

College of Engineering, Trivandrum

Department of Computer Science and Engineering



CS333 APPLICATION SOFTWARE DEVELOPMENT LAB

LABORATORY REPORT 3

Learning Basic SQL queries(Part 1)

Student Name

1. Justine Biju(S5)

Student ID

170445(Roll No:37)

Submission Date : 29/07/2019

1 Introduction

A Database usually involves in addition, deletion, updating and querying of data. SQL provides many queries to work with but there are four queries that plays a crucial role in any Database Management System. They are:

1. SELECT

- (a) A SELECT statement retrieves zero or more rows from one or more database tables or database views.
- (b) SELECT is the most commonly used data query language (DQL) command.
- (c) The SELECT statement has many optional clauses like WHERE, GROUP BY, HAVING etc.

2. INSERT

- (a) The INSERT INTO statement is used to insert new records in a table.
- (b) It has a compulsory clause VALUES where the values are entered in order of the selected columns.

3. UPDATE

- (a) The UPDATE statement is used to modify the existing records in a table.
- (b) It has two clauses SET and WHERE.

4. DELETE

- (a) The DELETE statement is used to delete existing records in a table.
- (b) It has a WHERE clause used to specify the checking condition.

All these SQL queries are supported by PostgreSQL and can be implemented in a similar fashion.

2 Implementation in PostgreSQL

1. First we need to create a database to work with.
This can be done using the following query:

```
CREATE DATABASE test;
```

To ensure the creation of the database we can list all the existing databases using:

```
\l
```

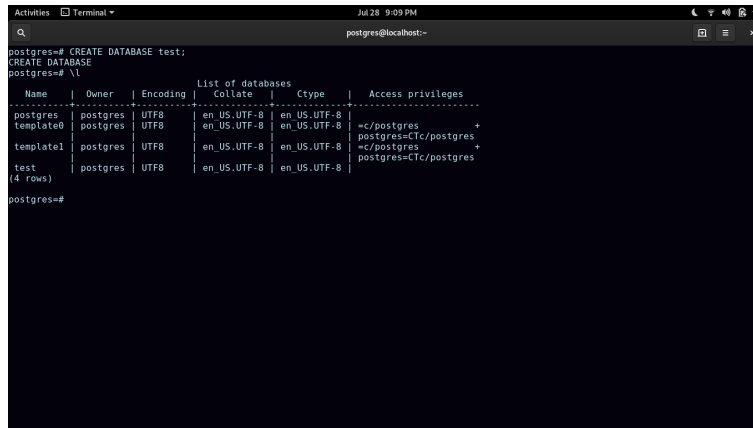


Figure 1: Creating Database 'test'

2. Next we need to connect to the newly created database. This is done as:

```
\c test
```

We can view the tables present in the database. Since this is a newly created database we won't have any tables but, we can do this by the following command.

```
\d
```

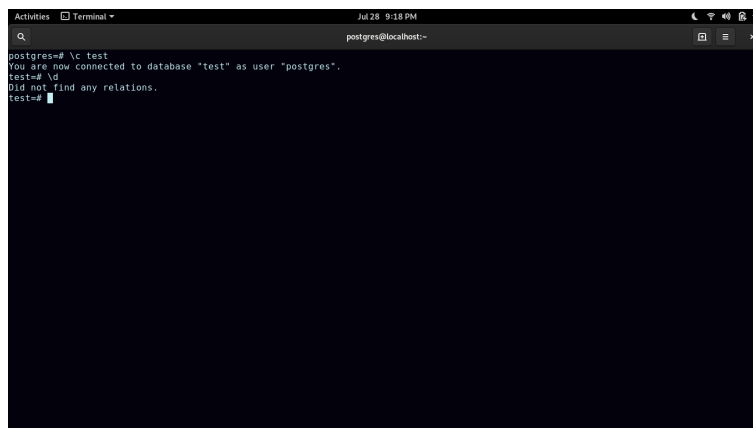
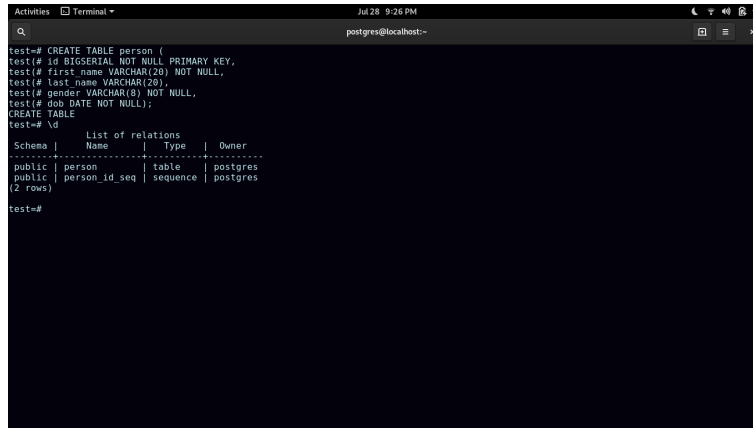


