

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

28th March 2022

1. GATE 2010

Creation:

- mysql> use 20bcs021_faizan;
- mysql> CREATE TABLE Passenger
 - > (
 - > pid INT,
 - > pname VARCHAR(25),
 - > age INT
 - >);
- mysql> INSERT INTO Passenger VALUES
 - > (0, 'Sachin', 65),
 - > (1, 'Rahul', 66),
 - > (2, 'Sourav', 67),
 - > (3, 'Anil', 69);
- mysql> SELECT * FROM Passenger;

```
mysql> SELECT * FROM Passenger;
+-----+-----+-----+
| pid | pname | age |
+-----+-----+-----+
| 0 | Sachin | 65 |
| 1 | Rahul | 66 |
| 2 | Sourav | 67 |
| 3 | Anil | 69 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

- mysql> CREATE TABLE Reservation
-> (
-> pid INT,
-> class VARCHAR(3),
-> tid INT
->);
- mysql> INSERT INTO Reservation VALUES
-> (0, 'AC', 8200),
-> (1, 'AC', 8201),
-> (2, 'SC', 8201),
-> (5, 'AC', 8203),
-> (1, 'SC', 8204),
-> (3, 'AC', 8202);
- mysql> SELECT * FROM Reservation;

```
mysql> SELECT * FROM Reservation;
+-----+-----+-----+
| pid | class | tid |
+-----+-----+-----+
| 0 | AC | 8200 |
| 1 | AC | 8201 |
| 2 | SC | 8201 |
| 5 | AC | 8203 |
| 1 | SC | 8204 |
| 3 | AC | 8202 |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

Query:

- What pids are returned by the following SQL query for the above instance of the tables?

```
SELECT pid
FROM Reservation
WHERE class 'AC' AND
      EXISTS (SELECT *
              FROM Passenger
              WHERE age > 65 AND
                    Passenger.pid = Reservation.pid);
```

- (a) 1, 0 (b) 1, 2 (c) 1, 5 (d) 1, 3

OUTPUT:

```
+-----+
| pid |
+-----+
|    1 |
|    3 |
+-----+
2 rows in set (0.00 sec)
```

ANSWER: Option (d) 1, 3

2.GATE 2010

Creation:

- mysql> CREATE TABLE Suppliers
-> (
-> sid INT,
-> sname VARCHAR(25),
-> city VARCHAR(25),
-> street VARCHAR(25)
->);
- mysql> INSERT INTO Suppliers VALUES
-> (3, 'Dinesh', 'Mumbai', 'Lal Chowk'),
-> (1, 'Yash', 'Mumbai', 'Greek Rd'),
-> (5, 'Ayush', 'New Delhi', 'Chawri Bazaar'),
-> (4, 'Abdullah', 'Bhopal', 'Jalal Rd'),
-> (2, 'Tarvinder', 'Punjab', 'Bathinda Rd');

- mysql> SELECT * FROM Suppliers;

```
mysql> SELECT * FROM Suppliers;
+-----+-----+-----+-----+
| sid | sname   | city    | street   |
+-----+-----+-----+-----+
| 3   | Dinesh  | Mumbai  | Lal Chowk |
| 1   | Yash    | Mumbai  | Greek Rd  |
| 5   | Ayush   | New Delhi | Chawri Bazaar |
| 4   | Abdullah | Bhopal   | Jalal Rd  |
| 2   | Tarvinder | Punjab  | Bathinda Rd |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

- mysql> CREATE TABLE Parts
-> (
-> pid INT,
-> pname VARCHAR(10),

- ```

-> color VARCHAR(10)
->);
mysql> INSERT INTO Parts VALUES
-> (8, 'Pulley', 'Black'),
-> (3, 'Hand Rail', 'Blue'),
-> (2, 'Shaper', 'Blue'),
-> (1, 'Bolts', 'White'),
-> (6, 'Wheel Rail', 'Red'),
-> (4, 'Blower', 'Blue');

```
- ```

mysql> SELECT * FROM Parts;

```

```

mysql> SELECT * FROM Parts;
+-----+-----+-----+
| pid | pname    | color |
+-----+-----+-----+
| 8   | Pulley   | Black |
| 3   | Hand Rail | Blue  |
| 2   | Shaper   | Blue  |
| 1   | Bolts    | White |
| 6   | Wheel Rail | Red   |
| 4   | Blower   | Blue  |
+-----+-----+-----+
6 rows in set (0.00 sec)

```

