

# College of Engineering, Trivandrum

Department of Computer Science and Engineering



## CS333 APPLICATION SOFTWARE DEVELOPMENT LAB

---

### LABORATORY REPORT 4

Learning Basic SQL queries(Part 2)

---

**Student Name**

1. Justine Biju(S5)

**Student ID**

170445(Roll No:37)

Submission Date : 29/07/2019

# 1 Introduction

There are many SQL queries that help support out basic queries that SQL offers. Few of what we are going to discuss are listed below:

## 1. ALTER

- (a) The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- (b) The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

## 2. RENAME

- (a) We use this clause along with ALTER to rename a column name.
- (b) TO clause is also used to specify the new name.

## 3. SELECT DISTINCT

- (a) The SELECT DISTINCT statement is used to return only distinct (different) values.
- (b) The output ignores redundant data.

## 4. SQL IN

- (a) The IN operator allows you to specify multiple values in a WHERE clause.
- (b) The IN operator is a shorthand for multiple OR conditions.

## 5. SQL BETWEEN

- (a) The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
- (b) The BETWEEN operator is inclusive: begin and end values are included.

## 6. SQL ALIASES

- (a) SQL aliases are used to give a table, or a column in a table, a temporary name.
- (b) Aliases are often used to make column names more readable.
- (c) An alias only exists for the duration of the query.

## 7. SQL AND

- (a) The AND operator displays a record if all the conditions separated by AND are TRUE.

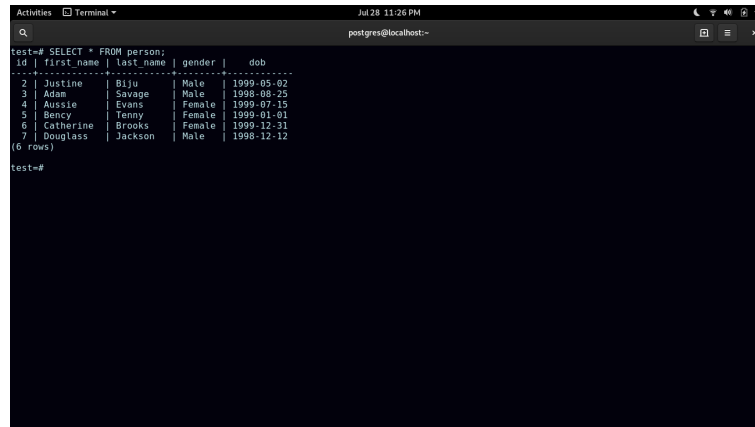
## 8. SQL OR

- (a) The OR operator displays a record if any of the conditions separated by OR is TRUE.

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

## 2 Implementation in PostgreSQL

Let's take a look at the database we are going to work with.



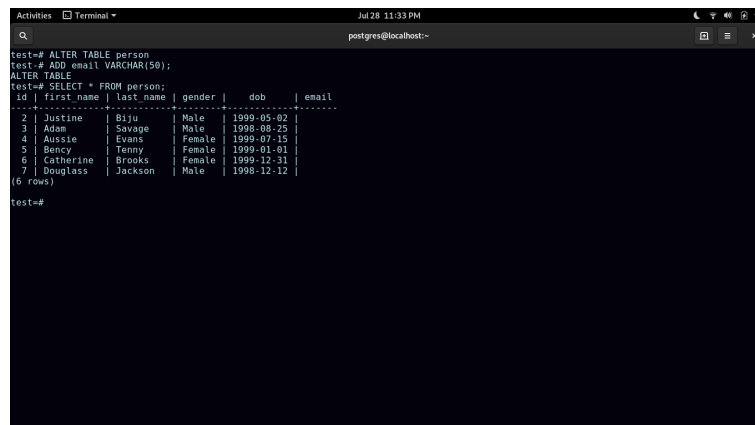
```
test=# SELECT * FROM person;
 id | first_name | last_name | gender | dob
-----+-----+-----+-----+-----
 2 | Justine    | Biju      | Male   | 1999-05-02
 3 | Adam       | Savage    | Male   | 1998-08-25
 4 | Aussie     | Evans     | Female | 1999-07-15
 5 | Bency      | Tenny     | Female | 1999-01-01
 6 | Catherine  | Brooks    | Female | 1999-12-31
 7 | Douglass   | Jackson   | Male   | 1998-12-12
(6 rows)
```

