

# Set Transform

## Contents:

- *Basic Usage*
- *Syntax and Parameters*
  - *col1, col2*
  - *value*
  - *group*
- *Examples*
  - *Example - Clean up marketing contact data with replace, set, and extract*
  - *Example - Using \$col placeholder*

**NOTE:** Transforms are a part of the underlying language, which is not directly accessible to users. This content is maintained for reference purposes only. For more information on the user-accessible equivalent to transforms, see *Transformation Reference*.

Replaces values in the specified column or columns with the specified value, which can be a literal or an expression. Expressions can use conditional functions to filter the set of rows.

The `set` transform is used to replace entire cell values. For replacement of partial cell values using literals or patterns, use the `replace` transform. See *Replace Transform*.

## Basic Usage

### Literal example:

```
set col: Country value: 'USA'
```

**Output:** Sets the values of all rows in the `Country` column to `USA`.

### Multi-column Literal example:

```
set col: SSN,Phone value: '##REDACTED###'
```

**Output:** Sets the values of all rows in the `SSN` and `Phone` columns to `##REDACTED##`.

### Expression example:

```
set col: isAmerica value: IF(Country == 'USA', true, 'false')
```

**Output:** If the value in the `Country` column is `USA`, then the value in `isAmerica` is set to `true`.

### Placeholder example:

You can substitute a placeholder value for the column name, which is useful if you are applying the same function across multiple columns. For example:

```
set col:score1,score2 value:IF ($col == 0, AVERAGE($col), $col)
```

**Output:** In the above transform, the values in `score1` and `score2` are set to the average of the column value when the value in the column is 0. Note that the computation of average is applied across all rows in the column, instead of just the filtered rows.

**Window function example:**

You can use window functions in your `set` transforms:

```
set col: avgSales value: ROLLINGAVERAGE(POS_Sales, 7, 0) group: saleDate order: saleDate
```

**Output:** Calculate the value in the column of `avgSales` to be the rolling average of the `POS_Sales` values for the preceding seven days, grouped and ordered by the `saleDate` column. For more information, see *Window Functions*.

**Syntax and Parameters**

```
set col:col1,[col2] value:(expression) [group: group_col]
```

Token	Required?	Data Type	Description
set	Y	transform	Name of the transform
col1	Y	string	Column name
col2	N	string	Column name
value	Y	string	Expression that generates the value to store in the column
group	N	string	If you are using aggregate or window functions, you can specify a <code>group</code> expression to identify the subset of records to apply the <code>value</code> expression.

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

**col1, col2**

Identifies the column and optional additional columns to which to apply the transform.

```
set col: MyCol value: 'myNewString'
```

**Output:** Sets value in `MyCol` column to `myNewString`.

**Usage Notes:**

Required?	Data Type
Yes	String (column name)

**value**

Identifies the expression that is applied by the transform. The `value` parameter can be one of the following types:

- test predicates that evaluate to Boolean values (`value: myAge == '30'` yields a `true` or `false` value), or
- computational expressions (`value: abs(pow(myCol,3))`).

The expected type of `value` expression is determined by the transform type. Each type of expression can contain combinations of the following:

- **literal values:** value: 'Hello, world'
- **column references:** value: amountOwed \* 10
- **functions:** value: left(myString, 4)
- **combinations:** value: abs(pow(myCol,3))

The types of any generated values are re-inferred by the platform.

**Usage Notes:**

Required?	Data Type
Yes	String (literal, column name, or expression)

**group**

Identifies the column by which the dataset is grouped for purposes of applying the transform.

**NOTE:** Transforms that use the `group` parameter can result in non-deterministic re-ordering in the data grid. However, you should apply the `group` parameter, particularly on larger datasets, or your job may run out of memory and fail. To enforce row ordering, you can use the `sort` transform. For more information, see *Sort Transform*.

If the value parameter contains aggregate or window functions, you can apply the `group` parameter to specify subsets of records across which the value computation is applied.

You can specify one or more columns by which to group using comma-separated column references.

**Usage Notes:**

Required?	Data Type
No	String (column name)

**Examples**

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

**Example - Clean up marketing contact data with replace, set, and extract**

This example illustrates the different uses of the following transformations to replace or extract cell data:

- `set` - defines the values to use in a predefined column. See *Set Transform*.

