Name: SAHIL

Course: B.Sc. (H) Computer Science

Roll No.: 20221474

Submitted to: Kamlesh Kumar Raghuvanshi

Practical : Database Management System

Creating Database and Tables

Creating database:

Query: create database sahil;
use sahil;

Output:

```
mysql> create database sahil;
Query OK, 1 row affected (0.01 sec)

mysql> use sahil;
Database changed
```

Create table student:

Query:

```
create table STUDENT (
RollNo Char(6) primary key,
```

StudentName Varchar(20),

Course Varchar(10),

DOB date

);

Output:

```
mysql>  create table STUDENT (
    ->  RollNo Char(6) primary key,
    ->   StudentName Varchar(20),
    ->   Course Varchar(10),
    ->   DOB date
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Create table Society:

Query:

create table SOCIETY (

SocID Char(6) primary key,

SocName Varchar(20),

MentorName Varchar(15),

TotalSeats int unsigned

);

Output:

```
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)
```

Create table Enrollment:

Query:

```
create table ENROLLMENT (
  RollNo Char(6),
  SID Char(6),
  DateOfEnrollment date,
  foreign key (RollNo) references STUDENT(RollNo),
  foreign key (SID) references SOCIETY(SocID),
  primary key (RollNo, SID)
);
```

Output:

```
mysql> create table ENROLLMENT (
    ->   RollNo Char(6),
    ->   SID Char(6),
    ->   DateOfEnrollment date,
    ->   foreign key (RollNo) references STUDENT(RollNo),
    ->   foreign key (SID) references SOCIETY(SocID),
    ->   primary key (RollNo, SID)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

Q1. Retrieve names of students enrolled in any society.

Query:

```
select studentname
from student
inner join enrollment on STUDENT.RollNo = ENROLLMENT.RollNo;
```

Output:

```
mysql> select studentname
    -> from student
    -> inner join enrollment on STUDENT.RollNo = ENROLLMENT.RollNo;
+-------------+
| studentname |
+-------------+
| Munender    |
| Sushant     |
| Ritik       |
| Vishal      |
| Tanu        |
| Ravikant    |
| Nitin       |
| Neha        |
| Suraj       |
| Ritesh      |
| Mishti      |
| Nilam       |
| Chanchal    |
| Ashish      |
| Divya       |
| Muskan      |
| Usopp       |
| Sanji       |
| Nami        |
| Mohit       |
| Vijay       |
| Varun       |
| Arun        |
| Annu        |
+-------------+
24 rows in set (0.03 sec)
```

Q2. Retrieve all society names.

Query:

select socname from society;

Output:

```
mysql> select socname from society;
+----------+
| socname  |
+----------+
| NSS      |
| YUVA     |
| Sashakt  |
| Dancing  |
| Debating |
+----------+
5 rows in set (0.02 sec)
```

Q3. Retrieve students' names starting with letter ‘A’.

Query:

select studentname from student where studentname like 'A%';

Output:

```
mysql> select studentname from student where studentname like 'A%';
+-------------+
| studentname |
+-------------+
| Anamika     |
| Aditya      |
| Aman        |
| Ashish      |
| Arun        |
| Ankit       |
| Arjun       |
| Annu        |
+-------------+
8 rows in set (0.01 sec)
```

Q4. Retrieve students' details studying in courses ‘computer science’ or ‘chemistry’.

Query:

select * from student where course in ('BSC(H)CHEM', 'BSC(H)CS');

Output:

```
mysql> select * from student where course in ('BSC(H)CHEM', 'BSC(H)CS');
+--------+-------------+------------+------------+
| RollNo | StudentName | Course     | DOB        |
+--------+-------------+------------+------------+
| 1433   | Ravikant    | Bsc(H)CS   | 2005-01-22 |
| 1455   | Munender    | Bsc(H)CS   | 2003-10-02 |
| 1468   | Mayank      | Bsc(H)CS   | 2003-06-23 |
| 1478   | Sushant     | Bsc(H)CS   | 2003-07-19 |
| x1234  | Rahul       | BSC(H)CHEM | 2001-10-28 |
| x1235  | Anamika     | BSC(H)CHEM | 2007-11-22 |
| x1238  | Aditya      | BSC(H)CHEM | 2001-11-15 |
| x1249  | Nishant     | BSC(H)CS   | 2001-03-17 |
| x2345  | Aman        | BSC(H)CHEM | 2001-01-23 |
| x2348  | Mohit       | BSC(H)CHEM | 2002-01-20 |
| x2349  | Harsh       | BSC(H)CHEM | 2002-08-21 |
| x2350  | Ashish      | BSC(H)CHEM | 2001-08-11 |
| x2351  | Sandeep     | BSC(H)CHEM | 2001-09-15 |
| x4569  | Sanjay      | BSC(H)CS   | 2001-05-23 |
| y1239  | Varun       | BSC(H)CS   | 2001-11-21 |
| y1240  | Arun        | BSC(H)CS   | 2001-01-01 |
| y1241  | Mishti      | BSC(H)CS   | 2001-02-05 |
| y1242  | Ankit       | BSC(H)CS   | 2001-03-07 |
| y1243  | Kavita      | BSC(H)CS   | 2001-06-02 |
| y1244  | Kiran       | BSC(H)CS   | 2002-07-12 |
| y1245  | Karan       | BSC(H)CS   | 2003-08-22 |
| y1246  | Suraj       | BSC(H)CS   | 2001-09-14 |
| y1247  | Sachin      | BSC(H)CS   | 2001-05-12 |
| z1249  | Nitin       | BSC(H)CS   | 2001-02-12 |
| z4579  | Arjun       | BSC(H)CS   | 2001-06-14 |
| z4589  | Himanshu    | BSC(H)CS   | 2003-04-17 |
+--------+-------------+------------+------------+
26 rows in set (0.01 sec)
```

