

11

Session Eleven: Create Get & Transform queries

The goal is to transform data into information, and information into insight.

*Carly Fiorina (1954-),
American former technology business executive and former
Presidential candidate in the 2016 Republican primaries.*

Get and Transform is a re-branding of a tool that was previously called *Power Query*. The *Power Query* tool was previously only available as an add-in for the pro plus version of Excel 2010/2013.

Get & Transform is a powerful and complex professional tool. Entire books have been written to explain how to use it. This session will teach you everything you need to know in order to use this tool effectively.

By the end of the session you'll have skills that could potentially save an enormous amount of time in your day-to-day work with Excel.

Session Objectives

By the end of this session you will be able to:

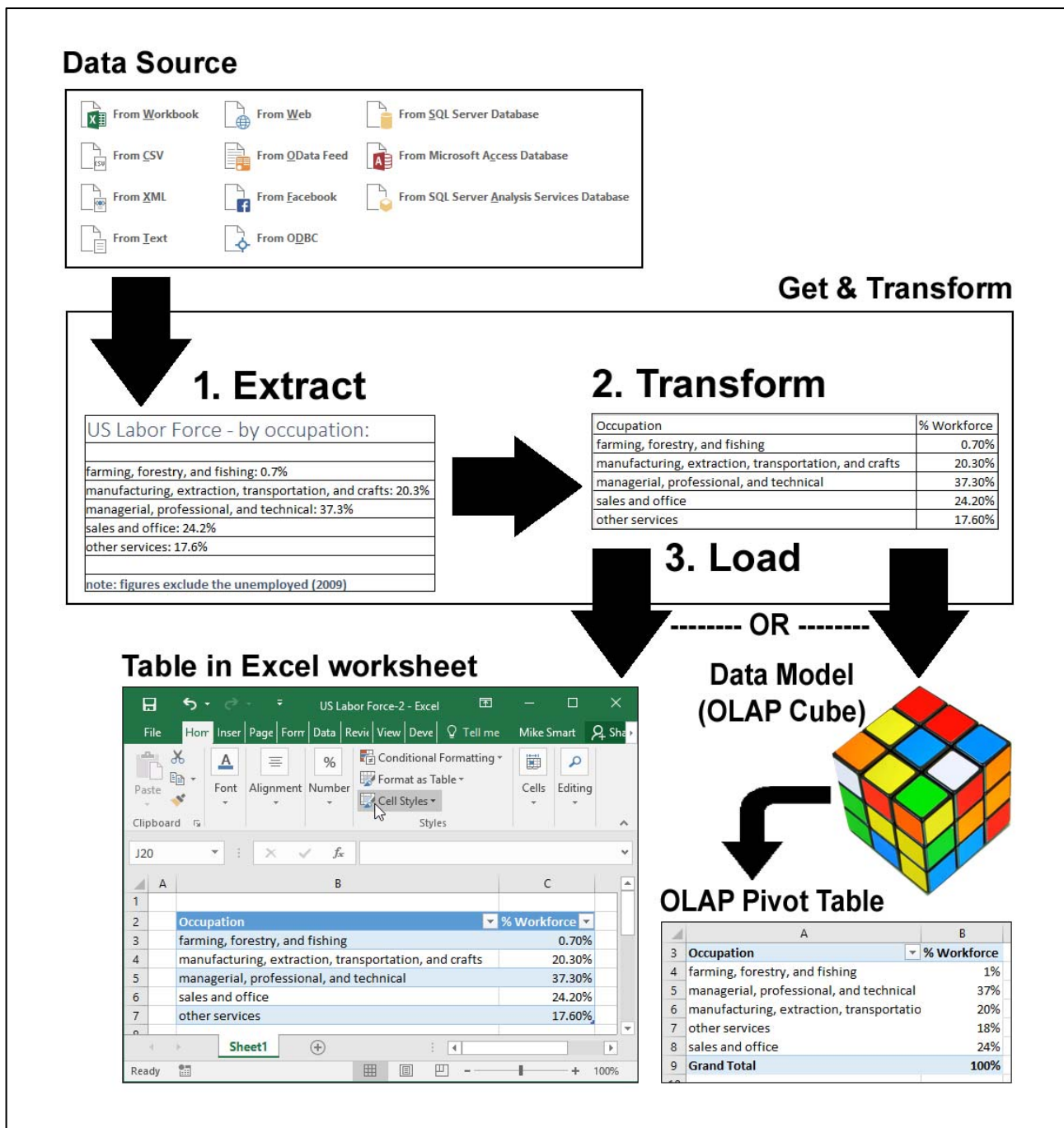
- Understand Get & Transform and ETL
- Create a simple extract and load web query
- Understand queries and connections
- Move, remove, rename, filter and sort columns
- Split delimited data
- Specify data types
- Understand steps and PQFL
- Remove empty, error and top and bottom rows
- Understand and work with null values
- Transform date and time columns
- Transform number columns
- Add a custom calculated column
- Create an aggregated data query
- Unpivot aggregated data
- Work with multiple queries
- Create an append query
- Understand normal and de-normalized data
- Create a simple two-table merged query
- Create a five-table merged query

Lesson 11-1: Understand Get & Transform and ETL

Get & Transform is an advanced ETL tool. ETL is an acronym for *Extract, Transform and Load*.

Up until now, you've transformed your data inside Excel itself. The new *Get & Transform* tool enables you to transform data before it is loaded into Excel. This provides many advantages.

The diagram below should make it clear what is actually meant by the terms Extract, Transform and Load.



note

Confusing terminology

In the world of IT we are very fond of terminology.

The process that I describe as *Data Transformation* is often referred to using the following terminology:

- Data Wrangling
- Data Cleansing
- Data Scrubbing
- Data Shaping
- Data Pre-Processing
- Data Munging

Quite what each of the terms above precisely mean seems to depend on who you ask.

In this session I will exclusively use the term *Data Transformation*.

You can, of course, use the above terminology liberally in meetings. This will make you sound very important and help to confuse your co-workers.

Personally I'd rather be referred to as a transformer than a munger, scrubber or wrangler.

The ETL methodology

Extract: Most data that is analyzed with Excel doesn't start its life in an Excel workbook but is imported from an external data source (often a relational database or CSV file). *Extract* simply means moving this data from the external data source into the Get & Transform tool.

Transform: Extracted data often isn't in a form that can be easily analyzed by Excel. There may be unwanted columns, badly named fields or corrupted data. The Get & Transform tool includes many features that enable you to clean your data before loading it into an Excel table or data model. This cleaning process is called Data Transformation (see sidebar).

Load: This simply means exporting the transformed data from the Get & Transform tool. Get & Transform can export transformed data to an Excel table or an Excel data model (OLAP cube) for use with an OLAP pivot table or 3D Map.

Why is ETL better than ELT?

When you add and delete columns, change column headers, re-format columns and add calculated columns to tables you are *transforming* your data. This means that, up until now, you've been using an ELT methodology (*Extract* from data source, *Load* into Excel and *Transform* within Excel).

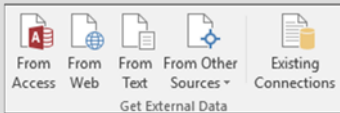
Here are of the advantages in using Get & Transform to implement an ETL methodology instead of using standard Excel features:

1. **Re-usable and sharable queries:** The Get & Transform tool generates a re-usable *Query*. This means that all of the Get & Transform actions can be repeated to refresh the data in your worksheet with a single click. You'll learn more about queries later, in: *Lesson 11-3: Understand queries and connections*.
2. **Automatically refreshed data:** You can configure a query to automatically refresh an Excel table at a timed interval. This means that if your data source contains rapidly changing data, you can keep your workbook data up to date at all times. You'll learn how to automatically refresh data later, in: *Lesson 11-3: Understand queries and connections*.
3. **The ability to transform big data:** Get & Transform does not share Excel's limits (of approximately a million rows of data) so it can be used to transform big data (sending the result directly to an OLAP cube, 3D Map, or to an Excel worksheet after aggregation). You'll learn more about pre-aggregating data later, in: *Lesson 11-13: Create an aggregated data query*.
4. **Better tools:** Get & Transform has some advanced transformation tools that are not found in the standard Excel product (for example, the *Unpivot Columns* tool that you'll learn about in: *Lesson 11-14: Unpivot aggregated data*).
5. **The ability to combine data:** Get & Transform enables you to merge queries. This enables you to combine relational data from disparate data sources to create a de-normalized data extract. You'll learn about this later, in: *Lesson 11-16: Create an append query* and *Lesson 11-18: Create a simple two-table merged query*.

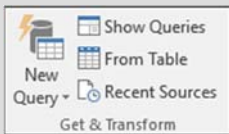
note

Get & Transform makes the old Get External Data tools redundant

You'll still find the *Get External Data* group on the *Data* tab of the ribbon.



Get External Data is an *Extract* and *Load* tool.



Get & Transform is an *Extract*, *Transform* and *Load* (ETL) tool.

In this lesson you used *Get & Transform* as a simple *Extract* and *Load* tool, as you didn't transform the data in any way.

Even when you do not need to transform data, you should still use the new *Get & Transform* tool in preference to the older *Get External Data* tools.

Get & Transform works more reliably than the older *Get External Data* tools and (as you'll discover later, in: *Lesson 11 5: Understand queries and connections*) generates queries that are easier to manage and share.

note

Why not use a real currency exchange site?

In earlier versions of this book I directed readers to a "real" exchange rate site that subsequently changed, making the lesson impossible to follow.

For this book I've published my own exchange rate data on the ExcelCentral.com website.

Lesson 11-2: Create a simple extract and load web query

There are many sites on the Internet containing useful data that is constantly changing. Obvious examples are currency exchange rates and stock prices. This type of information is often presented on a web page.

Get & Transform allows you to create a web query that will extract data from a specified table on a specified web page

When a web query has been created, it is possible to refresh the information in your worksheet at any time both manually and automatically at a specified time interval.

- 1 Open a new blank workbook.
- 2 Open your web browser and go to:
http://ExcelCentral.com/webquery/exchangerates

Type the above URL into your web browser's address bar.

A simple web page appears showing eight current (fictitious) currency exchange rates.

Notice that the exchange rates update every five seconds.

- 3 Create a web query to extract currency exchange rates from:
http://ExcelCentral.com/webquery/exchangerates

1. Return to Excel and select cell A1.
2. Click:

Data→Get & Transform→New Query→
From Other Sources→From Web

The *From Web* dialog appears, requesting a URL.

3. Type: **http://ExcelCentral.com/webquery/exchangerates** into the *URL* box.

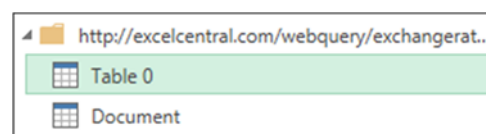


4. Click the *OK* button.

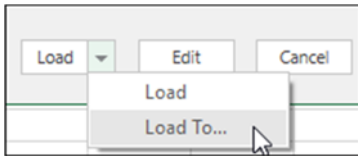
The *Navigator* pane appears, listing a single table and a document.

On a more complicated web page you could see many tables. This is a simple page with only one table.

5. Select the *Table 0* table.



A preview of the table appears in the right hand pane of the dialog.



note

The Load To dialog options

1. *Table in New Worksheet.*



Sends the output to an Excel table in a new worksheet.

2. *Table in Existing Worksheet.*



Sends the output to an Excel table in a specified location in an existing worksheet (the option you used in this lesson).

3. *Add this data to the Data Model.*



If this check box is checked, the data returned by the query will be sent to the data model. The data model can then be used as the data source for an OLAP pivot table or 3D Map.

4. *Only Create Connection.*



This option allows you to save a query to use later. You'll learn how to use saved queries later, in: *Lesson 11-3: Understand queries and connections.*

The *Only Create Connection* option can be used in conjunction with the *Add this data to the Data Model* option to work with *big data* (data that contains more than 1,048,576 rows). This enables you to directly analyze the data returned by the query in an OLAP Pivot Table without first loading it into an Excel table.

You learned about working with big data in: *Lesson 9-8: Work with big data.*

Notice that there are three buttons at the bottom of the dialog: *Load, Edit* and *Cancel*.

The *Edit* button would take you to the query editor window where you would be able to transform your data in many ways. You'll use the transformation tools later in this session.

For the moment you will use the *Get & Transform* tool only to *extract* and *load* the data. You've already *extracted* the data and are now ready to *load* it into the worksheet.

4 Load the extracted data into cell A1 on the currently selected worksheet

1. Click: Load (drop-down arrow)→Load To...

The *Load To* dialog appears. Notice that the default load location is a table in a new worksheet.

2. Click the *Existing Worksheet* option button.

Cell A1 is already selected, so the data will be loaded into a table in cell A1 of the currently selected worksheet.

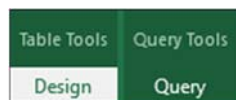
See sidebar for a discussion of the other load options.

3. Click the *Load* button.

The exchange rates are displayed in a table upon the worksheet.

| | A | B | C | D | E |
|---|--------------------|-------------|--------|----------|---------|
| 1 | Place | Currency | Symbol | 1 USD | Inverse |
| 2 | Atlantis | Orich | ATO | 0.8061 | 1.2406 |
| 3 | Narnia | Lion | NAL | 0.6848 | 1.4602 |
| 4 | Middle Earth | Castar | MEC | 67.7414 | 0.0148 |
| 5 | Hogsmeade Village | Gringott | HOG | 1.2182 | 0.8209 |
| 6 | Kingdom of Zamunda | Pound | ZAP | 0.9407 | 1.0631 |
| 7 | El Dorado | Gold Dollar | ELD | 1.4877 | 0.6722 |
| 8 | Lilliput | Sprug | LIS | 125.6879 | 0.008 |
| 9 | San Seriffe | Corona | SSC | 1.4199 | 0.7043 |

Note that this is a rather special table. If you click inside the table, you'll see a new *Query Tools* ribbon tab. This tells you that the table is linked to a *Get & Transform* query.



