DBMS PRACTICAL FILE

NAME: LAKSHITA RAWAT

ROLL NO: 16102

COURSE: BSC(HONS)CS – SEC A

Q1

I. Create and use the following student-society database schema for a college to answer the given (sample) queries using the standalone SQL editor.

STUDENT	Roll No	StudentName	Course	DOB
	Char(6)	Varchar(20)	Varchar(10)	Date

SOCIETY	SocID	SocName	MentorName	TotalSeats
	Char(6)	Varchar(20)	Varchar(15)	Unsigned int

ENROLLMENT	Roll No	SID	DateOfEnrollment
	Char(6)	Char(6)	Date

CREATING STUDENT TABLE:

```
mysql> CREATE TABLE STUDENT (
    ->     RollNo CHAR(6) PRIMARY KEY,
    ->     StudentName VARCHAR(20),
    ->     Course VARCHAR(10),
    ->     DOB DATE
    -> );
Query OK, 0 rows affected (0.05 sec)
```

INSERTING VALUES IN STUDENT:

```
mysql> ALTER TABLE STUDENT MODIFY COLUMN Course VARCHAR(50);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> INSERT INTO STUDENT (RollNo, StudentName, Course, DOB) VALUES
    -> ('S001', 'John Doe', 'Computer Science', '2000-01-10'),
    -> ('S002', 'Jane Smith', 'Electrical Engineering', '1999-05-20'),
    -> ('S003', 'Alice Johnson', 'Physics', '2001-03-15');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

CREATING SOCIETY TABLE:

```
mysql> CREATE TABLE SOCIETY (
    ->     SocID CHAR(6) PRIMARY KEY,
    ->     SocName VARCHAR(20),
    ->     MentorName VARCHAR(15),
    ->     TotalSeats INT UNSIGNED
    -> );
Query OK, 0 rows affected (0.04 sec)
```

INSERTING VALUES IN SOCIETY TABLE:

```
mysql> INSERT INTO SOCIETY (SocID, SocName, MentorName, TotalSeats) VALUES
    -> ('SS001', 'Debate Club', 'Mr. Smith', 50),
    -> ('SS002', 'Chess Club', 'Ms. Johnson', 30),
    -> ('SS003', 'Drama Club', 'Mrs. Lee', 40);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

CREATING ENROLLMENT TABLE:

```
mysql> CREATE TABLE ENROLLMENT (
    ->     RollNo CHAR(6),
    ->     SID CHAR(6),
    ->     DateOfEnrollment DATE,
    ->     FOREIGN KEY (RollNo) REFERENCES STUDENT(RollNo),
    ->     FOREIGN KEY (SID) REFERENCES SOCIETY(SocID)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

INSERTING VALUES IN ENROLLMENT TABLE:

```
mysql> INSERT INTO ENROLLMENT (RollNo, SID, DateOfEnrollment) VALUES
    -> ('S001', 'SS001', '2023-02-14'),
    -> ('S002', 'SS002', '2023-03-20'),
    -> ('S003', 'SS003', '2023-01-05'),
    -> ('S001', 'SS003', '2023-04-10');
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

INSERTING MORE VALUES TO EACH TABLE:

```
mysql> INSERT INTO STUDENT (RollNo, StudentName, Course, DOB) VALUES
    -> ('S004', 'Emma Watson', 'Literature', '2000-07-25'),
    -> ('S005', 'Michael Johnson', 'Mathematics', '2002-04-12'),
    -> ('S006', 'Sophia Garcia', 'Computer Science', '2001-09-08'),
    -> ('S007', 'Daniel Lee', 'Physics', '2000-11-30'),
    -> ('S008', 'Olivia Martinez', 'Chemistry', '2001-03-18');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO SOCIETY (SocID, SocName, MentorName, TotalSeats) VALUES
    -> ('SS004', 'Music Club', 'Mr. Brown', 60),
    -> ('SS005', 'Art Club', 'Ms. White', 35),
    -> ('SS006', 'Photography Club', 'Mrs. Adams', 25),
    -> ('SS007', 'Coding Club', 'Mr. Wilson', 45),
    -> ('SS008', 'Basketball Team', 'Coach Johnson', 20);
Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO ENROLLMENT (RollNo, SID, DateOfEnrollment) VALUES
    -> ('S004', 'SS001', '2023-05-10'),
    -> ('S005', 'SS002', '2023-06-15'),
    -> ('S006', 'SS003', '2023-07-20'),
    -> ('S007', 'SS004', '2023-08-25'),
    -> ('S008', 'SS005', '2023-09-30');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

SHOW TABLE:

```
mysql> select *from student;
+--------+-----------------+-------------------------+------------+
| RollNo | StudentName     | Course                  | DOB        |
+--------+-----------------+-------------------------+------------+
| S001   | John Doe        | Computer Science        | 2000-01-10 |
| S002   | Jane Smith      | Electrical Engineering  | 1999-05-20 |
| S003   | Alice Johnson   | Physics                 | 2001-03-15 |
| S004   | Emma Watson     | Literature              | 2000-07-25 |
| S005   | Michael Johnson | Mathematics             | 2002-04-12 |
| S006   | Sophia Garcia   | Computer Science        | 2001-09-08 |
| S007   | Daniel Lee      | Physics                 | 2000-11-30 |
| S008   | Olivia Martinez | Chemistry               | 2001-03-18 |
+--------+-----------------+-------------------------+------------+
8 rows in set (0.00 sec)

mysql> select *from society;
+-------+------------------+---------------+------------+
| SocID | SocName          | MentorName    | TotalSeats |
+-------+------------------+---------------+------------+
| SS001 | Debate Club      | Mr. Smith     |         50 |
| SS002 | Chess Club       | Ms. Johnson   |         30 |
| SS003 | Drama Club       | Mrs. Lee      |         40 |
| SS004 | Music Club       | Mr. Brown     |         60 |
| SS005 | Art Club         | Ms. White     |         35 |
| SS006 | Photography Club | Mrs. Adams    |         25 |
| SS007 | Coding Club      | Mr. Wilson    |         45 |
| SS008 | Basketball Team  | Coach Johnson |         20 |
+-------+------------------+---------------+------------+
8 rows in set (0.00 sec)

mysql> select *from enrollment;
+--------+-------+------------------+
| RollNo | SID   | DateOfEnrollment |
+--------+-------+------------------+
| S001   | SS001 | 2023-02-14       |
| S002   | SS002 | 2023-03-20       |
| S003   | SS003 | 2023-01-05       |
| S001   | SS003 | 2023-04-10       |
| S004   | SS001 | 2023-05-10       |
| S005   | SS002 | 2023-06-15       |
| S006   | SS003 | 2023-07-20       |
| S007   | SS004 | 2023-08-25       |
| S008   | SS005 | 2023-09-30       |
+--------+-------+------------------+
```

QUERIES:

1. **Retrieve names of students enrolled in any society.**