Q5. Retrieve students’ names whose roll no either starts with ‘X’ or ‘Z’ and ends with ‘9’

Query:

select studentname from student where rollno like 'X%9' or rollno like 'Z%9';

Output:

```
mysql> select studentname from student where rollno like 'X%9' or rollno like 'Z%9';
+-------------+
| studentname |
+-------------+
| Nishant     |
| Harsh       |
| Sanjay      |
| Ritik       |
| Ritesh      |
| Vijay       |
| Nitin       |
| Arjun       |
| Himanshu    |
| Vishal      |
| Vikram      |
| Divya       |
| Tanu        |
| Neha        |
| Nilam       |
| Muskan      |
| Sheetal     |
| Riya        |
| Annu        |
+-------------+
19 rows in set (0.01 sec)
```

Q6. Find society details with more than N TotalSeats where N is to be input by the user

Query:

select * from society TotalSeats > 19;

Output:

```
mysql> select * from society where TotalSeats > 19;
+-------+---------+---------------+------------+
| SocID | SocName | MentorName    | TotalSeats |
+-------+---------+---------------+------------+
| 222   | YUVA    | Rajesh Singh  |         23 |
| 333   | Sashakt | Dr Ankit      |         48 |
| 444   | Dancing | Mahavir Phogat |         33 |
+-------+---------+---------------+------------+
3 rows in set (0.01 sec)
```

Q7. Update society table for mentor name of a specific society

Query:

update society

set MentorName = 'Rohit Sharma'

where SocName = 'Dancing';

Output:

```
mysql> update society
    -> set MentorName = 'Rohit Sharma'
    -> where SocName = 'Dancing';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Q8. Find society names in which more than five students have enrolled.

Query:

```
select SocName
from SOCIETY s
inner join ENROLLMENT e on s.SocID = e.SID
group by SocName
having count(*) > 5;
```

Output:

```
mysql> select SocName
    -> from SOCIETY s
    -> inner join ENROLLMENT e on s.SocID = e.SID
    -> group by SocName
    -> having count(*) > 5;
+----------+
| SocName  |
+----------+
| Debating |
+----------+
1 row in set (0.01 sec)
```

Q9. Find the name of the youngest student enrolled in society 'NSS'.

Query:

```
select StudentName, DOB
```

from student s

inner join ENROLLMENT e on s.RollNo = e.RollNo

where e.SID = '111'

order by DOB DESC

limit 1;

Output:

```
mysql> select StudentName, DOB
    -> from student s
    -> inner join ENROLLMENT e on s.RollNo = e.RollNo
    -> where e.SID = '111'
    -> order by DOB DESC
    -> limit 1;
+-------------+------------+
| StudentName | DOB        |
+-------------+------------+
| Tanu        | 2011-12-22 |
+-------------+------------+
1 row in set (0.00 sec)
```

Q10. Find the name of most popular society (on the basis of enrolled students)

Query:

select SocName

from society s

inner join ENROLLMENT e on s.SocID = e.SID

group by SocName

order by count(*) DESC

limit 1;

Output:

```
mysql>  select SocName
    -> from society s
    -> inner join ENROLLMENT e on s.SocID = e.SID
    -> group by SocName
    -> order by count(*) DESC
    -> limit 1;
+----------+
| SocName  |
+----------+
| Debating |
+----------+
1 row in set (0.00 sec)
```

Q11. Find the name of two least popular societies (on the basis of enrolled students)

Query:

select SocName

from society s

inner join ENROLLMENT e on s.SocID = e.SID

group by SocName

order by count(*) ASC

limit 2;

Output:

```
mysql> select SocName
    -> from society s
    -> inner join ENROLLMENT e on s.SocID = e.SID
    -> group by SocName
    -> order by count(*) ASC
    -> limit 2;
+---------+
| SocName |
+---------+
| YUVA    |
| Sashakt |
+---------+
2 rows in set (0.00 sec)
```

Q12. Find the student names who are not enrolled in any society

Query:

```
select StudentName
from STUDENT s
left join ENROLLMENT e on s.RollNo = e.RollNo
where e.SID is null;
```

Output:

```
mysql> select StudentName
    -> from STUDENT s
    -> left join ENROLLMENT e on s.RollNo = e.RollNo
    -> where e.SID is null;
+-------------+
| StudentName |
+-------------+
| Mayank      |
| Luffy       |
| Zoro        |
| Robin       |
| Priya       |
| Jyoti       |
| Rahul       |
| Anamika     |
| Aditya      |
| Nishant     |
| Aman        |
| Harsh       |
| Sandeep     |
| Sanjay      |
| Ankit       |
| Kavita      |
| Kiran       |
| Karan       |
| Suraj       |
| Sachin      |
| Arjun       |
| Himanshu    |
| Vikram      |
| Sheetal     |
| Riya        |
+-------------+
25 rows in set (0.00 sec)
```

Q13. Find the student names enrolled in at least two societies

Query:

SELECT StudentName

FROM STUDENT s

```
INNER JOIN ENROLLMENT e1 ON s.RollNo = e1.RollNo
INNER JOIN ENROLLMENT e2 ON s.RollNo = e2.RollNo
WHERE e1.SID <> e2.SID
GROUP BY StudentName
HAVING COUNT(*) >= 2;
```

Output:

```
mysql> SELECT StudentName
    -> FROM STUDENT s
    -> INNER JOIN ENROLLMENT e1 ON s.RollNo = e1.RollNo
    -> INNER JOIN ENROLLMENT e2 ON s.RollNo = e2.RollNo
    -> WHERE e1.SID <> e2.SID
    -> GROUP BY StudentName
    -> HAVING COUNT(*) >= 2;
Empty set (0.00 sec)
```

Q14. Find society names in which maximum students are enrolled

Query:

