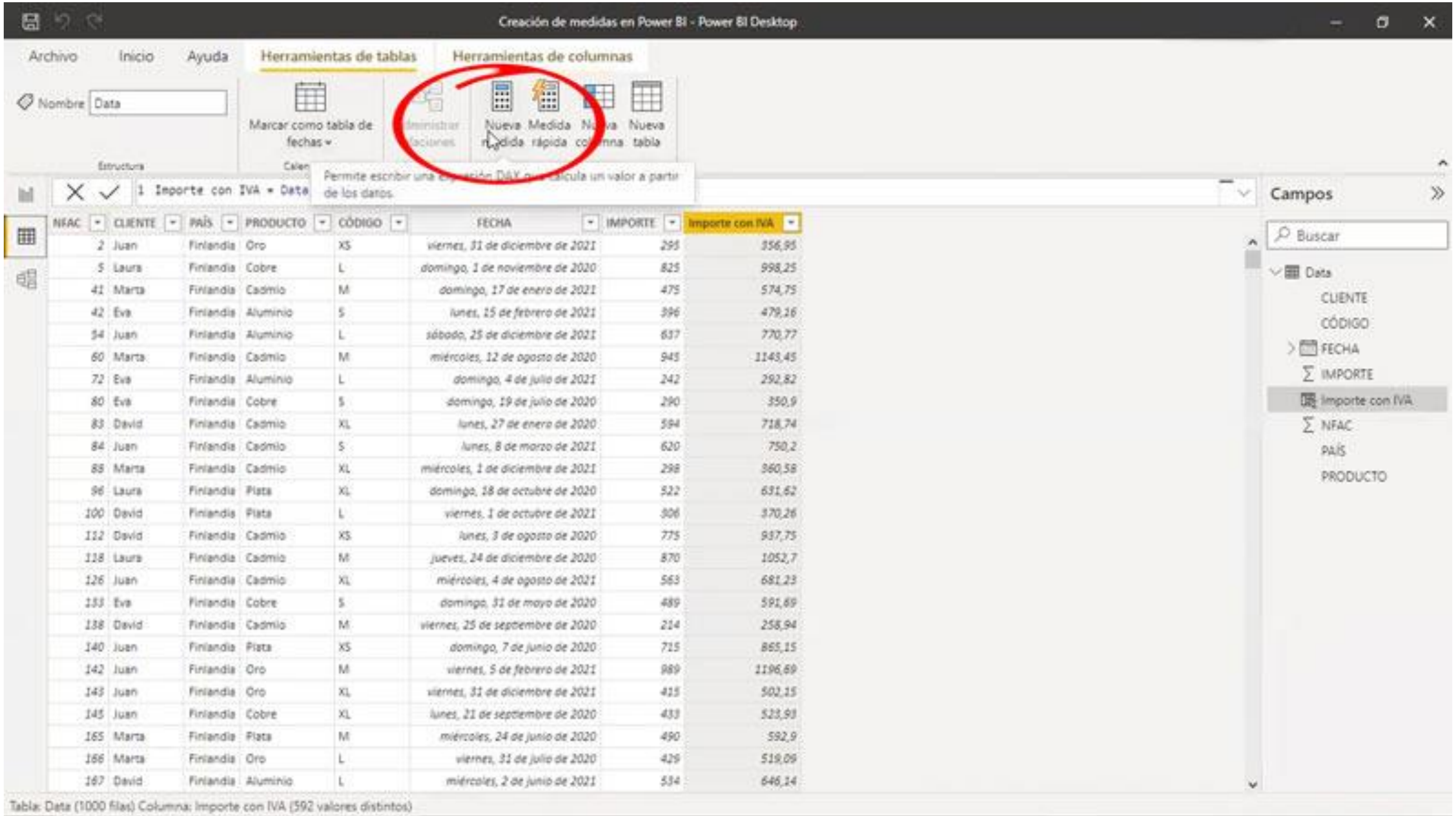
CONTENIDOS

https://github.com/jorloicono/AF-DAX-AVANZADO

PRESENTACIONES

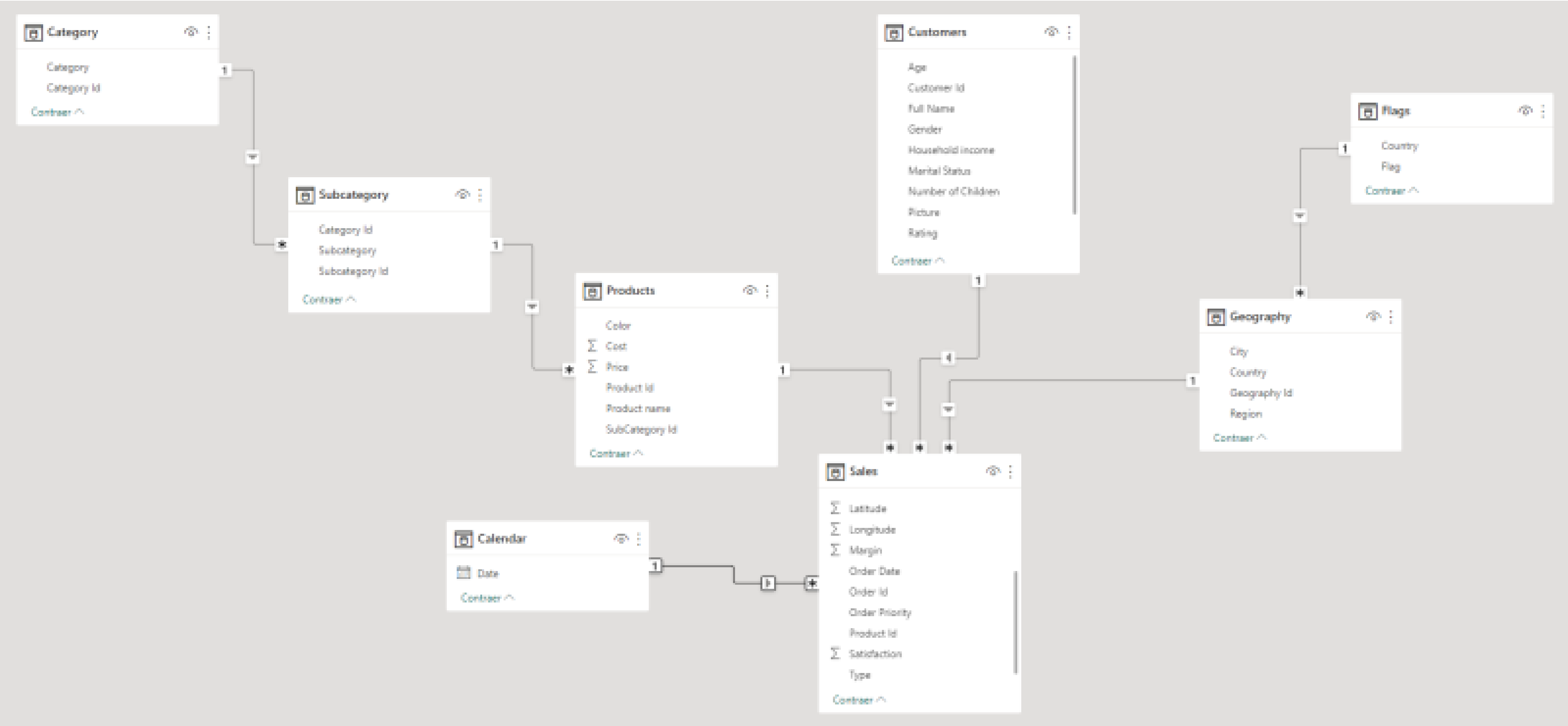


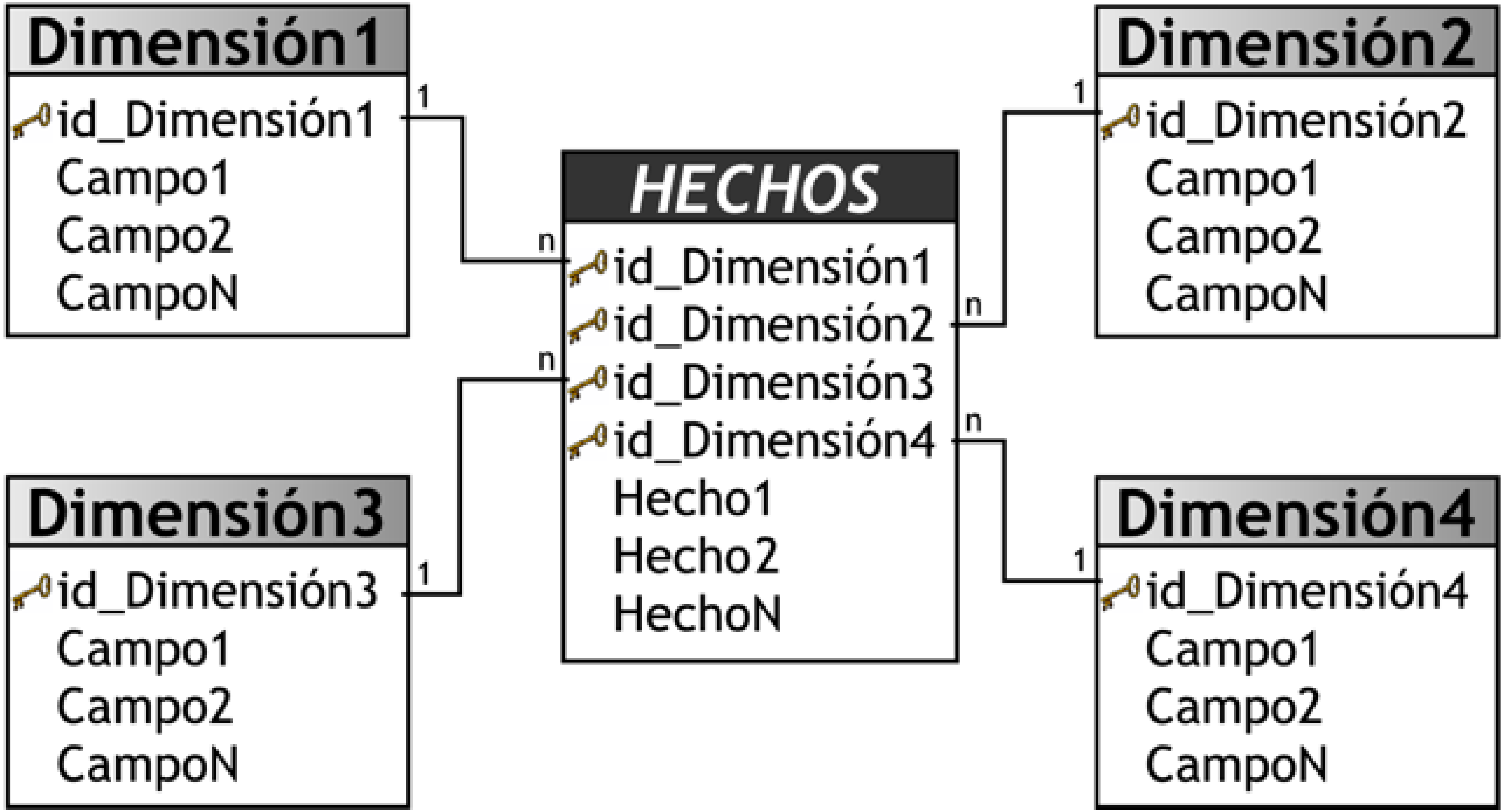
Permite escribir una expresión DAX que calcula un valor a partir de los datos.

```
1 Importe con IVA = Data
```

NFAC	CLIENTE	PAÍS	PRODUCTO	CÓDIGO	FECHA	IMPORTE	Importe con IVA
2	Juan	Finlandia	Oro	XS	viernes, 31 de diciembre de 2021	295	356,95
5	Laura	Finlandia	Cobre	L	domingo, 1 de noviembre de 2020	825	998,25
41	Marta	Finlandia	Cadmio	M	domingo, 17 de enero de 2021	475	574,75
42	Eva	Finlandia	Aluminio	S	lunes, 15 de febrero de 2021	396	479,16
54	Juan	Finlandia	Aluminio	L	sábado, 25 de diciembre de 2021	637	770,77
60	Marta	Finlandia	Cadmio	M	miércoles, 12 de agosto de 2020	945	1143,45
72	Eva	Finlandia	Aluminio	L	domingo, 4 de julio de 2021	242	292,82
80	Eva	Finlandia	Cobre	S	domingo, 19 de julio de 2020	290	350,9
83	David	Finlandia	Cadmio	XL	lunes, 27 de enero de 2020	594	718,74
84	Juan	Finlandia	Cadmio	S	lunes, 8 de marzo de 2021	620	750,2
88	Marta	Finlandia	Cadmio	XL	miércoles, 1 de diciembre de 2021	298	360,58
96	Laura	Finlandia	Plata	XL	domingo, 18 de octubre de 2020	522	631,62
100	David	Finlandia	Plata	L	viernes, 1 de octubre de 2021	306	370,26
112	David	Finlandia	Cadmio	XS	lunes, 3 de agosto de 2020	775	937,75
118	Laura	Finlandia	Cadmio	M	jueves, 24 de diciembre de 2020	870	1052,7
126	Juan	Finlandia	Cadmio	XL	miércoles, 4 de agosto de 2021	563	681,23
133	Eva	Finlandia	Cobre	S	domingo, 31 de mayo de 2020	489	591,69
138	David	Finlandia	Cadmio	M	viernes, 25 de septiembre de 2020	214	258,94
140	Juan	Finlandia	Plata	XS	domingo, 7 de junio de 2020	715	865,15
142	Juan	Finlandia	Oro	M	viernes, 5 de febrero de 2021	989	1196,69
143	Juan	Finlandia	Oro	XL	viernes, 31 de diciembre de 2021	415	502,15
145	Juan	Finlandia	Cobre	XL	lunes, 21 de septiembre de 2020	433	523,93
165	Marta	Finlandia	Plata	M	miércoles, 24 de junio de 2020	490	592,9
166	Marta	Finlandia	Oro	L	viernes, 31 de julio de 2020	429	519,09
167	David	Finlandia	Aluminio	L	miércoles, 2 de junio de 2021	534	646,14

Tabla: Data (1000 filas) Columna: Importe con IVA (592 valores distintos)


Category
Category
Category Id
Contraer ^
Subcategory
Category Id
Subcategory
Subcategory Id
Contraer ^
Products
Color
Cost
Price
Product Id
Product name
SubCategory Id
Contraer ^
Customers
Age
Customer Id
Full Name
Gender
Household income
Marital Status
Number of Children
Picture
Rating
Contraer ^
Flags
Country
Flag
Contraer ^
Geography
City
Country
Geography Id
Region
Contraer ^
Sales
Latitude
Longitude
Margin
Order Date
Order Id
Order Priority
Product Id
Satisfaction
Type
Contraer ^
Calendar
Date
Contraer ^
1
*
1
*
1
*
1
*
1
*
1
*
1
*



Dimensión1

- id_Dimensión1
- Campo1
- Campo2
- CampoN

Dimensión2

- id_Dimensión2
- Campo1
- Campo2
- CampoN

HECHOS

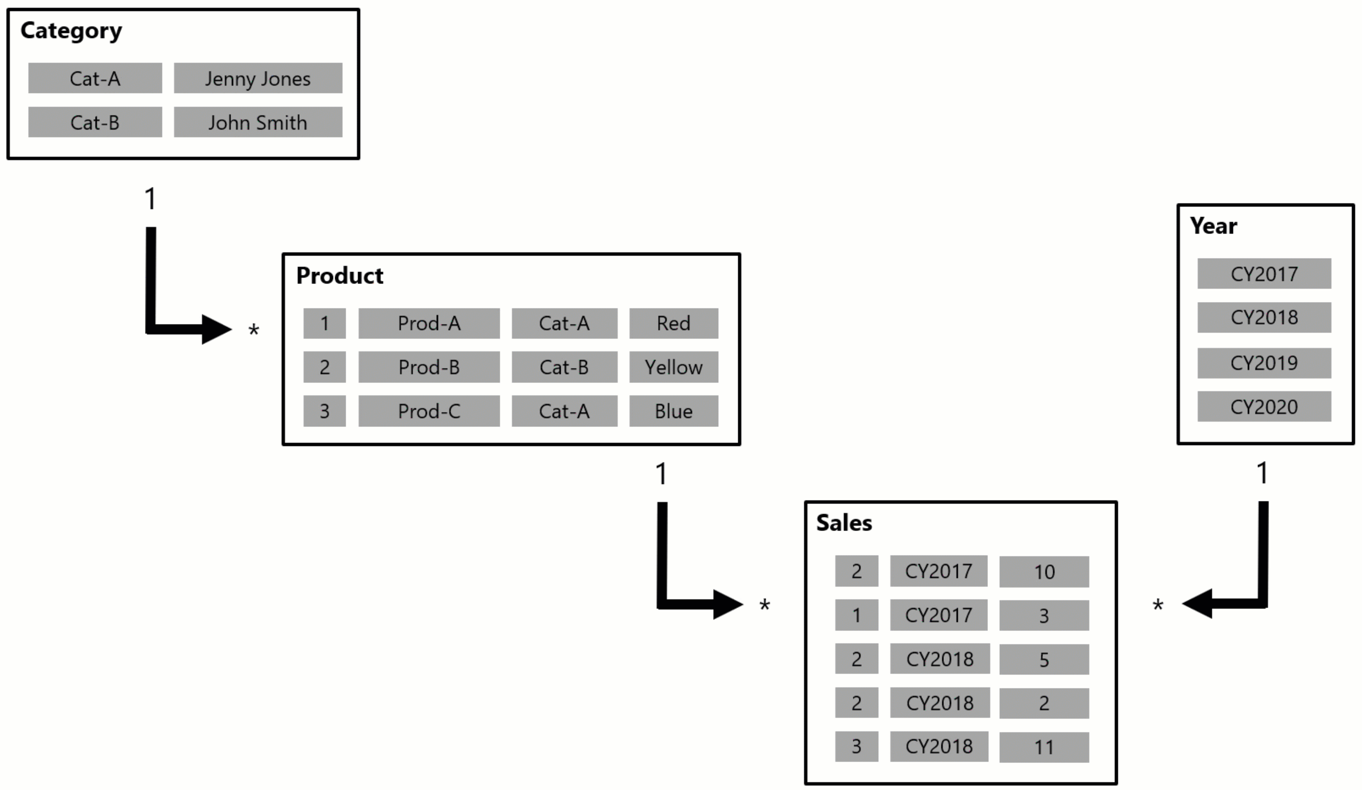
- id_Dimensión1
- id_Dimensión2
- id_Dimensión3
- id_Dimensión4
- Hecho1
- Hecho2
- HechoN

Dimensión3

- id_Dimensión3
- Campo1
- Campo2
- CampoN

Dimensión4

- id_Dimensión4
- Campo1
- Campo2
- CampoN



Category

Cat-A	Jenny Jones
Cat-B	John Smith

Product

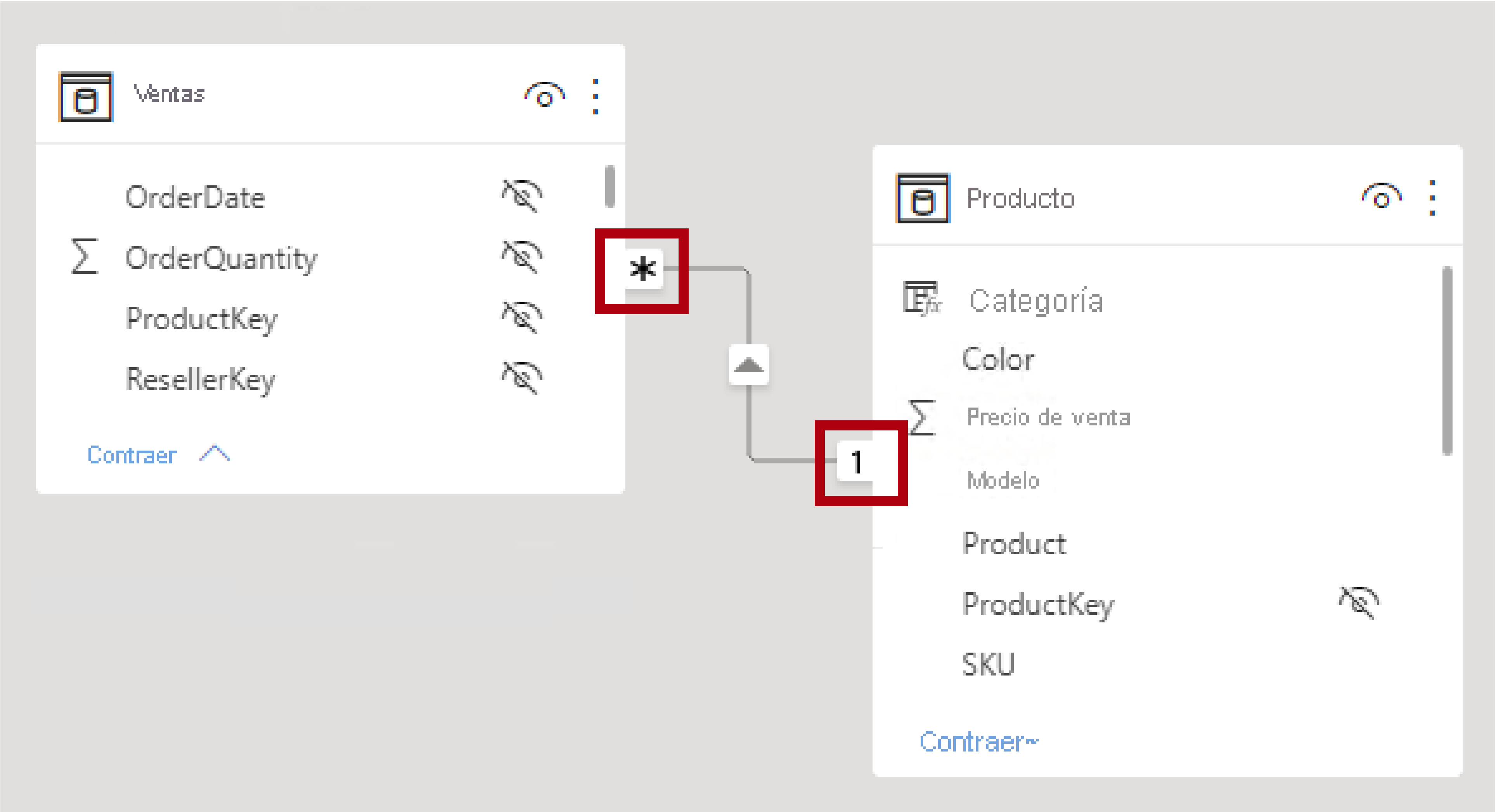
1	Prod-A	Cat-A	Red
2	Prod-B	Cat-B	Yellow
3	Prod-C	Cat-A	Blue

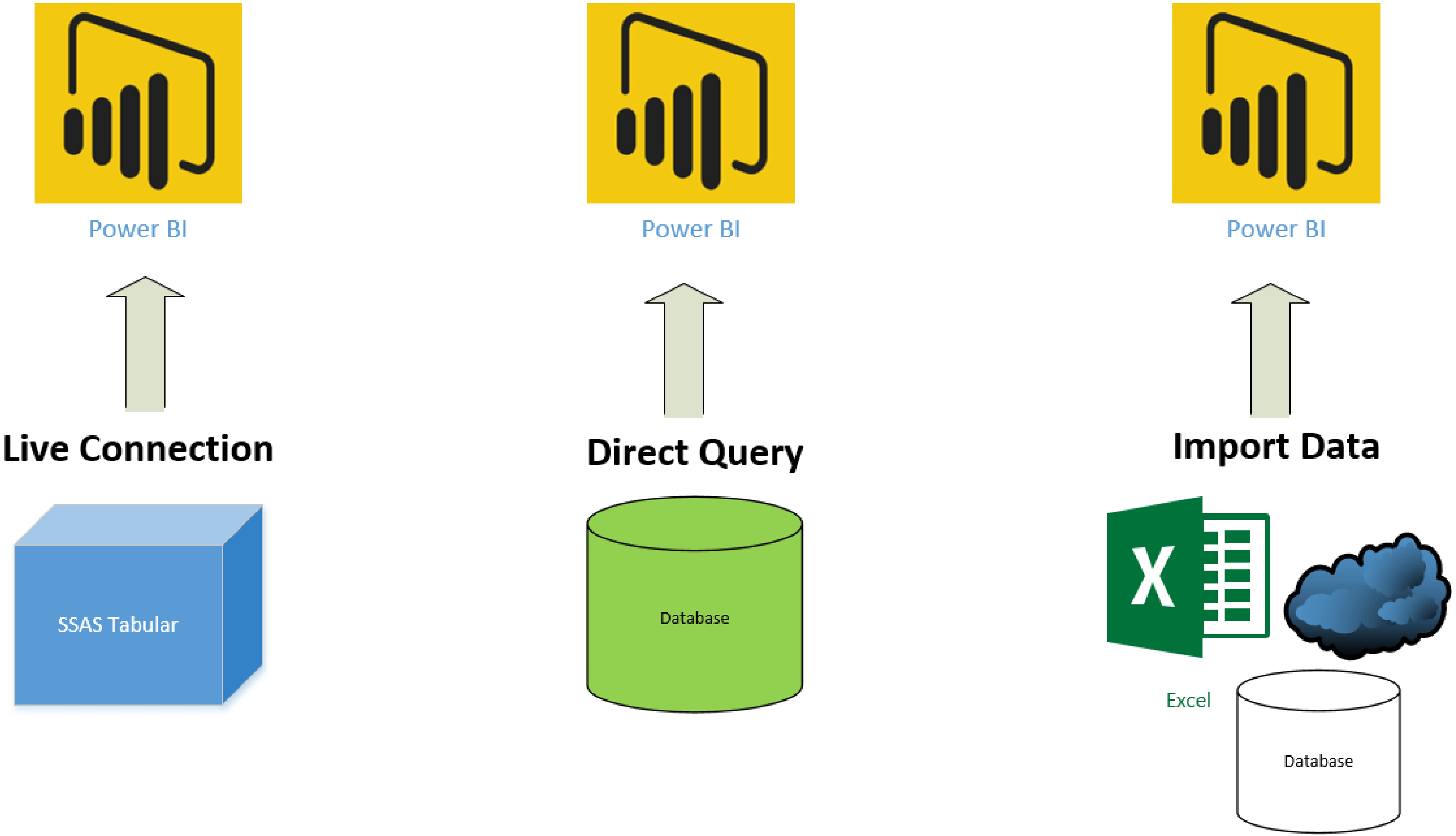
Year

CY2017
CY2018
CY2019
CY2020

Sales

2	CY2017	10
1	CY2017	3
2	CY2018	5
2	CY2018	2
3	CY2018	11


Ventas
OrderDate
∑ OrderQuantity
ProductKey
ResellerKey
Contraer
*
1
Producto
Categoría
Color
∑ Precio de venta
Modelo
Product
ProductKey
SKU
Contraer



Power BI

Live Connection

SSAS Tabular

Power BI

Direct Query

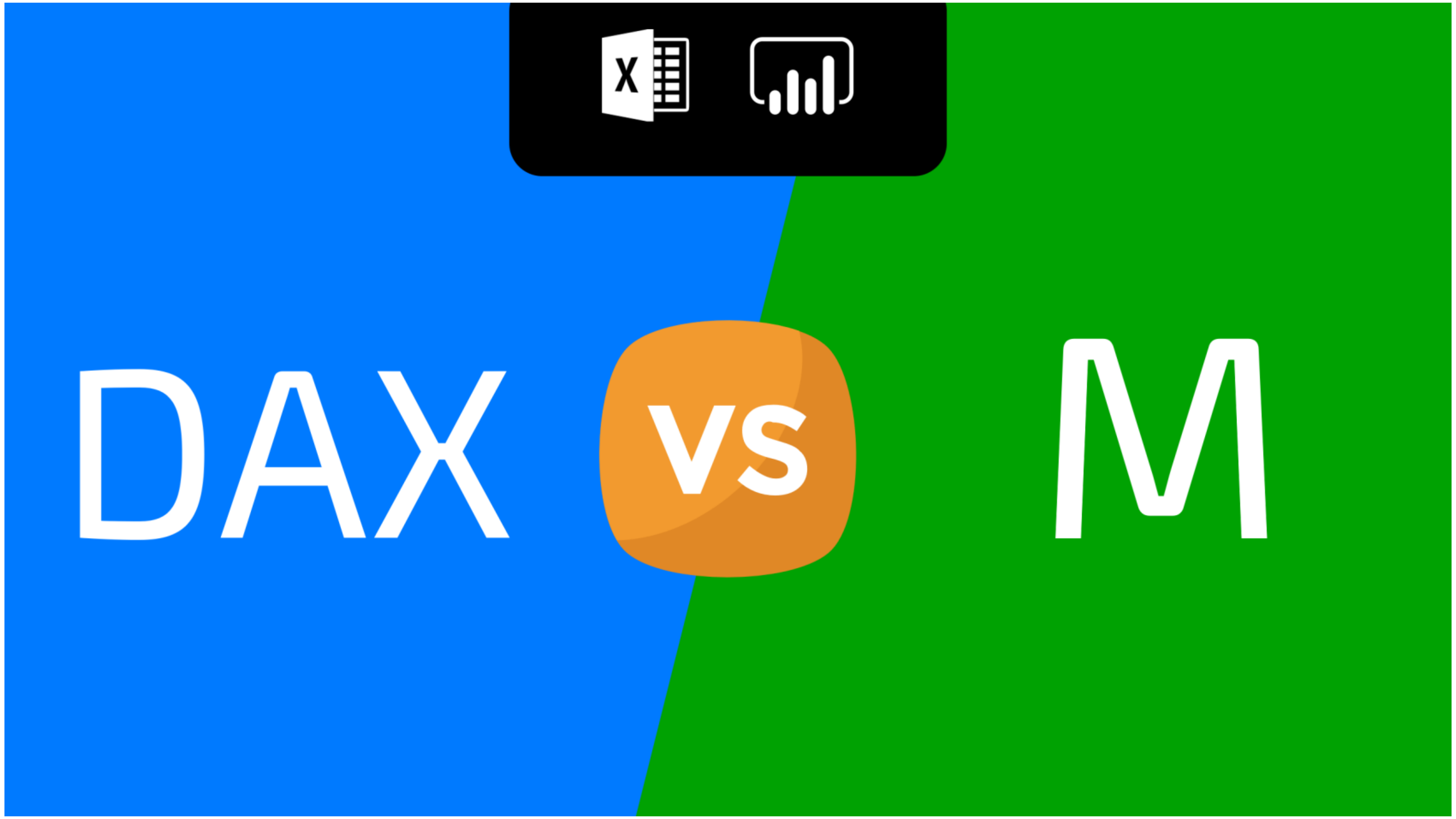
Database

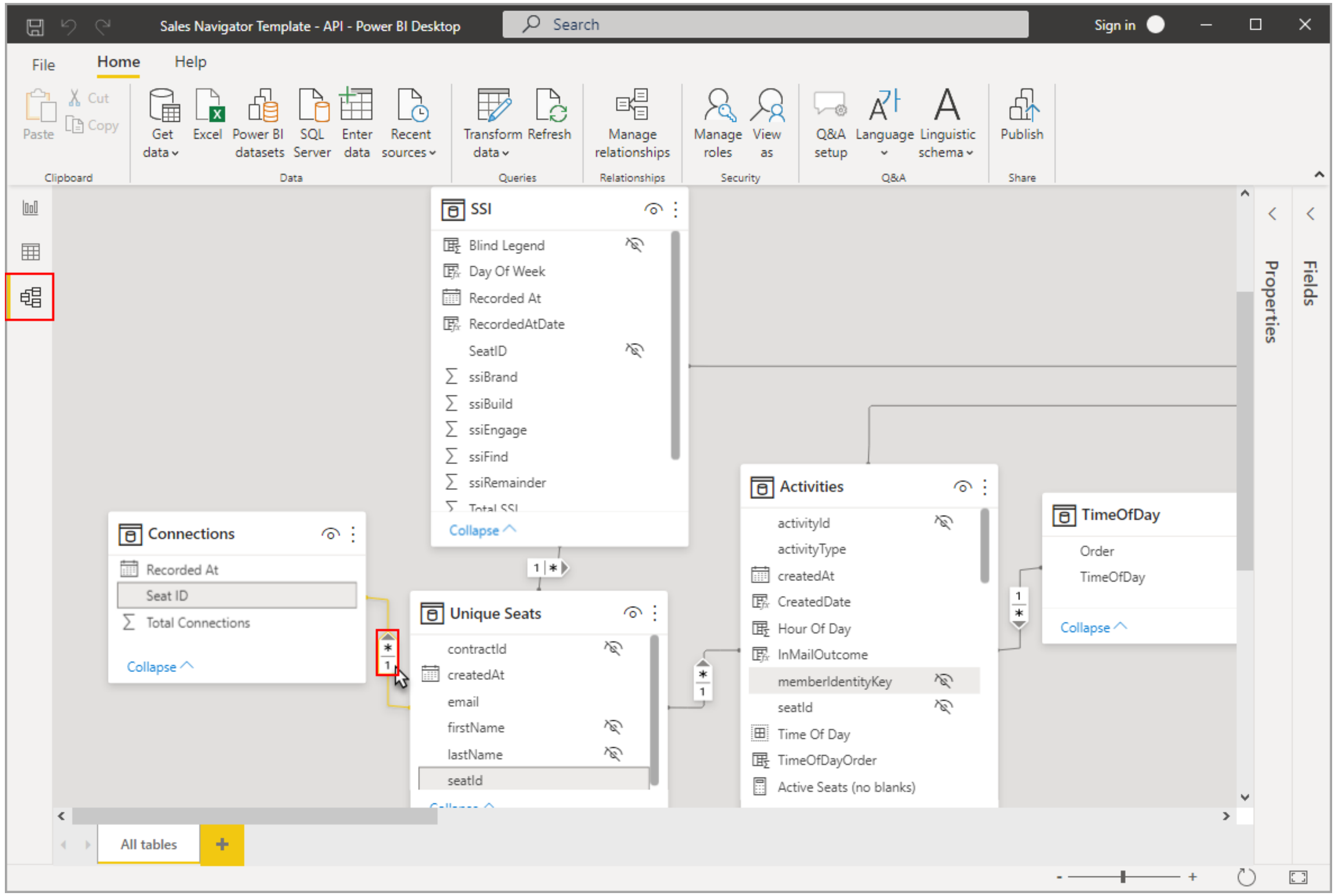
Power BI

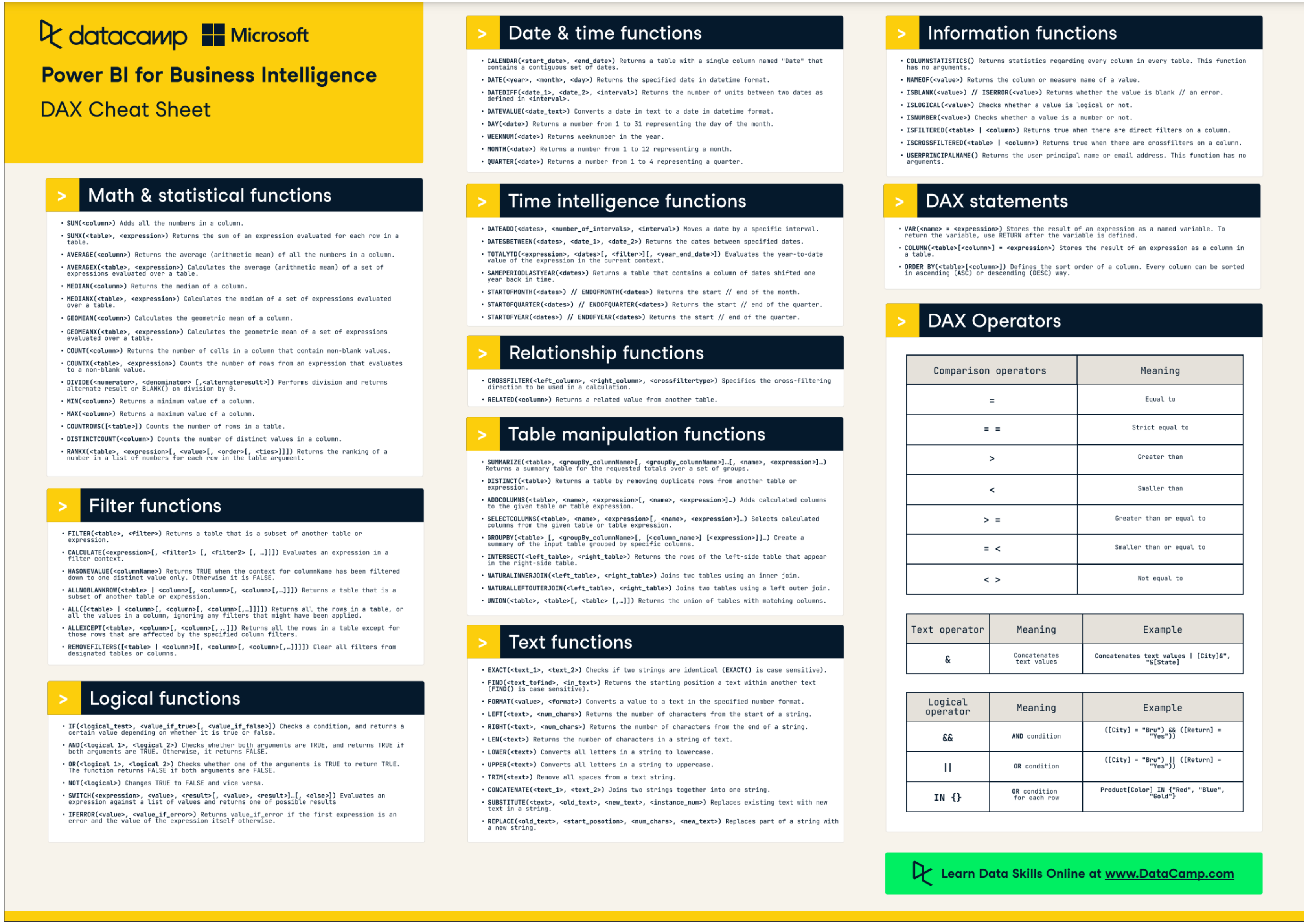
Import Data

Excel

Database


DAX
vs
M


Sales Navigator Template - API - Power BI Desktop
Search
Sign in
File
Home
Help
Paste
Cut
Copy
Clipboard
Get data
Excel
Power BI datasets
SQL Server
Enter data
Recent sources
Data
Transform data
Refresh
Queries
Manage relationships
Relationships
Manage roles
View as
Security
Q&A setup
Language
Linguistic schema
Q&A
Publish
Share
SSI
Blind Legend
Day Of Week
Recorded At
RecordedAtDate
SeatID
ssiBrand
ssiBuild
ssiEngage
ssiFind
ssiRemainder
Total SSI
Collapse
Connections
Recorded At
Seat ID
Total Connections
Collapse
Unique Seats
contractId
createdAt
email
firstName
lastName
seatId
Collapse
Activities
activityId
activityType
createdAt
CreatedDate
Hour Of Day
InMailOutcome
memberIdentityKey
seatId
Time Of Day
TimeOfDayOrder
Active Seats (no blanks)
TimeOfDay
Order
TimeOfDay
Collapse
1
*
Properties
Fields
All tables
+



datacamp Microsoft

Power BI for Business Intelligence

DAX Cheat Sheet

> Math & statistical functions

- **SUM(<column>)** Adds all the numbers in a column.
- **SUMX(<table>, <expression>)** Returns the sum of an expression evaluated for each row in a table.
- **AVERAGE(<column>)** Returns the average (arithmetic mean) of all the numbers in a column.
- **AVERAGEX(<table>, <expression>)** Calculates the average (arithmetic mean) of a set of expressions evaluated over a table.
- **MEDIAN(<column>)** Returns the median of a column.
- **MEDIANX(<table>, <expression>)** Calculates the median of a set of expressions evaluated over a table.
- **GEOMEAN(<column>)** Calculates the geometric mean of a column.
- **GEOMEANX(<table>, <expression>)** Calculates the geometric mean of a set of expressions evaluated over a table.
- **COUNT(<column>)** Returns the number of cells in a column that contain non-blank values.
- **COUNTX(<table>, <expression>)** Counts the number of rows from an expression that evaluates to a non-blank value.
- **DIVIDE(<numerator>, <denominator> [,<alternateresult>])** Performs division and returns alternate result or BLANK() on division by 0.
- **MIN(<column>)** Returns a minimum value of a column.
- **MAX(<column>)** Returns a maximum value of a column.
- **COUNTROWS([<table>])** Counts the number of rows in a table.
- **DISTINCTCOUNT(<column>)** Counts the number of distinct values in a column.
- **RANKX(<table>, <expression>[, <value>[, <order>[, <ties>]]])** Returns the ranking of a number in a list of numbers for each row in the table argument.

