



**B.M.S. COLLEGE OF ENGINEERING, BANGALORE-
19**

(Autonomous College under VTU)

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

**DATABASE MANAGEMENT SYSTEM LABORATORY
RECORD**

NAME: POTANA KUNDANA SAI PRIYA

USN: 1BM19CS112

PROGRAM: BACHELOR OF ENGINEERING

SEMESTER: IV

SESSION: APR-JUL 2021

COURSE CODE: 19CS4PCDBM

COURSE TITLE: DATABASE MANAGEMENT SYSTEM

CREDITS: 4

DBMS Lab List

Experiment #	Name of Experiment
1	Insurance Database
2	Banking Enterprise Database
3	Supplier Database
4	Student Faculty Database
5	Airline Flight Database
6	Order Processing Database
7	Book dealer Database
8	Student Enrolment Database
9	Movie Database
10	College Database

PROGRAM 1: INSURANCE DATABASE

Consider the insurance database given below. The primary keys are underlined and the data types are specified.

PERSON (driver-id#: string, name: string, address: string)

CAR (Regno#: string, model: string, year: int)

ACCIDENT (report-number: int, date: date, location: string)

OWNS (driver-id#: string, Regno: string)

PARTICIPATED (driver-id: string, Regno: string, report-number: int, damage-amount: int)

- a. Create the above tables by properly specifying the primary keys and the foreign keys

```
create table PERSON
(
    Driver_id varchar(60),
    Name varchar(60),
    Address varchar(80),
    primary key(Driver_id)
);
create table CAR
(
    Regno varchar(60),
    Model varchar(20),
    Year int,
    primary key(Regno)
);
create table ACCIDENT
(
    Report_number int,
    Date date,
    Location varchar(80),
    primary key(Report_number)
);
create table OWNS
(
    Driver_id varchar(60),
    Regno varchar(60),
    primary key(Driver_id,Regno),
    foreign key(Driver_id) references PERSON(Driver_id),
    foreign key(Regno) references CAR(Regno)
);
create table PARTICIPATED
(
    Driver_id varchar(60),
    Regno varchar(60),
    Report_number int,
    Damage_amount int,
```

```
primary key(Driver_id,Regno,Report_number),
foreign key(Driver_id) references PERSON(Driver_id),
foreign key(Regno) references CAR(Regno),
foreign key(Report_number) references ACCIDENT(Report_number)
);
```

b. Enter at least 5 tuples for each relation.

```
insert into PERSON(Driver_id,Name,Address) values('P1','Akshay','NR COLONY'),
('P2','Jyoti','JAYANAGAR'),
('P3','Radha','TYAGRAJNAGAR'),
('P5','Rajat','BANASANKARI'),
('P6','Jatin','BASVANGUDI');
```

```
insert into CAR(Regno, Model, Year) values('KA056TB','BMW',2019),
('KA456BT','SUZUKI',2001),
('KA467BT','HYUNDAI',2003),
('KA098BT','MAHINDRA SCORPIO',2009),
('KA648BT','TATA ALTROZ',2007);
```

```
insert into ACCIDENT(Report_number,Date,Location) values(096,'2020-04-14','MG ROAD'),
(045,'2009-07-28','DHARWAD'),
(235,'2003-07-22','HUBLI'),
(123,'2010-03-22','BASVANGUDI'),
(453,'2010-09-21','JAYANAGAR');
```

```
insert into OWNS(Driver_id,Regno) values('P1','KA456BT'),
('P6','KA467BT'),
('P5','KA467BT'),
('P3','KA098BT'),
('P2','KA648BT');
```

```
insert into PARTICIPATED(Driver_id,Regno,Report_number,Damage_amount)
values('P1','KA456BT',045,5000),
('P6','KA467BT',235,8900),
('P5','KA467BT',123,4000),
('P3','KA098BT',453,6500),
('P2','KA648BT',096,1200);
```

```
P1|Akshay|NR COLONY
P2|Jyoti|JAYANAGAR
P3|Radha|TYAGRAJNAGAR
P5|Rajat|BANASANKARI
P6|Jatin|BASVANGUDI
```

```
KA056TB | BMW | 2019
KA456BT | SUZUKI | 2001
KA467BT | HYUNDAI | 2003
KA098BT | MAHINDRA SCORPIO | 2009
KA648BT | TATA ALTROZ | 2007
```

```
96 | 2020-04-14 | MG ROAD
45 | 2009-07-28 | DHARWAD
235 | 2003-07-22 | HUBLI
123 | 2010-03-22 | BASVANGUDI
453 | 2010-09-21 | JAYANAGAR
```

```
P1 | KA456BT
P2 | KA648BT
P3 | KA098BT
P5 | KA467BT
P6 | KA467BT
```

```
P1 | KA456BT | 45 | 5000
P6 | KA467BT | 235 | 8900
P5 | KA467BT | 123 | 4000
P3 | KA098BT | 453 | 6500
P2 | KA648BT | 96 | 1200
```

- c. Demonstrate how you
- Update the damage amount for the car with a specific Regno in the accident with report number 12 to 25000.

update PARTICIPATED set Damage_amount=25000 where Report_number=235 and Regno='KA467BT';

```
P1 | KA456BT | 45 | 5000
P6 | KA467BT | 235 | 25000
P5 | KA467BT | 123 | 4000
P3 | KA098BT | 453 | 6500
P2 | KA648BT | 96 | 1200
```

- Add a new accident to the database.

insert into ACCIDENT values(002,'2021-02-12','Jalahalli');

```
96 | 2020-04-14 | MG ROAD  
45 | 2009-07-28 | DHARWAD  
235 | 2003-07-22 | HUBLI  
123 | 2010-03-22 | BASVANGUDI  
453 | 2010-09-21 | JAYANAGAR  
2 | 2021-02-12 | Jalahalli
```

- d. Find the total number of people who owned cars that involved in accidents in 2010.

```
select count(*) from ACCIDENT where date between '2010-01-01' and '2010-12-31';
```

```
2
```

- e. Find the number of accidents in which cars belonging to a specific model were involved.

```
select count(*) from PARTICIPATED join CAR on CAR.Regno=PARTICIPATED.Regno where  
CAR.Model="TATA ALTROZ";
```

```
1
```

PROGRAM 2: BANKING ENTERPRISE DATABASE

2. Consider the following database for a banking enterprise.

BRANCH (branch-name#: string, branch-city: string, assets: real)

ACCOUNTS (accno: int, branch-name: string, balance: real)

DEPOSITOR (customer-name#: string, customer-street: string, customer-city: string)

LOAN (loan-number#: int, branch-name: string, amount: real)

BORROWER (customer-name: string, loan-number: int)

