Power Query #2

Import, Clean, Transform, and Load Data in Excel

Table of Contents

History of Power Query 3

What Is Power Query & M Code? 3

Where is Power Query in Excel? 3

Connect to Data Sources Using the Get & Transform group 4

Cleaning Data? 4

Transforming Data? 5

Loading Data from Power Query Editor into Excel 5

Project Goal for Video 02 6

Steps in Project: 7

- 1) Download Zipped Folder 7
- On-Premises Folder Path or File Path 9
- Source Data and Destination Data 9
- Delimiters & Csv & Txt Files 9
- Csv File Structure 10
- Excel File Structure & Excel Tables 10
- Access Database File Structure 11
- 2) Import: Table From Csv File 11
- Power Query Editor 12
- Advanced Editor & M Code 14
- 3) Load as Connection Only 15
- M Code values 16
- M Code expressions 18
- Let expression 19
- Keywords 20
- 4) Import: Table From an Excel File 20
- 5) Import: Table From Access Database 23
- Lookup terminology 25
- 6) Clean: Split Description Field in fSales query 25
- Query Options to Turn Off Automatic Data Types 27

7) Clean: Convert ISO Dates to Real Serial Number Dates 29
Clean: Converting International Dates to Local Dates 30
Power Query Merge Feature (Join Operations Used by the Merge Feature) 32
8) Transform: Use Merge Feature to Look up Product Price 33
9) Investigating Transform Errors and Fixing Problem 35
Add Column Tab in Power Query Ribbon 38
10) Transform: Multiply Feature to Add New Column 38
Table.AddColumn Function 39
Data Types 40
11) Transform: Custom Columns to Add New Column 41
Rounding in Power Query 42
12) Transform: Remove Unwanted Columns 44
13) Edit Load Location 45
14) Load Query Output to an Excel Table, Load a Second Time To The PivotTable Cache, Make Chart 46
15) Refreshing when source data changes 49
16) DataSource.NotFound Error 50
Summary 52
Keyboard shortcuts seen in video 52

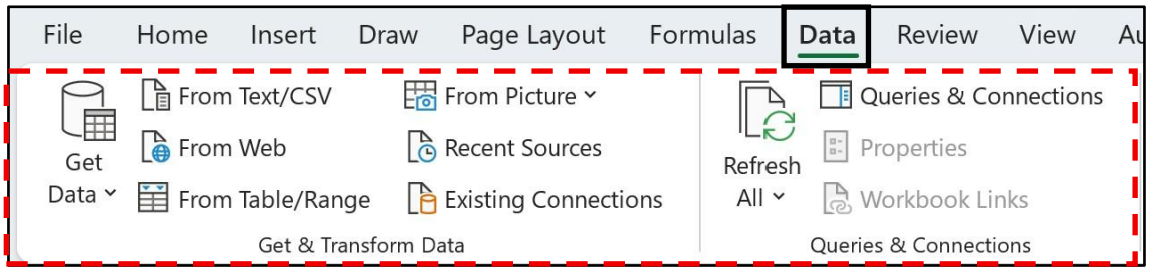
History of Power Query

Since 1994 and the Excel 5.0 version, using a PivotTable was the preferred method of converting a table of data into useful summary reports and chart visuals. It was easy to use a PivotTable if it was based on a correctly structured table with column names and records. But often the data came from many different sources and did not have the correct table structure to create the reports and chart visuals. To get the correct structure for the PivotTable tool, it was necessary to use VBA code or manually convert the data to the correct structure. Power Query was first introduced in Excel in 2013, two decades after the PivotTable was invented. Power Query made it dramatically easier to take data from different data sources and then clean, transform, and load the data to the best location in an Excel workbook file. Then, in 2015, Power BI Desktop was invented to help us easily report and visualize data. Microsoft embedded Power Query in Power BI Desktop to make connecting to and structuring data easy. Finally, in 2023, Microsoft gave us the full Power Query tool in the online Power BI service; many call it Dataflow. Now, with whatever tool you may be using, Power Query allows you to connect to almost any data source and transform the data into exactly the structure you need to create your data analysis and analytic solutions.

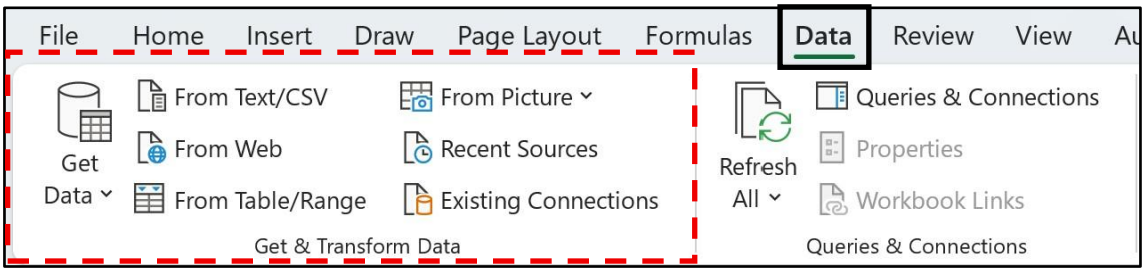
What Is Power Query & M Code?

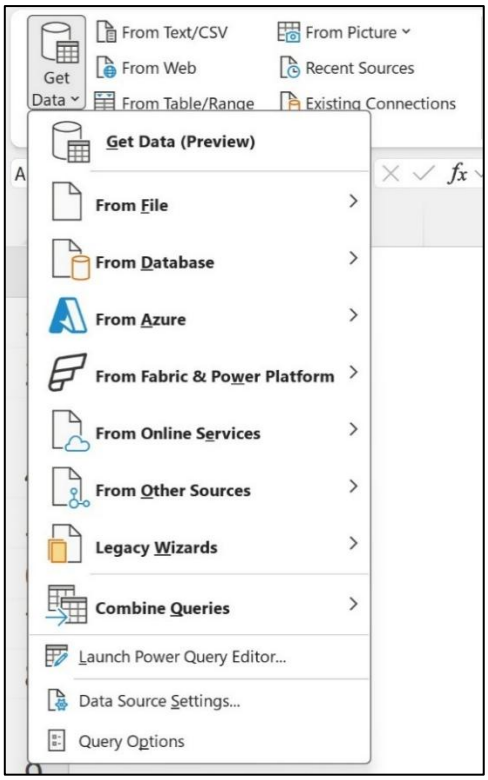
- Power Query is a tool in Excel (and Power BI) that allows you to connect to data wherever the data may be (online, database, text files, Excel files) and import, clean and transform the data before loading it to the worksheet, a PivotTable or the Data Model.
- What is so amazing about Power Query is that it can work with data in any location, such as: databases, tables, records, lists, worksheets, Excel Tables, text files, online locations, and many others.
- And once you gather the data into the Power Query Editor, you can use the user interface to create all the steps needed to get your data into the correct structure for the analysis that you need. And, when you use the user interface to add steps to your query, Power Query writes all the formulas and code for you in the computer language M Code.
- In Power Query, the formulas are written with M Code, a case-sensitive, function-based computer language that “mashes up” your data together into to just the structure that you want. The “mash up” term is where Microsoft got the name “M Code”.
- As you become proficient with Power Query and M Code, you’ll learn to create queries by combining the user interface with your own handwritten M Code—an approach you’ll practice throughout this class.

Where is Power Query in Excel?

- Power Query is in the Data Tab in the Excel Ribbon in the Get & Transform Data and the Queries & Connections groups:
- 

Connect to Data Sources Using the Get & Transform group

- Connect to data sources in Excel Power Query in the Get & Transform group in the Data Tab in the Excel Ribbon:
- 

- The Get Data dropdown provides even more data connection options:
- 

Cleaning Data?

- Cleaning data involves taking data that you cannot use and cleaning it so that you can use it.
- For example, if you are given an ISO (International Standard Organization) date you cannot use it as a real date (serial number date) and must clean it to become a real date. The picture is below:
ISO Date		Real Date (Serial Number Date)
20260625	=> Into =>	6/25/2026
- As a second example, if you download numbers from a financial web site that contains a letter to indicate the unit, you must clean it and remove the letter. The picture is below:
ISO Date		Real Date (Serial Number Date)
2.7 M	=> Into =>	2.7

Transforming Data?

- Transforming data involves changing the structure of the container that holds the data so that you can use the data.
- For example, you may get data in a Csv file or a Json file, and you must transform it into a proper table. In this video we will see how to transform a Csv file into a table.
- As a second example, you may get records of data in a column and need to transform the column of records into a proper table. The picture is below:

Data
01/05/2023
Chantel
$1,645
01/06/2023
Jim
$4,829
01/08/2023
Chantel
$3,635
01/11/2023
Chantel
$2,713

→

Date	Person	Sales
1/5/2023	Chantel	1,645
1/6/2023	Jim	4,829
1/8/2023	Chantel	3,635
1/11/2023	Chantel	2,713

- As a third example, you may have a table with multiple columns and you only need to use two of them. In this case, you would remove the unnecessary columns. The picture is below:

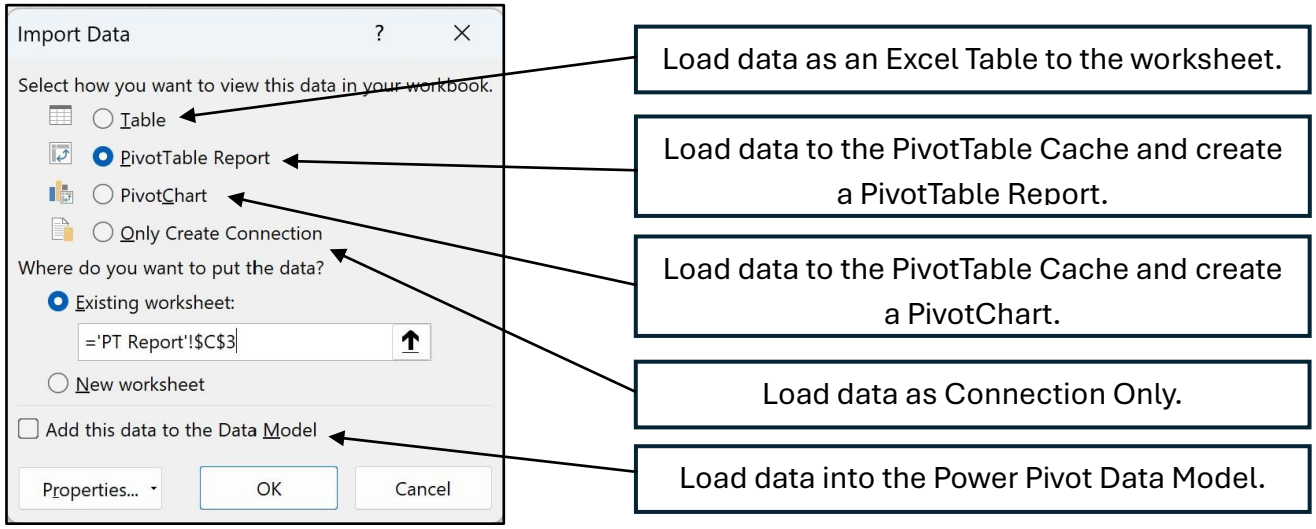
Date	Person	Product ID	Units Sold
1/5/2023	Chantel	Q4369	45
1/6/2023	Jim	S5189	43
1/8/2023	Chantel	A3451	75
1/11/2023	Chantel	Q4369	61
1/13/2023	Jim	S5189	89

→

Person	Units Sold
Chantel	45
Jim	43
Chantel	75
Chantel	61
Jim	89

Loading Data from Power Query Editor into Excel

- Loading data from the Power Query Editor in Excel allows you to load to the worksheet, a PivotTable, a Pivot Chart, Connection Only or the Data Model.
- Here is a picture of the Import Data dialog box used to load data from the Power Query Editor:



Project Goal for Video 02

The company sells boomerang products and wants to summarize net sales by boomerang product and sales channel.

1) Import Source Data:

PowerQuery-02.csv
PowerQuery-02.xlsx
PowerQuery-02.accdb

1) Sales Table in Csv File (Fact Table)

```
Date,Description,Units
20261113,375-Web Site,2
20261015,250-Web Site,60
20261030,200-Web Site,6
20260615,375-Web Site,120
20251207,350-Web Site,1
```

2) Lookup Table in an Excel file (Dimension Table)

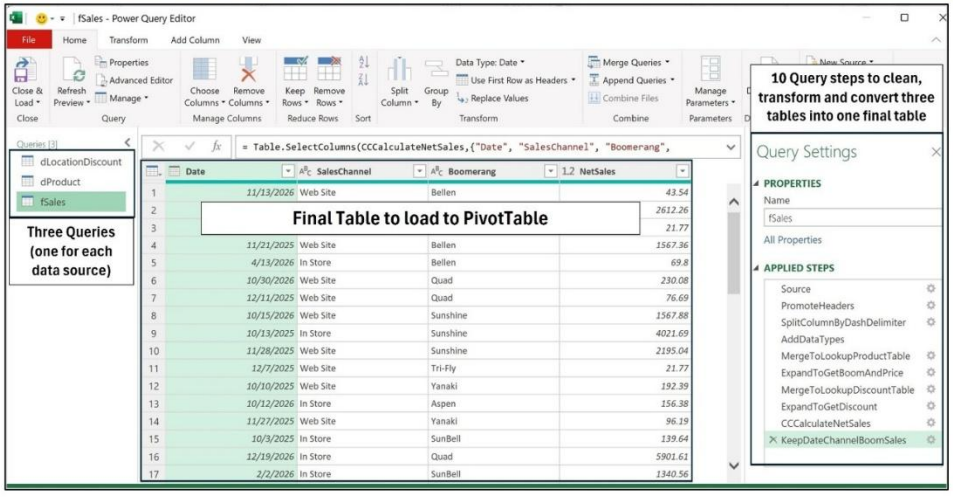
ProductID	Boomerang	Category	Flight Range (m)	COGS	Price
200	Quad	Freestyle	20	21.79	43.95
225	Carlota	Freestyle	20	17.28	34.95
250	Sunshine	Fast Catch	20	16.36	29.95
275	GelFast	Fast Catch	20	29.6	49.95
300	Eagle	Beginner	22	8.68	19.95

3) Lookup Table in an Access database file (Dimension Table)

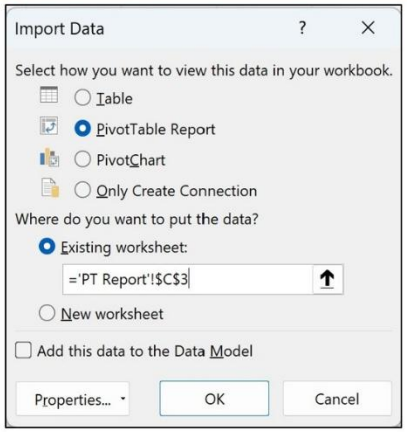
LocationDiscount

ID	Sales Locatio	% Discount	Click to Add
1	Phone	0.00%	
2	Web Site	12.75%	
3	Store	6.75%	

2) Clean & Transform the Data:



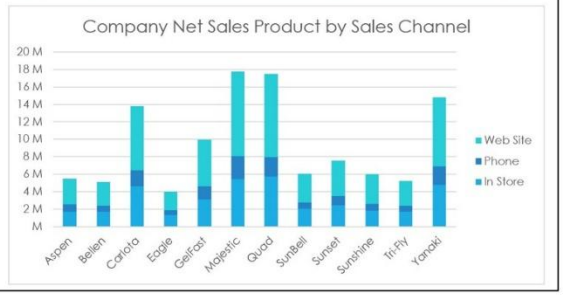
3) Load the Data to a PivotTable:



4) Create Table, PivotTables and Charts:

Date	SalesChannel	Boomerang	Net Sales
11/13/2026	Web Site	Bellen	43.54
6/15/2026	Web Site	Bellen	2612.26
10/12/2026	Web Site	Bellen	21.77
11/21/2025	Web Site	Bellen	1567.36
4/13/2026	In Store	Bellen	69.8
10/30/2026	Web Site	Quad	230.08
12/11/2025	Web Site	Quad	76.69
10/15/2026	Web Site	Sunshine	1567.88
10/13/2025	In Store	Sunshine	4021.69
11/28/2025	Web Site	Sunshine	2195.04
12/7/2025	Web Site	Tri-Fly	21.77
10/10/2025	Web Site	Yanaki	192.39
10/12/2026	In Store	Aspen	156.38
11/27/2025	Web Site	Yanaki	96.19
10/3/2025	In Store	SunBell	139.64
12/19/2026	In Store	Quad	5901.61
2/2/2026	In Store	SunBell	1340.56
11/13/2025	Web Site	SunBell	78.39
11/5/2026	In Store	Tri-Fly	1395.95
12/10/2026	Phone	Carlota	104.85
11/2/2025	Web Site	Yanaki	128.26
9/15/2026	In Store	SunBell	1005.42
10/28/2025	Phone	SunBell	179.7
12/2/2026	Phone	SunBell	89.85
9/12/2025	Web Site	Eagle	1879.89
11/29/2025	Web Site	Aspen	97.55
12/4/2026	Web Site	GelFast	130.74
11/23/2026	Web Site	Majestic	4706.26
11/22/2025	In Store	Yanaki	102.81
12/7/2026	In Store	Eagle	18.6
10/25/2025	Web Site	Aspen	48.77

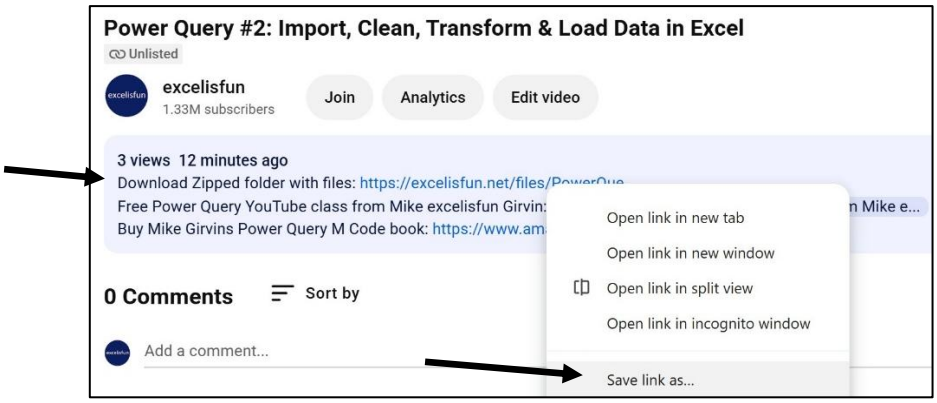
Sum of Net Sale SalesChannel				
Boomerang	In Store	Phone	Web Site	Grand Total
Aspen	1,721,406.78	806,357.50	2,976,870.55	5,504,634.83
Bellen	1,695,105.07	703,365.45	2,705,436.56	5,103,907.08
Carlota	4,609,779.58	1,809,990.60	7,375,217.29	13,794,987.47
Eagle	1,293,044.99	576,415.35	2,142,639.14	4,012,099.48
GelFast	3,124,756.47	1,459,289.25	5,358,112.23	9,942,157.95
Majestic	5,460,841.92	2,582,602.25	9,707,103.70	17,750,547.87
Quad	5,706,687.09	2,273,006.10	9,495,907.21	17,475,600.40
SunBell	2,033,045.80	750,277.45	3,289,652.59	6,072,975.84
Sunset	2,437,338.33	1,027,234.35	4,101,655.12	7,566,227.80
Sunshine	1,843,132.77	757,016.20	3,386,129.92	5,986,278.89
Tri-Fly	1,694,267.65	686,125.00	2,860,952.98	5,241,345.63
Yanaki	4,754,604.19	2,131,830.75	7,931,986.13	14,818,421.07
Grand Total	36,374,010.64	15,563,510.25	61,331,663.42	113,269,184.31



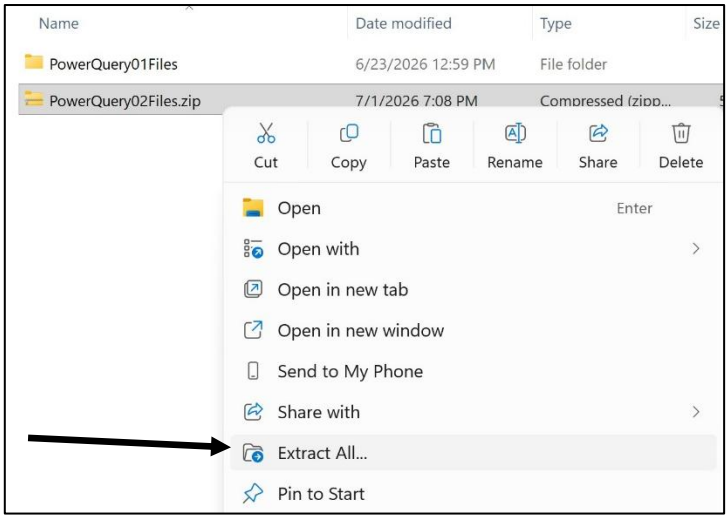
Steps in Project:

1) Download Zipped Folder

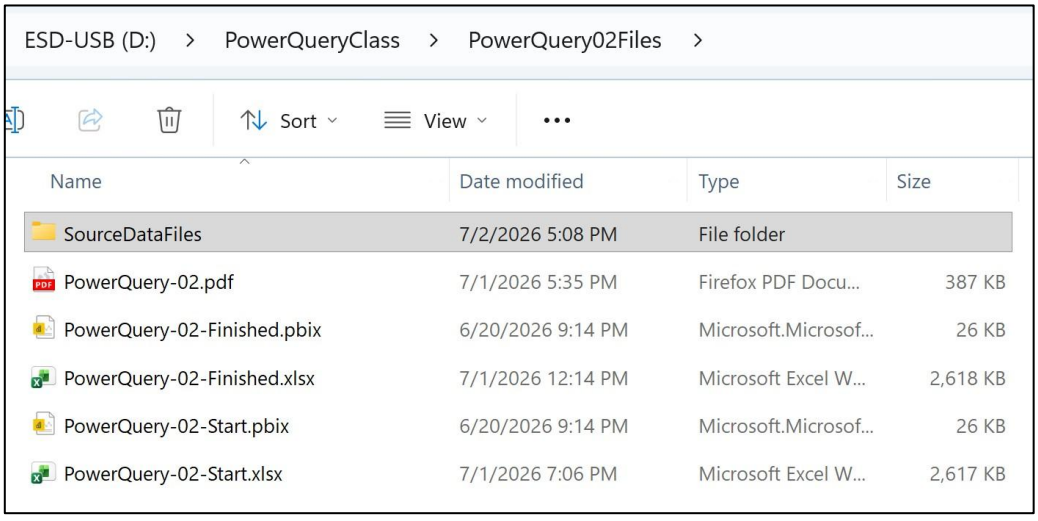
1. In the description area below our YouTube Video #2, right-click the link and click “Save link as…” to download the zipped folder with all the files for this video, as seen here:



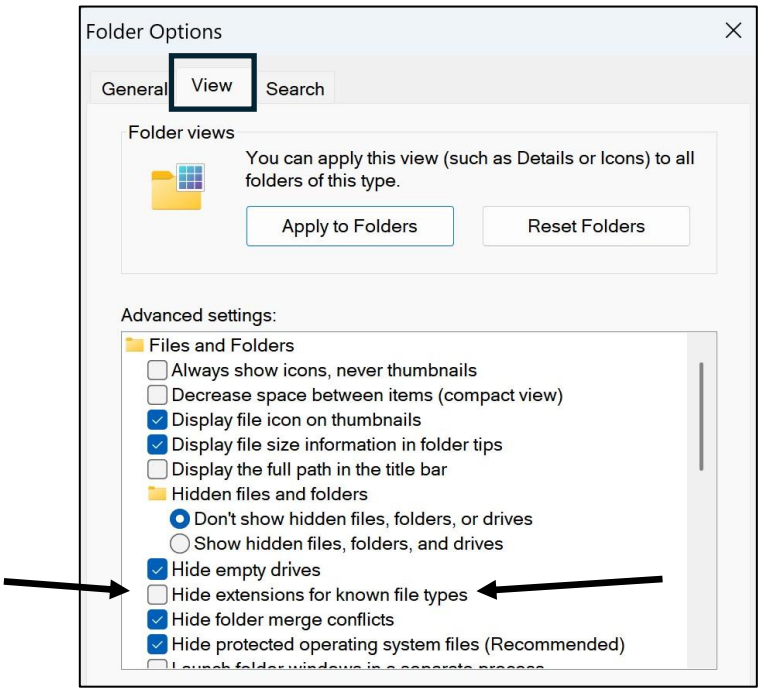
2. After you save the zipped folder to the Power Query folder that you created on the desktop, open Windows File Explorer (Window Key + E), navigate to the Power Query folder, right-click the zipped folder and click “Extract All…”, as seen below. In the next dialog box, click the Extract button.



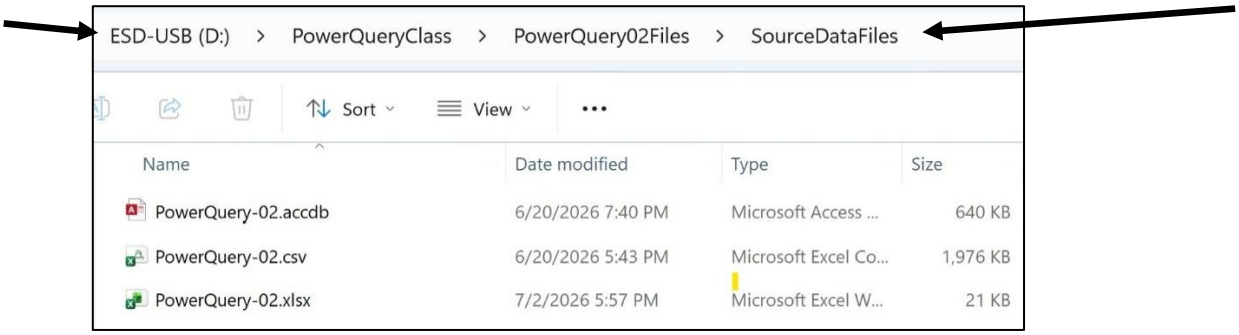
3. You should see five files and one folder named SourceDataFiles.



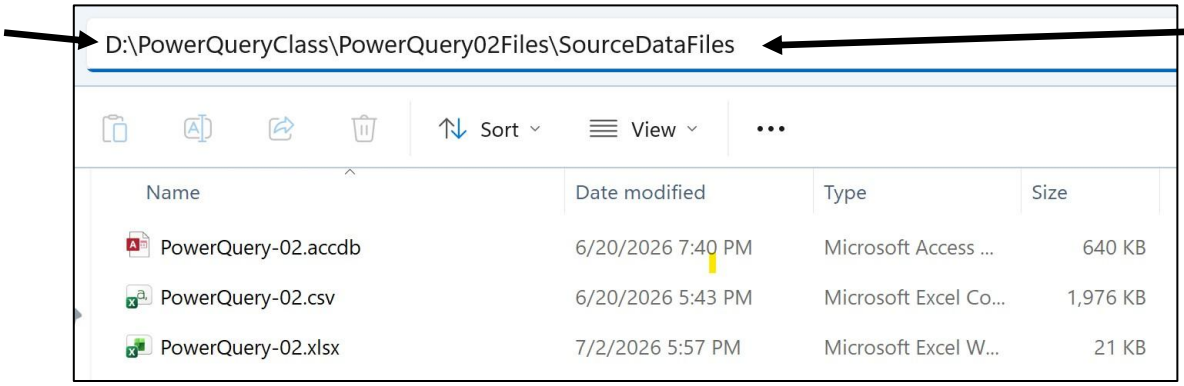
4. If you do not see the file extensions in Windows File Explorer, from the Start button, type "Folders Options" and navigate to the View tab and then uncheck the checkbox for "**Hide extensions for known file types**".



5. Look inside the SourceDataFiles folder and you should see the three files that we will import into Power Query. In the picture below, notice the folder path.



6. In Windows File Explorer, click in the address bar. The picture below shows the folder path that is for my computer (Michael Girvin's computer). Your computer will have a unique folder path. This folder path is important because when we import the files into Power Query, Power Query will memorize this **On-Premises Folder Path** as the source data path for the data being imported.



