NAME-Barsha Routh

Reg no-24MCA0164

DATABASE SYSTEMS

PROFESSOR(Prof\_id, Prof\_name, Email, Mobile, Specialty, Dept\_id)

SCHOOL(SCode, Scl\_name, Prof\_id, Location)

DEPARTMENT(Dept\_id, Dname, SCode, Prof\_id)

COURSE(Crs\_code, Crs\_name, Description, Credits, Hours)

CLASS(Cls\_code, Slot, Stime, Etime, Crs\_code, Prof\_id, Room\_no, Sem\_code, Day\_of\_week)

SEMESTER(Sem\_code, Term, Year, Sdate, Edate)

STUDENT(Reg\_no, Sname, Address, DoB, Email, Mobile, Dept\_id, Prof\_id)

ENROLL(Cls\_code, Reg\_no, Enroll\_time, Grade)

STUDENT\_VISA(Reg\_no, Visa\_status)

PROGRAMME(Prog\_code, Prog\_name, Prog\_preamble, Scode, Dept\_id)

1. Create the above tables.

**SOLUTION**

create table PROFESSOR(Prof\_id varchar(5),Prof\_name varchar(20),Email varchar(20),Mobile number(10),Speciality varchar(10),Dept\_id varchar(5));

create table SCHOOL(SCode varchar(5),Scl\_name varchar(20),Prof\_id varchar(5),Location varchar(10));

create table DEPARTMENT(Dept\_id varchar(5),Dname varchar(20),Scode varchar(5),Prof\_id varchar(5));

create table COURSE (Crs\_code varchar(5),Crs\_name varchar(10),Description varchar(30),Credits number(2),Hours number(2));

CREATE TABLE CLASS (Cls\_code VARCHAR(5),Slot VARCHAR(5),Stime TIMESTAMP(0),Etime TIMESTAMP(0),Crs\_code VARCHAR(5), Prof\_id VARCHAR(5),Room\_no VARCHAR(5), Sem\_code VARCHAR(5), Day\_of\_week VARCHAR(10));

Create table SEMESTER(sem\_code varchar(5),term varchar(6),Year varchar(4),Sdate date,Edate date);

Create table STUDENT(Reg\_no varchar(5),Sname varchar(10),Address varchar(20),DoB date,Email varchar(20),Mobile varchar(10),Dept\_id varchar(5),Prof\_id varchar(5));

Create table ENROLL(Cls\_code varchar(5),Reg\_no varchar(5),Enroll\_time timestamp(0), Grade CHAR(1) CHECK (Grade IN ('S', 'A', 'B', 'C', 'D')));

Create table STUDENT\_VISA(Reg\_no varchar(5),Visa\_status Varchar(20));

Create table PROGRAMME(Prog\_code varchar(5), Prog\_name varchar(10), Prog\_preamble varchar(50), Scode varchar(5), Dept\_id varchar(5));

OUTPUT IN THE NEXT QUESTION--

1. Enter data into the above tables. (Learn also how to enter data interactively.). Display the content of each table. Use column formatting while displaying data.

**SOLUTION**

INTERACTIVE ENTRY OF DATA

insert into PROFESSOR values(&Prof\_id,&Prof\_name,&Email,&Mobile,&Specialty,&Dept\_id);

insert into DEPARTMENT values(&Dept\_id, &Dname, &SCode, &Prof\_id);

insert into SCHOOL values(&SCode, &Scl\_name, &Prof\_id , &Location);

insert into COURSE values(&Crs\_code , &Crs\_name , &Description , &Credits , &Hours);

insert into CLASS values(&Cls\_code , &Slot , &Stime , &Etime , &Crs\_code , &Prof\_id , &Room\_no, &Sem\_code , &Day\_of\_week);

insert into SEMESTER values(&Sem\_code, &Term , &Year, &Sdate, &Edate);

insert into STUDENT values(&Reg\_no , &Sname , &Address, &DoB, &Email , &Mobile, &Dept\_id , &Prof\_id );

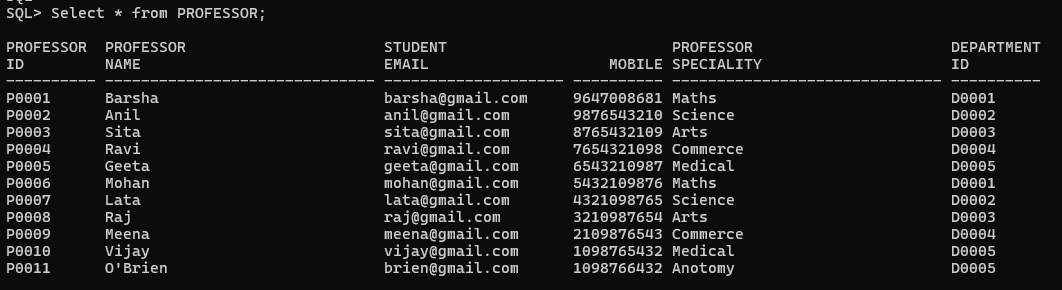
insert into ENROLL values(&Cls\_code, &Reg\_no , &Enroll\_time , &Grade);

insert into STUDENT\_VISA values(&Reg\_no, &Visa\_status);

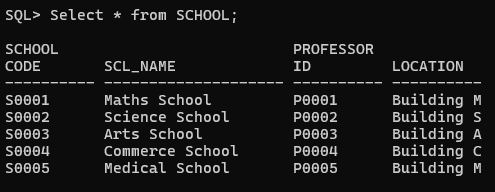
insert into PROGRAMME values(&Prog\_code , &Prog\_name, &Prog\_preamble, &Scode , &Dept\_id);

