UP & RUNNING WITH

MICROSOFT POWER BI DESKTOP

★★★★★ *With Best-Selling Instructors **Chris Dutton** & **Aaron Parry***

MAVEN® ANALYTICS

COURSE STRUCTURE

This is a **project-based course** designed for students looking for a practical, hands-on, and highly engaging approach to learning Power BI Desktop for business intelligence

Course resources include:

- ★ **Downloadable PDF eBook** (*200+ pages*) containing all course slides, assignments and reference materials
- ★ **Quizzes** and **Assignments** to reinforce key concepts and simulate real-world scenarios, with step-by-step solution videos
- ★ Complete **Bonus Project** to test your abilities and apply the skills developed throughout the course to a brand-new data set

COURSE OUTLINE

1	**Introducing Power BI Desktop**	*Installing Power BI Desktop, exploring the Power BI workflow, comparing Power BI vs. Excel, etc.*
2	**Connecting & Shaping Data**	*Connecting to data, shaping & transforming tables, using profiling tools, editing, merging & appending queries, etc.*
3	**Creating a Data Model**	*Building relational models, creating table relationships, understanding cardinality and filter flow, etc.*
4	**Calculating Measures with DAX**	*Understanding DAX syntax, adding calculated columns and measures, writing common formulas and functions, etc.*
5	**Visualizing Data with Dashboards**	*Inserting charts and visuals, customizing formats, editing interactions, applying filters and bookmarks, etc.*
6	**Optimizing Power BI Performance**	*Exploring common Power BI optimization tools within the Optimize and External tools menus*

COURSE PROJECT

THE SITUATION

You’ve just been hired as a Business Intelligence Analyst by **AdventureWorks***, a global manufacturing company that produces cycling equipment and accessories

THE BRIEF

The management team needs a way to **track KPIs** (*sales*, *revenue*, *profit*, *returns*), **compare regional performance**, **analyze product-level trends**, and **identify high-value customers**.

All you’ve been given is a **folder of raw csv files**, which contain information about transactions, returns, products, customers, and sales territories.

THE OBJECTIVE

Use Power BI Desktop to:

- Connect and transform the raw data
- Build a relational data model
- Create calculated columns and measures with DAX
- Design an interactive dashboard to visualize the data

**This data is provided by Microsoft for informational purposes only as an aid to illustrate a concept. These samples are provided “as is” without warranty of any kind. The example companies, organizations, products, domain names, e-mail addresses, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, person, place, or event is intended or should be inferred.*

SETTING EXPECTATIONS

1. What you see on your screen **may not always match mine**
   - *Power BI Desktop features are updated frequently, with new versions released each month*
   - ***NOTE:** Power BI is currently only compatible with PC/Windows (not available for Mac)*

2. This course is designed to help you build **foundational skills**
   - *Our goal is to help you build a deep foundational understanding of the Power BI desktop workflow; some topics may be simplified, and we won't cover some advanced tools (M code, advanced DAX, R/Python visuals, etc.)*

3. This is a **hands-on** and **project-based** learning experience
   - *You will get the most value out of this course if you follow along closely with the demos and assignments; we'll be working through the entire BI workflow to create a professional-quality dashboard from scratch*

4. We will not cover **Power BI Service** as part of this course
   - *This course focuses on Power BI Desktop specifically; online sharing and collaboration features (app.powerbi.com) require a separate account and are covered in-depth in a separate course*

INTRODUCING POWER BI

MEET POWER BI

In this section we'll **introduce Power BI Desktop**, review the download and installation process, adjust default settings, and explore the Power BI interface and workflow

TOPICS WE'LL COVER:

- Introducing Power BI
- Power BI vs. Excel
- Installation Options
- Adjusting Settings
- Interface & Workflow
- Helpful Resources

GOALS FOR THIS SECTION:

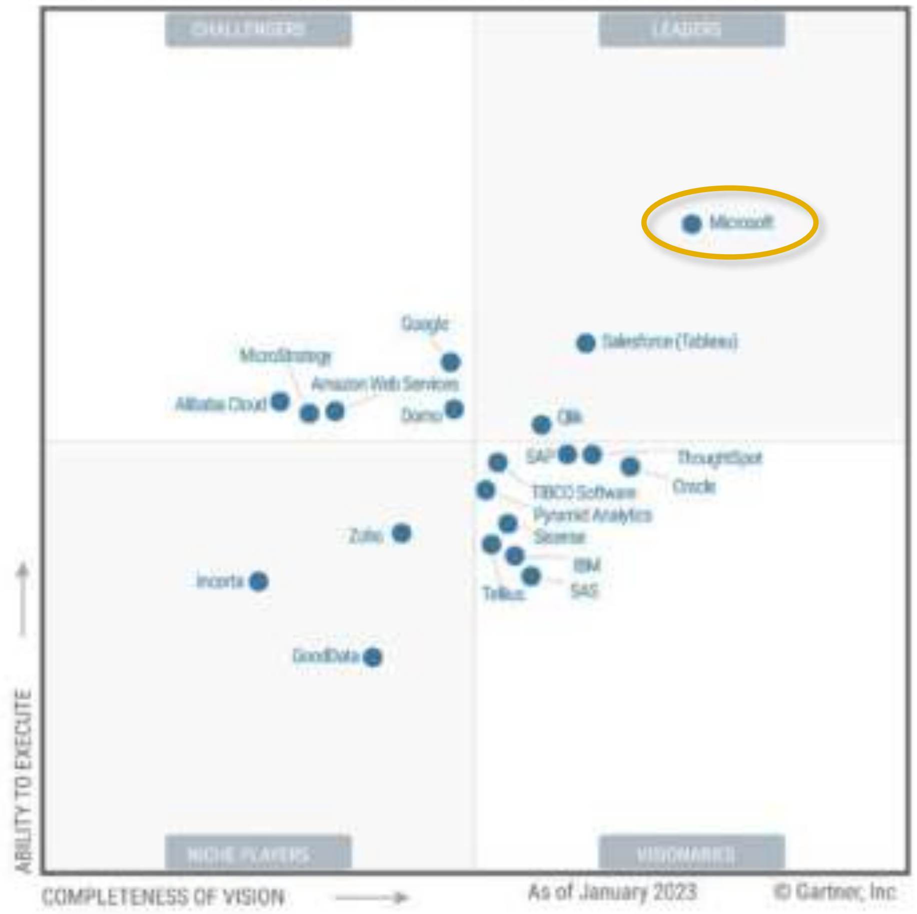
- Download and install Power BI Desktop, and adjust the settings for our course project
- Understand the role that Power BI plays within the broader Microsoft ecosystem
- Explore core components of the Power BI Desktop interface
- Review the business intelligence workflow that we'll follow as we build our course project

MEET POWER BI

Microsoft Power BI is a self-service business intelligence platform, which includes both desktop and web-based applications for connecting, modeling, and visualizing data

Learn more at **powerbi.microsoft.com**

Figure 1: Magic Quadrant for Analytics and Business Intelligence Platforms



WHY POWER BI?

✓ Connect, transform and load millions of rows of data

- *Access data from virtually anywhere (database tables, flat files, web, cloud services, folders, etc.), and create fully automated workflows to extract, transform and load data for analysis*

✓ Build relational models to blend data from multiple sources

- *Create table relationships to analyze holistic performance across an entire relational data model*

✓ Define complex calculations using Data Analysis Expressions (DAX)

- *Enhance datasets and enable advanced analytics with powerful and portable DAX expressions*

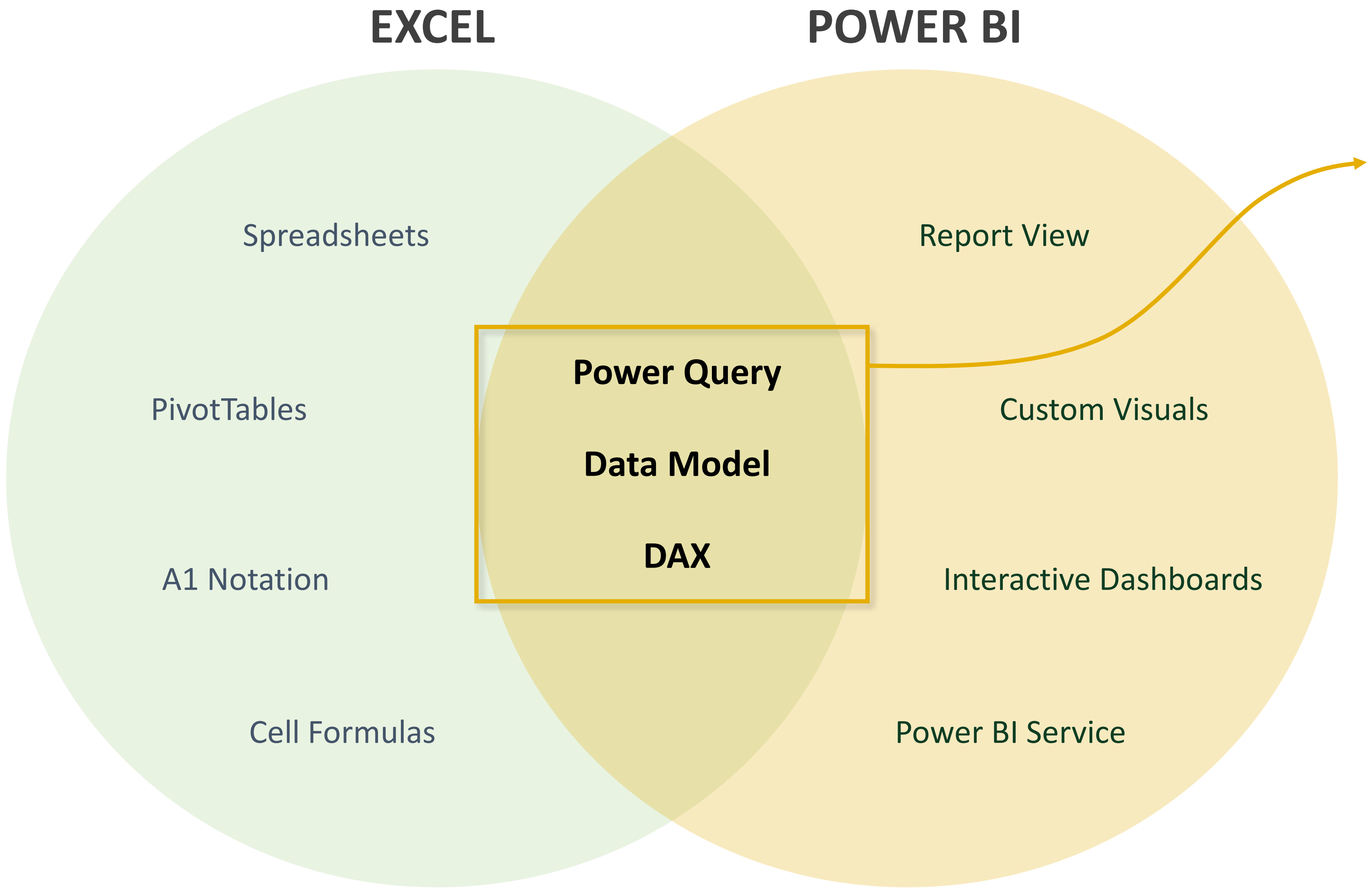
✓ Bring data to life with interactive reports and dashboards

- *Build professional-quality reports and dashboards with best-in-class visualization tools*

✓ Develop a versatile, in-demand skill set

- *Power BI is the industry leader in self-service BI, and the skills you build in this course will be highly transferrable*

EXCEL VS. POWER BI



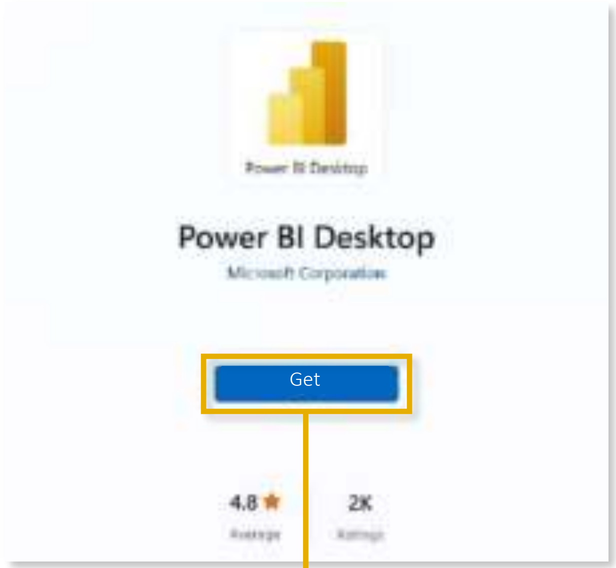
Excel and Power BI are built on top of the **same analytics engines**

- Power BI takes the same data transformation and modeling capabilities and adds **powerful visualization and publishing tools**
- Transitioning is easy; you can import an **entire data model** directly from Excel!

INSTALLING POWER BI DESKTOP

1) Download from **Microsoft store**

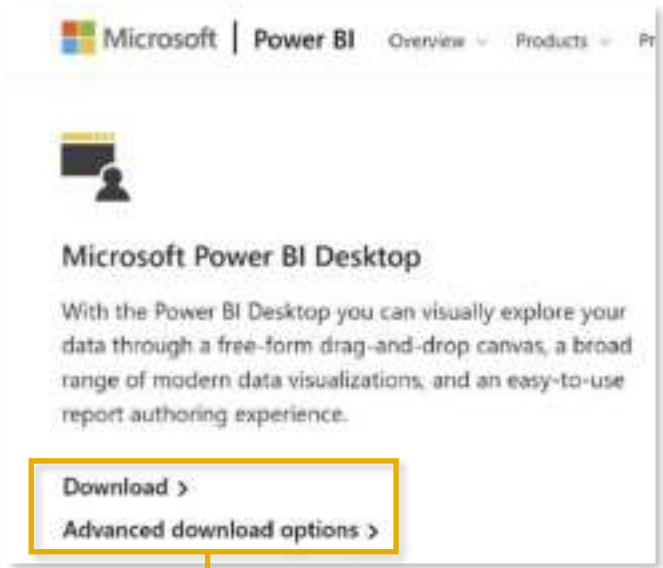
apps.microsoft.com



- Windows handles **automatic updates**
- Updates only elements that have been changed
- Doesn't require administrator access

2) Download **manually from web**

powerbi.microsoft.com/downloads



Microsoft | Power BI Overview Products

Microsoft Power BI Desktop

With the Power BI Desktop you can visually explore your data through a free-form drag-and-drop canvas, a broad range of modern data visualizations, and an easy-to-use report authoring experience.

Download >

Advanced download options >

- **No automatic updates** (allows version control)
- Downloads an executable installation file
- Administrator access may be required

3) Install as part of **Microsoft 365**

microsoft.com/en-us/microsoft-365

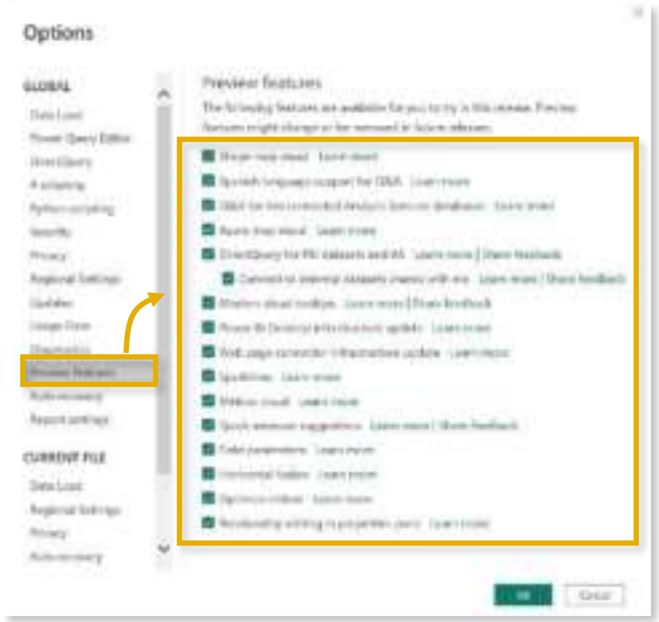
- Power BI Desktop is included as part of select enterprise Office/Microsoft 365 subscriptions
- If your company uses a compatible version of Microsoft 365, talk to an admin about getting access to Power BI

HEY THIS IS IMPORTANT!

You do **NOT** need to register for a Power BI Pro account to access Power BI Desktop

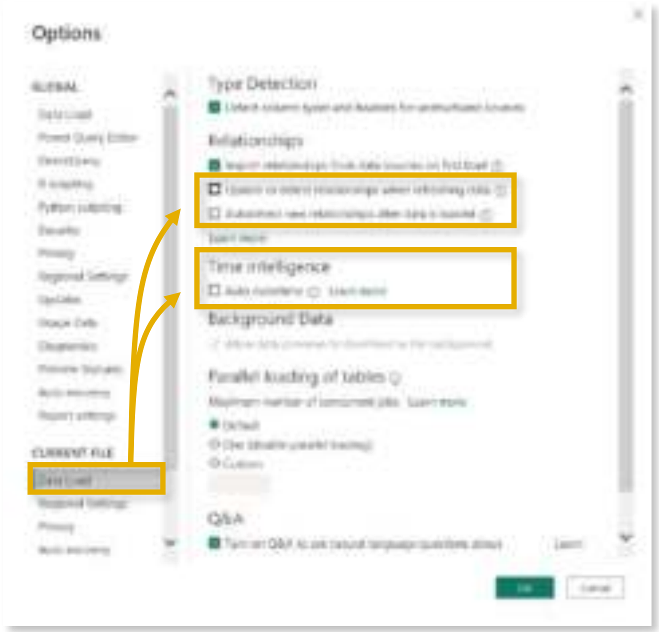
POWER BI SETTINGS

Global > **Preview Features**



Select **all available preview features** by default (these change with each monthly release)

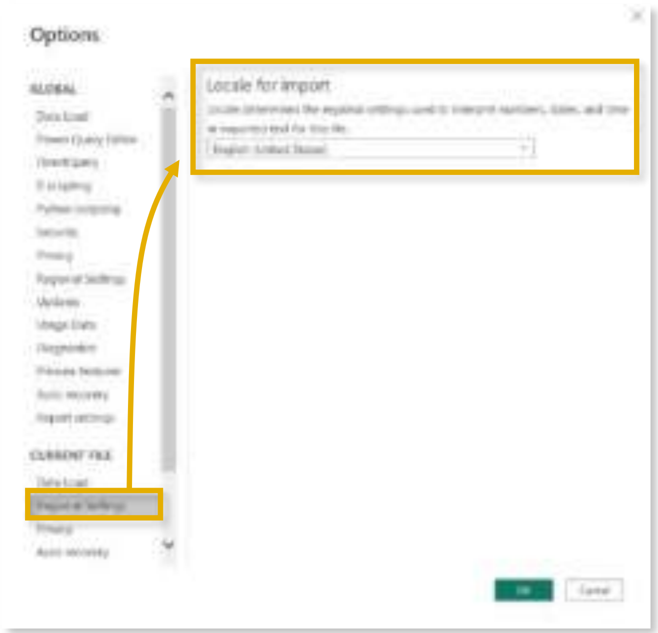
Current File > **Data Load**



Make sure the following options are **NOT selected**:

- ***Update or delete relationships when refreshing data***
- ***Autodetect new relationships after data is loaded***
- ***Time Intelligence > Auto date/time***

Current File > **Regional Settings**



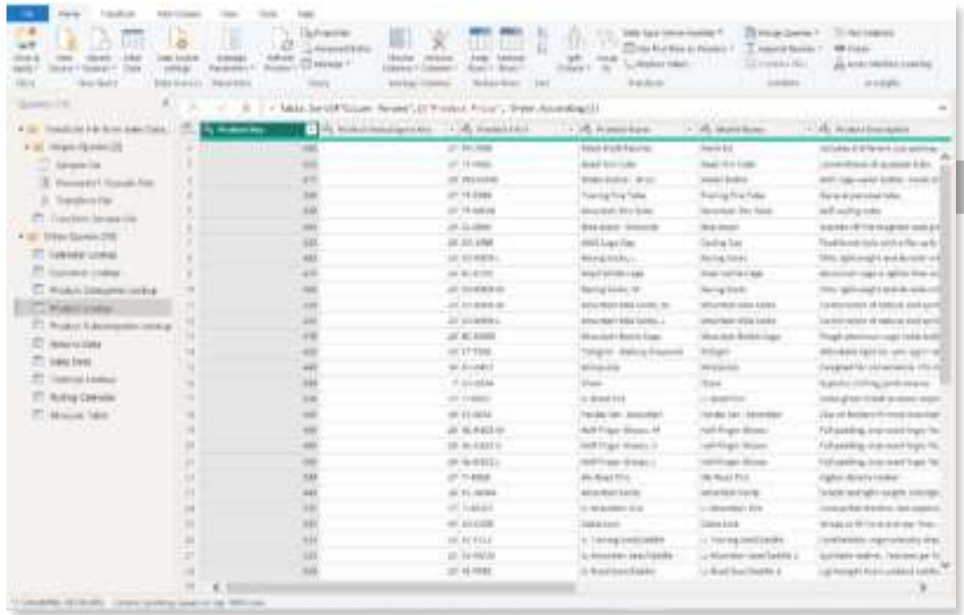
Select ***"English (United States)"*** from the dropdown menu (this will align with the data in course project files)

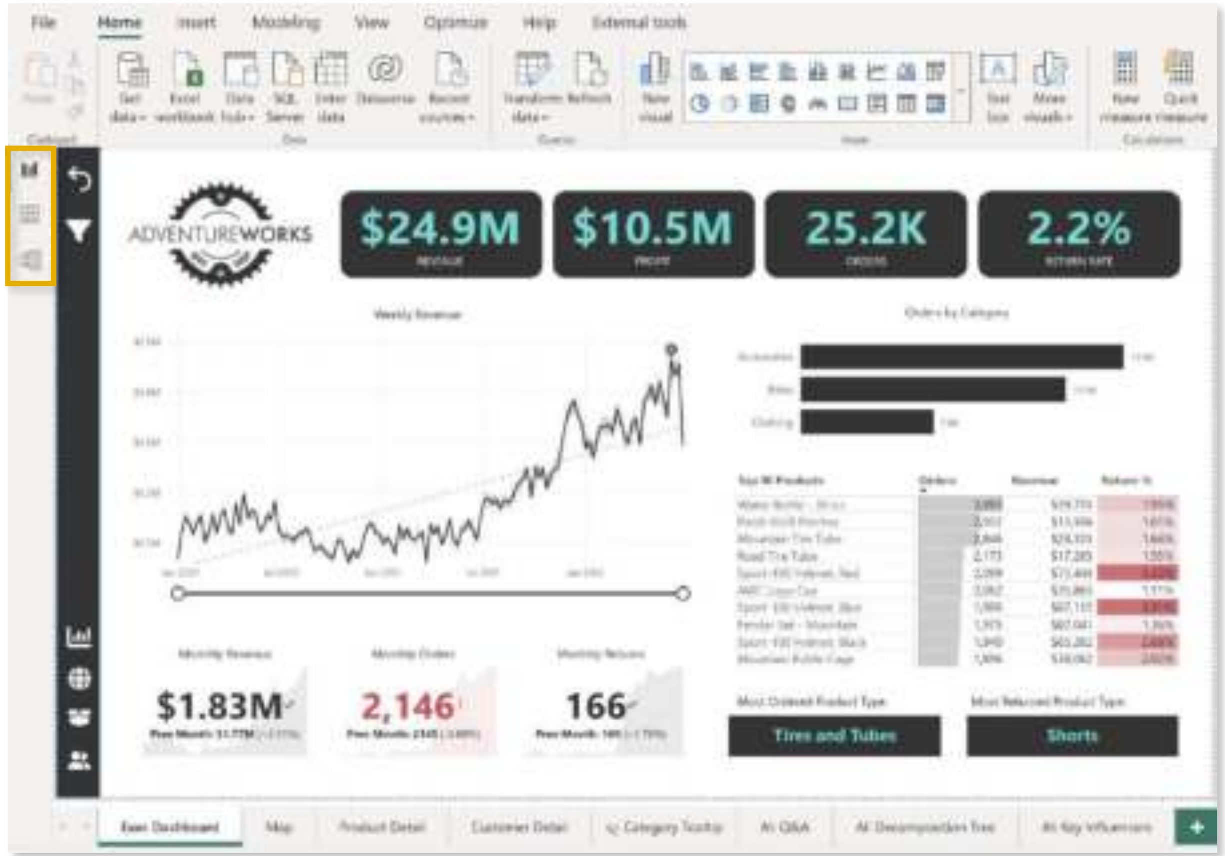
HEY THIS IS IMPORTANT!

Options under **CURRENT FILE** need to be adjusted **every time you open a new Power BI workbook** (these settings do not persist across new .pbix files)

POWER BI WORKFLOW

Raw data is extracted and transformed in the **Power Query editor**, then loaded to the Power BI “front-end”





Power Query Editor

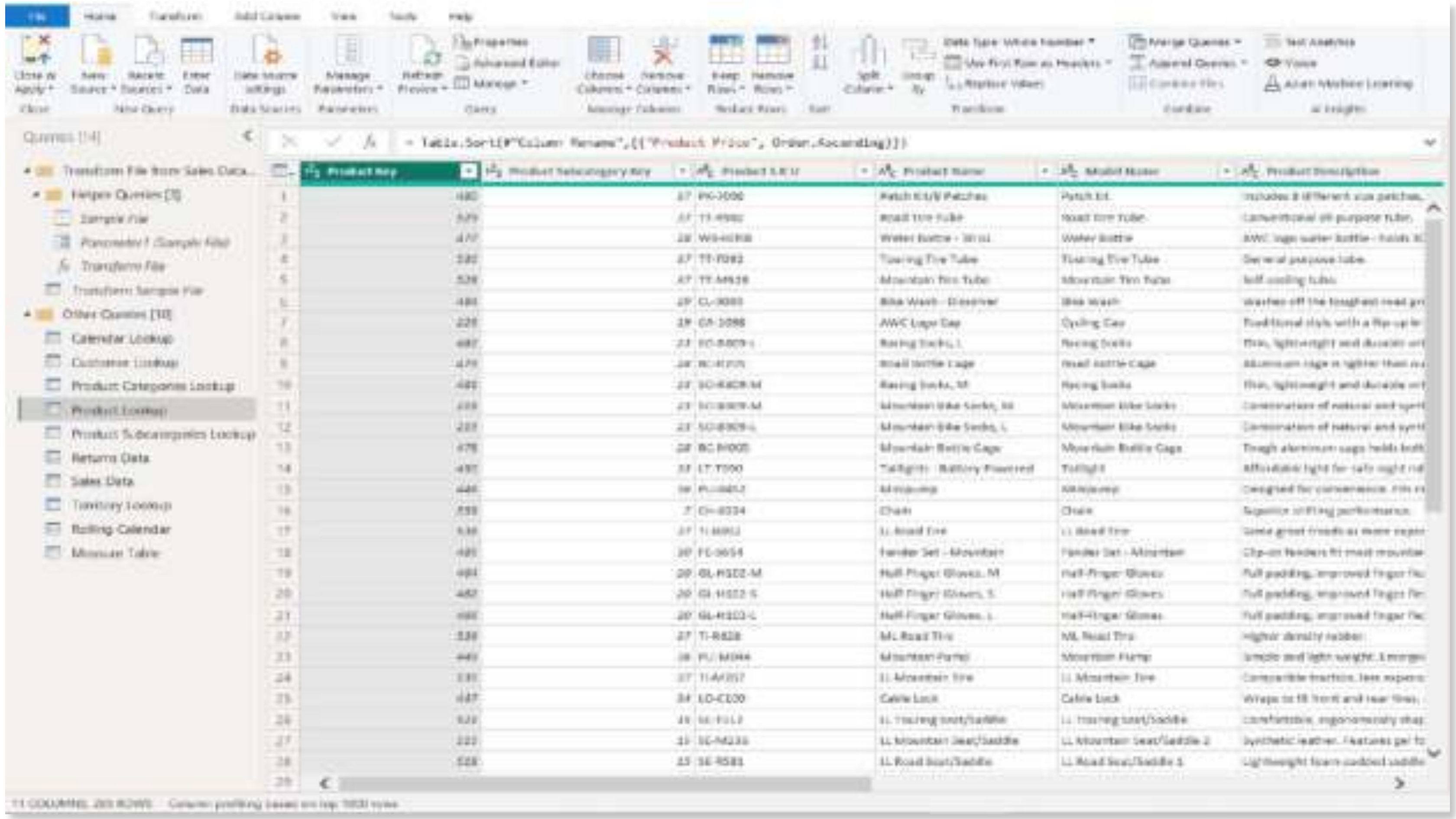
Power BI “Back-End”

Model View **Data View** **Report View**

Power BI “Front-End”

POWER BI WORKFLOW

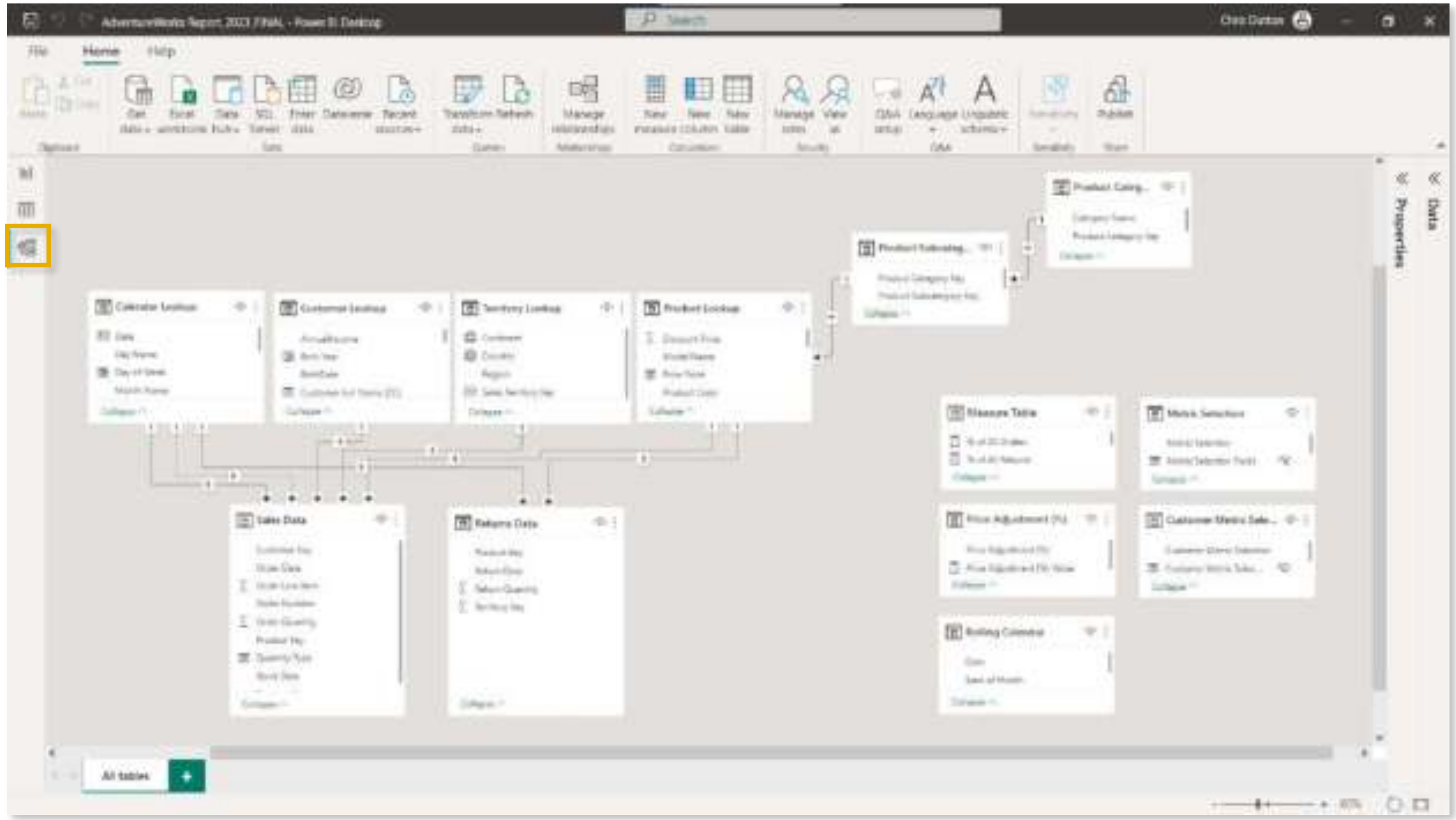
Power Query Editor Model View Data View Report View



1 Data is loaded & transformed in the **Power Query Editor**

POWER BI WORKFLOW

Power Query Editor **Model View** Data View Report View

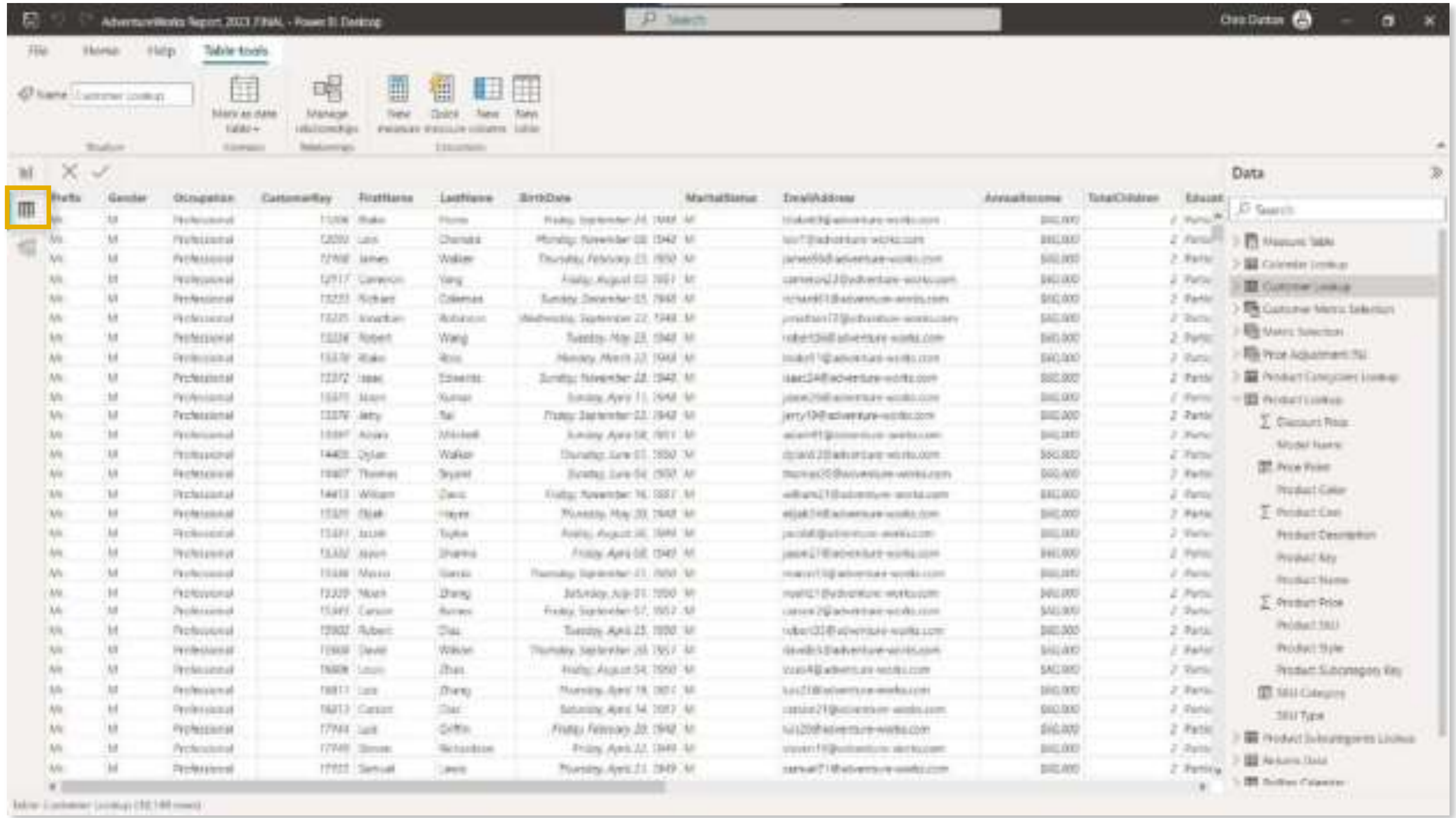


1 Data is loaded & transformed in the **Power Query Editor**

2 Data models are configured in the **Model View**

POWER BI WORKFLOW

Power Query Editor | Model View | **Data View** | Report View



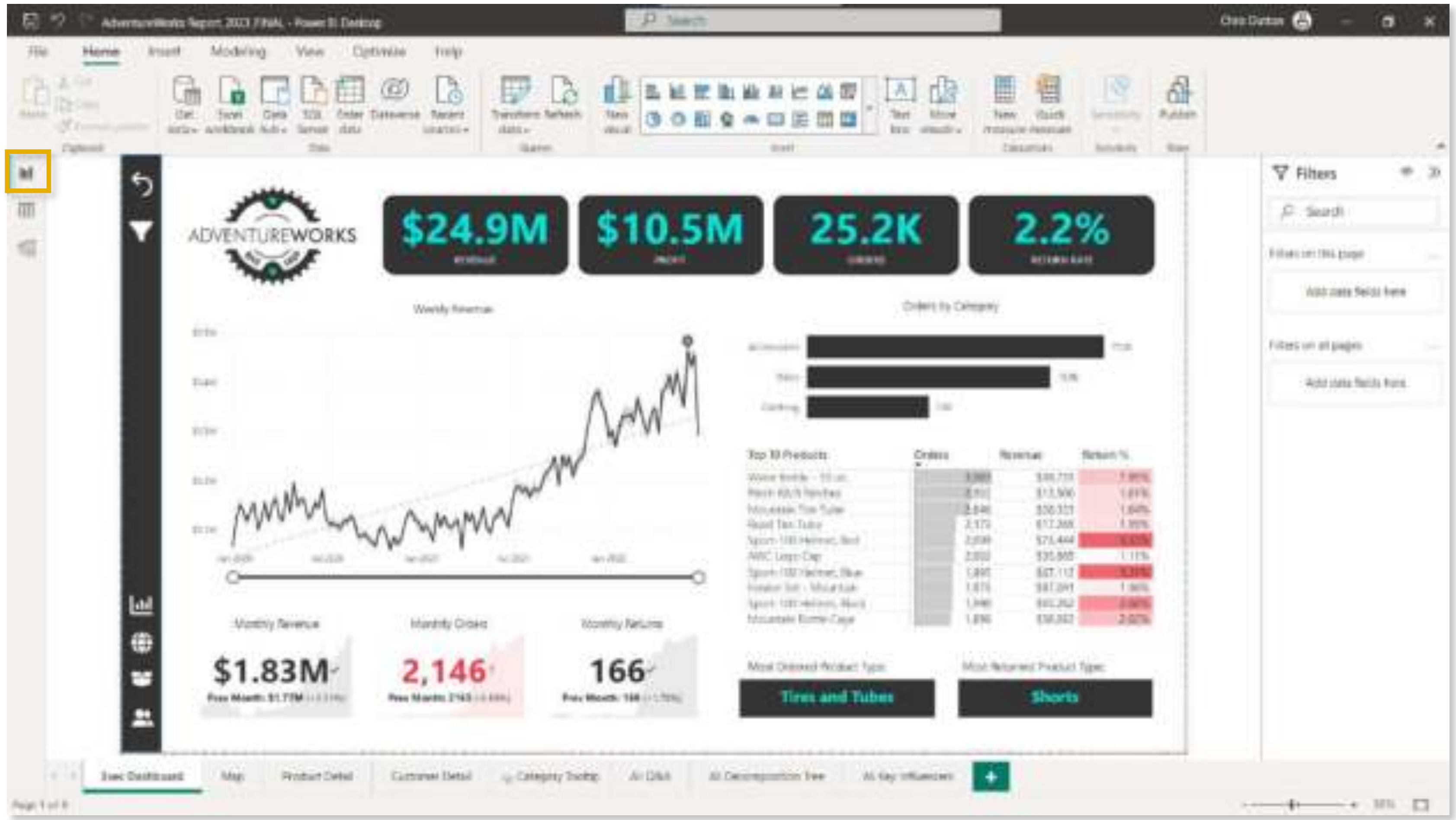
1 Data is loaded & transformed in the **Power Query Editor**

2 Data models are configured in the **Model View**

3 Table features & calculations are added in the **Data View**

POWER BI WORKFLOW

Power Query Editor Model View Data View **Report View**



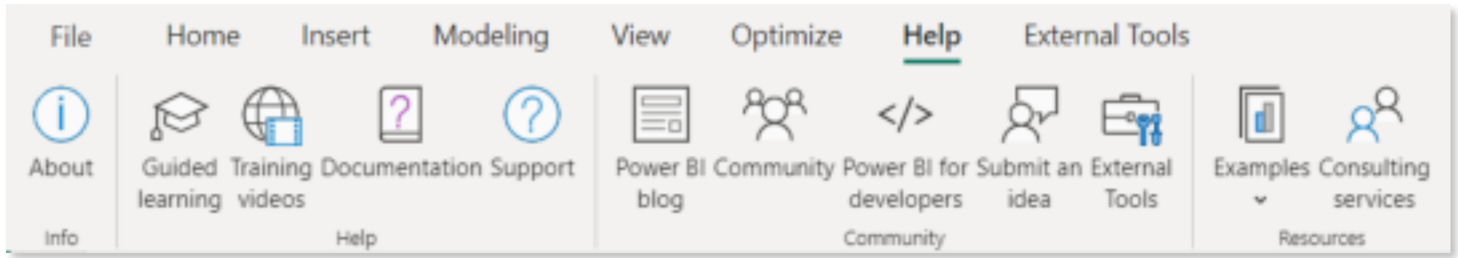
1 Data is loaded & transformed in the **Power Query Editor**

2 Data models are configured in the **Model View**

3 Table features & calculations are added in the **Data View**

4 Visuals & reports are designed in the **Report View**

HELPFUL RESOURCES



The **Help** tab includes documentation, training videos, sample files, templates, and links to support blogs and communities

The **Microsoft Power BI blog** (*powerbi.microsoft.com/blog*) publishes monthly summaries to showcase new features

The **Microsoft Power BI YouTube Channel** publishes demos, feature summaries, and advanced tutorials (check out “**Guy in a Cube**” too!)

