DAX (Data Analytics Expression)

DAX is a powerful formula language, because it specifically designed for creating formulas and expressions that operates the data.

You can perform wide range of analysis and reporting with the DAX.

Karthimeena

DAX Data types

- Numeric
- Text
- Date and Time
- Boolean
- Currency
- Percentage
- Duration
- Binary
- Variant
- Table type
- Other

Karthimeena

What type of calculations can you perform with DAX

1. Calculated Columns
2. Calculated Tables
3. Measures

Calculated Columns

- By writing DAX Formula, you can add a new column to the table.
- Your formula determines whether it duplicates the exiting column from a table or generating a new series of values.
- Formula should return a single column with list of values
- You can create this in report view, model view and table view

```
Sales_Amount = Sales[Unit Price] * Sales[Order Quantity]
```

DAX formula creating a simple sales calculations and returns a series of sales values

Karthimeena

Calculated Columns

- Calculated columns create additional data in the data model, so this increases the model size if there are many calculated columns.
- Calculations are performed on row-by-row basis.
- Calculated column values are refreshed during the entire model's refreshed time if the model uses import mode.
- If the model uses direct query, the formula is not calculated within the model. Instead, the formula pushed to data source and the calculation performed at query time in the database.

Calculated Columns

- Few Interview Questions.
  - How does calculated column differ from other columns ?
  - When and how calculated columns are refreshed ?
  - When would you choose calculated columns instead of measures ?
  - Pros and cons of calculated columns ?

Karthimeena

Calculated Tables

- By writing DAX Formula, you can add a new table to your model.
- Your formula determines whether it duplicates the exiting table or generating a new series of values for calculated table.
- Your formula should return a table instead of returning a single value.
- You can create this in report view, model view and table view
- Calculated tables are recalculated, if any of the tables are refreshed or updated.

Karthimeena

Calculated Tables

```
fact_summarize_date = SUMMARIZE(
                        'fact_premiums', fact_premiums[date],
                        fact_premiums[final_premium_amt(INR)])
```

- DAX formula was created to summarize the ‘fact_premiums’ table by date. This returns a table of summarized values
- This always imported into your model, so this increases your model size and can prolong your data refresh time.
- Just like other tables, calculated tables have a relationship with other tables.

Karthimeena

Calculated Tables

- Few Interview Questions.
  - How does calculated table differ from other physical tables ?
  - How do you create calculated tables?
  - In which scenarios would you choose calculated tables?
  - Pros and cons of calculated tables ?

Karthimeena

Measures

- Measure is a calculated value based on the data in your model
- Measures are evaluated at query time and it takes memory for DAX formula and not the results.
- There are two types of measures
  - Implicit Measure
  - Explicit Measure

Measures

- Implicit Measures
  - This is automatically created by power bi when you drag and drop your numerical values into visual area such as metrics, tables.
  - You can have flexibility to change the summarization method of applied implicit measures.

Measures

- Explicit Measures
  - These are created by the developers to perform specific calculation as per the need.
  - By writing DAX Formula, to create a measure to your model and must return single value.
  - You can organize all your measures in display folders for better readability.

Measures

- Few Interview Questions.
  - How does measure differ from calculated column ?
  - How do you create implicit/explicit measures ?
  - What type of calculations can you perform with measures ?
  - How do you handle errors in DAX measures ?
  - What strategies can you use to optimize the performance of measures ?

Karthimeena

CALCULATE

- CALCULATE is primarily used function to modify the context in which a calculation is performed and this will allow you to apply filters on the calculations.

```
Weekday_Sale =
var revenue = CALCULATE(
                SUM[Revenue],                    ← Evaluating sum of revenue
                dim_date[month]= 'Febrarury',    ← Filter 1
                dim_date[week_type] = "Weekday") ← Filter 2
return revenue                                   ← Return statement
```

CALCULATE

- The code creates a measure to our model
- Code calculates the revenue amount for current month and weekday
- for e.g.: now we are in February, formula returns the revenue for weekday's in February's and excludes weekend.
- Used VARIABLES for better readability and debugging.
- SUM is an aggregate function used in the code, which is default function exists in DAX

Karthimeena

CALCULATE

- Do comment if you know any other ways to achieve the sales that occurred on weekdays within the current month.

In Day 6, I will share mine.

Karthimeena

CALCULATE

Here is other way to achieve the same weekday sales with CALCULATE function.

- Total_Revenue = SUM(fact_premiums[final_premium_amt(INR)])
- Above code creates a measure by aggregating ‘final_premium_amount’
- One can use this measure in multiple places, reducing the need duplicate the same calculation

Karthimeena

CALCULATE

```
Weekday_Sale =
var revenue = CALCULATE(
                [Total_Revenue],
Filter 1 ---> FILTER(dim_date, dim_date[month]= 'Febrarury' &&
Filter 2 ---> dim_date[week_type] = "Weekday"))
return revenue
```

Measure

- Used FILTER functions to achieve the revenue of current month's weekday sale

Karthimeena

CALCULATE

- Here is one more way to achieve the same weekday sales with CALCULATE function

```
Weekday_Sale =
var revenue = CALCULATE(
                [Total_Revenue],
Filter 1 ---->  dim_date[week_type] = "Weekday“,
Filter 2 ---->  VALUES(dim_date[month]))
return revenue
```

Explicit Measure

Karthimeena

VALUES

- Formula used new function which is VALUES.
- Instead of using a variable to set the current month, we apply context of current month using VALUES function.
- This calculation ensures, it is performed for weekdays with in the current month

Karthimeena

CALCULATETABLE

- It is used to change the filter context of an entire table rather then single expression.
- If there is a need to generate a table instead of scalar value, choose this CALCULATETABLE
- Syntax is similar to CALCULATE.
- This function is not supported for direct query mode when used in calculated columns or row-level-security.

Karthimeena

CALCULATETABLE

```
Online_Sale=
var Sales = CALCULATETABLE(
        fact_premiums,
        fact_premiums[sales_mode] = "Online-App"
        )
return Sales
```

Table name

Filter 1

- Above code filters the table “fact_premiums” with the sales_mode is Online_App. The result of the above code stores in a new table “Online_Sales”.
- Structure of the new table(online_sale) is same like original table(fact_premiums)

Karthimeena

CALCULATETABLE

Considerable facts when use CALCULATETABLE

- Cannot be applied in measures like filter functions
- You need enough memory to use calculate tables.
- Calculated tables increase the processing time in DAX.
- Calculated tables are static and cannot modified dynamically based on user interactions.
- There is risk of data redundancy or duplication of data based on how calculated tables are implemented.