- a. Create the above tables by properly specifying the primary keys and the foreign keys

```
create table BRANCH
(
  branch_name varchar(60),
  branch_city varchar(60),
  assets real,
  primary key(branch_name)
);
create table ACCOUNTS
(
  accno int,
  branch_name varchar(60),
  balance real,
  primary key(accno,branch_name),
  foreign key(branch_name) references BRANCH(branch_name)on delete cascade on update
  cascade
);
create table DEPOSITOR
(
  customer_name varchar(60),
  accno int,
  primary key(customer_name,accno),
  foreign key(accno) references ACCOUNTS(accno)on delete cascade on update cascade
);
create table CUSTOMER
(
  customer_name varchar(60),
  customer_street varchar(60),
  customer_city varchar(60),
  primary key(customer_name),
  foreign key(customer_name) references DEPOSITOR(customer_name)on delete cascade on
  update cascade
);
create table LOAN
(
```

```

loan_number int,
branch_name varchar(60),
amount real,
primary key(loan_number,branch_name),
foreign key(branch_name) references BRANCH(branch_name)
);
create table BORROWER
(
customer_name varchar(60),
loan_number int,
primary key(customer_name,loan_number),
foreign key(customer_name) references DEPOSITOR(customer_name) on delete cascade on
update cascade,
foreign key(loan_number) references LOAN(loan_number) on delete cascade on update
cascade
);

```

- b. Enter at least 5 tuples for each relation.

```

insert into BRANCH values
('ICICI','DHARWAD',50000),
('SBI','BANGALORE',45000),
('SBI-BASVANGUDI','BANGALORE',37000),
('ALLAHABADH','HUBLI',15000),
('KARNATAKA-BANK','BELGAUM',90000);

```

```

insert into ACCOUNTS values
(12345,'ALLAHABADH',4000),
(67890,'SBI-BASVANGUDI',7500),
(34567,'KARNATAKA-BANK',9000),
(12389,'ICICI',5600),
(49389,'ICICI',1780),
(34567,'ICICI',5000);

```

```

insert into DEPOSITOR values
('KRISHNA',12345),
('RADHA',12389),
('BALRAM',34567),
('AKSHAY',49389),
('PALLAVI',67890),
('AKSHAY',34567);

```

```

insert into CUSTOMER values
('KRISHNA','NR COLONY','BANGALORE'),
('RADHA','BANSANKARI','BANGALORE'),
('BALRAM','VIDYANAGAR','DHARWAD'),
('AKSHAY','RAMNAGAR','HUBLI'),
('PALLAVI','RAMNAGAR','HUBLI');

```


insert into LOAN values
(1,'ALLAHABADH',56000),
(2,'ICICI',47890),
(3,'KARNATAKA-BANK',45000),
(4,'SBI',45000),
(5,'SBI',20000);

insert into BORROWER values
('KRISHNA',4),
('RADHA',5),
('PALLAVI',5),
('AKSHAY',1),
('BALRAM',3);

ICICI | DHARWAD | 50000.0
SBI | BANGALORE | 45000.0
SBI - BASVANGUDI | BANGALORE | 37000.0
ALLAHABADH | HUBLI | 15000.0
KARNATAKA-BANK | BELGAUM | 90000.0

12345 | ALLAHABADH | 4000.0
67890 | SBI - BASVANGUDI | 7500.0
34567 | KARNATAKA-BANK | 9000.0
12389 | ICICI | 5600.0
49389 | ICICI | 1780.0

AKSHAY | 34567
AKSHAY | 49389
BALRAM | 34567
KRISHNA | 12345
PALLAVI | 67890
RADHA | 12389

1 | ALLAHABADH | 56000.0
2 | ICICI | 47890.0
3 | KARNATAKA-BANK | 45000.0
4 | SBI | 45000.0
5 | SBI | 20000.0

AKSHAY | 1
BALRAM | 3
KRISHNA | 4
PALLAVI | 5
RADHA | 5

- c. Find all the customers who have at least two accounts at the main branch.

```
select customer_name from DEPOSITOR join ACCOUNTS on  
DEPOSITOR.accno=ACCOUNTS.accno where ACCOUNTS.branch_name='ICICI' group by  
DEPOSITOR.customer_name having count(DEPOSITOR.customer_name)>=2;
```

AKSHAY

- d. Find all the customers who have an account at all the branches located in a specific city.

```
select customer_name from DEPOSITOR join ACCOUNTS on  
ACCOUNTS.accno=DEPOSITOR.accno join BRANCH on  
BRANCH.branch_name=ACCOUNTS.branch_name where BRANCH.branch_city='DHARWAD'  
GROUP BY DEPOSITOR.customer_name  
having count(DISTINCT BRANCH.branch_name)=(SELECT COUNT(branch_name) FROM  
BRANCH where branch_city='DHARWAD');
```

RADHA

- e. Demonstrate how you delete all account tuples at every branch located in a specific city.

```
DELETE FROM ACCOUNTS WHERE branch_name in (select branch_name from BRANCH where  
branch_city='BELGAUM');
```

```
12345|ALLAHABADH|4000.0  
67890|SBI-BASVANGUDI|7500.0  
12389|ICICI|5600.0  
49389|ICICI|1780.0  
34567|ICICI|5000.0
```

PROGRAM 3: SUPPLIER DATABASE

3. Consider the following schema:

SUPPLIERS (sid#: int, sname: string, address: string)

PARTS (pid#: int, pname: string, color: string)

CATALOG (sid: int, pid: int, cost:real)

```
create table suppliers
(
sid integer primary key,
sname varchar(20),
address varchar(50)
);
create table parts
(
pid integer primary key,
pname varchar(20),
color varchar(10)
);
create table catalog
(
sid integer,
pid integer,
cost real,
primary key(sid,pid),
foreign key(sid) references suppliers(sid) on delete cascade on update cascade,
foreign key(pid) references parts(pid) on delete cascade on update cascade
);
```

```
insert into suppliers(sid,sname,address) VALUES
(001,'Rohan','Mangalore'),
(002,'Avni','Bangalore'),
(003,'Pratibha','Bagalkot'),
(004,'Rahul','Udupi'),
(005,'Prithvi','Hassan');
```

```
insert into parts(pid,pname,color) VALUES
(001,'Pipe','white'),
(002,'Screw','red'),
(003,'Nail','black'),
(004,'Tap','grey'),
(005,'bottle','red'),
(006,'plywood','brown');
```

```
insert into catalog(sid,pid,cost) VALUES
(001,001,50.00),
```

(001,006,120.00),
(002,002,75),
(002,005,100),
(003,002,45),
(003,003,75),
(004,001,140),
(004,002,38),
(004,003,42),
(004,004,310),
(004,005,79),
(004,006,110),
(005,002,50),
(005,003,48);

select * from suppliers;
select * from parts;
select * from catalog;

1|Rohan|Mangalore
2|Avni|Bangalore
3|Pratibha|Bagalkot
4|Rahul|Udupi
5|Prithvi|Hassan

1|Pipe|white
2|Screw|red
3|Nail|black
4|Tap|grey
5|bottle|red
6|plywood|brown

1|1|50.0
1|6|120.0
2|2|75.0
2|5|100.0
3|2|45.0
3|3|75.0
4|1|140.0
4|2|38.0
4|3|42.0
4|4|310.0
4|5|79.0
4|6|110.0
5|2|50.0
5|3|48.0