Power BI User Groups (Power BIUG) are communities of users, which include both local meet-ups and helpful online forums (*pbiusergroup.com*)

MONTHLY UPDATES

Power BI is updated monthly, so you may notice ongoing changes to settings, options, tools, etc. Reference the links below to stay up-to-date on product updates and new feature releases:

Power BI Desktop
https://docs.microsoft.com/en-us/power-bi/fundamentals/desktop-latest-update

Power BI Service
https://docs.microsoft.com/en-us/power-bi/fundamentals/service-whats-new

Power Platform
https://learn.microsoft.com/en-us/dynamics365/release-plans/

CONNECTING & SHAPING DATA

CONNECTING & SHAPING DATA

In this section we'll connect to source files and cover some of the most common techniques for **extracting**, **cleaning**, and **shaping data** to prepare it for modeling and analysis

TOPICS WE'LL COVER:

- Intro to Power Query
- Data Connectors
- The Query Editor
- Connection Modes
- Data QA & Profiling
- Table Transformations
- Calendar Tools
- Combining Queries

GOALS FOR THIS SECTION:

- Explore Power BI's query editor and understand the role that Power Query plays in the larger BI workflow
- Introduce different types of connectors and connectivity modes available for getting data into Power BI
- Review tools for checking data quality and key profiling metrics like column distribution, empty values, errors and outliers
- Transform tables using text, numerical and date/time tools, pivot and group records, and create new conditional columns
- Practice combining, modifying and refreshing queries

FRONT-END VS. BACK-END

Power BI Desktop essentially has two distinct environments: a **front-end** and a **back-end**

- The **front-end** includes the **Data**, **Model** & **Report** views, where most of the modeling, analysis and visualization takes place
- The **back-end** includes the **Power Query Editor**, where raw data is extracted, transformed, and loaded to the front-end (ETL)

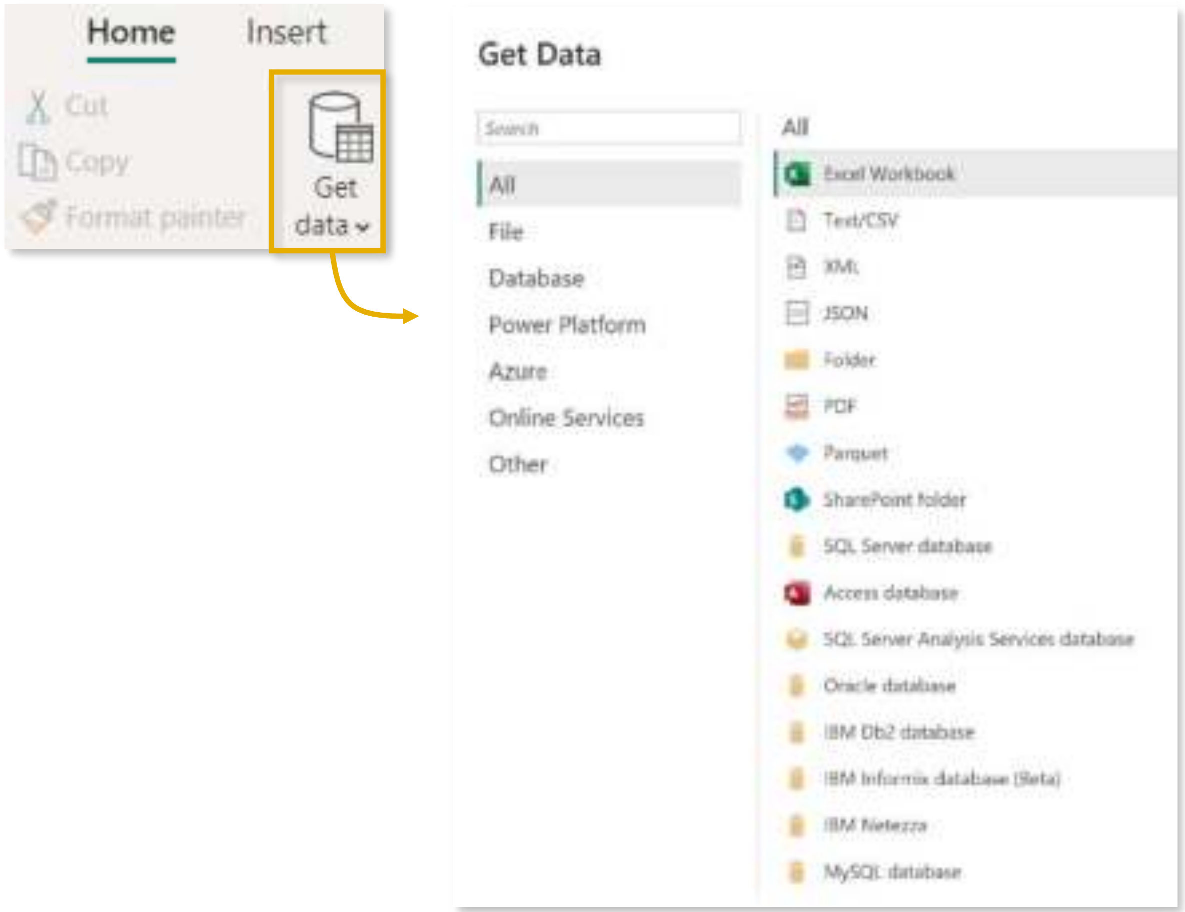
BACK-END

- **Connect** & **extract** data using pre-built connectors
- **Profile** & **QA** the data to explore, clean and prepare it for modeling and analysis
- **Transform** & **shape** tables to add new features, modify values, group records, or sort and filter columns
- **Merge** or **append** queries to join and combine them prior to loading to the front-end
- Perform **advanced transformations** using custom M code (out of scope for this course)

FRONT-END

- **Build data models** by creating table relationships between primary and foreign keys
- **Add calculated measures** & **columns** using Data Analysis Expressions (DAX)
- **Design reports** to visualize the data and create interactive, dynamic dashboards
- **Publish** & **share** your Power BI workbooks using Power BI Service (cloud application)

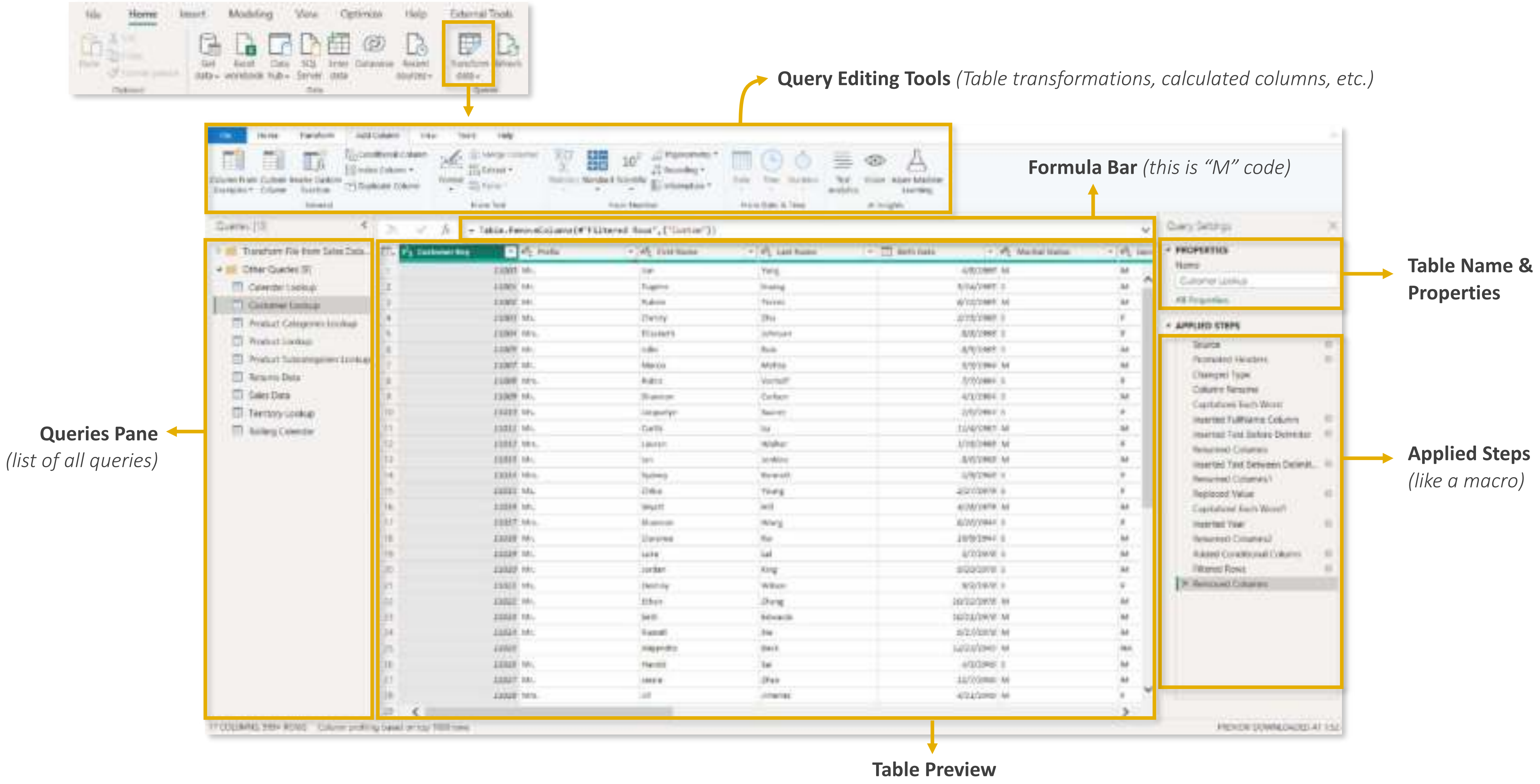
TYPES OF DATA CONNECTORS



Power BI can connect to virtually **any** type of source data, including (*but not limited to*):

- **Flat files & Folders** (*csv, text, xlsx, etc.*)
- **Databases** (*SQL, Access, Oracle, IBM, etc.*)
- **Power Platform** (*Datasets, Datamarts, Dataflows, Dataverse, etc.*)
- **Azure** (*Azure SQL, Analysis Services, Databricks, etc.*)
- **Online Services** (*SharePoint, GitHub, Dynamics 365, Google Analytics, Salesforce, Power BI Service, etc.*)
- **Other** (*Web feeds, R scripts, Spark, Hadoop, etc.*)

POWER QUERY EDITOR

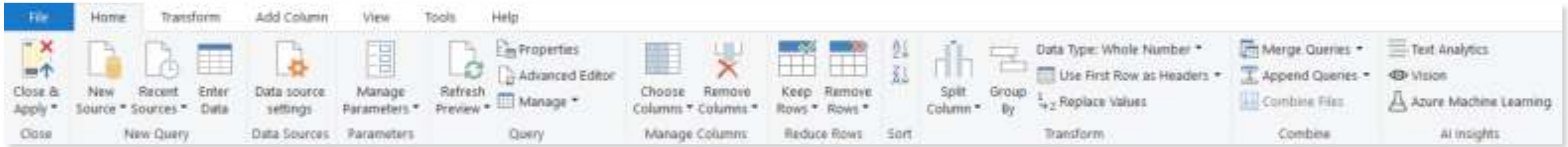

Query Editing Tools *(Table transformations, calculated columns, etc.)*
Formula Bar *(this is "M" code)*
Table Name & Properties
Queries Pane *(list of all queries)*
Applied Steps *(like a macro)*
Table Preview

```
= Table.RemoveColumns(#"Filtered Rows",{"Custom"})
```

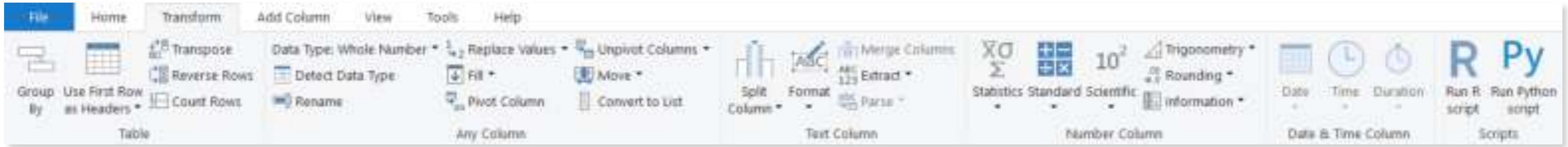
In older versions of Power BI, the Transform Data option may be named "Edit Queries"

QUERY EDITING TOOLS

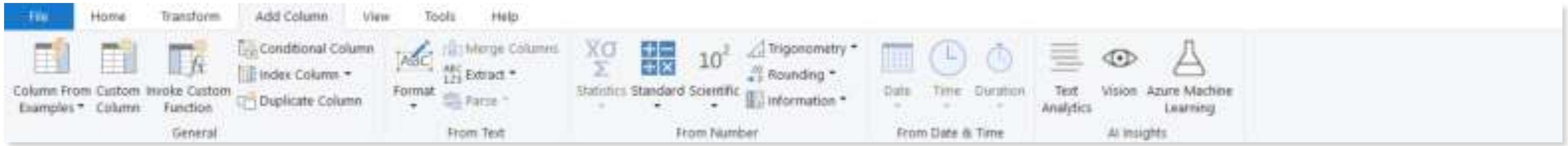
*The **HOME** tab includes **general settings** and **common table transformation tools***



*The **TRANSFORM** tab includes tools to **modify existing columns** (splitting/grouping, transposing, extracting text, etc.)*



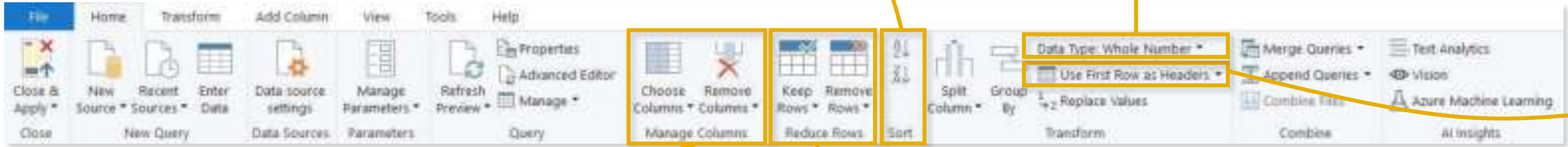
*The **ADD COLUMN** tools **create new columns** (based on conditional rules, text operations, calculations, dates, etc.)*



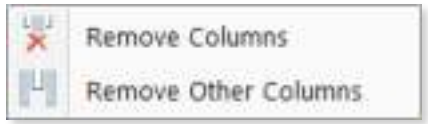
BASIC TABLE TRANSFORMATIONS

Sort values *(A-Z, Low-High, etc.)*

Change data type *(date, $, %, text, etc.)*

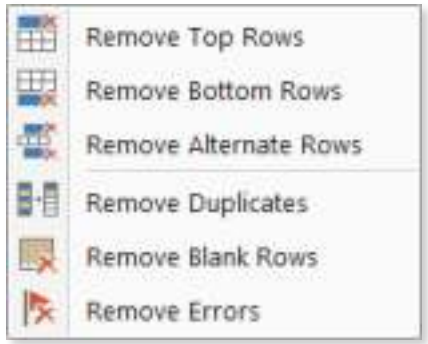


Promote headers



Choose or remove columns

Tip: *use the "Remove Other Columns" option if you always want a specific set*

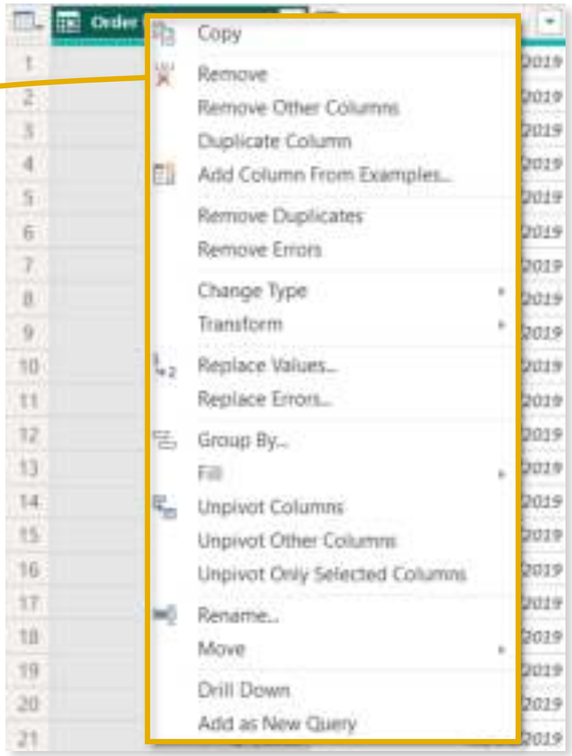


Keep or remove rows

Tip: *use "Remove Duplicates" to create new lookup tables from scratch*

Duplicate, **move** or **rename** columns

Tip: *Right-click column headers to access common tools*



ASSIGNMENT: TABLE TRANSFORMATIONS

NEW MESSAGE

From: **Ethan T. Langer** *(Analytics Manager)*

Subject: **Welcome aboard!**

Hello, and welcome to the team!

We're excited that you'll be helping us develop our new internal reports in Power BI. Looks like you've already gotten started, but we have some new data to add to the model.

Could you please create two new queries to connect to the **Product Category Lookup** and **Product Subcategory Lookup** files attached, and help with a few modifications to the product table?

Thanks!
-ETL

Product Category Lookup
Product Subcategory Lookup

Reply Forward

Key Objectives

1. Create queries to connect to the two new .csv files
2. Name your queries **Product Category Lookup** and **Product Subcategory Lookup**
3. Confirm that column headers have been promoted and that all data types are correct
4. Add a new column to extract all characters before the dash ("-") in the **Product SKU** column, and name it "**SKU Type**"
5. Update the **SKU Type** calculation above to return all characters before *second* dash, instead of the first
6. Replace zeros (**0**) in the **Product Style** column with "**NA**"
7. Close and load to your data model

SOLUTION: TABLE TRANSFORMATIONS

NEW MESSAGE

From: **Ethan T. Langer** *(Analytics Manager)*

Subject: **Welcome aboard!**

Hello, and welcome to the team!

We're excited that you'll be helping us develop our new internal reports in Power BI. Looks like you've already gotten started, but we have some new data to add to the model.

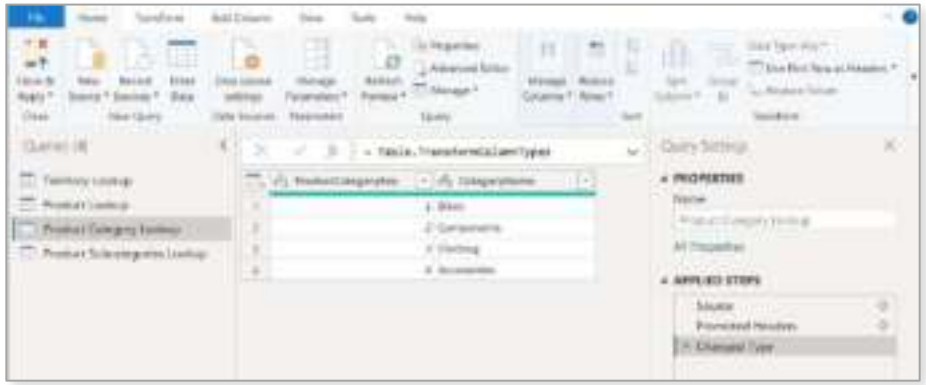
Could you please create two new queries to connect to the **Product Category Lookup** and **Product Subcategory Lookup** files attached, and help with a few modifications to the product table?

Thanks!
-ETL

Product Category Lookup
Product Subcategory Lookup

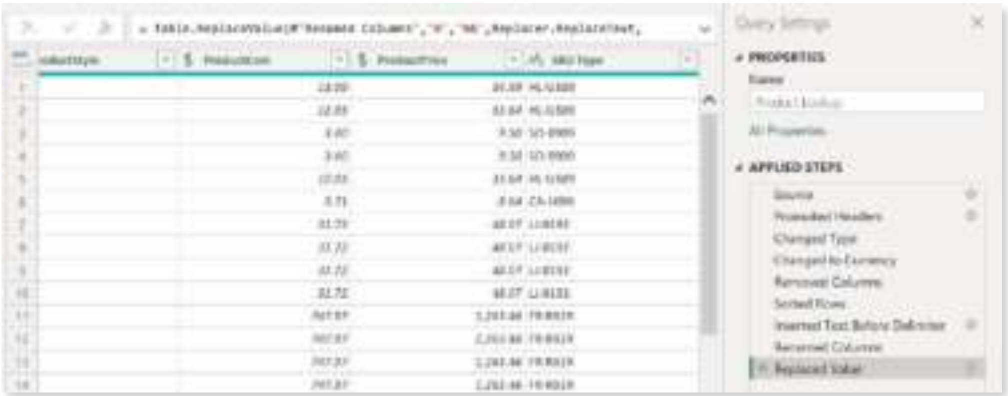
Reply Forward

Solution Preview



= Table.TransformColumnTypes

APPLIED STEPS: Source, Promoted Headers, Changed Type



= Table.ReplaceValue(#"Renamed Columns","0","NA",Replacer.ReplaceText,

APPLIED STEPS: Source, Promoted Headers, Changed Type, Changed to Currency, Removed Columns, Sorted Rows, Inserted Text Before Delimiter, Renamed Columns, Replaced Value

PRO TIP: STORAGE & CONNECTION MODES

Power BI Desktop supports several types of **storage** and **connection modes**:

- **Import**: Tables are stored in-memory within Power BI and queries are fulfilled by cached data *(default)*
- **DirectQuery**: Tables are connected directly to the source and queries are executed on-demand at the data source
- **Composite Model (Dual)**: Tables come from a mix of Import and DirectQuery modes, or integrate multiple DirectQuery tables
- **Live Connection**: Connect to pre-published Power BI datasets in Power BI Service or Azure Analysis Services

Import

- ✓ Dataset is less than 1GB (after compression) & fast performance
- ✓ Source data does not change frequently
- ✓ No restrictions on Power Query, data modeling, and DAX functions

DirectQuery

- ✓ Dataset is too large to be stored in-memory
- ✓ Source data changes frequently and reports must reflect changes
- ✓ Company policy states that data can only be accessed from the original source

Composite Model

- ✓ Boost performance by setting appropriate storage for each table
- ✓ Combine a DirectQuery model with additional imported data
- ✓ Create a single model from two or more DirectQuery models

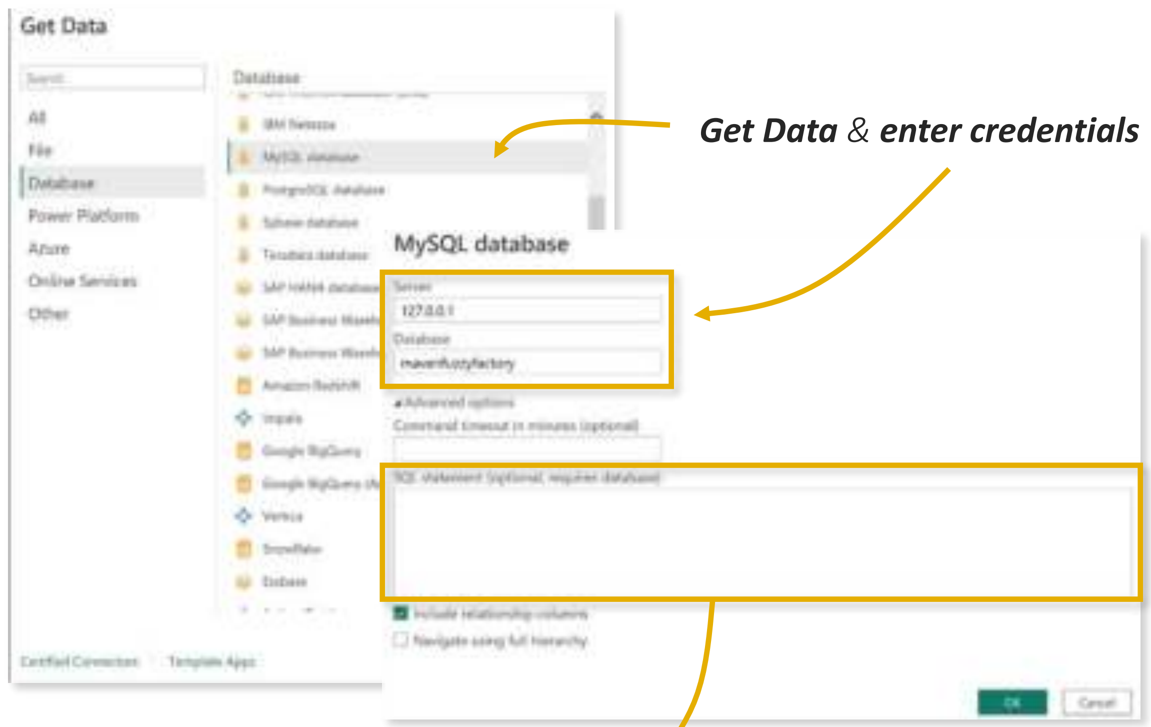
Live Connection

- ✓ Create one dataset that serves as a central source of truth
- ✓ Analyst teams can create different reports from the same source
- ✓ Multi-developer teams where one user builds the model and another works on visualization

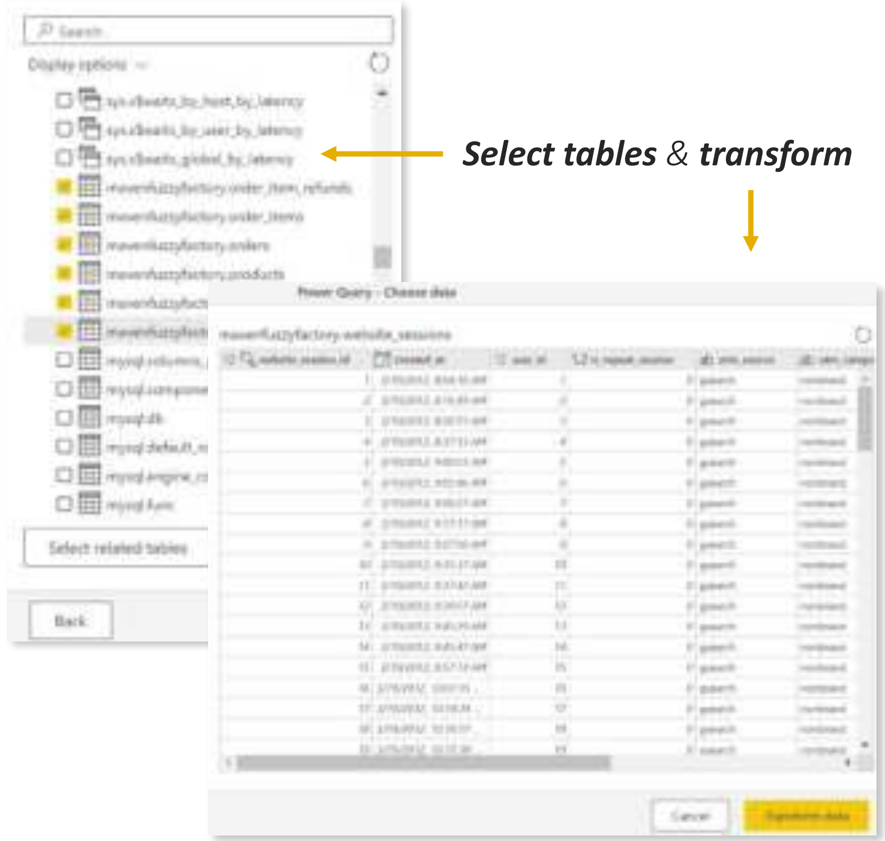
Learn more: *https://learn.microsoft.com/en-us/power-bi/connect-data/service-dataset-modes-understand*

CONNECTING TO A DATABASE

Power Query can connect to data from various **database sources** including SQL Server, MS Access, MySQL, PostgreSQL, Oracle, SAP, and more



Get Data & ***enter credentials***

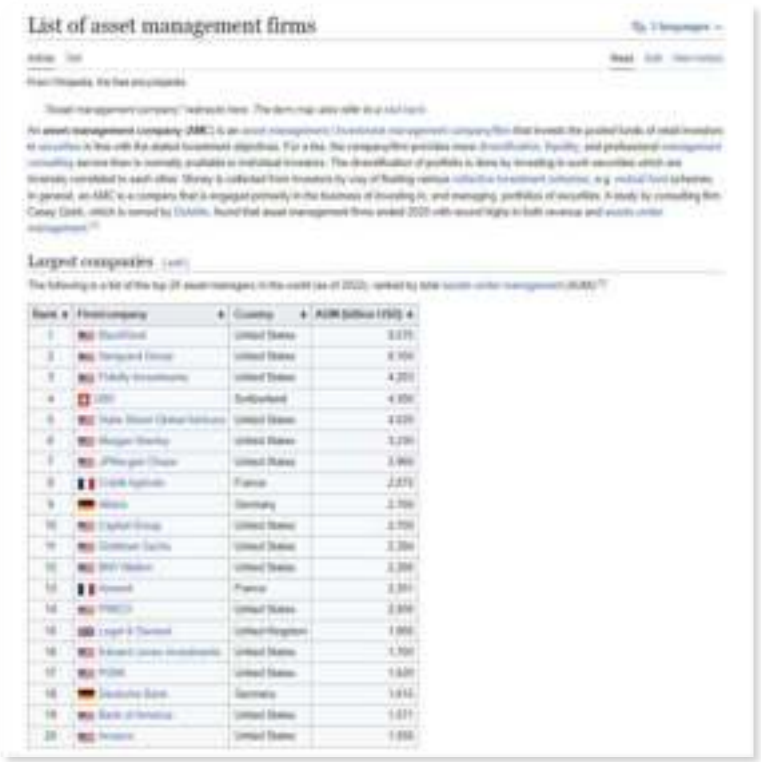

Search
Display options
sys.x$waits_by_host_by_latency
sys.x$waits_by_user_by_latency
sys.x$waits_global_by_latency
mavenfuzzyfactory.order_item_refunds
mavenfuzzyfactory.order_items
mavenfuzzyfactory.orders
mavenfuzzyfactory.products
Select related tables
Back
Power Query - Choose data
mavenfuzzyfactory.website_sessions
website session id
created at
user id
is repeat session
utm source
Cancel
Transform data

Select tables & ***transform***

Write custom or advanced queries *with SQL statements (optional)*

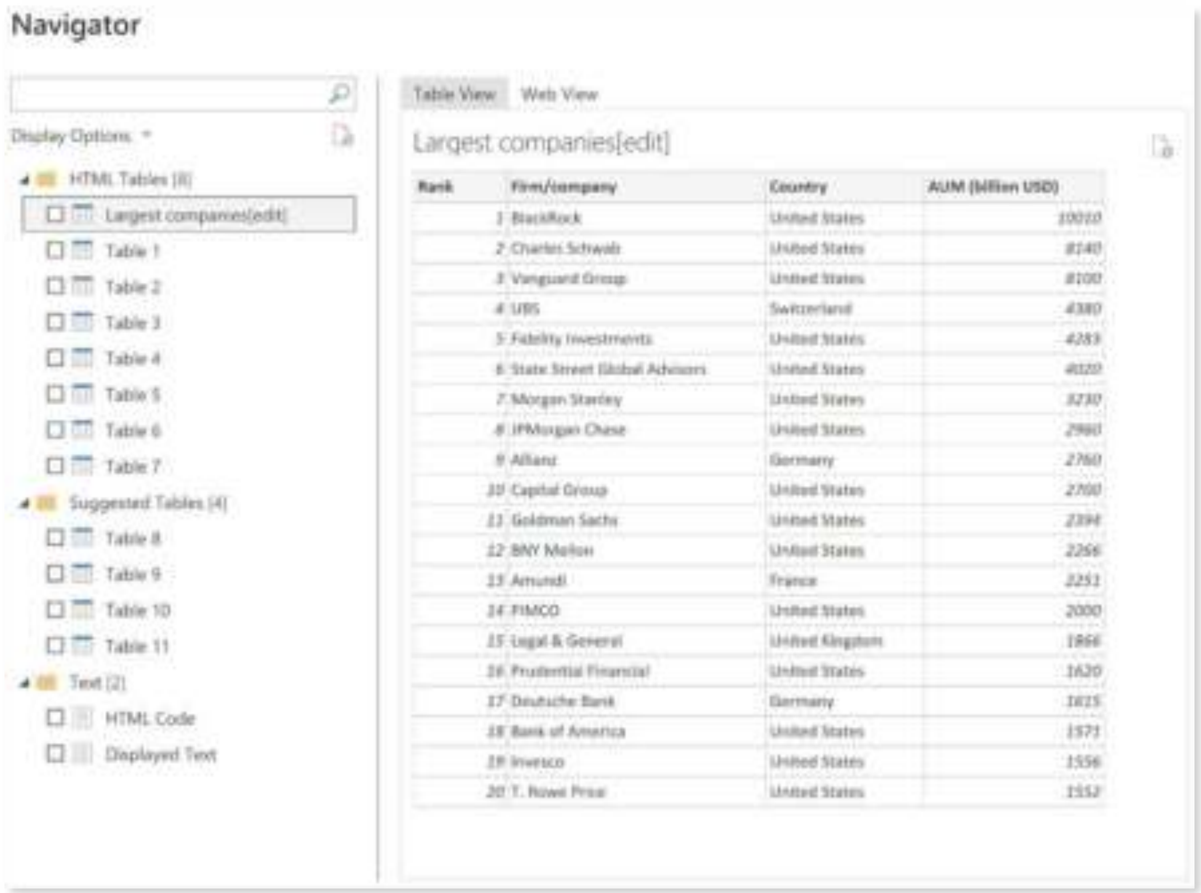
EXTRACTING DATA FROM THE WEB

Power Query includes a native **Web connector** for importing web-hosted files (csv, xlsx, etc.) or scraping URLs for anything that Power Query can identify as a structured table


List of asset management firms

Largest companies [edit]

https://en.wikipedia.org/wiki/List_of_asset_management_firms


Navigator

Display Options

- HTML Tables [8]
  - Largest companies[edit]
  - Table 1
  - Table 2
  - Table 3
  - Table 4
  - Table 5
  - Table 6
  - Table 7
- Suggested Tables [4]
  - Table 8
  - Table 9
  - Table 10
  - Table 11
- Text [2]
  - HTML Code
  - Displayed Text

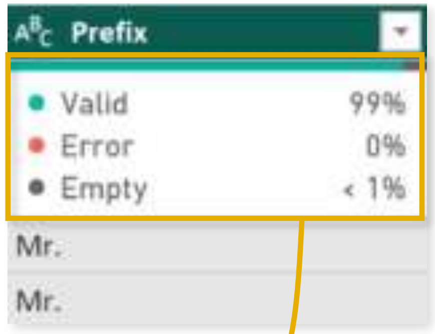
Table View | Web View

Largest companies[edit]

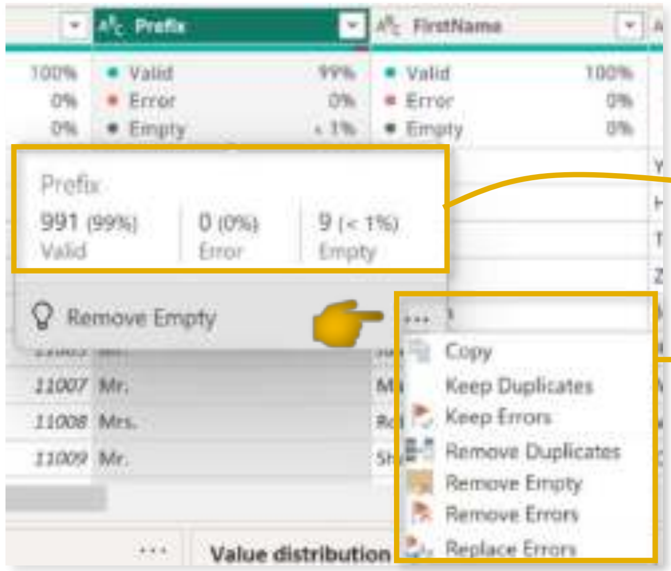
Rank	Firm/company	Country	AUM (billion USD)
1	BlackRock	United States	10010
2	Charles Schwab	United States	8140
3	Vanguard Group	United States	8100
4	UBS	Switzerland	4380
5	Fidelity Investments	United States	4283
6	State Street Global Advisors	United States	4020
7	Morgan Stanley	United States	3230
8	JPMorgan Chase	United States	2960
9	Allianz	Germany	2760
10	Capital Group	United States	2700
11	Goldman Sachs	United States	2394
12	BNY Mellon	United States	2256
13	Amundi	France	2251
14	PIMCO	United States	2000
15	Legal & General	United Kingdom	1866
16	Prudential Financial	United States	1620
17	Deutsche Bank	Germany	1615
18	Bank of America	United States	1571
19	Invesco	United States	1556
20	T. Rowe Price	United States	1552

DATA PROFILING: COLUMN QUALITY

Profiling tools like **column quality**, **column distribution**, and **column profile** allow you to explore the quality, composition, and distribution of your data before loading it into the Power BI front-end



Column quality *shows the percentage of values within a column that are* ***valid****, contain* ***errors****, or are* ***empty***



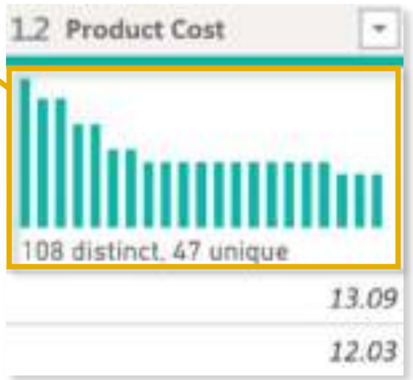
Hover over the column quality box to see the ***number of records*** *in each category*

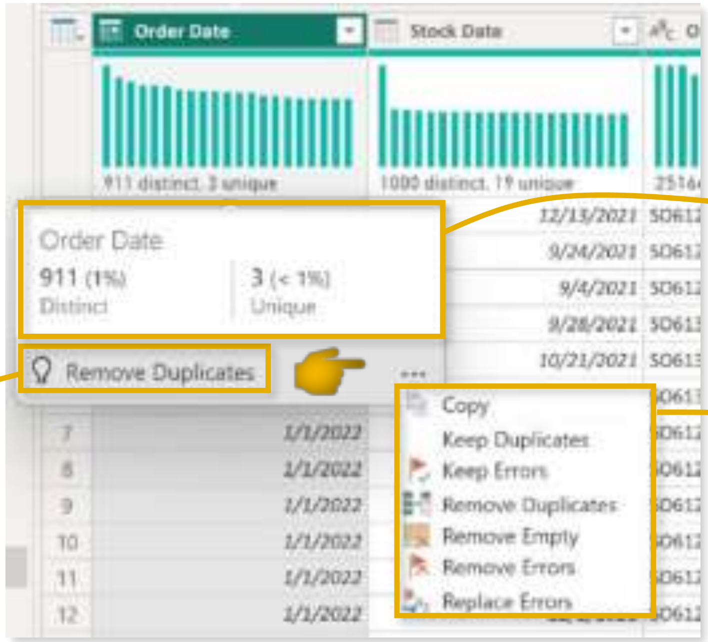
Click the ***options menu*** *to remove duplicates, errors or empty values*

PRO TIP: Profiling tools are a great way to **quickly find and address common data quality issues in one place**, instead of having to manually apply multiple tools or filters

DATA PROFILING: COLUMN DISTRIBUTION

***Column distribution** provides a sample distribution of the data in a column*





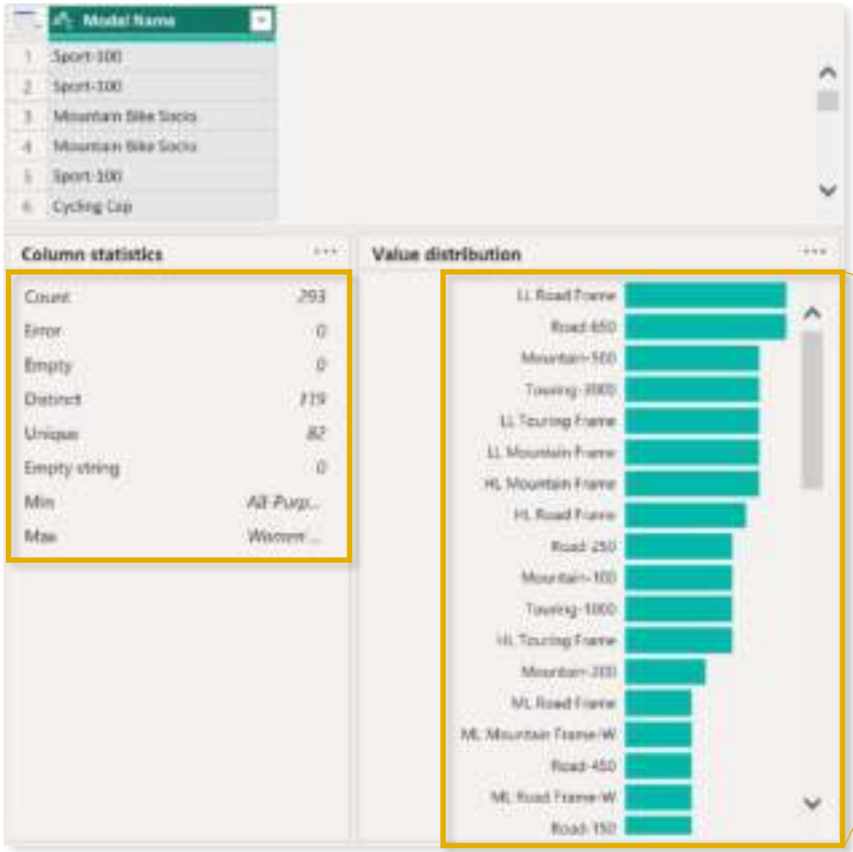
*Hover over the column quality box to see the **number of distinct & unique records***

*Click the **options menu** to remove duplicates, errors or empty values*

***Suggested action** based on column distribution results*

DATA PROFILING: COLUMN PROFILE

Column profile *provides a more holistic view of the data in a column, including a sample distribution and profiling statistics*



	Model Name
1	Sport-100
2	Sport-100
3	Mountain Bike Socks
4	Mountain Bike Socks
5	Sport-100
6	Cycling Cap

Column statistics

Count	293
Error	0
Empty	0
Distinct	119
Unique	82
Empty string	0
Min	All-Purp...
Max	Women ...

