SUMIF Function

Contents:

Generates the sum of rows in each group that meet a specific condition.

> **NOTE:** When added to a transform, this function is applied to the sample in the data grid. If you change your sample or run the job, the computed values for this function are updated. Transforms that change the number of rows in subsequent recipe steps do not affect the values computed for this step.

To perform a simple summing of rows without conditionals, use the `SUM` function. See *SUM Function*.

Basic Usage

```
pivot value: SUMIF(timeoutSecs, errors >= 1) group:serverAddress limit:1
```

Output: Generates a two-column table containing the unique values for `serverAddress` and the sum of the `timeoutSecs` column for that `serverAddress` value when the `errors` value is greater than or equal to `1`. The `limit` parameter defines the maximum number of output columns.

Syntax and Arguments

```
pivot value:SUMIF(col_ref, test_expression) [group:group_col_ref] [limit:limit_count]
```

Argument	Required?	Data Type	Description
col_ref	Y	string	Reference to the column you wish to evaluate.
test_expression	Y	string	Expression that is evaluated. Must resolve to `true` or `false`

For more information on syntax standards, see *Language Documentation Syntax Notes*.

For more information on the `group` and `limit` parameters, see *Pivot Transform*.

col_ref

Name of the column whose values you wish to use in the calculation. Column must be a numeric (Integer or Decimal) type.

Usage Notes:

Required?	Data Type	Example Value
Yes	String that corresponds to the name of the column	`myValues`

test_expression

This parameter contains the expression to evaluate. This expression must resolve to a Boolean (`true` or `false`) value.

Usage Notes:

Required?	Data Type	Example Value
Yes	String expression that evaluates to `true` or `false`	`(LastName == 'Mouse' && FirstName == 'Mickey')`

Examples

> **Tip:** For additional examples, see *Common Tasks*.

Example - Summarize Voter Registrations

This example illustrates how you can use the following conditional calculation functions to analyze polling data:

- `SUMIF` - Sum of a set of values by group that meet a specified condition. See *SUMIF Function*.
- `COUNTDISTINCTIF` - Sum of a set of values by group that meet a specified condition. See *COUNTDISTINCTIF Function*.

Source:

Here is some example polling data across 16 precincts in 8 cities across 4 counties, where registrations have been invalidated at the polling station, preventing voters from voting. Precincts where this issue has occurred previously have been added to a watch list (`precinctWatchList`).

totalReg	invalidReg	precinctWatchList	precinctId	cityId	countyId
731	24	y	1	1	1
743	29	y	2	1	1
874	0		3	2	1
983	0		4	2	1
622	29		5	3	2
693	0		6	3	2
775	37	y	7	4	2
1025	49	y	8	4	2
787	13		9	5	3
342	0		10	5	3
342	39	y	11	6	3
387	28	y	12	6	3
582	59		13	7	4
244	0		14	7	4
940	6	y	15	8	4
901	4	y	16	8	4

Transform:

First, you want to sum up the invalid registrations (`invalidReg`) by city:

```
derive type:single value:SUMIF(invalidReg, precinctWatchList == "y") group:cityId
as:'invalidRegbyCityId'
```

The `invalidRegbyCityId` column contains invalid registrations across the entire city. Now, at the county level, you want to identify the number of precincts that were on the watch list and were part of a city-wide registration problem. This step performs an aggregation:

```
pivot value:COUNTDISTINCTIF(precinctId, invalidRegbyCityId > 60) group:countyId limit:1
```

Results:

countyId	countdistinctif_precinctId
1	0
2	2
3	2
4	0

The voting officials in counties 2 and 3 should investigate their precinct registration issues.