# Escaping Strings in Transforms

This section describes how to escape strings in your transforms.

In the platform, the backslash character (\) is used to escape values within strings. The character following the escaping character is treated as a string literal.

For example, the following value is used to represent a matching value of & only:

```
value: `\&`
```

Escaping can be applied to parameters in functions. For example, in the data grid, you have the following values in a column:

| MyStringCol |
|---|
| This works. |
| You can't break this. |
| Not broken yet. |

To find the value `can't`, you could enter the following pattern:

```
derive type:single value: FIND(MyStringCol, 'can\'t',true,0) as:'MyFindResults'
```

The above transform results in the following:

| MyStringCol | MyFindResults |
|---|---|
| This works. | |
| You can't break this. | 4 |
| Not broken yet. | |

All pattern type markers can be escaped if using the marking character in a string:

| Pattern type | Marker | Escaped character |
|---|---|---|
| literal value | ' | \' |
| Trifacta® pattern | ` | \` |
| Regular expression | / | \/ |

**A note on JSON:**

In the data grid, JSON Objects and arrays include additional escaping to show that the values are strings. For example, the data grid shows:

```
{"re\"becca","hello"}
```

The first JSON element displayed in the GUI is `re\"becca`, but the desired match is `re"becca`.

> **Tip:** For best results in pattern matching, you should make selections in the data grid and modify if necessary.

Below, you can see how this JSON pattern is specified in an example `unnest` transform:

```
unnest col: myCol keys:'["re\\\"becca"]'
```

- The `keys` value must be single-quoted. Since the keys are specified for Object data, the square bracket notation is used.
- Within the square brackets, the individual keys must be double-quoted.
- The first two backslashes (`\\`) indicate that you are escaping a single backslash character.
- The third backslash indicates that you are escaping the double-quote that is part of the string to match.

In the following example, you are trying to match on the above string, including the double-quotes around it: `"re\"becca"`.

```
unnest col: myCol keys:'[\"re\\\"becca\"]'
```

The bracketing double-quotes must be escaped, too.