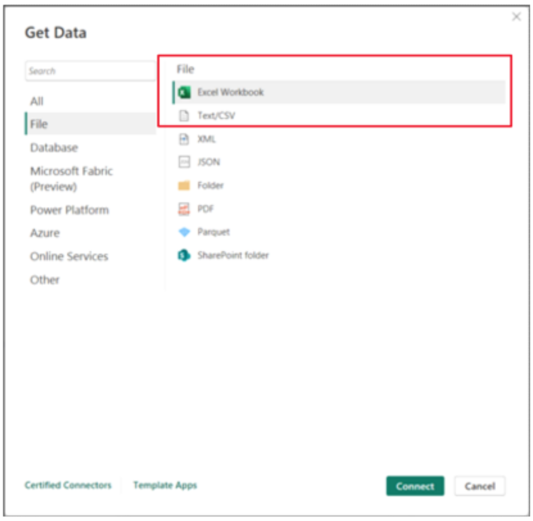
Power BI Key Summary Notes (Part 1)

Wednesday, 25 October 2023 9:51 pm

Getting Data into Power BI

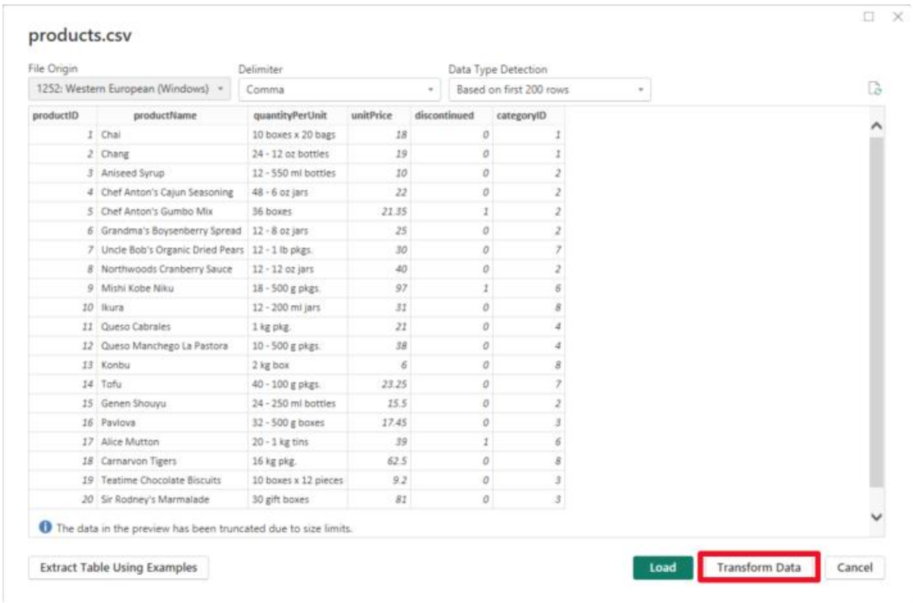


For the test, the format of the data set is most likely in excel or CSV.

CSV Format

If the test comprises of 6 tables and if the source files are in CSV format, you will have six separate CSV files. Each of these files will need to be loaded into power BI one at a time. There is no strict rule dictating the specific order in which you should load the files. However, the best practice is to load dimension tables before the fact tables.

- Click on the Load button if your dataset is ready for analysis and does not require any cleaning or transformation (rare).
- Otherwise, click on the Transform Data button, and you will directed to the Power Query Editor. This editor is where you can perform various data transformation and prepare your data for analysis in power BI.



products.csv

File Origin: 1252: Western European (Windows)

Delimiter: Comma

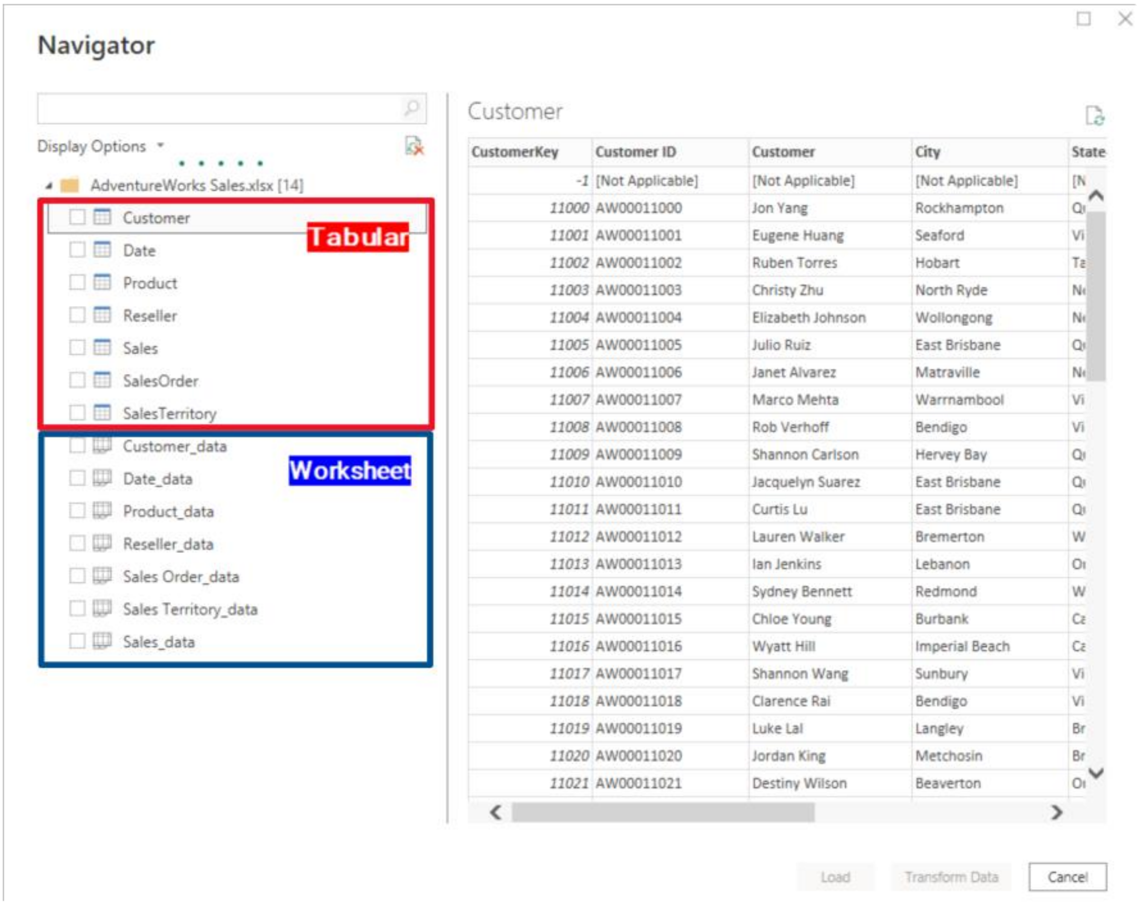
Data Type Detection: Based on first 200 rows

productID	productName	quantityPerUnit	unitPrice	discontinued	categoryID
1	Chai	10 boxes x 20 bags	18	0	1
2	Chang	24 - 12 oz bottles	19	0	1
3	Aniseed Syrup	12 - 550 ml bottles	10	0	2
4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	22	0	2
5	Chef Anton's Gumbo Mix	36 boxes	21.35	1	2
6	Grandma's Boysenberry Spread	12 - 8 oz jars	25	0	2
7	Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	30	0	7
8	Northwoods Cranberry Sauce	12 - 12 oz jars	40	0	2
9	Mishi Kobe Niku	18 - 500 g pkgs.	97	1	6
10	Ikura	12 - 200 ml jars	31	0	8
11	Queso Cabrales	1 kg pkg.	21	0	4
12	Queso Manchego La Pastora	10 - 500 g pkgs.	38	0	4
13	Konbu	2 kg box	6	0	8
14	Tofu	40 - 100 g pkgs.	23.25	0	7
15	Genen Shouyu	24 - 250 ml bottles	15.5	0	2
16	Pavlova	32 - 500 g boxes	17.45	0	3
17	Alice Mutton	20 - 1 kg tins	39	1	6
18	Carnarvon Tigers	16 kg pkg.	62.5	0	8
19	Teatime Chocolate Biscuits	10 boxes x 12 pieces	9.2	0	3
20	Sir Rodney's Marmalade	30 gift boxes	81	0	3

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

Excel Format

If the source file is in excel format, it is quite likely that all the tables will be in the same excel document, and separated by worksheets. There is also a likelihood that the data in the excel file is already in tabular format. In this situation, you will see a set of tables and a set of worksheets during the data loading process.



Navigator

Display Options

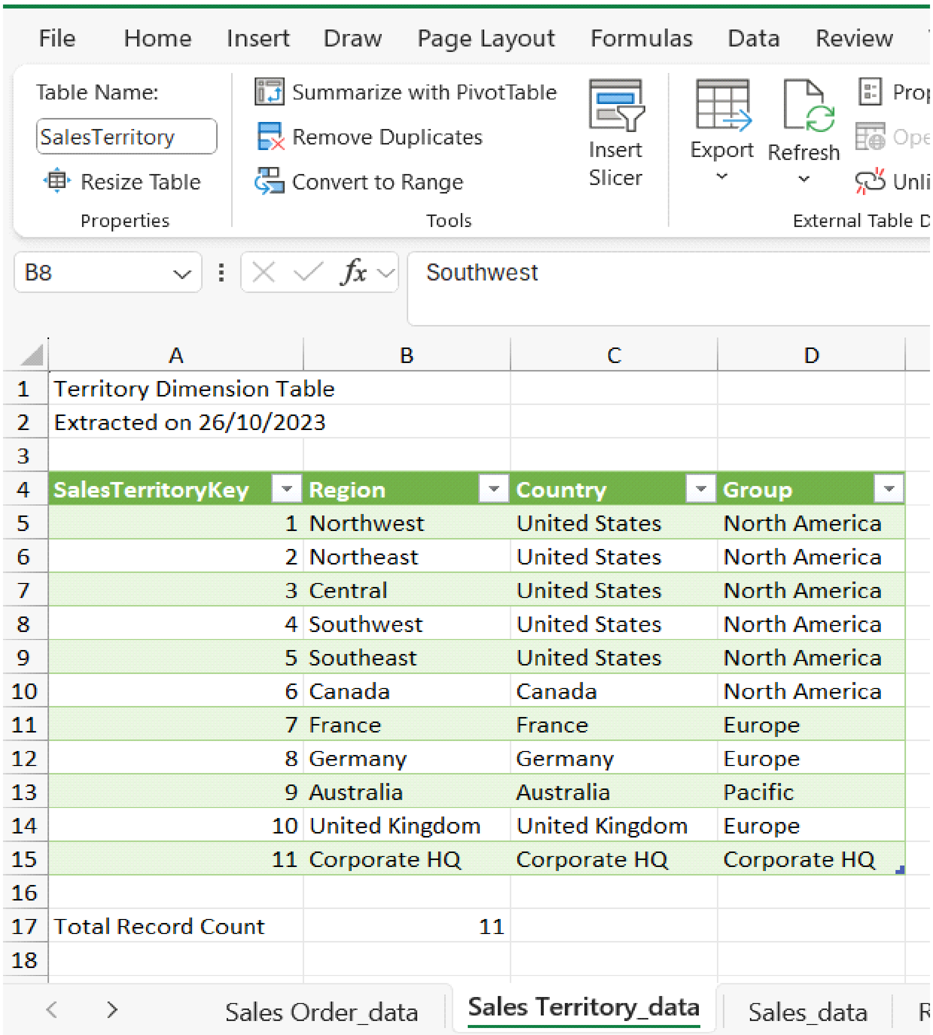
AdventureWorks Sales.xlsx [14]

- □ Customer
- □ Date
- □ Product
- □ Reseller
- □ Sales
- □ SalesOrder
- □ SalesTerritory
- □ Customer_data
- □ Date_data
- □ Product_data
- □ Reseller_data
- □ Sales Order_data
- □ Sales Territory_data
- □ Sales_data

