**NAME – CHANDAN - KUMAR**

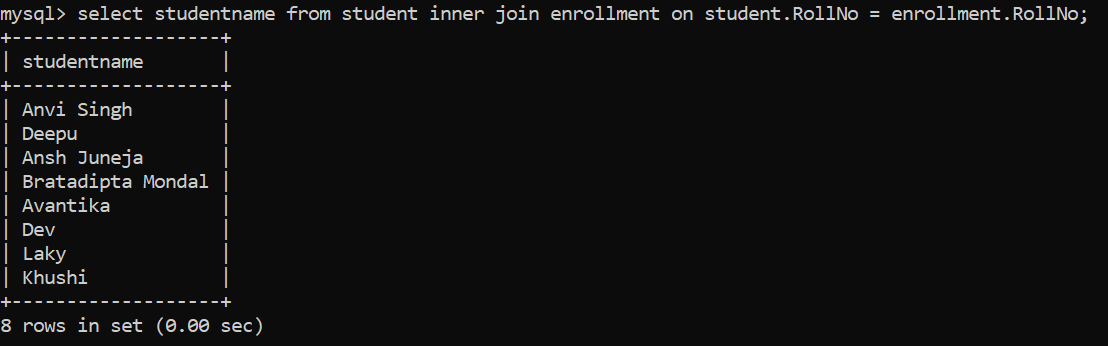
**ROLL-NO – 16022**

**COURSE –B.sc(Hons)Computer Science**

**SUB – D.B.M.S**

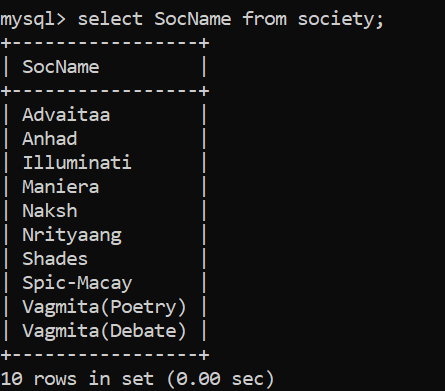
**PRACTICAL FILE : Subbmited to- Rakesh sir**

1. Create and use the following student-society database schema for a college to answer the given (sample) queries using the standalone SQL editor.
2. Retrieve names of students enrolled in any society.

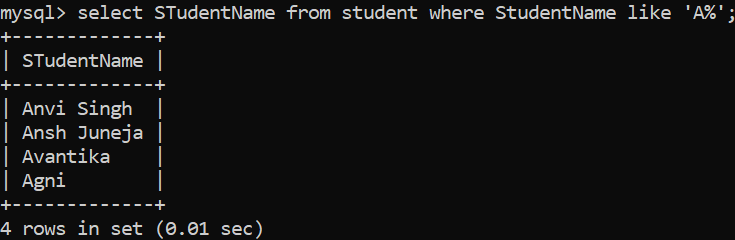


1. Retrieve all society names.

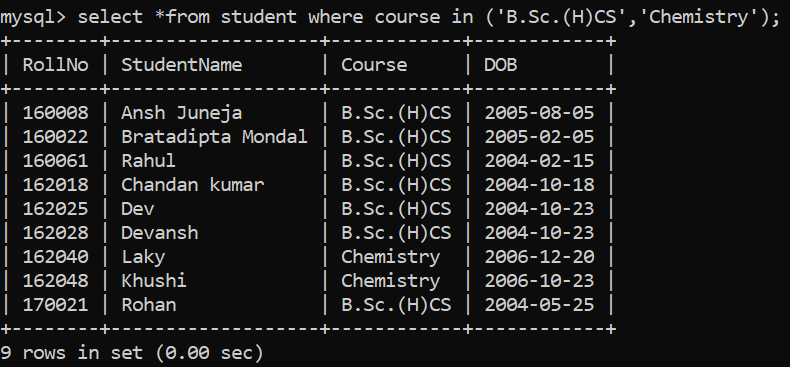
Ans:



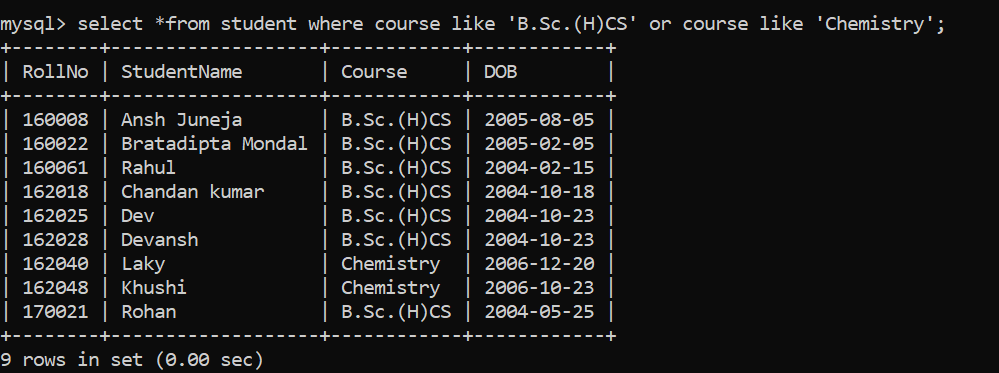
1. Retrieve students' names starting with letter ‘A’.



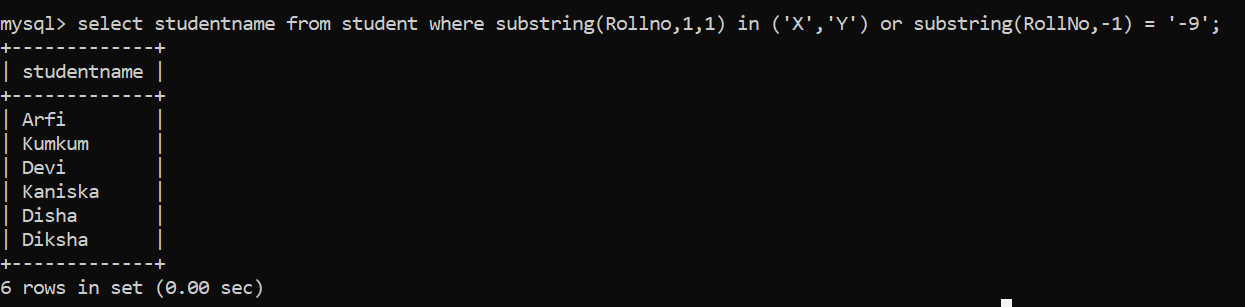
1. Retrieve students' details studying in courses ‘computer science’ or ‘chemistry’.

