MICROSOFT POWER-BI

SERIES 5

DAX
(DATA ANALYSIS EXPRESSION)

- MAYURI .D.

INTRO

Data Analysis Expressions (DAX) is a formula expression language used in Analysis Services, Power BI, and Power Pivot in Excel.
DAX formulas include functions, operators, and values to perform advanced calculations and queries on data in related tables and columns in tabular data models.

DAX formulas are used in measures, calculated columns, calculated tables, and row-level security.

- **Calculated columns** are computed during data load and are stored in the data model, often used for intermediate calculations.

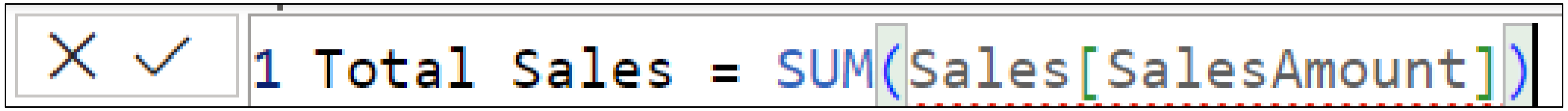
- **Measures** are dynamic calculations evaluated at query time and provide flexibility and efficiency for reporting.

- **Calculated table** is a computed object, based on a formula expression, derived from all or part of other tables in the same model. Instead of querying and loading values into your new table's columns from a data source, a DAX formula defines the table's values.

- With **row-level security**, a DAX formula must evaluate to a Boolean TRUE/FALSE condition, defining which rows can be returned by the results of a query by members of a particular role.

SYNTAX

Syntax includes the various elements that make up a formula, or more simply, how the formula is written.
For example, here's a simple DAX formula for a measure:



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

The measure name, **Total Sales**.
The equals sign operator **(=),** which indicates the beginning of the formula. When calculated, it will return a result.
The DAX function **SUM**, which adds up all of the numbers in the Sales[SalesAmount] column.
Parenthesis **(),** which surround an expression that contains one or more arguments. An argument passes a value to a function.
The referenced table, **Sales**.
The referenced column, **[SalesAmount],** in the Sales table. With this argument, the SUM function knows on which column to aggregate a SUM.

FUNCTIONS

Functions are predefined formulas that perform calculations by using specific values, called arguments, in a particular order or structure.
Arguments can be other functions, another formula, expression, column references, numbers, text, logical values such as TRUE or FALSE, or constants.

DAX includes the following categories of functions:-
Aggregation functions, Date and Time functions, Filter functions, Financial functions, Information functions, Logical functions, Math and trig functions, Other functions, Parent and child functions, Relationship functions, Statistical functions, Table manipulation functions, Text functions, Time intelligence functions.

Some of the common DAX functions—

- **AVERAGE-**- Returns the average (arithmetic mean) of all the numbers in a column.

  syntax-- AVERAGE(<column>)

- **AVERAGEX-**- Calculates the average (arithmetic mean) of a set of expressions evaluated over a table

  syntax-- AVERAGEX(<table>,<expression>)

- **DISTINCTCOUNT**- Counts the number of distinct values in a column

  syntax-- DISTINCTCOUNT(<column>)

- **SUM–** Adds all the numbers in a column
  syntax--   SUM(<column>)

- **SUMX**- Returns the sum of an expression evaluated for each row in a table
  syntax--   SUMX(<table>,<expression>)

- **CALENDAR** Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is from the specified start date to the specified end date, inclusive of those two dates.
  syntax--   CALENDAR(<start_date>, <end_date>)

- **DATEDIFF--** Returns the number of interval boundaries between two dates.
  syntax--   DATEDIFF(<Date1>, <Date2>, <Interval>)

- **NETWORKDAYS–** Returns the number of whole workdays between two dates (inclusive). Parameters specify which and how many days are weekend days. Weekend days and days specified as holidays are not considered as workdays.
  syntax-- NETWORKDAYS(<start_date>, <end_date>[, <weekend>, <holidays>])

- **ALL--** Returns all the rows in a table, or all the values in a column, ignoring any filters that might have been applied. This function is useful for clearing filters and creating calculations on all the rows in a table.
  syntax-- ALL([<table> | <column>[, <column>[, <column>[,…]]]])

- **CALCULATE--** Evaluates an expression in a modified filter context
  syntax-- CALCULATE(<expression>[, <filter1> [, <filter2> [, …]]])

- **SWITCH–** Evaluates an expression against a list of values and returns one of multiple possible result expressions. This function can be used to avoid having multiple nested IF statements.
- syntax-- SWITCH(<expression>, <value>, <result>[, <value>, <result>]…[, <else>])

- **DIVIDE-** Performs division and returns alternate result or BLANK() on division by 0.
  syntax-- DIVIDE(<numerator>, <denominator> [,<alternateresult>])

- **GENERATESERIES-** Returns a single column table containing the values of an arithmetic series, that is, a sequence of values in which each differs from the preceding by a constant quantity. The name of the column returned is Value.
  syntax-- GENERATESERIES(<startValue>, <endValue>[, <incrementValue>])

- **CONCATENATE–** Joins two text strings into one text string.
  syntax-- CONCATENATE(<text1>, <text2>)

- **DATESBETWEEN--** Returns a table that contains a column of dates that begins with a specified start date and continues until a specified end date.
  syntax-- DATESBETWEEN(<Column>, <StartDate>, <EndDate>)

- **TOTALMTD--** Evaluates the value of the expression for the month to date, in the current context.
  syntax-- TOTALMTD(<expression>,<dates>[,<filter>])

- **TOTALQTD--** Evaluates the value of the expression for the dates in the quarter to date, in the current context.
  syntax-- TOTALQTD(<expression>,<dates>[,<filter>])

