

Sort Transform

Contents:

- *Basic Usage*
- *Syntax and Parameters*
 - *order*
- *Examples*
 - *Example - sort methods*
 - *Example - Sort by original row numbers*

Sorts the dataset based on one or more columns in ascending or descending order. You can also sort based on the order of rows when the dataset was created.

Limitations:

NOTE: This transform is intended primarily for use in the Transformer page. Sort order may not be preserved in the output files.

- If you generate a new sample after a `sort` transform has been applied, the sort order is not retained. You can re-apply the sort step, although the following limitations still apply.
- Sort order is not preserved on output in the following conditions:
 - Output is a multi-part file.
 - Output is generated by a running environment that does not support sorting.

Basic Usage

```
sort order:LastName
```

Output: Dataset is sorted in alphabetically ascending order based on the values in the `LastName` column, assuming that the values are strings.

Syntax and Parameters

```
sort order:column_ref
```

| Token | Required? | Data Type | Description |
|-------|-----------|-----------|--|
| sort | Y | transform | Name of the transform |
| order | Y | string | Name of column or columns by which to sort |

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

order

Identifies the column or set of columns by which the dataset is sorted.

- Multiple column names can be separated by commas.
- Ranges of columns cannot be specified.

The order can be reversed by adding a negative sign in front of the column name:

```
sort order: -ProductName
```

Multi-column sorts: You can also specify multi-column sorts. The following example sorts first by the inverse order of `ProductName`, and within that sort, rows are sorted by `ProductColor`:

```
sort order: -ProductName,ProductColor
```

Sort by original row numbers: As an input value, this parameter also accepts the `SOURCEROWNUMBER` function, which performs the sort according to the original order of rows when the dataset was created.

```
sort order: SOURCEROWNUMBER()
```

See *SOURCEROWNUMBER Function*.

Usage Notes:

| Required? | Data Type |
|-----------|----------------------|
| Yes | String (column name) |

Data is sorted based on the data type of the source:

| Data Type of Source | Sort Order |
|---------------------|------------|
| Integer | Numerical |
| Decimal | Numerical |
| Datetime | Numerical |
| All others | String |

Examples

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

Example - sort methods

Source:

The column without a name identifies the original row numbers. In the data grid, this information is available when you hover over the black dot to the left of a row of data.

| | CustId | FirstName | LastName | City | State | LastOrder |
|---|--------|-----------|----------|---------------|-------|-----------|
| 1 | 1001 | Skip | Jones | San Francisco | CA | 25 |
| 2 | 1002 | Adam | Allen | Oakland | CA | 1099 |
| 3 | 1003 | David | Wiggins | Oakland | MI | 125.25 |
| 4 | 1004 | Amanda | Green | Detroit | MI | 452.5 |
| 5 | 1005 | Colonel | Mustard | Los Angeles | CA | 950 |
| 6 | 1006 | Pauline | Hall | Saginaw | MI | 432.22 |
| 7 | 1007 | Sarah | Miller | Cheyenne | WY | 724.22 |

| | | | | | | |
|---|------|--------|---------|------------|----|--------|
| 8 | 1008 | Teddy | Smith | Juneau | AK | 852.11 |
| 9 | 1009 | Joelle | Higgins | Sacramento | CA | 100 |

Transform:

First, you might want to clean up the number formatting in the `LastOrder` column. The following formats the values to always include two digits after the decimal point:

```
set col:LastOrder value:NUMFORMAT>LastOrder, '####.00')
```

Now, you're interested in the highest value for your customers' most recent orders. You can apply the following sort:

```
sort order: -LastOrder
```

