**Regex Operators in PostgreSQL**

PostgreSQL provides several operators for working with regex:

~: Case-sensitive match.

~\*: Case-insensitive match.

!~: Case-sensitive non-match.

!~\*: Case-insensitive non-match.

**Basic regex syntax you'll encounter in PostgreSQL.**

**Dot (.)**

.: Matches any single character except a newline.

**Example:**

SELECT 'hello' ~ 'h.llo'; -- Returns true (matches 'hello')  
  
SELECT username

FROM users

WHERE username ~ 'j.n'; -- Matches usernames like 'jon', 'jan', 'jen', etc.

**Caret (^)**

^: Matches the start of a string.

**Example:**

SELECT 'hello' ~ '^h'; -- Returns true (string starts with 'h')  
  
SELECT username

FROM users

WHERE username ~ '^j'; -- Matches usernames starting with 'j'

**Dollar Sign ($)**

$: Matches the end of a string.

Example:

SELECT 'hello' ~ 'o$'; -- Returns true (string ends with 'o')  
  
SELECT username

FROM users

WHERE username ~ 'e$'; -- Matches usernames that end with 'e'

**Asterisk (\*)**

\*: Matches zero or more occurrences of the preceding element.

Example:

SELECT 'heeeellooo' ~ 'e\*'; -- Returns true (matches multiple 'e')  
  
SELECT username

FROM users

WHERE username ~ 'a\*b'; -- Matches usernames like 'b', 'ab', 'aab', 'aaab', etc.

**Plus (+)**

+: Matches one or more occurrences of the preceding element.

Example:

SELECT 'helloooo' ~ 'o+'; -- Returns true (matches multiple 'o')  
  
SELECT username

FROM users

WHERE username ~ 'a+b'; -- Matches usernames like 'ab', 'aab', 'aaab', etc.

**Question Mark (?)**

?: Matches zero or one occurrence of the preceding element.

Example:

SELECT 'hello' ~ 'he?l'; -- Returns true (matches 'hello')  
  
SELECT username

FROM users

WHERE username ~ 'a?b'; -- Matches usernames like 'b', 'ab'

**Square Brackets ([ ])**

[abc]: Matches any one character inside the brackets (a, b, or c).

[a-z]: Matches any character in the range from a to z.

Example:

SELECT 'hello' ~ '[hH]'; -- Returns true (matches either 'h' or 'H')

SELECT 'hello' ~ '[a-z]'; -- Returns true (matches any lowercase letter)  
  
SELECT username

FROM users

WHERE username ~ '[aeiou]'; -- Matches usernames containing any vowel

**Caret Inside Square Brackets ([^ ])**

[^abc]: Matches any character except 'a', 'b', or 'c'.

Example:

SELECT 'hello' ~ '[^aeiou]'; -- Returns true (matches any consonant)

SELECT username

FROM users

WHERE username ~ '[^a-z]'; -- Matches usernames containing non-lowercase letters

**Parentheses ( ) for Grouping**

(abc): Groups a pattern together.

Example:

SELECT 'helloworld' ~ '(hello)'; -- Returns true (matches 'hello')

SELECT username

FROM users

WHERE username ~ '(abc|def)'; -- Matches usernames containing either 'abc' or 'def'  
  
  
**1. | (Alternation / OR Operator)**

The pipe symbol (|) is used to represent **alternation** or **OR** in regular expressions. It matches either the pattern on the left side **or** the pattern on the right side.  
  
SELECT \*

FROM users

WHERE username ~ 'bob|alice'; -- Matches 'bob' or 'alice'  
  
  
**Summary of Escape Sequences in Regular Expressions**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Escape Sequence** | **Description** | | --- | --- | | \\. | Matches a literal dot (.) | | \\\* | Matches a literal asterisk (\*) | | \\( | Matches a literal opening parenthesis (() | | \\) | Matches a literal closing parenthesis ()) | | \\\\ | Matches a literal backslash (\) | | \\d | Matches any digit (0-9) | | \\w | Matches any word character (letters, digits, and underscore) | | \\s | Matches any whitespace character (spaces, tabs, newlines) | | \\b | Matches a word boundary | | \\n | Matches a newline character | | \\t | Matches a tab character | | \\uXXXX | Matches a Unicode character with hexadecimal code XXXX | |

**{n}: Matches exactly n occurrences of the preceding element.**  
SELECT \*

FROM users

WHERE username ~ 'a{3}'; -- Matches usernames with exactly 3 consecutive 'a's

**{n,}: Matches n or more occurrences.**  
SELECT \*

FROM users

WHERE username ~ 'a{2,}'; -- Matches usernames with at least 2 'a's in a row

**{n,m}: The {n,m} quantifier specifies that the preceding element must appear at least n times but no more than m times.**  
SELECT \*

