Power Query #12

Dynamic Folder Path in Excel

Excel.Workbook Function

Import Multiple Excel Sheets or Tables From Multiple Excel Workbooks

Append Tables with Added Attribute Columns with Table.Combine

Table of Contents

Project Goals for Video 12 2

Dynamic Folder/File Path in Excel Workbook 3

Folder.Contents vs. Folder.Files functions 4

Excel.Workbook Function 5

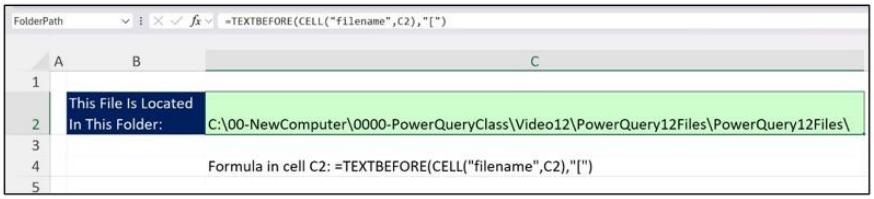
The arguments for the Excel.Workbook function 5

Excel.Workbook's Table of Objects and Attributes of Those Objects 6

Keyboards used in video 8

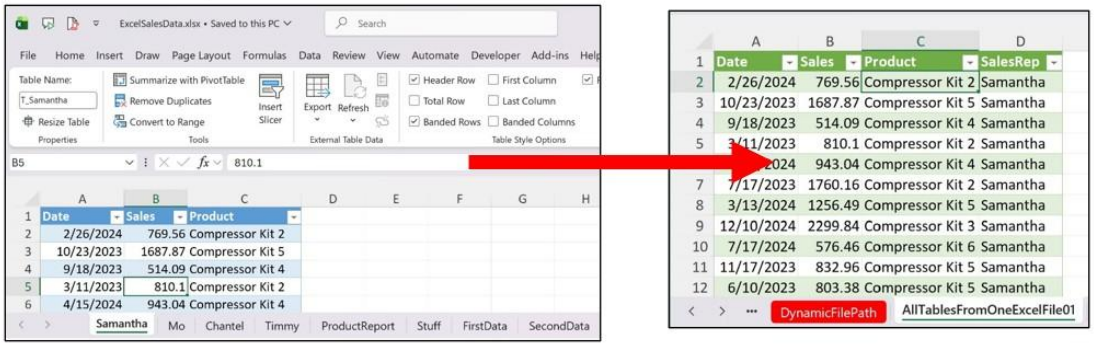
Project Goals for Video 12

1) Create a Dynamic Folder/File Path that avoids DataSource.NotFound error from On Premises Paths. Only works in Excel. Does not work in Power BI.

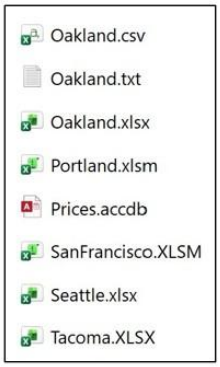


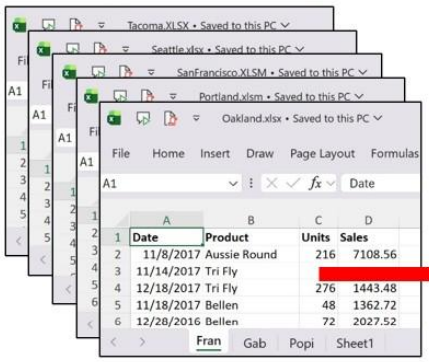
2) Using the Excel.Workbook function in a destination Excel file, from a single source data Excel file with many objects, we need to import multiple Excel Tables that start with "T_" and append them into a single table with the worksheet names as new columns.

