LISTMIN Function

Computes the minimum of all numeric values found in input array. Input can be an array literal, a column of arrays, or a function returning an array. Input values must be of Integer or Decimal type.

When this function is invoked, all of the values in the input array are passed to the corresponding columnar function. Some restrictions may apply. See *MIN Function*.

Basic Usage

Literal example:

```
derive type:single value:LISTMIN([0,0,2,4,6,8,10,12,14,16,18,20]) as:'myArrayMin'
```

Output: Generates an output column containing the minimum of all values in the literal array: `0`.

Column example:

```
derive type:single value:LISTMIN(myArray) as:'myArrayMin'
```

Output: Generates an output column containing the minimum of all values in the arrays of the `myArray` column.

Syntax and Arguments

```
derive type:single value:LISTMIN(array_ref)
```

Argument	Required?	Data Type	Description
array_ref	Y	Array	Array literal, reference to column containing arrays, or function returning an array

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

array_ref

Reference to an array can be an array literal, function returning an array, or a single column containing arrays.

- If the input is not a valid numeric array, null values are returned.
- Non-numerical values within an input array are not factored in the computation.
- Multiple columns and wildcards are not supported.

Usage Notes:

Required?	Data Type	Example Value
Yes	Array	`myArray`

Examples

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

Example - Math functions for lists (arrays)

This example describes how to generate random array (list) data and then to apply the following math functions to your arrays.

- `LISTSUM` - Sum all values in the array. See *LISTSUM Function*.
- `LISTMIN` - Minimum value of all values in the array. See *LISTMIN Function*.
- `LISTMAX` - Maximum value of all values in the array. See *LISTMAX Function*.
- `LISTAVERAGE` - Average value of all values in the array. See *LISTAVERAGE Function*.
- `LISTVAR` - Variance of all values in the array. See *LISTVAR Function*.
- `LISTSTDEV` - Standard deviation of all values in the array. See *LISTSTDEV Function*.
- `LISTMODE` - Most common value of all values in the array. See *LISTMODE Function*.

Source:

For this example, you can generate some randomized data using the following steps. First, you need to seed an array with a range of values using the RANGE function:

```
derive type: single value: RANGE(5, 50, 5) as: 'myArray1'
```

Then, unpack this array, so you can add a random factor:

```
unnest col: myArray1 keys: '[0]', '[1]', '[2]', '[3]', '[4]', '[5]', '[6]', '[7]', '[8]',
'[9]' pluck: true markLineage: true
```

Add the randomizing factor. Here, you are adding randomization around individual values: x-1 < x < x+4.

```
set col: myArray1_0~myArray1_8 value: IF(RAND() > 0.5, $col + (5 * RAND()), $col - RAND())
```

To make the numbers easier to manipulate, you can round them to two decimal places:

```
set col: myArray1_0~myArray1_8 value: ROUND($col, 2)
```

Renest these columns into an array:

```
nest col: myArray1_0, myArray1_1, myArray1_2, myArray1_3, myArray1_4, myArray1_5,
myArray1_6, myArray1_7, myArray1_8 into: array as: 'myArray2'
```

Delete the unused columns:

```
drop col: myArray1_0~myArray1_8,myArray1 action: Drop
```

Your data should look similar to the following:

myArray2
["8.29","9.63","14.63","19.63","24.63","29.63","34.63","39.63","44.63"]
["8.32","14.01","19.01","24.01","29.01","34.01","39.01","44.01","49.01"]
["4.55","9.58","14.58","19.58","24.58","29.58","34.58","39.58","44.58"]
["9.22","14.84","19.84","24.84","29.84","34.84","39.84","44.84","49.84"]
["8.75","13.36","18.36","23.36","28.36","33.36","38.36","43.36","48.36"]
["8.47","14.76","19.76","24.76","29.76","34.76","39.76","44.76","49.76"]
["4.93","9.99","14.99","19.99","24.99","29.99","34.99","39.99","44.99"]
["4.65","14.98","19.98","24.98","29.98","34.98","39.98","44.98","49.98"]
["7.80","14.62","19.62","24.62","29.62","34.62","39.62","44.62","49.62"]
["9.32","9.96","14.96","19.96","24.96","29.96","34.96","39.96","44.96"]

Transform:

These steps demonstrate the individual math functions that you can apply to your list data without unnesting it:

> **NOTE:** The NUMFORMAT function has been wrapped around each list function to account for any floating-point errors or additional digits in the results.

