

# College of Engineering, Trivandrum

Department of Computer Science and Engineering



## CS333 APPLICATION SOFTWARE DEVELOPMENT LAB

---

### LABORATORY REPORT 5

#### Aggregate Functions

---

**Student Name**

1. Justine Biju(S5)

**Student ID**

170445(Roll No:37)

Submission Date : 29/07/2019

# 1 Introduction

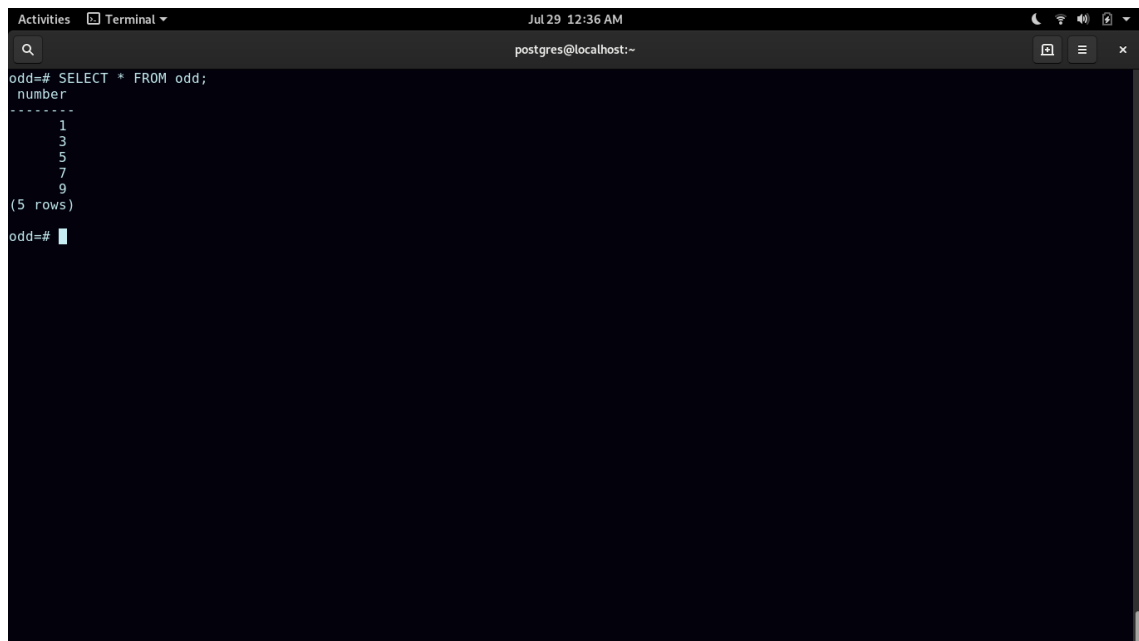
SQL has many aggregate functions that help us in performing many mathematical functions. Some of them are listed below:

1. AVG()
  - (a) The AVG() function returns the average value of a numeric column.
2. MAX()
  - (a) The MAX() function returns the maximum value of a numeric column.
3. MIN()
  - (a) The MIN() function returns the minimum value of a numeric column.
4. COUNT()
  - (a) The COUNT() function returns the number of rows that matches a specified criteria.
5. SUM()
  - (a) The SUM() function returns the total sum of a numeric column.

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.

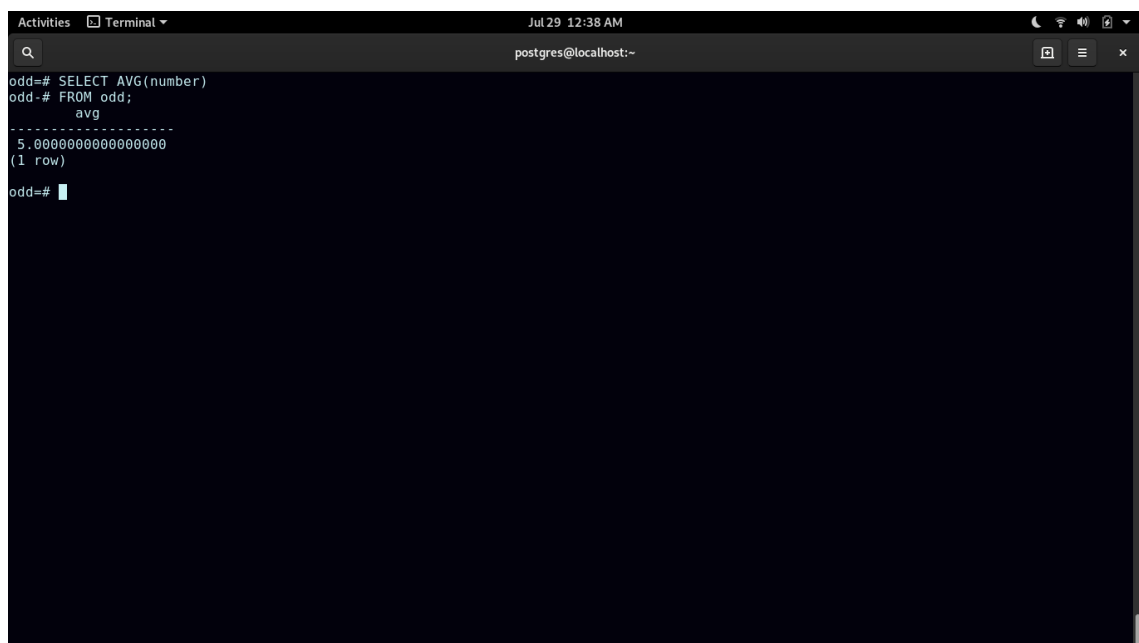
A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The user enters 'odd=# SELECT \* FROM odd;' and the output shows a table with one column 'number' and five rows of odd integers from 1 to 9. The prompt returns to 'odd=#'.

```
odd=# SELECT * FROM odd;
number
-----
 1
 3
 5
 7
 9
(5 rows)
odd=#
```

