***DBMS***

***PRACTICAL LIST***

***SUBMITTED BY: Daksh Sahni*** [B.Sc(H)CS] 16023

***SUBMITTED TO: Mr. Rakesh Kumar***

**1.) Create and use the following student-society database schema for a college to answer the queries using the standalone SQL editor**

**2.) Do the following database administration commands:**

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

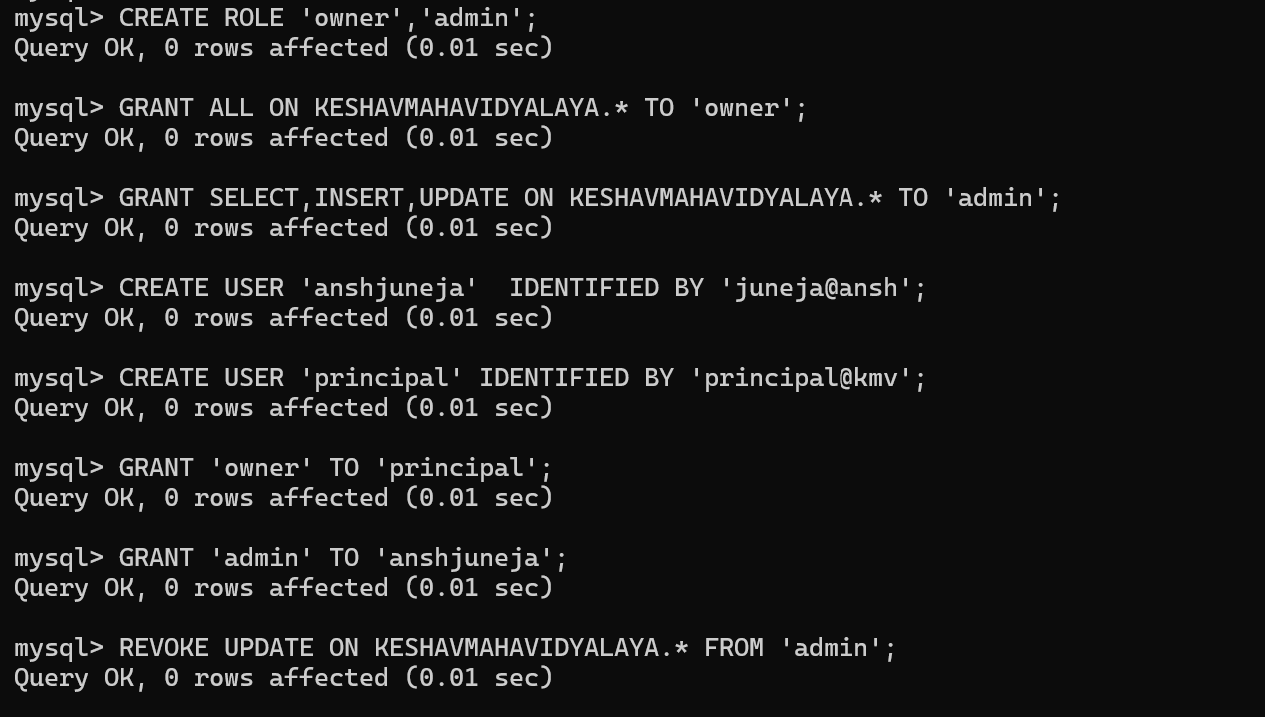
**3.) Execute Queries through a high-level language using ODBC Connection.**

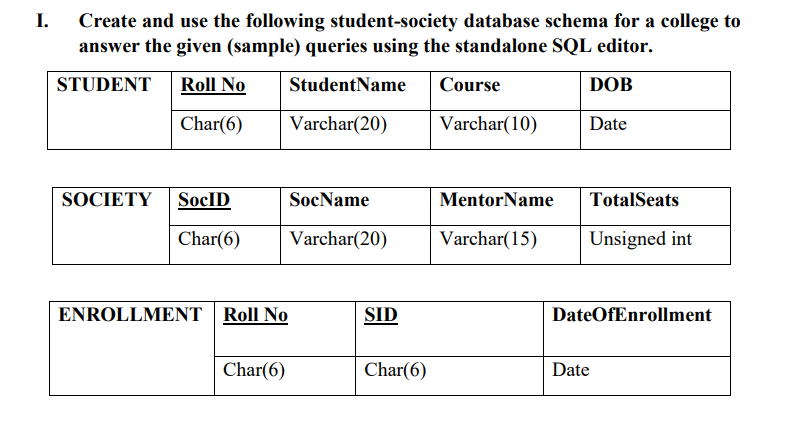
**4.) Implement the COMPANY database schema and execute the queries.**