The catalog relation lists the prices charged for parts by suppliers. Write the following queries in SQL:

- a) Find the pnames of parts for which there is some supplier.

```
select distinct parts.pname from parts,catalog where parts.pid = catalog.pid;
```

```
Pipe  
plywood  
Screw  
bottle  
Nail  
Tap
```

- b) Find the snames of suppliers who supply every part.

```
select sname from suppliers s,catalog c where s.sid=c.sid group by s.sid,sname having  
count(pid)=6;
```

```
Rahul
```

- c) Find the snames of suppliers who supply every red part.

```
select s.sname from suppliers s where s.sid in (select c.sid from catalog c,parts p where  
c.pid=p.pid and p.color='red' group by c.sid having count(c.pid)=(select count(*) from parts p  
where p.color='red'));
```

```
Avni  
Rahul
```

- d) Find the pnames of parts supplied by Rahul and no one else.

```
SELECT P.pname FROM Parts P, Catalog C, Suppliers S WHERE P.pid = C.pid AND C.sid = S.sid  
AND S.sname = "Rahul" AND NOT EXISTS ( SELECT * FROM Catalog C1, Suppliers S1 WHERE  
P.pid = C1.pid AND C1.sid = S1.sid AND S1.sname <>"Rahul" );
```

```
Tap
```

- e) Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

```
SELECT DISTINCT C.sid FROM Catalog C WHERE C.cost > ( SELECT AVG (C1.cost) FROM Catalog  
C1 WHERE C1.pid = C.pid );
```

```
1  
2  
3  
4
```

- f) For each part, find the sname of the supplier who charges the most for that part.

```
SELECT P.pid, S.sname FROM Parts P, Suppliers S, Catalog C WHERE C.pid = P.pid AND C.sid =  
S.sid AND C.cost = (SELECT MAX(C1.cost) FROM Catalog C1 WHERE C1.pid = P.pid);
```

```
6|Rohan  
2|Avni  
5|Avni  
3|Pratibha  
1|Rahul  
4|Rahul
```

- g) Find the sid of suppliers who supply only red parts.

```
select s.sid from suppliers s where s.sid in(select c.sid  
from catalog c where c.sid not in (select distinct(ca.sid)  
from catalog ca,parts p where ca.pid=p.pid and p.color!='red'));
```

```
2
```

PROGRAM 4: STUDENT FACULTY DATABASE

4. Consider the following database for student enrollment for course:

STUDENT (snum#: int, sname: string, major: string, level: string, age:int)

CLASS (name: string, meets-at: time, room: string, fid: int)

ENROLLED (snum: int, cname: string)

FACULTY (fid#: int, fname: string, deptid: int)

```
create table student(snum int,sname varchar(60),major varchar(60),level varchar(6),
    age int,primary key(snum));
```

```
create table faculty(fid int,fname varchar(60),deptid int,primary key(fid));
```

```
create table class(cname varchar(60),meets_at timestamp,room varchar(60),fid int,
    primary key(cname),foreign key(fid) references faculty(fid));
```

```
create table enrolled(snum int,cname varchar(60),primary key(snum,cname),foreign key(cname)
    references class(cname),foreign key(snum) references student(snum));
```

```
insert into student values(1,'john','cs','sr',19),
    (2,'smith','cs','jr',20),(3,'jacob','cv','sr',20),(4,'tom','cs','jr',20),
    (5,'rahul','cs','jr',20),(6,'rita','cs','sr',21);
```

```
insert into faculty values(11,'harish',1000),(12,'manav',1000),
    (13,'mira',1001),(14,'shiva',1002),(15,'nupur',1000);
```

```
insert into class values('class1','12/11/15 10:15:16','R1',14),
    ('class10','12/11/15 10:15:16','R128',14),
    ('class2','12/11/15 10:15:20','R2',12),
    ('class3','12/11/15 10:15:25','R3',12),
    ('class4','12/11/15 10:15:20','R4',14),
    ('class5','12/11/15 20:15:20','R3',15),
    ('class6','12/11/15 13:20:20','R2',14),
    ('class7','12/11/15 10:10:10','R3',14);
```

```
insert into enrolled values(1,'class1'),(2,'class1'),(3,'class3'),
    (4,'class3'),(5,'class4'),(1,'class5'),(2,'class5'),(3,'class5'),
    (4,'class5'),(5,'class5');
```

```
select*from student;
```

```
select*from class;
```

```
select * from enrolled;
```

```
select*from faculty;
```

Enrolled has one record per student-class pair such that the student is enrolled in the class. Level is a 2 character code with 4 different values.

- a) Find the names of all juniors (level=JR) who are enrolled in a class taught by

```
select distinct s.sname from student s,class c,enrolled e,faculty f where s.snum=e.snum and e.cname=c.cname and c.fid=f.fid and f.fname='shiva' and s.level='jr';
```

```
smith
rahul
```

- b) Find the names of all classes that either meet in room R128 or have 5 or more students enrolled.

```
select c.cname from class c where c.room='R128' or c.cname in(select e.cname from enrolled e,class c where c.cname=e.cname group by e.cname having count(*)>=5);
```

```
class10
class5
```

- c) Find the names of all students who are enrolled in 2 classes that meet at the same time.

```
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.snum=e2.snum and e1.cname<>e2.cname and e1.cname=c1.cname and e2.cname=c2.cname and c1.meets_at=c2.meets_at);
```

- d) Find the names of faculty members who teach in every room in which some class is taught

```
select f.fname from faculty f where f.fid in(select fid from class group by fid having count(*)=(select count(distinct room)from class));
```

```
shiva
```

- e) Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than 5

```
select distinct f.fname from faculty f where 5>(select count(e.snum) from class c,enrolled e where c.cname=e.cname and c.fid=f.fid);
```

```
harish
manav
mira
shiva
```

- f) Find the names of students who are not enrolled in any class.

`select distinct s.sname from student s where s.snum not in (select e.snum from enrolled e);`

```

sname
-----
rita

```

- g) For each age value that appears in students, find the level value that appears most often.