Figure 1: Entities in the table 'person'

1. Lets use the ALTER query to add an email column to the database.  
This can be done using the following query:

```
ALTER TABLE person
ADD email VARCHAR(50);

SELECT * FROM person;
```



```
test=# ALTER TABLE person
test=# ADD email VARCHAR(50);
ALTER TABLE
test=# SELECT * FROM person;
 id | first_name | last_name | gender | dob | email
-----+-----+-----+-----+-----+-----
 2 | Justine    | Biju      | Male   | 1999-05-02 |
 3 | Adam       | Savage    | Male   | 1998-08-25 |
 4 | Aussie     | Evans     | Female | 1999-07-15 |
 5 | Bency      | Tenny     | Female | 1999-01-01 |
 6 | Catherine  | Brooks    | Female | 1999-12-31 |
 7 | Douglass   | Jackson   | Male   | 1998-12-12 |
(6 rows)
```

Figure 2: Adding 'email' column

2. Now lets use ALTER RENAME to rename the 'id' column to 'p\_id'

```
ALTER TABLE person
RENAME id to p_id;

SELECT * FROM person;
```

```

test=# ALTER TABLE person
      RENAME id TO p_id;
ALTER TABLE
test=# SELECT * FROM person;
 p_id | first_name | last_name | gender | dob       | email
-----+-----+-----+-----+-----+-----
  2   | Justine    | Biju     | Male   | 1999-05-02 |
  3   | Adam       | Savage   | Male   | 1998-08-25 |
  4   | Aussie    | Evans    | Female | 1999-07-15 |
  5   | Bency     | Tenny    | Female | 1999-01-01 |
  6   | Catherine | Brooks   | Female | 1999-12-31 |
  7   | Douglass  | Jackson  | Male   | 1998-12-12 |
(6 rows)

test=#

```

Figure 3: Renaming 'id' to 'pid'

3. We can select different entities from a database using the SELECT DISTINCT query.

```

SELECT DISTINCT gender
FROM person;

```

```

test=# SELECT DISTINCT gender
      FROM person;
 gender
-----
 Female
 Male
(2 rows)

test=#

```

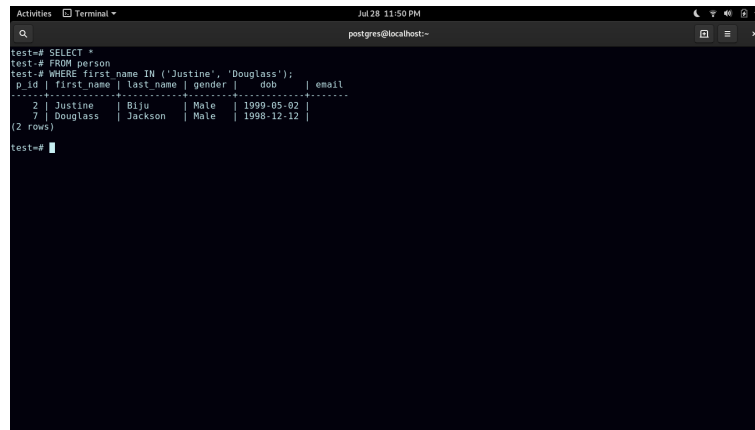
Figure 4: Distinct 'gender' in DB

4. We can use IN to check records with multiple values.

```

SELECT *
FROM person
WHERE first_name IN ('Justine', 'Douglass');