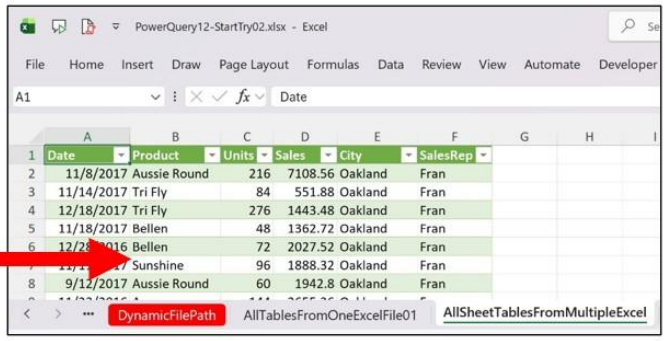
ExcelSalesData.xlsx



3) Using the From Folder feature & Excel.Workbook function in a destination Excel file, from multiple source data Excel files, we need to import worksheets that do not start with "Sheet" and append them into a single table with file and worksheet names as new columns.

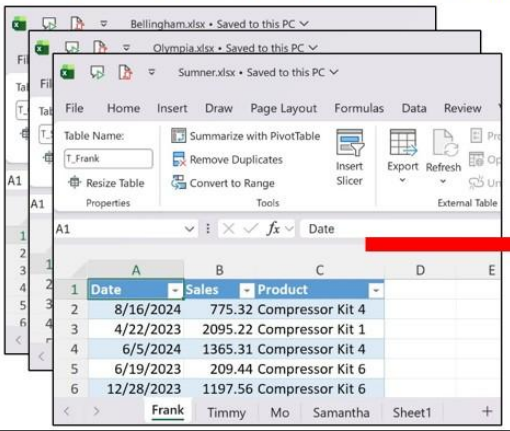


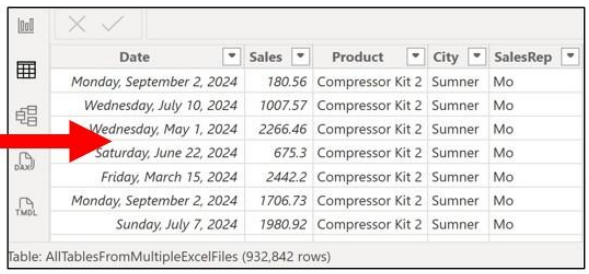




4) Using the From Folder feature & Excel.Workbook & Table.AddColumn functions in a destination Power BI Desktop file, from multiple source data Excel files, we need to import Excel Tables with a proper naming convention and append then into a single table with file and worksheet names as new columns.