**Tip:** Use the `derive` transform to generate a new column containing a defined set of values. See *Derive Transform*.

- `replace` - replaces a string literal or pattern appearing in the values of a column with a specific string. See *Replace Transform*.
- `extract` - extracts a pattern-based value from a column and stores it in a new column. See *Extract Transform*.

**Source:**

The following dataset contains contact information that has been gathered by your marketing platform from actions taken by visitors on your website. You must clean up this data and prepare it for use in an analytics platform.

LeadId	LastName	FirstName	Title	Phone	Request
LE160301001	Jones	Charles	Chief Technical Officer	415-555-1212	reg
LE160301002	Lyons	Edward		415-012-3456	download whitepaper
LE160301003	Martin	Mary	CEO	510-555-5555	delete account
LE160301004	Smith	Talia	Engineer	510-123-4567	free trial

**Transformation:**

**Title column:** For example, you first notice that some data is missing. Your analytics platform recognizes the string value, "#MISSING#" as an indicator of a missing value. So, you click the missing values bar in the Title column. Then, you select the Replace suggestion card. Note that the default replacement is a null value, so you click **Edit** and update it:

<b>Transformation Name</b>	Edit column with formula
<b>Parameter: Columns</b>	Title
<b>Parameter: Formula</b>	if(ismissing([Title]), '#MISSING#', Title)

**Request column:** In the Request column, you notice that the `reg` entry should be cleaned up. Add the following transformation, which replaces that value:

<b>Transformation Name</b>	Replace text or pattern
<b>Parameter: Column</b>	Request
<b>Parameter: Find</b>	`{start}reg{end}`
<b>Parameter: Replace with</b>	Registration

The above transformation uses a Pattern as the expression of the `on` parameter. This expression indicates to match from the start of the cell value, the string literal `reg`, and then the end of the cell value, which matches on complete cell values of `reg` only.

This transformation works great on the sample, but what happens if the value is `Reg` with a capital `R`? That value might not be replaced. To improve the transformation, you can modify the transformation with the following Pattern in the `on` parameter, which captures differences in capitalization:

<b>Transformation Name</b>	Replace text or pattern
<b>Parameter: Column</b>	Request
<b>Parameter: Find</b>	`{start}{{[R r]}eg{end}`
<b>Parameter: Replace with</b>	'Registration'

Add the above transformation to your recipe. Then, it occurs to you that all of the values in the `Request` column should be capitalized in title or proper case:

<b>Transformation Name</b>	Edit column with formula
<b>Parameter: Columns</b>	Request
<b>Parameter: Formula</b>	proper(Request)

Now, all values are capitalized as titles.

**Phone column:** You might have noticed some issues with the values in the `Phone` column. In the United States, the prefix 555 is only used for gathering information; these are invalid phone numbers.

In the data grid, you select the first instance of 555 in the column. However, it selects all instances of that pattern, including ones that you don't want to modify. In this case, continue your selection by selecting the similar instance of 555 in the other row. In the suggestion cards, you click the Replace Text or Pattern transformation.

Notice, however, that the default Replace Text or Pattern transformation has also highlighted the second 555 pattern in one instance, which could be a problem in other phone numbers not displayed in the sample. You must modify the selection pattern for this transformation. In the `on:` parameter below, the `Pattern` has been modified to match only the instances of 555 that appear in the second segment in the phone number format:

<b>Transformation Name</b>	Replace text or pattern
<b>Parameter: Column</b>	Phone
<b>Parameter: Find</b>	`{start}%{3}-555-%*{end}`
<b>Parameter: Replace with</b>	'#INVALID#'
<b>Parameter: Match all occurrences</b>	true

Note the wildcard construct has been added (`%*`). While it might be possible to add a pattern that matches on the last four characters exactly (`%{4}`), that matching pattern would not capture the possibility of a phone number having an extension at the end of it. The above expression does.

**NOTE:** The above transformation creates values that are mismatched with the Phone Number data type. In this example, however, these mismatches are understood to be for the benefit of the system consuming your Trifacta output.