Figure 2: Connecting to database 'test'

3. We can now create a new table using the following query

```
CREATE TABLE person (  
  id BIGSERIAL NOT NULL PRIMARY KEY,  
  first_name VARCHAR(20) NOT NULL,  
  last_name VARCHAR(20),  
  gender VARCHAR(8) NOT NULL,  
  dob DATE NOT NULL);
```

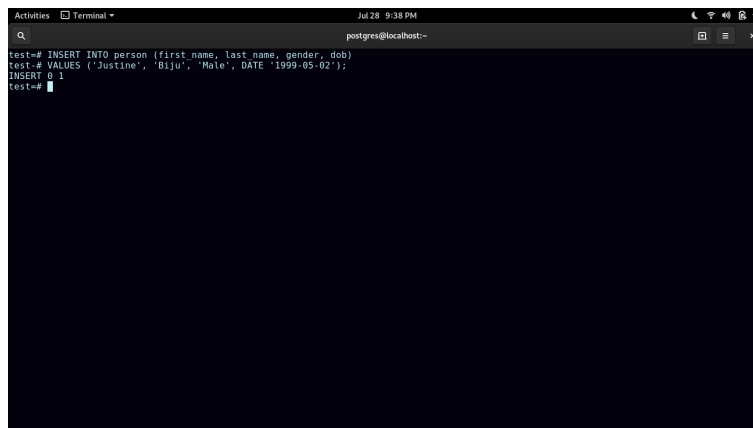


```
test=# CREATE TABLE person (  
test=# id BIGSERIAL NOT NULL PRIMARY KEY,  
test=# first_name VARCHAR(20) NOT NULL,  
test=# last_name VARCHAR(20),  
test=# gender VARCHAR(8) NOT NULL,  
test=# dob DATE NOT NULL);  
CREATE TABLE  
test=# \d  
      List of relations  
Schema | Name          | Type  | Owner  
-----|-----|-----|-----  
public | person        | table  | postgres  
public | person_id_seq | sequence | postgres  
(2 rows)  
test=#
```

Figure 3: Creating table 'person'

4. We can now enter values using the INSERT INTO query.

```
INSERT INTO person (first_name , last_name , gender ,  
  dob)  
VALUES ( 'Justine' , 'Biju' , 'Male' ,  
  DATE '1999-02-05');
```



```
test=# INSERT INTO person (first_name, last_name, gender, dob)  
test=# VALUES ('Justine', 'Biju', 'Male', DATE '1999-05-02');  
INSERT 0 1  
test=#
```

Figure 4: Inserting details of a person

5. To ensure the insertion of the person's details we can use the SELECT query to display the contents of the table.

```
SELECT * FROM person;
```

```
Activities Terminal Jul 28 9:40 PM postgres@localhost:~
test=# INSERT INTO person (first_name, last_name, gender, dob)
test=# VALUES ('Justine', 'Biju', 'Male', DATE '1999-05-02');
INSERT 0 1
test=# SELECT * FROM person;
 id | first_name | last_name | gender | dob
-----+-----+-----+-----+-----
  1 | Justine    | Biju      | Male   | 1999-05-02
(1 row)
test=#
```