5 Refresh the data shown in the table.

Because the table is linked to a *Get & Transform* query, you can execute the query at any time to update the exchange rates.

1. Click inside the table to select it.
2. Click: Query Tools→Query→Load→Refresh.

Every time you click the *Refresh* button you'll see the values change to show up-to-the-second exchange rates.

Later, in: *Lesson 11-3: Understand queries and connections*, you'll learn how to automatically refresh the query at a timed interval (of as little as one minute). The worksheet data is then always up to date.

6 Save your work as *Exchange Rates-1*.

note**Where are queries and connections saved?**

When you save an Excel workbook containing a query, the query and connection are automatically saved as part of the workbook.

note**Auto-refresh causes focus to move to the auto-refreshed table**

Auto-refresh connections to Excel tables have an annoying "feature". Here is a scenario that demonstrates the problem.

1. You create an Excel table that is linked to a *Get & Transform* query in a workbook named *Workbook-1*.
2. You set the connection to automatically refresh.
(These two steps are exactly what you did in this lesson).
3. You open another workbook called *Workbook-2*.
4. While you are working in *Workbook-2*, the table in *Workbook-1* automatically updates.
5. You are automatically taken to the *Workbook-1* worksheet that contains the updated table.

There isn't a simple solution to this problem (apart from closing the workbook that is automatically refreshing).

Note that there is not a problem if you are working in a different worksheet in *Workbook-1* (in the above scenario). In this case you will remain on the selected worksheet.

Lesson 11-3: Understand queries and connections

A *Get & Transform* query contains a *Connection* that it uses to connect to a data source. The connection is used to *Extract* raw data. The query also knows how to *Transform* the data.

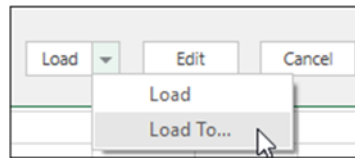
Crucially, the query does not know what to do with the data after transforming it. In other words, it does not contain any *Load* information.

This provides many advantages. You can share a query with other users who can decide where they want to load the resulting query data.

1. Open a new blank workbook.
2. Create a web query to extract currency exchange rates from: <http://ExcelCentral.com/webquery/exchangerates>

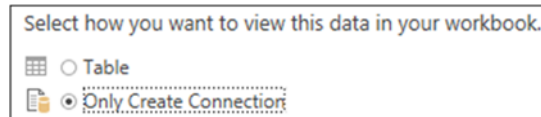
You learned how to do this in: *Lesson 11-2: Create a simple extract and load web query*.

3. Save the query without loading the data anywhere.
 1. Click: Load→Load To...



The *Load To* dialog appears. Notice that the default load location is a table in a new worksheet.

2. Click the *Only Create Connection* option button.



Only Create Connection is rather confusing terminology, as you are actually creating a *Query* that uses a connection (and not a simple connection).

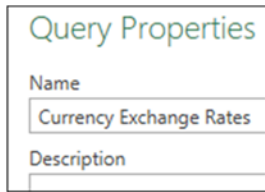
3. Click the *Load* button.
Once again, this is rather confusing as you are not loading the data anywhere but saving a query along with a connection.
4. View query information using the *Workbook Queries* task pane.

1. The *Workbook Queries* task pane should be visible on the right of your screen. If you have closed it, click:
Data→Get & Transform→Show Queries
... to bring it back again.
2. Hover the mouse cursor over the *Table 0* query.

A large amount of information about the query is shown in an information box.

5 Give the query a meaningful name.

1. Right-click on the *Table 0* query in the *Workbook Queries* task pane.
2. Click: *Properties* from the shortcut menu.
3. Type: **Currency Exchange Rates** into the *Name* box.
4. Click the *OK* button.



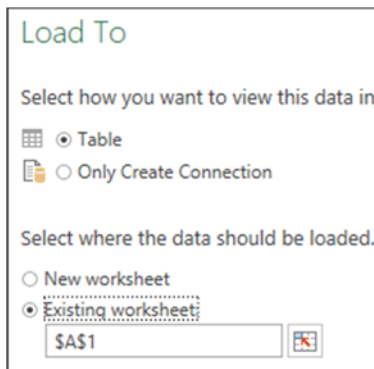
6 Use the *Currency Exchange Rates* query to provide data to a table on the current worksheet.

1. Right-click on the *Currency Exchange Rates* query in the *Workbook Queries* task pane.
2. Click: *Load To...* from the shortcut menu.
3. Click the *Table* option button.
4. Click the *Existing worksheet* option button.
5. Click the *Load* button.

The data returned by the query is displayed in a table on the current worksheet.

6. Click inside the table to select it.
7. Look in the: *Table Tools*→*Design*→*Properties*→*Table Name* box.

Notice that Excel has automatically named the table: *Currency_Exchange_Rates*.



7 Change the connection properties so that the query refreshes every one minute.

1. Click: *Data*→*Connection*→*Connections*.

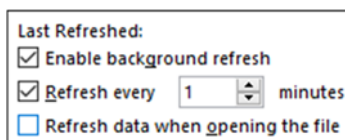
The *Workbook Connections* dialog appears, showing a single connection as: *Query – Currency Exchange Rates*.

This is a little confusing as the dialog only refers to the connection used to extract the data for the *Currency Exchange Rates* query and not to the query itself.

2. Click the *Properties...* button.

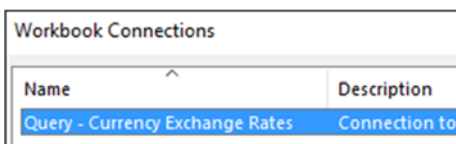
The *Connection Properties* dialog appears.

3. On the *Usage* tab, check the *Refresh every* check box and set the time interval to 1 minute.
4. Click the *OK* button and then the *Close* button to close both dialogs.
5. Observe the table. Notice that every 60 seconds the currency exchange rates all refresh to show current values.



8 Save your work as *Exchange Rates-2* and close the workbook.

You need to close the workbook to prevent the auto-refresh focus problem described on the facing page sidebar.



Lesson 11-4: Move, remove, rename, filter and sort columns

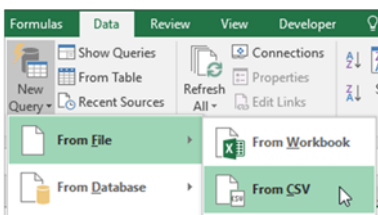
- 1 Open a new blank workbook.
- 2 Create a *Get & Transform* query to extract the values contained in the *Employees-1* comma separated values (CSV) file.

The CSV file format is often used when exporting data from other applications. If you were to open the *Employees-1* CSV file in *Notepad*, you'd see this:

You can see that the first line of the file has field names separated by commas. The second line then has field values, also separated by commas.

Get & Transform can use a CSV file as a data source.

1. Click:
Data→Get & Transform→New Query→From File→From CSV
2. Navigate to the *Employees-1* CSV file in your sample files folder.



| Name | Date modified | Type |
|-------------|------------------|---|
| Employees-1 | 18/02/2016 14:23 | Microsoft Excel Comma Separated Values File |

3. Either double-click the file name or click once on the file name to select and then click the *Open* button.

The *Get & Transform* query editor window opens.

| | Last Name | First N... | Job Description | Title Of Courtesy | Birth Date | Address | City | Region | Postal C... | Cou... |
|---|-----------|------------|-----------------------|-------------------|------------|----------------------------|----------|--------|-------------|--------|
| 1 | Davolio | Nancy | Sales Representative | Ms. | 08/12/1968 | 507 - 20th Ave. E. Apt. 2A | Seattle | WA | 98122 | USA |
| 2 | Fuller | Andrew | Vice President, Sales | Dr. | 19/02/1952 | 908 W. Capital Way | Tacoma | WA | 98401 | USA |
| 3 | Leverling | Janet | Sales Representative | Ms. | 30/08/1963 | 722 Moss Bay Blvd. | Kirkland | WA | 98033 | USA |

Notice that, this time, *Get & Transform* didn't ask you to select a table, as a CSV file can only contain a single table.

You can see that the sample file contains thirteen different fields with an assortment of information about each employee. In this lesson you will transform this into a compact telephone number listing by removing many of the existing fields.

- 3 Remove unwanted columns so that only the *Last Name*, *First Name*, *Job Description*, *Title of Courtesy*, *Country* and *Home Phone* fields remain.

1. Click: Home→Manage Columns→Choose Columns.

The *Choose Columns* dialog appears.

Employees-1 (CSV)

Choose Columns

Choose the columns to keep

Search Columns

- (Select All Columns)
- Last Name
- First Name
- Job Description
- Title Of Courtesy
- Birth Date
- Address
- City
- Region
- Postal Code
- Country
- Home Phone
- Extension
- Reports To

2. Click the (*Select All Columns*) check box to uncheck all check boxes.
3. Check the boxes for the fields you want to keep.
4. Click the *OK* button.

The query editor grid now shows only six columns.

4 Rename the *Title Of Courtesy* field to: **Title**

1. Double-click on the text: *Title Of Courtesy* in the column header.
2. Type: **Title** followed by the <Enter> key.

5 Change the order of the columns so that they match the screen grab below.

Click and drag each column header to the required position.

| | Country | Title | First Name | Last Name | Job Description | Home Phone |
|---|---------|-------|------------|-----------|-----------------------|----------------|
| 1 | USA | Ms. | Nancy | Davolio | Sales Representative | (206) 555-9857 |
| 2 | USA | Dr. | Andrew | Fuller | Vice President, Sales | (206) 555-9482 |
| 3 | USA | Ms. | Janet | Leverling | Sales Representative | (206) 555-3412 |

6 Alphabetically sort, first by *Country* and then by Last Name

1. Click on the drop-down arrow to the right of the *Country* column header.
2. Select *Sort Ascending* from the drop-down dialog.
3. Click on the drop-down arrow to the right of the *Last Name* column header.
4. Select *Sort Ascending* from the drop-down dialog.

Notice that this has resulted in a two-level sort. All countries are grouped together and, within each country, last names are sorted from A-Z.

7 Filter the list so that only USA telephone numbers are shown.

1. Click the drop-down arrow to the right of the *Country* column header.
2. Remove the check from the *UK* check box and click the *OK* button.

8 Close the query editor and load the query result to a table in cell A1 of the worksheet (there is only one worksheet in this workbook).

1. Click: Home→Close→Close & Load (drop-down arrow)→Close and Load To...

The *Load To* dialog appears.

2. Select the *Table* and *Existing Worksheet* options.
3. Make sure that cell **A1** is entered into the *Existing Worksheet* cell box.
4. Click: *Load*.

9 Save your work as *Phone List-1*.

note

Other ways to remove unwanted columns in the Query Editor

1. Select one or more columns by clicking on the column headers. (You can use **Ctrl-Click** to select individual columns or **Shift-Click** to select an adjacent set of columns).

2. Right-click on any of the selected column headers and (from the shortcut menu) either click *Remove Columns* (to remove the selected columns) or *Remove Other Columns* (to remove the unselected columns).

These shortcut menu options are also available on the ribbon at:

Home→Manage Columns→Remove Columns (drop-down list)

note**Get & Transform only retrieves data from the workbook file**

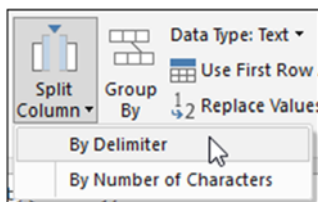
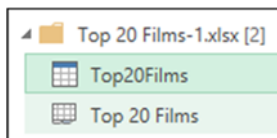
In this lesson you use a Get & Transform query to extract data from a table.

Both the Get & Transform query, the source (Get) table, and the target (Load) table reside in the same workbook.

If you change a value in the source table and then refresh the target table, your changes will not update in the target table.

This is because the query is accessing the workbook file on the hard drive and not the in-memory version of the workbook that you are working with.

You would have to change a value in the source table, save the workbook and then refresh the target table to see the result of the change.

**Top 20 Films-1**

Lesson 11-5: Split delimited data

- 1 Open *Top 20 Films-1* from your sample files folder.

The sample file has a single table named: *Top20Films*.

| | A | B | C | D |
|---|---------------|------------------|--------|---------------|
| 3 | Title | Studio | Gross* | Director |
| 4 | Avatar (2009) | 20th Century Fox | 2,187 | James Cameron |

The table is very easy to read for a human but is badly designed from a data perspective. In: *Lesson 2-1: Split fixed width data using Text to Columns*, you learned why you should not have more than one data element in each table field.

In this sample file the *Title* column contains two data elements (*Title* and *Date*). They should be in separate columns.

- 2 Create a *Get & Transform* query to extract the values contained in the *Top20Films* table.

Up until now you've used *Get & Transform* to connect to external data sources. In this lesson you'll query an Excel table and load the query result back into the same workbook.

1. Click: Data→Get & Transform→New Query→From File→From Workbook.
2. Select the *Top 20 Films-1* workbook (the workbook you currently have open).
3. Click the *Import* button.

The Get & Transform Navigator screen appears. Note that two elements are shown.

The element named with spaces is the entire *Top 20 Films* worksheet and includes header and footer information.

The element named without spaces is the *Top20Films* Excel table.

4. Click on the *Top20Films* table and click the *Edit* button to enter the Get & Transform query editor.

- 3 Extract the date from the *Title* column and place it into a new column.

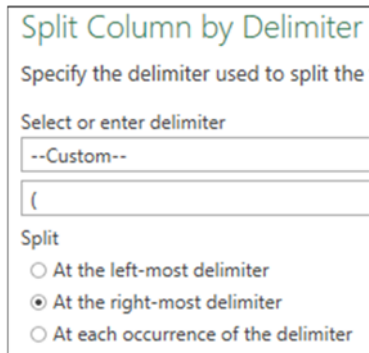
In: *Lesson 2-2: Split delimited data using Text to Columns*, you learned how you could achieve this using Excel's *Text to Columns* tool.

Get & Transform's *Split Column* feature works in a similar way.