Figure 1: Entities in the table 'odd'

1. Lets find the average of the given numbers using the AVG() function.  
This can be done using the following query:

```
SELECT AVG(number)
FROM odd;
```

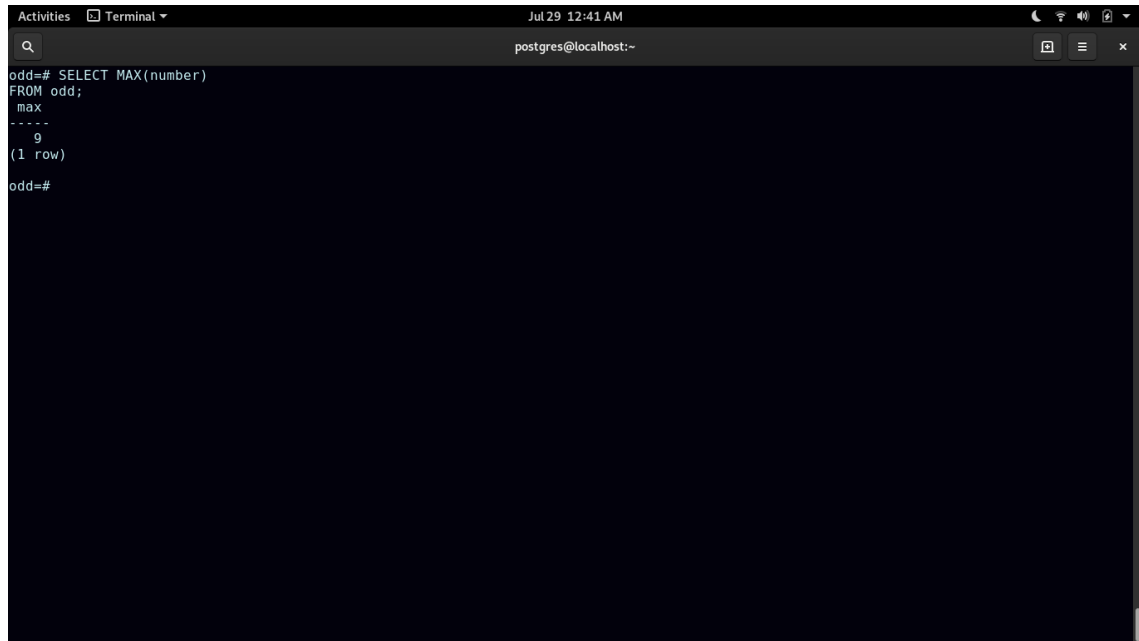
A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The user enters 'odd=# SELECT AVG(number) FROM odd;' and the output shows a single row with the average value 5.0000000000000000. The prompt returns to 'odd=#'.

```
odd=# SELECT AVG(number)
FROM odd;
 avg
-----
5.0000000000000000
(1 row)
odd=#
```

Figure 2: Using AVG() function

2. Now lets find the largest number using MAX() function.

```
SELECT MAX(number)
FROM odd;
```

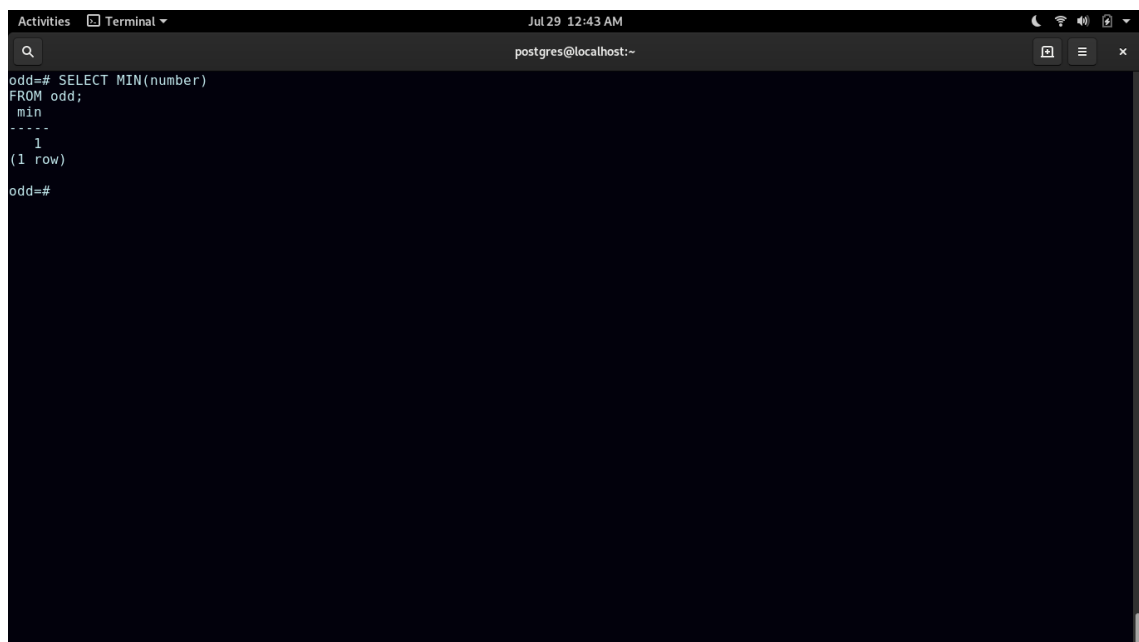


```
Activities Terminal Jul 29 12:41 AM postgres@localhost:~
odd=# SELECT MAX(number)
FROM odd;
max
-----
  9
(1 row)
odd=#
```

