Dax Expressions In Power BI

Calculated Columns

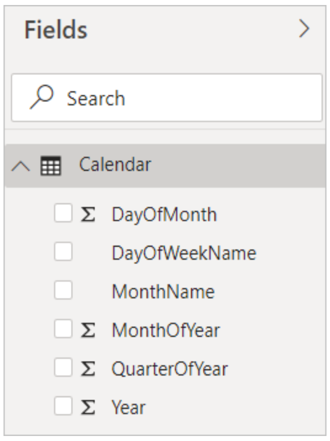
- Sometimes the data you're analysing doesn't contain a particular field that you need to get your desired results. **Calculated columns** are useful for this situation. Calculated columns use Data Analysis Expressions (DAX) formulas to define a column's values.
- This tool is useful for anything from putting together text values from a couple of different columns to calculating a numeric value from other values.
- For example, let's say your data has **City** and **State** fields, but you want a single **Location** field that has both, like "Miami, FL".
- Calculated columns are similar to measures in that both are based on DAX formulas, but they differ in how they're used. You often use measures in a visualization's **Values** area, to calculate results based on other fields. You use calculated columns as new **Fields** in the rows, axes, legends, and group areas of visualizations.

Create calculated columns in Power BI

With calculated columns, you can add new data to a table already in your model. But instead of querying and loading values into your new column from a data source, you create a Data Analysis Expressions (DAX) formula that defines the column's values. In Power BI Desktop, calculated columns are created by using the new column feature in **Report** view, **Data** view, or **Model** view.

Unlike custom columns that are created as part of a query by using **Add Custom Column** in Power Query Editor, calculated columns that are created in **Report** view, **Data** view, or **Model** view are based on data you've already loaded into the model. For example, you might choose to concatenate values from two different columns in two different but related tables, do addition, or extract substrings.

Calculated columns you create appear in the **Fields** list just like any other field, but they'll have a special icon showing its values are the result of a formula. You can name your columns whatever you want and add them to a report visualization just like other fields.

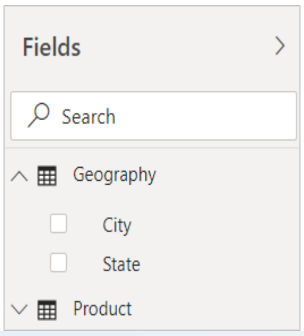


Calculated columns calculate results by using DAX, a formula language meant to work with relational data like in Power BI Desktop. DAX includes a library of over 200 functions, operators, and constructs. It provides immense flexibility in creating formulas to calculate results for just about any data analysis need. To learn more about DAX, see Learn DAX basics in Power BI Desktop.

DAX formulas are similar to Excel formulas. In fact, DAX has many of the same functions as Excel. DAX functions, however, are meant to work over data interactively sliced or filtered in a report, like in Power BI Desktop. In Excel, you can have a different formula for each row in a table. In Power BI, when you create a DAX formula for a new column, it will calculate a result for every row in the table. Column values are recalculated as necessary, like when the underlying data is refreshed, and values have changed.

Let's look at an example.

Jeff is a shipping manager at Contoso and wants to create a report showing the number of shipments to different cities. Jeff has a **Geography** table with separate fields for city and state. But, Jeff wants their reports to show the city and state values as a single value on the same row. Right now, Jeff's **Geography** table doesn't have the wanted field.



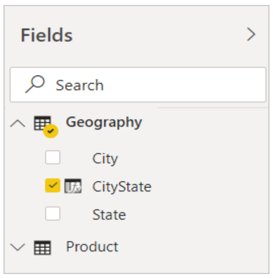
But with a calculated column, Jeff can put together the cities from the **City** column with the states from the **State** column.

