

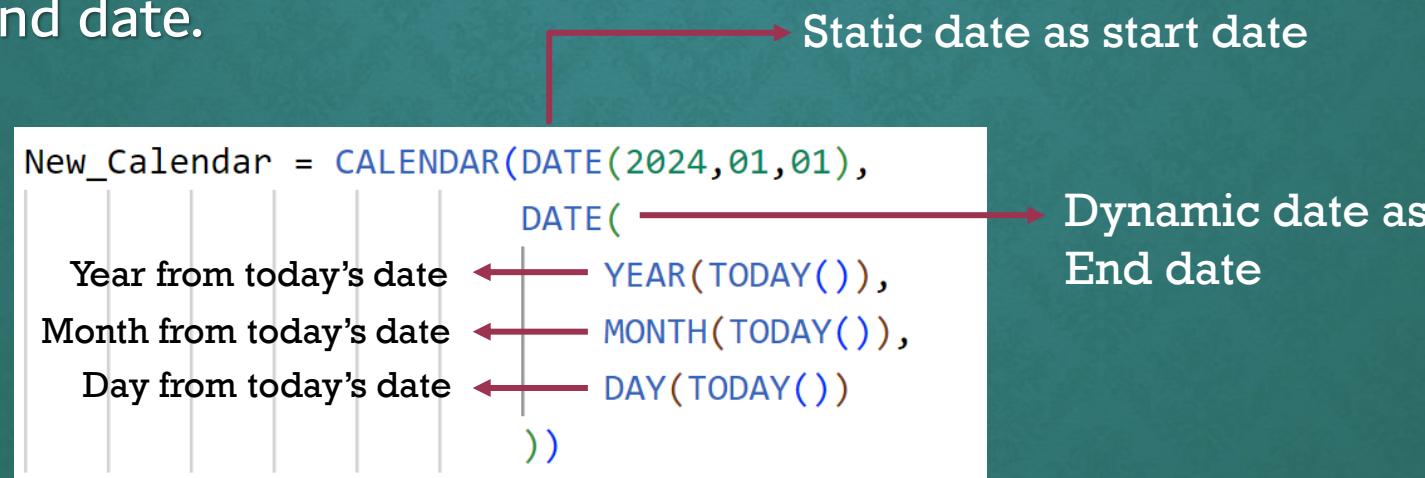
CALENDAR

- This function used to generate a set of dates in a data model.
- It returns a single column of continuous date from the given start date to the end date.

```
New_Calendar = CALENDAR(DATE(2024,01,01),  
                         DATE(  
                             YEAR(TODAY()),  
                             MONTH(TODAY()),  
                             DAY(TODAY())))
```

Static date as start date

Dynamic date as End date



CALENDARAUTO

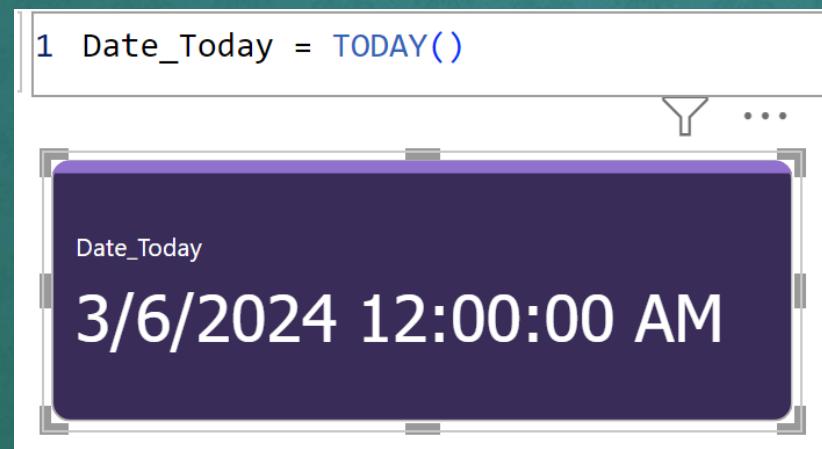
- This function also performs the same task as CALENDAR(), with significant difference of being its date range is calculated automatically based on the dates present in our model.
- Start date: It takes available minimum date as start date if the earliest date is not present in the column
- End date: It takes available maximum date as end date if the latest date is not present in the column

```
1 New_CalendarAuto = CALENDARAUTO()
```



TODAY

- TODAY()
 - Returns the current date.
 - It returns the time value as 12.00 PM for all time



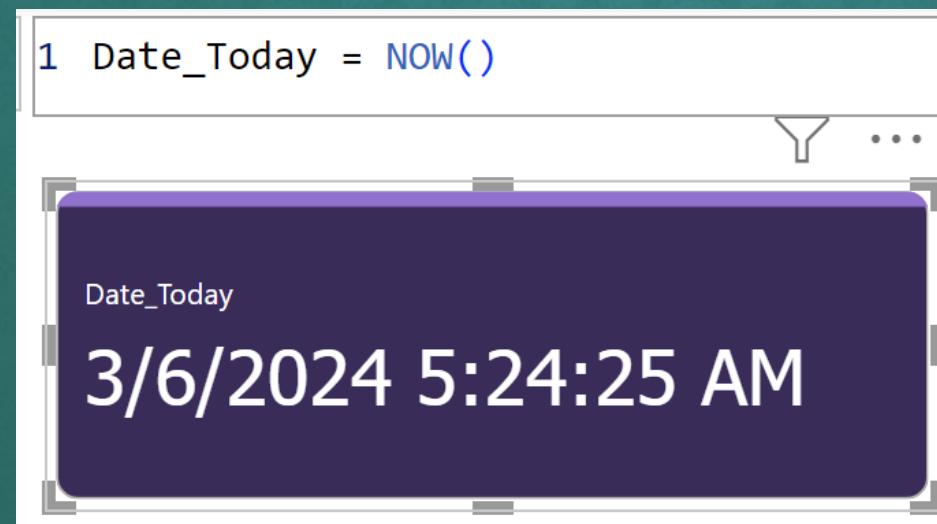
The image shows a screenshot of a Microsoft Power Automate or similar low-code platform. At the top, there is a code editor window containing the following snippet:

```
1 Date_Today = TODAY()
```

