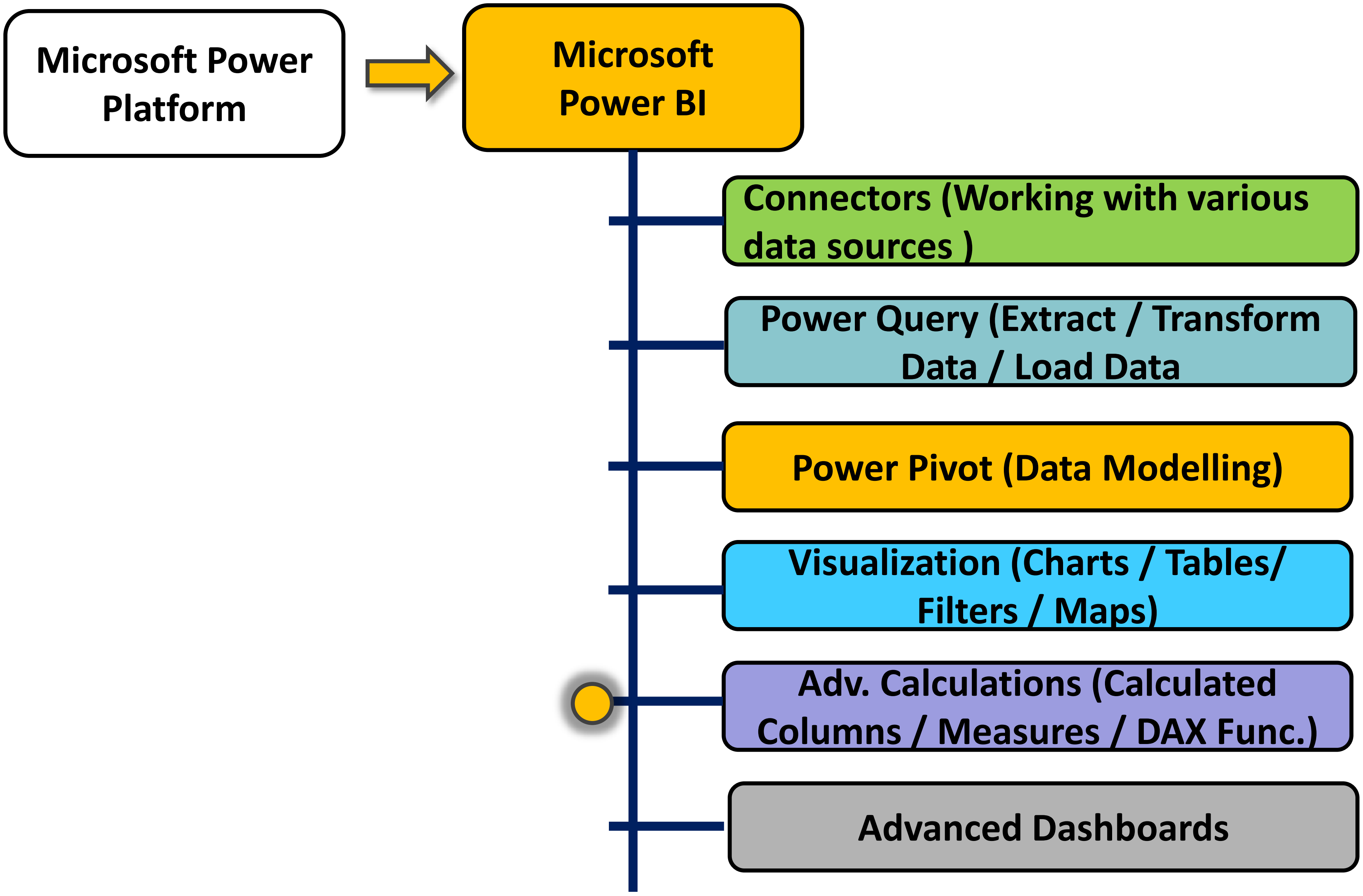
Power BI

Microsoft Power BI DAX Essentials

Amit Agarwal
Business Intelligence & Analytics Consultant
amitprabhash73@gmail.com

Course Modules


Microsoft Power Platform
Microsoft Power BI
Connectors (Working with various data sources)
Power Query (Extract / Transform Data / Load Data
Power Pivot (Data Modelling)
Visualization (Charts / Tables/ Filters / Maps)
Adv. Calculations (Calculated Columns / Measures / DAX Func.)
Advanced Dashboards

Agenda Day -4

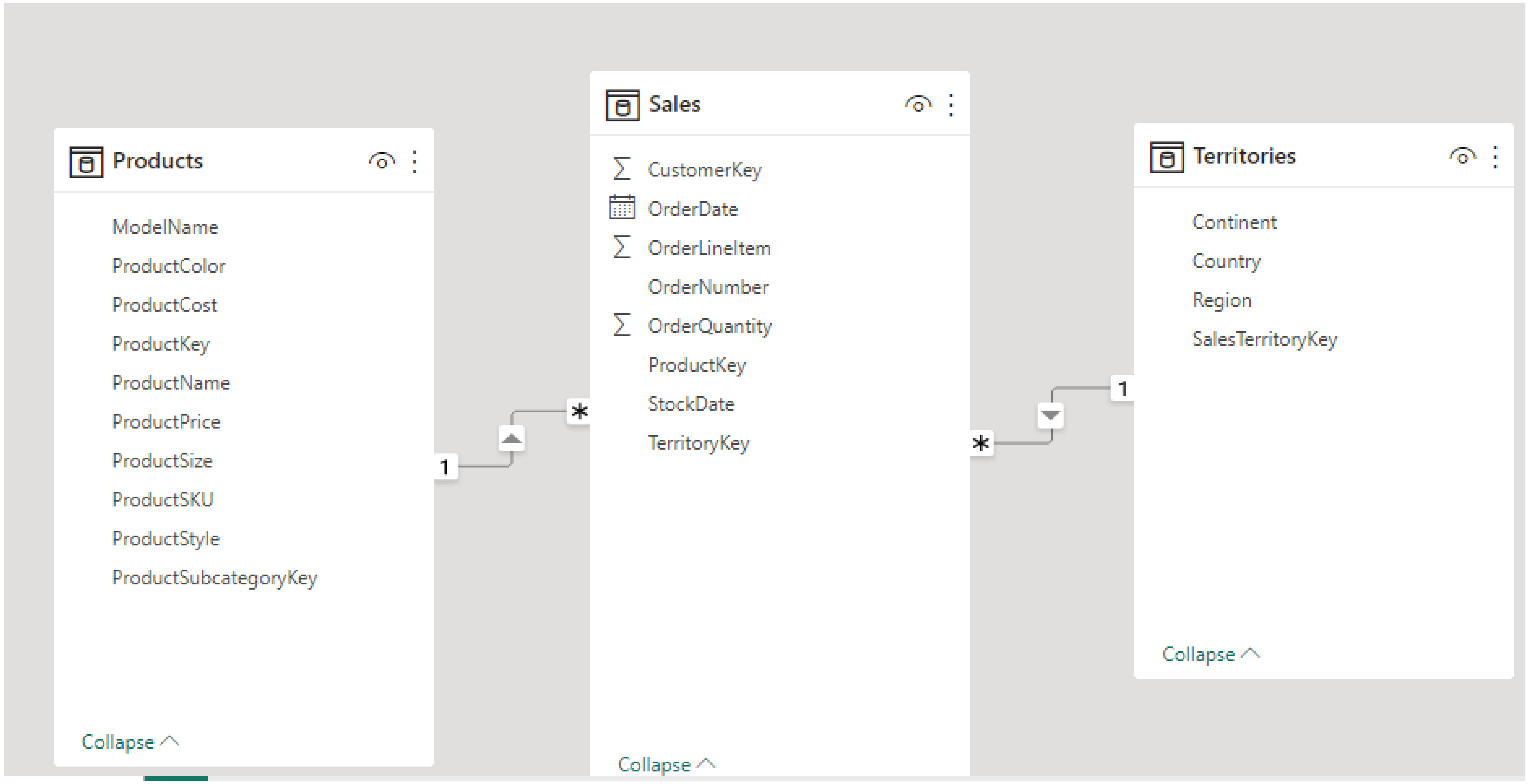
- **Power BI DAX Basics: What is DAX?**
- **Power BI DAX Basics: How does it work?**
  - Syntax
  - Context
  - Functions
- **Power BI DAX Basics: Calculated Columns & Measures**
  - **Calculated Columns**
  - **Measures**
  - **Calculated Columns vs Measures**

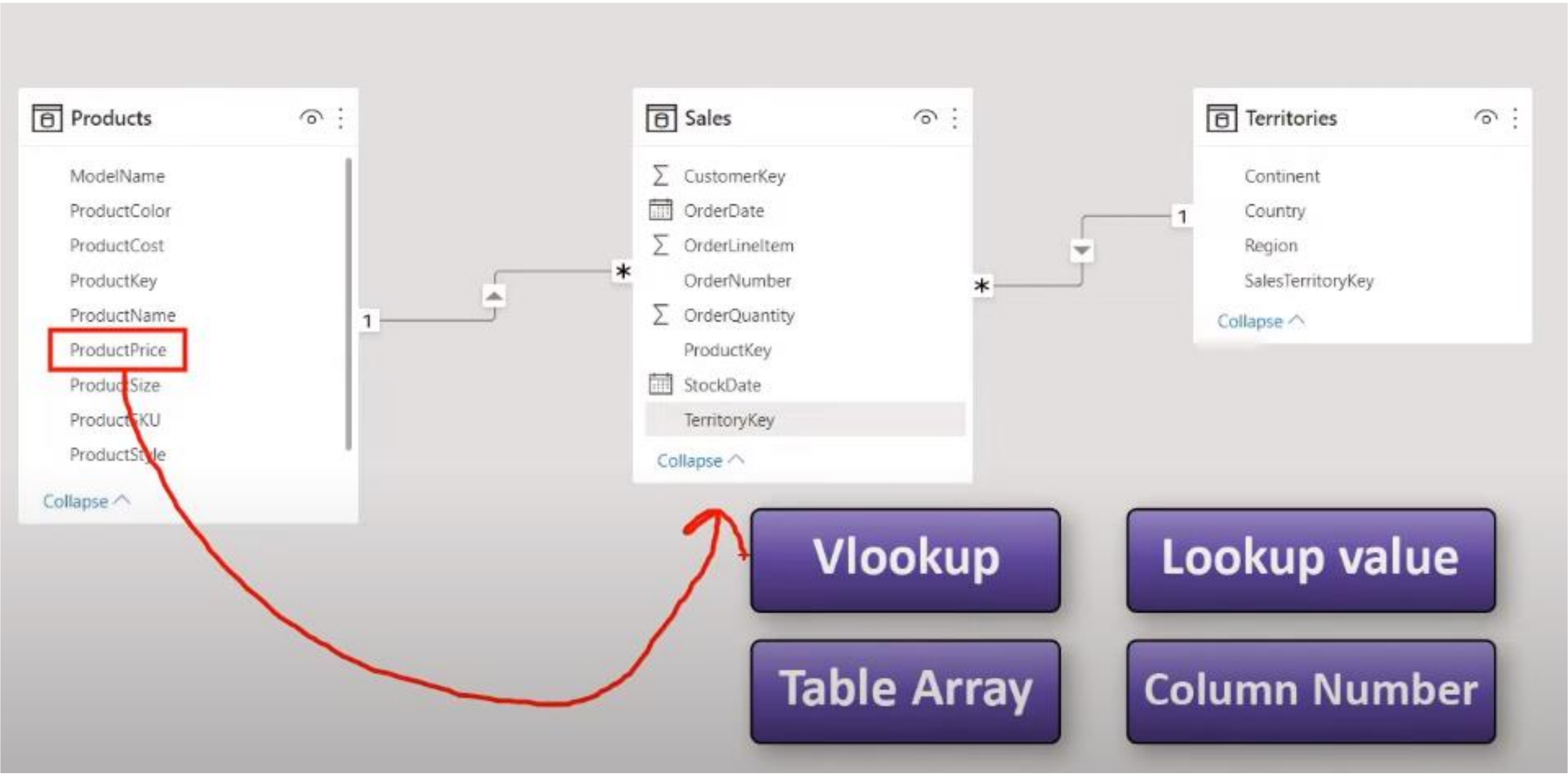
Day - 4

- **Power BI DAX Basics: Types of Functions in DAX**
  - Aggregate Functions
  - Count Functions
  - Date-Time Functions
  - Mathematical Functions
  - Logical Functions
  - Information Functions
  - Text Functions

- **Power BI DAX Basics: Creating your First Measure**

Data Modelling

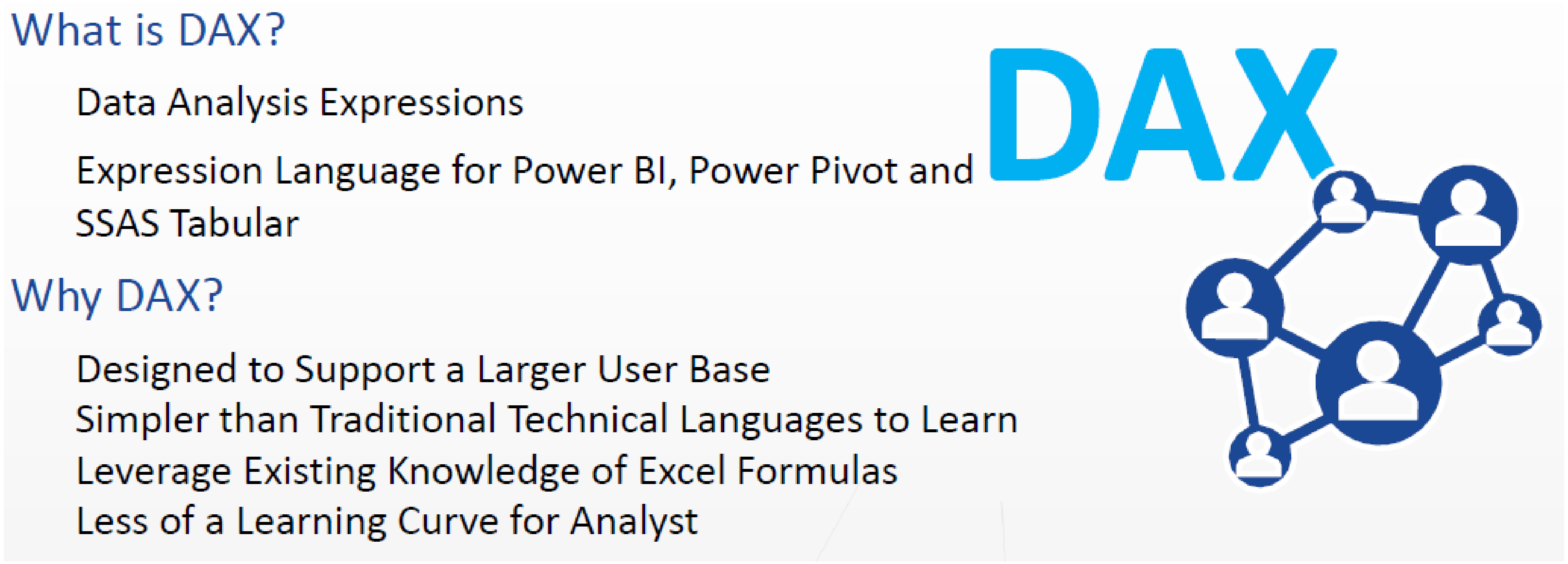



Products
ModelName
ProductColor
ProductCost
ProductKey
ProductName
ProductPrice
ProductSize
ProductSKU
ProductStyle
Collapse
1
*
Sales
CustomerKey
OrderDate
OrderLineItem
OrderNumber
OrderQuantity
ProductKey
StockDate
TerritoryKey
Collapse
*
1
Territories
Continent
Country
Region
SalesTerritoryKey
Collapse
Vlookup
Lookup value
Table Array
Column Number

Unit 1

Power BI DAX Basics
What is DAX?

DAX (Data Analysis Expression)



What is DAX?

Data Analysis Expressions

Expression Language for Power BI, Power Pivot and SSAS Tabular

Why DAX?

Designed to Support a Larger User Base
Simpler than Traditional Technical Languages to Learn
Leverage Existing Knowledge of Excel Formulas
Less of a Learning Curve for Analyst

What is DAX Used For ?

Calculated Columns

Create New Columns on a table

Method for Connecting Disparate Data Sources with Multiple Key Columns

Calculated Measures

Create Dynamic Calculations for Reporting

Time Intelligence

Calculated Tables

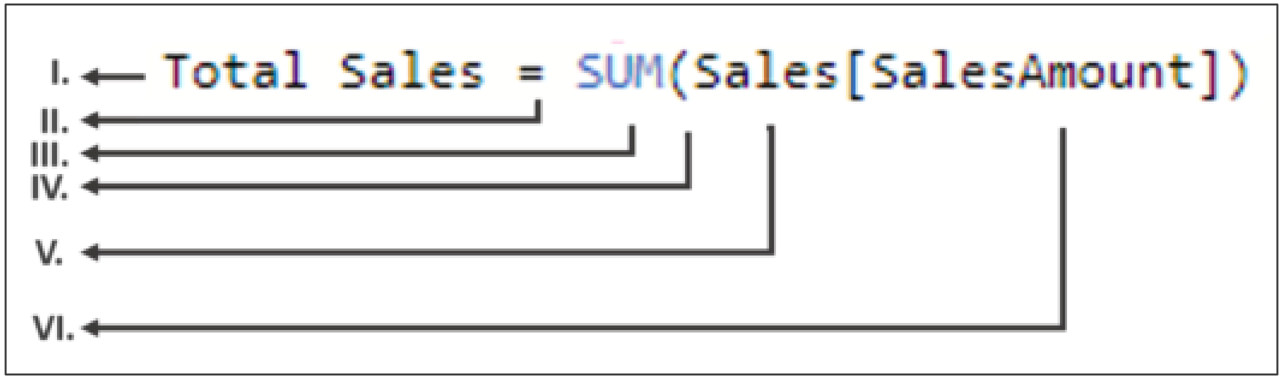
Create a new table derived from another table

Can be used to create a date table when one doesn't exist already

Power BI DAX Basics: How does it work?

There are three fundamental concepts: Syntax, Context, and Functions.

Syntax - Syntax comprises of various components that make up a formula and how it's written. Look at this simple DAX formula.



```
Total Sales = SUM(Sales[SalesAmount])
```

I. Total Sales is the Measure Name.

II. The **equals sign operator (=)** indicates the beginning of the formula.

III. SUM adds up all of the numbers in the column, **Sales[SalesAmount]**.

IV. There are these **parentheses ()** that surround an expression containing one or more arguments. All functions require at least one argument.

V. Sales is the table referenced.

VI. An **argument** passes a value to a function. The referenced column **[SalesAmount]** is an argument with which, the SUM function knows the column on which it has to aggregate a SUM.

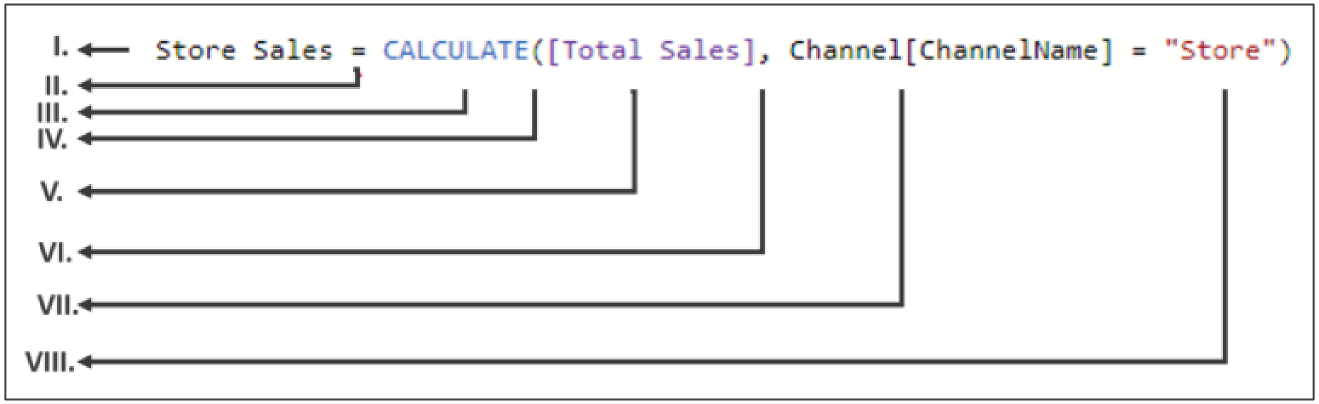
Context

- **Context** is one of the most important of the **3 DAX concepts**. When one speaks of context, this may refer to one of the two types; **Row context** and **Filter context**.

- Used predominantly whilst speaking of Measures, **the Row-Context is most easily thought of as the Current Row.** It applies whenever a formula has a function that applies filters to identify a single row in a table.

- **Filter-Context is a little more difficult to understand than the Row-Context.** You can most easily think of the **Filter-Context as one or more filters applied in a calculation.** The Filter-Context doesn't exist in the Row-context's stead. Rather, it applies in addition to the former.

Context



```
Store Sales = CALCULATE([Total Sales], Channel[ChannelName] = "Store")
```

I. The measure name **Store Sales**.

II. The **equals sign operator (=)** indicates the beginning of the formula.

III. The **CALCULATE** function evaluates an expression, as an argument.

IV. Parenthesis **()** surround an expression containing one or more arguments.

V. A measure **[Total Sales]** in the same table as an expression.

VI. A **comma (,)** separates the first expression argument from the filter argument.

VII. The fully qualified referenced column, **Channel[ChannelName]** is our **Row-Context. Each row in this column specifies a channel, Store, Online, etc.**

VIII. The particular value, **Store** is used as a filter. **This is our Filter-Context.**

This formula ensures that the Total Sales Measure is calculated only for rows in the Channel[ChannelName] Column with the value "Store", as a filter.

Functions

- **Functions** are predefined, structured, and ordered formulae. They perform calculations using arguments passed on to them. These arguments can be numbers, text, logical values, or other functions

DAX Calculation Types: Calculated Columns & Measures

- **The two most common uses for DAX are Calculated Columns and Calculated Measures.**

CALCULATED COLUMNS

Used to add new columns to a table providing more ways to describe and break down the data.

For example, you may add an age column to a customer table so that sales and profit margin can be analyzed and broken down by age demographic. Another common use case for creating calculated columns is to create a unique key on a table, which may be necessary to define a relationship between two tables.

Measures

MEASURES ARE DYNAMIC calculations that recalculate depending on how a report is viewed or filtered.
For example, if a user changed a time-range slider on a report , the measures on that report would be recalculated to reflect the time-range selected. Unlike calculated columns which are calculated during processing of the data model , measures are calculated at runtime when a report is opened or when a user interacts with the filters on a report . Therefore, the results of a measure are always changing and are not stored in your database.

Calculated Columns

- When you create a data model on the Power BI Desktop, you can extend a table by creating new columns. The content of the columns is defined by a DAX expression, evaluated row by row or in the context of the current row across that table.
- **In data models for DAX, however, all calculated columns occupy space in memory and are computed during table processing.**
- This behaviour is helpful in resulting in better user experience but it uses precious RAM and hence, is a bad habit in production because each intermediate calculation is stored in RAM and wastes precious space.

Measures

- There is another way of defining calculations in a DAX model, **useful if you need to operate on aggregate values instead of on a row-by-row basis.** These calculations are measures. One of the requirements of DAX is that a measure needs to be defined in a table. However, the measure does not really belong to the table. So, you can move a measure from one table to another one without losing its functionality.

Calculated Columns vs Measures

- Measures and calculated columns both use DAX expressions. The difference is the context of evaluation. A measure is evaluated in the context of the cell evaluated in a report or in a DAX query, whereas a calculated column is computed at the row level within the table it belongs to.
- Even if they look similar, there is a big difference between calculated columns and measures. The value of a calculated column is computed during a data refresh and uses the current row as a context; it does not depend on user interaction in the report.

When to use Calculated Columns

- **Define a calculated column whenever you want to do the following:**
  - Place the calculated results in a slicer, or see results in rows or columns in a pivot table (as opposed to the values area), or in the axes of a chart, or use the result as a filter condition in a DAX query.
  - Define an expression that is strictly bound to the current row. For example, **Price * Quantity** cannot work on an average or on a sum of the two columns.
  - Categorize text or numbers. For example, a range of values for a measure.

When to use a Measure

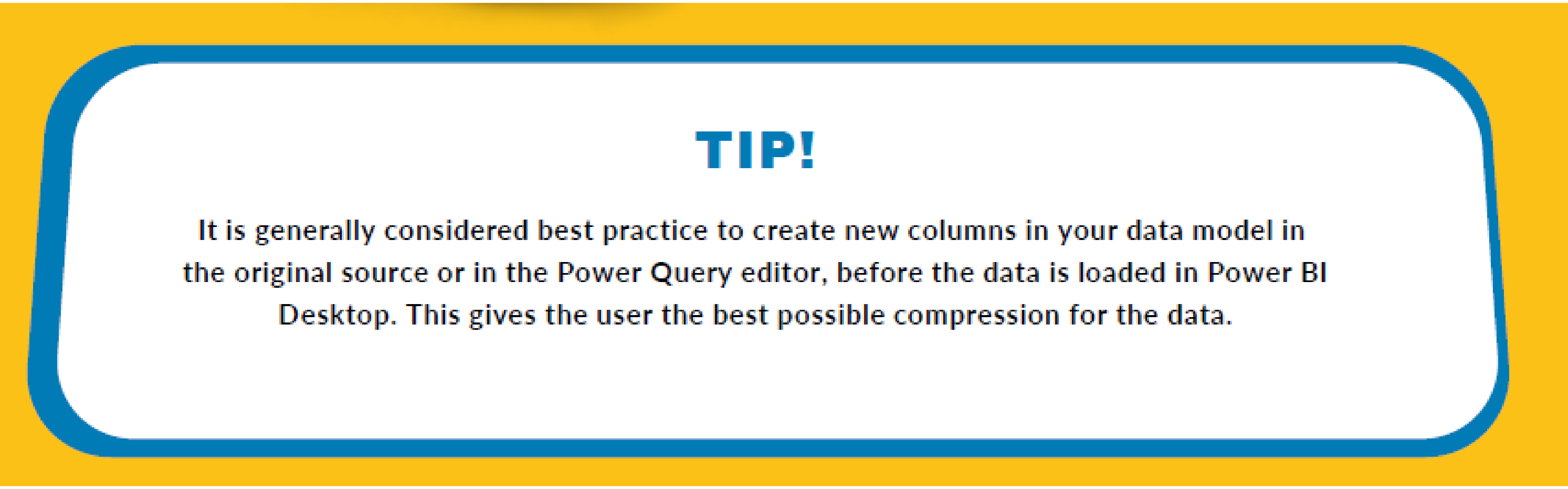
- A measure operates on aggregations of data defined by the current context, which depends on the filter applied in the report – such as slicer, rows, and columns selection in a pivot table, or axes and filters applied to a chart.
- So, you must define a measure whenever you want to display resulting calculation values that reflect user selections, such as;

- When you calculate the profit percentage on a certain selection of data.
- When you Calculate Ratios of a Product compared to all Products but keep the filter both by year and region.

COLUMNS VS MEASURES

The results of calculated columns are immediately viewable in the table.	The results of calculated measures are not stored in the table.
Calculated columns can be used in filters and slicers.	Calculated measures can not be used in filters and slicers.
Calculated columns are updated when the data model is refreshed.	Calculated measures are dynamic and calculated as filters are applied.
Calculated columns take up more space in the data model.	Calculated measures do not take up space in the model.
Calculated columns are not dynamic.	Calculated measures are dynamic and always changing based on the filters applied.

Best Practise

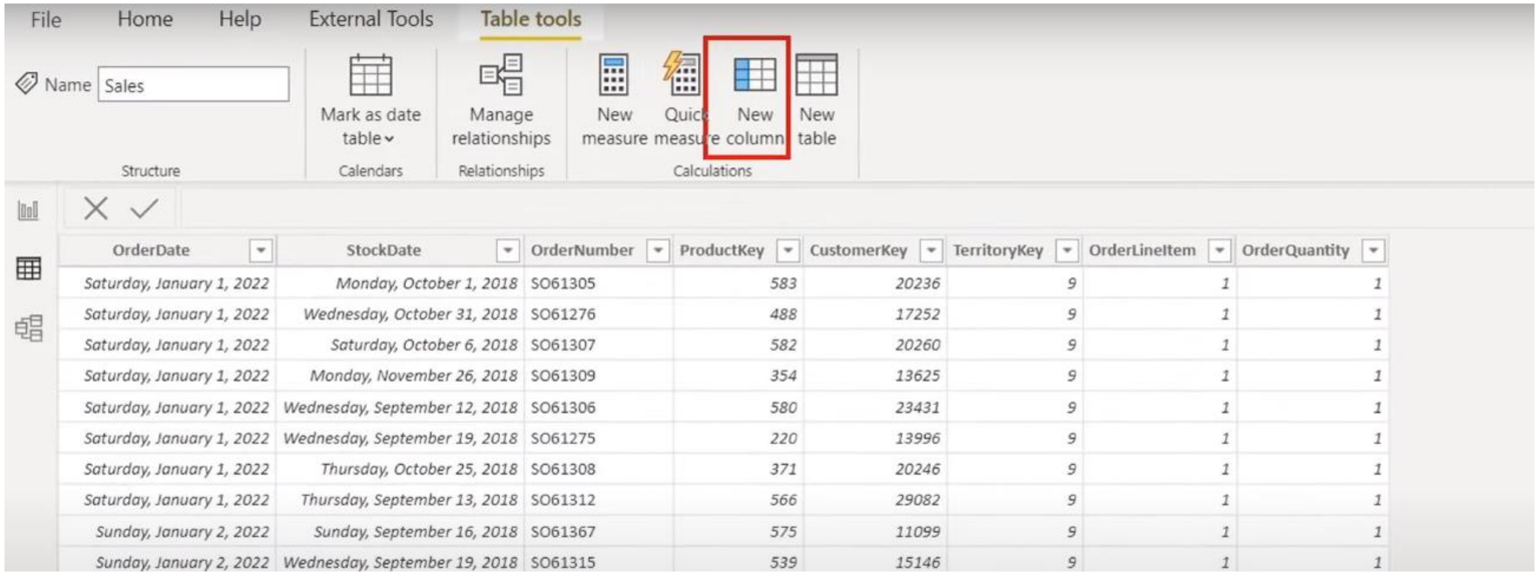


TIP!

It is generally considered best practice to create new columns in your data model in the original source or in the Power Query editor, before the data is loaded in Power BI Desktop. This gives the user the best possible compression for the data.

