

# 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*.
- For more information on null values, see *Manage Null Values*.

**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.

- `ISVALID` - Returns `true` if the input matches the specified data type. See *VALID Function*.
- `ISMISMATCHED` - Returns `true` if the input does not match the specified data type. See *ISMISMATCHED Function*.
- `ISMISSING` - Returns `true` if the input value is missing. See *ISMISSING Function*.
- `ISNULL` - Returns `true` if the input value is null. See *ISNULL Function*.
- `NULL` - Generates a null value. See *NULL Function*.

## 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:

<b>Transformation Name</b>	New formula
<b>Parameter: Formula type</b>	Single row formula
<b>Parameter: Formula</b>	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:

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

<b>Transformation Name</b>	New formula
<b>Parameter: Formula type</b>	Single row formula
<b>Parameter: Formula</b>	(ISMISSING(State)    ISMISSING(Qty))
<b>Parameter: New column name</b>	'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:

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

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

<b>Transformation Name</b>	Filter rows
<b>Parameter: Condition</b>	Custom formula
<b>Parameter: Type of formula</b>	Custom single
<b>Parameter: Condition</b>	ISNULL('status')
<b>Parameter: Action</b>	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