LEARNING MADE EASY

2nd Edition

Microsoft® Excel® Power Pivot & Power Query

for dummies®

A Wiley Brand

Import and manage multiple data sources directly in Excel

Automate data cleanup and transformation tasks

Create interactive multidimensional reporting models

Michael Alexander

Microsoft® Excel® Power Pivot & Power Query

2nd Edition

by Michael Alexander

for dummies®
A Wiley Brand

Microsoft® Excel® Power Pivot & Power Query For Dummies®, 2nd Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Copyright © 2022 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Trademarks: Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc., and may not be used without written permission. Microsoft and Excel are registered trademarks of Microsoft Corporation in the United States and other countries. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc., is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: WHILE THE PUBLISHER AND AUTHORS HAVE USED THEIR BEST EFFORTS IN PREPARING THIS WORK, THEY MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES REPRESENTATIVES, WRITTEN SALES MATERIALS OR PROMOTIONAL STATEMENTS FOR THIS WORK. THE FACT THAT AN ORGANIZATION, WEBSITE, OR PRODUCT IS REFERRED TO IN THIS WORK AS A CITATION AND/OR POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE PUBLISHER AND AUTHORS ENDORSE THE INFORMATION OR SERVICES THE ORGANIZATION, WEBSITE, OR PRODUCT MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING PROFESSIONAL SERVICES. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR YOUR SITUATION. YOU SHOULD CONSULT WITH A SPECIALIST WHERE APPROPRIATE. FURTHER, READERS SHOULD BE AWARE THAT WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ. NEITHER THE PUBLISHER NOR AUTHORS SHALL BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER COMMERCIAL DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit https://hub.wiley.com/community/support/dummies.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2022930084

ISBN 978-1-119-84448-8 (pbk); ISBN 978-1-119-84449-5 (ebk); ISBN 978-1-119-84450-1 (ebk)

Contents at a Glance

Introduction 1

Part 1: Supercharged Reporting with Power Pivot 5

CHAPTER 1: Thinking Like a Database 7
CHAPTER 2: Introducing Power Pivot 17
CHAPTER 3: The Pivotal Pivot Table 29
CHAPTER 4: Using External Data with Power Pivot 63
CHAPTER 5: Working Directly with the Internal Data Model 89
CHAPTER 6: Adding Formulas to Power Pivot 103
CHAPTER 7: Diving into DAX 121

Part 2: Wrangling Data with Power Query 141

CHAPTER 8: Introducing Power Query 143
CHAPTER 9: Power Query Connection Types 159
CHAPTER 10: Transforming Your Way to Better Data 175
CHAPTER 11: Making Queries Work Together 207
CHAPTER 12: Extending Power Query with Custom Functions 225

Part 3: The Part of Tens 243

CHAPTER 13: Ten Ways to Improve Power Pivot Performance 245
CHAPTER 14: Ten Tips for Working with Power Query 253

Index 263

Table of Contents

INTRODUCTION 1
About This Book 2
Foolish Assumptions 3
Icons Used in This Book 3
Beyond the Book 4
Where to Go from Here 4

PART 1: SUPERCHARGED REPORTING WITH POWER PIVOT 5

CHAPTER 1: Thinking Like a Database 7
Exploring the Limits of Excel and How Databases Help 7
Scalability 8
Transparency of analytical processes 9
Separation of data and presentation 10
Getting to Know Database Terminology 11
Databases 11
Tables 11
Records, fields, and values 12
Queries 13
Understanding Relationships 13

CHAPTER 2: Introducing Power Pivot 17
Understanding the Power Pivot Internal Data Model 18
Linking Excel Tables to Power Pivot 20
Preparing Excel tables 21
Adding Excel Tables to the data model 22
Creating relationships between Power Pivot tables 24
Managing existing relationships 26
Using the Power Pivot data model in reporting 27

CHAPTER 3: The Pivotal Pivot Table 29
Introducing the Pivot Table 30
Defining the Four Areas of a Pivot Table 30
Values area 30
Row area 31
Column area 31
Filter area 32
Creating Your First Pivot Table 33
Changing and rearranging a pivot table 36
Adding a report filter 37
Keeping the pivot table fresh 38

Customizing Pivot Table Reports 40
Changing the pivot table layout. 40
Customizing field names 41
Applying numeric formats to data fields 42
Changing summary calculations 43
Suppressing subtotals 44
Showing and hiding data items 47
Hiding or showing items without data 49
Sorting the pivot table 51
Understanding Slicers. 52
Creating a Standard Slicer 54
Getting Fancy with Slicer Customizations 56
Size and placement 56
Data item columns 57
Miscellaneous slicer settings 58
Controlling Multiple Pivot Tables with One Slicer. 58
Creating a Timeline Slicer. 59

CHAPTER 4: **Using External Data with Power Pivot** 63
Loading Data from Relational Databases 64
Loading data from SQL Server. 64
Loading data from Microsoft Access databases 70
Loading data from other relational database systems 72
Loading Data from Flat Files 75
Loading data from external Excel files 76
Loading data from text files. 78
Loading data from the Clipboard 81
Loading Data from Other Data Sources. 82
Refreshing and Managing External Data Connections 83
Manually refreshing Power Pivot data 83
Setting up automatic refreshing 84
Preventing Refresh All. 85
Editing the data connection 86

CHAPTER 5: **Working Directly with the Internal Data Model**. . . . 89
Directly Feeding the Internal Data Model 89
Managing Relationships in the Internal Data Model 95
Managing Queries and Connections 96
Creating a New Pivot Table Using the Internal Data Model 97
Filling the Internal Data Model with Multiple External Data Tables. . . 98

CHAPTER 6: **Adding Formulas to Power Pivot**. 103
Enhancing Power Pivot Data with Calculated Columns. 103
Creating your first calculated column 104
Formatting calculated columns 105

Referencing calculated columns in other calculations. 106
Hiding calculated columns from end users 107
Utilizing DAX to Create Calculated Columns 108
Identifying DAX functions that are safe for calculated columns. 108
Building DAX-driven calculated columns 110
Month sorting in Power Pivot–driven pivot tables 112
Referencing fields from other tables 113
Nesting functions. 115
Understanding Calculated Measures 116
Creating a calculated measure 116
Editing and deleting calculated measures 118
Free Your Data with Cube Functions 119

CHAPTER 7: **Diving into DAX** 121
DAX Language Fundamentals 121
Using DAX operators. 125
Applying conditional logic in DAX 126
Working with DAX aggregate functions 128
Exploring iterator functions and row context 129
Understanding Filter Context 133
Getting context transitions with the CALCULATE function 135
Adding flexibility with the FILTER function. 137

PART 2: WRANGLING DATA WITH POWER QUERY 141

CHAPTER 8: **Introducing Power Query** 143
Power Query Basics 144
Starting the query 144
Understanding query steps 150
Refreshing Power Query data 152
Managing existing queries 153
Understanding Column-Level Actions 155
Understanding Table Actions 157

CHAPTER 9: **Power Query Connection Types** 159
Importing Data from Files 160
Getting data from Excel workbooks 160
Getting data from CSV and text files. 161
Getting data from PDF files 163
Getting data from folders. 164
Importing Data from Database Systems 165
A connection for every database type 165
Getting data from other data systems 167
Walk-through: Getting data from a database 168

Managing Data Source Settings 170
Data Profiling with Power Query 171
Data Profiling options 172
Data Profiling quick actions 173

CHAPTER 10: **Transforming Your Way to Better Data** 175
Completing Common Transformation Tasks 176
Removing duplicate records 176
Filling in blank fields 178
Concatenating columns 179
Changing case 181
Finding and replacing specific text 181
Trimming and cleaning text 183
Extracting the left, right, and middle values 184
Splitting columns using character markers 187
Pivoting and unpivoting fields 189
Creating Custom Columns 193
Concatenating with a custom column 195
Understanding data type conversions 196
Spicing up custom columns with functions 197
Adding conditional logic to custom columns 199
Grouping and Aggregating Data 201
Working with Custom Data Types 203

CHAPTER 11: **Making Queries Work Together** 207
Reusing Query Steps 208
Understanding the Append Feature 211
Creating the needed base queries 212
Appending the data 213
Understanding the Merge Feature 216
Understanding Power Query joins 216
Merging queries 217
Understanding Fuzzy Match 221

CHAPTER 12: **Extending Power Query with Custom Functions** 225
Creating and Using a Basic Custom Function 225
Creating a Function to Merge Data from Multiple Excel Files 229
Creating Parameter Queries 236
Preparing for a parameter query 236
Creating the base query 238
Creating the parameter query 239

PART 3: THE PART OF TENS 243

CHAPTER 13: Ten Ways to Improve Power Pivot Performance 245

Limit the Number of Rows and Columns in Your Data Model Tables 246
Use Views Instead of Tables 246
Avoid Multi-Level Relationships 246
Let the Back-End Database Servers Do the Crunching 247
Beware of Columns with Many Unique Values 248
Limit the Number of Slicers in a Report 248
Create Slicers Only on Dimension Fields 249
Disable the Cross-Filter Behavior for Certain Slicers 250
Use Calculated Measures Instead of Calculated Columns 250
Upgrade to 64-Bit Excel 251

CHAPTER 14: Ten Tips for Working with Power Query 253

Getting Quick Information from the Queries & Connections Pane 253
Organizing Queries in Groups 254
Selecting Columns in Queries Faster 255
Renaming Query Steps 256
Quickly Creating Reference Tables 257
Viewing Query Dependencies 258
Setting a Default Load Behavior 259
Preventing Automatic Data Type Changes 259
Disabling Privacy Settings to Improve Performance 261
Disabling Relationship Detection 261

INDEX 263

Introduction

Over the past few years, the concept of self-service business intelligence (BI) has taken over the corporate world. Self-service BI is a form of business intelligence in which end users can independently generate their own reports, run their own queries, and conduct their own analyses, without the need to engage the IT department.

The demand for self-service BI is a direct result of several factors:

- **More power users:** Organizations are realizing that no single enterprise reporting system or BI tool can accommodate all their users. Predefined reports and high-level dashboards may be sufficient for casual users, but a large portion of today's users are savvy enough to be considered power users. Power users have a greater understanding of data analysis and prefer to perform their own analysis, often within Excel.
- **Changing analytical needs:** In the past, business intelligence primarily consisted of IT-managed dashboards showing historic data on an agreed-upon set of key performance metrics. Managers now demand more dynamic predictive analysis, the ability to perform data discovery iteratively, and the freedom to take the hard left and right turns on data presentation. These managers often turn to Excel to provide the needed analytics and visualization tools.
- **Speed of BI:** Users are increasingly dissatisfied with the inability of IT to quickly deliver new reporting and metrics. Most traditional BI implementations fail specifically because the need for changes and answers to new questions overwhelmingly outpaces the IT department's ability to deliver them. As a result, users often find ways to work around the perceived IT bottleneck and ultimately build their own shadow BI (under the radar) solutions in Excel.

Recognizing the importance of the self-service BI revolution and the role Excel plays in it, Microsoft has made substantial investments in making Excel a player in the self-service BI arena by embedding both Power Pivot and Power Query directly into Excel.

You can integrate multiple data sources, define relationships between data sources, process analysis services cubes, and develop interactive dashboards that can be shared on the web. Indeed, the new Microsoft BI tools blur the line between Excel analysis and what is traditionally IT enterprise-level data management and reporting capabilities.

With these new tools in the Excel wheelhouse, it's becoming important for business analysts to expand their skill sets to new territory, including database management, query design, data integration, multidimensional reporting, and a host of other skills. Excel analysts have to expand their skill set knowledge base from the one-dimensional spreadsheets to relational databases, data integration, and multidimensional reporting.

That's where this book comes in. Here, you're introduced to the mysterious world of Power Pivot and Power Query. You find out how to leverage the rich set of tools and reporting capabilities to save time, automate data clean-up, and substantially enhance your data analysis and reporting capabilities.

About This Book

The goal of this book is to give you a solid overview of the self-service BI functionality offered by Power Pivot and Power Query. Each chapter guides you through practical techniques that enable you to

- Extract data from databases and external files for use in Excel reporting
- Scrape and import data from the web
- Build automated processes to clean and transform data
- Easily slice data into various views on the fly, gaining visibility from different perspectives
- Analyze large amounts of data and report them in a meaningful way
- Create powerful, interactive reporting mechanisms and dashboards

Within this book, you may note that some web addresses break across two lines of text. If you're reading this book in print and want to visit one of these web pages, simply key in the web address exactly as it's noted in the text, pretending as though the line break doesn't exist. If you're reading this as an e-book, you've got it easy — just click the web address to be taken directly to the web page.

Foolish Assumptions

Over the past few years, Microsoft has adopted an agile release cycle, allowing the company to release updates to Microsoft Office and the power BI tools practically monthly. This is great news for those who love seeing new features added to Power Pivot and Power Query. (It's not-so-great news if you're trying to document the features of these tools in a book.)

My assumption is that Microsoft will continue to add new bells and whistles to Power Pivot and Power Query at a rapid pace after publication of this book. So you may encounter new functionality not covered here.

The good news is that both Power Pivot and Power Query have stabilized and already have a broad feature set. So I'm also assuming that although changes will be made to these tools, they won't be so drastic as to turn this book into a doorstop. The core functionality covered in these chapters will remain relevant — even if the mechanics change a bit.

Icons Used in This Book

As you look in various places in this book, you see icons in the margins that indicate material of interest (or not, as the case may be). This section briefly describes each icon in this book.

Tips are beneficial because they help you save time or perform a task without having to do a lot of extra work. The tips in this book are time-saving techniques or pointers to resources that you should check out to get the maximum benefit from Excel.

Try to avoid doing anything marked with a Warning icon, which (as you might expect) represents a danger of one sort or another.

Whenever you see this icon, think *advanced* tip or technique. You might find these tidbits of useful information just too boring for words, or they could contain the solution you need to get a program running. Skip these bits of information whenever you like.

If you get nothing else out of a particular chapter or section, remember the material marked by this icon. This text usually contains an essential process or a bit of information you ought to remember.

Paragraphs marked with this icon reference the sample files for the book.

Beyond the Book

In addition to the book you have in your hands, you can access some extra content online. Check out the free Cheat Sheet for lists of Power Query text functions and Power Query date functions that are good to know. Just go to `www.dummies.com` and type **Excel Power Pivot & Power Query For Dummies Cheat Sheet** in the Search box.

If you want to follow along with the examples in this book, you can download the sample files at `www.dummies.com/go/excelpowerpivotpowerqueryfd2e`. The files are organized by chapter.

Where to Go from Here

It's time to start your self-service BI adventure! If you're primarily interested in Power Pivot, start with Chapter 1. If you want to dive right into Power Query, jump to Part 2, which begins at Chapter 8.

1 Supercharged Reporting with Power Pivot

IN THIS PART . . .

Think about data like a relational database.

Create your own Power Pivot data model.

Explore the workings of pivot tables.

Use external data with Power Pivot.

Manage the Power Pivot internal data model.

Create your own formulas in Power Pivot.

Delve deeper into the DAX formula language.

IN THIS CHAPTER

- Examining traditional Excel limitations
- Keeping up with database terminology
- Looking into relationships

Chapter 1
Thinking Like a Database

With the introduction of business intelligence (BI) tools such as Power Pivot and Power Query, it's becoming increasingly important for Excel analysts to understand core database principles. Unlike traditional Excel concepts, where the approach to developing solutions is relatively intuitive, you need to have a basic understanding of database terminology and architecture in order to get the most benefit from Power Pivot and Power Query. This chapter introduces you to a handful of fundamental concepts that you should know before taking on the rest of this book.

Exploring the Limits of Excel and How Databases Help

Years of consulting experience have brought this humble author face to face with managers, accountants, and analysts who all have had to accept this simple fact: Their analytical needs had outgrown Excel. They all faced fundamental challenges that stemmed from one or more of Excel's three problem areas: scalability, transparency of analytical processes, and separation of data and presentation.

Scalability

Scalability is the ability of an application to develop flexibly to meet growth and complexity requirements. In the context of this chapter, scalability refers to Excel's ability to handle ever-increasing volumes of data.

Imagine that you're working in a small company and using Excel to analyze its daily transactions. As time goes on, you build a robust process complete with all the formulas, pivot tables, and macros you need in order to analyze the data that is stored in your neatly maintained worksheet.

As the amount of data grows, you will first notice performance issues. The spreadsheet will become slow to load and then slow to calculate. Why does this happen? It has to do with the way Excel handles memory. When an Excel file is loaded, the entire file is loaded into RAM. Excel does this to allow for quick data processing and access. The drawback to this behavior is that every time the data in your spreadsheet changes, Excel has to reload the entire document into RAM. The net result in a large spreadsheet is that it takes a great deal of RAM to process even the smallest change. Eventually, every action you take in the gigantic worksheet is preceded by an excruciating wait.

Your pivot tables will require bigger pivot caches, almost doubling the Excel workbook's file size. Eventually, the workbook will become too big to distribute easily. You may even consider breaking down the workbook into smaller workbooks (possibly one for each region). This causes you to duplicate your work.

In time, you may eventually reach the 1,048,576-row limit of the worksheet. What happens then? Do you start a new worksheet? How do you analyze two datasets on two different worksheets as one entity? Are your formulas still good? Will you have to write new macros?

These are all issues that need to be addressed.

Of course, you will also encounter the Excel power customers, who will find various clever ways to work around these limitations. In the end, though, these methods will always be simply workarounds. Eventually, even these power customers will begin to think less about the most effective way to perform and present analysis of their data and more about how to make data "fit" into Excel without breaking their formulas and functions. Excel is flexible enough that a proficient customer can make most things fit just fine. However, when customers think only in terms of Excel, they're undoubtedly limiting themselves, albeit in an incredibly functional way.

In addition, these capacity limitations often force Excel customers to have the data prepared for them. That is, someone else extracts large chunks of data from

a large database and then aggregates and shapes the data for use in Excel. Should the serious analyst always be dependent on someone else for their data needs? What if an analyst could be given the tools to access vast quantities of data without being reliant on others to provide data? Could that analyst be more valuable to the organization? Could that analyst focus on the accuracy of the analysis and the quality of the presentation instead of routine Excel data maintenance?

A relational database system (such as Access or SQL Server) is a logical next step for the analyst who faces an ever-increasing data pool. Database systems don't usually have performance implications with large amounts of stored data, and are built to address large volumes of data. An analyst can then handle larger datasets without requiring the data to be summarized or prepared to fit into Excel. Also, if a process ever becomes more crucial to the organization and needs to be tracked in a more enterprise-acceptable environment, it will be easier to upgrade and scale up if that process is already in a relational database system.

Transparency of analytical processes

One of Excel's most attractive features is its flexibility. Each individual cell can contain text, a number, a formula, or practically anything else the customer defines. Indeed, this is one of the fundamental reasons that Excel is an effective tool for data analysis. Customers can use named ranges, formulas, and macros to create an intricate system of interlocking calculations, linked cells, and formatted summaries that work together to create a final analysis.

So what is the problem? The problem is that there is no transparency of analytical processes. It is extremely difficult to determine what is actually going on in a spreadsheet. Anyone who has had to work with a spreadsheet created by someone else knows all too well the frustration that comes with deciphering the various gyrations of calculations and links being used to perform analysis. Small spreadsheets that are performing modest analysis are painful to decipher, and large, elaborate, multi-worksheet workbooks are virtually impossible to decode, often leaving you to start from scratch.

Compared to Excel, database systems might seem rigid, strict, and unwavering in their rules. However, all this rigidity comes with a benefit.

Because only certain actions are allowable, you can more easily come to understand what is being done within structured database objects such as queries or stored procedures. If a dataset is being edited, a number is being calculated, or any portion of the dataset is being affected as part of an analytical process, you can readily see that action by reviewing the query syntax or the stored procedure code. Indeed, in a relational database system, you never encounter hidden formulas, hidden cells, or dead named ranges.

Separation of data and presentation

Data should be separate from presentation; you don't want the data to become too tied into any particular way of presenting it. For example, when you receive an invoice from a company, you don't assume that the financial data on that invoice is the true source of your data. It is *a presentation of* your data. It can be presented to you in other manners and styles on charts or on websites, but such representations are never the actual source of the data.

What exactly does this concept have to do with Excel? People who perform data analysis with Excel tend, more often than not, to fuse the data, the analysis, and the presentation. For example, you often see an Excel workbook that has 12 worksheets, each representing a month. On each worksheet, data for that month is listed along with formulas, pivot tables, and summaries. What happens when you're asked to provide a summary by quarter? Do you add more formulas and worksheets to consolidate the data on each of the month worksheets? The fundamental problem in this scenario is that the worksheets actually represent data values that are fused into the presentation of the analysis.

The point being made here is that data should not be tied to a particular presentation, no matter how apparently logical or useful it may be. However, in Excel, it happens all the time.

In addition, as discussed earlier in this chapter, because all manners and phases of analysis can be done directly within a spreadsheet, Excel cannot effectively provide adequate transparency to the analysis. Each cell has the potential to hold formulas, be hidden, and contain links to other cells. In Excel, this blurs the line between analysis and data, which makes it difficult to determine exactly what is going on in a spreadsheet. Moreover, it takes a great deal of effort in the way of manual maintenance to ensure that edits and unforeseen changes don't affect previous analyses.

Relational database systems inherently separate analytical components into tables, queries, and reports. By separating these elements, databases make data less sensitive to changes and create a data analysis environment in which you can easily respond to new requests for analysis without destroying previous analyses.

You may find that you manipulate Excel's functionalities to approximate this database behavior. If so, you must consider that if you're using Excel's functionality to make it behave like a database application, perhaps the real thing just might have something to offer. Utilizing databases for data storage and analytical needs would enhance overall data analysis and would allow Excel power customers to focus on the presentation in their spreadsheets.

In these days of big data, customers demand more, not less, complex data analysis. Excel analysts will need to add tools to their repertoires to avoid being simply "spreadsheet mechanics." Excel can be stretched to do just about anything, but maintaining such creative solutions can be a tedious manual task. You can be sure that the sexy aspect of data analysis does not lie in the routine data management within Excel; rather, it lies in leveraging BI Tools such as providing clients with the best solution for any situation.

Getting to Know Database Terminology

The terms *database, table, record, field,* and *value* indicate a hierarchy from largest to smallest. These same terms are used with virtually all database systems, so you should learn them well.

Databases

Generally, the word *database* is a computer term for a collection of information concerning a certain topic or business application. A database helps you organize this related information in a logical fashion for easy access and retrieval. Certain older database systems used the term *database* to describe individual tables. The current use of *database* applies to all elements of a database system.

Databases aren't only for computers. Manual databases are sometimes referred to as manual filing systems or manual database systems. These filing systems usually consist of people, papers, folders, and filing cabinets — paper is the key to a manual database system. In a real-life manual database system, you probably have in-baskets and out-baskets and some type of formal filing method. You access information manually by opening a file cabinet, removing a file folder, and finding the correct piece of paper. Customers fill out paper forms for input, perhaps by using a keyboard to input information that is printed on forms. You find information by manually sorting the papers or by copying information from many papers to another piece of paper (or even into an Excel spreadsheet). You may use a spreadsheet or calculator to analyze the data or display it in new and interesting ways.

Tables

A database stores information in a carefully defined structure known as a table. A *table* is just a container for raw information (called *data*), similar to a folder in a manual filing system. Each table in a database contains information about a single

entity, such as a person or product, and the data in the table is organized into rows and columns. A relational database system stores data in related tables. For example, a table containing employee data (names and addresses) may be related to a table containing payroll information (pay date, pay amount, and check number).

To use database wording, a table is an object. As you design and work with databases, it's important to see each table as a unique entity and to see how each table relates to the other objects in the database.

In most database systems, you can view the contents of a table in a spreadsheet-like form called a *datasheet*, composed of rows and columns (known as *records* and *fields*, respectively — see the following section). Although a datasheet and a spreadsheet are superficially similar, a datasheet is quite a different type of object. You typically cannot make changes or add calculations directly within a table. Your interaction with tables will primarily come in the form of queries or views — see the later section "Queries".

Records, fields, and values

A database table is divided into rows (called *records*) and columns (called *fields*), with the first row (the heading on top of each column) containing the names of the fields in the database.

Each row is a single record containing fields that are related to that record. In a manual system, the rows are individual forms (sheets of paper), and the fields are equivalent to the blank areas on a printed form that you fill in.

Each column is a field that includes many properties specifying the type of data contained within the field and how the database should handle the field's data. These properties include the name of the field (Company) and the type of data in the field (Text). A field may include other properties as well. For example, the Address field's Size property tells the database the maximum number of characters allowed for the address.

At the intersection of a record and a field is a *value* — the actual data element. For example, in a field named Company, a company name entered into that field would represent one data value.

REMEMBER

When working with Microsoft Access, the term *field* is used to refer to an attribute stored in a record. In many other database systems, including SQL Server, *column* is the expression you hear most often in place of *field* — field and column mean the same thing. The exact terminology that's used relies somewhat on the context of the database system underlying the table containing the record.

Queries

Most relational database systems allow the creation of queries (sometimes called views). A query extracts information from the tables in the database; a query selects and defines a group of records that fulfill a certain condition. Most database outputs are based on queries that combine, filter, or sort data before it's displayed. Queries are often called from other database objects, such as stored procedures, macros, or code modules. In addition to extracting data from tables, queries can be used to change, add, or delete database records.

An example of a query is when a person at the sales office tells the database, "Show me all customers, in alphabetical order by name, who are located in Massachusetts and who made a purchase over the past six months." Or "Show me all customers who bought Chevrolet car models within the past six months, and display them sorted by customer name and then by sale date."

Rather than ask the question using English words, a person uses a special syntax, such as Structured Query Language (or SQL), to communicate to the database what the query will need to do.

Understanding Relationships

After you understand the basic terminology of databases, it's time to focus on one of their more useful features: A *relationship* is the mechanism by which separate tables are related to each other. You can think of a relationship as a kind of VLOOKUP, in which you relate the data in one data range to the data in another data range using an index or a unique identifier. In databases, relationships do the same thing, but without the hassle of writing formulas.

Relationships are important because most of the data you work with fits into a multidimensional hierarchy of sorts. For example, you may have a table showing customers who buy products. These customers require invoices that have invoice numbers. Those invoices have multiple lines of transactions listing what they bought. A hierarchy exists there.

Now, in the one-dimensional spreadsheet world, this data typically would be stored in a flat table, like the one shown in Figure 1-1.

Because customers have more than one invoice, the customer information (in this example, CustomerID and CustomerName) has to be repeated. This causes a problem when that data needs to be updated.

	A	B	C	D	E	F
1	CustomerID	CustomerName	InvoiceNumber	InvoiceDate	Quantity	UnitPrice
2	BAKERSEM0001	Baker's Emporium Inc.	ORDST1025	5/8/2005	1	19.95
3	BAKERSEM0001	Baker's Emporium Inc.	ORDST1025	5/8/2005	5	1759.95
4	BAKERSEM0001	Baker's Emporium Inc.	ORDST1025	5/8/2005	4	9.95
5	BAKERSEM0001	Baker's Emporium Inc.	STDINV2251	4/12/2007	4	9.95
6	AARONFIT0001	Aaron Fitz Electrical	ORDST1026	5/8/2005	5	9.95
7	AARONFIT0001	Aaron Fitz Electrical	ORDST1026	5/8/2005	3	1759.95
8	AARONFIT0001	Aaron Fitz Electrical	ORDST1026	5/8/2005	2	79.95
9	AARONFIT0001	Aaron Fitz Electrical	STDINV2252	4/12/2007	3	1759.95
10	AARONFIT0001	Aaron Fitz Electrical	STDINV2252	4/12/2007	5	9.95
11	METROPOL0001	Metropolitan Fiber Systems	ORD1002	5/7/2004	1	9.95
12	AARONFIT0001	Aaron Fitz Electrical	INV1024	2/10/2004	1	119.95
13	AARONFIT0001	Aaron Fitz Electrical	INV1025	2/15/2004	1	109.95
14	LECLERC0001	LeClerc & Associates	ORDPH1005	5/10/2004	2	189.95
15	MAGNIFIC0001	Magnificent Office Images	ORD1000	5/8/2004	1	359.95
16	HOLLINGC0001	Holling Communications Inc.	ORD1001	5/10/2004	2	59.95
17	MAHLERST0001	Mahler State University	ORDST1008	5/10/2004	1	5999.95

FIGURE 1-1: Data is stored in an Excel spreadsheet using a flat-table format.

For example, imagine that the name of the company Aaron Fitz Electrical changes to Fitz and Sons Electrical. Looking at Figure 1-1, you see that multiple rows contain the old name. You would have to ensure that every row containing the old company name is updated to reflect the change. Any rows you miss will not correctly map back to the right customer.

Wouldn't it be more logical and efficient to record the name and information of the customer only one time? Then, rather than have to write the same customer information repeatedly, you could simply have some form of customer reference number.

This is the idea behind relationships. You can separate customers from invoices, placing each in their own tables. Then you can use a unique identifier (such as CustomerID) to relate them together.

Figure 1-2 illustrates how this data would look in a relational database. The data would be split into three separate tables: Customers, InvoiceHeader, and InvoiceDetails. Each table would then be related using unique identifiers (CustomerID and InvoiceNumber, in this case).

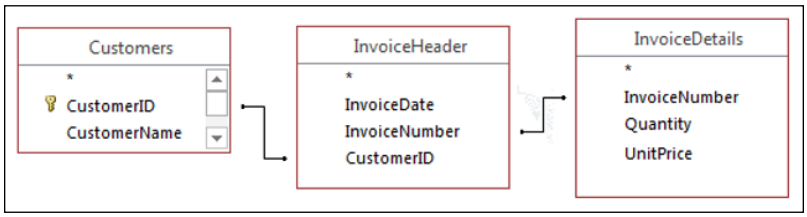


FIGURE 1-2: Databases use relationships to store data in unique tables and simply relate these tables to each other.

The Customers table would contain a unique record for each customer. That way, if you need to change a customer's name, you would need to make the change in only that record. Of course, in real life, the Customers table would include other attributes, such as customer address, customer phone number, and customer start date. Any of these other attributes could also be easily stored and managed in the Customers table.

The most common relationship type is a *one-to-many* relationship. That is, for each record in one table, one record can be matched to many records in a separate table. For example, an invoice header table is related to an invoice detail table. The invoice header table has a unique identifier: Invoice Number. The invoice detail will use the Invoice Number for every record representing a detail of that particular invoice.

Another kind of relationship type is the *one-to-one* relationship: For each record in one table, one and only one matching record is in a different table. Data from different tables in a one-to-one relationship can technically be combined into a single table.

Finally, in a *many-to-many* relationship, records in both tables can have any number of matching records in the other table. For instance, a database at a bank may have a table of the various types of loans (home loan, car loan, and so on) and a table of customers. A customer can have many types of loans. Meanwhile, each type of loan can be granted to many customers.

If your head is spinning from all this database talk, don't worry. You don't need to be an expert database modeler to use Power Pivot. But it's important to understand these concepts. The better you understand how data is stored and managed in databases, the more effectively you'll leverage Power Pivot for reporting.

IN THIS CHAPTER

- Getting to know the Internal Data Model
- Activating the Power Pivot add-in
- Linking to Excel data
- Managing relationships

Chapter 2
Introducing Power Pivot

Over the past decade or so, corporate managers, eager to turn impossible amounts of data into useful information, drove the business intelligence (BI) industry to innovate new ways of synthesizing data into meaningful insights. During this period, organizations spent lots of time and money implementing big enterprise reporting systems to help keep up with the hunger for data analytics and dashboards.

Recognizing the importance of the BI revolution and the place that Excel holds within it, Microsoft proceeded to make substantial investments in improving Excel's BI capabilities. It specifically focused on Excel's *self-service* BI capabilities and its ability to better manage and analyze information from the increasing number of available data sources.

The key product of that endeavor was essentially Power Pivot (introduced in Excel 2010 as an add-in). With Power Pivot came the ability to set up relationships between large, disparate data sources. For the first time, Excel analysts were able to add a relational view to their reporting without the use of problematic functions such as VLOOKUPS. The ability to merge data sources with hundreds of thousands of rows into one analytical engine within Excel was groundbreaking.

With the release of Excel 2016, Microsoft incorporated Power Pivot directly into Excel. The powerful capabilities of Power Pivot are available out of the box!

In this chapter, you get an overview of those capabilities by exploring the key features, benefits, and capabilities of Power Pivot.

Understanding the Power Pivot Internal Data Model

At its core, Power Pivot is essentially a SQL Server Analysis Services engine made available by way of an in-memory process that runs directly within Excel. Its technical name is the xVelocity analytics engine. However, in Excel, it's referred to as the Internal Data Model.

Every Excel workbook contains an *Internal Data Model*, a single instance of the Power Pivot in-memory engine. The most effective way to interact with the Internal Data Model is to use the Power Pivot Ribbon interface (see Figure 2-1).

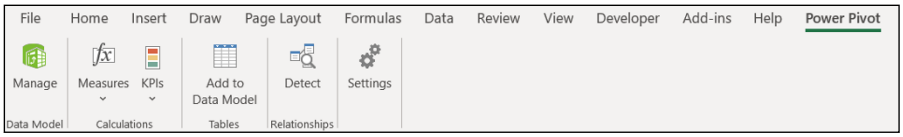


FIGURE 2-1: The Power Pivot Ribbon interface.

The Power Pivot Ribbon interface exposes the full set of functionalities you don't get with the standard Excel Data tab. Here are a few examples of functionality available with the Power Pivot interface:

- You can browse, edit, filter, and apply custom sorting to data.
- You can create custom calculated columns that apply to all rows in the data import.
- You can define a default number format to use when the field appears in a pivot table.
- You can easily configure relationships via the handy Graphical Diagram view.
- You can choose to prevent certain fields from appearing in the PivotTable Field List.

As with everything else in Excel, the Internal Data Model does have limitations. Most Excel users will not likely hit these limitations, because Power Pivot's compression algorithm is typically able to shrink imported data to about one-tenth its original size. For example, a 100MB text file would take up only approximately 10MB in the Internal Data Model.

Nevertheless, it's important to understand the maximum and configurable limits for Power Pivot Data Models. Table 2-1 highlights them.

WHERE'S THE POWER PIVOT TAB?

Organizations often install Excel in accordance with their own installation policies. In some organizations, Excel is installed without the PowerPivot add-in activated, so the Power Pivot tab won't be visible. If you don't see the Power Pivot tab shown in Figure 2-1, you can follow these steps to activate it:

1. **Go up to the Excel Ribbon and choose File Options.**
2. **Select the Add-Ins option on the left.**
3. **From the Manage drop-down list, select COM Add-Ins and click Go.**
4. **In the list of available COM Add-Ins, check the box next to Microsoft Office Power Pivot for Excel and click OK.**
5. **If the Power Pivot tab doesn't appear on the Ribbon, quit and restart Excel.**

TABLE 2-1 Limitations of the Internal Data Model

Object	Specification
Data model size	In 32-bit environments, Excel workbooks are subject to a 2GB limit. This includes the in-memory space shared by Excel, the Internal Data Model, and add-ins that run in the same process. In 64-bit environments, there are no hard limits on file size. Workbook size is limited only by available memory and system resources.
Number of tables in the data model	No hard limits exist on the count of tables. However, all tables in the data model cannot exceed 2,147,483,647 bytes.
Number of rows in each table in the data model	1,999,999,997
Number of columns and calculated columns in each table in the data model	The number cannot exceed 2,147,483,647 bytes.
Number of distinct values in a column	1,999,999,997
Characters in a column name	100 characters
String length in each field	It's limited to 536,870,912 bytes (512MB), equivalent to 268,435,456 Unicode characters (256 mega-characters).

A WORD ON COMPATIBILITY

Since Excel 2010 was released, Microsoft has made several versions of Power Pivot available. Different versions of Power Pivot are being used, depending on the version of Excel. Be careful when sharing Power Pivot workbooks in environments where some of your audience is using earlier versions of Excel while others are using more recent versions of Excel. Opening and refreshing a workbook that contains a Power Pivot model created with an older version of the Power Pivot add-in will trigger an automatic upgrade of the underlying model. When this happens, users with older versions of Excel will no longer be able to use the Power Pivot model in the workbook.

Power Pivot workbooks created in a version of Excel that is older than your version should give you no problems. However, you won't be able use Power Pivot workbooks created in a version of Excel newer than your version.

Linking Excel Tables to Power Pivot

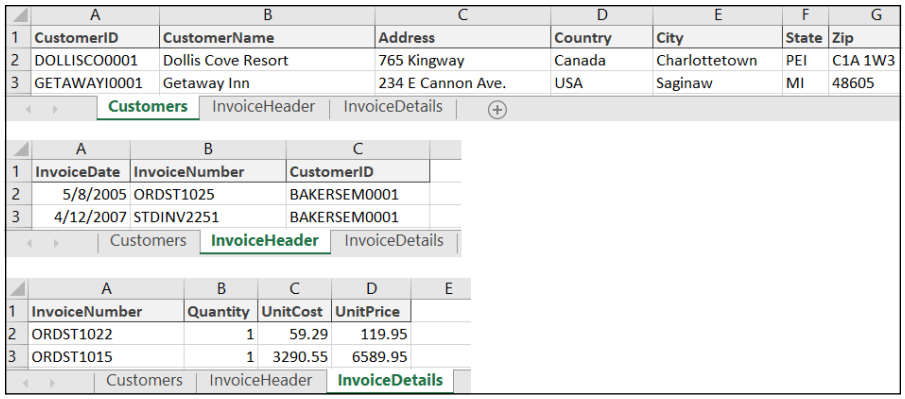
The first step in using Power Pivot is to fill it with data. You can either import data from external data sources or link to Excel tables in your current workbook. I cover importing data from external data sources in Chapter 4. For now, let me start this walkthrough by showing you how to link three Excel tables to Power Pivot.

You can find the sample file for this chapter on this book's companion website at `www.dummies.com/go/excelpowerpivotpowerqueryfd2e` in the workbook named `Chapter 2 Samples.xlsx`.

In this scenario, you have three data sets in three different worksheets: Customers, InvoiceHeader, and InvoiceDetails (see Figure 2-2).

The Customers data set contains basic information, such as CustomerID, Customer Name, and Address. The InvoiceHeader data set contains data that points specific invoices to specific customers. The InvoiceDetails data set contains the specifics of each invoice.

To analyze revenue by customer and month, it's clear that you first need to somehow join these three tables together. In the past, you would have to go through a series of gyrations involving VLOOKUP or other clever formulas. But with Power Pivot, you can build these relationships in just a few clicks.



	A	B	C	D	E	F	G
1	CustomerID	CustomerName	Address	Country	City	State	Zip
2	DOLLISCO0001	Dollis Cove Resort	765 Kingway	Canada	Charlottetown	PEI	C1A 1W3
3	GETAWAYI0001	Getaway Inn	234 E Cannon Ave.	USA	Saginaw	MI	48605

Customers | InvoiceHeader | InvoiceDetails

	A	B	C
1	InvoiceDate	InvoiceNumber	CustomerID
2	5/8/2005	ORDST1025	BAKERSEM0001
3	4/12/2007	STDINV2251	BAKERSEM0001

Customers | InvoiceHeader | InvoiceDetails

	A	B	C	D	E
1	InvoiceNumber	Quantity	UnitCost	UnitPrice	
2	ORDST1022	1	59.29	119.95	
3	ORDST1015	1	3290.55	6589.95	

Customers | InvoiceHeader | InvoiceDetails

FIGURE 2-2: You want to use Power Pivot to analyze the data in the Customers, InvoiceHeader, and InvoiceDetails worksheets.

Preparing Excel tables

When linking Excel data to Power Pivot, best practice is to first convert the Excel data to explicitly named tables. Although not technically necessary, giving tables friendly names helps track and manage your data in the Power Pivot data model. If you don't convert your data to tables first, Excel does it for you and gives your tables useless names like Table1, Table2, and so on.

Follow these steps to convert each data set into an Excel table:

1. **Go to the Customers tab and click anywhere inside the data range.**
2. **Press Ctrl+T on the keyboard.**

   This step opens the Create Table dialog box, shown in Figure 2-3.

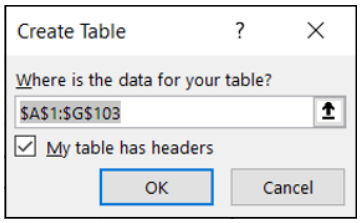


FIGURE 2-3: Convert the data range into an Excel table.

3. **In the Create Table dialog box, ensure that the range for the table is correct and that the My Table Has Headers check box is selected. Click the OK button.**

   You should now see the Table Design tab on the Ribbon.

4. **Click the Table Design tab, and use the Table Name input to give your table a friendly name, as shown in Figure 2-4.**

   This step ensures that you can recognize the table when adding it to the Internal Data Model.

5. **Repeat Steps 1 through 4 for the Invoice Header and Invoice Details data sets.**

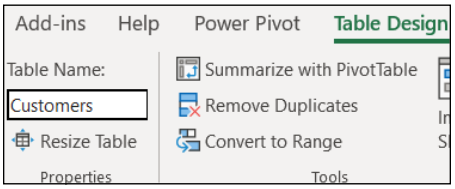


FIGURE 2-4: Give your newly created Excel table a friendly name.

Adding Excel Tables to the data model

After you convert your data to Excel tables, you're ready to add them to the Power Pivot data model. Follow these steps to add the newly created Excel tables to the data model using the Power Pivot tab:

1. **Place the cursor anywhere inside the Customers Excel table.**
2. **Go to the Power Pivot tab on the Ribbon and click the Add to Data Model command.**

Power Pivot creates a copy of the table and opens the Power Pivot window, shown in Figure 2-5.

Although the Power Pivot window looks like Excel, it's a separate program altogether. Notice that the grid for the Customers table offers row numbers but no column references. Also notice that you cannot edit the data within the table. This data is simply a snapshot of the Excel table you imported.

Additionally, if you look at the Windows taskbar at the bottom of the screen, you can see that Power Pivot has a separate window from Excel. You can switch between Excel and the Power Pivot window by clicking each respective program on the taskbar.

TIP

If your Windows taskbar combines taskbar buttons, the Power Pivot button may be hidden with the Excel group of buttons. Click or mouse over the Excel icon on the taskbar to reach the Power Pivot button.

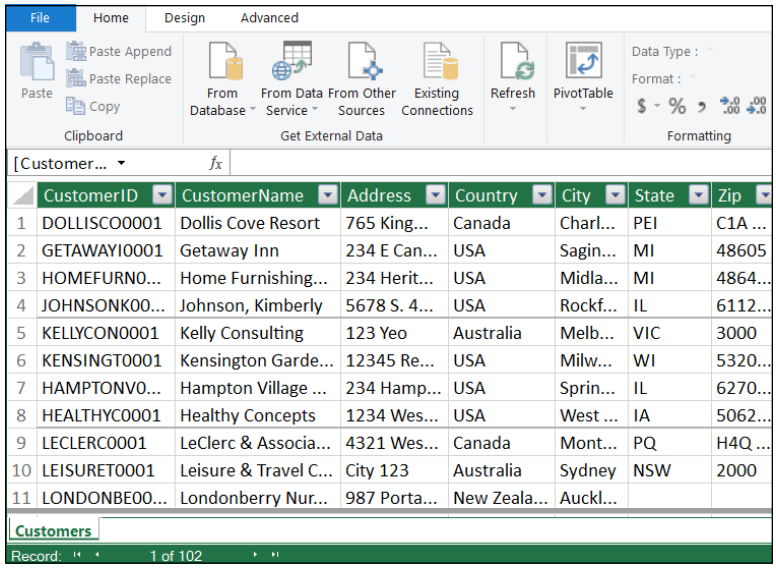


FIGURE 2-5: The Power Pivot window shows all the data that exists in your data model.

Repeat Steps 1 and 2 in the preceding list for your other Excel tables: InvoiceHeader, InvoiceDetails. After you've imported all your Excel tables into the data model, the Power Pivot window will show each data set on its own tab, as shown in Figure 2-6.

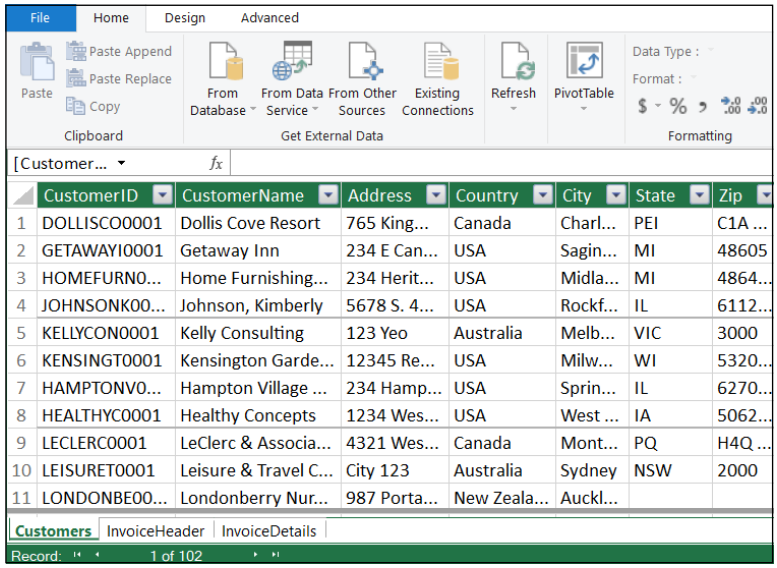


FIGURE 2-6: Each table you add to the data model is placed on its own tab in Power Pivot.

REMEMBER

Because the data you just imported into Power Pivot comes from an Excel table within the current workbook, Power Pivot will consider these linked tables. So, even though the data shown in Power Pivot is a snapshot at the time you added it, the data automatically updates when you edit the source table in Excel. Linked tables are the only kind of data source that automatically refreshes as the data within changes.

Creating relationships between Power Pivot tables

At this point, Power Pivot knows that you have three tables in the data model but has no idea how the tables relate to one another. You connect these tables by defining relationships between the Customers, Invoice Details, and Invoice Header tables. You can do so directly within the Power Pivot window.

TIP

If you've inadvertently closed the Power Pivot window, you can easily reopen it by clicking the Manage command button on the Power Pivot Ribbon tab.

Follow these steps to create relationships between your tables:

1. **Activate the Power Pivot window and click the Diagram View command button on the Home tab.**

   The Power Pivot screen you see shows a visual representation of all tables in the data model, as shown in Figure 2-7.

TIP

   You can move the tables in Diagram view by simply clicking and dragging them.

   The idea is to identify the primary index keys in each table and connect them. In this scenario, the Customers table and the Invoice Header table can be connected using the CustomerID field. The Invoice Header and Invoice Details tables can be connected using the InvoiceNumber field.

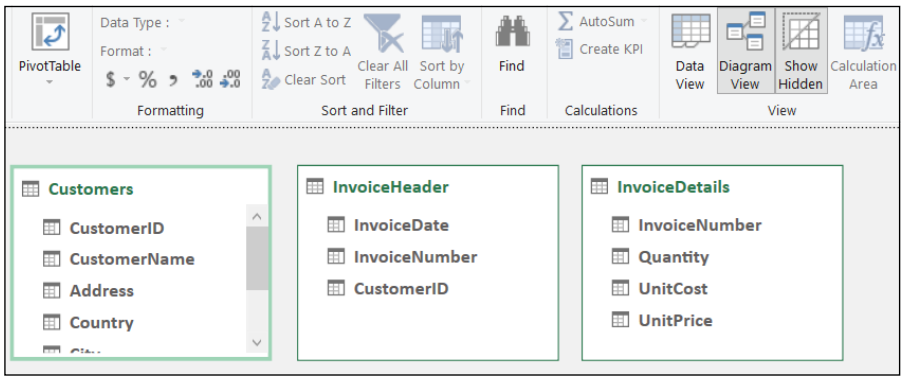


FIGURE 2-7: Diagram view allows you to see all tables in the data model.

2. **Click and drag a line from the CustomerID field in the Customers table to the CustomerID field in the Invoice Header table, as demonstrated in Figure 2-8.**

3. **Click and drag a line from the InvoiceNumber field in the Invoice Header table to the InvoiceNumber field in the Invoice Details table.**

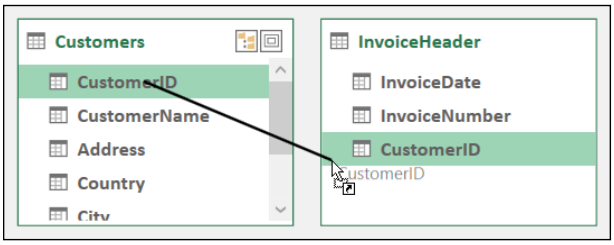


FIGURE 2-8: To create a relationship, you simply click and drag a line between the fields in your tables.

At this point, your diagram will look similar to Figure 2-9. Notice that Power Pivot shows a line between the tables you just connected. In database terms, these are referred to as *joins.*

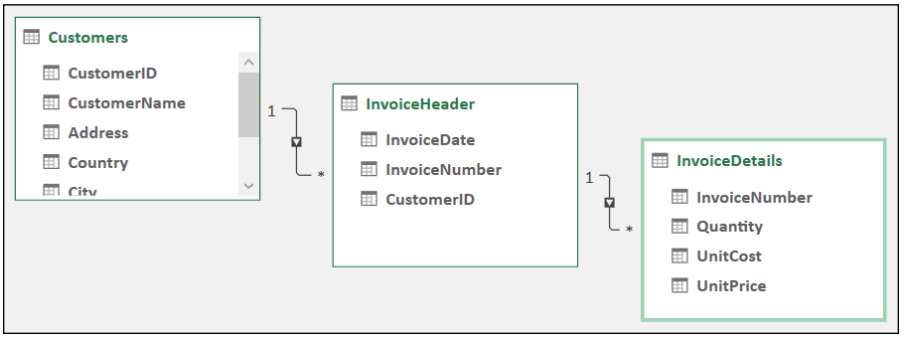


FIGURE 2-9: When you create relationships, the Power Pivot diagram shows join lines between tables.

The joins in Power Pivot are always one-to-many joins. This means that when a table is joined to another, one of the tables has unique records with unique index numbers (CustomerID for example), while the other can have many records where index numbers are duplicated.

Notice in Figure 2-9 that the join lines have arrows pointing from a table to another table. The arrows in these join lines will always point to the table that has the duplicated index. In this case, the Customers table contains a unique list of customers, each having its own unique identifier. No CustomerID in that table is duplicated. The Invoice header table has many rows for each CustomerID; each customer can have many invoices.

TIP

To close the diagram and return to seeing the data tables, click the Data View command in the Power Pivot window.

Managing existing relationships

If you need to edit or delete a relationship between two tables in your data model, you can do so by following these steps:

1. **Open the Power Pivot window, select the Design tab, and then select the Manage Relationships command.**
2. **In the Manage Relationships dialog box, shown in Figure 2-10, click the relationship you want to work with and click Edit or Delete.**

   If you click Edit, the Edit Relationship dialog box (shown in Figure 2-11) appears. The columns used to form the relationship are highlighted. Here, you can redefine the relationship by simply selecting the appropriate columns. You can also use the Active check box to disable or enable the relationship.

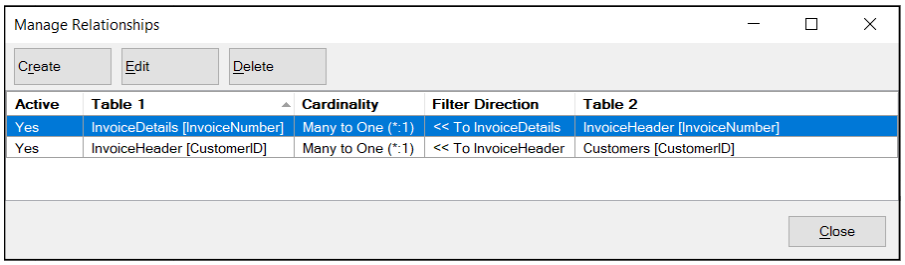


FIGURE 2-10: Use the Manage Relationships dialog box to edit or delete existing relationships.

REMEMBER

In Figure 2-9, you see a graphic of an arrow between the list boxes. The graphic has an asterisk next to the list box on the left, and a number 1 next to the list box on the right. The number 1 basically indicates that the model will use the table listed on the right as the source for a unique primary key.

Every relationship must have a field that you designate as the primary key. Primary key fields are necessary in the data model to prevent aggregation errors and duplications. In that light, the Excel data model must impose some strict rules around the primary key.

You cannot have any duplicates or null values in a field being used as the primary key. So the Customers table (refer to Figure 2-9) must have all unique values in the CustomerID field, with no blanks or null values. This is the only way that Excel can ensure data integrity when joining multiple tables.

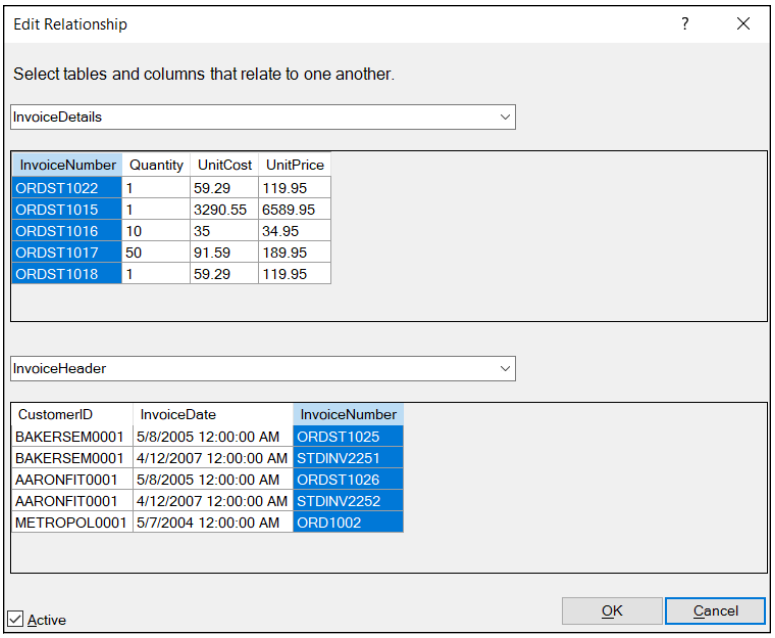


FIGURE 2-11: Use the Edit Relationship dialog box to adjust the tables and field names that define the selected relationship.

At least one of your tables must contain a field that serves as a primary key — that is, a field that contains only unique values and no blanks.

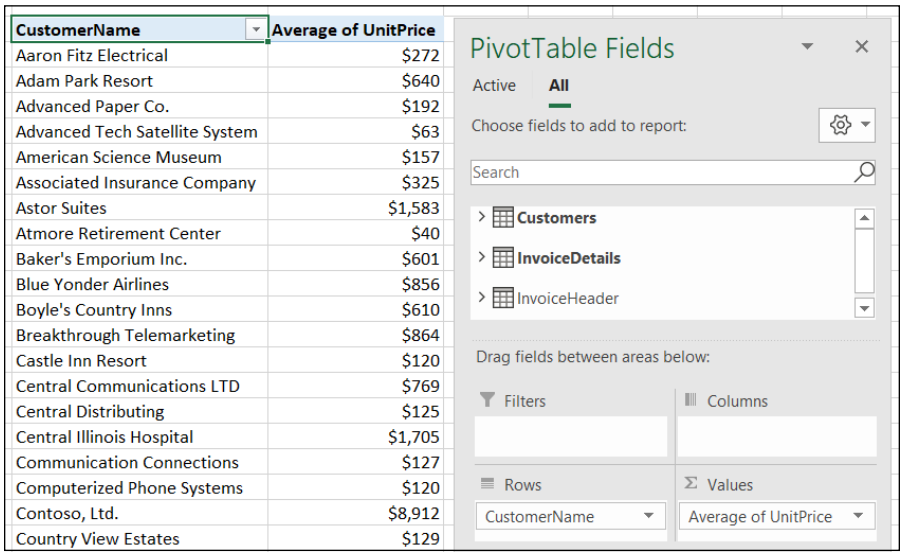
Using the Power Pivot data model in reporting

After you define the relationships in your Power Pivot data model, it's essentially ready for action. In terms of Power Pivot, *action* means analysis with a pivot table. In fact, all Power Pivot data is presented through the framework of pivot tables.

In Chapter 3, you dive deep into the workings of pivot tables. For now, dip just a toe in and create a simple pivot table from your new Power Pivot data model:

1. **Activate the Power Pivot window, select the Home tab, and then click the Pivot Table command button.**
2. **Specify whether you want the pivot table placed on a new worksheet or an existing sheet.**
3. **Build out the needed analysis just as you would build out any other standard pivot table, using the Pivot Field List.**

The pivot table shown in Figure 2-12 contains all tables in the Power Pivot data model. Unlike a standard pivot table, where you can use fields from only one table, the relationships defined the internal data model allow you to use any of the fields from any of the tables. With this configuration, you have a powerful cross-table analytical engine in the form of a familiar pivot table. Here, you can see that you're calculating the average unit price by customer.



CustomerName	Average of UnitPrice
Aaron Fitz Electrical	$272
Adam Park Resort	$640
Advanced Paper Co.	$192
Advanced Tech Satellite System	$63
American Science Museum	$157
Associated Insurance Company	$325
Astor Suites	$1,583
Atmore Retirement Center	$40
Baker's Emporium Inc.	$601
Blue Yonder Airlines	$856
Boyle's Country Inns	$610
Breakthrough Telemarketing	$864
Castle Inn Resort	$120
Central Communications LTD	$769
Central Distributing	$125
Central Illinois Hospital	$1,705
Communication Connections	$127
Computerized Phone Systems	$120
Contoso, Ltd.	$8,912
Country View Estates	$129

FIGURE 2-12: You now have a Power Pivot-driven pivot table that aggregates across multiple tables.

In the days before Power Pivot, this analysis would have been a bear to create. You would have had to build VLOOKUP formulas to get from Customer Number to Invoice Number, and then another set of VLOOKUP formulas to get from Invoice Numbers to Invoice Details. And after all that formula building, you still would have had to find a way to aggregate the data to the average unit price per customer.

IN THIS CHAPTER

- Getting to know pivot tables
- Laying out the geography of a pivot table
- Building your first pivot table
- Creating top and bottom reports
- Understanding, creating, and formatting slicers
- Sprucing up slicers with customization
- Controlling multiple pivot tables with slicers
- Using timeline slicers

Chapter 3

The Pivotal Pivot Table

When creating Power Pivot data models, you will have to use some form of pivot table structure to expose the data in those models available to your audience.

Pivot tables have a reputation for being complicated, but if you're new to pivot tables, rest easy. This chapter gives you the fundamental understanding you need in order to analyze and report on the data in your Power Pivot data model. After completing this introduction, you'll be pleasantly surprised at how easy it is to create and use pivot tables.

You can find the sample files for this chapter on this book's companion website at `www.dummies.com/go/excelpowerpivotpowerqueryfd2e` in the workbooks named `Chapter 3 Samples.xlsx` and `Chapter 3 Slicers.xlsx`.

Introducing the Pivot Table

A *pivot table* is a robust tool that allows you to create an interactive view of your dataset, commonly referred to as a *pivot table report*. With a pivot table report, you can quickly and easily categorize your data into groups, summarize large amounts of data into meaningful analyses, and interactively perform a wide variety of calculations.

Pivot tables get their name from the way they allow you to drag and drop fields within the pivot table report to dynamically change (or *pivot*) perspective and give you an entirely new analysis using the same data source.

Think of a pivot table as an object you can point at your dataset. When you look at your dataset through a pivot table, you can see your data from different perspectives. The dataset itself doesn't change, and it's not connected to the pivot table. The pivot table is simply a tool you're using to dynamically change analyses, apply varying calculations, and interactively drill down to the detail records.

The reason a pivot table is so well suited for reporting is that you can refresh the analyses shown through the pivot table by simply updating the dataset that it points to. You can set up the analysis and presentation layers only one time; then, to refresh the reporting mechanism, all you have to do is click a button.

Let's start this exploration of pivot tables with a lesson on the anatomy of a pivot table.

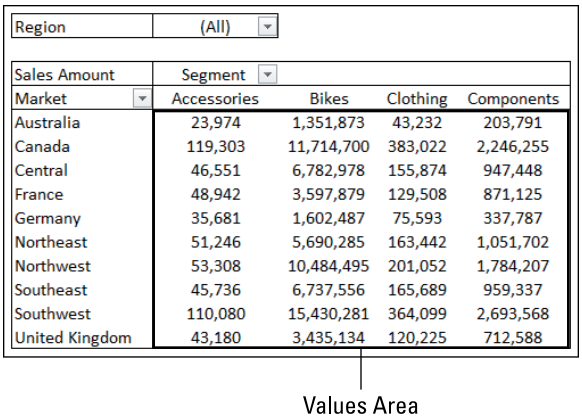
Defining the Four Areas of a Pivot Table

A pivot table is composed of four areas. The data you place in these areas defines both the utility and appearance of the pivot table. Take a moment to understand the function of each of these four areas.

Values area

The *values area,* as shown in Figure 3-1, is the large, rectangular area below and to the right of the column and row headings. In the example in Figure 3-1, the values area contains a sum of the values in the Sales Amount field.

The values area calculates and counts data. The data fields that you drag and drop there are typically those that you want to measure — fields, such as Sum of Revenue, Count of Units, or Average of Price.

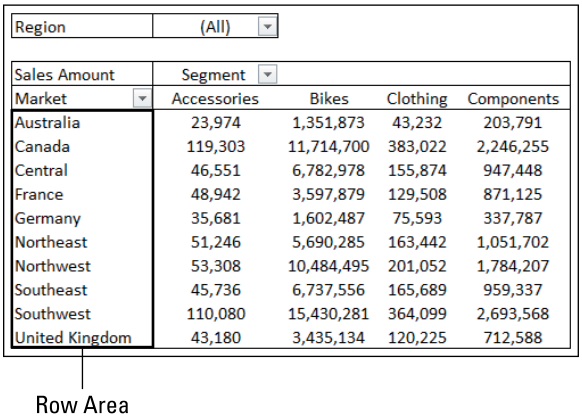


Region	(All)			
Sales Amount	Segment			
Market	Accessories	Bikes	Clothing	Components
Australia	23,974	1,351,873	43,232	203,791
Canada	119,303	11,714,700	383,022	2,246,255
Central	46,551	6,782,978	155,874	947,448
France	48,942	3,597,879	129,508	871,125
Germany	35,681	1,602,487	75,593	337,787
Northeast	51,246	5,690,285	163,442	1,051,702
Northwest	53,308	10,484,495	201,052	1,784,207
Southeast	45,736	6,737,556	165,689	959,337
Southwest	110,080	15,430,281	364,099	2,693,568
United Kingdom	43,180	3,435,134	120,225	712,588

FIGURE 3-1: The values area of a pivot table calculates and counts data.

Row area

The *row area* is shown in Figure 3-2. Placing a data field into the row area displays the unique values from that field down the rows of the left side of the pivot table. The row area typically has at least one field, although it's possible to have no fields.



Region	(All)			
Sales Amount	Segment			
Market	Accessories	Bikes	Clothing	Components
Australia	23,974	1,351,873	43,232	203,791
Canada	119,303	11,714,700	383,022	2,246,255
Central	46,551	6,782,978	155,874	947,448
France	48,942	3,597,879	129,508	871,125
Germany	35,681	1,602,487	75,593	337,787
Northeast	51,246	5,690,285	163,442	1,051,702
Northwest	53,308	10,484,495	201,052	1,784,207
Southeast	45,736	6,737,556	165,689	959,337
Southwest	110,080	15,430,281	364,099	2,693,568
United Kingdom	43,180	3,435,134	120,225	712,588

FIGURE 3-2: The row area of a pivot table gives you a row-oriented perspective.

The types of data fields that you would drop here include those that you want to group and categorize, such as Products, Names, and Locations.

Column area

The *column area* is composed of headings that stretch across the top of columns in the pivot table.

As you can see in Figure 3-3, the column area stretches across the top of the columns. In this example, it contains the unique list of business segments.

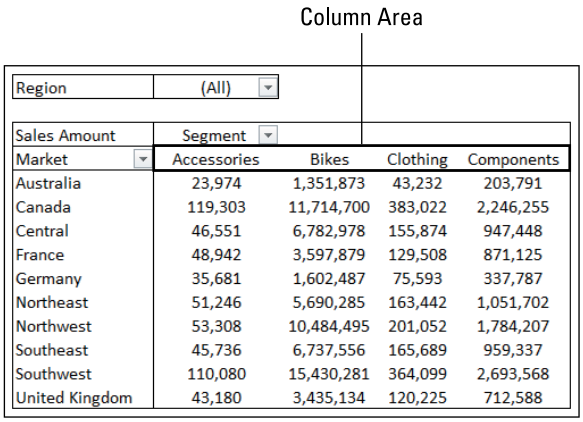


FIGURE 3-3: The column area of a pivot table gives you a column-oriented perspective.

Placing a data field into the column area displays the unique values from that field in a column-oriented perspective. The column area is ideal for creating a data matrix or showing trends over time.

Filter area

The *filter area* is an optional set of one or more drop-down lists at the top of the pivot table. In Figure 3-4, the filter area contains the Region field, and the pivot table is set to show all regions.

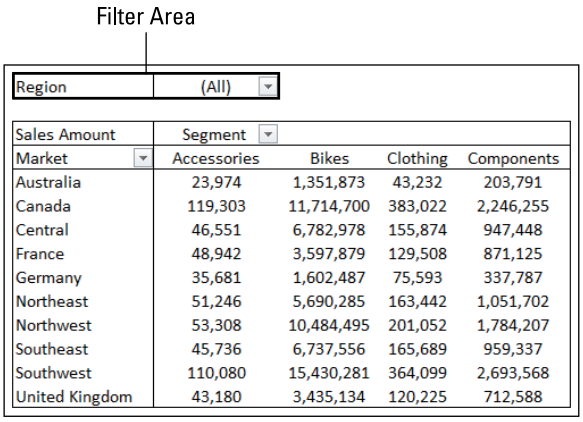


FIGURE 3-4: The filter area allows you to easily apply filters to the pivot table report.

Placing data fields into the filter area allows you to filter the entire pivot table based on your selections. The types of data fields that you might drop here include those that you want to isolate and focus on; for example, Region, Line of Business, and Employees.

Creating Your First Pivot Table

Now that you have a good understanding of the basic structure of a pivot table, it's time to try your hand at creating your first pivot table.

TIP

You can find the sample file for this chapter on this book's companion website.

Follow these steps:

1. **Click any single cell inside the *data source;* it's the table you use to feed the pivot table.**

   If you're following along, the data source would be the table found on the Sample Data tab.

2. **Select the Insert tab on the Ribbon and then click the PivotTable command (shown in Figure 3-5).**

   This step opens the Create PivotTable dialog box, as shown in Figure 3-6. As you can see, this dialog box asks you to specify the location of the source data and the place where you want to put the pivot table.

REMEMBER

Notice that in the Create PivotTable dialog box, Excel makes an attempt to fill in the range of your data for you. In most cases, Excel gets this right. However, always make sure that the correct range is selected.

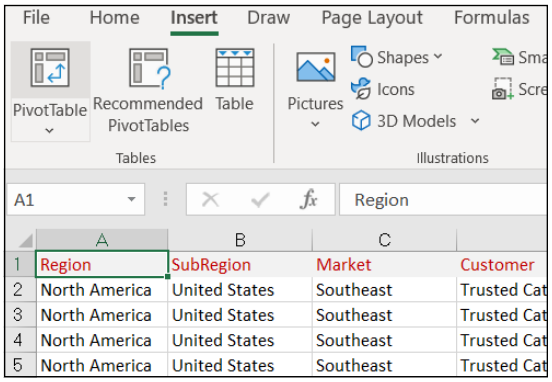


FIGURE 3-5: Start a pivot table via the Insert tab.

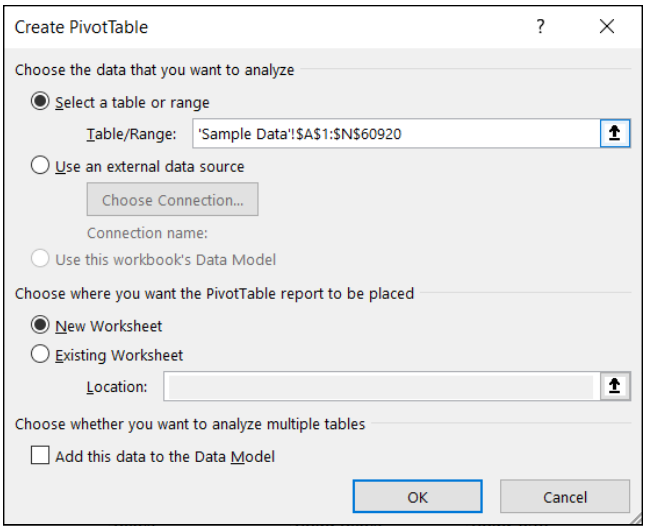


FIGURE 3-6: The Create PivotTable dialog box.

Also note in Figure 3-6 that the default location for a new pivot table is New Worksheet. This means your pivot table is placed in a new worksheet within the current workbook. You can change this by selecting the Existing Worksheet option and specifying the worksheet where you want the pivot table placed.

3. **Click OK.**

At this point, you have an empty pivot table report on a new worksheet. Next to the empty pivot table, you see the PivotTable Fields task pane, shown in Figure 3-7.

