CASE Function

Contents:

The CASE function allows you to perform multiple conditional tests on a set of expressions within a single statement. When a test evaluates to `true`, a corresponding output is generated. Outputs may be a literal or expression.For more information on the `IF` function, see *IF Function*.

Basic Usage

Example:

```
derive type:single value:CASE([ Qty <= 10, 'low_qty', Qty >=100, 'high_qty', 'med_qty'])
as:'Qty_Eval'
```

Output: Generates a new `Qty_Eval` column, in each row contains a text string based on the evaluation of the `Qty` column:

- Qty <= 10 : `low_qty`
- Qty >= 100 : `high_qty`
- All other values of Qty : `med_qty`

Syntax and Arguments

In the following, If the `testX` expression evaluates to `true`, then the `resultX` value is the output.

- Test expressions are evaluated in the listed order.
- Text expressions and results are paired values in an array.
- You must include one or more test expressions.
- Each test must include a result expression. Result expression can be a literal value or an expression that evaluates to a value of a supported data type.
- If a quoted value is included as a test expression, it is evaluated as the value to write for all values that have not yet matched a test (else expression).

```
CASE([test1, 'result1',test2, 'result2', testn, 'resultn','result_else'])
```

Argument	Required?	Data Type	Description
test1, test2, testn	Y	expression	Expression that is evaluated. Must resolve to `true` or `false`
result1, result2, result2, result_else	Y	string	Quoted string that is written if the corresponding test expression evaluates to `true`.

All of these expressions can be constants (strings, integers, or any other supported literal values) 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*.

test1, test2, testn

These parameters contain the expressions to evaluate. This expression must resolve to a Boolean (`true` or `false`) value.

> **NOTE:** The syntax of a test expression follows the same syntax as the `IF` function. For example, you must use the double-equals signs to compare values (`status == 'Ok'`).

Usage Notes:

Required?	Data Type	Example Value
Yes	Expression that evaluates to `true` or `false`	`(OrderAge > 90)`

result1, result2, result2, result_else

If the corresponding test expression evaluates to `true`, this value is written as the result.

These expressions can literals of any data type or expressions that evaluate to literals of any data type.

Usage Notes:

Required?	Data Type	Example Value
Yes	Literal value or expression	See examples below.

Examples

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

Example - Basic Usage

The following data represents orders received during the week. Discounts are applied to the orders based on the following rules:

- The standard discount is 5%.
- If an order is for fewer less than 10 units, then the discount is reduced by 5%.
- If an order is for more than 20 units, then the discount is increased by 5%.
- The special Friday discount is 2% more than the standard discount.

OrdDate	CustId	Qty	Std_Disc
5/8/17	C001	4	0.05
5/9/17	C002	11	0.05
5/10/17	C003	4	0.05
5/11/17	C001	25	0.05
5/12/17	C002	19	0.05

Transforms:

To determine the day of the week, you can use the following transform:

```
derive type:single value:WEEKDAY(Date)
```

You can build the discount rules into the following transform, which generates the `Disc` column:

```
derive type:single value:CASE([Qty<10, Std_Disc - 0.05, Qty>=20, Std_Disc + 0.05,
weekday_Date == 5, Std_Disc + 0.02, Std_Disc]) as:'Disc'
```

Results:

OrdDate	CustId	Qty	Std_Disc	Disc
5/8/17	C001	4	0.05	0
5/9/17	C002	11	0.05	0.05
5/10/17	C003	4	0.05	0
5/11/17	C001	25	0.05	0.1
5/12/17	C002	19	0.05	0.07