**OUTPUT**

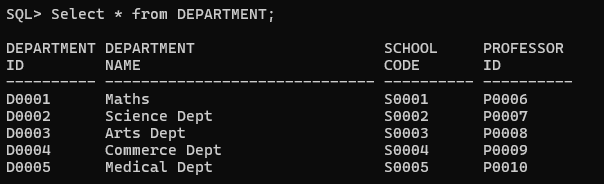
PROFESSOR



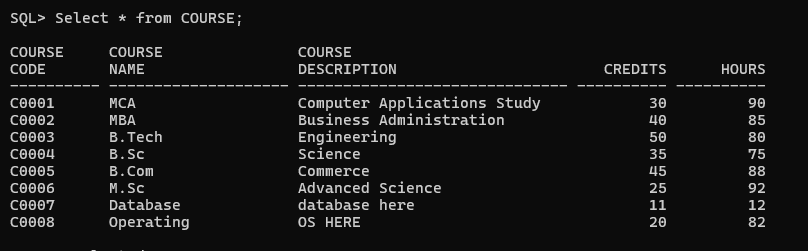
SCHOOL



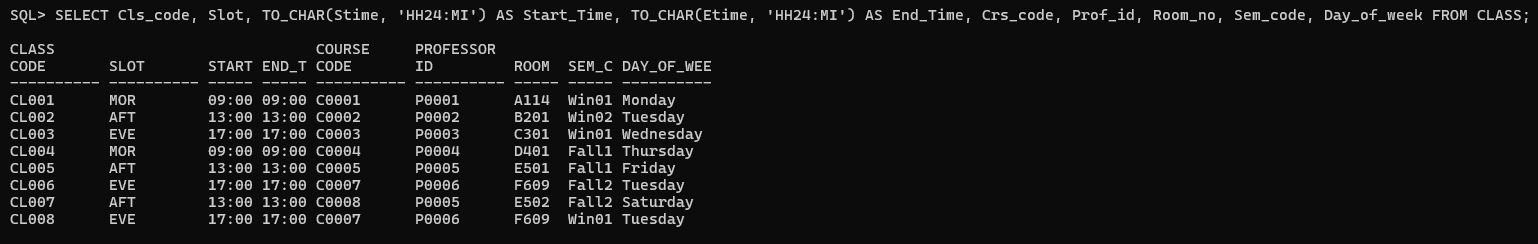
DEPARTMENT



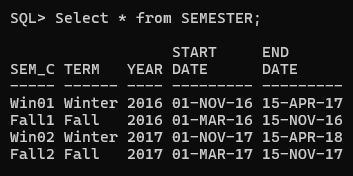
COURSE



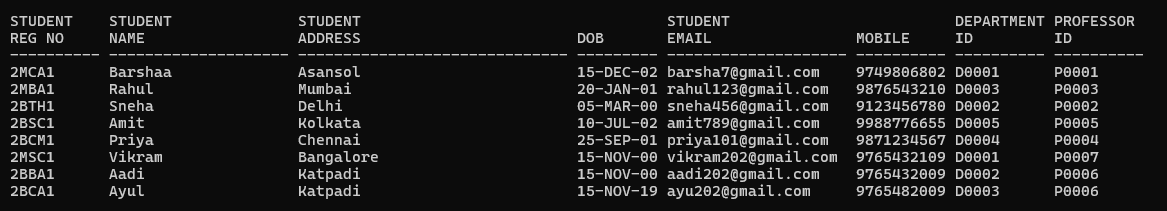
CLASS



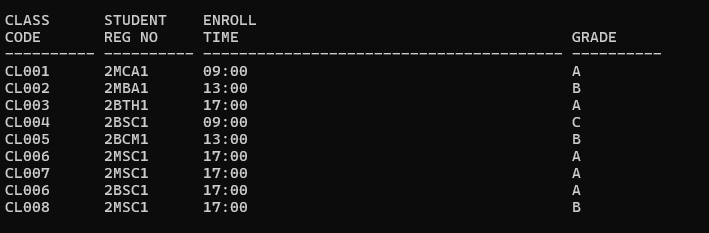
SEMESTER



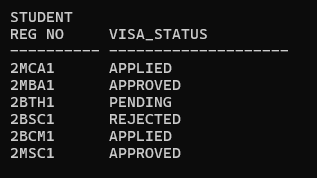
STUDENT



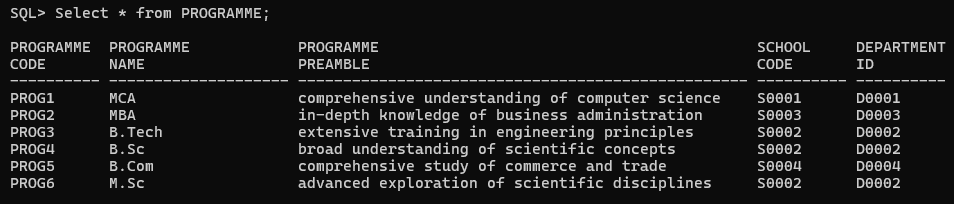
ENROLL



STUDENT VISA



PROGRAMME



1. **Alter or Recreate the above tables with primary key and foreign key and the following integrity constraints assigning name to integrity constrain**

**SOLUTION**

--PRIMARY KEYS

ALTER TABLE PROFESSOR ADD CONSTRAINT pk\_professor PRIMARY KEY (Prof\_id);

ALTER TABLE SCHOOL ADD CONSTRAINT pk\_school PRIMARY KEY (SCode);

ALTER TABLE DEPARTMENT ADD CONSTRAINT pk\_department PRIMARY KEY (Dept\_id);

ALTER TABLE COURSE ADD CONSTRAINT pk\_course PRIMARY KEY (Crs\_code);

ALTER TABLE CLASS ADD CONSTRAINT pk\_class PRIMARY KEY (Cls\_code);

ALTER TABLE SEMESTER ADD CONSTRAINT pk\_semester PRIMARY KEY (Sem\_code);

ALTER TABLE STUDENT ADD CONSTRAINT pk\_student PRIMARY KEY (Reg\_no);

ALTER TABLE ENROLL ADD CONSTRAINT pk\_enroll PRIMARY KEY (Cls\_code, Reg\_no);

ALTER TABLE STUDENT\_VISA ADD CONSTRAINT pk\_student\_visa PRIMARY KEY (Reg\_no);

ALTER TABLE PROGRAMME ADD CONSTRAINT pk\_programme PRIMARY KEY (Prog\_code);