**LeadId column:** You might have noticed that the lead identifier column (`LeadId`) contains some embedded information: a date value and an identifier for the instance within the day. The following steps can be used to break out this information. The first one creates a separate working column with this information, which allows us to preserve the original, unmodified column:

<b>Transformation Name</b>	New formula
<b>Parameter: Formula type</b>	Single row formula
<b>Parameter: Formula</b>	LeadId
<b>Parameter: New column name</b>	'LeadIdworking'

You can now work off of this column to create your new ones. First, you can use the following replace transformation to remove the leading two characters, which are not required for the new columns:

<b>Transformation Name</b>	Replace text or pattern
----------------------------	-------------------------

<b>Parameter: Column</b>	LeadIdworking
<b>Parameter: Find</b>	'LE'
<b>Parameter: Replace with</b>	' '

Notice that the date information is now neatly contained in the first characters of the working column. Use the following to extract these values to a new column:

<b>Transformation Name</b>	Extract text or pattern
<b>Parameter: Column to extract from</b>	LeadIdworking
<b>Parameter: Option</b>	Custom text or pattern
<b>Parameter: Text to extract</b>	`{start}%{6}`

The new `LeadIdworking2` column now contains only the date information. Cleaning up this column requires reformatting the data, retyping it as a Datetime type, and then applying the `dateformat` function to format it to your satisfaction. These steps are left as a separate exercise.

For now, let's just rename the column:

<b>Transformation Name</b>	Rename columns
<b>Parameter: Option</b>	Manual rename
<b>Parameter: Column</b>	LeadIdworking1
<b>Parameter: New column name</b>	'LeadIdDate'

In the first working column, you can now remove the date information using the following:

<b>Transformation Name</b>	Replace text or pattern
<b>Parameter: Column</b>	LeadIdworking
<b>Parameter: Find</b>	`{start}%{6}`
<b>Parameter: Replace with</b>	' '

You can rename this column to indicate it is a daily identifier:

<b>Transformation Name</b>	Rename columns
<b>Parameter: Option</b>	Manual rename
<b>Parameter: Column</b>	LeadIdworking
<b>Parameter: New column name</b>	'LeadIdDaily'

## Results:

LeadId	LeadIdDaily	LeadIdDate	LastName	FirstName	Title	Phone	Request
LE160301001	001	160301	Jones	Charles	Chief Technical Officer	#INVALID#	Registration

LE160301002	002	160301	Lyons	Edward	#MISSING#	415-012-3456	Download Whitepaper
LE160301003	003	160301	Martin	Mary	CEO	#INVALID#	Delete Account
LE160301004	004	160301	Smith	Talia	Engineer	510-123-4567	Free Trial

### Example - Using \$col placeholder

This example illustrates how you can use the following conditional calculation functions to analyze weather data:

- **AVERAGEIF** - Average of a set of values by group that meet a specified condition. See *AVERAGEIF Function*.
- **MINIF** - Minimum of a set of values by group that meet a specified condition. See *MINIF Function*.
- **MAXIF** - Maximum of a set of values by group that meet a specified condition. See *MAXIF Function*.
- **VARIF** - Variance of a set of values by group that meet a specified condition. See *VARIF Function*.
- **STDEVIF** - Standard deviation of a set of values by group that meet a specified condition. See *STDEVIF Function*.

### Source:

Here is some example weather data:

date	city	rain	temp	wind
1/23/17	Valleyville	0.00	12.8	6.7
1/23/17	Center Town	0.31	9.4	5.3
1/23/17	Magic Mountain	0.00	0.0	7.3
1/24/17	Valleyville	0.25	17.2	3.3
1/24/17	Center Town	0.54	1.1	7.6
1/24/17	Magic Mountain	0.32	5.0	8.8
1/25/17	Valleyville	0.02	3.3	6.8
1/25/17	Center Town	0.83	3.3	5.1
1/25/17	Magic Mountain	0.59	-1.7	6.4
1/26/17	Valleyville	1.08	15.0	4.2
1/26/17	Center Town	0.96	6.1	7.6
1/26/17	Magic Mountain	0.77	-3.9	3.0
1/27/17	Valleyville	1.00	7.2	2.8
1/27/17	Center Town	1.32	20.0	0.2
1/27/17	Magic Mountain	0.77	5.6	5.2
1/28/17	Valleyville	0.12	-6.1	5.1
1/28/17	Center Town	0.14	5.0	4.9
1/28/17	Magic Mountain	1.50	1.1	0.4
1/29/17	Valleyville	0.36	13.3	7.3
1/29/17	Center Town	0.75	6.1	9.0
1/29/17	Magic Mountain	0.60	3.3	6.0

### Transformation:

The following computes average temperature for rainy days by city:

<b>Transformation Name</b>	New formula
<b>Parameter: Formula type</b>	Single row formula
<b>Parameter: Formula</b>	AVERAGEIF(temp, rain > 0)
<b>Parameter: Group rows by</b>	city
<b>Parameter: New column name</b>	'avgTempWRain'