Customer

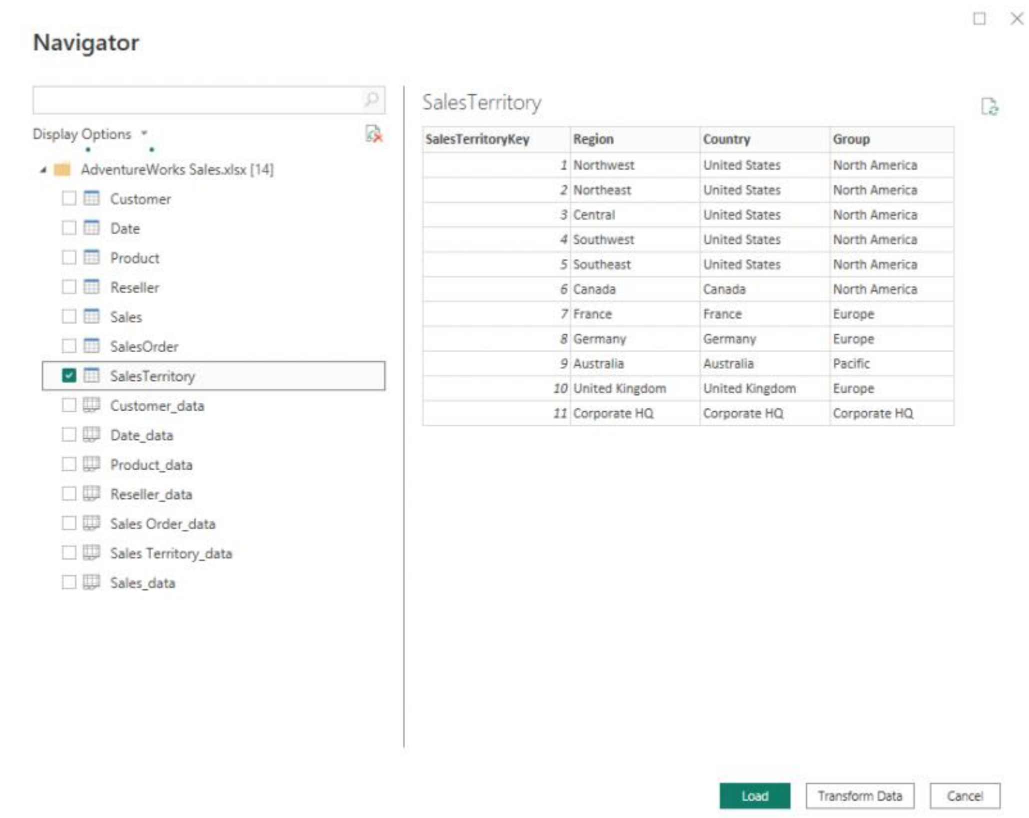
CustomerKey	Customer ID	Customer	City	State
-1	[Not Applicable]	[Not Applicable]	[Not Applicable]	[N
11000	AW00011000	Jon Yang	Rockhampton	Q
11001	AW00011001	Eugene Huang	Seaford	Vi
11002	AW00011002	Ruben Torres	Hobart	Ta
11003	AW00011003	Christy Zhu	North Ryde	N
11004	AW00011004	Elizabeth Johnson	Wollongong	N
11005	AW00011005	Julio Ruiz	East Brisbane	Q
11006	AW00011006	Janet Alvarez	Matraville	N
11007	AW00011007	Marco Mehta	Warrnambool	Vi
11008	AW00011008	Rob Verhoff	Bendigo	Vi
11009	AW00011009	Shannon Carlson	Hervey Bay	Q
11010	AW00011010	Jacquelyn Suarez	East Brisbane	Q
11011	AW00011011	Curtis Lu	East Brisbane	Q
11012	AW00011012	Lauren Walker	Bremerton	W
11013	AW00011013	Ian Jenkins	Lebanon	O
11014	AW00011014	Sydney Bennett	Redmond	W
11015	AW00011015	Chloe Young	Burbank	Ca
11016	AW00011016	Wyatt Hill	Imperial Beach	Ca
11017	AW00011017	Shannon Wang	Sunbury	Vi
11018	AW00011018	Clarence Rai	Bendigo	Vi
11019	AW00011019	Luke Lal	Langley	Br
11020	AW00011020	Jordan King	Metchosin	Br
11021	AW00011021	Destiny Wilson	Beaverton	O

If the data is already in a structured tabular format within the worksheets, it's a good practice to select and load these tabular sets directly (red).



B8 = Southwest

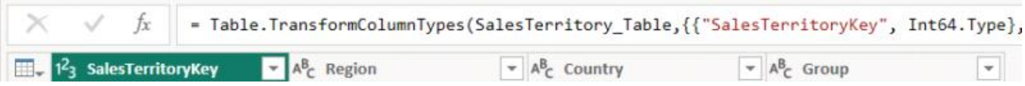
	A	B	C	D
1	Territory Dimension Table			
2	Extracted on 26/10/2023			
3				
4	SalesTerritoryKey	Region	Country	Group
5	1	Northwest	United States	North America
6	2	Northeast	United States	North America
7	3	Central	United States	North America
8	4	Southwest	United States	North America
9	5	Southeast	United States	North America
10	6	Canada	Canada	North America
11	7	France	France	Europe
12	8	Germany	Germany	Europe
13	9	Australia	Australia	Pacific
14	10	United Kingdom	United Kingdom	Europe
15	11	Corporate HQ	Corporate HQ	Corporate HQ
16				
17	Total Record Count	11		
18				



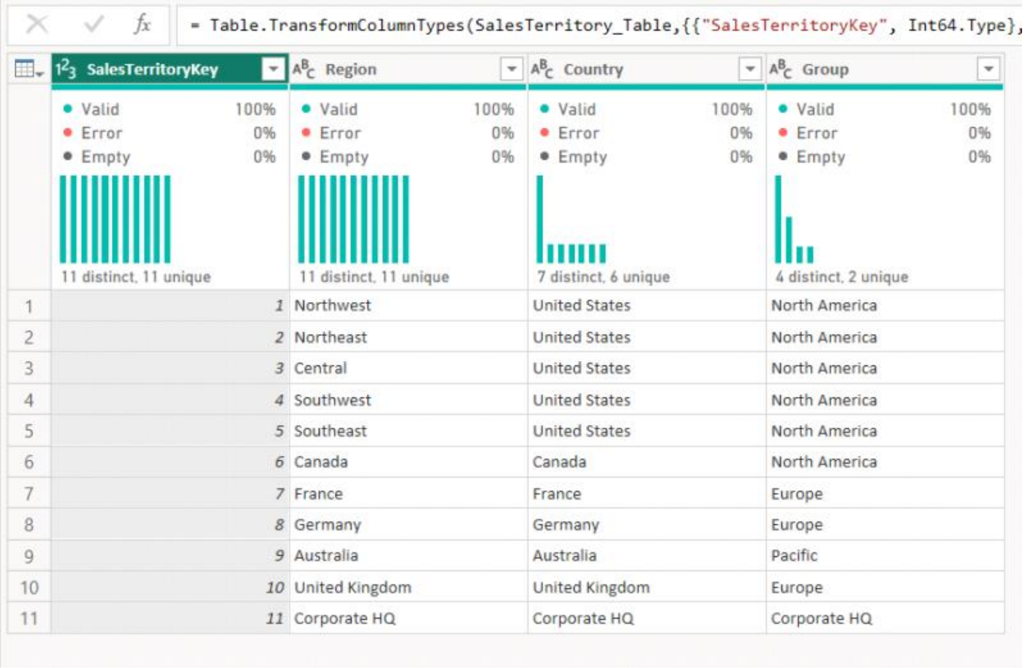
- ☐ Customer
- ☐ Date
- ☐ Product
- ☐ Reseller
- ☐ Sales
- ☐ SalesOrder
- ☑ SalesTerritory
- ☐ Customer_data
- ☐ Date_data
- ☐ Product_data
- ☐ Reseller_data
- ☐ Sales Order_data
- ☐ Sales Territory_data
- ☐ Sales_data

SalesTerritory

SalesTerritoryKey	Region	Country	Group
1	Northwest	United States	North America
2	Northeast	United States	North America
3	Central	United States	North America
4	Southwest	United States	North America
5	Southeast	United States	North America
6	Canada	Canada	North America
7	France	France	Europe
8	Germany	Germany	Europe
9	Australia	Australia	Pacific
10	United Kingdom	United Kingdom	Europe
11	Corporate HQ	Corporate HQ	Corporate HQ



```
= Table.TransformColumnTypes(SalesTerritory_Table,{{"SalesTerritoryKey", Int64.Type},
```

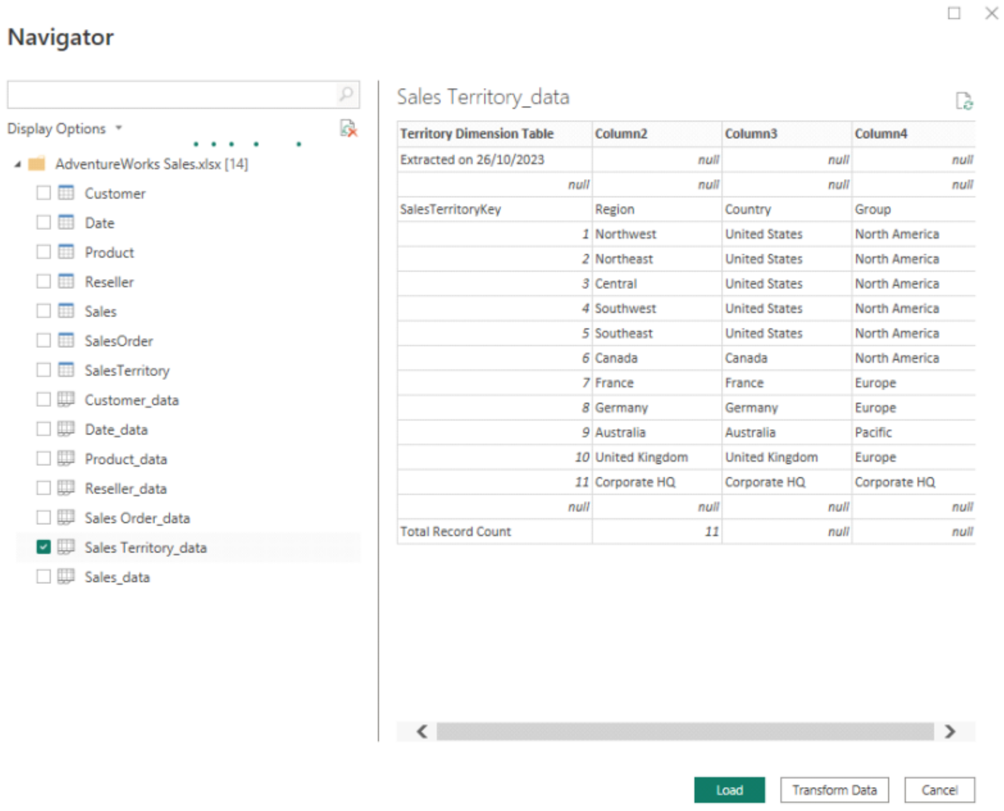


```
= Table.TransformColumnTypes(SalesTerritory_Table,{{"SalesTerritoryKey", Int64.Type},
```

	1²3 SalesTerritoryKey	A^B_C Region	A^B_C Country	A^B_C Group
	● Valid 100% ● Error 0% ● Empty 0%	● Valid 100% ● Error 0% ● Empty 0%	● Valid 100% ● Error 0% ● Empty 0%	● Valid 100% ● Error 0% ● Empty 0%
	11 distinct, 11 unique	11 distinct, 11 unique	7 distinct, 6 unique	4 distinct, 2 unique
1	1	Northwest	United States	North America
2	2	Northeast	United States	North America
3	3	Central	United States	North America
4	4	Southwest	United States	North America
5	5	Southeast	United States	North America
6	6	Canada	Canada	North America
7	7	France	France	Europe
8	8	Germany	Germany	Europe
9	9	Australia	Australia	Pacific
10	10	United Kingdom	United Kingdom	Europe
11	11	Corporate HQ	Corporate HQ	Corporate HQ

Loading data from worksheets (blue) will also import content beyond the data tables. This content is typically irrelevant for data analysis and may require additional transformations.

For example - If you select the Sales Territory_data worksheet, you will need to perform additional transformations such as Remove Top 2 Rows, Remove Bottom 2 Rows, Use First Row as Headers.