Figure 5: Viewing the entire table 'person'

6. Now suppose we need to update a person's details, we can do that by the UPDATE query.

```
UPDATE person
SET last_name = 'B'
WHERE first_name = 'Justine' AND id = 1;

SELECT * FROM person;
```

```
Activities Terminal Jul 28 9:46 PM postgres@localhost:~
test=# UPDATE person
test=# SET last_name = 'B'
test=# WHERE first_name = 'Justine' AND id = '1';
UPDATE 1
test=# SELECT * FROM person;
 id | first_name | last_name | gender | dob
-----+-----+-----+-----+-----
  1 | Justine    | B         | Male   | 1999-05-02
(1 row)
test=#
```

Figure 6: Updating a record

7. Lastly, lets delete a record from the table using the DELETE query.

```
DELETE FROM person
WHERE id = 1;

SELECT * FROM person;
```

```
Activities Terminal Jul 28 9:51 PM postgres@localhost:~
test=# DELETE FROM person
test=# WHERE id = 1;
DELETE 1
test=# SELECT * FROM person;
 id | first_name | last_name | gender | dob
-----+-----+-----+-----+-----
(0 rows)
test=#
```

Figure 7: Deleting a record

3 Questions

1. Display the details of all the employees.

```
SELECT * FROM Employee;
```

```
Activities Terminal Jul 29 9:13 PM postgres@localhost:~
exp3=# SELECT * FROM Employee;
 emp_id | emp_name | dept | Salary(in US $)
-----+-----+-----+-----
 1 | Micheal | Production | 3500
 2 | Joe | Production | 2500
 3 | Smith | Sales | 2250
 4 | David | Marketing | 2600
 5 | Richard | Sales | 1600
 6 | Jessy | Marketing | 1800
 7 | Jane | Sales | 2000
 8 | Janet | Production | 3000
 9 | Neville | Marketing | 2750
10 | Richardson | Sales | 1800
(10 rows)
exp3=#
```

Figure 8: Question 1

2. Display the names and id's of all employees.

```
SELECT Emp_id , Emp_name
FROM Employee;
```

```

exp3=# SELECT Emp_id, Emp_name FROM Employee;
emp_id | emp_name
-----+-----
1 | Micheal
2 | Joe
3 | Smith
4 | David
5 | Richard
6 | Jessie
7 | Jane
8 | Janet
9 | Neville
10 | Richardson
(10 rows)
exp3=#

```

Figure 9: Question 2

3. Delete the entry corresponding to employee id:10.

```

DELETE FROM Employee
WHERE Emp_id = 10;

SELECT * FROM Employee;

```

```

exp3=# DELETE FROM Employee
exp3=# WHERE Emp_id = 10;
DELETE 1
exp3=# SELECT * FROM Employee;
emp_id | emp_name | dept | Salary(in US $)
-----+-----+-----+-----
1 | Micheal | Production | 2500
2 | Joe | Production | 2500
3 | Smith | Sales | 2250
4 | David | Marketing | 2900
5 | Richard | Sales | 1600
6 | Jessie | Marketing | 1800
7 | Jane | Sales | 2000
8 | Janet | Production | 3000
9 | Neville | Marketing | 2750
(9 rows)
exp3=#

```

Figure 10: Question 3

4. Insert a new tuple to the table. The salary field of the new employee should be kept NULL.

```

INSERT INTO Employee(Emp_name, Dept)
VALUES ( 'Johnson' , 'Sales' );

SELECT * FROM Emplolyee;

```

```
Activities Terminal Jul 29 10:44 PM postgres@localhost:~
exp3=# INSERT INTO Employee(Emp_name, Dept)
VALUES ('Johnson', 'Sales');
INSERT 0 1
exp3=# SELECT * FROM Employee;
 emp_id | emp_name | dept | Salary(in US $)
-----+-----+-----+-----
 1 | Micheal | Production | 2500
 2 | Joe | Production | 2500
 3 | Smith | Sales | 2250
 4 | David | Marketing | 2900
 5 | Richard | Sales | 1600
 6 | Jessy | Marketing | 1800
 7 | Jane | Sales | 2600
 8 | Janet | Production | 3800
 9 | Neville | Marketing | 2750
11 | Johnson | Sales |
(10 rows)
exp3=#
```