On-Premises Folder Path or File Path

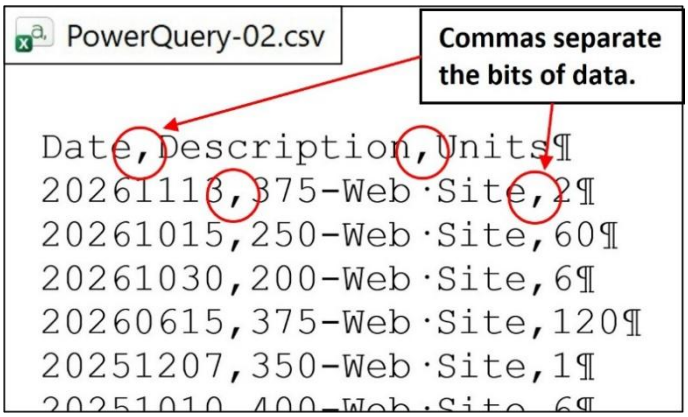
- When you import data into Power Query from a folder or a specific file, Power Query memorizes the source data path so it can find the data again when you refresh. If you connect to a folder, Power Query stores the **On-Premises Folder Path**. If you connect to a specific file, it stores the **On-Premises File Path**.
- These paths refer to locations on your computer or local network. If you later move, rename, or delete the folder or file, Power Query will no longer know where to find the source data, and you may see a "DataSource.NotFound: File or Folder" error. Later in this video, we will see how to update the source data path so Power Query can refresh the data again.
- On-premises file and folder paths are different from online data source paths. For example, a SQL Server database or website address uses an online connection or internet address, so it does not have the same problem caused by moving a local file or folder.

Source Data and Destination Data

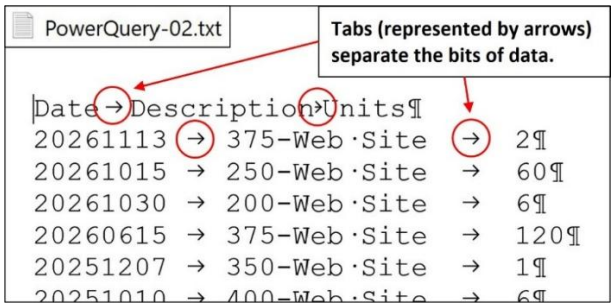
- When you build a data analysis solution in Excel or Power BI Desktop, you often use Power Query to connect to an external data source and import the data into the Excel or Power BI Desktop file. The external data is called the **Source Data**, and the imported data is called **Destination Data**.
- When the source data is located on your computer or local network, the on-premises folder/file path is used to connect the destination data to the source data during refresh.
- When the source data is located online, the online address is used to connect the destination data to the source data during refresh.

Delimiters & Csv & Txt Files

- A delimiter is a character or set of characters that separates bits of data.
- Common delimiters are commas or tabs, but you can have most any delimiter that you might want.
- The Csv file format uses a comma delimiter and the file extension ".csv" and are used to move tables of data from one system to another. In the picture below, you can see that the first row has the field names Date, Description, and Units and subsequent rows contain sales transaction records.



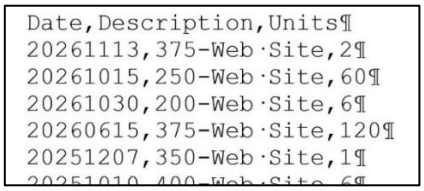
- The Txt file format uses a tab delimiter and the file extension ".txt" and are used to move tables of data from one system to another. In the picture below, you can see that the first row has the field names Date, Description, and Units and subsequent rows contain sales transaction records.



- Since the beginning of databases over 50 years ago, Csv files and Txt files have been used for moving data from one system to another system.

Csv File Structure

- A **Csv file** contains a single table of data with comma delimiters to separate the bits of data. In the first row, the field names are separated by commas. In subsequent rows the data in each record are separated by commas. When we use Power Query to import the Csv file, it can correctly interpret the file as a table of data. Here is a picture of the Csv file we will use in this video:



Excel File Structure & Excel Tables

- Whereas a Csv file contains a single table, **Excel files** can contain many tables and many objects. Power Query makes it easy to pick only the objects that you want from a given file.
- Objects that can be in an Excel file:
  - Worksheets.
  - Excel Tables (Ctrl + T).
  - Defined Names.
  - Print Areas.
  - Advanced Filter automatic Defined Names: Criteria and Extract.
  - Auto Filtered automatic and hidden Define Name: FilterDatabase.
- When you store tables of data in Excel files, you should convert proper tables into Excel Tables using the keyboard Ctrl + T. Although some other Excel objects may contain table-like data and can be imported with Power Query, **Excel Tables** are the most reliable table structure and are consistently interpreted as tables by Power Query. Here is picture of the Excel Table that we will import:

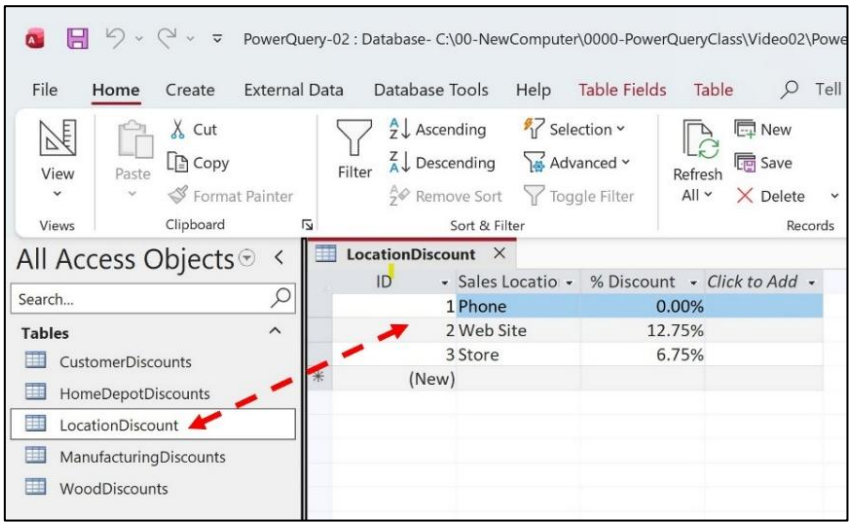


Lookup Table:

ProductID	Boomerang	Category	Flight Range (m)	COGS	Price
200	Quad	Freestyle	20	21.79	43.95
225	Carlota	Freestyle	20	17.28	34.95
250	Sunshine	Fast Catch	20	16.36	29.95
275	GelFast	Fast Catch	20	29.6	49.95
300	Eagle	Beginner	22	8.68	19.95
325	Aspen	Beginner	25	16.64	27.95
350	Tri-Fly	Freestyle	25	11.37	24.95
375	Bellen	Beginner	25	12.67	24.95
400	Yanaki	Mid-Range	30	20.23	36.75
425	SunBell	Mid-Range	35	14.83	29.95
450	Majestic	Long Range	40	21.52	44.95
475	Sunset	Long Range	45	19.66	39.95

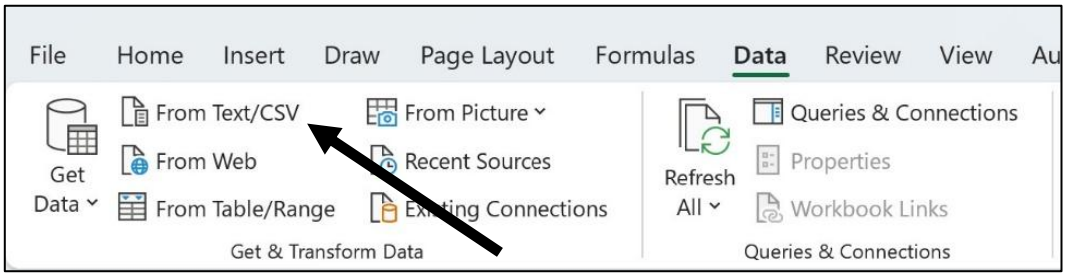
Access Database File Structure

- Like Excel files, Access Database files can have many objects such as tables, queries, and reports. Power Query makes it easy to pick only the objects that you want from a given file. In the picture below you can see five tables on the left. We will import the LocationDiscount table.

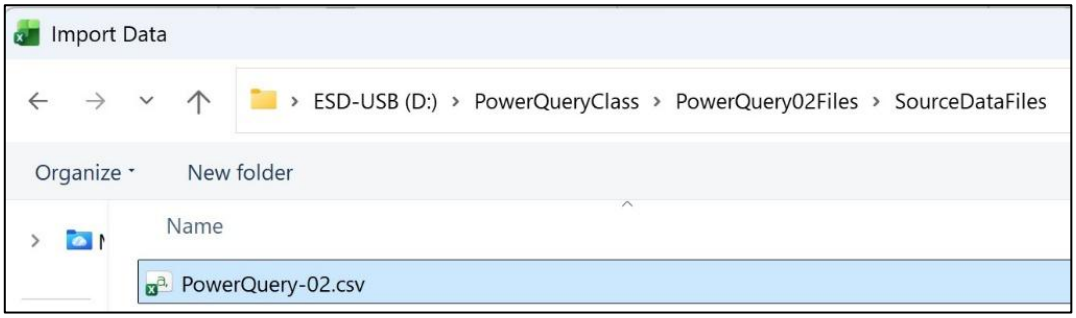


2) Import: Table From Csv File

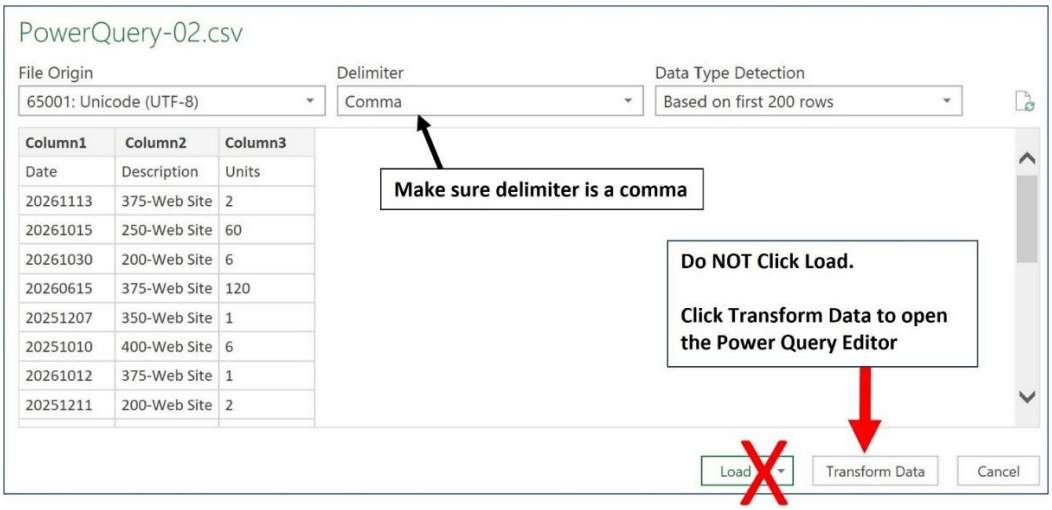
1. In the Excel file with the name "PowerQuery-02-Start.xlsx", select the worksheet "Table & PT Report", then go to the Data Ribbon tab, and in the Get & Transform group, click the "From Text/CSV" button, as shown below.



2. In the Import Data dialog box, navigate to the "PowerQuery-02.csv" file, select the file, then click the Insert button, as shown below.



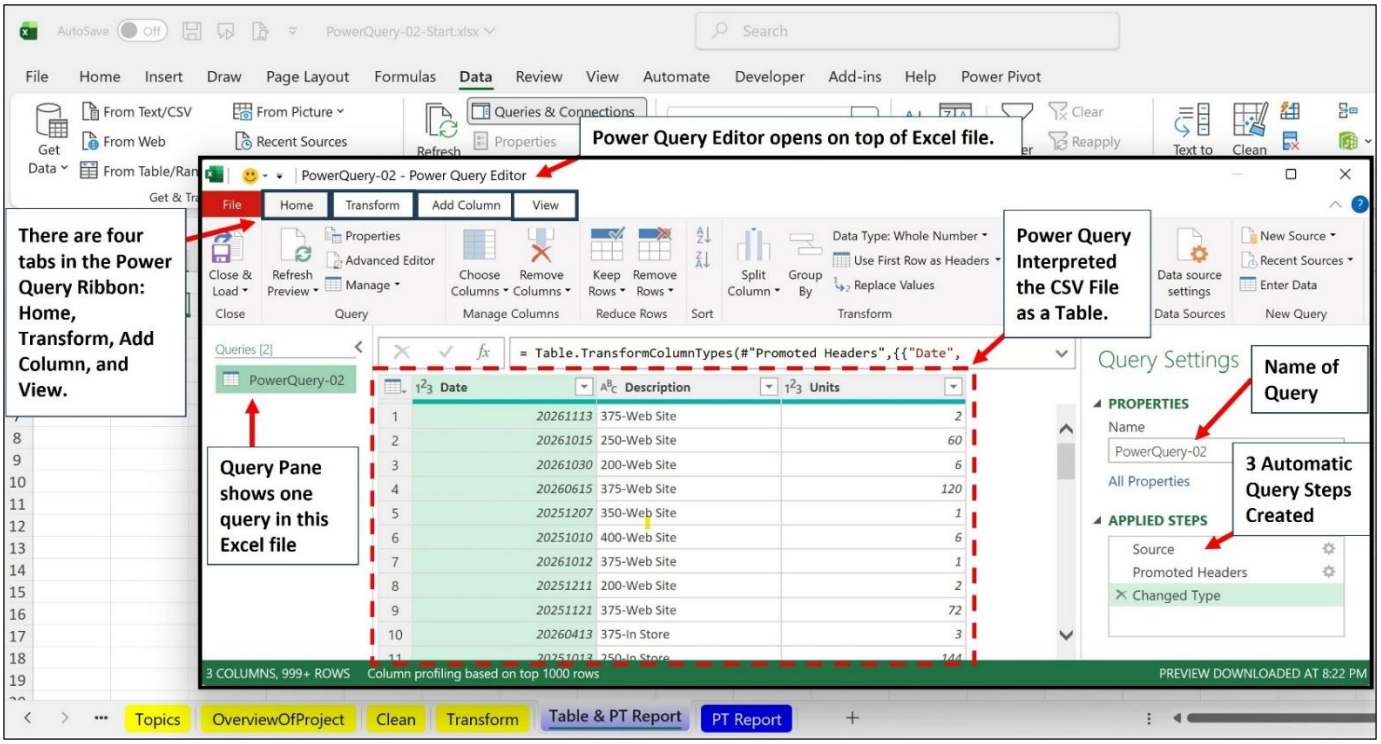
3. In the next dialog box, make sure that the delimiter is a comma, then click The Transform Data button to bring the Csv file table into the Power Query Editor.



Column1	Column2	Column3
Date	Description	Units
20261113	375-Web Site	2
20261015	250-Web Site	60
20261030	200-Web Site	6
20260615	375-Web Site	120
20251207	350-Web Site	1
20251010	400-Web Site	6
20261012	375-Web Site	1
20251211	200-Web Site	2

Power Query Editor

4. As shown in the picture below, after you click the Transform Data button, the Power Query Editor opens on top of the Excel file. On the left you can see a single query in the Queries Pane. On the right in the Query Settings Task Pane, you can see the name of the query and the three automatic query steps that were created to convert the Csv file table into a proper table. In the middle, you can see the proper table in the middle of the Power Query Editor. Finally, notice that there are four tabs in the Power Query Ribbon: Home, Transform, Add Column, and View.



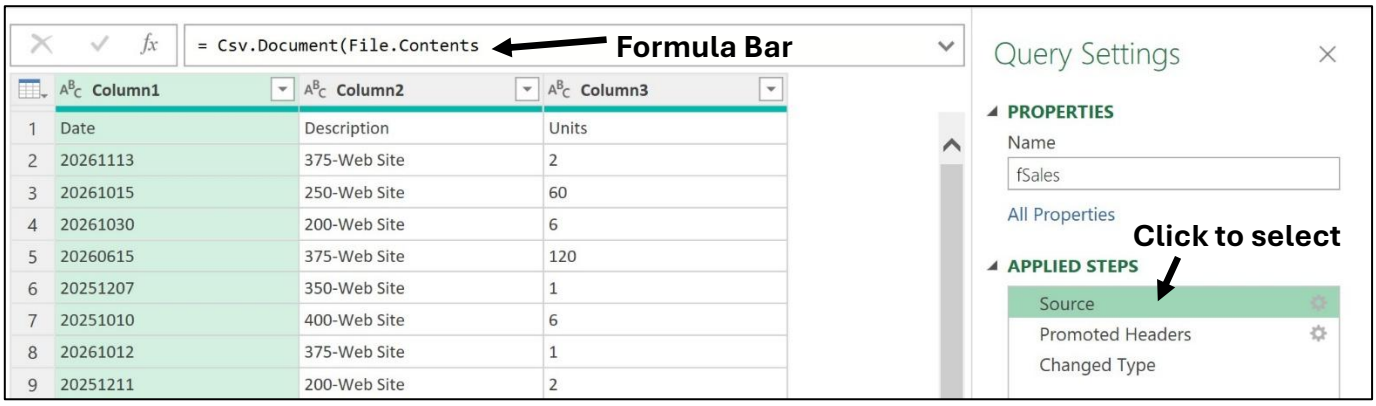
Note: If Power Query created only one automatic step, you can change your settings, so it creates the three automatic steps shown here. To do this:

1. In the Power Query Editor, click File.
2. Select Options and settings, then Query options.
3. In the Query Options dialog box, under Current Workbook, click Data Load.
4. Check Detect column types and headers for unstructured sources.

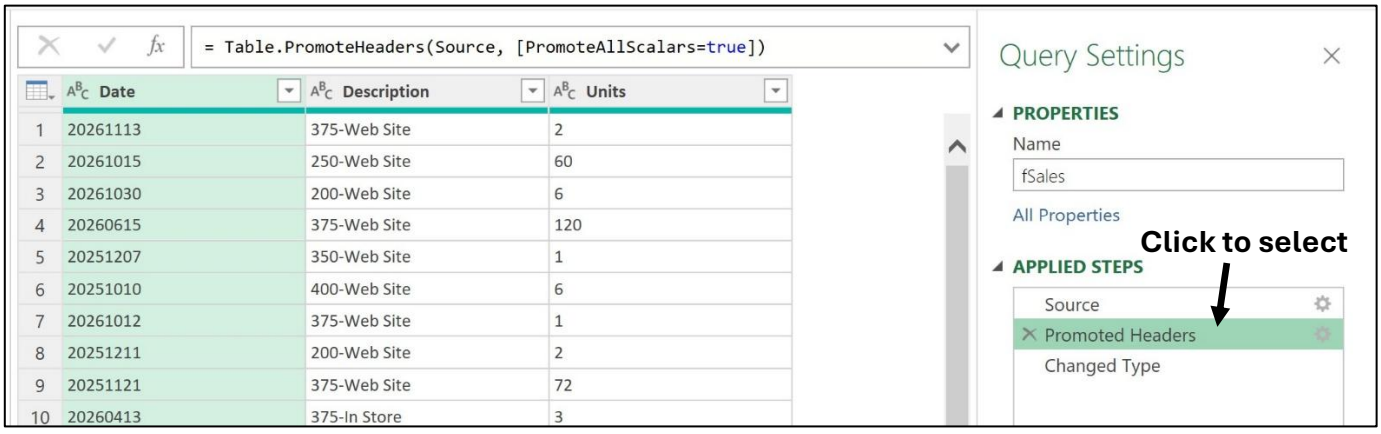
5. Using the Name textbox under Properties in the Query Settings Task Pane, name the Query “fSales”. The prefix “f” stands for fact table, and the “Sales” represents the sales record transactions.



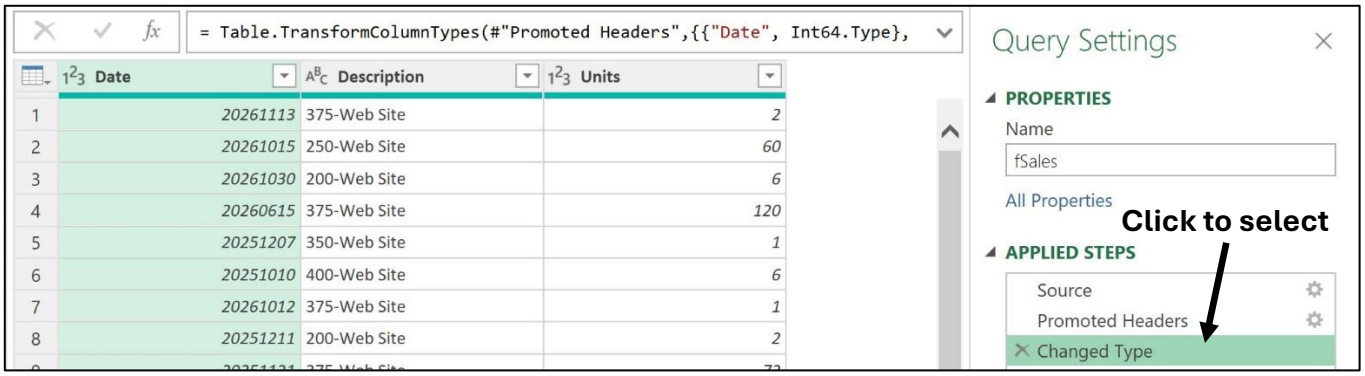
6. Select the Source query step under Applied Steps in the Query Settings Task Pane and notice that the query step results show a table with the field names Column1, Column2, Column3, and the field names from the Csv file are in the first row as a record. This is not the proper table that was delivered after all three query steps. In this initial Source step, Power Query sees the comma separated data as just a number of text records. Also notice in the formula bar that the Power Query M Code functions used for this step are named Csv.Document and File.Contents. These are easy to interpret function names, and such names are a hallmark of the M Code functions. Still further, notice that all the data is aligned to the left. This is the default alignment for text and is your visual cue that all the values are text values. Finally, the query step name, Source, and the functions in the formula bar are the M Code, which is the case-sensitive, function-based computer languages behind every Power Query query.



7. Select the second step in the Applied Steps list, Promoted Headers. Notice that this step takes the field names in the first row and promotes them to field names (column names or header names are synonyms). Also notice that the function name in the formula bar is Table.PromoteHeaders. Finally, the query step name, Promoted Headers, and the function in the formula bar are part of the M Code for this query step.

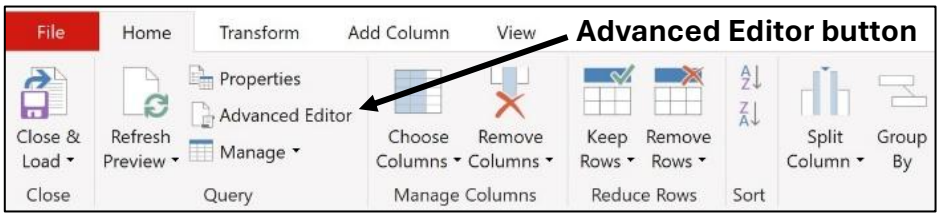


8. Select the third step in the Applied Steps list, Changed Type. Notice that this step adds data types so that the text field is aligned to the left and the two number fields are aligned to the right. Also notice the function name in the formula bar: Table.TransformColumnTypes. Finally, the query step name, Changed Type, and the function in the formula bar are part of the M Code for this query step.

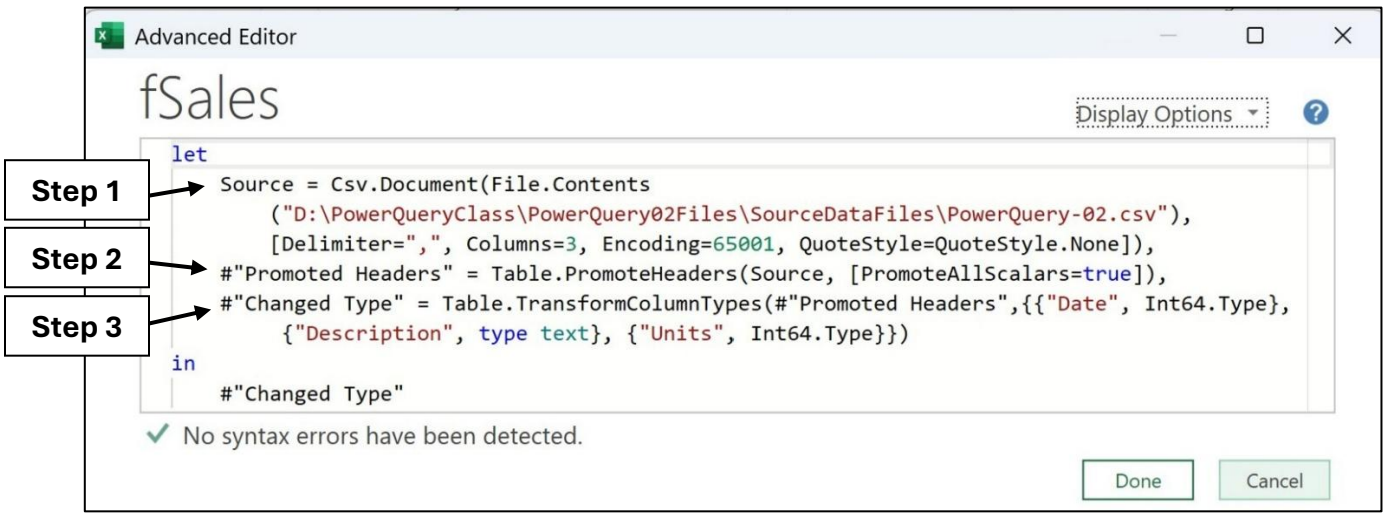


Advanced Editor & M Code

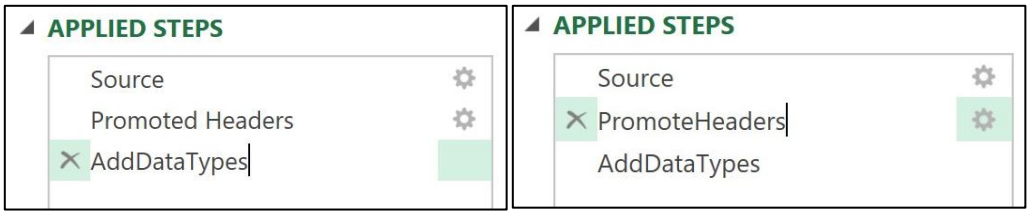
9. To view all the M Code that Power Query created for you, in the Home tab in the Power Query Ribbon, in the Query group, click the Advanced Editor button, as shown below:



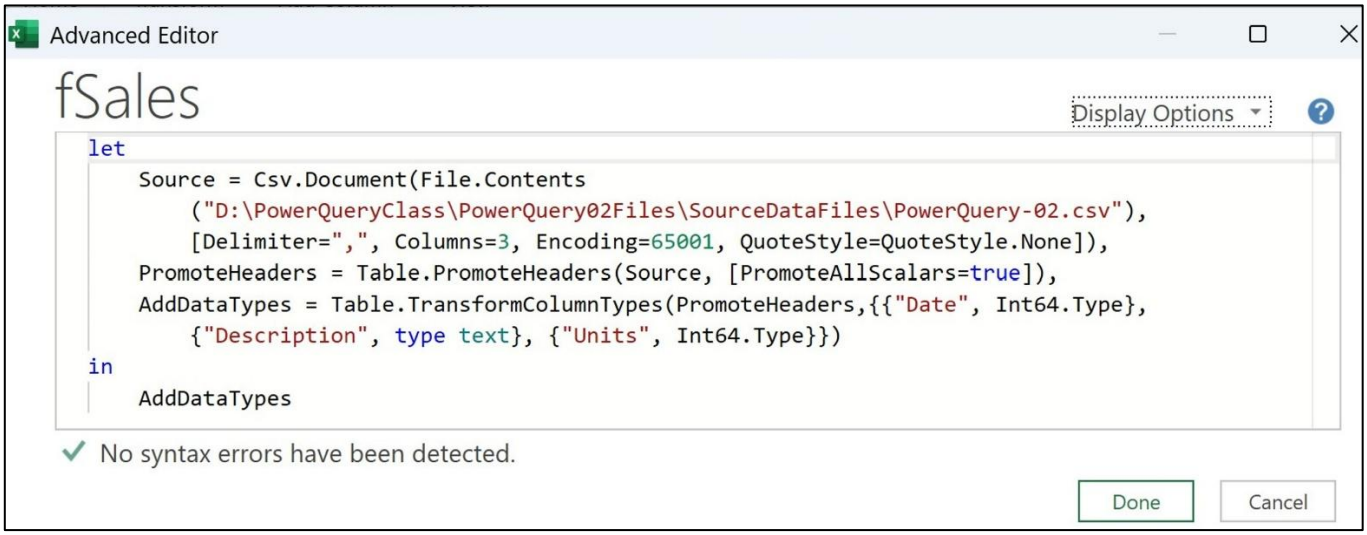
10. In the Advanced Editor window, you can view all the M Code that Power Query created for the **fSales** query. Notice the three query step names, followed by an equal sign, and then the M Code functions and formula elements that define each step. We will return to this later and examine it in more detail. For now, notice that the step name, or step identifier, **Source** does not require a # sign or quotation marks because it contains no spaces. However, the other two step identifiers, **#"Promoted Headers"** and **#"Changed Type"**, do require them because they include spaces. In M Code, any identifier—such as a query step name or query name—that contains spaces must be written with a # sign and quotation marks. This tells Power Query that the text is an identifier, not a regular text value. Regular text values use quotation marks only.



