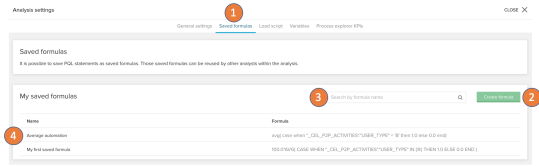


# Saved formulas

Saved formulas are a useful tool to save PQL statements. The saved formulas may be also be used by other analysts in the analysis.

## Manage saved formulas



1. Access the saved formulas in the analysis settings.
2. To create new saved formulas add them with the button 'Create formula'.
3. Search the existing list of saved formulas.
4. To edit or view the saved formula click on the list entry.

## Create saved formulas



1. Set a name for the saved formula. This name will be shown in the data chooser, formula editor and function library of the visual editor.
2. Create a description for the formula. This description is available for the analysts in the function reference.
3. Add parameters to abstract parts of your formula and make them adjustable. More on parameters follows in the next section.
4. Define the formula statement you want to save.
5. You can access the full formula editor for easier formula creation.
6. Cancel or save your actions. Only on save new saved formulas will be stored.



### Formula statements

All PQL statements supported in the Celonis analyses can be stored in a saved formula.

The only exception are variables. **Variables are not supported in saved formulas!**

To apply variables within saved formulas, read the next section.

## Apply Saved formulas in the analysis

Saved formulas can be used in all formula inputs in the analysis.

## Syntax

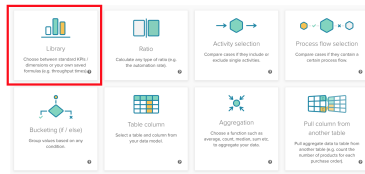
```
KPI("<Name of the saved formula>" [, '<Parameter Input>'])
```

Fast access to the saved formulas is provided with:

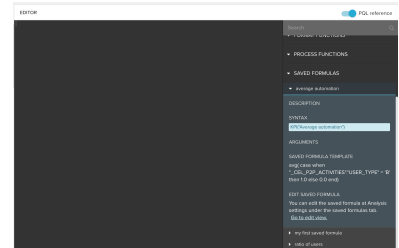
The data chooser



The library in the visual editor



The pql reference



## Parameters

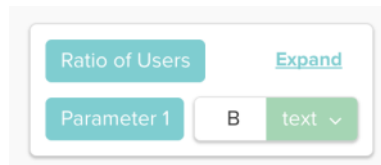
Parameters allow you to abstract a part of the saved formula's statement and the parameter will provide a type specific input in the visual editor. The end user re-using the saved formula in a component has to set the input for the parameter you defined either in code or in the provided input block of the visual editor. Parameters are set on the saved formulas by putting the parameter's placeholder, e.g. {p1} into the formula. The following parameter types are available:

### Parameter type text

Saved formula



Applied on visual editor: Input B will replace {p1} on the formula statement.



Applied on code editor: B will replace {p1} on the formula statement.

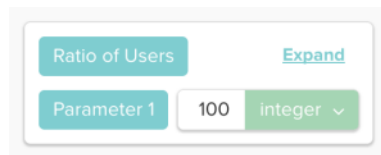
```
KPI("Ratio of Users", 'B')
```

### Parameter type integer

Saved formula



Applied on visual editor: Input 100 will replace {p1} on the formula statement.



Applied on code editor: 100 will replace {p1} on the formula statement.

```
KPI("Ratio of Users", 100)
```

### Parameter type decimal

Saved formula



Applied on visual editor: Input 1.0 will replace {p1} on the formula statement.

Applied on code editor: 1.0 will replace {p1} on the formula statement.

```
KPI("Ratio of Users", 1.0)
```

## Parameter type condition

Saved formula

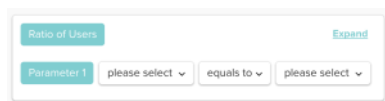


Applied on visual editor: User type equals B will replace {p1} on the formula statement.

Original state:



With input values:



Applied on code editor: "\_CEL\_P2P\_ACTIVITIES"."USER\_TYPE" = 'B' will replace {p1} on the formula statement.

```
KPI("Ratio of Users",
  "_CEL_P2P_ACTIVITIES"."
  USER_TYPE" = 'B' )
```

## Parameter type column

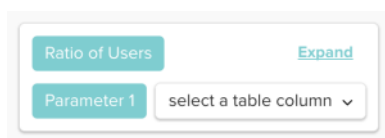
Saved formula

(optionally a default value for the selected column can be set)



Applied on visual editor: The selected column via the available dropdown will replace {p1} on the formula statement.

Initial state:



Selected columns:



Applied on code editor: "\_CEL\_P2P\_ACTIVITIES"."USER\_TYPE" will replace {p1} on the formula statement.

```
KPI("Ratio of Users",
  "_CEL_P2P_ACTIVITIES"."
  USER_TYPE" )
```

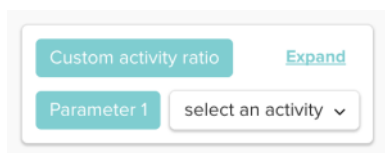
## Parameter type activity name

Saved formula

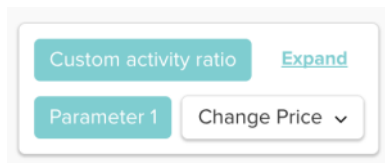


Applied on visual editor: The selected activity via the available dropdown will replace {p1} on the formula statement.

Initial state:



Selected activity:



Applied on code editor: 'Change Price' will replace {p1} on the formula statement.

```
KPI("Custom activity ratio",
  'Change Price' )
```

## Parameter type table

Saved formula

(optionally a default value for the selected table can be set)

Applied on visual editor: The selected table via the available dropdown will replace {p1} on the formula statement.

Initial state:

Selected table:

Applied on code editor: "\_CEL\_P2P\_ACTIVITIES" will replace {p1} on the formula statement.

```
KPI("Custom Count Items",
  "_CEL_P2P_ACTIVITIES")
```

## Parameter type date and time

Saved formula

Applied on visual editor: The selected timestamp will replace {p1} on the formula statement.

Applied on code editor: 2017-12-6 8:56:45:00 will replace {p1} on the formula statement.

```
KPI("Datediff against
eventlog", {d '2017-12-6 8:56:
45:00'})
```

## Combine parameters and variables to adjust formulas from the analyses frontend

You can use a combination of dropdown buttons and the saved formulas to allow users with only view rights to adjust what KPI they are shown in a chart or table. The following example will showcase how you can set up the Rework Ratio to be dependent on the activities selected by the user:

1. Create a saved formula with a text parameter to replace the activity names in your case when statement.
2. Input the saved formula's KPI statement with the variable name input to the parameters value.
3. Create a Dropdown that writes to the variable.
4. Now selecting multiple or a single value on the dropdown will change the calculated value accordingly.

3

Button dropdown



Title

Button color

Pick color...

Load Entries

Formula

```
["_CEL_P2P_ACTIVITIES", "ACTIVITY_EN"]
```

Write to variable

Separator

.

WRAPPING CHARACTERS

Start

'

End

'

Escape character

☒ Allow multiple selections

☐ Component is not filtered with selections

Your variable will look like

"Value1";Value2"