Jeff right-clicks on the **Geography** table and then selects **New Column**. Jeff then enters the following DAX formula into the formula bar:

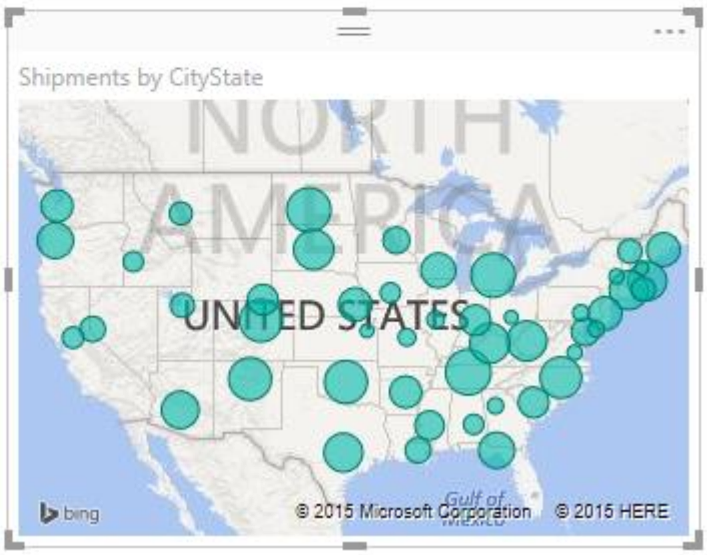
CityState = [City] & "," & [State]

This formula creates a new column named **CityState**. For each row in the **Geography** table, it takes values from the **City** column, adds a comma and a space, and then concatenates values from the **State** column.

Now Jeff has the wanted field.

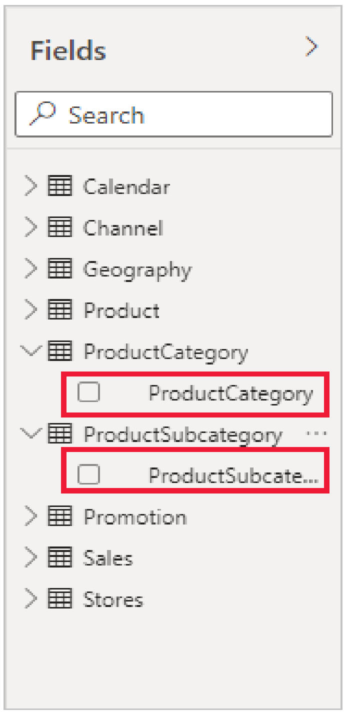


Jeff can now add it to the report canvas along with the number of shipments. With minimal effort, Jeff now has a **CityState** field that can be added to just about any type of visualization. When Jeff creates a new map, Power BI Desktop already knows how to read the city and state values in the new column.

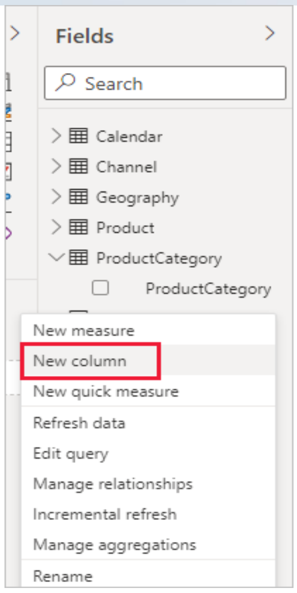


Create a calculated column with values from related tables

In your Sales Report, you want to display product categories and subcategories as single values, like "Cell phones – Accessories", "Cell phones – Smartphones & PDAs", and so on. There's no field in the **Fields** list that gives you that data, but there's a **ProductCategory** field and a **ProductSubcategory** field, each in its own table. You can create a calculated column that combines values from these two columns. DAX formulas can use the full power of the model you already have, including relationships between different tables that already exist.



1.To create your new column in the **ProductSubcategory** table, right-click or select the ellipsis **...** next to **ProductSubcategory** in the **Fields** pane, and choose **New column** from the menu.

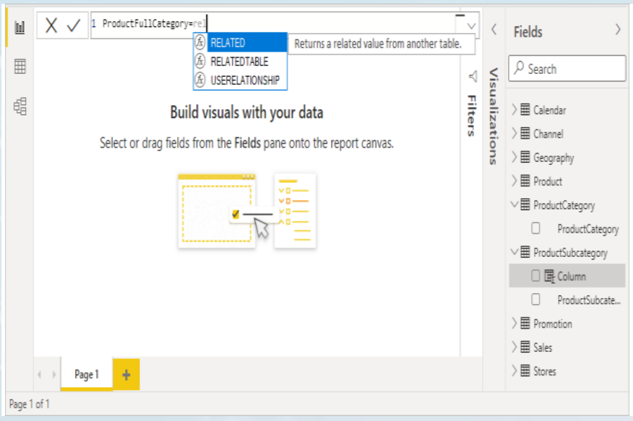


When you choose **New column**, the **Formula bar** appears along the top of the Report canvas, ready for you to name your column and enter a DAX formula.