11. Click the Done button in the Advanced Editor. Then in the Applied Steps list, select Changed Type, press the **F2 key to put the identifier into edit mode** and change the name to AddDataTypes (you can name it what you would like, just do not use any spaces. Then remove the spaces from the Promoted Headers identifier. The names that I used are shown below:

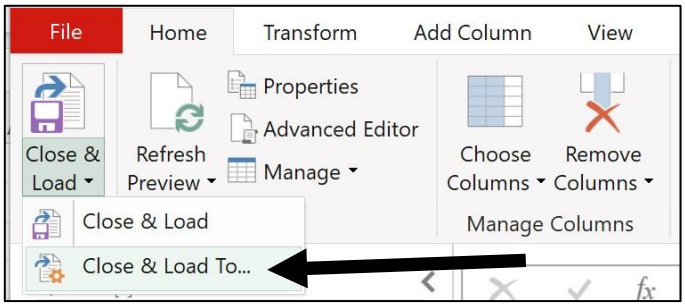


12. Open the Advanced Editor. As shown below the M Code is much easier to read when identifiers, whether query steps or query names, do not use spaces. And, in your first query, you are already editing the M Code! Bravo! Not only that but notice that we created a query and Power Query wrote the steps for us. Later, we will come back to this query and use the user interface to add steps, but in that case too, Power Query will write the M Code for use, even though we may give it some tweaks here and there.

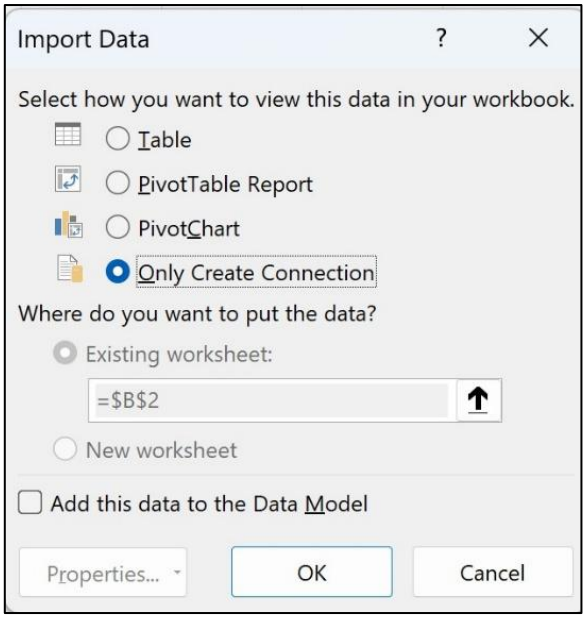


3) Load as Connection Only

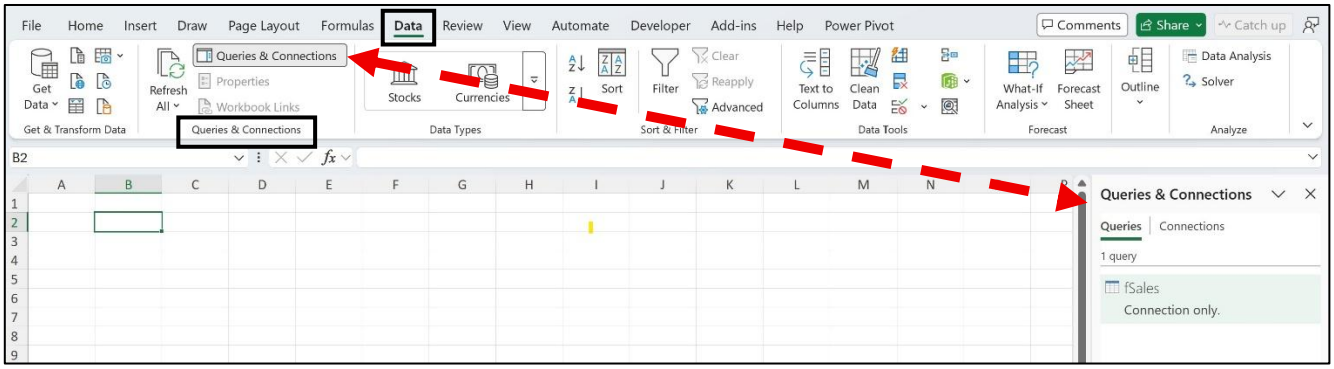
13. For the time being we are done with the fSales query. We will come back to this query and add steps after we import the two other lookup / dimension tables. Now we want to load this query as a Connection Only query. A **Connection Only query** is a query that we do not load to the worksheet, a PivotTable, or the Data Model, but instead we keep it in the Power Query Editor so that we can work with it later. To load this query as a Connection Only query, in the Home tab, in the Close group, click the Close & Load dropdown arrow (do not click the big Close & Load button because that will load the query to the worksheet), then click the Close & Load To... option, as shown below:



14. In the Import Data dialog box, select the dialog button Only Create Connection, then click OK.

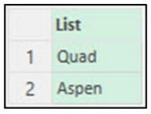
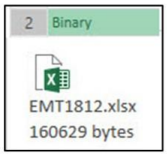


15. After you load, the Queries & Connections Task Pane opens on the right. As shown below, if it does not show up, then click the Queries & Connections button in the Queries & Connections group in the Data tab in the Excel Ribbon.



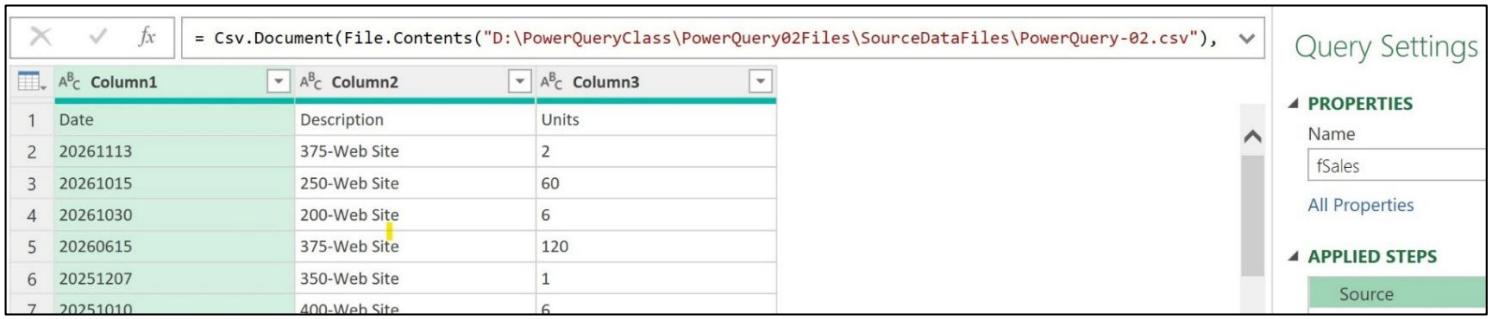
M Code values

- In our first example we created a query that delivered a table. A table value is one of 15 possible **M Code values** that are possible in the M Code case-sensitive, function-based computer language.
- M Code values are at the heart of M Code and Power Query. Having a good understanding of M Code values is the key to becoming a Power Query guru. If you understand the M code values, everything else will fall into place. There is a list of the 15 possible values in M Code on the following page.
- In this video we will create three queries that each deliver a table value. We will also learn about other M Code values such as text, logical, list, and others. In later videos in this class, we will examine all the different M Code values and use them to help us create data analysis solutions.

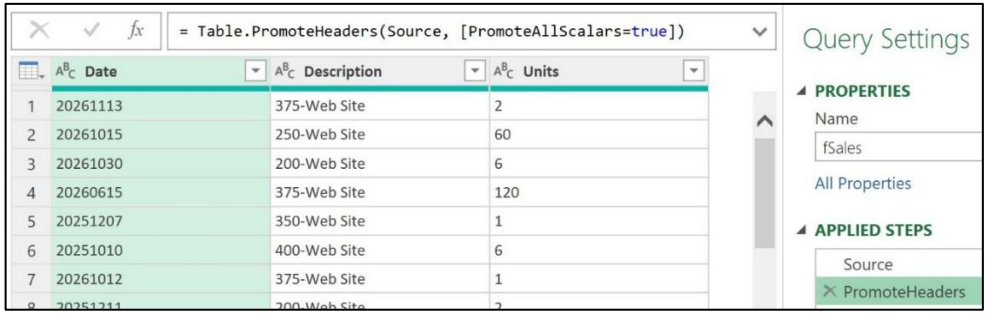
	Value	Description	M Code to Hard Code Value
1	Null	Absence of data.	Example: null
2	Logical	Boolean: true or false.	Example: true or false
3	Text	A string of Unicode characters.	Examples: "Quad", "A430G"
4	Number	Used for numeric and arithmetic operations.	Examples: -43 , 0 , 43.46 , 9.9e-5
5	Time	Time for 24 hour day as decimal between 0 and 1.	#time(hour, minute, seconds) Example: #time(11,10,57) = 11:10:57
6	Date	Date as a serial number (Common Era Gregorian calendar). Dates span from 01/01/0001 to 12/31/9999.	#date(year, month, day) Example: #date(1598, 01, 15)= 1/15/1598
7	DateTime	Datetime value contains both date and time.	#datetime(year, month, day, hour, minutes, seconds) Example: #datetime(2021,10,01,11,10,57)) = 10/01/21 11:10:57 AM
8	DateTime Zone	Represents a UTC (Universal Coordinated Time) date/time with a time-zone offset (last two arguments: hours & minutes).	#datetimezone(year, month, day, hour, minute, second, offset-hours, offset-minutes) Example: #datetimezone(2021,10,01,11,10,57,09,00) = 10/01/21 11:10:57 AM + 9 hours
9	Duration	An amount of time in units of days, hours, minutes, and seconds. Not supported in Dataflow.	#duration(days, hours, minutes, seconds) Example: #duration(01,01,10,0) = 1 day and 1 hour 10 minutes
10	Table	Table with field names and records.	Example: #table({"Boom Product", "Sales"}, {{"Quad",43},{"Aspen",35}}) = 
11	Record	An ordered sequence of fields.	Example: [Boom Product = "Quad", Sales = 43] = 
12	List	A sequence of M Code values.	Example: {“Quad”,”Aspen”} = 
13	Binary	Represents a sequence of bytes.	Example: Excel file = 
14	Function	Custom function that defines the variables and the mapping for those variables to deliver a value.	Example: Function calculates the effective rate = (APR, Periods) => Number.Power(1+APR/Periods, Periods)-1
15	Type	Classifies values with 1 of 16 data types.	Example: "type number" for defining the Decimal data type. The expressions: Value.Type(43) = "number" and the expression Value.Type(Value.Type(43)) = "type".

M Code expressions

- An **expression** is any M code that delivers an M code value.
- As an example, in the picture below the first query step uses the M Code functions File.Contents and Csv.Document to deliver an **M Code table value** with the default field names Column1, Column2, Column3. The identifier, Source is defined to deliver this same table value when used later in the query.
- As a second example, the File.Contents function requires an **M code text value** that defines a file path. In the example below, the on-premises file path is provided as the first argument to File.Contents, using the correct quotation syntax for an M Code text value.



- As a third example, in the picture below, the second query step uses the M code function Table.PromoteHeaders to return an M Code table value with the correct field names Date, Description, and Units. The first argument of Table.PromoteHeaders requires an M code table value, and the Source identifier from the previous step provides that value. The function then promotes the first row of the Source table to field names and returns a new M code table value with the correct headers. Finally, the PromoteHeaders identifier is defined to return this same table value when referenced later in the query.
- As a fourth example, in the picture below, the second argument of the Table.PromoteHeaders function defines the options for promoting headers. This argument requires an **M code record value**, which uses square brackets to define the record. Inside the record value, you can see the keyword *true*. This keyword provides the **M code logical value** *true*. Together, the record instructs the Table.PromoteHeaders function to promote all scalar header values in the table.

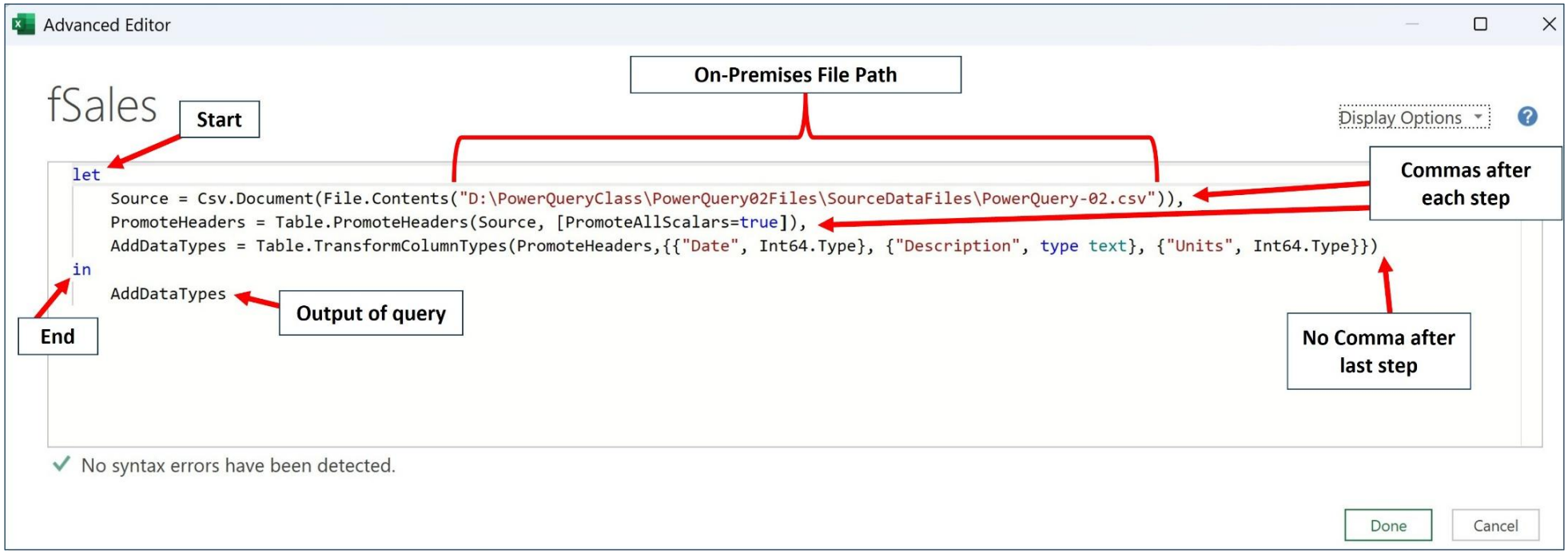


- Next, we want to look at the important let expression.

Let expression

The let expression in M code allows you to define query steps, also known as variables, and then use those variables to return a final M code value, or query output. If you are familiar with the LET function in an Excel worksheet, it works in a similar way: define variables, use the variables, and then return a final value. Here are the parts of a let expression:

- The keyword *let* starts the let expression.
- Each query step or variable:
    1. Starts with an identifier, or name of variable.
    2. An equal sign comes after the variable identifier.
    3. Following the equal sign is a function or other M Code expressions that can deliver an M Code value.
    4. If there is more than one variable, you must type a comma at the end of the variable step to allow the let expression to deliver an intermediate M Code value. The comma is a signal to Power Query that it must deliver an intermediate M Code value that can be acted on later in the query. If you only have a single variable in the query a comma is not needed
    5. For the last variable you do not type a comma, but instead you press Enter and type the keyword *in*. The keyword *in* after a variable step that does not end with a comma signals that query has ended and it can deliver a final M Code value or output.
- After the *in* keyword, you usually repeat the identifier for the last variable step, but you do not have to. You can type any M Code that delivers an M Code value.



- Here is a cheat sheet for the let expression:

```
let
   VariableName1 = Expression,
    #"Variable Name 2" = Expression,
   VariableName3 = Expression
in
   Output
     (Output is usually the last variable name, but can
     be any variable, query name or expression)

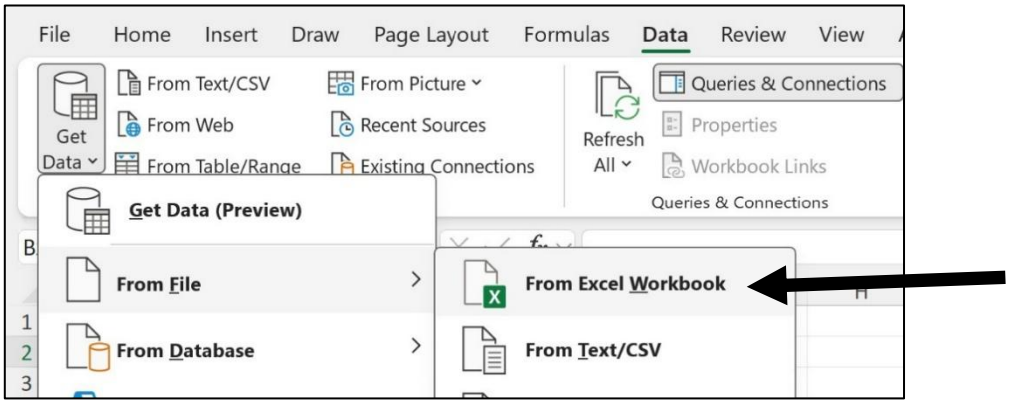
// one-line comment
/* multiple line comments */
```

Keywords

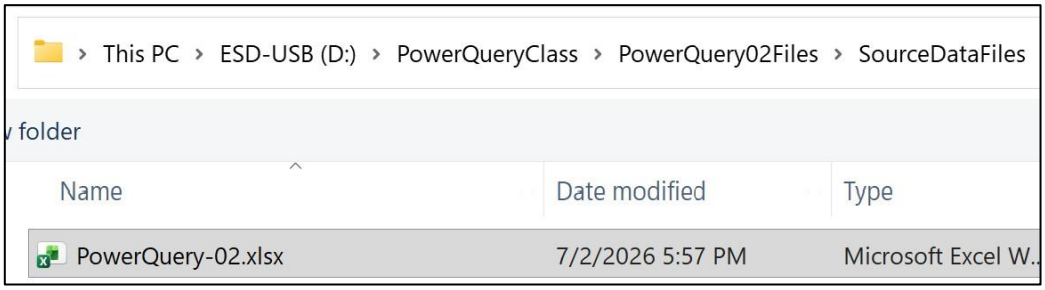
- A keyword is a reserved word that has a predefined use. For example:
  - *let* is a keyword that indicates the start of a let expression.
  - *in* is a keyword that ends the query steps and precedes the query output.
  - *true* is a keyword that delivers the logical M Code value TRUE.
  - *each* allows a formula to calculate in each row of a table or list.
  - *type* assigns the data type.

4) Import: Table From an Excel File

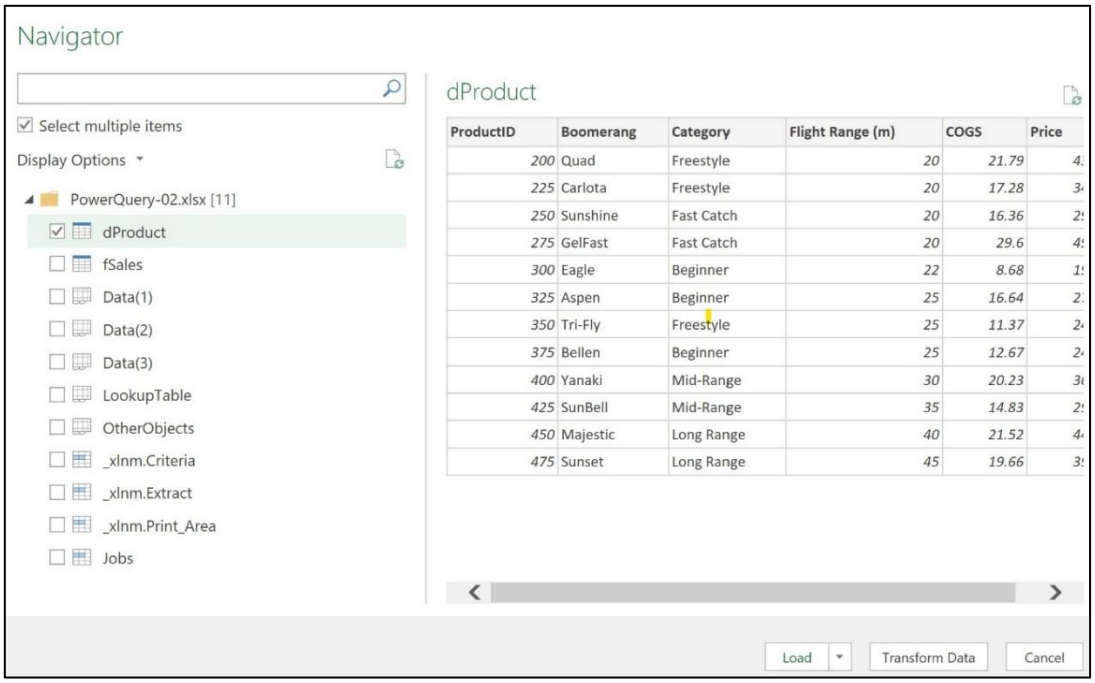
1. To import a table from within an Excel file, in the Data tab in the Excel Ribbon, in the Get & Transform group, click the Get Data dropdown arrow, then hover over From File, then click the From Excel Workbook option, as shown below:



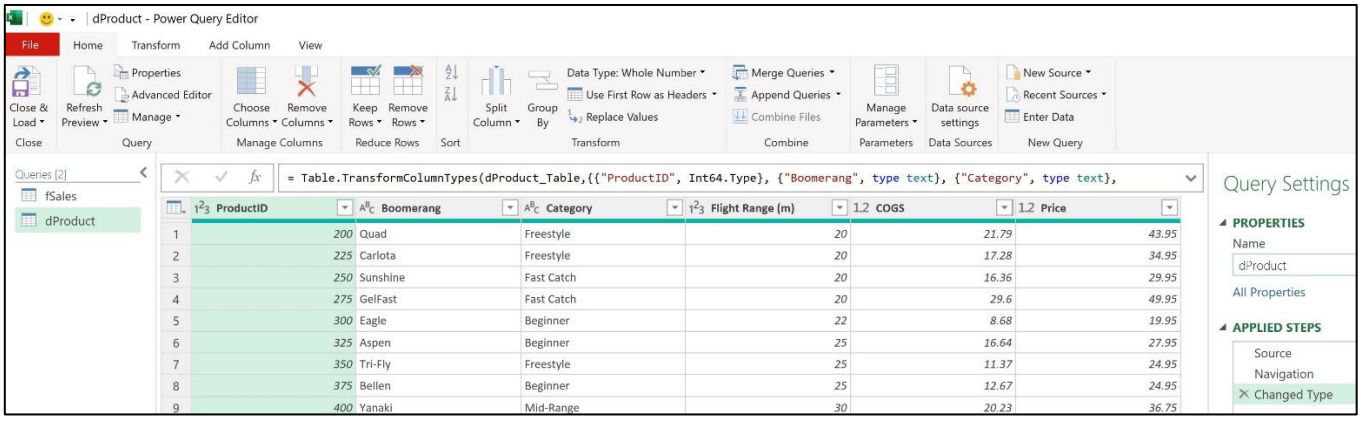
2. Navigate to the Excel file with the name “PowerQuery-02.csv”, select the file, then click the Insert button, as shown below.as shown below:



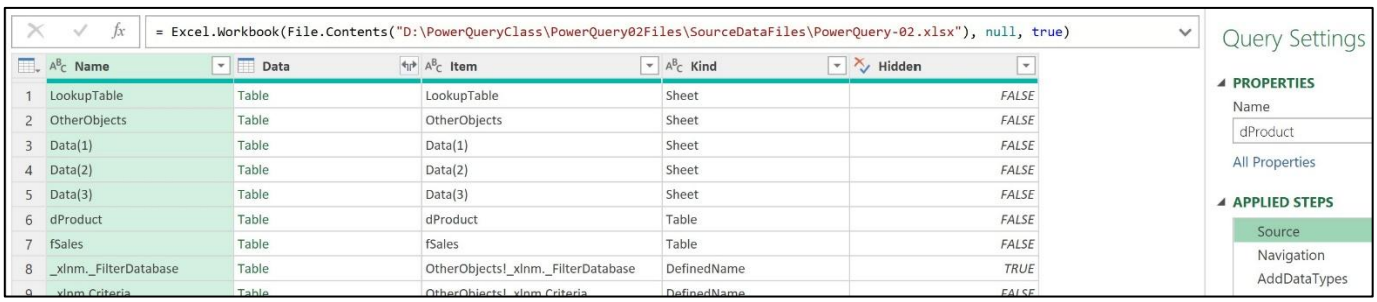
3. When you import data from a source with more than one object, you will see the Navigator dialog box. As shown below, the Navigator dialog box shows all the objects in the Excel file. On the left near the top, check the check box “select Multiple items,” then check the check box for the Excel Table object “dProduct”. Click the Transform Data button (not Load button).



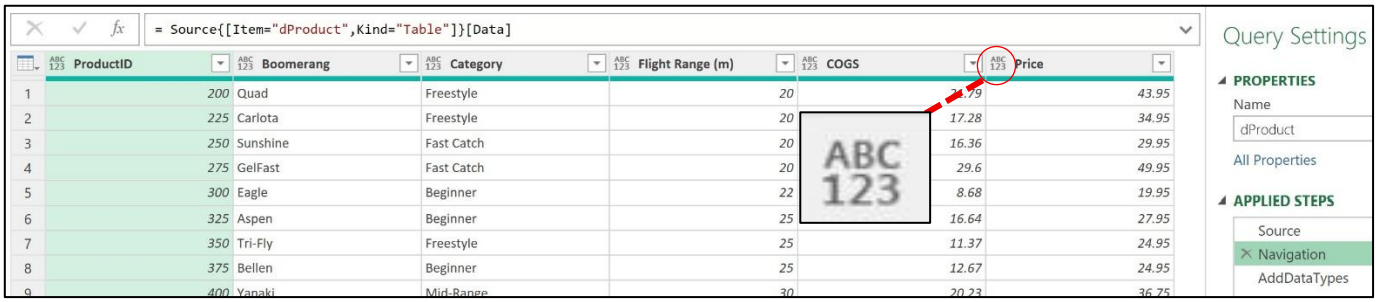
4. You will see the Power Query Editor window, as shown below. Notice that in the Queries Pane on the left, there are now two queries in this Excel file. The name for the query is dProduct – that is a good name so we will keep it. The “d” prefix stands for dimension table (lookup table). In the Applied Steps list there are three automatic steps. Rename the last step from Changed Type to “AddDataTypes”.