Figure 3: Using MAX() function

3. Similarly lets use the MIN() function to find the smallest number.

```
SELECT MIN(number)
FROM odd;
```

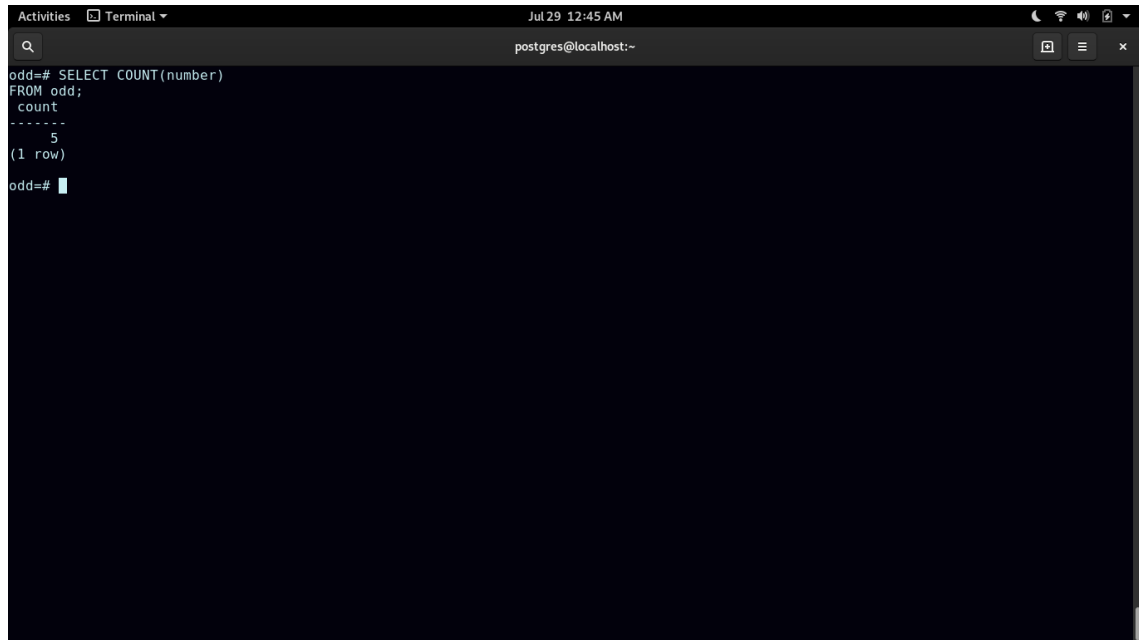


```
Activities Terminal Jul 29 12:43 AM postgres@localhost:~
odd=# SELECT MIN(number)
FROM odd;
min
-----
  1
(1 row)
odd=#
```

Figure 4: Using the MIN() function

4. COUNT() function can be used to count the number of NON NULL values in a column

```
SELECT COUNT(number)
FROM odd;
```

A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The user enters the SQL query 'SELECT COUNT(number) FROM odd;'. The output shows a single row with the value '5' under the column 'count', with '(1 row)' below it. The prompt 'odd=#' is visible at the bottom left.

```
odd=# SELECT COUNT(number)
FROM odd;
 count
-----
      5
(1 row)
odd=#
```

Figure 5: Using the COUNT() function

5. Lets find the sum of the numbers present in the database using the SUM() function.

```
SELECT SUM(number)
FROM odd;
```

```
Activities Terminal Jul 29 12:49 AM postgres@localhost:~
odd=# SELECT SUM(number)
FROM odd;
sum
-----
25
(1 row)
odd=#
```

Figure 6: Using SUM() function

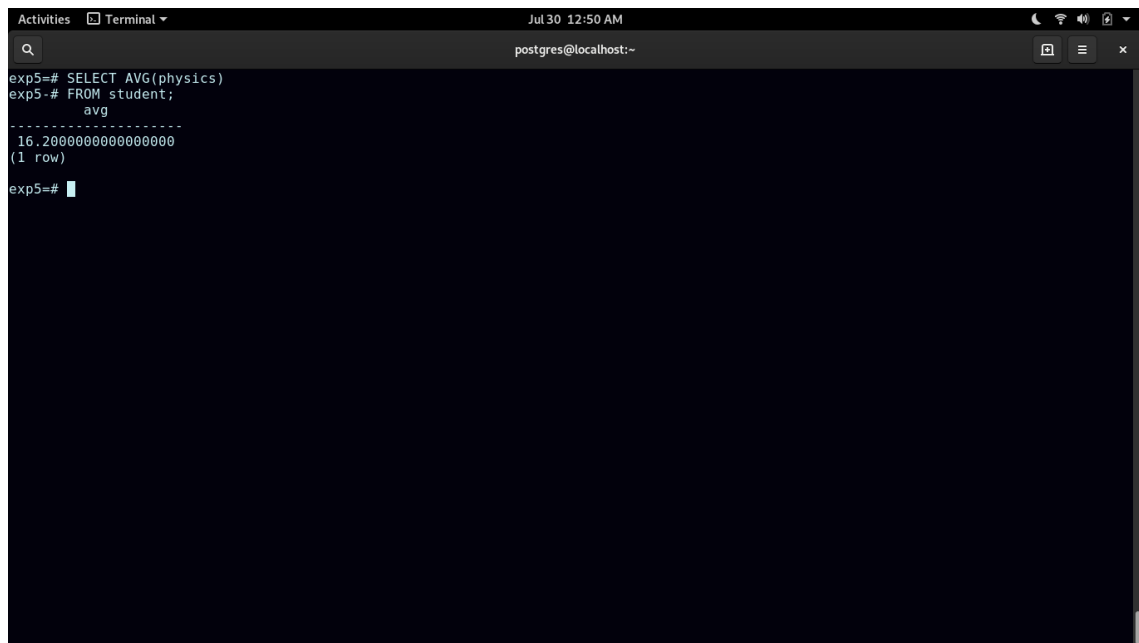
### 3 Questions

```
Activities Terminal Jul 29 11:46 PM postgres@localhost:~
exp4=# SELECT * FROM car_details;
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 | 8.0
5 | Beetle | Volkswagen | Germany | 21.0
6 | Etios | Toyota | Japan | 7.2
7 | Sail | Chevrolet | USA | 5.0
8 | Aria | Tata | India | 7.0
9 | Passat | Volkswagen | Germany | 25.0
10 | SX4 | Maruti | Japan | 6.7
(10 rows)
exp4=#
```

