

Keep Transform

NOTE: Transforms are a part of the underlying language, which is not directly accessible to users. This content is maintained for reference purposes only. For more information on the user-accessible equivalent to transforms, see *Transformation Reference*.

Retains a set of rows in your dataset, which are specified by the conditional in the `row` expression. All other rows are removed from the dataset.

The `keep` transform is the opposite of the `delete` transform. See *Delete Transform*.

Basic Usage

```
keep row:(customerStatus == 'active')
```

Output: For each row in the dataset, if the value of the `customerStatus` column is `active`, then the row is retained. Otherwise, the row is deleted from the dataset.

Syntax and Parameters

```
keep row:(expression)
```

Token	Required?	Data Type	Description
keep	Y	transform	Name of the transform
row	Y	string	Expression identifying the row or rows to keep. If expression evaluates to <code>true</code> for a row, the row is retained.

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

row

Expression to identify the row or rows on which to perform the transform. Expression must evaluate to `true` or `false`.

Examples:

Expression	Description
<code>Score >= 50</code>	<code>true</code> if the value in the <code>Score</code> column is greater than 50.
<code>LEN(LastName) > 8</code>	<code>true</code> if the length of the value in the <code>LastName</code> column is greater than 8.
<code>ISMISSING([Title])</code>	<code>true</code> if the row value in the <code>Title</code> column is missing.
<code>ISMISMATCHED(Score, ['Integer'])</code>	<code>true</code> if the row value in the <code>Score</code> column is mismatched against the <code>Integer</code> data type.

For the `keep` transform, if the expression for the `row` parameter evaluates to `true` for a row, it is kept in the dataset. Otherwise, it is removed.

Example:

```
keep row: (lastOrder >= 10000 && status == 'Active')
```

Output: Retains all rows in the dataset where the `lastOrder` value is greater than or equal to 10,000 and the customer status is `Active`.

Usage Notes:

Required?	Data Type
Yes	Expression that evaluates to <code>true</code> or <code>false</code>

Examples

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

Example - Remove old products and keep new orders

This examples illustrates how you can keep and delete rows from your dataset using the following transforms:

- `delete` - Deletes a set of rows as evaluated by the conditional expression in the `row` parameter. See *Delete Transform*.
- `keep` - Retains a set of rows as evaluated by the conditional expression in the `row` parameter. All other rows are deleted from the dataset. See *Keep Transform*.

Source:

Your dataset includes the following order information. You want to edit your dataset so that:

- All orders for products that are no longer available are removed. These include the following product IDs: P 100, P101, P102, P103.
- All orders that were placed within the last 90 days are retained.

OrderId	OrderDate	ProdId	ProductName	ProductColor	Qty	OrderValue
1001	6/14/2015	P100	Hat	Brown	1	90
1002	1/15/2016	P101	Hat	Black	2	180
1003	11/11/2015	P103	Sweater	Black	3	255
1004	8/6/2015	P105	Cardigan	Red	4	320
1005	7/29/2015	P103	Sweeter	Black	5	375
1006	12/1/2015	P102	Pants	White	6	420
1007	12/28/2015	P107	T-shirt	White	7	390
1008	1/15/2016	P105	Cardigan	Red	8	420
1009	1/31/2016	P108	Coat	Navy	9	495

Transformation:

First, you remove the orders for old products. Since the set of products is relatively small, you can start first by adding the following:

NOTE: Just preview this transformation. Do not add it to your recipe yet.

Transformation Name	Filter rows
Parameter: Condition	Custom formula
Parameter: Type of formula	Custom single
Parameter: Condition	(ProdId == 'P100')
Parameter: Action	Delete matching rows

When this step is previewed, you should notice that the top row in the above table is highlighted for removal. Notice how the transformation relies on the ProdId value. If you look at the ProductName value, you might notice that there is a misspelling in one of the affected rows, so that column is not a good one for comparison purposes.

You can add the other product IDs to the transformation in the following expansion of the transformation, in which any row that has a matching ProdId value is removed:

Transformation Name	Filter rows
Parameter: Condition	Custom formula
Parameter: Type of formula	Custom single
Parameter: Condition	(ProdId == 'P100' ProdId == 'P101' ProdId == 'P102' ProdId == 'P103')
Parameter: Action	Delete matching rows

When the above step is added to your recipe, you should see data that looks like the following:

OrderId	OrderDate	ProdId	ProductName	ProductColor	Qty	OrderValue
1004	8/6/2015	P105	Cardigan	Red	4	320
1007	12/28/2015	P107	T-shirt	White	7	390
1008	1/15/2016	P105	Cardigan	Red	8	420
1009	1/31/2016	P108	Coat	Navy	9	495

Now, you can filter out of the dataset orders that are older than 90 days. First, add a column with today's date:

Transformation Name	New formula
Parameter: Formula type	Single row formula

Parameter: Formula	'2/25/16'
Parameter: New column name	'today'

Keep the rows that are within 90 days of this date using the following:

Transformation Name	Filter rows
Parameter: Condition	Custom formula
Parameter: Type of formula	Custom single
Parameter: Condition	<code>datedif(OrderDate,today,day) <= 90</code>
Parameter: Action	Keep matching rows

Don't forget to delete the `today` column, which is no longer needed:

Transformation Name	Delete columns
Parameter: Columns	<code>today</code>
Parameter: Action	Delete selected columns

Results:

OrderId	OrderDate	ProdId	ProductName	ProductColor	Qty	OrderValue
1007	12/28/2015	P107	T-shirt	White	7	390
1008	1/15/2016	P105	Cardigan	Red	8	420
1009	1/31/2016	P108	Coat	Navy	9	495