```
SELECT SocName, COUNT(*) AS TotalEnrolledStudents
FROM SOCIETY s
INNER JOIN ENROLLMENT e ON s.SocID = e.SID
GROUP BY SocName
ORDER BY TotalEnrolledStudents DESC
LIMIT 1;
```

Output:

```
mysql> SELECT SocName, COUNT(*) AS TotalEnrolledStudents
    -> FROM SOCIETY s
    -> INNER JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY SocName
    -> ORDER BY TotalEnrolledStudents DESC
    -> LIMIT 1;
+----------+-----------------------+
| SocName  | TotalEnrolledStudents |
+----------+-----------------------+
| Debating |                     8 |
+----------+-----------------------+
1 row in set (0.02 sec)
```

Q15. Find names of all students who have enrolled in any society and society names in which at least one student has enrolled

Query:

```
SELECT StudentName, SocName
FROM STUDENT s
INNER JOIN ENROLLMENT e ON s.RollNo = e.RollNo
INNER JOIN SOCIETY soc ON e.SID = soc.SocID;
```

Output:

```
mysql> SELECT StudentName, SocName
    -> FROM STUDENT s
    -> INNER JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> INNER JOIN SOCIETY soc ON e.SID = soc.SocID;
+-------------+----------+
| StudentName | SocName  |
+-------------+----------+
| Munender    | NSS      |
| Sushant     | NSS      |
| Ritik       | NSS      |
| Vishal      | NSS      |
| Tanu        | NSS      |
| Ravikant    | YUVA     |
| Nitin       | YUVA     |
| Neha        | YUVA     |
| Suraj       | Sashakt  |
| Ritesh      | Sashakt  |
| Mishti      | Sashakt  |
| Nilam       | Sashakt  |
| Chanchal    | Dancing  |
| Ashish      | Dancing  |
| Divya       | Dancing  |
| Muskan      | Dancing  |
| Usopp       | Debating |
| Sanji       | Debating |
| Nami        | Debating |
| Mohit       | Debating |
| Vijay       | Debating |
| Varun       | Debating |
| Arun        | Debating |
| Annu        | Debating |
+-------------+----------+
24 rows in set (0.00 sec)
```

Q16. Find names of students who are enrolled in any of the three societies ‘Debating’, ‘Dancing’and ‘Sashakt’

Query:

SELECT StudentName

FROM student s

JOIN ENROLLMENT e ON s.RollNo = e.RollNo

WHERE e.SID IN ('333', '444', '555');

Output:

```
mysql> SELECT StudentName
    -> FROM student s
    -> JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> WHERE e.SID IN ('333', '444', '555');
+-------------+
| StudentName |
+-------------+
| Suraj       |
| Ritesh      |
| Mishti      |
| Nilam       |
| Chanchal    |
| Ashish      |
| Divya       |
| Muskan      |
| Usopp       |
| Sanji       |
| Nami        |
| Mohit       |
| Vijay       |
| Varun       |
| Arun        |
| Annu        |
+-------------+
16 rows in set (0.00 sec)
```

Q17. Find society names such that its mentor has a name with ‘Gupta’ in it.

Query:

SELECT SocName, MentorName

FROM SOCIETY

WHERE MentorName LIKE '%Gupta%';

Output:

```
mysql> SELECT SocName, MentorName
    -> FROM SOCIETY
    -> WHERE MentorName LIKE '%Gupta%';
+---------+----------------+
| SocName | MentorName     |
+---------+----------------+
| NSS     | Akhilesh Gupta |
+---------+----------------+
1 row in set (0.00 sec)
```

Q18. Find the society names in which the number of enrolled students is only 10% of its capacity.

Query:

SELECT SocName, MAX(TotalSeats) AS TotalSeats, COUNT(*) AS EnrolledStudents

FROM SOCIETY s

INNER JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY SocName

HAVING COUNT(*) = 0.1 * MAX(TotalSeats);

Output:

```
mysql> SELECT SocName, MAX(TotalSeats) AS TotalSeats, COUNT(*) AS EnrolledStudents
    -> FROM SOCIETY s
    -> INNER JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY SocName
    -> HAVING COUNT(*) = 0.1 * MAX(TotalSeats);
Empty set (0.00 sec)
```

Q19. Display the vacant seats for each society.

Query:

```
SELECT SocName, MAX(TotalSeats) AS TotalSeats, COUNT(*) AS EnrolledStudents, MAX(TotalSeats) - COUNT(*) AS VacantSeats

FROM SOCIETY s

LEFT JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY SocName;
```

Output:

```
mysql> SELECT SocName, MAX(TotalSeats) AS TotalSeats, COUNT(*) AS EnrolledStudents, MAX(TotalSeats) - COUNT(*) AS VacantSeats
    -> FROM SOCIETY s
    -> LEFT JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY SocName;
+----------+------------+------------------+-------------+
| SocName  | TotalSeats | EnrolledStudents | VacantSeats |
+----------+------------+------------------+-------------+
| Dancing  |         33 |                4 |          29 |
| Debating |         17 |                8 |           9 |
| NSS      |         15 |                5 |          10 |
| Sashakt  |         48 |                4 |          44 |
| YUVA     |         23 |                3 |          20 |
+----------+------------+------------------+-------------+
5 rows in set (0.00 sec)
```

Q20. Increment Total Seats of each society by 10%

Query:

```
UPDATE SOCIETY

SET TotalSeats = TotalSeats * 1.1;
```

Output:

```
mysql> UPDATE SOCIETY
    -> SET TotalSeats = TotalSeats * 1.1;
Query OK, 5 rows affected (0.01 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

Q21. Add the enrollment fees paid (‘yes’/’No’) field in the enrollment table

Query:

```
ALTER TABLE ENROLLMENT

ADD EnrollmentFee VARCHAR(3) CHECK(EnrollmentFee IN ('yes', 'no'));
```

Output:

```
mysql> ALTER TABLE ENROLLMENT
    -> ADD EnrollmentFee VARCHAR(3) CHECK(EnrollmentFee IN ('yes', 'no'));
Query OK, 18 rows affected (0.10 sec)
Records: 18  Duplicates: 0  Warnings: 0
```

Q22. Update date of enrollment of society id ‘s1’ to ‘2018-01-15’, ‘s2’ to current date and ‘s3’ to ‘2018-01-02’.