> Filter functions

- **FILTER(<table>, <filter>)** Returns a table that is a subset of another table or expression.
- **CALCULATE(<expression>[, <filter1> [, <filter2> [, …]]])** Evaluates an expression in a filter context.
- **HASONEVALUE(<columnName>)** Returns TRUE when the context for columnName has been filtered down to one distinct value only. Otherwise it is FALSE.
- **ALLNOBLANKROW(<table> | <column>[, <column>[, <column>[,…]]])** Returns a table that is a subset of another table or expression.
- **ALL([<table> | <column>[, <column>[, <column>[,…]]]])** Returns all the rows in a table, or all the values in a column, ignoring any filters that might have been applied.
- **ALLEXCEPT(<table>, <column>[, <column>[,..]])** Returns all the rows in a table except for those rows that are affected by the specified column filters.
- **REMOVEFILTERS([<table> | <column>][, <column>[, <column>[,…]]]])** Clear all filters from designated tables or columns.

> Logical functions

- **IF(<logical_test>, <value_if_true>[, <value_if_false>])** Checks a condition, and returns a certain value depending on whether it is true or false.
- **AND(<logical 1>, <logical 2>)** Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE. Otherwise, it returns FALSE.
- **OR(<logical 1>, <logical 2>)** Checks whether one of the arguments is TRUE to return TRUE. The function returns FALSE if both arguments are FALSE.
- **NOT(<logical>)** Changes TRUE to FALSE and vice versa.
- **SWITCH(<expression>, <value>, <result>[, <value>, <result>]…[, <else>])** Evaluates an expression against a list of values and returns one of possible results
- **IFERROR(<value>, <value_if_error>)** Returns value_if_error if the first expression is an error and the value of the expression itself otherwise.

> Date & time functions

- **CALENDAR(<start_date>, <end_date>)** Returns a table with a single column named "Date" that contains a contiguous set of dates.
- **DATE(<year>, <month>, <day>)** Returns the specified date in datetime format.
- **DATEDIFF(<date_1>, <date_2>, <interval>)** Returns the number of units between two dates as defined in **<interval>**.
- **DATEVALUE(<date_text>)** Converts a date in text to a date in datetime format.
- **DAY(<date>)** Returns a number from 1 to 31 representing the day of the month.
- **WEEKNUM(<date>)** Returns weeknumber in the year.
- **MONTH(<date>)** Returns a number from 1 to 12 representing a month.
- **QUARTER(<date>)** Returns a number from 1 to 4 representing a quarter.

> Time intelligence functions

- **DATEADD(<dates>, <number_of_intervals>, <interval>)** Moves a date by a specific interval.
- **DATESBETWEEN(<dates>, <date_1>, <date_2>)** Returns the dates between specified dates.
- **TOTALYTD(<expression>, <dates>[, <filter>][, <year_end_date>])** Evaluates the year-to-date value of the expression in the current context.
- **SAMEPERIODLASTYEAR(<dates>)** Returns a table that contains a column of dates shifted one year back in time.
- **STARTOFMONTH(<dates>) // ENDOFMONTH(<dates>)** Returns the start // end of the month.
- **STARTOFQUARTER(<dates>) // ENDOFQUARTER(<dates>)** Returns the start // end of the quarter.
- **STARTOFYEAR(<dates>) // ENDOFYEAR(<dates>)** Returns the start // end of the quarter.

> Relationship functions

- **CROSSFILTER(<left_column>, <right_column>, <crossfiltertype>)** Specifies the cross-filtering direction to be used in a calculation.
- **RELATED(<column>)** Returns a related value from another table.

> Table manipulation functions

- **SUMMARIZE(<table>, <groupBy_columnName>[, <groupBy_columnName>]…[, <name>, <expression>]…)** Returns a summary table for the requested totals over a set of groups.
- **DISTINCT(<table>)** Returns a table by removing duplicate rows from another table or expression.
- **ADDCOLUMNS(<table>, <name>, <expression>[, <name>, <expression>]…)** Adds calculated columns to the given table or table expression.
- **SELECTCOLUMNS(<table>, <name>, <expression>[, <name>, <expression>]…)** Selects calculated columns from the given table or table expression.
- **GROUPBY(<table> [, <groupBy_columnName>[, [<column_name>] [<expression>]]…)** Create a summary of the input table grouped by specific columns.
- **INTERSECT(<left_table>, <right_table>)** Returns the rows of the left-side table that appear in the right-side table.
- **NATURALINNERJOIN(<left_table>, <right_table>)** Joins two tables using an inner join.
- **NATURALLEFTOUTERJOIN(<left_table>, <right_table>)** Joins two tables using a left outer join.
- **UNION(<table>, <table>[, <table> [,…]])** Returns the union of tables with matching columns.

> Text functions

- **EXACT(<text_1>, <text_2>)** Checks if two strings are identical (**EXACT()** is case sensitive).
- **FIND(<text_tofind>, <in_text>)** Returns the starting position a text within another text (**FIND()** is case sensitive).
- **FORMAT(<value>, <format>)** Converts a value to a text in the specified number format.
- **LEFT(<text>, <num_chars>)** Returns the number of characters from the start of a string.
- **RIGHT(<text>, <num_chars>)** Returns the number of characters from the end of a string.
- **LEN(<text>)** Returns the number of characters in a string of text.
- **LOWER(<text>)** Converts all letters in a string to lowercase.
- **UPPER(<text>)** Converts all letters in a string to uppercase.
- **TRIM(<text>)** Remove all spaces from a text string.
- **CONCATENATE(<text_1>, <text_2>)** Joins two strings together into one string.
- **SUBSTITUTE(<text>, <old_text>, <new_text>, <instance_num>)** Replaces existing text with new text in a string.
- **REPLACE(<old_text>, <start_posotion>, <num_chars>, <new_text>)** Replaces part of a string with a new string.

> Information functions

- **COLUMNSTATISTICS()** Returns statistics regarding every column in every table. This function has no arguments.
- **NAMEOF(<value>)** Returns the column or measure name of a value.
- **ISBLANK(<value>) // ISERROR(<value>)** Returns whether the value is blank // an error.
- **ISLOGICAL(<value>)** Checks whether a value is logical or not.
- **ISNUMBER(<value>)** Checks whether a value is a number or not.
- **ISFILTERED(<table> | <column>)** Returns true when there are direct filters on a column.
- **ISCROSSFILTERED(<table> | <column>)** Returns true when there are crossfilters on a column.
- **USERPRINCIPALNAME()** Returns the user principal name or email address. This function has no arguments.

> DAX statements

- **VAR(<name> = <expression>)** Stores the result of an expression as a named variable. To return the variable, use RETURN after the variable is defined.
- **COLUMN(<table>[<column>] = <expression>)** Stores the result of an expression as a column in a table.
- **ORDER BY(<table>[<column>])** Defines the sort order of a column. Every column can be sorted in ascending (**ASC**) or descending (**DESC**) way.

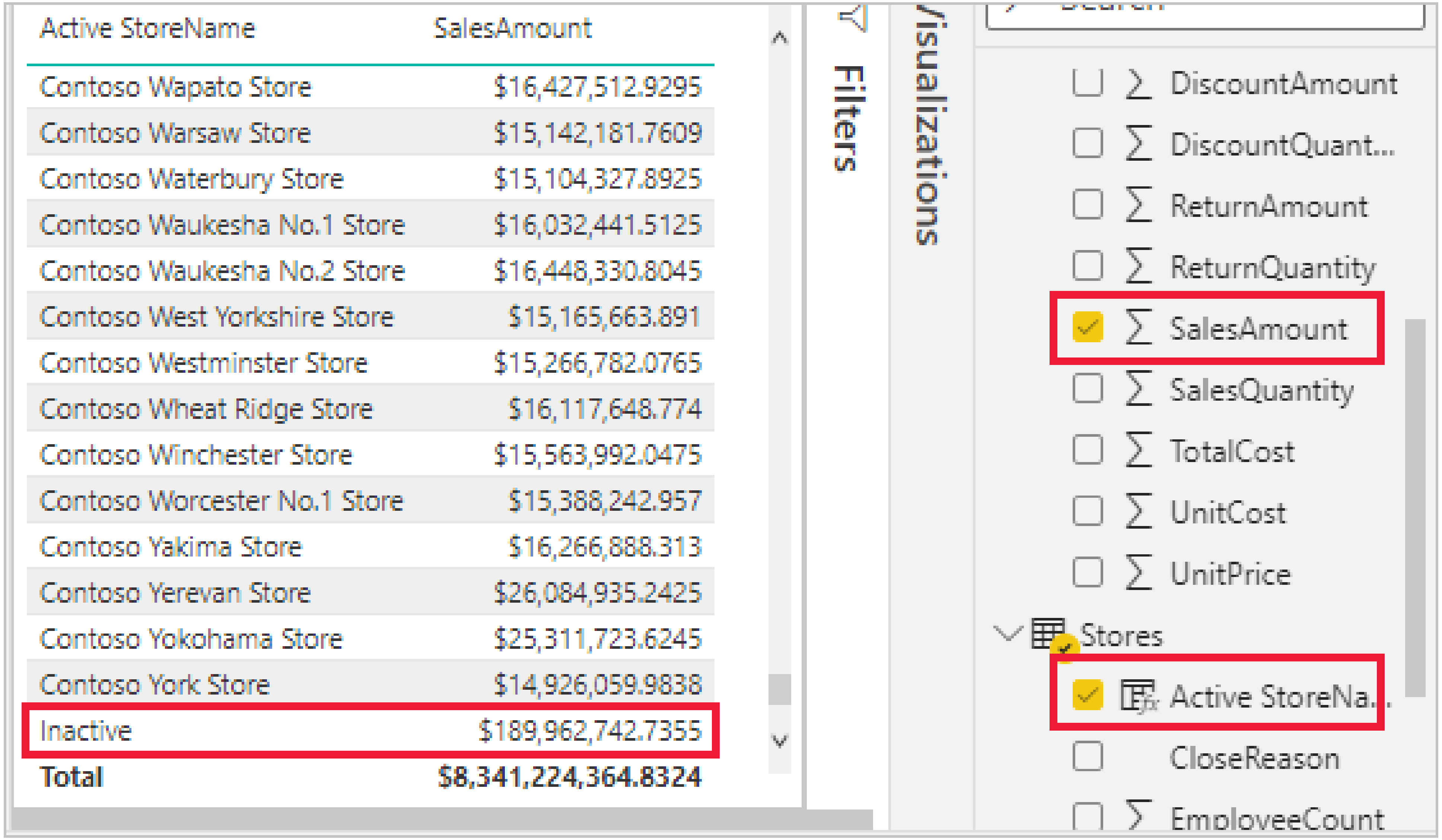
> DAX Operators

Comparison operators	Meaning
=	Equal to
= =	Strict equal to
>	Greater than
<	Smaller than
> =	Greater than or equal to
= <	Smaller than or equal to
< >	Not equal to

Text operator	Meaning	Example
&	Concatenates text values	Concatenates text values | [City]&", "&[State]

Logical operator	Meaning	Example
&&	**AND** condition	([City] = "Bru") && ([Return] = "Yes"))
||	**OR** condition	([City] = "Bru") || ([Return] = "Yes"))
IN {}	**OR** condition for each row	Product[Color] IN {"Red", "Blue", "Gold"}

Learn Data Skills Online at www.DataCamp.com

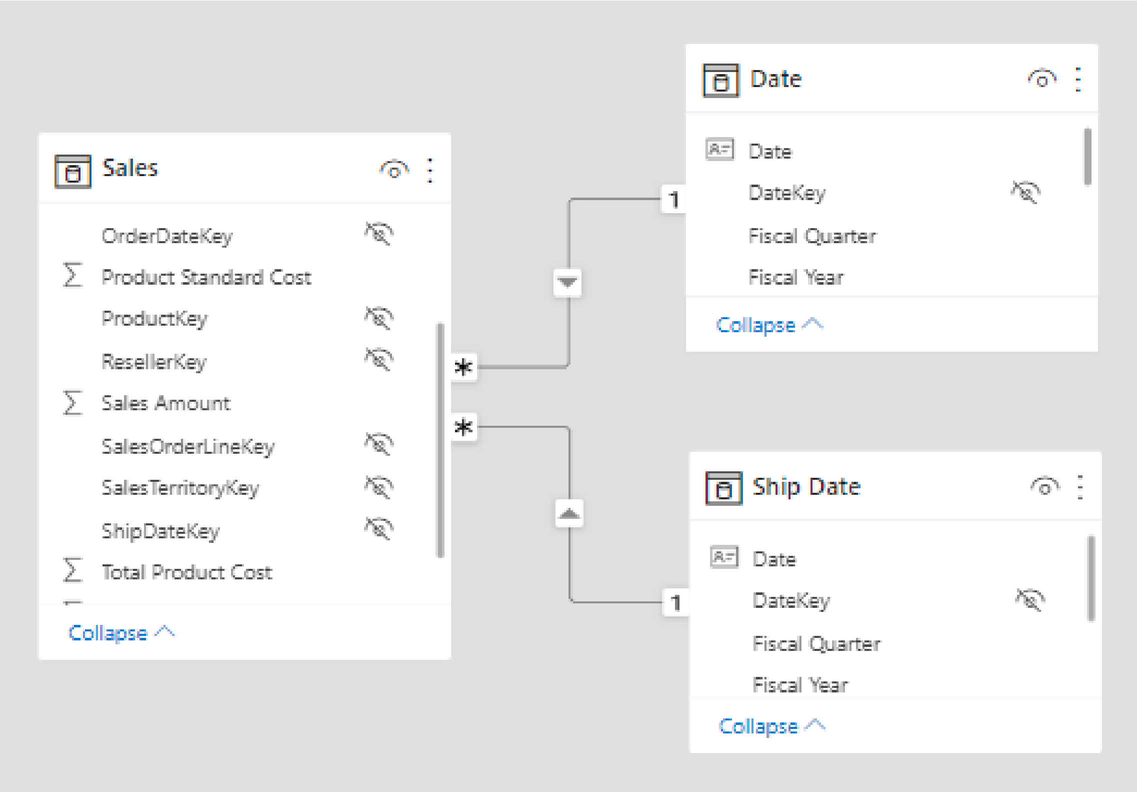


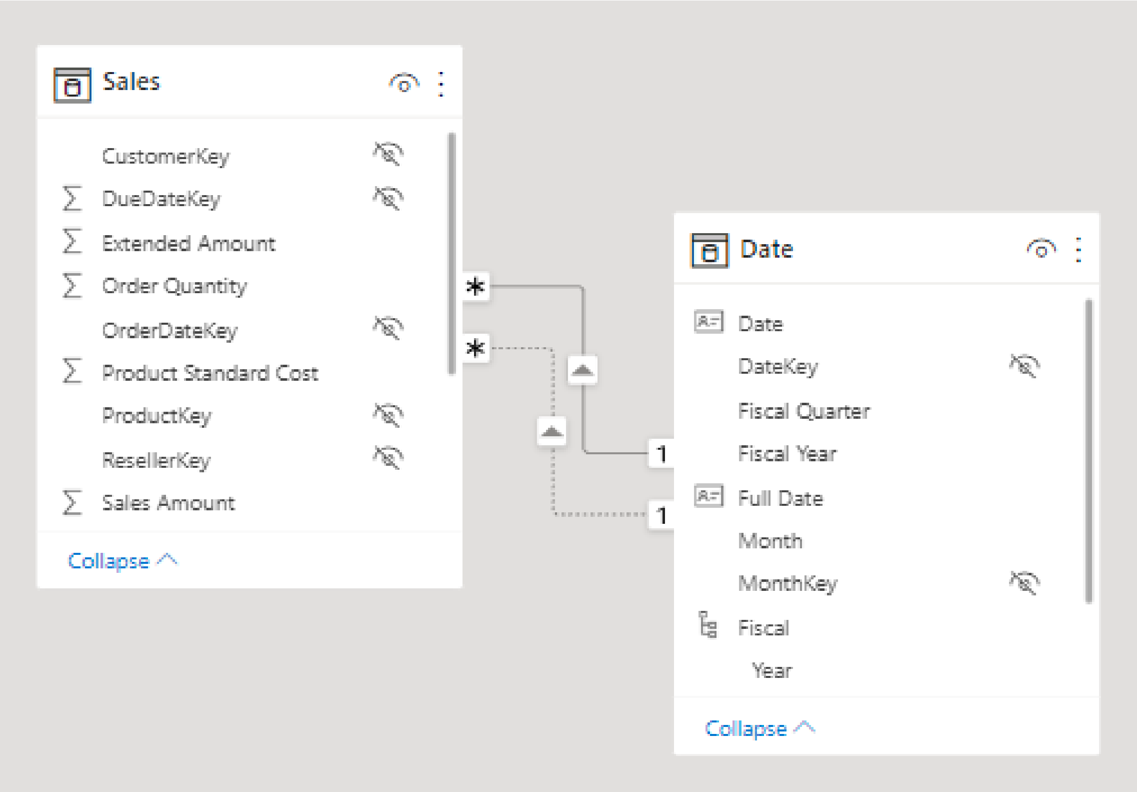
Active StoreName	SalesAmount
Contoso Wapato Store	$16,427,512.9295
Contoso Warsaw Store	$15,142,181.7609
Contoso Waterbury Store	$15,104,327.8925
Contoso Waukesha No.1 Store	$16,032,441.5125
Contoso Waukesha No.2 Store	$16,448,330.8045
Contoso West Yorkshire Store	$15,165,663.891
Contoso Westminster Store	$15,266,782.0765
Contoso Wheat Ridge Store	$16,117,648.774
Contoso Winchester Store	$15,563,992.0475
Contoso Worcester No.1 Store	$15,388,242.957
Contoso Yakima Store	$16,266,888.313
Contoso Yerevan Store	$26,084,935.2425
Contoso Yokohama Store	$25,311,723.6245
Contoso York Store	$14,926,059.9838
Inactive	$189,962,742.7355
Total	**$8,341,224,364.8324**

□ ∑ DiscountAmount
□ ∑ DiscountQuant...
□ ∑ ReturnAmount
□ ∑ ReturnQuantity
☑ ∑ SalesAmount
□ ∑ SalesQuantity
□ ∑ TotalCost
□ ∑ UnitCost
□ ∑ UnitPrice

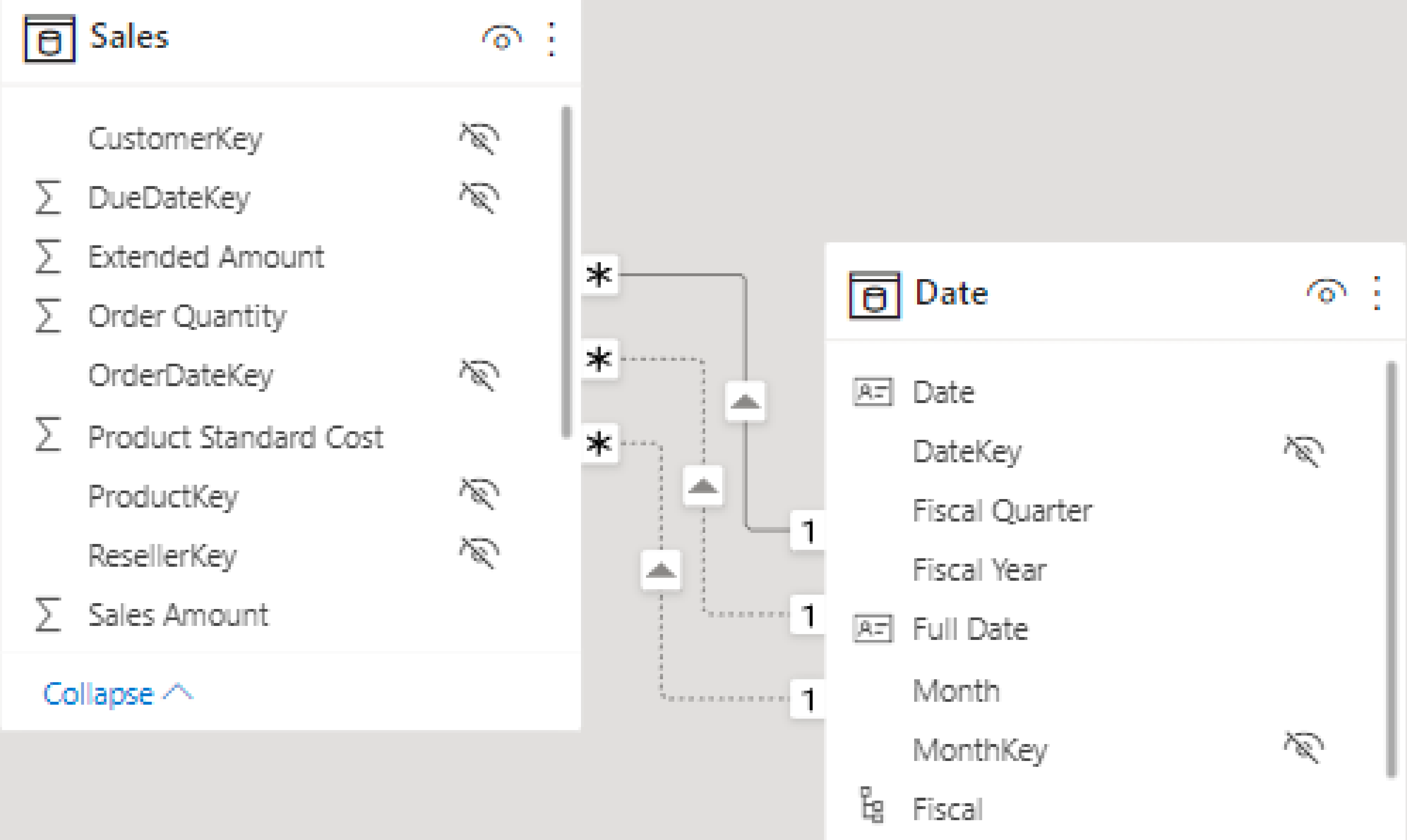
Stores

☑ Active StoreNa...
□ CloseReason
□ ∑ EmployeeCount

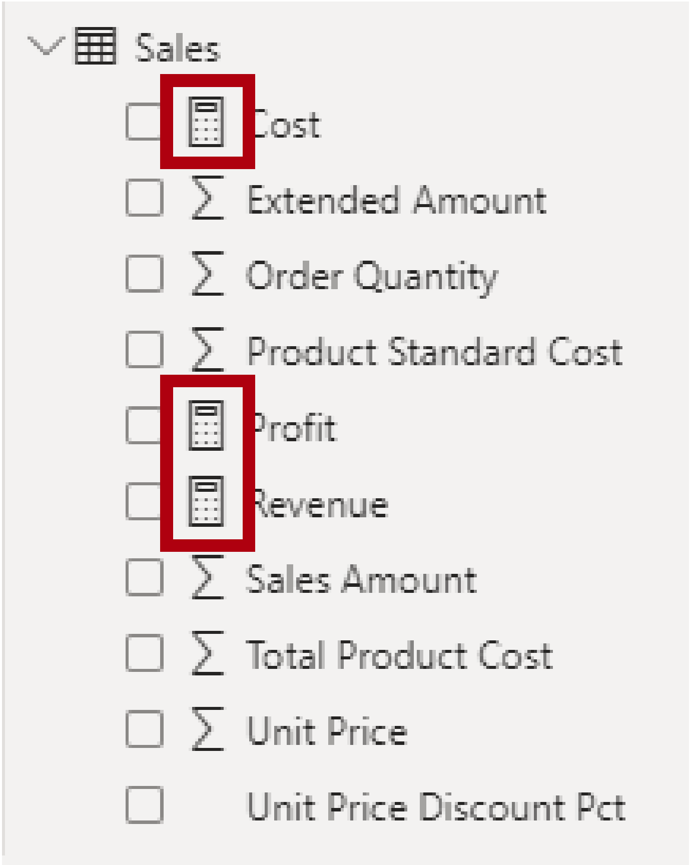




DIMENSIONES REALIZADORAS DE ROLES

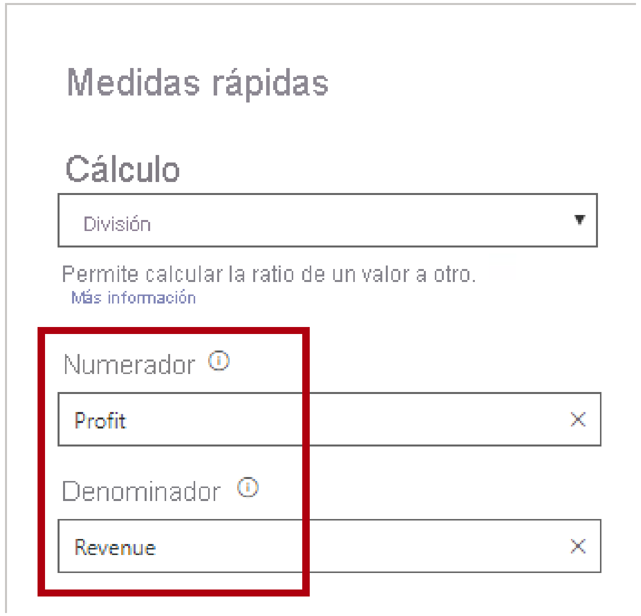

Sales
CustomerKey
DueDateKey
Extended Amount
Order Quantity
OrderDateKey
Product Standard Cost
ProductKey
ResellerKey
Sales Amount
Collapse
*
*
*
1
1
1
Date
Date
DateKey
Fiscal Quarter
Fiscal Year
Full Date
Month
MonthKey
Fiscal

LABORATORIO 1



Sales

- □ 🖩 Cost
- □ Σ Extended Amount
- □ Σ Order Quantity
- □ Σ Product Standard Cost
- □ 🖩 Profit
- □ 🖩 Revenue
- □ Σ Sales Amount
- □ Σ Total Product Cost
- □ Σ Unit Price
- □ Unit Price Discount Pct



Medidas rápidas

Cálculo

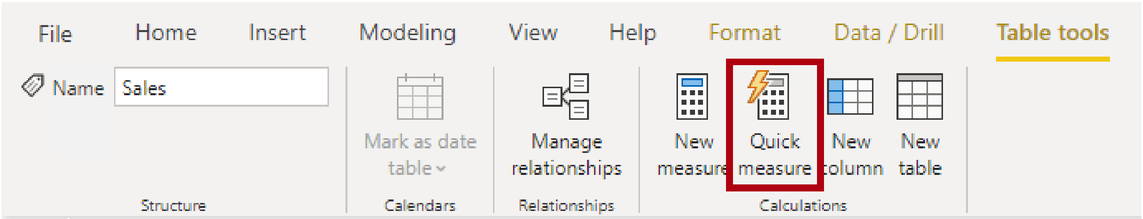
División

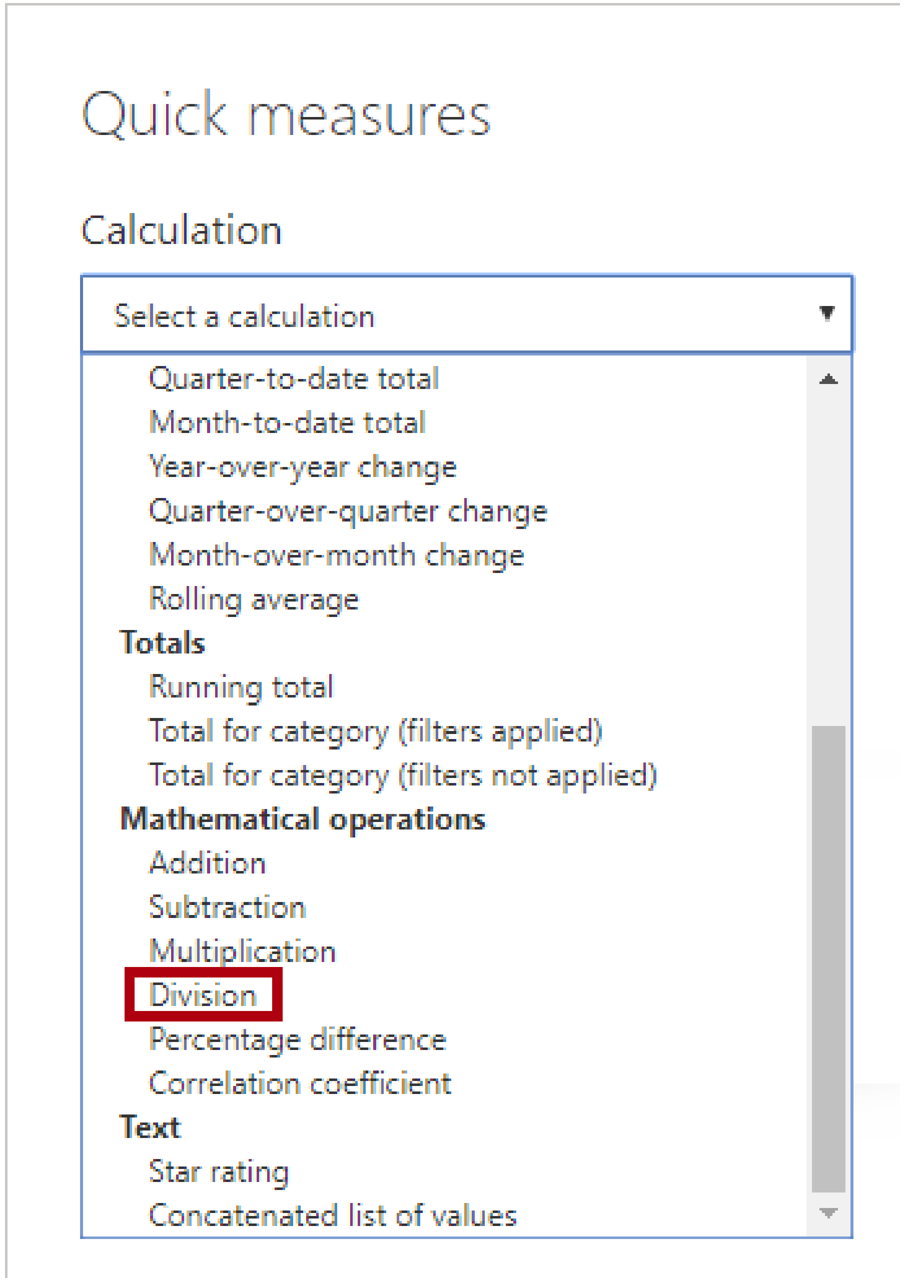
Permite calcular la ratio de un valor a otro.

