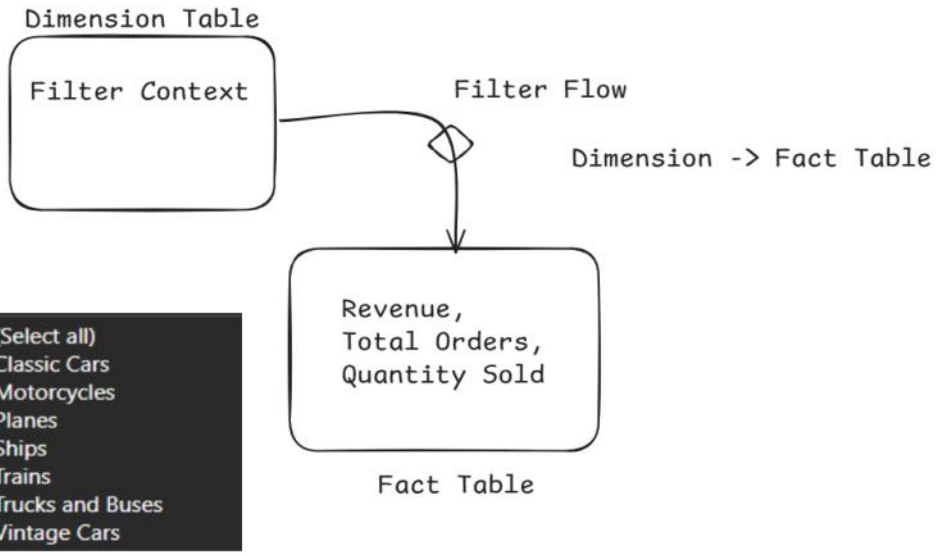
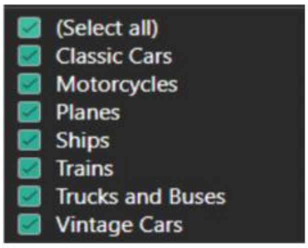
CN-PowerBI-Lec05-DAX Functions in Power BI - II

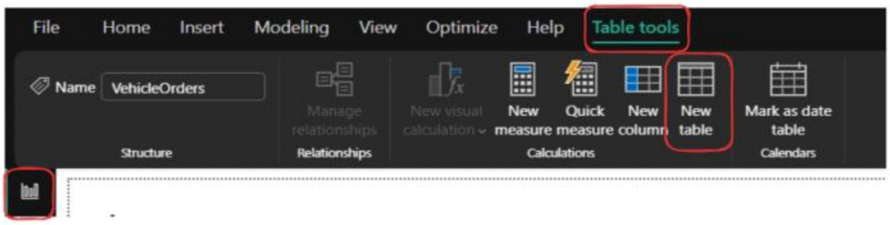
DAX Functions in Power BI - II

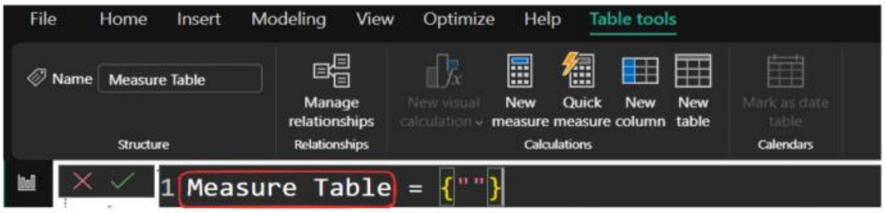




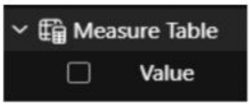
Measure Table

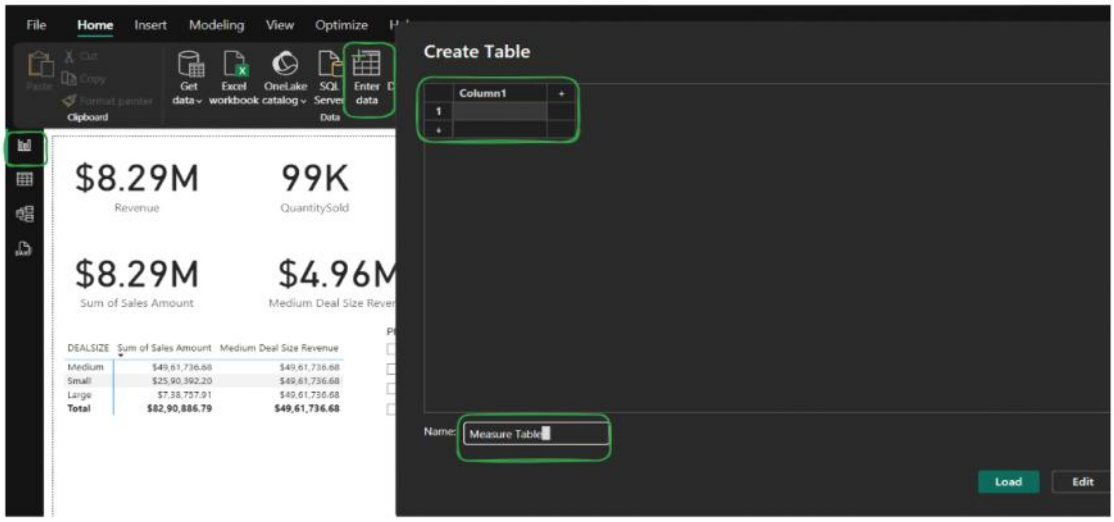
Option1 :





```
Measure Table = {""}
```

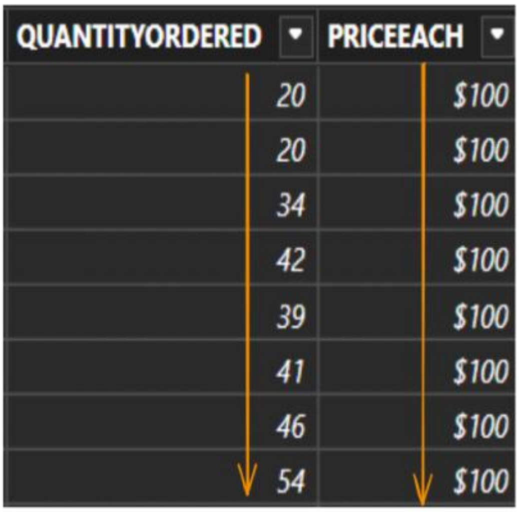




DEALSIZE	Sum of Sales Amount	Medium Deal Size Revenue
Medium	$49,61,736.68	$49,61,736.68
Small	$25,90,392.20	$49,61,736.68
Large	$7,38,757.91	$49,61,736.68
Total	$82,90,886.79	$49,61,736.68

Name: Measure Table

SUMX → Iterative Function, used to calculate result over iterating into multiple rows

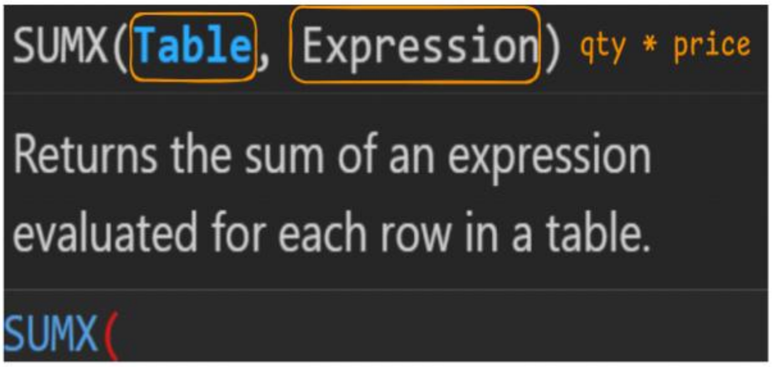


QUANTITYORDERED	PRICEEACH
20	$100
20	$100
34	$100
42	$100
39	$100
41	$100
46	$100
54	$100

qty * price -> for each iteration

SUMX(qty * price)

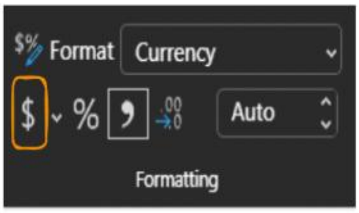
(20*100) + (20 * 100) + (34 * 100) +
(42*100) + (39 * 100) + ...