Query:

```
UPDATE ENROLLMENT

SET DateOfEnrollment = '2018-01-15'

WHERE SID = 's1';
```

```
UPDATE ENROLLMENT
SET DateOfEnrollment = CURDATE()
WHERE SID = 's2';

UPDATE ENROLLMENT
SET DateOfEnrollment = '2018-01-02'
WHERE SID = 's3';
```

Output:

```
mysql> UPDATE ENROLLMENT
    -> SET DateOfEnrollment = '2018-01-15'
    -> WHERE SID = 's1';
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> UPDATE ENROLLMENT
    -> SET DateOfEnrollment = CURDATE()
    -> WHERE SID = 's2';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE ENROLLMENT
    -> SET DateOfEnrollment = '2018-01-02'
    -> WHERE SID = 's3';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Q23. Create a view to keep track of society names with the total number of students enrolled in it.

Query:

```
CREATE VIEW SocietyEnrollmentView
AS
SELECT SocName, COUNT(*) AS TotalEnrolledStudents
FROM SOCIETY s
INNER JOIN ENROLLMENT e ON s.SocID = e.SID
GROUP BY SocName;
```

Output:

```
mysql> CREATE VIEW SocietyEnrollmentView
    -> AS
    -> SELECT SocName, COUNT(*) AS TotalEnrolledStudents
    -> FROM SOCIETY s
    -> INNER JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY SocName;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

```
mysql> select * from SocietyEnrollmentView;
+----------+-----------------------+
| SocName  | TotalEnrolledStudents |
+----------+-----------------------+
| Dancing  |                     4 |
| Debating |                     8 |
| Jajba    |                     2 |
| Josh     |                     1 |
| NSS      |                     5 |
| Sashakt  |                     4 |
| Tedx     |                     1 |
| YUVA     |                     3 |
+----------+-----------------------+
8 rows in set (0.01 sec)
```

Q24. Find student names enrolled in all the societies.

Query:

```
SELECT StudentName
```

```
FROM STUDENT
WHERE RollNo IN (
    SELECT RollNo
    FROM ENROLLMENT
    GROUP BY RollNo
     HAVING COUNT(DISTINCT SID) = (SELECT COUNT(DISTINCT SID) FROM ENROLLMENT)
);
```

Output:

```
mysql> SELECT StudentName
    -> FROM STUDENT
    -> WHERE RollNo IN (
    ->     SELECT RollNo
    ->     FROM ENROLLMENT
    ->     GROUP BY RollNo
    ->      HAVING COUNT(DISTINCT SID) = (SELECT COUNT(DISTINCT SID) FROM ENROLLMENT)
    -> );
Empty set (0.00 sec)
```

Q25. Count the number of societies with more than 5 students enrolled in it

Query:

```
SELECT COUNT(*) AS TotalSocieties
FROM (
    SELECT s.SocID
    FROM SOCIETY s
    INNER JOIN ENROLLMENT e ON s.SocID = e.SID
    GROUP BY s.SocID
```

```
    HAVING COUNT(*) > 5
) AS SocietyCounts;
```

Output:

```
mysql> SELECT COUNT(*) AS TotalSocieties
    -> FROM (
    ->     SELECT s.SocID
    ->     FROM SOCIETY s
    ->     INNER JOIN ENROLLMENT e ON s.SocID = e.SID
    ->     GROUP BY s.SocID
    ->     HAVING COUNT(*) > 5
    -> ) AS SocietyCounts;
+----------------+
| TotalSocieties |
+----------------+
|              1 |
+----------------+
1 row in set (0.00 sec)
```

Q26. Add column Mobile number in student table with default value ‘9999999999

Query:

```
ALTER TABLE STUDENT
ADD MobileNumber VARCHAR(10) DEFAULT '9999999999';
```

