RIGHT Function

Contents:

Matches the right set of characters in a string, as specified by parameter. The string can be specified as a column reference or a string literal.

- Since the `RIGHT` function matches based on fixed numeric values, changes to the length or structure of a data field can cause your recipe to fail to properly execute.
- The `RIGHT` function requires an integer value for the number of characters to match. If you need to match strings using patterns, you should use the `ENDSWITH` transform instead. See *ENDSWITH Function*.

Basic Usage

Column reference example:

```
derive type:single value:RIGHT(MyString,3)
```

Output: The rightmost (last) three letters of the `MyName` column value are written to the new column.

String literal example:

```
derive type:single value:RIGHT('Hello, World',5)
```

Output: The string `World` is written to the new column.

Syntax and Arguments

```
derive type:single value:RIGHT(column_string,end_count)
```

Argument	Required?	Data Type	Description
column_string	Y	string	Name of the column or string literal to be applied to the function
end_count	Y	integer (positive)	Count of characters from the end of the source string to apply to the match

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

column_string

Name of the column or string constant to be searched.

- Missing string or column values generate missing string results.
- String constants must be quoted (`'Hello, World'`).
- Multiple columns and wildcards are not supported.

Usage Notes:

Required?	Data Type	Example Value
Yes	String literal or column reference	`myColumn`

end_count

Count of characters from the right end of the string to include in the match.

- Value must a non-negative integer. If the value is `0`, then the match fails for all strings.
- If this value is greater than the length of the string, then the match is the entire string.
- References to columns of integer data type are not supported.

Usage Notes:

Required?	Data Type	Example Value
Yes	Integer (non-negative)	5

Examples

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

Example - Parse segments of social security numbers

Social security numbers follow a regular format:

```
XXX-XX-XXXX
```

Each of the separate numeric groups corresponds to a specific meaning:

- XXX - Area value that corresponds to a geographic location that surrounds the SSN applicant's address
- XX - Group number identifies the order in which the numbers are assigned within an area
- XXX - Serial number of the individual within the area and group groupings.
- For more information, see *http://www.usrecordsearch.com/ssn.htm*.

Source:

You want to analyze some social security numbers for area, group, and serial information. However, your social security number data is messy:

> **NOTE:** The following sample contains invalid social security numbers for privacy reasons. If you use this data in the application, it fails validation for the SSN data type.

ParticipantId	SocialNum
1001	805-88-2013
1002	845221914
1003	865 22 9291
1004	892-732213

Transform:

When the above data is imported, the `SocialNum` column might or might not be inferred as SSN data type. Either way, you should clean up your data, using the following transforms:

```
replace col: SocialNum on: '-' with: '' global: true
```

```
replace col: SocialNum on: ' ' with: '' global: true
```

At this point, your `SocialNum` data should be inferred as SSN type and consistently formatted as a set of digits:

ParticipantId	SocialNum
1001	805882013
1002	845221914
1003	865229291
1004	892732213

From this more consistent data, you can now break out the area, group, and serial values from the column:

```
derive type:single value: LEFT(SocialNum, 3) as: 'SSN_area'
```

```
derive type:single value: SUBSTRING(SocialNum, 3,5) as: 'SSN_group'
```

```
derive type:single value: RIGHT(SocialNum, 4) as: 'SSN_serial'
```

If desired, you can re-order the three new columns and delete the source column:

```
move col: SSN_serial after: SSN_area
```

```
move col: SSN_group after: SSN_area
```

```
drop col:SocialNum
```

Results:

If you complete the previous transform steps, your data should look like the following:

ParticipantId	SSN_area	SSN_group	SSN_serial
1001	805	88	2013
1002	845	22	1914
1003	865	22	9291
1004	892	73	2213