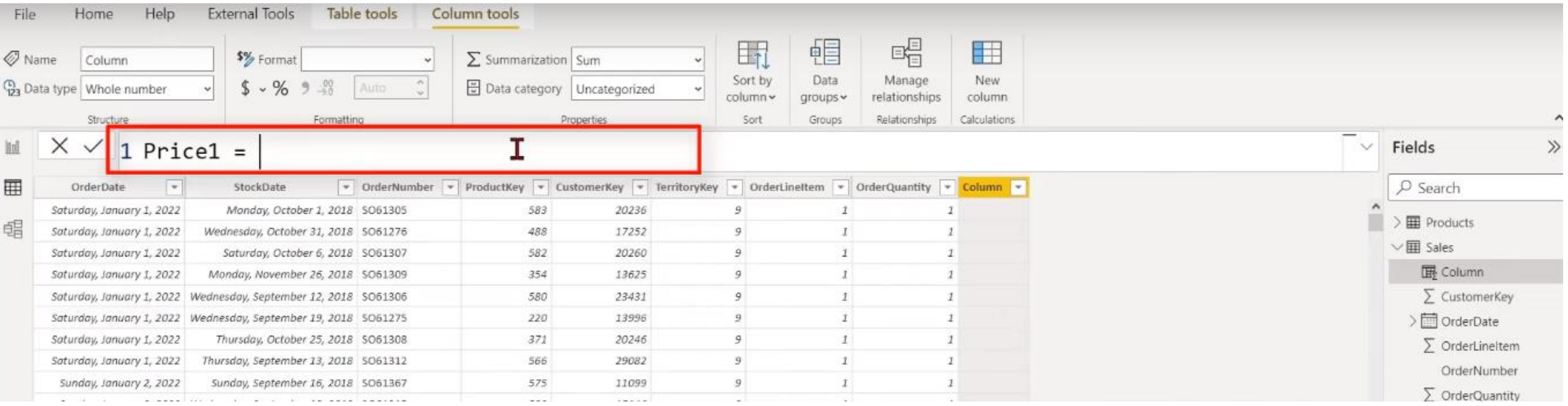
Create Calculated Column

Create a New Column



Name: Sales

OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity
Saturday, January 1, 2022	Monday, October 1, 2018	SO61305	583	20236	9	1	1
Saturday, January 1, 2022	Wednesday, October 31, 2018	SO61276	488	17252	9	1	1
Saturday, January 1, 2022	Saturday, October 6, 2018	SO61307	582	20260	9	1	1
Saturday, January 1, 2022	Monday, November 26, 2018	SO61309	354	13625	9	1	1
Saturday, January 1, 2022	Wednesday, September 12, 2018	SO61306	580	23431	9	1	1
Saturday, January 1, 2022	Wednesday, September 19, 2018	SO61275	220	13996	9	1	1
Saturday, January 1, 2022	Thursday, October 25, 2018	SO61308	371	20246	9	1	1
Saturday, January 1, 2022	Thursday, September 13, 2018	SO61312	566	29082	9	1	1
Sunday, January 2, 2022	Sunday, September 16, 2018	SO61367	575	11099	9	1	1
Sunday, January 2, 2022	Wednesday, September 19, 2018	SO61315	539	15146	9	1	1



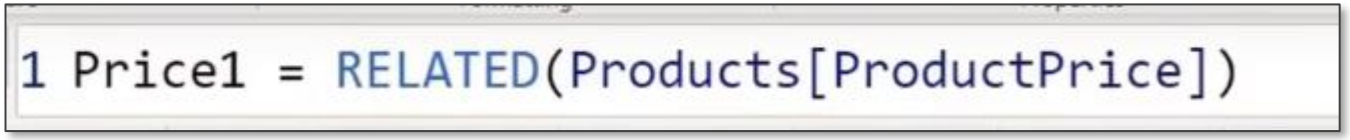
Name: Column
Data type: Whole number
Summarization: Sum
Data category: Uncategorized

```
1 Price1 =
```

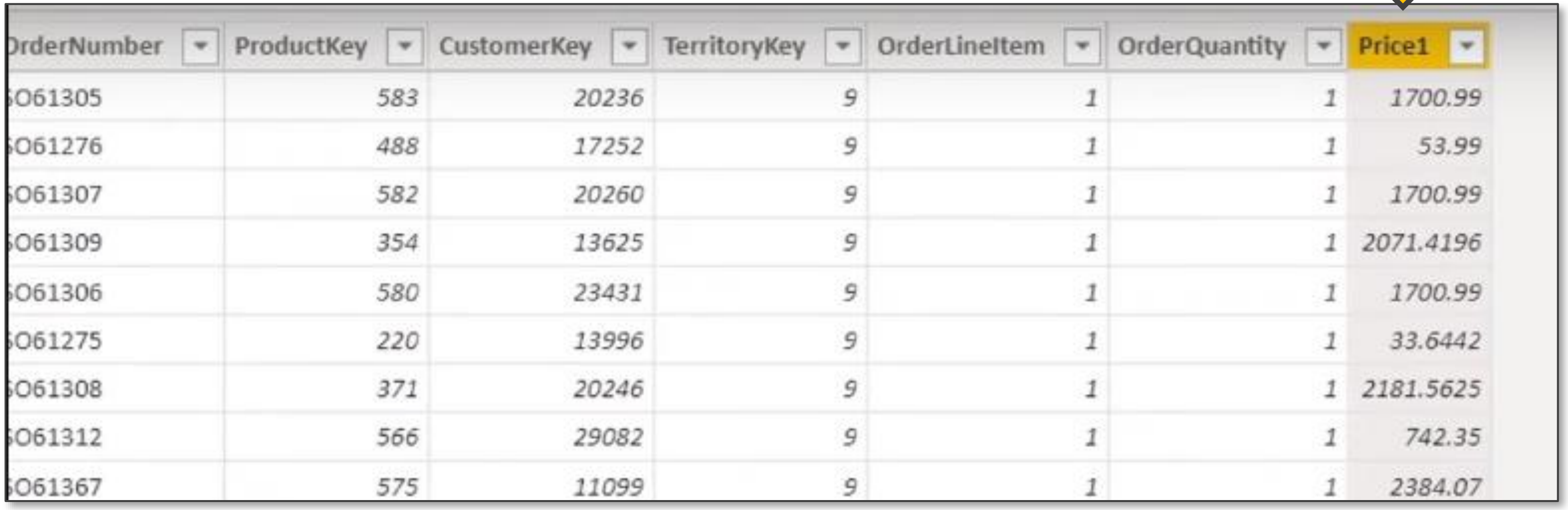
OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity	Column
Saturday, January 1, 2022	Monday, October 1, 2018	SO61305	583	20236	9	1	1	
Saturday, January 1, 2022	Wednesday, October 31, 2018	SO61276	488	17252	9	1	1	
Saturday, January 1, 2022	Saturday, October 6, 2018	SO61307	582	20260	9	1	1	
Saturday, January 1, 2022	Monday, November 26, 2018	SO61309	354	13625	9	1	1	
Saturday, January 1, 2022	Wednesday, September 12, 2018	SO61306	580	23431	9	1	1	
Saturday, January 1, 2022	Wednesday, September 19, 2018	SO61275	220	13996	9	1	1	
Saturday, January 1, 2022	Thursday, October 25, 2018	SO61308	371	20246	9	1	1	
Saturday, January 1, 2022	Thursday, September 13, 2018	SO61312	566	29082	9	1	1	
Sunday, January 2, 2022	Sunday, September 16, 2018	SO61367	575	11099	9	1	1	

- Replacement Function in Power BI Dax is RELATED

It’s the VLOOKUP of Excel that's replaced in Power BI with RELATED



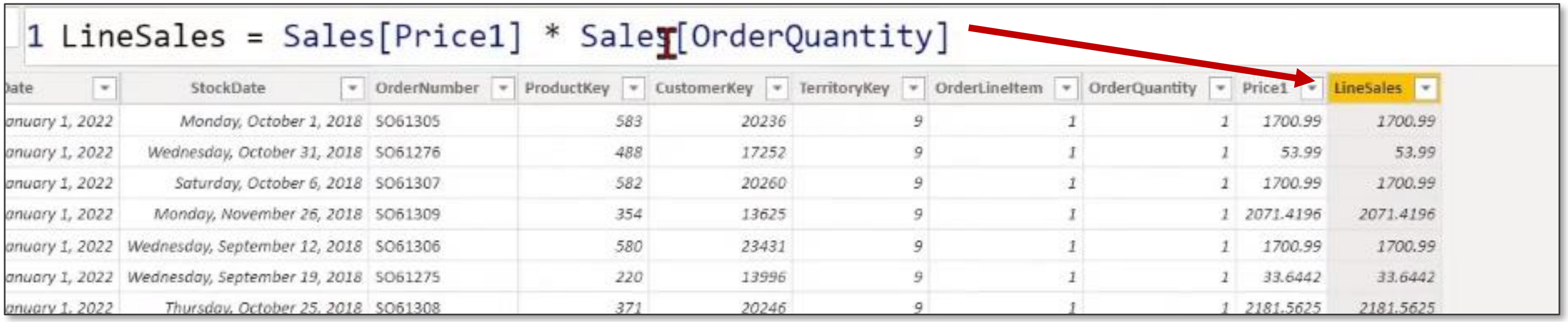
```
1 Price1 = RELATED(Products[ProductPrice])
```



rderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity	Price1
O61305	583	20236	9	1	1	1700.99
O61276	488	17252	9	1	1	53.99
O61307	582	20260	9	1	1	1700.99
O61309	354	13625	9	1	1	2071.4196
O61306	580	23431	9	1	1	1700.99
O61275	220	13996	9	1	1	33.6442
O61308	371	20246	9	1	1	2181.5625
O61312	566	29082	9	1	1	742.35
O61367	575	11099	9	1	1	2384.07

- Delete the relationship and check, the values in new column disappear.
- Create the relationship again, notice that values pop up again.

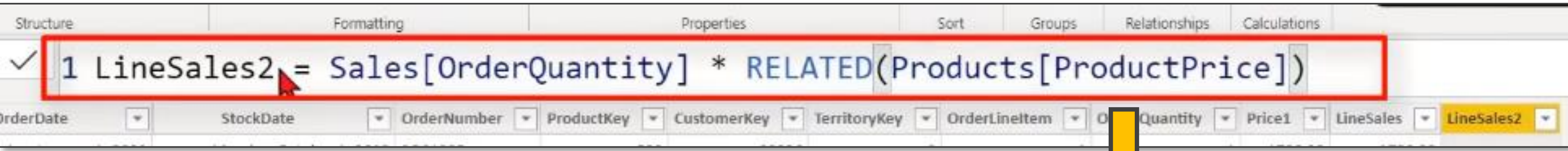
- Create a New Column for Sales



```
LineSales = Sales[Price1] * Sales[OrderQuantity]
```

ate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity	Price1	LineSales
anuary 1, 2022	Monday, October 1, 2018	SO61305	583	20236	9	1	1	1700.99	1700.99
anuary 1, 2022	Wednesday, October 31, 2018	SO61276	488	17252	9	1	1	53.99	53.99
anuary 1, 2022	Saturday, October 6, 2018	SO61307	582	20260	9	1	1	1700.99	1700.99
anuary 1, 2022	Monday, November 26, 2018	SO61309	354	13625	9	1	1	2071.4196	2071.4196
anuary 1, 2022	Wednesday, September 12, 2018	SO61306	580	23431	9	1	1	1700.99	1700.99
anuary 1, 2022	Wednesday, September 19, 2018	SO61275	220	13996	9	1	1	33.6442	33.6442
anuary 1, 2022	Thursday, October 25, 2018	SO61308	371	20246	9	1	1	2181.5625	2181.5625

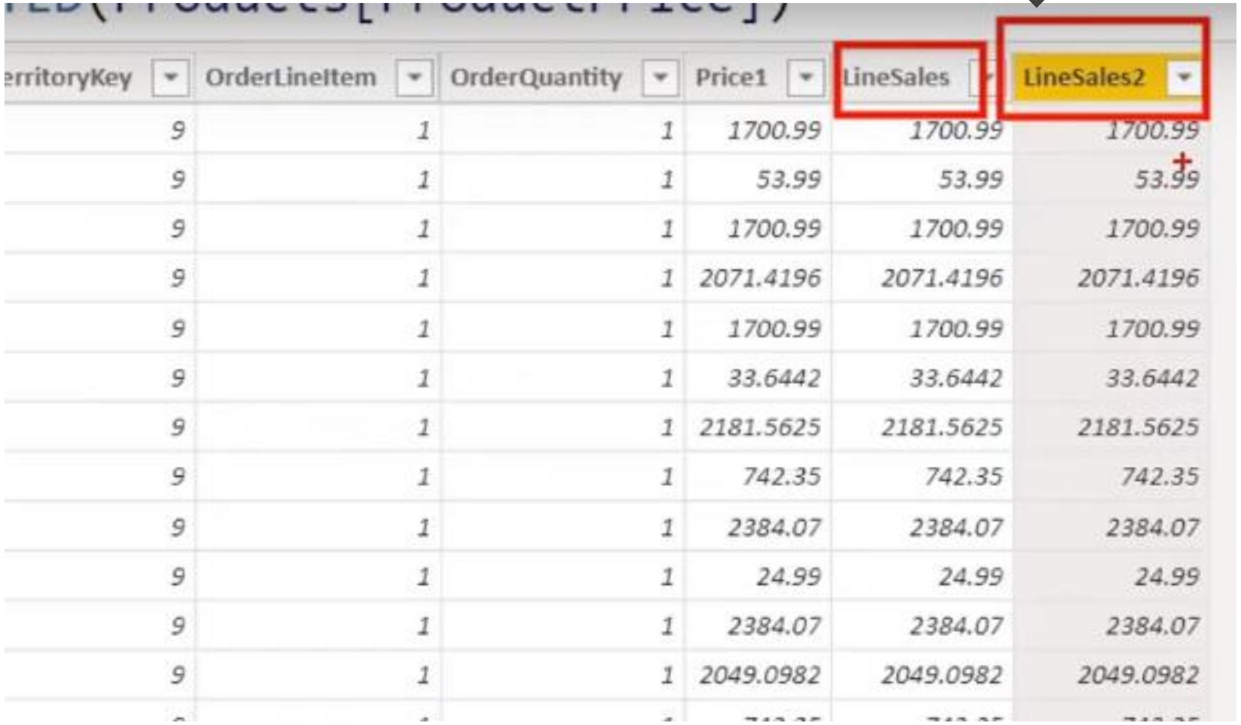
- Combine the previous two steps into one



Structure | Formatting | Properties | Sort | Groups | Relationships | Calculations

```
1 LineSales2 = Sales[OrderQuantity] * RELATED(Products[ProductPrice])
```

rderDate | StockDate | OrderNumber | ProductKey | CustomerKey | TerritoryKey | OrderLineItem | O Quantity | Price1 | LineSales | LineSales2

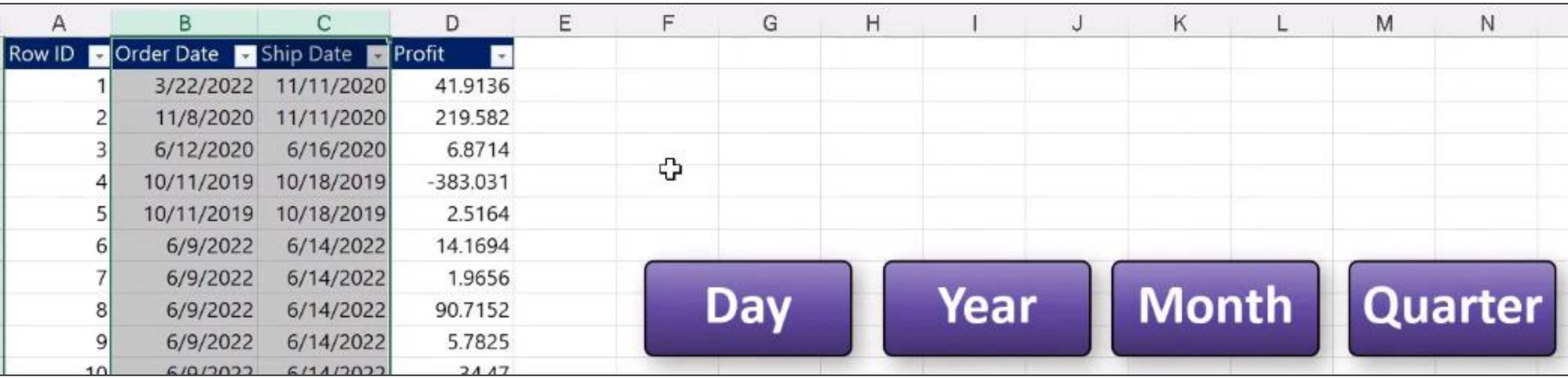


rritoryKey	OrderLineItem	OrderQuantity	Price1	LineSales	LineSales2
9	1	1	1700.99	1700.99	1700.99
9	1	1	53.99	53.99	53.99
9	1	1	1700.99	1700.99	1700.99
9	1	1	2071.4196	2071.4196	2071.4196
9	1	1	1700.99	1700.99	1700.99
9	1	1	33.6442	33.6442	33.6442
9	1	1	2181.5625	2181.5625	2181.5625
9	1	1	742.35	742.35	742.35
9	1	1	2384.07	2384.07	2384.07
9	1	1	24.99	24.99	24.99
9	1	1	2384.07	2384.07	2384.07
9	1	1	2049.0982	2049.0982	2049.0982

Basic Dax Functions (Text Functions)

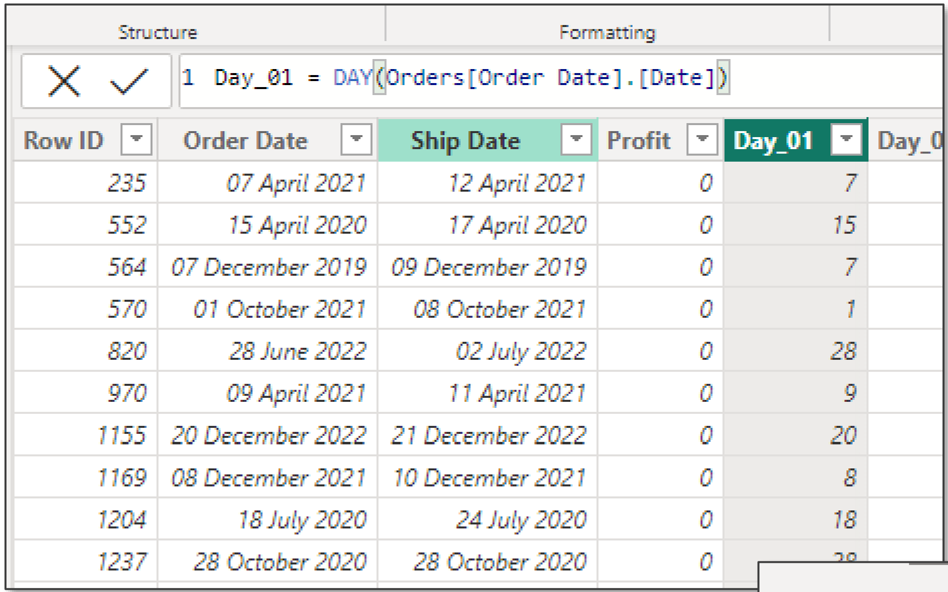
- Basic Date Functions
- Basic Text Functions
- Basic Number Functions
- Basic Logical Functions
- Advanced DAX – CALCULATE() Function

Basic Date Functions



Row ID	Order Date	Ship Date	Profit
1	3/22/2022	11/11/2020	41.9136
2	11/8/2020	11/11/2020	219.582
3	6/12/2020	6/16/2020	6.8714
4	10/11/2019	10/18/2019	-383.031
5	10/11/2019	10/18/2019	2.5164
6	6/9/2022	6/14/2022	14.1694
7	6/9/2022	6/14/2022	1.9656
8	6/9/2022	6/14/2022	90.7152
9	6/9/2022	6/14/2022	5.7825
10	6/9/2022	6/14/2022	34.47

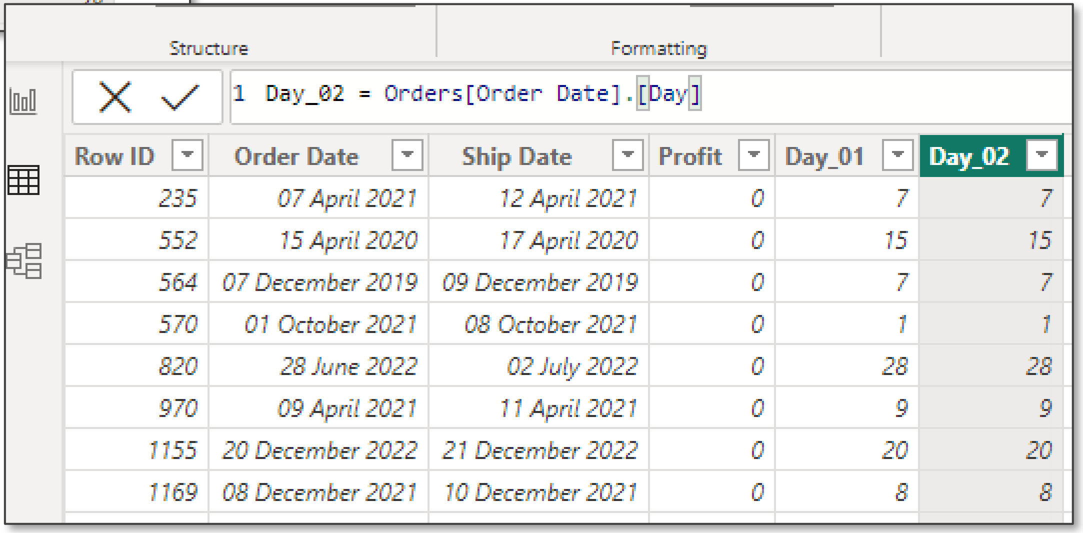
DAY() function



Structure | Formatting

```
Day_01 = DAY(Orders[Order Date].[Date])
```

Row ID	Order Date	Ship Date	Profit	Day_01	Day_0
235	07 April 2021	12 April 2021	0	7	
552	15 April 2020	17 April 2020	0	15	
564	07 December 2019	09 December 2019	0	7	
570	01 October 2021	08 October 2021	0	1	
820	28 June 2022	02 July 2022	0	28	
970	09 April 2021	11 April 2021	0	9	
1155	20 December 2022	21 December 2022	0	20	
1169	08 December 2021	10 December 2021	0	8	
1204	18 July 2020	24 July 2020	0	18	
1237	28 October 2020	28 October 2020	0	28	

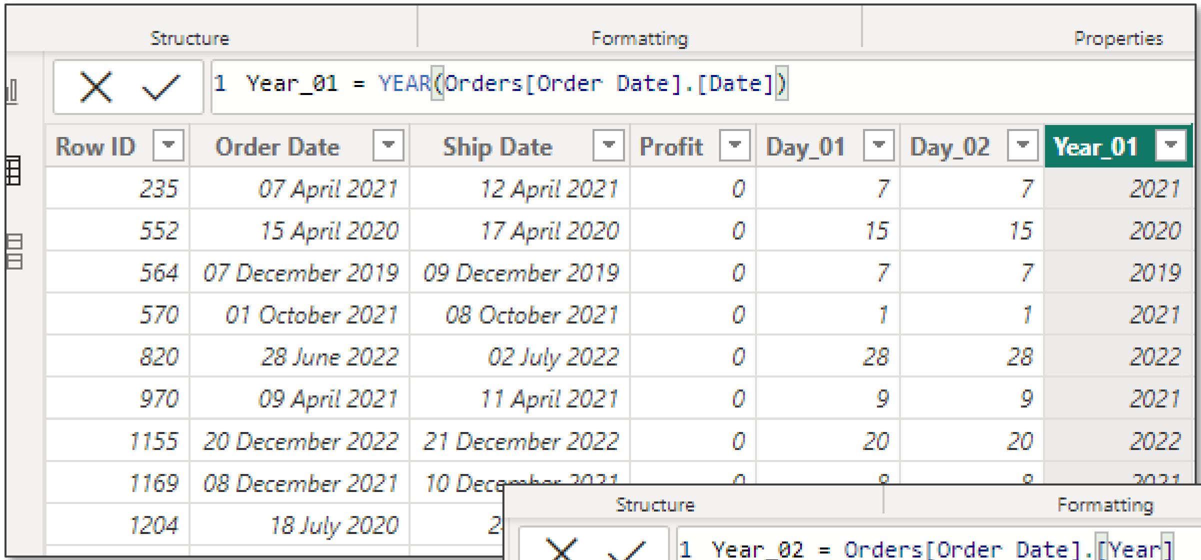


Structure | Formatting

```
Day_02 = Orders[Order Date].[Day]
```

Row ID	Order Date	Ship Date	Profit	Day_01	Day_02
235	07 April 2021	12 April 2021	0	7	7
552	15 April 2020	17 April 2020	0	15	15
564	07 December 2019	09 December 2019	0	7	7
570	01 October 2021	08 October 2021	0	1	1
820	28 June 2022	02 July 2022	0	28	28
970	09 April 2021	11 April 2021	0	9	9
1155	20 December 2022	21 December 2022	0	20	20
1169	08 December 2021	10 December 2021	0	8	8

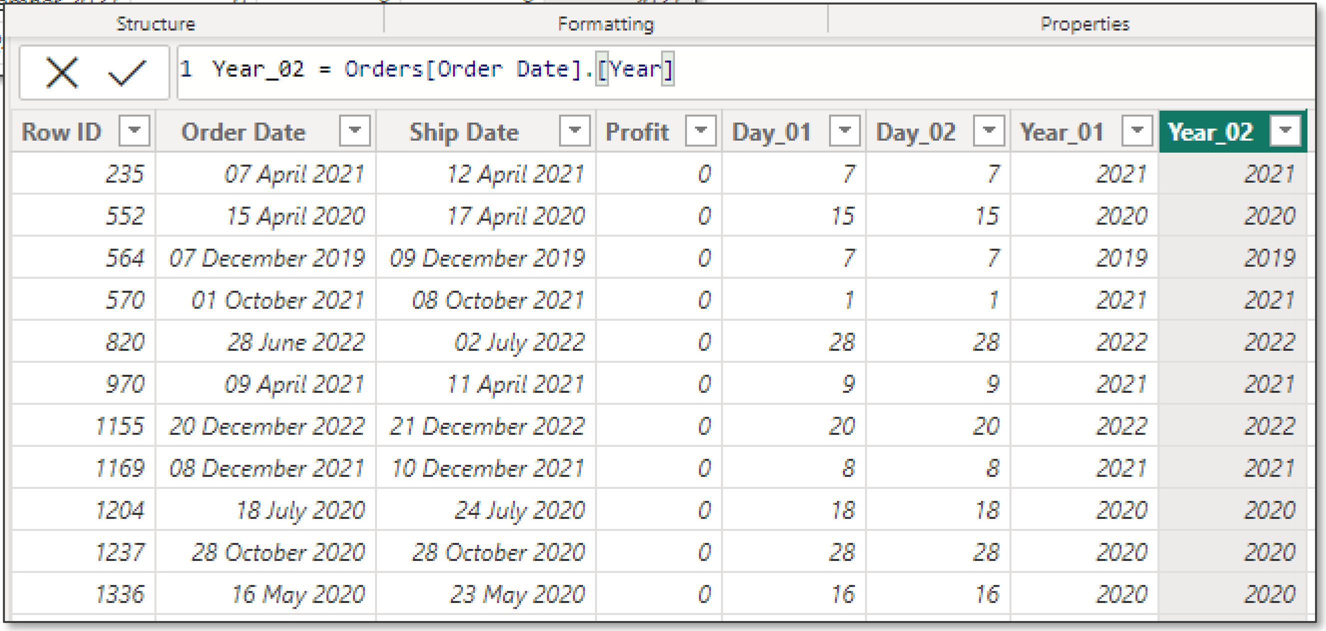
YEAR() Function



Structure | Formatting | Properties

```
Year_01 = YEAR(Orders[Order Date].[Date])
```

Row ID	Order Date	Ship Date	Profit	Day_01	Day_02	Year_01
235	07 April 2021	12 April 2021	0	7	7	2021
552	15 April 2020	17 April 2020	0	15	15	2020
564	07 December 2019	09 December 2019	0	7	7	2019
570	01 October 2021	08 October 2021	0	1	1	2021
820	28 June 2022	02 July 2022	0	28	28	2022
970	09 April 2021	11 April 2021	0	9	9	2021
1155	20 December 2022	21 December 2022	0	20	20	2022
1169	08 December 2021	10 Dece				
1204	18 July 2020	2				

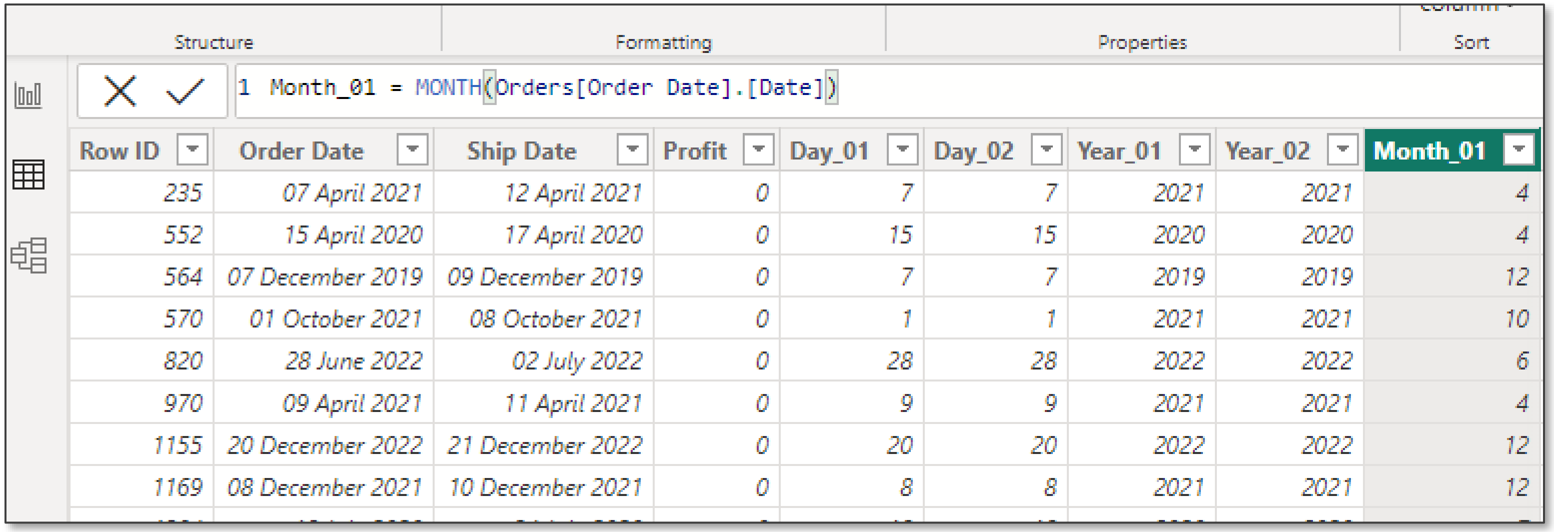


Structure | Formatting | Properties

```
Year_02 = Orders[Order Date].[Year]
```

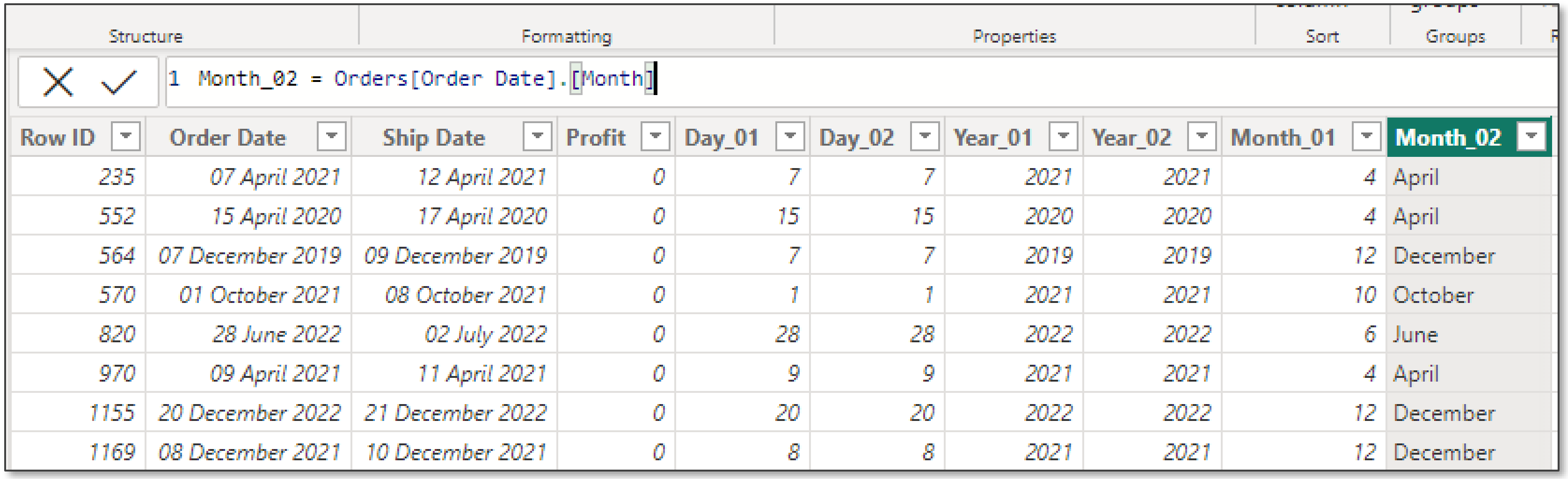
Row ID	Order Date	Ship Date	Profit	Day_01	Day_02	Year_01	Year_02
235	07 April 2021	12 April 2021	0	7	7	2021	2021
552	15 April 2020	17 April 2020	0	15	15	2020	2020
564	07 December 2019	09 December 2019	0	7	7	2019	2019
570	01 October 2021	08 October 2021	0	1	1	2021	2021
820	28 June 2022	02 July 2022	0	28	28	2022	2022
970	09 April 2021	11 April 2021	0	9	9	2021	2021
1155	20 December 2022	21 December 2022	0	20	20	2022	2022
1169	08 December 2021	10 December 2021	0	8	8	2021	2021
1204	18 July 2020	24 July 2020	0	18	18	2020	2020
1237	28 October 2020	28 October 2020	0	28	28	2020	2020
1336	16 May 2020	23 May 2020	0	16	16	2020	2020