- ```

mysql> CREATE TABLE Catalogue
-> (
-> sid INT,
-> pid INT,
-> cost DECIMAL(10,2)
->);

```
- ```

mysql> INSERT INTO Catalogue VALUES
-> (5, 3, 1200.00),
-> (3, 2, 2000.00),
-> (2, 4, 1000.00),
-> (1, 6, 1500.00),
-> (4, 1, 500.00),
-> (4, 8, 800.00);

```
- ```

mysql> SELECT * FROM Catalogue;

```

```

mysql> SELECT * FROM Catalogue;
+-----+-----+-----+
| sid | pid | cost |
+-----+-----+-----+
5	3	1200.00
3	2	2000.00
2	4	1000.00
1	6	1500.00
4	1	500.00
4	8	800.00
+-----+-----+-----+
6 rows in set (0.00 sec)

```

## Query:

1. Consider the following relational schema:

```
Suppliers (sid:integer, sname:string, city:string, street:string)
Parts (pid:integer, pname:string, color:string) Catalog(sid:integer,
pid:integer, cost:real)
```

**Consider the following relational query on the above database:**

```
SELECT S.sname
FROM Suppliers S
WHERE S.sid NOT IN (SELECT C.sid FROM Catalog C
WHERE C.pid NOT IN (SELECT P.pid FROM Parts P
WHERE P.color<> 'blue'));
```

**Assume that relations corresponding to the above schema are not empty. Which one of the following is the correct interpretation of the above query?**

- (a) Find the names of all suppliers who have not supplied a non-blue part.
- (b) Find the names of all suppliers who have supplied only blue parts.
- (c) Find the names of all suppliers who have not supplied only blue parts.
- (d) Find the names of all suppliers who have supplied a non-blue part

**OUTPUT:**

```
+-----+
| sname |
+-----+
| Yash |
| Abdullah |
+-----+
2 rows in set (0.01 sec)
```

**ANSWER:** Option (c) Find the names of all suppliers who have not supplied only blue parts.

**EXPLANATION:** The innermost subquery returns the pids of the non-blue parts, and since the middle subquery uses NOT IN, therefore this query searches for pids having blue parts and returns the sids of the suppliers. Now since the outermost query also uses a NOT IN statement, it returns the names of suppliers who have NOT supplied only the blue parts. Here, Yash only supplies a Red part and Abdullah supplies both a Blue part and a Black part, so both of them have not supplied only blue parts.

### 3. GATE 2004

## Creation:

- ```
mysql> CREATE TABLE Employee3
-> (
-> name VARCHAR(25),
-> sex CHAR,
-> salary DECIMAL(10,2),
-> deptName VARCHAR(20)
-> );
```
- ```
mysql> INSERT INTO Employee3 VALUES
-> ('Amit', 'M', 30000.00, 'Management'),
-> ('Rita', 'F', 60000.00, 'Headquarters'),
-> ('Jitendra', 'M', 80000.00, 'Headquarters'),
-> ('Riya', 'F', 40000.00, 'Research'),
-> ('Ravish', 'M', 20000.00, 'Outreach'),
-> ('Prachi', 'F', 25000.00, 'Management'),
-> ('Sita', 'F', 65000.00, 'Research'),
```

```
-> ('Utkarsh', 'M', 30000.00, 'Research');
```

- mysql> SELECT \* FROM Employee3;

```
mysql> SELECT * FROM Employee3;
```

| name     | sex | salary   | deptName     |
|----------|-----|----------|--------------|
| Amit     | M   | 30000.00 | Management   |
| Rita     | F   | 60000.00 | Headquarters |
| Jitendra | M   | 80000.00 | Headquarters |
| Riya     | F   | 40000.00 | Research     |
| Ravish   | M   | 20000.00 | Outreach     |
| Prachi   | F   | 25000.00 | Management   |
| Sita     | F   | 65000.00 | Research     |
| Utkarsh  | M   | 30000.00 | Research     |

8 rows in set (0.00 sec)

## Query:

### 1. The employee information in a company is stored in the relation

Employee (name, sex, salary, deptName)

**Consider the following SQL query**