Figure 11: Question 4

5. Find the details of all employees working in the marketing department.

```
SELECT * FROM Employee
WHERE Dept = 'Marketing';
```

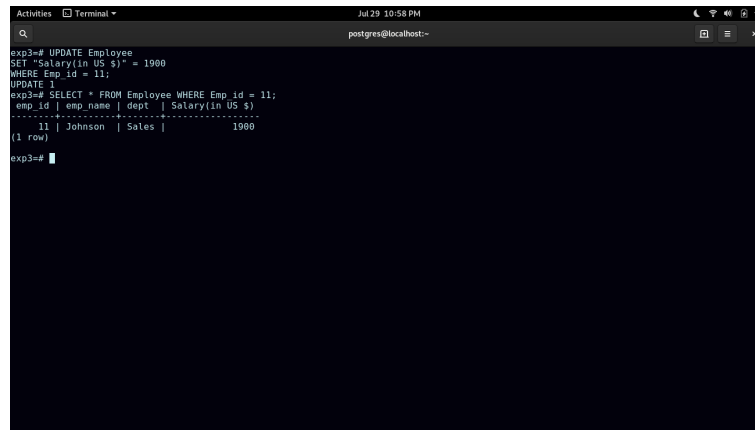
```
Activities Terminal Jul 29 10:49 PM postgres@localhost:~
exp3=# SELECT * FROM Employee
exp3=# WHERE Dept = 'Marketing';
 emp_id | emp_name | dept | Salary(in US $)
-----+-----+-----+-----
 4 | David | Marketing | 2900
 6 | Jessy | Marketing | 1800
 9 | Neville | Marketing | 2750
(3 rows)
exp3=#
```

Figure 12: Question 5

6. Add the salary details of the newly added employee.

```
UPDATE Employee
SET "Salary(in US $)" = 1900
WHERE Emp_id = 11;

SELECT * FROM Employee
WHERE Emp_id = 11;
```

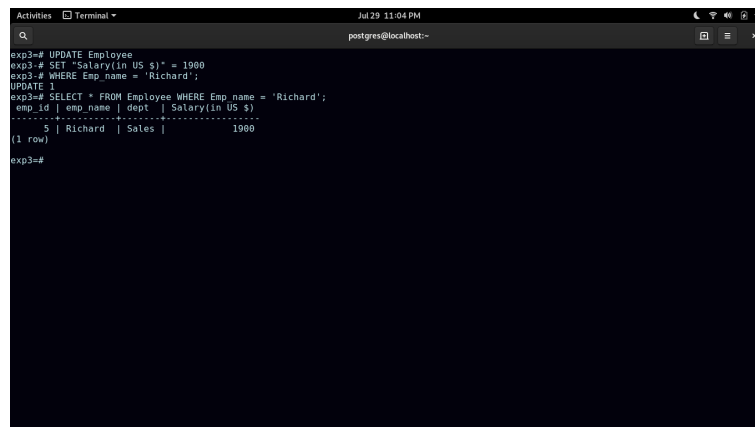
```
exp3=# UPDATE Employee
SET "Salary(in US $)" = 1900
WHERE Emp_id = 11;
UPDATE 1
exp3=# SELECT * FROM Employee WHERE Emp_id = 11;
 emp_id | emp_name | dept | Salary(in US $)
-----+-----+-----+-----
      11 | Johnson  | Sales |           1900
(1 row)
exp3=#
```

Figure 13: Question 6

7. Update the salary of Richard to 1900\$.

```
UPDATE Employee
SET "Salary(in US $)" = 1900
WHERE Emp_name = 'Richard';

SELECT * FROM Employee
WHERE Emp_name = 'Richard';
```