MONTH() Function



```
Month_01 = MONTH(Orders[Order Date].[Date])
```

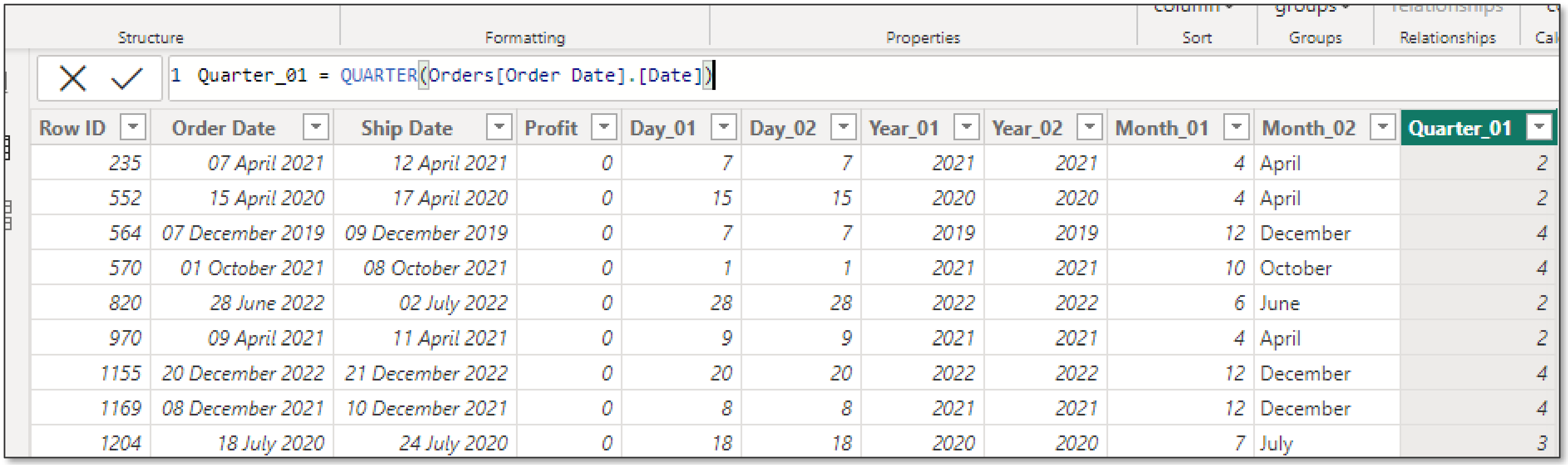
Row ID	Order Date	Ship Date	Profit	Day_01	Day_02	Year_01	Year_02	Month_01
235	07 April 2021	12 April 2021	0	7	7	2021	2021	4
552	15 April 2020	17 April 2020	0	15	15	2020	2020	4
564	07 December 2019	09 December 2019	0	7	7	2019	2019	12
570	01 October 2021	08 October 2021	0	1	1	2021	2021	10
820	28 June 2022	02 July 2022	0	28	28	2022	2022	6
970	09 April 2021	11 April 2021	0	9	9	2021	2021	4
1155	20 December 2022	21 December 2022	0	20	20	2022	2022	12
1169	08 December 2021	10 December 2021	0	8	8	2021	2021	12



```
Month_02 = Orders[Order Date].[Month]
```

Row ID	Order Date	Ship Date	Profit	Day_01	Day_02	Year_01	Year_02	Month_01	Month_02
235	07 April 2021	12 April 2021	0	7	7	2021	2021	4	April
552	15 April 2020	17 April 2020	0	15	15	2020	2020	4	April
564	07 December 2019	09 December 2019	0	7	7	2019	2019	12	December
570	01 October 2021	08 October 2021	0	1	1	2021	2021	10	October
820	28 June 2022	02 July 2022	0	28	28	2022	2022	6	June
970	09 April 2021	11 April 2021	0	9	9	2021	2021	4	April
1155	20 December 2022	21 December 2022	0	20	20	2022	2022	12	December
1169	08 December 2021	10 December 2021	0	8	8	2021	2021	12	December

QUARTER() Function

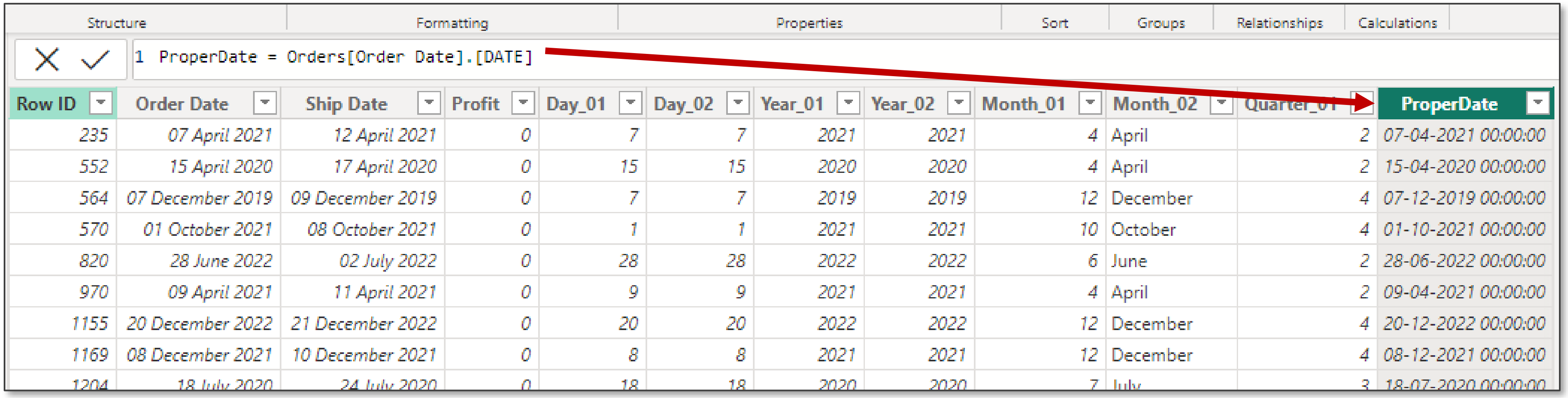


```
1  Quarter_01 = QUARTER(Orders[Order Date].[Date])
```

Row ID	Order Date	Ship Date	Profit	Day_01	Day_02	Year_01	Year_02	Month_01	Month_02	Quarter_01
235	07 April 2021	12 April 2021	0	7	7	2021	2021	4	April	2
552	15 April 2020	17 April 2020	0	15	15	2020	2020	4	April	2
564	07 December 2019	09 December 2019	0	7	7	2019	2019	12	December	4
570	01 October 2021	08 October 2021	0	1	1	2021	2021	10	October	4
820	28 June 2022	02 July 2022	0	28	28	2022	2022	6	June	2
970	09 April 2021	11 April 2021	0	9	9	2021	2021	4	April	2
1155	20 December 2022	21 December 2022	0	20	20	2022	2022	12	December	4
1169	08 December 2021	10 December 2021	0	8	8	2021	2021	12	December	4
1204	18 July 2020	24 July 2020	0	18	18	2020	2020	7	July	3

.[DATE] Function

.[DATE] Converts Date to a proper Date and Time Format



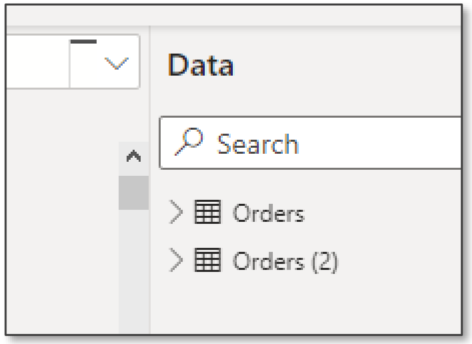
```
1  ProperDate = Orders[Order Date].[DATE]
```

Row ID	Order Date	Ship Date	Profit	Day_01	Day_02	Year_01	Year_02	Month_01	Month_02	Quarter_01	ProperDate
235	07 April 2021	12 April 2021	0	7	7	2021	2021	4	April	2	07-04-2021 00:00:00
552	15 April 2020	17 April 2020	0	15	15	2020	2020	4	April	2	15-04-2020 00:00:00
564	07 December 2019	09 December 2019	0	7	7	2019	2019	12	December	4	07-12-2019 00:00:00
570	01 October 2021	08 October 2021	0	1	1	2021	2021	10	October	4	01-10-2021 00:00:00
820	28 June 2022	02 July 2022	0	28	28	2022	2022	6	June	2	28-06-2022 00:00:00
970	09 April 2021	11 April 2021	0	9	9	2021	2021	4	April	2	09-04-2021 00:00:00
1155	20 December 2022	21 December 2022	0	20	20	2022	2022	12	December	4	20-12-2022 00:00:00
1169	08 December 2021	10 December 2021	0	8	8	2021	2021	12	December	4	08-12-2021 00:00:00
1204	18 July 2020	24 July 2020	0	18	18	2020	2020	7	July	3	18-07-2020 00:00:00

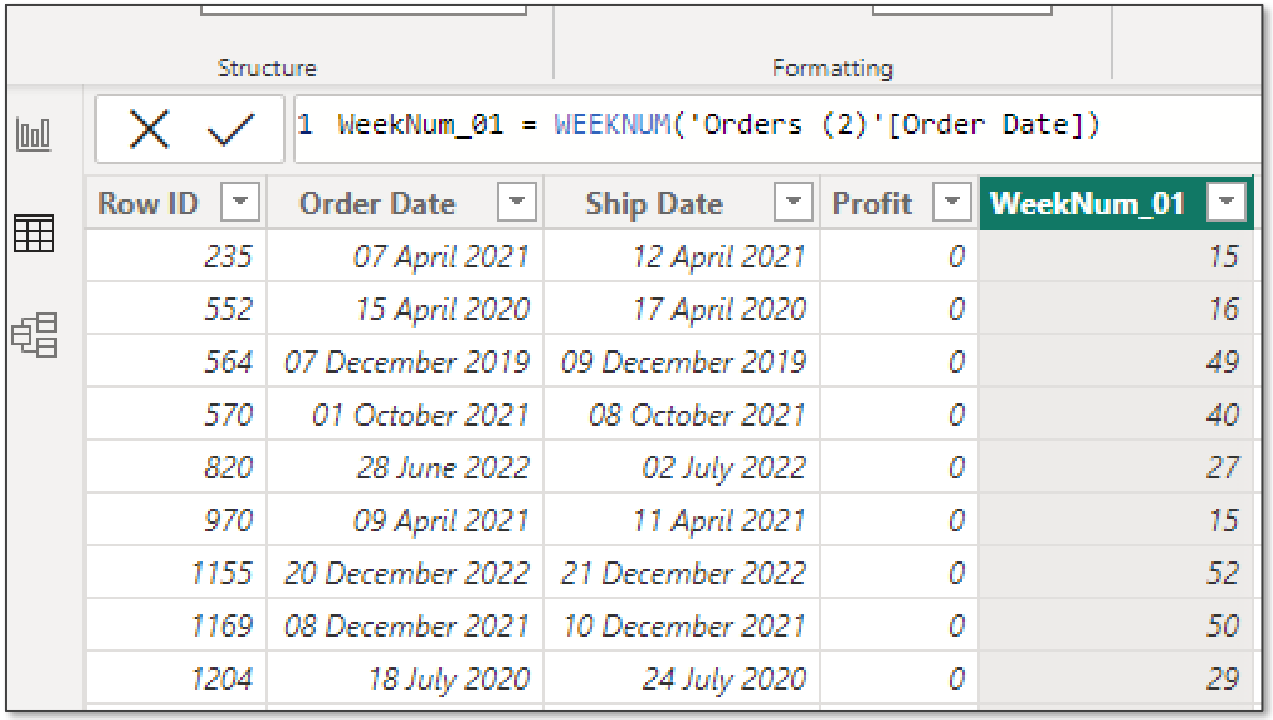
Note: Most of the Functions will be using a .[DATE] Format

More Date Functions

Go to Power Query Editor & Duplicate the Query



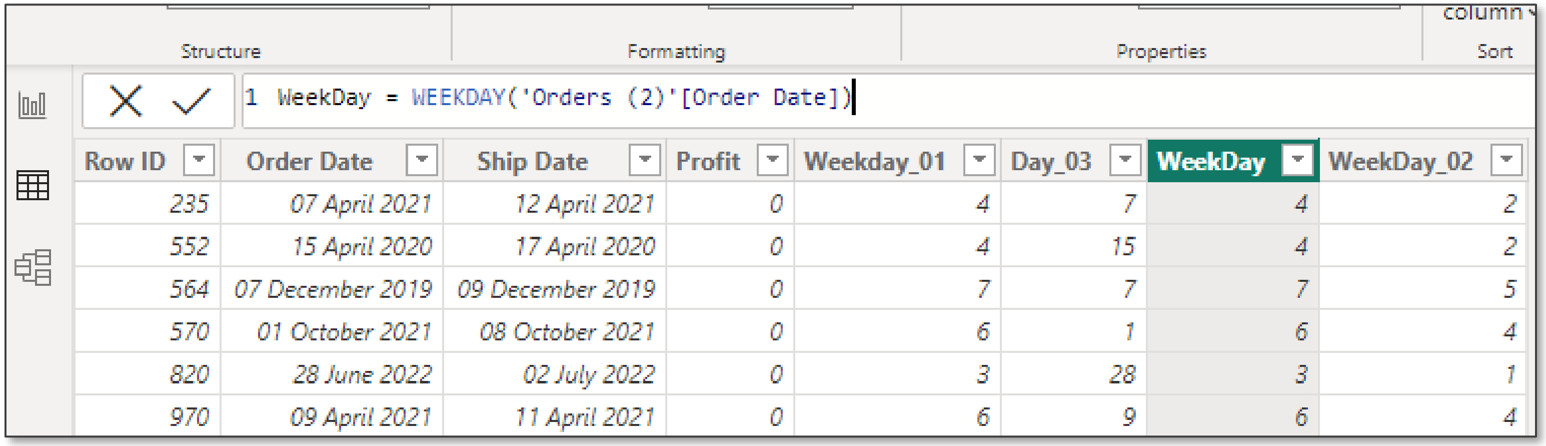
WEEKNUM() Function



```
1 WeekNum_01 = WEEKNUM('Orders (2)'[Order Date])
```

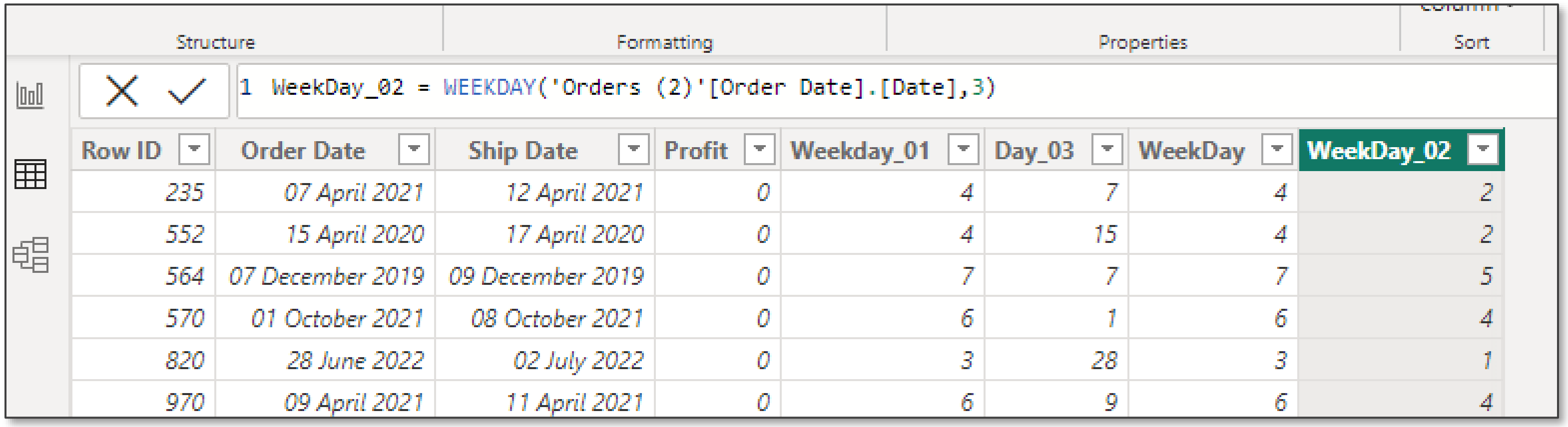
Row ID	Order Date	Ship Date	Profit	WeekNum_01
235	07 April 2021	12 April 2021	0	15
552	15 April 2020	17 April 2020	0	16
564	07 December 2019	09 December 2019	0	49
570	01 October 2021	08 October 2021	0	40
820	28 June 2022	02 July 2022	0	27
970	09 April 2021	11 April 2021	0	15
1155	20 December 2022	21 December 2022	0	52
1169	08 December 2021	10 December 2021	0	50
1204	18 July 2020	24 July 2020	0	29

WEEKDAY(Function) – Default – 1 is Sunday



```
1  WeekDay = WEEKDAY('Orders (2)'[Order Date])
```

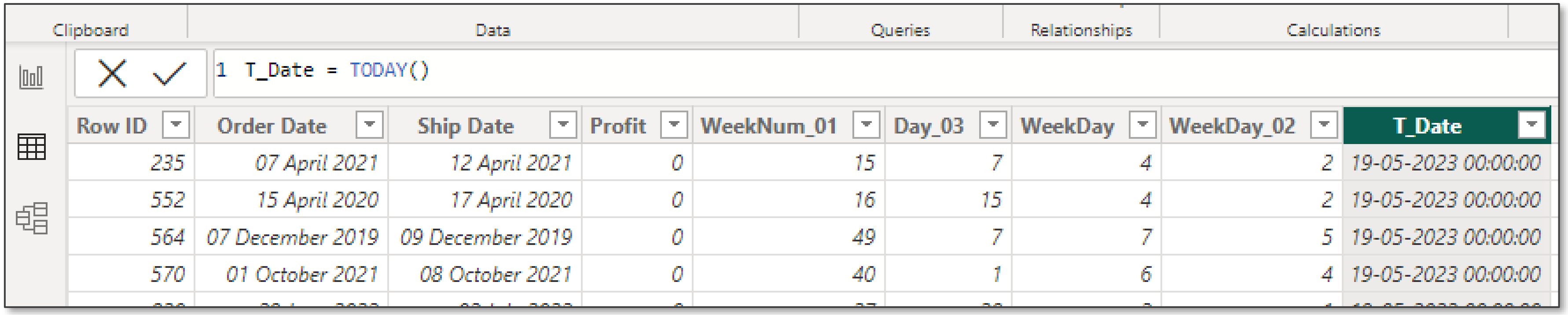
Row ID	Order Date	Ship Date	Profit	Weekday_01	Day_03	WeekDay	WeekDay_02
235	07 April 2021	12 April 2021	0	4	7	4	2
552	15 April 2020	17 April 2020	0	4	15	4	2
564	07 December 2019	09 December 2019	0	7	7	7	5
570	01 October 2021	08 October 2021	0	6	1	6	4
820	28 June 2022	02 July 2022	0	3	28	3	1
970	09 April 2021	11 April 2021	0	6	9	6	4



```
1  WeekDay_02 = WEEKDAY('Orders (2)'[Order Date].[Date],3)
```

Row ID	Order Date	Ship Date	Profit	Weekday_01	Day_03	WeekDay	WeekDay_02
235	07 April 2021	12 April 2021	0	4	7	4	2
552	15 April 2020	17 April 2020	0	4	15	4	2
564	07 December 2019	09 December 2019	0	7	7	7	5
570	01 October 2021	08 October 2021	0	6	1	6	4
820	28 June 2022	02 July 2022	0	3	28	3	1
970	09 April 2021	11 April 2021	0	6	9	6	4

TODAY()

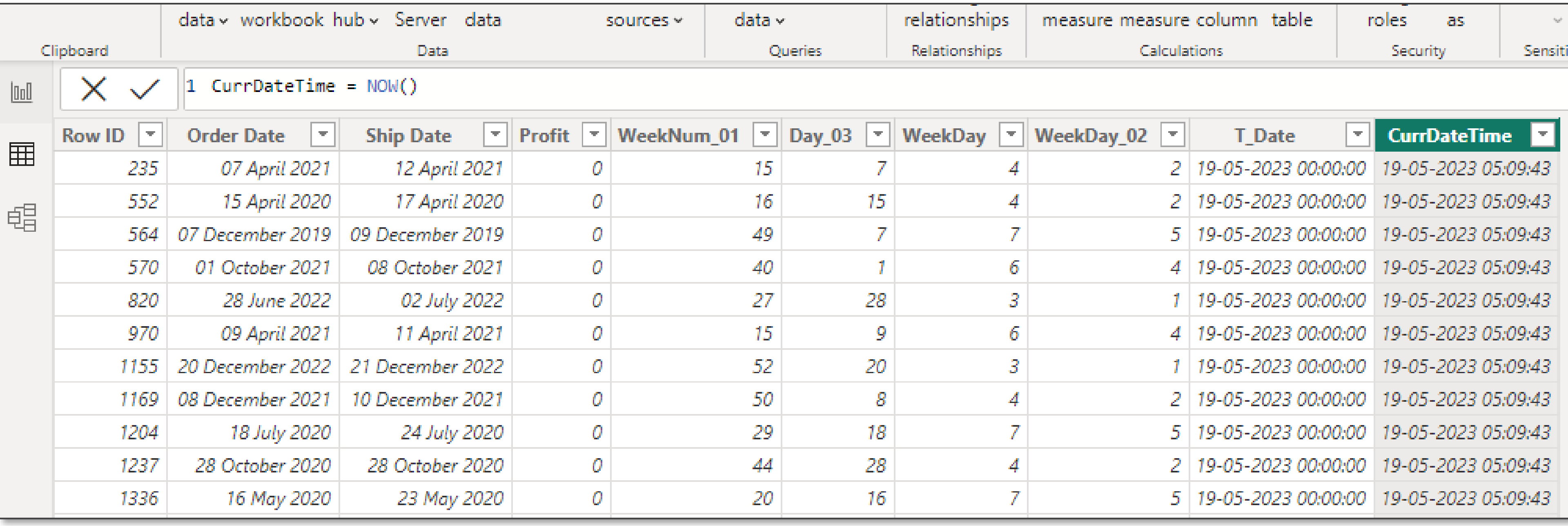


```
1 T_Date = TODAY()
```

Row ID	Order Date	Ship Date	Profit	WeekNum_01	Day_03	WeekDay	WeekDay_02	T_Date
235	07 April 2021	12 April 2021	0	15	7	4	2	19-05-2023 00:00:00
552	15 April 2020	17 April 2020	0	16	15	4	2	19-05-2023 00:00:00
564	07 December 2019	09 December 2019	0	49	7	7	5	19-05-2023 00:00:00
570	01 October 2021	08 October 2021	0	40	1	6	4	19-05-2023 00:00:00

You can change the format of the Date as you like.

NOW()



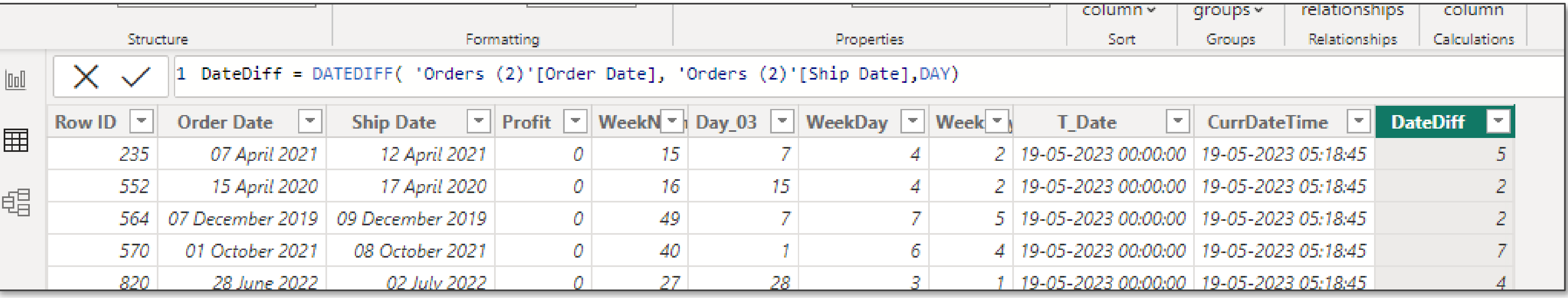
data ⌄ workbook hub ⌄ Server data sources ⌄ data ⌄ relationships measure measure column table roles as
Clipboard Data Queries Relationships Calculations Security Sensiti

```
1 CurrDateTime = NOW()
```

Row ID	Order Date	Ship Date	Profit	WeekNum_01	Day_03	WeekDay	WeekDay_02	T_Date	CurrDateTime
235	07 April 2021	12 April 2021	0	15	7	4	2	19-05-2023 00:00:00	19-05-2023 05:09:43
552	15 April 2020	17 April 2020	0	16	15	4	2	19-05-2023 00:00:00	19-05-2023 05:09:43
564	07 December 2019	09 December 2019	0	49	7	7	5	19-05-2023 00:00:00	19-05-2023 05:09:43
570	01 October 2021	08 October 2021	0	40	1	6	4	19-05-2023 00:00:00	19-05-2023 05:09:43
820	28 June 2022	02 July 2022	0	27	28	3	1	19-05-2023 00:00:00	19-05-2023 05:09:43
970	09 April 2021	11 April 2021	0	15	9	6	4	19-05-2023 00:00:00	19-05-2023 05:09:43
1155	20 December 2022	21 December 2022	0	52	20	3	1	19-05-2023 00:00:00	19-05-2023 05:09:43
1169	08 December 2021	10 December 2021	0	50	8	4	2	19-05-2023 00:00:00	19-05-2023 05:09:43
1204	18 July 2020	24 July 2020	0	29	18	7	5	19-05-2023 00:00:00	19-05-2023 05:09:43
1237	28 October 2020	28 October 2020	0	44	28	4	2	19-05-2023 00:00:00	19-05-2023 05:09:43
1336	16 May 2020	23 May 2020	0	20	16	7	5	19-05-2023 00:00:00	19-05-2023 05:09:43

Refresh and Check

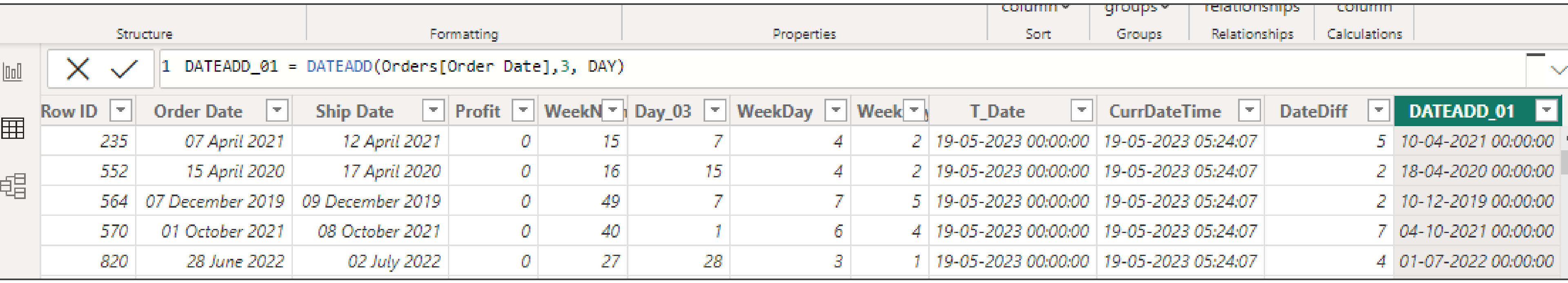
DATEDIFF()



```
1 DateDiff = DATEDIFF( 'Orders (2)'[Order Date], 'Orders (2)'[Ship Date],DAY)
```

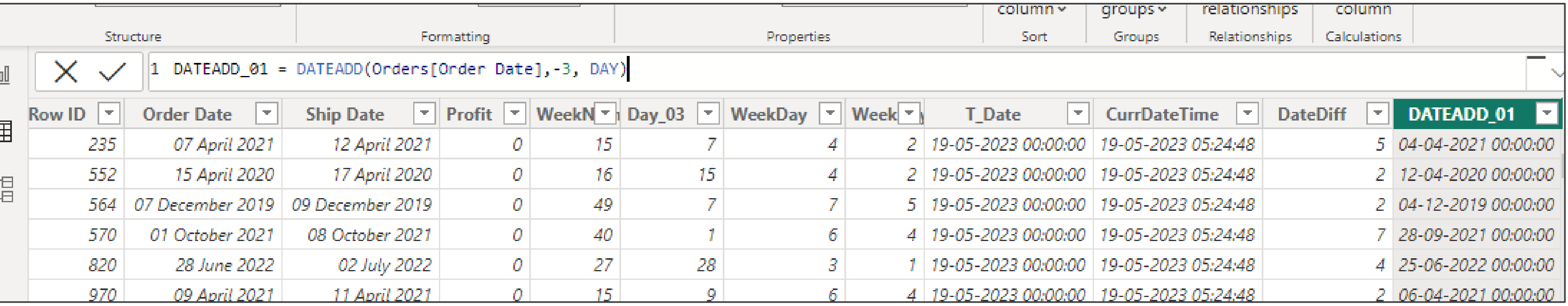
Row ID	Order Date	Ship Date	Profit	WeekN	Day_03	WeekDay	Week	T_Date	CurrDateTime	DateDiff
235	07 April 2021	12 April 2021	0	15	7	4	2	19-05-2023 00:00:00	19-05-2023 05:18:45	5
552	15 April 2020	17 April 2020	0	16	15	4	2	19-05-2023 00:00:00	19-05-2023 05:18:45	2
564	07 December 2019	09 December 2019	0	49	7	7	5	19-05-2023 00:00:00	19-05-2023 05:18:45	2
570	01 October 2021	08 October 2021	0	40	1	6	4	19-05-2023 00:00:00	19-05-2023 05:18:45	7
820	28 June 2022	02 July 2022	0	27	28	3	1	19-05-2023 00:00:00	19-05-2023 05:18:45	4