Value distribution

Column statistics *provide more detailed profiling metrics, including:*

Count = **293**

(total number of values in column)

Distinct Count = **119**

(total number of distinct values, whether they appear once or multiple times)

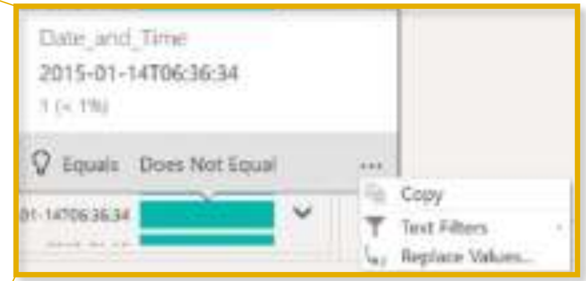
Unique = **82**

(total number of values that appear exactly once)

Min & Max

(lowest and highest observed values)

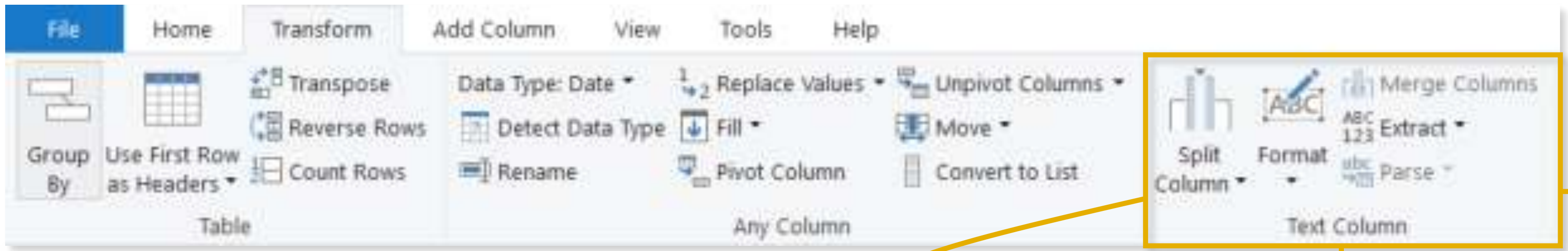
Note: *Typically only useful for numerical values*

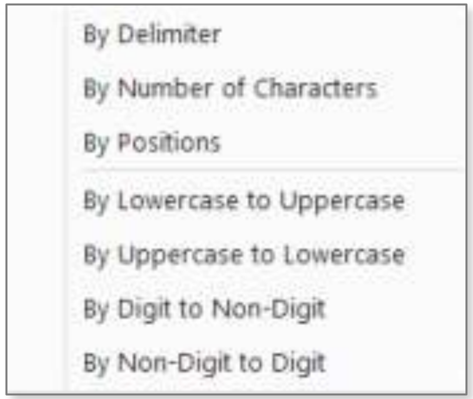


Date_and_Time
2015-01-14T06:36:34
1 (< 1%)

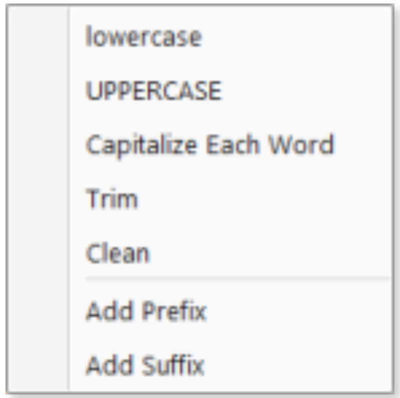
Hover over the value distribution bar for ***suggested transformations*** *and additional options*

TEXT TOOLS



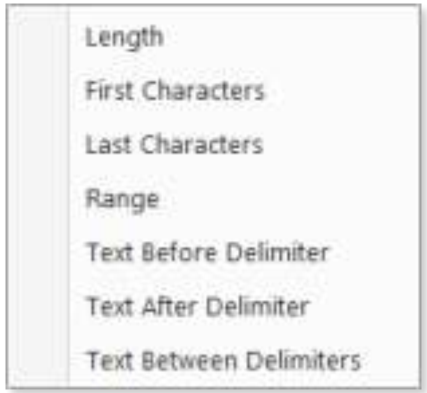


Split a text column based on a specific delimiter, number of characters, or other attributes



Format a text column to upper, lower or proper case, or add a prefix or suffix

Tip: *Use "Trim" to eliminate leading & trailing spaces, or "Clean" to remove non-printable characters*



Extract characters from text based on fixed lengths, first/last characters, ranges or delimiters

HEY THIS IS IMPORTANT!

You can access many tools from both the **Transform** and **Add Column** menus - the difference is whether you want to **ADD** a new column or **OVERWRITE** an existing one

ASSIGNMENT: TEXT TOOLS

NEW MESSAGE

From: **Ethan T. Langer** *(Analytics Manager)*

Subject: **Customer domains**

Hi!

We're looking to better understand where our customers may be coming from, based on their email domains.

Could you please create a new column in the customer table that will allow us do this?

Thanks!
-ETL

Reply Forward

Key Objectives

1. Duplicate the email address column and name it "**Domain Name"**
2. In the new column, remove all text/characters except for the domain name
3. Use transformation steps to clean up and capitalize the domain names (i.e. "**Adventure Works**")
4. Save & Apply changes

SOLUTION: TEXT TOOLS

NEW MESSAGE

From: **Ethan T. Langer** *(Analytics Manager)*

Subject: **Customer domains**

Hi!

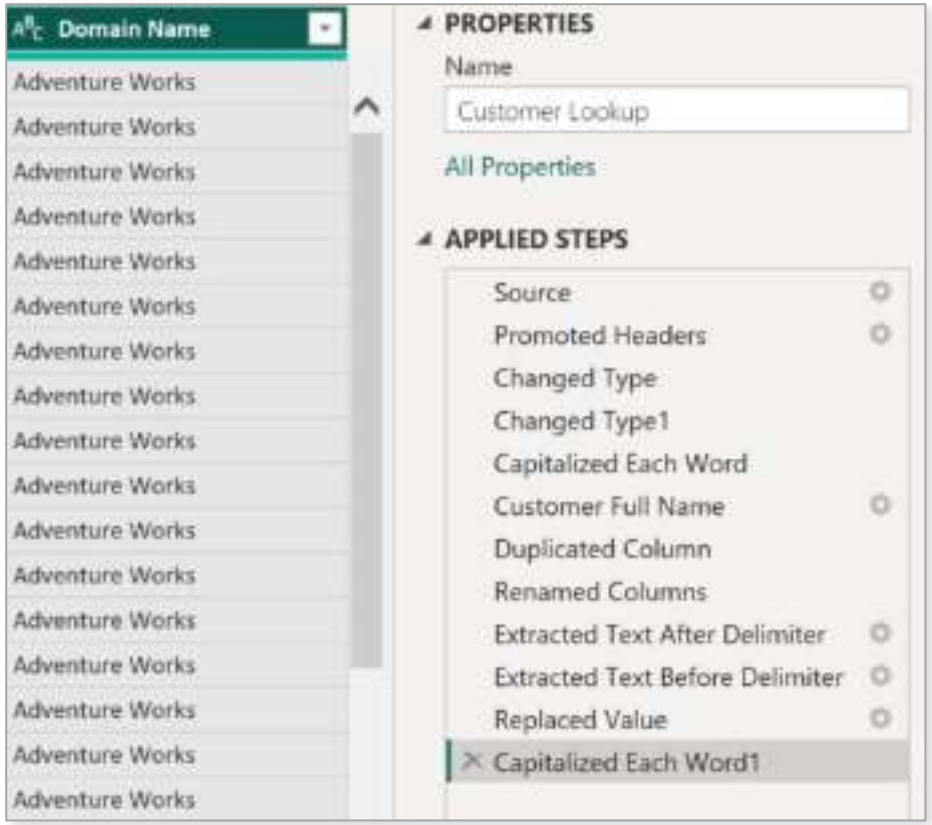
We're looking to better understand where our customers may be coming from, based on their email domains.

Could you please create a new column in the customer table that will allow us do this?

Thanks!
-ETL

Reply Forward

Solution Preview

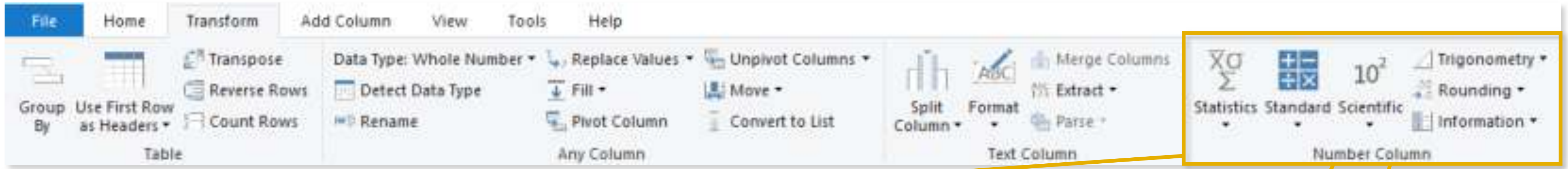


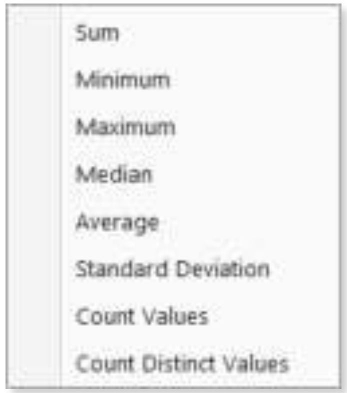
Domain Name
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works
Adventure Works

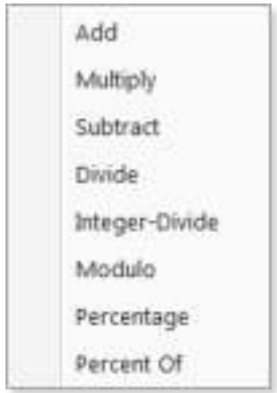
PROPERTIES
Name
Customer Lookup
All Properties

APPLIED STEPS
- Source
- Promoted Headers
- Changed Type
- Changed Type1
- Capitalized Each Word
- Customer Full Name
- Duplicated Column
- Renamed Columns
- Extracted Text After Delimiter
- Extracted Text Before Delimiter
- Replaced Value
- Capitalized Each Word1

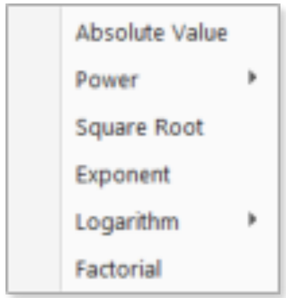
NUMERICAL TOOLS



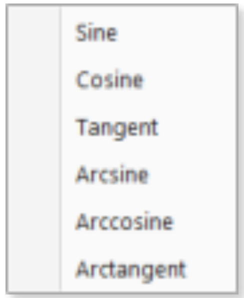




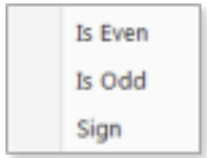
Standard



Scientific



Trigonometry



Information tools allow you to define binary flags (*1/0 or TRUE/FALSE*) to mark rows as even, odd, positive or negative

Statistics functions allow you to evaluate basic stats for a selected column (sum, min/max, average, count, count distinct, etc.)

Note: *These tools return a SINGLE value, and are commonly used to explore a table rather than prepare it for loading*

Standard, **Scientific** and **Trigonometry** tools allow you to apply standard operations (addition, multiplication, division, etc.) or more advanced calculations (power, logarithm, sine, tangent, etc.) to each value in a column

Note: *Unlike the Statistics tools, these are applied to each row in the table*

ASSIGNMENT: NUMERICAL TOOLS

NEW MESSAGE

From: **Ethan T. Langer** *(Analytics Manager)*

Subject: **Need some stats for leadership**

Hi again,

Leadership is asking us to validate some high-level stats about our products and customers. Can you please help me answer the following questions?

We don't really need to store these values anywhere, so make sure to restore the tables back to their original state once you're done pulling the stats.

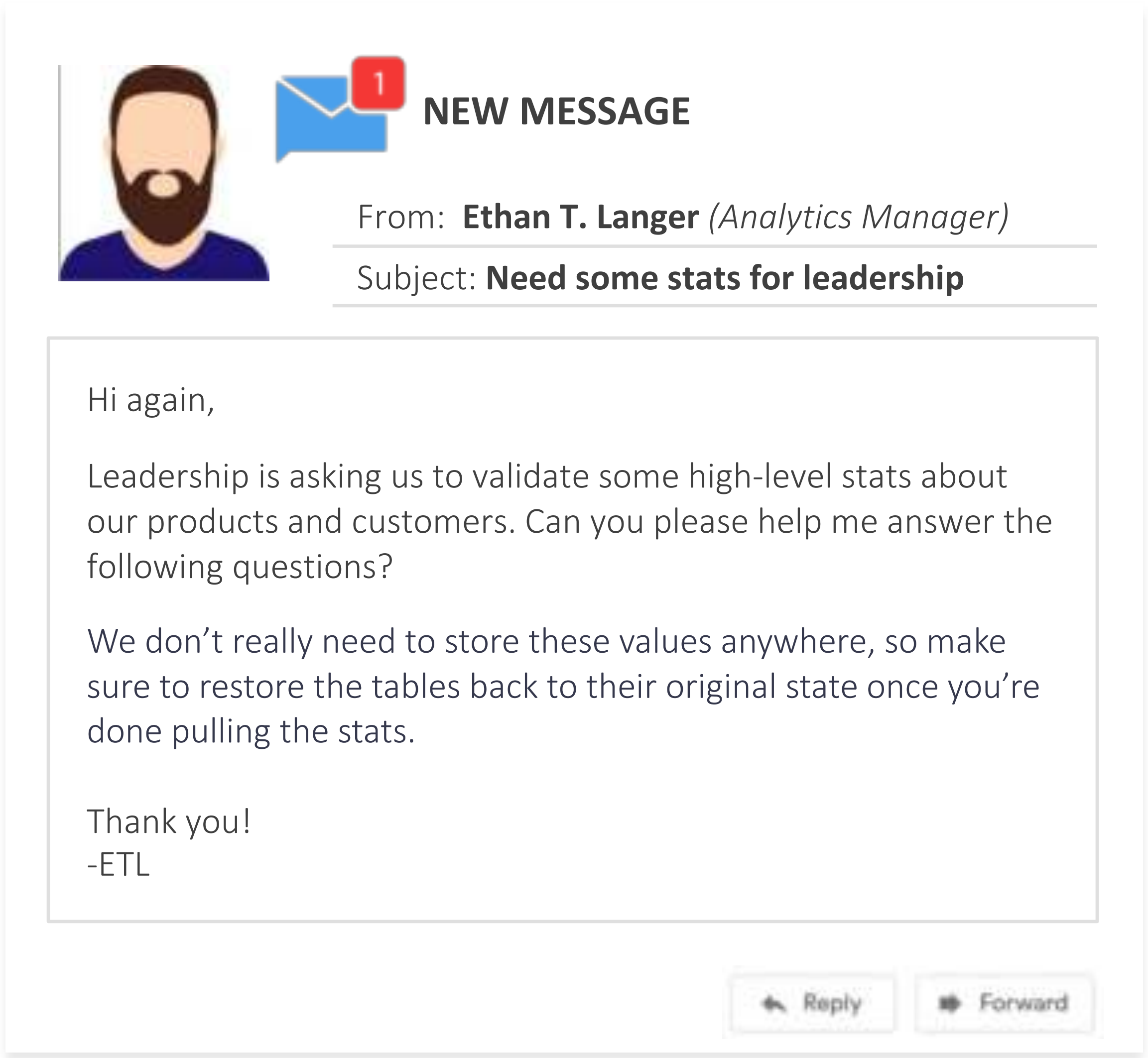
Thank you!
-ETL

Reply Forward

Key Objectives

1. What is our average product cost?
2. How many colors do we sell our products in?
3. How many distinct customers do we have?
4. What is the maximum annual customer income?
5. Return the tables to their original state

SOLUTION: NUMERICAL TOOLS



NEW MESSAGE

From: **Ethan T. Langer** *(Analytics Manager)*

Subject: **Need some stats for leadership**

Hi again,

Leadership is asking us to validate some high-level stats about our products and customers. Can you please help me answer the following questions?

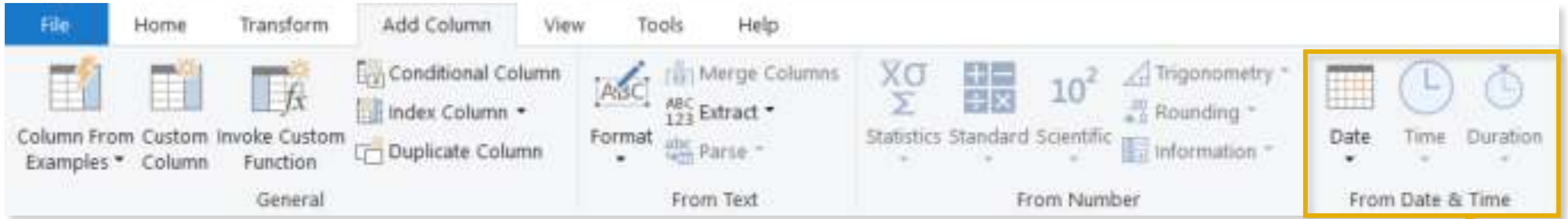
We don't really need to store these values anywhere, so make sure to restore the tables back to their original state once you're done pulling the stats.

Thank you!
-ETL

Solution Preview

1. What is our average product cost? **($413.66)**
2. How many colors do we sell our products in? **(10)**
3. How many distinct customers do we have? **(18,148)**
4. What is the maximum annual customer income? **($170k)**
5. Return the tables to their original state

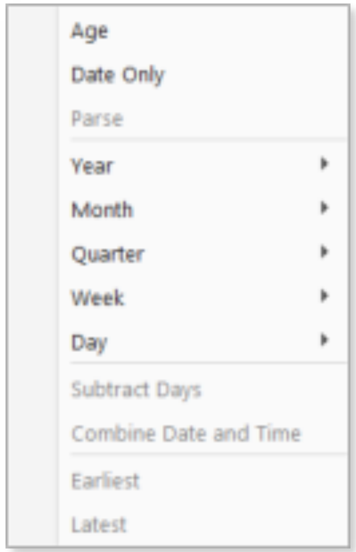
DATE & TIME TOOLS



Date & Time tools are relatively straight-forward, and include the following options:

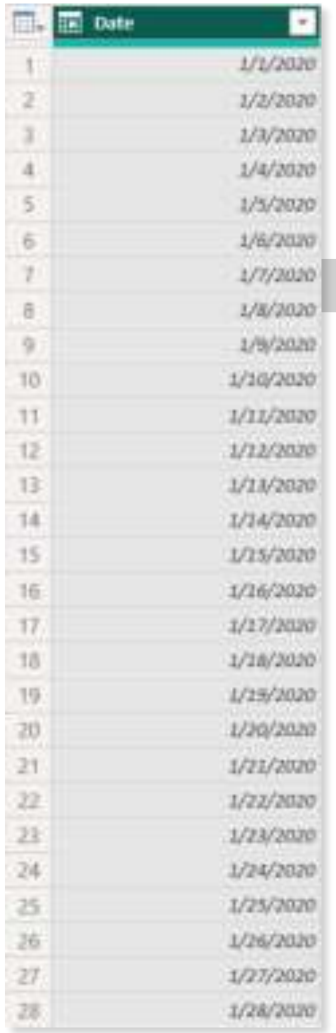
- **Age**: Difference between the current date and the date in each row
- **Date Only:** Removes the time component from a date/time field
- **Year/Month/Quarter/Week/Day**: Extracts individual components from a date field (time-specific options include Hour, Minute, Second, etc.)
- **Earliest/Latest:** Evaluates the earliest or latest date from a column as a single value (can only be accessed from the "Transform" menu)

Note: *You will almost always want to perform these operations from the "Add Column" menu to build out new fields, rather than transforming an individual date/time column*

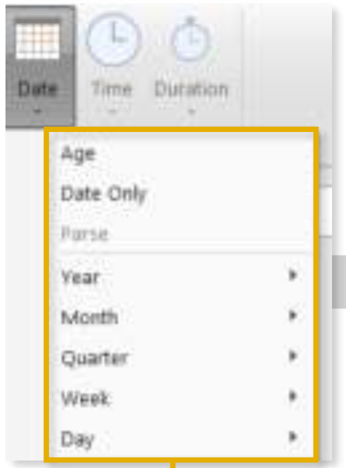


PRO TIP: Load up a table containing a **single date column** and use Date tools to build out an **entire calendar table**

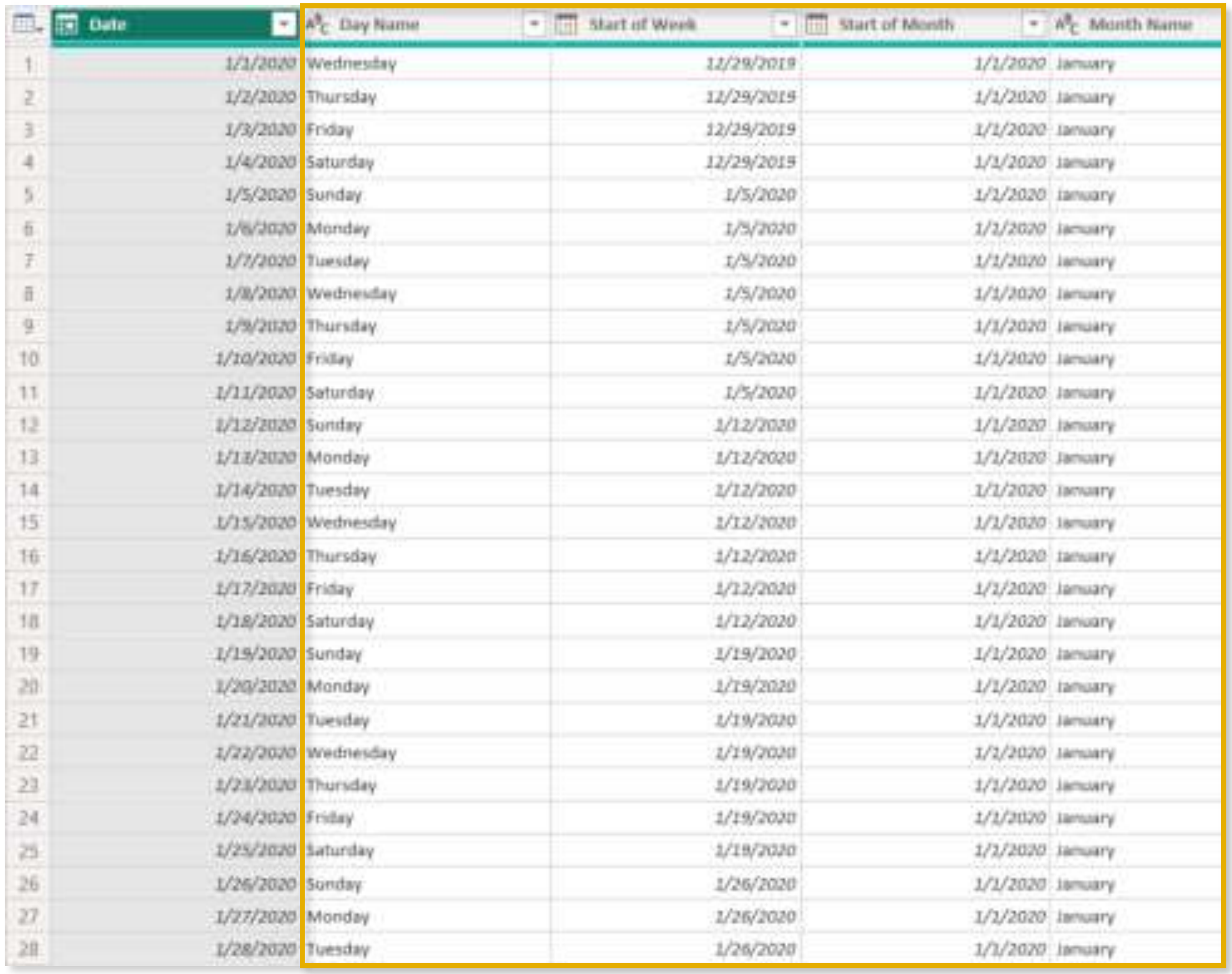
CREATING A CALENDAR TABLE



	Date
1	1/1/2020
2	1/2/2020
3	1/3/2020
4	1/4/2020
5	1/5/2020
6	1/6/2020
7	1/7/2020
8	1/8/2020
9	1/9/2020
10	1/10/2020
11	1/11/2020
12	1/12/2020
13	1/13/2020
14	1/14/2020
15	1/15/2020
16	1/16/2020
17	1/17/2020
18	1/18/2020
19	1/19/2020
20	1/20/2020
21	1/21/2020
22	1/22/2020
23	1/23/2020
24	1/24/2020
25	1/25/2020
26	1/26/2020
27	1/27/2020
28	1/28/2020

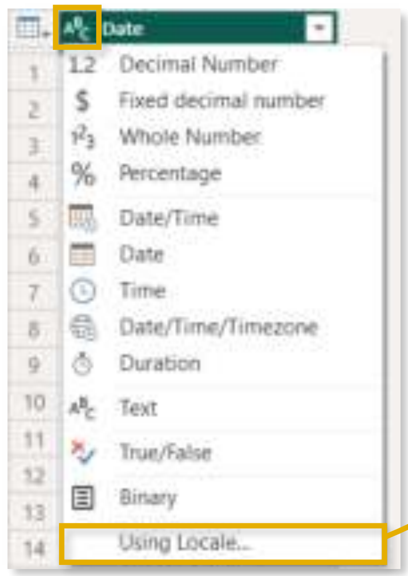


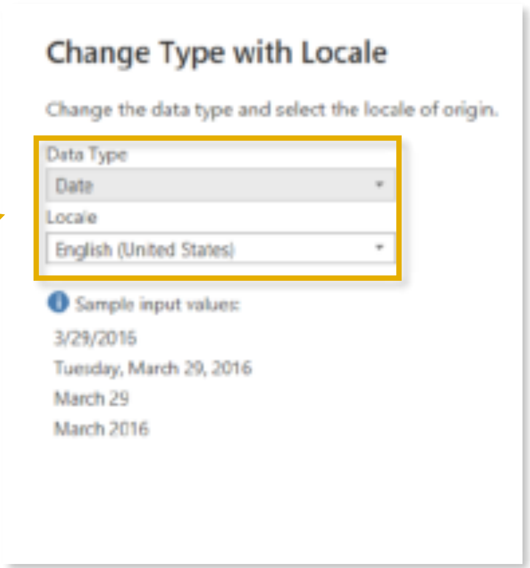
Use the **Date** options in the **Add Column** menu to quickly build out an entire calendar table from a list of dates



	Date	Day Name	Start of Week	Start of Month	Month Name
1	1/1/2020	Wednesday	12/29/2019	1/1/2020	January
2	1/2/2020	Thursday	12/29/2019	1/1/2020	January
3	1/3/2020	Friday	12/29/2019	1/1/2020	January
4	1/4/2020	Saturday	12/29/2019	1/1/2020	January
5	1/5/2020	Sunday	1/5/2020	1/1/2020	January
6	1/6/2020	Monday	1/5/2020	1/1/2020	January
7	1/7/2020	Tuesday	1/5/2020	1/1/2020	January
8	1/8/2020	Wednesday	1/5/2020	1/1/2020	January
9	1/9/2020	Thursday	1/5/2020	1/1/2020	January
10	1/10/2020	Friday	1/5/2020	1/1/2020	January
11	1/11/2020	Saturday	1/5/2020	1/1/2020	January
12	1/12/2020	Sunday	1/12/2020	1/1/2020	January
13	1/13/2020	Monday	1/12/2020	1/1/2020	January
14	1/14/2020	Tuesday	1/12/2020	1/1/2020	January
15	1/15/2020	Wednesday	1/12/2020	1/1/2020	January
16	1/16/2020	Thursday	1/12/2020	1/1/2020	January
17	1/17/2020	Friday	1/12/2020	1/1/2020	January
18	1/18/2020	Saturday	1/12/2020	1/1/2020	January
19	1/19/2020	Sunday	1/19/2020	1/1/2020	January
20	1/20/2020	Monday	1/19/2020	1/1/2020	January
21	1/21/2020	Tuesday	1/19/2020	1/1/2020	January
22	1/22/2020	Wednesday	1/19/2020	1/1/2020	January
23	1/23/2020	Thursday	1/19/2020	1/1/2020	January
24	1/24/2020	Friday	1/19/2020	1/1/2020	January
25	1/25/2020	Saturday	1/19/2020	1/1/2020	January
26	1/26/2020	Sunday	1/26/2020	1/1/2020	January
27	1/27/2020	Monday	1/26/2020	1/1/2020	January
28	1/28/2020	Tuesday	1/26/2020	1/1/2020	January

CHANGE TYPE WITH LOCALE





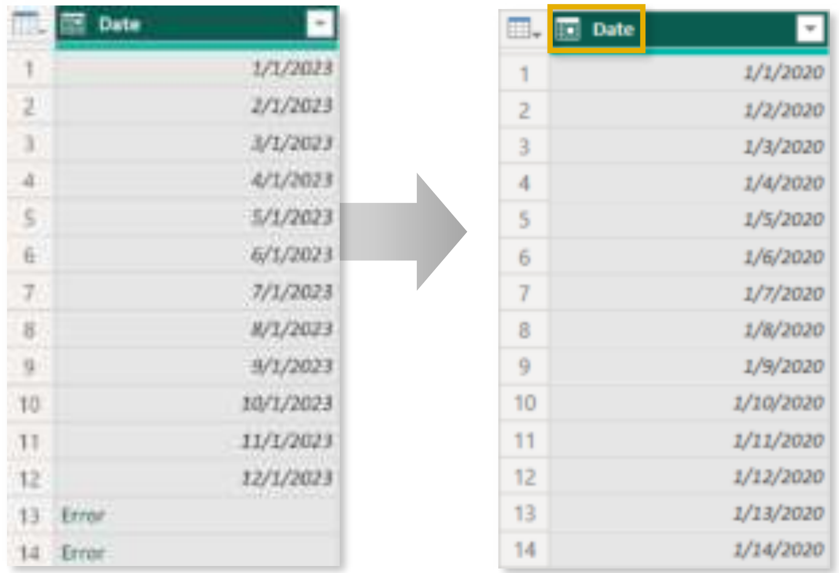
Change Type with Locale

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



	Date
1	1/1/2023
2	2/1/2023
3	3/1/2023
4	4/1/2023
5	5/1/2023
6	6/1/2023
7	7/1/2023
8	8/1/2023
9	9/1/2023
10	10/1/2023
11	11/1/2023
12	12/1/2023
13	Error
14	Error

	Date
1	1/1/2020
2	1/2/2020
3	1/3/2020
4	1/4/2020
5	1/5/2020
6	1/6/2020
7	1/7/2020
8	1/8/2020
9	1/9/2020
10	1/10/2020
11	1/11/2020
12	1/12/2020
13	1/13/2020
14	1/14/2020

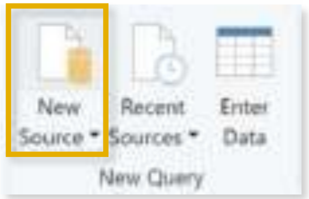
1) Left click the data type icon in the column header and select the **Using Locale** option

2) Select **Date** as the data type and **English (United States)** as the locale for all datasets in this course (regardless of your actual location)

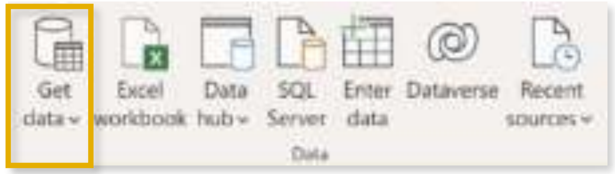
3) Confirm that the **data type is correctly recognized**. You should see a calendar icon next to the column name in the header and no errors in the column

PRO TIP: ROLLING CALENDARS

1 Create a new **blank query** & name it "***Rolling Calendar***"

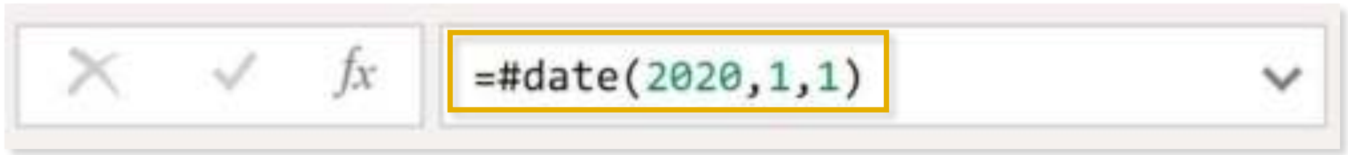


Power Query: *New Source > Blank Query*



Front end: *Get Data > Blank Query*

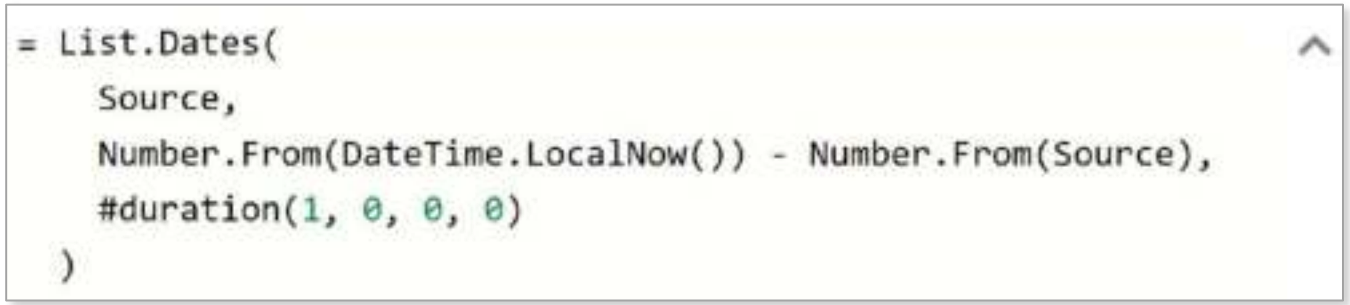
2 In the formula bar, type a **"literal"** to generate a start date:



```
=#date(2020,1,1)
```

Format as: *YYYY, MM, DD*

3 Click the fx icon to **add a custom step**, and enter the following formula to generate a list of dates between the start date and the current day:



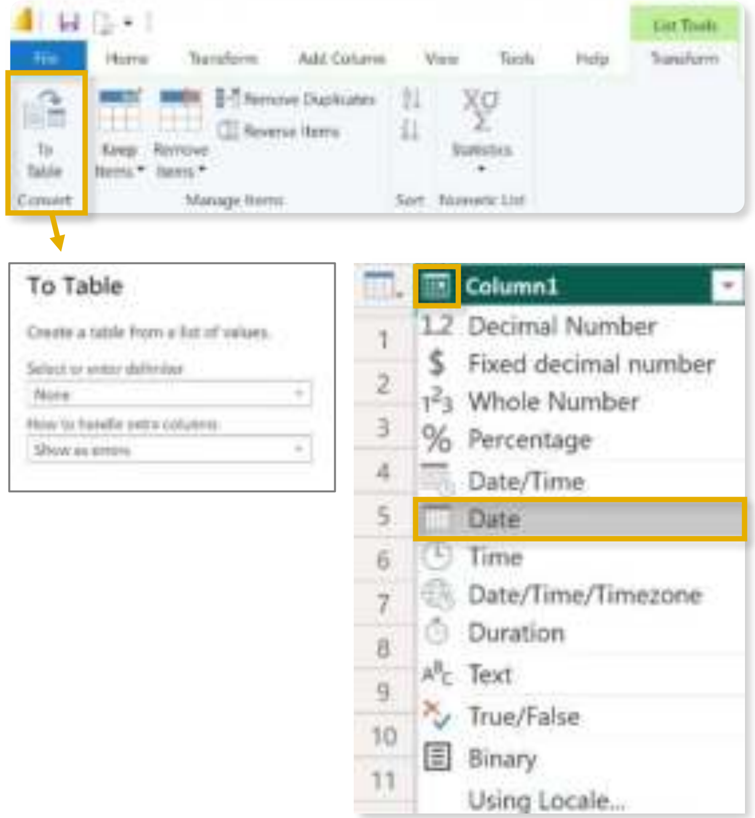
```
= List.Dates(
    Source,
    Number.From(DateTime.LocalNow()) - Number.From(Source),
    #duration(1, 0, 0, 0)
  )
```

Note: *If your first applied step is named something other than "**Source**", use that name in your formula (this is common for non-US users)*

PRO TIP: ROLLING CALENDARS

4 Convert the resulting list into a **Table** and set the data type as a **Date**

5 Rename the column to "**Date**" and add calculated date columns (year, month, quarter, etc.) using the **Add Column** tools

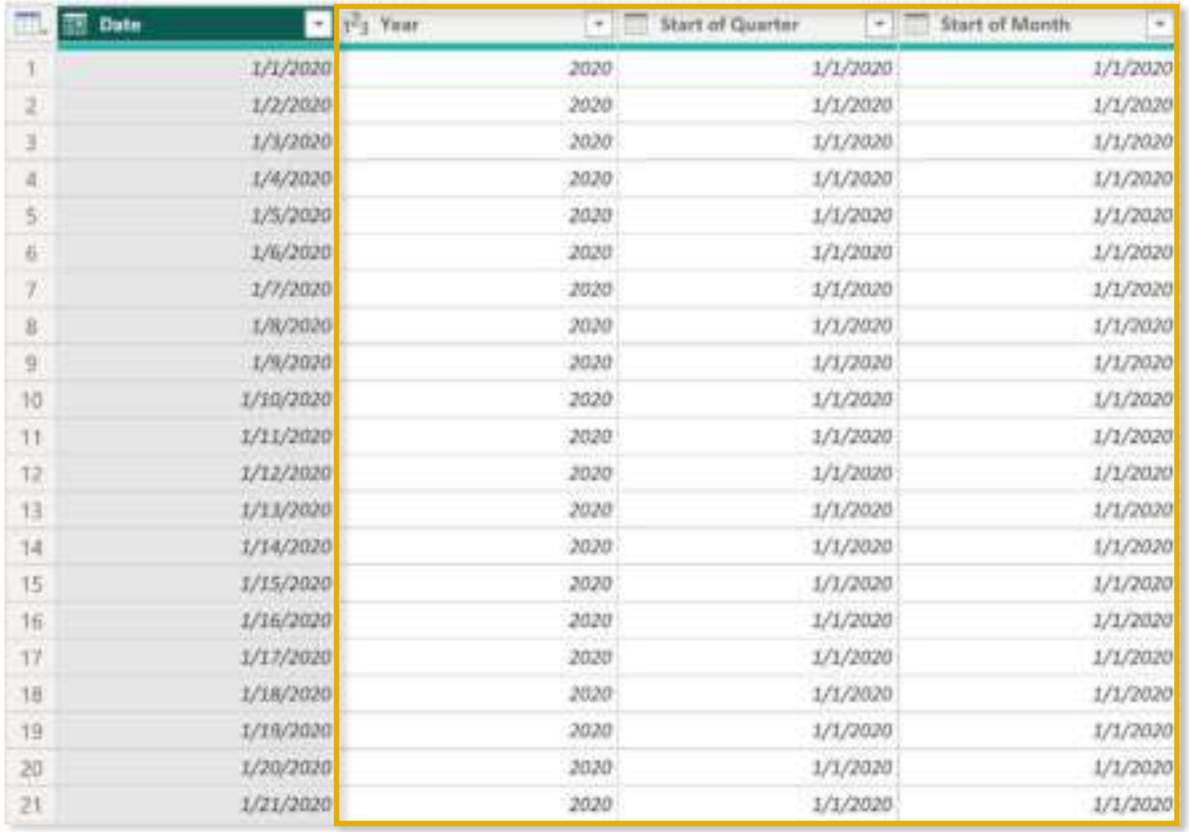


To Table

Create a table from a list of values.

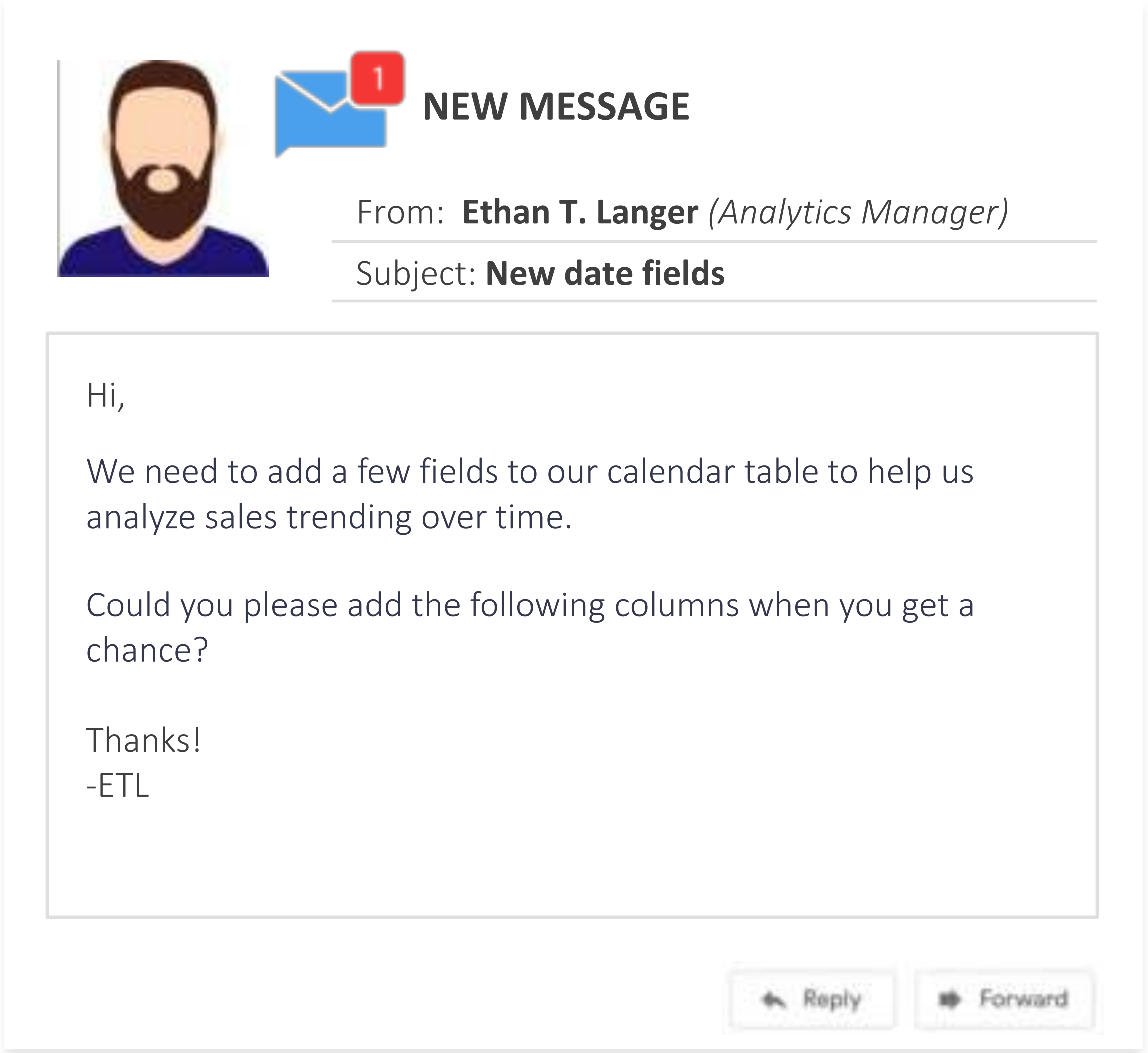
Select or enter delimiter: None

How to handle extra columns: Show as errors



	Date	Year	Start of Quarter	Start of Month
1	1/1/2020	2020	1/1/2020	1/1/2020
2	1/2/2020	2020	1/1/2020	1/1/2020
3	1/3/2020	2020	1/1/2020	1/1/2020
4	1/4/2020	2020	1/1/2020	1/1/2020
5	1/5/2020	2020	1/1/2020	1/1/2020
6	1/6/2020	2020	1/1/2020	1/1/2020
7	1/7/2020	2020	1/1/2020	1/1/2020
8	1/8/2020	2020	1/1/2020	1/1/2020
9	1/9/2020	2020	1/1/2020	1/1/2020
10	1/10/2020	2020	1/1/2020	1/1/2020
11	1/11/2020	2020	1/1/2020	1/1/2020
12	1/12/2020	2020	1/1/2020	1/1/2020
13	1/13/2020	2020	1/1/2020	1/1/2020
14	1/14/2020	2020	1/1/2020	1/1/2020
15	1/15/2020	2020	1/1/2020	1/1/2020
16	1/16/2020	2020	1/1/2020	1/1/2020
17	1/17/2020	2020	1/1/2020	1/1/2020
18	1/18/2020	2020	1/1/2020	1/1/2020
19	1/19/2020	2020	1/1/2020	1/1/2020
20	1/20/2020	2020	1/1/2020	1/1/2020
21	1/21/2020	2020	1/1/2020	1/1/2020

ASSIGNMENT: CALENDAR TABLES



NEW MESSAGE

From: **Ethan T. Langer** *(Analytics Manager)*

Subject: **New date fields**

Hi,

We need to add a few fields to our calendar table to help us analyze sales trending over time.