```
Select deptName
```

```
From Employee
```

```
Where sex = 'M'
```

```
Group by deptName
```

```
Having avg(salary) >
```

```
(select avg (salary) from Employee);
```

**It returns the names of the department in which**

- (a) the average salary of male employees is more than the average salary in the company
- (b) the average salary is more than the average salary in the company
- (c) the average salary of male employees is more than the average salary of all male employees in the company
- (d) the average salary of male employees is more than the average salary of employees in the same department.

**OUTPUT:**

```
+-----+
| deptName |
+-----+
| Headquarters |
+-----+
1 row in set (0.01 sec)
```

**ANSWER:** Option (a) the average salary of male employees is more than the average salary in the company

**EXPLANATION:** The subquery has no conditional and therefore returns the average salary of ALL employees in the company, be it male or female. The outer query compares this avg sal by the avg sal of male employees and returns those who have a greater avg sal than those in the company. Here in the table, we can see that the avg(salary) for ALL employees is 43750, while for Headquarters the average salary for male employees is 80000, therefore the result of the query is 'Headquarters'.

## 4.GATE 2005

### Creation:

- mysql> CREATE TABLE book  
-> (  
-> title VARCHAR(100),  
-> price DECIMAL(8,2)  
-> );
- mysql> INSERT INTO book VALUES  
-> ('The Invisible Life of Addie Larue', 750.00),  
-> ('Maybe You should Talk to Someone', 550.00),  
-> ('Verity', 600.00),  
-> ('The Giver Of Stars', 700.00),  
-> ('The Love Hypothesis', 800.00),  
-> ('Malibu Rising', 525.00),  
-> ('People we Meet on Vacation', 625.00),  
-> ('The Defining Decade', 650.00);
- mysql> SELECT \* FROM book;

```
mysql> SELECT * FROM book;
+-----+-----+
| title | price |
+-----+-----+
The Invisible Life of Addie Larue	750.00
Maybe You should Talk to Someone	550.00
Verity	600.00
The Giver Of Stars	700.00
The Love Hypothesis	800.00
Malibu Rising	525.00
People we Meet on Vacation	625.00
The Defining Decade	650.00
+-----+-----+
8 rows in set (0.00 sec)
```

### Query:

1. The relation book (title, price) contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list?

```
Select title
From book as B
Where (Select count(*)
 from book as T
 Where T.price > B.price) < 5;
```

(a) Titles of the four most expensive books

- (b) Title of the fifth most inexpensive book
- (c) Title of the fifth most expensive book
- (d) Titles of the five most expensive books

**OUTPUT:**

```
+-----+
| title |
+-----+
| The Invisible Life of Addie Larue |
| The Giver Of Stars |
| The Love Hypothesis |
| People we Meet on Vacation |
| The Defining Decade |
+-----+
5 rows in set (0.00 sec)
```

**ANSWER:** Option (d) Titles of the five most expensive books

**EXPLANATION:** The inner subquery returns the count of books which have a greater price (expensive) than a specific book taken up from the outer query. The range of values for this subquery is 0,1,2,... and so on. So, for the comparison ( $< 5$ ), it has the following options 0,1,2,3,4 which adds up to a total of five books. Therefore this query returns the titles of the five most expensive books.

## 5.GATE 2006

### Creation:

- ```
mysql> CREATE TABLE enrolled
    -> (
    -> student VARCHAR(30) NOT NULL,
    -> course VARCHAR(15) NOT NULL,
    -> PRIMARY KEY (student, course)
    -> );
```
- ```
mysql> INSERT INTO enrolled VALUES
 -> ('Arjun', 'Frontend'),
 -> ('Divaker', 'WebD'),
 -> ('Fatima', 'App Dev'),
 -> ('Yash', 'AI'),
 -> ('Pallavi', 'IoT'),
 -> ('Pallavi', 'Linux'),
 -> ('Arjun', 'Game Dev');
```
- ```
mysql> SELECT * FROM enrolled;
```



