Text Matching

Trifacta® Wrangler Enterprise supports the following types of text matching clauses:

- **String literals** match specified strings exactly. Written using single quotes ('...') or double quotes ("...").
- **Regular expressions** enable pattern-based matching. Regular expressions are written using forward slashes (/.../). The syntax is based on *RE2* and *PCRE* regular expressions.

  > **NOTE:** Regular expressions are considered a developer-level capability and can have significant consequences if they are improperly specified. Unless you are comfortable with regular expressions, you should use Trifacta patterns instead.

- **Trifacta patterns** are custom selectors for patterns in your data and provide a simpler and more readable alternative to regular expressions. They are written using backticks (`...`).

> **Tip:** You can create patterns to match source values in a column by example. By providing example matches for values in your source column, you can rapidly build complex pattern-based matches. For more information on transformation by example, see *Overview of TBE*.

- **Column names** are simple text strings in Wrangle. If the column name contains a space, it must be bracketed in curly braces: `{my Column Name}`. For more information, see *Rename Columns*.

The following are example Trifacta patterns:

> **Tip:** After using Trifacta patterns, regular expressions, or string literals in a recipe step, you can reuse them in your transformations where applicable. See *Pattern History Panel*.

Pattern	Description
%	match any character, exactly once
%?	match any character, zero or one times
%*	match any character, zero or more times
%+	match any character, one or more times
%{3}	match any character, exactly three times
%{3,5}	match any character, 3, 4, or 5 times
#	digit character [0-9]
{any}	match any character, exactly once
{start}	match the start of the line
{end}	match the end of the line
{alpha}	alpha character [A-Za-z_]
{upper}	uppercase alpha character [A-Z_]
{lower}	lowercase alpha character [a-z_]
{digit}	digit character [0-9]
{delim}	single delimiter character e.g :, ,, |, /, -, ., \s
{delim-ws}	single delimiter and all the whitespace around it
{alpha-numeric}	match a single alphanumeric character

{alphanum-underscore}	match a single alphanumeric character or underscore character
{at-username}	match `@username` values
{hashtag}	match `#hashtag` values
{zip}, {hex}, {phone}, {email}	extensible types, as regexes
{state}, {state-abbrev}	extensible types continued
{month}, {month-abbrev}, {url}	extensible types continued
{ip-address}, {hex-ip-address}	extensible types continued
{time}, {bool}	extensible types continued
{[...]}	character class matches characters in brackets
{![...]}	negated class matches characters not in brackets
(...)	grouping, including captures
#, %, ?, *, +, {, }, (,), \, ', \n, \t	escaped characters or pattern modifiers Use a double backslash (`\\`) to denote an escaped string literal. For more information, see *Escaping Strings in Transforms*.
|	logical OR

- Logical AND is the implied operator when you concatenate text matching patterns.
- Logical NOT is managed using negated classes.

See also *Capture Group References*.