**15 MARCH**

**PostgreSQL single row functions:**

* Mathematical Functions - perform mathematical operations on a given value or expression, such as ABS(), CEIL(), FLOOR(), ROUND() etc.
* String Functions - manipulate and operate on strings, such as CONCAT(), LENGTH(), LOWER(), UPPER() etc.
* Date and Time Functions - manipulate and operate on date and time data types, such as NOW(), DATE\_PART(), DATE\_TRUNC(), EXTRACT() etc.
* Conditional Functions - return different values based on a given condition, such as CASE, COALESCE(), NULLIF() etc.
* Conversion Functions - convert data types to other data types, such as CAST(), TO\_CHAR(), TO\_DATE() etc.
* Aggregate Functions - operate on groups of rows and return a single result for each group, such as AVG(), COUNT(), MAX(), MIN(), SUM() etc.

**PostgreSQL single row functions on strings:**

* LENGTH(string) - returns the number of characters in a string.
* LOWER(string) - converts all characters in a string to lowercase.
* UPPER(string) - converts all characters in a string to uppercase.
* INITCAP(string) - capitalizes the first letter of each word in a string.
* SUBSTRING(string, from, to) - extracts a substring from a string based on the specified starting and ending positions.
* REPLACE(string, old, new) - replaces all occurrences of a substring in a string with another substring.
* TRIM(string) - removes any leading or trailing spaces from a string.
* CONCAT(string1, string2) - concatenates two strings together.
* SPLIT\_PART(string, delimiter, position) - splits a string into substrings based on a delimiter, and returns the substring at the specified position.
* STRPOS(string, substring) - returns the position of the first occurrence of a substring in a string.

Yes, you can give regular expressions as arguments to some PostgreSQL single row functions, such as

* REGEXP\_REPLACE()
* REGEXP\_MATCH()
* REGEXP\_SPLIT\_TO\_ARRAY()
* REGEXP\_SPLIT\_TO\_TABLE()

REGEXP\_REPLACE() - replaces all occurrences of a pattern in a string with a replacement string. The first argument is the input string, the second argument is the pattern to be replaced, and the third argument is the replacement string.

SELECT REGEXP\_REPLACE('Hello World', 'o', '0'); -- returns 'Hell0 W0rld'

In this example, the regular expression 'o' is used to find all occurrences of the letter "o" in the input string "Hello World", and replace them with the number "0".

**ASSIGNMENT**

CREATE TABLE emails( email text );

INSERT INTO emails VALUES ('ravathujahnavi\*\*@gmail.com');

INSERT INTO emails VALUES ('ravathu12jahnavi@gmail.com');

INSERT INTO emails VALUES ('ravathu12@gmail.com');

INSERT INTO emails VALUES ('12ravathu@gmail.com');

INSERT INTO emails VALUES ('122$\*#ravathu2001@gmail.com');

**Query to get user name:**

SELECT REGEXP\_REPLACE( SPLIT\_PART(email,'@',1),'[^[:alpha:]]','','g') AS name FROM emails;

**Query to get domain name:**

SELECT SUBSTRING(

SPLIT\_PART(email,'@',2),

1,

POSITION('.' in SPLIT\_PART(email, '@', 2))-1

)

AS domain\_name FROM emails;

**NOTE**:

1. The username can have any characters in it along with the name, except ‘@’.