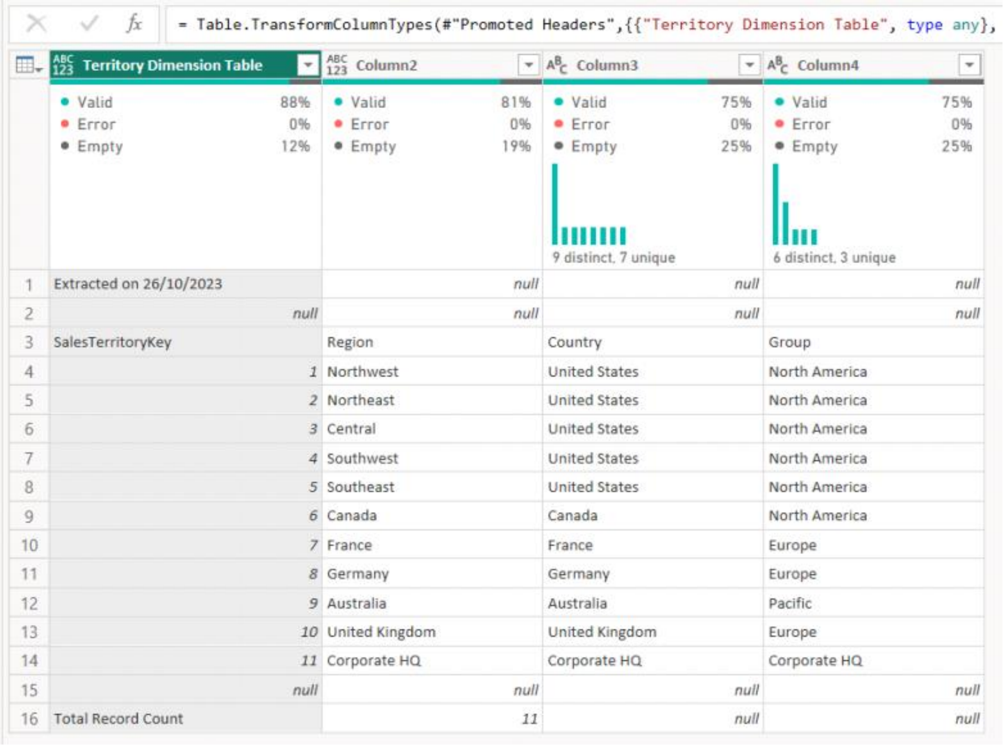
Navigator

Display Options ▾

◢ AdventureWorks Sales.xlsx [14]
- □ Customer
- □ Date
- □ Product
- □ Reseller
- □ Sales
- □ SalesOrder
- □ SalesTerritory
- □ Customer_data
- □ Date_data
- □ Product_data
- □ Reseller_data
- □ Sales Order_data
- ☑ Sales Territory_data
- □ Sales_data

Sales Territory_data

Territory Dimension Table	Column2	Column3	Column4
Extracted on 26/10/2023	null	null	null
null	null	null	null
SalesTerritoryKey	Region	Country	Group
1	Northwest	United States	North America
2	Northeast	United States	North America
3	Central	United States	North America
4	Southwest	United States	North America
5	Southeast	United States	North America
6	Canada	Canada	North America
7	France	France	Europe
8	Germany	Germany	Europe
9	Australia	Australia	Pacific
10	United Kingdom	United Kingdom	Europe
11	Corporate HQ	Corporate HQ	Corporate HQ
null	null	null	null
Total Record Count	11	null	null



```
= Table.TransformColumnTypes(#"Promoted Headers",{{"Territory Dimension Table", type any},
```

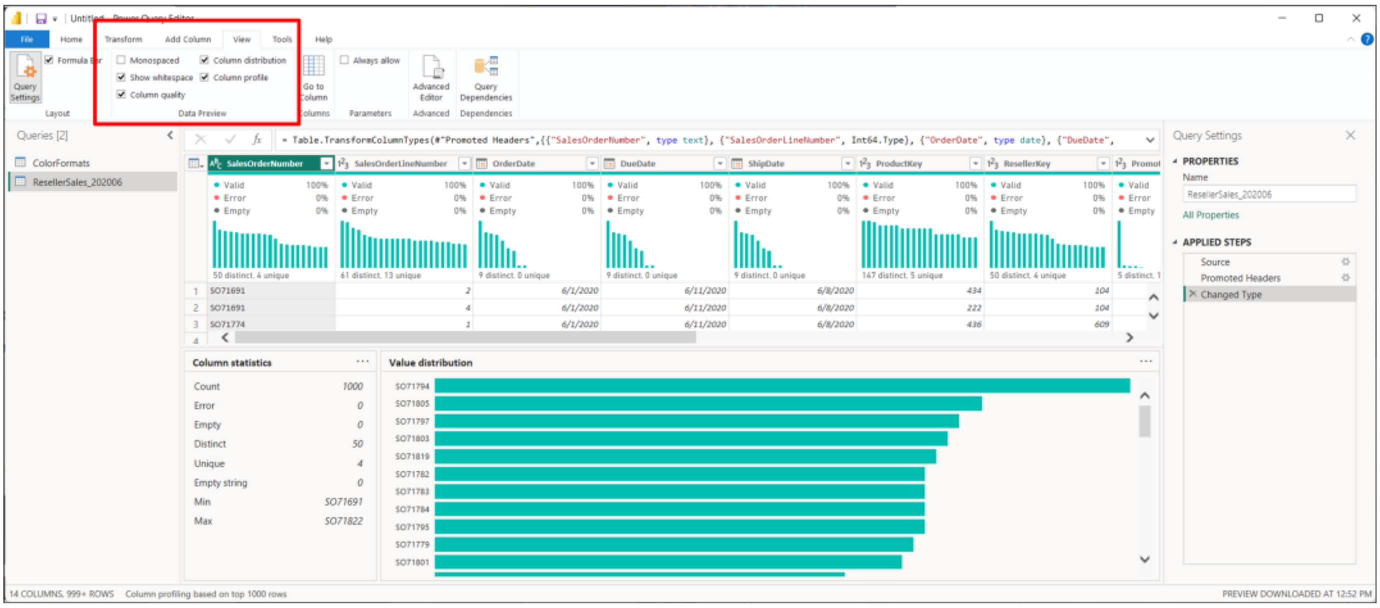
	Territory Dimension Table	Column2	Column3	Column4
	● Valid 88% ● Error 0% ● Empty 12%	● Valid 81% ● Error 0% ● Empty 19%	● Valid 75% ● Error 0% ● Empty 25% 9 distinct, 7 unique	● Valid 75% ● Error 0% ● Empty 25% 6 distinct, 3 unique
1	Extracted on 26/10/2023	null	null	null
2	null	null	null	null
3	SalesTerritoryKey	Region	Country	Group
4	1	Northwest	United States	North America
5	2	Northeast	United States	North America
6	3	Central	United States	North America
7	4	Southwest	United States	North America
8	5	Southeast	United States	North America
9	6	Canada	Canada	North America
10	7	France	France	Europe
11	8	Germany	Germany	Europe
12	9	Australia	Australia	Pacific
13	10	United Kingdom	United Kingdom	Europe
14	11	Corporate HQ	Corporate HQ	Corporate HQ
15	null	null	null	null
16	Total Record Count	11	null	null

To sum it up, when presented with the choice between tabular and worksheet options, go with the **tabular option**. **Avoid selecting both options**, as doing so can result in duplicated tables, additional system storage, and the import of extraneous data that is not needed for your analysis.

Explore, Clean and Transform Data

Profiling data is about studying the nuances of the data: determining anomalies, examining and developing the underlying data structures, and querying data statistics such as row counts, value distributions, minimum and maximum values, averages, and so on. This concept is important because it allows you to shape and organize the data so that interacting with the data and identifying the distribution of the data is uncomplicated, therefore helping to make your task of working with the data on the front end to develop report elements near effortless.

Using Excel in combination with Power BI for data exploration can be a valuable approach, especially when the dataset is in CSV or Excel format. Excel allows you to quickly open and view the dataset, making it easy to assess the data's structure, conten t and quality. NOTE : **Create a copy of the original dataset for data exploration** to ensure that the original dataset (which is used by Power BI) remains intact



☑ Formula Bar
☐ Monospaced
☑ Show whitespace
☑ Column quality
☑ Column distribution
☑ Column profile
☐ Always allow

```
= Table.TransformColumnTypes(#"Promoted Headers",{{"SalesOrderNumber", type text}, {"SalesOrderLineNumber", Int64.Type}, {"OrderDate", type date}, {"DueDate",
```

	SalesOrderNumber	SalesOrderLineNumber	OrderDate	DueDate	ShipDate	ProductKey	ResellerKey	Promot
	● Valid 100% ● Error 0% ● Empty 0% 50 distinct, 4 unique	● Valid 100% ● Error 0% ● Empty 0% 61 distinct, 13 unique	● Valid 100% ● Error 0% ● Empty 0% 9 distinct, 0 unique	● Valid 100% ● Error 0% ● Empty 0% 9 distinct, 0 unique	● Valid 100% ● Error 0% ● Empty 0% 9 distinct, 0 unique	● Valid 100% ● Error 0% ● Empty 0% 147 distinct, 5 unique	● Valid 100% ● Error 0% ● Empty 0% 50 distinct, 4 unique	● Valid ● Error ● Empty 5 distinct, 1
1	SO71691	2	6/1/2020	6/11/2020	6/8/2020	434	104	
2	SO71691	4	6/1/2020	6/11/2020	6/8/2020	222	104	
3	SO71774	1	6/1/2020	6/11/2020	6/8/2020	436	609	
4								

Column statistics

Count	1000
Error	0
Empty	0
Distinct	50
Unique	4
Empty string	0
Min	SO71691
Max	SO71822

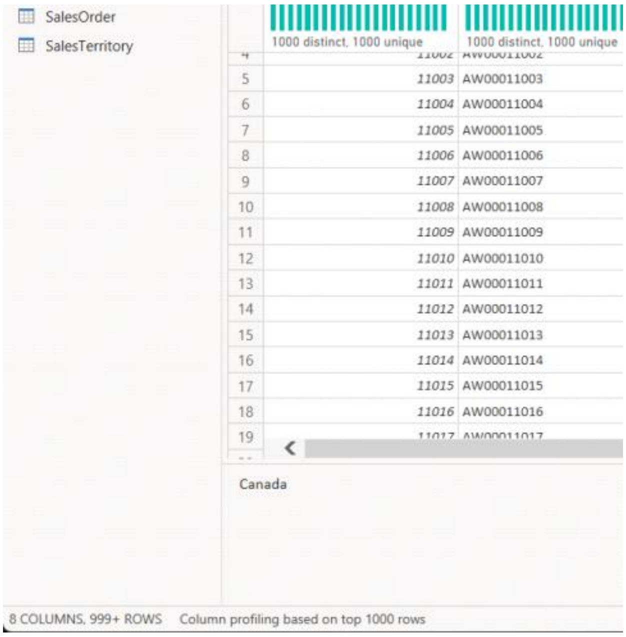
Value distribution

14 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 12:52 PM

- Column distribution shows you the distribution of the data within the column
- Column profile gives you more details about the data in the column, including basic statistics
- Value distribution graph tells you the counts for each unique value in that specific column.
- Column Statistics will also include how many zeroes and null values exist, along with the average value in the column, the standard deviation of the values in the column, and how many even and odd values are in the column.

NOTE : The default column profiling is based on the top 1000 rows. You should change this to "Column profiling based on entire data set" for a more accurate analysis.