DATEADD()



```
DATEADD_01 = DATEADD(Orders[Order Date],3, DAY)
```

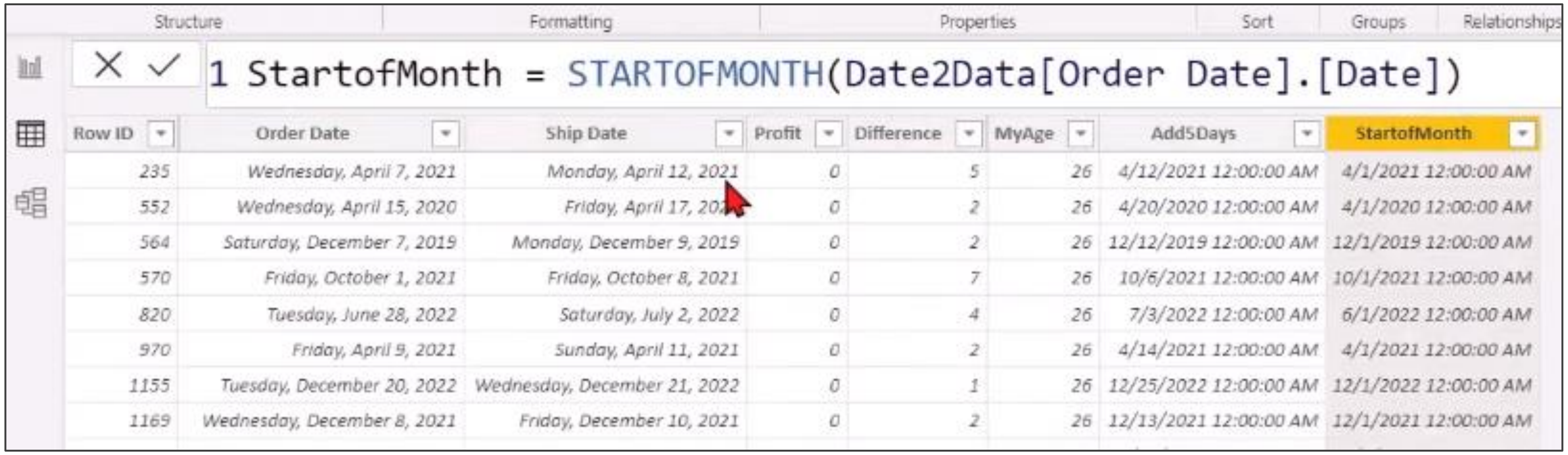
Row ID	Order Date	Ship Date	Profit	WeekN	Day_03	WeekDay	Week	T_Date	CurrDateTime	DateDiff	DATEADD_01
235	07 April 2021	12 April 2021	0	15	7	4	2	19-05-2023 00:00:00	19-05-2023 05:24:07	5	10-04-2021 00:00:00
552	15 April 2020	17 April 2020	0	16	15	4	2	19-05-2023 00:00:00	19-05-2023 05:24:07	2	18-04-2020 00:00:00
564	07 December 2019	09 December 2019	0	49	7	7	5	19-05-2023 00:00:00	19-05-2023 05:24:07	2	10-12-2019 00:00:00
570	01 October 2021	08 October 2021	0	40	1	6	4	19-05-2023 00:00:00	19-05-2023 05:24:07	7	04-10-2021 00:00:00
820	28 June 2022	02 July 2022	0	27	28	3	1	19-05-2023 00:00:00	19-05-2023 05:24:07	4	01-07-2022 00:00:00



```
DATEADD_01 = DATEADD(Orders[Order Date],-3, DAY)
```

Row ID	Order Date	Ship Date	Profit	WeekN	Day_03	WeekDay	Week	T_Date	CurrDateTime	DateDiff	DATEADD_01
235	07 April 2021	12 April 2021	0	15	7	4	2	19-05-2023 00:00:00	19-05-2023 05:24:48	5	04-04-2021 00:00:00
552	15 April 2020	17 April 2020	0	16	15	4	2	19-05-2023 00:00:00	19-05-2023 05:24:48	2	12-04-2020 00:00:00
564	07 December 2019	09 December 2019	0	49	7	7	5	19-05-2023 00:00:00	19-05-2023 05:24:48	2	04-12-2019 00:00:00
570	01 October 2021	08 October 2021	0	40	1	6	4	19-05-2023 00:00:00	19-05-2023 05:24:48	7	28-09-2021 00:00:00
820	28 June 2022	02 July 2022	0	27	28	3	1	19-05-2023 00:00:00	19-05-2023 05:24:48	4	25-06-2022 00:00:00
970	09 April 2021	11 April 2021	0	15	9	6	4	19-05-2023 00:00:00	19-05-2023 05:24:48	2	06-04-2021 00:00:00

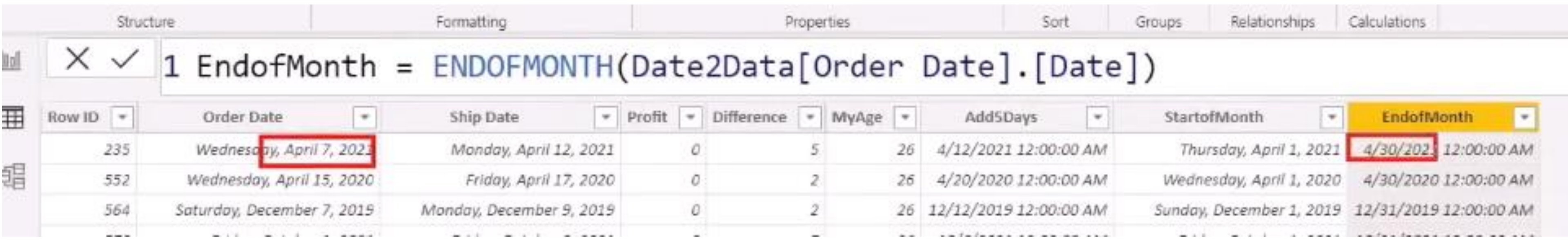
STARTOFMONTH()



```
1 StartofMonth = STARTOFMONTH(Date2Data[Order Date].[Date])
```

Row ID	Order Date	Ship Date	Profit	Difference	MyAge	Add5Days	StartofMonth
235	Wednesday, April 7, 2021	Monday, April 12, 2021	0	5	26	4/12/2021 12:00:00 AM	4/1/2021 12:00:00 AM
552	Wednesday, April 15, 2020	Friday, April 17, 202	0	2	26	4/20/2020 12:00:00 AM	4/1/2020 12:00:00 AM
564	Saturday, December 7, 2019	Monday, December 9, 2019	0	2	26	12/12/2019 12:00:00 AM	12/1/2019 12:00:00 AM
570	Friday, October 1, 2021	Friday, October 8, 2021	0	7	26	10/6/2021 12:00:00 AM	10/1/2021 12:00:00 AM
820	Tuesday, June 28, 2022	Saturday, July 2, 2022	0	4	26	7/3/2022 12:00:00 AM	6/1/2022 12:00:00 AM
970	Friday, April 9, 2021	Sunday, April 11, 2021	0	2	26	4/14/2021 12:00:00 AM	4/1/2021 12:00:00 AM
1155	Tuesday, December 20, 2022	Wednesday, December 21, 2022	0	1	26	12/25/2022 12:00:00 AM	12/1/2022 12:00:00 AM
1169	Wednesday, December 8, 2021	Friday, December 10, 2021	0	2	26	12/13/2021 12:00:00 AM	12/1/2021 12:00:00 AM

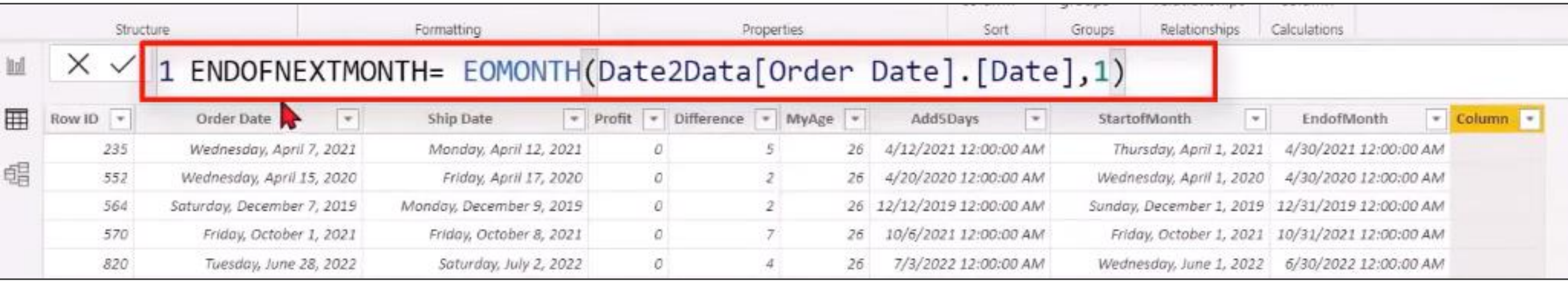
ENDOFMONTH()



```
EndofMonth = ENDOFMONTH(Date2Data[Order Date].[Date])
```

Row ID	Order Date	Ship Date	Profit	Difference	MyAge	Add5Days	StartofMonth	EndofMonth
235	Wednesday, April 7, 2021	Monday, April 12, 2021	0	5	26	4/12/2021 12:00:00 AM	Thursday, April 1, 2021	4/30/2021 12:00:00 AM
552	Wednesday, April 15, 2020	Friday, April 17, 2020	0	2	26	4/20/2020 12:00:00 AM	Wednesday, April 1, 2020	4/30/2020 12:00:00 AM
564	Saturday, December 7, 2019	Monday, December 9, 2019	0	2	26	12/12/2019 12:00:00 AM	Sunday, December 1, 2019	12/31/2019 12:00:00 AM

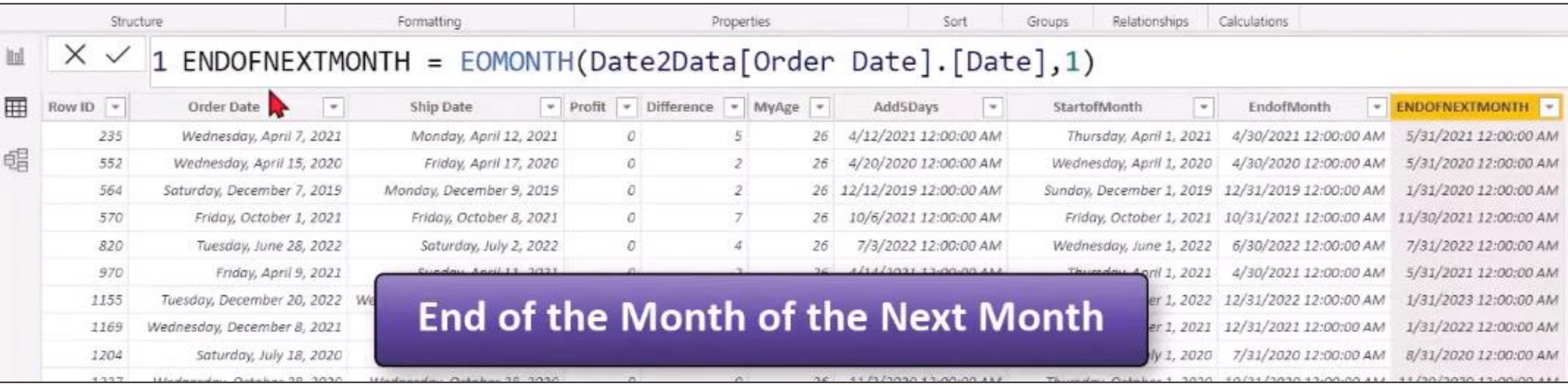
End of Next Month – EOMONTH()



Structure | Formatting | Properties | Sort | Groups | Relationships | Calculations

```
1 ENDOFNEXTMONTH= EOMONTH(Date2Data[Order Date].[Date],1)
```

Row ID	Order Date	Ship Date	Profit	Difference	MyAge	Add5Days	StartofMonth	EndofMonth	Column
235	Wednesday, April 7, 2021	Monday, April 12, 2021	0	5	26	4/12/2021 12:00:00 AM	Thursday, April 1, 2021	4/30/2021 12:00:00 AM	
552	Wednesday, April 15, 2020	Friday, April 17, 2020	0	2	26	4/20/2020 12:00:00 AM	Wednesday, April 1, 2020	4/30/2020 12:00:00 AM	
564	Saturday, December 7, 2019	Monday, December 9, 2019	0	2	26	12/12/2019 12:00:00 AM	Sunday, December 1, 2019	12/31/2019 12:00:00 AM	
570	Friday, October 1, 2021	Friday, October 8, 2021	0	7	26	10/6/2021 12:00:00 AM	Friday, October 1, 2021	10/31/2021 12:00:00 AM	
820	Tuesday, June 28, 2022	Saturday, July 2, 2022	0	4	26	7/3/2022 12:00:00 AM	Wednesday, June 1, 2022	6/30/2022 12:00:00 AM	



Structure | Formatting | Properties | Sort | Groups | Relationships | Calculations

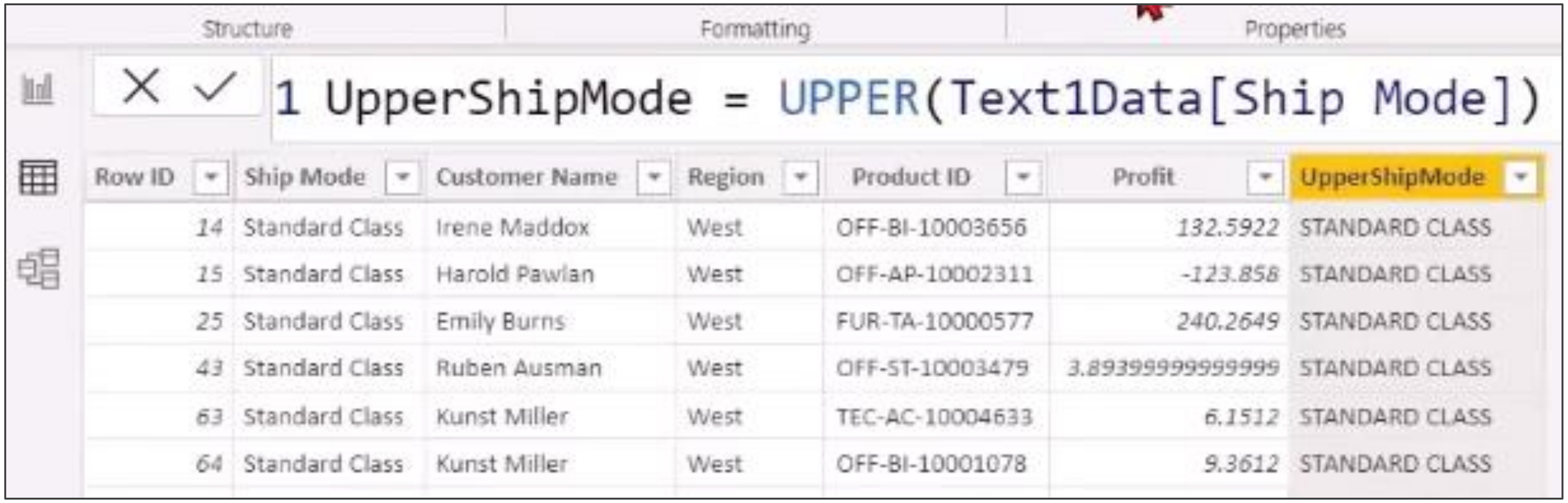
```
1 ENDOFNEXTMONTH = EOMONTH(Date2Data[Order Date].[Date],1)
```

Row ID	Order Date	Ship Date	Profit	Difference	MyAge	Add5Days	StartofMonth	EndofMonth	ENDOFNEXTMONTH
235	Wednesday, April 7, 2021	Monday, April 12, 2021	0	5	26	4/12/2021 12:00:00 AM	Thursday, April 1, 2021	4/30/2021 12:00:00 AM	5/31/2021 12:00:00 AM
552	Wednesday, April 15, 2020	Friday, April 17, 2020	0	2	26	4/20/2020 12:00:00 AM	Wednesday, April 1, 2020	4/30/2020 12:00:00 AM	5/31/2020 12:00:00 AM
564	Saturday, December 7, 2019	Monday, December 9, 2019	0	2	26	12/12/2019 12:00:00 AM	Sunday, December 1, 2019	12/31/2019 12:00:00 AM	1/31/2020 12:00:00 AM
570	Friday, October 1, 2021	Friday, October 8, 2021	0	7	26	10/6/2021 12:00:00 AM	Friday, October 1, 2021	10/31/2021 12:00:00 AM	11/30/2021 12:00:00 AM
820	Tuesday, June 28, 2022	Saturday, July 2, 2022	0	4	26	7/3/2022 12:00:00 AM	Wednesday, June 1, 2022	6/30/2022 12:00:00 AM	7/31/2022 12:00:00 AM
970	Friday, April 9, 2021	Sunday, April 11, 2021	0	2	26	4/14/2021 12:00:00 AM	Thursday, April 1, 2021	4/30/2021 12:00:00 AM	5/31/2021 12:00:00 AM
1155	Tuesday, December 20, 2022	We	[illegible]	[illegible]	[illegible]	[illegible]	er 1, 2022	12/31/2022 12:00:00 AM	1/31/2023 12:00:00 AM
1169	Wednesday, December 8, 2021	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	er 1, 2021	12/31/2021 12:00:00 AM	1/31/2022 12:00:00 AM
1204	Saturday, July 18, 2020	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ly 1, 2020	7/31/2020 12:00:00 AM	8/31/2020 12:00:00 AM

End of the Month of the Next Month

TEXT Functions

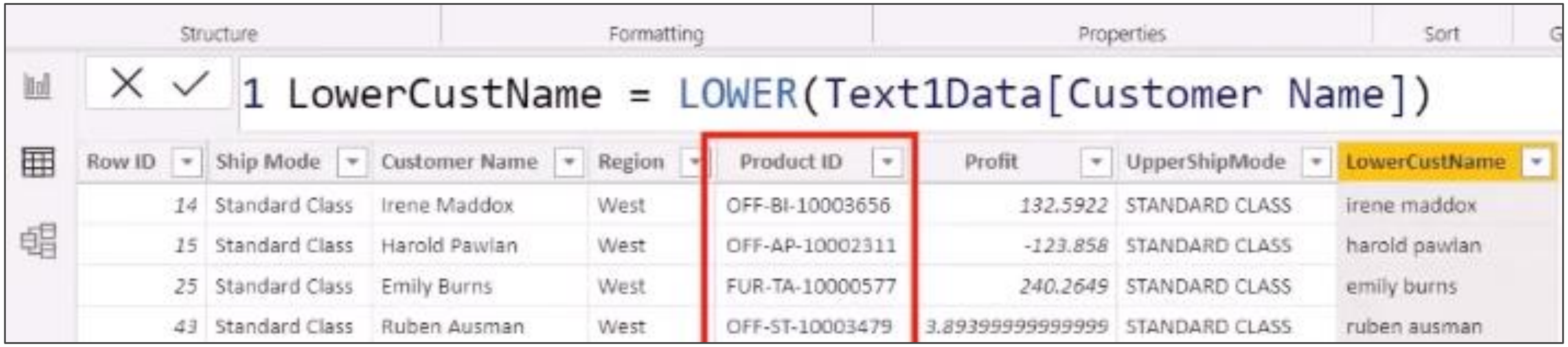
UPPER() Function



```
1 UpperShipMode = UPPER(Text1Data[Ship Mode])
```

Row ID	Ship Mode	Customer Name	Region	Product ID	Profit	UpperShipMode
14	Standard Class	Irene Maddox	West	OFF-BI-10003656	132.5922	STANDARD CLASS
15	Standard Class	Harold Pawlan	West	OFF-AP-10002311	-123.858	STANDARD CLASS
25	Standard Class	Emily Burns	West	FUR-TA-10000577	240.2649	STANDARD CLASS
43	Standard Class	Ruben Ausman	West	OFF-ST-10003479	3.89399999999999	STANDARD CLASS
63	Standard Class	Kunst Miller	West	TEC-AC-10004633	6.1512	STANDARD CLASS
64	Standard Class	Kunst Miller	West	OFF-BI-10001078	9.3612	STANDARD CLASS

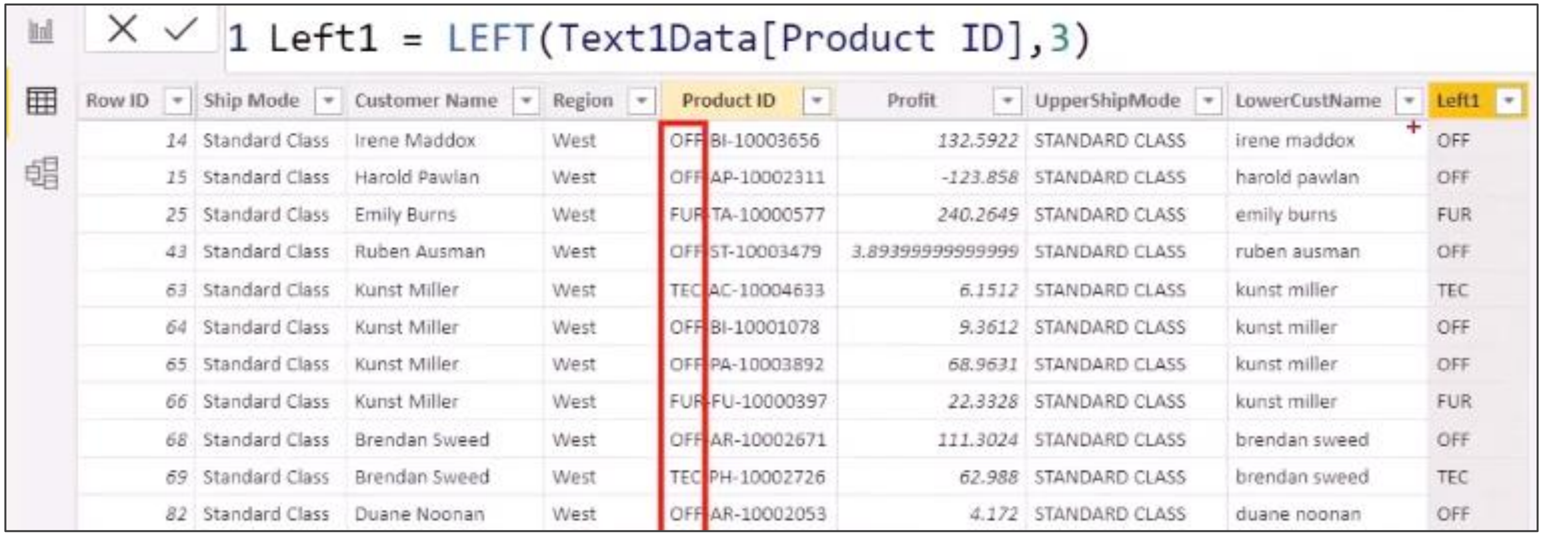
LOWER()



```
LowerCustName = LOWER(Text1Data[Customer Name])
```

Row ID	Ship Mode	Customer Name	Region	Product ID	Profit	UpperShipMode	LowerCustName
14	Standard Class	Irene Maddox	West	OFF-BI-10003656	132.5922	STANDARD CLASS	irene maddox
15	Standard Class	Harold Pawlan	West	OFF-AP-10002311	-123.858	STANDARD CLASS	harold pawlan
25	Standard Class	Emily Burns	West	FUR-TA-10000577	240.2649	STANDARD CLASS	emily burns
43	Standard Class	Ruben Ausman	West	OFF-ST-10003479	3.89399999999999	STANDARD CLASS	ruben ausman

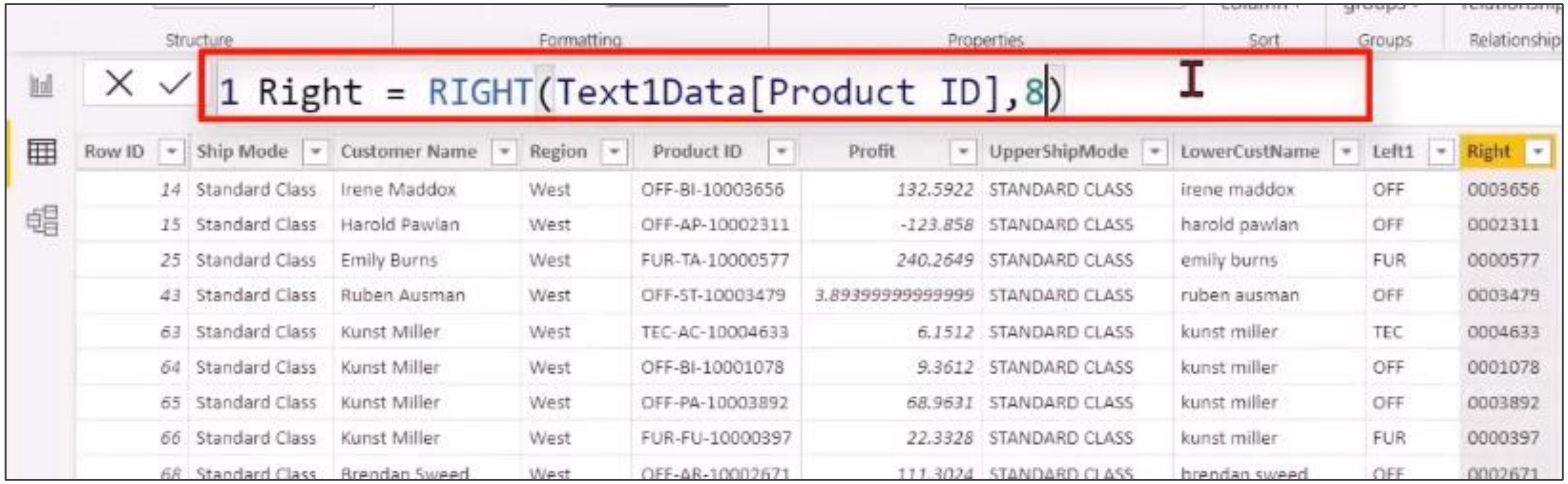
LEFT()



```
1 Left1 = LEFT(Text1Data[Product ID],3)
```

Row ID	Ship Mode	Customer Name	Region	Product ID	Profit	UpperShipMode	LowerCustName	Left1
14	Standard Class	Irene Maddox	West	OFF-BI-10003656	132.5922	STANDARD CLASS	irene maddox	OFF
15	Standard Class	Harold Pawlan	West	OFF-AP-10002311	-123.858	STANDARD CLASS	harold pawlan	OFF
25	Standard Class	Emily Burns	West	FUR-TA-10000577	240.2649	STANDARD CLASS	emily burns	FUR
43	Standard Class	Ruben Ausman	West	OFF-ST-10003479	3.89399999999999	STANDARD CLASS	ruben ausman	OFF
63	Standard Class	Kunst Miller	West	TEC-AC-10004633	6.1512	STANDARD CLASS	kunst miller	TEC
64	Standard Class	Kunst Miller	West	OFF-BI-10001078	9.3612	STANDARD CLASS	kunst miller	OFF
65	Standard Class	Kunst Miller	West	OFF-PA-10003892	68.9631	STANDARD CLASS	kunst miller	OFF
66	Standard Class	Kunst Miller	West	FUR-FU-10000397	22.3328	STANDARD CLASS	kunst miller	FUR
68	Standard Class	Brendan Sweed	West	OFF-AR-10002671	111.3024	STANDARD CLASS	brendan sweed	OFF
69	Standard Class	Brendan Sweed	West	TEC-PH-10002726	62.988	STANDARD CLASS	brendan sweed	TEC
82	Standard Class	Duane Noonan	West	OFF-AR-10002053	4.172	STANDARD CLASS	duane noonan	OFF

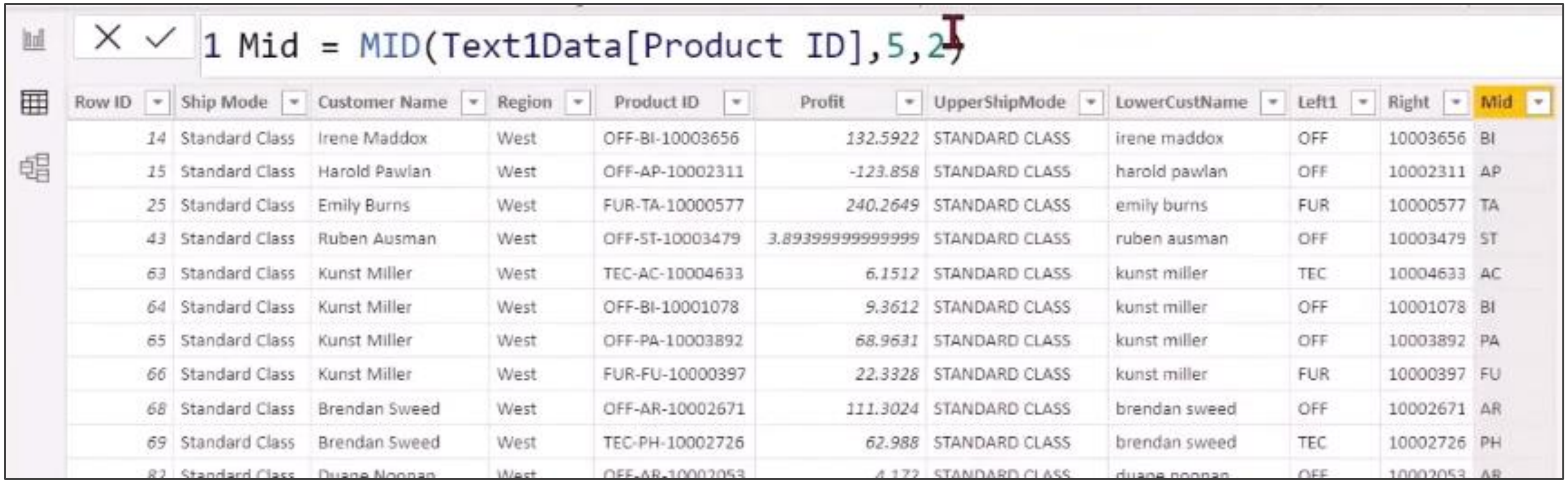
RIGHT()



```
1 Right = RIGHT(Text1Data[Product ID],8)
```

Row ID	Ship Mode	Customer Name	Region	Product ID	Profit	UpperShipMode	LowerCustName	Left1	Right
14	Standard Class	Irene Maddox	West	OFF-BI-10003656	132.5922	STANDARD CLASS	irene maddox	OFF	0003656
15	Standard Class	Harold Pawlan	West	OFF-AP-10002311	-123.858	STANDARD CLASS	harold pawlan	OFF	0002311
25	Standard Class	Emily Burns	West	FUR-TA-10000577	240.2649	STANDARD CLASS	emily burns	FUR	0000577
43	Standard Class	Ruben Ausman	West	OFF-ST-10003479	3.89399999999999	STANDARD CLASS	ruben ausman	OFF	0003479
63	Standard Class	Kunst Miller	West	TEC-AC-10004633	6.1512	STANDARD CLASS	kunst miller	TEC	0004633
64	Standard Class	Kunst Miller	West	OFF-BI-10001078	9.3612	STANDARD CLASS	kunst miller	OFF	0001078
65	Standard Class	Kunst Miller	West	OFF-PA-10003892	68.9631	STANDARD CLASS	kunst miller	OFF	0003892
66	Standard Class	Kunst Miller	West	FUR-FU-10000397	22.3328	STANDARD CLASS	kunst miller	FUR	0000397
68	Standard Class	Brendan Sweed	West	OFF-AR-10002671	111.3024	STANDARD CLASS	brendan sweed	OFF	0002671

MID()