Más información

Numerador: Profit

Denominador: Revenue





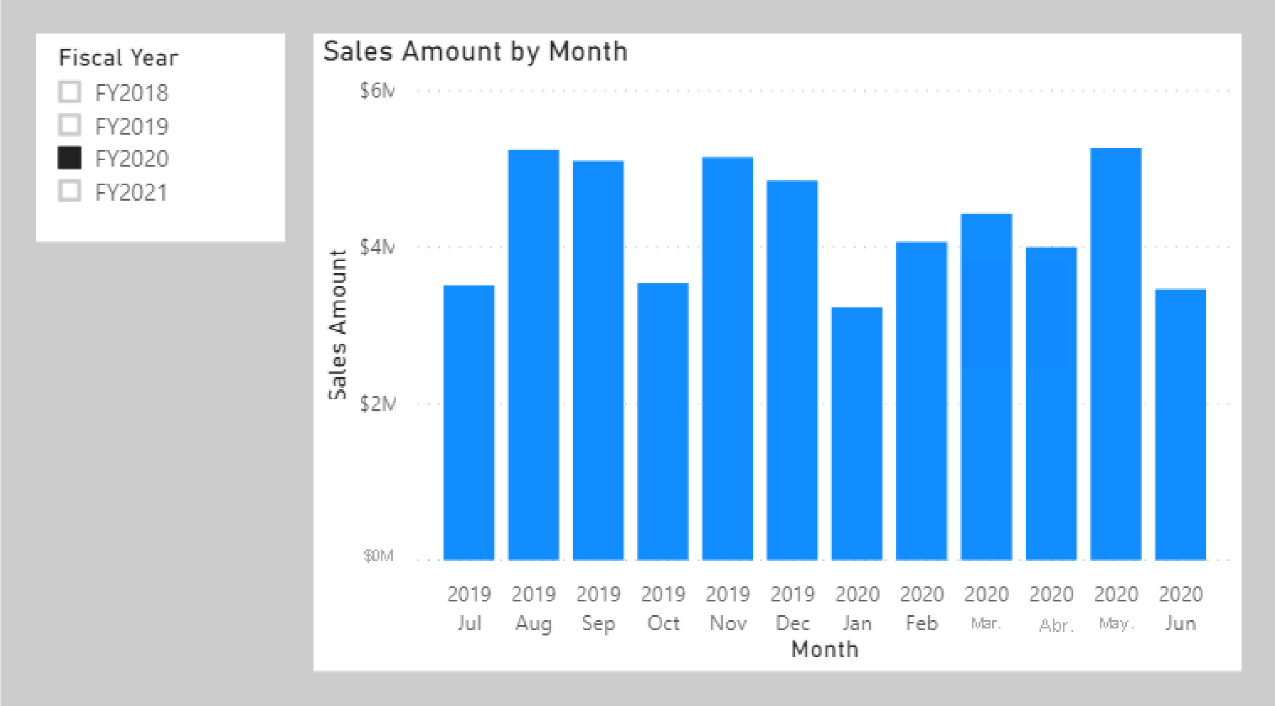
Quick measures

Calculation

Select a calculation

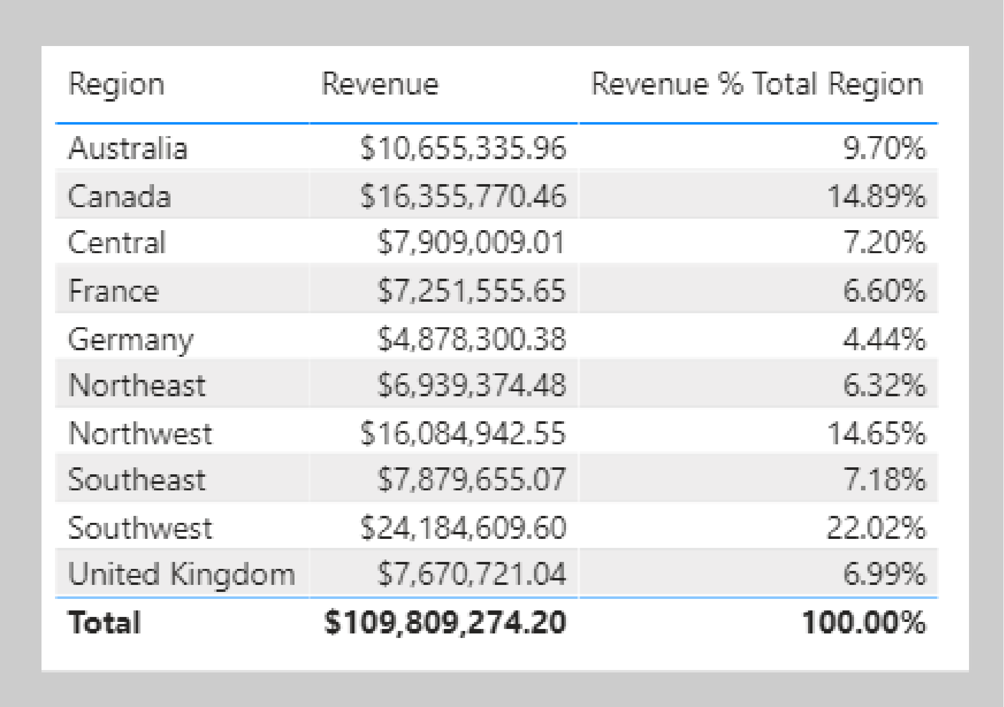
LABORATORIO 2

CONTEXTO



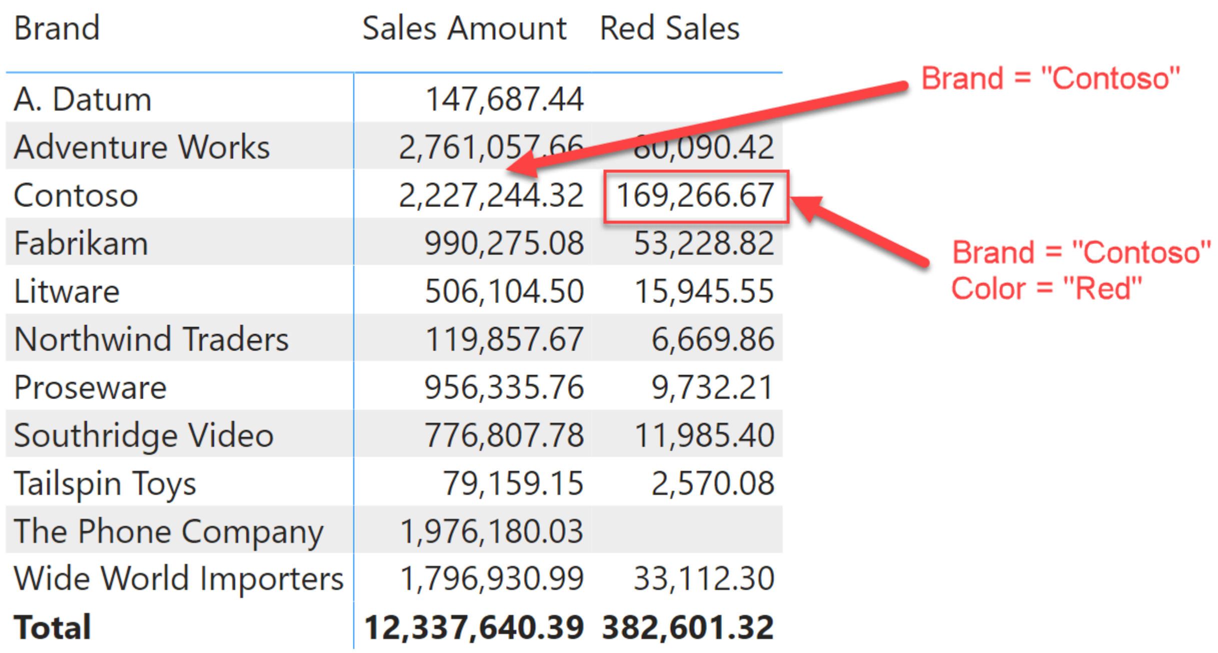
Fiscal Year

- □ FY2018
- □ FY2019
- ■ FY2020
- □ FY2021



Region	Revenue	Revenue % Total Region
Australia	$10,655,335.96	9.70%
Canada	$16,355,770.46	14.89%
Central	$7,909,009.01	7.20%
France	$7,251,555.65	6.60%
Germany	$4,878,300.38	4.44%
Northeast	$6,939,374.48	6.32%
Northwest	$16,084,942.55	14.65%
Southeast	$7,879,655.07	7.18%
Southwest	$24,184,609.60	22.02%
United Kingdom	$7,670,721.04	6.99%
Total	**$109,809,274.20**	**100.00%**

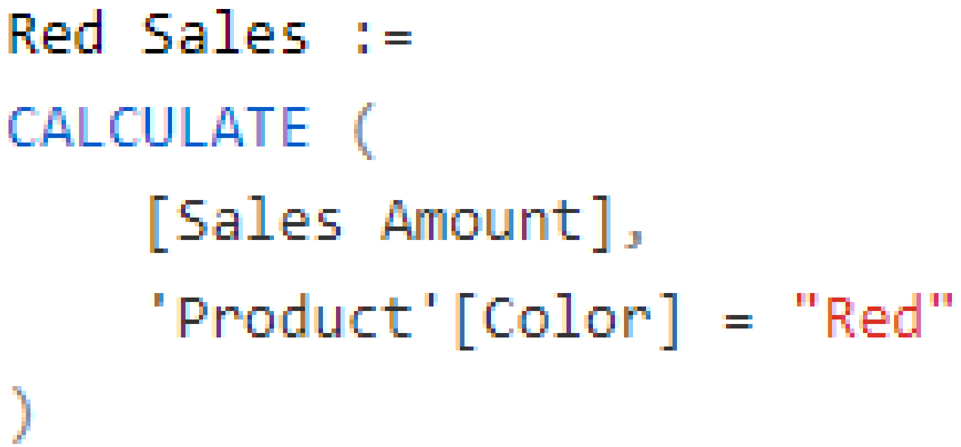
CALCULATE



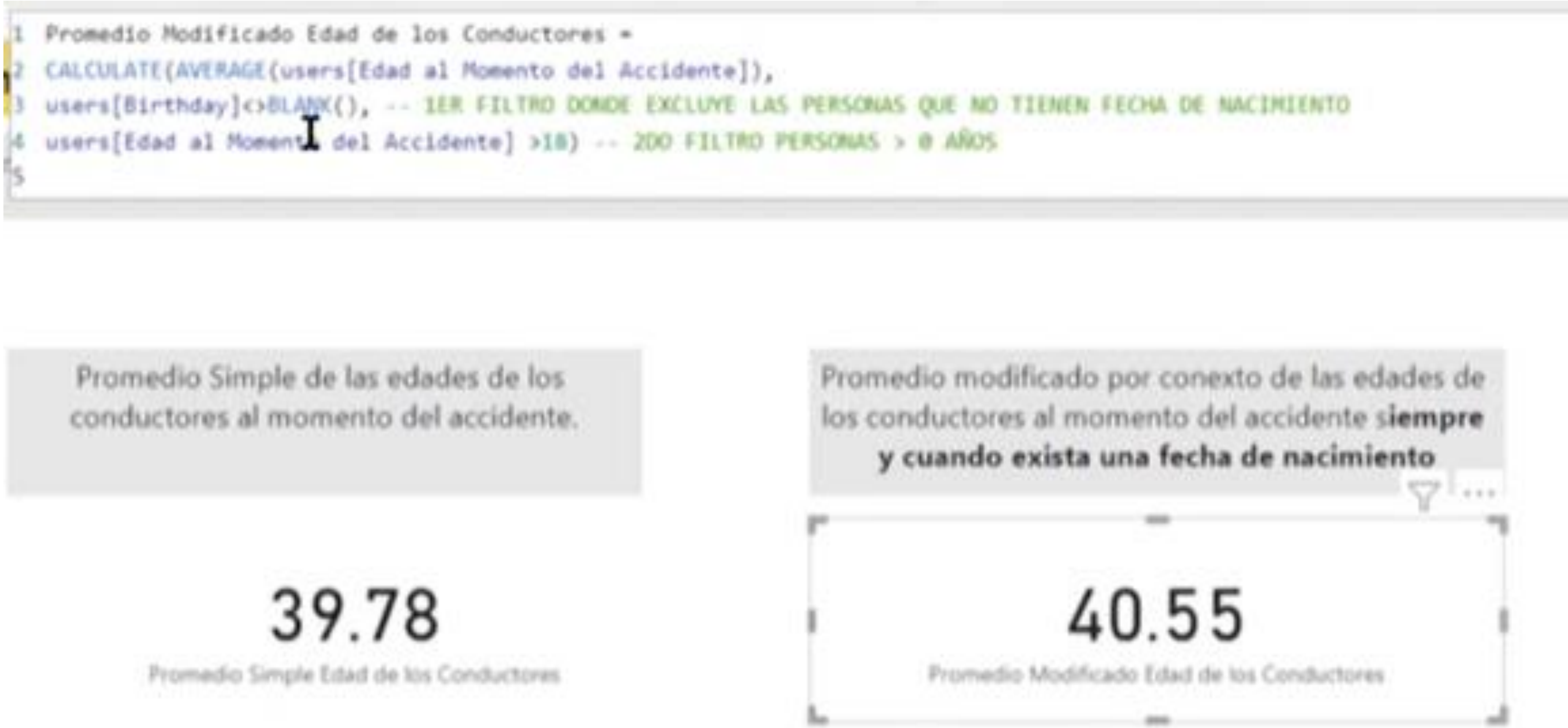
Brand	Sales Amount	Red Sales
A. Datum	147,687.44	
Adventure Works	2,761,057.66	60,090.42
Contoso	2,227,244.32	169,266.67
Fabrikam	990,275.08	53,228.82
Litware	506,104.50	15,945.55
Northwind Traders	119,857.67	6,669.86
Proseware	956,335.76	9,732.21
Southridge Video	776,807.78	11,985.40
Tailspin Toys	79,159.15	2,570.08
The Phone Company	1,976,180.03	
Wide World Importers	1,796,930.99	33,112.30
Total	**12,337,640.39**	**382,601.32**

Brand = "Contoso"

Brand = "Contoso"
Color = "Red"



```
Red Sales :=
CALCULATE (
    [Sales Amount],
    'Product'[Color] = "Red"
)
```



```
Promedio Modificado Edad de los Conductores =
CALCULATE(AVERAGE(users[Edad al Momento del Accidente]),
users[Birthday]<>BLANK(), -- 1ER FILTRO DONDE EXCLUYE LAS PERSONAS QUE NO TIENEN FECHA DE NACIMIENTO
users[Edad al Momento del Accidente] >18) -- 2DO FILTRO PERSONAS > 0 AÑOS
```

Promedio Simple de las edades de los conductores al momento del accidente.

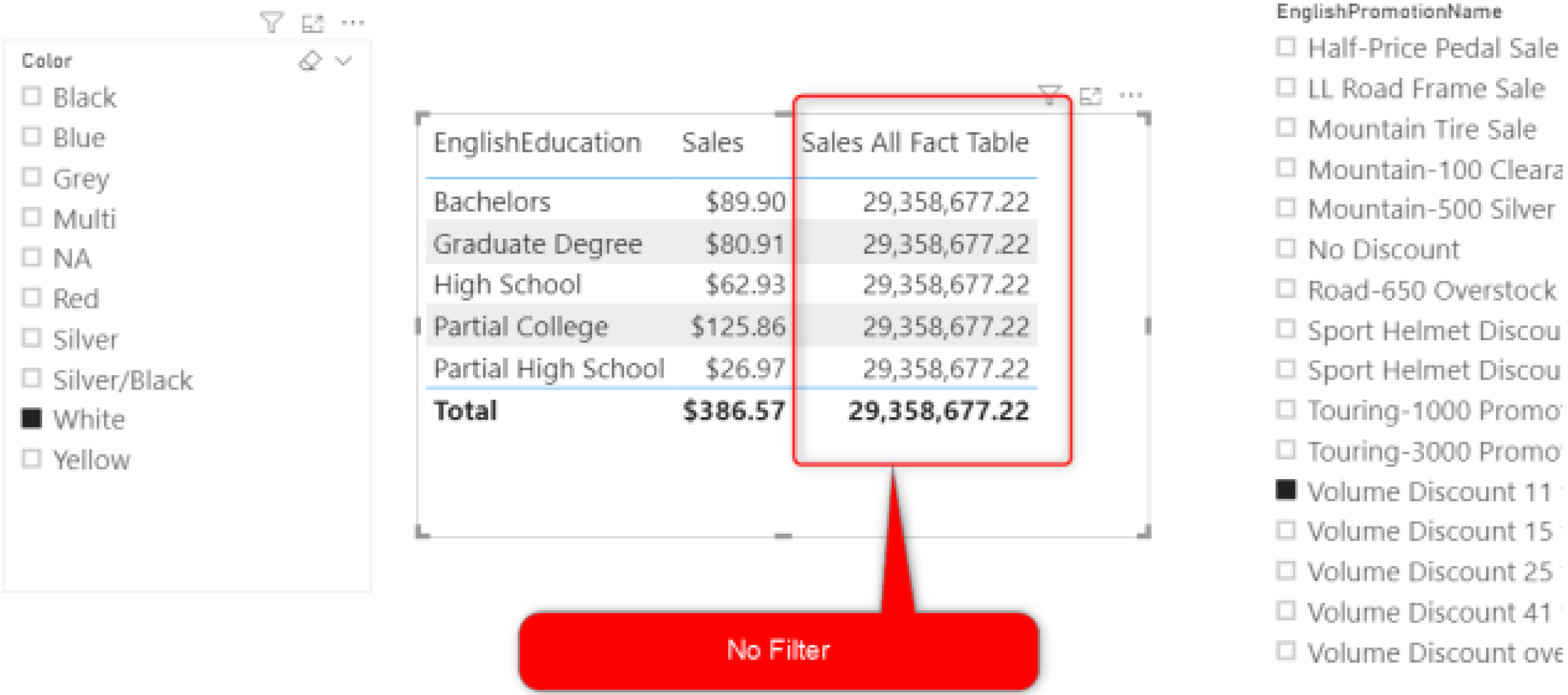
39.78

Promedio Simple Edad de los Conductores

Promedio modificado por conexto de las edades de los conductores al momento del accidente s**iempre y cuando exista una fecha de nacimiento**

40.55

Promedio Modificado Edad de los Conductores



Color

- ☐ Black
- ☐ Blue
- ☐ Grey
- ☐ Multi
- ☐ NA
- ☐ Red
- ☐ Silver
- ☐ Silver/Black
- ■ White
- ☐ Yellow

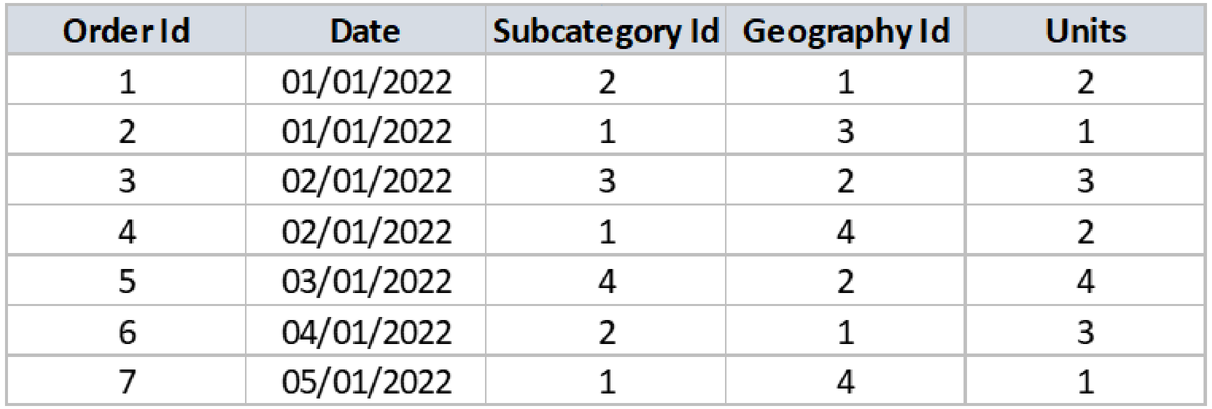
EnglishEducation	Sales	Sales All Fact Table
Bachelors	$89.90	29,358,677.22
Graduate Degree	$80.91	29,358,677.22
High School	$62.93	29,358,677.22
Partial College	$125.86	29,358,677.22
Partial High School	$26.97	29,358,677.22
Total	**$386.57**	**29,358,677.22**

No Filter

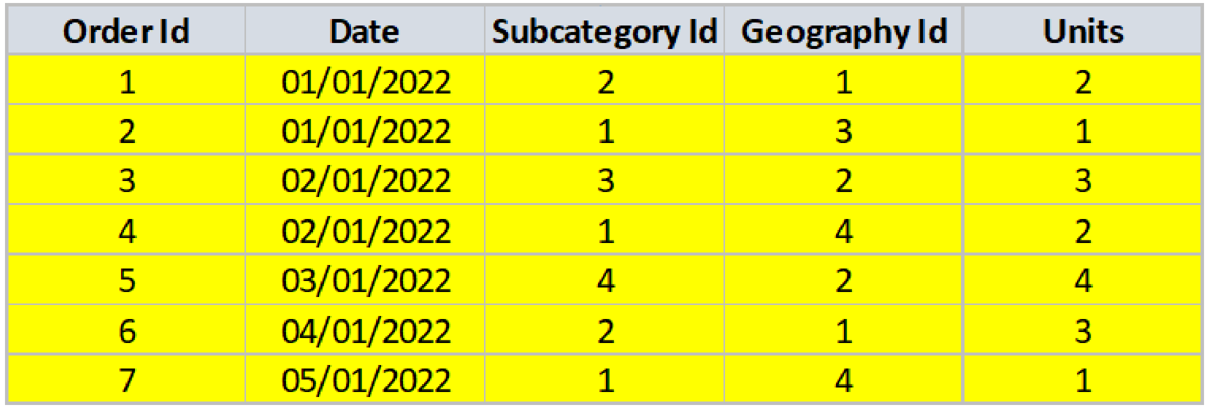
EnglishPromotionName

- ☐ Half-Price Pedal Sale
- ☐ LL Road Frame Sale
- ☐ Mountain Tire Sale
- ☐ Mountain-100 Cleara
- ☐ Mountain-500 Silver
- ☐ No Discount
- ☐ Road-650 Overstock
- ☐ Sport Helmet Discou
- ☐ Sport Helmet Discou
- ☐ Touring-1000 Promo
- ☐ Touring-3000 Promo
- ■ Volume Discount 11
- ☐ Volume Discount 15
- ☐ Volume Discount 25
- ☐ Volume Discount 41
- ☐ Volume Discount ove

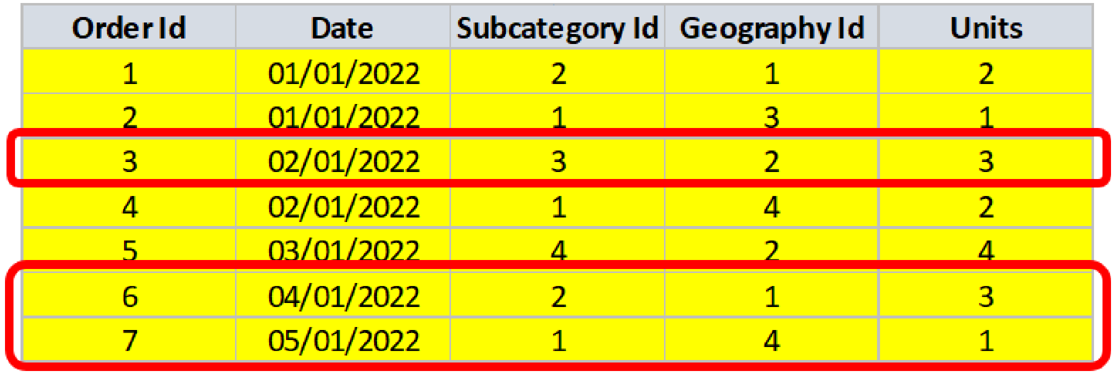
CONTEXTO DE FILTRO



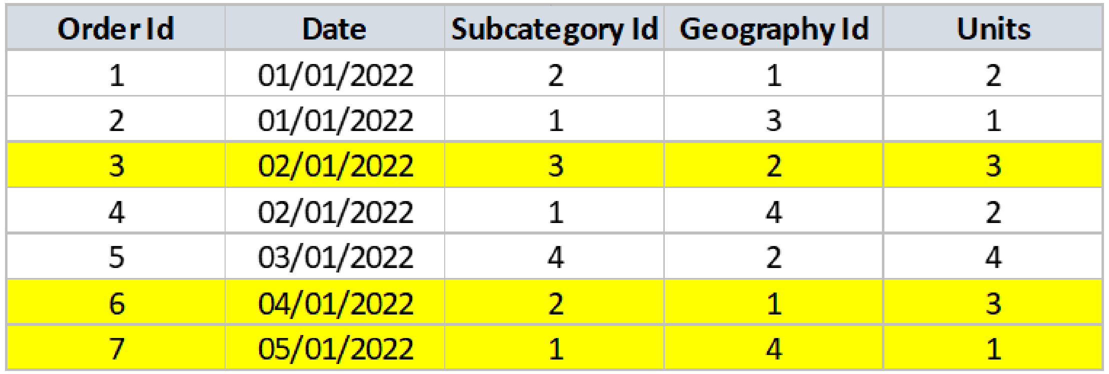
Order Id	Date	Subcategory Id	Geography Id	Units
1	01/01/2022	2	1	2
2	01/01/2022	1	3	1
3	02/01/2022	3	2	3
4	02/01/2022	1	4	2
5	03/01/2022	4	2	4
6	04/01/2022	2	1	3
7	05/01/2022	1	4	1



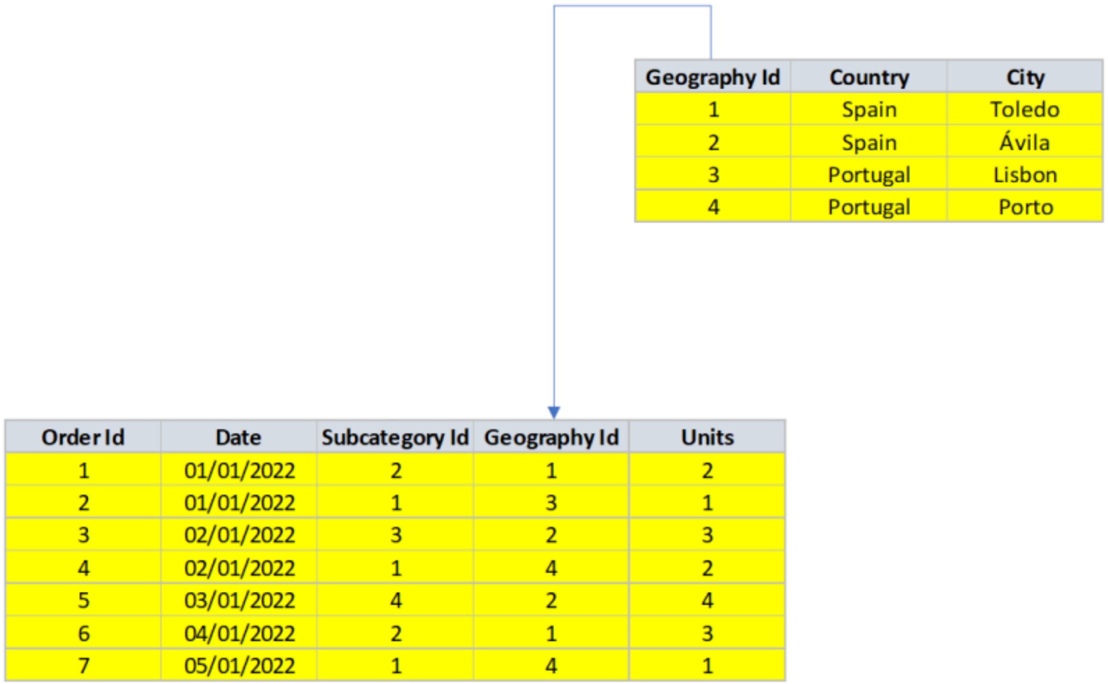
Order Id	Date	Subcategory Id	Geography Id	Units
1	01/01/2022	2	1	2
2	01/01/2022	1	3	1
3	02/01/2022	3	2	3
4	02/01/2022	1	4	2
5	03/01/2022	4	2	4
6	04/01/2022	2	1	3
7	05/01/2022	1	4	1



Order Id	Date	Subcategory Id	Geography Id	Units
1	01/01/2022	2	1	2
2	01/01/2022	1	3	1
3	02/01/2022	3	2	3
4	02/01/2022	1	4	2
5	03/01/2022	4	2	4
6	04/01/2022	2	1	3
7	05/01/2022	1	4	1

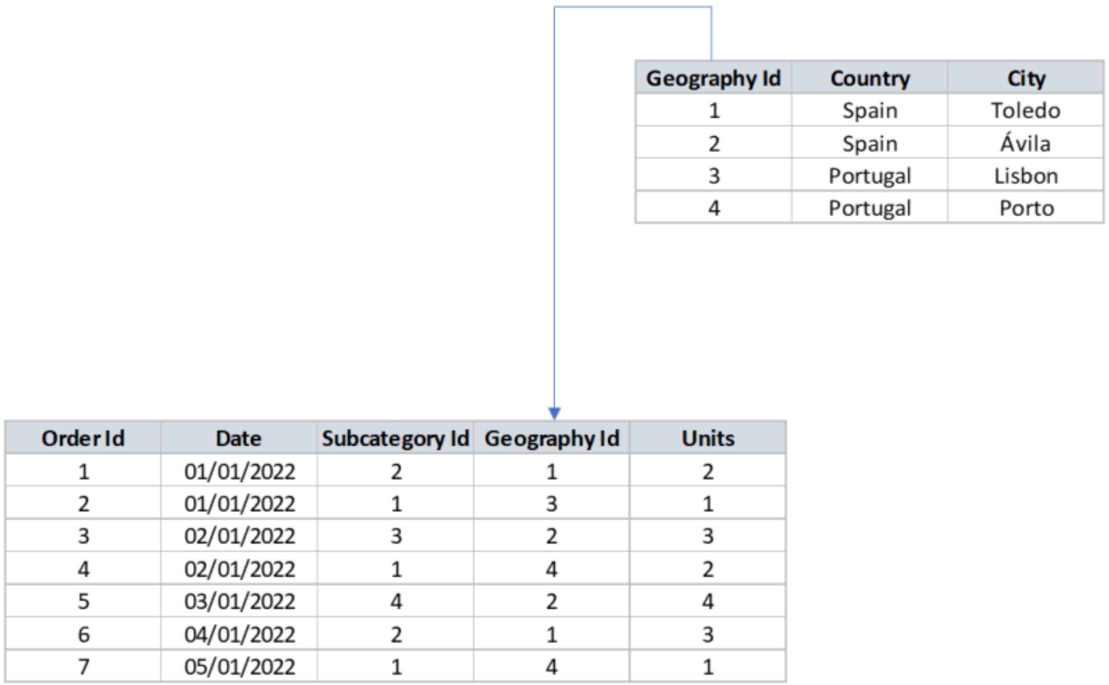


Order Id	Date	Subcategory Id	Geography Id	Units
1	01/01/2022	2	1	2
2	01/01/2022	1	3	1
3	02/01/2022	3	2	3
4	02/01/2022	1	4	2
5	03/01/2022	4	2	4
6	04/01/2022	2	1	3
7	05/01/2022	1	4	1



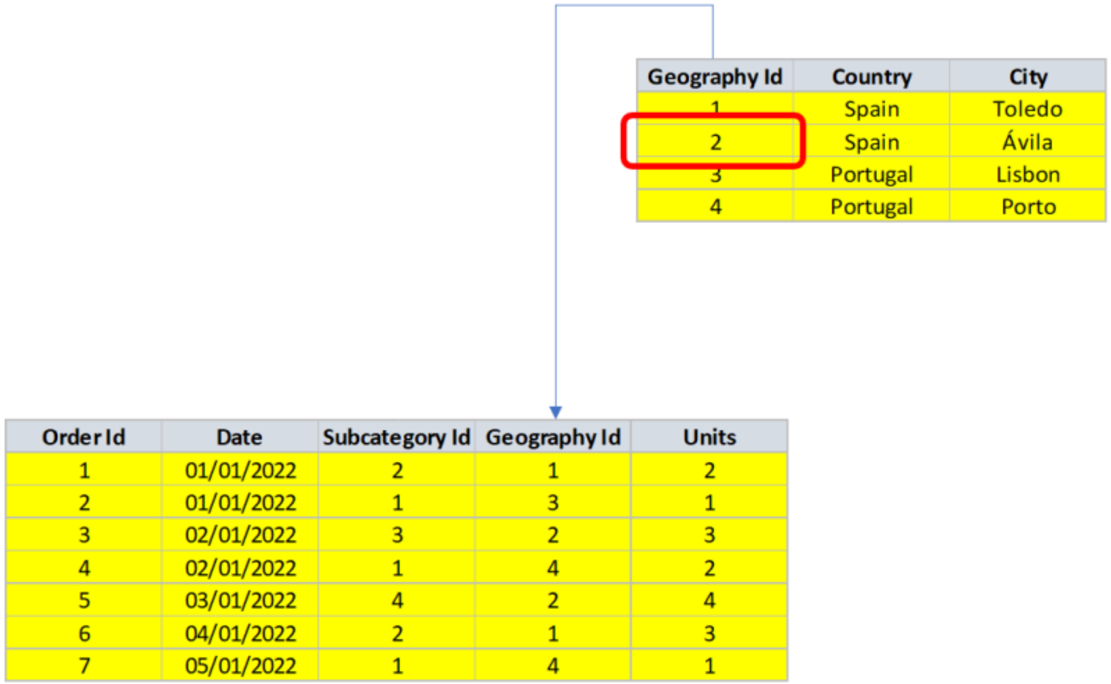
Geography Id	Country	City
1	Spain	Toledo
2	Spain	Ávila
3	Portugal	Lisbon
4	Portugal	Porto

Order Id	Date	Subcategory Id	Geography Id	Units
1	01/01/2022	2	1	2
2	01/01/2022	1	3	1
3	02/01/2022	3	2	3
4	02/01/2022	1	4	2
5	03/01/2022	4	2	4
6	04/01/2022	2	1	3
7	05/01/2022	1	4	1



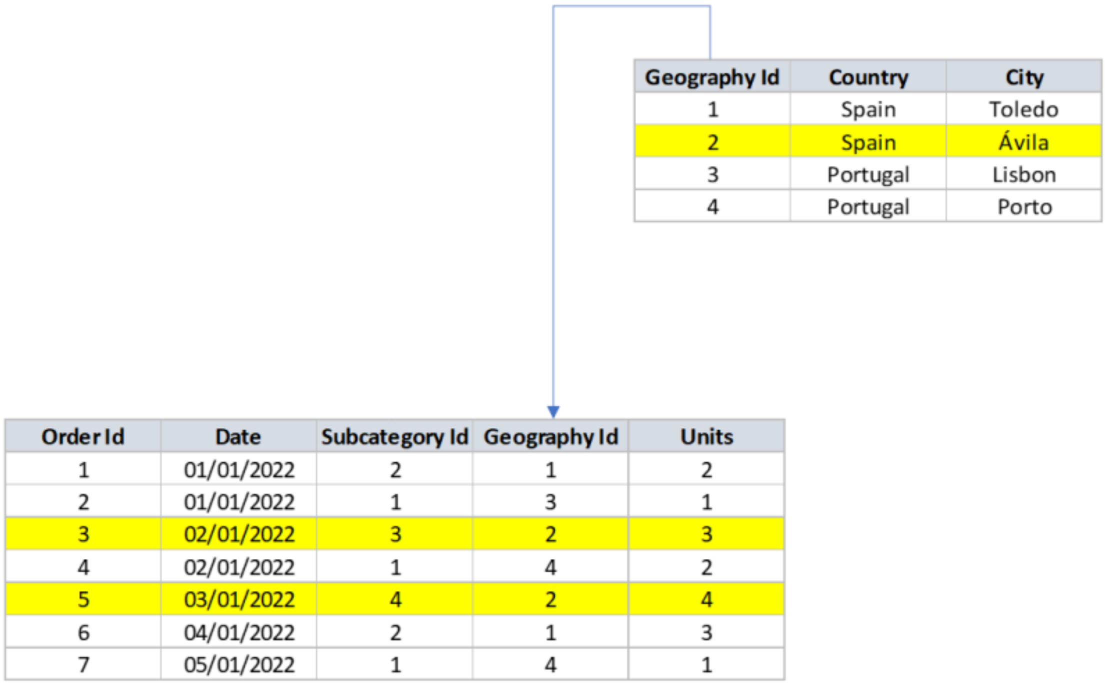
Geography Id	Country	City
1	Spain	Toledo
2	Spain	Ávila
3	Portugal	Lisbon
4	Portugal	Porto

Order Id	Date	Subcategory Id	Geography Id	Units
1	01/01/2022	2	1	2
2	01/01/2022	1	3	1
3	02/01/2022	3	2	3
4	02/01/2022	1	4	2
5	03/01/2022	4	2	4
6	04/01/2022	2	1	3
7	05/01/2022	1	4	1