Figure 7: Entries in 'student'

1. Find the class average for the subject 'Physics'

```
SELECT AVG(physics)
FROM student;
```

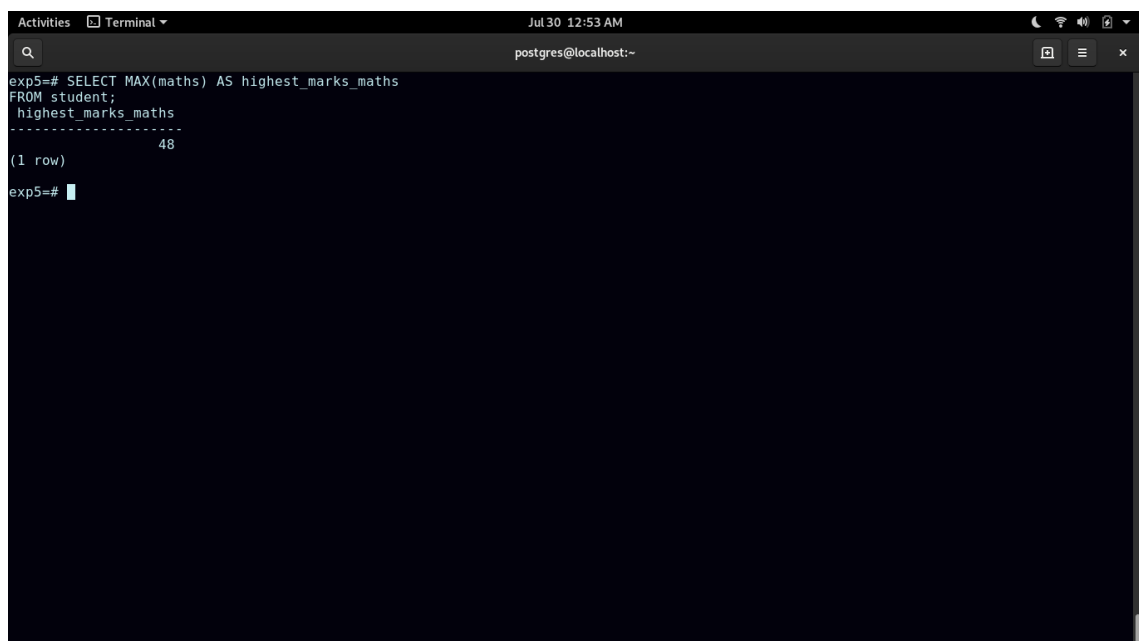
A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The user enters the SQL query: 'exp5=# SELECT AVG(physics) FROM student;'. The output is: '-----', '16.2000000000000000', '(1 row)', and the prompt 'exp5=#' is shown again.

```
exp5=# SELECT AVG(physics)
exp5=# FROM student;
-----
16.2000000000000000
(1 row)
exp5=#
```

Figure 8: Question 1

2. Find the highest marks for mathematics (To be displayed as highest\_marks\_maths).

```
SELECT MAX(maths) AS
highest_marks_maths
FROM student;
```

