|  |
| --- |
| **Regular Expressions** |

**Using RLIKE**

The RLIKE function can be used to match on the basis of regular expressions. 

A regular expression is a pattern that describes the general shape of a string.  There is a special notation for describing the features we would like to see in matching strings.   

First, a literal string matches the string.  So, the pattern ‘cat’ matches ‘cat’.  However, it also matches ‘catacomb’ and ‘the cat sat on the mat’.  The pattern cat matches ‘cat’ anywhere inside the target string. 

If you want to match only the word ‘cat’, then the pattern would need to be ‘^cat$’.   The caret (^) means “anchor to the start of the string;” in other words, the first thing at the start of a matching string is the word ‘cat’.  The dollar sign ($) means “anchor to the end of the string”; in other words, the last thing in the string must be the word ‘cat’.  So the pattern ‘^cat$’ can match only the string ‘cat’ and nothing else. 

Regular expressions also support wildcards, just as LIKE does.  However, the wildcard is different.  There is only one, the dot (.) that will match any single character.  So, ‘.at’ matches ‘cat’, ‘bat’, ‘mat’ and so on. 

You only need a single wildcard character because you can also specify how often characters (including wildcards) can appear in a string.

For example, the special \* character after a character means that character may appear zero or more times.  So, the pattern ‘n\*’ matches ‘’, ‘nn’, ‘nnn’, and so on.  You can group characters with parentheses, so ‘(cat)\*’ matches ‘’, ‘cat’, ‘catcat’, ‘catcatcat’, and so on.  You can also use the wildcard, so ‘.\*’ matches any number of any character – basically anything. 

Also, the plus sign (+) means that the character or string before it should be repeated one or more times, and the question mark (?) means to match zero times or one time.  You can also list a specific range, so for example, ‘(cat)(2,4)’ matches ‘catcat’, ‘catcatcat’, and ‘catcatcatcat’. 

As well as listing specific characters and strings, you can list sets of characters.  These appear in square brackets.  For example, the pattern  ‘[a-z]’ matches any single letter and ‘[a-z]\*’ matches any number of letters. 

Finally, there are a number of character classes, which are predefined sets.  For example, [[:alnum:]] matches any alphanumeric character. 

**Example 1:**

*Find all customers with the characters ‘ea’ in the customer name.*

This will find all customer names that contain the string ‘ea’ somewhere inside them.