LEN Function

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

Wrangle vs. SQL: This function is part of Wrangle , a proprietary data transformation language. Wrangle is not SQL. For more information, see *Wrangle Language*.

Basic Usage

Column reference example:

```
len(MyName)
```

Output: Returns the number of characters in the value in column `MyName`.

String literal example:

```
len('Hello, World')
```

Output: Returns the value `12`.

Syntax and Arguments

```
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	

Transformation:

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.

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`if(len(ProductId) <> 6, 'Error-length-ProductId','')`
Parameter: New column name	`'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:

Transformation Name	`Merge columns`
Parameter: Columns	`ErrProductName,ErrProductIdlength`
Parameter: Separator	`''`
Parameter: New column name	`'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:

Transformation Name	`Edit column with formula`
Parameter: Columns	`DataValidationErrors`

Parameter: Formula	`trim(DataValidationErrors)`

Transformation Name	`Delete columns`
Parameter: Columns	`ErrProductName,ErrProductIdLength`
Parameter: Action	`Delete selected columns`

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