NULL Function

The `NULL` function generates null values.

- The `ISNULL` function tests for the presence of null values. See *ISNULL Function*.
- Null values are different from missing values.
  - To test for missing values, see *ISMISSING Function*.

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

```
null()
```

Output: Returns a null value.

```
if((isnull(FirstName) || isnull(LastName)), null(), 'not null') as:'status'
```

Output: If there are null values in either the `FirstName` or `LastName` column, generate a null value in the `status` column. Otherwise, the returned value is `not null`.

Syntax and Arguments

There are no arguments for this function.

Examples

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

Example - Type check functions

This example illustrates how various type checking functions can be applied to your data.

Functions:

Item	Description
VALID Function	Tests whether a set of values is valid for a specified data type and is not a null value.
ISMISMATCHED Function	Tests whether a set of values is not valid for a specified data type.
ISMISSING Function	The `ISMISSING` function tests whether a column of values is missing or null. For input column references, this function returns `true` or `false`.
ISNULL Function	The `ISNULL` function tests whether a column of values contains null values. For input column references, this function returns `true` or `false`.
NULL Function	The `NULL` function generates null values.

Source:

Some source values that should match the State and Integer data types:

State	Qty

CA	10
OR	-10
WA	2.5
ZZ	15
ID	
	4

Transformation:

Invalid State values: You can test for invalid values for State using the following:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`ISMISMATCHED (State, 'State')`

The above transform flags rows 4 and 6 as mismatched.

> **NOTE:** A missing value is not valid for a type, including String type.

Invalid Integer values: You can test for valid matches for Qty using the following:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`(ISVALID (Qty, 'Integer') && (Qty > 0))`
Parameter: New column name	`'valid_Qty'`

The above transform flags as valid all rows where the `Qty` column is a valid integer that is greater than zero.

Missing values: The following transform tests for the presence of missing values in either column:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`(ISMISSING(State) || ISMISSING(Qty))`
Parameter: New column name	`'missing_State_Qty'`

After re-organizing the columns using the `move` transform, the dataset should now look like the following:

State	Qty	mismatched_State	valid_Qty	missing_State_Qty
CA	10	false	true	false
OR	-10	false	false	false
WA	2.5	false	false	false

ZZ	15	true	true	false
ID		false	false	true
	4	false	true	true

Since the data does not contain null values, the following transform generates null values based on the preceding criteria:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`((mismatched_State == 'true') || (valid_Qty == 'false') || (missing_State_Qty == 'true')) ? NULL() : 'ok'`
Parameter: New column name	`'status'`

You can then use the `ISNULL` check to remove the rows that fail the above test:

Transformation Name	`Filter rows`
Parameter: Condition	`Custom formula`
Parameter: Type of formula	`Custom single`
Parameter: Condition	`ISNULL('status')`
Parameter: Action	`Delete matching rows`

Results:

Based on the above tests, the output dataset contains one row:

State	Qty	mismatched_State	valid_Qty	missing_State_Qty	status
CA	10	false	true	false	ok