The idea here is to add the fields you need into the pivot table by using the four *drop zones* found in the PivotTable Field List: Filters, Columns, Rows, and Values. Pleasantly enough, these drop zones correspond to the four areas of the pivot table described at the beginning of this chapter.

TIP

If clicking the pivot table doesn't open the PivotTable Fields dialog box, you can manually open it by right-clicking anywhere inside the pivot table and selecting Show Field List.

Now, before you go wild and start dropping fields into the various drop zones, you should ask yourself two questions: "What am I measuring?" and "How do I want to see it?" The answers to these questions give you some guidance when determining which fields go where.

For your first pivot table report, measure the dollar sales by market. This automatically tells you that you need to work with the Sales Amount field and the Market field.

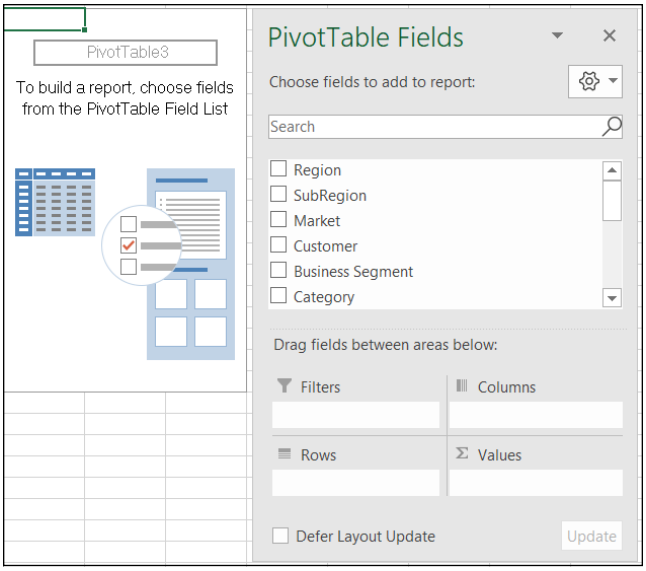


FIGURE 3-7: The PivotTable Fields task pane.

How do you want to see that? You want markets to be listed down the left side of the report and the sales amount to be calculated next to each market. Remembering the four areas of the pivot table, you need to add the Market field to the Rows drop zone and add the Sales Amount field to the Values drop zone.

4. **Select the Market check box in the list, as shown in Figure 3-8.**

Now that you have regions in the pivot table, it's time to add the dollar sales.

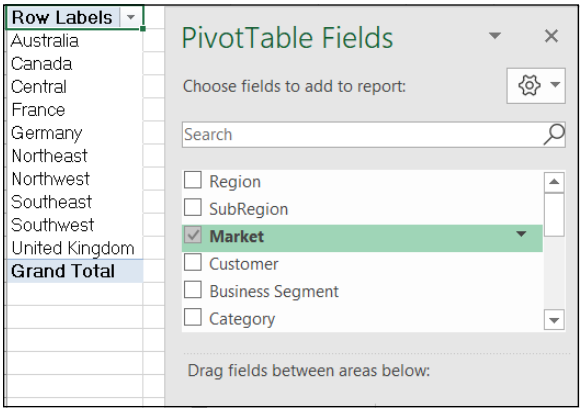


FIGURE 3-8: Select the Market check box.

5. **Select the Sales Amount check box in the list, as shown in Figure 3-9.**

TIP

Selecting a check box that is *non-numeric* (text or date) automatically places that field into the row area of the pivot table. Selecting a check box that is *numeric* automatically places that field in the values area of the pivot table.

What happens if you need fields in the other areas of the pivot table? Well, rather than select the field's check box, you can drag any field directly to the different drop zones.

One more thing: When you add fields to the drop zones, you may find it difficult to see all the fields in each drop zone. You can expand the PivotTable Fields dialog box by clicking and dragging the borders of the dialog box.

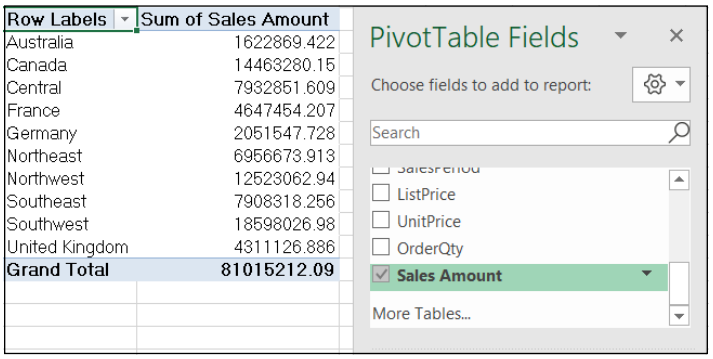


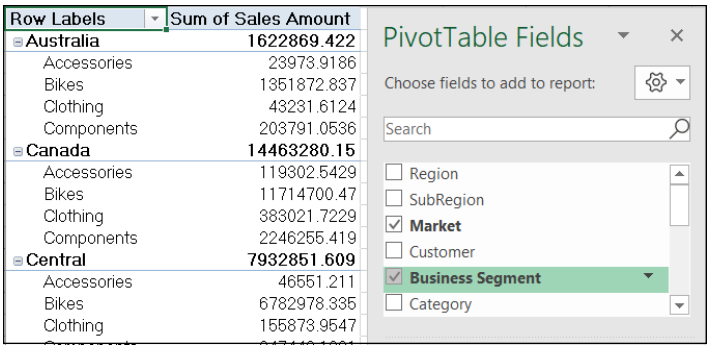
FIGURE 3-9: Add the Sales Amount field by selecting its check box.

As you can see, you have just analyzed the sales for each market in just five steps! That's an amazing feat, considering that you start with more than 60,000 rows of data. With a little formatting, this modest pivot table can become the starting point for a management report.

Changing and rearranging a pivot table

Now, here's the wonderful thing about pivot tables: You can add as many layers of analysis as made possible by the fields in the source data table. Say that you want to show the dollar sales that each market earned by business segment. Because the pivot table already contains the Market and Sales Amount fields, all you have to add is the Business Segment field.

So, simply click anywhere on the pivot table to reopen the PivotTable Fields task pane, and then select the Business Segment check box. Figure 3-10 illustrates what the pivot table should look like now.



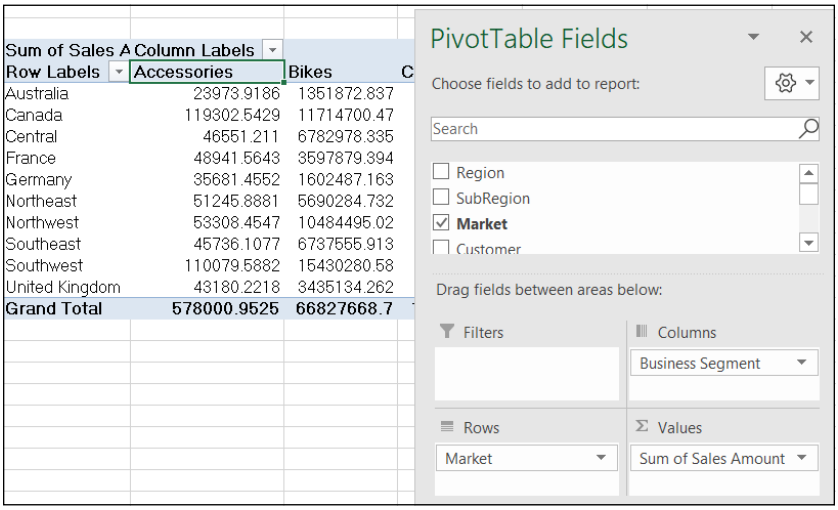
Row Labels	Sum of Sales Amount
Australia	1622869.422
Accessories	23973.9186
Bikes	1351872.837
Clothing	43231.6124
Components	203791.0536
Canada	14463280.15
Accessories	119302.5429
Bikes	11714700.47
Clothing	383021.7229
Components	2246255.419
Central	7932851.609
Accessories	46551.211
Bikes	6782978.335
Clothing	155873.9547

FIGURE 3-10: Adding a layer of analysis is as easy as bringing in another field.

TIP

If clicking the pivot table doesn't open the PivotTable Fields task pane, you can manually open it by right-clicking anywhere inside the pivot table and selecting Show Field List.

Imagine that your manager says that this layout doesn't work for them. They want to see business segments displayed across the top of the pivot table report. No problem: Simply drag the Business Segment field from the Rows drop zone to the Columns drop zone. As you can see in Figure 3-11, this instantly restructures the pivot table to your manager's specifications.



Sum of Sales A	Column Labels		C
Row Labels	Accessories	Bikes	
Australia	23973.9186	1351872.837	
Canada	119302.5429	11714700.47	
Central	46551.211	6782978.335	
France	48941.5643	3597879.394	
Germany	35681.4552	1602487.163	
Northeast	51245.8881	5690284.732	
Northwest	53308.4547	10484495.02	
Southeast	45736.1077	6737555.913	
Southwest	110079.5882	15430280.58	
United Kingdom	43180.2218	3435134.262	
Grand Total	578000.9525	66827668.7	

FIGURE 3-11: Your business segments are now column oriented.

Adding a report filter

Often, you're asked to produce reports for one particular region, market, or product. Rather than work hours and hours building separate reports for every possible

analysis scenario, you can leverage pivot tables to help create multiple views of the same data. For example, you can do so by creating a region filter in the pivot table.

Click anywhere on the pivot table to reopen the PivotTable Fields task pane, and then drag the Region field to the Filters drop zone. This adds a drop-down selector to the pivot table, shown in Figure 3-12 (cell B1). You can then use this selector to analyze one particular region at a time.

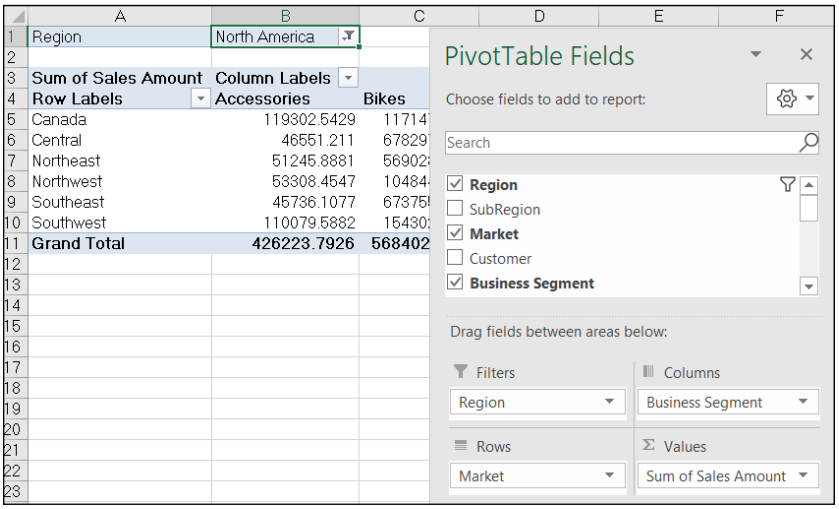


FIGURE 3-12: Adding Region to the Filters drop zone displays a Region drop-down list.

Keeping the pivot table fresh

In Hollywood, it's important to stay fresh and relevant. As boring as the pivot tables may seem, they'll eventually become the stars of your reports. So it's just as important to keep your pivot tables fresh and relevant.

As time goes by, your data may change and grow with newly added rows and columns. The action of updating your pivot table with these changes is *refreshing* your data.

The pivot table report can be refreshed by simply right-clicking inside the pivot table report and selecting Refresh, as shown in Figure 3-13.

Sometimes, *you're* the data source that feeds your pivot table changes in structure. For example, you may have added or deleted rows or columns from the data table. These types of changes affect the range of the data source, not just a few data items in the table.

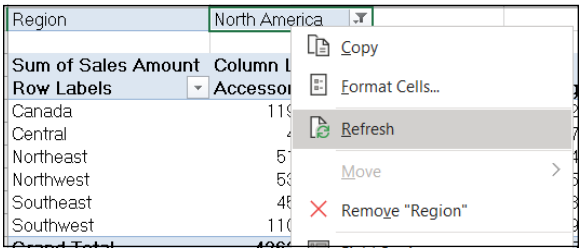


FIGURE 3-13: Refreshing the pivot table captures changes made to your data.

In these cases, performing a simple Refresh of the pivot table won't do. You have to update the range being captured by the pivot table. Here's how:

1. **Click anywhere inside the pivot table to select the PivotTable Analyze context tab on the Ribbon.**
2. **Click Change Data Source, as shown in Figure 3-14.**

   The Change PivotTable Data Source dialog box appears.
3. **Change the range selection to include any new rows or columns (see Figure 3-15).**
4. **Click OK to apply the change.**

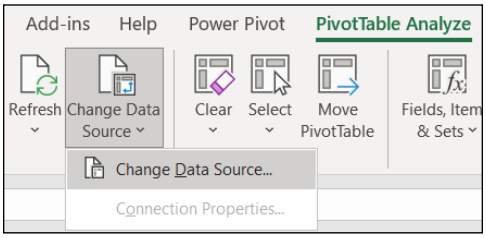


FIGURE 3-14: Changing the range that feeds the pivot table.

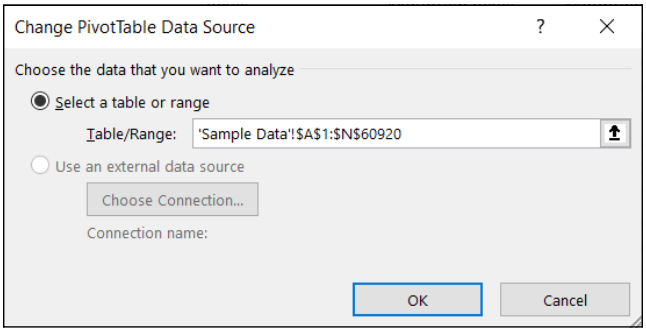


FIGURE 3-15: Select the new range that feeds the pivot table.

Customizing Pivot Table Reports

The pivot tables you create often need to be tweaked to get the look and feel you're looking for. In this section, I cover some of the options you can adjust to customize your pivot tables to suit your reporting needs.

Changing the pivot table layout

Excel gives you a choice in the layout of the data in a pivot table. The three layouts, shown side by side in Figure 3-16, are the Compact Form, Outline Form, and Tabular Form. Although no layout stands out as better than the others, I prefer using the Tabular Form layout because it seems easiest to read and it's the layout that most people who have seen pivot tables are used to.

Compact Form Layout

Row Labels	Sales
Australia	**1622869.422**
Accessories	23973.9186
Bikes	1351872.837
Clothing	43231.6124
Components	203791.0536
Canada	**14463280.15**
Accessories	119302.5429
Bikes	11714700.47
Clothing	383021.7229
Components	2246255.419
Central	**7932851.609**
Accessories	46551.211
Bikes	6782978.335
Clothing	155873.9547
Components	947448.1091
France	**4647454.207**
Accessories	48941.5643
Bikes	3597879.394
Clothing	129508.0548
Components	871125.1938
Germany	**2051547.729**
Accessories	35681.4552
Bikes	1602487.163
Clothing	75592.5945
Components	337786.516

Outline Form Layout

Market	Segment	Sales
Australia		**1622869.422**
	Accessories	23973.9186
	Bikes	1351872.837
	Clothing	43231.6124
	Components	203791.0536
Canada		**14463280.15**
	Accessories	119302.5429
	Bikes	11714700.47
	Clothing	383021.7229
	Components	2246255.419
Central		**7932851.609**
	Accessories	46551.211
	Bikes	6782978.335
	Clothing	155873.9547
	Components	947448.1091
France		**4647454.207**
	Accessories	48941.5643
	Bikes	3597879.394
	Clothing	129508.0548
	Components	871125.1938
Germany		**2051547.729**
	Accessories	35681.4552
	Bikes	1602487.163
	Clothing	75592.5945
	Components	337786.516

Tabular Form Layout

Market	Segment	Sales
Australia	Accessories	23973.9186
	Bikes	1351872.837
	Clothing	43231.6124
	Components	203791.0536
Australia Total		**1622869.422**
Canada	Accessories	119302.5429
	Bikes	11714700.47
	Clothing	383021.7229
	Components	2246255.419
Canada Total		**14463280.15**
Central	Accessories	46551.211
	Bikes	6782978.335
	Clothing	155873.9547
	Components	947448.1091
Central Total		**7932851.609**
France	Accessories	48941.5643
	Bikes	3597879.394
	Clothing	129508.0548
	Components	871125.1938
France Total		**4647454.207**
Germany	Accessories	35681.4552
	Bikes	1602487.163
	Clothing	75592.5945
	Components	337786.516
Germany Total		**2051547.729**

FIGURE 3-16: The three layouts for a pivot table report.

The layout you choose affects not only the look and feel of your reporting mechanisms but also, possibly, the way you build and interact with any reporting models based on your pivot tables.

Changing the layout of a pivot table is easy. Follow these steps:

1. **Click anywhere inside the pivot table to select the Design context tab on the Ribbon.**
2. **Click the Report Layout icon and choose the layout you like (see Figure 3-17).**

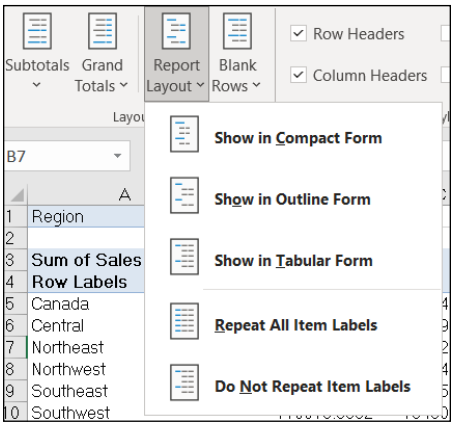


FIGURE 3-17: Changing the layout of the pivot table.

Customizing field names

Notice that every field in the pivot table has a name. The fields in the row, column, and filter areas inherit their names from the data labels in the source table. The fields in the values area are given a name, such as Sum of Sales Amount.

Sometimes you might prefer the name Total Sales instead of the unattractive default name, such as Sum of Sales Amount. In these situations, the ability to change your field names is handy. To change a field name, follow these steps:

1. **Right-click any value within the target field.**

   For example, if you want to change the name of the field Sum of Sales Amount, right-click the field name, or any value under that field.

2. **Select Value Field Settings, as shown in Figure 3-18.**

   The Value Field Settings dialog box appears.

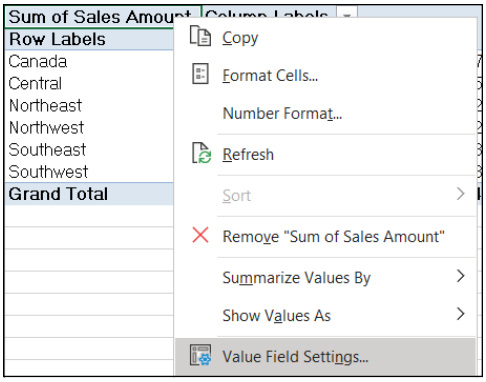


FIGURE 3-18: Right-click the target field to select the Value Field Settings option.

3. **Enter the new name in the Custom Name input box, shown in Figure 3-19.**
4. **Click OK to apply the change.**

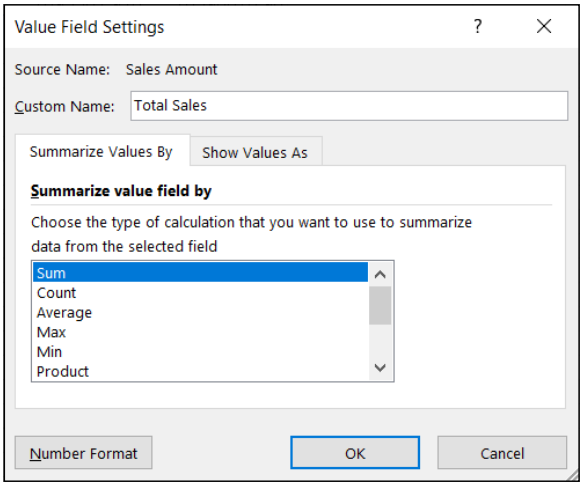


FIGURE 3-19: Use the Custom Name input box to change the name of the field.

TIP

If you use the name of the data label used in the source table, you receive an error. For example, if you rename Sum of Sales Amount as Sales Amount, you see an error message because there's already a Sales Amount field in the source data table. Well, this is kind of lame, especially if Sales Amount is exactly what you want to name the field in your pivot table.

To get around this, you can name the field and add a space to the end of the name. Excel considers Sales Amount (followed by a space) to be different from Sales Amount. This way, you can use the name you want and no one will notice that it's any different.

Applying numeric formats to data fields

Numbers in pivot tables can be formatted to fit your needs; that is, formatted as currency, percentage, or number. You can easily control the numeric formatting of a field using the Value Field Settings dialog box. Here's how:

1. **Right-click the target field's name or any value within the target field.**

   For example, if you want to change the format of the values in the Sales Amount field, right-click the field name or any value under that field.

2. **Select Value Field Settings.**

   The Value Field Settings dialog box appears.

3. **Click the Number Format button.**

   The Format Cells dialog box opens.

4. **Apply the number format you desire, just as you typically would on your spreadsheet.**

5. **Click OK to apply the changes.**

   After you set the formatting for a field, the applied formatting persists, even if you refresh or rearrange the pivot table.

Changing summary calculations

When creating the pivot table report, Excel, by default, summarizes your data by either counting or summing the items. Rather than choose `Sum` or `Count`, you might want to choose functions, such as `Average`, `Min`, `Max`, for example. In all, 11 options are available, including

- `Sum`: Adds all numeric data.
- `Count`: Counts all data items within a given field, including numeric-, text-, and date-formatted cells.
- `Average`: Calculates an average for the target data items.
- `Max`: Displays the largest value in the target data items.
- `Min`: Displays the smallest value in the target data items.
- `Product`: Multiplies all target data items together.
- `Count Numbers`: Counts only the numeric cells in the target data items.
- `StdDevP` and `StdDev`: Calculates the standard deviation for the target data items. Use `StdDevP` if your dataset contains the complete population. Use `StdDev` if your dataset contains a sample of the population.
- `VarP` and `Var`: Calculates the statistical variance for the target data items. Use `VarP` if your data contains a complete population. If your data contains only a sampling of the complete population, use `Var` to estimate the variance.

You can easily change the summary calculation for any given field by taking the following actions:

1. **Right-click any value within the target field.**
2. **Select Value Field Settings.**

   The Value Field Settings dialog box appears.

3. **Choose the type of calculation you want to use from the list of calculations (see Figure 3-20).**

4. **Click OK to apply the changes.**

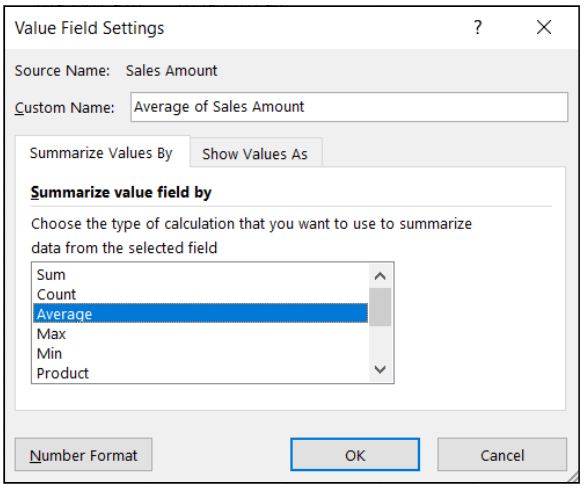


FIGURE 3-20: Changing the type of summary calculation used in a field.

REMEMBER

Did you know that a single blank cell causes Excel to count instead of sum? That's right: If all cells in a column contain numeric data, Excel chooses `Sum`. If only one cell is either blank or contains text, Excel chooses `Count`.

Be sure to pay attention to the fields that you place into the values area of the pivot table. If the field name starts with *Count Of*, Excel is counting the items in the field instead of summing the values.

Suppressing subtotals

Notice that every time you add a field to the pivot table, Excel adds a subtotal for that field. At times, however, the inclusion of subtotals either doesn't make sense or simply hinders a clear view of the pivot table report. For example, Figure 3-21 shows a pivot table in which the subtotals inundate the report with totals that hide the real data you're trying to report.

	A	B	C	D	E
1	Region	SubRegion	Market	Business Segment	Sum of Sales Amount
2	North America	United States	Central	Accessories	46,551
3				Bikes	6,782,978
4				Clothing	155,874
5				Components	947,448
6			Central Total		7,932,852
7			Northeast	Accessories	51,246
8				Bikes	5,690,285
9				Clothing	163,442
10				Components	1,051,702
11			Northeast Total		6,956,674
12			Northwest	Accessories	53,308
13				Bikes	10,484,495
14				Clothing	201,052
15				Components	1,784,207
16			Northwest Total		12,523,063
17			Southeast	Accessories	45,736
18				Bikes	6,737,556
19				Clothing	165,689
20				Components	959,337
21			Southeast Total		7,908,318
22			Southwest	Accessories	110,080
23				Bikes	15,430,281
24				Clothing	364,099
25				Components	2,693,568
26			Southwest Total		18,598,027
27		United States Total			53,918,934
28	North America Total				53,918,934

FIGURE 3-21: Subtotals sometimes muddle the data you're trying to show.

Removing all subtotals at one time

You can remove all subtotals at one time by taking these actions:

1. **Click anywhere inside the pivot table to select the Design context tab on the Ribbon.**
2. **Click the Subtotals icon and select Do Not Show Subtotals, as shown in Figure 3-22.**

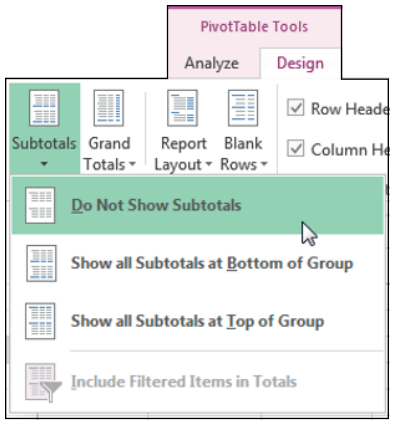


FIGURE 3-22: Use the Do Not Show Subtotals option to remove all subtotals at one time.

As you can see in Figure 3-23, the same report without subtotals is much more pleasant to review.

	A	B	C	D	E
1	Region	SubRegion	Market	Business Segment	Sum of Sales Amount
2	North America	United States	Central	Accessories	46,551
3				Bikes	6,782,978
4				Clothing	155,874
5				Components	947,448
6			Northeast	Accessories	51,246
7				Bikes	5,690,285
8				Clothing	163,442
9				Components	1,051,702
10			Northwest	Accessories	53,308
11				Bikes	10,484,495
12				Clothing	201,052
13				Components	1,784,207
14			Southeast	Accessories	45,736
15				Bikes	6,737,556
16				Clothing	165,689
17				Components	959,337
18			Southwest	Accessories	110,080
19				Bikes	15,430,281
20				Clothing	364,099
21				Components	2,693,568
22	Grand Total				53,918,934

FIGURE 3-23: The report shown in Figure 3-21, without subtotals.

Removing the subtotals for only one field

Maybe you want to remove the subtotals for only one field? In such a case, you can take the following actions:

1. **Right-click any value within the target field.**
2. **Select Field Settings.**

   The Field Settings dialog box appears.
3. **Choose the None option under Subtotals, as shown in Figure 3-24.**
4. **Click OK to apply the changes.**

Removing grand totals

In certain instances, you may want to remove the grand totals from the pivot table. Follow these steps:

1. **Right-click anywhere on the pivot table.**
2. **Select PivotTable Options.**

   The PivotTable Options dialog box appears.

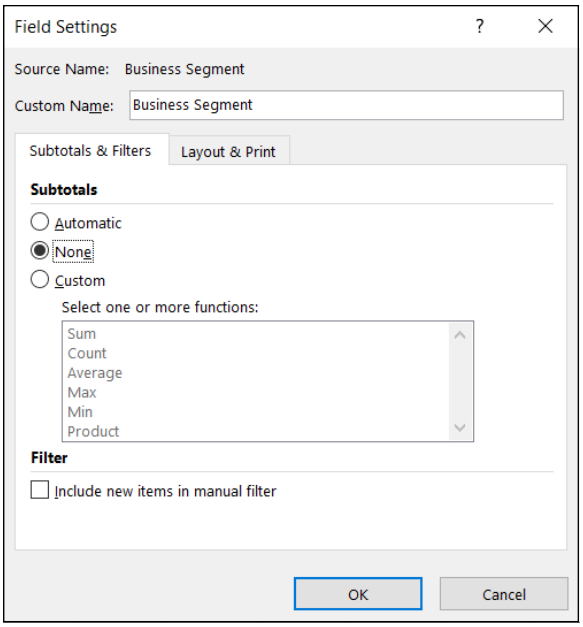


FIGURE 3-24: Choose the None option to remove subtotals for one field.

3. **Click the Totals & Filters tab.**
4. **Click the Show Grand Totals for Rows check box to deselect it.**
5. **Click the Show Grand Totals for Columns check box to deselect it.**
6. **Click OK to apply your changes.**

Showing and hiding data items

A pivot table summarizes and displays all records in a source data table. In certain situations, however, you may want to inhibit certain data items from being included in the pivot table summary. In these situations, you can choose to hide a data item.

In terms of pivot tables, hiding doesn't mean simply preventing the data item from being shown on the report. Hiding a data item also prevents it from being factored into the summary calculations.

In the pivot table illustrated in Figure 3-25, I show sales amounts for all business segments by market. In this example, I want to show totals without taking sales from the Bikes segment into consideration. In other words, I want to hide the Bikes segment.

	A	B	C
1	Market	Business Segment	Sum of Sales Amount
2	Australia	Accessories	$23,974
3		Bikes	$1,351,873
4		Clothing	$43,232
5		Components	$203,791
6	Australia Total		$1,622,869
7	Canada	Accessories	$119,303
8		Bikes	$11,714,700
9		Clothing	$383,022
10		Components	$2,246,255
11	Canada Total		$14,463,280
12	Central	Accessories	$46,551
13		Bikes	$6,782,978
14		Clothing	$155,874
15		Components	$947,448
16	Central Total		$7,932,852
17	France	Accessories	$48,942

FIGURE 3-25: To remove Bikes from this analysis . . .

You can hide the Bikes Business Segment by clicking the Business Segment drop-down arrow and deselecting the Bikes check box, as shown in Figure 3-26.

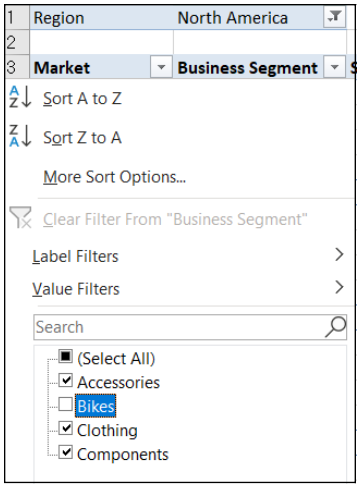


FIGURE 3-26: . . . deselect the Bikes check box.

After you click OK to close the selection box, the pivot table instantly recalculates, leaving out the Bikes segment. As you can see in Figure 3-27, the Market total sales now reflect the sales without Bikes.

You can just as quickly reinstate all hidden data items for the field. You simply click the Business Segment drop-down arrow and click the Select All check box, as shown in Figure 3-28.

	A	B	C
1	Market	Business Segment	Sum of Sales Amount
2	Australia	Accessories	$23,974
3		Clothing	$43,232
4		Components	$203,791
5	Australia Total		$270,997
6	Canada	Accessories	$119,303
7		Clothing	$383,022
8		Components	$2,246,255
9	Canada Total		$2,748,580
10	Central	Accessories	$46,551
11		Clothing	$155,874
12		Components	$947,448
13	Central Total		$1,149,873

FIGURE 3-27: The analysis from Figure 3-25, without the Bikes segment.

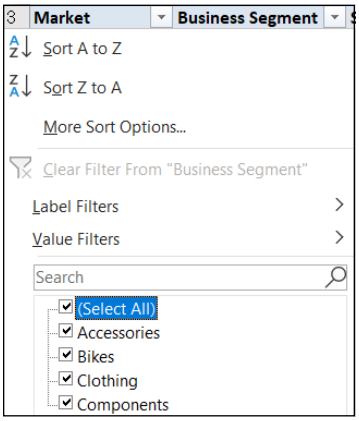


FIGURE 3-28: Clicking the Select All check box forces all data items in that field to become unhidden.

Hiding or showing items without data

By default, the pivot table shows only data items that have data. This inherent behavior may cause unintended problems for your data analysis.

Look at Figure 3-29, which shows a pivot table with the SalesPeriod field in the row area and the Region field in the filter area. Note that the Region field is set to (All) and that every sales period appears in the report.

If you choose Europe in the filter area, only a portion of all the sales periods is shown (see Figure 3-30). The pivot table shows only those sales periods that apply to the Europe region.

From a reporting perspective, it isn't ideal if half the year's data disappears every time customers select Europe.

	A	B
1	Region	(All)
2		
3	SalesPeriod	Sum of Sales Amount
4	01/01/08	$713,230
5	02/01/08	$1,900,797
6	03/01/08	$1,455,282
7	04/01/08	$883,011
8	05/01/08	$2,269,722
9	06/01/08	$1,137,250
10	07/01/08	$2,411,569
11	08/01/08	$3,615,926

FIGURE 3-29: All sales periods are showing.

	A	B
1	Region	Europe
2		
3	SalesPeriod	Sum of Sales Amount
4	07/01/08	$180,241
5	08/01/08	$448,373
6	09/01/08	$373,122
7	10/01/08	$119,384
8	11/01/08	$330,026
9	12/01/08	$254,011
10	01/01/09	$71,313
11	02/01/09	$264,487

FIGURE 3-30: Filtering for the Europe region causes certain sales periods to disappear.

Here's how you can prevent Excel from hiding pivot items without data:

1. **Right-click any value within the target field.**

   In this example, the target field is the SalesPeriod field.

2. **Select Field Settings.**

   The Field Settings dialog box appears.

3. **Select the Layout & Print tab in the Field Settings dialog box.**

4. **Select the Show Items with No Data option, as shown in Figure 3-31.**

5. **Click OK to apply the change.**

As you can see in Figure 3-32, after you choose the Show Items with No Data option, all sales periods appear whether the selected region had sales that period or not.

Now that you're confident that the structure of the pivot table is locked, you can use it to feed charts and other components on your report.

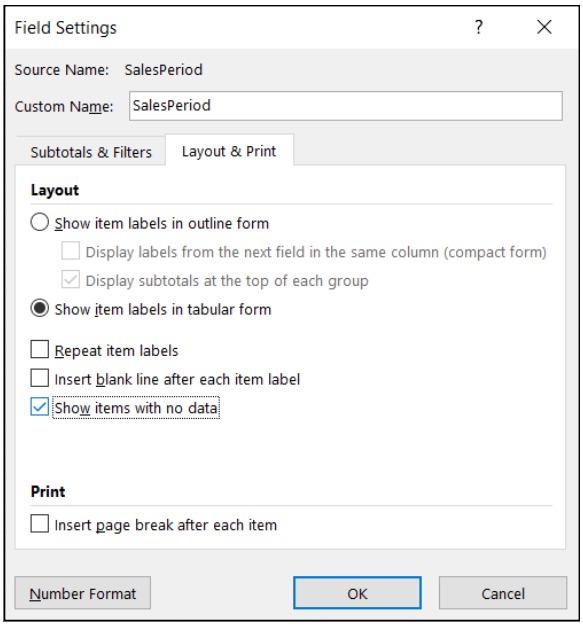
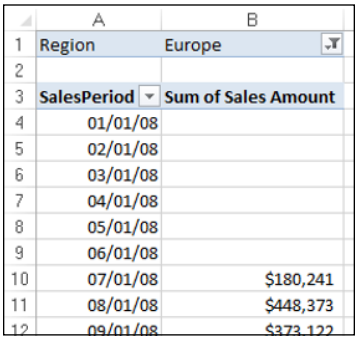


FIGURE 3-31: Select the Show Items with No Data option to force Excel to display all data items.



	A	B
1	Region	Europe
2		
3	SalesPeriod	Sum of Sales Amount
4	01/01/08	
5	02/01/08	
6	03/01/08	
7	04/01/08	
8	05/01/08	
9	06/01/08	
10	07/01/08	$180,241
11	08/01/08	$448,373
12	09/01/08	$373,122

FIGURE 3-32: All sales periods are now displayed, even if there is no data to be shown.

Sorting the pivot table

By default, items in each pivot field are sorted in ascending sequence based on the item name. Excel gives you the freedom to change the sort order of the items in the pivot table.

Like many actions you can perform in Excel, you have lots of different ways to sort data within a pivot table. The easiest way is to apply the sort directly in the pivot table. Here's how:

1. **Right-click any value within the *target field* — the field you need to sort.**

   In the example shown in Figure 3-33, you want to sort by Sales Amount.

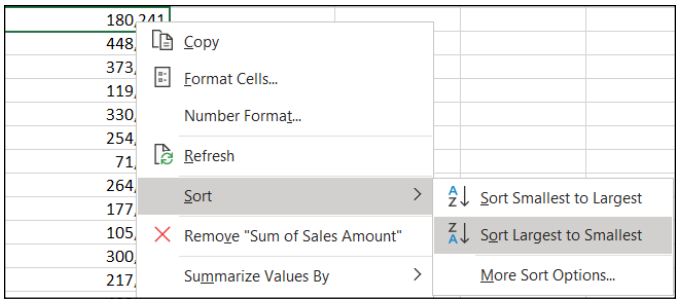


FIGURE 3-33: Applying a sort to a pivot table field.

2. **Select Sort and then select the sort direction.**

   The changes take effect immediately and persist while you work with the pivot table.

Understanding Slicers

Slicers allow you to filter your pivot table in a way that's similar to the way Filter fields filter a pivot table. The difference is that slicers offer a user-friendly interface, enabling you to better manage the filter state of your pivot table reports.

As useful as Filter fields are, they have always had a couple of drawbacks.

First of all, Filter fields are not cascading filters — the filters don't work together to limit selections when needed. For example, in Figure 3-34, you can see that the Region filter is set to the North America region. However, the Market filter still allows you to select markets that are clearly not in the North America region (Germany, for example). Because the Market filter is not in any way limited based on the Region Filter field, you have the annoying possibility of selecting a market that could yield no data because it's not in the North America region.

Another drawback is that Filter fields don't provide an easy way to tell what exactly is being filtered when you select multiple items. In Figure 3-35, you can see an example. The Market filter has been limited to four markets. However, notice that the Market filter value shows `(Multiple Items)`. By default, Filter fields show `(Multiple Items)` when you select more than one item. The only way to tell what has been selected is to click the drop-down menu. You can imagine the confusion on a printed version of this report, in which you can't click down to see which data items make up the numbers on the page.

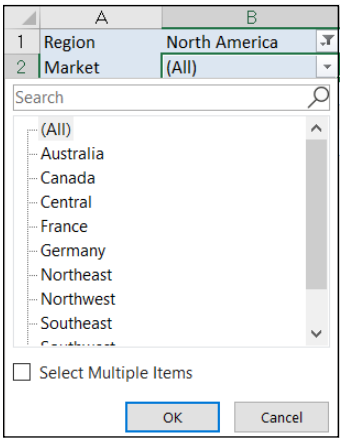


FIGURE 3-34: Default pivot table Filter fields do not work together to limit filter selections.

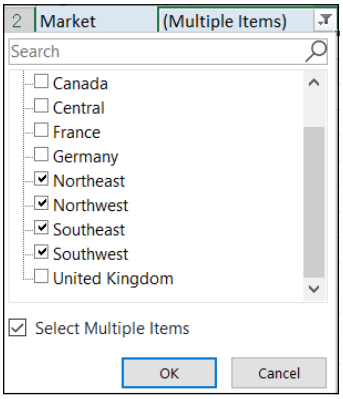
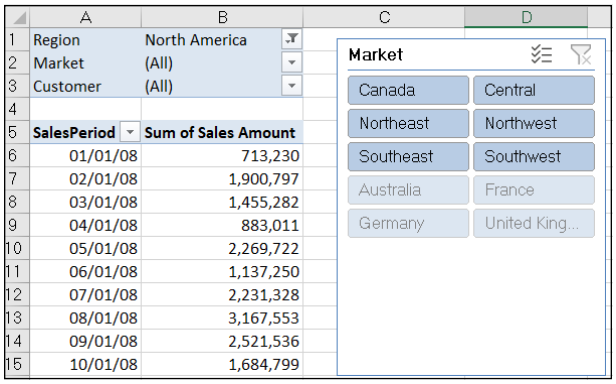


FIGURE 3-35: Filter fields show the phrase `(Multiple Items)` whenever multiple selections are made.

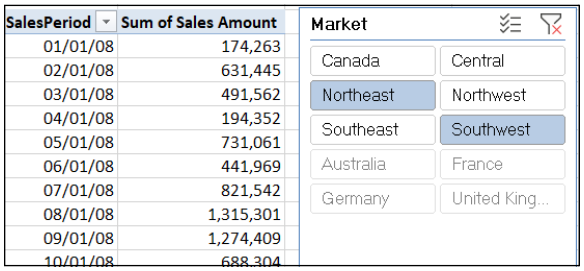
By contrast, slicers don't have these issues. Slicers respond to one another. As you can see in Figure 3-36, the Market slicer visibly highlights the relevant markets when the North America region is selected. The rest of the markets are muted, signaling that they are not part of the selected region.

When selecting multiple items in a slicer, you can easily see that multiple items have been chosen. In Figure 3-37, you can see that the pivot table is being filtered by the Northeast and Southwest markets.



	A	B	C	D
1	Region	North America		
2	Market	(All)		
3	Customer	(All)		
4				
5	SalesPeriod	Sum of Sales Amount		
6	01/01/08	713,230		
7	02/01/08	1,900,797		
8	03/01/08	1,455,282		
9	04/01/08	883,011		
10	05/01/08	2,269,722		
11	06/01/08	1,137,250		
12	07/01/08	2,231,328		
13	08/01/08	3,167,553		
14	09/01/08	2,521,536		
15	10/01/08	1,684,799		

FIGURE 3-36: Slicers work together to show you relevant data items based on your selection.



SalesPeriod	Sum of Sales Amount
01/01/08	174,263
02/01/08	631,445
03/01/08	491,562
04/01/08	194,352
05/01/08	731,061
06/01/08	441,969
07/01/08	821,542
08/01/08	1,315,301
09/01/08	1,274,409
10/01/08	688,304

FIGURE 3-37: Slicers do a better job at displaying multiple item selections.

Creating a Standard Slicer

Enough talk. It's time to create your first slicer. Just follow these steps:

1. **Place the cursor anywhere inside the pivot table, and then go up to the Ribbon and click the PivotTable Analyze tab. There, click the Insert Slicer icon, shown in Figure 3-38.**

   This step opens the Insert Slicers dialog box, shown in Figure 3-39. Select the fields you want to filter. In this example, the Region and Market slicers are created.

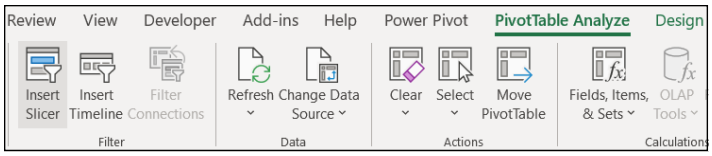


FIGURE 3-38: Inserting a slicer.

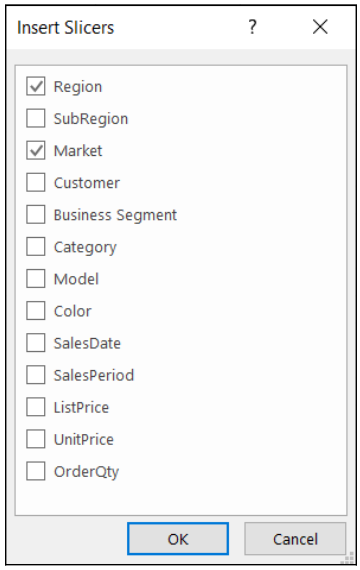


FIGURE 3-39: Select the fields for which you want slicers created.

2. After the slicers are created, simply click the filter values to filter the pivot table.

As you can see in Figure 3-40, clicking Midwest in the Region slicer not only filters the pivot table, but the Market slicer also responds by highlighting the markets that belong to the Midwest region.

You can also select multiple values by holding down the Ctrl key on the keyboard while selecting the needed filters. In Figure 3-41, I held down the Ctrl key while selecting Baltimore, California, Charlotte, and Chicago. This highlights not only the selected markets in the Market slicer but also their associated regions in the Region slicer.

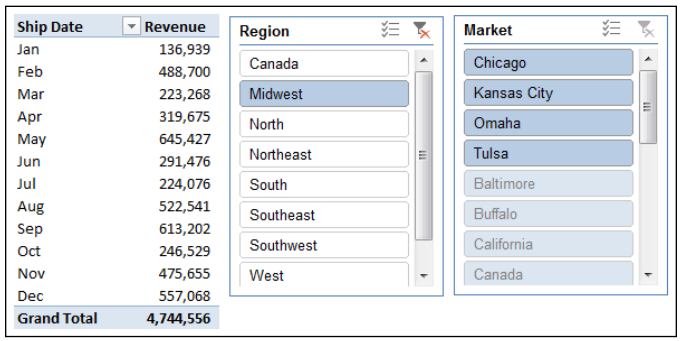
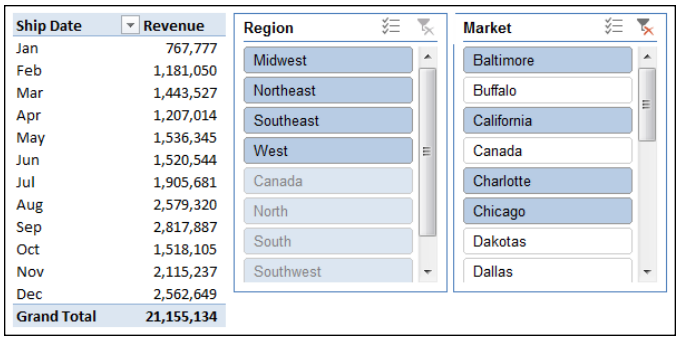


FIGURE 3-40: Select the fields you want filtered using slicers.



Ship Date	Revenue
Jan	767,777
Feb	1,181,050
Mar	1,443,527
Apr	1,207,014
May	1,536,345
Jun	1,520,544
Jul	1,905,681
Aug	2,579,320
Sep	2,817,887
Oct	1,518,105
Nov	2,115,237
Dec	2,562,649
Grand Total	21,155,134

FIGURE 3-41: The fact that you can see the current filter state gives slicers a unique advantage over Filter fields.

TIP

To clear the filtering on a slicer, simply click the Clear Filter icon on the target slicer, as shown in Figure 3-42.

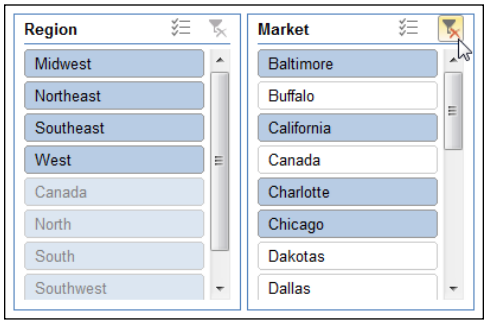


FIGURE 3-42: Clearing the filters on a slicer.

Getting Fancy with Slicer Customizations

The following sections cover a few formatting adjustments you can make to your slicers.

Size and placement

A slicer behaves like a standard Excel shape object in that you can move it around and adjust its size by clicking it and dragging its position points (see Figure 3-43).

You can also right-click the slicer and select Size and Properties. This brings up the Format Slicer pane (see Figure 3-44), allowing you to adjust the size of the slicer, how the slicer should behave when cells are shifted, and whether the slicer should appear on a printed copy of your report.

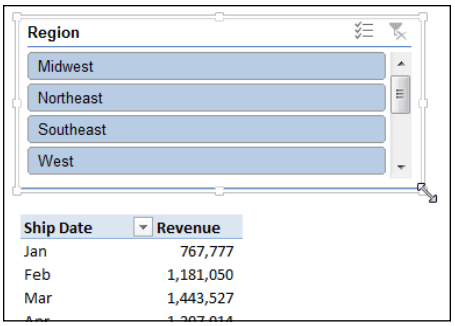


FIGURE 3-43: Adjust the slicer size and placement by dragging its position points.

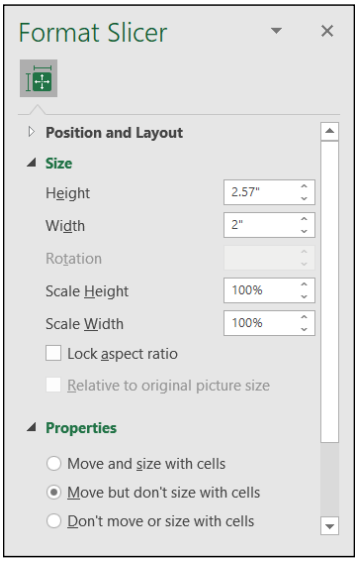


FIGURE 3-44: The Format Slicer pane offers more control over how the slicer behaves in relation to the worksheet it's on.

Data item columns

By default, all slicers are created with one column of data items. You can change this number by right-clicking the slicer and selecting Size and Properties. This opens the Format Slicer pane. Under the Position and Layout section, you can specify the number of columns in the slicer. Adjusting the number to 2, as shown in Figure 3-45, forces the data items to be displayed in two columns, adjusting the number to 3 forces the data items to be displayed in three columns, and so on.

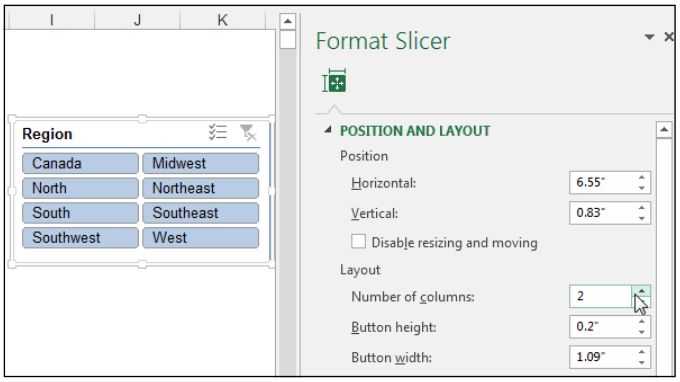


FIGURE 3-45: Adjust the Number of Columns property to display the slicer data items in more than one column.

Miscellaneous slicer settings

Right-clicking the slicer and selecting Slicer Settings opens the Slicer Settings dialog box, shown in Figure 3-46. Using this dialog box, you can control the look of the slicer's header, how the slicer is sorted, and how filtered items are handled.

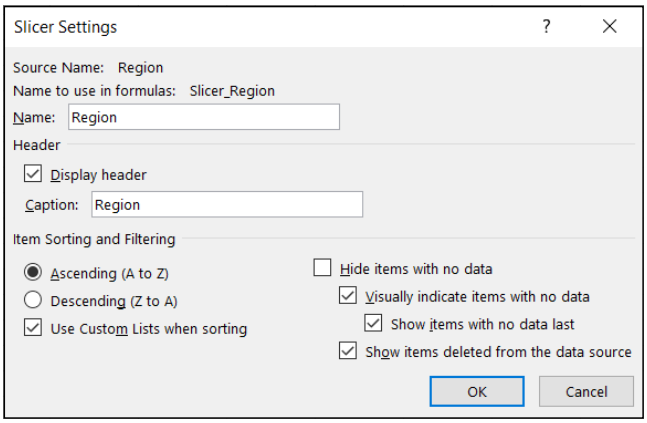


FIGURE 3-46: The Slicer Settings dialog box.

Controlling Multiple Pivot Tables with One Slicer

Another advantage you gain with slicers is that each slicer can be tied to more than one pivot table; that is to say, any filter you apply to your slicer can be applied to multiple pivot tables.

To connect the slicer to more than one pivot table, simply right-click the slicer and select Report Connections. This opens the Report Connections dialog box, shown in Figure 3-47. Place a check mark next to any pivot table that you want to filter using the current slicer.

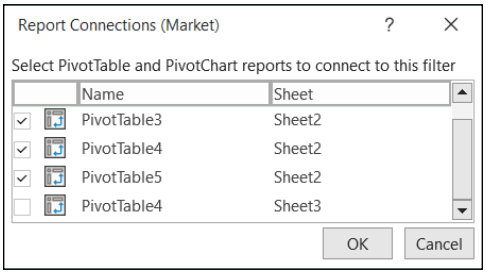


FIGURE 3-47: Choose the pivot tables to be filtered by this slicer.

At this point, any filter you apply to the slicer is applied to all connected pivot tables. Controlling the filter state of multiple pivot tables is a powerful feature, especially in reports that run on multiple pivot tables.

Creating a Timeline Slicer

The Timeline slicer works in the same way a standard slicer does, in that it lets you filter a pivot table using a visual selection mechanism rather than the old Filter fields. The difference is that the Timeline slicer is designed to work exclusively with date fields, providing an excellent visual method to filter and group the dates in the pivot table.

To create a Timeline slicer, the pivot table must contain a field where *all* data is formatted as a date. It's not enough to have a column of data that contains a few dates. All values in the date field must be a valid date and formatted as such.

To create a Timeline slicer, follow these steps:

1. **Place the cursor anywhere inside the pivot table, and then click the PivotTable Analyze tab on the Ribbon. There, click the Insert Timeline command.**

   The Insert Timelines dialog box, shown in Figure 3-48, appears, showing you all available date fields in the chosen pivot table.

2. **In the Insert Timelines dialog box, select the date fields for which you want to create the timeline.**

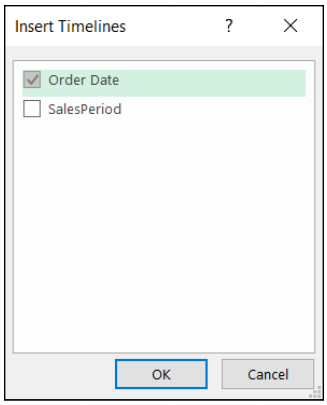


FIGURE 3-48: Select the date fields for which you want slicers created.

After the Timeline slicer is created, you can filter the data in the pivot table and pivot chart, using this dynamic data-selection mechanism. Figure 3-49 demonstrates how selecting Mar, Apr, and May in the Timeline slicer automatically filters the pivot chart.

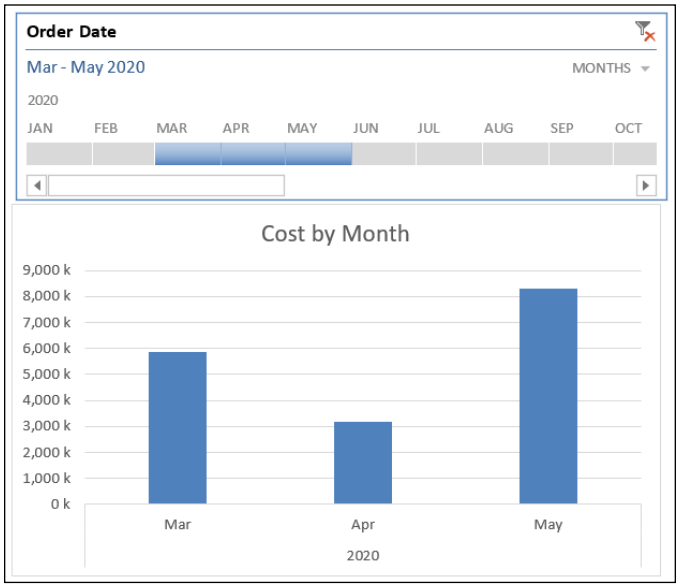


FIGURE 3-49: Click a date selection to filter the pivot table or pivot chart.

Figure 3-50 illustrates how you can expand the slicer range with the mouse to include a wider range of dates in your filtered numbers.

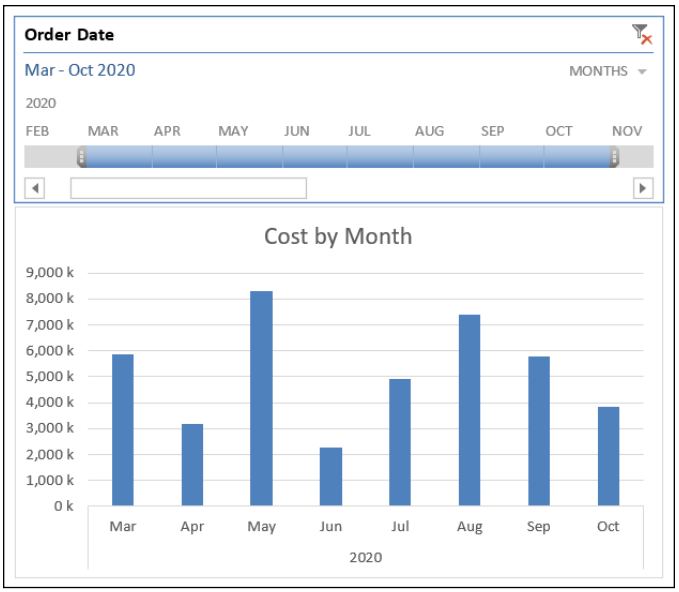


FIGURE 3-50: You can expand the range on the Timeline slicer to include more data in the filtered numbers.

Want to quickly filter the pivot table by quarters? Well, that's easy with a Timeline slicer. Simply click the time period drop-down menu and select Quarters. As you can see in Figure 3-51, you can also switch to Years or Days, if needed.

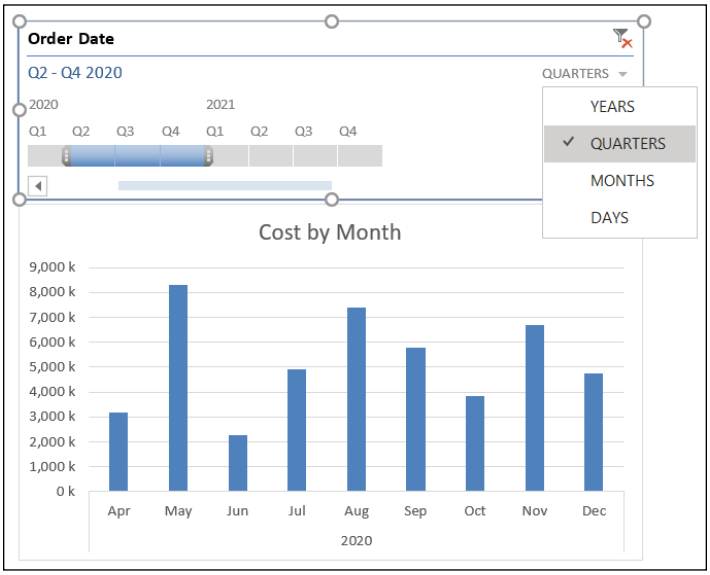


FIGURE 3-51: Quickly switch among Quarters, Years, Months, and Days.

IN THIS CHAPTER

- Importing from relational databases
- Importing from flat files
- Importing data from other data sources
- Refreshing and managing external data connections

Chapter 4

Using External Data with Power Pivot

In Chapter 2, I start an exploration of Power Pivot by showing you how to load the data already contained within the workbook you're working on. But as you discover in this chapter, you're not limited to using only the data that already exists in your Excel workbook.

Power Pivot has the ability to reach outside the workbook and import data found in external data sources. Indeed, what makes Power Pivot powerful is its ability to consolidate data from disparate data sources and build relationships between them. You can theoretically create a Power Pivot data model that contains some data from a SQL Server table, some data from a Microsoft Access database, and even data from a one-off text file.

In this chapter, I help you continue your journey by taking a closer look at the mechanics of importing external data into your Power Pivot data models.

REMEMBER

This chapter has no associated sample file. But don't worry: You can easily translate the information found here to your own data sources.

Loading Data from Relational Databases

One of the more common data sources used by Excel analysts is the relational database. It's not difficult to find an analyst who frequently uses data from Microsoft Access, SQL Server, or Oracle databases. In this section, I walk you through the steps for loading data from external database systems.

Loading data from SQL Server

SQL Server databases are some of the most commonly used for the storing of enterprise-level data. Most SQL Server databases are managed and maintained by the IT department. To connect to a SQL Server database, you have to work with your IT department to obtain Read access to the database you're trying to pull from.

After you have access to the database, open the Power Pivot window (select PowerPivot➪Manage from the Excel Ribbon) and then click the From Other Sources command button on the Home tab. This opens the Table Import Wizard dialog box, shown in Figure 4-1. There, select the Microsoft SQL Server option and then click the Next button.

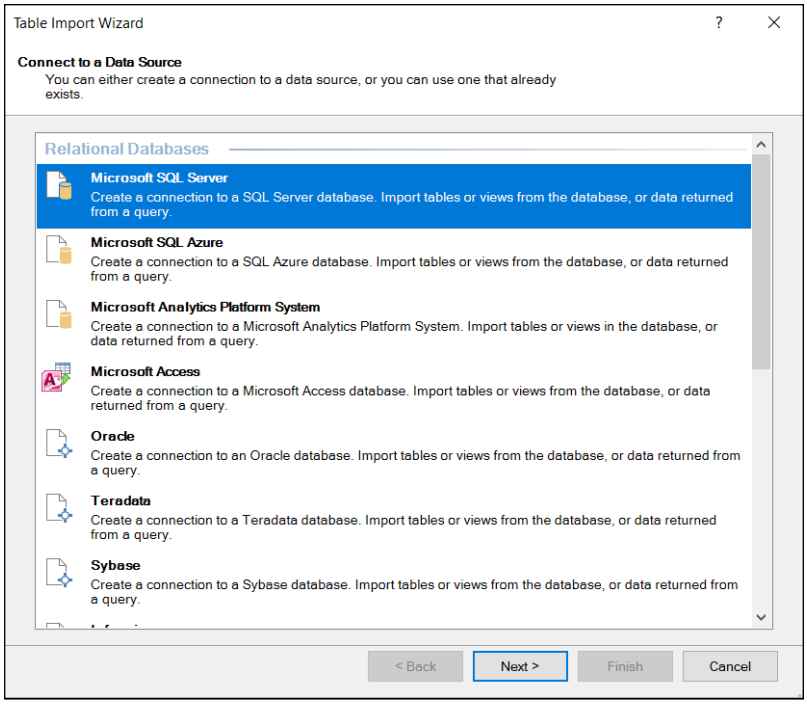


FIGURE 4-1: Open the Table Import Wizard and select Microsoft SQL Server.

The Table Import Wizard now asks for all the information it needs to connect to your database (see Figure 4-2). On this screen, you need to provide the information for the options described in this list:

- **Friendly Connection Name:** The Friendly Name field allows you to specify your own name for the external source. You typically enter a name that is descriptive and easy to read.
- **Server Name:** This is the name of the server that contains the database you're trying to connect to. You get this from your IT department when you gain access. (Your server name will be different from the one shown in Figure 4-2.)
- **Log On to the Server:** These are your login credentials. Depending on how your IT department gives you access, select either Windows Authentication or SQL Server Authentication. Windows Authentication essentially means that the server recognizes you by your windows login. SQL Server Authentication means that the IT department created a distinct username and password for you. If you're using SQL Server Authentication, you need to provide a username and password.
- **Save My Password:** You can select the check box next to Save My Password if you want your username and password to be stored in the workbook. Your connections can then remain refreshable when being used by other people. This option obviously has security considerations, because anyone can view the connection properties and see your username and password. You should use this option only if your IT department has set you up with an application account (an account created specifically to be used by multiple people).
- **Database Name:** Every SQL Server can contain multiple databases. Enter the name of the database you're connecting to. You can get it from your IT department whenever someone gives you access.

After you enter all the pertinent information, click the Next button to see the next screen, shown in Figure 4-3. You have the choice of selecting from a list of tables and views or writing your own custom query using SQL syntax. In most cases, you choose the option to select from a list of tables.

The Table Import Wizard reads the database and shows you a list of all available tables and views (see Figure 4-4). Tables have an icon that looks like a grid, and views have an icon that looks like a box on top of another box.

The idea is to place a check mark next to the tables and views you want to import. In Figure 4-4, note the check mark next to the MasterDates table. The Friendly Name column allows you to enter a new name that will be used to reference the table in Power Pivot.

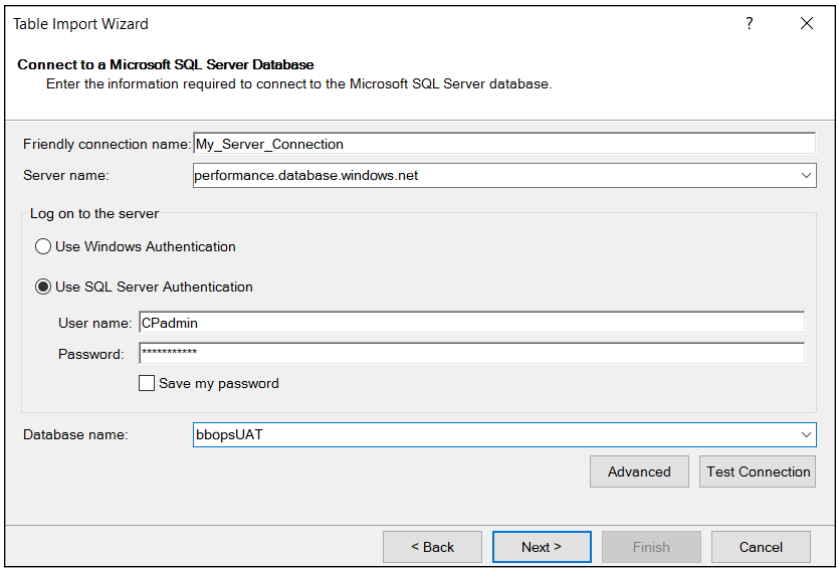


FIGURE 4-2: Provide the basic information needed to connect to the target database.

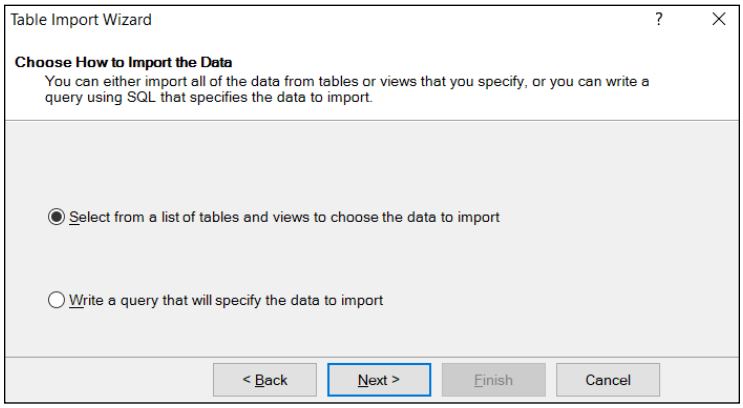


FIGURE 4-3: Choose to select from a list of tables and views.

TIP

In Figure 4-4, you see the Select Related Tables button. After you select one or more tables, you can click this button to tell Power Pivot to scan for, and automatically select, any other tables that have a relationship with the table(s) you've already selected. This feature is handy to have when sourcing large databases with dozens of tables.

REMEMBER

Importing a table imports all columns and records for that table. This can have an impact on the size and performance of your Power Pivot data model. You will often find that you need only a handful of the columns from the tables you import. In these cases, you can use the Preview & Filter button.

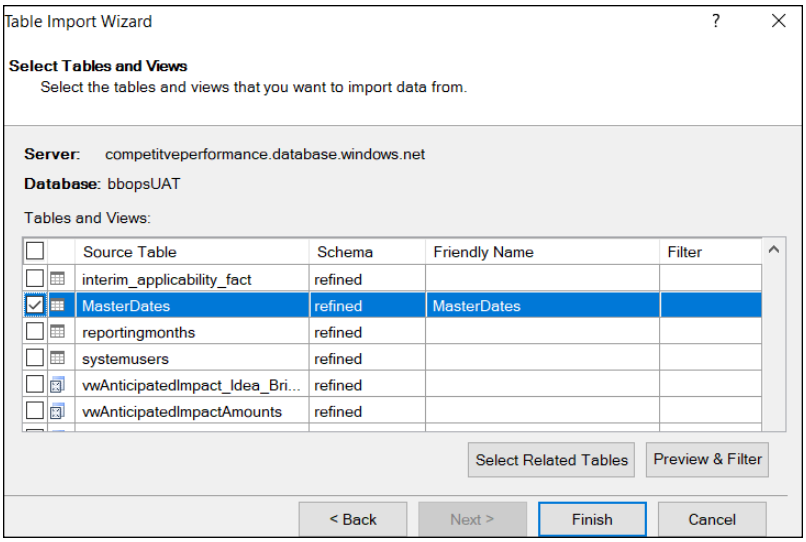


FIGURE 4-4: The Table Import Wizard offers up a list of tables and views.

IMPORTING TABLES VERSUS IMPORTING VIEWS

You may recall from reading Chapter 1 that views are query objects that are built to extract subsets of data from database tables based on certain predefined conditions. (That's a mouthful!) Views are typically created by someone familiar with the database as a kind of canned reporting mechanism that outputs a ready-to-use data set.

There are pros and cons to importing tables versus views.

Tables come with the benefit of defined relationships. When you import tables, Power Pivot can recognize the relationships between the tables and automatically duplicate the relationships in the data model. Tables are also more transparent, allowing you to see all the raw unfiltered data. However, when you import tables, you have to have some level of understanding of the database schema and how the values within the tables are utilized in context of the organization's business rules. In addition, importing a table imports all the columns and records, whether you need them or not. To keep the size of your Power Pivot data model manageable, this often forces you to take the extra step of explicitly filtering out the columns you don't need.

