DAX

IN POWER BI

DAX (DATA ANALYSIS EXPRESSIONS)

It is a formula language and a collection of functions used to create custom calculations. It helps us create new information from data already in our model.

Table which I have used to explain DAX functions:

Customer_id	Product_id	Quantity	Store_id	Transaction_date
1001	2001	2	5001	10-04-2024
1001	2003	1	5001	10-04-2024
1002	2002	3	5002	11-04-2024
1003	2004	1	5003	12-04-2024
1003	2005	2	5003	12-04-2024
1004	2002	1	5004	13-04-2024
1004	2003	4	5004	13-04-2024
1005	2001	2	5005	14-04-2024
1005	2005	1	5005	14-04-2024
1005	2005	2	5005	

Customer Table

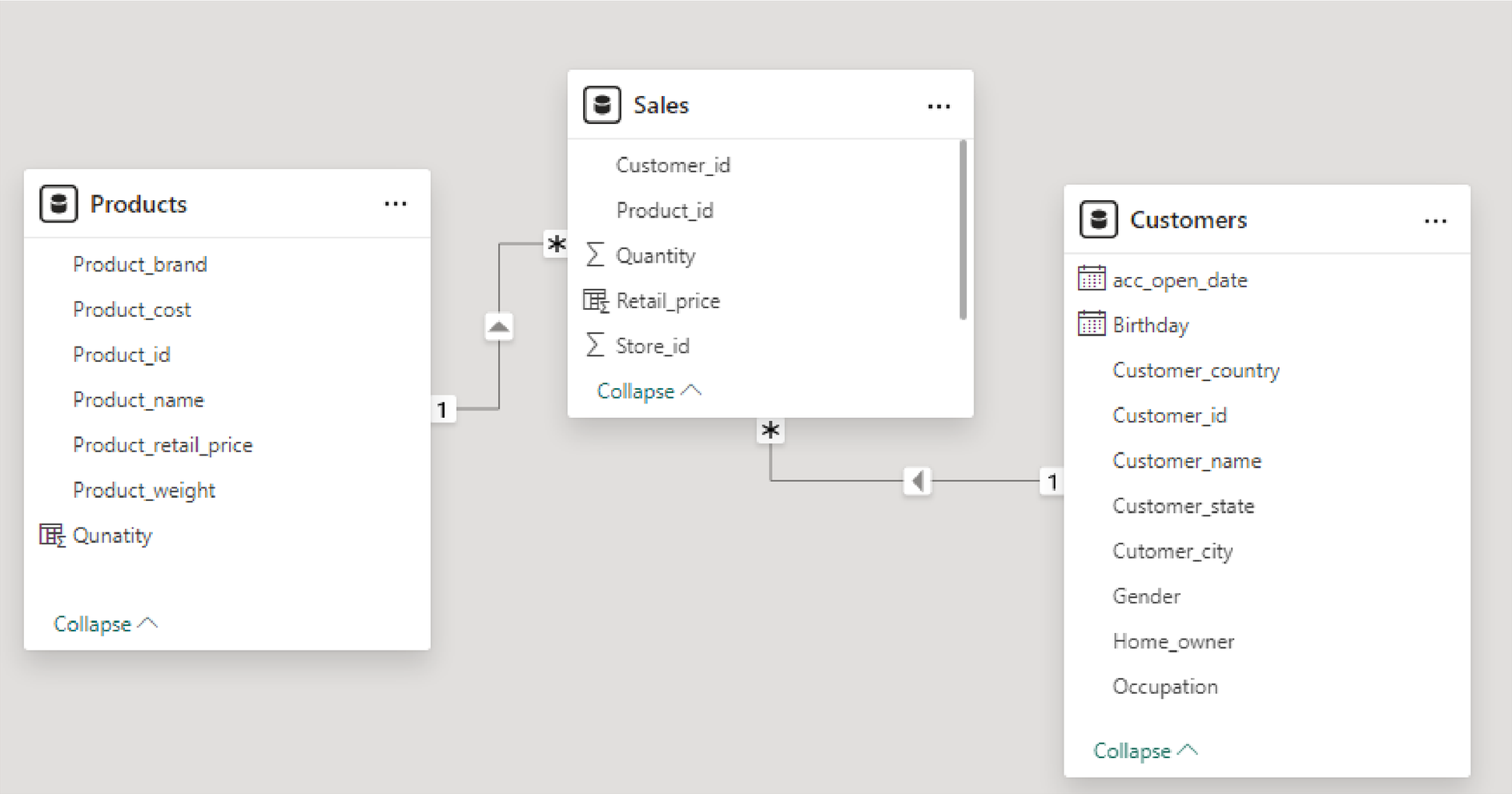
acc_open_date	Birthday	Cutomer_city	Customer_country	Customer_id	Customer_name	Customer_state	Gender	Home_owner	Occupation
15-10-2023	20-05-1990	New York City	USA	1001	John Smith	New York	Male	Yes	Clerical
28-09-2022	10-11-1985	London	UK	1002	Emily Jones	London	Female	No	Managerial
12-03-2024	03-08-1978	Paris	France	1003	Jacques Dupont	Île-de-France	Male	Yes	Managerial
05-06-2023	15-02-1995	Sydney	Australia	1004	Sarah Lee	New South Wales	Female	Yes	Managerial
20-12-2022	25-10-1980	Tokyo	Japan	1005	Takeshi Tanaka	Tokyo	Male	No	clerical

