



*Part of the Power BI  
Bootcamp Series*

# Microsoft Power BI Intermediate Bootcamp

-----  
**Data Connections and Transformations**

## Instructors:

Nicolás Lagreste Zucchini

María Florencia Hourcouripé



O'REILLY®

# Class 01





# Course Agenda

## **Class 1: Introduction and getting started with M**

- Course Framework and Introduction
- Data Sources and Connectivity
- Getting Started with M: A Step Beyond the UI
- Combining Data

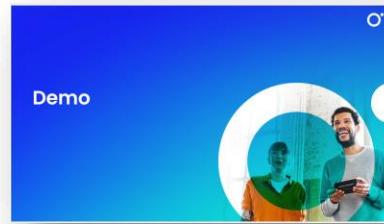
## **Class 2: Important tools in the M language**

- Understanding Object Structures and Data Types
- Intermediate Data Transformation
- Advanced Query Techniques and Error Handling
- Final Comments and Advanced Tips



Our Method on each topic:  
Demo example, followed by exercises to practice!

- **DEMO – Flor & Nico**
  
- **EXERCISE – Your turn!**



\*Students should have the latest version of **Power BI Desktop installed.**





# Instructors

**Follow us on LinkedIn:**



Nicolás Lagreste Zucchini



María Florencia Hourcouripé

**Let's Collaborate:** Interested in a personalized mentorship or consulting session? Contact us:

-  [nico@analyticmood.com](mailto:nico@analyticmood.com)
-  [flor@analyticmood.com](mailto:flor@analyticmood.com)

**Visit Our Website:**

- [www.analyticmood.com](http://www.analyticmood.com) for resources, upcoming events, and blog articles.

# Course Framework and Introduction



# Course Overview

## 1. Introduction to Power Query

- Understand the structure, workflow, and foundational concepts of Power Query.

## 2. Connecting to Data Sources

- Learn how to connect to and access a wide range of data sources.

## 3. M Language for Advanced Transformations

- Write custom code for complex data transformations beyond the user interface.

## 4. Combining Data

- Master techniques to merge and integrate data from multiple sources and tables.





# Before getting started

- We will work almost exclusively in the **Power Query** section of Power BI.
- We'll help you with concepts you've heard of but may not have worked with before.
- This is not an advanced course; it is specifically designed for **intermediate Power BI users** who are already familiar with using Power Query through the UI.
- Have **Power BI Desktop installed** (it is recommended to use the latest version available).
  - We will use January 2025 version in the Demos.
  - Microsoft web page: <https://www.microsoft.com/en-us/download/details.aspx?id=58494>





# Before getting started

- To be better prepared, review our introductory course: “**Power BI Fundamentals**” or attend our live event: “**Power BI Bootcamp**”



Power BI Fundamentals: Building a Strong Foundation in Data Analysis

By Maria Florencia Hourcouripé and Nicolás Lagreste Zucchini

[O'Reilly Media, Inc.](#) • February 2024

Write the [first review](#) 5h 14m Includes quizzes

<https://learning.oreilly.com/course/power-bi-fundamentals/0636920986317/>



# Starting Point for the Bootcamp

We will begin with the project from our previous Power BI Bootcamp.

To get started, download the following files:

Download Here: [www.analyticmood.com/resources](http://www.analyticmood.com/resources)

## 1. Power BI Project:

- Pbix: Exercise 1 to 10 and FinalPowerQuery

## 2. DataSources:

- Sales\_2022\_2023.xlsx
- Sales\_data\_2024.csv
- Budget 2024.pdf
- Budget 2022\_2023

Ensure all files are downloaded before proceeding.



\*Students should have the latest version of  
**Power BI Desktop installed.**



Let's do the following quiz to warm up



## Question 1

### What is Power Query?

- a) An advanced programming language
- b) An ETL (Extract, Transform, Load) tool in Excel and Power BI
- c) A type of chart in Power BI



## Question 2

**What language is used in Power Query to transform data?**

- a) DAX
- b) SQL
- c) M



## Question 3

**In which format are Power Query queries saved in Power BI?**

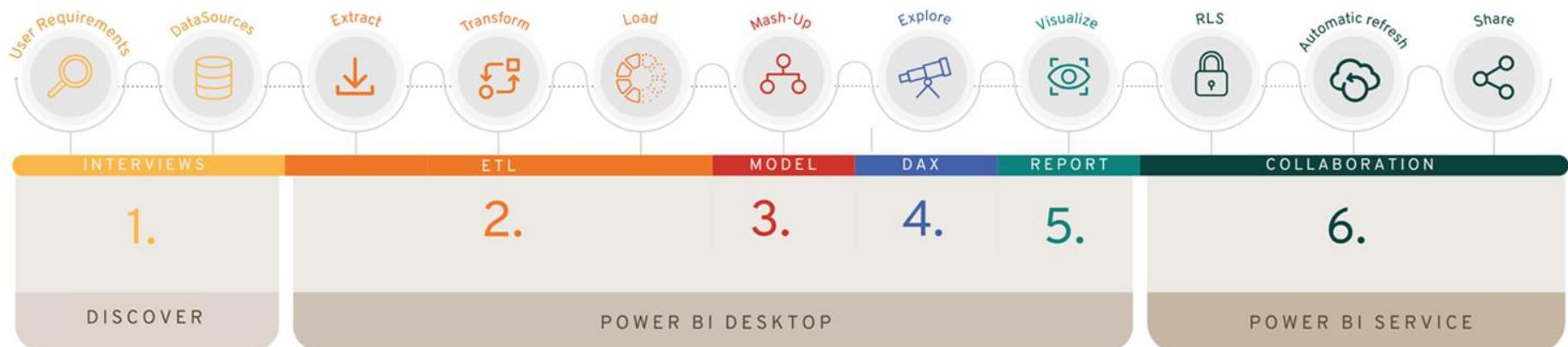
- a) In a SQL database file
- b) In a .pbix file
- c) In a text file



# Flow of Report Development

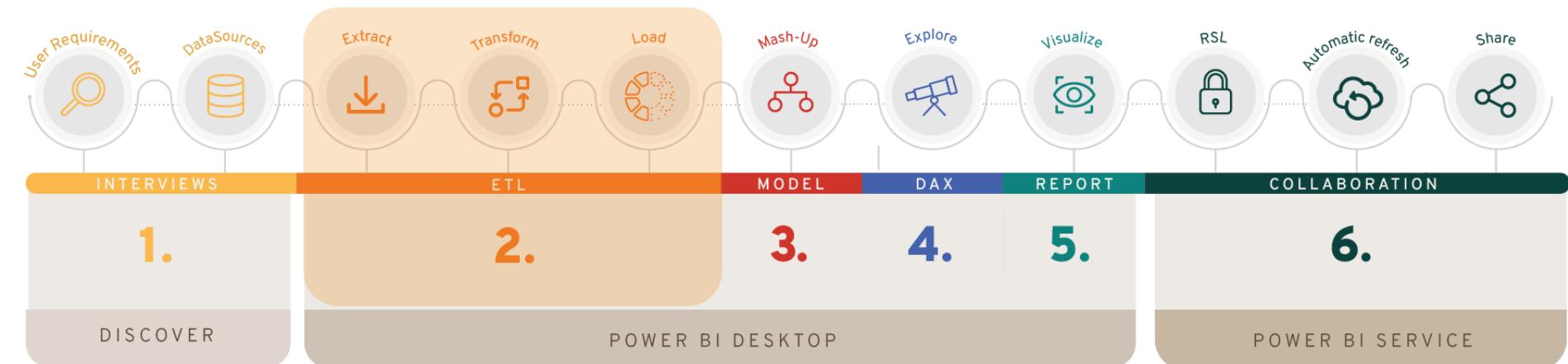


# Our Method: Flow of Report Development





# Our Method: Flow of Report Development





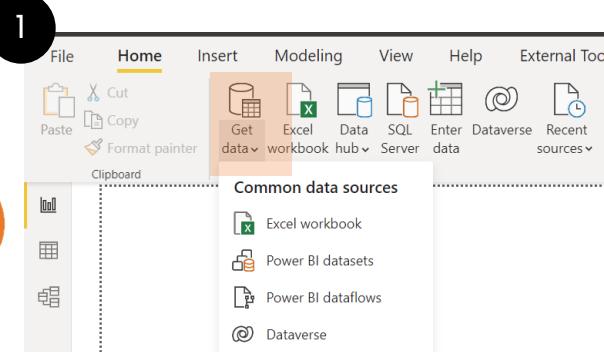
# What is Power Query?



# Power Query

Clean and transform data sources

Extract



2

SalesKey

	DateKey	channelKey	StoreKey	ProductKey
1	04/07/2010 0:00:00	1	247	
2	04/10/2010 0:00:00	1	52	
3	15/05/2010 0:00:00	1	40	
4	3388423	27/06/2010 0:00:00	1	67
5	3383821	20/10/2010 0:00:00	1	70
6	3382789	12/07/2010 0:00:00	1	238
7	3374643	26/10/2010 0:00:00	1	282
8	3371112	23/05/2010 0:00:00	1	21
9	3368889	29/04/2010 0:00:00	1	167
10	3365947	25/03/2010 0:00:00	1	48
11	3353916	29/04/2010 0:00:00	1	194
12	3349672	30/06/2010 0:00:00	1	8
13	3346418	02/05/2010 0:00:00	1	102
14	3329410	29/10/2010 0:00:00	1	146
15	3327925	04/06/2010 0:00:00	1	93

INTERVIEWS

ETL

MODEL

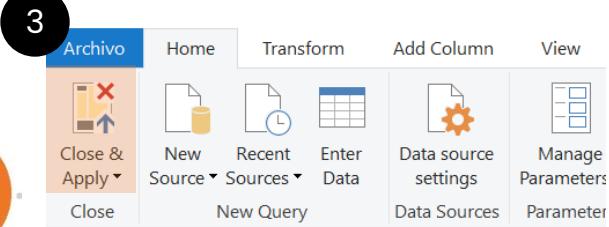
DAX

REPORT

COLLABORATION



Load

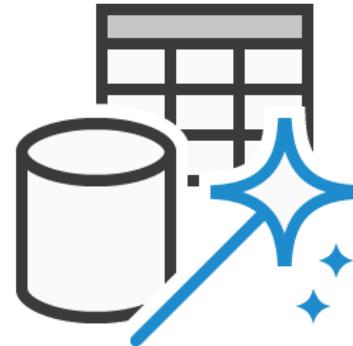




# What is Power Query?

A powerful tool for **data connection, cleaning, and transformation**.

- Built-in interface with advanced capabilities using M language.
- Works seamlessly with multiple data sources.
- Ideal for automating repetitive data tasks.

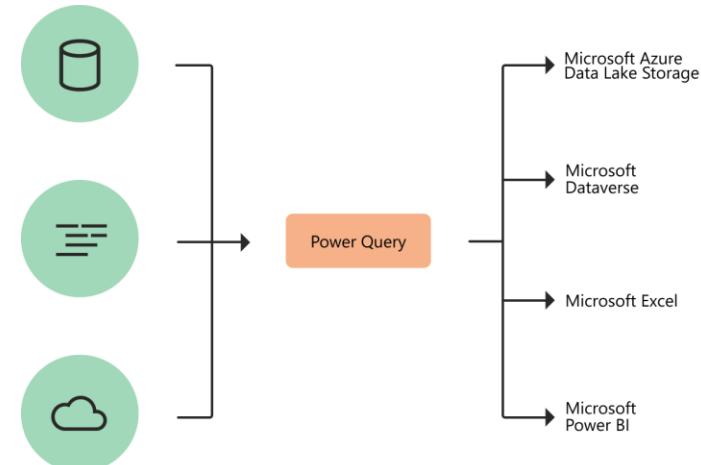




# How Power Query helps with data acquisition

Business users spend up to **80%** of their time preparing data, which slows down analysis and decision-making.

Power Query **streamlines this process**, making it easier to connect, transform, and update data efficiently for faster insights.



# DEMO 01

Change data sources





# Exercise 1

## Change data sources



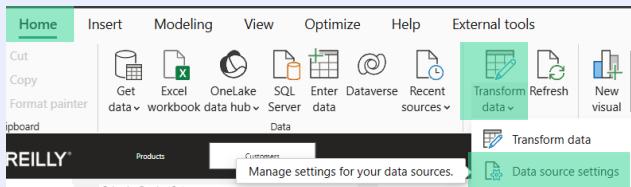
*Let's do this together*

# Exercise 1: Change the connections of the data sources

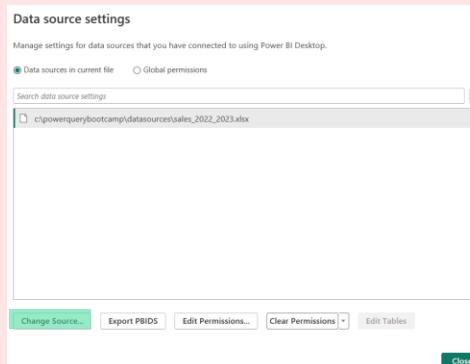
**Goal:** Learn to connect data sources on your personal computer.

Open the file **Exercise1.pbix** (Location: C:\PowerQueryBootcamp\Exercise1).

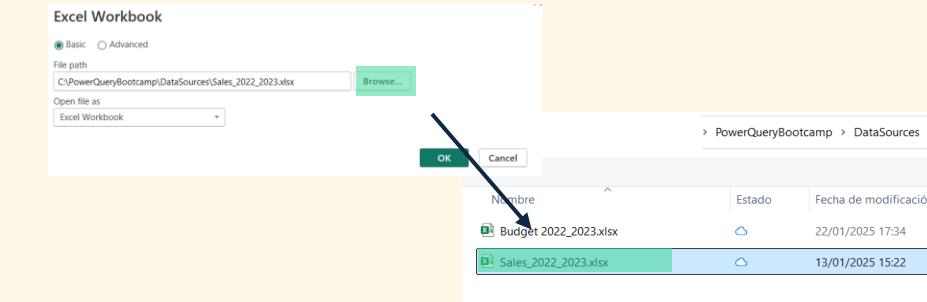
- 1 Click on **Home / Transform Data / Data Source Setting**



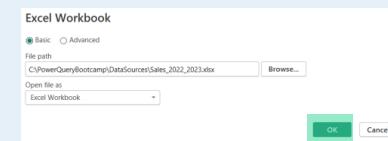
- 2 Click on **Change Source** at the bottom



- 3
- 1 Click on **Browse** at the bottom
  - 2 Go to **PowerQueryBootcamp/DataSources**
  - 3 Select **Sales\_2022\_2023**
  - 4 Click **OK**, then **Open**



- 4 In **Excel Workbook** window click on **OK** at the bottom

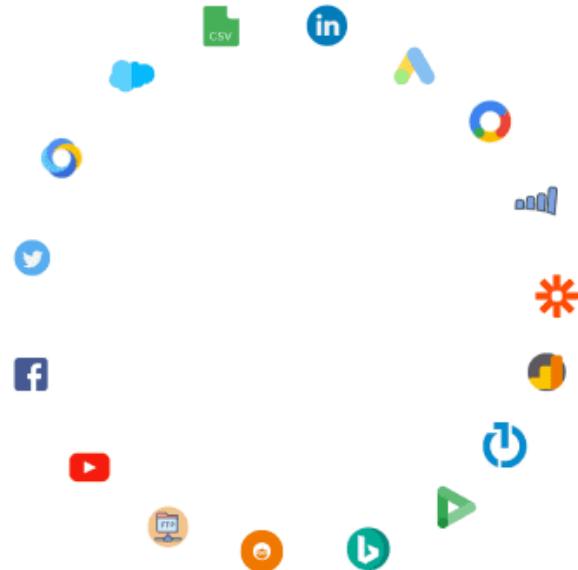




# Overview of Power BI Data Sources



# Power BI Data Sources





# Power BI Data Sources

## Types of Connectors

- Regular
- Custom
- Certified

## Types of Connections

- DirectQuery
- Import Mode
- Live Connection

## Benefits of Using Connections

- Up-to-date Information
- Flexibility in Data Handling
- Resource Efficiency
- Scalability



# Connectors

- Power Query offers a wide range of connectors for various data sources.
- For a complete list of available connectors and their details, check the link provided below.

<https://learn.microsoft.com/en-us/power-query/connectors/>

Connector	Excel	Power BI (Semantic models)	Power BI (Dataflows)	Fabric (Dataflow Gen2)	Power Apps (Dataflows)	Customer Insights (Dataflows)	Analysis Services
Azure Synapse Analytics (SQL DW) By Microsoft	✓	✓					
Azure Synapse Analytics workspace (Beta) By Microsoft	✗	✓					
Azure SQL database By Microsoft	✓	✓					
Access Database By Microsoft	✓	✓	✓	✓	✓	✓	✓
Active Directory By Microsoft	✓	✓	✓	✗	✗	✗	✓
Acterys (Beta) By Acterys	✗	✓	✓	✓	✗	✗	✗
	✗	✓	✓	✓	✗	✗	✗



# Connectors – M functions

- Power Query supports various file formats, utilizing M functions for efficient data retrieval and transformation. Common formats and their M functions are listed below.

File Format	M Functions
Azure Storage	AzureStorage.BlobContents, AzureStorage.Blobs, AzureStorage.DataLake, AzureStorage.DataLakeContents, AzureStorage.Tables
Binary	File.Contents
Excel	Excel.Workbook, Excel.CurrentWorkbook
Folder	Folder.Contents, Folder.Files
HDFS	Hdfs.Contents, Hdfs.Files
JSON	Json.Document
PDF	Pdf.Tables
Parquet	Parquet.Document
Text/CSV	Csv.Document
XML	Xml.Document, Xml.Tables

# DEMO 02

- Excel
- PDF
- SharePoint
- Azure Blob Storage
- SQL





# Accessing Files



# Accessing Files

- In Power BI Desktop, you can connect directly to data sources, but within Power Query, there's also the **option to retrieve data**.
- Below, you'll see the difference between the access buttons in both interfaces.