Output:

```
mysql> ALTER TABLE STUDENT
    -> ADD MobileNumber VARCHAR(10) DEFAULT '9999999999';
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> select * from student;
+--------+-------------+------------+------------+--------------+
| RollNo | StudentName | Course     | DOB        | MobileNumber |
+--------+-------------+------------+------------+--------------+
| 1413   | Chanchal    | Bsc(H)_CS  | 2003-08-03 | 9999999999   |
| 1433   | Ravikant    | Bsc(H)CS   | 2005-01-22 | 9999999999   |
| 1455   | Munender    | Bsc(H)CS   | 2003-10-02 | 9999999999   |
| 1468   | Mayank      | Bsc(H)CS   | 2003-06-23 | 9999999999   |
| 1478   | Sushant     | Bsc(H)CS   | 2003-07-19 | 9999999999   |
| 1901   | Luffy       | BA(H)Hindi | 2003-04-16 | 9999999999   |
| 2002   | Zoro        | BA(H)Hindi | 2003-02-12 | 9999999999   |
| 2103   | Usopp       | BA(H)Hindi | 2003-05-11 | 9999999999   |
| 2204   | Sanji       | BA(H)Hindi | 2004-01-21 | 9999999999   |
| 2305   | Nami        | BA(H)Hindi | 2004-11-25 | 9999999999   |
| 2507   | Robin       | BA(H)Hindi | 2005-09-09 | 9999999999   |
| 5895   | Suraj       | Bsc(H)Math | 2003-04-16 | 9999999999   |
| 624694 | Priya       | BA(HONS)   | 2001-07-12 | 9999999999   |
| 624699 | Jyoti       | BA(HONS)   | 2001-12-11 | 9999999999   |
| x1234  | Rahul       | BSC(H)CHEM | 2001-10-28 | 9999999999   |
| x1235  | Anamika     | BSC(H)CHEM | 2007-11-22 | 9999999999   |
| x1238  | Aditya      | BSC(H)CHEM | 2001-11-15 | 9999999999   |
| x1249  | Nishant     | BSC(H)CS   | 2001-03-17 | 9999999999   |
| x2345  | Aman        | BSC(H)CHEM | 2001-01-23 | 9999999999   |
| x2348  | Mohit       | BSC(H)CHEM | 2002-01-20 | 9999999999   |
| x2349  | Harsh       | BSC(H)CHEM | 2002-08-21 | 9999999999   |
| x2350  | Ashish      | BSC(H)CHEM | 2001-08-11 | 9999999999   |
| x2351  | Sandeep     | BSC(H)CHEM | 2001-09-15 | 9999999999   |
| x4569  | Sanjay      | BSC(H)CS   | 2001-05-23 | 9999999999   |
| x4599  | Ritik       | BSC(H)MATH | 2002-03-19 | 9999999999   |
| x4619  | Ritesh      | BSC(H)MATH | 2003-04-11 | 9999999999   |
| x4629  | Vijay       | BSC(H)MATH | 2002-06-13 | 9999999999   |
| y1239  | Varun       | BSC(H)CS   | 2001-11-21 | 9999999999   |
| y1240  | Arun        | BSC(H)CS   | 2001-01-01 | 9999999999   |
| y1241  | Mishti      | BSC(H)CS   | 2001-02-05 | 9999999999   |
| y1242  | Ankit       | BSC(H)CS   | 2001-03-07 | 9999999999   |
| y1243  | Kavita      | BSC(H)CS   | 2001-06-02 | 9999999999   |
| y1244  | Kiran       | BSC(H)CS   | 2002-07-12 | 9999999999   |
| y1245  | Karan       | BSC(H)CS   | 2003-08-22 | 9999999999   |
| y1246  | Suraj       | BSC(H)CS   | 2001-09-14 | 9999999999   |
| y1247  | Sachin      | BSC(H)CS   | 2001-05-12 | 9999999999   |
| z1249  | Nitin       | BSC(H)CS   | 2001-02-12 | 9999999999   |
| z4579  | Arjun       | BSC(H)CS   | 2001-06-14 | 9999999999   |
| z4589  | Himanshu    | BSC(H)CS   | 2003-04-17 | 9999999999   |
| z4639  | Vishal      | BSC(H)MATH | 2001-10-24 | 9999999999   |
| z4649  | Vikram      | BSC(H)MATH | 2001-11-26 | 9999999999   |
| z4659  | Divya       | BSC(H)MATH | 2010-11-26 | 9999999999   |
| z4669  | Tanu        | BSC(H)MATH | 2011-12-22 | 9999999999   |
| z4679  | Neha        | BSC(H)MATH | 2006-10-22 | 9999999999   |
| zX4669 | Nilam       | BA(HONS)   | 2004-03-21 | 9999999999   |
| zX4679 | Muskan      | BA(HONS)   | 2002-01-12 | 9999999999   |
| zX4689 | Sheetal     | BA(HONS)   | 2001-02-11 | 9999999999   |
| zX4699 | Riya        | BA(HONS)   | 2001-02-10 | 9999999999   |
| zz4669 | Annu        | BA(HONS)   | 2003-11-22 | 9999999999   |
+--------+-------------+------------+------------+--------------+
49 rows in set (0.00 sec)
```

Q27. Find the total number of students whose age is > 20 years.

Query:

```
SELECT COUNT(*) AS TotalStudentsGreater20
FROM STUDENT
WHERE YEAR(CURDATE()) - YEAR(DOB) > 20;
```

Output:

```
mysql> SELECT COUNT(*) AS TotalStudentsGreater20
    -> FROM STUDENT
    -> WHERE YEAR(CURDATE()) - YEAR(DOB) > 20;
+------------------------+
| TotalStudentsGreater20 |
+------------------------+
|                     40 |
+------------------------+
1 row in set (0.00 sec)
```

Q28. Find names of students who are born in 2001 and are enrolled in at least one society

Query:

```
SELECT StudentName
FROM STUDENT s
INNER JOIN ENROLLMENT e ON s.RollNo = e.RollNo
WHERE YEAR(DOB) = 2001;
```

Output:

```
mysql> SELECT StudentName
    -> FROM STUDENT s
    -> INNER JOIN ENROLLMENT e ON s.RollNo = e.RollNo
    -> WHERE YEAR(DOB) = 2001;
+-------------+
| StudentName |
+-------------+
| Vishal      |
| Nitin       |
| Mishti      |
| Ashish      |
| Varun       |
| Arun        |
+-------------+
6 rows in set (0.00 sec)
```

Q29. Count all societies whose name starts with ‘S’ and ends with ‘t’ and at least 5 students are enrolled in the society.