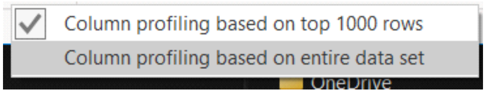


SalesOrder
SalesTerritory

	1000 distinct, 1000 unique	1000 distinct, 1000 unique
5	11003	AW00011003
6	11004	AW00011004
7	11005	AW00011005
8	11006	AW00011006
9	11007	AW00011007
10	11008	AW00011008
11	11009	AW00011009
12	11010	AW00011010
13	11011	AW00011011
14	11012	AW00011012
15	11013	AW00011013
16	11014	AW00011014
17	11015	AW00011015
18	11016	AW00011016

Canada

8 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows



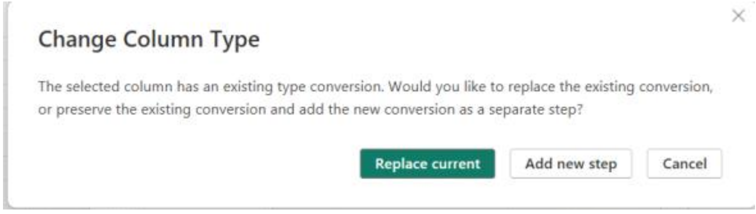
☑ Column profiling based on top 1000 rows
Column profiling based on entire data set

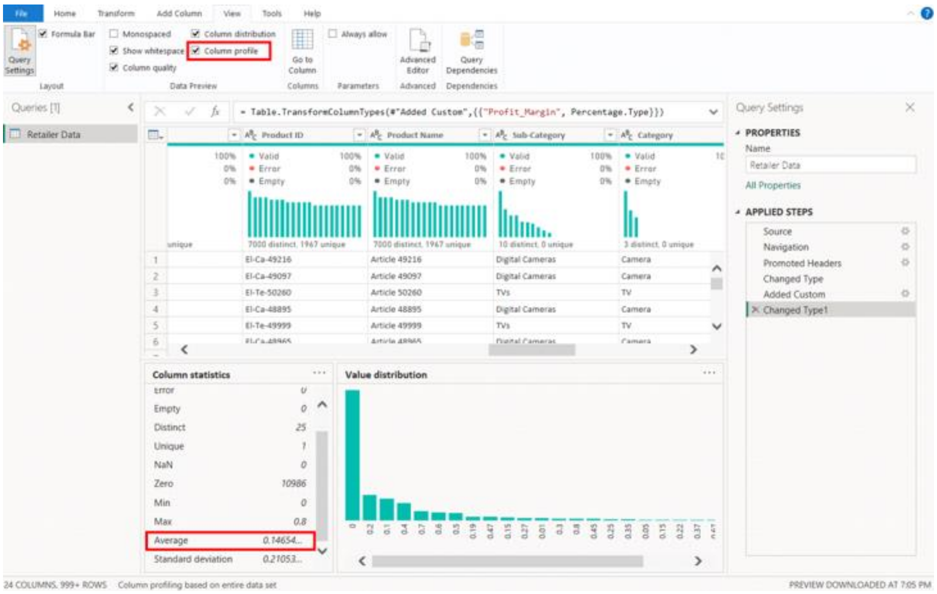
Cardinality is a term that is used to describe the uniqueness of the values in a column. Cardinality is also used in the context of the relationships between two tables, where it describes the direction of the relationship.

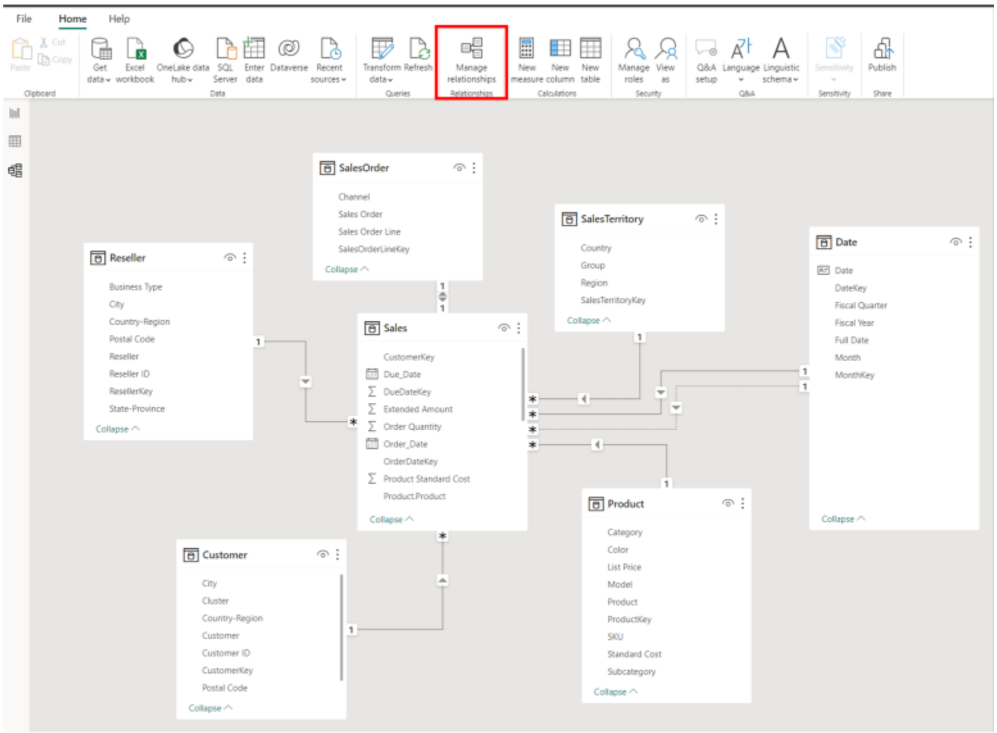
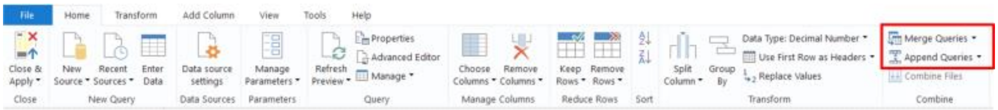
- Distinct - The total number of unique values in the column
  SQL representation: Distinct = SELECT DISTINCT column from table;

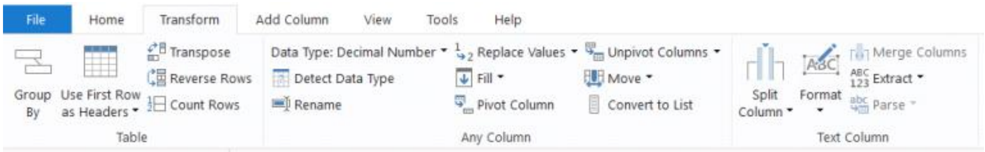
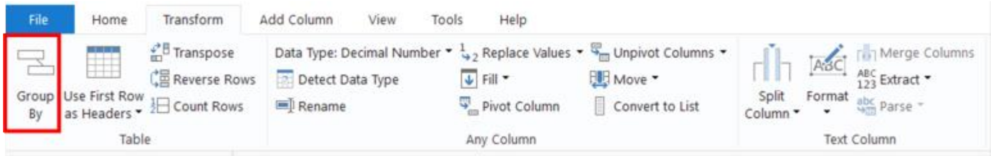
- Unique - The total number of distinct values that only appear once
- SQL representation : Distinct = SELECT DISTINCT column from table WHERE COUNT(DISTINCT column) = 1;

Data Exploration Checklist

Task	What to Check?	Potential Resolution
Data Overview	Verify the number of records and columns	N/A
Data Types	Confirm that data types are appropriate	Adjust column data types if needed. If prompted with the "Change Type" prompt and with options "Replace current" and "Add new step", choose "Add new step". This will provide flexibility to undo or redo the step, without impacting the data types for the rest of the columns.  Change Column Type The selected column has an existing type conversion. Would you like to replace the existing conversion, or preserve the existing conversion and add the new conversion as a separate step? Replace current / Add new step / Cancel

		1. Select the column 2. Go to the "Transform" tab 3. Click "Data Type" and choose the appropriate data type
Column Names	Ensure column names are clear and consistent	Rename columns for clarity. 1. Select the column 2. Go to the "Transform" tab 3. Click "Rename" and enter the new name for the column
Missing values	Identify columns with missing data	Handle missing values with care. DO NOT immediately drop rows with missing values as these rows may contain valuable information, if discarded, could lead to loss of important insights. - Determine the percentage of Missing Data - If a column has a high percentage of missing data, replace the missing data with an appropriate value may be more reasonable than dropping them - Understand why the data is missing. Is it missing at random, or is there a pattern or reason behind the missing data? - Consider the potential impact of missing values on your analysis or reporting e.g. the impact on sales revenue reporting if a customer is removed from the dataset If the decision is to drop the rows with missing values 1. Select the column with missing values 2. Click on the filter icon 3. Uncheck the "null" or "blank" value If the decision is to replace the missing value with an appropriate value such as mean, median, mode (for categorical column). You can easily compute these statistics in excel, using AVERAGE(column), MEDIAN(column) and MODE(column) or use the "Column Profile" option to get the mean statistics 1. Select the column with missing values 2. Go to the "Transform" tab 3. Click "Replace Values" 4. For Value To Find, enter null 5. For Replace With, enter the mean, median or mode value  ☑ Formula Bar ☐ Monospaced ☑ Show whitespace ☑ Column quality ☑ Column distribution ☑ Column profile ☐ Always allow = Table.TransformColumnTypes(#"Added Custom",{{"Profit_Margin", Percentage.Type}}) Column statistics Error 0 Empty 0 Distinct 25 Unique 1 NaN 0 Zero 10986 Min 0 Max 0.8 Average 0.14654... Standard deviation 0.21053... APPLIED STEPS: Source, Navigation, Promoted Headers, Changed Type, Added Custom, Changed Type1 24 COLUMNS, 999+ ROWS Column profiling based on entire data set PREVIEW DOWNLOADED AT 7:05 PM
Duplicates	Identify duplicated columns and rows	Duplicate Columns 1. Right click on one of the duplicate columns that you want to remove 2. Select "Remove" Duplicate Rows 1. In the Queries pane on the left, select the table that contains the duplicate rows 2. Go to the "Home" tab 3. Click on "Remove Rows" 4. Choose "Remove Duplicates"
Outliers	Outliers are data points that deviate significantly from	For now, you can assume the outliers are genuine values unless they are considered implausible or too far fetched

	the rest of the data and can have a substantial impact on your analysis e.g. age = 200 years old	
Data Validation	Ensure that the data is consistent and does not contain any obvious errors, such as negative order quantities or prices Validate the data conforms to expected business rules e.g. order quantities must be greater than zero	Depending on the data validation issues
Data Relationships	Verify that relationships between tables are correctly established and ensure the data relationships support the intended analysis se the "Model" view to visualize, review and validate the relationships between tables. • Review and validate foreign key relationships • Ensure referential integrity by checking that linked columns have matching values.	Use the "Model" view and "Manage relationships" to define, edit or delete the relationships between tables  File Home Help Paste Cut Copy Clipboard Get data Excel workbook OneLake data hub SQL Server Enter data Dataverse Recent sources Data Transform data Refresh Queries Manage relationships Relationships New measure New column New table Calculations Manage roles View as Security Q&A setup Language Linguistic schema Q&A Sensitivity Publish Share SalesOrder Channel Sales Order Sales Order Line SalesOrderLineKey SalesTerritory Country Group Region SalesTerritoryKey Date Date DateKey Fiscal Quarter Fiscal Year Full Date Month MonthKey Reseller Business Type City Country-Region Postal Code Reseller Reseller ID ResellerKey State-Province Sales CustomerKey Due_Date DueDateKey Extended Amount Order Quantity Order_Date OrderDateKey Product Standard Cost Product.Product Product Category Color List Price Model Product ProductKey SKU Standard Cost Subcategory Customer City Cluster Country-Region Customer Customer ID CustomerKey Postal Code Collapse 1 * Use "Merge Queries" and "Append Queries" if there is a need to combine data from multiple tables.  File Home Transform Add Column View Tools Help Close & Apply Close New Source Recent Sources Enter Data New Query Data source settings Data Sources Manage Parameters Parameters Refresh Preview Properties Advanced Editor Manage Query Choose Columns Remove Columns Manage Columns Keep Rows Remove Rows Reduce Rows Sort Split Column Group By Data Type: Decimal Number Use First Row as Headers Replace Values Transform Merge Queries Append Queries Combine Files Combine **Merge Queries** 1. Under the Home tab, click on the "Merge Queries" dropdown 2. Choose the "Merge Queries" or "Merge Queries as New" 3. In the Merge dialogue, select the two tables that you want to merge 4. Choose the columns in each table that will serve as the keys for the merge 5. Define the type of join • When you merge queries, you are **combining the data from multiple tables into one based on a column that is common between the tables**. This process is **similar to the JOIN clause in SQL**. Essentially, you will be adding columns from one table (or query) into another. To merge two tables, you must have a column that is the key between the two tables. •