1. Click anywhere in the *Title* column.
2. Click: Transform→Text Column→Split Column (drop-down list)→By Delimiter.

Note that there's also a *By Number of Characters* option. You could use this option to split fixed width data (you learned

about fixed width data in: *Lesson 2-1: Split fixed width data using Text to Columns*).



The *Split Column by Delimiter* dialog appears.

3. Select -- Custom -- from the *Select or enter delimiter* drop-down list.
4. Type an opening bracket into the text box.
5. Click the: *At the right-most delimiter* option button.

In this data it doesn't really matter which option you select as there is only one opening bracket in each title. The option you have just selected specifies that the split should take place at the right-most occurrence of a bracket character.

6. Click the *OK* button.

The *Title* field now only contains the film title and a new column has been inserted showing the year along with a closing bracket.

| | Title.1 | Title.2 | Studio |
|---|---------|---------|----------------------------|
| 1 | Avatar | 2009) | 20th Century Fox |
| 2 | Titanic | 1997) | 20th Century Fox/Paramount |

4. Rename the *Title.1* column header to: **Title** and the *Title.2* column header to: **Year**.

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.

5. Remove the closing bracket from the *Year* column.

1. Click anywhere in the *Year* column.
2. Click: Transform → Any Column → Replace Values.

The *Replace Values* dialog appears.

3. Type a closing bracket into the *Value to Find* box.
4. Leave the *Replace With* box empty.

This will replace any closing brackets found in the column with nothing.

5. Click the *OK* button.

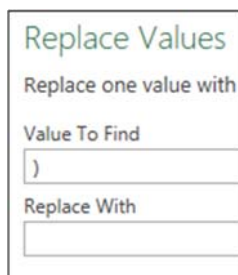
All of the closing brackets are removed from the *Year* column.

6. Close the query editor and load the query result to a table in cell A1 of a new worksheet.

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.

| | A | B | C |
|---|--------|------|------------------|
| 1 | Title | Year | Studio |
| 2 | Avatar | 2009 | 20th Century Fox |

7. Save your work as *Top 20 Films-2*.



note**Solutions to the “penny rounding” problem**

Consider the simple addition of two values:

| | A | | A |
|---|-------|---|---------|
| 1 | 1.206 | 1 | \$ 1.21 |
| 2 | 2.206 | 2 | \$ 2.21 |
| 3 | 3.412 | 3 | \$ 3.41 |

In the examples above, cell A3 has the formula: =A1+A2.

In the right-hand example, the values have been formatted as *currency*.

Best practice

Excel has a ROUND function that can be used to round the result to two decimal places at the point of calculation.

The new formula would become:

```
=ROUND(B1,2) +  
ROUND(B2,2)
```

Changing precision

Click: File→Options→Advanced.

In the *When Calculating This Workbook* group, click the *Set precision as displayed* check box.

This alters the default behaviour of Excel so that each value in the worksheet is regarded as being precisely the same as its display value when performing calculations.

Using this option can cause data to become inaccurate as it is a global setting, affecting the entire worksheet (and any other worksheets you may open before turning it off).

Use this feature with extreme caution, or better still, don't use it at all!

Lesson 11-6: Specify data types

What are data types?

Excel only supports four native data types: *Number*, *Text*, *Logical* and *Error*.

| Data Type | Example |
|-----------|----------------------|
| Number | 22.4567 |
| Text | John Smith |
| Logical | TRUE or FALSE |
| Error | #DIV/0, #N/A, #NULL! |

Relational database products such as Access and SQL Server have far more data types. For example, Access supports six different numeric data types (such as *Integer* and *Currency*).

Excel simulates data types by formatting

In: *Lesson 3-7: Understand date serial numbers*, you learned that Excel uses the *Number* data type to store date and time values. The ingenious use of date serial numbers enables Excel to mimic a *date/time* data type.

You're familiar with using Excel's formatting features to display numbers on a worksheet with a fixed number of decimal places (simulating the *Decimal* data type) or as whole numbers with no decimal places (simulating the *Integer* data type).

Common Excel data type problems

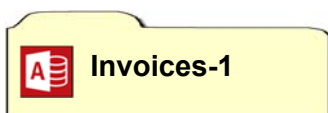
Excel's lack of a *Currency* or *Decimal* data type often causes rounding errors (sometimes referred to as the “penny rounding problem”). The sidebar discusses this problem and its solution. This isn't a problem with SQL Server and Access.

Excel's lack of a *Date* data type means that Excel is unable to work with dates that are before 1900. The SQL Server *Date* data type can work with dates as old as the year 1,000 AD.

You can see that there must often be problems trying to convert values from other data sources to make them usable by Excel.

Data type conversion

The *Get & Transform* tool supports eleven data types (shown in the facing page sidebar). In order to be able to import data from sources such as SQL Server, Access and Excel, *Get & Transform* must convert the source data into one of the data types it supports.



It is often possible for *Get & Transform* to guess the correct data type to convert the source data into. For example, if numeric values had no decimal places, *Get & Transform* might guess that the *Whole Number* data type was most appropriate.

Get & Transform will often play it safe and convert most data into the *Text* data type. You then need to let *Get & Transform* know the correct data type to use.

- 1 Open a new blank workbook.
- 2 Create a *Get & Transform* query to extract the values from the *Invoice* table in the *Invoices-1* Access database.
 1. Click: Data→Get & Transform→New Query→From Database→From Microsoft Access Database.
 2. Navigate to the *Invoices-1* Access file in your sample files folder and open it.
 3. Select the *Invoice* table and then click the *Edit* button.

3 Remove the *InvoiceID* column.

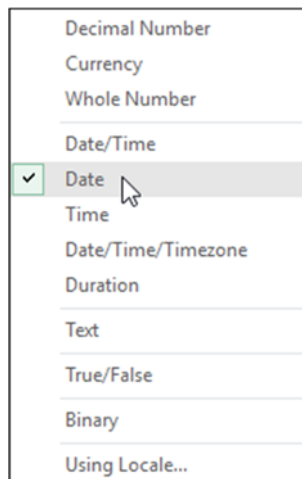
You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.

4 Correct the *InvoiceDate* and *InvoiceTime* data types.

You can immediately see that the *InvoiceDate* and *InvoiceTime* columns have the wrong data type.

| InvoiceDate | InvoiceTime |
|-------------------|-------------------|
| 02/01/2016 00:00: | 30/12/1899 13:40: |
| 02/01/2016 00:00: | 30/12/1899 13:40: |
| 02/01/2016 00:00: | 30/12/1899 13:40: |
| 02/01/2016 00:00: | 30/12/1899 13:40: |

1. Right-click on the *InvoiceDate* column header.
2. Hover the mouse cursor over: *Change Type* from the shortcut menu.
3. A list of data types appears with the *Date/Time* data type checked.
4. Click: *Date* to change the data type to *Date*.
5. Repeat this process for the *InvoiceTime* field, changing the data type to *Time*.



5 Inspect and correct the other data types.

You can inspect and correct the data types that *Get & Transform* has applied for the other fields using the same technique as in the last step.

You can see that *Get & Transform* is confused with the *InvoiceNumber* field (identified as text) and the *InvoiceQty* field (that it does not seem to have applied any data type to).

Apply the *Whole Number* data type to the *Invoice Number* and *InvoiceQty* fields.

6 Close the query editor and load the query result to a table that begins in cell A1 of a new worksheet.

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.

7 Save your work as *Invoices-2*.

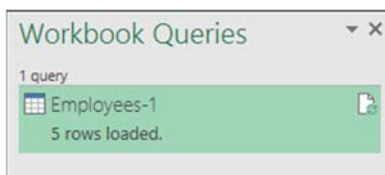
Lesson 11-7: Understand steps and PQFL

Every transformation that you apply in *Get & Transform* generates a formula that is written in the *Power Query Formula Language* (PQFL). A PQFL formula looks very much like an Excel formula. Unlike Excel formulas, PQFL language formulas are case sensitive.

Get & Transform creates a PQFL formula every time you apply a transformation action.

1 Open *Phone List-1* from your sample files folder.

This is the workbook that you created in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.



Notice that the *Workbook Queries* task pane is shown on the right hand side of the worksheet.

If you don't see this task pane, you may have accidentally closed it.

To bring it back:

Click: Data→Get & Transform→Show Queries.

2 Open the query in the query editor.

Double-click on the *Employees-1* query (visible in the *Workbook Queries* task pane).

The *Query Editor* opens, showing the rows returned by the query.

Notice that there is a *Query Settings* task pane on the right-hand side of the query editor. This should look the same as (or similar to) the one shown in the sidebar.

In the *Applied Steps* list you can see all of the transformations that you applied to this query in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.

Each of the *Applied Steps* contains a PQFL formula.

3 Inspect the PQFL formula for the *Filtered Rows* step.

You may remember that in: *Lesson 11-4: Move, remove, rename, filter and sort columns*, the last transformation you specified was to filter the list to only show USA employees.

This created the *Filtered Rows* step.

1. If the formula bar is not visible check: View→Show→Formula Bar.
2. Click on the *Filtered Rows* step in the *Query Settings* task pane. Observe the PQFL formula in the formula bar.

```
fx = Table.SelectRows("#Sorted Rows", each ([Country] = "USA"))
```

Even without knowing PQFL, you can probably guess broadly how this formula works.

4. Manually edit the PQFL formula for the *Filtered Rows* step so that UK employees are shown instead of USA employees.

Phone List-1

trivia

What's in a name?

The Power Query Formula Language (PQFL)

As you are aware, *Get & Transform* is a re-branding of a product that was originally known as *Power Query*.

The old name lives on in the name of the query language.

The M Language

When PQFL was being developed, Microsoft's original internal code name for the language was "M Language".

The letter *M* comes from the common Business Intelligence term: *Mashup*.

A *Data Mashup* simply means combining data from different data sources.

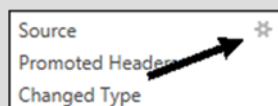
Much of the PQFL help refers to the language in a rather long-winded way as:

The Power Query Formula Language (informally known as "M").

note

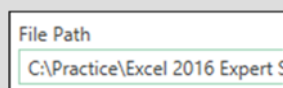
Some steps have arguments

Notice that some steps have a small gear wheel icon next to them, such as the *Source* step shown below:



When you see this icon, you know that the step has arguments that you can edit.

If you click on the icon for the *Source* step, you'll find that you are able to edit the file location of the original CSV file.



Where you see "USA" in the formula bar, change the text to read "UK" and press the <Enter> key.

The grid changes to show UK employees.

| | Country | Title | First Name | Last Name | Job Description | Home Phone |
|---|---------|-------|------------|-----------|----------------------|---------------|
| 1 | UK | Mr. | Steven | Buchanan | Sales Manager | (71) 555-4848 |
| 2 | UK | Ms. | Anne | Dodsworth | Sales Representative | (71) 555-4444 |

Of course, in the real world it would be easier to simply change the filter using the filter button next to *Country*.



This does, however, illustrate the link between the things you are doing using the ribbon and grid, and the PQFL formula that is generated to implement each of your actions.

5 Step through the *Applied Steps* one by one.

1. Click on the *Source* action in the *Query Settings* task pane.

This was one of two actions that *Get & Transform* automatically created when you first opened the CSV file. You can see the original CSV file just after import. Notice that the column headers have not yet been named.

2. Click on the *Promoted Headers* action.

This action was also automatically created when you first opened the CSV file.

Get & Transform intelligently guessed that the first row of the CSV file contained column headers. The term *Promoted* means that the first row is used to define the column names.

3. Click on each of the other actions in turn and watch your data transform as each transformation is applied.

6 Delete the *Sorted Rows* step.

1. You may remember that one of the transformations you applied was to sort the data in A-Z order, first by *Country* and then by *Last Name*.

If you delete the *Sorted Rows* step, the data will be listed in the order that it originally appeared in the CSV file.

2. Click on the *Sorted Rows* step to select it.

Notice that a delete icon has appeared on the left-hand side of the step.



3. Click the *Delete* icon.
4. Click the *Delete* button to confirm that you wish to delete.

The step is removed.

5. Click on the last *Filtered Rows* step.

Notice that the *Last Name* column is no longer sorted in alphabetical order.

7 Close the query editor (click *Keep* if prompted)

8 Save your work as *Phone List-2*

Lesson 11-8: Remove empty, error and top and bottom rows

While you can freely remove columns from a query, you need to use different techniques when removing rows.

When you simply need a subset of the rows in the source data, this is normally done by applying a filter.

Sometimes data contains error rows, or top and bottom rows, that need to be removed.

- 1 Open a new blank workbook.
- 2 Create a *Get & Transform* query to extract the values from the *Gold-1* CSV file in your sample files folder.

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.

This file contains average monthly spot price of gold (in US dollars) for the period January 2006 to December 2015.

- 3 Use the values in the first row as column headers.

In: *Lesson 11-4: Move, remove, rename, filter and sort columns*, you worked with a well-structured CSV file and *Get & Transform* was able to automatically detect that the first row should be used as column headers.

The first rows in this CSV file are a little harder to understand:

| | Column1 | Column2 |
|---|------------|---------------|
| 1 | Date | Average Price |
| 2 | Month/Year | in US Dollars |
| 3 | Jan 2006 | 548.1404 |
| 4 | Feb 2006 | 554.2743478 |

You can see that the column header information is contained in both rows 1 and 2.

Click: Home→Transform→Use First Row as Headers.

This solves part of the problem, as the values in row 1 are now used as column headers.

| | Date | Average Price |
|---|------------|---------------|
| 1 | Month/Year | in US Dollars |
| 2 | Jan 2006 | 548.1404 |
| 3 | Feb 2006 | 554.2743478 |

- 4 Remove row 1.

You'll often find one or more rows of unwanted header or footer data at the top or bottom of an imported data table.

Get & Transform provides a fast way to remove it.

1. Click: Home→Reduce Rows→Remove Rows→Remove Top Rows.

The *Remove Top Rows* dialog appears.