`select s.age,s.level from student s group by s.age,s.level having s.level in(select s1.level from student s1 where s1.age=s.age group by s1.level,s1.age having count(*)>=all(select count(*) from student s2 where s1.age=s2.age group by s2.level,s2.age));`

```

+-----+-----+
| age  | level |
+-----+-----+
| 19   | sr    |
| 20   | jr    |
| 21   | sr    |
+-----+-----+

```

PROGRAM 5: AIRLINE FLIGHT DATABASE

5. Consider the following database that keeps 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)

CREATE TABLE FLIGHTS

```
(
  FLNO INTEGER PRIMARY KEY,
  FFROM VARCHAR(15) NOT NULL,
  TTO VARCHAR(15) NOT NULL,
  DISTANCE INTEGER,
  DEPARTS TIMESTAMP,
  ARRIVES TIMESTAMP,
  PRICE REAL
);
```

CREATE TABLE AIRCRAFT

```
(
  AID INTEGER PRIMARY KEY,
  ANAME VARCHAR(10),
  CRUISINGRANGE INTEGER
);
```

CREATE TABLE EMPLOYEES

```
(
  EID INTEGER PRIMARY KEY,
  ENAME VARCHAR(15),
  SALARY REAL(10,2)
);
```

CREATE TABLE CERTIFIED

```
(
  EID INTEGER NOT NULL,
  AID INTEGER NOT NULL,
  PRIMARY KEY (EID, AID),
  FOREIGN KEY (EID) REFERENCES EMPLOYEES(EID),
  FOREIGN KEY (AID) REFERENCES AIRCRAFT(AID)
);
```

```
insert into aircraft values(101,'747',3000),(102,'Boeing',900),
(103,'647',800),(104,'Dreamliner',10000),(105,'Boeing',3500),
(106,'707',1500),(107,'Dream', 120000);
```

```
insert into employees values (701,'A',50000),(702,'B',100000),
(703,'C',150000),(704,'D',90000),(705,'E',40000),(706,'F',60000),(707,'G',90000);
```

```
insert into certified values(701,101),(701,102),(701,106),(701,105),(702,104),(703,104),
(704,104),(702,107),(703,107),(704,107),(702,101),(703,105),(704,105),(705,103);
```

```
insert into flights values(101,'Bangalore','Delhi',2500,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 17:15:31',5000);
```

```
insert into flights values(102,'Bangalore','Lucknow',3000,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 11:15:31',6000);
```

```
insert into flights values(103,'Lucknow','Delhi',500,TIMESTAMP '2005-05-13 12:15:31',TIMESTAMP '2005-05-13 17:15:31',3000);
```

```
insert into flights values(107,'Bangalore','Frankfurt',8000,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 22:15:31',60000);
```

```
insert into flights values(104,'Bangalore','Frankfurt',8500,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 23:15:31',75000);
```

```
insert into flights values(105,'Kolkata','Delhi',3400, TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 09:15:31',7000);
```

```
select * from flights;  
select * from aircraft;  
select * from certified;  
select * from EMPLOYEES;
```

```
101|747|3000  
102|Boeing|900  
103|647|800  
104|Dreamliner|10000  
105|Boeing|3500  
106|707|1500  
107|Dream|120000
```

```

701 | 101
701 | 102
701 | 105
701 | 106
702 | 101
702 | 104
702 | 107
703 | 104
703 | 105
703 | 107
704 | 104
704 | 105
704 | 107
705 | 103

701 | A | 50000.0
702 | B | 100000.0
703 | C | 150000.0
704 | D | 90000.0
705 | E | 40000.0
706 | F | 60000.0
707 | G | 90000.0

```

FLNO	FFROM	TTO	DISTANCE	DEPARTS	ARRIVES	PRICE
101	Bangalore	Delhi	2500	2005-05-13 07:15:31	2005-05-13 17:15:31	5000
102	Bangalore	Lucknow	3000	2005-05-13 07:15:31	2005-05-13 11:15:31	6000
103	Lucknow	Delhi	500	2005-05-13 12:15:31	2005-05-13 17:15:31	3000
104	Bangalore	Frankfurt	8500	2005-05-13 07:15:31	2005-05-13 23:15:31	75000
105	Kolkata	Delhi	3400	2005-05-13 07:15:31	2005-05-13 09:15:31	7000
107	Bangalore	Frankfurt	8000	2005-05-13 07:15:31	2005-05-13 22:15:31	60000

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 have salaries more than Rs.80,000.

```

SELECT DISTINCT A.aname FROM Aircraft A
WHERE A.Aid IN (SELECT C.aid FROM Certified C, Employees E
WHERE C.aid = E.aid AND NOT EXISTS ( SELECT * FROM Employees E1
WHERE E1.aid = E.aid AND E1.salary < 80000 ));

```

aname
747
Dreamliner
Boeing
Dream

- b) For each pilot who is certified for more than 3 aircrafts, find the eid and the maximum cruising range of the aircraft for which the person is certified.

```
SELECT C.eid, MAX(A.cruisingrange) FROM Certified C, Aircraft A
WHERE C.aid = A.aid
GROUP BY C.eid HAVING COUNT(*) > 3 ;
```

eid	MAX(A.cruisingrange)
701	3500

- c) Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.

```
SELECT DISTINCT E.ename FROM Employees E
WHERE E.salary < ( SELECT MIN(F.price) FROM Flights F
WHERE F.ffrom = 'Bangalore' AND F.tto = 'Frankfurt' );
```

ename
A
E

- d) For all aircraft with cruisingrange over 1000 Kms, find the name of the aircraft and the average salary of all pilots certified for this aircraft.

```
SELECT Temp.name, Temp.AvgSalary
FROM ( SELECT A.aid, A.aname AS name, AVG (E.salary) AS AvgSalary
FROM Aircraft A, Certified C, Employees E
WHERE A.aid = C.aid AND C.eid = E.eid AND A.cruisingrange > 1000
GROUP BY A.aid, A.aname ) AS Temp;
```

name	AvgSalary
747	75000.000000
Dreamliner	113333.333333
Boeing	96666.666667
707	50000.000000
Dream	113333.333333

- e) Find the names of pilots certified for some Boeing aircraft.

```
SELECT DISTINCT E.ename
FROM Employees E, Certified C, Aircraft A
WHERE E.eid = C.eid AND C.aid = A.aid AND A.aname LIKE 'Boeing%';
```

ename
A
C
D

- f) Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

```
SELECT A.aid FROM Aircraft A
WHERE A.cruisingrange > ( SELECT MIN(F.distance) FROM Flights F
WHERE F.ffrom = 'Bangalore' AND F.tto = 'Delhi');
```

aid
101
104
105
107

- g) A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.

```
SELECT F.departs,F.flno FROM Flights F
WHERE F.flno IN (( SELECT F0.flno FROM Flights F0
WHERE F0.ffrom = 'Bangalore' AND F0.tto = 'Delhi' AND extract(hour from F0.arrives) < 18)
UNION
(SELECT F0.flno FROM Flights F0, Flights F1
WHERE F0.ffrom = 'Bangalore' AND F0.tto <> 'Delhi'
AND F0.tto = F1.ffrom AND F1.tto = 'Delhi')
```

```
AND F1.departs > F0.arrives AND extract(hour from F1.arrives) < 18 )
UNION
(SELECT F0.flno FROM Flights F0, Flights F1, Flights F2
WHERE F0.ffrom = 'Bangalore' AND F0.tto = F1.ffrom
AND F1.tto = F2.ffrom AND F2.tto = 'Delhi'
AND F0.tto <> 'Delhi' AND F1.tto <> 'Delhi'
AND F1.departs > F0.arrives AND F2.departs > F1.arrives AND extract(hour from F2.arrives) < 18 ));
```