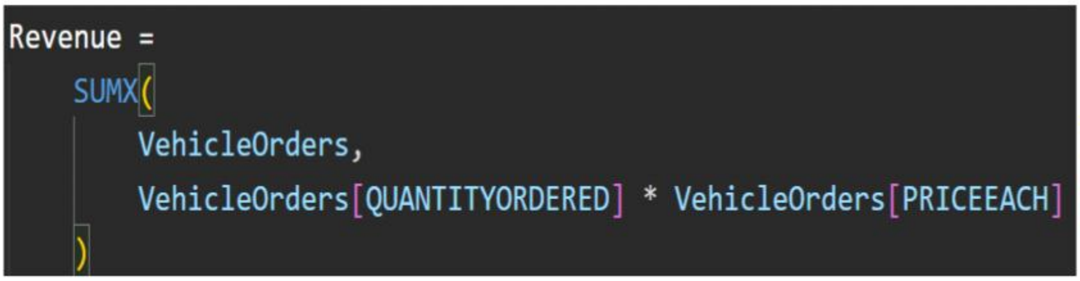


SUMX(Table, Expression) qty * price

Returns the sum of an expression evaluated for each row in a table.

SUMX(

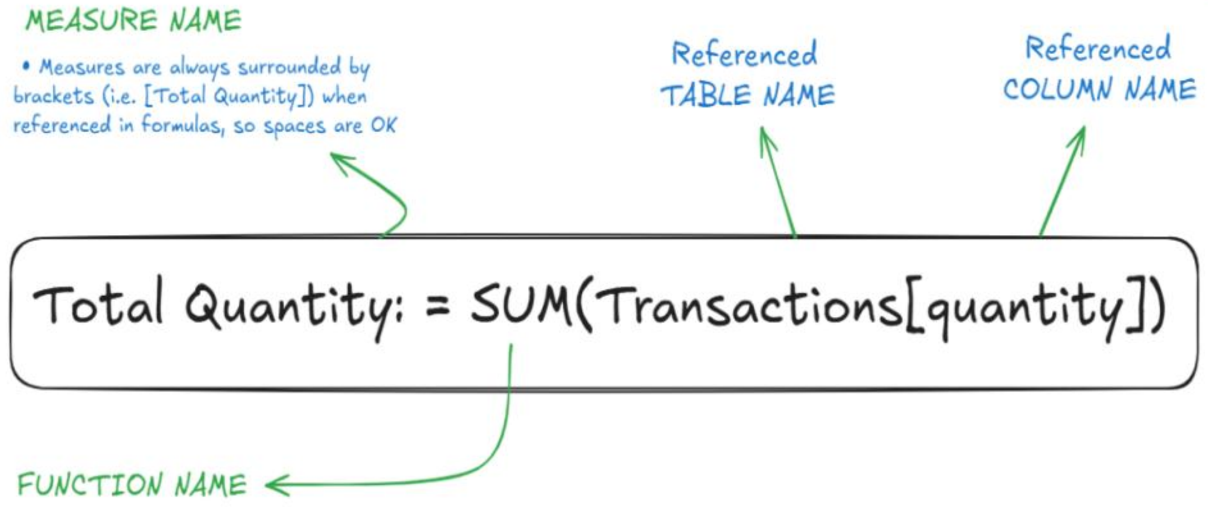




```
Revenue =
    SUMX(
        VehicleOrders,
        VehicleOrders[QUANTITYORDERED] * VehicleOrders[PRICEEACH]
    )
```

```
sales = price * qty
print(sum(sales))
print(sum(price * qty))
```

DAX SYNTAX



MEASURE NAME

- Measures are always surrounded by brackets (i.e. [Total Quantity]) when referenced in formulas, so spaces are OK

Referenced TABLE NAME

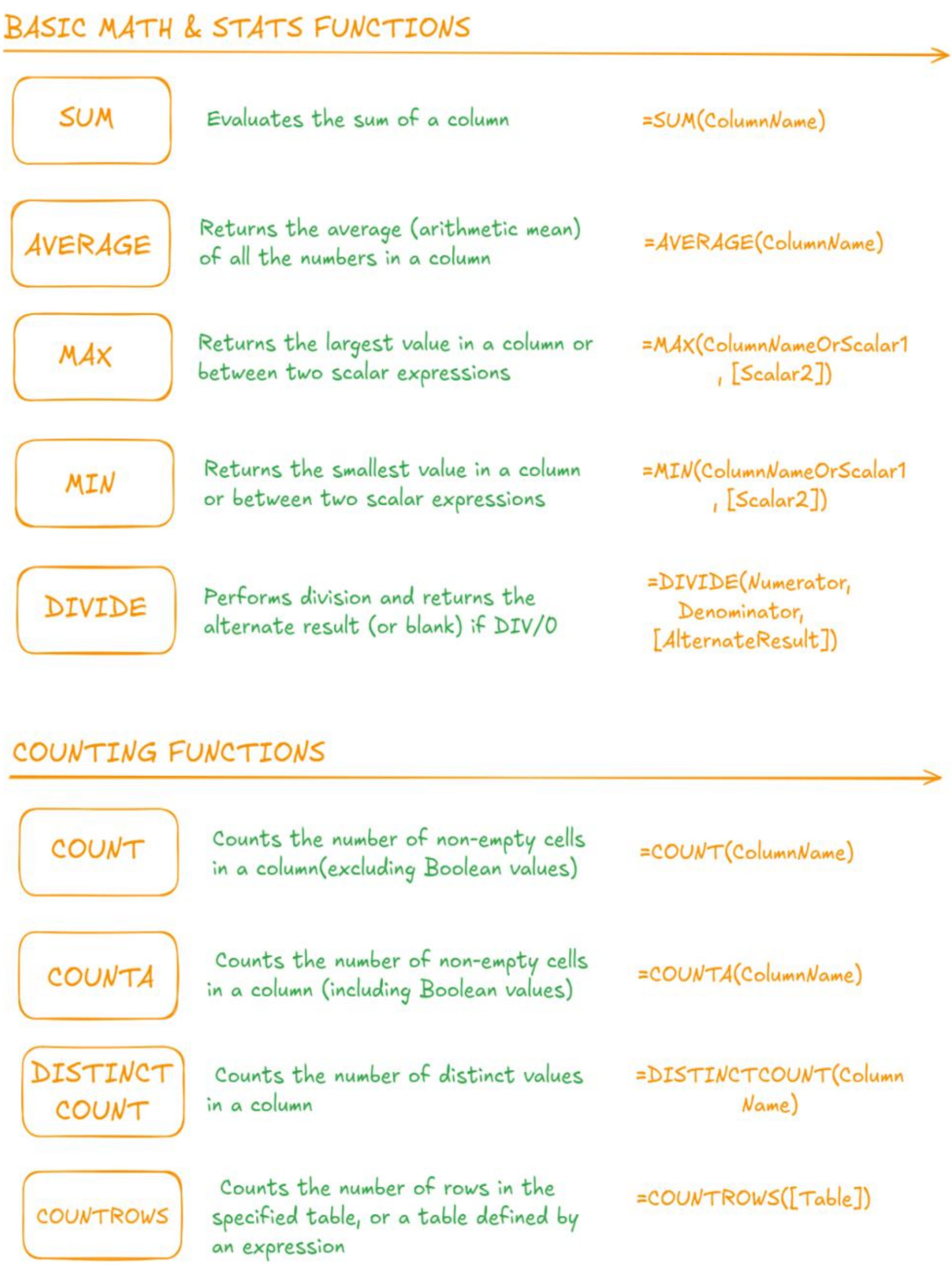
Referenced COLUMN NAME

```
Total Quantity: = SUM(Transactions[quantity])
```

