

ROLLINGKTHLARGEST Function

Contents:

- *Basic Usage*
 - *Syntax and Arguments*
 - *col_ref*
 - *k_integer*
 - *rowsBefore_integer, rowsAfter_integer*
 - *Examples*
 - *Example - ROLLINGKTHLARGEST functions*
-

Computes the rolling *k*th largest value forward or backward of the current row. Inputs can be Integer, Decimal, or Datetime.

KTHLARGEST extracts the ranked value from the values in a column, where $k=1$ returns the maximum value. The value for k must be between 1 and 1000, inclusive. For purposes of this calculation, two instances of the same value are treated as separate values. So, if your dataset contains three rows with column values 10, 9, and 9, then KTHLARGEST returns 9 for $k=2$ and $k=3$.

ROLLINGKTHLARGEST computes the KTHLARGEST value across a defined window of values within a column.

- If an input value is missing or null, it is not factored in the computation. For example, for the first row in the dataset, the rolling calculation of previous values is undefined.
- The row from which to extract a value is determined by the order in which the rows are organized based on the `order` parameter.
- 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.
- Inputs:
 - Required column name
 - Required *k*th value, which is a positive integer
 - Two optional integer parameters that determine the window backward and forward of the current row. The default integer parameter values are `-1` and `0`, which computes the rolling function from the current row back to the first row of the dataset.
- This function works with the following transforms:
 - *Window Transform*
 - *Set Transform*
 - *Derive Transform*

For more information on a non-rolling version of this function, see *KTHLARGEST Function*.

Wrangle vs. SQL: This function is part of Wrangle , a proprietary data transformation language. Wrangle is not SQL. For more information, see *Wrangle Language*.

Basic Usage

Column example:

```
rollingkthlargest(myCol, 2)
```

Output: Returns the rolling second largest of all values in the `myCol` column.

Rows before example:

```
rollingkthlargest(myNumber, 2, 3)
```

Output: Returns the rolling second largest value of the current row and the two previous row values in the `myNumber` column.

Rows before and after example:

```
rollingkthlargest(myNumber, 4, 3, 2)
```

Output: Returns the rolling fourth largest value of the two previous row values, the current row value, and the two rows after the current one in the `myNumber` column.

Syntax and Arguments

```
rollingkthlargest(col_ref, rowsBefore_integer, rowsAfter_integer) order: order_col [group: group_col]
```

Argument	Required?	Data Type	Description
<code>col_ref</code>	Y	string	Name of column whose values are applied to the function
<code>k_integer</code>	Y	integer (positive)	The ranking of the value to extract from the source column
<code>rowsBefore_integer</code>	N	integer	Number of rows before the current one to include in the computation
<code>rowsAfter_integer</code>	N	integer	Number of rows after the current one to include in the computation

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 compute the function. Inputs must be Integer, Decimal, or Datetime values.

NOTE: If the input is in Datetime type, the output is in unixtime format. You can wrap these outputs in the `DATEFORMAT` function to output the results in the appropriate Datetime format. See *DATEFORMAT Function*.

- Multiple columns and wildcards are not supported.

Usage Notes:

Required?	Data Type	Example Value
Yes	String (column reference to Integer or Decimal values)	<code>myColumn</code>

`k_integer`

Integer representing the ranking of the unique value to extract from the source column. Duplicate values are treated as separate values for purposes of this function's calculation.

NOTE: The value for k must be an integer between 1 and 1,000 inclusive.

- $k=1$ represents the maximum value in the column.
- If k is greater than or equal to the number of values in the column, the minimum value is returned.
- Missing and null values are not factored into the ranking of k .

Usage Notes:

Required?	Data Type	Example Value
Yes	Integer (positive)	4

rowsBefore_integer, rowsAfter_integer

Integers representing the number of rows before or after the current one from which to compute the rolling function, including the current row. For example, if the first value is 5, the current row and the four rows after it are used in the computation. Negative values for `rowsAfter_integer` compute the rolling function from rows preceding the current one.