departs	flno
2005-05-13 07:15:31	101
2005-05-13 07:15:31	102

PROGRAM 6: ORDER DATABASE

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*)

```
CREATE DATABASE ORDER;
```

```
CREATE TABLE SALESMAN
```

```
(  
  SALESMAN_ID int,  
  NAME VARCHAR (20),  
  CITY VARCHAR (20),  
  COMMISSION VARCHAR (20),  
  PRIMARY KEY (SALESMAN_ID)  
);
```

```
CREATE TABLE CUSTOMER
```

```
(  
  CUSTOMER_ID INT,  
  CUST_NAME VARCHAR (20),  
  CITY VARCHAR (20),  
  GRADE INT (3),  
  SALESMAN_ID int,  
  PRIMARY KEY (CUSTOMER_ID),  
  FOREIGN KEY (SALESMAN_ID) REFERENCES SALESMAN (SALESMAN_ID) ON DELETE SET NULL  
);
```



```
CREATE TABLE ORDERS
(
ORD_NO INT,
PURCHASE_AMT INT,
ORD_DATE DATE,
CUSTOMER_ID INT,
SALESMAN_ID INT,
PRIMARY KEY (ORD_NO),
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID) ON DELETE CASCADE,
FOREIGN KEY (SALESMAN_ID) REFERENCES SALESMAN (SALESMAN_ID) ON DELETE CASCADE
);
```

```
INSERT INTO SALESMAN VALUES (1000, 'JOHN', 'BANGALORE', '25 %'),
(2000, 'RAVI', 'BANGALORE', '20 %'),
(3000, 'KUMAR', 'MYSORE', '15 %'),
(4000, 'SMITH', 'DELHI', '30 %'),
(5000, 'HARSHA', 'HYDRABAD', '15%');
```

```
INSERT INTO CUSTOMER VALUES (10, 'PREETHI','BANGALORE', 100, 1000),  
(11, 'VIVEK','MANGALORE', 300, 1000),  
(12, 'BHASKAR','CHENNAI', 400, 2000),  
(13, 'CHETHAN','BANGALORE', 200, 2000),  
(14, 'MAMATHA','BANGALORE', 400, 3000);
```

```
INSERT INTO ORDERS VALUES (50, 5000, '2017-04-04', 10, 1000),  
(51, 450, '20-JAN-17', 10, 2000),  
(52,1000,'24-FEB-17',13,2000),  
(53,3500,'13-APR-17',14,3000),  
(54, 550, '09-MAR-17', 12, 2000);
```

Write SQL queries to

1. Count the customers with grades above Bangalore's average.

```
SELECT GRADE, COUNT (DISTINCT CUSTOMER_ID)
FROM CUSTOMER
GROUP BY GRADE
HAVING GRADE > (SELECT
AVG(GRADE) FROM CUSTOMER
WHERE CITY='BANGALORE');
```

Showing rows 0 - 1 (2 total, Query took 0.0162 seconds.)

```
SELECT GRADE, COUNT(DISTINCT CUSTOMER_ID) FROM CUSTOMER GROUP BY GRADE HAVING GRADE > (SELECT AVG(GRADE) FROM CUSTOMER WHERE CITY='BANGALORE')
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: 25 Filter rows:

+ Options

GRADE	COUNT(DISTINCT CUSTOMER_ID)
300	1
400	2

2. Find the name and numbers of all salesmen who had more than one customer.

```
SELECT SALESMAN_ID, NAME
FROM SALESMAN S
WHERE (SELECT COUNT
(*) FROM CUSTOMER C
WHERE C. SALESMAN_ID=S.SALESMAN_ID) > 1;
```

Showing rows 0 - 1 (2 total, Query took 0.0044 seconds.)

```
SELECT SALESMAN_ID, NAME FROM SALESMAN S WHERE (SELECT COUNT(*) FROM CUSTOMER C WHERE C.SALESMAN_ID=S.SALESMAN_ID) > 1
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

				SALESMAN_ID	NAME
<input type="checkbox"/>	Edit	Copy	Delete	1000	JOHN
<input type="checkbox"/>	Edit	Copy	Delete	2000	RAVI

3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)

```
SELECT S.SALESMAN_ID, S.NAME, C.CUST_NAME,
S.COMMISSION FROM SALESMAN S, CUSTOMER C
WHERE
S.CITY=C.CITY UNION
SELECT S.SALESMAN_ID,S.NAME,'NO MATCH',S.COMMISSION
FROM SALESMAN S
WHERE CITY NOT IN
(SELECT CITY
FROM
CUSTOMER)
ORDER BY 1 ASC;
```

Showing rows 0 - 8 (9 total, Query took 0.0230 seconds.)

```
SELECT S.SALESMAN_ID, S.NAME, C.CUST_NAME, S.COMMISSION FROM SALESMAN S, CUSTOMER C WHERE S.CITY=C.CITY UNION SELECT S.SALESMAN_ID,S.NAME,'NO MATCH',S.COMMISSION FROM SALESMAN S WHERE CITY NOT IN (SELECT CITY FROM CUSTOMER) ORDER BY 1 ASC
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

SALESMAN_ID	NAME	CUST_NAME	COMMISSION
1000	JOHN	PREETHI	25 %
1000	JOHN	CHETHAN	25 %
1000	JOHN	MAMATHA	25 %
2000	RAVI	PREETHI	20 %
2000	RAVI	CHETHAN	20 %
2000	RAVI	MAMATHA	20 %
3000	KUMAR	NO MATCH	15 %
4000	SMITH	NO MATCH	30 %
5000	HARSHA	NO MATCH	15 %

4. Create a view that finds the salesman who has the customer with the highest order of a day.

```
CREATE VIEW V_SALESMAN AS SELECT O.ORDER_DATE, S.SALESMAN_ID, S.NAME FROM  
SALESMAN S, ORDERS O WHERE S.SALESMAN_ID = O.SALESMAN_ID AND  
O.PURCHASE_AMOUNT = (SELECT MAX(PURCHASE_AMOUNT) FROM ORDERS C  
WHERE C.ORDER_DATE = O.ORDER_DATE);
```

				ORD_DATE	SALESMAN_ID	NAME
<input type="checkbox"/>	 Edit	 Copy	 Delete	2017-05-04	1000	JOHN
<input type="checkbox"/>	 Edit	 Copy	 Delete	2017-01-20	2000	RAVI
<input type="checkbox"/>	 Edit	 Copy	 Delete	2017-02-24	2000	RAVI
<input type="checkbox"/>	 Edit	 Copy	 Delete	2017-04-13	3000	KUMAR
<input type="checkbox"/>	 Edit	 Copy	 Delete	2017-03-09	2000	RAVI

5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

```
DELETE FROM SALESMAN WHERE SALESMAN_ID=1000;
```

✓ 1 row affected. (Query took 0.0207 seconds.)

DELETE FROM SALESMAN WHERE SALESMAN_ID=1000

[Edit inline] [Edit] [Create PHP code]

PROGRAM 7: BOOK DEALER DATABASE

The following tables are maintained by a book dealer:

AUTHOR(author-id: int, name: String, city: String, country: String)
PUBLISHER(publisher-id: int, name: String, city: String, country: String)
CATALOG (book-id: int, title: String, author-id: int, publisher-id: int, category-id: int, year: int, price: int)
CATEGORY(category-id: int, description: String)
ORDER-DETAILS(order-no: int, book-id: int, quantity: int)

i. Create the above tables by properly specifying the primary keys and the foreign keys.