```
mysql> SELECT * FROM enrolled;
+-----+-----+
| student | course |
+-----+-----+
| Arjun   | Frontend |
| Arjun   | Game Dev |
| Divaker | WebD     |
| Fatima  | App Dev  |
| Pallavi | IoT      |
| Pallavi | Linux    |
| Yash    | AI       |
+-----+-----+
7 rows in set (0.00 sec)
```

- mysql> CREATE TABLE paid
 > (
 > student VARCHAR(30) NOT NULL,
 > amount DECIMAL(8,2) NOT NULL,
 > PRIMARY KEY (student)
 >);
- mysql> INSERT INTO paid VALUES
 > ('Arjun', 1000.00),
 > ('Fatima', 800.00),
 > ('Divaker', 450.00),
 > ('Yash', 650.00),
 > ('Pallavi', 600.00);
- mysql> SELECT * FROM paid;

```
mysql> SELECT * FROM paid;
+-----+-----+
| student | amount |
+-----+-----+
| Arjun   | 1000.00 |
| Divaker | 450.00  |
| Fatima  | 800.00  |
| Pallavi | 600.00  |
| Yash    | 650.00  |
+-----+-----+
5 rows in set (0.00 sec)
```

Query:

1. Consider the relation enrolled (student, course) in which (student, course) is the primary key, and the relation paid (student, amount) where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Given the following four queries:

Query1: select student from enrolled where student in (select student from paid)

OUTPUT:

```
+-----+
| student |
+-----+
| Arjun   |
| Arjun   |
| Divaker |
| Fatima  |
| Pallavi |
| Pallavi |
| Yash    |
+-----+
```

Query2: `select student from paid where student in (select student from enrolled)`

OUTPUT:

student
Arjun
Divaker
Fatima
Pallavi
Yash

Query3: `select E.student from enrolled E, paid P where E.student = P.student`

OUTPUT:

student
Arjun
Arjun
Divaker
Fatima
Pallavi
Pallavi
Yash

Query4: `select student from paid where exists (select * from enrolled where enrolled.student = paid.student)`

OUTPUT:

student
Arjun
Divaker
Fatima
Pallavi
Yash

Which one of the following statements is correct?

- (a) All queries return identical row sets for any database
- (b) Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.
- (c) There exist databases for which Query3 returns strictly fewer rows than Query2.
- (d) There exist databases for which Query4 will encounter an integrity violation at runtime.

ANSWER: Option (b) Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.

EXPLANATION: Q1 and Q3 return the same row sets while Q2 and Q4 return the exact same row sets. Therefore, Option (a) is ruled out. Q3 returns a greater number of rows than Q2, since the primary key in enrolled table is a unique pair of student and course, which reduces down the number of matches. Option (c) is ruled out in this case. Q4 works satisfactorily for all databases. Therefore the answer is option (b).

6.GATE 2006

Creation:

- mysql> CREATE TABLE account
-> (
-> customer VARCHAR(25) NOT NULL,
-> balance DECIMAL(10,2) NOT NULL,
-> PRIMARY KEY (customer)
->);
- mysql> INSERT INTO account VALUES
-> ('Younus', 160000.00),
-> ('Hadiya', 120000.00),
-> ('Chandler', 170500.00),
-> ('Joey', 340000.00),
-> ('Phoebe', 300000.00),
-> ('Rachel', 200000.00);
- mysql> SELECT * FROM account;

