

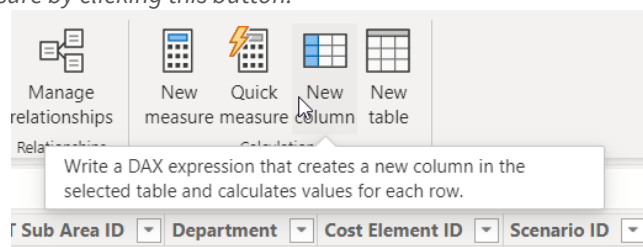
Hint

We select Fact since we want to create new columns in the Fact table. If we have another table selected, the new column is created in the other column.

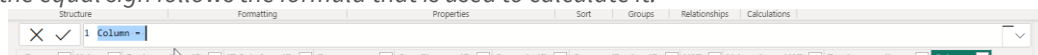
- Let's create a **VAT** column: It should display the "VAT" rate from the table "regions_countries_VAT" next to our transactions in the Fact table.

Hint

You create a measure by clicking this button:



Calculated columns always start with the name of the new column before the equal sign. After the equal sign follows the formula that is used to calculate it.

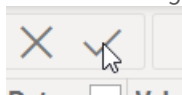


The measure makes use of the `RELATED()` function since it is retrieving a value that is in a related table:

Measure name = `RELATED('Related table name'[Related column name])`

⚠ Please do not copy-paste formulas from this sheet. When you type formulas, the auto-complete function will help you avoid mistakes. Additionally, the hints given may be incomplete or have different names for tables and columns. Always use your own data model as the starting point for your formula. Only write a formula once you are certain you know what you want to achieve with it. Otherwise, feel free to ask for help.

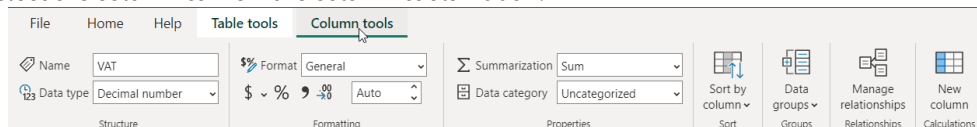
Once you finished writing your formula, you can press the tick icon to save it:



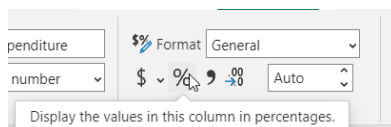
The new column should look like this:

VAT
0.1
0.1
0.1
0.1

8. Before moving on to the next column, let's format the VAT column so all values are shown as percentages. Select the column to view the column tools ribbon:



9. Click the percentage icon in the Formatting tab. All visuals using the VAT column will now use the percentage formatting when displaying the data.



10. Create a **Value_times_VAT** column: It displays the VAT paid on any transaction. It is calculated by multiplying the value without tax by the tax rate.

Solution

```
2 Value_times_VAT = [VAT]*[value]
```

11. Create a **Total_expenditure** column: It displays the sum of the previously created columns Value and Value_times_VAT.

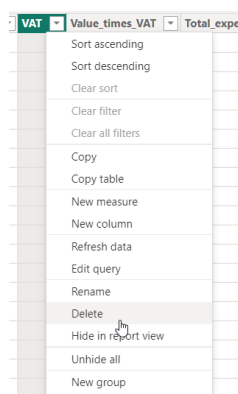
Solution

```
1 Total_expenditure = [value]+[value_times_VAT]
```

Bonus Exercise

We now have our value before tax **'Fact'[Value]** and our value after tax **'Fact'[Total_expenditure]**. In the process of getting to this, we had to create two intermediary columns: **'Fact'[VAT]** and **'Fact'[Value_times_VAT]**. Because of this, our table now looks quite cluttered, especially because most analyses will not require them. Our task is to remove the intermediary columns to improve our model for usability and performance.

1. Try deleting **'Fact'[VAT]** to declutter the model. What happens?



2. All other columns that built on it now show an error:

Value_times_VAT	Total_expenditure
#ERROR	#ERROR
#ERROR	#ERROR
#ERROR	#ERROR
#ERROR	#ERROR
#ERROR	#ERROR

3. Add '**Fact**'[VAT] back and see if the error disappears:

VAT = RELATED(regions_countries_VAT[VAT])

So since that didn't work, how can we remove our intermediary columns? The solution is to use **variables** within one formula. Variables are declared with **var**. To demark the final formula that is returned we use **return**.