--Add FK

alter table PROFESSOR add constraint fk\_Dept foreign key(Dept\_id) references DEPARTMENT(Dept\_id) deferrable initially deferred;

alter table SCHOOL add constraint fk\_Prof\_id foreign key(Prof\_id) references PROFESSOR(Prof\_id) deferrable initially deferred;

alter table DEPARTMENT add constraint fk\_Profid foreign key(Prof\_id) references PROFESSOR(Prof\_id) deferrable initially deferred;

alter table DEPARTMENT add constraint fk\_SCode foreign key(SCode) references SCHOOL(SCode) deferrable initially deferred;

alter table CLASS add constraint fk\_Crs\_code foreign key(Crs\_code) references COURSE(Crs\_code) deferrable initially deferred;

alter table CLASS add constraint fk\_Prfid foreign key(Prof\_id) references PROFESSOR (Prof\_id) deferrable initially deferred;

alter table CLASS add constraint fk\_SemCode foreign key(Sem\_code) references Semester(Sem\_code) deferrable initially deferred;

alter table STUDENT add constraint fk\_Depid foreign key(Dept\_id) references DEPARTMENT (Dept\_id) deferrable initially deferred;

alter table STUDENT add constraint fk\_Proid foreign key(Prof\_id) references PROFESSOR (Prof\_id) deferrable initially deferred;

alter table Enroll add constraint fk\_ClsCode foreign key(Cls\_code) references CLASS (Cls\_code) deferrable initially deferred;

alter table Enroll add constraint fk\_RegNo foreign key(Reg\_no) references STUDENT (Reg\_no) deferrable initially deferred;

alter table PROGRAMME add constraint fk\_SC foreign key(Scode) references SCHOOL (SCode) deferrable initially deferred;

alter table PROGRAMME add constraint fk\_DepI foreign key(Dept\_id) references DEPARTMENT (Dept\_id) deferrable initially deferred;

1. i**) Prof\_id must have exactly five characters and their email and mobile number are unique. The email address must have @ as one of the characters and mobile number must have exactly ten characters.**

**ii) Use timestamp data type without fractional parts of seconds for start time and end time column of class table**

**iii) The Sem\_code should start with either ‘Win’ or ‘Fall’ and Term column can assume only one of two values {Winter, Fall}.**

**iv) Email and mobile column in student table should have same characteristics as those in professor table.**

**v) The enroll\_time in the enroll table should be of timestamp data type without fractional parts of seconds. The grade may assume one of the values in {‘S’, ‘A’, ‘B’, ‘C’, ‘D’}**

**vi) Use ‘on delete cascade’ or ‘on delete set null’ clause as requirements. Use deferrable constraint, if required.**

**vii) Additional (innovative) integrity constraints, if any, may be specified by you.**

--PROFESSOR

alter table PROFESSOR add constraint uk\_email unique(Email);

alter table PROFESSOR add constraint uk\_Mobile unique(Mobile);

alter table PROFESSOR add constraint chk\_Len CHECK(LENGTH(Prof\_id)=5);

alter table PROFESSOR add constraint chk\_Len\_Mob CHECK(LENGTH(Mobile)=10);

alter table PROFESSOR add constraint chk\_Email\_atTheRate CHECK(Email like '%@%');

--SEMESTER

alter table SEMESTER add constraint chk\_SemesCod CHECK(sem\_code Like 'Win%' or sem\_code like 'Fall%' );

alter table SEMESTER add constraint chk\_Term CHECK(Term IN ('Winter', 'Fall') );

--STUDENT

alter table STUDENT add constraint uk\_stu\_email unique(Email);

alter table STUDENT add constraint uk\_stu\_Mobile unique(Mobile);

alter table STUDENT add constraint chk\_Len\_StuMob CHECK(LENGTH(Mobile)=10);

alter table STUDENT add constraint chk\_stuEmail\_atTheRate CHECK(Email like '%@%');

--ENROLL

alter table Enroll add constraint chk\_value CHECK(Grade IN ('S', 'A','B','C','D'));

**4. . In built functions**

**(i) Test the string manipulation functions – UPPER, LOWER, INITCAP, LENGTH, LPAD, RPAD, LTRIM, RTRIM and TRIM, using select queries on data present in the tables. Use one query each for demonstration of one function**

SELECT UPPER(Sname) FROM STUDENT;

SELECT LOWER(Sname) FROM STUDENT;

SELECT INITCAP(Visa\_status) FROM STUDENT\_VISA;

SELECT LENGTH(Dname) FROM DEPARTMENT;

SELECT LPAD(Visa\_status,12,'\*') FROM STUDENT\_VISA;

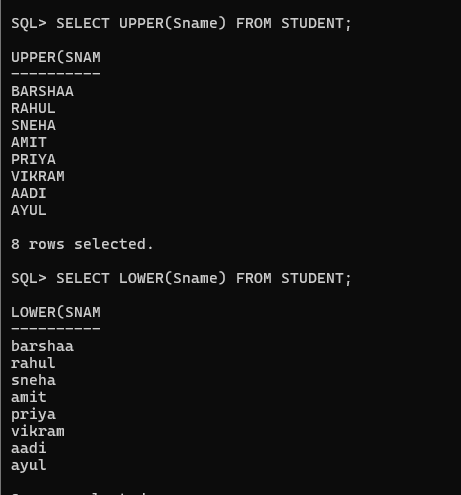
SELECT RPAD(Visa\_status,12,'\*') FROM STUDENT\_VISA;

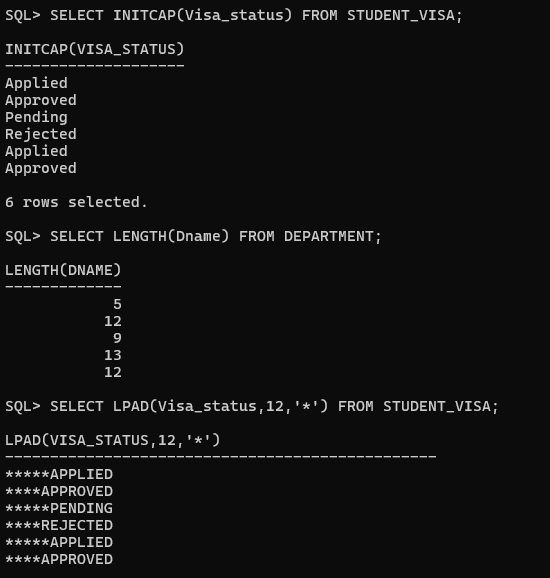
SELECT TRIM(emp\_name) FROM Emp1;

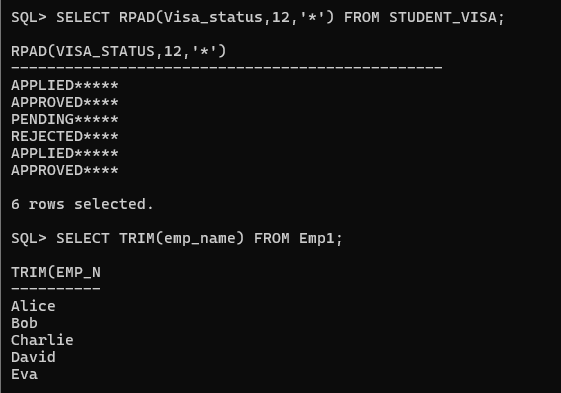
SELECT RTRIM(emp\_name,' ') FROM Emp1;