```
-- create a database
create database BOOK_DEALER;
use BOOK_DEALER;
-- create a table
CREATE TABLE AUTHOR
(
  AUTHORID INT,
  NAME VARCHAR (15),
  CITY VARCHAR (15),
  COUNTRY VARCHAR (15),
  PRIMARY KEY(AUTHORID)
);
desc AUTHOR;
```

Field	Type	Null	Key	Default	Extra
AUTHORID		int	NO	PRI	NULL
NAME	varchar(15)		YES		NULL
CITY	varchar(15)		YES		NULL
COUNTRY	varchar(15)		YES		NULL

```
-- create a table
CREATE TABLE PUBLISHER
(
  PUBLISHERID INT,
  NAME VARCHAR(15),
  CITY VARCHAR(15),
  COUNTRY VARCHAR(15),
  PRIMARY KEY(PUBLISHERID)
);
desc PUBLISHER;
```

Field	Type	Null	Key	Default	Extra
PUBLISHERID		int	NO	PRI	NULL
NAME	varchar(15)		YES		NULL
CITY	varchar(15)		YES		NULL
COUNTRY	varchar(15)		YES		NULL

```
-- create a table
CREATE TABLE CATEGORY
(
CATEGORYID INT,
DESCRIPTION VARCHAR(15),
PRIMARY KEY(CATEGORYID)
);
desc CATEGORY;
```

Field	Type	Null	Key	Default	Extra
CATEGORYID	int	NO	PRI	NULL	
DESCRIPTION	varchar(15)	YES			NULL

```
-- create a table
CREATE TABLE CATALOG
(
BOOKID INT,
TITLE VARCHAR(15),
AUTHORID INT,
PUBLISHERID INT,
CATEGORYID INT,
YEAR INT,
PRICE INT,
PRIMARY KEY(BOOKID),
FOREIGN KEY(AUTHORID) REFERENCES AUTHOR(AUTHORID),
FOREIGN KEY(PUBLISHERID) REFERENCES PUBLISHER(PUBLISHERID),
FOREIGN KEY(CATEGORYID) REFERENCES CATEGORY(CATEGORYID)
);
desc CATALOG;
```

Field	Type	Null	Key	Default	Extra
BOOKID	int	NO	PRI	NULL	
TITLE	varchar(15)	YES			NULL
AUTHORID	int	YES		MUL	NULL
PUBLISHERID	int	YES		MUL	NULL
CATEGORYID	int	YES		MUL	NULL
YEAR	int	YES		NULL	
PRICE	int	YES		NULL	

```
-- create a table
CREATE TABLE ORDER_DETAILS
(
ORDERNO INT,
BOOKID INT,
QUANTITY INT,
PRIMARY KEY(ORDERNO,BOOKID),
FOREIGN KEY(BOOKID) REFERENCES CATALOG(BOOKID)
);
desc ORDER_DETAILS;
```

Field	Type	Null	Key	Default	Extra
ORDERNO	int	NO	PRI	NULL	
BOOKID	int	NO	PRI	NULL	
QUANTITY	int	YES			NULL

ii. Enter at least five tuples for each relation.

-- insert values into AUTHOR

INSERT INTO AUTHOR VALUES

(101,'ABC','DELHI','INDIA'),
 (102,'TONY','HAYHOOD','USA'),
 (103,'GHI','PATNA','INDIA'),
 (104,'JKL','BELM','SRILANKA'),
 (105,'MND','BANGALORE','INDIA');
 SELECT * FROM AUTHOR;

AUTHORID	NAME	CITY	COUNTRY
101	ABC	DELHI	INDIA
102	TONY	HAYHOOD	USA
103	GHI	PATNA	INDIA
104	JKL	BELM	SRILANKA
105	MND	BANGALORE	INDIA

-- insert values into PUBLISHER

INSERT INTO PUBLISHER VALUES

(1001,'POOJA','BANGALORE','INDIA'),
 (1002,'PALAK','DELHI','INDIA'),
 (1003,'PRANAV','MUMBAI','INDIA'),
 (1004,'RAM','RANCHI','INDIA'),
 (1005,'ROHAN','VADODRA','INDIA');
 SELECT * FROM PUBLISHER;

PUBLISHERID	NAME	CITY	COUNTRY
1001	POOJA	BANGALORE	INDIA
1002	PALAK	DELHI	INDIA
1003	PRANAV	MUMBAI	INDIA
1004	RAM	RANCHI	INDIA
1005	ROHAN	VADODRA	INDIA

-- insert values into CATALOG

INSERT INTO CATALOG VALUES

(1000001,'dbms',101,1001,10001,1998,235),
 (1000002,'or',101,1002,10003,1997,255),
 (1000003,'cn',102,1003,10002,2001,352),
 (1000004,'se',102,1003,10001,2002,523),
 (1000005,'ada',103,1004,10004,2003,124);
 SELECT * FROM CATALOG;

BOOKID	TITLE	AUTHORID	PUBLISHERID	CATEGORYID	YEAR	PRICE
1000001	dbms	101	1001	10001	1998	235
1000002	or	101	1002	10003	1997	255
1000003	cn	102	1003	10002	2001	352
1000004	se	102	1003	10001	2002	523
1000005	ada	103	1004	10004	2003	124


```
-- insert values into CATEGORY
INSERT INTO CATEGORY VALUES
(10001, 'cs'),
(10002, 'med'),
(10003, 'bio'),
(10004, 'meteor'),
(10005, 'mech');
SELECT * FROM CATEGORY;
```

CATEGORYID	DESCRIPTION
10001	cs
10002	med
10003	bio
10004	meteor
10005	mech

```
-- insert values into ORDER_DETAILS
INSERT INTO ORDER_DETAILS VALUES
(1, 1000001, 12),
(1, 1000002, 2),
(2, 1000002, 15),
(3, 1000003, 23),
(4, 1000003, 14),
(5, 1000005, 7);
SELECT * FROM ORDER_DETAILS;
```

ORDERNO	BOOKID	QUANTITY
1	1000001	12
1	1000002	2
2	1000002	15
3	1000003	23
4	1000003	14
5	1000005	7

iii. Give the details of the authors who have 2 or more books in the catalog and the price of the books in the catalog and the year of publication is after 2000.