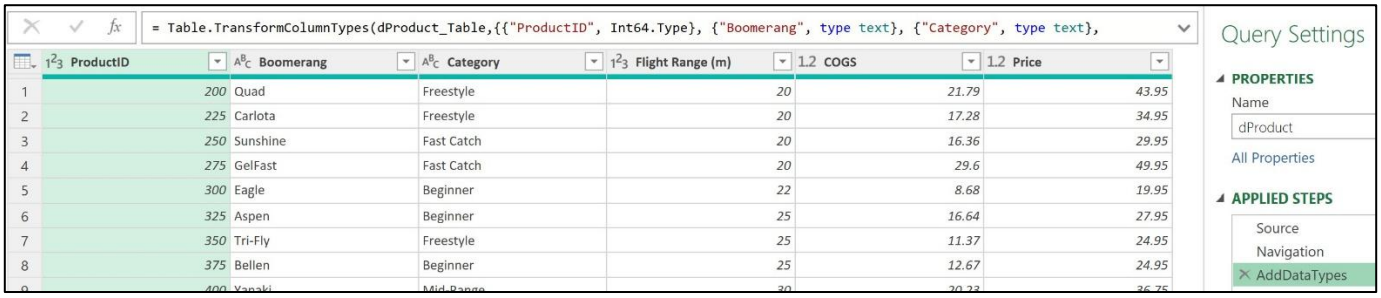
5. In the Applied Steps list, select the Source step. As shown below, in the formula bar you can see the M Code functions Excel.Workbook and File.Contents, as well as the On-Premises File Path deliver all the objects from the Excel file.



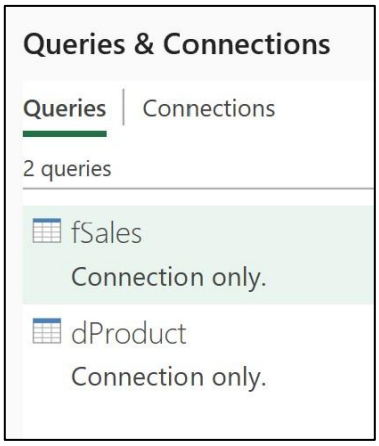
6. In the Applied Steps list, select the Navigation step. As shown below, the formula bar displays the M code two-way lookup formula, which gets the dProduct table from the list of Excel file objects. We will learn how to do this in a later video. Notice that there are no data types applied at the top of the columns. You can tell because each column's data type icon is ABC/123, which indicates that no specific data type has been assigned. However, unlike the first query, where the values came from a text file and had the default left alignment to indicate text values, the number values in this table have the default right alignment, indicating that they are being interpreted as number values. This happens because the Excel Table data source has already defined those values as numbers.



7. In the Applied Steps list, select the AddDataTypes step. As shown below, in the formula bar you can see the Table.TransformColumnTypes M Code function. Even though the data appeared to have the correct default alignment to indicate data types, we added data types in the final step to ensure data consistency. If any values were inconsistent with the assigned data types, Power Query would return an error, which would help us track down the issue.

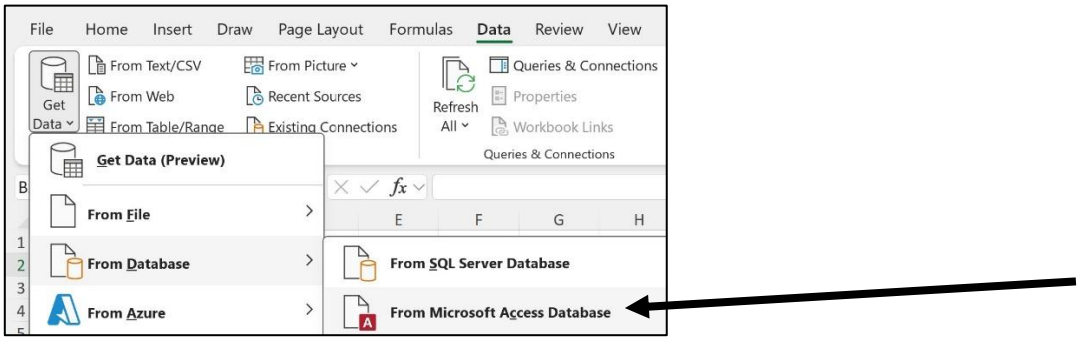


8. Load the dProduct query as a Connection Only query. In the Queries & Connections Task Pane, you will see two queries loaded as Connection Only.

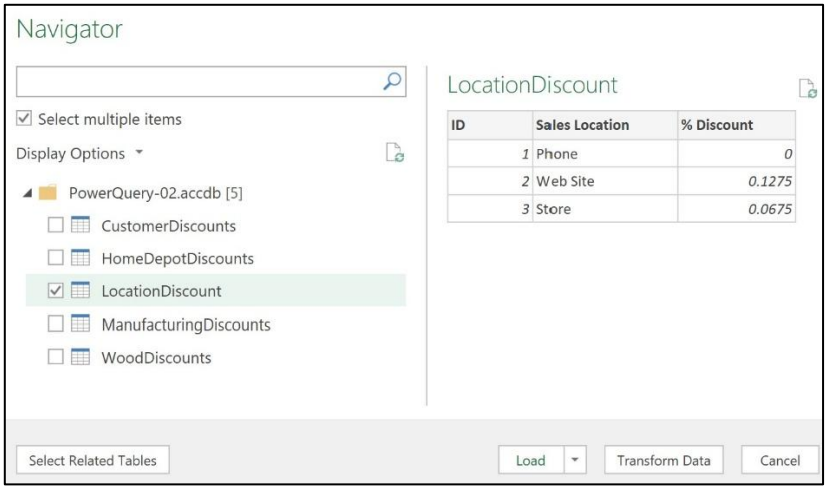


5) Import: Table From Access Database

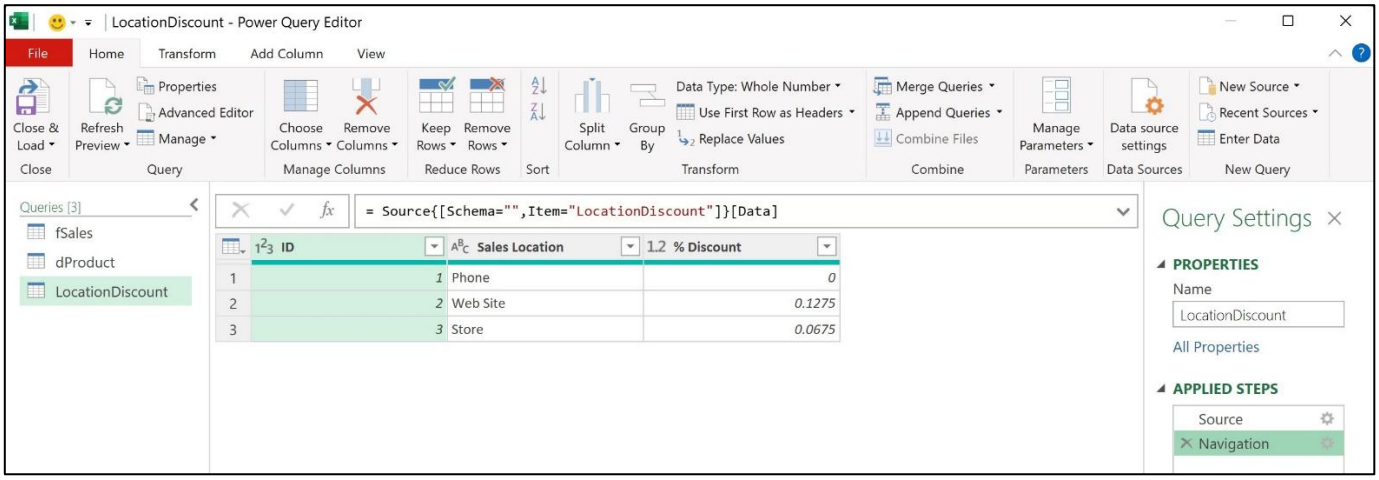
1. To import a table from within an Access database file, in the Data tab in the Excel Ribbon, in the Get & Transform group, click the Get Data dropdown arrow, then hover over From Database, then click the From Excel Microsoft Access Database option, as shown below:



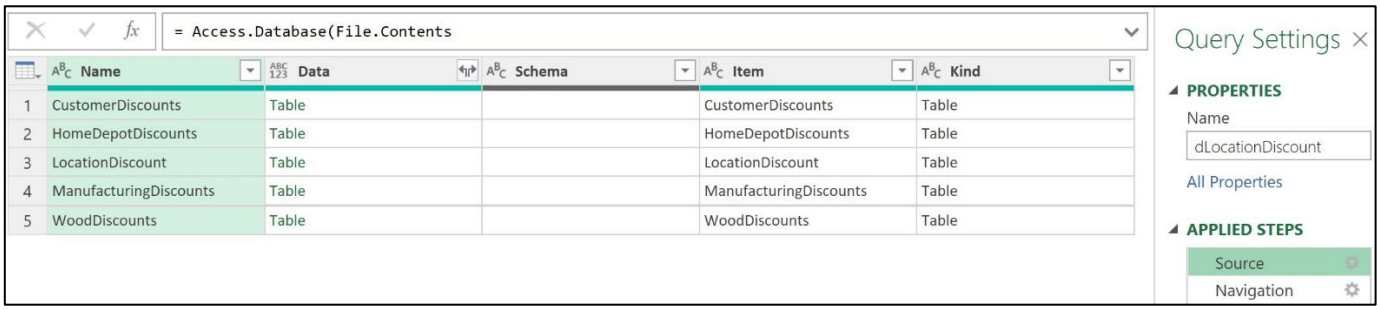
2. As shown in the picture below, because the Access database has multiple objects, the Navigator dialog box appears to let you decide which objects to import. Check the check box "select Multiple items," then check the check box for the Access table object, LocationDiscount. Click the Transform Data button (not Load button).



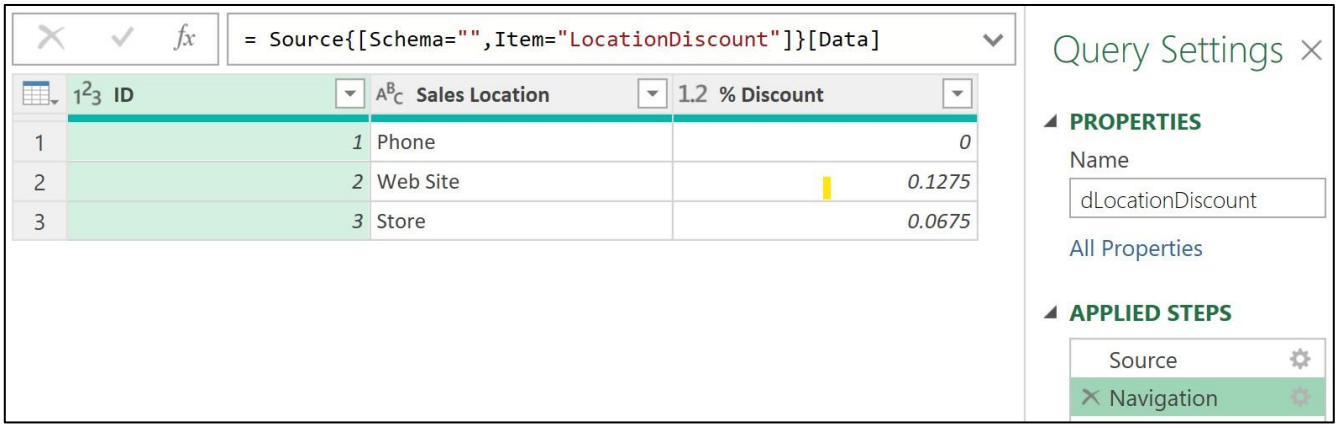
3. As shown in the picture below, the Queries pane on the left shows that this Excel file now contains three queries. On the right, the Query Settings pane shows two automatically created steps under Applied Steps. In the Name box, the query still has the default name inherited from the Access database. For consistent naming, rename the query with a "d" prefix to indicate that it is a dimension/lookup table.



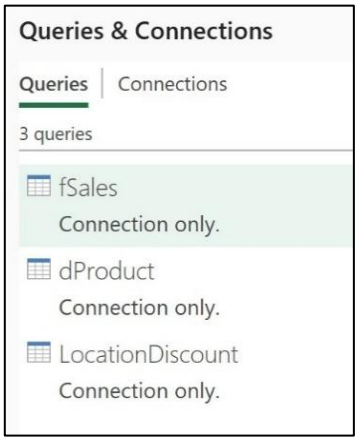
4. In the Applied Steps list, select the Source step. As shown below, in the formula bar you can see the M Code functions Access.Database and File.Contents deliver all the objects from the Access file.



5. In the Applied Steps list, select the Navigation step. As shown below, the formula bar displays the M code two-way lookup formula, which gets the LocationDiscount table from the list of Access file objects. We will learn how to do this two-way lookup from a table in a later video. However, notice that the first part of the M Code expression references the table from the previous step using the identifier Source, then everything after Source is a row by column look up to retrieve the LocationDiscount Access database table.



6. Load the dLocationDiscount query as a Connection Only query. In the Queries & Connections Task Pane, you will see three queries loaded as Connection Only.



7. Next, we want to open the Power Query Editor and clean and transform the data in the fSales query.

Lookup terminology

Dimension or Lookup Tables:

- Lookup tables always have a unique identifier.
- The unique identifier is called a **primary key**.
- To ensure accurate look up, primary keys NEVER have duplicate values.

Fact tables:

- A column in a fact table that contains values from a dimension table's primary key is called a **foreign key**.
- Foreign keys can be used as look up values.
- Foreign key columns may contain duplicates.

Lookup/Dimension Table:

One-side Primary Key

ProductID	Boomerang	Price
200	Quad	43.95
225	Carlota	34.95
250	Sunshine	29.95
325	Aspen	27.95
375	Bellen	24.95

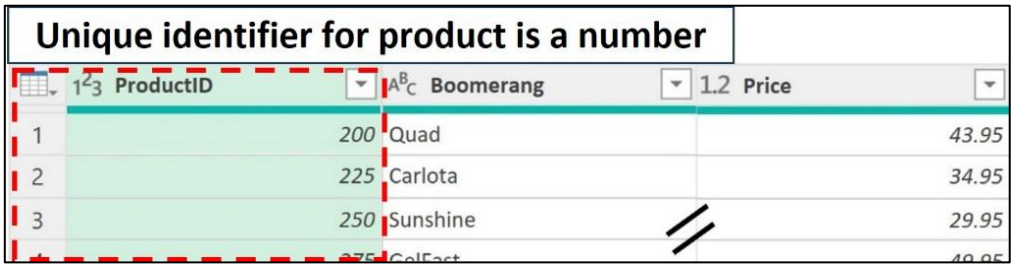
Fact Table:

Many-side Foreign Key

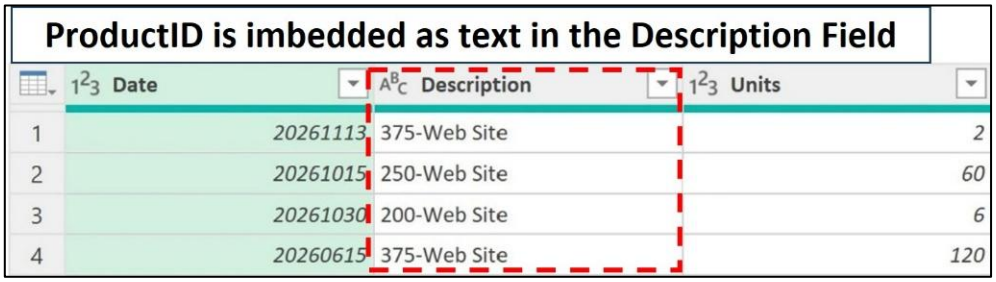
Date	ProductID	Units
10/30/2026	200	5
10/15/2026	250	10
11/13/2026	375	25
6/15/2026	375	100
1/22/2026	200	15
3/20/2026	250	1
3/21/2026	250	10
5/22/2026	250	25
12/23/2026	375	50

6) Clean: Split Description Field in fSales query

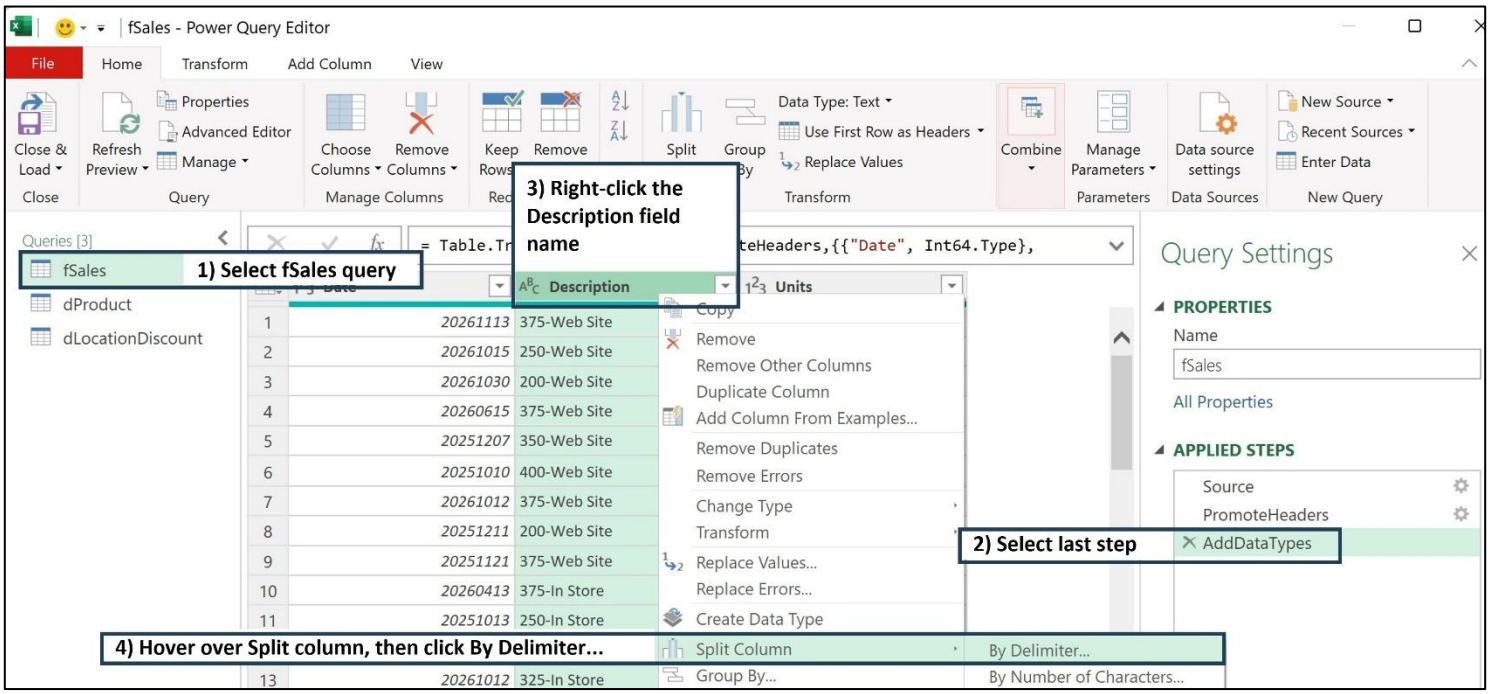
1) As shown in the picture below, the primary key for the dProduct table is the ProductID field. This primary key contains numeric values. To look up product prices in the fSales fact table, we need a foreign key column that stores the ProductID as a numeric value.



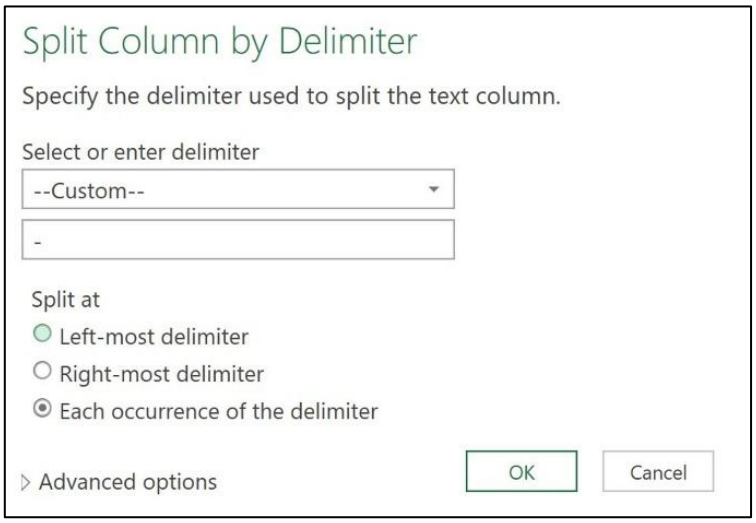
2) As shown in the picture below, the Description field in the fSales table has a data-cleaning problem: the product ID needed for the look up is embedded as text before a dash delimiter. To perform the look up, we need to split the column at the dash delimiter and change the extracted product ID to the Whole Number data type, so the foreign key value matches the ProductID primary key in the dProduct table. We can easily accomplish this using the Power Query Spilt feature.



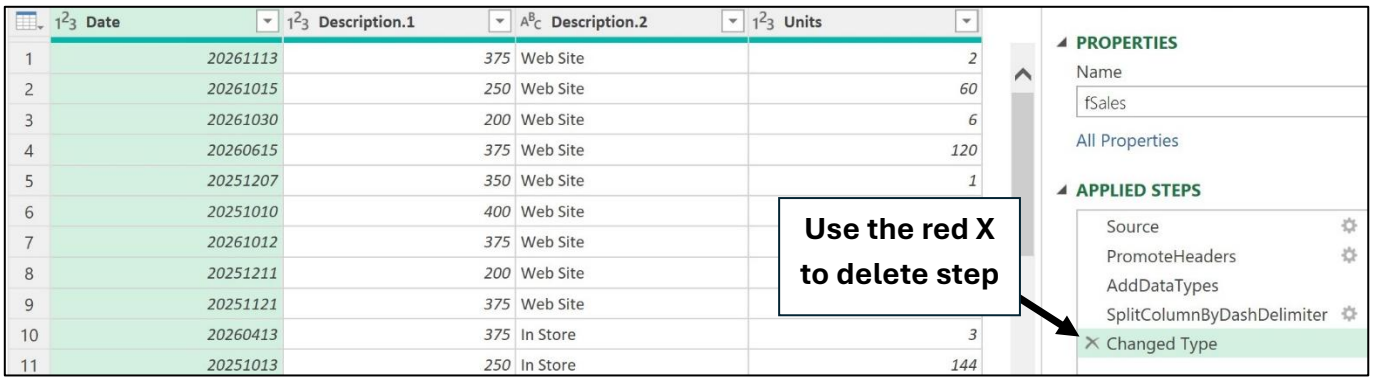
3) From the Excel worksheet, to open the Power Query Editor, you can double click the fSales query name in the Queries & Connections task pane, or you can use the keyboard **Alt + F12**.
4) As shown in the picture below, complete the following steps:
    1. In the queries pane, select the fSales query.
    2. In Applied Steps, select the last step.
    3. Right-click the Description field name.
    4. From the menu, hover over Split Column, then click By Delimiter... from the submenu.



5) As shown in the picture below, in the Split Column by Delimiter dialog box, from the Select or enter delimiter dropdown arrow, select --Custom--, then in the text box below type a dash, and then click OK.

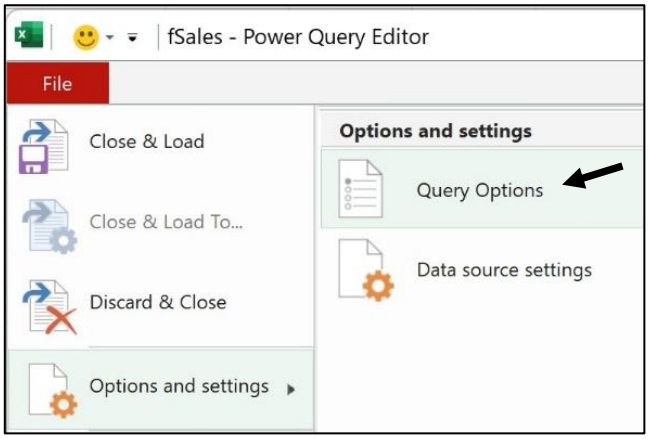


6) As shown in the picture below, in Applied Steps, two steps were added. We do not need the Changed Type step, so to delete the step, click the red X mark to the left of the Changed Type name. If your computer did not add the Changed Type step, then you do not need to delete this step. Then rename the Split Column by Delimiter step so there are no spaces.

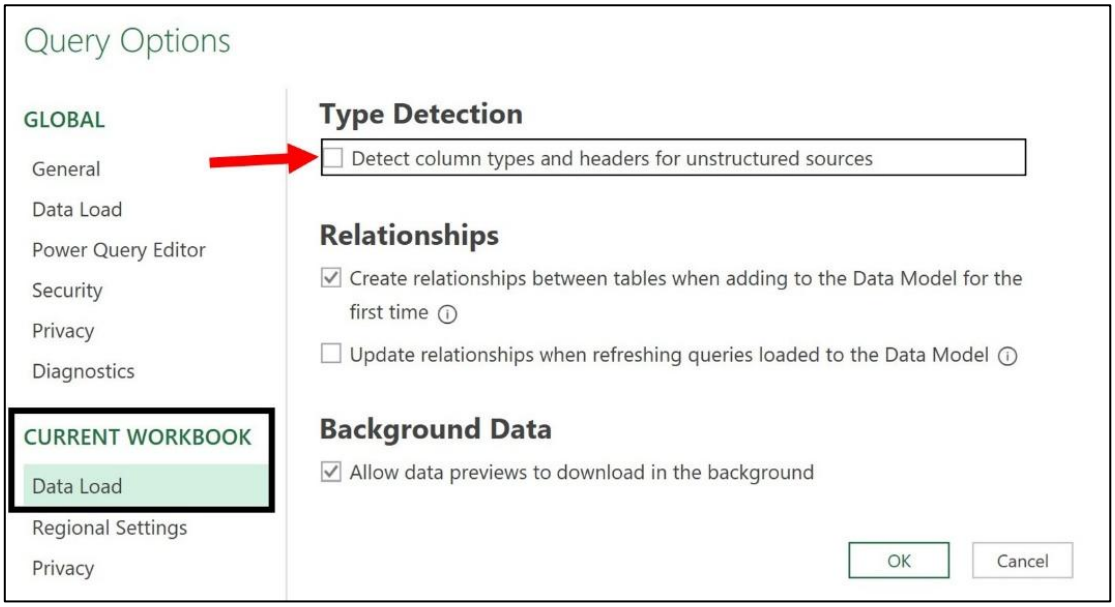


Query Options to Turn Off Automatic Data Types

- If you do not want the automatic step to add data types in the current Excel file, as shown in the picture below, in the Power Query Editor, click the File menu, hover over Options and settings, then click Query Options.

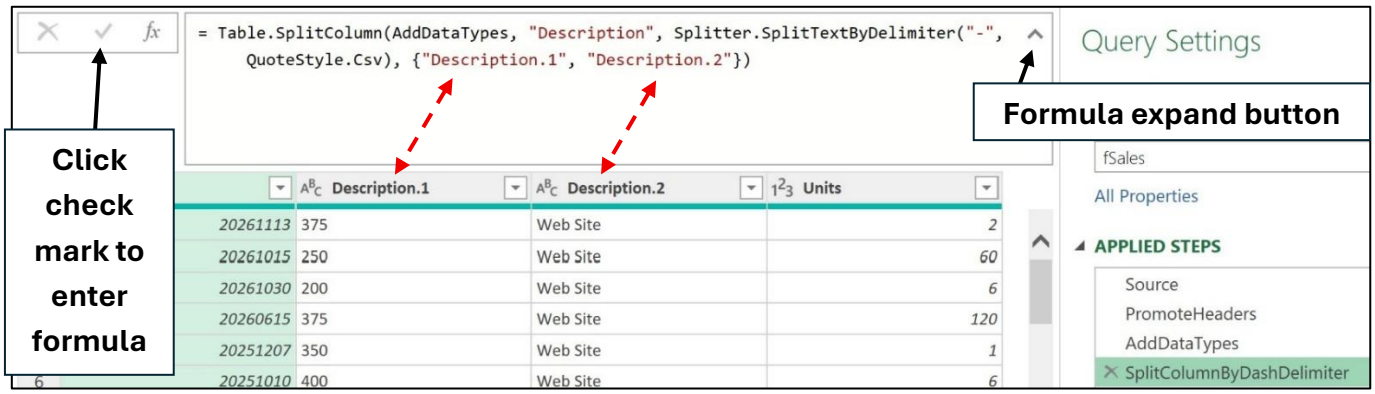


- As shown in the picture below, In the Query Options dialog box, on the left, under Current Workbook, click Data Load, then on the right, uncheck the check box for Detect column types and headers for unstructured sources.