- **TOTALYTD--** Evaluates the year-to-date value of the expression in the current context

  syntax-- TOTALYTD(<expression>,<dates>[,<filter>],[<year_end_date>])

- **DATESYTD-** Returns a table that contains a column of the dates for the year to date, in the current context.

  syntax-- DATESYTD(<dates> [,<year_end_date>])

- **DATESMTD-** Returns a table that contains a column of the dates for the month to date, in the current context.

  syntax-- DATESMTD(<dates>)

- **DATESQTD-** Returns a table that contains a column of the dates for the quarter to date, in the current context.

  syntax-- DATESQTD(<dates>)

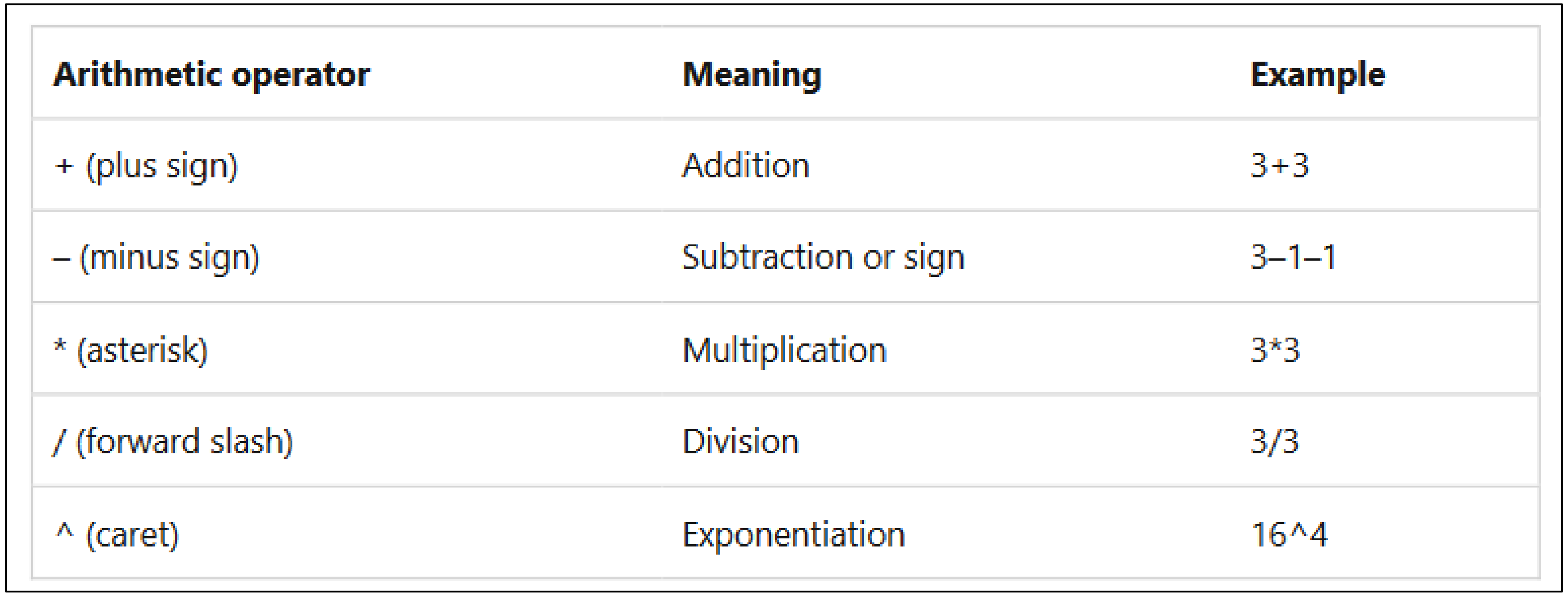
DAX OPERATORS

The DAX language uses operators to create expressions that compare values, perform arithmetic calculations, or work with strings.

There are four different types of calculation operators: -
Arithmetic operator,
Comparison operator,
Text concatenation operator, and
Logical operator.

