POSTGRESQL - WHERE CLAUSE

The PostgreSQL WHERE clause is used to specify a condition while fetching the data from single table or joining with multiple tables.

If the given condition is satisfied, only then it returns specific value from the table. You can filter out rows that you don't want included in the result-set by using the WHERE clause.

The WHERE clause not only is used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc., which we would examine in subsequent chapters.

Syntax

The basic syntax of SELECT statement with WHERE clause is as follows:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [search_condition]
```

You can specify a *search_condition* using comparison or logical operators. like >, <, =, LIKE, NOT, etc. Below examples would make this concept clear.

Example:

Consider the table COMPANY having records as follows:

```
testdb# select * from COMPANY;
 id | name  | age | address   | salary
----+-------+-----+-----------+--------
  1 | Paul  |  32 | California|  20000
  2 | Allen |  25 | Texas     |  15000
  3 | Teddy |  23 | Norway    |  20000
  4 | Mark  |  25 | Rich-Mond |  65000
  5 | David |  27 | Texas     |  85000
  6 | Kim   |  22 | South-Hall|  45000
  7 | James |  24 | Houston   |  10000
(7 rows)
```

Here are simple examples showing usage of PostgreSQL Logical Operators. Following SELECT statement will list down all the records where AGE is greater than or equal to 25 **AND** salary is greater than or equal to 65000.00:

```
testdb=# SELECT * FROM COMPANY WHERE AGE >= 25 AND SALARY >= 65000;
```

Above PostgreSQL statement will produce the following result:

```
 id | name  | age |  address   | salary
----+-------+-----+------------+--------
  4 | Mark  |  25 | Rich-Mond  |  65000
  5 | David |  27 | Texas      |  85000
(2 rows)
```

Following SELECT statement lists down all the records where AGE is greater than or equal to 25 **OR** salary is greater than or equal to 65000.00:

```
testdb=# SELECT * FROM COMPANY WHERE AGE >= 25 OR SALARY >= 65000;
```

Above PostgreSQL statement will produce the following result:

```
 id | name  | age | address     | salary
```

```
----+-------+-----+-------------+--------
  1 | Paul  |  32 | California  |  20000
  2 | Allen |  25 | Texas       |  15000
  4 | Mark  |  25 | Rich-Mond   |  65000
  5 | David |  27 | Texas       |  85000
(4 rows)
```

Following SELECT statement lists down all the records where AGE is not NULL which means all the records because none of the record is having AGE equal to NULL:

```
testdb=#  SELECT * FROM COMPANY WHERE AGE IS NOT NULL;
```

Above PostgreSQL statement will produce the following result:

```
  id | name  | age | address    | salary
 ----+-------+-----+------------+--------
   1 | Paul  |  32 | California |  20000
   2 | Allen |  25 | Texas      |  15000
   3 | Teddy |  23 | Norway     |  20000
   4 | Mark  |  25 | Rich-Mond  |  65000
   5 | David |  27 | Texas      |  85000
   6 | Kim   |  22 | South-Hall |  45000
   7 | James |  24 | Houston    |  10000
(7 rows)
```

Following SELECT statement lists down all the records where NAME starts with 'Pa', does not matter what comes after 'Pa'.

```
testdb=# SELECT * FROM COMPANY WHERE NAME LIKE 'Pa%';
```

Above PostgreSQL statement will produce the following result:

```
 id | name | age |address    | salary
----+------+-----+-----------+--------
  1 | Paul |  32 | California|  20000
```

Following SELECT statement lists down all the records where AGE value is either 25 or 27:

```
testdb=# SELECT * FROM COMPANY WHERE AGE IN ( 25, 27 );
```

Above PostgreSQL statement will produce the following result:

```
 id | name  | age | address    | salary
----+-------+-----+------------+--------
  2 | Allen |  25 | Texas      |  15000
  4 | Mark  |  25 | Rich-Mond  |  65000
  5 | David |  27 | Texas      |  85000
(3 rows)
```

Following SELECT statement lists down all the records where AGE value is neither 25 nor 27:

```
testdb=# SELECT * FROM COMPANY WHERE AGE NOT IN ( 25, 27 );
```

Above PostgreSQL statement will produce the following result:

```
 id | name  | age | address    | salary
----+-------+-----+------------+--------
  1 | Paul  |  32 | California |  20000
  3 | Teddy |  23 | Norway     |  20000
  6 | Kim   |  22 | South-Hall |  45000
  7 | James |  24 | Houston    |  10000
(4 rows)
```

Following SELECT statement lists down all the records where AGE value is in BETWEEN 25 AND 27:

```
testdb=# SELECT * FROM COMPANY WHERE AGE BETWEEN 25 AND 27;
```

Above PostgreSQL statement will produce the following result:

```
 id | name  | age | address    | salary
----+-------+-----+------------+--------
  2 | Allen |  25 | Texas      |  15000
  4 | Mark  |  25 | Rich-Mond  |  65000
  5 | David |  27 | Texas      |  85000
(3 rows)
```

Following SELECT statement makes use of SQL subquery where subquery finds all the records with AGE field having SALARY > 65000 and later WHERE clause is being used along with EXISTS operator to list down all the records where AGE from the outside query exists in the result returned by sub-query:

```
testdb=# SELECT AGE FROM COMPANY
        WHERE EXISTS (SELECT AGE FROM COMPANY WHERE SALARY > 65000);
```

Above PostgreSQL statement will produce the following result:

```
 age
-----
  32
  25
  23
  25
  27
  22
  24
(7 rows)
```

Following SELECT statement makes use of SQL subquery where subquery finds all the records with AGE field having SALARY > 65000 and later WHERE clause is being used along with > operator to list down all the records where AGE from outside query is greater than the age in the result returned by sub-query:

```
testdb=# SELECT * FROM COMPANY
        WHERE AGE > (SELECT AGE FROM COMPANY WHERE SALARY > 65000);
```

Above PostgreSQL statement will produce the following result:

```
 id | name | age | address    | salary
----+------+-----+------------+--------
  1 | Paul |  32 | California |  20000
```