The screenshot shows the Microsoft Power BI Desktop interface. The top navigation bar includes File, Home, Insert, Modeling, View, Optimize, Help, and External tools. The Insert tab is selected, highlighting the 'Get data' button. A red box highlights this button. Below the ribbon, the 'Common data sources' section is expanded, listing various options like Excel workbook, Power BI semantic models, Dataflows, and Analysis Services. The main workspace displays a 'Add data to your repo' message and three buttons: 'Import data from Excel', 'Import data from SQL Server', and 'Paste data into a'. At the bottom, a link says 'Get data from another source'.

*Power BI Desktop*

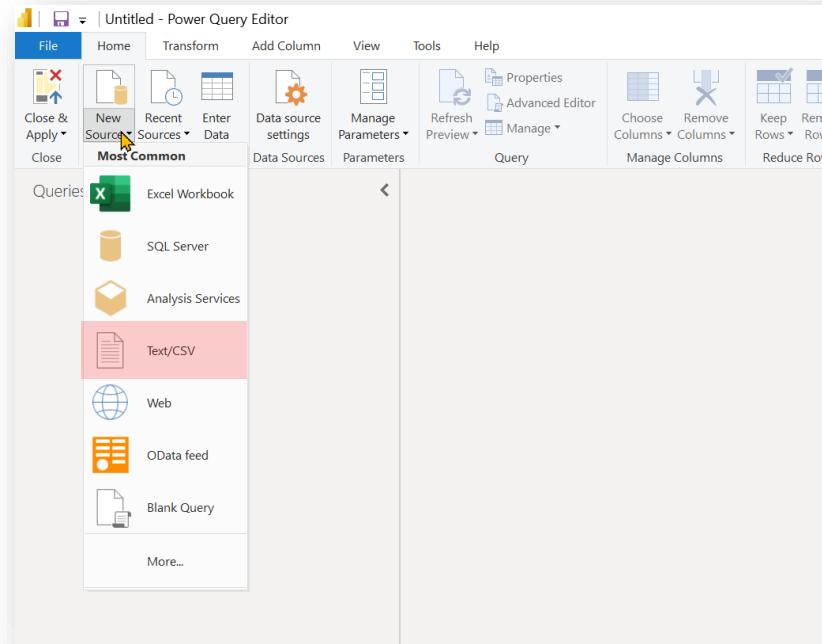
The screenshot shows the Microsoft Power Query Editor interface. The top navigation bar includes File, Home, Transform, Add Column, View, Tools, and Help. The Home tab is selected, highlighting the 'New Source' button. A red box highlights this button. Below the ribbon, the 'Most Common' data source list is visible, including Excel Workbook, SQL Server, Analysis Services, Text/CSV, Web, OData feed, and Blank Query. The main workspace displays a list of data sources. At the bottom, a link says 'Manage Columns'.

*Power Query*



# Text/csv

- Power Query automatically treats CSVs as structured files with a comma delimiter. For text files, it will try to detect if delimiter-separated values are present and infer the delimiter for structured data import.





# Text/CSV by Example

- This feature allows you to manually specify sample values from a Text/CSV file, and Power Query will automatically extract the rest of the data based on your input.

**EmployeeRole.txt**

File Origin  
1252: Western European (Windows)

Column1
ProjectManager
Jose 45 Development Samuel@company.com
DataScientist
Lucia 34 DataAnalysis Lucia@company.com
HRDirector
Andrea 47 HumanResources Andrea@company.com
MarketingCoordinator
Diego 30 Publicity Diego@company.com
OperationsManager
Isabel 39 Operations Isabel@company.com
ChiefFinancialOfficer
Carlos 54 Finanzas Carlos@company.com
QualityAssurance
Sofia 37 QA Sofia@company.com

The data in the preview has been truncated due to size limits.

**Extract Table Using Examples**

**Extract Table Using Examples - EmployeeRole.txt**

File Origin  
1252: Western European (Windows)

Column1	Column2	Column3
1 ProjectManager	Jose	45
2 DataScientist	Lucia	34
3 HRDirector	Andrea	47
4 MarketingCoordinator	Diego	30
5 OperationsManager	Isabel	39
6 ChiefFinancialOfficer	Carlos	54
7 QualityAssurance	Sofia	37
8 BusinessAnalyst	Ricardo	41
9 ComplianceOfficer	Alejandra	48
10 ITSupport	Juan	32
11		
12		

OK Cancel



# Excel

- Connecting Excel to Power Query in Power BI allows users to import and transform data from Excel spreadsheets. Here's a quick overview of the process:

The screenshot illustrates the process of connecting an Excel workbook to Power BI. On the left, the 'Data Sources' ribbon tab is selected, and the 'Excel Workbook' option is highlighted with a red arrow. The main area shows the 'Navigator' pane with a tree view of the 'Profits' sheet from the 'ExcelDataSource.xlsx' file. A red arrow points from the 'Excel Workbook' selection to the 'Profits' node in the tree. To the right, a preview of the 'Profits' table is displayed, showing columns for Region, Product, Date, Customer, Quantity, and Revenue.

Region	Product	Date	Customer	Quantity	Revenue
East	DEF	39448	Ford	1000	2
Central	DEF	39449	Verizon	100	1
East	DEF	39451	Merck	800	1
East	XYZ	39451	Texaco	400	
East	DEF	39454	State Farm	1000	2
East	ABC	39454	General Motors	400	
Central	ABC	39456	General Motors	800	1
Central	XYZ	39457	Wal-Mart	900	2
Central	ABC	39459	IBM	300	
East	XYZ	39461	AT&T	100	
East	ABC	39462	Verizon	500	
East	ABC	39463	Citigroup	600	1
West	DEF	39466	Verizon	100	
East	ABC	39468	SBC Communications	800	1
West	ABC	39468	Merck	200	
West	ABC	39470	General Motors	800	1
East	ABC	39471	IBM	600	1
Central	ABC	39472	Citigroup	1000	2
East	ABC	39473	IBM	400	
East	DEF	39476	General Motors	700	1
East	ABC	39476	Texaco	400	
East	DEF	39477	Wal-Mart	300	
East	ABC	39478	General Motors	800	1



# Connecting with SharePoint



# SharePoint

Power Query enables connections to SharePoint **document libraries and lists**, allowing users to retrieve and transform SharePoint files and folders or list data.

- **Prerequisites:** Requires appropriate access to the SharePoint **site** or **library**.
- **Capabilities Supported:** Import mode; connection to document libraries and SharePoint lists.
- **Limitations:**
  - DirectQuery is not supported.
  - Large lists may result in slower performance, and permission management can be complicated.
  - The size of files and lists can significantly impact performance, and non-Office file types may not be fully supported.





# Excel in SharePoint

*With Power Query, Excel files stored in SharePoint can be accessed and integrated into Power BI.*

- **Prerequisites:** Requires access to the SharePoint site or Excel file with proper permissions
- **Capabilities Supported:** Import mode, Excel file manipulation
- **Limitations:**
  - DirectQuery is not supported
  - Accessing multiple large sheets may slow down queries





# Utilizing Azure Blob Storage



# Azure Blob Storage

The Blob Storage connector allows Power Query to retrieve data from Azure Blob Storage in various formats, such as CSV, JSON, and images.

- **Prerequisites:** Access to Azure Storage account and permissions (Account key or SAS token)
- **Capabilities Supported:** Import mode, Binary access, hierarchical navigation of Blob containers
- **Limitations:**
  - DirectQuery is not supported
  - Handling very large blobs can slow query execution, especially for complex transformations
  - Concurrent requests and request size parameters need adjustment for optimal performance



Storage table



Storage blob



Storage queue



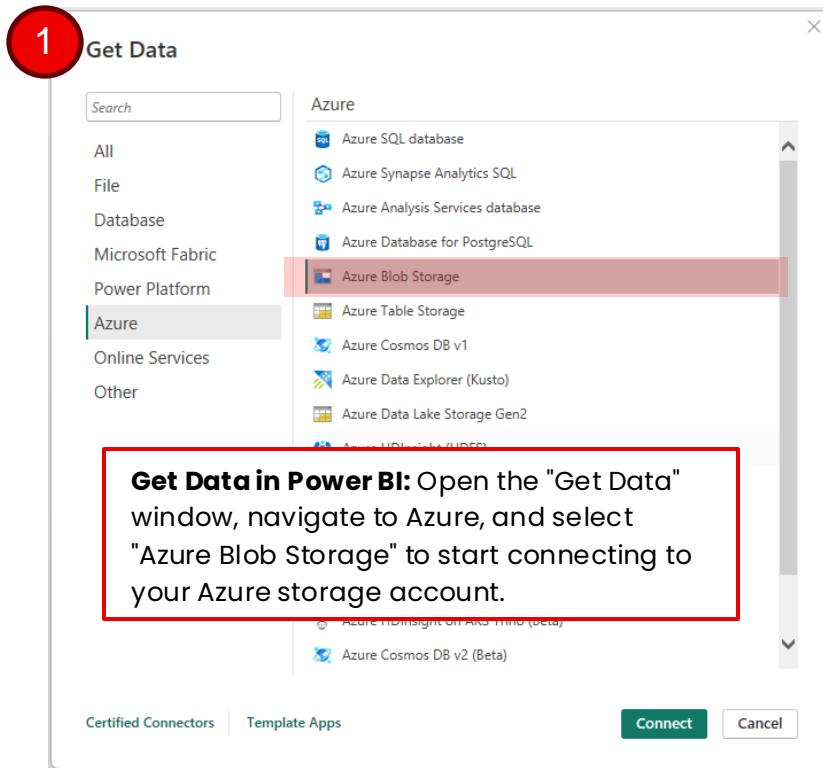
## Azure Blob Storage – Benefits

- Connecting Power BI with Azure Blob Storage offers several key benefits:
  - Cost-effective data storage
  - Flexible data access
  - Scalability for large datasets
  - Seamless data refresh capabilities
  - Secure data management with encryption and access control
  - Integration with other Azure services for enhanced analytics



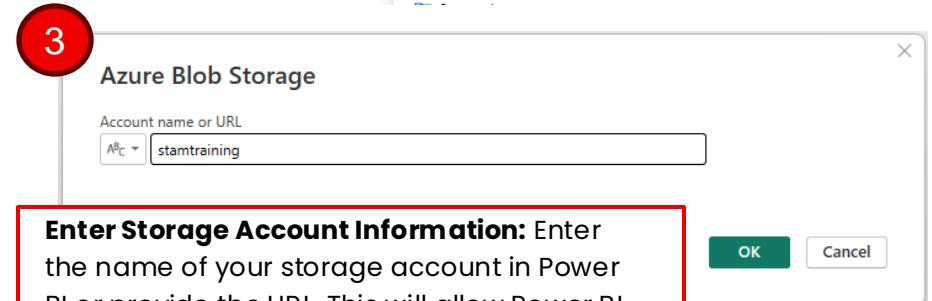
# Azure Blob Storage

To access a Blob Storage Account, follow these steps:



The screenshot shows the 'Storage accounts' page in the Microsoft Azure portal. The 'stamtraining' account is selected and highlighted with a red box and a red circle containing the number 2. A large red box encloses the following text:

**Access the Azure Portal:** In Azure, go to your storage account (in this case, "**stamtraining**") to locate your storage options and access your account information. ([portal.azure.com](https://portal.azure.com))





# Azure Blob Storage

To access a Blob Storage Account, follow these steps:

The screenshot shows the Azure Portal interface with two numbered steps highlighted by red circles.

**Step 4:** Shows the "stamtraining" Storage account overview page. The "Access keys" option in the left sidebar is highlighted with a red box and a yellow hand cursor icon pointing at it. The "Key1" and "Key2" sections are partially visible below.

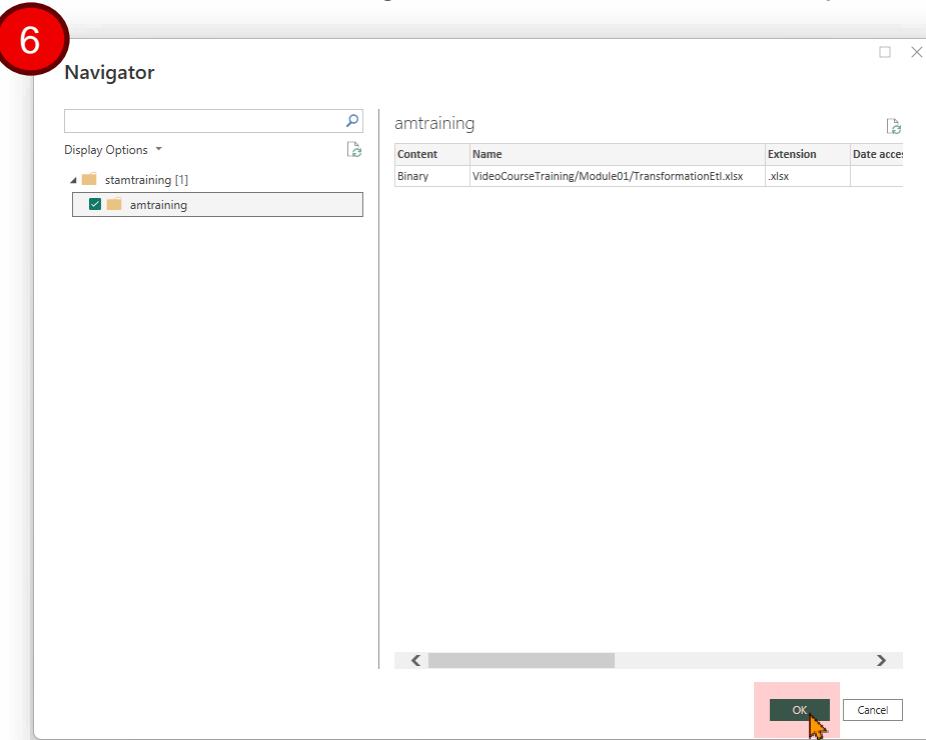
**Step 5:** Shows the "stamtraining | Access keys" page. The "Access keys" option in the left sidebar is highlighted with a red box and a yellow hand cursor icon pointing at it. The "Key1" and "Key2" sections are fully visible, showing "Key" and "Connection string" fields with red boxes around them and "Show" buttons to the right.

**Access Keys in Azure Portal:** Navigate to "Access keys" in your Azure Storage account to get the credentials required to connect securely to your storage from Power BI.



# Azure Blob Storage

To access a Blob Storage Account, follow these steps:



**Select Data in Navigator:** Once connected, Power BI will show the available files in the Blob Storage. Select the folder you want to connect to and click "OK."



# Connecting with Local Folders



# Local Folder

Power Query connects to local or network folders, enabling you to aggregate and transform data from multiple files within a directory.

- **Prerequisites:** Local or network access to the folder; permissions to read the files.
- **Capabilities Supported:** Import mode, file combination.
- **Limitations:**
  - No DirectQuery support
  - Network paths may require additional permission setups, and inconsistent file structures can complicate transformations
  - Large datasets may take longer to load, and the connector doesn't work well with very deeply nested folder structures



# Custom Functions to Combine Files



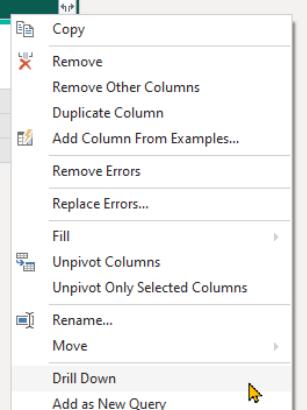
Dynamically combine multiple Excel files from a folder, leveraging functions **like Excel.Workbook** to extract and promote headers, and **Table.Combine** to merge data from each file seamlessly.

1

= Table.AddColumn(#"Removed Other Columns", "Custom", each Excel.Workbook([Content],true))

	ABC	Custom
1	Valid Error Empty	100 % 0 % 0 %
2	Valid Error Empty	100 % 0 % 0 %
3	Binary	Table
4	Binary	Table
5	Binary	Table

2



The Excel.Workbook function is used with the true parameter to promote the headers automatically from each file.

Before combining the data, we need to generate a list of tables from the files. To do this, drill-down by clicking on the header of the column.



# Custom Functions to Combine Files

QUICK  
TIPS

Dynamically combine multiple Excel files from a folder, leveraging functions **like Excel.Workbook** to extract and promote headers, and **Table.Combine** to merge data from each file seamlessly.

3

✓ fx = #Removed Other Columns1[Data]

List  
Table  
Table  
Table

This list will be used as a parameter in the **Table.Combine** function.

4

= Table.Combine(#Removed Other Columns1[Data])

A <sup>B</sup> C	Name	A <sup>B</sup> C	ProductNumber	A <sup>B</sup> C	Color	12	StandardCost
● Valid	100 %	● Valid	100 %	● Valid	83 %	● Valid	100 %
● Error	0 %	● Error	0 %	● Error	0 %	● Error	0 %
● Empty	0 %	● Empty	0 %	● Empty	17 %	● Empty	0 %
1 HL Mountain Frame - Black, 46	FR-M94B-46	Black				739,041	
2 HL Mountain Frame - Black, 38	FR-M94B-38	Black				739,041	
3 HL Mountain Frame - Silver, 38	FR-M94S-38	Silver					
4 Sport-100 Helmet, Blue	HL-U509-B	Blue					
5 AWC Logo Cap	CA-1098	Multi					
6 Long-Sleeve Logo Jersey, S	UJ-0192-S	Multi					
7 Long-Sleeve Logo Jersey, M	UJ-0192-M	Multi				38,4923	
8 Long-Sleeve Logo Jersey, L	UJ-0192-L	Multi				38,4923	
9 Long-Sleeve Logo Jersey, XL	UJ-0192-X	Multi				38,4923	
10 HL Road Frame - Red, 62	FR-R92R-62	Red				868,6342	
11 HL Road Frame - Red, 44	FR-R92R-44	Red				868,6342	

The **Table.Combine** function is used to merge all data without hardcoding the column structure.

