LAB PROGRAM 1

1. Consider the following schema for a Library Database:

BOOK (BOOK_ID, TITLE, PUBLISHER_NAME, PUB_YEAR)

BOOK_AUTHORS (BOOK_ID, AUTHOR_NAME)

PUBLISHER (NAME, ADDRESS, PHONE)

BOOK_COPIES (BOOK_ID, BRANCH_ID, NO_OF_COPIES)

BOOK_LENDING (BOOK_ID, BRANCH_ID, CARD_NO, DATE_OUT, DUE_DATE)

LIBRARY_BRANCH (BRANCH_ID, BRANCH_NAME, ADDRESS)

CARD (CARD_NO)

Write SQL queries to

a) Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch.
b) Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.
c) Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
d) Create a view of all books and its number of copies that are currently available in the library.
e) Delete a book in the BOOK table. Update the contents of other tables to reflect this data manipulation operation.

Solution:

```
create table publisher(
name varchar(10) primary key,
pub_address varchar(10),
phone int);

create table library_branch(
branch_id int primary key,
branch_name varchar(10),
address varchar(10));

create table card(
card_no int primary key);
```

```
create table book(
book_id int primary key,
title varchar(20),
pub_name varchar(10) references publisher(name) on delete set null,
pub_year varchar(20)
);
create table book_author(
book_id int references book(book_id) on delete cascade,
author_name varchar(10),
primary key(book_id,author_name)
);
create table book_copies(
book_id int references book(book_id) on delete cascade,
branch_id int references library_branch(branch_id) on delete set null,
no_of_copies int,
primary key(book_id,branch_id)
);

create table book_lending(
book_id int references book(book_id) on delete cascade,
branch_id int references library_branch(branch_id) on delete cascade,
card_no int references card(card_no) on delete cascade,
date_out date,
due_date date,
primary key(book_id,branch_id,card_no)
);

insert into publisher values('Jaico Publishing House','Mumbai',3260618);
insert into publisher values('Penguin Random House','New York',18007333000);
insert into publisher values('Hachette India','Gurgaon',911244195000);
```

```
insert into publisher values('24by7Publishing','Kolkata',09433444334);
insert into publisher values('Srishti Publishers','Delhi',01141751981);

insert into library_branch values(1,'ISE','VVCE');
insert into library_branch values(2,'CSE','VVCE');
insert into library_branch values(3,'AIML','VVCE');
insert into library_branch values(4,'MECH','VVCE');
insert into library_branch values(5,'EC','VVCE');

insert into card values(1000);
insert into card values(1001);
insert into card values(1002);
insert into card values(1003);
insert into card values(1004);

insert into book values(100,'The Monk who sold his Ferrari','Jaico Publishing House',1996);
insert into book values(101,'The God of Small Things','Penguin Random House',1997);
insert into book values(102,'Durbar','Hachette India',2012);
insert into book values(103,'Kanishka','24by7Publishing',2017);
insert into book values(104,'Life Is What You Make It','Srishti Publishers',2011);

insert into book_author values(100,'Robin Sharma');
insert into book_author values(101,'Arundhati Roy');
insert into book_author values(102,'Tavleen Singh');
insert into book_author values(103,'Manoj Krishnan');
insert into book_author values(104,'Preeti Shenoy');

insert into book_copies values(100,1,6);
insert into book_copies values(101,2,7);
insert into book_copies values(102,3,8);
```

```
insert into book_copies values(103,4,9);
insert into book_copies values(104,5,10);
insert into book_copies values(102,2,10);
insert into book_copies values(104,1,10);

insert into book_lending values(100,1,1000,'01-Jan-2017','01-Feb-2017');
insert into book_lending values(101,2,1001,'01-Feb-2017','01-Mar-2017');
insert into book_lending values(102,3,1002,'01-Apr-2017','01-May-2017');
insert into book_lending values(103,4,1003,'01-Apr-2017','01-May-2017');
insert into book_lending values(104,5,1004,'01-May-2017','01-Jun-2017');
insert into book_lending values(102,1,1000,'10-Jan-2017','10-Feb-2017');
insert into book_lending values(103,2,1000,'10-Jan-2017','10-Feb-2017');
insert into book_lending values(101,2,1003,'01-Apr-2017','01-May-2017');
insert into book_lending values(104,3,1003,'01-Apr-2017','01-May-2017');
```