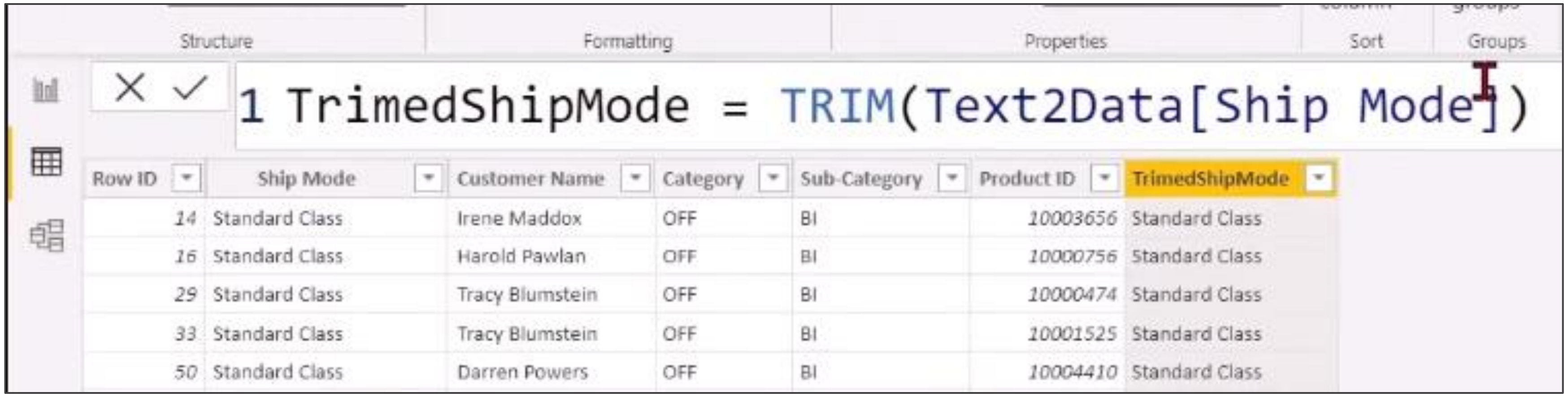
```
1 Mid = MID(Text1Data[Product ID],5,2)
```

Row ID	Ship Mode	Customer Name	Region	Product ID	Profit	UpperShipMode	LowerCustName	Left1	Right	Mid
14	Standard Class	Irene Maddox	West	OFF-BI-10003656	132.5922	STANDARD CLASS	irene maddox	OFF	10003656	BI
15	Standard Class	Harold Pawlan	West	OFF-AP-10002311	-123.858	STANDARD CLASS	harold pawlan	OFF	10002311	AP
25	Standard Class	Emily Burns	West	FUR-TA-10000577	240.2649	STANDARD CLASS	emily burns	FUR	10000577	TA
43	Standard Class	Ruben Ausman	West	OFF-ST-10003479	3.89399999999999	STANDARD CLASS	ruben ausman	OFF	10003479	ST
63	Standard Class	Kunst Miller	West	TEC-AC-10004633	6.1512	STANDARD CLASS	kunst miller	TEC	10004633	AC
64	Standard Class	Kunst Miller	West	OFF-BI-10001078	9.3612	STANDARD CLASS	kunst miller	OFF	10001078	BI
65	Standard Class	Kunst Miller	West	OFF-PA-10003892	68.9631	STANDARD CLASS	kunst miller	OFF	10003892	PA
66	Standard Class	Kunst Miller	West	FUR-FU-10000397	22.3328	STANDARD CLASS	kunst miller	FUR	10000397	FU
68	Standard Class	Brendan Sweed	West	OFF-AR-10002671	111.3024	STANDARD CLASS	brendan sweed	OFF	10002671	AR
69	Standard Class	Brendan Sweed	West	TEC-PH-10002726	62.988	STANDARD CLASS	brendan sweed	TEC	10002726	PH
82	Standard Class	Duane Noonan	West	OFF-AR-10002053	4.172	STANDARD CLASS	duane noonan	OFF	10002053	AR

Power Query – M Lang Numbering starts from 0 **Index - 0**

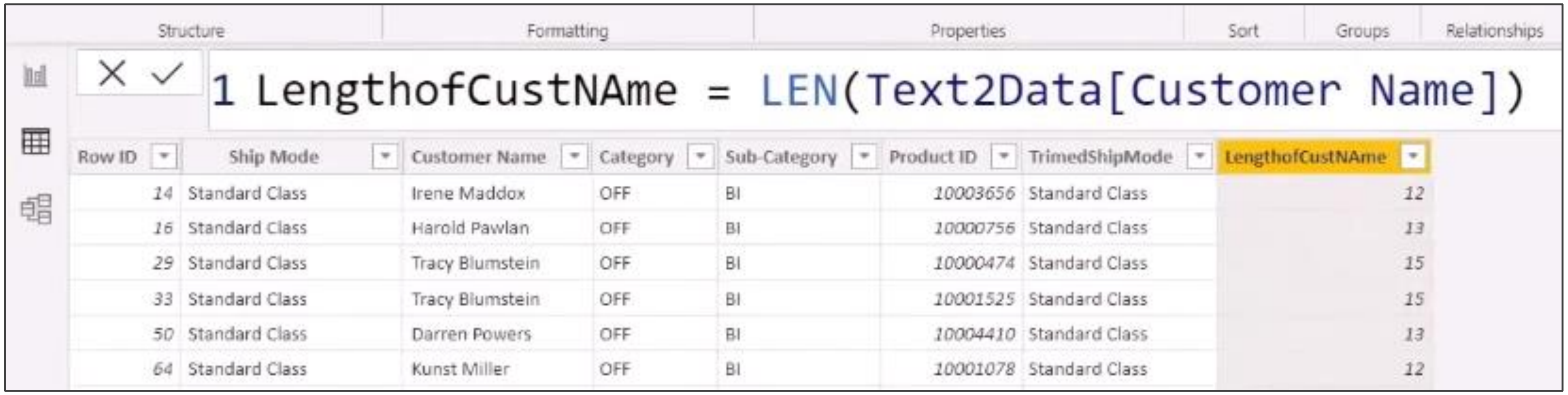
Power Pivot – DAX Numbering starts from 1 **Index - 1**

TRIM() & LENGTH()



```
1 TrimedShipMode = TRIM(Text2Data[Ship Mode])
```

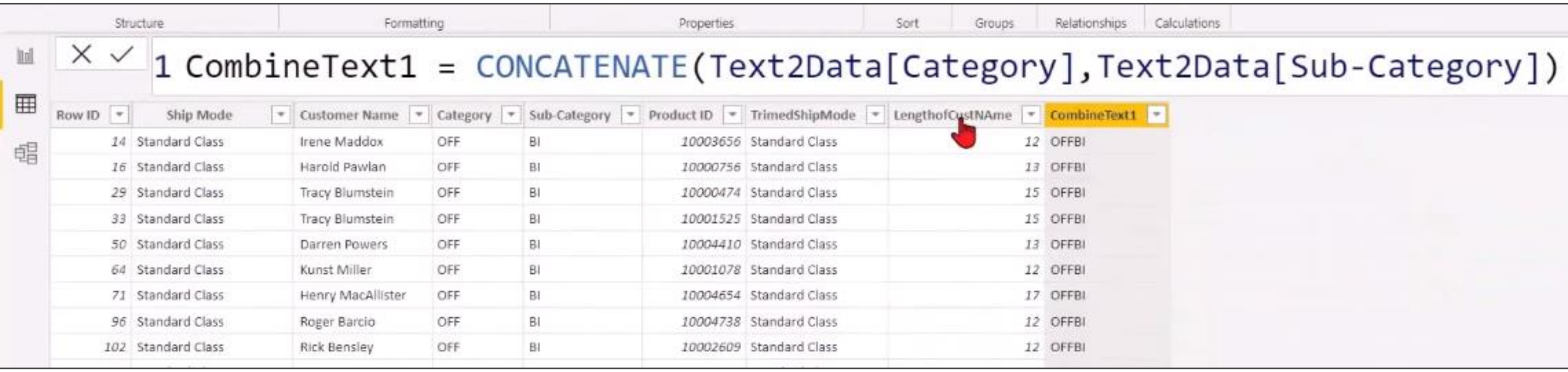
Row ID	Ship Mode	Customer Name	Category	Sub-Category	Product ID	TrimedShipMode
14	Standard Class	Irene Maddox	OFF	BI	10003656	Standard Class
16	Standard Class	Harold Pawlan	OFF	BI	10000756	Standard Class
29	Standard Class	Tracy Blumstein	OFF	BI	10000474	Standard Class
33	Standard Class	Tracy Blumstein	OFF	BI	10001525	Standard Class
50	Standard Class	Darren Powers	OFF	BI	10004410	Standard Class



```
1 LengthofCustNAme = LEN(Text2Data[Customer Name])
```

Row ID	Ship Mode	Customer Name	Category	Sub-Category	Product ID	TrimedShipMode	LengthofCustNAme
14	Standard Class	Irene Maddox	OFF	BI	10003656	Standard Class	12
16	Standard Class	Harold Pawlan	OFF	BI	10000756	Standard Class	13
29	Standard Class	Tracy Blumstein	OFF	BI	10000474	Standard Class	15
33	Standard Class	Tracy Blumstein	OFF	BI	10001525	Standard Class	15
50	Standard Class	Darren Powers	OFF	BI	10004410	Standard Class	13
64	Standard Class	Kunst Miller	OFF	BI	10001078	Standard Class	12

CONCATENATE()

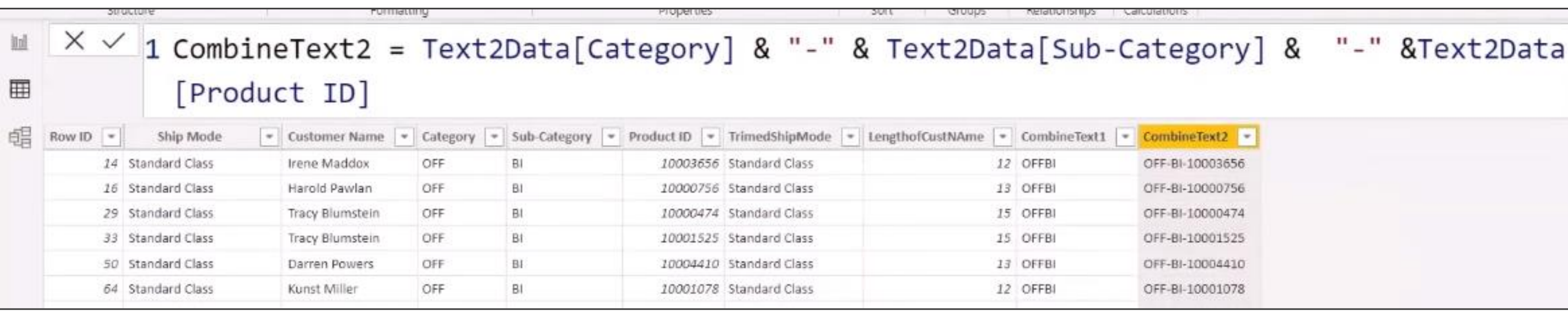


```
1 CombineText1 = CONCATENATE(Text2Data[Category],Text2Data[Sub-Category])
```

Row ID	Ship Mode	Customer Name	Category	Sub-Category	Product ID	TrimedShipMode	LengthofCustNAme	CombineText1
14	Standard Class	Irene Maddox	OFF	BI	10003656	Standard Class	12	OFFBI
16	Standard Class	Harold Pawlan	OFF	BI	10000756	Standard Class	13	OFFBI
29	Standard Class	Tracy Blumstein	OFF	BI	10000474	Standard Class	15	OFFBI
33	Standard Class	Tracy Blumstein	OFF	BI	10001525	Standard Class	15	OFFBI
50	Standard Class	Darren Powers	OFF	BI	10004410	Standard Class	13	OFFBI
64	Standard Class	Kunst Miller	OFF	BI	10001078	Standard Class	12	OFFBI
71	Standard Class	Henry MacAllister	OFF	BI	10004654	Standard Class	17	OFFBI
96	Standard Class	Roger Barcio	OFF	BI	10004738	Standard Class	12	OFFBI
102	Standard Class	Rick Bensley	OFF	BI	10002609	Standard Class	12	OFFBI

CONCATENATE() can take only TWO PARAMETERS

Using & to concatenate

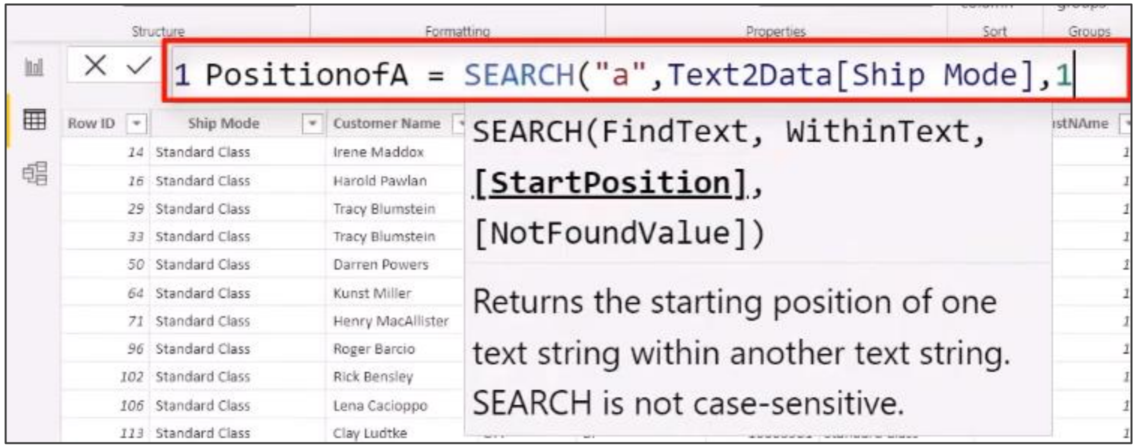


```
1 CombineText2 = Text2Data[Category] & "-" & Text2Data[Sub-Category] &  "-" &Text2Data
  [Product ID]
```

Row ID	Ship Mode	Customer Name	Category	Sub-Category	Product ID	TrimedShipMode	LengthofCustNAme	CombineText1	CombineText2
14	Standard Class	Irene Maddox	OFF	BI	10003656	Standard Class	12	OFFBI	OFF-BI-10003656
16	Standard Class	Harold Pawlan	OFF	BI	10000756	Standard Class	13	OFFBI	OFF-BI-10000756
29	Standard Class	Tracy Blumstein	OFF	BI	10000474	Standard Class	15	OFFBI	OFF-BI-10000474
33	Standard Class	Tracy Blumstein	OFF	BI	10001525	Standard Class	15	OFFBI	OFF-BI-10001525
50	Standard Class	Darren Powers	OFF	BI	10004410	Standard Class	13	OFFBI	OFF-BI-10004410
64	Standard Class	Kunst Miller	OFF	BI	10001078	Standard Class	12	OFFBI	OFF-BI-10001078

SUBSTITUTE() Function
SEARCH() Function

SEARCH()



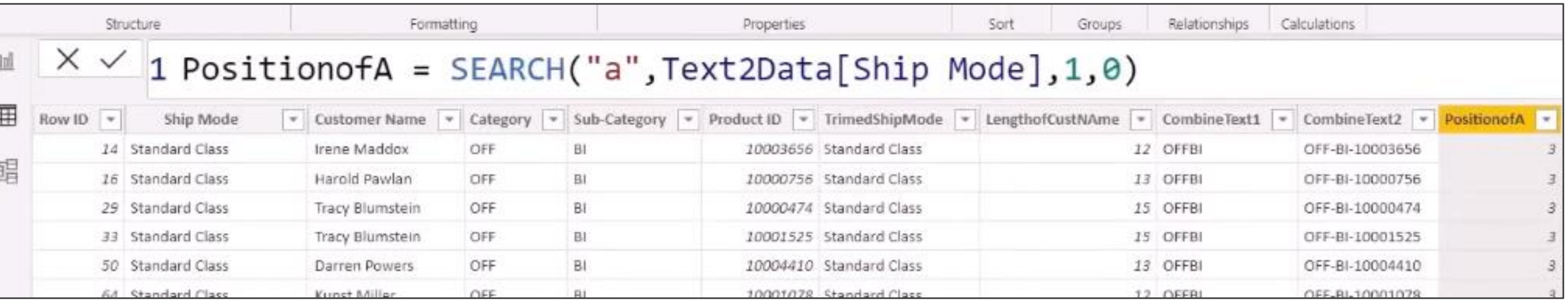
Structure | Formatting | Properties | Sort | Groups

```
1 PositionofA = SEARCH("a",Text2Data[Ship Mode],1
```

SEARCH(FindText, WithinText, **[StartPosition]**, [NotFoundValue])

Returns the starting position of one text string within another text string. SEARCH is not case-sensitive.

Row ID	Ship Mode	Customer Name
14	Standard Class	Irene Maddox
16	Standard Class	Harold Pawlan
29	Standard Class	Tracy Blumstein
33	Standard Class	Tracy Blumstein
50	Standard Class	Darren Powers
64	Standard Class	Kunst Miller
71	Standard Class	Henry MacAllister
96	Standard Class	Roger Barcio
102	Standard Class	Rick Bensley
106	Standard Class	Lena Cacioppo
113	Standard Class	Clay Ludtke



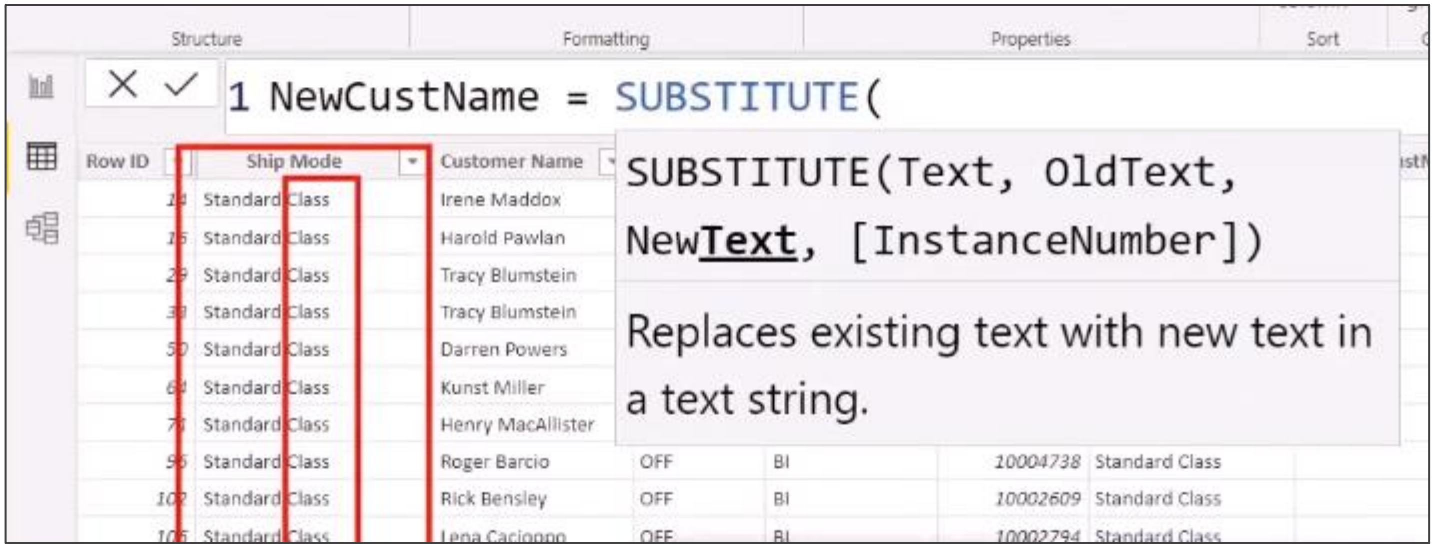
Structure | Formatting | Properties | Sort | Groups | Relationships | Calculations

```
1 PositionofA = SEARCH("a",Text2Data[Ship Mode],1,0)
```

Row ID	Ship Mode	Customer Name	Category	Sub-Category	Product ID	TrimedShipMode	LengthofCustNAme	CombineText1	CombineText2	PositionofA
14	Standard Class	Irene Maddox	OFF	BI	10003656	Standard Class	12	OFFBI	OFF-BI-10003656	3
16	Standard Class	Harold Pawlan	OFF	BI	10000756	Standard Class	13	OFFBI	OFF-BI-10000756	3
29	Standard Class	Tracy Blumstein	OFF	BI	10000474	Standard Class	15	OFFBI	OFF-BI-10000474	3
33	Standard Class	Tracy Blumstein	OFF	BI	10001525	Standard Class	15	OFFBI	OFF-BI-10001525	3
50	Standard Class	Darren Powers	OFF	BI	10004410	Standard Class	13	OFFBI	OFF-BI-10004410	3
64	Standard Class	Kunst Miller	OFF	BI	10001078	Standard Class	12	OFFBI	OFF-BI-10001078	3

SUBSTITUTE()

SUBSTITUTE() Function **Text** **Character**



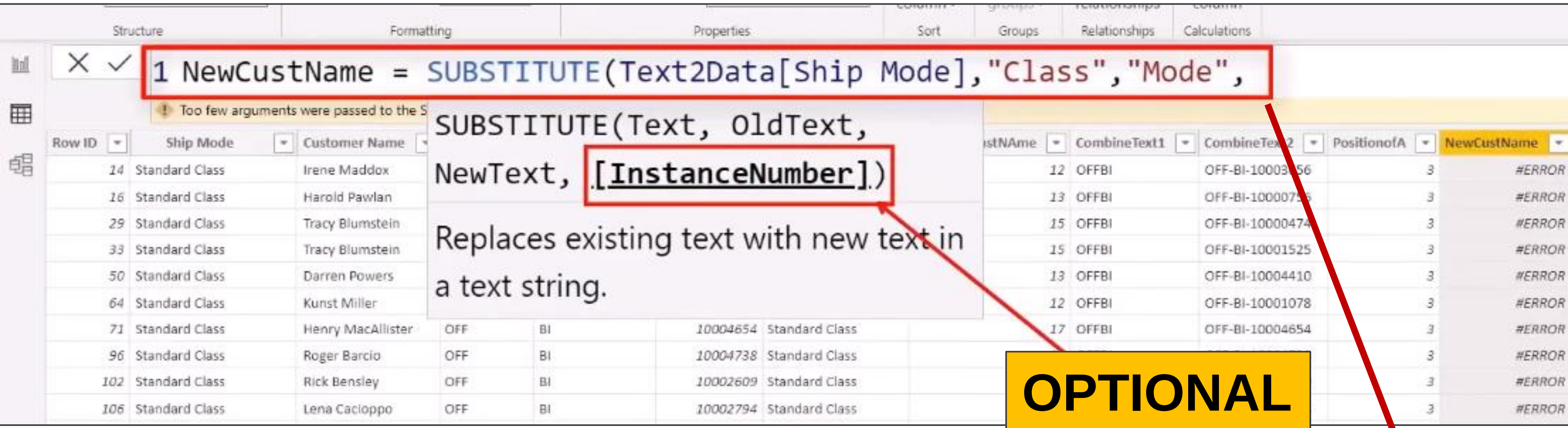
Structure Formatting Properties Sort

```
1 NewCustName = SUBSTITUTE(
```

SUBSTITUTE(Text, OldText, NewText, [InstanceNumber])

Replaces existing text with new text in a text string.

Row ID	Ship Mode	Customer Name				
14	Standard Class	Irene Maddox				
15	Standard Class	Harold Pawlan				
29	Standard Class	Tracy Blumstein				
30	Standard Class	Tracy Blumstein				
50	Standard Class	Darren Powers				
64	Standard Class	Kunst Miller				
71	Standard Class	Henry MacAllister				
96	Standard Class	Roger Barcio	OFF	BI	10004738	Standard Class
102	Standard Class	Rick Bensley	OFF	BI	10002609	Standard Class
105	Standard Class	Lena Cacioppo	OFF	BI	10002794	Standard Class



```
1 NewCustName = SUBSTITUTE(Text2Data[Ship Mode],"Class","Mode",
```

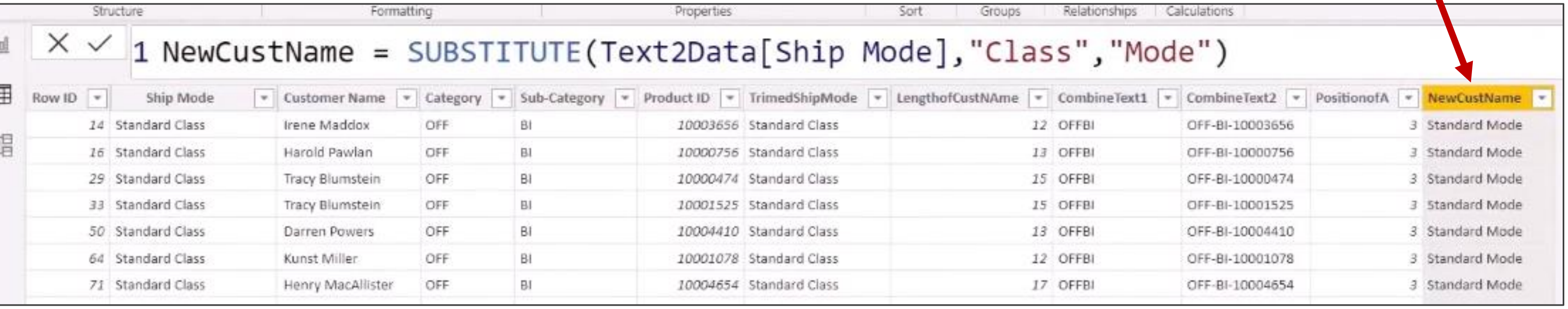
⚠ Too few arguments were passed to the S

```
SUBSTITUTE(Text, OldText,
NewText, [InstanceNumber])
```

Replaces existing text with new text in a text string.

Row ID	Ship Mode	Customer Name					stNAme	CombineText1	CombineTex 2	PositionofA	NewCustName
14	Standard Class	Irene Maddox					12	OFFBI	OFF-BI-10003656	3	#ERROR
16	Standard Class	Harold Pawlan					13	OFFBI	OFF-BI-10000756	3	#ERROR
29	Standard Class	Tracy Blumstein					15	OFFBI	OFF-BI-1000047	3	#ERROR
33	Standard Class	Tracy Blumstein					15	OFFBI	OFF-BI-10001525	3	#ERROR
50	Standard Class	Darren Powers					13	OFFBI	OFF-BI-10004410	3	#ERROR
64	Standard Class	Kunst Miller					12	OFFBI	OFF-BI-10001078	3	#ERROR
71	Standard Class	Henry MacAllister	OFF	BI	10004654	Standard Class	17	OFFBI	OFF-BI-10004654	3	#ERROR
96	Standard Class	Roger Barcio	OFF	BI	10004738	Standard Class				3	#ERROR
102	Standard Class	Rick Bensley	OFF	BI	10002609	Standard Class				3	#ERROR
106	Standard Class	Lena Cacioppo	OFF	BI	10002794	Standard Class				3	#ERROR

OPTIONAL

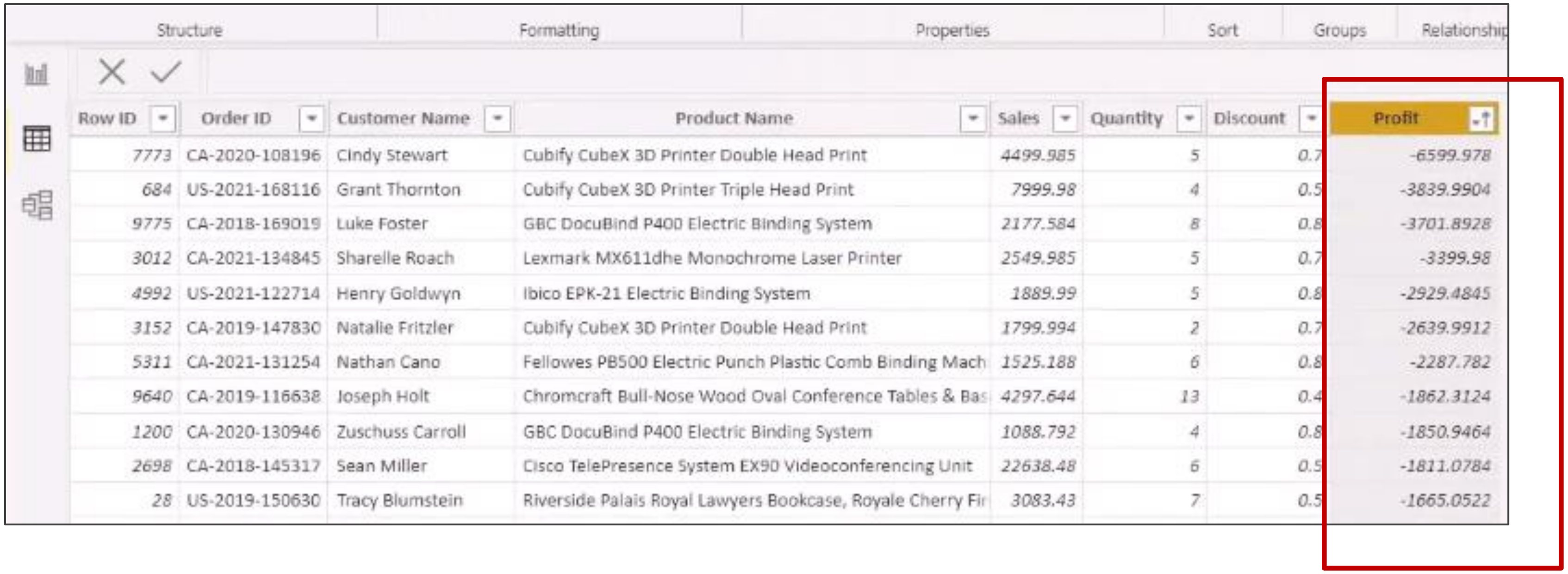


```
1 NewCustName = SUBSTITUTE(Text2Data[Ship Mode],"Class","Mode")
```

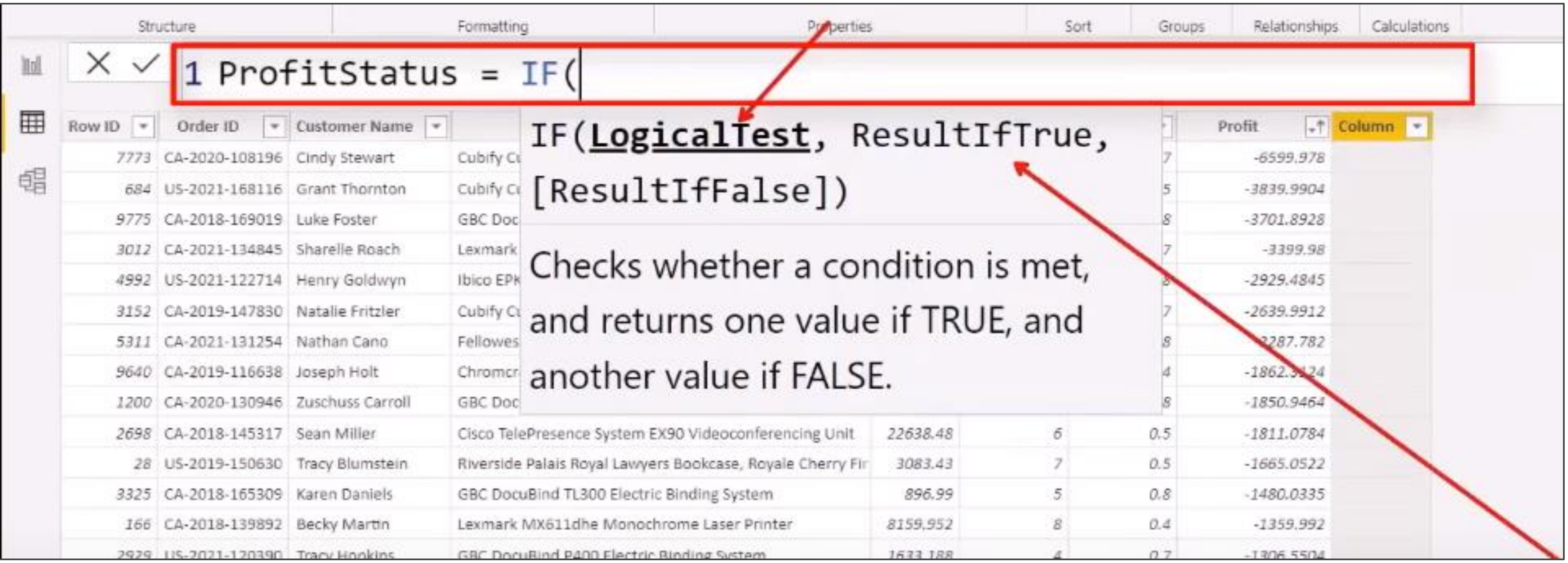
Row ID	Ship Mode	Customer Name	Category	Sub-Category	Product ID	TrimedShipMode	LengthofCustNAme	CombineText1	CombineText2	PositionofA	NewCustName
14	Standard Class	Irene Maddox	OFF	BI	10003656	Standard Class	12	OFFBI	OFF-BI-10003656	3	Standard Mode
16	Standard Class	Harold Pawlan	OFF	BI	10000756	Standard Class	13	OFFBI	OFF-BI-10000756	3	Standard Mode
29	Standard Class	Tracy Blumstein	OFF	BI	10000474	Standard Class	15	OFFBI	OFF-BI-10000474	3	Standard Mode
33	Standard Class	Tracy Blumstein	OFF	BI	10001525	Standard Class	15	OFFBI	OFF-BI-10001525	3	Standard Mode
50	Standard Class	Darren Powers	OFF	BI	10004410	Standard Class	13	OFFBI	OFF-BI-10004410	3	Standard Mode
64	Standard Class	Kunst Miller	OFF	BI	10001078	Standard Class	12	OFFBI	OFF-BI-10001078	3	Standard Mode
71	Standard Class	Henry MacAllister	OFF	BI	10004654	Standard Class	17	OFFBI	OFF-BI-10004654	3	Standard Mode