4. Try writing a function using variables by following this example. Make sure to adjust the formula in case your columns have different names.


```
Total_expenditure_var =
var VAT = RELATED(regions_countries_VAT[VAT])
var Value_times_VAT = [VAT]*[Value]
return [Value]+[Value_times_VAT]
```

5. Do the values match with the previous Total_expenditure column?

Total_expenditure	VAT	Total_expenditure_var
11208.263	0.1	11208.263
1369.5	0.1	1369.5
1369.5	0.1	1369.5
1369.5	0.1	1369.5
11208.274	0.1	11208.274


Exercise 3.2: DAX measures

Thanks to the Scenario table added in Exercise 3.1, we can now distinguish between our plan values and our actual expenses.

1.  In data view: add following **measures** to the fact table:
 - a. A measure named **Actual**: Shows the sum of total expenditures in the filter context of Scenario = "Actual"

Hint

You will want to sum up all entries in calculated column "Total_expenditures" in Fact table where ScenarioID equals 1, which corresponds to scenario description = "Actual" in Scenario table. To change a filter context, we always use the function **CALCULATE(<calculation>, <filter context>)**.

It is a good idea to check your work on measures with a table in the  report view, since the results are not immediately visible. For this measure, try creating a table that shows Date, Total_expenditures and Actual as columns:

Date	Total_expenditure	Actual
01/01/2014	154.859.206,67	\$65.194.791
01/02/2014	147.675.226,30	\$73.155.763
01/03/2014	158.914.417,70	\$80.537.287
01/04/2014	150.055.958,75	\$73.636.319
01/05/2014	153.795.015,98	\$81.498.047
01/06/2014	159.567.319,54	\$81.823.846
01/07/2014	158.079.417,39	\$80.917.518
01/08/2014	147.321.833,79	\$76.409.791
01/09/2014	151.314.922,52	\$79.692.486
01/10/2014	149.705.517,95	\$77.467.590
01/11/2014	151.458.555,86	\$81.070.742
01/12/2014	176.491.059,76	\$104.477.905
Total	1.859.238.452,21	\$955.882.084

Solution

```
1 Actual = CALCULATE(SUM([total_expenditure]), Scenario[ScenarioDescription] = "Actual") + 0
```

- b. Plan: Sum over total_expenditure with filter on Scenario[ScenarioDescription]= "Plan" (analogous to measure "Actual")

Solution

```
1 Plan = CALCULATE(SUM([total_expenditure]), Scenario[ScenarioDescription] = "Plan") + 0
```

- c. perc_deviation_plan_actual: values from column [Actual] minus values from column [Plan] divided by [Plan]

Hint

In maths, we try to always avoid dividing by 0. Since we have to divide by [Plan], we must make sure that [Plan] is never empty or equal to 0. Since this is the case, we can use an IF(<this>, <then that>, <or else>) statement to circumvent this problem.

Challenge: Alternatively, we can use the DIVIDE(<numerator>, <denominator>, <exception>) function.

Solution

(using an IF() statement)

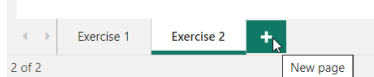
```
perc_deviation_plan_actual = IF([Plan]>0, ([Actual]-[Plan])/[Plan],0)
```

Solution

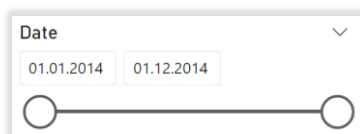
(using a DIVIDE() statement)

```
perc_deviation_plan_actual = DIVIDE([Actual]-[Plan],[Plan],0)
```

2.  In report view: add a new sheet and call it “Exercise 3”



3. Add following visualizations:
- Visualize the percentage difference between planned and actual expenditures (perc_deviation_plan_actual) per country in a clustered column chart. Give the chart a meaningful title.
 - Add a table with countries and their actual expenditures. Add a filter for those countries that do not have planned expenditures. Rename the column “Actual Expenditure” (Double click on “Actual” in Values section of visual)
 - Add a title for the table: “Countries with no planned expenditures”
 - Add a slicer with date values from Fact table:



Bonus Exercise

1. In ‘Fact’ table, create a measure named **max_date** for the maximum available date so that we can easily find the latest date shown.