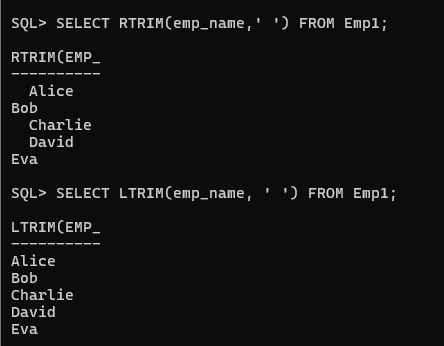
SELECT LTRIM(emp\_name, ' ') FROM Emp1;

**SOLUTION**





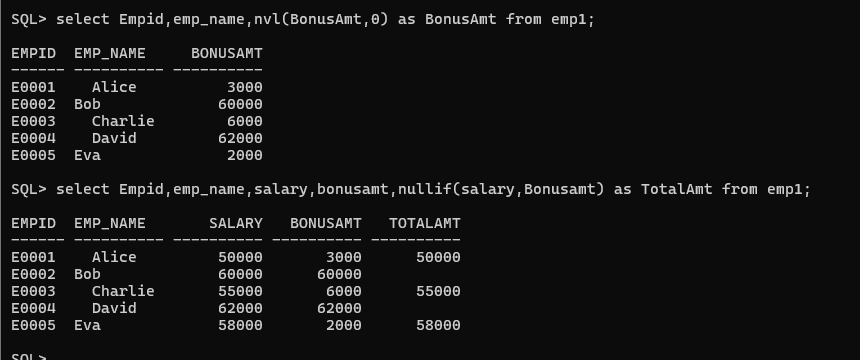




(**ii) Write query to illustrate usage of NVL function and NULLIF function.**

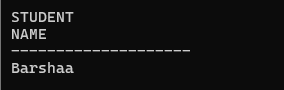
select Empid,emp\_name,nvl(BonusAmt,0) as BonusAmt from emp1;

select Empid,emp\_name,salary,bonusamt,nullif(salary,Bonusamt) as TotalAmt from emp1;  --not returning null when not equal



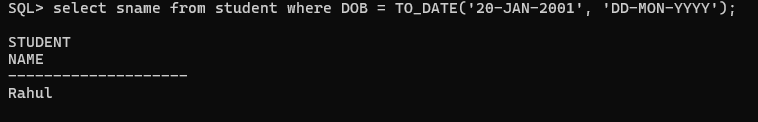
(**iii) Display the name of the students who were born on a specified month.**

select sname from student where extract(month from DOB)=12;



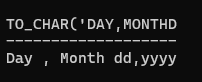
**(iv) Display the name of the students with a specified date of birth.**

select sname from student where DOB = TO\_DATE('20-JAN-2001', 'DD-MON-YYYY');



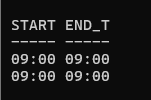
(v**) Display the date of birth of a specified student in the format ‘Day of week, Month dd, yyyy’**.

select TO\_CHAR('Day , Month dd,yyyy') from student where sname='Barshaa';



**(vi) Display the hour and minutes of the start time and end time of a specified slot.**

SELECT TO\_CHAR(Stime, 'HH24:MI') AS Start\_Time, TO\_CHAR(Etime, 'HH24:MI') AS End\_Time FROM CLASS WHERE Slot = 'MOR';



**(vii) Display the day of week of the start date and end date of Winter semester 17–18.**

SELECT

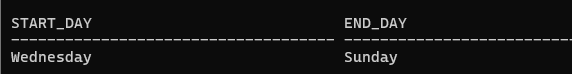
    TO\_CHAR(Sdate, 'Day') AS Start\_Day,

    TO\_CHAR(Edate, 'Day') AS End\_Day

FROM SEMESTER

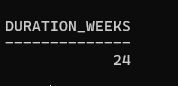
WHERE Term = 'Winter'

  AND Year = '2017';



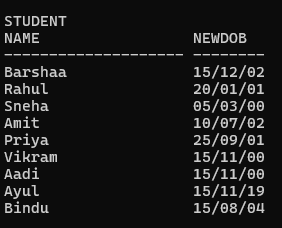
**(viii) Display the duration of Winter semester 17–18 in terms of number of weeks.**

SELECT ROUND((Edate - Sdate) / 7) AS Duration\_Weeks FROM SEMESTER WHERE Term = 'Winter' AND Year = '2017';



**(ix) Store date in the format dd/mm/yy for DOB of newly admitted student.**

SELECT sname, TO\_CHAR(DOB, 'DD/MM/YY') AS NewDob FROM student;



**(x) Test the numeric functions – CEIL, FLOOR, TRUCATE, MIN, MAX, AVG,COUNT using select queries on data present in the tables. Use one query each for demonstration of one function.**

SELECT CEIL(AVG(Credits)) FROM COURSE;

SELECT FLOOR(AVG(Credits)) FROM COURSE;

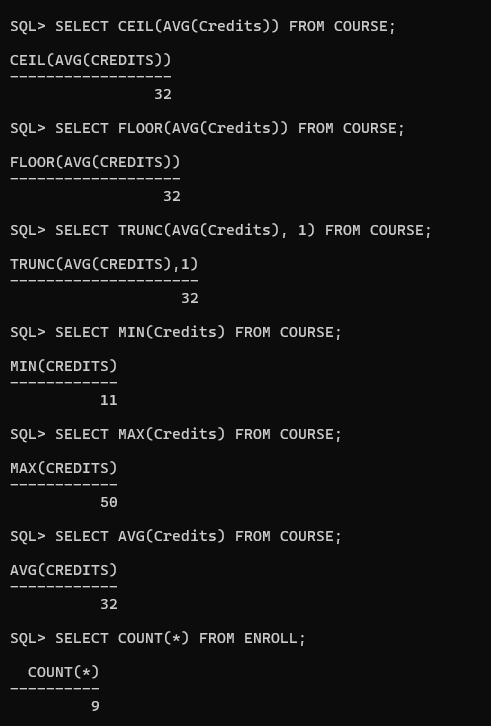
SELECT TRUNC(AVG(Credits), 1) FROM COURSE;

SELECT MIN(Credits) FROM COURSE;

SELECT MAX(Credits) FROM COURSE;