Karthimeena

CALCULATETABLE

Here will see an other scenario in CALCULATETABLE.

- Create a table that combines data from two related tables

```
weekend_table = CALCULATETABLE(
                    fact_premiums,          ← Table 1
                    RELATEDTABLE(dim_date),  ← Table 2
                    dim_date[week_type] = "Weekend")  ← Filter 1
```

- Above code filters the fact_premiums table for payments made on week ends.
- dim_date and fact_premiumns are having single directional relationship (one-to-many)

Karthimeena

CALCULATE VS CALCULATETABLE

CALCULATE	CALCULATETABLE
Evaluates an expression in a context modified by filter	Evaluates an table expression in a context modified by filters.
	If you use this function within the DAX measure, it is going to return virtual table
This allocates less space in mode	This allocates more memory in your model
It operates at row-level, allowing you to modify the filter context for each individual calculation	This works at table level and can be more efficient when applying filters to entire table

Karthimeena

RELATED

- RELATED, which is used to retrieve a related value from another table and it returns a scalar value.
- This requires establishing a relationship between current table and the table with related information that you need.
- Syntax is simple with single parameter, that specifies the column that contains the value you want to retrieve.
- It is used in measures, calculated columns to get the related value of current context.
- This won’t work between the tables which is not having any relationship.

Karthimeena

RELATED

- Here, we have two tables Product (dimension)and Sales(fact)
- They have established one-to-many relationship.
- Calculate the sales by each product.

```
Sales_by_Product = SUMX(
    Sales,
    RELATED('Product'[List Price]) *
    Sales[Order Quantity])
```

Related Table

Table 1

Column name

Karthimeena

RELATEDTABLE

- RELATEDTABLE is a function used to retrieve a table that related to current row in the data model
- It doesn’t retrieve individual values directly, instead of return the entire related table
- When data source is Direct Query, It doesn’t work in creating calculated columns or row-level security.

Karthimeena

RELATEDTABLE

Here is a scenario to calculate total number of sales for top products by revenue.

- Here, we have two tables Product (dimension)and Sales(fact)
- ‘Revenue’ is a column in Sales table
- In code, First calculating the top products by their sales from the sales table.
- Then, we are counting number of sales occurred for top products.

Karthimeena

RELATEDTABLE

```
Top_Products_Sale =
var Top_Products = TOPN(3, 'Product', Sales[Revenue])
return
SUMX(
        Top_Products,
        COUNTROWS(RELATEDTABLE(Sales)))
```

Finding top 3 products by the revenue

Related Table

- RELATEDTABLE(Sales) retrieves the sales data related to the Top_Products.
- COUNTROWS, count the number of rows

Karthimeena

RELATED vs RELATEDTABLE

RELATED	RELATEDTABLE
RELATED() is a function, used to retrieve a related value from another table	RELATEDTABLE() function used to get table of related rows from another table
It works with in the context of established relationship between the tables	It works with in the context of established relationship between the tables
Typically used in calculated columns and measures to fetch a single related value	Typically used in calculated columns and measures to retrieve a table of related rows
Can fetch scalar value for aggregation or comparison	Often used for aggregations of filtering operations

Karthimeena

Related vs Relatedtable

Few Interview Questions.

1. Purpose of RELATED() and RELATEDTABLE() ?
2. Return type of RELATED() ?
3. When would you be use RELATEDTABLE() function ?
4. Explain the differences between RELATED() and RELATEDTABLE()
5. In which scenario would you prefer to use RELATED() over RELATEDTABLE()

Karthimeena

FILTER

- FILTER() is a function used to filter the rows or columns from a table based on specific conditions.
- It returns a table of rows instead of scalar values
- Filter can be used in various scenarios such as calculating columns, performing a conditional aggregation, measures.
- Filter() is commonly used with other DAX functions such as CALCULATE, SUMMARIZE and more
- This function not supported to use in Direct Query when used in calculated columns or RLS rules.

Karthimeena

FILTER

```
Revenue_by_Terrirory = CALCULATE(
                        [Revenue],
                        FILTER('Sales Territory',
                        'Sales Territory'[Country] = "United States"))
```

Filtering table

Filter condition

- Above measure returns the ‘Revenue’ for the country United States.
- The measure applies the filter condition to the ‘Sales Territory’ table

Karthimeena

REMOVE FILTERS

- REMOVEFILTERS() removes all applied filters to the table, it restores the original state of the table without any filters.
- This function useful when you want calculate any measures or calculation is evaluated without any filters.
- This won't return any value or a table.
- This function not supported to use in Direct Query when used in calculated columns or RLS rules.

```
Revenue_withoutFilter = CALCULATE([Revenue],
                                  REMOVEFILTERS())
```

Removing all

Karthimeena

KEEP FILTERS

- KEEPFILTERS() essentially opposite to REMOVEFILTERS()
- This function useful to retain the existing filters in a column or table.
- This function not supported to use in Direct Query when used in calculated columns or RLS rules.

Karthimeena

KEEP FILTERS

Here is a scenario to find out the revenue by territory for ‘United States’ and their revenue should be greater than 1000.

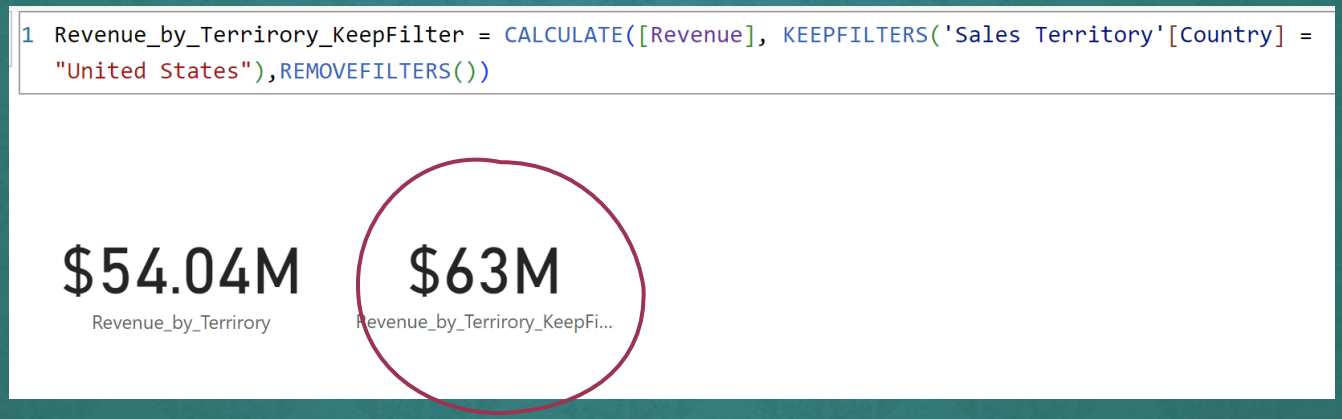
Code applied those two filters and shows the revenue in card.

