to see what is new!

Microsoft Virtual Academy

Build your knowledge of Microsoft technologies with free expert-led online training from Microsoft Virtual Academy (MVA). MVA offers a comprehensive library of videos, live events, and more to help you learn the latest technologies and prepare for certification exams. You'll find what you need here:

http://www.microsoftvirtualacademy.com

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:

https://aka.ms/examref779/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*.

Stay in touch

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

Important: How to use this book to study for the exam

Certification exams validate your on-the-job experience and product knowledge. To gauge your readiness to take an exam, use this Exam Ref to help you check your understanding of the skills tested by the exam. Determine the topics you know well and the areas in which you need more experience. To help you refresh your skills in specific areas, we have also provided "Need more review?" pointers, which direct you to more in-depth information outside the book.

The Exam Ref is not a substitute for hands-on experience. This book is not designed to teach you new skills.

We recommend that you round out your exam preparation by using a combination of available study materials and courses. Learn more about available classroom training at *http://www.microsoft.com/learning*. Microsoft Official Practice Tests are available for many exams at *http://aka.ms/practicetests*. You can also find free online courses and live events from Microsoft Virtual Academy at *http://www.microsoftvirtualacademy.com*.

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list for each exam is available on the Microsoft Learning website: *http://aka.ms/examlist*.

Note that this Exam Ref is based on this publicly available information and the author's experience. To safeguard the integrity of the exam, authors do not have access to the exam questions.

CHAPTER 1

Consume and transform data by using Microsoft Excel

As a business intelligence professional, chances are you have spent a lot of time wrangling with data that comes in many shapes and sizes and from a multitude of different sources. This is a challenge that most analysts face daily. Data is scattered, and sometimes not managed properly, which complicates the process of making data easily accessible for analytics. The task of consuming and transforming data in Excel has long been one of the hardest, and yet most important, skills for a data analyst to master as they prepare data sets for consumption and presentation. After all, it is all about the data. If your data is of a poor degree of quality or it is not structured properly, it can lead to challenges when using your data for meaningful and reliable analysis.

IMPORTANT

Have you read page xix?

It contains valuable information regarding the skills you need to pass the exam.

For years Excel professionals have relied upon their mastery of advanced Excel functions such as: VLOOKUP, IF, FIND, CLEAN, and SUBSTITUTE, on top of traditional data import means.

In this chapter, start by looking at the different sources that Excel can connect into using the newly organized Get & Transform Data functionality. Once you have connected to a source, we turn your attention to transforming the data and performing any cleansing that is deemed appropriate for making your data sets easier to navigate and ultimately more reliable.

The focus of this chapter is to connect, transform, and cleanse data into individual tables using what is now known as Get & Transform. In Chapter 2 "Model data," we dive deeper into the Data Model and discuss data modeling best practices, optimization techniques, and how to set up your model for a rich end-user self-service experience. In Chapter 3 "Visualize data," you focus on using the data in the data model for reporting within Excel.

Skills in this chapter:

- Skill 1.1: Import from data sources
- Skill 1.2: Perform data transformations
- Skill 1.3: Cleanse data

Skill 1.1: Import from data sources

One of the great things about Power Query, which is part of the Get & Transform functionality, is that the process of connecting to source systems and cleansing data has been made substantially easier to perform and maintain over traditional Excel techniques.

The first stop when getting data is to become familiar with the Excel 2016 ribbon. In this section, you focus on connecting to data sources via the **Get & Transform Data** group in the **Data** tab. When you connect to any source, you have many options. First, you can just create a query, which is a stored definition. You can load data into an Excel table, or you can also choose to load to the Data Model. Which method you choose depends on how you will be using your data. For purposes of the exam, we focus on loading data to the Excel Data Model.

This section covers how to:

- Connect to and import from databases, files, and folders
- Connect to Microsoft SQL Azure and Big Data
- Import from other Excel workbooks
- Link to data from other sources

Connect to and import from databases, files, and folders

Databases and files are some of the most common data sources used when connecting to data for analytics purposes. In this Skill, we also connect to a folder structure, which is a very convenient way to acquire data from multiple files that share the same format.

Excel can connect to the following database sources:

- SQL Server database
- Access database
- SQL Server Analysis Services database
- Oracle database
- IBM DB2 database
- MySQL database
- PostgreSQL database
- Sybase database
- Teradata database
- SAP HANA database

Analytics in Microsoft Excel—Then

Before diving into the new methods that are available in Excel for making the overall process of data analysis easier (which is the focus of this book), we need to look at how Excel used to work.

Traditionally, you had to combine data you were acquiring from external systems into single objects, such as an Excel data table, so you could use PivotTables to slice and dice or to build other objects such as PivotCharts. This method typically posed a few problems.

- First, you would often end up with mixed grains of data in a table (for example, an order total repeating with each order line item) that would need to be known by the consumer at analysis time to prevent producing incorrect results.
- Second is that you typically needed to do some sort of workaround to deal with the one million row limit in Excel. This often involved aggregating data (e.g. from day-level to month-level), which causes loss of granularity, and this meant your analysis could only go to the month level, and your users would be asking for day-level analysis. The very nature of analytics is that you ideally want the lowest level of detail in your models to support "the next question" that users inevitably ask. Million-plus row datasets are now a reality.
- Third is that combining, transforming, and cleaning data is a challenge. Pure Excel functions, such as VLOOKUP(), typically do not perform well with larger datasets. Mixed into this problem is that to do more complex transformations, users often found themselves injecting VBA or SQL into their solutions that drove up complexity.
- Lastly, the sources that were available were limited. Today, businesses want to mashup data from many different internal and external systems. This is no easy task with traditional Excel methods.

Analytics in Microsoft Excel—Now

In Excel 2010, a new feature—called Power Pivot—was introduced along with Power Maps, Power Query, and Power View. Microsoft was making a concerted effort to improve analytical capabilities within Excel. When Power Pivot was introduced, Excel expert Bill Jellen declared it as the greatest thing to happen to Excel in 20 years. That is high praise!

Suddenly, the 1,048,576-row limit in Excel was gone and you can now import considerably larger data sets into PowerPivot, which is also known as the Excel Data Model. More on this in Chapter 2 in the section named "Understanding the Excel data model.." And data did not have to be mashed into a single table as it had before so PivotTables could query the data. Also, Microsoft introduced Power Query, which made importing, transforming, and cleansing data much easier and more performant.

A Data Model is a new approach for integrating data from multiple tables, relating those tables, and building a queryable model within an Excel workbook. Within Excel, Data Models are used transparently, providing tabular data that can be consumed by PivotTables and PivotCharts. The engine that drives this is known as xVelocity which is the same in-memory engine that drives SQL Server Analytics Services Tabular models.

EXAM TIP

For those of you who do both Excel and Power BI development, take caution and remind yourself when taking this exam that you are doing the Excel-based exam. It is very easy to answer a question from the point of view of a Power BI user because there are so many similarities. Do not let the differences trick you.

Connecting to sources

Now, explore the different sources you can acquire data from using Excel Get & Transform. To connect to a data source, Click the **Data** tab and then on the **Get & Transform Data** group, click **Get Data**.

The menu options shown in Figure 1-1 categorize the data sources into the following groupings. **From File** > **From Database** > **From Azure** > **From Online Services** and **From Other Sources**.

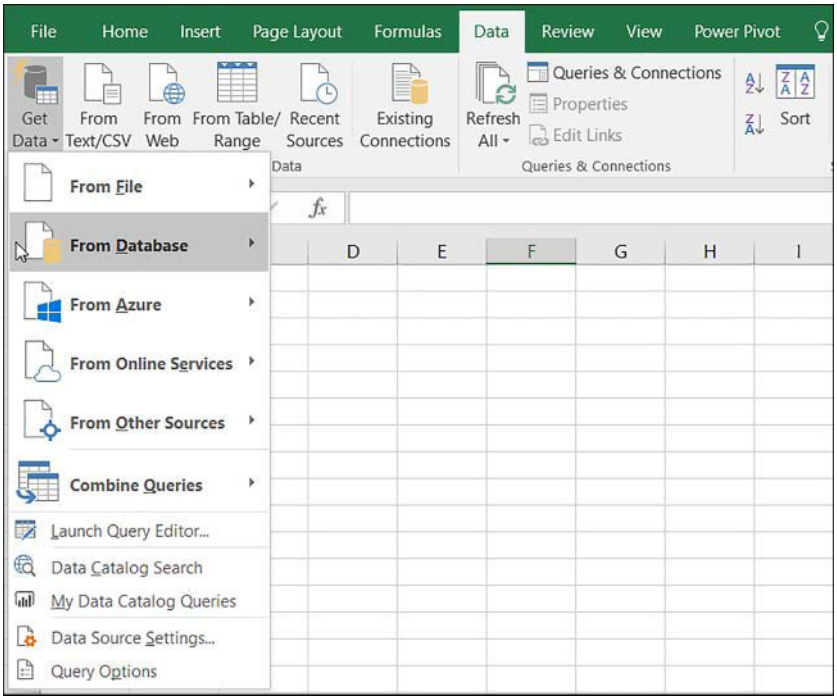

FIGURE 1-1 The Get Data Function

There are three ways to analyze data in Excel. You can import data into Excel using the Data Model, or you can use traditional Excel means, such as an Excel table. If you are using Analysis Services, you also have the option to connect directly into the database and query information live. This allows you to interact with the data without importing it into the spreadsheet, which keeps workbook size down and might also improve performance.

The examples in this guide indicate which data sources to use when following the hands-on examples. In Skill 1.1, you use many different data sources. Following along is optional, but doing so helps reinforce the concepts. In Skills 1.2 and beyond, you move to your local file system and use files that are available on the book's companion site, which simplifies the overall process.

EXAM TIP

The data connectors that are available in Get & Transform Data are important. You do not need to know the details of each system they connect to, but you should know enough about the specifics of the connector from the Get & Transform perspective regarding what can be configured and where common problems lie.

Connecting to SQL Server

One of the most popular enterprise relational database management systems in use today is Microsoft SQL Server. In this example, we connect to an AdventureWorks2016 Database to obtain the following two tables:

- [AdventureWorks2016].[Sales].[SalesOrderHeader]
- [AdventureWorks2016].[Sales].[SalesOrderDetail]

NOTE **DOWNLOADING THE ADVENTUREWORKS DATABASES**

Throughout the book, you will be interacting with AdventureWorks data through a SQL Database and both Analysis Services Multidimensional and Tabular models. The files and installation instructions are available on the GitHub sites listed below:

The AdventureWorks SQL Database files named AdventureWorksDW2016.bak and AdventureWorks2016.bak can be downloaded from *https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks*.

The AdventureWorks Multidimensional and Tabular Models named adventure-works-multidimensional-model-full-database-backup.zip and adventure-works-tabular-model-1200-full-database-backup.zip can be found at *https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks-analysis-services*.

To Connect to a Microsoft SQL Server database, follow these steps:

1. Select **Data** > **Get Data** > **From Database** > **From SQL Server Database**. You are then presented with the window in Figure 1-2. Your **SQL Server Database** dialog may be contracted and if so, click **Advanced Options** to mirror Figure 1-2. Now type the name of your **Server** and leave all other defaults. Do not click **OK** until you have read the following discussion of the options in this dialog.

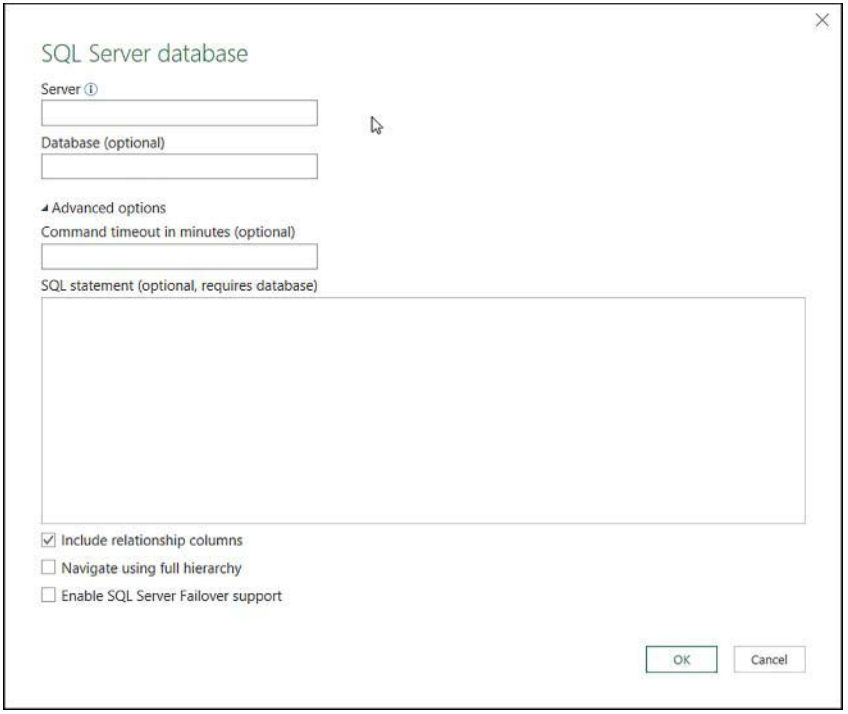

FIGURE 1-2 The SQL Server Database connection configuration dialog

There are several configurations that can be made in the dialog:

- **Server** Here you specify a mandatory server name that you wish to connect to with an option to include the port number.
- **Database (Optional)** If you know the name of the database you want to connect to, type the name here. If you leave the field blank, when you move to the **Navigator** dialog as shown in Figure 1-6; you are allowed to choose from a list of databases for which your credentials have access.

If you expanded the Advanced options, you can configure a few more properties:

- **Command Timeout In Minutes(Optional)** This allows the SQL query to time out should the query not return in the allotted timeframe.
- **SQL Statement (Optional, Requires Database)** In this text box, you can write a Native Database Query. This can be very helpful in situations in which you already have a complex query written that you know cannot be written in the Query Editor, or when you simply do not want to repeat the work. If you choose this option, you are prompted to add a value in the **Database** field. This query may cross databases, despite the Database field indicating a single database.

- **Include Relationship Columns** If chosen, you can subsequently choose the **Select Related Tables** option in Figure 1-7. If not, the metadata that drives the relationships is not included and no related tables are found, even if you know referential integrity exists at the database layer. The default for this option is selected.
- **Navigate Using Full Hierarchy** If chosen, you can navigate the Hierarchy of SQL Server objects from the server down to databases, then schemas, and finally objects within schemas. If disabled, you navigate from server to databases, and then all objects from all schemas. The default for this selection is cleared.
- **Enable SQL Server Failover Support** If chosen, your query can take advantage of local high availability through the server-instance level. The default selection is cleared.

NEED MORE REVIEW? **IMPORT DATA FROM DATABASE USING NATIVE DATABASE QUERY**

For more information, see the following article, "Import Data from Database using a Native Database Query."

https://support.office.com/en-us/article/Import-Data-from-Database-using-Native-Database-Query-Power-Query-f4f448ac-70d5-445b-a6ba-302db47a1b00.

2. When you are done configuring the options in Figure 1-2, Click **OK**. If this is your first time connecting to this source, the dialog box in Figure 1-3 appears. Here you enter the credentials to connect to the database. If you are using your Windows credentials, make the appropriate selection. If you are going to connect using SQL Server Credentials, click the **Database** tab and Figure 1-4 appears, where you can enter those credentials. Click **Connect** after you have specified the credentials.

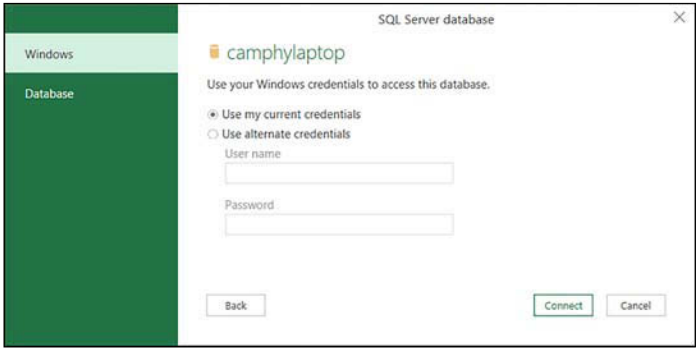

FIGURE 1-3 SQL Server Database Windows-based credentials

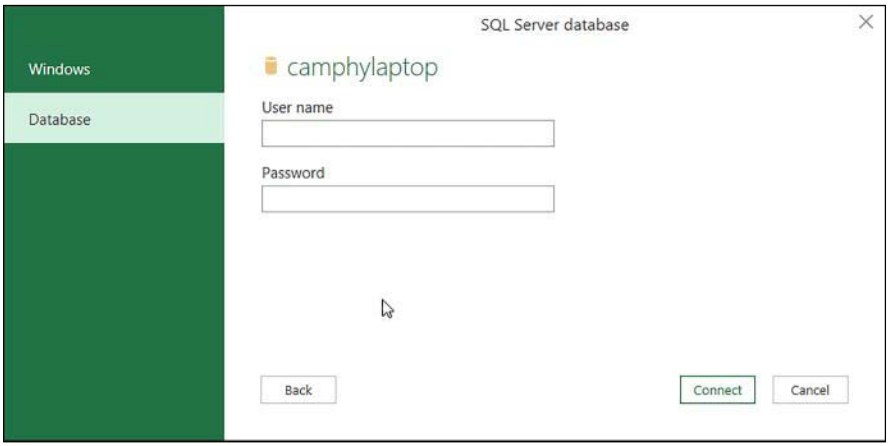

FIGURE 1-4 SQL Server Database SQL User credentials

3. If this is the first time connecting to the specified source, you might be presented with a subsequent dialog box that prompts you to select the authentication mode for the connection. The database in this example does not support encryption, so the dialog shown in Figure 1-5 appears. If you have the option presented to you (in this example we do not) and you do not want to connect using an encrypted connection, clear this check box, and then click **Connect** or **OK**.

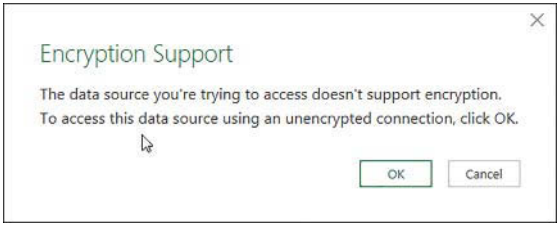

FIGURE 1-5 Encryption Support dialog

NEED MORE REVIEW? **ENCRYPTED CONNECTIONS**

If your connection supports encryption and you clear the Encrypt connection check box, or your database does not support encryption, as in Figure 1-5, data is transferred from SQL Server to Excel in plain text. A malicious user might be able to intercept and examine unencrypted data.

4. You are brought to the **Navigator** dialog as shown in Figure 1-6. Because you did not type a database name in Figure 1-2, you see the server name followed by a list of databases. Expand out the **AdventureWorks2016** database and find the two tables that you named at the beginning of the Connect to a Microsoft SQL Server database exercise.

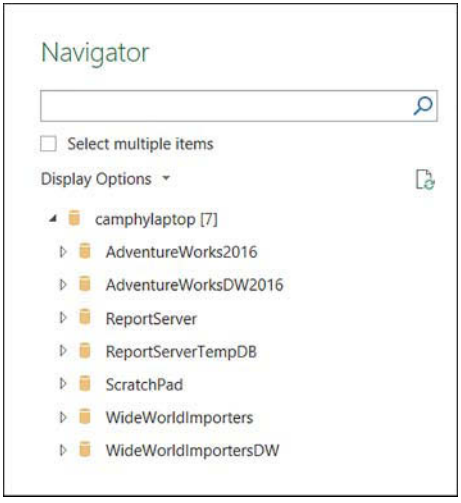

FIGURE 1-6 SQL Server Object Navigator at the server and database level

5. Navigate down the tree to the specific object that you need, as in Figure 1-7. Notice that SQL tables, views, table valued functions, and scalar functions are visible in the object hierarchy. Also, examine your options on this screen:
   - The search box as highlighted by the first callout bubble (callout area 1) is useful if you know the name or part of the name of the object that you want to find, and you have a large number of objects.
   - Callout area 2 shows the two tables you want. The problem is that at this point, you can only select one table. To solve this, click the **Select Multiple Items** check box that appears in callout area one; doing so allows you to choose multiple tables. You can now select the tables you wanted.
   - In callout area 3, notice that you can select **Select Related Tables**. If you selected **Include Relationship Columns** in Figure 1-2, this function performs as expected. This uses any referential integrity that is in the database to help determine what the related tables are. You do not need to choose this option as you only need the two tables mentioned in the opening of the exercise.

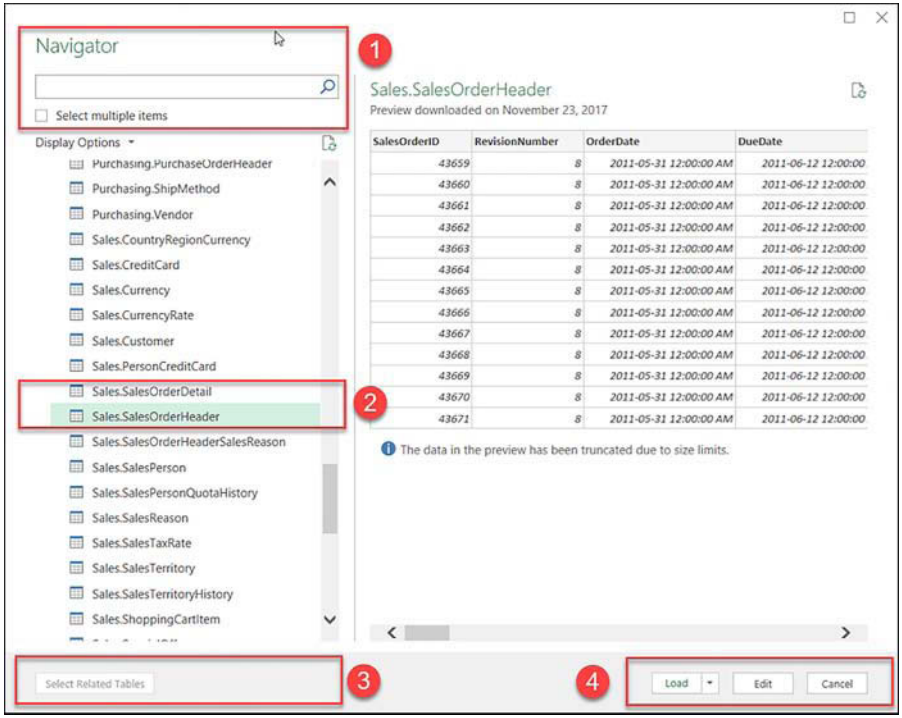

FIGURE 1-7 SQL Server Object Navigator expanded to the objects you want

6. Now that you have the two tables chosen, you can explore the options available in callout area 4 of Figure 1-7. If you are ready to transform and cleanse the data in the selected tables, click the **Edit** button to open the Query Editor. Do not click **Edit** now, because we will do that in Skills 1.2 and 1.3. In this section, we explore the **Load** button. Click on the drop-down arrow on the right side of the **Load** button to open the dialog box shown in Figure 1-8.

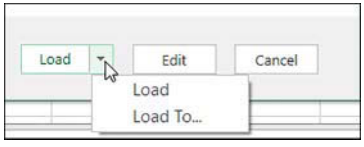

FIGURE 1-8 Load Options

7. From here, click **Load** to open the **Import Data** dialog in Figure 1-9. Choose whether to load your data and where to store it in the **Import Data** dialog box. The first three radio buttons under **Select How You Want To View This Data In Your Workbook** (**Table**, **PivotTable Report**, and **PivotChart**) all load data directly into the Excel workbook. Notice that if you select one of the first three radio buttons, you also have the option of loading data to the data model by checking **Add This Data To The Data Model**. Be aware that if you choose one of the first three options and check the **Add This To The**

Data Model option, you double up the data in your workbook because it is stored in both the object chosen by your radio box selection and the data model. It is not recommended to load data to both locations; in fact, now you are encouraged to load to the data model. You also have the **Only Create Connection** radio box, which will *not* load the data anywhere. For our example, choose **Only Create Connection** and click **OK** to create the connection in Excel as shown in Figure 1-10.