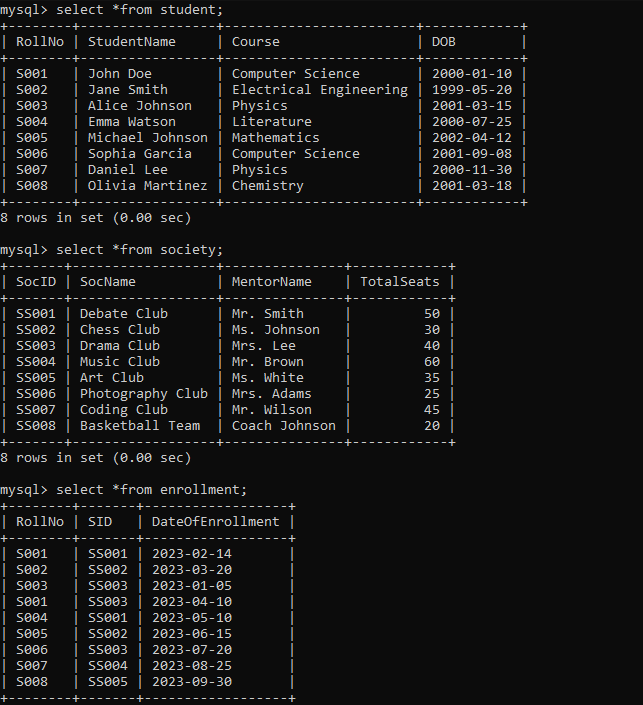
**2.) Do the following database administration commands:**

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

privileges from a role, create index

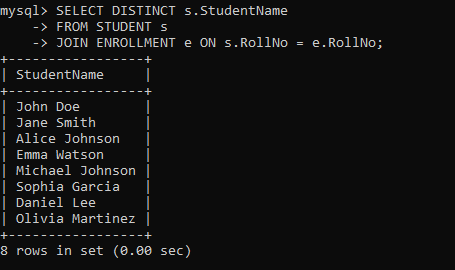




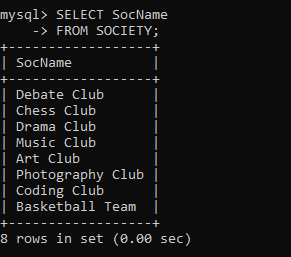
**SHOW TABLE**

**QUERIES:**

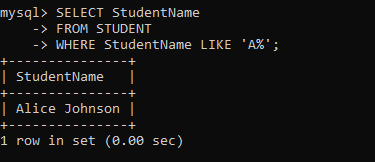
1. **Retrieve names of students enrolled in any society.**



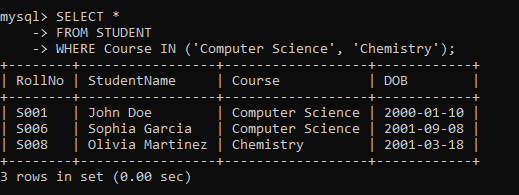
1. **Retrieve all society names.**



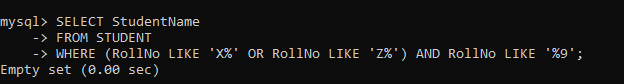
1. **Retrieve students' names starting with letter ‘A’.**



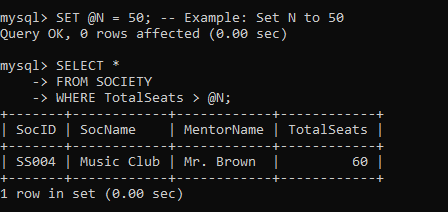
1. **Retrieve students' details studying in courses ‘computer science’ or ‘chemistry’.**