Could you please add the following columns when you get a chance?

Thanks!
-ETL

Reply
Forward

Key Objectives

Add the following columns to the calendar table:

1. **Month Name** (e.g. "January")
2. **Month Number** (e.g. "1")
3. **Start of Year** (e.g. "1/1/2020")
4. **Year** (e.g. "2020")

SOLUTION: CALENDAR TABLES

NEW MESSAGE

From: **Ethan T. Langer** *(Analytics Manager)*

Subject: **New date fields**

Hi,

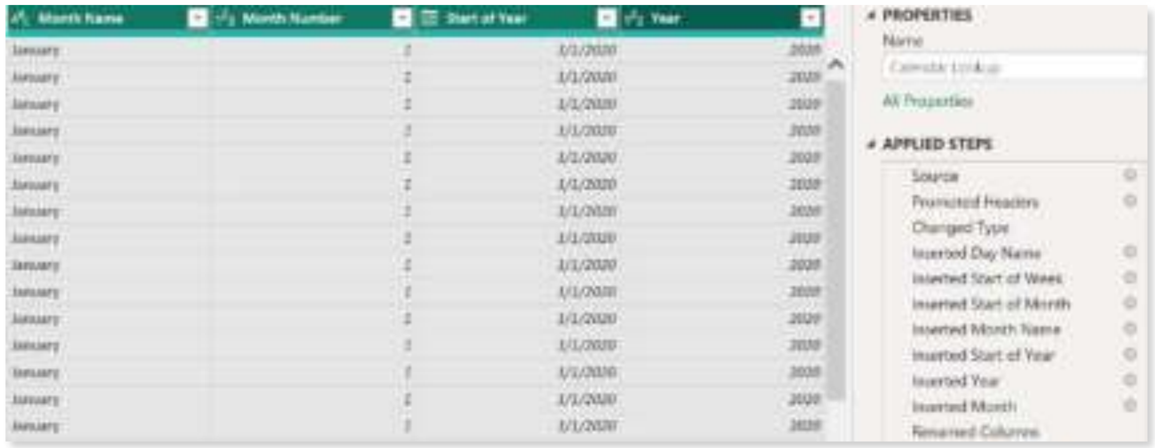
We need to add a few fields to our calendar table to help us analyze sales trending over time.

Could you please add the following columns when you get a chance?

Thanks!
-ETL

Reply Forward

Solution Preview

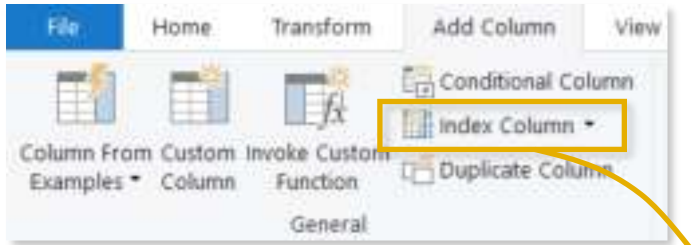


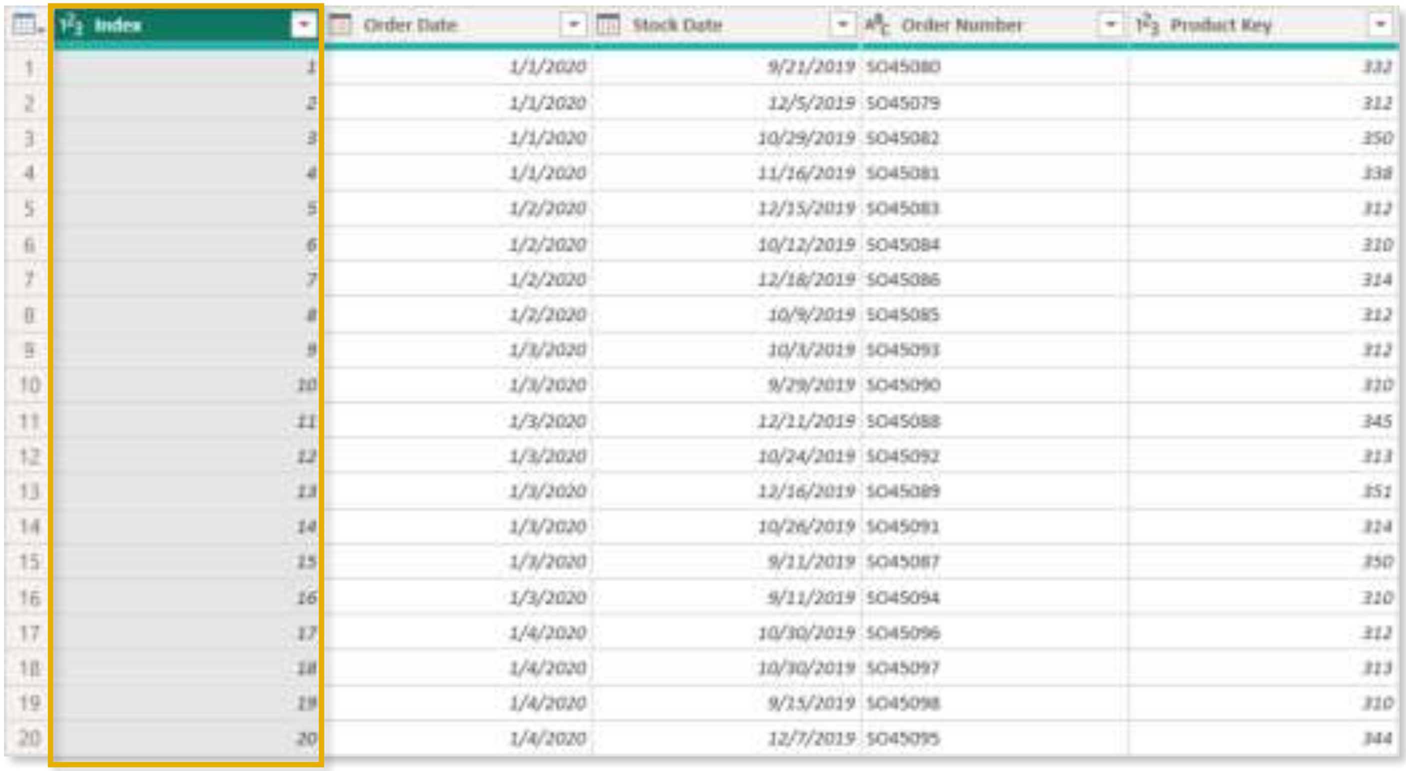
Month Name	Month Number	Start of Year	Year
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020
January	1	1/1/2020	2020

PROPERTIES
Name
Calendar Lookup
All Properties

APPLIED STEPS
- Source
- Promoted Headers
- Changed Type
- Inserted Day Name
- Inserted Start of Week
- Inserted Start of Month
- Inserted Month Name
- Inserted Start of Year
- Inserted Year
- Inserted Month
- Renamed Columns

INDEX COLUMNS



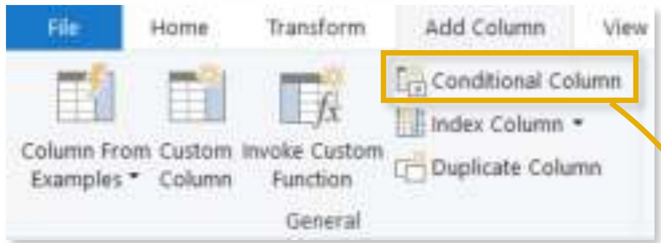


	Index	Order Date	Stock Date	Order Number	Product Key
1	1	1/1/2020	9/21/2019	SO45080	332
2	2	1/1/2020	12/5/2019	SO45079	312
3	3	1/1/2020	10/29/2019	SO45082	350
4	4	1/1/2020	11/16/2019	SO45081	338
5	5	1/2/2020	12/15/2019	SO45083	312
6	6	1/2/2020	10/12/2019	SO45084	310
7	7	1/2/2020	12/18/2019	SO45086	314
8	8	1/2/2020	10/9/2019	SO45085	312
9	9	1/3/2020	10/3/2019	SO45093	312
10	10	1/3/2020	9/29/2019	SO45090	310
11	11	1/3/2020	12/11/2019	SO45088	345
12	12	1/3/2020	10/24/2019	SO45092	313
13	13	1/3/2020	12/16/2019	SO45089	351
14	14	1/3/2020	10/26/2019	SO45091	314
15	15	1/3/2020	9/11/2019	SO45087	350
16	16	1/3/2020	9/11/2019	SO45094	310
17	17	1/4/2020	10/30/2019	SO45096	312
18	18	1/4/2020	10/30/2019	SO45097	313
19	19	1/4/2020	9/15/2019	SO45098	310
20	20	1/4/2020	12/7/2019	SO45095	344

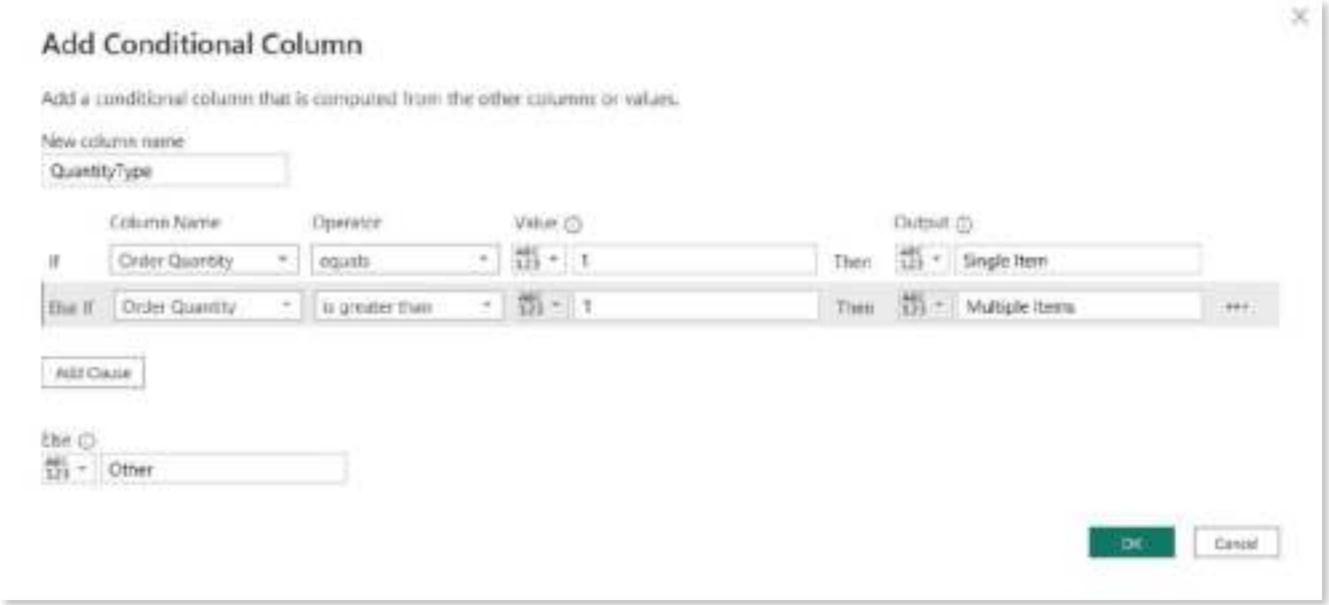
Index Columns contain a list of sequential values that can be used to identify each unique row in a table (*typically starting from 0 or 1*)

These are often used to create **unique IDs** that can be used to form relationships between tables (*more on that later!*)

CONDITIONAL COLUMNS



Conditional Columns allow you to define new fields based on logical rules and conditions (IF/THEN statements)



Add Conditional Column

Add a conditional column that is computed from the other columns or values.

New column name: QuantityType

	Column Name	Operator	Value		Output
If	Order Quantity	equals	1	Then	Single Item
Else If	Order Quantity	is greater than	1	Then	Multiple Items

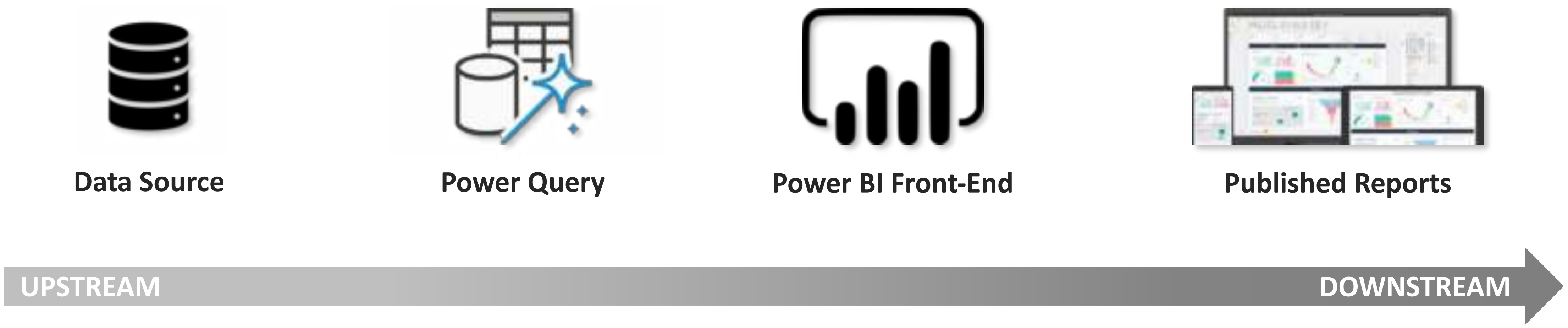
Else: Other

Here we're creating a conditional column named **Quantity Type**, which is based on **Order Quantity**:

- *If Order Quantity* ***=1****, Quantity Type = "****Single Item****"*
- *Else If Order Quantity* ***>1****, Quantity Type = "****Multiple Items****"*
- *Else; Quantity Type = "****Other****"*

CALCULATED COLUMN BEST PRACTICES

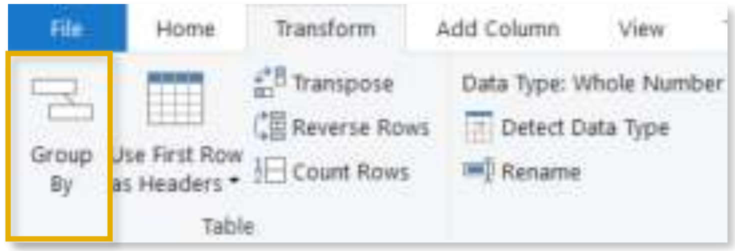
As a best practice, table transformations and column calculations should ideally happen **as close to the original data source as possible**, to optimize performance and speed



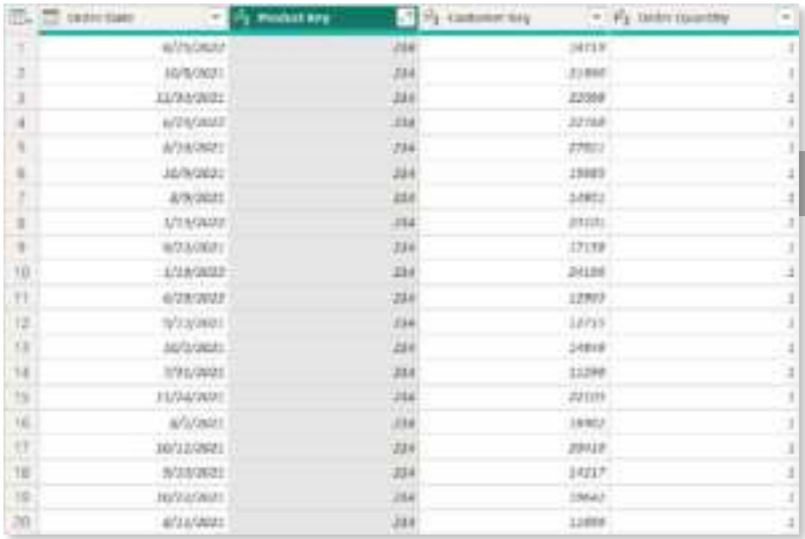
HEY THIS IS IMPORTANT!

This is not a strict rule or requirement but can significantly impact performance for very large or complex data models. Where you define calculations often depends on several factors *(accessibility, complexity, business requirements, etc.)*, so we will practice creating columns using both Power Query and the Power BI front-end (DAX) throughout this course

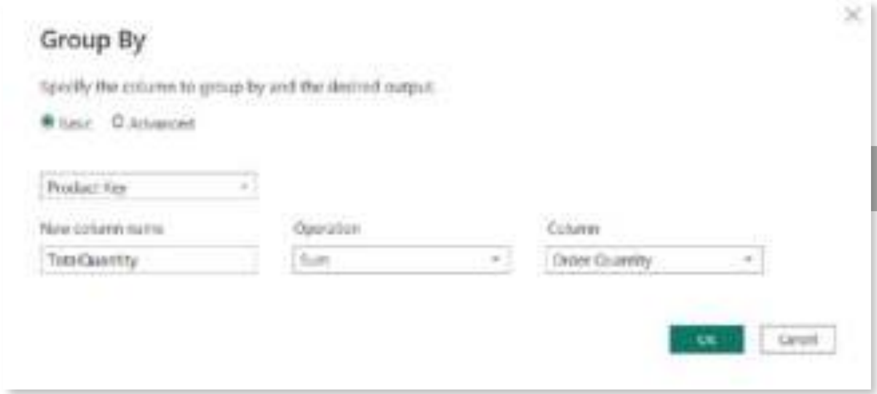
GROUPING & AGGREGATING



Group By allows you to aggregate data at a different level or “grain” (*i.e. group daily records into monthly, aggregate transactions by store, etc.*)



	Order Date	Product Key	Customer Key	Order Quantity
1	6/15/2022	214	14719	1
2	10/8/2021	214	21800	1
3	11/11/2021	214	22008	1
4	6/24/2022	214	22168	1
5	8/18/2021	214	27811	1
6	10/9/2021	214	15683	1
7	8/9/2021	214	14851	1
8	1/13/2022	214	21131	1
9	8/23/2021	214	17139	1
10	1/18/2022	214	24128	1
11	6/28/2022	214	12903	1
12	9/13/2021	214	12715	1
13	10/2/2021	214	14818	1
14	7/31/2021	214	11298	1
15	11/24/2021	214	22109	1
16	8/2/2021	214	19902	1
17	10/12/2021	214	29018	1
18	9/23/2021	214	14217	1
19	10/22/2021	214	19642	1
20	8/11/2021	214	11888	1



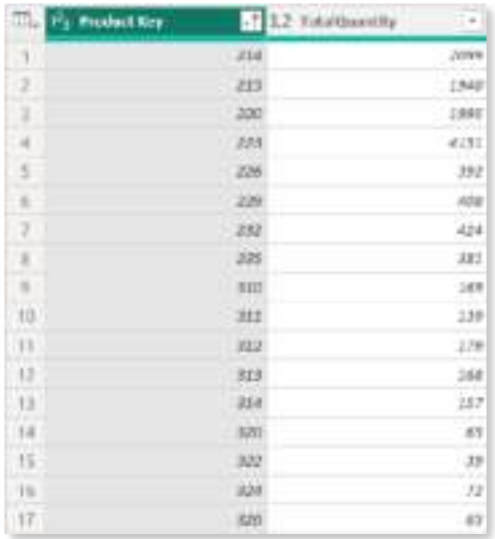
Group By

Specify the column to group by and the desired output.

[✓ Basic] Advanced

Product Key

New column name: TotalQuantity
Operation: Sum
Column: Order Quantity

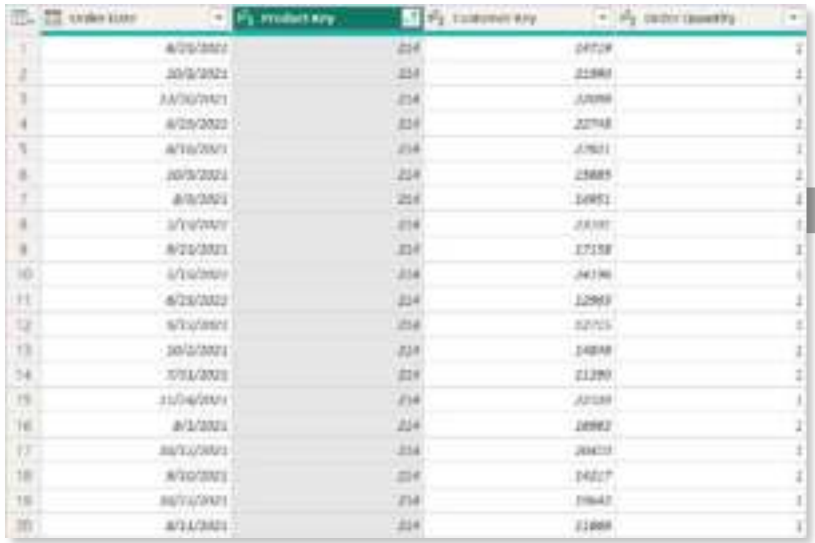


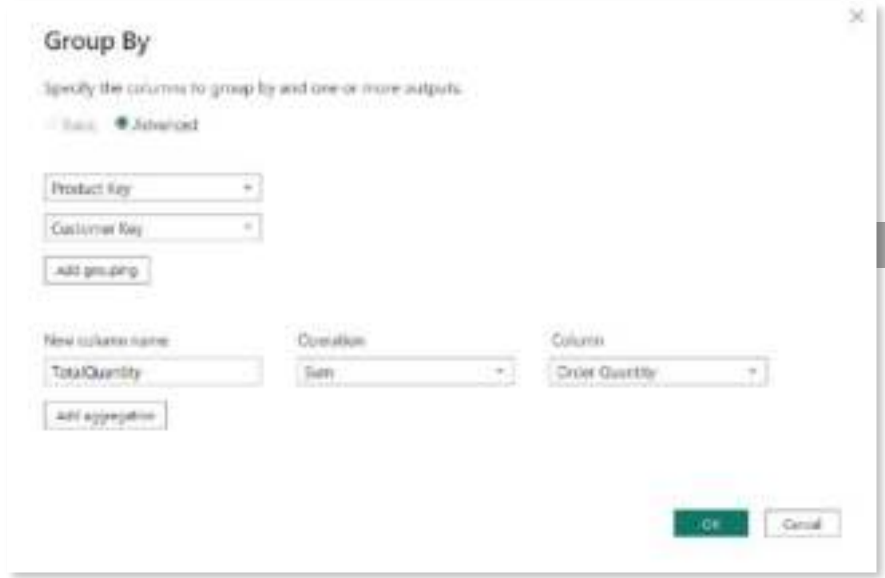
	Product Key	TotalQuantity
1	214	2099
2	215	1940
3	220	1985
4	223	4151
5	226	392
6	229	408
7	232	424
8	235	383
9	310	169
10	311	139
11	312	179
12	313	168
13	314	157
14	320	85
15	322	39
16	324	72
17	326	63

Here we’re transforming a daily, transaction-level table into a summary of ***Total Quantity*** *by* ***Product Key***

NOTE: *Any fields not specified in the Group By settings are lost*

GROUPING & AGGREGATING





Group By

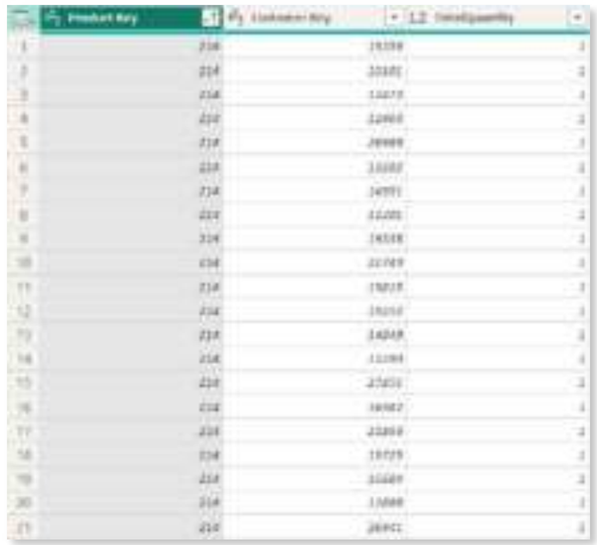
Specify the columns to group by and one or more outputs.

Basic [✓ Advanced]

Product Key

Customer Key

New column name	Operation	Column
TotalQuantity	Sum	Order Quantity

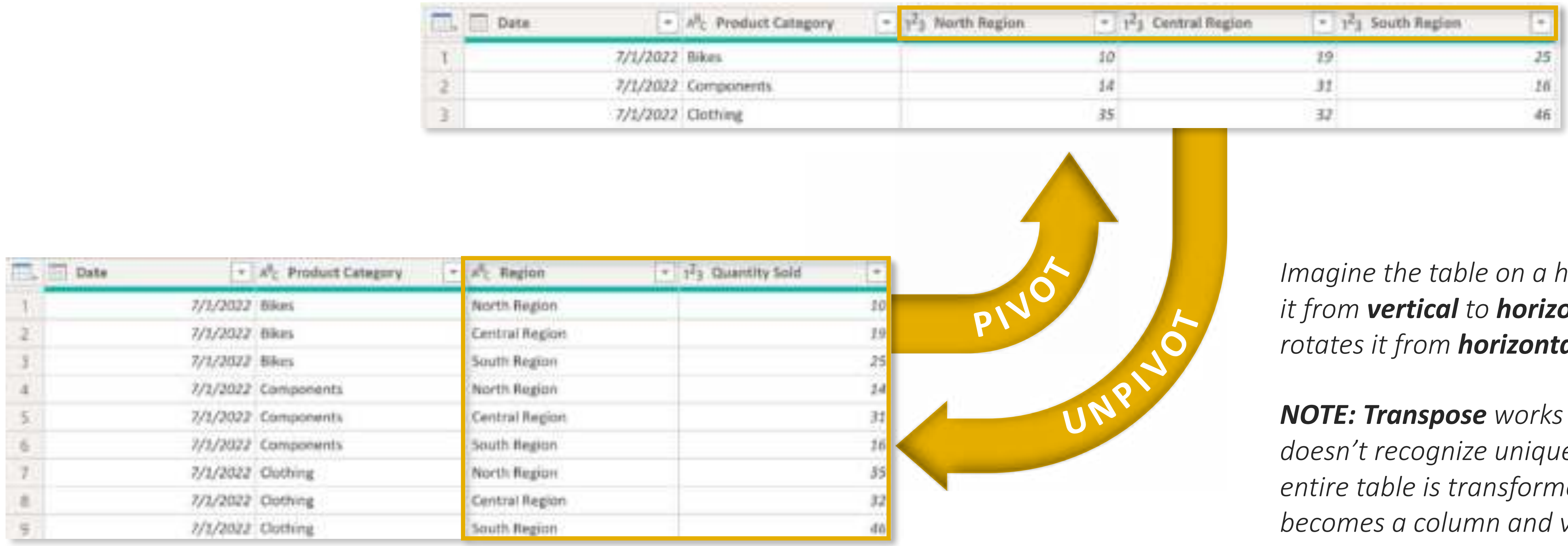


*This time we're transforming the daily, transaction-level table into a summary of **Total Quantity** grouped by both **Product Key** and **Customer Key** (using the "Advanced" option)*

***NOTE:** This is like creating a PivotTable in Excel and pulling in **Sum of Order Quantity** with **Product Key** and **Customer Key** as row labels*

PIVOTING & UNPIVOTING

Pivoting describes the process of turning **distinct row values into columns**, and **unpivoting** describes the process of turning **distinct columns into rows**



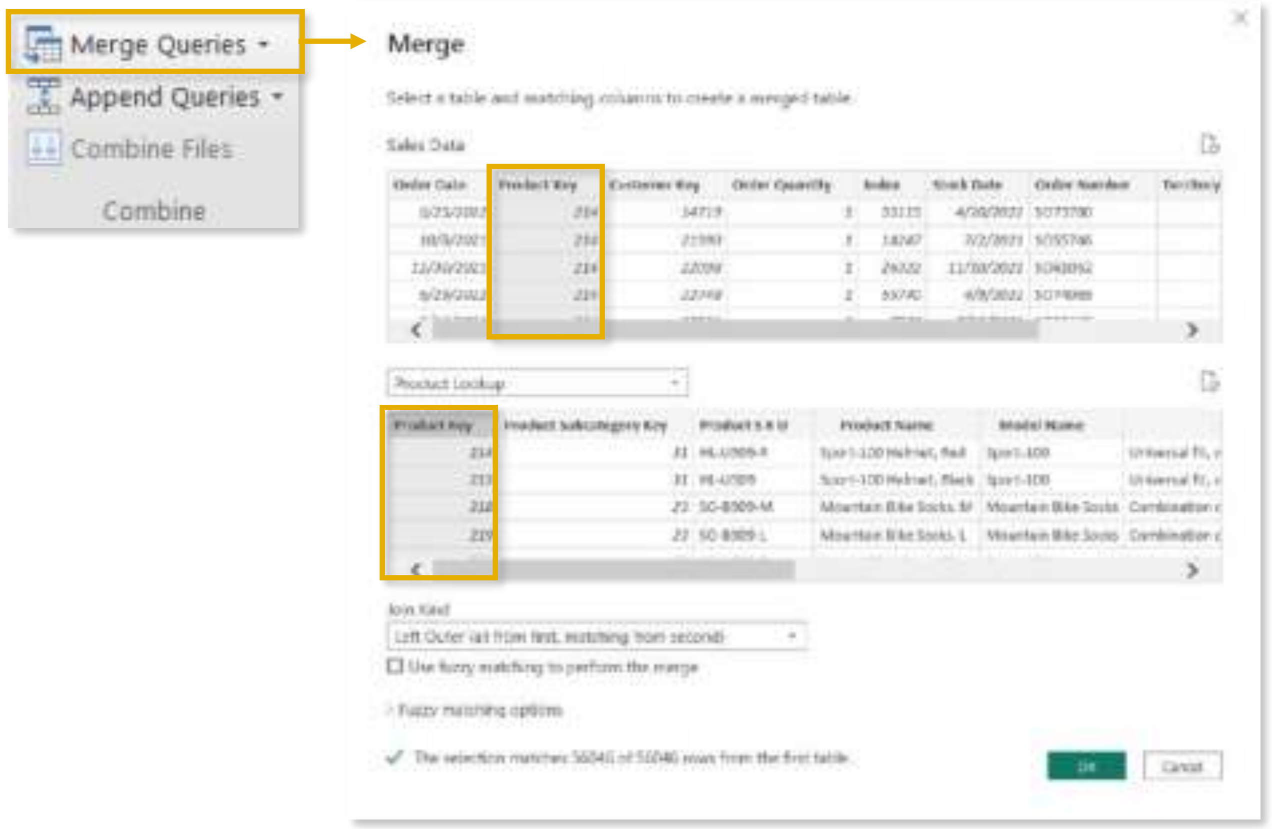
	Date	Product Category	North Region	Central Region	South Region
1	7/1/2022	Bikes	10	19	25
2	7/1/2022	Components	14	31	16
3	7/1/2022	Clothing	35	32	46

	Date	Product Category	Region	Quantity Sold
1	7/1/2022	Bikes	North Region	10
2	7/1/2022	Bikes	Central Region	19
3	7/1/2022	Bikes	South Region	25
4	7/1/2022	Components	North Region	14
5	7/1/2022	Components	Central Region	31
6	7/1/2022	Components	South Region	16
7	7/1/2022	Clothing	North Region	35
8	7/1/2022	Clothing	Central Region	32
9	7/1/2022	Clothing	South Region	46

*Imagine the table on a hinge; **pivoting** rotates it from **vertical** to **horizontal**, and **unpivoting** rotates it from **horizontal** to **vertical***

***NOTE: Transpose** works very similarly, but doesn't recognize unique values; instead, the entire table is transformed so that each row becomes a column and vice versa*

MERGING QUERIES



Merge

Select a table and matching columns to create a merged table

Sales Data

Order Date	Product Key	Customer Key	Order Quantity	Index	Stock Date	Order Number	Territory
1/25/2022	214	14719	1	33115	4/20/2022	SO73700	
10/3/2021	214	21593	1	14247	7/2/2021	SO55746	
11/30/2021	214	22096	1	26022	11/30/2021	SO63052	
9/29/2022	219	22748	1	55740	4/8/2022	SO79888	

Product Lookup

Product Key	Product Subcategory Key	Product SKU	Product Name	Model Name	
214	31	HL-U509-R	Sport-100 Helmet, Red	Sport-100	Universal fit,
213	31	HL-U509	Sport-100 Helmet, Black	Sport-100	Universal fit,
218	23	SO-B909-M	Mountain Bike Socks, M	Mountain Bike Socks	Combination
219	23	SO-B909-L	Mountain Bike Socks, L	Mountain Bike Socks	Combination

Join Kind

Left Outer (all from first, matching from second)

☐ Use fuzzy matching to perform the merge

› Fuzzy matching options

✓ The selection matches 56046 of 56046 rows from the first table.

OK Cancel

Merging queries allows you to **join tables** based on a common column (like a lookup in Excel)

*In this case we're merging the **Sales Data** table with the **Product Lookup** table, which share a common **Product Key** column*

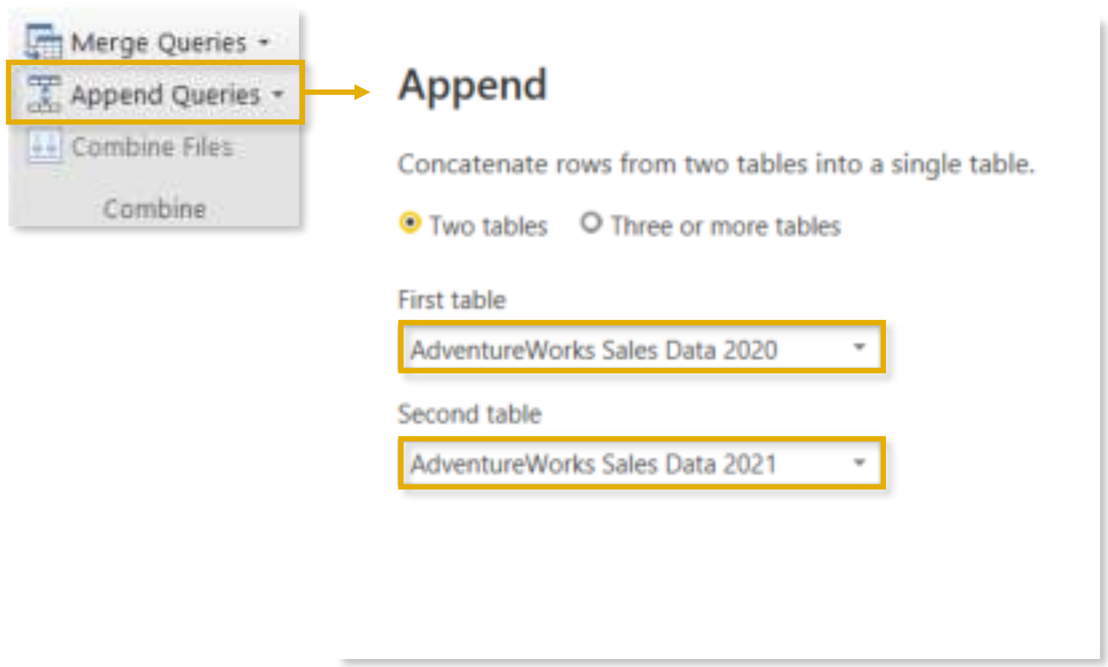
***NOTE**: Merging **adds columns** to an existing table/query*

HEY THIS IS IMPORTANT!

Just because you can merge tables, doesn't mean you should!

In many cases, it's better to keep tables separate and define **relationships** between them in the data model (*more on that soon!*)

APPENDING QUERIES



Append

Concatenate rows from two tables into a single table.