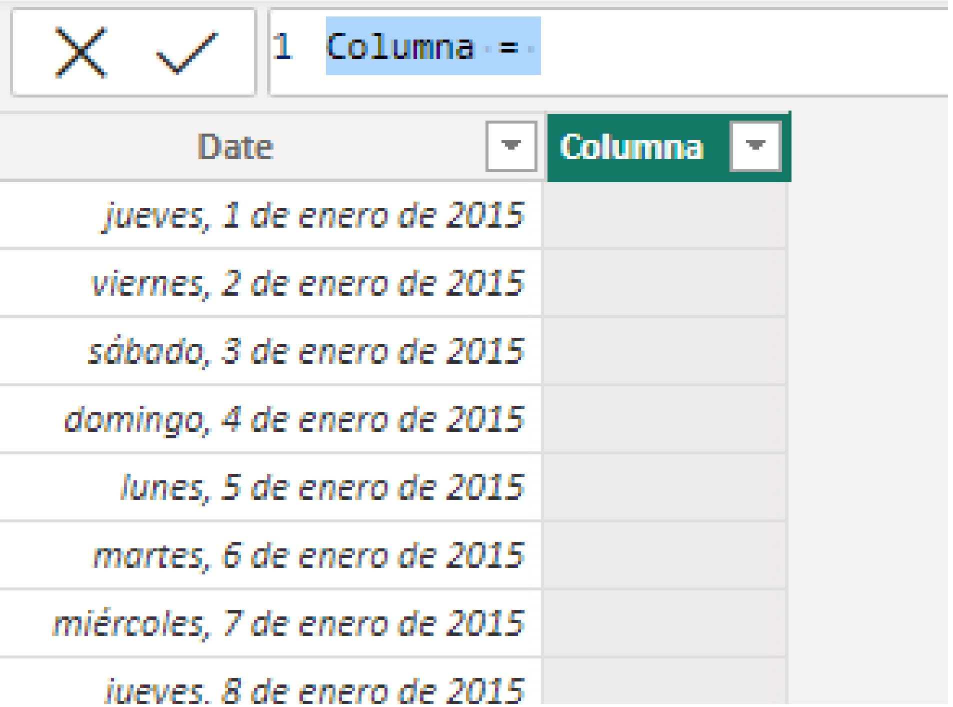
Geography Id	Country	City
1	Spain	Toledo
2	Spain	Ávila
3	Portugal	Lisbon
4	Portugal	Porto

Order Id	Date	Subcategory Id	Geography Id	Units
1	01/01/2022	2	1	2
2	01/01/2022	1	3	1
3	02/01/2022	3	2	3
4	02/01/2022	1	4	2
5	03/01/2022	4	2	4
6	04/01/2022	2	1	3
7	05/01/2022	1	4	1



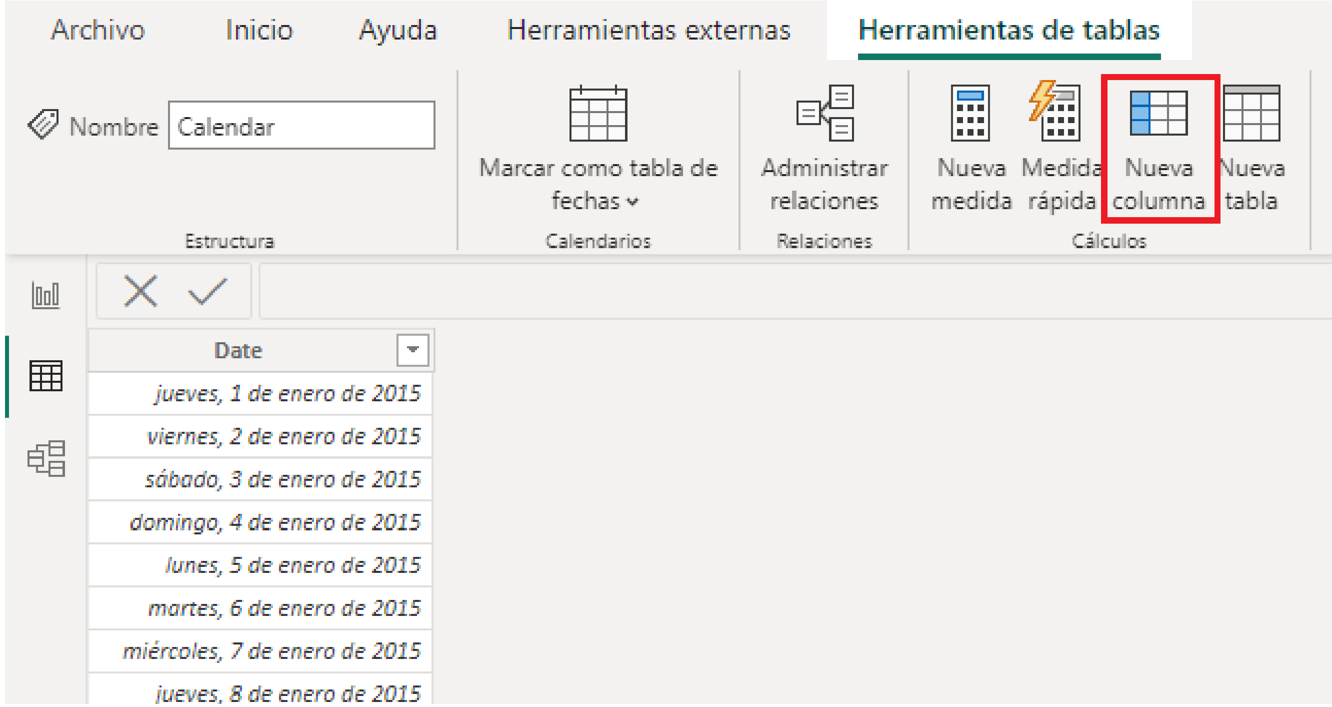
Geography Id	Country	City
1	Spain	Toledo
2	Spain	Ávila
3	Portugal	Lisbon
4	Portugal	Porto

Order Id	Date	Subcategory Id	Geography Id	Units
1	01/01/2022	2	1	2
2	01/01/2022	1	3	1
3	02/01/2022	3	2	3
4	02/01/2022	1	4	2
5	03/01/2022	4	2	4
6	04/01/2022	2	1	3
7	05/01/2022	1	4	1



```
1 Columna =
```

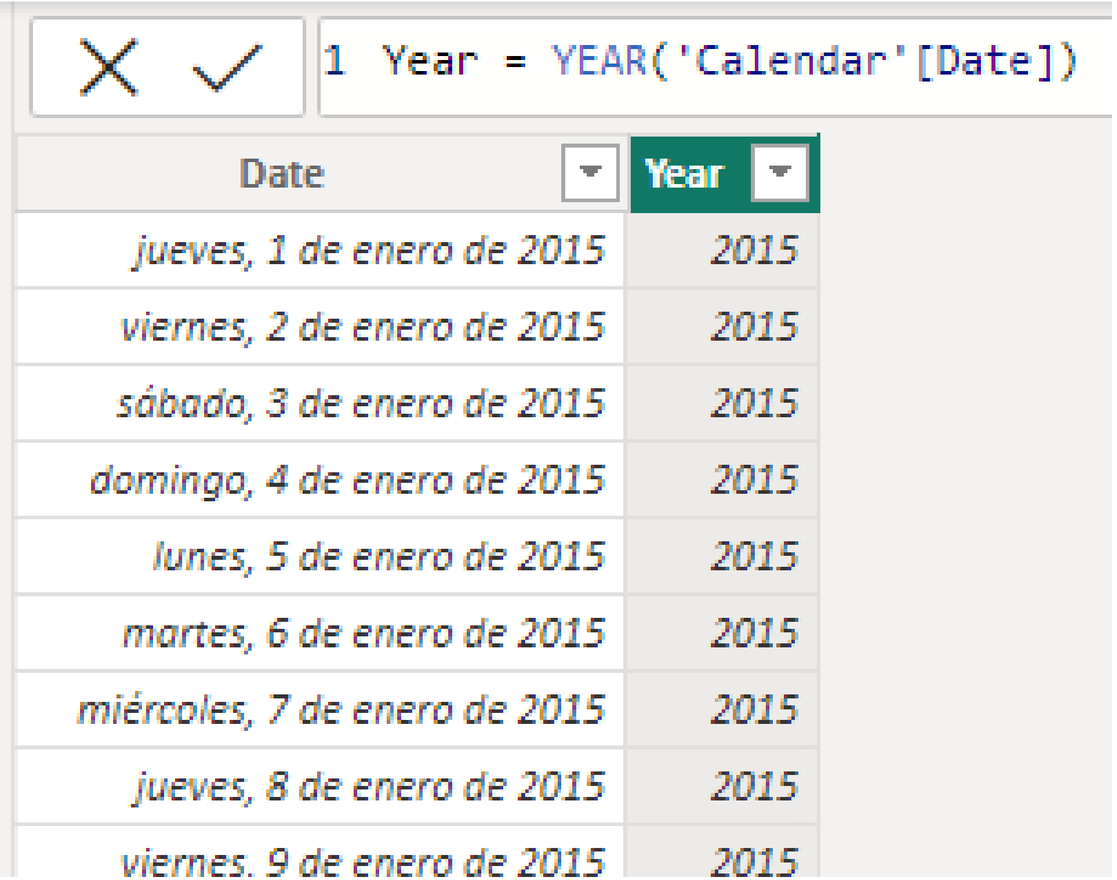
Date	Columna
jueves, 1 de enero de 2015	
viernes, 2 de enero de 2015	
sábado, 3 de enero de 2015	
domingo, 4 de enero de 2015	
lunes, 5 de enero de 2015	
martes, 6 de enero de 2015	
miércoles, 7 de enero de 2015	
jueves, 8 de enero de 2015	



Nombre: Calendar

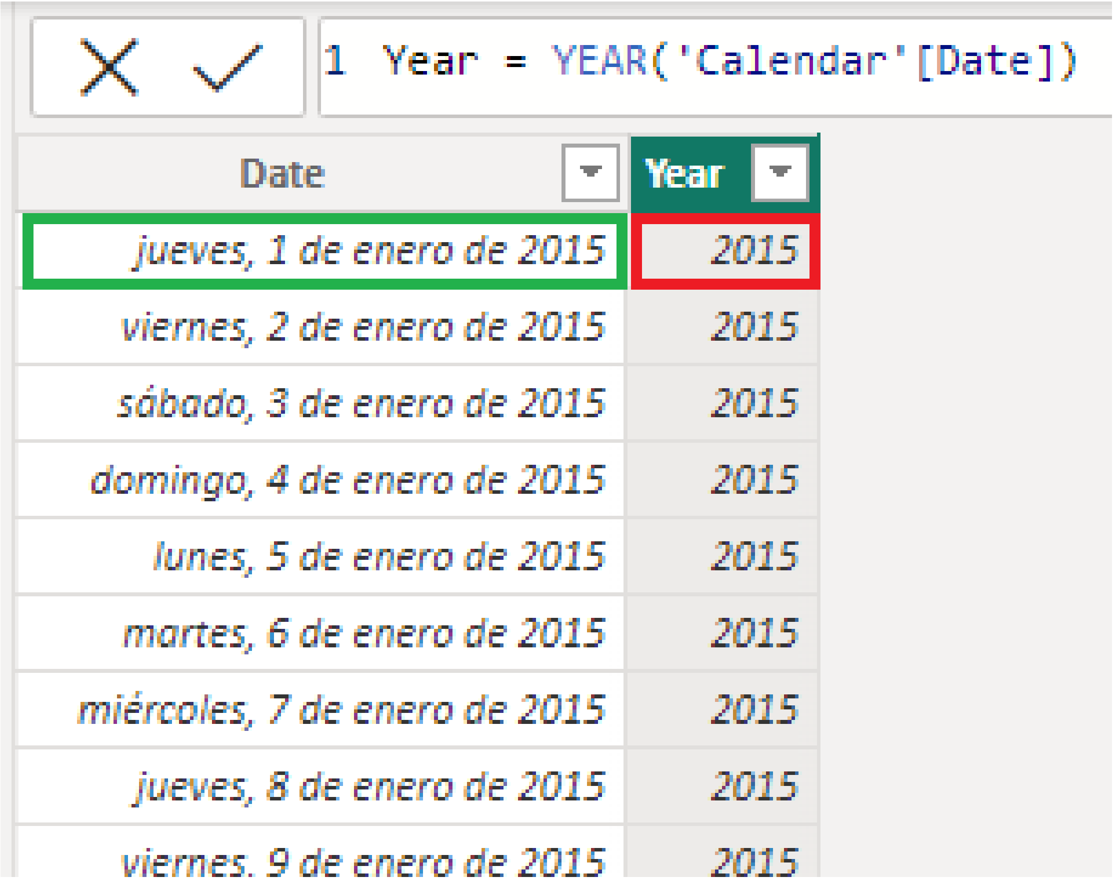
Date
jueves, 1 de enero de 2015
viernes, 2 de enero de 2015
sábado, 3 de enero de 2015
domingo, 4 de enero de 2015
lunes, 5 de enero de 2015
martes, 6 de enero de 2015
miércoles, 7 de enero de 2015
jueves, 8 de enero de 2015

CONTEXTO DE FILA



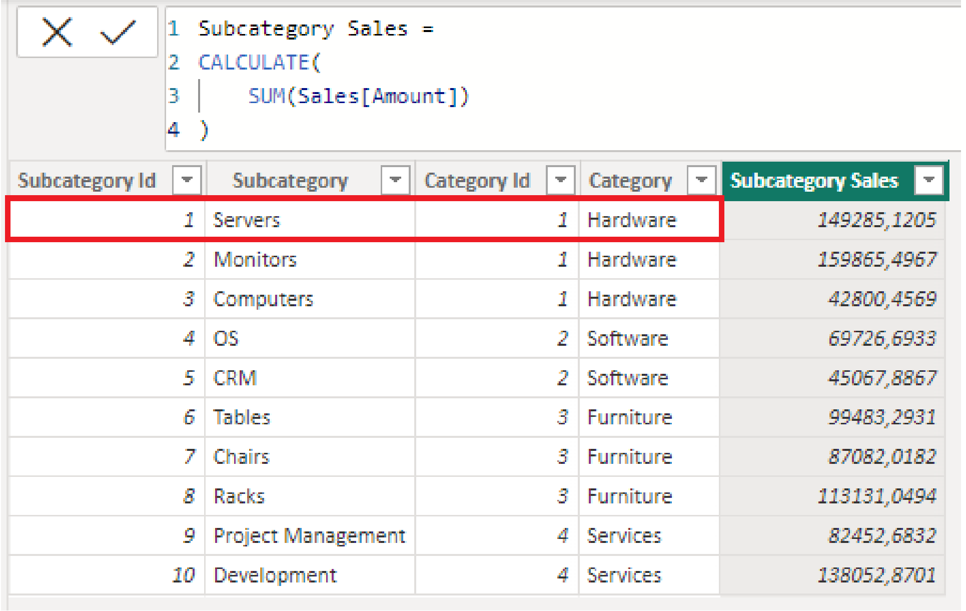
```
Year = YEAR('Calendar'[Date])
```

Date	Year
jueves, 1 de enero de 2015	2015
viernes, 2 de enero de 2015	2015
sábado, 3 de enero de 2015	2015
domingo, 4 de enero de 2015	2015
lunes, 5 de enero de 2015	2015
martes, 6 de enero de 2015	2015
miércoles, 7 de enero de 2015	2015
jueves, 8 de enero de 2015	2015
viernes, 9 de enero de 2015	2015



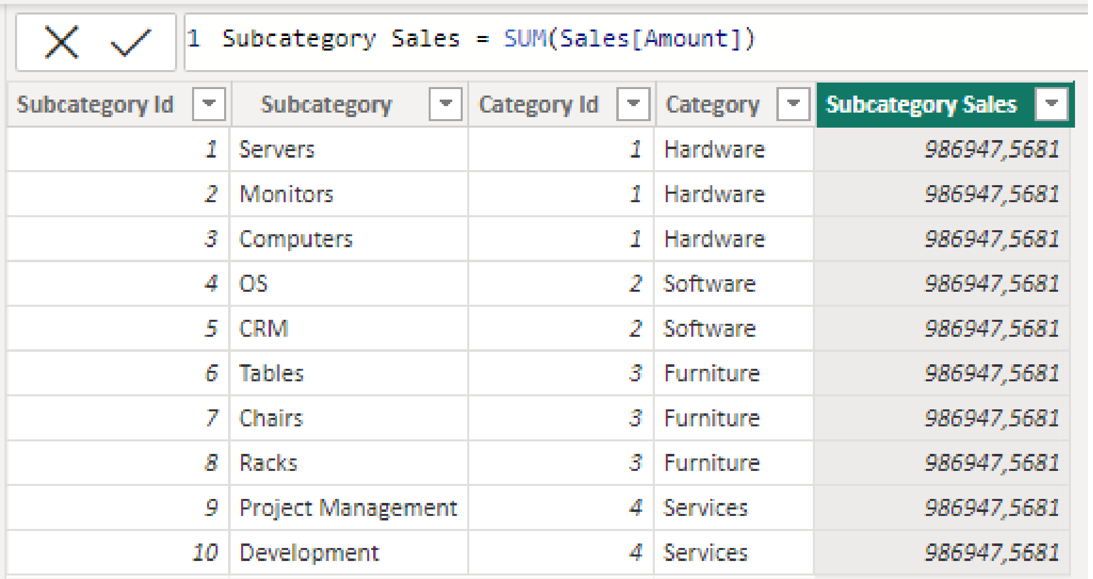
```
Year = YEAR('Calendar'[Date])
```

Date	Year
jueves, 1 de enero de 2015	2015
viernes, 2 de enero de 2015	2015
sábado, 3 de enero de 2015	2015
domingo, 4 de enero de 2015	2015
lunes, 5 de enero de 2015	2015
martes, 6 de enero de 2015	2015
miércoles, 7 de enero de 2015	2015
jueves, 8 de enero de 2015	2015
viernes, 9 de enero de 2015	2015



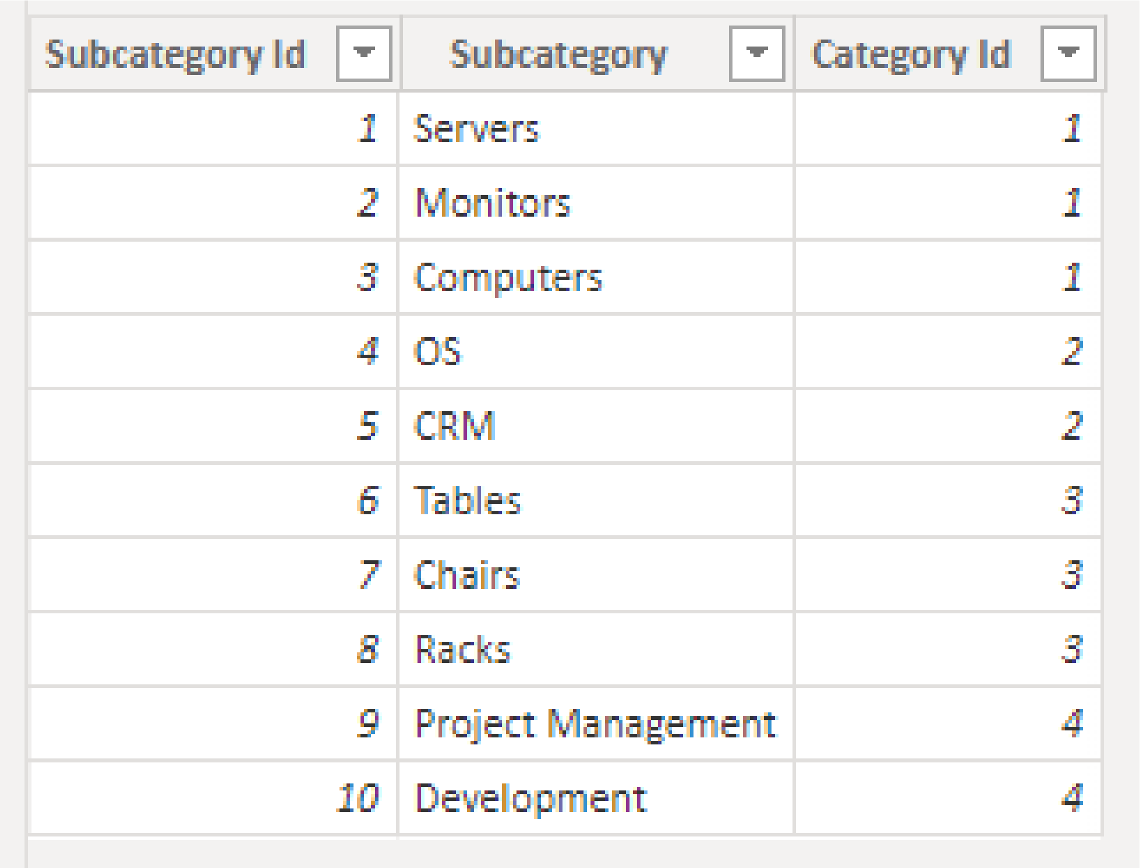
```
Subcategory Sales =
CALCULATE(
    SUM(Sales[Amount])
)
```

Subcategory Id	Subcategory	Category Id	Category	Subcategory Sales
1	Servers	1	Hardware	149285,1205
2	Monitors	1	Hardware	159865,4967
3	Computers	1	Hardware	42800,4569
4	OS	2	Software	69726,6933
5	CRM	2	Software	45067,8867
6	Tables	3	Furniture	99483,2931
7	Chairs	3	Furniture	87082,0182
8	Racks	3	Furniture	113131,0494
9	Project Management	4	Services	82452,6832
10	Development	4	Services	138052,8701



```
Subcategory Sales = SUM(Sales[Amount])
```

Subcategory Id	Subcategory	Category Id	Category	Subcategory Sales
1	Servers	1	Hardware	986947,5681
2	Monitors	1	Hardware	986947,5681
3	Computers	1	Hardware	986947,5681
4	OS	2	Software	986947,5681
5	CRM	2	Software	986947,5681
6	Tables	3	Furniture	986947,5681
7	Chairs	3	Furniture	986947,5681
8	Racks	3	Furniture	986947,5681
9	Project Management	4	Services	986947,5681
10	Development	4	Services	986947,5681



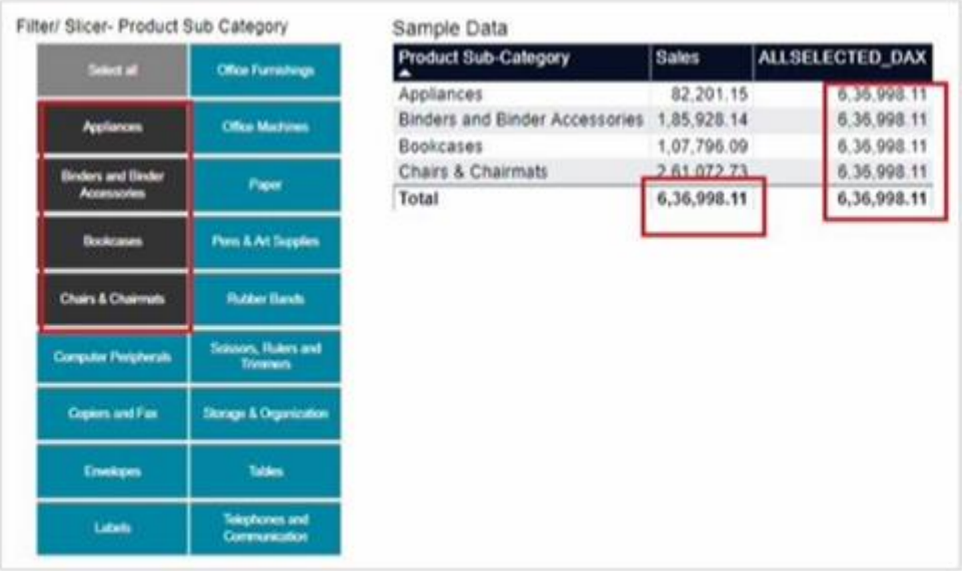
Subcategory Id	Subcategory	Category Id
1	Servers	1
2	Monitors	1
3	Computers	1
4	OS	2
5	CRM	2
6	Tables	3
7	Chairs	3
8	Racks	3
9	Project Management	4
10	Development	4

LABORATORIO 3

EJERCICIO 1

DAX ALLSELECTED

Devuelve todas las filas de una tabla o todos los valores de una columna, ignorando cualquier filtro que se haya aplicado dentro de la consulta, pero manteniendo los filtros que provienen del exterior.



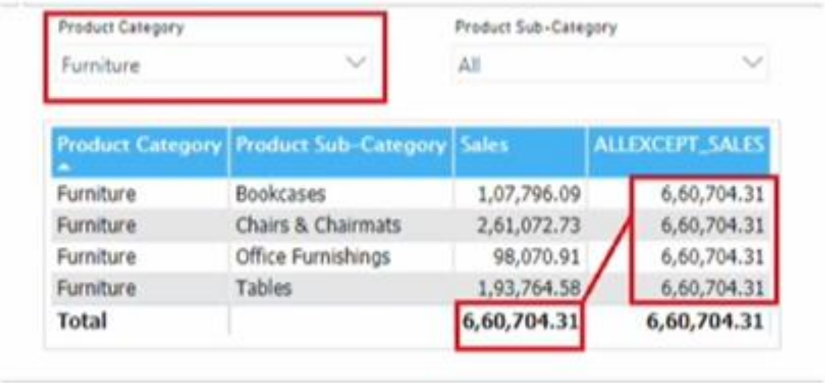
Filter/ Slicer- Product Sub Category

Sample Data

Product Sub-Category	Sales	ALLSELECTED_DAX
Appliances	82,201.15	6,36,998.11
Binders and Binder Accessories	1,85,928.14	6,36,998.11
Bookcases	1,07,796.09	6,36,998.11
Chairs & Chairmats	2,61,072.73	6,36,998.11
Total	**6,36,998.11**	**6,36,998.11**

DAX ALLEXCEPT

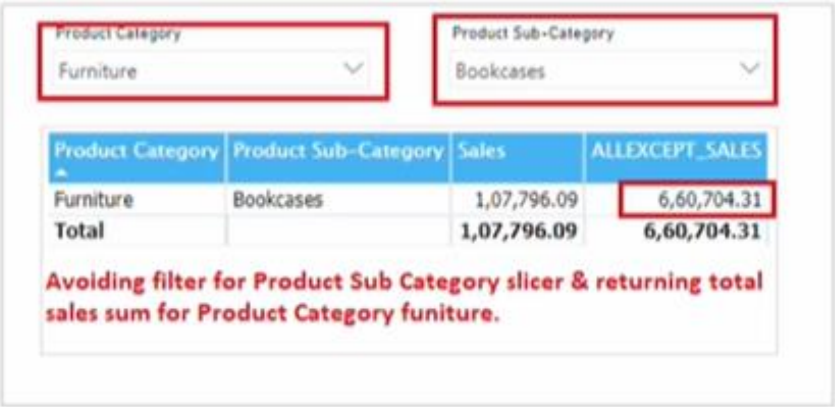
Devuelve todas las filas de una tabla excepto aquellas filas que se ven afectadas por los filtros de columna especificados.



Product Category: Furniture

Product Sub-Category: All

Product Category	Product Sub-Category	Sales	ALLEXCEPT_SALES
Furniture	Bookcases	1,07,796.09	6,60,704.31
Furniture	Chairs & Chairmats	2,61,072.73	6,60,704.31
Furniture	Office Furnishings	98,070.91	6,60,704.31
Furniture	Tables	1,93,764.58	6,60,704.31
Total		6,60,704.31	6,60,704.31

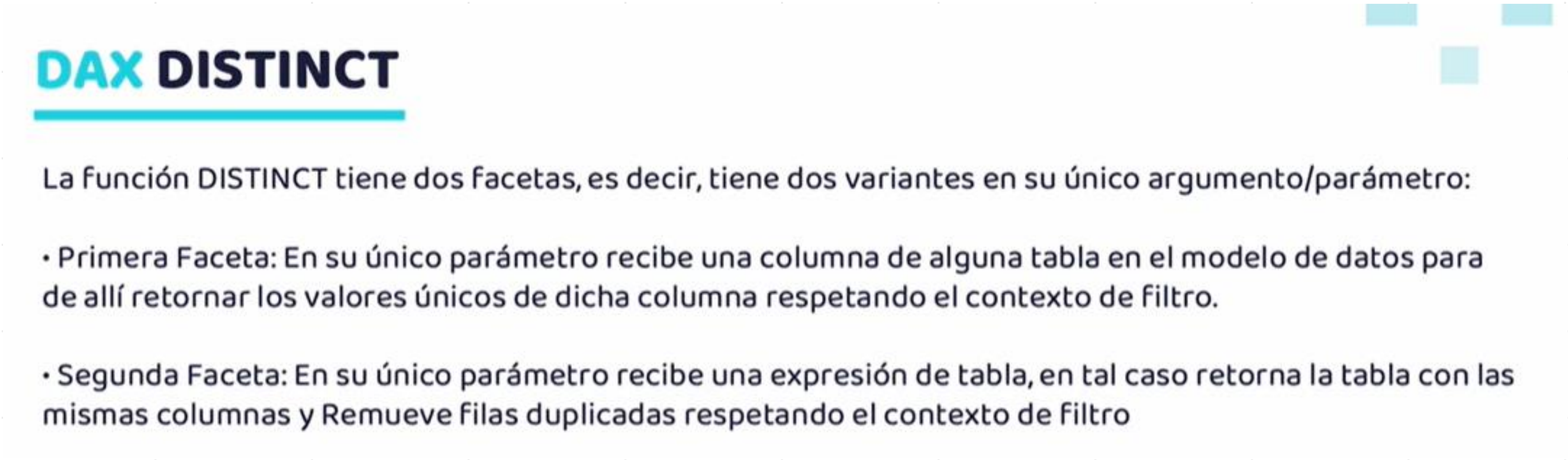


Product Category: Furniture

Product Sub-Category: Bookcases

Product Category	Product Sub-Category	Sales	ALLEXCEPT_SALES
Furniture	Bookcases	1,07,796.09	6,60,704.31
Total		1,07,796.09	6,60,704.31

Avoiding filter for Product Sub Category slicer & returning total sales sum for Product Category funiture.

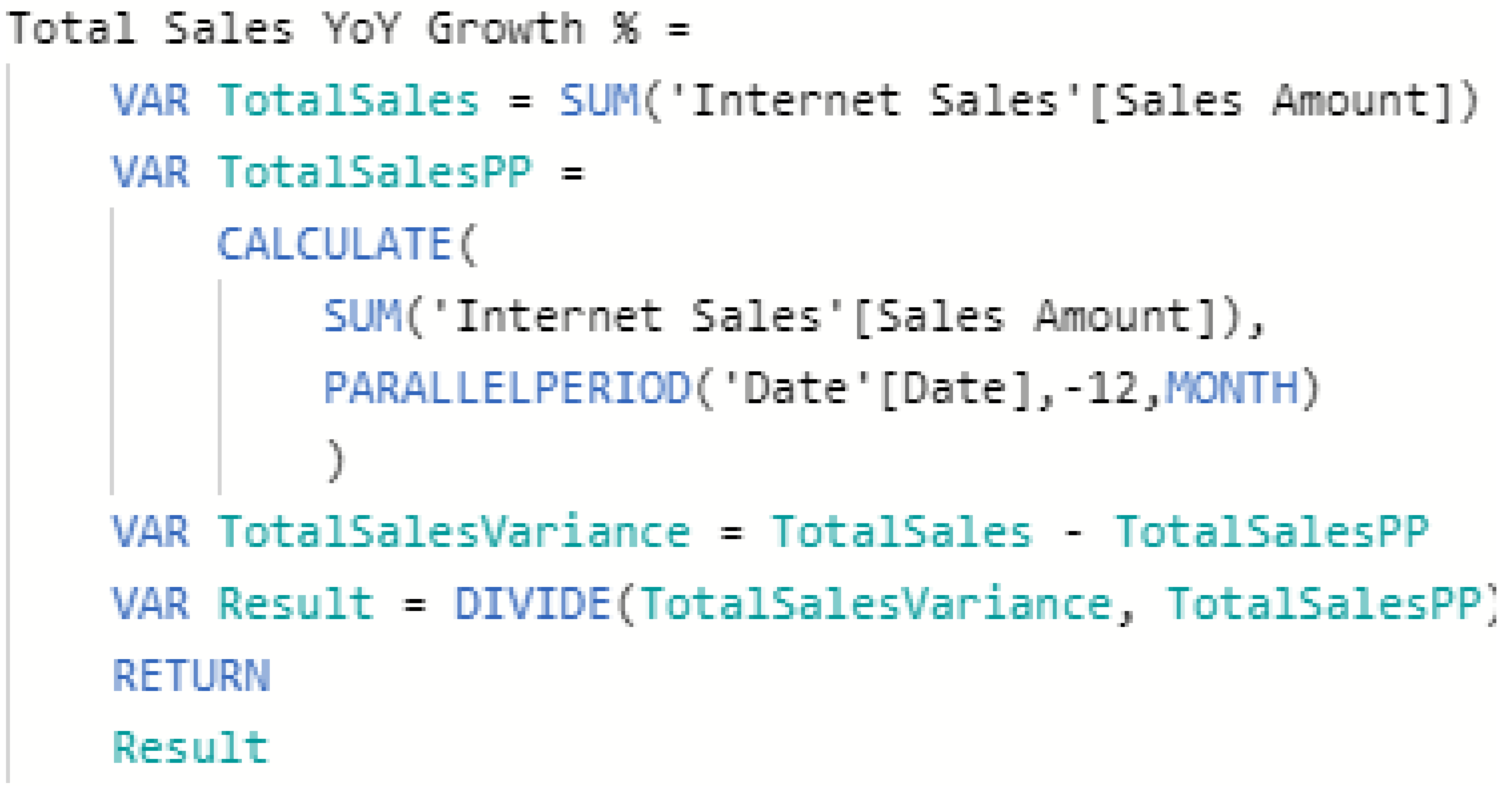


DAX DISTINCT

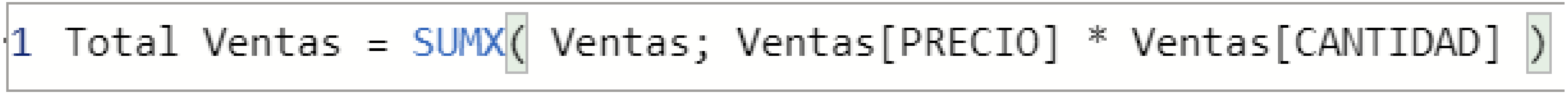
La función DISTINCT tiene dos facetas, es decir, tiene dos variantes en su único argumento/parámetro:

- Primera Faceta: En su único parámetro recibe una columna de alguna tabla en el modelo de datos para de allí retornar los valores únicos de dicha columna respetando el contexto de filtro.

