GREATERTHAN Function

Contents:

Returns `true` if the first argument is greater than but not equal to the second argument. Equivalent to the > operator.

- Each argument can be a literal Integer or Decimal number, a function returning a number, or a reference to a column that contains numbers.

Since the function returns a Boolean value, it can be used as a function or a conditional.

> **NOTE:** Within an expression, you might choose to use the corresponding operator, instead of this function. For more information, see *Comparison Operators*.

Basic Usage

```
greaterthan(Errors, 10)
```

Output: Returns `true` when the value in the `Errors` column is greater than 10.

Syntax and Arguments

```
greaterthan(value1, value2)
```

Argument	Required?	Data Type	Description
value1	Y	string	The first value. This can be a number, a function returning a number, or a column containing numbers.
value2	Y	string	The second value. This can be a number, a function returning a number, or a column containing numbers.

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

value1, value2

Names of the column, expressions, or literals to compare.

- Missing values generate missing string results.

Usage Notes:

Required?	Data Type	Example Value
Yes	Column reference, function, or numeric or String value	`myColumn`

Examples

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

Example - Basic Comparison Functions

This simple example demonstrate available comparison functions:

- `LESSTHAN` - See *LESSTHAN Function*.
- `LESSTHANEQUAL` - See *LESSTHANEQUAL Function*.
- `EQUAL` - See *EQUAL Function*.
- `NOTEQUAL` - See *NOTEQUAL Function*.
- `GREATERTHAN` - See *GREATERTHAN Function*.
- `GREATERTHANEQUAL` - See *GREATERTHANEQUAL Function*.

Source:

colA	colB
1	11
2	10
3	9
4	8
5	7
6	6
7	5
8	4
9	3
10	2
11	1

Transformation:

Add the following transforms to your recipe, one for each comparison function:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`LESSTHAN(colA, colB)`
Parameter: New column name	`'lt'`

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	LESSTHANEQUAL(colA, colB)
Parameter: New column name	'lte'

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	EQUAL(colA, colB)
Parameter: New column name	'eq'

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	NOTEQUAL(colA, colB)
Parameter: New column name	'neq'

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	GREATERTHAN(colA, colB)
Parameter: New column name	'gt'

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	GREATERTHANEQUAL(colA, colB)
Parameter: New column name	'gte'

Results:

colA	colB	gte	gt	neq	eq	lte	lt
1	11	false	false	true	false	true	true
2	10	false	false	true	false	true	true
3	9	false	false	true	false	true	true
4	8	false	false	true	false	true	true
5	7	false	false	true	false	true	true
6	6	true	false	false	true	true	false
7	5	true	true	true	false	false	false
8	4	true	true	true	false	false	false

9	3	true	true	true	false	false	false
10	2	true	true	true	false	false	false
11	1	true	true	true	false	false	false

Example - Using Comparisons to Test Ranges

In the town of Circleville, citizens are allowed to maintain a single crop circle in their backyard, as long as it confirms to the town regulations. Below is some data on the size of crop circles in town, with a separate entry for each home. Limits are displayed in the adjacent columns, with the `inclusive` columns indicating whether the minimum or maximum values are inclusive.

> **Tip:** As part of this exercise, you can see how to you can extend your recipe to perform some simple financial analysis of the data.

Source:

Location	Radius_ft	minRadius_ft	minInclusive	maxRadius_ft	maxInclusive
House1	55.5	10	Y	25	N
House2	12	10	Y	25	N
House3	14.25	10	Y	25	N
House4	3.5	10	Y	25	N
House5	27	10	Y	25	N

Transformation:

After the data is loaded into the Transformer page, you can begin comparing column values:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`LESSTHANEQUAL(Radius_ft,minRadius_ft)`
Parameter: New column name	`'tooSmall'`

While accurate, the above transform does not account for the `minInclusive` value, which may be changed as part of your steps. Instead, you can delete the previous transform and use the following, which factors in the other column:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`IF(minInclusive == 'Y',LESSTHANEQUAL(Radius_ft,minRadius_ft), LESSTHAN(Radius_ft,minRadius_ft))`
Parameter: New column name	`'tooSmall'`

In this case, the `IF` function tests whether the minimum value is inclusive (values of `10` are allowed). If so, the `LESSTHANEQUAL` function is applied. Otherwise, the `LESSTHAN` function is applied. For the maximum limit, the following step applies:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`IF(maxInclusive == 'Y', GREATERTHANEQUAL(Radius_ft, maxRadius_ft),GREATERTHAN(Radius_ft,maxRadius_ft))`
Parameter: New column name	`'tooBig'`

Now, you can do some analysis of this data. First, you can insert a column containing the amount of the fine per foot above the maximum or below the minimum. Before the first `derive` command, insert the following, which is the fine (`$15.00`) for each foot above or below the limits:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`15`
Parameter: New column name	`'fineDollarsPerFt'`

At the end of the recipe, add the following new line, which calculates the fine for crop circles that are too small:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`IF(tooSmall == 'true', (minRadius_ft - Radius_ft) * fineDollarsPerFt, 0.0)`
Parameter: New column name	`'fine_Dollars'`

The above captures the too-small violations. To also capture the too-big violations, change the above to the following:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`IF(tooSmall == 'true', (minRadius_ft - Radius_ft) * fineDollarsPerFt, if(tooBig == 'true', (Radius_ft - maxRadius_ft) * fineDollarsPerFt, '0.0'))`
Parameter: New column name	`'fine_Dollars'`

In place of the original "false" expression (`0.0`), the above adds the test for the too-big values, so that all fines are included in a single column. You can reformat the `fine_Dollars` column to be in dollar format:

Transformation Name	`Edit column with formula`

Parameter: Columns	fine_Dollars
Parameter: Formula	NUMFORMAT(fine_Dollars, '$###.00')

Results:

After you delete the columns used in the calculation and move the remaining ones, you should end up with a dataset similar to the following:

Location	fineDollarsPerFt	Radius_ft	minRadius_ft	minInclusive	maxRadius_ft	maxInclusive	fineDollars
House1	15	55.5	10	Y	25	N	$457.50
House2	15	12	10	Y	25	N	$0.00
House3	15	14.25	10	Y	25	N	$0.00
House4	15	3.5	10	Y	25	N	$97.50
House5	15	27	10	Y	25	N	$30.00

Now that you have created all of the computations for generating these values, you can change values for `minRadius_ft` , `maxRadius_ft` , and `fineDollarsPerFt` to analyze the resulting fine revenue. Before or after the transform where you set the value for `fineDollarsPerFt`, you can insert something like the following:

Transformation Name	Edit column with formula
Parameter: Columns	minRadius_ft
Parameter: Formula	'12.5'

After the step is added, select the last line in the recipe. Then, you can see how the values in the `fineDollars` column have been updated.