FUNCTION NAME

COMMON FUNCTION CATEGORIES

MATH & STATS Functions	LOGICAL Functions	TEXT Functions	FILTER Functions	TABLE Functions	DATE & TIME Functions	RELATIONSHIP Functions
Functions used for aggregation or iterative, row-level calculations	Functions that use conditional expressions (IF/THEN statements)	Functions used to manipulate text strings or value formats	Functions used to manipulate table and filter contexts	Functions that create or manipulate tables and output tables vs. scalar values	Functions used to manipulate date & time values or handle time intelligence calculations	Functions used to manage &modify table relationships
Common Examples: • SUM • AVERAGE • MAX/MIN • DIVIDE • COUNT/COUNTA • COUNTROWS • DISTINCTCOUNT	Common Examples: • IF • IFERROR • AND • OR • NOT • SWITCH • TRUE • FALSE	Common Examples: • CONCATENATE • COMBINEVALUES • FORMAT • LEFT/MID/RIGHT • UPPER/LOWER • LEN • SEARCH/FIND • REPLACE • SUBSTITUTE • TRIM	Common Examples: • CALCULATE • FILTER • ALL • ALLEXCEPT • ALLSELECTED • KEEPFILTERS • REMOVEFILTERS • SELECTEDVALUE	Common Examples: • SUMMARIZE • ADDCOLUMNS • GENERATESERIES • DISTINCT • VALUES • UNION • INTERSECT • TOPN	Common Examples: • DATE • DATEDIFF • YEARFRAC • YEAR/MONTH • DAY/HOUR • TODAY/NOW • WEEKDAY • WEEKNUM • NETWORKDAYS	Common Examples: • RELATED • RELATEDTABLE • CROSSFILTER • USERELATIONSHIP
Iterator Functions: • SUMX • AVERAGEX • MAXX/MINX • RANKX • COUNTX					Time Intelligence: • DATESYTD • DATESMTD • DATEADD • DATESBETWEEN	



BASIC MATH & STATS FUNCTIONS

SUM	Evaluates the sum of a column	=SUM(ColumnName)
AVERAGE	Returns the average (arithmetic mean) of all the numbers in a column	=AVERAGE(ColumnName)
MAX	Returns the largest value in a column or between two scalar expressions	=MAX(ColumnNameOrScalar1 , [Scalar2])
MIN	Returns the smallest value in a column or between two scalar expressions	=MIN(ColumnNameOrScalar1 , [Scalar2])
DIVIDE	Performs division and returns the alternate result (or blank) if DIV/0	=DIVIDE(Numerator, Denominator, [AlternateResult])

COUNTING FUNCTIONS

COUNT	Counts the number of non-empty cells in a column(excluding Boolean values)	=COUNT(ColumnName)
COUNTA	Counts the number of non-empty cells in a column (including Boolean values)	=COUNTA(ColumnName)
DISTINCT COUNT	Counts the number of distinct values in a column	=DISTINCTCOUNT(Column Name)
COUNTROWS	Counts the number of rows in the specified table, or a table defined by an expression	=COUNTROWS([Table])

BASIC LOGICAL FUNCTIONS

IF	Checks if a given condition is met and returns one value if the condition is TRUE, and another if the condition is FALSE	=IF(LogicalTest, ResultIfTrue, [ResultIfFalse])
IFERROR	Evaluates an expression and returns a specified value if it returns an error, otherwise returns the expression itself	=IFERROR(Value, ValueIfError)
SWITCH	Evaluates an expression against a list of values and returns one of multiple possible expressions	=SWITCH(Expression, Value1, Result1, ..., [Else])
AND	Checks whether both arguments are TRUE to return TRUE, otherwise returns FALSE	=AND(Logical1, Logical2)
OR	Checks whether any argument is TRUE to return TRUE, otherwise returns FALSE	=OR(Logical1, Logical2)

Note: Use the && and || operators to include more than two conditions

TEXT FUNCTIONS

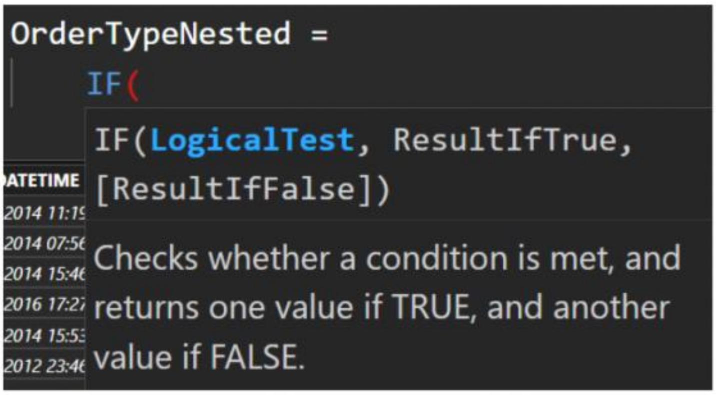
LEN	Returns the number of characters in a string	=LEN(Text)
CONCATENATE	Joins two text strings into one	=CONCATENATE(Text1, Text2)
UPPER /LOWER	Converts a string to upper or lower case	=UPPER/LOWER (Text)
LEFT/ RIGHT/MID	Returns a number of characters from the start/middle/end of a text string	=LEFT/RIGHT(Text, [NumChars]) =MID(Text, StartPosition, NumChars)
SUBSTITUTE	Replaces an instance of existing text with new text in a string	=SUBSTITUTE(Text, OldText, NewText, [InstanceNumber])
SEARCH	Returns the position where a specified string or character is found, reading left to right	=SEARCH(FindText, WithinText, [StartPosition], [NotFoundValue])

BASIC DATE & TIME FUNCTIONS

TODAY/NOW	Returns the current date or exact time	=TODAY/NOW()
DAY/MONTH /YEAR	Returns the day of the month (1-31), month of the year (1-12), or year of a given date	=DAY/MONTH/YEAR(Date)
HOUR/MINUTE /SECOND	Returns the hour (0-23), minute (0-59), or second (0-59) of a given datetime value	=HOUR/MINUTE/ SECOND(Datetime)
WEEKDAY/ WEEKNUM	Returns a weekday number from 1 (Sunday) to 7 (Saturday), or the week # of the year	=WEEKDAY/WEEKNUM(Date, [ReturnType])
EOMONTH	Returns the date of the last day of the month, +/- a specified number of months	=EOMONTH(StartDate, Months)
DATEDIFF	Returns the difference between two dates, based on a given interval (day, hour, year, etc.)	=DATEDIFF(Date1, Date2, Interval)

Order Type:

```
0 - 25 -> Regular Order
26 - 50 -> Medium Order
51 - 75 -> Bulk Order
>75 -> Large Order
```

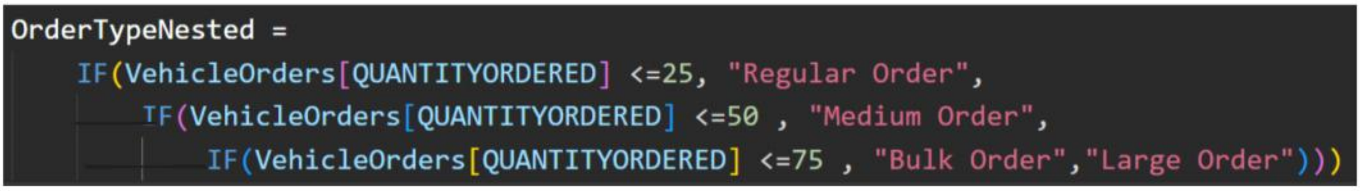


```
OrderTypeNested =
    IF(
```

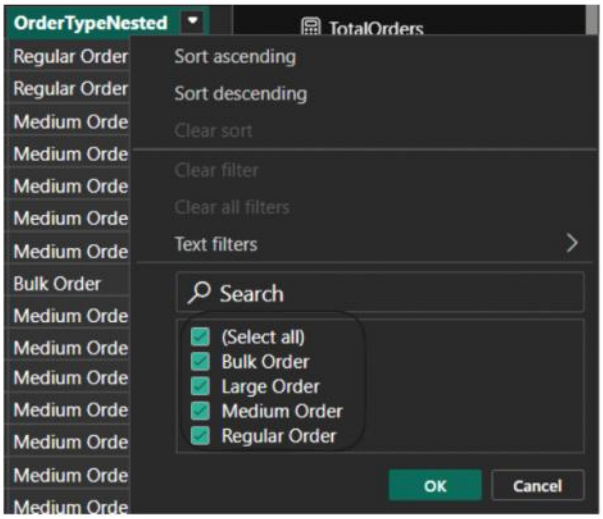
IF(LogicalTest, ResultIfTrue, [ResultIfFalse])

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

C.C



```
OrderTypeNested =
    IF(VehicleOrders[QUANTITYORDERED] <=25, "Regular Order",
        IF(VehicleOrders[QUANTITYORDERED] <=50 , "Medium Order",
            IF(VehicleOrders[QUANTITYORDERED] <=75 , "Bulk Order","Large Order")))
```

