LEN Function

Returns the number of characters in a specified string. String value can be a column reference or string literal.

Basic Usage

Column reference example:

```
derive type:single value:LEN(MyName)
```

Output: The number of characters in the value in column `MyName` is written to a new column.

String literal example:

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

Output: The value `12` is written to the new column.

Syntax and Arguments

```
derive type:single value:LEN(column_string)
```

Argument	Required?	Data Type	Description
column_string	Y	string	Name of the column or string literal to be applied to the function

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`

Examples

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

Example - Fixed Length Strings

Source:

Your product identifiers follow a specific structure that you'd like to validate in your recipe. In the following example data, the `productId` column should contain values of length `6`.

You can see that there is already a column containing validation errors for the `ProductName` column. Values in the `ProductId` column that are not this length should be flagged in a new column. Then, you should merge the two columns together to create a `ValidationError` column.

ProductName	ProductId	ErrProductName
Chocolate Bunnie	123456	Error-ProductName
Chocolate Squirl	88442286	Error-ProductName
Chocolate Gopher	12345	

Transform:

To validate the length of the values in `ProductId`, enter the following transform. Note that the `as` parameter enables you to rename the column as part of the transform.

```
derive type:single value: IF(LEN(ProductId) <> 6, 'Error-length-ProductId','') '' as:
'ErrProductIdLength'
```

The dataset now looks like the following:

ProductName	ProductId	ErrProductName	ErrProductIdLength
Chocolate Bunnie	123456	Error-ProductName	
Chocolate Squirrel	88442286	Error-ProductName	Error-length-ProductId
Chocolate Gopher	12345		Error-length-ProductId

You can blend the two error columns into a single `DataValidationErrors` error column using the following `merge` transform. Note again the use of the `as` parameter:

```
merge col:ErrProductName,ErrProductIdlength with:' ' as:'DataValidationErrors'
```

To clean up the data, you might want to do the following, which trims out the whitespace in the `DataValidationErrors` column and removes the two individual error columns:

```
set col:DataValidationErrors value:TRIM(DataValidationErrors)
```

```
drop col:ErrProductName
```

```
drop col:ErrProductIdLength
```

Results:

The final dataset should look like the following:

ProductName	ProductId	DataValidationErrors
Chocolate Bunnie	123456	Error-ProductName
Chocolate Squirrel	88442286	Error-ProductName Error-length-ProductId
Chocolate Gopher	12345	Error-length-ProductId