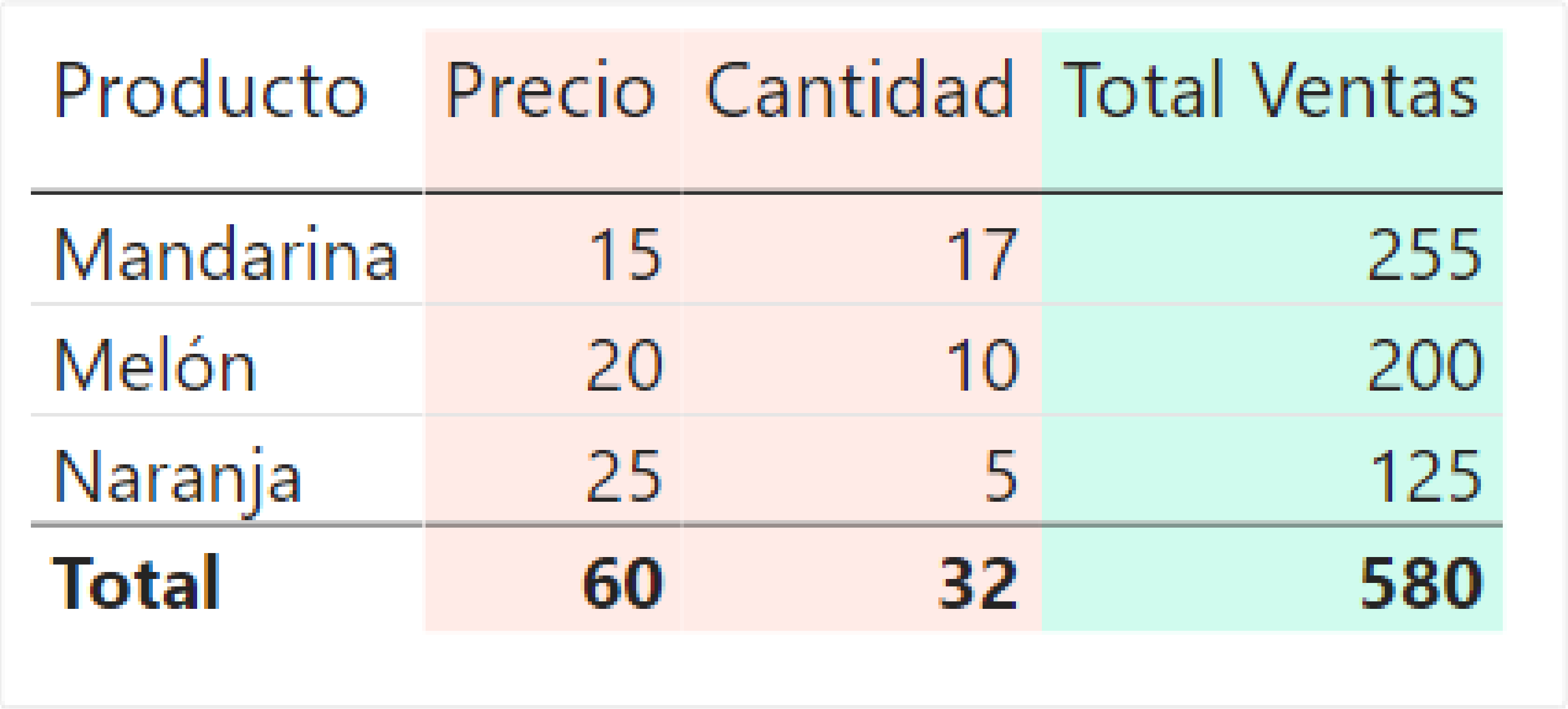
- Segunda Faceta: En su único parámetro recibe una expresión de tabla, en tal caso retorna la tabla con las mismas columnas y Remueve filas duplicadas respetando el contexto de filtro



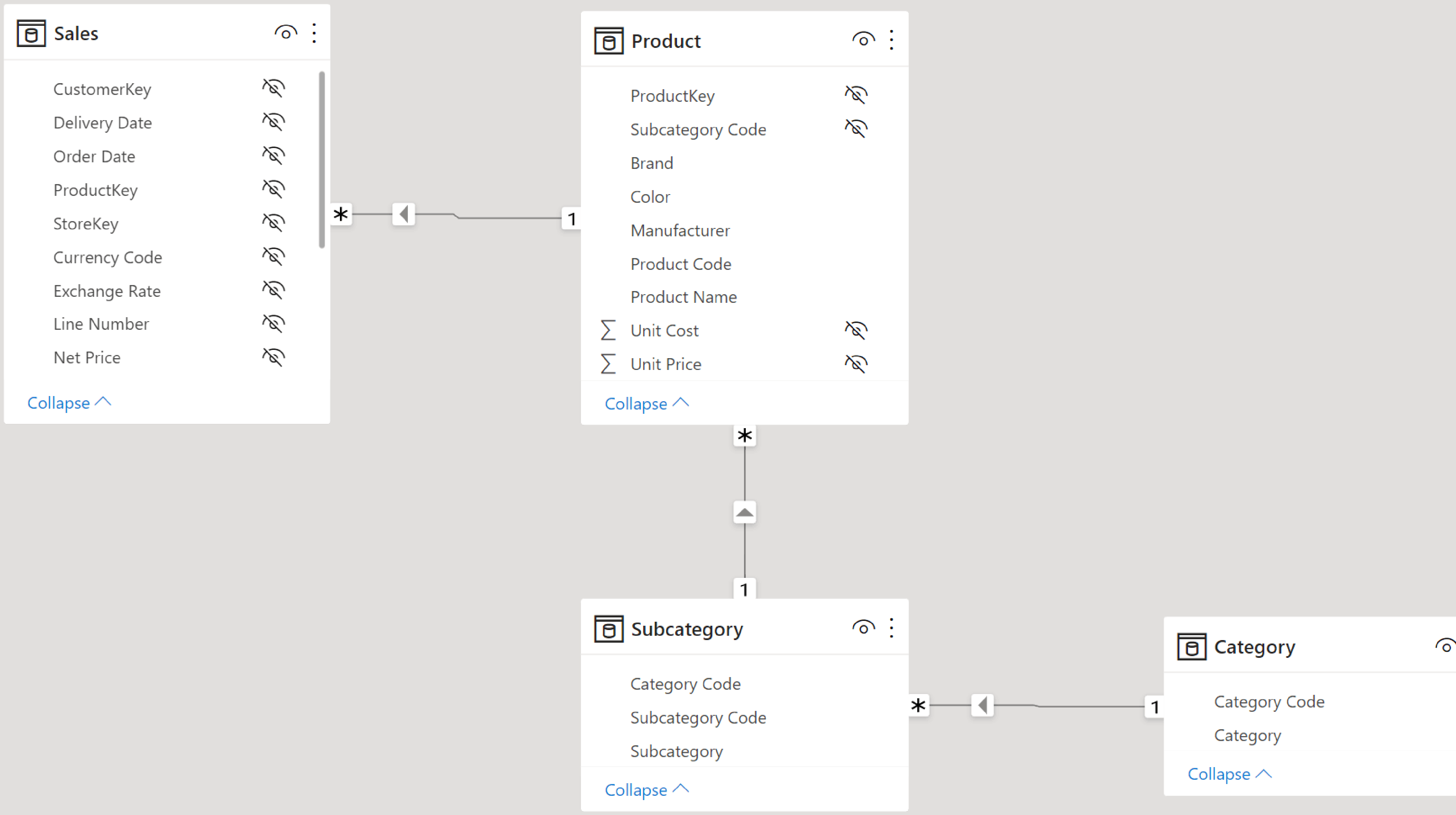
```
Total Sales YoY Growth % =
    VAR TotalSales = SUM('Internet Sales'[Sales Amount])
    VAR TotalSalesPP =
        CALCULATE(
            SUM('Internet Sales'[Sales Amount]),
            PARALLELPERIOD('Date'[Date],-12,MONTH)
            )
    VAR TotalSalesVariance = TotalSales - TotalSalesPP
    VAR Result = DIVIDE(TotalSalesVariance, TotalSalesPP)
    RETURN
    Result
```



```
Total Ventas = SUMX( Ventas; Ventas[PRECIO] * Ventas[CANTIDAD] )
```

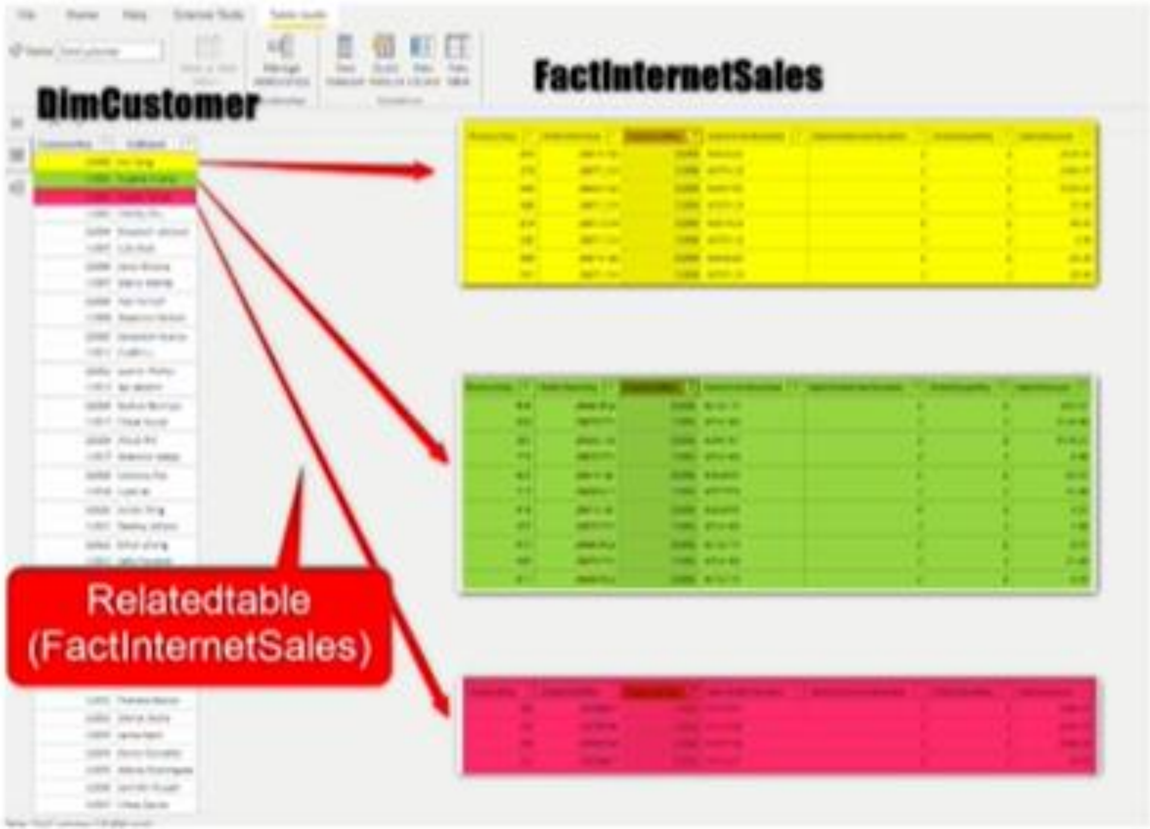


Producto	Precio	Cantidad	Total Ventas
Mandarina	15	17	255
Melón	20	10	200
Naranja	25	5	125
Total	**60**	**32**	**580**


Sales
CustomerKey
Delivery Date
Order Date
ProductKey
StoreKey
Currency Code
Exchange Rate
Line Number
Net Price
Collapse
*
1
Product
ProductKey
Subcategory Code
Brand
Color
Manufacturer
Product Code
Product Name
Unit Cost
Unit Price
Collapse
*
1
Subcategory
Category Code
Subcategory Code
Subcategory
Collapse
*
1
Category
Category Code
Category
Collapse

DAX RELATED TABLES

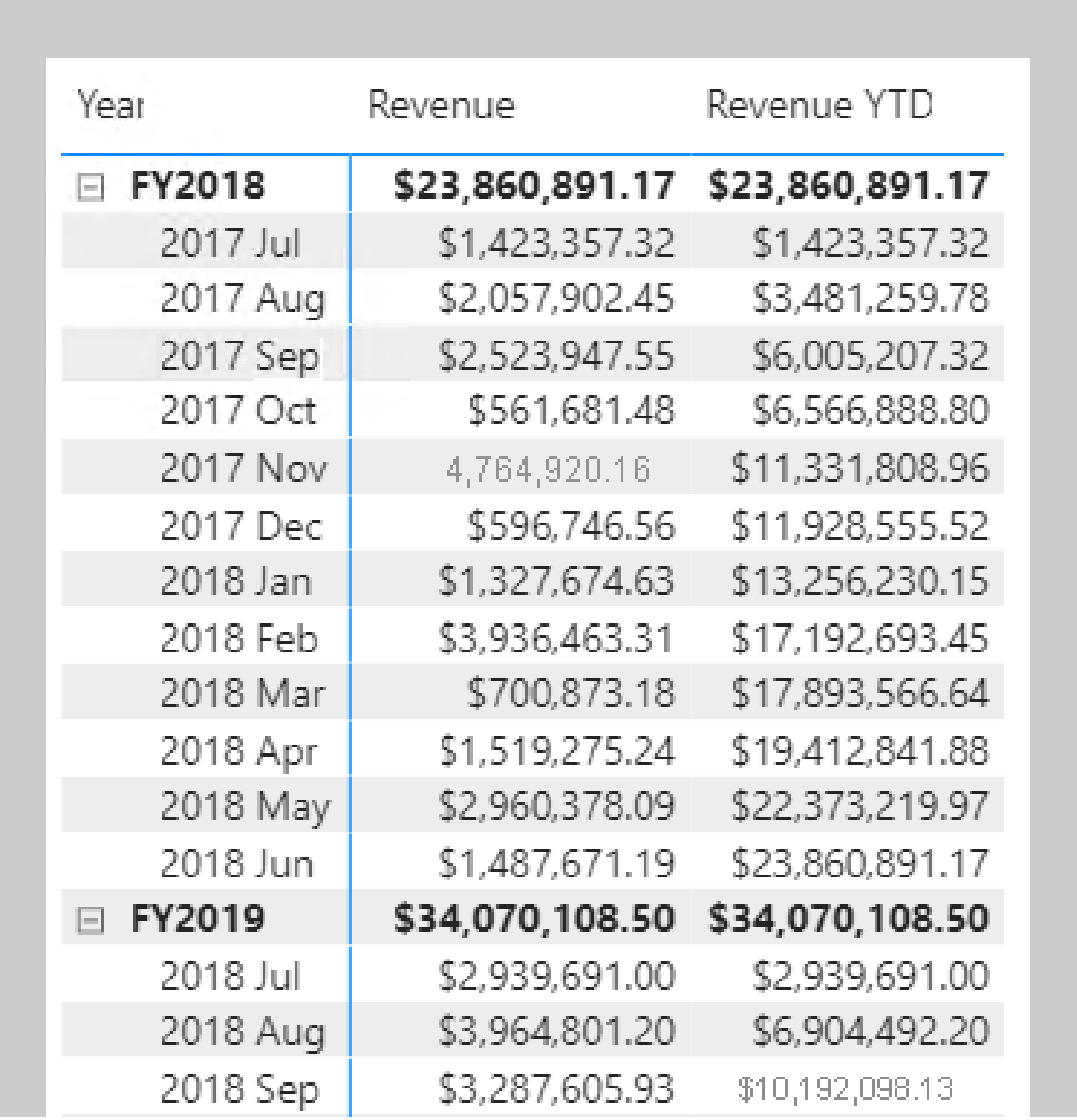
Devuelve las tablas relacionadas filtradas para que solo incluya las filas relacionadas.



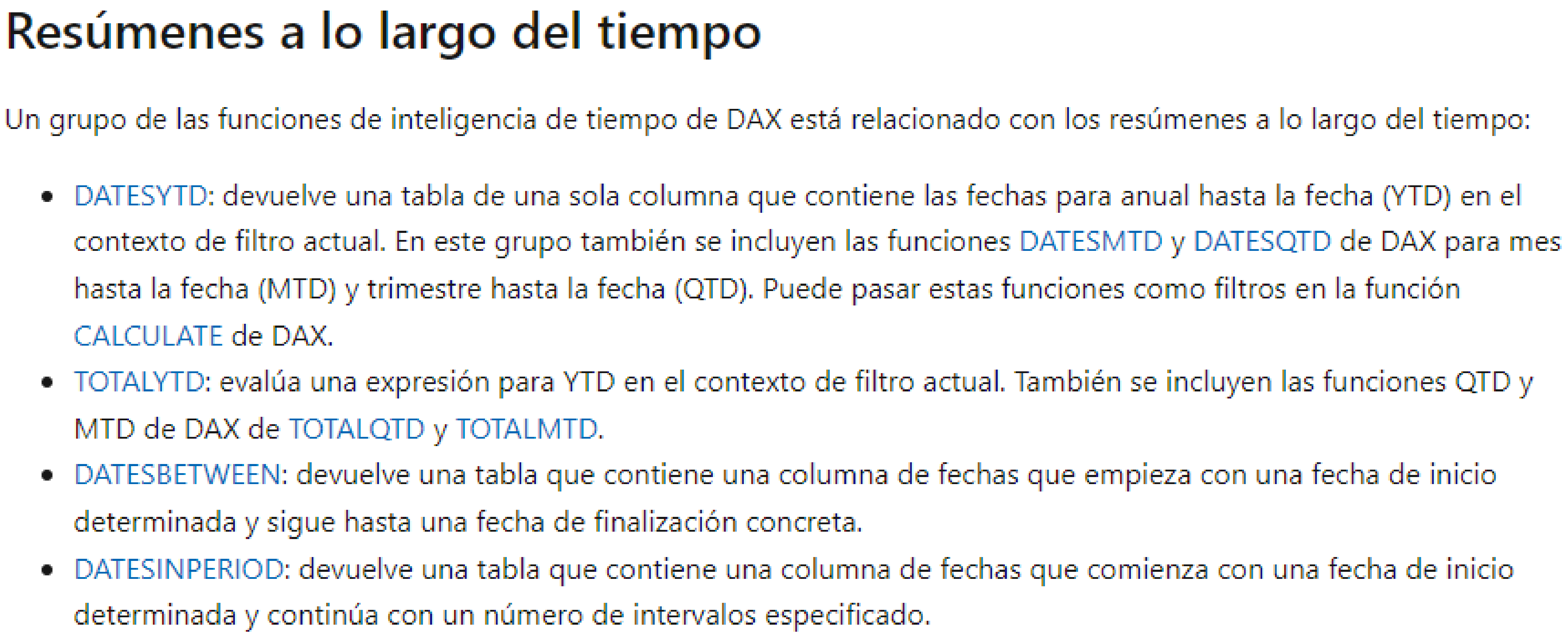
LABORATORIO 4

EJERCICIO 2

TIME INTELLIGENCE



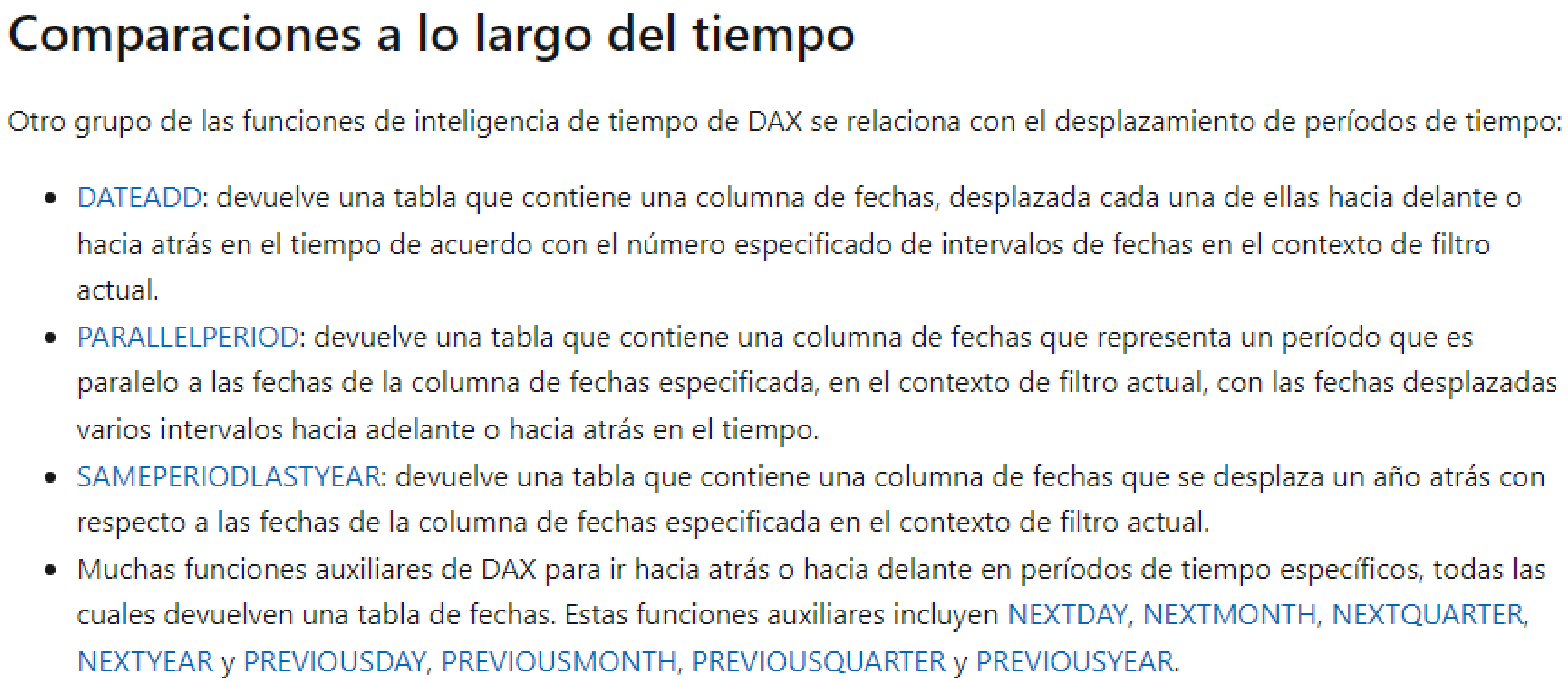
Year	Revenue	Revenue YTD
⊟ **FY2018**	**$23,860,891.17**	**$23,860,891.17**
2017 Jul	$1,423,357.32	$1,423,357.32
2017 Aug	$2,057,902.45	$3,481,259.78
2017 Sep	$2,523,947.55	$6,005,207.32
2017 Oct	$561,681.48	$6,566,888.80
2017 Nov	4,764,920.16	$11,331,808.96
2017 Dec	$596,746.56	$11,928,555.52
2018 Jan	$1,327,674.63	$13,256,230.15
2018 Feb	$3,936,463.31	$17,192,693.45
2018 Mar	$700,873.18	$17,893,566.64
2018 Apr	$1,519,275.24	$19,412,841.88
2018 May	$2,960,378.09	$22,373,219.97
2018 Jun	$1,487,671.19	$23,860,891.17
⊟ **FY2019**	**$34,070,108.50**	**$34,070,108.50**
2018 Jul	$2,939,691.00	$2,939,691.00
2018 Aug	$3,964,801.20	$6,904,492.20
2018 Sep	$3,287,605.93	$10,192,098.13



Resúmenes a lo largo del tiempo

Un grupo de las funciones de inteligencia de tiempo de DAX está relacionado con los resúmenes a lo largo del tiempo:

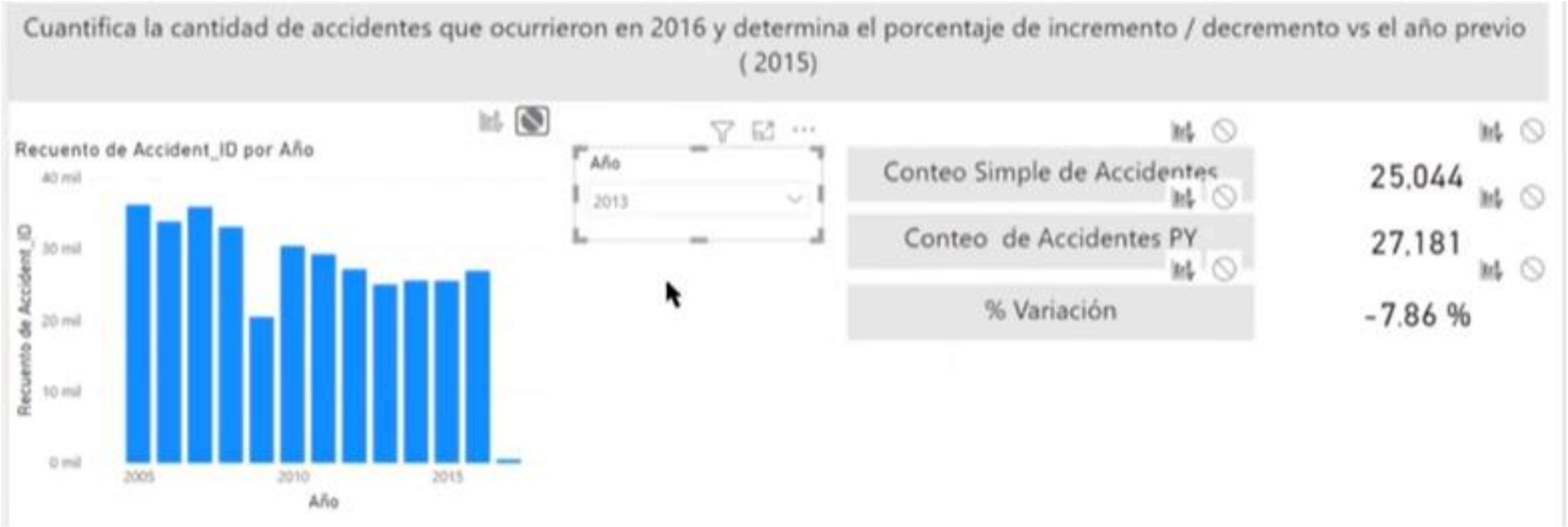
- DATESYTD: devuelve una tabla de una sola columna que contiene las fechas para anual hasta la fecha (YTD) en el contexto de filtro actual. En este grupo también se incluyen las funciones DATESMTD y DATESQTD de DAX para mes hasta la fecha (MTD) y trimestre hasta la fecha (QTD). Puede pasar estas funciones como filtros en la función CALCULATE de DAX.
- TOTALYTD: evalúa una expresión para YTD en el contexto de filtro actual. También se incluyen las funciones QTD y MTD de DAX de TOTALQTD y TOTALMTD.
- DATESBETWEEN: devuelve una tabla que contiene una columna de fechas que empieza con una fecha de inicio determinada y sigue hasta una fecha de finalización concreta.
- DATESINPERIOD: devuelve una tabla que contiene una columna de fechas que comienza con una fecha de inicio determinada y continúa con un número de intervalos especificado.



Comparaciones a lo largo del tiempo

Otro grupo de las funciones de inteligencia de tiempo de DAX se relaciona con el desplazamiento de períodos de tiempo:

- DATEADD: devuelve una tabla que contiene una columna de fechas, desplazada cada una de ellas hacia delante o hacia atrás en el tiempo de acuerdo con el número especificado de intervalos de fechas en el contexto de filtro actual.
- PARALLELPERIOD: devuelve una tabla que contiene una columna de fechas que representa un período que es paralelo a las fechas de la columna de fechas especificada, en el contexto de filtro actual, con las fechas desplazadas varios intervalos hacia adelante o hacia atrás en el tiempo.
- SAMEPERIODLASTYEAR: devuelve una tabla que contiene una columna de fechas que se desplaza un año atrás con respecto a las fechas de la columna de fechas especificada en el contexto de filtro actual.
- Muchas funciones auxiliares de DAX para ir hacia atrás o hacia delante en períodos de tiempo específicos, todas las cuales devuelven una tabla de fechas. Estas funciones auxiliares incluyen NEXTDAY, NEXTMONTH, NEXTQUARTER, NEXTYEAR y PREVIOUSDAY, PREVIOUSMONTH, PREVIOUSQUARTER y PREVIOUSYEAR.

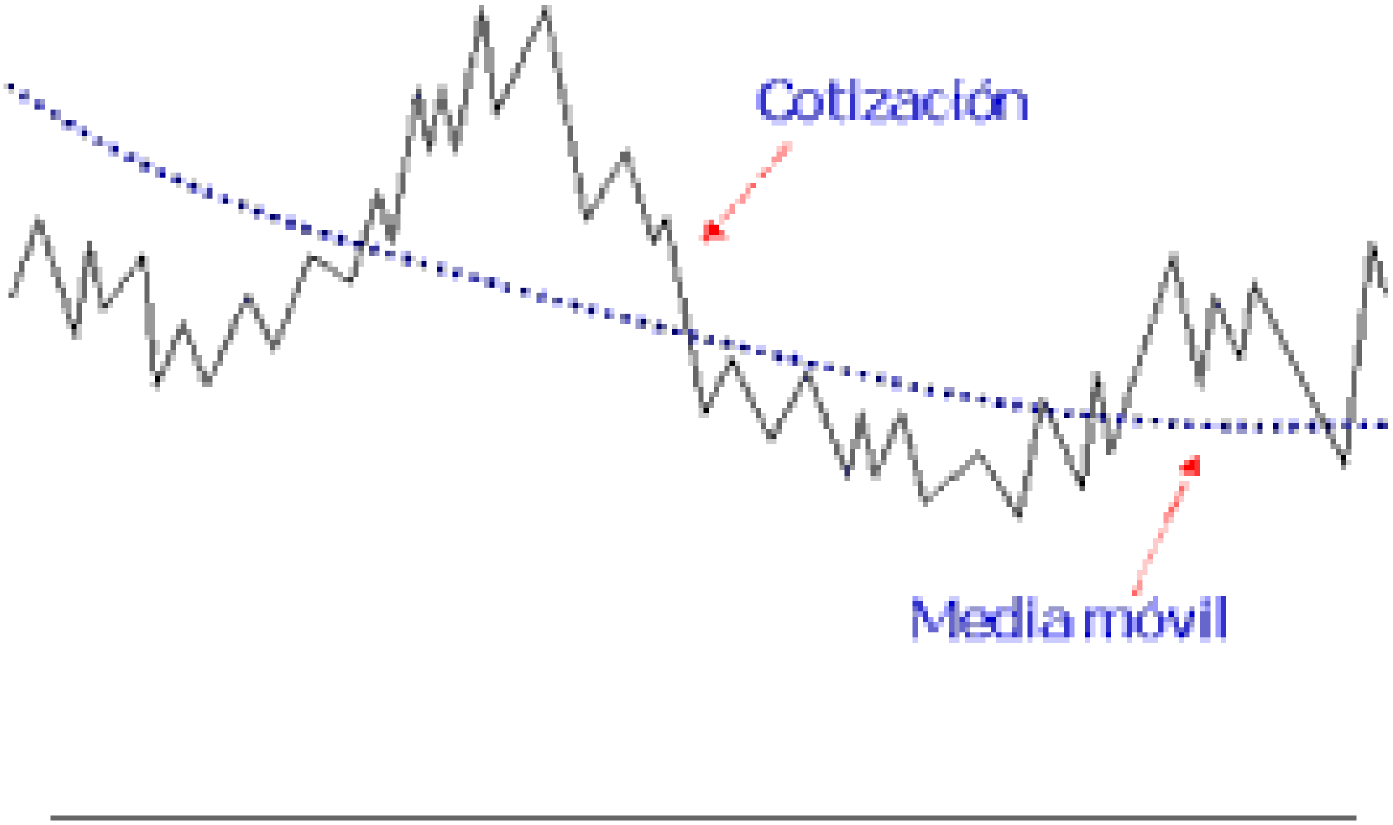


Cuantifica la cantidad de accidentes que ocurrieron en 2016 y determina el porcentaje de incremento / decremento vs el año previo (2015)

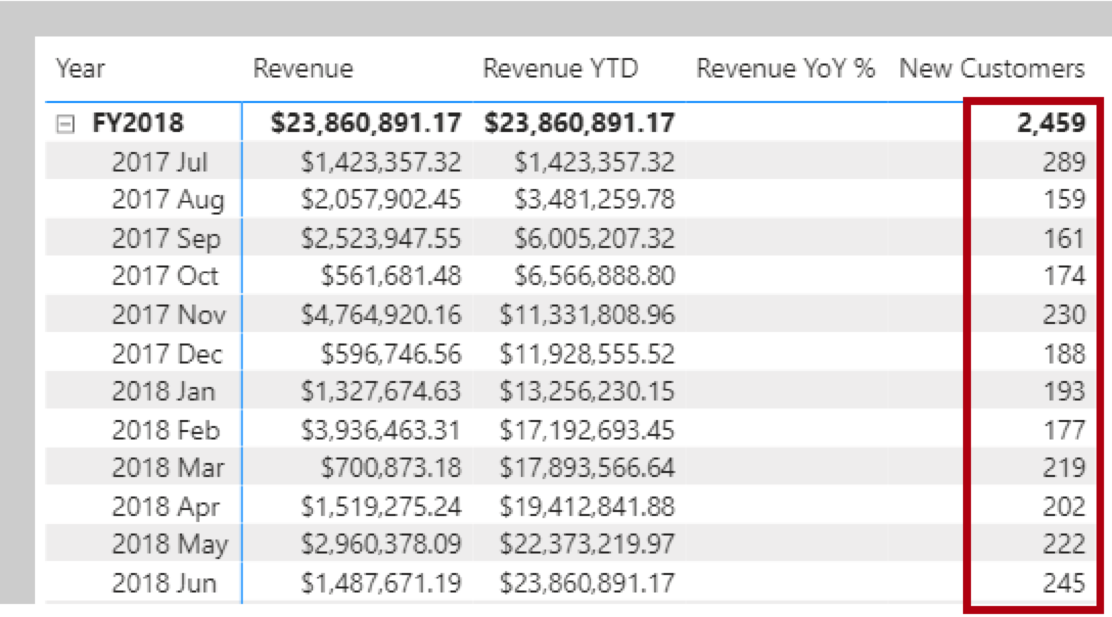
Año: 2013

Conteo Simple de Accidentes	25,044
Conteo de Accidentes PY	27,181
% Variación	-7.86 %