2.By default, a new calculated column is named **Column**. If you don't rename it, new columns will be named **Column 2**, **Column 3**, and so on. You want your column to be more identifiable, so while the **Column** name is already highlighted in the formula bar, rename it by typing **ProductFullCategory**, and then type an equals (**=**) sign.

3.You want the values in your new column to start with the name in the **ProductCategory** field. Because this column is in a different but related table, you can use the RELATED function to help you get it.

After the equals sign, type **r**. A dropdown suggestion list shows all of the DAX functions beginning with the letter R. Selecting each function shows a description of its effect. As you type, the suggestion list scales closer to the function you need. Select **RELATED**, and then press **Enter**.



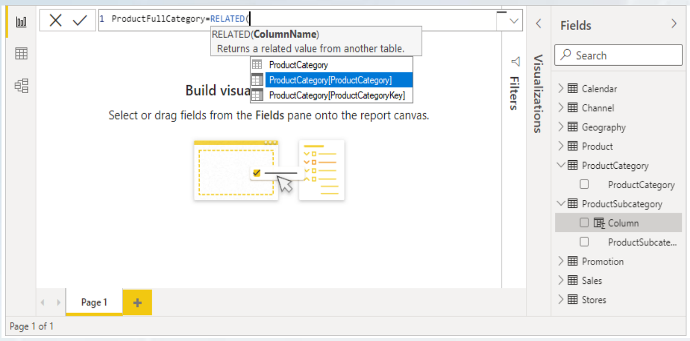
```
1 ProductFullCategory=rel
```

Returns a related value from another table.

Build visuals with your data

Select or drag fields from the **Fields** pane onto the report canvas.

An opening parenthesis appears, along with another suggestion list of the related columns you can pass to the RELATED function, with descriptions and details of expected parameters.



```
1 ProductFullCategory=RELATED(
```

RELATED(**ColumnName**)
Returns a related value from another table.

Build visua

Select or drag fields from the **Fields** pane onto the report canvas.

4.You want the **ProductCategory** column from the **ProductCategory** table.
Select **ProductCategory[ProductCategory]**, press **Enter**, and then type a closing parenthesis.

Syntax errors are most often caused by a missing or misplaced closing parenthesis, although sometimes Power BI Desktop will add it for you.

5.You want dashes and spaces to separate the **ProductCategories** and **ProductSubcategories** in the new values, so after the closing parenthesis of the first expression, type a space, ampersand (**&**), double-quote (**"**), space, dash (**-**), another space, another double-quote, and another ampersand. Your formula should now look like this:

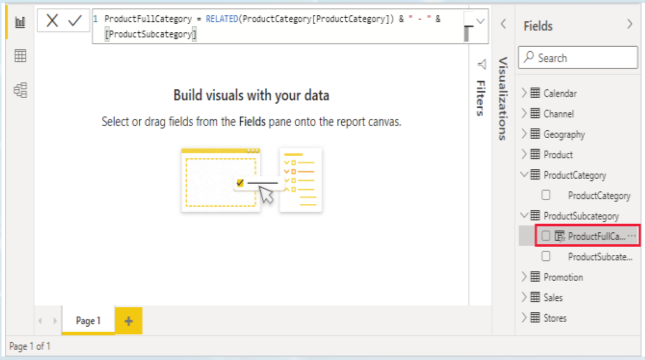
ProductFullCategory = RELATED(ProductCategory[ProductCategory]) & " - " &

If you need more room, select the down chevron on the right side of the formula bar to expand the formula editor. In the editor, press **Alt + Enter** to move down a line, and **Tab** to move things over.

6.Enter an opening bracket (**[**), and then select the **[ProductSubcategory]** column to finish the formula.

You didn't need to use another RELATED function to call the **ProductSubcategory** table in the second expression, because you're creating the calculated column in this table. You can enter **[ProductSubcategory]** with the table name prefix (fully qualified) or without (non-qualified).

7.Complete the formula by pressing **Enter** or selecting the checkmark in the formula bar. The formula validates, and the **ProductFullCategory** column name appears in the **ProductSubcategory** table in the **Fields** pane.