Ans: Method 1

Method 2

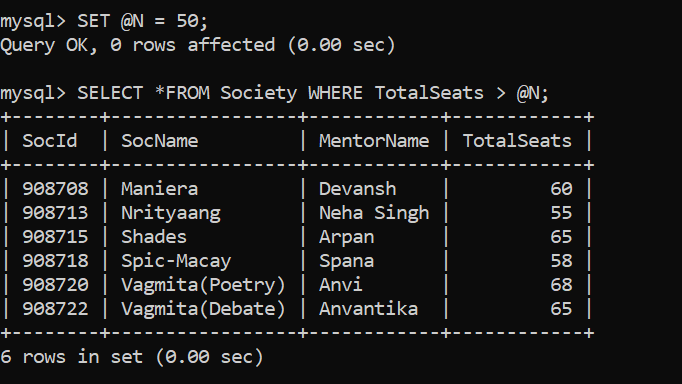


1. Retrieve students’ names whose roll no either starts with ‘X’ or ‘Z’ and ends with ‘9’.

Ans: 

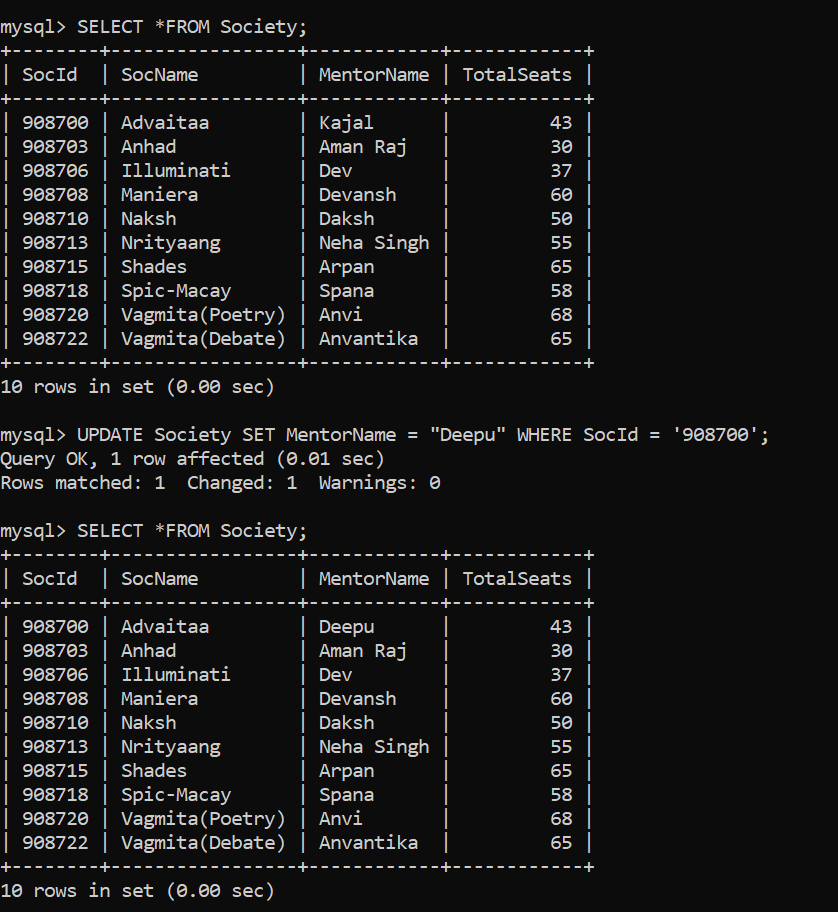
1. Find society details with more than N TotalSeats where N is to be input by the user.

Ans:

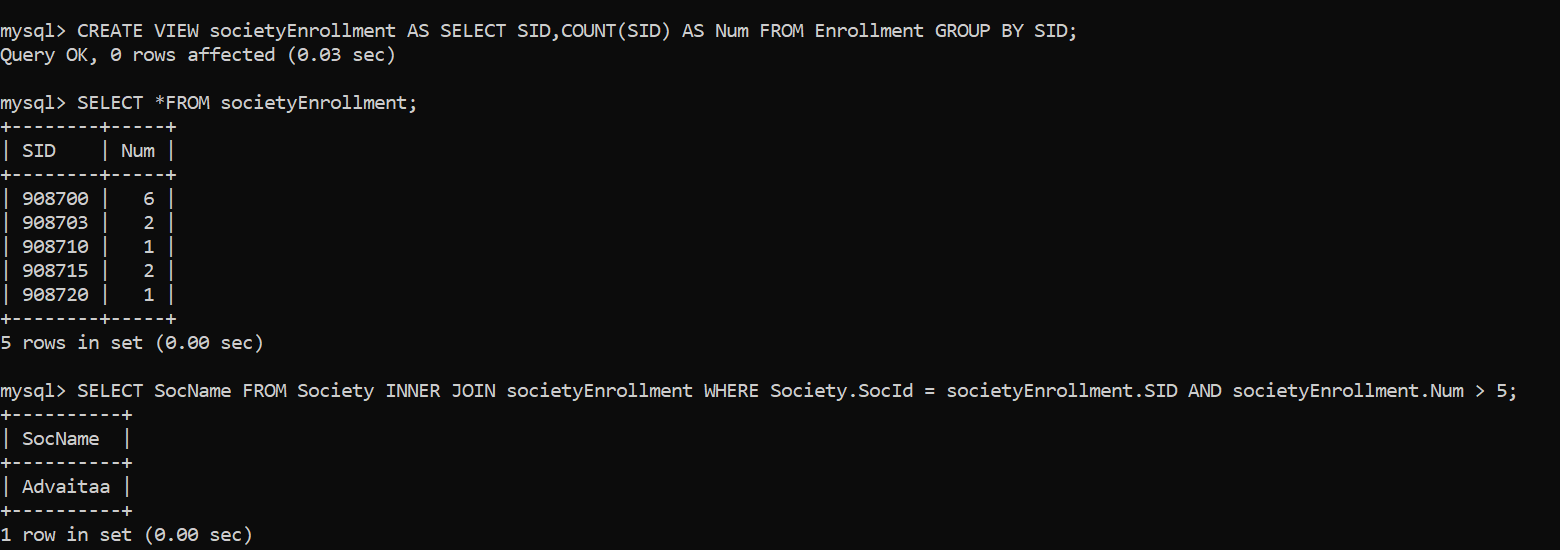


1. Update society table for mentor name of a specific society.

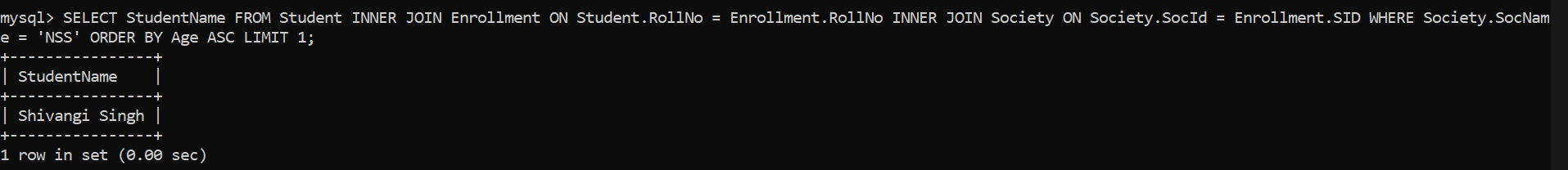
Ans:



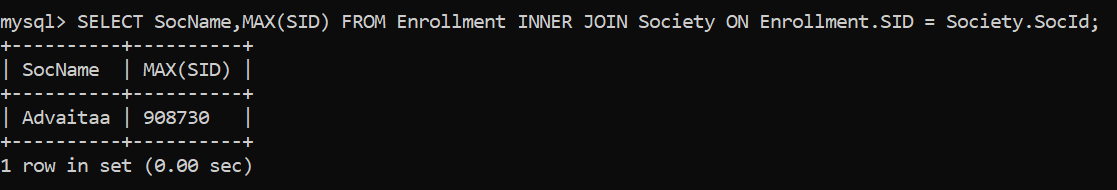
1. Find society names in which more than five students have enrolled.

