PREV Function

Contents:

Extracts the value from a column that is a specified number of rows before the current value.

- The row from which to extract a value is determined by the order in which the rows are organized at the time that the transform is executed. If you are working on a randomly generated sample of your dataset, the values that you see for this function might not correspond to the values that are generated on the full dataset during job execution.
- If the previous value is missing or null, this function generates a missing value.
- You can use the `group` and `order` parameters to define the groups of records and the order of those records to which this transform is applied.
- This function works with the following transforms:
  - *Window Transform*
  - *Set Transform*
  - *Derive Transform*

Basic Usage

```
window value:PREV(myNumber, 1) order:Date
```

Output: Generates the new column, which contains the value in the row in the `myNumber` column immediately preceding the current row, when ordered by `Date`.

Syntax and Arguments

```
window value:PREV(col_ref, k_integer) order: order_col [group: group_col]
```

Argument	Required?	Data Type	Description
col_ref	Y	string	Name of column whose values are applied to the function
k_integer	Y	integer (positive)	Number of rows before the current one from which to extract the value

For more information on the `order` and `group` parameters, see *Window Transform*.

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

col_ref

Name of the column whose values are used to extract the value that is `k-integer` values before the current one.

- Multiple columns and wildcards are not supported.

Usage Notes:

Required?	Data Type	Example Value
Yes	String (column reference)	`myColumn`

k_integer

Integer representing the number of rows before the current one from which to extract the value.

- Value must be a positive integer. For negative values, see *NEXT Function*.
- `k=1` represents the immediately preceding row value.
- If k is greater than or equal to the number of values in the column, all values in the generated column are missing. If a `group` parameter is applied, then this parameter should be no more than the maximum number of rows in the groups.
- If the range provided to the function exceeds the limits of the dataset, then the function generates a null value.
- If the range of the function is valid but includes missing values, the function generates a missing, non-null value.

Usage Notes:

Required?	Data Type	Example Value
Yes	Integer	4

Examples

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

Example - Examine prior order history

The following dataset contains orders for multiple customers over a period of a few days, listed in no particular order. You want to assess how order size has changed for each customer over time and to provide offers to your customers based on changes in order volume.

Source:

Date	CustId	OrderId	OrderValue
1/4/16	C001	Ord002	500
1/11/16	C003	Ord005	200
1/20/16	C002	Ord007	300
1/21/16	C003	Ord008	400
1/4/16	C001	Ord001	100
1/7/16	C002	Ord003	600
1/8/16	C003	Ord004	700
1/21/16	C002	Ord009	200
1/15/16	C001	Ord006	900

Transform:

When the data is loaded into the Transformer page, you can use the PREV function to gather the order values for the previous two orders into a new column. The trick is to order the window transform by the date and group it by customer:

```
window value: PREV(OrderValue, 1) order: Date group: CustId
```

```
window value: PREV(OrderValue, 2) order: Date group: CustId
```

```
rename col: window to: 'OrderValue_1'
```

```
rename col: window1 to: 'OrderValue_2'
```

You should now have the following columns in your dataset: Date, CustId, OrderId, OrderValue, OrderValue_1, OrderValue_2.

The two new columns represent the previous order and the order before that, respectively. Now, each row contains the current order (OrderValue) as well as the previous orders. Now, you want to take the following customer actions:

- If the current order is more than 20% greater than the sum of the two previous orders, send a rebate.
- If the current order is less than 90% of the sum of the two previous orders, send a coupon.
- Otherwise, send a holiday card.

To address the first one, you might add the following, which uses the IF function to test the value of the current order compared to the previous ones:

```
derive type:single value: IF(OrderValue >= (1.2 * (OrderValue_1 + OrderValue_2)), 'send
rebate', 'no action') as: 'CustomerAction'
```

You can now see which customers are due a rebate. Now, edit the above and replace it with the following, which addresses the second condition. If neither condition is valid, then the result is send holiday card.

```
derive type:single value: IF(OrderValue >= (1.2 * (OrderValue_1 + OrderValue_2)), 'send
rebate', IF(OrderValue <= (1.2 * (OrderValue_1 + OrderValue_2)), 'send coupon', 'send
holiday card')) as: 'CustomerAction'
```

Results:

After you drop the OrderValue_1 and OrderValue_2 columns, your dataset should look like the following. Note that since the transforms with PREV functions grouped by CustId, the order of records has changed.

Date	CustId	OrderId	OrderValue	CustomerAction
1/4/16	C001	Ord001	100	send rebate
1/7/16	C001	Ord002	500	send rebate
1/15/16	C001	Ord006	900	send rebate
1/8/16	C003	Ord004	700	send rebate
1/11/16	C003	Ord005	200	send rebate
1/21/16	C003	Ord008	400	send coupon
1/7/16	C002	Ord003	600	send rebate
1/20/16	C002	Ord007	300	send rebate
1/21/16	C002	Ord009	200	send coupon