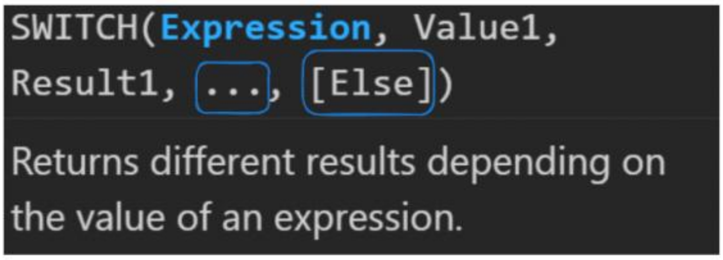


Regular Order
Regular Order
Medium Orde
Medium Orde
Medium Orde
Medium Orde
Medium Orde
Bulk Order
Medium Orde
Medium Orde
Medium Orde
Medium Orde
Medium Orde
Medium Orde
Medium Orde

☑ (Select all)
☑ Bulk Order
☑ Large Order
☑ Medium Order
☑ Regular Order

```
<= 10 : "orderType1"
<= 20 : "orderType2"
<= 30 : "orderType3"
<= 40 : "orderType4"
<= 50 : "orderType5"
<= 60 : "orderType6"
<= 70 : "orderType7"
<= 80 : "orderType8"
<= 90 : "orderType9"
<= 100 : "orderType10"
```

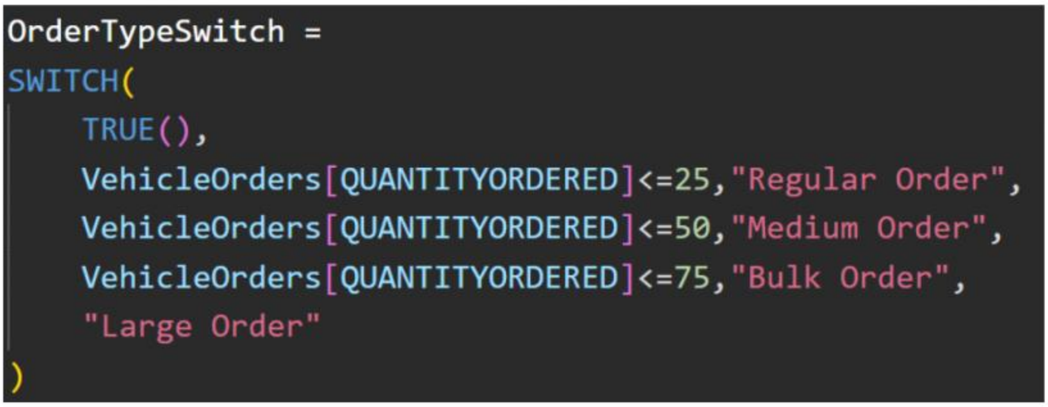
SWITCH → It replaces the nested if function, making the query easy to write and understand.
Also helps to Reduce the filter.



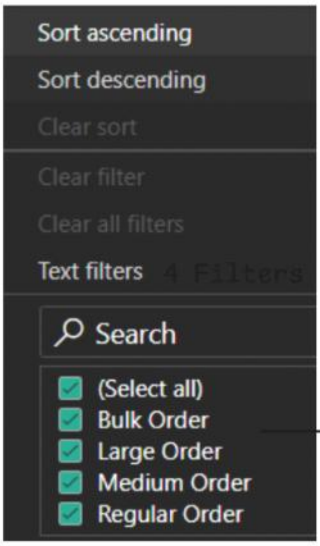
SWITCH(Expression, Value1, Result1, ..., [Else])

Returns different results depending on the value of an expression.

```
while True:
    if(x == 10):
        break
```



```
OrderTypeSwitch =
SWITCH(
    TRUE(),
    VehicleOrders[QUANTITYORDERED]<=25,"Regular Order",
    VehicleOrders[QUANTITYORDERED]<=50,"Medium Order",
    VehicleOrders[QUANTITYORDERED]<=75,"Bulk Order",
    "Large Order"
)
```



☑ (Select all)
☑ Bulk Order
☑ Large Order
☑ Medium Order
☑ Regular Order

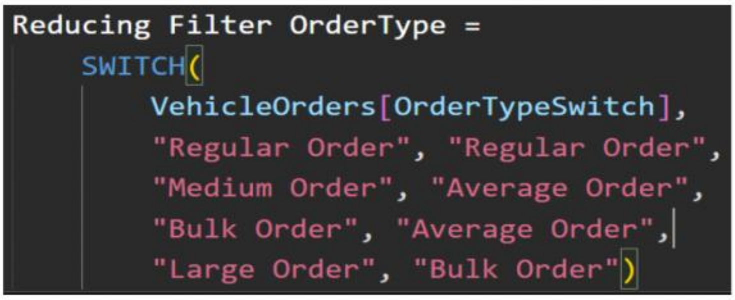
```
pass:   # placeholder

while True:
    if condition:
        pass
    else :
        print("xyz")
```

3 Filters →

```
0-25 : Regular Order
26-75 : Average Order
>75 : Bulk Order
```

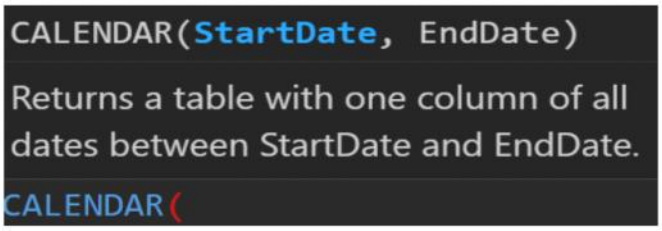
```
Regular Order -> Regular Order
Medium , Bulk -> Average Order
Large Order -> Bulk Order
```



```
Reducing Filter OrderType =
    SWITCH(
        VehicleOrders[OrderTypeSwitch],
        "Regular Order", "Regular Order",
        "Medium Order", "Average Order",
        "Bulk Order", "Average Order",
        "Large Order", "Bulk Order")
```

Dimension Table : Which stores date in consecutive order. [Calendar Table]

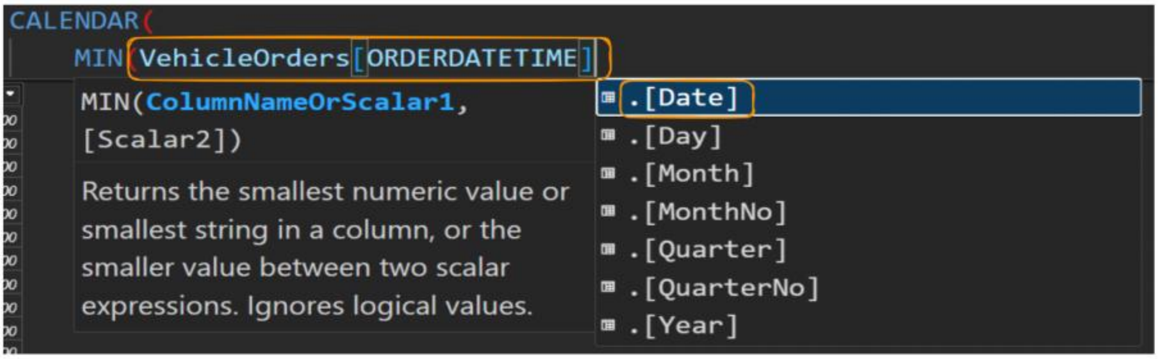
1M
6M
1Yr
3Yr
5Yr
10Yr
Max
Price
PE Ratio
More
Alerts
Volume
45k
40k
35k
30k
25k
20k
15k
10k
5k
0
5 Aug
5 Sep
6 Oct
6 Nov
7 Dec
7 Jan
7 Feb
10 Mar
10 Apr
11 May
11 Jun
12 Jul
160,000
150,000
140,000
130,000
120,000
110,000
100,000
Price on NSE
☑ Price on NSE
□ 50 DMA
□ 200 DMA
Volume



```
CALENDAR(StartDate, EndDate)
```

Returns a table with one column of all dates between StartDate and EndDate.

