EXAMPLE - Quote Parameter

This example demonstrates how the `quote` parameter can be used for more sophisticated splitting of columns of data using the `split` transform.

Source:

In this example, the following CSV data, which contains contact information, is imported into the application:

```
LastName,FirstName,Role,Company,Address,Status
Wagner,Melody,VP of Engineering,Example.com,"123 Main Street, Oakland, CA 94601",Prospect
Gruber,Hans,"Director, IT",Example.com,"456 Broadway, Burlingame, CA, 94401",Customer
Franks,Mandy,"Sr. Manager, Analytics",Tricorp,"789 Market Street, San Francisco, CA, 94105",Customer
```

Transform:

When this data is pulled into the application, some initial parsing is performed for you:

column2	column3	column4	column5	column6	column7
LastName	FirstName	Role	Company	Address	Status
Wagner	Melody	VP of Engineering	Example.com	"123 Main Street, Oakland, CA 94601"	Prospect
Gruber	Hans	"Director, IT"	Example.com	"456 Broadway, Burlingame, CA, 94401"	Customer
Franks	Mandy	"Sr. Manager, Analytics"	Tricorp	"789 Market Street, San Francisco, CA, 94105"	Customer

When you open the Recipe Panel, you should see the following transforms:

```
splitrows col: column1 on: '\r' quote: '"'
```

```
split col: column1 on: ',' limit: 5 quote: '"'
```

The first transform splits the raw source data into separate rows in the carriage return character (`\r`), ignoring all values between the double-quote characters. Note that this value must be escaped. The double-quote character does not require escaping. While there are no carriage returns within the actual data, the application recognizes that these double-quotes are identifying single values and adds the quote value.

The second transform splits each row of data into separate columns. Since it is comma-separated data, the application recognizes that this value is the column delimiter, so the `on` value is set to the comma character (`,`). In this case, the quoting is necessary, as there are commas in the values in `column4` and `column6`, which are easy to clean up.

To finish clean up of the dataset, you can promote the first row to be your column headers:

```
header
```

You can remove the quotes now. Note that the following applies to two columns:

```
replace col: Role, Address with: '' on: `"` global: true
```

Now, you can split up the `Address` column. You can highlight one of the commas and the space after it in the column, but make sure that your final statement looks like the following:

```
split col: Address on: ', ' limit: 2
```

Notice that there is some dirtiness to the resulting `Address3` column:

Address3
CA 94601
CA, 94401
CA, 94105

Use the following to remove the comma. In this case, it's important to leave the space between the two values in the column, so the `on` value should only be a comma. Below, the `width` value is two single quotes:

```
replace col: Address3 with: '' on: `,` global: true
```

You can now split the `Address3` column on the space delimiter:

```
split col: Address3 on: `{delim}`
```

Since the data is regularly formatted, you can use the Trifacta® pattern `{delim}`.

Results:

After you rename the columns, you should see the following:

LastName	FirstName	Role	Company	Address	City	State	Zipcode	Status
Wagner	Melody	VP of Engineering	Example.com	123 Main Street	Oakland	CA	94601	Prospect
Gruber	Hans	Director, IT	Example.com	456 Broadway	Burlingame	CA	94401	Customer
Franks	Mandy	Sr. Manager, Analytics	Tricorp	789 Market Street	San Francisco	CA	94105	Customer