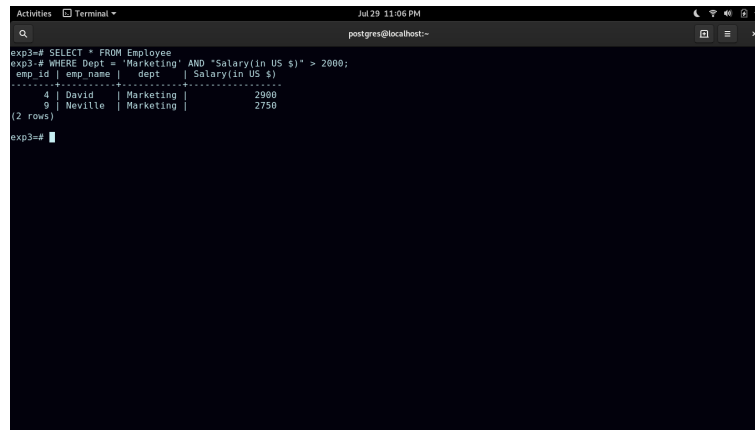


```
exp3=# UPDATE Employee
exp3=# SET "Salary(in US $)" = 1900
exp3=# WHERE Emp_name = 'Richard';
UPDATE 1
exp3=# SELECT * FROM Employee WHERE Emp_name = 'Richard';
 emp_id | emp_name | dept | Salary(in US $)
-----+-----+-----+-----
       5 | Richard  | Sales |           1900
(1 row)
exp3=#
```

Figure 14: Question 7

8. Find the details of all employees who are working for marketing and has a salary greater than 2000\$.

```
SELECT * FROM Employee
WHERE Dept = 'Marketing' AND
"Salary(in US $)" > 2000;
```



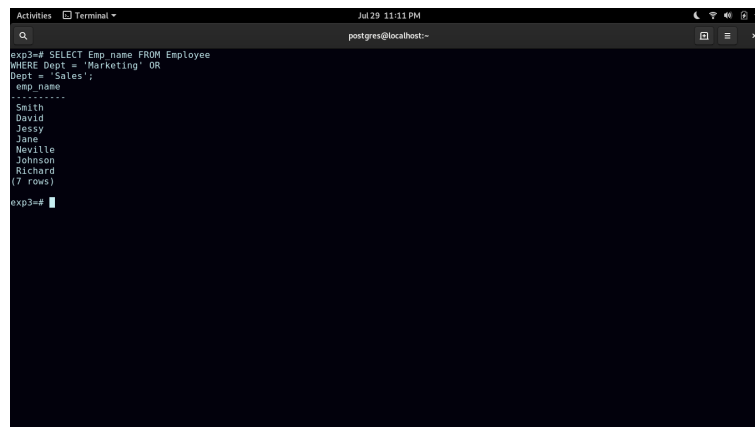
```
exp3=# SELECT * FROM Employee
exp3=# WHERE Dept = 'Marketing' AND "Salary(in US $)" > 2000;
 emp_id | emp_name | dept | Salary(in US $)
-----+-----+-----+-----
      4 | David   | Marketing | 2900
      9 | Neville | Marketing | 2750
(2 rows)

exp3=#
```

Figure 15: Question 8

9. List the names of all employees working in the sales department or marketing department

```
SELECT Emp_name FROM Employee
WHERE Dept = 'Marketing' OR
Dept = 'Sales';
```



```
exp3=# SELECT Emp_name FROM Employee
WHERE Dept = 'Marketing' OR
Dept = 'Sales';
 emp_name
-----
Smith
David
Jessy
Jane
Neville
Johnson
Richard
(7 rows)

exp3=#
```

Figure 16: Question 9

10. List the names and department of all employees whose salary is between 2300\$ and 3000\$.

```
SELECT * FROM Employee
WHERE "Salary(in US $)" BETWEEN
2300 AND 3000;
```

```

exp3=# SELECT * FROM Employee
WHERE "Salary(in US $)" BETWEEN 2300 AND 3000;
 emp_id | emp_name | dept   | Salary(in US $)
-----+-----+-----+-----
      1 | Micheal  | Production | 2500
      2 | Joe      | Production | 2500
      4 | David    | Marketing | 2900
      8 | Janet    | Production | 3000
      9 | Neville  | Marketing | 2750
(5 rows)

exp3=#

