Table manipulation functions

DAX IN POWER BI

Maarten Van den Broeck
Content Developer at DataCamp

datacamp

Table manipulation functions overview

Previously seen functions

```
DISTINCT(<table> | <table>)
```

Removes duplicate rows from a table or values from a column

```
SELECTCOLUMNS(<table>, <name>, <expression>)
```

Returns the selected columns from another table as a new table

New functions

```
ADDCOLUMNS(<table>, <name>, <expression>)
```

Returns the input table appended with the selected columns from another table

```
SUMMARIZE(<table>,
          <groupBy_columnName>,
          <name>,
          <expression>)
```

Returns a summary table for the requested totals over a set of groups

ADDCOLUMNS()

```
ADDCOLUMNS(<table>, <name>, <expression>)
```

Returns the input table appended with the selected columns from another table

```
ADDCOLUMNS(Fact_table,
           "Profit",
           Revenue - Costs)
```

ADDCOLUMNS()

```
ADDCOLUMNS(<table>, <name>, <expression>)
```

Returns the input table appended with the selected columns from another table

```
ADDCOLUMNS(Fact_table,
           "Profit",
           Revenue - Costs)
```

Revenue	Costs	Profit
100	25	**75**
150	25	**125**

ADDCOLUMNS()

```
ADDCOLUMNS(<table>, <name>, <expression>)
```

Returns the input table appended with the selected columns from another table

```
ADDCOLUMNS(Fact_table,
           "Profit",
           Revenue - Costs)
```

Revenue	Costs	Profit
100	25	**75**
150	25	**125**

```
SELECTCOLUMNS(<table>, <name>, <expression>)
```

Returns the selected columns from another table as a new table

```
SELECTCOLUMNS(Fact_table,
              "Profit",
              Revenue - Costs)
```

Profit
75
125

SUMMARIZE()

```
SUMMARIZE(<table>,
          <groupBy_columnName>,
          <name>,
          <expression>)
```

Returns a summary table for the requested totals over a set of groups

SUMMARIZE()

```
SUMMARIZE(<table>,
          <groupBy_columnName>,
          <name>,
          <expression>)
```

Returns a summary table for the requested totals over a set of groups

Year	Category	Amount
2019	Tickets	50
2019	Postcards	500
2020	Tickets	200
2020	Tickets	400

```
SUMMARIZE(Amounts,
          Amounts[Year],
          Amounts[Category],
          "Total Amount",
          SUM(Amounts[Amount]))
```

SUMMARIZE()

```
SUMMARIZE(<table>,
          <groupBy_columnName>,
          <name>,
          <expression>)
```

Returns a summary table for the requested totals over a set of groups

Year	Category	Amount
2019	Tickets	50
2019	Postcards	500
2020	Tickets	200
2020	Tickets	400

```
SUMMARIZE(Amounts,
          Amounts[Year],
          Amounts[Category],
          "Total Amount",
          SUM(Amounts[Amount]))
```

Year	Category	Total Amount
2019	Tickets	50
2019	Postcards	500
2020	Tickets	600

SUMMARIZE() best practices

- Created columns of `SUMMARIZE()` can give unexpected results based on context

```
SUMMARIZE(Amounts,
          Amounts[Year],
          Amounts[Category]),
          "Total Amount",
          SUM(Amounts[Amount])
```

- Best practice is to wrap `ADDCOLUMNS()` around `SUMMARIZE()` when creating new columns

```
ADDCOLUMNS(
    SUMMARIZE(Amounts,
                Amounts[Year],
                Amounts[Category]),
    "Total Amount",
    SUM(Amounts[Amount])
)
```

Let's practice!

DAX IN POWER BI

Table manipulations using DAX

DAX IN POWER BI

Maarten Van den Broeck
Content Developer at DataCamp

datacamp

Let's practice!