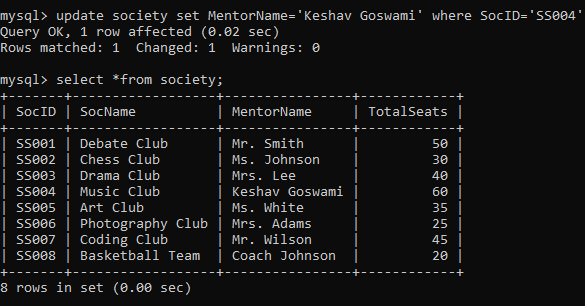
1. **Retrieve students’ names whose roll no either starts with ‘X’ or ‘Z’ and ends with ‘9’**



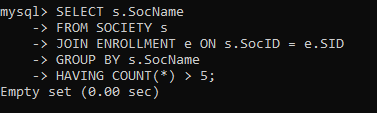
1. **Find society details with more than N TotalSeats where N is to be input by the user**



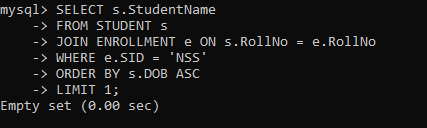
1. **Update society table for mentor name of a specific society**



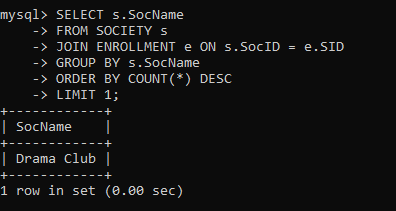
1. **Find society names in which more than five students have enrolled**



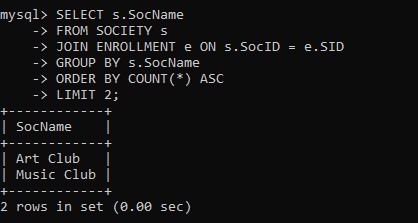
1. **Find the name of youngest student enrolled in society ‘NSS’**



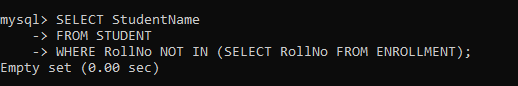
1. **Find the name of most popular society (on the basis of enrolled students)**



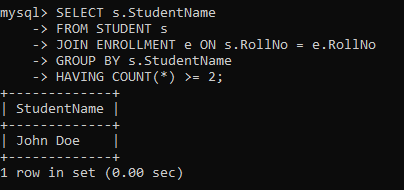
1. **Find the name of two least popular societies (on the basis of enrolled students)**



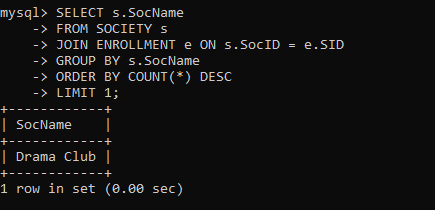
1. **Find the student names who are not enrolled in any society**



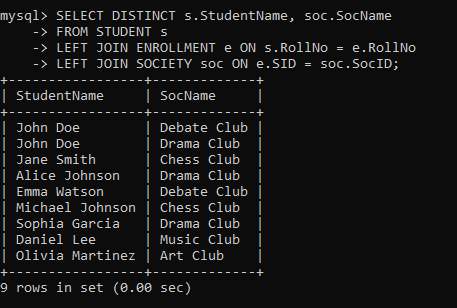
1. **Find the student names enrolled in at least two societies**



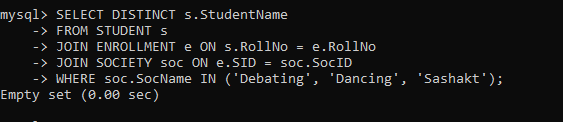
1. **Find society names in which maximum students are enrolled**



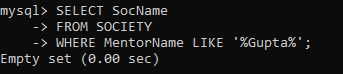
1. **Find names of all students who have enrolled in any society and society names in which at least one student has enrolled**