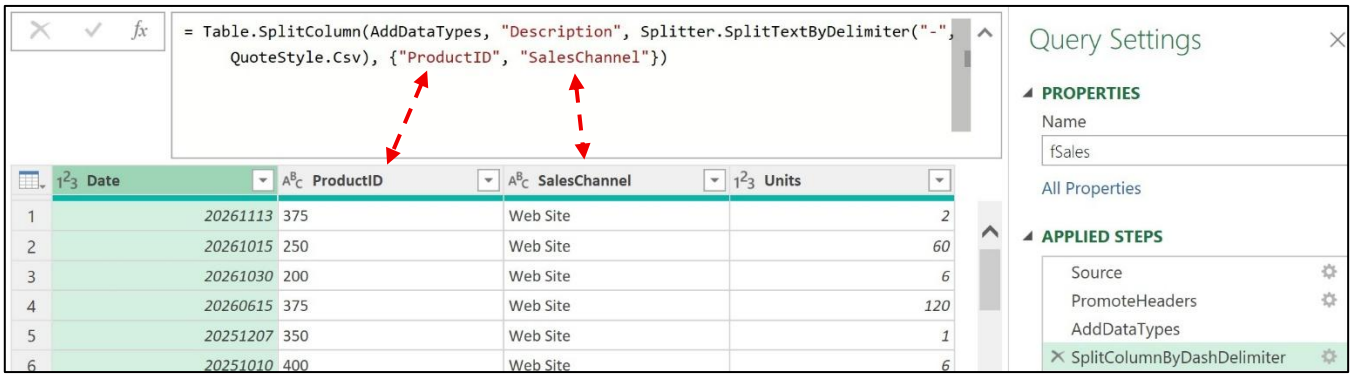


- I have set my options this way, so the change will be reflected throughout the remainder of this video.

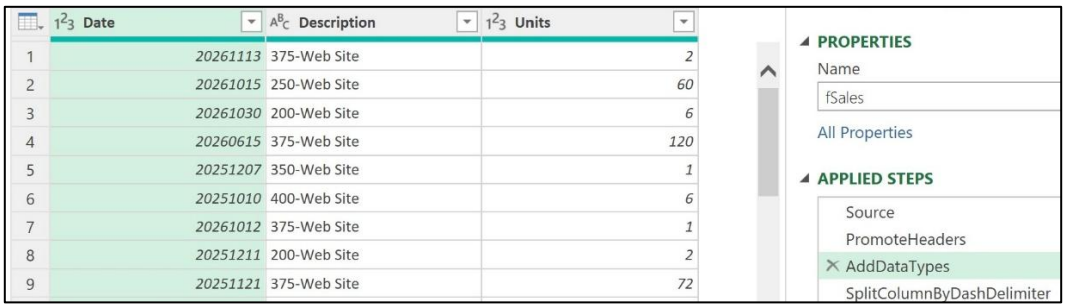
7) You will learn many of the details of M Code as you progress through this class, however, when you are learning Power Query and M Code for the first time, you can often use common sense to understand how functions work and what each argument does. As shown in the picture below, click the formula expand button to see the full formula. Notice that the fourth argument of the Table.SplitColumn function contains two text values that match the names of the two new columns. From this, you can infer that the purpose of the fourth argument is to name the new columns. Rather than adding an extra step to rename the columns, you can edit the M Code directly in the formula bar by selecting each name and typing a more informative name. Because this argument expects a list of text values, be sure to keep the opening and closing curly brackets, which are the M Code syntax for a list value. Also make sure each new name remains in quotation marks, so Power Query receives valid text values. To the left of the formula bar, use the check mark to enter the edited formula. When editing in the formula bar, the check mark is more reliable than the Enter key.



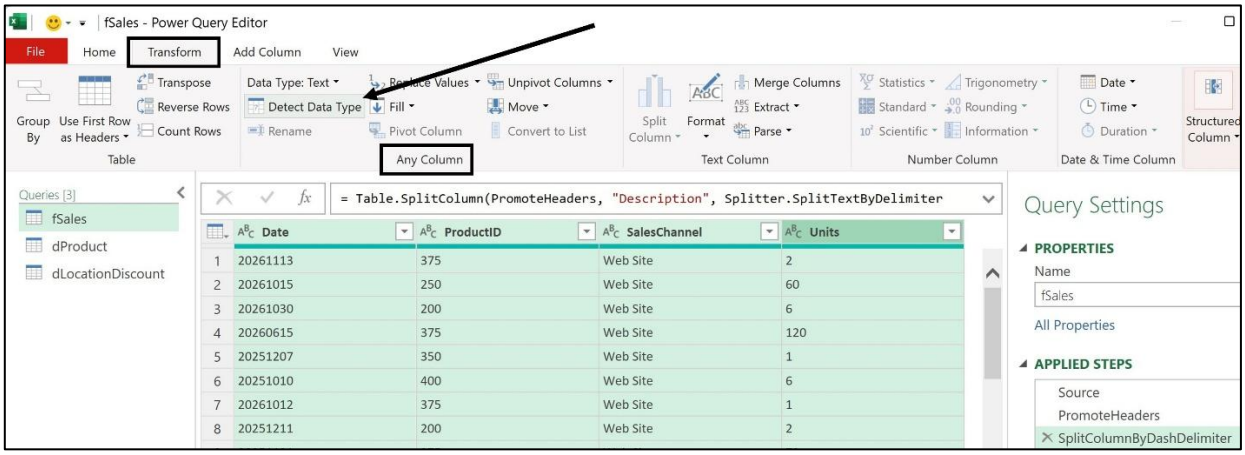
8) After you click the check mark, as shown in the picture below, the columns have new column names. Notice that the ProductID column contains text values aligned to the left. To use those values as lookup values, we must assign the Whole Number data type to the column. We could add a new step to set the data types, but then we would have two separate query steps for data type changes. Although that is sometimes the best method, in this case we can delete the previous AddDataTypes step and add all the data types in one step after the SplitColumnByDashDelimiter step. By removing unnecessary or repetitive query steps, the query tends to be executed more efficiently. However, before deleting any step, make sure it will not adversely affect later steps.



9) Select the AddDataTypes step, review the three data types that were added to the three columns, and ask yourself: “If I remove these data types, will that interfere with later steps?” As shown in the picture on the next page, in this situation, deleting the AddDataTypes step will not affect the SplitColumnByDashDelimiter step. To delete the step, click the red X to the left of the step name.

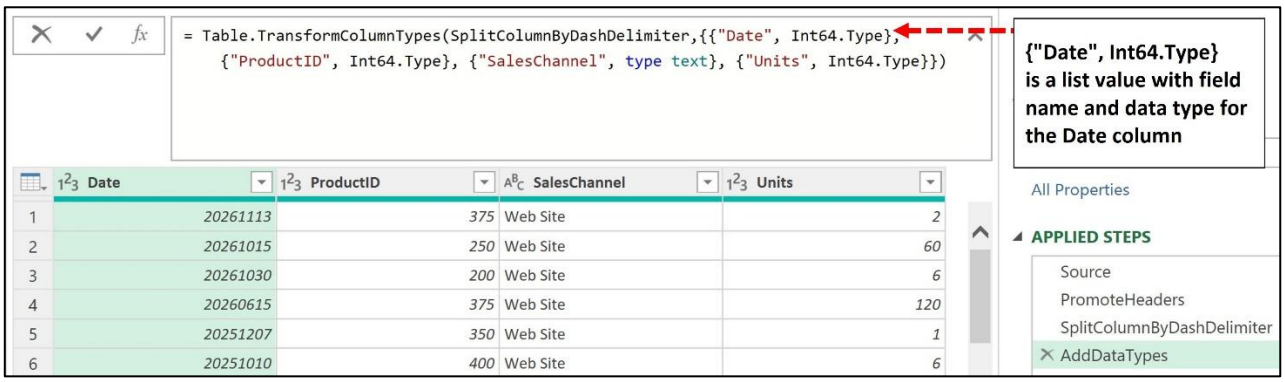


10) As shown in the picture below, with the SplitColumnByDashDelimiter step selected, select the four columns. To do this, click the Date column header, hold down the Shift key, and then click the Units column header. Next, go to the Transform tab in the Power Query Ribbon. In the Any Column group, click Detect Data Type. This button usually identifies the correct data types. If it does not, you can click the data type icon at the top of each column and set the data types manually. Name the new step AddDataTypes.

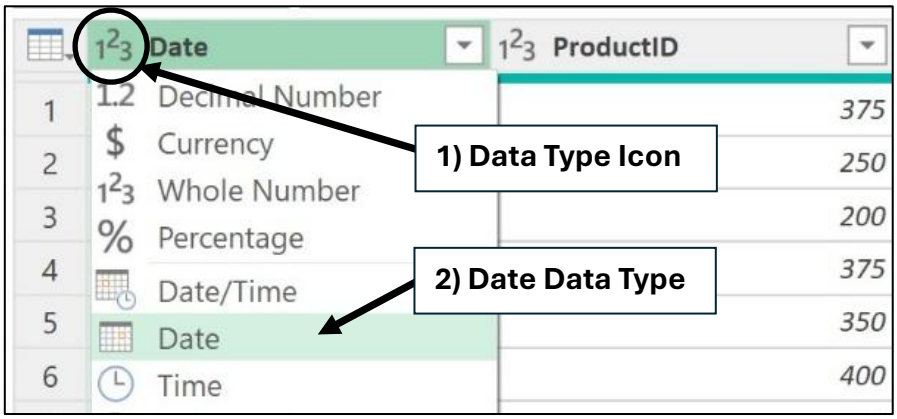


7) Clean: Convert ISO Dates to Real Serial Number Dates

1. It is not uncommon to receive dates in ISO format, which lists the four-digit year, two-digit month, and two-digit day from left to right. For example, 20261113 represents 11/13/2026, or November 13, 2026. However, when this value is stored as a number or text value, it is not a true date with an underlying serial number, so it cannot be used in date calculations, such as subtracting one date from another to calculate the number of days between them. As shown in the picture below, the values in the Date column are stored as numbers. Also, notice that the second argument in the Table.TransformColumnTypes function contains a list within a list with pairs of field names and the data type for each column. For the Date column the current data type is Int64.Type which is the **M Code type value** for the whole number data type.

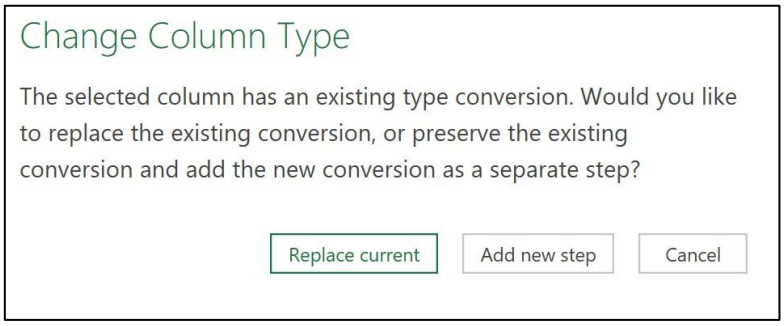


2. Whether the ISO date is text or a number, as shown in the picture below, to convert the ISO date to a real date: 1) Click the data type icon and 2) Select Date from the menu.

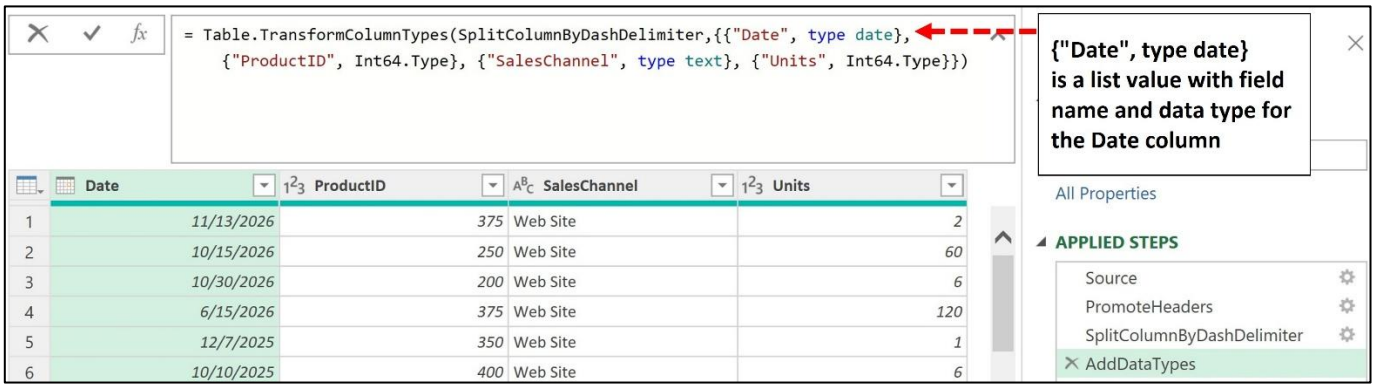


Note: In earlier version of Power Query this method for converting ISO into true dates only worked if the ISO date values were text. But in later versions of Power Query, this method works on text or numeric values.

3. As shown in the picture below, in the Change Column Type dialog box, click the Replace current option to replace the whole number data type with a date data type.



4. As shown in the picture below, for the Date column the data type is now *type date* which is the M Code type value for a true serial numbers date.

5. 

Clean: Converting International Dates to Local Dates

Later in this class, we will study dates and times—including international date formats—in greater detail. However, because many viewers of my excelisfun videos on YouTube are outside the United States and want to follow along with the examples, I want to point out the **Using Locale** option for data types.

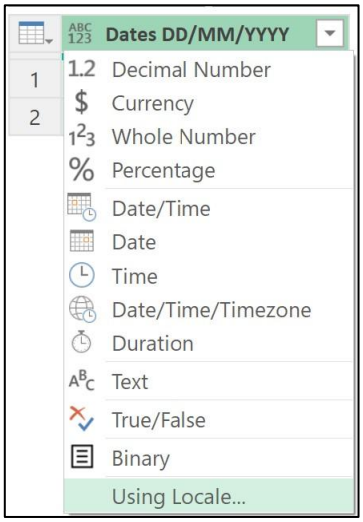
The examples I provide use dates based on the U.S. format: **MM/DD/YYYY**. If you are in France, where dates are commonly formatted as **DD/MM/YYYY**, applying a date data type directly may produce errors. Dates on our computers are interpreted according to the Regional Settings in the Control Panel.

The good news is that the **Using Locale** feature lets us specify the country or region where the dates originated. Power Query can then convert those dates correctly based on the selected locale and your computer's regional settings. In this example, I will show how to take French-formatted dates and convert them to the correct U.S. dates. I do not have a practice file for you to follow along for this example.

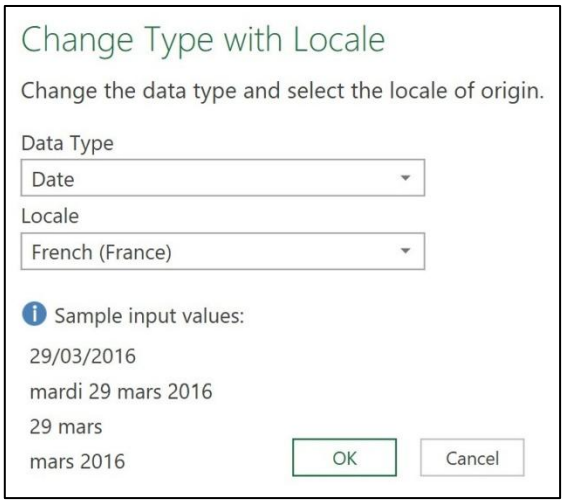
1) As shown in the picture below, the French dates have the default left alignment indicating that they are text values rather than real date values. If I applied a Date data type, I would get an error.



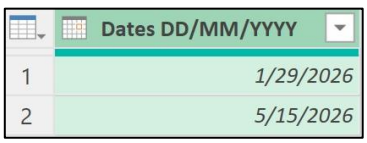
2) Rather than use the Date option from the data type list, as shown in the picture below, I will use the last option, Using Local...



3) As shown in the picture below, I set the Data Type to Date and the Local to French (France), then click OK.



4) By using the Using Local... option, the text dates 29/01/2026 and 15/05/2026 are now the real dates 1/29/2026 and 5/15/2026.



Power Query Merge Feature (Join Operations Used by the Merge Feature)

When making data transformations, join operations using the **Merge feature** in Power Query are commonly used to merge tables. A **join operation** connects two tables based on a primary key column (usually in a lookup or dimension table) and a foreign key column (usually in a fact table) to merge the two tables into a new table. The primary key table has a primary key column with a unique list of values, and the rest of the columns contain attributes for each value in the primary key.

You have probably already experienced a join operation like the left outer join in in the picture below. This figure shows a Product dimension table as an example of a primary key table. The ProductID column is the primary key, and the attribute columns are Product, Cost, and Price. The figure also shows a Sales Transaction fact table as an example of a foreign key table. The ProductID column is the foreign key, and the remaining columns provide details about the transactions, such as date of sale, customer ID, and units sold. In this left outer join, all records from the left are joined with only matching Price column records from the right to create a new merged table.

Left Table: Sales Transaction Table — **Foreign Key** (Many Side)

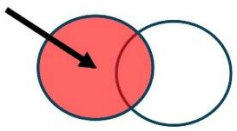
Date	ProductID	CustomerID	UnitsSold
2/18/2024	Q69	SW01	168
1/29/2024	C43	ITW02	144
2/6/2024	C43	PCC01	120
1/5/2024	Q69	PCC01	120

Right Table: Product Table — **Primary Key** (One Side)

ProductID	Product	Cost	Price
C43	Carlota	11.50	26.00
S11	Sunshine	9.50	19.95
Q69	Quad	22.00	43.00

Left Outer Join:

(All rows from the left only matching rows from the right)

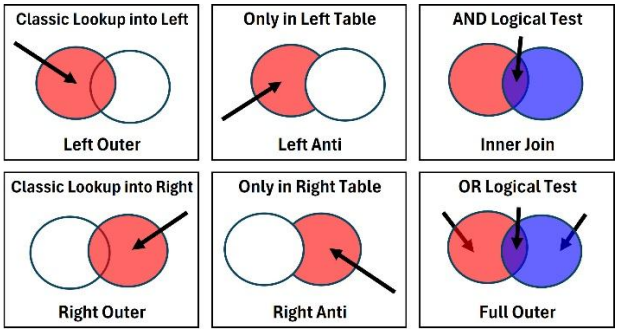


New Merged Table:

Using Left Outer Join on Product ID Column:

Date	ProductID	CustomerID	UnitsSold	Price
2/18/2024	Q69	SW01	168	43.00
1/29/2024	C43	ITW02	144	26.00
2/6/2024	C43	PCC01	120	26.00
1/5/2024	Q69	PCC01	120	43.00

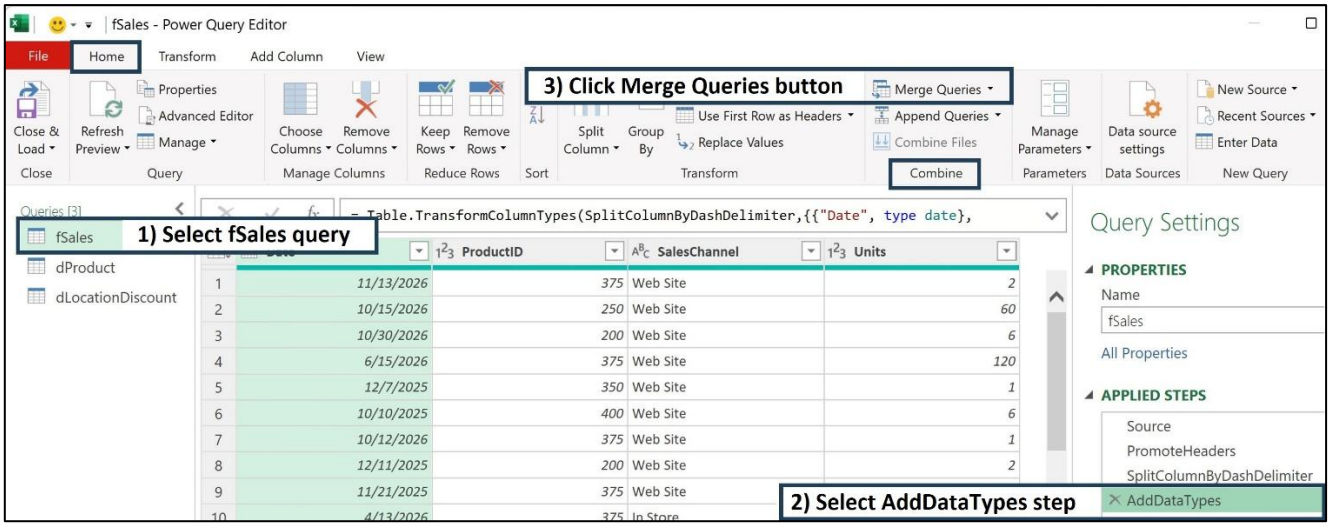
The figure below uses Venn diagrams to illustrate the six types of joins commonly used in data analysis to determine the structure of the merged table. The Venn diagram in the picture above illustrates the left outer join occurring in this example by highlighting the full circle on the left and only the intersection of the left and right circles on the right. Only product prices that have product IDs on the left are in the merged table. The one Sunshine record on the right has no matching product ID on the left, so the price for that product is not in the merged table.



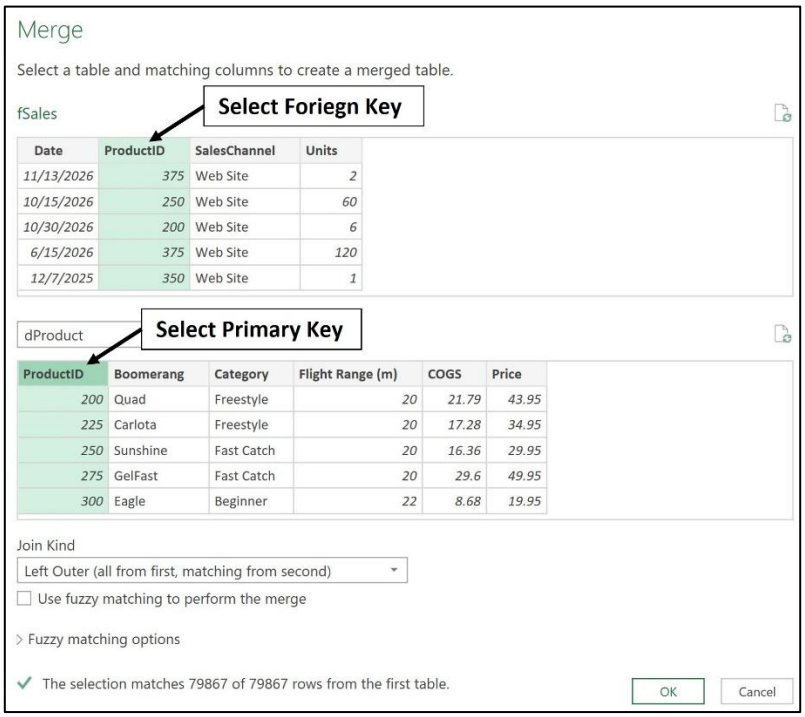
In this video (video #2), we will use the Merge feature with a left outer join to look up the product name and price from the dProduct table and return those items to the fSales table. We will use the same type of merge and join to look up the sales discount from the dLocationDiscount table and return it to the fSales table.

8) Transform: Use Merge Feature to Look up Product Price

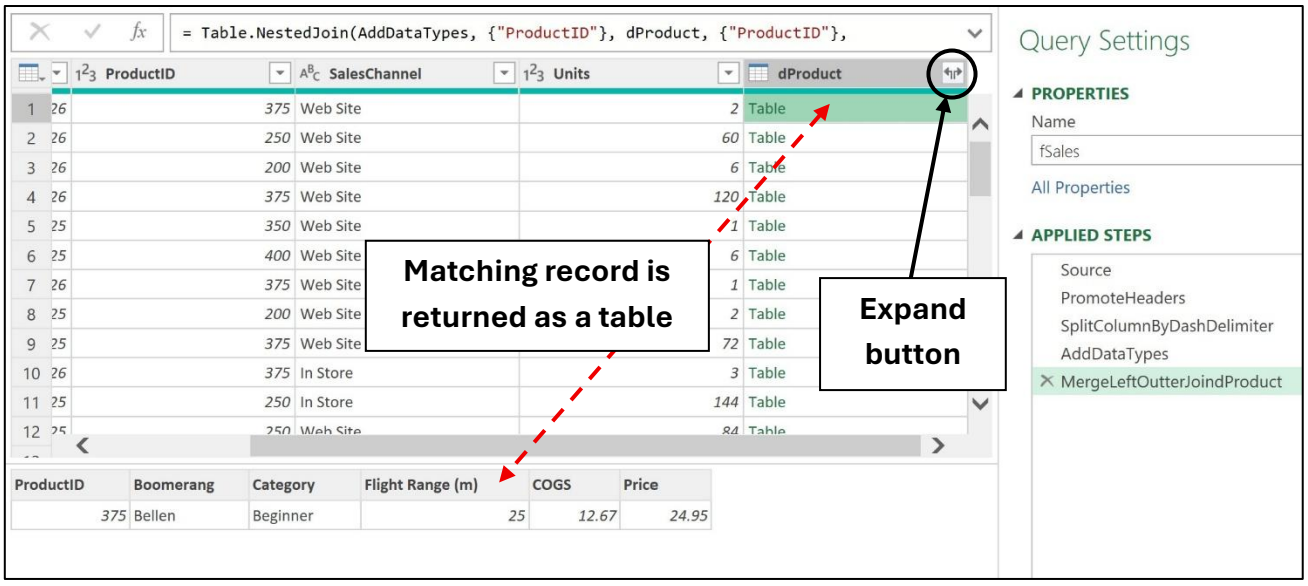
1) To look up the product name and price from the dProduct table and return those items to the fSales table, as shown in the picture below, on the left in the Power Query Editor, in the Queries Pane, select the fSales query. On the right, select the last query step in the Applied Steps list. Then go to the Home tab in the Power Query Ribbon, and in the Combine group, click the Merge Queries button (not the dropdown arrow).



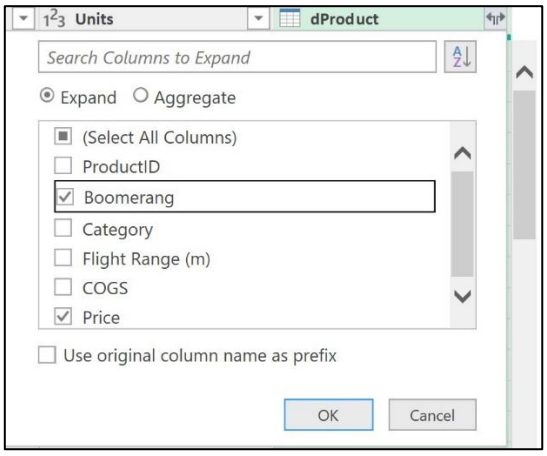
2) When the Merge dialog box appears, the fSales table is automatically placed as the top, or left, table. To identify the foreign key, select the ProductID field in the top table. Then, from the dropdown list below the top table, select the dProduct table. The dProduct table is considered the bottom, or right, table. To identify the primary key, select the ProductID field in the bottom table. From the Join Kind dropdown list below the bottom table, select Left Outer (all from first, matching from second). Notice that the message at the bottom says, "The selection matches 79867 of 79867 rows from the first table." This means that every lookup value from the fSales table was found in the dProduct table. Then click OK. The finished dialog box is shown below.