Karthimeena

KEEP FILTERS

Here is a scenario to find out the revenue by territory only for ‘United States’

```
1  Revenue_by_Terrirory_KeepFilter = CALCULATE([Revenue], KEEPFILTERS('Sales Territory'[Country] =
   "United States"),REMOVEFILTERS())
```



- The code uses KEEPFILTER for existing filters and remove all other filters.
- It indicates to others reading your code, that specifically you want to retain the few existing filters

Karthimeena

FILTERS

Few Interview Questions

1. Purpose of using FILTERS() in dax ?
2. Differences between KEEPFILTERS(), FILTERS() ?
3. When would you use REMOVEFILTERS() ?
4. How does KEEPFILTERS() interacts with existing filter context in DAX ?

Karthimeena

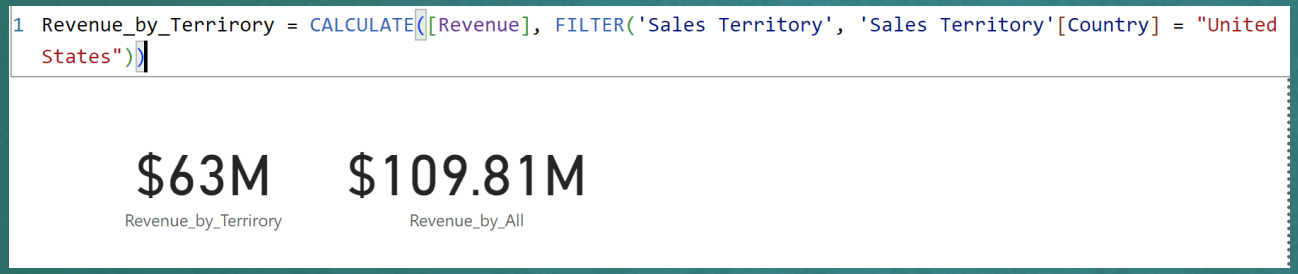
ALL

- ALL is a function which simply return all the rows from a table
- It ignores any filters that might have been applied to a table.
- You cannot use any table or column expression with in the ALL function

ALL

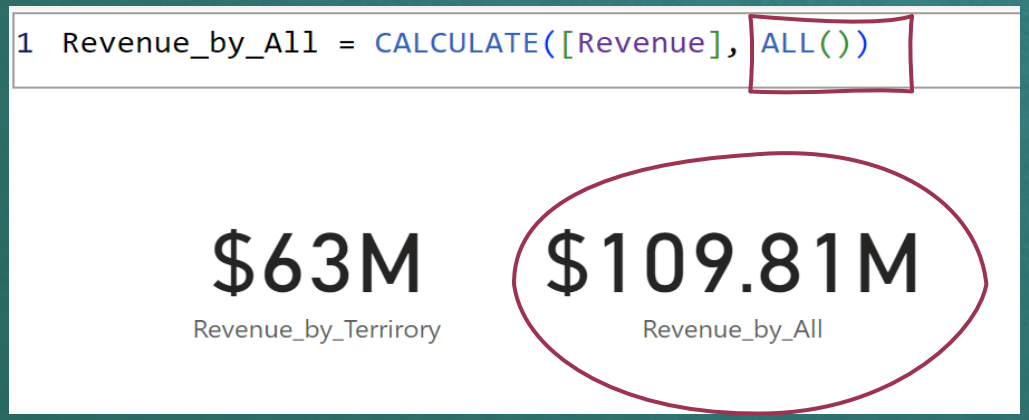
- ALL() - It is only be used to clear filters but not return a table.

```
1  Revenue_by_Terrirory = CALCULATE([Revenue], FILTER('Sales Territory', 'Sales Territory'[Country] = "United States"))
```



Calculating Revenue for United States

```
1  Revenue_by_All = CALCULATE([Revenue], ALL())
```



Calculating Revenue for All.
ALL() removes all the filters on the table in the current context

Karthimeena

ALL

- ALL(table)
  - Removes all the filters from the specified table and returns all the rows from te table.

```
1  Revenue_by_All = CALCULATE([Revenue], ALL('Sales Territory'))
```

$63M
Revenue_by_Terrirory

$109.81M
Revenue_by_All

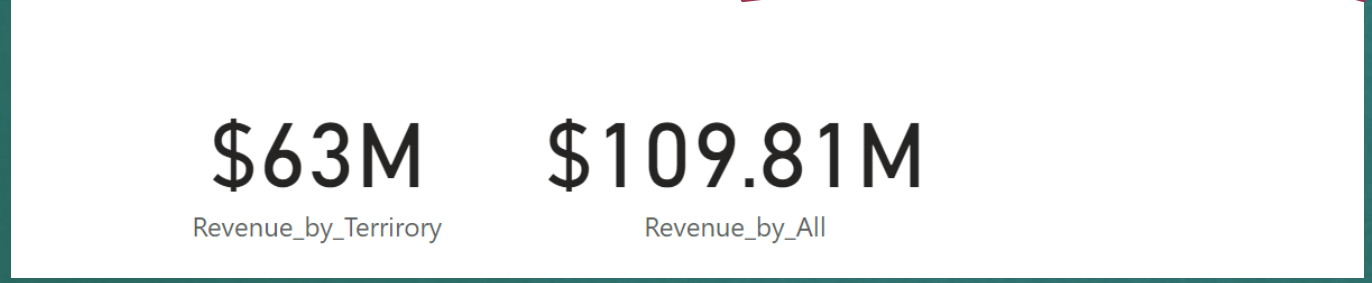
Calculating Revenue for All.
It removes all the filters on the specified table.

Karthimeena

ALL

- ALL([column])
  - Removes all the filters from the specified column in the table.

```
1  Revenue_by_All = CALCULATE([Revenue], ALL('Sales Territory'[Country]))
```



Calculating Revenue for All.
It removes all the filters on the specified column.

Karthimeena

ALLEXCEPT

- ALLEXCEPT also used to remove all the filters but except filters that have been applied to the specified columns.
- It accepts table or column as a parameter

```
1  Revenue_by_All =
2  VAR Revenue_UnitedStates =  CALCULATE([Revenue], FILTER('Sales Territory', 'Sales Territory'[Country] =
   "United States"), FILTER('Sales Order', 'Sales Order'[Channel] = "Internet"))
3  VAR Reveue_by_allexcept = CALCULATE([Revenue], ALLEXCEPT('Sales Order', 'Sales Order'[Channel]))
4  RETURN Reveue_by_allexcept
```

- It keeps the filter on ‘Channel’ from the sales order table and removes all other filters.