[✓ Two tables] Three or more tables

First table
AdventureWorks Sales Data 2020

Second table
AdventureWorks Sales Data 2021

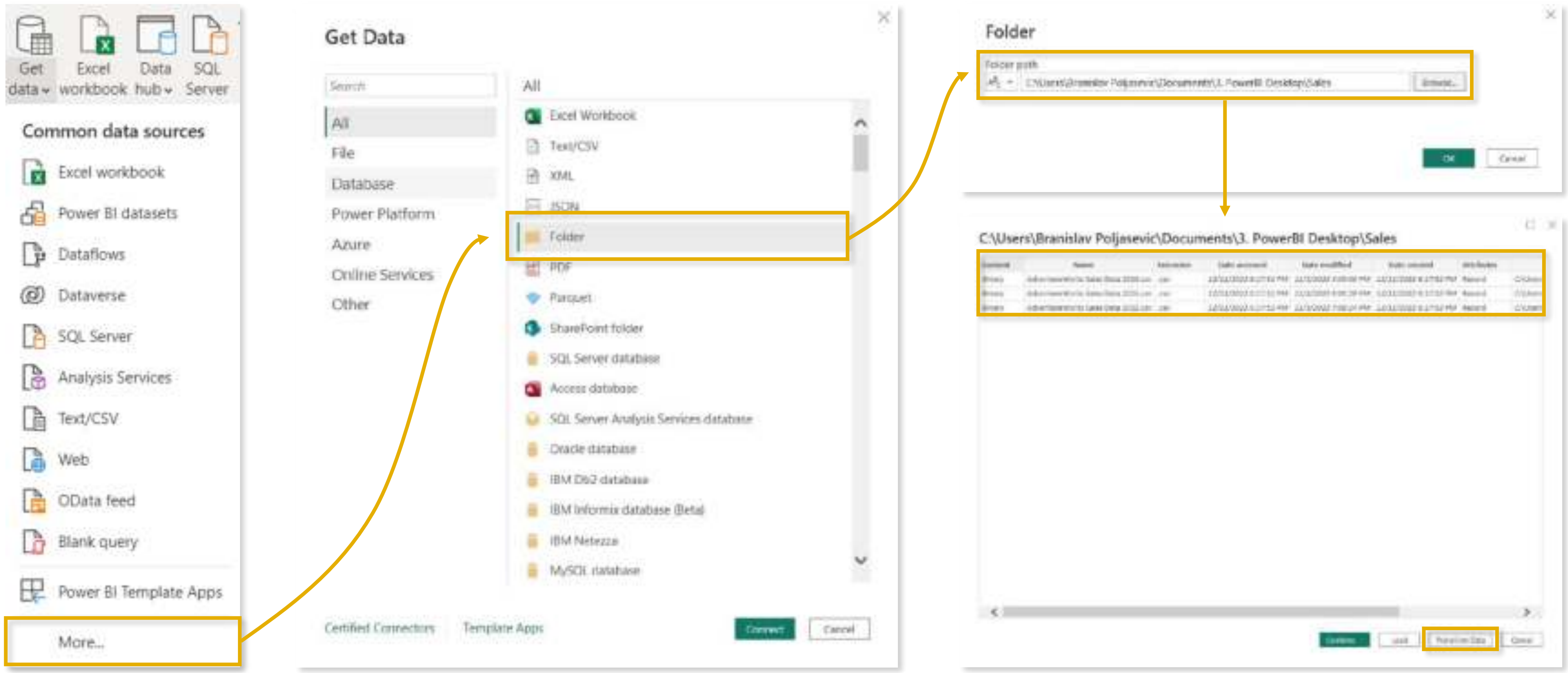
Appending queries allows you to **combine** or **stack** tables sharing the exact same column structure and data types

*Here we're appending the **AdventureWorks Sales 2020** table to the **AdventureWorks Sales 2021** table, which is valid since they share identical table structures*

***NOTE:** Appending **adds rows** to an existing table/query*

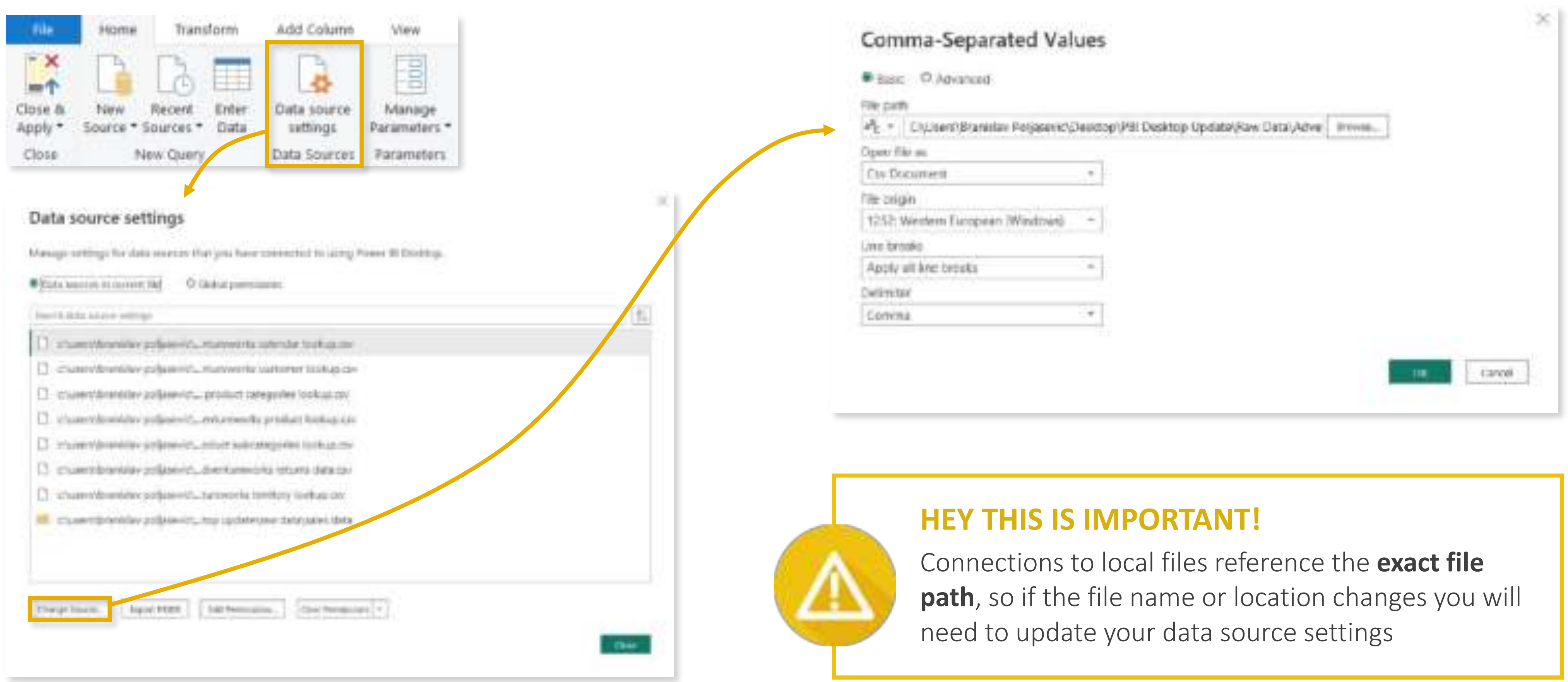
PRO TIP: *Use the **Folder** option (Get Data > More > Folder) to **append all files within a specified folder** (assuming they share the same structure); as you add new files, simply refresh the query and they will automatically append!*

PRO TIP: APPENDING FILES FROM A FOLDER



DATA SOURCE SETTINGS

Data Source Settings allow you to manage existing data connections, file paths and permissions



Comma-Separated Values

[✓ Basic] Advanced

File path: C:\Users\Branislav Poljasevic\Desktop\PBI Desktop Update\Raw Data\Adve

Open File as: Csv Document

File origin: 1252: Western European (Windows)

Line breaks: Apply all line breaks

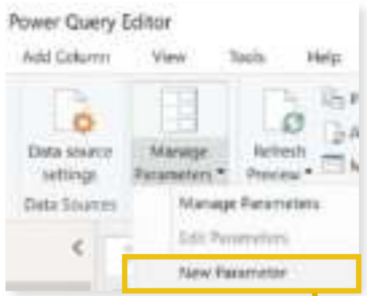
Delimiter: Comma

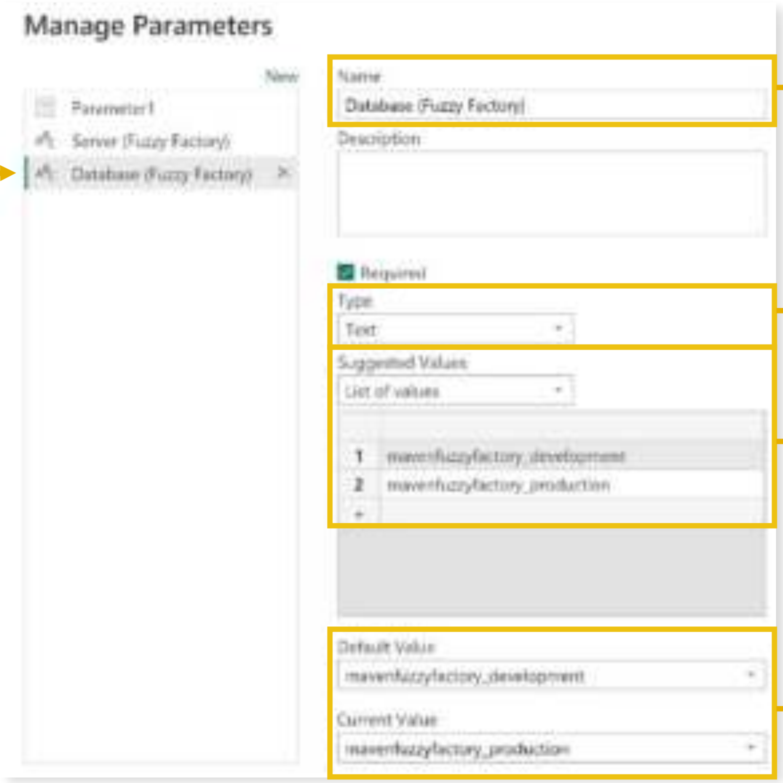
HEY THIS IS IMPORTANT!

Connections to local files reference the **exact file path**, so if the file name or location changes you will need to update your data source settings

PRO TIP: DATA SOURCE PARAMETERS

Use **parameters** to dynamically manage and update connection paths in the Power Query editor





Name
Database (Fuzzy Factory)

Description

✓ Required

Type
Text

Suggested Values
List of values

1	mavenfuzzyfactory_development
2	mavenfuzzyfactory_production
*	

Default Value
mavenfuzzyfactory_development

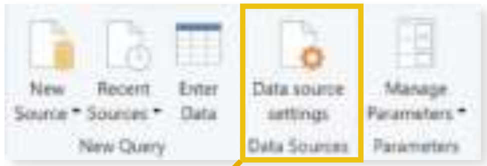
Current Value
mavenfuzzyfactory_production

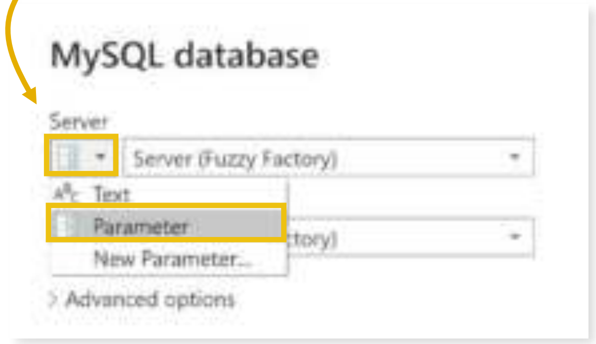
*Parameter **name***
(Name of the query/table)

*Parameter **type***
(Any value, text, date, etc.)

*Parameter **value***
(Any value, list, query)

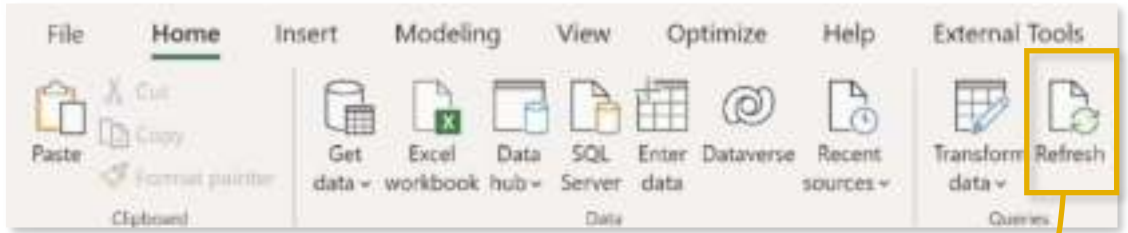
*Parameter **type***
(Default & current)





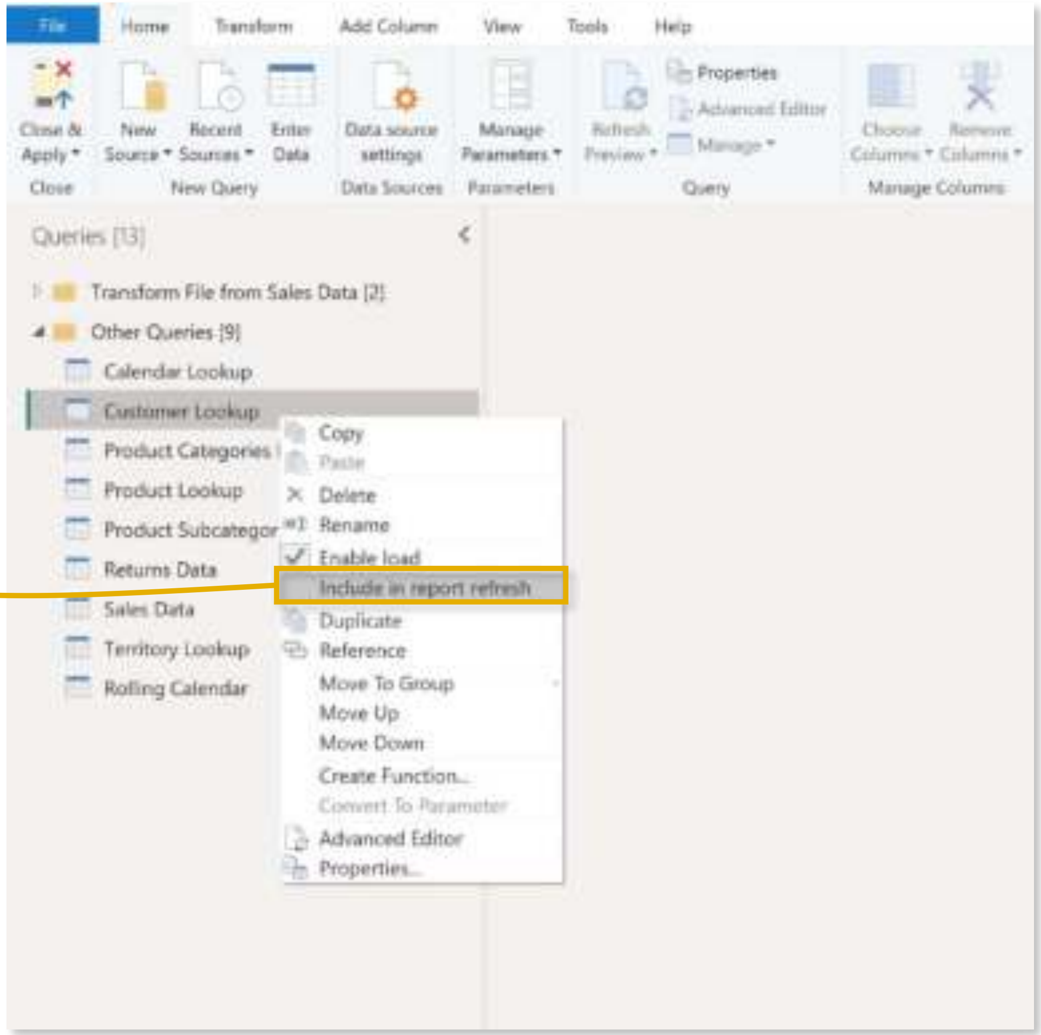
Update** Server & Database connection text values **with parameters

REFRESHING QUERIES



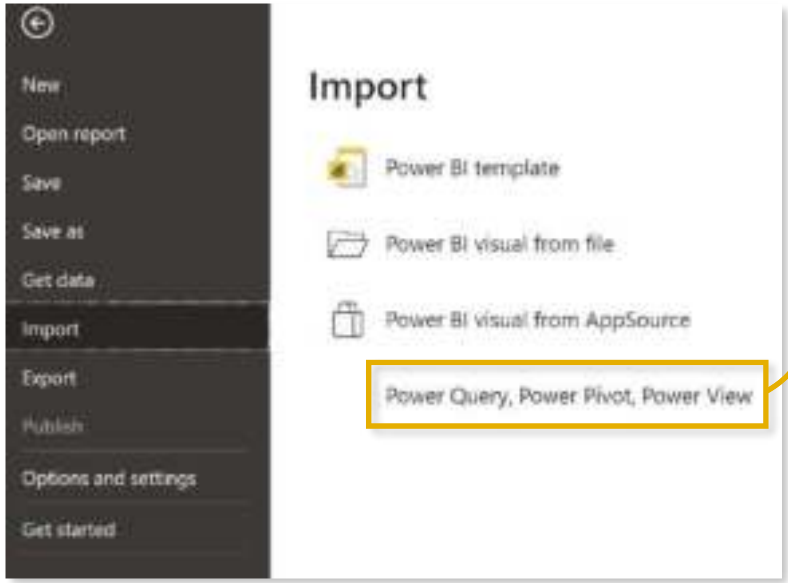
By default, ***all queries*** will refresh when you use the **Refresh** command from the **Home** tab

From the Query Editor, uncheck **Include in report refresh** to exclude individual queries from the refresh



PRO TIP: Exclude queries from refresh that don't change often (like lookups or static data tables)

PRO TIP: IMPORTING EXCEL MODELS



Already have a fully-built model in Excel?

You can import models built in Excel directly into Power BI Desktop using: ***Import > Power Query, Power Pivot, Power View***

Imported models retain the following:

- Data source **connections** and **queries**
- Query editing procedures and **applied steps**
- Table **relationships**, **hierarchies**, **field settings**, etc.
- All **calculated columns** and **DAX measures**

PRO TIP: If you are more comfortable working in Excel, build your models there first then import to Power BI!

**In older versions of Power BI, this import option may be called "Excel Workbook Contents"*

POWER QUERY BEST PRACTICES

Get organized before connecting and loading data

- *Define clear and intuitive table/query names from the start, and establish an organized file/folder structure if you are working with local flat files to avoid changes to file names or paths*

Disable report refresh for any static data sources

- *There's no need to constantly refresh data sources that don't change, like lookups or static data tables*

When working with large tables, only load the data you need

- *Don't include hourly data when you only need daily, or transaction-level data when only need a product-level summary (extra data will only slow your report down!)*

CREATING A DATA MODEL

CREATING A DATA MODEL

In this section we'll cover **foundational data modeling topics** like normalization, fact and dimension tables, primary and foreign keys, relationship cardinality and filter flow

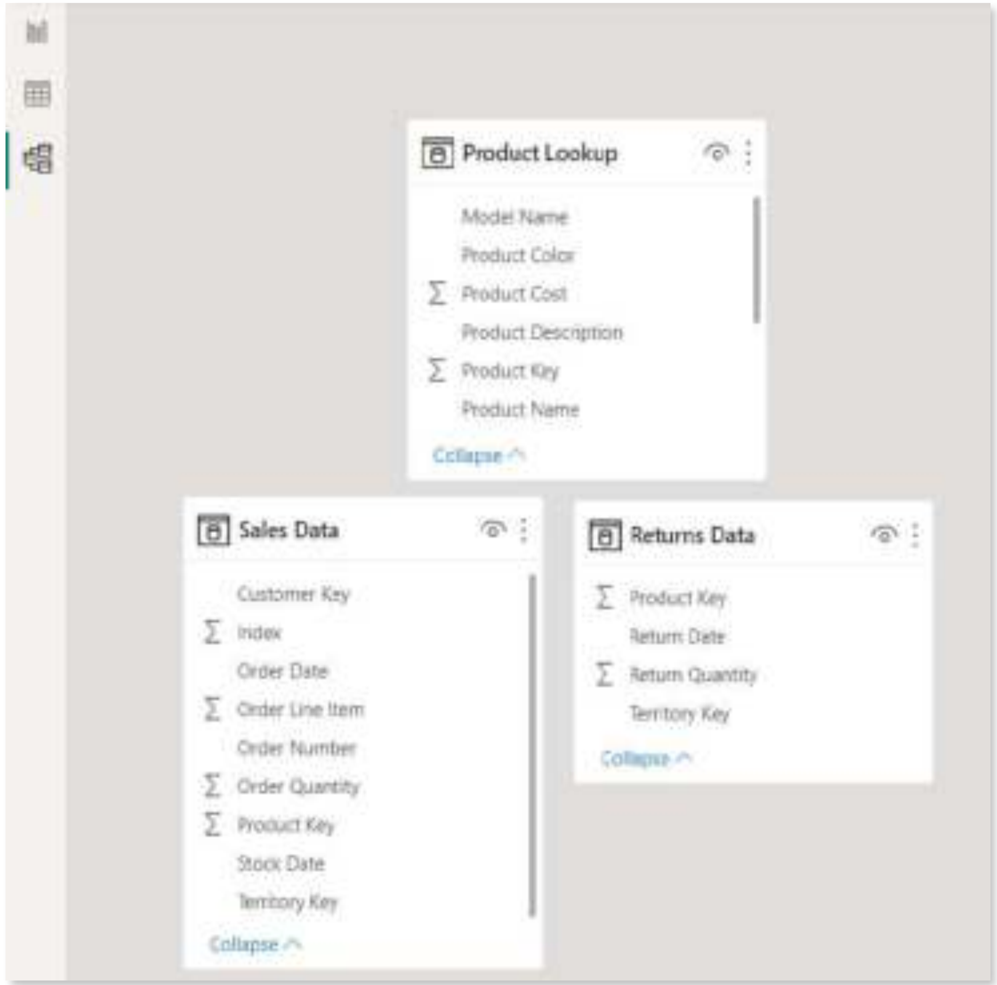
TOPICS WE'LL COVER:

- Data Modeling 101
- Normalization
- Facts & Dimensions
- Primary & Foreign Keys
- Cardinality
- Filter Flow
- Common Schemas
- Hierarchies

GOALS FOR THIS SECTION:

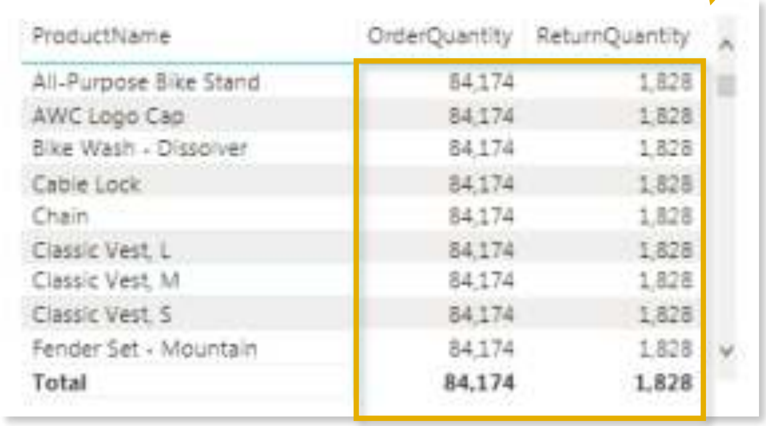
- Understand the basic principles of data modeling, including normalization, fact & dimension tables and common schemas
- Create table relationships using primary and foreign keys, and discuss different types of relationship cardinality
- Configure report filters and trace filter context as it flows between related tables in the model
- Explore data modeling options like hierarchies, data categories and hidden fields

WHAT IS A DATA MODEL?



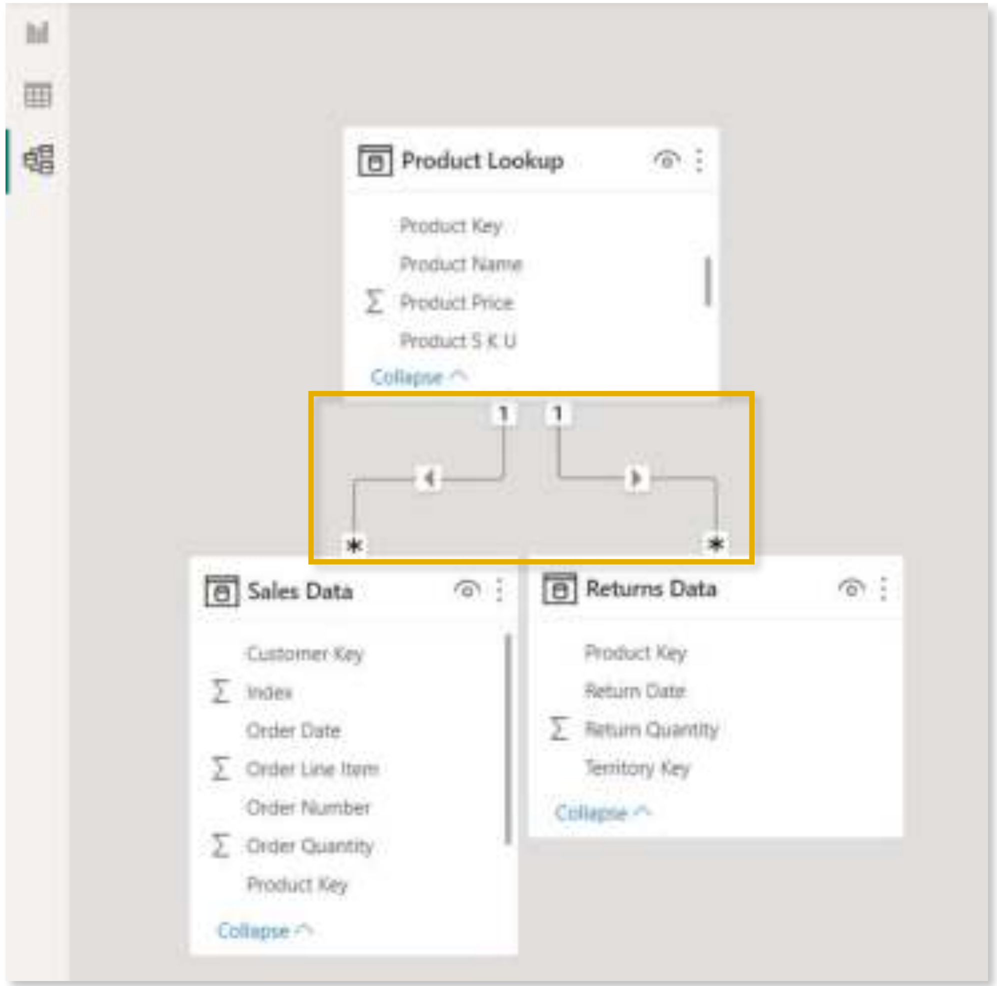
This **IS NOT** a data model

- This is a collection of independent tables, which share no connections or relationships
- If you tried to visualize **Orders** and **Returns** by **Product**, this is what you'd get



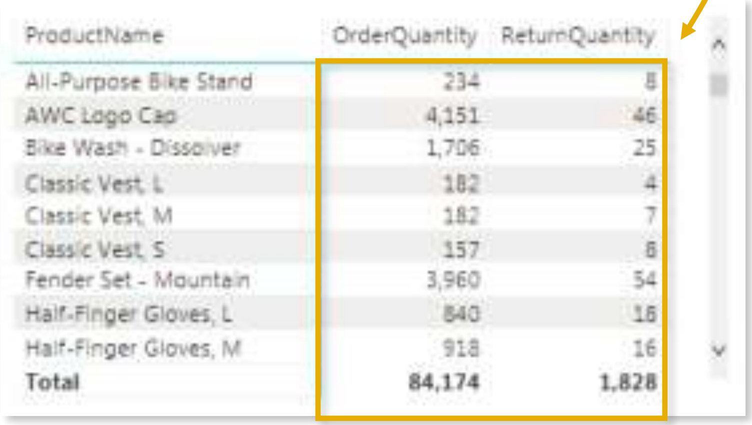
ProductName	OrderQuantity	ReturnQuantity
All-Purpose Bike Stand	84,174	1,828
AWC Logo Cap	84,174	1,828
Bike Wash - Dissolver	84,174	1,828
Cable Lock	84,174	1,828
Chain	84,174	1,828
Classic Vest, L	84,174	1,828
Classic Vest, M	84,174	1,828
Classic Vest, S	84,174	1,828
Fender Set - Mountain	84,174	1,828
Total	**84,174**	**1,828**

WHAT IS A DATA MODEL?



This **IS** a data model! 😃

- The tables are connected via relationships, based on a common field (Product Key)
- Now **Sales** and **Returns** data can be filtered using fields from the **Product Lookup** table!



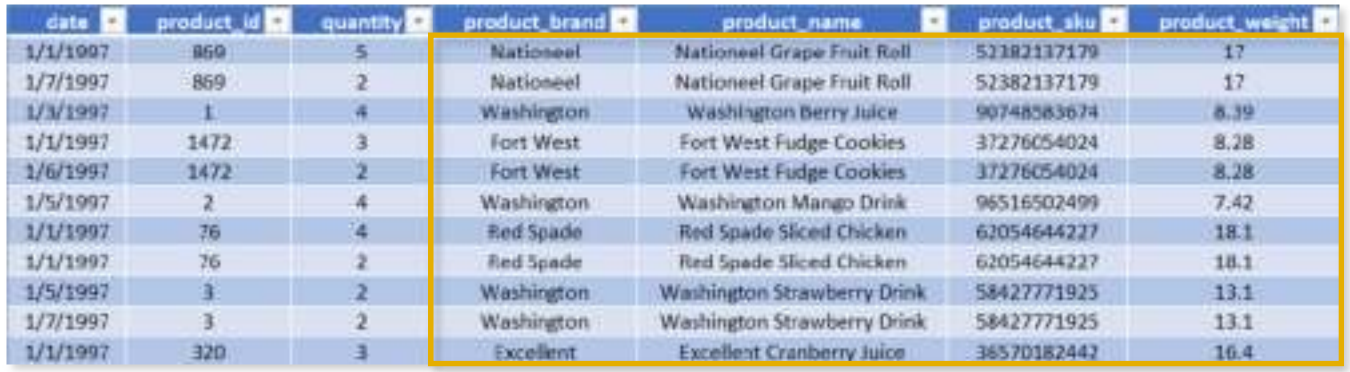
ProductName	OrderQuantity	ReturnQuantity
All-Purpose Bike Stand	234	8
AWC Logo Cap	4,151	46
Bike Wash - Dissolver	1,706	25
Classic Vest, L	182	4
Classic Vest, M	182	7
Classic Vest, S	157	8
Fender Set - Mountain	3,960	54
Half-Finger Gloves, L	840	16
Half-Finger Gloves, M	918	16
Total	**84,174**	**1,828**

DATABASE NORMALIZATION

Normalization is the process of organizing the tables and columns in a relational database to reduce redundancy and preserve data integrity. It’s commonly used to:

- **Eliminate redundant data** to decrease table sizes and improve processing speed & efficiency
- **Minimize errors and anomalies** from data modifications (inserting, updating or deleting records)
- **Simplify queries** and structure the database for meaningful analysis

In a normalized database, each table should serve a **distinct** and **specific** purpose (*i.e. product information, transaction records, customer attributes, store details, etc.*)



date	product_id	quantity	product_brand	product_name	product_sku	product_weight
1/1/1997	869	5	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/7/1997	869	2	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/3/1997	1	4	Washington	Washington Berry Juice	90748583674	8.39
1/1/1997	1472	3	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/6/1997	1472	2	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/5/1997	2	4	Washington	Washington Mango Drink	96516502499	7.42
1/1/1997	76	4	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/1/1997	76	2	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/5/1997	3	2	Washington	Washington Strawberry Drink	58427771925	13.1
1/7/1997	3	2	Washington	Washington Strawberry Drink	58427771925	13.1
1/1/1997	320	3	Excellent	Excellent Cranberry Juice	36570182442	16.4

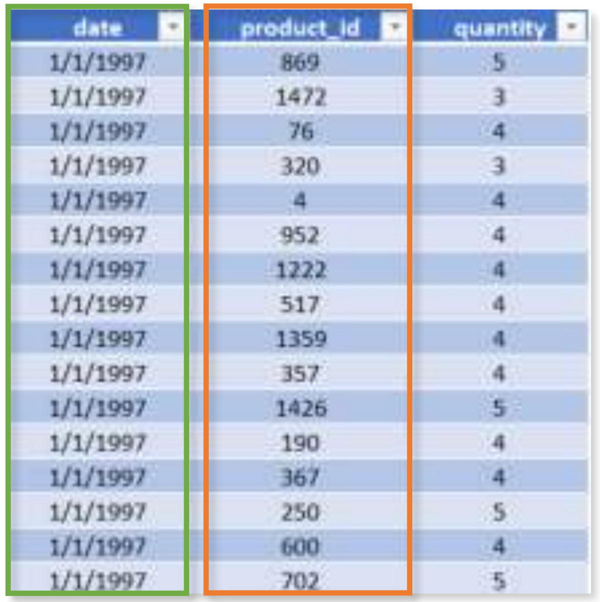
*Models that aren’t normalized contain **redundant, duplicate data**. In this case, all of the product-specific fields could be stored in a separate table containing a unique record for each **product id***

This may not seem critical now, but minor inefficiencies can become major problems at scale!

FACT & DIMENSION TABLES

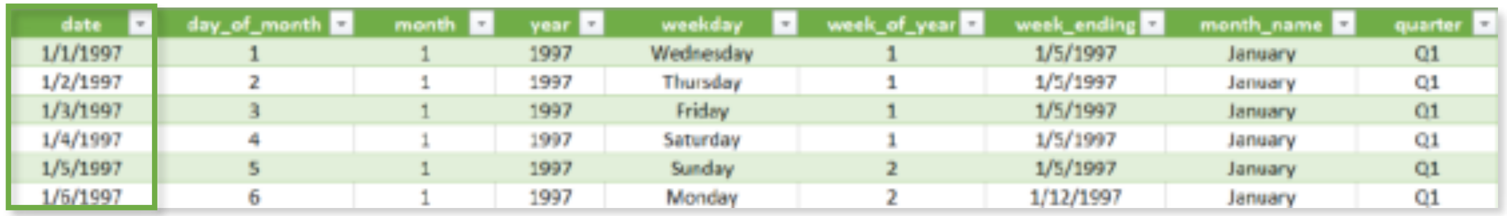
Data models generally contain two types of tables: **fact** (“data”) tables, and **dimension** (“lookup”) tables:

- **Fact tables** contain **numerical values** or metrics used for summarization *(sales, orders, transactions, pageviews, etc.)*
- **Dimension tables** contain **descriptive attributes** used for filtering or grouping *(products, customers, dates, stores, etc.)*



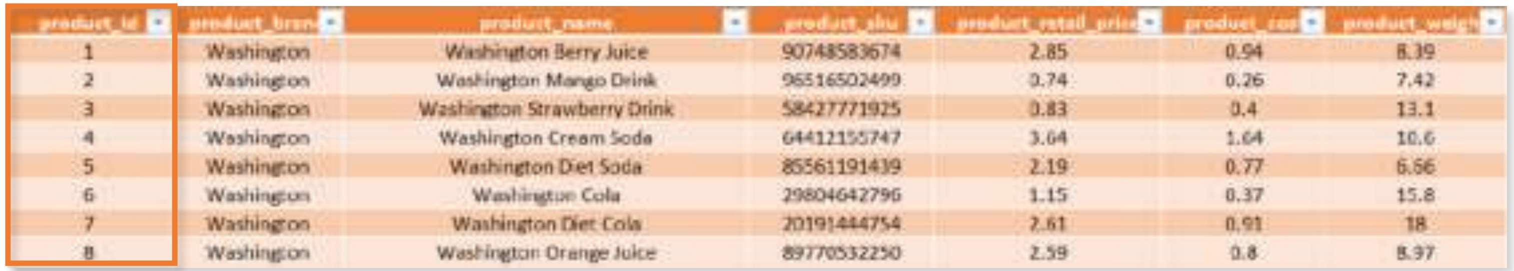
date	product_id	quantity
1/1/1997	869	5
1/1/1997	1472	3
1/1/1997	76	4
1/1/1997	320	3
1/1/1997	4	4
1/1/1997	952	4
1/1/1997	1222	4
1/1/1997	517	4
1/1/1997	1359	4
1/1/1997	357	4
1/1/1997	1426	5
1/1/1997	190	4
1/1/1997	367	4
1/1/1997	250	5
1/1/1997	600	4
1/1/1997	702	5

*This **Fact** table contains **quantity** values, along with **date** and **product_id** fields*



date	day_of_month	month	year	weekday	week_of_year	week_ending	month_name	quarter
1/1/1997	1	1	1997	Wednesday	1	1/5/1997	January	Q1
1/2/1997	2	1	1997	Thursday	1	1/5/1997	January	Q1
1/3/1997	3	1	1997	Friday	1	1/5/1997	January	Q1
1/4/1997	4	1	1997	Saturday	1	1/5/1997	January	Q1
1/5/1997	5	1	1997	Sunday	2	1/5/1997	January	Q1
1/6/1997	6	1	1997	Monday	2	1/12/1997	January	Q1

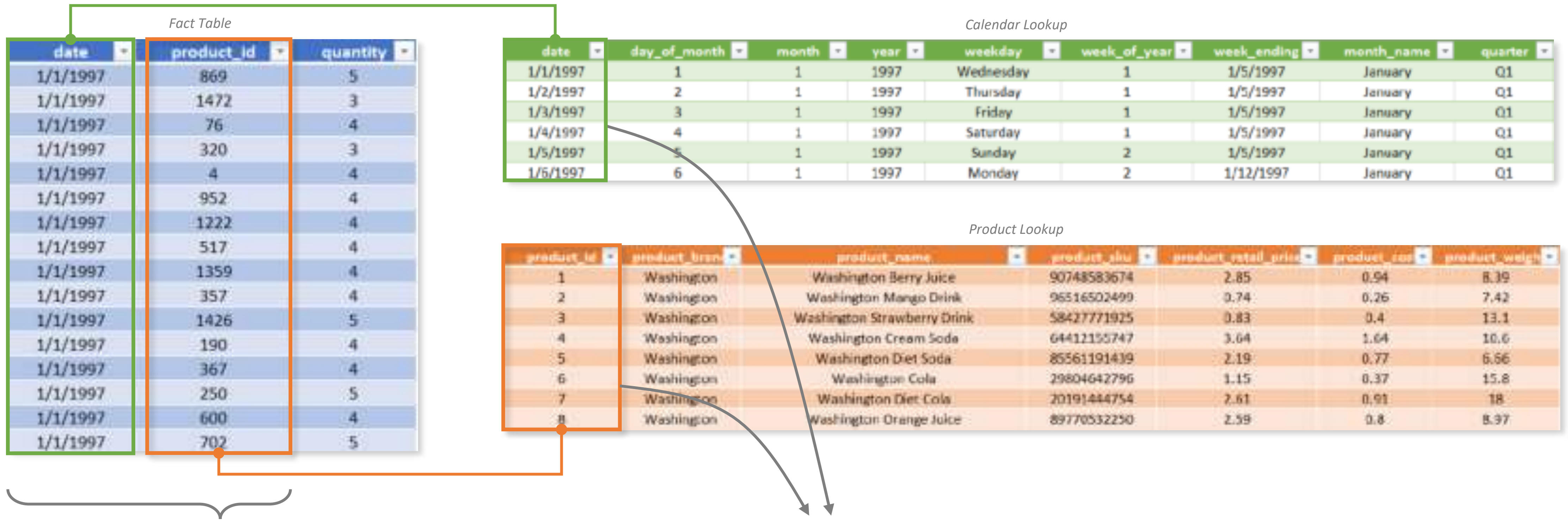
*This **Calendar Lookup** table contains attributes about each **date** (month, year, quarter, etc.)*



product_id	product_bran	product_name	product_sku	product_retail_pric	product_co	product_weigh
1	Washington	Washington Berry Juice	90748583674	2.85	0.94	8.39
2	Washington	Washington Mango Drink	96516502499	0.74	0.26	7.42
3	Washington	Washington Strawberry Drink	58427771925	0.83	0.4	13.1
4	Washington	Washington Cream Soda	64412155747	3.64	1.64	10.6
5	Washington	Washington Diet Soda	85561191439	2.19	0.77	6.66
6	Washington	Washington Cola	29804642796	1.15	0.37	15.8
7	Washington	Washington Diet Cola	20191444754	2.61	0.91	18
8	Washington	Washington Orange Juice	89770532250	2.59	0.8	8.97

*This **Product Lookup** table contains attributes about each **product_id** (brand, SKU, price, etc.)*

PRIMARY & FOREIGN KEYS



Fact Table

date	product_id	quantity
1/1/1997	869	5
1/1/1997	1472	3
1/1/1997	76	4
1/1/1997	320	3
1/1/1997	4	4
1/1/1997	952	4
1/1/1997	1222	4
1/1/1997	517	4
1/1/1997	1359	4
1/1/1997	357	4
1/1/1997	1426	5
1/1/1997	190	4
1/1/1997	367	4
1/1/1997	250	5
1/1/1997	600	4
1/1/1997	702	5

Calendar Lookup

date	day_of_month	month	year	weekday	week_of_year	week_ending	month_name	quarter
1/1/1997	1	1	1997	Wednesday	1	1/5/1997	January	Q1
1/2/1997	2	1	1997	Thursday	1	1/5/1997	January	Q1
1/3/1997	3	1	1997	Friday	1	1/5/1997	January	Q1
1/4/1997	4	1	1997	Saturday	1	1/5/1997	January	Q1
1/5/1997	5	1	1997	Sunday	2	1/5/1997	January	Q1
1/6/1997	6	1	1997	Monday	2	1/12/1997	January	Q1

Product Lookup

product_id	product_bran	product_name	product_sku	product_retail_pric	product_cos	product_weigh
1	Washington	Washington Berry Juice	90748583674	2.85	0.94	8.39
2	Washington	Washington Mango Drink	96516502499	0.74	0.26	7.42
3	Washington	Washington Strawberry Drink	58427771925	0.83	0.4	13.1
4	Washington	Washington Cream Soda	64412155747	3.64	1.64	10.6
5	Washington	Washington Diet Soda	85561191439	2.19	0.77	6.66
6	Washington	Washington Cola	29804642796	1.15	0.37	15.8
7	Washington	Washington Diet Cola	20191444754	2.61	0.91	18
8	Washington	Washington Orange Juice	89770532250	2.59	0.8	8.97

These are **foreign keys** (FK)

*They contain multiple instances of each value, and relate to **primary keys** in dimension tables*

These are **primary keys** (PK)

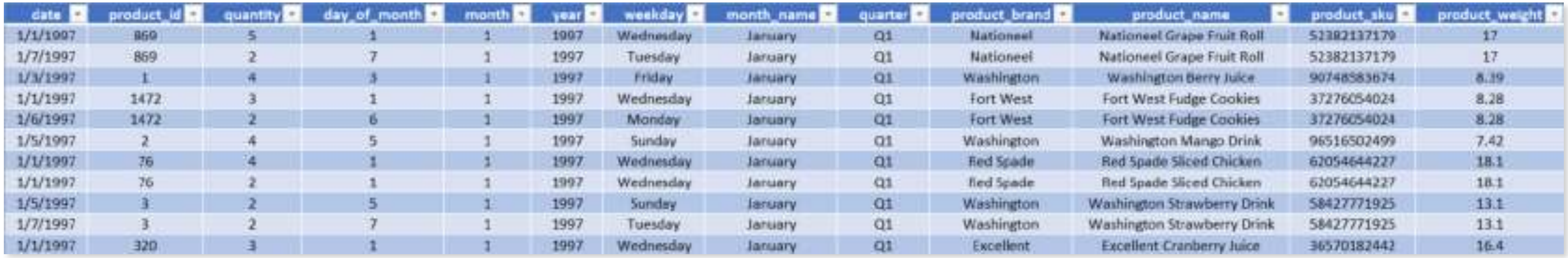
*They uniquely identify each row of the table, and relate to **foreign keys** in fact tables*

RELATIONSHIPS VS. MERGED TABLES

Can't I just merge queries or use lookup functions to ***pull everything into one single table****?*

- Anonymous confused man

Original ***Fact Table*** *fields* | *Attributes from* ***Calendar Lookup*** *table* | *Attributes from* ***Product Lookup*** *table*



date	product_id	quantity	day_of_month	month	year	weekday	month_name	quarter	product_brand	product_name	product_sku	product_weight
1/1/1997	869	5	1	1	1997	Wednesday	January	Q1	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/7/1997	869	2	7	1	1997	Tuesday	January	Q1	Nationeel	Nationeel Grape Fruit Roll	52382137179	17
1/3/1997	1	4	3	1	1997	Friday	January	Q1	Washington	Washington Berry Juice	90748583674	8.39
1/1/1997	1472	3	1	1	1997	Wednesday	January	Q1	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/6/1997	1472	2	6	1	1997	Monday	January	Q1	Fort West	Fort West Fudge Cookies	37276054024	8.28
1/5/1997	2	4	5	1	1997	Sunday	January	Q1	Washington	Washington Mango Drink	96516502499	7.42
1/1/1997	76	4	1	1	1997	Wednesday	January	Q1	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/1/1997	76	2	1	1	1997	Wednesday	January	Q1	Red Spade	Red Spade Sliced Chicken	62054644227	18.1
1/5/1997	3	2	5	1	1997	Sunday	January	Q1	Washington	Washington Strawberry Drink	58427771925	13.1
1/7/1997	3	2	7	1	1997	Tuesday	January	Q1	Washington	Washington Strawberry Drink	58427771925	13.1
1/1/1997	320	3	1	1	1997	Wednesday	January	Q1	Excellent	Excellent Cranberry Juice	36570182442	16.4

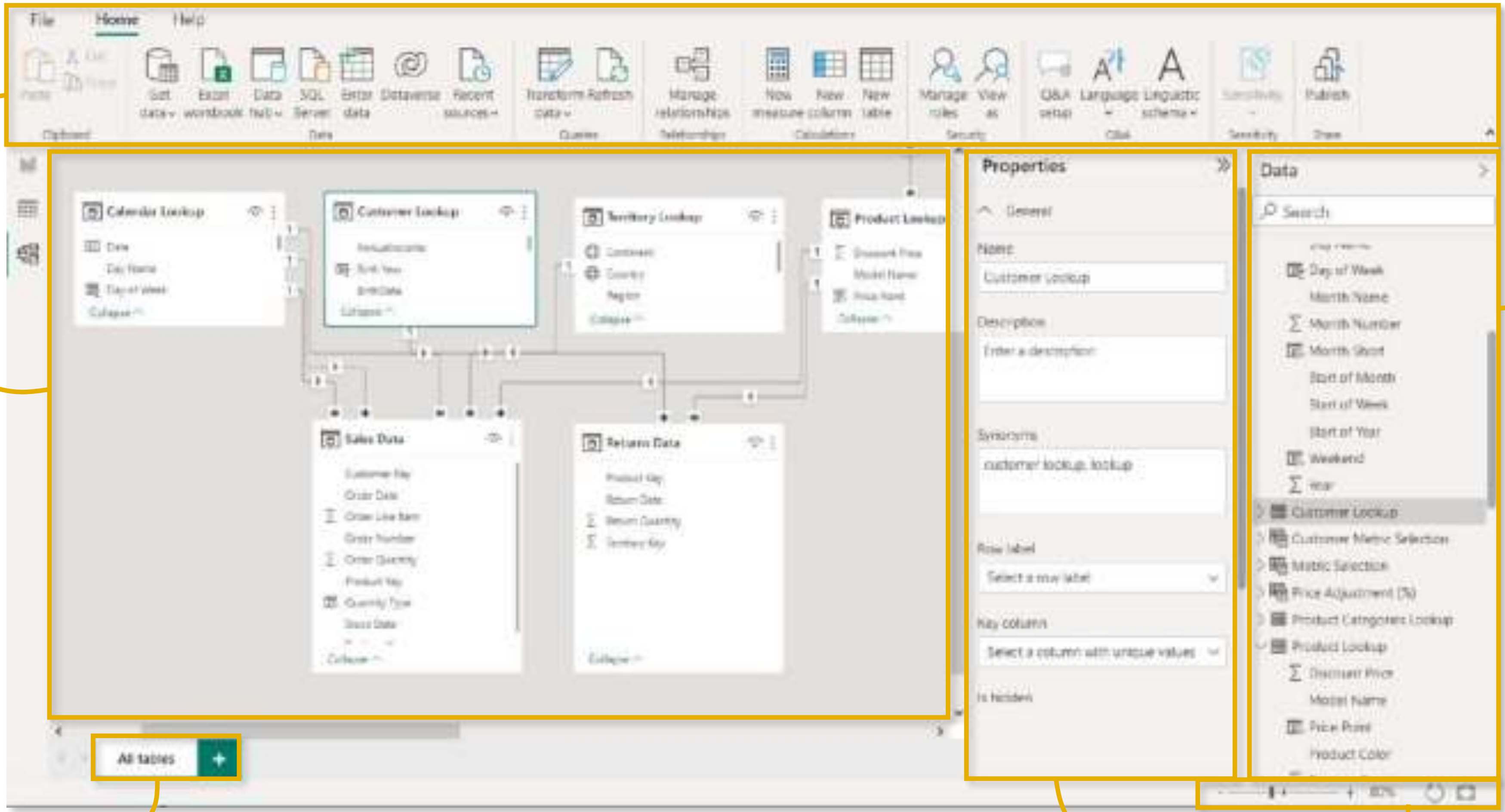
You can, **but it's extremely inefficient!**

- Merging tables creates **redundancy** and often requires **significantly more memory and processing power** to analyze compared to a relational model with multiple small tables

THE MODEL VIEW

Menu Ribbon
(Home, Help)

Model canvas



Data / Field List

Model layout tabs

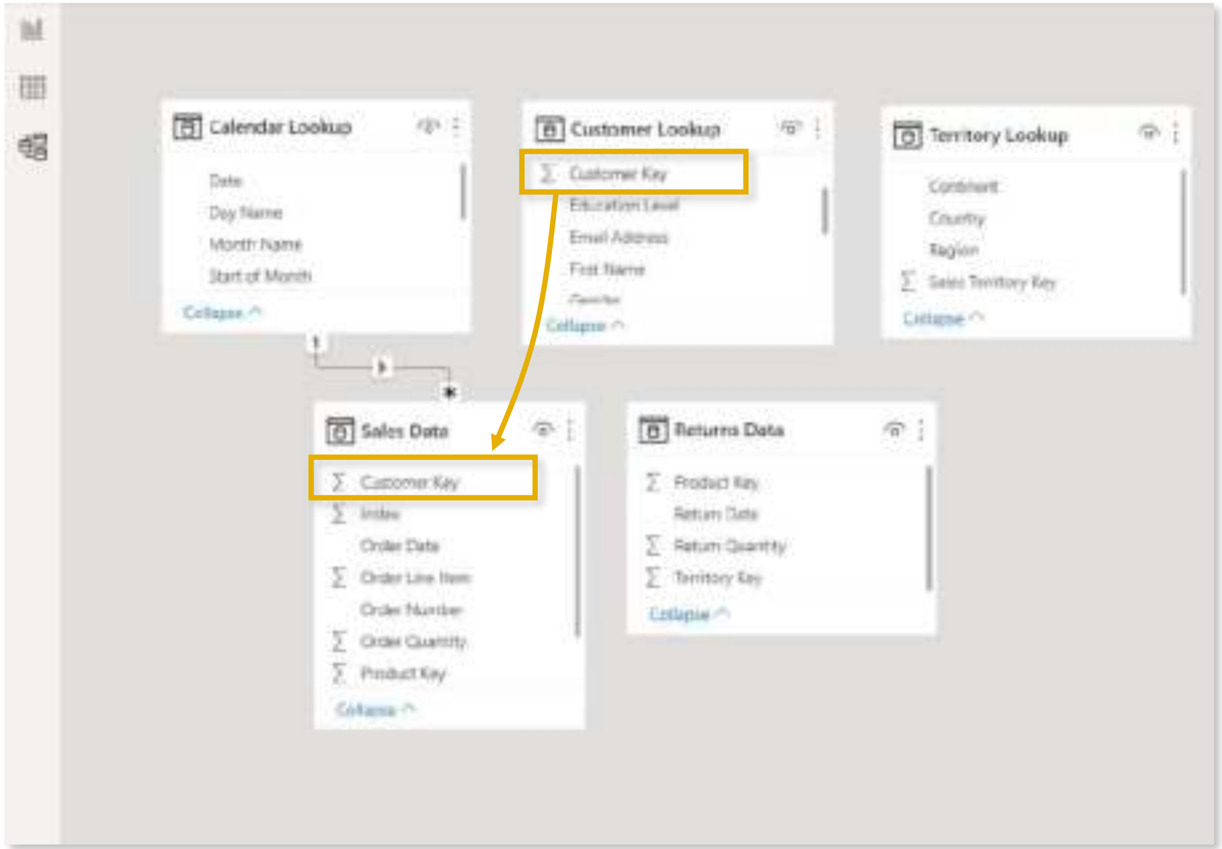
Properties pane
(Name, synonym, format, etc.)