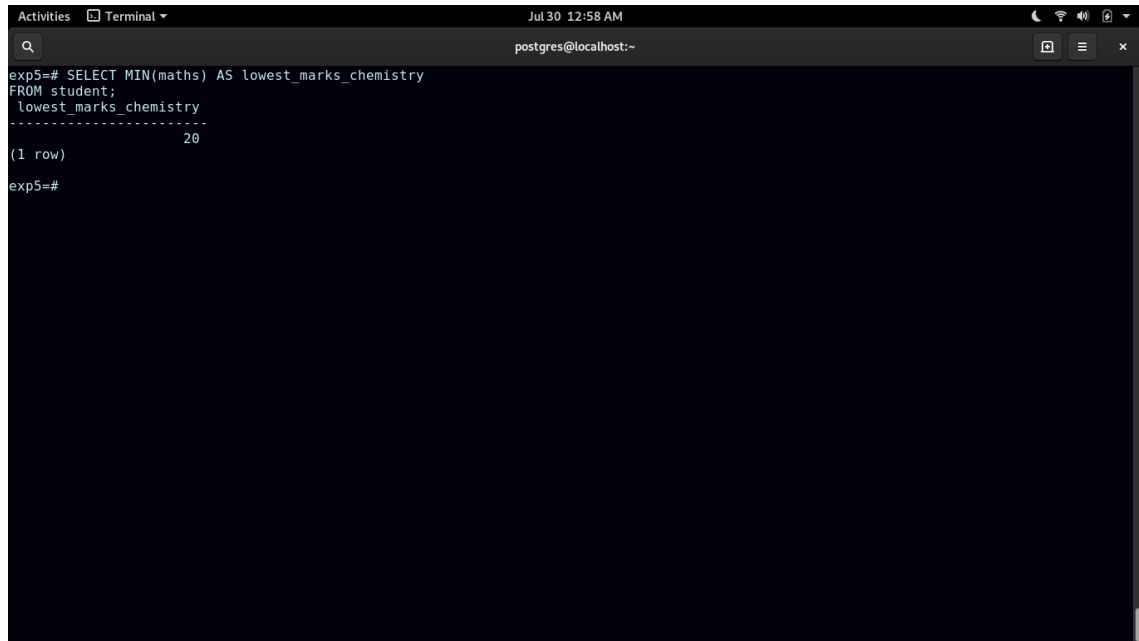
A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The user enters the SQL query: 'exp5=# SELECT MAX(maths) AS highest\_marks\_maths FROM student;'. The output is: '-----', 'highest\_marks\_maths', '48', '(1 row)', and the prompt 'exp5=#' is shown again.

```
exp5=# SELECT MAX(maths) AS highest_marks_maths
exp5=# FROM student;
-----
highest_marks_maths
48
(1 row)
exp5=#
```

Figure 9: Question 2

3. Find the lowest marks for chemistry (To be displayed as lowest\_mark\_chemistry)

```
SELECT MIN(maths) AS  
lowest_marks_chemistry  
FROM student;
```



The screenshot shows a terminal window titled 'Activities Terminal' with a search bar and a close button. The terminal displays the following SQL query and its output:

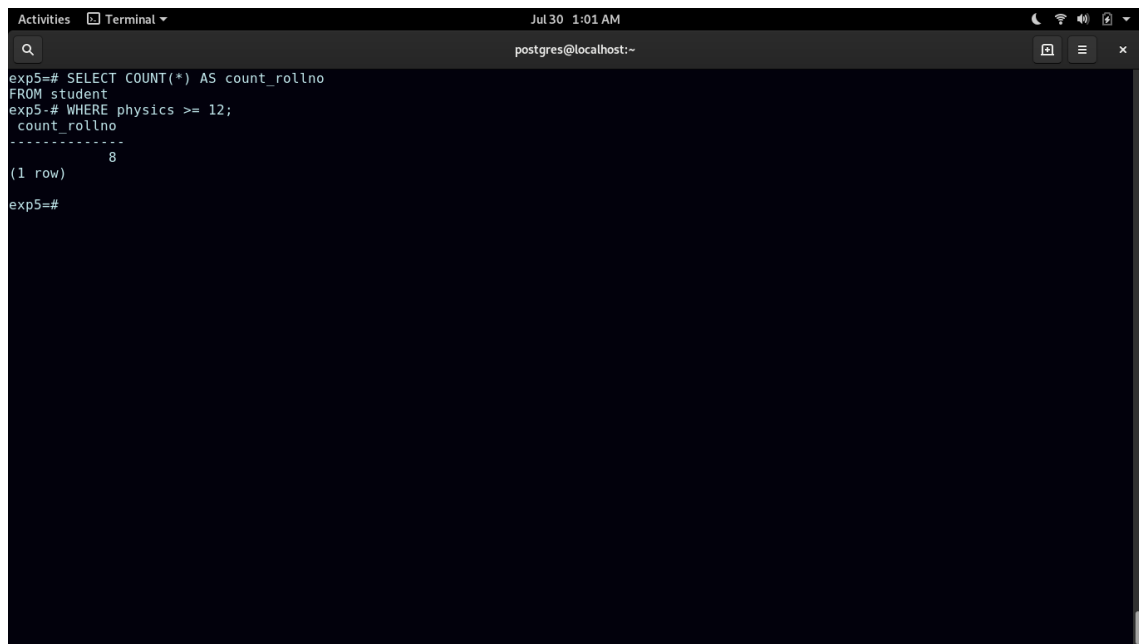
```
exp5=# SELECT MIN(maths) AS lowest_marks_chemistry  
FROM student;  
lowest_marks_chemistry  
-----  
20  
(1 row)  
exp5=#
```

Figure 10: Question 3

4. Find the total number of students who has got a 'pass' in physics.

```
SELECT COUNT(*) AS count_rollno  
FROM student  
WHERE physics >= 12;
```