The following computes maximum wind for sub-zero days by city:

<b>Transformation Name</b>	New formula
<b>Parameter: Formula type</b>	Single row formula
<b>Parameter: Formula</b>	MAXIF(wind,temp < 0)
<b>Parameter: Group rows by</b>	city
<b>Parameter: New column name</b>	'maxWindSubZero'

This step calculates the minimum temp when the wind is less than 5 mph by city:

<b>Transformation Name</b>	New formula
<b>Parameter: Formula type</b>	Single row formula
<b>Parameter: Formula</b>	MINIF(temp,wind<5)
<b>Parameter: Group rows by</b>	city
<b>Parameter: New column name</b>	'minTempWind5'

This step computes the variance in temperature for rainy days by city:

<b>Transformation Name</b>	New formula
<b>Parameter: Formula type</b>	Single row formula
<b>Parameter: Formula</b>	VARIF(temp,rain >0)
<b>Parameter: Group rows by</b>	city
<b>Parameter: New column name</b>	'varTempWRain'

The following computes the standard deviation in rainfall for Center Town:

<b>Transformation Name</b>	New formula
<b>Parameter: Formula type</b>	Single row formula
<b>Parameter: Formula</b>	STDEVIF(rain,city=='Center Town')



<b>Parameter: Group rows by</b>	city
<b>Parameter: New column name</b>	'stDevRainCT'

You can use the following transforms to format the generated output. Note the \$col placeholder value for the multi-column transforms:

<b>Transformation Name</b>	Edit column with formula
<b>Parameter: Columns</b>	stDevRainCenterTown,maxWindSubZero
<b>Parameter: Formula</b>	numformat(\$col,'##.##')

Since the following rely on data that has only one significant digit, you should format them differently:

<b>Transformation Name</b>	Edit column with formula
<b>Parameter: Columns</b>	varTempWRain,avgTempWRain,minTempWind5
<b>Parameter: Formula</b>	numformat(\$col,'##.#')

### Results:

date	city	rain	temp	wind	avgTempWRain	maxWindSubZero	minTempWind5	varTempWRain	stDevRain
1/23 /17	Valley ville	0.00	12.8	6.7	8.3	5.1	7.2	63.8	0.37
1/23 /17	Center Town	0.31	9.4	5.3	7.3		5	32.6	0.37
1/23 /17	Magic Mountain	0.00	0.0	7.3	1.6	6.43	-3.9	12	0.37
1/24 /17	Valley ville	0.25	17.2	3.3	8.3	5.1	7.2	63.8	0.37
1/24 /17	Center Town	0.54	1.1	7.6	7.3		5	32.6	0.37
1/24 /17	Magic Mountain	0.32	5.0	8.8	1.6	6.43	-3.9	12	0.37
1/25 /17	Valley ville	0.02	3.3	6.8	8.3	5.1	7.2	63.8	0.37
1/25 /17	Center Town	0.83	3.3	5.1	7.3		5	32.6	0.37
1/25 /17	Magic Mountain	0.59	-1.7	6.4	1.6	6.43	-3.9	12	0.37
1/26 /17	Valley ville	1.08	15.0	4.2	8.3	5.1	7.2	63.8	0.37
1/26 /17	Center Town	0.96	6.1	7.6	7.3		5	32.6	0.37
1/26 /17	Magic Mountain	0.77	-3.9	3.0	1.6	6.43	-3.9	12	0.37

1/27 /17	Valley ville	1.00	7.2	2.8	8.3	5.1	7.2	63.8	0.37
1/27 /17	Cente r Town	1.32	20.0	0.2	7.3		5	32.6	0.37
1/27 /17	Magic Mount ain	0.77	5.6	5.2	1.6	6.43	-3.9	12	0.37
1/28 /17	Valley ville	0.12	-6.1	5.1	8.3	5.1	7.2	63.8	0.37
1/28 /17	Cente r Town	0.14	5.0	4.9	7.3		5	32.6	0.37
1/28 /17	Magic Mount ain	1.50	1.1	0.4	1.6	6.43	-3.9	12	0.37
1/29 /17	Valley ville	0.36	13.3	7.3	8.3	5.1	7.2	63.8	0.37
1/29 /17	Cente r Town	0.75	6.1	9.0	7.3		5	32.6	0.37
1/29 /17	Magic Mount ain	0.60	3.3	6.0	1.6	6.43	-3.9	12	0.37