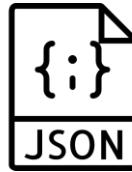


# Connecting to Other File Types (JSON, PDF, Web Content)



# Connecting to Other File Types

- Many users work with M by accessing files and folders from cloud storage, network drives, or even locally.
- Power Query supports a variety of formats like **JSON**, **XML**, **PDF**, and more, each with its own M function to make data extraction and transformation easy and efficient.





# JSON

*Power Query can retrieve and transform data from JSON files, typically used for APIs or data exchange in structured format.*

- **Prerequisites:** JSON file or access to an API providing JSON data.
- **Capabilities Supported:** Import mode, deep parsing of nested structures.
- **Limitations:**
  - Complex or deeply nested JSON structures can slow down performance
  - API rate limits can affect how much data you can pull





## PDF

The PDF connector in Power Query extracts structured data (tables) from PDF files, making it easy to pull information from scanned or formatted documents.

- **Prerequisites:** Requires access to PDF files.
- **Capabilities Supported:** Import mode, table extraction.
- **Limitations:**
  - Only basic table structures are accurately extracted
  - Complex layouts may require manual adjustments
  - Not all PDFs are parsed correctly, especially for multi-page or complexly formatted documents
  - Performance decreases with larger files





## Web Content

*Power Query allows users to retrieve data from web pages or APIs, supporting dynamic web content extraction.*

- **Prerequisites:** Public or API URL; for API, may require authentication.
- **Capabilities Supported:** Import mode, handling HTML and JSON content.
- **Limitation:**
  - Performance depends heavily on website or API response time.
  - Parsing HTML structures can be inconsistent, depending on the site's format.
  - Websites with dynamic content (JavaScript-generated data) are not supported unless APIs are used.





# Connecting to SQL Database and Other Connectivity Strategies



# Azure SQL

Azure SQL connector provides access to cloud-based SQL databases, allowing for both import and DirectQuery modes in Power BI.

- **Prerequisites:** Requires an Azure SQL Database and connection credentials (login, password, or AAD authentication).
- **Capabilities Supported:** Import mode or DirectQuery providing query folding.
- **Limitations:**
  - Complex queries in DirectQuery mode can affect performance
  - Query folding may not occur for advanced transformations
  - Complex joins and nested queries may perform slowly under DirectQuery. It's recommended to optimize SQL queries at the database level





# Exercise 2

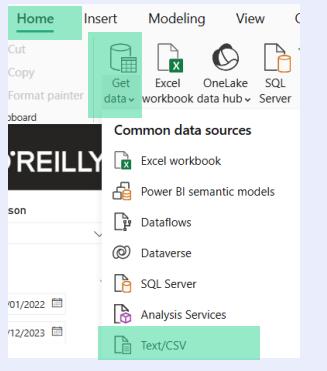
Load new files



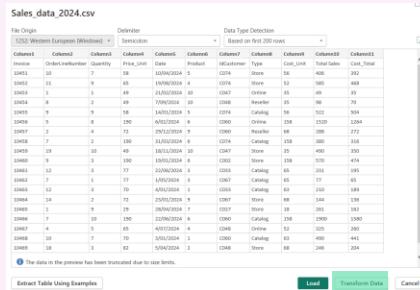
## Exercise 2: Load new files

**Goal:** Learn to load different types of files.

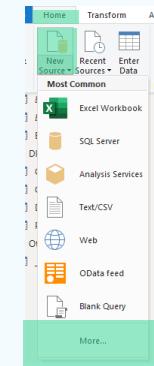
- 1 Load 'Sales\_data\_2024' table:  
Select **Home**, later **Text/SCV**



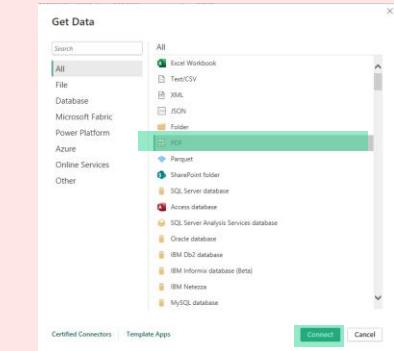
- 2 Select the path where the file **Sales\_data\_2024** is stored and click **Open**, then **Transform Data**.



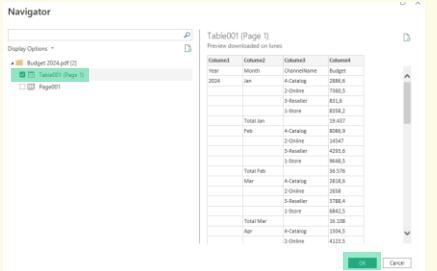
- 3 Load 'Budget\_2024' Select **Home**, **New Source**, and **More**



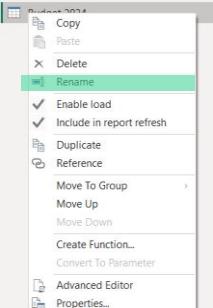
- 4 In window **Get Data**, select **PDF** and later **Connect** button



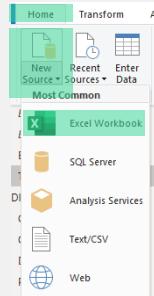
- 5 Browse to the location where the file **Budget\_2024.pdf** is stored and click **Open**. After that, check 'Table 001 (page 1)' to 'Budget 2024' and click **OK**.



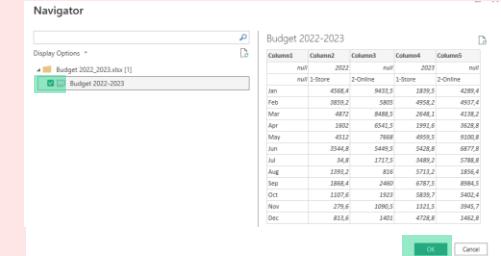
- 6 Change the table name from 'Table 001 (page 1)' to 'Budget 2024'



- 7 Load 'Budget\_2022\_2023' Select **Home**, **New Source/Excel Workbook**



- 8 Select the path where the file **Budget\_2022\_2023** is stored and click **Open**. Check 'Budget 2022-2023', and **OK** button



# DEMO 03

- Organizing Queries





# Exercise 3

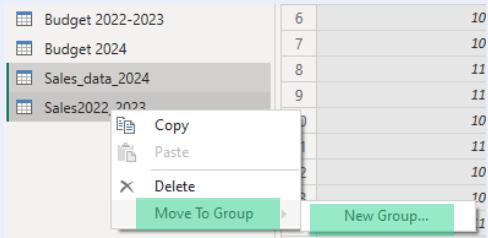
## Organizing queries



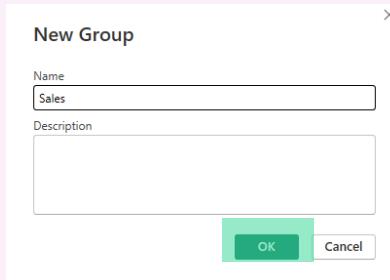
## **Exercise 3: Organize the project, before starting to work**

**Goal:** Organize the project tables into folders.

- Go to Home > Transform Data
  - Hold down **Ctrl** and select 'Sales 2022\_2023' and 'Sales\_data\_2024'
  - Right-click, choose Move to Group > New Group.

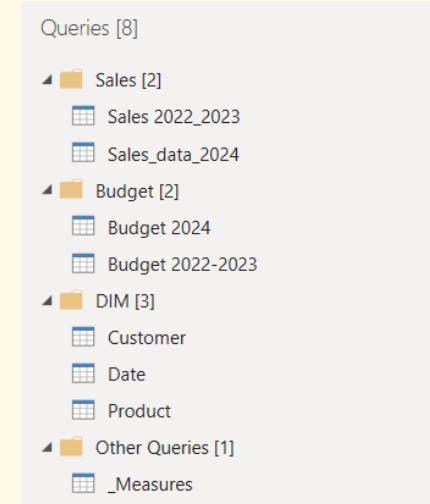


- 2 Name the folder 'Sales' and click **OK**



Automatically, those tables will be move in the new folder 'Sales'

- Select the queries 'Budget 2022-2023' and 'Budget 2024'.
  - Repeat Step 1 and 2, and rename the new folder to 'Budget'.
  - Next, select the tables Customer, Date, and Product.
  - Create a new folder and name it 'DIM'.





# Break time



# Getting Started with M



# The powerful M-language

- **Power Query is often seen as a no-code or low-code tool.**
- However, **to build more complex queries** and fully leverage Power Query's capabilities, **understanding its formula language, M, is essential.**



## What is M?

- A **Functional language** that is widely used across **Microsoft platforms**.
- Integrated into **Power BI Desktop, Excel, Dataflows, Power Platform, SQL Server, and Dynamics**.
- Allows you to **extract, transform and load data** from multiple sources like databases, Excel files, and web services.



# M in Power Query (Power BI Desktop)

- During this training, **we will focus on how M operates within Power Query in Power BI Desktop.**
- Every action performed through the UI (merging tables, filtering, etc.) generates M code in the background.



# Flexibility, Customization & Efficiency with M

## Why Learn M?

- Tailor data transformations to meet specific needs.
- Writing efficient M code can significantly improve the performance of your queries.



# Understanding the M Code Behind Your Steps (COPILOT)

# Experience-Generated Code (UI) in Power Query

Power Query's UI enables us to perform complex data transformations, with just a few clicks—no coding required.

The diagram illustrates the process of generating code through the Power Query user interface. It shows the ribbon menu, the 'Transform' tab selected, and the 'Merge Columns' dialog box. Three red arrows point from the dialog box down to the resulting transformed data table.

**Power Query ribbon:**

- Archivo
- Home
- Transform
- Add Column
- View
- Tools
- Help

**Merge Columns dialog box:**

Choose how to merge the selected columns.

Separator: Space

New column name (optional): Full Name

**Transformed Data Table:**

	Full Name	Suffix	CompanyName
1	OrlandoN.Gee	null	A Bike Store
2	KeithHarris	null	Progressive Sports
3	DonnaF.Carreras	null	Advanced Bike Com
4	JanetM.Gates	null	Modular Cycle Syst
5	LucyHarrington	null	Metropolitan Sport
6	RosmarieL.Carroll	null	Aerobic Exercise Co
7	DominicP.Gash	null	Associated Bikes
8	KathleenM.Garza	null	Rural Cycle Empori

# The M Code Behind Your Steps

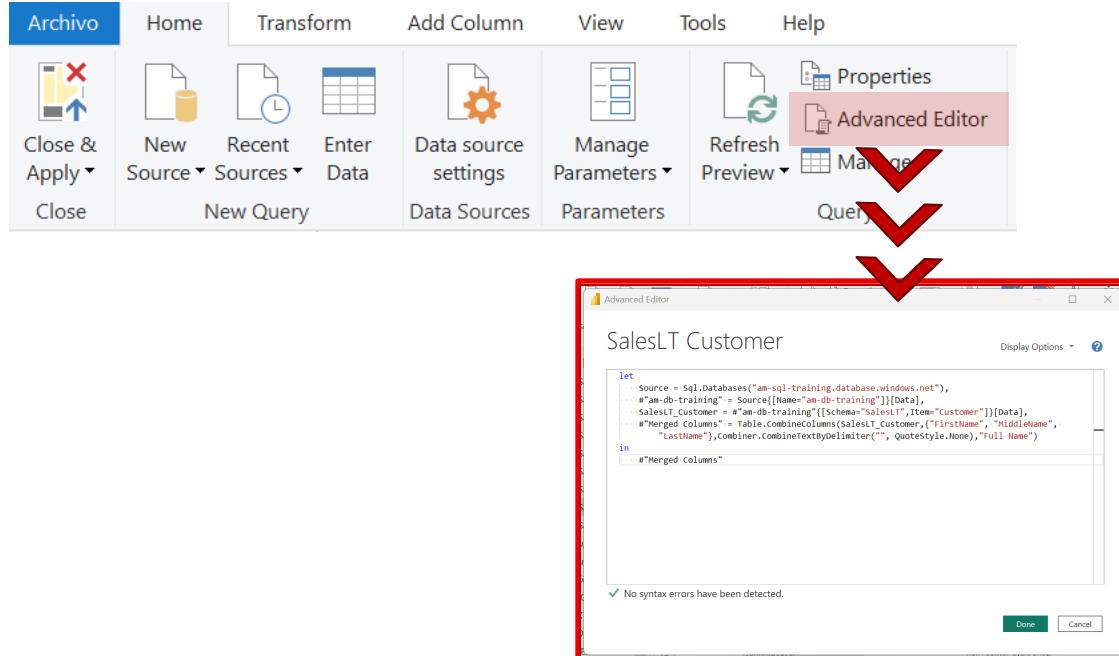
Each step in Power Query generates a corresponding variable in M code.

The screenshot shows the Power Query Editor interface. The top navigation bar includes Archivo, Home, Transform, Add Column, View (which is highlighted in pink), Tools, and Help. The View tab has several options: Formula Bar (checked), Monospaced (unchecked), Column distribution (unchecked), Show whitespace (checked), Column profile (unchecked), Always allow (unchecked), Go to Column (button), Advanced Editor (button), and Query Dependencies (button). Below the View tab are sections for Layout, Data Preview, Columns, Parameters, Advanced, and Dependencies. The Data Preview section shows a table with columns: Full Name, Suffix, CompanyName, SalesPerson, and Email. A red box highlights the formula bar with the M code: `= Table.CombineColumns(SalesLT_Customer, {"FirstName", "MiddleName", "LastName"}, Combiner.CombineTextByDelimiter("", QuoteStyle.None), "Full Name")`. To the right of the preview is a Query Settings pane. The Properties section shows the name as SalesLT Customer. The Applied Steps section lists three steps: Source, Navigation, and Merged Columns. A red box highlights the Merged Columns step. The table data is as follows:

	Full Name	Suffix	CompanyName	SalesPerson	Email
1	OrlandoN.Gee	null	A Bike Store	adventure-works\pamelal0	
2	KeithHarris	null	Progressive Sports	adventure-works\david8	keith
3	DonnaF.Carreras	null	Advanced Bike Components	adventure-works\jillian0	donn
4	JanetM.Gates	null	Modular Cycle Systems	adventure-works\jillian0	janet
5	LucyHarrington	null	Metropolitan Sports Supply	adventure-works\shu0	lucyC
6	RosmarieJ.Carroll	null	Aerobic Exercise Company	adventure-works\linda3	rosm
7	DominicP.Gash	null	Associated Bikes	adventure-works\shu0	domi
8	KathleenM.Garza	null	Rural Cycle Emporium	adventure-works\josé1	kathl

# Advanced Editor

To view the full M code generated by your query steps, **click on 'Advanced Editor'** in Power Query. This allows you to see and modify the code behind the transformations , giving us flexibility to perform advanced transformations and customize our data processing.





# Copilot to comment your code

**Adding comments to your code is crucial for understanding and maintaining your queries, especially in complex transformations.**

The screenshot shows a Microsoft 365 Chat window. At the top, there's a navigation bar with a 'M365 Chat' tab and a search bar containing 'Please generate comments for t...'. Below the bar, there are icons for 'New chat', 'Copilot', and 'Copilot AI'. The main area has two sections: 'You' and 'Copilot'. In the 'You' section, a user has pasted M code that connects to a database, retrieves a table, and performs a column transformation. In the 'Copilot' section, the user has requested comments for each step. Copilot has responded with comments explaining the purpose of each line of code. The code and comments are as follows:

```
let
    // Connect to the SQL Server database at the specified server address
    Source = Sql.Databases("am-sql-training.database.windows.net"),
    #"am-db-training" = Source[[Name="am-db-training"]][Data],
    SalesLT_Customer = #am-db-training[[Schema="SalesLT",Item="Customer"]][Data],
    #"Merged Columns" = Table.CombineColumns(SalesLT_Customer,{ "FirstName", "MiddleName", "LastName"},Combiner.CombineTextByDelimiter("",QuoteStyle.None),"Full Name")
in
    #"Merged Columns"
```

Sure! Here are the comments explaining each step of the M code:

```
let
    // Connect to the SQL Server database at the specified server address
    Source = Sql.Databases("am-sql-training.database.windows.net"),
    #"am-db-training" = Source[[Name="am-db-training"]][Data],
    SalesLT_Customer = #am-db-training[[Schema="SalesLT",Item="Customer"]][Data],
    #"Merged Columns" = Table.CombineColumns(SalesLT_Customer,{ "FirstName", "MiddleName", "LastName"},Combiner.CombineTextByDelimiter("",QuoteStyle.None),"Full Name")
in
    #"Merged Columns"
```

**You can ask Copilot to generate comments for you:**

1. Copy the code from the Advanced Editor
2. Request Copilot to add comments
3. Paste the updated code back into the Advanced Editor



# Copilot to comment your code

You can **view the comments for each step** by clicking the '**i** (information) icon next to each step.

The screenshot shows a data processing interface with a table on the left and a 'Query Settings' sidebar on the right.