Gold-1 (CSV)

2. Type: **1** into the *Number of rows* box.
3. Click the *OK* button.

5 Change the data types to appropriate values.

You learned about data types in: *Lesson 11-6: Specify data types.*

1. Right-click on the *Average Price* column header.
2. Click: *Change Type*→*Currency*.
3. Right-click on the *Month/Year* column header.
4. Click: *Change Type* from the shortcut menu.

Notice that the data type is currently set to *Text*.

5. Click: *Date* to change the data type.

Notice that two interesting things have happened.

| | Date | Average Price |
|----|------------|---------------|
| 11 | 01/11/2006 | 625.9192 |
| 12 | 01/12/2006 | 628.3808 |
| 13 | Error | 603.0572 |
| 14 | 01/01/2007 | 628.2828 |

Because there is no information to suggest the correct day of the month, *Get & Transform* has guessed that an appropriate date might be the first day of each month. Later, in: *Lesson 11-10: Transform date and time columns*, you'll find out how you could choose a different day of the month.

There were some values in this column (for example the words: *Average 2007*) that could not be converted to dates. In these cases, Excel is showing the word *Error* in place of a date.

6 Remove error rows.

Click: *Home*→*Reduce Rows*→*Remove Errors*.

The error rows are removed from the grid.

7 Close the query editor and load the query result to a table in cell A1 of the worksheet (there is only one worksheet in this workbook).

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns.*

| | A | B | C |
|---|------------|---------------|---|
| 1 | Date | Average Price | |
| 2 | 01/01/2006 | 548.1404 | |
| 3 | 01/02/2006 | 554.2743 | |
| 4 | 01/03/2006 | 555.3573 | |
| 5 | 01/04/2006 | 609.2191 | |

8 Save your work as *Gold-2*.

Lesson 11-9: Understand and work with null values

When you see a blank cell in a worksheet, you might interpret a numeric field as being zero. Computers (and databases) are more sophisticated and support a concept called null. Null is not a value but is the absence of a value.

Empty cells often do not contain null values, but are left blank because most readers will assume a value that is not stated.

Here's an example:

With nulls

| | A | B | C |
|---|------------|-----------|-------------|
| 1 | First Name | Last Name | Occupation |
| 2 | John | Smith | Electrician |
| 3 | Bill | Jones | |
| 4 | Arthur | Williams | Bricklayer |

Without nulls

| | A | B | C |
|---|------------|-----------|-------------|
| 1 | First Name | Last Name | Occupation |
| 2 | John | Smith | Electrician |
| 3 | Bill | Jones | Unemployed |
| 4 | Arthur | Williams | Bricklayer |

What is the status of Bill Jones in the *With nulls* example above? Is he unemployed, or do you simply not know? Excel would consider the blank cell as having a null value (in other words, you don't know whether Bill has an occupation or not).

Ideally the nulls should be replaced with words similar to: *Not Stated*, *Withheld*, *Unemployed*, *Retired* or *Unknown*.

The issue is far more common (and can cause more errors) with numeric fields. Consider this example:

With nulls

| | A | B |
|---|---------|----------|
| 1 | Item | Quantity |
| 2 | Apples | 25 |
| 3 | Oranges | |
| 4 | Pears | 35 |
| 5 | | |

Without nulls

| | A | B |
|---|---------|----------|
| 1 | Item | Quantity |
| 2 | Apples | 25 |
| 3 | Oranges | 0 |
| 4 | Pears | 35 |
| 5 | | |

In the *With nulls* example above, a human might assume that we have zero oranges. Excel would just note the absence of any meaningful data (in other words, a null value) and assume that you have no idea how many oranges you have.

Null values will also affect many of Excel's functions.

In the *With nulls* example above, the formula `=AVERAGE(B2:B4)` returns a value of 30.

In the *Without nulls* example above, the same formula returns a value of 20. A **COUNT** function would also return different results (2 and 3).

The above two examples both illustrate the concept of false null values.

Sometimes (though rarely) you may have real null values (see sidebar for examples). In these special cases it may be better to leave the cells empty

1 Open a new blank workbook.

note

Examples of real null values

1. A questionnaire may request a date of birth that some respondents refuse to supply.

Clearly the respondent does not have a date of birth, you simply don't know what it is. The empty cell contains a real null value.

2. You record the temperature every day in degrees Celsius.

Sometimes you forget to record the temperature.

Clearly a value of 0 degrees is valid, so it would be wrong to replace all of the empty cells with a zero.

The empty cells contain real null values. There was a temperature on those days but you don't know what it was.

Olympics-1 (CSV)

2 Create a *Get & Transform* query to extract the values from the *Olympics-1* CSV file in your sample files folder.

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns.*

| | Rank | Country | Continent | Summer Olympics | Years (Summer Olympics) | Winter Olympics | Years (Winter Olympics) |
|----|------|---------------------|-----------|-----------------|-------------------------|-----------------|-------------------------|
| 10 | 8 | Soviet Union/Russia | Europe | | 1 1980 | | 1 2014 |
| 11 | 8 | Greece | Europe | | 2 1896, 2004 | | null |
| 12 | 8 | Australia | Oceania | | 2 1956, 2000 | | null |
| 13 | 8 | Norway | Europe | | null | | 2 1952, 1994 |
| 14 | 8 | Austria | Europe | | null | | 2 1964, 1976 |

note

Why do the Year columns not contain null values?

Get & Transform will represent a null textual value as an *Empty String*.

An empty string can be thought of as a textual value that does not contain any actual text.

For example, the Excel formula:

`=“Extra” & “” & “terrestrial”`

... is actually concatenating an empty string in between the words *Extra* and *terrestrial*.

If you typed the above formula into an Excel cell, the result would be the single word:

Extraterrestrial

Database designers can argue all day about whether a blank textual value should be represented as an empty string or a null value.

In the context of data that will be used by Excel there isn't any real difference, as Excel will regard a cell containing an empty string to be the same as any other empty cell.

In this lesson's data you could, of course, replace the empty text values with meaningful text such as: *Not Applicable*. The resulting data would then produce different results with Excel functions such as `COUNTA` and `COUNTBLANK`.

This data contains a list of countries that have hosted one or more Olympic Games.

Notice that there are several null values.

In the above grab you can see that Norway and Austria have a null value in the *Summer Olympics* column.

These are false null values. You actually know that Norway and Austria have never hosted a Summer Olympics. The value should more properly be stated as zero.

Notice also that there are empty values in the *Years* columns when a country has never hosted the Olympics. This is because *Get & Transform* has assigned the *Text* data type to these columns (see sidebar).

3 Correct null values.

It is generally a good objective to remove false null values from data (unless you really intend the value to signify the absence of data).

In this case, the nulls clearly signify zero values

1. Click on the *Summer Olympics* column header to select the entire column.
2. Hold down the **<Ctrl>** key and then click on the *Winter Olympics* column header.

Both columns are now selected.

3. Click: Home → Transform → Replace Values.

The *Replace Values* dialog appears.

4. Type **null** into the *Value To Find* box.
5. Type **0** into the *Replace With* box.
6. Click the *OK* button.

The null values are replaced with zeroes.

4 Close the query editor and load the query result to a table in cell A1 of the existing worksheet (there is only one worksheet in this workbook).

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns.*

5 Save your work as *Olympics-2*.

note**How can I return day and month names instead of numbers?**

When you return day and month numbers instead of names you are able to sort by week or month number and have the days or months appear in the correct order.

This why the Get & Transform *Day of Week* and *Month* transformations return numbers instead of text.

For example:

1=January, 2=February
And
0=Monday, 1=Tuesday

Sometimes, for display purposes, you'd rather have textual month and day values such as:

Jan, Feb, Mar
and
Mon, Tue, Wed

It is possible to do this using a *Custom Column Formula* inside a custom column.

You'll learn about custom columns later in: *Lesson 11-12: Add a custom calculated column.*

There's a sidebar in this lesson that shows how to use the *ToText* function to convert a date value into a textual representation of a month or day.

Lesson 11-10: Transform date and time columns

When source data contains transactional information (such as a listing of individual sales), you will often want to analyze total sales by *Quarter*, *Month* or *Year*.

As you learned in: *Lesson 8-19: Group by date*, regular pivot tables have always had the ability to group by *Seconds*, *Minutes*, *Hours*, *Days*, *Months*, *Quarters* and *Years*. In Excel 2016, this ability has also been added to OLAP pivot tables (a hugely useful improvement).

This means that you probably won't have to create *Second*, *Minute*, *Hour*, *Day*, *Month*, *Quarter* and *Year* fields within your *Get & Transform* queries in order to address most business requirements.

While pivot tables can group by these seven different date and time fields, *Get & Transform* supports sixteen additional date and time grouping types. For example:

- You might want to group by the *Week of Year*, to compare this week's sales with the same week's sales last year.
- You might want to group by *Day of Week*, to discover the average sales figure for *Monday*.

In this lesson you'll learn how to add a new *Week of Year* grouping field to a *Get & Transform* query and then use it to analyze sales by week in an OLAP pivot table.

- 1 Open a new blank workbook.
- 2 Create a *Get & Transform* query to extract the values from the *DVD Sales-1* Excel workbook in your sample files folder.

The *DVD Sales-1* workbook contains a single Excel table called *DVDSales*.

1. Click: Data→Get & Transform→New Query→From File→From Workbook.
2. Navigate to and select the *DVD Sales-1* Excel workbook file in your sample files folder.
3. Click the *Import* button.
4. Select the *DVDSales* table.
5. Click the *Edit* button to open the query in the *Get & Transform* editor.

| | No | Date | Customer | Empl... |
|---|--------|------------|---------------------|------------|
| 1 | 136438 | 02/10/2014 | Silver Screen Video | Lee, Frank |
| 2 | 136438 | 02/10/2014 | Silver Screen Video | Lee, Frank |
| 3 | 136438 | 02/10/2014 | Silver Screen Video | Lee, Frank |

- 3 Create a duplicate *Date* column named *Week*.

When you transform a date column you'll usually want to retain the original date column.

1. Click anywhere in the *Date* column.

DVD Sales-1

2. Click: Add Column→General→Duplicate Column.
A duplicate column is added, named: *Date-Copy*.
3. Double-click in the column header and type: **Week**
4. Press the <Enter> key.

| Total | Week |
|--------|------|
| 106.13 | 40 |
| 54.56 | 41 |
| 140.31 | 41 |
| 202.36 | 41 |
| 50.01 | 41 |

- 4 Transform the value in the *Week* column so that it displays the week number of each transaction.

Click: Transform→Date & Time Column→Date→Week→Week of Year.

The values in the *Week* column change to show the week number for each transaction.

- 5 Load the query results directly into the data model.

This time you will output the query result directly to the data model instead of a worksheet table.

This results in a more compact workbook, both in file size (when you save it) and in memory usage (when it is open). This would also be the only option open to you when working with *big data* (data with over 1,048,576 rows).

1. Click: File→Close & Load To...
The *Load To* dialog appears.
2. Click the *Only Create Connection* option button.
3. Check the *Add this data to the Data Model* check box.
4. Click the *Load* button.

You are returned to the worksheet.

- 6 Create an OLAP pivot table from the data model.

You learned how to do this in: *Lesson 9-16: Create an OLAP pivot table using a many-to-many relationship*.

- 7 Use the OLAP pivot table to show sales by *Customer* for each *Year* and *Week*.

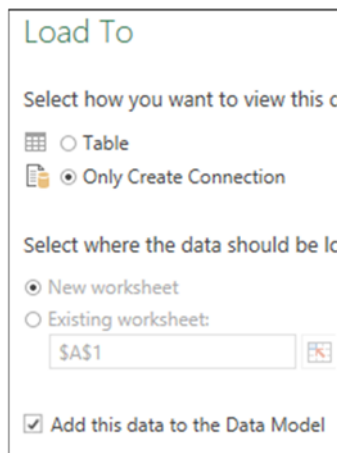
Do this using the same technique you learned in: *Lesson 8-19: Group by date*.

The field settings are:

| FILTERS | | COLUMNS | |
|----------|--|--------------|------|
| | | Date (Year) | Week |
| ROWS | | VALUES | |
| Customer | | Sum of Total | |

| | A | B | C | D | E |
|---|----------------------------------|---------------|--------|--------|--------|
| 1 | Sum of Total | Column Labels | | | |
| 2 | | 2014 | | | |
| 3 | Row Labels | 40 | 41 | 42 | 43 |
| 4 | Addison-Freelander Screen Agency | | | | 289.69 |
| 5 | AV Supplies | 639.34 | | | 514.31 |
| 6 | Box Office Supplies | 667.27 | 784.73 | 400.89 | |

- 8 Save your work as *DVD Sales-2*



Lesson 11-11: Transform number columns

You will often find a need to transform the scale and rounding of number columns.

For example, in this table, column C shows numbers using the *Thousand* number scale and percentages as floating point numbers:

| | A | B | C | D |
|---|----------|----------|-------------------------|-------------|
| 3 | From | To | Individuals (thousands) | % In Group |
| 4 | 0 | 2,500.00 | 12,686 | 5.996350951 |
| 5 | 2,500.00 | 4,999.00 | 7,202 | 3.404203023 |
| 6 | 5,000.00 | 7,499.00 | 9,645 | 4.558947259 |

You might prefer to return the *Individuals* values without scaling and the *% In Group* values rounded to four decimal places, like this:

| | A | B | C | D |
|---|------|------|-------------|------------|
| 1 | From | To | Individuals | % In Group |
| 2 | 0 | 2500 | 12686000 | 5.9964 |
| 3 | 2500 | 4999 | 7202000 | 3.4042 |
| 4 | 5000 | 7499 | 9645000 | 4.5589 |

You could, of course, do this inside Excel.

By transforming the values within *Get & Transform* you will only need a single linked table within the workbook to correctly display values. You are then able to right-click → refresh the table to update any future changes that may occur in the data source.

- 1 Open a new blank workbook.
- 2 Create a *Get & Transform* query to extract the values from the *IncomeDistribution* table in the *USA Income-1* Excel workbook in your sample files folder.