Logical Functions

IF Function | **IFERROR Function**



Row ID	Order ID	Customer Name	Product Name	Sales	Quantity	Discount	Profit
7773	CA-2020-108196	Cindy Stewart	Cubify CubeX 3D Printer Double Head Print	4499.985	5	0.7	-6599.978
684	US-2021-168116	Grant Thornton	Cubify CubeX 3D Printer Triple Head Print	7999.98	4	0.5	-3839.9904
9775	CA-2018-169019	Luke Foster	GBC DocuBind P400 Electric Binding System	2177.584	8	0.8	-3701.8928
3012	CA-2021-134845	Sharelle Roach	Lexmark MX611dhe Monochrome Laser Printer	2549.985	5	0.7	-3399.98
4992	US-2021-122714	Henry Goldwyn	Ibico EPK-21 Electric Binding System	1889.99	5	0.8	-2929.4845
3152	CA-2019-147830	Natalie Fritzler	Cubify CubeX 3D Printer Double Head Print	1799.994	2	0.7	-2639.9912
5311	CA-2021-131254	Nathan Cano	Fellowes PB500 Electric Punch Plastic Comb Binding Mach	1525.188	6	0.8	-2287.782
9640	CA-2019-116638	Joseph Holt	Chromcraft Bull-Nose Wood Oval Conference Tables & Bas	4297.644	13	0.4	-1862.3124
1200	CA-2020-130946	Zuschuss Carroll	GBC DocuBind P400 Electric Binding System	1088.792	4	0.8	-1850.9464
2698	CA-2018-145317	Sean Miller	Cisco TelePresence System EX90 Videoconferencing Unit	22638.48	6	0.5	-1811.0784
28	US-2019-150630	Tracy Blumstein	Riverside Palais Royal Lawyers Bookcase, Royale Cherry Fir	3083.43	7	0.5	-1665.0522

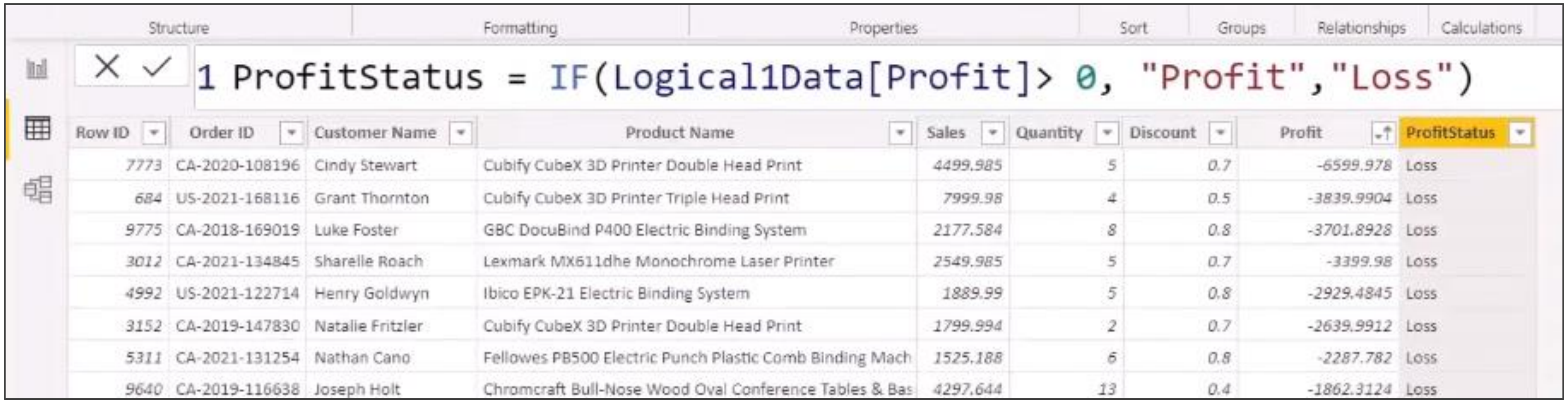


```
ProfitStatus = IF(
```

```
IF(LogicalTest, ResultIfTrue,
[ResultIfFalse])
```

Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE.

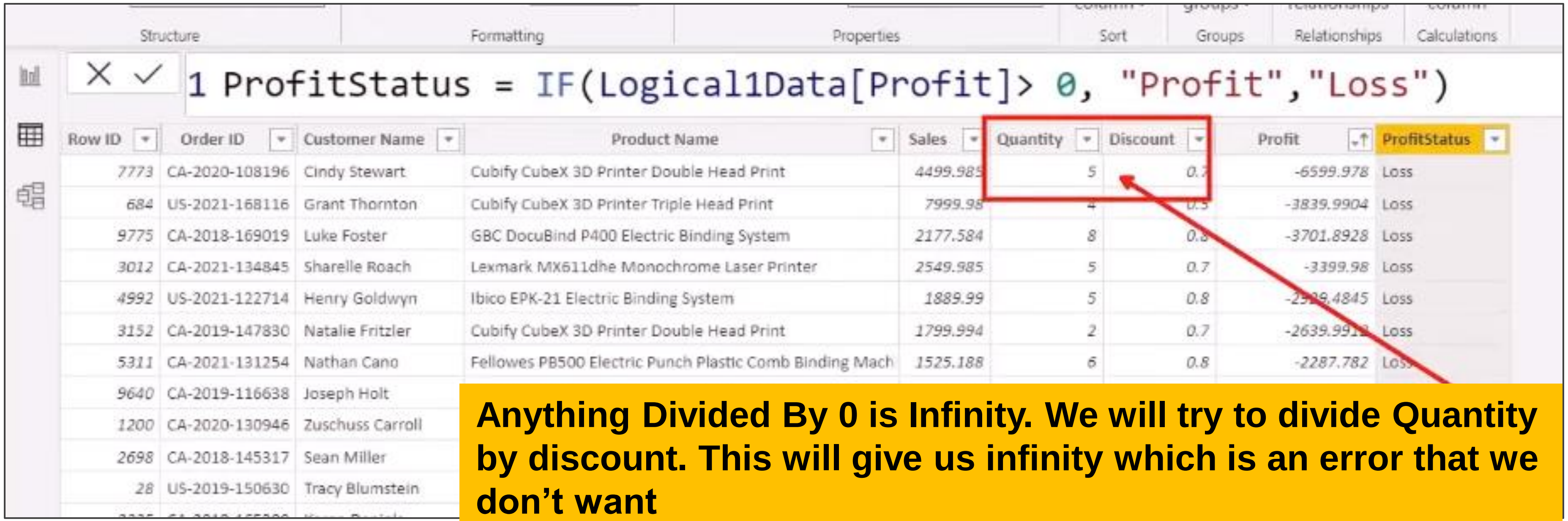
Row ID	Order ID	Customer Name					Profit	Column
7773	CA-2020-108196	Cindy Stewart	Cubify Cu		7		-6599.978	
684	US-2021-168116	Grant Thornton	Cubify Cu		5		-3839.9904	
9775	CA-2018-169019	Luke Foster	GBC Doc		8		-3701.8928	
3012	CA-2021-134845	Sharelle Roach	Lexmark		7		-3399.98	
4992	US-2021-122714	Henry Goldwyn	Ibico EPK		[illegible]		-2929.4845	
3152	CA-2019-147830	Natalie Fritzler	Cubify Cu		7		-2639.9912	
5311	CA-2021-131254	Nathan Cano	Fellowes		8		[illegible]287.782	
9640	CA-2019-116638	Joseph Holt	Chromcr		4		-1862.3124	
1200	CA-2020-130946	Zuschuss Carroll	GBC Doc		8		-1850.9464	
2698	CA-2018-145317	Sean Miller	Cisco TelePresence System EX90 Videoconferencing Unit	22638.48	6	0.5	-1811.0784	
28	US-2019-150630	Tracy Blumstein	Riverside Palais Royal Lawyers Bookcase, Royale Cherry Fir	3083.43	7	0.5	-1665.0522	
3325	CA-2018-165309	Karen Daniels	GBC DocuBind TL300 Electric Binding System	896.99	5	0.8	-1480.0335	
166	CA-2018-139892	Becky Martin	Lexmark MX611dhe Monochrome Laser Printer	8159.952	8	0.4	-1359.992	
2929	US-2021-120390	Tracy Hopkins	GBC DocuBind P400 Electric Binding System	1633.188	4	0.7	-1306.5504	



```
1 ProfitStatus = IF(Logical1Data[Profit]> 0, "Profit","Loss")
```

Row ID	Order ID	Customer Name	Product Name	Sales	Quantity	Discount	Profit	ProfitStatus
7773	CA-2020-108196	Cindy Stewart	Cubify CubeX 3D Printer Double Head Print	4499.985	5	0.7	-6599.978	Loss
684	US-2021-168116	Grant Thornton	Cubify CubeX 3D Printer Triple Head Print	7999.98	4	0.5	-3839.9904	Loss
9775	CA-2018-169019	Luke Foster	GBC DocuBind P400 Electric Binding System	2177.584	8	0.8	-3701.8928	Loss
3012	CA-2021-134845	Sharelle Roach	Lexmark MX611dhe Monochrome Laser Printer	2549.985	5	0.7	-3399.98	Loss
4992	US-2021-122714	Henry Goldwyn	Ibico EPK-21 Electric Binding System	1889.99	5	0.8	-2929.4845	Loss
3152	CA-2019-147830	Natalie Fritzler	Cubify CubeX 3D Printer Double Head Print	1799.994	2	0.7	-2639.9912	Loss
5311	CA-2021-131254	Nathan Cano	Fellowes PB500 Electric Punch Plastic Comb Binding Mach	1525.188	6	0.8	-2287.782	Loss
9640	CA-2019-116638	Joseph Holt	Chromcraft Bull-Nose Wood Oval Conference Tables & Bas	4297.644	13	0.4	-1862.3124	Loss

IFERROR()

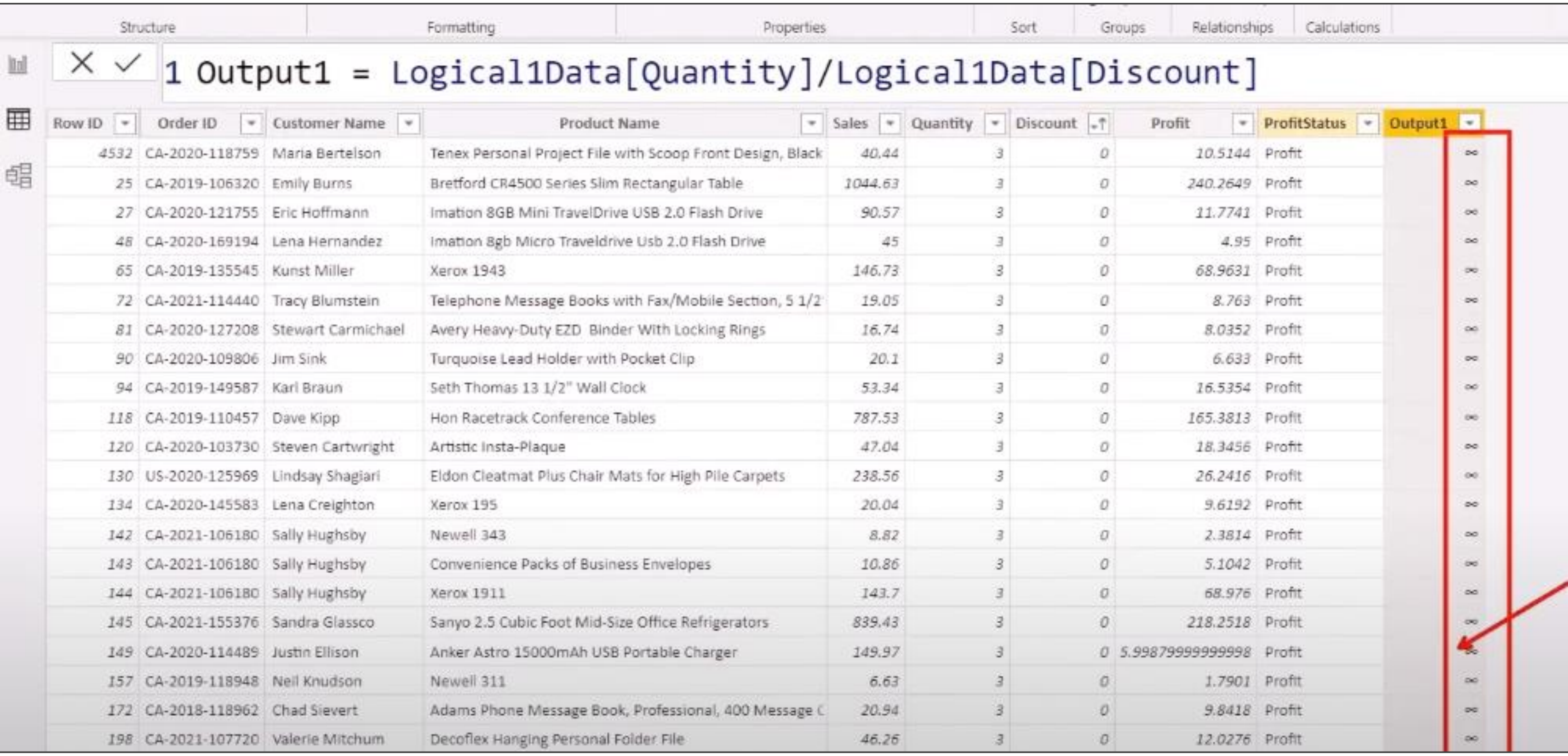


Structure | Formatting | Properties | Sort | Groups | Relationships | Calculations

```
1 ProfitStatus = IF(Logical1Data[Profit]> 0, "Profit","Loss")
```

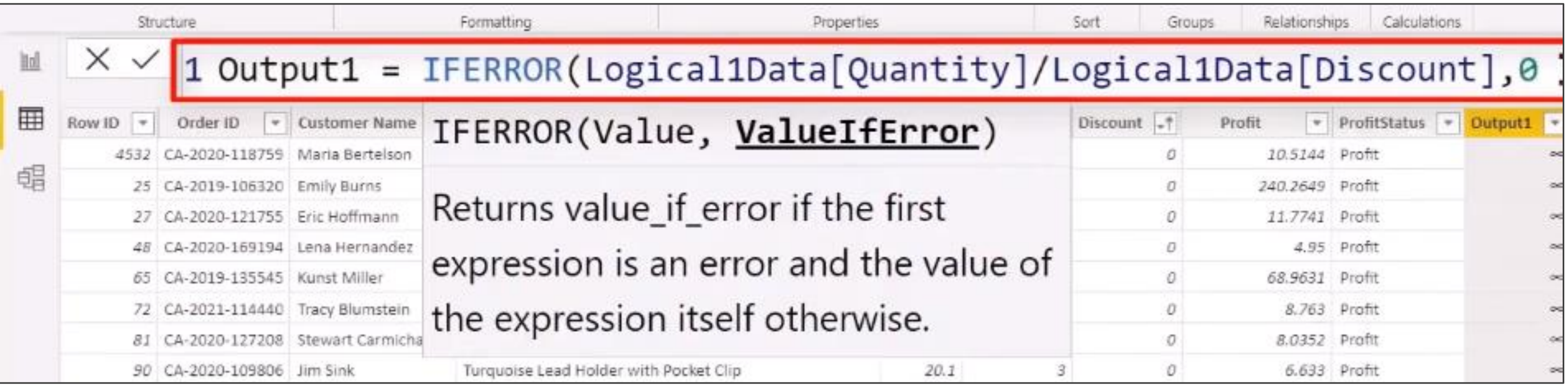
Row ID	Order ID	Customer Name	Product Name	Sales	Quantity	Discount	Profit	ProfitStatus
7773	CA-2020-108196	Cindy Stewart	Cubify CubeX 3D Printer Double Head Print	4499.985	5	0.7	-6599.978	Loss
684	US-2021-168116	Grant Thornton	Cubify CubeX 3D Printer Triple Head Print	7999.98	4	0.5	-3839.9904	Loss
9775	CA-2018-169019	Luke Foster	GBC DocuBind P400 Electric Binding System	2177.584	8	0.8	-3701.8928	Loss
3012	CA-2021-134845	Sharelle Roach	Lexmark MX611dhe Monochrome Laser Printer	2549.985	5	0.7	-3399.98	Loss
4992	US-2021-122714	Henry Goldwyn	Ibico EPK-21 Electric Binding System	1889.99	5	0.8	-2929.4845	Loss
3152	CA-2019-147830	Natalie Fritzler	Cubify CubeX 3D Printer Double Head Print	1799.994	2	0.7	-2639.9912	Loss
5311	CA-2021-131254	Nathan Cano	Fellowes PB500 Electric Punch Plastic Comb Binding Mach	1525.188	6	0.8	-2287.782	Loss
9640	CA-2019-116638	Joseph Holt						
1200	CA-2020-130946	Zuschuss Carroll						
2698	CA-2018-145317	Sean Miller						
28	US-2019-150630	Tracy Blumstein						

Anything Divided By 0 is Infinity. We will try to divide Quantity by discount. This will give us infinity which is an error that we don't want



```
1 Output1 = Logical1Data[Quantity]/Logical1Data[Discount]
```

Row ID	Order ID	Customer Name	Product Name	Sales	Quantity	Discount	Profit	ProfitStatus	Output1
4532	CA-2020-118759	Maria Bertelson	Tenex Personal Project File with Scoop Front Design, Black	40.44	3	0	10.5144	Profit	∞
25	CA-2019-106320	Emily Burns	Bretford CR4500 Series Slim Rectangular Table	1044.63	3	0	240.2649	Profit	∞
27	CA-2020-121755	Eric Hoffmann	Imation 8GB Mini TravelDrive USB 2.0 Flash Drive	90.57	3	0	11.7741	Profit	∞
48	CA-2020-169194	Lena Hernandez	Imation 8gb Micro Traveldrive Usb 2.0 Flash Drive	45	3	0	4.95	Profit	∞
65	CA-2019-135545	Kunst Miller	Xerox 1943	146.73	3	0	68.9631	Profit	∞
72	CA-2021-114440	Tracy Blumstein	Telephone Message Books with Fax/Mobile Section, 5 1/2	19.05	3	0	8.763	Profit	∞
81	CA-2020-127208	Stewart Carmichael	Avery Heavy-Duty EZD Binder With Locking Rings	16.74	3	0	8.0352	Profit	∞
90	CA-2020-109806	Jim Sink	Turquoise Lead Holder with Pocket Clip	20.1	3	0	6.633	Profit	∞
94	CA-2019-149587	Karl Braun	Seth Thomas 13 1/2" Wall Clock	53.34	3	0	16.5354	Profit	∞
118	CA-2019-110457	Dave Kipp	Hon Racetrack Conference Tables	787.53	3	0	165.3813	Profit	∞
120	CA-2020-103730	Steven Cartwright	Artistic Insta-Plaque	47.04	3	0	18.3456	Profit	∞
130	US-2020-125969	Lindsay Shagiari	Eldon Cleatmat Plus Chair Mats for High Pile Carpets	238.56	3	0	26.2416	Profit	∞
134	CA-2020-145583	Lena Creighton	Xerox 195	20.04	3	0	9.6192	Profit	∞
142	CA-2021-106180	Sally Hughsby	Newell 343	8.82	3	0	2.3814	Profit	∞
143	CA-2021-106180	Sally Hughsby	Convenience Packs of Business Envelopes	10.86	3	0	5.1042	Profit	∞
144	CA-2021-106180	Sally Hughsby	Xerox 1911	143.7	3	0	68.976	Profit	∞
145	CA-2021-155376	Sandra Glassco	Sanyo 2.5 Cubic Foot Mid-Size Office Refrigerators	839.43	3	0	218.2518	Profit	∞
149	CA-2020-114489	Justin Ellison	Anker Astro 15000mAh USB Portable Charger	149.97	3	0	5.99879999999998	Profit	∞
157	CA-2019-118948	Neil Knudson	Newell 311	6.63	3	0	1.7901	Profit	∞
172	CA-2018-118962	Chad Sievert	Adams Phone Message Book, Professional, 400 Message C	20.94	3	0	9.8418	Profit	∞
198	CA-2021-107720	Valerie Mitchum	Decoflex Hanging Personal Folder File	46.26	3	0	12.0276	Profit	∞



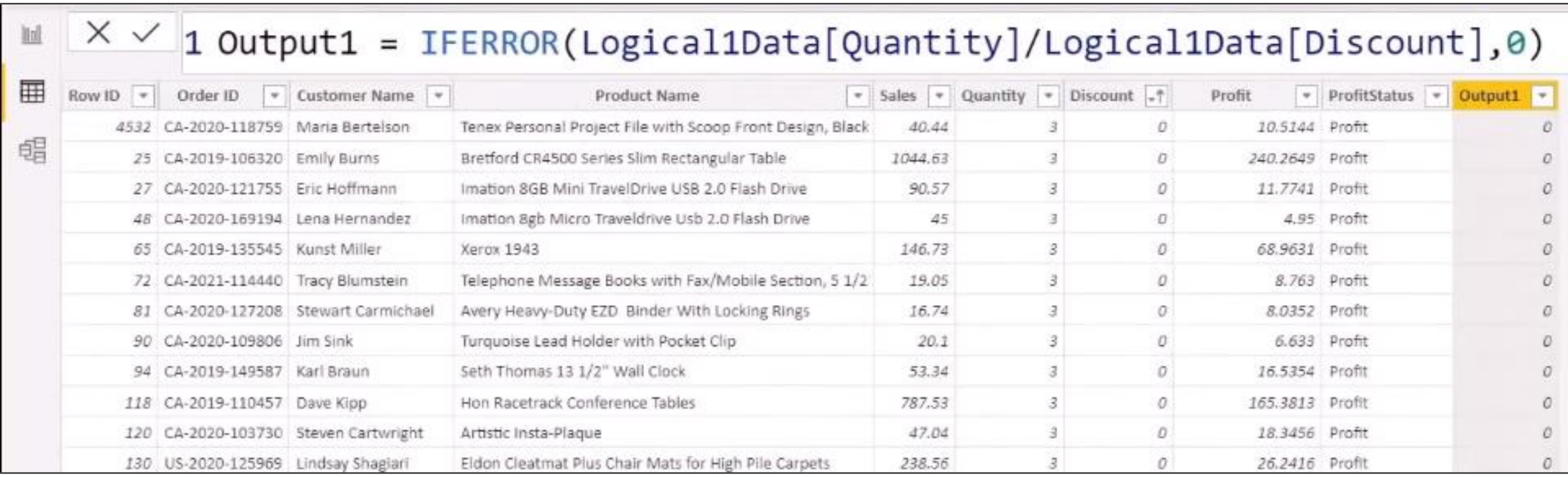
Structure | Formatting | Properties | Sort | Groups | Relationships | Calculations

```
1 Output1 = IFERROR(Logical1Data[Quantity]/Logical1Data[Discount],0
```

IFERROR(Value, **ValueIfError**)

Returns value_if_error if the first expression is an error and the value of the expression itself otherwise.

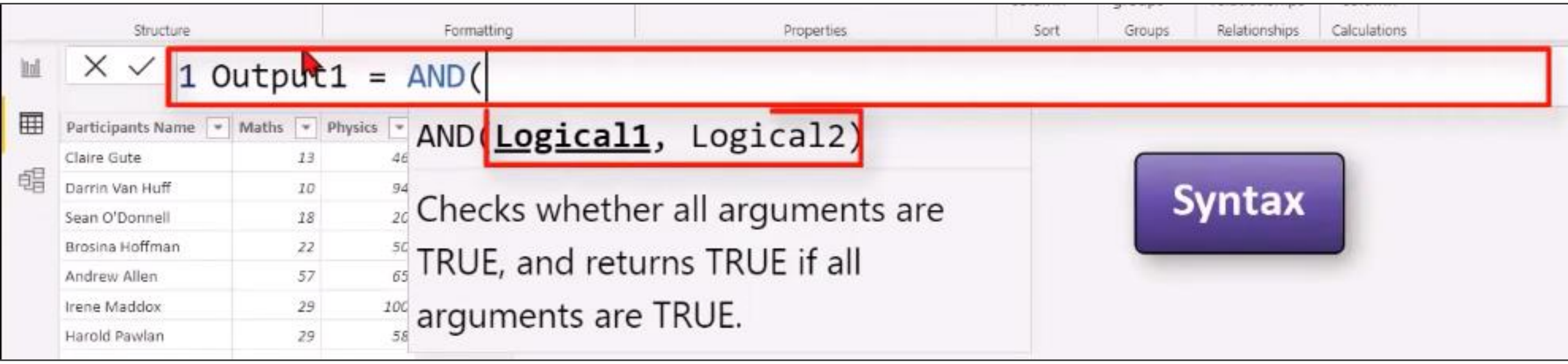
Row ID	Order ID	Customer Name	Product Name	Sales	Quantity	Discount	Profit	ProfitStatus	Output1
4532	CA-2020-118759	Maria Bertelson				0	10.5144	Profit	∞
25	CA-2019-106320	Emily Burns				0	240.2649	Profit	∞
27	CA-2020-121755	Eric Hoffmann				0	11.7741	Profit	∞
48	CA-2020-169194	Lena Hernandez				0	4.95	Profit	∞
65	CA-2019-135545	Kunst Miller				0	68.9631	Profit	∞
72	CA-2021-114440	Tracy Blumstein				0	8.763	Profit	∞
81	CA-2020-127208	Stewart Carmicha				0	8.0352	Profit	∞
90	CA-2020-109806	Jim Sink	Turquoise Lead Holder with Pocket Clip	20.1	3	0	6.633	Profit	∞



```
1 Output1 = IFERROR(Logical1Data[Quantity]/Logical1Data[Discount],0)
```

Row ID	Order ID	Customer Name	Product Name	Sales	Quantity	Discount	Profit	ProfitStatus	Output1
4532	CA-2020-118759	Maria Bertelson	Tenex Personal Project File with Scoop Front Design, Black	40.44	3	0	10.5144	Profit	0
25	CA-2019-106320	Emily Burns	Bretford CR4500 Series Slim Rectangular Table	1044.63	3	0	240.2649	Profit	0
27	CA-2020-121755	Eric Hoffmann	Imation 8GB Mini TravelDrive USB 2.0 Flash Drive	90.57	3	0	11.7741	Profit	0
48	CA-2020-169194	Lena Hernandez	Imation 8gb Micro Traveldrive Usb 2.0 Flash Drive	45	3	0	4.95	Profit	0
65	CA-2019-135545	Kunst Miller	Xerox 1943	146.73	3	0	68.9631	Profit	0
72	CA-2021-114440	Tracy Blumstein	Telephone Message Books with Fax/Mobile Section, 5 1/2	19.05	3	0	8.763	Profit	0
81	CA-2020-127208	Stewart Carmichael	Avery Heavy-Duty EZD Binder With Locking Rings	16.74	3	0	8.0352	Profit	0
90	CA-2020-109806	Jim Sink	Turquoise Lead Holder with Pocket Clip	20.1	3	0	6.633	Profit	0
94	CA-2019-149587	Karl Braun	Seth Thomas 13 1/2" Wall Clock	53.34	3	0	16.5354	Profit	0
118	CA-2019-110457	Dave Kipp	Hon Racetrack Conference Tables	787.53	3	0	165.3813	Profit	0
120	CA-2020-103730	Steven Cartwright	Artistic Insta-Plaque	47.04	3	0	18.3456	Profit	0
130	US-2020-125969	Lindsay Shagiari	Eldon Cleatmat Plus Chair Mats for High Pile Carpets	238.56	3	0	26.2416	Profit	0

AND()

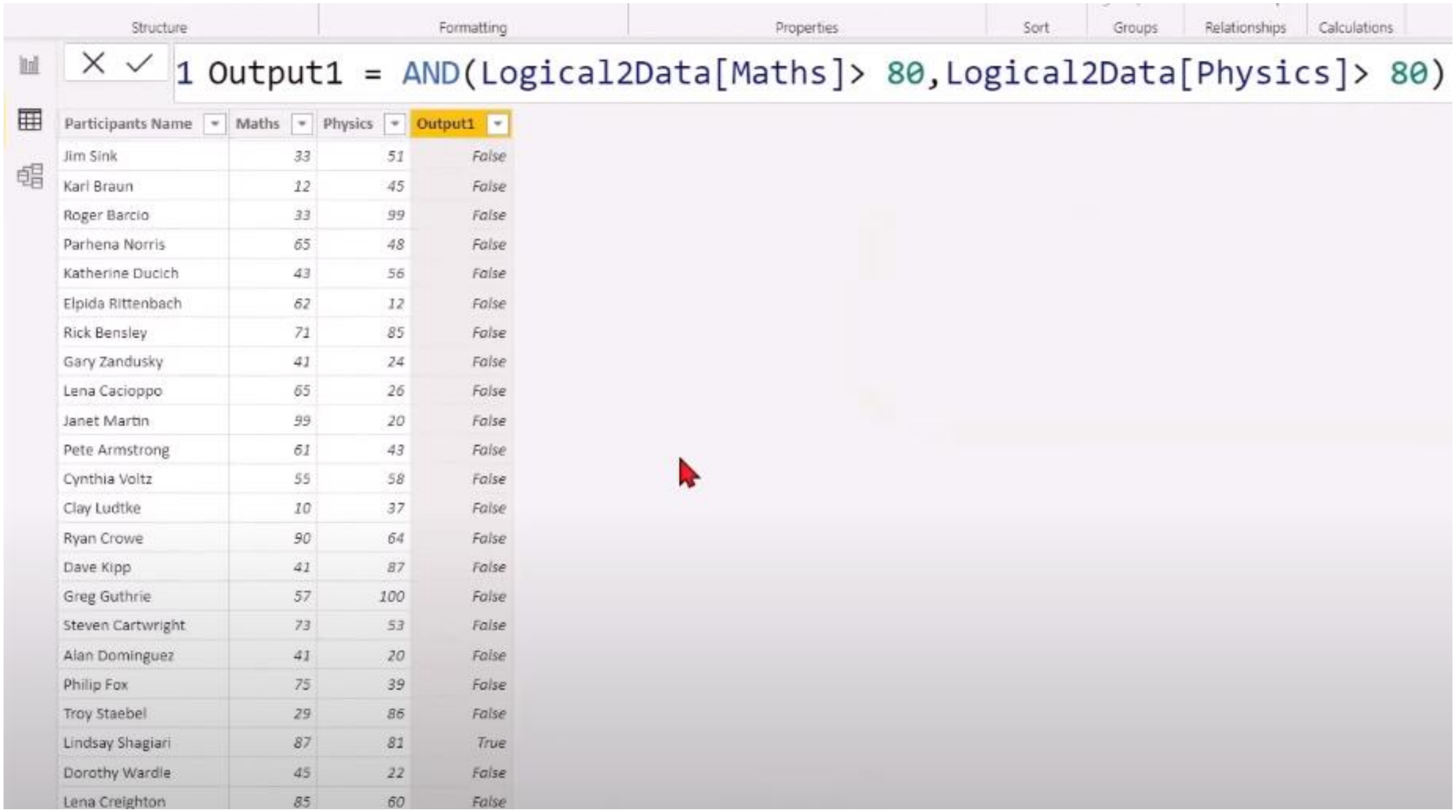


```
1 Output1 = AND(
```

AND(Logical1, Logical2)

Checks whether all arguments are TRUE, and returns TRUE if all arguments are TRUE.