```
mysql> SELECT DISTINCT s.StudentName
    -> FROM STUDENT s
    -> JOIN ENROLLMENT e ON s.RollNo = e.RollNo;
+-----------------+
| StudentName     |
+-----------------+
| John Doe        |
| Jane Smith      |
| Alice Johnson   |
| Emma Watson     |
| Michael Johnson |
| Sophia Garcia   |
| Daniel Lee      |
| Olivia Martinez |
+-----------------+
8 rows in set (0.00 sec)
```

2. **Retrieve all society names.**

```
mysql> SELECT SocName
    -> FROM SOCIETY;
+------------------+
| SocName          |
+------------------+
| Debate Club      |
| Chess Club       |
| Drama Club       |
| Music Club       |
| Art Club         |
| Photography Club |
| Coding Club      |
| Basketball Team  |
+------------------+
8 rows in set (0.00 sec)
```

3. **Retrieve students' names starting with letter ‘A’.**

```
mysql> SELECT StudentName
    -> FROM STUDENT
    -> WHERE StudentName LIKE 'A%';
+---------------+
| StudentName   |
+---------------+
| Alice Johnson |
+---------------+
1 row in set (0.00 sec)
```

4. **Retrieve students' details studying in courses ‘computer science’ or ‘chemistry’.**

```
mysql> SELECT *
    -> FROM STUDENT
    -> WHERE Course IN ('Computer Science', 'Chemistry');
+--------+-----------------+------------------+------------+
| RollNo | StudentName     | Course           | DOB        |
+--------+-----------------+------------------+------------+
| S001   | John Doe        | Computer Science | 2000-01-10 |
| S006   | Sophia Garcia   | Computer Science | 2001-09-08 |
| S008   | Olivia Martinez | Chemistry        | 2001-03-18 |
+--------+-----------------+------------------+------------+
3 rows in set (0.00 sec)
```

5. **Retrieve students’ names whose roll no either starts with ‘X’ or ‘Z’ and ends with ‘9’**

```
mysql> SELECT StudentName
    -> FROM STUDENT
    -> WHERE (RollNo LIKE 'X%' OR RollNo LIKE 'Z%') AND RollNo LIKE '%9';
Empty set (0.00 sec)
```

6. **Find society details with more than N TotalSeats where N is to be input by the user**

```
mysql> SET @N = 50; -- Example: Set N to 50
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT *
    -> FROM SOCIETY
    -> WHERE TotalSeats > @N;
+-------+------------+------------+------------+
| SocID | SocName    | MentorName | TotalSeats |
+-------+------------+------------+------------+
| SS004 | Music Club | Mr. Brown  |         60 |
+-------+------------+------------+------------+
1 row in set (0.00 sec)
```

7. **Update society table for mentor name of a specific society**

```
mysql> update society set MentorName='Keshav Goswami' where SocID='SS004'
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select *from society;
+-------+------------------+----------------+------------+
| SocID | SocName          | MentorName     | TotalSeats |
+-------+------------------+----------------+------------+
| SS001 | Debate Club      | Mr. Smith      |         50 |
| SS002 | Chess Club       | Ms. Johnson    |         30 |
| SS003 | Drama Club       | Mrs. Lee       |         40 |
| SS004 | Music Club       | Keshav Goswami |         60 |
| SS005 | Art Club         | Ms. White      |         35 |
| SS006 | Photography Club | Mrs. Adams     |         25 |
| SS007 | Coding Club      | Mr. Wilson     |         45 |
| SS008 | Basketball Team  | Coach Johnson  |         20 |
+-------+------------------+----------------+------------+
8 rows in set (0.00 sec)
```

8. **Find society names in which more than five students have enrolled**

```
mysql> SELECT s.SocName
    -> FROM SOCIETY s
    -> JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY s.SocName
    -> HAVING COUNT(*) > 5;
Empty set (0.00 sec)
```

9. **Find the name of youngest student enrolled in society ‘NSS’**

```
mysql> SELECT s.StudentName
    -> FROM STUDENT s
    -> JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> WHERE e.SID = 'NSS'
    -> ORDER BY s.DOB ASC
    -> LIMIT 1;
Empty set (0.00 sec)
```

10. **Find the name of most popular society (on the basis of enrolled students)**

```
mysql> SELECT s.SocName
    -> FROM SOCIETY s
    -> JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY s.SocName
    -> ORDER BY COUNT(*) DESC
    -> LIMIT 1;
+------------+
| SocName    |
+------------+
| Drama Club |
+------------+
1 row in set (0.00 sec)
```

11. Find the name of two least popular societies (on the basis of enrolled students)

```
mysql> SELECT s.SocName
    -> FROM SOCIETY s
    -> JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY s.SocName
    -> ORDER BY COUNT(*) ASC
    -> LIMIT 2;
+------------+
| SocName    |
+------------+
| Art Club   |
| Music Club |
+------------+
2 rows in set (0.00 sec)
```

12. Find the student names who are not enrolled in any society

```
mysql> SELECT StudentName
    -> FROM STUDENT
    -> WHERE RollNo NOT IN (SELECT RollNo FROM ENROLLMENT);
Empty set (0.00 sec)
```

13. Find the student names enrolled in at least two societies

```
mysql> SELECT s.StudentName
    -> FROM STUDENT s
    -> JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> GROUP BY s.StudentName
    -> HAVING COUNT(*) >= 2;
+-------------+
| StudentName |
+-------------+
| John Doe    |
+-------------+
1 row in set (0.00 sec)
```

14. Find society names in which maximum students are enrolled