Ans: 

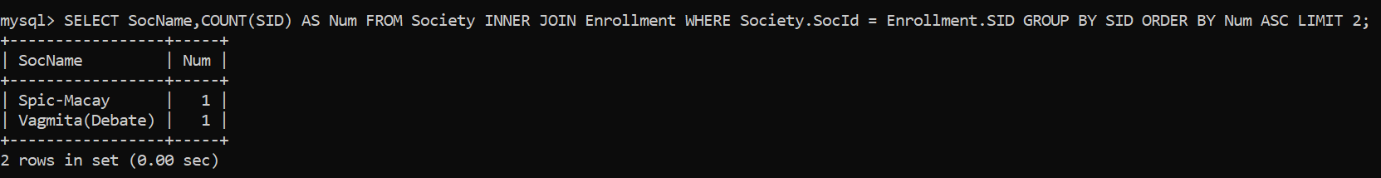
1. Find the name of youngest student enrolled in society ‘NSS’.

Ans: 

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

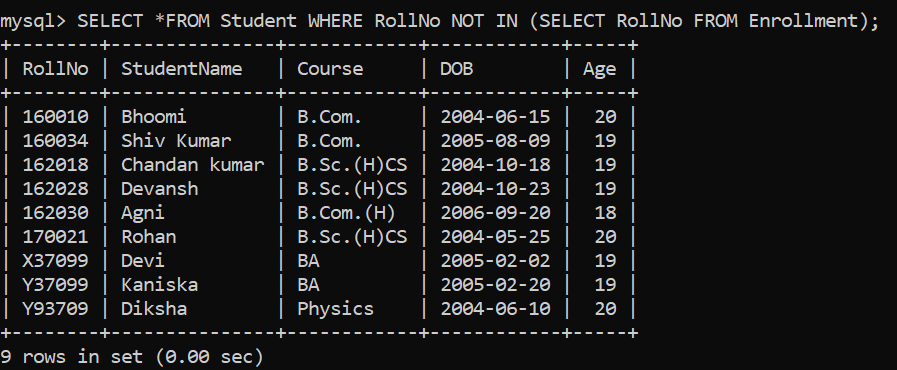
Ans: 

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

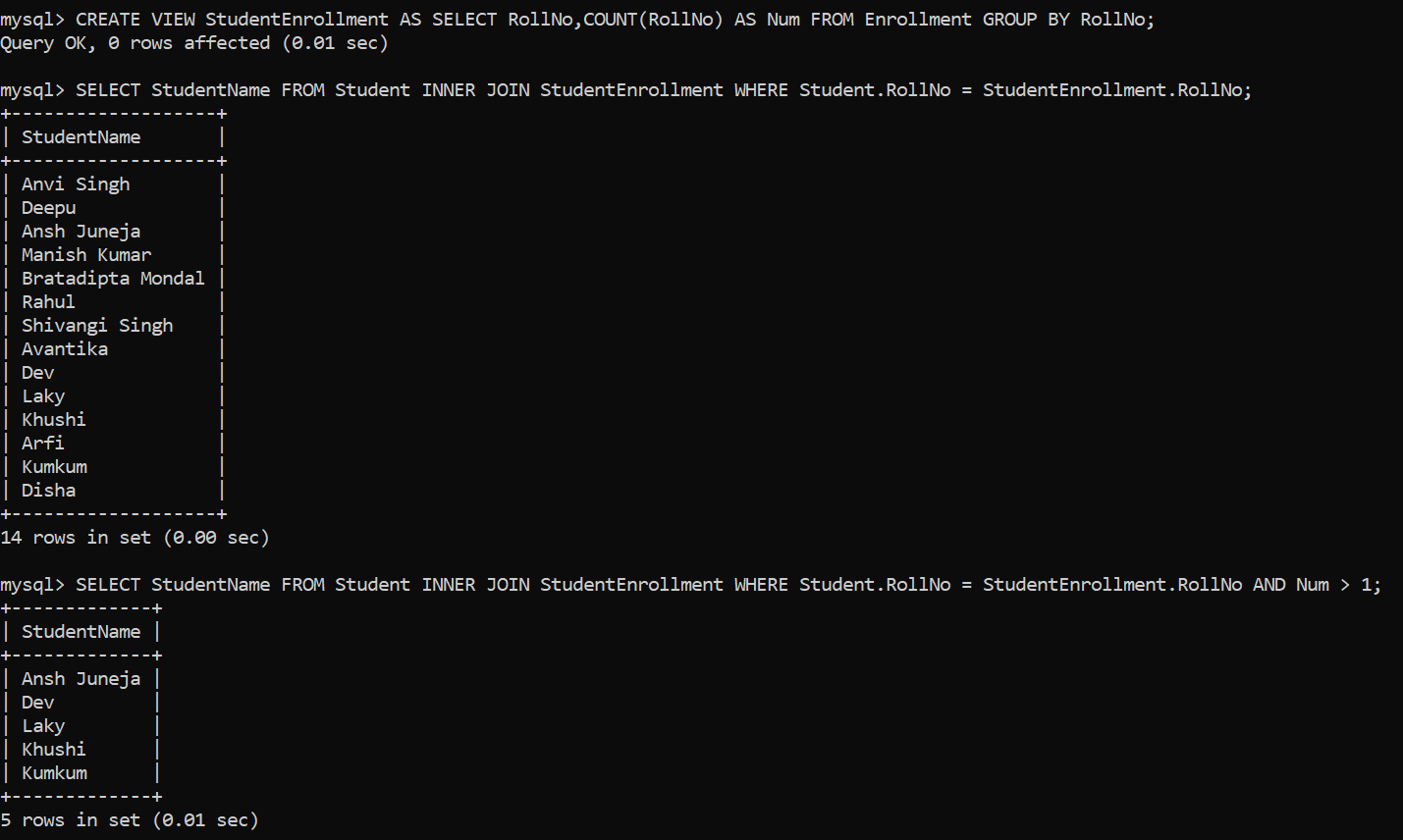
Ans: 

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

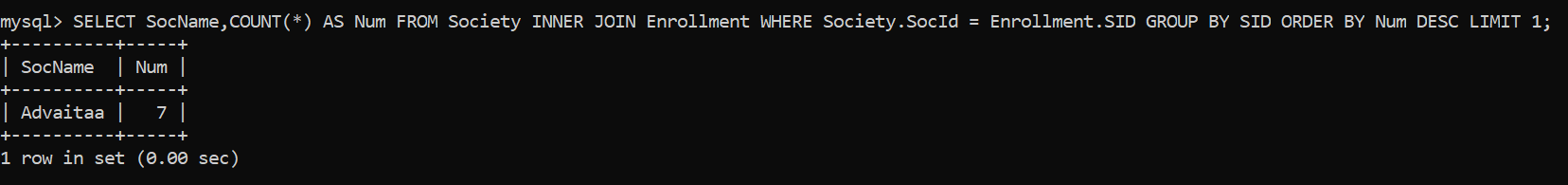
Ans:



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

Ans: 

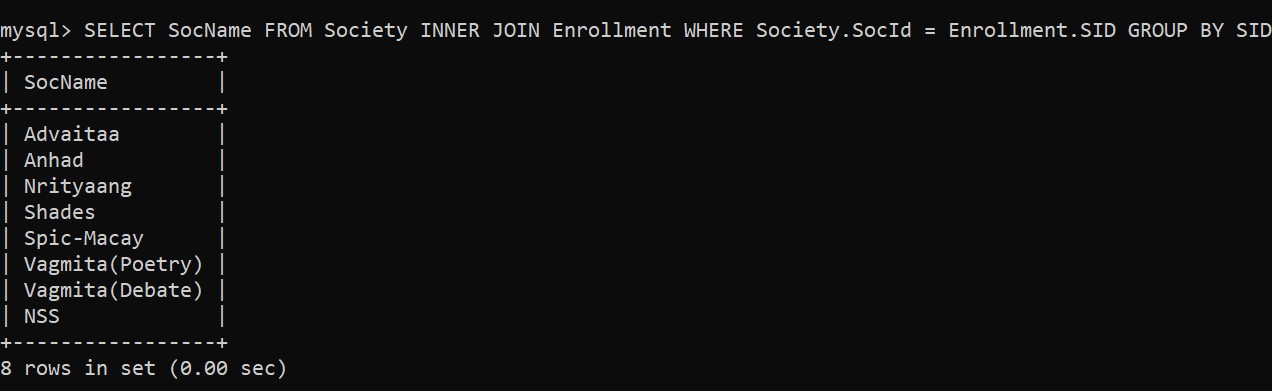
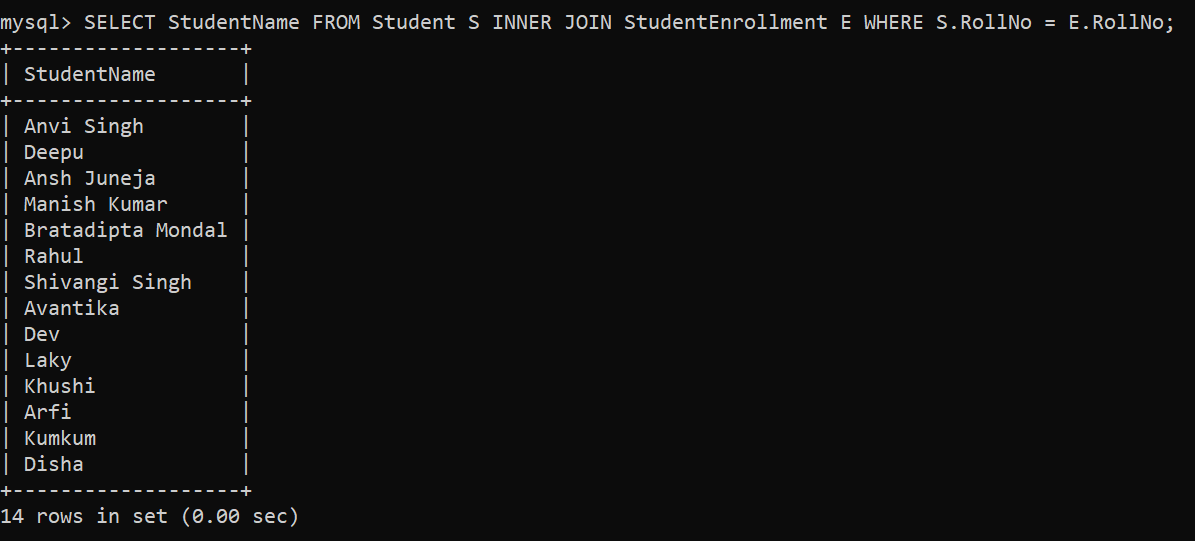
1. Find society names in which maximum students are enrolled.

Ans: 

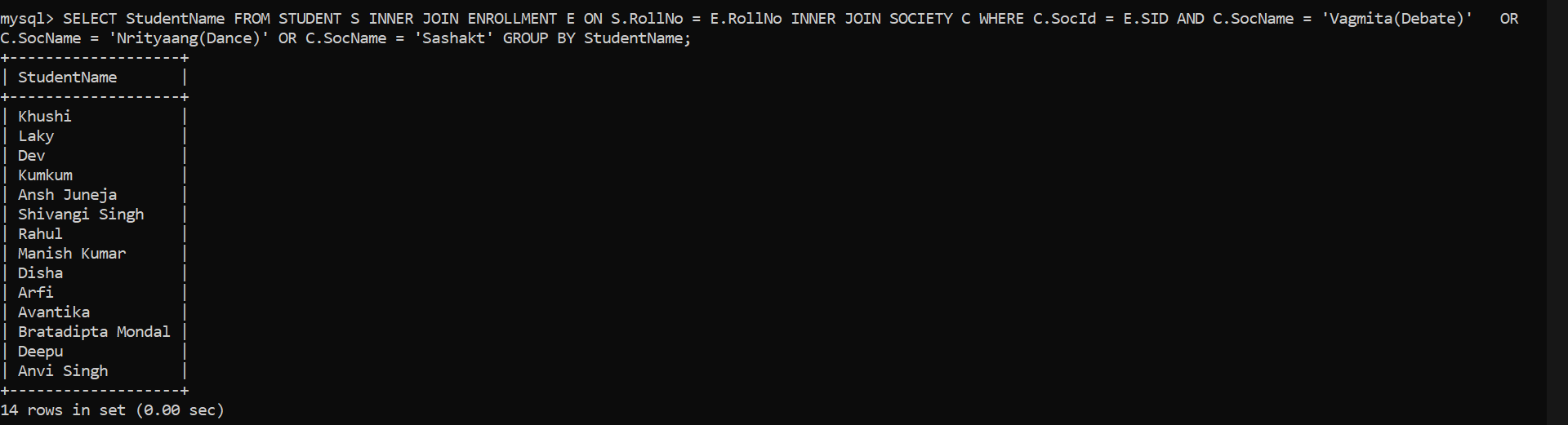
1. Find names of all students who have enrolled in any society and society names in which at least one student has enrolled.

Ans:

Note : StudentEnrollment is a view that’s record store number of societies are join by students.

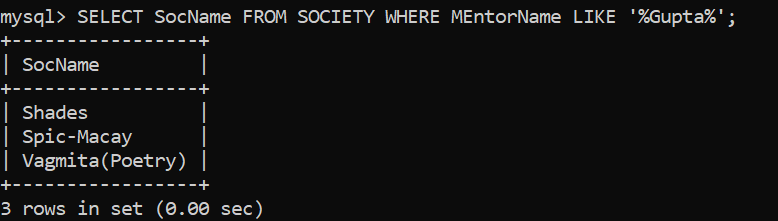


1. Find names of students who are enrolled in any of the three societies ‘Debating’, ‘Dancing’ and ‘Sashakt’.

Ans: 

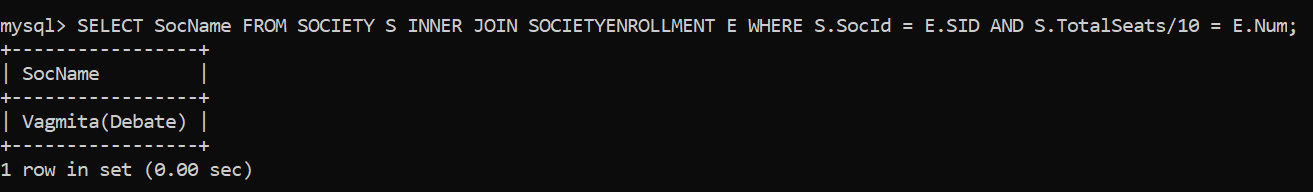
1. Find society names such that its mentor has a name with ‘Gupta’ in it.