Queries

Q1.

```
select b.book_id,b.title,b.pub_name,a.author_name,c.branch_id,c.no_of_copies
from Book1 b, book1_author a,book_copies c
where b.book_id=a.book_id and a.book_id=c.book_id;
```

Q2

```
select card_no from book_lending
where date_out between '01-JAN-2017' and '01-jun-2017'
group by card_no
having count(*)>=3;
```

Q3.

```
create view book_pub_year as
select pub_year from book1;

select * from book_pub_year;
```

Q4.

```
create view book_in_lib as

select b.book_id,b.title,c.branch_id,c.no_of_copies

from book1 b, book_copies c

where b.book_id=c.book_id;


select * from book_in_lib;
```

Q5.

```
delete from book1 where book_id=103;
```

------*----------*----------*----------

LAB PROGRAM 2

2. Consider the following schema for Order Database:

SALESMAN (SALESMAN_ID, NAME, CITY, COMMISSION)

CUSTOMER (CUSTOMER_ID, CUST_NAME, CITY, GRADE, SALESMAN_ID)

ORDERS (ORD_NO, PURCHASE_AMT, ORD_DATE, CUSTOMER_ID, SALESMAN_ID)

Write SQL queries to

a) Count the customers with grades above Bangalore's average.
b) Find the name and numbers of all salesmen who had more than one customer.
c) List all the salesman and indicate those who have and do not have customers in their cities (Use UNION operation.)
d) Create a view that finds the salesman who has the customer with the highest order of a day.
e) Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

Solution:

```
create table Salesman(

Salesman_ID int primary key,

Name varchar(20),

City varchar(20),

Commission varchar(20));


Create table Customer(

Customer_ID int primary key,
```

```
City varchar(20),
Grade int,
Salesman_ID references Salesman(Salesman_ID) on delete set null,
Cust_Name varchar(20));

create table Orders(Ord_No int primary key,
Purchase_Amt Number,
Ord_Date date,
Customer_ID references Customer(Customer_ID) on delete cascade,
Salesman_ID references Salesman(Salesman_ID) on delete cascade);

insert into salesman values(1000,'Rita','Mysuru','25%');
insert into salesman values(1001,'Jeevan','Bangalore','20%');
insert into salesman values(1002,'Baby','Bangalore','15%');
insert into salesman values(1004,'Abhi','Shivamogga','30%');

insert into customer values(1,'Bangalore',100,1000,'Mita');
insert into customer values(2,'Mysuru',200,1001,'Gita');
insert into customer values(3,'Mangalore',300,1000,'Anna');
insert into customer values(4,'Bangalore',400,1001,'Louis');
insert into customer values(5,'Bangalore',500,1002,'rose');

insert into orders values(50,5000,'04-May-2017',1,1000);
insert into orders values(51,450,'20-Jun-2017',1,1001);
insert into orders values(52,1000,'20-Jun-2017',3,1000);
insert into orders values(53,3500,'13-Apr-2017',4,1002);
insert into orders values(54,550,'09-Mar-2017',5,1001);
insert into orders values(55,5500,'13-Apr-2017',4,1004);
```

```
/* Q1 */
Select cust_name
from customer
where grade > (Select avg(grade)
        from customer
        where city='Bangalore');

/* Q2 */
select s.salesman_id, s.name ,count(*) as no_of_customers
from salesman s, customer c
where s.salesman_id=c.salesman_id
group by s.salesman_id, s.name
having count(*)>1;

/* Q3 using UNION */
select s.name , c.cust_name, s.city
from salesman s, customer c
where s.city=c.city
union
select s.name, c.cust_name , 'NO MATCH'
from salesman s, customer c
where s.city not in (select city from customer)

/* Q4 using Group by */
create view MAX_AMT_ODR as
Select o.ord_date, s.salesman_id, s.name
from salesman s, orders o
where s.salesman_id=o.salesman_id and o.purchase_amt in (select max(purchase_amt)
                                                         from orders
                                                         group by ord_date)
```

```
select * from MAX_AMT_ODR;


/* Q4 using correlated queries */

create view salesman_order as

select b.Ord_Date, a.Salesman_ID,Name

from Salesman a, Orders B

where A.Salesman_ID=b.Salesman_ID and b.Purchase_Amt= (select max(Purchase_Amt)

                                                       from Orders C

                                                      where C.Ord_Date=b.Ord_Date);

select * from salesman_order;


/* Q5 */

delete from salesman

where salesman_id=1000;
```