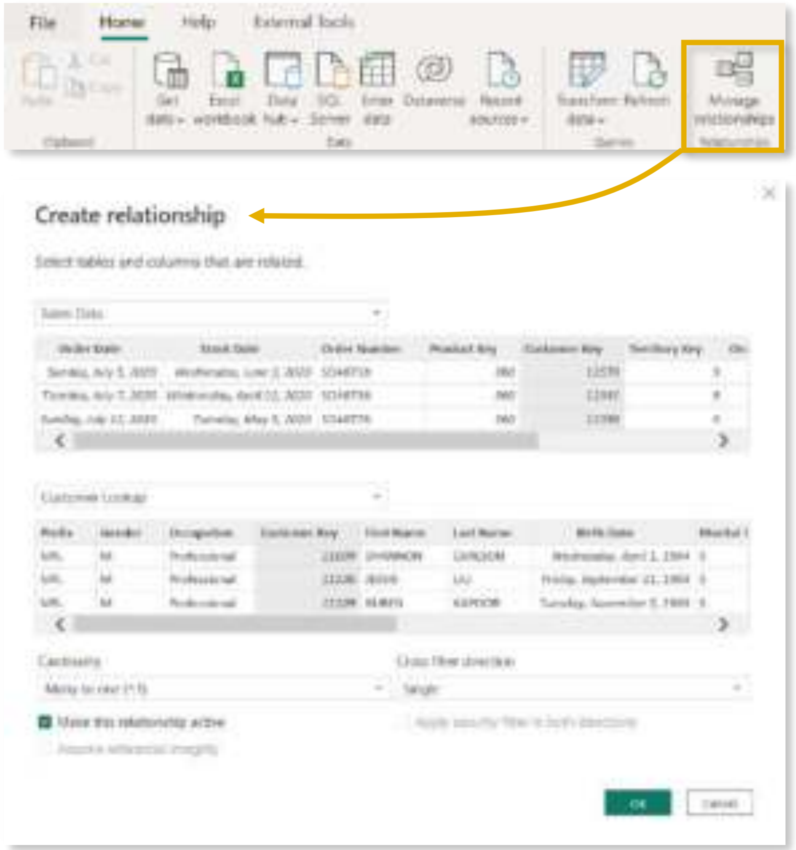
View Options
(Zoom, Reset Layout, Fit to Page)

CREATING TABLE RELATIONSHIPS

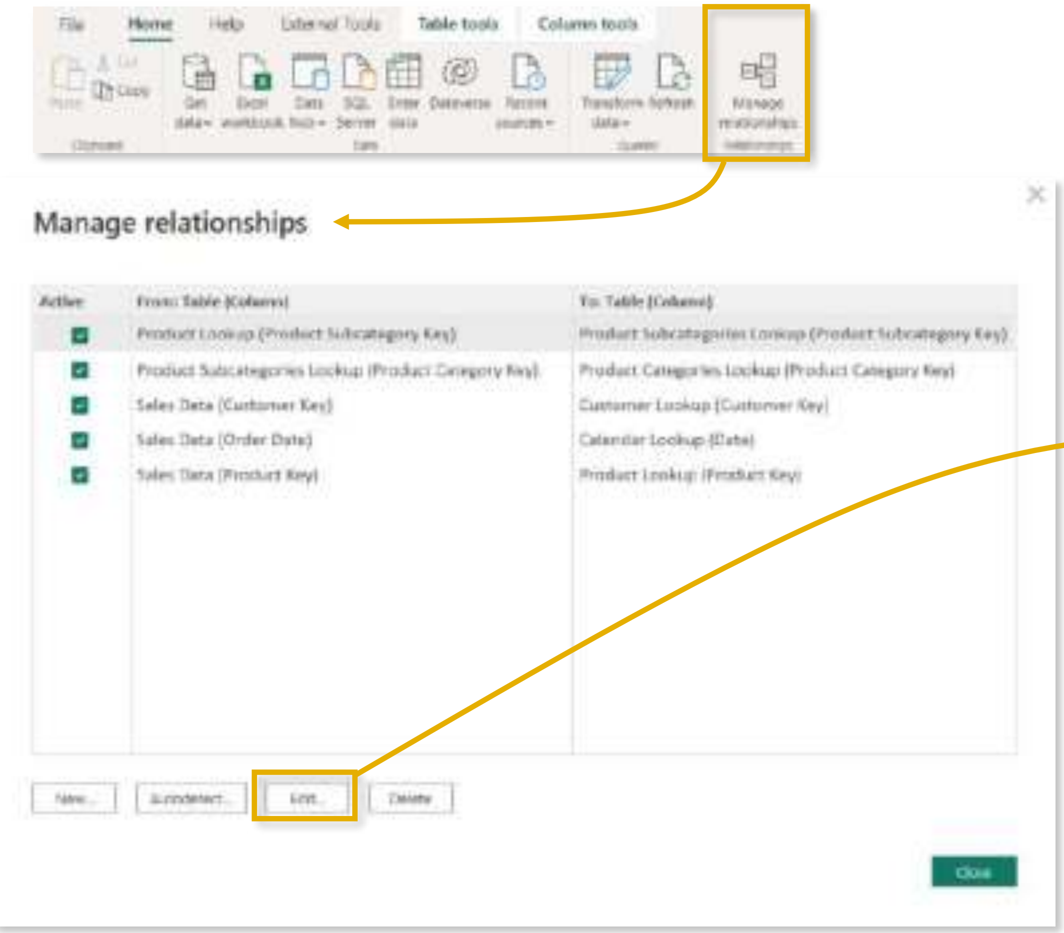
OPTION 1: Click and drag to connect primary and foreign keys within the **Model** view

OPTION 2: Add or detect relationships using the **Manage Relationships** dialog box



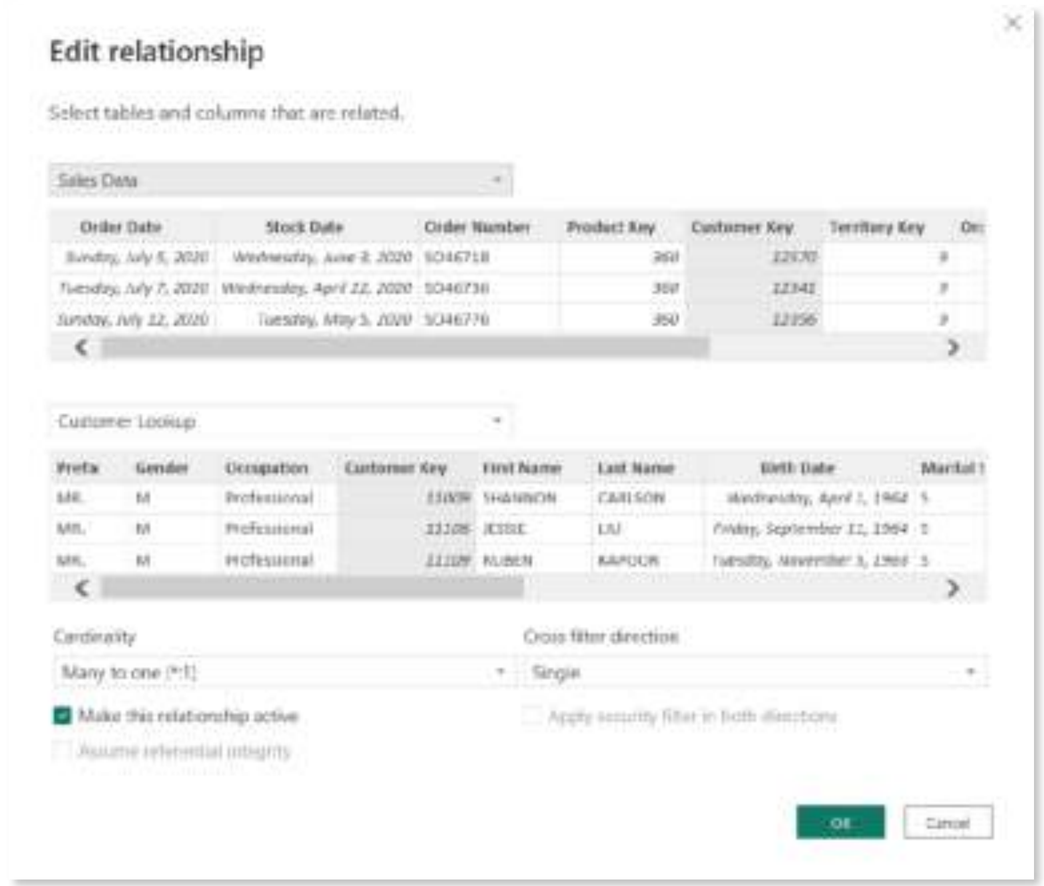


MANAGING & EDITING RELATIONSHIPS



Manage relationships

Active	From: Table (Column)	To: Table (Column)
✓	Product Lookup (Product Subcategory Key)	Product Subcategories Lookup (Product Subcategory Key)
✓	Product Subcategories Lookup (Product Category Key)	Product Categories Lookup (Product Category Key)
✓	Sales Data (Customer Key)	Customer Lookup (Customer Key)
✓	Sales Data (Order Date)	Calendar Lookup (Date)
✓	Sales Data (Product Key)	Product Lookup (Product Key)



Edit relationship

Select tables and columns that are related.

Sales Data

Order Date	Stock Date	Order Number	Product Key	Customer Key	Territory Key	Or
Sunday, July 5, 2020	Wednesday, June 3, 2020	SO46718	360	12570	9	
Tuesday, July 7, 2020	Wednesday, April 22, 2020	SO46736	360	12341	9	
Sunday, July 12, 2020	Tuesday, May 5, 2020	SO46776	350	12356	9	

Customer Lookup

Prefix	Gender	Occupation	Customer Key	First Name	Last Name	Birth Date	Marital
MR.	M	Professional	11009	SHANNON	CARLSON	Wednesday, April 1, 1964	S
MR.	M	Professional	11108	JESSE	LIU	Friday, September 12, 1964	S
MR.	M	Professional	11109	RUBEN	KAPOOR	Tuesday, November 3, 1964	S

Cardinality: Many to one (*:1)

Cross filter direction: Single

[✓] Make this relationship active

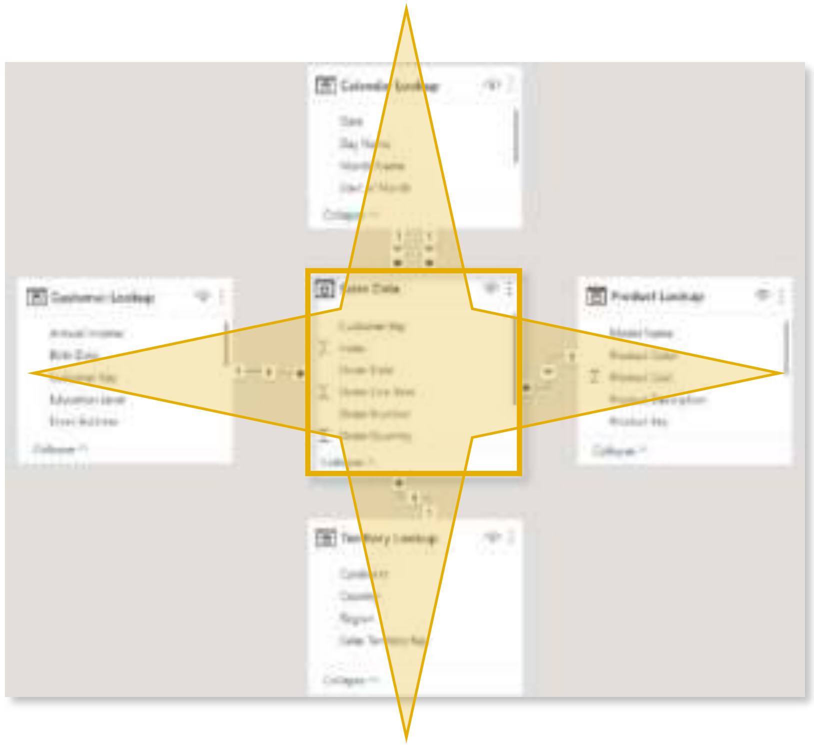
[ ] Apply security filter in both directions

[ ] Assume referential integrity

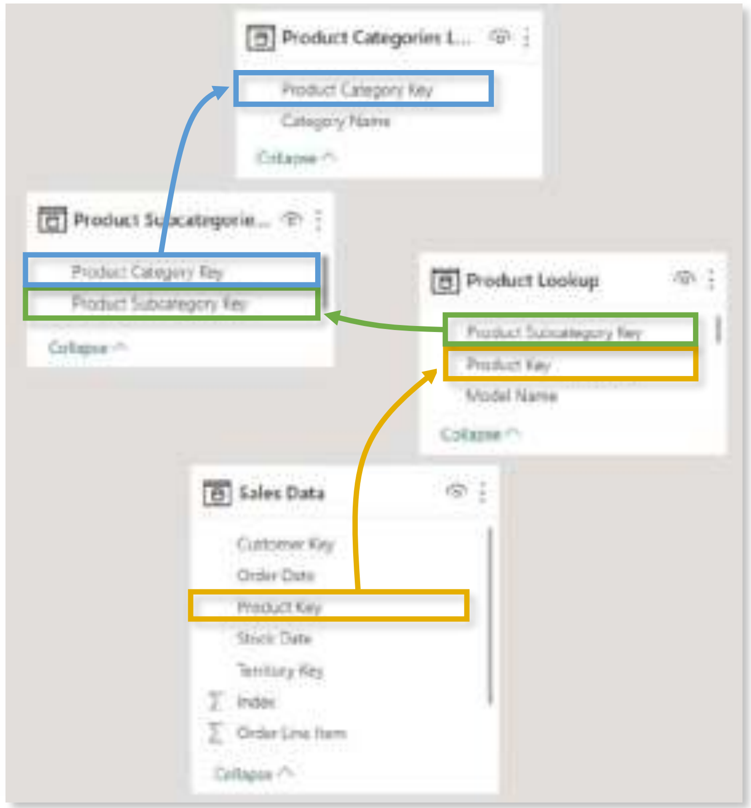
Launch the **Manage Relationships** dialog box or double-click a relationship to modify it

Editing tools allow you to **activate or deactivate** relationships and manage **cardinality** and **filter direction** – more on that soon!

STAR & SNOWFLAKE SCHEMAS



A **star schema** is the simplest and most common type of data model, characterized by a single fact table surrounded by related dimension tables



A **snowflake schema** is an extension of a star, and includes relationships between dimension tables and related sub-dimension tables

ASSIGNMENT: TABLE RELATIONSHIPS

NEW MESSAGE

From: **Dana Modelle** *(Analyst)*

Subject: **Need a favor…**

Hey there,

Ethan shared the data model you've been working on, and we might have an issue…

Last night I left my laptop open, and my cat Dennis somehow got his paws on our model. Now all the relationships are gone!

Could you please rebuild the model, including all three product tables? I owe you one!

-Dana

Reply Forward

Key Objectives

1. Delete all existing table relationships
2. Create a star schema by creating relationships between the Sales, Calendar, Customer, Product and Territories tables
3. Connect all three product tables (Product, Subcategory, Category) in a snowflake schema
4. Use the matrix visual to confirm that you can filter Order Quantity values using fields from each dimension table

SOLUTION: TABLE RELATIONSHIPS

NEW MESSAGE

From: **Dana Modelle** *(Analyst)*

Subject: **Need a favor…**

Hey there,

Ethan shared the data model you've been working on, and we might have an issue…

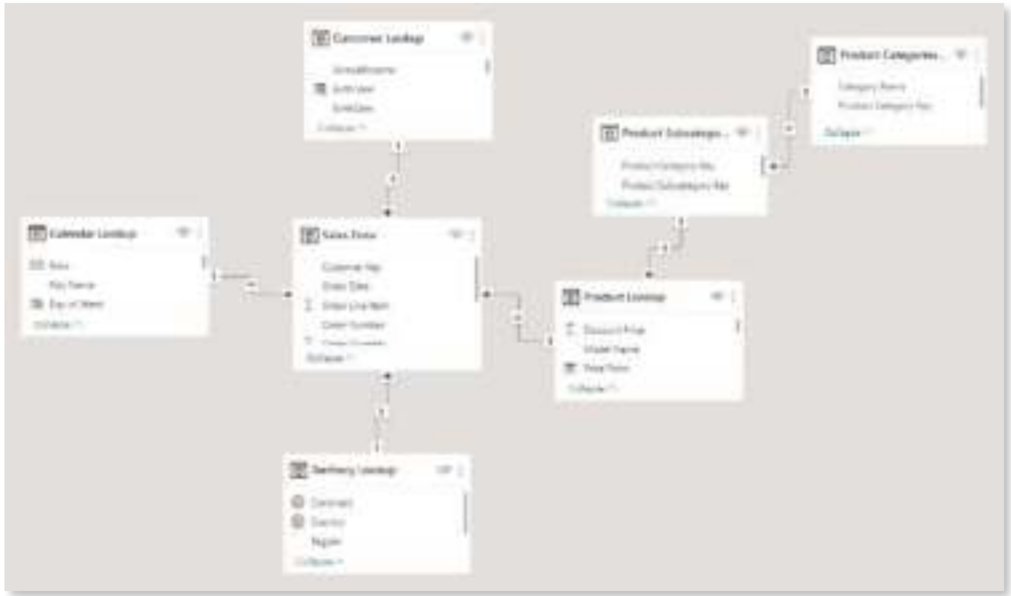
Last night I left my laptop open, and my cat Dennis somehow got his paws on our model. Now all the relationships are gone!

Could you please rebuild the model, including all three product tables? I owe you one!

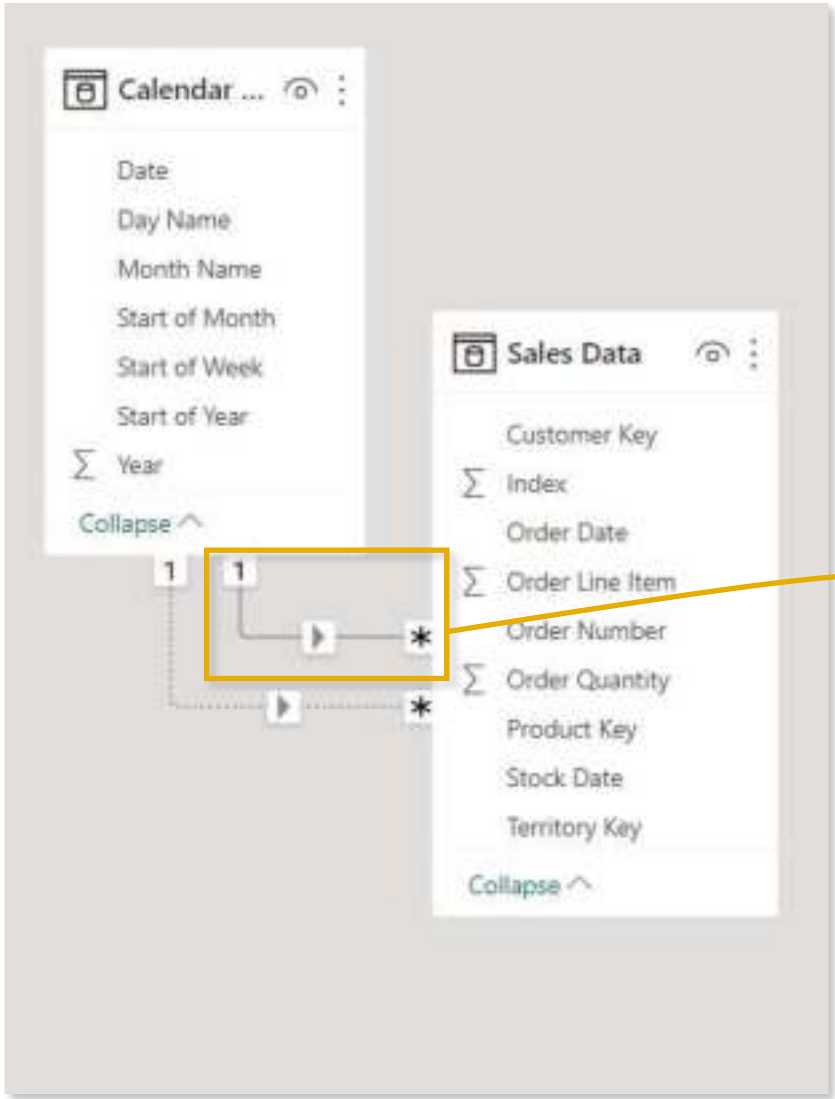
-Dana

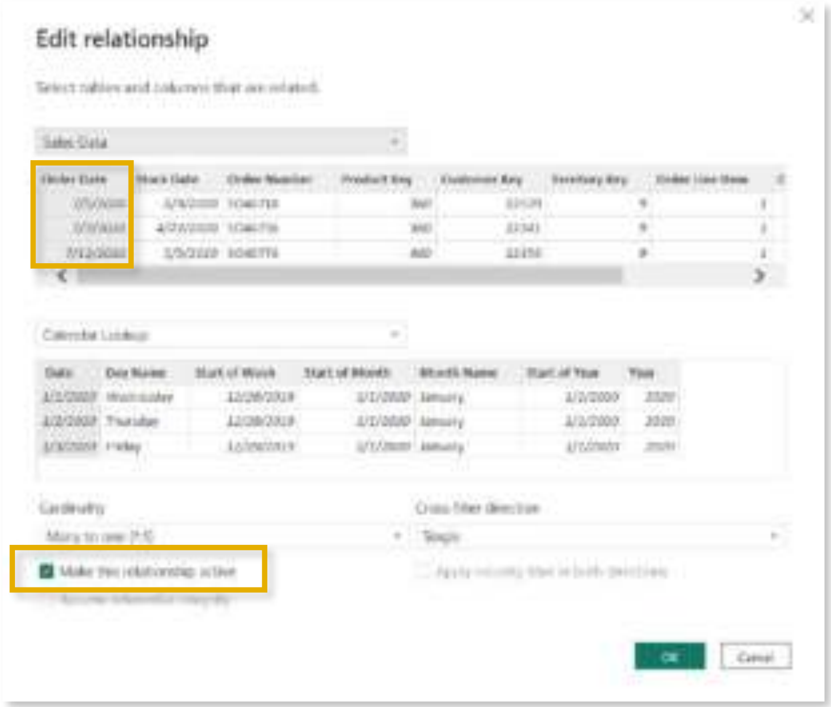
Reply Forward

Solution Preview



PRO TIP: ACTIVE & INACTIVE RELATIONSHIPS





Edit relationship

Select tables and columns that are related.

Sales Data

Order Date	Stock Date	Order Number	Product Key	Customer Key	Territory Key	Order Line Item
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Calendar Lookup

Date	Day Name	Start of Week	Start of Month	Month Name	Start of Year	Year
1/1/2020	Wednesday	12/29/2019	1/1/2020	January	1/1/2020	2020
1/2/2020	Thursday	12/29/2019	1/1/2020	January	1/1/2020	2020
1/3/2020	Friday	12/29/2019	1/1/2020	January	1/1/2020	2020

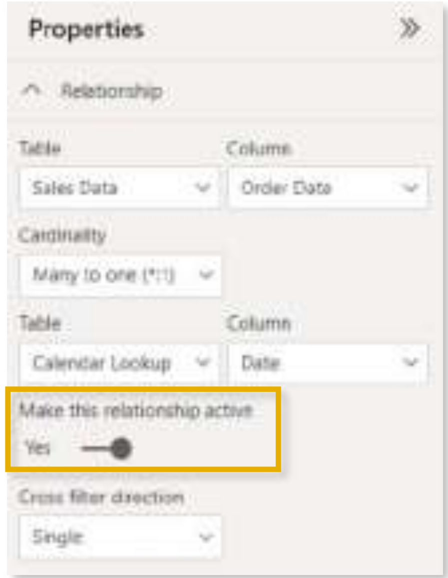
Cardinality: Many to one (*:1)

Cross filter direction: Single

☑ Make this relationship active

☐ Apply security filter in both directions

☐ Assume referential integrity



Properties

Relationship

Table: Sales Data — Column: Order Date

Cardinality: Many to one (*:1)

Table: Calendar Lookup — Column: Date

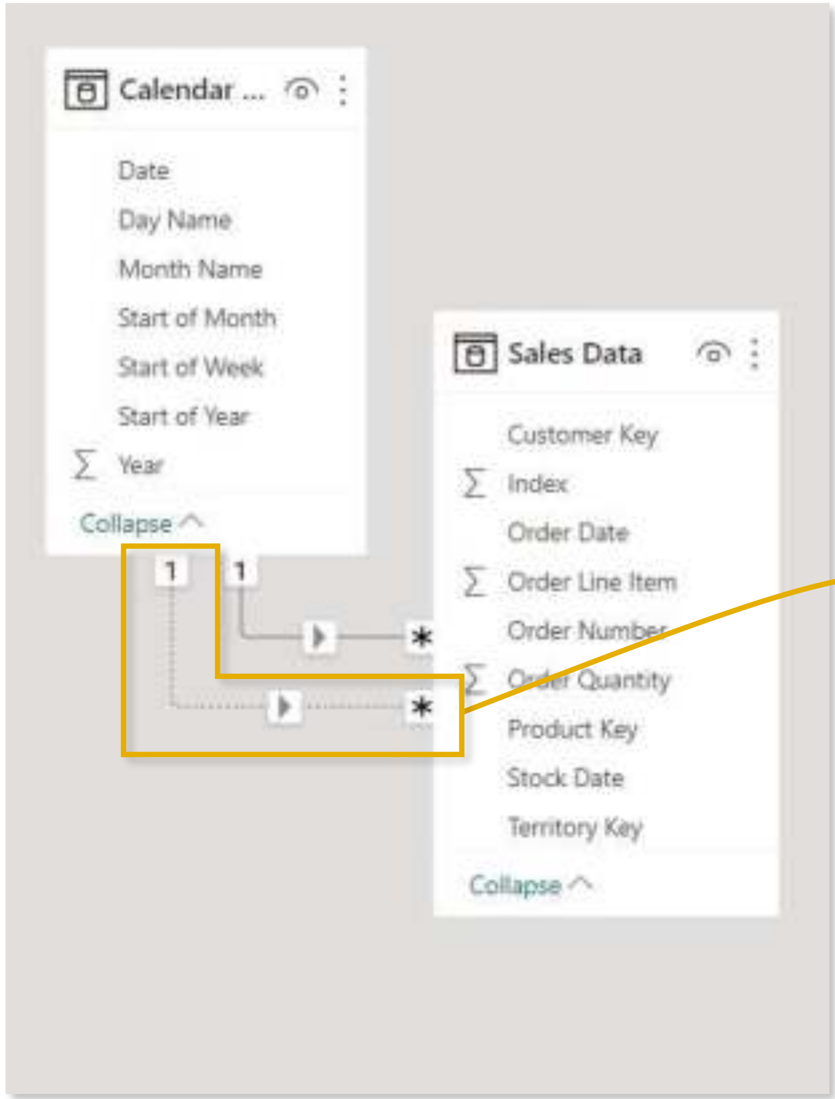
Make this relationship active: Yes

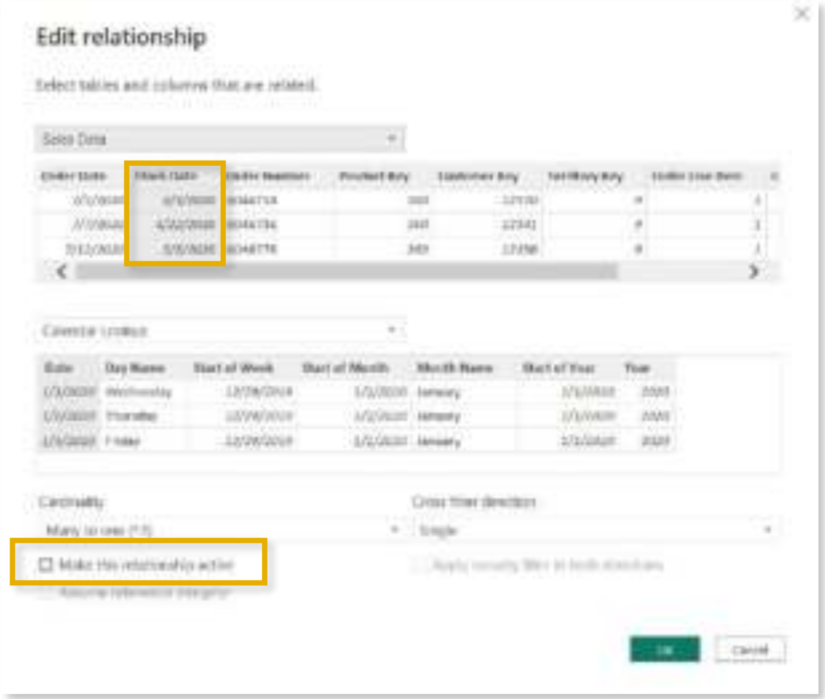
Cross filter direction: Single

The **Sales Data** table contains two date fields (**Order Date** & **Stock Date**), but there can only be ***one active relationship*** to the Date key in the Calendar table

You can set relationships to active or inactive from either the **Edit Relationships** dialog box or the **Properties** (you must deactivate one before activating another)

PRO TIP: ACTIVE & INACTIVE RELATIONSHIPS





Edit relationship

Select tables and columns that are related.

Sales Data

Order Date	Stock Date	Order Number	Product Key	Customer Key	Territory Key	Order Line Item
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Calendar Lookup

Date	Day Name	Start of Week	Start of Month	Month Name	Start of Year	Year
1/1/2020	Wednesday	12/29/2019	1/1/2020	January	1/1/2020	2020
1/2/2020	Thursday	12/29/2019	1/1/2020	January	1/1/2020	2020
1/3/2020	Friday	12/29/2019	1/1/2020	January	1/1/2020	2020

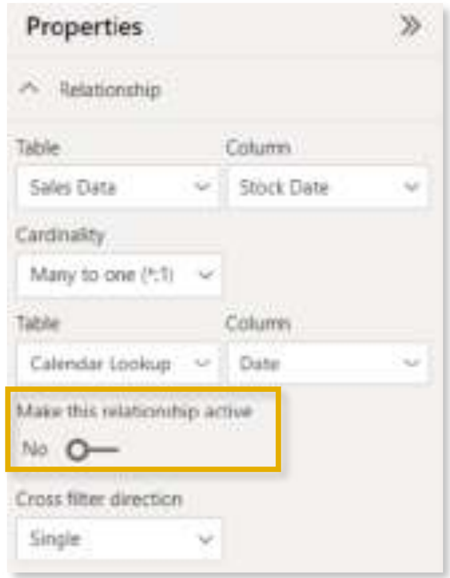
Cardinality: Many to one (*:1)

Cross filter direction: Single

☐ Make this relationship active

Assume referential integrity

Apply security filter in both directions



Properties

Relationship

Table	Column
Sales Data	Stock Date

Cardinality: Many to one (*:1)

Table	Column
Calendar Lookup	Date

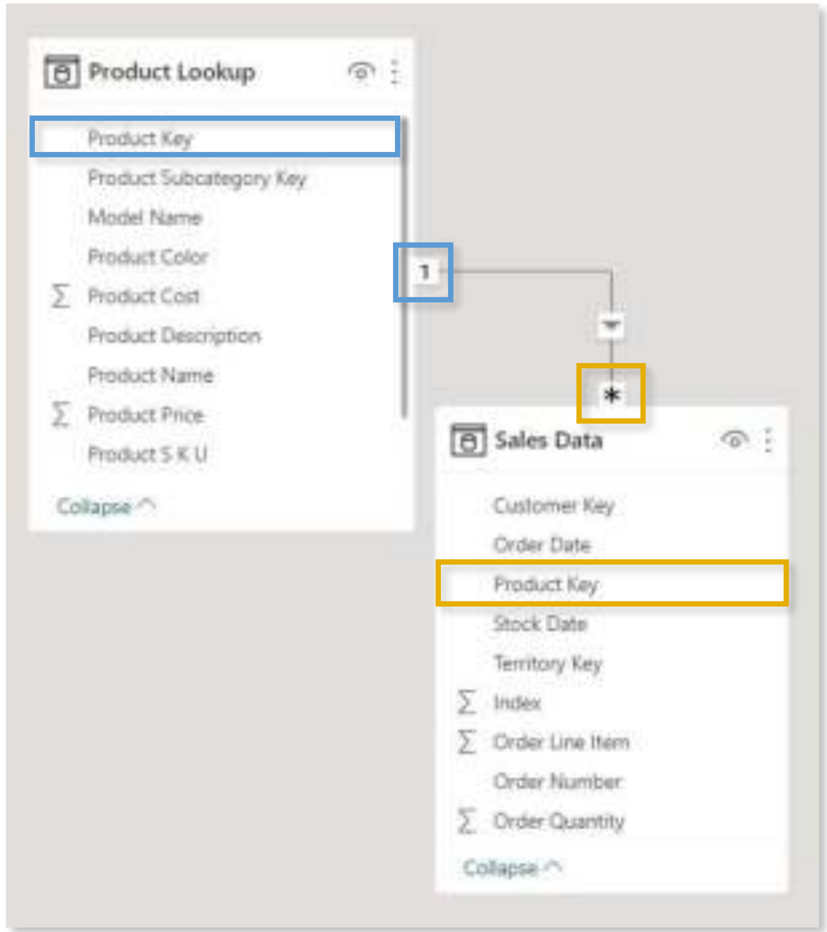
Make this relationship active: No

Cross filter direction: Single

The **Sales Data** table contains two date fields (**Order Date** & **Stock Date**), but there can only be ***one active relationship*** to the Date key in the Calendar table

You can set relationships to active or inactive from either the **Edit Relationships** dialog box or the **Properties** (you must deactivate one before activating another)

RELATIONSHIP CARDINALITY



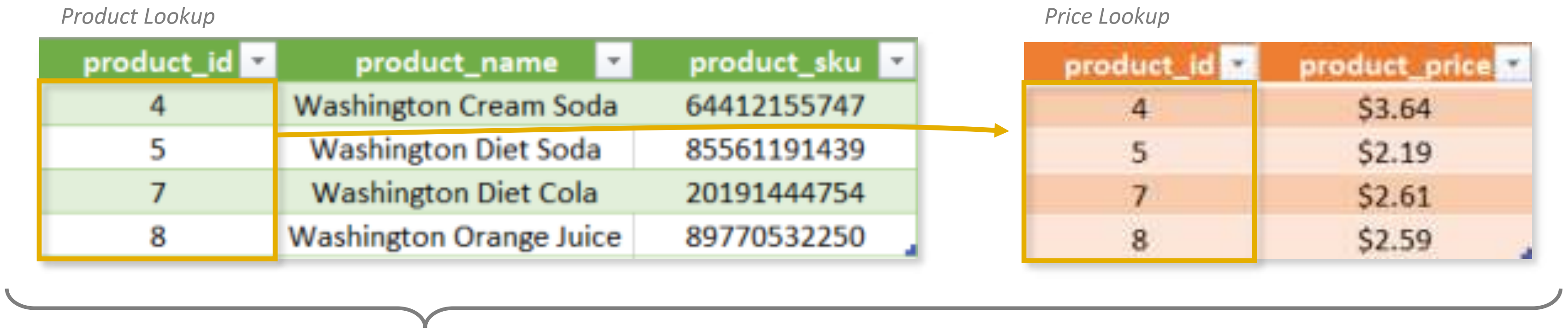
Cardinality refers to the uniqueness of values in a column

- Ideally, all relationships in the data model should follow a **one-to-many** cardinality: **one** instance of each primary key, and **many** instances of each foreign key

*In this example there is only **ONE instance of each Product Key** in the Product table (noted by a "**1**"), since each row contains **attributes of a single product** (name, SKU, description, price, etc.)*

*There are **MANY instances of each Product Key** in the Sales table (noted by an asterisk *****), since there are **multiple sales for each product***

EXAMPLE: ONE-TO-ONE CARDINALITY



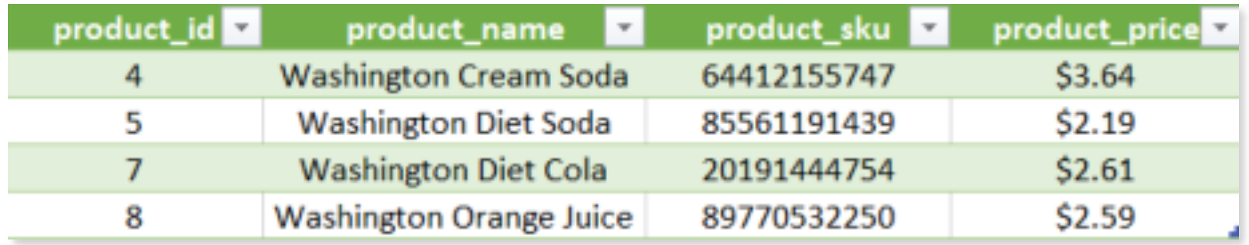
Product Lookup

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

Price Lookup

product_id	product_price
4	$3.64
5	$2.19
7	$2.61
8	$2.59

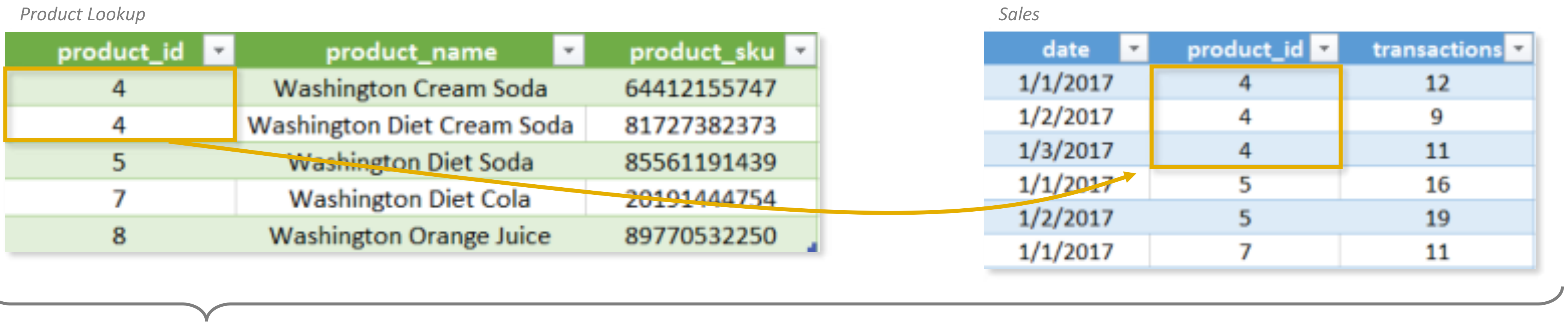
- Connecting the two tables above using **product_id** creates a **one-to-one relationship**, since each product ID only appears once in each table
- This isn't necessarily a "bad" relationship, but you can simplify the model by merging the tables into a single, valid dimension table



product_id	product_name	product_sku	product_price
4	Washington Cream Soda	64412155747	$3.64
5	Washington Diet Soda	85561191439	$2.19
7	Washington Diet Cola	20191444754	$2.61
8	Washington Orange Juice	89770532250	$2.59

***NOTE:** this still respects the rules of normalization, since all rows are unique and capture product-specific attributes*

EXAMPLE: MANY-TO-MANY CARDINALITY

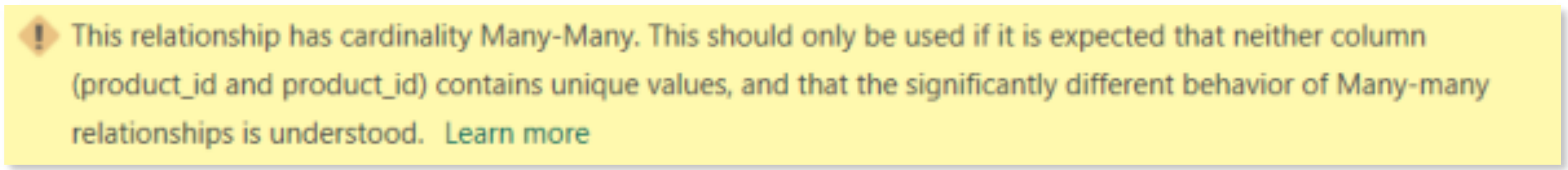


Product Lookup

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
4	Washington Diet Cream Soda	81727382373
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

Sales

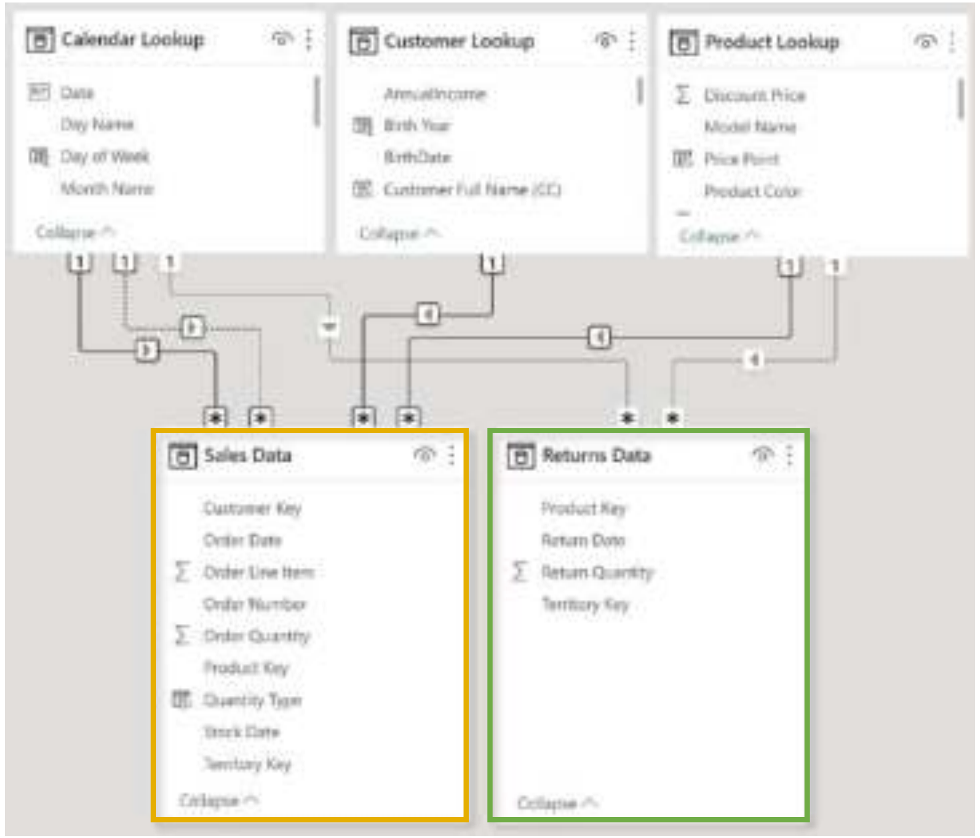
date	product_id	transactions
1/1/2017	4	12
1/2/2017	4	9
1/3/2017	4	11
1/1/2017	5	16
1/2/2017	5	19
1/1/2017	7	11



This relationship has cardinality Many-Many. This should only be used if it is expected that neither column (product_id and product_id) contains unique values, and that the significantly different behavior of Many-many relationships is understood. Learn more

- If we try to connect the tables above using **product_id**, we'll get a **many-to-many relationship** warning since there are multiple instances of product_id in both tables
- Even if we force this relationship, how would we know which product was actually sold on each date – **Cream Soda** or **Diet Cream Soda**?