Sales Table

Product_brand	Product_cost	Product_id	Product_name	Product_retail_price	Product_weight
BRAND_A	1000	2001	P1	100	2 kg
BRAND_B	2000	2002	P2	500	0.2 kg
BRAND_C	3000	2003	P3	300	0.5 kg
BRAND_D	4000	2004	P4	400	1 kg
BRAND_E	5000	2005	P5	500	0.1 kg

Product Table

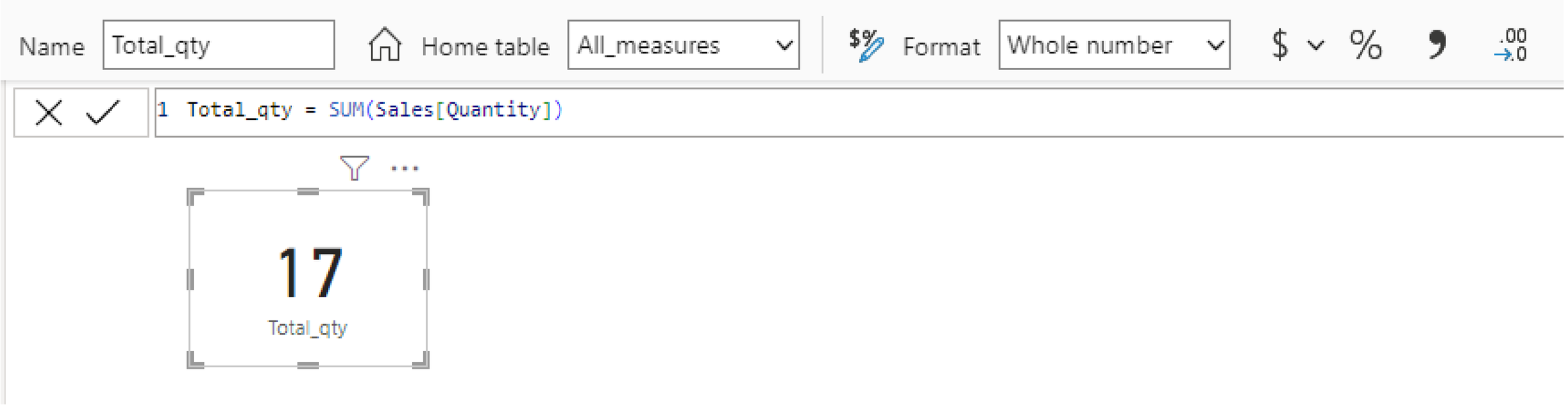
Data Model:


Products
Product_brand
Product_cost
Product_id
Product_name
Product_retail_price
Product_weight
Qunatity
Collapse
Sales
Customer_id
Product_id
Quantity
Retail_price
Store_id
Collapse
Customers
acc_open_date
Birthday
Customer_country
Customer_id
Customer_name
Customer_state
Cutomer_city
Gender
Home_owner
Occupation
Collapse
1
*
*
1

Here are some of the important functions:

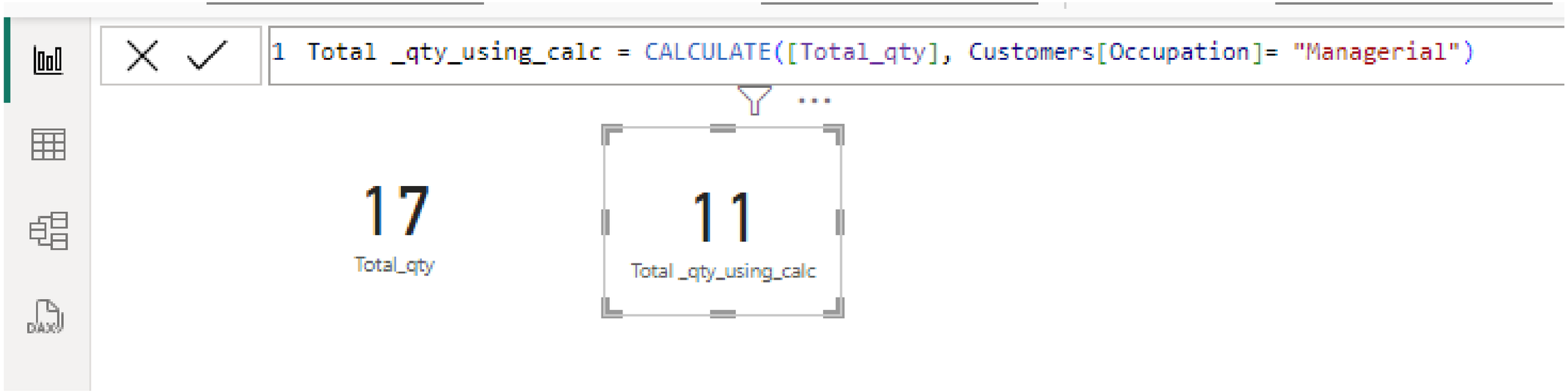
SUM: Return the sum of the values in a column or an expression.

Syntax: SUM(<column_or_expression>)



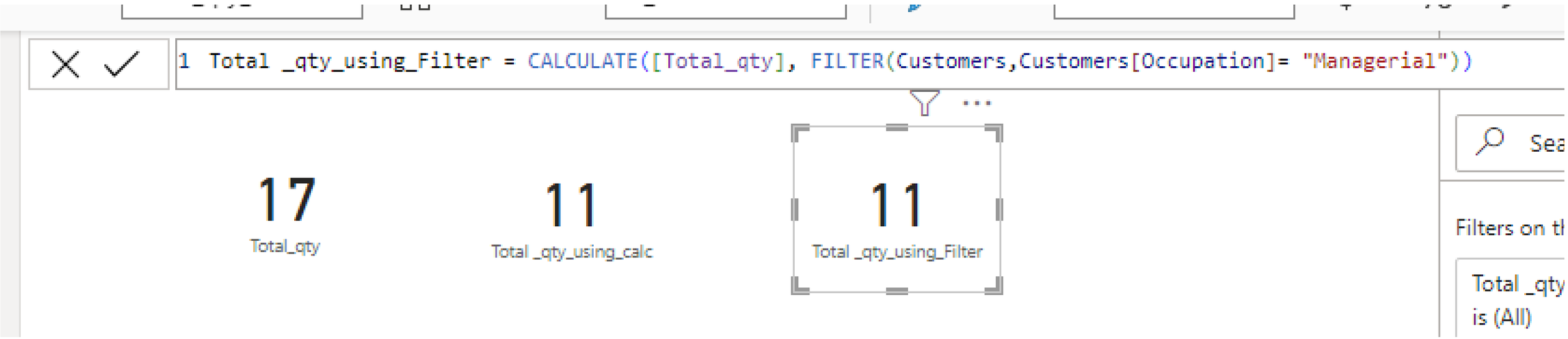
CALCULATE: Evaluates an expression in a modified filter context.

Syntax:
CALCULATE(<expression>[, <filter1> [, <filter2> [, ...]]])



FILTER: Returns a table that represents a subset of another table or expression.

Syntax: FILTER(<table>,<filter>)



VALUES: Returns a one-column table that contains the distinct values from the specified column.