		<u>**Append Queries**</u> 1. Under the Home tab, click on the "Append Queries" dropdown 2. Choose the "Append Queries" or "Append Queries as New" 3. In the Append dialogue, select the tables that you want to append - When you append queries, you will be **adding rows of data to another table or query** - **Append queries will NOT remove duplicates** - **Append works best when the columns match exactly**. If columns in source queries are different, append still works, but will create one column in the output per each new column, if one of the sources does not have that column the cell value of that column for those rows will be null
Data Standardization	Review if the data is standardized across columns e.g. column names, data values (especially units of measures)	Use the "Transform" features to standardize the data  File Home Transform Add Column View Tools Help Group By Use First Row as Headers Transpose Reverse Rows Count Rows Data Type: Decimal Number Detect Data Type Rename Replace Values Fill Pivot Column Unpivot Columns Move Convert to List Split Column Format Merge Columns Extract Parse Table Any Column Text Column
Data Enrichment	- Need to create aggregated data for reporting. - Creating hierarchies for drill-down capabilities. - Need to create calculated columns for custom calculations.	<u>**New Aggregated Column**</u> Use the "Group By" transformation to group data by one or more columns and apply aggregation e.g. count, count distinct, sum, average  File Home Transform Add Column View Tools Help Group By Use First Row as Headers Transpose Reverse Rows Count Rows Data Type: Decimal Number Detect Data Type Rename Replace Values Fill Pivot Column Unpivot Columns Move Convert to List Split Column Format Merge Columns Extract Parse Table Any Column Text Column 1. Under the Transform tab, click on the "Group By" 2. Specify the name for the new aggregated column 3. Select the columns that you want to group your data 4. Choose the aggregation operation 5. Select the column that you want to perform the aggregation operation on <u>**New Hierarchies**</u> 1. In any of the views, right click on one of the selected columns and choose "Create Hierarchy" 2. Right click on the rest of the columns and choose "Create Hierarchy" 3. Right click on one of the selected columns and select "Add to hierarchy" and the newly created hierarchy 4. Rename and organize the hierarchy <u>**New Calculated Columns**</u> Under the Add Column tab, click on the 1. "Custom Column" to create a new calculated column using M language or 2. "Conditional Column" to create a new column based on specified if-else conditions. This is similar to CASE in SQL
Initial Data Insights	Note any initial observations or trends	

Power BI Key Summary Notes (Part 2)

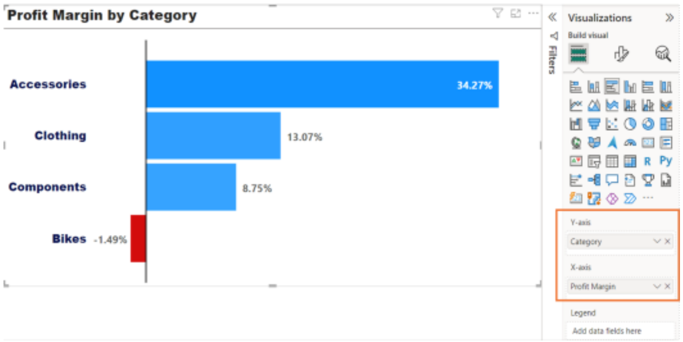
Wednesday, 25 October 2023 9:51 pm

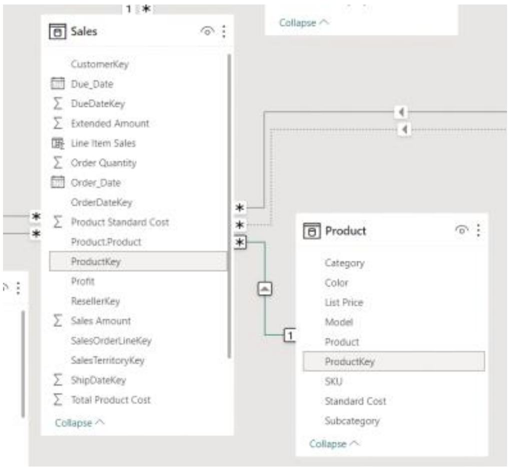
DAX

DAX stands for Data Analysis expressions, and is a **formula language used in Power BI, Power Pivot, and Analysis Services**. It is mainly used for **creating custom calculations and aggregations in data models**. There are **four types of calculations** **[Calculated Columns , Calculated Measures , Calculated Tables]** that can be created using DAX.

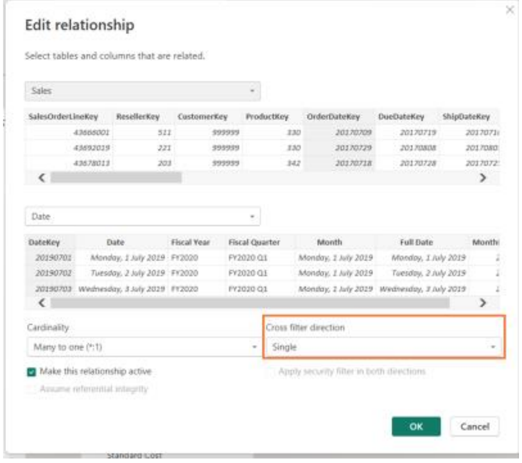
A DAX formula **always starts with an equal sign (=)**. After the equal sign, you can provide any expression that evaluates to a single scalar (single value that can take the form of integer, floats, string, date/time, boolean) or expressions that be converted to a scalar.

In DAX, when you reference a column, it is a common practice to **specify the table name <u>before</u> the column name** e.g. Sales[Order_Quantity], where Sales is the table name, and Order_Quantity is the column name. This notation **provides clarity and avoids confusion** especially when Order_Quantity can also be found in other tables such as purchase order line item. This is similar to SQL syntax, especially in situations where multiple tables are involved in a query.

Type of Calculations	Definitions	Use Case	Example
Calculated Columns	**New columns that you add to an existing table** in your data model. **These columns are computed row by row based on a DAX expression**	Useful for adding new data to your dataset	To add a new calculated column named "Line Item Sales" to the Sales table and calculate the total sales for each sales order line, taking into account the order quantity, unit price and the unit price discount (%) Line Item Sales = Sales[Order Quantity] * Sales[Unit Price] * (1-Sales[Unit Price Discount Pct])
Calculated Measures	Used to **perform calculations on aggregated data within the context of a visualization or a report** **Context sensitive** and will **adapt their calculations based on the filters, slicers and visualization** in the report Can be **reused in multiple visualizations and reports**	Ideal for creating KPIs or complex aggregations such as Year-on-Year (YoY) Growth, Quarter-by-Quarter Sales Performance"	To visualize the profit margin for each product category • Create a Calculated Measure for Profit Margin Profit Margin = (SUM(Sales[Sales Amount]) - SUM(Sales[Total Product Cost])) / SUM(Sales[Sales Amount]) • Drag the column that corresponds to "Product Category" and the "Profit Margin" measure onto a visual such as a bar chart  Profit Margin by Category Accessories 34.27% Clothing 13.07% Components 8.75% Bikes -1.49% Filters Visualizations Build visual R Py Legend Add data fields here Y-axis Category X-axis Profit Margin **Note** : As the visualization is based on the "Category" column in the Product table, it is important to ensure that the **relationship** between the sales and products table **is established and active (solid line)**.



In power BI, **multiple relationships between two tables can be created** but **ONLY ONE of these relationships can be active at a time**. The active relationship determines how filtering and calculations work in the data model.



Edit relationship

Select tables and columns that are related.

Sales

SalesOrderLineKey	ResellerKey	CustomerKey	ProductKey	OrderDateKey	DueDateKey	ShipDateKey
43666001	511	999999	330	20170709	20170719	2017071
43692019	221	999999	330	20170729	20170808	2017080
43678013	203	999999	342	20170718	20170728	2017072

Date

DateKey	Date	Fiscal Year	Fiscal Quarter	Month	Full Date	Month
20190701	Monday, 1 July 2019	FY2020	FY2020 Q1	Monday, 1 July 2019	Monday, 1 July 2019	1
20190702	Tuesday, 2 July 2019	FY2020	FY2020 Q1	Monday, 1 July 2019	Tuesday, 2 July 2019	1
20190703	Wednesday, 3 July 2019	FY2020	FY2020 Q1	Monday, 1 July 2019	Wednesday, 3 July 2019	1

Cardinality: Many to one (*:1)

Cross filter direction: Single

☑ Make this relationship active

☐ Assume referential integrity

☐ Apply security filter in both directions

OK Cancel

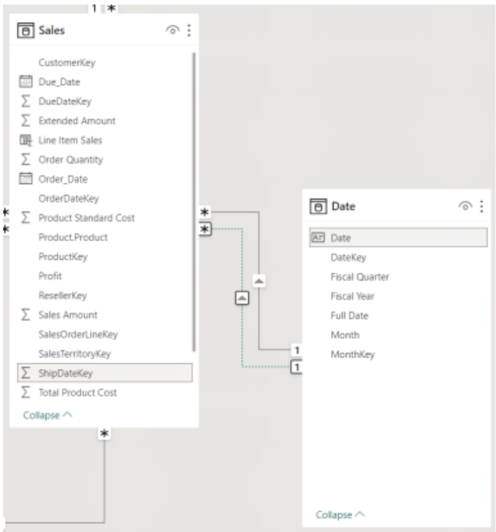
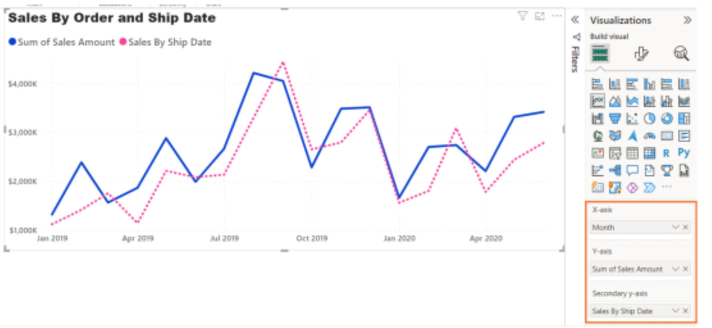
Cross filter direction determines the **flow of data filtering between tables in a relationship**. If the cross filter direction is set to "**Both**" between these tables, **filtering can occur in both directions**. This means that selecting a specific product in the Products table will filter the Sales table to show only the sales data related to that product. Similarly, selecting a specific sales record in the Sales table will filter the Products table to display only the relevant product information.

However, **using bidirectional filtering can create complex interactions** and may **result in unexpected or undesired filtering outcomes**. For example, if a user selects a specific product in the Products table, it can filter the Sales table to show relevant sales data. However, this bidirectional filtering can also impact the Products table, potentially leading to a distorted view of product information.

Additionally, enabling both cross filter directions can introduce performance issues, as the bidirectional filtering requires more computational resources to process and maintain the synchronized filters between tables.

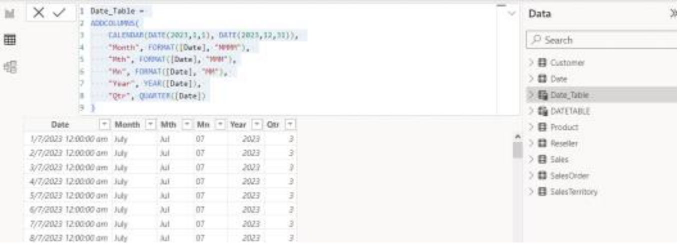
Temporarily Override the active relationship between two tables

The **USERELATIONSHIP** function allows you to **temporarily override the active relationship between two tables**. This function is **useful in situations where you have multiple relationships between tables**, and you want to **perform calculations or create measures based on a different relationship temporarily**.

