Column Aggregation Function

The Column Aggregation function performs aggregation operations (sum, min, max, or count) on numeric values row by row from an input column and shows the cumulative results row by row in a results column. This function could be used for the following types of calculations:

- opening/closing stock values
- identifying minimum/maximum temperature over time
- counting discrete product values

Cumulating values within a single column in an Integrator transform has always been possible, but required a custom Groovy script to achieve the right outcome. The column aggregation function simplifies these types of calculations.

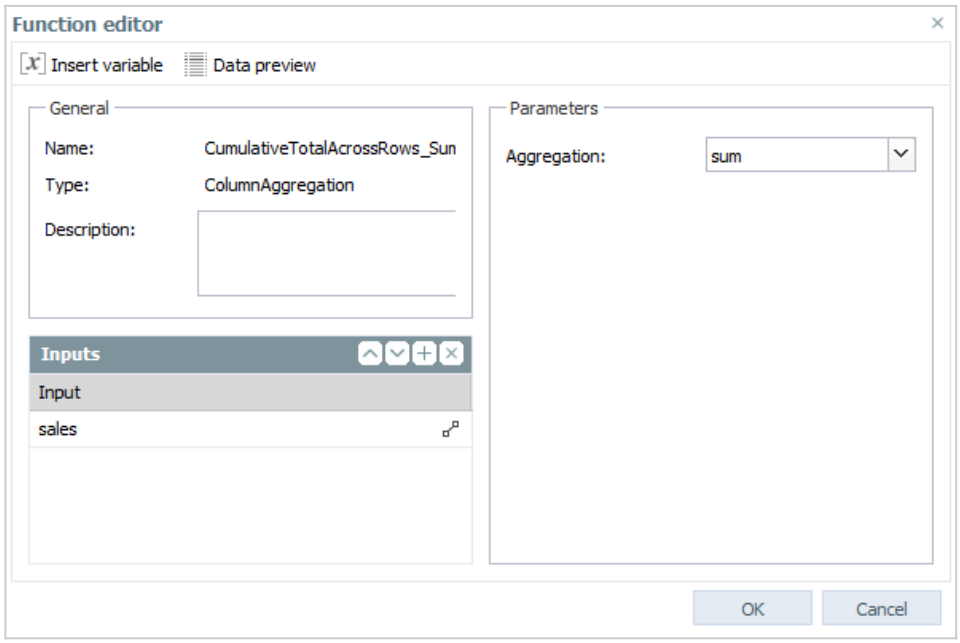
Function editor

The function is defined in the **Function editor** of a Field Transform. This function can have maximum 2 inputs (columns):

Column 1 : Value to aggregate on (mandatory)
Column 2 : A condition that can reset the aggregation (optional)

The screenshot below shows the **Function editor** with a single input,

sales and aggregation set to **sum**.



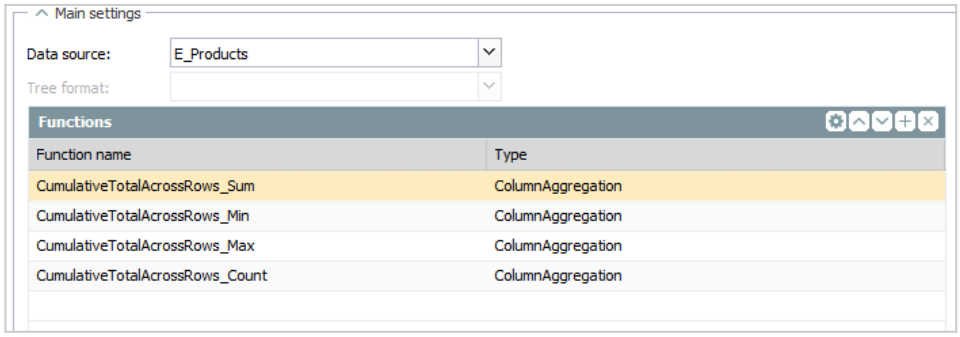
The output column shows an aggregated value of all rows from an input column, up to the current row.

Examples

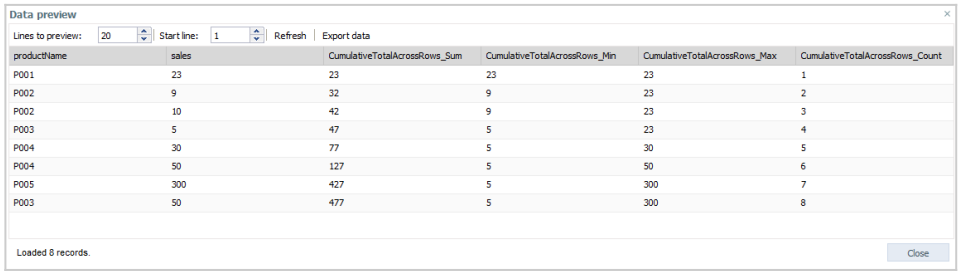
You can see column aggregation in action in the Integrator sample project **sampleFieldTransform**. This sample project has two transforms that use column aggregation: one with a single input column and one with two input columns. In each case, the data source is an extract, **E_Products**, which contains product names (**productName**) and products sold (**sales**).

