MATCHES Function

Contents:

Returns `true` if a value contains a string or pattern. The value to search can be a string literal or a reference to a column of String type.

Since the `MATCHES` function returns a Boolean value, it can be used as both a function and as a conditional.

Tip: When you select values in a histogram for a column of Array type, the function that identifies the values on which to perform a transform is typically `MATCHES`.

Tip: If you need the location of the matched string within the source, use the `FIND` function. See *FIND Function*.

Basic Usage

Column reference example:

```
delete row: MATCHES(ProdId, 'Fun Toy')
```

Output: Deletes any row in which the value in the `ProdId` column value contains the string literal `Fun Toy`.

String literal example:

```
derive type:single value: MATCHES('Hello, World', 'Hello')
```

Output: For all values in the dataset returns `true`.

Syntax and Arguments

```
derive type:single value:MATCHES(column_string,string_pattern)
```

Argument	Required?	Data Type	Description
column_string	Y	string	Name of column or string literal to be searched
string_pattern	Y	string	String literal or pattern to find

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

column_string

Name of the column or string literal to be searched.

- Missing string or column values generate missing string results.
  - String constants must be quoted (`'Hello, World'`).
- Multiple columns can be specified as an array (`matches([Col1,Col2],'hello')`.

Usage Notes:

Required?	Data Type	Example Value
Yes	String	`MyColumn`

string_pattern

String literal, Trifacta® pattern, or regular expression to match against the source column-string.

- Column references are not supported.

Usage Notes:

Required?	Data Type	Example Value
Yes	String literal or pattern	`'home page'`

Examples

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

Example - Filtering log data

In downloaded log files, you might see error messages of the following type:

- `INFO` - status information on the process
- `WARNING` - system encountered a non-fatal error during execution
- `ERROR` - system encountered an error, which might have caused the job to fail.

For purposes of analysis, you might want to filter out the data for `INFO` and `WARNING` messages.

Source:

Here is example data from a log file of a failed job:

log
2016-01-29T00:14:24.924Z com.example.hadoopdata.monitor.spark_runner.ProfilerServiceClient [pool-13-thread-1] INFO com.example.hadoopdata.monitor.spark_runner.BatchProfileSparkRunner - Spark Profiler URL - http://localhost:4006/
2016-01-29T00:14:40.066Z com.example.hadoopdata.monitor.spark_runner.BatchProfileSparkRunner [pool-13-thread-1] INFO com.example.hadoopdata.monitor.spark_runner.BatchProfileSparkRunner - Spark process ID was null.
2016-01-29T00:14:40.067Z com.example.hadoopdata.monitor.spark_runner.BatchProfileSparkRunner [pool-13-thread-1] INFO com.example.hadoopdata.monitor.spark_runner.BatchProfileSparkRunner - ----------END SPARK JOB----------
2016-01-29T00:14:44.961Z com.example.hadoopdata.joblaunch.server.BatchPollingWorker [pool-4-thread-2] ERROR com.example.hadoopdata.joblaunch.server.BatchPollingWorker - Job '128' threw an exception during execution

2016-01-29T00:14:44.962Z com.example.hadoopdata.joblaunch.server.BatchPollingWorker [pool-4-thread-2] INFO com.example.hadoopdata.joblaunch.server.BatchPollingWorker - Making sure async worker is stopped
2016-01-29T00:14:44.962Z com.example.hadoopdata.joblaunch.server.BatchPollingWorker [pool-4-thread-2] INFO com.example.hadoopdata.joblaunch.server.BatchPollingWorker - Notifying monitor for job '128', code 'FAILURE'
2016-01-29T00:14:44.988Z com.example.hadoopdata.monitor.client.MonitorClient [pool-4-thread-2] INFO com.example.hadoopdata.monitor.client.MonitorClient - Request succeeded to monitor ip-0-0-0-0.example.com:8001

Transform:

When the above data is loaded into the application, you might want to break up the data into separate columns, which splits them on the `Z` character at the end of the timestamp:

```
split col: column1 on: `Z `
```

Then, you can rename the two columns: `Timestamp` and `Log_Message`. To filter out the `INFO` and `WARNING` messages, you can use the following transforms, which match on the string literals to identify these messages:

```
delete row: MATCHES(Log_Message, '] INFO ')
```

```
delete row: MATCHES(Log_Message, '] WARNING ')
```

Results:

After the above steps, the data should look like the following:

Timestamp	Log_Message
2016-01-29T00:14:44.961	com.example.hadoopdata.joblaunch.server.BatchPollingWorker [pool-4-thread-2] ERROR com.example.hadoopdata.joblaunch.server.BatchPollingWorker - Job '128' threw an exception during execution