			 Sales CustomerKey Due_Date DueDateKey Extended Amount Line Item Sales Order Quantity Order_Date OrderDateKey Product Standard Cost Product.Product ProductKey Profit ResellerKey Sales Amount SalesOrderLineKey SalesTerritoryKey ShipDateKey Total Product Cost Collapse Date Date DateKey Fiscal Quarter Fiscal Year Full Date Month MonthKey Collapse 1 * For example, if there are multiple relationships between "Date" and "Sales" and there is a need to calculate sales based on the "ShipDateKey" for a specific measure, you can use USERELATIONSHIP to specify the desired relationship. Date[Date]=Sales[OrderDateKey] - Active relationship Date[Date]=Sales[ShipDateKey] - **Inactive** relationship Sales By Ship Date = CALCULATE(SUM(Sales[Sales Amount]), USERELATIONSHIP(Sales[ShipDateKey], 'Date'[DateKey]))  Sales By Order and Ship Date ● Sum of Sales Amount ● Sales By Ship Date $4,000K $3,000K $2,000K $1,000K Jan 2019 Apr 2019 Jul 2019 Oct 2019 Jan 2020 Apr 2020 Filters Visualizations Build visual R Py X-axis Month Y-axis Sum of Sales Amount Secondary y-axis Sales By Ship Date
Calculated Tables	**Generated within power BI model.** Calculated tables are **useful for creating custom tables to support specific analysis aggregation or calculation** **Static Calculated Tables** • Created with fixed structures and data, and without reference to other tables in the model • Once created, the values in these tables do not change unless the formula or the underlying data is modified. **Dynamic Calculated Tables**	**Static Calculated Tables** • Date Table • Lookup Table • Auxiliary Table **Dynamic Calculated Tables** • Advanced Filtering Logic based on user selection • Generate table with different structures and data based on slicer choices • Support complex calculations that require context awareness	1. DATE TABLE is required if you want to • Leverage the time intelligence functions i.e. perform calculations and analyses based on dates • Slice, dice and filter data based on time periods. This is done by establishing relationships between the date table and other data tables in the model • Drill down and explore the data at different levels of granularity e.g. quarter, month, year • Sort data in logical chronological order and filter data based on specific time periods How to create a DATE TABLE? • By range of date : **DATETABLE = CALENDAR(DATE(2016,1,1), DATE(2022,1,1))** • Based on the date range in the dataset : **DATETABLE = CALENDARAUTO()** 2. Create a Date Table with additional date part columns Date_Table = ADDCOLUMNS(CALENDAR(DATE(2023,1,1), DATE(2023,12,31)), "Month", FORMAT([Date], "MMMM"), "Mth", FORMAT([Date], "MMM"), "Mn", FORMAT([Date], "MM"), "Year", YEAR([Date]),

- Define using DAX expressions
- Data is generated on the fly
- Useful for advanced calculation and data modelling

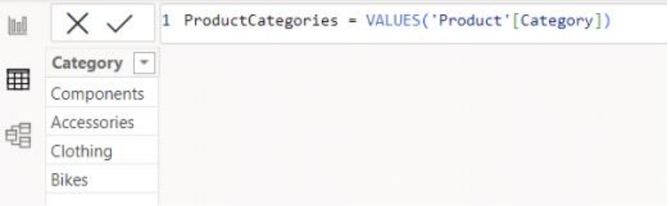
```
    "Qtr", QUARTER([Date])
)
```



```
Date_Table =
ADDCOLUMNS(
    CALENDAR(DATE(2023,1,1), DATE(2023,12,31)),
    "Month", FORMAT([Date], "MMMM"),
    "Mth", FORMAT([Date], "MMM"),
    "Mn", FORMAT([Date], "MM"),
    "Year", YEAR([Date]),
    "Qtr", QUARTER([Date])
)
```

Date	Month	Mth	Mn	Year	Qtr
1/7/2023 12:00:00 am	July	Jul	07	2023	3
2/7/2023 12:00:00 am	July	Jul	07	2023	3
3/7/2023 12:00:00 am	July	Jul	07	2023	3
4/7/2023 12:00:00 am	July	Jul	07	2023	3
5/7/2023 12:00:00 am	July	Jul	07	2023	3
6/7/2023 12:00:00 am	July	Jul	07	2023	3
7/7/2023 12:00:00 am	July	Jul	07	2023	3
8/7/2023 12:00:00 am	July	Jul	07	2023	3

3. Define a table capturing the unique list of product categories based on column values from a table



```
ProductCategories = VALUES('Product'[Category])
```

Category
Components
Accessories
Clothing
Bikes

```
ProductCategories = VALUES('Product'[Category])
```

4. Create a static table similar to the SQL CREATE TABLE syntax

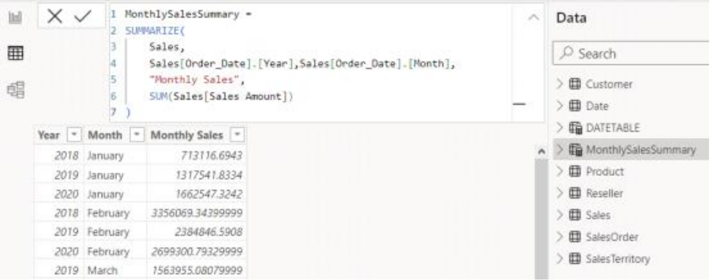
```
CategoryTable =
DATATABLE(
    "CategoryID", INTEGER,
    "CategoryName", STRING,
    {
        {1, "Electronics"},
        {2, "Apparel"},
        {3, "Groceries"}
    }
)
```

5. Create a dynamic table from the Sales table that lists only the sales data for the current year

```
SalesCurrentYear =
FILTER(
    Sales,
    YEAR(Sales[Order_Date]) = YEAR(TODAY())
)
```

6. Create a dynamic table from the Sales table that provides a summary of the monthly sales.

```
MonthlySalesSummary =
SUMMARIZE(
    Sales,
    Sales[Order_Date].[Year],Sales[Order_Date].[Month],
    "Monthly Sales",
    SUM(Sales[Sales Amount])
)
```



```
MonthlySalesSummary =
SUMMARIZE(
    Sales,
    Sales[Order_Date].[Year],Sales[Order_Date].[Month],
    "Monthly Sales",
    SUM(Sales[Sales Amount])
)
```

Year	Month	Monthly Sales
2018	January	713116.6943
2019	January	1317541.8334
2020	January	1662547.3242
2018	February	3356069.34399999
2019	February	2384846.5908
2020	February	2699300.79329999
2019	March	1563955.08079999

NOTE :

- All tables must have unique name
- Column names must also be unique within each table
- Both table and column names are CASE-INSENSITIVE e.g. SALES, Sales, sales all refer to the same table
- If the name of a table contains spaces, reserved keywords, or disallowed characters, you must enclose the table name in single quotation marks.

DAX Building Blocks

<table>
<tr><th>Building Block</th><th>Description</th><th>Example</th></tr>
<tr><td>Constants</td><td>Literal values</td><td>123, "Hello"</td></tr>
<tr><td>Operators</td><td>Arithmetic, Comparison, Concatenation</td><td>
<table>
<tr><td>+</td><td>Addition</td></tr>
<tr><td>-</td><td>Subtraction</td></tr>
<tr><td>*</td><td>Multiplication</td></tr>
<tr><td>/</td><td>Division</td></tr>
<tr><td>^</td><td>Exponentiation</td></tr>
<tr><td>%</td><td>Modulus (Remainder Division)</td></tr>
<tr><td>=</td><td>Equal To</td></tr>
<tr><td><></td><td>Not Equal To</td></tr>
<tr><td>></td><td>Greater Than</td></tr>
<tr><td><</td><td>Less Than</td></tr>
<tr><td>>=</td><td>Greater Than or Equal To</td></tr>
<tr><td><=</td><td>Less Than or Equal To</td></tr>
<tr><td>&</td><td>Concatenation</td></tr>
<tr><td>&&</td><td>Logical AND</td></tr>
<tr><td>||</td><td>Logical OR</td></tr>
<tr><td>!</td><td>Logical NOT</td></tr>
</table>
</td></tr>
<tr><td>Functions</td><td>Pre-built operations</td><td>
<table>
<tr><th>Category</th><th>Function</th><th>Description</th></tr>
<tr><td>AGGREGATION</td><td>SUM</td><td>Sums up values in a column</td></tr>
<tr><td></td><td>AVERAGE</td><td>Calculates the average</td></tr>
<tr><td></td><td>COUNT</td><td>Counts the number of rows</td></tr>
<tr><td></td><td>MIN</td><td>Returns the minimum value</td></tr>
<tr><td></td><td>MAX</td><td>Returns the maximum value</td></tr>
<tr><td>DATE TIME</td><td>NOW</td><td>Returns the current date and time</td></tr>
<tr><td></td><td>TODAY</td><td>Returns the current date</td></tr>
<tr><td></td><td>YEAR</td><td>Returns the year of a date</td></tr>
<tr><td></td><td>MONTH</td><td>Returns the month of a date</td></tr>
<tr><td></td><td>DAY</td><td>Returns the day of a date</td></tr>
<tr><td>TEXT</td><td>UPPER</td><td>Converts text to uppercase</td></tr>
<tr><td></td><td>LOWER</td><td>Converts text to lowercase</td></tr>
<tr><td></td><td>LEN</td><td>Returns the length of a text string</td></tr>
<tr><td></td><td>CONCATENATE</td><td>Concatenates two or more text strings</td></tr>
<tr><td></td><td>REPLACE</td><td>Replaces characters within text</td></tr>
<tr><td></td><td>LEFT</td><td>Returns a specified number of characters from the beginning of a text string</td></tr>
<tr><td></td><td>RIGHT</td><td>Returns a specified number of characters from the end of a text string</td></tr>
<tr><td>LOGICAL</td><td>IF</td><td>Checks a condition and returns a value</td></tr>
<tr><td></td><td>AND</td><td>Returns TRUE if all conditions are true</td></tr>
<tr><td></td><td>OR</td><td>Reverses a logical value</td></tr>
<tr><td>LOOKUP</td><td>RELATED</td><td>Returns a related value from another table</td></tr>
<tr><td></td><td>LOOKUPVALUE</td><td>Searches for a value in a table</td></tr>
<tr><td></td><td>ALL</td><td>Removes filters from a table or column</td></tr>
<tr><td></td><td>FILTER</td><td>Returns a table after applying a filter expression</td></tr>
<tr><td>STATISTICAL</td><td>RANKX</td><td>Ranks a number in a list of numbers</td></tr>
<tr><td>INFORMATION</td><td>ISBLANK</td><td>Checks for a blank or null value</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td><b>ISTEXT</b></td><td>Checks if a value is text</td></tr>
<tr><td></td><td><b>ISNUMBER</b></td><td>Checks if a value is a number</td></tr>
<tr><td>TIME INTELLIGENCE</td><td><b>TOTALYTD</b></td><td>Calculate the total from the beginning of the year to a specified date, within the current filter context</td></tr>
<tr><td></td><td><b>DATESYTD</b></td><td>Returns a table of dates starting from the beginning of the year to a specified end date in the current filter context</td></tr>
<tr><td></td><td><b>SAMEPERIODLASTYEAR</b></td><td>Returns a set of dates that is the same size and shape as the specified dates, but it starts one year earlier</td></tr>
<tr><td></td><td><b>PARALLELPERIOD</b></td><td>Returns a table of dates that represents a period parallel to the dates in the specified date column, in the current context, at the specified level of granularity</td></tr>
<tr><td></td><td><b>ENDOFMONTH</b></td><td>Returns the last date of the month, for the specified dates.</td></tr>
</table>
Refer to DAX GUIDE or Microsoft DAX Reference for the list of complete functions
</td></tr>
<tr><td><b>References</b></td><td>Pointers to columns or tables</td><td>
<table>
<tr><th>Type of References</th><th>Example</th><th>Description</th></tr>
<tr><td><b>Basic Column Reference</b></td><td>[Revenue]</td><td>Refers to the Revenue column in the current context</td></tr>
<tr><td><b>Table and Column</b></td><td>Sales.Revenue</td><td>Explicitly refers to the Revenue column in the Sales table</td></tr>
<tr><td><b>Related Table (Based on Relationship)</b></td><td>RELATED(Customer[Name])</td><td>Refers to the Name column in a related Customers table</td></tr>
</table>
</td></tr>
<tr><td><b>Variables</b></td><td>Named containers for values or expressions</td><td>VAR x = 1</td></tr>
</table>

DAX Syntax Structure

Type	Description	Example
Function(Expressions)	Typical and most common way to use most DAX functions for calculations	SUM(Table[Column])
Direct References	Directly reference a column without a wrapping function	Table[Column]
Operators	Combine or compare expressions using operators	[Measure1] + [Measure2] Table[Column1] > Table[Column2]
Row and Filter Contexts	Formulas that rely on the row context, especially in calculated columns	[Column1] * [Column2]
Complex Expressions with Multiple Functions	Nest multiple functions and expressions for advanced calculations	CALCULATE(SUM(Table[Column]), FILTER(ALL(Table), Table[OtherColumn] = "Value"))
Constants and Values	Use constants, strings, numbers, or boolean values directly	"Hello, World!" TRUE 123