ARITHMETIC OPERATORS

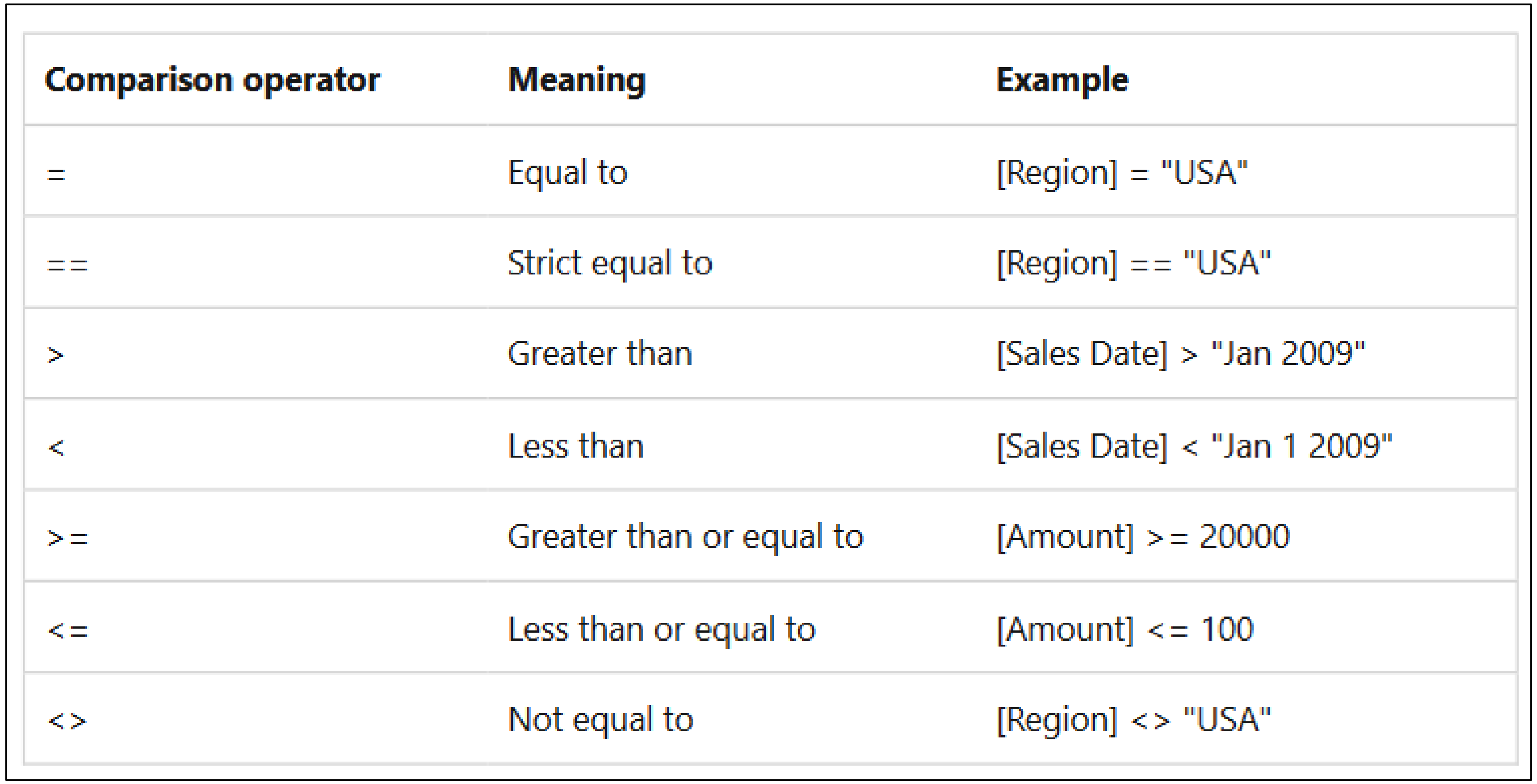
To perform basic mathematical operations such as addition, subtraction, or multiplication; combine numbers; and produce numeric results, use the following arithmetic operators



Arithmetic operator	Meaning	Example
+ (plus sign)	Addition	3+3
– (minus sign)	Subtraction or sign	3–1–1
* (asterisk)	Multiplication	3*3
/ (forward slash)	Division	3/3
^ (caret)	Exponentiation	16^4

COMPARISON OPERATORS

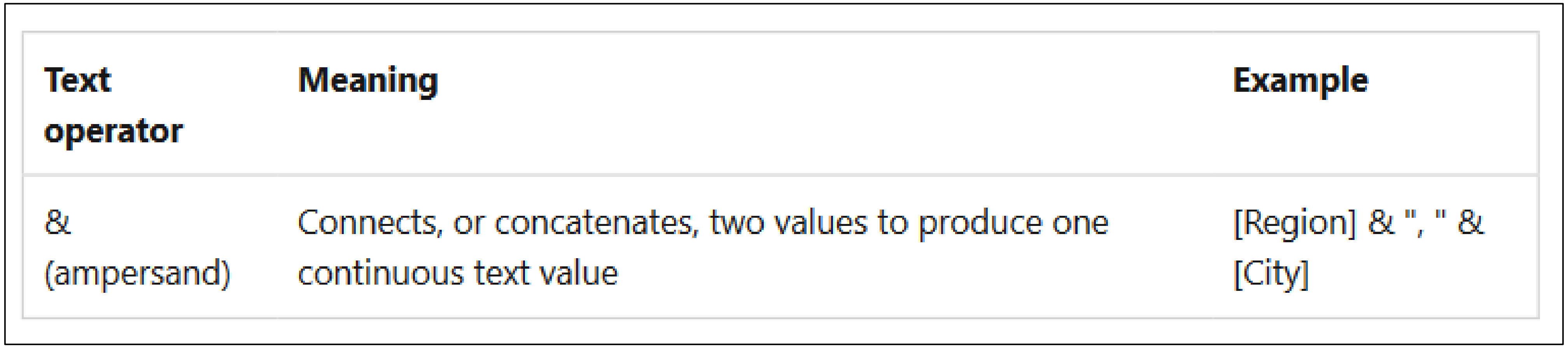
You can compare two values with the following operators. When two values are compared by using these operators, the result is a logical value, either TRUE or FALSE



Comparison operator	Meaning	Example
=	Equal to	[Region] = "USA"
==	Strict equal to	[Region] == "USA"
>	Greater than	[Sales Date] > "Jan 2009"
<	Less than	[Sales Date] < "Jan 1 2009"
>=	Greater than or equal to	[Amount] >= 20000
<=	Less than or equal to	[Amount] <= 100
<>	Not equal to	[Region] <> "USA"