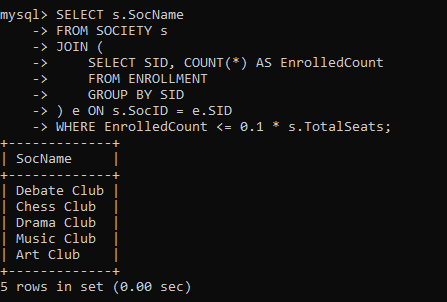
1. **Find names of students who are enrolled in any of the three societies ‘Debating’, ‘Dancing’, and ‘Sashakt’.**



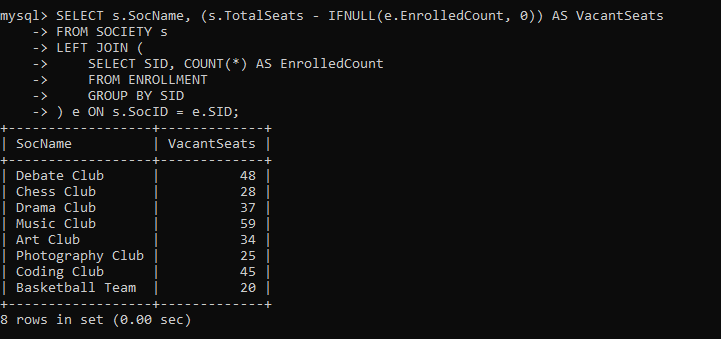
1. **Find society names such that its mentor has a name with ‘Gupta’ in it.**



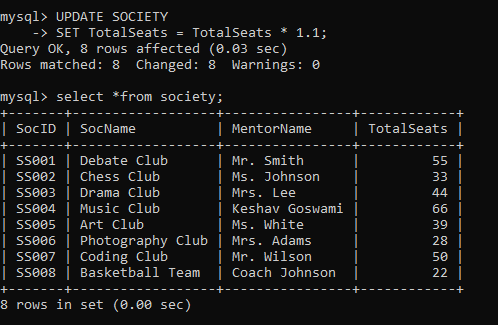
1. **Find the society names in which the number of enrolled students is only 10% of its capacity.**



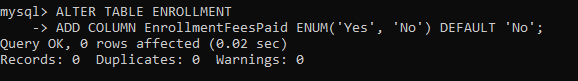
1. **Display the vacant seats for each society.**



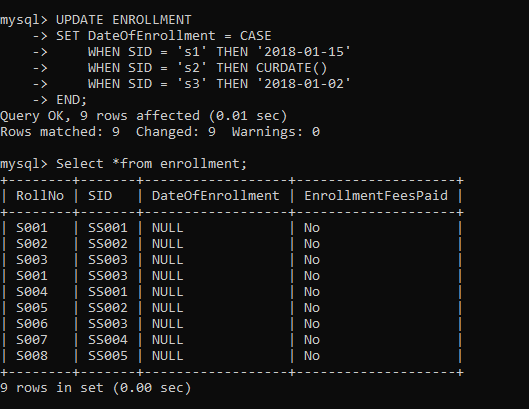
1. **Increment Total Seats of each society by 10%**



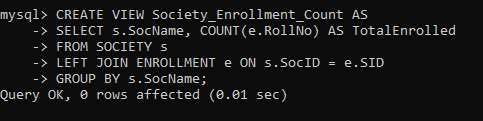
1. **Add the enrollment fees paid (‘yes’/’No’) field in the enrollment table.**



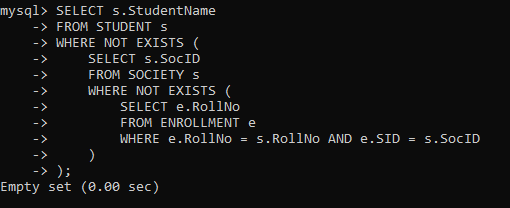
1. **Update date of enrollment of society id ‘s1’ to ‘2018-01-15’, ‘s2’ to current date and ‘s3’ to ‘2018-01-02’.**