Ans:

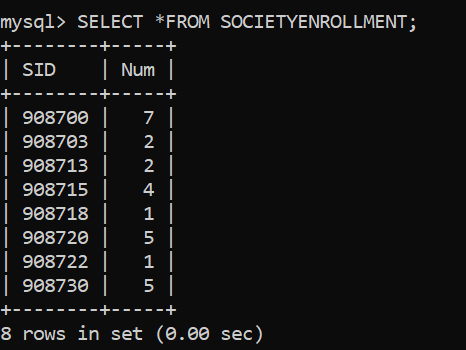


1. Find the society names in which the number of enrolled students is only 10% of its capacity.

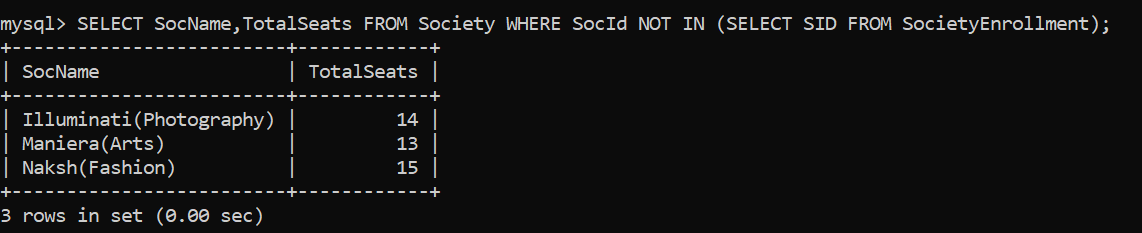
Ans:



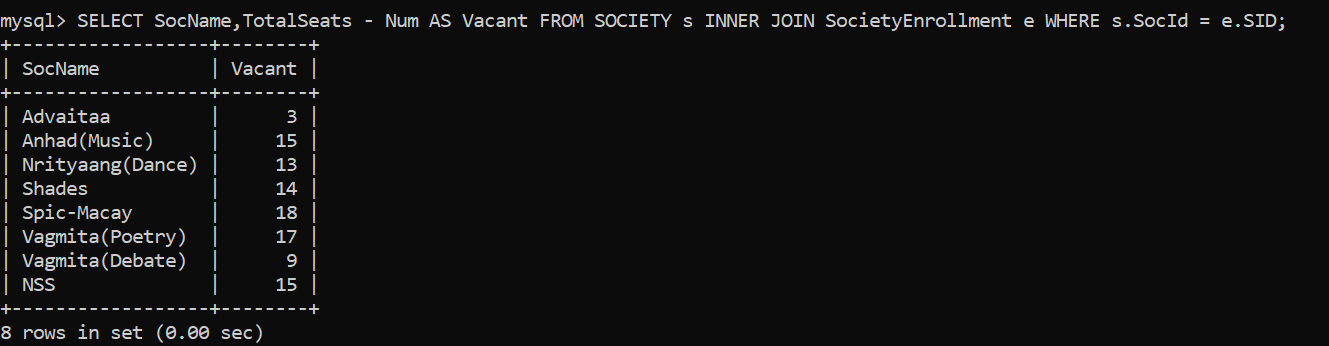
**NOTE : SocietyEnrollment Table store which number of students are belong a particular society.**



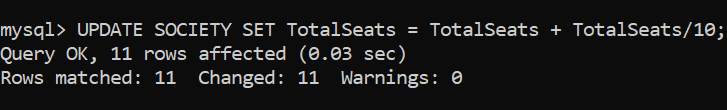
1. Display the vacant seats for each society.

Ans: 

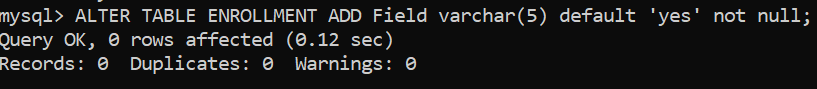
**NOTE : SocietyEnrollment Table store which number of students are belong a particular society.**



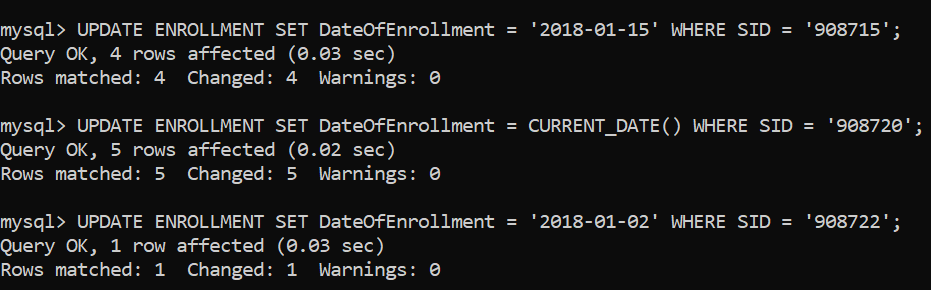
1. Increment Total Seats of each society by 10%.

Ans: 

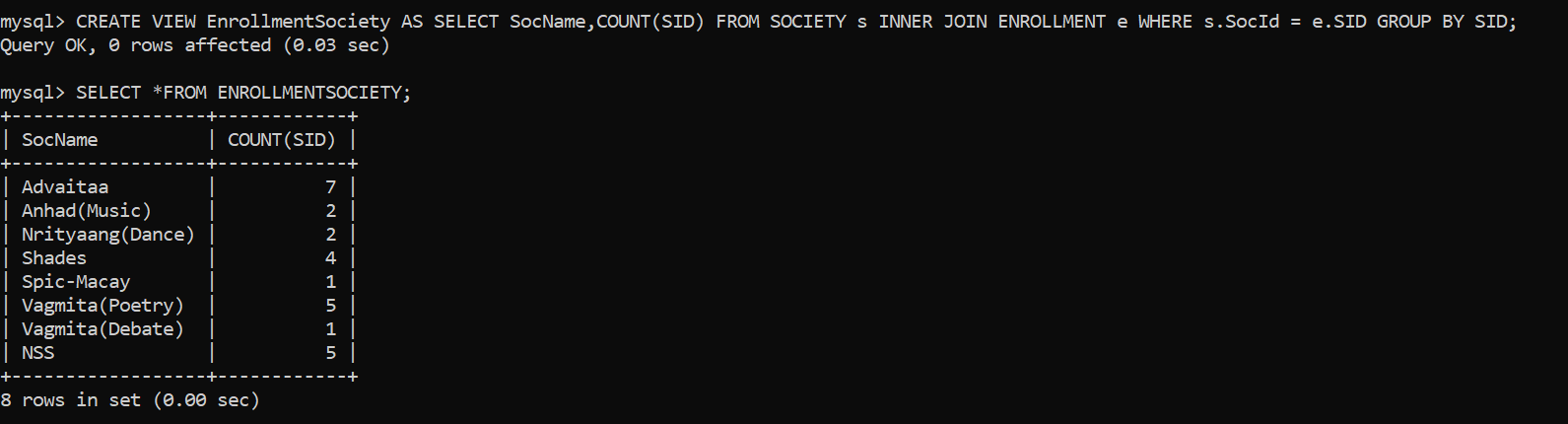
1. Add the enrollment fees paid (‘yes’/’No’) field in the enrollment table.