Explore Common DAX Formulae/ Syntax

Sales

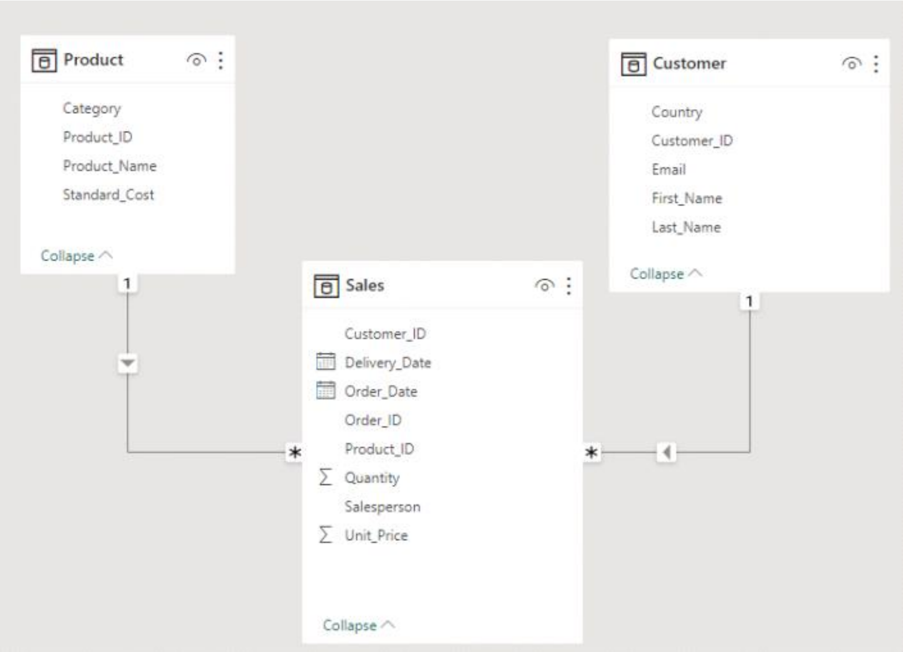
Order_ID	Order_Date	Delivery_Date	Product_ID	Customer_ID	Quantity	Unit_Price	Salesperson

Product

Product_ID	Product_Name	Standard_Cost	Category

Customer

Customer_ID	First_Name	Last_Name	Country	Email



AGGREGATIONS

There are several similarities in the basic aggregation functions between DAX and SQL.

SQL - Uses a SELECT statement to **specify the aggregation** and **specifies the table from which the data is coming from**
DAX - The **table and column are specified within the aggregation function** itself

Function	Description	DAX Formula	SQL Syntax

SUM()	Sums the values of **ONE** column	Total Quantity Sold = SUM(Sales[Quantity])	SELECT SUM(Quantity) AS 'Total Quantity Sold'
			FROM Sales;

SUMX()	Sums the results of an expression evaluated for each row in a table e.g. To calculate the Total Sales Amount, there is a need to multiply Unit_Price with Quantity for each row first, followed by a summation of these row results to produce a single scalar	Total Sales Amount =
		SUMX(
		Sales, Sales[Quantity]*Sales[Unit_Price]
		)
		
		If there is already a column e.g. Sales_Amt in the Sales table that captures the sales amount for each line item (sub-total), you can use the SUM function to compute the total sales amount i.e.
		
		SUM(Sales[Sales_Amt])
			SELECT SUM(Quantity*Unit_Price) AS 'Total Sales Amount'
			FROM Sales;

AVERAGE()	Returns the average of the values in **ONE** column	Average Unit Price =
		AVERAGE(Sales[Unit_Price])
			SELECT AVG(Unit_Price) AS 'Average Unit Price'
			FROM Sales;

AVERAGEX()	Averages the results of an expression evaluated for each row in a table e.g. To calculate the Average Sales Amount, there is a need to multiply Unit_Price with Quantity for each row first, followed by an average of these row results to produce a single scalar	Average Sales Amount =
		AVERAGEX(
		Sales, Sales[Quantity]*Sales[Unit_Price]
		)
			SELECT AVG(Quantity*Unit_Price) AS 'Average Sales Amount'
			FROM Sales;

MIN()	Returns the smallest value in a column	Earliest Order Date = MIN(Sales[Order_Date])	SELECT MIN(Order_Date) AS 'Earliest Order Date'
			FROM Sales;

MAX()	Returns the largest value in a column	Latest Delivery Date =
		MAX(Sales[Delivery_Date])
			SELECT MAX(Delivery_Date) AS 'Latest Delivery Date'
			FROM Sales;

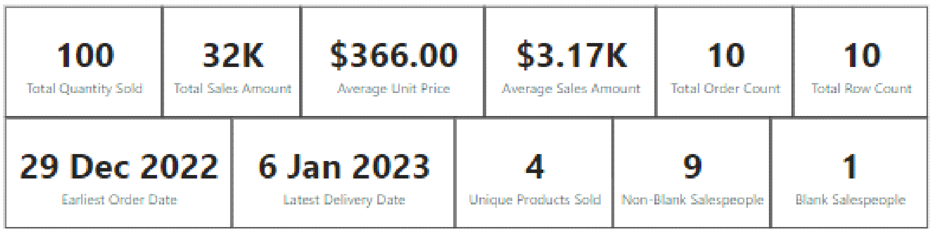
COUNT()	Counts the number of rows in a table or column	Total Order Count = COUNT(Sales[Order_ID])	SELECT COUNT(Order_ID) AS 'Total Order Count'
			FROM Sales;

COUNTROWS()	To count the number of rows in a table, you can also use COUNTROWS()	Total Row Count = COUNTROWS(Sales)	SELECT COUNT(*) AS 'Total Row Count'
			FROM Sales;

COUNTA()	Counts the number of non-blank rows in a column	Non-Blank Salespeople =
		COUNTA(Sales[Salesperson])
			SELECT COUNT(Salesperson) AS 'Non-Blank Salespeople'
			FROM Sales
			WHERE Salesperson IS NOT NULL;

COUNTBLANK()	Counts the number of blank rows in a column	Blank Salespeople =
		COUNTBLANK(Sales[Salesperson])
			SELECT COUNT(Salesperson) AS 'Non-Blank Salespeople'
			FROM Sales
			WHERE Salesperson IS NULL;

DISTINCTCOUNT()	Counts the number of distinct values in a column	Unique Products Sold =
		DISTINCTCOUNT(Sales[Product_ID])
			SELECT COUNT(DISTINCT Product_ID) AS 'Unique Products Sold'
			FROM Sales;



100	32K	$366.00	$3.17K	10	10
Total Quantity Sold	Total Sales Amount	Average Unit Price	Average Sales Amount	Total Order Count	Total Row Count

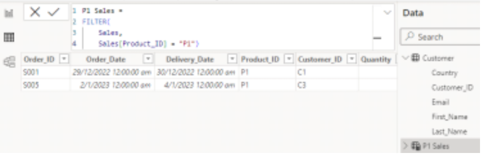
29 Dec 2022	6 Jan 2023	4	9	1
Earliest Order Date	Latest Delivery Date	Unique Products Sold	Non-Blank Salespeople	Blank Salespeople

LOOKUP

The lookup function helps in retrieving data from a different table or column based on some key or condition

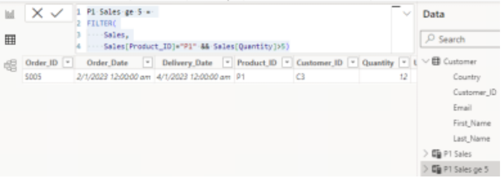
<table>
<tr><th>Function</th><th>Description</th><th>DAX Formula</th><th>SQL Syntax</th></tr>
<tr><td>FILTER()</td><td>Returns a table that has been <b>filtered based on a certain condition</b> (equivalent to <b>WHERE clause in SQL</b>)</td><td>To capture the sales data from the Sales table that are related to Product P1 only

P1 Sales =
FILTER(
   Sales,
   Sales[Product_ID] = "P1")


P1 Sales =
FILTER(
   Sales,
   Sales[Product_ID] = "P1")
<table>
<tr><th>Order_ID</th><th>Order_Date</th><th>Delivery_Date</th><th>Product_ID</th><th>Customer_ID</th><th>Quantity</th></tr>
<tr><td>S001</td><td>29/12/2022 12:00:00 am</td><td>30/12/2022 12:00:00 am</td><td>P1</td><td>C1</td><td></td></tr>
<tr><td>S005</td><td>2/1/2023 12:00:00 am</td><td>4/1/2023 12:00:00 am</td><td>P1</td><td>C3</td><td></td></tr>
</table>

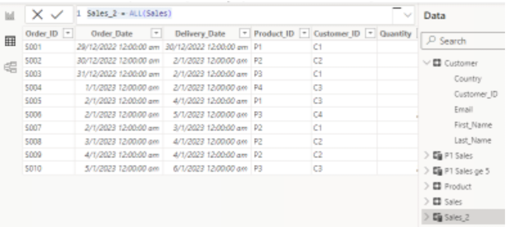
To capture the sales data from the Sales table that are related to Product P1 only and with quantity sold > 5. Use the Logical AND operator && or Logical OR || for more than one filtering condition

P1 Sales ge 5 =
FILTER(
   Sales,
   Sales[Product_ID]="P1" && Sales[Quantity]>5)

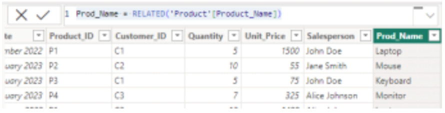

P1 Sales ge 5 =
FILTER(
   Sales,
   Sales[Product_ID]="P1" && Sales[Quantity]>5)
<table>
<tr><th>Order_ID</th><th>Order_Date</th><th>Delivery_Date</th><th>Product_ID</th><th>Customer_ID</th><th>Quantity</th></tr>
<tr><td>S005</td><td>2/1/2023 12:00:00 am</td><td>4/1/2023 12:00:00 am</td><td>P1</td><td>C3</td><td>12</td></tr>
</table>

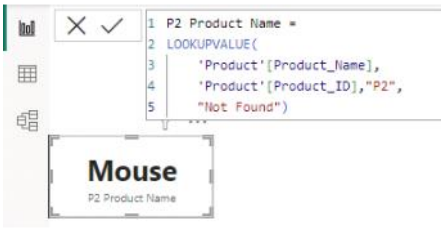
</td><td>SELECT *
FROM Sales
WHERE Product_ID = 'P1';

SELECT *
FROM Sales
WHERE Product_ID = 'P1'
AND Quantity &gt; 5;</td></tr>
<tr><td>ALL()</td><td>Removes any filters that might have been applied to the specified table or table column, essentially returning the entire table</td><td>Sales_2 = ALL(Sales)


Sales_2 = ALL(Sales)
<table>
<tr><th>Order_ID</th><th>Order_Date</th><th>Delivery_Date</th><th>Product_ID</th><th>Customer_ID</th><th>Quantity</th></tr>
<tr><td>S001</td><td>29/12/2022 12:00:00 am</td><td>30/12/2022 12:00:00 am</td><td>P1</td><td>C1</td><td></td></tr>
<tr><td>S002</td><td>30/12/2022 12:00:00 am</td><td>2/1/2023 12:00:00 am</td><td>P2</td><td>C2</td><td></td></tr>
<tr><td>S003</td><td>31/12/2022 12:00:00 am</td><td>2/1/2023 12:00:00 am</td><td>P3</td><td>C1</td><td></td></tr>
<tr><td>S004</td><td>1/1/2023 12:00:00 am</td><td>2/1/2023 12:00:00 am</td><td>P4</td><td>C3</td><td></td></tr>
<tr><td>S005</td><td>2/1/2023 12:00:00 am</td><td>4/1/2023 12:00:00 am</td><td>P1</td><td>C3</td><td></td></tr>
<tr><td>S006</td><td>2/1/2023 12:00:00 am</td><td>5/1/2023 12:00:00 am</td><td>P3</td><td>C4</td><td></td></tr>
<tr><td>S007</td><td>2/1/2023 12:00:00 am</td><td>3/1/2023 12:00:00 am</td><td>P2</td><td>C1</td><td></td></tr>
<tr><td>S008</td><td>3/1/2023 12:00:00 am</td><td>4/1/2023 12:00:00 am</td><td>P2</td><td>C2</td><td></td></tr>
<tr><td>S009</td><td>4/1/2023 12:00:00 am</td><td>4/1/2023 12:00:00 am</td><td>P2</td><td>C2</td><td></td></tr>
<tr><td>S010</td><td>5/1/2023 12:00:00 am</td><td>6/1/2023 12:00:00 am</td><td>P3</td><td>C3</td><td></td></tr>
</table>