Query:

```
SELECT COUNT(*) AS TotalMatchingSocieties
FROM SOCIETY s
INNER JOIN ENROLLMENT e ON s.SocID = e.SID
WHERE SocName LIKE 'S%' AND SocName LIKE '%t'
GROUP BY SocName
HAVING COUNT(*) >= 5;
```

Output:

```
mysql> SELECT COUNT(*) AS TotalMatchingSocieties
    -> FROM SOCIETY s
    -> INNER JOIN ENROLLMENT e ON s.SocID = e.SID
    -> WHERE SocName LIKE 'S%' AND SocName LIKE '%t'
    -> GROUP BY SocName
    -> HAVING COUNT(*) >= 5;
Empty set (0.00 sec)
```

Q30. Display the following information:

Society name, Mentor name, Total Capacity, Total Enrolled ,Unfilled Seats

Query:

SELECT SocName, MentorName, MAX(TotalSeats) AS TotalSeats, COUNT(*) AS EnrolledStudents, MAX(TotalSeats) - COUNT(*) AS VacantSeats

FROM SOCIETY s

LEFT JOIN ENROLLMENT e ON s.SocID = e.SID

GROUP BY SocName, MentorName;

Output:

```
mysql> SELECT SocName, MentorName, MAX(TotalSeats) AS TotalSeats, COUNT(*) AS EnrolledStudents, MAX(TotalSeats) - COUNT(*) AS VacantSeats
    -> FROM SOCIETY s
    -> LEFT JOIN ENROLLMENT e ON s.SocID = e.SID
    -> GROUP BY SocName, MentorName;
+----------+---------------+------------+------------------+-------------+
| SocName  | MentorName    | TotalSeats | EnrolledStudents | VacantSeats |
+----------+---------------+------------+------------------+-------------+
| Dancing  | Rohit Sharma  |         36 |                4 |          32 |
| Debating | Rajat Dalal   |         19 |                8 |          11 |
| NSS      | Akhilesh Gupta|         17 |                5 |          12 |
| Sashakt  | Dr Ankit      |         53 |                4 |          49 |
| YUVA     | Rajesh Singh  |         25 |                3 |          22 |
+----------+---------------+------------+------------------+-------------+
```

II. Do the following database administration commands: create user, create role, grant

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

Output:

```
mysql> create user 'sahil'@'localhost' identified by 'sahil1234';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> CREATE ROLE editor;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> GRANT SELECT, INSERT, UPDATE ON new.student TO editor;
Query OK, 0 rows affected (0.04 sec)

mysql> REVOKE SELECT, INSERT, UPDATE ON new.student FROM editor;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> CREATE INDEX idx_student_name ON student(StudentName);
Query OK, 0 rows affected, 1 warning (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 1
```

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

CODE:

C: > Users > sahil > Downloads > ODBC.py > ...