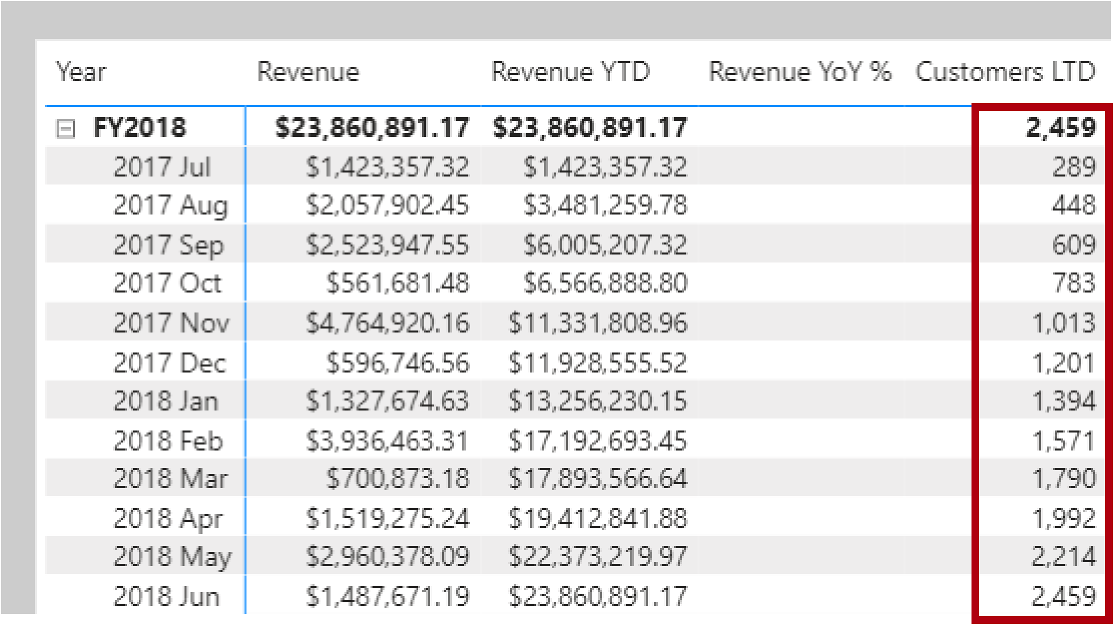
MEDIA MOVIL



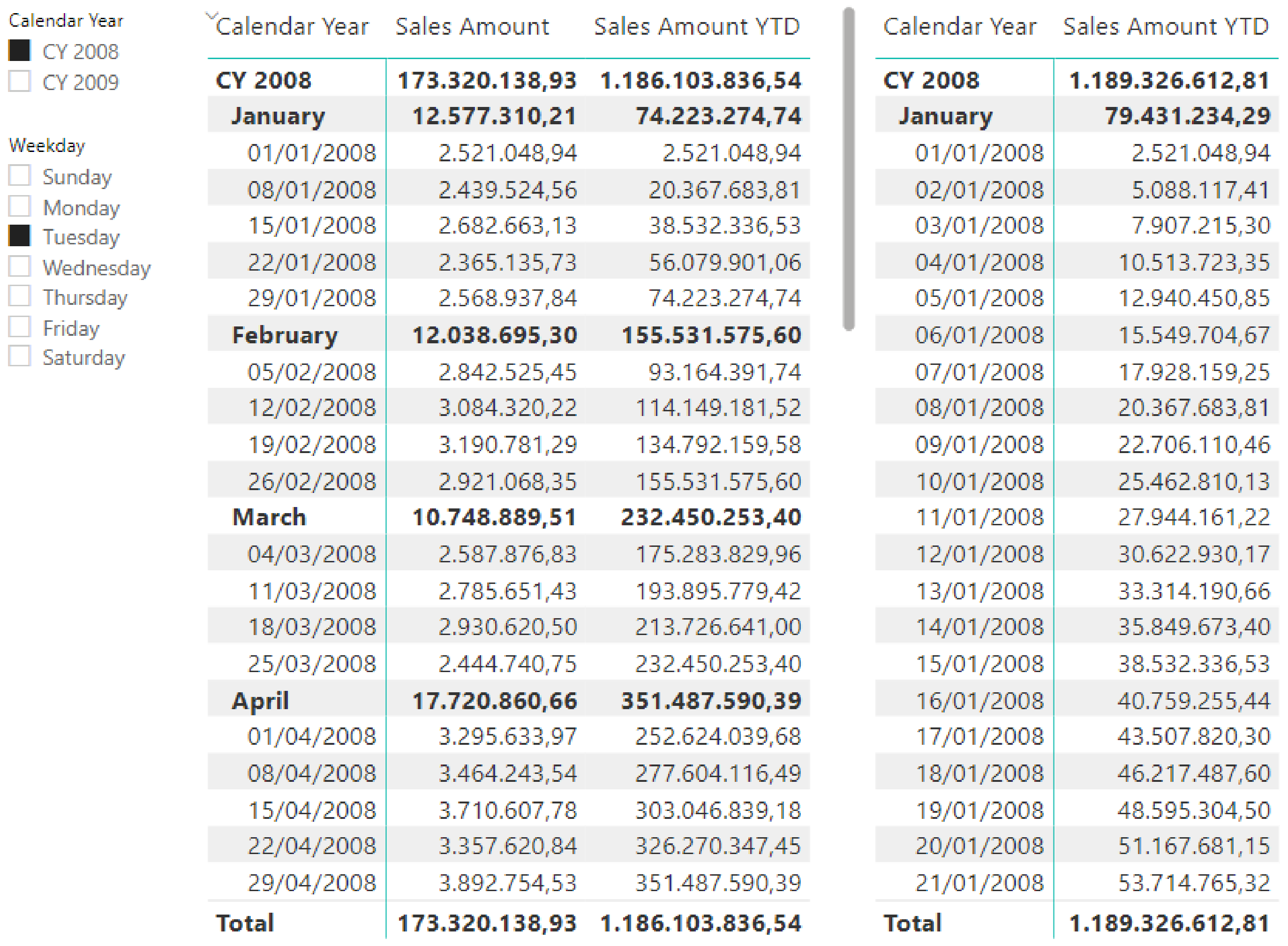
LABORATORIO 5



Year	Revenue	Revenue YTD	Revenue YoY %	New Customers
⊟ **FY2018**	**$23,860,891.17**	**$23,860,891.17**		**2,459**
2017 Jul	$1,423,357.32	$1,423,357.32		289
2017 Aug	$2,057,902.45	$3,481,259.78		159
2017 Sep	$2,523,947.55	$6,005,207.32		161
2017 Oct	$561,681.48	$6,566,888.80		174
2017 Nov	$4,764,920.16	$11,331,808.96		230
2017 Dec	$596,746.56	$11,928,555.52		188
2018 Jan	$1,327,674.63	$13,256,230.15		193
2018 Feb	$3,936,463.31	$17,192,693.45		177
2018 Mar	$700,873.18	$17,893,566.64		219
2018 Apr	$1,519,275.24	$19,412,841.88		202
2018 May	$2,960,378.09	$22,373,219.97		222
2018 Jun	$1,487,671.19	$23,860,891.17		245



Year	Revenue	Revenue YTD	Revenue YoY %	Customers LTD
⊟ **FY2018**	**$23,860,891.17**	**$23,860,891.17**		**2,459**
2017 Jul	$1,423,357.32	$1,423,357.32		289
2017 Aug	$2,057,902.45	$3,481,259.78		448
2017 Sep	$2,523,947.55	$6,005,207.32		609
2017 Oct	$561,681.48	$6,566,888.80		783
2017 Nov	$4,764,920.16	$11,331,808.96		1,013
2017 Dec	$596,746.56	$11,928,555.52		1,201
2018 Jan	$1,327,674.63	$13,256,230.15		1,394
2018 Feb	$3,936,463.31	$17,192,693.45		1,571
2018 Mar	$700,873.18	$17,893,566.64		1,790
2018 Apr	$1,519,275.24	$19,412,841.88		1,992
2018 May	$2,960,378.09	$22,373,219.97		2,214
2018 Jun	$1,487,671.19	$23,860,891.17		2,459



Calendar Year

■ CY 2008

□ CY 2009

Weekday

□ Sunday

□ Monday

■ Tuesday

□ Wednesday

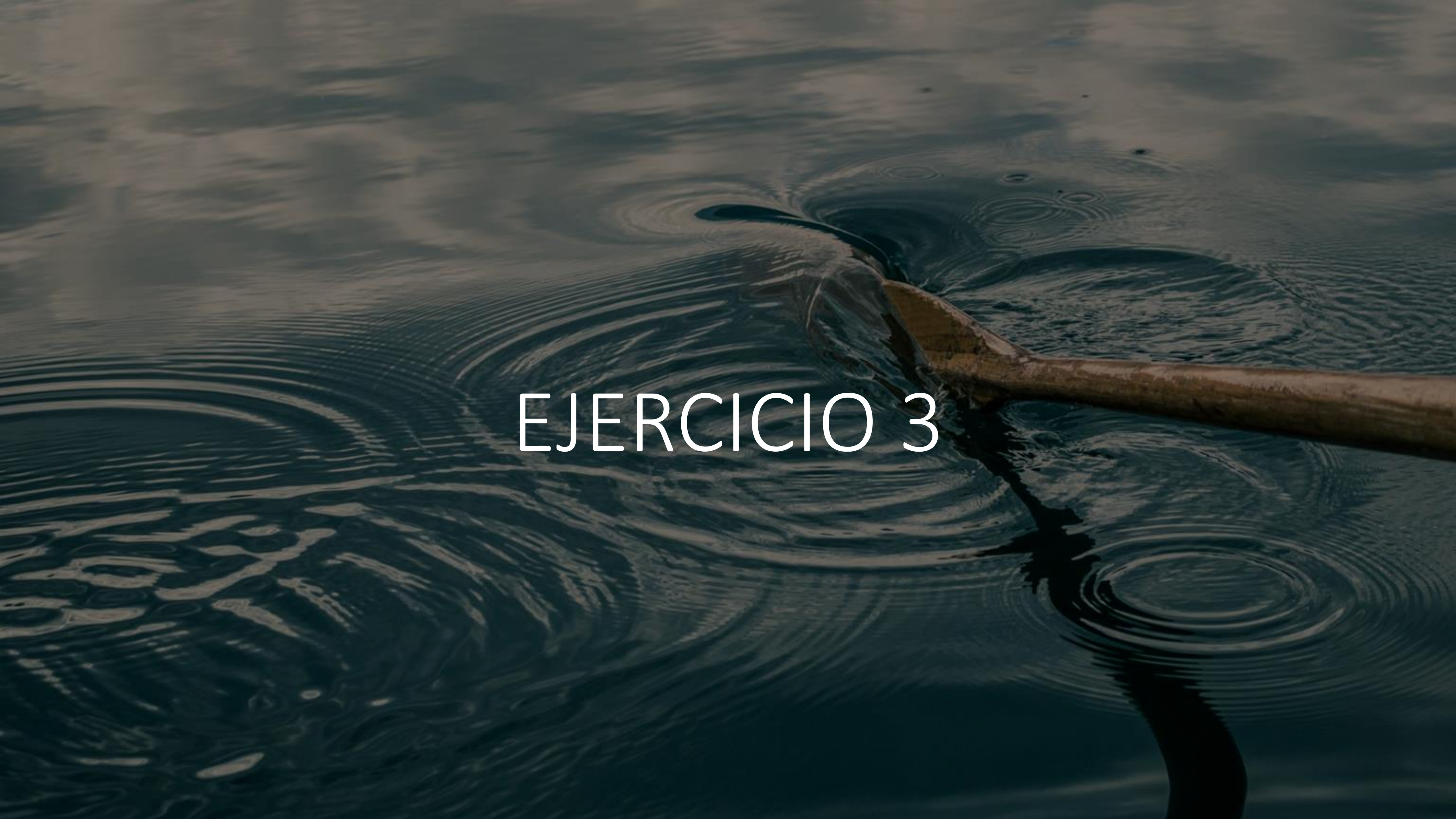
□ Thursday

□ Friday

□ Saturday

Calendar Year	Sales Amount	Sales Amount YTD
CY 2008	**173.320.138,93**	**1.186.103.836,54**
January	**12.577.310,21**	**74.223.274,74**
01/01/2008	2.521.048,94	2.521.048,94
08/01/2008	2.439.524,56	20.367.683,81
15/01/2008	2.682.663,13	38.532.336,53
22/01/2008	2.365.135,73	56.079.901,06
29/01/2008	2.568.937,84	74.223.274,74
February	**12.038.695,30**	**155.531.575,60**
05/02/2008	2.842.525,45	93.164.391,74
12/02/2008	3.084.320,22	114.149.181,52
19/02/2008	3.190.781,29	134.792.159,58
26/02/2008	2.921.068,35	155.531.575,60
March	**10.748.889,51**	**232.450.253,40**
04/03/2008	2.587.876,83	175.283.829,96
11/03/2008	2.785.651,43	193.895.779,42
18/03/2008	2.930.620,50	213.726.641,00
25/03/2008	2.444.740,75	232.450.253,40
April	**17.720.860,66**	**351.487.590,39**
01/04/2008	3.295.633,97	252.624.039,68
08/04/2008	3.464.243,54	277.604.116,49
15/04/2008	3.710.607,78	303.046.839,18
22/04/2008	3.357.620,84	326.270.347,45
29/04/2008	3.892.754,53	351.487.590,39
Total	**173.320.138,93**	**1.186.103.836,54**

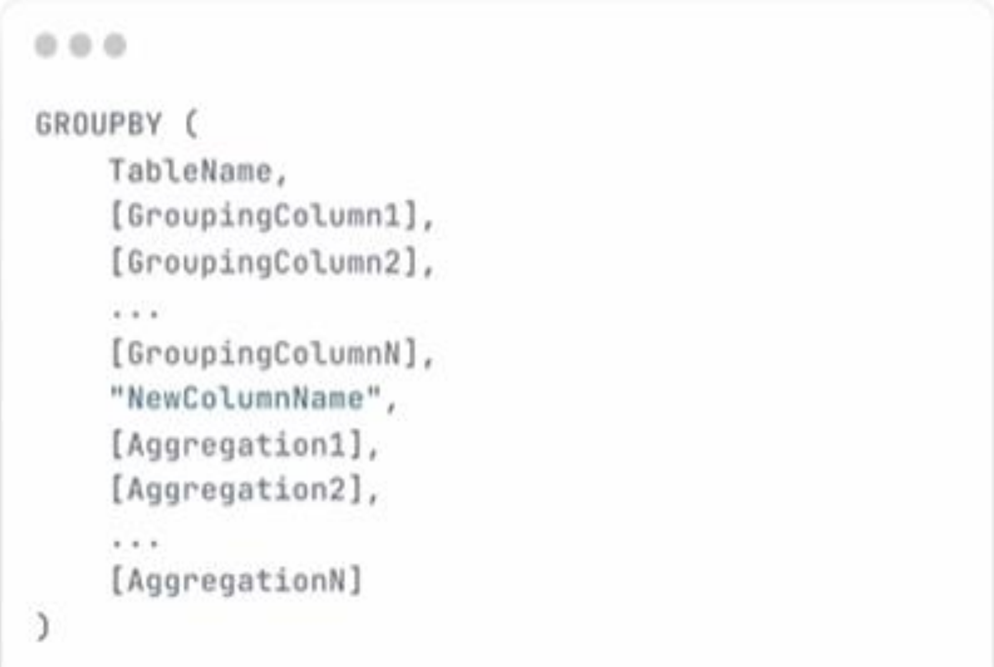
Calendar Year	Sales Amount YTD
CY 2008	**1.189.326.612,81**
January	**79.431.234,29**
01/01/2008	2.521.048,94
02/01/2008	5.088.117,41
03/01/2008	7.907.215,30
04/01/2008	10.513.723,35
05/01/2008	12.940.450,85
06/01/2008	15.549.704,67
07/01/2008	17.928.159,25
08/01/2008	20.367.683,81
09/01/2008	22.706.110,46
10/01/2008	25.462.810,13
11/01/2008	27.944.161,22
12/01/2008	30.622.930,17
13/01/2008	33.314.190,66
14/01/2008	35.849.673,40
15/01/2008	38.532.336,53
16/01/2008	40.759.255,44
17/01/2008	43.507.820,30
18/01/2008	46.217.487,60
19/01/2008	48.595.304,50
20/01/2008	51.167.681,15
21/01/2008	53.714.765,32
Total	**1.189.326.612,81**



EJERCICIO 3

DAX GROUP BY

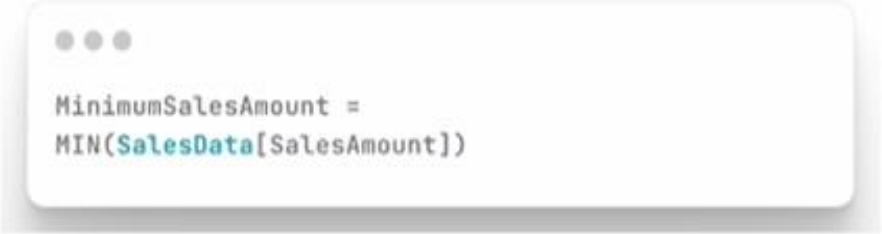
GROUPBY se usa para realizar varias agregaciones en un solo recorrido de tabla.



```
GROUPBY (
    TableName,
    [GroupingColumn1],
    [GroupingColumn2],
    ...
    [GroupingColumnN],
    "NewColumnName",
    [Aggregation1],
    [Aggregation2],
    ...
    [AggregationN]
)
```

DAX Mínimo

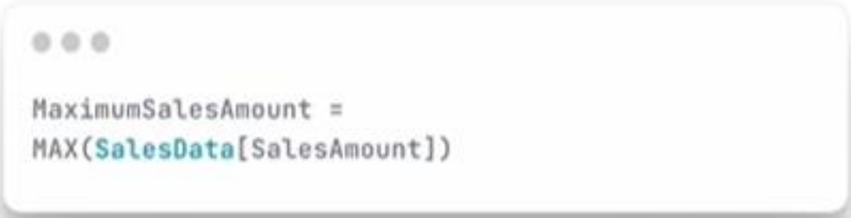
La función MINX retorna el valor más pequeño de todos los valores parciales que «salieron» como resultado de una expresión que se evalúa fila a fila en una tabla



```
MinimumSalesAmount =
MIN(SalesData[SalesAmount])
```

DAX Máximo

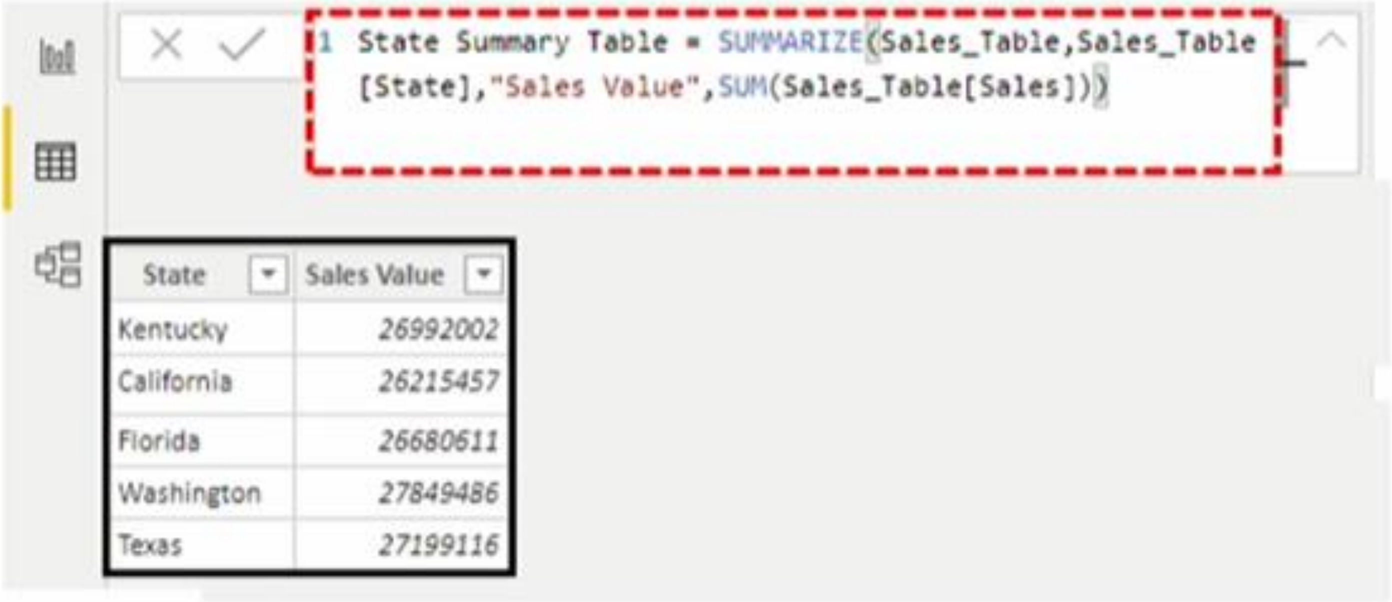
La función MAXX retorna el valor más grande de todos los valores parciales que «salieron» como resultado de una expresión que se evalúa fila a fila en una tabla



```
MaximumSalesAmount =
MAX(SalesData[SalesAmount])
```

DAX SUMMARIZE

Crea un resumen de la tabla de entrada agrupada por las columnas especificadas.



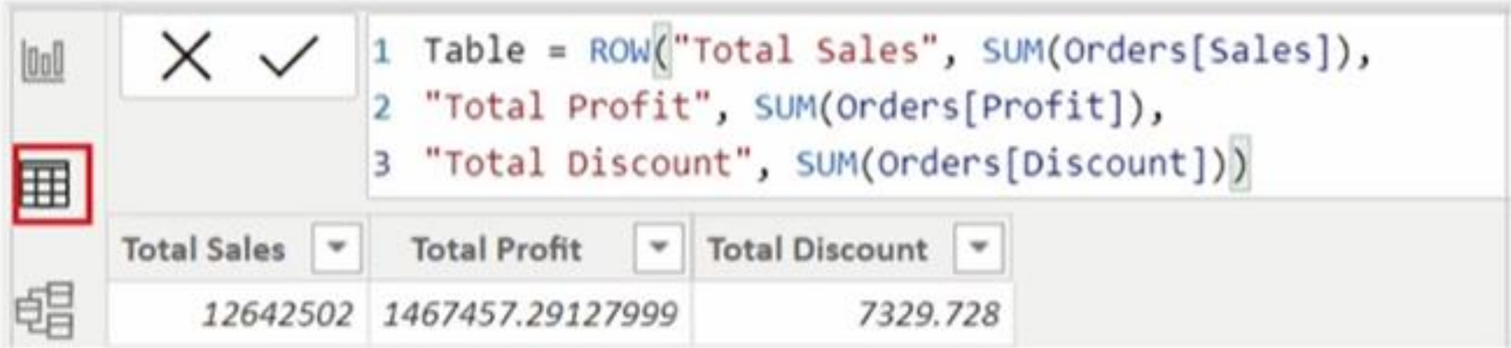
```
State Summary Table = SUMMARIZE(Sales_Table,Sales_Table
[State],"Sales Value",SUM(Sales_Table[Sales]))
```

State	Sales Value
Kentucky	26992002
California	26215457
Florida	26680611
Washington	27849486
Texas	27199116

EJERCICIO 4

DAX ROW

Devuelve una tabla de una sola fila con nuevas columnas especificadas por las expresiones DAX.

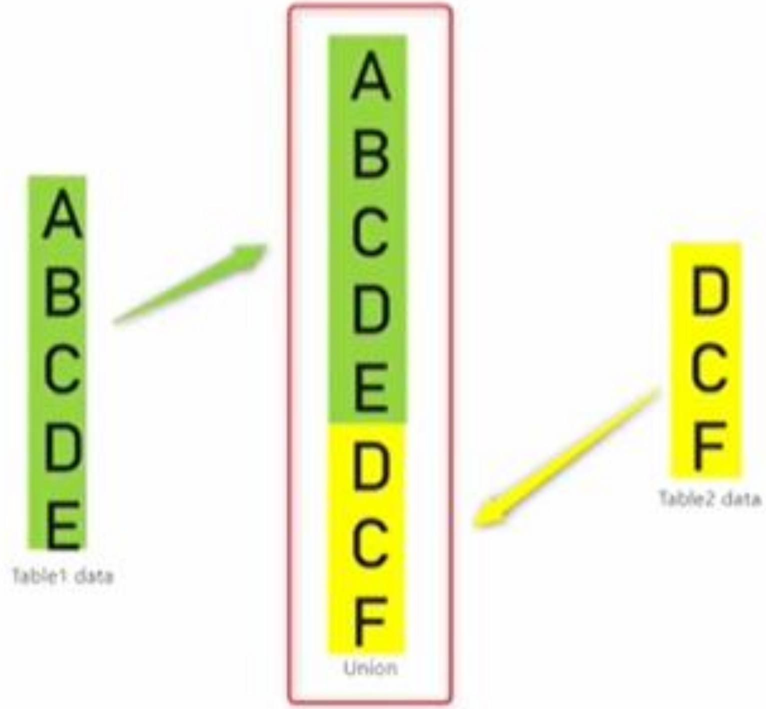


```
Table = ROW("Total Sales", SUM(Orders[Sales]),
"Total Profit", SUM(Orders[Profit]),
"Total Discount", SUM(Orders[Discount]))
```

Total Sales	Total Profit	Total Discount
12642502	1467457.29127999	7329.728

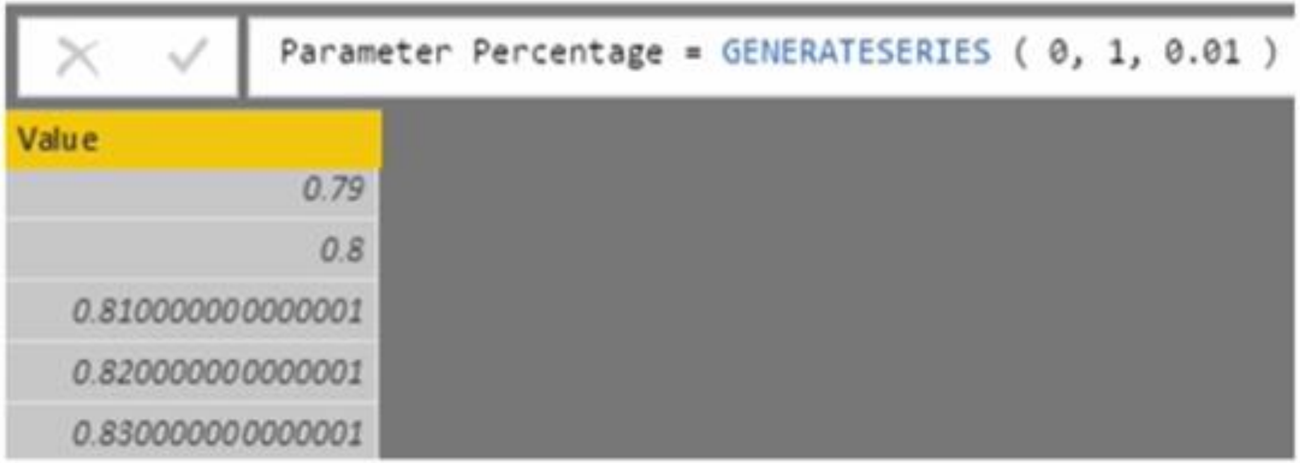
DAX UNION

Devuelve la unión de las tablas cuyas columnas coinciden.



DAX GENERATESERIES

Devuelve una tabla con una columna, rellena con valores secuenciales de principio a fin.

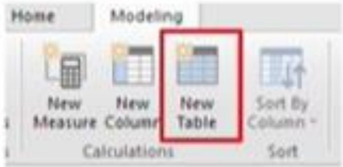


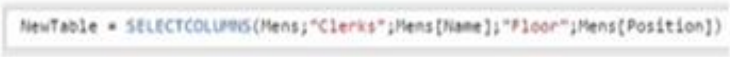
```
Parameter Percentage = GENERATESERIES ( 0, 1, 0.01 )
```

Value
0.79
0.8
0.810000000000001
0.820000000000001
0.830000000000001

DAX SELECTCOLUMNS

Devuelve una tabla con columnas seleccionadas de la tabla y nuevas columnas especificadas por las expresiones DAX.





```
NewTable = SELECTCOLUMNS(Mens;"Clerks";Mens[Name];"Floor";Mens[Position])
```

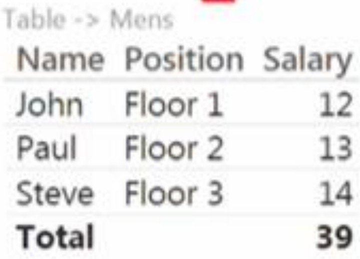


Table -> Mens

Name	Position	Salary
John	Floor 1	12
Paul	Floor 2	13
Steve	Floor 3	14
Total		**39**

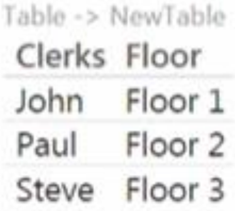
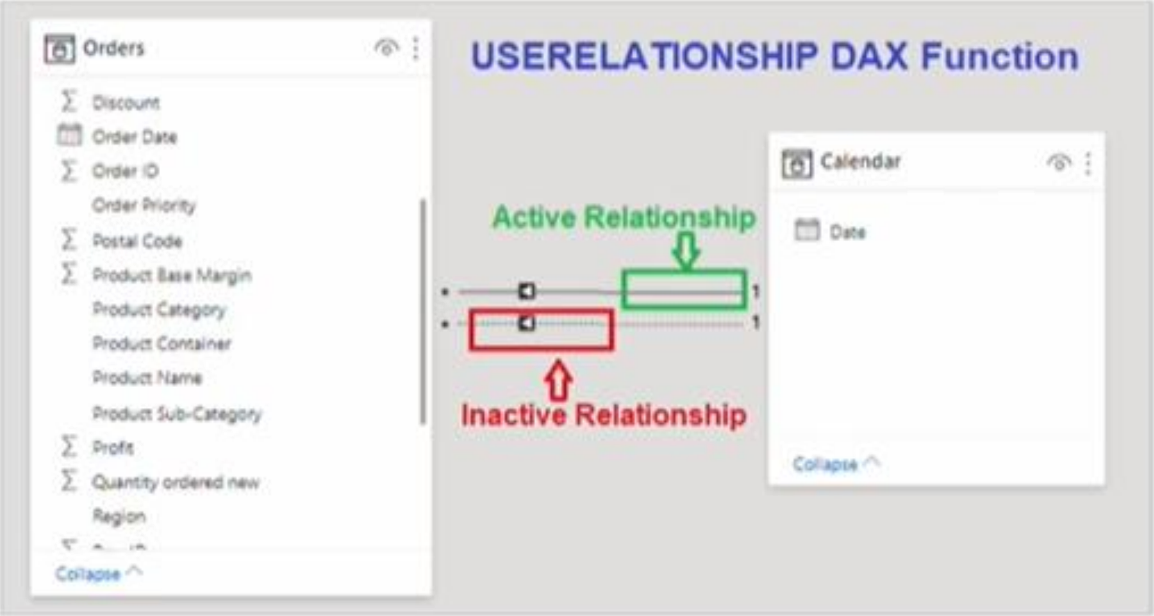


Table -> NewTable

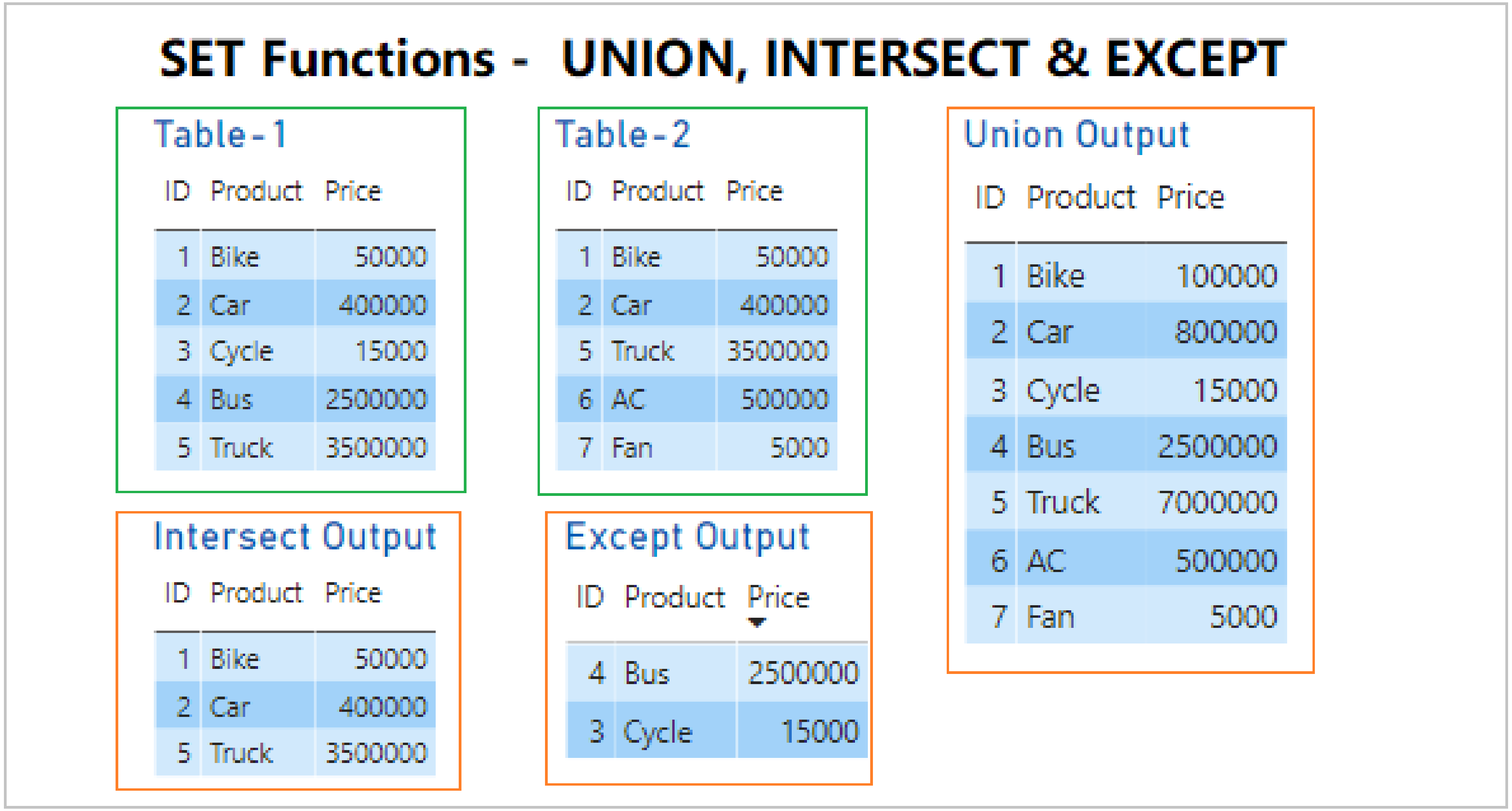
Clerks	Floor
John	Floor 1
Paul	Floor 2
Steve	Floor 3

DAX USERELATIONSHIP

Especifica una relación existente que se usará en la evaluación de una expresión DAX. La relación se define nombrando, como argumentos, las dos columnas que sirven como puntos finales.