8. Once you have reviewed the figures below, you can close your Excel spreadsheet.

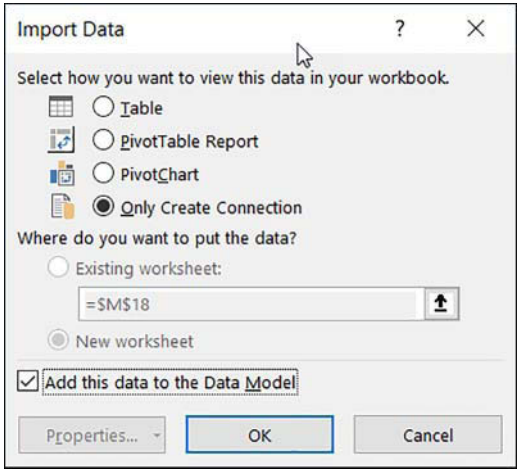

FIGURE 1-9 Import Data options

NOTE **DEFAULT QUERY LOAD SETTINGS**

The default query load settings can be configured in the Query Editor. Refer to the article below that discusses the default behavior and configuration options:

https://support.office.com/en-us/article/Add-a-query-to-an-Excel-worksheet-Power-Query-ca69e0f0-3db1-4493-900c-6279bef08df4#setdefault.

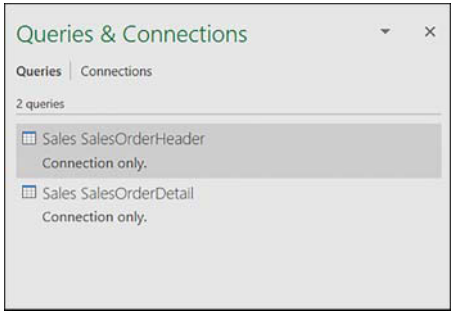

FIGURE 1-10 Queries & Connections Listing

Microsoft Access

Access is a popular desktop management system in many organizations. To connect to an Access database follow the steps below:

1. To connect to Access, select the **Data** tab > **Get Data** > **From Database** > **From Microsoft Access Database**.
2. You are presented with a Windows **Import Data** dialog box where you can navigate to the database file to which you want to connect. Choose the AdventureWorks2014 Access database in the book supplied folder at \Chapter 1\Access\ and click Open
3. Once you have chosen the database, you are presented with the **Navigator** dialog box as shown in Figure 1-11.
4. You may click Cancel.

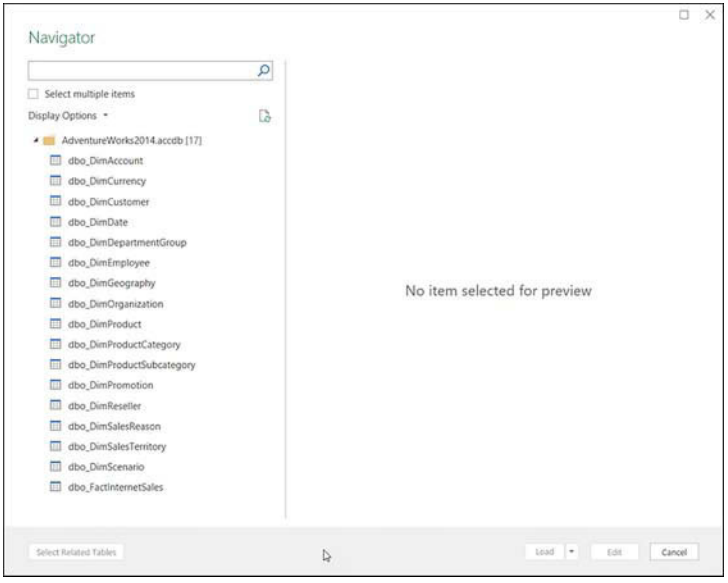

FIGURE 1-11 Access Database Object Navigator

If you had tried to connect to an Access database with a password, you would have encountered the error displayed in Figure 1-12.

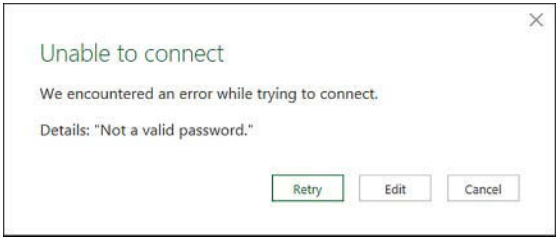

FIGURE 1-12 Unable to connect to a password-enabled Access Database

Analysis Services

Analysis Services provides dimensional data that is well suited for business reports and client applications such as Power BI, Excel, Reporting Services, and other data visualization tools. Data in Analysis Services can be structured in one of two ways:

- Multidimensional OLAP cubes
- Tabular models

You have two options for analyzing data in Analysis Services. First, you can use a live online connection which enables you to slice and dice data in the cube in a PivotTable or PivotChart without loading the data into your workbook. Second, you have the option to import the data from the cube into your workbook.

If you installed the Tabular and Multidimensional Adventure works databases as directed earlier in the Note titled *Downloading the Adventure Works databases*, you can follow along with this example. Follow these steps to Connect to Analysis Services using a live connection:

1. Click **Get Data** > **From Database** > **From Analysis Services**.
2. On the opening screen of the **Data Connection Wizard** in Figure 1-13, choose a **Server Name** and add type your **Log On Credentials**. In the **Server Name** field, you can either choose to connect to a tabular or multidimensional instance of Analysis Services. Click **Next** once you are finished.

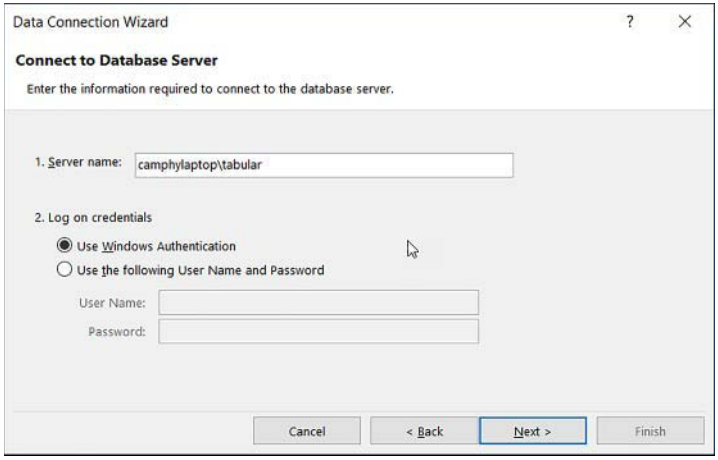

FIGURE 1-13 Data Connection Wizard to connect to SSAS Server

3. In this example, you connect to a tabular instance, which means you will see the **Data Connection Wizard** screen in Figure 1-14, where you choose the database to which you want to connect. Choose **Adventure Works Internet Sales Model** and click **Next**.

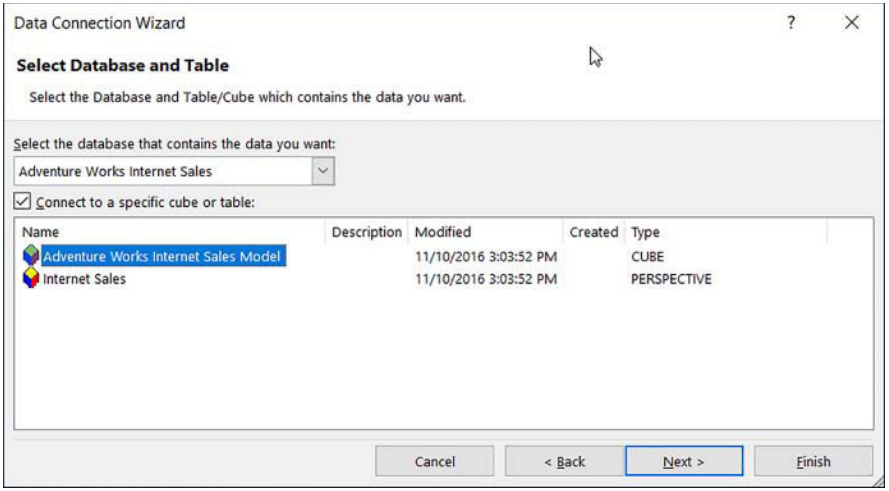

FIGURE 1-14 Select Database and Table/Cube to connect to

4. Next, you Save Data Connection File and Finish in the dialog shown in Figure 1-15. Doing saves your data as an Office Data Connection File. You provide both a **File Name** and a **Friendly Name** along with any **Search Keywords**, which help with searchability. Also, click **Authentication Settings** to modify those settings if needed. Once you have the values configured, click **Finish** and the Import Data dialog will appear, as shown in Figure 1-16.

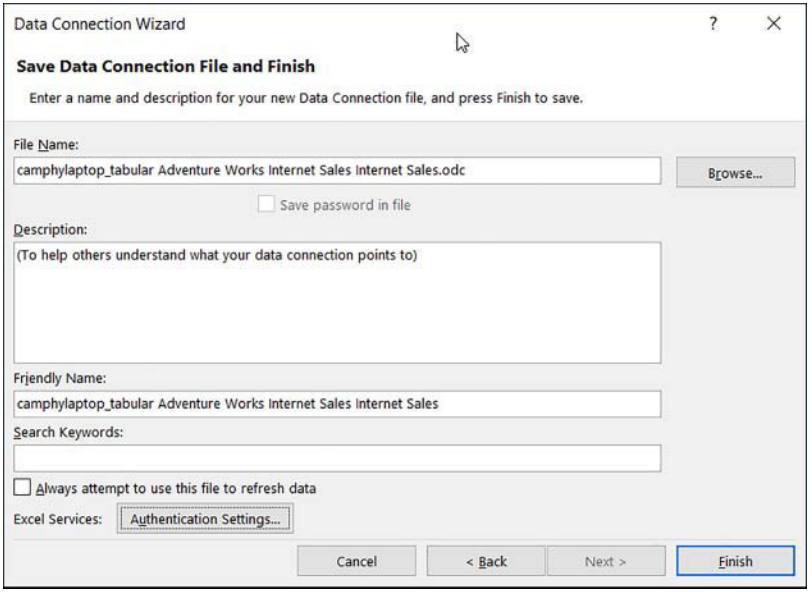

FIGURE 1-15 Save Data Connection File and Finish

***NEED MORE REVIEW?* OFFICE DATA CONNECTION FILES**

See the following article for more information on Office Data Connection Files: *https://support.office.com/en-us/article/Create-edit-and-manage-connections-to-external-data-89d44137-f18d-49cf-953d-d22a2eea2d46*.

***NEED MORE REVIEW?* CONNECTING TO AZURE ANALYSIS SERVICES**

For more about Azure Analysis Services, see the article at *https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-odc*.

5. In the Import Data dialog as shown in Figure 1-16, notice that you cannot select to store data in a table or the data model. Recall that with a live connection, no data is stored in the workbook.
6. Click Cancel.

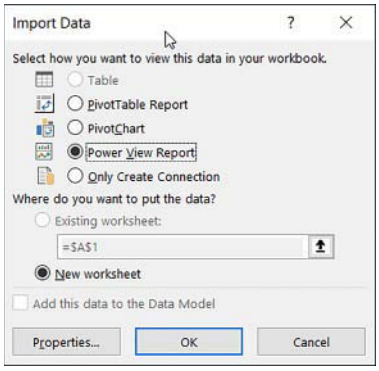

FIGURE 1-16 Import Data dialog

Now that you have completed the live connection, you can do the steps to import data into the workbook:

1. Click **Get Data** > **From Database** > **From SQL Server Analysis Services Database (Import)**.
2. You are presented with the **SQL Server Analysis Services Database** dialog in Figure 1-17. Here you are required to enter a **Server** with an optional port number and an optional **Database**. Also, you have an option to create an MDX or DAX query depending on the type of server to which you are connecting. Once you are done, click **OK**.

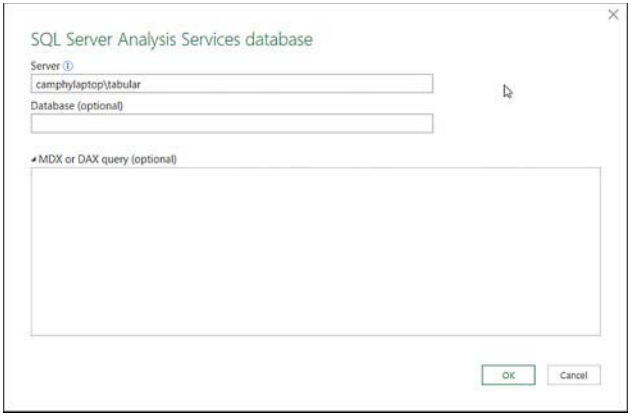

FIGURE 1-17 SQL Server Analysis Services database connection

3. If this is your first time connecting to this resource, you are prompted to enter credentials. Enter them and select **Connect.**
4. In the **Navigator** dialog in Figure 1-18, you navigate the objects on the server to which you are connected. From this point, choose to **Load** to the default location, **Load to** a select location, or **Edit** to begin transforming the data. You can click **Cancel** at this point, as this is as far as we will go.

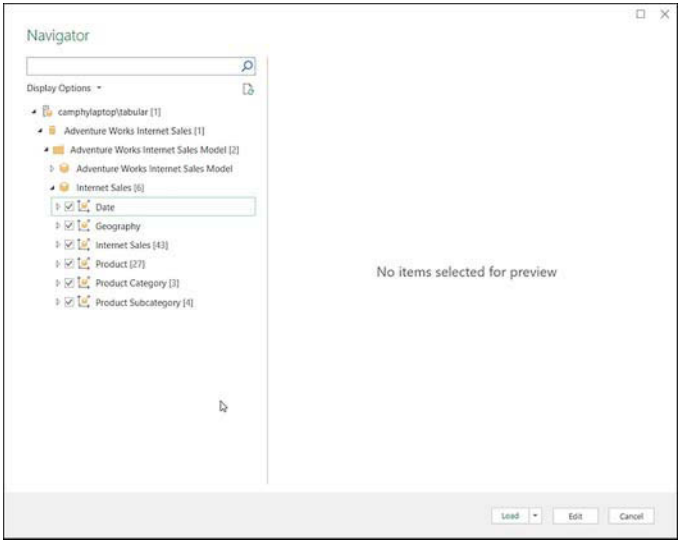

FIGURE 1-18 Analysis Service Object Navigator

NEED MORE REVIEW? **GETTING DATA FROM AN ANALYSIS SERVICES**

See this article for more information on how to retrieve data from Analysis Services: *https://support.office.com/en-us/article/Get-data-from-Analysis-Services-ba86270b-5cc2-4bb9-a21d-8bafc20f0cd3*.

Oracle

Follow these steps to connect to an Oracle database:

1. To get data from an Oracle Database, click the **Data** tab > **Get Data** > **From Database** > **From Oracle Database**. To connect, you first need to ensure that you have Oracle client software v8.1.7 or greater on your computer. If not, you receive the error shown in Figure 1-19.

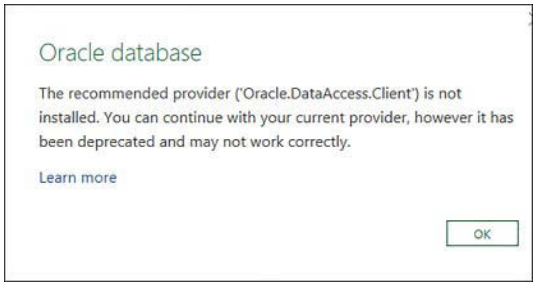

FIGURE 1-19 Oracle client error

NOTE **INSTALLING THE NECESSARY DRIVERS**

If you encounter the issue as described in Figure 1-19, you can go to the following location to learn more about which drivers to install *https://support.office.com/en-us/article/Connect-to-an-Oracle-database-Power-Query-d7fbd231-a705-4eb7-83b3-a66bfb678395*.

2. Once you have resolved the driver issue, you can continue to the **Oracle database** dialog box, as shown in Figure 1-20. Here you configure the options much like when configuring an SQL Server connection. Examine the options below and click **OK** when you have it configured properly.
   - **Server** Specify the Oracle Server. If a SID is required, it can be specified as Server-Name/SID.
   - **Command Timeout In Minutes (Optional)** This allows the SQL query to time out should the query not return in the allotted timeframe.
   - **SQL Statement (Optional)** If you want to import data using native database query, specify your query here.
   - **Include Relationship Columns** If chosen, this allows you to choose the **Select Related Table** option as shown in Figure 1-7. If not, the metadata that drives the

relationships are not included and no related tables are found, even if you know referential integrity exists at the database layer. The default selection is unchecked.

- **Navigate Using Full Hierarchy** If selected, you can navigate the Hierarchy of SQL Server objects from the server down to the databases, then schemas, and finally objects within schemas. If it is not selected, you navigate from the server to the databases, then all objects from all schemas. The default selection is unchecked.

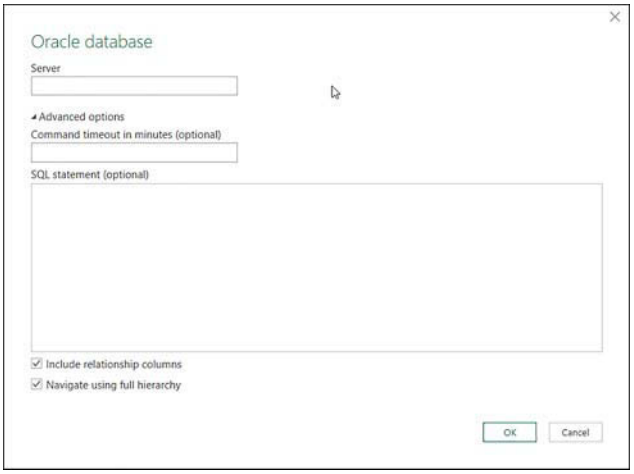

FIGURE 1-20 Oracle Connection Configuration

3. If the database requires credentials, enter them in the **Access A Database** dialog box.
4. Click **Connect**.
5. There is no need to save your work.

Other Database Management Systems

Excel can connect to many other Database Management Systems (DBMS) outside of SQL Server, SQL Server Analysis Services, Oracle, and Access. The pattern to connect repeats through the various DBMS. As seen in the Oracle example, one of the first roadblocks you might encounter is not having the correct drivers installed to allow Power Query to make the connection. This error comes even before you have a chance to specify server names and credentials.

NOTE **INFORMATION ON HOW TO CONNECT TO OTHER DATABASE MANAGEMENT SYSTEMS**

For more information on how to connect to other database systems, read the following article:*https://support.office.com/en-us/article/Import-data-from-external-data-sources-Power-Query-be4330b3-5356-486c-a168-b68e9e616f5a*.

Text/CSV

Text and CSV data are very common file types when doing analytics.

Follow these steps to connect to a Text/CSV source:

1. To get data from a text file, click the **Data** tab > **Get Data** > **From File** > **From Text/CSV**.
2. When prompted with the Windows **Import Data** file dialog box, navigate to \Chapter 1\Advanced Example 1\Append Examples\United States Sales.txt. This is a tab-delimited file. You are then presented with the dialog shown in Figure 1-21. Note the following options that you may configure. When complete, click Cancel.
   - **File Origin** Power Query detects the File Origin.
   - **Delimiter** Power Query detects the delimiter in use. Available delimiters are:
     - Tab
     - Comma
     - Colon
     - Equals Sign
     - Semicolon
     - Space
     - Custom
     - Fixed
   - **Data Type Detection** This is where you choose if and how Power Query performs data type detection. The options are:
     - Based on first 200 rows; this is the default
     - Based on entire dataset
     - Do not detect data types

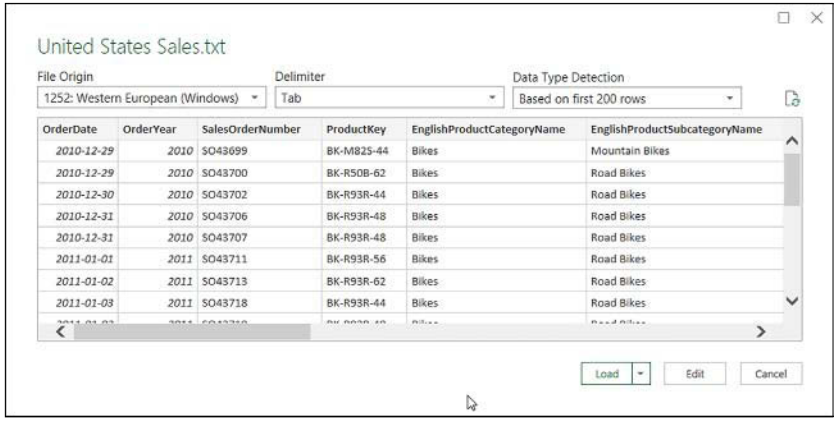

FIGURE 1-21 Text file configuration

***NEED MORE REVIEW?* POWER QUERY INTERNATIONALIZATION**

For information about Power Query Internationalization, see the following article: *https://support.office.com/en-us/article/Internationalization-Power-Query-d42b9390-1fff-413f-8120-d7df0ced20b9*.

Connect to XML

XML or eXtensible Markup Language has been around for many years and is now a common format for moving and storing data. It was designed to be both human- and machine-readable. Follow these steps to Connect to an XML source:

1. Click the **Data** tab > **Get Data** > **From File** > **From XML**.
2. Find the file in **\Chapter 1\XML JSON\XML Internet Orders.xml**.
3. When you click **Open**, the **Navigator** appears, as shown in Figure 1-22. Here you can choose which node you would like to process.
4. When complete, click Cancel.

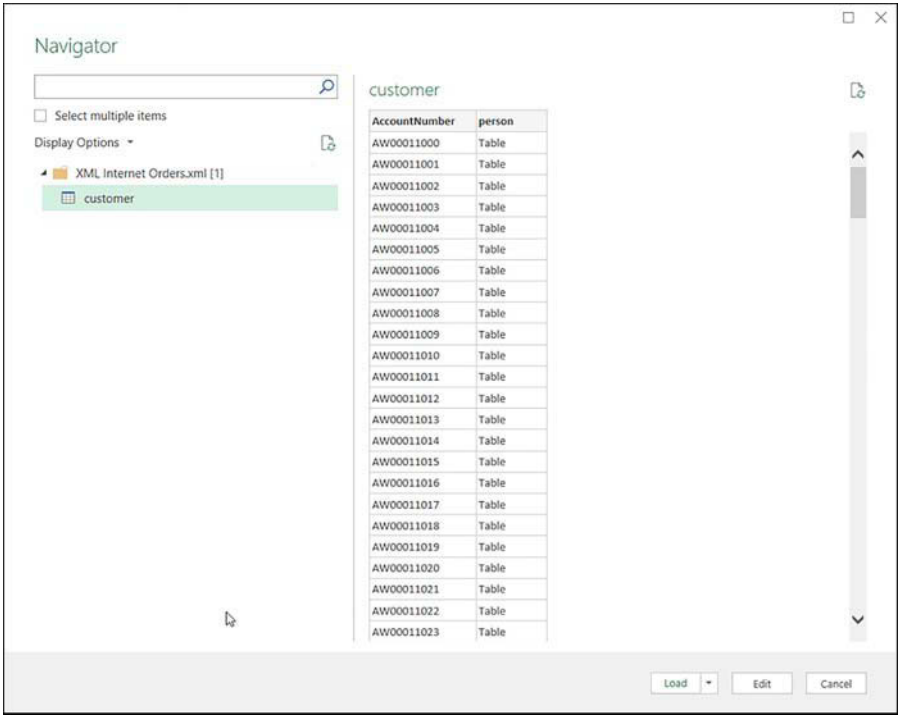

FIGURE 1-22 XML Navigator

Connect to JSON

JSON (or JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write. It is also easy for machines to parse and generate. It has an advantage over XML in that similar data volumes tend to be much smaller when wrapped into JSON.