(continued)

(continued)

Views are often cleaner data sets because they are already optimized to include only the columns and data that are necessary. In addition, you don't need to have an intimate knowledge of the database schema. Someone with that knowledge has already done the work for you — joined the correct tables, applied the appropriate business rules, and optimized output, for example. What you lose with views, however, is the ability for Power Pivot to automatically recognize and build relationships within the data model. Also, if you don't have the rights to open the views in Design mode, you lose transparency because you cannot see exactly what the view is doing to come up with its final output.

In terms of which is better to use — tables or views — it's generally considered a best practice to use views whenever possible. They not only provide you with cleaner, more user-friendly data but can also help streamline your Power Pivot data model by limiting the amount of data you import. Regardless, using tables is by no means frowned upon and is often the only option because of the lack of database rights or availability of predefined views. You may even find yourself importing both tables and views from the same database.

Click the table name to highlight it in blue (refer to Figure 4-4), and then click the Preview & Filter button. The Table Import Wizard opens the Preview Selected Table screen, shown in Figure 4-5. You can see all columns available in the table, with a sampling of rows.

Each column header has a check box next to it, indicating that the column will be imported with the table. Removing the check mark tells Power Pivot to not include that column in the data model. For instance, in Figure 4-5, only the first three columns are checked; the unchecked columns won't be imported.

You also have the option to filter out certain records. Figure 4-6 demonstrates that clicking on the drop-down arrow for any of the columns opens a Filter menu that allows you to specify criterion to filter out unwanted records. This works just like the standard filtering in Excel. You can select and deselect the data items in the filtered list, or, if there are too many choices, you can apply a broader criteria by clicking Date Filters above the list. (If you're filtering a textual column, it's Text Filters.)

After you finish selecting your data and applying any needed filters, you can click the Finish button on the Table Import Wizard to start the import process. The import log, shown in Figure 4-7, shows the progress of the import and summarizes the import actions taken after completion.

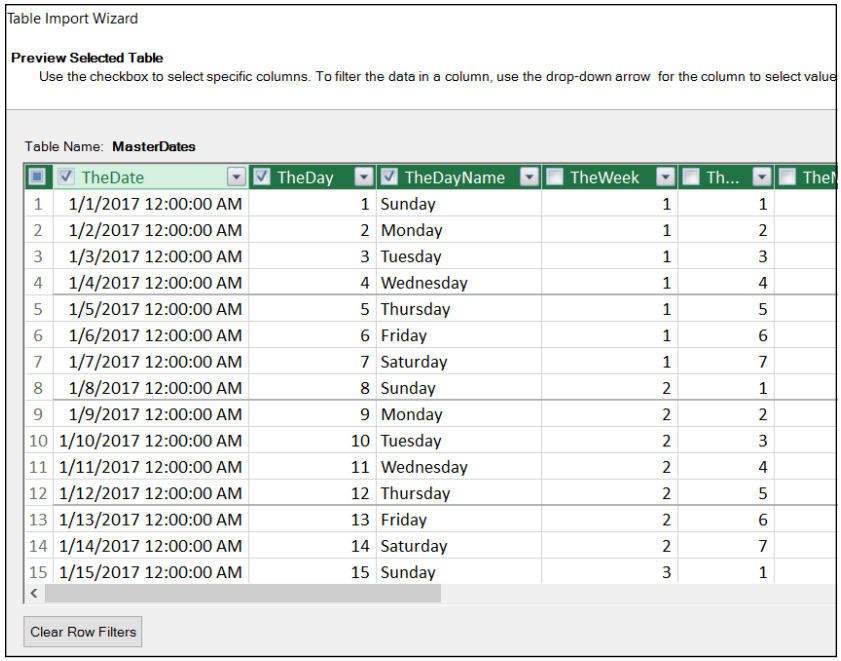


FIGURE 4-5: The Preview & Filter screen allows you to uncheck columns you don't need.

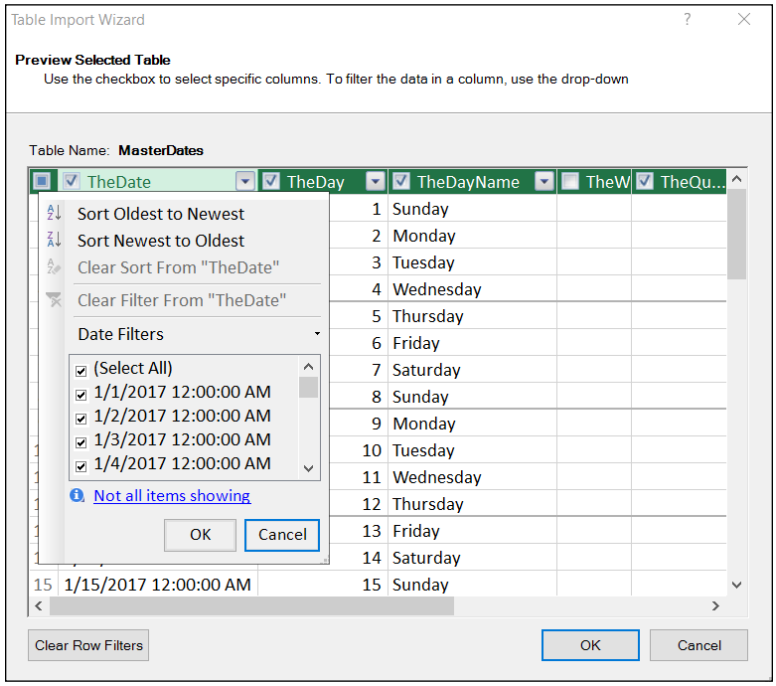


FIGURE 4-6: Use the drop-down arrows next to each column to filter out unneeded records.

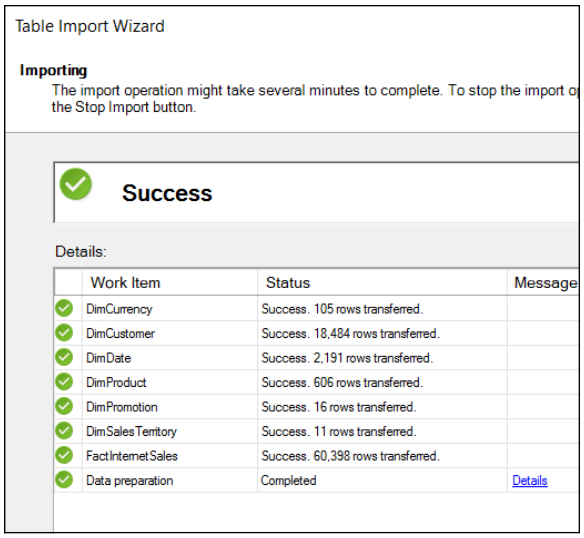


FIGURE 4-7: The last screen of the Table Import Wizard shows you the progress of your import actions.

The final step in loading data from SQL Server is to review and create any needed relationships. Back in the Power Pivot window, click the Diagram View command button on the Home tab. Power Pivot opens the diagram screen, where you can view and edit relationships as needed.

Refer to Chapter 2 for a refresher on managing relationships for tables imported into the internal data model.

TIP

Don't panic if you feel like you've botched the column-and-record filtering on your imported Power Pivot table. Simply select the worrisome table in the Power Pivot window and open the Edit Table Properties dialog box (choose Design ➪ Table Properties). Note that this dialog box is basically the same Preview & Filter screen you encounter in the Import Table Wizard (refer to Figure 4-5). From here, you can select columns you originally filtered out, edit record filters, clear filters, or even use a different table/view.

Loading data from Microsoft Access databases

Because Microsoft Access has traditionally been made available with the Microsoft Office suite of applications, Access databases have long been used by organizations to store and manage mission-critical departmental data. Walk into any organization, and you will likely find several Access databases that contain useful data.

Unlike SQL Server databases, Microsoft Access databases are typically found on local desktops and directories. This means you can typically import data from Access without the help of your IT department.

Open the Power Pivot window and click the From Other Sources command button on the Home tab. This opens the Table Import Wizard dialog box, shown in Figure 4-8. Select the Microsoft Access option, and then click the Next button.

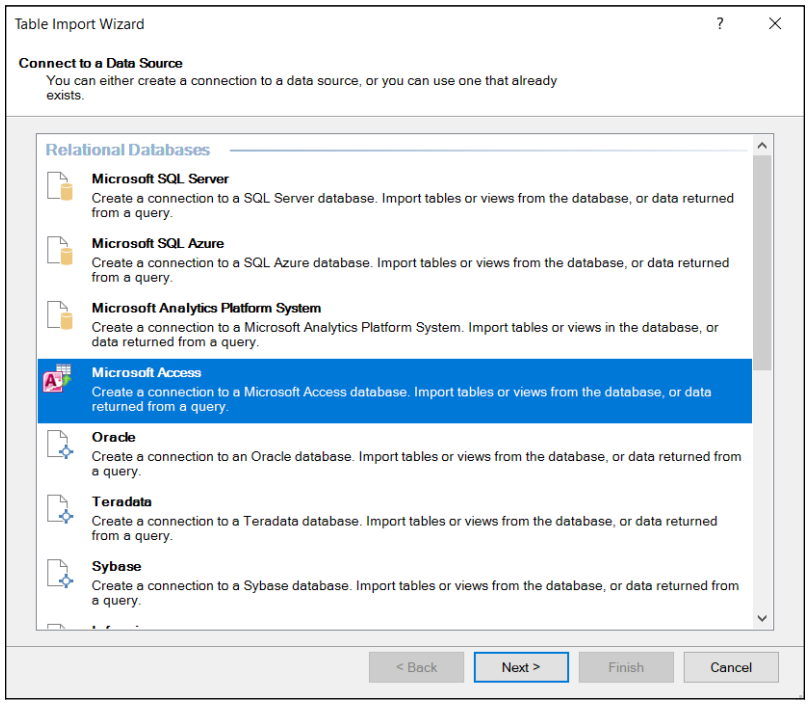


FIGURE 4-8: Open the Table Import Wizard and select Microsoft Access.

The Table Import Wizard asks for all the information it needs to connect to your database (see Figure 4-9).

On this screen, you need to provide the information for these options:

- **Friendly Connection Name:** The Friendly Name field allows you to specify your own name for the external source. You typically enter a name that is descriptive and easy to read.
- **Database Name:** Enter the full path of your target Access database. You can use the Browse button to search for and select the database you want to pull from.

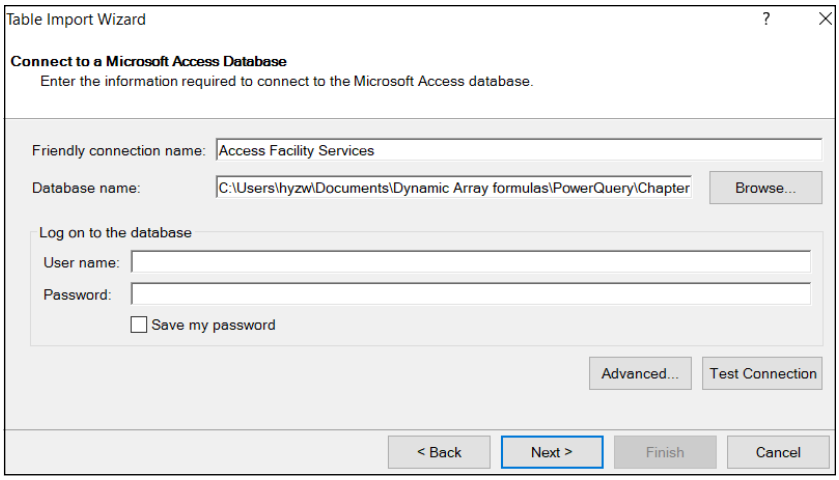


FIGURE 4-9: Provide the basic information needed to connect to the target database.

- **Log On to the Database:** Most Access databases aren't password protected. But if you're connecting one that does require a username and password, enter your login credentials.
- **Save My Password:** You can select the check box next to Save My Password if you want your username and password to be stored in the workbook. Then your connections can remain "refreshable" when being used by other people. Keep in mind that anyone can view the connection properties and see your username and password.

WARNING

Because Access databases are essentially desktop files (`.mdb` or `.accdb`), they're susceptible to being moved, renamed, or deleted. Be aware that the connections in your workbook are hard coded, so if you do move, rename, or delete your Access database, you can no longer connect it.

At this point, you can click the Next button to continue with the Table Import Wizard. From here on out, the process is virtually identical to importing SQL Server data, covered in the last section (starting at Figure 4-3).

Loading data from other relational database systems

Whether your data lives in Oracle, dBase, or MySQL, you can load data from virtually any relational database system. As long as you have the appropriate database drivers installed, you have a way to connect Power Pivot to your data.

Open the Power Pivot window and click the From Other Sources command button on the Home tab. This opens the Table Import Wizard dialog box, shown in Figure 4-10. The idea is to select the appropriate relational database system. If you need to import data from Oracle, select Oracle. If you need to import data from Sybase, select Sybase.

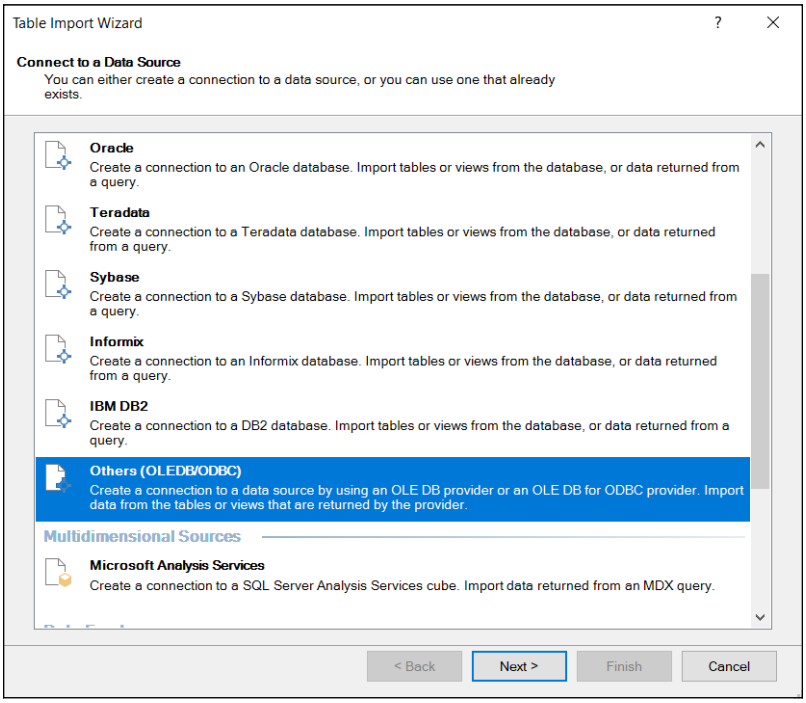


FIGURE 4-10: Open the Table Import Wizard and select your target relational database system.

Connecting to any of these relational systems takes you through roughly the same steps as importing SQL Server data, earlier in this chapter. You may see some alternative dialog boxes based on the needs of the database system you select.

Understandably, Microsoft cannot possibly create a named connection option for every database system out there. So you may not find your database system listed. In this case, simply select the Others option (OLEDB/ODBC). Selecting this option opens the Table Import Wizard, starting with a screen asking you to enter or paste the connection string for your database system (see Figure 4-11).

You may be able to get this connection string from your IT department. If you're having trouble finding the correct syntax for your connection string, you can use the Build button to create the string via a set of dialog boxes. Pressing the Build button opens the Data Link Properties dialog box, shown in Figure 4-12.

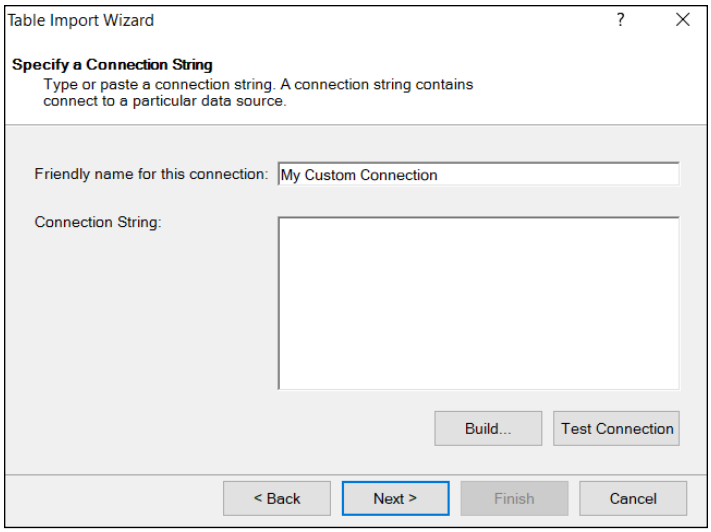


FIGURE 4-11: Enter the connection string for your database system.

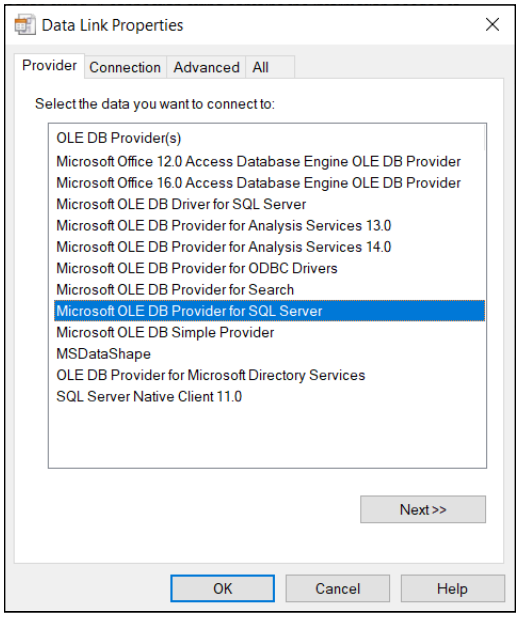


FIGURE 4-12: Use the Data Link Properties dialog box to configure a custom connection string to your relational database system.

Start with the Provider tab, selecting the appropriate driver for your database system. The list you see on your computer will be different from the list shown in Figure 4-13. Your list will reflect the drivers you have installed on your own machine.

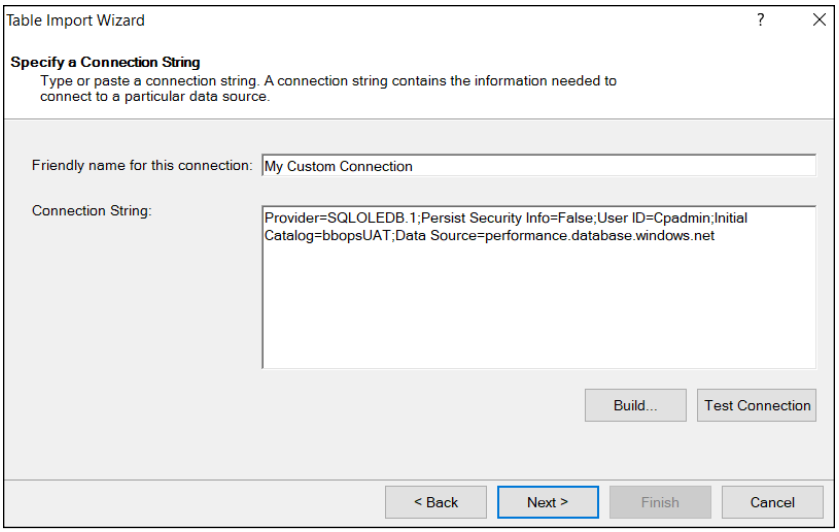


FIGURE 4-13: The Table Import Wizard displays the final syntax for your connection string.

After selecting a driver, move through each tab on the Data Link Properties dialog box and enter the necessary information. When it's complete, click OK to return to the Table Import Wizard, where you see the connection string input box populated with the connection string needed to connect to your database system (see Figure 4-13).

Again, from here on out, the process is virtually identical to importing SQL Server data, as covered earlier in this chapter (starting at Figure 4-3).

REMEMBER

To connect to any database system, you must have that system's drivers installed on your PC. Because SQL Server and Access are Microsoft products, their drivers are virtually guaranteed to be installed on any PC with Windows installed. The drivers for other database systems, however, need to be explicitly installed — typically, by the IT department either at the time the machine is loaded with corporate software or upon demand. If you don't see the needed drivers for your database system, contact your IT department.

Loading Data from Flat Files

The term *flat file* refers to a file that contains some form of tabular data without any sort of structural hierarchy or relationship between records. The most common types of flat files are Excel files and text files. Whether anyone likes to admit it or not, a ton of important data is maintained in flat files. In this section, I tell you how to import these flat file data sources into the Power Pivot data model.

Loading data from external Excel files

In Chapter 2, I show you how to create linked tables by loading Power Pivot with the data contained within the same workbook. Linked tables have a distinct advantage over other types of imported data in that they immediately respond to changes in the source data within the workbook. If you change the data in one of the tables in the workbook, the linked table within the Power Pivot data model automatically changes. The real-time interactivity you get with linked tables is especially helpful if you're making frequent changes to your data.

The drawback to linked tables is that the source data must be stored in the same workbook as the Power Pivot data model. This isn't always possible. You'll encounter plenty of scenarios where you need to incorporate Excel data into your analysis, but that data lives in another workbook. In these cases, you can use Power Pivot's Table Import Wizard to connect to external Excel files.

Open the Power Pivot window and click the From Other Sources command button on the Home tab. This opens the Table Import Wizard dialog box, shown in Figure 4-14. Select the Excel File option and then click the Next button.

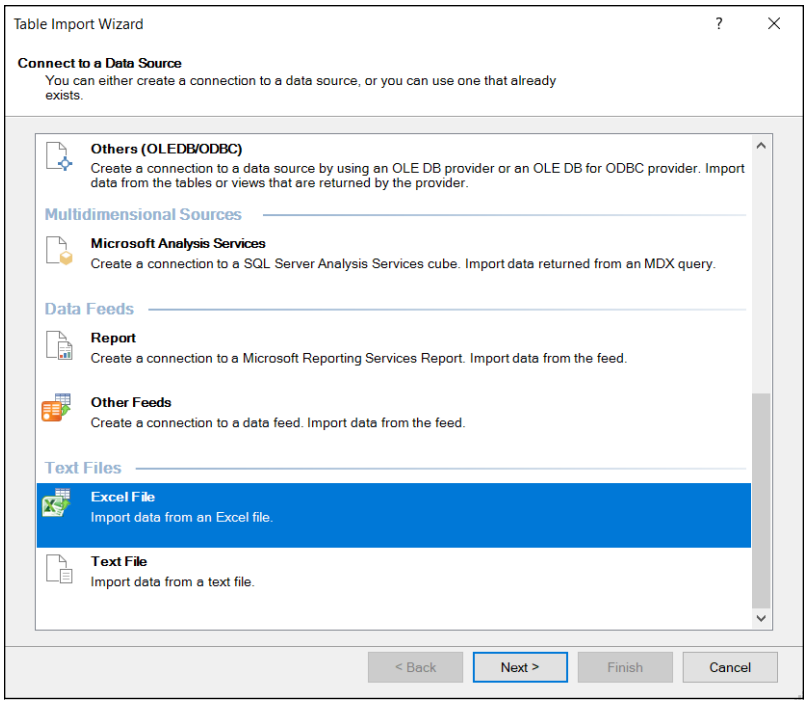


FIGURE 4-14: Open the Table Import Wizard and select Excel File.

The Table Import Wizard asks for all the information it needs to connect to your target workbook (see Figure 4-15).

On this screen, you need to provide the following information:

- **Friendly Connection Name:** In the Friendly Connection Name field, you specify your own name for the external source. You typically enter a name that is descriptive and easy to read.
- **Excel File Path:** Enter the full path of your target Excel workbook. You can use the Browse button to search for and select the workbook you want to pull from.
- **Use First Row as Column Headers:** In most cases, your Excel data will have column headers. Select the check box next to Use First Row As Column Headers to ensure that your column headers are recognized as headers when imported.

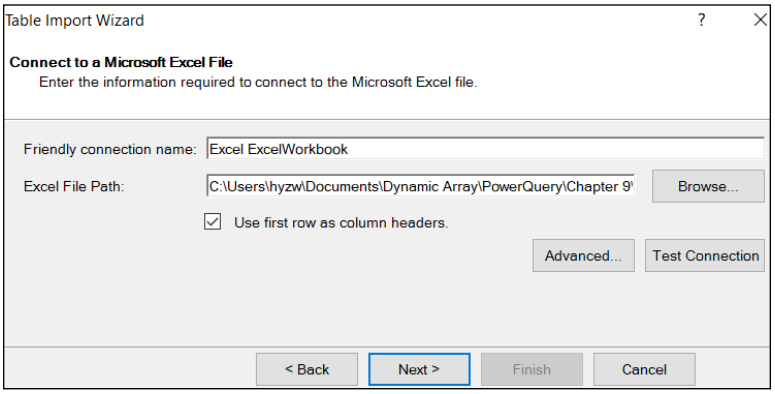


FIGURE 4-15: Provide the basic information needed to connect to the target workbook.

After you enter all the pertinent information, click the Next button to see the next screen, shown in Figure 4-16. You see a list of all worksheets and named ranges in the chosen Excel workbook. Place a check mark next to the data set you want to import. The Friendly Name column allows you to enter a new name that will be used to reference the data in Power Pivot.

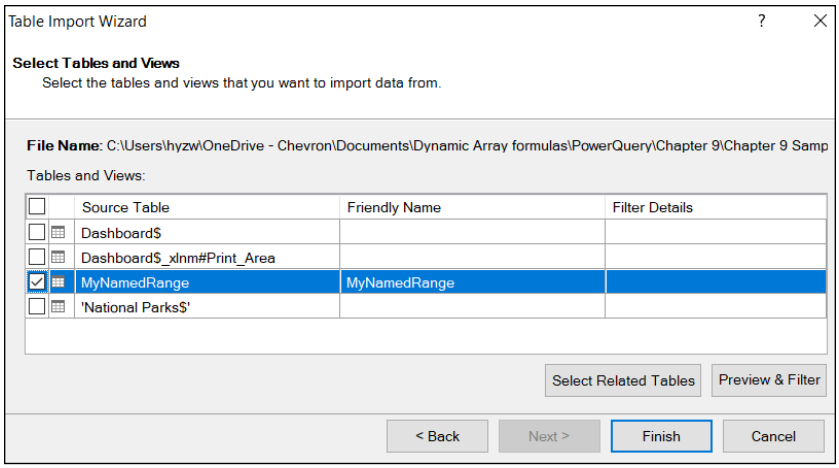


FIGURE 4-16: Select the data sources to import.

REMEMBER

As discussed earlier in this chapter, in the section "Loading Data from Relational Databases," you can use the Preview & Filter button to filter out unwanted columns and records, if needed. Otherwise, continue with the Table Import Wizard to complete the import process. As always, be sure to review and create relationships to any other tables you've loaded into the Power Pivot data model.

REMEMBER

Loading external Excel data doesn't give you the same interactivity you get with linked tables. As with importing database tables, the data you bring from an external Excel file is simply a snapshot. You need to refresh the data connection to see any new data that may have been added to the external Excel file (see "Refreshing and Managing External Data Connections," later in this chapter).

Loading data from text files

The text file is another type of flat file used to distribute data. This type of file is commonly output from legacy systems and websites. Excel has always been able to consume text files. With Power Pivot, you can go further and integrate them with other data sources.

Open the Power Pivot window and click the From Other Sources command button on the Home tab. This opens the Table Import Wizard dialog box shown in Figure 4-17. Select the Text File option and then click the Next button.

The Table Import Wizard asks for all the information it needs to connect to the target text file (see Figure 4-18).

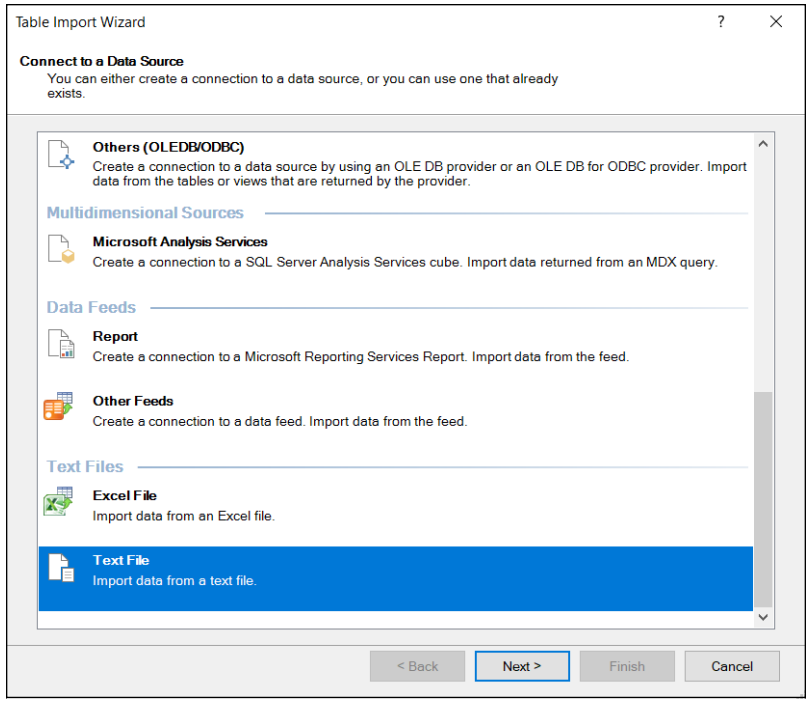


FIGURE 4-17: Open the Table Import Wizard and select Text File.

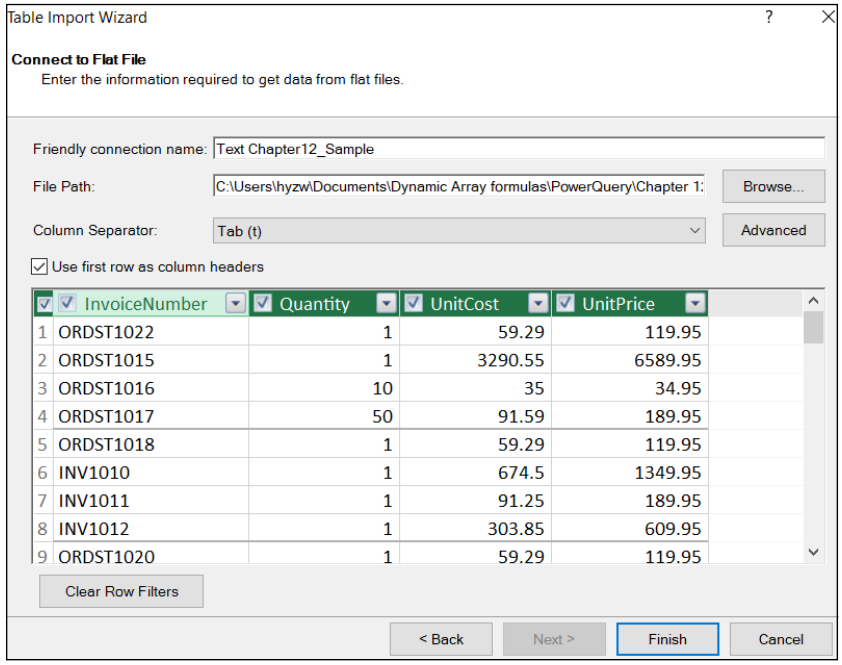


FIGURE 4-18: Provide the basic information needed to connect to the target text file.

On this screen, you provide the following information:

- **Friendly Connection Name:** The Friendly Connection Name field allows you to specify your own name for the external source. You typically enter a name that is descriptive and easy to read.
- **File Path:** Enter the full path of your target text file. You can use the Browse button to search for and select the file you want to pull from.
- **Column Separator:** Select the character used to separate the columns in the text file. Before you can do this, you need to know how the columns in your text file are delimited. For instance, a comma-delimited file will have commas separating its columns. A tab-delimited file will have tabs separating the columns. The drop-down list in the Table Import Wizard includes choices for the more common delimiters: Tab, Comma, Semicolon, Space, Colon, and Vertical bar.
- **Use First Row as Column Headers:** If your text file contains header rows, be sure to select the check box next to Use First Row as a Column Headers. This ensures that the column headers are recognized as headers when imported.

Notice that you see an immediate preview of the data in the text file. Here, you can filter out any unwanted columns by simply removing the check mark next to the column names. You can also use the drop-down arrows next to each column to apply any record filters.

Clicking the Finish button immediately starts the import process. Upon completion, the data from your text file will be part of the Power Pivot data model. As always, be sure to review and create relationships to any other tables you've loaded into Power Pivot.

Anyone who's worked with text files in Excel knows that they're notorious for importing numbers that look like numbers, but are really coded as text. In standard Excel, you use Text to Columns to fix these kinds of issues. Well, this can be a problem in Power Pivot, too.

TIP

When importing text files, take the extra step of verifying that all columns have been imported with the correct data formatting. You can use the formatting tools found on the Power Pivot window's Home tab to format any column in the data model.

Loading data from the Clipboard

Power Pivot includes an interesting option for loading data straight from the Clipboard — that is to say, pasting data you've copied from some other place. This option is meant to be used as a one-off technique to quickly get useful information into the Power Pivot data model.

As you consider this option, keep in mind that there is no real data source. It's just you manually copying and pasting. You have no way to refresh the data, and you have no way to trace back to where you copied the data from.

Imagine that you've received the Word document shown in Figure 4-19. You like the nifty table of holidays within the document, and you believe it would be useful in your Power Pivot data model.

Holiday	Year	Month	Day
Martin Luther King Jr. Day	2022	Jan	17
Confederate Memorial Day	2022	Jan	19
Washington's Birthday	2022	Feb	21
Texas Independence Day	2022	Mar	2
Cesar Chavez Day	2022	Mar	31
Good Friday	2022	Apr	15
San Jacinto Day	2022	Apr	21
Memorial Day	2022	May	30
Juneteenth	2022	Jun	19
Independence Day	2022	Jul	4
Labor Day	2022	Sep	5
Veterans Day	2022	Nov	11
Thanksgiving	2022	Nov	24
Christmas Eve	2022	Dec	24
Christmas Day	2022	Dec	25

FIGURE 4-19: You can copy data straight out of Microsoft Word.

You can copy the table and then go to the Power Pivot window and click the Paste command on the Home tab. This opens the Paste Preview dialog box, shown in Figure 4-20, where you can review what exactly will be pasted. You won't see many options here. You can specify the name that will be used to reference the table in Power Pivot, and you can specify whether the first row is a header.

Clicking the OK button imports the pasted data into Power Pivot without a lot of fanfare. At this point, you can adjust the data formatting and create the needed relationships.

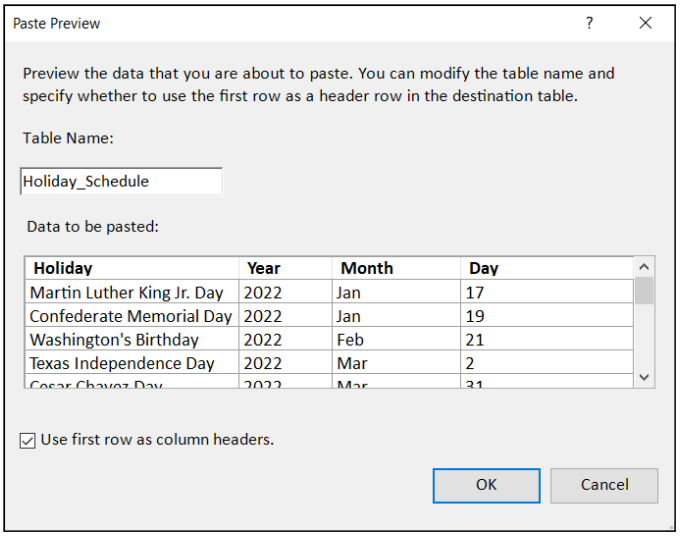


FIGURE 4-20: The Paste Preview dialog box gives you a chance to see what you're pasting.

Loading Data from Other Data Sources

At this point, I've covered the data sources that are most important to a majority of Excel analysts. Still, there are a few more data sources that Power Pivot is able to connect to and load data from. I touch on some of these data sources later in this book, though others remain out of scope.

Although these data sources are not likely to be used by your average analyst, it's worth dedicating a few lines to each one, if only to know that they exist and are available if ever you should need them:

- **Microsoft SQL Azure:** SQL Azure is a cloud-based relational database service that some companies use as an inexpensive way to gain the benefits of SQL Server without taking on the full cost of hardware, software, and IT staff. Power Pivot can load data from SQL Azure in much the same way as the other relational databases I talk about in this chapter.
- **Microsoft Analytics Platform System:** Azure Synapse Analytics is an analytics service that allows for data integration, enterprise data warehousing, and big data analytics. From a Power Pivot perspective, it's no different than connecting to any other relational database.
- **Microsoft Analysis Services:** Analysis Services is Microsoft's OLAP (Online Analytical Processing) product. The data in Analysis Services is traditionally stored in a multidimensional cube.

- **Report:** The curiously named Report data source refers to SQL Server Reporting Services reports. In a very basic sense, Reporting Services is a business intelligence tool used to create stylized PDF-style reports from SQL Server data. In the context of Power Pivot, a Reporting Services Report can be used as a data-feed service, providing a refreshable connection to the underlying SQL Server data.
- **Other Feeds:** The Other Feeds data source allows you to import data from OData web services into Power Pivot. OData connections are facilitated by XML Atom files. Point the OData connection to the URL of the `.atomsvcs` file and you essentially have a connection to the published web service.

Refreshing and Managing External Data Connections

When you load data from an external data source into Power Pivot, you essentially create a static snapshot of that data source at the time of creation. Power Pivot uses that static snapshot in its Internal Data Model.

As time goes by, the external data source may change and grow with newly added records. However, Power Pivot is still using its snapshot, so it can't incorporate any of the changes in your data source until you take another snapshot.

The action of updating the Power Pivot data model by taking another snapshot of your data source is called *refreshing* the data. You can refresh manually, or you can set up an automatic refresh.

Manually refreshing Power Pivot data

On the home tab of the Power Pivot window, you see the Refresh command. Click the drop-down arrow below it to see two options shown in Figure 4-21: Refresh and Refresh All.

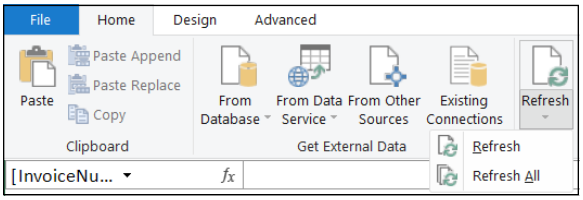


FIGURE 4-21: Power Pivot allows you to refresh one table or all tables.

Use the Refresh option to refresh the Power Pivot table that's active. That is to say, if you're on the Products_Table tab in Power Pivot, clicking Refresh reaches out to the external source and requests an update for only that table. This works nicely when you need to strategically refresh only certain data sources.

Use the Refresh All option to refresh all tables in the Power Pivot data model.

Setting up automatic refreshing

You can configure your data sources to automatically pull the latest data and refresh Power Pivot.

Go to the Data tab on the Ribbon and select the Queries & Connections command. The Queries & Connections task pane, shown in Figure 4-22, opens. Right-click the data connection you want to work with and then click the Properties button.

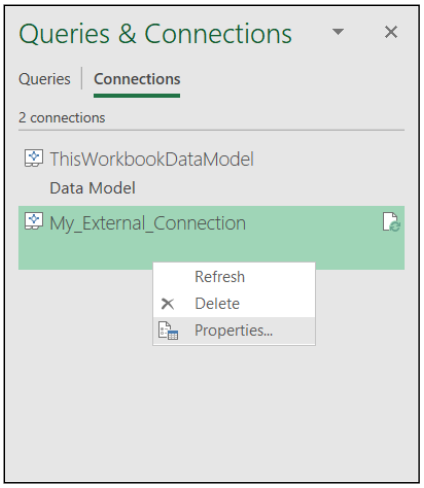


FIGURE 4-22: Select a connection and click the Properties button.

With the Connection Properties dialog box open, select the Usage tab. Here, you'll find an option to refresh the chosen data connection every X minutes and an option to refresh the data connection when the Excel workbook is opened (see Figure 4-23):

- **Refresh Every X Minutes:** Placing a check next to this option tells Excel to automatically refresh the chosen data connection a specified number of minutes. This refreshes all tables associated with that connection.

» **Refresh Data When Opening the File:** Placing a check mark next to this option tells Excel to automatically refresh the chosen data connection after opening of the workbook. This refreshes all tables associated with that connection as soon as the workbook is opened.

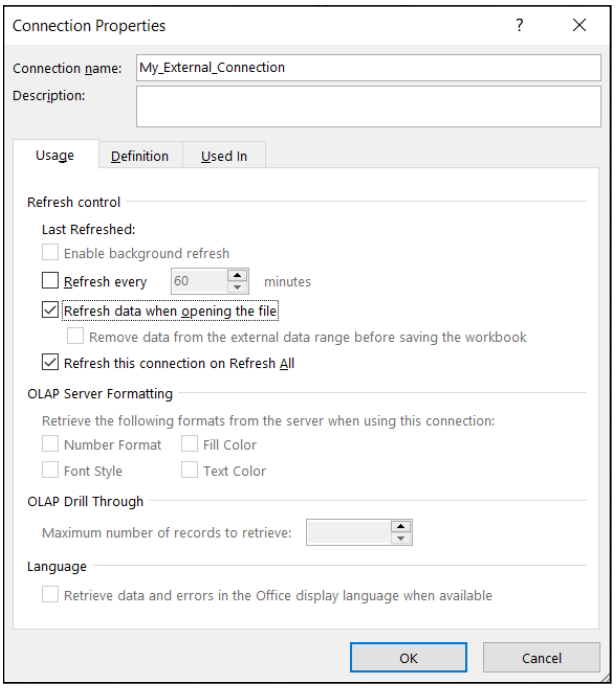


FIGURE 4-23: The Connection Properties dialog box lets you configure the chosen data connection to refresh automatically.

Preventing Refresh All

Earlier in this section, you see that you can refresh all connections that feed Power Pivot, by using the Refresh All command (refer to Figure 4-21). Well, there are actually two more places where you can click Refresh All in Excel: on the Data tab in the Excel Ribbon and on the PivotTable Analyze tab you see when working in a pivot table.

Clicking any Refresh All button anywhere in Excel essentially completely reloads Power Pivot, refreshes all pivot tables, and updates all workbook data connections. If your Power Pivot data model imports millions of lines of data from an external data source, you may well want to avoid using the Refresh All feature.

Luckily, you have a way to prevent certain data connections from refreshing when Refresh All is selected. Go to the Data tab on the Excel Ribbon and select the Queries & Connections command. This opens the Queries & Connections task pane, where you right-click the data connection you want to configure, and then click the Properties button.

When the Connection Properties dialog box has opened, select the Usage tab and then remove the check mark next to the Refresh This Connection on Refresh All (as shown in Figure 4-24).

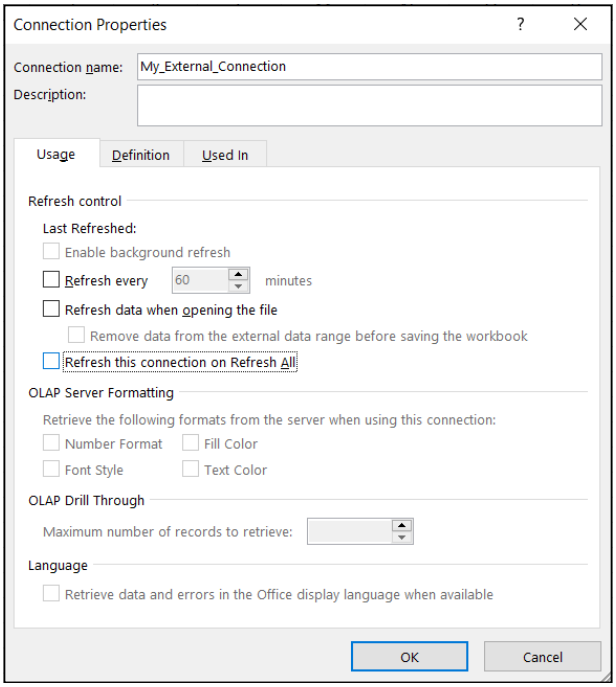


FIGURE 4-24: The Connection Properties dialog box lets you configure the chosen data connection to ignore the Refresh All command.

Editing the data connection

In certain instances, you may need to edit the source data connection after you've already created it. Unlike refreshing, where you simply take another snapshot of the same data source, editing the source data connection allows you to go back and reconfigure the connection itself. Here are a few reasons you may need to edit the data connection:

- The location or server or data source file has changed.
- The name of the server or data source file has changed.

- You need to edit your login credentials or authentication mode.
- You need to add tables you left out during initial import.

In the Power Pivot window, go to the Home tab and click the Existing Connections command button. The Existing Connections dialog box, shown in Figure 4-25, opens. Your Power Pivot connections are under the Power Pivot Data Connections subheading. Choose the data connection that needs editing.

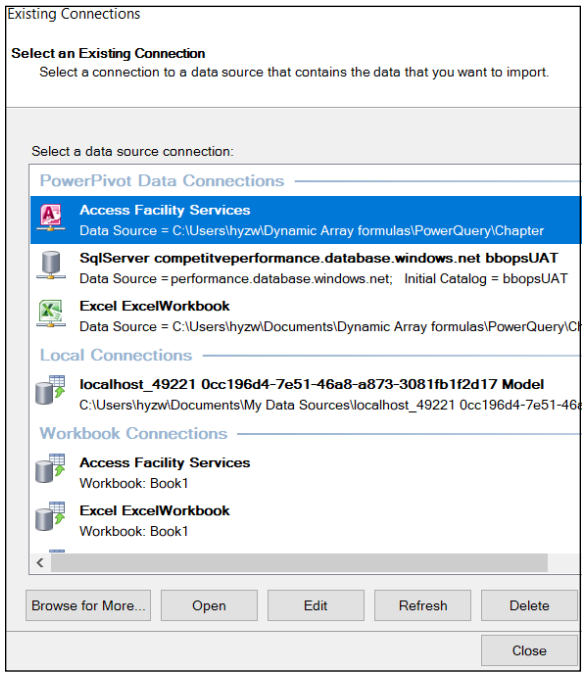


FIGURE 4-25: Use the Existing Connections dialog box to reconfigure your Power Pivot source data connections.

After your target data connection is selected, look to the Edit and Open buttons. The button you click depends on what you need to change:

- **Edit button:** Lets you reconfigure the server address, file path, and authentication settings.
- **Open button:** Lets you import a new table from the existing connection, which is handy when you've inadvertently missed a table during the initial loading of data.
- **Refresh button:** Lets you refresh the selected data source.
- **Browse for More:** Lets you quickly establish a new data connection by pointing to an existing Office Database Connection (`.odc`) file.

IN THIS CHAPTER

- Interacting with the internal data model directly
- Starting a pivot table from the internal data model
- Using multiple tables with the internal data model

Chapter 5

Working Directly with the Internal Data Model

In the preceding chapters, you use the Power Pivot add-in to work with the internal data model. But as you'll see in this chapter, you can use a combination of pivot tables and Excel data connections to directly interact with the internal data model, without the Power Pivot add-in.

You can find the sample files for this chapter on this book's companion website at `www.dummies.com/go/excelpowerpivotpowerqueryfd2e`. These include the `Chapter 5 Sample File.xlsx` Excel workbook and the `Facility Services.accdb` Access database.

Directly Feeding the Internal Data Model

Imagine that you have the Transactions table you see in Figure 5-1, and on another worksheet you have an Employees table (see Figure 5-2) that contains information about the employees.

	A	B	C	D
1	Sales_Rep	Invoice_Date	Sales_Amount	Contracted Hours
2	4416	1/5/2021	111.79	2
3	4416	1/5/2021	111.79	2
4	160006	1/5/2021	112.13	2
5	6444	1/5/2021	112.13	2
6	160006	1/5/2021	145.02	3
7	52661	1/5/2021	196.58	4
8	6444	1/5/2021	204.20	4
9	51552	1/5/2021	225.24	3
10	55662	1/6/2021	86.31	2
11	1336	1/6/2021	86.31	2
12	60224	1/6/2021	86.31	2
13	54564	1/6/2021	86.31	2

FIGURE 5-1: This table shows transactions by employee number.

	A	B	C	D
1	Employee_Number	Last_Name	First_Name	Job_Title
2	21	SIOCAT	ROBERT	SERVICE REPRESENTATIVE 3
3	42	BREWN	DONNA	SERVICE REPRESENTATIVE 3
4	45	VAN HUILE	KENNETH	SERVICE REPRESENTATIVE 2
5	104	WIBB	MAURICE	SERVICE REPRESENTATIVE 2
6	106	CESTENGIAY	LUC	SERVICE REPRESENTATIVE 2
7	113	TRIDIL	ROCH	SERVICE REPRESENTATIVE 2
8	142	CETE	GUY	SERVICE REPRESENTATIVE 3
9	145	ERSINEILT	MIKE	SERVICE REPRESENTATIVE 2
10	162	GEBLE	MICHAEL	SERVICE REPRESENTATIVE 2
11	165	CERDANAL	ALAIN	SERVICE REPRESENTATIVE 3
12	201	GEIDRIOU	DOMINIC	TEAMLEAD 1
13	213	MERAN	DENIS	SERVICE REPRESENTATIVE 3

FIGURE 5-2: This table provides information on employees: first name, last name, and job title.

You need to create an analysis that shows sales by job title. This would normally be difficult given the fact that sales and job title are in two separate tables. But with the internal data model, you can follow these simple steps:

1. **Click inside the Transactions data table and start a new pivot table by choosing Insert➪PivotTable from the Ribbon.**
2. **In the Create PivotTable dialog box, place a check next to the option called Add This Data to the Data Model (see Figure 5-3) and then click OK.**
3. **Click inside the Employees data table and start a new pivot table by choosing Insert➪PivotTable from the Ribbon.**

   Again, be sure to select the Add This Data to the Data Model option, as shown in Figure 5-4.

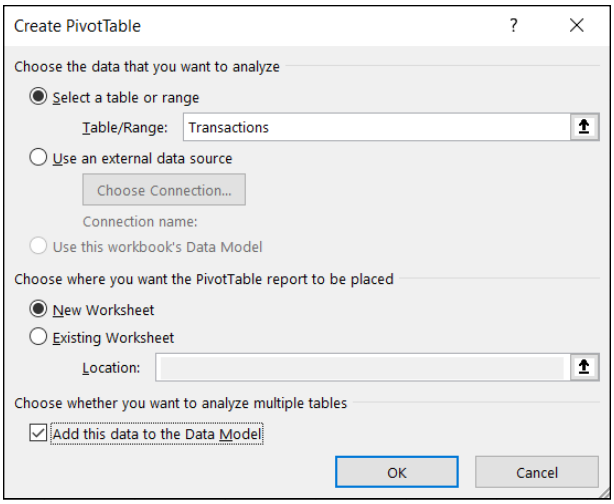


FIGURE 5-3: When you create a new pivot table from the Transactions table, be sure to select Add This Data to the Data Model.

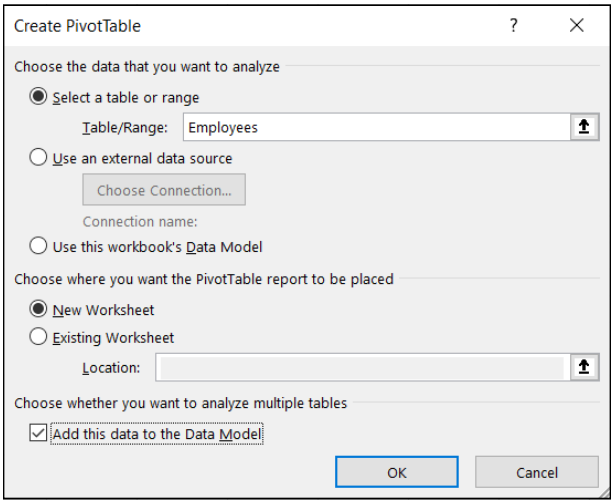


FIGURE 5-4: Create a new pivot table from the Employees table, and select Add This Data to the Data Model.

TIP

Notice that in Figures 5-3 and 5-4, the Create PivotTable dialog boxes are referencing named ranges. That is to say, each table was given a specific name. When you're adding data to the internal data model, it's a best practice to name the data tables. This way, you can easily recognize your tables in the internal data model.

If you don't name your tables, the internal data model shows them as Range1, Range2, and so on.

4. **To give the data table a name, simply highlight all data in the table, and then select Formulas ➪ Define Name command from the Ribbon. In the dialog box, enter a name for the table.**

   Repeat for all other tables.

5. **After both tables have been added to the internal data model, open the PivotTable Fields list and choose the ALL selector, as shown in Figure 5-5.**

   This step shows both ranges in the field list.

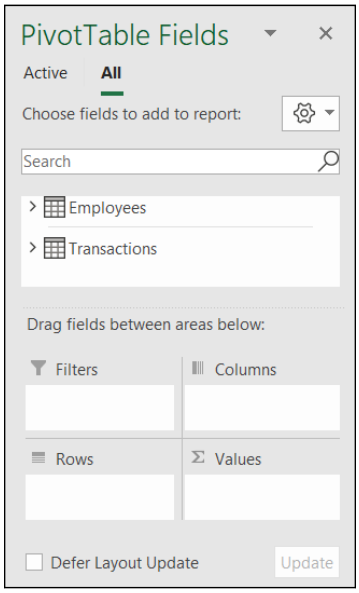


FIGURE 5-5: Select ALL in the PivotTable Fields list to see both tables in the internal data model.

6. **Build out the pivot table as normal. In this case, Job_Title is placed in the Row area, and Sales_Amount goes to the Values area.**

   As you can see in Figure 5-6, Excel immediately recognizes that you're using two tables from the internal data model and prompts you to create a relationship between them. You have the option to let Excel autodetect the relationships between your tables or to click the Create button. Always create the relationships yourself, to avoid any possibility of Excel getting it wrong.

7. **Click the Create button.**

   Excel opens the Create Relationship dialog box, shown in Figure 5-7. There, you select the tables and fields that define the relationship. In Figure 5-7, you can see that the Transactions table has a Sales_Rep field. It's related to the Employees table via the Employee_Number field.

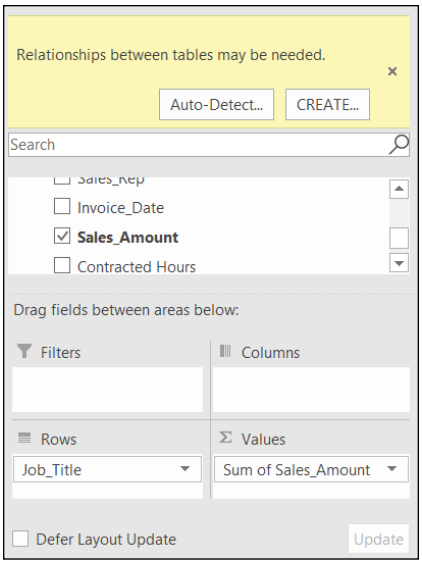


FIGURE 5-6: When Excel prompts you, choose to create the relationship between the two tables.

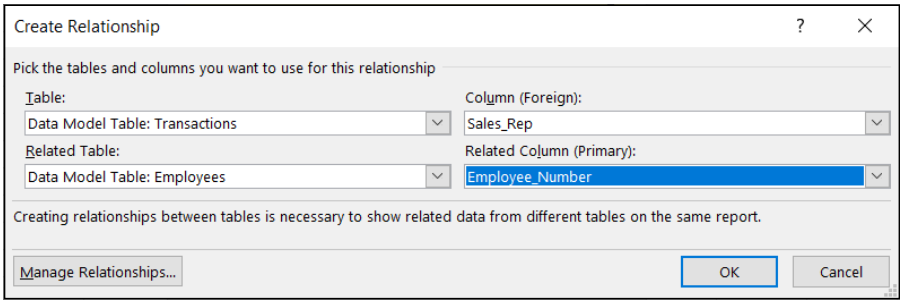


FIGURE 5-7: Build the appropriate relationship using the Table and Column drop-down lists.

After you create the relationship, you have a single pivot table that effectively uses data from both tables to create the analysis you need. Figure 5-8 illustrates that, by using the Excel internal data model, you've achieved the goal of showing sales by job title.

REMEMBER

In Figure 5-7, you see that the lower-right drop-down is named Related Column (Primary). The term *primary* means that the internal data model uses this field from the associated table as the primary key.

A *primary key* is a field that contains only unique non-null values (no duplicates or blanks). Primary key fields are necessary in the data model to prevent aggregation errors and duplications. Every relationship you create must have a field designated as the primary key.

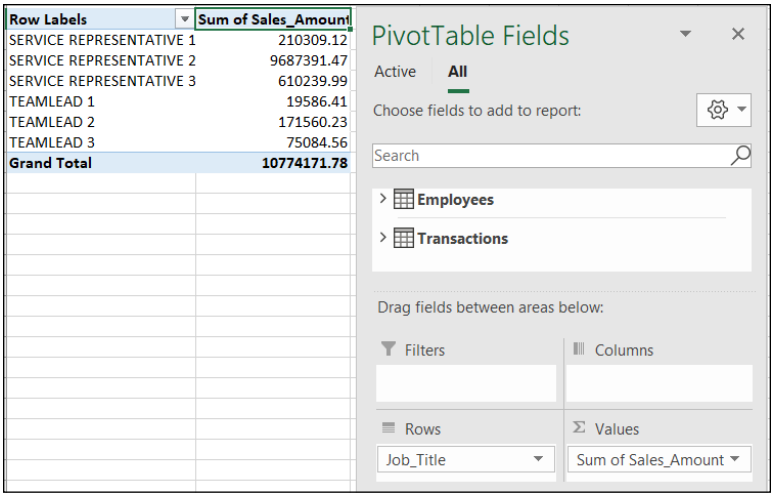


FIGURE 5-8: You've achieved your goal of showing sales by job title.

THE LIMITATIONS OF POWER PIVOT-DRIVEN PIVOT TABLES

Pivot tables built on top of Power Pivot or the internal data model come with limitations that could be showstoppers in terms of your reporting needs. Here's a quick rundown of the limitations you should consider before deciding to base your pivot table reporting on Power Pivot or the internal data model:

- The Group feature is disabled for Power Pivot–driven pivot tables. You can't roll dates into months, quarters, or years, for example.
- In a standard pivot table, you can double-click a cell in the pivot to drill into to the rows that make up the figure in that cell. In Power Pivot–driven pivot tables, however, you see only the first 1,000 rows.
- Power Pivot–driven pivot tables don't allow you to create the traditional Calculated Fields and Calculated Items found in standard Excel pivot tables.
- Workbooks that use the Power Pivot data model can't be refreshed or configured if opened in a version of Excel earlier than Excel 2013.
- You can't use custom lists to automatically sort the data in your Power Pivot–driven pivot tables.
- Neither the Product nor Count Numbers summary calculations are available in Power Pivot–driven pivot tables.

The Employees table (in the scenario in Figure 5-7) must have all unique values in the Employee_Number field, with no blanks or null values. This is the only way that Excel can ensure data integrity when joining multiple tables.

Managing Relationships in the Internal Data Model

After you assign tables to the internal data model, you might need to adjust the relationships between the tables. To make changes to the relationships in an internal data model, click the Data tab on the Ribbon and select the Relationships command. The Manage Relationships dialog box, shown in Figure 5-9, opens.

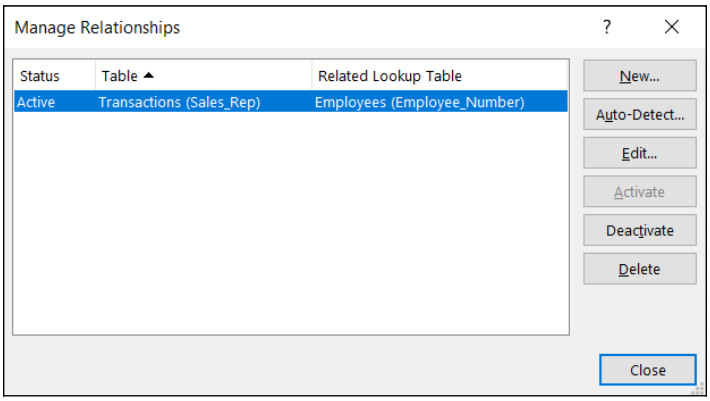


FIGURE 5-9: The Manage Relationships dialog box enables you to make changes to the relationships in the internal data model.

Here, you'll find the following commands:

- **New:** Create a new relationship between two tables in the internal data model.
- **Auto-Detect:** Ask Power Pivot to automatically detect and create relationships.
- **Edit:** Alter the selected relationship.
- **Activate:** Enforce the selected relationship, telling Excel to consider the relationship when aggregating and analyzing the data in the internal data model.

- **Deactivate:** Turn off the selected relationship, telling Excel to ignore the relationship when aggregating and analyzing the data in the internal data model.
- **Delete:** Remove the selected relationship.

Managing Queries and Connections

Select the Data tab on the Ribbon and then select the Queries & Connections command. Excel will activate the Queries & Connections task pane (see Figure 5-10). At the top of the task pane, you'll see two tabs: Queries and Connections. The Queries tab lets you view and manage the queries within the current workbook. The Connections tab lets you manage the connection information stored in your workbook.

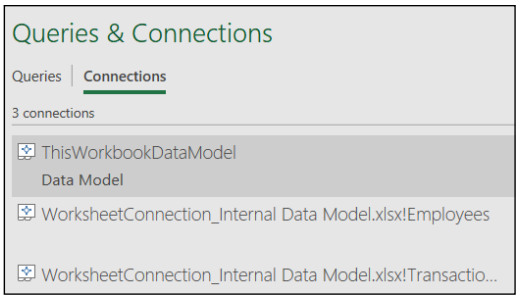


FIGURE 5-10: Use the Queries & Connections task pane to manage the queries and connections in the internal data model.

If you receive a workbook that is unfamiliar to you, it's best practice to activate Queries & Connections just to see if you're dealing with any external connections or queries in the internal data model of the workbook.

Right-click any of the entries on the Connections tab to expose a shortcut menu for that entry, allowing you to refresh the connection, delete the connection, or edit the connection properties.

TIP

The connection name for the internal data model will always be ThisWorkbookDataModel. Excel won't allow you to delete the ThisWorkbookDataModel connection.

Creating a New Pivot Table Using the Internal Data Model

In certain instances, you may want to create a pivot table from scratch using the existing internal data model as the source data. Here are the steps to do so:

1. **Choose Insert➪PivotTable from the Ribbon.**

   The Create PivotTable dialog box opens.

2. **Select the Use an External Data Source option, as shown in Figure 5-11, and then click the Choose Connection button.**

   You see the Existing Connections dialog box, as shown in Figure 5-12.

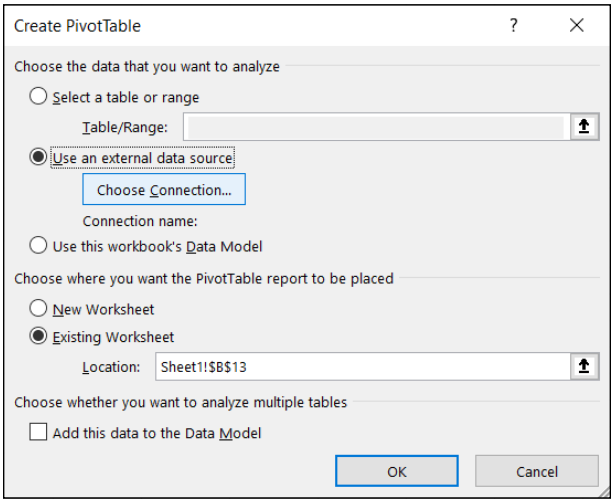


FIGURE 5-11: Open the Create PivotTable dialog box and choose the external data-source option.

3. **On the Tables tab, select Tables in Workbook Data Model, and then click the Open button.**

   You return to the Create PivotTable dialog box.

4. **Click the OK button to create the pivot table.**

   If all goes well, you see the PivotTable Fields dialog box with all tables that are included in the internal data model, as shown in Figure 5-13.

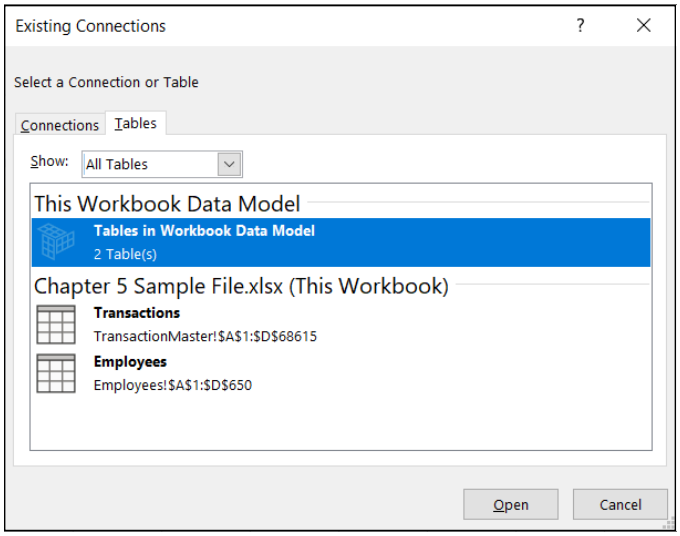


FIGURE 5-12: Use the Existing Connections dialog box to select the Data Model as the data source for your pivot table.

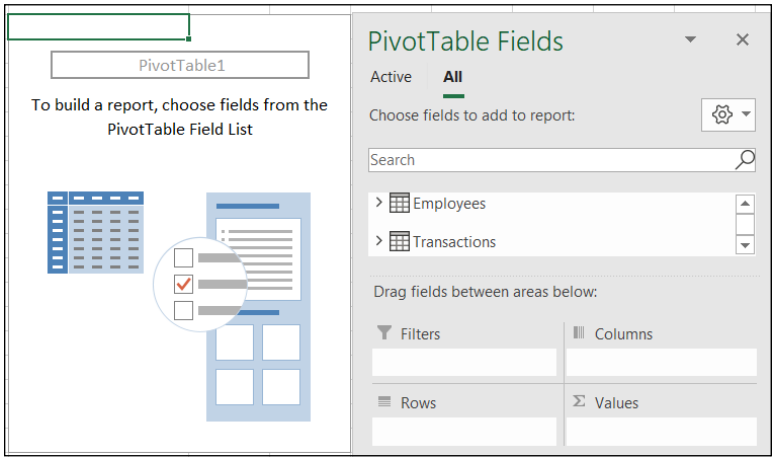


FIGURE 5-13: The newly created pivot table shows all tables in the internal data model.

Filling the Internal Data Model with Multiple External Data Tables

Suppose you have an Access database that contains a normalized set of tables. You want to analyze the data in that database in Excel, so you decide to use the new Excel internal data model to expose the data you need through a pivot table.

To accomplish this task, follow these steps:

1. **Select Data ➪ Get Data ➪ From Database ➪ From Microsoft Access Database (see Figure 5-14).**

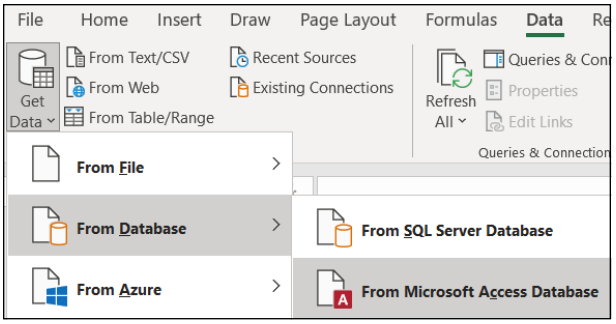


FIGURE 5-14: Getting data from a Microsoft Access database.

2. **Browse to your target Access database and open it.**

   The Navigator dialog box opens.

3. **Place a check mark next to the Enable Selection of Multiple Tables option (see Figure 5-15).**

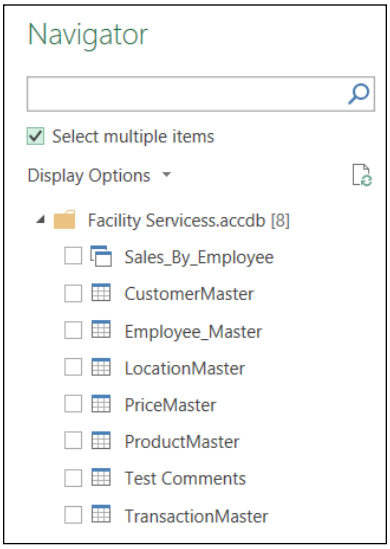


FIGURE 5-15: Enable the selection of multiple tables.

4. **Place a check mark next to each table that you want to import into the internal data model.**

5. **Click the drop-down arrow next to the Load button and select the Load To option as shown in Figure 5-16.**

   The Import Data dialog box opens (see Figure 5-17).