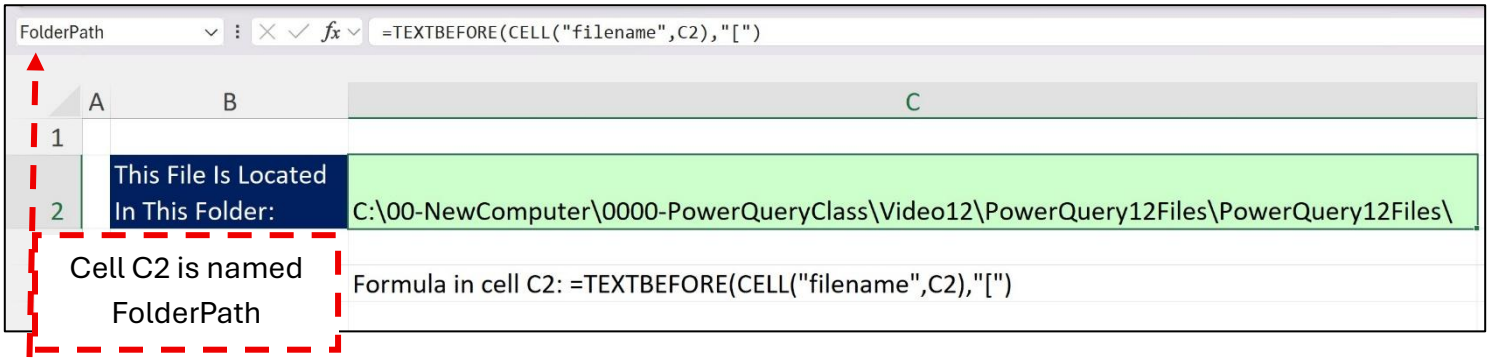




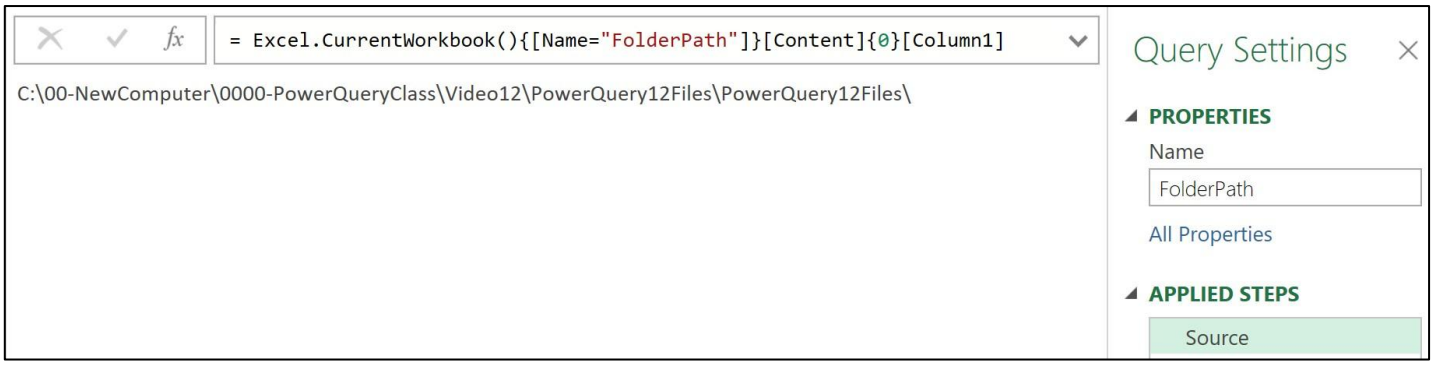


Dynamic Folder/File Path in Excel Workbook

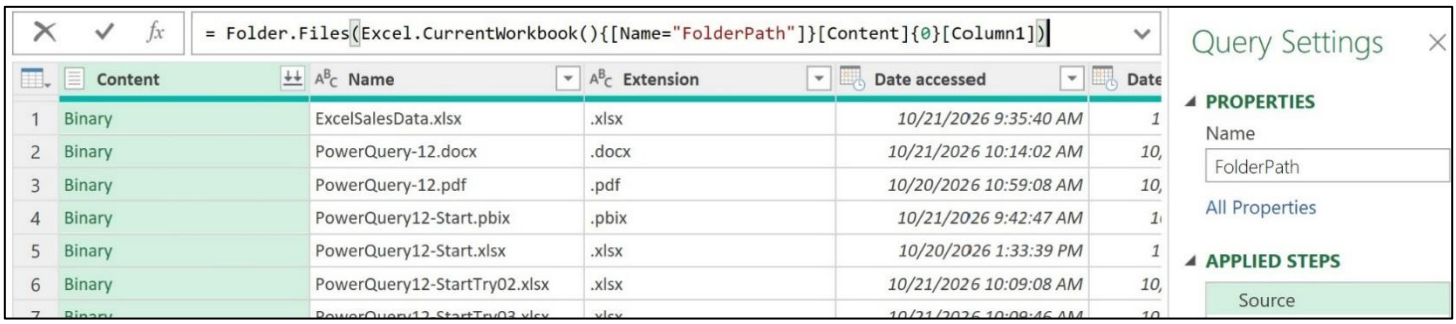
1. This does not work in Power BI Desktop.
2. This tip works in Excel because we have a worksheet and the CELL function.
3. The CELL function can return the full file and worksheet path for the current workbook. As shown below, you can use the TEXTBEFORE function to remove the workbook and worksheet reference from the end of the path, leaving only the folder path.