**Table Data:**

CompanyName	SalesPerson	Email
Bike Store	adventure-works\pamela0	orlan
gressive Sports	adventure-works\david8	keith
dvanced Bike Components	adventure-works\jillian0	donn
Iodular Cycle Systems	adventure-works\jillian0	janet
etro	Combine the "FirstName", "MiddleName", and "LastName" columns into a single column named "Full Name"	
erobic Exercise Company	adventure-works\linda3	rosm
associated Bikes	adventure-works\laura0	donn

**Query Settings Sidebar:**

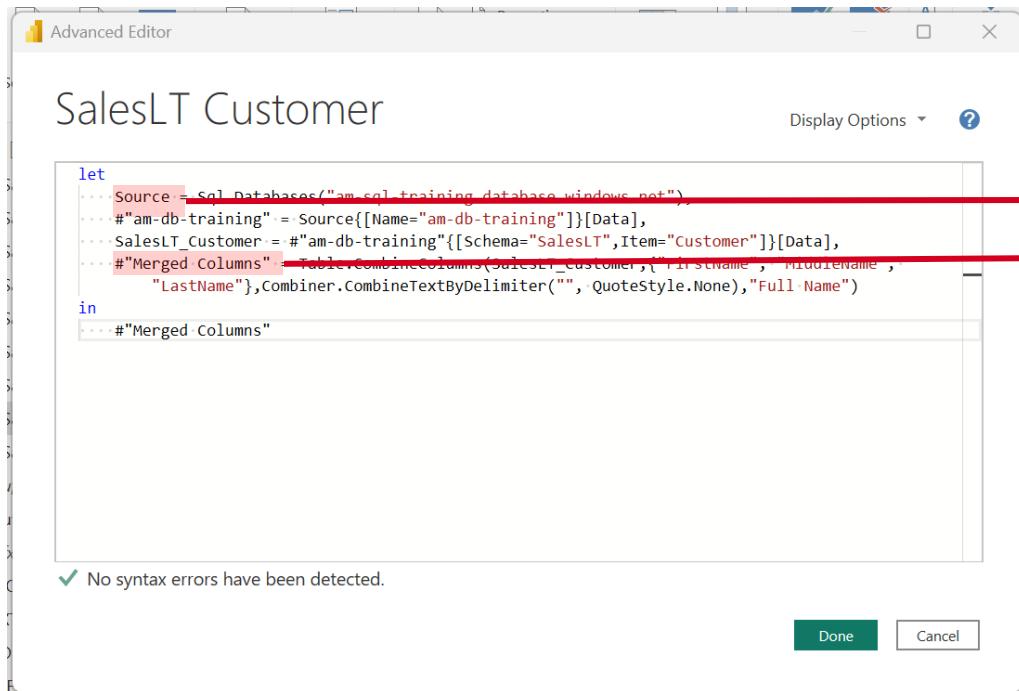
- Query Settings** (with an 'X' icon)
- PROPERTIES**
  - Name: SalesLT Customer
  - All Properties
- APPLIED STEPS**
  - Source
  - Navigation
  - Merged Columns (highlighted with a red border)



# Effective Naming and Commenting in M

# Naming the steps

Using **clear, descriptive step names**—whether with or without spaces—improves the understanding and maintenance of your code, especially in complex queries.



The screenshot shows the Microsoft Power Query Advanced Editor window. The title bar says "Advanced Editor" and the main area has a title "SalesLT Customer". The code pane contains the following M code:

```
let
    Source = Sql.Databases("am_db_training"),
    #"am-db-training" = Source[[Name="am-db-training"]][Data],
    SalesLT_Customer = #"am-db-training"{[Schema="SalesLT",Item="Customer"]}[Data],
    #"Merged Columns" = Table.CombineColumns(SalesLT_Customer, {"FirstName", "MiddleName", "LastName"}, Combiner.CombineTextByDelimiter("", QuoteStyle.None), "Full Name")
in
    #"Merged Columns"
```

Two red arrows point from the code to the right. The first arrow points to the line "Source = Sql.Databases("am\_db\_training")". The second arrow points to the line "#"am-db-training" = Source[[Name="am-db-training"]][Data]".

 No syntax errors have been detected.

Buttons at the bottom: Done and Cancel.

Without spaces:

With spaces: **#"Step With Spaces"**



# How to Comment in M

**Use comments to make your M code easier to understand and maintain.**

# SalesLT Customer

## Display Options ▾

?

```
let
    /* This is a multi-line
    comment */
    Source = Sql.Databases("am-sql-training.database.windows.net"),
    #"am-db-training" = Source{[Name="am-db-training"]}[Data],
    SalesLT_Customer = #"am-db-training"{[Schema="SalesLT",Item="Customer"]}[Data]
    in
        // This is a single-line comment
        #Merged Columns" = Table.CombineColumns(SalesLT_Customer, {"FirstName",
            ("", QuoteStyle.None), "Full Name"})
        #Merged Columns"
```

Multi-line comment: /\* This is a multi-line comment \*/

Single-line comment: // This is a comment

# Case Sensitivity in M

M is **case-sensitive**, meaning 'Step1' and 'step1' are treated as two separate variables. This applies to all functions, expressions, and variable names.



The screenshot shows the Advanced Editor interface with the title "SalesLT Customer". The code is as follows:

```
let
    Source = Sql.Databases("am-db-training;database=windows.net"),
    #"am-db-training" = Source{[Name="am-db-training"]}[Data],
    SalesLT_Customer = #"am-db-training"{[Schema="SalesLT",Item="Customer"]}[Data],
    #"Merged Columns" = Table.CombineColumns(SalesLT_Customer, {"", "}),
    ("", QuoteStyle.None), "Full Name")
in
    #"Merged Columns"
```

Annotations with red arrows and text indicate errors:

- An arrow points from "sql.databases" to the text "sql.databases → ERROR".
- An arrow points from "Table.CombineColumns" to the text "table.CombineColumns→ ERROR".
- An arrow points from the variable name "#"Merged Columns"" to the text "#"merged Columns"→ ERROR".

At the bottom left, there is a green checkmark and the text "No syntax errors have been detected." At the bottom right are "Done" and "Cancel" buttons.



# Exploring the M Code Structure

# Commas & Structure

SalesLT Customer → M code starts with LET

```
let Source = Sql.Databases("am-sql-training.database.windows.net"),
    #"am-db-training" = Source{[Name="am-db-training"]}[Data],
    SalesLT_Customer = #"am-db-training"{{Schema="SalesLT",Item="Customer"}}[Data],
    #"Merged Columns" = Table.CombineColumns(SalesLT_Customer,{ "FirstName", "MiddleName",
        "LastName"},Combiner.CombineTextByDelimiter("", QuoteStyle.None), "Full Name")
in #"Merged Columns"
```

↓ M code ends with IN

✓ No syntax errors have been detected.

Done Cancel

Every step in M ends with a comma, except the last one.

# M Steps as Variables

**Each step is a variable that stores the result of a calculation or transformation.**

These steps can return tables, records, or values, which can be reused in subsequent steps.

## SalesLT Customer

Display Optic

```
let
    Source = Sql.Databases("am-sql-training.database.windows.net"),
    #"am-db-training" = Source{[Name="am-db-training"]}[Data],
    SalesLT_Customer = #"am-db-training"{[Schema="SalesLT",Item="Customer"]}[Data],
    #"Merged Columns" = Table.CombineColumns(SalesLT_Customer, {"FirstName", "MiddleName", "LastName"}, Combiner.CombineTextByDelimiter
        (" ", QuoteStyle.None), "Full Name")
in
    Source
```

Name	Type	Kind
1 am-db-training	Table	Database

## SalesLT Customer

Display Optic

```
let
    Source = Sql.Databases("am-sql-training.database.windows.net"),
    #"am-db-training" = Source{[Name="am-db-training"]}[Data],
    SalesLT_Customer = #"am-db-training"{[Schema="SalesLT",Item="Customer"]}[Data],
    #"Merged Columns" = Table.CombineColumns(SalesLT_Customer, {"FirstName", "MiddleName", "LastName"}, Combiner.CombineTextByDelimiter
        (" ", QuoteStyle.None), "Full Name")
in
    #"Merged Columns"
```

CustomerID	NameStyle	Title	Full Name	Suffix
1	1	FALSE	Mr.	Orlando N. Gee
2	2	FALSE	Mr.	Keith Harris
3	3	FALSE	Ms.	Donna F. Carreras
4	4	FALSE	Ms.	Janet M. Gates
5	5	FALSE	Mr.	Lucy Harrington
6	6	FALSE	Ms.	Rosmarie J. Carroll
7	7	FALSE	Mr.	Dominic P. Gash
8	10	FALSE	Ms.	Kathleen M. Garza

# Step Order & Dependencies

The IN statement references the output step, which is typically the final transformation. This step can reference earlier steps, and those steps can further reference others, creating a chain of dependencies.

The screenshot shows the Microsoft Power BI Advanced Editor window. The title bar says "Advanced Editor". The main area contains the following Power Query M code:

```
let
    Source = Sql.Databases("am-sql-training.database.windows.net"),
    #"am-db-training" = Source{[Name="am-db-training"]}[Data],
    SalesLT_Customer = #"am-db-training"{[Schema="SalesLT",Item="Customer"]}[Data],
    #"Merged Columns" = Table.CombineColumns(SalesLT_Customer, {"FirstName", "MiddleName", "LastName"}, Combiner.CombineTextByDelimiter
        (" ", quoteStyle=None), "Full Name")
in
    #"Merged Columns"
```

Red arrows highlight several parts of the code to illustrate dependencies:

- An arrow points from the word "Source" to the line `Source = Sql.Databases("am-sql-training.database.windows.net")`.
- An arrow points from the identifier `"#am-db-training"` to the line `#"am-db-training" = Source{[Name="am-db-training"]}[Data]`.
- An arrow points from the identifier `SalesLT_Customer` to the line `SalesLT_Customer = #"am-db-training"{[Schema="SalesLT",Item="Customer"]}[Data]`.
- An arrow points from the identifier `"#Merged Columns"` to the line `#"Merged Columns" = Table.CombineColumns(SalesLT_Customer, {"FirstName", "MiddleName", "LastName"}, Combiner.CombineTextByDelimiter`.
- A large red arrow points from the identifier `"#Merged Columns"` down to the final line `in  
 #"Merged Columns"`, indicating it is the final output step.

At the bottom left, there is a green checkmark icon and the text "No syntax errors have been detected." At the bottom right are "Done" and "Cancel" buttons.

# QUIZ TIME

Let's do a quick quiz to reinforce our knowledge.



## Question 1

**Which of the following statements is true about Power Query's formula language, M?**

- A) M is only used for simple data transformations and cannot handle complex queries.
- B) M is essential for creating custom functions and advanced data transformations in Power Query.
- C) M is a programming language that is only used for data visualization.
- D) M is automatically generated by Power Query, so users never need to interact with it.



## Question 2

**Why should you use comments in your M code in Power Query?**

- A) To improve the performance of your queries
- B) To prevent errors in the query
- C) To make your M code easier to understand and maintain
- D) To automatically generate the final results of the query



## Question 3

**Which of the following M code fragments contains an error based on the specified rules?**

A.

LET

$x = 5,$

$y = x + 3,$

IN

$y$

B.

LET

$x = 5,$

$y = x + 3,$

$z = y * 2$

IN

$z$

C.

LET

$x = 5$

$y = x + 3,$

IN

$y$

# DEMO 04

- Promote headers
- Fill
- Unpivot
- Transposing
- Merge columns
- Filter
- Replace





# Combining Queries

- Business analysts need to perform complex transformations that usually involve **a combination of multiple queries**.
- Combining queries is essential for performing complex transformations in scenarios where different tables or files store fragmented pieces of information.
- Users typically need to **merge data horizontally** to enrich a table with **additional columns** that are not available within the main query when it is loaded from a data source or append tables to gather all relevant information.
- In **Power Query** you can transform data and also combine queries in two ways:

Merge

Allows users to combine two or more queries by matching records based on one or more common fields (keys), effectively adding new columns from another table.

Append

Enables users to **combine data vertically** by stacking rows from different tables into one. It is especially useful when datasets share the same structure but contain different periods, locations, or categories.

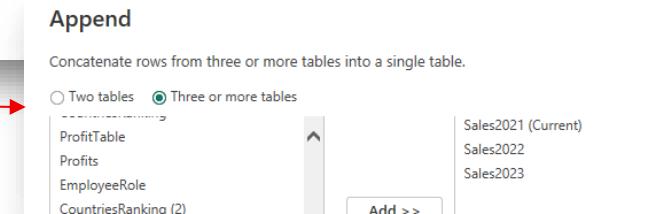
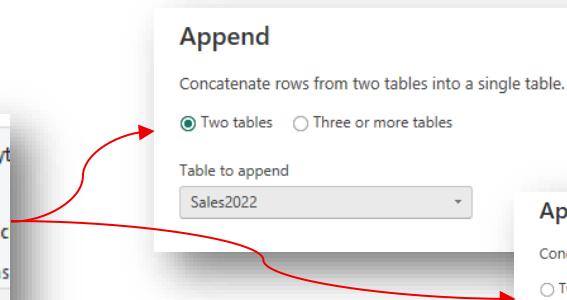
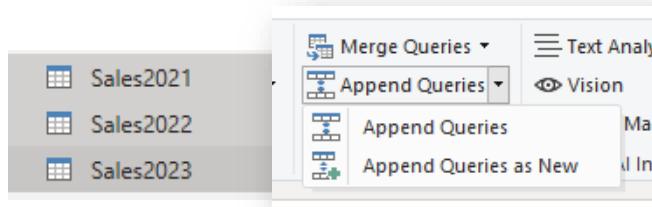


# Appending Queries

- Appending queries enables users to combine data vertically, stacking multiple tables into one.
- Use Case:
  - Appending is used when you need to combine datasets with the same structure (columns).



**Example:** appending monthly or yearly sales reports from different regions into one master file. Each region's data has the same columns, and appending helps you consolidate these datasets.





# Unpivoting and Transposing

- Pivoting, unpivoting, and transposing data are powerful methods for reshaping tables in Power Query.
- Use Case:
  - Unpivoting** is useful when you have column headers that contain values, and you need to convert them into rows.
  - Transpose** is helpful when you need to switch rows and columns.



**For instance**, transforming columns like "January Sales" and "February Sales" into a single "Month" column with a corresponding "Sales" column.



**Example:** A wide table of monthly sales per city can be unpivoted to turn months into rows for a cleaner time-series analysis.



# Grouping Data

- Grouping data is a key functionality for summarizing and analyzing **large datasets**.
- Use Case:
  - Summarize data by specific categories, like summing up total sales by region or counting the number of orders by customer.



**Example:** If you have a dataset with daily transactions, you could group by "CustomerID" to get the total spending per customer, or by "Region" to summarize sales per region.

	Name	ParentProductCategoryName	ProductNumber	Color
1	HL Mountain Frame - Black, 46	Components	FR-M94B-46	Black
2	HL Mountain Frame - Black, 38	Components	FR-M94B-38	Black
3	HL Mountain Frame - Silver, 38	Components	FR-M94S-38	Silver
4	Sport-100 Helmet, Blue	Accessories	HL-U509-B	Blue
5	AWC Logo Cap	Clothing	CA-1098	Multi
6	Long-Sleeve Logo Jersey, S	Clothing	U-0192-S	Multi
7	Long-Sleeve Logo Jersey, M	Clothing	U-0192-M	Multi
8	Long-Sleeve Logo Jersey, L	Clothing	U-0192-L	Multi
9	Long-Sleeve Logo Jersey, XL	Clothing	U-0192-X	Multi
10	HL Road Frame - Red, 62	Components	FR-R92R-62	Red
11	HL Road Frame - Red, 44	Components	FR-R92R-44	Red
12	HL Road Frame - Red, 48	Components	FR-R92R-48	Red
13	HL Road Frame - Red, 52	Components	FR-R92R-52	Red
14	HL Road Frame - Red, 56	Components	FR-R92R-56	Red
15	LL Road Frame - Black, 58	Components	FR-R38B-58	Black



Power Query ribbon:

- Group By
- Use First Row as Headers
- Transpose
- Reverse Rows
- Count Rows
- Detect Data Type
- Fill
- Replace Values
- Unpivot Columns
- Move
- Rename
- Pivot Column
- Convert to List
- Any Column
- Split Column

Queries pane:

ParentProductCategoryName	Year	Count
Components	2022	32
Accessories	2022	3
Clothing	2022	7
Bikes	2022	30
Bikes	2023	23
Components	2023	39
Clothing	2023	16
Accessories	2023	7
Clothing	2024	12
Accessories	2024	19
Components	2024	61
Bikes	2024	44



ONLY

# Fuzzy grouping in Power Query ONLINE

- Fuzzy grouping in Power Query (Only ONLINE) allows you to group similar but not exactly identical text values. It helps in scenarios where you need to consolidate data with slight differences, such as misspelled categories or product names.

	ProductID	ProductName	Category	Price
1	1	Blue Mountain Bike	Bicycles	450
2	2	Mountain Bike	Bicycle	480
3	3	Mountain Bike Pro	Bicycles	550
4	4	Road Bike	Bikes	300
5	5	Red Road Bike	Road Bicycle	350
6	6	BMX Bike	BMX	200
7	7	Off-road Bike	Offroad Bicycles	500
8	8	Classic Road Bike	Bikes	320
9	9	Cruiser Bike	Bycicle	275
10	10	Trail Bike	Bickles	450
11	11	Speed Bike	Speed Cycle	400
12	12	Offroad Mountain	Off-Road Bicycles	550
13	13	Hybrid Bike	Hybrid Bicycles	370
14	14	Hybrid Bicycle	Hybrids	365
15	15	Touring Bike	Touring Bicycles	420
16	16	Smartphone	Electronics	
17	17	Mobile Phone	Electronics	
18	18	Laptop	Computers	
19	19	Notebook	Electronics	
20	20	Gaming Laptop	Electronic Devices	
21	21	Tablet	Electronics	
22	22	Desktop PC	Computers	900
23	23	Smartwatch	Wearable Devices	300
24	24	Fitness Tracker	Wearables	250

Group by

Specify the column to group by and the desired output.