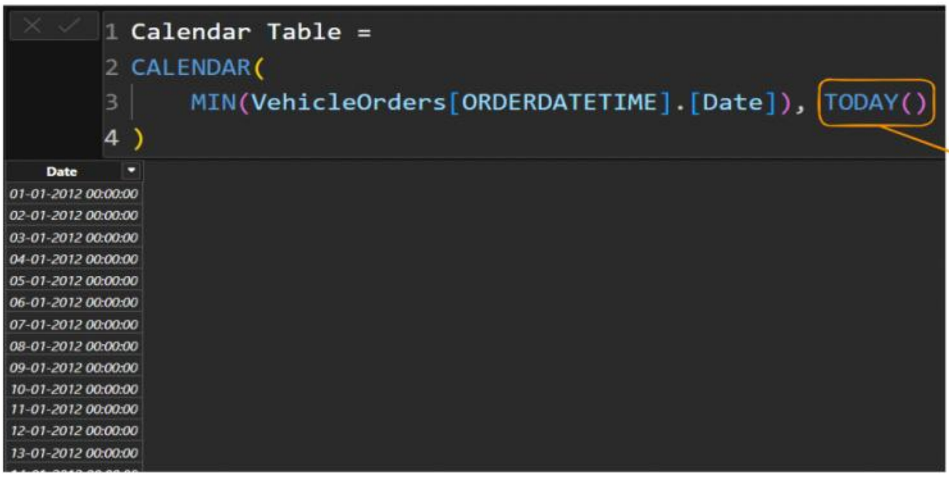
```
CALENDAR(
```



```
CALENDAR(
    MIN(VehicleOrders[ORDERDATETIME]
```

```
MIN(ColumnNameOrScalar1,
[Scalar2])
```

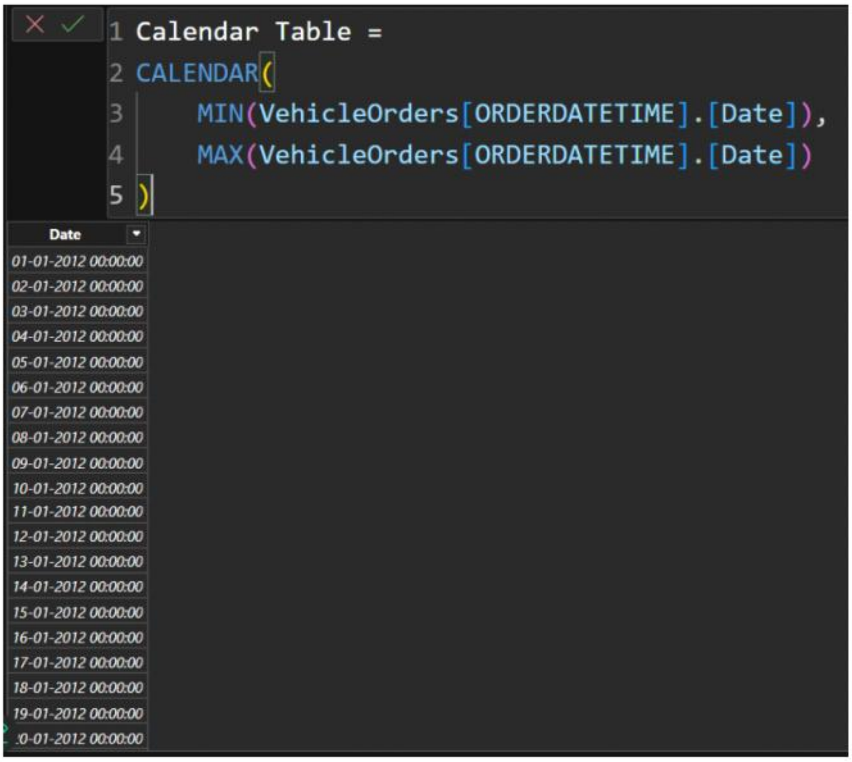
Returns the smallest numeric value or smallest string in a column, or the smaller value between two scalar expressions. Ignores logical values.



```
Calendar Table =
CALENDAR(
    MIN(VehicleOrders[ORDERDATETIME].[Date]), TODAY()
)
```

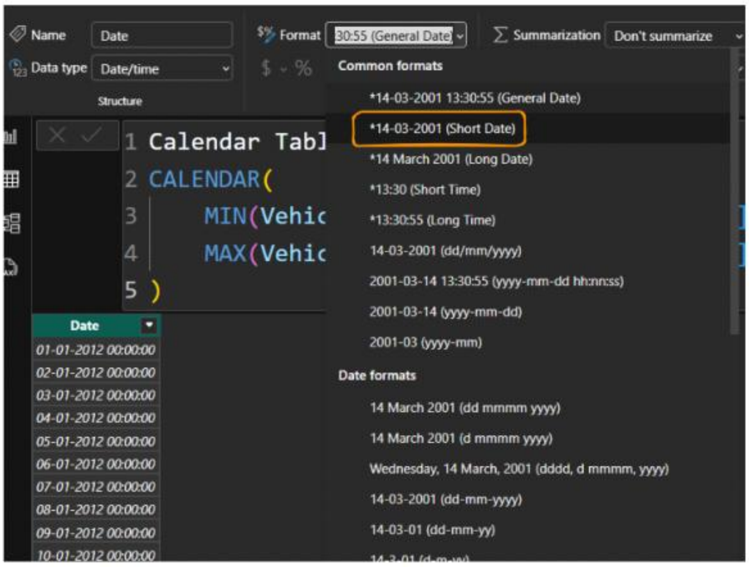
Rolling Callender

Date
01-01-2012 00:00:00
02-01-2012 00:00:00
03-01-2012 00:00:00
04-01-2012 00:00:00
05-01-2012 00:00:00
06-01-2012 00:00:00
07-01-2012 00:00:00
08-01-2012 00:00:00
09-01-2012 00:00:00
10-01-2012 00:00:00
11-01-2012 00:00:00
12-01-2012 00:00:00
13-01-2012 00:00:00



```
Calendar Table =
CALENDAR(
    MIN(VehicleOrders[ORDERDATETIME].[Date]),
    MAX(VehicleOrders[ORDERDATETIME].[Date])
)
```

Date
01-01-2012 00:00:00
02-01-2012 00:00:00
03-01-2012 00:00:00
04-01-2012 00:00:00
05-01-2012 00:00:00
06-01-2012 00:00:00
07-01-2012 00:00:00
08-01-2012 00:00:00
09-01-2012 00:00:00
10-01-2012 00:00:00
11-01-2012 00:00:00
12-01-2012 00:00:00
13-01-2012 00:00:00
14-01-2012 00:00:00
15-01-2012 00:00:00
16-01-2012 00:00:00
17-01-2012 00:00:00
18-01-2012 00:00:00
19-01-2012 00:00:00
20-01-2012 00:00:00



Name: Date

Data type: Date/time

Format: 30:55 (General Date)

Summarization: Don't summarize

Common formats

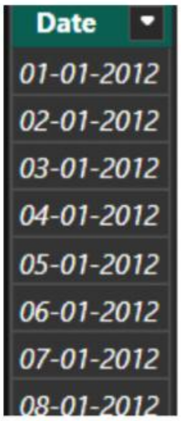
- *14-03-2001 13:30:55 (General Date)
- *14-03-2001 (Short Date)
- *14 March 2001 (Long Date)
- *13:30 (Short Time)
- *13:30:55 (Long Time)
- 14-03-2001 (dd/mm/yyyy)
- 2001-03-14 13:30:55 (yyyy-mm-dd hh:nn:ss)
- 2001-03-14 (yyyy-mm-dd)
- 2001-03 (yyyy-mm)

Date formats

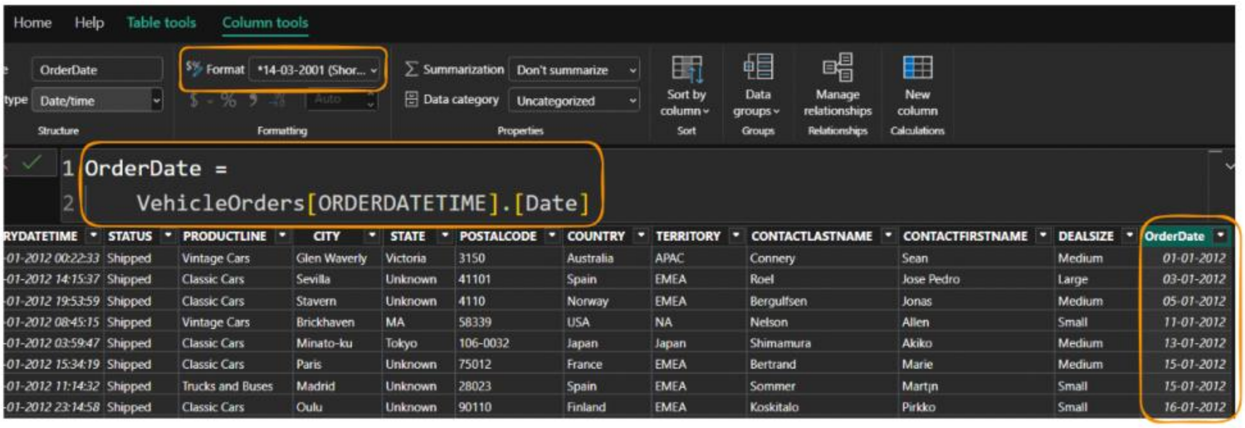
- 14 March 2001 (dd mmmm yyyy)
- 14 March 2001 (d mmmm yyyy)
- Wednesday, 14 March, 2001 (dddd, d mmmm, yyyy)
- 14-03-2001 (dd-mm-yyyy)
- 14-03-01 (dd-mm-yy)
- 14-3-01 (d-m-yy)

```
Calendar Tabl
CALENDAR(
    MIN(Vehic
    MAX(Vehic
)
```