----------*----------*----------*----------

LAB PROGRAM 3

3. Consider the schema for Company Database:

EMPLOYEE(SSN, NAME, ADDRESS, SEX, SALARY, SUPERSSN, DNO)

DEPARTMENT(DNO, DNAME, MGRSSN, MGRSTARTDATE)

DLOCATION(DNO,DLOC)

PROJECT(PNO, PNAME, PLOCATION, DNO)

WORKS_ON(SSN, PNO, HOURS)

Write SQL queries to

a) Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that Controls the project.
b) Show the resulting salaries if every employee working on the 'IoT' project is Given a 10 percent raise.
c) Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department
d) Retrieve the name of each employee who works on the entire projects controlledby department number 5 (use NOT EXISTS operator).

e) For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.

Solution:

```
create table Employee(
SSN int primary  key,
Name varchar(20),
Address varchar(20),
Sex varchar(10),
Salary int,
Super_SSN int references Employee(SSN) on delete cascade
);

create table Department(
DNO int primary key,
Dname varchar(20),
Manager_SSN int references Employee(SSN) on delete cascade,
Start_Date date
);

alter table Employee add DNO int references Department(DNO) on delete cascade;

create table Dlocation(
DNO int references Department(DNO) on delete cascade,
Dloc varchar(20),
primary key(DNO,Dloc)
);

create table Project(
PNO int primary key,
```

```
Pname varchar(20),
Plocation varchar(20),
DNO int references Department(DNO) on delete cascade
);

create table Works_On(
SSN int references Employee(SSN) on delete cascade,
PNO int references Project(PNO) on delete cascade,
Hours int,
primary key(SSN,PNO)
);

insert into Employee values(1,'Scott','Bangalore','Male',650000,null,null);
insert into Employee values(2,'Mary','Delhi','Female',700000,null,null);
insert into Employee values(3,'Louis','Bangalore','Male',500000,null,null);
insert into Employee values(4,'Cindu','New York','Female',450000,null,null);
insert into Employee values(5,'Baby','Delhi','Female',650000,null,null);
insert into Employee values(6,'Siemen','Bangalore','Male',650000,null,null);
insert into Employee values(7,'Rosy','Bangalore','Female',850000,null,null);
insert into Employee values(8,'Ronaldo','Bangalore','Male',800000,null,null);
insert into Employee values(9,'Messi','New York','Male',850000,null,null);
insert into Employee values(10,'Ron','Bangalore','Male',800000,null,null);
insert into Employee values(11,'Harry','New York','Male',850000,null,null);
insert into Employee values(12,'Daniel','Bangalore','Male',800000,null,null);
insert into Employee values(13,'Austin','New York','Male',850000,null,null);

insert into Department values(50,'Development',1,'01-Jan-2018');
insert into Department values(51,'HR',2,'01-Feb-2017');
insert into Department values(52,'Accounts',5,'01-Mar-2018');
insert into Department values(53,'Networking',8,'05-May-2016');
```

```
update Employee set Super_SSN=3 ,DNO=50 where SSN=1;
update Employee set Super_SSN=7 ,DNO=51 where SSN=2;
update Employee set Super_SSN=4 ,DNO=50 where SSN=3;
update Employee set Super_SSN=6 ,DNO=50 where SSN=4;
update Employee set Super_SSN=3 ,DNO=51 where SSN=5;
update Employee set Super_SSN=3 ,DNO=52 where SSN=6;
update Employee set Super_SSN=1 ,DNO=52 where SSN=7;
update Employee set Super_SSN=1 ,DNO=53 where SSN=8;
update Employee set Super_SSN=4 ,DNO=53 where SSN=9;
update Employee set Super_SSN=3 ,DNO=50 where SSN=10;
update Employee set Super_SSN=3 ,DNO=50 where SSN=11;
update Employee set Super_SSN=3 ,DNO=50 where SSN=12;
update Employee set Super_SSN=3 ,DNO=53 where SSN=13;

insert into Dlocation values(50,'Bangalore');
insert into Dlocation values(50,'New York');
insert into Dlocation values(50,'Delhi');
insert into Dlocation values(51,'Bangalore');
insert into Dlocation values(51,'Delhi');
insert into Dlocation values(52,'Bangalore');
insert into Dlocation values(52,'New York');
insert into Dlocation values(53,'Bangalore');
insert into Dlocation values(53,'Delhi');

insert into Project values(1000,'IOT','Bangalore',50);
insert into Project values(1001,'Web Development','Delhi',51);
insert into Project values(1002,'Embedded Systems','Bangalore',50);
insert into Project values(1003,'Hacking','New York',52);
insert into Project values(1004,'Cryptography','Delhi',51);
insert into Project values(1005,'Automata','Bangalore',53);
```

```
insert into Works_On values(1,1000,10);
insert into Works_On values(2,1000,8);
insert into Works_On values(3,1000,5);
insert into Works_On values(1,1001,6);
insert into Works_On values(2,1002,7);
insert into Works_On values(3,1005,8);
insert into Works_On values(4,1002,5);
insert into Works_On values(4,1003,4);
insert into Works_On values(5,1001,6);
insert into Works_On values(6,1004,10);
insert into Works_On values(7,1000,8);
insert into Works_On values(7,1003,6);
insert into Works_On values(8,1004,5);
insert into Works_On values(9,1002,7);
```