Click here to ask Blackbox to help you code faster

```python
import pyodbc

# Connect to the MySQL database using ODBC
conn = pyodbc.connect('DSN=sahil_dbms;UID=root;PWD=sahil')

# Create a cursor object to execute queries
cursor = conn.cursor()

# 1. Retrieve names of students enrolled in any society
query_1 = "SELECT s.StudentName FROM student s INNER JOIN ENROLLMENT e ON s.RollNo = e.RollNo;"
cursor.execute(query_1)
print("\nQuery 1 Results:")
for row in cursor.fetchall():
    print(row[0])

# 2. Retrieve all society names
query_2 = "SELECT SocName FROM SOCIETY;"
cursor.execute(query_2)
print("\nQuery 2 Results:")
for row in cursor.fetchall():
    print(row[0])

# 3. Retrieve students' names starting with letter ‘A’
query_3 = "SELECT StudentName FROM student WHERE StudentName LIKE 'A%';"
cursor.execute(query_3)
print("\nQuery 3 Results:")
for row in cursor.fetchall():
    print(row[0])

# 4. Retrieve students' details studying in courses ‘computer science’ or ‘chemistry’
query_4 = "SELECT * FROM student WHERE Course IN ('computer science', 'chemistry');"
cursor.execute(query_4)
print("\nQuery 4 Results:")
for row in cursor.fetchall():
    print(row)

# 5. Retrieve students’ names whose roll no either starts with ‘X’ or ‘Z’ and ends with ‘9’
query_5 = "SELECT * FROM student WHERE (RollNo LIKE 'X%' OR RollNo LIKE 'Z%') AND RollNo LIKE '%9';"
cursor.execute(query_5)
print("\nQuery 5 Results:")
for row in cursor.fetchall():
    print(row[0])

# 6. Find society details with more than N TotalSeats where N is to be input by the user
N = input("Enter the value of N: ")
query_6 = f"SELECT * FROM SOCIETY WHERE TotalSeats > {N};"
cursor.execute(query_6)
print("\nQuery 6 Results:")
for row in cursor.fetchall():
    print(row)

# 7. Update society table for mentor name of a specific society
MentorName = input("Enter the new mentor name: ")
SocID = input("Enter the Society ID: ")
query_7 = f"UPDATE SOCIETY SET MentorName = '{MentorName}' WHERE SocID = '{SocID}';"
cursor.execute(query_7)
```

```python
conn.commit()

# Updated record
query_select = f"SELECT * FROM SOCIETY WHERE SocID = '{SocID}';"
cursor.execute(query_select)
updated_record = cursor.fetchone()
print("\nUpdated Record:")
print(updated_record)

# 8. Find society names in which more than five students have enrolled
query_8 = "SELECT SocName FROM SOCIETY s INNER JOIN ENROLLMENT e ON s.SocID = e.SID GROUP BY SocName HAVING COUNT(*) > 5;"
cursor.execute(query_8)
print("\nQuery 8 Results:")
for row in cursor.fetchall():
    print(row[0])

# 9. Find the name of youngest student enrolled in society 'NSS'
query_9 = "SELECT StudentName FROM student WHERE RollNo IN (SELECT RollNo FROM ENROLLMENT WHERE SID = 'NSS') ORDER BY DOB ASC LIMIT 1;"
cursor.execute(query_9)
print("\nQuery 9 Results:")
for row in cursor.fetchall():
    print(row[0])

# 10. Find the name of most popular society (on the basis of enrolled students)
query_10 = "SELECT SocName FROM SOCIETY s INNER JOIN ENROLLMENT e ON s.SocID = e.SID GROUP BY SocName ORDER BY COUNT(*) DESC LIMIT 1;"
cursor.execute(query_10)
print("\nQuery 10 Results:")
for row in cursor.fetchall():
    print(row[0])

# 11. Find the name of two least popular societies (on the basis of enrolled students)
query_11 = "SELECT SocName FROM SOCIETY s INNER JOIN ENROLLMENT e ON s.SocID = e.SID GROUP BY SocName ORDER BY COUNT(*) ASC LIMIT 2;"
cursor.execute(query_11)
print("\nQuery 11 Results:")
for row in cursor.fetchall():
    print(row[0])

# 12. Find the student names who are not enrolled in any society
query_12 = "SELECT StudentName FROM student WHERE RollNo NOT IN (SELECT RollNo FROM ENROLLMENT);"
cursor.execute(query_12)
print("\nQuery 12 Results:")
for row in cursor.fetchall():
    print(row[0])

# 13. Find the student names enrolled in at least two societies
query_13 = "SELECT RollNo FROM ENROLLMENT GROUP BY RollNo HAVING COUNT(*) >= 2;"
cursor.execute(query_13)
print("\nQuery 13 Results:")
for row in cursor.fetchall():
    print(row[0])

# 14. Find society names in which maximum students are enrolled
query_14 = "SELECT SocName FROM SOCIETY s INNER JOIN ENROLLMENT e ON s.SocID = e.SID GROUP BY SocName ORDER BY COUNT(*) DESC LIMIT 1;"
cursor.execute(query_14)
print("\nQuery 14 Results:")
for row in cursor.fetchall():
    print(row[0])
```

```python

    # 15. Find names of all students who have enrolled in any society and society names in which at least one student has enrolled
    query_15 = "SELECT StudentName FROM student WHERE RollNo IN (SELECT RollNo FROM ENROLLMENT);"
    cursor.execute(query_15)
    print("\nQuery 15 Results:")
    for row in cursor.fetchall():
        print(row[0])

    # 16. Find names of students who are enrolled in any of the three societies ‘Debating’, ‘Dancing’ and ‘Sashakt’.
    query_16 = "SELECT StudentName FROM student WHERE RollNo IN (SELECT RollNo FROM ENROLLMENT WHERE SID IN ('Debating', 'Dancing', 'Sashakt'));"
    cursor.execute(query_16)
    print("\nQuery 16 Results:")
    for row in cursor.fetchall():
        print(row[0])

    # 17. Find society names such that its mentor has a name with ‘Gupta’ in it.
    query_17 = "SELECT SocName FROM SOCIETY WHERE MentorName LIKE '%Gupta%';"
    cursor.execute(query_17)
    print("\nQuery 17 Results:")
    for row in cursor.fetchall():
        print(row[0])
    # 18. Find the society names in which the number of enrolled students is only 10% of its capacity.
    query_18 = "SELECT SocName FROM SOCIETY s INNER JOIN ENROLLMENT e ON s.SocID = e.SID GROUP BY SocName HAVING COUNT(*) = 0.1 * MAX(TotalSeats);"
    cursor.execute(query_18)
    print("\nQuery 18 Results:")
    for row in cursor.fetchall():
        print(row[0])
    # 19. Display the vacant seats for each society.
    query_19 = "SELECT s.SocName, (s.TotalSeats - COUNT(e.RollNo)) AS VacantSeats FROM SOCIETY s LEFT JOIN ENROLLMENT e ON s.SocID = e.SID GROUP BY s.SocName, s.TotalSeats;"
    cursor.execute(query_19)
    print("\nQuery 19 Results:")
    for row in cursor.fetchall():
        print(row[0], row[1])

    # 20. Increment Total Seats of each society by 10%
    query_20 = "UPDATE SOCIETY SET TotalSeats = TotalSeats + CEILING(0.1 * TotalSeats);"
    cursor.execute(query_20)
    print("\nQuery 20 Results: Updated total seats by 10% successfully")

    # 21. Add the enrollment fees paid ('Yes'/'No') field in the enrollment table.
    query_21 = "ALTER TABLE ENROLLMENT ADD COLUMN EnrollmentFee VARCHAR(3);"
    cursor.execute(query_21)
    print("\nQuery 21 Results: Enrollment fees paid field added successfully")

    # 22. Update date of enrollment of society id 's1' to '2018-01-15', 's2' to current date and 's3' to '2018-01-02'.
    # Update date of enrollment for society id 's1' to '2018-01-15'
    query_22_s1 = "UPDATE ENROLLMENT SET DateOfEnrollment = '2018-01-15' WHERE SID = 'ABC110';"
    cursor.execute(query_22_s1)

    # Update date of enrollment for society id 's2' to current date
    query_22_s2 = "UPDATE ENROLLMENT SET DateOfEnrollment = CURDATE() WHERE SID = 'ABC103';"
    cursor.execute(query_22_s2)

    # Update date of enrollment for society id 's3' to '2018-01-02'
    query_22_s3 = "UPDATE ENROLLMENT SET DateOfEnrollment = '2018-01-02' WHERE SID = 'ABC106';"
    cursor.execute(query_22_s3)
    conn.commit()
```

```python

    print("\nQuery 22 Results: Date of enrollment updated successfully")


    # 23. Create a view to keep track of society names with the total number of students enrolled in it.
    query_23 = "CREATE VIEW SocietyEnrollmentCount 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;"
    cursor.execute(query_23)
    print("\nQuery 23 Results: Society enrollment count view created successfully")

    # 24. Find student names enrolled in all the societies.
    query_24 = "SELECT RollNo FROM (SELECT RollNo, COUNT(DISTINCT SID) AS num_societies FROM ENROLLMENT GROUP BY RollNo) AS temp WHERE num_societies = (SELECT COUNT(DISTINCT SocID) FROM SOCIETY);"
    cursor.execute(query_24)
    print("\nQuery 24 Results:")
    for row in cursor.fetchall():
        print(row[0])

    # 25. Count the number of societies with more than 5 students enrolled in it
    query_25 = "SELECT COUNT(*) FROM (SELECT DISTINCT SID FROM ENROLLMENT GROUP BY SID HAVING COUNT(*) > 5) AS tmp;"
    cursor.execute(query_25)
    print("\nQuery 25 Results:")
    for row in cursor.fetchall():
        print(row[0])

    # 26. Add column Mobile number in student table with default value '9999999999'
    query_26 = "ALTER TABLE student ADD COLUMN MobileNumber VARCHAR(10) DEFAULT '9999999999';"
    cursor.execute(query_26)
    print("\nQuery 26 Results: Mobile number column added to student table successfully")

    # 27. Find the total number of students whose age is > 20 years.
    query_27 = "SELECT COUNT(*) FROM student WHERE DATEDIFF(CURRENT_DATE(), DOB) / 365.25 > 20;"
    cursor.execute(query_27)
    print("\nQuery 27 Results:")
    for row in cursor.fetchall():
        print(row[0])

    # 28. Find names of students who are born in 2001 and are enrolled in at least one society.
    query_28 = "SELECT StudentName FROM student WHERE YEAR(DOB) = 2001 AND RollNo IN (SELECT RollNo FROM ENROLLMENT);"
    cursor.execute(query_28)
    print("\nQuery 28 Results:")
    for row in cursor.fetchall():
        print(row[0])

    # 29. Count all societies whose name starts with 'S' and ends with 't' and at least 5 students are enrolled in the society.
    query_29 = "SELECT COUNT(*) FROM (SELECT SocName FROM SOCIETY WHERE SocName LIKE 'S%t' GROUP BY SocName HAVING COUNT(*) >= 5) AS tmp;"
    cursor.execute(query_29)
    print("\nQuery 29 Results:")
    for row in cursor.fetchall():
        print(row[0])

    # 30. Display the following information: Society name, Mentor name, Total Capacity, Total Enrolled, Unfilled Seats
    query_30 = "SELECT s.SocName, s.MentorName,s.TotalSeats,COUNT(e.RollNo) AS TotalEnrolled, s.TotalSeats - COUNT(e.RollNo) AS UnfilledSeats from society s LEFT join enrollment e ON s.SocID = e.SID GROUP BY s.SocName, s.MentorName, s.TotalSeats;"
    cursor.execute(query_30)
    print("\nQuery 30 Results:")
    for row in cursor.fetchall():
        print(row)

    # Close the cursor and connection
```

```
228     cursor.close()
229     conn.close()
230
```

