IF Function

Contents:

The `IF` function allows you to build if/then/else conditional logic within your transforms.

> **NOTE:** The `IF` function is interchangeable with ternary operators. You should use this function instead of ternary construction.

> **NOTE:** If you are running your job on Spark, avoid creating single conditional transformations with deeply nested sets of conditions. On Spark, these jobs can time out, and deeply nested steps can be difficult to debug. Instead, break up your nesting into smaller conditional transformations of multiple steps.

Basic Usage

Example:

```
derive type:single value:IF(State == 'NY','New York, New York!','some other place')
as:'isNewYork'
```

Output: Generates a new `isNewYork` column, in which each row contains the value `New York, New York!` when the corresponding value in the `State` column is `NY`. Otherwise, the value in `isNewYork` is `some other place`.

Nested IF Example:

You can build `IF` statements within `IF` statements as in the following example, in which the second `IF` is evaluated if the first one evaluates to `false`:

```
derive type:single value:IF(State == 'NY',0.05,IF(State=='CA',0.08,0))
as:'CoTaxRatesByState'
```

A more detailed nested example is available below.

Syntax and Arguments

In the following, if the `test expression` evaluates to `true`, the `true_expression` is executed. Otherwise, the `false_expression` is executed.

```
IF(test_expression, true_expression,false_expression)
```

Argument	Required?	Data Type	Description

test_expression	Y	string	Expression that is evaluated. Must resolve to `true` or `false`
true_expression	Y	string	Expression that is executed if `test_expression` is `true`
false_expression	N	string	Expression that is executed if `test_expression` is `false`

All of these expressions can be constants (strings, integers, or any other supported literal value) or sophisticated elements of logic, although the test expression must evaluate to a Boolean value.

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

test_expression

This parameter contains the expression to evaluate. This expression must resolve to a Boolean (`true` or `false`) value.

Usage Notes:

Required?	Data Type	Example Value
Yes	String (expression that evaluates to `true` or `false`	`(LastName == 'Mouse' && FirstName == 'Mickey')`

true_expression, false_expression

The `true_expression` determines the value or conditional that is generated if the `test_expression` evaluates to `true`. If the test is `false`, then the `false_expression` applies.

These expressions typically generate output values and can use a combination of literals, functions, and column references.

- A true expression is required. You can insert a blank expression (`""`).
- If a false expression is not provided, false results yield a value of `false`.

Usage Notes:

Required?	Data Type	Example Value
Yes	String (expression)	See examples below.

Examples

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

Example - Basic Usage

Example data:

X	Y
true	true
true	false
false	true
false	false

Transforms:

```
derive type:single value:IF((X == Y), 'yes','no') as: 'equals'
```

Results:

Your output looks like the following:

X	Y	equals
true	true	yes
true	false	no
false	true	no
false	false	yes

Example - Stock Quotes

This example demonstrates how you can chain together multiple if/then/else conditions within a single transform step.

You have a set of stock prices that you want to analyze. Based on a set of rules, you want to determine any buy, sell, or hold action to take.

Source:

Ticket	Qty	BuyPrice	CurrentPrice
GOOG	10	705.25	674.5
FB	100	84.00	101.125
AAPL	50	125.25	97.375
MSFT	100	38.875	45.25

Transform:

You can perform evaluations of this data using the `IF` function to determine if you want to take action.

> **NOTE:** For a larger dataset, you might maintain your buy, sell, and hold evaluations for each stock in a separate reference dataset that you join to the source dataset before performing comparisons between column values. See *Join Panel*.

To assist in evaluation, you might first want to create columns that contain the cost (`Basis`) and the current value (`CurrentValue`) for each stock:

```
derive type:single value:(Qty * BuyPrice) as:'Basis'
```

```
derive type:single value:(Qty * CurrentPrice) as:'CurrentValue'
```

Now, you can build some rules based on the spread between `Basis` and `CurrentValue`.

Single `IF` version: In this case, the most important action is determining if it is time to sell. The following rule writes a `sell` notification if the current value is $1000 or more than the cost. Otherwise, no value is written to the action column.

```
derive type:single value: IF((CurrentValue - 1000 > Basis), 'sell','') as:'action'
```

Nested `IF` version: But what about buying more? The following transform is an edit to the previous one. In this new version, the sell test is performed, and if `false`, writes a `buy` action if the `CurrentPrice` is within 10% of the `BuyPrice`.

This second evaluation is performed after the first one, as it replaces the else clause, which did nothing in the previous version. In the Recipe Panel, click the previous transform and edit it, replacing it with the new version:

```
derive type:single value: IF((CurrentValue - 1000 > Basis), 'sell', IF((abs(CurrentValue -
Basis) <= (Basis * 0.1)),'buy','hold')) as:'action'
```

If neither test evaluates to `true`, the written action is `hold`.

You might want to format some of your columns using dollar formatting, as in the following:

> **NOTE:** The following formatting inserts a dollar sign ($) in front of the value, which changes the data type to String.

```
set col:BuyPrice value:NUMFORMAT(BuyPrice, '$ ##,###.00')
```

Results:

After moving your columns, your dataset should look like the following, if you completed the number formatting steps:

Ticket	Qty	BuyPrice	CurrentPrice	Basis	CurrentValue	action
GOOG	10	705.25	$ 674.50	$ 7,052.50	$ 6,745.00	buy
FB	100	84.00	$ 101.13	$ 8,400.00	$ 10,112.50	sell
AAPL	50	125.25	$ 97.38	$ 6,262.50	$ 4,868.75	hold
MSFT	100	38.88	$ 45.25	$ 3,887.50	$ 4,525.00	hold