Basic  Advanced

Group by \*

New column name \*

Operation \*  Column \*

Use fuzzy grouping

Fuzzy group options

Similarity threshold

0.5

Ignore case

Group by combining text parts

By adjusting the similarity threshold, you can control how closely the values need to match to be grouped together, making it easier to clean and analyze data with variations.

Learn more

OK

Cancel

	Category	Count
1	Bicycles	12
2	Hybrid Bicycles	2
3	Electronics	5
4	Wearable Devices	2
5	Computers	2
6	BMX	1



	Category	Count
1	Bicycles	15
2	Electronics	5
3	Wearable Devices	2
4	Computers	2





# Exercise 4

Applying M language: *Working  
with a budget table*



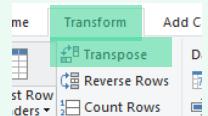
# Exercise 4: Applying Unpivoting and Transposing Data

**Goal:** Learn how to use the Unpivoting and Transposing function.

1 Go to Home > Transform data

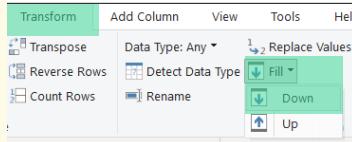
Select 'Budget 2022\_2023'.

Click on **Transform** and then **Transpose**



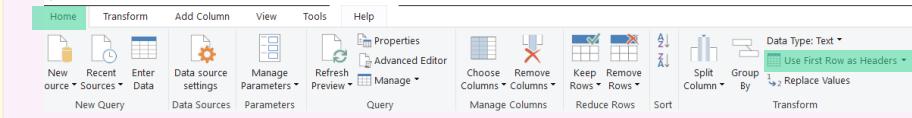
2

Select 'Column 1', click on **Transform>Fill>Down**



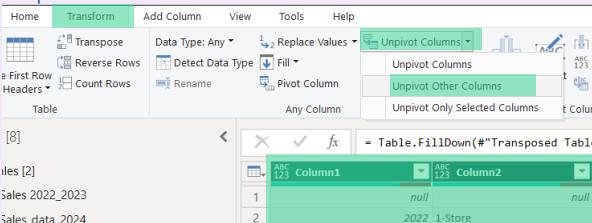
3

Select **Home** and **Use First Row as Headers**



4 Apply unpivot:

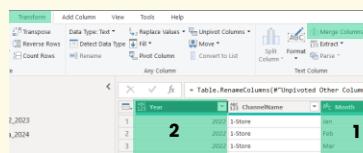
Select the columns 1 and 2, Click on **Transform > Unpivot columns > Unpivot other columns**.



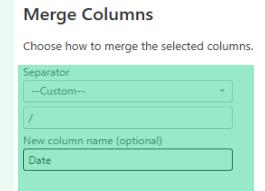
5 Rename the columns as follows:

- Column1 → Year
- Column2 → ChannelName
- Attribute → Month
- Value → Budget

6 Merge Columns: Select first column 'Month' and then 'Year'. Select **Transform/Merge Columns**.



7 Separator = Custom Type '/'  
New column = Date

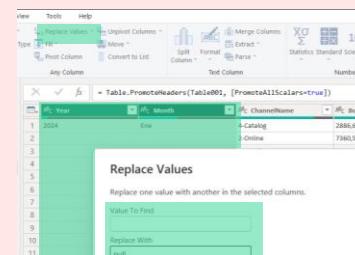


8 Select 'Budget 2024'. Click on **Home** and **Use First Row as Headers**.

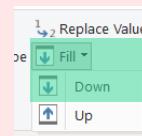


9

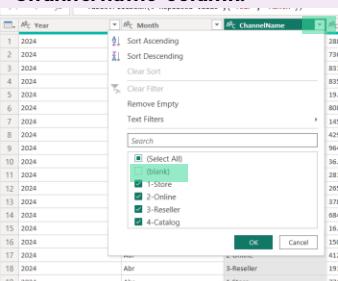
a. Select 'Year' and 'Month'. b. In the **Replace Values** Transform / Replace Values Window, write 'null' in the **Replace With** field



c. Fill Down



10 Remove blank values from the 'Channel name' column:



11 Repeat step 6 and 7, now for query 'Budget 2024'.

O'REILLY®

# Class 02





# Course Agenda

## **Class 1: Introduction and getting started with M**

- Course Framework and Introduction
- Data Sources and Connectivity
- Getting Started with M: A Step Beyond the UI
- Combining data

## **Class 2: Important tools in the M language**

- Understanding Object Structures and Data Types.
- Intermediate Data Transformation
- Advanced Query Techniques and Error Handling
- Final comments and Advanced tips



Our Method on each topic:  
Demo example, followed by exercises to practice!

- DEMO – Flor & Nico



- EXERCISE – Your turn!



\*Student should have the  
latest version of  
**Power BI Destkop installed.**



# Instructors

**Follow us on LinkedIn:**



Nicolás Lagreste Zucchini



María Florencia Hourcouripé

**Let's Collaborate:** Interested in a personalized mentorship or consulting session? Contact us:

-  [nico@analyticmood.com](mailto:nico@analyticmood.com)
-  [flor@analyticmood.com](mailto:flor@analyticmood.com)

**Visit Our Website:**

- [www.analyticmood.com](http://www.analyticmood.com) for resources, upcoming events, and blog articles.

# DEMO 05

- Append





# Exercise 5

*Organizing and Applying  
Append*

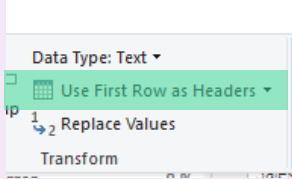


# Exercise 5: Append process

**Goal:** Learn how to use the append function

- 1 Go to Home > Transform data  
Select 'Sales\_data\_2024'.

Home > Use first rows as headers:

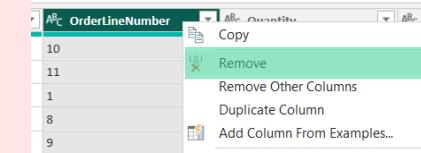


- 2 Compare the queries 'Sales\_data\_2024' and 'Sales 2022\_2023' to ensure that the columns are matching:

- a In the table 'Sales\_data\_2024', rename the following columns to match those in Sales\_2022\_2023:

- Price\_Unit → Unit Price
- Cost\_Unit → Unit Cost
- Cost\_Total → Total Cost
- Product → IdProduct

- b Remove column 'OrderLineNumber':  
First select the column, later click right



- 4 In Append Window select table 'Sales 2022\_2023' as Second Table

## Append

Concatenate rows from two tables into a single table.

Two tables  Three or more tables

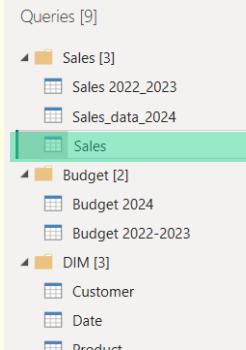
First table

Sales\_data\_2024

Second table

Sales 2022\_2023

- 5 Rename the new query to 'Sales'



- 6 Repeat steps 3 and 4 to append 'Budget 2022\_2023' and 'Budget\_2024'.

## Append

Concatenate rows from two tables into a single table.

Two tables  Three or more tables

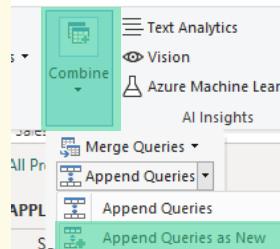
First table

Budget 2022-2023

Second table

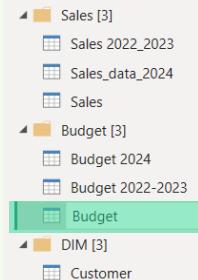
Budget 2024

- 3 Home, Combine, and Append query as a new



- 7 Rename the new query to 'Budget'

Queries [10]



# DEMO 06

- Remove duplicates
- Reference
- Duplicates
- Split
- Rename
- Merge

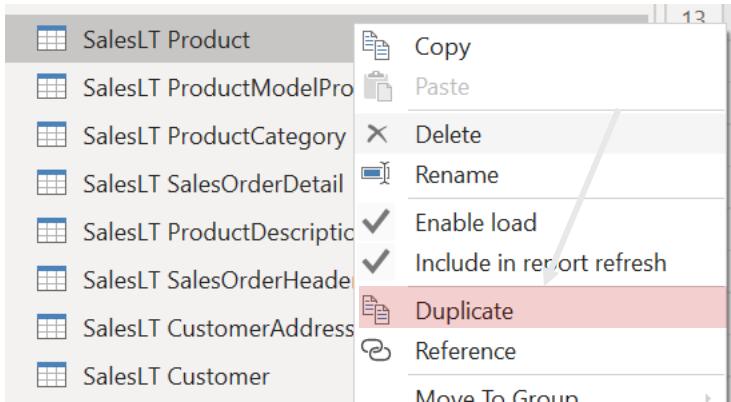




# References and Duplicates

# Duplicates

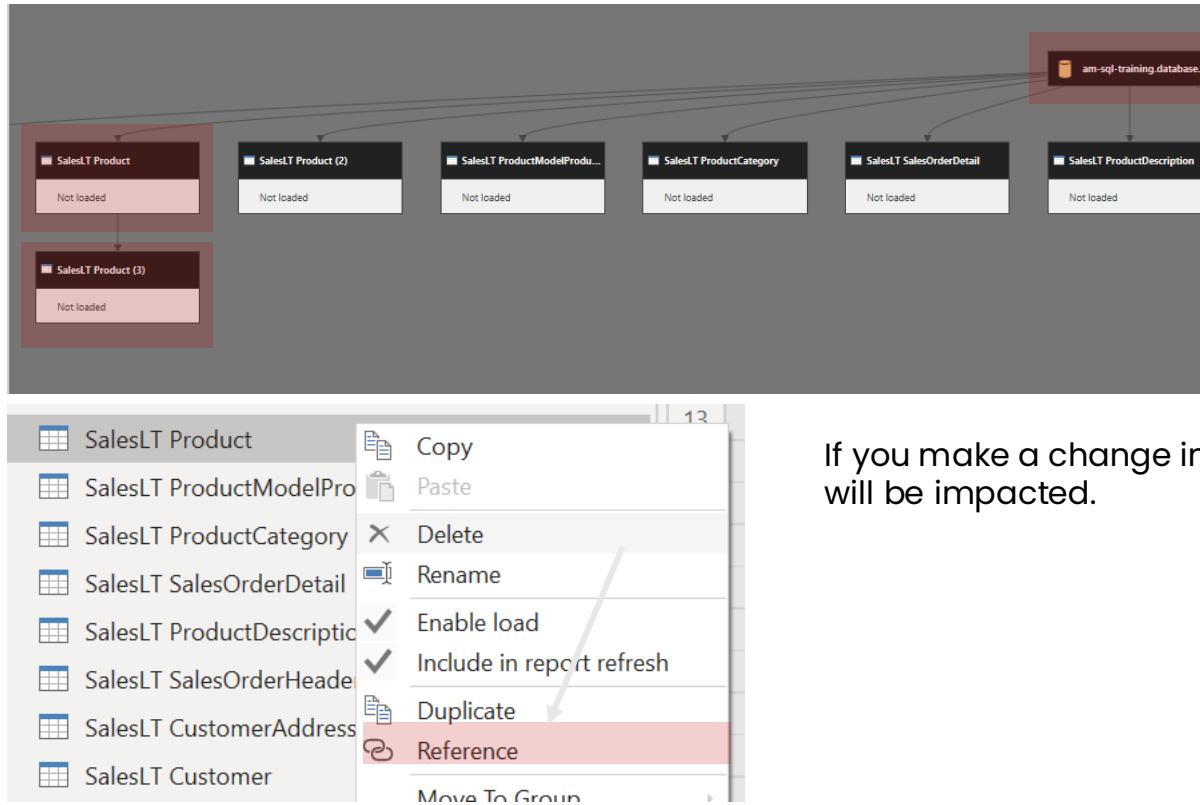
**Duplicate will duplicate the code of the query.** It copies a query with all the applied steps of it as a new query.



You can make changes in the original or the new query, and they will NOT affect each other.

# References

When you reference a query, **the new query will have only one step: sourcing from the original query.**



If you make a change in the original query, the new query will be impacted.

# Other new queries

## Append Queries as New / or Merge Queries as New is a Reference action

The screenshot shows the Power BI Data Editor interface. On the left, there's a preview of a table with columns: ProductID, Name, ProductNumber, Color, StandardCost, ListPrice, and Size. The table contains 10 rows of product data. On the right, the 'PROPERTIES' pane shows the name is set to 'Append1'. Below it, the 'APPLIED STEPS' pane shows a single step labeled 'Source'. The formula bar at the top shows the formula: `= Table.Combine({#"SalesLT Product", #"SalesLT Product (2)"}).`

ProductID	Name	ProductNumber	Color	StandardCost	ListPrice	Size
680	HL Road Frame - Black, 58	FR-R92B-58	Black	1.059,31	1.431,50	58
706	HL Road Frame - Red, 58	FR-R92R-58	Red	1.059,31	1.431,50	58
707	Sport-100 Helmet, Red	HL-U509-R	Red	13,09	34,99	
708	Sport-100 Helmet, Black	HL-U509	Black	13,09	34,99	
709	Mountain Bike Socks, M	SO-B909-M	White	3,40	9,50	M
710	Mountain Bike Socks, L	SO-B909-L	White	3,40	9,50	L
711	Sport-100 Helmet, Blue	HL-U509-B	Blue	13,09	34,99	
712	AWC Logo Cap	CA-1098	Multi	6,92	8,99	
713	Long-Sleeve Logo Jersey, S	UJ-0192-S	Multi	38,49	49,99	S
714	Long-Sleeve Logo Jersey, M	UJ-0192-M	Multi	38,49	49,99	M

# Other new queries

## Add as New Query is a Duplicate action

1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00
1:00	13/06/2008 0:00:00	08/06/2008 0:00:00

The screenshot shows the Power BI desktop interface. A context menu is open over a row in a table. The menu items are: Copy, Date/Time Filters, Replace Values..., Drill Down, and Add as New Query. The 'Add as New Query' option is highlighted with a red box. At the bottom left, there is a formula bar with the text: = SalesLT\_SalesOrderHeader{[SalesOrderID=71816]}[DueDate]. Below it, the value 13/06/2008 0:00:00 is displayed. On the right side, there are two sections: 'PROPERTIES' and 'APPLIED STEPS'. The 'PROPERTIES' section shows the name '71816' and a link to 'All Properties'. The 'APPLIED STEPS' section shows a list with 'Source' at the top, followed by 'Navigation'.

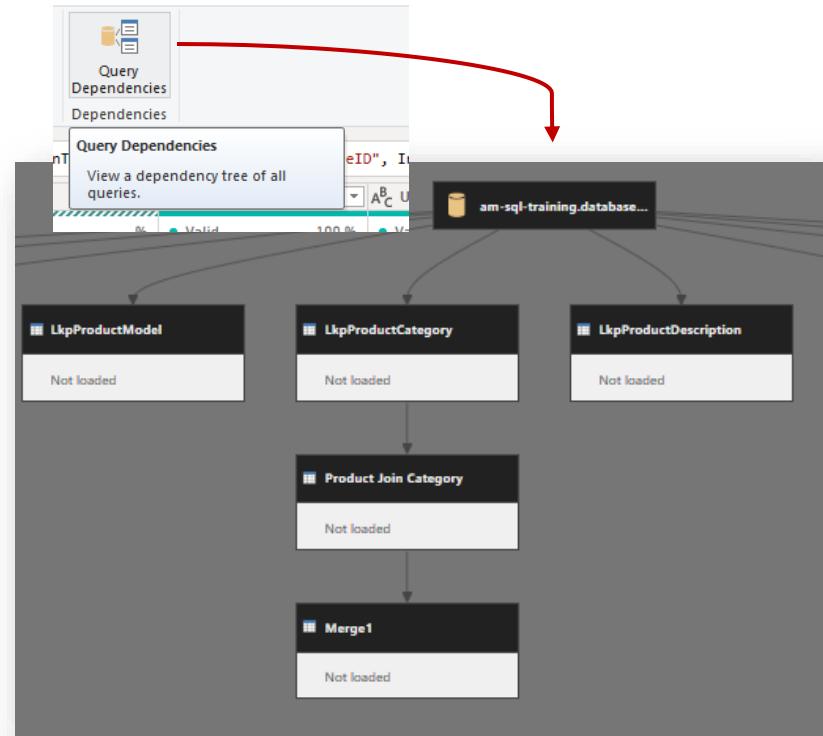
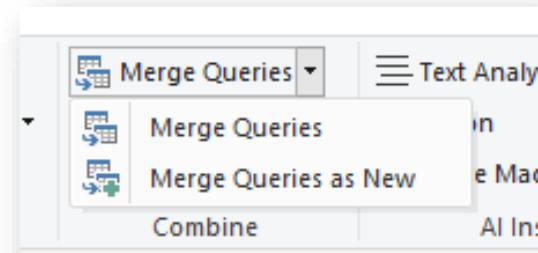


# Merging Data from Multiple Sources



# Combining Queries – Merge

- A Merge query combines two existing queries by matching rows based on a common column. This operation adds columns from the secondary table to the primary table.
- There are two types:
  - **Inline Merge:** Merges data within the same query, creating a new step.
  - **Intermediate Merge:** Creates a new query for each merge operation.