Date
01-01-2012 00:00:00
02-01-2012 00:00:00
03-01-2012 00:00:00
04-01-2012 00:00:00
05-01-2012 00:00:00
06-01-2012 00:00:00
07-01-2012 00:00:00
08-01-2012 00:00:00
09-01-2012 00:00:00
10-01-2012 00:00:00



Date
01-01-2012
02-01-2012
03-01-2012
04-01-2012
05-01-2012
06-01-2012
07-01-2012
08-01-2012



Name: OrderDate

Data type: Date/time

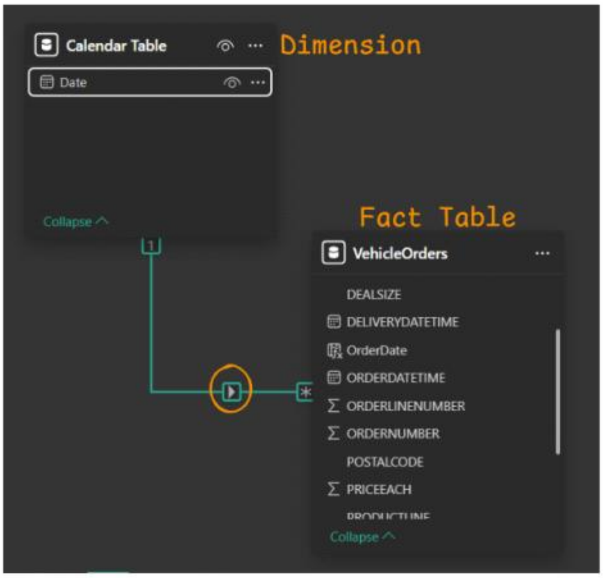
Format: *14-03-2001 (Shor...

Summarization: Don't summarize

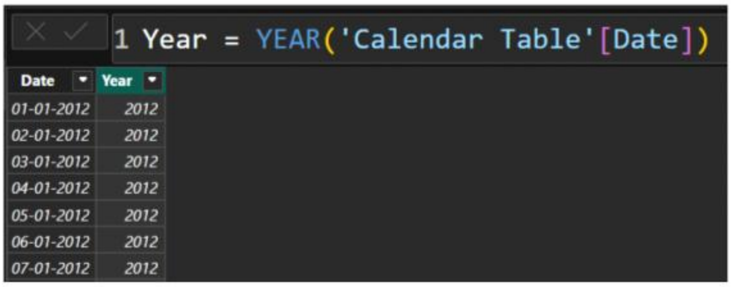
Data category: Uncategorized

```
OrderDate =
    VehicleOrders[ORDERDATETIME].[Date]
```

RYDATETIME	STATUS	PRODUCTLINE	CITY	STATE	POSTALCODE	COUNTRY	TERRITORY	CONTACTLASTNAME	CONTACTFIRSTNAME	DEALSIZE	OrderDate
-01-2012 00:22:33	Shipped	Vintage Cars	Glen Waverly	Victoria	3150	Australia	APAC	Connery	Sean	Medium	01-01-2012
-01-2012 14:15:37	Shipped	Classic Cars	Sevilla	Unknown	41101	Spain	EMEA	Roel	Jose Pedro	Large	03-01-2012
-01-2012 19:53:59	Shipped	Classic Cars	Stavern	Unknown	4110	Norway	EMEA	Bergulfsen	Jonas	Medium	05-01-2012
-01-2012 08:45:15	Shipped	Vintage Cars	Brickhaven	MA	58339	USA	NA	Nelson	Allen	Small	11-01-2012
-01-2012 03:59:47	Shipped	Classic Cars	Minato-ku	Tokyo	106-0032	Japan	Japan	Shimamura	Akiko	Medium	13-01-2012
-01-2012 15:34:19	Shipped	Classic Cars	Paris	Unknown	75012	France	EMEA	Bertrand	Marie	Medium	15-01-2012
-01-2012 11:14:32	Shipped	Trucks and Buses	Madrid	Unknown	28023	Spain	EMEA	Sommer	Martjn	Small	15-01-2012
-01-2012 23:14:58	Shipped	Classic Cars	Oulu	Unknown	90110	Finland	EMEA	Koskitalo	Pirkko	Small	16-01-2012

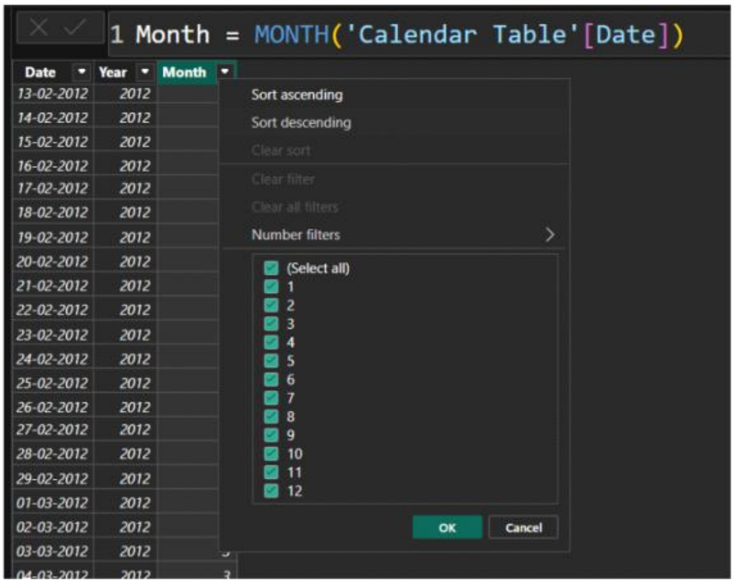


Filter Flow : Single Directional



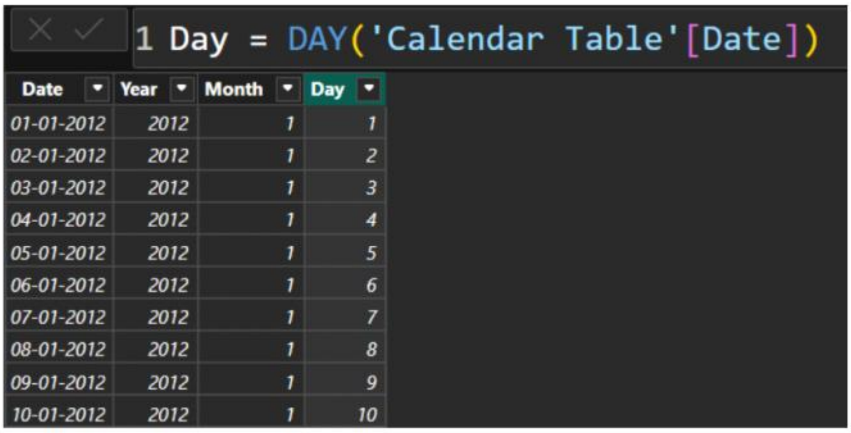
```
Year = YEAR('Calendar Table'[Date])
```

Date	Year
01-01-2012	2012
02-01-2012	2012
03-01-2012	2012
04-01-2012	2012
05-01-2012	2012
06-01-2012	2012
07-01-2012	2012