You learned how to do this in: *Lesson 11-10: Transform date and time columns*.

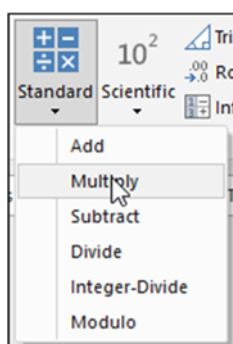
This workbook contains details of individual incomes in the United States in 2010:

| | From | To | Individuals (thousands) | % In Group |
|----|-------|-------|-------------------------|-------------|
| 21 | 50000 | 52499 | 6320 | 2.987303958 |
| 22 | 52500 | 54999 | 2186 | 1.033266844 |
| 23 | 55000 | 57499 | 3455 | 1.633091009 |
| 24 | 57500 | 59999 | 1876 | 0.886737694 |

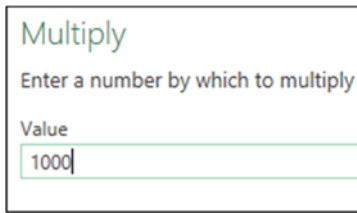
You can see that 6,320,000 Americans had an income between \$50,000 and \$52,499, which constituted 2.987303958% of all individuals.

- 3 Change the scale of the *Individuals (thousands)* column to show the number without thousand scaling.

1. Click anywhere in the *Individuals (thousands)* column.
2. Click: Transform → Number Column → Standard → Multiply.
The *Multiply* dialog appears.



USA Income-1



3. Type: **1000** into the *Value* box.
4. Click the *OK* button.

All values in the *Individuals (thousands)* column are transformed:

| | From | To | Individuals (thousands) | % In Group |
|---|------|------|-------------------------|-------------|
| 1 | 0 | 2500 | 12686000 | 5.996350951 |
| 2 | 2500 | 4999 | 7202000 | 3.404203023 |
| 3 | 5000 | 7499 | 9645000 | 4.558947259 |

- 4 Change the column header of the *Individuals (thousands)* column to: **Individuals**

1. Double-click on the *Individuals (thousands)* column header.
2. Replace the existing text with: **Individuals**
3. Press the <Enter> key.

- 5 Round the values in the *% In Group* column so that values are rounded to four decimal places.

1. Click anywhere in the *% In Group* column.
2. Click: Transform→Number columns→Rounding→Round...

The *Round* dialog appears.

3. Type **4** into the *Decimal Places* box.
4. Click the *OK* button.

All values in the *% In Group* column are rounded to four decimal places:

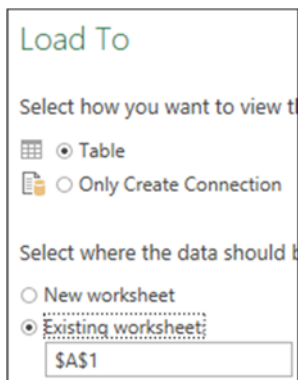
| | From | To | Individuals | % In Group |
|---|------|------|-------------|------------|
| 1 | 0 | 2500 | 12686000 | 5.9964 |
| 2 | 2500 | 4999 | 7202000 | 3.4042 |
| 3 | 5000 | 7499 | 9645000 | 4.5589 |

- 6 Close the query editor and load the query result to a table beginning in cell A1 of the existing worksheet (there is only one worksheet in this workbook).

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns.*

| | A | B | C | D |
|---|------|------|-------------|------------|
| 1 | From | To | Individuals | % In Group |
| 2 | 0 | 2500 | 12686000 | 5.9964 |
| 3 | 2500 | 4999 | 7202000 | 3.4042 |
| 4 | 5000 | 7499 | 9645000 | 4.5589 |

- 7 Save your work as *USA Income-2*.



Lesson 11-12: Add a custom calculated column

One of the shortcomings of an OLAP pivot table is its inability to add calculated columns.

When your OLAP pivot table uses a *Get & Transform* query as its data source (either directly by loading to the data model, or indirectly by loading to an interim Excel table), this restriction effectively vanishes.

In a few clicks you can add a calculated column to the *Get & Transform* query and then refresh your pivot table to show the new calculated field.

- 1 Open a new blank workbook.
- 2 Create a *Get & Transform* query to extract the values from the *Sales* table in the *Bonus-1* Excel workbook in your sample files folder.

You learned how to do this in: *Lesson 11-10: Transform date and time columns*.

This workbook contains details of individual sales generated by a DVD wholesaler:

| | Order | Order | Customer | Employee | Title | Genre | Qty | Total |
|---|--------|------------|---------------------|--------------------|---------------------------------------|-----------|-----|--------|
| 1 | 136438 | 02/10/2014 | Silver Screen Video | Lee, Frank | Lawrence of Arabia | Biography | 15 | 122.76 |
| 2 | 136438 | 02/10/2014 | Silver Screen Video | Lee, Frank | The Discreet Charm of the Bourgeoisie | Comedy | 9 | 67.46 |
| 3 | 136438 | 02/10/2014 | Silver Screen Video | Lee, Frank | Berlin Alexanderplatz | Drama | 25 | 250.6 |
| 4 | 136438 | 02/10/2014 | Silver Screen Video | Lee, Frank | Gone With The Wind | Drama | 14 | 107.72 |
| 5 | 136439 | 03/10/2014 | Cinefocus DVD | Diamond, Elizabeth | Mouchette | Drama | 5 | 31.77 |

You can see that *Frank Lee* sold order 136438 to *Silver Screen Video*. The order was for four different DVD titles.

By adding the *Qty* and *Total* field values in rows 1-4 you can see that Frank sold 63 DVDs on this order, for a total price of \$548.54.

In this lesson you will calculate a monthly bonus for each employee. The bonus will be calculated as 3% of the total sale value.

- 3 Add a custom column to the *Get & Transform* query named: **Bonus** that will calculate 3% of the value shown in the *Total* column.
 1. Click: Add Column→General→Add Custom Column.
 2. The *Add Custom Column* dialog appears.
 3. Type: **Bonus** into the *New column name* box.
 4. Click inside the *Custom column formula* box.
 5. Click the *Total* field in the right-hand list.
 6. Click the <<*Insert* button to add the *Total* field to the *Custom column formula* window.

Notice that the field name is shown inside square brackets. This is PQFL syntax for a field name.

Bonus-1

note

The Date.ToText PQFL function

In: *Lesson 11-7: Understand steps and PQFL*, you learned that there is a powerful formula language (PQFL) implementing every action you perform.

Microsoft have engineered the user interface to address most users' needs without having to learn PQFL.

Even without understanding PQFL in depth there is a PQFL function that is easy to use when creating custom queries from date values.

Create a textual day-of-the-week or month-of-the-year from a date

In: *Lesson 11-10: Transform date and time columns*, you learned how to transform a date value.

When you transform a date value into the day of the week using the technique shown in: *Lesson 11-10: Transform date and time columns*, a numerical value is returned (for example, 0 = Monday and 6 = Sunday).

If you wanted to return more comprehensible textual days (from a column called *Date*), you could use the *Custom Column Formula*:

```
=Date.ToText([Date],"ddd")
```

This will return dates in the format: *Mon, Tue, Wed...*

You can use a similar technique to return month names as text rather than numbers with the *Custom Column Formula*:

```
=Date.ToText([Date],"MMM")
```

This will return the month name in the format: *Jan, Feb, Mar...*

You can use different format strings to provide many different textual representations of dates.

- Click to the right of the *[Total]* field and Type: ***0.03** to complete the formula.

Multiplying the *Total* field by 0.03 will calculate 3% of the field's value.

| |
|------------------------|
| New column name |
| Bonus |
| Custom column formula: |
| = [Total]*0.03 |

- Click the *OK* button.

A new custom (calculated) column called *Bonus* is added to the query and shows values that are 3% of the value in the *Total* column:

| Employ... | Title | Genre | Q. | Total | Bonus |
|------------|-------------------------|-----------|----|--------|--------|
| Lee, Frank | Lawrence of Arabia | Biography | 15 | 122.76 | 3.6828 |
| Lee, Frank | The Discreet Charm of t | Comedy | 9 | 67.46 | 2.0238 |
| Lee, Frank | Berlin Alexanderplatz | Drama | 25 | 250.6 | 7.518 |

You can see that *Frank Lee* made \$3.68 bonus on his sale of 15 copies of *Lawrence of Arabia*.

- Set the data type of the *Bonus* field to *Currency*.
You learned how to do this in: *Lesson 11-6: Specify data types*.
- Load the query results directly into the data model and then create an OLAP pivot table from the data model.
You did this in: *Lesson 11-10: Transform date and time columns*.
- Select fields in the OLAP pivot table that will show monthly *Bonus* earned for each *Employee*.

Do this using the same technique you learned in: *Lesson 8-19: Group by date*.

The field settings are:

| | |
|----------------|----------------------|
| FILTERS | COLUMNS |
| | Order Date (Year) ▼ |
| | Order Date (Month) ▼ |
| ROWS | VALUES |
| Employee ▼ | Sum of Bonus ▼ |

You can now see the monthly bonus earned by each Employee for each year and month in the period Oct 2014 to Mar 2016.

| | A | B | C | D | E |
|---|----------------|-----------------|---------|---------|------------|
| 1 | Sum of Bonus | Column Labels ▼ | | | |
| 2 | | 2014 | | | 2014 Total |
| 3 | Row Labels ▼ | Oct | Nov | Dec | |
| 4 | Anderson, Jane | | \$45.78 | \$0.76 | \$46.54 |
| 5 | Armstrong, Dan | | \$43.62 | \$32.19 | \$75.80 |
| 6 | Ashe, Lucille | \$20.26 | \$8.45 | \$18.73 | \$47.43 |

Note that the currency symbol shown in your own pivot table may be different depending upon the Windows locale settings.

- Save your work as *Bonus-2*.

Lesson 11-13: Create an aggregated data query

Normally you will use *Get & Transform* to transform data and then use a pivot table to aggregate (summarize) the query result.

There may be some special circumstances where you would like the query to aggregate values *before* sending the query results to Excel.

This lesson will bring together many of the skills you have learned so far in this session.

- 1 Open a new blank workbook.
- 2 Create a *Get & Transform* query to extract the values from the *Products and Categories* view in the *Inventory-1* Access database in your sample files folder.

You learned about views in: *Lesson 9-5: Efficiently import data using a view*.

You learned how to connect a *Get & Transform* query to an Access database in: *Lesson 11-6: Specify data types*.

This view returns inventory details. Each inventory item has been allocated a *Category*.

| | ProductName | CategoryName | QuantityPerUnit | UnitPrice | UnitsInStock | UnitsOnOrder | ReorderLevel | Discontinued |
|---|--------------------|--------------|--------------------|-----------|--------------|--------------|--------------|--------------|
| 1 | Chai | Beverages | 10 boxes x 20 bags | 18 | 39 | 0 | 10 | FALSE |
| 2 | Chang | Beverages | 24 - 12 oz bottles | 19 | 17 | 40 | 25 | FALSE |
| 3 | Guaraná Fantástica | Beverages | 12 - 355 ml cans | 4.5 | 20 | 0 | 0 | TRUE |

Your challenge in this lesson will be to create a *Get & Transform* query that will return aggregated (summarized) values for each *Category*.

For each *Category*, the query needs to report the total *Units in Stock*, *Units on Order*, *Inventory Value* and *Projected Inventory Value* (the value that would result if all goods on order had already arrived).

- 3 Remove the *ReorderLevel*, *QuantityPerUnit* and *Discontinued* fields.
- 4 Set the data type of the *UnitPrice* column to *Currency* (if it isn't already set to *Currency*) and the data type of the *UnitsInStock* and *UnitsOnOrder* columns to *Whole Number*.
- 5 Add a custom column named *Inventory Value* that contains a value equal to the *UnitPrice* field multiplied by the *UnitsInStock* field.

You learned how to do this in: *Lesson 11-12: Add a custom calculated column*.

Add Custom Column

New column name
Inventory Value

Custom column formula:
=[UnitPrice]*[UnitsInStock]



Add Custom Column

New column name

Custom column formula:

note

Advantages of an aggregated query over a pivot table

Automatic refresh capability

You can schedule automatic refreshes with an aggregated query.

In: *Lesson 11-3: Understand queries and connections*, you learned how to automatically refresh a table linked to a *Get & Transform* query every minute.

It isn't possible to automatically refresh data in a pivot table.

Small file and memory size

A worksheet table does not have any associated PivotTable data cache or OLAP cube. This results in a much smaller small file size and memory requirement.

6 Add a custom column named *Projected Inventory Value* that contains a value equal to the *Inventory Value* field added to the *UnitPrice* field multiplied by the *UnitsOnOrder* field.

You learned how to do this in: *Lesson 11-12: Add a custom calculated column*.

7 Set the data type of both custom columns to *Currency*.

You learned how to do this in: *Lesson 11-6: Specify data types*.

8 Group the query by *CategoryName* to show aggregated values for the total of the three numeric columns.

1. Click: Transform → Table → Group By.

The *Group By* dialog appears.

2. Remove any existing *Group by* fields by clicking the minus button next to them.

3. Click the + button and select *CategoryName* as the *Group by* field.

Note that you can have more than one *Group by* field but, in this case, only one is needed.

4. Type **Units in Stock** into the *New column name* box.

5. Select: *Sum* from the *Operation* drop-down list.

6. Select *UnitsInStock* from the *Column* drop-down list.

7. Click the + button on the right of the dialog to add a second group field for *UnitsOnOrder*.

8. Add two more fields so that your dialog looks like this:

Group by +

CategoryName -

| New column name | Operation | Column | |
|---------------------------|-----------|---------------------------|---|
| Units in Stock | Sum | UnitsInStock | - |
| Units on Order | Sum | UnitsOnOrder | - |
| Inventory Value | Sum | Inventory Value | - |
| Projected Inventory Value | Sum | Projected Inventory Value | - |

9. Click the *OK* button.

9 Close the query editor and load the query result to a table in cell A1 of the existing worksheet (there is only one worksheet in this workbook).