Hint

Use the MAX() function with the Date column from the Fact table

Solution

```
1 max_date = MAX(Fact[Date])
```

- Add a card visual that displays the latest date. Format the visual, so it looks like in the screenshot on the next page.
- Play with the date slicer. Why does the latest available date change?
- If not done already, “Exercise 3.xlsx”, import sheet ‘Date’ and make sure it is connected to the ‘Fact’ table
- In date table: add a measure with the maximum available date and name it **universal_max_date**

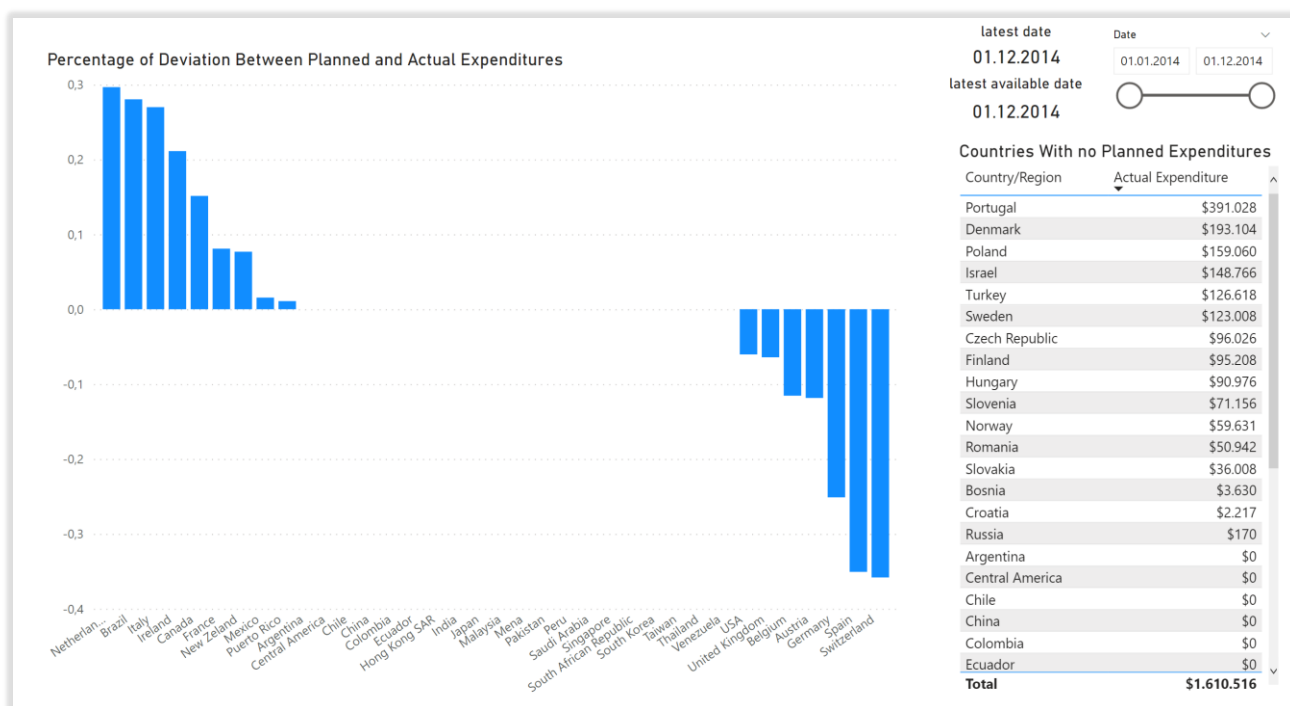
Hint

Write the same measure as in part 1 but use the [Date] column from the ‘Date’ table instead of from the ‘Fact’ table.

Solution

```
1 universal_max_date = max(Date[Date])
```

- Add another card visual containing universal_max_date and give it a descriptive title.
- Try formatting the report, so it looks like in the below screenshot. (The values in the left visual may differ from the screenshot since the data source has changed since the last update)



Challenge:

Create the **universal_max_date** using the 'Fact'[Date] column and a filter context using the formulas CALCULATE() and REMOVEFILTERS().

For help, read these articles from Microsoft:

[CALCULATE function \(DAX\) - DAX | Microsoft Learn](#)

[REMOVEFILTERS function \(DAX\) - DAX | Microsoft Learn](#)