CONNECTING MULTIPLE FACT TABLES



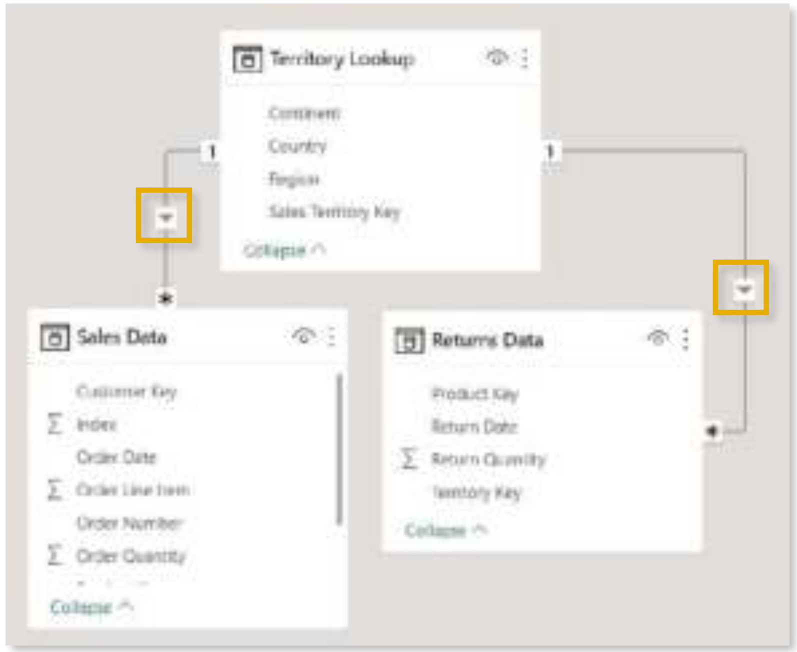
This model contains two fact tables: **Sales Data** and **Returns Data**

- Since there is no primary/foreign key relationship, we can't connect them directly to each other
- But we *can* connect each fact table to related lookups, which allows us to filter both sales and returns data **using fields from any shared lookup tables**
- We can view orders and returns by product since both tables relate to Product Lookup, but we can't view returns by customer since no relationship exists

HEY THIS IS IMPORTANT!

Generally speaking, fact tables should **connect through shared dimension tables, not directly to each other**

FILTER CONTEXT & FLOW



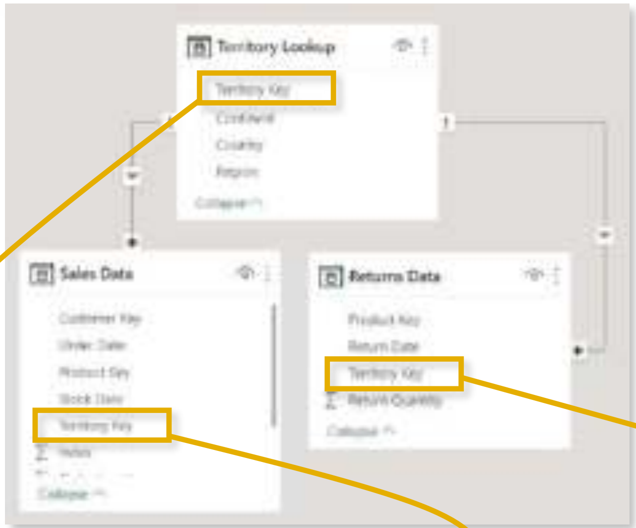
Here we have two data tables (**Sales Data** and **Returns Data**), connected to **Territory Lookup**

The arrows show the **filter direction**, and point from the one (**1**) side of the relationship to the many (*****) side

- When you filter a table, that **filter context** is passed to any related "downstream" tables, following the arrow's direction
- Filter context CANNOT flow "upstream"

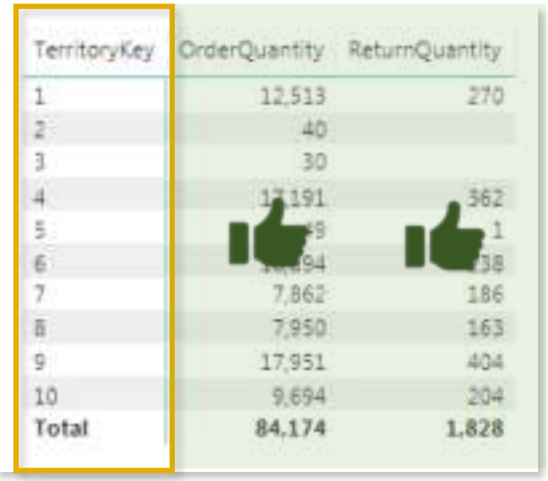
PRO TIP: Arrange lookup tables above fact tables in your model as a visual reminder that **filters always flow downstream**

EXAMPLE: FILTER FLOW



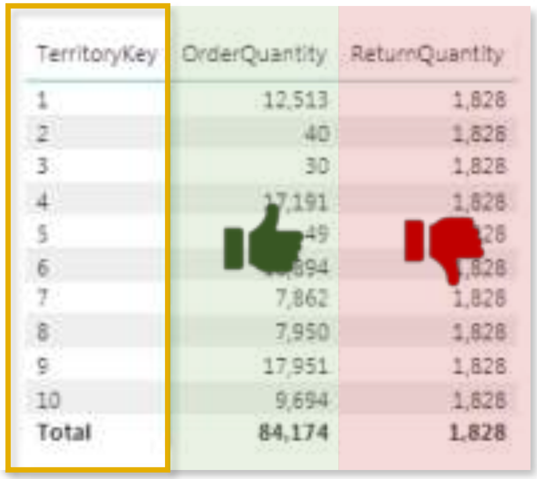
In this model, the only way to filter both **Sales** and **Returns** data by **Territory** is to use the **Territory Key** from the lookup table, which is upstream and related to both fact tables

- Filtering using Territory Key from the **Sales** table yields **incorrect Returns values**, since the filter context can't flow to any other table
- Filtering using Territory Key from the **Returns** table yields **incorrect Sales values**, and is limited to territories that exist in the returns table



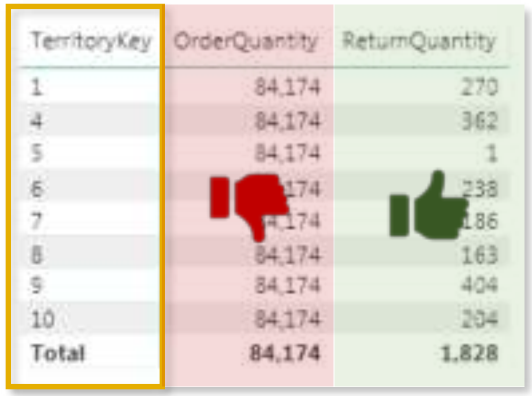
TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
2	40	
3	30	
4	17,191	362
5	[illegible]	1
6	[illegible]	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

*Filtering by **Territory Lookup**[Territory Key]*



TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	1,828
2	40	1,828
3	30	1,828
4	17,191	1,828
5	[illegible]	1,828
6	[illegible]	1,828
7	7,862	1,828
8	7,950	1,828
9	17,951	1,828
10	9,694	1,828
Total	**84,174**	**1,828**

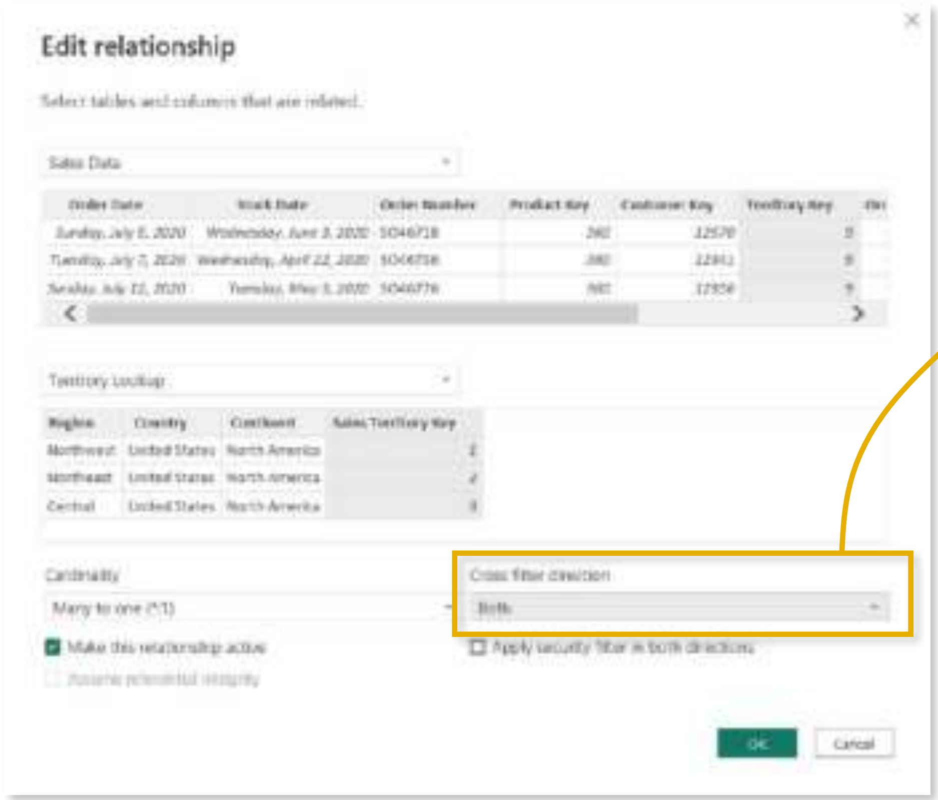
*Filtering by **Sales Data**[Territory Key]*



TerritoryKey	OrderQuantity	ReturnQuantity
1	84,174	270
4	84,174	362
5	84,174	1
6	84,174	238
7	84,174	186
8	84,174	163
9	84,174	404
10	84,174	204
Total	**84,174**	**1,828**

*Filtering by **Returns Data**[Territory Key]*

BI-DIRECTIONAL FILTERS



Edit relationship

Select tables and columns that are related.

Sales Data

Order Date	Stock Date	Order Number	Product Key	Customer Key	Territory Key
Sunday, July 5, 2020	Wednesday, June 3, 2020	SO46718	260	12570	5
Tuesday, July 7, 2020	Wednesday, April 22, 2020	SO46758	380	12941	9
Sunday, July 12, 2020	Tuesday, May 5, 2020	SO46776	560	12958	9

Territory Lookup

Region	Country	Continent	Sales Territory Key
Northwest	United States	North America	1
Northeast	United States	North America	2
Central	United States	North America	3

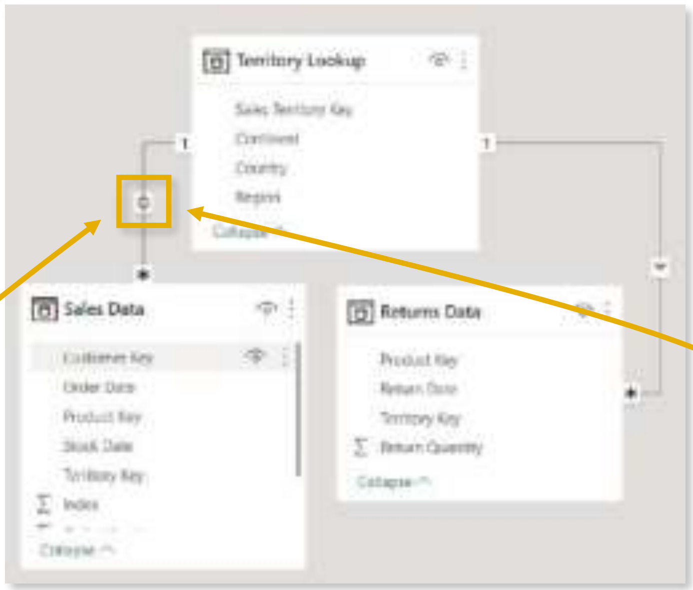
Cardinality: Many to one (*:1)

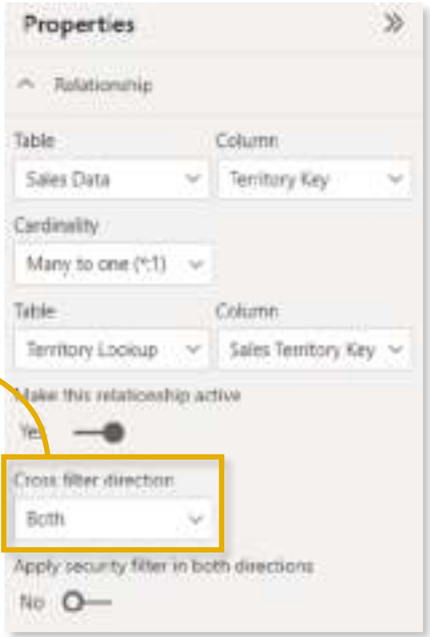
Cross filter direction: Both

[✓] Make this relationship active

[ ] Apply security filter in both directions

[ ] Assume referential integrity





Properties

Relationship

Table	Column
Sales Data	Territory Key

Cardinality: Many to one (*:1)

Table	Column
Territory Lookup	Sales Territory Key

Make this relationship active: Yes

Cross filter direction: Both

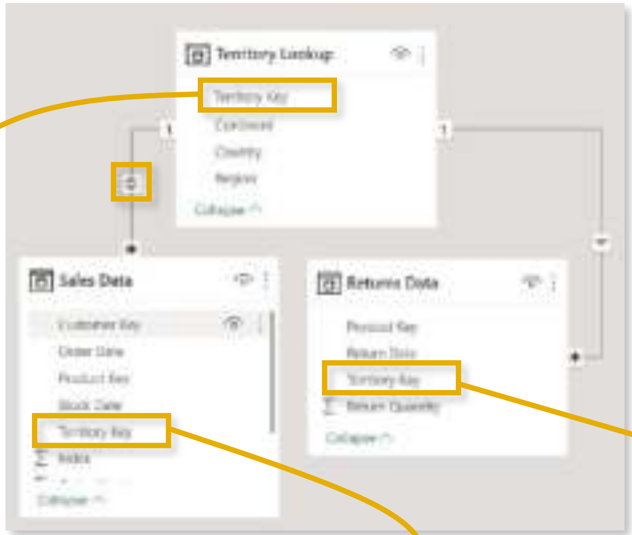
Apply security filter in both directions: No

Updating the **cross-filter direction** from **Single** to **Both** allows filter context to flow in either direction

- In this example, filters applied to the **Sales** table can pass up to the **Territory Lookup** table, then down to **Returns**

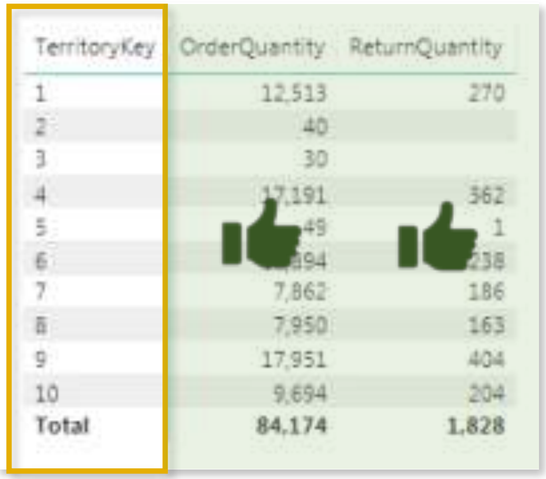
**The "Apply security filter in both directions" option relates to row-level security (RLS) settings, which are out of scope for this course*

EXAMPLE: BI-DIRECTIONAL FILTERS



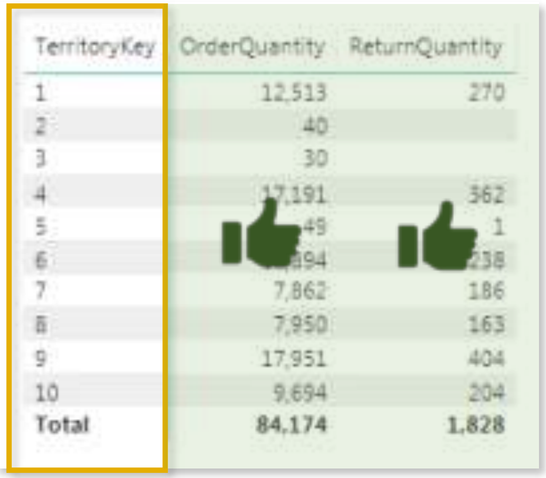
With two-way cross-filtering enabled between **Sales** and **Territory**, we now see correct values using **Territory Key** from *either* table

- Filter context can now pass up to the **Territory Lookup** table, then downstream to **Returns**
- However, we still see incorrect values when filtering using Territory Key from the **Returns** table, since the filter context is isolated to that single table



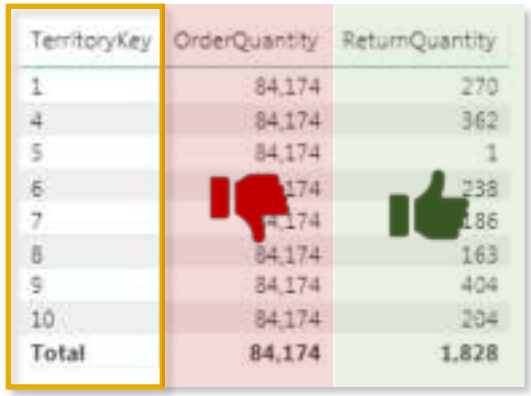
TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
2	40	
3	30	
4	17,191	362
5	[illegible]49	1
6	[illegible]894	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

*Filtering by **Territory Lookup**[Territory Key]*



TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
2	40	
3	30	
4	17,191	362
5	[illegible]49	1
6	[illegible]894	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

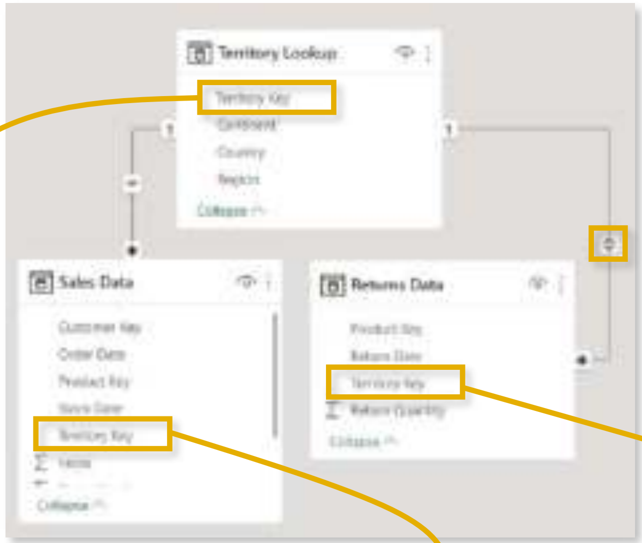
*Filtering by **Sales Data**[Territory Key]*



TerritoryKey	OrderQuantity	ReturnQuantity
1	84,174	270
4	84,174	362
5	84,174	1
6	[illegible]174	238
7	[illegible]174	[illegible]86
8	84,174	163
9	84,174	404
10	84,174	204
Total	**84,174**	**1,828**

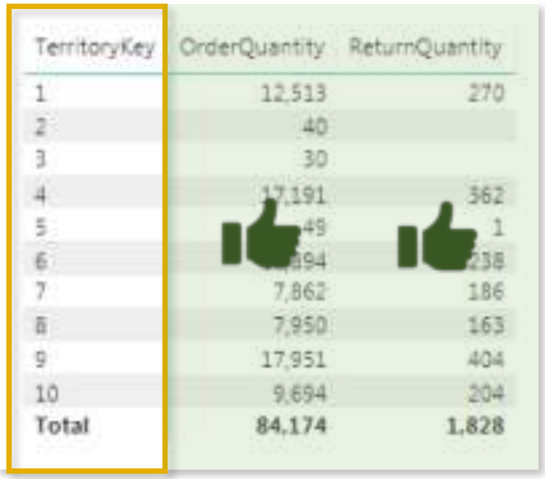
*Filtering by **Returns Data**[Territory Key]*

EXAMPLE: BI-DIRECTIONAL FILTERS



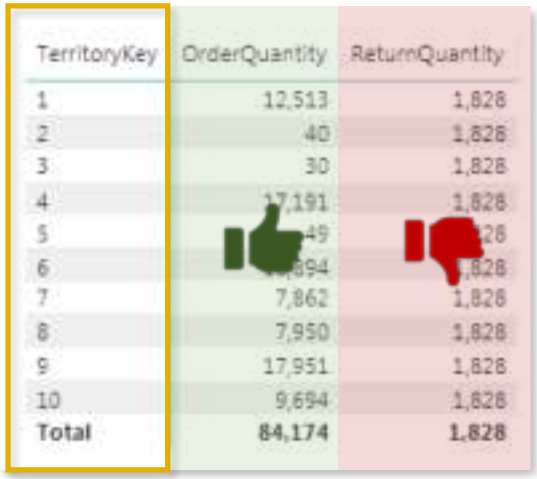
In this case, we've enabled two-way cross-filtering between the **Returns** and **Territory** tables

- As expected, we now see incorrect values when filtering using Territory Key from the **Sales** table, since the filter context is isolated to that single table
- While the values *appear* to be correct when filtering using Territory Key from the **Returns** table, we're **missing sales data** from any territories that didn't appear in the returns table (specifically Territories **2** & **3**)



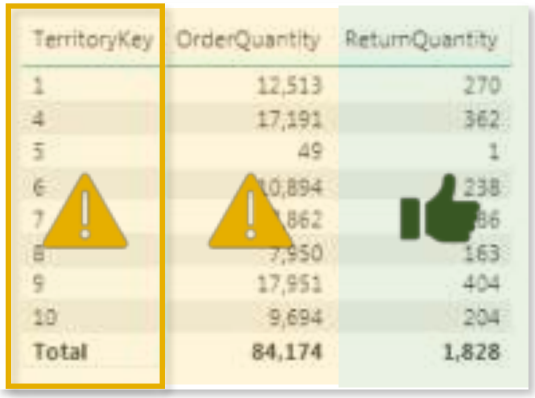
TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
2	40	
3	30	
4	17,191	362
5	49	1
6	10,894	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

*Filtering by **Territory Lookup**[Territory Key]*



TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	1,828
2	40	1,828
3	30	1,828
4	17,191	1,828
5	49	1,828
6	10,894	1,828
7	7,862	1,828
8	7,950	1,828
9	17,951	1,828
10	9,694	1,828
Total	**84,174**	**1,828**

*Filtering by **Sales Data**[Territory Key]*

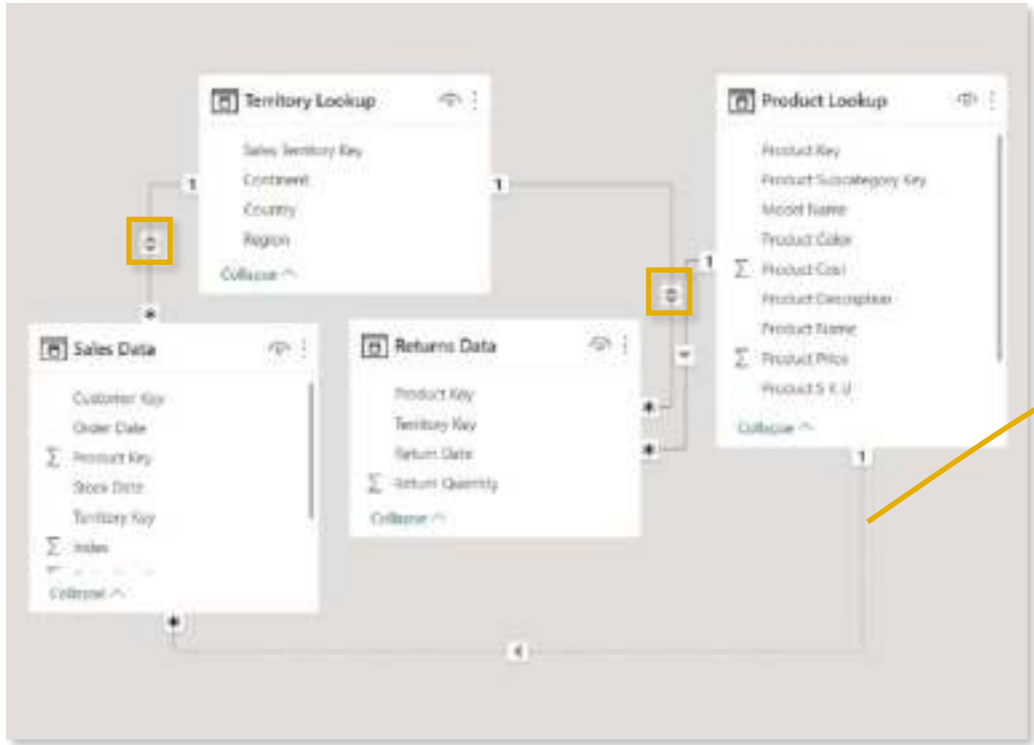


TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
4	17,191	362
5	49	1
6	10,894	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

*Filtering by **Returns Data**[Territory Key]*

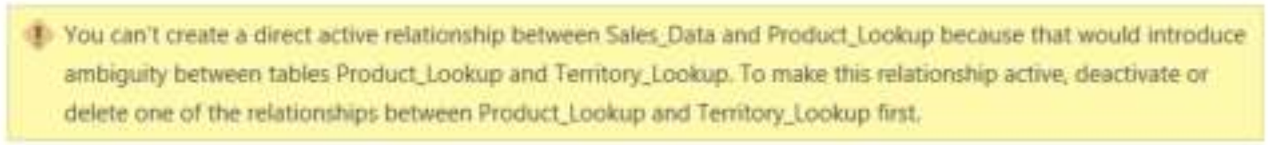
*Territories 2 & 3 don't exist in the **Returns** table, so they aren't included in the filter context that passes to **Territory Lookup** and **Sales***

AMBIGUITY



Use two-way filters carefully, and **only when necessary**

- Using multiple two-way filters can cause **ambiguity** by introducing multiple filter paths between tables

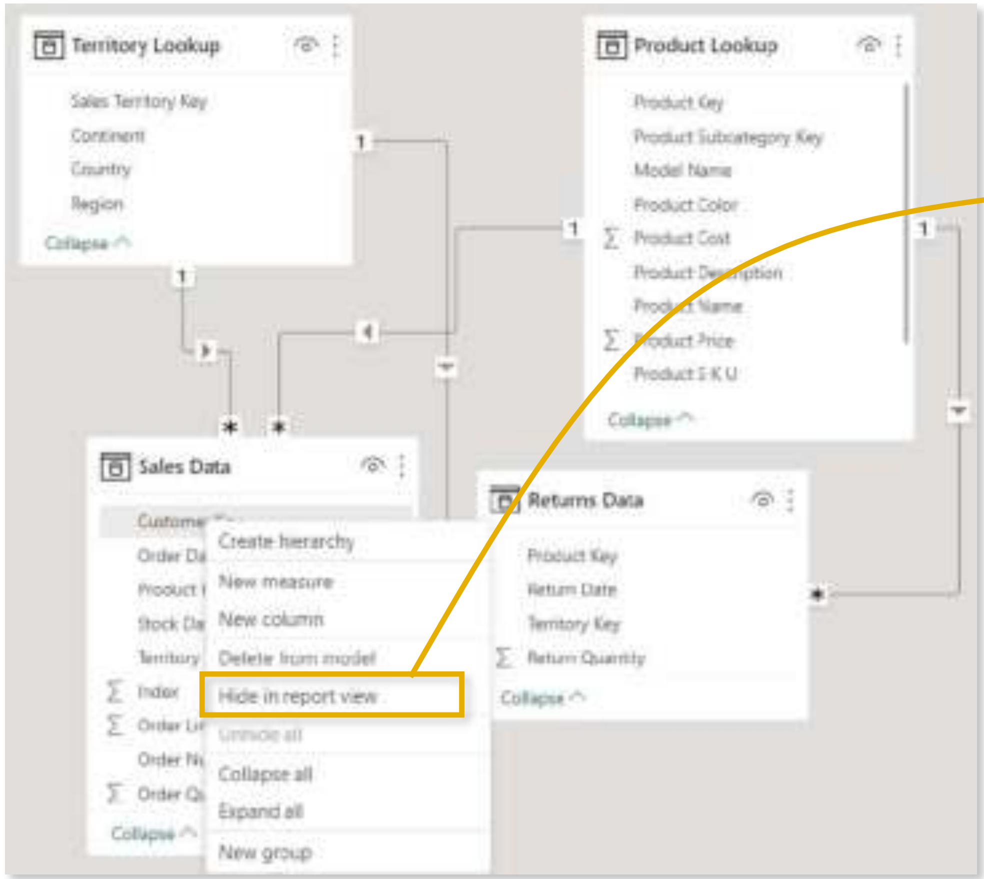

You can't create a direct active relationship between Sales_Data and Product_Lookup because that would introduce ambiguity between tables Product_Lookup and Territory_Lookup. To make this relationship active, deactivate or delete one of the relationships between Product_Lookup and Territory_Lookup first.

*In this example, filter context from the **Product** table can pass down to **Returns** and up to **Territory Lookup**, which would be filtered based on the Territory Keys passed from the Returns table*

*With an active relationship between **Product** and **Sales** as well, filter context could pass through **either the Sales or Returns table to reach the Territory Lookup table**, which could yield conflicting filter context*

PRO TIP: Design your models with **one-way filters** and **1:many cardinality** unless more complex relationships are absolutely necessary

HIDING FIELDS



Hide in Report View makes fields inaccessible from the Report tab, but still available in **Data** and **Model** views

- This can be controlled by right-clicking a field in the Data or Model view, or by selecting "**Is hidden**" in the Properties pane
- This is commonly used to prevent users from filtering using invalid fields, reduce clutter, or to hide irrelevant metrics from view

PRO TIP: Hide the **foreign keys** in fact tables to force users to filter using **primary keys** in dimension tables

ASSIGNMENT: FILTER FLOW

NEW MESSAGE

From: **Dana Modelle** *(Analyst)*

Subject: **Larry's gone rogue!**

Hey there, we've got another problem.

Larry from Sales just sent me this screenshot. I think he must have downloaded our Power BI model and messed with some relationships, because I KNOW we had sales for product 338.

Can you help diagnose what's going on, and prevent him from doing this again?

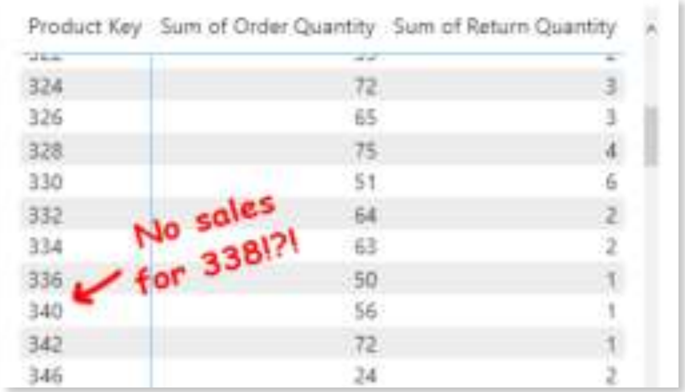
-Dana

P.S. Kevin says hi

Reply Forward

Key Objectives

1. Replicate Larry's matrix below to diagnose what he must have done to the model*



Product Key	Sum of Order Quantity	Sum of Return Quantity
324	72	3
326	65	3
328	75	4
330	51	6
332	64	2
334	63	2
336	50	1
340	56	1
342	72	1
346	24	2

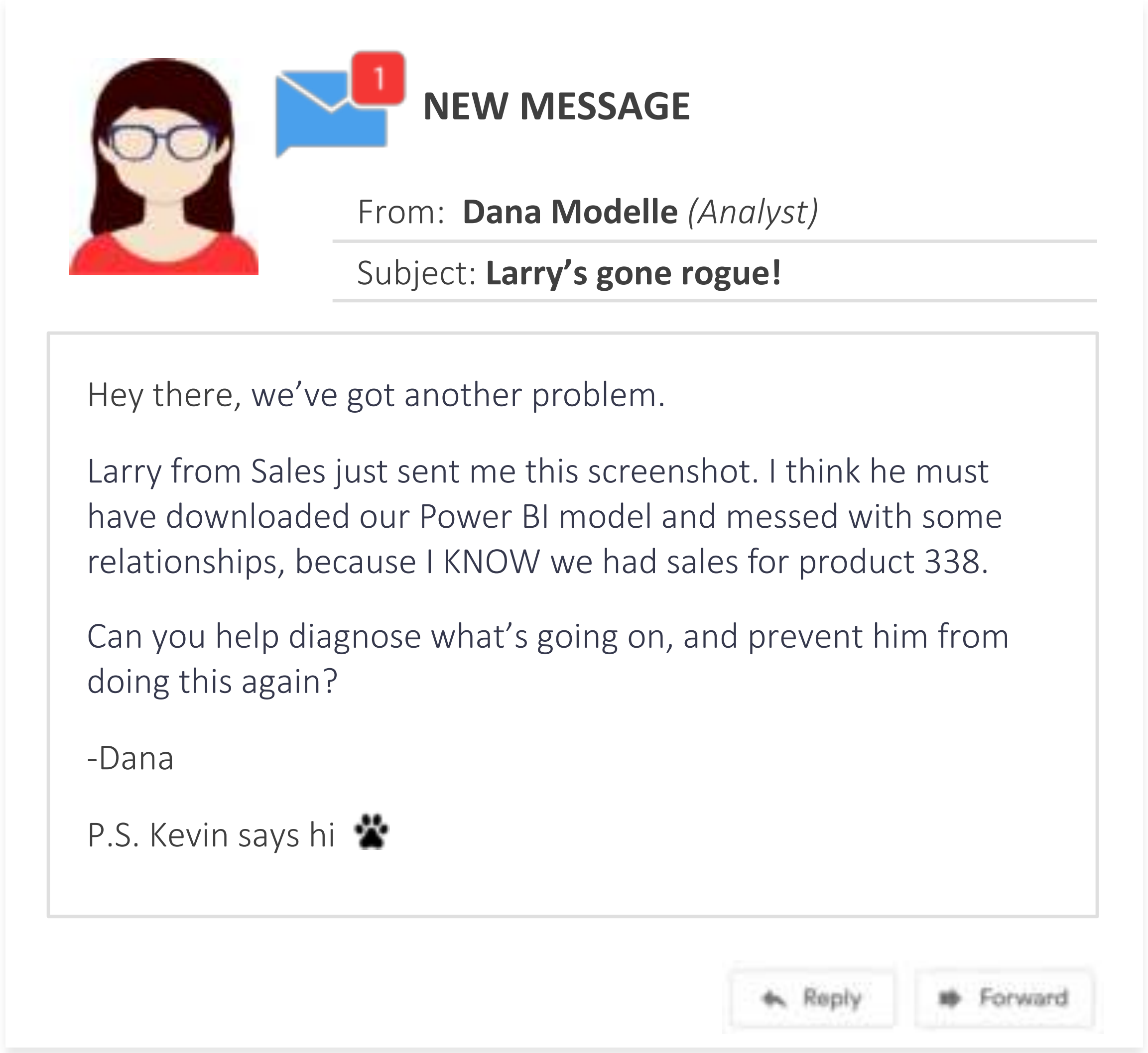
No sales for 338!?!

- Which product is #338?
- Why didn't Larry's matrix show any orders?

2. Hide any remaining foreign keys to prevent other users from making the same mistake

***Hint**: you may need to temporarily change a relationship to bi-directional*

SOLUTION: FILTER FLOW



NEW MESSAGE

From: **Dana Modelle** *(Analyst)*

Subject: **Larry's gone rogue!**

Hey there, we've got another problem.

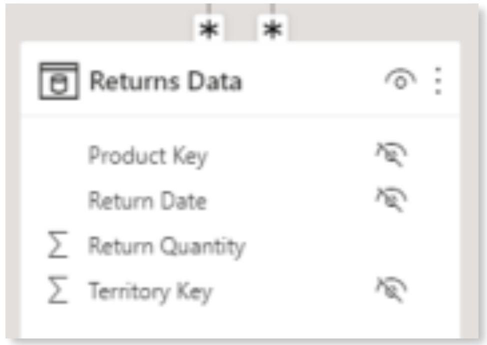
Larry from Sales just sent me this screenshot. I think he must have downloaded our Power BI model and messed with some relationships, because I KNOW we had sales for product 338.