Ans: 

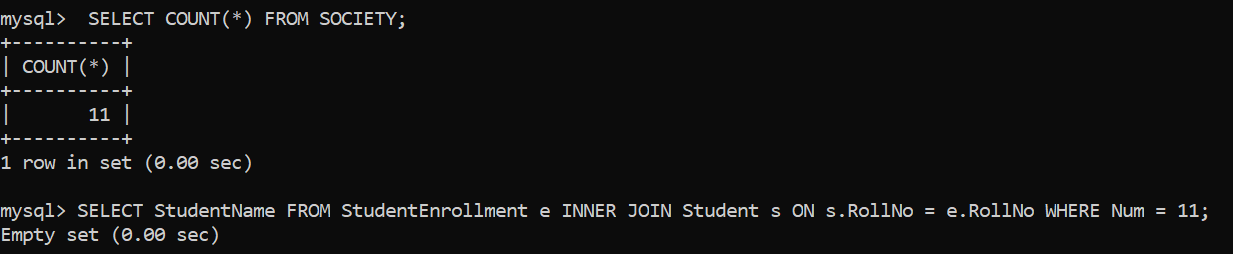
1. Update date of enrollment of society id ‘s1’ to ‘2018-01-15’, ‘s2’ to current date and ‘s3’ to ‘2018-01-02’.

Ans: 

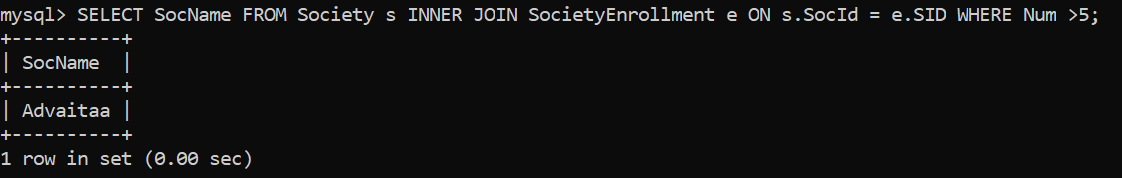
1. Create a view to keep track of society names with the total number of students enrolled in it.

Ans: 

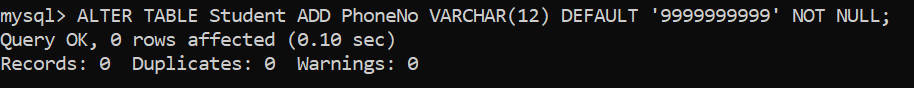
1. Find student names enrolled in all the societies.

Ans: 

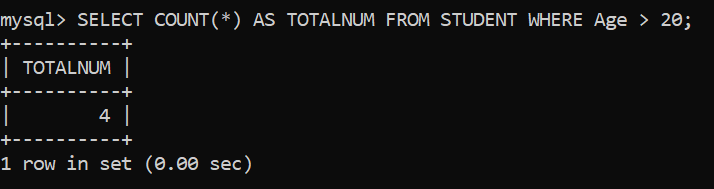
1. Count the number of societies with more than 5 students enrolled in it.

Ans: 

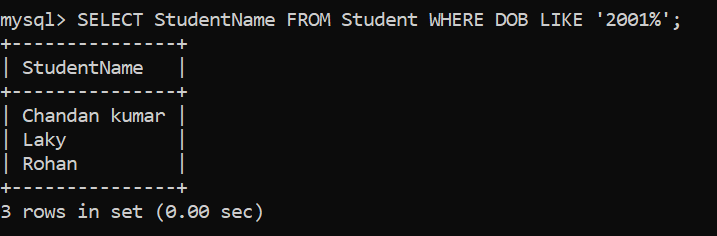
1. Add column Mobile number in student table with default value ‘9999999999’.

Ans : 

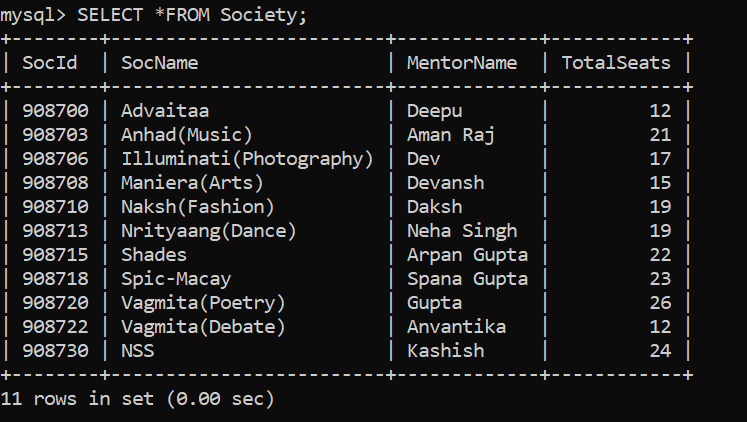
1. Find the total number of students whose age is > 20 years.

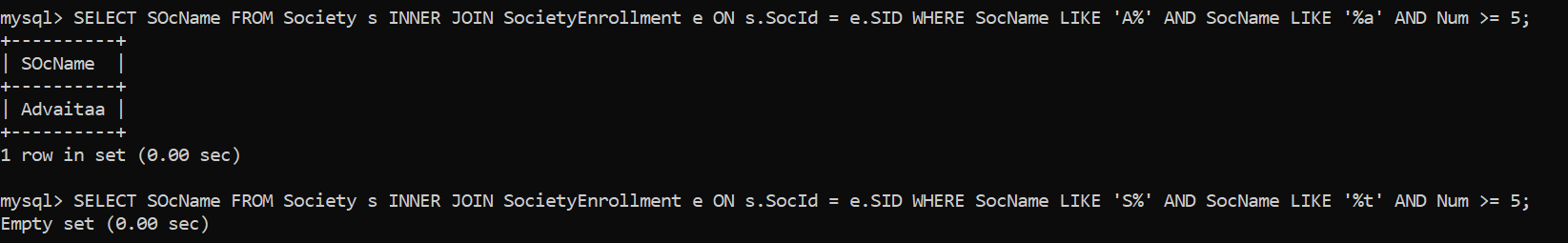
Ans: 

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

Ans: 