```
mysql> SELECT * FROM account;
+-----+-----+
| customer | balance |
+-----+-----+
| Chandler | 170500.00 |
| Hadiya   | 120000.00 |
| Joey     | 340000.00 |
| Phoebe   | 300000.00 |
| Rachel   | 200000.00 |
| Younus   | 160000.00 |
+-----+-----+
6 rows in set (0.00 sec)
```

Query:

2. Consider the relation `account (customer, balance)` where `customer` is a primary key and there are no null values. We would like to rank customers according to decreasing balance. The customer with the largest balance gets rank 1. Ties are not broke but ranks are skipped: if exactly two customers have the largest balance they each get rank 1 and rank 2 is not assigned

Query1: select A.customer, count(B.customer)
from account A, account B
where A.balance <=B.balance
group by A.customer

OUTPUT:

customer	count(B.customer)
Younus	5
Hadiya	6
Chandler	4
Rachel	3
Phoebe	2
Joey	1

Query2: select A.customer, 1+count(B.customer)
 from account A, account B
 where A.balance < B.balance
 group by A.customer

OUTPUT:

customer	1+count(B.customer)
Younus	5
Hadiya	6
Rachel	3
Phoebe	2
Chandler	4

Consider these statements about Query1 and Query2.

1. Query1 will produce the same row set as Query2 for some but not all databases.
2. Both Query1 and Query2 are correct implementation of the specification
3. Query1 is a correct implementation of the specification but Query2 is not
4. Neither Query1 nor Query2 is a correct implementation of the specification
5. Assigning rank with a pure relational query takes less time than scanning in decreasing balance order assigning ranks using ODBC.

Which two of the above statements are correct?

- (a) 2 and 5
- (b) 1 and 3
- (c) 1 and 4
- (d) 3 and 5

ANSWER: Option (c) 1 and 4

EXPLANATION: Q1 returns the ranks as according to number of duplicate values of balance, for 'k' customers holding the same balance, the rank given is 'k' and not rank 1,2,... etc. Q2 can never return a rank 1 unless there's an empty set (count(B.customer) = 0 implies 1+count(B.customer) = 1). Statement 4 is therefore correct, so the correct option is option (c). Statement 1 is true for empty set returned by both queries.

7.GATE 2011

Creation:

- mysql> use 20bcs021_faizan;
- mysql> CREATE TABLE Loan_Records
-> (
-> borrower varchar(15),
-> bank_manager varchar(15),
-> loan_amt decimal(10,2)
->);
- mysql> INSERT INTO Loan_Records values
-> ('Ramesh', 'Sunderajan', 10000.00),
-> ('Suresh', 'Ramgopal', 15000.00),
-> ('Mahesh', 'Sunderajan', 7000.00);
- mysql> SELECT * FROM Loan_Records;

```
mysql> SELECT * FROM Loan_Records;
+-----+-----+-----+
| borrower | bank_manager | loan_amt |
+-----+-----+-----+
| Ramesh   | Sunderajan   | 10000.00 |
| Suresh   | Ramgopal     | 15000.00 |
| Mahesh   | Sunderajan   | 7000.00  |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

Query:

1. What is the output of the following SQL query?

```
SELECT Count(*)
FROM ((SELECT Borrower, Bank_Manager
FROM Loan_Records) AS S
NATURAL JOIN (SELECT Bank_Manager, Loan_Amount
FROM Loan_Records) AS T);
```

- (a) 3 (b) 5 (c) 9 (d) 6

OUTPUT:

```
+-----+
| COUNT(*) |
+-----+
|         5 |
+-----+
1 row in set (0.01 sec)
```

ANSWER: Option (b) 5

EXPLANATION: As a result of natural join from first subquery, two entries of bank_manager as 'Sunderajan' are created in the temporary table S, which is joined to temporary table T by a common attribute values of bank_manager. Since there are two duplicate 'Sunderajan', there will be four entries in the result table after joining, each one from S matched on to each one from T (2x2), and one entry as a common entry ('Ramgopal').

8.GATE 2007

Creation:

- mysql> create table employee4 as select * from employee2;
//since this table is similar to table employee2 with slight modifications
- mysql> SELECT * FROM employee4;

```
mysql> SELECT * FROM employee4;
+-----+-----+-----+-----+
| empId | name  | salary | department |
+-----+-----+-----+-----+
| 101   | Amit  | 25000  | 1          |
| 102   | Sunil | 20000  | 1          |
| 103   | Rakesh | 18000  | 5          |
| 104   | Ajay  | 16000  | 5          |
| 105   | Suhail | 20000  | 1          |
| 106   | Arif  | 18000  | 1          |
| 107   | Suresh | 24000  | 5          |
| 108   | Vijay | 22000  | 5          |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

Query:

1. Consider the table employee(empId, name, department, salary) and the two queries Q1, Q2 below. Assuming that department 5 has more than one employee, and we want to find the employees who get higher salary than anyone in the department 5, which one of the statements is TRUE for any arbitrary employee table?

Q1: Select e.empId
From employee e
Where not exists
(Select * From employee s where s.department = "5" and s.salary
>=e.salary)

OUTPUT:

```
+-----+
| empId |
+-----+
| 101   |
+-----+
```

Query2: Select e.empId
From employee e
Where e.salary > Any
(Select distinct salary From employee s Where s.department = "5")

OUTPUT:

empId
101
102
103
105
106
107
108

- (a) Q1 is the correct query
- (b) Q2 is the correct query
- (c) Both Q1 and Q2 produce the same answer.
- (d) Neither Q1 nor Q2 is the correct query.

ANSWER: Option (a) Q1 is the correct query

EXPLANATION: ANY keyword matches for any conditional returned by the subquery, while EXISTS ensures for only one match. Therefore, Q1 compares the query to all results from the subquery and returns the correct answer of the employee having higher salary than ANYONE ELSE in department 5. Here, Q1 returns the empId as 101, which is the highest paid employee, but Q2 returns all paid employees except the lowest paid.

9.GATE 2000

Creation:

- ```
mysql> CREATE TABLE r
 -> (
 -> w INT,
 -> X INT
 ->);
```
- ```
mysql> INSERT INTO r VALUES
    -> (4, 8),
    -> (7, 5),
    -> (3, 6),
    -> (2, 1),
    -> (0, 4),
    -> (5, 3);
```
- ```
mysql> SELECT * FROM r;
```

```
mysql> SELECT * FROM r;
```

| w | x |
|---|---|
| 4 | 8 |
| 7 | 5 |
| 3 | 6 |
| 2 | 1 |
| 0 | 4 |
| 5 | 3 |

```
6 rows in set (0.00 sec)
```

- mysql> CREATE TABLE s  
-> (  
-> y INT,  
-> z INT  
-> );
- mysql> INSERT INTO s VALUES  
-> (5, 2),  
-> (9, 0),  
-> (4, 1);
- mysql> SELECT \* FROM s;

```
mysql> SELECT * FROM s;
```

| y | z |
|---|---|
| 5 | 2 |
| 9 | 0 |
| 4 | 1 |

```
3 rows in set (0.00 sec)
```

## Query:

1. Given relations  $r(w, x)$  and  $s(y, z)$  the result of  
select distinct w, x  
From r, s

OUTPUT:

| w | x |
|---|---|
| 4 | 8 |
| 7 | 5 |
| 3 | 6 |
| 2 | 1 |
| 0 | 4 |
| 5 | 3 |

(which is the same as the output for 'r')

is guaranteed to be same as r, provided

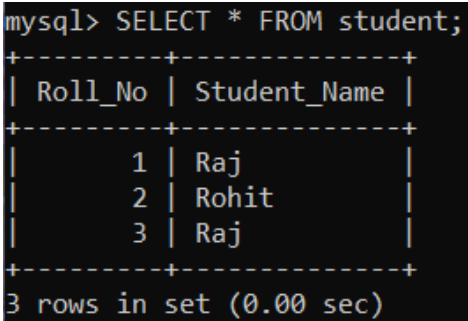
- (a) r has no duplicates and s is non-empty
- (b) r and s have no duplicates
- (c) s has no duplicates and r is non-empty
- (d) r and s have the same number of tuples

**ANSWER:** Option (a) r has no duplicates and s is non-empty

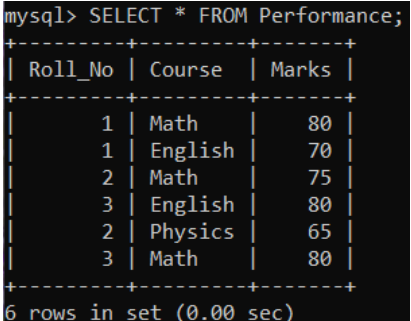


## 10. GATE 2015

### Creation:

- mysql> CREATE TABLE student  
-> (  
-> Roll\_No INT,  
-> Student\_Name VARCHAR(25)  
-> );
- mysql> INSERT INTO student VALUES  
-> (1, 'Raj'),  
-> (2, 'Rohit'),  
-> (3, 'Raj');
- mysql> SELECT \* FROM student;  


| Roll_No | Student_Name |
|---------|--------------|
| 1       | Raj          |
| 2       | Rohit        |
| 3       | Raj          |

3 rows in set (0.00 sec)
- mysql> CREATE TABLE Performance  
-> (  
-> Roll\_No INT,  
-> Course VARCHAR(10),  
-> Marks INT  
-> );
- mysql> INSERT INTO Performance VALUES  
-> (1, 'Math', 80),  
-> (1, 'English', 70),  
-> (2, 'Math', 75),  
-> (3, 'English', 80),  
-> (2, 'Physics', 65),  
-> (3, 'Math', 80);
- mysql> SELECT \* FROM Performance;  


| Roll_No | Course  | Marks |
|---------|---------|-------|
| 1       | Math    | 80    |
| 1       | English | 70    |
| 2       | Math    | 75    |
| 3       | English | 80    |
| 2       | Physics | 65    |
| 3       | Math    | 80    |

6 rows in set (0.00 sec)

# Query:

## 1. Consider the following SQL query.