Follow these steps to Connect to a JSON source:

1. Click the **Data** tab > **Get Data** > **From File** > **From JSON**.
2. Find the file in \Chapter 1\XML JSON\JSON Internet Orders.json.
3. When you click **Import** in the **Import Data** dialog box, the **Query Editor** appears, as shown in Figure 1-23.

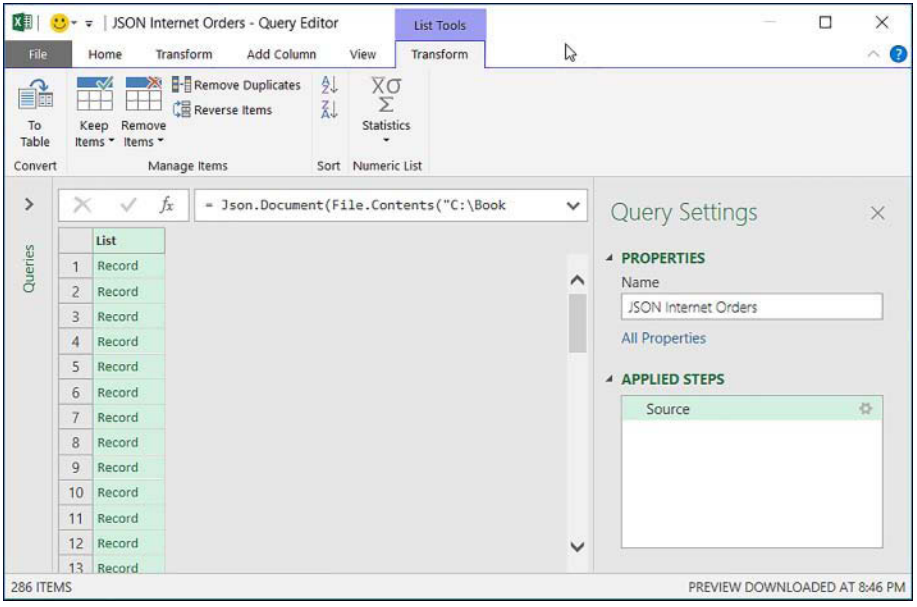

FIGURE 1-23 Initial Query Editor with JSON source

Connect to a folder

It is very common in business scenarios to have multiple files that are divided up among users in the organization and that are all structured in the same way. The only difference is the data they contain. Corporate budgeting is a very good example of this. File structures are tightly controlled by the managers of the budget process. This control around the structure enables the final data extraction process to occur without error.

In situations like this, the folder connector is a very useful way to use multiple files in one operation, placing them into a single table. The number of files can go up or down, and Power Query grabs all the files that are in the folder. The criteria for this:

- All files must reside in the same folder.
- All files must share the same structure.

Follow these steps to Connect to a Folder source:

1. Click the **Data** tab > **Get Data** > **From File** > **From Folder**.
2. In the **Folder** dialog, click **Browse**.
3. In the **Browse For Folder** dialog box, navigate to the folder at \Chapter 1\Folder\. Once you have the folder, click **OK**.
4. In the **Folder** dialog, now click **OK**.
5. When this is complete, you will be presented with the dialog shown in Figure 1-24 where you can see the six files that are in the folder. Click **Combine** > **Combine & Edit** to bring them into one table.

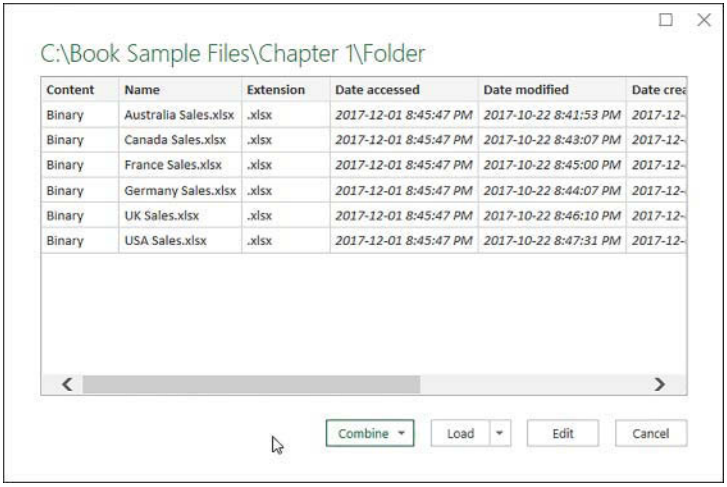

FIGURE 1-24 Files in the folder that was chosen

6. From here you are brought to the **Combine Files** dialog in Figure 1-25. Here you choose:
   - **Example File** Choose which file to use as the sample file.
   - **Select The Object To Be Extracted From Each File** I have chosen Sheet 1 from our example.
   - **Skip Files With Errors** This option allows you to skip any files that contain errors.

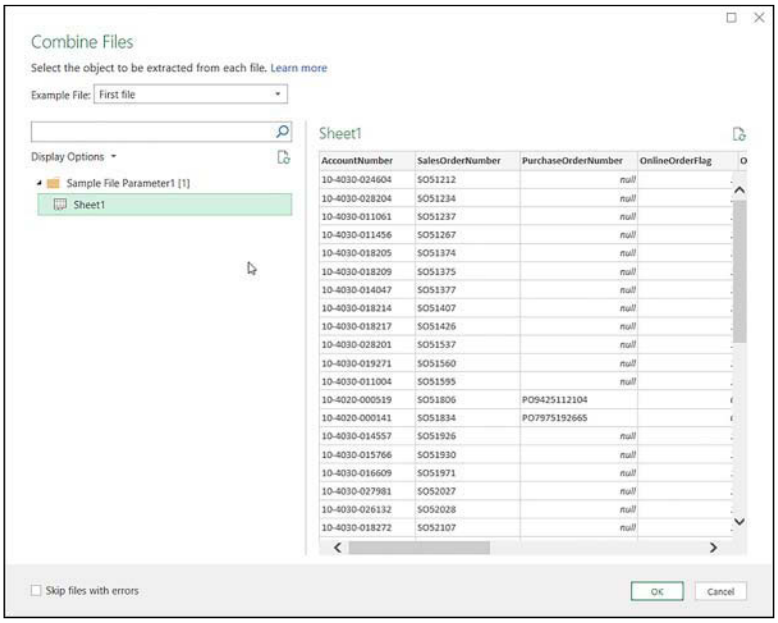

FIGURE 1-25 Combine Files dialog box

7. Click **Cancel** at this point. In Skill 1.2, you go through an end-to-end example on how to perform these steps along with Transforms.

Connect to a SharePoint folder

Fundamentally, connecting to a folder and a SharePoint folder are similar. SharePoint typically provides a more flexible and portable means for sharing files than a traditional file share or folder. Follow these steps to Connect to a SharePoint Folder source:

1. Click the **Data** tab > **Get Data** > **From File** > From SharePoint Folder.
2. On the **SharePoint folder** dialog, enter the root URL for the SharePoint Site, not including subfolders. Click **OK** when done.
3. In the next screen, select **Anonymous** > **Windows** > or **Microsoft Account** as seen in Figure 1-26. Here you can also indicate at which level in the SharePoint environment the permissions will apply. There are three levels where permissions can be applied in this example.
4. You may click Cancel.

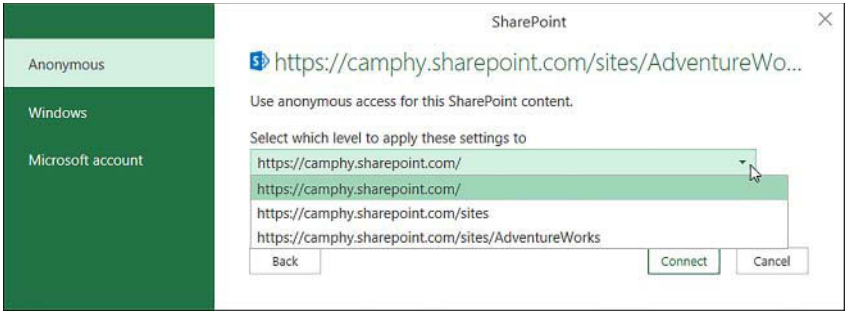

FIGURE 1-26 SharePoint site folder configuration

Connect to Microsoft SQL Azure and Big Data

Azure data sources are growing in popularity due to their total cost of ownership, their scalability, and their flexibility. These sources are used by more and more organizations, so the need to easily access data increases. Excel can connect to the following Azure sources:

- Azure SQL Server
- Azure SQL Data Warehouse
- Azure Data Lake Store
- Azure HDInsight
- Azure BLOB
- Azure Table Storage

> ***NOTE*** **AZURE SUBSCRIPTION**
>
> **If you choose to try any of the examples in this section, you need access to Azure. For this, you can set up a free trial account at the following location: *https://azure.microsoft.com/en-ca/free/*.**

Azure SQL and Azure SQL Data Warehouse

Connecting to Azure SQL and Azure SQL Data Warehouse is similar to connecting to SQL Server. To connect, you need to use the fully qualified name of your server, which is available in your Azure portal. The format is: *<your database name>.database.windows.net*. Additionally, you need to ensure that the firewall rules are correctly configured to access the databases once they are set up.

Azure Data Lake

Azure Data Lake is a scalable cloud-based data storage and analytics service. It gives users fast and efficient alternatives to deploying and managing big data infrastructure. Data of all shapes and sizes can be stored within Azure Data Lake, and if data processing is needed, you can take advantage of Azure Data Lake Analytics.

In this example, use the same example that is used throughout the book to show bringing in data from files and folders.

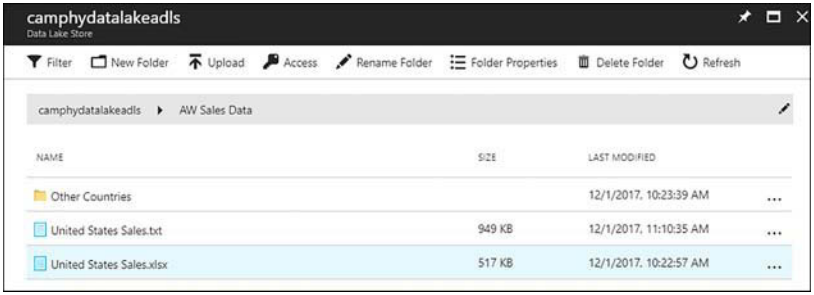

FIGURE 1-27 Azure Data Lake folder explorer

Follow the steps below to Connect to Azure Data Lake:

1. To get data from an Azure Data lake, click the **Data** tab > **Get Data** > **From Azure** > **From Azure Data Lake Store**.
2. You are presented with the **Azure Data Lake Store** dialog box shown in Figure 1-28. To connect, enter the following information and then click **OK**:
   - **URL** *adl://<your data lake store name>.azuredatalakestore.net/*
   - The URL can be:
     - The root directory of your data lake store
     - A specific folder
     - A single file

FIGURE 1-28 Azure Data Lake Store URL dialog box

3. You are then prompted to connect with an organizational account that has access to the resource as shown in Figure 1-29. Sign in with those credentials and click **Connect** when you are done. You are presented with the dialog shown in Figure 1-30.

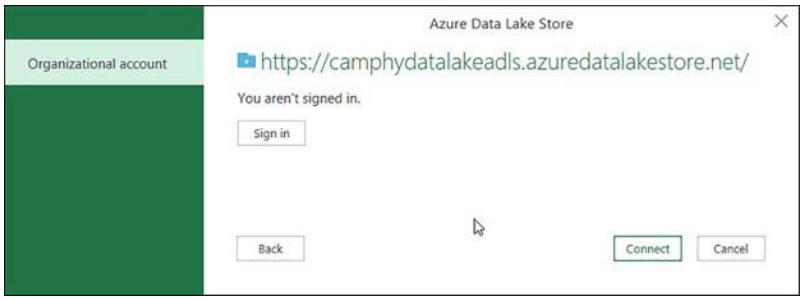

FIGURE 1-29 Azure Data Lake Store Sign in dialog Box

4. You may click **Cancel.**

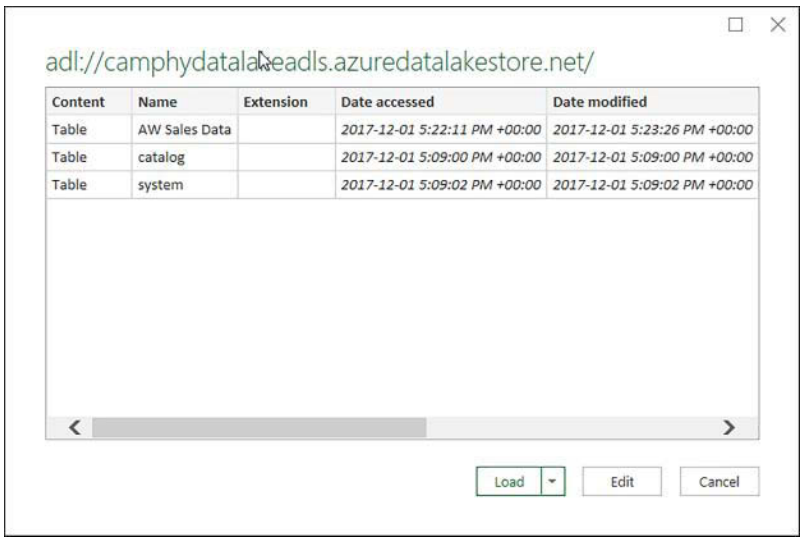

FIGURE 1-30 Azure Data Lake Store root level folder

NEED MORE REVIEW? **CONNECTING EXCEL TO AZURE DATA LAKE STORAGE**

Consult the following article for more details on how to work with Azure Data Lake:

https://blogs.msdn.microsoft.com/azuredatalake/2017/07/19/analyze-data-in-azure-data-lake-store-using-familiar-and-powerful-Excel-2016/.

Import from Excel workbooks

Data can be imported from workbooks external to the one that you are working in, as well as data from within your current workbook. For example, you can have tables and ranges in your workbook that you want to import. Now, it is time to see this option in an example.

Follow these steps to **Get Data** from a **Table/Range**:

5. With Excel closed, open the workbook at **\Chapter 1\Excel\Table Range Example.xlsx** as you would normally open and xlsx file.
6. In this file, you can see two objects: an Excel table named tblSales and a Named Range, rngRegions.
7. First, highlight any part of the Excel table and then click **Get Data** > **From Other Sources** > **From Table/Range**. The Query Editor appears where you can perform any shaping. Click **Close & Load** to move back to Excel.
8. Now highlight the entire range and click **Get Data** > **From Other Sources** > **From Table/Range**. You are are taken to the Query Editor where you can perform any shaping. Click Close & Load to move back to Excel.
9. Notice how both the query names have been picked up from the name of the Excel object that you have imported.

Link to data from other sources

There are many other data sources to which Power Query can connect. Below are the categories of other sources and the specific sources to which you can connect.

From Online Services

With the popularity of online applications, more and more data now resides in cloud-based solutions. For example, in a CRM-based Analytics scenario, some customer data might reside in Dynamics 365 CRM, and additional information might reside in Facebook. Power Query allows you to mash these sources together to help enable a 360-degree view of your customer. Get & Transform functionality makes it easy to connect to the following online sources:

- SharePoint Online List
- Microsoft Exchange Online
- Microsoft Dynamics 365 (online)
- Facebook
- Salesforce Objects
- Salesforce Reports

From Other Sources

There are many other connectors available, such as the following:

- Web
- Microsoft Query
- SharePoint List
- OData Feed
- Hadoop File (HDFS)
- Active Directory
- Microsoft Exchange
- ODBC
- OLDB
- Blank Query

Power Query can connect to Web sources

When doing analytics, you might find the need to acquire data from external sources to augment solutions. For example, you might need population data as part of your work. If your organization does not have this available internally, you can get it yourself and mash it in. For this example, look at getting population data from Wikipedia by following these steps to connect to a Web source: **Get Data** > **From Other Sources** > **From Web**.

1. You are presented with the **From Web** dialog box shown in Figure 1-31. If you choose the Basic option, you are asked to provide type URL. For this demo use the following URL and click OK: *https://en.wikipedia.org/wiki/List_of_countries_by_population_(United_Nations)*.

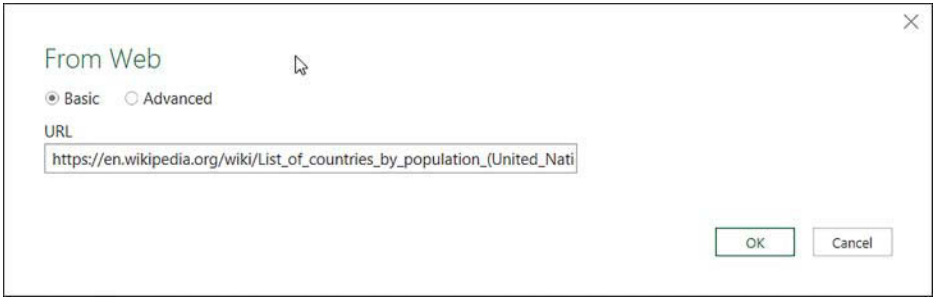

FIGURE 1-31 Web URL configuration dialog box

2. In the Access Web content dialog shown in Figure 1-32, you configure credentials to access the data. You can choose Anonymous, but you also have the following options to choose from, depending on the location of the web resource you are trying to connect to:

- Anonymous
- Windows
- Basic
- Web API
- Organizational account

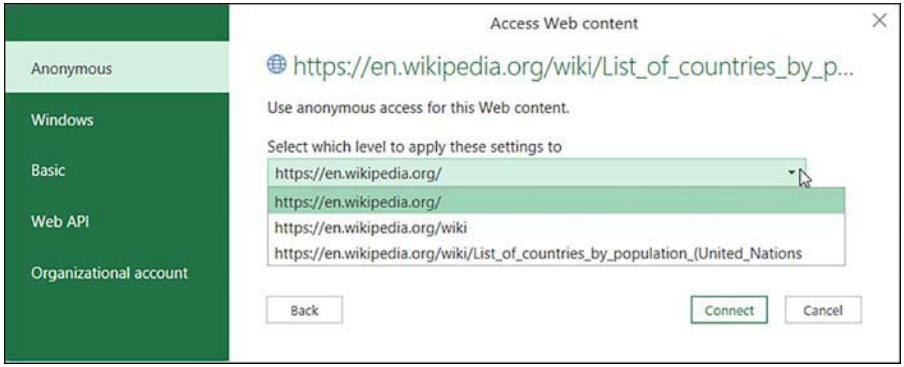

FIGURE 1-32 Web Content configuration dialog box

3. Also, note that you need to use Select Which Level To Apply These Settings To. If you click the drop-down box in the Access Web Content dialog box shown in Figure 1-32, you can see three levels where security can be applied: the root of Wikipedia, the next level down in the folder structure (wiki), or the actual web page itself. This is discussed more in the section on Privacy Levels in Power Query. For now, leave it at the root level, and click Connect.
4. You should now see the Navigator window as shown in Figure 1-33. Power Query inspects the web page for suitable structures to import and has found three objects that are listed. Click through each to see what they look like. You need to select Countries And Areas Ranked By Population In 2017. Note that when you have this highlighted, you can toggle between the Table View and Web View on the right of the window. This is helpful in situations in which there are many options, and you want to preview what you are selecting.
5. Go ahead and click Edit to move the Query Editor to do further processing, or you may click Cancel.

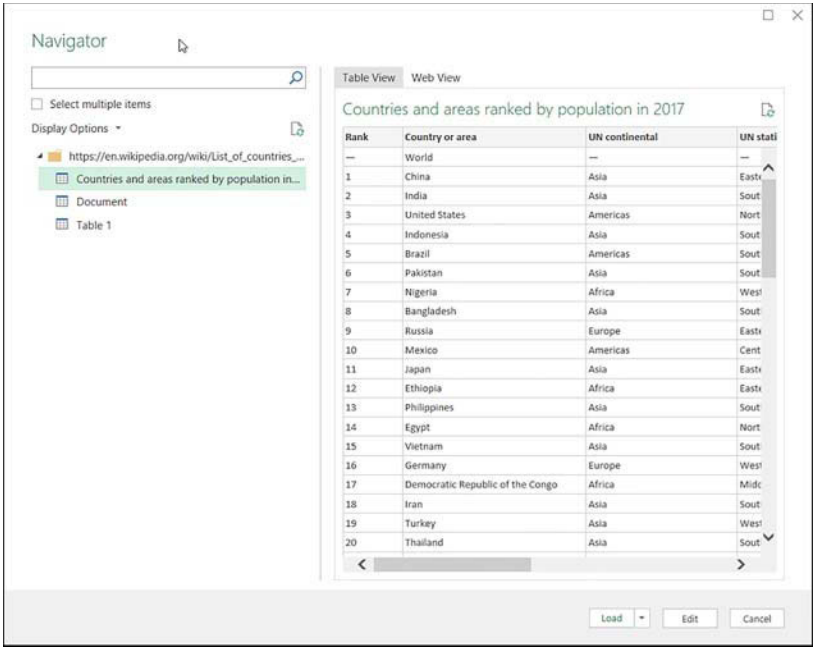

FIGURE 1-33 Web Page Object Navigator

Blank Query

You can use the Blank Query to create advanced queries that use the M language within Power Query. M provides a wide variety of formulas that are used to build complex expressions and transformation steps.

When you choose a Blank Query, you are immediately taken to the Query Editor where you can begin to build out your query using the various wizards, or you can go directly into the Advanced Editor to code your steps. More information about the Advanced Editor is provided in Skill sections 1.2 and 1.3.

Privacy Levels

Now that you have worked with several data sources, it is worth discussing Privacy Levels. Privacy Levels are rules that define isolation levels between data sources. These rules might affect performance of queries, and in some cases, your queries are not executed at all if Privacy Levels conflict between sources.

To view the current Privacy Levels in your workbook, click **Get Data** > **Data Source Settings**. Note, this can be done from the Query Editor as well. You should see the Data source settings dialog shown in Figure 1-34. From there, you select the multidimensional source and click Edit Permissions. Note that you can also change the Credential used at this point. You have the option to change Privacy Level to the following values:

- None
- Public
- Organizational
- Private

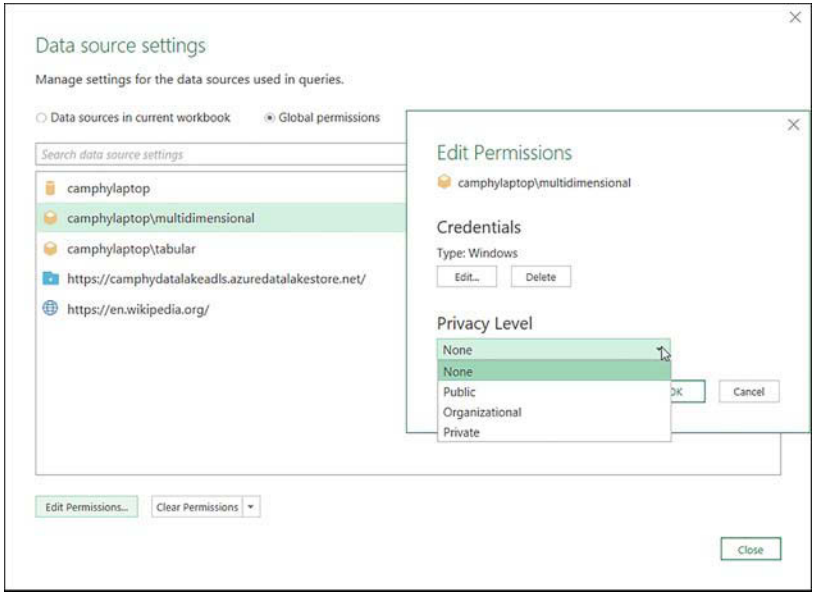

FIGURE 1-34 Data source settings

To view the settings on how your workbook treats privacy levels, click **Get Data** > **Query Options**. The Query Options dialog will appear as shown in Figure 1-35.