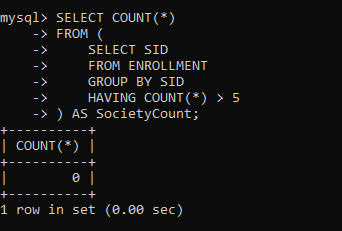
1. **Create a view to keep track of society names with the total number of students enrolled in it.**



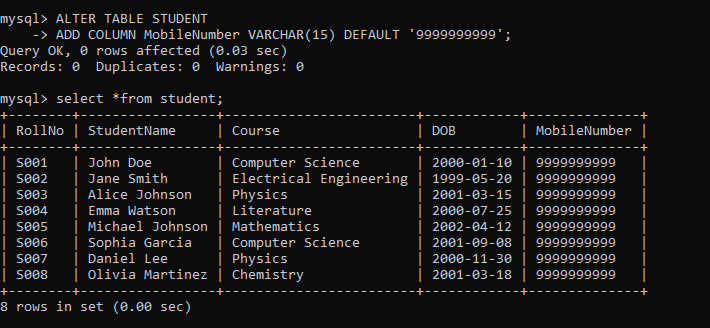
1. **Find student names enrolled in all the societies.**



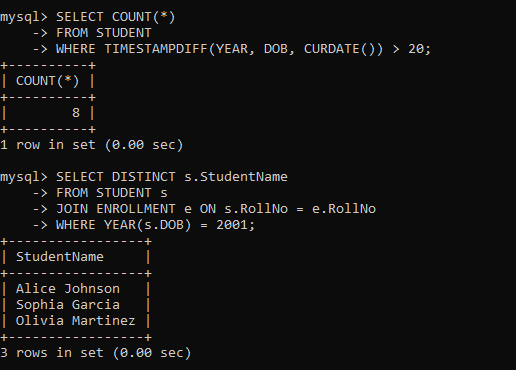
1. **Count the number of societies with more than 5 students enrolled in it**



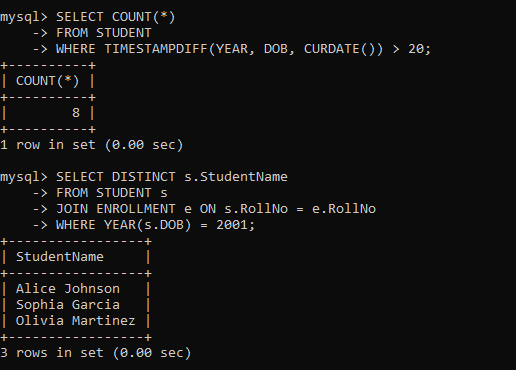
1. **Add column Mobile number in student table with default value ‘9999999999’**



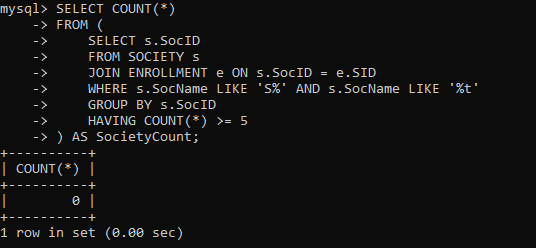
1. **Find the total number of students whose age is > 20 years.**



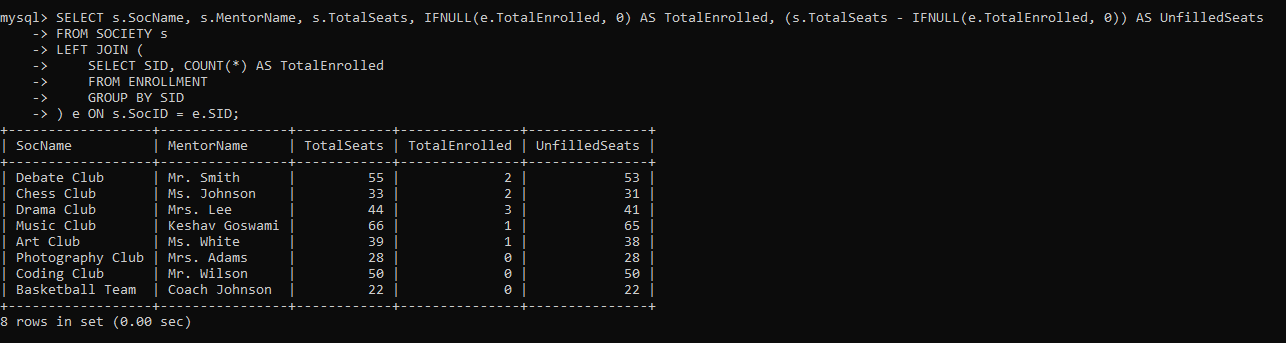
1. **Find names of students who are born in 2001 and are enrolled in at least one society.**