4. As long as you keep related Excel files and folders in a designated folder—and move, zip, or send that folder with all its contents intact—the dynamic folder/file path will change when the location changes and the path will remain correct. This avoids on-premises folder/file path issues and the dreaded DataSource.NotFound error.
5. Using the Defined Name feature, name cell C2 FolderPath.
6. Using the From Table/Range option in the Data tab, Get & Transform group, import the Defined Name.
   - By default, it creates a new Defined Name "FromArray_1" and assigns that name as the query name.
   - Change the query name to FolderPath.
   - Delete the automatic Changed Type and Promote Headers steps (if they appear).
   - Edit the formula to change the default name "FromArray_1" to FolderPath and add the two-way lookup formula element to the end of the formula: {0}[Column1], as shown below:



7. If you refer to the folder path in other queries you get a Formula.Firewall error (this error indicates that data leakage may occur). To avoid this, as shown in the picture on the next page, you use the Folder.Files function to lookup the contents of the folder location.
8. Load the query as a Only Create Connection query.
9. Open the Name Manager with the keyboard Ctrl + F3 and delete the Defined Name "FromArray_1".



	Content	Name	Extension	Date accessed	Date
1	Binary	ExcelSalesData.xlsx	.xlsx	10/21/2026 9:35:40 AM	1
2	Binary	PowerQuery-12.docx	.docx	10/21/2026 10:14:02 AM	10,
3	Binary	PowerQuery-12.pdf	.pdf	10/20/2026 10:59:08 AM	10,
4	Binary	PowerQuery12-Start.pbix	.pbix	10/21/2026 9:42:47 AM	1
5	Binary	PowerQuery12-Start.xlsx	.xlsx	10/20/2026 1:33:39 PM	1
6	Binary	PowerQuery12-StartTry02.xlsx	.xlsx	10/21/2026 10:09:08 AM	10,

Folder.Contents vs. Folder.Files functions

The **Folder.Contents** function imports all content directly inside the folder: both files and folders. This function does not retrieve any files from within subfolders. The **Folder.Files** function imports all files directly inside the folder and from within all subfolders.

Excel.Workbook Function

When you want to import data from an Excel file, you can use the **Excel.Workbook** function, which extracts all the objects from a specified external Excel workbook file and delivers a table of objects and attributes of those objects to the Power Query Editor. Excel.Workbook imports the following objects:

- Excel Tables
- User-created Defined Names
- Worksheets with all sheet content (including empty cells between objects)
- Automatically created defined names when you use these features:
  - Print range
  - Advanced filter criteria and/or extract ranges
  - The auto filter feature (this hidden name is not shown in the Name Manager dialog box)
- Defined Names that are NOT imported:
  - LAMBDA, formula names, and Dynamic Spilled Arrays

The arguments for the Excel.Workbook function

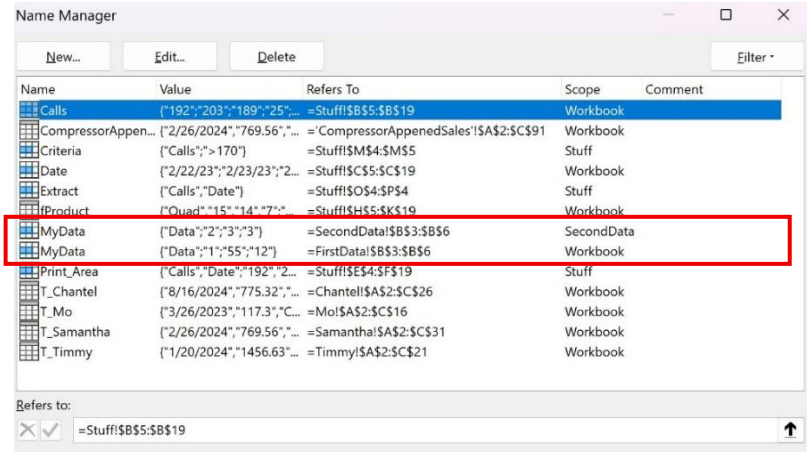
Excel.Workbook(**workbook** as binary, **[useHeaders]** as any, **[delayTypes]** as nullable logical) as table

- **workbook**:
  - A binary value representing the contents of an Excel file.
- **[useHeaders]**:
  - true/false value indicating whether the first row of each worksheet should be treated as a header.
  - Default: false.
  - You can also pass an options record with the properties: UseHeaders, DelayTypes, or InferSheetDimensions (useful when the dimensions metadata is incorrect - only in Open XML Excel files, not legacy Excel files).
  - If headers are numbers or dates (which I have never seen), if there are different operating system cultures set, you might run into issues.
- **delayTypes**:
  - true/false value indicating whether the columns of each returned table should be left untyped.
  - Default: false.
  - There are two potential problems when you use the true option:
    - First, it does not always apply the correct data types. (I have seen it misinterpret an integer as a decimal number.)
    - Second, if you use this function to define data types, the query may run more slowly than if you define data types in a separate step. Because using false in the delayTypes argument is convenient, I often do so when the datasets are small and contain no integer data.