TEXT CONCATENATION OPERATOR

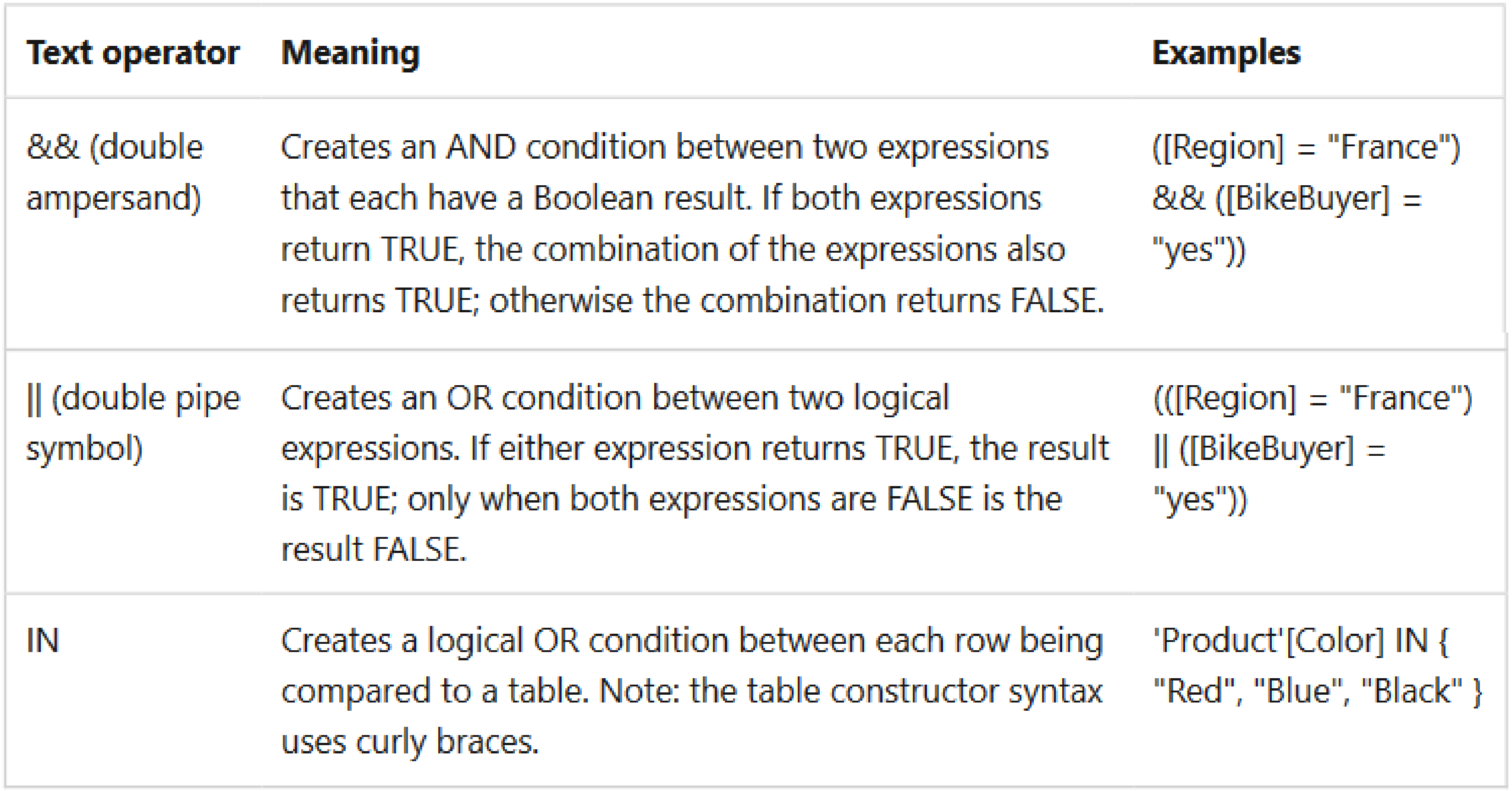
Use the ampersand (&) to join, or concatenate, two or more text strings to produce a single piece of text.



Text operator	Meaning	Example
& (ampersand)	Connects, or concatenates, two values to produce one continuous text value	[Region] & ", " & [City]

LOGICAL OPERATORS

Use logical operators (&&) and (||) to combine expressions to produce a single result.



Text operator	Meaning	Examples
&& (double ampersand)	Creates an AND condition between two expressions that each have a Boolean result. If both expressions return TRUE, the combination of the expressions also returns TRUE; otherwise the combination returns FALSE.	([Region] = "France") && ([BikeBuyer] = "yes"))
|| (double pipe symbol)	Creates an OR condition between two logical expressions. If either expression returns TRUE, the result is TRUE; only when both expressions are FALSE is the result FALSE.	(([Region] = "France") || ([BikeBuyer] = "yes"))
IN	Creates a logical OR condition between each row being compared to a table. Note: the table constructor syntax uses curly braces.	'Product'[Color] IN { "Red", "Blue", "Black" }

ERROR HANDLING IN DAX

The **BLANK** function represents missing or undefined values in DAX calculations.

ISBLANK checks if a value is blank and returns TRUE if it is, otherwise FALSE.

IFERROR provides a way to handle errors by returning a specified value if an error occurs in a calculation

FORMATING DAX FORMULAS

- Proper formatting improves the readability and maintainability of DAX code, making it easier to understand and debug.
- Use indentation and line breaks to separate different parts of the formula clearly.
- Add comments to explain complex logic and document your thought process.
- Consistent formatting practices help in collaborative environments and future proof your reports.