Sum of all values in the array (list):

```
derive type: single value: NUMFORMAT(LISTSUM(myArray2), '#.##') as: 'arraySum'
```

Minimum of all values in the array (list):

```
derive type: single value: NUMFORMAT(LISTMIN(myArray2), '#.##') as: 'arrayMin'
```

Maximum of all values in the array (list):

```
derive type: single value: NUMFORMAT(LISTMAX(myArray2), '#.##') as: 'arrayMax'
```

Average of all values in the array (list):

```
derive type: single value: NUMFORMAT(LISTAVERAGE(myArray2), '#.##') as: 'arrayAvg'
```

Variance of all values in the array (list):

```
derive type: single value: NUMFORMAT(LISTVAR(myArray2), '#.##') as: 'arrayVar'
```

Standard deviation of all values in the array (list):

```
derive type: single value: NUMFORMAT(LISTSTDEV(myArray2), '#.##') as: 'arrayStDv'
```

Mode (most common value) of all values in the array (list):

```
derive type: single value: NUMFORMAT(LISTMODE(myArray2), '#.##') as: 'arrayMode'
```

Results:

Results for the first four math functions:

myArray2	arrayAvg	arrayMax	arrayMin	arraySum
["8.29","9.63","14.63","19.63","24.63","29.63","34.63","39.63","44.63"]	25.04	44.63	8.29	225.33
["8.32","14.01","19.01","24.01","29.01","34.01","39.01","44.01","49.01"]	28.93	49.01	8.32	260.4
["4.55","9.58","14.58","19.58","24.58","29.58","34.58","39.58","44.58"]	24.58	44.58	4.55	221.19
["9.22","14.84","19.84","24.84","29.84","34.84","39.84","44.84","49.84"]	29.77	49.84	9.22	267.94
["8.75","13.36","18.36","23.36","28.36","33.36","38.36","43.36","48.36"]	28.4	48.36	8.75	255.63
["8.47","14.76","19.76","24.76","29.76","34.76","39.76","44.76","49.76"]	29.62	49.76	8.47	266.55
["4.93","9.99","14.99","19.99","24.99","29.99","34.99","39.99","44.99"]	24.98	44.99	4.93	224.85
["4.65","14.98","19.98","24.98","29.98","34.98","39.98","44.98","49.98"]	29.39	49.98	4.65	264.49

["7.80","14.62","19.62","24.62","29.62","34.62","39.62","44.62","49.62"]	29.42	49.62	7.8	264.76
["9.32","9.96","14.96","19.96","24.96","29.96","34.96","39.96","44.96"]	25.44	44.96	9.32	229

Results for the statistical functions:

myArray2	arrayMode	arrayStDv	arrayVar
["8.29","9.63","14.63","19.63","24.63","29.63","34.63","39.63","44.63"]		12.32	151.72
["8.32","14.01","19.01","24.01","29.01","34.01","39.01","44.01","49.01"]		13.03	169.78
["4.55","9.58","14.58","19.58","24.58","29.58","34.58","39.58","44.58"]		12.92	166.8
["9.22","14.84","19.84","24.84","29.84","34.84","39.84","44.84","49.84"]		13.02	169.46
["8.75","13.36","18.36","23.36","28.36","33.36","38.36","43.36","48.36"]		12.84	164.95
["8.47","14.76","19.76","24.76","29.76","34.76","39.76","44.76","49.76"]		13.14	172.56
["4.93","9.99","14.99","19.99","24.99","29.99","34.99","39.99","44.99"]		12.92	166.93
["4.65","14.98","19.98","24.98","29.98","34.98","39.98","44.98","49.98"]		13.9	193.16
["7.80","14.62","19.62","24.62","29.62","34.62","39.62","44.62","49.62"]		13.23	175.08
["9.32","9.96","14.96","19.96","24.96","29.96","34.96","39.96","44.96"]		12.21	149.17

Since all values are unique within an individual array, there is no most common value in any of them, which yields empty values for the `arrayMode` column.