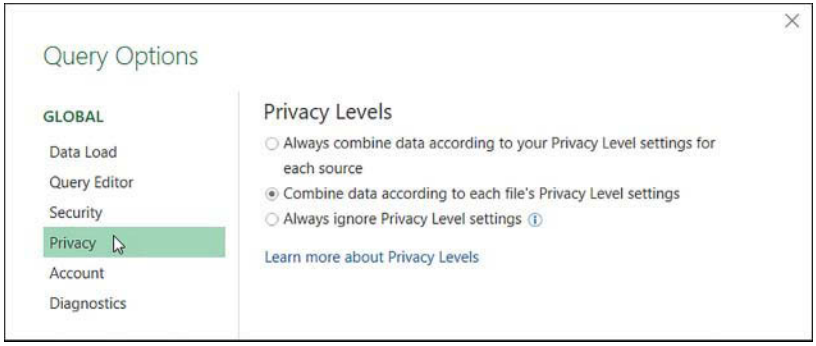

FIGURE 1-35 Privacy settings in Query options

NEED MORE REVIEW? **PRIVACY LEVELS**

For a more in-depth discussion around Privacy Levels, consult the following article: *https://support.office.com/en-us/article/Privacy-levels-Power-Query-CC3EDE4D-359E-4B28-BC72-9BEE7900B540*.

Skill 1.2: Perform data transformations

Now that you have seen examples of how to connect to the vast array of data sources that are available in Excel, let's shift attention to the different methods that are available to users to transform data using the Query Editor, which is the primary focus of Skill 1.2 and Skill 1.3.

Rarely is data in the state that you need it to be in for production quality reporting. If it is, count yourself lucky. If not, the Query Editor has a very rich set of features that allows users to perform transformations and cleansing activities with ease. You often hear this referred to as Data Shaping, which has become the more modern term for Extract Transform and Load (ETL).

The front-to-back process of importing data, performing transformations, and then visualizing data is highly iterative, and tools such as Excel are well-suited to enable fast iterations. Data can be taken from front to back, and as it is used at each stage, feedback can be provided that can be pushed back into earlier steps that make later steps in the process easier. For example, maybe your model has no easy way in the first iteration to perform analysis by Year. As this feedback is provided, the data shaping process can create a Year column, so that is available in the Data Model for end users. This is a simplistic example, but it is a very common way to iterate on improving models for widespread usage.

This section covers how to:

- Design and implement basic and advanced transformations
- Apply business rules
- Change data format to support visualization
- Filter data
- Format data

Design and implement basic and advanced transformations

In this skill section, we review the methods available in the Query Editor for performing basic and advanced transformations of your data. The bulk of time in data shaping is spent taking data tables and fields that come in many shapes and sizes and preparing them for easy to use and consistent analytics. Many times, when performing analysis, you are merely happy getting any data, no matter how good or bad it is. You then take advantage of techniques available to standardize and cleanse the data for use.

For the most part, the visual tools available in the ribbon and menu options more than cover your transformation needs. However, as you become more advanced with using the Query Editor you might start diving into the M language that is available for scripting steps by typing code versus using the GUI commands. The Power Query M formula language is optimized for building highly flexible data mashup queries. It's a functional, case-sensitive language which can be used with Power BI Desktop, Power Query in Excel, Get & Transform in Excel 2016, Power Apps, and now SQL Server Analysis Services 2017. Through this Skill, M scripts are highlighted as they are output from transformations using the GUI because this a great way to gain familiarity with the language. This is done by first showing the individual lines of code that are generated with each step. Then you are introduced to the Advanced Editor for viewing and maintaining the entire script that has been generated by applying transformation steps to your data.

NOTE **QUERY EDITOR AND POWER QUERY**

Originally, Query Editor was called Power Query, and to this date, still shows up in documentation on many sites, including the Microsoft Office support site. As of the time of this writing, the two terms are used interchangeably, and questions on the exam might show using either term.

Importing data to support basic transformations

Let's start off with basic transformations by importing data from sources that do not need much in the way of transformations. This approach is very common when bringing in data from a Data Warehouse where many of the transformation steps that need to be performed on data have already been handled by the warehouse team. However, even well-formed data warehouses occasionally do not have all the data for analysis in the state an analyst requires. Perhaps something was never incorporated into the warehouse, or maybe a business rule changed, or occasionally, unaccounted data sneaks into the warehouse. For these examples, import FactInternetSales and DimCustomer from the AdventureWorksDW2016 SQL Server databases.

Perform take these steps to make the connection to the data source:

1. Click **Data** tab > **Get Data** > **From Database** > **From SQL Server Database**.
2. In the SQL Server database dialog, configure the following options as below and in Figure 1-36, and then click OK.
   - **Server** Your server name
   - **Database** AdventureWorksDW2016

   Leave all other items as their defaults

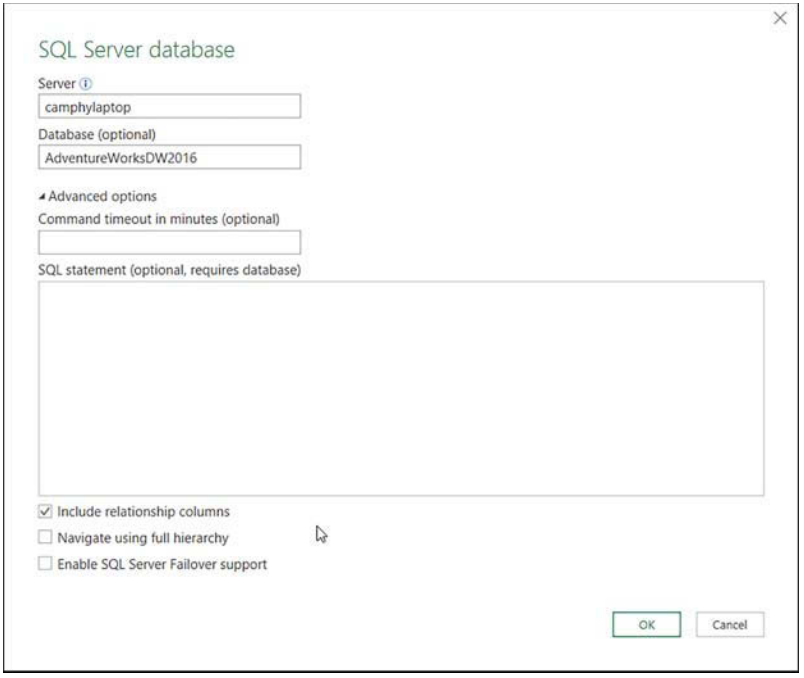

FIGURE 1-36 SQL Server Database configuration

3. In the Navigator dialog (as in Figure 1-37), you can check the Select Multiple Items check box so you can choose the FactInternetSales and DimCustomer tables that you want to bring in. Next, click the drop-down on the right side of the **Load** button and choose **Load To**.
4. Make the following selections in the **Import Data** dialog, and then click OK:
   - **Select How You Want To View This Data In Your Workbook** Ensure that Only Create Connection is selected.
   - **Add This Data To The Data Model** Ensure that this option is not selected.

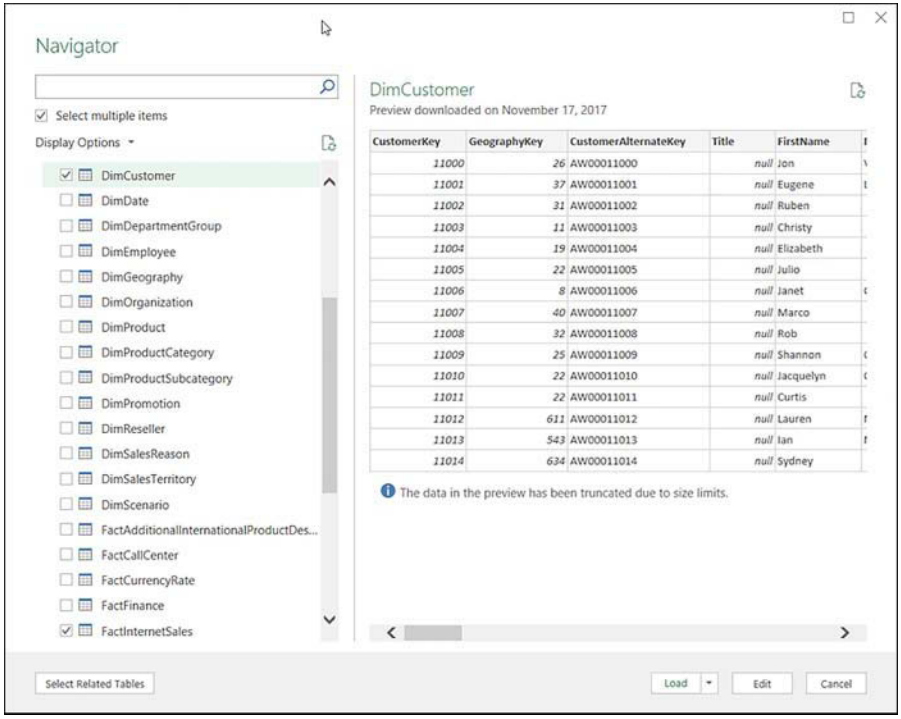

FIGURE 1-37 The data source Navigator

5. Once you have completed this, you have only created a connection to the targeted data source. At this point, no data has been brought into any Excel objects, which is fine for now because you are only working on transformations. Once the data is ready for consumption, you can load it to a consumable location. You should now see the screen in Figure 1-38. Both tables show up under Queries and are in a state of Connection Only. If you do not see the **Queries & Connections** pane on the right side of the screen, go the **Data** tab and in the **Queries & Connections** group, click the **Queries & Connections** command. This will toggle the pane on and off.

NOTE **ONLY CREATE CONNECTION**

Depending on the size of the data that you are loading, choosing to only create the connection in the Import Data dialog can help avoid a potentially lengthy load process when you are in the early stages of data shaping. Later when you are working in the Query Editor and want to save your work, you can close and load the work.

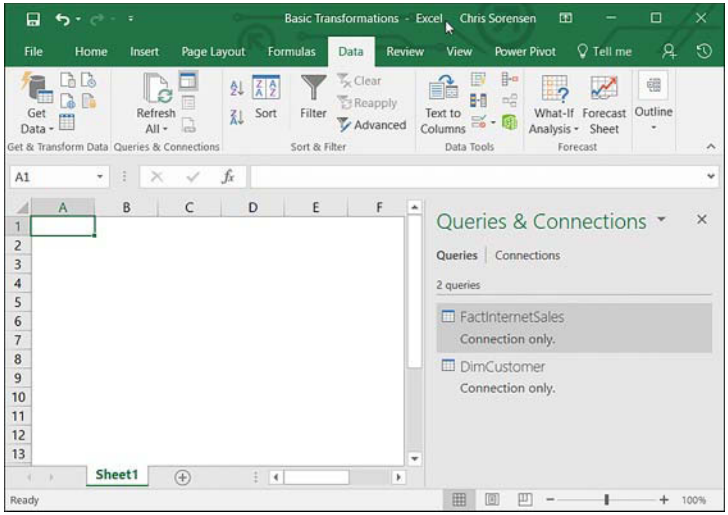

FIGURE 1-38 Excel window showing connections and queries

6. Now, look at a few options that are available from the Excel window shown in Figure 1-38. If you hover over either connection in the Queries & Connections pane, you can view the "peek" window as shown in Figure 1-39, which is a snapshot of information about the query. As you will see later in this section, this window also contains many functions that are available in the Query Editor window, as shown later in this section.

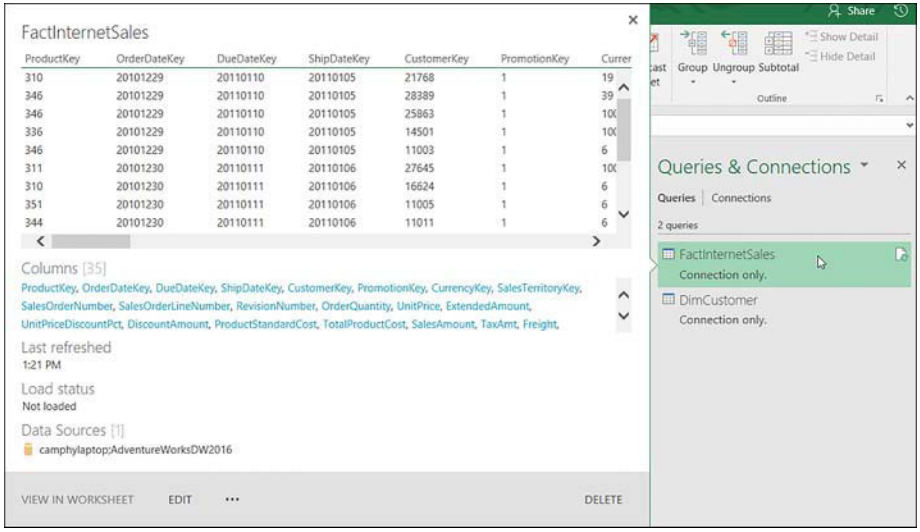

FIGURE 1-39 Query and connection peek window

7. Other options that are available in the pane can be found by right-clicking on one of the queries, as shown in Figure 1-40. As with the peek menu in Figure 1-39, many of the commands in this window can also be performed in the Query Editor.
8. Now that you have explored some of the options in the Excel window, right-click **FactInternetSales** and click **Edit** in the context menu to open the Query Editor (see Figure 1-40). This takes you directly to the highlighted query in the Query Editor, as shown in Figure 1-41.
9. Save your Query Editor work by clicking **Home** tab > **Close Group** > **Close & Load**. This will close the Query Editor and send you back to Excel.
10. Save your workbook as Chapter01Exercise01.xlsx as it will be used in subsequent demos.

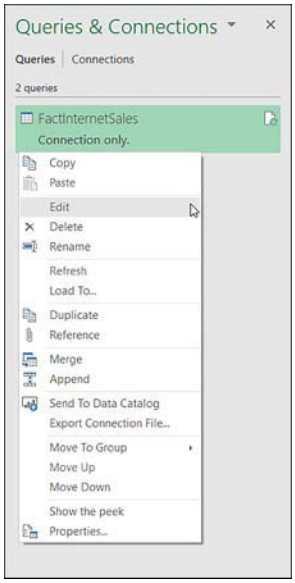

FIGURE 1-40 Query and context menu

Query Editor Overview

Before you begin the process of transforming data, review the parts of the **Query Editor** in Figure 1-41.

1. **Ribbon** This is where Data Shaping tasks are located. They are grouped into like functionality by tab and then by tab groupings. Keep in mind that many of the data-shaping tasks are also available in other locations.
2. **Queries pane** This lists the objects that you build over the course of shaping data. Also—as you will see soon—as this list of queries grows, you can add Groups, which are folders where similar objects can be placed. This will be demonstrated shortly.

3. **Formula bar** As you build data-shaping steps, the resulting M code that is generated for that step is displayed here. If you cannot see the formula bar as shown, click **View** tab > **Layout** group > **Formula** bar. You can also use the formula bar to type in your own custom M transformations.
4. **Query settings pane** Shows the name of the query and the Applied Steps that have been taken to transform the data. This will be discussed in detail as you build examples up.
5. **Data preview pane** Displays a preview of the data resulting from your query. It is in this window where you select columns that you are looking to shape. To help with performance, the Query Editor takes a snapshot of the data and caches it. If you need to refresh the cache, you have a few options. If you want to refresh only the query you have selected, you can click the **Refresh** Preview command from the Home tab, Query group above the formula bar shown in Figure 1-41. If you want to refresh the preview for all your queries, select the **Home** > **Query** group > **Refresh Preview** drop down, and choose Refresh All. Note that in this example, you can see that the last Data preview for FactInternetSales was from November 17, 2017, as shown in the very bottom right side of the screen in Figure 1-41. This is also indicated above the formula bar as shown in Figure 1-42.

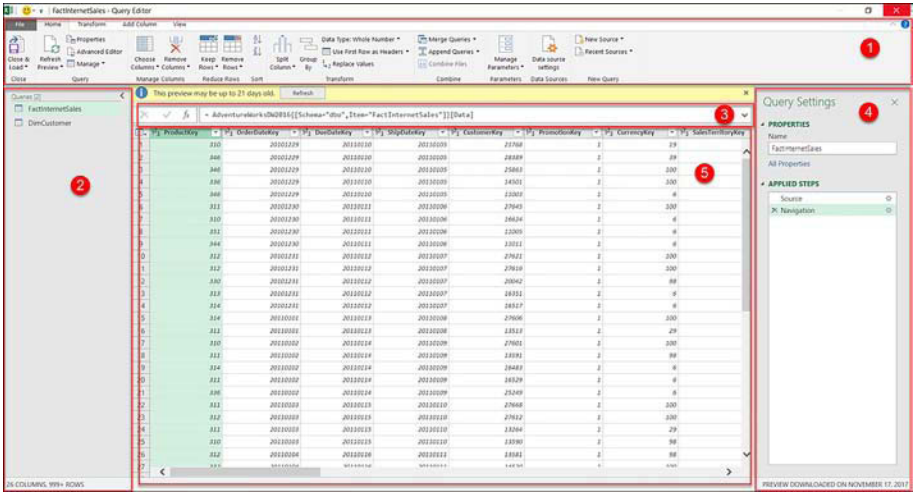

FIGURE 1-41 Query Editor

FIGURE 1-42 Data preview cache age indicator

Perform basic transformations

Now that you have reviewed the Query Editor landscape, let's perform some transformations on the FactInternetSales table, and then we'll look at the DimCustomer table. Ensure that the FactInternetSales query is selected in the Queries pane and that the Query Settings pane is open. If not, it can be toggled on and off from **Home** > **View** > **Query Settings**.

Let's pick up from the Importing data to support basic transformations steps where you imported FactInternetSales and DimCustomer. Note that you have already performed two steps, which are listed in the Query Settings pane under APPLIED STEPS. These came from the initial screens that you ran through to make the connection. The first step named Source shows the server you connected to and the second shows that you are bringing the FactInternetSales table from the dbo schema in the database named AdventureWorksDW2016. To view this metadata, click the gear icon to the right of each step.

Depending on your source, the Query Editor may attempt to recognize header rows and data types as well. Because you are connecting to a SQL Server, both can be determined through database metadata, so this step does not need to be performed.

1. The next thing you notice is that there are lots of columns in this table, many of which are not needed for this analysis; you should remove the unneeded columns. There are a few options for doing this. Follow the steps below to remove columns that you do not need: Ensure that you are in the Query Editor, which was the last step in the previous exercise.
2. With the FactInternetSales table selected in the Queries pane, click **Home** > **Manage Columns** > **Choose Columns** command drop-down. From here, you have two options available. The first option is to choose Go to column, which allows you to select a column from a list of sorted columns in the query. This is good for when you have a large table and want to get to a column without scrolling through the Data preview pane. The second option is Choose Columns, which allows you to choose only the columns you want to keep by selecting them in the dialog box. Click Choose Columns and then in the Choose Columns dialog box, ensure only the following columns are checked and click OK when complete:
   - CustomerKey
   - SalesOrderNumber
   - SalesOrderLineNumber
   - OrderQuantity
   - UnitPrice
   - UnitPriceDiscountPct
   - DiscountAmount
   - SalesAmount
   - TaxAmt
   - Freight
   - OrderDate

3. Notice That you now have a third step named **Removed Other Columns** as shown in Figure 1-43.

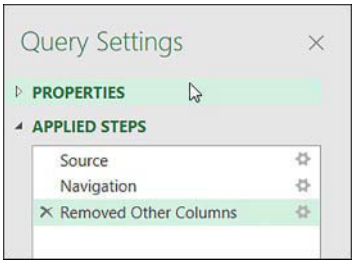

FIGURE 1-43 Query Settings

4. This step name may be meaningful now, but will it mean something to the next developer? Likely not. In instances like this, you should document your steps by highlighting the step you want to document, right-clicking on it to open the context menu, and then choosing Properties. You are then presented with the Step Properties dialog box shown in Figure 1-44; here, you can rename the step to something more meaningful and can also add a description, which is very useful for complex transformations. Click Cancel, as we will leave the default values.
5. Note if you just want to change the name of the step, you can do that by right-clicking the step name in the Applied Steps portion of the Query Settings pane, and then selecting Rename in the context menu. This enables the step name in Applied Steps to be directly edited.

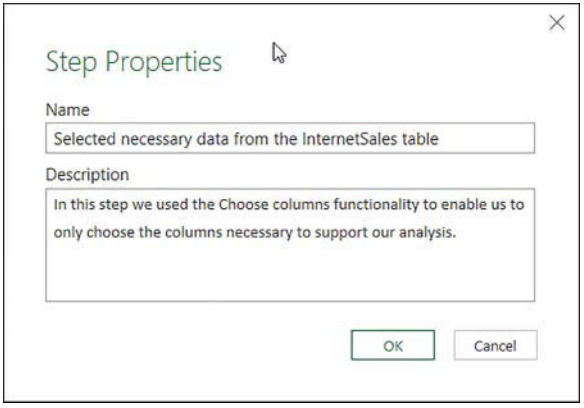

FIGURE 1-44 Step Properties

6. Look at the M code that was generated after you finished choosing the columns that you wanted. The code is in Listing 1-1 and can be viewed in the Formula bar in the Query Editor.

7. Save your Query Editor work by clicking the **Home tab** > **Close group** > **Close & Load**. This will close the Query Editor and send you back to Excel.
8. Save your workbook as **Chapter01Exercise01.xlsx** because it will be used in subsequent demos.

LISTING 1-1 Code generated by the Choose Columns step

```
= Table.SelectColumns(dbo_FactInternetSales,{"CustomerKey", "SalesOrderNumber",
 "SalesOrderLineNumber", "OrderQuantity", "UnitPrice", "UnitPriceDiscountPct",
 "DiscountAmount", "SalesAmount", "TaxAmt", "Freight", "OrderDate"})
```

EXAM TIP

M questions are on the exam. As opposed to having a separate section that calls out how to write M, I have chosen to demonstrate by example. As Applied Steps are created in the Query Editor, highlight the M code that is generated by the engine. This allows you to perform a function using the GUI, so you become familiar with the M code.

With the Chapter01Exercise01.xlsx file that you have created still open, let's explore the Query Editor further. Now that you have only the data you want to look at in the Data preview pane, review the data types that you have so far. The highlighted area of Figure 1-45 is one place where you can go to determine the data type for each column. The data types determine what type of operations you can perform on the column, and how much storage is required in the model.

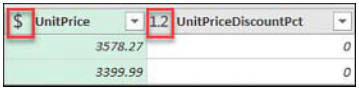

FIGURE 1-45 Data Types for each column

Clicking on the highlighted area in Figure 1-45 reveals a list of different supported data types, as shown in Figure 1-46. If you see the ABC123 icon displayed (not seen here), this signifies the Any data type has been set for the column. The data types can also be changed in the **Home** tab in the **Transform** group in the **Data Type** drop-down. Note that some data types exist only in Query Editor and are converted once you load the data. An example of this is the percentage data type, which is available in Query Editor, but once you load the data, it is converted to a decimal.