Can you help diagnose what's going on, and prevent him from doing this again?

-Dana

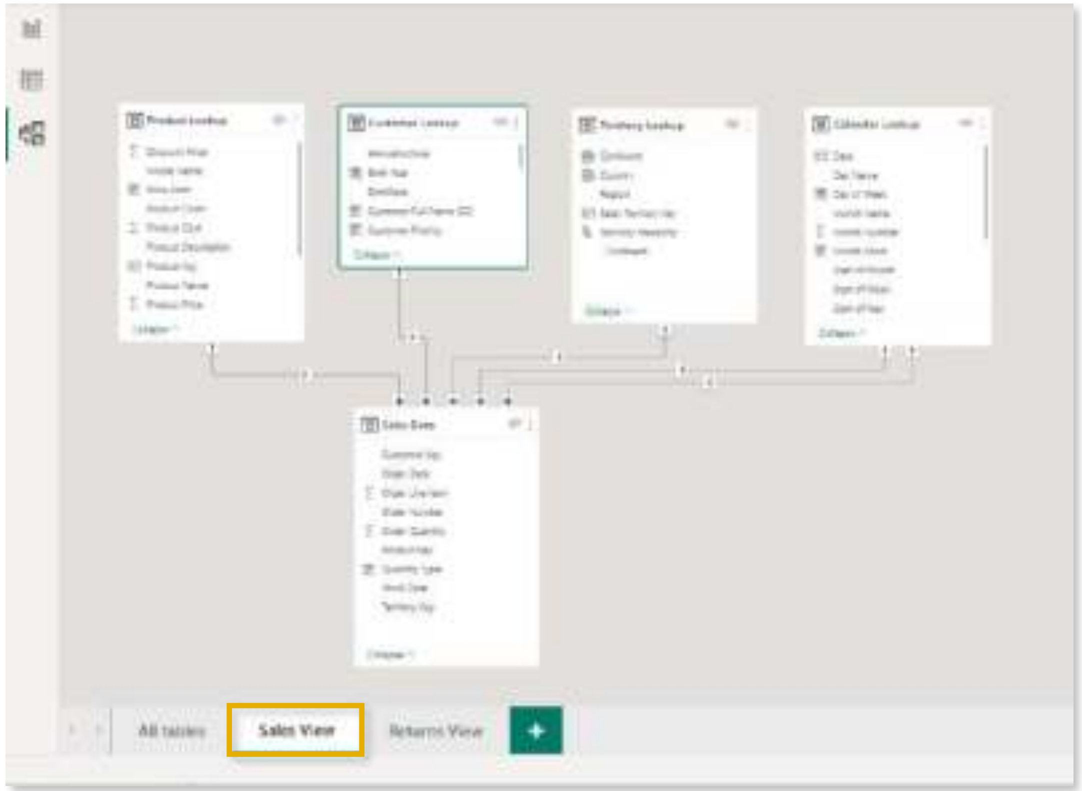
P.S. Kevin says hi

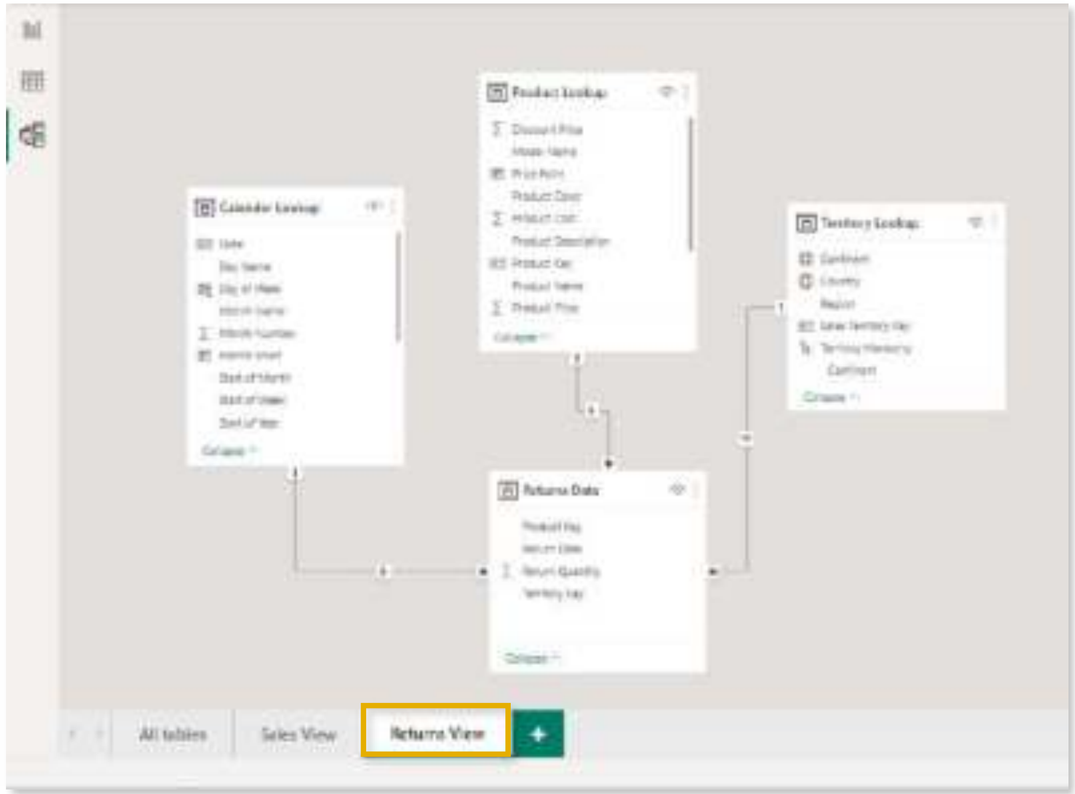
Solution Preview

1. Larry must have changed the relationship between **Returns Data** and **Product Lookup** to **bi-directional**, and filtered his matrix using product_id from the Returns table
   - Road bike (Road-650 Black, 44)
   - Product 338 doesn't exist in the Returns table, so it was excluded when that filter context passed to the Sales table
2. 

Returns Data
- Product Key
- Return Date
- Return Quantity
- Territory Key

PRO TIP: MODEL LAYOUTS





Model layouts allow you to create custom views to show specific portions of large, complex models

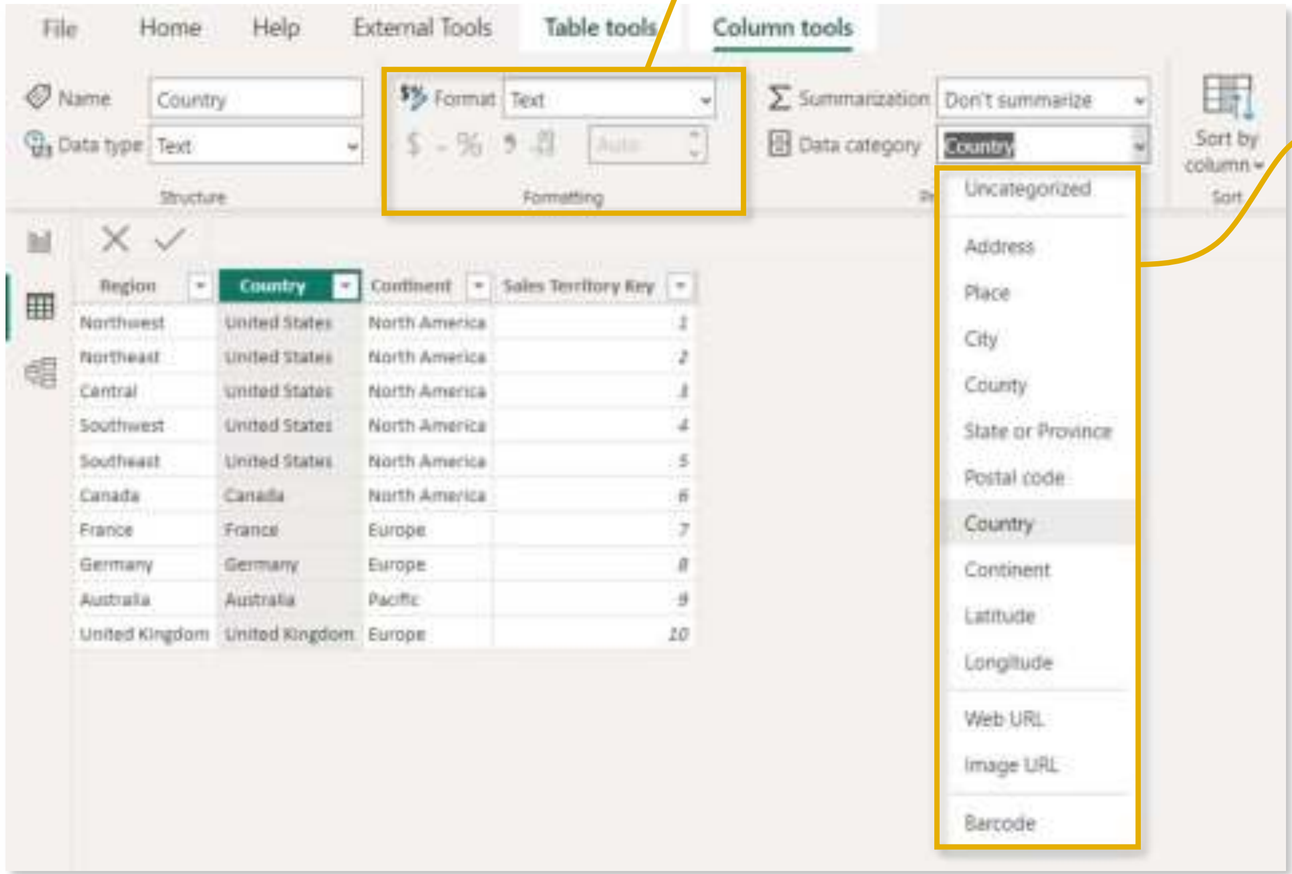
- Here we've created a **Sales View** displaying only tables related to sales, and a **Returns View** displaying only tables related to returns *(**Note**: this doesn't actually create duplicate tables)*

DATA FORMATS & CATEGORIES

Customize **data formats** from the Column tools menu in the **Data** view or the Properties pane in the **Model** view

Assign **data categories** for geospatial fields, URLs or barcodes

- This is commonly used to help Power BI map location-based fields like addresses, countries, cities, coordinates, zip codes, etc.



Name: Country
Data type: Text
Format: Text
Summarization: Don't summarize
Data category: Country

Uncategorized
Address
Place
City
County
State or Province
Postal code
Country
Continent
Latitude
Longitude
Web URL
Image URL
Barcode

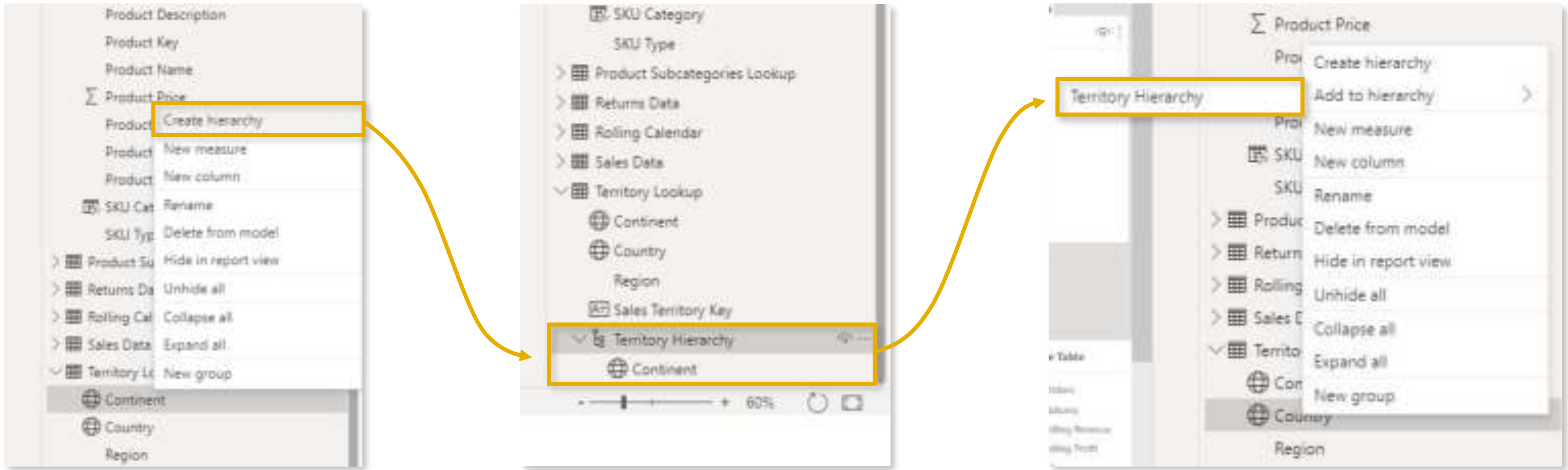
Region	Country	Continent	Sales Territory Key
Northwest	United States	North America	1
Northeast	United States	North America	2
Central	United States	North America	3
Southwest	United States	North America	4
Southeast	United States	North America	5
Canada	Canada	North America	6
France	France	Europe	7
Germany	Germany	Europe	8
Australia	Australia	Pacific	9
United Kingdom	United Kingdom	Europe	10

Select All
Europe
North America
Pacific
UNITED KINGDOM
IRELAND
London
NETHERLANDS
Amsterdam
GERMANY
POLAND
BELGIUM
Paris
FRANCE
AUSTRIA
HUNGARY
LITHUANIA
Bing

HIERARCHIES

Hierarchies are groups of columns that reflect multiple levels of granularity

- For example, a **Geography hierarchy** might include **Country**, **State** and **City** fields
- Hierarchies are treated as a **single item** in tables and reports, allowing users to "drill up" and "drill down" through each level



*In the **Data** pane, right-click a field and select **Create hierarchy***

*This hierarchy contains "Continent", and is named "**Territory Hierarchy**"*

*Right-click another field (like "Country") and select **Add to Hierarchy** (or drag it in!)*

ASSIGNMENT: HIERARCHIES

NEW MESSAGE

From: **Dana Modelle** *(Analyst)*

Subject: **Adding a date hierarchy**

Good morning!

Hoping you can help with a quick request.

Since we'll be doing a lot of time-series analysis, Ethan asked us to add a date hierarchy to the model so that users can quickly view trends at any level of granularity (year, month, day, etc.)

Please get that added before our afternoon call. Thanks!

-Dana

Reply Forward

Key Objectives

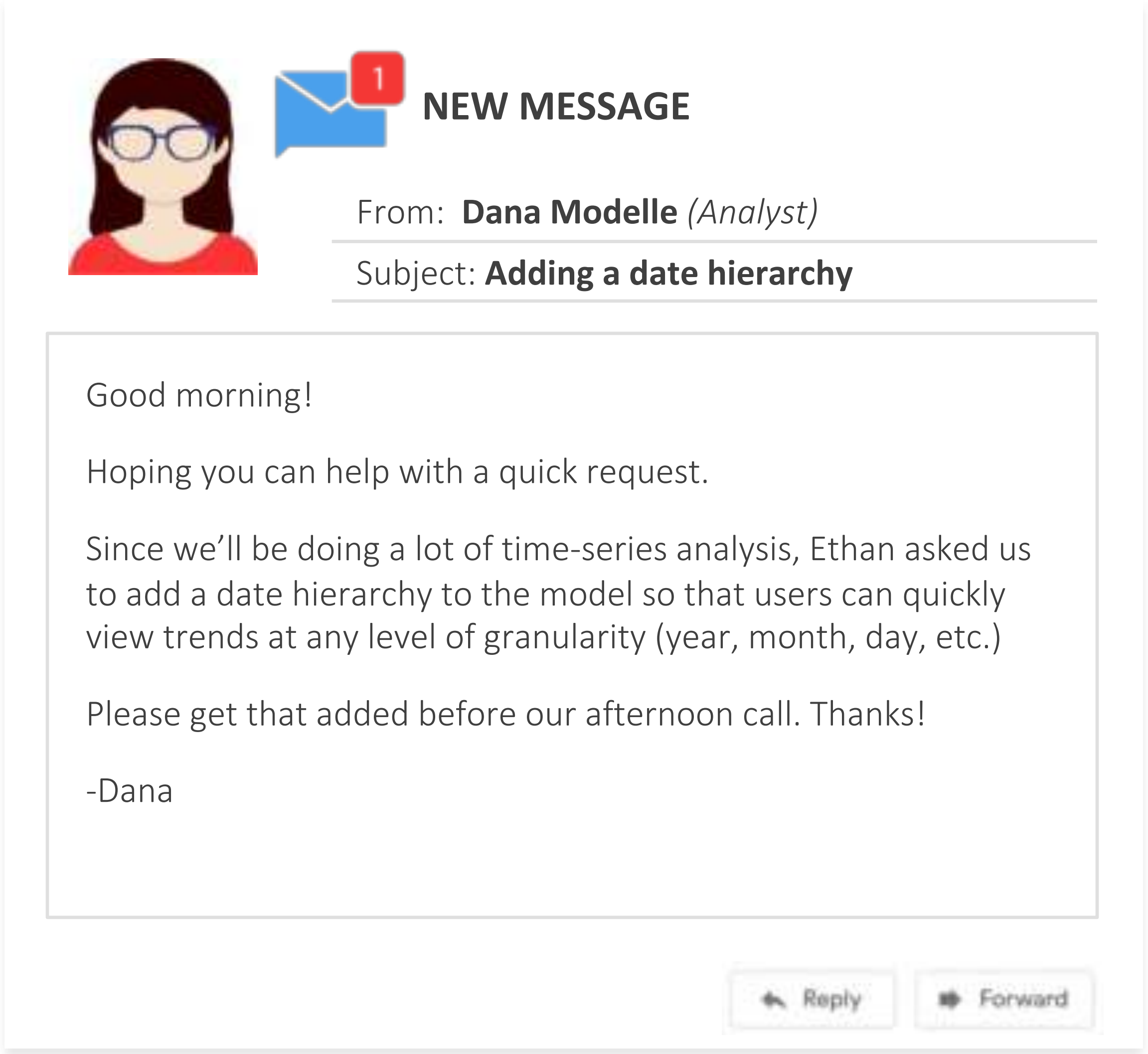
1. Create a new hierarchy based on the **Start of Year** field, and name it "**Date Hierarchy"**

2. Right-click or drag to add fields until your hierarchy contains the following (in this order):

   - **Start of Year**
   - **Start of Month**
   - **Start of Week**
   - **Date**

3. Add your new hierarchy to the matrix visual (on rows) and practice drilling up and down between each level of granularity

SOLUTION: HIERARCHIES



NEW MESSAGE

From: **Dana Modelle** *(Analyst)*

Subject: **Adding a date hierarchy**

Good morning!

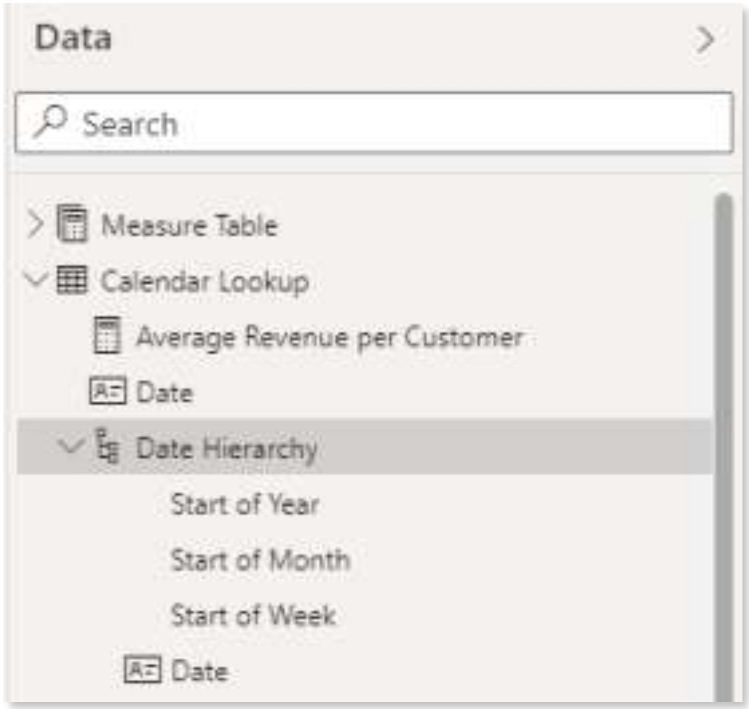
Hoping you can help with a quick request.

Since we'll be doing a lot of time-series analysis, Ethan asked us to add a date hierarchy to the model so that users can quickly view trends at any level of granularity (year, month, day, etc.)

Please get that added before our afternoon call. Thanks!

-Dana

Solution Preview



Data

Search

- Measure Table
- Calendar Lookup
  - Average Revenue per Customer
  - Date
  - Date Hierarchy
    - Start of Year
    - Start of Month
    - Start of Week
    - Date

DATA MODEL BEST PRACTICES

⭐ Focus on building a normalized model from the start

- *Leverage relationships and make sure that each table serves a clear, distinct purpose*

⭐ Organize dimension tables above data tables in your model

- *This serves as a visual reminder that filters always flow "downstream"*

⭐ Avoid complex relationships unless absolutely necessary

- *Aim to use 1-to-many table relationships and one-way filters whenever possible*

⭐ Hide fields from report view to prevent invalid filter context

- *This forces report users to filter using primary keys from dimension tables*

CALCULATED FIELDS WITH DAX

CALCULATED FIELDS WITH DAX

In this section we'll use **Data Analysis Expressions (DAX)** to add calculated columns & measures to our model, and introduce topics like row & filter context, iterators and more

TOPICS WE'LL COVER:

- DAX 101
- Columns & Measures
- Row & Filter Context
- DAX Syntax
- Common Functions
- Calculate
- Iterators
- Time Intelligence

GOALS FOR THIS SECTION:

- Introduce DAX fundamentals and learn when to use calculated columns and measures
- Understand the difference between row context and filter context, and how they impact DAX calculations
- Learn DAX formula syntax, basic operators and common function categories *(math, logical, text, date/time, filter, etc.)*
- Explore nested functions, and more complex topics like iterators and time intelligence patterns

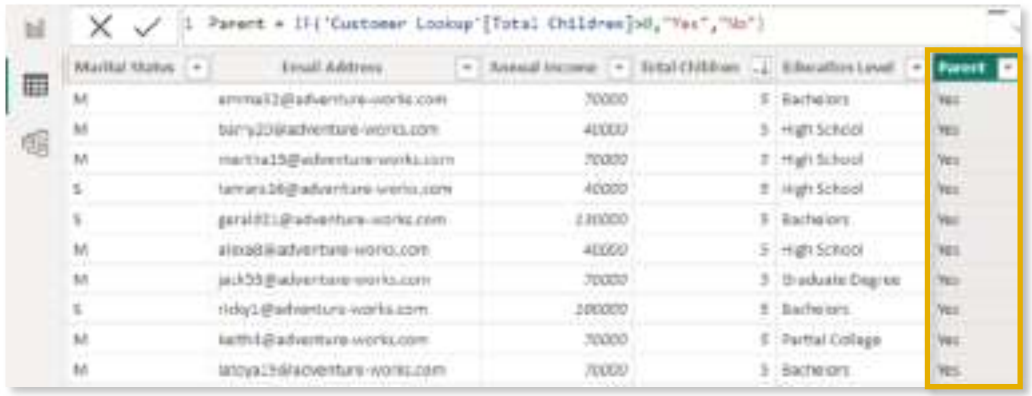
MEET DAX

Data Analysis Expressions (commonly known as **DAX**) is the formula language that drives the Power BI front-end. With DAX, you can:

- Go beyond the capabilities of traditional spreadsheet formulas, with powerful and flexible functions built specifically to work with relational data models
- Add **calculated columns** *(for filtering)* and **measures** *(for aggregation)* to enhance data models

Two ways to use DAX

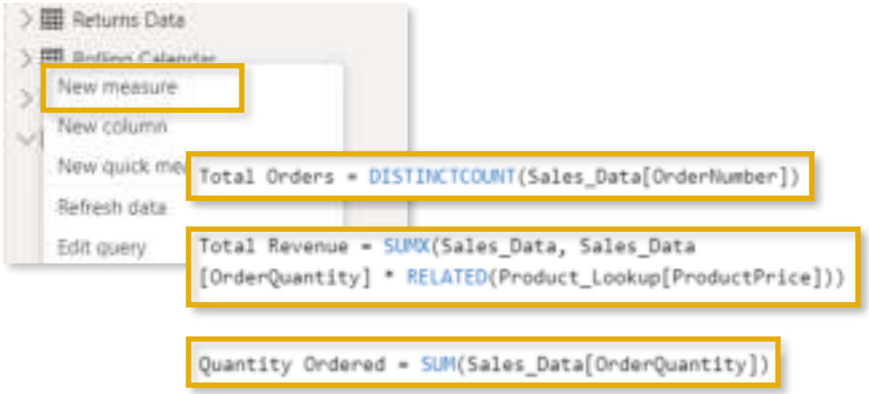
Calculated Columns



```
1 Parent = IF('Customer Lookup'[Total Children]>0,"Yes","No")
```

Marital Status	Email Address	Annual Income	Total Children	Education Level	Parent
M	emma13@adventure-works.com	70000	5	Bachelors	Yes
M	barry23@adventure-works.com	40000	3	High School	Yes
M	martha15@adventure-works.com	70000	2	High School	Yes
S	tamara26@adventure-works.com	40000	2	High School	Yes
S	gerald11@adventure-works.com	130000	5	Bachelors	Yes
M	alexa8@adventure-works.com	40000	5	High School	Yes
M	jack55@adventure-works.com	70000	3	Graduate Degree	Yes
S	ricky1@adventure-works.com	100000	2	Bachelors	Yes
M	keith1@adventure-works.com	70000	5	Partial College	Yes
M	latoya13@adventure-works.com	70000	3	Bachelors	Yes

Measures



```
Total Orders = DISTINCTCOUNT(Sales_Data[OrderNumber])
```

```
Total Revenue = SUMX(Sales_Data, Sales_Data
[OrderQuantity] * RELATED(Product_Lookup[ProductPrice]))
```

```
Quantity Ordered = SUM(Sales_Data[OrderQuantity])
```

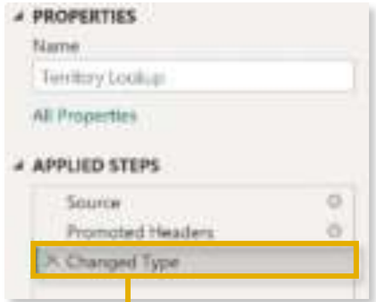
M VS. DAX

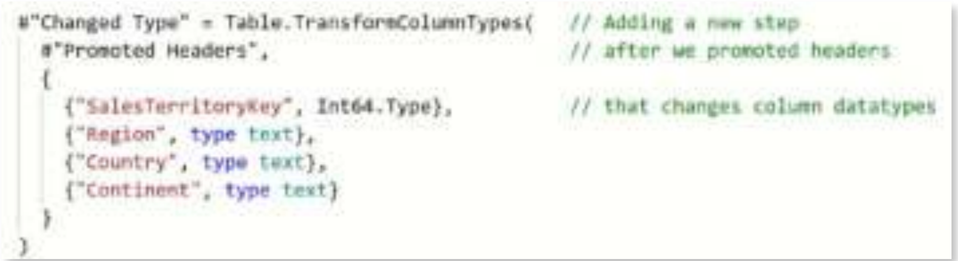
M and **DAX** are two distinct functional languages used within Power BI Desktop:

- **M** is used in the Power Query editor, and is designed specifically for extracting, transforming and loading data
- **DAX** is used in the Power BI front-end, and is designed specifically for analyzing relational data models

M

Query Editor:

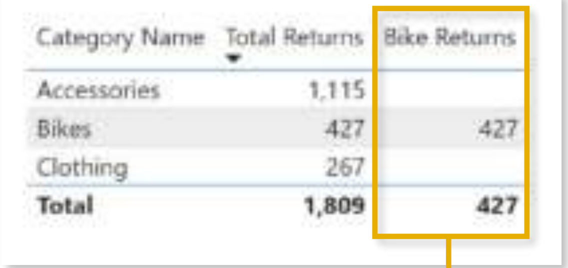




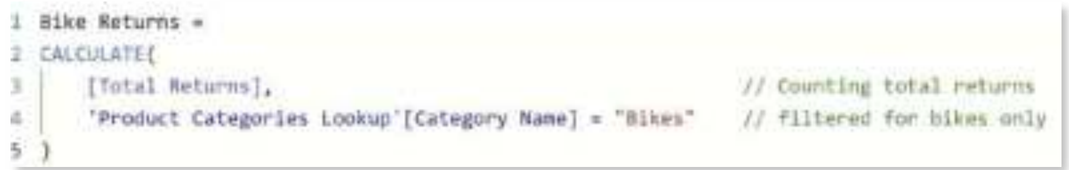
```
#"Changed Type" = Table.TransformColumnTypes(   // Adding a new step
  #"Promoted Headers",                          // after we promoted headers
  {
    {"SalesTerritoryKey", Int64.Type},          // that changes column datatypes
    {"Region", type text},
    {"Country", type text},
    {"Continent", type text}
  }
)
```

DAX

Report View:



Category Name	Total Returns	Bike Returns
Accessories	1,115	
Bikes	427	427
Clothing	267	
Total	**1,809**	**427**



```
Bike Returns =
CALCULATE(
    [Total Returns],                                            // Counting total returns
    'Product Categories Lookup'[Category Name] = "Bikes"       // filtered for bikes only
)
```

CALCULATED COLUMNS

Calculated columns allow you to add new, formula-based columns to tables in a model

- Calculated columns refer to **entire tables** or **columns** *(no A1-style cell references)*
- Calculated columns **generate values for each row**, which are visible within tables in the Data view
- Calculated columns understand **row context**; they're great for defining properties based on information in each row, but generally useless for aggregation *(sum, count, etc.)*

HEY THIS IS IMPORTANT!

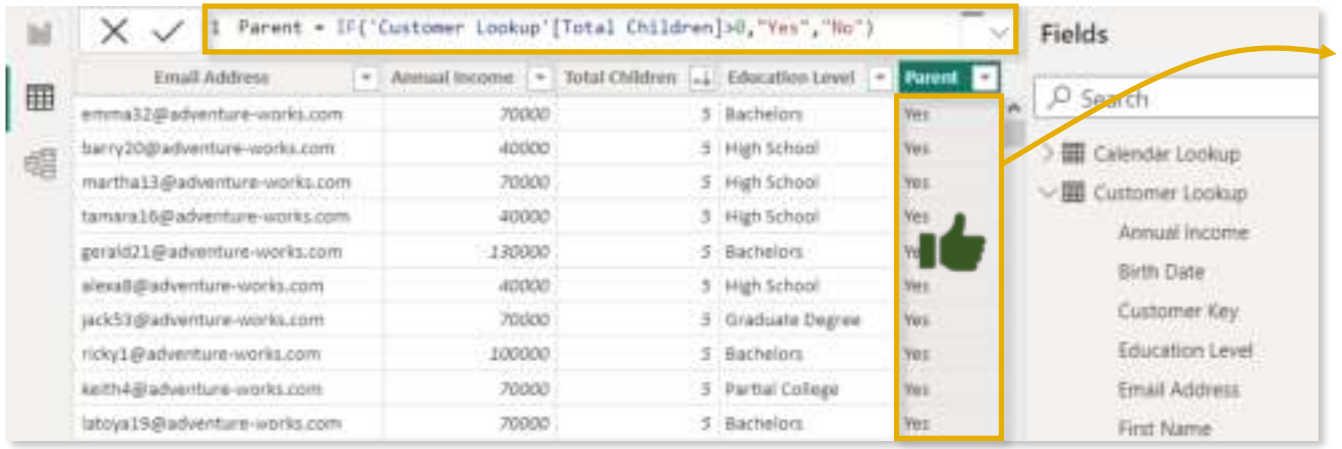
As a rule of thumb, use calculated columns to "stamp" static, fixed values to each row in a table (*or go upstream and use the Query Editor!*)

DO NOT use calculated columns for aggregation – this is what **measures** are for!

PRO TIP:

Calculated columns are typically used for **filtering** & **grouping** data, rather than creating aggregate numerical values

EXAMPLE: CALCULATED COLUMNS



```
1 Parent = IF('Customer Lookup'[Total Children]>0,"Yes","No")
```

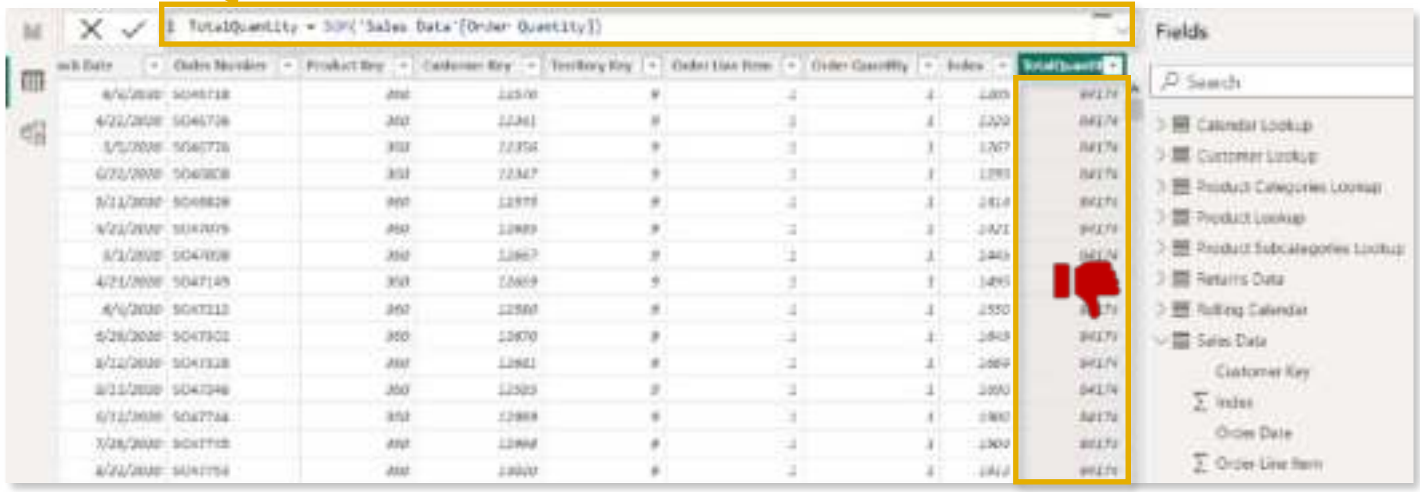
Email Address	Annual Income	Total Children	Education Level	Parent
emma32@adventure-works.com	70000	5	Bachelors	Yes
barry20@adventure-works.com	40000	5	High School	Yes
martha13@adventure-works.com	70000	5	High School	Yes
tamara16@adventure-works.com	40000	5	High School	Yes
gerald21@adventure-works.com	130000	5	Bachelors	Yes
alexa8@adventure-works.com	40000	5	High School	Yes
jack53@adventure-works.com	70000	5	Graduate Degree	Yes
ricky1@adventure-works.com	100000	5	Bachelors	Yes
keith4@adventure-works.com	70000	5	Partial College	Yes
latoya19@adventure-works.com	70000	5	Bachelors	Yes

In this case we've added a **calculated column** named **Parent**, which equals "***Yes***" if the [Total Children] field is greater than 0, and "***No***" otherwise

- Since calculated columns understand **row context**, a new value is calculated in each row based on the value in the [Total Children] column
- This is a **valid use** of calculated columns; it creates a new row "property" that we can use to filter or segment any related data within the model

Here we're using an aggregation function (SUM) to calculate a new column named **TotalQuantity**

- Since this is an aggregation function, **the same grand total** is returned in *every row* of the table
- This is **not a valid use** of calculated columns; these values are statically "stamped" onto the table and can't be filtered, sliced, etc.



```
1 TotalQuantity = SUM('Sales Data'[Order Quantity])
```

...ck Date	Order Number	Product Key	Customer Key	Territory Key	Order Line Item	Order Quantity	Index	TotalQuantity
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

DAX MEASURES

Measures are DAX formulas used to generate new calculated values

- Like calculated columns, measures reference **entire tables** or **columns** *(no A1-style cell references)*
- Unlike calculated columns, **measures** aren't visible within tables; they can only be "seen" within a visualization like a chart or matrix (*similar to a calculated field in a PivotTable*)
- Measures evaluate based on **filter context**, which means they recalculate when the fields or filters around them change

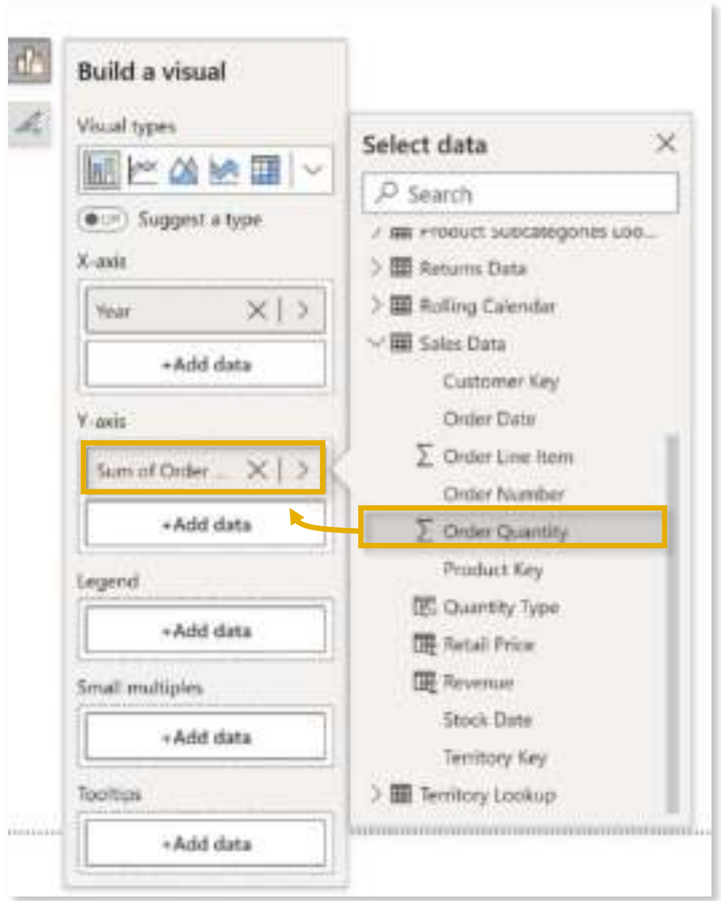
HEY THIS IS IMPORTANT!

As a rule of thumb, use measures when a single row can't give you the answer, or when you need to **aggregate** values across multiple rows in a table

PRO TIP:

Use measures to create **numerical, calculated values** that can be analyzed in the "**values**" field of a report visual

IMPLICIT VS. EXPLICIT MEASURES



Example of an ***implicit measure***

Implicit measures are created when you drag raw numerical fields into a report visual and manually select an aggregation mode (*Sum, Average, Min, Max, Count, etc.*)

Explicit measures are created when you actually write a DAX formula and define a new measure that can be used within the model

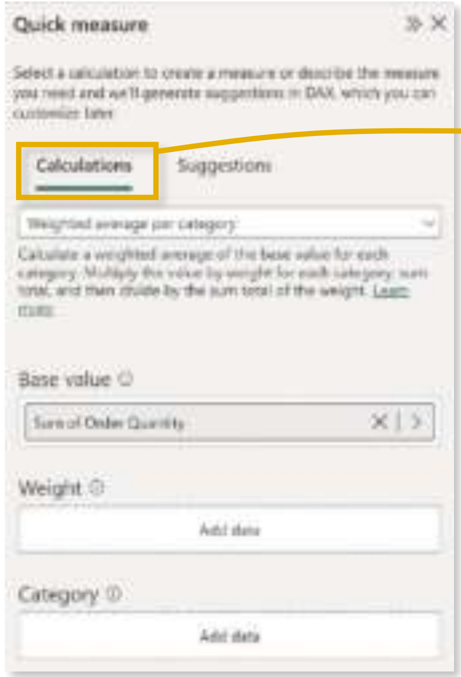
HEY THIS IS IMPORTANT!

Implicit measures are only accessible within the **specific visualization** in which they were created, and cannot be referenced elsewhere

Explicit measures can be used **anywhere in the report**, and referenced by other DAX calculations to create “measure trees”

QUICK MEASURES

Quick measures automatically create formulas based on pre-built templates or natural language prompts



Quick measure

Select a calculation to create a measure or describe the measure you need and we'll generate suggestions in DAX, which you can customize later.

Calculations | Suggestions

Weighted average per category

Calculate a weighted average of the base value for each category. Multiply the value by weight for each category, sum total, and then divide by the sum total of the weight. Learn more

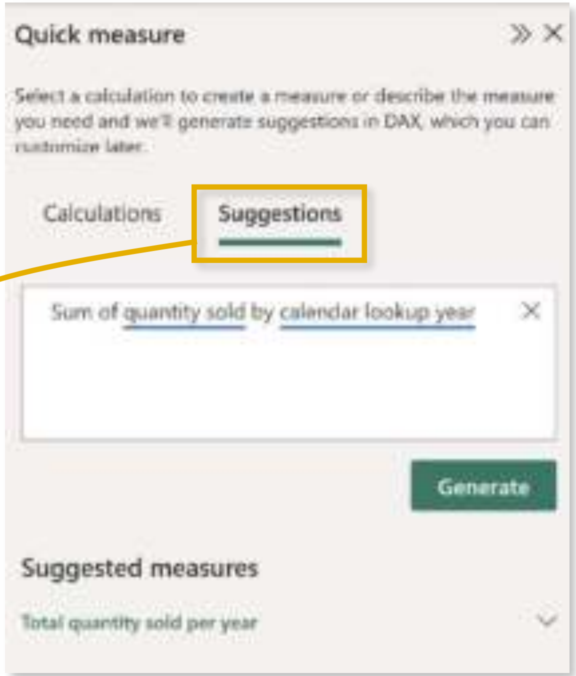
Base value: Sum of Order Quantity

Weight: Add data

Category: Add data

Quick measure **calculations** can be used to build measures using **predefined templates** *(weighted averages, percent difference, time intelligence, etc.)*

Quick measure **suggestions** can be used to find suggested measures based on **natural language queries** *(i.e. "sum of quantity sold by calendar year")*



Quick measure

Select a calculation to create a measure or describe the measure you need and we'll generate suggestions in DAX, which you can customize later.

Calculations | Suggestions

Sum of quantity sold by calendar lookup year

Generate

Suggested measures

Total quantity sold per year

PRO TIP:

Quick measures can be a great learning tool for beginners or for building more complex formulas but use them with caution; **mastering DAX requires a deep understanding of the underlying theory**!