```
SELECT * FROM AUTHOR A WHERE A.AUTHORID IN (SELECT C.AUTHORID FROM CATALOG C WHERE
YEAR>2000 AND C.PRICE > (SELECT AVG (PRICE) FROM CATALOG) GROUP BY C.AUTHORID HAVING
COUNT(AUTHORID)>=2);
```

AUTHORID	NAME	CITY	ACOUNTRY
102	TONY	HAYHOOD	USA

iv. Find the author of the book which has maximum sales.

```
SELECT NAME FROM AUTHOR A,CATALOG C WHERE A.AUTHORID=C.AUTHORID AND BOOKID IN
(SELECT BOOKID FROM ORDER_DETAILS WHERE QUANTITY= (SELECT MAX(QUANTITY) FROM
ORDER_DETAILS));
```

name

TONY

v. Demonstrate how you increase the price of books published by a specific publisher by 10%.
UPDATE CATALOG SET PRICE = PRICE*1.1 WHERE PUBLISHERID IN (SELECT PUBLISHERID FROM PUBLISHER WHERE NAME= 'POOJA');

✓ 1 row affected. (Query took 0.0177 seconds.)

```
Update catalog set price = price*1.1 where publisherid in ( Select publisherid from publisher where name= 'POOJA')
```

PROGRAM 8: STUDENT ENROLLMENT DATABASE

Consider the following database of student enrollment in courses and books adopted for each course.

STUDENT (regno: String, name: String, major: String, bdate: date)

COURSE (course #: int, cname: String, dept: String)

ENROLL (regno: String, cname: String, sem: int, marks: int)

BOOK_ADOPTION (course #: int, sem: int, book-ISBN: int)

TEXT (book-ISBN:int, book-title:String, publisher: String, author: String)

- i. Create the above tables by properly specifying the primary keys and the foreign keys.

CREATE TABLE student

```
(  
  regno VARCHAR(15),  
  name VARCHAR(20),  
  major VARCHAR(20),  
  bdate DATE,  
  PRIMARY KEY (regno)  
);
```

CREATE TABLE course

```
(  
  courseno INT,  
  cname VARCHAR(20),  
  dept VARCHAR(20),  
  PRIMARY KEY (courseno)  
);
```

CREATE TABLE enroll

```
(  
  regno VARCHAR(15),  
  courseno INT,  
  sem INT(3),  
  marks INT(4),  
  PRIMARY KEY (regno,courseno),  
  FOREIGN KEY (regno) REFERENCES student (regno),  
  FOREIGN KEY (courseno) REFERENCES course (courseno)  
);
```

CREATE TABLE text

```
(  
  book_isbn INT(5),  
  book_title VARCHAR(20),  
  publisher VARCHAR(20),  
  author VARCHAR(20),  
  PRIMARY KEY (book_isbn)  
);
```

CREATE TABLE book_adoption

```
(  
  courseno INT,  
  sem INT(3),  
  book_isbn INT(5),  
  PRIMARY KEY (courseno,book_isbn),  
  FOREIGN KEY (courseno) REFERENCES course (courseno),  
  FOREIGN KEY (book_isbn) REFERENCES text(book_isbn)  
);
```

ii. Enter at least five tuples for each relation.

```
INSERT INTO student (regno,name,major,bdate) VALUES
('1pe11cs001','a','sr',19931230),
('1pe11cs002','b','sr',19930924),
('1pe11cs003','c','sr',19931127),
('1pe11cs004','d','sr',19930413),
('1pe11cs005','e','jr',19940824);
```

```
INSERT INTO course VALUES
(111,'OS','CSE'),
(112,'EC','CSE'),
(113,'SS','ISE'),
(114,'DBMS','CSE'),
(115,'SIGNALS','ECE');
```

```
INSERT INTO text VALUES
(10,'DATABASE SYSTEMS','PEARSON','SCHIELD'),
(900,'OPERATING SYS','PEARSON','LELAND'),
(901,'CIRCUITS','HALL INDIA','BOB'),
(902,'SYSTEM SOFTWARE','PETERSON','JACOB'),
(903,'SCHEDULING','PEARSON','PATIL'),
(904,'DATABASE SYSTEMS','PEARSON','JACOB'),
(905,'DATABASE MANAGER','PEARSON','BOB'),
(906,'SIGNALS','HALL INDIA','SUMIT');
```

```
INSERT INTO enroll (regno,courseno,sem,marks) VALUES
('1pe11cs001',115,3,100),
('1pe11cs002',114,5,100),
('1pe11cs003',113,5,100),
('1pe11cs004',111,5,100),
('1pe11cs005',112,3,100);
```

```
INSERT INTO book_adoption (courseno,sem,book_isbn) VALUES
(111,5,900),
(111,5,903),
(111,5,904),
(112,3,901),
(113,3,10),
(114,5,905),
(113,5,902),
(115,3,906);
```

iii. Demonstrate how you add a new text book to the database and make this book be adopted by some department.

```
INSERT INTO text VALUES (906,'SIGNALS','HALL INDIA','SUMIT');
INSERT INTO book_adoption VALUES (115,3,906);
```

iv. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.

```
SELECT c.courseno,t.book_isbn,t.book_title FROM course c,book_adoption ba,text t
WHERE c.courseno=ba.courseno AND ba.book_isbn=t.book_isbn AND c.dept='CSE' AND 2<( SELECT
COUNT(book_isbn) FROM book_adoption b WHERE c.courseno=b.courseno) ORDER BY t.book_title;
```

Showing rows 0 - 2 (3 total, Query took 0.0060 seconds.) [book_title: DATABASE SYSTEMS... - SCHEDULING...]

```
SELECT c.courseno,t.book_isbn,t.book_title FROM course c,book_adoption ba,text t WHERE c.courseno=ba.courseno AND ba.book_isbn=t.book_isbn AND c.dept='CSE' AND 2<( SELECT COUNT(book_isbn) FROM book_adoption b WHERE c.courseno=b.courseno) ORDER BY t.book_title
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

courseno	book_isbn	book_title
111	904	DATABASE SYSTEMS
111	900	OPERATING SYS
111	903	SCHEDULING

v. List any department that has all its adopted books published by a specific publisher.

```
SELECT DISTINCT c.dept FROM course c WHERE c.dept IN (SELECT c.dept FROM course c,book_adoption b,text t WHERE c.courseno=b.courseno AND t.book_isbn=b.book_isbn AND t.publisher='PEARSON');
```