FROM users

WHERE username ~ 'a{2,4}'; -- Matches usernames with 2 to 4 'a's in a row

| **Escape Sequence** | **Description** | **Example** | **Explanation** |
| --- | --- | --- | --- |
| \\ | Escapes the backslash (\) character. This is necessary to include a literal backslash in your regex. | \\\\ | Matches a single backslash. Since backslash is an escape character, it needs to be escaped by using \\. |
| \n | Matches a newline character (Line Feed, LF, ASCII 10). | 'a\nb' | Matches the string "a" followed by a newline and then "b". |
| \r | Matches a carriage return character (ASCII 13). | 'a\rb' | Matches "a", followed by a carriage return, and then "b". |
| \t | Matches a tab character (ASCII 9). | 'a\tb' | Matches "a", followed by a tab, and then "b". |
| \b | Matches a word boundary. This is used to match positions where a word starts or ends. | '\\bword\\b' | Matches "word" only if it is surrounded by word boundaries (e.g., spaces, punctuation). |
| \B | Matches a position where a word boundary is **not** present. | '\\Babc\\B' | Matches "abc" if it is not at the start or end of a word. It would not match "abc" in "abc def" but would match in "abc123". |
| \d | Matches any digit character (0-9). | '\\d{3}' | Matches exactly 3 digits (e.g., "123" or "456"). |
| \D | Matches any non-digit character (i.e., anything except 0-9). | '\\D+' | Matches one or more non-digits (e.g., "abc" or "#$%"). |
| \w | Matches any "word" character, which includes letters (A-Z, a-z), digits (0-9), and the underscore (\_). | '\\w+' | Matches one or more word characters (e.g., "word", "abc123", or "underscore\_"). |
| \W | Matches any character that is **not** a word character (anything except letters, digits, or underscore). | '\\W' | Matches any non-word character (e.g., spaces, punctuation, or special characters like @, #, !). |
| \s | Matches any whitespace character (spaces, tabs, newlines, carriage returns, etc.). | '\\s' | Matches any whitespace character (e.g., space, tab, newline). |
| \S | Matches any character that is **not** a whitespace character. | '\\S' | Matches any non-whitespace character (e.g., letters, digits, punctuation). |
| \Z | Matches the end of the string with the optional presence of a newline. | 'abc\\Z' | Matches the string "abc" only if it is at the end of the string, but allows an optional newline at the end. |
| \z | Matches the absolute end of the string, meaning no newlines are allowed at the end. | 'abc\\z' | Matches the string "abc" only at the absolute end of the string, without any newline characters allowed. |
| \A | Matches the start of the string (similar to ^ but without multiline behavior). | '\\Astart' | Matches "start" only if it is at the very beginning of the string, even with multiline matching. |
| \G | Matches the position where the last match ended. This is useful for match continuations. | '\\Gabc' | Matches "abc" if it directly follows the previous match, similar to a continuation pattern. |
| \X | Matches any Unicode extended grapheme cluster (can be multi-character sequences). | '\\X' | Matches any extended grapheme (e.g., combining characters, emoji sequences). |
| \ (Literal) | A backslash is used to escape characters that otherwise have a special meaning (e.g., \*, +, ?, etc.). | '\\d' | Matches a literal d or the escape for digits \d when needed. |

**More Examples of Use Cases:**

* **\b (Word Boundary):**  
  \\bcat\\b matches "cat" in "a cat", but not in "scattering" (since "cat" is not at a word boundary in this case).
* **\d (Digit Matching):**  
  \\d{2,4} matches any number of digits that are between 2 and 4 digits long, such as "12", "123", or "1234".
* **\s (Whitespace Matching):**  
  \\s+ matches one or more whitespace characters (spaces, tabs, newlines). This can be used to find excessive spaces between words.
* **\S (Non-Whitespace Matching):**  
  \\S+ matches any sequence of non-whitespace characters, useful for extracting words from text or filenames.

**Important Notes:**

1. **Backslashes and Escaping:**  
   In PostgreSQL regular expressions, since backslash (\) is an escape character, you need to use double backslashes (\\) to represent a single backslash. For example, \\d represents a digit, while \d on its own would result in an error.
2. **Word Boundaries:**  
   \b is a zero-width assertion and doesn't consume characters. It's simply used to assert a position where a word starts or ends.
3. **\A vs ^ and \z vs $:**  
   While ^ and $ can be used in regular expressions to match the beginning and end of a string, respectively, \A and \z are stricter, ensuring matching occurs at the very start and end of the string without being affected by multiline options.

This should give you a more comprehensive view of the escape sequences used in PostgreSQL regular expressions within pgAdmin.