```
mysql> SELECT s.SocName
    -> FROM SOCIETY s
    -> JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY s.SocName
    -> ORDER BY COUNT(*) DESC
    -> LIMIT 1;
+------------+
| SocName    |
+------------+
| Drama Club |
+------------+
1 row in set (0.00 sec)
```

15. Find names of all students who have enrolled in any society and society names in which at least one student has enrolled

```
mysql> SELECT DISTINCT s.StudentName, soc.SocName
    -> FROM STUDENT s
    -> LEFT JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> LEFT JOIN SOCIETY soc ON e.SID = soc.SocID;
+-----------------+-------------+
| StudentName     | SocName     |
+-----------------+-------------+
| John Doe        | Debate Club |
| John Doe        | Drama Club  |
| Jane Smith      | Chess Club  |
| Alice Johnson   | Drama Club  |
| Emma Watson     | Debate Club |
| Michael Johnson | Chess Club  |
| Sophia Garcia   | Drama Club  |
| Daniel Lee      | Music Club  |
| Olivia Martinez | Art Club    |
+-----------------+-------------+
9 rows in set (0.00 sec)
```

16. Find names of students who are enrolled in any of the three societies ‘Debating’, ‘Dancing’, and ‘Sashakt’.

```
mysql> SELECT DISTINCT s.StudentName
    -> FROM STUDENT s
    -> JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> JOIN SOCIETY soc ON e.SID = soc.SocID
    -> WHERE soc.SocName IN ('Debating', 'Dancing', 'Sashakt');
Empty set (0.00 sec)
```

17. Find society names such that its mentor has a name with ‘Gupta’ in it.

```
mysql> SELECT SocName
    -> FROM SOCIETY
    -> WHERE MentorName LIKE '%Gupta%';
Empty set (0.00 sec)
```

18. Find the society names in which the number of enrolled students is only 10% of its capacity.

```
mysql> SELECT s.SocName
    -> FROM SOCIETY s
    -> JOIN (
    ->     SELECT SID, COUNT(*) AS EnrolledCount
    ->     FROM ENROLLMENT
    ->     GROUP BY SID
    -> ) e ON s.SocID = e.SID
    -> WHERE EnrolledCount <= 0.1 * s.TotalSeats;
+-------------+
| SocName     |
+-------------+
| Debate Club |
| Chess Club  |
| Drama Club  |
| Music Club  |
| Art Club    |
+-------------+
5 rows in set (0.00 sec)
```

19. Display the vacant seats for each society.

```
mysql> SELECT s.SocName, (s.TotalSeats - IFNULL(e.EnrolledCount, 0)) AS VacantSeats
    -> FROM SOCIETY s
    -> LEFT JOIN (
    ->     SELECT SID, COUNT(*) AS EnrolledCount
    ->     FROM ENROLLMENT
    ->     GROUP BY SID
    -> ) e ON s.SocID = e.SID;
+------------------+-------------+
| SocName          | VacantSeats |
+------------------+-------------+
| Debate Club      |          48 |
| Chess Club       |          28 |
| Drama Club       |          37 |
| Music Club       |          59 |
| Art Club         |          34 |
| Photography Club |          25 |
| Coding Club      |          45 |
| Basketball Team  |          20 |
+------------------+-------------+
8 rows in set (0.00 sec)
```

20. Increment Total Seats of each society by 10%

```
mysql> UPDATE SOCIETY
    -> SET TotalSeats = TotalSeats * 1.1;
Query OK, 8 rows affected (0.03 sec)
Rows matched: 8  Changed: 8  Warnings: 0

mysql> select *from society;
+-------+------------------+----------------+------------+
| SocID | SocName          | MentorName     | TotalSeats |
+-------+------------------+----------------+------------+
| SS001 | Debate Club      | Mr. Smith      |         55 |
| SS002 | Chess Club       | Ms. Johnson    |         33 |
| SS003 | Drama Club       | Mrs. Lee       |         44 |
| SS004 | Music Club       | Keshav Goswami |         66 |
| SS005 | Art Club         | Ms. White      |         39 |
| SS006 | Photography Club | Mrs. Adams     |         28 |
| SS007 | Coding Club      | Mr. Wilson     |         50 |
| SS008 | Basketball Team  | Coach Johnson  |         22 |
+-------+------------------+----------------+------------+
8 rows in set (0.00 sec)
```

21. Add the enrollment fees paid (‘yes’/’No’) field in the enrollment table.

```
mysql> ALTER TABLE ENROLLMENT
    -> ADD COLUMN EnrollmentFeesPaid ENUM('Yes', 'No') DEFAULT 'No';
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

22. Update date of enrollment of society id ‘s1’ to ‘2018-01-15’, ‘s2’ to current date and ‘s3’ to ‘2018-01-02’.

```
mysql> UPDATE ENROLLMENT
    -> SET DateOfEnrollment = CASE
    ->     WHEN SID = 's1' THEN '2018-01-15'
    ->     WHEN SID = 's2' THEN CURDATE()
    ->     WHEN SID = 's3' THEN '2018-01-02'
    -> END;
Query OK, 9 rows affected (0.01 sec)
Rows matched: 9  Changed: 9  Warnings: 0

mysql> Select *from enrollment;
+--------+-------+------------------+--------------------+
| RollNo | SID   | DateOfEnrollment | EnrollmentFeesPaid |
+--------+-------+------------------+--------------------+
| S001   | SS001 | NULL             | No                 |
| S002   | SS002 | NULL             | No                 |
| S003   | SS003 | NULL             | No                 |
| S001   | SS003 | NULL             | No                 |
| S004   | SS001 | NULL             | No                 |
| S005   | SS002 | NULL             | No                 |
| S006   | SS003 | NULL             | No                 |
| S007   | SS004 | NULL             | No                 |
| S008   | SS005 | NULL             | No                 |
+--------+-------+------------------+--------------------+
9 rows in set (0.00 sec)
```

23. Create a view to keep track of society names with the total number of students enrolled in it.

```
mysql> CREATE VIEW Society_Enrollment_Count AS
    -> SELECT s.SocName, COUNT(e.RollNo) AS TotalEnrolled
    -> FROM SOCIETY s
    -> LEFT JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY s.SocName;
Query OK, 0 rows affected (0.01 sec)
```