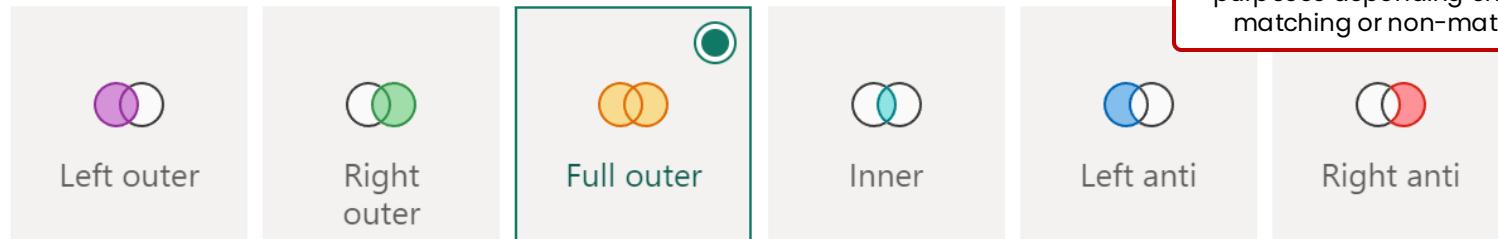




# Combining Queries – Merge

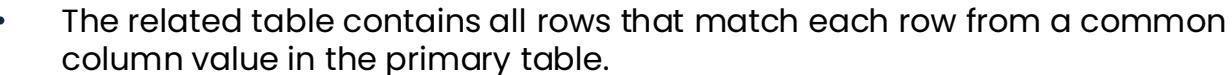
- There are several types of **joins** used to combine data from two tables based on common columns:

Join kind



- Left Outer:** Returns all rows from the first (left) table, and matching rows from the second (right) table. If no match is found, null values are returned for the right table's columns.
- Right Outer:** Returns all rows from the right table and matching rows from the left. Non-matching left table rows will return null.
- Full Outer :** Returns all rows from both tables. Where no match is found, null values are returned.
- Inner:** Returns only matching rows from both tables.
- Anti (Left/Right):** Returns rows from one table where no match is found in the other.

# Combining Queries – Merge



## Merge

Select a table and matching columns to create a merged table.

Product Join Category

ProductID	Name	ProductNumber	ListPrice	ProductCategoryID
680	HL Road Frame - Black, 58	FR-R92B-58	1431,5	18
706	HL Road Frame - Red, 58	FR-R92R-58	1431,5	18
707	Sport-100 Helmet, Red	HL-US09-R	34,99	35
708	Sport-100 Helmet, Black	HL-US09	34,99	35
709	Mountain Bike Socks, M	SO-B909-M	9,5	27



LkpProductCategory

ProductCategoryID	ParentProductCategoryID	Name
1	null	Bikes
2	null	Components
3	null	Clothing
4	null	Accessories
5	1	Mountain Bikes

Join Kind



Use fuzzy matching to perform the merge

Fuzzy matching options

The screenshot shows the 'LkpProductCategory' dialog box from Microsoft Power BI. At the top, there's a header bar with tabs for '1', '2', '3', 'ProductCategoryID', and 'LkpProductCategory'. To the right of the header are icons for 'A', 'Z', and a downward arrow. Below the header is a search bar with a placeholder '(Select All Columns)' and a clear button (X). Underneath the search bar are four checked checkboxes: '(Select All Columns)', 'ProductCategoryID', 'ParentProductCategoryID', and 'Name'. Below these checkboxes is a checked checkbox for 'Use original column name as prefix'. At the bottom right are 'OK' and 'Cancel' buttons. The background of the dialog shows a list of tables: 'Table', 'Table', 'Table', 'Table', 'Table', 'Table', 'Table', 'Table', and 'Table'.

r	1.2 ListPrice	1.3 ProductCategoryID	A B C	Name_1
100 %	● Valid	100 %	● Valid	100 %
0 %	● Error	0 %	● Error	0 %
0 %	● Empty	0 %	● Empty	0 %
	49,99		25	Jerseys
	49,99		25	Jerseys
	1364,5		16	Mountain Frames
	1364,5		16	Mountain Frames
	1364,5		16	Mountain Frames
	1364,5		16	Mountain Frames
	1349,6		16	Mountain Frames
	1349,6		16	Mountain Frames
	1349,6		16	Mountain Frames
	1349,6		16	Mountain Frames
	1364,5		16	Mountain Frames
	3578,27		6	Road Bikes
	3578,27		6	Road Bikes
	3578,27		6	Road Bikes
	3578,27		6	Road Bikes



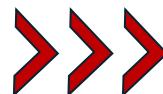
# Merging Data - Semi Joins



# Combining Queries – Left and Right Semi Join



- **Left Semi-Join** returns only the rows from the left table that have matching rows in the right table, but it doesn't bring in any columns from the right table. It effectively filters the left table to only include rows that exist in the right table based on the join column(s).
- **Right Semi-Join:** Keeps only rows from the right table that have matches in the left table, no columns from the left table are added.
- **Use Case:**
  - This is useful when you want to filter your data to only include records that have a match in another table, but you don't need any information from the matching table.





# Combining Queries – Left and Right Semi Join



In this example, we're using a **Left Semi Join** to return only the employees that have a corresponding department in the **Department** table.

Employee						
	$\text{IdEmployee}$	$\text{Name}$	$\text{Age}$	$\text{City}$	$\text{IdDepartment}$	
	Valid	100 %	Valid	100 %	Valid	100 %
	Error	0 %	Error	0 %	Error	0 %
	Empty	0 %	Empty	0 %	Empty	0 %
1	1 Alice		25	New York	8	
2	2 Bob		30	Chicago	1	
3	3 Carol		28	Boston	3	
4	4 Dave		35	Seattle	6	
5	5 Eve		22	Miami	10	

Department				
	$\text{Id}$	$\text{Department}$	$\text{Country}$	
	Valid	100 %	Valid	100 %
	Error	0 %	Error	0 %
	Empty	0 %	Empty	0 %
1	1 HR		USA	
2	3 Marketing		USA	
3	4 Finance		USA	
4	6 IT		Canada	
5	7 Sales		USA	

```
= Table.NestedJoin(#"Changed Type", {"IdDepartment"}, Department, {"Id"}, "Department", JoinKind.LeftSemi)
```

	$\text{IdEmployee}$	$\text{Name}$	$\text{Age}$	$\text{City}$	$\text{IdDepartment}$	Department
	Valid	100 %	Valid	100 %	Valid	100 %
	Error	0 %	Error	0 %	Error	0 %
	Empty	0 %	Empty	0 %	Empty	0 %
1	2 Bob		30	Chicago	1	Table
2	3 Carol		28	Boston	3	Table
			35	Seattle	6	Table

**Left Semi Join** keeps the rows from the left table (**Employee**) only where a match is found in the right table (**Department**), but it doesn't return the data from the right table.

Manual adjustment in the Power Query M code to use the `JoinKind.LeftSemi`



# Exercise 6

## References and Duplicates



# Exercise 6: Create Dimensions

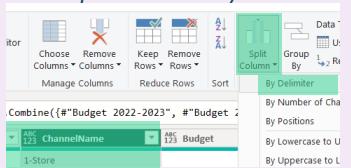
**Goal:** Learn a way to create dimensions

1 Go to Home > Transform data

Select 'Budget'

Select 'ChannelName'

Home > Split Column > By Delimiter



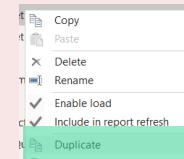
Rename Columns:

- ChannelName.1 → IdChannel
- ChannelName.2 → ChannelName

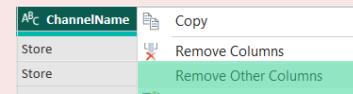
ABC ChannelName.1	ABC ChannelName.2
1	Store
1	Store
1	ABC IdChannel
1	ABC ChannelName
1	1
1	Store
1	Store

2 Duplicate query 'Budget':

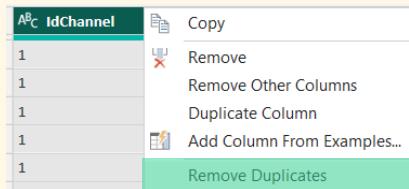
Right-click on the name of query Budget / **Duplicate**



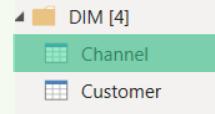
3 Remove columns 'Date' and 'Budget' using 'Remove other columns'



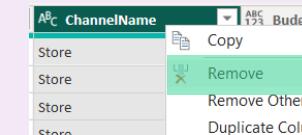
4 Remove Duplicates: Right-click on the column 'IdChannel' / **Remove Duplicates**



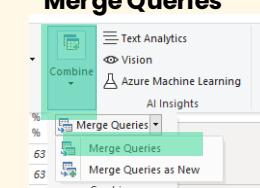
5 Rename: Rename the query to Channel and move the query to DIM folder



6 Select 'Budget' and remove 'ChannelName'

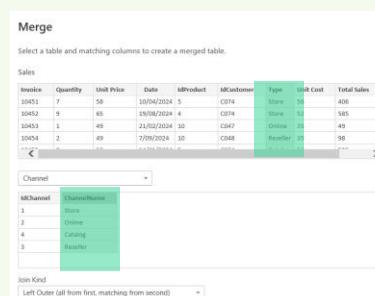


7 Select 'Sales' and click 'Merge Queries'

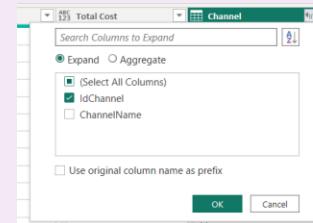


8 Select query 'Channel' and:

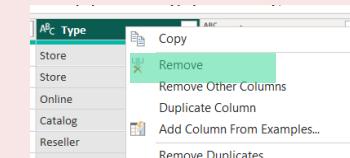
- Column 'Type' in Sales
- Column 'ChannelName' in 'Channel'.



9 Expand 'IdChannel'



10 Remove column 'Type'



# DEMO 07

- Understanding Object Structures
- Data Types Examples
- Using Locale





# Understanding Object Structures and Data Types

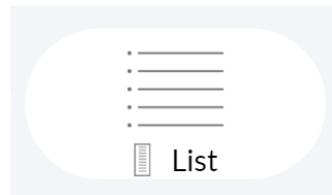


# Understanding Object Structures and Data Types

- **Structured/Abstract types**

Power Query organizes data into **five distinct value structures**.

This doesn't refer to the data type but rather to how the data itself is structured and stored.



A single columnar object with no column header

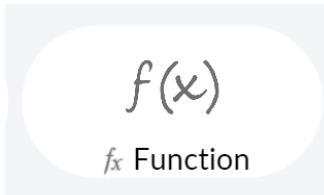
List	
1	Adjustable Race
2	All-Purpose Bike Stand
3	AWC Logo Cap
4	BB Ball Bearing
5	Bearing Ball
6	Bike Wash - Dissolver
7	Blade
8	Cable Lock
9	Chain



A row of a table with colu header

Name	Sales.SalesTerritory
Data	Table
Schema	Sales
Item	SalesTerritory
Kind	Table

Record column headers



Pre-built capabilities that can be used to transform objects

Enter Parameter  
@CustomerID (optional)  
Example: 123

Invoke Clear

function (@CustomerID as nullable)

We use functions when we have a repeating process in data manipulation.



Combination of multiples row and multiple columns

BusinessEntityID	AccountNumber
1	AUSTRALI0001
2	ALLENSON0001
3	ADVANCED0001
4	TRIKES0001
5	MORGANB0001
6	CYCLING0001
7	CHICAGO0002
8	GREENWOO0001
9	COMPETE0001
10	INTERNA0001
11	LIGHTSP0001
12	TRAINING0001



Any single part data type considered as a primitive value

X ✓ fx = "# Expanded Data"

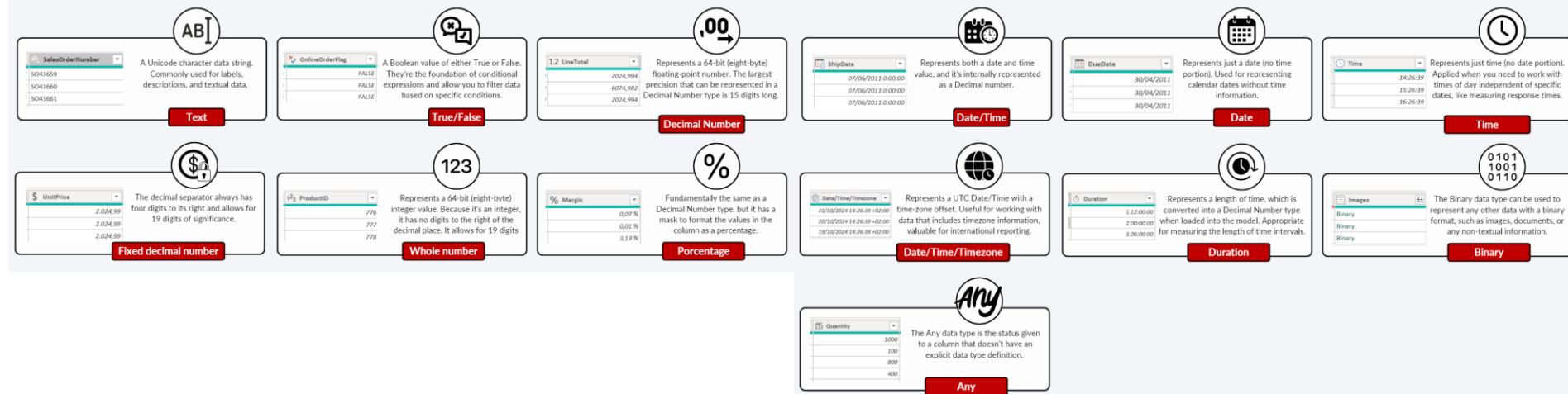
Multi



# Understanding Object Structures and Data Types

- **Primitive/Non-Abstract types**

Data types define the **nature of the information contained within each column**, facilitating data organization and analysis.





# Non-Abstract Types: Data Types

# Data types

Data types define the **nature of the information contained within each column**, facilitating data organization and analysis.

**Every column (or field) in a dataset is assigned a specific data type (such as text, number, date, etc.).** All the values within that column must match or conform to this assigned data type.





# Data types

**Primitive types are the fundamental data types from which all other data types are created.**

1. Data types play an important role in **data validation** and **preventing errors**.
2. Defining data types can make your **queries run faster**. When you set types for columns, Power Query optimizes how it stores and retrieves that data, cutting down on memory use.
3. By specifying a data type, Power Query can **better handle errors**. If Power Query encounters a value that doesn't fit the assigned type, it either rejects it or converts it, depending on your expression.
4. Data types can be important when **integrating with other systems or when exporting data**. Certain systems require specific data types for their operations or interfaces

# Turn Off Automatically Detect Column Types

1. By default, Power Query automatically adds query steps that promote headers and change data types in your query after certain transformations are performed.

2. You can modify the Power Query options settings to prevent this from happening.

Options

**GLOBAL**

- Data Load
- Power Query Editor
- DirectQuery
- R scripting
- Python scripting
- Security
- Privacy
- Regional Settings
- Updates
- Usage Data
- Diagnostics
- Preview features
- Save and Recover
- Report settings
- Copilot (preview)

**CURRENT FILE**

- Data Load
- Regional Settings
- Privacy
- Auto recovery
- Published semantic model settings
- Query reduction
- Report settings

**Type Detection**

- Always detect column types and headers for unstructured sources
- Detect column types and headers for unstructured sources according to each file's setting
- Never detect column types and headers for unstructured sources

**Background Data**

- Always allow data previews to download in the background
- Allow data previews to download in the background according to each file's setting
- Never allow data previews to download in the background

**Parallel loading of tables**

When you load data into Power BI (via import or DirectQuery), each data table is backed by a Power Query query. These queries are evaluated simultaneously instead of one-by-one, which can speed up the process. In certain situations, you might want to adjust the default number of simultaneous query evaluations and memory used. [Learn more](#)

Maximum number of simultaneous evaluations:

Maximum memory used per simultaneous evaluation (MB):

**Time intelligence**

Auto date/time for new files  [Learn more](#)

**Data Cache Management Options**

Currently used: 3,72 GB

Maximum allowed (MB):

# How to define a column data type

The data type of a column is displayed on the left side of the column heading with an icon that symbolizes the data type.



## How to define a column data type

- Data type detection
- Transform / Data Type
- By selecting the icon on the left side of the column heading.