6. **Choose the PivotTable Report option and click OK to create the base pivot.**

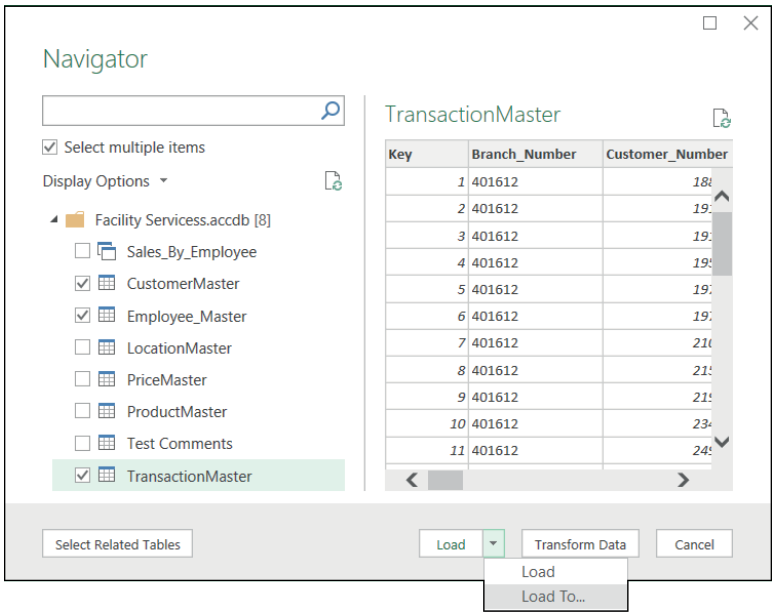


FIGURE 5-16: Place a check next to each table you want import to the internal data model, then activate the Load To option.

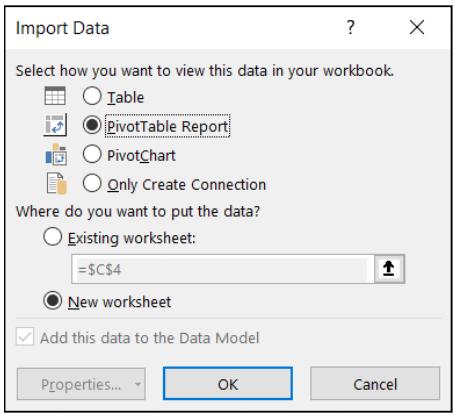


FIGURE 5-17: Create a PivotTable Report from the Import Data dialog.

You now have a pivot table based on external data imported into the internal data model (see Figure 5-18). A quick look at the Pivot Table Field list shows all the external data sources imported into the internal data model.

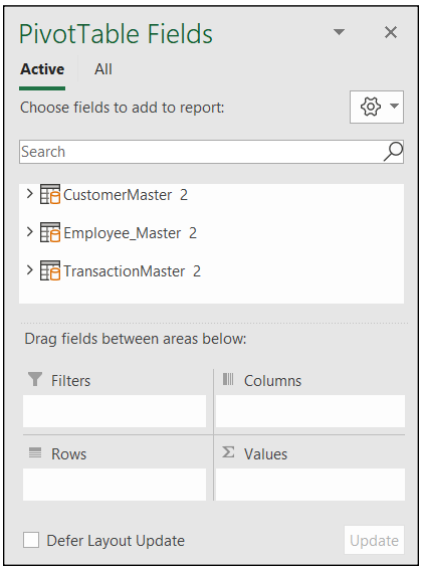


FIGURE 5-18: You're ready to build your pivot table analysis based on multiple external data tables!

In just a few clicks, you've created a powerful platform to build and maintain pivot table analysis based on data in an Access database!

TIP

When you import tables from multiple data sources, Excel tries to detect and create relationships between the tables. It typically does a good job at recognizing the appropriate relationships, especially when your tables contain common column names such as EmployeeID and SalesRep. Though Excel gets the relationships right in most cases, it's always best to confirm the right relationships were created before using your pivot table. Use the Manage Relationships dialog box (shown in Figure 5-9) to double-check the relationships. To activate the Manage Relationships dialog, click inside your pivot table and then choose PivotTable Analyze➪Relationships.

IN THIS CHAPTER

- Creating, formatting, and hiding your own calculated columns
- Creating calculated columns by using DAX
- Creating calculated measures
- Breaking out of pivot tables with cube functions

Chapter 6

Adding Formulas to Power Pivot

When analyzing data with Power Pivot, you often find the need to expand your analysis to include data based on calculations that are not in the original data set. Power Pivot has a robust set of functions (called *DAX* functions) that allow you to perform mathematical operations, recursive calculations, data lookups, and much more.

This chapter introduces you to DAX functions and provides the ground rules for building your own calculations in Power Pivot data models.

Enhancing Power Pivot Data with Calculated Columns

Calculated columns are columns you create to enhance a Power Pivot table with your own formulas. When you enter calculated columns directly in the Power Pivot window, they become part of the source data you use to feed your pivot table. Calculated columns work at the row level. That is to say, the formulas you create

in a calculated column perform their operations based on the data in each individual row. For example, if you have a Revenue column and a Cost column in your Power Pivot table, you could create a new column that calculates [Revenue] minus [Cost]. This simple calculation is valid for each row in the data set.

Calculated measures are used to perform more complex calculations that work on an aggregation of data. These calculations are applied directly to a pivot table, creating a sort of virtual column that can't be seen in the Power Pivot window. Calculated measures are needed whenever you need to calculate based on an aggregated grouping of rows — for example, the sum of [Year2] minus the sum of [Year1].

Creating your first calculated column

Creating a calculated column works much like building formulas in an Excel table. Follow these steps to create a calculated column:

1. **Open the Power Pivot Formulas.xlsx sample file, activate the Power Pivot window (by clicking the Manage button on the Power Pivot Ribbon tab), and then select the InvoiceDetails tab.**

   In the table, you see an empty column on the far right, labeled Add Column.

2. **Click on the first blank cell in that column.**

3. **On the Formula bar, enter the following formula (as shown in Figure 6-1):**

   ```
   =[UnitPrice]*[Quantity]
   ```

4. **Press Enter.**

   The formula populates the entire column, and Power Pivot automatically renames the column to Calculated Column 1.

5. **Double-click on the column label and rename the column Total Revenue.**

   You can rename any column in the Power Pivot window by double-clicking the column name and entering a new name. Alternatively, you can right-click any column and choose the Rename option.

TIP

TIP

You can build calculated columns by clicking instead of typing. For example, rather than manually enter **=[UnitPrice]*[Quantity]**, you can enter the equal sign (=), click the UnitPrice column, type the asterisk (*), and then click the Quantity column. You can also enter your own static data. For example, you can enter a formula to calculate a 10-percent tax rate by entering **=[UnitPrice]*1.10**.

[Total Revenue] ▾ f_x =[UnitPrice]*[Quantity]

	InvoiceN...	Quantity	UnitCost	UnitPrice	Total Revenue
1	ORDST1022	1	59.29	119.95	119.95
2	ORDST1015	1	3290.55	6589.95	6589.95
3	ORDST1016	10	35	34.95	349.5
4	ORDST1017	50	91.59	189.95	9497.5
5	ORDST1018	1	59.29	119.95	119.95
6	INV1010	1	674.5	1349.95	1349.95
7	INV1011	1	91.25	189.95	189.95
8	INV1012	1	303.85	609.95	609.95
9	ORDST1020	1	59.29	119.95	119.95

FIGURE 6-1: Start the calculated column by entering an operation on the Formula bar.

Each calculated column you create is automatically available in any pivot table connected to the Power Pivot Data Model. You don't have to take any action to get your calculated columns into the pivot table. Figure 6-2 shows the Total Revenue calculated column in the PivotTable Fields List. These calculated columns can be used just as you would use any other field in the pivot table.

Row Labels	Sum of Total Revenue
Aaron Fitz Electrical	45668.4
Adam Park Resort	6238.5
Advanced Paper Co.	131930.45
Advanced Tech Satellite System	2278.7
American Science Museum	6357.8
Associated Insurance Company	1299.8
Astor Suites	174604.55
Atmore Retirement Center	39.95
Baker's Emporium Inc.	18418
Blue Yonder Airlines	26138.3
Boyle's Country Inns	1829.85
Breakthrough Telemarketing	91437
Castle Inn Resort	239.9
Central Communications LTD	36816.2

PivotTable Fields

Active All

Choose fields to add to report:

Search

InvoiceDetails
- InvoiceNumber
- Quantity
- UnitCost
- UnitPrice
- Total Revenue

FIGURE 6-2: Calculated columns automatically show up in the PivotTable Fields List.

TIP

If you need to edit the formula in a calculated column, find the calculated column in the Power Pivot window, click the column, and then make changes directly on the Formula bar.

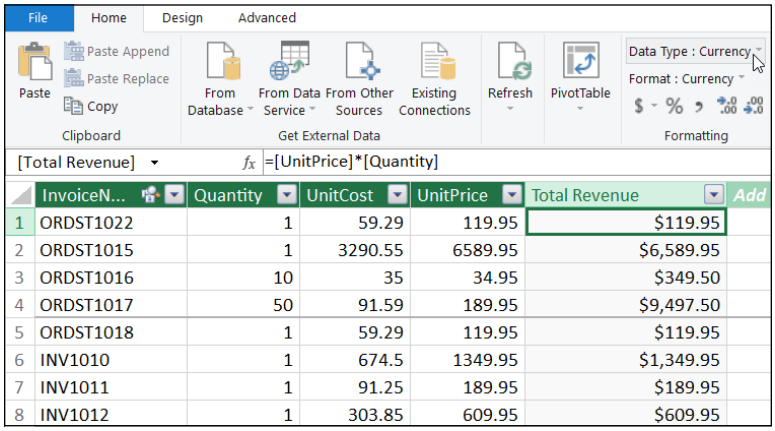
See Chapter 2 for a refresher on how to create a pivot table from Power Pivot.

Formatting calculated columns

You often need to change the formatting of Power Pivot columns to appropriately match the data within them. For example, you may want to show numbers as currency, remove decimal places, or display dates in a certain way.

You're by no means limited to formatting only calculated columns. The following steps can be used to format any column you see in the Power Pivot window:

1. **In the Power Pivot window, click on the column you want to format.**
2. **Go to the Home tab of the Power Pivot window and find the Formatting group (see Figure 6-3).**
3. **Use the options to alter the formatting of the column as you see fit.**



	InvoiceN...	Quantity	UnitCost	UnitPrice	Total Revenue	Add
1	ORDST1022	1	59.29	119.95	$119.95	
2	ORDST1015	1	3290.55	6589.95	$6,589.95	
3	ORDST1016	10	35	34.95	$349.50	
4	ORDST1017	50	91.59	189.95	$9,497.50	
5	ORDST1018	1	59.29	119.95	$119.95	
6	INV1010	1	674.5	1349.95	$1,349.95	
7	INV1011	1	91.25	189.95	$189.95	
8	INV1012	1	303.85	609.95	$609.95	

FIGURE 6-3: You can use the formatting tools found on the Power Pivot window's Home tab to format any column in the Data Model.

TIP

Veteran Excel pivot table users know that changing pivot table number formats one data field at a time is a pain. One fantastic feature of Power Pivot formatting is that any format you apply to the columns in the Power Pivot window is automatically applied to all pivot tables connected to the Data Model.

Referencing calculated columns in other calculations

As with all calculations in Excel, Power Pivot allows you to reference a calculated column as a variable in another calculated column. Figure 6-4 illustrates this concept with a new calculated column named Gross Margin. Notice that on the Formula bar, the calculation is using the following formula:

```
=[Total Revenue]-([UnitCost]*[Quantity])
```

[Gross Margin] =[Total Revenue]-([UnitCost]*[Quantity])

	InvoiceN...	Quantity	UnitCost	UnitPrice	Total Revenue	Gross Margin
1	ORDST1022	1	59.29	119.95	$119.95	60.66
2	ORDST1015	1	3290.55	6589.95	$6,589.95	3299.4
3	ORDST1016	10	35	34.95	$349.50	-0.5
4	ORDST1017	50	91.59	189.95	$9,497.50	4918
5	ORDST1018	1	59.29	119.95	$119.95	60.66
6	INV1010	1	674.5	1349.95	$1,349.95	675.45
7	INV1011	1	91.25	189.95	$189.95	98.7
8	INV1012	1	303.85	609.95	$609.95	306.1
9	ORDST1020	1	59.29	119.95	$119.95	60.66
10	ORDST1021	1	59.29	119.95	$119.95	60.66

FIGURE 6-4: The new Gross Margin calculation is using the previously created [Total Revenue] and calculated column.

Hiding calculated columns from end users

Because calculated columns can reference each other, you can imagine creating columns simply as helper columns for other calculations. You may not want your end users to see these columns in your client tools. (In this context, *client tools* refers to pivot tables, Power View dashboards, and Power Map.)

Similar to hiding columns on an Excel worksheet, Power Pivot allows you to hide any column. (It doesn't have to be a calculated column.) To hide columns, select the columns you want hidden, right-click the selection, and then choose the Hide from Client Tools option (as shown in Figure 6-5).

	InvoiceN...	Quantity	UnitCost	UnitPrice	Total Revenue
1	ORDST1022	1	59.29	119.95	
2	ORDST1015	1	3290.55	6589.95	$
3	ORDST1016	10	35	34.95	
4	ORDST1017	50	91.59	189.95	$
5	ORDST1018	1	59.29	119.95	
6	INV1010	1	674.5	1349.95	$
7	INV1011	1	91.25	189.95	
8	INV1012	1	303.85	609.95	
9	ORDST1020	1	59.29	119.95	
10	ORDST1021	1	59.29	119.95	$119.95

Create Relationship...
Navigate to Related Table
Copy
Insert Column
Delete Columns
Rename Column
Freeze Columns
Unfreeze All Columns
Hide from Client Tools
Column Width...
Filter
Description...

FIGURE 6-5: Right-click and select Hide from Client Tools.

REMEMBER

When a column is hidden, it doesn't show as an available selection in the PivotTable Fields List. However, if the column you're hiding is already part of the pivot report (meaning you've already dragged it onto the pivot table), hiding the column doesn't automatically remove it from the report. Hiding merely affects the ability to see the column in the PivotTable Fields List.

Note in Figure 6-6 that Power Pivot recolors columns based on their attributes. Hidden columns are subdued and grayed-out, whereas calculated columns that are not hidden have a darker (black) header.

	InvoiceN...	Quantity	UnitCost	UnitPrice	Total Revenue	Gross Margin
1	ORDST1022	1	59.29	119.95	$119.95	60.66
2	ORDST1015	1	3290.55	6589.95	$6,589.95	3299.4
3	ORDST1016	10	35	34.95	$349.50	-0.5
4	ORDST1017	50	91.59	189.95	$9,497.50	4918
5	ORDST1018	1	59.29	119.95	$119.95	60.66
6	INV1010	1	674.5	1349.95	$1,349.95	675.45
7	INV1011	1	91.25	189.95	$189.95	98.7
8	INV1012	1	303.85	609.95	$609.95	306.1

FIGURE 6-6: Hidden columns are grayed-out, and calculated columns have darker headings.

TIP

To unhide columns, select the hidden columns in the Power Pivot window, right-click on the selection, and then choose the Unhide from Client Tools option.

Utilizing DAX to Create Calculated Columns

Data Analysis Expressions, or DAX, is essentially the formula language that Power Pivot uses to perform calculations within its own construct of tables and columns. The DAX formula language comes supplied with its own set of functions. Some of these functions can be used in calculated columns for row-level calculations, and others are designed to be used in calculated measures to aggregate operations.

In this section, I touch on some of the DAX functions that you can leverage in calculated columns.

REMEMBER

The examples of DAX demonstrated in this chapter are meant to give you a sense of how calculated columns and calculated measures work. You can explore DAX more fully in Chapter 7.

Identifying DAX functions that are safe for calculated columns

Earlier in this chapter, you use the Formula bar within the Power Pivot window to enter calculations. Next to the Formula bar, you may have noticed the Insert Function button: the button labeled *fx*. It's similar to the Insert Function button in Excel. Clicking this button opens the Insert Function dialog box, shown in Figure 6-7. Using this dialog box, you can browse, search for, and insert the available DAX functions.

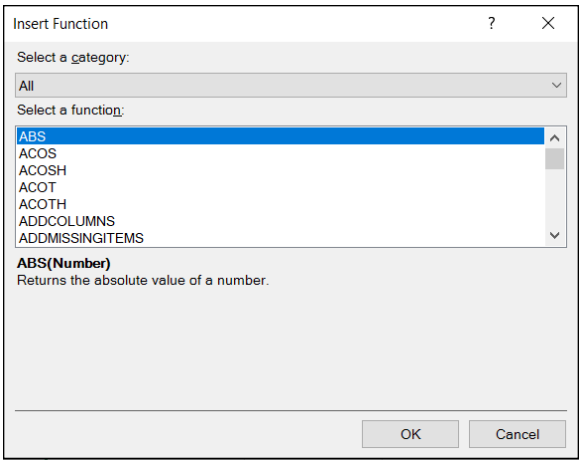


FIGURE 6-7: The Insert Function dialog box shows you all available DAX functions.

As you look through the list of DAX functions, notice that many of them look like the common Excel functions that most people are familiar with. But make no mistake: They aren't Excel functions. Whereas Excel functions work with cells and ranges, these DAX functions are designed to work at the table and column levels.

To understand what I mean, start a new calculated column on the Invoice Details tab. Click on the Formula bar and type a good old SUM function: `SUM([Gross Margin])`. The result is shown in Figure 6-8.

=sum([Gross Margin])

ntity	UnitCost	UnitPrice	Total Revenue	Gross Margin	Calculated Column 1
1	59.29	119.95	$119.95	60.66	928378.069999998
1	3290.55	6589.95	$6,589.95	3299.4	928378.069999998
10	35	34.95	$349.50	-0.5	928378.069999998
50	91.59	189.95	$9,497.50	4918	928378.069999998
1	59.29	119.95	$119.95	60.66	928378.069999998
1	674.5	1349.95	$1,349.95	675.45	928378.069999998
1	91.25	189.95	$189.95	98.7	928378.069999998
1	303.85	609.95	$609.95	306.1	928378.069999998

FIGURE 6-8: The DAX SUM function can only sum the column as a whole.

As you can see, the SUM function sums the entire column. This is because Power Pivot and DAX are designed to work with tables and columns. Power Pivot has no construct for cells and ranges. It doesn't even have column letters and row numbers on its grid. Though you would normally reference a range (as in an Excel SUM function), DAX basically takes the entire column.

The bottom line is that not all DAX functions can be used with calculated columns. Because a calculated column evaluates at the row level, only DAX functions that evaluate single data points can be used in a calculated column.

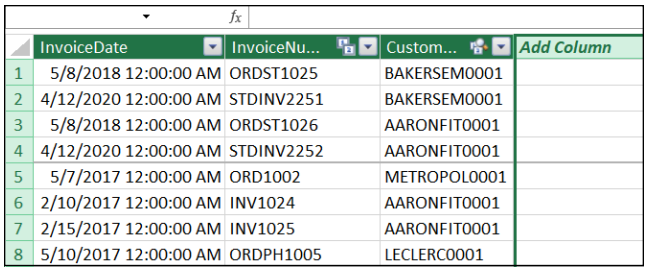
Here's a good rule of thumb: If the function requires an array or a range of cells as an argument, it isn't viable in a calculated column.

So, functions such as SUM, MIN, MAX, AVERAGE, and COUNT don't work in calculated columns. Functions that require only single data-point arguments work quite well in calculated columns: functions such as YEAR, MONTH, MID, LEFT, RIGHT, IF, and IFERROR.

Building DAX-driven calculated columns

To demonstrate the usefulness of employing a DAX function to enhance calculated columns, let's return to the walk-through example. Go to the Power Pivot window and select the InvoiceHeader tab on the Ribbon. If you've accidentally closed the Power Pivot window, you can open it by clicking the Manage command button on the Power Pivot Ribbon tab.

The InvoiceHeader tab, shown in Figure 6-9, contains an InvoiceDate column. Although this column is valuable in the raw table, the individual dates aren't convenient when analyzing the data with a pivot table. It would be beneficial to have a column for Month and a column for Year. This way, you could aggregate and analyze the data by month and year.



	InvoiceDate	InvoiceNu...	Custom...	Add Column
1	5/8/2018 12:00:00 AM	ORDST1025	BAKERSEM0001	
2	4/12/2020 12:00:00 AM	STDINV2251	BAKERSEM0001	
3	5/8/2018 12:00:00 AM	ORDST1026	AARONFIT0001	
4	4/12/2020 12:00:00 AM	STDINV2252	AARONFIT0001	
5	5/7/2017 12:00:00 AM	ORD1002	METROPOL0001	
6	2/10/2017 12:00:00 AM	INV1024	AARONFIT0001	
7	2/15/2017 12:00:00 AM	INV1025	AARONFIT0001	
8	5/10/2017 12:00:00 AM	ORDPH1005	LECLERC0001	

FIGURE 6-9: DAX functions can help enhance the invoice header data with Year and Month time dimensions.

For this endeavor, you use the DAX functions YEAR( ), MONTH( ), and FORMAT( ) to add some time dimensions to the Data Model. Follow these steps:

1. **In the InvoiceHeader table, click on the first blank cell in the empty column labeled Add Column, on the far right.**
2. **On the Formula bar, type** =YEAR([InvoiceDate]) **and then press Enter.**

   Power Pivot automatically renames the column to Calculated Column 1.
3. **Double-click on the column label and rename the column** Year**.**

4. **Starting in the next column, click on the first blank cell in the empty column labeled Add Column, on the far right.**

5. **On the Formula bar, type** =MONTH([InvoiceDate])**, and then press Enter.**

   Power Pivot automatically renames the column to Calculated Column 1.

6. **Double-click on the column label and rename the column** Month**.**

7. **Starting in the next column, click on the first blank cell in the empty column labeled Add Column, on the far right.**

8. **On the Formula bar, type** =FORMAT([InvoiceDate],"mmm") **and then press Enter.**

   Power Pivot automatically renames the column to Calculated Column 1.

9. **Double-click on the column label and rename the column** Month Name**.**

After completing these steps, you should have three new calculated columns similar to the ones shown in Figure 6-10.

	InvoiceDate	InvoiceNu...	Custom...	Year	Month	Month Name
1	5/8/2018 12:00:00 AM	ORDST1025	BAKERSEM0001	2018	5	May
2	4/12/2020 12:00:00 AM	STDINV2251	BAKERSEM0001	2020	4	Apr
3	5/8/2018 12:00:00 AM	ORDST1026	AARONFIT0001	2018	5	May
4	4/12/2020 12:00:00 AM	STDINV2252	AARONFIT0001	2020	4	Apr
5	5/7/2017 12:00:00 AM	ORD1002	METROPOL0001	2017	5	May
6	2/10/2017 12:00:00 AM	INV1024	AARONFIT0001	2017	2	Feb
7	2/15/2017 12:00:00 AM	INV1025	AARONFIT0001	2017	2	Feb
8	5/10/2017 12:00:00 AM	ORDPH1005	LECLERC0001	2017	5	May

FIGURE 6-10: Using DAX functions to supplement a table with Year, Month, and Month Name columns.

As I mention earlier in this chapter, creating calculated columns automatically makes them available through the PivotTable Fields List (see Figure 6-11).

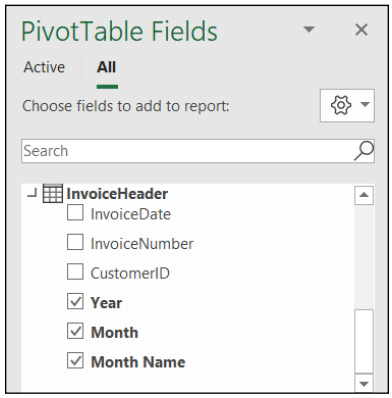


FIGURE 6-11: DAX calculations are immediately available in any connected pivot table.

Month sorting in Power Pivot–driven pivot tables

One of the more annoying aspects of Power Pivot is that it doesn't inherently know how to sort months. Unlike standard Excel, Power Pivot doesn't use the built-in custom lists that define the order of month names. Whenever you create a calculated column such as [Month Name] and place it into your pivot table, Power Pivot puts those months in alphabetical order. Figure 6-12 illustrates this in a pivot table designed to show average revenue by month.

Row Labels	Average of Total Revenue
⊟ Aaron Fitz Electrical	
Apr	$1,121.88
Feb	$826.75
Jan	$396.49
Mar	$248.14
May	$1,829.83
Sep	$59.95
⊟ Adam Park Resort	
Apr	$839.92
Jan	$599.50
May	$59.90
Sep	$2,399.95
⊟ Advanced Paper Co.	
Apr	$13,049.13
Jan	$359.80
⊟ Advanced Tech Satellite System	
Apr	$49.95
Jul	$138.24
May	$949.75

FIGURE 6-12: Month names in Power Pivot-driven pivot tables don't automatically sort in month order.

The fix for this problem is fairly easy. Open the Power Pivot window and select the Home tab. There, click the Sort by Column command button. The Sort by Column dialog box the opens, as shown in Figure 6-13.

The idea is to select the column you want sorted and then select the column you want to sort by. In this scenario, you want to sort Month Name by month.

After you confirm the change, it initially appears as though nothing has happened. The reason is that the sort order you defined isn't for the Power Pivot window. The sort order is applied to the pivot table. You can switch over to Excel to see the result in the pivot table (see Figure 6-14).

TIP

Pivot tables based on your data model will first inherit the formatting and sorting explicitly applied in the data model, then will apply any formatting you set in the pivot table itself. In other words, any formatting you apply in the pivot table itself will supersede the formatting and sorting applied via the Power Pivot window.

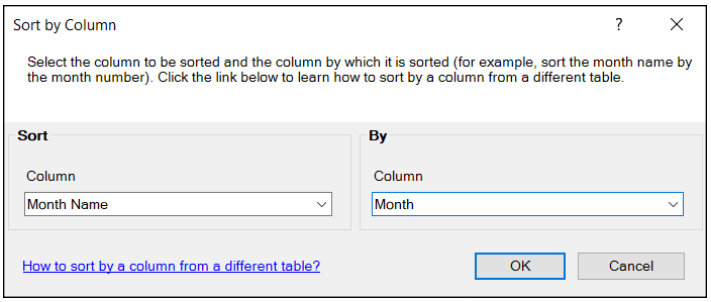


FIGURE 6-13: The Sort by Column dialog box lets you define how columns are sorted.

Row Labels	Average of Total Revenue
Aaron Fitz Electrical	
Jan	$396.49
Feb	$826.75
Mar	$248.14
Apr	$1,121.88
May	$1,829.83
Sep	$59.95
Adam Park Resort	
Jan	$599.50
Apr	$839.92
May	$59.90
Sep	$2,399.95
Advanced Paper Co.	
Jan	$359.80
Apr	$13,049.13
Advanced Tech Satellite System	
Apr	$49.95
May	$949.75
Jul	$138.24

FIGURE 6-14: The month names now show in the correct month order.

Referencing fields from other tables

Sometimes, the operation you're trying to perform with a calculated column requires you to utilize fields from other tables within the Power Pivot Data Model. For example, you may need to account for a customer-specific discount amount from the Customers table (see Figure 6-15) when creating a calculated column in the InvoiceDetails table.

	Custome...	CustomerName	Discount Amount	Address
1	DOLLISCO0001	Dollis Cove Resort	11%	765 Kingway
2	GETAWAYI0001	Getaway Inn	10%	234 E Cannon Ave.
3	HOMEFURN0001	Home Furnishings Limited	25%	234 Heritage Ave.
4	JOHNSONK0001	Johnson, Kimberly	12%	5678 S. 42nd Ave.
5	KELLYCON0001	Kelly Consulting	5%	123 Yeo
6	KENSINGT0001	Kensington Gardens Resort	13%	12345 Redmond Rd
7	HAMPTONV0001	Hampton Village Eatery	20%	234 Hampton Village
8	HEALTHYC0001	Healthy Concepts	11%	1234 Westown Road

FIGURE 6-15: The discount amount in the Customers table can be used in a calculated column in another table.

To accomplish this, you can use a DAX function named RELATED. Similar to VLOOKUP in standard Excel, the RELATED function allows you to look up values from one table in order to use them in another.

Follow these steps to create a new calculated column that displays a discounted amount for each transaction in the InvoiceDetails table:

1. **In the InvoiceDetails table, click on the first blank cell in the empty column labeled Add Column, on the far right.**
2. **On the Formula bar, type** =RELATED(**.**

   As soon as you enter the open parenthesis, a menu of available fields (shown in Figure 6-16) is displayed. Note that the items in the list represent the table name followed by the field name in brackets. In this case, you're interested in the Customers[Discount Amount] field.

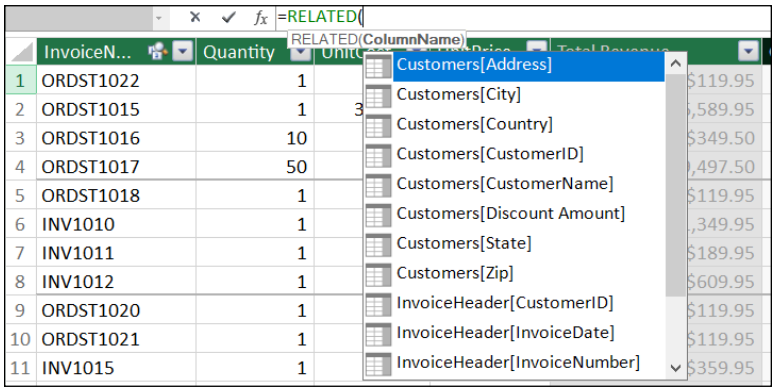


FIGURE 6-16: Use the RELATED function to look up a field from another table.

3. **Double-click the Customers[Discount Amount] field and then press Enter.**

   Power Pivot automatically renames the column to Calculated Column 1.
4. **Double-click on the column label and rename the column** Discount%**.**
5. **Starting in the next column, click on the first blank cell in the empty column labeled Add Column, on the far right.**
6. **On the Formula bar, type** =[UnitPrice]*[Quantity]*(1-[Discount%]) **and then press Enter.**

   Power Pivot automatically renames the column to Calculated Column 1.
7. **Double-click on the column label and rename the column** Discounted Revenue**.**

The reward for your efforts is a new column that uses the discount percent from the Customers table to calculate discounted revenue for each transaction. Figure 6-17 illustrates the new calculated column.

=RELATED(Customers[Discount Amount])

	UnitCost	UnitPrice	Total Revenue	Discount%	Discounted Revenue
1	59.29	119.95	$119.95	13.00%	$104.36
1	3290.55	6589.95	$6,589.95	11.00%	$5,865.06
10	35	34.95	$349.50	9.00%	$318.05
50	91.59	189.95	$9,497.50	6.00%	$8,927.65
1	59.29	119.95	$119.95	9.00%	$109.15
1	674.5	1349.95	$1,349.95	15.00%	$1,147.46
1	91.25	189.95	$189.95	17.00%	$157.66
1	303.85	609.95	$609.95	5.00%	$579.45

FIGURE 6-17: The final discount amount calculated column using the Discount% column from the Customers table.

REMEMBER

The RELATED function leverages the relationships you defined when creating the data model to perform the lookup. So, this list of choices contains only the fields that are available based on the relationships you defined.

Nesting functions

In the example from the preceding section, you first create a Discount% column using the RELATED function, and then you use that column in another calculated column to calculate the discount amount.

You don't necessarily have to create multiple calculated columns to accomplish a task like this one. You could instead nest the RELATED function into the discount amount calculation. The following line shows the syntax for the nested calculation:

```
=[UnitPrice]*[Quantity]*(1-RELATED(Customers[Discount Amount]))
```

As you can see, *nesting* simply means to embed functions within a calculation. In this case, rather than use the RELATED function in a separate Discount% field, you can embed it directly into the discounted revenue calculation.

Nesting functions can definitely save time and even improve performance in larger data models. On the other hand, complicated nested functions can be harder to read and understand.

Understanding Calculated Measures

You can enhance the functionality of your Power Pivot reports by using a kind of calculation called a calculated measure. *Calculated measures* are not applied to the Power Pivot window like calculated columns. Instead, they're applied directly to the pivot table, creating a sort of virtual column that isn't visible in the Power Pivot window. You use calculated measures when you need to calculate based on an aggregated grouping of rows.

Creating a calculated measure

Imagine that you want to show the difference in unit costs between the years 2020 and 2019 for each of your customers. Think about what technically has to be done to achieve this calculation: You have to figure out the sum of unit costs for 2020, determine the sum of unit costs for 2019, and then subtract the sum of 2020 from the sum of 2019. This calculation simply can't be completed using calculated columns. Using calculated measures is the only way to calculate the cost variance between 2020 and 2019.

Follow these steps to create a calculated measure:

REMEMBER

1. **Start with a pivot table created from a Power Pivot Data Model.**

   The `Power Pivot Formulas.xlsx` workbook contains the Calculated Measures tab with a pivot table already created.

2. **Click the Power Pivot tab on the Excel Ribbon, and choose Measures ➪ New Measure.**

   This step opens the Measure dialog box, shown in Figure 6-18.

3. **In the Measure dialog box, set the following inputs:**

   - *Table name:* Choose the table you want to contain the calculated measure when looking at the PivotTable Fields List. Don't sweat this decision too much. The table you select has no bearing on how the calculation works. It's simply a preference on where you want to see the new calculation within the PivotTable Fields List.
   - *Measure name:* Give the calculated measure a descriptive name.
   - *Description:* Enter a friendly description to document what the calculation does.
   - *Formula:* Enter the DAX formula that will calculate the results of the new field.

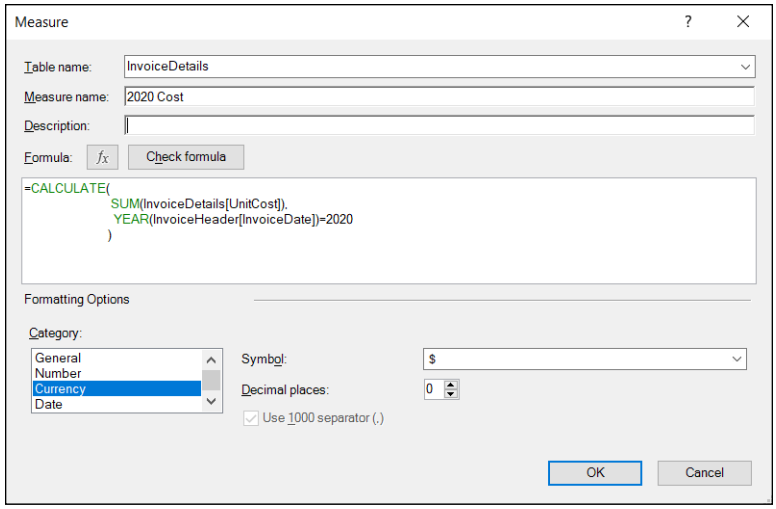


FIGURE 6-18: Creating a new calculated measure.

In this example, you use the following DAX formula:

```
=CALCULATE(
SUM(InvoiceDetails[UnitCost]),
YEAR(InvoiceHeader[InvoiceDate])=2020
)
```

This formula uses the CALCULATE function to sum the Total Revenue column from the InvoiceDetails table, where the Year column in the InvoiceHeader is equal to 2020.

- *Formatting Options:* Specify the formatting for the calculated measure results.

4. **Click the Check Formula button to ensure that there are no syntax errors.**

   If your formula is well formed, you see the message `No errors in formula`. If the formula has errors, you see a full description.

5. **Click the OK button to confirm the changes and close the dialog box.**

   You see your newly created calculated measure in the pivot table.

6. **Repeat Steps 2–5 for any other calculated measure you need to create.**

In this example, you need a measure to show the 2019 cost:

```
=CALCULATE(
SUM(InvoiceDetails[UnitCost]),
YEAR(InvoiceHeader[InvoiceDate])=2019
)
```

You also need a measure to calculate the variance:

```
=[2020 Revenue]-[2019 Revenue]
```

Figure 6-19 illustrates the newly created calculated measures. The calculated measures are applied to each customer, displaying the variance between their 2020 and 2019 costs. As you can see, each calculated measure is available for selection in the PivotTable Fields List.

2020 Cost	2019 Cost	2020 vs 2019
$3,239	$7,922	($4,683)
$1,244	$1,308	($64)
$56	$56	$0
$225	$186	$39
$324		$324
$14,060	$16,791	($2,731)
$1,859		$1,859
$2,525	$6,781	($4,256)
$3,436	$3,020	$416
$2,906	$9,069	($6,163)
$8		$8
$59		$59
$35,461	$2,233	$33,228

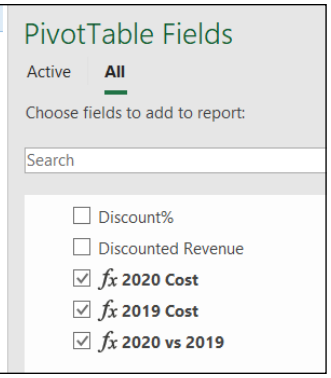


FIGURE 6-19: Calculated measures can be seen in the PivotTable Fields List.

TIP

Always attempt to achieve readability by using carriage returns and spaces. In Figure 6-18, the DAX calculation is entered with carriage returns and spaces. This is purely for readability purposes. DAX ignores white spaces and isn't case sensitive, so it's quite forgiving on how you structure the calculation.

Editing and deleting calculated measures

You may find that you need to either edit or delete a calculated measure. You can do so by following these steps:

1. **Click anywhere inside the pivot table, click the Power Pivot tab on the Excel Ribbon, and choose Measures ➪ Manage Measures.**

   This step opens the Manage Measures dialog box, shown in Figure 6-20.

2. **Select the target calculated measure, and click one of these two buttons:**

   - *Edit:* Opens the Measure dialog box, where you can make changes to the calculation setting.
   - *Delete:* Opens a message box asking you to confirm that you want to remove the measure. After you confirm, the calculated measure is removed.

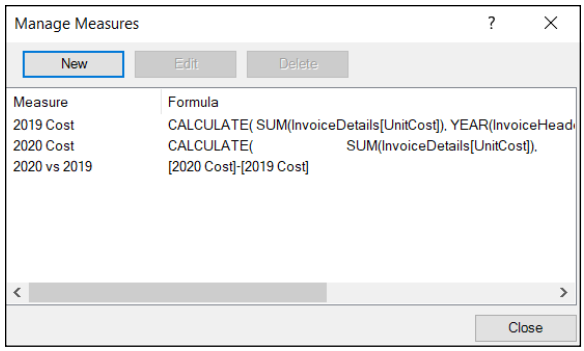


FIGURE 6-20: The Manage Measures dialog box lets you edit or delete your calculated measures.

Free Your Data with Cube Functions

Cube functions are Excel functions that can be used to access the data in a Power Pivot Data Model outside the constraints of a pivot table. Although cube functions aren't technically used to create calculations themselves, they can be used to free PowerPivot data so that it can be used with formulas you may have in other parts of your Excel spreadsheet.

One of the easiest ways to start exploring cube functions is to allow Excel to convert your Power Pivot pivot table into cube functions. The idea is to tell Excel to replace all cells in the pivot table with a formula that connects back to the Power Pivot Data Model.

The `Power Pivot Formulas.xlsx` workbook contains a Cube Functions tab with a pivot table already created. Place your cursor anywhere inside the pivot table, and then select PivotTable Analyze➪ OLAP Tools➪ Convert to Formulas.

After a second or two, the cells that used to house a pivot table are now homes for Cube formulas. Figure 6-21 illustrates the cube functions.

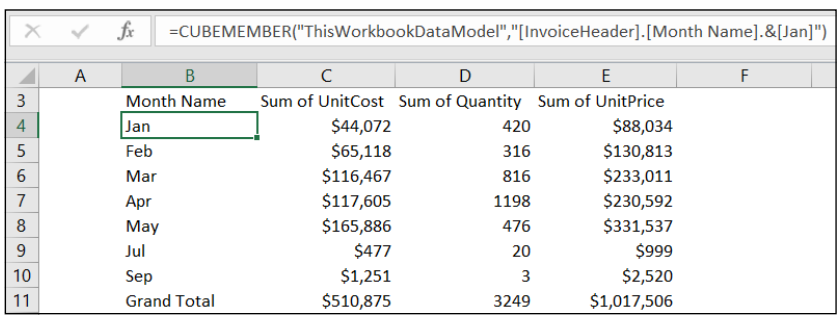


FIGURE 6-21: These cells are now a series of Cube functions.

If your pivot table contains a report filter field, the dialog box shown in Figure 6-22 activates. This dialog box gives you the option of converting your filter drop-down selectors to Cube formulas. If you select this option, the drop-down selectors are removed, leaving a static formula.

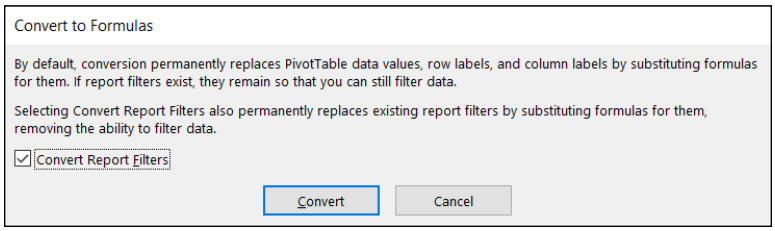
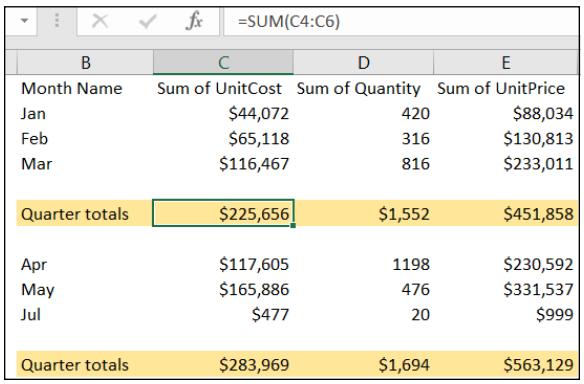


FIGURE 6-22: Excel gives you the option of converting your report filter fields.

If you need to have your filter drop-down selectors intact so that you can continue to change the selections in the filter field interactively, be sure to leave the Convert Report Filters option unchecked when clicking the Convert button.

Now that the values you see are no longer part of a PivotTable object, you can insert rows and columns, you can add your own calculations, or you can combine the data with other formulas in your spreadsheet. For instance, in Figure 6-23, you can see I added two quarter total rows in the middle of the pivot data. I'm only able to do this because the pivot data is no longer a single PivotTable object. Instead, each cell is a cube formula that can be moved around like any other formula.



	B	C	D	E
	Month Name	Sum of UnitCost	Sum of Quantity	Sum of UnitPrice
	Jan	$44,072	420	$88,034
	Feb	$65,118	316	$130,813
	Mar	$116,467	816	$233,011
	Quarter totals	$225,656	$1,552	$451,858
	Apr	$117,605	1198	$230,592
	May	$165,886	476	$331,537
	Jul	$477	20	$999
	Quarter totals	$283,969	$1,694	$563,129

FIGURE 6-23: Cube functions give you the flexibility of restructuring your pivot data without losing the link to your data model.

The bottom line is that cube functions give you the flexibility to free your Power Pivot data from the confines of a pivot table and then use it in all sorts of ways by simply moving formulas around.

IN THIS CHAPTER

- Understanding the fundamentals of the DAX language
- Looking at filter context

Chapter 7
Diving into DAX

This chapter rounds your exploration of Power Pivot with a closer look at the DAX formula language. In Chapter 6, you use a bit of DAX when exploring the mechanics of adding your own calculated columns and calculated measures. Now that you know where DAX expressions fit in your Power Pivot data model, it's time to take a dive into DAX.

In this chapter, you gain an understanding of the fundamentals of building DAX expressions and explore some commonly used DAX functions that will help you start your DAX journey.

ON THE WEB

The sample file for this chapter is called `Adventure Works.xlsx` and can be found on this book's companion website at `www.dummies.com/go/excelpowerpivotpowerqueryfd2e`. The sample file contains a ready-to-use Power Pivot model with all the data need to follow along in this chapter.

DAX Language Fundamentals

Just like Excel functions, every DAX function outputs a result based on arguments you provide. The difference is that, in Excel, the arguments you use provide cell references such as A2:A10, and in DAX, you use table and column names. For instance, you could use `SUM(A2:A10)` to get the sum of the given Excel range. However, a similar DAX formula would look more like the following:

```
SUM(Sales[OrderQuantity])
```

Notice I'm using both the table name, `Sales`, and the column name, `[OrderQuantity]`. When referencing a column in a DAX function, you always include the table name. If the name of the table contains a space or special characters, use single quotes around the table name, like this:

```
SUM('Internet Sales'[OrderQuantity])
```

Let's create a starting measure by following these steps:

1. **Open the `Adventure Works.xlsx` file, select the Power Pivot tab on the Excel Ribbon, and choose Measures ➪ New Measure.**

   The Measure dialog box, shown in Figure 7-1, appears.

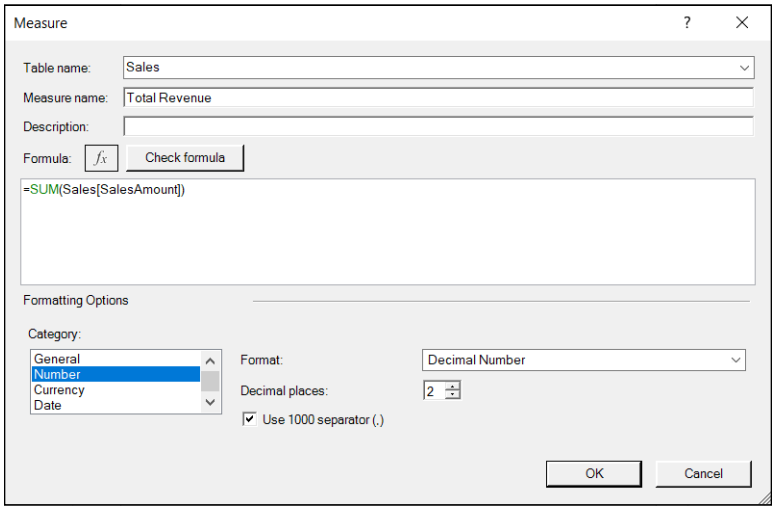


FIGURE 7-1: Creating a new measure that calculates Total Revenue.

2. **In the Measure dialog box, set the following inputs:**

   - *Table name:* Choose the Sales table from the drop-down to ensure you can easily find your new measure within the PivotTable Field List.
   - *Measure name:* Enter **Total Revenue** as the measure name.
   - *Formula:* Enter the following DAX formula:

     ```
     =SUM(Sales[SalesAmount])
     ```

   - *Formatting Options:* Choose to format the results as a number with two decimal places and a thousands separator (refer to Figure 7-1).

3. **Click OK to close the dialog box and create your measure.**

   At this point, you'll need to create a pivot table to see your new measure in action.

4. **Select the Power Pivot tab on the Excel Ribbon and choose the Manage command.**

   The Power Pivot window opens.

5. **Choose Home ➪ PivotTable.**

   The Create PivotTable dialog box appears.

6. **Choose the New Worksheet option and click OK.**

7. **Find the Total Revenue measure under the Sales table in the Pivot Table Field List and place a check mark next to it.**

In Figure 7-2, calculated measures have a distinct icon in the Pivot Table Field List, making it easy to identify them.

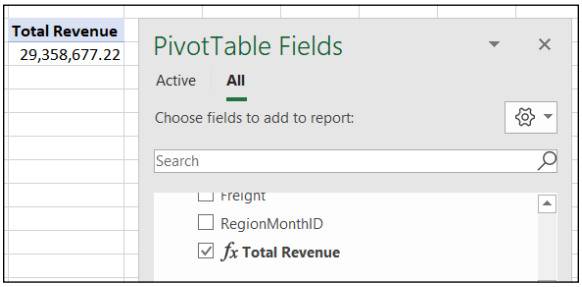


FIGURE 7-2: The results of a calculated measure can be seen by adding it to a pivot table.

Repeat Steps 1 through 3 each time you want to add a new measure to your pivot table. Take a moment to create a new measure with the following inputs:

- **Table name:** Sales.
- **Measure name:** Total Units.
- **Formula:** Enter the following DAX formula:

```
=SUM(Sales[OrderQuantity])
```

- **Formatting Options:** Choose to format the results as a whole number with zero decimal places and a thousand separator (see Figure 7-3).

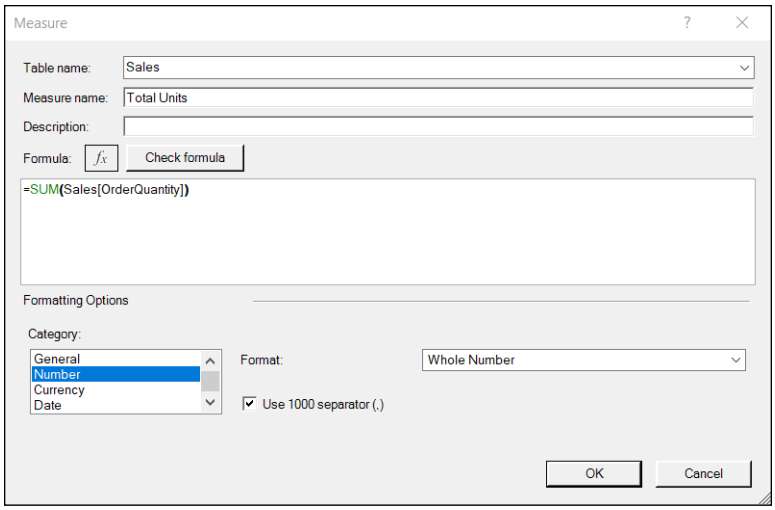


FIGURE 7-3: Creating a new measure that will calculate Total Units.

DAX allows you to use measures as arguments in other measures. Figure 7-4 illustrates the creation of a new measure called Revenue per Unit using the newly created measures:

```
=[Total Revenue]/[Total Units]
```

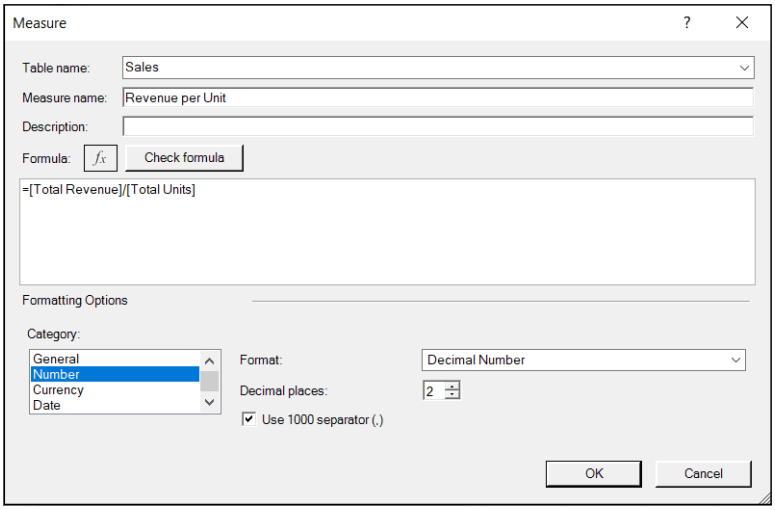


FIGURE 7-4: DAX allows you to use existing measure as arguments in other measures.

Measure names are encapsulated with brackets but not prefixed with a table name. This makes it easy to look at your formulas and distinguish measures from columns, because columns will have a table prefix while measure will not.

TIP

DAX is not a case-sensitive language, so it's quite forgiving of uppercase and lowercase variances. DAX also ignores spaces so you can freely add carriage returns and extra spaces to improve the readability of your formulas.

Calculated measures on a pivot table will automatically recalculate when the underlying data is refreshed, when the pivot table is filtered with a slicer or a page filter, or when you change the structure of your pivot table. For instance, Figure 7-5 illustrates how the measures recalculate to show the appropriate results for the dimensions included in the pivot.

Row Labels	Total Revenue	Total Units	Revenue per Unit
Australia	9,061,000.58	13,345	678.98
Canada	1,977,844.86	7,620	259.56
Central	3,000.83	20	150.04
France	2,644,017.71	5,558	475.71
Germany	2,894,312.34	5,625	514.54
Northeast	6,532.47	27	241.94
Northwest	3,649,866.55	8,993	405.86
Southeast	12,238.85	39	313.82
Southwest	5,718,150.81	12,265	466.22
United Kingdon	3,391,712.21	6,906	491.13
Grand Total	**29,358,677.22**	**60,398**	**486.09**

Row Labels	Total Revenue	Total Units	Revenue per Unit
Accessories	700,759.96	36,092	19.42
Bikes	28,318,144.65	15,205	1,862.42
Clothing	339,772.61	9,101	37.33
Grand Total	**29,358,677.22**	**60,398**	**486.09**

FIGURE 7-5: DAX formulas recalculate to show appropriate results based on the dimensions included in the pivot table.

TIP

Another handy aspect of measures: They aren't tied to a particular pivot table. Calculated measures are created in the Power Pivot data model, which means any pivot table that uses the internal data model as its source can make use of them.

Using DAX operators

The DAX language allows most of the standard operators you're accustomed to using in Excel. In fact, you'll notice I used a division operator in Figure 7-4. Table 7-1 lists the operators DAX allows when building your calculated measures.

TABLE 7-1 DAX Operators

Operator	Purpose	Example
()	Parentheses are used to establish mathematical order of operations.	([Measure1]+[Measure2])*10
+	Mathematical operators are used to define the operation to be performed.	[Measure1]+[Measure2]
–		[Measure1]-[Measure2]
*		[Measure1]*[Measure2]
/		[Measure1]/[Measure2]
^		[Measure1]^[Measure2]
=	Comparison operators are used to evaluate the contrast between values.	[Measure1]=[Measure2]
>		[Measure1]<>[Measure2]
<		[Measure1]>=[Measure2]
&	A single ampersand is used to concatenate a string of values together.	"Total is: " & [Measure1]
AND	The AND logical operator is used to evaluate a condition between two expressions. DAX allows the word AND or the use of two ampersands. They both apply the same behavior.	[Measure1] >100 AND [Measure2] <200
&&		[Measure1] >100 && [Measure2] <200
OR	The OR logical operator can be used to evaluate a condition between two expressions. DAX allows the word OR or the use of two pipe characters. They both apply the same behavior.	[Measure1] >0 OR [Measure2] >0
||		[Measure1] >0 || [Measure2] >0
NOT	The NOT logical operation can be used to return a true or false based on a comparison to a defined expression. DAX allows the word NOT or the use of the exclamation point. They both apply the same behavior.	NOT ([Measure1] = [Measure2])
!		! ([Measure1] = [Measure2])

Applying conditional logic in DAX

DAX enables conditional logic checks through the `IF` function. Like the `IF` function in Excel, the DAX version requires three arguments:

- The condition to test
- The value to return if the condition is true
- The value to return if the condition is false

For example, the following DAX formula will return `Tier1` if the referenced measure, `Total Revenue`, is greater than 100,000; otherwise, `Tier2` will be returned:

```
=IF([Total Revenue] > 100000, "Tier1", "Tier2")
```

It's not uncommon to want to return a blank if a logic check fails. In these situations, you can employ the `BLANK` function. The following formula performs the same logic check as the previous formula, except it returns blank if the condition check fails:

```
=IF([Total Revenue] > 100000, "Tier1", BLANK())
```

You can go the other way and check if an expression is blank by using the `ISBLANK` function. `ISBLANK` returns either `TRUE` or `FALSE`. As an example, the following statement checks the Name column in the Customer table for a blank value. If the name is blank, the word `Classifed` is returned; otherwise, the customer name is returned.

```
=IF(ISBLANK(Customer[Name]), "Classified", Customer[Name])
```

You may find the need to use nested `IF` statements to apply multiple conditional checks in a single measure. DAX allows this in the same way Excel does in traditional worksheet formulas. This next formula illustrates a nested `IF` statement that returns a text label based on the conditional check of the [ListPrice] column in the Products table:

```
=IF(Products[ListPrice] > 1000, "A",
     IF(Products[ListPrice] > 500, "B",
         IF(Products[ListPrice] > 100, "C", "D")
         )
     )
```

REMEMBER

When building larger formulas such as the previous nested `IF` statement, it's often helpful to make use of carriage returns and indentations to improve the readability of your formula. There is no formatting standard per se. The idea is to apply formatting that makes it easy for you to read, find errors, and edit as needed.

As useful as nested `IF` statements can be, they can get unwieldy and difficult to read when they require many nested layers. In such cases, consider using the `SWITCH` function to perform the same complex logic checks in a much cleaner way.

The following formula uses the cleaner SWITCH function to perform the same operation as the previous nested IF statement:

```
=SWITCH(
         TRUE(),
         Products[ListPrice] > 1000, "A",
         Products[ListPrice] > 500, "B",
         Products[ListPrice] > 100, "C",
         "D"
         )
```

Finally, DAX offers the IFERROR function to check for an error in an expression and then return a specified result if an error is encountered. The following example uses the IFERROR function to perform the classic check for zero before dividing two measures. Here, if the operation produces an error, a 0 is returned.

```
=IFERROR([Total Cost]/[Units Sold],0)
```

TIP

Although it doesn't quite fit under the umbrella of conditional logic, it's worth mentioning the DIVIDE function. The DIVIDE function provides a safe method of dividing two values without the need to check for potential errors due to a 0 in the denominator or any such nonsense. Simply enter the numerator, denominator, and value to return if an error occurs. Here's a simple example where we're dividing the [Total Cost] measure by the [Units Sold] measure:

```
=DIVIDE([Total Cost], [Units Sold],0)
```

Working with DAX aggregate functions

Aggregate functions live up to their namesake by calculating an aggregated value from all the records in a given column. DAX offers the obligatory Fab Four of aggregate functions: SUM, AVERAGE, MIN, and MAX. These functions are used just the way you think they would be:

```
=SUM(Sales[Sales Amount])
=AVERAGE(Sales[Sales Amount])
=MIN(Sales[Sales Amount])
=MAX(Sales[Sales Amount])
```

There are, however, a few points to keep in mind when working with the SUM, AVERAGE, MIN, and MAX aggregate functions:

- They work only on columns that have a numeric or date data type.
- They ignore any text or blank values in the target column.
- They only aggregate rows after all filters and slicers have been applied to the source data. This means the results returned will be an aggregation of only the rows that met the complete filter context for the pivot table.

Another set of aggregate functions are those that return a count of some sort. These include the following:

- `COUNT`: Returns the count of non-blank numeric or date values in a given column.
- `COUNTA`: Returns the count of all values in a given column regardless of data type.
- `COUNTBLANK`: Returns the count of blank cells in a given column.
- `COUNTROWS`: Returns the number of rows in a table.
- `DISTINCTCOUNT`: Returns the number of *unique* values in a given column. If a value appears more than once, it will be counted only once. It's worth noting that `DISTINCTCOUNT` will count blanks as another unique value, adding 1 to the count for `BLANK`.

Exploring iterator functions and row context

In some situations, you can't rely on standard aggregation functions to return correct answers because the math needed only works when applied to individual rows. To fully understand what this means, take a gander at Figure 7-6.

Row Labels	Sum of OrderQuantity	Sum of UnitPrice
SO72656	8	2480.91
SO70714	8	2434.92
SO58845	8	691.91
SO72927	7	1347.79
SO71961	7	2406.93
SO74869	7	115.92
SO64542	7	2406.93
SO61412	7	2429.93
SO62984	7	2425.93
SO64042	7	2399.23
SO60233	7	2406.93
SO58572	7	750.74
SO54784	7	233.23
SO54042	7	145.42

FIGURE 7-6: I need a calculated measure that returns OrderQuantity * UnitPrice as the Realized Sales.

For each OrderNumber, I'm showing the [Sum of OrderQuantity] and the [Sum of UnitPrice]. I need a calculated measure to get [Realized Sales], which is simply [OrderQuantity]*[UnitPrice].

To save time, I've already created a [Realized Sales] measure, which you can review by selecting the Power Pivot tab on the Excel Ribbon and choosing Measures➪Manage Measures. Select the [Realized Sales] measure and then click the Edit button to see the dialog box shown in Figure 7-7. The formula being used is:

```
=Sum(Sales[OrderQuantity])*Sum(Sales[UnitPrice])
```

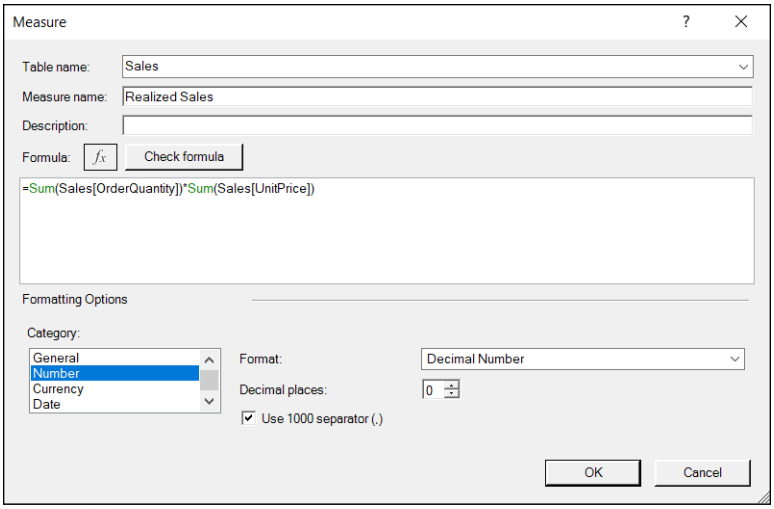


FIGURE 7-7: The [Realized Sales] measure uses a simple Sum aggregator function for each column.

As you can see in Figure 7-8, adding the [Realized Sales] measure to the pivot table shows promising results. The math makes sense for each row, and the results look accurate. However, you have to consider that this view is at the individual OrderNumber level (the most granular you can get with this data). Check out Figure 7-9 to see what happens when you replace [OrderNumber] with Region. The [Realize Sales] measure all becomes overly inflated.

The reason for the breakdown is relatively clear if you stop and look at the results. For the Australia Region, you're showing 13,345 units sold at a UnitPrice of 9,061,000 each. Well, that simply can't be true. UnitPrice and OrderQuantity need to be multiplied at each individual-row level to get an accurate dollar per quantity *before* any aggregation is performed.

Row Labels	Sum of OrderQuantity	Sum of UnitPrice	Realized Sales
SO72656	8	2480.91	19,847
SO70714	8	2434.92	19,479
SO58845	8	691.91	5,535
SO72927	7	1347.79	9,435
SO71961	7	2406.93	16,849
SO74869	7	115.92	811
SO64542	7	2406.93	16,849
SO61412	7	2429.93	17,010
SO62984	7	2425.93	16,982
SO64042	7	2399.23	16,795
SO60233	7	2406.93	16,849
SO58572	7	750.74	5,255
SO54784	7	233.23	1,633

FIGURE 7-8: The [Realized Sales] measure looks good at the granular OrderNumber level.

Row Labels	Sum of OrderQuantity	Sum of UnitPrice	Realized Sales
Australia	13,345	9,061,000.58	120,919,052,799
Canada	7,620	1,977,844.86	15,071,177,849
Central	20	3,000.83	60,017
France	5,558	2,644,017.71	14,695,450,456
Germany	5,625	2,894,312.34	16,280,506,902
Northeast	27	6,532.47	176,377
Northwest	8,993	3,649,866.55	32,823,249,895
Southeast	39	12,238.85	477,315
Southwest	12,265	5,718,150.81	70,133,119,712
United Kingdom	6,906	3,391,712.21	23,423,164,528
Grand Total	**60,398**	**29,358,677.22**	**1,773,205,386,776**

FIGURE 7-9: The math falls apart when you move to any granularity above OrderNumber (such as Region).

In other words, the measure needs to retain what is known as *row context.* When a measure retains row context, it sees and can interact with values for each row in which it performs its intended operation. Unfortunately, the `SUM` function has no row context, no insight into individual row data. `SUM` aggregates all the data in the specified column without capturing or noticing individual row data.

This is where you turn to iterator functions. *Iterator functions*, sometimes referred to as *X-functions*, are designed to iterate through and perform operations on each individual row record before aggregating results. The Fab Four aggregate functions have iterator versions of themselves: `SUMX`, `AVERAGEX`, `MINX`, and `MAXX`.

Iterator functions require two arguments: the table to iterate through and an expression to apply to each row. As an example, you can use the following formula (see Figure 7-10):

```
=SUMX(Sales, Sales[OrderQuantity] * Sales[UnitPrice])
```

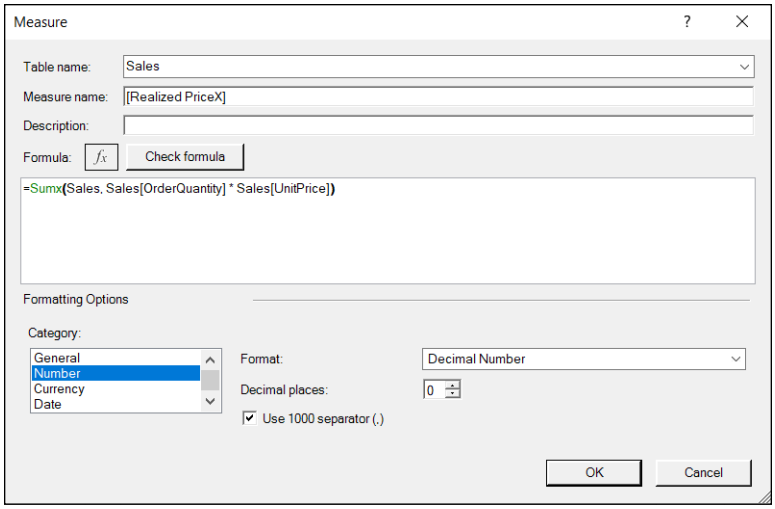


FIGURE 7-10: Using the SUMX function enables the measure to retain row context when calculating the defined expression.

This formula tells the measure to iterate through the Sales table, run Sales[OrderQuantity] * Sales[UnitPrice] on each row, and *then* sum all the calculated products into an aggregated result.

Figure 7-11 demonstrates the difference between the original [Realized Sales] measure using SUM and the new [Realized PriceX] measure using SUMX.

Row Labels	Realized Sales	[Realized PriceX]
Australia	120,919,052,799	9,061,001
Canada	15,071,177,849	1,977,845
Central	60,017	3,001
France	14,695,450,456	2,644,018
Germany	16,280,506,902	2,894,312
Northeast	176,377	6,532
Northwest	32,823,249,895	3,649,867
Southeast	477,315	12,239
Southwest	70,133,119,712	5,718,151
United Kingdom	23,423,164,528	3,391,712
Grand Total	**1,773,205,386,776**	**29,358,677**

FIGURE 7-11: The new [Realized PriceX] measure remains accurate now at every aggregation.

It's worth mentioning that X-functions can be used like standard aggregate functions. For instance, the following formulas produce the same result with little to no difference in performance:

```
=SUMX(Sales, Sales[Sales Amount])
=SUM(Sales[Sales Amount]
```

The truth is, you can use iterator functions — SUMX, AVERAGEX, MINX, and so on — exclusively, without getting into too much trouble.

Understanding Filter Context

Filter context is a topic that eludes most analysts first starting with DAX. It's a bit difficult to visualize without an example to work through, so let's jump in with a simple scenario.

Imagine you've been given the Excel table shown in Figure 7-12 and asked to calculate the sum of Sales Amount for red bikes sold in the Northeast. Before you can calculate the sum, you need to apply the required filters by using the filter drop-downs to select the following:

- **[Market]:** 'Northeast'
- **[Business Segment]:** 'Bikes'
- **[Color]:** 'Red'

	A	B	C	D	E
1				Calculate Sum of Sales Amount>>	
2					
3	Market	Customer	Business Segment	Color	Sales Amount
4	Southeast	Trusted Catalog Store	Bikes	Silver	$6,120
5	Southeast	Trusted Catalog Store	Bikes	Black	$4,050
6	Southeast	Trusted Catalog Store	Bikes	Black	$6,075
7	Southeast	Trusted Catalog Store	Clothing	White	$46
8	Southeast	Trusted Catalog Store	Bikes	Silver	$2,040
9	Southeast	Trusted Catalog Store	Bikes	Black	$6,075
10	Southeast	Trusted Catalog Store	Bikes	Silver	$6,120
11	Southeast	Trusted Catalog Store	Bikes	Silver	$4,080

FIGURE 7-12: You need to calculate the sum of Sales Amount in cell E1, but only after applying filters.

In a sense, these filter selections change the state of your source data and establish a new context for your final calculation. The calculation result shown in Figure 7-13 is accurate for the specific state you put the data in by filtering.

	A	B	C	D	E
1				Calculate Sum of Sales Amount>>	$1,614,418
2					
3	Market	Customer	Business Segment	Color	Sales Amount
30	Northeast	Weekend Tours	Bikes	Red	$4,374
31	Northeast	Weekend Tours	Bikes	Red	$4,294
32	Northeast	Weekend Tours	Bikes	Red	$2,147
33	Northeast	Weekend Tours	Bikes	Red	$875
34	Northeast	Weekend Tours	Bikes	Red	$839
35	Northeast	Weekend Tours	Bikes	Red	$419
36	Northeast	Weekend Tours	Bikes	Red	$839
37	Northeast	Weekend Tours	Bikes	Red	$2,624

FIGURE 7-13: The resulting answer is for the specific filter context applied.

When you boil away the complexities, filter context can generally be described as the state of your source data after you've applied all needed filters.

In the world of DAX, not only are calculations run against a filter context, but every cell in a pivot table has its *own* filter context. Figure 7-14 illustrates this with a pivot table containing the results of the [Total Revenue] calculated measure you created at the beginning of this chapter.