DAX IN POWER BI

Time intelligence functions

DAX IN POWER BI

Maarten Van den Broeck
Content Developer at DataCamp

datacamp

Time intelligence functions

- Manipulate and compare data using time periods

- Compare current period with previous period
- Estimate monthly/quarterly/yearly goals
- Many time intelligence functions exist

Time intelligence functions returning a date

- `NEXTDAY(<dates>)`
  - *Returns the next day*

dates	NEXTDAY
2009-07-07	2009-07-08
2009-07-08	2009-07-09
2009-07-09	2009-07-10

Time intelligence functions returning a date

- `NEXTDAY(<dates>)`
  - *Returns the next day*
- `SAMEPERIODLASTYEAR(<dates>)`
  - *Returns the last year*

dates	NEXTDAY	LASTYEAR
2009-07-07	2009-07-08	2008-07-07
2009-07-08	2009-07-09	2008-07-08
2009-07-09	2009-07-10	2008-07-09

- `DATESBETWEEN(<dates>, <start_date>, <end_date>)`
  - *Returns dates between start and end date*

Time intelligence functions returning a date

- `NEXTDAY(<dates>)`
  - *Returns the next day*
- `SAMEPERIODLASTYEAR(<dates>)`
  - *Returns the last year*

dates	NEXTDAY	LASTYEAR
2009-07-07	2009-07-08	2008-07-07
2009-07-08	2009-07-09	2008-07-08
2009-07-09	2009-07-10	2008-07-09

- `DATESBETWEEN(<dates>, <start_date>, <end_date>)`
  - *Returns dates between start and end date*

dates	DATESBETWEEN
2009-07-07	
2009-07-08	2009-07-08
2009-07-09	2009-07-09
2009-07-10	

Time intelligence functions returning a date

```
Mid Season Sales =
CALCULATE(
    SUM(Fact_Table[Sales]),
    DATESBETWEEN(Dim_Date[Date Key],
                 DATE(2014, 10, 04),
                 DATE(2014, 10, 26)
                 )
)
```

Time intelligence functions returning a value

```
TOTALYTD(<expression>, <dates> [,<filter>])
TOTALQTD(<expression>, <dates> [,<filter>])
TOTALMTD(<expression>, <dates> [,<filter>])
```

Returns the year, quarter, or month to date value of the expression.

```
Sum_YTD =
TOTALYTD(
    SUM(Fact_Table[Value]),
    Dim_Date[Date Key]
)
```

Time intelligence functions returning a value

```
TOTALYTD(<expression>, <dates> [,<filter>])
TOTALQTD(<expression>, <dates> [,<filter>])
TOTALMTD(<expression>, <dates> [,<filter>])
```

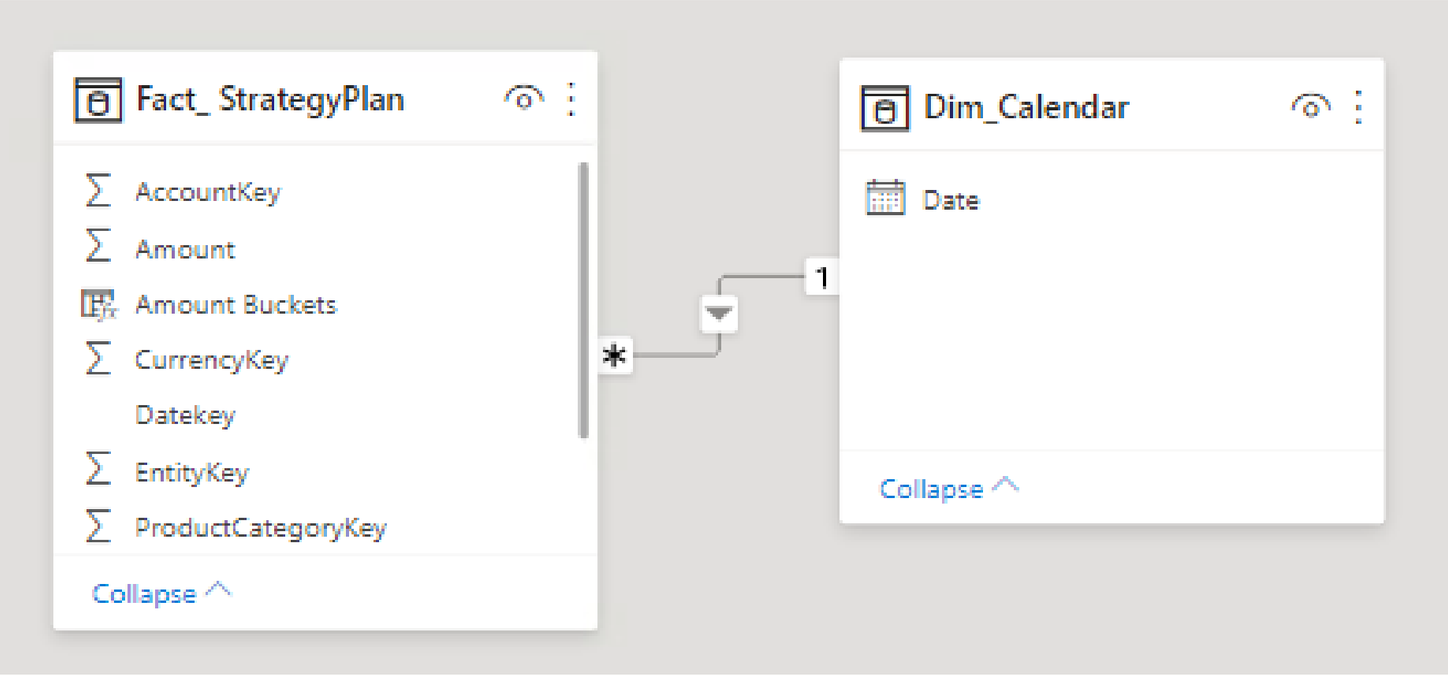
Returns the year, quarter, or month to date value of the expression.

```
Sum_YTD =
TOTALYTD(
    SUM(Fact_Table[Orders]),
    Dim_Date[Date Key]
)
```