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.

| | A | B | C | D | E |
|---|--------------|----------------|----------------|-----------------|---------------------------|
| 1 | CategoryName | Units in Stock | Units on Order | Inventory Value | Projected Inventory Value |
| 2 | Beverages | 559 | 60 | 12,480.25 | 13,850.25 |
| 3 | Condiments | 507 | 170 | 12,023.55 | 14,423.55 |
| 4 | Confections | 386 | 180 | 10,392.20 | 13,009.70 |

10 Save your work as *Inventory-2*.

Lesson 11-14: Unpivot aggregated data

Recognizing pivoted data

You'll often find data showing rolled-up totals in columns. For example, consider this data:

| | A | B | C | D |
|---|----------------------------------|----------|----------|----------|
| 2 | Customer | Jan | Feb | Mar |
| 3 | Addison-Freelander Screen Agency | 2,195.49 | 1,673.32 | 488.52 |
| 4 | AV Supplies | 1,298.66 | 1,862.95 | 1,792.25 |
| 5 | Box Office Supplies | 795.39 | 3,029.79 | 1,793.12 |

The data shows aggregated monthly sales.

Data in this format is nicely presented for human readers but is not useful when you need to analyze it using an Excel worksheet or pivot table.

Here is the correct format for easy analysis of the above data:

| | A | B | C |
|----|----------------------------------|-------|---------|
| 2 | Customer | Month | Sales |
| 3 | Addison-Freelander Screen Agency | Jan | 2195.49 |
| 4 | Addison-Freelander Screen Agency | Feb | 1673.32 |
| 5 | Addison-Freelander Screen Agency | Mar | 488.52 |
| 6 | AV Supplies | Jan | 1298.66 |
| 7 | AV Supplies | Feb | 1862.95 |
| 8 | AV Supplies | Mar | 1792.25 |
| 9 | Box Office Supplies | Jan | 795.39 |
| 10 | Box Office Supplies | Feb | 3029.79 |
| 11 | Box Office Supplies | Mar | 1793.12 |

This transformation would take a substantial amount of time using traditional Excel tools but takes just three clicks in the *Get & Transform* query editor.

Many more things can be done when the data is transformed into the correct format.

- You can create a pivot table from it.
- You can filter by month.
- You can combine the query result with other data (you'll learn how to do this later, in: *Lesson 11-18: Create a simple two-table merged query*).

- 1 Open a new blank worksheet.
- 2 Create a *Get & Transform* query to extract the values from the *MonthlySales* table in the *Sales by Employee-1* Excel workbook in your sample files folder.

You learned how to do this in: *Lesson 11-10: Transform date and time columns*.

Sales by Employee-1

note

Why are there two Month fields shown in the pivot table field list

When you add the *Month* PivotTable field to the *Columns* box, two different values are shown:



In: *Lesson 11-6: Specify data types*, it was noted that Excel does not support any true *Date* or *Time* data types.

Instead, Excel relies upon formatting numerical values that are regarded as being *date serial numbers*.

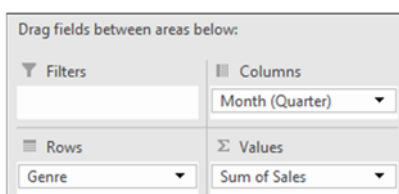
Date serial numbers were covered in depth in: *Lesson 3-7: Understand date serial numbers*.

Get & Transform originally assigned the *Text* data type to the *Month* column.

When you changed the data type of the *Month* column to *Date*, *Get & Transform* assigned the first day of each month to the date values in the column.

The OLAP pivot table was not sure whether you wanted the date values (such as 01/01/15) or month values (such as *Jan*, *Feb*, *Mar*) in the column headers.

For this reason, it helpfully added both *Month (Month)* to show the *Month* name and *Month* (the field name on its own) to show the full date.



This workbook contains monthly total sales figures for each employee.

| | Employee | Genre | Jan-15 | Feb-15 | Mar-15 |
|---|---------------|-----------|--------|--------|--------|
| 1 | Anderson,Jane | Action | 0 | 173.29 | 409.48 |
| 2 | Anderson,Jane | Adventure | 0 | 0 | 0 |
| 3 | Anderson,Jane | Biography | 0 | 0 | 0 |
| 4 | Anderson,Jane | Comedy | 116 | 0 | 777.2 |
| 5 | Anderson,Jane | Crime | 217.12 | 152.22 | 0 |

You can see that this is a classic example of pivoted data. The *Employee* and *Genre* columns don't have a problem but all of the other columns (*Jan-15* to *Dec-15*) need to be unpivoted.

3 Unpivot all of the monthly total columns.

1. Click on the *Jan-15* column header to select it.
2. Hold down the <Shift> key.
3. Click on the *Dec-15* column header.
All monthly columns are now selected.
4. Click: Transform → Any Column → Unpivot Columns.

The columns are unpivoted:

| | Employee | Genre | Attribute | Value |
|---|---------------|--------|-----------|--------|
| 1 | Anderson,Jane | Action | Jan-15 | 0 |
| 2 | Anderson,Jane | Action | Feb-15 | 173.29 |
| 3 | Anderson,Jane | Action | Mar-15 | 409.48 |
| 4 | Anderson,Jane | Action | Apr-15 | 0 |

4 Rename the Attribute column to: Month and the Value column to: Sales

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.

5 Change the data type of the Month column to Date.

You learned how to do this in: *Lesson 11-6: Specify data types*.

6 Load the query results into the data model and create an OLAP pivot table from the data model.

You learned how to do this in: *Lesson 11-10: Transform date and time columns*.

7 Select fields in the OLAP pivot table that will show quarterly Sales for each Genre.

Do this using the same technique you learned in: *Lesson 8-19: Group by date*.

The field settings are shown in the sidebar.

| | A | B | C | D | E | F |
|---|--------------|---------------|----------|----------|----------|-------------|
| 1 | Sum of Sales | Column Labels | | | | |
| 2 | Genre | Qtr1 | Qtr2 | Qtr3 | Qtr4 | Grand Total |
| 3 | Action | 4,992.07 | 3,762.46 | 6,733.32 | 7,607.52 | 23,095.37 |
| 4 | Adventure | 700.07 | 1,446.15 | 1,311.80 | 1,594.27 | 5,052.29 |
| 5 | Animation | 775.66 | 567.65 | 782.71 | 472.34 | 2,598.36 |

8 Save your work as: Sales by Employee-2.

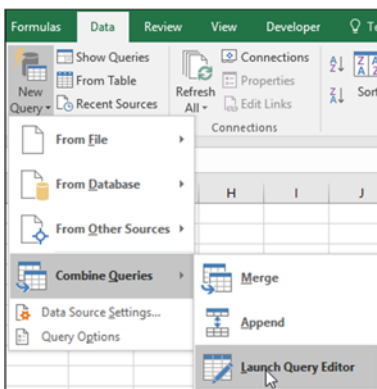
Lesson 11-15: Work with multiple queries

Up until now you've worked with a single query.

Get & Transform has the ability to combine data from multiple queries. You'll learn how to do this in: *Lesson 11-16: Create an append query* and *Lesson 11-18: Create a simple two-table merged query*.

As each query can potentially draw data from a different data source, it is possible to combine data from one or more web queries, CSV files, Excel workbooks and database tables or views.

You could create queries individually (as you have up to now in this session). This lesson will teach you how to work more efficiently by creating multiple queries in a single *Get & Transform* session.



- 1 Open a new blank worksheet.
- 2 Open the Get & Transform query editor.

Up until now you've always selected the data source type before entering the query editor. You could do that in this lesson but, for variety, you'll do things a little differently.

Click: Data→Get & Transform→New Query→Combine Queries→Launch Query Editor.

The empty query editor opens.

- 3 Create four queries named *Confections*, *DairyProduce*, *Beverages* and *Condiments* from the four sample files of the same name in the *Session 11/Append* folder in your sample files folder.

The *Session 11/Append* folder contains three Excel sample files and one CSV sample file. Each of the files contains data with the same structure (the same number of columns and the same column header names in each column).

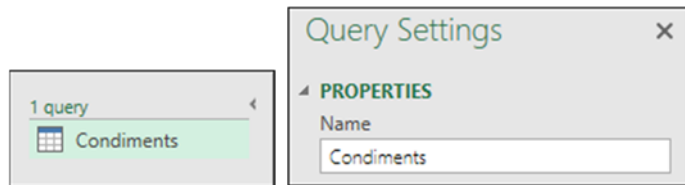
| Name | Type | Size |
|--------------|---|-------|
| Beverages | Microsoft Excel Comma Separated Values File | 1 KB |
| Condiments | Microsoft Excel Worksheet | 10 KB |
| Confections | Microsoft Excel Worksheet | 11 KB |
| DairyProduce | Microsoft Excel Worksheet | 10 KB |

1. Click: Home→New Query→New Source→File→Excel.
2. Navigate to the *Session 11/Append* folder and double-click the *Condiments* workbook.
3. Select the *Condiments* table within the workbook.
4. Click the OK button.

The *Get & Transform* query editor now shows the contents of the *Condiments* table.

Notice that the query editor has automatically named the query *Condiments* and that it has appeared in the query list on the left of the screen.

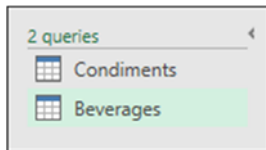
**Set of four files in:
Session 11/Append**



5. Click: Home→New Query→New Source→File→CSV.
6. Navigate to the *Session 11/Append* folder and double-click the *Beverages* CSV file.

The *Get & Transform* query editor now shows the contents of the *Condiments* table.

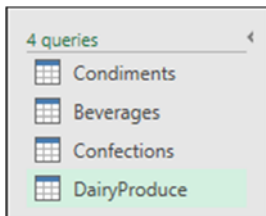
Notice that the query editor has automatically named the query *Beverages* and that it has appeared underneath the *Condiments* query in the query list on the left of the screen.



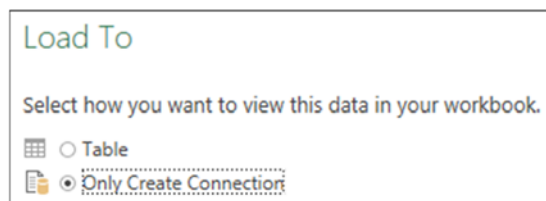
7. Repeat the same process for the *Confections* and *DairyProduce* Excel workbooks.

4 Save the four queries as connection only queries.

The four queries will be used in the next lesson. For the moment you do not want to load the data returned by the queries anywhere.

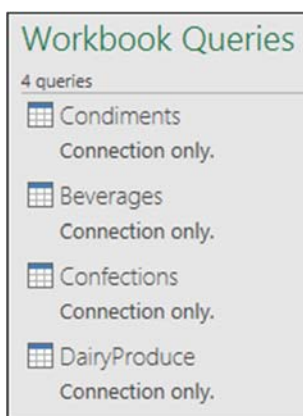


1. Click Home→Close→Close & Load (Drop-down list)→Close & Load To...
2. Click the *Only Create Connection* option button.



3. Click the *Load* button.

You are returned to the empty workbook with four *Connection only* queries shown in the *Workbook Queries* task pane.



5 Save your work as *Multiple Queries-1*.

note

The ability to append multiple tables without editing PQFL code is now available to some Office 365 subscribers

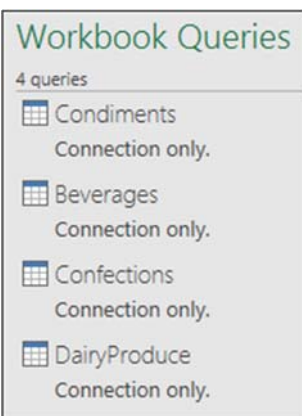
In: *Lesson 1-1: Check that your Excel version is up to date* you learned that subscription purchasers of Excel 2016 (via the Office 365 program) may find new features appearing on their copy of Excel that are not available to users who purchased a perpetual license.

Current channel Office 365 subscribers may find that they now see a new *Three or more tables* option in the *Append* dialog used in this lesson. This improvement was added on May 4th 2016.

If you are an Office 365 subscriber and would like to find out more about detail improvements to Excel 2016, we maintain a list and explanation of all subscriber-only updates to date on our web site at:

<http://forums.excelcentral.com/>

You'll find an *Office 365 Updates* link on this page.



Multiple Queries-1

Lesson 11-16: Create an append query

You will often find a need to append rows of data from different data sources.

In: *Lesson 11-15: Work with multiple queries*, you created four queries that returned data from similar workbooks.

This is the data that was returned from the *Condiments* table in the *Condiments* Excel workbook:

| | A | B | C |
|---|------------------------------|----------------------------|------------|
| 1 | Product Name | Supplier | Category |
| 2 | Aniseed Syrup | Exotic Liquids | Condiments |
| 3 | Chef Anton's Cajun Seasoning | New Orleans Cajun Delights | Condiments |
| 4 | Genen Shouyu | Mayumi's | Condiments |

... and here is the data returned from the *Beverages* CSV file:

| | A | B | C |
|---|------------------|-----------------------------|-----------|
| 1 | Product Name | Supplier | Category |
| 2 | Chai | Exotic Liquids | Beverages |
| 3 | Chang | Exotic Liquids | Beverages |
| 4 | Chartreuse verte | Aux joyeux eccl,siaistiques | Beverages |

Notice that the column headers (field names) are identical and that each column has the same data type.

You'll often find that you have rows of data residing in different data sources that you need to combine into one table that contains all rows of data. You can do this with a *Get & Transform append query*.

If the above two queries were appended, the resulting table would look like this:

| | A | B | C |
|---|------------------------------|-----------------------------|------------|
| 1 | Product Name | Supplier | Category |
| 2 | Aniseed Syrup | Exotic Liquids | Condiments |
| 3 | Chef Anton's Cajun Seasoning | New Orleans Cajun Delights | Condiments |
| 4 | Genen Shouyu | Mayumi's | Condiments |
| 5 | Chai | Exotic Liquids | Beverages |
| 6 | Chang | Exotic Liquids | Beverages |
| 7 | Chartreuse verte | Aux joyeux eccl,siaistiques | Beverages |

- 1 Open *Multiple Queries-1* from your sample files folder (if it isn't already open).

