POSTGRESQL CLASS 6

SCALAR AND AGGREGATE FUNCTIONS

UPPER AND LOWER

- In PostgreSQL, the UPPER function is used to convert a string into upper case.
- The LOWER function is used to convert a string into lower case.
- The syntax for UPPER function is:
- SELECT UPPER(columnName) FROM tableName;
- SELECT LOWER(columnName) FROM tableName;

LENGTH

- This is a function used to get the length of characters in a string
- SELECT LENGTH(columnName) FROM tableName;

FORMAT

- PostgreSQL FORMAT() function formats arguments based on a format string.
- These are the arguments it can take
- 1) format_string

- The format_string argument is a string that specifies how the result string should be formatted.

- The format_string consists of text and format specifiers. Text is copied directly to the result string while the format specifiers are placeholders for the arguments to be inserted into the result string.

- The following shows the syntax of the format specifier:

- %[position][flags][width]type

FORMAT

- A format specifier starts with % character. It has three optional components position, flags, withand a required component type.
- position
- The position specifies which argument to be inserted in the result string. The position is in the form n$ where n is the argument index. The first argument starts from 1.
- If you omit the position component, the default is the next argument in the list.

FORMAT

- type
- type is the type of format conversion to use to produce the format specifier's output.
- The permitted values for type argument are as follows:
- s formats the argument value as a string. NULL is treated as an empty string.
- I treats the argument value as an SQL identifier.
- L quotes the argument value as an SQL literal.
- We often use I and L for constructing dynamic SQL statements.
- If you want to include % in the result string, use double percentages %%

FORMAT

- SELECT FORMAT('Hello, %s','PostgreSQL');
- This will output ‘Hello, World’
- SELECT
- FORMAT('%s, %s',last_name, first_name) full_name
- FROM
- customer;
- ORDER BY
- full_name;

NOW

- We can get the current date and time by using the PostgreSQL NOW() function. The PostgreSQL NOW() function's return type is the timestamp with the time zone. Depending upon the current database server's time zone setting, the PostgreSQL NOW() function returns us the current date and time. We can adjust the result returned by the PostgreSQL NOW() function to other timezones as well. Also, we can get the current date and time without a timezone. This function can be used for giving the default value to the column of the table.
- SELECT NOW()

SUBSTRING

- The PostgreSQL substring function is used to extract a string containing a specific number of characters from a particular position of a given string.
- SUBSTRING('people',2,3)
- This will output: 'eop'

IF

- The if statement determines which statements to execute based on the result of a boolean expression.
- PL/pgSQL provides you with three forms of the if statements.

- if then
- if then else
- if then elsif

IF

- **If-then statements**
- *if condition then*
- *statements;*
- *end if;*
- The if statement executes statements if a condition is true. If the condition evaluates to false, the control is passed to the next statement after the END if part.

IF

- **if-then-else statements**
- *if condition then*
- *statements;*
- *Else*
- *Alternative-statements*
- *end if;*
- The if then else statement executes the statements in the if branch if the condition evaluates to true; otherwise, it executes the statements in the else branch..

IF

- **if-then-elseif-then else statements**
- *if condition_1 then*
- *statement_1;*
- *elsif condition_2 then*
- *statement_2*
- else
- else-statement;
- end if;
- The if and ifthen else statements evaluate one condition. However, the if then elsif statement evaluates multiple conditions.
- If a condition is true, the corresponding statement in that branch is executed.
- For example, if the condition_1 is true then the if then ELSif executes the statement_1 and stops evaluating the other conditions.
- If all conditions evaluate to false, the if then elsif executes the statements in the else branch.

REPLACE

- Sometimes, you want to search and replace a string in a column with a new one such as replacing outdated phone numbers, broken URLs, and spelling mistakes.
- To search and replace all occurrences of a string with a new one, you use the REPLACE() function.
- The syntax is: REPLACE(source, old_text, new_text);
- SELECT
- REPLACE ('ABC AA', 'A', 'Z');
- This will ouput : “ZBC ZZ”

ROUND()

- The PostgreSQL ROUND() function rounds a numeric value to its nearest integer or a number with the number of decimal places.
- The following illustrates the syntax of the ROUND() function:
- ROUND (source [, n]);
- **Arguments**
- The ROUND() function accepts 2 arguments:
- 1) source
- The source argument is a number or a numeric expression that is to be rounded.
- 2) n
- The n argument is an integer that determines the number of decimal places after rounding.
- The n argument is optional. If you omit the n argument, its default value is 0.
- The ROUND() function returns a result whose type is the same as the input if you omit the second argument.
- In case if you use both arguments, the ROUND() function returns a numeric value.

ROUND()

- SELECT
- ROUND(10.5);
- The result is:
- 11
- SELECT
- ROUND(10.812, 2);
- The result is
- 10.81