**1st Problem**

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)

Create Table **Publisher**

( Name varchar(20),

Address varchar(20),

Phone int,

Primary Key(Name));

Create Table **Library\_Branch**

( Branch\_id int,

Branch\_Name varchar(20),

Address varchar(20),

Primary Key(Branch\_id));

Create Table **Book**  
( Book\_id int,

Title varchar(20),

Publisher\_Name varchar(20),

Pub\_Year int,

Primary Key(Book\_id),

Foreign Key(Publisher\_Name) references Publisher(Name) on delete cascade);

Create Table **Book\_Authors**

( Book\_id int,

Author\_Name varchar(20),

Primary Key(Book\_id),

Foreign Key(Book\_id) references Book(Book\_id) on delete cascade);

Create Table **Book\_copies**

( Book\_id int,

Branch\_id int,

No\_of\_copies int,

Primary Key(Book\_id,Branch\_id),

Foreign Key(Book\_id) references Book(Book\_id) on delete cascade);

Create Table **Book\_Lending**

( Book\_id int,

Branch\_id int,

Card\_no int,

Date\_out date,

Due\_date date,

Primary Key(Book\_id,Branch\_id,Card\_no),

Foreign Key(Book\_id) references Book(Book\_id) on delete cascade,

Foreign Key(Branch\_id) references Library\_Branch(Branch\_id) on delete cascade);

**Publisher**

SQL>insert into publisher values('&name','&address',&phone);

SQL>select \* from publisher;

SQL>commit;

**Name Address Phone**

Pearson Bengaluru 9954782546

BPB Delhi 8945765478

McGraw\_Hill London 9784578123

Sudha Bengaluru 8745912456

Technical Kolkata 7845612457

**Library\_Branch**

SQL>insert into library\_branch values(&**Branch\_id**,'&**Branch\_Name**', '&**Address**');

SQL>select \* from library\_branch;

SQL>commit;

**Branch\_id Branch\_Name Address**

10 JPNagar Bengaluru

11 JayaNagar Bengaluru

12 SSPuram Tumakuru

13 BHRoad Gubbi

14 MGRoad Delhi

**Book**

SQL>insert into book values(&**Book\_id,’&Title’,’&Pub\_Name’,&Pub\_Year**);

SQL>select \* from book;

SQL>commit;

**Book\_id Title Pub\_Name Pub\_Year**

1 DBMS McGraw\_Hill 2017

2 ADBMS McGraw\_Hill 2018

3 CN Pearson 2016

4 CG Sudha 2018

5 OS Pearson 2016

**Book\_Authors**

SQL>insert into **Book\_Authors values**(&**Book\_id,’&Author\_Name’**);

SQL>select \* from **Book\_Authors**;

SQL>commit;

**Book\_id Author\_Name**

1 Navathe

2 Navathe

3 Nadir

4 Angel

5 Galvin

**Book\_Copies**

SQL>insert into **Book\_Copies** values(&**Book\_id,&Branch\_id,&No\_of\_copies**);

SQL>select \* from **Book\_Copies**;

SQL>commit;

**Book\_id Branch\_id No\_of\_copies**

1 10 10

1 11 5

2 12 2

2 13 5

3 14 7

4 11 3

5 10 1

**Book\_Lending**

SQL>insert into **Book\_Lending** values(&**Book\_id,&Branch\_id,&Card\_No,’&Date\_out’,’&Due\_date’**);

SQL>select \* from **Book\_Lending**;

SQL>commit;

**Book\_id Branch\_id Card\_No Date\_out Due\_date**

1 10 101 1-Jan-2017 1-Jan-2018

3 14 101 1-Feb-2017 1-Jun-2017

2 13 101 1-Mar-2017 10-Aug-2017

4 11 101 1-Apr-2017 10-Aug-2017

1 11 104 1-May-2017 10-Aug-2017

**Write SQL queries to**

1. Retrieve details of all books in the library – id, title, name of publisher,

authors, number of copies in each branch, etc.

Select B.Book\_ID, BC.Branch\_ID, title,Publisher\_name, author\_name,No\_of\_copies

from Book B, Book\_Authors BA,Book\_Copies BC

Where B.Book\_ID = BA.BOOK\_ID and

B.BOOK\_ID = BC.BOOk\_ID;

**Output**

BOOK\_ID BRANCH\_ID TITLE PUBLISHER\_NAME AUTHOR\_NAME NO\_OF\_COPIES

--------- ---------- -------------------- -------------------- -------------------- ------------

1 10 DBMS McGraw\_Hill Navathe 10

1 11 DBMS McGraw\_Hill Navathe 5

2 12 ADBMS McGraw\_Hill Navathe 2

2 13 ADBMS McGraw\_Hill Navathe 5

3 14 CN Pearson Nadir 7

4 11 CG Sudha Angel 3

5 10 OS Pearson Galvin 1

2. Get the particulars of borrowers who have borrowed more than 3 books, but

from Jan 2017 to Jun 2017.