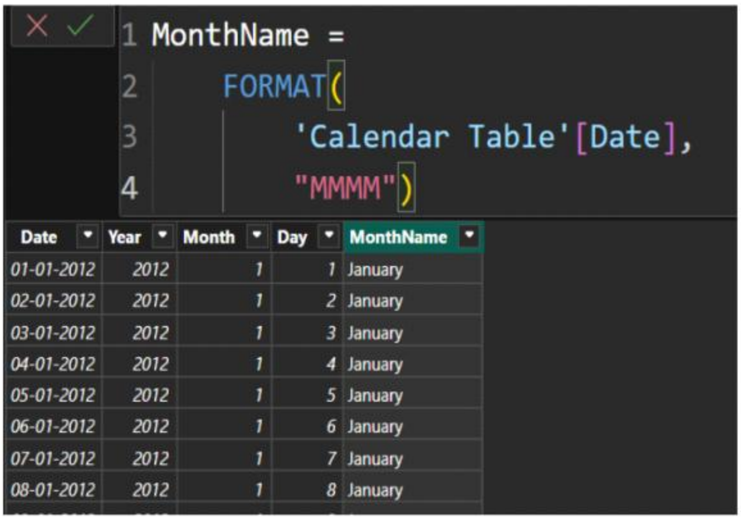
```
Month = MONTH('Calendar Table'[Date])
```

Date	Year	Month
13-02-2012	2012	
14-02-2012	2012	
15-02-2012	2012	
16-02-2012	2012	
17-02-2012	2012	
18-02-2012	2012	
19-02-2012	2012	
20-02-2012	2012	
21-02-2012	2012	
22-02-2012	2012	
23-02-2012	2012	
24-02-2012	2012	
25-02-2012	2012	
26-02-2012	2012	
27-02-2012	2012	
28-02-2012	2012	
29-02-2012	2012	
01-03-2012	2012	
02-03-2012	2012	
03-03-2012	2012	3
04-03-2012	2012	3



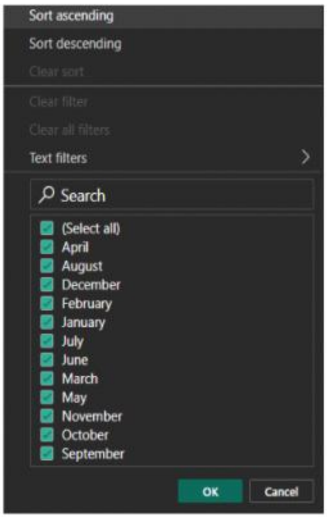
```
Day = DAY('Calendar Table'[Date])
```

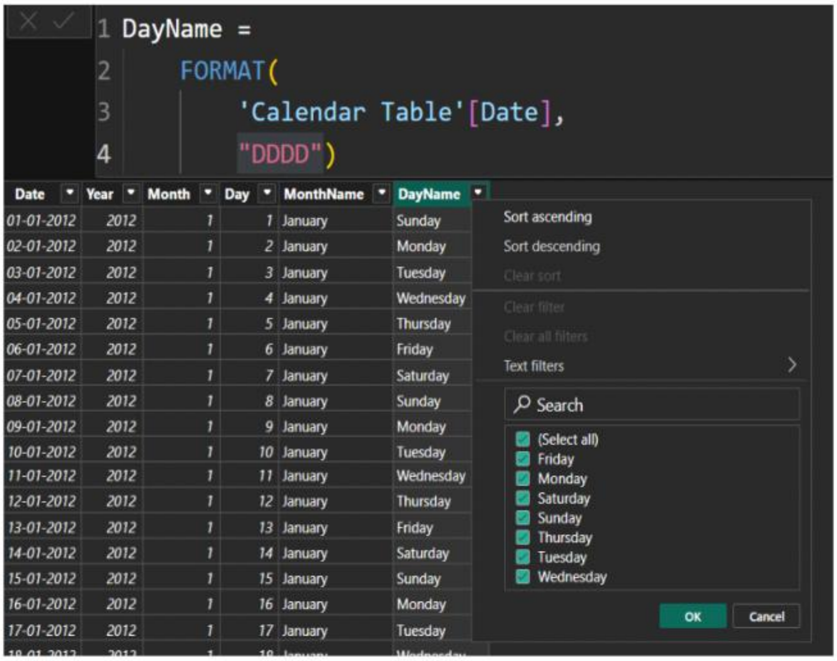
Date	Year	Month	Day
01-01-2012	2012	1	1
02-01-2012	2012	1	2
03-01-2012	2012	1	3
04-01-2012	2012	1	4
05-01-2012	2012	1	5
06-01-2012	2012	1	6
07-01-2012	2012	1	7
08-01-2012	2012	1	8
09-01-2012	2012	1	9
10-01-2012	2012	1	10



```
MonthName =
    FORMAT(
        'Calendar Table'[Date],
        "MMMM")
```

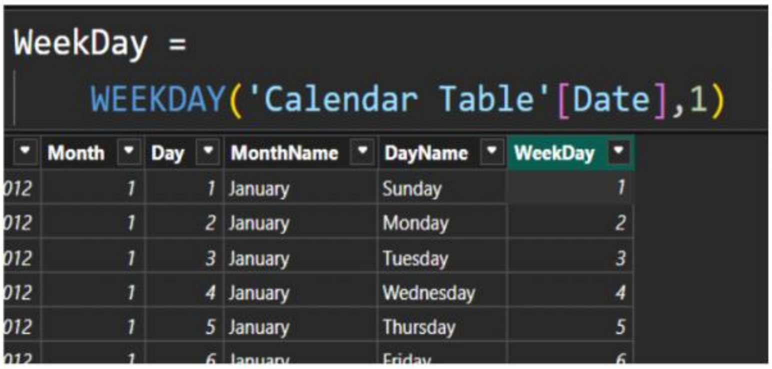
Date	Year	Month	Day	MonthName
01-01-2012	2012	1	1	January
02-01-2012	2012	1	2	January
03-01-2012	2012	1	3	January
04-01-2012	2012	1	4	January
05-01-2012	2012	1	5	January
06-01-2012	2012	1	6	January
07-01-2012	2012	1	7	January
08-01-2012	2012	1	8	January





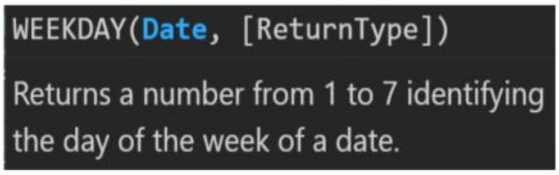
```
DayName =
    FORMAT(
        'Calendar Table'[Date],
        "DDDD")
```

Date	Year	Month	Day	MonthName	DayName
01-01-2012	2012	1	1	January	Sunday
02-01-2012	2012	1	2	January	Monday
03-01-2012	2012	1	3	January	Tuesday
04-01-2012	2012	1	4	January	Wednesday
05-01-2012	2012	1	5	January	Thursday
06-01-2012	2012	1	6	January	Friday
07-01-2012	2012	1	7	January	Saturday
08-01-2012	2012	1	8	January	Sunday
09-01-2012	2012	1	9	January	Monday
10-01-2012	2012	1	10	January	Tuesday
11-01-2012	2012	1	11	January	Wednesday
12-01-2012	2012	1	12	January	Thursday
13-01-2012	2012	1	13	January	Friday
14-01-2012	2012	1	14	January	Saturday
15-01-2012	2012	1	15	January	Sunday
16-01-2012	2012	1	16	January	Monday
17-01-2012	2012	1	17	January	Tuesday



```
WeekDay =
    WEEKDAY('Calendar Table'[Date],1)
```

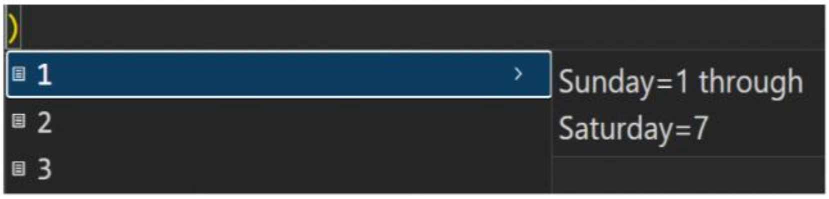
	Month	Day	MonthName	DayName	WeekDay
012	1	1	January	Sunday	1
012	1	2	January	Monday	2
012	1	3	January	Tuesday	3
012	1	4	January	Wednesday	4
012	1	5	January	Thursday	5



```
WEEKDAY(Date, [ReturnType])
```

