Cognizant | Academy
Business Focused. Learner Centric.

# Oracle Version 12c

## Regular Expression

## Enabling Objectives

After completing this chapter, in the next 90 minutes you will be able to :

- Implement Regular expression functions to retrieve data which matches a specific pattern
- Write at least one constraint on the table with the usage of regular Expression for implementing Data Validation

## Regular Expressions & Metacharacters

- Regular Expressions are mechanisms for describing patterns in Text data.
- They are a collections of literals and/ or metacharacters used to perform
    - complex search
    - data extraction
    - data  transformation or modifications operations
- Metacharacters are collections of special characters used in regular expressions for describing or searching patterns.

## Regular expression metacharacters

| Meta characters | Meaning | Examples |
|---|---|---|
| \ | Indicates that the matched character is a special character, a literal, or a backreference | \n ---- matches new line<br>\\------ matches ‘\’<br>\( ----- matches ‘(‘ |
| . | Matches any single character except newline | hob.it matches hobait, hobbit and so |
| ^ | Matches the position at the start of the string<br>If used inside [^ ] negates the expression | ^A matches Abba, Able<br>[^Ff] matches anything except upper or lower case F<br>[^a-z] matches everything except<br>lower case a to z. |
| $ | Matches the position at the end of the string | B$ matches bob, cab and so on. |
| * | Matches characters zero or more times | tr* matches tree, trunk, tr and so on. |
| + | Matches characters one or more times | tr+ matches tree, trunk and so on. |
| ? | Matches characters zero or one times | tr? matches tr, trn |

## Regular expression metacharacters

| Meta characters | Meaning | Examples |
|---|---|---|
| x \| y | Matches x or y (x,y are one or more characters) | t(a\|e\|i)n matches tan, tin, Pakistan |
| ( pattern ) | A subexpression that matches the specified patter | anatom(y\|ies) matches anatomy , anatomies |
| { n} | Matches a character exactly n times | hob{2}it  matches hobbit |
| { n, } | Matches a character n or more times | hob{2,}it matches hobbit, hobbbit etc |
| { n , m } | Matches a character n to m times | Hob{2,4} matches hobbit, hobbbit and hobbbbit |
| [ abc ] | Matches any of the enclosed characters | [ab]bc matches abc and bbc |
| [a –z] | Matches any characters in the specified range | [a –c]bc matches abc, bbc, cbc |

## Regular Expression

## REGEXP_LIKE()

- Allow you to specify what type of character you are looking for
- Class names should be specified in lower case

| Character Class | Description |
|---|---|
| [:alpha:] | Alphabetic characters |
| [:lower:] | Lowercase alphabetic characters |
| [:upper:] | Uppercase alphabetic characters |
| [:digit:] | Numeric digits |
| [:alnum:] | Alphanumeric characters |
| [:space:] | Space characters such as newline, vertical tab |
| [:punct:] | Punctuation characters |
| [:cntrl:] | Control characters |
| [:print:] | Printable characters |

## REGEXP_LIKE()

- Similar to LIKE operator
- Searches for patterns that satisfy the regular expression
- allows you to perform regular expression matching in the WHERE clause of a SELECT, INSERT, UPDATE, or DELETE statement

REGEXP_LIKE(x, pattern [, match_option])

  - x refers to search string or column
  - pattern refers to regular expression
  - match_option allows matching for case, ignoring newlines and matching across multiple lines
    - 'c' specifies case sensitive matching (default).
    - 'i' specifies case insensitive matching.
    - 'n' allows to use the match-any-character
    - 'm' treats x as multiple line.

- REGEXP_LIKE returns true when the source x matches the pattern

## REGEXP_LIKE()

Lend a Hand:

```
select  *  from employees where
regexp_like(to_char(dob,  ‘YYYY’) ,‘^199[5-8 ]$’ );
```

Output:

Retrieves employees who have joined in between 1995 and 1998.

```
select * from employees where regexp_like( e_name, ‘^j’ , ’i’);
```

Output:

Retrieves details of employees whose first name starts with ‘J’ or  ‘j’.

## REGEXP_LIKE()

Lend a Hand:

```
select *  from employees where regexp_like(e_name, ‘^Ste(v|ph)en$’);
```

Output:

Retrieves details of employees with name of Stephen or Steven
( where name begins with ‘Ste’ and ends with ‘en’ and in between is ‘ph’ or ‘v’)

## REGEXP_INSTR()

- Searches for patterns in x and returns the position at which the pattern occurs

```
REGEXP_INSTR(x, pattern [,start [,occurrence
[,return_option [,match_option ] ] ] ] )
```

  - x  refers to search string or column
  - start refers to position to begin the search
  - Occurrence refers to indicates which occurrence of pattern should be returned
  - Return_option refers to Indicates what integer to return
  - Match_option refers to change the default matching

- The REGEXP_INSTR function returns a numeric value.