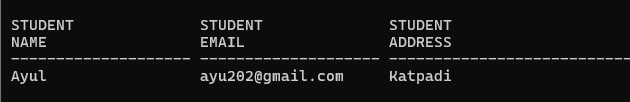
SELECT AVG(Credits) FROM COURSE;

SELECT COUNT(\*) FROM ENROLL;



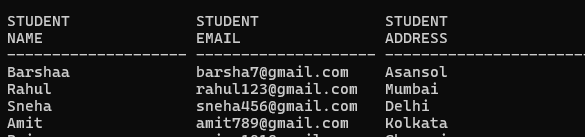
1. **Write Queries for**
2. **Display name, email address and address for those students who live in Katpadi area and whose name has an l as the third character.**

SELECT sname, Email, Address FROM Student WHERE Address = 'Katpadi' AND sname LIKE '\_\_\_l%';



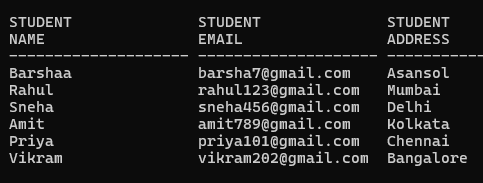
1. **Display name, email address and address for those students who are not from Tamil Nadu.**

SELECT sname, Email, Address FROM Student WHERE Address != 'Tamil Nadu';



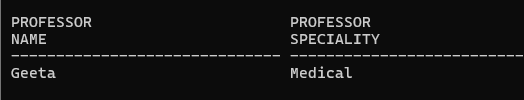
1. **Display name, email address and address of foreign students only.**

SELECT STUDENT.SNAME, STUDENT.EMAIL,STUDENT.ADDRESS FROM STUDENT INNER JOIN STUDENT\_VISA ON STUDENT.REG\_NO = STUDENT\_VISA.REG\_NO;

****

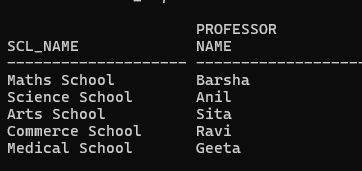
**(iv) List the name of professors along with their specialty who belong to School of Medicine.**

SELECT PROFESSOR.PROF\_NAME, PROFESSOR.SPECIALITY FROM PROFESSOR INNER JOIN SCHOOL ON SCHOOL.SCL\_NAME = 'Medical School' AND SCHOOL.PROF\_ID = PROFESSOR.PROF\_ID;



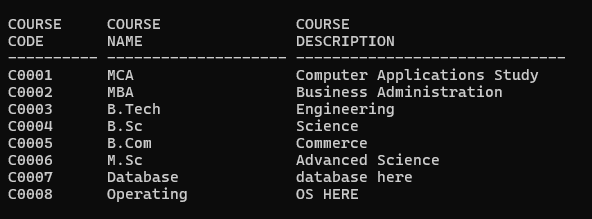
**v.Display name of the school and name of professor who chairs the school.**

SELECT SCHOOL.SCL\_NAME, PROFESSOR.PROF\_NAME FROM PROFESSOR INNER JOIN SCHOOL ON SCHOOL.PROF\_ID = PROFESSOR.PROF\_ID;



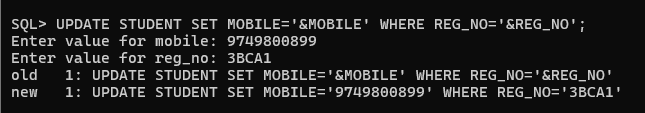
1. **List course code, course name and course description in alphabetic order of course code.**

SELECT CRS\_CODE,CRS\_NAME, DESCRIPTION FROM COURSE ORDER BY CRS\_CODE;



1. **Change the mobile number of a student interactively.**

UPDATE STUDENT SET MOBILE='&MOBILE' WHERE REG\_NO='&REG\_NO';



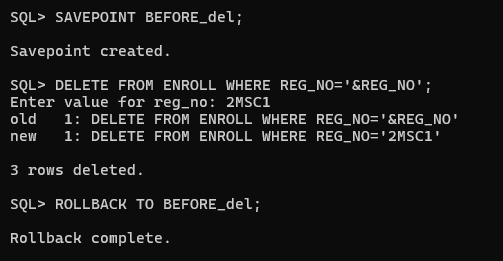
**(viii) Remove enrollment information of a student from a particular course interactively. How would you recover the data?**

SAVEPOINT BEFORE\_del;

DELETE FROM ENROLL WHERE REG\_NO='&REG\_NO';

--we can recover data with the help of ROLLBACK

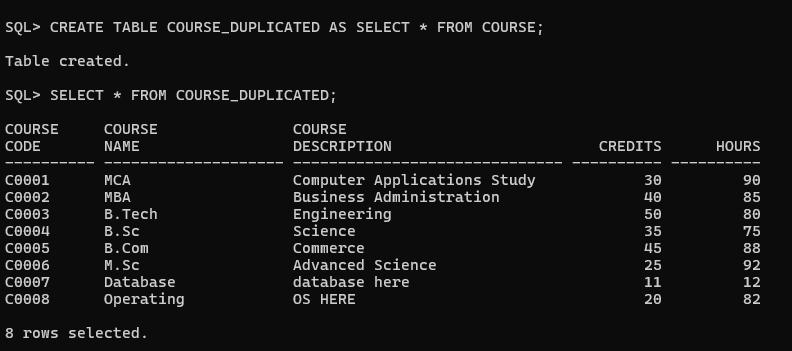
ROLLBACK TO BEFORE\_del;



**(ix) Create a duplicate of course table.**

CREATE TABLE COURSE\_DUPLICATED AS SELECT \* FROM COURSE;

SELECT \* FROM COURSE\_DUPLICATED;



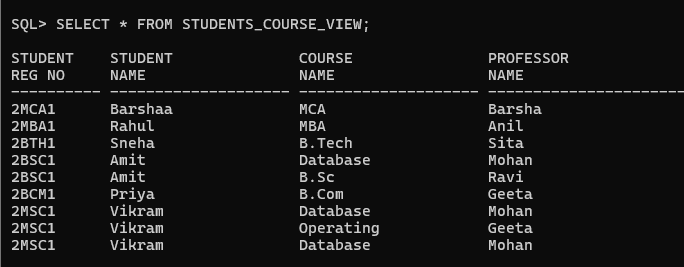
**(x) Create a view for list of students (Reg\_no, Sname) and the courses they have registered along with name of professors teaching the course.**

