Apply Conditional Transformations

Contents:

In your recipe steps, you can apply conditional logic to determine if transformational changes should occur. You can build logical tests into your transformations in multiple levels:

- **Single- and multi-case transformations:** Use case-based transformations to test if-then or case logic against your dataset and to apply the specified results.
- **Conditional functions:** IF and CASE functions can be applied to any transformation that accepts functional expressions.
- **Logical operators:** You can use AND or OR logic to build your conditional expressions.

> **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.

Single- and Multi-Case Transformations

Through the Transform Builder, you can build conditional tests using if/then/else or case logic to manipulate on the data.

1. In the Search panel in the Transformer page, enter `case`.
2. You can choose one of three different logical transformations:
   a. **If-then-else:** Specify any logical test that evaluates to `true` or `false` and specify values if `true` (then) or if `false` (else).
   b. **Single-column case:** Test for explicit values in a column and, if true, write specific values to the new column.
   c. **Custom conditions:** Specify any number of case statements, which can have completely independent expressions:
      i. Case 1 is tested, and a value is written if `true`.
      ii. If Case 1 is false, then Case 2 is tested. If `true`, a different value can be written.
      iii. Supports an arbitrary number of independent conditional cases.
3. Specify the other parameters, including the name of the new column.

After the transformation is added to the recipe, actions can then be taken based on the values in this new column.

Conditional Functions

You can also apply conditional logical as part of your function definitions for other transformations.

IF function

For example, the following replaces values in the same column with `IN` if they are greater than 0.5 or `OUT` otherwise:

Transformation Name	`Edit column with formula`
Parameter: Columns	`testCol`
Parameter: Formula	`IF($col >= 0.5, 'IN','OUT')`

In the above, the token `$col` is a reference back to the value defined for the column (`testCol` in this case). However, you can replace it with a reference to any column in the dataset.

You can use the IF function in any transformation that accepts functional inputs. For more information, see *IF Function*.

CASE function

You can chain together IF functions in the following manner:

Transformation Name	`Edit column with formula`
Parameter: Columns	`testCol`
Parameter: Formula	`IF($col >= 0.5, 'IN',(IF($col >= 0.35, 'MAYBE IN','OUT')))`

However, these can become problematic to debug. Instead, you can use the CASE function to assist in building more complex logical trees. The following is more legible and easier to manage:

Transformation Name	`Edit column with formula`
Parameter: Columns	`testCol`
Parameter: Formula	`CASE([ $col >= 0.75, 'IN', $col >= 0.35, 'MAYBE IN', 'OUT'])`

If test	**Test**	**Output if `true`**
If:	`$col >= 0.75`	`IN`
If above is `false`:	`$col >= 0.35`	`MAYBE IN`
If above is `false`:	default	`OUT`

For more information, see *CASE Function*.

Logical Operators

Logical operators can be applied to your function expressions to expand the range of your logical tests.

In the above example, suppose you have a second column called, `Paid`, which contains Boolean values. You could expand the previous statement to include a test to see if `Paid=true`:

Transformation Name	`Edit column with formula`
Parameter: Columns	`testCol`
Parameter: Formula	`CASE([ ($col >= 0.75 && Paid == true), 'IN', ($col >= 0.35 && Paid == true), 'MAYBE IN', 'OUT'])`

The above performs a logical AND operation on the two expressions in each tested case. The logical operator is `&&`.

You can also reference explicit functions to perform logical tests. The above might be replaced with the following:

Transformation Name	`Edit column with formula`
Parameter: Columns	`testCol`
Parameter: Formula	`CASE([ AND($col >= 0.75, Paid == true), 'IN', AND($col >= 0.35, Paid == true), 'MAYBE IN', 'OUT'])`

Logic	Logical Operator	Logical Function
Logical AND	`(exp1 && exp2)`	`AND(exp1,exp2)`
Logical OR	`(exp1 || exp2)`	`OR(exp1,exp2)`
Logical NOT	`!(exp1 == exp2)`	`NOT(exp1,exp2)`

Depending on the structure of your transformation and your preferences, either form may be used.

- For more information, see *Logical Operators*.
- For more information, see *Logical Functions*.