24. Find student names enrolled in all the societies.

```
mysql> SELECT s.StudentName
    -> FROM STUDENT s
    -> WHERE NOT EXISTS (
    ->     SELECT s.SocID
    ->     FROM SOCIETY s
    ->     WHERE NOT EXISTS (
    ->         SELECT e.RollNo
    ->         FROM ENROLLMENT e
    ->         WHERE e.RollNo = s.RollNo AND e.SID = s.SocID
    ->     )
    -> );
Empty set (0.00 sec)
```

25. Count the number of societies with more than 5 students enrolled in it

```
mysql> SELECT COUNT(*)
    -> FROM (
    ->     SELECT SID
    ->     FROM ENROLLMENT
    ->     GROUP BY SID
    ->     HAVING COUNT(*) > 5
    -> ) AS SocietyCount;
+----------+
| COUNT(*) |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)
```

26. Add column Mobile number in student table with default value '9999999999'

```
mysql> ALTER TABLE STUDENT
    -> ADD COLUMN MobileNumber VARCHAR(15) DEFAULT '9999999999';
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select *from student;
+--------+-----------------+-------------------------+------------+--------------+
| RollNo | StudentName     | Course                  | DOB        | MobileNumber |
+--------+-----------------+-------------------------+------------+--------------+
| S001   | John Doe        | Computer Science        | 2000-01-10 | 9999999999   |
| S002   | Jane Smith      | Electrical Engineering  | 1999-05-20 | 9999999999   |
| S003   | Alice Johnson   | Physics                 | 2001-03-15 | 9999999999   |
| S004   | Emma Watson     | Literature              | 2000-07-25 | 9999999999   |
| S005   | Michael Johnson | Mathematics             | 2002-04-12 | 9999999999   |
| S006   | Sophia Garcia   | Computer Science        | 2001-09-08 | 9999999999   |
| S007   | Daniel Lee      | Physics                 | 2000-11-30 | 9999999999   |
| S008   | Olivia Martinez | Chemistry               | 2001-03-18 | 9999999999   |
+--------+-----------------+-------------------------+------------+--------------+
8 rows in set (0.00 sec)
```

27. Find the total number of students whose age is > 20 years.

```
mysql> SELECT COUNT(*)
    -> FROM STUDENT
    -> WHERE TIMESTAMPDIFF(YEAR, DOB, CURDATE()) > 20;
+----------+
| COUNT(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT DISTINCT s.StudentName
    -> FROM STUDENT s
    -> JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> WHERE YEAR(s.DOB) = 2001;
+-----------------+
| StudentName     |
+-----------------+
| Alice Johnson   |
| Sophia Garcia   |
| Olivia Martinez |
+-----------------+
3 rows in set (0.00 sec)
```

28. Find names of students who are born in 2001 and are enrolled in at least one society.


```
mysql> SELECT COUNT(*)
    -> FROM STUDENT
    -> WHERE TIMESTAMPDIFF(YEAR, DOB, CURDATE()) > 20;
+----------+
| COUNT(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT DISTINCT s.StudentName
    -> FROM STUDENT s
    -> JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> WHERE YEAR(s.DOB) = 2001;
+-----------------+
| StudentName     |
+-----------------+
| Alice Johnson   |
| Sophia Garcia   |
| Olivia Martinez |
+-----------------+
3 rows in set (0.00 sec)
```

29. Count all societies whose name starts with ‘S’ and ends with ‘t’ and at least 5 students are enrolled in the society.

```
mysql> SELECT COUNT(*)
    -> FROM (
    ->     SELECT s.SocID
    ->     FROM SOCIETY s
    ->     JOIN ENROLLMENT e ON s.SocID = e.SID
    ->     WHERE s.SocName LIKE 'S%' AND s.SocName LIKE '%t'
    ->     GROUP BY s.SocID
    ->     HAVING COUNT(*) >= 5
    -> ) AS SocietyCount;
+----------+
| COUNT(*) |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)
```

30. Display the following information:Society name Mentor name Total Capacity Total Enrolled Unfilled Seats

```
mysql> SELECT s.SocName, s.MentorName, s.TotalSeats, IFNULL(e.TotalEnrolled, 0) AS TotalEnrolled, (s.TotalSeats - IFNULL(e.TotalEnrolled, 0)) AS UnfilledSeats
    -> FROM SOCIETY s
    -> LEFT JOIN (
    ->     SELECT SID, COUNT(*) AS TotalEnrolled
    ->     FROM ENROLLMENT
    ->     GROUP BY SID
    -> ) e ON s.SocID = e.SID;
+-----------------+---------------+------------+---------------+---------------+
| SocName         | MentorName    | TotalSeats | TotalEnrolled | UnfilledSeats |
+-----------------+---------------+------------+---------------+---------------+
| Debate Club     | Mr. Smith     |         55 |             2 |            53 |
| Chess Club      | Ms. Johnson   |         33 |             2 |            31 |
| Drama Club      | Mrs. Lee      |         44 |             3 |            41 |
| Music Club      | Keshav Goswami|         66 |             1 |            65 |
| Art Club        | Ms. White     |         39 |             1 |            38 |
| Photography Club| Mrs. Adams    |         28 |             0 |            28 |
| Coding Club     | Mr. Wilson    |         50 |             0 |            50 |
| Basketball Team | Coach Johnson |         22 |             0 |            22 |
+-----------------+---------------+------------+---------------+---------------+
8 rows in set (0.00 sec)
```

Q2

II. **Do the following database administration commands:**

create user, create role, grant privileges to a role, revoke privileges from a role, create index

```
mysql> CREATE ROLE 'owner','admin';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL ON KESHAVMAHAVIDYALAYA.* TO 'owner';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT,INSERT,UPDATE ON KESHAVMAHAVIDYALAYA.* TO 'admin';
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER 'teacher'  IDENTIFIED BY 'teacher@kmv';
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER 'principal' IDENTIFIED BY 'principal@kmv';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT 'owner' TO 'principal';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT 'admin' TO 'teacher';
Query OK, 0 rows affected (0.01 sec)

mysql> REVOKE UPDATE ON KESHAVMAHAVIDYALAYA.* FROM 'admin';
Query OK, 0 rows affected (0.01 sec)
```