Karthimeena

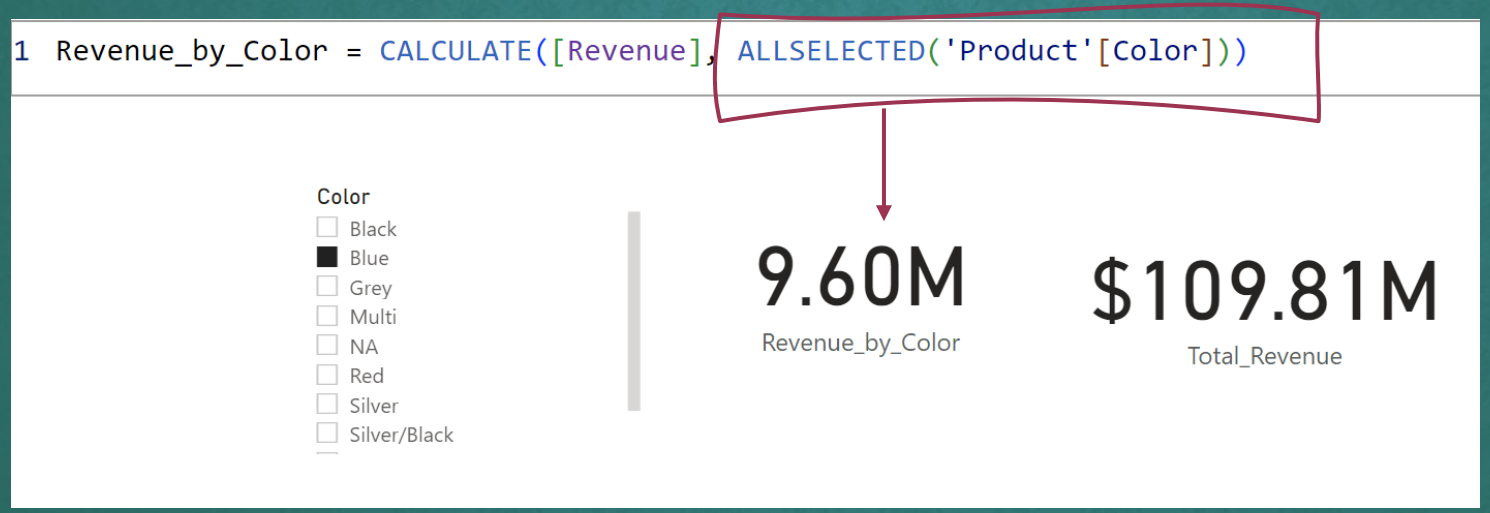
ALLSELECTED

- ALL SELECTED is a function, returns all the rows and columns mentioned in the context disregarding any filters that might have been applied to a table.
- Argument is either table name or column name.

Karthimeena

ALLSELECTED

- Calculating revenue only for the selected color disregarding filters applied on the table



- Total_Revenue uses ALL to remove all the filters applied on the table

Karthimeena

LOOKUPVALUE

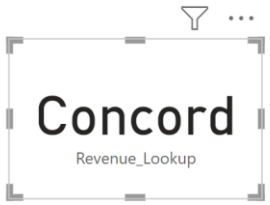
- LOOKUPVALUE is useful to retrieve the value of a specified column in the table based on one or more conditions.
- This will return a BLANK value if no match found based on specified condition and no alternate value provided.
- If multiple rows matches the search value and the column values are identical, then that values are returned.
- It throws an error or alternate result if column values are not identical.

Karthimeena

LOOKUPVALUE

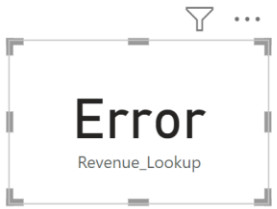
- This returns the customer's city based on postal code

```
1  Revenue_Lookup = LOOKUPVALUE(Customer[City], Customer[Postal Code], "94519", "Error")
```



- This code throws alternate result, since it doesn't meet the criteria

```
1  Revenue_Lookup = LOOKUPVALUE(Customer[City], Customer[Postal Code], "94591", "Error")
```



Karthimeena

SUM

- It adds all the values in a column and the column should be numeric type.

Karthimeena

SUMX

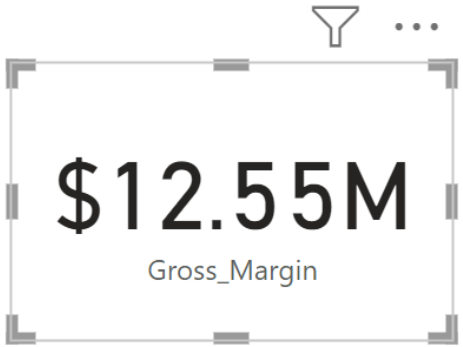
- This useful to performing calculations based on applied expressions on a row-by-row basis.
- It accepts table as first argument and second argument is a column that contains a value you want to sum.
- It returns a decimal number.
- This function ignores if there is BLANK and logical values while adding numbers.

Karthimeena

SUMX

- This function returns the gross margin amount by calculating an expression on a row-by-row basis.

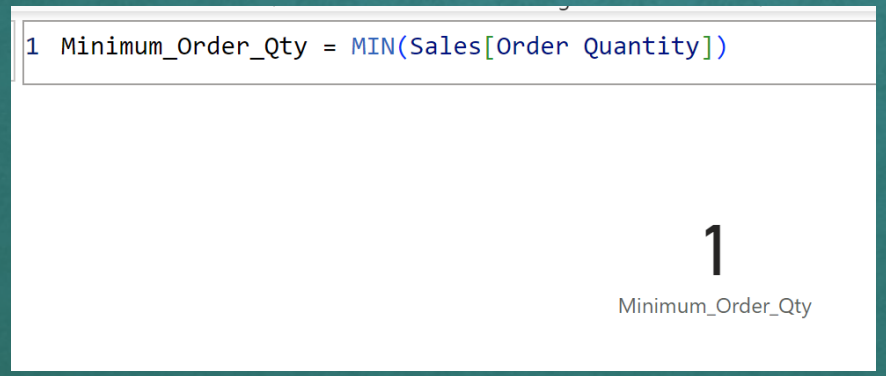
```
1  Gross_Margin = SUMX(Sales, Sales[Sales Amount] - Sales[Total Product Cost])
```



Karthimeena

MIN

- This function is helpful for finding the minimum value of a given column.