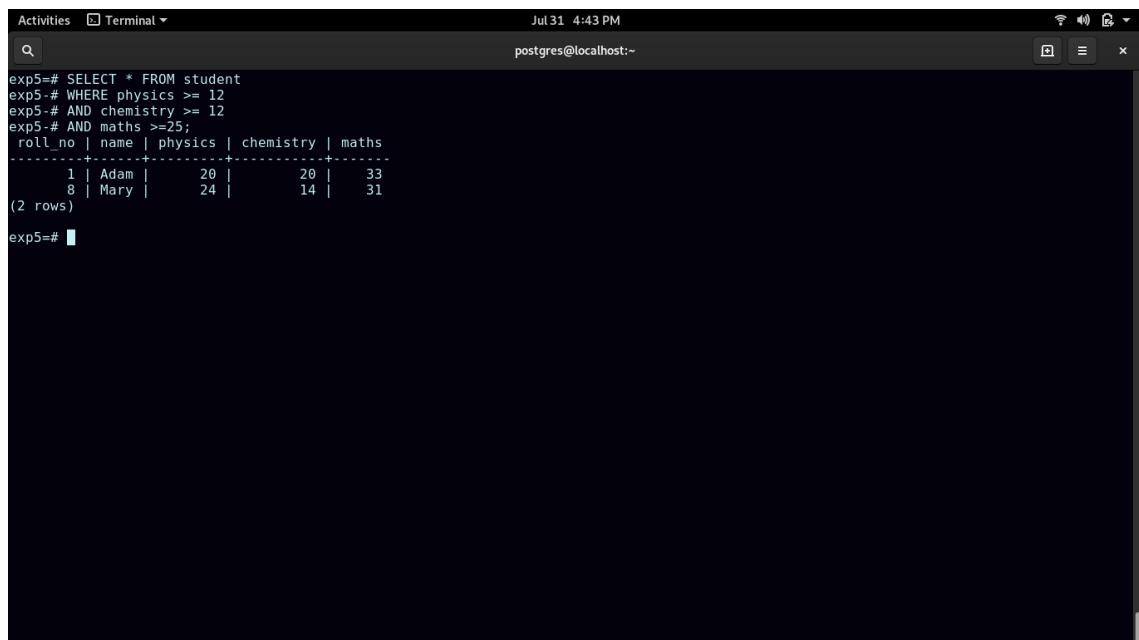


```
Activities Terminal Jul 30 1:01 AM postgres@localhost:~
exp5=# SELECT COUNT(*) AS count_rollno
FROM student
exp5=# WHERE physics >= 12;
count_rollno
-----
8
(1 row)
exp5=#
```

Figure 11: Question 4

5. Generate the list of students who have passed in all the subjects

```
SELECT * FROM student
WHERE physics >= 12
AND chemistry >= 12
AND maths >=25;
```



```
Activities Terminal Jul 31 4:43 PM postgres@localhost:~
exp5=# SELECT * FROM student
exp5=# WHERE physics >= 12
exp5=# AND chemistry >= 12
exp5=# AND maths >=25;
 roll_no | name | physics | chemistry | maths
-----+-----+-----+-----+-----
1 | Adam | 20 | 20 | 33
8 | Mary | 24 | 14 | 31
(2 rows)
exp5=#
```

Figure 12: Question 5

6. Generate a rank list for the class. Indicate Pass/Fail. Ranking based on total marks obtained by the students.

```

SELECT *,
    (physics+chemistry+maths) AS "Total Marks",
CASE
WHEN physics >=12 AND chemistry >=12 AND maths >=25
THEN 'P'
ELSE 'F'
END AS result
FROM student;

```

The screenshot shows a terminal window titled "Terminal" with the date and time "Jul 31 5:03 PM". The user is logged in as "postgres@localhost". The terminal displays the execution of an SQL query labeled "exp5=#". The query is a SELECT statement that calculates the total marks for each student and determines if they passed based on specific criteria. The results are displayed in a table format with 7 columns: roll\_no, name, physics, chemistry, maths, Total Marks, and result. There are 10 rows of data, corresponding to the 10 students in the 'student' table. The results show that 3 students (Adam, Mary, and Zack) passed, while 7 students (Bob, Bright, Duke, Elvin, Fetcher, Georgina, and Tom) failed.

```

exp5=# SELECT *,
    (physics+chemistry+maths) AS "Total Marks",
CASE
WHEN physics >=12 AND chemistry >=12 AND maths >=25
THEN 'P'
ELSE 'F'
END AS result
FROM student;

```

roll_no	name	physics	chemistry	maths	Total Marks	result
1	Adam	20	20	33	73	P
2	Bob	18	9	41	68	F
3	Bright	22	7	31	60	F
4	Duke	13	21	20	54	F
5	Elvin	14	22	23	59	F
6	Fetcher	2	10	48	60	F
7	Georgina	22	12	22	56	F
8	Mary	24	14	31	69	P
9	Tom	19	15	24	58	F
10	Zack	8	20	36	64	F