Syntax: VALUES(<TableNameOrColumnName>)

```
Total _qty_using_values = CALCULATE([Total_qty], FILTER(VALUES(Customers[Occupation]),Customers[Occupation]= "Managerial"))
```

17	11	11	11
Total_qty	Total _qty_using_calc	Total _qty_using_Filter	Total _qty_using_values

NOTE:

In the earlier slides, we observed that using **CALCULATE** alone and employing a combination of **CALCULATE**, **FILTER**, and **VALUES** yield the same result. However, the latter approach represents a more optimized way of writing DAX.

ALL: Returns all the rows in a table, or all the values in a column, ignoring any filters that might have been applied.

Syntax:

ALL([<table> | <column>[, <column>[, <column>[,...]]]])

```
All_Measure = CALCULATE([Total_qty], ALL(Customers))
```

Customer_name	Total_qty	All_Measure
Emily Jones	3	17
Jacques Dupont	3	17
John Smith	3	17
Sarah Lee	5	17
Takeshi Tanaka	3	17
Total	**17**	**17**

Customer_name

- [ ] Emily Jones
- [ ] Jacques Dupont
- [ ] John Smith
- [ ] Sarah Lee
- [ ] Takeshi Tanaka

ALL SELECTED: This function gets the context that represents all rows and columns in the query, while keeping explicit filters and contexts other than row and column filters. .

Syntax:

ALL([<table> | <column>[, <column>[, <column>[,...]]]])

```
All_selected_Measure = CALCULATE([Total_qty], ALLSELECTED(Customers))
```

Customer_name	Total_qty	All_Measure	All_selected_Measure
Emily Jones	3	17	17
Jacques Dupont	3	17	17
John Smith	3	17	17
Sarah Lee	5	17	17
Takeshi Tanaka	3	17	17
Total	**17**	**17**	**17**

Customer_name
- ☐ Emily Jones
- ☐ Jacques Dupont
- ☐ John Smith
- ☐ Sarah Lee
- ☐ Takeshi Tanaka

Without Adding Explicit Filter

```
All_selected_Measure = CALCULATE([Total_qty], ALLSELECTED(Customers))
```

Customer_name	Total_qty	All_Measure	All_selected_Measure
John Smith	3	17	3
Total	**3**	**17**	**3**

Customer_name
- ☐ Emily Jones
- ☐ Jacques Dupont
- ■ John Smith
- ☐ Sarah Lee
- ☐ Takeshi Tanaka

After Adding Explicit Filter - Name

ALL EXCEPT: Removes all context filters in the table except filters that have been applied to the specified columns.

Syntax:
ALLEXCEPT(<table>,<column>[,<column>[,...]])

```
All_Except_Measure = CALCULATE([Total_qty], ALLEXCEPT(Customers,Customers[Cutomer_city]))
```

Customer_name	Total_qty	All_Measure	All_selected_Measure	All_Except_Measure
Emily Jones	3	17	17	17
Jacques Dupont	3	17	17	17
John Smith	3	17	17	17
Sarah Lee	5	17	17	17
Takeshi Tanaka	3	17	17	17
Total	**17**	**17**	**17**	**17**

Customer_name
- Emily Jones
- Jacques Dupont
- John Smith
- Sarah Lee
- Takeshi Tanaka

Without Filter

```
All_Except_Measure = CALCULATE([Total_qty], ALLEXCEPT(Customers,Customers[Cutomer_city]))
```

Customer_name	Cutomer_city	Total_qty	All_Measure	All_selected_Measure	All_Except_Measure
Emily Jones	London	3	17	17	3
Jacques Dupont	Paris	3	17	17	3
John Smith	New York City	3	17	17	3
Sarah Lee	Sydney	5	17	17	5
Takeshi Tanaka	Tokyo	3	17	17	3
Total		**17**	**17**	**17**	**17**

Customer_name
- Emily Jones
- Jacques Dupont
- John Smith
- Sarah Lee
- Takeshi Tanaka

After using Filter - city

COUNTROWS(): This function counts the number of rows in the specified table

Syntax: COUNTROWS([<table>])

COUNT(): Counts the number of rows in the specified column that contain non-blank values.

Syntax: COUNT([<table>])

COUNTX(): This function is used to count the number of rows in a table that meet specified conditions.