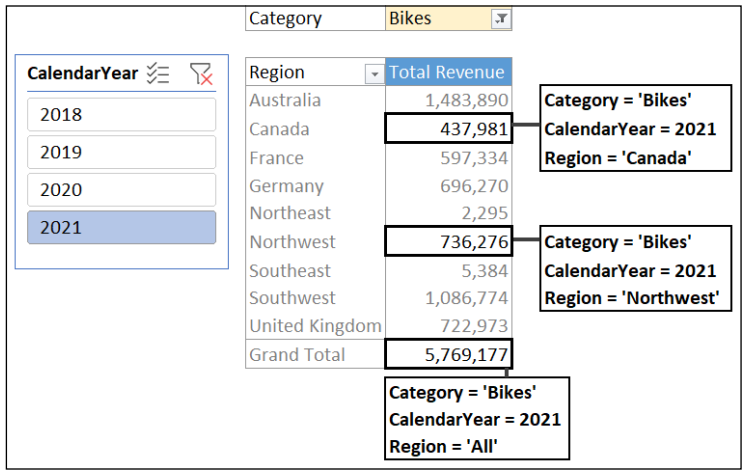


FIGURE 7-14: Each cell in a pivot table contains its own filter context.

TECHNICAL STUFF

You may hear the term *query context* thrown around DAX circles. Query context is an alternate term for filters applied via a pivot table as opposed to filters applied because of a DAX expression. Although Microsoft's documentation acknowledges the term *query context*, most DAX developers don't make a distinction between query context and filter context; they refer to all filters applied as the filter context.

Each individual cell in the pivot table represents a different run of the [Total Revenue] measure using that cell's unique filter context. For instance, because cell E5 has a different filter context than cell E9 does, the [Total Revenue] measure had to run once for cell E5 and once for cell E9.

When the [Total Revenue] measure was being calculated for cell E5, the Power Pivot filtered the internal data model on the dimensions defined in that cell's filter context. The relationships between each table automatically propagated the applied filtering from the one to the many side of the one-to-many relationship. Said another way, the filter context flowed in the direction of the arrows (see Figure 7-15). For instance, filtering the Products table on the [Category] column will automatically filter the Sales table because the arrow flows from Products to Sales. After all the necessary filtering is applied, the [Total Revenue] column ran the math on the data model in its filtered state and output the result to cell E5.

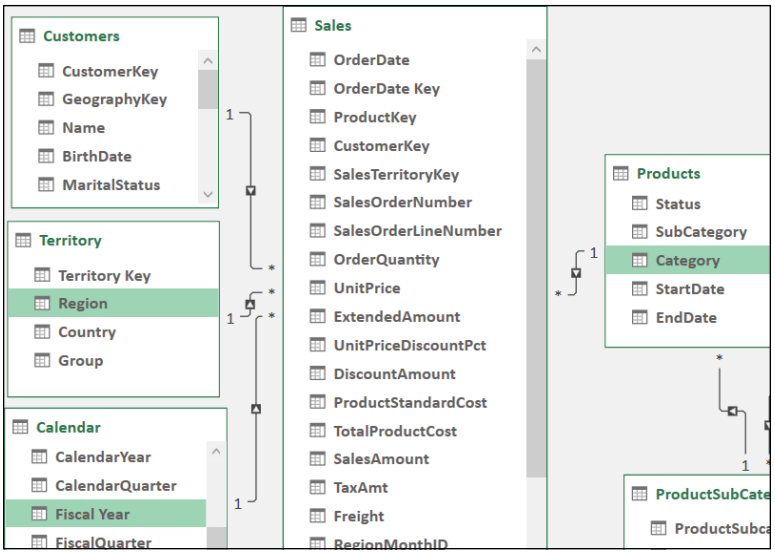


FIGURE 7-15: Filter context is propagated between tables via relationships in the direction of the arrows.

Getting context transitions with the CALCULATE function

The `CALCULATE` function is what is known as a *context transition* function. *Context transition,* in this capacity, essentially means overriding the current filter context and defining a new one. With the `CALCULATE` function, you can create DAX measures that supplement existing pivot data by including data in a completely different filter context.

The `CALCULATE` function requires, at a minimum, a measure expression that can be evaluated to a result. The following formula calculates the sum of Sales Amount but won't do much in the way of defining a new filter context until you add conditions.

```
=CALCULATE(SUM(Sales[SalesAmount]))
```

Adding a condition to the `CALCULATE` function adds utility to your measure and establishes a new filter context. In this formula, you added a condition asking for the sum of sales for the Bikes product category.

```
=CALCULATE(
  SUM(Sales[SalesAmount]),
  Products[Category] = "Bikes"
)
```

As always, you can use existing measures as the CALCULATE expression argument. Here, I'm using the [Total Revenue] measure created at the beginning of this chapter:

```
=CALCULATE(
  [Total Revenue],
  Products[Category] = "Bikes"
)
```

I used the CALCULATE formula to create a new measure called [Bike Sales] and added it to a pivot table with the [Total Revenue] measure. The new [Bike Sales] measure has its own product category filter context and won't respond to the category filter of the pivot table. Notice in Figure 7-16 that the figures for [Bikes Sales] don't change in the pivot table even when a new Product Category is selected in the slicer.

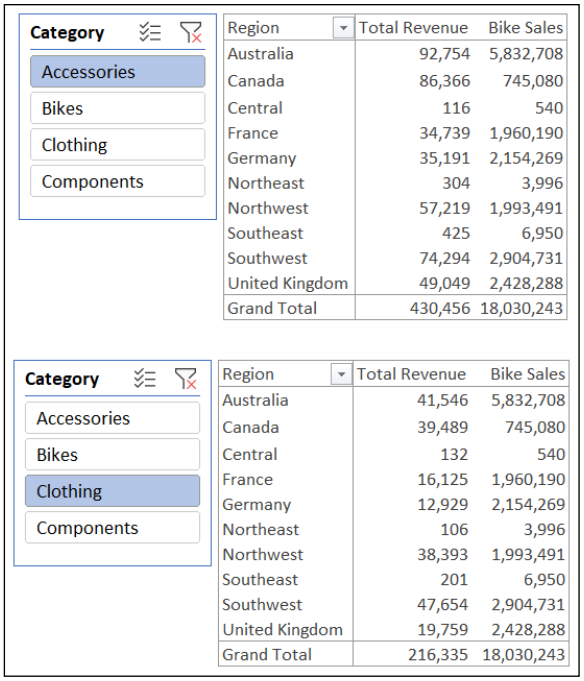


FIGURE 7-16: The [Bike Sales] measure has its own product category context, so it doesn't respond to the pivot slicer.

You can add as many conditions to the CALCULATE function as you'd like. The following formula adds to the filter context of [Bike Sales] by adding a condition for Fiscal Year:

```
=CALCULATE(
 SUM(Sales[SalesAmount]),
 Products[Category] = "Bikes",
 'Calendar'[Fiscal Year] = 2020
)
```

Adding flexibility with the FILTER function

The FILTER function returns a table of values that meet a specified condition. For example, the following expression returns a table of Products where the [DealerPrice] is less than the [StandardCost].

```
=FILTER(Products,
     DIVIDE(Products[DealerPrice],Products[StandardCost])<1
        )
```

The FILTER function works like an iterator function, iterating through the rows of the specified table (Products, in this case) and evaluating whether each row meets the specified condition. Rows that meet the condition are output in the table result.

The resulting table can be used in conjunction with other DAX functions to create a targeted filter context based on your defined conditions. In this case, you can use the previous FILTER expression with CALCULATE. The following formula does just that.

First, the FILTER function creates a table of Products where the [DealerPrice] is less than the [StandardCost]. The CALCULATE function then uses the resulting table to get the sum of sales for all the products in the FILTER table results.

```
=CALCULATE(SUM(Sales[SalesAmount]),
      FILTER(Products,
      DIVIDE(Products[DealerPrice],Products[StandardCost])<1
        )
      )
```

In Figure 7-17, you can see I used the `CALCULATE` and `FILTER` formula to create a new measure called LowProfitSales. Then I added it to a pivot table.

With some basic formula building, you've been able to exert a good bit of control over new measures and how they respond to the filter context of your reporting.

WHERE TO GO FROM HERE

DAX is a huge topic that goes well beyond the scope of this book. Hopefully, the basic concepts in this chapter will inspire you to pursue DAX a little further. Yes, DAX is a journey of time and practice, but the good news is that there are plenty of resources out there that can help you on your path. Here are some resources that you can leverage as you continue delving into DAX:

- **Chris Webb's BI Blog (`https://blog.crossjoin.co.uk/category/dax`):** Chris Webb's blog is focused on helping people make sense of their business data with Power Query and Power Pivot. With several years' worth of articles, Chris's blog is a rich source for DAX concepts.
- **Excelerator BI (`https://exceleratorbi.com.au`):** Matt Allington's website is a rich vein for articles on the latest Power BI trends. Check it out for a wide array of DAX-related tutorials.
- **P3 Adaptive (`https://p3adaptive.com`):** Rob Collie has been in the DAX world since its arrival in 2010. He has a knack for explaining the ins and outs of DAX from the perspective of Excel analysts. Rob's blog offers hundreds of excellent articles and tutorials.
- **RADACAD (`https://radacad.com`):** RADACAD is a group founded by Reza Rad. Reza and his team offer training to those who are looking to build their DAX muscles. RADACAD also publishes a free newsletter every month, keeping readers up to date with all the new changes in the Power Pivot realm.
- **SQLB: How to Learn DAX (`www.sqlbi.com/guides/dax`):** Alberto Ferrari and Marco Russo have helped countless Excel analysts make the leap into DAX. With a generous number of articles and examples, their website is a must for anyone learning DAX.

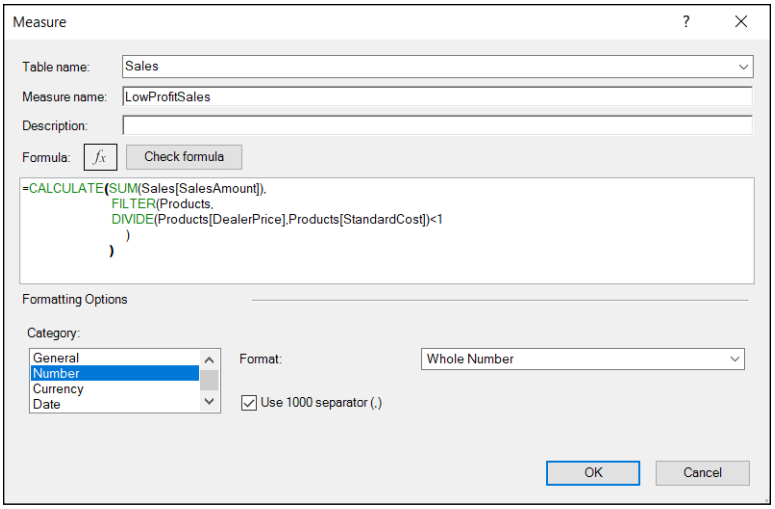


FIGURE 7-17: Using `CALCULATE` and `FILTER` together to establish a new filter context.

2

Wrangling Data with Power Query

IN THIS PART . . .

Discover the fundamentals of using Power Query to import and process data from various data sources.

Connect to external data sources using Power Query.

Explore Power Query tools and formulas that can automate and simplify your data-transformation processes.

Uncover methods of merging and appending multiple queries to go beyond simple data imports.

Get the skinny on creating your own custom functions to extend the functionality of Power Query.

IN THIS CHAPTER

- Spelling out the Power Query basics
- Understanding Query steps
- Managing existing queries
- Overviewing query actions

Chapter 8
Introducing Power Query

In information management, the term ETL (Extract, Transform, Load) refers to the three separate functions typically required to integrate disparate data sources: extract, transform, and load. The extraction function refers to the reading of data from a specified source and extracting a desired subset of data. The transformation function refers to the cleaning, shaping, and aggregating of data to convert it to the desired structure. The loading function refers to the actual importing or writing of the resulting data to a target location.

Excel analysts have been manually performing ETL processes for years — although they rarely call it ETL. Every day, millions of Excel users manually pull data from a source location, manipulate that data, and integrate it into their reporting. This process requires lots of manual effort.

Power Query enhances the ETL experience by offering an intuitive mechanism to extract data from a wide variety of sources, perform complex transformations on that data, and then load the data into a workbook or the Internal Data Model.

In this chapter, you explore the basics of the Power Query Add-in. You also get a glimpse of how it can help you save time and automate the steps needed to ensure that clean data is imported into your reporting models.

Power Query Basics

In this section, I walk you through a simple example of using Power Query. Imagine that you need to import Microsoft Corporation stock prices from the past 30 days by using Yahoo! Finance. For this scenario, you need to perform a web query to pull the data you need from Yahoo! Finance.

Starting the query

To start the query, follow these steps:

1. **In Excel, select the Get Data command in the Get & Transform Data group on the Data tab and then choose From Other Sources ➪ From Web (see Figure 8-1).**

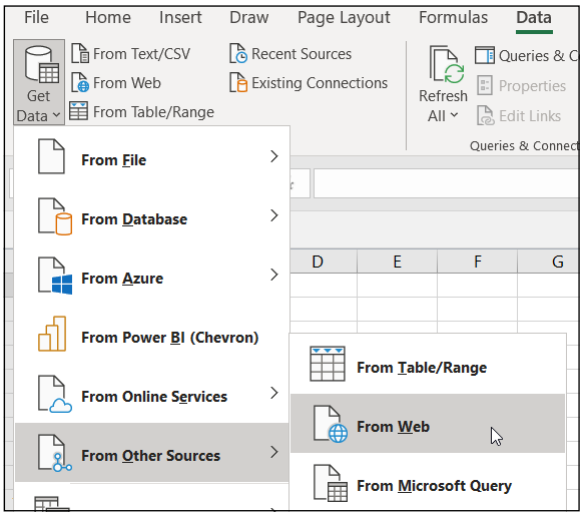


FIGURE 8-1: Starting a Power Query web query.

2. **In the dialog box that appears, enter the URL for the data you need, as shown in Figure 8-2.**

   In this example, you type `http://finance.yahoo.com/q/hp?s=MSFT`.

   After a bit of gyrating, the Navigator dialog box shown in Figure 8-3 appears. You can select the data source that you want to extract. Click on each table to see a preview of the data.

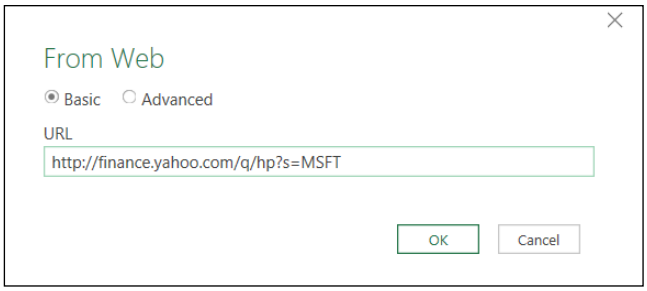


FIGURE 8-2: Enter the target URL containing the data you need.

3. In this case, Table 0 holds the historical stock data you need, so click Table 0 in the list box on the left and then click the Transform Data button.

You may have noticed that the Navigator dialog box, shown in Figure 8-3, offers a Load button (next to the Transform Data button). You can use this button to skip any editing and import your targeted data as is. If you're sure that you won't need to transform or shape your data in any way, click the Load button to import the data directly into the data model or a spreadsheet in your workbook.

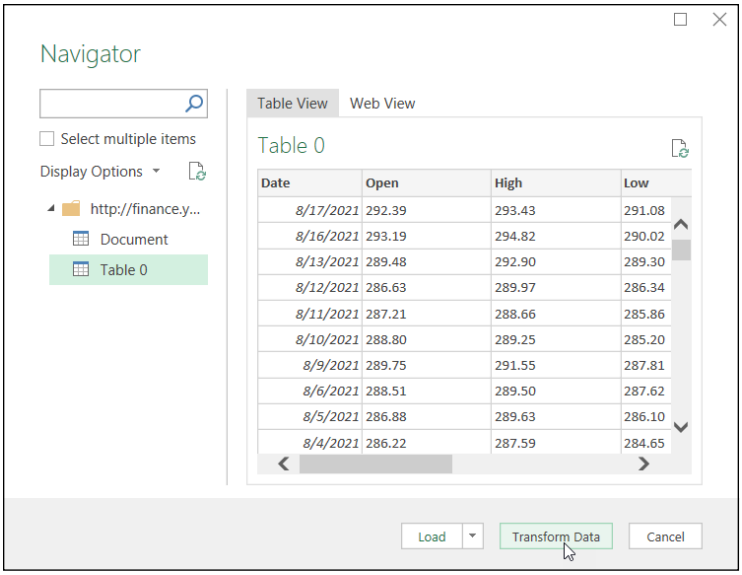


FIGURE 8-3: Select the correct data source and then click the Transform Data button.

WARNING

Excel has another From Web command button, on the Data tab in the Get External Data group. This unfortunate duplicate command is the legacy web-scraping capability found in all Excel versions since Excel 2000. The Power Query version of the From Web command (found under the Get Data drop-down) goes beyond simple web scraping. Power Query is able to pull

data from advanced web pages, and it can manipulate the data. Make sure you're using the correct feature when pulling data from the web.

When you click the Transform Data button, Power Query activates a new Query Editor window, which contains its own Ribbon and a preview pane that shows a preview of the data (see Figure 8-4). You can apply certain actions to shape, clean, and transform the data before importing.

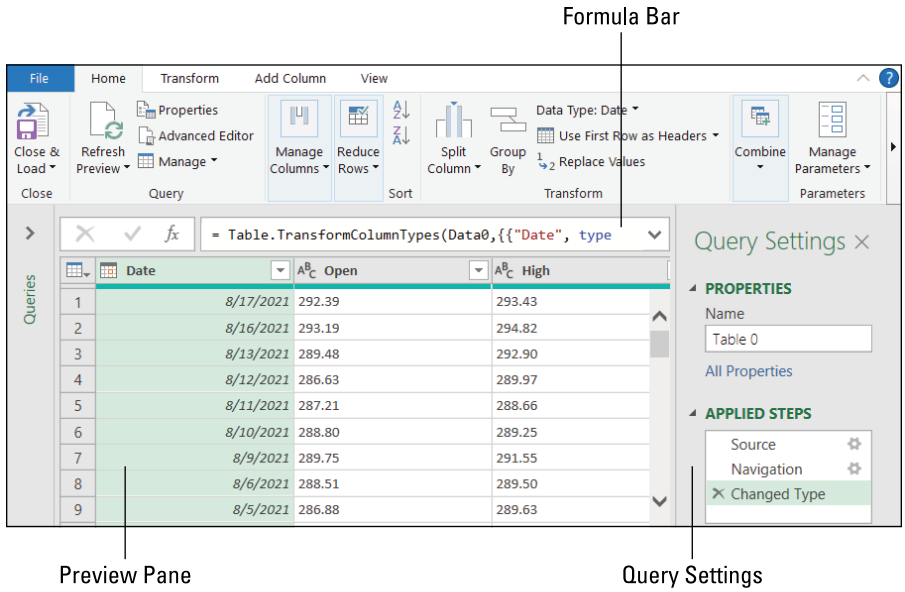


FIGURE 8-4: The Query Editor window allows you to shape, clean, and transform data.

The idea is to work with each column shown in the Query Editor, applying the necessary actions that will give you the data and structure you need. You can dive deeper into column actions later in this chapter. For now, continue toward the goal of getting the last 30 days of stock prices for Microsoft Corporation.

4. **Click the High field and then hold down the Ctrl key on your keyboard while you click the Low field and the Close fields.**

5. **Right-click and choose Change Type ➪ Currency, as shown in Figure 8-5.**

   This ensures that the Date field is formatted as a proper date. Power Query will ask if you want to replace the current step or add a new step.

6. **Choose Add a New Step.**

7. **Remove all unnecessary columns by right-clicking each one and selecting Remove.**

   (Besides the Date field, the only other columns you need are the High, Low, and Close fields.)

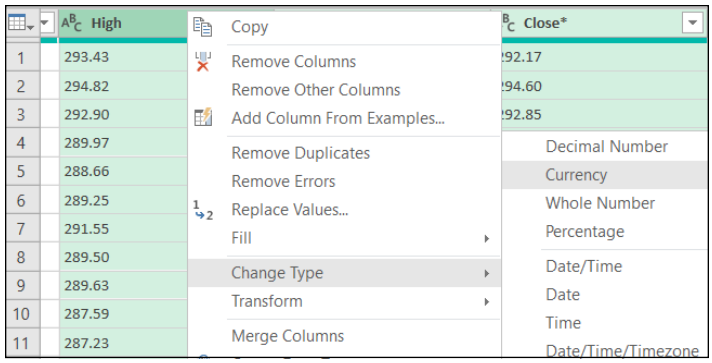


FIGURE 8-5: Change the data type of the High, Low, and Close fields to currency format.

Alternatively, you can hold down the Ctrl key on the keyboard, select the columns you want to keep, right-click any selected column, and then choose Remove Other Columns (see Figure 8-6).

You may notice that some of the rows show the word Error. These are rows that contained text values that could not be converted.

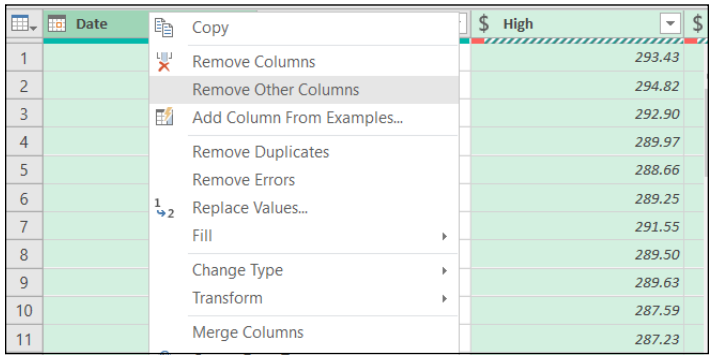


FIGURE 8-6: Select the columns you want to keep, and then select Remove Other Columns to get rid of them.

8. **Remove the Error rows by right-clicking the High field and selecting Remove Errors, as shown in Figure 8-7.**

9. **After all the errors are removed, right-click the Date field and select the Duplicate Column option.**

   A new column (named Date -Copy) is added to the preview.

10. **Right-click the newly added column, select the Rename option, and then rename the column** Week Of**.**

11. **Right-click the Week Of column you just created and choose Transform ➪ Week ➪ Start of Week, as shown in Figure 8-8.**

    Excel transforms the date to display the start of the week for a given date.

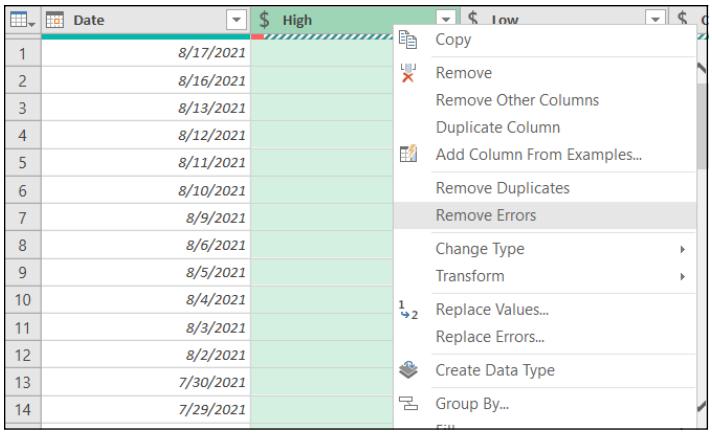


FIGURE 8-7: Removing errors caused by text values that could not be converted to currency.

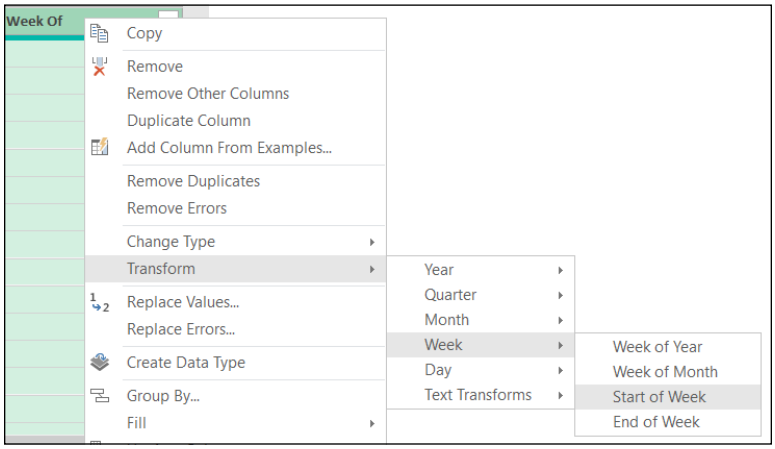


FIGURE 8-8: The Power Query Editor can be used to apply transformation actions such as displaying the start of the week for a given date.

12. When you've finished configuring your Power Query feed, click the Close & Load drop-down found on the Home tab of the Power Query Editor to reveal the two options shown in Figure 8-9:

- *Close & Load:* Saves your query and outputs the results to a new worksheet in your workbook as an Excel table. You can choose the Close & Load To option to activate the Import Data dialog box (see Figure 8-9). There, you can choose to output the results to a specific worksheet or to the internal data model.

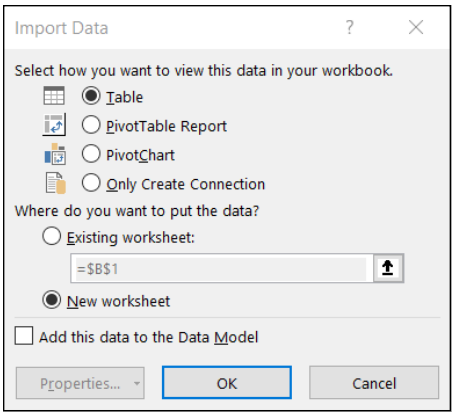


FIGURE 8-9: The Import Data dialog box gives you more control over how the results of queries are used.

- *Close & Load To:* Activates the Import Data dialog box (see Figure 8-9). There, you can choose to output the results to a specific worksheet. The Import Data dialog box also enables you to save the query as a query connection only, which means you'll be able to use the query in various processes without needing to output the results anywhere.

13. **Select the New Worksheet option button to output your results as a table on a new worksheet in the active workbook.**

At this point, you have a table similar to the one shown in Figure 8-10, which can be used to produce the pivot table you need.

	A	B	C	D	E
1	Date	High	Low	Close*	Week Of
2	8/17/2021	293.43	291.08	292.86	8/15/2021
3	8/16/2021	294.82	290.02	294.6	8/15/2021
4	8/13/2021	292.9	289.3	292.85	8/8/2021
5	8/12/2021	289.97	286.34	289.81	8/8/2021
6	8/11/2021	288.66	285.86	286.95	8/8/2021
7	8/10/2021	289.25	285.2	286.44	8/8/2021
8	8/9/2021	291.55	287.81	288.33	8/8/2021
9	8/6/2021	289.5	287.62	289.46	8/1/2021
10	8/5/2021	289.63	286.1	289.52	8/1/2021
11	8/4/2021	287.59	284.65	286.51	8/1/2021
12	8/3/2021	287.23	284	287.12	8/1/2021
13	8/2/2021	286.77	283.74	284.82	8/1/2021
14	7/30/2021	286.66	283.91	284.91	7/25/2021
15	7/29/2021	288.62	286.08	286.5	7/25/2021
16	7/28/2021	290.15	283.83	286.22	7/25/2021
17	7/27/2021	289.58	282.95	286.54	7/25/2021

FIGURE 8-10: Your final query pulled from the internet: transformed, put into an Excel table, and ready to use in a pivot table.

Take a moment to appreciate what Power Query allowed you to do just now. With a few clicks, you searched the internet, found some base data, shaped the data to keep only the columns you needed, and even manipulated that data to add an extra Week Of dimension to the base data. This is what Power Query is about: enabling you to easily extract, filter, and reshape data without the need for any programmatic coding skills.

TIP

You can get back to the Power Query Editor window for any query by activating the Queries & Connections task pane. On the Excel Ribbon, choose Data➪Queries & Connections. From here, you can simply right-click a query and select Edit.

Understanding query steps

Power Query uses its own formula language (known as the "M" language) to codify your queries. As with macro recording, each action you take when working with Power Query results in a line of code being written into a query step. Query steps are embedded M code that allow your actions to be repeated each time you refresh your Power Query data.

To explore this concept, open the Power Query Editor for the table you just created. Right-click anywhere in the table shown in Figure 8-10 and choose Table➪Edit Query. You can see the query steps for your queries in the Query Settings pane (see Figure 8-11).

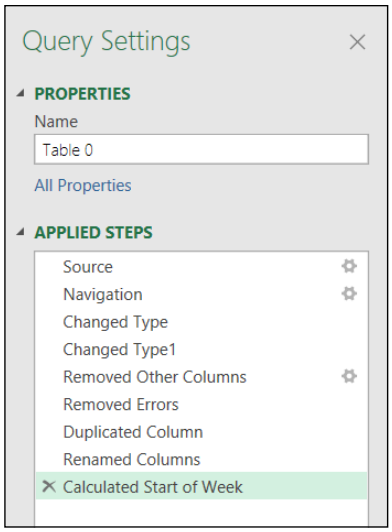


FIGURE 8-11: You can view and manage query steps in the Applied Steps section of the Query Settings pane.

Each query step represents an action you took to get to a data table. You can click on any step to see the underlying M code in the Power Query formula bar. For example, clicking the step called Removed Errors reveals the code for that step in the formula bar.

TIP

If you don't see the Query Settings pane, click the Query Settings command on the View tab of the Power Query Editor Ribbon. The View tab also contains the Formula Bar check box, allowing you to expose Formula bar that displays the M syntax for each given step.

When you click on a query step, the data shown in the preview pane shows you what the data looked like up to and including the step you clicked. For example, in Figure 8-11, clicking the step before the Removed Other Columns step lets you see what the data looked like before you removed the non-essential columns.

You can right-click on any step to see a menu of options for managing your query steps. Figure 8-12 illustrates the following options:

- **Edit Settings:** Edit the arguments or parameters that defines the selected step.
- **Rename:** Give the selected step a meaningful name.
- **Delete:** Remove the selected step. Be aware that removing a step can cause errors if subsequent steps depend on the deleted step.
- **Delete Until End:** Remove the selected step and all following steps.
- **Insert Step After:** Insert a step after the selected step.
- **Move Up:** Move the selected step up in the order of steps.
- **Move Down:** Move the selected step down in the order of steps.
- **Extract Previous:** Create a new query using the steps prior to the selected step. This feature is covered in Chapter 11.

VIEWING THE ADVANCED QUERY EDITOR

Power Query gives you the option to view and edit a query's embedded M code directly. While in the Power Query Editor, click the View tab on the Ribbon and select Advanced Editor. The Advanced Editor dialog box is little more than a space for you to type your own M code. Advanced users can use the M language to extend the capabilities of Power Query by directly coding their own steps in the Advanced Editor. I touch on the M language in Chapter 12 of this book.

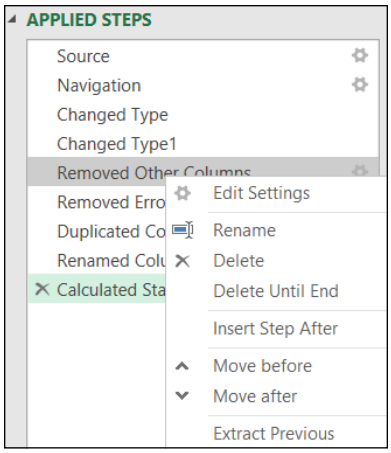


FIGURE 8-12: Right-click on any query step to edit, rename, delete, or move the step.

Refreshing Power Query data

Power Query data is in no way connected to the source data used to extract it. A Power Query data table is merely a snapshot. In other words, as the source data changes, Power Query doesn't automatically keep up with the changes; you need to intentionally refresh your query.

If you chose to load your Power Query results to an Excel table in the existing workbook, you can manually refresh by right-clicking on the table and selecting the Refresh option.

If you chose to load your Power Query data to the internal data model, you need to choose Data➪Queries & Connections and then right-click the target query and select the Refresh option.

To get a bit more automated with the refreshing of queries, you can configure your data sources to automatically refresh the Power Query data. To do so, follow these steps:

1. **Select the Data tab in the Excel Ribbon, and click the Queries & Connections command.**

   The Queries & Connections task pane appears.

2. **Right-click the Power Query data connection you want to refresh and then select the Properties option.**

   The Properties dialog box opens.

3. **Select the Usage tab.**

4. **Set the options to refresh the chosen data connection:**

   - *Refresh Every X Minutes:* Tells Excel to automatically refresh the chosen data every specified number of minutes. Excel refreshes all tables associated with that connection.
   - *Refresh Data When Opening the File:* Tells Excel to automatically refresh the chosen data connection after opening the workbook. Excel refreshes all tables associated with that connection as soon as the workbook is opened.

These refresh options are useful when you want to ensure that your customers are working with the latest data. Of course, setting these options does not preclude the ability to manually refresh the data using the Refresh command on the Home tab.

Managing existing queries

As you add various queries to a workbook, you need a way to manage them. Excel accommodates this need by offering the Queries & Connections task pane, which enables you to edit, duplicate, refresh, and generally manage all existing queries in the workbook.

Open the Queries & Connections task pane by selecting the Show Queries & Connections command on the Data tab of the Excel Ribbon. Find the query you want to work with, and right-click it to take any one of the actions described in the following list (see Figure 8-13):

- **Edit:** Open the Query Editor, where you can modify the query steps.
- **Delete:** Delete the selected query.
- **Refresh:** Refresh the data in the selected query.
- **Load To:** Activate the Import Data dialog box, where you can redefine where the selected query's results are used.
- **Duplicate:** Create a copy of the query.
- **Reference:** Create a new query that references the output of the original query.
- **Merge:** Merge the selected query with another query in the workbook by matching specified columns.
- **Append:** Append the results of another query in the workbook to the selected query.

- **Send to Data Catalog:** Publish and share the selected query via a Microsoft Power BI server that your IT department sets up and manages.
- **Export Connection File:** Save an Office Data Connection (.odc) file with the connection credentials for the query's source data.
- **Move to Group:** Move the selected query into a logical group that you create for better organization.
- **Move Up:** Move the selected query up in the Queries & Connections pane.
- **Move Down:** Move the selected query down in the Queries & Connections pane.
- **Show the Peek:** Show a preview of the query results for the selected query.
- **Properties:** Rename the query and add a friendly description.

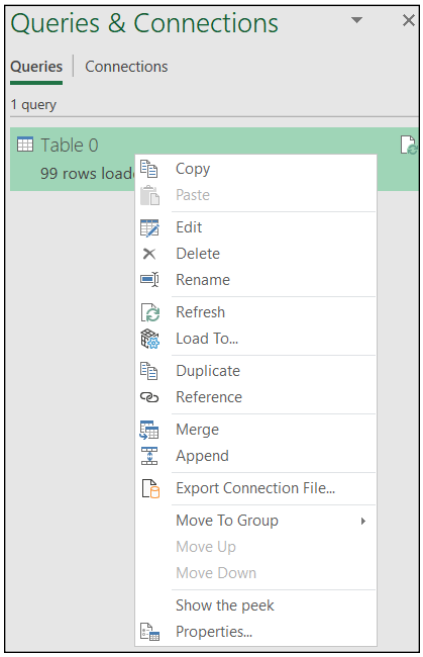


FIGURE 8-13: Right-click any query in the Queries & Connections pane to see the available management options.

The Queries & Connections pane is especially useful when your workbook contains several queries. Think of it as a kind of table of contents that allows you to easily find and interact with the queries in your workbook.

Understanding Column-Level Actions

Right-clicking a column in the Power Query Editor activates a shortcut menu that shows a full list of the actions you can take. You can also apply certain actions to multiple columns at one time by selecting two or more columns before right-clicking. Table 8-1 explains the commands you see when right-clicking a column within the Power Query Editor.

TABLE 8-1 Column-Level Actions

Action	Purpose	Available with Multiple Columns?
Remove	Remove the selected column from the Power Query data.	Yes
Remove Other Columns	Remove all non-selected columns from the Power Query data.	Yes
Duplicate Column	Create a duplicate of the selected column as a new column placed on the far right end of the table. The name given to the new column is Copy of X, where X is the name of the original column.	No
Add Column from Examples	Similar to Excel's Flash Fill feature, this command creates data in a new column from examples you provide. Power Pivot automatically fills in data when it senses a pattern.	Yes
Remove Duplicates	Remove all rows from the selected column where the values duplicate earlier values. The row with the first occurrence of a value isn't removed.	Yes
Remove Errors	Remove rows containing errors in the selected column.	Yes
Change Type	Change the data type of the selected column to any of these types: Binary, Date, Date/Time, Date/Time/Timezone, Duration, Number, Currency, Decimal Number, Whole Number, Percentage, Text, Time, or Using Locale (which localizes data types to the country you specify).	Yes
Transform	Change the way values in the column are rendered. You can choose from the following options: Lowercase, Uppercase, Capitalize Each Word, Left, Trim, Clean, and Length. If the values in the column are date/time values, the options are Date, Time, Day, Month, Year, or Day of Week. If the values in the column are number values, the options are Round, Absolute Value, Factorial, Base-10 Logarithm, Natural Logarithm, Power, and Square Root.	Yes

(continued)

TABLE 8-1 *(continued)*

Action	Purpose	Available with Multiple Columns?
Replace Values	Replace one value in the selected column with another specified value.	Yes
Replace Errors	Replace unsightly error values with your own, friendlier text.	Yes
Create Data Type	Stores multiple columns of data in one column as metadata, allowing you to expose all the data you need without taking up space in your worksheet. Excel formulas can interact with these rich data types to expose the stored data within.	Yes
Group By	Aggregate data by row values. For example, you can group by state and either count the number of cities in each state or sum the population of each state.	Yes
Fill	Fill empty cells in the column with the value of the first non-empty cell. You have the option to fill up or fill down.	Yes
Unpivot Other Columns	Transpose the unselected columns from column-oriented to row-oriented or vice versa.	Yes
Unpivot Selected Columns	Transpose the selected columns from column oriented to row oriented or vice versa.	Yes
Rename	Rename the selected column to a name you specify.	No
Move	Move the selected column to a different location in the table. You have these choices for moving the column: Left, Right, To Beginning, and To End.	Yes
Drill Down	Navigate to the contents of the column. This option is used with tables that contain metadata representing embedded information.	No
Add as New Query	Create a new query with the content of the column, by referencing the original query in the new one. The name of the new query is the same as the column header of the selected column.	No
Split Column (Ribbon only)	Split the value of a single column into two or more columns, based on a number of characters or a given delimiter, such as a comma, semicolon, or tab.	No
Merge Column (Ribbon only)	Merge the values of two or more columns into a single column that contains a specified delimiter, such as a comma, semicolon, or tab.	Yes

TIP

All column-level actions available in Power Query are also available on the Query Editor Ribbon, so you can either choose the convenience of right-clicking to quickly select an action or use the more visual Ribbon menu. A few useful column-level actions are found only on the Ribbon, as described in Table 8-1.

Understanding Table Actions

While you're in the Query Editor, Power Query lets you apply certain actions to an entire data table. You can see the available table-level actions by clicking the Table Actions icon, shown in Figure 8-14.

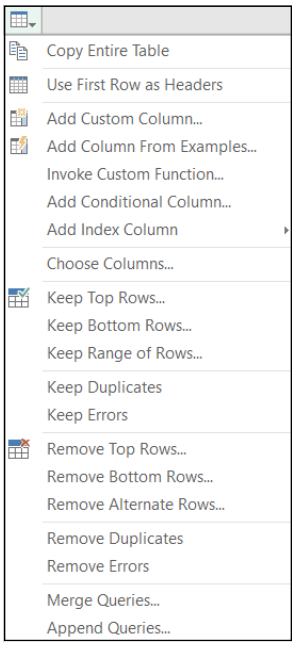


FIGURE 8-14: Click the Table Actions icon in the upper-left corner of the Query Editor Preview pane to see the table-level actions you can use to transform the data.

Table 8-2 lists the more commonly used table-level actions and describes the primary purpose of each one.

TIP

All table-level actions available in Power Query are also available on the Power Query Editor Ribbon, so you can either choose the convenience of right-clicking to quickly select an action or use the more visual Ribbon menu.

TABLE 8-2 Table-Level Actions

Action	Purpose
Use First Row as Headers	Replace each table header name with the values in the first row of each column.
Add Custom Column	Insert a new column after the last column of the table. The values in the new column are determined by the value or formula you define.
Add Column from Example	Similar to Excel's Flash Fill feature, this command creates data in a new column from examples you provide. Power Pivot automatically fills in data when it senses a pattern.
Add Conditional Column	Insert a new column that contains the results of a specified `IF...THEN...ELSE` statement.
Add Index Column	Insert a new column containing a sequential list of numbers starting from 1, 0, or another specified value you define.
Choose Columns	Choose the columns you want to keep in the query results.
Keep Top Rows	Remove all but the top N number of rows. You specify the number threshold.
Keep Bottom Rows	Remove all but the bottom N number of rows. You specify the number threshold.
Keep Range of Rows	Remove all rows except the ones that fall within a range you specify.
Keep Duplicates	Remove all but duplicated rows.
Keep Errors	Remove all but duplicated rows with error values.
Remove Top Rows	Remove the top N rows from the table.
Remove Bottom Rows	Remove the bottom N rows from the table.
Remove Alternate Rows	Remove alternate rows from the table, starting at the first row to remove and specifying the number of rows to remove and the number of rows to keep.
Remove Duplicates	Remove all rows where the values in the selected columns duplicate earlier values. The row with the first occurrence of a value set isn't removed.
Remove Errors	Remove rows containing errors in the selected columns.
Merge Queries	Create a new query that merges the current table with another query in the workbook by matching specified columns.
Append Queries	Create a new query that appends the results of another query in the workbook to the current table.

IN THIS CHAPTER

- Extracting data from files
- Getting data from external databases
- Importing from other nonstandard data systems
- Understanding the Data Profiling feature

Chapter 9
Power Query Connection Types

Microsoft has invested a great deal of time and resources in ensuring that Power Query has the ability to connect to a wide array of data sources. Whether you need to pull data from an external website, a text file, a database system, Facebook, or a web service, Power Query can accommodate most, if not all, of your source data needs.

You can see all available connection types by clicking on the Get Data drop-down arrow on the Data tab of the Excel Ribbon. Power Query offers the ability to pull from a wide array of data sources, as described in this list:

- **From File:** Pulls data from specified Excel files, text files, CSV files, XML files, or folders
- **From Database:** Pulls data from a database such as Microsoft Access, SQL Server, or SQL Server Analysis Services
- **From Azure:** Pulls data from Microsoft's Azure Cloud service
- **From PowerBI:** Pulls data from PowerBI data sets made available through your organization's PowerBI service.

- **From Online Services:** Pulls data from online software-as-a-service (SaaS) applications such as Salesforce.com, Microsoft Dynamics 365, and SharePoint lists
- **From Other Sources:** Pulls data from a wide array of internet, cloud, and other ODBC data sources

In this chapter, I help you explore the various connection types that can be leveraged to import external data.

Importing Data from Files

Organizational data is often stored in files such as text files, CSV files, and even other Excel workbooks. It's not uncommon to use these kinds of files as data sources for data analysis. Power Query offers several connection types that enable the importing of data from external files.

REMEMBER

The files you import don't necessarily have to be on your own PC. You can import files on network drives as well as in cloud repositories such as Google Drive and Microsoft OneDrive.

Getting data from Excel workbooks

You can import data from other Excel workbooks by selecting Data➪Get Data➪From File➪From Workbook from the Excel Ribbon.

Excel opens the Import Data dialog box where you can browse for the Excel file you want to work with. Note that you can import any kind of Excel file, including macro-enabled workbooks and template workbooks.

After you've selected a file, the Navigator pane activates (see Figure 9-1), showing you all the data sources available in the workbook.

The idea here is to select the data source you want and then either load or transform the data using the buttons at the bottom of the Navigator pane. Click the Load button to skip any editing and import your targeted data as is. Click the Transform Data button if you want to transform or shape the data before completing the import.

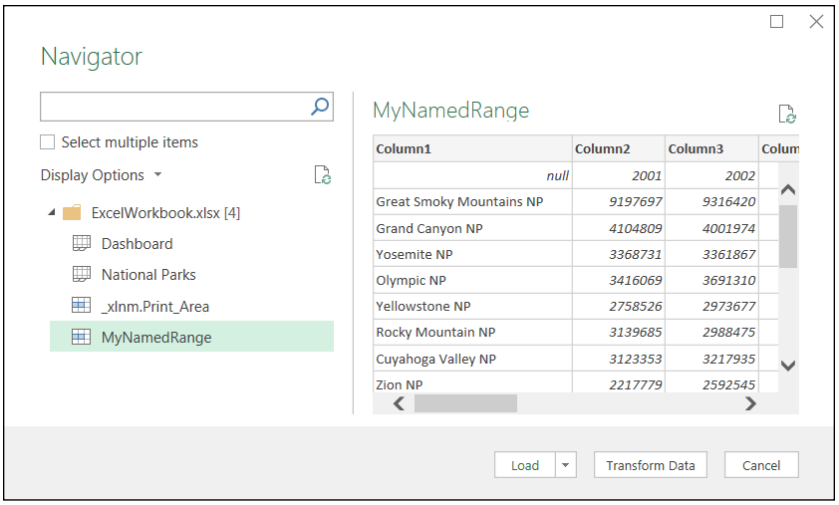


FIGURE 9-1: Select the data sources you want to work with, and then click the Load button.

In terms of Excel workbooks, a *data source* is either a worksheet or a defined named range. The icons next to each data source let you distinguish which sources are worksheets and which are named ranges. In Figure 9-1, the source named MyNamedRange is a defined named range, and the source named National Parks is a worksheet.

You can import multiple sources at a time by selecting the Select Multiple Items check box and then placing a check mark next to each worksheet and named range that you want imported.

REMEMBER

Power Query won't bring in charts, pivot tables, shapes, VBA code, or any other objects that may exist within a workbook. Power Query simply imports the data found in the used cell ranges of the workbook.

Getting data from CSV and text files

Text files are commonly used to store and distribute data because of their inherent ability to hold many thousands of bytes of data without having an inflated file size. Text files can do this by foregoing all the fancy formatting, leaving only the text.

A *comma-separated value (CSV)* file is a kind of text file that contains commas to delimit (separate) values into columns of data.

To import a text file, select Data➪Get Data➪From File➪From Text/CSV on the Excel Ribbon. Excel opens the Import Data dialog box, where you can browse for, and select, a text or CSV file.

Power Query opens the dialog box shown in Figure 9-2. Here, you can preview the contents and specify how the file should be imported. Note the drop-down options at the top of the dialog:

- **File Origin:** Define what encoding standards to use. This option is useful when handling data that comes from different regions of the world.
- **Delimiter:** Specify how the contents are *delimited* (separated). Some text files are tab delimited, meaning they contain tab characters that separate text values into columns of data. Other text files are comma delimited, while others still are delimited by another character such as a space or a colon. Use the Delimiter drop-down to tell Power Query which delimiter to look for when separating values into columns.
- **Data Type Detection:** When you import text files, Power Query will use the first 200 rows to guess the data types for each of the columns in the data. For instance, if the first 200 rows of a particular column are made up of numbers, Power Query will automatically change the data type of that column to numeric after importing the file. The Data Type Detection drop-down allows you to tell Power Query to analyze the entire file (as opposed to the first 200 rows) when guessing the data types. You also have the option of telling Power Query not to change any data types.

Click the Load button to import the data directly into your workbook. Click the Transform Data button to bring the data source into the Query Editor, where you can apply your edits and then click the Close & Load command to complete the import.

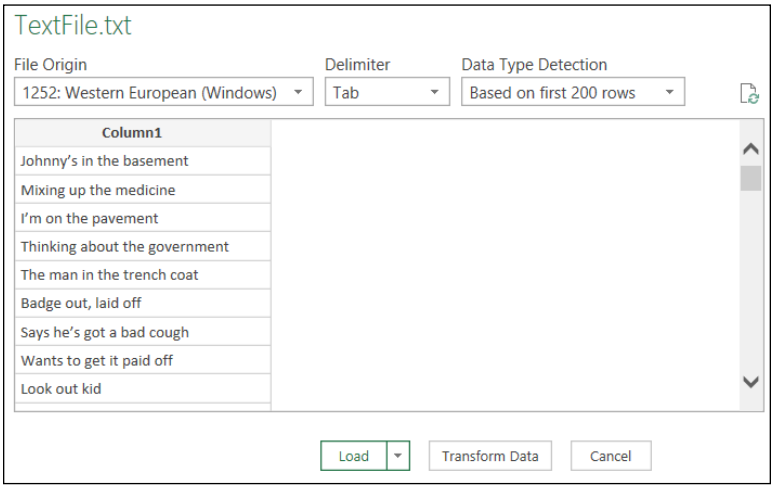


FIGURE 9-2: Preview the data and use the option drop-down menus to tell Power Query how to import the data.

Getting data from PDF files

Power Query now offers the ability to import data from PDFs. You can access PDF data by going to the Excel Ribbon and choosing Data➪ Get Data➪ From File➪ From PDF. The Import Data dialog box appears, allowing you to browse for your target PDF. After a few seconds, the Navigator dialog box, shown in Figure 9-3, opens, showing you the available tables and pages found in your chosen file.

Notice that both structured tables and pages are shown, allowing you the option of importing a specific table or an entire page from the PDF. Simply click the item you want to import and then click the Load button to import directly into your workbook or click the Transform Data button to clean the source data before importing.

You can even import multiple items from your PDFs by placing a selecting the Select Multiple Items check box (see Figure 9-3).

TIP

Rarely does the data from a PDF come in clean. You'll almost always need to click the Transform Data button in order to clean up column names, remove empty spaces, and generally remove unwanted data elements.

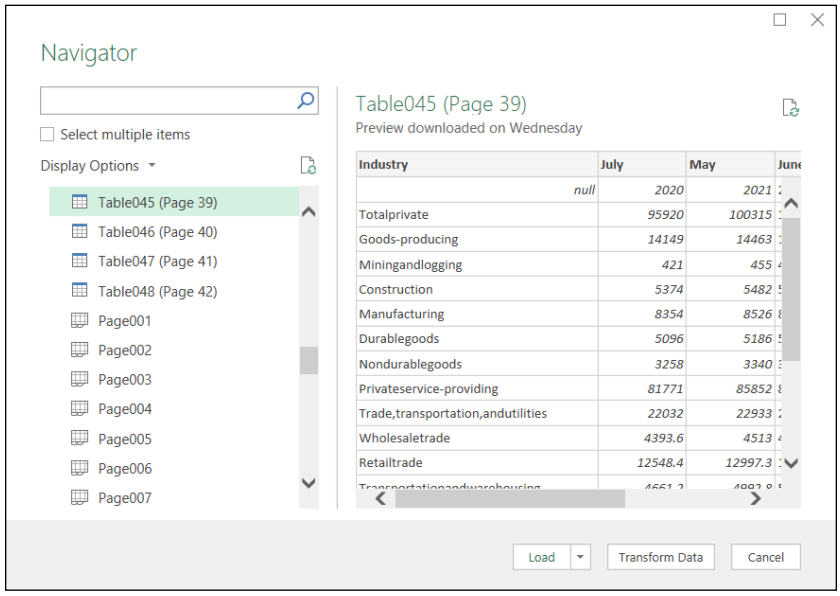
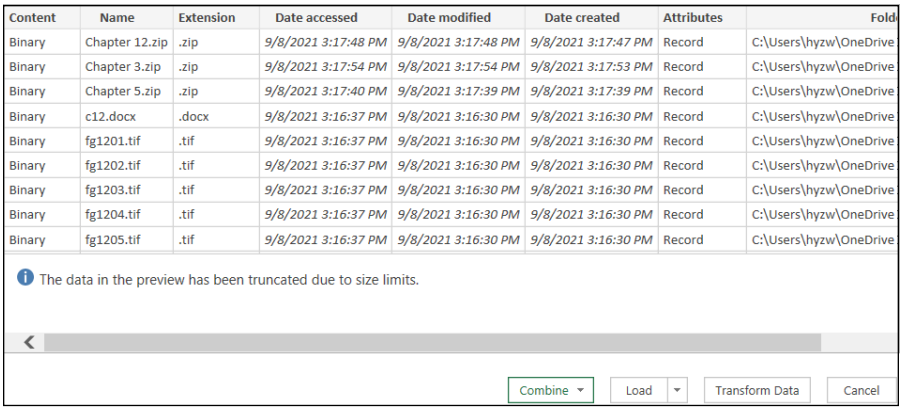


FIGURE 9-3: The available tables and pages in the PDF are shown in the Navigator dialog box.

Getting data from folders

Power Query has the ability to use the Windows file system as a data source, enabling you to import a list of folder contents for a specified directory. This comes in handy when you need to create a list of all the files in a particular folder.

From the Excel Ribbon, select Data➪Get Data➪From File➪From Folder. After you browse for the folder (directory) you want to use, the dialog box shown in Figure 9-4 opens.



Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Fold
Binary	Chapter 12.zip	.zip	9/8/2021 3:17:48 PM	9/8/2021 3:17:48 PM	9/8/2021 3:17:47 PM	Record	C:\Users\hyzw\OneDrive
Binary	Chapter 3.zip	.zip	9/8/2021 3:17:54 PM	9/8/2021 3:17:54 PM	9/8/2021 3:17:53 PM	Record	C:\Users\hyzw\OneDrive
Binary	Chapter 5.zip	.zip	9/8/2021 3:17:40 PM	9/8/2021 3:17:39 PM	9/8/2021 3:17:39 PM	Record	C:\Users\hyzw\OneDrive
Binary	c12.docx	.docx	9/8/2021 3:16:37 PM	9/8/2021 3:16:30 PM	9/8/2021 3:16:30 PM	Record	C:\Users\hyzw\OneDrive
Binary	fg1201.tif	.tif	9/8/2021 3:16:37 PM	9/8/2021 3:16:30 PM	9/8/2021 3:16:30 PM	Record	C:\Users\hyzw\OneDrive
Binary	fg1202.tif	.tif	9/8/2021 3:16:37 PM	9/8/2021 3:16:30 PM	9/8/2021 3:16:30 PM	Record	C:\Users\hyzw\OneDrive
Binary	fg1203.tif	.tif	9/8/2021 3:16:37 PM	9/8/2021 3:16:30 PM	9/8/2021 3:16:30 PM	Record	C:\Users\hyzw\OneDrive
Binary	fg1204.tif	.tif	9/8/2021 3:16:37 PM	9/8/2021 3:16:30 PM	9/8/2021 3:16:30 PM	Record	C:\Users\hyzw\OneDrive
Binary	fg1205.tif	.tif	9/8/2021 3:16:37 PM	9/8/2021 3:16:30 PM	9/8/2021 3:16:30 PM	Record	C:\Users\hyzw\OneDrive

FIGURE 9-4: Data preview of the files in the target folder.

The incoming data contains a row for each file contained inside the folder, including any files in subfolders. Click the Load button to import the data directly into your workbook. Click the Transform Data button to bring data source into the Query Editor.

In the Power Query Editor (see Figure 9-5), you'll see the imported table details the key attributes for each file, such as filename, file extension, date created, and date modified. You can even click the Expand icon in the Attributes field and choose to display some of the more advanced attributes for each file.

After you have all the attributes you need, you can click the Close & Load command on the Home tab to complete the import.

TIP

The files that are listed include all files contained in subfolders inside the folder you specified. Unfortunately, the resulting output is not hyperlinked back to the actual folder contents. In other words, you can't open the individual files from the query table.

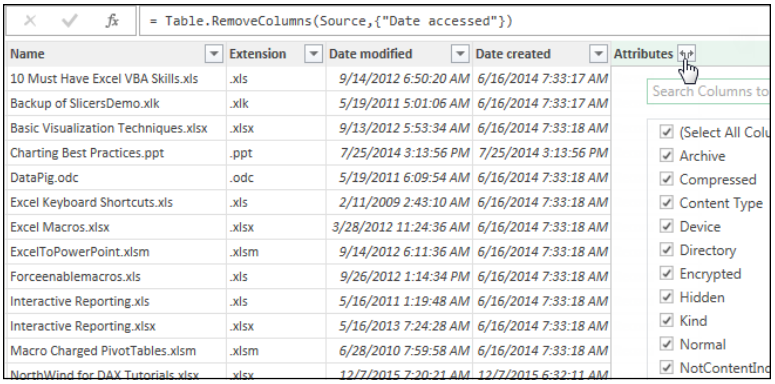


FIGURE 9-5: Use the Power Query Editor to add more file attributes to the import.

Importing Data from Database Systems

In smart organizations, the task of data management is not performed by Excel; rather, it's performed primarily by database systems such as Microsoft Access and SQL Server. Databases like these not only store millions of rows of data, but also ensure data integrity and allow for the rapid search and retrieval of data by way of queries and views.

A connection for every database type

Power Query offers options to connect to a wide array of database types. Microsoft has been keen to add connection types for as many commonly used databases as it can.

Relational and OLAP databases

Choose Data➪ Get Data➪ From Database and you see the list of databases shown in Figure 9-6. Power Query has the ability to connect to virtually any database commonly used today: SQL Server, Microsoft Access, Oracle, MySQL, and so forth.

Azure databases

If your organization has a Microsoft Azure cloud database or a subscription to Microsoft Azure Marketplace, an entire set of connection types is designed to import data from Azure databases (see Figure 9-7). You can get to these connection types by choosing Data➪ Get Data➪ From Azure.

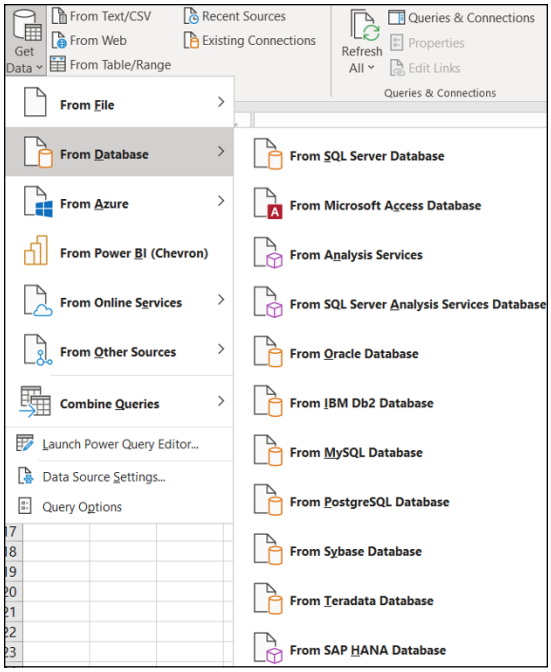


FIGURE 9-6: Power Query offers connection types for many of the popular database systems now in use.

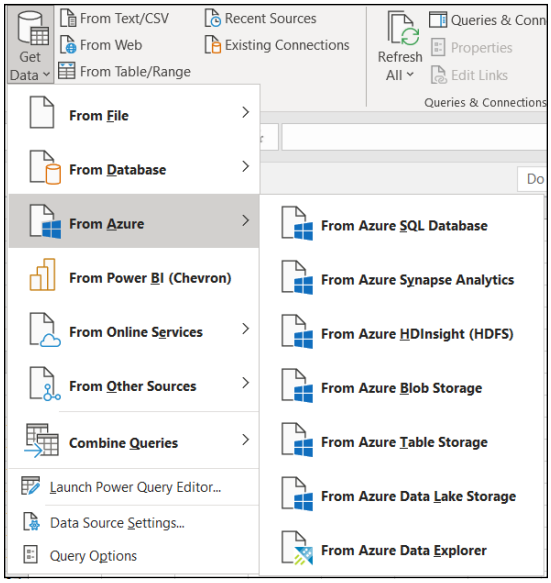


FIGURE 9-7: Tools for connection to Microsoft Azure cloud database services.

ODBC connections to nonstandard databases

If you're using a unique, nonstandard database system that isn't listed under From Database (refer to Figure 9-6) or From Azure (refer to Figure 9-7), not to worry: As long as your database system can be connected to via an ODBC connection string, Power Query can connect to it.

Choose Data➪ Get Data➪ From Other Data Sources to see a list of other connection types. Click the From ODBC option shown in Figure 9-8 to start a connection to your unique database via an ODBC connection string.

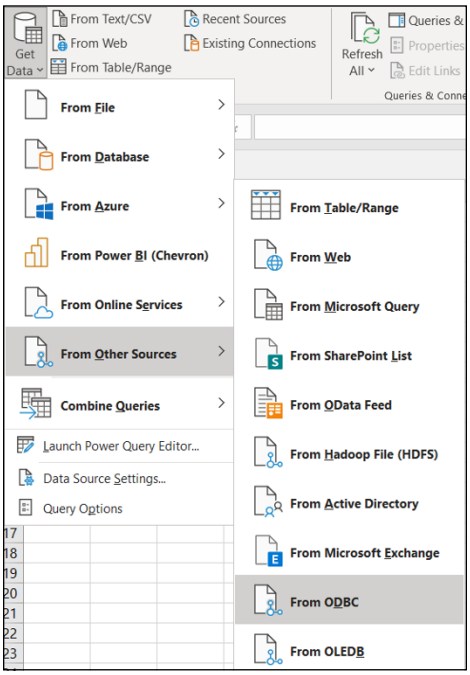


FIGURE 9-8: Starting an ODBC connection.

Getting data from other data systems

In addition to ODBC, Figure 9-8 illustrates other kinds of data systems that can be leveraged by Power Query.

Some of these data systems (SharePoint, Microsoft Exchange) are popular systems that are used in many organizations to store data and manage emails. Other systems, such as OData Feeds and Hadoop, are less-common services used to work with very large volumes of data. These are often mentioned in conversations about big data. And of course, the From Web option (demonstrated in Chapter 8) is an integral connection type for any analyst who leverages data from the internet.

Clicking any of these connections opens a set of dialog boxes customized for the selected connection. These dialog boxes ask for the basic parameters that Power Query needs in order to connect to the specified data source; parameters such as file path, URL, server name, and credentials.

Each connection type requires its own, unique set of parameters, so each of their dialog boxes is different. Luckily, Power Query rarely needs more than a handful of parameters to connect to any single data source, so the dialog boxes are relatively intuitive and hassle-free.

Walk-through: Getting data from a database

It would be redundant to walk through the process of connection to every type of database available. However, it would be useful to walk through the basic steps of connecting a database.

Here are the steps for connecting to one of the more ubiquitous database systems — Microsoft Access:

1. **Choose Data ➪ Get Data ➪ From Database ➪ From Microsoft Access Database.**
2. **Browse for your target database. You can use the Facility Services.accdb database, found in the sample files for this book.**

   After Power Query connects to the database, the Navigator pane, shown in Figure 9-9, activates. There, you see all database objects available to you, including tables and views (or *queries,* in Access lingo).
3. **Click the Sales_By_Employee view.**

   The Navigator pane displays a preview of the Sales_By_Employee data. If you want to transform or shape this data, click the Transform Data button. In this case, the data looks fine as is.
4. **Click the Load button to complete the import.**

   After a bit of processing, Power Query loads the data to a new Excel worksheet and adds the new query to the Workbook Queries pane, as shown in Figure 9-10.

REMEMBER

You can select multiple tables and views by selecting the Select Multiple Items check box and then placing a check mark next to each database object you want imported.

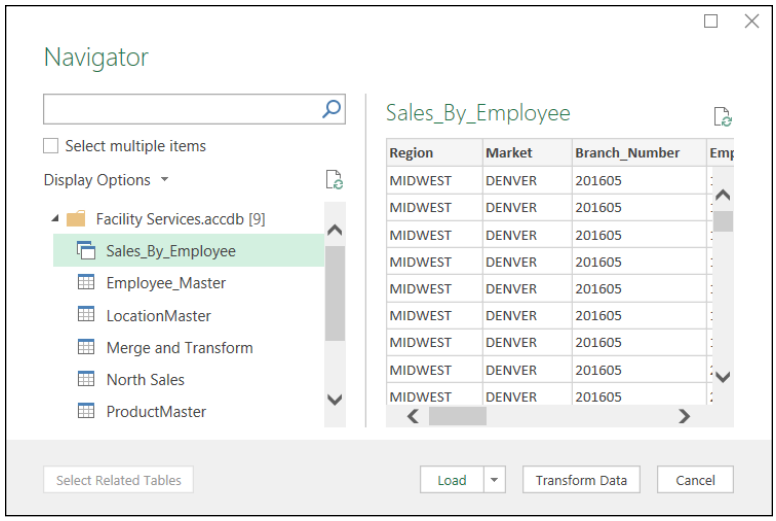


FIGURE 9-9: Select the view you want imported, and then click the Load button.

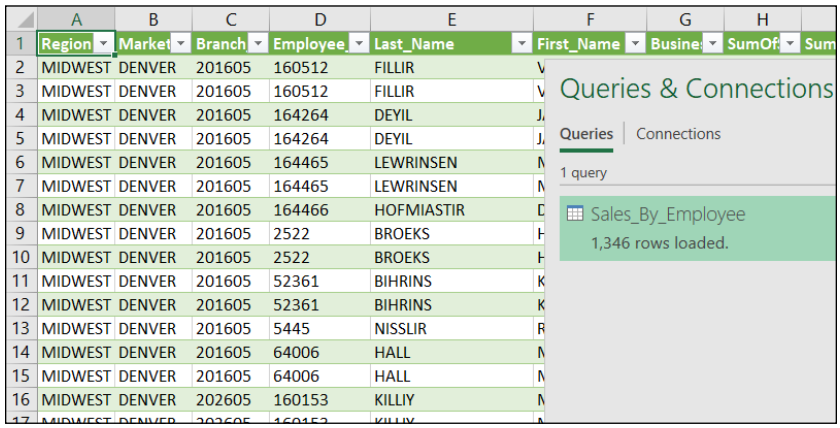


FIGURE 9-10: The final imported database data.

REMEMBER

The icon next to each database object distinguishes whether that object is a table or a view. Views have an icon that looks like two overlapping grids. See the icon for the Sales_By_Employee view, shown in Figure 9-9, to get the idea.

It's a best practice to use views whenever possible. Views are often cleaner data sets because they're already optimized to include only the columns and data that are necessary. (This improves query performance and helps minimize the workbook's file size.) In addition, you don't need to have an intimate knowledge of the database architecture. Someone with that knowledge has already done the work for you — joined the correct tables, applied the appropriate business rules, and optimized output, for example.

Managing Data Source Settings

Every time you connect to any web-based data source or data source that requires some level of credentials, Power Query *caches* (stores) the settings for that data source.

Suppose that you connect to a SQL Server database, enter all your credentials, and import the data you need. At the moment of successful connection, Power Query caches information about that connection in a file located on your local PC. It includes the connection string, username, password, and privacy settings, for example.

The purpose of all this caching is so that you don't have to reenter credentials every time you need to refresh your queries. That's nifty, but what happens when your credentials are changed? Well, the short answer is those queries will fail until the data source settings are updated.

You can edit data source settings by activating the Data Source Settings dialog box. To do so, choose Data➪ Get Data➪ Data Source Settings.

The Data Source Settings dialog box, shown in Figure 9-11, contains a list of all credentials-based data sources previously used in queries. Select the data source you need to change, and then click the Edit Permissions button.

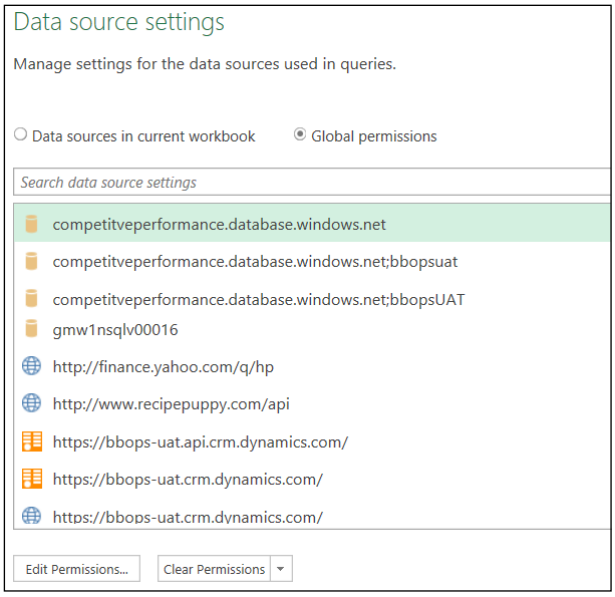


FIGURE 9-11: Edit a data source by selecting it and clicking the Edit Permissions button.

Another dialog box opens — this time, specific to the data source you selected (see Figure 9-12). This dialog box enables you to edit credentials as well as other data privacy settings.

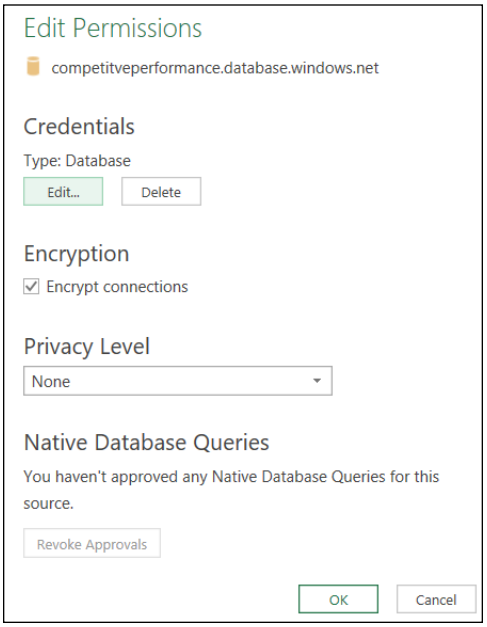


FIGURE 9-12: Edit a data source by selecting it and clicking the Transform Data button.