(10 rows)

```

exp5=#

```

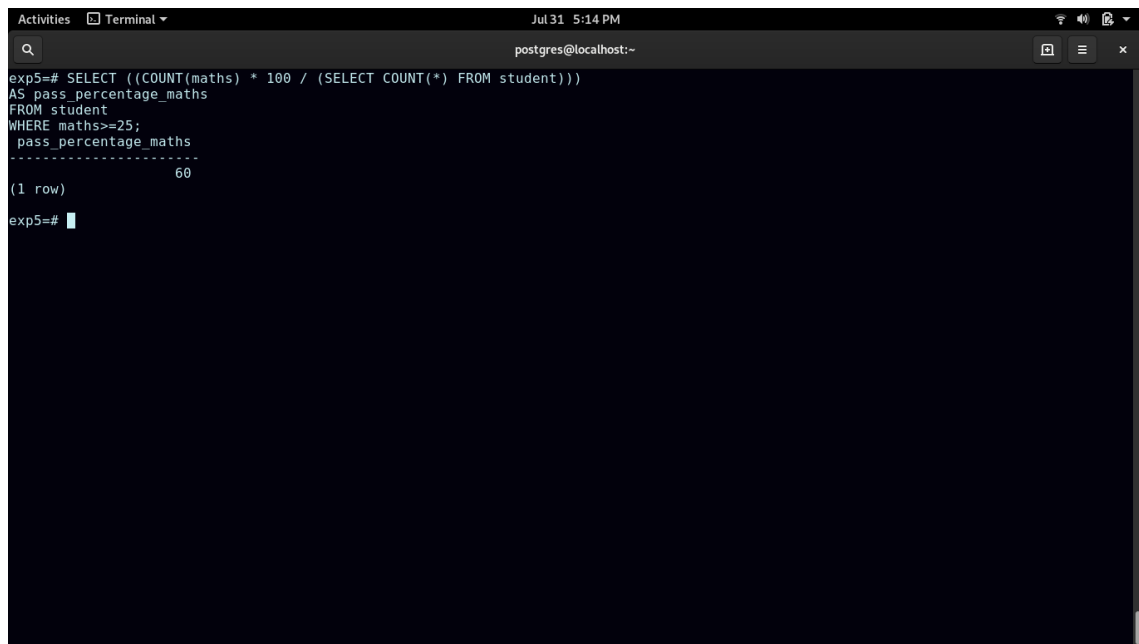
Figure 13: Question 6

7. Find pass percentage of the class for mathematics.

```

SELECT ((COUNT(maths) * 100 /
    (SELECT COUNT(*) FROM student)))
AS pass_percentage_maths
FROM student
WHERE maths >=25;

```

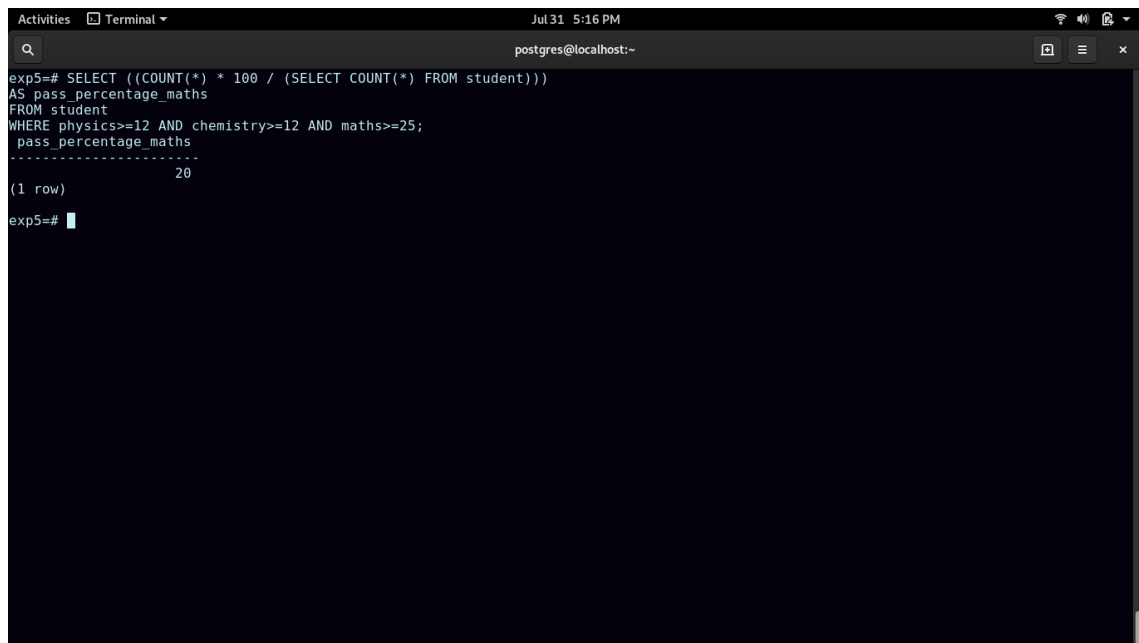
A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The user enters a SQL query: 'exp5=# SELECT ((COUNT(maths) \* 100 / (SELECT COUNT(\*) FROM student))) AS pass\_percentage\_maths FROM student WHERE maths >= 25;'. The output shows a single row with the value '60'. The prompt returns to 'exp5=#'.

```
exp5=# SELECT ((COUNT(maths) * 100 / (SELECT COUNT(*) FROM student)))
AS pass_percentage_maths
FROM student
WHERE maths >= 25;
pass_percentage_maths
-----
60
(1 row)
exp5=#
```

Figure 14: Question 7

8. Find the overall pass percentage for all class.

```
SELECT ((COUNT(*) * 100 /
(SELECT COUNT(*) FROM student)))
AS pass_percentage
FROM student
WHERE physics >= 12 AND
chemistry >= 12 AND maths >= 25;
```