Q3

III. **Execute queries given in part I through a high-level language using ODBC connection.**

CODE OF THIRD PART:

```
import mysql.connector

db= mysql.connector.connect(
    host = "localhost",
    user = "root",
    password = "Divya@1234",
```

```
    database = "KESHAVMAHAVIDYALAYA"
)
cursor=db.cursor()


# Function to execute SQL queries
def execute_query(query):
    cursor.execute(query)
    rows = cursor.fetchall()

    print("Answer of Executed Query : ")
    print("--"*130)
    for row in rows:
        print(row)
    print("--"*130)

# Function to display menu options
def display_menu():
    print("\t\tMenu:")
    print("--"*130)
    print("1. Retrieve names of students enrolled in
any society.")
    print("2. Retrieve all society names.")
```

```
    print("3. Retrieve students' names starting with letter ‘A’.")
    print("4. Retrieve students' details studying in Courses ‘computer science’ or ‘chemistry’.")
    print("5. Retrieve students’ names whose roll no either starts with ‘X’ or ‘Z’ and ends with ‘9’.")
    print("6. Find society details with more than N TotalSeats where N is to be input by the user.")
    print("7. Update society table for mentor name of a specific society.")
    print("8. Find society names in which more than five students have enrolled.")
    print("9. Find the name of youngest student enrolled in society ‘NSS’.")
    print("10. Find the name of most popular society (on the basis of enrolled students).")
    print("11. Find the name of two least popular societies (on the basis of enrolled students).")
    print("12. Find the student names who are not enrolled in any society.")
    print("13. Find the student names enrolled in at least two societies.")
    print("14. Find society names in which maximum students are enrolled.")
```

```
    print("15. Find names of all students who have enrolled in any society and society names in which at least one student has enrolled.")
    print("16. Find names of students who are enrolled in any of the three societies ‘Debating’, ‘Dancing’ and ‘Sashakt’.")
    print("17. Find society names such that its mentor has a name with ‘Gupta’ in it.")
    print("18. Find the society names in which the number of enrolled students is only 10% of its capacity.")
    print("19. Display the vacant seats for each society.")
    print("20. Increment Total Seats of each society by 10%.")
    print("21. Add the enrollment fees paid ('yes'/'No') field in the enrollment table.")
    print("22. Update date of enrollment of society id ‘s1’ to ‘2018-01-15’, ‘s2’ to current date and ‘s3’ to ‘2018-01-02’.")
    print("23. Create a view to keep track of society names with the total number of students enrolled in it.")
```

```
    print("24. Find student names enrolled in all the societies.")
    print("25. Count the number of societies with more than 5 students enrolled in it.")
    print("26. Add column Mobile number in student table with default value ‘9999999999’.")
    print("27. Find the total number of students whose age is > 20 years.")
    print("28. Find names of students who are born in 2001 and are enrolled in at least one society.")
    print("29. Count all societies whose name starts with ‘S’ and ends with ‘t’ and at least 5 students are enrolled in the society.")
    print("30. Display the following information: Society name, Mentor name, Total Capacity, Total Enrolled, Unfilled Seats")
    print("0. Exit")
    print("--"*130)


# Main function
def main():
    while True:
        display_menu()
```

```
        choice = input("Enter your choice: ")

        if choice == '0':
            print("Exiting program.")
            break
        elif choice == '1':
            execute_query("""
                select distinct(Student_name) from student
join enrollment as e on e.Roll_no = student.Roll_no;
            """)
        elif choice == '2':
            execute_query("select SocName from
society;")
        elif choice == '3':
            execute_query("select  Student_name from
student where Student_name like 'A%';")
        elif choice == '4':
            execute_query("select * from student where
Course = 'CS' or Course = 'BMS';")
        elif choice == '5':
            execute_query("select  * from student where
Roll_no like '1%' or Roll_no like '2%' and Roll_no like
'%2';")
        elif choice == '6':
```

```
            num = int(input("Enter the value of N: "))
            execute_query(f"Select * from society where
TotalSeats > {num};")
        elif choice == '7':
            execute_query("update  society set
MentorName='Akshay' where SocName='NSS';")
        elif choice == '8':
            execute_query("""
                select SocName from society
    join enrollment as e on e.SID = society.SocID
    group by SocName
    having count(SocName)>3;
            """)
        elif choice == '9':
            execute_query("""
                select Student_name from student
    join enrollment as e on student.Roll_no=e.Roll_no
    join society as s on e.SID=s.SocID
    order by DOB desc
    limit 1;
            """)
        elif choice == '10':
            execute_query("""
```

```
            select SocName,count(SocName) from
society
    join enrollment as e on e.SID=society.SocID
    group by SocName order by count(SocName) desc
    limit 1;
            """)
        elif choice == '11':
            execute_query("""
                select SocName,count(SocName) from
society
    join enrollment as e on e.SID = society.SocID
    group by SocName order by COUNT(SocName) asc
    limit 2;
            """)
        elif choice == '12':
            execute_query("""
                select Student_name from student
    where Student_name not in (select Student_name
from student, enrollment where
student.Roll_no=enrollment.Roll_no);
            """)
        elif choice == '13':
            execute_query("""
```

```
            select Student_name from
student,enrollment where
student.Roll_no=enrollment.Roll_no
    group by Student_name
    having count(Student_name)>1;
            """)
        elif choice == '14':
            execute_query("""
                select SocName,count(SocName) from
society,enrollment where
society.SocID=enrollment.SID
    group by SocName;
            """)
        elif choice == '15':
            execute_query("""
                -- Names of all students who have enrolled
in any society
                select distinct(Student_name) from
student,enrollment where
student.Roll_no=enrollment.Roll_no;
                -- Society names in which at least a student
is enrolled
```

```
            select distinct(SocName) from
society,enrollment where
society.SocID=enrollment.SID;
            """)
        elif choice == '16':
            execute_query("""
                select distinct(Student_name),SocName
from student
    join enrollment as e on e.Roll_no=student.Roll_no
    join society as s on s.SocID=e.SID
    where SocName='Naksh' or SocName='NSS' or
SocName='Shades';
            """)
        elif choice == '17':
            execute_query("""
                select SocName from society
    where MentorName like '%i%';
            """)
        elif choice == '18':
            execute_query("""
                select
society.SocName,count(society.SocName) from
society
```

```
    join (select SocName,count(SocName) as enrolled
from society join enrollment on
enrollment.SID=society.SocID group by SocName)
    as new on new.SocName=society.SocName
    where enrolled>=0.1*TotalSeats group by
society.SocName;
            """)
        elif choice == '19':
            execute_query("""
                select society.SocName,society.TotalSeats-
new.enrolled as vacant_seats from society
    join (select SocName,count(SocName) as enrolled
from society join enrollment on
enrollment.SID=society.SocID group by SocName)
    as new on new.SocName=society.SocName;
            """)
        elif choice == '20':
            execute_query("update society set
TotalSeats=TotalSeats+0.1*TotalSeats;")
        elif choice == '21':
            execute_query("alter table enrollment add
column enrollment_fees_paid varchar(3);")
        elif choice == '22':
            execute_query("""
```

```
            set @currdate=current_date();
    update enrollment set DateOfEnrollment=
    case SID
    when '1603' then '2018-01-15'
    when '1604' then @currdate
    when '1605' then '2018-01-02'
    end
    where SID in ('1603','1604','1605');
        """)
      elif choice == '23':
        execute_query("""
            create view students_in_society as
    select SocName,count(SocName) from society join
enrollment as e on e.SID=society.SocID
    group by SocName;
        """)
      elif choice == '24':
        no_of_societies = int(input("Enter the number
of societies in your database: "))
        execute_query(f"""
            select Student_name from
    (select *,count(enrollment.SID) as
societies_enrolled from student join enrollment on
```

```
enrollment.Roll_no= student.Roll_no group by
Student_name) as enrolled
   where societies_enrolled={no_of_societies};
         """)
      elif choice == '25':
         execute_query("""
            select count(SocName) from (select
SocName,count(SocName) as enrolled from
society,enrollment where
enrollment.SID=society.SocID group by SocName) as
new where enrolled>2;
         """)
      elif choice == '26':
         execute_query("alter table student add
column mobile_number int default 99999999;")
      elif choice == '27':
         execute_query("""
            select count(*) from
   (select timestampdiff(year,DOB,current_date()) as
age from student) as new
   where age>20;
         """)
      elif choice == '28':
         execute_query("""
```

```
            select Student_name from
student,enrollment where
student.Roll_no=enrollment.Roll_no and DOB like
'%2001%';
            """)
        elif choice == '29':
            execute_query("""
                select count(SocName) from
    (select SocName,count(SocName) as enrolled from
society,enrollment
    where society.SocID=enrollment.SID
    group by SocName) as new
    where enrolled>2 and SocName like "N%S";
            """)
        elif choice == '30':
            execute_query("""
                select
society.SocName,MentorName,TotalSeats,enrolled,(T
otalSeats-enrolled) as unfilled
    from
    (select society.SocName,count(society.SocName)
as enrolled from society
    Natural Join enrollment group by society.socName)
     as new_enrolled,society
```

```
    where society.SocName=new_enrolled.SocName;
        """)
      else:
        print("Invalid choice. Please enter a valid
option.")


if _name_ == "_main_":
   main()
```