PRACTICAL IMPLEMENTATION OF DAX ON ECOMM SALES ANALYSIS PROJECT

AIM OF THE PROJECT—

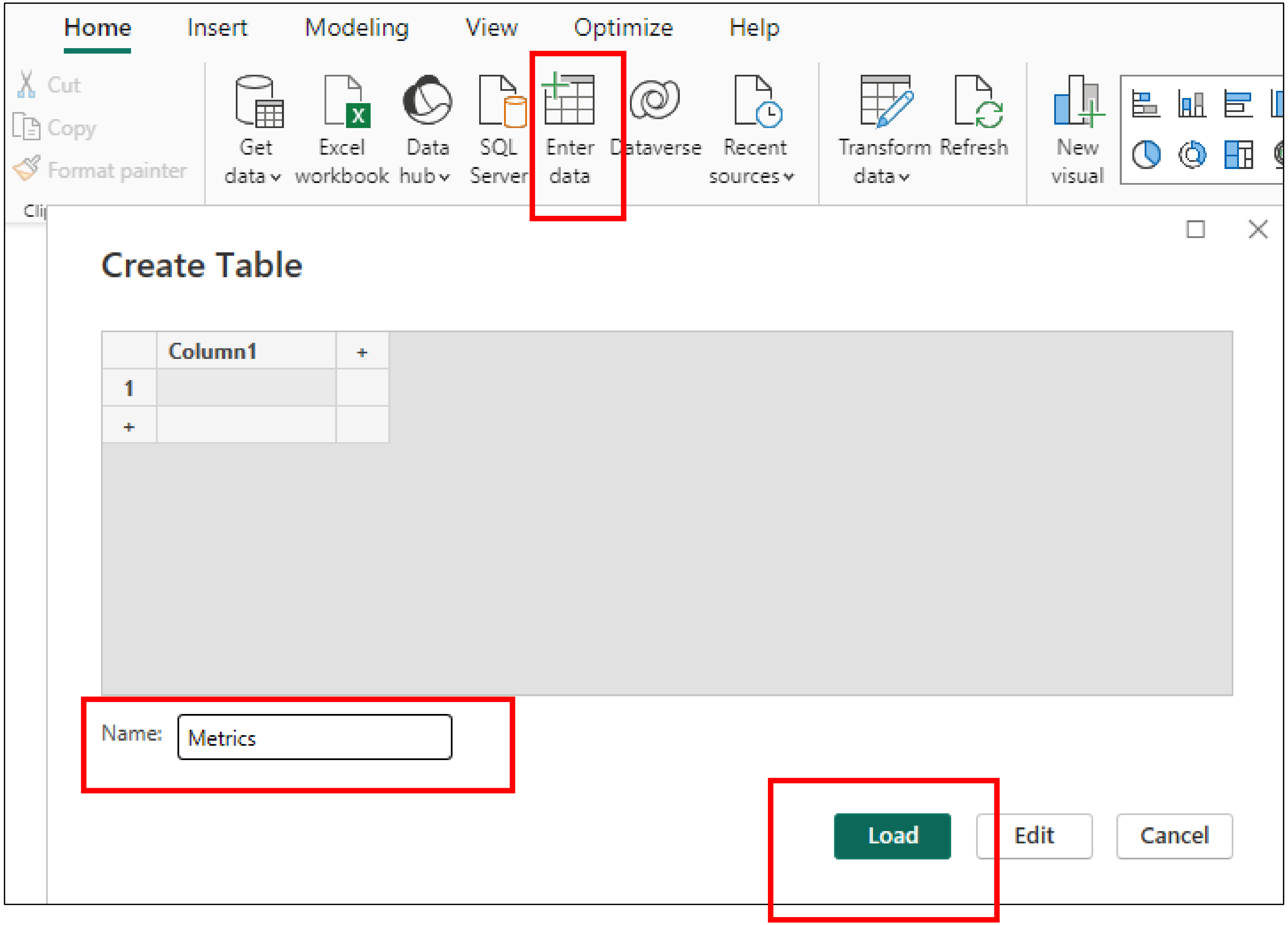
The primary goal is to provide actionable insights into the e-commerce landscape, pinpointing areas for enhancement and growth through comprehensive analysis.

On having a look at aim of project and available dataset structure, you can decide the primary columns or measures needed during visualization process.

Here, you can create basic measures like –
(YTD= YearToDate, YOY= YearOverYear, PYTD= PreviousYTD)
YTD sales, YTD Qty, YTD Profit, YTD Profit margin
YOY Sales, YOY Qty, YOY Profit, YOY Profit margin
To get YOY, you will need--
PYTD Sales, PYTD Qty, PYTD Profit, PYTD Profit margin

First of all, create a separate table for measures, so that it won’t get confusing in the dataset tables. You can create them in existing tables itself.

Click on “Enter Data” under Home tab.
Name the table
Click Load
On right side in DATA panel, the new table is displayed.