Queries

```
/* Q1 using union*/
select p.pno
from employee e, department d, project p
where p.dno=d.dno and d.manager_ssn=e.ssn and e.name='Scott'
union
select w.pno
from employee e, works_on w
where e.ssn=w.ssn and e.name='Scott'

/* Q2 */
select name, 1.1* salary as raised_salary
from employee e, project p, works_on w
where p.pname='IOT' and p.pno=w.pno and w.ssn=e.ssn
```

```
/* Q3 */

select sum(Salary) as sumsal, avg(Salary) as avgsal, min(Salary) as minsal, max(Salary) as
maxsal

from Employee e, Department d

where e.dno=d.dno and d.Dname='Accounts';


/* Q4 using  NOT Exists*/

select e.name

from Employee e

where not exists(

                    (select PNO from Project where DNO=5)

                     minus

                    (select PNO from Works_On where e.SSN=SSN)

                    );


/* Q5 */

select e.dno, count(*) as Employee_count

from employee e, department d

where e.dno=d.dno and salaray> 600000 and e.dno in ( select dno

                                                     from employee

                                                     group by dno

                                                    having count(*)>5)

group by e.dno ;
```

LAB PROGRAM 4

4. The following relations keep track of airline flight information:

FLIGHTS (FLNO: INTEGER, FROM: STRING, TO: STRING, DISTANCE: INTEGER, DEPARTS:TIME, ARRIVES: TIME, PRICE: INTEGER)

AIRCRAFT (AID: INTEGER, ANAME: STRING, CRUISINGRANGE: INTEGER)

CERTIFIED (EID: INTEGER, AID: INTEGER)

EMPLOYEES (EID: INTEGER, ENAME: STRING, SALARY: INTEGER)

Note that the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft, and only pilots are certified to fly.

Write each of the following queries in SQL.

a) Find the names of aircraft such that all pilots certified to operate them earn more than $80,000.
b) For each pilot who is certified for more than three aircraft, find the eid and the maximum cruisingrange of the aircraft for which she or he is certified.
c) For all aircraft with cruisingrange over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.
d) Print the enames of pilots who can operate planes with cruising range greater than 3000 miles but are not certified on any Boeing aircraft.
e) Print the name and salary of every nonpilot whose salary is more than the average salary for pilots.