```

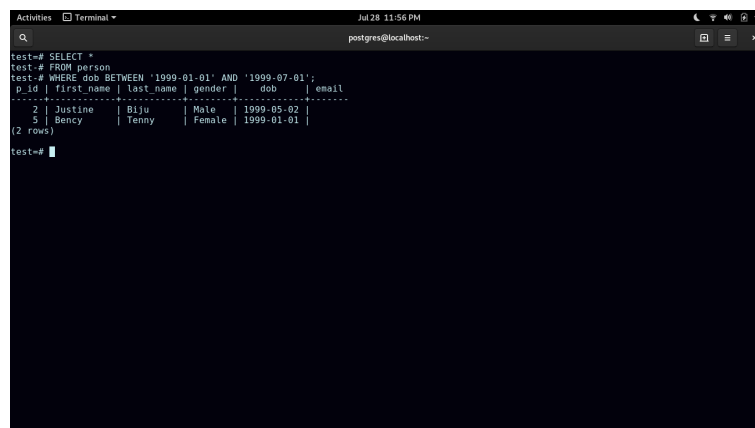


```
test=# SELECT *
test=# FROM person
test=# WHERE first_name IN ('Justine', 'Douglass');
 p_id | first_name | last_name | gender | dob       | email
-----+-----+-----+-----+-----+-----
  2   | Justine    | Biju     | Male   | 1999-05-02 |
  7   | Douglass   | Jackson  | Male   | 1998-12-12 |
(2 rows)
test=#
```

Figure 5: Using the IN query

5. We can use BETWEEN clause in SELECT to display records that have values in a particular range.

```
SELECT *
FROM person
WHERE dob BETWEEN '1999-01-01' AND '1999-07-01';
```

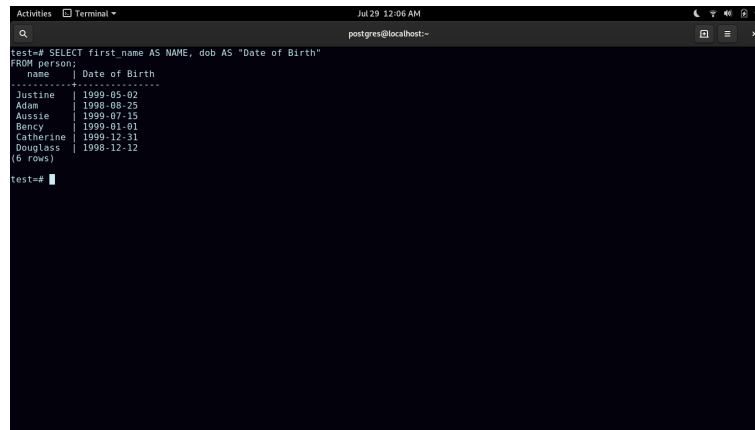


```
test=# SELECT *
test=# FROM person
test=# WHERE dob BETWEEN '1999-01-01' AND '1999-07-01';
 p_id | first_name | last_name | gender | dob       | email
-----+-----+-----+-----+-----+-----
  2   | Justine    | Biju     | Male   | 1999-05-02 |
  5   | Bency     | Tenny    | Female | 1999-01-01 |
(2 rows)
test=#
```

Figure 6: Using BETWEEN clause

6. We can use aliases to make the database to a user.

```
SELECT first_name AS NAME,
dob AS "Date of Birth"
FROM person;
```



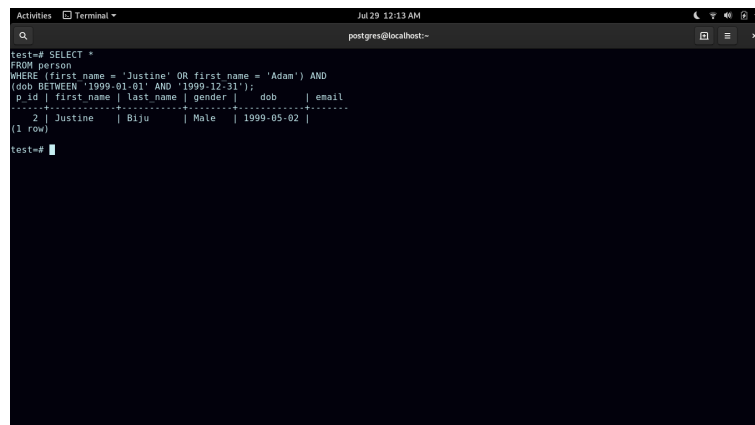
```
test=# SELECT first_name AS NAME, dob AS "Date of Birth"
FROM person;
 name | Date of Birth 
-----|-----
 Justine | 1999-05-02
 Adam | 1998-08-25
 Aussie | 1999-07-15
 Bency | 1999-01-01
 Catherine | 1999-12-31
 Douglass | 1998-12-12
(6 rows)

test=#
```