Output:

```
Query 1 Results:
Munender
Sushant
Ritik
Vishal
Tanu
Ravikant
Nitin
Neha
Suraj
Ritesh
Mishti
Nilam
Chanchal
Ashish
Divya
Muskan
Usopp
Sanji
Nami
Mohit
Vijay
Varun
Arun
Annu
Jyoti
Aditya
Rahul
Aman

Query 2 Results:
NSS
YUVA
Sashakt
Dancing
Debating
Jajba
Tedx
Josh
```

```
Query 3 Results:
Anamika
Aditya
Aman
Ashish
Arun
Ankit
Arjun
Annu

Query 4 Results:

Query 5 Results:
x1249
x2349
x4569
x4599
x4619
x4629
z1249
z4579
z4589
z4639
z4649
z4659
z4669
z4679
zX4669
zX4679
zX4689
zX4699
zz4669
Enter the value of N: 50

Query 6 Results:
('222', 'YUVA', 'Rahul Gandhi', 53)
('333', 'Sashakt', 'Dr Ankit', 107)
('444', 'Dancing', 'Rohit Sharma', 73)
Enter the new mentor name: Amit Saini
Enter the Society ID: 222

Updated Record:
('222', 'YUVA', 'Amit Saini', 53)

Query 8 Results:
Debating

Query 9 Results:

Query 10 Results:
Debating

Query 11 Results:
Tedx
Josh
```

```
Query 12 Results:
Mayank
Luffy
Zoro
Robin
Priya
Anamika
Nishant
Harsh
Sandeep
Sanjay
Ankit
Kavita
Kiran
Karan
Suraj
Sachin
Arjun
Himanshu
Vikram
Sheetal
Riya

Query 13 Results:

Query 14 Results:
Debating

Query 15 Results:
Chanchal
Ravikant
Munender
Sushant
Usopp
Sanji
Nami
Suraj
Jyoti
Rahul
Aditya
Aman
Mohit
Ashish
Ritik
Ritesh
Vijay
Varun
Arun
Mishti
Nitin
Vishal
Divya
Tanu
Neha
Nilam
Muskan
Annu
```

```
Query 16 Results:

Query 17 Results:
NSS

Query 18 Results:

Query 19 Results:
Dancing 69
Debating 33
Jajba 26
Josh 36
NSS 32
Sashakt 103
Tedx 47
YUVA 50

Query 20 Results: Updated total seats by 10% successfully

Query 21 Results: Enrollment fees paid field added successfully

Query 22 Results: Date of enrollment updated successfully

Query 23 Results: Society enrollment count view created successfully

Query 24 Results:

Query 25 Results:
1

Query 26 Results: Mobile number column added to student table successfully

Query 27 Results:
42

Query 28 Results:
Jyoti
Rahul
Aditya
Aman
Ashish
Varun
Arun
Mishti
Nitin
Vishal
```

```
Query 29 Results:
0

Query 30 Results:
('Dancing', 'Rohit Sharma', 81, 4, 77)
('Debating', 'Rajat Dalal', 46, 8, 38)
('Jajba', 'Sandeep Thakur', 31, 2, 29)
('Josh', 'Gaurav Kumar', 41, 1, 40)
('NSS', 'Akhilesh Gupta', 41, 5, 36)
('Sashakt', 'Dr Ankit', 118, 4, 114)
('Tedx', 'Nikhil Singh', 53, 1, 52)
('YUVA', 'Amit Saini', 59, 3, 56)
PS C:\Users\sahil>
```