This is the workbook that you created in: *Lesson 11-15: Work with multiple queries*.

The workbook is empty apart from four *Connection only* queries that point to four different data sources.

- 2 Create an append query that will return appended data from the *Condiments*, *Beverages*, *Confections* and *DairyProduce* queries.

1. Click: Data→Get & Transform→New Query→Combine Queries→Append.

note

PQFL formulas and blank spaces

In this lesson I was careful to name one of the queries: *DairyProduce* (without a space) rather than *Dairy Produce* (with a space).

If you include spaces in query names you will have to refer to them in a special way within PQFL formulas.

To refer to a query called *Dairy Produce* (with a space) you'd need to type:

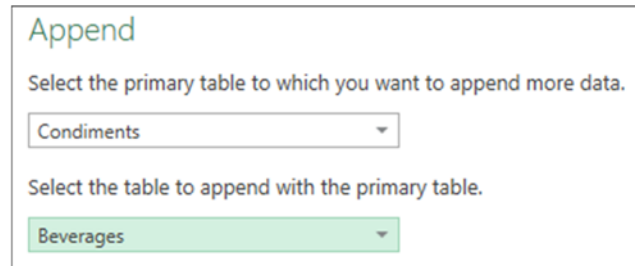
#"Dairy Produce"

The complete formula would be:

= Table.Combine({Condiments, Beverages, Confections, #"Dairy Produce"})

The *Append* dialog appears.

2. Select *Condiments* in the *Select the primary table to which you want to append more data* box.
3. Select *Beverages* in the *Select the table to append with the primary table* box.



4. Click the *OK* button.

The *Get & Transform* editor appears, showing the rows from the *Beverages* query appended to the rows from the *Condiments* query.

Unfortunately, the *Append* dialog only allows two queries to be appended. Fortunately, it is very easy to amend the PQFL formula to append all four tables (see sidebar facing page).

5. Look at the PQFL formula in the formula bar.



You can probably guess what you need to do to add the other two tables.

6. Edit the formula (in the formula bar) so that it reads:

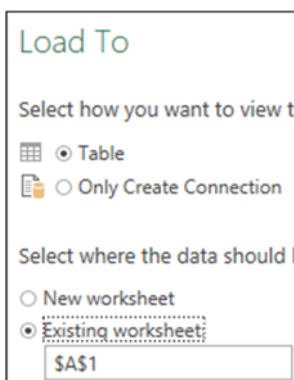


Be very careful to spell **DairyProduce** without a space. I was careful to name the query in this way, as the PQFL language needs a special syntax to specify queries with spaces (see sidebar).

As soon as you change the PQFL code and press the **<Enter>** key, the *Confections* and *DairyProduce* data is appended to the existing data displayed in the preview window.

Sometimes the data you need to append may have different column header names, different data types, or spurious extra columns. If that were the case, you'd have to use the skills you have learned in this session to transform the relevant query so that it matched the other data.

3. Rename your query to: **Full Stock List**
4. Close the query editor and load the query result to a table in cell A1 of the existing worksheet (there is only one worksheet in this workbook).
You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns.*
5. Save your work as *Multiple Queries-2*.



trivia

Third normal form

Dr E.F. Codd (1923-2003) is a cult figure to every database designer worthy of the name.

He was not only the inventor of the relational database (while working for IBM) but he also wrote the rulebook for excellence in database design.

Codd's books are essential (and even exciting) reading for anybody who designs relational databases for a living. I have read all of them and hung on every word!

Codd defined a set of rules that are collectively called third normal form (or 3NF for short).

Codd's rules are broken into three parts: First Normal Form, Second Normal Form and Third Normal Form.

Each set of rules builds upon the earlier set so that a database design cannot conform to 2NF unless it first conforms to 1NF (and a 3NF design must first conform to 2NF).

In theory, all of the corporate databases that you work with should be third normal form compliant. In practice you'll often find database design errors that may make some databases difficult to work with.

Lesson 11-17: Understand normal and de-normalized data

What is normal data?

Most relational databases conform to design rules that are collectively called *Third Normal Form* (often abbreviated to 3NF).

A data source that complies with these rules is said to contain normal (or normalized) data. Compliance with 3NF is a pre-requisite for an efficient relational database that can reliably allow rows to be added, edited or deleted.

What is de-normalized data?

Before the *data model* and *OLAP pivot table* were introduced (in Excel 2013), Excel pivot tables could only use a single table of source data.

When data is extracted from multiple tables (usually residing in a relational database such as SQL Server, Oracle or Access) into a single table, the data in the single table is referred to as *de-normalized data*.

You could argue that Excel's new data modeling abilities have removed the need for de-normalized data and this is often the best approach. The advantages of using a de-normalized single-table data source instead of a data model will be discussed later, in *Lesson 11-18: Create a simple two-table merged query (sidebar)*.

De-normalizing by adding derived values

One of the most important normal form rules prohibits *derived* values in a relational database table. There are very good reasons for this (see facing page sidebar).

A *derived value* is a value that can be calculated from other columns in the same row. In: *Lesson 11-13: Create an aggregated data query*, you worked with this view:

| | A | B | C | D |
|---|--------------------|--------------|--------------|-----------|
| 1 | ProductName | CategoryName | UnitsInStock | UnitPrice |
| 2 | Chai | Beverages | 39 | 18.00 |
| 3 | Chang | Beverages | 17 | 19.00 |
| 4 | Guaraná Fantástica | Beverages | 20 | 4.50 |

The challenge in: *Lesson 11-13: Create an aggregated data query*, was to calculate the total *Inventory Value*. A correctly designed database cannot contain a column showing *Inventory Value* because it can be calculated by multiplying the *UnitsInStock* column by the *UnitPrice* column (in other words it would be a *derived value*).

You de-normalized the data by adding a calculated (derived) field to show the *Inventory Value*. Here is the de-normalized data that you created:

Add Custom Column

New column name

Custom column formula:

note

Why are derived values a bad thing in a relational database?

The problem with a derived field is that it allows you to ask a database the same question in two different ways.

This often results in two different answers.

When a database can answer the same question with different results it is referred to as being corrupt.

Imagine that the database designer (in this lesson's example) had designed the *Product* table with an *Inventory Value* field.

Every time a programmer wrote code that updated the *UnitPrice* and *UnitsInStock* field it would be necessary to also remember to update the *Inventory Value* field.

At some point, a routine might be coded that "forgot" to update the *Inventory Value* field. This wouldn't really be the programmer's fault, as it shouldn't have been there in the first place.

Now imagine that the CEO asks two analysts to provide the total inventory value.

The first analyst computes the value by multiplying the *QuantityPerUnit* by the *UnitPrice* and sums the resulting values.

The second analyst sums the *Inventory Value* field.

The CEO, presented with two very different figures, asks which figure is correct.

The analysts advise that both of them are correct.

Perhaps the CEO would fire the analysts or programmer at this point, but really he should fire the database designer.

| | A | B | C | D | E |
|---|--------------------|--------------|--------------|-----------|-----------------|
| 1 | ProductName | CategoryName | UnitsInStock | UnitPrice | Inventory Value |
| 2 | Chai | Beverages | 39 | 18.00 | 702.00 |
| 3 | Chang | Beverages | 17 | 19.00 | 323.00 |
| 4 | Guaraná Fantástica | Beverages | 20 | 4.50 | 90.00 |

De-normalizing by combining tables

In: *Lesson 9-3: Understand primary and foreign keys*, you learned about related tables. Here's the data that you worked with:

| CategoryID | CategoryName | Description |
|------------|--------------|--|
| 1 | Beverages | Soft drinks, coffees, teas, beers, and ales |
| 2 | Condiments | Sweet and savory sauces, relishes, spreads, and seasonings |
| 3 | Confections | Desserts, candies, and sweet breads |

| ProductID | ProductName | CategoryID | QuantityPerUnit | UnitPrice |
|-----------|-------------------------|------------|---------------------|-----------|
| 1 | Chai | 1 | 10 boxes x 20 bags | 18.00 |
| 3 | Aniseed Syrup | 2 | 12 - 550 ml bottles | 10.00 |
| 16 | Pavlova | 3 | 32 - 500 g boxes | 17.45 |
| 24 | Guaraná Fantástica | 1 | 12 - 355 ml cans | 4.50 |
| 25 | NuNuCa Nuß-Nougat-Creme | 3 | 20 - 450 g glasses | 14.00 |
| 34 | Sasquatch Ale | 1 | 24 - 12 oz bottles | 14.00 |

An important normal form rule is that you cannot have information about more than one entity in the same table. Think of an entity as a noun (or thing). A *Category* is a noun and a *Product* is a noun.

Now consider the de-normalized version of the above tables that you worked with in: *Lesson 9-5: Efficiently import data using a view*:

| | A | B | C | D |
|---|------------------|--------------|---------------------|-----------|
| 1 | ProductName | CategoryName | QuantityPerUnit | UnitPrice |
| 2 | Alice Mutton | Meat/Poultry | 20 - 1 kg tins | 39.00 |
| 3 | Aniseed Syrup | Condiments | 12 - 550 ml bottles | 10.00 |
| 4 | Boston Crab Meat | Seafood | 24 - 4 oz tins | 18.40 |

You can see that this de-normalized view breaks normal form rules because the *Category Name* field rightly belongs in the *Category* table.

It is quite correct for a database administrator to create this type of de-normalized view within a correctly designed relational database, as the view is only used for reporting purposes.

Later, in: *Lesson 11-18: Create a simple two-table merged query*, you'll learn how to create this type of de-normalized table using the *Get & Transform* tool.

Learning more about third normal form

Third normal form is a subject that sometimes even confuses specialized IT personnel. Apart from professional specialist database designers it is unusual to find IT workers who can create a correctly normalized database from a business requirement specification.

This lesson has only presented a simplified overview of third normal form but it will be sufficient for your work with *Get & Transform*.

If you want to learn more about third normal form, an Internet search will uncover a huge amount of information upon this widely misunderstood subject.

note

What is a join?

When a relationship is defined within a query it is called a *join*.

There are different types of join that define the data that is returned when there is a missing foreign key in a foreign key column.

There is a drop-down list at the bottom of the *Merge* dialog used in this lesson, that allows you to specify a *Join Kind*.

| |
|--|
| Left Outer (all from first, matching from second) |
| Right Outer (all from second, matching from first) |
| Full Outer (all rows from both) |
| Inner (only matching rows) |
| Left Anti (rows only in first) |
| Right Anti (rows only in second) |

The *Left Outer join* is the default join kind and it is the join type that you'll use nearly all of the time. In this lesson, imagine that there are some *Product* rows that have a missing (or invalid) *CategoryID*.

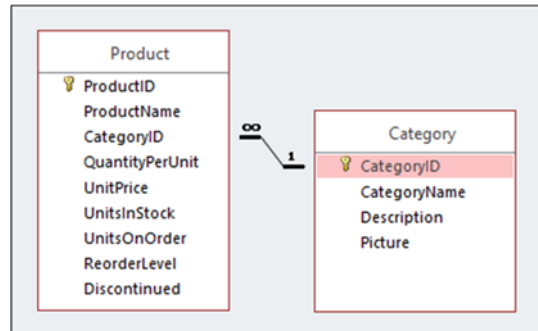
The *Left Outer Join* returns all of the rows in the *Product* table. No category data is returned if there is a missing *CategoryID* (in other words the *Category* field will be left blank).

Sometimes you might want to only return *Product* records if the *Category ID* field is not blank and there is a matching *CategoryID* field in the *Category* table. This type of join is called an *Inner join*.

With this knowledge, you should be able to figure out (from the description provided in brackets) what the other four (and lesser used) join types would return in different situations.

Lesson 11-18: Create a simple two-table merged query

In: *Lesson 9-1: Import tables from an external relational database*, you imported two tables from an Access database into two Excel tables. One table contained *Product* information and the other contained *Category* information.



In: *Lesson 9-3: Understand primary and foreign keys* you learned that the two tables had a relationship linking the *CategoryID* primary key (in the *Category* table) to the *CategoryID* foreign key (in the *Product* table).

- 1 Open a new blank workbook.
- 2 Create two *Get & Transform* queries to extract the values from the *Product* and *Category* tables in the *Two Tables-1* Access database in your sample files folder.

Because both of these tables are contained in the same database there's a really quick way to do this.

1. Click: *Data*→*Get & Transform*→*New Query*→*From Database*→*From Microsoft Access Database*.
2. Navigate to the *Two Tables-1* database and open it.
3. Check the *Select Multiple items* check box.
4. Check the check boxes next to each of the two tables.
5. Click the *Edit* button.

The query editor opens and two queries are shown in the left-hand query list. One query for each table.



- 3 Create a merged query to show the *Category* name in the *Product* query.

1. Select the *Product* query in the left-hand query list.
2. Click: *Home*→*Combine*→*Merge Queries*.

The *Merge* dialog appears.

3. Click in the *CategoryID* (foreign key) column in the *Product* table shown at the top of the *Merge* dialog.

note

VLOOKUP, View, Data Model or de-normalized Get & Transform query?

You have now discovered four different ways to work with relational data.

You may wonder when to use each. Here are the advantages of each approach:

1. VLOOKUP

This is the most primitive way to create a de-normalized data extract from related tables. It may also be the one you see most often as it is the only method that most Excel users understand.

The VLOOKUP method is very slow and cumbersome to implement and maintain. It is the least preferred way to work with relational data.

2. View

A *View* is the best way to obtain a de-normalized data extract from a relational database.

While *Views* are wonderful, they can only be used with a relational database data source and can only be created by specialized IT staff.

3. Data Model

An excellent (and usually the best) way to work with relational data.

