Extractkv Transform

Contents:

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*.

Extracts key-value pairs from a source column and writes them to a new column.

Source column must be of String type, although the data can be formatted as other data types. The generated column is of Object type.

Your source column (`MyKeyValues`) is formatted in the following manner:

```
key1=value1,key2=value2
```

Basic Usage

The following transform extracts the key-value pairs. The `key` parameter contains a single pattern that matches all keys that you want to extract:

```
extractkv col: MyKeyValues key:`{alpha}+{digit}` valueafter: '=' delimiter: ','
```

Output: The generated column contains data that looks like the following:

```
{"key1":"value1","key2":"value2"}
```

If the source data contained additional keys which were not specified in the transform, those key-value pairs would not appear in the generated column.

Syntax and Parameters

```
extractkv col:column_ref delimiter:string_literal_pattern key:string_literal_pattern
valueafter:string_literal_pattern [as:'new_column_name']
```

Parameter	Required?	Data Type	Description
extractkv	Y	transform	Name of the transform
col	Y	string	Source column name

delimiter	Y	string	String literal or pattern that identifies the separator between key-value pairs
key	Y	string	Pattern that identifies the key to match
valueafter	Y	string	String literal or pattern after which is located a key's value
as	N	string	Name of the newly generated column

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

col

Identifies the column to which to apply the transform. You can specify only one column.

Usage Notes:

Required?	Data Type
Yes	String (column name)

delimiter

Specifies the character or pattern that defines the end of a key-value pair. This value can be specified as a String literal, regular expression, or Trifacta® pattern .

In the following:

```
{ key1=value1,key2=value2 }
```

The delimiter is the comma (`','`). The final key-value pair does not need a delimiter.

Tip: You can insert the Unicode equivalent character for this parameter value using a regular expression of the form `/\uHHHH/`. For example, `/\u0013/` represents Unicode character `0013` (carriage return). For more information, see *Supported Special Regular Expression Characters*.

Usage Notes:

Required?	Data Type
Yes	String (literal, regular expression, or Trifacta pattern)

key

Specifies the pattern used to extract the keys from a source column by the `extractkv` transform. For the following data:

```
{ key1=value1,key2=value2 }
```

The keys are represented in the transform by the following parameter and value:

```
key:`{alpha}+{digit}`
```

This pattern matches all keys that begin with a letter and end with a digit. If the source data contains other keys, they do not appear in the extracted data.

Usage Notes:

Required?	Data Type
Yes	Single pattern representing the individual keys to extract.

valueafter

Specifies the character or pattern after which the value is specified in a key-value pair. This value can be specified as a String literal, regular expression, or Trifacta® pattern.

For the following:

```
{ key1=value1,key2=value2 }
```

The `valueafter` string is the equals sign (`'='`).

Usage Notes:

Required?	Data Type
Yes	String (literal, regular expression, or Trifacta pattern)

as

Name of the new column that is being generated. If the `as` parameter is not specified, a default name is used.

Usage Notes:

Required?	Data Type
No	String (column name)

Examples

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

Example - extracting key values from car data and the unnesting into separate columns

This example shows how you can unpack data nested in an Object into separate columns using the following transforms:

- `extractkv` - Removes key-value pairs from a source string. See *Extract Transform*.
- `unnest` - Unpacks nested data in separate rows and columns. See *Unnest Transform*.

Source:

You have the following information on used cars. The `VIN` column contains vehicle identifiers, and the `Properties` column contains key-value pairs describing characteristics of each vehicle. You want to unpack this data into separate columns.

VIN	Properties

XX3 JT4522	year=2004,make=Subaru,model=Impreza,color=green,mileage=125422,cost=3199
HT4 UJ9122	year=2006,make=VW,model=Passat,color=silver,mileage=102941,cost=4599
KC2 WZ9231	year=2009,make=GMC,model=Yukon,color=black,mileage=68213,cost=12899
LL8 UH4921	year=2011,make=BMW,model=328i,color=brown,mileage=57212,cost=16999

Transform:

Add the following transform, which identifies all of the key values in the column as beginning with alphabetical characters.

- The `valueafter` string identifies where the corresponding value begins after the key.
- The `delimiter` string indicates the end of each key-value pair.

```
extractkv col:Properties key:`{alpha}+` valueafter:`=` delimiter:`,`
```

Now that the Object of values has been created, you can use the `unnest` transform to unpack this mapped data. In the following, each key is specified, which results in separate columns headed by the named key:

```
unnest col:extractkv_Properties keys:'year','make','model','color','mileage','cost'
```

Results:

When you delete the unnecessary Properties columns, the dataset now looks like the following:

VIN	year	make	model	color	mileage	cost
XX3 JT4522	2004	Subaru	Impreza	green	125422	3199
HT4 UJ9122	2006	VW	Passat	silver	102941	4599
KC2 WZ9231	2009	GMC	Yukon	black	68213	12899
LL8 UH4921	2011	BMW	328i	brown	57212	16999