Create Aggregations

Contents:

You can apply aggregate functions to groups of values in one or more columns to generate aggregated data. Depending on how you configure the Group By transformation, the output of these transformations is a new table or one or more columns in the current dataset.

Limitations

- The Group By transformation does not support nested expressions. You cannot insert multiple nested expressions in your computed value.
- The Group By transformation supports aggregation functions only. For more information, see *Aggregate Functions*.

Example Data

The following table contains test score data from a set of students for four separate tests, spread over two days:

Student	TestDate	TestNum	TestScore
Anna	09/08/2018	1	84
Ben	09/08/2018	1	71
Caleb	09/08/2018	1	76
Danielle	09/08/2018	1	87
Anna	09/08/2018	2	92
Ben	09/08/2018	2	86
Caleb	09/08/2018	2	99
Danielle	09/08/2018	2	73
Anna	09/15/2018	3	86
Ben	09/15/2018	3	99
Caleb	09/15/2018	3	86
Danielle	09/15/2018	3	80
Anna	09/15/2018	4	85
Ben	09/15/2018	4	87
Caleb	09/15/2018	4	79
Danielle	09/15/2018	4	93

Copyright © 2021 Trifacta Inc.

Aggregating across all rows (no grouping)

You can perform basic computations across all rows of the dataset. For example, the following transformation creates a new column containing the average test score for all students:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`ROUND(AVERAGE(Score),2)`
Parameter: New column name	`avg_TestScore`

The above results in a new column called, `average_TestScore`, containing the single value `85.19`, which is the average of all students' test scores rounded to two decimal places.

> **NOTE:** These types of aggregations are known as **flat aggregations**. In larger datasets, performing flat aggregations can be computationally intensive. Be careful in computing any aggregation functions across a large number of rows.

Aggregate grouped-by rows

For the above example data, suppose you are interested in the average score for each student. In this case, you must compute the average (`AVERAGE(TestScore)`) for each student.

In the previous transformation, you used the New Formula transformation. When you are computing aggregations across groups of values in a column, you must use the Group By transformation:

Transformation Name	`Group By`
Parameter: Group By	`Student`
Parameter: Values	`AVERAGE(TestScore)`
Parameter: Type	`Group by as new column(s)`

Note that the above transformation does not contain the rounding function. Nested expressions are not supported in the Group By transformation. To round the values, add the following transformation as the next step:

Transformation Name	`Edit column with formula`
Parameter: Columns	`average_TestScore`
Parameter: Formula	`ROUND(average_TestScore,2)`

You may wish to rename the newly generated column to something like `average_TestScorePerStudent` instead. See *Rename Columns*.

The output data should look like the following:

Student	TestDate	TestNum	TestScore	average_TestScorePerStudent	average_TestScore
Anna	09/08/2018	1	84	86.75	85.19
Ben	09/08/2018	1	71	85.75	85.19

Copyright © 2021 Trifacta Inc.

Caleb	09/08/2018	1	76	85	85.19
Danielle	09/08/2018	1	87	83.25	85.19
Anna	09/08/2018	2	92	86.75	85.19
Ben	09/08/2018	2	86	85.75	85.19
Caleb	09/08/2018	2	99	85	85.19
Danielle	09/08/2018	2	73	83.25	85.19
Anna	09/15/2018	3	86	86.75	85.19
Ben	09/15/2018	3	99	85.75	85.19
Caleb	09/15/2018	3	86	85	85.19
Danielle	09/15/2018	3	80	83.25	85.19
Anna	09/15/2018	4	85	86.75	85.19
Ben	09/15/2018	4	87	85.75	85.19
Caleb	09/15/2018	4	79	85	85.19
Danielle	09/15/2018	4	93	83.25	85.19

Generate new aggregation table

Suppose you wish to calculate the minimum, maximum, and average scores for each test. In this case, it may be more useful to create a new table in which the student names have been removed:

Transformation Name	`Group By`
Parameter: Group By	`TestNum`
Parameter: Values1	`MAX(TestScore)`
Parameter: Values2	`MIN(TestScore)`
Parameter: Values3	`AVERAGE(TestScore)`
Parameter: Type	`Group by as new table`

The resulting data looks like the following:

TestNum	max_TestScore	min_TestScore	average_TestScore
1	87	71	79.5
2	99	73	87.5
3	99	80	87.75
4	93	79	86

Tip: In this case, when you replace the existing table with a completely new table, data that is not included in the aggregation is lost. You can add columns to the list of values if you wish to bring forward untouched columns into the new table. You may also consider building aggregation tables in a recipe that is extended from the previous recipe, so that you can continue to work with the other columns in your dataset.

Copyright © 2021 Trifacta Inc.