Below the code editor is a preview pane with a purple header labeled "Date_Today". The preview pane displays the result of the code execution: "3/6/2024 12:00:00 AM".

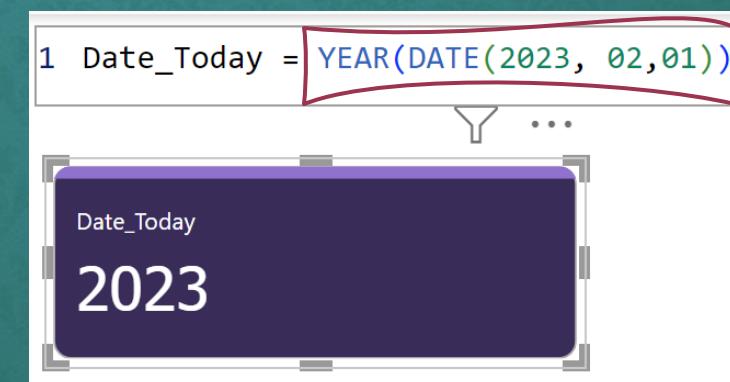
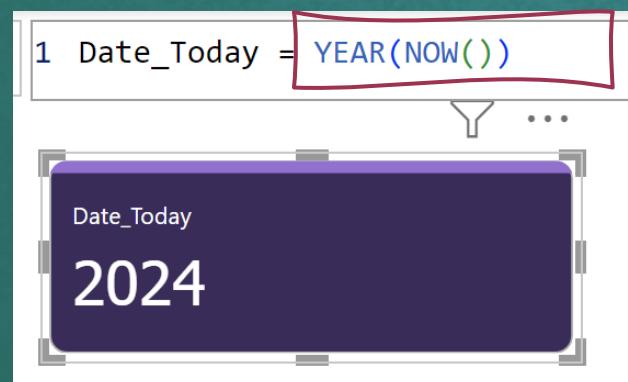
NOW

- NOW()
 - This also returns the current date.
 - Significant difference is it returns the exact time of the day.
 - And it follows UTC date time always.



YEAR

- YEAR() Returns the year of given date.
- This accepts date as a parameter.



YEARFRAC

- YEARFRAC() is used to calculate the portion of the year has been passed between two given dates.
- It accepts 3 parameter such as start date, end date and basis.
- The parameter ‘basis’ indicates type of day count (0,1,2,3,4).
- The third parameter is optional, therefore it takes 0 and follows the 30/360 [US] standard.
- 1 - actual/actual
- 2 - actual/ 360
- 3 - actual/365
- 4 - 30/360 [European]

YEARFRAC

- Below code used basis as 1 and follows the actual standard.
- Fraction = actual days between given days/ actual days in the year
 - = 61/366
 - = 0.166

```
1 year_fraction = YEARFRAC(DATE(2024,01,01), DATE(2024,03,01),1)
```