Create Table

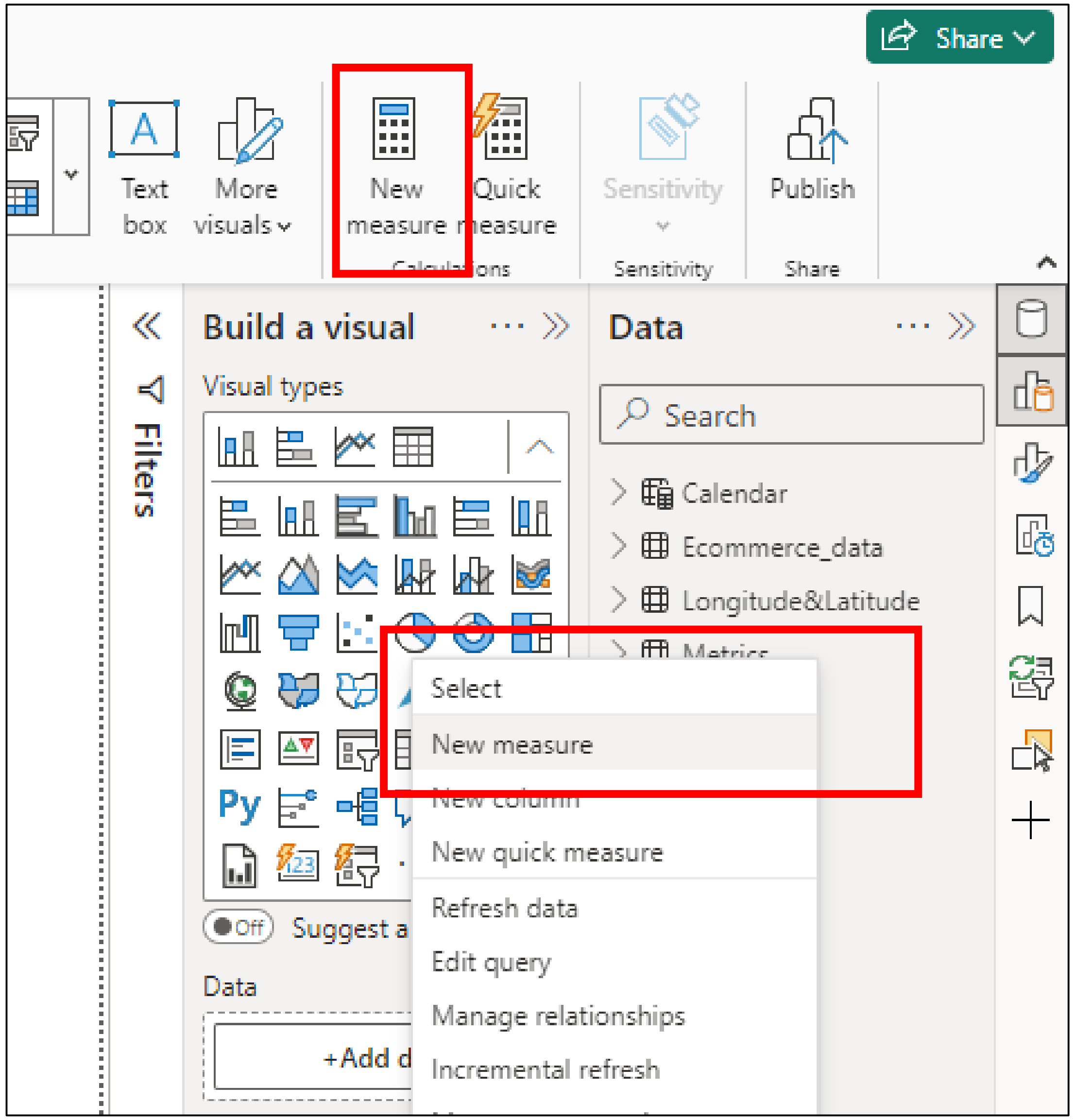
	Column1	+
1		
+		

Name: Metrics

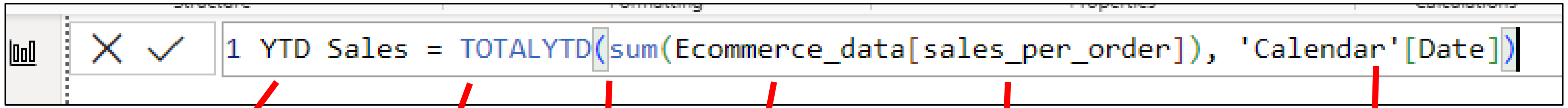
To create 1st measure right click on new table → create measure

OR

Click on “New Measure” under Home tab



Write following query and press Enter



```
YTD Sales = TOTALYTD(sum(Ecommerce_data[sales_per_order]), 'Calendar'[Date])
```

Measure name

Specify function

Aggregate function to perform on column

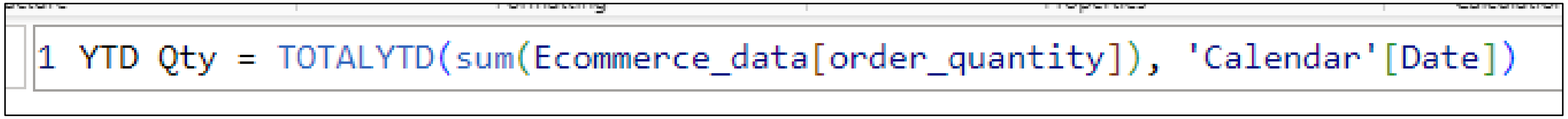
Referenced Table name

Referenced Column name

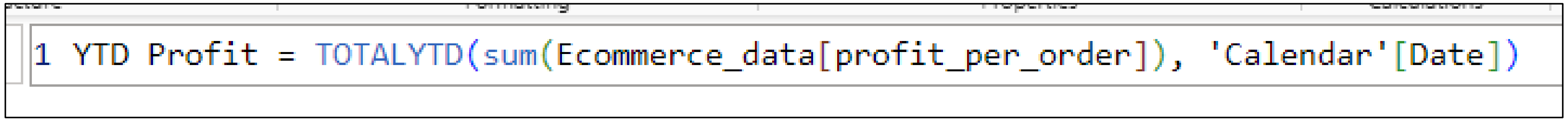
Date for YTD

(First, it calculates the sum of the sales per order from the `ecmm_data` table.
Then, it takes this sum and computes the year-to-date total using the dates provided in the `Calendar[Date]` column.)

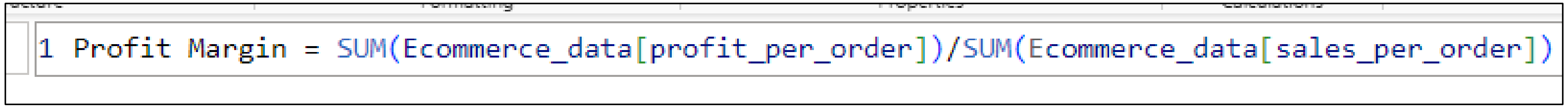
Following same process write other queries.