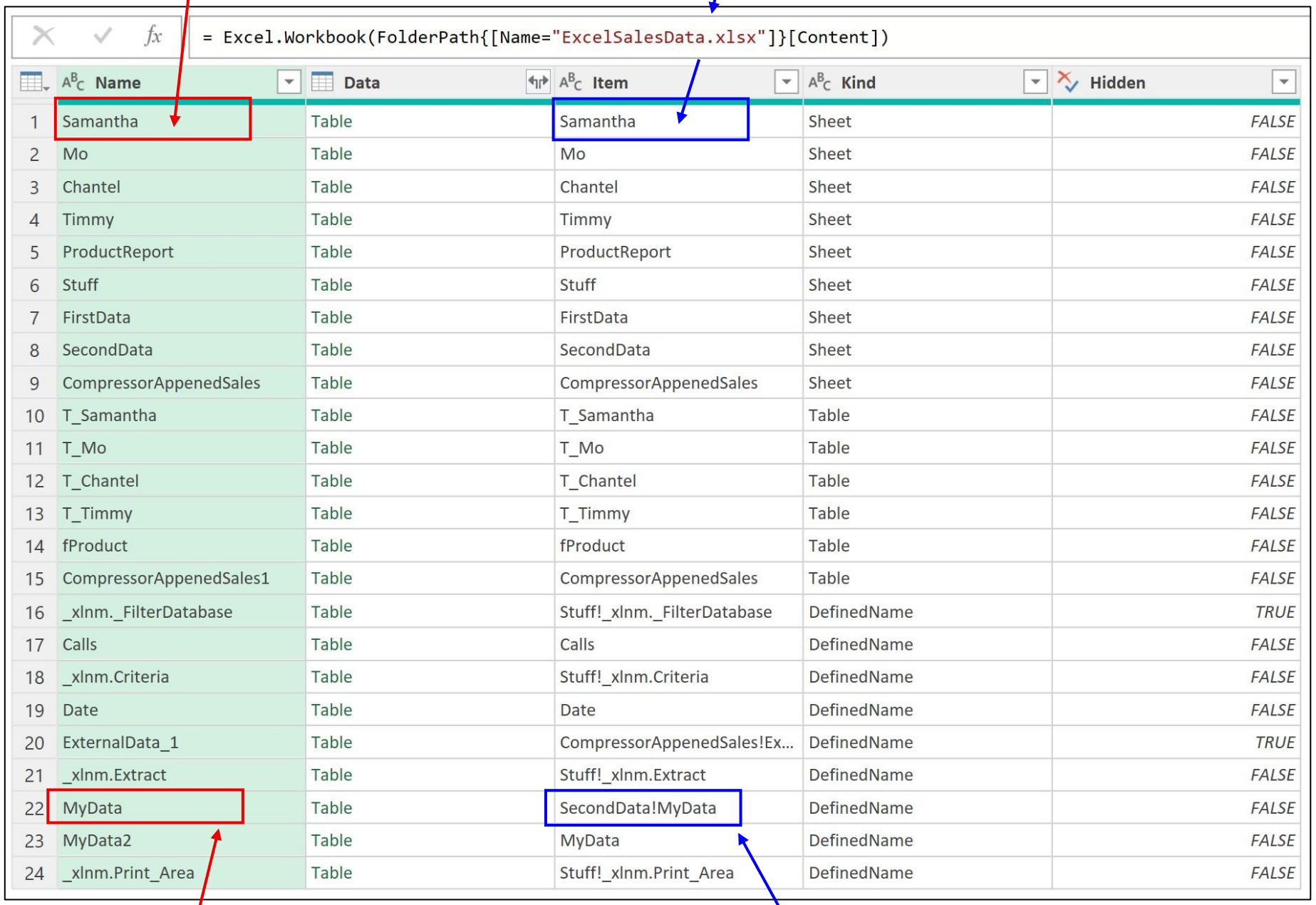
Excel.Workbook's Table of Objects and Attributes of Those Objects