Date_Today

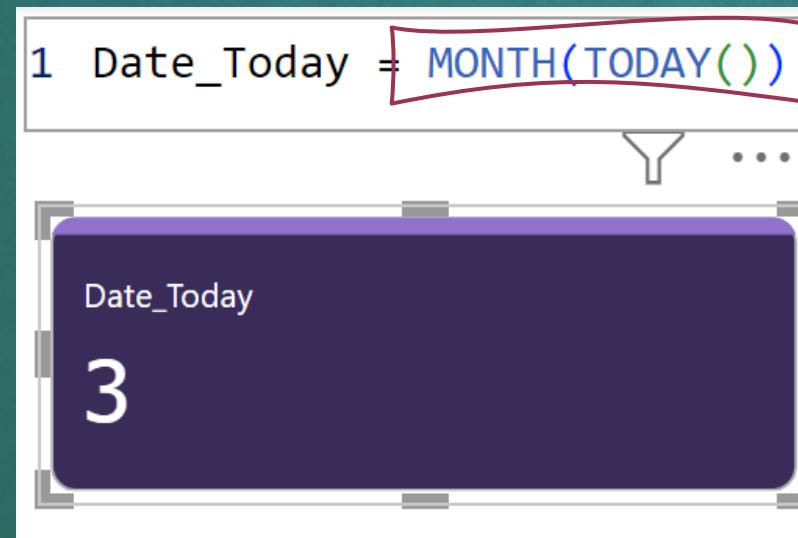
3/6/2024

year_fraction

0.16

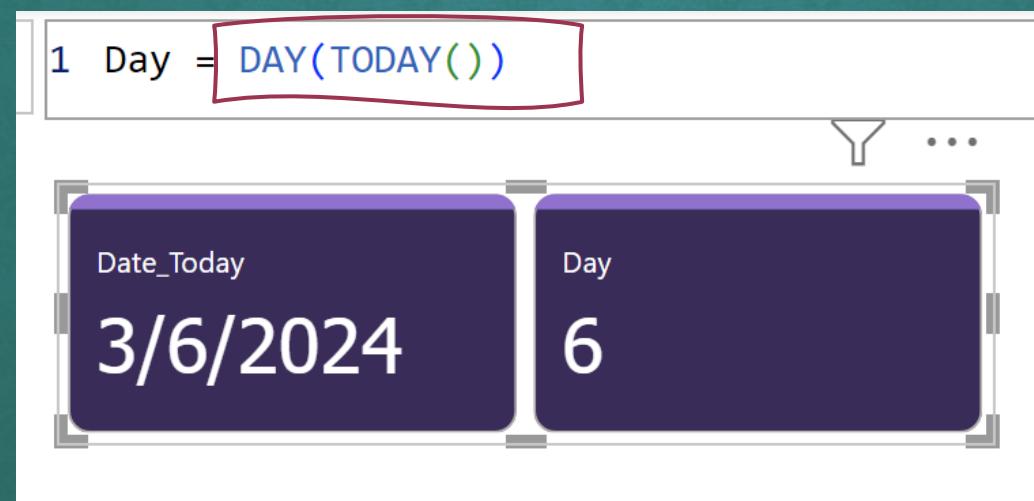
MONTH

- MONTH() Returns the month of given date.
- This requires date as a parameter and returns integer values as month number from 1 to 12.



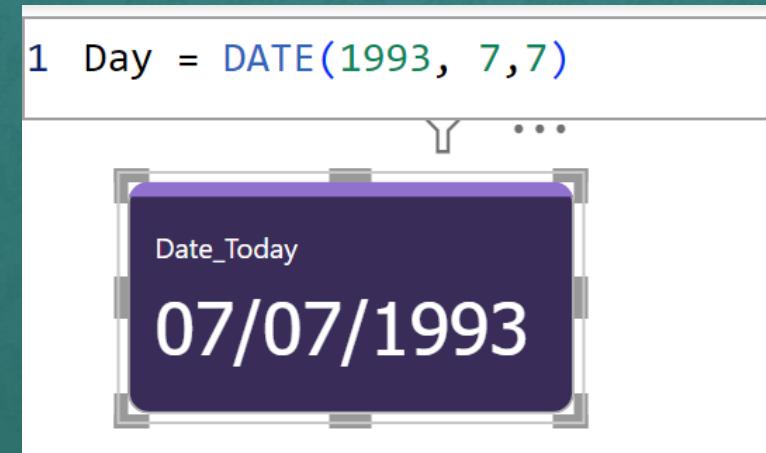
DAY

- DAY() Returns the day of given date.
- This requires date as a parameter and returns an integer, that indicating day of the month.



DATE

- DATE() function converts your integer inputs to corresponding date.
- It accepts 3 integer parameters and there is no optional parameters.

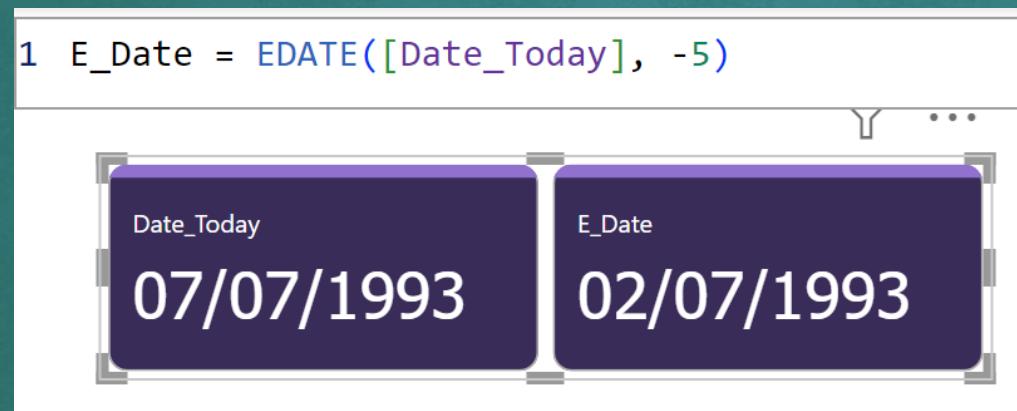


EDATE

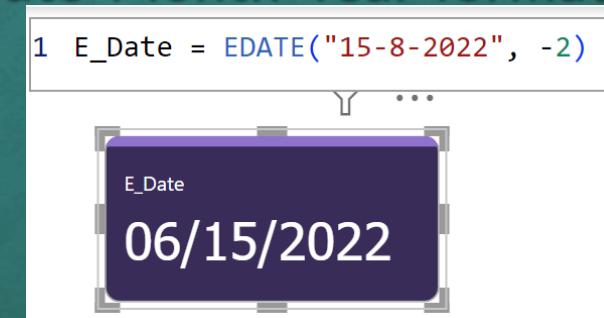
- EDATE() function returns the date, that is indicated number of months before or after the provided start date.
- It accepts date as first parameter and integer as another parameter.
- In the example, used measure as date parameter and it returned the date after 5 months.
- Return value either positive or negative based the order of start and end date values.

EDATE

- In the below example, it returned the date before 5 months.



- Here, I used a static date in the Date-Month-Year format and it returned 2 months earlier date.