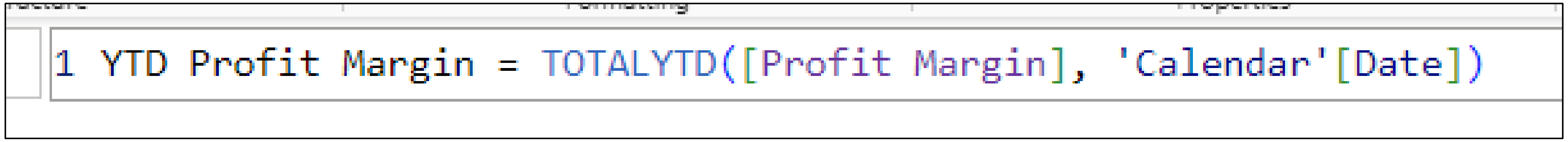
```
YTD Qty = TOTALYTD(sum(Ecommerce_data[order_quantity]), 'Calendar'[Date])
```



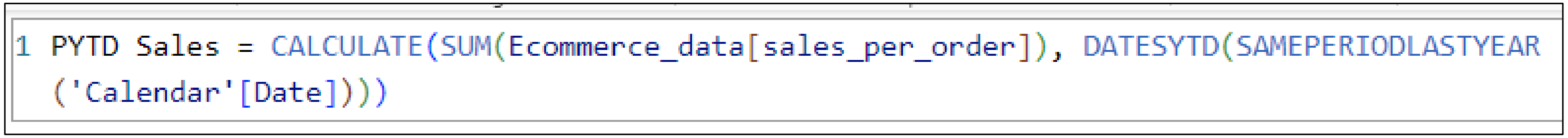
```
YTD Profit = TOTALYTD(sum(Ecommerce_data[profit_per_order]), 'Calendar'[Date])
```



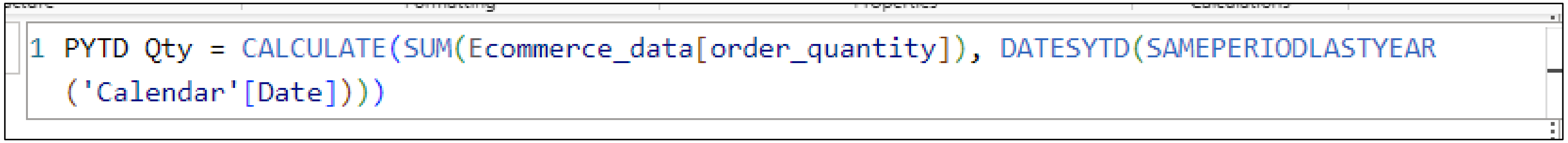
```
Profit Margin = SUM(Ecommerce_data[profit_per_order])/SUM(Ecommerce_data[sales_per_order])
```



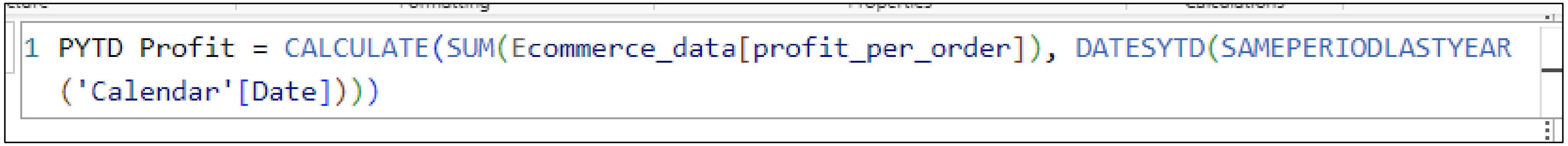
```
YTD Profit Margin = TOTALYTD([Profit Margin], 'Calendar'[Date])
```



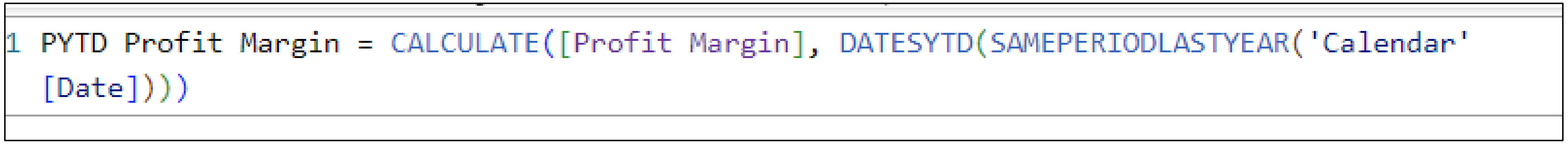
```
PYTD Sales = CALCULATE(SUM(Ecommerce_data[sales_per_order]), DATESYTD(SAMEPERIODLASTYEAR
    ('Calendar'[Date])))
```



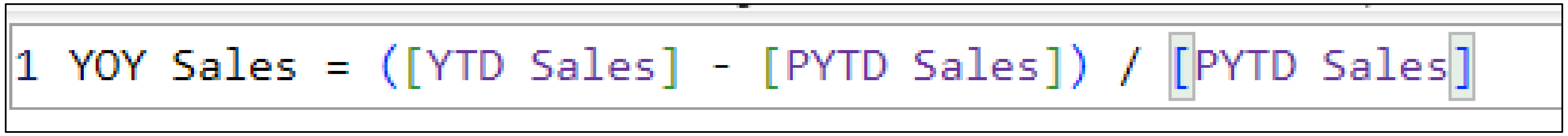
```
PYTD Qty = CALCULATE(SUM(Ecommerce_data[order_quantity]), DATESYTD(SAMEPERIODLASTYEAR
    ('Calendar'[Date])))
```



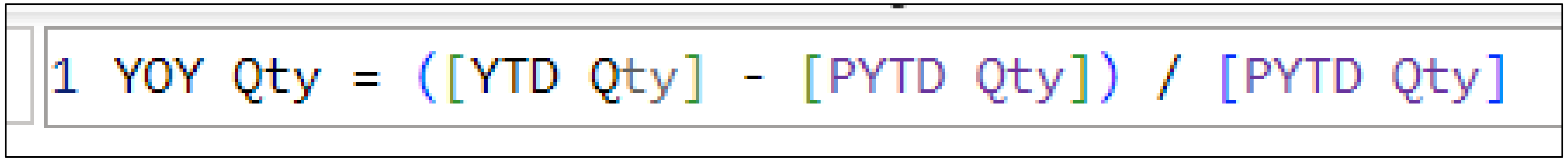
```
PYTD Profit = CALCULATE(SUM(Ecommerce_data[profit_per_order]), DATESYTD(SAMEPERIODLASTYEAR
    ('Calendar'[Date])))
```