Q4

IV. **Students should implement the COMPANY database schema from Chapter 3 [1] and execute the solved queries of Chapter 7 [1].**

TABLE CREATION:

```
mysql> CREATE TABLE EMPLOYEE (
    ->     Fname VARCHAR(50),
    ->     Minit CHAR(1),
    ->     Lname VARCHAR(50),
    ->     Ssn CHAR(9) PRIMARY KEY,
    ->     Bdate DATE,
    ->     Address VARCHAR(100),
    ->     Sex CHAR(1),
    ->     Salary DECIMAL(10, 2),
    ->     Super_ssn CHAR(9),
    ->     Dno INT
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE DEPARTMENT (
    ->     Dname VARCHAR(50),
    ->     Dnumber INT PRIMARY KEY,
    ->     Mgr_ssn CHAR(9),
    ->     Mgr_start_date DATE
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE DEPT_LOCATIONS (
    ->     Dnumber INT,
    ->     Dlocation VARCHAR(100),
    ->     PRIMARY KEY (Dnumber, Dlocation)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE PROJECT (
    ->     Pname VARCHAR(50),
    ->     Pnumber INT PRIMARY KEY,
    ->     Plocation VARCHAR(100),
    ->     Dnum INT,
    ->     FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> CREATE TABLE WORKS_ON (
    ->     Essn CHAR(9),
    ->     Pno INT,
    ->     Hours DECIMAL(5, 2),
    ->     PRIMARY KEY (Essn, Pno),
    ->     FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
    ->     FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE DEPENDENT (
    ->     Essn CHAR(9),
    ->     Dependent_name VARCHAR(50),
    ->     Sex CHAR(1),
    ->     Bdate DATE,
    ->     Relationship VARCHAR(20),
    ->     PRIMARY KEY (Essn, Dependent_name),
    ->     FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

INSERTING VALUES IN TABLES:

```
mysql> INSERT INTO EMPLOYEE (Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno)
    -> VALUES
    -> ('John', 'M', 'Doe', '123456789', '1990-05-15', '123 Main St', 'M', 50000.00, NULL, 1),
    -> ('Jane', 'K', 'Smith', '234567890', '1992-08-20', '456 Elm St', 'F', 60000.00, '123456789', 2),
    -> ('Alice', 'L', 'Johnson', '345678901', '1988-03-10', '789 Oak St', 'F', 55000.00, '123456789', 1),
    -> ('Bob', 'P', 'Williams', '456789012', '1995-01-25', '101 Maple St', 'M', 52000.00, '234567890', 3),
    -> ('Michael', 'J', 'Brown', '567890123', '1987-11-30', '202 Pine St', 'M', 65000.00, '234567890', 2),
    -> ('Emily', 'S', 'Davis', '678901234', '1993-06-05', '303 Cedar St', 'F', 48000.00, '345678901', 3),
    -> ('David', 'R', 'Martinez', '789012345', '1990-09-12', '404 Birch St', 'M', 70000.00, '345678901', 1),
    -> ('Sarah', 'T', 'Wilson', '890123456', '1994-04-18', '505 Walnut St', 'F', 59000.00, '789012345', 2);
Query OK, 8 rows affected (0.02 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO DEPARTMENT (Dname, Dnumber, Mgr_ssn, Mgr_start_date)
    -> VALUES
    -> ('HR', 1, '123456789', '2020-01-01'),
    -> ('Finance', 2, '234567890', '2019-05-01'),
    -> ('IT', 3, '345678901', '2018-10-01');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO DEPT_LOCATIONS (Dnumber, Dlocation)
    -> VALUES
    -> (1, 'New York'),
    -> (2, 'Los Angeles'),
    -> (3, 'Chicago');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO PROJECT (Pname, Pnumber, Plocation, Dnum)
    -> VALUES
    -> ('Employee Portal', 1, 'New York', 1),
    -> ('Financial Analysis', 2, 'Los Angeles', 2),
    -> ('Database Upgrade', 3, 'Chicago', 3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO WORKS_ON (Essn, Pno, Hours)
    -> VALUES
    -> ('123456789', 1, 40),
    -> ('234567890', 2, 35),
    -> ('345678901', 3, 45),
    -> ('456789012', 1, 30),
    -> ('567890123', 2, 40),
    -> ('678901234', 3, 35),
    -> ('789012345', 1, 40),
    -> ('890123456', 2, 45);
Query OK, 8 rows affected (0.03 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO DEPENDENT (Essn, Dependent_name, Sex, Bdate, Relationship)
    -> VALUES
    -> ('123456789', 'Alice Doe', 'F', '2010-03-01', 'Daughter'),
    -> ('234567890', 'Bob Smith', 'M', '2008-05-15', 'Son'),
    -> ('345678901', 'Mary Johnson', 'F', '2012-09-20', 'Daughter'),
    -> ('456789012', 'James Williams', 'M', '2015-11-10', 'Son'),
    -> ('567890123', 'Emma Brown', 'F', '2018-02-05', 'Daughter'),
    -> ('678901234', 'Charlie Davis', 'M', '2014-07-20', 'Son'),
    -> ('789012345', 'Olivia Martinez', 'F', '2016-10-12', 'Daughter'),
    -> ('890123456', 'Noah Wilson', 'M', '2019-12-25', 'Son');
Query OK, 8 rows affected (0.01 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

SHOWING TABLES:

```
mysql> select *from department;
+----------+---------+-----------+----------------+
| Dname    | Dnumber | Mgr_ssn   | Mgr_start_date |
+----------+---------+-----------+----------------+
| HR       |       1 | 123456789 | 2020-01-01     |
| Finance  |       2 | 234567890 | 2019-05-01     |
| IT       |       3 | 345678901 | 2018-10-01     |
+----------+---------+-----------+----------------+
3 rows in set (0.00 sec)

mysql> select *from dependent;
+-----------+------------------+------+------------+--------------+
| Essn      | Dependent_name   | Sex  | Bdate      | Relationship |
+-----------+------------------+------+------------+--------------+
| 123456789 | Alice Doe        | F    | 2010-03-01 | Daughter     |
| 234567890 | Bob Smith        | M    | 2008-05-15 | Son          |
| 345678901 | Mary Johnson     | F    | 2012-09-20 | Daughter     |
| 456789012 | James Williams   | M    | 2015-11-10 | Son          |
| 567890123 | Emma Brown       | F    | 2018-02-05 | Daughter     |
| 678901234 | Charlie Davis    | M    | 2014-07-20 | Son          |
| 789012345 | Olivia Martinez  | F    | 2016-10-12 | Daughter     |
| 890123456 | Noah Wilson      | M    | 2019-12-25 | Son          |
+-----------+------------------+------+------------+--------------+
8 rows in set (0.00 sec)

mysql> select *from dept_locations;
+---------+-------------+
| Dnumber | Dlocation   |
+---------+-------------+
|       1 | New York    |
|       2 | Los Angeles |
|       3 | Chicago     |
+---------+-------------+
3 rows in set (0.00 sec)
```

```
mysql> select *from employee;
+---------+-------+----------+-----------+------------+---------------+------+----------+-----------+------+
| Fname   | Minit | Lname    | Ssn       | Bdate      | Address       | Sex  | Salary   | Super_ssn | Dno  |
+---------+-------+----------+-----------+------------+---------------+------+----------+-----------+------+
| John    | M     | Doe      | 123456789 | 1990-05-15 | 123 Main St   | M    | 50000.00 | NULL      |    1 |
| Jane    | K     | Smith    | 234567890 | 1992-08-20 | 456 Elm St    | F    | 60000.00 | 123456789 |    2 |
| Alice   | L     | Johnson  | 345678901 | 1988-03-10 | 789 Oak St    | F    | 55000.00 | 123456789 |    1 |
| Bob     | P     | Williams | 456789012 | 1995-01-25 | 101 Maple St  | M    | 52000.00 | 234567890 |    3 |
| Michael | J     | Brown    | 567890123 | 1987-11-30 | 202 Pine St   | M    | 65000.00 | 234567890 |    2 |
| Emily   | S     | Davis    | 678901234 | 1993-06-05 | 303 Cedar St  | F    | 48000.00 | 345678901 |    3 |
| David   | R     | Martinez | 789012345 | 1990-09-12 | 404 Birch St  | M    | 70000.00 | 345678901 |    1 |
| Sarah   | T     | Wilson   | 890123456 | 1994-04-18 | 505 Walnut St | F    | 59000.00 | 789012345 |    2 |
+---------+-------+----------+-----------+------------+---------------+------+----------+-----------+------+
8 rows in set (0.00 sec)

mysql> select *from project;
+-------------------+---------+-------------+------+
| Pname             | Pnumber | Plocation   | Dnum |
+-------------------+---------+-------------+------+
| Employee Portal   |       1 | New York    |    1 |
| Financial Analysis|       2 | Los Angeles |    2 |
| Database Upgrade  |       3 | Chicago     |    3 |
+-------------------+---------+-------------+------+
3 rows in set (0.00 sec)

mysql> select *from works_on;
+-----------+-----+-------+
| Essn      | Pno | Hours |
+-----------+-----+-------+
| 123456789 |   1 | 40.00 |
| 234567890 |   2 | 35.00 |
| 345678901 |   3 | 45.00 |
| 456789012 |   1 | 30.00 |
| 567890123 |   2 | 40.00 |
| 678901234 |   3 | 35.00 |
| 789012345 |   1 | 40.00 |
| 890123456 |   2 | 45.00 |
+-----------+-----+-------+
8 rows in set (0.00 sec)
```