# Data types

## Number



Whole number



Decimal Number



Fixed decimal number



Percentage

## Text



Text

## logical



True/False

## Date/ Time zone



Date/Time



Date/Time/Timezone



Date



Duration



Time

## Any



Any

## Binary

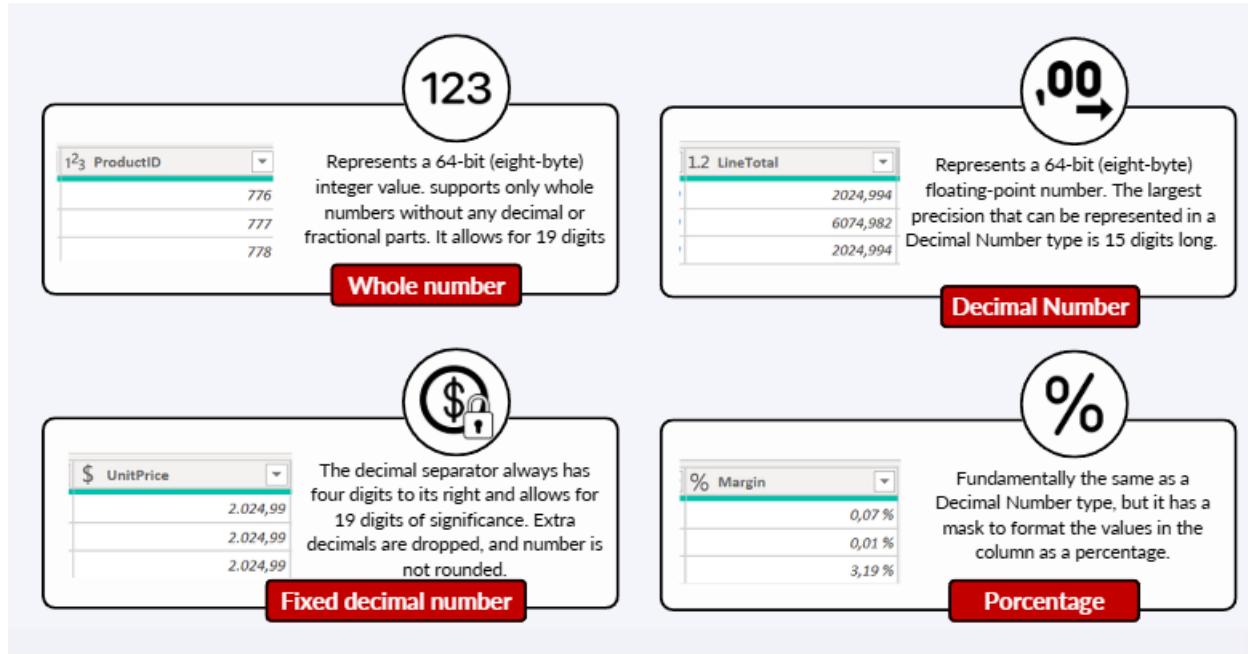


Binary

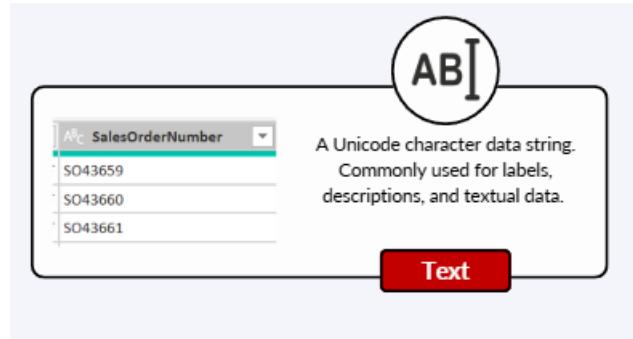
# Data types: Any



# Data types: Number

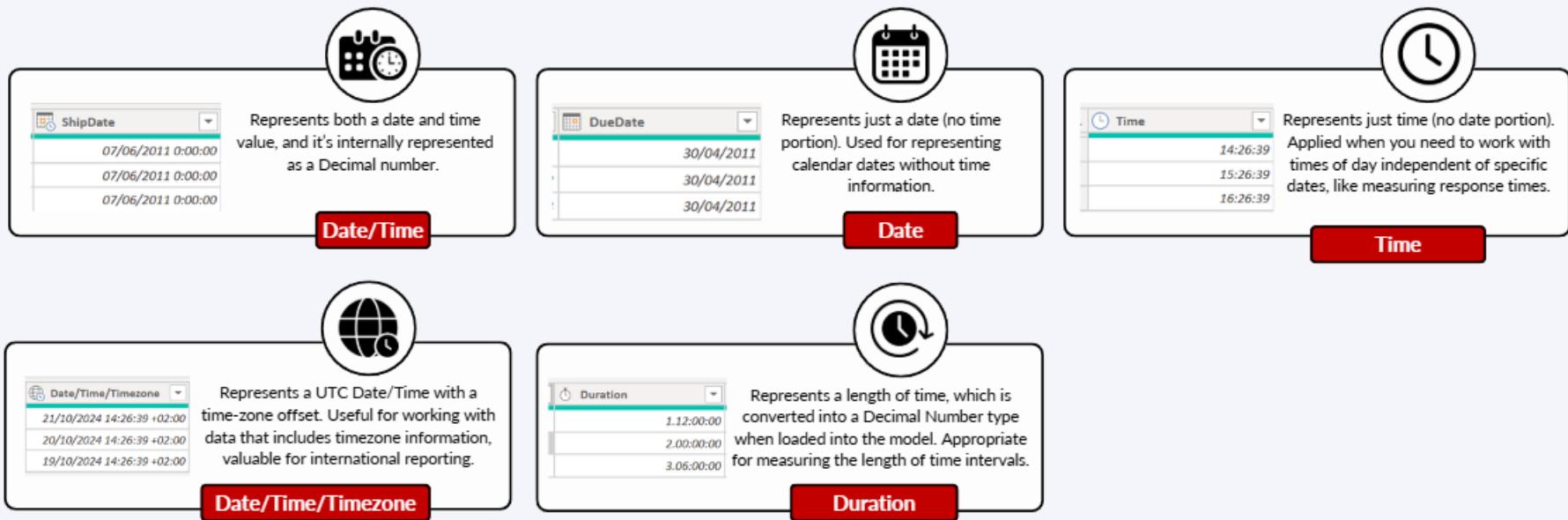


# Data types: Text





# Data types: Date/Time zone



# Data types: logical/boolean

The diagram features a light gray rectangular background with a thin black border. Inside, on the left, is a screenshot of a Microsoft Excel spreadsheet showing a column named "OnlineOrderFlag" with three rows of data, all of which are set to "FALSE". To the right of the spreadsheet is a circular icon containing two smaller icons: a speech bubble with an asterisk (\*) and a checkmark inside a square. A curved black line connects the bottom of the speech bubble icon to a red rectangular button at the bottom center of the slide, which contains the text "True/False" in white. To the right of the spreadsheet, there is explanatory text: "A Boolean value of either True or False. They're the foundation of conditional expressions and allow you to filter data based on specific conditions."

OnlineOrderFlag

FALSE

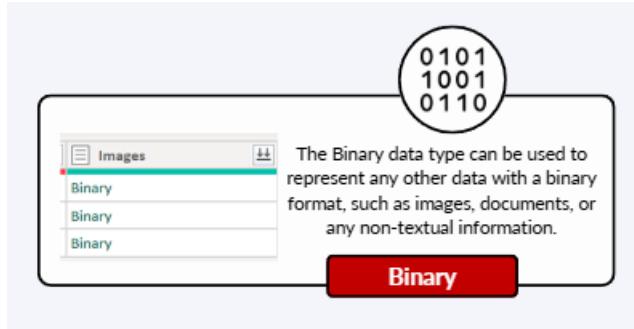
FALSE

FALSE

True/False

A Boolean value of either True or False.  
They're the foundation of conditional  
expressions and allow you to filter data  
based on specific conditions.

# Data types: Binary



# Data type vs Format

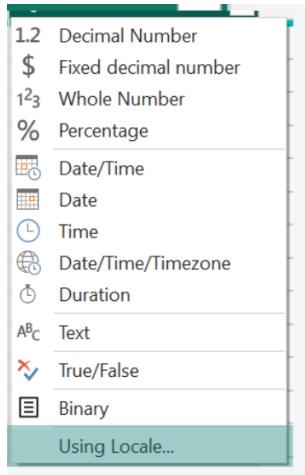
**Data Type:** Defines the kind of data (text, number, date) and determines how much memory is allocated to store values. It directly influences how data is stored and used in calculations.

**Format:** Controls how a value is displayed without affecting their underlying data structure or storage.

**\*Key Difference: Data types affect the backend (storage and memory usage), while formats only influence the frontend (visual presentation)**

# Effect of locale/culture

Different regions around the world follow varied data conventions



**Change Type with Locale**

Change the data type and select the locale of origin.

Data Type: Date

Locale: Spanish (Spain, International Sort)

Sample input values:  
29/03/2016  
martes, 29 de marzo de 2016  
29 de marzo  
marzo de 2016

**Change Type with Locale**

Change the data type and select the locale of origin.

Data Type: Date

Locale: English (United States)

Sample input values:  
3/29/2016  
Tuesday, March 29, 2016  
March 29  
March 2016

**Locale affects formatting conventions**



# Exercise 7

## Change Data Type



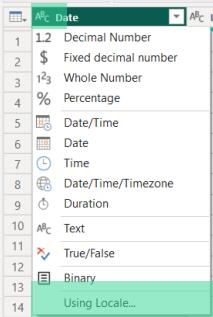
# Exercise 7: Change data types

**Goal:** Change data types

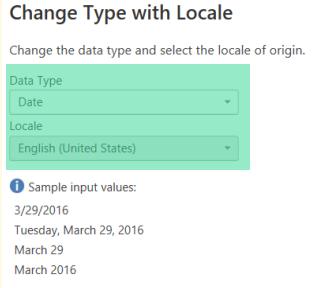
1 Go to Home > Transform Data

Select 'Budget' query.

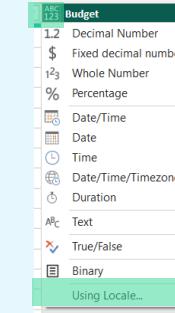
Select 'Date' column. Click on the data type and select *Using Local*



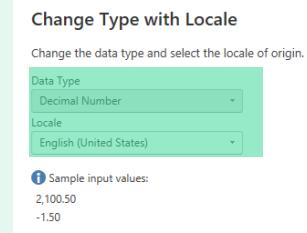
2 In 'Data type', select 'Date' and in 'Locale' select 'English (United States)'



3 Select 'Budget' column. Click on the data type and select *Using Locale*

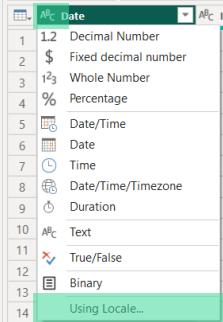


4 In 'Data type', select 'Decimal Number' and in 'Locale' select 'English (United States)'



5 Select 'Sales' query.

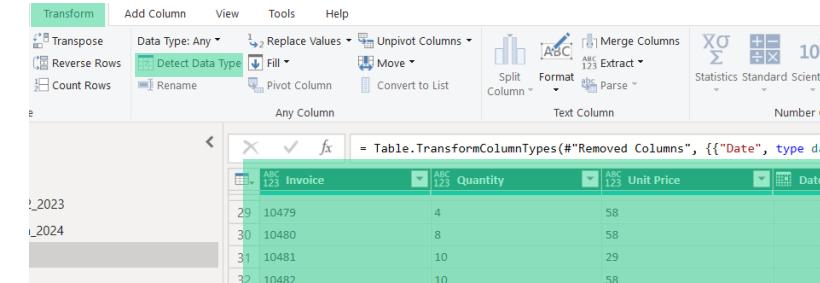
Select 'Date' column. Click on the data type and select *Using Local*



6 In 'Data type', select 'Date' and in 'Locale' select 'Spanish (Spain, International Sort)'



7 Since there are no more decimal values or dates in the table, we will **select all columns in the table by pressing Ctrl + A**, then go to **Transform / Detect Data Type**.





# Break time

# DEMO 08

- Overview of Abstract Types
- Utilizing Parameters





# Utilizing Parameters

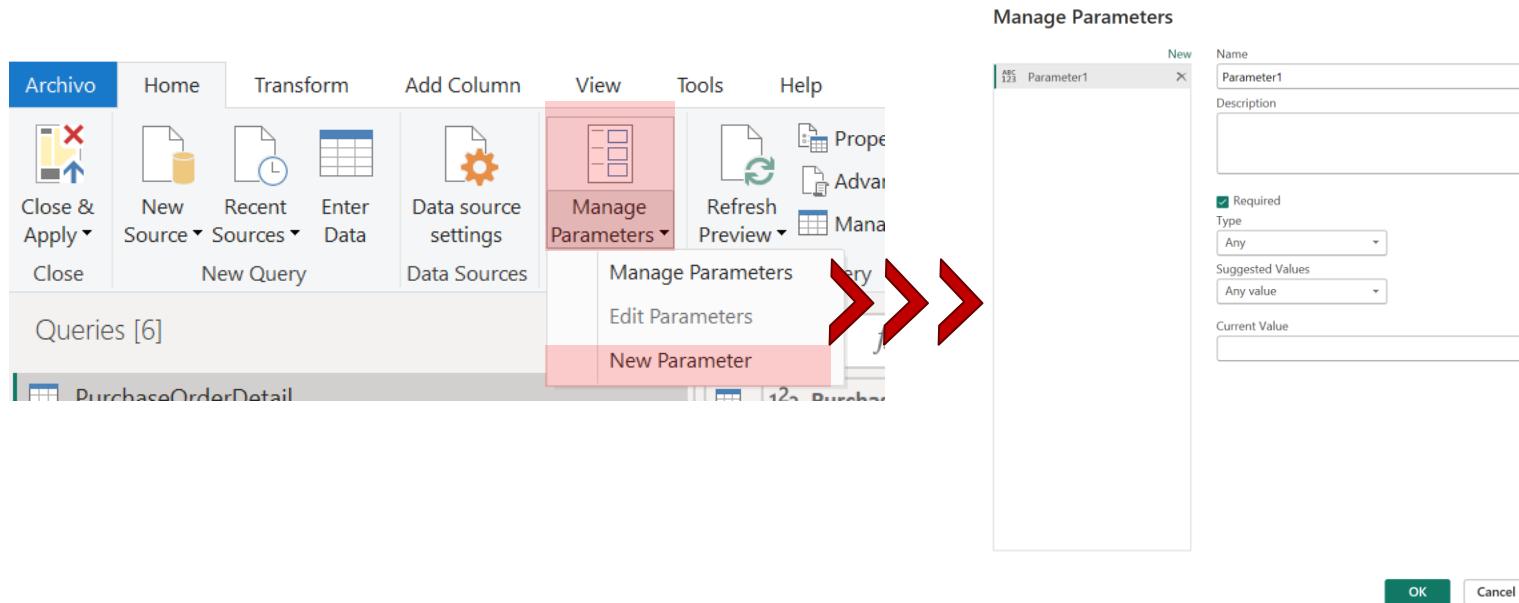
# Parameters

Parameters in Power Query play an important role in making queries dynamic.

- **They allow for the input of scalar values like dates, numbers, or text.**
- **This design enables you to make external modifications** to a single parameter, which then automatically propagates updates across multiple queries or steps within a query.

# Create Parameters in Power Query

- **Manage Parameters:**



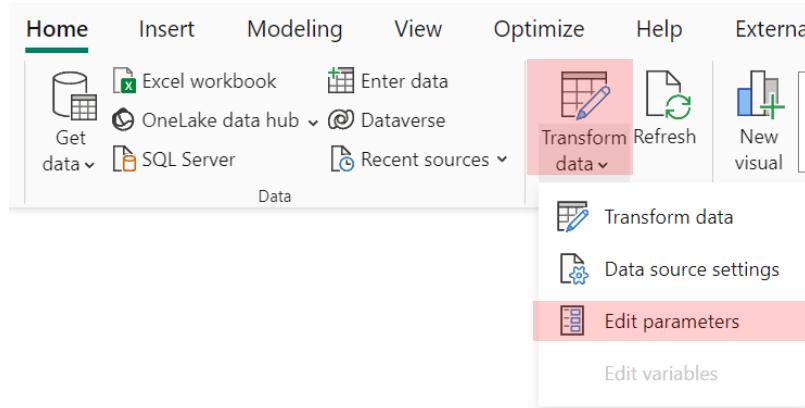
# Parameters

Parameters in Power Query play an important role in making queries dynamic.

- **They allow for the input of scalar values like dates, numbers, or text.**
- **This design enables you to make external modifications** to a single parameter, which then automatically propagates updates across multiple queries or steps within a query.

# Edit Parameters in Power BI Desktop or Power BI Service

## Power BI Desktop



## Power BI Service

The screenshot shows the 'Settings for SQL' page in the Power BI Service. At the top, there are links for 'View semantic model' and 'Refresh history'. Below that is a section for 'Semantic model description' with a text input field. Further down are sections for 'Gateway and cloud connections' and 'Data source credentials'. The 'Parameters' section is expanded, showing a parameter named 'Parameter1' with the value '123' in an input field. At the bottom of the page are 'Apply' and 'Discard' buttons.

Settings for SQL

[View semantic model](#)

[Refresh history](#)

Semantic model description

Describe the contents of this semantic model.

500 characters left

Apply Discard

Gateway and cloud connections

Data source credentials

Parameters

Parameter1  
123

Apply Discard



# Exercise 8

## Creating Parameters



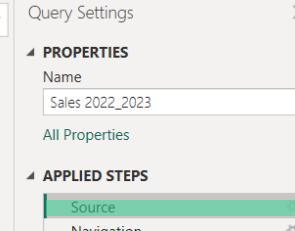
# Exercise 8: Creating Parameters

**Goal:** Learn to create parameters

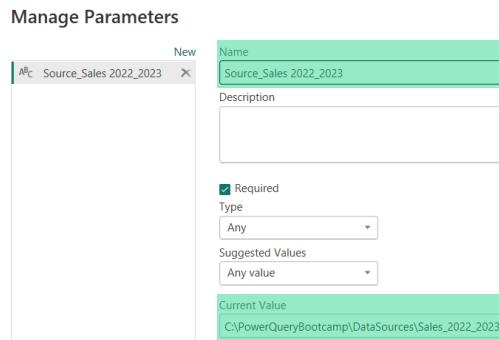
- 1 Go to the Source step and copy the URL from the 'Sales2022\_2023' table

The screenshot shows the Power Query Editor interface with the 'Source' step selected. The table contains the following data:

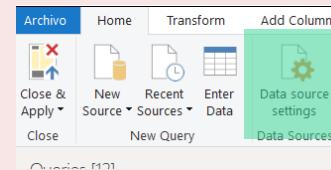
Item	Kind	Hidden
S_Sales 2022-2023	Sheet	FALSE
S_Product	Sheet	FALSE
S_Date	Sheet	FALSE
S_Customer	Sheet	FALSE
Sales2022_2023	Table	FALSE
Product	Table	FALSE



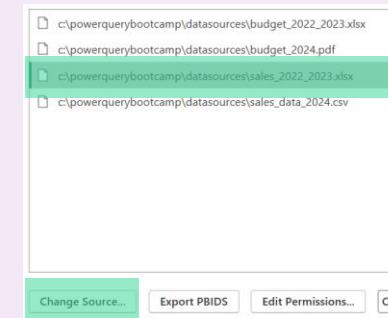
- 3 Name the parameter 'Source\_sales\_2022\_2023' and paste the URL you copied in step1



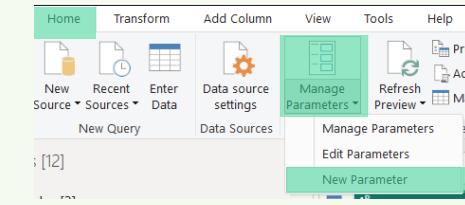
- 4 Click on Home > Data Source Setting



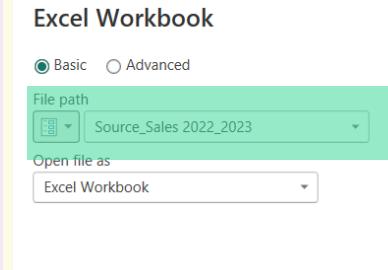
- 5 Select URL from 'sales\_2022\_2023'. Click 'Change Source'



- 2 Create a new parameter:  
Click on Home > Manage Parameter > New Parameter.