Single input column

The example **transformT_ColumnAggregationSingleInput** has a single input column, **sales**. The transform consists of 4 functions, each of which performs a different aggregation on the source column:

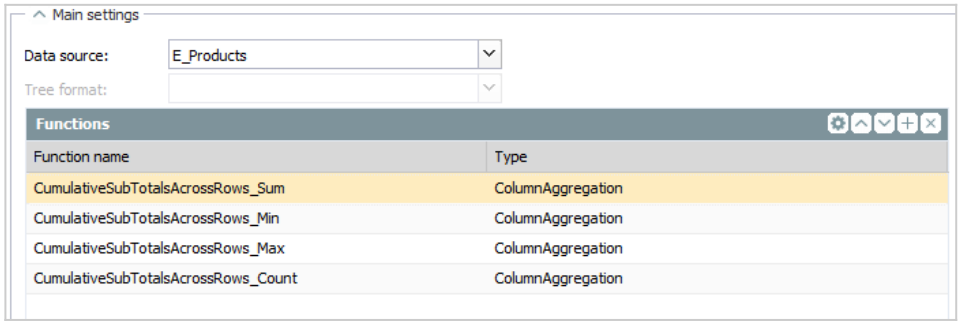


The data preview shows the columns that result from the aggregation on the sales column.

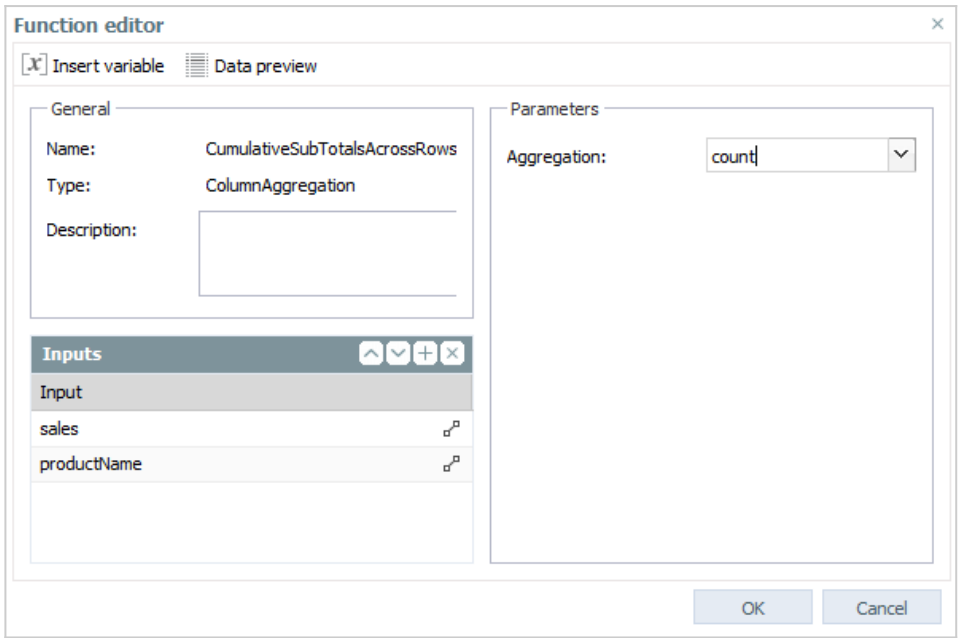


Two input columns

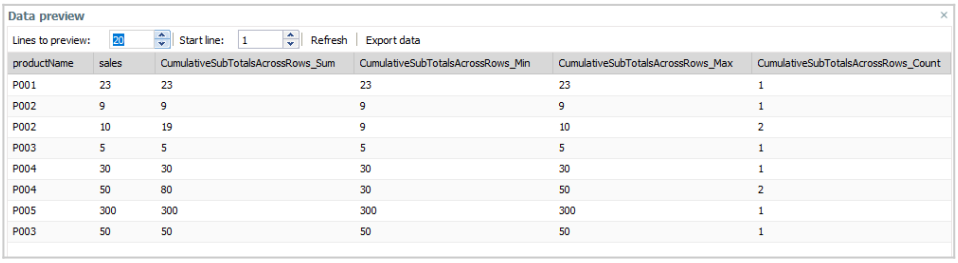
The example **T_ColumnAggregationTwoInputs** has two input columns, **sales** and **productName**. As in the previous example, the transform consists of 4 functions, each of which performs a different aggregation on the source column:



However, in this case there are two input columns defined in the Function editor, **sales** and **productName**. The aggregations will be grouped by the second input (in this case, **productName**).



The resulting columns show the figures from the **sales** column aggregated with respect to the values in the **productName** column. The **productName** values “reset” the aggregation in the respective results columns.



Data preview

Lines to preview: 20 Start line: 1 Refresh Export data

productName	sales	CumulativeSubTotalsAcrossRows_Sum	CumulativeSubTotalsAcrossRows_Min	CumulativeSubTotalsAcrossRows_Max	CumulativeSubTotalsAcrossRows_Count
P001	23	23	23	23	1
P002	9	9	9	9	1
P002	10	19	9	10	2
P003	5	5	5	5	1
P004	30	30	30	30	1
P004	50	80	30	50	2
P005	300	300	300	300	1
P003	50	50	50	50	1