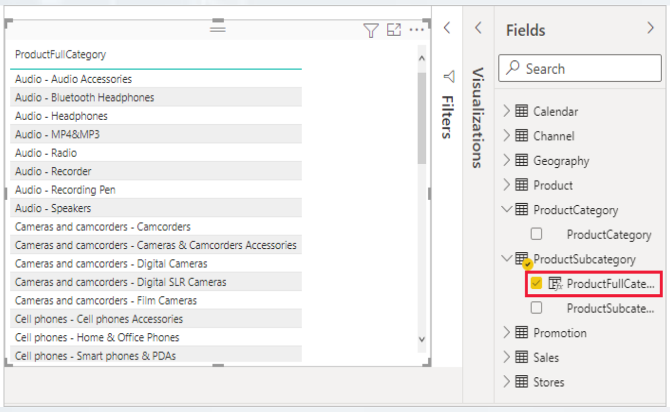
```
ProductFullCategory = RELATED(ProductCategory[ProductCategory]) & " - " &
[ProductSubcategory]
```

In Power BI Desktop, calculated columns have a special icon in the **Fields** pane, showing that they contain formulas. In the Power BI service (your Power BI site), there's no way to change formulas, so calculated columns don't have icons.

Use your new column in a report

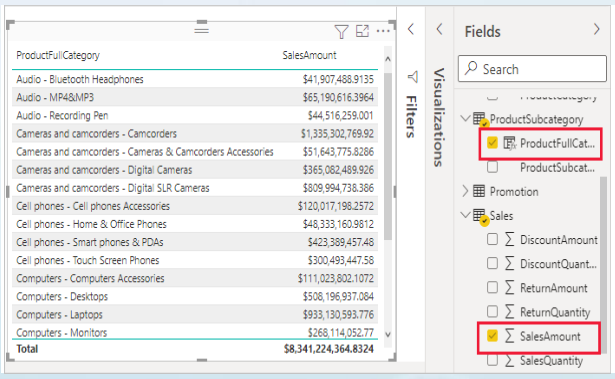
Now you can use your new **ProductFullCategory** column to look at **SalesAmount** by **ProductFullCategory**.

1. Select or drag the **ProductFullCategory** column from the **ProductSubcategory** table onto the Report canvas to create a table showing all of the **ProductFullCategory** names.



ProductFullCategory
Audio - Audio Accessories
Audio - Bluetooth Headphones
Audio - Headphones
Audio - MP4&MP3
Audio - Radio
Audio - Recorder
Audio - Recording Pen
Audio - Speakers
Cameras and camcorders - Camcorders
Cameras and camcorders - Cameras & Camcorders Accessories
Cameras and camcorders - Digital Cameras
Cameras and camcorders - Digital SLR Cameras
Cameras and camcorders - Film Cameras
Cell phones - Cell phones Accessories
Cell phones - Home & Office Phones
Cell phones - Smart phones & PDAs

2. Select or drag the **SalesAmount** field from the **Sales** table into the table to show the **SalesAmount** for each **ProductFullCategory**.



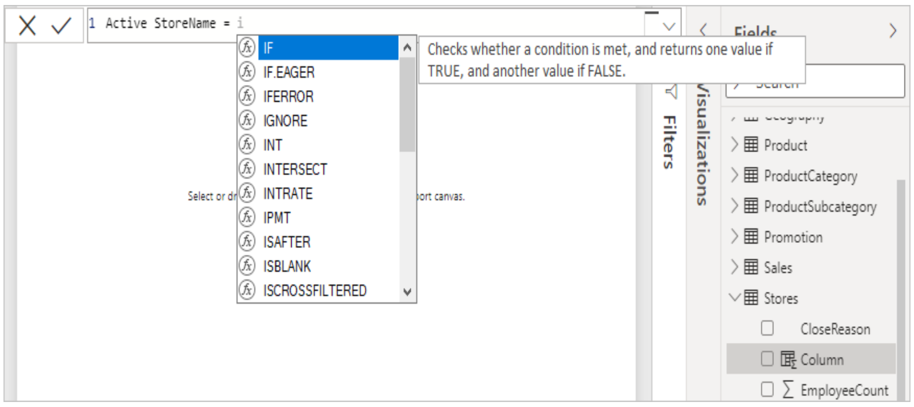
ProductFullCategory	SalesAmount
Audio - Bluetooth Headphones	$41,907,488.9135
Audio - MP4&MP3	$65,190,616.3964
Audio - Recording Pen	$44,516,259.001
Cameras and camcorders - Camcorders	$1,335,302,769.92
Cameras and camcorders - Cameras & Camcorders Accessories	$51,643,775.8286
Cameras and camcorders - Digital Cameras	$365,082,489.926
Cameras and camcorders - Digital SLR Cameras	$809,994,738.386
Cell phones - Cell phones Accessories	$120,017,198.2572
Cell phones - Home & Office Phones	$48,333,160.9812
Cell phones - Smart phones & PDAs	$423,389,457.48
Cell phones - Touch Screen Phones	$300,493,447.58
Computers - Computers Accessories	$111,023,802.1072
Computers - Desktops	$508,196,937.084
Computers - Laptops	$933,130,593.776
Computers - Monitors	$268,114,052.77
Total	**$8,341,224,364.8324**