Figure 7: Using an alias for column name

7. We can use AND and/or OR clause to group conditional statements.

```
SELECT *
FROM person
WHERE (first_name = 'Justine' OR
first_name = 'Adam') AND
(dob BETWEEN '1999-01-01' AND
'1999-12-31');
```

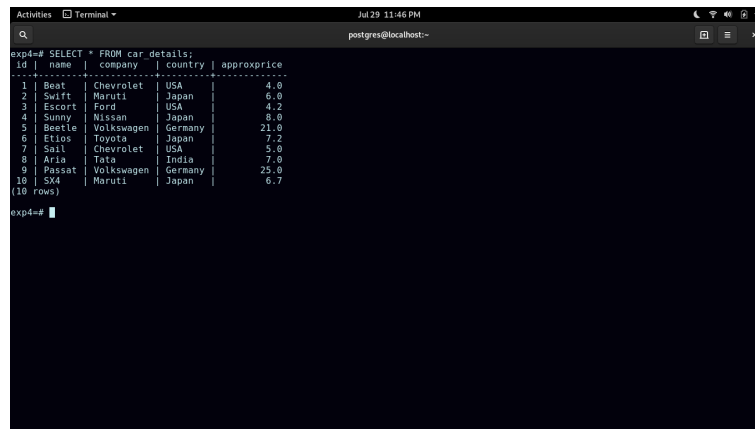


```
test=# SELECT *
FROM person
WHERE (first_name = 'Justine' OR first_name = 'Adam') AND
(dob BETWEEN '1999-01-01' AND '1999-12-31');
 p_id | first_name | last_name | gender | dob      | email 
-----|-----
 2 | Justine | Biju | Male | 1999-05-02 |
(1 row)

test=#
```

Figure 8: Using AND and OR clause

### 3 Questions



A terminal window titled 'Terminal' with a search bar and window controls. It shows a PostgreSQL session on 'postgres@localhost:~'. The user 'exp4=#' has executed the query 'SELECT \* FROM car\_details;'. The result is a table with 10 rows and 5 columns: id, name, company, country, and approxprice. The data is as follows:

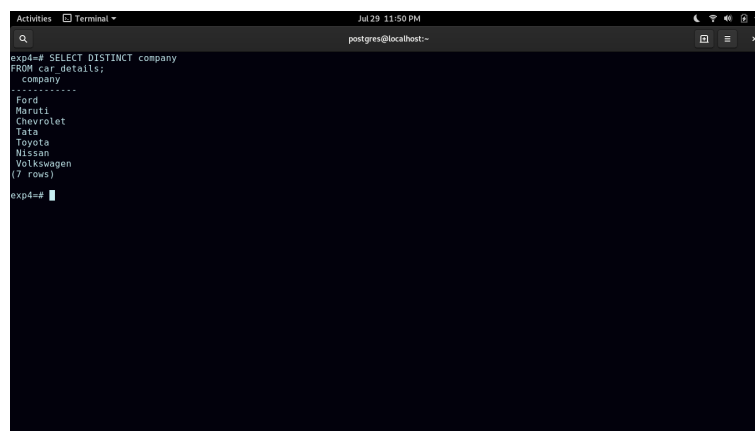
id	name	company	country	approxprice
1	Beat	Chevrolet	USA	4.0
2	Swift	Maruti	Japan	6.0
3	Escort	Ford	USA	4.2
4	Sunny	Nissan	Japan	3.0
5	Beetle	Volkswagen	Germany	21.0
6	Ertos	Toyota	Japan	7.2
7	Sail	Chevrolet	USA	5.0
8	Aria	Tata	India	7.0
9	Passat	Volkswagen	Germany	25.0
10	Sk4	Maruti	Japan	6.7

The terminal shows '(10 rows)' and the prompt 'exp4=#' is ready for the next command.

