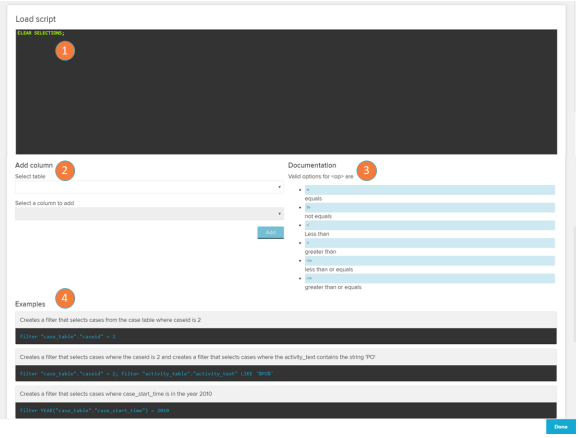
Filter Editor

Filters offer the possibility to predefine what part of the data is shown in the respective part of the analysis.

Filter application and Load Scripts



Filters are implemented in form of a load script. These load scripts can be set for:

- Documents: Load scripts in documents will be applied to the whole document. Filters set here will be global and applied to all analysis parts. You can also define selections that will be applied when the analysis is reloaded (see the Load Scripts page).
- Sheets: Filters that are defined for sheets will filter down everything on the sheet. But other parts of the analysis will not be affected (see item 3 of the Sheets settings section).
- Components: Filters for single components only have an effect on this component.
- Stories: Filters in stories will define what part of the data will actually be shown in the download. The analyst can predefine which selection / filter the download will show.

The Load Scripts screen is composed of the .

1. Load Scripts are defined using PQL statements on the text box. If more than one script is applied, they should be separated by a semicolon. For syntax info, see the 'Filter Syntax' section below
2. The add column section aids the input of Load Scripts. It allows the user to select a column from the connected databases and returns a default script based on it.
   The default script has a format: *FILTER "TABLE_NAME"."COLUMN_NAME" <op> <value>*
3. The documentation shows the possible values the *<op>* (operator) might assume in the Load Script formula.
4. Here you can find examples of useful Load Scripts formulas. Remember to replace the table names and columns according to the ones of the data tables connected to your current analysis.

Filter Syntax

The filter syntax defines the types of inputs the filter editor load script accepts.

Syntax	Example	Description
FILTER PROCESS <STATEMENT> <activity>	FILTER PROCESS EQUALS 'Create Purchase Order'	This example would filter for all cases that include the activity "Create Purchase Order".
FILTER <FUNCTION>("TABLE_NAME"."COLUMN_NAME") <op> <value>	FILTER YEAR("EVENTLOG"."EVENTTIME") = 2010	This example would filter for all occurrences in 2010.
FILTER "TABLE_NAME"."COLUMN_NAME" <op> <value>	FILTER "EVENTLOG"."USER_TYPE" != 'BATCH'	This example would filter for non-automated actions in the process.

Process Query Language (PQL)

Process Query Language (PQL) is an extension to the normal SQL used to query databases. PQL has been especially designed to query and filter process flows and process patterns.
Consequently, PQL offers many additional commands to improve the analysis of processes using Celonis Process Mining 4. Besides, PQL supports all standard SQL commands of the used database server (e.g. MS SQL, SAP HANA and Oracle). Therefore all standard commands and functions of the database server can be used in Celonis 4, too.

The full documentation on PQL Statements can be found here.

Some of the most common examples:

Statement	Description	Example
A OR B	Returns true if one of the arguments is true, otherwise false.	'true' OR 'false' = 'true'
A AND B	Returns true if both of the arguments are true, otherwise false.	'true' AND 'false' = 'false'
A EQUAL B	Returns true if A is equal to B, otherwise false.	3 = 3 = 'true'
A NOT_EQ_TO B	Returns true if A is not equal to B, otherwise false.	4 != 5 = 'true'
A > B	Returns true if A is greater than B, otherwise false.	2 > 4 = 'false'
A >= B	Returns true if A is greater than or equal to B, otherwise false.	3 >= 3 = 'true'
A < B	Returns true if A is less than B, otherwise false.	5 < 3 = 'false'
A <= B	Returns true if A is less than or equal to B, otherwise false.	2 <= 3 = 'true'
A + B	Returns the result of adding the values of A and B.	3 + 3 = 6
A - B	Returns the result of substracting the values of A and B.	4 - 1 = 3
A * B	Returns the result of multiplication the values of A and B.	2*3 = 6
A / B	Returns the result of dividing the values of A and B.	8 / 2 = 4
PROCESS EQUALS 'Activity1'	Returns the cases that include the defined activity.	PROCESS_EQUAL 'Delivery'
PROCESS EQUALS START 'Activity 1' TO 'Activity 2' TO ANY TO 'Activity 3' END	Returns the cases that fulfill the defined process. TO ANY can also be included in the process. Another syntax would be: PROCESS # ^ 'Activity 1' -> 'Activity 2' -> * -> 'Activity 3' $. 1. implicates START, ^ = FROM, -> = TO, * = ANY and $ = END	PROCESS EQUALS START 'Create Purchase Order' TO ANY TO 'Delivery' END PROCESS # ^'Create Purchase Order' -> * -> 'Delivery' $
A LIKE 'abcd'	Returns 'true' if S1 equals S2. Premise: S1 and S2 are strings. Be careful: output is Boolean.	LIKE("Windows","Apple") = 'false'
A IN ('abcd', 'abc', 'abcdefg')	All values of A within an interval (B,C)	(4, 5, 2, 8) BETWEEN 3 AND 5 = 4, 5
ISNULL(X)	Returns '1' if the input column contains a null-element and '0' in all other cases.	ISNULL('Salary') = 1
CASE WHEN X LIKE Y THEN A ELSE B END	Returns A if X is equal to Y. Otherwise, B is returned.	CASE WHEN 'apples' LIKE 'oranges' THEN 'wow' ELSE 'impossible' END