Create a calculated column that uses an IF function

The Contoso Sales Sample contains sales data for both active and inactive stores. You want to ensure that active store sales are clearly separated from inactive store sales in your report by creating an **Active StoreName** field. In the new **Active StoreName** calculated column, each active store will appear with the store's full name, while the sales for inactive stores will be grouped together in one line item called **Inactive**.

Fortunately, the **Stores** table has a column named **Status**, with values of "On" for active stores and "Off" for inactive stores, which we can use to create values for our new **Active StoreName** column. Your DAX formula will use the logical IF function to test each store's **Status** and return a particular value depending on the result. If a store's **Status** is "On", the formula will return the store's name. If it's "Off", the formula will assign an **Active StoreName** of "Inactive".

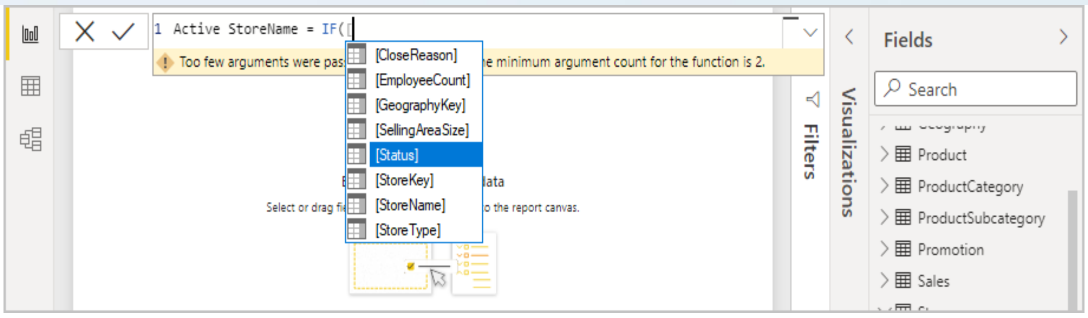
1. Create a new calculated column in the **Stores** table and name it **Active StoreName** in the formula bar.
2. After the = sign, begin typing **IF**. The suggestion list will show what you can add. Select **IF**.



```
Active StoreName = i
```

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

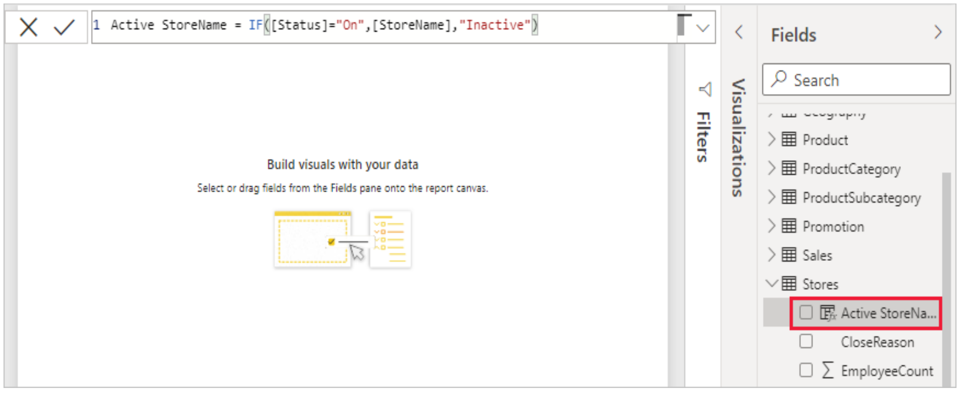
3. The first argument for **IF** is a logical test of whether a store's **Status** is "On". Type an opening bracket **[**, which lists columns from the **Stores** table, and select **[Status]**.



```
Active StoreName = IF([
```