- If the REGEXP_INSTR function does not find any occurrence of pattern, it will return 0.

## REGEXP_INSTR()

Lend a Hand:

```
select regexp_instr('But, soft! What light through yonder window breaks? ',
'I          [[:alpha:]] {4}') as result from dual;
```

Output : 17

Returns the position of I in the given expression

```
select regexp_instr('But, soft! What light through yonder window breaks? ',
's [[:alpha:]] {3}', 1, 2) as result from dual;
```

Output : 45

Returns the position of the second occurrence of s [[:alpha:]] {3}
from the beginning of the string.

## REGEXP_INSTR()

Lend a Hand:

```
select regexp_instr(‘ 500 Oracle Parkway, redwood Shores,
CA’,’[^ ]+’,1, 6) from dual;
```

Output : 37

Returns the position of the 6th blank space from the beginning of the text.

## REGEXP_REPLACE()

- Replace a sequence of characters in a string with another set of characters.

```
REGEXP_REPLACE(x, pattern [,replace_string [,start
[,occurrence [,match_option ] ] ] ] )
```

  - x  refers to search string or column
  - Pattern refers to expression being searched
  - replace_string refers to Replacement String
  - start refer to Position to begin the search
  - occurrence refers to indicates which occurrence of pattern should be returned
  - match_option refers to change the default matching

- The REGEXP_REPLACE function returns a string value.

## REGEXP_REPLACE

Lend a Hand:

select regexp_instr(‘But, soft! What light through yonder window breaks? ‘,‘ I [[:alpha:]] {4} ’, ‘sound’) as result from dual;

Output:

Replaces ‘light’ with ‘sound’ in the given string

select regexp_replace( 515.123.4567, ‘{[[:digit:]] {3})\.([[:digit:]]{3})\.([[:digit:]]{4}’, ‘(\1) \2-\3’) from dual;

Output:

Transforms  the pattern xxx.xxx.xxxx to (xxx) xxx-xxxx

## REGEXP_SUBSTR()

- Returns a substring from a string  that matches the pattern

REGEXP_SUBSTR(x, pattern [,start[,occurrence [,match_option ] ] ]  )

  - x  refers to search string or column
  - pattern refers to expression being searched
  - start refers to position to begin the search
  - occurrence refers to indicates which occurrence of pattern should be returned
  - match_option refers to change the default matching

- The REGEXP_SUBSTR function returns a string value.

- If the REGEXP_SUBSTR function does not find any occurrence of pattern, it will return NULL.

## REGEXP_SUBSTR

Lend a Hand:

select regexp_substr( ‘But, soft! What light through yonder window breaks? ‘,‘ l [[:alpha:]] {4} ’ ) from dual;

Output: light

Returns substring that matches the regular expression l[[:alpha:]] {4}

select regexp_substr(‘Joe Smith, 12345 Berry lane, Orta, CA 91234’,‘[[:digit:]] {5} $’) from dual;

Output: 91234

The query finds 5 digit number anchored to the end of the line

## Applying Regular Expressions

Regular_expressions can be used in check constraint for data validation:

Example:

```
alter table employee add constraint chk_ename
check(regexp_like (e_name,  ’^ [[:alpha:]]+$’))
```

Result:

This will constrain the e_name column to only contain alphanumeric characters (like, no spaces or punctuations are allowed)

## Test Your Understanding

1. Construct the regular expression that can fetch details of employees having ‘a’ or ‘e’ or ‘m’ in their names.

2. Construct the regular expression that fetches the details of those departments having ‘a’ two times in their names.

## Recap

In this course we have learnt about the following:

Regular Expressions are used for

- performing complex string processing and manipulation.
- Data Validation, Identification of duplicate word occurrences , detecting of extraneous white spaces and parsing strings.
- determining valid formats of phone numbers, zip codes, social security numbers, IP addresses, file and path names and so on.
- locating patterns such as HTML tags, email addresses, numbers or dates.

## Re-State Objectives

After completing this chapter, in the next 60 minutes you will be able to :

- Implement Regular expression functions to retrieve data which matches a specific pattern
- Write at least one constraint on the table with the usage of regular Expression for implementing Data Validation

## Thank You

You have successfully completed Regular Expression Support