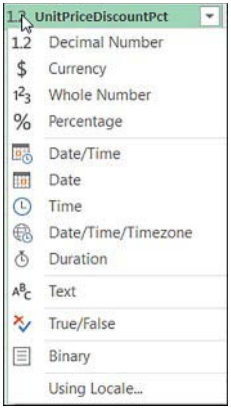

FIGURE 1-46 Possible Data Types context menu

As you can see, when you move into the section on DAX in Chapter 2.2, the available Data Types in the Query Editor contains two data types that are not available in Data or Report view: These are **Date/Time/Timezone** and **Duration**. When a column with these data types is loaded into the Data Model and viewed in Data or Report view, the following occurs. A Date/Time/Timezone data type is converted into a Date/Time, and a column with a Duration data type is converted into a decimal number. Follow these steps to change a data type:

1. Open the **Chapter01Exercise01.xlsx** file and start the Query Editor if you do not already have it open.
2. Change the Data Type for UnitPriceDiscountPct from a Decimal to a Percentage by highlighting the column in the Data preview pane and then navigating to the Home tab in the and the Transform group in the Data Type drop-down. Choose Percentage.
3. Observe the resulting M code in the Formula bar as shown in Listing 1-2.

LISTING 1-2 M Code generated the change in data type

```
= Table.TransformColumnTypes(#"Removed Other Columns",{{"UnitPriceDiscountPct",
Percentage.Type}})
```

If you need to rearrange the position of your columns, you have several options. First, you can select **Move** from the **Transform** tab, or right-click the column header and select **Move** from the context menu. Move provides you with options to move the column to the:

- Beginning of your query
- End of the query
- Left
- Right

Alternatively, you can drag and drop the columns where needed. Follow these steps to move the OrderDate column in FactInternetSales to the beginning of the Query:

1. Select the OrderDate column header in the Data preview pane.
2. Drag it into position at the beginning of the query column in the Data preview pane.
3. Observe the resulting M code in the Formula bar as shown in Listing 1-3.

LISTING 1-3 M Code generated by the Move columns steps

```
= Table.ReorderColumns(#"Changed Type",{"OrderDate", "CustomerKey",
 "SalesOrderNumber", "SalesOrderLineNumber", "OrderQuantity", "UnitPrice",
 "UnitPriceDiscountPct", "DiscountAmount", "SalesAmount", "TaxAmt", "Freight"})
```

4. Now let's add some extra columns to the FactInternetSales table to support analysis. In this example, you are going to add some additional date and time columns by taking advantage of some built-in functions. You need to add a year, month name, and month number to support analytics. Perform the following steps to add columns: Select the OrderDate column in the Data preview pane.
5. Go to the Add Column tab, From Date & Time group, and select Date as shown in Figure 1-47. Here you have several functions that you can perform on a Date Time column to easily extract parts of the date into new columns.

FIGURE 1-47 Date Transforms

6. Next choose **Year** > **Year**. This adds a new column at the end of the table with the value of year. The M code is shown in Listing 1-4. For each row in the table, this M code has determined the year value and has added it into the newly added Year column. If you do not like the name that the new column was given, you can edit the M code by changing the value in the "Year" to some other value; alternatively, you can do it in an additional step and perform a rename. Before you do that, add the other columns.

LISTING 1-4 M Code generated by adding the Year column

```
= Table.AddColumn(#"Reordered Columns", "Year", each Date.Year([OrderDate]),
 type number)
```

7. Repeat steps 1-3 two times, once to add a Month, which gives you a month number, and once to add a Month Name. When you are done, the table should look like the something like table shown in Figure 1-48. Recall that these additional columns will appear at the end of your table.

1.2 Year	1.2 Month	ABC Month Name
2010	12	December
2010	12	December
2010	12	December
2010	12	December
2010	12	December

FIGURE 1-48 Snapshot of current exercise steps

NOTE **DATE AND TIME TRANSFORMS**

If you want to transform Date and Time, you can do this either from the Transform tab or the Add Columns tab, but they produce different results. Choosing from the Transform tab transforms an existing column in place; choosing from the Add Column tab adds an additional column.

8. Now that you are done, take note of the data types. Year and month are both decimal formats, and the Month name is text format. Make the following changes to the data types:
   - **Year** should be a Whole Number
   - **Month** should be a Whole Number
   - **Month Name** can remain as Text
9. Now move the three columns to the beginning of the query by using the Move functionality. To do this, press the Control key and click on the three new columns. Now right-click to be presented with the context menu where you can choose **Move** > **To Beginning**.

The last thing that you want to do with FactInternetSales is to give the query a name, and give some of its columns better names. Follow these steps to rename a query and columns:

1. First, rename the query to a friendlier name, such as InternetSales. This is important because this name shows up in the data model once you choose **Load To The Data Model**. In the Query Settings pane under Properties, change the Name value to InternetSales as shown in Figure 1-49. Note that you can do this from the Queries pane by right-clicking on the Query and choosing Rename from the context menu.

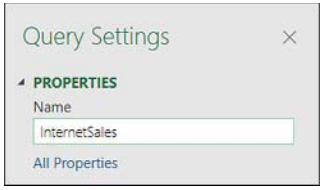

FIGURE 1-49 Query Settings Query Name

2. Now rename the following columns according to Table 1-1. You can do this by right-clicking on the column names in the Data preview pane and choosing **Rename.** Do the same by clicking **Transform tab** > **Any Column group** > **Rename**, or more easily by double-clicking on the column name in the Data preview pane and simply typing a new name. The resulting M code for this step is in Listing 1-5.

TABLE 1-1 Renaming InternetSales columns

Original column name	New column name
Month	Month Number
Month Name	Full Month Name

LISTING 1-5 M Code generated by adding the Year column

```
= Table.RenameColumns(#"Reordered Columns1",{{"Month", "Month Number"}, {"Month
 Name", "Full Month Name"}})
```

3. Finally, look at the cumulative result of the M code for all the transforms by going to the Advanced Editor from either the View or Home tabs while the InternetSales query is selected. The sample of the Advanced Editor is shown in Figure 1-50. Your code might be different depending how you did the above steps. Also, note that the dialog box does not explicitly title it as the Advanced Editor.

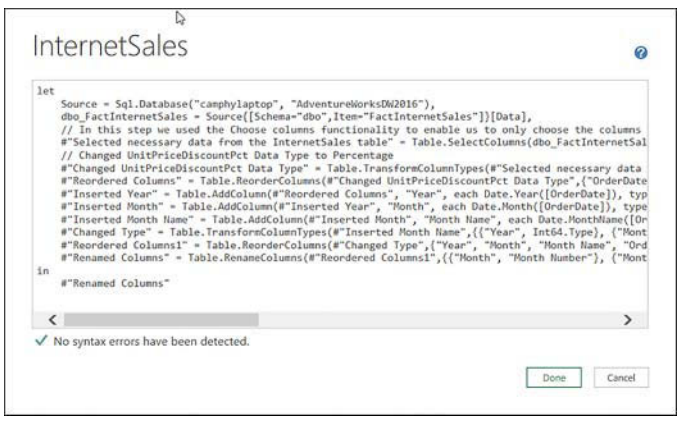

FIGURE 1-50 Advanced Editor

LISTING 1-6 All M Code applied to Fact Internet Sales

```
let
    Source = Sql.Databases("localhost"),
    AdventureWorksDW2016 = Source{[Name="AdventureWorksDW2016"]}[Data],
    dbo_FactInternetSales =
AdventureWorksDW2016{[Schema="dbo",Item="FactInternetSales"]}[Data],
    #"Removed Other Columns" = Table.SelectColumns(dbo_FactInternetSales,{"CustomerKey",
 "SalesOrderNumber", "SalesOrderLineNumber", "OrderQuantity", "UnitPrice",
 "UnitPriceDiscountPct", "DiscountAmount", "SalesAmount", "TaxAmt", "Freight",
 "OrderDate"}),
    #"Changed Type" = Table.TransformColumnTypes(#"Removed Other
 Columns",{{"UnitPriceDiscountPct", Percentage.Type}}),
    #"Reordered Columns" = Table.ReorderColumns(#"Changed Type",{"OrderDate",
 "CustomerKey", "SalesOrderNumber", "SalesOrderLineNumber", "OrderQuantity",
"UnitPrice",
 "UnitPriceDiscountPct", "DiscountAmount", "SalesAmount", "TaxAmt", "Freight"}),
    #"Inserted Year" = Table.AddColumn(#"Reordered Columns", "Year", each
 Date.Year([OrderDate]), type number),
    #"Inserted Month" = Table.AddColumn(#"Inserted Year", "Month", each
 Date.Month([OrderDate]), type number),
    #"Inserted Month Name" = Table.AddColumn(#"Inserted Month", "Month Name", each
 Date.MonthName([OrderDate]), type text),
    #"Changed Type1" = Table.TransformColumnTypes(#"Inserted Month Name",{{"Year",
 Int64.Type}, {"Month", Int64.Type}}),
    #"Reordered Columns1" = Table.ReorderColumns(#"Changed Type1",{"Year", "Month",
 "Month Name", "OrderDate", "CustomerKey", "SalesOrderNumber", "SalesOrderLineNumber",
 "OrderQuantity", "UnitPrice", "UnitPriceDiscountPct", "DiscountAmount", "SalesAmount",
 "TaxAmt", "Freight"}),
    #"Renamed Columns" = Table.RenameColumns(#"Reordered Columns1",{{"Month", "Month
 Number"}, {"Month Name", "Full Month Name"}})
in
    #"Renamed Columns"
```

Now, look at performing some transformations on the DimCustomer table. In the next steps you will:

- Remove unneeded columns.
- Merge columns.
- Extract values from existing columns.

First, remove columns that are not needed for analysis. With the DimCustomer query selected in the Queries pane, select the following columns by highlighting the first one, pressing the Control key, and then highlighting the remaining columns. Alternatively, you can make changes column by column (the resulting code will be the same). Either way, once you have one column (or all columns) chosen, you can right-click and select Remove Columns from the context menu or do the same from the Manage Columns group on the Home tab, which is shown in Figure 1-51. The resulting M code is shown in Listing 1-7.

- GeographyKey
- NameStyle
- TotalChildren

- NumberChildrenAtHome
- EnglishEducation
- SpanishEducation
- FrenchEducation
- EnglishOccupation
- SpanishOccupation
- FrenchOccupation
- HouseOwnerFlag
- NumberCarsOwned

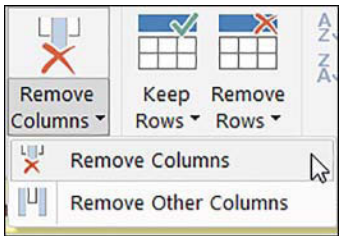

FIGURE 1-51 Remove Columns Options

LISTING 1-7 All M Code applied by removing columns from DimCustomer

```
= Table.RemoveColumns(dbo_DimCustomer,{"GeographyKey", "NameStyle", "TotalChildren",
 "NumberChildrenAtHome", "EnglishEducation", "SpanishEducation", "FrenchEducation",
 "EnglishOccupation", "SpanishOccupation", "FrenchOccupation", "HouseOwnerFlag",
 "NumberCarsOwned"})
```

NOTE **REMOVE OTHER COLUMNS**

If you have a table that has many columns and you only want to keep a few, you can use the Remove Other Columns transform, which keeps only the columns you have selected. The M function for this is Table.SelectColumns.

Now that you have only the columns you want, we will extract the username from the email address column and create a new column. This Extract command is used to extract characters from columns and can perform the operations shown in Figure 1-52.

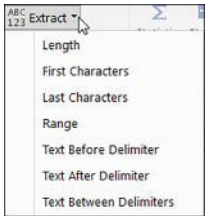

FIGURE 1-52 Extract options

Perform the following steps to extract the username from the email columns:

1. Ensure the EmailAddress column in the DimCustomer query is selected.
2. Click the Add Column tab and find the Extract command under the From Text grouping. From here, choose the Text Before Delimiter option.
3. Configure the values per Text Before Delimiter dialog in Figure 1-53. The result of this is to select all the characters before @ symbol, to start the scan from the beginning of the text field and to skip 0 delimiters. Click **OK** when complete.

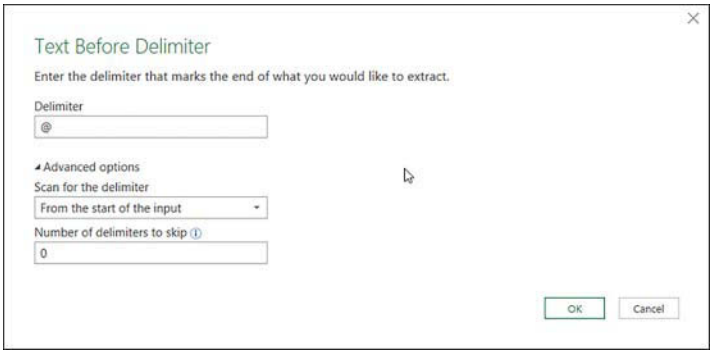

FIGURE 1-53 Extract Text Before Delimiter dialog

4. Find the new column that has been added to the end of the table and rename it to **Username**. The resulting M code for both the extract and rename is shown in Listing 1-8.

LISTING 1-8 M code for the Extract of username and rename of new column

```
= Table.AddColumn(#"Removed Columns", "Text Before Delimiter", each
Text.BeforeDelimiter([EmailAddress], "@", 0), type text)
= Table.RenameColumns(#"Inserted Text Before Delimiter",{{"Text Before Delimiter",
 "Username"}})
```

Lastly, merge the three name columns together using the Merge function by performing the following steps:

1. Ensure that DimCustomer is the selected query in the Query pane.
2. Highlight the FirstName column, press and hold the Control key and then select the MiddleName and LastName fields.
3. Click the Add Column tab and find the Merge command under the From Text grouping. This brings you to the Merge Columns dialog screen in Figure 1-54. Here you can choose the separator that you would like to put between the columns and can give the new column a name. Choose Space as the Separator and name the New Column Full Name (with a space). The resulting M code is shown in Listing 1-9.

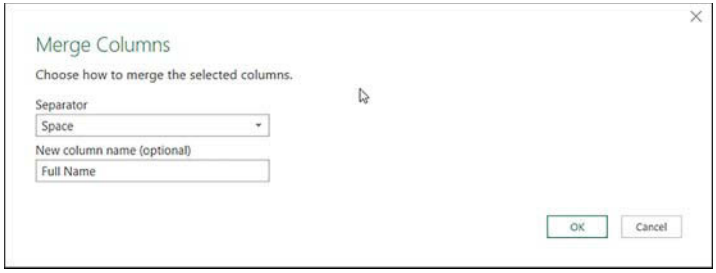

FIGURE 1-54 Merge Columns dialog

LISTING 1-9 M code for the Merging of columns

```
= Table.AddColumn(#"Renamed Columns", "Full Name", each
Text.Combine({Text.From([CustomerKey], "en-CA"), [FirstName], [MiddleName], [LastName]},
"
"), type text)
```

Listing 1-10 contains the resulting combined M code for all of the steps you applied to DimCustomer. You will find this by opening the Advanced Editor.

LISTING 1-10 M code for the Extract of username and Rename of new column

```
let
    Source = Sql.Databases("localhost"),
    AdventureWorksDW2016 = Source{[Name="AdventureWorksDW2016"]}[Data],
    dbo_DimCustomer = AdventureWorksDW2016{[Schema="dbo",Item="DimCustomer"]}[Data],
    #"Removed Columns" = Table.RemoveColumns(dbo_DimCustomer,{"GeographyKey",
"NameStyle",
 "TotalChildren", "NumberChildrenAtHome", "EnglishEducation", "SpanishEducation",
 "FrenchEducation", "EnglishOccupation", "SpanishOccupation", "FrenchOccupation",
 "HouseOwnerFlag", "NumberCarsOwned"}),
    #"Inserted Text Before Delimiter" = Table.AddColumn(#"Removed Columns", "Text Before
 Delimiter", each Text.BeforeDelimiter([EmailAddress], "@", 0), type text),
    #"Renamed Columns" = Table.RenameColumns(#"Inserted Text Before Delimiter",{{"Text
 Before Delimiter", "Username"}}),
    #"Inserted Merged Column" = Table.AddColumn(#"Renamed Columns", "Full Name", each
 Text.Combine({Text.From([CustomerKey], "en-CA"), [FirstName], [MiddleName],
[LastName]},
 " "), type text)
in
    #"Inserted Merged Column"
```

Now that you have completed what you need to transform both FactInternetSales and DimCustomer, it is time to do some housekeeping by performing the following steps:

1. Rename DimCustomer query to Customer.
2. Start organizing the queries that you have in the Queries pane. Create a new Group in the Query Pane named Fact Queries. Do this by right-clicking in an open space in the Query pane and then select New Group in the context menu per Figure 1-55. In the New Group dialog that opens, configure the following and click OK.

- **Name** Fact Queries.
- **Description** This is the group where you place fact queries. Any queries in this folder are moved to the Data Model when you load to it.

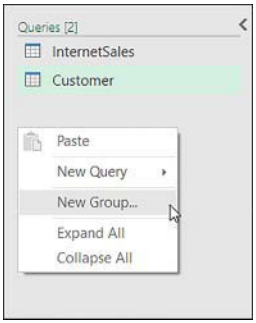

FIGURE 1-55 Create a New Group

3. Create a second group with the following values:
   - **Name** Dimension Queries.
   - **Description** This is the group where you place dimension queries. Any queries in this folder are moved to the Data Model when you load to it.
4. Now that you have these two groups created, the queries pane should look like Figure 1-56.

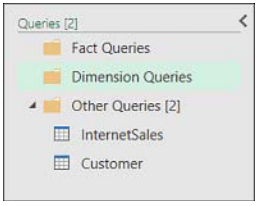

FIGURE 1-56 New Groups in the Queries Pane

5. What you need to do next is to move the InternetSales query to the Fact Queries group and move Customer to the Dimension Queries group. This can be done with a drag and drop, or by right-clicking on either query and selecting Move To Group and then selecting the proper group from the context menu. Note that you can also create a new group from this menu. Once you have done this, the Queries pane looks like Figure 1-57. When you create your first group, a new folder called Other Queries is generated by the Query Editor and this group cannot be deleted because it becomes the default group for new objects.

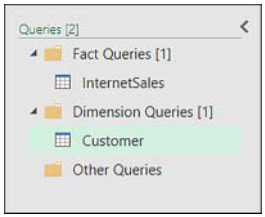

FIGURE 1-57 Queries moved to groups

6. As a final step to this transformation, load this data through to the Data Model. To do this, save your work in the Query Editor; in the Home tab, choose Close & Load. This brings you back into Excel.
7. In the Queries & Connections pane in Excel, right-click on InternetSales and choose Load To from the context menu, which brings you to Figure 1-58. Originally, you had asked to Only Create Connection. Now that you are done with Transforms, check the Add this to the data model check box. When done click OK and notice that the status of the Query in Excel has changed to show the rows loaded.

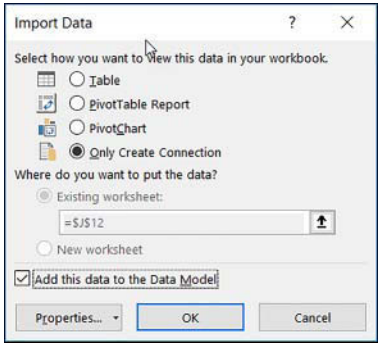

FIGURE 1-58 Import Data dialog

Another concept called Query Folding is worth pointing out since you are connecting to a relational database management system. Power Query attempts to translate the APPLIED STEPS into the data source's native language where supported. To see whether Query Folding takes place, right-click on any step in the Query Settings pane and choose View Native Query from the context menu. If the step cannot be selected because it is dimmed in the context menu, it means that Query Folding does not take place or has been disabled at some point in the steps. This is the case with InternetSales as it currently stands. Query folding ends after the Removed Other Columns step. Perform the following steps see the entire query run using query folding:

1. Open the Query Editor and select the InternetSales Query.
2. Remove the step that converts UnitPriceDiscountPct to a Percentage data type by clicking the x to the left of the step name or by right-clicking and selecting **Delete** from the context menu.

3. Remove the step that inserts the Month Name and all the step after it by right-clicking the Inserted Month Name step and choosing Delete Until End from the context menu. Now right-click on the last step and highlight the View Native Query. Your screen should now look like Figure 1-59.

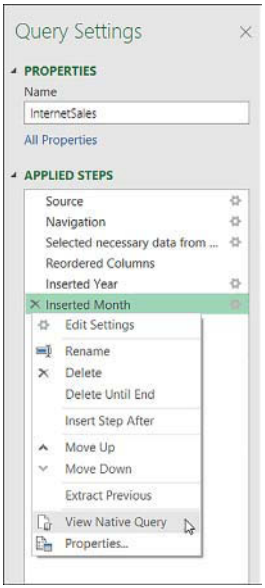

FIGURE 1-59 View Native Query

4. Now Click on View Native Query to open the Native Query dialog (Figure 1-60).

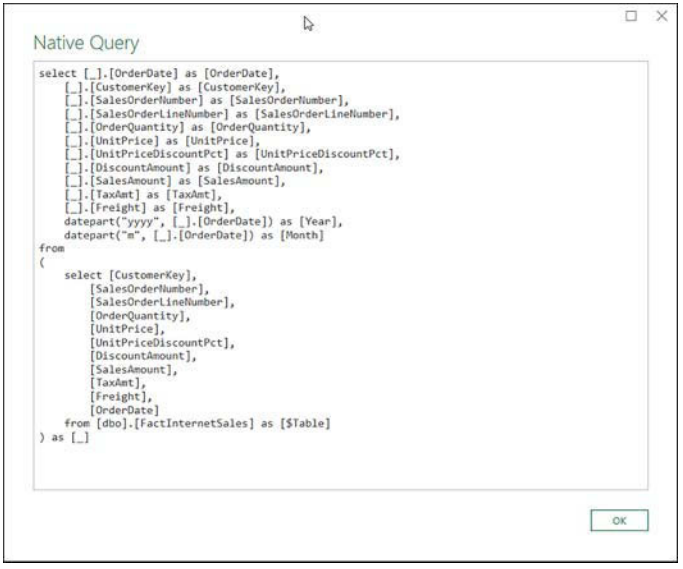

FIGURE 1-60 Native Query window

MORE INFO **QUERY FOLDING**

For more information on Query Folding, see Koen Verbeeck's article titled Query Folding in Power Query to Improve Performance: *https://www.mssqltips.com/sqlservertip/3635/query-folding-in-power-query-to-improve-performance/.*

5. Click **OK** and then close the Query Editor. When the Query Editor Keep your changes dialog box opens, click **Discard** to remove the changes that you made to get Query folding to work.

Advanced transformations

Now that you have done some basic transformations, it is time to look at some more advanced techniques. In this example, you use multiple text files to build out a star schema that is based on the Fact Internet Sales schema in AdventureWorksDW2016.

To start this exercise, you are going to combine files in a folder named Countries, which contains five Excel workbooks, and then append the combined query to a file with U.S. data. The Countries folder contains a spreadsheet for Internet Sales in each country with which Adventure Works does business. These files share the same structure, and all contain a column named Country, which is set to the country's name. However, the U.S. file does not have a column for country name and it is implied that each row of data is for the U.S. To solve this imbalance, add a custom column to the U.S. file.

Follow these steps to combine the files in the Other Countries folder using a folder import and the Combine Files transform.