CREATE VIEW STUDENTS\_COURSE\_VIEW AS

SELECT STUDENT.REG\_NO, STUDENT.SNAME, COURSE.CRS\_NAME, PROFESSOR.PROF\_NAME FROM STUDENT INNER JOIN ENROLL ON ENROLL.REG\_NO = STUDENT.REG\_NO INNER JOIN CLASS ON ENROLL.CLS\_CODE = CLASS.CLS\_CODE INNER JOIN COURSE ON COURSE.CRS\_CODE = CLASS.CRS\_CODE

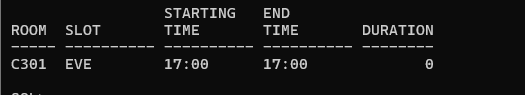
INNER JOIN PROFESSOR ON PROFESSOR.PROF\_ID = CLASS.PROF\_ID;

SELECT \* FROM STUDENTS\_COURSE\_VIEW;



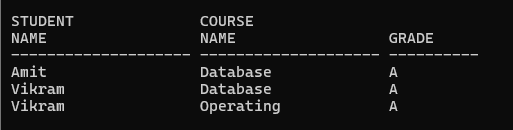
**(xi) List the room number, slot, start time, end time and duration of every class held on Wednesdays in descending order of room number.**

SELECT ROOM\_NO, SLOT, TO\_CHAR(Stime, 'HH24:MI') AS STIME, TO\_CHAR(Etime, 'HH24:MI') AS ETIME, EXTRACT (HOUR FROM ETIME - STIME) AS "DURATION" FROM CLASS WHERE DAY\_OF\_WEEK = 'Wednesday' ORDER BY ROOM\_NO DESC;

****

**(xii) Display the name and grade of a student in different courses underwent in fall semester 2017 – 18.**

SELECT STUDENT.SNAME, COURSE.CRS\_NAME, ENROLL.GRADE FROM STUDENT INNER JOIN ENROLL ON ENROLL.REG\_NO = STUDENT.REG\_NO INNER JOIN CLASS ON CLASS.CLS\_CODE = ENROLL.CLS\_CODE INNER JOIN COURSE ON COURSE.CRS\_CODE = CLASS.CRS\_CODE INNER JOIN SEMESTER ON SEMESTER.SEM\_CODE = CLASS.SEM\_CODE AND SEMESTER.TERM = 'Fall' AND SEMESTER.YEAR = 2017;



**(xiii) Find out name of students who have taken Database Systems course as well as Operating Systems course in fall semester 2016 – 17.**

SELECT STUDENT.SNAME FROM STUDENT INNER JOIN ENROLL

ON ENROLL.REG\_NO = STUDENT.REG\_NO INNER JOIN CLASS

ON CLASS.CLS\_CODE = ENROLL.CLS\_CODE

INNER JOIN COURSE

ON CLASS.CRS\_CODE = COURSE.CRS\_CODE

AND COURSE.CRS\_CODE IN ('C0007',

'C0008')

INNER JOIN SEMESTER

ON SEMESTER.SEM\_CODE = CLASS.SEM\_CODE

AND SEMESTER.TERM = 'Fall'

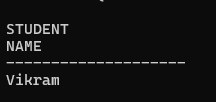
AND SEMESTER.YEAR = 2017

GROUP BY

STUDENT.SNAME

HAVING

COUNT(DISTINCT COURSE.CRS\_CODE) = 2;



**(xiv) Find out name of students who have taken Database Systems course but have not taken Operating Systems course in winter semester 2017 – 18.**

SELECT STUDENT.SNAME, COURSE.CRS\_CODE FROM STUDENT

INNER JOIN ENROLL ON ENROLL.REG\_NO = STUDENT.REG\_NO

INNER JOIN CLASS ON ENROLL.CLS\_CODE = CLASS.CLS\_CODE

INNER JOIN COURSE ON COURSE.CRS\_CODE = CLASS.CRS\_CODE AND COURSE.CRS\_CODE IN ('C0007','C0008')

INNER JOIN SEMESTER ON SEMESTER.SEM\_CODE = CLASS.SEM\_CODE AND SEMESTER.YEAR = 2017

AND SEMESTER.TERM = 'Winter'

MINUS

SELECT STUDENT.SNAME, COURSE.CRS\_CODE FROM STUDENT

INNER JOIN ENROLL ON ENROLL.REG\_NO = STUDENT.REG\_NO

INNER JOIN CLASS ON ENROLL.CLS\_CODE = CLASS.CLS\_CODE

INNER JOIN COURSE ON COURSE.CRS\_CODE = CLASS.CRS\_CODE

AND COURSE.CRS\_CODE = 'C0008' INNER JOIN SEMESTER ON SEMESTER.SEM\_CODE = CLASS.SEM\_CODE AND SEMESTER.YEAR = 2018

AND SEMESTER.TERM = 'Win01';



**(xv) List the registration number and name of the students who have registered for maximum number of credits in Winter 17-18 semester.**

SELECT S.Reg\_no, S.Sname

FROM STUDENT S

JOIN ENROLL E ON S.Reg\_no = E.Reg\_no

JOIN CLASS C ON E.Cls\_code = C.Cls\_code

JOIN COURSE CR ON C.Crs\_code = CR.Crs\_code

JOIN SEMESTER SEM ON C.Sem\_code = SEM.Sem\_code

WHERE SEM.Term = 'Win01'

AND SEM.Year = '2016'

GROUP BY S.Reg\_no, S.Sname

HAVING SUM(CR.Credits) = (

SELECT MAX(Total\_Credits)

FROM (

SELECT SUM(CR.Credits) AS Total\_Credits

FROM STUDENT S

JOIN ENROLL E ON S.Reg\_no = E.Reg\_no

JOIN CLASS C ON E.Cls\_code = C.Cls\_code

JOIN COURSE CR ON C.Crs\_code = CR.Crs\_code

JOIN SEMESTER SEM ON C.Sem\_code = SEM.Sem\_code

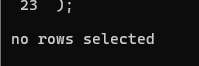
WHERE SEM.Term = 'Win01'

AND SEM.Year = '2016'

GROUP BY S.Reg\_no

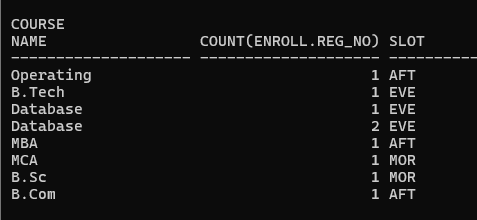
)

);