```
SELECT S.Student_Name, sum (P.Marks)
FROM Student S, Performance P
WHERE S.Roll_No =P.Roll_No
GROUP BY S.Student_Name
```

### OUTPUT:

| Student_Name | sum(P.Marks) |
|--------------|--------------|
| Raj          | 310          |
| Rohit        | 140          |

The number of rows that will be returned by the SQL query is \_\_\_\_\_.

ANSWER: 2

## 11. GATE 2015

# Creation:

- mysql> CREATE TABLE Cinema  
-> (  
-> theater VARCHAR(20),  
-> address VARCHAR(50),  
-> capacity INT  
-> );
- mysql> INSERT INTO Cinema VALUES  
-> ('IMAX Phoenix', 'Vile Parle, Mumbai', 7000),  
-> ('IMAX SGC', 'SGC, Navi Mumbai', 9000),  
-> ('Rathore Cinema', 'Indira Mall, Delhi', 5000),  
-> ('Ujala Cinema', 'Shantiniketan Mall, Ranchi', 2000);
- mysql> SELECT \* FROM Cinema;

```
mysql> SELECT * FROM Cinema;
```

| theater        | address                    | capacity |
|----------------|----------------------------|----------|
| IMAX Phoenix   | Vile Parle, Mumbai         | 7000     |
| IMAX SGC       | SGC, Navi Mumbai           | 9000     |
| Rathore Cinema | Indira Mall, Delhi         | 5000     |
| Ujala Cinema   | Shantiniketan Mall, Ranchi | 2000     |

4 rows in set (0.00 sec)

# Query:

1. Consider the following relation Cinema (theater, address, capacity)

Which of the following options will be needed at the end of the SQL query

```
SELECT P1.address
```

```
FROM Cinema P1
```

such that it always finds the addresses of theaters of theaters with maximum capacity?

- (a) WHERE P1.capacity > = Any (select P2. capacity from Cinema P2)
- (b) WHERE P1.capacity > All (select max (P2. capacity) from Cinema P2)
- (c) WHERE P1.capacity > Any (select max (P2. capacity) from Cinema P2)
- (d) WHERE P1.capacity > = All (select P2. capacity from Cinema P2)

## OUTPUT:

Query (a) –

| address                    |
|----------------------------|
| Vile Parle, Mumbai         |
| SGC, Navi Mumbai           |
| Indira Mall, Delhi         |
| Shantiniketan Mall, Ranchi |

Query (b) –

```
Empty set (0.00 sec)
```

Query (c) –

```
Empty set (0.00 sec)
```

Query (d) –

| address          |
|------------------|
| SGC, Navi Mumbai |

**ANSWER:** Option (d) WHERE P1.capacity > = All (select P2. capacity from Cinema P2)