Karthimeena

MINX

- This calculates the minimum value based on some filters based on some expression for each row in table.
- It accepts table as first argument, expression or filters as second argument.

```
1  Minimum_Order_Qty = MINX(Sales, Sales[ProductKey])
```

212

Minimum_Order_Qty

Karthimeena

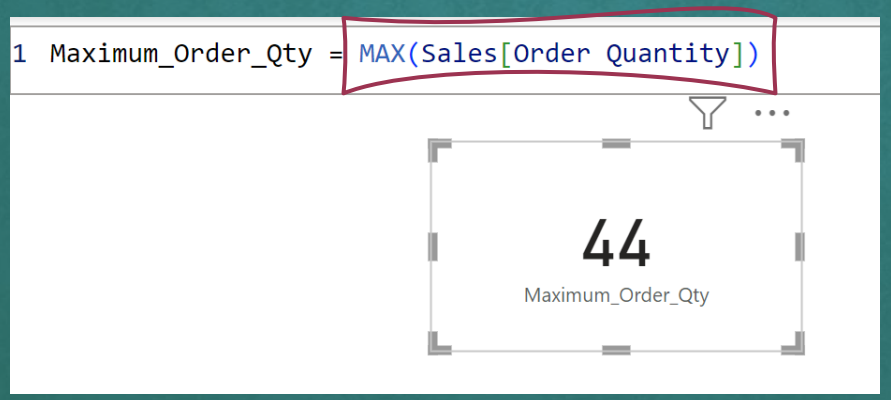
MINA

- This function performs as MIN with added feature to handle the non-numeric columns
- This calculates the minimum value of the column including number, text and date.
- Minimum_Order_Qty = MINA('Sales Order'[Sales Order Line])
- This returns value 0, since the data type of the column is TEXT, it does not find any value

Karthimeena

MAX

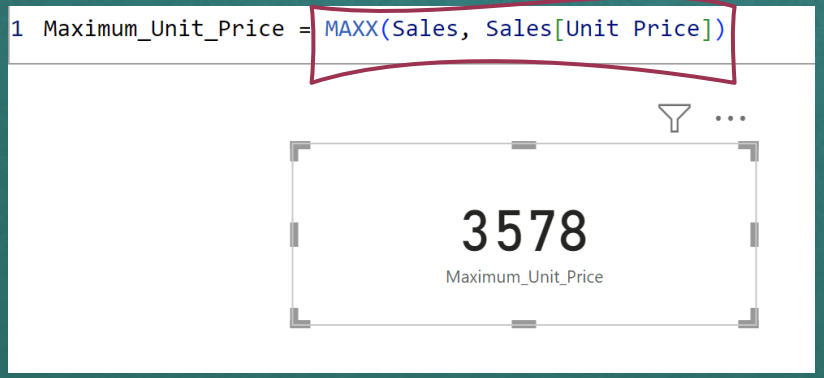
- This function is helpful for finding the greatest value of a given column.



Karthimeena

MAXX

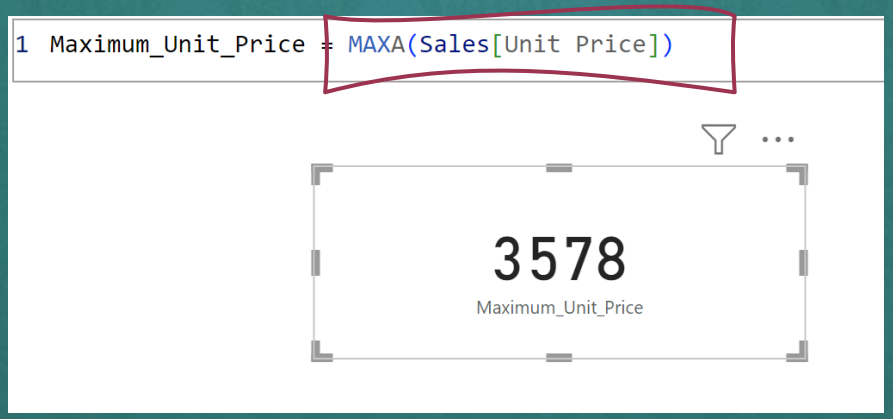
- This function calculates the maximum values of a given column based on expression for each row in a table.
- It accepts table as first argument, expression or filters as second argument.
- This evaluates number, date and Text.



Karthimeena

MAXA

- This function finds largest values of the column.
- It handles the non-numeric columns such as number and date.
- This calculates the maximum value of the column including number, text and date.

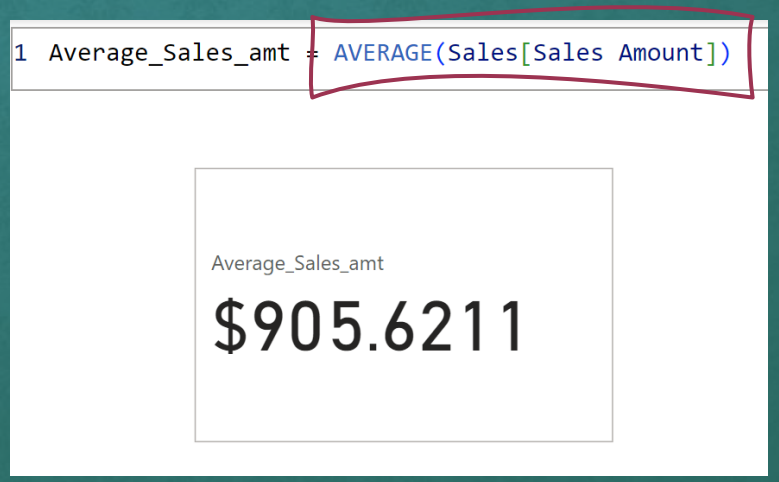


- MAXA() returns value 0, if the data type of the column is TEXT, it does not find any value

Karthimeena

AVERAGE

- This function calculates the arithmetic average operation of all the numbers in a column and returns the result.
- If the column contains value 0 are included.
- If the column contains TEXT, this won't perform operation and just returns the blank value.



Karthimeena

AVERAGEX

- If you want to perform average operation for a set of numbers instead of entire column in table, this AVERAGEX will do the same.
- Function takes table as first argument and expression as second argument.
- You cannot include non-numeric values and both arguments are required.