Figure 9: Entries in 'car\_details'

1. List the names of all companies as mentioned in the database

```
SELECT DISTINCT company
FROM car_details;
```



A terminal window showing the execution of the query 'SELECT DISTINCT company FROM car\_details;'. The result is a list of 7 distinct company names: Ford, Maruti, Chevrolet, Tata, Toyota, Nissan, and Volkswagen. The terminal shows '(7 rows)' and the prompt 'exp4=#' is ready for the next command.

company
Ford
Maruti
Chevrolet
Tata
Toyota
Nissan
Volkswagen

Figure 10: Question 1

2. List the names of all countries having car production companies

```
SELECT DISTINCT country
FROM car_details;
```

```
Activities Terminal Jul 29 11:52 PM postgres@localhost:~
exp4=# SELECT DISTINCT country
FROM car_details;
country
-----
USA
Germany
India
Japan
(4 rows)
exp4=#
```

Figure 11: Question 2

3. List the details of all cars within a price range 4 to 7 lakhs

```
SELECT * FROM car_details
WHERE approxprice BETWEEN
4.0 AND 7.0;
```

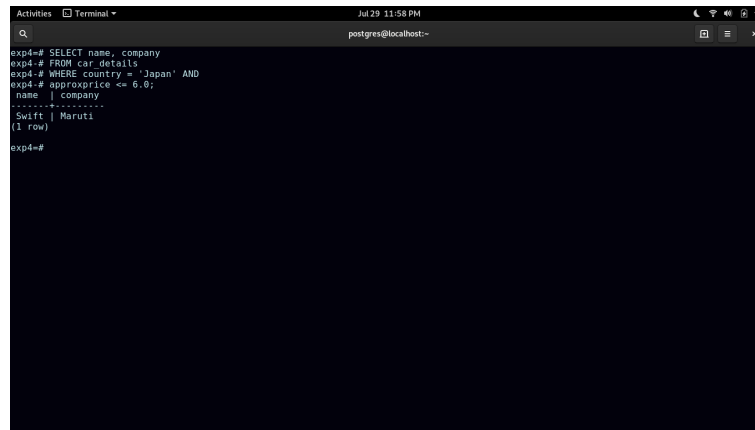
```
Activities Terminal Jul 29 11:55 PM postgres@localhost:~
exp4=# SELECT * FROM car_details
exp4=# WHERE approxprice BETWEEN
exp4=# 4.0 AND 7.0;
 id | name | company | country | approxprice
-----+-----+-----+-----+-----
  1 | Beat | Chevrolet | USA | 4.0
  2 | Swift | Maruti | Japan | 5.0
  3 | Escort | Ford | USA | 4.2
  7 | Sail | Chevrolet | USA | 5.0
  8 | Aris | Tata | India | 7.0
 10 | SX4 | Maruti | Japan | 6.7
(6 rows)
exp4=#
```

Figure 12: Question 3

4. List the name and company of all cars originating from Japan and having price<sub>j</sub>=6 lakhs

```
SELECT name, company
FROM car_details
WHERE country = 'Japan' AND
approxprice <= 6.0;
```