DATEDIFF

- DATEDIFF() function calculates the difference between two days.
- Can set the return value in various types such as year, quarter, month, week, day, hour.
- It accepts 3 parameter such as start date, end date and the interval.
- Below example, interval as day and returns the number of days in positive.

```
1 date_diff_day = DATEDIFF(DATE(2024, 1, 11), DATE(2024, 3,10),DAY)
```

date_diff_day	59
date_diff_month	2

DATEDIFF

- Below example, interval as month and returns the number of days in positive.

```
1 date_diff_month = DATEDIFF(DATE(2024, 1, 11), DATE(2024, 3,10), MONTH)
```

date_diff_day	59
date_diff_month	2

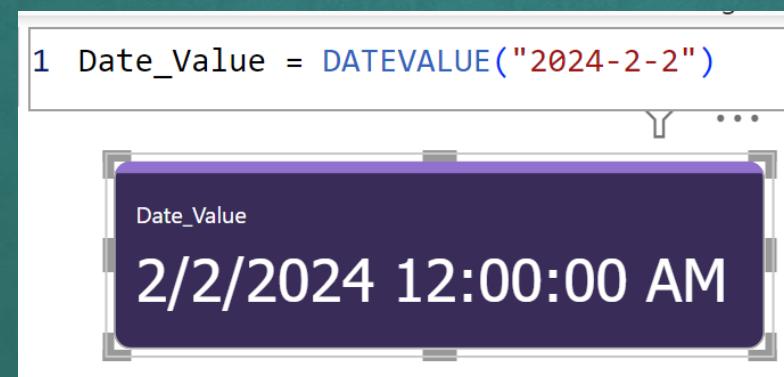
- Calculating age with DATEDIFF()

```
1 Age = DATEDIFF(date(2014, 7,7), TODAY(), YEAR)
```

Age	10
-----	----

DATEVALUE

- DATEVALUE() function used to convert in the form of text into date value.
- It has a single parameter as input and returns a date in datetime type.
- It uses the local date/time settings of the model to determine the date.



The image shows a software interface with a code editor at the top and a preview panel below it. The code editor contains the following line of code:

```
1 Date_Value = DATEVALUE("2024-2-2")
```

The preview panel displays the result of the code execution:

Date_Value
2/2/2024 12:00:00 AM

DATEADD

- DATEADD() function used to add specified number of units (day/month/year/hour/minute/second) to given date and return a new date value.
- It has a three different parameter as input.
- Start date as first parameter, it accepts date column as a parameter
- Number as second parameter – number of units add/remove from the starting date
- Interval as third parameter – type of interval to add (day/month/year/hour/minute/second) .

DATEADD

```
1 Date_add = DATEADD('Date'[Date], 3, MONTH)
```

Start Date
9/15/2017

Date_add
12/15/2017

- In the example, adding months as interval. Selecting start date as input to add 3 months.

DATESBETWEEN

- DATESBETWEEN() function used to find the dates between the given start and end date.
- It returns a table of dates.
- It has a three different parameter as input.
- Date as first parameter, it accepts date column as a parameter
- Start and End date.
- When start and end dates are BLANK, the function takes earliest/latest value in the date column as start/end date.

DATESBETWEEN

- DATESBETWEEN() function returning the dates between first of July to 10th of July as given in the code.

```
1 Dates_Between_column =  
2 DATESBETWEEN('Date'[Date], DATE(2017, 7, 1) , DATE(2017, 7, 10))
```

The screenshot shows the Power BI Data View interface. At the top, there is a code editor window containing the following DAX code:

```
1 Dates_Between_column =  
2 DATESBETWEEN('Date'[Date], DATE(2017, 7, 1) , DATE(2017, 7, 10))
```

A red box highlights the second line of the code, which is the DATESBETWEEN function call. Below the code editor is a table view showing the generated dates:

Date
7/1/2017 12:00:00 AM
7/2/2017 12:00:00 AM
7/3/2017 12:00:00 AM
7/4/2017 12:00:00 AM
7/5/2017 12:00:00 AM
7/6/2017 12:00:00 AM
7/7/2017 12:00:00 AM
7/8/2017 12:00:00 AM
7/9/2017 12:00:00 AM
7/10/2017 12:00:00 AM

DATESBETWEEN

- DATESBETWEEN() function returning the dates between first of July to 5th of July as given in the code.
- Begin date given as BLANK(), so this function takes earliest date as start date from the given column.

```
1 Dates_Between =  
2 DATESBETWEEN('Date'[Date], BLANK(), DATE(2017, 7, 5))
```

Date
7/1/2017 12:00:00 AM
7/2/2017 12:00:00 AM
7/3/2017 12:00:00 AM
7/4/2017 12:00:00 AM
7/5/2017 12:00:00 AM

DATESINPERIOD

- DATESINPERIOD() function used to find the dates between the given start date and continuous for the specified number and interval.
- This is similar to the DATESBETWEEN(), but it calculates the range of dates differently. While DATESBETWEEN() requires specifying both start and end date directly, DATESINPERIOD() takes only start date and calculates range of dates based on specified interval and type.

DATESINPERIOD

- It has a 4 different parameter as input.
- Date as first parameter, it accepts date column as a parameter
- Start date as second parameter
- Number of intervals to add to/subtract from the start date.
- Interval as third parameter – type of interval to add (day/month/year/hour/minute/second) .