Karthimeena

AVERAGEX

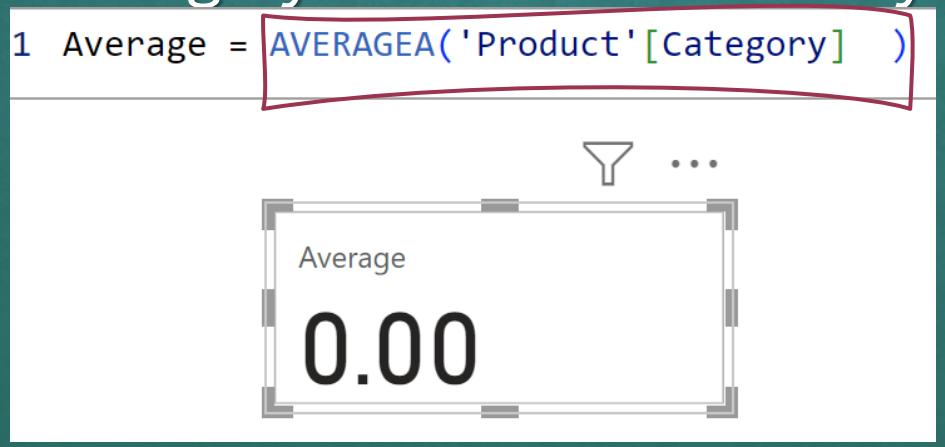
- This DAX code calculates the average value of top 10 numbers in the table.
- TopN function returns the top N values as a table to AVERAGEX function.

```
1  Average_Sales_amt = AVERAGEX(TOPN(10, Sales,Sales[Sales Amount]), [Sales Amount])
```

Karthimeena

AVERAGEA

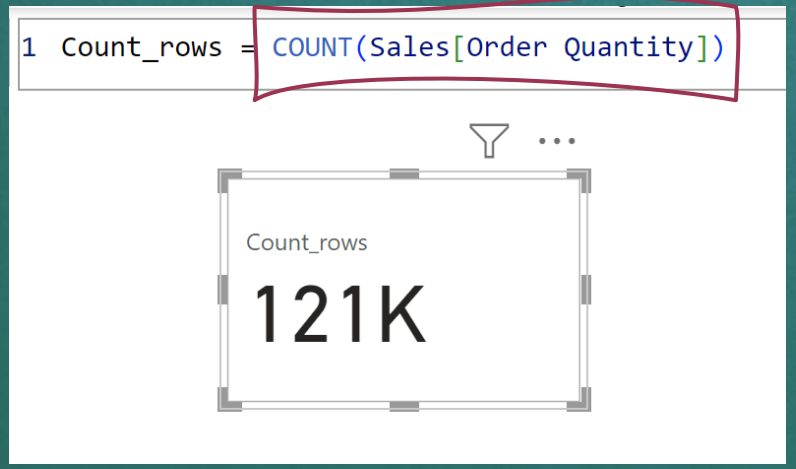
- This returns the average of values in a column and it handled the non-numeric columns.
- Values that contain non-numeric text,0, empty (“”) are treated as 0 when calculating the average.
- The column ‘Category’ does not contain any numeric values.



Karthimeena

COUNT

- This returns number of rows in a given column that contains non-blank values.
- It counts the string, date, number.
- COUNT() accepts a column as an arguments and returns integer value.



Karthimeena

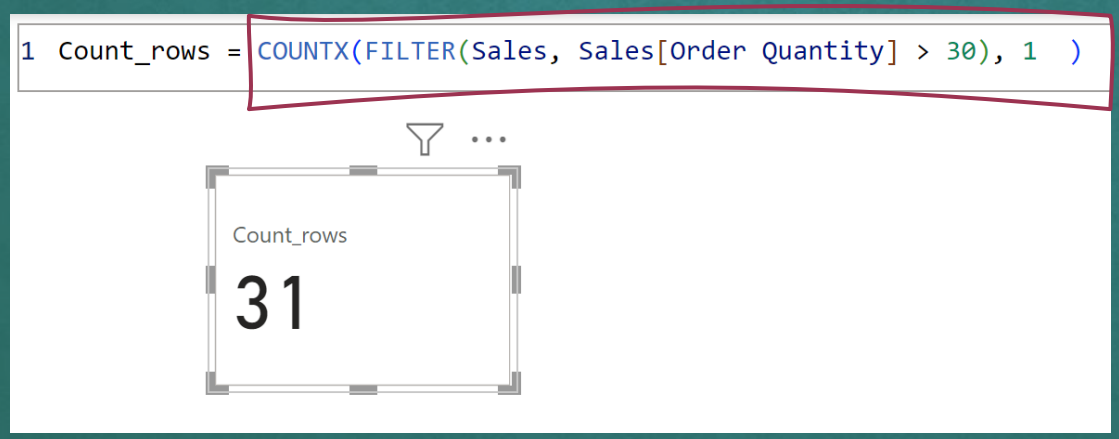
COUNTX

- This used to count the number of rows in a table or table expression that satisfies a specified condition.
- COUNTX() accepts table as an first arguments and expression as an second argument.
- It count only non-blank values from the expression.

Karthimeena

COUNTX

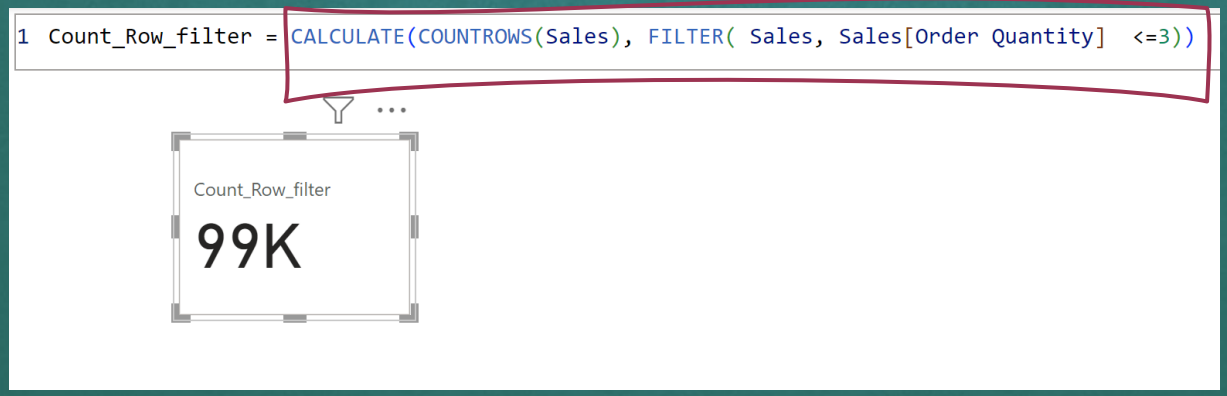
- Below code, uses filter as the first argument and ‘1’ as expression to count each rows from filtered table.