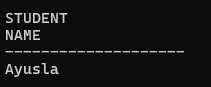
**(xvi) List the name of the course and the number of students registered in each slot for course under different faculty members.**

SELECT COURSE.CRS\_NAME, COUNT(ENROLL.REG\_NO), SLOT FROM COURSE, ENROLL, CLASS WHERE ENROLL.CLS\_CODE = CLASS.CLS\_CODE AND CLASS.CRS\_CODE = COURSE.CRS\_CODE GROUP BY ENROLL.REG\_NO, COURSE.CRS\_NAME, SLOT;



**(xvii) Find out the name of the students who have registered in all the courses being taught by Prof. O’Brien in Winter 17-18.**

SELECT STUDENT.SNAME FROM STUDENT, PROFESSOR, CLASS, ENROLL WHERE ENROLL.CLS\_CODE = CLASS.CLS\_CODE AND ENROLL.REG\_NO = STUDENT.REG\_NO AND CLASS.PROF\_ID = PROFESSOR.PROF\_ID AND PROFESSOR.PROF\_NAME = 'O''Brien';



**(xviii) List the registration number of the students who registered in Database Systems course on November 17, 2017**

SELECT STUDENT.REG\_NO

    FROM STUDENT, ENROLL, CLASS, COURSE

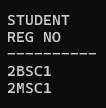
    WHERE ENROLL.REG\_NO = STUDENT.REG\_NO

    AND ENROLL.CLS\_CODE = CLASS.CLS\_CODE

    AND CLASS.CRS\_CODE = COURSE.CRS\_CODE

    AND TO\_CHAR(ENROLL.ENROLL\_TIME, 'DD-MM-YYYY') = '17-11-2017'

    AND COURSE.CRS\_NAME = 'Database';



**(xix) Write a query to display the grade of a student given his/her registration number and the course name for Fall semester 17–18.**

SELECT ENROLL.REG\_NO

    FROM ENROLL, CLASS, SEMESTER

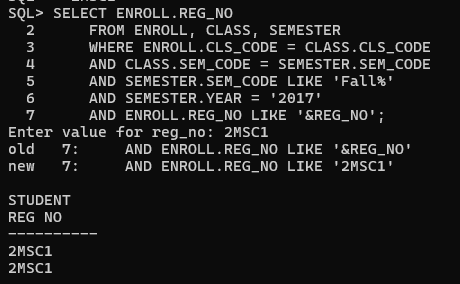
    WHERE ENROLL.CLS\_CODE = CLASS.CLS\_CODE

    AND CLASS.SEM\_CODE = SEMESTER.SEM\_CODE

    AND SEMESTER.SEM\_CODE LIKE 'Fall%'

    AND SEMESTER.YEAR = '2017'

    AND ENROLL.REG\_NO LIKE '&REG\_NO';

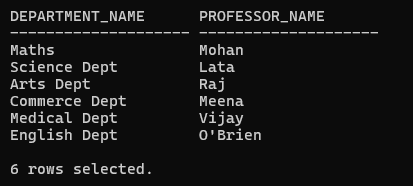


**(xx) List the name of departments and the name professors who is in charge of the department.**

SELECT Dept.Dname AS Department\_Name, Prof.Prof\_name AS Professor\_Name

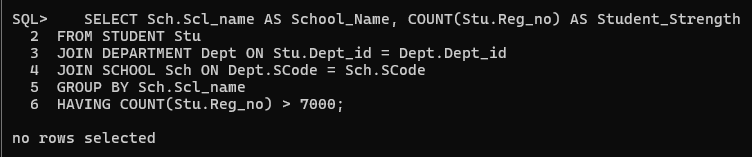
FROM DEPARTMENT Dept

JOIN PROFESSOR Prof ON Dept.Prof\_id = Prof.Prof\_id;



**(xxi) List the name of schools with students’ strength higher than 7000.**

SELECT Sch.Scl\_name AS School\_Name, COUNT(Stu.Reg\_no) AS Student\_Strength FROM STUDENT Stu JOIN DEPARTMENT Dept ON Stu.Dept\_id = Dept.Dept\_id JOIN SCHOOL Sch ON Dept.SCode = Sch.SCode GROUP BY Sch.Scl\_name HAVING COUNT(Stu.Reg\_no) > 7000;



**(xxii) List the name of the department(s) under school of medicine with student strength higher than the average students of all the departments in the school.**

**(xxiii) Given the registration number of a student, display the total credits registered by him/her in Winter 17–18**

SELECT S.Reg\_no, SUM(CR.Credits) AS Total\_Credits FROM ENROLL E

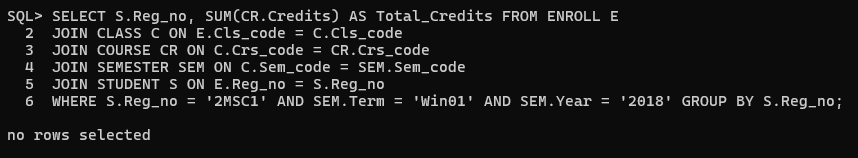
JOIN CLASS C ON E.Cls\_code = C.Cls\_code

JOIN COURSE CR ON C.Crs\_code = CR.Crs\_code

JOIN SEMESTER SEM ON C.Sem\_code = SEM.Sem\_code

JOIN STUDENT S ON E.Reg\_no = S.Reg\_no

WHERE S.Reg\_no = '2MSC1' AND SEM.Term = 'Win01' AND SEM.Year = '2018' GROUP BY S.Reg\_no;



**(xxiv) Given the registration number of a student, display her/his grade in the course she/he registered in Fall 17–18.**

SELECT E.Cls\_code, C.Crs\_code, C.Crs\_name, E.Grade

FROM ENROLL E

JOIN CLASS CL ON E.Cls\_code = CL.Cls\_code

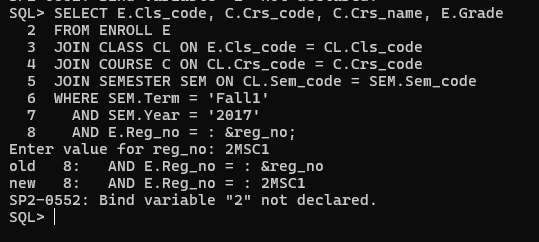
JOIN COURSE C ON CL.Crs\_code = C.Crs\_code

JOIN SEMESTER SEM ON CL.Sem\_code = SEM.Sem\_code

WHERE SEM.Term = 'Fall1'