Showing rows 0 - 0 (1 total, Query took 0.0086 seconds.)

```
SELECT DISTINCT c.dept FROM course c WHERE c.dept IN ( SELECT c.dept FROM course c,book_adoption b,text t WHERE c.courseno=b.courseno AND t.book_isbn=b.book_isbn AND t.publisher='PEARSON')
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

dept
CSE

PROGRAM 9: MOVIE DATABASE

Consider the schema for Movie Database:

ACTOR(Act_id, Act_Name, Act_Gender)
DIRECTOR(Dir_id, Dir_Name, Dir_Phone)
MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)
MOVIE_CAST(Act_id, Mov_id, Role)
RATING(Mov_id, Rev_Stars)

Create database movie;

Use movie;

create table Actor

(
act_id integer primary key,
act_name varchar(100),
act_gender varchar(10)
);

create table Director

(
dir_id integer primary key,
dir_name varchar(200),
dir_phone varchar(100)
);

create table Movies

(
mov_id integer primary key,
mov_title varchar(255),
mov_year year,
mov_lang varchar(100),
dir_id int,
foreign key (dir_id) references Director(dir_id)
);

create table Movie_cast

(act_id int,
foreign key (act_id) references Actor(act_id),
mov_id int,
foreign key(mov_id) references Movies(mov_id),
role varchar(100),
primary key(act_id,mov_id)
);

create table Rating

(
mov_id integer primary key ,
foreign key(mov_id) references Movies(mov_id),
rev_stars integer
);

insert into Actor values

(1001, 'Tom Crusie','M'),
(1002, 'Chris Hemsworth','M'),
(1003, 'Angelina Jolie','F'),
(1004, 'Margot Robbie','F'),
(1005, 'Kate Winslet','F'),
(1006, 'Robert Downey','M');

insert into Director values

```
(9001, 'Hitchcock',9874562154),
(9002, 'Steven Spielberg',9874560054),
(9003, 'Joseph Levitan',9874562178),
(9004, 'Christopher Loyd',9874564454),
(9005, 'Yash Chopra',9874562994),
(9006, 'Tom Jones',9874503154);
```

insert into Movies values

```
(101,'Iron Man',2014,'English',9001), (102,'Prosperity',2001,'Spanish',9001),
(103,'Spiderman',1998,'English',9002), (104,'Star Wars',1999,'English',9003),
(105,'Thor',2017,'English',9002),(106,'Captain America',1994,'English',9004);
```

insert into Movie_cast values

```
(1001,101,'Joey'),
(1001,102,'Conor'),
(1002,102,'Tim'),
(1003,103,'Kate'),
(1004,104,'Claire'),
(1006,105,'Sally'),
(1005,106,'Jo'),
(1002,106,'Craft'),
(1002,104,'Josh'),
(1005,105,'Roy');
```

insert into Rating values (101,4),

```
(102,3),
(103,5),
(104,2),
(105,4),
(106,3);
```

Write SQL queries to

i. List the titles of all movies directed by 'Hitchcock'.

select mov_title from Movies where dir_id in (select dir_id from Director where dir_name='Hitchcock');

Showing rows 0 - 1 (2 total, Query took 0.0035 seconds.)

```
select mov_title from Movies where dir_id in (select dir_id from Director where dir_name='Hitchcock')
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options				mov_title
<input type="checkbox"/>	Edit	Copy	Delete	Iron Man
<input type="checkbox"/>	Edit	Copy	Delete	Prosperity

ii. Find the movie names where one or more actors acted in two or more movies.

select distinct m.mov_title,c.act_id from Movies m, Movie_cast c where m.mov_id=c.mov_id and c.act_id in (select act_id from Movie_cast group by act_id having count(mov_id)>1);

Showing rows 0 - 6 (7 total, Query took 0.0215 seconds.)

```
select distinct m.mov_title,c.act_id from Movies m, Movie_cast c where m.mov_id=c.mov_id and c.act_id in (select act_id from Movie_cast group by act_id having count(mov_id)>1)
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

mov_title	act_id
Iron Man	1001
Prosperity	1001
Prosperity	1002
Star Wars	1002
Thor	1005
Captain America	1002
Captain America	1005

iii. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

select act_name from Actor where act_id in (select a.act_id from (select act_id from Movie_cast natural join Movies where mov_year<2000)a inner join (select act_id from Movie_cast natural join Movies where mov_year>2015)b on a.act_id=b.act_id);

Showing rows 0 - 0 (1 total, Query took 0.0083 seconds.)

```
select act_name from Actor where act_id in (select a.act_id from (select act_id from Movie_cast natural join Movies where mov_year<2000)a inner join (select act_id from Movie_cast natural join Movies where mov_year>2015)b on a.act_id=b.act_id)
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

act_name
Kate Winslet

iv. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.

Select mov_title, max(rev_stars) from movie inner join rating using(mov_id) group by mov_title having max(rev_stars)>0 order by mov_title;

Showing rows 0 - 5 (6 total, Query took 0.0042 seconds.) [mov_title: CAPTAIN AMERICA... - THOR...]

```
SELECT mov_title, MAX(rev_stars) FROM movies INNER JOIN rating USING(mov_id) GROUP BY mov_title HAVING MAX(rev_stars)>0 ORDER BY mov_title
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

mov_title	MAX(rev_stars)
Captain America	3
Iron Man	4
Prosperity	3
Spiderman	5
Star Wars	2
Thor	4

v. Update rating of all movies directed by 'Steven Spielberg' to 5.

update Rating set rev_stars=5 where mov_id in (select mov_id from Movies inner join Director on Movies.dir_id=Director.dir_id and Director.dir_name='Steven Spielberg');

1 row affected. (Query took 0.0195 seconds.)

```
update Rating set rev_stars=5 where mov_id in (select mov_id from Movies inner join Director on Movies.dir_id=Director.dir_id and Director.dir_name='Steven Spielberg')
```


PROGRAM 10: COLLEGE DATABASE

Consider the schema for College Database:

STUDENT(USN, SName, Address, Phone, Gender)

SEMSEC(SSID, Sem, Sec)

CLASS(USN, SSID)

SUBJECT(Subcode, Title, Sem, Credits)

IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

Write SQL queries to

i. List all the student details studying in fourth semester 'C' section.

SELECT S.*, SS.SEM, SS.SEC FROM STUDENT S, SEMSEC SS, CLASS C WHERE S.USN = C.USN AND SS.SSID = C.SSID AND SS.SEM = 4 AND SS.SEC='C';

USN	SNAME	ADDRESS	PHONE	GENDER	SEM	SEC
1BI15CS091	MALINI	MANGALURU	235464	F	4	C

ii. Compute the total number of male and female students in each semester and in each section.

SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT FROM STUDENT S, SEMSEC S, CLASS C WHERE S.USN = C.USN AND SS.SSID = C.SSID GROUP BY SS.SEM, SS.SEC, S.GENDER ORDER BY SEM;

SEM	SEC	GENDER	COUNT
3	A	M	1
3	B	M	1
3	C	F	1
4	A	F	1
4	A	M	1
4	B	F	1
4	C	F	1
7	A	F	1
7	A	M	2
8	A	F	1
8	A	M	1
8	B	F	1
8	C	M	1

iii. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.

CREATE VIEW STUDENT_TEST1_MARKS_V AS SELECT TEST1, SUBCODE FROM IAMARKS WHERE USN = '1BI15CS101';

SELECT * FROM STUDENT_TEST1_MARKS_V;

TEST1	SUBCODE
15	10CS81
12	10CS82
19	10CS83
20	10CS84
15	10CS85