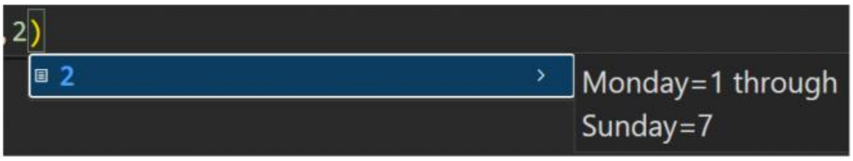
Returns a number from 1 to 7 identifying the day of the week of a date.

```
1 -> Sunday[1] : Saturday[7]
```



```
)
1
2
3
```

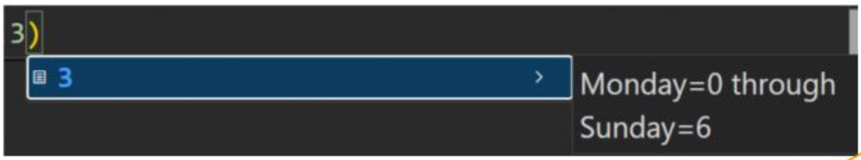
Sunday=1 through Saturday=7



,2)

2

Monday=1 through
Sunday=7

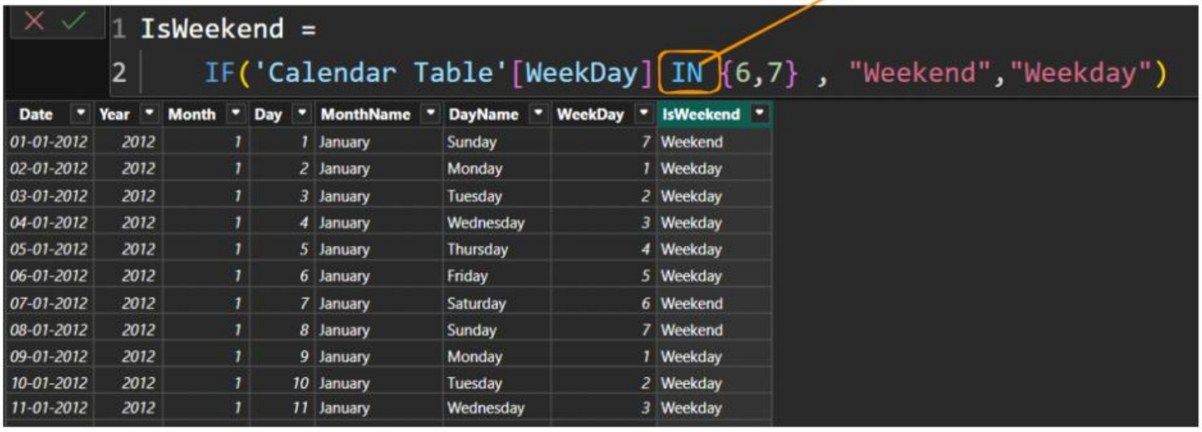


3)

3

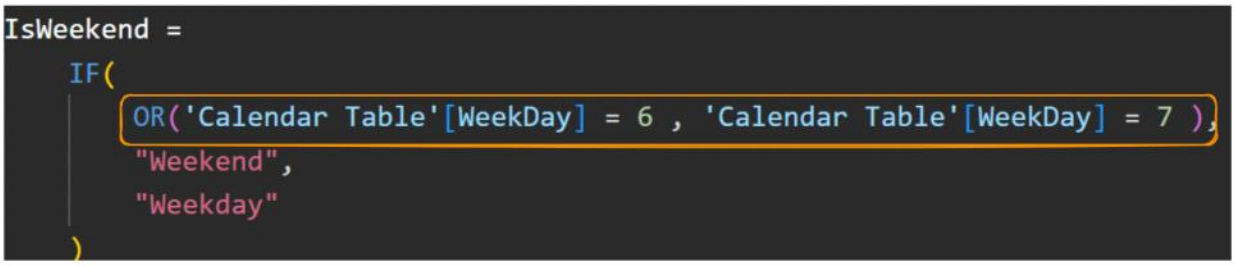
Monday=0 through
Sunday=6

Multiple OR

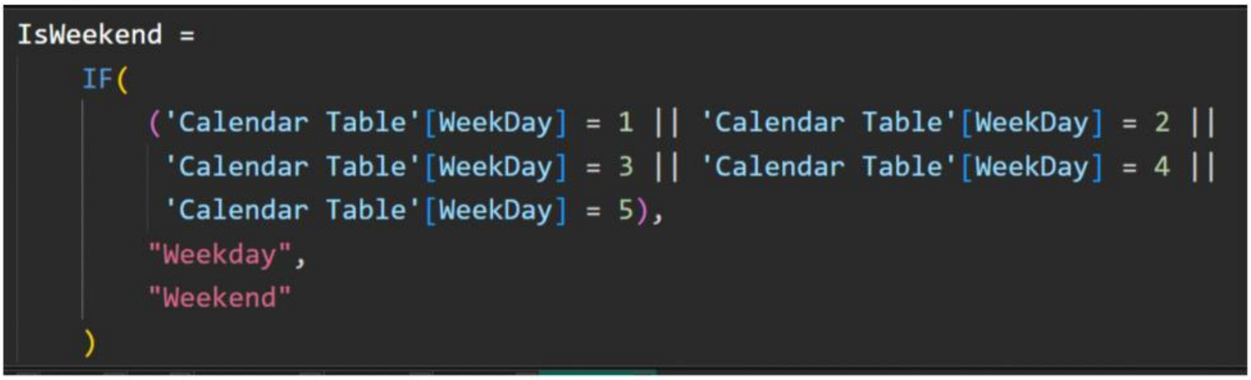


```
IsWeekend =
    IF('Calendar Table'[WeekDay] IN {6,7} , "Weekend","Weekday")
```

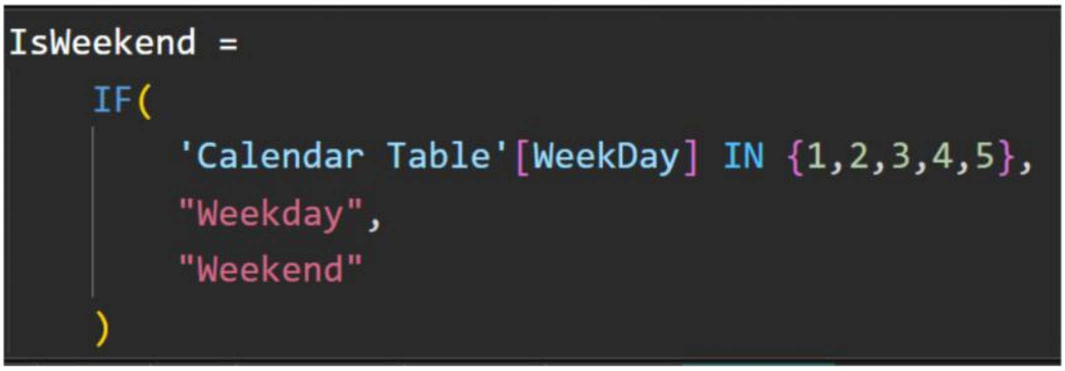
Date	Year	Month	Day	MonthName	DayName	WeekDay	IsWeekend
01-01-2012	2012	1	1	January	Sunday	7	Weekend
02-01-2012	2012	1	2	January	Monday	1	Weekday
03-01-2012	2012	1	3	January	Tuesday	2	Weekday
04-01-2012	2012	1	4	January	Wednesday	3	Weekday
05-01-2012	2012	1	5	January	Thursday	4	Weekday
06-01-2012	2012	1	6	January	Friday	5	Weekday
07-01-2012	2012	1	7	January	Saturday	6	Weekend
08-01-2012	2012	1	8	January	Sunday	7	Weekend
09-01-2012	2012	1	9	January	Monday	1	Weekday
10-01-2012	2012	1	10	January	Tuesday	2	Weekday
11-01-2012	2012	1	11	January	Wednesday	3	Weekday



```
IsWeekend =
    IF(
        OR('Calendar Table'[WeekDay] = 6 , 'Calendar Table'[WeekDay] = 7 ),
        "Weekend",
        "Weekday"
    )
```



```
IsWeekend =
    IF(
        ('Calendar Table'[WeekDay] = 1 || 'Calendar Table'[WeekDay] = 2 ||
         'Calendar Table'[WeekDay] = 3 || 'Calendar Table'[WeekDay] = 4 ||
         'Calendar Table'[WeekDay] = 5),
        "Weekday",
        "Weekend"
    )
```



```
IsWeekend =
    IF(
        'Calendar Table'[WeekDay] IN {1,2,3,4,5},
        "Weekday",
        "Weekend"
    )
```