Solution:

```
CREATE TABLE flight(
   no INT,
   frm VARCHAR(20),
   too VARCHAR(20),
   distance INT,
   departs VARCHAR(20),
   arrives VARCHAR(20),
   price REAL,
   PRIMARY KEY (no) );


CREATE TABLE aircraft(
   aid INT,
   aname VARCHAR(20),
   cruisingrange INT,
   PRIMARY KEY (aid) );


 CREATE TABLE employees(
   eid INT,
```

```
  ename VARCHAR(20),
  salary INT,
  PRIMARY KEY (eid) );

CREATE TABLE certified(
  eid INT,
  aid INT,
  PRIMARY KEY (eid,aid),
  FOREIGN KEY (eid) REFERENCES employees (eid),
  FOREIGN KEY (aid) REFERENCES aircraft (aid) );

INSERT INTO flight VALUES (1,'Bangalore','Mangalore',360,'10:45:00','12:00:00',10000);
INSERT INTO flight VALUES (2,'Bangalore','Delhi',5000,'12:15:00','04:30:00',25000);
INSERT INTO flight VALUES (3,'Bangalore','Mumbai',3500,'02:15:00','05:25:00',30000);
INSERT INTO flight VALUES (4,'Delhi','Mumbai',4500,'10:15:00','12:05:00',35000);
INSERT INTO flight VALUES (5,'Delhi','Frankfurt',18000,'07:15:00','05:30:00',90000);
INSERT INTO flight VALUES (6,'Bangalore','Frankfurt',19500,'10:00:00','07:45:00',95000);
INSERT INTO flight VALUES (7,'Bangalore','Frankfurt',17000,'12:00:00','06:30:00',99000);

INSERT INTO aircraft values (123,'Airbus',1000);
INSERT INTO aircraft values (302,'Boeing',5000);
INSERT INTO aircraft values (306,'Jet01',5000);
INSERT INTO aircraft values (378,'Airbus380',8000);
INSERT INTO aircraft values (456,'Aircraft',500);
INSERT INTO aircraft values (789,'Aircraft02',800);
INSERT INTO aircraft values (951,'Aircraft03',1000);

INSERT INTO employees VALUES (1,'Ajay',30000);
INSERT INTO employees VALUES (2,'Ajith',85000);
INSERT INTO employees VALUES (3,'Arnab',50000);
```

```
INSERT INTO employees VALUES (4,'Harry',45000);
INSERT INTO employees VALUES (5,'Ron',90000);
INSERT INTO employees VALUES (6,'Josh',75000);
INSERT INTO employees VALUES (7,'Ram',100000);
INSERT INTO employees VALUES (8,'Manoj',80000);

INSERT INTO certified VALUES (1,123);
INSERT INTO certified VALUES (2,123);
INSERT INTO certified VALUES (1,302);
INSERT INTO certified VALUES (5,302);
INSERT INTO certified VALUES (7,302);
INSERT INTO certified VALUES (1,306);
INSERT INTO certified VALUES (2,306);
INSERT INTO certified VALUES (1,378);
INSERT INTO certified VALUES (2,378);
INSERT INTO certified VALUES (4,378);
INSERT INTO certified VALUES (6,456);
INSERT INTO certified VALUES (3,456);
INSERT INTO certified VALUES (5,789);
INSERT INTO certified VALUES (6,789);
INSERT INTO certified VALUES (3,951);
INSERT INTO certified VALUES (1,951);
INSERT INTO certified VALUES (1,789);
```

Queries

```
/* Q1 */
select distinct aname
from aircraft a,employees e,certified c
where a.aid=c.aid and e.eid=c.eid and e.salary>80000;
```

```
/* Q2 */
select e.eid , max(cruisingrange)
from employees e, certified c, aircraft a
where e.eid=c.eid and a.aid=c.aid
group by e.eid
having count(*)>3;

/* Q3 */
select a.aname , avg(salary)
from employees e ,certified c, aircraft a
where c.eid=e.eid and a.aid=c.aid and cruisingrange>1000
group by a.aname;

/* Q4 */
select ename from employees
where eid in (select eid from certified where aid in
( select aid from aircraft where cruisingrange>=3000 and aname<>'boeing'));

/* Q5 */
select eid,ename,salary from employees
where salary> (select avg(salary) from employees e, certified c where e.eid = c.eid)
and eid not in (select eid from certified);
```

LAB PROGRAM 5

5. Consider the following relations:

STUDENT(SNUM: INTEGER, SNAME: STRING, MAJOR: STRING, LEVEL: STRING, AGE: INTEGER)

CLASS(NAME: STRING, MEETS AT: STRING, ROOM: STRING, FID: INTEGER)

ENROLLED(SNUM: INTEGER, CNAME: STRING)

FACULTY(FID: INTEGER, FNAME: STRING, DEPTID: INTEGER)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class.

Write the following queries in SQL. No duplicates should be printed in any of the answers.

a) Find the names of all Juniors (level = JR) who are enrolled in a class taught by Rakesh.
b) Find the age of the oldest student who is either a History major or enrolled in a course taught by Ravi.
c) Find the names of all students who are enrolled in two classes that meet at the same time.
d) For each faculty member that has taught classes only in room R128, print the faculty member's name and the total number of classes she or he has taught.
e) For each level, print the level and the average age of students for that level.