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



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



**3.) Execute Queries through a high-level language using ODBC Connection.**

*import* mysql.connector

db= mysql.connector.connect(

*host* = "localhost",

*user* = "root",

*password* = "Juneja@AJ",

*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,(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*:

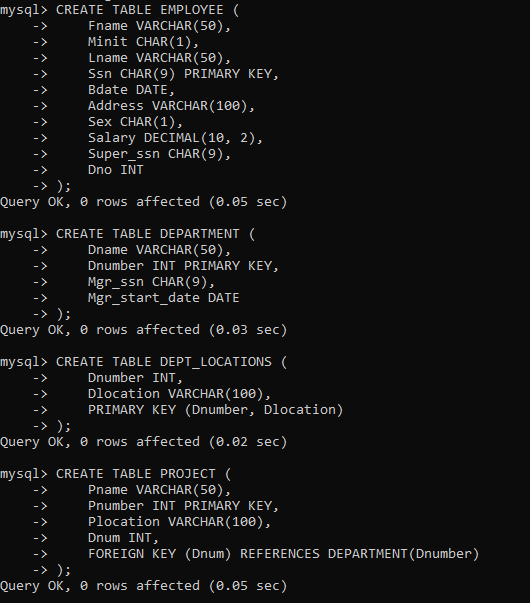
print("Invalid choice. Please enter a valid option.")

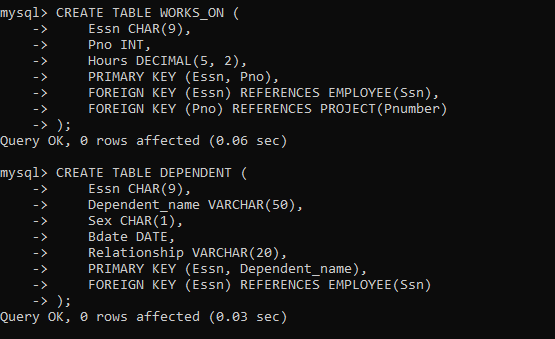
*if* \_\_name\_\_ == "\_\_main\_\_":

main()

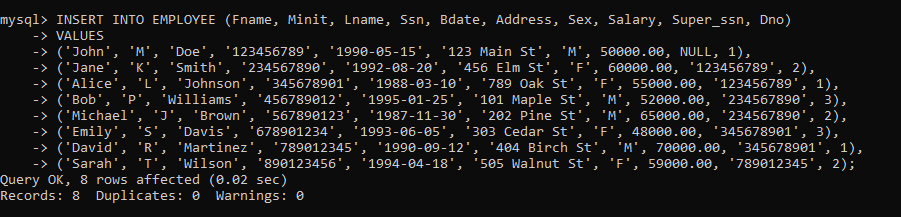
**4.) Implement the COMPANY database schema and execute the queries.**