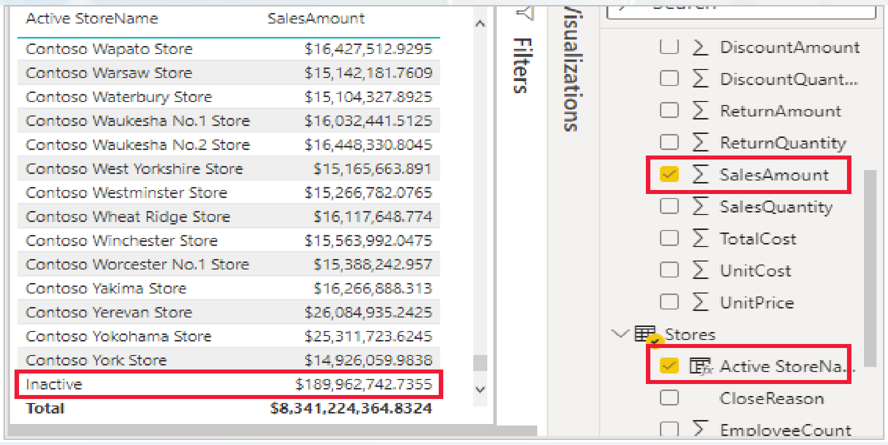
Too few arguments were pas… e minimum argument count for the function is 2.

4. Right after **[Status]**, type **="On"**, and then type a comma (**,**) to end the argument. The tooltip suggests that you now need to add a value to return when the result is TRUE.
5. If the store's status is "On", you want to show the store's name. Type an opening bracket (**[**) and select the **[StoreName]** column, and then type another comma. The tooltip now indicates that you need to add a value to return when the result is FALSE.
6. You want the value to be "Inactive", so type **"Inactive"**, and then complete the formula by pressing **Enter** or selecting the checkmark in the formula bar. The formula validates, and the new column's name appears in the **Stores** table in the **Fields** pane.



```
Active StoreName = IF([Status]="On",[StoreName],"Inactive")
```

7. You can use your new **Active StoreName** column in visualizations just like any other field. To show **Sales Amounts** by **Active StoreName**, select the **Active StoreName** field or drag it onto the Report canvas, and then choose the **SalesAmount** field or drag it into the table. In this table, active stores appear individually by name, but inactive stores are grouped together at the end as **Inactive**.



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**

DAX Date Function

Returns the specified date in datetime format.

Syntax

DATE (<year>, <month>, <day>)

DAX date functions always return a datetime data type. However, you can use formatting to display dates as serial numbers if you want.

Example

```
= DATE (2016,8,5) returns 8/5/2016 12:00:00 AM

= DATE (2016,8,45) returns 9/14/2016 12:00:00 AM

= DATE (2016,8, -5) returns 7/26/2016 12:00:00 AM

= DATE (2016,15,15) returns 3/15/2017 12:00:00 AM
```

DAX DATEDIFF Function

Returns the count of interval boundaries crossed between two dates.

Syntax

```
DATEDIFF (<start_date>, <end_date>, <interval>)
```

If start_date is larger than end_date, an error value is returned.

The values given to the parameter interval are constants and not strings. Hence, they should not be enclosed in double quotation marks.

Example

```
= DATEDIFF (DATE (2016,1,1), DATE (2016,3,31), MONTH) returns 2.

= DATEDIFF (DATE (2016,1,1), DATE (2016,4,1), MONTH) returns 3.

= DATEDIFF (DATE (2016,1,1), DATE (2016,3,31), DAY) returns 90.

= DATEDIFF (DATE (2016,1,1), DATE (2016,3,31), HOUR) returns 2160.

= DATEDIFF (DATE (2016,1,1), DATE (2016,3,31), SECOND) returns 7776000.
```