1. Open a new blank Excel workbook.
2. Select **Data** tab > Get Data > **From File** > **From Folder**.
3. When the Folder Path dialog box opens, browse to the location where you downloaded the Book Sample file to, then navigate to **\Chapter 1\Advanced Example 1\Append Examples\Other Countries**.
4. A dialog box like the one in Figure 1-61 will appear, showing the five named .xlsx files you want to append. Scroll though and note the metadata that is extracted for each file. Once you are done, select Edit. As a note, you can choose Combine And Edit to open the Query Editor with a fully combined set. In this example, you are looking to show you the details as they are examinable.

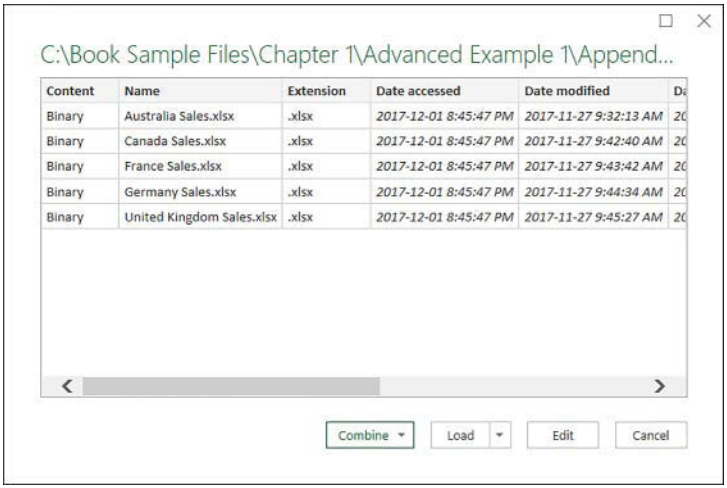

FIGURE 1-61 Files in Folder Source

5. Select the first column named Content as shown in Figure 1-62.

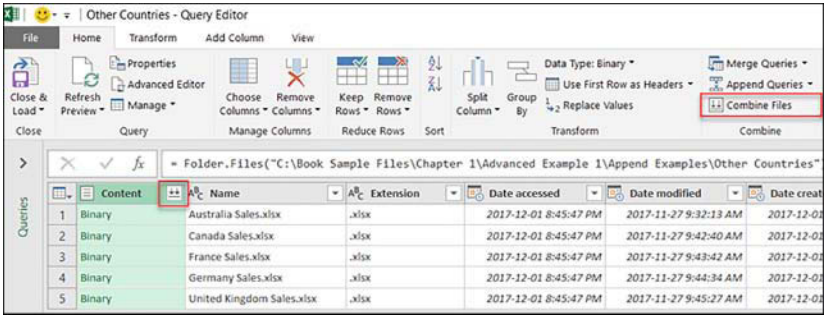

FIGURE 1-62 Combine Files Options

6. Click the icon on the right side of the column header as highlighted in Figure 1-62 OR from the Ribbon Choose the **Home** tab > **Combine Group** > **Combine Files**. Both options move you to the Combine Files dialog in Figure 1-63. In this dialog you make the following selections:

   - **Example File** You leave the First file, but you can select any of the files as an example via the drop-down. This provides the template from which all other file structures are compared.
   - **Sample File Parameter** Here you choose which object you want to load from the spreadsheet. Select tblSales.
   - **Skip Files With Errors** Leave this unchecked. If checked, files that do not match will be skipped.

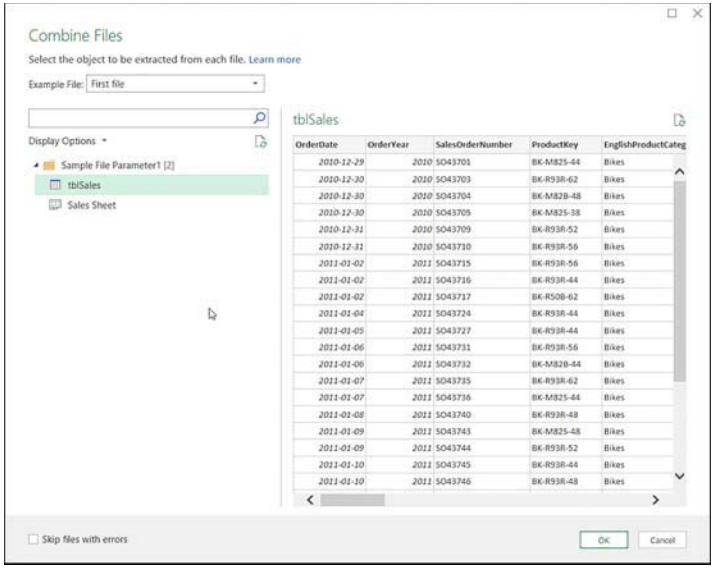

FIGURE 1-63 Combine Files dialog

7. Once you are done, click OK, which completes the process of combining the files into a table called Other Countries. Several other objects are created in the Queries pane that are used to support loading and combining the files from the folder source.

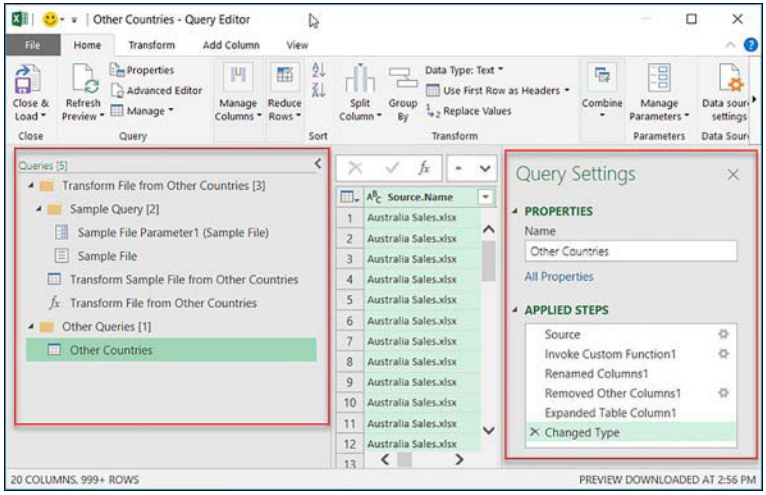

FIGURE 1-64 Results of File combine

8. Remove the column named Source.Name.

Now that you have combined the data for other countries' Internet sales, it is time to Append the U.S. and Other Country queries together with it so that you have the complete

picture of Internet Sales across all countries. Since the U.S. file is missing the Country Column, which is contained in the Other Countries data set, you need to prepare that data set first.

For Append to work as expected, the queries must have the same number of columns, with the same names and same data types within each column. If the columns in a source query are different, Append still works, but it will create one new column for each new column in the queries. The source that does not have that column will be assigned a null value.

Perform the following steps to Add the column, rearrange the data set, and then Append the sets:

1. If you are in the Query Editor, from the **Home** tab > **New Query** group > click **New Source** > **File** > **Excel**.
2. When the Import Data dialog box opens, browse to the location where you downloaded the Book Sample file and navigate to **\Chapter 1\Advanced Example 1\Append ExamplesUnited States Sales.xslx** and click **Open** on the file.
3. In the Navigator, choose tblSales from the spreadsheet and click OK.
4. Rename the Query to US Sales from tblSales.
5. Next, you need to add a new column named EnglishCountryRegionName to the U.S. Sales query and default its value to **United States**. To do this, click the Add Column tab and choose Custom Column from the General grouping, as shown in Figure 1-65. Configure the values as shown in Figure 1-65 and then click OK. See Listing 1-11 for the M code that will be generated from this step.

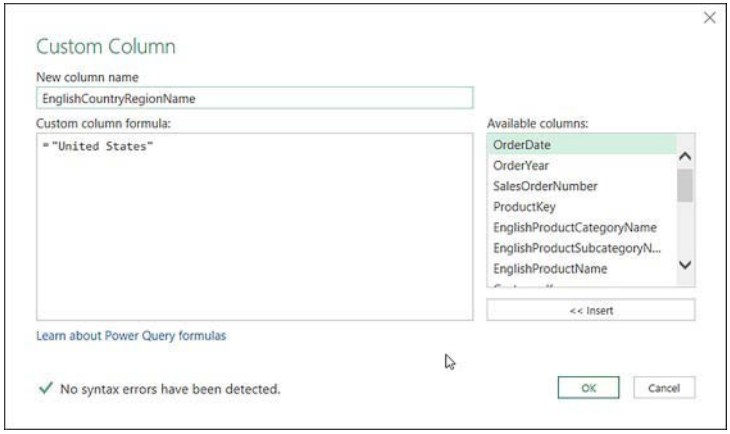

FIGURE 1-65 Custom Column dialog

LISTING 1-11 M code for the Custom column

```
= Table.AddColumn(#"Changed Type", "EnglishCountryRegionName ", each "United States")
```

6. Next change the data type from the **Any Type**, which is denoted with the ABC123 icon, to **Text.**

7. Our queries are now aligned, so now you can go ahead and append them. First, highlight the US Sales Query and from the Home tab, Choose Append from the Combine grouping. You can choose Append or Append Queries As New. Choose Append Queries As New to create a new query.
8. You should now see the Append dialog in Figure 1-65, which is where you choose what to append. Notice that you are only appending two files but using this method you can choose three or more. For this step, configure the values as shown in Figure 1-65. Once done, click **OK**.

FIGURE 1-66 Append dialog

9. You will now have a new query named Append1, which is the default name given to the newly created query. Rename this query to Internet Sales All Countries. Notice that this query only has one entry in Applied Steps.
10. Now look at the M code for the Append, which is in Listing 1-12. Notice that this code resides in the first step named Source of the newly created query.

LISTING 1-12 M code for the Append as New Query function

```
= Table.Combine({#"US Sales", #"Other Countries"})
```

You have completed the steps to Append the data together that is the fact table in the star schema that you are populating. Now you need to bring in some other dimensions. The first ones to bring in are the Product, Product Subcategory, and Product Category files. Once you have these loaded, use the Merge function to bring them into one query.

To begin the process of merging data, perform the following steps:

1. If you are in the Query Editor, click **Home** > **New Query** > **New Source** > **File** > **Excel**.
2. When the Import File dialog box opens, browse to the location where you downloaded the Book Sample file and navigate to **\Chapter 1\Advanced Example 1\Merge Examples** then bring each of the files in one by one and choose the table object in each file. Repeat steps 1 and 2 until the Product, Product Subcategory**,** and Product Category Excel files have been imported.
3. Rename each of the queries as below:
   - tblProducts should be renamed to Products

- tblProductSubcategory should be renamed to ProductSubcategory
- Table1 to ProductCategory

4. Now, you need to merge the queries so that all the values in the hierarchy are in the Product table. For the first merge, select the Product query and then choose **Home** tab, **Combine Group** > **Merge Queries**.
5. Configure the values as shown in the Merge dialog in Figure 1-67. The Products query should show up at the top because it was the query you selected when you executed the Merge command.
6. Next, choose what table you want to merge to it by choosing the ProductSubcategory table from the drop-down box. Once this has been selected, you need to tell Power Query which fields you want to merge.
7. In this example, you only join the ProductSubcategoryKey in each table by highlighting each column. Note that it is possible to use multiple fields to merge the columns. In that case, the order in which you choose columns matters.
8. Lastly, you need to choose a Join Kind. In this example, you choose Left Outer because you want all rows in the first query, regardless as to whether a match exists in the right-hand (bottom) query. However, even with this configuration, notice that you have an error at the bottom of the screen which says that you need to Select columns of the same type to continue. What happened? To find out, cancel out of this window for now so you can correct the issue.

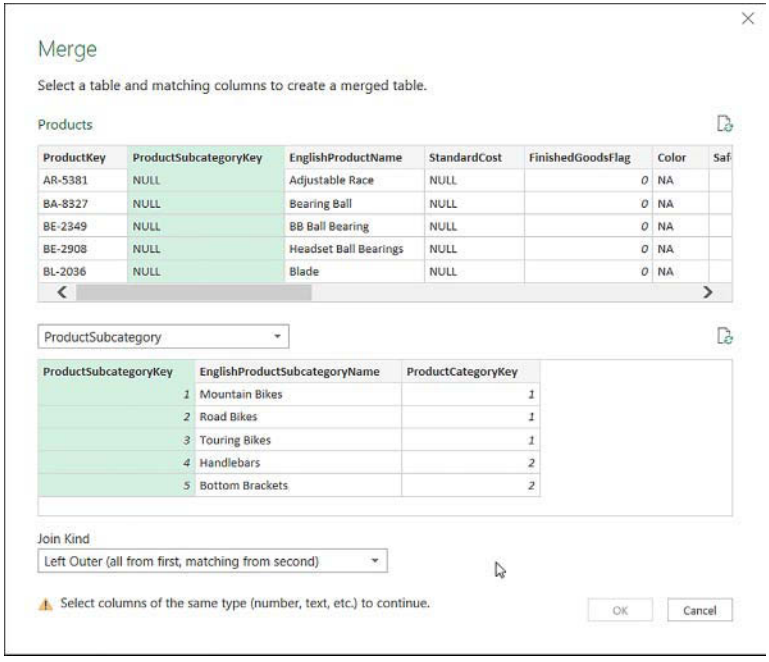

FIGURE 1-67 Merge dialog

EXAM TIP

Be sure to understand how to Merge queries well. Also, be familiar with the available Join Kinds and how each works. Join kinds are a classic exam question across many technologies.

9. Upon inspecting the queries, you'll notice that the ProductSubcategoryKey in the Product table has been defaulted to the Text data type by Power Query, and the ProductSubcategoryKey in the ProductSubcategory query value is an integer. To fix this make the following change:
   A. Make the ProductSubcategoryKey in the Product Query a Whole Number. When making this change, you might see the warning message in Figure 1-68. Power Query is asking if it should just replace the conversion it made with this one, or if you intended on creating a separate step in Query Settings. Select **Replace Current**.

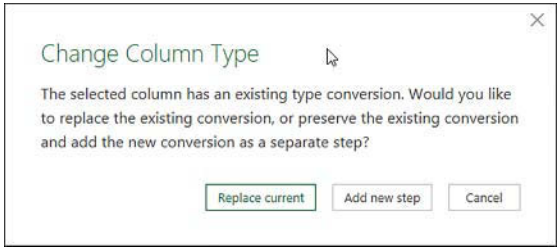

FIGURE 1-68 Change Column Type Question dialog

10. We can now repeat the instructions in Step 4. At the bottom of Figure 1-67, you should now see the message in Figure 1-69. Once complete press OK.

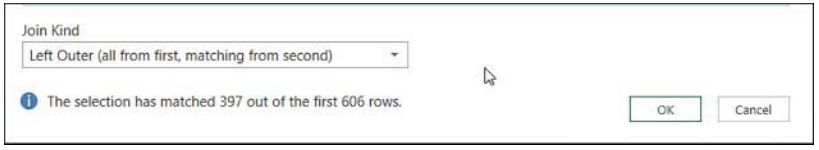

FIGURE 1-69 Join Match Information

11. The next step is to finish the Merge process by selecting which columns from the Product Subcategory table to keep in the Product table. To do this, select in the Product query and move to the far-right side where you see the last column represented as a Table shown in Figure 1-70. Also, see the M code in Listing 1-13.

FIGURE 1-70 Screenshot showing the table expander

LISTING 1-13 M code for the Merge Function

```
= Table.NestedJoin(#"Changed
 Type",{"ProductSubcategoryKey"},ProductSubcategory,{"ProductSubcategoryKey"},
"ProductSubcategory",JoinKind.LeftOuter)
```

12. Click on the table expand icon highlighted in Figure 1-70 and then you are presented with the screen in Figure 1-71. Configure the values as shown. ProductSubcategorykey would be redundant to keep, so uncheck it. You need the Product Category key to perform the next Merge. The resulting M code the table expanded is in Listing 1-14. More information on the configuration values is provided below.
    - **Expand** Use this if you want to bring in rows one for one according to the Join Type.
    - **Aggregate** Used to aggregate values before the merge.
    - **Use original column name as prefix** This prepends all new column names with the original table name.

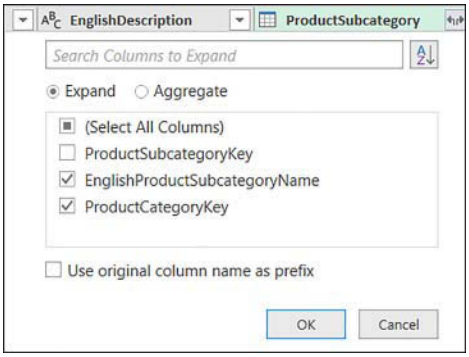

FIGURE 1-71 Expanded Column Selector

LISTING 1-14 M code for the Table Expand Function

```
= Table.ExpandTableColumn(#"Merged Queries", "ProductSubcategory",
 {"EnglishProductSubcategoryName", "ProductCategoryKey"},
 {"EnglishProductSubcategoryName", "ProductCategoryKey"})
```

13. Now that you have merged the two tables that you needed, you can now merge Product with Product Category. To do this, ensure that the Product query is highlighted and choose Merge.
14. Configure the Merge with the values in Figure 1-72 and then click OK.

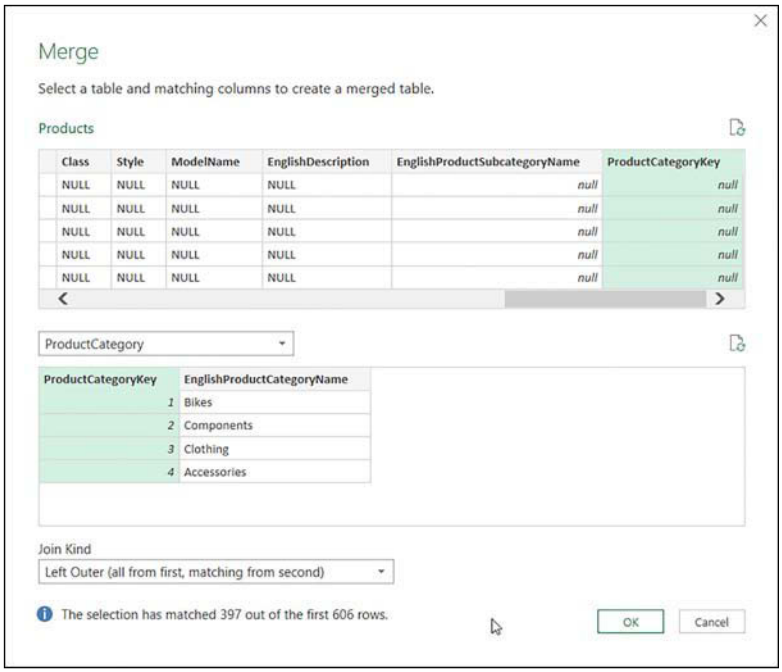

FIGURE 1-72 Merge dialog

15. Once complete, expand the new table out as shown in step 12, check the EnglishProductCategoryName, and uncheck the Use Original Column Name As Prefix option as in Figure 1-73. Click OK when complete.

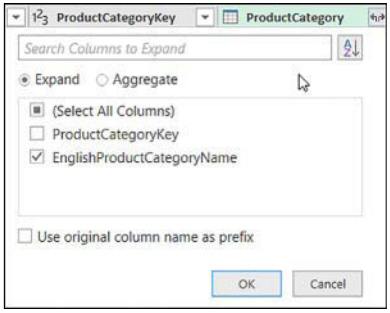

FIGURE 1-73 Expanded Column Selector

16. Now, you have all columns that you need in the Product table that enables a traditional Star Schema Product Dimension versus the snowflake that you started with. Remove columns that you do not need by using the Choose Column command. Keep the following columns and then put them in the order below using Move:

- ProductKey
- EnglishProductName

- EnglishProductSubcategoryName
- EnglishProductCategoryName
- Color
- ListPrice
- Size
- SizeRange
- Weight

Now that you have cleaned up the table, you have two tasks left to make the table consumer-ready. Replace the null values in EnglishProductSubcategoryName and the EnglishProductCategoryName columns with Text value values. Replace each with Undefined Subcategory and Undefined Category, respectively.

1. Ensure that the Products query is selected in the Queries pane of the Query Editor.
2. To replace the values, right-click on the EnglishProductSubcategoryName column and choose Replace Values from the context menu to open the Replace Values dialog in Figure 1-74. In this figure, you are replacing the null values in EnglishProductSubcategoryName with the text value Undefined Subcategory. This makes the end-user reporting easier to understand. Take note that you have two additional advanced options available that you can specify to use. Once done, click **OK**.

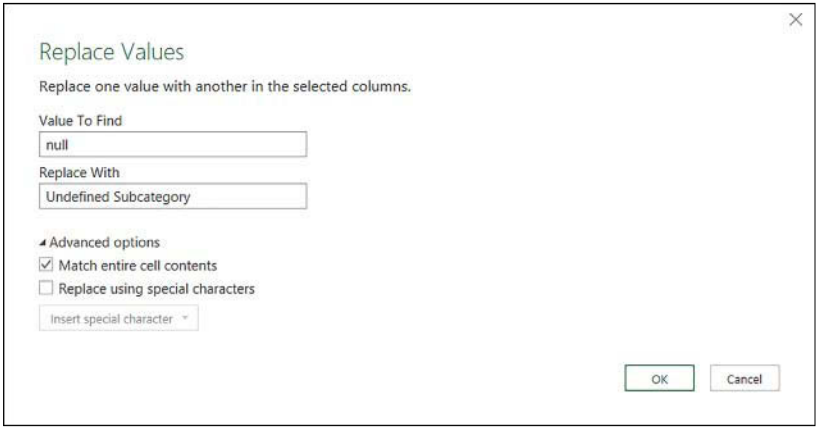

FIGURE 1-74 Replace Values dialog

LISTING 1-15 M code for the Replace Values Function

```
= Table.ReplaceValue(#"Reordered Columns",null,"Undefined
 Subcategory",Replacer.ReplaceValue,{"EnglishProductSubcategoryName"})
```

3. Repeat Step 2 for the EnglishProductCategoryName column and replace null with Undefined Category.

4. Now get rid of any duplicate rows in the Products query. To do this, click the ProductKey column then right-click it to get the context menu and choose Remove Duplicates.
5. You now have a well-formed product dimension for your users to consume. At this point it is worth looking at the Query Dependencies that are forming between the sources you have imported and merged together. In the **View** tab > **Dependencies** group > click **Query Dependencies** to see the screen in Figure 1-75. You can see the sources and how they feed into queries and then how the queries merge. You can also see the load state, and if you had any query metadata defined it would show up in the diagram as well. Click Close to close the diagram.

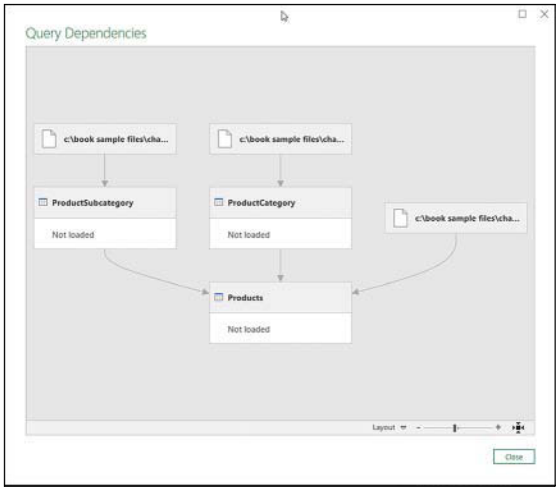

FIGURE 1-75 Query Dependencies

6. Try to delete a query that has a dependency to see what happens. Highlight the Product-Subcategory query, right-click, and select Delete from the context menu. You should see the dialog shown in Figure 1-76. Click Close, and you can continue to bring in additional queries for your model.