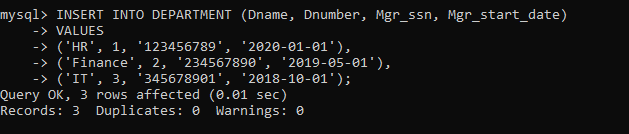
**TABLE CREATION:**

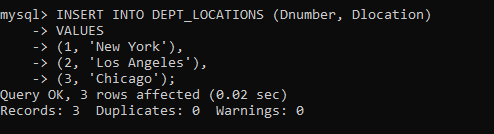


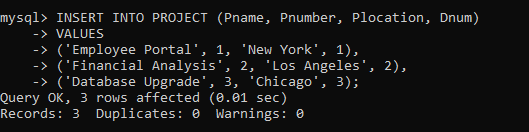


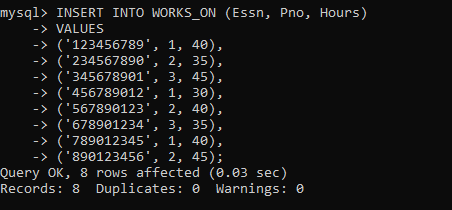
**INSERTING VALUES IN TABLES:**

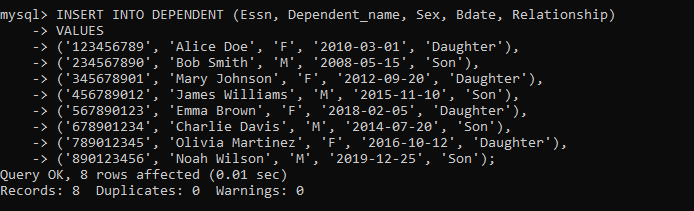




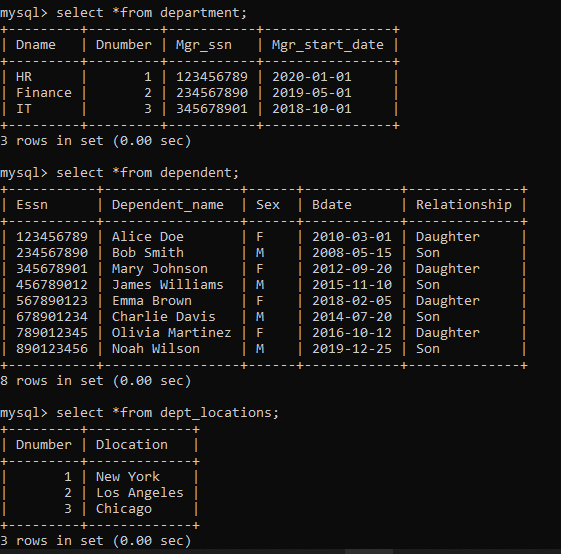


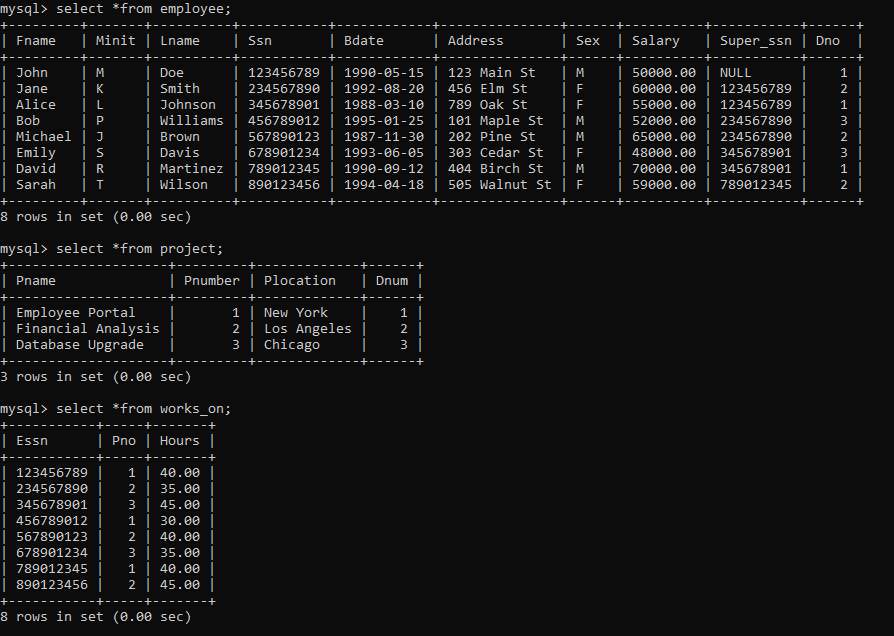






**SHOWING TABLES:**





**QUERIES:**