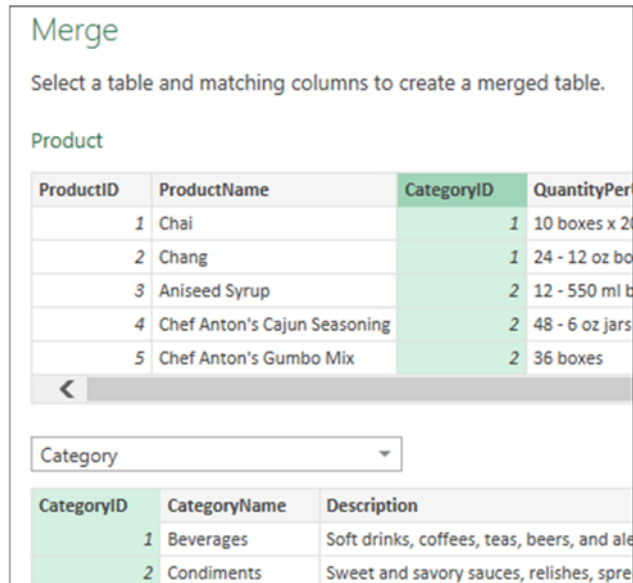
A *data model* can only be used as the data source for an OLAP pivot table or 3D Map and cannot be automatically refreshed at a timed interval.

4. Get & Transform Query

Unlike the data model, a de-normalized extract created by a *Get & Transform* query can be used as the data source of a linked table as well as a pivot table.


Get & Transform queries can also be automatically refreshed.

4. Select the *Category* table from the drop-down list in the middle of the dialog.
5. Click in the *CategoryID* (primary key) column in the *Category* table.
6. Click *OK* to merge the queries.



Notice that a new column has appeared on the right of the preview window.

- 4 Expand the new column to see the contents of the *Category* table.

1. Click the expand button  on the right of the *NewColumn* column header.
2. Click the *OK* button.

The *Category* names are now shown in the preview window.

- 5 Remove the *ProductID*, *CategoryID*, *NewColumn.CategoryID*, *NewColumn.Description* and *New.Column.Picture* columns from the query.

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns.*

- 6 Move the *NewColumn.CategoryName* field so that it is next to the *ProductName* field.

You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns.*

| | ProductName | NewColumn.CategoryName | QuantityPerUnit |
|---|-------------|------------------------|--------------------|
| 1 | Chai | Beverages | 10 boxes x 20 bags |
| 2 | Chang | Beverages | 24 - 12 oz bottles |

- 7 Close the query editor, creating *Connection only* queries.

You learned how to do this in: *Lesson 11-15: Work with multiple queries.*

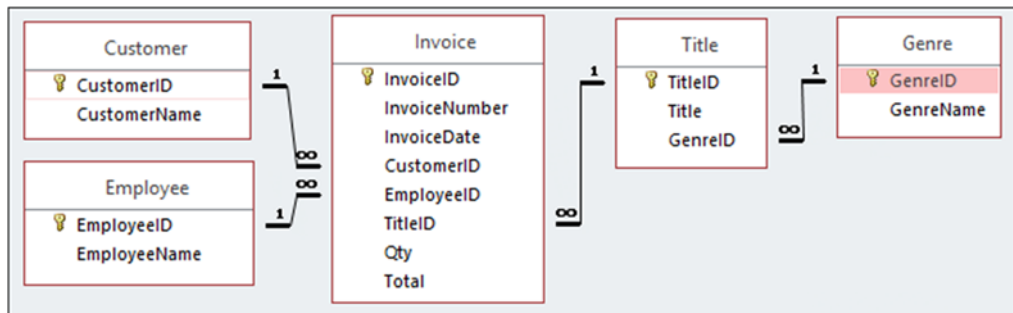
- 8 Save your work as *Two Table Merge-1*.

Lesson 11-19: Create a five-table merged query

In this lesson you'll create a *Merge* query to extract data from five related queries and then combine them all into a single de-normalized table.

- 1 Open a new blank workbook.

Here is the schema for the tables used in this lesson:



You may recognize the schema as the one you used in the exercise at the end of: *Session Nine: Data Modeling, OLAP and Business Intelligence*. In the exercise you created a relational *data model* from this data.

The challenge for this lesson will be to extract a single de-normalized table from the related tables using a *Get & Transform Merge Query*. The single-table extract will provide the values of every field in the database (seventeen fields).

- 2 Create five queries containing all of the data from all five tables in the *Film Sales-1* Excel workbook.

Because all of these tables are contained in the same workbook, there's a really quick way to do this.

1. Click: Data→Get & Transform→New Query→From File→From Workbook.
2. Navigate to the *Film Sales-1* workbook and open it.
3. Check the *Select multiple items* check box.
4. Check the check boxes next to each of the five tables (see sidebar).
5. Click the *Edit* button.

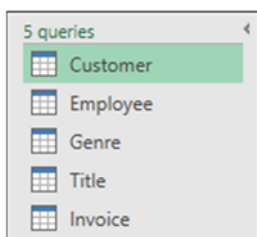
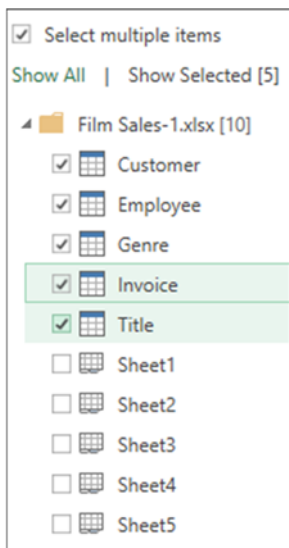
The query editor opens and five queries are shown in the left-hand query list, there is one query for each table (see sidebar).

- 3 Merge the *Genre* query into the *Title* query.

You learned how to do this in: *Lesson 11-18: Create a simple two-table merged query*.

Select the *Title* query and then merge the *Genre* query into the *Title* query using the *GenreID* key to link the tables.

When you expand the new column, the *Title* query should display all fields from both the *Title* and *Genre* tables.



Film Sales-1

- 4 Merge the *Customer* query into the *Invoice* query.
 The key field for this merge will be the *CustomerID*.
 The *Invoice* query should now return all fields from both the *Invoice* and *Customer* tables.
- 5 Merge the *Employee* query into the *Invoice* query.
 The key field for this merge will be the *EmployeeID*.
 The *Invoice* query should now return all fields from the *Invoice*, *Customer* and *Employee* tables.
- 6 Merge the *Title* query into the *Invoice* query.
 The key field for this merge will be the *TitleID*.
 Remember that the *Title* query is itself a merged query (that includes fields from the *Genre* table). By merging the *Title* query with the *Invoice* query, you will make both the *Title* and *Genre* table's data available within the *Invoice* query.
- 7 Expand all of the new columns.
 The *Invoice* query should now return all fields from the *Invoice*, *Customer*, *Employee*, *Title* and *Genre* tables (every field in the entire database).
- 8 Remove all of the primary and foreign key columns from the query (all of the fields that are suffixed with *ID*).
 You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.
- 9 Rename the columns, removing the *NewColumn* prefix.
 You learned how to do this in: *Lesson 11-4: Move, remove, rename, filter and sort columns*.
- 10 Close the query editor, creating *Connection only* queries.
 You learned how to do this in: *Lesson 11-15: Work with multiple queries*.
- 11 Create a new linked table using the *Invoice* query.
 1. Right-click on the *Invoice* query in the *Workbook Queries* task pane.
 2. Select *Load to...* from the shortcut menu
 The *Load To* dialog appears.
 3. Click the *Table* option button and the *Existing worksheet* option button.
 4. Select cell A1 as the location for the table and click the *Load* button.

| | A | B | C | D | E | F | G | H |
|---|---------------|-------------|-----|--------|---------------|---------------------|---------------------------------------|-----------|
| 1 | InvoiceNumber | InvoiceDate | Qty | Total | EmployeeName | CustomerName | Title | GenreName |
| 2 | 136438 | 01/10/2012 | 15 | 122.76 | Lee, Frank | Silver Screen Video | Lawrence of Arabia | Biography |
| 3 | 136438 | 01/10/2012 | 9 | 67.46 | Lee, Frank | Silver Screen Video | The Discreet Charm of the Bourgeoisie | Comedy |
| 4 | 136442 | 03/10/2012 | 6 | 55.76 | Newhart, Anna | AV Supplies | The Discreet Charm of the Bourgeoisie | Comedy |

- 12 Save your work as *Five Table Query-1*.

Session 11: Exercise

For this exercise there is an *Exercise* folder in your session eleven sample files folder that contains three workbooks: *Customer-1*, *Order-1* and *Product-1*.

- 1 Open a new blank workbook.
- 2 The *Customer-1* workbook contains three tables called: *FranceTable*, *UKTable* and *USATable*. Create three *Connection only* queries that return the data from each table.
- 3 Create a new *Append* query called *AllCustomers* that will return all of the rows contained in the other three queries.
- 4 The *Order-1* and *Product-1* workbooks each contain a single table (*OrderTable* and *ProductTable*). Create two new *Connection only* queries that return the data from each table.
- 5 Add a *Custom Column* to the *OrderTable* query named *Total Price* that will calculate the total price of each row using the formula: **[Quantity]*[Unit Price]**
- 6 Set the data type of the new *Total Price* column to: *Decimal Number*.
- 7 Merge the *ProductTable* query into the *OrderTable* query.
- 8 Merge the *AllCustomers* query into the *OrderTable* query.
- 9 Expand the fields in the new columns so that all details from the *ProductTable* and *AllCustomers* query are visible.
- 10 Remove all of the primary and foreign key columns and the *Country* column (but not the *NewColumn.1.Country* column) from the query.
- 11 Rename all of the merged fields by removing the *NewColumn* and *NewColumn.1* prefixes.
- 12 Load the *OrderTable* query into the *data model*.
- 13 Use the *data model* to create an OLAP pivot table showing monthly total sales by year and month for each *Company Name* and *Product Name*.

| | A | B | C | D | E |
|----|------------------------|---------------|----------|----------|-------------|
| 1 | Sum of Total Price | Column Labels | | | |
| 2 | | +2014 | +2015 | +2016 | Grand Total |
| 3 | Row Labels | | | | |
| 4 | Around the Horn | 1,379.00 | 6,589.00 | 5,838.50 | 13,806.50 |
| 5 | Camembert Pierrot | | 510.00 | | 510.00 |
| 6 | Chang | | 285.00 | | 285.00 |
| 7 | Chocolade | | 153.00 | | 153.00 |
| 8 | Filo Mix | | 266.00 | | 266.00 |
| 9 | Fløtemysost | | 258.00 | | 258.00 |
| 10 | Gnocchi di nonna Alice | 608.00 | | | 608.00 |
| 11 | Gorgonzola Telino | | 625.00 | 812.50 | 1,437.50 |

- 14 Save your work as: *Exercise 11-End*.

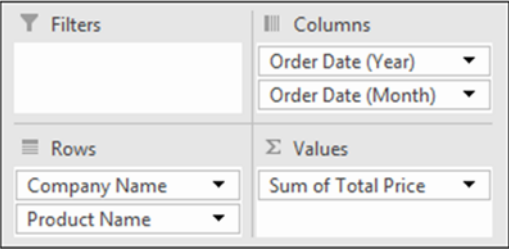
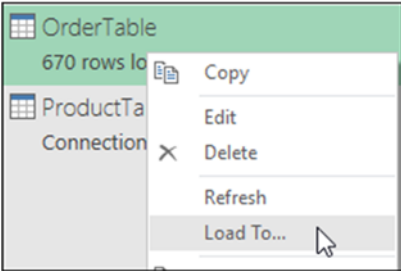

Customer-1, Order-1, Product-1

If you need help
slide the page to
the left



Session 11: Exercise Answers

These are the three questions that students find the most difficult to answer:

| Q 13 | Q 12 | Q 7,8 |
|---|---|--|
| <p>1. Click: Insert→Tables→Pivot Table. The <i>Create Pivot Table</i> dialog appears.</p> <p>2. Click the <i>Use this workbook's Data Model</i> option button.</p> <p>3. Click: the <i>OK</i> button.</p> <p>Because the pivot table uses the <i>data model</i> as source data, it is created as an OLAP pivot table.</p> <p>4. Set the field settings as follows:</p>  <p>This was covered in: <i>Lesson 9-11: Understand OLAP pivot tables.</i></p> | <p>1. Right-click on the <i>OrderTable</i> query in the <i>Workbook Queries</i> task pane.</p> <p>2. Click <i>Load To...</i> from the shortcut menu.</p>  <p>3. Check the: <i>Add this data to the Data Model</i> check box.</p>  <p>This was covered in: <i>Lesson 11-10: Transform date and time columns.</i></p> | <p>1. Open the <i>Order Table</i> query in the <i>Get & Transform</i> editor.</p> <p>2. Click: Home→Combine→Merge Queries.</p> <p>3. Select the <i>ProductTable</i> from the drop-down list in the middle of the dialog.</p> <p>4. Click anywhere in the <i>ProductID</i> column in the top and bottom tables.</p> <p>5. Click <i>OK</i>.</p> <p>6. Do the same thing for the <i>AllCustomers</i> query (linking on the <i>CustomerID</i> field).</p> <p>This was covered in: <i>Lesson 11-18: Create a simple two-table merged query.</i></p> |

If you have difficulty with the other questions, here are the lessons that cover the relevant skills:

- 2,4** Refer to: **Lesson 11-15: Work with multiple queries.**
- 3** Refer to: **Lesson 11-16: Create an append query.**
- 5** Refer to: **Lesson 11-12: Add a custom calculated column.**
- 6** Refer to: **Lesson 11-6: Specify data types.**
- 9** Refer to: **Lesson 11-18: Create a simple two-table merged query.**
- 10** Refer to: **Lesson 9-3: Understand primary and foreign keys and Lesson 11-4: Move, remove, rename, filter and sort columns.**
- 11** Refer to: **Lesson 11-4: Move, remove, rename, filter and sort columns.**