DATESINPERIOD

- In the code, Interval as DAY and number of interval set to 10, the function returns dates starting from July 5th and extending 10 days later.

```
1 Dates_Between_column =  
2 DATESINPERIOD('Date'[Date], DATE(2017, 7, 5), 10, DAY)
```

The screenshot shows a Power BI interface with a table named 'Date'. The table has one column labeled 'Date' and contains the following 10 rows:

Date
7/5/2017 12:00:00 AM
7/6/2017 12:00:00 AM
7/7/2017 12:00:00 AM
7/8/2017 12:00:00 AM
7/9/2017 12:00:00 AM
7/10/2017 12:00:00 AM
7/11/2017 12:00:00 AM
7/12/2017 12:00:00 AM
7/13/2017 12:00:00 AM
7/14/2017 12:00:00 AM

FIRSTDATE

- FIRSTDATE() function used to find the date in the given column of date values.
- This returns the table with single column and single row with date value.



The screenshot shows a Power BI interface. At the top, there is a code editor window containing the following DAX code:

```
1 First_Date = FIRSTDATE('Date'[Date])
```

A red curved arrow highlights the `FIRSTDATE` function. Below the code editor is a preview pane. The preview pane has a dark purple header labeled "First_Date". The main body of the preview pane displays the date "7/1/2017".

FIRSTNONBLANK

- This function is useful to find the first non blank value in the given column based on the current context.
- This has two parameters, first one specifies the column from which to retrieve the non-blank value.
- Second parameter is an optional. This is an expression that evaluates the column, such as measure or calculated column.
- Return type contains a table with single column and row with value.

FIRSTNONBLANK

- Below code returns the first non-blank values from the column ‘CustomerKey’ who has their revenue is 10000.
- Here, [Revenue] is a measure.

```
1 First_Non_Blank = FIRSTNONBLANK(Sales[CustomerKey], Sales[Revenue] = 10000)
```

The screenshot shows a Power BI interface. At the top, there is a code editor window containing the DAX formula: `1 First_Non_Blank = FIRSTNONBLANK(Sales[CustomerKey], Sales[Revenue] = 10000)`. Below the code editor is a visual representation of the result, which is a dark purple rounded rectangle with a white border. Inside this rectangle, the text "First_Non_Blank" is displayed above the value "11003". Above the purple box, there is a small icon of a Y-axis with three dots next to it, indicating a list or table structure.

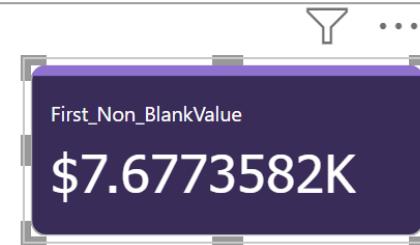
FIRSTNONBLANKVALUE

- This function is also useful to find the first non blank value in the given column based on the expression.
- This has two parameters, first one specifies the column <column> from which to retrieve the non-blank value.
- Second parameter is an expression <expression> that evaluates the column.
- Significant difference is FIRSTNONBLANKVALUE() uses their <columnname> in the filter context of their <expression>

FIRSTNONBLANKVALUE

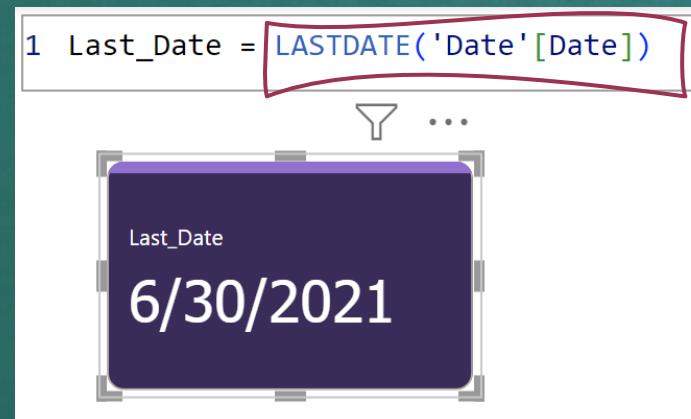
- Below code evaluates the revenue for united states .

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



LASTDATE

- LASTDATE() function used to find the last date in the current context of specified date column.
- It has a single parameter, that is date column.
- It returns a table of single column and row with date value.
- Below example, simply returns the last date from date table



The image shows a screenshot of the Microsoft Power BI desktop application. At the top, there is a code editor window with the following DAX query:

```
1 Last_Date = LASTDATE('Date'[Date])
```

A red box highlights the `LASTDATE` function and its argument. Below the code editor is a small preview pane with a funnel icon and three dots, showing a single row of data. The data is displayed in a dark purple rounded rectangle with a white border, labeled "Last_Date" at the top. The value "6/30/2021" is shown in large white text.

LASTNONDATE

- LASTNONDATE() function returns the last date in the current context of specified column where the expression is non-blank.
- It has two parameter, First is date column from which you want to retrieve the last non-blank value.
- Second parameter is an expression evaluated for each row when the column expression is not blank.
- It returns a table of single column and row with date value.

LASTNONBLANK

- Below example, used LASTNONBLANK() inside the CALCULATE() to determine the context in which the calculation is performed.

The screenshot shows a Power BI interface. On the left, there is a filter pane with a 'Month' dropdown set to '2017 Aug'. In the center, there is a visual card with the title 'Last_Date' and the value '\$100.6396357K'. On the right, the Power BI query editor is open, displaying the following DAX code:

```
1 Last_sale = CALCULATE(SUM(Sales[Sales Amount]), LASTNONBLANK('Date'[Date], [Revenue]))
```

A red box highlights the 'LASTNONBLANK' function in the code.