QUERIES:

```
mysql> SELECT E.Fname, E.Lname
    -> FROM EMPLOYEE E
    -> LEFT JOIN EMPLOYEE S ON E.Super_ssn = S.Ssn
    -> WHERE S.Ssn IS NULL;
+-------+-------+
| Fname | Lname |
+-------+-------+
| John  | Doe   |
+-------+-------+
1 row in set (0.00 sec)
```

```
mysql> SELECT E.Fname, E.Lname
    -> FROM EMPLOYEE E
    -> LEFT JOIN DEPENDENT D ON E.Ssn = D.Essn
    -> WHERE D.Essn IS NULL;
Empty set (0.00 sec)
```

```
mysql> SELECT E.Fname, E.Lname
    -> FROM EMPLOYEE E
    -> WHERE NOT EXISTS (
    ->     SELECT Pnumber
    ->     FROM PROJECT
    ->     WHERE Dnum = 5
    ->     EXCEPT
    ->     SELECT W.Pno
    ->     FROM WORKS_ON W
    ->     WHERE W.Essn = E.Ssn
    -> );
+---------+----------+
| Fname   | Lname    |
+---------+----------+
| John    | Doe      |
| Jane    | Smith    |
| Alice   | Johnson  |
| Bob     | Williams |
| Michael | Brown    |
| Emily   | Davis    |
| David   | Martinez |
| Sarah   | Wilson   |
+---------+----------+
8 rows in set (0.00 sec)
```

```
mysql> SELECT DISTINCT Essn
    -> FROM WORKS_ON
    -> WHERE Pno IN (1, 2, 3);
+-----------+
| Essn      |
+-----------+
| 123456789 |
| 234567890 |
| 345678901 |
| 456789012 |
| 567890123 |
| 678901234 |
| 789012345 |
| 890123456 |
+-----------+
8 rows in set (0.00 sec)
```

```
mysql> SELECT SUM(Salary) AS Total_Salary, MAX(Salary) AS Max_Salary, MIN(Salary) AS Min_Salary, AVG(Salary) AS Avg_Salary
    -> FROM EMPLOYEE;
+--------------+------------+------------+--------------+
| Total_Salary | Max_Salary | Min_Salary | Avg_Salary   |
+--------------+------------+------------+--------------+
|    459000.00 |   70000.00 |   48000.00 | 57375.000000 |
+--------------+------------+------------+--------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT SUM(E.Salary) AS Total_Salary,
    ->        MAX(E.Salary) AS Max_Salary,
    ->        MIN(E.Salary) AS Min_Salary,
    ->        AVG(E.Salary) AS Avg_Salary
    -> FROM EMPLOYEE E
    -> JOIN DEPARTMENT D ON E.Dno = D.Dnumber
    -> WHERE D.Dname = 'Research';
+--------------+------------+------------+------------+
| Total_Salary | Max_Salary | Min_Salary | Avg_Salary |
+--------------+------------+------------+------------+
|         NULL |       NULL |       NULL |       NULL |
+--------------+------------+------------+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT
    ->     (SELECT COUNT(*) FROM EMPLOYEE) AS Total_Employees,
    ->     (SELECT COUNT(*) FROM EMPLOYEE WHERE Dno = (SELECT Dnumber FROM DEPARTMENT WHERE Dname = 'Research')) AS Research_Department_Employees;
+-----------------+-----------------------------+
| Total_Employees | Research_Department_Employees |
+-----------------+-----------------------------+
|               8 |                           0 |
+-----------------+-----------------------------+
1 row in set (0.01 sec)
```

```
mysql> SELECT COUNT(DISTINCT Salary) AS Distinct_Salary_Count
    -> FROM EMPLOYEE;
+------------------------+
| Distinct_Salary_Count  |
+------------------------+
|                      8 |
+------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT D.Dnumber, COUNT(E.Ssn) AS Num_Employees, AVG(E.Salary) AS Avg_Salary
    -> FROM DEPARTMENT D
    -> LEFT JOIN EMPLOYEE E ON D.Dnumber = E.Dno
    -> GROUP BY D.Dnumber;
+---------+---------------+--------------+
| Dnumber | Num_Employees | Avg_Salary   |
+---------+---------------+--------------+
|       1 |             3 | 58333.333333 |
|       2 |             3 | 61333.333333 |
|       3 |             2 | 50000.000000 |
+---------+---------------+--------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT P.Pnumber, P.Pname, COUNT(W.Essn) AS Num_Employees
    -> FROM PROJECT P
    -> LEFT JOIN WORKS_ON W ON P.Pnumber = W.Pno
    -> GROUP BY P.Pnumber, P.Pname;
+---------+--------------------+---------------+
| Pnumber | Pname              | Num_Employees |
+---------+--------------------+---------------+
|       1 | Employee Portal    |             3 |
|       2 | Financial Analysis |             3 |
|       3 | Database Upgrade   |             2 |
+---------+--------------------+---------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT P.Pnumber, P.Pname, COUNT(W.Essn) AS Num_Employees
    -> FROM PROJECT P
    -> JOIN WORKS_ON W ON P.Pnumber = W.Pno
    -> GROUP BY P.Pnumber, P.Pname
    -> HAVING COUNT(W.Essn) > 2;
+---------+--------------------+---------------+
| Pnumber | Pname              | Num_Employees |
+---------+--------------------+---------------+
|       1 | Employee Portal    |             3 |
|       2 | Financial Analysis |             3 |
+---------+--------------------+---------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT P.Pnumber, P.Pname, COUNT(W.Essn) AS Num_Employees
    -> FROM PROJECT P
    -> JOIN WORKS_ON W ON P.Pnumber = W.Pno
    -> JOIN EMPLOYEE E ON W.Essn = E.Ssn
    -> WHERE E.Dno = 5
    -> GROUP BY P.Pnumber, P.Pname;
Empty set (0.00 sec)

mysql> SELECT E.Dno, COUNT(*) AS Num_Employees
    -> FROM EMPLOYEE E
    -> WHERE E.Salary > 40000
    -> GROUP BY E.Dno
    -> HAVING COUNT(*) > 5;
Empty set (0.00 sec)
```