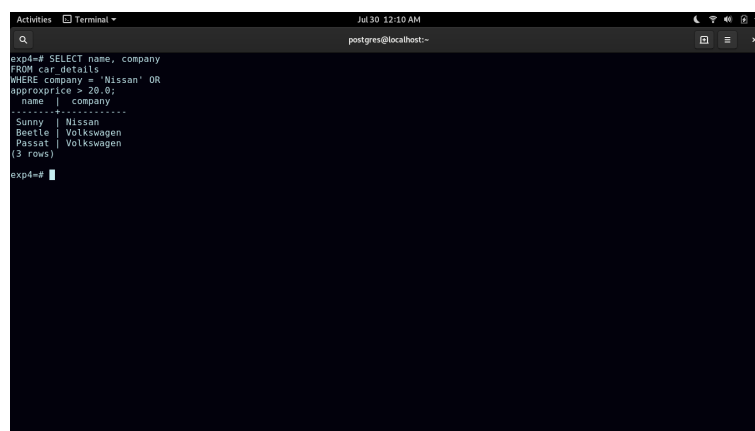


```
exp4=# SELECT name, company
exp4=# FROM car_details
exp4=# WHERE country = 'Japan' AND
exp4=# approxprice <= 6.0;
 name | company
-----+-----
 Swift | Maruti
(1 row)
exp4=#
```

Figure 13: Question 4

5. List the names and the companies of all cars either from Nissan or having a price greater than 20 lakhs.

```
SELECT name, company
FROM car_details
WHERE company = 'Nissan' OR
approxprice > 20.0;
```



```
exp4=# SELECT name, company
exp4=# FROM car_details
exp4=# WHERE company = 'Nissan' OR
exp4=# approxprice > 20.0;
 name | company
-----+-----
 Sunny | Nissan
 Beetle | Volkswagen
 Passat | Volkswagen
(3 rows)
exp4=#
```

Figure 14: Question 5

6. List the names of all cars produced by (Maruti,Ford).Use SQL IN statement.

```
SELECT name
FROM car_details
WHERE company in ( 'Maruti' , 'Ford' );
```

```
Activities Terminal Jul 30 12:12 AM postgres@localhost:~
exp4=# SELECT name
FROM car_details
WHERE company in ('Maruti', 'Ford');
-----
name
-----
Swift
Escort
SX4
(3 rows)
exp4=#
```

Figure 15: Question 6

- Alter the table cars to add a new field year (model release year). Update the year column for all the rows in the database.

```
ALTER TABLE car_details
ADD year INT DEFAULT 2015;

SELECT * FROM car_details;
```

```
Activities Terminal Jul 30 12:15 AM postgres@localhost:~
exp4=# ALTER TABLE car_details
exp4=# ADD year INT DEFAULT 2015;
exp4=# SELECT * FROM car_details;
 id | name | company | country | approxprice | year
-----
 1 | Beat | Chevrolet | USA | 4.0 | 2015
 2 | Swift | Maruti | Japan | 6.0 | 2015
 3 | Escort | Ford | USA | 4.2 | 2015
 4 | Sunny | Nissan | Japan | 8.0 | 2015
 5 | Beetle | Volkswagen | Germany | 21.0 | 2015
 6 | Etios | Toyota | Japan | 7.2 | 2015
 7 | Sail | Chevrolet | USA | 5.0 | 2015
 8 | Aria | Tata | India | 7.0 | 2015
 9 | Passat | Volkswagen | Germany | 25.0 | 2015
10 | SX4 | Maruti | Japan | 6.7 | 2015
(10 rows)
exp4=#
```

Figure 16: Question 7

- Display the names of all cars as Car\_name (while displaying the name attribute should be listed as car\_aliases)

```
SELECT name AS Car_name
FROM car_details;
```

```
Activities Terminal Jul 30 12:18 AM postgres@localhost:~
exp4=# SELECT name AS Car_name
exp4=# FROM car_details;
 car_name
-----
Beat
Swift
Escort
Sunny
Beetle
Etios
Sail
Aria
Passat
SX4
(10 rows)
exp4=#
```