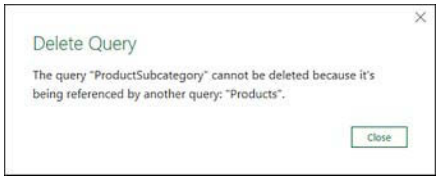

FIGURE 1-76 Delete Query warning

So far in this example, you have brought in Internet Sales data by combining and merging data from files and you have merged the product tables into one table. To help support our analysis, the business has asked you to also bring in Customer and Customer demographic information. Follow these steps to prepare the Customer query:

1. If you are in the Query Editor, click **Home** tab > **New Query** group > **New Source** > **File** > **Excel**.
2. When the Import Data dialog box opens, browse to the location where you downloaded the Book Sample file and navigate to **\Chapter 1\Advanced Example 1\Customers.xlsx** and click **Import**.
3. In the Navigator dialog, choose the tblCustomers object and then Click **OK**.
4. In the Query Editor, rename the newly imported query to Customers.
5. Now you want to bring in some additional Demographic information to the customer query from the AdventureWorks database. Connect to the SQL Server where your AdventureWorks2016 is located and configure the SQL Server Database connection as in Figure 1-77. Note that since we are writing a SQL query, we must supply a Database name. The SQL statement is shown in Listing 1-16. Once done Click OK.
6. Click OK on the next screen, which is a preview of the data that will be returned query. This will bring you back to the Query Editor.

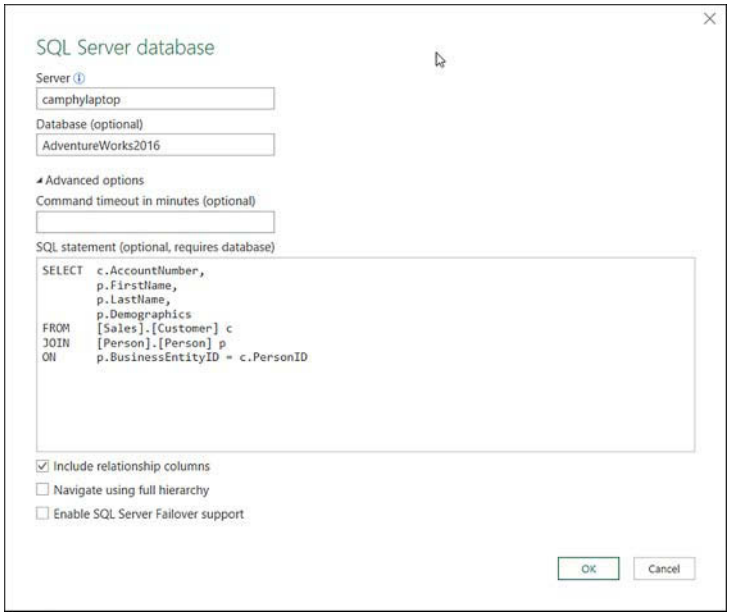

FIGURE 1-77 SQL Server Configuration dialog

LISTING 1-16 SQL Code to obtain the extended demographics information

```
SELECT    c.AccountNumber,
     p.FirstName,
     p.LastName,
     p.Demographics
FROM    [Sales].[Customer] c
```

```
JOIN    [Person].[Person] p
ON    p.BusinessEntityID = c.PersonIDRename the new query to CustomerDemographics
```

7. In the Data preview pane, highlight the demographics column in your new query. Notice that the data in the column is XML data. You want to parse this out by going to the **Transform** tab > **Text Column** group > Parse, and click the **XML** command. You can see that the XML is transformed into a table as in Figure 1-78. Also, the M code is shown in Listing 1-17.

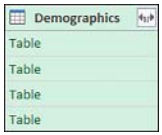

FIGURE 1-78 XML Transformed to Table dialog

LISTING 1-17 SQL Code to obtain the extended demographics information

```
= Table.TransformColumns(Source,{{"Demographics", Xml.Tables}})
```

8. Click the Expand table icon in Figure 1-78 to display the attributes that are in the XML string. You might only see a few values in the dialog, so it might be necessary to click Load More to see all attributes.
9. In the dialog, choose the TotalPurchaseYTD column and do not check Use original column name as prefix. Click OK when you are done. Listing 1-18 shows the M code that is generated by this step.

LISTING 1-18 Table Expand XML code

```
= Table.ExpandTableColumn(#"Parsed XML", "Demographics", {"TotalPurchaseYTD"},
{"TotalPurchaseYTD"})
```

10. Change the data type to Currency.
11. Next, Merge this piece of data to the Customers query using the steps described earlier in the Merge section. Highlight on the Customers query and choose **Home** tab, **Combine** group, and click **Merge Queries**.
12. In the Merge dialog box, ensure that Customers is the top table and then choose CustomerDemographics as the second table. Choose Left Outer Join Between CustomerKey in the Customers Query and AccountKey in the CustomerDemographics Query. Click **OK**.
13. Once you do this, you may be presented with Figure 1-79, which is asking you to configure Privacy levels between the sources. For both, choose **Private** and Click **Save**.

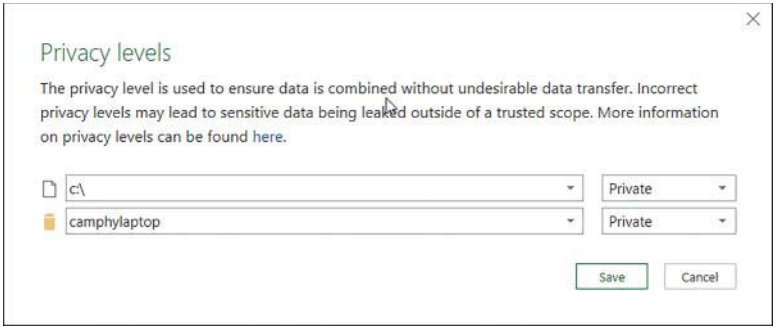

FIGURE 1-79 Privacy level settings dialog

14. In the CustomerDemographics column at the end of the Customers query, expand the table and choose the TotalPurchaseYTD column, and deselect Use original column name as prefix. Click **OK** when you are done.
15. We now have the TotalPurchaseYTD demographic information in the Customers query.

The last table that you need to bring into the model is Sales Territories. As you can see, this table contains a hierarchy that is embedded into one field. Take this column and use the Split function to create fields for each level of the hierarchy, then capitalize the top level of the hierarchy per a business requirement.

1. If you are in the Query Editor, Click **Home** > **New Query** > **New Source** > **File** > **Excel**.
2. When the Import Data dialog box opens, browse to the location where you downloaded the Book Sample file and navigate to **\Chapter 1\Advanced Example 1\Sales Territories**.xlsx and click **Import**.
3. In the Navigator dialog, choose the tblSalesTerritories object and then Click **OK** to bring it into the Query Editor.
4. Rename the query to SalesTerritories.
5. Highlight on the SalesTerritoryRollup column and then in the Transform tab, Text Column grouping click Split Column by Delimiter. Note that you have the option to also Split by characters if your field had that requirement. If you use the Split function from the Transform tab, the original column is replaced when you are complete. Configure the following options and Click OK when don2e. Listing 1-19 contains the M code for the Split.
   - **Select Or Enter Delimiter** You have the option of choose common delimiters such as columns, tabs, and spaces or entering a Custom delimiter like the one in this example. Type the pipe symbol | as your custom delimiter.
   - **Split At** You can choose Left-Most Delimiter, Right-Most Delimiter, or Each Occurrence Of The Delimiter. Select Each Occurrence Of The Delimiter.
   - **Split Into** Your options are Columns or Rows; choose Columns.

- **Number of columns to split into** You can choose how many columns that you want your data Split into. In our example, type **3**.
- **Quote Character** Choose the quote character to detect. Leave the default setting.
- **Split using special characters** Available if your Split requires special characters.

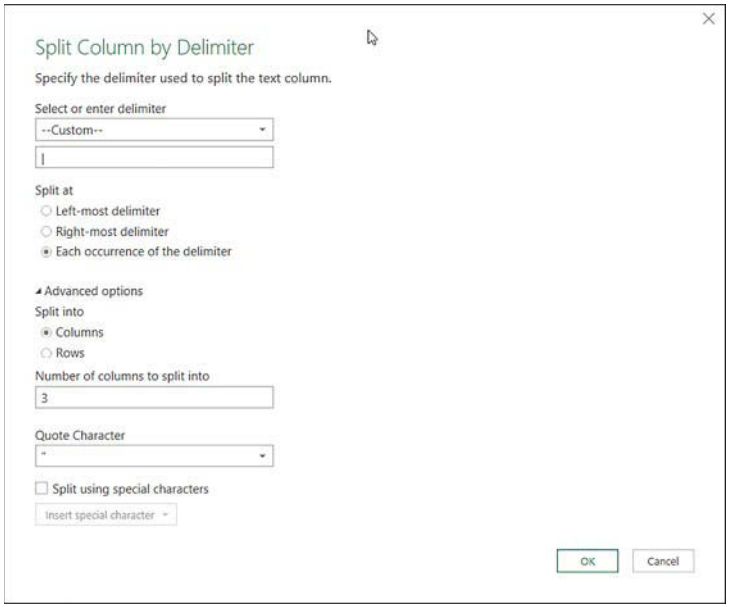

FIGURE 1-80 Split Column by Delimiter dialog

LISTING 1-19 Split M code

```
= Table.SplitColumn(#"Changed Type", "SalesTerritoryRollup",
 Splitter.SplitTextByDelimiter("|", QuoteStyle.Csv), {"SalesTerritoryRollup.1",
 "SalesTerritoryRollup.2", "SalesTerritoryRollup.3"})
```

6. You should now see **three** columns in the Data preview pane. Rename each as described below:
   - **SalesTerritoryRollup.1** to Territory
   - **SalesTerritoryRollup.2** to Country
   - **SalesTerritoryRollup.3** to Continent
7. Now capitalize the Continent column. Highlight the Continent field, right-click, and choose **Transform** > **Uppercase** from the context menu. The M code for this step is shown in Listing 1-20.

LISTING 1-20 UPPERCASE transform M code

```
= Table.TransformColumns(#"Renamed Columns",{{"Continent", Text.Upper}})
```

Now that you have a model that is complete and ready for consumption, it is time for you to load the data into to the Data Model.

1. First you should categorize the objects in Queries pane. In this example, you create a group for queries that are eventually output to the Data Model, and one for objects that are used only in the Transformation process that will never be exposed to the end users. Create the Groups and move the objects to the Groups as shown and in Figure 1-81.

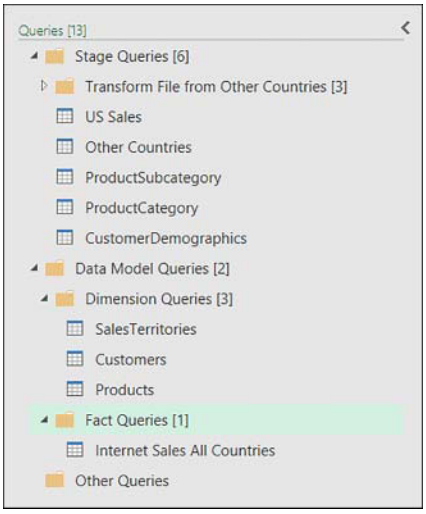

FIGURE 1-81 Queries pane with groupings

2. Once you have done this, close the Query Editor and then in Excel, ensure that the Queries and Connections pane is open as in Figure 1-82. On each of SalesTerritories > Customers > Products and Internet Sales All Countries, right-click and choose Load To. In the Import Data dialog box, check the Add This Data To The Data Model option. These tables are in the Excel Data Model, which is covered more in Skill 2.1 from Chapter 2.

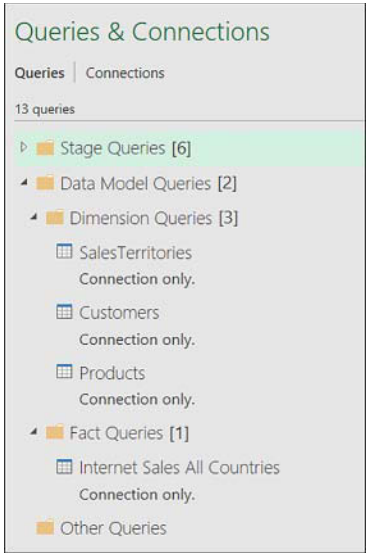

FIGURE 1-82 Queries & Connections

Some other items worth discussing that fall into the category of Query management are Data Source Settings, Manage Queries, and Recent Sources.

To Manage Data Source Settings, from the Query Editor go to the Home tab, Data source, and click Data Source Settings to open the dialog where you can choose **Manage Data Source Credentials** > **Privacy Levels** > and change the source as in Figure 1-83.

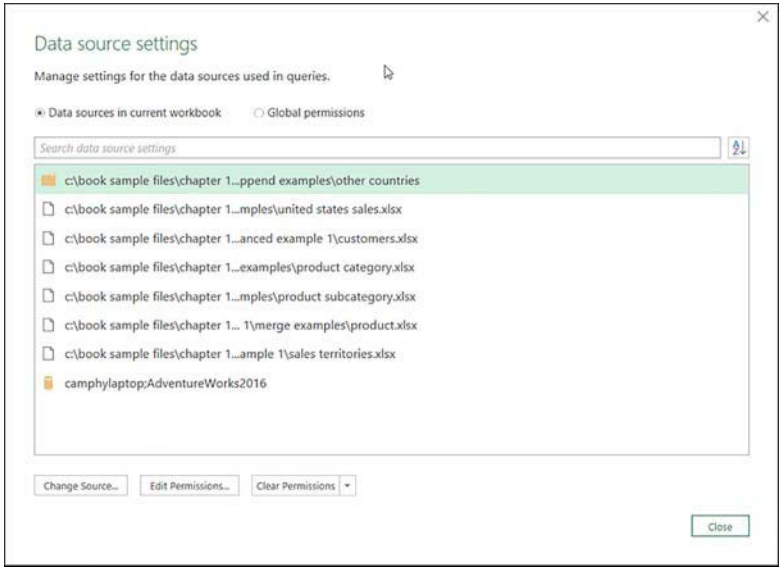

FIGURE 1-83 Data source settings

To change the source, highlight the first data source and click Change Source. When moving the source, you are presented with the same dialog box that you initially configured when connecting to your source. Remember that the structures after the initial configuration need to remain the same or your subsequent query steps will likely fail.

To manage Credentials and Privacy levels, Click **Edit Permissions** to be presented with the dialog shown in Figure 1-84.

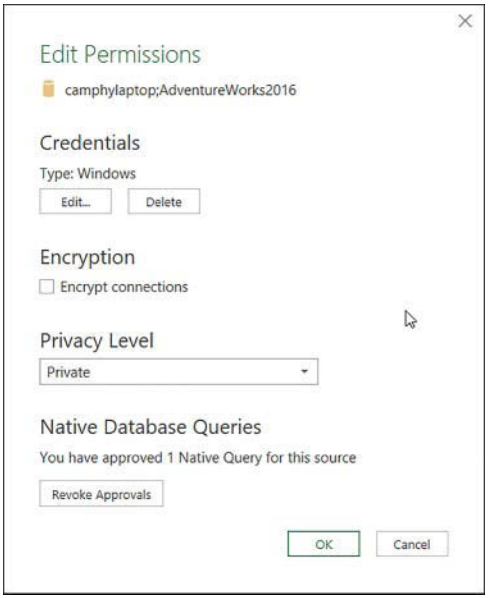

FIGURE 1-84 Edit Permissions dialog

To Manage queries, go to the **Query Editor** at **Home** > **Query** > **Manage**. You can perform the following three commands:

- **Delete** Does as advertised and deletes the object if it has no dependencies.
- **Reference** This option creates a new Query that references the query you want. The new query is created with a source step that calls the referenced query. This is useful when you have repetitive logic that needs to be applied to many queries that you want to write once and refer to many times.
- **Duplicate** This option creates a distinct copy of the query you want to copy and after the duplication process it's completely disconnected from the original.

Recent Sources can be managed in the Query Editor as well. To do this, click the Home tab, New Query, and click on Recent Sources, which shows the connection information for sources that you have connected to. It can help speed up the process of connecting to data sources that you have already connected to.

Apply business rules

Now, let's examine some of the functions that are available for applying business rules to your data. Often in operational systems, data is entered without following appropriate business rules. It is then up to the data-shaping process to apply those business rules so analysis can be performed. This typically involves using functions within your data-shaping steps that enable you to provide more complex logic.

Columns from examples

Columns from examples is a user-friendly method for building new columns from existing ones. Power Query attempts to recognize the pattern that you are entering, trying to extract text according to your pattern. It is modeled after Flash Fill in Excel. Try to use this to extract the username from the part of the email address before the @ symbol:

1. Go to the Customers query and highlight on the EmailAddress column.
2. Then go to **Add Column** tab, **General** group, **Columns From Examples** > **From Selection**, as shown in Figure 1-85. The last column initially starts blank. In that column, you begin to type the pattern that you want to find in the Email Address column and the detection process begins. If you type jon24 in the first row and then press Enter, Power Query determines that the Text before delimiter function should be applied to the column. If this is what you need, Click **OK** to complete.

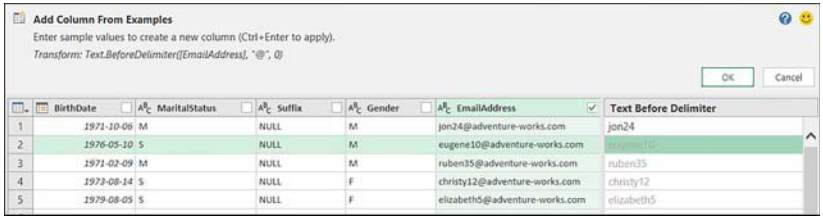

FIGURE 1-85 Add Column from example

Invoke custom function

To invoke a custom function, you first need to create one. You are going to create a function that takes the EngishCountryRegionName and passes out the ISO-ALPHA-3 value. Follow these steps to create the function and invoke it:

1. If you are in the Query Editor go to the **Home tab** > **New Query** > **New Source** > **Other Sources** > **Blank Query**. This creates a blank query in the Other Queries Group as shown in Figure 1-86.

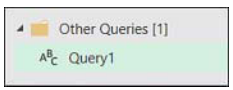

FIGURE 1-86 Screenshot of new Blank Query

2. Rename the new query from Query1 to ConvertCountryNameToISO-ALPHA-3.
3. With the query still selected in the Queries pane, go to the Home tab and click the Advanced Editor.
4. Using Widows File Explorer, navigate to where you downloaded the Book Sample files and **Open \Chapter 1\Function\ConvertCountryNameToISO-ALPHA-3.txt** using your favorite text editor.
5. In the Advanced Editor, delete any existing code, then cut and paste the code from the text file to the Advanced Editor.
6. With the **ConvertCountryNameToISO-ALPHA-3** function highlighted, test it by entering the parameter value of **Canada** in the **Input** text box. Click **Invoke** and the output should be **CAN**. Note that this is case-sensitive, so you need to handle this when invoking the function.
7. Now invoke the custom function by opening the **InternetSalesAllCountries** query and click **Add Column** tab > **General** group > **Invoke Custom Function**. Configure the dialog as shown in Figure 1-87 and below. Click **OK** and then verify that the new column is added.
   - **New column name** EnglishCountryRegionISO3Code
   - **Function query** ConvertCountryNameToISO-ALPHA-3
   - **Input (optional)** EnglishCountryRegionName

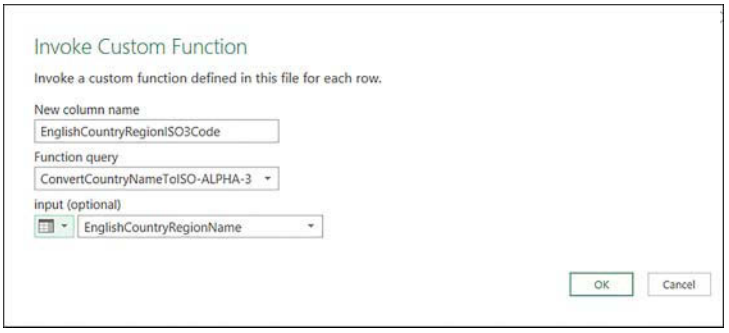

FIGURE 1-87 Invoke Custom Function dialog

Conditional Column

A Conditional Column operates as an IF statement. A new column is created in the query based on one or more conditional expressions you define within the Add Conditional Column dialog box.

In this example, create a new column based on the customer's yearly income. Place them into four categories: student, low income, middle income, and high income. This information can be used to help target marketing promotions as well as with customer profiling.

Follow these steps:

1. Select the **YearlyIncome** column in the **Customer** query**.**
2. Select **Add Column** tab > **General** group > **Conditional Column**.
3. In the **Add Conditional Column** dialog box, enter **Income Category** as the **New Column Name** and configure the values as in Figure 1-88.
4. Click **OK** when complete.

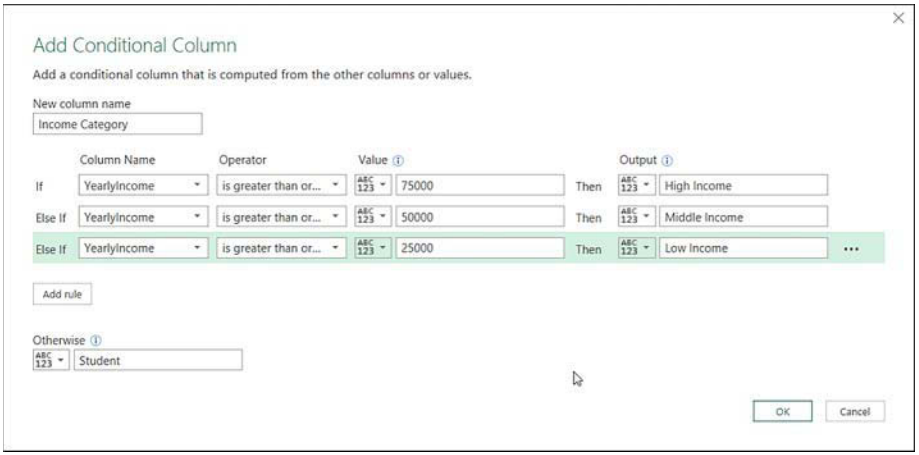

FIGURE 1-88 Add Conditional Column dialog

Index Column

An Index column can be added to a table for many purposes. There are three options for creating an Index column:

- **From 0** Creates an index that starts at 0 and increments by 1
- **From 1** Creates an index that starts at 1 and increments by 1
- **Custom** Allows you to choose the start and increment values

Change data format to support visualization

One of the jobs of the person performing data shaping is to shape the data in a manner that makes building visualizations as easy as possible. The target state of any data set is that is to set it up so that consumers can simply drag and drop values from the Excel data model into PivotCharts or PivotTables for analysis. If users find themselves having to perform transformations when they are building visualizations, these situations are normally candidates for pushing back into the data-shaping layer, especially if they are often-used fields for display or filtering.

Group By

A common task for visualizations is to provide pre-aggregated data. In this example, you take the InternetSalesAllCountries and create a new query that Aggregates Sales Amount by Year and Country. Follow these steps to perform this:

1. Highlight the **Internet Sale All Countries** query.
2. Right-click on it and select **Reference** from the context menu. This allows you to take advantage of the effort put into creating **Internet Sales All Countries**.
3. A new query is created with a single step that refers to Internet Sales All Countries.
4. Rename the query **Internet Sales By Year And Country**.
5. From the Home tab go to **Transform** group and choose **Group By**.
6. Configure the **Group By** as in Figure 1-89 and then click **OK**.