Karthimeena

COUNTROWS

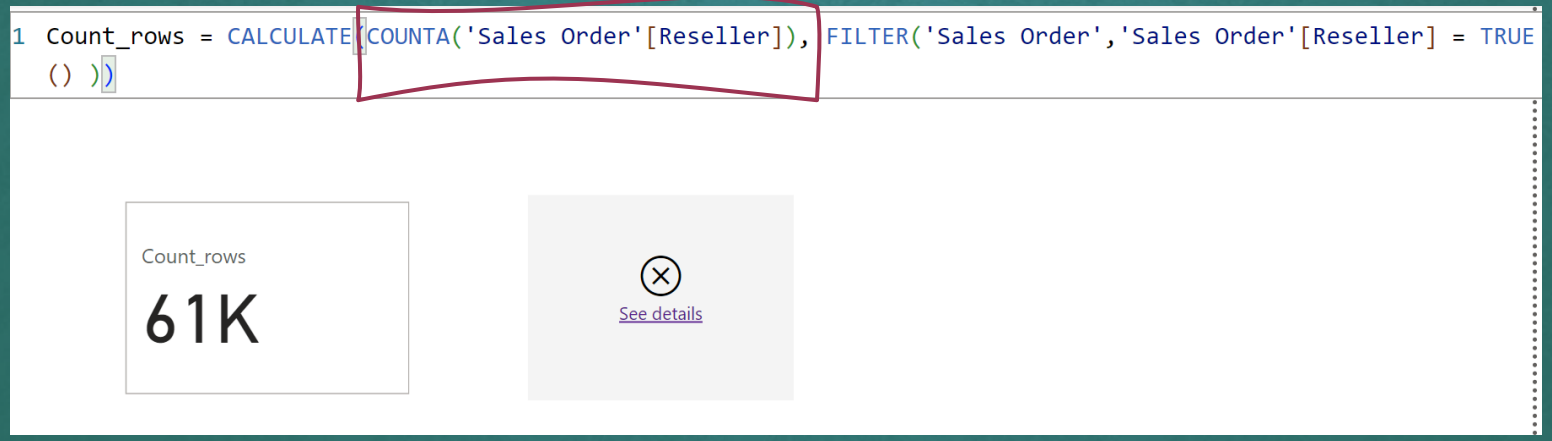
- COUNTROWS() used to count the number of rows in the table or given expression.
- The parameter is optional here.
- If parameter is not given, it takes default value as the home table of current context



Karthimeena

COUNTA

- COUNTA() used to count the number of rows in the table contains non-blank values.
- COUNTA() supports BOOLEAN data type.

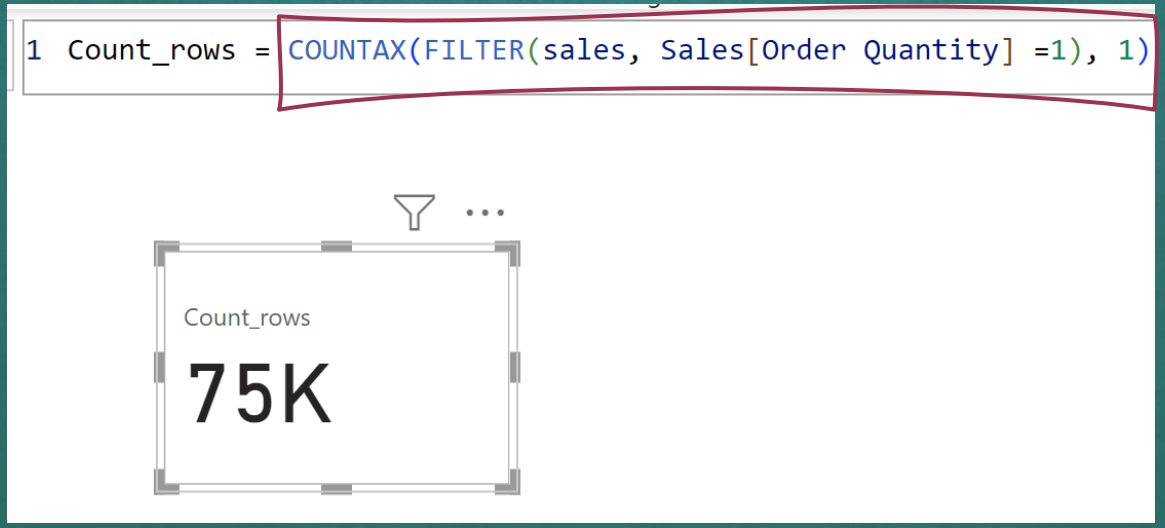


Karthimeena

COUNTAX

- COUNTAX() function counts the number of rows of a given expression in the context.
- The following code counts the number of rows from filtered table where order quantity is equal to 1 and uses 1 as a expression to count the rows.

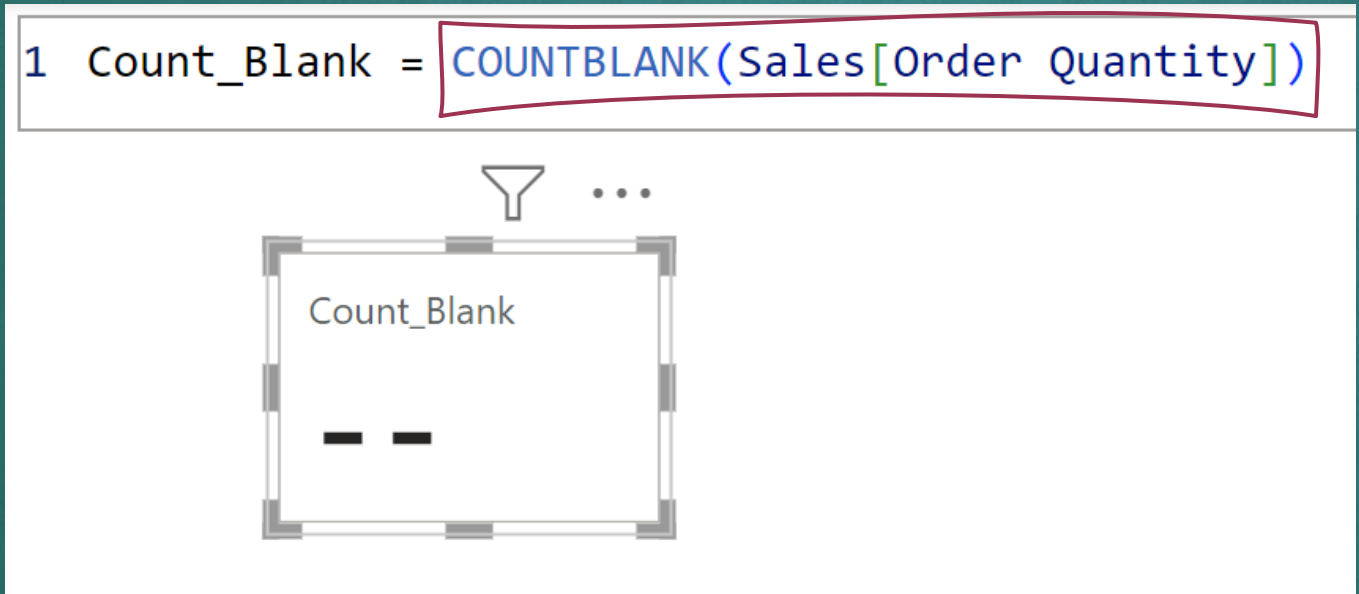
```
1  Count_rows = COUNTAX(FILTER(sales, Sales[Order Quantity] =1), 1)
```