Figure 17: Question 8

9. Rename the attribute name to car\_name

```
ALTER TABLE car_details
RENAME name TO car_name;

SELECT * FROM car_details;
```

```
Activities Terminal Jul 30 12:21 AM postgres@localhost:~
exp4=# ALTER TABLE car_details
exp4=# RENAME name TO car_name;
ALTER TABLE
exp4=# SELECT * FROM car_details;
 id | car_name | company | country | approxprice | year
-----+-----+-----+-----+-----+-----
 1 | Beat     | Chevrolet | USA     | 4.0 | 2015
 2 | Swift    | Maruti   | Japan   | 6.0 | 2015
 3 | Escort   | Ford     | USA     | 4.2 | 2015
 4 | Sunny    | Nissan   | Japan   | 8.0 | 2015
 5 | Beetle   | Volkswagen | Germany | 21.0 | 2015
 6 | Etios    | Toyota   | Japan   | 7.2 | 2015
 7 | Sail     | Chevrolet | USA     | 5.0 | 2015
 8 | Aria     | Tata     | India   | 7.0 | 2015
 9 | Passat   | Volkswagen | Germany | 25.0 | 2015
10 | SX4      | Maruti   | Japan   | 6.7 | 2015
(10 rows)
exp4=#
```

Figure 18: Question 9

10. List the car manufactured by Toyota(to be displayed as cars\_Toyota)

```
SELECT car_name AS CARS_TOYOTA
FROM car_details
WHERE company = 'Toyota ';
```

```
Activities Terminal Jul 30 12:24 AM postgres@localhost:~
exp4=# SELECT car_name AS CARS_TOYOTA
exp4=# FROM car_details
exp4=# WHERE company = 'Toyota';
cars_toyota
-----
Etios
(1 row)
exp4=#
```

Figure 19: Question 10

11. List the details of all cars in alphabetical order

```
SELECT * FROM car_details
ORDER BY car_name;
```

```
Activities Terminal Jul 30 12:27 AM postgres@localhost:~
exp4=# SELECT * FROM car_details
exp4=# ORDER BY car_name;
 id | car_name | company | country | approxprice | year
-----
 0 | Arian | Tata | India | 7.0 | 2015
 1 | Beat | Chevrolet | USA | 4.0 | 2015
 5 | Beetle | Volkswagen | Germany | 21.0 | 2015
 3 | Escort | Ford | USA | 4.2 | 2015
 6 | Etios | Toyota | Japan | 7.2 | 2015
 9 | Passat | Volkswagen | Germany | 25.0 | 2015
 7 | Sail | Chevrolet | USA | 5.0 | 2015
 4 | Sunny | Nissan | Japan | 8.0 | 2015
 2 | Swift | Maruti | Japan | 6.0 | 2015
10 | SX4 | Maruti | Japan | 6.7 | 2015
(10 rows)
exp4=#
```

Figure 20: Question 11

12. List the details of all cars from cheapest to costliest.

```
SELECT * FROM car_details
ORDER BY approxprice;
```

```
Activities Terminal Jul 30 12:29 AM postgres@localhost:~
exp4=# SELECT * FROM car_details
ORDER BY approxprice;
 id | car_name | company | country | approxprice | year
-----+-----+-----+-----+-----+-----
 1 | Beat     | Chevrolet | USA     | 4.0 | 2015
 3 | Escort   | Ford      | USA     | 4.2 | 2015
 7 | Sail     | Chevrolet | USA     | 5.0 | 2015
 2 | Swift    | Maruti    | Japan   | 6.0 | 2015
10 | SX4      | Maruti    | Japan   | 6.7 | 2015
 8 | Aria     | Tata      | India   | 7.0 | 2015
 6 | Etios    | Toyota    | Japan   | 7.2 | 2015
 4 | Sunny    | Nissan    | Japan   | 8.0 | 2015
 5 | Beetle   | Volkswagen | Germany | 21.0 | 2015
 9 | Passat   | Volkswagen | Germany | 25.0 | 2015
(10 rows)

exp4=#
```

Figure 21: Question 12

## 4 Result

- Successfully implemented ALTER, RENAME, SELECT DISTINCT, SQL IN, BETWEEN, AND and OR queries on PostgreSQL.