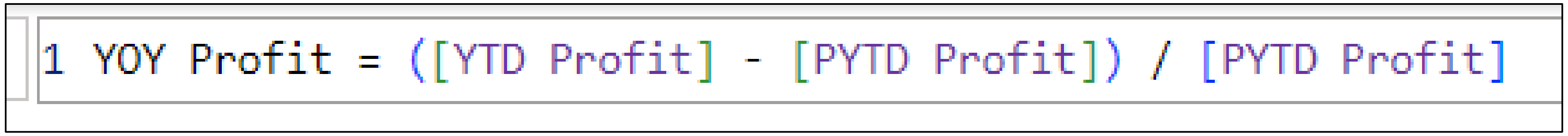
```
PYTD Profit Margin = CALCULATE([Profit Margin], DATESYTD(SAMEPERIODLASTYEAR('Calendar'
    [Date])))
```



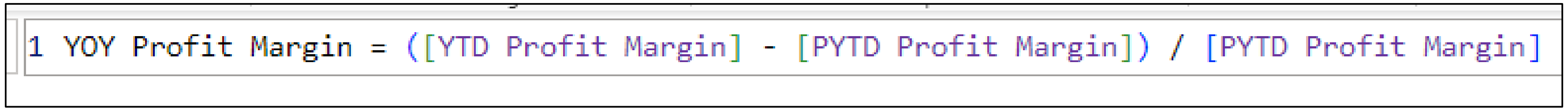
```
YOY Sales = ([YTD Sales] - [PYTD Sales]) / [PYTD Sales]
```



```
YOY Qty = ([YTD Qty] - [PYTD Qty]) / [PYTD Qty]
```



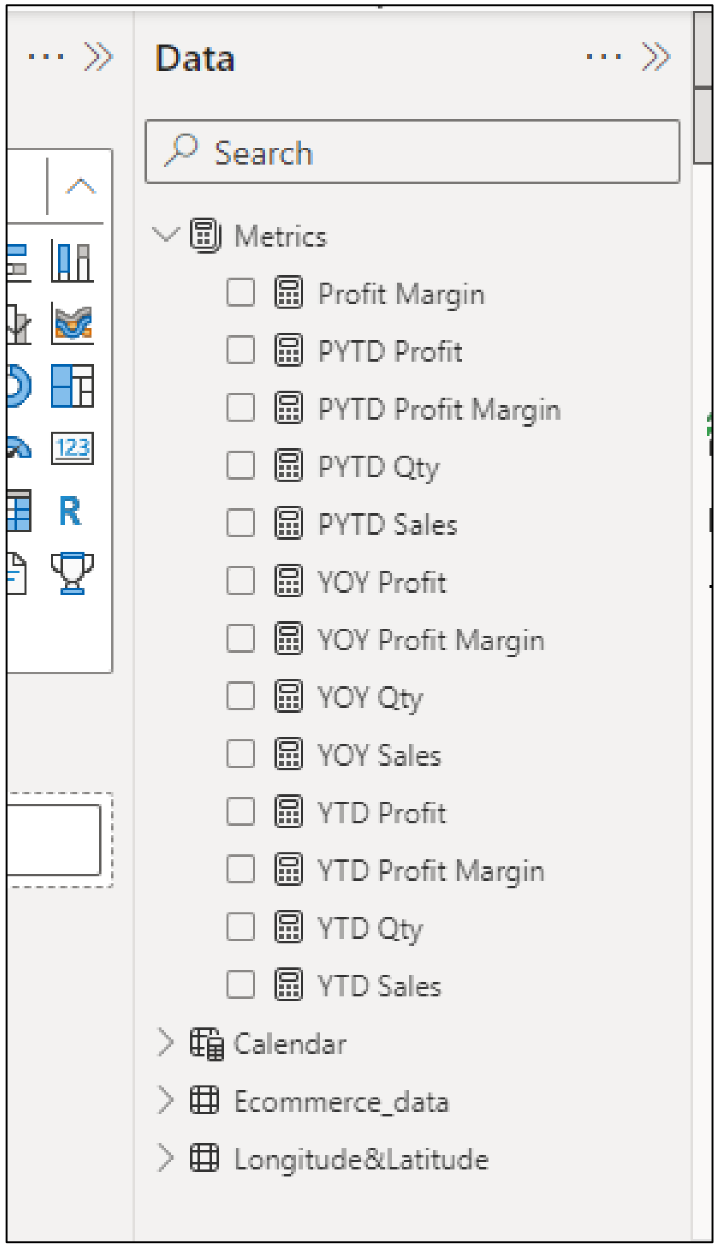
```
YOY Profit = ([YTD Profit] - [PYTD Profit]) / [PYTD Profit]
```



```
YOY Profit Margin = ([YTD Profit Margin] - [PYTD Profit Margin]) / [PYTD Profit Margin]
```

List of primary measures created.

Other measures can be created as per requirement during visualization process.



Data

Search

Metrics
- Profit Margin
- PYTD Profit
- PYTD Profit Margin
- PYTD Qty
- PYTD Sales
- YOY Profit
- YOY Profit Margin
- YOY Qty
- YOY Sales
- YTD Profit
- YTD Profit Margin
- YTD Qty
- YTD Sales

Calendar

Ecommerce_data

Longitude&Latitude

THANK YOU

- MAYURI .D.