- `rowBefore=1` generates the current row value only.
- `rowBefore=-1` uses all rows preceding the current one.
- If `rowsAfter` is not specified, then the value 0 is applied.
- If a `group` parameter is applied, then these parameter values should be no more than the maximum number of rows in the groups.

Usage Notes:

Required?	Data Type	Example Value
No	Integer	4

Examples

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

Example - ROLLINGKTHLARGEST functions

This example describes how to use the following rolling computational functions:

- `ROLLINGKTHLARGEST` - computes the k th largest value from a rolling window of rows before and after the current row. Duplicate values are treated as having the same k values. See *ROLLINGKTHLARGEST Function*.
- `ROLLINGKTHLARGESTUNIQUE` - computes the unique k th largest value from a rolling window of rows before and after the current row. Duplicate values are treated as having different k values. See *ROLLINGKTHLARGESTUNIQUE Function*.

The following dataset contains daily counts of server restarts across three servers over the preceding week. High server restart counts can indicate poor server health. In this example, you are interested in knowing for each server the rolling highest and second highest count of restarts per server over the previous week.

Source:

Date	Server	Restarts
2/21/18	s01	4
2/21/18	s02	0
2/21/18	s03	0
2/22/18	s01	4
2/22/18	s02	1
2/22/18	s03	2
2/23/18	s01	2
2/23/18	s02	3
2/23/18	s03	4
2/24/18	s01	1
2/24/18	s02	0
2/24/18	s03	2
2/25/18	s01	5
2/25/18	s02	0
2/25/18	s03	4
2/26/18	s01	1
2/26/18	s02	2
2/26/18	s03	1
2/27/18	s01	1
2/27/18	s02	2
2/27/18	s03	2

Transformation:

First, you want to maintain the row information as a separate column. Since data is ordered already by the Date column, you can use the following:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	ROWNUMBER()
Parameter: New column name	'entryId'

Use the following function to compute the rolling *k*th largest value of server restarts per server over the previous week. In this case, you can use the ROLLINGKTHLARGEST function, setting *k*=1. Uniqueness doesn't matter for calculating the highest value:

Transformation Name	New formula
Parameter: Formula type	Multiple row formula
Parameter: Formula	rollingkthlargest(Restarts, 1, 6, 0)

Parameter: Sort Rows by	Server
Parameter: Group Rows by	Server
Parameter: New column name	'rollingkthlargest_1'

Use the following function to compute the rolling second highest value. In this case, you can use ROLLINGKTHLARGESTUNIQUE:

Transformation Name	New formula
Parameter: Formula type	Multiple row formula
Parameter: Formula	rollingkthlargestunique(Restarts, 2, 6, 0)
Parameter: Sort Rows by	Server
Parameter: Group Rows by	Server
Parameter: New column name	'rollingKthLargestUnique_2'

Results:

entryId	Date	Server	Restarts	rollingKthLargestUnique_2	rollingkthlargest_Restarts
3	2/21/18	s02	0	0	0
6	2/22/18	s02	1	0	1
9	2/23/18	s02	3	1	3
12	2/24/18	s02	0	1	3
15	2/25/18	s02	0	1	3
18	2/26/18	s02	2	2	3
21	2/27/18	s02	2	2	3
4	2/21/18	s03	0	0	0
7	2/22/18	s03	2	0	2
10	2/23/18	s03	4	2	4
13	2/24/18	s03	2	2	4
16	2/25/18	s03	4	2	4
19	2/26/18	s03	1	2	4
22	2/27/18	s03	2	2	4
2	2/21/18	s01	4	4	4
5	2/22/18	s01	4	4	4
8	2/23/18	s01	2	2	4
11	2/24/18	s01	1	2	4
14	2/25/18	s01	5	4	5
17	2/26/18	s01	1	4	5
20	2/27/18	s01	1	4	5