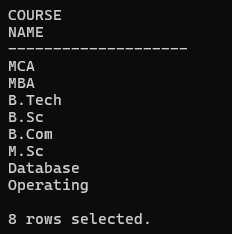
AND SEM.Year = '2017'

AND E.Reg\_no = : &reg\_no;



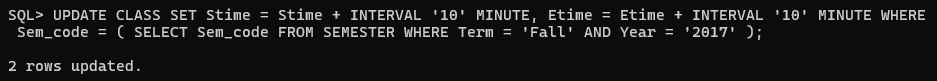
**(xxv) Display the name of the courses that are not being offered in Winter 17–18.**

SELECT C.Crs\_name FROM COURSE C WHERE C.Crs\_code NOT IN ( SELECT CL.Crs\_code FROM CLASS CL JOIN SEMESTER SEM ON CL.Sem\_code = SEM.Sem\_code WHERE SEM.Term = 'Win01' AND SEM.Year = '2017' );



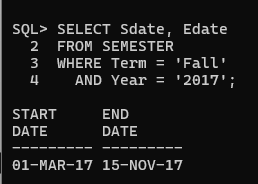
**(xxvi) Write necessary SQL statement to advance the start time and end time of every class by ten minutes in Fall 18–19.**

UPDATE CLASS SET Stime = Stime + INTERVAL '10' MINUTE, Etime = Etime + INTERVAL '10' MINUTE WHERE Sem\_code = ( SELECT Sem\_code FROM SEMESTER WHERE Term = 'Fall' AND Year = '2017' );



**(xxvii) Write necessary SQL statement to advance the start date and end date of Fall 18–19 semester by one week with respect to Fall semester of 17 – 18.**

**BEFORE**

****

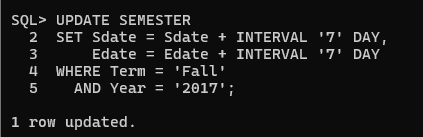
UPDATE SEMESTER

SET Sdate = Sdate + INTERVAL '7' DAY,

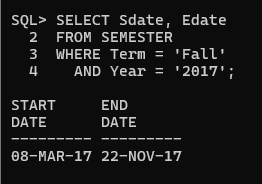
Edate = Edate + INTERVAL '7' DAY

WHERE Term = 'Fall'

AND Year = '2017';



**AFTER**

****

**(xxviii) Find out the name list of students who had secured ‘S’ grade in at least 50% of the courses cleared by her/him.**

SELECT S.Sname

FROM STUDENT S

WHERE

(SELECT COUNT(\*)

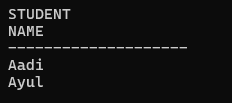
FROM ENROLL E

WHERE E.Reg\_no = S.Reg\_no AND E.Grade = 'S') >=

(SELECT COUNT(\*) / 2

FROM ENROLL E

WHERE E.Reg\_no = S.Reg\_no);



**(xxix) Given the registration number of a student, find out his/her free slots.**

SELECT DISTINCT Cl.Slot

FROM CLASS Cl

WHERE Cl.Slot NOT IN (

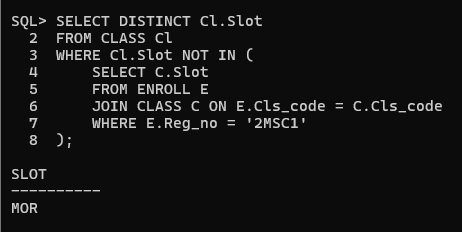
SELECT C.Slot

FROM ENROLL E

JOIN CLASS C ON E.Cls\_code = C.Cls\_code

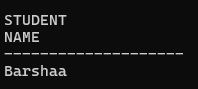
WHERE E.Reg\_no = '2MSC1'

);



**(xxx) Find out the name list of students who have classes in the afternoon session only a specific day of the week.**

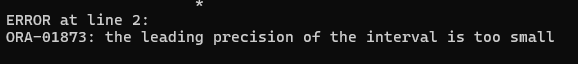
SELECT DISTINCT S.Sname FROM STUDENT S JOIN ENROLL E ON S.Reg\_no = E.Reg\_no JOIN CLASS C ON E.Cls\_code = C.Cls\_code WHERE C.Day\_of\_week = 'Monday' -- Specify the day of the week AND Slot=’Aft’;



**(xxxi) Add a column named ‘Duration’ (to indicate duration of a class) with appropriate data type to the CLASS table and populate the column from values of start time and end time columns.**

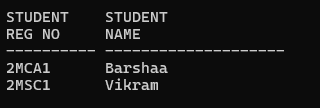
ALTER TABLE CLASS ADD Duration INTERVAL DAY TO SECOND;

UPDATE CLASS SET Duration = Etime - Stime;

****

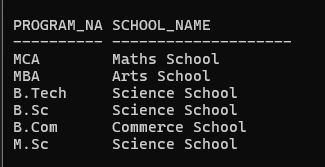
**(xxxiii) Find out the list of students who are undergoing MCA program.**

SELECT S.Reg\_no, S.Sname FROM STUDENT S JOIN PROGRAMME P ON S.Dept\_id = P.Dept\_id WHERE P.Prog\_name = 'MCA';



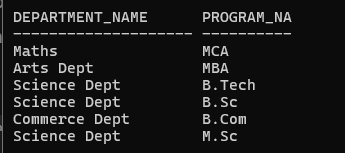
**(xxxiv) Display the name of programs and the name of school offering the program.**

SELECT Prog.Prog\_name AS Program\_Name, Sch.Scl\_name AS School\_Name FROM PROGRAMME Prog JOIN SCHOOL Sch ON Prog.SCode = Sch.SCode;



**(xxxv) Display the name of the departments and the name of the program controlled by the department.**

SELECT Dept.Dname AS Department\_Name, Prog.Prog\_name AS Program\_Name FROM DEPARTMENT Dept JOIN PROGRAMME Prog ON Dept.Dept\_id = Prog.Dept\_id;



**(xxxvi) Find the school which has highest school strength (i.e number of students)**

SELECT Scl\_name, Student\_Count FROM ( SELECT Sch.Scl\_name, COUNT(Stu.Reg\_no) AS Student\_Count FROM STUDENT Stu JOIN DEPARTMENT Dept ON Stu.Dept\_id = Dept.Dept\_id JOIN SCHOOL Sch ON Dept.SCode = Sch.SCode GROUP BY Sch.Scl\_name ORDER BY Student\_Count DESC ) WHERE ROWNUM = 1;