Year	Month	Value	Sum_YTD
2021	Jan	6,532	6,532
2021	Feb	4,263	10,795
2021	Mar	1,256	12,051
Total		**12,051**	**12,051**

Best practices for time intelligence functions

- Use a separate date dimension table



A date column in the fact table could contain missing dates!

Let's practice!

DAX IN POWER BI

Time intelligence in Power BI

DAX IN POWER BI

Maarten Van den Broeck
Content Developer at DataCamp

datacamp

Let's practice!

DAX IN POWER BI

Congratulations!

DAX IN POWER BI

Maarten Van den Broeck
Content Developer at DataCamp

datacamp

DAX opens doors

- DAX formulas are used in:
  - Measures
  - Calculated columns
  - Calculated tables
  - Row-level security

Course overview

Chapter 1

- Row vs. query vs. filter context
- Implicit vs. explicit measures
- Use of variables with `VAR`

Chapter 2

- Filtering functions: `CALCULATE()`, `FILTER()`
- Counting functions: `COUNT()`, `DISTINCTCOUNT()`, `COUNTROWS()`

Chapter 3

- Logical functions: `IF()`, `SWITCH()`
- Row-level security: `USERPRINCIPALNAME()`

Chapter 4

- Table manipulation functions: `ADDCOLUMNS()`, `SUMMARIZE()`
- Time intelligence functions: `SAMEPERIODLASTYEAR()`, `TOTALYTD()`

Use it or lose it

Download our cheat sheet on the course landing page!

datacamp

Glossary of DAX functions and operators

> Math & statistical functions

- **SUM(<column>)**
  Adds all the numbers in a column.
- **AVERAGE(<column>)**
  Returns the average (arithmetic mean) of all the numbers in a column.
- **SUMX(<table>, <expression>)**
  Returns the sum of an expression evaluated for each row in a table.
- **COUNTX(<table>, <expression>)**
  Counts the number of rows from an expression that evaluates to a non-blank value.
- **AVERAGEX(<table>, <expression>)**
  Calculates the average (arithmetic mean) of a set of expressions evaluated over a table.
- **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.
- **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.

> 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 multiple possible result

> Date & time functions

- **CALENDAR(<start_date>, <end_date>)**
  Returns a table with a single column named "Date" that contains a contiguous set of dates.

> Time intelligence functions

- **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.

> Relationship functions

- **CROSSFILTER()**
  Specifies the cross-filtering direction to be used in a calculation.
- **RELATED()**
  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.

> Text functions

- **SUBSTITUTE(<text>, <old_text>, <new_text>, <instance_num>)**
  Replaces existing text with new text in a string.

> Information functions

- **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.

> Other functions

- **BLANK()**
  Returns a blank.

> 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"}

Can't find the function you're looking for?
Take a look at the Microsoft documentation

datacamp

See you soon!

DAX IN POWER BI