Click the Edit button to make changes to the credentials for the data source. The credentials editing screen will differ based on the data source you're working with, but again, the input dialog boxes are relatively intuitive and easy to update.

Power Query caches data source settings in a file located on your local PC. Even though you may have deleted a particular query, the data source setting is retained for possible future use. This can lead to a cluttered list of old and current data sources. You can clean out old items by selecting the data source in the Data Source Settings dialog box and clicking the Clear Permissions button.

Data Profiling with Power Query

When importing a new data source, it's often useful to understand the intricacies and pitfalls of the data before you start working with it. For instance, how many records are empty? How many unique values are there in a given column? What are the minimum and maximum values? Power Query's data profiling capabilities allow you to know your data and identify potential issues before using it.

In this section, I fill you in on some of the ways you can leverage data profiling in Power Query to get a better understanding of your data and address problem areas before they become a problem later in your reporting processes.

Data Profiling options

While in the Power Query Editor window, click the View tab to see options for data profiling in the Data View Group (see Figure 9-13).

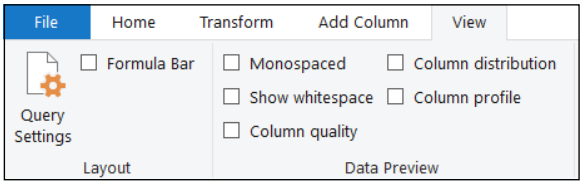


FIGURE 9-13: Data Profiling options are found in the Data View group under the View tab.

Take a moment to review the purpose of each option.

- **Monospaced:** Converts the font in the Data Preview window to monospaced, making it easier to see differences in data.
- **Show Whitespace:** Useful for calling out inline carriage returns and other invisible space characters.
- **Column Quality:** Displays the percentage of column values that are empty, the percentage that are rendered as errors, and the percentage that are considered valid values. This option is the most powerful in terms of providing an at-a-glance view of your data.
- **Column Distribution:** Provides a histogram visual displaying how many distinct and unique records are found in the values in each of the columns.
- **Column Profile:** Provides a useful way to see detailed descriptive statistics on a chosen column, such as the number of records with a 0 value, the minimum value in the column, the maximum value, the average value, and the standard deviation of all values in the column.

Be aware that the data profiler in Power Query, by default, only profiles the first 1,000 records. You can tell the profile to use the entire data set to get a more complete picture of your data. Figure 9-14 illustrates how to change the scope of data profiling to the entire data set.

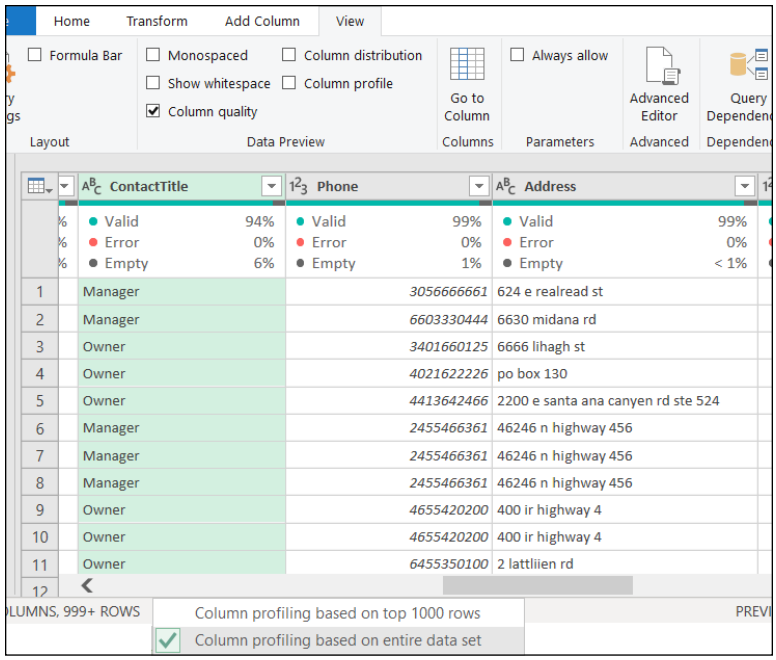


FIGURE 9-14: Choose Column Profiling Based on Entire Data Set to get a more complete picture of your data.

Data Profiling quick actions

When you select the Column Quality action, you'll see a set of figures that represent the percentage of values in a column that are valid, contain empty records, and are rendered as an error. Hovering over these percentages exposes a pop-up containing an ellipsis (see Figure 9-15). Clicking the ellipsis activates a shortcut menu allowing you to apply quick actions such as Remove Errors, Remove Empty, and Remove Duplicates.

When selecting the Column profile action, you'll two new panes below the data preview window: Column Statistics and Value Distribution. As you can see in Figure 9-16, the Value Distribution pane contains a histogram visual displaying the distribution of values. Right-clicking any of the bars reveals a quick action menu allowing you to apply transformations based on the type of data in that column. In this case, right-clicking the bar for zero reveals the options for Numbers Filter and Replace Values.

TIP

The quick actions exposed via the data profiler are simply an easy way to find and apply needed transformations. They aren't any different from those found in the Power Query Editor Ribbon and those exposed by simply right-clicking a value in the data preview window.

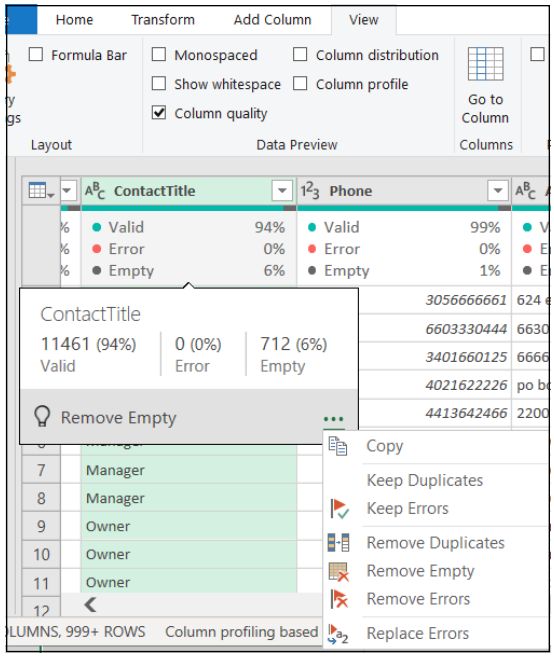


FIGURE 9-15: Exposing the quick actions for a column using the data column quality ellipsis.

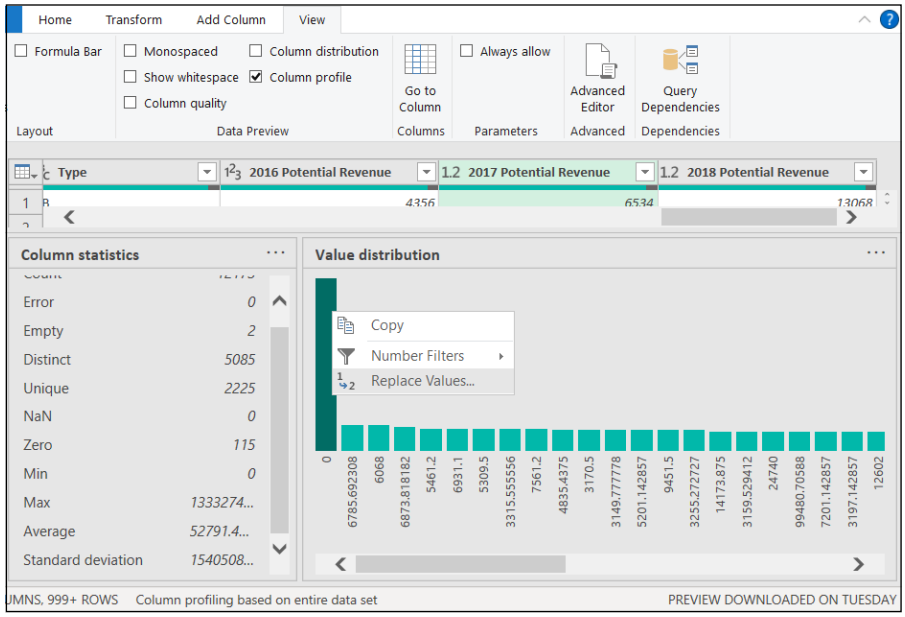


FIGURE 9-16: Right-clicking a column profile histogram bar exposes the quick actions for the associated value.

IN THIS CHAPTER

- Performing common transformations
- Creating your own custom columns
- Understanding Power Query formulas
- Applying conditional logic
- Grouping and aggregating data
- Working with custom data types

Chapter 10

Transforming Your Way to Better Data

Wouldn't it be great if all the data sources you work with were clean and ready to use? Unfortunately, that's not the case — you often receive data that is unpolished, or raw. That is to say, the data may have duplicates or blank fields or inconsistent text, for example.

Data transformation generally entails certain actions that are meant to "clean" your data — actions such as establishing a table structure, removing duplicates, cleaning text, removing blanks, and even adding your own calculations.

In this chapter, I introduce you to some of the tools and techniques in Power Query that make it easy for you to clean and massage your data.

ON THE WEB

You can follow along with the examples in this chapter by downloading the LeadList.txt sample file from `www.dummies.com/go/excelpowerpivotpowerqueryfd2e`. After you download it, you can import the sample file into Power Query by choosing Data➪ Get & Transform Data➪ From Text/CSV and then pointing to `LeadList.txt`.

Completing Common Transformation Tasks

Many of the unpolished data sets that come to you will require various types of transformation actions. This section covers some of the more common transformation tasks you will have to perform, such as removing duplicates, finding and replacing text, filling empty cells, and splitting or joining text values.

Removing duplicate records

Duplicate records are absolute analysis killers. The effect that duplicate records have on your analysis can be far-reaching, corrupting almost every metric, summary, and analytical assessment you produce. It is for this reason that finding and removing duplicate records should be your first priority when you receive a new data set.

Before you begin examining the data set to find and remove duplicate records, consider how you define a duplicate record. Look at the table shown in Figure 10-1, where you see 11 records. Of the 11 records, how many are duplicates?

SicCode	PostalCode	CompanyNumber	DollarPotential	City	State	Address
1389	77032	11147805	$9,517.00	houston	tx	6000 n sem heirten pkwy e
1389	77032	11147848	$9,517.00	houston	tx	43410 e herdy rd
1389	77042	11160116	$7,653.00	houston	tx	40642 rachmend ave ste 600
1389	77051	11165400	$9,517.00	houston	tx	5646 helmis rd
1389	77057	11173241	$9,517.00	houston	tx	2514 san filape st ste 6600
1389	77060	11178227	$7,653.00	houston	tx	100 n sem heirten pkwy e ste 100
1389	77073	11190514	$9,517.00	houston	tx	4660 rankan rd # 400
1389	77049	11218412	$7,653.00	houston	tx	4541 mallir read 6
1389	77040	13398882	$18,379.00	houston	tx	3643 wandfirn rd
1389	77040	13399102	$18,379.00	houston	tx	3643 wandfirn rd
1389	77077	13535097	$7,653.00	houston	tx	44160 wisthiamir rd ste 100

FIGURE 10-1: Does this table have duplicate records? It depends on how you define them.

If you were to define a duplicate record in Figure 10-1 as a duplication of only the SicCode, you would find 10 duplicate records. That is, of the 11 records shown, 1 record has a unique SicCode, and the other 10 are duplications. Now, if you were to expand your definition of a duplicate record to a duplication of both SicCode and PostalCode, you would find only two duplicates: the duplication of postal codes 77032 and 77040. Finally, if you were to define a duplicate record as a duplication of the unique value of SicCode, PostalCode, and CompanyNumber, you would find no duplicates.

This example shows that having two records with the same value in a column doesn't necessarily mean that you have a duplicate record. It's up to you to

determine which field or combination of fields best defines a unique record in the data set.

After you have a clear idea of which field or fields best make up a unique record in the table, you can remove duplicates easily by using the Remove Duplicates command.

Figure 10-2 illustrates the removal of duplicate rows based on three columns. Note the importance of selecting the columns that define a duplicate. In this case, the combination of Address, CompanyNumber, and CompanyName defines a duplicate record. You select these columns before clicking the Remove Duplicates command on the Home tab of the Power Query ribbon.

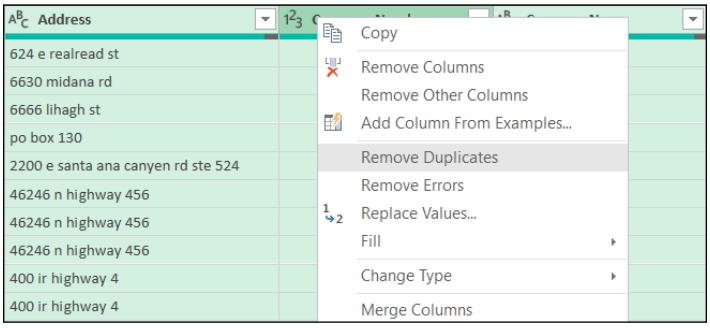


FIGURE 10-2: Removing duplicate records.

WARNING

The Remove Duplicates command essentially looks for distinct values in the columns you selected and then removes all records necessary to end up with a unique list of values. If you select only one column before giving the Remove Duplicates command, Power Query uses only that one column you selected to determine the unique list of values, which undoubtedly removes too many records — records that aren't truly duplicates. For this reason, be sure to select all columns that define a duplicate.

If you make a mistake and remove duplicates based on the wrong set of columns, don't worry: You can always use the Query Settings pane to delete that step. Right-click on the Removed Duplicates step and select Delete (see Figure 10-3). Alternatively, you can click the X next to the Remove Duplicates step.

TIP

If you don't see the Query Settings pane in the Power Query Editor window, choose View➪ Query Settings to activate the Query Settings pane.

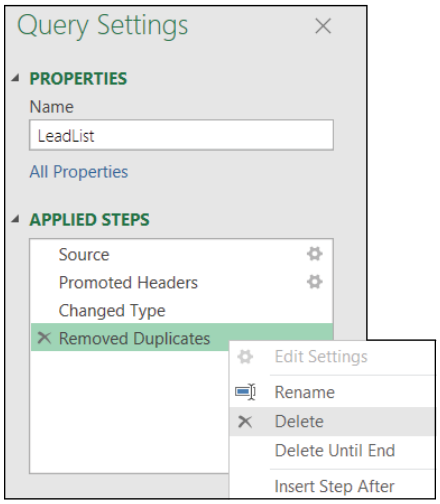


FIGURE 10-3: Undo the removal of records by deleting the Removed Duplicates step.

Filling in blank fields

There are two kinds of blank values: null and empty string. A *null* is essentially a numerical value of nothing, whereas an empty string is equivalent to entering two quotation marks ("") in a cell.

Blank fields aren't necessarily a bad thing, but having an excessive number of blanks in your data can lead to unexpected problems when analyzing it.

Your job is to decide whether to leave the blanks in the data set or fill them with actual values. Consider the following best practices:

- **Use blanks sparingly.** Working with a data set is a much less daunting task when you don't have to test continually for blank values.
- **Use alternatives whenever possible.** Represent missing values with some logical missing-value code whenever possible.
- **Never use null values in number fields.** Use zero instead of null in a currency or a number field that will be used in calculations.

Replacing null values

Power Query shows the word *null* for any null value in your data. Replacing the null values is as simple as selecting the column or columns you want to fix and then selecting the Replace Values command.

The Replace Values dialog box shown in Figure 10-4 appears. The key here is to enter the word **null** as the Value to Find value. You can then enter the value that you want to use instead. In this case, you can enter **0** as the Replace With value.

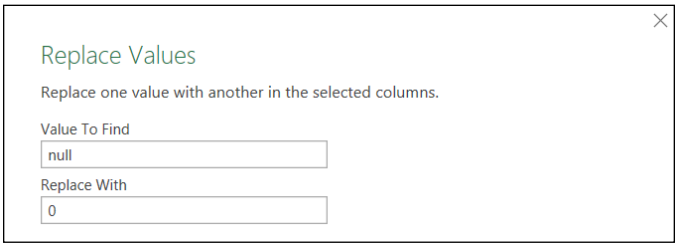


FIGURE 10-4: Replacing null with 0.

Filling in empty strings

To follow best practices, represent missing values in a field with some logical value code whenever possible. For example, in Figure 10-5, I want to tag with the word *Undefined* any record with a missing title in the ContactTitle field.

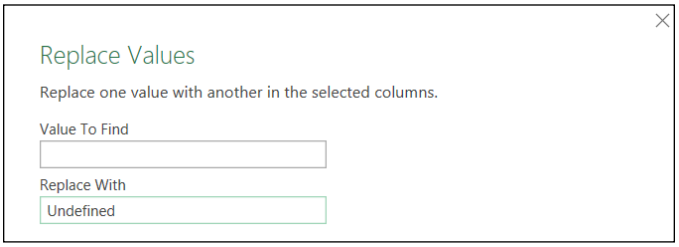


FIGURE 10-5: Replacing empty strings with the word *Undefined*.

You can do so by clicking on ContactTitle, selecting the Replace Values command, and then entering the word **Undefined** in the Replace Values dialog box. As you can see in Figure 10-5, because you're replacing an empty string, there's no need to enter anything in the Value to Find input box.

TIP

If you need to adjust or correct the step where you replace values, you can reopen the Replace Values dialog box by clicking the Gear icon next to the name for that step. This is true for any action that requires a dialog box to complete. Clicking on the Gear icon next to any step name opens the appropriate dialog box for that step.

Concatenating columns

You can easily *concatenate* (join) the values in two or more columns. In Power Query, you do this by using the Merge Columns command. The Merge Columns

command concatenates the values in two or more fields and outputs the newly merged values into a new column.

First choose the columns you want to concatenate, and then select the Transform tab and then the Merge Columns command, as shown in Figure 10-6.

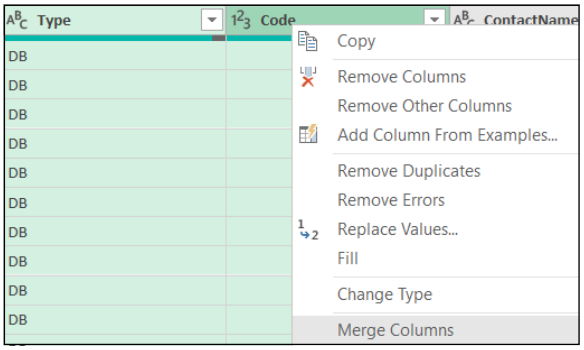


FIGURE 10-6: Merging the Type and Code fields.

The Merge Columns dialog box opens, as shown in Figure 10-7. You have the option of choosing from a list of the most commonly used delimiters (comma, space, tab, etc.). You can also select the Custom option to enter your own delimiter. In Figure 10-7, a hyphen (-) is used.

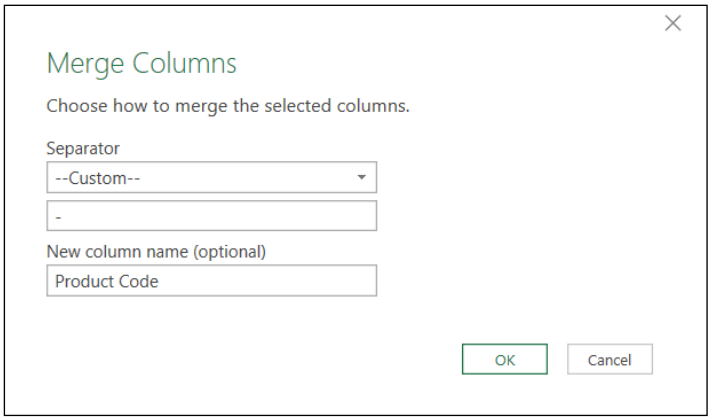


FIGURE 10-7: The Merge Columns dialog box.

As you can see, you can also name the new column that will be created.

The reward for your efforts is a new field containing the concatenated values from the original columns (see Figure 10-8). The resulting column will be named Merged. You can rename the column by right-clicking it and selecting the Rename option.

Product Code	ContactName
DB-100199	DAIMIRT, TAM, G.
DB-100199	THEMPSENJR, MAKE, G.
DB-100199	SCETT, ANDY, T.
DB-100199	MCKINZAE, DAVE, G.
DB-100199	NILSEN, REBIRT, T.
DB-100199	KILLIRMAN, DAVAD, G.
DB-200	CELIMAN, TERRANCE, G.
DB-	SANSENE, TERRANCE, G.
DB-100199	GIRVES, STIPHIN, G.

FIGURE 10-8: The original columns are removed and replaced with a new, merged column.

This feature is nifty, but notice that Power Query removes the original Type and Code columns. In some instances, you'll definitely want to concatenate values but retain the source columns. In those instances, it's useful to first copy the column and perform the extraction on the duplicate column. You can create a copy of a column by right-clicking the column and selecting Duplicate Column. When the duplicate column is created, it will be the last column (at the far right) of the table.

Changing case

Making sure that the text in your data has the correct capitalization may sound trivial, but it's important. Imagine that you receive a customer table that has an address field where all addresses are lowercase. How will that look on labels, form letters, or invoices? Fortunately, Power Query has a few built-in functions that make changing the case of your text a snap.

For example, the ContactName field (see Figure 10-9) contains names that are formatted in all uppercase letters. To change these names to the more appropriate proper case, you can use the Format command found on the Transform tab. The Format command has options for lowercase, uppercase, and proper case (capitalize each word).

REMEMBER

Selecting the Capitalize Each Word option reformats all values in the selected column to proper case.

Finding and replacing specific text

Imagine that you work in a company named BLVD, Inc. One day, the president of your company informs you that the abbreviation *blvd* on all addresses is now deemed an infringement of your company's trademarked name and must be changed to *Boulevard* as soon as possible. How would you go about meeting this new requirement?

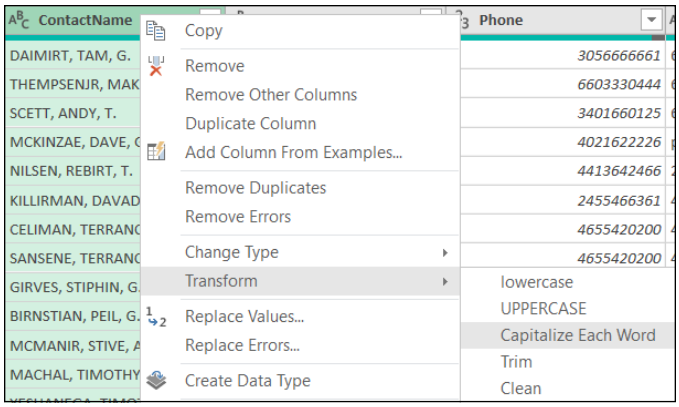


FIGURE 10-9: Reformatting the ContactName field to proper case.

The Replace Values function is ideal in a situation like this. Right-click the Address field, and then click the Replace Values command.

In the Replace Values dialog box (shown in Figure 10-10), simply fill the Value to Find input box with the value you want to find, and then fill the Replace With input box with the value you want to use as a replacement.

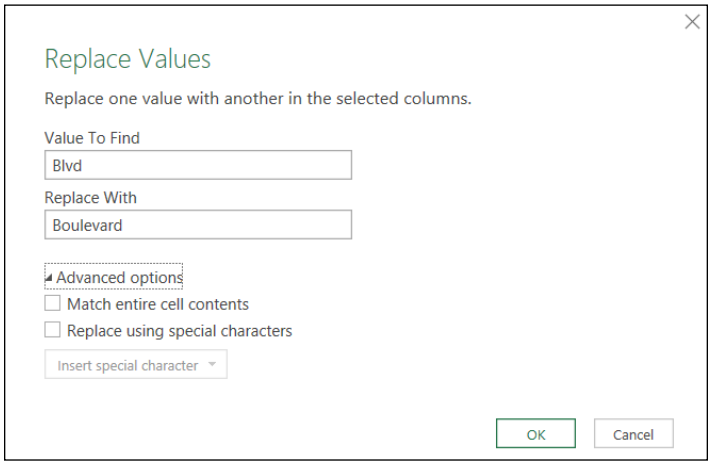


FIGURE 10-10: Replacing text values.

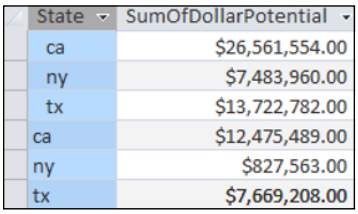
Note that clicking on Advanced Options reveals two optional settings, which are described in this list:

- **Match entire cell contents:** Selecting this option tells Power Query to replace the specified value only if that value makes up the entire contents of the record. This is useful when you're attempting to replace a value such as 0 (zero) without replacing, for instance, all the zeros in the number 1,000.

» **Replace Using Special Characters:** Selecting this option allows you to use special invisible characters such as line feed, carriage return, or tab as replacement text. This option is useful when you want to force an indent or reposition the text so that it shows up on two lines.

Trimming and cleaning text

When you receive a data set from a mainframe system, a data warehouse, or even a text file, it isn't uncommon to have field values that contain leading and trailing spaces. These spaces can cause some abnormal results, especially when you're appending values with leading and trailing spaces to other values that are clean. To demonstrate this concept, look at the data set in Figure 10-11.



State	SumOfDollarPotential
ca	$26,561,554.00
ny	$7,483,960.00
tx	$13,722,782.00
ca	$12,475,489.00
ny	$827,563.00
tx	$7,669,208.00

FIGURE 10-11: Leading spaces can cause issues in analysis.

This view is intended to be an aggregate view that displays the sum of the dollar potential for California, New York, and Texas. However, the leading spaces are forcing each state into two sets, preventing you from discerning the accurate totals.

You can easily remove leading and trailing spaces by using the Trim function in Power Query. Figure 10-12 demonstrates how you would update a field to remove the leading and trailing spaces by using the Trim command found on the Transformation tab.

Again, the Trim command is applied to any column or columns you select. So, you can fix multiple columns at a time by simply selecting them before selecting the Trim command.

Figure 10-12 also shows the Clean command (beneath Trim). Whereas Trim removes leading and trailing spaces, the Clean command removes any invisible characters, such as carriage returns and other nonprintable characters that may slip in from external source systems. These characters are typically rendered in Excel as question marks or square boxes. But in Power Query, they show up as spaces.

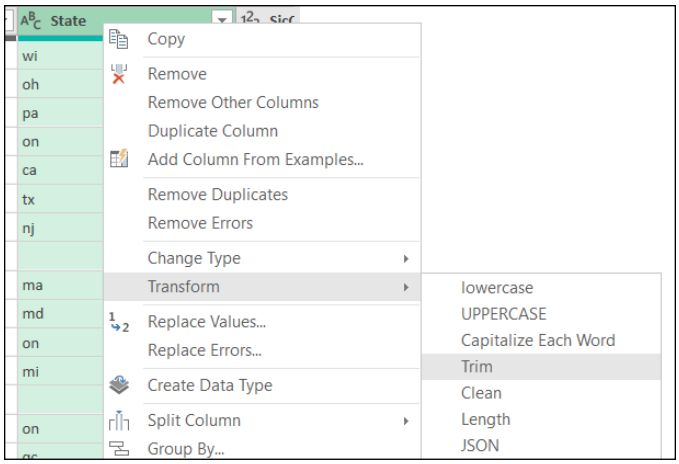


FIGURE 10-12: The Trim command.

If the source system that supplies your data has a nasty habit of including strange characters and leading spaces, you can apply the Trim and Clean functions to sanitize the data set.

TIP

You may already know that the `TRIM` function in Excel removes the leading spaces, trailing spaces, and excess spaces within the given text. Power Query's `TRIM` function removes leading and trailing spaces, but doesn't touch the excess spaces in the text. If excess spaces are a problem in your data, you can deal with them by using the Replace Values function to replace a given number of spaces with only one space.

Extracting the left, right, and middle values

In Excel, the `RIGHT` function, the `LEFT` function, and the `MID` function allow you to extract portions of a string starting from different positions:

- `LEFT`: Returns a specified number of characters, starting from the leftmost character of the string. The required arguments for the `LEFT` function are the text you're evaluating and the number of characters you want returned. For example, `LEFT("70056–3504", 5)` would return five characters starting from the leftmost character (`70056`).
- `RIGHT`: Returns a specified number of characters starting from the rightmost character of the string. The required arguments for the `RIGHT` function are the text you're evaluating and the number of characters you want returned. For example, `RIGHT("Microsoft", 4)` would return four characters starting from the rightmost character (`soft`).

- `MID`: Returns a specified number of characters starting from a specified character position. The required arguments for the `MID` function are the text you're evaluating, the starting position, and the number of characters you want returned. For example, `MID("Lonely", 2, 3)` would return either three characters starting from the second character or character number 2 in the string (one).

Power Query has equivalent functions exposed through the Extract command, found on the Transformation tab (see Figure 10-13). The Extract command allows you to get specified characters from a value.

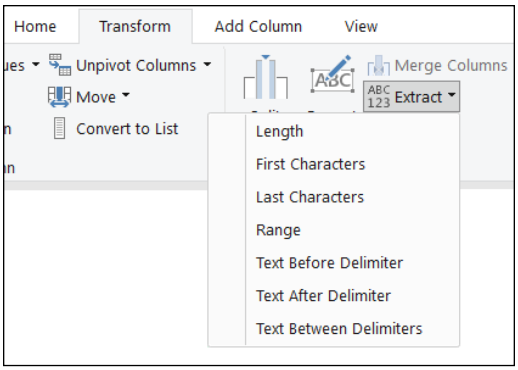


FIGURE 10-13: The Extract command allows you to pull out parts of the text found in a column.

The options under the Extract command are described in this list:

- **Length:** Transforms a given column into numbers that represent the number of characters in each row (similar to Excel's `LEN` function).
- **First Characters:** Transforms a given column to show a specified number of characters from the beginning of text in each row (similar to Excel's `LEFT` function).
- **Last Characters:** Transforms a given column to show a specified number of characters from the end of text in each row (similar to Excel's `RIGHT` function).
- **Range:** Transforms a given column to show a specified number of characters starting from a specified character position (similar to Excel's `MID` function).

REMEMBER

Applying the Extract command to a column effectively replaces the original text with the results of the operation you choose to apply. That is to say, the original text isn't visible in the table after you apply the Extract command. For this reason, you may want to first copy the column and perform the extraction on the duplicate column.

You can create a copy of a column by right-clicking on the column and selecting Duplicate Column. When the duplicate column is created, it's the last (rightmost) column of the table.

Extracting first and last characters

To extract the first *N* characters of text, highlight the column, select Extract ➪ First Characters, and then use the dialog box shown in Figure 10-14 to specify the number of characters you want to extract. In this case, the first three characters of the Phone field are extracted.

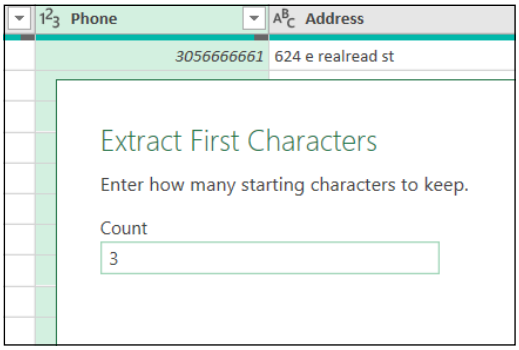


FIGURE 10-14: Extracting the first three characters of the Phone field.

To extract the last *N* characters of text, highlight the column, select Extract ➪ Last Characters, and then use the dialog box to specify the number of characters you want extracted.

Extracting middle characters

To extract the middle *N* characters of text, highlight the column and select Extract ➪ Range. The dialog box shown in Figure 10-15 opens.

The idea here is to tell Power Query to extract a specific number of characters starting from a certain position in the text. For example, the SicCode field is a 4-digit field. If you want to extract the two middle numbers of the SicCode, you would tell Power Query to start at the second character and extract two characters from there.

As you can see in Figure 10-15, the starting index is set to 2 (starting at the second character) and the number of characters is set to 2 (extract two characters from the starting index).

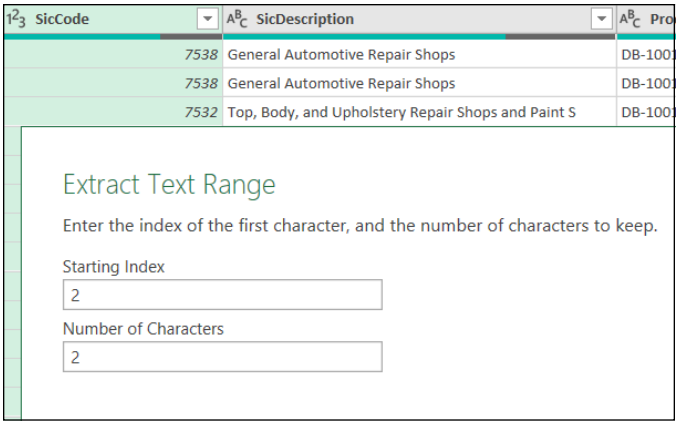


FIGURE 10-15: Extracting the two middle characters of the SicCode.

Splitting columns using character markers

Have you ever gotten a data set where two or more distinct pieces of data were jammed into one field and separated by commas? For example, a field labeled Address may have a single text value that represents address, city, state, and postal code. In a proper data set, this text would be split into four fields.

In Figure 10-16, you can see that the values in the ContactName field are strings that represent Last name, First name, and Middle initial. Imagine that you need to split this column string into three separate fields.

ContactName	ContactTitle	
DAIMIRT, TAM, G.	Manager	30
THEMPSENJR, MAKE, G.	Manager	66
SCETT, ANDY, T.	Owner	34
MCKINZAE, DAVE, G.	Owner	40
NILSEN, REBIRT, T.	Owner	44
KILLIRMAN, DAVAD, G.	Manager	24
CELIMAN, TERRANCE, G.	Owner	46
SANSENE, TERRANCE, G.	Owner	46
GIRVES, STIPHIN, G.	Owner	64
BIRNSTIAN, PEIL, G.	Manager	14
MCMANIR, STIVE, A.	Plant Manager	40

FIGURE 10-16: The Split Column command can easily split the ContactName Field into three separate columns.

Although this isn't a straightforward undertaking in Excel, it can be done fairly easily with the Split Column command (found on the Transform tab).

Selecting the Split Column command reveals two options; this list describes what you can do with them:

- **By Delimiter:** Split a column based on specific characters such as commas, semicolons, or spaces. This option is useful for parsing names or addresses or any field that contains multiple data points separated by delimiting characters.
- **By Number of Characters:** Split a column based on a specified number of characters — useful for parsing uniform text at a defined character position.
- **By Positions:** Split a column based on fixed numeric positions you specify.
- **By Lowercase to Uppercase:** Split a column where the case changes from lowercase to uppercase.
- **By Uppercase to Lowercase:** Split a column where the case changes from uppercase to lowercase.
- **By Digit to Non-Digit:** Split a column where the previous character is a digit, and the next consecutive character is a non-digit.
- **By Non-digit to Digit:** Split a column where the previous character is a non-digit and the next consecutive character is a digit.

In the example (refer to Figure 10-16), the contact names are made up of last names, first names, and middle initials, all separated *(delimited)* by commas. So the By Delimiter option is the one I show you how to use.

You can highlight the ContactName field and select Split Column ➪ By Delimiter to open the Split by Column Delimiter dialog box, shown in Figure 10-17.

This list describes the inputs:

- **Select or Enter Delimiter:** Use the drop-down menu to choose the delimiter that will define where the values should be split. If the delimiter isn't listed as a choice on the drop-down list, you can select the Custom option and define your own.
- **Split:** Select how you want Power Query to use the specified delimiter. Power Query can split the column only on the first occurrence of the delimiter (the leftmost delimiter) — effectively creating two columns. Alternatively, you can tell Power Query to split the column only on the last occurrence of the delimiter (the rightmost delimiter) — again, creating two columns. The third option is to tell Power Query to split the column at each occurrence of the delimiter.

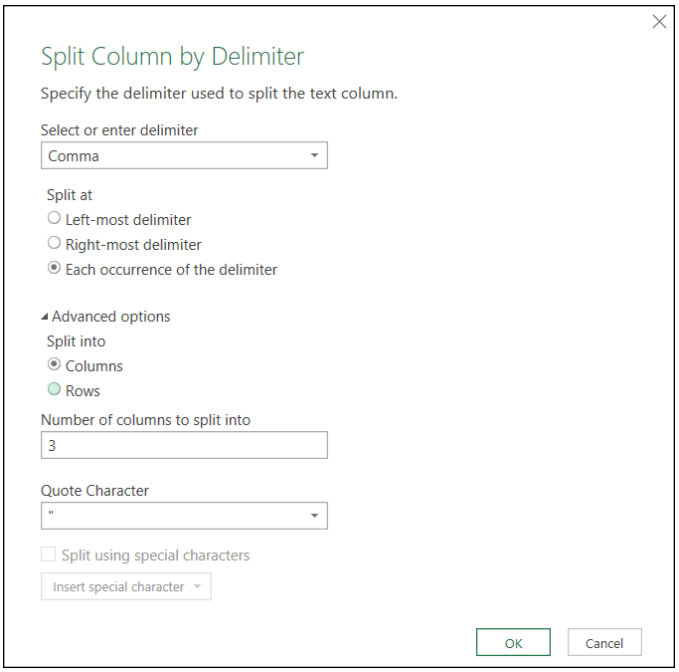


FIGURE 10-17: Splitting the ContactName column at every occurrence of a comma.

» **Advanced Options:** By default, selecting the option to split the column at each occurrence of the delimiter creates as many columns as there are delimiters. You can use the advanced options to override the default and limit the number of columns to create. You also have the advanced option to split your value into new rows instead of new columns.

Figure 10-18 shows the new columns created after the ContactName column is split at each comma. As you can see, three new fields are created. You can rename a field by right-clicking the field name and selecting the Rename option.

Pivoting and unpivoting fields

You often encounter data sets like the one shown in Figure 10-19, where important headings (like Month) are spread across the top of the table, pulling double duty as column labels and actual data values. This matrix layout is easy to look at in a spreadsheet, but it causes problems when attempting to perform any kind of data analysis that requires aggregation or grouping, for example.

Power Pivot offers an easy way to unpivot and pivot columns, allowing you to quickly convert matrix-style tables to tabular data sets (and vice versa).

ContactName.1	ContactName.2	ContactName.3	Contac
DAIMIRT	TAM	G.	Manager
THEMPSENJR	MAKE	G.	Manager
SCETT	ANDY	T.	Owner
MCKINZAE	DAVE	G.	Owner
NILSEN	REBIRT	T.	Owner
KILLIRMAN	DAVAD	G.	Manager
CELIMAN	TERRANCE	G.	Owner
SANSENE	TERRANCE	G.	Owner
GIRVES	STIPHIN	G.	Owner
BIRNSTIAN	PEIL	G.	Manager

FIGURE 10-18: The ContactName field has been split successfully into three columns.

Product_Description	Jan	Feb	Mar	Apr	May
Cleaning & Housekeeping Services	6219.66	4263.92	5386.12	6443.99	436
Facility Maintenance and Repair	3255.82	9490	4409.23	4957.62	885
Fleet Maintenance	5350.03	8924.71	6394.43	6522.46	946
Green Plants and Foliage Care	2415.08	2579.61	2401.91	2981.01	270
Landscaping/Grounds Care	5474.22	4500.52	5324.36	5705.68	526
Predictive Maintenance/Preventative Maintenance	9810.95	10180.23	9626.31	11700.73	1094
Cleaning & Housekeeping Services	2840.76	2997.18	2096.78	4102.2	4
Facility Maintenance and Repair	16251.01	35878.99	18368.55	21843.53	287
Fleet Maintenance	22574.77	36894.89	22016.38	27871.1	3198
Green Plants and Foliage Care	48250.9	90013.42	51130.17	75527.58	6941
Landscaping/Grounds Care	19401.16	21190.57	21292	20918.35	1946

FIGURE 10-19: Matrix layouts are problematic for data analysis.

Unpivot Columns command

The Unpivot Columns command lets you select a set of columns and convert those columns into two columns: one column consisting of the old column labels and another containing the old column data.

For instance, in Figure 10-19, the month columns can be unpivoted by selecting the months and then clicking the Unpivot Columns command.

The resulting table is shown in Figure 10-20. Note that the month labels are now entries in a new column named Attribute. The month values are now in a new column named Value. You can, of course rename these columns to Month and Revenue, for example.

Unpivot Other Columns command

As helpful as the Unpivot Columns command is, it has a flaw: You have to explicitly select the months that you want unpivoted. But what if the number of columns is ever growing? What if you unpivot January through June, but next month a new data set will arrive with July and then August and then September? Because

the Unpivot Columns command forces you to essentially hard-code the columns you want unpivoted, you have to redo the unpivot each and every month.

Product_Description	Attribute	Value
Cleaning & Housekeeping Services	Jan	6219.66
Cleaning & Housekeeping Services	Feb	4263.92
Cleaning & Housekeeping Services	Mar	5386.12
Cleaning & Housekeeping Services	Apr	6443.99
Cleaning & Housekeeping Services	May	4360.14
Cleaning & Housekeeping Services	Jun	5097.46
Cleaning & Housekeeping Services	Jul	7566.19
Cleaning & Housekeeping Services	Aug	4263.92
Cleaning & Housekeeping Services	Sep	7245.64
Cleaning & Housekeeping Services	Oct	3847.15
Cleaning & Housekeeping Services	Nov	6540.21
Cleaning & Housekeeping Services	Dec	5610.45
Facility Maintenance and Repair	Jan	3255.82
Facility Maintenance and Repair	Feb	9490

FIGURE 10-20: All months are now in a tabular format.

Fortunately, you can avoid this problem with the Unpivot Other Columns command. This nifty command allows you to unpivot by selecting the columns that you want to remain static and telling Power Query to unpivot all other columns.

For instance, Figure 10-21 demonstrates that rather than select the month columns, you can select the Market and Product_Description columns and then select Unpivot Other Columns from the Unpivot Columns drop-down menu.

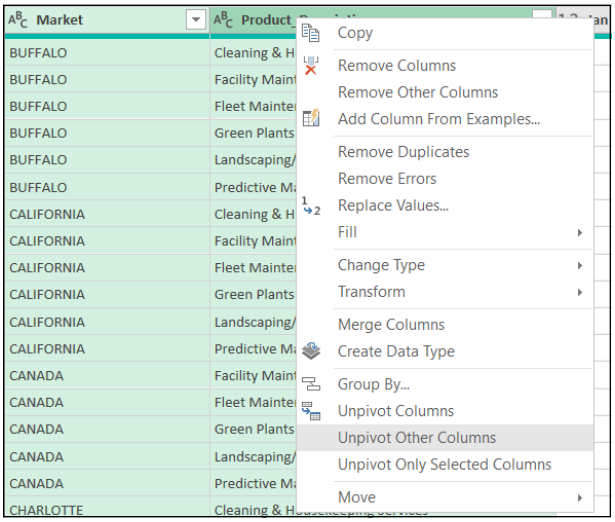


FIGURE 10-21: Use Unpivot Other Columns when the number of matrix columns is variable.

Now, it doesn't matter how many new month columns are added or removed each month. Your query always unpivots the correct columns.

TIP

Always use the Unpivot Other Columns option. Even if you don't anticipate new matrix columns, it's always a good bet to use the option that offers more flexibility for those unexpected changes in data.

Pivot Columns command

If you find that you need to transform your data from a tabular layout to a matrix-style layout, you can use the Pivot Columns command.

Simply select the columns that will make up the header labels and values for the new matrix columns, and then select the Pivot Column command, shown in Figure 10-22.

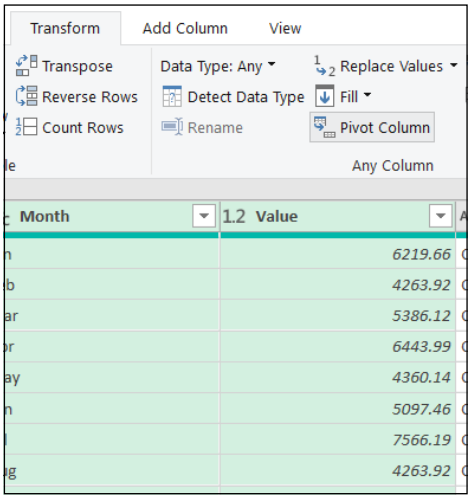


FIGURE 10-22: Pivoting the Month and Value columns.

Before finalizing the pivot operation, Power Query opens a dialog box (shown in Figure 10-23) to confirm the value column and the aggregation method. By default, Power Query uses the Sum operation to aggregate the data into the matrix format. You can override this default setting by selecting a different operation (count, average, or median, for example). You can even specify that you don't want aggregation performed. Clicking the OK button finalizes the pivot operation.

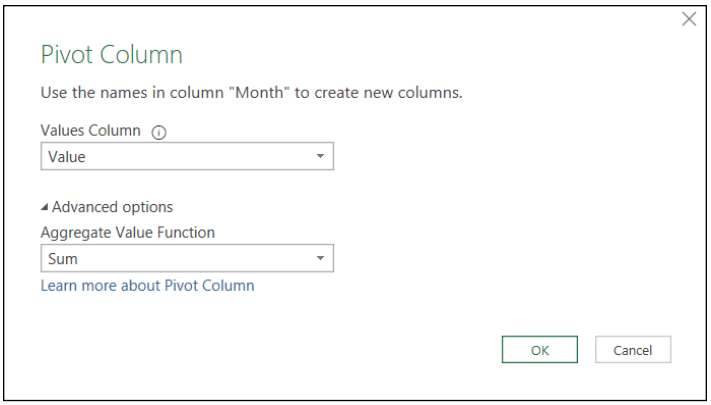


FIGURE 10-23: Confirm the aggregation operation to finalize the pivot transformation.

Creating Custom Columns

When transforming your data, you sometimes have to add your own columns to extract key data points, create new dimensions, or even create your own calculations.

You start a new custom column by selecting the Add Column tab and clicking the Custom Column command. The Custom Column dialog box (shown in Figure 10-24) appears; here you specify the contents of your new column through the use of Power Query formulas. When you add a new custom column, it won't do anything until you provide a formula that gives it some utility.

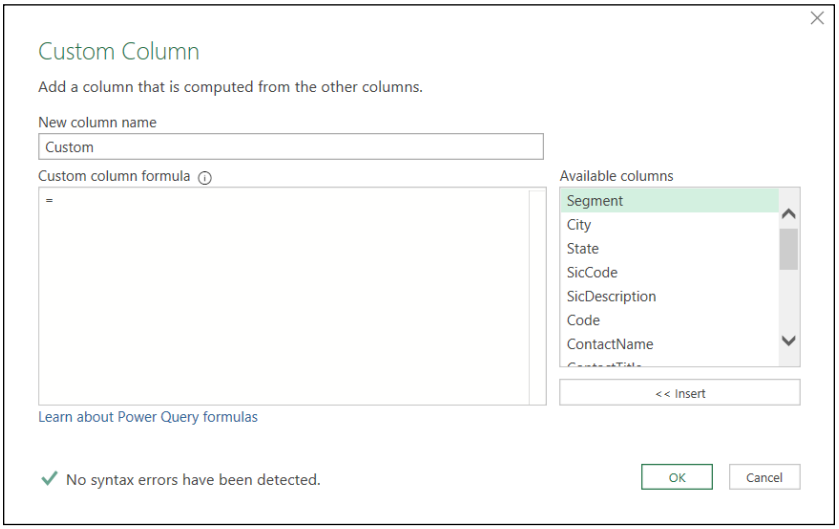


FIGURE 10-24: The Custom Column dialog box.

As for the Custom Column dialog box, there's not much to it. The inputs are described in this list:

- **New column name:** An input box where you enter a name for the column you're creating.
- **Available columns:** A list box that contains the names of all columns in the query. Double-click any column name in this list box to automatically place it in the formula area.
- **Custom column formula:** The area where you type the formula.

As in Excel, a formula can be as simple as `=1` or as complicated as an `if` statement that applies some conditional logic. Over the next few sections, I walk you through a few examples of creating custom columns to go beyond the functionality provided via the user interface.

But before diving into building Power Query formulas, you should understand how Power Query formulas differ from those in Excel. Here are some high-level differences to be aware of:

- **No cell references.** You can't reach outside the Custom Column dialog box to select a range of cells. Power Query formulas work by referencing columns, not cells.
- **Excel functions don't work.** The Excel functions you're used to don't work in Power Query. Power Query has many of the same kinds of functions as Excel, but it has its own formula language.
- **Everything is case sensitive.** In Excel, you can type in all lowercase or all uppercase letters and your formulas will work. Not so in Power Query. To Power Query, sum, Sum, and SUM are three different items, and only one of them is acceptable.
- **Data types matter.** Some fields are text fields, other fields are number fields, and still others are date fields. Excel does a good job of handling formulas that mix fields of differing data types. The Power Query formula language, which is extremely sensitive to data types, doesn't have the built-in intelligence to gracefully handle data type mismatches. Data type issues are resolved with conversion functions, as covered later in this chapter.
- **No tool tips or intelligence help.** Excel is quick to throw up a tool tip or a menu of options when you start entering a new formula. Power Query has none of that. As of this writing, Power Query offers only a Learn About Power Query Formulas link to a Microsoft site dedicated to Power Query.

Don't panic. Power Query formulas are not as gloomy as they sound. Let's start with a simple custom column.

Concatenating with a custom column

Earlier in this chapter, I tell you how to concatenate values from two or more columns by using the Merge Columns command. Although this command is easy to use, it results in the original source columns being removed. You will likely want to concatenate values but still retain the source columns.

In these instances, you can create your own custom column. Follow these steps to create a new column that merges the Type and Code columns:

1. **While in the Query Editor, choose Add Column ➪ Custom Column.**
2. **Place the cursor in the Custom Column Formula area (after the equal sign).**
3. **Find the Type column in the Available Columns list and double-click on it.**

   You see [Type] pop into the formula area after the equal sign.
4. **After [Type], enter the following text:** & "-" &**.**

   This step ensures that the values in the two columns are separated by a hyphen.
5. **Enter** Number.ToText()**.**

   `Number.ToText()` is a Power Query function that converts a number to text format on the fly so that it can be used with other text. In this case, because the Code field is formatted as a number, you need convert it on the fly to join it to the Type field. I tell you more about data type conversions later in this chapter.
6. **Place the cursor between the parentheses for the** `Number.ToText()` **function and then find the Code column in the Available Columns list and double-click it.**

   You see `[Code]` pop into the formula area between the parentheses.
7. **In the New Column Name input, enter** MyFirstColumn**.**

   At this point, the dialog box should look similar to the one shown in Figure 10-25. Note the message at the bottom of the dialog box: `No syntax errors have been detected.` This message refers to the syntax you entered. Every time you create or adjust a formula, you'll want to ensure that this message states that no errors have been detected.
8. **Click OK to add the custom column.**

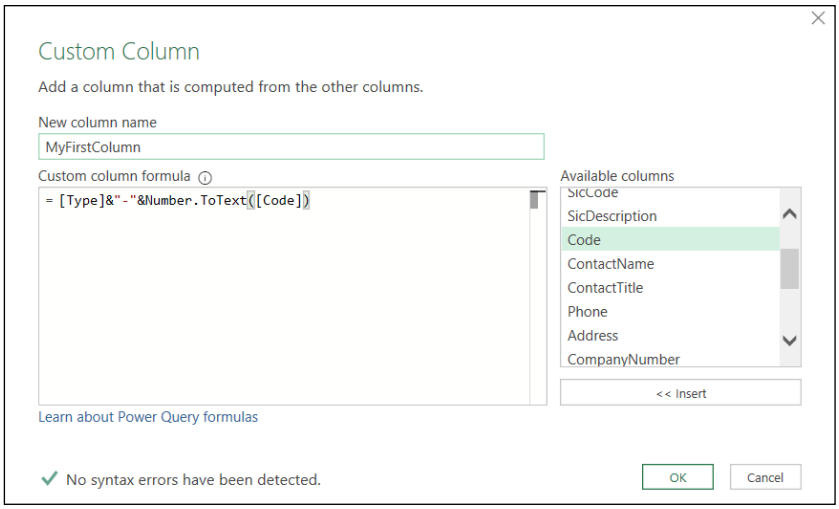


FIGURE 10-25: A formula to merge the Type and Code columns.

If all goes well, you have a new custom column that concatenates two fields. In this basic example, you see the basic foundation of how Power Query formulas work.

Understanding data type conversions

When working with formulas in Power Query, you inevitably need to perform some action on fields that have differing data types, as in the exercise in the previous section, where I show you how to merge the Type column (a text field) with the Code column (a numeric field). In that example, you use a conversion function to change the data type of the Code field so that it can be temporarily treated as a text field.

A conversion function does exactly what it sounds like: It converts data from one data type to another.

Table 10-1 lists common conversion functions. As demonstrated in the previous section, you simply wrap these functions around the columns that need converting.

To find and change the data type for a field, place the cursor in the field and then select the Data Type drop-down menu on the Transform tab (see Figure 10-26). The data type at the top is the type of field the cursor is in. You can edit the data type for the field by selecting a new type from the drop-down list.

TABLE 10-1 Common Conversion Functions

Convert From	To	Function
Date	Text	`Date.ToText()`
Time	Text	`Time.ToText()`
Number	Text	`Number.ToText()`
Text	Number	`Number.FromText()`
Text Dates	Date	`Date.FromText()`
Numeric Dates	Date	`Date.From()`

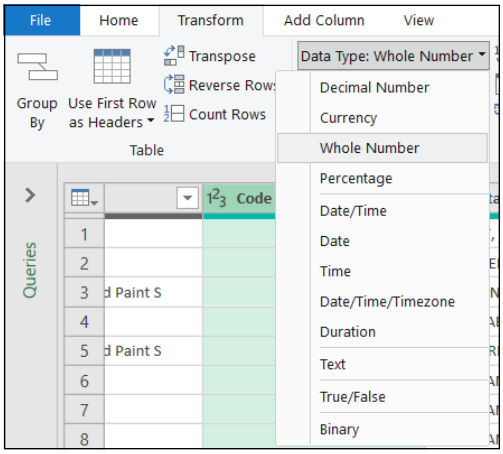


FIGURE 10-26: Use the Data Type drop-down menu to discover and select the data type of a given field.

Spicing up custom columns with functions

With a few basic fundamentals and a little knowledge of Power Query functions, you can create transformations that go beyond what you can do by using the Query Editor. In this example, I show you how to use a custom column to pad numbers with zeros.

You may encounter a situation where key fields are required to have a certain number of characters to make the data able to interface with peripheral platforms such as ADP or SAP. Suppose that the CompanyNumber field must be ten characters long. Those company numbers that aren't ten characters long must be padded with enough leading zeros to create a ten-character string.

The secret to this supplying the proper number of character is to add ten leading zeros to every company number, regardless of the current length, and then pass them through a function similar to the `RIGHT` function, which extracts only the rightmost ten characters.

For example, you would first convert company number 29875764 to 0000000000029875764; then you would use the `RIGHT` function to extract only the rightmost ten characters, leaving you with 0029875764.

Although you follow essentially two steps, you can accomplish the same result with only one custom column. Here's how:

1. **While in the Query Editor, choose Add Column ➪ Custom Column.**
2. **Place the cursor in the Custom Column Formula area (after the equal sign).**
3. **Enter ten zeros in quotes (as in** "0000000000"**) followed by an ampersand (&).**
4. **Enter** Number.ToText()**.**
5. **Place the cursor between the parentheses for the** `Number.ToText()` **function and then find the CompanyNumber column in the Available Columns list and double-click it.**

   You see [CompanyNumber] pop into the formula area between the parentheses.

   At this point, the formula area should contain this syntax:

   ```
   "0000000000"&Number.ToText([CompanyNumber])
   ```

   This formula results in nothing more than a concatenation of ten zeros and the CompanyNumber. The goal is to go further and extract only the rightmost ten characters. Unfortunately, the `RIGHT` function is an Excel function that doesn't work in Power Query. However, Power Query does have an equivalent function named `Text.End()`. Like the `RIGHT` function, the `Text.End()` function requires a couple of parameters: the text expression and the number of characters to extract:

   ```
   Text.End([MyText], 10)
   ```

   In this example, the text expression is the formula, and the number of characters to extract is 10.

6. **Enter** Text.End **before your existing formula, and then follow the formula with,** 10**.**

   Here's the final syntax:

   Text.End("0000000000"&Number.ToText([CompanyNumber]), 10)

7. **In the New Column Name input, enter** TenDigitCustNumber**.**

   At this point, the dialog box should look similar to the one shown in Figure 10-27. Again, note the message at the bottom of the dialog box. This message will tell you if you have a syntax error in your formula. Make sure that the message at the bottom of the dialog box reads `No syntax errors have been detected`.

8. **Click OK to apply the custom column.**

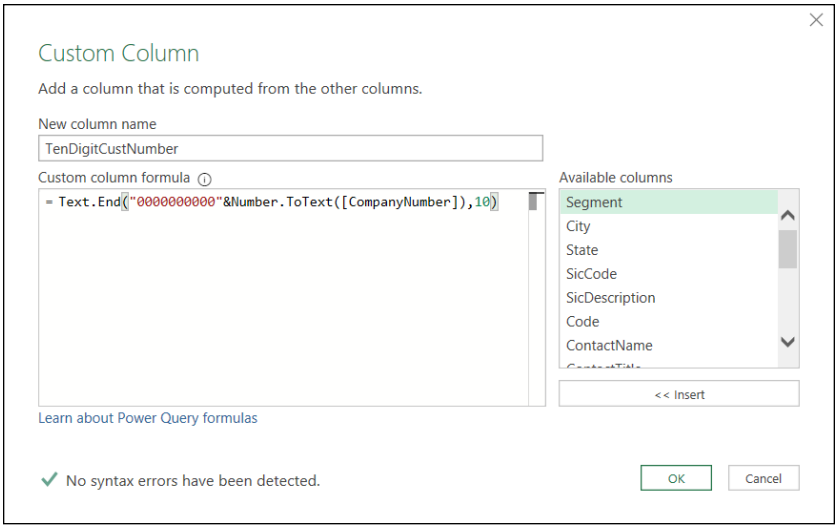


FIGURE 10-27: A formula to create a consistent ten-digit padded CompanyNumber.

Table 10-2 lists other Power Query functions that are useful in extending the capabilities of custom columns. Take a moment to examine the list of functions and note how they differ from their Excel equivalents. Remember that Power Query functions are case sensitive.

Adding conditional logic to custom columns

As you might notice in Table 10-2, Power Query has a built-in `IF` function. The `IF` function is designed to test for conditions and provide different outcomes based on the results of those tests. In this section, you'll see how you can control the output of your custom columns by utilizing Power Query's `IF` function.

TABLE 10-2 Useful Transformation Functions

Excel Function	Power Query Function
LEFT([Text], [Number])	Text.Start([Text], [Number])
RIGHT([Text], [Number])	Text.End([Text], [Number])
MID([Text], [StartPosition], [Number])	Text.Range([Text], [StartPosition], [Number])
FIND([Find], [Within])	Text.PositionOf([Within], [Find])
IF([Expression], [Result1], [Result2])	if [Expression] then [Result1] else [Result2]
IFERROR([Procedure], [FailResult])	try [Procedure] otherwise [FailResult]

As in Excel, Power Query's `IF` function evaluates a specific condition and returns a result based on a true or false determination:

```
if [Expression] then [Result1] else [Result2]
```

REMEMBER

In Excel, you think of commas in an `IF` function as `then` and `else` statements. The formula IF(Babies = 2 , "Twins", "Not Twins") would translate to this: If Babies equals 2, then Twins, else Not Twins In Power Query, you don't use commas. You spell out the entire expression.

You can also use the `IF` function to save steps in your analytical processes and, ultimately, save time. For example, you may need to tag customers as either large customers or small customers, based on their dollar potential. You decide to add a custom column that contains either "LARGE" or "SMALL" based on the revenue potential of the customer.

With the help of the `IF` function, you can tag all customers with one custom column that uses this formula:

```
IF [2020 Potential Revenue]>=10000 then "LARGE" else "SMALL"
```

This function tells Power Query to evaluate the [2020 Potential Revenue] field for each record. If the potential record is greater than or equal to 10,000, use the word LARGE; if not, use the word SMALL.

Figure 10-28 demonstrates this `if` statement as it is applied in the Custom Column dialog box.

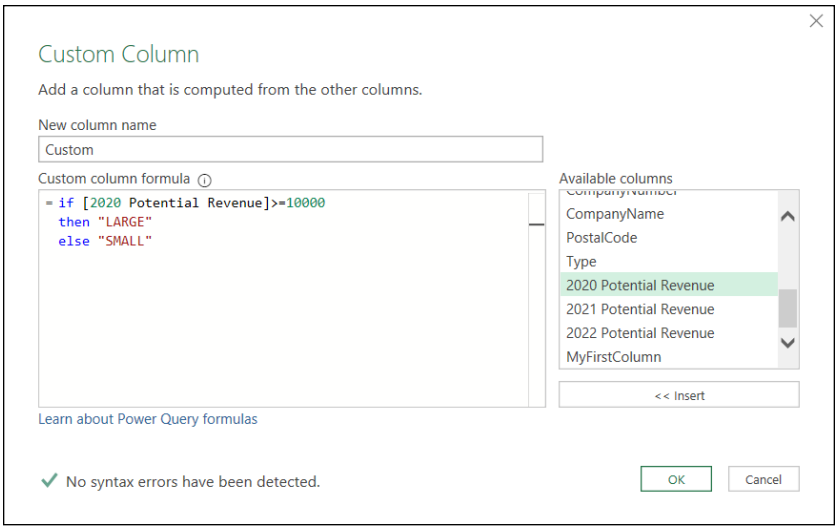


FIGURE 10-28: Applying an IF statement in a custom column.

TIP

Power Query pays no attention to white space, so you can add as many spaces and carriage returns as you want. As long as the correct case and spelling are used, Power Query doesn't complain.

Figure 10-28 illustrates how separating formulas into separate lines can make them much easier to read.

Grouping and Aggregating Data

In some cases, you may need to transform your data set into compact groups in order to get it into a manageable size of unique values. You may even need to summarize numerical values into an aggregate view. An *aggregate view* is a grouped snapshot of your data that shows sums, averages, counts, and more.

Power Query offers a Group By feature that enables you quickly group data and create aggregate views. Follow these steps to use the Group By feature:

1. **While in the Query Editor, select the Group By command on the Transform tab.**

   The Group By dialog box opens.

2. **From the Group By drop-down menu, select the field you want to group by. Click the plus sign (+) above the Group By drop-down list to add additional fields to grouping.**

   Figure 10-29 shows grouping by State and City.

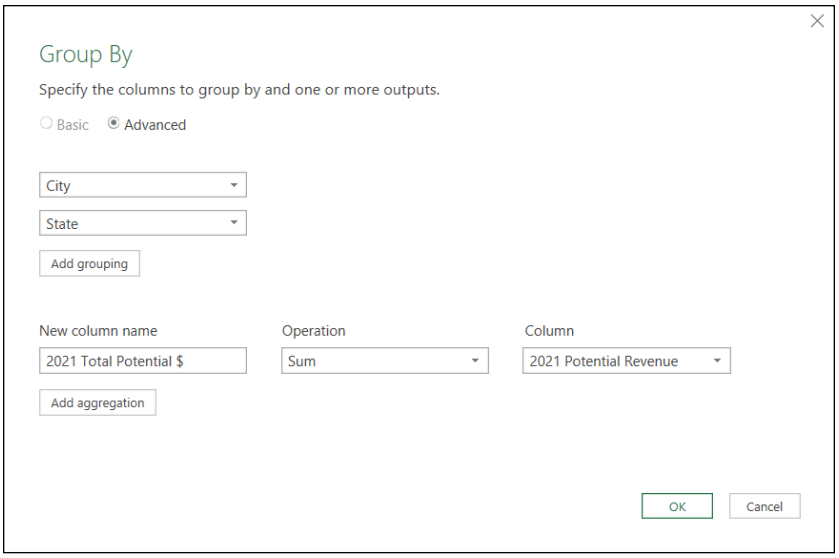


FIGURE 10-29: Using the Group By dialog box to create a view of 2021 Total Potential by State and City.

3. **Use the New Column Name input box to give the new aggregate column a name (for example, 2021 Total Potential).**
4. **From the Operation drop-down list, select the kind of aggregation you want to apply (Sum, Count, Avg, Min, Max, and so on).**
5. **Use the Column drop-down list to choose the column that will be aggregated (for example, 2021 Potential Revenue).**
6. **Click the OK button to confirm and apply your changes.**

   Figure 10-30 illustrates the resulting output.

	City	State	2021 Total Potential $
1	adams	wi	6534
2	akron	oh	54754.34471
3	allentown	pa	77517.35
4	alliston	on	5757.260526
5	anaheim	ca	97655.01111
6	austin	tx	77736.85322
7	avenel	nj	34776.08889

FIGURE 10-30: The resulting aggregate view by State and City.

REMEMBER

When you apply the Group By feature, Power Query removes all columns that were not used when configuring the Group By dialog box. This leaves you with a clean view of just your grouped data.

Working with Custom Data Types

The Custom Data feature of Power Query allows you to store multiple columns of data in one column as metadata. Excel formulas can then interact with these rich data types to expose the stored data within. In this section, you explore the basics of using custom data types in your reporting.

You can follow along with the examples in this chapter by downloading the `CustomDataTypes.xlsx` sample file from `www.dummies.com/go/excelpowerpivotpowerqueryfd2e`. Open the file, and then choose Data➪From Table/Range.

To be frank, the term *custom data type* is a bit unfortunate, because it doesn't really correlate with the traditional meaning of data type (number, date, currency, and so on). In Power Query, you can think of a custom data type as a kind of container that allows you to store the data for many columns and then use that data elsewhere in your workbook.

You can get a sense of the power of this feature by following these steps:

1. **In the Query Editor, right-click the Employee column and select Create Data Type.**

   The Create Data Type dialog box, shown in Figure 10-31, appears.

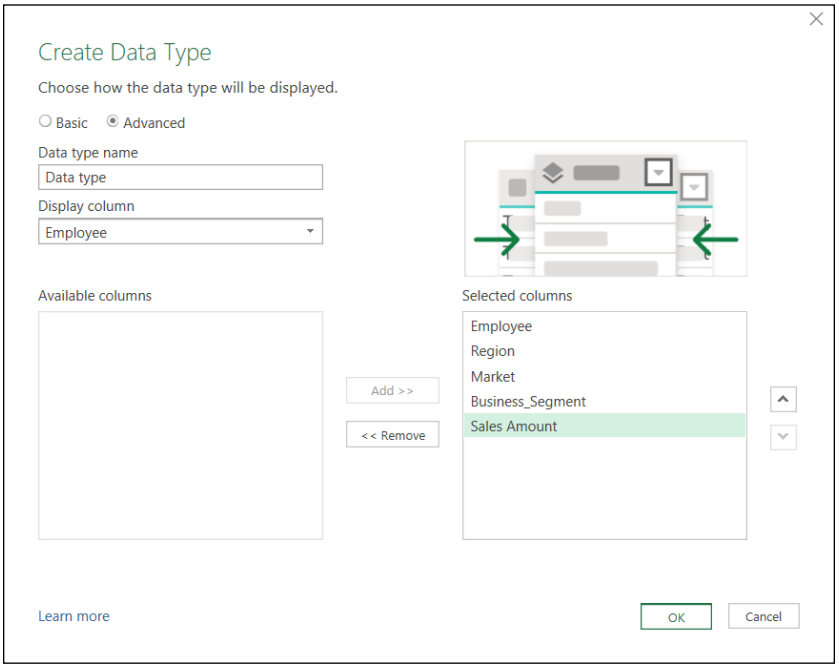


FIGURE 10-31: Creating a custom data type.

2. **Click the Advanced option to reveal a list of available fields.**

   The idea here is that the Display column (Employee, in this case) will be a kind of container that will hold data from other columns you specify.

3. **Select the Region column in the list of available columns and then click the Add button.**

   Repeat for each column in the list of available fields.

4. **Click the OK button to confirm your changes.**

At this point, your data preview window will show only the Employee column because that's the column you selected as your Display column. As you can see in Figure 10-32, clicking any value in the Employee column now results in a table below the data preview window, which shows all the data contained for that value.

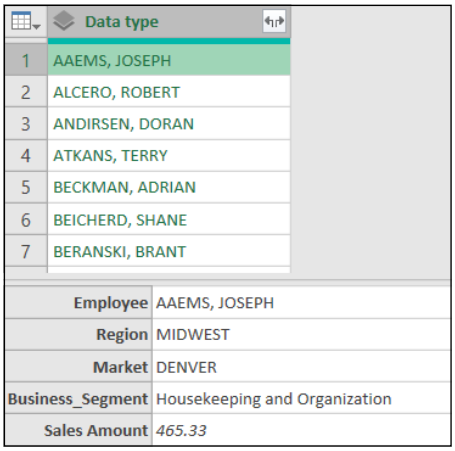


FIGURE 10-32: Each value in a data type column contains the data for underlying columns, as shown below the data preview window.

Click the Close & Load button on the Home tab of the Power Query Editor to send your results to a new worksheet. In Figure 10-33, you can see that the output of this query is a list of values from the Employee column, each containing the Data Type icon designed to let you know that the value is part of a data type. The smart icon to the right of the table allows you to add any of the underlying values for that data type.

The real power of custom data types is the ability to reference underlying values via simple formulas. To illustrate one way to use data types, take a look at Figure 10-34. Here, I loaded a data validation drop-down with the values from the Employee column (the Data Type header). You can see the icon next to the selected employee name, confirming the value is a data type. Now you can reference that value with a simple formula to see the available columns underneath.