Karthimeena

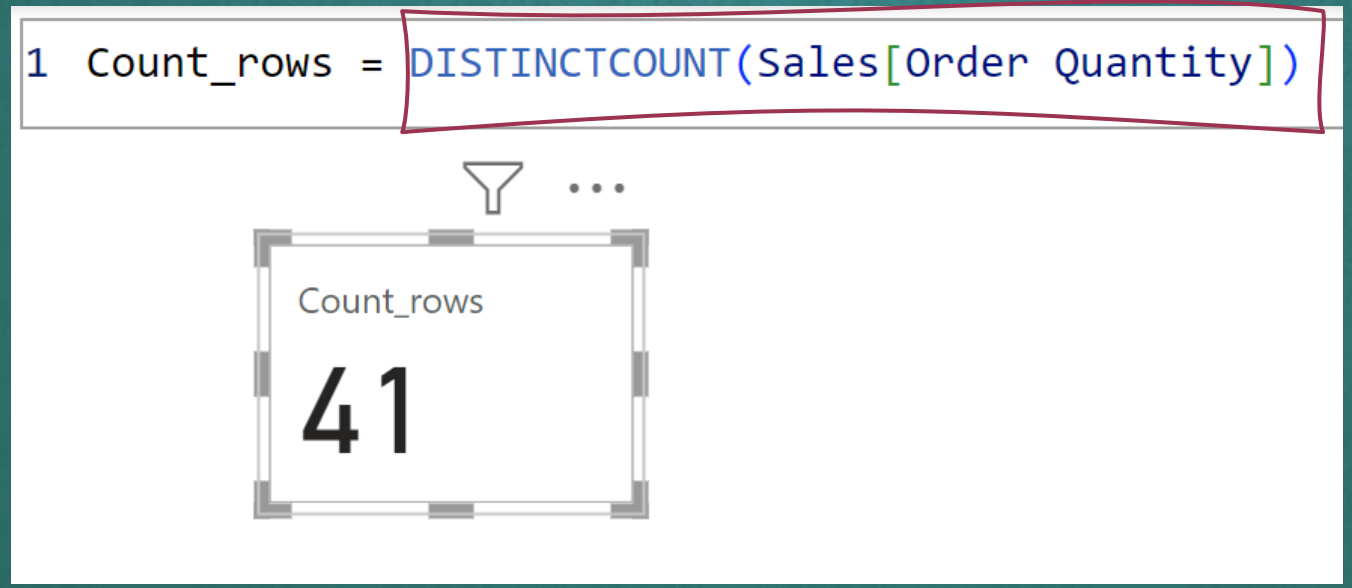
COUNTBLANK

- COUNTBLANK() function counts the number of rows with BLANK value.



DISTINCTCOUNT

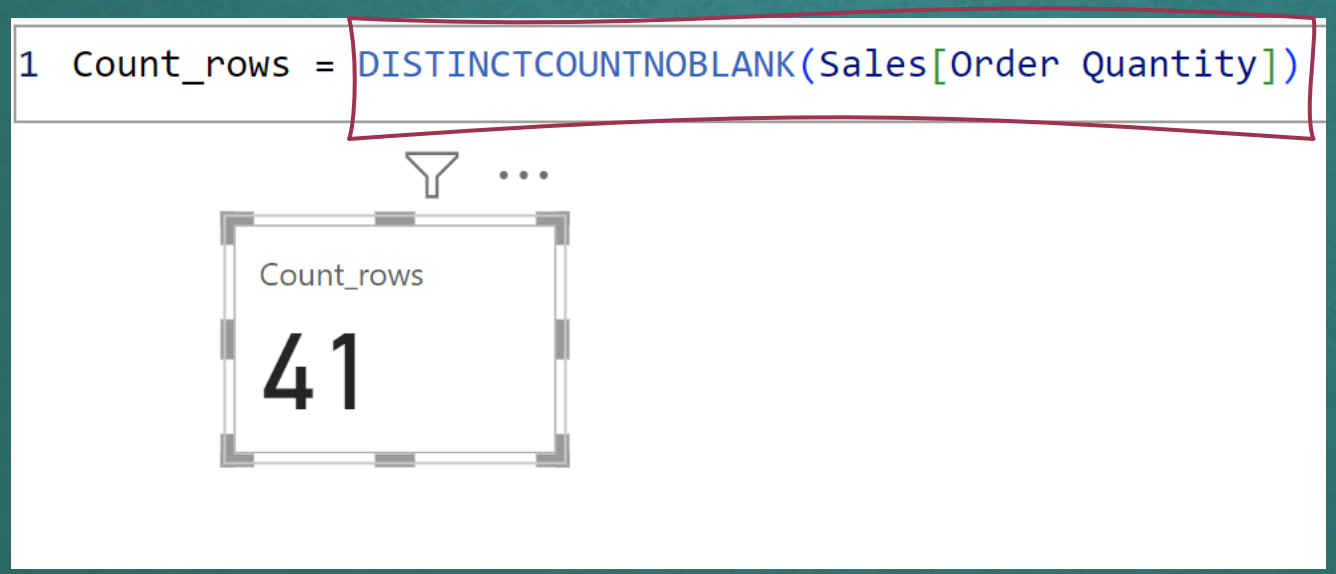
- This function counts the number of distinct values in a table.



Karthimeena

DISTINCTCOUNTNOBLANK

- This function counts the number of distinct values in a table and does not include blank values if they occurs.



Karthimeena