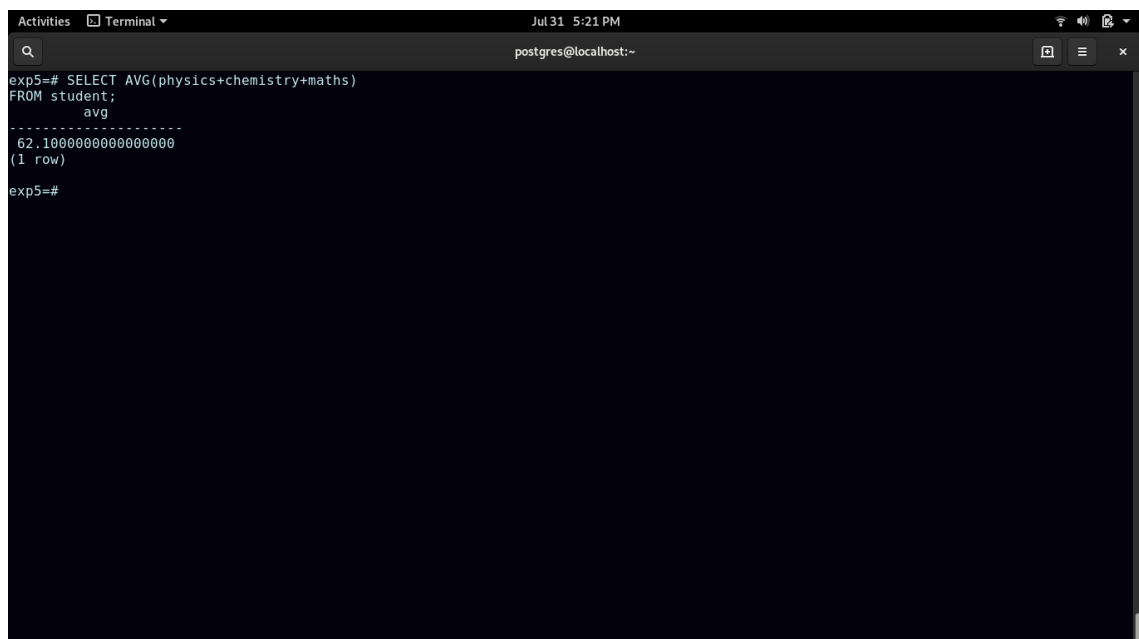
A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The SQL query entered is: 'exp5=# SELECT ((COUNT(\*) \* 100 / (SELECT COUNT(\*) FROM student))) AS pass\_percentage\_maths FROM student WHERE physics>=12 AND chemistry>=12 AND maths>=25;'. The output shows a single row with the value '20'.

```
exp5=# SELECT ((COUNT(*) * 100 / (SELECT COUNT(*) FROM student)))
AS pass_percentage_maths
FROM student
WHERE physics>=12 AND chemistry>=12 AND maths>=25;
-----
pass_percentage_maths
-----
(1 row)
20
exp5=#
```

Figure 15: Question 8

9. Find the class average.

```
SELECT AVG(physics+chemistry+maths)
FROM student;
```

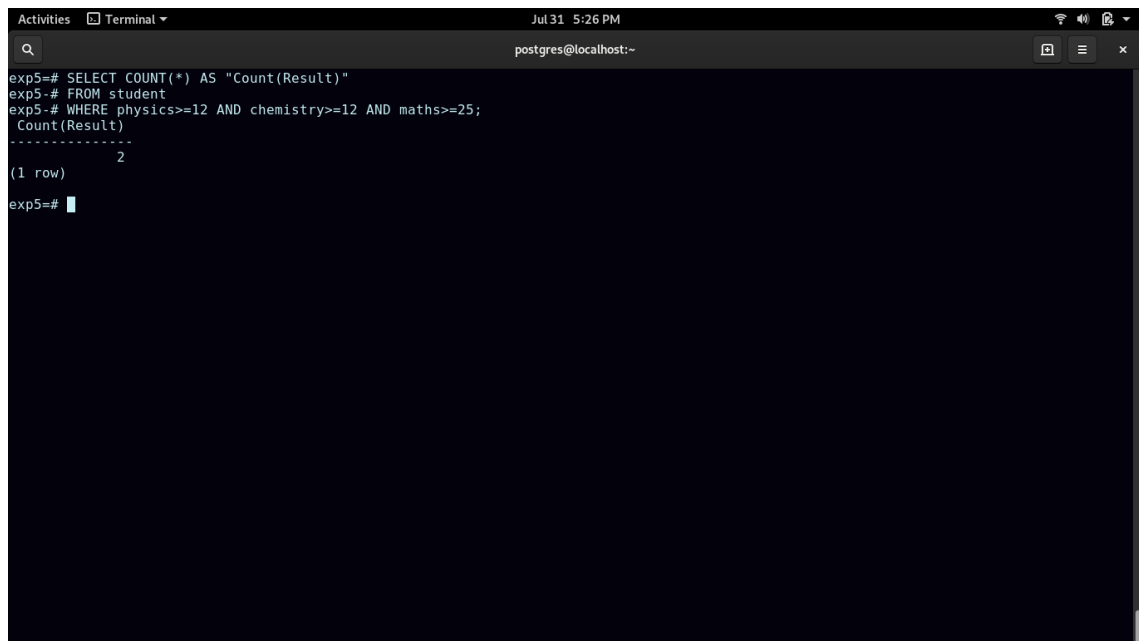
A terminal window titled 'Terminal' with a search bar and window controls. The prompt is 'postgres@localhost:~'. The SQL query entered is: 'exp5=# SELECT AVG(physics+chemistry+maths) FROM student;'. The output shows a single row with the value '62.100000000000000000000000000000'.

```
exp5=# SELECT AVG(physics+chemistry+maths)
FROM student;
-----
avg
-----
62.100000000000000000000000000000
(1 row)
exp5=#
```

Figure 16: Question 9

10. Find the total number of students who have got a Pass.

```
SELECT COUNT(*) AS "Count(Result)"
FROM student
WHERE physics >=12 AND
chemistry >=12 AND maths >=25;
```

A terminal window titled 'Terminal' with a search bar and window controls. The terminal shows a PostgreSQL session where a SQL query is executed. The query counts the number of rows in the 'student' table where physics is greater than or equal to 12, chemistry is greater than or equal to 12, and maths is greater than or equal to 25. The result is a single row with the value 2.

```
exp5=# SELECT COUNT(*) AS "Count(Result)"
exp5=# FROM student
exp5=# WHERE physics >=12 AND chemistry >=12 AND maths >=25;
Count(Result)
-----
2
(1 row)
exp5=#
```

Figure 17: Question 10

## 4 Result

- Successfully used the aggregate functions AVG(), MAX(), MIN(), COUNT(), SUM() in PostgreSQL.