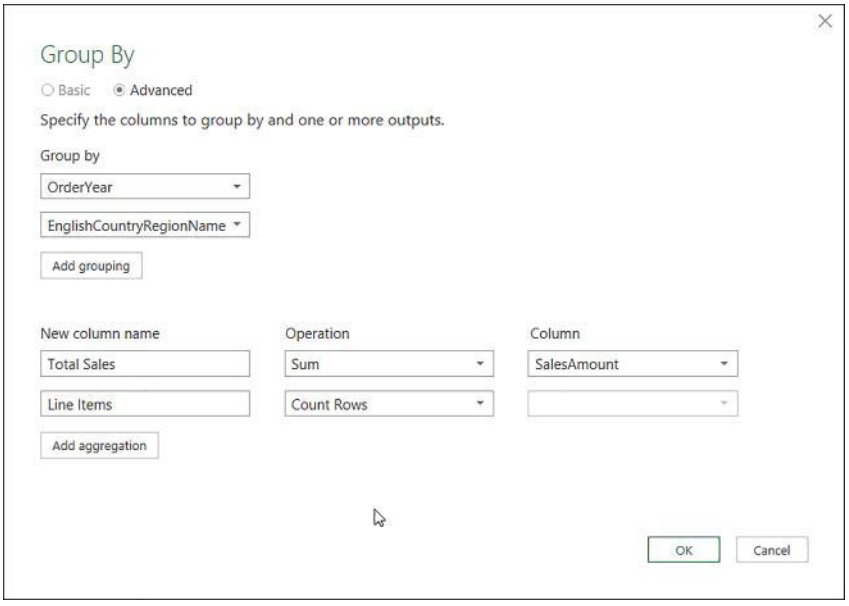

FIGURE 1-89 Group By dialog

Filter data

One of the often-overlooked simple performance tuning mechanisms in any data analysis tool is to only bring in the data that is necessary to support the analytics at hand. Ultimately you want to build flexible solutions that can handle the next question that is asked of it—building flexible and performant applications is your job as the person building these solutions for your organization.

Text Filters

If you have a text-based field highlighted, in the Data preview pane you can select the drop-down that presents you with the filtering criteria for the data type that has focus. Figure 1-90 has the drop-down on the field highlighted.

FIGURE 1-90 Filter selector location

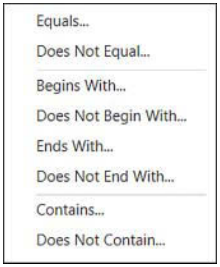

FIGURE 1-91 Text Filters

Numeric Filters

If you have a numeric-based field highlighted, from the Data preview pane you can select the drop-down that presents you with the filtering criteria for the data type that has the focus.

FIGURE 1-92 Numeric Filters

Date Filters

If you have a date-based field highlighted, from the Data preview pane you can select the drop-down that presents you with the filtering criteria for the data type that has focus. There are many available Date filters.

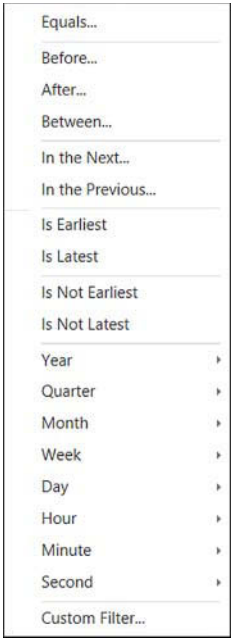

FIGURE 1-93 Date Filters

NEED MORE REVIEW? **DATA TYPE FILTERS**

For more review on the usage of data type filters, see *https://support.office.com/en-us/article/Filter-a-table-Power-Query-b5610630-f5bf-4ba4-9217-a628f9b89353*.

Parameters

One way to limit the data that is brought in for analysis is to take advantage of Parameters. In this example, you restrict the years that are brought into the Internet Sales All Countries query. Follow these steps to parameterize the value:

1. **Home** > **Parameters** > **Manage Parameters** > **New Parameter**.
2. Configure the values as in Figure 1-94 and then **Click OK**.

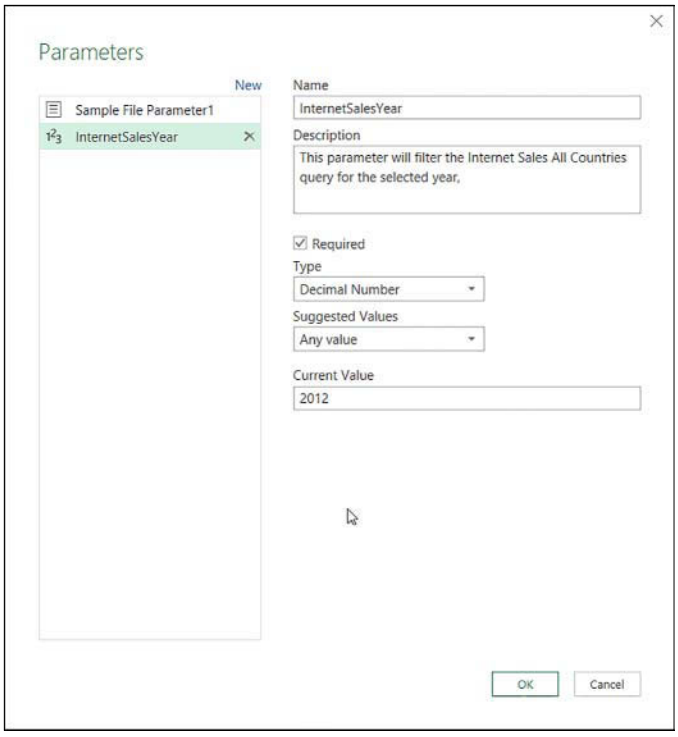

FIGURE 1-94 Manage Parameters dialog

3. You should now see a parameter in the Queries pane that has been created with the name InternetSalesYear and with "2012" in parentheses, which denotes the current parameter value.

FIGURE 1-95 Parameter with its current value

4. Now use the filter to restrict the OrderYear in Internet Sales All Countries. In the Data preview pane, click the drop-down box next to the OrderYear column title to view the context menu. Now choose **Number Filters** > **Equals** to get the Filter Rows dialog box shown in Figure 1-96. Choose Equals and ensure that you choose a parameter from the second drop-down box, then select the InternetSalesYear parameter. Click **OK** when done and observe that the Internet Sales query now only contains data for the year set in the parameter. The generated M code is shown in Listing 1-21.

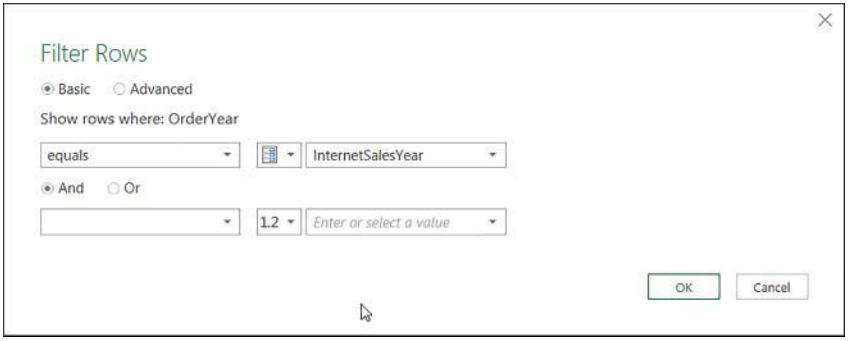

FIGURE 1-96 Filter Rows dialog

LISTING 1-21 Filter Rows with Parameter M code

```
= Table.SelectRows(#"Changed Type", each [OrderYear] = InternetSalesYear)
```

Now enhance the parameter by making the user choose from a list versus hardcoding values.

1. To make this modification, Click Manage Parameters and configure the Parameter values for InternetSalesYear as shown in Figure 1-97. Be sure that in the dialog box that you have the **InternetSalesYear** parameter highlighted on the left.

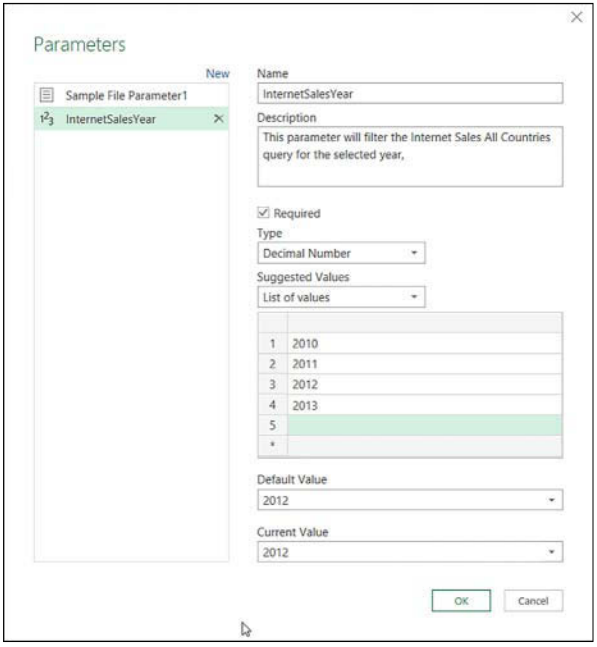

FIGURE 1-97 Manage Parameters dialog

2. Now the values for the parameter are restricted to the list of values you provided. To verify, observe the data in the **Internet Sales All Countries** query.

Lastly, make one more change that drives the parameter values from a dynamic list rather than a hardcoded one as you just did.

1. First, you need to create a list of values by making a list. To do this, duplicate the **Internet Sales All Countries** query. Highlight the **Internet Sales All Countries** query, right-click on it and select **Duplicate** from the context menu.
2. Rename the query to **ListValidYears**.
3. Remove the **Filtered Rows** step from **Applied Steps** in the **Internet Sales All Countries** query.
4. In the **ListValidYears** query, highlight the **OrderYear** column.
5. Then in the **Transform** tab > **Any Column** group > choose **Convert To List**.
6. Now you have a new tools tab named **List Tools**. From here, select **Transform** > **Manage Items** > **Remove Duplicates**, which gives you a list of years that are valid for that dataset.
7. Open the **Manage Parameters** dialog and change the **Suggested Values** to **Query** and the Query drop-down to **ListValidYears**.

Format data

Data often needs to be formatted for final presentation. The format functions can be found in the both the **Transform** and **Add Columns** tabs, in the **From Text** grouping.

- **Lowercase** Returns the lowercase of a text value.
- **Uppercase** Returns the uppercase of a text value.
- **Capitalize Each Word** Returns a text value with first letters of all words converted to uppercase.
- **Trim** Removes any occurrence of spaces at the beginning or end of a string.
- **Clean** Returns the original text value with non-printable characters removed.
- **Add Prefix** Adds a prefix to the text you are working with.
- **Add Suffix** Adds a suffix to the text you are working with.

Skill 1.3: Cleanse data

When performing analysis, it is important to first ensure that you are working with data that is fit for this purpose. This involves taking any necessary steps to improve the quality of data before using it. Data cleansing, as it is commonly known, is the process of detecting and correcting or removing data from a data set, before use, if it not fit for purpose. It is very common for data values to be missing, incomplete, or inaccurate.

It is often up to you as the analyst to make the determination as to when data does not fit your needs and to act. Fortunately, Power Query has a vast array of functions that allow you to clean data, manage incomplete data by replacing values, filling in missing values, keeping rows, removing rows and even invoking sophisticated functions that can be used to detect business rule violations within data fields.

In addition to managing data quality issues, Power Query can help to clean up data that is delivered to you in the form of reports that often need some cleanup prior to you getting at the data that is contained within them.

This section covers how to:

- Manage incomplete data
- Handle data received as a report

Manage incomplete data

In this example, extract competitor sales data from a table in an Excel spreadsheet that has missing data and an incorrect header. You also need to replace from data in one of the cells with text that makes more sense for the users when they are performing analysis as someone clearly made some personal comments in one of the fields.

Follow these steps to cleanse this data:

1. From within in Excel, navigate to **Data** > **Get & Transform Data** > **Get Data** > **From File** > **From Workbook**.
2. In the Import File dialog box, navigate to **\Chapter 1\Excel\Manage incomplete data.xlsx** and click **Open**.
3. In the Navigator, choose **tblCompetitorSales** and then click **Edit**. When you move to the Query Editor, your Data preview pane should look like Figure 1-98. Notice that the header is not correct and that Power Query was not able to determine data types for two of the columns as denoted by the Any data type icon **ABC123**.

	Column1	Column2	Column3
1	OrderYear	EnglishCountryRegionName	Total Sales
2	2010	Australia (look bob)	899120.54
3	null	Canada (look bob)	221852.74
4	null	France (look bob)	37399.89
5	null	United Kingdom (look bob)	18176.5532
6	null	United States (look bob)	44501.6946
7	2011	Australia (look bob)	215353508.9
8	null	Canada (look bob)	57157179.84

FIGURE 1-98 Data preview pane

4. The first activity to perform is to Promote Headers since the items in **Row 1** appear to be your headers. To do this, go to the **Transform** tab > **Use First Row As Headers**.
5. Change the Data Type for **OrderYear** to **Whole Number** and **Total Sales** to **Currency**.
6. Next, it appears the **Year** columns should repeat itself until the data in the rows moves to the next year. To do this, ensure the **OrderYear** has focus and go to the **Transform** tab, **Any Column** group, **Fill Down**. Notice that there is also an option to **Fill Up** should the use case arise.
7. Now let's get rid of the portion of the text that says (look bob) in the **EnglishCountryRegionName** field. Go to the **Transform** tab > **Any Column** group > **Replace Values**. When the dialog box opens, ensure to **Value To Find** is "(look bob)" without the quotes and the **Replace With** is an empty string. (Do not put anything in the box.)
8. Now we should ensure that the **EnglishCountryRegionName** has no blank characters at the beginning or end of each value. Highlight the column and go to the **Transform** tab > **Text Column** group > **Format** > **Trim**.
9. And finally Rename the query from **tblCompetitorSales** to **CompetitorSales**.
10. Once you are done, be sure to review the M code generated by these steps.

Handle data received as a report

Often when you are asked to perform analysis, it can be hard to find a data source to support it that comes in the traditional rows and columns format. Sometimes the best way to get the data is to have someone just run a report for you and then to perform the work of cleaning things up so that you can get at the data it contains.

Unpivot data

In this example, take a report that has been made available to you and perform the steps to make the data ready for analysis.

Follow these steps to cleanse this data:

1. From Excel, navigate to **Home** > **Get Data** > **From File** > **From Workbook**.
2. In the Import File dialog box, navigate to **\Chapter 1\Advanced Example 3\Pivot Examples\Product Category Sales By Year and Month Pivot Report.xlsx** and click **Open**.
3. In the object **Navigator,** choose **Sheet 2** and then click **Edit**. When you move to the Query Editor, your Data preview pane should look like Figure 1-99.

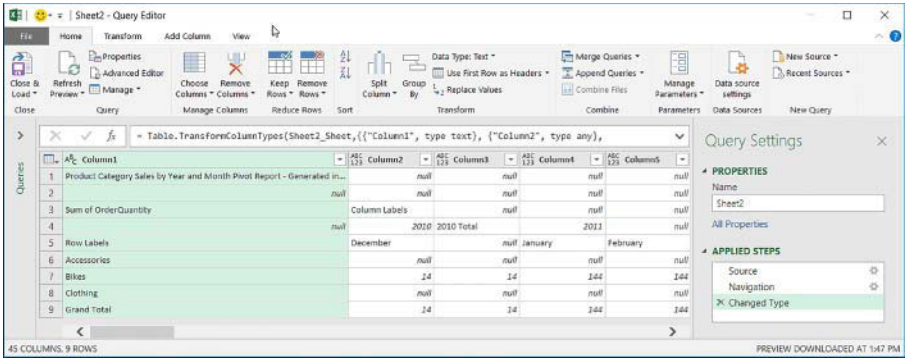

FIGURE 1-99 Data preview pane with initial data load

4. First, you need to remove the rows at the top of the report. To do this, go to the **Home** tab > **Reduce Rows** group **> Remove Top Rows.** When the Remove Top Rows dialog box opens, choose **2** as the number of rows.
5. Next, you need to Transpose the table. Click **Transform** tab > **Table** group > **Transpose.**
6. Now Remove the column named **Column1.**
7. Fill down **Column2 by** clicking the **Transform** tab > **Any Column** group > **Fill Down**.
8. Promote the first row to Headers since the items in Row 1 now appear to be your headers. To do this, choose the **Transform** tab > **Use First Row As Headers**.
9. Now, you need to remove the totals rows in **Column1** (notice that the Column1 name is now back, even though it is not the same column) by filtering. You can do this by manually selecting the values, but this would not be flexible going forward. To be more dynamic, you first need convert the data type of **Column1** to **Text** so that you can take advantage of a wider range of filtering options as the Any data type only has a few options for filtering.
10. You are now ready to filter the rows out. Right-click **Column1** and in the context menu, navigate to the **Text Filter** option, and in the sub-menu, choose **Does Not Contain**.
11. In the dialog box, type **Total**.
12. The goal of this step is to take the values in each of the categories and unpivot them into one row per Year, Month and Category. To do this, ensure that **Column1** and **Row Labels** are selected in the Data preview pane. Now select the **Transform** tab > **Any Column** group > **Unpivot Other Columns**.
13. Rename the columns and change the data types as below:
    - **Column1** Name=Year and Data Type=Whole Number
    - **Row Labels** Name=Months and Data Type=Text
    - **Attribute** Name=Product Category, Data Type=Text
    - **Values** Name=Sales, Data Type=Currency

14. Filter out the Grand Total column by clicking the drop-down box next to the **Product-Category** field name > select **Text Filters** > **Does Not Equal**. When the dialog box opens, type **Grand Total**.
15. Rename the query from **Sheet2** to **UnpivotExample**.
16. Save your work by going to **Home** tab > **Close** group > **Close & Load**.
17. In the Import Data dialog, choose to **Only Create the Connection** and check the **Add This Data To The Data Model** check box.

Pivot data

Now suppose that you want to Pivot a data set.

Follow these steps to Pivot your dataset:

1. In the Queries pane, click on the **UnpivotExample** query, then right-click and choose **Reference** from the context menu.
2. Rename the new query from **UnpivotExample (2)** to **PivotExample**.
3. The goal of this step is to move the Product Category values and move them into column headers with their associated values below. In the Data preview pane of the new query, highlight the **Product Category** field. Now click the **Transform** tab > **Any Column** group > **Pivot Column**.
4. In the dialog box that opens, choose **Sales** as the **Values Column** and select **Sum** as the Aggregate Value Function in the **Advanced** tab. Once you are done, Click **OK**.
5. In each of the three resulting columns, Replace the null with the number 0, since this is what this analysis requires.

	Year	Month	Bikes	Accessories	Clothing
10	2011	May	174	0	0
11	2011	November	208	0	0
12	2011	October	221	0	0
13	2011	September	185	0	0
14	2012	April	219	0	0
15	2012	August	294	0	0
16	2012	December	355	106	22
17	2012	February	260	0	0

FIGURE 1-100 Data preview Snippet from end of last exercise

Thought experiment

In this thought experiment, apply what you've learned in this chapter. Each Thought experiment is directly followed by a Thought experiment answer.

1. When loading data into your model you use the Query Editor and you select Load To. You are asked to select how you would like to view this data in your worksheet. Which selection will not load the data to a traditional Excel object?

   A. Create Connection Only
   B. Table
   C. PivotTable Report
   D. PivotChart

2. The marketing department has asked you to perform some analysis of 2017 sales. They have given you access to their Azure SQL Database which contains 1.2 million rows. What options do you have for importing this data into Excel knowing that they want to analyze all the data?
3. In which version of Excel was the Excel Data Model introduced? What was it originally called?
4. How many data models can a workbook have?
5. Name the four valid text file formats that you can find in Get & Transform.
6. If you want to be able to choose related tables when connecting to a SQL Server Database, what two things need to be in place for the related tables functionality work?
7. When you create a connection to an external data source such as an Analysis Services Database, where is connection information stored?
   A. Office XML (.oml File)
   B. Office Data Connection File (.odc)
   C. ODATA File
   D. ODBC
8. Which Data Source in Get & Transform Data allows you to connect to live data?
   A. Analysis Service Tabular
   B. Analysis Services Multi-Dimensional
   C. Text Files
   D. Web Page
9. When ingesting files using the From Folder or From SharePoint Folder, what characters must the files share?
10. What is Query Folding?
11. Which two ways can be used to reduce the number of rows in a data set?
   A. Filter Functions
   B. Remove Columns
   C. Parameters
   D. Extract
12. What is the difference between the Extract Power Query Function that exists on the Transform tab versus the Add Column tab?

Thought experiment answers

1. Answer **A**: Create Connection Only creates a connection to the source only. Data is not loaded to a Table, PivotTable Report, or PivotChart, but you do have the option of loading the data to the Excel Data Model.
2. The question said that the data is imported into Excel. Given this, the Excel Data Model is your only option as traditional Excel spreadsheets have a 1,048,576-row limit.
3. The Data Model was first introduced into Excel 2010 as PowerPivot. The two terms are still used interchangeably.
4. Each Excel workbook may only have one data model
5. Text, CSV, JSON, and XML.
6. First, the database must have referential integrity in place, and second you need to include relationship columns in the SQL Server Database connection configuration window.
7. Answer **B**: Office Data Connection File.
8. Answers **A** and **B**: Analysis Service Tabular and Dimensional.
9. The files all must share the same structure.
10. Query folding is when Power Query converts its transformations in the Native query language of the data source.
11. Answers **A** and **C**: Filter Functions and Parameters. Remove columns does not remove rows, and Extract is used to remove portions of text fields.
12. Like many functions, it is important to know the context in which you are using it. If you have chosen a column as the subject of an Extract, know that if you do this from the Add Column tab that a new column with the extracted value is created. If you do this from the Transform tab the column value is replaced with the newly extracted value.

Chapter summary

- A Data Model is used to integrate data from multiple sources into one or more tables inside an Excel workbook. Tables are then related so that Data Models can be used to provide tabular data that can be used by PivotTables, PivotCharts, and Power View reports.
- When importing data using a SQL database, you can optionally choose to write a native query instead of selecting object by object. This can be convenient when complex queries have already been written that you can reuse.
- When importing data from text files, Power Query tries to determine the File Locale, Delimiters also try to detect data types for each column.

- Excel can connect to the following Azure sources which commonly store data that can be used in Analysis, Azure SQL Server, SQL Data Warehouse, Data Lake Store, HDInsight, and BLOB and Table Storage.
- Privacy Levels are used to Isolate Data sets from each other.
- M is a functional case sensitive language that used to manage the data shaping process in Power Query. It is generated by the GUI are as you perform transformation and cleansing tasks. M code can be viewed in the Formula Bar or through the Advanced Editor.
- When creating Applied Steps, you can add step name of your choosing and a description through the Step Properties dialog.
- You can create columns from example tries to identify patterns in your data as supplied by you to accelerate the M code writing process.
- You can create custom functions in M and then call them row by row using the Invoke Custom Function call.
- Data for reporting can be formatted using Format functions such as Uppercase, Lowercase, Capitalize Each Word, Add Prefix/Suffix, Trim, and Clean.
- The Extract functions allow you to remove certain portions of text fields.
- If a column contains XML or JSON, you can use Parse XML or JSON to expand values.
- Query Dependencies is useful for viewing how your queries relate to each other. This is especially important as solutions grow.
- The Folder data source is useful way to combine one or many files from a directory if they all share the same structures.
- The Group By function can be used to aggregate data sets for use in the data shaping process or for consumption in the data model.
- Filters are used to remove entire rows that do not meet a specified criterion. Filters are specific to each data type in M. The most commonly used Filters are for Text, Number, and Date.
- Parameters are useful for enabling filtering of data in Power Query. Parameters can come from ad-hoc manual entry, a list of values as set-up in parameter management, or they can come from a query.
- Power Query has mechanisms to easily Pivot and Unpivot data sets.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z