Rows are sorted by the `LastOrder` column in descending order (largest to smallest):

| | CustId | FirstName | LastName | City | State | LastOrder |
|---|--------|-----------|----------|---------------|-------|-----------|
| 2 | 1002 | Adam | Allen | Oakland | CA | 1099.00 |
| 5 | 1005 | Colonel | Mustard | Los Angeles | CA | 950.00 |
| 8 | 1008 | Teddy | Smith | Juneau | AK | 852.11 |
| 7 | 1007 | Sarah | Miller | Cheyenne | WY | 724.22 |
| 4 | 1004 | Amanda | Green | Detroit | MI | 452.50 |
| 6 | 1006 | Pauline | Hall | Saginaw | MI | 432.22 |
| 3 | 1003 | David | Wiggins | Oakland | MI | 125.25 |
| 9 | 1009 | Joelle | Higgins | Sacramento | CA | 100.00 |
| 1 | 1001 | Skip | Jones | San Francisco | CA | 25.00 |

The above row numbers represent the original order of the rows. Now, you want to get your data geographically organized by sorting by city and state. You can perform multi-column sorts such as the following, which sorts first by `State` and then by `City` columns:

```
sort order: State, City
```

In the generated output, the data is first sorted by the `State` value. Each set of rows within the same `State` value is also sorted by the `City` value.

| | CustId | FirstName | LastName | City | State | LastOrder |
|---|--------|-----------|----------|---------------|-------|-----------|
| 8 | 1008 | Teddy | Smith | Juneau | AK | 852.11 |
| 5 | 1005 | Colonel | Mustard | Los Angeles | CA | 950.00 |
| 2 | 1002 | Adam | Allen | Oakland | CA | 1099.00 |
| 9 | 1009 | Joelle | Higgins | Sacramento | CA | 100.00 |
| 1 | 1001 | Skip | Jones | San Francisco | CA | 25.00 |
| 4 | 1004 | Amanda | Green | Detroit | MI | 452.50 |
| 3 | 1003 | David | Wiggins | Oakland | MI | 125.25 |
| 6 | 1006 | Pauline | Hall | Saginaw | MI | 432.22 |
| 7 | 1007 | Sarah | Miller | Cheyenne | WY | 724.22 |

Example - Sort by original row numbers

Source:

You have imported the following racer data on heat times from a CSV file. When loaded in the Transformer page, it looks like the following:

| (rowId) | column2 | column3 | column4 | column5 |
|---------|---------|---------|---------|---------|
| 1 | Racer | Heat 1 | Heat 2 | Heat 3 |
| 2 | Racer X | 37.22 | 38.22 | 37.61 |
| 3 | Racer Y | 41.33 | DQ | 38.04 |
| 4 | Racer Z | 39.27 | 39.04 | 38.85 |

In the above, the (rowId) column references the row numbers displayed in the data grid; it is not part of the dataset. This information is available when you hover over the black dot on the left side of the screen.

Transform:

You have examined the best performance in each heat according to the sample. You then notice that the data contains headers, but you forget how it was originally sorted. The data now looks like the following:

| (rowId) | column2 | column3 | column4 | column5 |
|---------|---------|---------|---------|---------|
| 1 | Racer Y | 41.33 | DQ | 38.04 |
| 2 | Racer | Heat 1 | Heat 2 | Heat 3 |
| 3 | Racer X | 37.22 | 38.22 | 37.61 |
| 4 | Racer Z | 39.27 | 39.04 | 38.85 |

While you can undo your sort steps to return to the original sort order, this approach works best if you did not include other steps in between that are based on the sort order.

If you have steps that require retaining your sort steps, you can revert to the original sort order by adding this transform step:

```
sort order:SOURCEROWNUMBER()
```

Then, you can create the header with the following simple step:

```
header
```

If you need to retain the sort order and not revert to the original, you can do the following to the previous example data:

```
header sourcerownumber:1
```

Results:

After you have applied the last header transform, your data should look like the following:

| (rowId) | Racer | Heat_1 | Heat_2 | Heat_3 |
|---------|---------|--------|--------|--------|
| 3 | Racer Y | 41.33 | DQ | 38.04 |
| 2 | Racer X | 37.22 | 38.22 | 37.61 |

| | | | | |
|---|---------|-------|-------|-------|
| 4 | Racer Z | 39.27 | 39.04 | 38.85 |
|---|---------|-------|-------|-------|

You can sort by the `Racer` column in ascending order to return to the original sort order.