Solution:

```
CREATE TABLE student(
        snum INT primary key,
        sname VARCHAR(10),
        major VARCHAR(10),
        lvl VARCHAR(10),
        age int
);

CREATE TABLE faculty(
        fid INT primary key,
        fname VARCHAR(20),
        deptid INT
);

CREATE TABLE class(
        cname VARCHAR(20) primary key,
        meets_at VARCHAR(10),
        room VARCHAR(10),
        fid INT references faculty(fid) on delete cascade
);

CREATE TABLE enrolled(
```

```
        snum INT references student(snum) on delete cascade,
        cname VARCHAR(20) references class(cname) on delete cascade,
        PRIMARY KEY(snum,cname)
);

INSERT INTO student VALUES(1,'John','History','Sr',19);
INSERT INTO student VALUES(2,'Sakhi','History','Jr',20);
INSERT INTO student VALUES(3,'Jhanavi','Maths','Sr',20);
INSERT INTO student VALUES(4,'Tarun','History','Jr',20);
INSERT INTO student VALUES(5,'Sid','History','Jr',20);
INSERT INTO student VALUES(6,'Harsha','History','Sr',21);

INSERT INTO faculty VALUES(11,'Rakesh',1000);
INSERT INTO faculty VALUES(12,'Mohan',1000);
INSERT INTO faculty VALUES(13,'Kumar',1001);
INSERT INTO faculty VALUES(14,'Ravi',1002);
INSERT INTO faculty VALUES(15,'Shan',1000);

INSERT INTO class VALUES('class1','02:00:00','r128',14);
INSERT INTO class VALUES('class8','10:00:00','r128',14);
INSERT INTO class VALUES('class2','10:00:00','r2',12);
INSERT INTO class VALUES('class3','10:00:00','r3',11);
INSERT INTO class VALUES('class4','04:00:00','r128',14);
INSERT INTO class VALUES('class5','08:00:00','r3',15);
INSERT INTO class VALUES('class6','10:00:00','r128',14);
INSERT INTO class VALUES('class7','10:00:00','r128',14);

INSERT INTO enrolled VALUES(1,'class1');
INSERT INTO enrolled VALUES(2,'class1');
INSERT INTO enrolled VALUES(3,'class3');
```

```
INSERT INTO enrolled VALUES(4,'class3');

INSERT INTO enrolled VALUES(5,'class4');

INSERT INTO enrolled VALUES(1,'class5');

INSERT INTO enrolled VALUES(2,'class5');

INSERT INTO enrolled VALUES(3,'class5');

INSERT INTO enrolled VALUES(4,'class5');

INSERT INTO enrolled VALUES(5,'class5');

INSERT INTO enrolled VALUES(6,'class5');

INSERT INTO enrolled VALUES(5,'class2');

INSERT INTO enrolled VALUES(2,'class6');

INSERT INTO enrolled VALUES(3,'class6');

INSERT INTO enrolled VALUES(4,'class7');

INSERT INTO enrolled VALUES(5,'class3');
```

Queries

```
/* Q1 Using Single Block query*/

select s.sname

from student s, class c, enrolled e, faculty f

where f.fname='Rakesh' and f.fid=c.fid and c.cname=e.cname and e.snum=s.snum and
s.lvl='Jr';


/* Q1 Using Nested query*/

select sname from student

where lvl='Jr' and snum in ( select snum from enrolled

                             where cname in( select cname from class

                             where fid in ( select fid from faculty

                             where fname='Rakesh')));


/* Q2 Using Single Block query*/

select max(age)

from student s, class c, enrolled e, faculty f
```

```
where f.fid=c.fid and c.cname=e.cname and e.snum=s.snum and (major='History' OR
f.fname='Ravi');


/* Q2 Using Nested query*/
select max(s.age )
from student s
where major='History' OR snum in (select snum from enrolled
                                  where cname in(select cname from class
                                  where fid in (select fid from faculty
                                  where fname=’ravi’)));


/* Q3 */
select Distinct S.Sname from Student S where S.Snum in
(Select E1.Snum
from Enrolled E1,Enrolled E2,Class C1,Class C2
where E1.Cname=C1.Cname and E2.Cname=C2.Cname and E1.Snum=E2.Snum and
E1.Cname<>E2.Cname and C1.Meets_at=C2.Meets_at);


/* Nested Q4 */
Select Distinct F.Fname , count(*) as CourseCount
from CLass C , Faculty F
where C.Fid = F.Fid and C.FID not  in (Select Fid
                                       from Class
                                       where Room IN (Select Room
                                       from Class where Room != 'r128'))
group by F.Fname;
/* Q5 */
select lvl, avg(age)
from student
group by lvl;
```