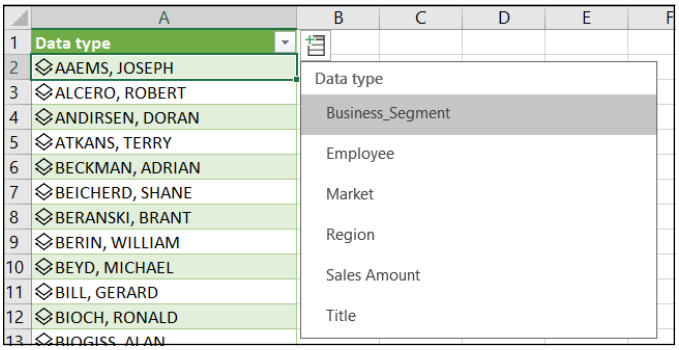


FIGURE 10-33: Data types have a special icon next to each value and allow you to see any values in underlying columns.

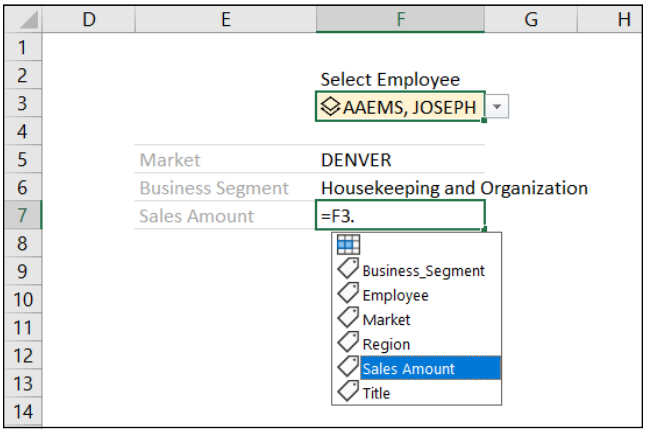


FIGURE 10-34: Referencing a data type value and entering the dot (.) operator allows you to select any of the underlying columns.

That's right. Reference a data type, enter the dot (.) operator, and you'll have access to any of the underlying column values. Selecting a new employee from the data validation drop-down automatically pulls that employee's data.

REMEMBER

Refreshing your query will not only bring in any new data type headers, but also automatically update any of the values resulting from a formula that references your data types.

IN THIS CHAPTER

- Reusing query steps
- Consolidating data with the Append feature
- Understanding join types
- Using the Merge feature
- Fuzzy matching when merging queries

Chapter 11

Making Queries Work Together

Data is frequently analyzed in layers, with each layer of analysis using or building on the previous layer. You may not know it, but you already build layers all the time. For instance, when you build a pivot table using the results of a Power Query output, you're layering your analysis. When you build a query based on a table created by a SQL Server view, you're also creating a layered analysis.

Sure, you would probably love to be able to analyze a single data source and call it a day. But that's not how data analysis works. You often find the need to build queries on top of other queries to get the results you're looking for. That's what this chapter is all about. In this chapter, I help you examine a few ways you can advance your data analysis by making your queries work together.

Reusing Query Steps

Data analysts commonly rely on the same main data tables for all kinds of analysis. Even the simple table shown in Figure 11-1 can be used to create different views: sales by employee, sales by business segment, or sales by region, for example.

	A	B	C	D	E	F
1	Region	Market	Last_Name	First_Name	Business_Segment	Sales Amount
2	MIDWEST	DENVER	AAEMS	JOSEPH	Housekeeping and Organization	$465.33
3	MIDWEST	DENVER	AAEMS	JOSEPH	Landscaping and Area Beautificat	$411.60
4	MIDWEST	DENVER	AAEMS	JOSEPH	Maintenance and Repair	$760.31
5	MIDWEST	DENVER	BEALIY	CHRISTOPHER	Maintenance and Repair	$2,125.38
6	MIDWEST	DENVER	BEALIY	CHRISTOPHER	Landscaping and Area Beautificat	$5,909.14
7	MIDWEST	DENVER	BEALIY	CHRISTOPHER	Maintenance and Repair	$39,829.79
8	MIDWEST	DENVER	BEWMAN	DIRK	Landscaping and Area Beautificat	$319.18
9	MIDWEST	DENVER	BEWMAN	DIRK	Maintenance and Repair	$119.38
10	MIDWEST	DENVER	BIHRINS	KURT	Landscaping and Area Beautificat	$914.20
11	MIDWEST	DENVER	BIHRINS	KURT	Maintenance and Repair	$17,645.38
12	MIDWEST	DENVER	BREWN	SCOTT	Maintenance and Repair	$112.01

FIGURE 11-1: This data can be used as the source for various levels of aggregated analysis.

Of course, you can build separate queries, each performing different grouping and aggregation steps, but that would mean repeating all the data clean-up steps you needed before performing any kind of analysis.

To get a better understanding of how query steps can help save time, take a moment to follow these steps:

1. **Open the `Sales By Employee.xlsx` workbook, found in the sample files for this book.**
2. **Place the cursor anywhere inside the table, and then choose Data ➪ From Table/Range.**

   Power Query opens the Query Editor.
3. **While in the Query Editor, click the Filter drop-down list for the Market field and filter out the Canada market. (Remove the check mark next to Canada.)**
4. **Select the Last_Name and First_Name fields, and then choose Transform ➪ Merge Columns.**

   The Merge Columns dialog box appears.
5. **Create a new Employee field, joining Last_Name and First_Name and separating them by a comma, as shown in Figure 11-2.**

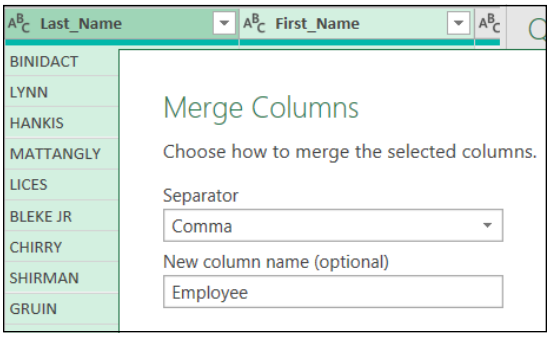


FIGURE 11-2: Merge the Last_Name and First_Name columns to create a new Employee field.

6. Select only the Employee column and click the Group By command on the Transform tab.

The Group By dialog box opens, as shown in Figure 11-3.

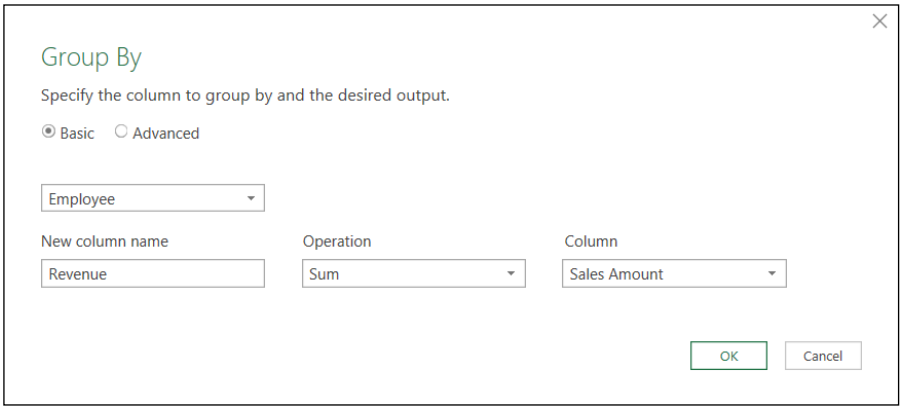


FIGURE 11-3: Group the Employee field and Sum Sales Amount to create a new Revenue column.

7. The goal is to Group By the Employee field to get the Sum of Sales Amount, as shown in Figure 11-3. Name the new aggregated column Revenue.

At this point, you've successfully created a view that shows total revenue by employee. As you can see in Figure 11-4, the query steps include all the preparation work you did before grouping.

What happens if you want to create another analysis using the same data? For instance, what if you want another view that shows Employee sales by business segment?

You could always start from Step 1 and import another copy of the source data, but you'd have to repeat the preparation steps (the steps for Filtered Rows and Merged Columns, in this case).

A better way is to reuse the steps you've already created by extracting them into a new query. The idea is to first decide what steps you want to reuse and then right-click the step immediately below it. In this scenario (refer to Figure 11-4), you keep all query steps until Grouped Rows.

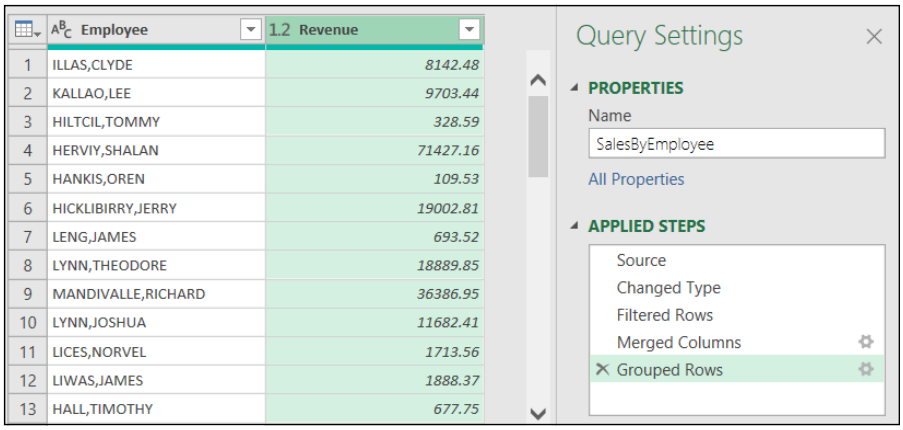


FIGURE 11-4: All the query steps before Grouped Rows are needed in order to prepare the data for grouping.

8. **Right-click the Grouped Rows step and select Extract Previous.**

   The Extract Steps dialog box opens.

9. **Name the new query SalesByBusiness, as shown in Figure 11-5. Click the OK button to confirm.**

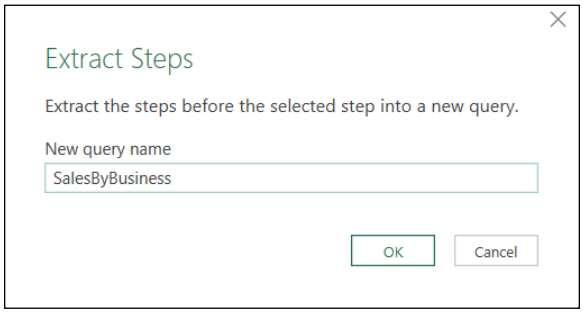


FIGURE 11-5: Naming the new query SalesByBusiness.

After you click OK, Power Query does two things:

- Moves all extracted steps to the newly created query
- Ties the original query to the new query

That is to say, both queries are sharing the extracted steps. You can see the new SalesByBusiness query in the pane on the left, as shown in Figure 11-6. You now can click on the SalesByBusiness query and start applying any needed transformations.

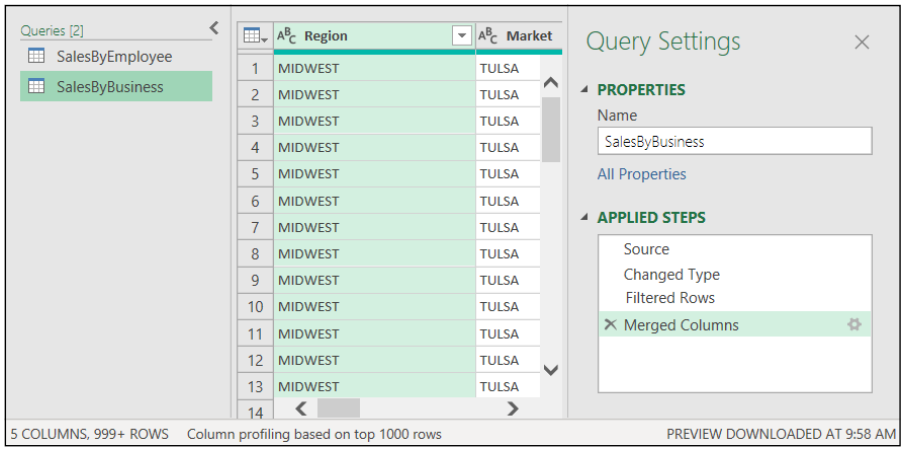


FIGURE 11-6: The two queries are now sharing the extracted steps.

This concept of extracting steps can be a bit confusing. The bottom line is that instead of starting from square one with a brand-new query, you're telling Power Query you want to create a new query that uses the steps you've already created.

REMEMBER

When two or more queries share extracted steps, the query that contains the extracted steps serves as the data source for the other queries. Because of this link, the query that contains the extracted steps cannot be deleted. You have to first delete all dependent queries before deleting the query that holds the extracted steps.

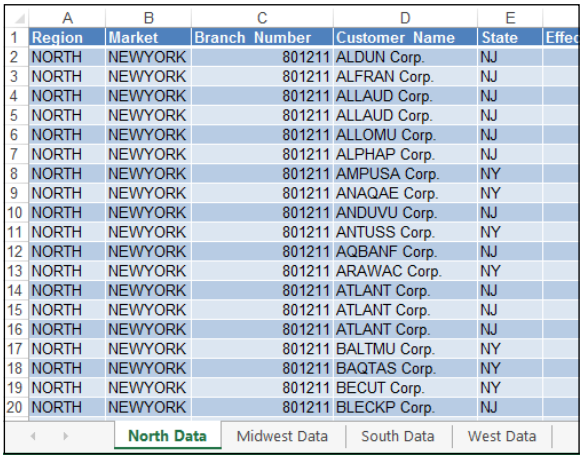
Understanding the Append Feature

Power Query's Append feature allows you to add the rows generated from one query to the results of another query. In other words, you copy records from one query and add them to the end of another.

The Append feature comes in handy when you need to consolidate multiple identical tables into one table. For example, if you have tables from the North, South, Midwest, and West regions, you can consolidate the data from each region into one table using the Append feature.

ON THE WEB

To help you better understand the Append feature, I'll walk you through an exercise that consolidates data from four different regions into one table. In this walkthrough, I use the region data found on four different tabs in the `Appending_Data.xlsx` sample file, shown in Figure 11-7.



	A	B	C	D	E
1	Region	Market	Branch_Number	Customer_Name	State
2	NORTH	NEWYORK	801211	ALDUN Corp.	NJ
3	NORTH	NEWYORK	801211	ALFRAN Corp.	NJ
4	NORTH	NEWYORK	801211	ALLAUD Corp.	NJ
5	NORTH	NEWYORK	801211	ALLAUD Corp.	NJ
6	NORTH	NEWYORK	801211	ALLOMU Corp.	NJ
7	NORTH	NEWYORK	801211	ALPHAP Corp.	NJ
8	NORTH	NEWYORK	801211	AMPUSA Corp.	NY
9	NORTH	NEWYORK	801211	ANAQAE Corp.	NY
10	NORTH	NEWYORK	801211	ANDUVU Corp.	NJ
11	NORTH	NEWYORK	801211	ANTUSS Corp.	NY
12	NORTH	NEWYORK	801211	AQBANF Corp.	NJ
13	NORTH	NEWYORK	801211	ARAWAC Corp.	NY
14	NORTH	NEWYORK	801211	ATLANT Corp.	NJ
15	NORTH	NEWYORK	801211	ATLANT Corp.	NJ
16	NORTH	NEWYORK	801211	ATLANT Corp.	NJ
17	NORTH	NEWYORK	801211	BALTMU Corp.	NY
18	NORTH	NEWYORK	801211	BAQTAS Corp.	NY
19	NORTH	NEWYORK	801211	BECUT Corp.	NY
20	NORTH	NEWYORK	801211	BLECKP Corp.	NJ

North Data | Midwest Data | South Data | West Data

FIGURE 11-7: The data found on each region tab needs to be consolidated into one table.

ON THE WEB

You can find the `Appending_Data.xlsx` workbook in the sample files for this book.

Creating the needed base queries

The Append feature works only on existing queries. That is to say, no matter what kind of data sources you have, you need to import them into Power Query before you can append them together. In this case, it means importing all the region tables into queries.

Follow these steps to import the needed base queries:

1. **Go to the North Data worksheet, place the cursor anywhere inside the table, and then choose Data ➪ From Table/Range.**

   The Query Editor activates, showing you the contents of the table you just imported.

   To finalize the creation of the query, you need to close and load the query. Now, because you're creating this query simply for the purpose of appending it to other queries, you don't need to close and load to the workbook. You can choose instead to close and load the data as connection-only.

2. **On the Home tab of the Query Editor, click the drop-down arrow under the Close & Load command and select Close & Load To.**

3. **In the Import Data dialog box, choose the option Only Create Connection, and then click the OK button.**

4. **Repeat Steps 1 through 3 for the other worksheets in the workbook.**

After you've created queries for each region, open the Queries & Connections pane (choose Data➪Queries & Connections in the Excel Ribbon) to see all queries. As you can see in Figure 11-8, each query is a connection-only query.

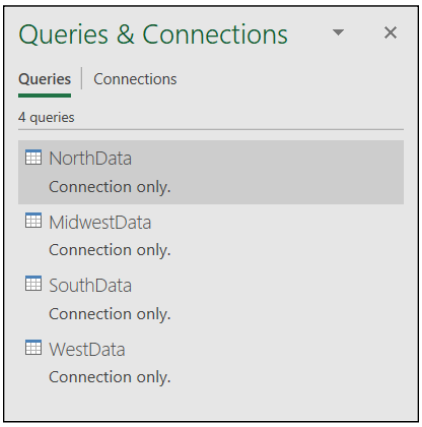


FIGURE 11-8: Create a connection-only query for each region.

Now that your data is in queries, you can start appending.

Appending the data

Follow these steps to append data from all other queries to the NorthData query:

1. **In the Queries & Connections pane, right-click the NorthData query and select Append.**

   The Append dialog box, shown in Figure 11-9, appears.

2. **Choose the Three or More Tables option at the top of the dialog box.**

   The Append dialog box reconfigures to show two list boxes. The Available Tables list (on the left) includes all the existing queries in your workbook. The Tables to Append list (on the right) contains the query to which you're currently appending data (the NorthData query in this scenario).

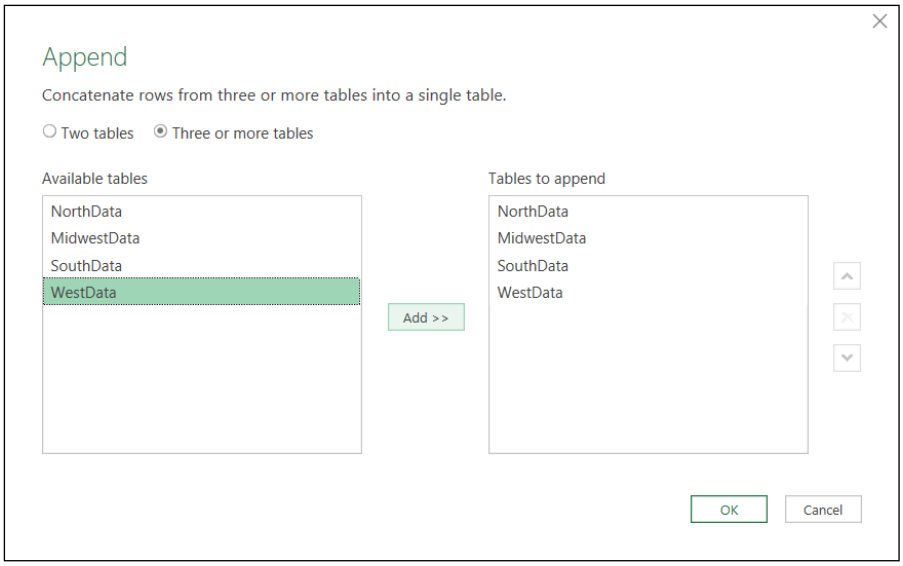


FIGURE 11-9: Appending multiple queries to NorthData.

3. **Select any query you want appended from the Available Tables list on the left and add it to the Tables to Append list box on the right.**
4. **Click OK to confirm your selections.**

   The Power Query Editor launches, giving you the opportunity to review and edit the results. You'll notice Power Query creates a new query called Append1.
5. **Rename the query in the Query Settings pane by typing** ConsolidatedView **in the Name box.**
6. **Click the Close & Load button to save and exit the Power Query Editor.**

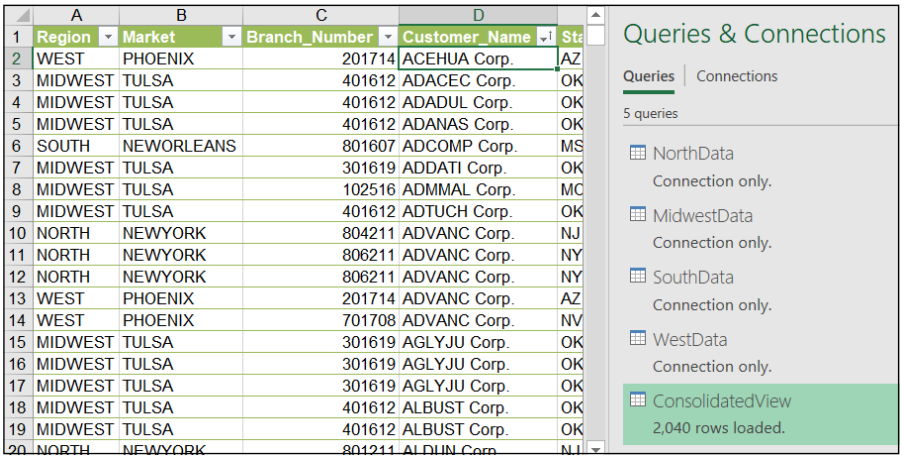
Figure 11-10 illustrates the final output. You've successfully created a consolidated table of region data.

WARNING

Note in Figure 11-9 that the NorthData query is on both the Available Tables list on the left and the Tables to Append list on the right. Be careful not to move the NorthData query to the right list box by mistake. If you do, you'll append the query to itself, effectively duplicating all the records within the query. Unless you have some strange requirement where creating exact copies of records is beneficial, you will want to avoid appending the current query to itself.

REMEMBER

As you append each query, you may be tempted to scroll down to the bottom of the data to see the newly added records. Unfortunately, the data preview in the Query Editor shows only a truncated sample set of records. Even if you scroll to the bottom of the preview, you're unlikely to see the appended data.



	A	B	C	D	
1	Region	Market	Branch_Number	Customer_Name	Sta
2	WEST	PHOENIX	201714	ACEHUA Corp.	AZ
3	MIDWEST	TULSA	401612	ADACEC Corp.	OK
4	MIDWEST	TULSA	401612	ADADUL Corp.	OK
5	MIDWEST	TULSA	401612	ADANAS Corp.	OK
6	SOUTH	NEWORLEANS	801607	ADCOMP Corp.	MS
7	MIDWEST	TULSA	301619	ADDATI Corp.	OK
8	MIDWEST	TULSA	102516	ADMMAL Corp.	MO
9	MIDWEST	TULSA	401612	ADTUCH Corp.	OK
10	NORTH	NEWYORK	804211	ADVANC Corp.	NJ
11	NORTH	NEWYORK	806211	ADVANC Corp.	NY
12	NORTH	NEWYORK	806211	ADVANC Corp.	NY
13	WEST	PHOENIX	201714	ADVANC Corp.	AZ
14	WEST	PHOENIX	701708	ADVANC Corp.	NV
15	MIDWEST	TULSA	301619	AGLYJU Corp.	OK
16	MIDWEST	TULSA	301619	AGLYJU Corp.	OK
17	MIDWEST	TULSA	301619	AGLYJU Corp.	OK
18	MIDWEST	TULSA	401612	ALBUST Corp.	OK
19	MIDWEST	TULSA	401612	ALBUST Corp.	OK
20	NORTH	NEWYORK	801211	ALDUN Corp.	NJ

FIGURE 11-10: The final output.

BEWARE OF MISMATCHED COLUMN LABELS

When you append one query to another, Power Query first scans the column labels for both queries to capture all column names. It then outputs all distinct column names and consolidates the data from both queries into the appropriate columns. It uses the column labels as a guide to knowing which data should be placed in which column.

If the column labels in your queries don't match, Power Query consolidates data for any match column, leaving null values in any columns that don't match.

Imagine that you have one query with the column labels Region and Revenue, and another query with the column labels Region and SalesAmount. Appending these two records yields a final table with all three columns: Region, Revenue, and SalesAmount. The records from the first query are entered into the Region and Revenue fields. The records from the second query are entered into the Region and SalesAmount fields, essentially leaving gaps in the Revenue and SalesAmount fields.

The bottom line is to make sure the column labels in your queries are identical before appending. As long as the column labels in each query are identical, Power Query can append the data correctly. Even if the columns in each query are positioned in a different sequence, Power Query can use the column labels to get all the data into the correct columns.

Understanding the Merge Feature

In your data adventures, you often find the need to build queries that join the data between two tables. For example, you may want to join an employee table to a transaction table to create a view that contains both transaction details and information on the employees who logged those transactions.

In this section, I describe how you can leverage the Merge feature in Power Query to join data from multiple queries.

Understanding Power Query joins

Similar to VLOOKUP or XLOOKUP in Excel, the Merge feature joins the records from one query to the records in another by matching on a unique identifier. An example of a unique identifier is Customer ID or Invoice Number.

You can join two data sets in one of several ways. The kind of join you apply is important because it determines which records are returned from each data set.

Power Query supports six kinds of joins, as described in the following list and shown in Figure 11-11:

- **Left Outer:** Tells Power Query to return all records from the first query, regardless of matching, *and* only those records from the second query that have matching values in the joined field
- **Right Outer:** Tells Power Query to return all records from the second query, regardless of matching, *and* only those records from the first query that have matching values in the joined field
- **Full Outer:** Tells Power Query to return all records from both queries, regardless of matching
- **Inner:** Tells Power Query to return only those records from both queries that have matching values
- **Left Anti:** Tells Power Query to return only those records from the first query that don't match any of the records from the second query
- **Right Anti:** Tells Power Query to return only those records from the first query that don't match any of the records from the second query

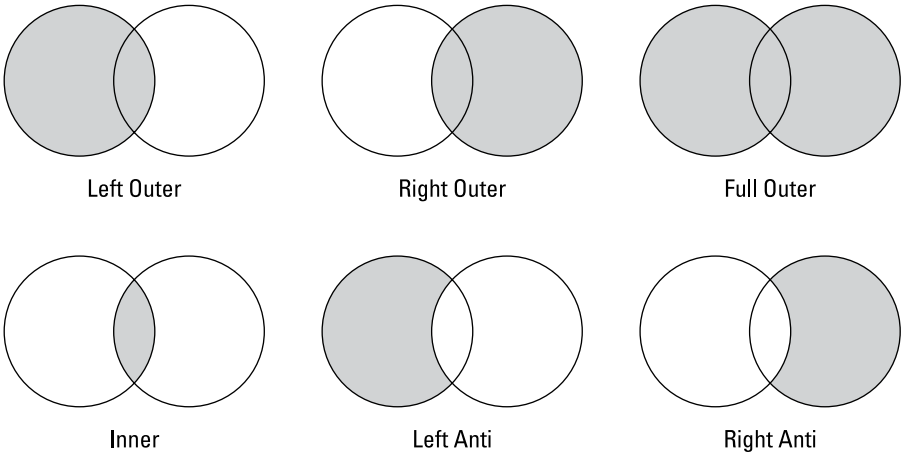


FIGURE 11-11: The kinds of joins supported by Power Query.

Merging queries

ON THE WEB

To better understand the Merge feature, I'll walk you through an exercise that merges interview questions and answers. In this walk-through, I use the predefined queries found in the `Merging_Data.xlsx` sample file available online at `www.dummies.com/go/excelpowerpivotpowerqueryfd2e`.

As you can see in Figure 11-12, two existing queries are in the Queries & Connections pane: Questions and Answers. These queries represent the questions and answers from the interview. The goal is to merge these two queries to create a new table showing questions and answers side-by-side.

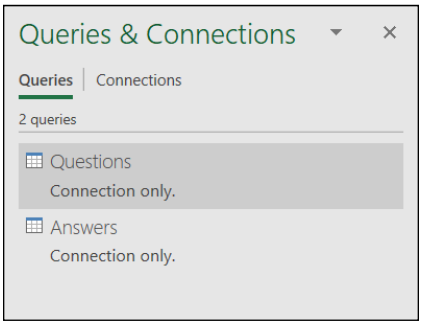


FIGURE 11-12: You need to merge the Questions and Answers queries into one table.

REMEMBER

The Merge feature can be used only with existing queries. That is to say, no matter what kind of data sources you have, you need to import them into Power Query before you can use them in a merge.

Follow these steps to perform the merge:

1. **Choose Data ➪ Get Data ➪ Combine Queries ➪ Merge (see Figure 11-13).**

   This step opens the Merge dialog box.

   In this dialog box, you use the drop-down boxes to select the queries you want to merge and then choose the columns that defined the unique identifier for each record. In this case, the InterviewID and QuestionID/AnswerID fields make up the unique identifier for each record.

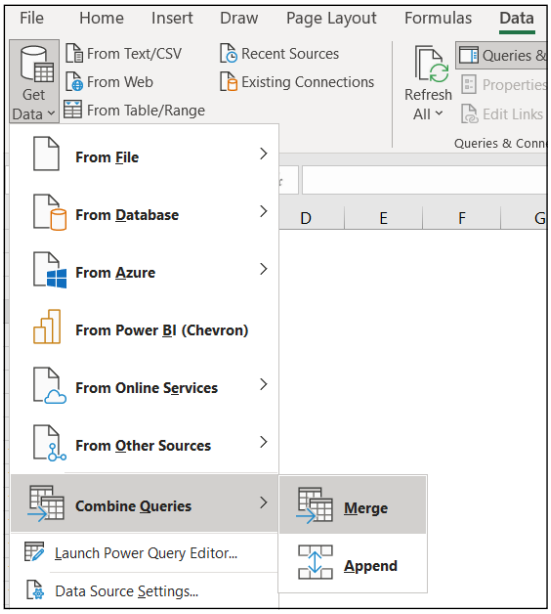


FIGURE 11-13: Activating the Merge dialog box.

2. **Select the Questions query in the top drop-down box.**
3. **Hold down the Ctrl key on the keyboard, and then click InterviewID and QuestionID — in that order.**
4. **Select the Answers query in the lower drop-down box.**
5. **Hold down the Ctrl key on the keyboard, and then click InterviewID and AnswerID — in that order.**
6. **Use the Join Kind drop-down box to select the kind of join you want Power Query to use. In this case, the default, Left Outer, works.**

7. **Click the OK button to finalize and open the Query Editor.**

Figure 11-14 illustrates the completed Merge dialog box. Note the small numbers 1 and 2 in the InterviewID and QuestionID fields. These small numbers are assigned based on the order in which you selected them (refer to Steps 3 and 5).

The order in which you selected the unique identifiers in each query matters. The two columns tagged with the small number 1 will be joined regardless of column labels. The two columns tagged with the small number 2 will also be joined.

TIP

At the bottom of the Merge dialog box, Power Query shows you how many records from the lower query match the top query, based on the unique identifiers you selected. Taking a look at Figure 11-14 again, you'll see about 17,600 answer records match the 26,910 question records. You don't need a 100 percent match for the merge to be valid. There might be a good reason that the records in the two queries don't all match up. In this case, not all questions were answered in all interviews, so the Answers query has fewer records.

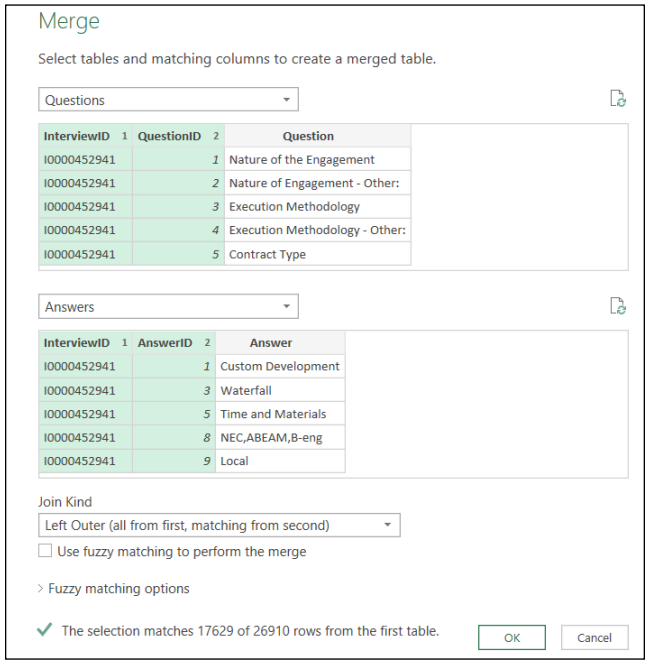


FIGURE 11-14: The Completed Merge dialog box.

8. **With the new merged query open in the Query Editor, click the Expand icon in the NewColumn field and choose the fields you want included in the final output (as shown in Figure 11-15). In this case, just choose the Answer field.**

   At this point, you can apply more transformations, if needed.

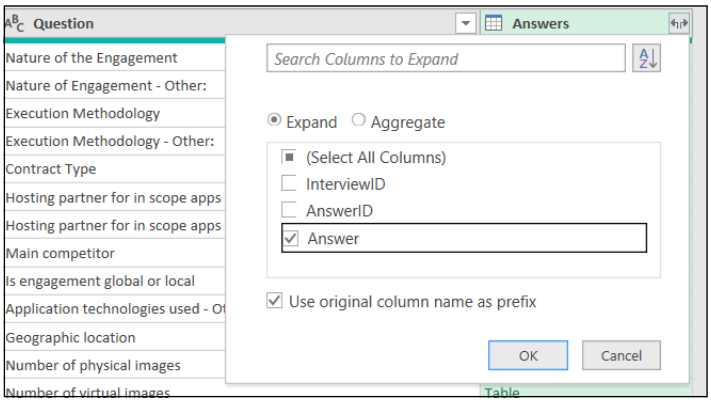


FIGURE 11-15: Expand the NewColumn field and choose the merged fields you want to output.

9. **When you're happy with the way things look, click the Close & Load command to output the results to the workbook.**

   Figure 11-16 shows the final merged query.

	A	B	C	D
1	InterviewID	QuestionID	Question	Answers.Answer
2	I0000452941	1	Nature of the Engagement	Custom Development
3	I0000452941	2	Nature of Engagement - Other:	
4	I0000452941	3	Execution Methodology	Waterfall
5	I0000452941	5	Contract Type	Time and Materials
6	I0000452941	4	Execution Methodology - Other:	
7	I0000452941	8	Main competitor	NEC,ABEAM,B-eng
8	I0000452941	9	Is engagement global or local	Local
9	I0000452941	6	Hosting partner for in scope apps	
10	I0000452941	10	Application technologies used - Other:	Java
11	I0000452941	7	Hosting partner for in scope apps - Other:	
12	I0000453446	1	Geographic location	Client data center
13	I0000453446	2	Number of physical images	100
14	I0000453446	3	Number of virtual images	300

FIGURE 11-16: The final table with merged questions and answers.

If you need to adjust or correct a merged query, right-click the query in the Queries & Connections pane and select Edit. Once the Power Query Editor opens, right-click the Source query step and select Edit Settings (see Figure 11-17). Alternatively, you can simply click the gear icon next to the Source query step. This action opens the Merge dialog box, where the necessary changes can be applied.

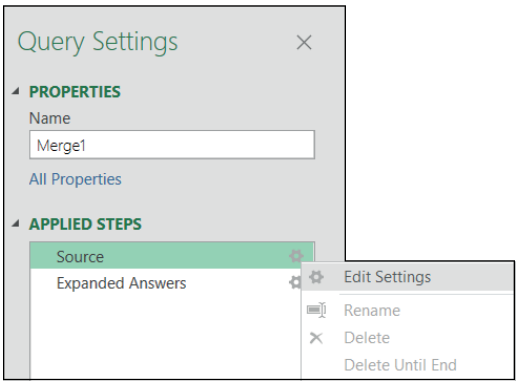


FIGURE 11-17: Right-click the Source query step and select Edit Settings to reactivate the Merge dialog box.

Understanding Fuzzy Match

ON THE WEB

In some cases, the tables you need to merge won't have any unique identifiers that match exactly. In these situations, you can leverage Power Query's Fuzzy Match feature. In this walk-through, you'll use the predefined Revenue and Employee queries found in the `FuzzyMatch.xlsx` sample file.

Follow these steps to perform the merge:

1. **Click Data ➪ Get Data ➪ Combine Queries ➪ Merge.**
2. **Select the Employee query in the top drop-down box, and then click the Last_Name column.**
3. **Select the Revenue query in the lower drop-down box and click the Employee column.**
4. **Use the Join Kind drop-down box to select the kind of join that you want Power Query to use.**

   In this case, the default Left Outer works.
5. **Select the Use Fuzzy Matching to Perform the Merge check box.**

6. **Click the arrow next to Fuzzy Matching Options to reveal the available configuration options.**

The Merge dialog box should look like the one shown in Figure 11-18. Here are your options:

- **Similarity Threshold:** The Similarity Threshold tells Power Query how similar two values need to be to match. The default value (applied if left blank) is .80, which roughly translates to an 80 percent similarity. Entering a value of 1 in the Similarity Threshold means you need the two values to match 100 percent to qualify as a match. The minimum value you can enter here is 0, but this causes all values to match each other. The number you place here really depends on the data you're using. In most cases, the default of .80 is the safest bet, because it will match a decent number of records without resulting in too many false matches.
- **Ignore Case:** The Ignore Case check box specifies what role character case (uppercase, lowercase, and so on) plays in the matching. The default behavior is case insensitive, which means case is ignored when matching values. If you need to match values taking character case into account, deselect the Ignore Case check box.

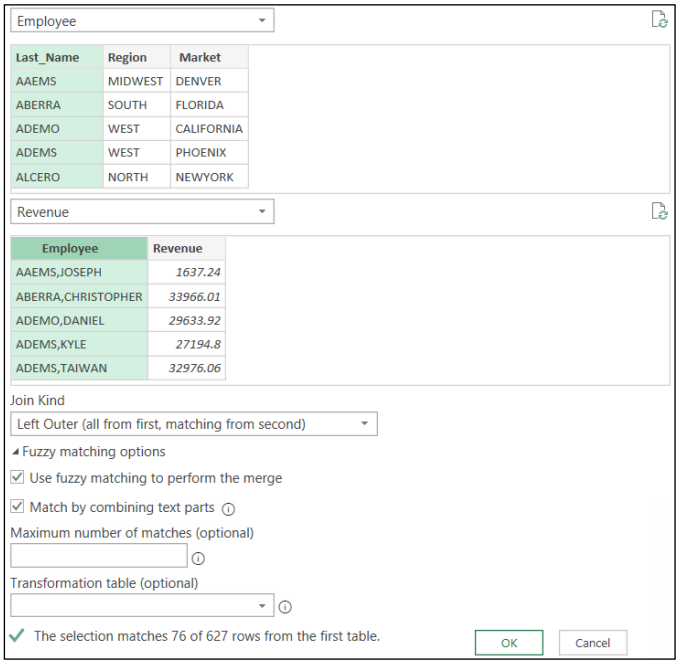


FIGURE 11-18: The Merge dialog box with Fuzzy Matching selected.

- **Match by Combining Text Parts:** You can tell Power Query to try to combine parts of text in each record to complete a match. For instance, if one of your tables contains the value "star slight," while the other table contains the value "starlight," Power Query will try to match "starlight" with "stars light." In other words, in addition to its normal matching algorithm, it will combine text and try the match again.
- **Maximum Number of Matches:** This option defines the maximum number of matching rows that will be returned for each record. For example, if you only want to find one matching row for each record, you'll specify a value of 1. The default behavior is to return all matches.
- **Transformation Table:** In some cases, you may already have a mapping table containing two values (in separate columns) which you've determined should match automatically. You can import your mapping table in a separate query, and then point to it in the Transformation table option.

7. **Enter** .60 **as the Similarity threshold.**

   Because you're dealing with names that are fairly unique, you can risk a lower similarity percent than the default.

8. **Enter** 1 **as the Maximum Number of Matches.**

   This will ensure that you only get one match per employee.

9. **Click the OK button to perform the merge and launch the Power Query Editor.**

10. Now you can click the Expand icon next to the Revenue column and choose the columns you want to be included in the final output. In this case, all you need is the Revenue column.

11. **Click the Close & Load command to output the results to the workbook.**

TIP

At the bottom of the Merge dialog box, Power Query shows you how many records from the lower query match the top query based on the unique identifiers that you selected. When using the fuzzy match feature, it's often useful to try different similarity thresholds to get more matches. You'll find that trial and error will often be necessary to find the right balance of matching as many records as possible without including too many false matches.

IN THIS CHAPTER

- Making a custom function
- Using custom functions in other queries
- Creating a parameter query

Chapter 12

Extending Power Query with Custom Functions

Power Query records all actions using its own formula language (known as the *M* language). When you connect to a data source and apply transformations to that data, Power Query diligently saves your actions as M code behind the scenes in query steps. The transformation steps can then be repeated when you refresh the data in your query.

That backstage coding is relatively transparent, and can, for the most part, be ignored for most data processing activities. In this chapter, I show you how to leverage the M language to extend the capabilities of Power Query to create your own custom functions and perform truly heroic data processing.

Creating and Using a Basic Custom Function

When building a custom function for Power Query, you're essentially doing nothing more than creating a query and manipulating its M code to return a desired result. That result can be an array, a data table, or a single value.

To help you gain a sense of the general steps taken to create a custom function, I show you now how to build a basic mathematical function that calculates profit. This function should be able to take a revenue amount and a cost amount and output a profit amount using this basic mathematical operation:

Revenue – Cost = Profit

For basic functions such as this one, you can start with a blank query and simply enter the needed M code from scratch. Follow these steps:

1. **Click the Data tab in Excel and select Get Data ➪ From Other Sources ➪ Blank Query.**

   This step activates the Query Editor window.

2. **On the Query Editor Ribbon, click on the View tab and select the Advanced Editor command.**

3. **When the Advanced Editor window opens, delete the starter syntax you see in the code input box.**

4. **Enter the following code into the code input box:**

```
let Profit = (Revenue, Cost)=>
Revenue–Cost
in Profit
```

   - Line 1 of the code tells Power Query that this is a function called Profit, requiring two parameters. For clarity, the two parameters are named Revenue and Cost, though Power Query doesn't care what you name them as long as the names start with a letter and have no spaces.
   - Line 2 in the code essentially tells Power Query to subtract the Cost parameter from the Revenue parameter.
   - Line 3 of the code tells Power Query to return the result.

   Figure 12-1 illustrates what the code looks like in the Advanced Editor window.

5. **Click the Done button to close the Advanced Editor window.**

6. **In the Query Settings pane, change the name of the query in the Name input box (see Figure 12-2) to** FunctionProfit**.**

   The goal here is to give your function a reasonably descriptive name, as opposed to Query1.

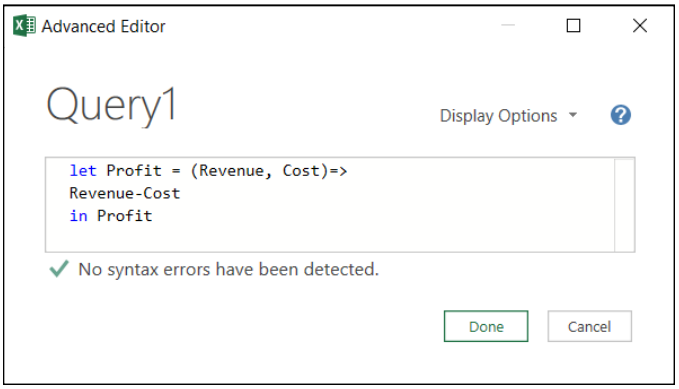


FIGURE 12-1: Enter your custom code in the Advanced Editor window.

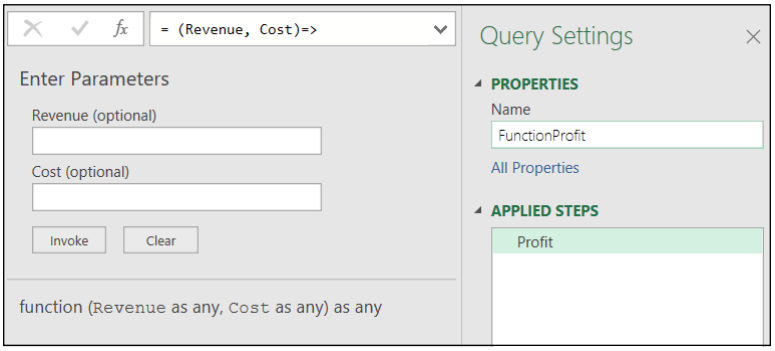


FIGURE 12-2: Give your custom function a friendly name.

7. At this point, you can select the Home tab of the Query Editor and click the Close & Load button.

Power Query adds the query to the Queries & Connections pane as a connection-only query. Queries recognized as functions are automatically saved as connection-only. You can now use your function in other queries that contain revenue and cost fields. For example, Figure 12-3 illustrates the contents of the `Chapter12_Sample.txt` text file, which you can find in the download files for this book.

This text file contains a table of invoices with the fields Qty, UnitCost, and UnitPrice. Your newly created function can be used to calculate profit using these fields.

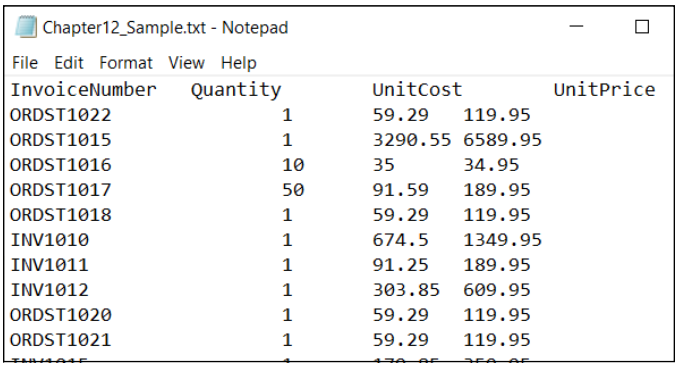


FIGURE 12-3: A text file containing Invoice details.

To create a new query from this text file, follow these steps:

1. **Click the Data tab in Excel and select Get Data ➪ From File ➪ From Text/CSV.**

   This step opens the Import Data dialog box.

2. **Browse for, and select, the `Chapter12_Sample.txt` file.**

   Power Query opens a preview of the data within the text file.

3. **Click the Transform Data button to activate the Power Query Editor.**

4. **While in the Query Editor, click the Add Column tab and then click the Custom Column button.**

   The Custom Column dialog box opens. Here, you can call the custom function and pass it the needed parameters.

5. **In this case, enter the following line:**

   ```
   = FunctionProfit([UnitPrice], [UnitCost])*[Quantity]
   ```

   This syntax calls the FunctionProfit custom function and passes the UnitPrice and UnitCost fields as the required parameters. The results are then multiplied by the Quantity field. The Custom Column dialog box should look similar to the one shown in Figure 12-4.

6. **Click the OK button to apply the custom column.**

   When you confirm the changes, Power Query triggers the function and calculates profit for each row in the data table (see Figure 12-5).

Although this example is quite basic, it demonstrates that you can define a function that requires parameters and then use the function in other queries. This simple technique is the foundation for creating more useful functions.

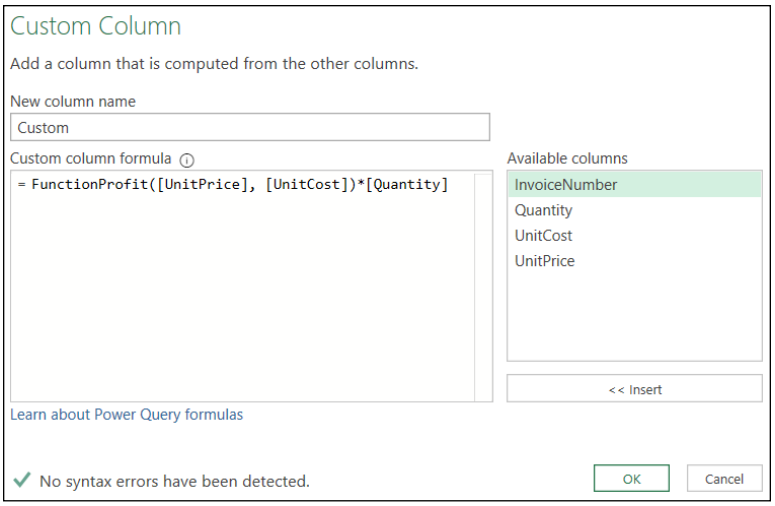


FIGURE 12-4: Use the Custom Column action to invoke your function.

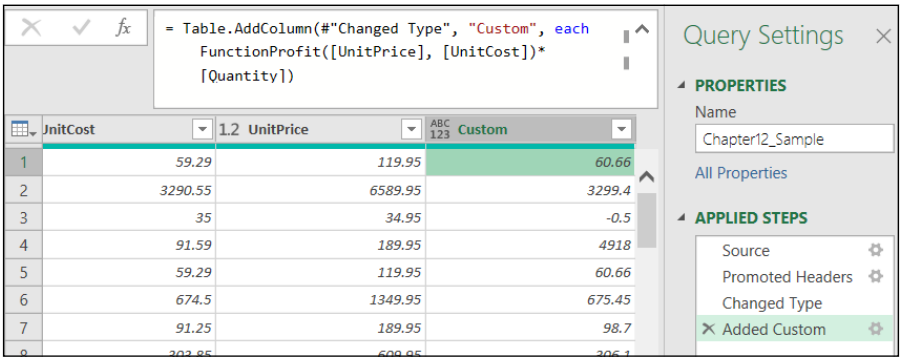


FIGURE 12-5: The custom column showing the results of the function for each row in the table.

REMEMBER

Power Query functions are stored in the workbook in which they reside. Unfortunately, there's no easy way to share functions between workbooks. If you start a new workbook, you need to re-create your functions in that new workbook.

Creating a Function to Merge Data from Multiple Excel Files

When building a basic function, such as the profit function you create in the earlier section "Creating and Using a Basic Custom Function," it's no big deal to start from a blank query and enter all the code from scratch. But for more complex functions, it's generally smarter to build a starter query via Query Editor and then manipulate the M code to accomplish what you need.

Imagine that you have a set of Excel files in a folder (see Figure 12-6). These files all contain a worksheet named MySheet that holds a table of data. The tables in each file have the same structure, but need to be combined into one file. This is a common task/nightmare that most Excel analysts have faced at one time or another. If you don't have a solid knowledge of Excel VBA programming, this task typically entails opening each file, copying the data on the MySheet tab, and then pasting the data into a single workbook.

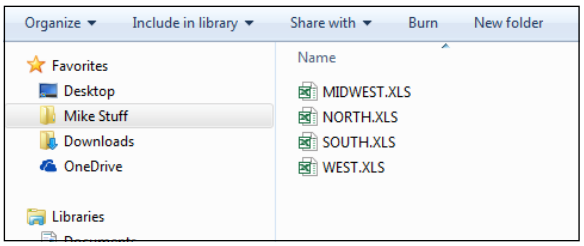


FIGURE 12-6: You need to merge into one table the data in all the Excel files in this folder.

Power Query has the ability to make short work of this task, but it requires a bit of direction via a custom function. Now, it would be difficult for most anyone to start from a blank query and type out the M code for the relatively complex function needed for this endeavor. Instead, you could build a starter query via Query Editor and then wrap the query in a function.

To help you understand this concept, I present the following steps:

1. **On the Excel Data tab, select Get Data ➪ From File ➪ From Workbook.**
2. **Browse to the folder that contains all the Excel files, and choose only one of them.**
3. **In the Navigator pane (shown in Figure 12-7), choose the sheet that holds the data that needs to be consolidated.**

   In this case, select MySheet then click the Transform Data button to open the Query Editor.
4. **Use the Query Editor to apply a few basic transformation actions to the data.**

   As an example, the Applied Steps shown in Figure 12-8 shows some basic transformation steps. For the purposes of this exercise, the specific steps you choose to apply aren't important.

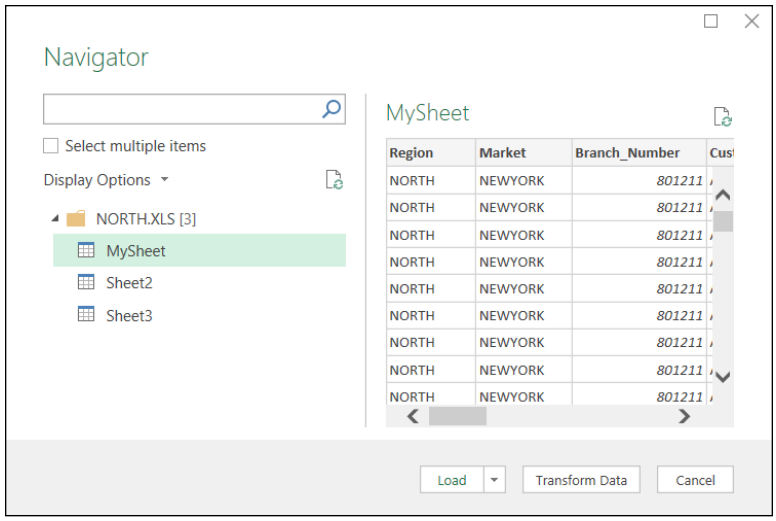


FIGURE 12-7: Connect to one of the Excel files in the target folder, and navigate to the sheet holding the data that needs to be consolidated.

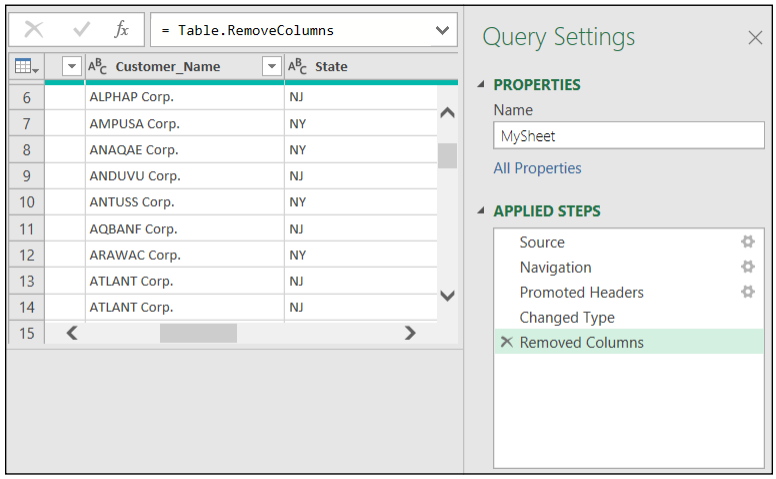


FIGURE 12-8: Use the Query Editor to apply any necessary transformation actions.

5. Open Advanced Editor window by clicking the View tab and selecting the Advanced Editor command.

Figure 12-9 demonstrates that as you build out the starter template, Power Query diligently creates the bulk of the code for your function. Note in the portion of the code that's highlighted in gray (for illustration), Power Query has hard-coded the file path and filename of the Excel file that was originally selected. The idea is to wrap this starter code in a function that passes a dynamic file path and filename.

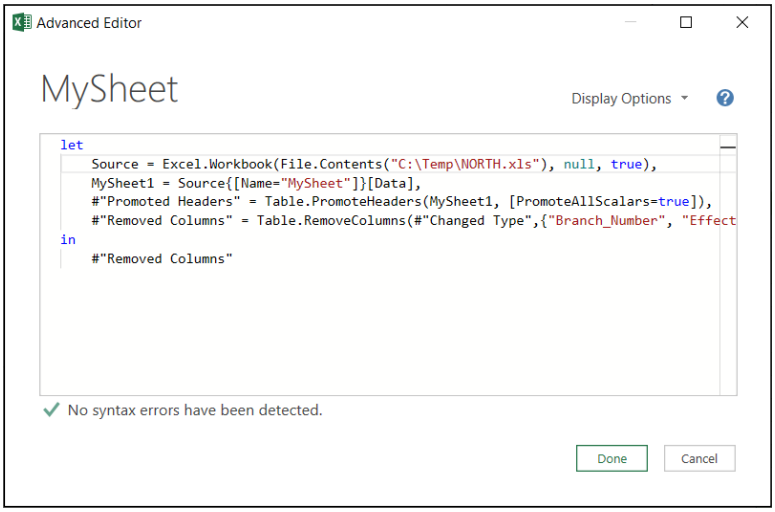


FIGURE 12-9: Open the Advanced Editor to see the starter code.

6. **Wrap the entire block of code with function tags, specifying that this function requires two parameters: FilePath and FileName. Also replace the hard-coded file path and filename with each respective parameter.**

   Here's the syntax shown in Figure 12-10:

```
let GetMyFiles=(FilePath, FileName) =>
let
Source = Excel.Workbook(File.Contents(FilePath&FileName),
  null, true),
MySheet1 = Source{[Name="MySheet"]}[Data],
#"Promoted Headers" = Table.PromoteHeaders(MySheet1,
  [PromoteAllScalars=true]),
#"Removed Columns" =
Table.RemoveColumns(#"Promoted Headers",{"Branch_Number",
  "Effective_Date"})
in
#"Removed Columns"
in GetMyFiles
```

7. **Click Done to close the Advanced Editor.**

   As you can see in Figure 12-11, the Query Editor window contains two input boxes for the two parameters we've defined (FilePath and FileName). This query is officially a function that accepts two parameters.

8. **In the Query Settings pane, change the name of the query in the Name input box. Give the function a reasonably descriptive name, such as (in this scenario) fnGetMyFiles.**

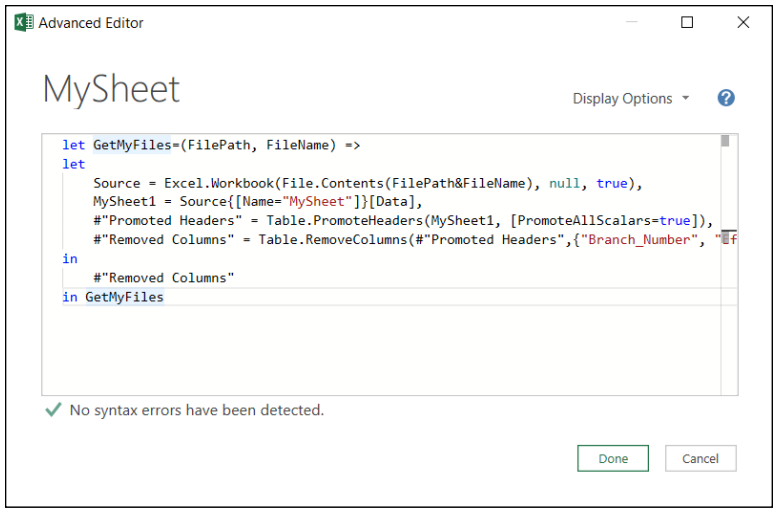


FIGURE 12-10: Wrapping the starter code with function tags and replacing the hard-coded names with your dynamic parameters.

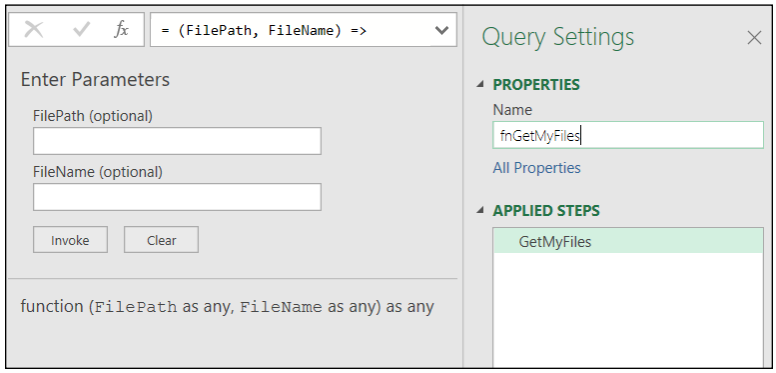


FIGURE 12-11: The Query Editor after defining parameters.

9. Click the Home tab of the Query Editor and click the Close & Load button.

At this point, the custom function is ready to be used on all files in the target folder.

10. Click the Data tab in Excel and select Get Data ➪ From File ➪ From Folder, browse to the folder that contains all the Excel files, and click the Open button.

A new window appears to show you a table displaying all the files in the chosen folder.

11. Click the Transform Data button.

The Query Editor window activates to display a table containing a record for each file in the chosen folder (see Figure 12-12). Each row contains attributes for each of the files listed. The columns you're interested in are Folder Path and Name, which provide the function with the needed FilePath and FileName parameters.

	Content	Folder Path	Name	Extension
1	Binary	C:\Temp\	MIDWEST.XLS	.XLS
2	Binary	C:\Temp\	NORTH.XLS	.XLS
3	Binary	C:\Temp\	SOUTH.XLS	.XLS
4	Binary	C:\Temp\	WEST.XLS	.XLS

FIGURE 12-12: Create a new query using the From Folder connection type to retrieve a table of all files in the target folder.

12. Click the Add Column tab, and then click the Custom Column command.

The Custom Column dialog box opens.

13. Invoke the function and pass the Folder Path and Name fields as parameters separated by commas (see Figure 12-13).

When you confirm your changes, Power Query triggers the function for each row in the data table. The function itself grabs the data from each file and returns a table array. Figure 12-14 shows the newly created custom column with the Expand icon next to the column name.

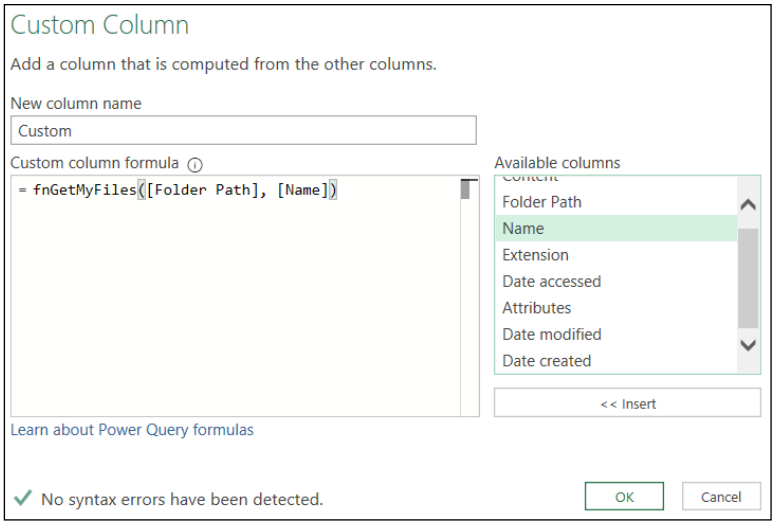


FIGURE 12-13: Use the Custom Column action to invoke the function.

	Date modified	Date created	Custom
1	9/7/2021 11:11:23 AM	9/8/2021 7:32:15 AM	Table
2	9/7/2021 11:11:23 AM	9/8/2021 7:32:15 AM	Table
3	9/7/2021 11:11:23 AM	9/8/2021 7:32:15 AM	Table
4	9/7/2021 11:11:23 AM	9/8/2021 7:32:15 AM	Table

FIGURE 12-14: Power Query triggers the function and returns a table array for each file in the folder.

14. Click the Expand icon for your new custom column.

You see a list of fields included in each table array, as shown in Figure 12-15.

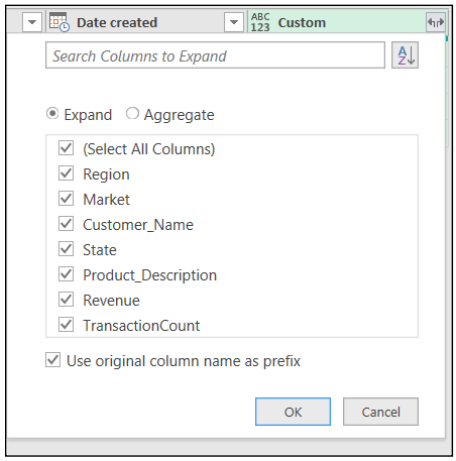


FIGURE 12-15: Click the Custom column header to expand the table arrays.

15. Choose which fields in the table array to show, select the Expand radio button, and then click the OK button.

With each table array expanded, Power Query exposes the columns pulled from each Excel file and adds the detailed records to the data preview. Figure 12-16 illustrates the data preview for the final combined table.

16. At this point, you can remove unneeded columns and then click the Close & Load command to output the combined table.

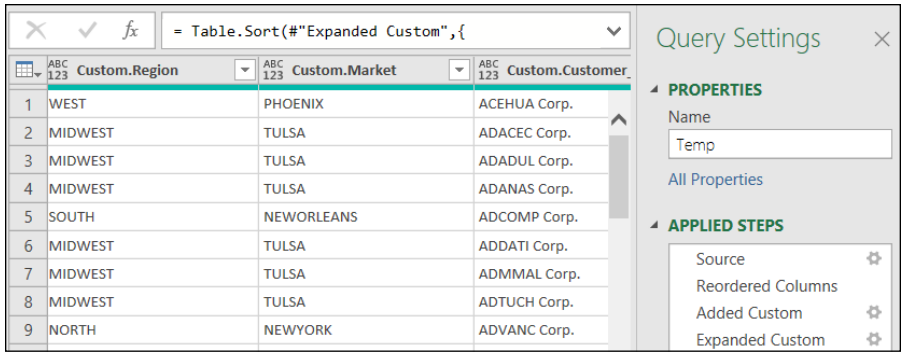


FIGURE 12-16: Power Query exposes the columns pulled from each Excel file and adds the detailed records to create the final combined view.

As you look at the final combined view, don't lose track of the fact that this relatively complex task was facilitated by a simple custom function. For all the steps required to accomplish this task, you expend very little effort on creating the code for the function. Power Query writes the code for the core functionality, and you simply wrap that code into a function.

The takeaway here is that you don't have to be an expert on Power Query's M language to pull together effective and useful custom functions. You can leverage the Query Editor to create some base code and then adjust from there.

Creating Parameter Queries

A parameter query is a kind of query that relies on one or more parameters to run. Although that sounds suspiciously like the custom functions covered earlier in this chapter (after all, they ran on parameters), there is a subtle difference.

A *parameter* query is one where *you* provide the parameters. So rather than have the parameters come from a predefined query, you enter the parameters. This comes in handy when creating interactive reporting for others to consume.

In this section, I walk you through creating your first parameter query.

Preparing for a parameter query

To create a proper parameter query, you first have to understand the parameters necessary to make your reporting interactive. The best way to gain this understanding is to explore the target data source.

In this scenario, I tell you how to build an interactive view of the top-grossing films for any given year and month. To accomplish this task, leverage the Box Office Mojo website. Box Office Mojo provides an array of box office reporting tools, including a monthly index of top-grossing films.

The URL for the monthly index includes a month parameter and a year parameter. Enter this URL into any browser and you see a list of the top-grossing films of January 2020:

```
www.boxofficemojo.com/month/january/2020/
```

A look at the website (shown in Figure 12-17) confirms that the URL opens a web page that contains the table you would expect to see: an index of movies for January 2020 box office. The parameters in the URL are working as expected.

Now that you know the year and month number are the parameters, you can get started.

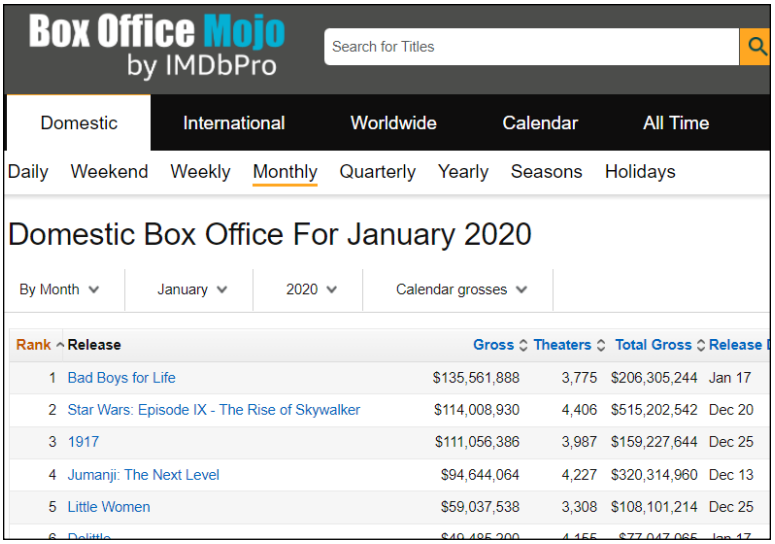


FIGURE 12-17: Confirming that the parameters in the URL actually work.

THE IMPORTANCE OF KNOWING YOUR PARAMETERS

You may notice that the Box Office Mojo web service only returns results when the name is all lowercase. Enter the following URL into your favorite web browser, and you get results:

```
www.boxofficemojo.com/month/may/2021
```

Entering the following URL, and you get nothing back:

```
www.boxofficemojo.com/month/May/2021
```

This is a good example of knowing what parameters the source database is expecting and in what form. In this case, the Box Office Mojo web service is clearly expecting lowercase month names. Now that you know that, you can make sure the month names you pass are lowercase.

Creating the base query

The best place to start is to create the base query. The *base* query is essentially the one that will pull the data you're working toward. In this scenario, you create a query that pulls the table shown in Figure 12-16 from the Box Office Mojo website.