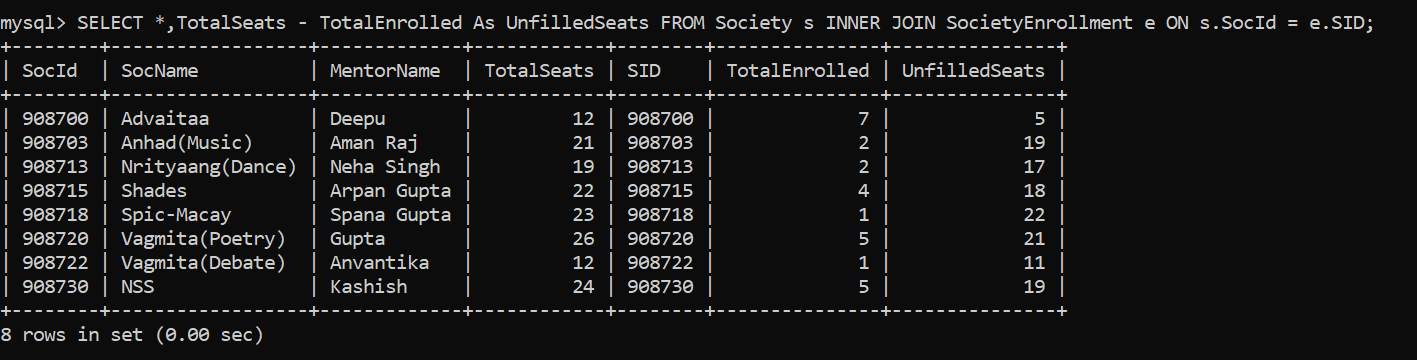
1. Count all societies whose name starts with ‘S’ and ends with ‘t’ and at least 5 students are enrolled in the society.

Ans: 



1. Display the following information:

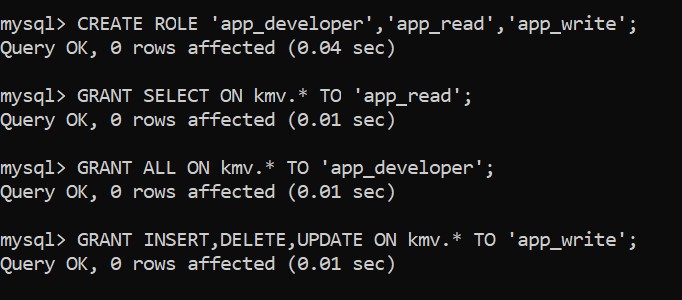
Society name Mentor name Total Capacity Total Enrolled Unfilled Seats

Ans: 

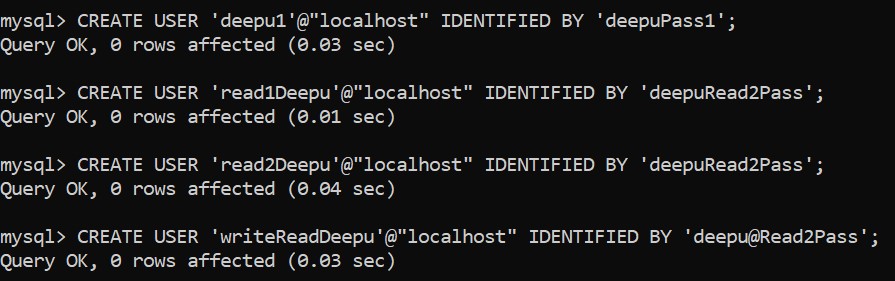
1. **Do the following database administration commands:**

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

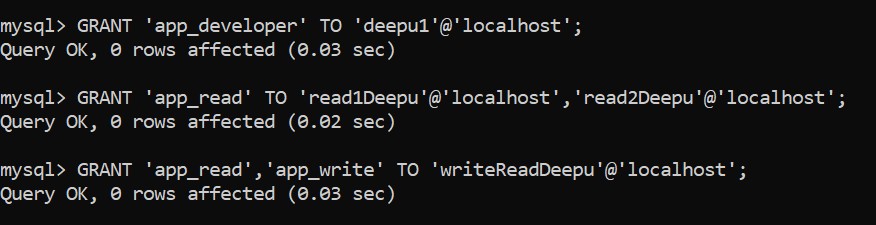
**Create Roles and grant privileges**



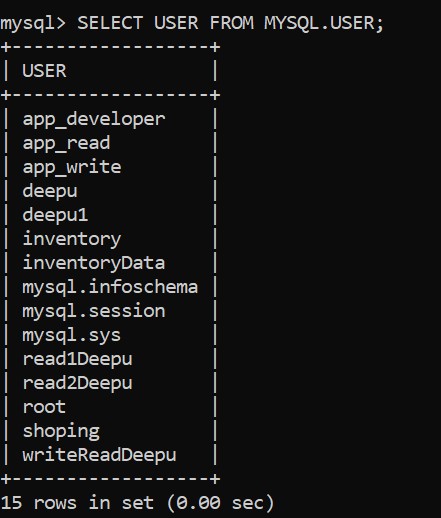
**Create Users**



**Role Provide to User**



**All Users**



**3.) Executec queries given in part 1 through a high level language ODBC connection .**

*import* mysql.connector

db= mysql.connector.connect(

*host* ="localhost",

*user* ="root",

*password*="9113"

)

cursor=db.cursor()

cursor.execute("use student\_society")

*# 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,(TotalSeats-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()

1. Create the Company database schems from Chapter 3 of book and Execute the queries of Chapter 7.

*--  Creating Database Company*

  CREATE DATABASE company;

*--  to connect to database in command line client*

  USE company;

*--  Creating Tables in Company  Database.*

*--  1. Creating employee Table--*

  create Table employee(

  fname varchar(15) not null,

  minit char,

  lname varchar(15) not null,

  ssn char(9) not null,

  DOB Date,

  address varchar(30),

  sex char,

  salary Decimal(10,2),

  super\_ssn char(9),

  dnumber int not null,

  primary key(ssn)

  );

*-- 2. Creating department Table*

  create table department(

  dname varchar(15) not null,

  dnumber int not null,

  mgr\_ssn char(9) not null,

  mgr\_start\_date Date,

  primary key(dnumber),

  unique(dname),

  foreign key(mgr\_ssn) references employee(ssn)

  );

*-- 3. Creating Dept\_Location Table*

  create table dept\_locations(

  dnumber int not null,

  dlocation varchar(15),

  primary key(dnumber,dlocation),

  foreign key(dnumber) references department(dnumber)

  );

*-- 4. Creating project Table*

  create table project(

  pname varchar(15),

  pnumber int,

  plocation varchar(15),

  dnumber int,

  primary key(pnumber),

  unique(pname),

  foreign key(dnumber) references department(dnumber)

  );

*-- 5. Creating Works\_on Table*

  create table works\_on(

  ssn char(9) not null,

  pnumber int not null,

  hours decimal(3,1) not null,

  primary key(ssn,pnumber),

  foreign key(ssn) references employee(ssn),

  foreign key(pnumber) references project(pnumber)

  );

*-- 6 Creating Dependent Table*

  create table dependent(

  ssn char(9),

  dependent\_name varchar(15),

  sex char,

  DOB date,

  relationship varchar(8),

  primary key(ssn,dependent\_name),

  foreign key(ssn) references employee(ssn)

  );

*--  Insert data into employee table*

*--  Insert data into employee table*

INSERT INTO employee

VALUES

('John', 'B', 'Smith', '123456789', '1965-01-09', '731 Fondren, Houston, TX', 'M', 30000, '333445555', 5),