- 6 Select Parameter option



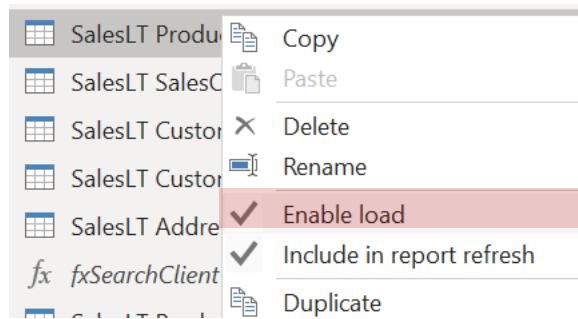
# DEMO 09

- Data Loading



# Minimize loaded queries

**Limit the number of queries loaded into memory** to optimize performance.





# Exercise 9

## Minimize Loaded Queries

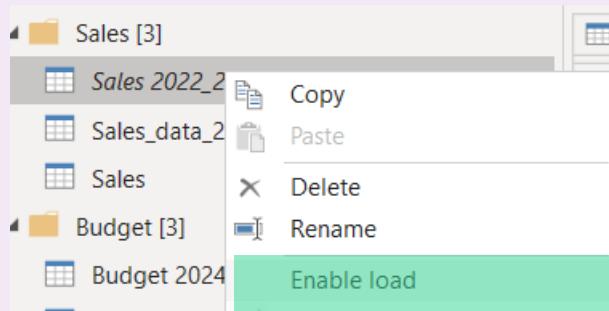


## Exercise 9: Minimize loaded queries

**Goal:** Learn to disable queries

- 1 Select query 'Sales\_data\_2024'

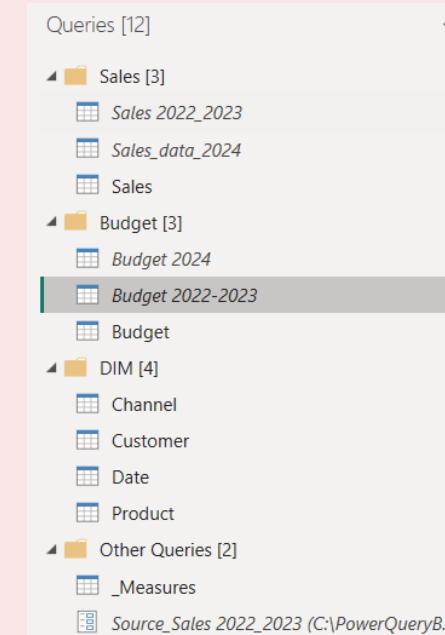
Right-click on the selected query, uncheck Enable Load



- 2

Repeat the process for queries:

- Sales\_data\_2024
- Budget 2024
- Budget 2022-2023



\*Notice that the query names are in italics, which means they will not be loaded into the final model

# DEMO 10

- Understanding Query Folding
- Types of Errors in Power Query





# Understanding Query Folding

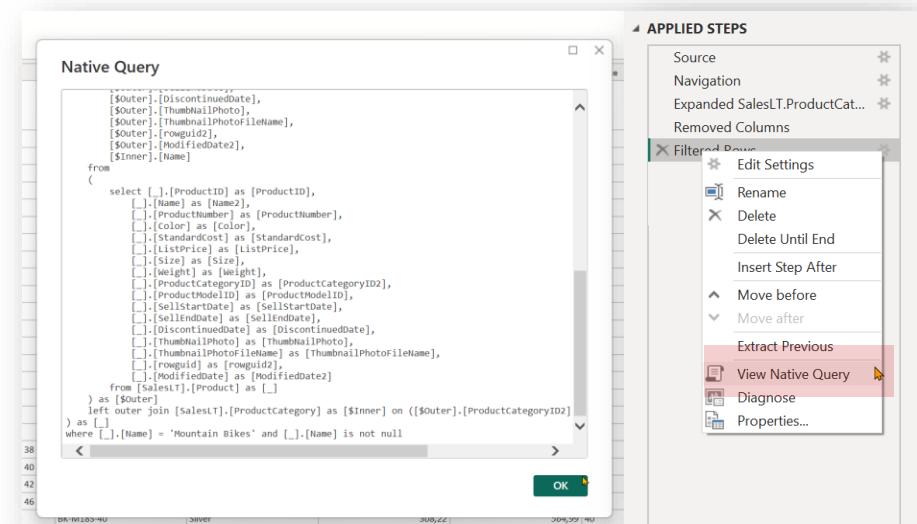
# Query Folding

## Understanding Query Folding

- Converts M code into a query language.
- Offloads complex processing to the data source.
- Generates a single SQL query executed directly on the data source for better performance!

## Benefits

- Enhances data refresh efficiency
- Essential for DirectQuery/Dual mode, ensuring queries fold properly
- Makes incremental refresh work as intended





# Query Folding

- Query folding is supported by sources like relational databases, OData, SSAS, and SharePoint lists, allowing transformations to push down to the data source.
- However, **not all file types support query folding**, such as Excel and text files.

Importantly, **not every data transformation will fold**:

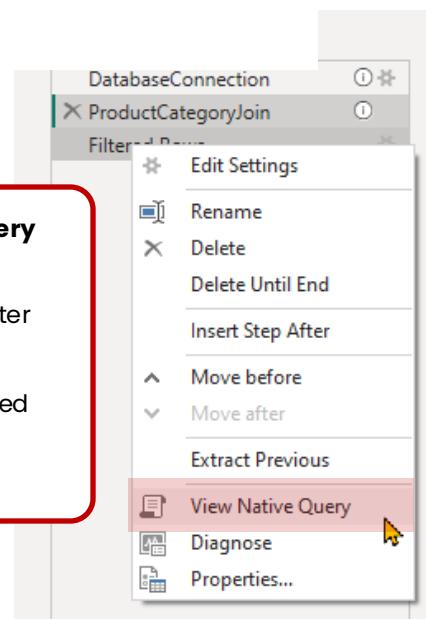
- |                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>✓ Removing or renaming columns</li><li>✓ Filtering rows</li><li>✓ Grouping and summarizing</li><li>✓ Merging or appending queries from the same data source</li><li>✓ Basic custom columns</li><li>✓ Pivoting and unpivoting</li></ul> | <ul style="list-style-type: none"><li>✗ Merging or appending queries across <b>different sources</b></li><li>✗ Complex custom columns</li><li>✗ Adding index columns</li><li>✗ Changing data types (this may vary depending on the transformation)</li></ul> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

# Query Folding – Native Query

- Use **Value.NativeQuery** function
  - The goal of this process is to execute the following SQL code, and to apply more transformations with Power Query that can be folded back to the source.

```
ProductCategoryJoin = Value.NativeQuery(DatabaseConnection, "
    SELECT
        p.ProductID,
        p.Name AS ProductName,
        p.ProductNumber,
        p.Color,
        p.StandardCost,
        p.ListPrice,
        p.Size,
        p.Weight,
        c.Name AS CategoryName,
        p.SellStartDate,
        p.SellEndDate,
        p.DiscontinuedDate
    FROM
        SalesLT.Product p
    JOIN
        SalesLT.ProductCategory c
    ON
        p.ProductCategoryID = c.ProductCategoryID"
    , null, [EnableFolding = true]),
```

This example uses the **Value.NativeQuery** function to run a SQL query and enable query folding with the optional parameter **[EnableFolding = true]**. This allows transformations like filtering to be pushed back to the database, optimizing performance.



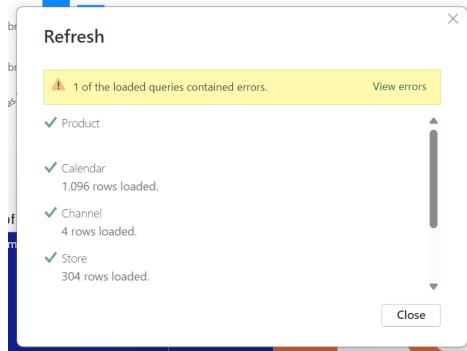


# Types of Errors in Power Query

# Types of Errors

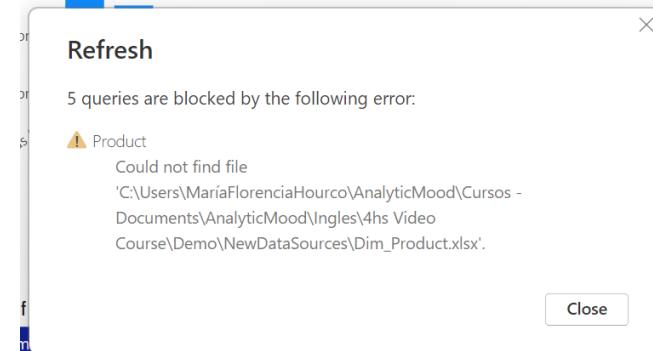
## Row level Errors

The model gets updated, and the cells with errors are replaced by blank values.



## Step Level Errors

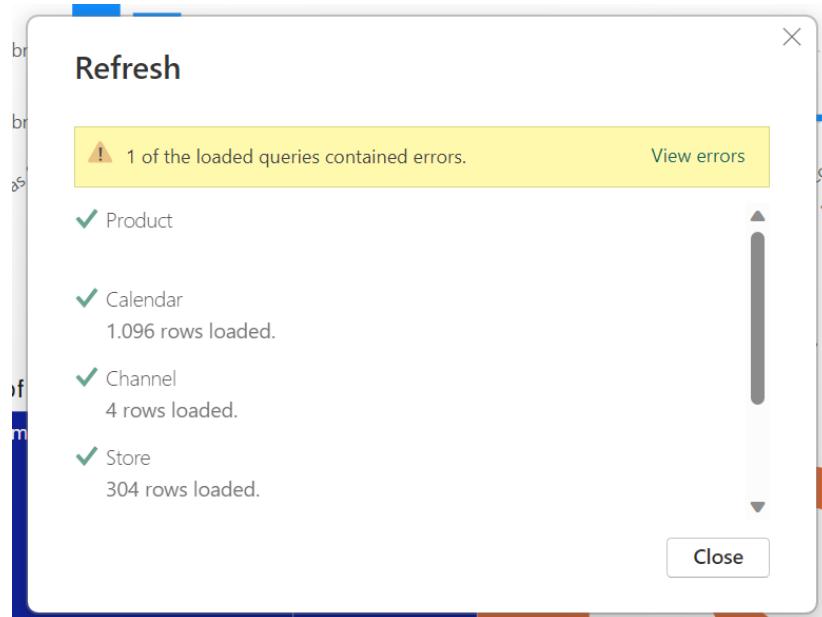
The model won't refresh until the errors are resolved.





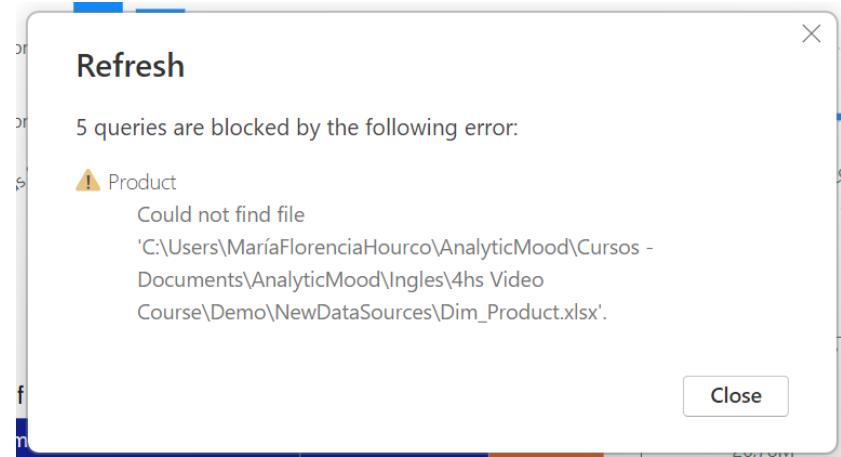
# Row-Level Errors

- The model gets updated, and the cells with errors are replaced by blank values.
- Row-level errors in Power Query can be caused by:
  - A cell value conversion issue to the specified data type in the column.
  - Incorrect formatting of one of the columns involved in creating an operational column.



# Step-Level Errors

- The model won't refresh until the errors are resolved.
- Step-level errors in Power Query can be caused by:
  - Can't find the source
  - The column of the table wasn't found
  - Invalid formulas
  - Duplicates or blanks in a table one to many





# Exercise 10

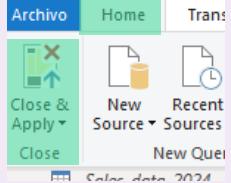
## Other Types of Errors



# Exercise 10: Other types of errors

**Goal:** Understand Different Types of Errors

- 1 Load the model into Power BI: Go to Home > Close and apply



\* If you are working with Exercise 10 downloaded

1. Go to the folder where your data sources are saved
2. Right-click on **Sales\_2022\_2023** > Copy as path
3. Open the query **Source\_Sales\_2022\_2023** in PowerQuery
4. Paste the copied path into Current Value
5. Remove the quotation marks (")

- 4 Change the data source to the new query we have created called 'Sales'

```
1 Sales = sum('Sales'[Total Sales])
```

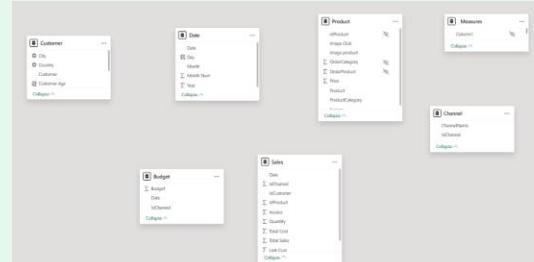
Repeat the process with:

- Costs
- Quantity

- 2 Once the model is loaded into Power BI, go to Measures and notice that all the measures that were created show an error

- 3 Select each of the measures that show an error

- 5 It's necessary to create relations again, because the sales query has changed.



Data Modeling for optimal structure & DAX for calculations

# Quiz

Let's do a final quiz!

## Question 1

### **What is the main purpose of Query Folding in Power BI?**

- A. It converts M code into a query language and offloads complex processing to the data source.
- B. It combines all the queries into a single M script for faster processing.
- C. It generates a single SQL query executed directly on the data source for better performance.
- D. It allows for manual editing of the SQL query to optimize performance.



## Question 2

**Which of the following transformations are supported by Query Folding in Power BI?**

- A. Removing or renaming columns
- B. Filtering rows, grouping and summarizing, merging or appending queries from the same data source
- C. Creating complex custom columns
- D. Using Excel or text files as data sources

## Question 3

**The model gets updated, and the cells with errors are replaced by blank values. Row-level errors in Power Query can be caused by:**

- A. A cell value conversion issue to the specified data type in the column.
- B. Incorrect formatting of one of the columns involved in creating an operational column.
- C. Both A and B.
- D. None of the above.

## Question 4

**In Power BI Desktop, an error message appears indicating that some values have been replaced by nulls. In the Power BI web service, no error is displayed when the dataset updates, even though there are row-level errors. What could be happening?**

- A. The Power BI web service cannot handle row-level errors.
- B. The Power BI web service ignores row-level errors and proceeds with the update.
- C. Power BI Desktop and Power BI Web always show the same errors.
- D. Row-level errors are automatically resolved in the Power BI web service.



# Recommendations for Next Steps

Part of the Power BI  
Bootcamp Series

## Advanced Learning Path:

- 1) Power BI Bootcamp: Building the Foundations for Power BI
- 2) Power Query: Data Connections and Transformations
- 3) Data Modeling for optimal structure & DAX for calculations
- 4) Visualizations for impactful reporting & Power BI Service

Aug 2, 9, 16 & 23  
2pm-6pm CET

This Bootcamp

COMING SOON

COMING SOON





# Stay Connected

Follow us on LinkedIn:



Nicolás Lagreste Zucchini



María Florencia Hourcouripé

**Let's Collaborate:** Interested in a personalized mentorship or consulting session? Contact us:

-  [nico@analyticmood.com](mailto:nico@analyticmood.com)
-  [flor@analyticmood.com](mailto:flor@analyticmood.com)

**Visit Our Website:**

- [www.analyticmood.com](http://www.analyticmood.com) for resources, upcoming events, and blog articles.

**Stay Tuned for More: Upcoming courses and events to take your skills to the next level!**

analytic mood



# Thank you!

The O'Reilly logo is centered on a background that transitions from red at the top left to orange and yellow at the bottom right. The logo consists of the word "O'REILLY" in a bold, white, sans-serif font. A registered trademark symbol (®) is positioned at the top right of the letter "Y".

O'REILLY®