EJERCICIO 5



SET Functions - UNION, INTERSECT & EXCEPT

Table-1

ID	Product	Price
1	Bike	50000
2	Car	400000
3	Cycle	15000
4	Bus	2500000
5	Truck	3500000

Table-2

ID	Product	Price
1	Bike	50000
2	Car	400000
5	Truck	3500000
6	AC	500000
7	Fan	5000

Union Output

ID	Product	Price
1	Bike	100000
2	Car	800000
3	Cycle	15000
4	Bus	2500000
5	Truck	7000000
6	AC	500000
7	Fan	5000

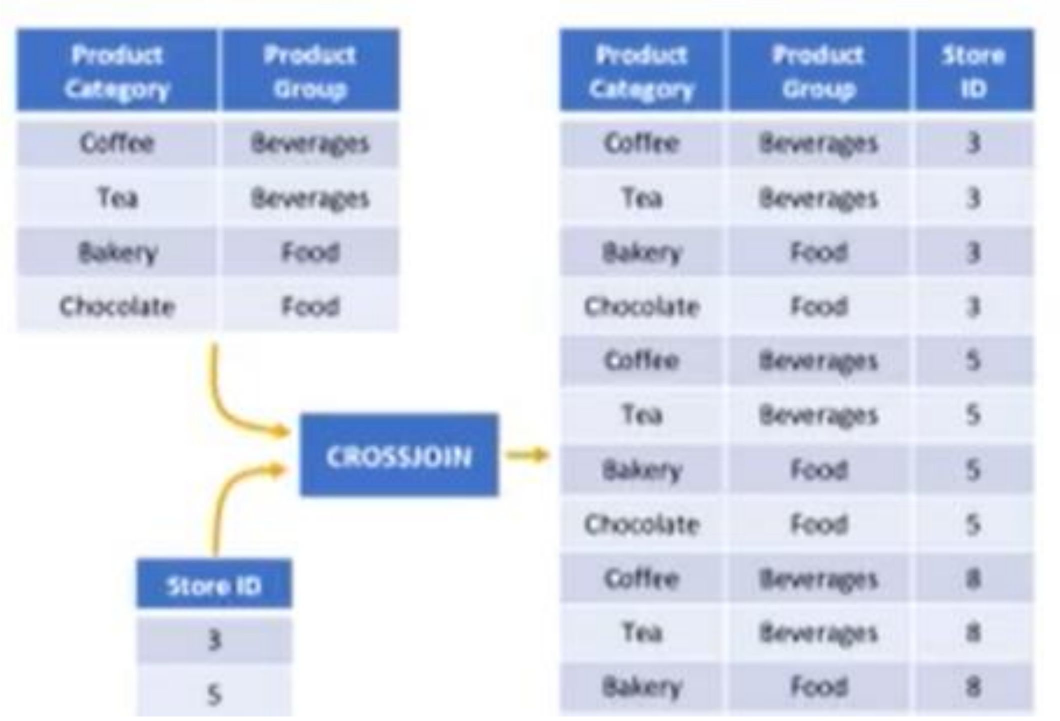
Intersect Output

ID	Product	Price
1	Bike	50000
2	Car	400000
5	Truck	3500000

Except Output

ID	Product	Price
4	Bus	2500000
3	Cycle	15000

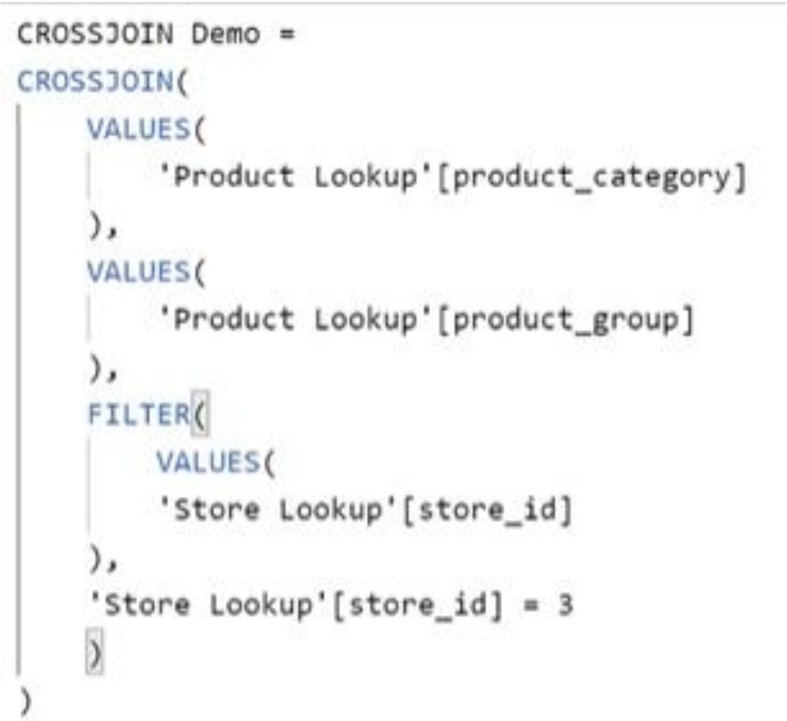
CROSSJOIN



Product Category	Product Group
Coffee	Beverages
Tea	Beverages
Bakery	Food
Chocolate	Food

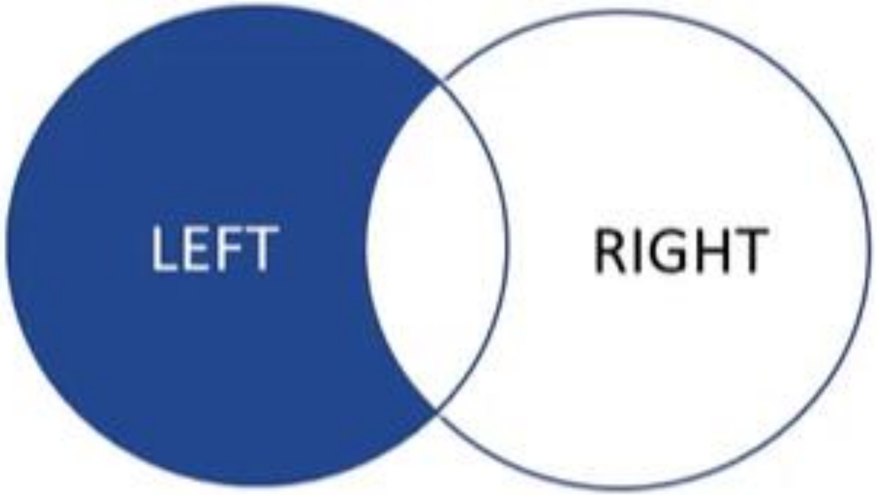
Store ID
3
5

Product Category	Product Group	Store ID
Coffee	Beverages	3
Tea	Beverages	3
Bakery	Food	3
Chocolate	Food	3
Coffee	Beverages	5
Tea	Beverages	5
Bakery	Food	5
Chocolate	Food	5
Coffee	Beverages	8
Tea	Beverages	8
Bakery	Food	8

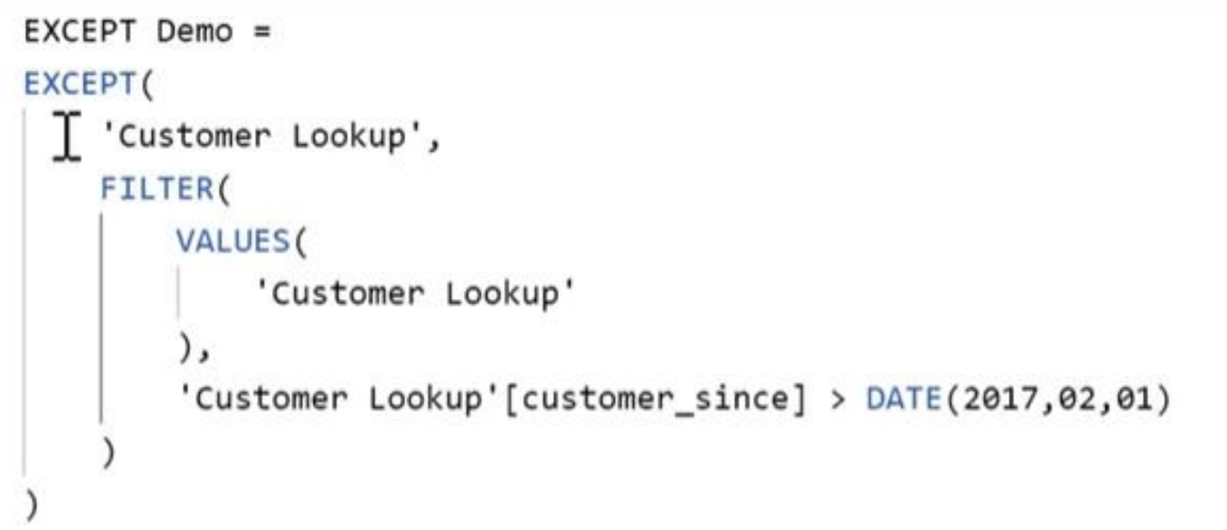


```
CROSSJOIN Demo =
CROSSJOIN(
    VALUES(
        'Product Lookup'[product_category]
    ),
    VALUES(
        'Product Lookup'[product_group]
    ),
    FILTER(
        VALUES(
        'Store Lookup'[store_id]
    ),
    'Store Lookup'[store_id] = 3
    )
)
```

EXCEPT

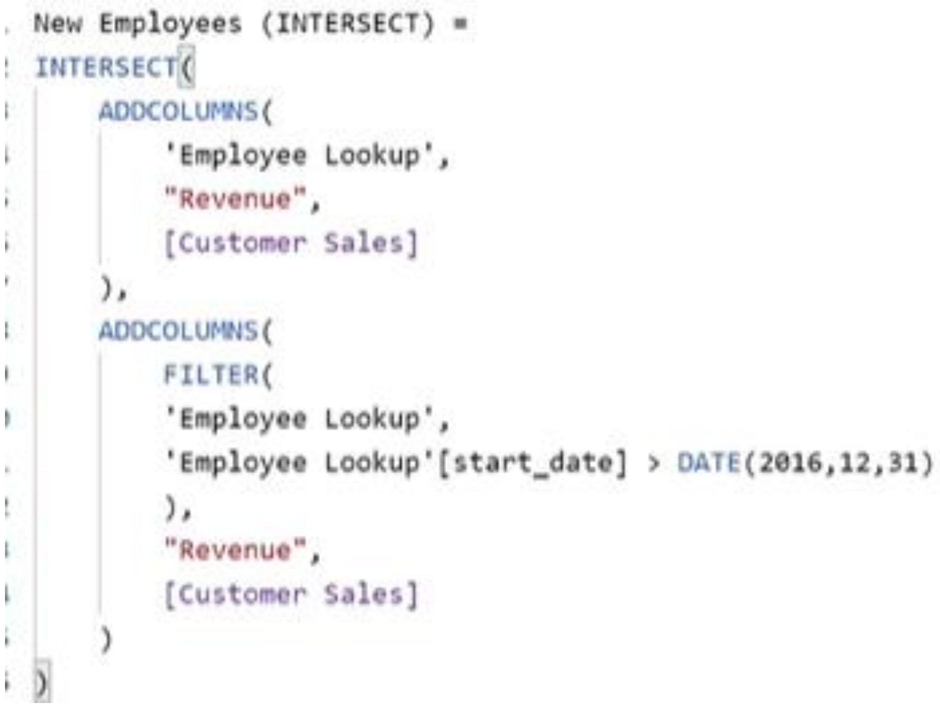


*Resulting table contains rows which ONLY appear in the **left table***

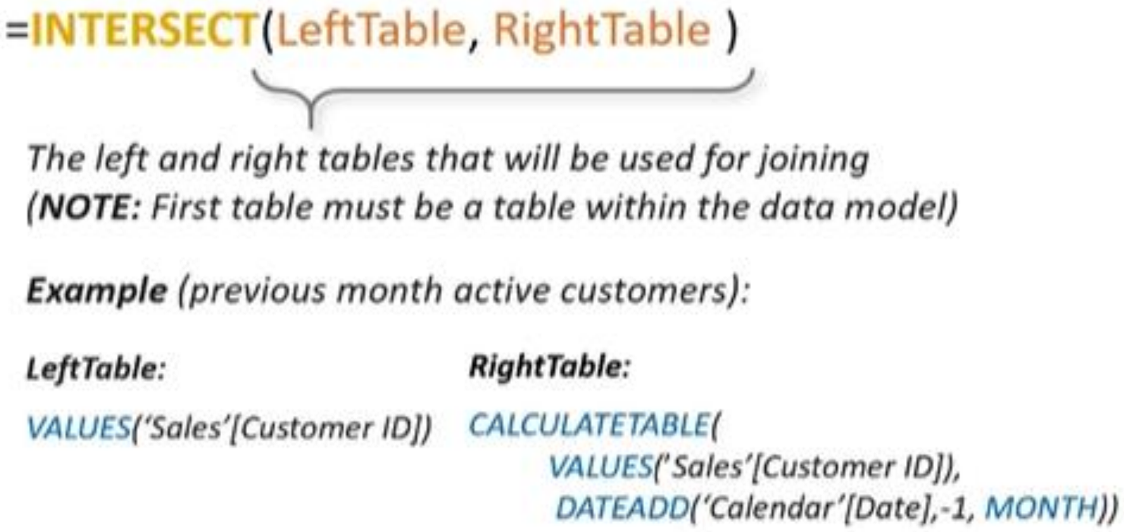


```
EXCEPT Demo =
EXCEPT(
    'Customer Lookup',
    FILTER(
        VALUES(
            'Customer Lookup'
        ),
        'Customer Lookup'[customer_since] > DATE(2017,02,01)
    )
)
```

INTERSECT



```
New Employees (INTERSECT) =
INTERSECT(
    ADDCOLUMNS(
        'Employee Lookup',
        "Revenue",
        [Customer Sales]
    ),
    ADDCOLUMNS(
        FILTER(
        'Employee Lookup',
        'Employee Lookup'[start_date] > DATE(2016,12,31)
        ),
        "Revenue",
        [Customer Sales]
    )
)
```



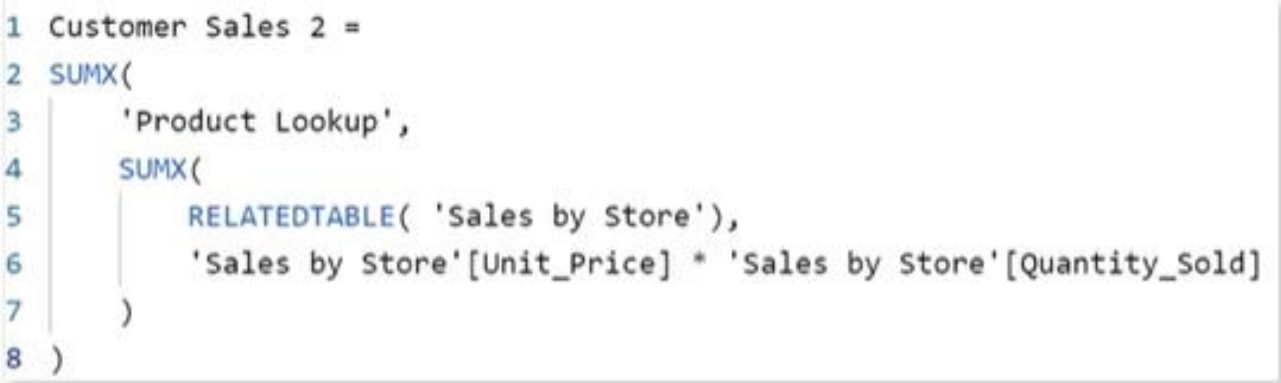
=INTERSECT(LeftTable, RightTable)

*The left and right tables that will be used for joining (**NOTE:** First table must be a table within the data model)*

***Example** (previous month active customers):*

LeftTable:	***RightTable:***

VALUES('Sales'[Customer ID])	*CALCULATETABLE(
	VALUES('Sales'[Customer ID]),
	DATEADD('Calendar'[Date],-1, MONTH))*

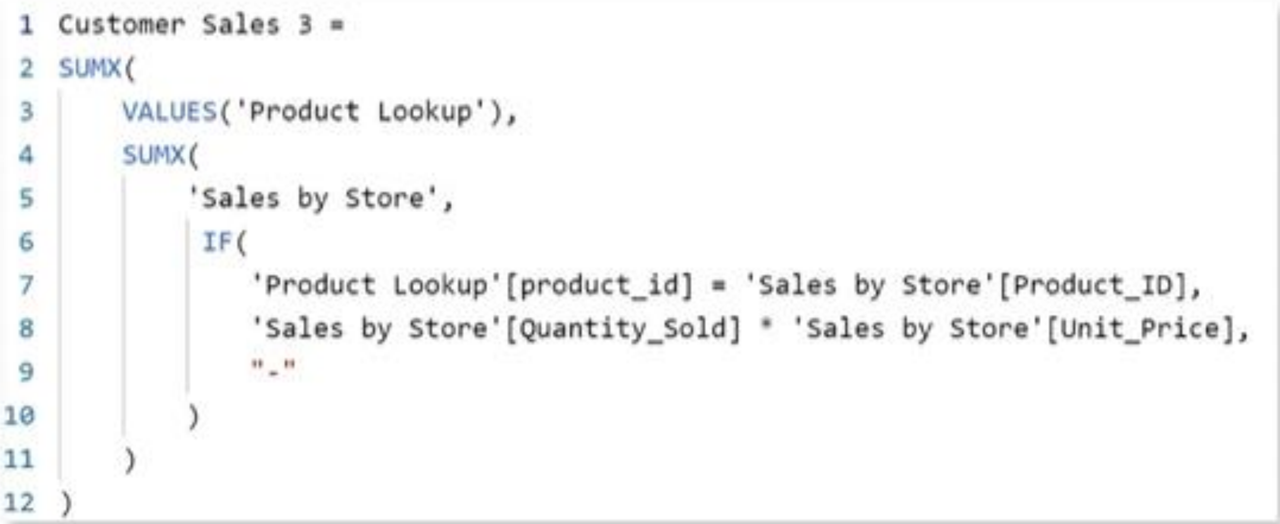


```
Customer Sales 2 =
SUMX(
    'Product Lookup',
    SUMX(
        RELATEDTABLE( 'Sales by Store'),
        'Sales by Store'[Unit_Price] * 'Sales by Store'[Quantity_Sold]
    )
)
```

Physical relationship

- Relationship between ‘Product Lookup’ and ‘Sales by Store’
- ‘Product Lookup’ contains **88 rows**
- ‘Sales by Store’ contains **907,841 rows**

Cardinality = 907,841

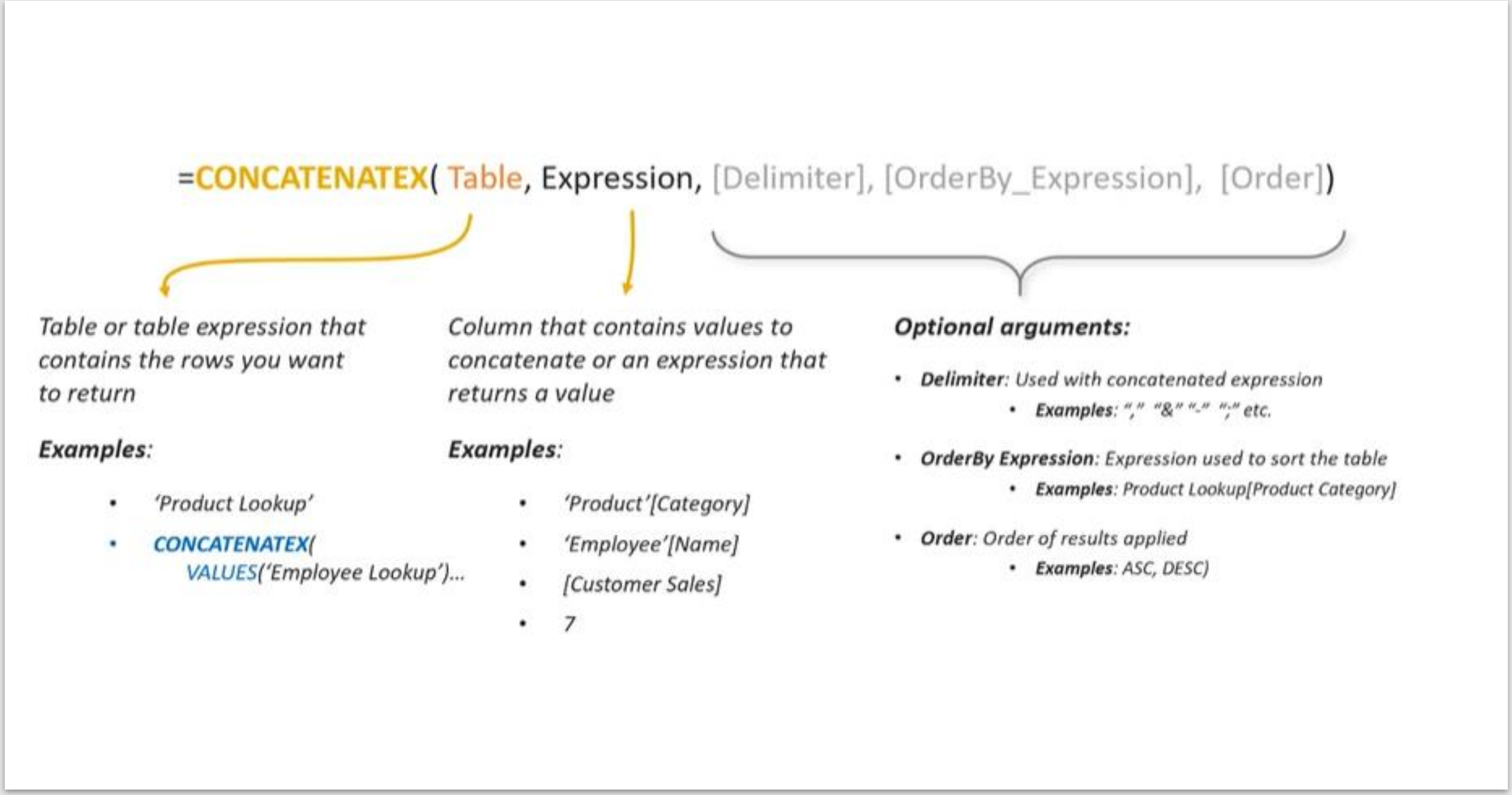


```
Customer Sales 3 =
SUMX(
    VALUES('Product Lookup'),
    SUMX(
        'Sales by Store',
        IF(
            'Product Lookup'[product_id] = 'Sales by Store'[Product_ID],
            'Sales by Store'[Quantity_Sold] * 'Sales by Store'[Unit_Price],
            "-"
        )
    )
)
```

Virtual relationship

- No physical relationship between ‘Product Lookup’ and ‘Sales by Store’
- ‘Product Lookup’ contains **88 rows**
- ‘Sales by Store’ contains **907,841 rows**

Cardinality = 79,890,008



=CONCATENATEX(Table, Expression, [Delimiter], [OrderBy_Expression], [Order])

Table or table expression that contains the rows you want to return

***Examples**:*

- *'Product Lookup'*
- ***CONCATENATEX(***
  VALUES('Employee Lookup')...

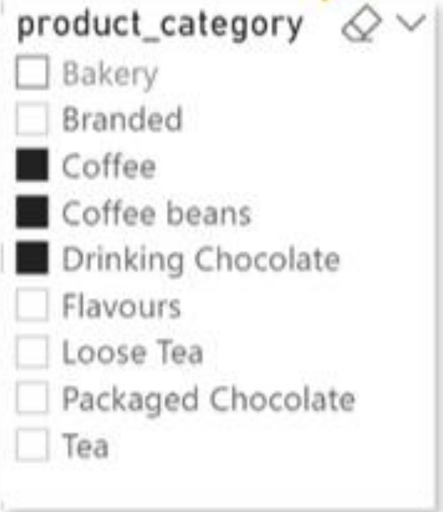
Column that contains values to concatenate or an expression that returns a value

***Examples**:*

- *'Product'[Category]*
- *'Employee'[Name]*
- *[Customer Sales]*
- *7*

Optional arguments:

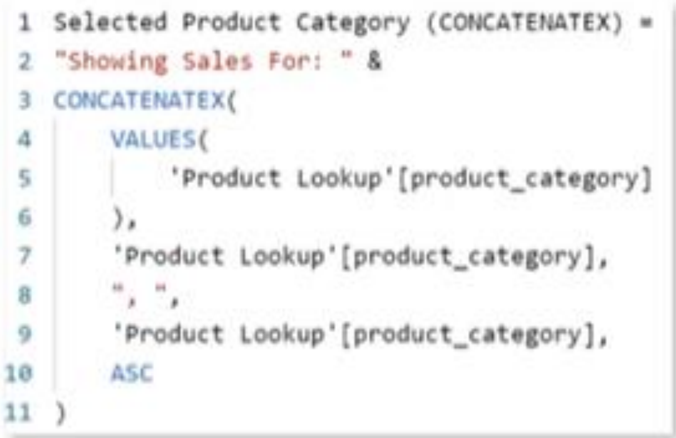
- ***Delimiter**: Used with concatenated expression*
  - ***Examples**: "," "&" "-" ";" etc.*
- ***OrderBy Expression**: Expression used to sort the table*
  - ***Examples**: Product Lookup[Product Category]*
- ***Order**: Order of results applied*
  - ***Examples**: ASC, DESC)*



product_category

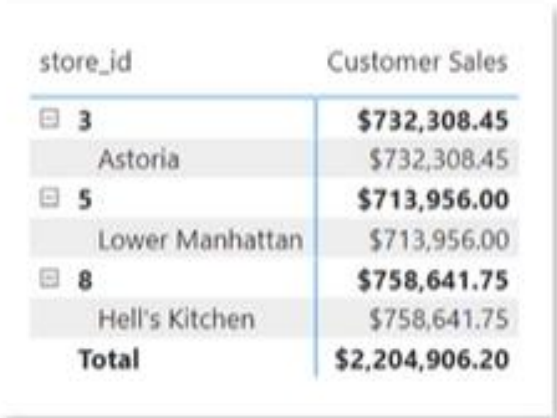
- □ Bakery
- □ Branded
- ■ Coffee
- ■ Coffee beans
- ■ Drinking Chocolate
- □ Flavours
- □ Loose Tea
- □ Packaged Chocolate
- □ Tea

*Product Category slicer is filtered to **Coffee**, **Coffee Beans** & **Drinking Chocolate***

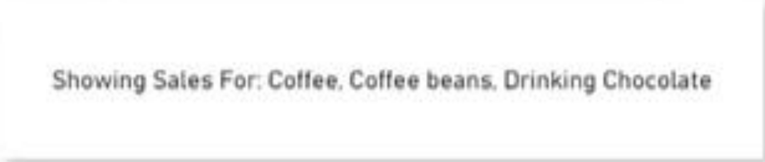


```
Selected Product Category (CONCATENATEX) =
"Showing Sales For: " &
CONCATENATEX(
    VALUES(
        'Product Lookup'[product_category]
    ),
    'Product Lookup'[product_category],
    ", ",
    'Product Lookup'[product_category],
    ASC
)
```

*The **Selected Product Category** measure uses **CONCATENATEX** to capture the selections from the slicer...*



store_id	Customer Sales
⊟ **3**	**$732,308.45**
Astoria	$732,308.45
⊟ **5**	**$713,956.00**
Lower Manhattan	$713,956.00
⊟ **8**	**$758,641.75**
Hell's Kitchen	$758,641.75
Total	**$2,204,906.20**



Showing Sales For: Coffee, Coffee beans, Drinking Chocolate

EJERCICIO 6

AVERAGEX()

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

=AVERAGEX(Table, Expression)

Table, or table expression, that contains the rows to evaluate

Examples*:*

- *'Calendar'*
- *'Product Lookup'*

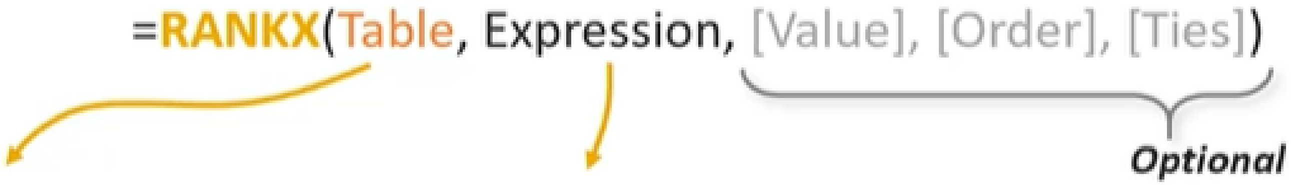
The expression that you want to evaluate

Examples*:*

- *[Customer Sales]*
- *SUM[quantity_sold]*

RANKX()

Returns the ranking of a number in a list of numbers for each row in the table argument



```
=RANKX(Table, Expression, [Value], [Order], [Ties])
```

*Table or **DAX expression** that returns a table*

Examples:

- *ALL('Product Lookup'[Product])*

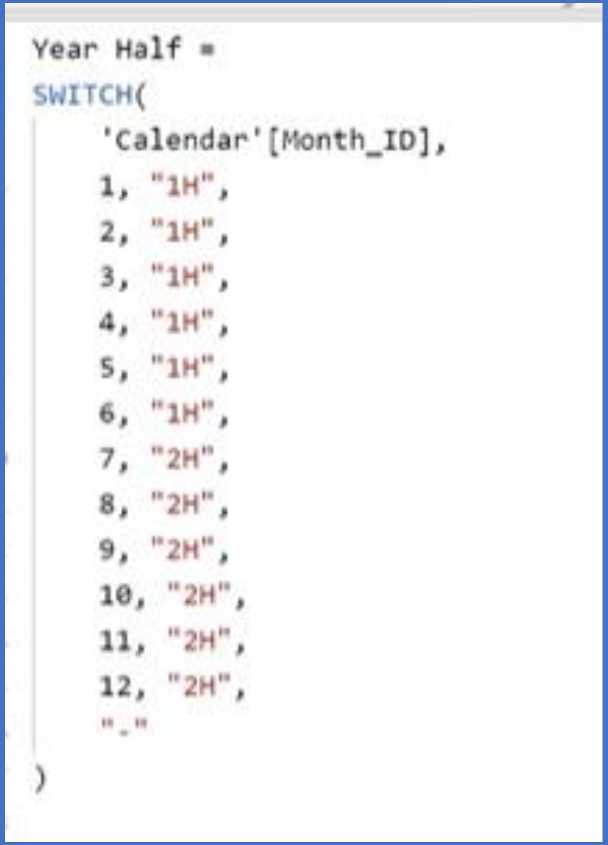
An expression that returns a scalar value, evaluated at each row of the table

***Examples**:*

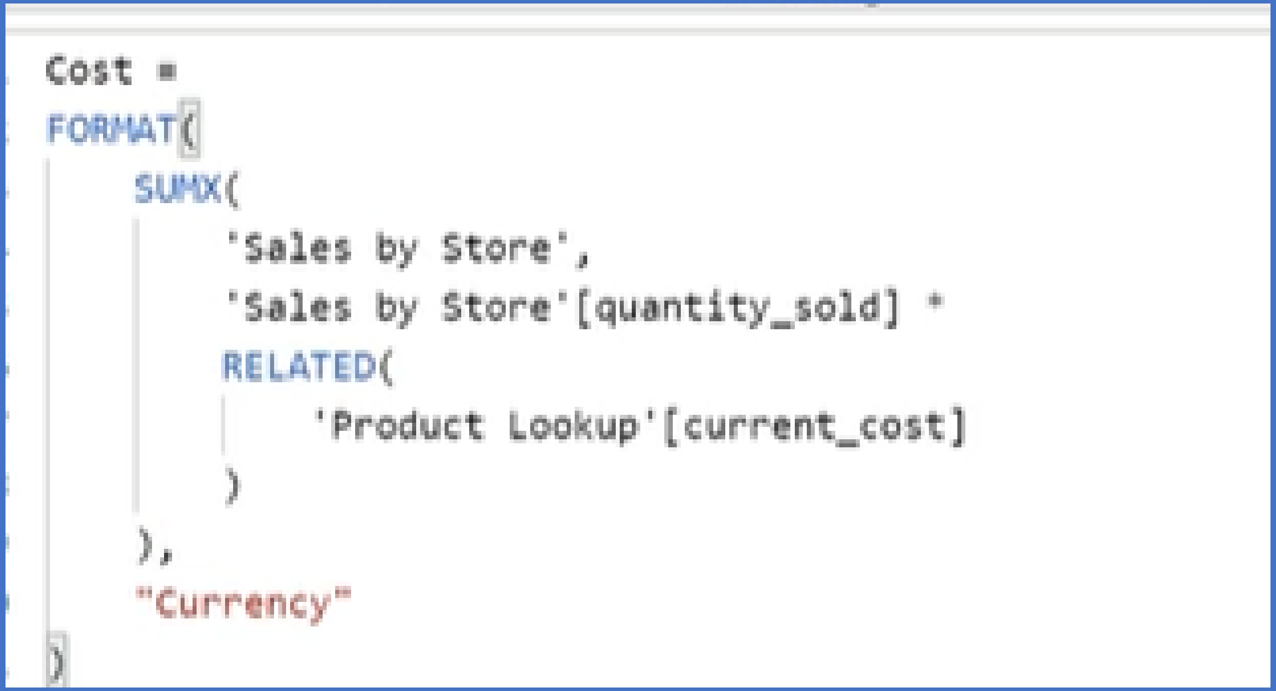
- *[Customer Sales]*
- *SUM('Sales by Store'[quantity_sold])*

Optional arguments:

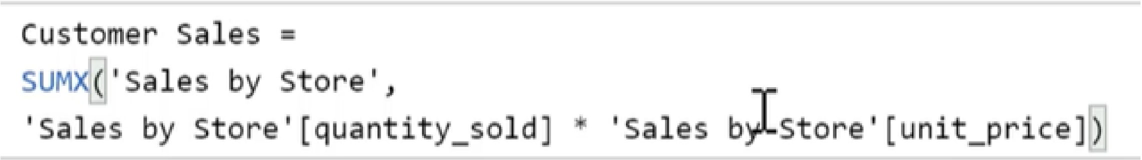
- ***Value**: Any DAX expression that returns a single scalar value whose rank is to be found (by default, the value in the **current row** is used)*
- ***Order**: Specifies how to rank (low-high vs. high-low)*
  - ***Examples**: ASC or DESC*
- ***Ties**: Determines how ties and following ranks are treated:*
  - ***SKIP** (default): Skips ranks after ties*
  - ***DENSE:** Shows the next rank, regardless of ties*



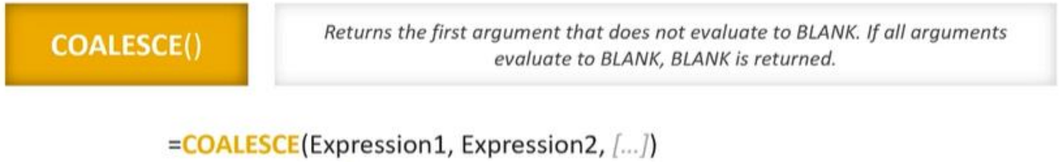
```
Year Half =
SWITCH(
    'Calendar'[Month_ID],
    1, "1H",
    2, "1H",
    3, "1H",
    4, "1H",
    5, "1H",
    6, "1H",
    7, "2H",
    8, "2H",
    9, "2H",
    10, "2H",
    11, "2H",
    12, "2H",
    "-"
)
```



```
Cost =
FORMAT(
    SUMX(
        'Sales by Store',
        'Sales by Store'[quantity_sold] *
        RELATED(
            'Product Lookup'[current_cost]
        )
    ),
    "Currency"
)
```



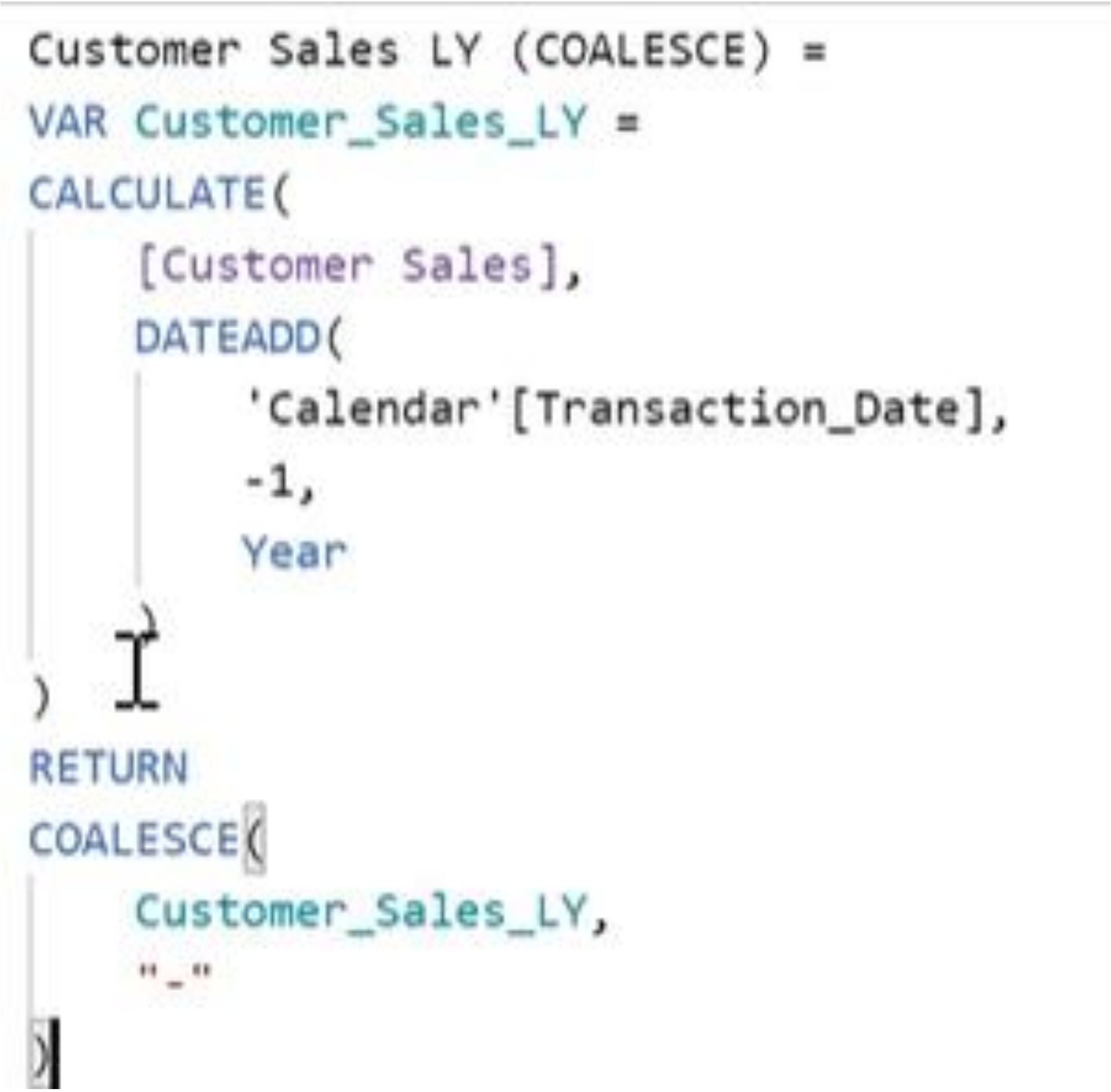
```
Customer Sales =
SUMX('Sales by Store',
'Sales by Store'[quantity_sold] * 'Sales by Store'[unit_price])
```



COALESCE()

Returns the first argument that does not evaluate to BLANK. If all arguments evaluate to BLANK, BLANK is returned.