</td><td>SELECT *
FROM Sales;</td></tr>
<tr><td>RELATED()</td><td>Returns a related value</td><td>Prod_Name = RELATED('Product'[Product_Name])</td><td>SELECT</td></tr>
</table>

	from another table. It **requires an active relationship between the tables**	
		Prod_Name = RELATED('Product'[Product_Name])
		te Product_ID Customer_ID Quantity Unit_Price Salesperson Prod_Name
		mber 2022 P1 C1 5 1500 John Doe Laptop
		uary 2023 P2 C2 10 55 Jane Smith Mouse
		uary 2023 P3 C1 5 75 John Doe Keyboard
		uary 2023 P4 C3 7 325 Alice Johnson Monitor
			Product.Product_Name AS Prod_Name
			FROM Sales
			LEFT JOIN Product
			ON Sales.Product_ID = Product.Product_ID;

LOOKUPVALUE()	Useful for looking up a single value in a table **without needing an active relationship**	P2 Product Name =
		LOOKUPVALUE(
		'Product'[Product_Name],
		'Product'[Product_ID],"P2"
		)
		
		P2 Product Name =
		LOOKUPVALUE(
		'Product'[Product_Name],
		'Product'[Product_ID],"P2",
		"Not Found")
		Mouse
		P2 Product Name
		
		LOOKUPVALUE(
		output result,
		search column name, search column1.value,
		search column2 name, search column2.value,)
		……,
		output result is no match is found)
			SELECT Product_Name
			FROM Product
			WHERE Product_ID = "P2"

CALCULATE

Purpose : CALCULATE is one of the most powerful functions in DAX. It used for **modifying the filter context within a DAX formula** and can be used to change filters, apply new filters, or even remove filters.

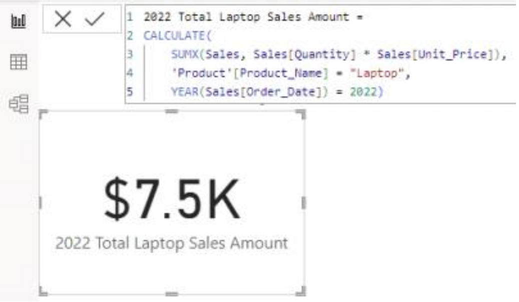
CALCULATE(<expression>, <filter1>, <filter2>, ...)

Usage : It is used when you want to alter the filter context within a specific calculation. For example, you might want to calculate a measure ignoring certain filters or applying additional ones.

Example : Calculate the Total Laptop Sales Amount in 2022
- Aggregation : Sales[Quantity] * Sales[Unit_Price]
- Aggregation Function : SUMX (Since we need to first evaluate the Sales Amount for EACH row first)
- Filtering Condition 1 : Year(Sales[Order_Date]) = 2022
- Filtering Condition 2 : Product[Product_Name] = "Laptop"

In this example, the primary DAX formula is to compute the Total Sales Amount and this is achieved with the SUMX function applied to Sales, multiplying Sales[Quantity] by Sales[Unit_Price]. We then use the CALCULATE function, which provides the capability to implement filters to the Total Sales Amount.

```
2022 Total Laptop Sales Amount =
CALCULATE(
   SUMX(Sales, Sales[Quantity] * Sales[Unit_Price]),
   'Product'[Product_Name] = "Laptop",
   YEAR(Sales[Order_Date]) = 2022)
```



```
2022 Total Laptop Sales Amount =
CALCULATE(
    SUMX(Sales, Sales[Quantity] * Sales[Unit_Price]),
    'Product'[Product_Name] = "Laptop",
    YEAR(Sales[Order_Date]) = 2022)
```

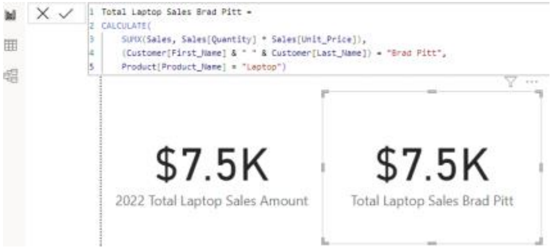
$7.5K

2022 Total Laptop Sales Amount

With CALCULATE, AGGREGATION, DATE, TEXT, LOOKUP functions and OPERATORS, we can create/ generate a wide variety of metrics and visuals, from basic sums and averages to complex, context-aware calculations that adjust based on user interactions in a Power BI report or dashboard.

Example : Calculate the Total Laptop Sales Amount for Customer 'Brad Pitt'
- Aggregation : Sales[Quantity] * Sales[Unit_Price]
- Aggregation Function : SUMX (Since we need to first evaluate the Sales Amount for EACH row first)
- Filtering Condition 1 : Customer[First_Name] & " " & Customer[Last_name] = "Brad Pitt" (& = Concatenation operator)
- Filtering Condition 2 : Product[Product_Name] = "Laptop"

```
Total Laptop Sales Brad Pitt =
CALCULATE(
   SUMX(Sales, Sales[Quantity] * Sales[Unit_Price]),
   (Customer[First_Name] & " " & Customer[Last_Name]) = "Brad Pitt",
   Product[Product_Name] = "Laptop")
```



```
Total Laptop Sales Brad Pitt =
CALCULATE(
    SUMX(Sales, Sales[Quantity] * Sales[Unit_Price]),
    (Customer[First_Name] & " " & Customer[Last_Name]) = "Brad Pitt",
    Product[Product_Name] = "Laptop")
```

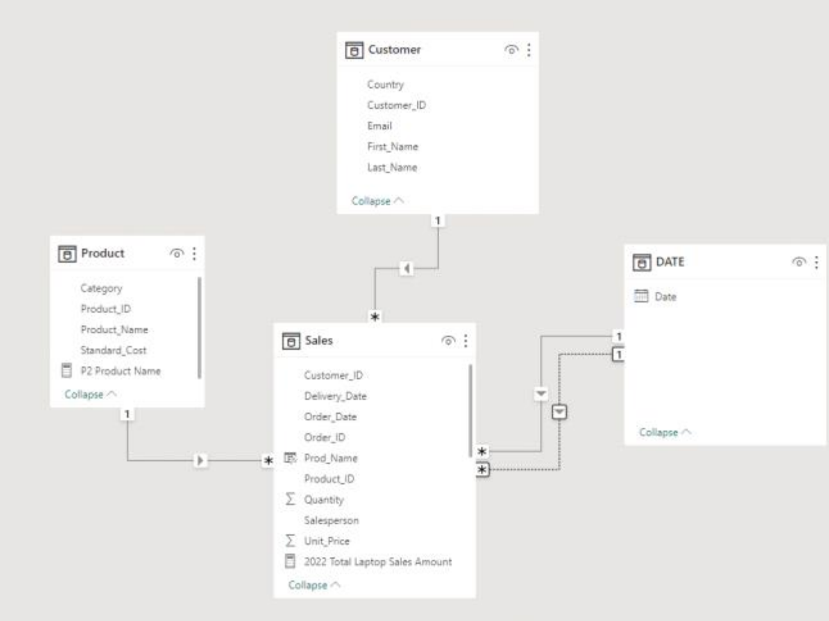
USERRELATIONSHIP

Purpose : The USERELATIONSHIP function is used to specify a relationship to be used for a particular calculation or measure, overriding the automatic relationship detected by the DAX engine.

Usage: You use this function when you want to perform a calculation that relies on a specific relationship in the data model. By default, DAX automatically selects the "active" relationship based on context. USERELATIONSHIP allows you to choose a different relationship for a particular calculation.

Example : Calculate and Visual the 2023 Total Sales Amount by both Order and Delivery Date

It is recommended to have a dedicated Date Table, which provides benefits such as handling of gaps in dates and analyzing time intelligence functions more efficiently.



There are two date relationships between the Sales and a hypothetical DATE table. Only one relationship can be active at a time.

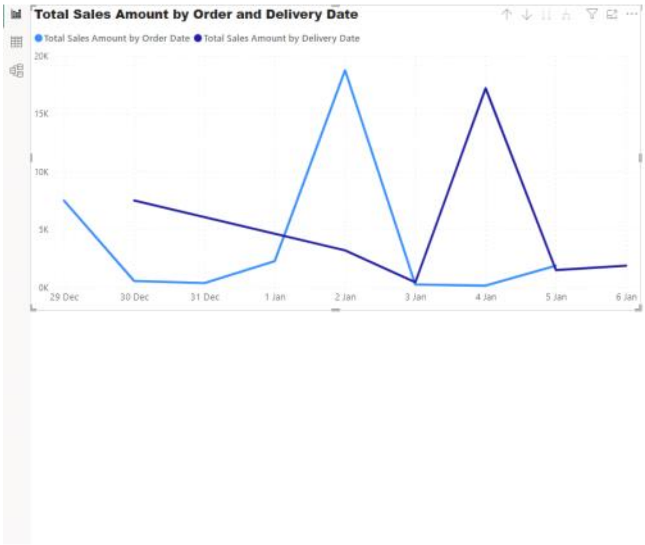
- Order_Date to DATE[Date] - Active
- Delivery_Date to DATE[Date] - Inactive

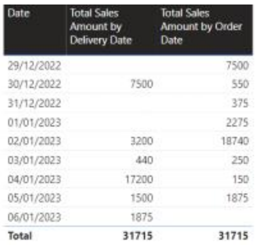
Calculate the Total Sales Amount by Order Date

```
Total Sales Amount by Order Date =
SUMX(
   Sales, Sales[Quantity]*Sales[Unit_Price]
)
```

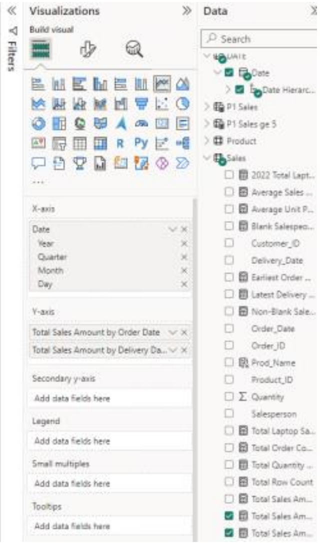
Calculate the Total Sales Amount by Delivery Date

```
Total Sales Amount by Delivery Date =
CALCULATE(
   SUMX(Sales, Sales[Quantity]*Sales[Unit_Price]),
   USERELATIONSHIP(Sales[Delivery_Date],'DATE'[Date])
)
```





Date	Total Sales Amount by Delivery Date	Total Sales Amount by Order Date
29/12/2022		7500
30/12/2022	7500	550
31/12/2022		375
01/01/2023		2275
02/01/2023	3200	18740
03/01/2023	440	250
04/01/2023	17200	150
05/01/2023	1500	1875
06/01/2023	1875	
Total	**31715**	**31715**



X-axis

Date
Year
Quarter
Month
Day

Y-axis

Total Sales Amount by Order Date
Total Sales Amount by Delivery Da...

Secondary y-axis

Add data fields here

Legend

Add data fields here

Small multiples

Add data fields here

Tooltips

Add data fields here

DATE
☑ Date
☑ Date Hierarc...
P1 Sales
P1 Sales ge 5
Product
Sales
□ 2022 Total Lapt...
□ Average Sales ...
□ Average Unit P...
□ Blank Salespeo...
□ Customer_ID
□ Delivery_Date
□ Earliest Order ...
□ Latest Delivery ...
□ Non-Blank Sale...
□ Order_Date
□ Order_ID
□ Prod_Name
□ Product_ID
□ Σ Quantity
□ Salesperson
□ Total Laptop Sa...
□ Total Order Co...
□ Total Quantity ...
□ Total Row Count
□ Total Sales Am...
☑ Total Sales Am...
☑ Total Sales Am...

Power BI
Summar...