PREVIOUSYEAR

- PREVIOUSYEAR() function used to return a range of dates from the previous year based on the date given in the current context.
- This function is useful when you want to compare or analyze the data from the previous year to the current date.
- It has 2 parameters, first one is date parameter – optional parameter.
- Second parameter is Year-end-date. Specifies the last date of the year to consider when calculating previous year.

PREVIOUSYEAR

```
1 Previous_year = CALCULATE([Total_Revenue], PREVIOUSYEAR('Date'[Date]))
```

Category	Total_Revenue	Previous_year
Accessories	\$1,096,341.49	\$92,735.3534
Bikes	\$43,484,661.12	\$26,328,476.8155
Clothing	\$1,294,061.73	\$485,587.1546
Components	\$6,003,210.20	\$3,610,092.4719
Total	\$51,878,274.54	\$30,516,891.7954

- In the example, function calculates the total revenue of the last year (2019).
- Total revenue is a measure, it calculates the total revenue of current year (2020)

SAMEPERIODLASTYEAR

- Use this function to retrieve the equivalent period from the previous year.
- It has a single parameter, dates – a column of date value.
- It returns the single column of date values.

The screenshot shows a Power BI interface. At the top, there is a code editor window containing DAX code:

```
1 Total_Revenue_LY = CALCULATE([Total_Revenue], SAMEPERIODLASTYEAR('Date'[Date]))
```

A red box highlights the `SAMEPERIODLASTYEAR` function. Below the code editor is a report view. On the left, there is a filter pane titled "Fiscal Year" with a dropdown set to "FY2019". The main area displays a table with three columns: "Category", "Total_Revenue", and "Total_Revenue_LY". The table data is as follows:

Category	Total_Revenue	Total_Revenue_LY
Accessories	\$138,901.55	\$36,814.8464
Bikes	\$28,544,881.62	\$22,590,983.47
Clothing	\$757,224.19	\$66,327.5346
Components	\$4,629,101.14	\$1,166,765.3153
Total	\$34,070,108.50	\$23,860,891.1663

PARALLELPERIOD

- Use PARALLELPERIOD() to get equivalent period of dates from specified number of intervals back or forward.
- It has 3 parameter. First is date column from where you want to get the date.
- Second is number of intervals to add or subtract from the dates.
- Third is interval type. This can be either year, month or quarter.
- Finally it returns a single column of date values.

PARALLELPERIOD

- Below code, returns the total revenue for corresponding period one year ago, as specified in the context.

```
1 Total_Revenue_Parallel = CALCULATE([Total_Revenue], PARALLELPERIOD('Date'[Date], 1, YEAR))
```

The screenshot shows a Power BI report interface. On the left, there is a dropdown filter labeled "Fiscal Year" set to "FY2020". To the right is a table with three columns: "Category", "Total_Revenue", and "Total_Revenue_Parallel". The table data is as follows:

Category	Total_Revenue	Total_Revenue_Parallel
Accessories	\$1,096,341.49	\$512,387.6532
Bikes	\$43,484,661.12	\$21,306,999.7636
Clothing	\$1,294,061.73	\$558,318.0526
Components	\$6,003,210.20	\$2,091,011.9184
Total	\$51,878,274.54	\$24,468,717.3878

STARTOFTYEAR

- STARTOFTYEAR() returns the first date of the year in the current context for specified column.
- It has two parameter. First parameter contains a column of dates.
- YearEndDate parameter is end date of the year and it's a optional parameter.
- It returns the a table containing single column and row with date value.

STARTOFTYEAR

Code returns the first date (start of the year) of each row.

```
1 start_of_year = STARTOFTYEAR('Date'[Date])
```

start_of_year

1/1/2019...

Date	start_of_year
7/6/2019	1/1/2019 12:00:00 AM
8/13/2019	1/1/2019 12:00:00 AM
9/12/2019	1/1/2019 12:00:00 AM
10/11/2019	1/1/2019 12:00:00 AM
10/30/2019	1/1/2019 12:00:00 AM
11/14/2019	1/1/2019 12:00:00 AM
12/5/2019	1/1/2019 12:00:00 AM
1/24/2020	1/1/2020 12:00:00 AM
2/9/2020	1/1/2020 12:00:00 AM

STARTOFMONT

- STARTOFMONT() returns the first date in the current context for specified column.
- It has single parameter. That's contains a column of dates.
- It returns the a table containing single column and row with date value.

STARTOFMONT

Code returns the first date of a month for each row in a table.

```
1 start_of_month = STARTOFGMONTH('Date' [Date])
```

start_of_month

7/1/2019

Date	start_of_month
7/6/2019	7/1/2019
8/13/2019	8/1/2019
9/12/2019	9/1/2019
10/11/2019	10/1/2019
10/30/2019	10/1/2019
11/14/2019	11/1/2019
12/5/2019	12/1/2019
1/24/2020	1/1/2020
2/9/2020	2/1/2020

STARTOFQUATER

- STARTOFQUATER() returns the first date of the quarter in the current context for specified column.
- It has single parameter. That's contains a column of dates.
- It returns the a table containing single column and row with date value.

STARTOFQUATER

Code returns the first date of the quarter for each row in a table.