('Franklin', 'T', 'Wong', '333445555', '1955-12-08', '638 Voss, Houston, TX', 'M', 40000, '888665555', 5),

('Alicia', 'J', 'Zelaya', '999887777', '1968-01-19', '3321 Castle, Spring, TX', 'F', 25000, '333445555', 4),

('Jennifer', 'S', 'Wallace', '987654321', '1941-06-20', '291 Berry, Bellaire, TX', 'F', 43000, '888665555', 4),

('Ramesh', 'K', 'Narayan', '666884444', '1962-09-15', '975 Fire Oak, Humble, TX', 'M', 38000, '333445555', 5),

('Joyce', 'A', 'English', '453453453', '1972-07-31', '5631 Rice, Houston, TX', 'F', 55000, NULL, 1),

('Ahmad', 'V', 'Jabbar', '987987987', '1969-03-29', '980 Dallas, Houston, TX', 'M', 25000, '888665555', 4),

('James', 'E', 'Borg', '888665555', '1937-11-10', '450 Stone, Houston, TX', 'M', 25000, '333445555', 5);

*--  Insert data into department table*

INSERT INTO department

VALUES

('Research', 5, '333445555', '1988-05-22'),

('Administration', 4, '987654321', '1995-01-01'),

('Headquarters', 1, '888665555', '1981-06-19');

*--  Insert data into DEPT\_LOCATIONS table*

INSERT INTO dept\_locations

VALUES

(1, 'Houston'),

(5, 'Houston'),

(5, 'Sugarland'),

(5, 'Bellaire'),

(4, 'Stafford');

*--  Insert data into WORKS\_ON table*

INSERT INTO works\_on

VALUES

('123456789', 1, 32.567),

('123456789', 2, 7.5),

('666884444', 3, 40.0),

('453453453', 1, 20.0),

('453453453', 2, 20.0),

('333445555', 2, 10.0),

('333445555', 3, 10.0),

('333445555', 10, 10.0),

('333445555', 20, 10.0),

('333445555', 30, 30.0),

('987654321', 30, 5.0),

('987654321', 20, 15.0);

*--  Insert data into project table*

INSERT INTO project

VALUES

('ProductX', 1, 'Bellaire', 5),

('ProductY', 2, 'Sugarland', 5),

('ProductZ', 3, 'Houston', 5),

('Computerization', 10, 'Stafford', 4),

('Reorganization', 20, 'Houston', 1),

('Newbenefits', 30, 'Stafford', 5);

*--  Insert data into DEPENDENT table*

INSERT INTO dependent

VALUES

('333445555', 'Alice', 'F', '1986-04-05', 'Daughter'),

('333445555', 'Theodore', 'M', '1983-10-25', 'Son'),

('333445555', 'Joy', 'F', '1958-05-03', 'Spouse'),

('987654321', 'Abner', 'M', '1942-02-28', 'Spouse'),

('123456789', 'Michael', 'M', '1988-01-04', 'Son'),

('123456789', 'Alice', 'F', '1988-12-30', 'Daughter'),

('888665555', 'Elizabeth', 'F', '1967-05-05', 'Spouse');

*-- Queries Executed on Company DataBase from Chapter-7 of Book*

*-- 1. Retrieve the names of all employees who do not have supervisors.*

  SELECT fname,lname

  FROM employee

  WHERE super\_ssn IS NULL;

*-- 2.*

  SELECT DISTINCT pnumber

  FROM project

  WHERE pnumber IN

    ( SELECT pnumber

    FROM project, department, employee

    WHERE project.dnumber = department.dnumber AND

    mgr\_ssn = ssn AND lname = 'Smith' )

  OR

  pnumber IN

    ( SELECT pnumber

    FROM works\_on, employee

    WHERE works\_on.ssn = employee.ssn AND lname = 'Smith' );

*-- 3.  Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.*

  SELECT fname,lname from employee,dependent

  WHERE dependent.ssn=employee.ssn AND dependent.sex = employee.sex;

*-- 4. Retrieve the names of employees who have no dependents.*

  select fname,lname from employee

  where employee.ssn not in (select dependent.ssn from dependent where employee.ssn=dependent.ssn);

*-- 5. List the names of managers who have at least one dependent*

  SELECT fname,lname from employee

  WHERE

  EXISTS (SELECT \* FROM department WHERE employee.ssn =department.mgr\_ssn)

  AND

  EXISTS (SELECT \* FROM dependent WHERE employee.ssn = dependent.ssn);

*-- 6. Retrieve the name of each employee who works on all the projects controlled by department number 5*

  SELECT fname,lname from employee

 WHERE NOT EXISTS (

  (

  SELECT pnumber

  FROM project

  WHERE project.dnumber = 5

  )

 EXCEPT (

 SELECT pnumber

 FROM works\_on

 WHERE employee.ssn = works\_on.ssn)

 );

*-- 7. . Retrieve the Social Security numbers of all employees who work on project numbers 1, 2, or 3.*

  SELECT DISTINCT Essn

  FROM WORKS\_ON

  WHERE Pno IN (1, 2, 3);

*-- 8. Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary*

  SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)

  FROM EMPLOYEE;

*-- 9. Find the sum of the salaries of all employees of the ‘Research’ department, as well as the maximum salary, the minimum salary, and the average salary in this department.*

  SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)

  FROM (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)

  WHERE Dname = ‘Research’;

*-- 10. Retrieve the total number of employees in the company and the number of employees in the ‘Research’ department*

  SELECT COUNT(\*) FROM EMPLOYEE;

  SELECT COUNT(\*)

  FROM EMPLOYEE, DEPARTMENT

  WHERE DNO = DNUMBER AND DNAME = ‘Research’;

*-- 11. Count the number of distinct salary values in the database.*

  SELECT COUNT (Salary)

  FROM EMPLOYEE;

*-- 12. For each department, retrieve the department number, the number of employees in the department, and their average salary*

  SELECT Dno, COUNT(\*), AVG (Salary)

  FROM EMPLOYEE

  GROUP BY Dno;

*-- 13. For each project, retrieve the project number, the project name, and the number of employees who work on that project.*

  SELECT Pnumber, Pname, COUNT(\*)

  FROM PROJECT, WORKS\_ON

  WHERE Pnumber = Pno

  GROUP BY Pnumber, Pname;

*-- 14.  For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.*

  SELECT Pnumber, Pname, COUNT(\*)

  FROM PROJECT, WORKS\_ON

  WHERE Pnumber = Pno

  GROUP BY Pnumber, Pname

  HAVING COUNT(\*) > 2;

*-- 15. . For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.*

  SELECT Pnumber, Pname, COUNT(\*)

  FROM PROJECT, WORKS\_ON, EMPLOYEE

  WHERE Pnumber = Pno AND Ssn = Essn AND Dno = 5

  GROUP BY Pnumber, Pname;

*-- 16.  For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.*

  SELECT Dno, COUNT(\*)

  FROM EMPLOYEE

  WHERE Salary>40000 AND Dno IN

  ( SELECT Dno

  FROM EMPLOYEE

  GROUP BY Dno

  HAVING COUNT(\*) > 5);

**THANK YOU !!**