=COALESCE(Expression1, Expression2, *[...]*)



```
Customer Sales LY (COALESCE) =
VAR Customer_Sales_LY =
CALCULATE(
    [Customer Sales],
    DATEADD(
        'Calendar'[Transaction_Date],
        -1,
        Year
    )
)
RETURN
COALESCE(
    Customer_Sales_LY,
    "-"
)
```

CROSSFILTER()

Specifies cross filtering direction to be used for the duration of the DAX expression. The relationship is defined by naming the two columns that serve as endpoints

=CROSSFILTER(LeftColumnName, RightColumnName2, CrossFilterType)

*The two columns you want to use. **Left column** is typically the "**many**" side and **right column** is typically the "**one**" side*

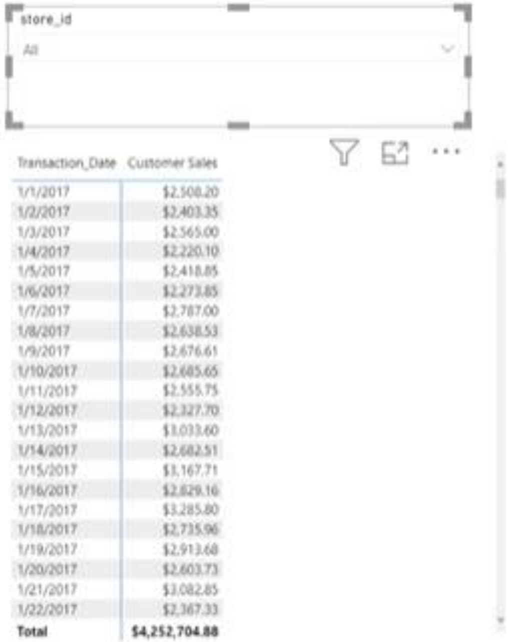
***Examples**:*

- *'Sales by Store'[customer_id]*
- *‘Customer Lookup'[customer_id]*

Specifies the direction of the CROSSFILTER

***Examples**:*

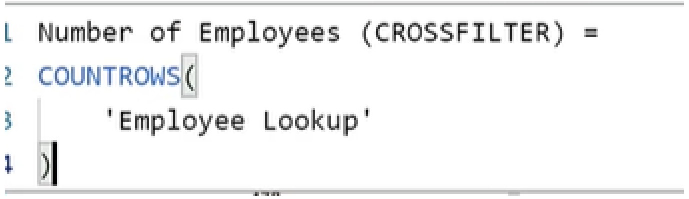
- *OneWay, Both, None*



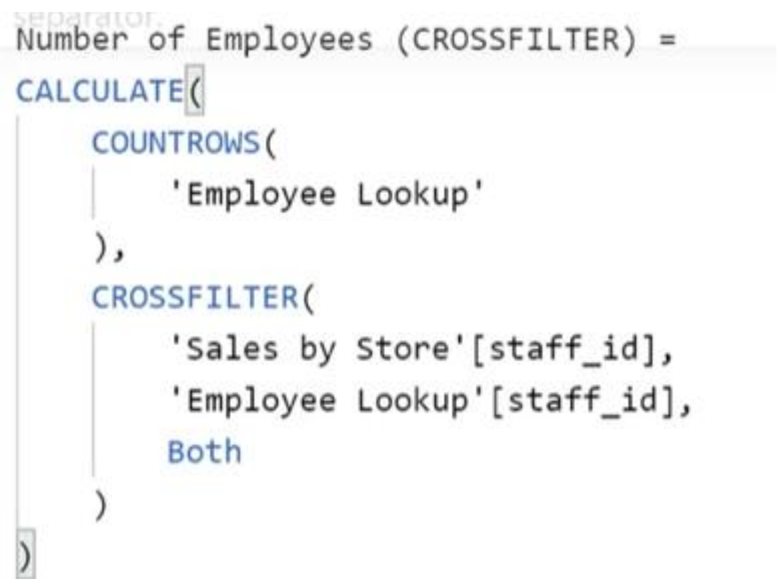
store_id

All

Transaction_Date	Customer Sales
1/1/2017	$2,508.20
1/2/2017	$2,403.35
1/3/2017	$2,565.00
1/4/2017	$2,220.10
1/5/2017	$2,418.85
1/6/2017	$2,273.85
1/7/2017	$2,787.00
1/8/2017	$2,638.53
1/9/2017	$2,676.61
1/10/2017	$2,685.65
1/11/2017	$2,555.75
1/12/2017	$2,327.70
1/13/2017	$3,033.60
1/14/2017	$2,682.51
1/15/2017	$3,167.71
1/16/2017	$2,829.16
1/17/2017	$3,285.80
1/18/2017	$2,735.96
1/19/2017	$2,913.68
1/20/2017	$2,603.73
1/21/2017	$3,082.85
1/22/2017	$2,367.33
Total	**$4,252,704.88**



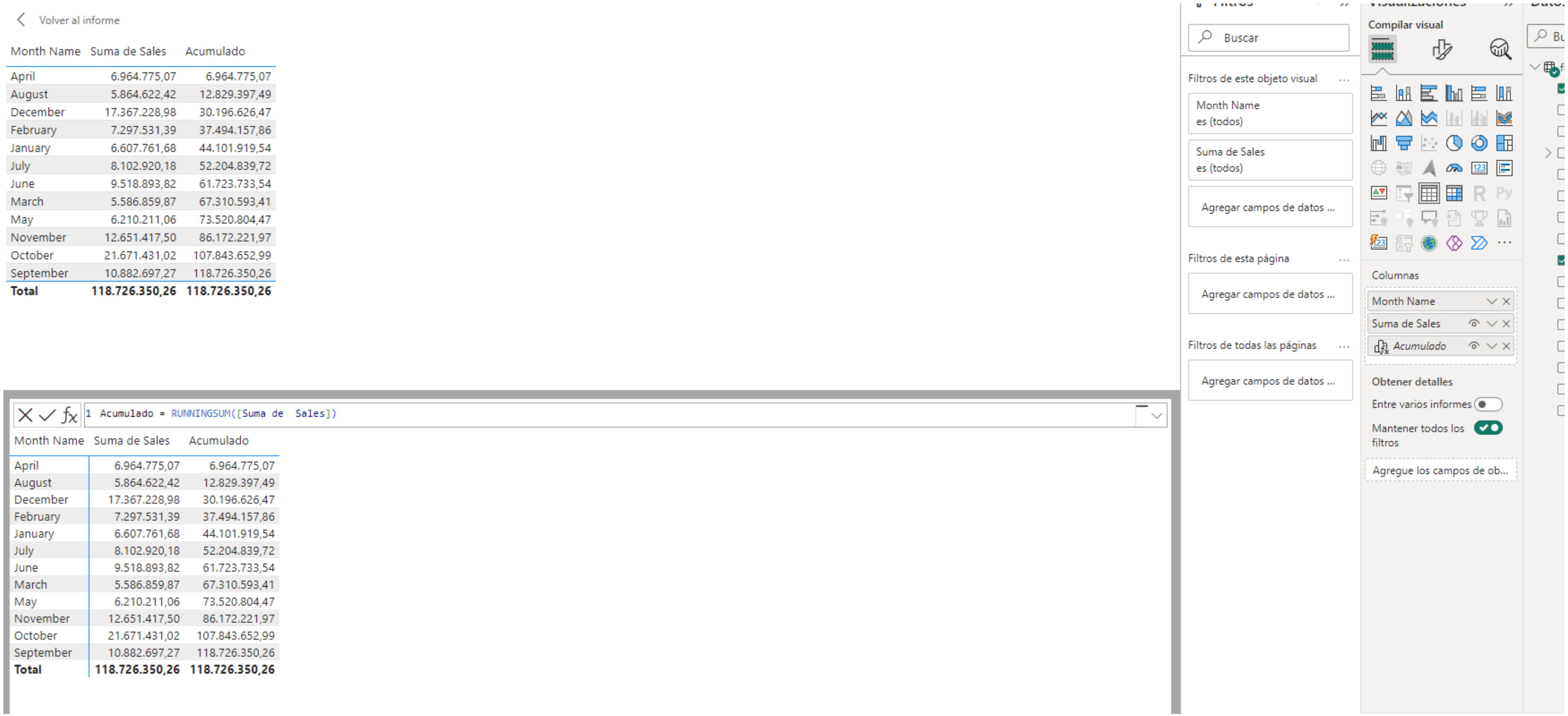
```
Number of Employees (CROSSFILTER) =
COUNTROWS(
    'Employee Lookup'
)
```



separator.

```
Number of Employees (CROSSFILTER) =
CALCULATE(
    COUNTROWS(
        'Employee Lookup'
    ),
    CROSSFILTER(
        'Sales by Store'[staff_id],
        'Employee Lookup'[staff_id],
        Both
    )
)
```

EJERCICIO 7



Month Name	Suma de Sales	Acumulado
April	6.964.775,07	6.964.775,07
August	5.864.622,42	12.829.397,49
December	17.367.228,98	30.196.626,47
February	7.297.531,39	37.494.157,86
January	6.607.761,68	44.101.919,54
July	8.102.920,18	52.204.839,72
June	9.518.893,82	61.723.733,54
March	5.586.859,87	67.310.593,41
May	6.210.211,06	73.520.804,47
November	12.651.417,50	86.172.221,97
October	21.671.431,02	107.843.652,99
September	10.882.697,27	118.726.350,26
Total	**118.726.350,26**	**118.726.350,26**

```
1 Acumulado = RUNNINGSUM([Suma de  Sales])
```

Month Name	Suma de Sales	Acumulado
April	6.964.775,07	6.964.775,07
August	5.864.622,42	12.829.397,49
December	17.367.228,98	30.196.626,47
February	7.297.531,39	37.494.157,86
January	6.607.761,68	44.101.919,54
July	8.102.920,18	52.204.839,72
June	9.518.893,82	61.723.733,54
March	5.586.859,87	67.310.593,41
May	6.210.211,06	73.520.804,47
November	12.651.417,50	86.172.221,97
October	21.671.431,02	107.843.652,99
September	10.882.697,27	118.726.350,26
Total	**118.726.350,26**	**118.726.350,26**

Buscar
Filtros de este objeto visual
Agregar campos de datos ...
Filtros de esta página
Agregar campos de datos ...
Filtros de todas las páginas
Agregar campos de datos ...
Compilar visual
Columnas
Obtener detalles
Agregue los campos de ob...
Bu

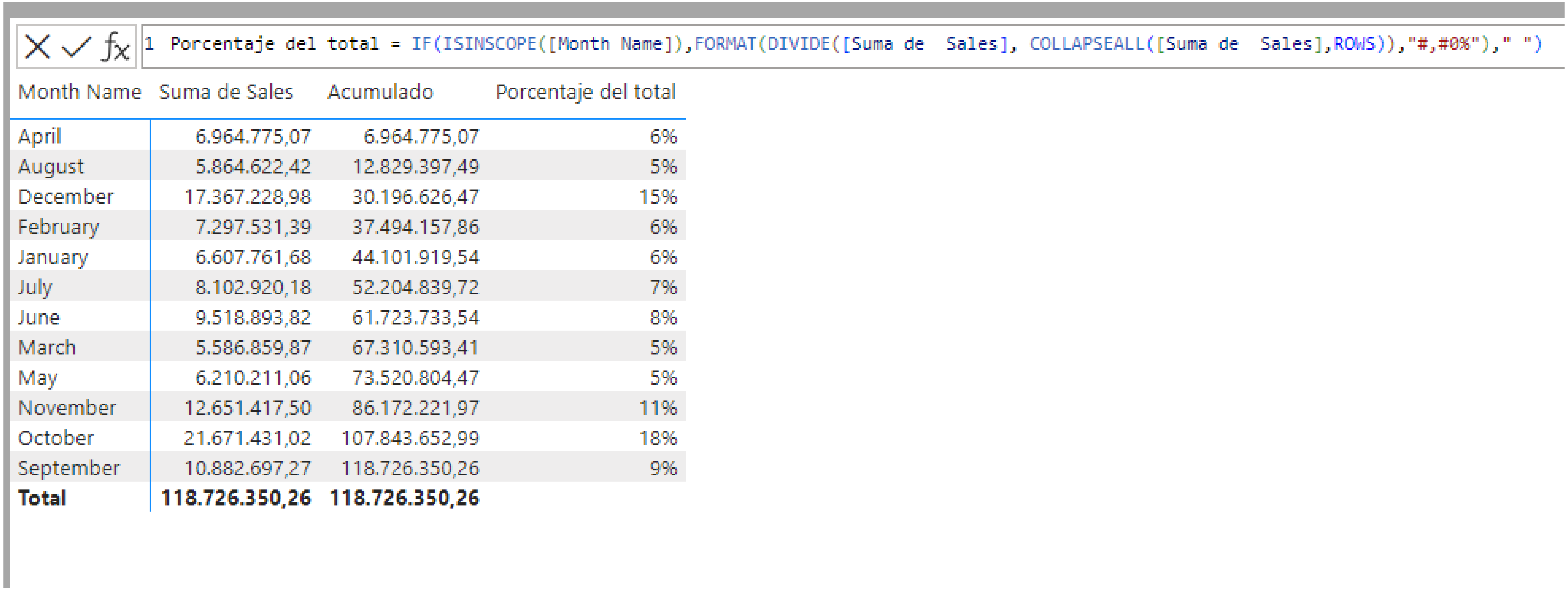
Month Name
es (todos)

Suma de Sales
es (todos)

Columnas: Month Name, Suma de Sales, Acumulado

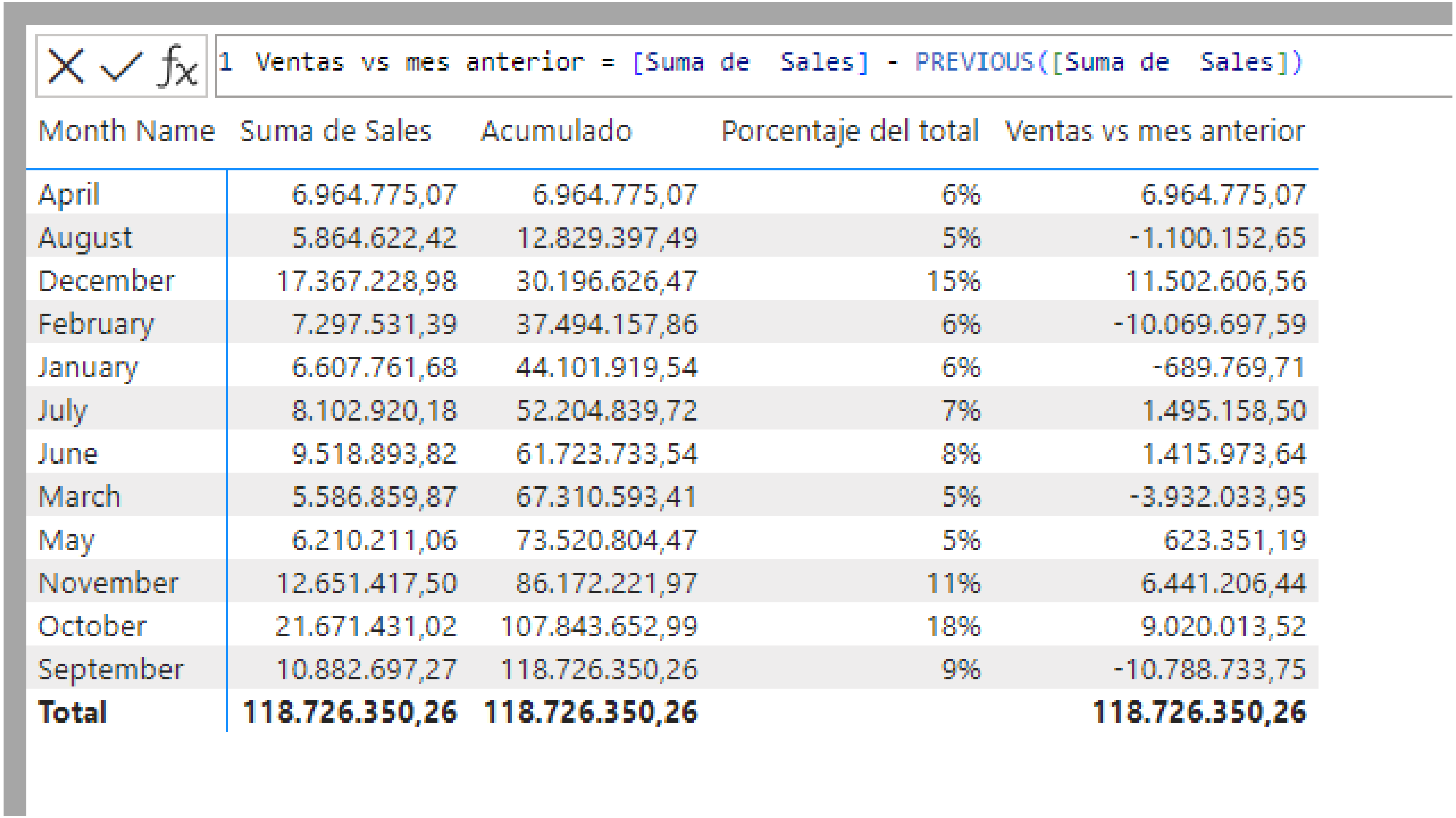
Entre varios informes ○

Mantener todos los filtros ●



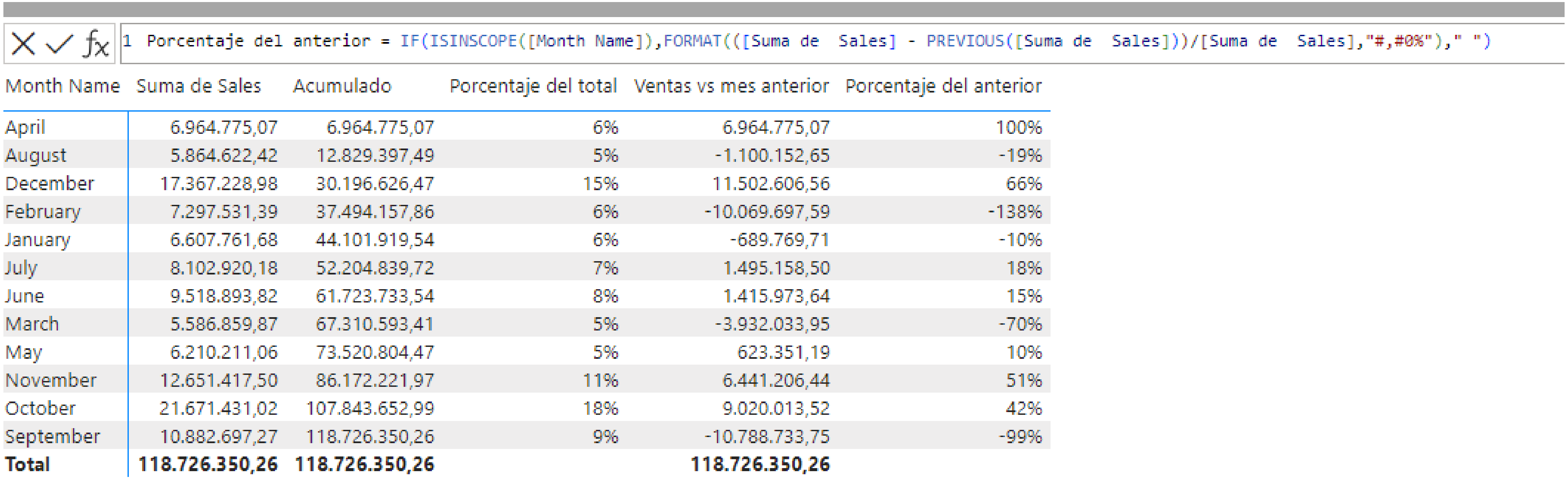
```
Porcentaje del total = IF(ISINSCOPE([Month Name]),FORMAT(DIVIDE([Suma de  Sales], COLLAPSEALL([Suma de  Sales],ROWS)),"#,#0%")," ")
```

Month Name	Suma de Sales	Acumulado	Porcentaje del total
April	6.964.775,07	6.964.775,07	6%
August	5.864.622,42	12.829.397,49	5%
December	17.367.228,98	30.196.626,47	15%
February	7.297.531,39	37.494.157,86	6%
January	6.607.761,68	44.101.919,54	6%
July	8.102.920,18	52.204.839,72	7%
June	9.518.893,82	61.723.733,54	8%
March	5.586.859,87	67.310.593,41	5%
May	6.210.211,06	73.520.804,47	5%
November	12.651.417,50	86.172.221,97	11%
October	21.671.431,02	107.843.652,99	18%
September	10.882.697,27	118.726.350,26	9%
Total	**118.726.350,26**	**118.726.350,26**	



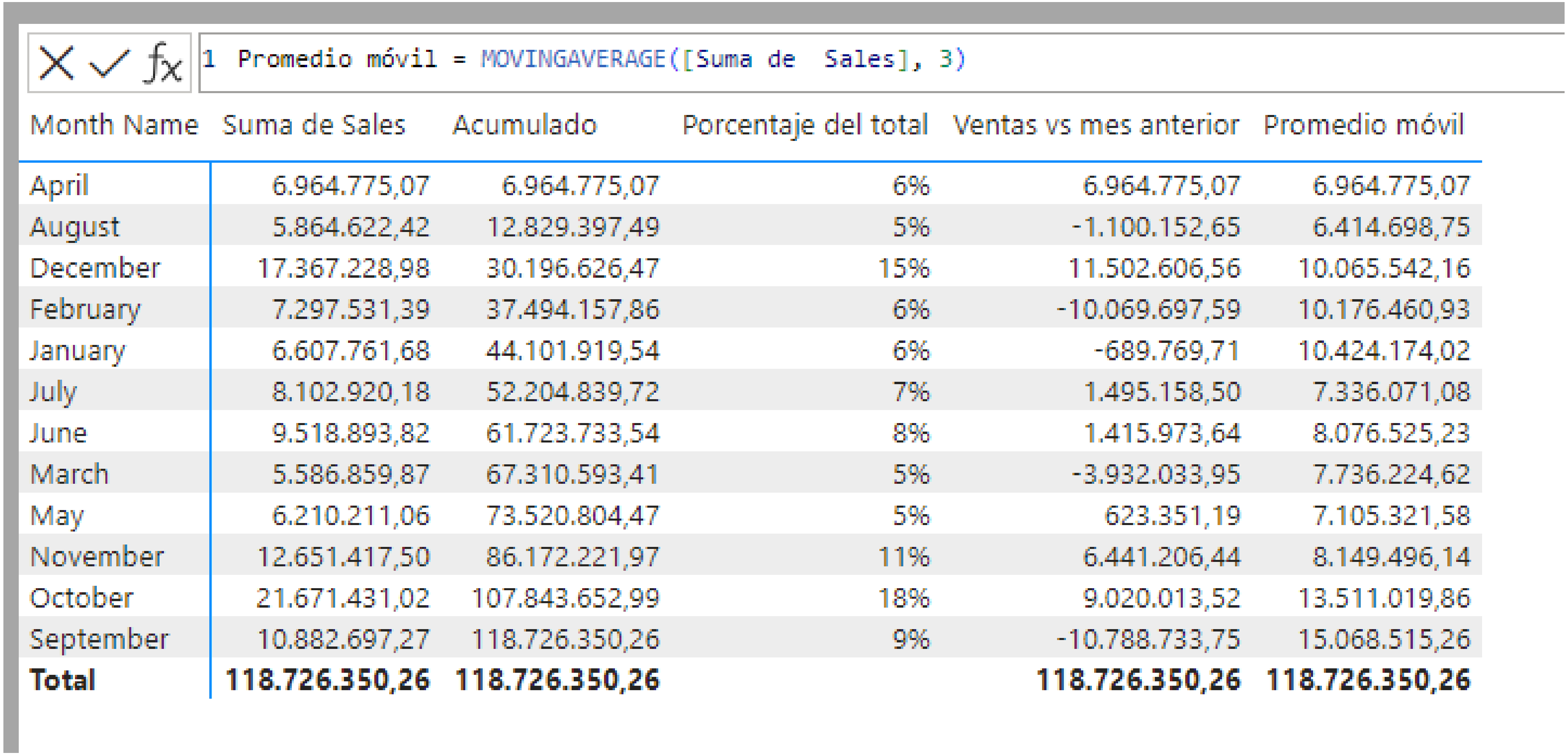
```
Ventas vs mes anterior = [Suma de  Sales] - PREVIOUS([Suma de  Sales])
```

Month Name	Suma de Sales	Acumulado	Porcentaje del total	Ventas vs mes anterior
April	6.964.775,07	6.964.775,07	6%	6.964.775,07
August	5.864.622,42	12.829.397,49	5%	-1.100.152,65
December	17.367.228,98	30.196.626,47	15%	11.502.606,56
February	7.297.531,39	37.494.157,86	6%	-10.069.697,59
January	6.607.761,68	44.101.919,54	6%	-689.769,71
July	8.102.920,18	52.204.839,72	7%	1.495.158,50
June	9.518.893,82	61.723.733,54	8%	1.415.973,64
March	5.586.859,87	67.310.593,41	5%	-3.932.033,95
May	6.210.211,06	73.520.804,47	5%	623.351,19
November	12.651.417,50	86.172.221,97	11%	6.441.206,44
October	21.671.431,02	107.843.652,99	18%	9.020.013,52
September	10.882.697,27	118.726.350,26	9%	-10.788.733,75
Total	**118.726.350,26**	**118.726.350,26**		**118.726.350,26**



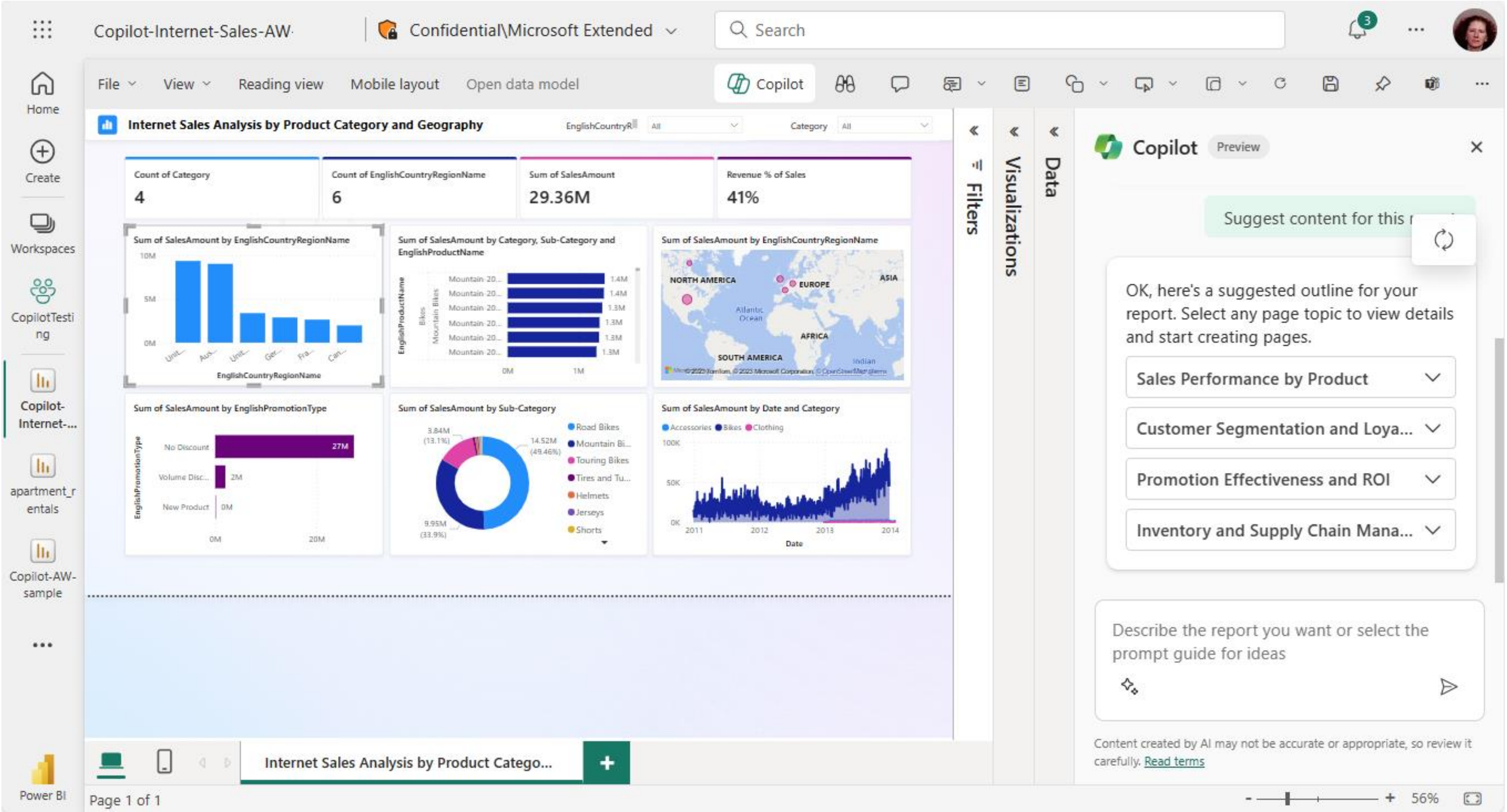
```
Porcentaje del anterior = IF(ISINSCOPE([Month Name]),FORMAT(([Suma de  Sales] - PREVIOUS([Suma de  Sales]))/[Suma de  Sales],"#,#0%")," ")
```

Month Name	Suma de Sales	Acumulado	Porcentaje del total	Ventas vs mes anterior	Porcentaje del anterior
April	6.964.775,07	6.964.775,07	6%	6.964.775,07	100%
August	5.864.622,42	12.829.397,49	5%	-1.100.152,65	-19%
December	17.367.228,98	30.196.626,47	15%	11.502.606,56	66%
February	7.297.531,39	37.494.157,86	6%	-10.069.697,59	-138%
January	6.607.761,68	44.101.919,54	6%	-689.769,71	-10%
July	8.102.920,18	52.204.839,72	7%	1.495.158,50	18%
June	9.518.893,82	61.723.733,54	8%	1.415.973,64	15%
March	5.586.859,87	67.310.593,41	5%	-3.932.033,95	-70%
May	6.210.211,06	73.520.804,47	5%	623.351,19	10%
November	12.651.417,50	86.172.221,97	11%	6.441.206,44	51%
October	21.671.431,02	107.843.652,99	18%	9.020.013,52	42%
September	10.882.697,27	118.726.350,26	9%	-10.788.733,75	-99%
Total	**118.726.350,26**	**118.726.350,26**		**118.726.350,26**	



```
Promedio móvil = MOVINGAVERAGE([Suma de  Sales], 3)
```

Month Name	Suma de Sales	Acumulado	Porcentaje del total	Ventas vs mes anterior	Promedio móvil
April	6.964.775,07	6.964.775,07	6%	6.964.775,07	6.964.775,07
August	5.864.622,42	12.829.397,49	5%	-1.100.152,65	6.414.698,75
December	17.367.228,98	30.196.626,47	15%	11.502.606,56	10.065.542,16
February	7.297.531,39	37.494.157,86	6%	-10.069.697,59	10.176.460,93
January	6.607.761,68	44.101.919,54	6%	-689.769,71	10.424.174,02
July	8.102.920,18	52.204.839,72	7%	1.495.158,50	7.336.071,08
June	9.518.893,82	61.723.733,54	8%	1.415.973,64	8.076.525,23
March	5.586.859,87	67.310.593,41	5%	-3.932.033,95	7.736.224,62
May	6.210.211,06	73.520.804,47	5%	623.351,19	7.105.321,58
November	12.651.417,50	86.172.221,97	11%	6.441.206,44	8.149.496,14
October	21.671.431,02	107.843.652,99	18%	9.020.013,52	13.511.019,86
September	10.882.697,27	118.726.350,26	9%	-10.788.733,75	15.068.515,26
Total	**118.726.350,26**	**118.726.350,26**		**118.726.350,26**	**118.726.350,26**



Copilot Preview

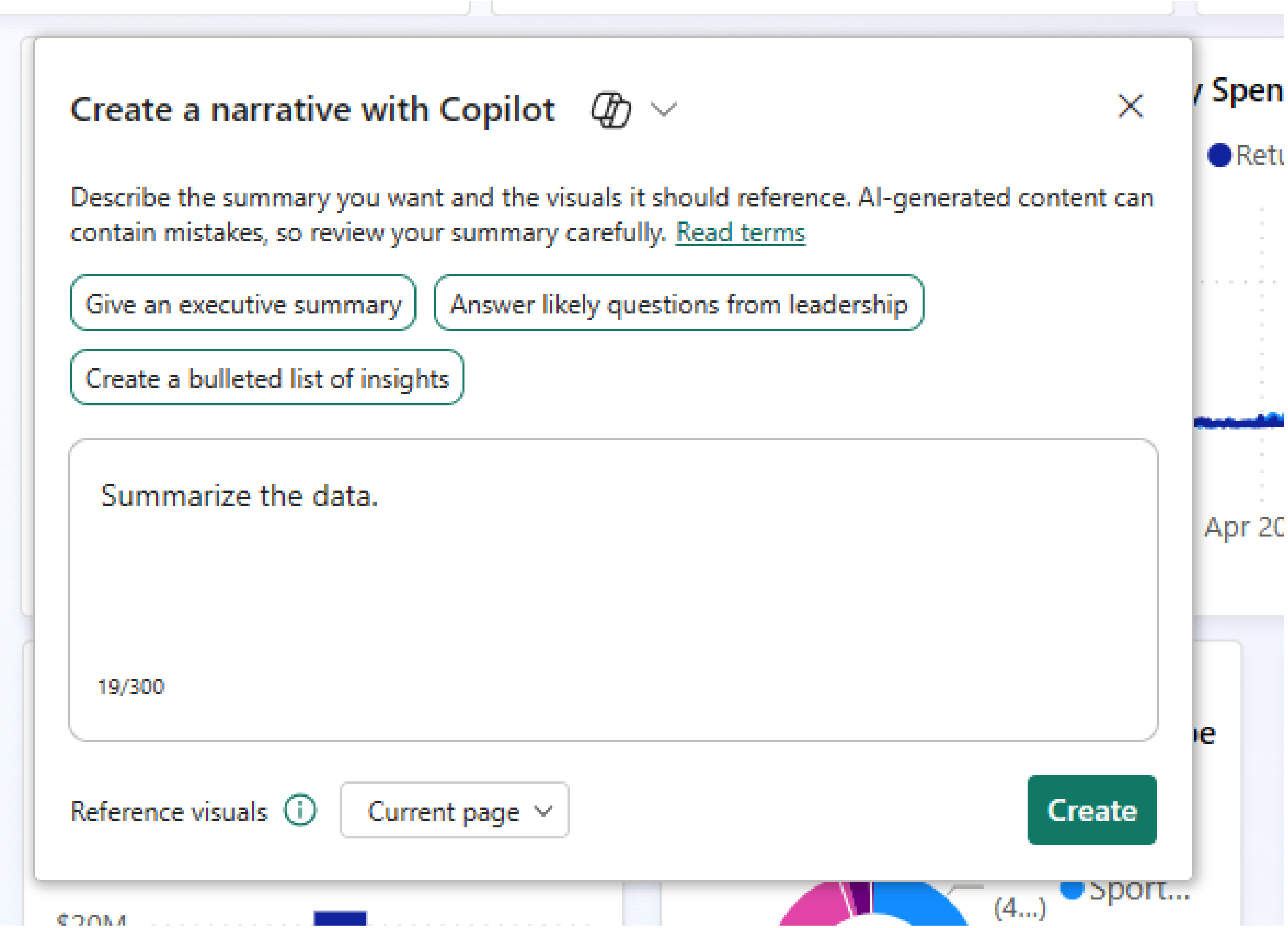
Suggest content for this r

OK, here's a suggested outline for your report. Select any page topic to view details and start creating pages.

- Sales Performance by Product
- Customer Segmentation and Loya...
- Promotion Effectiveness and ROI
- Inventory and Supply Chain Mana...

Describe the report you want or select the prompt guide for ideas

Content created by AI may not be accurate or appropriate, so review it carefully. Read terms



Create a narrative with Copilot

Describe the summary you want and the visuals it should reference. AI-generated content can contain mistakes, so review your summary carefully. Read terms

Give an executive summary
Answer likely questions from leadership
Create a bulleted list of insights

Summarize the data.

19/300

Reference visuals: Current page

Create
Spen
Retu
Apr 20
(4...)
Sport...
$20M

EJERCICIO 8

EJERCICIO 9