```

Figure 17: Question 10

11. Update the salary of all employees working in production department 12%.

```

UPDATE Employee
SET "Salary(in US $)" = "Salary(in US $)" * 1.12
WHERE Dept = 'Production';

```

```

exp3=# SELECT * FROM Employee;
 emp_id | emp_name | dept   | Salary(in US $)
-----+-----+-----+-----
      1 | Micheal  | Production | 2500
      2 | Joe      | Production | 2500
      3 | Smith    | Sales     | 2250
      4 | David    | Marketing | 2900
      6 | Jessie   | Marketing | 1800
      7 | Jane     | Sales     | 2000
      8 | Janet    | Production | 3000
      9 | Neville  | Marketing | 2750
     11 | Johnson  | Sales     | 1900
      5 | Richard  | Sales     | 1900
(10 rows)

exp3=# UPDATE Employee
SET "Salary(in US $)" = "Salary(in US $)" * 1.12
WHERE Dept = 'Production';
UPDATE 3
exp3=# SELECT * FROM Employee;
 emp_id | emp_name | dept   | Salary(in US $)
-----+-----+-----+-----
      3 | Smith    | Sales     | 2250
      4 | David    | Marketing | 2900
      6 | Jessie   | Marketing | 1800
      7 | Jane     | Sales     | 2000
      9 | Neville  | Marketing | 2750
     11 | Johnson  | Sales     | 1900
      5 | Richard  | Sales     | 1900
      1 | Micheal  | Production | 2800
      2 | Joe      | Production | 2800
      8 | Janet    | Production | 3360
(10 rows)

exp3=#

```

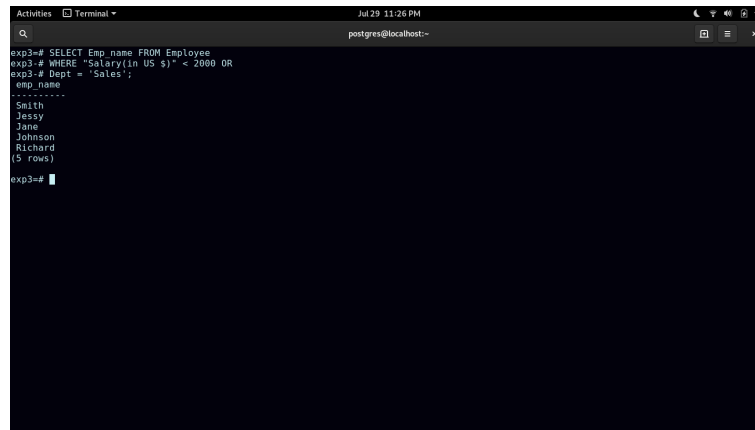
Figure 18: Question 11

12. Display the names of all employees whose salary is less than 2000\$ or working for sales department.

```

SELECT Emp_name FROM Employee
WHERE "Salary(in US $)" < 2000 OR
Dept = 'Sales';

```

A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The user enters a SQL query: 'exp3=# SELECT Emp_name FROM Employee WHERE "Salary(in US \$)" < 2000 OR Dept = 'Sales';'. The output shows a list of employee names: 'emp_name', '-----', 'Smith', 'Jessy', 'Jane', 'Johnson', 'Richard', and '(5 rows)'. The prompt returns to 'exp3=#'.

```
exp3=# SELECT Emp_name FROM Employee
exp3=# WHERE "Salary(in US $)" < 2000 OR
exp3=# Dept = 'Sales';
 emp_name
-----
 Smith
 Jessy
  Jane
Johnson
 Richard
(5 rows)

exp3=#
```

Figure 19: Question 12

4 Result

- Learned some basic PostgreSQL commands.
- Learned to set up and connect to a database.
- Learned to create a new table in a database.
- Successfully implemented SELECT, INSERT, UPDATE and DELETE queries on PostgreSQL.