Follow these steps:

1. **Open a new Excel workbook, and then select Data ➪ Get Data ➪ From Other Sources ➪ From Web.**
2. **Enter a starting URL and then click OK. You can use the following URL:**

   `www.boxofficemojo.com/month/january/2020`

3. **Use the Navigator pane to select the table containing the data you need (Table0 in this case), and then click the Transform Data button to open the Query Editor.**
4. **Use the Query Editor to apply any desired transformations.**

   Figure 12-18 illustrates a clean table that makes up the base query.

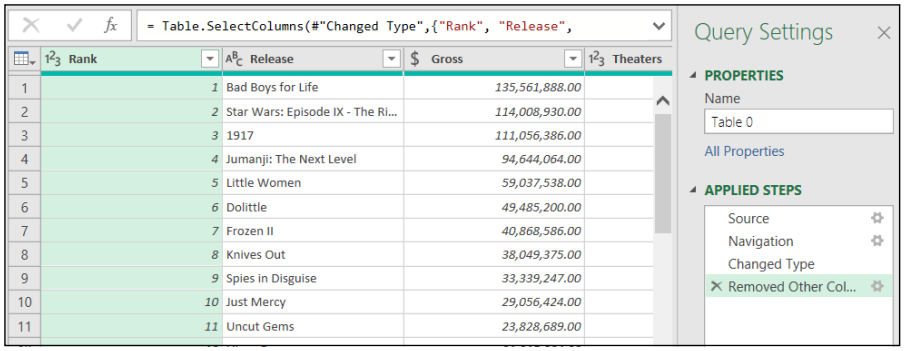


FIGURE 12-18: The clean base query.

5. **Open the Advanced Editor window by clicking the View tab and selecting the Advanced Editor command.**
6. **Wrap the entire block of code with function tags, specifying that this function requires two parameters: YearNum and MonthName. Also replace the hard-coded year and month in the URL with each respective parameter.**

   Here's the final syntax shown in Figure 12-19:

```
let TopMovies=(YearNum, MonthName) =>
let
```

```
Source = Web.Page(Web.Contents("www.boxofficemojo.com/
   month/" &
MonthName & "/" & Number.ToText(YearNum) & "/")),
Data0 = Source{0}[Data],
#"Removed Other Columns" = Table.
   SelectColumns(Data0,{"Rank", "Release", "Gross",
   "Theaters", "Total Gross", "Release Date",
   "Distributor"})
in
    #"Removed Other Columns"
in TopMovies
```

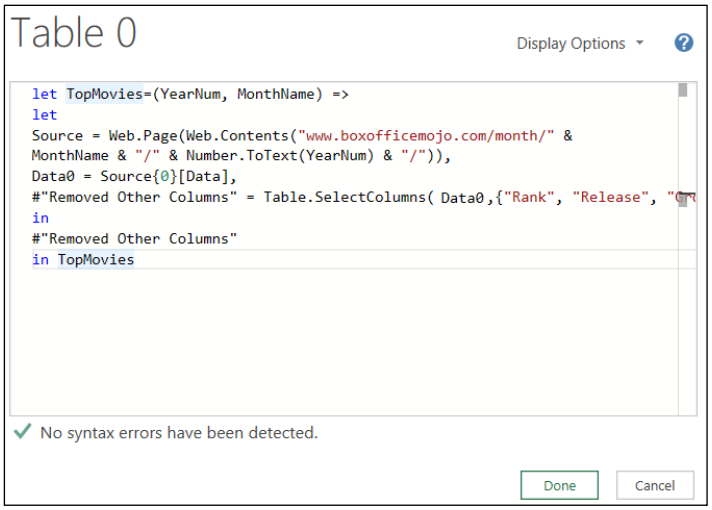


FIGURE 12-19: Wrapping the starter code with function tags and specifying a YearNum parameter and a MonthName parameter.

7. **Click Done to close the Advanced Editor.**
8. **In the Query Settings pane, change the name of the query in the Name input box. In this scenario, it's fnGetTopMovies.**
9. **Click the Home tab of the Query Editor and click the Close & Load button.**

You now have a fnGetTopMovies function, which can be used to pull web data from a custom function, and it's ready to be used on all files in the target folder.

Creating the parameter query

The final step is to create the parameter query. To do so, you need a simple table that will serve as the feeder for your dynamic parameters.

Staying in the same workbook where you created fnGetTopMovies, create a table similar to the one shown in Figure 12-20.

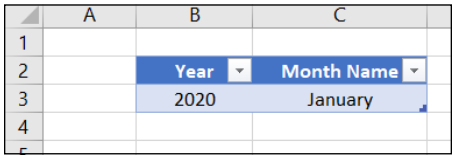


FIGURE 12-20: Create a simple parameter table.

From here, follow these steps:

1. **Place the cursor in the parameter table, and then select Data ➪ From Table/Range.**

   The Create Table dialog box opens.

2. **Click OK to continue.**

   The Query Editor opens with the parameter table.

3. **Click the Add Column tab, and then click the Custom Column command.**

4. **In the Custom Column dialog box, invoke the fnGetTopMovies function, passing the year and month fields as parameters (see Figure 12-21), and click OK.**

   Because you're mixing data from the web with data from Excel (though the parameter table can hardly be considered data), Power Query initiates a yellow security bar asking you for more information on data-privacy settings.

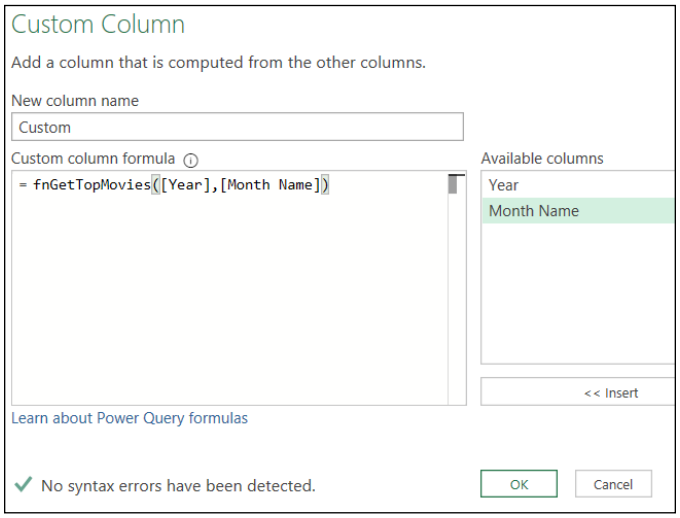


FIGURE 12-21: Use the Custom Column action to invoke the function.

5. **Click the Continue button inside the yellow security bar.**

   The Privacy Levels dialog box opens, as shown in Figure 12-22.

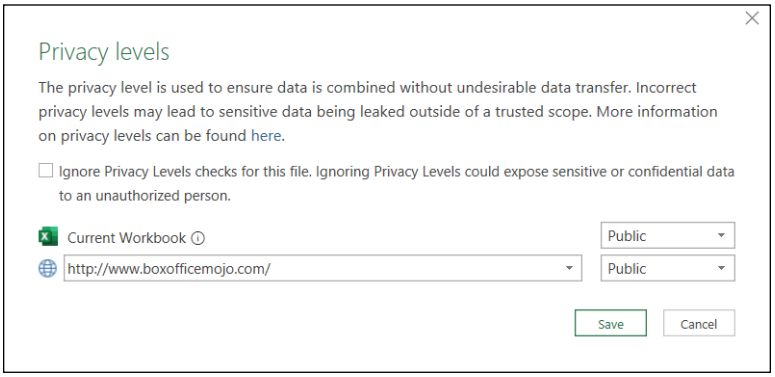


FIGURE 12-22: The combining of Excel and Web data triggers Power Query to ask about data privacy.

6. **Select Public for both the Current Workbook option and the website. Click the Save button to confirm and save the privacy levels.**

   Power Query, at this point, imports data from the website based on the year and month in the parameter table.

7. **The data imports as a table array, so click the green Table hyperlink.**

   Alternatively, you can click the Expand icon.

   Now that you're basically done, it's time to think about where the query should be loaded. If you simply click the Close & Load button, Power Query outputs the final parameter query in its own worksheet. However, it would be more practical to have the parameter table and query results on the same worksheet. This way, you can edit the parameters and see the results without having to flip between worksheets.

8. **Rather than click the Close & Load command button, click the drop-down arrow beneath the button and select the Close & Load To option.**

9. **In the Import Data dialog box, choose the Existing Worksheet option, ensuring that you select a cell beneath the parameter table. (See Figure 12-23.)**

10. **Click the OK button to finalize the query (see Figure 12-23).**

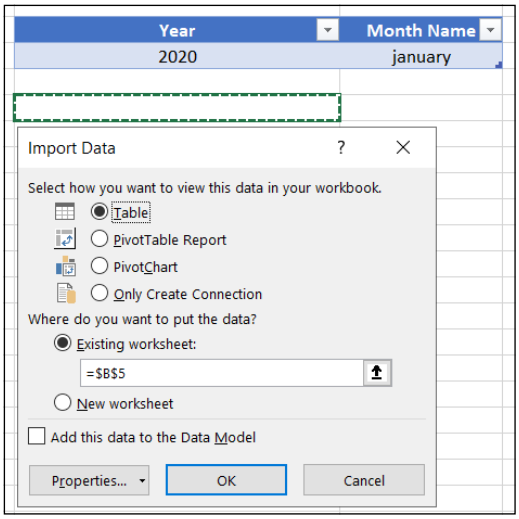


FIGURE 12-23: Choose to load the final query results under the parameters table.

Figure 12-24 illustrates the final parameter query. Take a moment to think about what's happening here. With this parameter query, you enter a year and a month and click Refresh (or press Ctrl+Alt+F5). Power Query then dynamically imports data back from the internet based on the parameters you entered — all without your having to enter more than three lines of M language syntax. Truly amazing.

	A	B	C	D	E	F
1						
2		Year	Month Name			
3		1994	july			
4						
5		Rank	Release	Gross	Theaters	Total Gross
6		1	The Lion King	145,511,104	2,624	312,855,561
7		2	Forrest Gump	134,489,249	2,365	329,694,499
8		3	True Lies	80,087,705	2,561	146,282,411
9		4	Speed	41,897,551	2,169	121,248,145
10		5	The Client	39,305,343	2,365	92,115,211
11		6	The Shadow	32,063,435	1,769	32,063,435
12		7	I Love Trouble	30,806,194	1,746	30,806,194
13		8	Blown Away	30,156,002	1,862	30,156,002
14		9	Angels in the Outfield	29,744,101	1,921	50,236,831
15		10	Wolf	23,100,104	2,117	65,002,597
16		11	Baby's Day Out	16,827,402	1,707	16,827,402

FIGURE 12-24: The final parameter query provides an interactive mechanism to flexibly pull data based on dynamic parameters, all with virtually no coding.

3 The Part of Tens

IN THIS PART . . .

Explore some best practices that can help you avoid Power Pivot performance issues.

Examine a few tips and tricks that can save you time when working with Power Query.

IN THIS CHAPTER

- Improving Power Pivot performance
- Best practices for avoiding lag
- Managing slicer performance
- Using views versus tables

Chapter 13

Ten Ways to Improve Power Pivot Performance

The word *performance* (as it relates to applications and reporting) is typically synonymous with speed — or how quickly an application performs certain actions such as opening within the browser, running queries, or filtering.

Because Power Pivot inherently paves the way for large amounts of data with fairly liberal restrictions, it isn't uncommon to produce reporting solutions that work but are unbearably slow. And nothing will turn your intended audience away from your slick new reports faster than painfully sluggish performance.

This chapter offers ten actions you can take to optimize the performance of your Power Pivot reports.

Limit the Number of Rows and Columns in Your Data Model Tables

One huge influence on Power Pivot performance is the number of columns you bring, or *import*, into the data model. Every column you import is one more dimension that Power Pivot has to process when loading a workbook. Don't import extra columns "just in case" — if you're not certain you will use certain columns, just don't bring them in. These columns are easy enough to add later if you find that you need them.

REMEMBER

More rows mean more data to load, more data to filter, and more data to calculate. Avoid selecting an entire table if you don't have to. Use a query or a view at the source database to filter for only the rows you need to import. After all, why import 400,000 rows of data when you can use a simple WHERE clause and import only 100,000?

Use Views Instead of Tables

Speaking of views, for best practice, use views whenever possible.

Though tables are more transparent than views — allowing you to see all the raw, unfiltered data — they come supplied with all available columns and rows, whether you need them or not. To keep your Power Pivot data model to a manageable size, you're often forced to take the extra step of explicitly filtering out the columns you don't need.

Views can not only provide cleaner, more user-friendly data but also help streamline your Power Pivot data model by limiting the amount of data you import.

Avoid Multi-Level Relationships

Both the number of relationships and the number of relationship layers have an impact on the performance of your Power Pivot reports. When building your model, follow best practice and have a single fact table containing primarily quantitative numerical data (facts) and dimension tables that relate to the facts directly. In the database world, this configuration is a *star schema*, as shown in Figure 13-1.

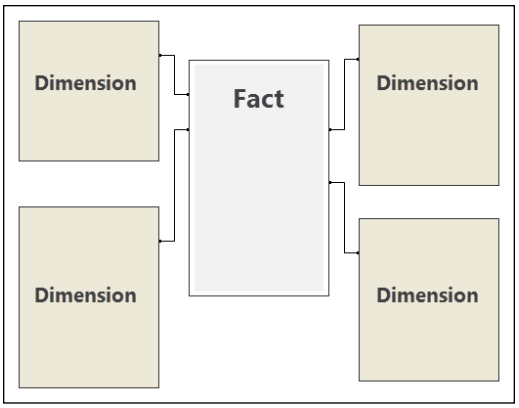


FIGURE 13-1: A star schema is the most efficient data model, with a single fact table and dimensions relating directly to it.

TIP

Avoid building models where dimension tables relate to other dimension tables. Figure 13-2 illustrates this configuration, also known as a *snowflake schema.* This configuration forces Power Pivot to perform relationship lookups across several dimension levels, which can be particularly inefficient, depending on the volume of data in the model.

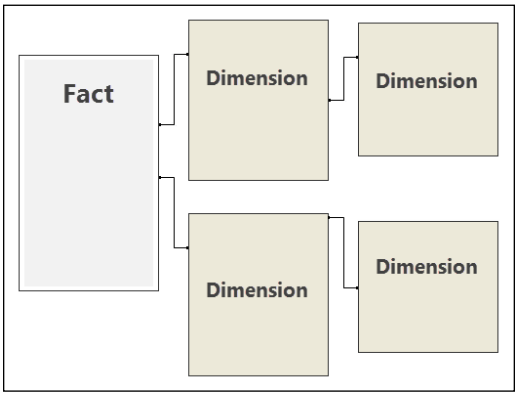


FIGURE 13-2: Snowflake schemas are less efficient, causing Power Pivot to perform chain lookups.

Let the Back-End Database Servers Do the Crunching

Most Excel analysts who are new to Power Pivot tend to pull raw data directly from the tables on their external database servers. After the raw data is in Power Pivot, they build calculated columns and measures to transform and aggregate the data as needed. For example, users commonly pull revenue and cost data and then create a calculated column in Power Pivot to compute profit.

So why make Power Pivot do this calculation when the back-end server could have handled it? The reality is that back-end database systems such as SQL Server have the ability to shape, aggregate, clean, and transform data much more efficiently than Power Pivot. Why not utilize their powerful capabilities to massage and shape data before importing it into Power Pivot?

Rather than pull raw table data, consider leveraging queries, views, and stored procedures to perform as much of the data aggregation and crunching work as possible. This leveraging reduces the amount of processing that Power Pivot will have to do and naturally improves performance.

Beware of Columns with Many Unique Values

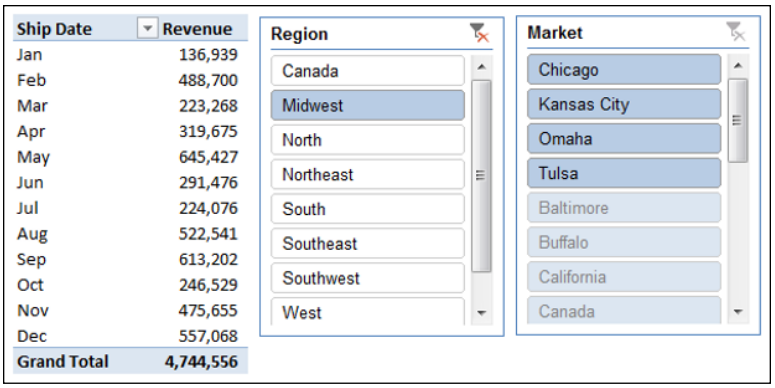
Columns that have a high number of unique values are particularly hard on Power Pivot performance. Columns such as Transaction ID, Order ID, and Invoice Number are often unnecessary in high-level Power Pivot reports and dashboards. So unless they are needed to establish relationships to other tables, leave them out of your model.

Limit the Number of Slicers in a Report

The slicer is one of the best new business intelligence (BI) features of Excel in recent years. Using slicers, you can provide your audience with an intuitive interface that allows for interactive filtering of your Excel reports and dashboards.

One of the more useful benefits of the slicer is that it responds to other slicers, providing a cascading filter effect. For example, Figure 13-3 illustrates not only that clicking on Midwest in the Region slicer filters the pivot table but that the Market slicer also responds, by highlighting the markets that belong to the Midwest region. Microsoft calls this behavior *cross-filtering.*

As useful as the slicer is, it is, unfortunately, extremely bad for Power Pivot performance. Every time a slicer is changed, Power Pivot must recalculate all values and measures in the pivot table. To do that, Power Pivot must evaluate every tile in the selected slicer and process the appropriate calculations based on the selection.



Ship Date	Revenue
Jan	136,939
Feb	488,700
Mar	223,268
Apr	319,675
May	645,427
Jun	291,476
Jul	224,076
Aug	522,541
Sep	613,202
Oct	246,529
Nov	475,655
Dec	557,068
Grand Total	4,744,556

FIGURE 13-3: Slicers work together to show relevant data items based on a selection.

Take this process a step further and imagine adding a second slicer: Because slicers cross-filter, each time you click one slicer, the other one changes also, so it's almost as though you clicked both of them. Power Pivot must now respond to both slicers, evaluating every tile in both slicers for each calculated measure in the pivot. Adding a second slicer effectively doubles the processing time. Add a third slicer, and you triple the processing time.

In short, a slicer is generally bad for Power Pivot performance. However, as mentioned at the beginning of this section, the functionality that the slicer brings to Excel BI solutions is too good to give up completely.

You can help to mitigate performance issues by limiting the number of slicers in your Power Pivot reports. Remove slicers one at a time, testing the performance of the Power Pivot report after each removal. You'll find that removing a single slicer is often enough to correct performance issues.

REMEMBER

Remove slicers that have low click rates. Some slicers hold filter values that, frankly, may never be utilized by your audience. For example, if a slicer allows your audience to filter by the current year or by last year, and the last year view is not often called up, consider removing the slicer or using the Pivot Table Filter drop-down list instead.

Create Slicers Only on Dimension Fields

Slicers tied to columns that contain lots of unique values will often cause a larger performance hit than columns containing only a handful of values. If a slicer contains a large number of tiles, consider using a Pivot Table Filter drop-down list instead.

REMEMBER

On a similar note, be sure to right-size column data types. A column with few distinct values is lighter than a column with a high number of distinct values. If you're storing the results of a calculation from a source database, reduce the number of digits (after the decimal) to be imported. This reduces the size of the dictionary and, possibly, the number of distinct values.

Disable the Cross-Filter Behavior for Certain Slicers

Disabling the cross-filter behavior of a slicer essentially prevents that slicer from changing selections when other slicers are clicked. This prevents the need for Power Pivot to evaluate the titles in the disabled slicer, thus reducing processing cycles. To disable the cross-filter behavior of a slicer, select Slicer Settings to open the Slicer Settings dialog box, shown in Figure 13-4. Then simply deselect the Visually Indicate Items with No Data option.

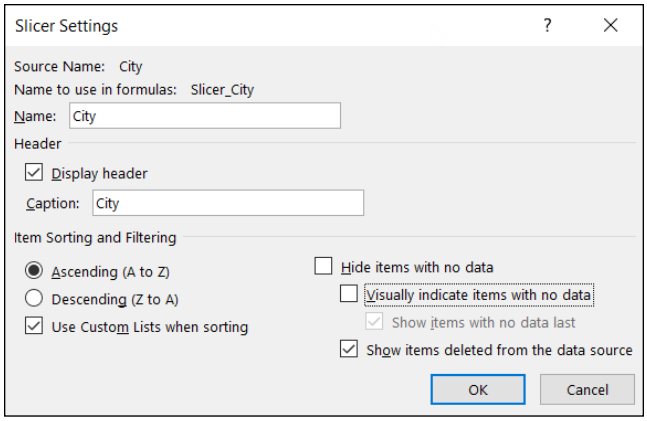


FIGURE 13-4: Deselecting the Visually Indicate Items option with No Data disables the slicer's cross-filter behavior.

Use Calculated Measures Instead of Calculated Columns

Use calculated measures instead of calculated columns, if possible. Calculated columns are stored as imported columns. Because calculated columns inherently interact with other columns in the model, they calculate every time the pivot table updates, whether they are being used or not. Calculated measures, on the other hand, calculate only at query time.

REMEMBER

Calculated columns resemble regular columns in that they both take up space in the model. In contrast, calculated measures are calculated on the fly and do not take space.

Upgrade to 64-Bit Excel

The suggestion in this section is somewhat obvious. If you continue to run into performance issues with your Power Pivot reports, you can always buy a better PC — in this case, by upgrading to a 64-bit PC with 64-bit Excel installed.

Power Pivot loads the entire data model into RAM whenever you work with it. The more RAM your computer has, the fewer performance issues you see. The 64-bit version of Excel can access more of your PC's RAM, ensuring that it has the system resources needed to crunch through bigger data models. In fact, Microsoft recommends 64-bit Excel for anyone working with models made up of millions of rows.

But before you hurriedly start installing 64-bit Excel, you need to answer these questions:

- **Do you already have 64-bit Excel installed?** You can find out by opening Excel and choosing File ➪ Account ➪ About Excel. A dialog box opens, specifying either `32-bit` or `64-bit` at the top.
- **Are your data models large enough?** Unless you're working with large data models, the move to 64-bit may not produce a noticeable difference in your work. How large is large? A Power Pivot workbook with a file size upward of 40 megabytes is considered large. If your workbook is 50 or more megabytes, you would definitely benefit from an upgrade.
- **Do you have a 64-bit operating system installed on your PC?** The 64-bit version of Excel will not install on a 32-bit operating system. You can find out whether you're running a 64-bit operating system by searching for the text *My PC 64-bit or 32-bit* at your favorite search engine. You'll see loads of sites that can walk you through the steps to determine your version.
- **Will your other add-ins stop working?** If you're using other add-ins, be aware that some of them may not be compatible with 64-bit Excel. You wouldn't want to install 64-bit Excel just to find that your trusted add-ins no longer work. Contact your add-in providers to ensure that they are 64-bit compatible. By the way, this advice includes add-ins for all Office products — not just Excel. When you upgrade Excel to 64-bit, you also have to upgrade the entire Office suite.

IN THIS CHAPTER

- Getting information from the Queries & Connections pane
- Organizing queries
- Referencing and duplicating queries
- Configuring Power Query options
- Viewing query dependencies

Chapter 14

Ten Tips for Working with Power Query

Over the past few years, Microsoft has added countless features to Power Query. It has truly become a rich tool set with multiple ways to perform virtually any action you can think of. This growth in functionality has paved the way to a good number of tips and tricks that can help you work more efficiently with your Power Query models.

This chapter presents ten of the more useful tips and tricks you can leverage to get the most out of Power Query.

Getting Quick Information from the Queries & Connections Pane

All the Power Query queries that live in a particular workbook can be viewed in the Queries & Connections pane. Choose Data➪Show Queries to activate the Queries & Connections pane.

In this pane, you can see some quick information about a query by simply hovering the cursor over it. You can see the data source for the query, the last time the query was refreshed, and a sneak peek of the data within the query. You can even click on column hyperlinks to peek at a particular column (see Figure 14-1).

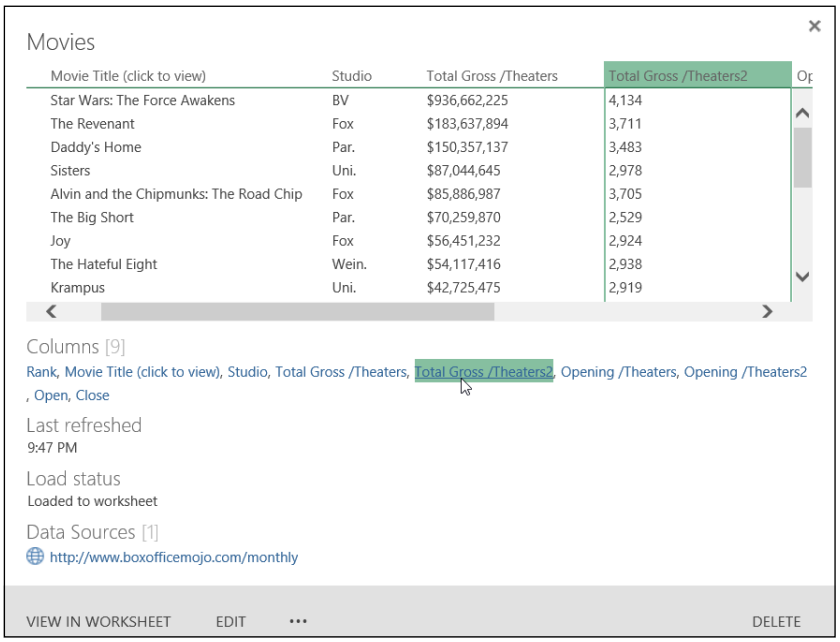


FIGURE 14-1: Hover the cursor over a query to get quick information, including sneak peeks of column contents.

TIP

It's always smart to reuse work wherever you can. Save time by duplicating the queries in your workbook. To do so, activate the Queries & Connections pane, right-click the query you want to copy, and then select Duplicate.

Organizing Queries in Groups

As you add queries to your workbook, your Queries & Connections pane may start to feel cluttered and disorganized. Do yourself a favor and organize your queries into groups.

Figure 14-2 illustrates the kinds of groups you can create. You can create a group only for custom functions or a group for queries sourced from external databases. You could even create a group where you store small reference tables. Each group is collapsible, so you can neatly pack away queries that you aren't working with.

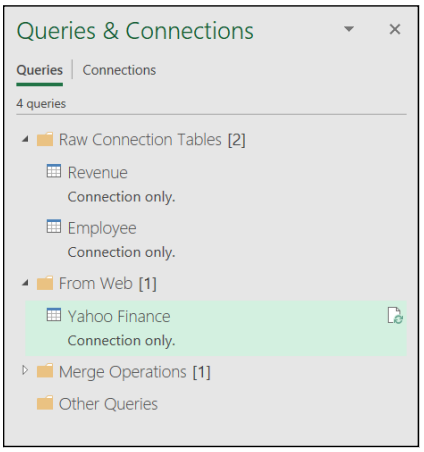


FIGURE 14-2: Queries can be organized into groups.

You can create a group by right-clicking a query in the Queries & Connections pane and selecting Move To Group➪New Group. To move a query to an existing group, right-click the query in the Queries & Connections pane, hover over Move To Group, and then select the group in which you want to see the target query. Right-clicking the group name will expose a set of options for managing the group itself.

Selecting Columns in Queries Faster

When dealing with a large table with dozens of columns in the Query Editor, it can be a pain to find and select the right columns to work with. You can avoid all that scrolling back and forth by choosing the Choose Columns command on the Home tab.

The dialog box shown in Figure 14-3 opens, showing you all available columns (including custom columns you may have added). You can easily find and select the columns you need.

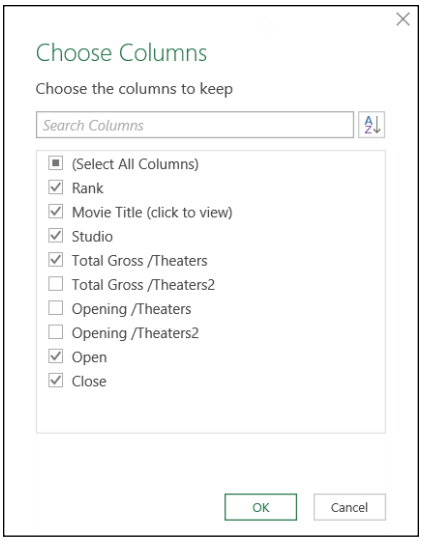


FIGURE 14-3: Use the Choose Columns command to find and select columns faster.

Renaming Query Steps

Every time you apply an action in the Query Editor, a new entry is made in the Query Settings pane, as shown in Figure 14-4. Query steps serve as a kind of audit trail for all the actions you've taken on the data.

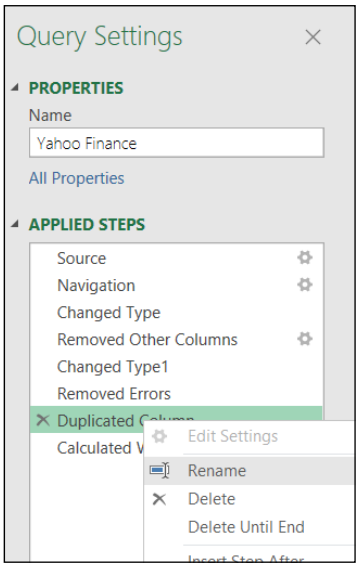


FIGURE 14-4: Get in the habit of renaming applied steps.

Query steps are automatically given generic names like Uppercased Text or Merged Columns. Why not take the time to add some clarity on what each step is doing? You can rename your steps step by right-clicking each step and selecting Rename.

Quickly Creating Reference Tables

A handful of columns in a data set always make for fantastic reference tables. For instance, if your data set contains a column with a list of product categories, it would be useful to create a reference table of all the unique values in that column.

Reference tables are often used to map data, feed menu selectors, serve as lookup values, and much more.

While in the Query Editor, you can right-click the column from which you want to create a reference table and then select Add as New Query, as shown in Figure 14-5.

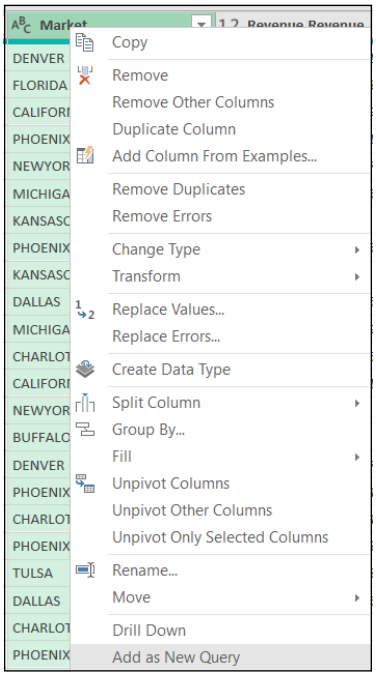


FIGURE 14-5: Create a new query from an existing column.

A new query is created, using the table you just pulled from as the source. The Query Editor jumps into action, showing only the column you selected. From here, you can use the Query Editor to clean up duplicates or remove blanks, for example.

Viewing Query Dependencies

On the View tab of the Power Query Editor window, you'll see the Query Dependencies command. Clicking this command activates the Query Dependencies dialog (see Figure 14-6), where you see a diagram displaying each of the queries in your workbook. Queries that rely on other queries have a line connecting them. This feature comes in handy when adopting someone else's workbook or even refreshing your memory about a workbook you worked on previously.

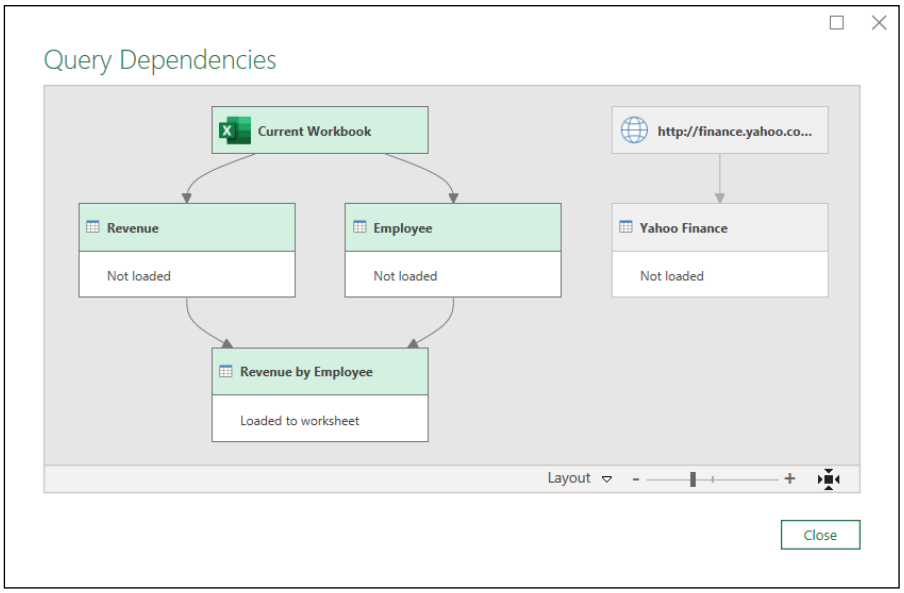


FIGURE 14-6: The Query Dependencies dialog box displays how each of your queries interacts with one another.

Truth be told, the utility of this feature *is* a bit limited. You can change the zoom and layout of the diagram, but you won't be able to move the objects around. As far as printing goes, you'll have to resort to screenshots, because Power Query doesn't offer a print feature.

Setting a Default Load Behavior

If you're working heavily with Power Pivot and with Power Query, chances are good that you load your Power Query queries to the Internal Data Model a majority of the time.

If you're one of those analysts who always loads to the Data Model, you can tweak the Power Query options to automatically load to the Data Model.

Choose Data➪Get Data➪Query Options to open the dialog box shown in Figure 14-7. Select Data Load in the Global section, select the Specify Custom Default Load Settings option button, and then select the Load to Data Model check box. This enables the options to load to the worksheet or Data Model by default.

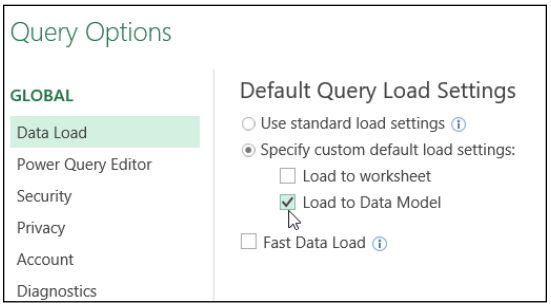


FIGURE 14-7: Use the Global Data Load options to set a default load behavior.

Preventing Automatic Data Type Changes

One of the more recent additions to Power Query is the ability to automatically detect data types and to proactively change data types. This type detection is most often applied when new data is introduced to the query.

For instance, Figure 14-8 shows the query steps after importing a text file. Note the Changed Type step, which was automatically performed by Power Query as part of its type detection feature.

Although Power Query does a decent job at guessing what data types should be used, applied data type changes can sometimes cause unexpected issues.

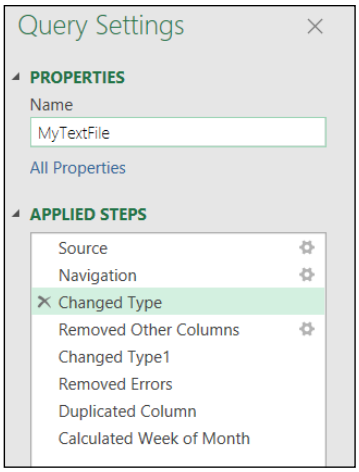


FIGURE 14-8: Power Query automatically adds a step to change data types when data is imported.

Some veterans of Power Query, frankly, find the type detection feature annoying. If data types need to be changed, *they* want to be the ones to make that determination.

If you'd rather handle data type changes without help from Power Query's type detection feature, you can turn it off.

Choose Data➪Get Data➪Query Options to open the dialog box shown in Figure 14-9. Select Data Load in the Current Workbook section, and then deselect the option to automatically detect column types and headers for unstructured sources.

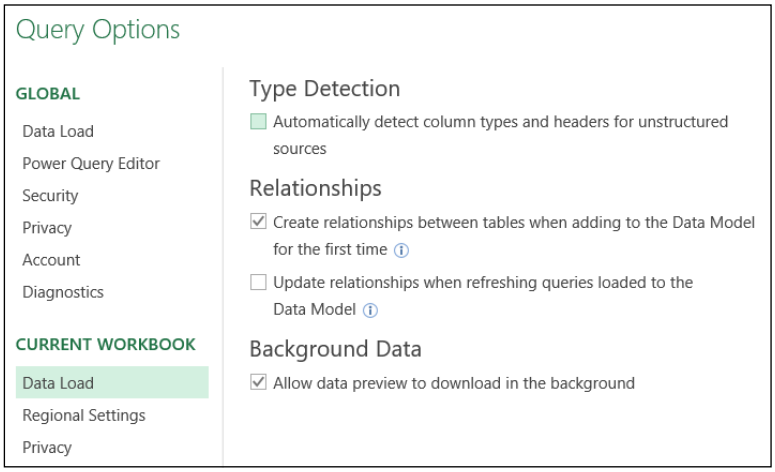


FIGURE 14-9: Disabling the type detection feature.

Disabling Privacy Settings to Improve Performance

The privacy-level settings in Power Pivot (explored in Chapter 11) are designed to protect organizational data as it gets combined with other sources. When you create a query that uses an external data source with an internal data source, Power Query stops the show to ask how you want to categorize the data privacy levels of each data source.

For a majority of analysts, who deal solely with organizational data, the privacy-level settings do little more than slow down queries and cause confusion.

Fortunately, you have the option to ignore privacy levels.

Choose Data➪Get Data➪Query Options to open the dialog box shown in Figure 14-10. Select Privacy in the Current Workbook section, and then choose the option to ignore privacy levels.

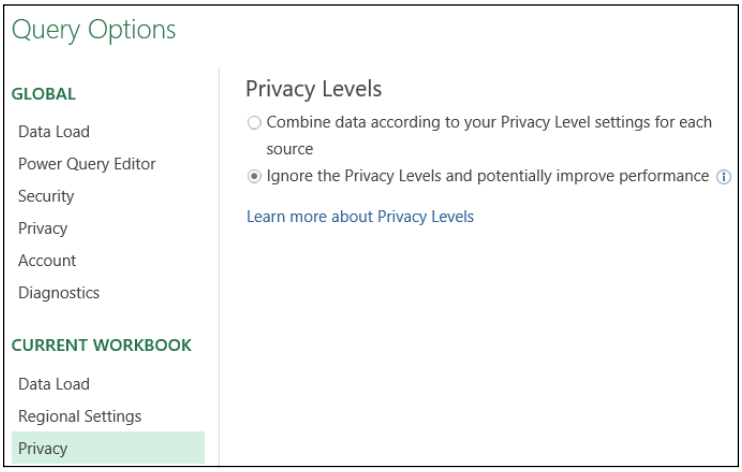


FIGURE 14-10: Disabling the privacy-level settings.

Disabling Relationship Detection

When you're building a query and choosing Load to Data Model as the output, Power Query, by default, attempts to detect relationships between queries and creates those relationships within the Internal Data Model. The relationships between queries are primarily driven by the defined query steps. For instance, if you were to merge two queries and then load the result into the Data Model, a relationship would be automatically created.

In larger data models with a dozen or so tables, Power Query's relationship detection can affect performance and increase the time it takes to load the Data Model.

You can avoid this hassle and even gain a performance boost by disabling relationship detection.

Choose Data➪Get Data➪Query Options to open the dialog box shown in Figure 14-11. Select Data Load in the Current Workbook section, and then deselect the option to create relationships when adding loading to the Data Model.

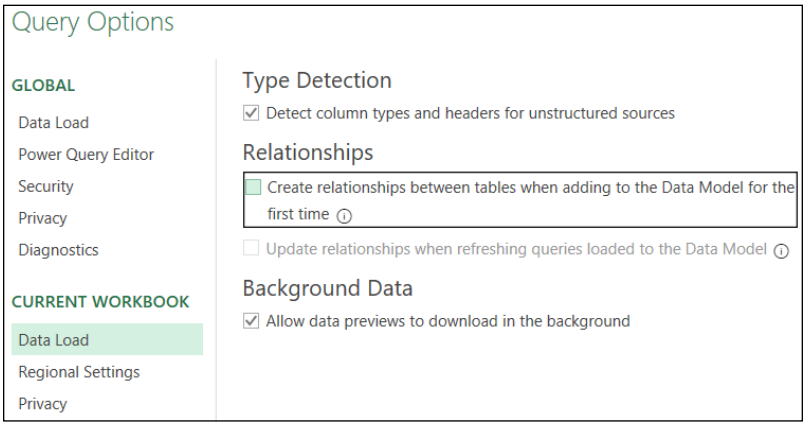


FIGURE 14-11: Disabling relationship detection.

Index

Symbols

- operator, 126
! operator, 126
* operator, 126
/ operator, 126
&& operator, 126
^ operator, 126
+ operator, 126
< operator, 126
= operator, 126
|| operator, 126
> operator, 126

A

actions
  Add as New Query, 156
  Add Column From Examples, 155, 158
  Add Conditional Column, 158
  Add Custom Column, 158
  Add Index Column, 158
  Append Queries, 158
  Change Type, 155
  Choose Columns, 158
  column-level, 155–157
  Create Data Type, 156
  Drill Down, 156
  Duplicate Column, 155
  Fill, 156
  Group By, 156
  Keep Bottom Rows, 158
  Keep Duplicates, 158
  Keep Errors, 158
  Keep Range of Rows, 158
  Keep Top Rows, 158
  Merge Column, 156
  Merge Queries, 158
  Move, 156
  Remove, 155
  Remove Alternate Rows, 158
  Remove Bottom Rows, 158
  Remove Duplicates, 155, 158
  Remove Errors, 155, 158
  Remove Other Columns, 155
  Remove Top Rows, 158
  Rename, 156
  Replace Errors, 156
  Replace Values, 156
  Split Column, 156
  table, 157–158
  Transform, 155
  Unpivot Other Columns, 156
  Unpivot Selected Columns, 156
  Use First Row as Headers, 158
Add as New Query action, 156
Add Column From Examples action, 155, 158
Add Conditional Column action, 158
Add Custom Column action, 158
Add Custom Column dialog box, 194, 200–201
Add Index Column action, 158
adding
  conditional logic to custom columns, 199–201
  Excel tables to data model, 22–24
  formulas to Power Pivot, 103–120
  report filters, 37–38
adjusting
  case, 181
  pivot table layouts, 40–41
  pivot tables, 36–37
  summary calculations, 43–44

Advanced Query Editor, 151
aggregate functions, 128–129
aggregate view, 201
aggregating data, 201–202
ampersand (&), DAX operators and, 126
analytical processes, transparency of, 9
AND operator, 126
Append dialog box, 213–214
Append feature
  about, 211–212
  appending data, 213–215
  creating base queries, 212–213
Append Queries action, 158
appending data, 213–215
applying
  conditional logic in DAX, 126–128
  numeric formats to data fields, 42–43
automatic refreshing, setting up, 84–85
AVERAGE function, 43, 110, 128–129
AVERAGEX function, 131–132
Azure databases, 165, 166

B

back-end database servers, 247–248
BI Blog, 138
building
  base queries, 212–213, 238–239
  calculated columns, 104–105
  calculated measures, 116–118
  custom columns, 193–201
  custom functions in Power Query, 225–229
  custom functions to merge data from multiple Excel files, 229–236
  DAX-driven calculated columns, 110–111
  parameter queries, 236–242
  pivot tables, 33–39
  pivot tables using Internal Data Model, 97–98
  reference tables, 257–258
  relationships between Power Pivot tables, 24–26
  slicers, 249–250
  standard slicers, 54–56
  timeline slicers, 59–61

C

CALCULATE function, 135–137
calculated columns, 250–251
calculated measures
  about, 116
  creating, 116–118
  deleting, 118–119
  editing, 118–119
  using instead of calculated columns, 250–251
case, changing, 181
case-sensitivity, in DAX, 125
Change PivotTable Data Source dialog box, 39
Change Type action, 155
changing
  case, 181
  pivot table layouts, 40–41
  pivot tables, 36–37
  summary calculations, 43–44
character markers, splitting columns using, 187–189
Cheat Sheet (website), 4
Choose Columns action, 158
Choose Columns command, 255–256
Clean command, 183–184
cleaning text, 183–184
Clipboard, loading data from, 81–82
column area, of pivot tables, 31–32
column-level actions, 155–157
columns
  about, 12
  adding conditional logic to custom, 199–201
  applying numeric formats to, 42–43
  calculated, 250–251
  concatenating, 179–181
  concatenating with custom, 195–196
  creating calculated, 104–105
  creating custom, 193–201
  customizing names, 41–42
  dimension, 249–250
  enhancing Power Pivot data with calculated, 103–108
  filling in blank, 178

formatting calculated, 105–106
hiding calculated, from end users, 107–108
limiting number of in data model tables, 246
with many unique values, 248
mismatched labels, 215
pivoting, 189–193
referencing from other pivot tables, 113–115
referencing in other calculations, 106–107
selecting in queries, 255–256
splitting using character markers, 187–189
unpivoting, 189–193
using functions in custom, 197–199
utilizing DAX to create calculated, 108–115
commands
Choose Columns, 255–256
Clean, 183–184
Custom Column, 193
Extract, 185
Merge Columns, 179–180, 195–196
Pivot Columns, 192–193
Queries & Connections, 84, 86
Refresh, 83–84
Refresh All, 85–86
Remove Duplicates, 177
Trim, 183–184
Unpivot Columns, 190–191
Unpivot Other Columns, 191–192
comma-separated value (CSV) files, getting data from, 161–162
comparison operators, 126
compatibility, 20
concatenating
columns, 179–181
with custom columns, 195–196
conditional logic
adding to custom columns, 199–201
applying in DAX, 126–128
Connection Properties dialog box, 84–85
controlling
connections, 96
data source settings, 170–171
existing queries, 153–154
external data connections, 83–87
multiple pivot tables with one slicer, 58–59
queries, 96
relationships, 26–27
relationships in Internal Data Model, 95–96
COUNT function, 43, 110, 129
Count Numbers function, 43
COUNTA function, 129
COUNTBLANK function, 129
COUNTROWS function, 129
Create Data Type action, 156
Create Data Type dialog box, 203–204
Create PivotTable dialog box, 33–36, 90–91, 97, 123
Create Table dialog box, 21–22, 240
creating
base queries, 212–213, 238–239
calculated columns, 104–105
calculated measures, 116–118
custom columns, 193–201
custom functions in Power Query, 225–229
custom functions to merge data from multiple Excel files, 229–236
DAX-driven calculated columns, 110–111
parameter queries, 236–242
pivot tables, 33–39
pivot tables using Internal Data Model, 97–98
reference tables, 257–258
relationships between Power Pivot tables, 24–26
slicers, 249–250
standard slicers, 54–56
timeline slicers, 59–61
cube functions, 119–120
Custom Column command, 193
Custom Column dialog box, 193, 228, 240
Custom Data feature (Power Query), 203–205
customizing
field names, 41–42
pivot table reports, 40–52
slicers, 56–58

D

data. *See also* external data
  aggregating, 201–202
  appending, 213–215
  getting from CSV files, 161–162
  getting from Excel workbooks, 160–161
  getting from folders, 164–165
  getting from other data systems, 167–169
  getting from PDF files, 163
  getting from text files, 161–162
  grouping, 201–202
  importing from database systems, 165–169
  importing from files, 160–165
  loading from Clipboard, 81–82
  loading from external Excel files, 76–78
  loading from flat files, 75–82
  loading from Microsoft Access databases, 70–72
  loading from other data sources, 82–83
  loading from other relational database systems, 72–75
  loading from relational databases, 64–75
  loading from SQL Server, 64–70
  loading from text files, 78–80
  separation of presentation and, 10–11
Data Analysis Expressions (DAX)
  about, 121
  aggregate functions, 128–129
  applying conditional logic in, 126–128
  CALCULATE function, 135–137
  filter context, 133–139
  FILTER function, 137–139
  iterator functions and row content, 129–132
  language fundamentals, 121–132
  operators, 125–126
  utilizing to create calculated columns, 108–115
data connections
  editing, 86–87
  managing, 96
  Power Query types, 159–174
  refreshing and managing external, 83–87
data items
  columns for, 57–58
  showing/hiding, 47–49
Data Link Properties dialog box, 73–74
data model, using in reporting, 27–28
data profiling
  about, 171–172
  options for, 172–173
  quick actions, 173–174
Data Source Settings dialog box, 170–171
data sources, managing settings, 170–171
data transformation
  about, 175
  adding conditional logic to custom columns, 199–201
  aggregating data, 201–202
  changing case, 181
  concatenating columns, 179–181
  concatenating with custom columns, 195–196
  creating custom columns, 193–201
  custom data types, 203–205
  data type conversions, 196–197
  extracting left, right, and middle values, 184–187
  filling in blank fields, 178–179
  filling in empty strings, 179
  finding and replacing specific text, 181–183
  grouping data, 201–202
  pivoting fields, 189–193
  removing duplicate records, 176–178
  replacing null values, 178–179
  splitting columns using character markers, 187–189
  tasks for, 176–193
  trimming and cleaning text, 183–184
  unpivoting fields, 189–193
  using functions with custom columns, 197–199
data types
  conversions, 196–197
  custom, 203–205
  preventing automatic changes, 259–260
database servers, back-end, 247–248
database systems, importing data from, 165–169
databases
  about, 11
  benefits of, 7–11
  terminology for, 11–13

DAX (Data Analysis Expressions)
about, 121
aggregate functions, 128–129
applying conditional logic in, 126–128
CALCULATE function, 135–137
filter context, 133–139
FILTER function, 137–139
iterator functions and row content, 129–132
language fundamentals, 121–132
operators, 125–126
utilizing to create calculated columns, 108–115
deleting
calculated measures, 118–119
duplicate records, 176–178
subtotals, 45–47
dependencies, query, 258
dialog boxes
Add Custom Column, 194, 200–201
Append, 213–214
Change PivotTable Data Source, 39
Connection Properties, 84–85
Create Data Type, 203–204
Create PivotTable, 33–36, 90–91, 97, 123
Create Table, 21–22, 240
Custom Column, 193, 228, 240
Data Link Properties, 73–74
Data Source Settings, 170–171
Edit Relationship, 26–27
Existing Connections, 87, 97–98
Extract Steps, 210
Format Cells, 43
Import, 148–149, 228
Import Data, 100, 161–162, 163, 213
Insert Slicers, 54–55
Insert Timelines, 59–61
Manage Measures, 118–119
Manage Relationships, 95–96
Measure, 116–118, 122
Merge, 218–221, 222–223
Merge Columns, 180, 208–209
Navigator, 99–100, 144
Paste Preview, 81–82
Privacy Levels, 241
Properties, 86, 152–153
Replace Values, 179, 182
Report Connections, 59
Slicer Settings, 58, 250
Sort by Column, 111–112
Split by Column Delimiter, 188–189
Table Import Wizard, 71, 73, 76–80
Value Field Settings, 41–42
dimension fields, 249–250
disabling
cross-filter behavior for slicers, 250
privacy settings, 261
relationship detection, 261–262
DISTINCTCOUNT function, 129
DIVIDE function, 128
dot (.) operator, 205
Drill Down action, 156
Duplicate Column action, 155

E

Edit button, 87
Edit Relationship dialog box, 26–27
editing
calculated measures, 118–119
data connections, 86–87
empty strings, filling in, 179
Excel (Microsoft)
creating custom functions to merge data from multiple files, 229–236
getting data from workbooks, 160–161
limits of, 7–11
loading data from external files, 76–78
upgrading to 64-bit, 251
Excel tables (Microsoft)
about, 11–12
adding to data model, 22–24
compared with views, 246
importing, 67–68
linking to Power Pivot, 20–29
preparing, 21–22

Excelerator BI, 138
Existing Connections dialog box, 87, 97–98
external data, using with Power Pivot, 63–87
external data tables, filling Internal Data Model with multiple, 98–101
Extract, Transform, Load (ETL), 143
Extract command, 185
Extract Steps dialog box, 210
extracting left, right, and middle values, 184–187

F

fields
  about, 12
  adding conditional logic to custom, 199–201
  applying numeric formats to, 42–43
  calculated, 250–251
  concatenating, 179–181
  concatenating with custom, 195–196
  creating calculated, 104–105
  creating custom, 193–201
  customizing names, 41–42
  dimension, 249–250
  enhancing Power Pivot data with calculated, 103–108
  filling in blank, 178
  formatting calculated, 105–106
  hiding calculated, from end users, 107–108
  limiting number of in data model tables, 246
  with many unique values, 248
  mismatched labels, 215
  pivoting, 189–193
  referencing from other pivot tables, 113–115
  referencing in other calculations, 106–107
  selecting in queries, 255–256
  splitting using character markers, 187–189
  unpivoting, 189–193
  using functions in custom, 197–199
  utilizing DAX to create calculated, 108–115
files, importing data from, 160–165
Fill action, 156
filling in
  blank fields, 178
  empty strings, 179
filling Internal Data Model with multiple external data tables, 98–101
filter area, of pivot tables, 32–33
filter context
  about, 133–135
  CALCULATE function, 135–137
  FILTER function, 137–139
Filter fields, 52
FILTER function, 137–139
finding and replacing specific text, 181–183
flat files, loading data from, 75–82
folders, getting data from, 164–165
Format Cells dialog box, 43
FORMAT() function, 110–111
Format Slicer pane, 57
formatting calculated columns, 105–106
Formula bar, 108–109
formulas, adding to Power Pivot, 103–120
Full Outer join, 216
functions
  aggregate, 128–129
  AVERAGE, 43, 110, 128–129
  AVERAGEX, 131–132
  CALCULATE, 135–137
  COUNT, 43, 110, 129
  Count Numbers, 43
  COUNTA, 129
  COUNTBLANK, 129
  COUNTROWS, 129
  cube, 119–120
  DISTINCTCOUNT, 129
  DIVIDE, 128
  FILTER, 137–139
  FORMAT(), 110–111
  IF, 110, 126–128, 199–201
  IFERROR, 110, 128
  ISBLANK, 127
  iterator, 129–132
  LEFT, 110, 184–187
  MAX, 43, 110, 128–129
  MAXX, 131–132
  MID, 110, 184–187
  MIN, 43, 110, 128–129

MINX, 131–132
MONTH, 110
nesting, 115
Product, 43
RELATED, 114–115
Replace Values, 182
RIGHT, 110, 184–187
StdDev, 43
StdDevP, 43
SUM, 43, 109–110, 128–129
SUMX, 131–132
SWITCH, 127–128
TRIM, 184
using with custom columns, 197–199
Var, 43
VarP, 43
VLOOKUP, 17, 28
YEAR, 110
functions, custom
about, 225
creating in Power Query, 225–229
creating parameter queries, 236–242
creating to merge data from multiple Excel files, 229–236
Power Query, 225–242
using in Power Query, 225–229
Fuzzy Match feature, 221–223

G

generating
base queries, 212–213, 238–239
calculated columns, 104–105
calculated measures, 116–118
custom columns, 193–201
custom functions in Power Query, 225–229
custom functions to merge data from multiple Excel files, 229–236
DAX-driven calculated columns, 110–111
parameter queries, 236–242
pivot tables, 33–39
pivot tables using Internal Data Model, 97–98
reference tables, 257–258
relationships between Power Pivot tables, 24–26
slicers, 249–250
standard slicers, 54–56
timeline slicers, 59–61
Group By action, 156
Group By feature (Power Query), 201–202
grouping data, 201–202
groups, organizing queries in, 254–255

H

hiding
calculated columns from end users, 107–108
data items, 47–49
items without data, 49–51

I

icons, explained, 3–4
identifying DAX functions safe for calculated columns, 108–110
IF function, 110, 126–128, 199–201
IFERROR function, 110, 128
Import Data dialog box, 100, 161–162, 163, 213
Import dialog box, 148–149, 228
importing
data from database systems, 165–169
data from files, 160–165
tables, 67–68
views, 67–68
improving
performance, 261
Power Pivot data with calculated columns, 103–108
Inner join, 216
Insert Slicers dialog box, 54–55
Insert Timelines dialog box, 59–61
Internal Data Model (Power Pivot)
about, 18–19, 89
creating new pivot tables using, 97–98
directly feeding, 89–95
filling with multiple external data tables, 98–101
limitations of Power Pivot-driven pivot tables, 94
managing queries and connections, 96
managing relationships in, 95–96

Internet resources
BI Blog, 138
Cheat Sheet, 4
Excelerator BI, 138
P3 Adaptive, 138
RADACAD, 138
SQLB, 138
ISBLANK function, 127
iterator function, 129–132

J

joins
defined, 25
Power Query, 216–217

K

Keep Bottom Rows action, 158
Keep Duplicates action, 158
Keep Errors action, 158
Keep Range of Rows action, 158
Keep Top Rows action, 158

L

layout, for pivot tables, 40–41
Left Anti join, 216
LEFT function, 110, 184–187
Left Outer join, 216
left values, extracting, 184–187
linking Excel tables to Power Pivot, 20–29
load behavior, setting default, 259
loading data
from Clipboard, 81–82
from external Excel files, 76–78
from flat files, 75–82
from Microsoft Access databases, 70–72
from other data sources, 82–83
from other relational database systems, 72–75
from relational databases, 64–75
from SQL Server, 64–70
from text files, 78–80

M

Manage Measures dialog box, 118–119
Manage Relationships dialog box, 95–96
managing
connections, 96
data source settings, 170–171
existing queries, 153–154
external data connections, 83–87
multiple pivot tables with one slicer, 58–59
queries, 96
relationships, 26–27
relationships in Internal Data Model, 95–96
many-to-many relationship, 15
mathematical operators, 126
MAX function, 43, 110, 128–129
MAXX function, 131–132
Measure dialog box, 116–118, 122
Merge Column action, 156
Merge Columns command, 179–180, 195–196
Merge Columns dialog box, 180, 208–209
Merge dialog box, 218–221, 222–223
Merge feature
about, 216
merging queries, 217–221
Power Query joins, 216–217
Merge Queries action, 158
merging queries, 217–221
Microsoft Access databases, 70–72, 168–169
Microsoft Analysis Services, 82
Microsoft Analytics Platform System, 82
Microsoft Excel
creating custom functions to merge data from multiple files, 229–236
getting data from workbooks, 160–161
limits of, 7–11
loading data from external files, 76–78
upgrading to 64-bit, 251

Microsoft Excel tables
about, 11–12
adding to data model, 22–24
compared with views, 246
importing, 67–68
linking to Power Pivot, 20–29
preparing, 21–22
Microsoft SQL Azure, 82
MID function, 110, 184–187
middle values, extracting, 184–187
MIN function, 43, 110, 128–129
MINX function, 131–132
modifying
case, 181
pivot table layouts, 40–41
pivot tables, 36–37
summary calculations, 43–44
MONTH function, 110
Move action, 156
multi-level relationships, 246–247

N

Navigator dialog box, 99–100, 144
nesting funtions, 115
nonstandard databases, 167
NOT operator, 126
null, 178
null values, replacing, 178–179
numeric formats, applying to data fields, 42–43

O

ODBC connections, 167
OLAP databases, 165
On the Web icon, 4
one-to-many relationship, 15
one-to-one relationship, 15
Open button, 87
operators, DAX, 125–126
OR operator, 126
organizing queries in groups, 254–255
Other Feeds data source, 83

P

P3 Adaptive, 138
parameter queries, creating, 236–242
parentheses (()), DAX operators and, 126
Paste Preview dialog box, 81–82
PDF files, getting data from, 163
performance
improving, 261
improving for Power Pivot, 245–251
Pivot Columns command, 192–193
pivot tables
creating using Internal Data Model, 97–98
limitations of Power Pivot-driven, 94
month sorting in Power-Pivot-driven, 111–112
referencing fields from other, 113–115
pivoting fields, 189–193
placing slicers, 56–57
Power Pivot
about, 17, 63
adding Excel tables to data model, 22–24
adding formulas to, 103–120
compatibility, 20
creating relationships between tables, 24–26
editing data connection, 86–87
enhancing data with calculated columns, 103–108
importing tables *vs.* importing views, 67–68
Internal Data Model, 18–19
linking Excel tables to, 20–28
loading data from Clipboard, 81–82
loading data from external Excel files, 76–78
loading data from flat files, 75–82
loading data from Microsoft Access databases, 70–72
loading data from other data sources, 82–83
loading data from other relational database systems, 72–75

Power Pivot *(continued)*
loading data from relational databases, 64–75
loading data from SQL Server, 64–70
loading data from text files, 78–80
managing existing relationships, 26–27
managing external data connections, 83–87
manually refreshing data, 83–84
Power Pivot tab, 19
preparing Excel tables, 21–22
preventing Refresh All, 85–86
refreshing external data connections, 83–87
setting up automatic refreshing, 84–85
tips for improving performance of, 245–251
using data model in reporting, 27–28
using external data with, 63–87
Power Pivot Ribbon interface, 18
Power Pivot tab, 19
Power Pivot tables
about, 29–30
adding report filters, 37–38
applying numeric formats to data fields, 42–43
areas of, 30–33
changing, 36–37
changing layout, 40–41
changing summary calculations, 43–44
column area, 31–32
controlling multiples with one slicer, 58–59
creating, 33–39
creating relationships between, 24–26
creating standard slicers, 54–56
creating timeline slicers, 59–61
customizing field names, 41–42
customizing reports, 40–52
customizing slicers, 56–58
data item columns, 57–58
filter area, 32–33
hiding data items, 47–49
hiding items without data, 49–51
miscellaneous slicer settings, 58
placement of slicers, 56–57
rearranging, 36–37
refreshing, 38–39
row area, 31
showing data items, 47–49
showing items without data, 49–51
size of slicers, 56–57
slicers, 52–54
sorting, 51–52
suppressing subtotals, 44–47
values area, 30–31
Power Query
about, 143–144
Advanced Editor, 151
column-level actions, 155–157
connection types, 159–174
custom functions, 225–242
data profiling with, 171–174
joins, 216–217
managing existing queries, 153–154
query steps, 150–152
refreshing data, 152–153
starting queries, 144–150
table actions, 157–158
tips for working with, 253–262
Power Query Editor, 150–152, 164–165
preparing
Excel tables, 21–22
for parameter queries, 236–237
presentation, separation of data and, 10–11
preventing
automatic data type changes, 259–260
Refresh All command, 85–86
primary key, 93
Privacy Levels dialog box, 241
privacy settings, disabling, 261
Product function, 43
Properties dialog box, 86, 152–153

Q

queries
about, 13, 207
Append feature, 211–215
appending data, 213–215

creating base, 212–213
Fuzzy Match feature, 221–223
managing, 96
managing existing, 153–154
Merge feature, 216–221
merging, 217–221
mismatched column labels, 215
organizing in groups, 254–255
Power Query joins, 216–217
renaming steps, 256–257
reusing steps, 208–211
selecting columns in, 255–256
starting, 144–150
steps for, 150–152
viewing dependencies, 258
Queries & Connections command, 84, 86
Queries & Connections pane, 153–154, 253–254
query context, 134
quotation marks (“”), 178

R

RADACAD, 138
raw information. *See* data
rearranging pivot tables, 36–37
records
about, 12
limiting number of in data model tables, 246
removing duplicate, 176–178
reference tables, creating, 257–258
referencing
calculated columns in other calculations, 106–107
fields from other pivot tables, 113–115
Refresh All command, 85–86
Refresh command, 83–84
refreshing
external data connections, 83–87
pivot tables, 38–39
Power Query data, 152–153
RELATED function, 114–115
relational database systems
about, 9
loading data from other, 72–75
relational databases, 64–75, 165
relationships
about, 13–15
creating between Power Pivot tables, 24–26
disabling detection, 261–262
managing, 26–27
managing in Internal Data Model, 95–96
multi-level, 246–247
Remember icon, 4
Remove action, 155
Remove Alternate Rows action, 158
Remove Bottom Rows action, 158
Remove Duplicates action, 155, 158
Remove Duplicates command, 177
Remove Errors action, 155, 158
Remove Other Columns action, 155
Remove Top Rows action, 158
Rename action, 156
renaming query steps, 256–257
Replace Errors action, 156
Replace Values action, 156
Replace Values dialog box, 179, 182
Replace Values function, 182
replacing null values, 178–179
Report Connections dialog box, 59
Report data source, 83
report filters, adding, 37–38
reports
limiting slicers in, 248–249
pivot table, 40–52
using Power Pivot data model in, 27–28
resources, Internet
BI Blog, 138
Cheat Sheet, 4
Excelerator BI, 138
P3 Adaptive, 138
RADACAD, 138
SQLB, 138
reusing query steps, 208–211
Right Anti join, 216

RIGHT function, 110, 184–187
Right Outer join, 216
right values, extracting, 184–187
row area, of pivot tables, 31
row context, 129–132
rows
  about, 12
  limiting number of in data model tables, 246
  removing duplicate, 176–178

S

scalability, as a benefit of databases, 8–9
selecting columns in queries, 255–256
self-service BI, demand for, 1
setting(s)
  data source, 170–171
  default load behavior, 259
  privacy, 261
setup, of automatic refreshing, 84–85
showing
  data items, 47–49
  items without data, 49–51
64-bit Excel, upgrading to, 251
sizing slicers, 56–57
Slicer Settings dialog box, 58, 250
slicers
  about, 52–54
  controlling multiple pivot tables with one, 58–59
  creating, 249–250
  creating standard, 54–56
  creating timeline, 59–61
  customizing, 56–58
  disabling cross-filter behavior for, 250
  limiting in reports, 248–249
snowflake schema, 247
Sort by Column dialog box, 111–112
sorting pivot tables, 51–52
Split by Column Delimiter dialog box, 188–189
Split Column action, 156
splitting columns using character markers, 187–189
SQL Server, loading data from, 64–70
SQLB, 138
starting queries, 144–150
StdDev function, 43
StdDevP function, 43
subtotals, suppressing, 44–47
SUM function, 43, 109–110, 128–129
summary calculations, changing, 43–44
SUMX function, 131–132
suppressing subtotals, 44–47
SWITCH function, 127–128

T

table actions, 157–158
Table Import Wizard, 65–80
Table Import Wizard dialog box, 71, 73, 76–80
tables (Excel)
  about, 11–12
  adding to data model, 22–24
  compared with views, 246
  importing, 67–68
  linking to Power Pivot, 20–29
  preparing, 21–22
tables (Power Pivot)
  about, 29–30
  adding report filters, 37–38
  applying numeric formats to data fields, 42–43
  areas of, 30–33
  changing, 36–37
  changing layout, 40–41
  changing summary calculations, 43–44
  column area, 31–32
  controlling multiples with one slicer, 58–59
  creating, 33–39
  creating relationships between, 24–26
  creating standard slicers, 54–56
  creating timeline slicers, 59–61
  customizing field names, 41–42
  customizing reports, 40–52
  customizing slicers, 56–58
  data item columns, 57–58

filter area, 32–33
hiding data items, 47–49
hiding items without data, 49–51
miscellaneous slicer settings, 58
placement of slicers, 56–57
rearranging, 36–37
refreshing, 38–39
row area, 31
showing data items, 47–49
showing items without data, 49–51
size of slicers, 56–57
slicers, 52–54
sorting, 51–52
suppressing subtotals, 44–47
values area, 30–31
target field, 51
Technical Stuff icon, 3
text
cleaning, 183–184
finding and replacing specific, 181–183
trimming, 183–184
text files
getting data from, 161–162
loading data from, 78–80
timeline slicers, 59–61
Tip icon, 3
Transform action, 155
transparency, of analytical processes, 9
Trim command, 183–184
TRIM function, 184
trimming text, 183–184

U

Unpivot Columns command, 190–191
Unpivot Other Columns action, 156
Unpivot Other Columns command, 191–192
Unpivot Selected Columns action, 156
unpivoting fields, 189–193
upgrading, to 64-bit Excel, 251
Use First Row as Headers action, 158

V

Value Field Settings dialog box, 41–42
values
about, 12
columns with many unique, 248
values area, of pivot tables, 30–31
Var function, 43
VarP function, 43
viewing query dependencies, 258
views
compared with tables, 246
importing, 67–68
VLOOKUP function, 17, 28

W

Warning icon, 3
Webb, Chris (blogger), 138
websites
BI Blog, 138
Cheat Sheet, 4
Excelerator BI, 138
P3 Adaptive, 138
RADACAD, 138
SQLB, 138

X

X-functions, 129–132

Y

YEAR function, 110

About the Author

Michael Alexander is a senior consultant at Slalom Consulting with more than 15 years of experience in data reporting management. He is the author of more than a dozen books on business analysis using Microsoft Excel and Microsoft Access. He has been named Microsoft Excel MVP for his contributions to the Excel community.

Author's Acknowledgments

My deepest thanks go to the professionals at John Wiley & Sons, Inc. for all the hours of work put into bringing this book to life. Thanks also to Elizabeth Kuball and Guy Hart-Davis for suggesting numerous improvements to the examples and text in this book. Finally, a special thank-you goes out to my family, for putting up with all the time I spent locked away on this project.

Publisher's Acknowledgments

Associate Editor: Elizabeth Stilwell
Project Editor: Elizabeth Kuball
Copy Editor: Elizabeth Kuball
Technical Editor: Guy Hart-Davis
Proofreader: Debbye Butler
Production Editor: Tamilmani Varadharaj
Cover Photos: © marco302/Getty Images

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.