The screenshot shows a Power BI interface. On the left, there is a code editor window containing the following DAX code:

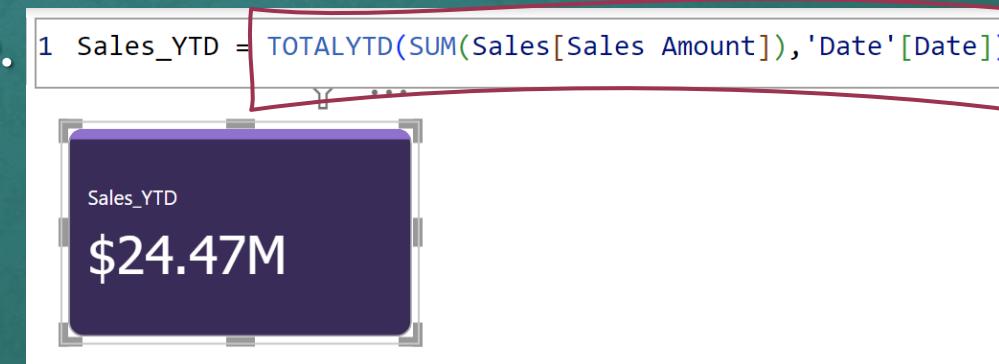
```
1 start_of_quater = STARTOFQUARTER('Date'[Date])
```

A red box highlights the function `STARTOFQUARTER`. A callout arrow points from this highlighted code to a data table on the right. The data table has three columns: `Date`, `start_of_year`, and `start_of_quater`. The `start_of_quater` column contains dates starting from July 1, 2019, and ending at January 1, 2020, with each row colored green. A purple callout box on the left displays the value `7/1/2019...`.

Date	start_of_year	start_of_quater
7/6/2019	1/1/2019 12:00:00 AM	7/1/2019 12:00:00 AM
8/13/2019	1/1/2019 12:00:00 AM	7/1/2019 12:00:00 AM
9/12/2019	1/1/2019 12:00:00 AM	7/1/2019 12:00:00 AM
10/11/2019	1/1/2019 12:00:00 AM	10/1/2019 12:00:00 AM
10/30/2019	1/1/2019 12:00:00 AM	10/1/2019 12:00:00 AM
11/14/2019	1/1/2019 12:00:00 AM	10/1/2019 12:00:00 AM
12/5/2019	1/1/2019 12:00:00 AM	10/1/2019 12:00:00 AM
1/24/2020	1/1/2020 12:00:00 AM	1/1/2020 12:00:00 AM
2/9/2020	1/1/2020 12:00:00 AM	1/1/2020 12:00:00 AM

TOTALYTD

- TOTALYTD() calculates the total year-to-date for specified expression up to the given date.
- It has 2 parameter. First parameter contains the expression for which you want to calculate the year-to-total.
- Second parameter contains the dates.
- It returns a scalar value.



TOTALQTD

- TOTALQTD() calculates the total quarter-to-date for specified expression up to the given date.
- Quarter-to-date – It's a period of beginning of current quarter to current date.
- It has 2 parameter. First parameter contains the expression for which you want to calculate the quarter-to-total.
- Second parameter contains the dates.
- It returns a scalar value.

TOTALQTD

- Below code, calculates the quarter-to-Date sales and returns a

```
1 Sales_QTD = TOTALQTD(SUM(Sales[Sales Amount]), 'Date'[Date])
```

Sales_QTD

\$12.73M

TOTALMTD

- TOTALMTD() calculates the total month-to-date for specified expression up to the given date.
- Month-to-date – It's a period of beginning of current month to current date.
- It has 2 parameter. First parameter contains the expression for which you want to calculate the month-to-total.
- Second parameter contains the dates.
- It returns a scalar value.

TOTALMTD

- Below code, calculates the Month-to-Date sales and returns a value

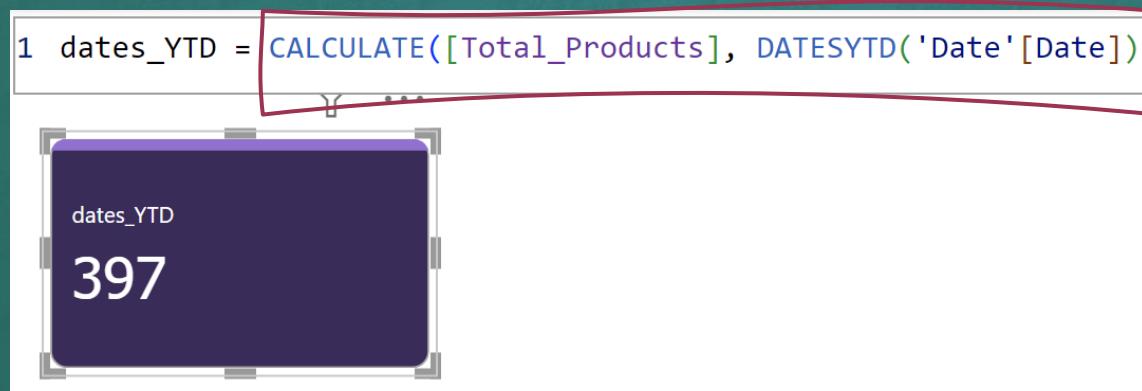
The screenshot shows a Power BI interface. At the top, there is a DAX code editor window containing the following text:

```
1 Sales_MTD = TOTALMTD(SUM(Sales[Sales Amount]), 'Date'[Date])
```

The term `TOTALMTD` is highlighted with a red box. Below the code editor is a small ellipsis icon (...). At the bottom is a dark purple rectangular visual card with a white border. The card displays the text "Sales_MTD" above "\$3.47M".