As shown in the picture below, a new column appears with the name dProduct. This column has a table data type and contains a table in each row. To see the table returned in each row, carefully click the green area to the right of the word Table in the first row. If you accidentally click the word Table, the table will be extracted as a new query step, which you do not want. After successfully clicking the green area in the first row, you will see the one matching record from the dProduct table returned as a table. This is remarkable because, in the next step, you can select as many columns as you would like, and they will be returned as individual columns. This is much easier than parallel methods in an Excel worksheet, which require a new formula for each look up. Rename the query step so that the name contains no spaces.



3) To look up the product name and price, click the expand button on the right side of the dProduct field name. As shown in the picture below, in the Expand dialog box, clear the (Select All Columns) and Use original column name as prefix check boxes. Then select Boomerang (product name) and Price. Click OK.

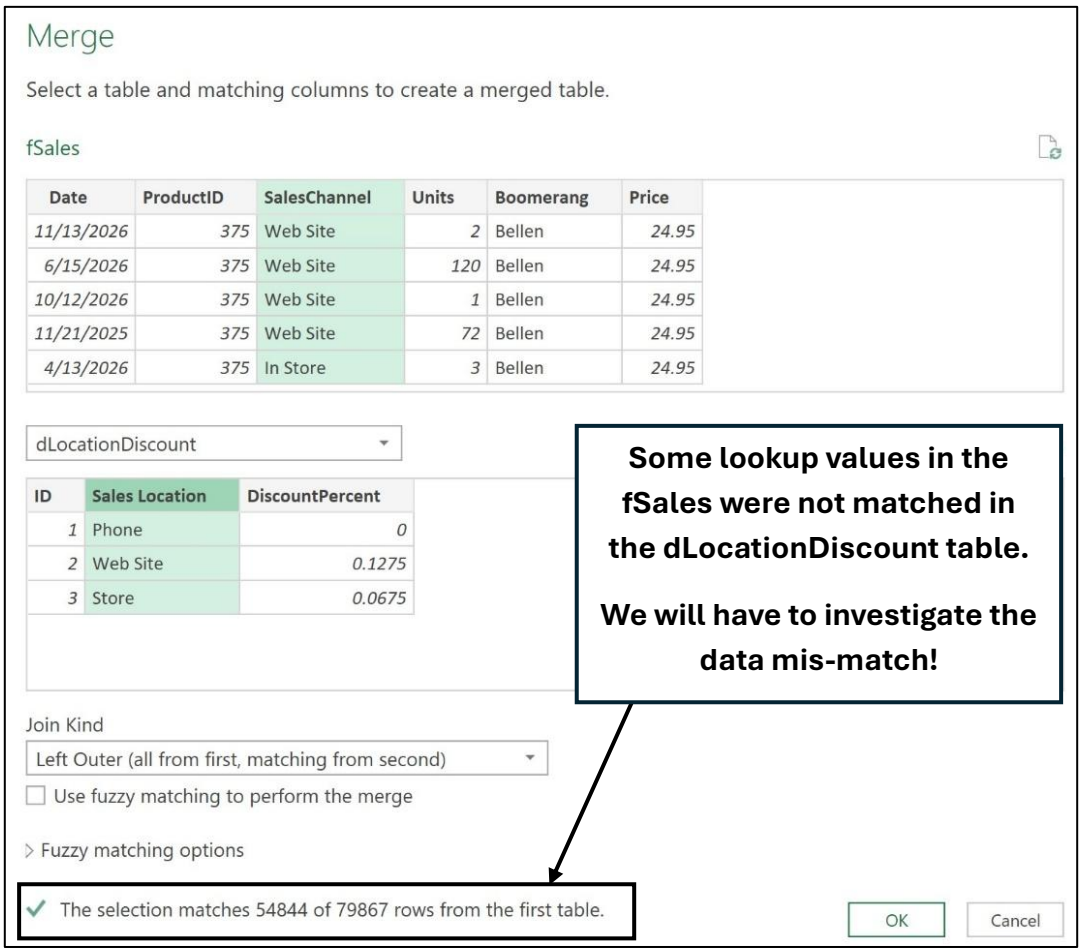


4) As shown in the picture below, You have successfully looked up two items with one merge action. Rename the steps so that there are no spaces.

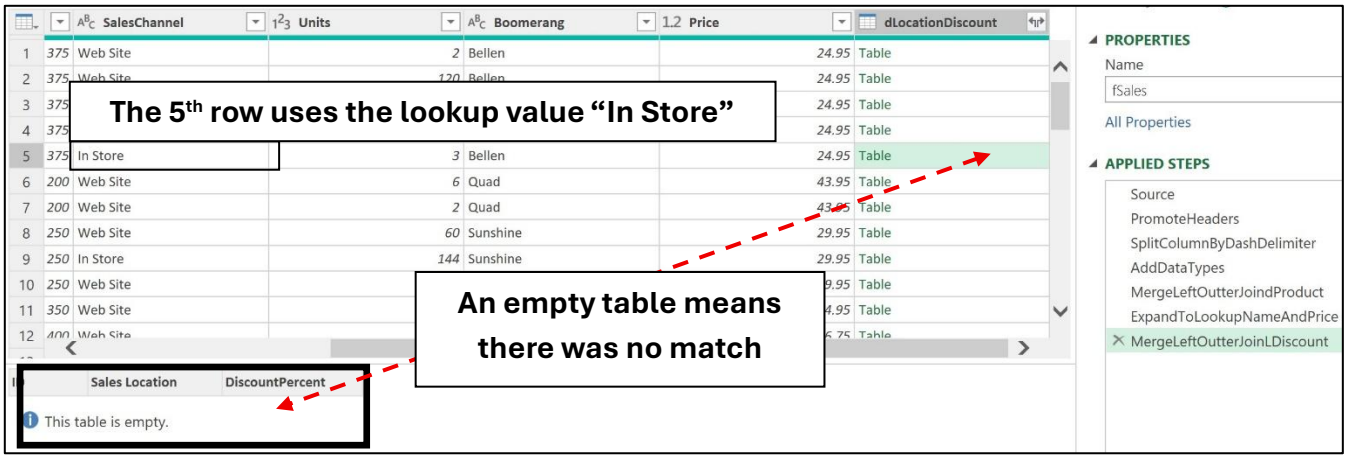


9) Investigating Transform Errors and Fixing Problem

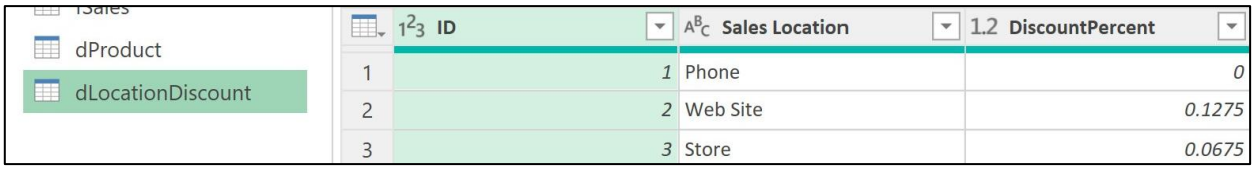
1. Using the same merge and join techniques as you did in the last section, from the fSales table look up the location discount from the dLocationDiscount table using the fSales table foreign key column named SalesChannel and the dLocationDiscount table primary key column named Sales Location. This time, however, there will be a data mis-match problem which is revealed in the Merge dialog box as shown below. The message at the bottom of the dialog box says: "The selection matches 54844 of 79867 rows from the first table." This means that there are 25,023 (79,867 – 54,844) items in the fSales table foreign key column, SalesChannel that are not in the dLocationDiscount table primary key column, Sales Location. We will have to investigate the issue. The fact that the names of the two columns are not the same is not a problem. These columns are allowed to have different names. Click OK in the Merge dialog box. Rename the new query step.



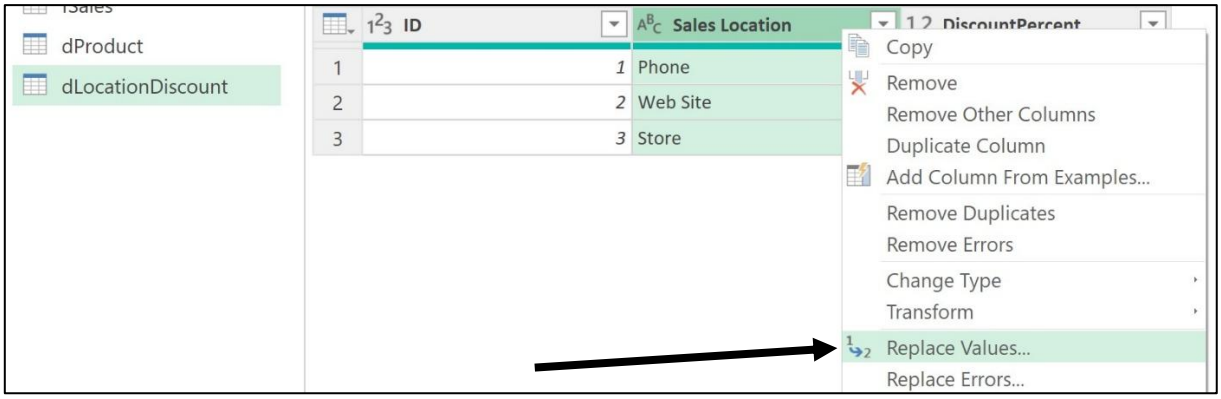
2. To investigate, we will start by examining the table returned in each row of the dLocationDiscount column. As shown in the picture below, the fifth row returns an empty table when using the lookup value “In Store.” We will keep this in mind as we next look at the values in the Sales Location column of the dLocationDiscount table.



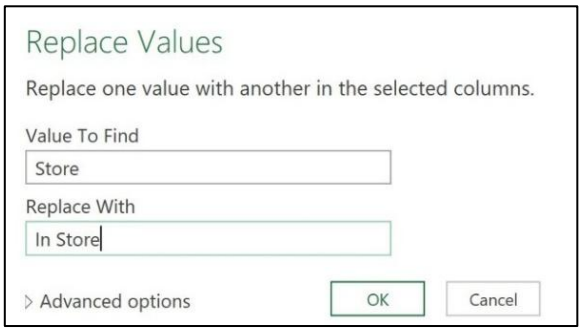
3. As shown in the picture below, in the Queries pane on the left side of Power Query Editor, select the dLocationDiscount table. In the Sales Location column, there is no “In Store” value, but there is a “Store” value. Ideally, you would talk with the people who initiated the data analysis project, determine which table contains the correct value, and then fix the incorrect data so that all lookup values in the foreign key column match the values in the primary key column. In the working world, however, this is not always possible. When it is not possible, the issue becomes a data-cleaning problem in Power Query. You can use the Replace Values feature in either the fSales table or the dLocationDiscount table. Since there is only one value to replace in the dLocationDiscount table, and assuming all “In Store” values in the fSales SalesChannel column are equivalent, let’s replace just the single “Store” with “In Store” in the Sales Location column in the dLocationDiscount table.



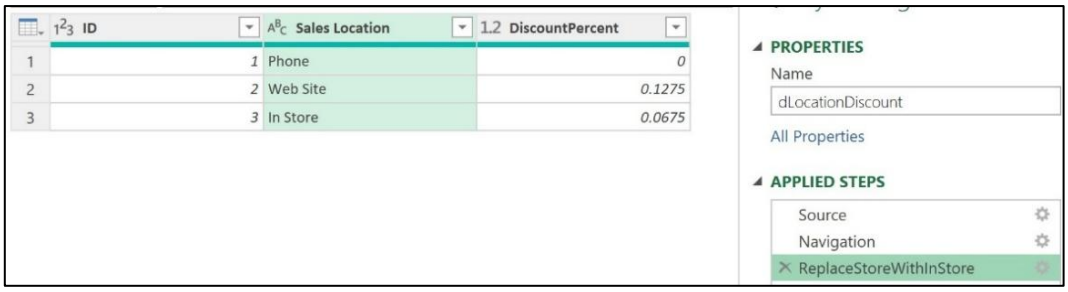
4. As shown in the picture below, Right-click the Sales Location field name and from the menu click Replace Values….



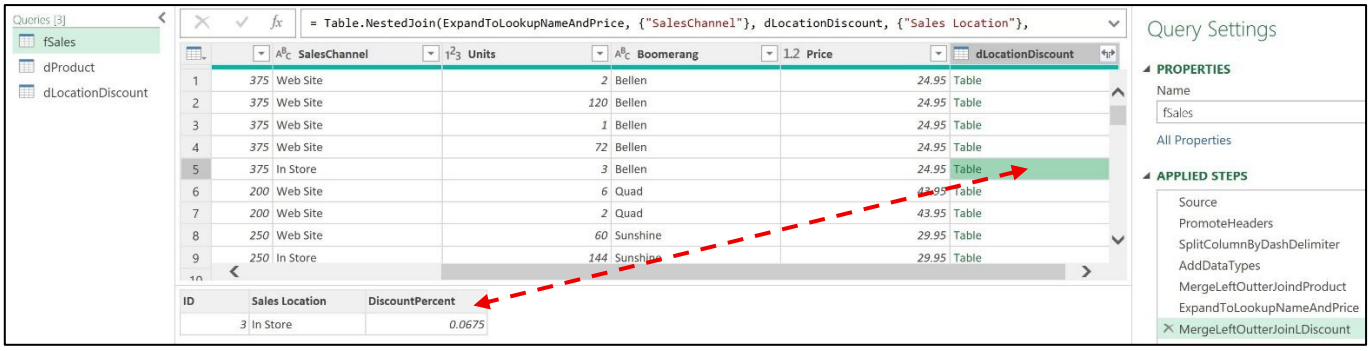
5. As shown in the picture below, type Store in the Value To Find text box, then type In Store in the Replace With text box. Click OK.



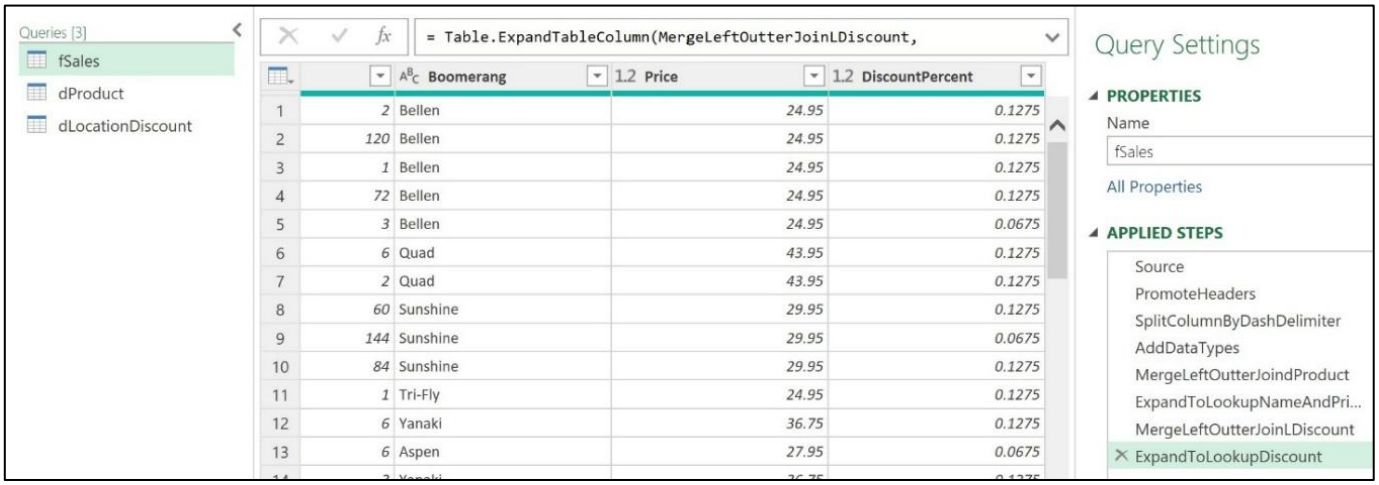
6. Rename the new query step.



7. Navigate back to the fSales query. As shown in the picture below, now the fifth row in the dLocationDiscount column returns a table with the correct matching record.



8. To look up the location discount, click the expand button on the right of the dLocationDiscount column, uncheck everything and check only the DiscountPercent column. Click OK. Rename the new query step. The result is shown in the picture below. Notice that in the fifth row you have successfully retrieved a location discount.



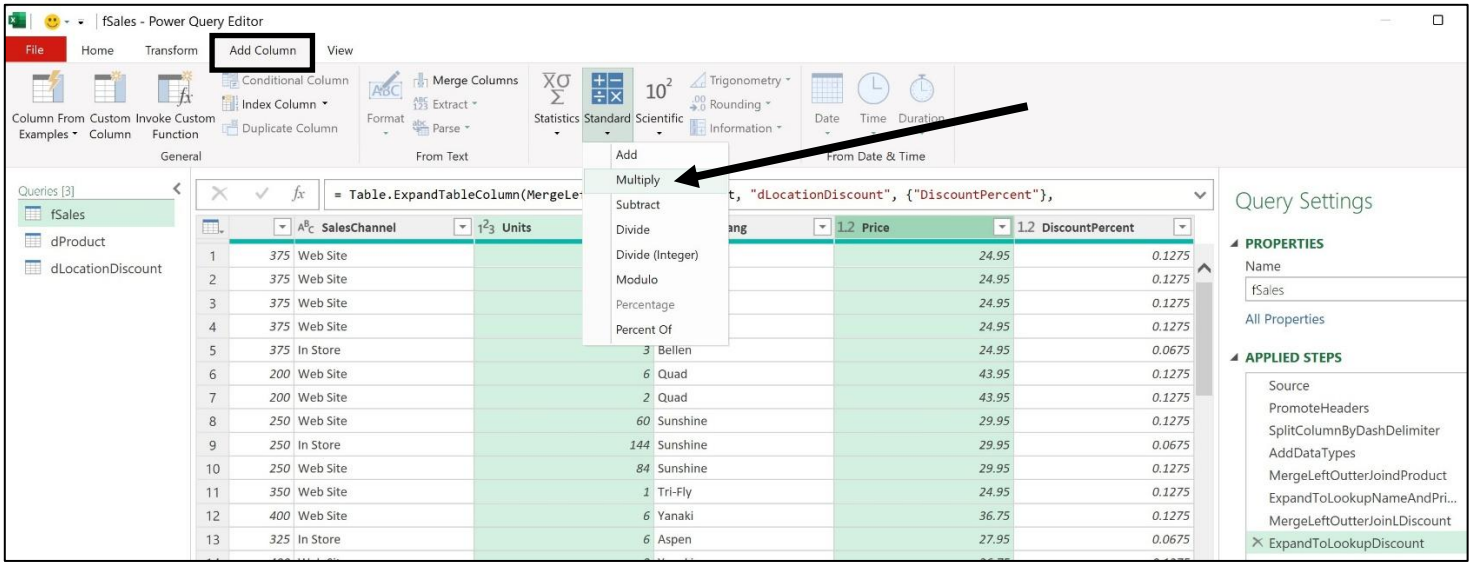
Add Column Tab in Power Query Ribbon

In the Power Query Editor, there are many helpful user interface options we can click, and Power Query writes the M Code for us. Perhaps the most useful options are found in the From Text, From Number, and From Date & Time groups on the Add Column tab of the Power Query ribbon. These options use the Table.AddColumn function to add a column containing a new M Code formula that performs a calculation in each row of the table. Excel worksheet formulas and DAX formulas in the Data Model do not offer a parallel feature that writes column formulas for you. That makes Power Query pretty cool!

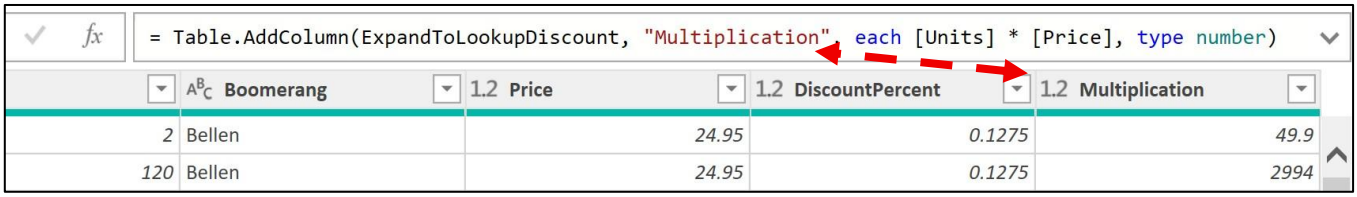
Now that we have looked up the product name, product price, and location discount in previous steps, the next step is to add a new column that multiplies units by price to calculate sales. The From Number group on the Add Column tab is perfect for this.

10) Transform: Multiply Feature to Add New Column

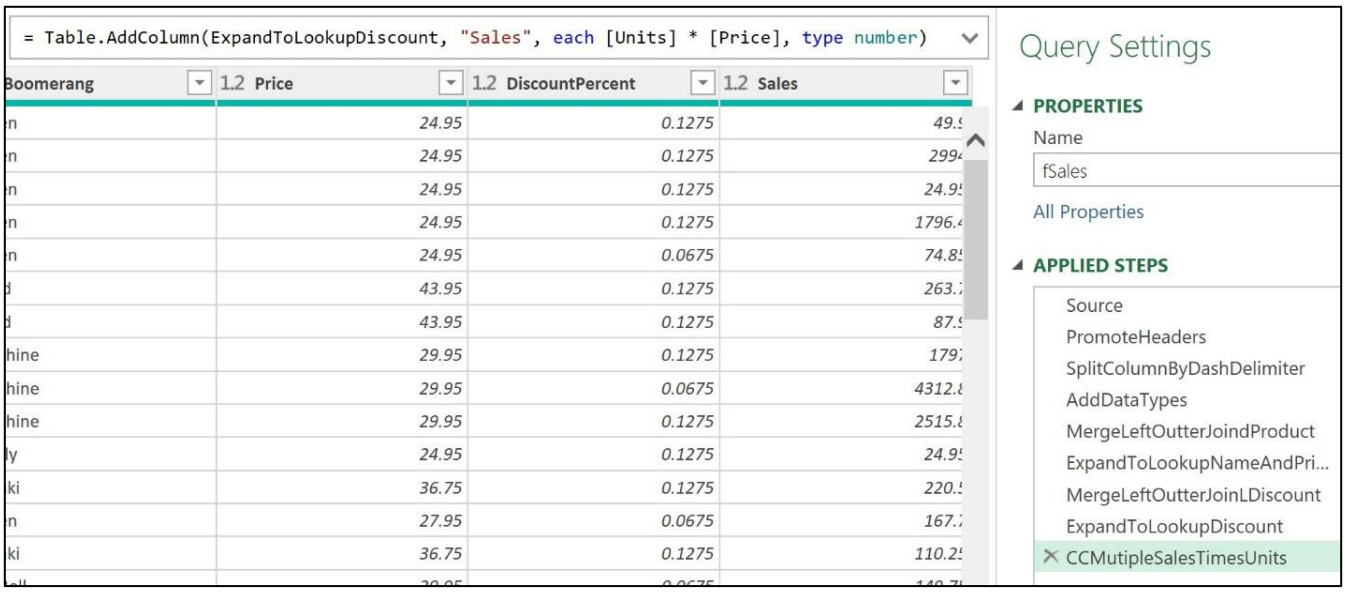
1. With the last query step selected in the fSales query, select the Units and Price column headers. To select both columns, click the Units column header, hold down Ctrl, and then click the Price column header. As shown below, go to the Add Column tab on the Power Query ribbon. In the From Number group, select the Standard dropdown, and then choose Multiply. Rename the new query step.



2. As shown below, you can infer that the second argument in the Table.AddColumn function assigns a name to the new column. Rather than add a new step to rename the column, click in the formula bar, highlight the text “Multiplication”, type the new column name “Sales”, then click the check mark on the left side of the formula bar. Rename the new query step.



3. The fSales table with the new column is shown in the picture on the next page.



4. The next column we need to add to the table is for Net Sales. But before we do that I want to take a closer look at the Table.AddColumn function and Data Types.

Table.AddColumn Function

You will use the amazing Table.AddColumn function often in this video class and in your Power Query career. So we would like to take a closer look at the four arguments in this function. The formula that we just created is shown below with a description of each of the four arguments:

= Table.AddColumn(

ExpandToLookupDiscount,	1st argument: the table you want to add a column to
"Sales",	2nd argument: the new column name as text
each [Units] * [Price],	3rd argument: a custom function (formula for each row)
type number	4th argument: optional; defines data type for new column

The first argument contains the output from the previous step, which is a table value. A table value is required in this argument. The column that is added by the function is added to this table. The second argument is the text value you typed in the formula bar to define the new column name. This argument requires a text value. The third argument requires a formula that will make a calculation in each row. This argument requires a custom function value. We will learn about custom function values in a later video. However, the programmers at Microsoft created the ***each* keyword** as a syntactical shorthand for a custom function, and it makes creating formulas in this third argument quite easy. You simply type the keyword *each*, followed by the formula. The formula that was automatically created is shown here:

each [Units] * [Price]

The *each* keyword is what allows the formula to make a calculation in each row. The square brackets around the field names are called a **Field Access Operator**, and they allow you to access the value in each row of the table.

The fourth argument is optional and allows you to define a **data type value** for the column without adding an extra query step. The following section provides a list of the 20 possible data type values.

Data Types

A list of the 20 Data Types available in Power Query is shown below. The fourth column, the M Code column, shows the M Code data type value that you use to define the given data type.