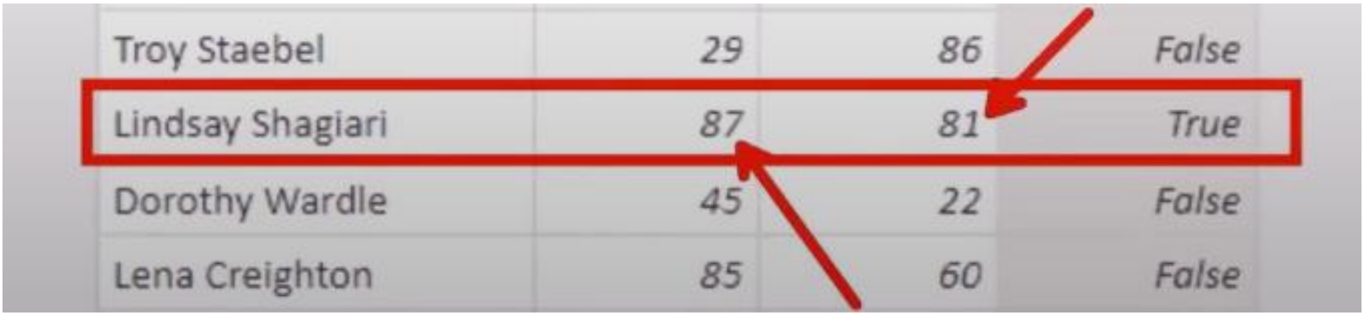
Participants Name	Maths	Physics
Claire Gute	13	46
Darrin Van Huff	10	94
Sean O'Donnell	18	20
Brosina Hoffman	22	50
Andrew Allen	57	65
Irene Maddox	29	100
Harold Pawlan	29	58



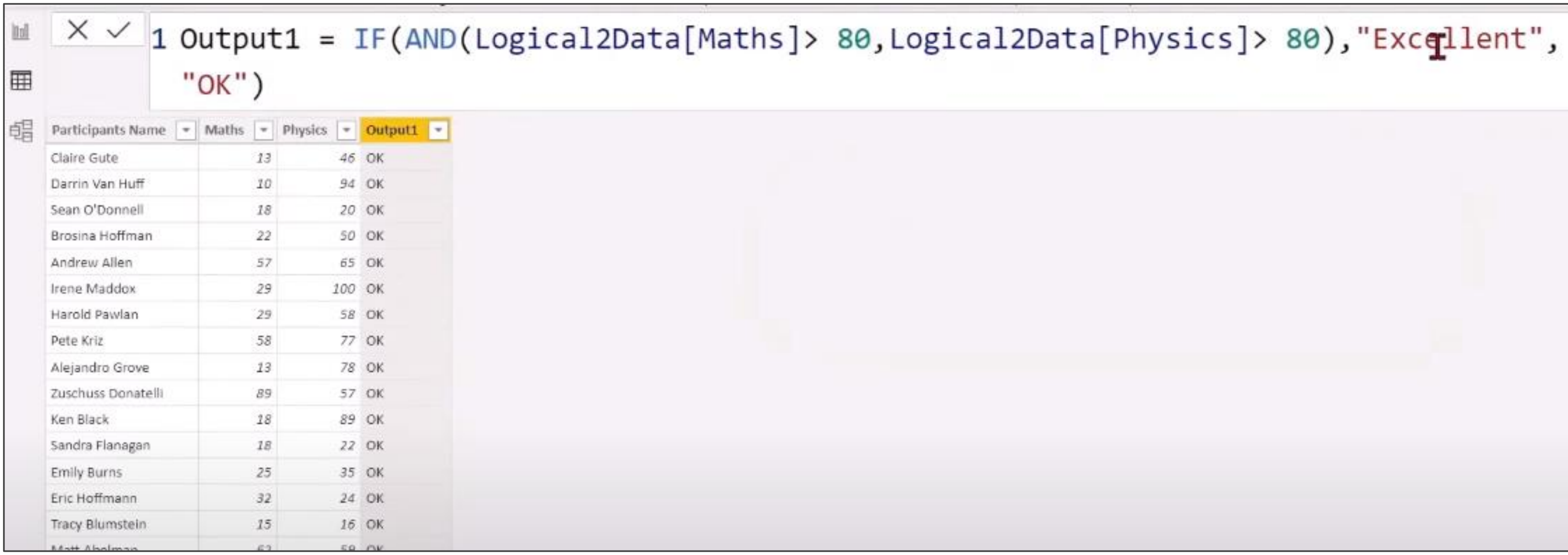
```
1 Output1 = AND(Logical2Data[Maths]> 80,Logical2Data[Physics]> 80)
```

Participants Name	Maths	Physics	Output1
Jim Sink	33	51	False
Karl Braun	12	45	False
Roger Barcio	33	99	False
Parhena Norris	65	48	False
Katherine Ducich	43	56	False
Elpida Rittenbach	62	12	False
Rick Bensley	71	85	False
Gary Zandusky	41	24	False
Lena Cacioppo	65	26	False
Janet Martin	99	20	False
Pete Armstrong	61	43	False
Cynthia Voltz	55	58	False
Clay Ludtke	10	37	False
Ryan Crowe	90	64	False
Dave Kipp	41	87	False
Greg Guthrie	57	100	False
Steven Cartwright	73	53	False
Alan Dominguez	41	20	False
Philip Fox	75	39	False
Troy Staebel	29	86	False
Lindsay Shagiari	87	81	True
Dorothy Wardle	45	22	False
Lena Creighton	85	60	False

IF(AND(),…..)



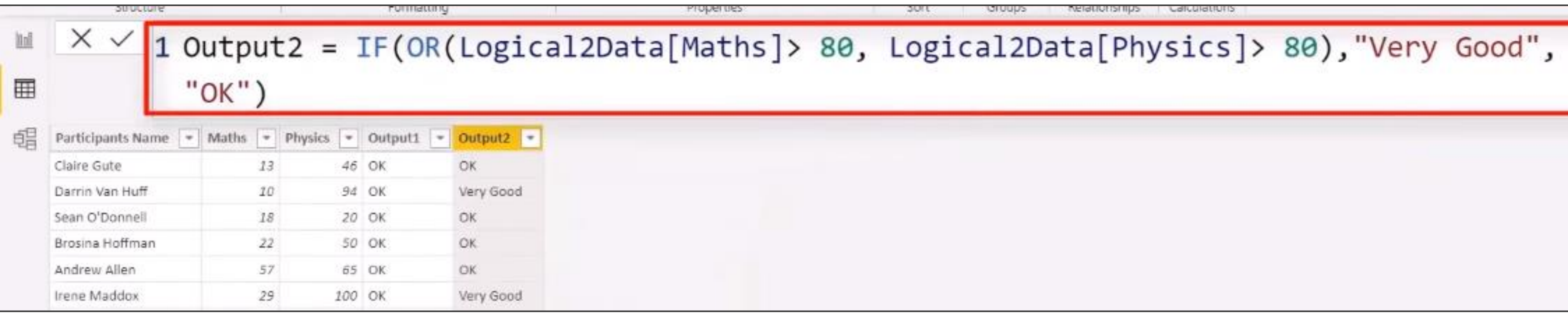
Troy Staebel	29	86	False
Lindsay Shagiari	87	81	True
Dorothy Wardle	45	22	False
Lena Creighton	85	60	False



```
Output1 = IF(AND(Logical2Data[Maths]> 80,Logical2Data[Physics]> 80),"Excellent",
  "OK")
```

Participants Name	Maths	Physics	Output1
Claire Gute	13	46	OK
Darrin Van Huff	10	94	OK
Sean O'Donnell	18	20	OK
Brosina Hoffman	22	50	OK
Andrew Allen	57	65	OK
Irene Maddox	29	100	OK
Harold Pawlan	29	58	OK
Pete Kriz	58	77	OK
Alejandro Grove	13	78	OK
Zuschuss Donatelli	89	57	OK
Ken Black	18	89	OK
Sandra Flanagan	18	22	OK
Emily Burns	25	35	OK
Eric Hoffmann	32	24	OK
Tracy Blumstein	15	16	OK
Matt Abelman	62	58	OK

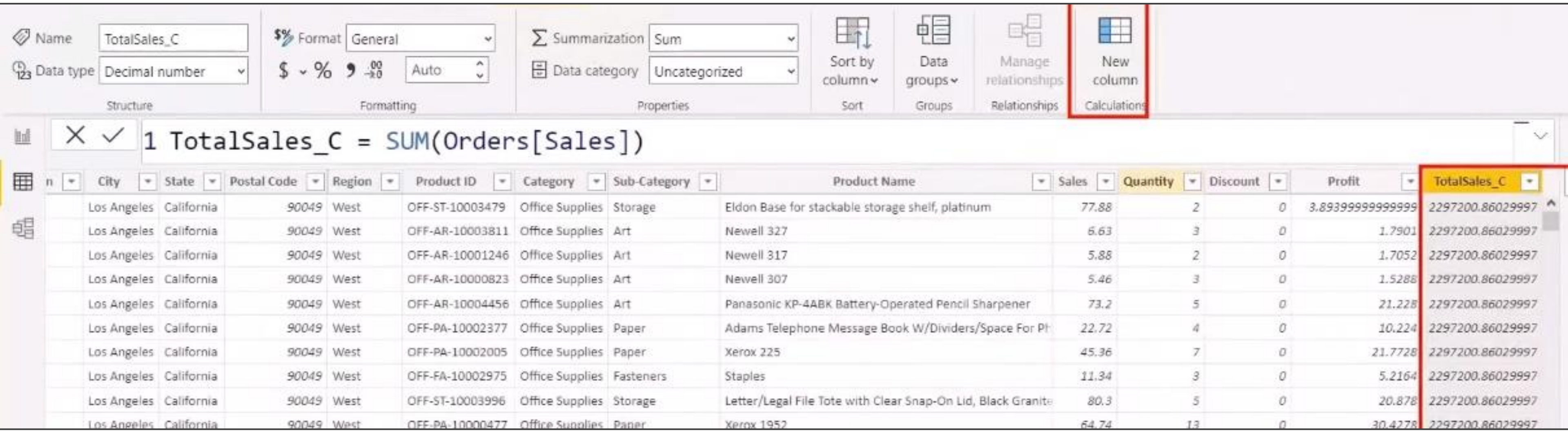
OR()



```
1 Output2 = IF(OR(Logical2Data[Maths]> 80, Logical2Data[Physics]> 80),"Very Good",
  "OK")
```

Participants Name	Maths	Physics	Output1	Output2
Claire Gute	13	46	OK	OK
Darrin Van Huff	10	94	OK	Very Good
Sean O'Donnell	18	20	OK	OK
Brosina Hoffman	22	50	OK	OK
Andrew Allen	57	65	OK	OK
Irene Maddox	29	100	OK	Very Good

Calculated Column



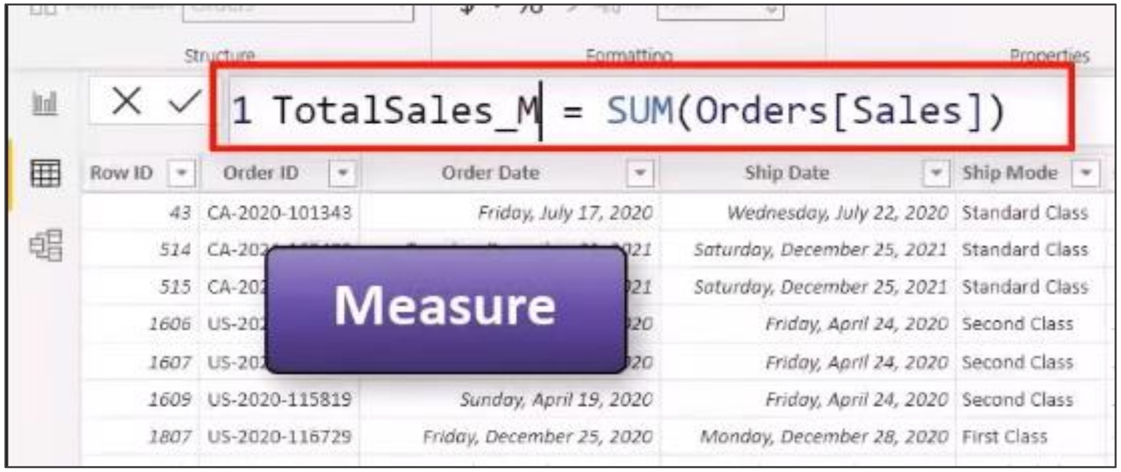
Name: TotalSales_C
Data type: Decimal number
Format: General
Auto
Summarization: Sum
Data category: Uncategorized

```
1 TotalSales_C = SUM(Orders[Sales])
```

n	City	State	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	TotalSales_C
	Los Angeles	California	90049	West	OFF-ST-10003479	Office Supplies	Storage	Eldon Base for stackable storage shelf, platinum	77.88	2	0	3.89399999999999	2297200.86029997
	Los Angeles	California	90049	West	OFF-AR-10003811	Office Supplies	Art	Newell 327	6.63	3	0	1.7901	2297200.86029997
	Los Angeles	California	90049	West	OFF-AR-10001246	Office Supplies	Art	Newell 317	5.88	2	0	1.7052	2297200.86029997
	Los Angeles	California	90049	West	OFF-AR-10000823	Office Supplies	Art	Newell 307	5.46	3	0	1.5288	2297200.86029997
	Los Angeles	California	90049	West	OFF-AR-10004456	Office Supplies	Art	Panasonic KP-4ABK Battery-Operated Pencil Sharpener	73.2	5	0	21.228	2297200.86029997
	Los Angeles	California	90049	West	OFF-PA-10002377	Office Supplies	Paper	Adams Telephone Message Book W/Dividers/Space For Ph	22.72	4	0	10.224	2297200.86029997
	Los Angeles	California	90049	West	OFF-PA-10002005	Office Supplies	Paper	Xerox 225	45.36	7	0	21.7728	2297200.86029997
	Los Angeles	California	90049	West	OFF-FA-10002975	Office Supplies	Fasteners	Staples	11.34	3	0	5.2164	2297200.86029997
	Los Angeles	California	90049	West	OFF-ST-10003996	Office Supplies	Storage	Letter/Legal File Tote with Clear Snap-On Lid, Black Granite	80.3	5	0	20.878	2297200.86029997
	Los Angeles	California	90049	West	OFF-PA-10000477	Office Supplies	Paper	Xerox 1952	64.74	13	0	30.4278	2297200.86029997

Measure

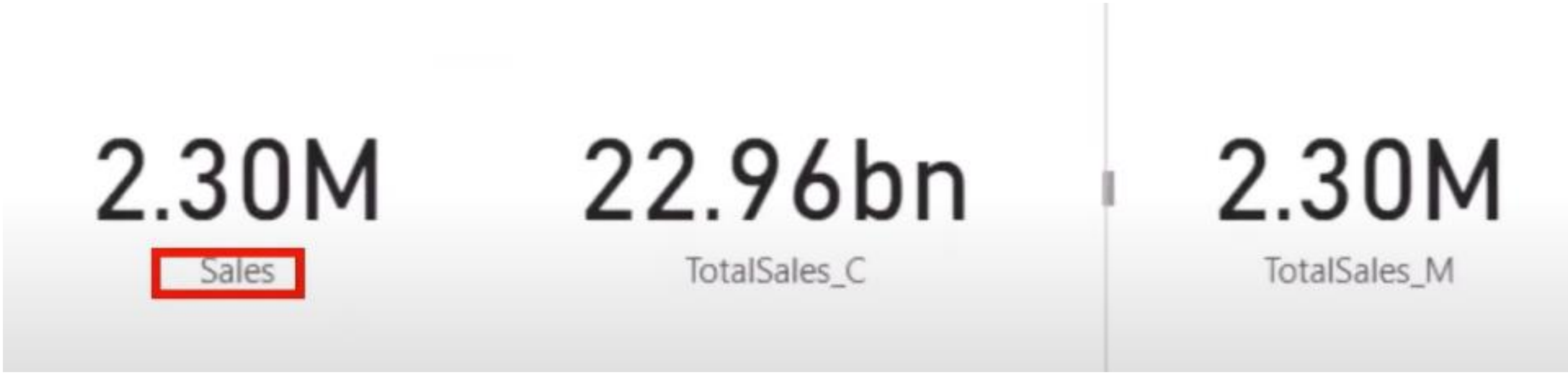
No New Column Gets added)



```
TotalSales_M = SUM(Orders[Sales])
```

Row ID	Order ID	Order Date	Ship Date	Ship Mode
43	CA-2020-101343	Friday, July 17, 2020	Wednesday, July 22, 2020	Standard Class
514	CA-202	21	Saturday, December 25, 2021	Standard Class
515	CA-202	21	Saturday, December 25, 2021	Standard Class
1606	US-202	20	Friday, April 24, 2020	Second Class
1607	US-202	20	Friday, April 24, 2020	Second Class
1609	US-2020-115819	Sunday, April 19, 2020	Friday, April 24, 2020	Second Class
1807	US-2020-116729	Friday, December 25, 2020	Monday, December 28, 2020	First Class

Measure



Drag 3 Cards and place:
Sales, Calculated column and Measure
Note the Difference

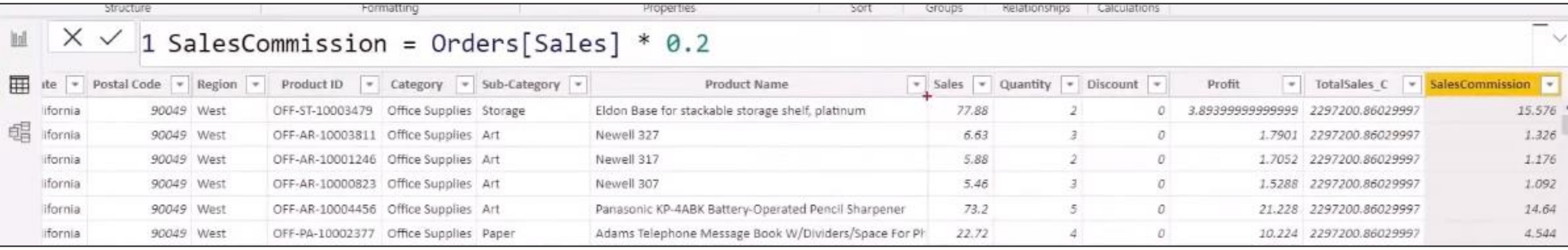
Note:

1. **If you want to do a calculation based on Rows, go for Calculated Columns**

2. **If you want a Column based calculation go for Measures**

Use of Calculated Columns

Calculated Sales Commission

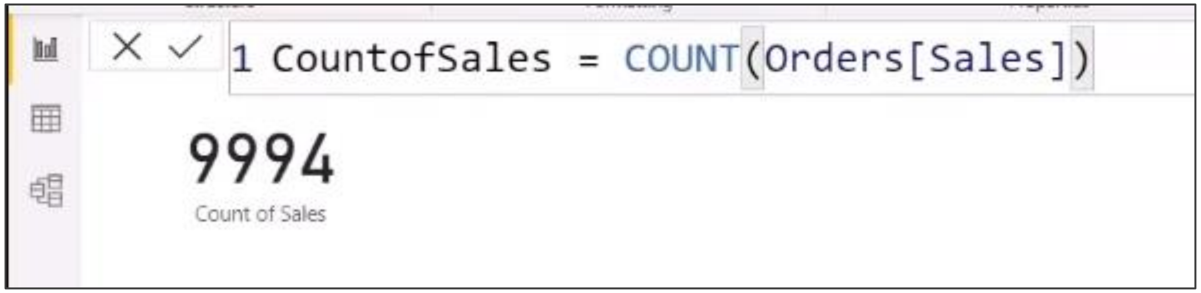


```
1 SalesCommission = Orders[Sales] * 0.2
```

ite	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	TotalSales_C	SalesCommission
ifornia	90049	West	OFF-ST-10003479	Office Supplies	Storage	Eldon Base for stackable storage shelf, platinum	77.88	2	0	3.89399999999999	2297200.86029997	15.576
ifornia	90049	West	OFF-AR-10003811	Office Supplies	Art	Newell 327	6.63	3	0	1.7901	2297200.86029997	1.326
ifornia	90049	West	OFF-AR-10001246	Office Supplies	Art	Newell 317	5.88	2	0	1.7052	2297200.86029997	1.176
ifornia	90049	West	OFF-AR-10000823	Office Supplies	Art	Newell 307	5.46	3	0	1.5288	2297200.86029997	1.092
ifornia	90049	West	OFF-AR-10004456	Office Supplies	Art	Panasonic KP-4ABK Battery-Operated Pencil Sharpener	73.2	5	0	21.228	2297200.86029997	14.64
ifornia	90049	West	OFF-PA-10002377	Office Supplies	Paper	Adams Telephone Message Book W/Dividers/Space For Pl	22.72	4	0	10.224	2297200.86029997	4.544

Measure Implicit & Explicit

- **Implicit – Where we simply drag & drop**
- **Explicit – Are the once's that we explicitly write for doing a calculation**



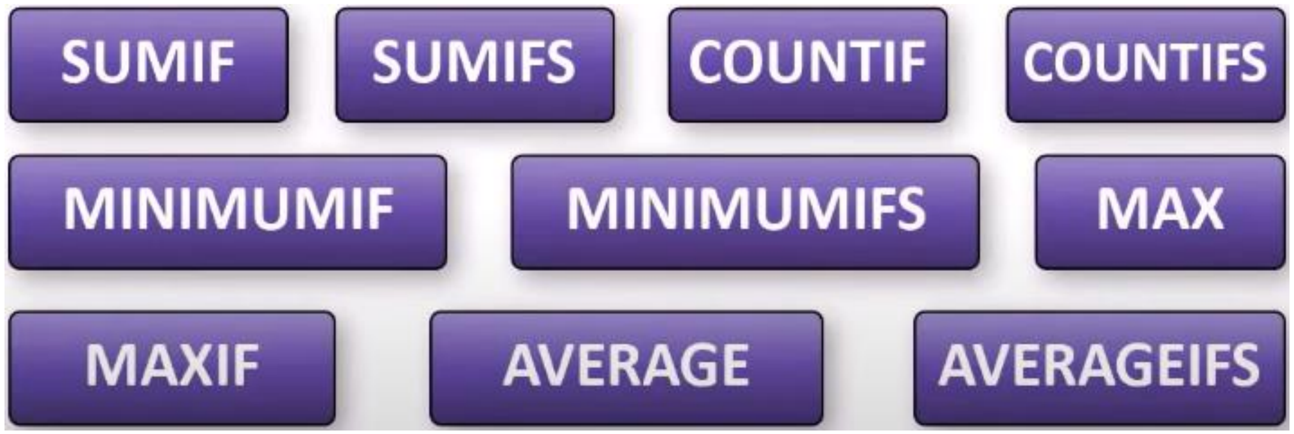
```
CountofSales = COUNT(Orders[Sales])
```

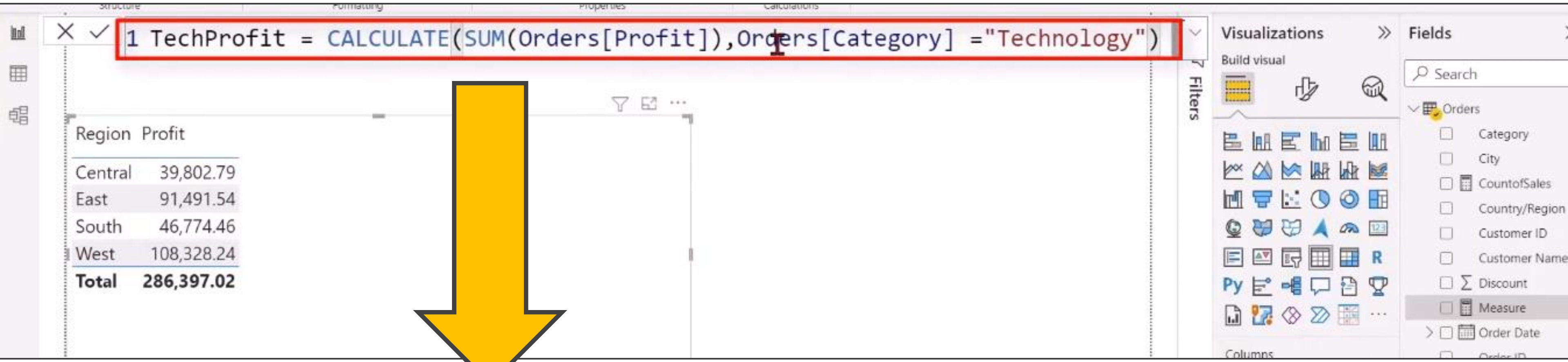
- **Create as many Measures possible, as these are light weight and can be reused and are faster that implicit functions.**

CALCULATE()

CALCULATE Function

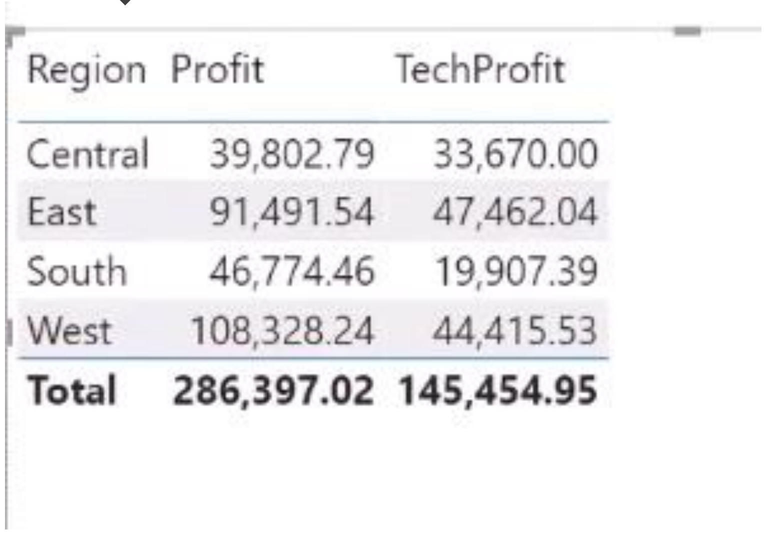
CALCULATE Can replace all these functions



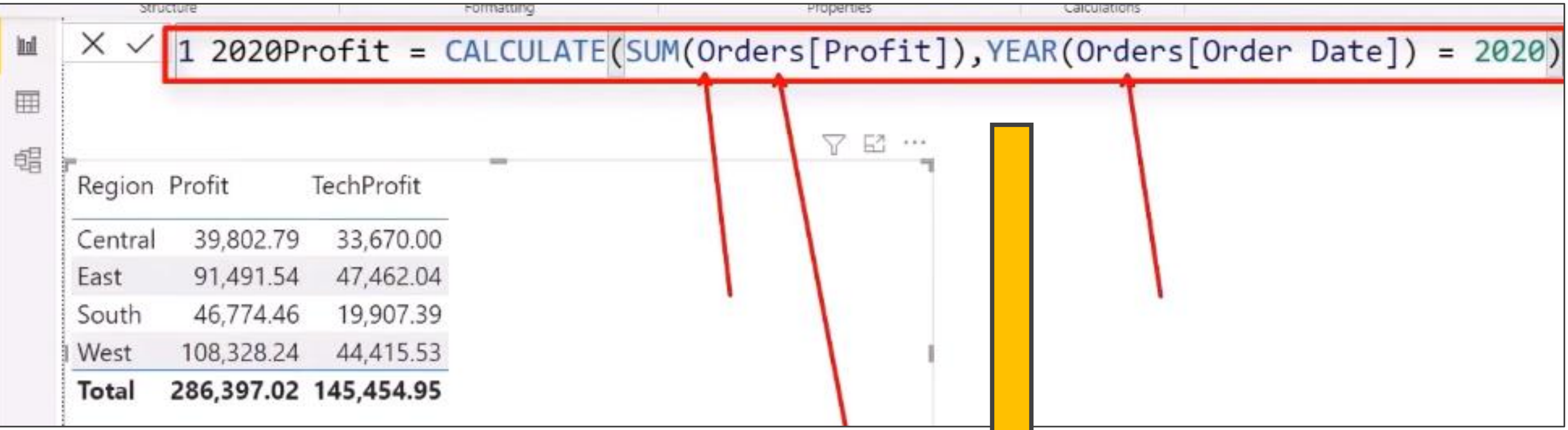


```
TechProfit = CALCULATE(SUM(Orders[Profit]),Orders[Category] ="Technology")
```

Region	Profit
Central	39,802.79
East	91,491.54
South	46,774.46
West	108,328.24
Total	**286,397.02**

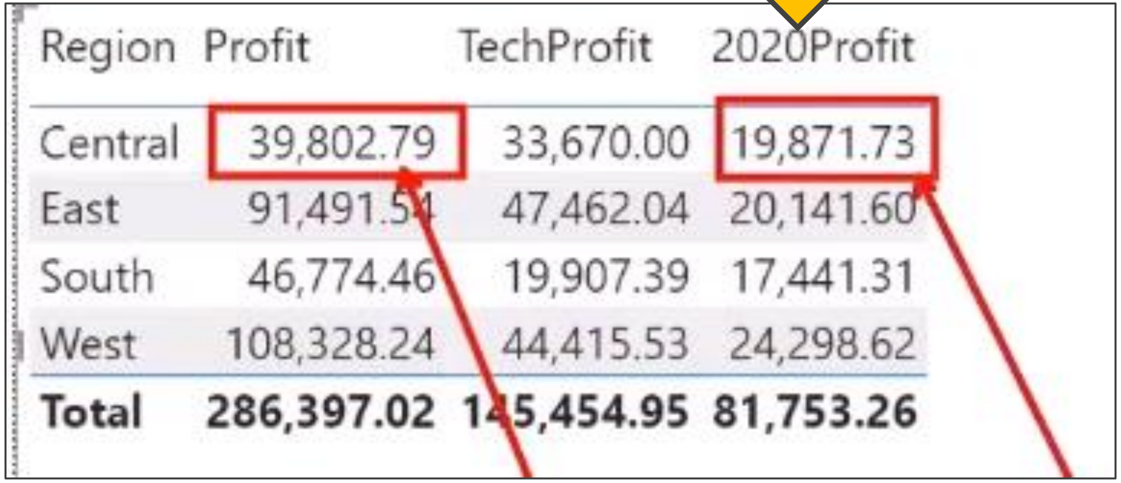


Region	Profit	TechProfit
Central	39,802.79	33,670.00
East	91,491.54	47,462.04
South	46,774.46	19,907.39
West	108,328.24	44,415.53
Total	**286,397.02**	**145,454.95**