Syntax: COUNTX([<table>])

```
Count_X = COUNTX(Sales,[Total_qty])
```

10	9	10
count_rows	Count_c	Count_X

SUMMARIZE: This function is used to create a summary table that aggregates data based on specified groupings or criteria. It's similar to SQL's GROUP BY clause.

Syntax:

SUMMARIZE (<table>, <groupBy_columnName>
[, <groupBy_columnName>]…[, <name>, <expression>]…)

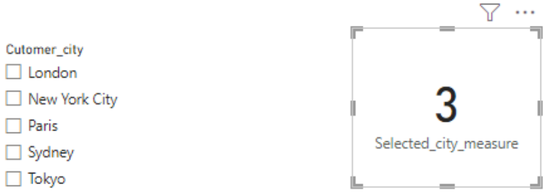
```
1 Summary = SUMMARIZE(Sales,Sales[Product_id],"Total qty", [Total_qty])
```

Product_id	Total qty
2001	4
2003	5
2002	4
2004	1
2005	5

SELECTEDVALUE(): Returns the value when the context for columnName has been filtered down to one distinct value only. Otherwise returns alternateResult.

Syntax:
SELECTEDVALUE(<columnName>[, <alternateResult>])

```
Selected_city_measure = CALCULATE([Total_qty],Customers[Cutomer_city] = SELECTEDVALUE(Customers[Cutomer_city],"Paris"))
```



Showing Defalut value - Paris

```
Selected_city_measure = CALCULATE([Total_qty],Customers[Cutomer_city] = SELECTEDVALUE(Customers[Cutomer_city],"Paris"))
```

Showing Selected value - Sydeny

ROW: This function generates a single-row table with specified column values.

Syntax:
ROW(<name>, <expression>[[,<name>, <expression>]...])

```
1 One_row_table = ROW("total quantity", [Total_qty])
```

total quantity
19

RELATED: Returns a related value from another table.

Syntax:

RELATED(<column>)

```
Retail_price = RELATED(Products[Product_retail_price])
```

Customer_id	Product_id	Quantity	Store_id	Transaction_date	Retail_price
1001	2001	2	5001	10 April 2024	$100
1001	2003	1	5001	10 April 2024	$300
1002	2002	3	5002	11 April 2024	$500
1003	2004	1	5003	12 April 2024	$400
1003	2005	2	5003	12 April 2024	$500
1004	2002	1	5004	13 April 2024	$500
1004	2003	4	5004	13 April 2024	$300
1005	2001	2	5005	14 April 2024	$100
1005	2005	1	5005	14 April 2024	$500
1005	2005	2	5005		$500

RELATEDTABLE: Retrieve a table of related records from a related table based on a defined relationship between two tables in a data model.

Syntax:

RELATEDTABLE(<tableName>)

```
1 Qunatity = SUMX(RELATEDTABLE(Sales),[Total_qty])
```

Product_brand	Product_cost	Product_id	Product_name	Product_retail_price	Product_weight	Qunatity
	1000	2001	P1	$100	2 kg	4
	2000	2002	P2	$500	0.2 kg	4
	3000	2003	P3	$300	0.5 kg	5
	4000	2004	P4	$400	1 kg	1
	5000	2005	P5	$500	0.1 kg	5

DYNAMIC DATE TABLE: Using Calender, Generate and Row

```
calender_table =
var a = CALENDAR(DATE(2018,01,01), DATE(YEAR(TODAY()), MONTH(TODAY()), DAY(TODAY())))
return
GENERATE(a,
var b  = [Date]
var c = YEAR([Date])
var d = MONTH([Date])
var e = DAY([Date])
RETURN
ROW("Year", c, "Month", d, "Day",e))
```

Date	Year	Month	Day
01-01-2020 00:00:00	2020	1	1
02-01-2020 00:00:00	2020	1	2
03-01-2020 00:00:00	2020	1	3
04-01-2020 00:00:00	2020	1	4
05-01-2020 00:00:00	2020	1	5
06-01-2020 00:00:00	2020	1	6
07-01-2020 00:00:00	2020	1	7
08-01-2020 00:00:00	2020	1	8
09-01-2020 00:00:00	2020	1	9
10-01-2020 00:00:00	2020	1	10
11-01-2020 00:00:00	2020	1	11
12-01-2020 00:00:00	2020	1	12
13-01-2020 00:00:00	2020	1	13

THANK YOU