DATESYTD

- DATESYTD() returns table of dates for year- to-date.
- It has two parameter. First one is a column that contains dates.
- Second parameter is optional, that is a date defines the year-end date.



The screenshot shows a Power BI interface. At the top, there is a code editor window containing the following DAX code:

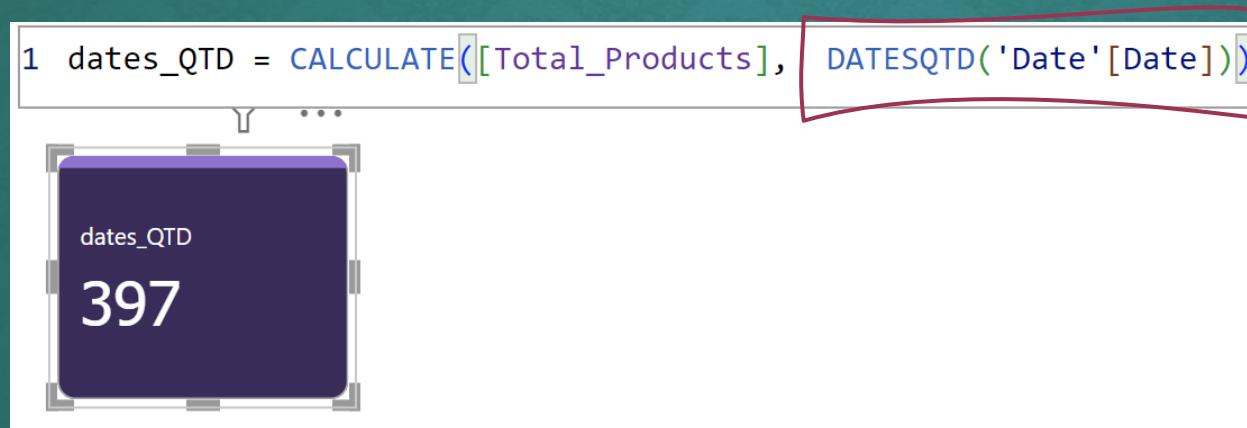
```
1 dates_YTD = CALCULATE([Total_Products], DATESYTD('Date'[Date]))
```

Below the code editor, a data card displays the result of the calculation:

dates_YTD
397

DATESQTD

- DATESQTD() returns table of dates for quarter- to-date.
- It has single parameter. A column that contains dates.



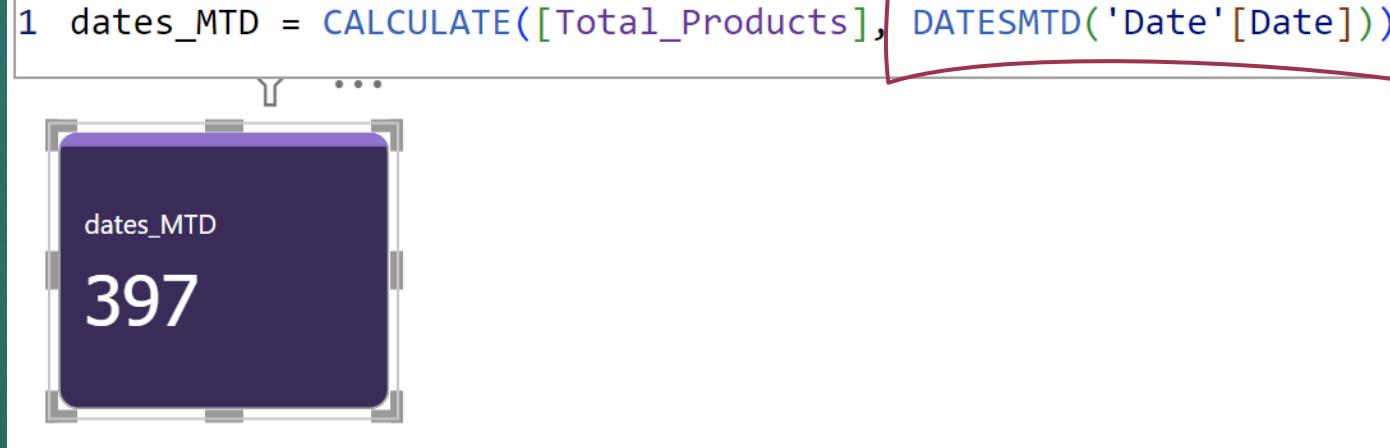
```
1 dates_QTD = CALCULATE([Total_Products], DATESQTD('Date'[Date]))
```

The screenshot shows a Power BI interface. At the top, there is a formula bar with the following DAX code:
1 dates_QTD = CALCULATE([Total_Products], DATESQTD('Date'[Date]))
The term DATESQTD('Date'[Date]) is highlighted with a red rectangular box.
Below the formula bar, there is a visual card with a purple background. The card has the title "dates_QTD" and the value "397".

DATESMTD

- DATESMTD() returns table of dates for month- to-date.
- It has single parameter. A column that contains dates.

```
1 dates_MTD = CALCULATE([Total_Products], DATESMTD('Date'[Date]))
```



The image shows a Power BI interface. At the top, there is a DAX formula in the Power Query editor:

```
1 dates_MTD = CALCULATE([Total_Products], DATESMTD('Date'[Date]))
```

Below the formula, a data card displays the result of the calculated column:

dates_MTD
397

A red callout box points from the text "DATESMTD" in the formula to the corresponding cell in the data card, highlighting the function being discussed.