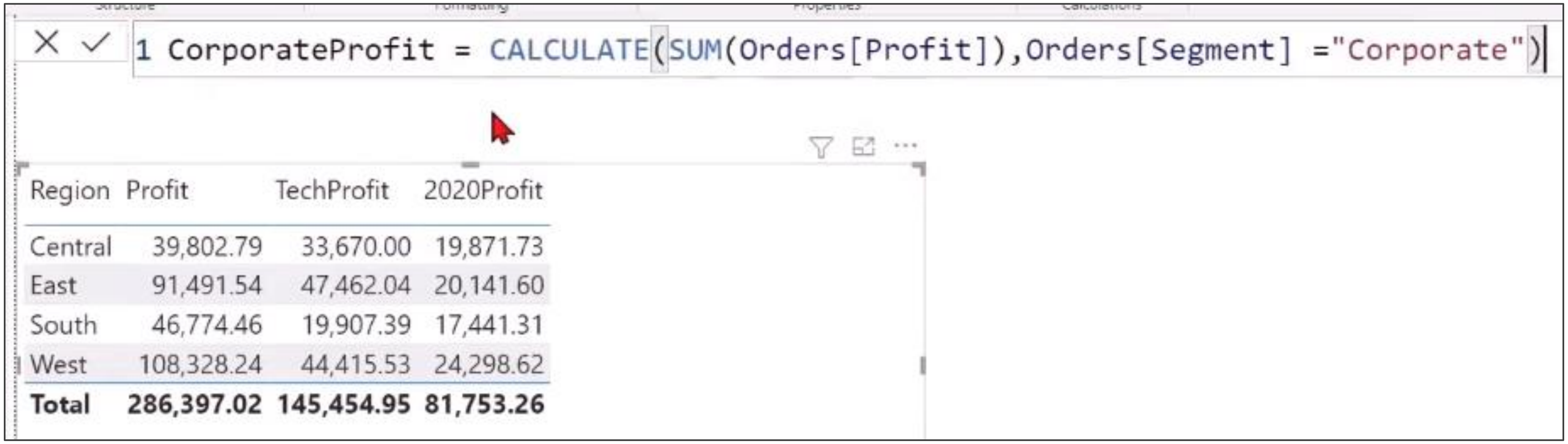


```
1 2020Profit = CALCULATE(SUM(Orders[Profit]),YEAR(Orders[Order Date]) = 2020)
```

Region	Profit	TechProfit
Central	39,802.79	33,670.00
East	91,491.54	47,462.04
South	46,774.46	19,907.39
West	108,328.24	44,415.53
Total	**286,397.02**	**145,454.95**

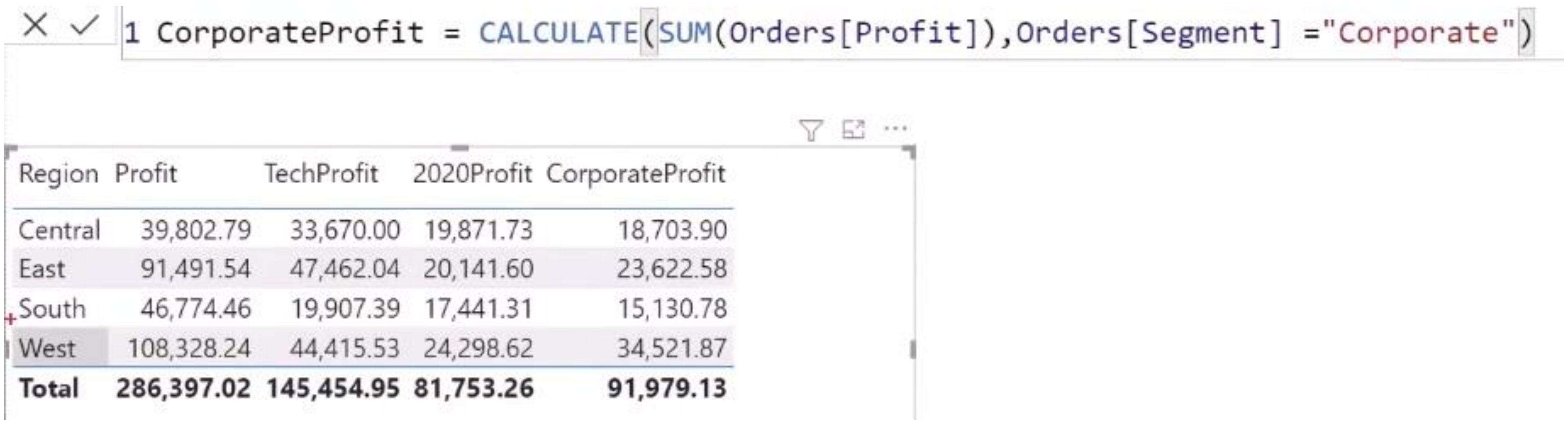


Region	Profit	TechProfit	2020Profit
Central	39,802.79	33,670.00	19,871.73
East	91,491.54	47,462.04	20,141.60
South	46,774.46	19,907.39	17,441.31
West	108,328.24	44,415.53	24,298.62
Total	**286,397.02**	**145,454.95**	**81,753.26**



```
CorporateProfit = CALCULATE(SUM(Orders[Profit]),Orders[Segment] ="Corporate")
```

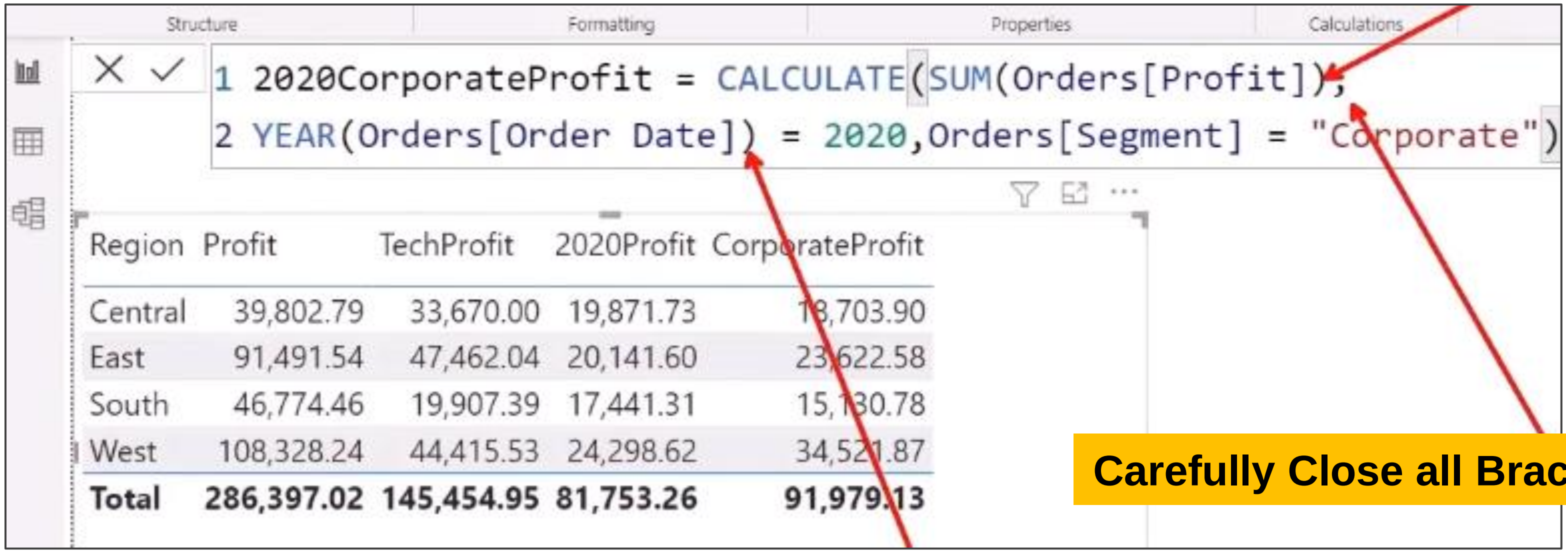
Region	Profit	TechProfit	2020Profit
Central	39,802.79	33,670.00	19,871.73
East	91,491.54	47,462.04	20,141.60
South	46,774.46	19,907.39	17,441.31
West	108,328.24	44,415.53	24,298.62
Total	**286,397.02**	**145,454.95**	**81,753.26**



```
CorporateProfit = CALCULATE(SUM(Orders[Profit]),Orders[Segment] ="Corporate")
```

Region	Profit	TechProfit	2020Profit	CorporateProfit
Central	39,802.79	33,670.00	19,871.73	18,703.90
East	91,491.54	47,462.04	20,141.60	23,622.58
South	46,774.46	19,907.39	17,441.31	15,130.78
West	108,328.24	44,415.53	24,298.62	34,521.87
Total	**286,397.02**	**145,454.95**	**81,753.26**	**91,979.13**

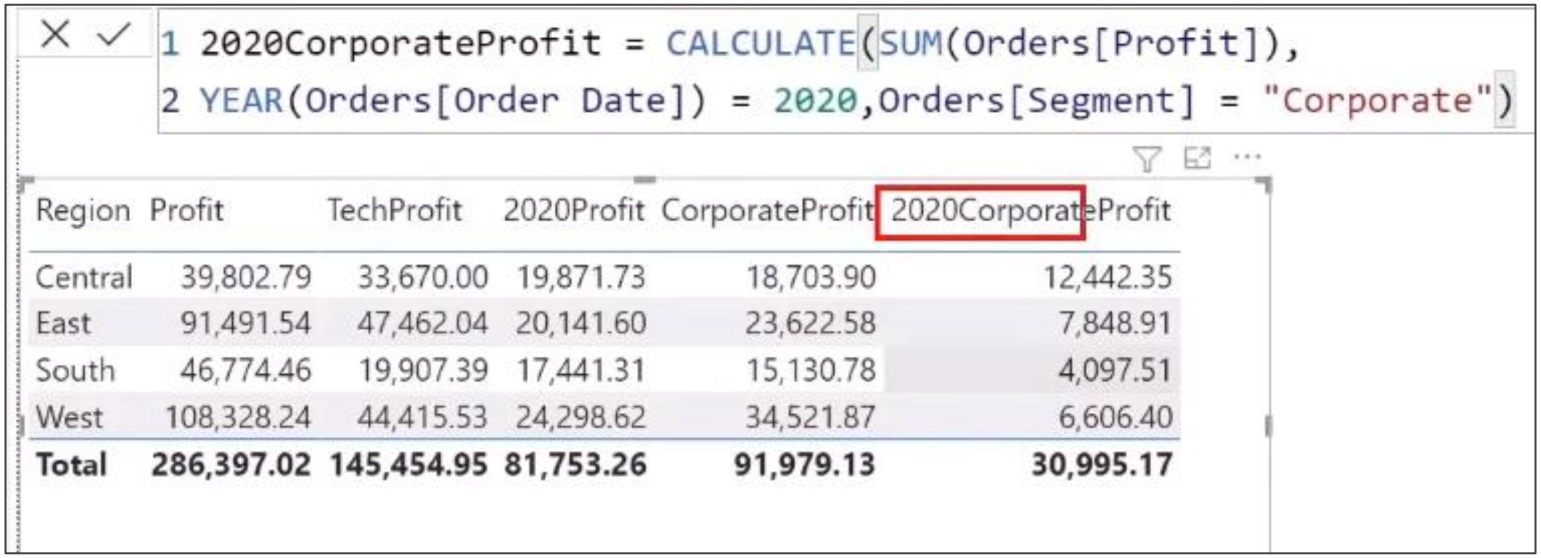
Calculate() - Using two Filters



```
2020CorporateProfit = CALCULATE(SUM(Orders[Profit]),
YEAR(Orders[Order Date]) = 2020,Orders[Segment] = "Corporate")
```

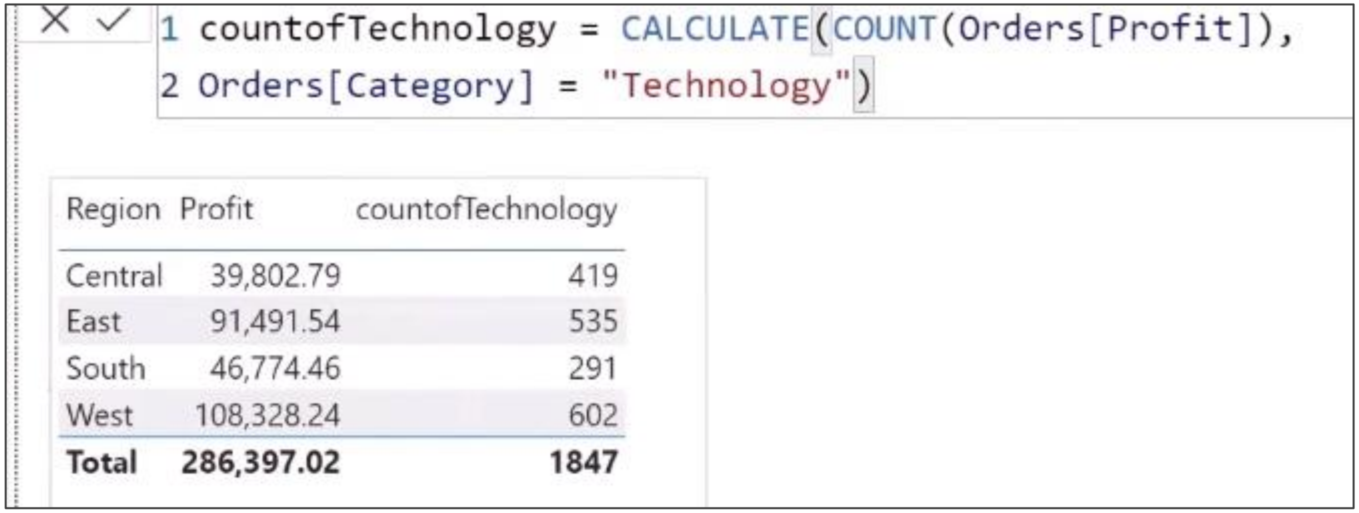
Region	Profit	TechProfit	2020Profit	CorporateProfit
Central	39,802.79	33,670.00	19,871.73	18,703.90
East	91,491.54	47,462.04	20,141.60	23,622.58
South	46,774.46	19,907.39	17,441.31	15,130.78
West	108,328.24	44,415.53	24,298.62	34,521.87
Total	**286,397.02**	**145,454.95**	**81,753.26**	**91,979.13**

Carefully Close all Brackets



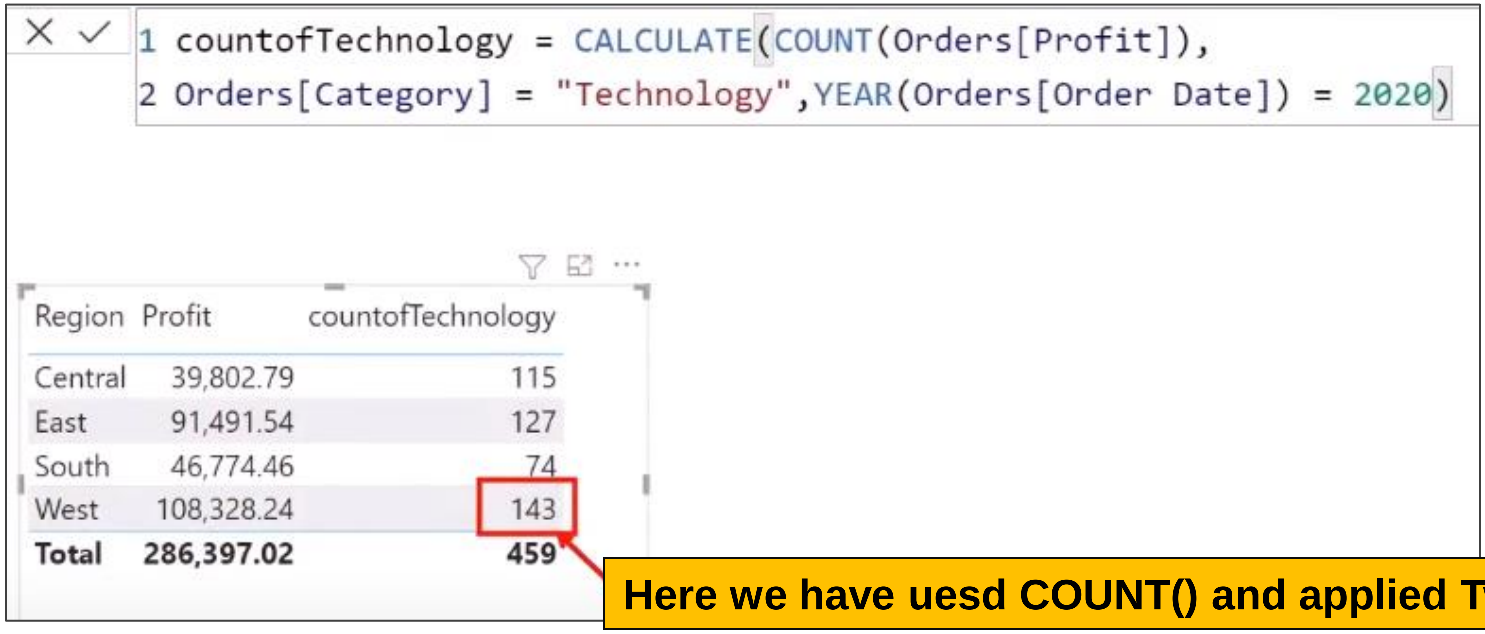
```
2020CorporateProfit = CALCULATE(SUM(Orders[Profit]),
YEAR(Orders[Order Date]) = 2020,Orders[Segment] = "Corporate")
```

Region	Profit	TechProfit	2020Profit	CorporateProfit	2020CorporateProfit
Central	39,802.79	33,670.00	19,871.73	18,703.90	12,442.35
East	91,491.54	47,462.04	20,141.60	23,622.58	7,848.91
South	46,774.46	19,907.39	17,441.31	15,130.78	4,097.51
West	108,328.24	44,415.53	24,298.62	34,521.87	6,606.40
Total	**286,397.02**	**145,454.95**	**81,753.26**	**91,979.13**	**30,995.17**



```
countofTechnology = CALCULATE(COUNT(Orders[Profit]),
Orders[Category] = "Technology")
```

Region	Profit	countofTechnology
Central	39,802.79	419
East	91,491.54	535
South	46,774.46	291
West	108,328.24	602
Total	**286,397.02**	**1847**

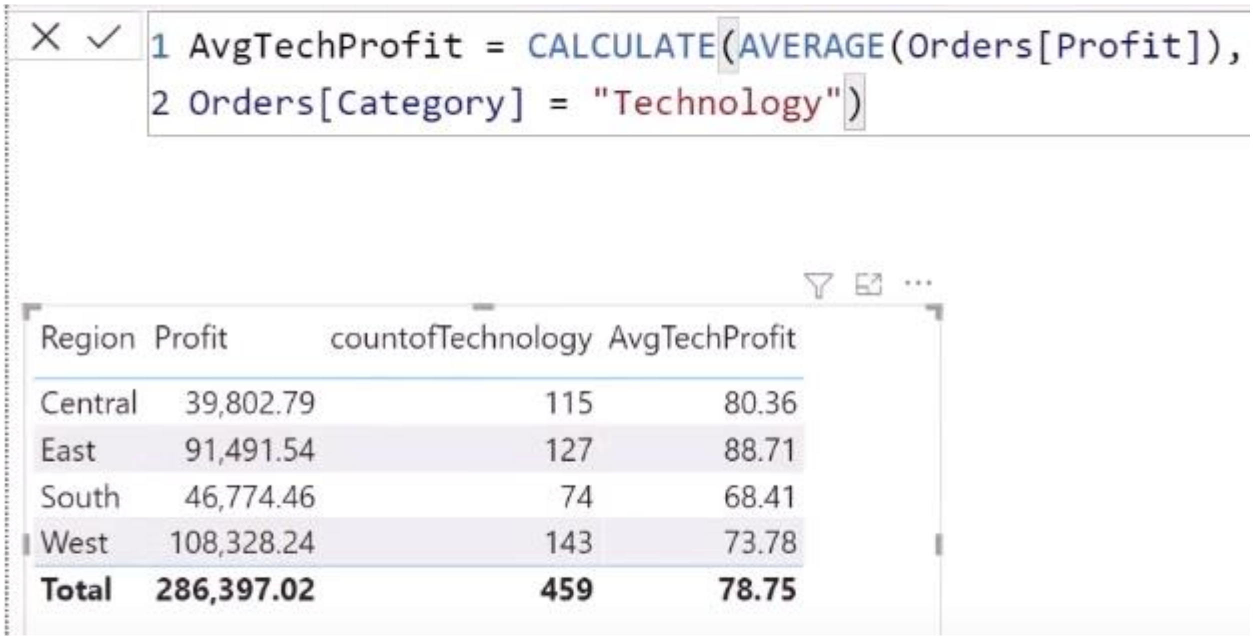


```
countofTechnology = CALCULATE(COUNT(Orders[Profit]),
Orders[Category] = "Technology",YEAR(Orders[Order Date]) = 2020)
```

Region	Profit	countofTechnology
Central	39,802.79	115
East	91,491.54	127
South	46,774.46	74
West	108,328.24	143
Total	**286,397.02**	**459**

Here we have uesd COUNT() and applied Two Filters

CALCULATE() with AVERAGE()



```
AvgTechProfit = CALCULATE(AVERAGE(Orders[Profit]),
Orders[Category] = "Technology")
```

Region	Profit	countofTechnology	AvgTechProfit
Central	39,802.79	115	80.36
East	91,491.54	127	88.71
South	46,774.46	74	68.41
West	108,328.24	143	73.78
Total	**286,397.02**	**459**	**78.75**

Power BI DAX Functions Syntax Guide

1. Aggregate Functions
2. Count Functions
3. Date & Time Functions
4. Mathematical Functions
5. Logical Functions
6. Information Function
7. Text Function

Aggregate Functions

MIN, MINA & MINX

MIN

This DAX function returns the minimum numeric value in a column, or between two scalar expressions.

Syntax

MIN(<column>)

Example

= MIN([ResellerMargin])

MINA

This DAX function returns the minimum value in a column, including any logical values and numbers represented as text.

Syntax

MINA(<column>)

Example

=MINA(([PostalCode])

MINX

This DAX function returns the minimum numeric value that results from evaluating an expression for each row of a table.

Syntax

MINX(<table>, < expression evaluated for each row>)

Example

= MINX(FILTER(InternetSales, InternetSales[SalesTerritoryKey] = 5), InternetSales[Freight] + InternetSales[TaxAmt])

MAX, MAXA & MAXX

MAX

This DAX function returns the maximum value in a column, including any logical values and numbers represented as text.

Syntax

MAX(<column>)

Example

=MAX([ResellerMargin])

MAXA

This DAX function returns the maximum value in a column, including any logical values and numbers represented as text.

Syntax

MAXA(<column>)

Example

= MAXA(([PostalCode])

MAXX

This DAX function returns the maximum numeric value that results from evaluating an expression for each row of a table.

Syntax

MAXX(<table>, <expression evaluated for each row>)

Example

=MAXX(FILTER(InternetSales, InternetSales[SalesTerritoryKey] = 5), InternetSales[Freight] + InternetSales[TaxAmt])

SUM & SUMX

SUM

This DAX function adds all the numbers in a column.

Syntax

```
SUM(<column>)
```

Example

```
= SUM(Sales[Amt])
```

SUMX

This DAX function returns the sum of an expression evaluated for each row in a table.

Syntax

```
SUMX(<table>, <expression evaluated for each row>)
```

Example

```
= SUMX(FILTER(InternetSales, InternetSales[SalesTerritoryID]=5),[Freight])
```

AVERAGE & AVERAGEX

AVERAGE

This DAX function returns the arithmetic mean of the values in a column.

Syntax

```
AVERAGE(<column>)
```

Example

```
= AVERAGE(InternetSales[ExtendedSalesAmount])
```

AVERAGEX

This DAX function calculates the arithmetic mean of a set of expressions evaluated over a table.

Syntax

```
AVERAGEX(<table>, <expression evaluated for each row>)
```

Example

```
= AVERAGEX(InternetSales, InternetSales[Freight]+ InternetSales[TaxAmt])
```

Count Functions

DISTINCTCOUNT & COUNT

DISTINCTCOUNT

This is a DAX function used to return the distinct count of items in a column. So, if there are multiple numbers of the same item, this function will count it as a single item.

Syntax

```
DISTINCTCOUNT(<column>)
```

Example

```
= DISTINCTCOUNT(ResellerSales_USD[SalesOrderNumber])
```

COUNT

This is a DAX function used to return the count of items in a column. So, if there are multiple numbers of the same item, this function will count it as separate items and not a single item.

Syntax

```
COUNT(<column>)
```

Examples

```
= COUNT([ShipDate])
```

COUNTA & COUNTROWS

COUNTA

This is a DAX function used to return the count of items, in a column, that is not empty.

Syntax

```
COUNTA(<column>)
```

Example

```
= COUNTA('Reseller'[Phone])
```

COUNTROWS

This is a DAX function that counts the number of rows in the specified table, or in a table defined by an expression.

Syntax

```
COUNTROWS(<table>)
```

Example

```
= COUNTROWS('Orders')
```

COUNTBLANK

COUNTBLANK

This is a DAX function that counts the number of blank cells in a column.

Syntax

```
COUNTBLANK(<column>)
```

Example

```
= COUNTBLANK(Reseller[BankName])
```

Date Functions

DATE

DATE

This DAX function returns the specified date in Date-Time format.

Syntax

```
DATE(<year>, <month>, <day>)
```

Example

```
=DATE(2019,12,17)
```

HOUR

This DAX function returns the specified hour as a number from 0 to 23 (12:00 A.M. to 11:00 P.M.).

Syntax

```
HOUR(<datetime>)
```

Example

```
=HOUR('Orders'[TransactionTime])
```

TODAY

This DAX function returns the current date.

Syntax

TODAY()

NOW

This DAX function returns the current date and time in Date-Time format.

Syntax

NOW()

EOMONTH

This DAX function returns the date in Date-Time format of the last day of the month, before or after a specified number of months.

Syntax

EOMONTH(<start_date>, <months>)

Example

= EOMONTH("March 3, 2008",1.5)

Mathematical Functions

Mathematical Functions

ABS

This DAX function returns the absolute value of the number given.

Syntax

```
ABS(<number>)
```

Example

```
= ABS([DealerPrice]-[ListPrice])
```

FACT

This DAX function returns the factorial of a number.

Syntax

```
FACT(<number>)
```

Example

```
= FACT([Values])
```

EXP

This DAX function returns the value of e raised to the power of the given number.

Syntax

```
EXP(<number>)
```

Example

```
= EXP([Power])
```

LN

This DAX function returns the natural log of the given number.

Syntax

```
LN(<number>)
```

Example

```
= LN([Values])
```

LOG

This DAX function returns the log with the base of the given number.

Syntax

LOG(<number>,<base>)

Example

All the following return the same result, 2.

```
=LOG(100,10)

=LOG(100)

=LOG10(100)
```

PI

This DAX function returns the value of Pi

Syntax

PI()

POWER

This DAX function returns the value of the first argument raised to the power of the second argument.

Syntax

```
POWER(<number>, <power>)
```

Example

```
=POWER(5,2)
```

QUOTIENT

This DAX function performs division returns the integer part of the quotient.

Syntax

```
QUOTIENT(<dividend>, <divisor>)
```

Example

```
= QUOTIENT(5,2)
```

SIGN

This DAX function returns the sign of a given number.

Syntax

```
SIGN(<number>)
```

Example

```
=SIGN( ([Sale Price] - [Cost Price]) )
```

SQRT

This DAX function returns the square root of the given number.

Syntax

```
SQRT(<number>)
```

Example

```
=SQRT(25)
```

Logical Functions

AND

This DAX function performs logical AND(conjunction) on two expressions. For AND to return true, both conditions specified have to be fulfilled.

Syntax

```
AND(<logical argument1>,<logical argument2>)
```

Example

```
=IF(AND(10 > 9, -10 < -1), "All true", "One or more false"
```

Because both conditions, passed as arguments, to the AND function are true, the formula returns "All True".

OR

This DAX function performs logical OR(disjunction) on two expressions. For OR to return true, either of the two conditions specified has to be fulfilled.

Syntax

OR(<logical argument1>,<logical argument2>)

Example

=IF(OR(10 > 9, -10 >-1), "True", "False"

Because one of the conditions, passed as arguments, to the OR function is true, the formula returns "True".

NOT

This DAX function performs logical NOT (negation) on given expression.

Syntax

NOT(<logical argument>)

Example

=NOT([CalculatedColumn1])

For each row in Calculated Column1, the NOT function returns the logical opposite of the given value.

IF

This DAX function tests a series of inputs for the one that fulfills the condition specified in the argument.

Syntax

IF(logical_test>,<value_if_true>, value_if_false)

Example

=IF([Calls]<200,"low",IF([Calls]<300,"medium","high"))

Information Functions

IFERROR

This DAX function evaluates an expression and returns a specified value if the expression returns an error.

Syntax

```
IFERROR(value, value_if_error)
```

Example

```
= IFERROR(25/0,9999)
```

ISBLANK

This DAX function returns TRUE or FALSE after checking whether a value is blank.

Syntax

```
ISBLANK(<value>)
```

Example

```
=IF( ISBLANK('CalculatedMeasures'[PreviousYearTotalSales]) , BLANK() , ( 'CalculatedMeasures'[Total Sales]-'CalculatedMeasures'[PreviousYearTotalSales] ) /'CalculatedMeasures'[PreviousYearTotalSales])
```

ISNUMBER

This DAX function returns TRUE or FALSE after checking whether a value is numeric.

Syntax

```
ISNUMBER(<value>)
```

Example

```
= IF(ISNUMBER(0), "Is number", "Is Not number")
```

ISTEXT

This DAX function returns TRUE or FALSE after checking whether a value is a text.

Syntax

```
ISTEXT(<value>)
```

Example

```
=IF(ISTEXT("text"), "Is Text", "Is Non-Text")
```

ISNONTEXT

This DAX function returns TRUE or FALSE after checking whether a value is non-text.

Syntax

```
ISNONTEXT(<value>)
```

Example

```
=IF(ISNONTEXT("text"), "Is Non-Text", "Is Text")
```

ISERROR

This DAX function returns TRUE or FALSE after checking whether a value is an error.

Syntax

ISERROE(<value>)

Example

```
=IF( ISERROR( SUM('ResellerSales_USD'[SalesAmount_USD])
/SUM('InternetSales_USD'[SalesAmount_USD]) ) , BLANK() ,
SUM('ResellerSales_USD'[SalesAmount_USD]) /SUM('InternetSales_USD'[SalesAmount_USD]) )
```

Text Functions

CONCATENATE

This DAX function joins two text strings into one.

Syntax

```
CONCATENATE(<text1>, <text2>)
```

Example

```
= CONCATENATE("Hello ", "World")
```

CONCATENATEX

This DAX function the result of an expression evaluated for each row in a table.

Syntax

```
CONCATENATEX(<table>, <expression>, [delimiter])
```

Example

```
= CONCATENATEX(Employees, [FirstName] & “ “ & [LastName], “,”)
```

FIXED

This DAX function rounds a number to the specified number of decimals and returns the result as text.

Syntax

```
FIXED(<number>, <decimals>, <no_commas>)
```

Example

```
= FIXED([PctCost],3,1)
```

REPLACE

This DAX function replaces part of a text string, based on the number of characters you specify, with a different text string.

Syntax

```
REPLACE(<old_text>, <start_num>, <num_chars>, <new_text>)
```

Example

```
= REPLACE('New Products'[Product Code],1,2,"OB")
```

SEARCH

This DAX function returns the number of the characters at which a specific text string is first found.

Syntax

SEARCH(<find_text>, <within_text>[, [<start_num>][, <NotFoundValue>]])

Example

= SEARCH("n","printer")

The formula returns 4 because "n" is the fourth character in the word "printer."

UPPER

This DAX function returns a text string in all uppercase letters.

Syntax

UPPER (<text>)

Example

= UPPER(['New Products'[Product Code])

?