As shown in the picture on the next page, from example #1 in the video, the Excel.Workbook function returns a table with five columns which returns the object's content and attributes from the Excel file named ExcelSalesData.xlsx. The definition of each of the five columns is listed here:

1. Name
    - The Name column contains the **user interface displayed Defined Name** of the object. The name in the Name column is often the same as the source Excel file internal identifier that is listed in the Item column. For example, in the first row, the name/identifier Samantha is the same in both the Name and Item columns. However, sometimes the names/identifier can be different. For example, in row 22, where the Defined Name in the Name column, MyData, was assigned by the user within the Scope of the worksheet, SecondData (as shown in the picture Name Manager below), but the source Excel file identifier in the Item column is defined as: SecondData!MyData. This difference ensures that the source Excel file can uniquely identify all objects in a file, even though in some cases there may be duplicate names in the user interface experience.
2. Data
    - The Data column contains the content for each name/identifier.
3. Item
    - The Item column contains the **source Excel file internal identifier** of the object. When performing Key Match Lookup, it is safer to use the Item column rather than the Name column.
4. Kind
    - This column contains the default identifier for the type of object that exists in the source Excel file. This column can be used to filter out objects that you do not want. For example, you can filter to just include Excel Table objects by specifying that you only want Table objects.
5. Hidden
    - This column states whether the object is hidden from the Name Manager dialog box. For example, in the picture on the next page, row 16 shows the object FilterDatabase, which is an object that is automatically created when you use the Auto Filter feature and which is hidden from view in the Name Manager dialog box.



User interface displayed Defined Name is the same as the **source Excel file internal identifier name**.



```
= Excel.Workbook(FolderPath{[Name="ExcelSalesData.xlsx"]}[Content])
```

	Name	Data	Item	Kind	Hidden
1	Samantha	Table	Samantha	Sheet	FALSE
2	Mo	Table	Mo	Sheet	FALSE
3	Chantel	Table	Chantel	Sheet	FALSE
4	Timmy	Table	Timmy	Sheet	FALSE
5	ProductReport	Table	ProductReport	Sheet	FALSE
6	Stuff	Table	Stuff	Sheet	FALSE
7	FirstData	Table	FirstData	Sheet	FALSE
8	SecondData	Table	SecondData	Sheet	FALSE
9	CompressorAppenedSales	Table	CompressorAppenedSales	Sheet	FALSE
10	T_Samantha	Table	T_Samantha	Table	FALSE
11	T_Mo	Table	T_Mo	Table	FALSE
12	T_Chantel	Table	T_Chantel	Table	FALSE
13	T_Timmy	Table	T_Timmy	Table	FALSE
14	fProduct	Table	fProduct	Table	FALSE
15	CompressorAppenedSales1	Table	CompressorAppenedSales	Table	FALSE
16	_xlnm._FilterDatabase	Table	Stuff!_xlnm._FilterDatabase	DefinedName	TRUE
17	Calls	Table	Calls	DefinedName	FALSE
18	_xlnm.Criteria	Table	Stuff!_xlnm.Criteria	DefinedName	FALSE
19	Date	Table	Date	DefinedName	FALSE
20	ExternalData_1	Table	CompressorAppenedSales!Ex...	DefinedName	TRUE
21	_xlnm.Extract	Table	Stuff!_xlnm.Extract	DefinedName	FALSE
22	MyData	Table	SecondData!MyData	DefinedName	FALSE
23	MyData2	Table	MyData	DefinedName	FALSE
24	_xlnm.Print_Area	Table	Stuff!_xlnm.Print_Area	DefinedName	FALSE

User interface displayed Defined Name is DIFFERENT than the **source Excel file internal identifier name**.

Keyboards used in video

- Alt + F12 = Opens Power Query Editor in Excel
- F2 = Rename Queries and Query Steps