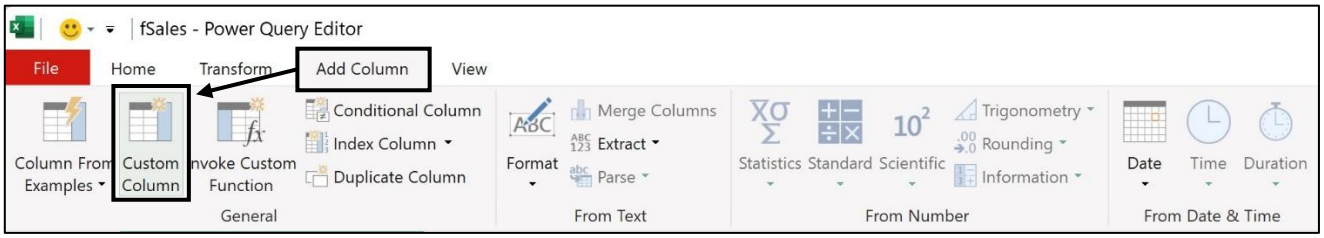
	Data Type	Icon	Short Description	M Code
1	Decimal number	1.2	Number up to 15 decimals.	type number
2	Currency (Fixed decimal number)	$	Number up to 4 decimals.	Currency.Type
3	Whole number	123	Number with no digit to right of decimal. It removes decimals using bankers rounding.	Int64.Type
4	Percentage	%	Number up to 15 decimals with % Number Format.	Percentage.Type
5	Date/Time		Serial number date and time together.	type datetime
6	Date		Serial number date.	type date
7	Time		Serial number time.	type time
8	Date/Time/Timezone		Represents a UTC date/time with a time-zone offset.	type datetimezone
9	Duration		Serial Number Length of Date and Time (not supported in Dataflow).	type duration
10	Text	ABC	Text.	type text
11	True/False		Boolean.	type logical
12	Binary		File like Excel file or Text file.	type binary
13	Any	ABC 123	used it will set numbers such as dates and decimals according to regional settings. No expression will every deliver an any value type, though.	type any
14	nullable		You can add the keyword nullable to data types like number so that the column can have the number or a null. However, many M Code features, such as List.Sum, Custom Functions with the type number, Merge feature, Append feature are programmed to silently ignore nulls. Functions like: Table.ColumnsOfType, Value.Is and Value.As do, for example, treat type number and type nullable number as different. For example, Value.Is(43, type nullable number) = TRUE and Value.Is(43,type null) = FALSE.	type nullable number
15	anynonnull		Any non-null value (all values excluding null).	type anynonnull
16	none		No values are classified. This type is sometimes used in helper functions that raise errors because any other value returned when the type is none will raise an error.	type none
17	Table		Table with one or more fields and one or more records. Example in the Table.AddColumn function: Table.AddColumn(PrevoiusStep, "Table", each #table(type table [Sales = number, Product = text],{{43,"Quad"},{69,"Carlota"}}), type table)	type table
18	Record		Record with one or more fields and one or more values associated with the fields. Example in the Table.AddColumn function: = Table.AddColumn(CCTableColumn, "Record", each _, type record)	type record
19	List		List with one or more items. Example in the Table.AddColumn function: = Table.AddColumn(CCRecordColumn, "List", each {"Quad",43}, type list)	type list
20	Function	fx	Built-in or Custom Function. Example in the Table.AddColumn function: = Table.AddColumn(CCRecordColumn, "Function", each (x) => x +1, type function)	type function

11) Transform: Custom Columns to Add New Column

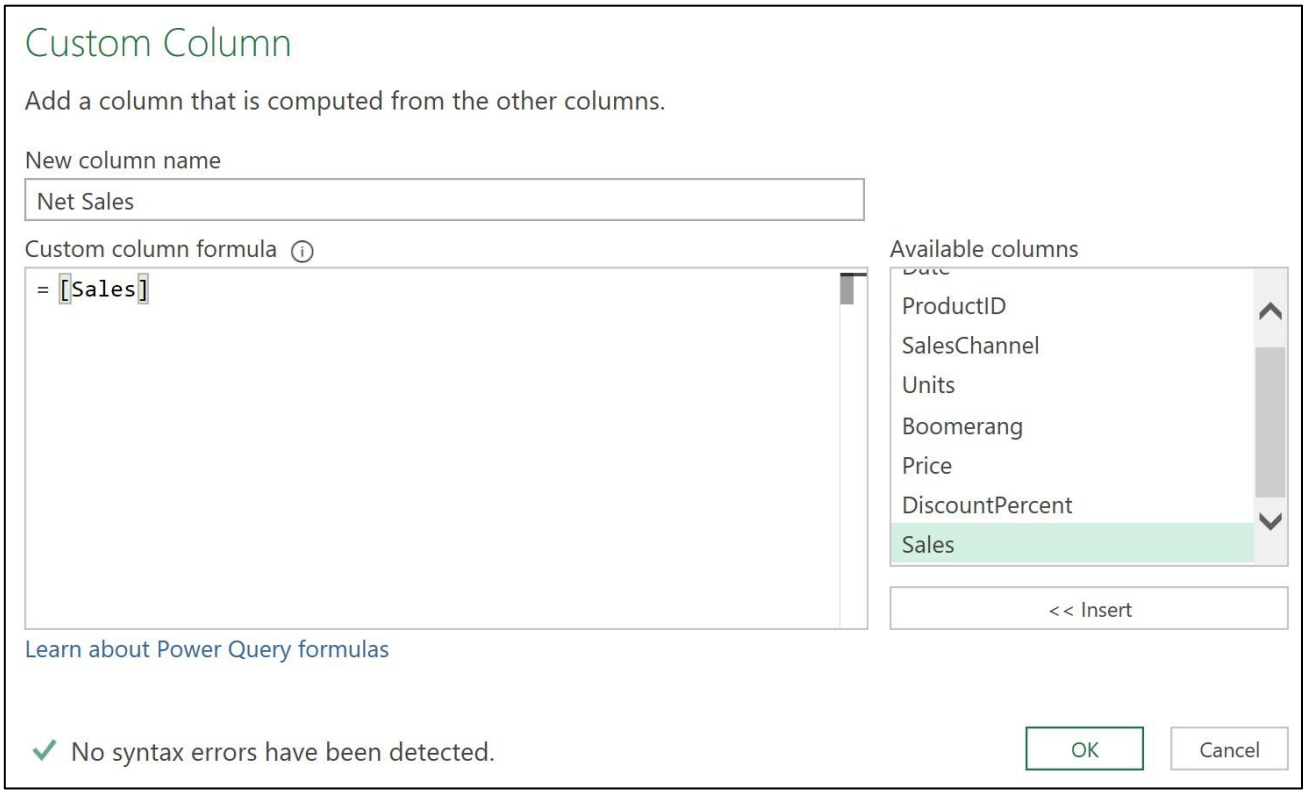
1. For our second column formula, we need to apply the location discount to the sales amount in each row. We have two options both of which calculate the correct net sales amount, as shown here:

   each [Sales] - [Sales] * [DiscountPercent]
   or
   each [Sales] * (1-[DiscountPercent])

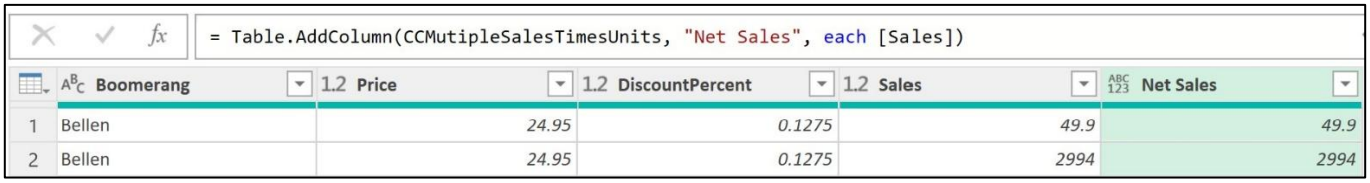
2. However, there is no user interface option in the From Number group on the Add Column tab to create either formula. When this happens, as shown below, you can create your own formula using the Custom Column button in the General group in the Add Column tab, as shown below:



3. The Custom Column dialog box is shown below. Type the column name NetSales in the New column name text box. Click after the equal sign in the Custom column formula area, and then, in the Available columns area, double-click the [Sales] field to insert it into the formula. Notice that we did not have to type the keyword *each*. It will be added automatically when we click OK. We are not done with the formula, but we want to click OK to check the progress of our formula; then we will come back to this dialog box to finish it in just a moment. Click OK.



4. As shown below, the Table.AddColumn function correctly uses the identifier from the previous query step as the table value in the first argument. The new column name that you typed into the dialog box appears correctly as a text value in the second argument. In the third argument, the keyword *each* is correctly placed before the formula. There is no data type value in the fourth argument, unlike when we created the column formula from the From Number group. When you use the Custom Column button, a data type value is not added to the formula. We will add it manually after we finish the formula. In the Applied Steps list rename the step "CCCalculateNetSales".

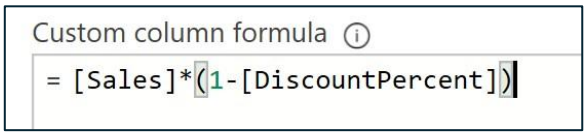


	Boomerang	Price	DiscountPercent	Sales	Net Sales
1	Bellen	24.95	0.1275	49.9	49.9
2	Bellen	24.95	0.1275	2994	2994

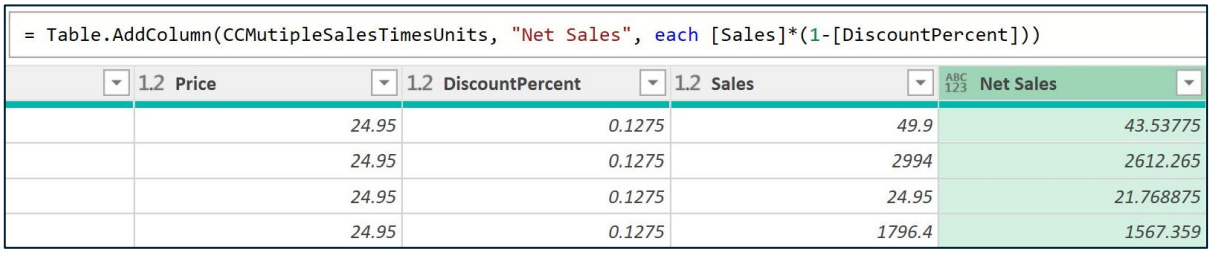
5. As shown below, to open the Custom Column dialog box and edit the formula, click the gear icon to the right of the CCCalculateNetSales step.



6. As shown below, create the result of the formula in the dialog box and click OK.



7. As shown below, the Net Sales amounts need to be rounded to the penny.



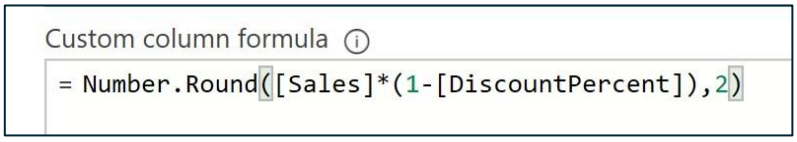
	Price	DiscountPercent	Sales	Net Sales
	24.95	0.1275	49.9	43.53775
	24.95	0.1275	2994	2612.265
	24.95	0.1275	24.95	21.768875
	24.95	0.1275	1796.4	1567.359

Rounding in Power Query

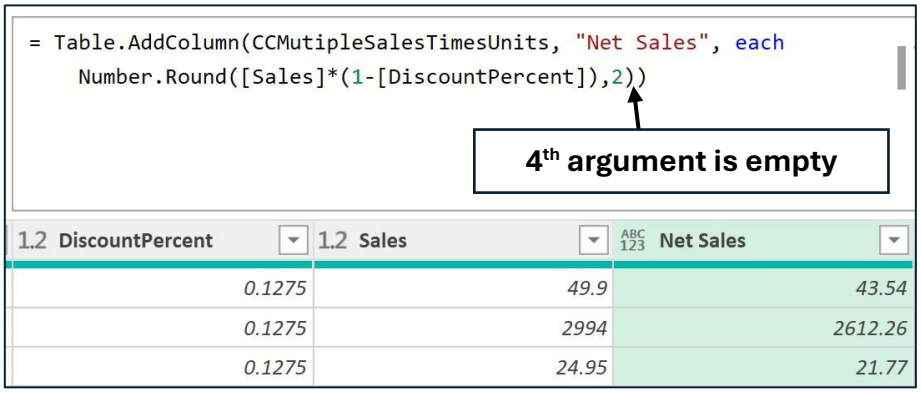
In Power Query, you can use several different rounding functions. If you want to match the rounding method commonly taught in school—where values ending in 5 or higher round up, and values below 5 round down—you can use the M function **Number.RoundAwayFromZero**. This is the same approach used by Excel's worksheet and DAX's ROUND function.

However, this method can introduce a slight upward bias because values exactly halfway between two numbers are always rounded away from zero. Power Query also provides function **Number.Round**, which uses banker's rounding, also known as Gaussian rounding or half-even rounding. Banker's rounding works like standard rounding except when the value is exactly halfway between two numbers. In that case, it rounds to the nearest even number. As a result, values ending in 5 sometimes round up and sometimes round down, which helps reduce rounding bias. Banker's rounding is commonly used in fields such as banking, manufacturing, and software development.

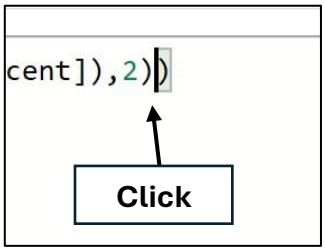
8. To open the Custom Column dialog box and edit the formula, click the gear icon to the right of the CCCalculateNetSales step in the Applied Steps list. Then, as shown in the picture below, edit the formula by using the Number.Round function and entering 2 as the second argument so the function rounds to the penny. Click OK.



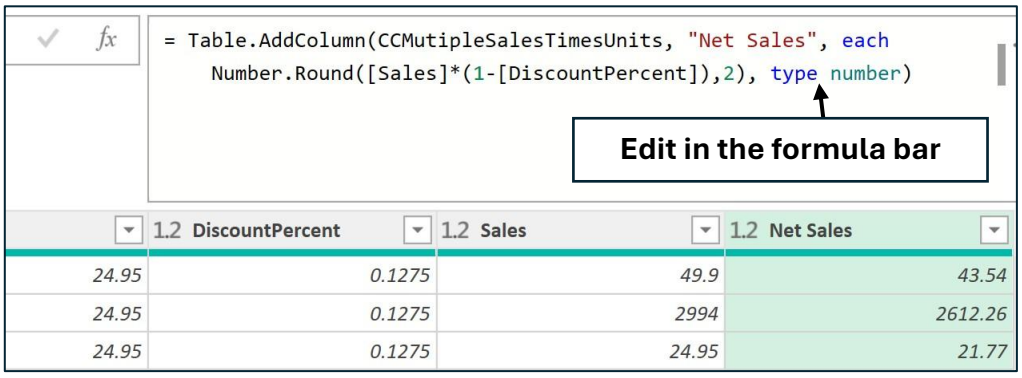
9. As shown below, the edited formula is correctly rounding the net sales amounts to the penny. However, the fourth argument in the Table.AddColumn function does not define a data type for the new column.



10. To edit the fourth argument of Table.AddColumn, click between the formula's second-to-last closing parenthesis and the final closing parenthesis, as shown below.



11. As shown below, to add a data type for the column, type a comma to move to the fourth argument of Table.AddColumn, and then type the M Code type value for the Decimal Number data type: type number.



12. Now that we have created the necessary columns for our table, we can remove columns that we do not need for analysis.

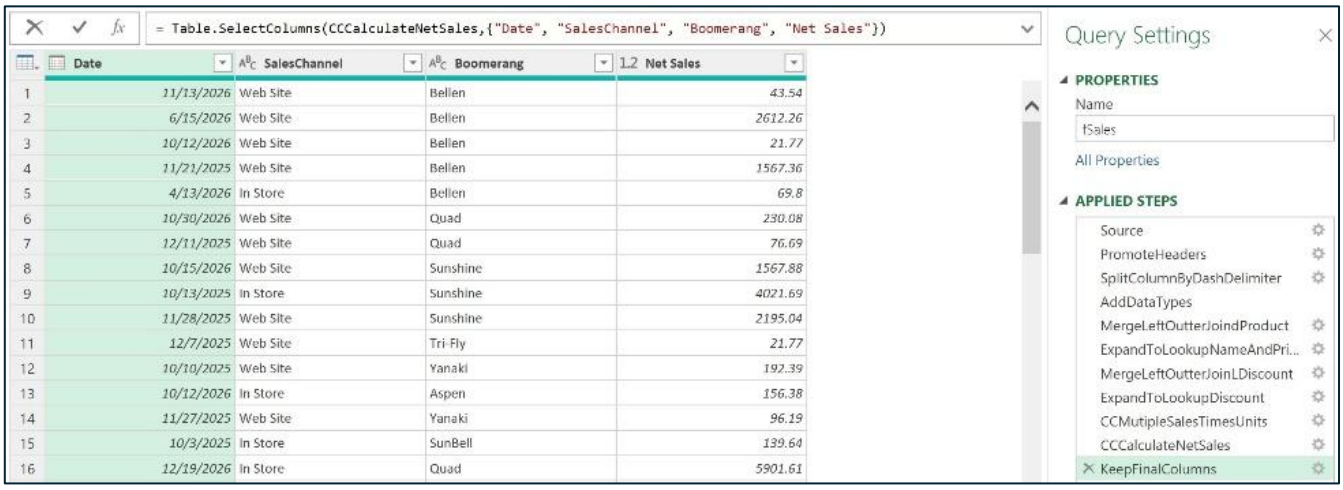
12) Transform: Remove Unwanted Columns

When cleaning and transforming data, always consider what data is needed in the final table or data model. For example, the table we just completed has nine columns, but our analysis requires only four. In this case, it is better to remove the unwanted columns for two reasons: first, fewer columns make the reporting area less cluttered; second, loading only the necessary data reduces the file size of the Excel file, Power BI file, or Data Model.

1. The only columns we need for analysis are Date, SalesChannel, Boomerang, and Net Sales. We want to keep those columns and remove the rest. When removing columns, the order in which you select them determines the order in which they appear in the final table. Select the columns in this order: Date, SalesChannel, Boomerang, and Net Sales. To select non-contiguous columns (columns not next to each other), click the first column, then hold down Ctrl while selecting the remaining columns. Then, as shown below, right-click any selected column and select Remove Other Columns.



2. The final table is shown below.



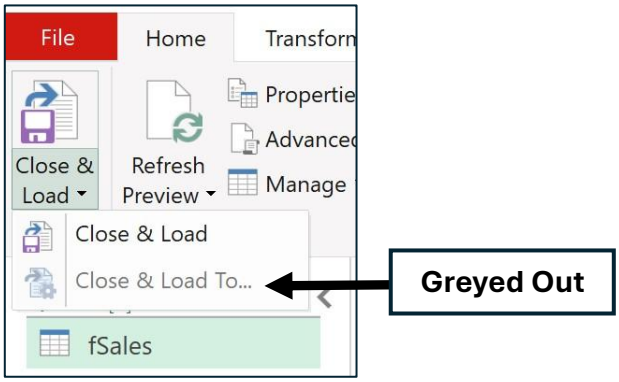
	Date	SalesChannel	Boomerang	Net Sales
1	11/13/2026	Web Site	Bellen	43.54
2	6/15/2026	Web Site	Bellen	2612.26
3	10/12/2026	Web Site	Bellen	21.77
4	11/21/2025	Web Site	Bellen	1567.36
5	4/13/2026	In Store	Bellen	69.8
6	10/30/2026	Web Site	Quad	230.08
7	12/11/2025	Web Site	Quad	76.69
8	10/15/2026	Web Site	Sunshine	1567.88
9	10/13/2025	In Store	Sunshine	4021.69
10	11/28/2025	Web Site	Sunshine	2195.04
11	12/7/2025	Web Site	Tri-Fly	21.77
12	10/10/2025	Web Site	Yanaki	192.39
13	10/12/2026	In Store	Aspen	156.38
14	11/27/2025	Web Site	Yanaki	96.19
15	10/3/2025	In Store	SunBell	139.64
16	12/19/2026	In Store	Quad	5901.61

3. When you open the Advanced Editor, as shown below, you can see the lovely let expression and M Code that you and Power Query created together:

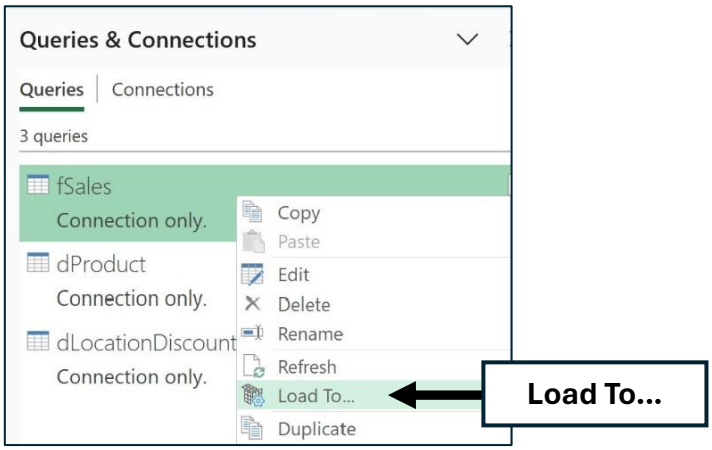
```
let
    Source = Csv.Document(File.Contents("D:\PowerQueryClass\PowerQuery02Files\SourceDataFiles\PowerQuery-02.csv"),[Delimiter=",", Columns=3, Encoding=65001, QuoteStyle=QuoteStyle.None]),
    PromoteHeaders = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
    SplitColumnByDashDelimiter = Table.SplitColumn(PromoteHeaders, "Description", Splitter.SplitTextByDelimiter("-", QuoteStyle.Csv), {"ProductID", "SalesChannel"}),
    AddDataTypes = Table.TransformColumnTypes(SplitColumnByDashDelimiter,{{"Date", type date}, {"ProductID", Int64.Type}, {"SalesChannel", type text}, {"Units", Int64.Type}}),
    MergeLeftOutterJoindProduct = Table.NestedJoin(AddDataTypes, {"ProductID"}, dProduct, {"ProductID"}, "dProduct", JoinKind.LeftOuter),
    ExpandToLookupNameAndPrice = Table.ExpandTableColumn(MergeLeftOutterJoindProduct, "dProduct", {"Boomerang", "Price"}, {"Boomerang", "Price"}),
    MergeLeftOutterJoinLDiscount = Table.NestedJoin(ExpandToLookupNameAndPrice, {"SalesChannel"}, dLocationDiscount, {"Sales Location"}, "dLocationDiscount", JoinKind.LeftOuter),
    ExpandToLookupDiscount = Table.ExpandTableColumn(MergeLeftOutterJoinLDiscount, "dLocationDiscount", {"DiscountPercent"}, {"DiscountPercent"}),
    CCMutipleSalesTimesUnits = Table.AddColumn(ExpandToLookupDiscount, "Sales", each [Units] * [Price], type number),
    CCCalculateNetSales = Table.AddColumn(CCMutipleSalesTimesUnits, "Net Sales", each Number.Round([Sales]*(1-[DiscountPercent]),2), type number),
    KeepFinalColumns = Table.SelectColumns(CCCalculateNetSales,{"Date", "SalesChannel", "Boomerang", "Net Sales"})
in
    KeepFinalColumns
```

13) Edit Load Location

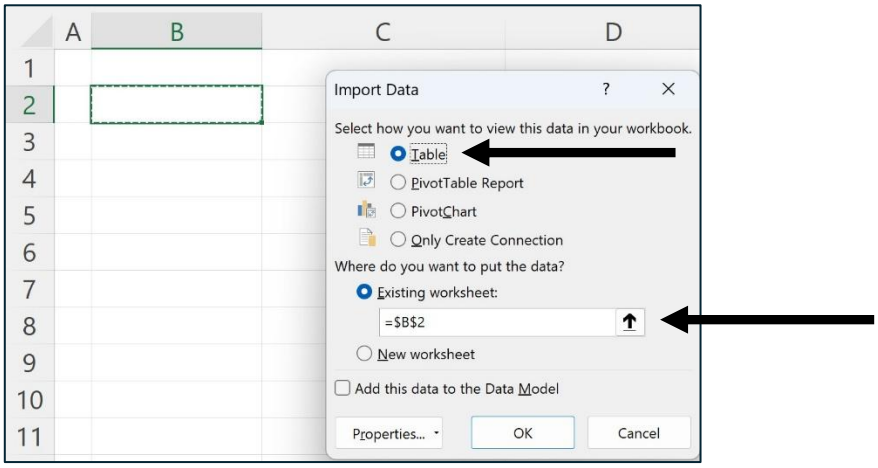
1. Now that we have finished editing the fSales query, we want to load the table to the worksheet. However, as shown below, if you click the Close & Load dropdown arrow in the Close group on the Home tab, the Close & Load To... option is greyed out. This is because after a query has been loaded, you cannot change its load location in the Power Query Editor; you must change it from the Excel worksheet. To do this, click Close & Load to load the query to the same location as before.



2. In the Excel workbook, go to the "Table PT Chart" worksheet and select cell B2. If the Queries & Connections pane is not open, select the Data tab on the Excel Ribbon. In the Queries & Connections group, click Queries & Connections. In the pane, right-click the fSales query, and then click Load To....



3. As shown below, in the Import Data dialog box, select the Table option, then select Existing worksheet. In the Existing worksheet text box, specify the cell where you want to load the table.



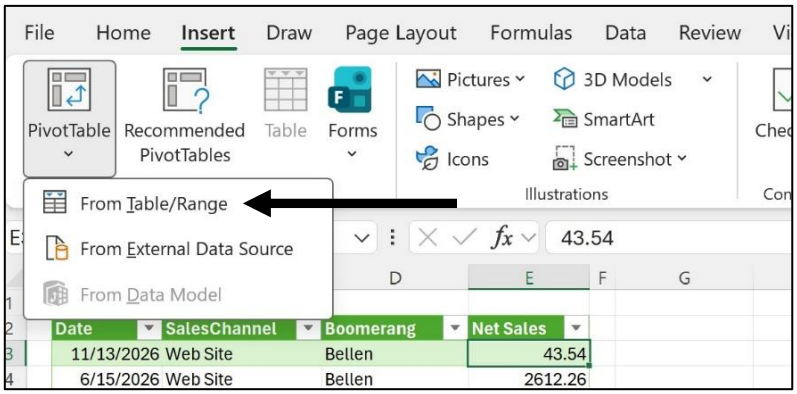
4. As shown below, when you load Power Query output to an Excel worksheet, it is loaded an Excel Table. Now that we have the table we want, we can create a PivotTable and Chart.



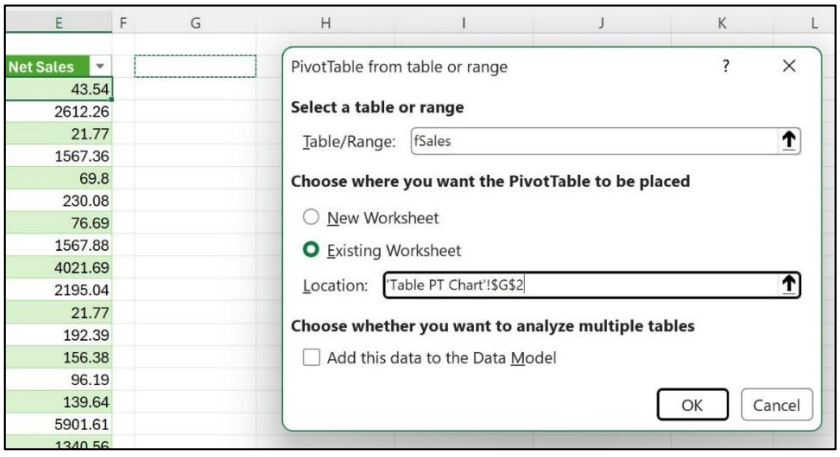
Date	SalesChannel	Boomerang	Net Sales
11/13/2026	Web Site	Bellen	43.54
6/15/2026	Web Site	Bellen	2612.26
10/12/2026	Web Site	Bellen	21.77
11/21/2025	Web Site	Bellen	1567.36
4/13/2026	In Store	Bellen	69.8
10/30/2026	Web Site	Quad	230.08
12/11/2025	Web Site	Quad	76.69
10/15/2026	Web Site	Sunshine	1567.88

14) Load Query Output to an Excel Table, Load a Second Time To The PivotTable Cache, Make Chart

1. The reporting and visualizing goal is to show the total net sales for boomerang products for each sales channel. To create a PivotTable, as shown below, select a cell inside the query output, select the Insert tab on the Excel Ribbon. In the Tables group, click the PivotTable dropdown arrow, then click From Table/Range. To create a PivotTable from a table with a keyboard shortcut use: Alt, N, V, T

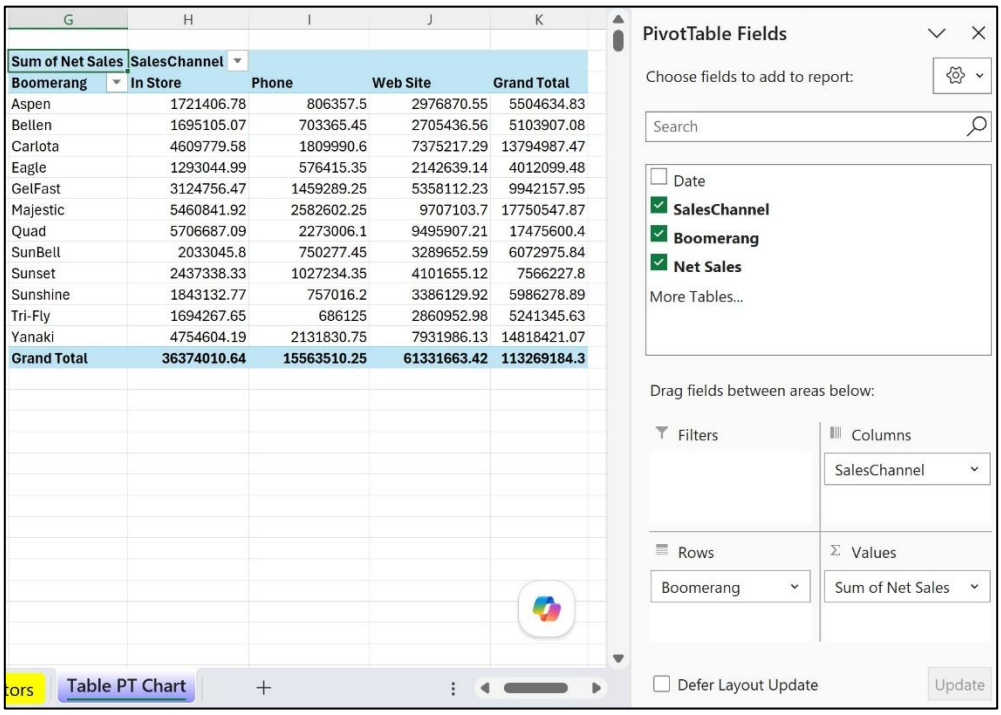


2. As shown below, in the PivotTable from table or range dialog box, select Existing Worksheet and point to cell G2. Click OK.

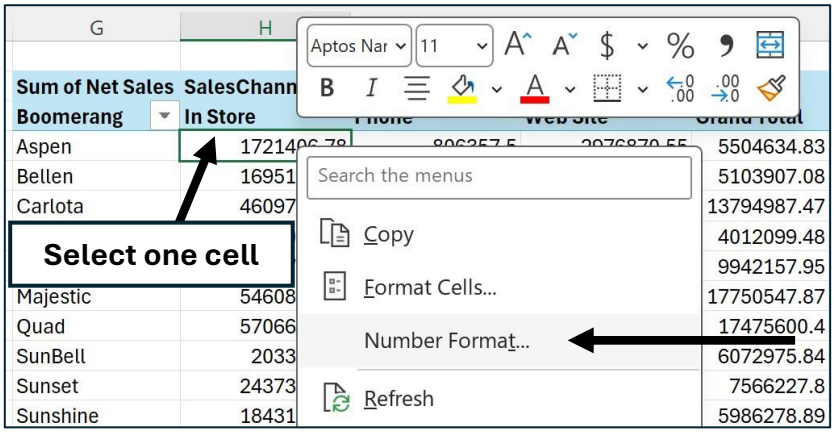


3. When making a PivotTable from an Excel Table that contains Query Output, you load the data to a second data destination called the PivotTable Cache. The PivotTable Cache is the behind-the-scenes location that PivotTables and PivotCharts use as their source data. If the original source data changes and you want the PivotTable and PivotCharts to update and reflect the changes in data, You must refresh both the Excel Table Query Output (right-click the Excel Table, then from menu select Refresh) and the PivotTable and PivotChart (right-click the PivotTable, then from the menu select Refresh).

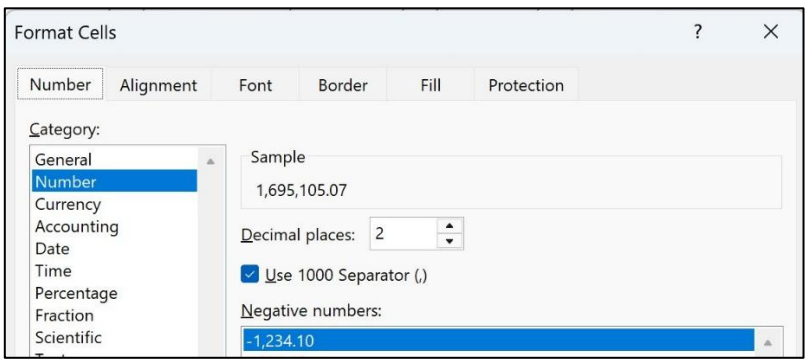
4. As shown below, to create the report, drag the SalesChannel field to the Columns area, drag the Boomerang field to the Rows area, and the Net Sales field to the Values area.

5. 

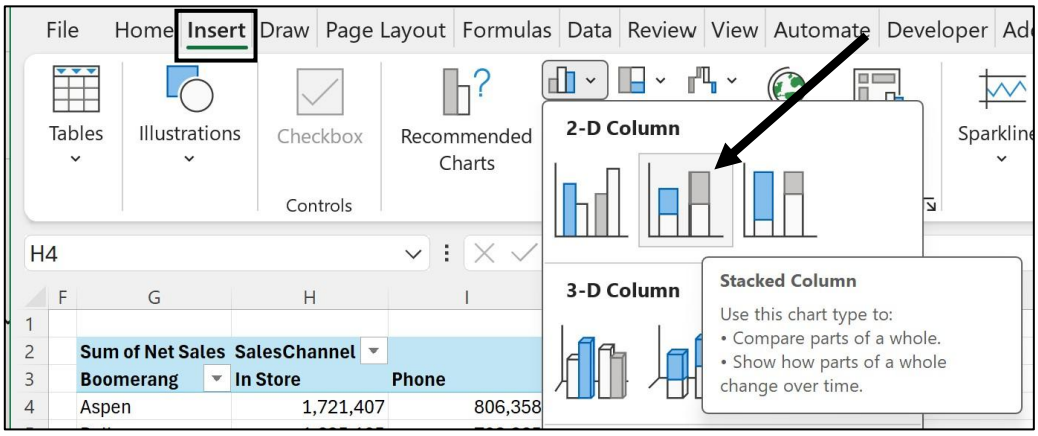
6. As shown below, to add Number Formatting to the Net Sales numbers, select one cell in the Values area of the PivotTable, right-click, select Number Format…. Do not select Format Cells. The Number Format… option adds number formatting to the entire field, rather than cells.

7. 

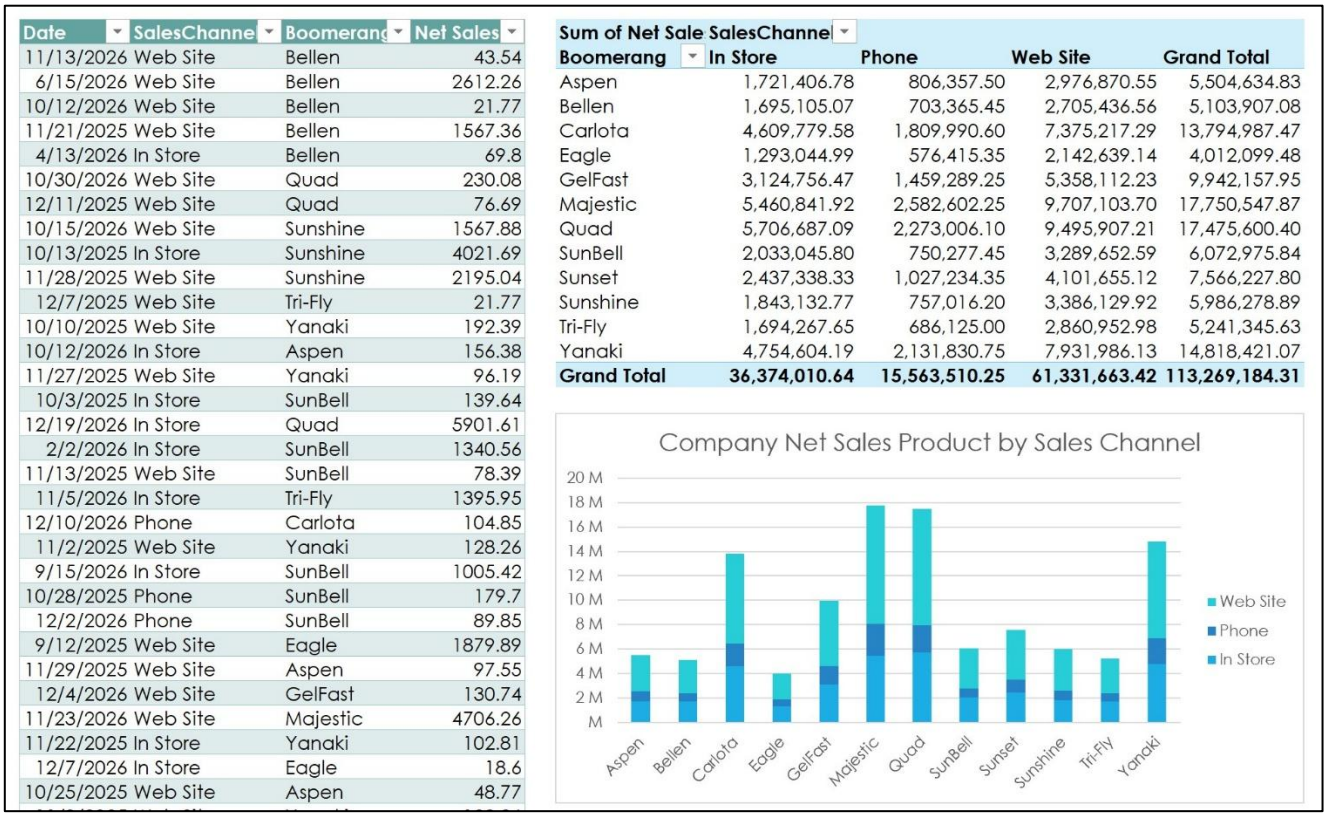
8. In the dialog box, add Number format, as shown below.



9. To visualize differences across categories from a cross tabulated report with one variable in the row area and one variable in the column area, you can use a Clustered Column chart or a Stacked Column chart. A Clustered Column chart emphasizes the difference across the variable in the legend within each horizontal-axis variable, while the Stacked Column chart emphasizes the differences across the variable in the horizontal axis. I am choosing the Stacked Column chart because I want to emphasize the heights of the variable in the horizontal axis while retaining the detail provided by the legend items within each column. To create this chart, select one cell in the report and then, as shown below, select the Insert tab. From the Chart group, select the dropdown arrow for the column chart, then select Stacked Column.



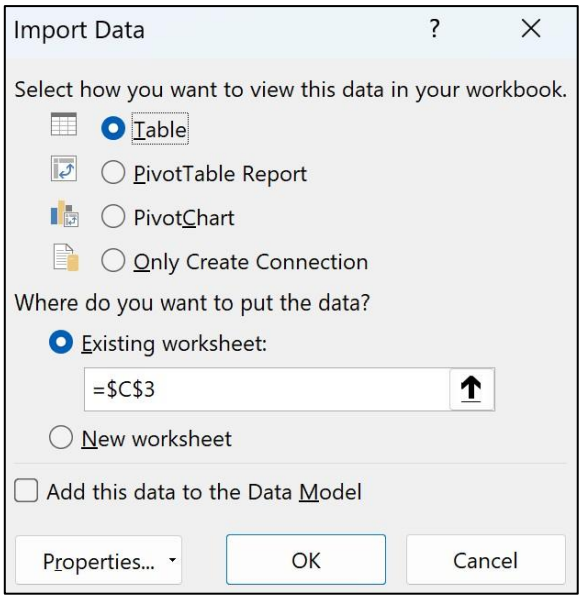
10. After formatting and adjusting the table, report and visualization, you might have a dashboard as shown below.

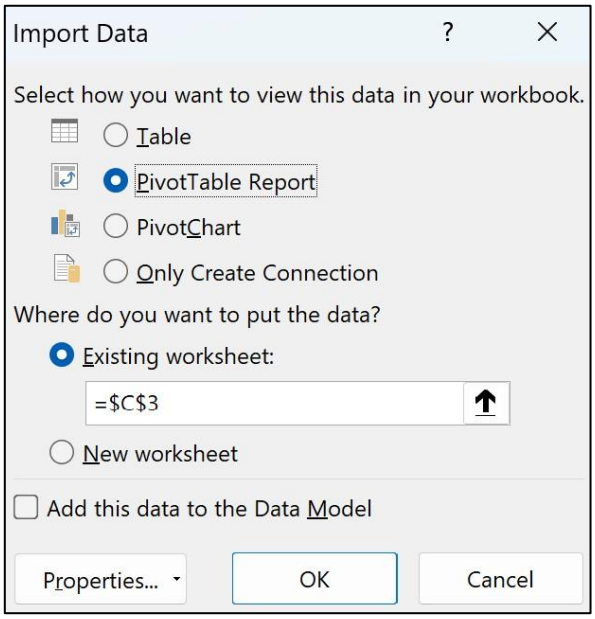


Date	SalesChanne	Boomerang	Net Sales
11/13/2026	Web Site	Bellen	43.54
6/15/2026	Web Site	Bellen	2612.26
10/12/2026	Web Site	Bellen	21.77
11/21/2025	Web Site	Bellen	1567.36
4/13/2026	In Store	Bellen	69.8
10/30/2026	Web Site	Quad	230.08
12/11/2025	Web Site	Quad	76.69
10/15/2026	Web Site	Sunshine	1567.88
10/13/2025	In Store	Sunshine	4021.69
11/28/2025	Web Site	Sunshine	2195.04
12/7/2025	Web Site	Tri-Fly	21.77
10/10/2025	Web Site	Yanaki	192.39
10/12/2026	In Store	Aspen	156.38
11/27/2025	Web Site	Yanaki	96.19
10/3/2025	In Store	SunBell	139.64
12/19/2026	In Store	Quad	5901.61
2/2/2026	In Store	SunBell	1340.56
11/13/2025	Web Site	SunBell	78.39
11/5/2026	In Store	Tri-Fly	1395.95
12/10/2026	Phone	Carlota	104.85
11/2/2025	Web Site	Yanaki	128.26
9/15/2026	In Store	SunBell	1005.42
10/28/2025	Phone	SunBell	179.7
12/2/2026	Phone	SunBell	89.85
9/12/2025	Web Site	Eagle	1879.89
11/29/2025	Web Site	Aspen	97.55
12/4/2026	Web Site	GelFast	130.74
11/23/2026	Web Site	Majestic	4706.26
11/22/2025	In Store	Yanaki	102.81
12/7/2026	In Store	Eagle	18.6
10/25/2025	Web Site	Aspen	48.77

Sum of Net Sale	SalesChanne			
Boomerang	In Store	Phone	Web Site	Grand Total
Aspen	1,721,406.78	806,357.50	2,976,870.55	5,504,634.83
Bellen	1,695,105.07	703,365.45	2,705,436.56	5,103,907.08
Carlota	4,609,779.58	1,809,990.60	7,375,217.29	13,794,987.47
Eagle	1,293,044.99	576,415.35	2,142,639.14	4,012,099.48
GelFast	3,124,756.47	1,459,289.25	5,358,112.23	9,942,157.95
Majestic	5,460,841.92	2,582,602.25	9,707,103.70	17,750,547.87
Quad	5,706,687.09	2,273,006.10	9,495,907.21	17,475,600.40
SunBell	2,033,045.80	750,277.45	3,289,652.59	6,072,975.84
Sunset	2,437,338.33	1,027,234.35	4,101,655.12	7,566,227.80
Sunshine	1,843,132.77	757,016.20	3,386,129.92	5,986,278.89
Tri-Fly	1,694,267.65	686,125.00	2,860,952.98	5,241,345.63
Yanaki	4,754,604.19	2,131,830.75	7,931,986.13	14,818,421.07
Grand Total	**36,374,010.64**	**15,563,510.25**	**61,331,663.42**	**113,269,184.31**

15) Refreshing when source data changes

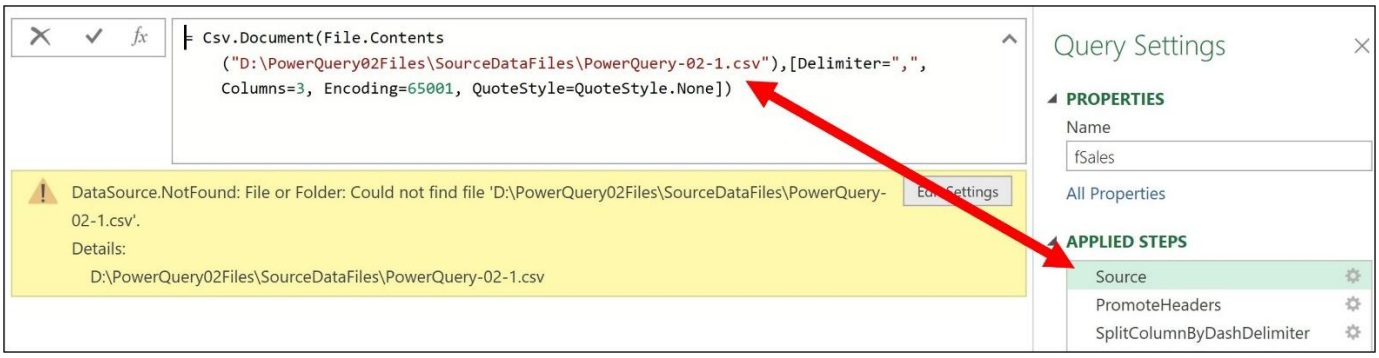
1. **Load to Excel Table in Worksheet and make PivotTable from Excel Table Query Output.**
   - When you load query output to the Excel worksheet as an Excel Table and make a PivotTable from the query output, the data from the query is loaded to the PivotTable Cache (behind the scenes data source for the PivotTable). This means that the data is loaded to two locations. In order to get the PivotTable report to refresh, you must: 1) right-click the query output and from the menu select Refresh, and then do the same thing in the PivotTable. This means you will have to refresh twice.
   - Import Data dialog box looks like this:
   - 

2. **Load directly to a PivotTable Report (loading directly to PivotTable Cache).**
   - When you load the query output to a PivotTable Report (PivotTable Cache), when source data changes and you need to refresh the PivotTable report, you right-click the PivotTable and from the menu select Refresh. This requires only one refresh.
   - Import Data dialog box looks like this:
   - 

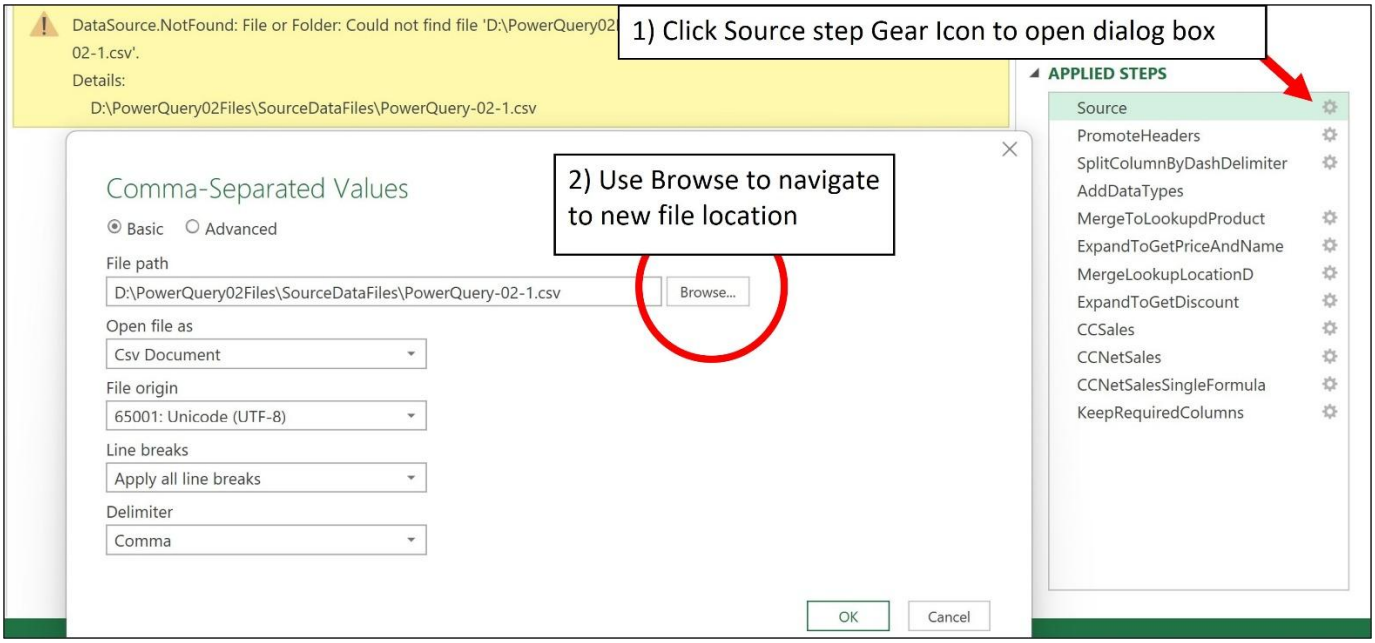
16) DataSource.NotFound Error

When you get the **DataSource.NotFound** error, it means that the On Premises file or folder path that Power Query is using to find the source data is no longer valid. Either the file name has changed, or the file was moved to a different location. There are four ways to correct this error:

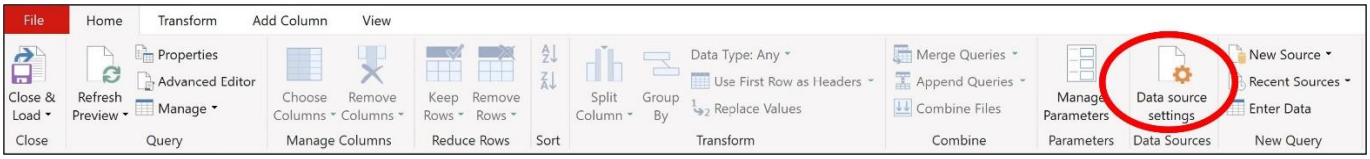
1. As shown in the picture below, you can select the Source step and in the formula bar, you can manually edit the file or folder path.



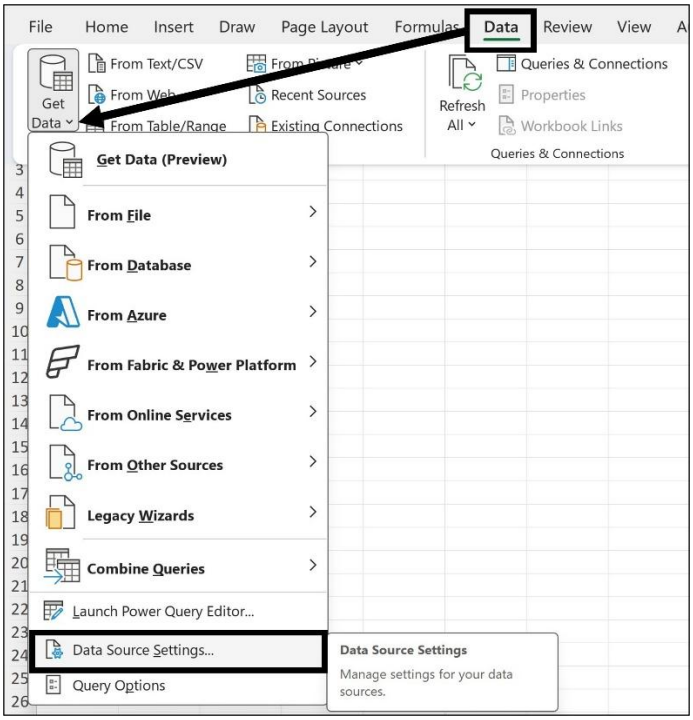
2. As shown in the picture below, you can click the Gear icon on the right side of the Source step to open the dialog box and then use the Browse button to navigate to the new location.



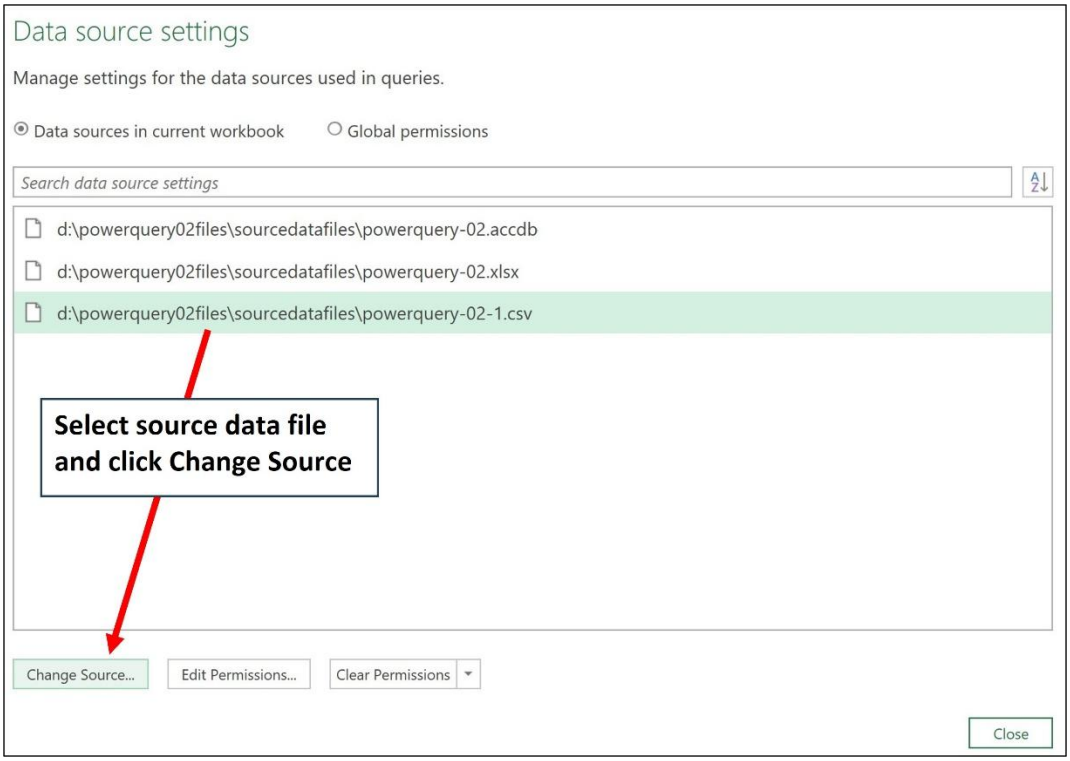
3. As shown in the picture below, on the Power Query ribbon, go to the Home tab and, in the Data Sources group, click Data source settings. If the data source is used in multiple queries, this method lets you change the source once & automatically update every query that uses that file or folder path. The fourth item in this list shows the steps that are required in the Data Source Settings dialog box.



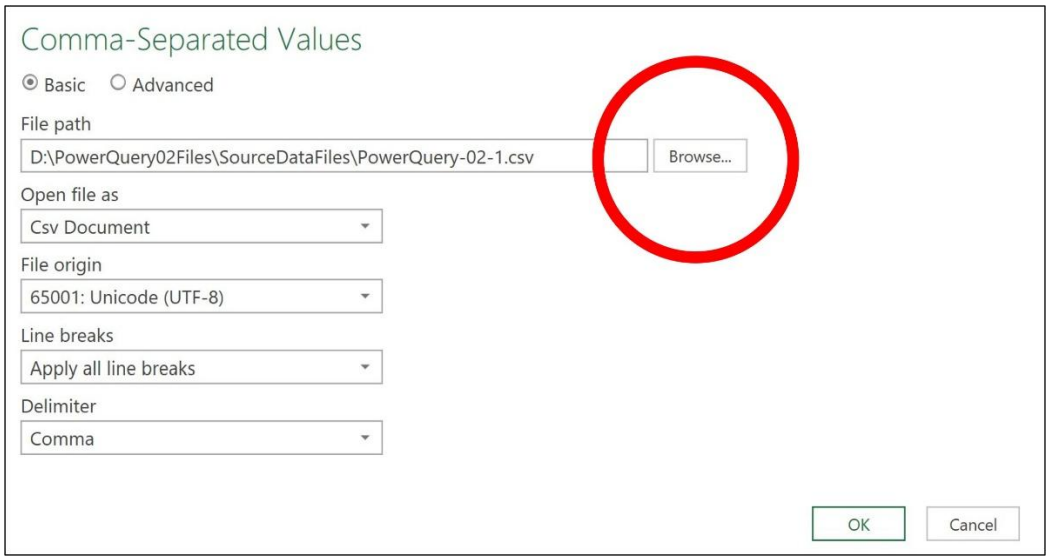
4. As shown in the picture below, If you are using the Excel Data Ribbon tab (and not the Power Query Editor Ribbon tab as shown in the previous step), you can access the same Data Source Settings dialog box by going to the Data tab, Get & Transform Data group, Get Data dropdown arrow and from the menu select Data Source Settings. This method lets you change the source once & automatically update every query that uses that file or folder path.



As shown in the picture below, In the Data Source Settings dialog box, select the source data file and then click the Change Source... button.



As shown in the picture below, you can then navigate to the correct data source and select that location to update the Power Query On Premises path.



Summary

In this video we had a thorough introduction to Power Query, M Code, and importing, cleaning, transforming, and loading data. See you next video for this same example in the app Power BI Desktop.

Keyboard shortcuts seen in video

1. Open Windows Explorer: ⊞ **+ E**
2. From a worksheet in an Excel file, to open the Power Query Editor: **Alt + F12**
3. In an Excel file, to convert a proper table to Excel Tables using the keyboard: **Ctrl + T**
4. In the Power Query Editor to rename an identifier, such as a query step or query name use: **F2**
5. Create a PivotTable from a table: Alt, N, V, T