Select Card\_no

From Book\_Lending

Where Date\_out between ’1-Jan-2017’ and ’30-Jun-2017’

Group by Card\_no

Having count(\*)>3;

**Output**

CARD\_NO

---------

101

3. Delete a book in BOOK table. Update the contents of other tables to reflect

this data manipulation operation.

Delete from BOOK

Where BOOK\_ID=3;

**OUTPUT**

1 row deleted.

SQL>select \* from book;

SQL>select \* from book\_copies;

SQL>select \* from BOOK\_AUTHORS;

SQL>select \* from book\_lending;

4. Partition the BOOK table based on year of publication. Demonstrate its

working with a simple query.

Create view v\_PYear

as

select pub\_year

from book;

Select \* from v\_Pyear;

**Output**

PUB\_YEAR

----------

2017

2018

2016

2018

2016

**5.** Create a view of all books and its number of copies that are currently available

in the Library.

Create view MyBooks

as

Select B.BOOK\_ID, B.Title,

sum(No\_of\_Copies)as NC

from BOOK B,BOOK\_COPIES BC

where B.BOOK\_ID = BC.BOOK\_ID

Group by B.BOOK\_ID,B.Title;

Select \* from MyBooks;

**Output**

BOOK\_ID TITLE NC

---------- -------------------- ----------

1 DBMS 15

2 ADBMS 7

3 CN 7

4 CG 3

5 OS 1

**2nd Problem**

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 table **Salesman**

( Salesman\_id int,

Name varchar(20),

City varchar(20),

Commission real,

Primary key(Salesman\_id));

Create table **Customer**

( Customer\_id int,

Cust\_Name varchar(20),

City varchar(20),

Grade int,

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);

SQL>insert into Salesman values(&**Salesman\_id,’&Name’,’&City’,&Commission**);

SQL>select \* from Salesman;

**Salesman\_id Name City Commission**

1000 john bangalore 25

2000 ravi bangalore 20

3000 kumar mysore 15

4000 smith delhi 30

5000 harsha hyderabad 15

SQL>insert into Customer values(&**Customer\_id,’&Cust\_Name’,’&City’,&Grade,**

**&Salesman\_id**);

SQL>select \* from Customer;

**Customer\_id Cust\_Name City Grade Salesman\_id**

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

SQL>insert into Orders values(&**Ord\_No,&Purchase\_Amt,’&Ord\_Date’,&Customer\_id, &Salesman\_id**);

SQL>select \* from Orders;

**Ord\_No Purchase\_Amt Ord\_Date Customer\_id Salesman\_id**

50 5000 4-MAY-2017 10 1000

51 450 4-MAY-2017 10 2000

52 1000 4-MAY-2017 13 2000

53 3500 4-MAY-2017 14 3000

54 550 4-MAY-2017 12 2000

Write SQL queries to

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

select grade, count(\*)

from customer

where grade>(select avg(grade)

from customer

where city=’bangalore’)

group by grade;

**OUTPUT**

GRADE COUNT(\*)

---------- ----------

400 2

300 1

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

Select salesman\_id,name

From salesman

Where salesman\_id in(select salesman\_id

From customer

Group by salesman\_id having count(\*)>1);

**OUTPUT**

SALESMAN\_ID NAME

----------- --------------------

1000 john

2000 ravi

3. List all the salesman and indicate those who have and don’t have customers in

their cities (Use UNION operation.)

select s.salesman\_id,cust\_name,name

from salesman S,customer C

where S.city=C.city

UNION

Select salesman\_id,name,’no match’

From salesman

Where city not in(select city

From customer);

**OUTPUT**

SALESMAN\_ID CUST\_NAME NAME

----------- -------------------- --------------------

1000 chethan john

1000 mamatha john

1000 preethi john

2000 chethan ravi

2000 mamatha ravi

2000 preethi ravi

3000 kumar no match

4000 smith no match

5000 harsha no match

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

Create view maxorders

As

Select S.salesman\_id,name,ord\_date

From salesman S,orders O

Where S.salesman\_id=O.salesman\_id

And

O.purchase\_amt=(select max(purchase\_amt)

From orders C

Where C.ord\_date=O.ord\_date);

Select \* from maxorders;

**OUTPUT**

SALESMAN\_ID NAME ORD\_DATE

----------- -------------------- ---------

1000 john 04-MAY-17

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;

**OUTPUT**

1 row deleted.

SQL>select \* from salesman;

SQL> select \* from customer;

SQL>select \* from orders;

**3rd  Problem**

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 table **Actor**

( Act\_id int,

Act\_Name varchar(20),

Act\_Gender varchar(20),

Primary key(Act\_id));

Create table **Director**

( Dir\_id int,

Dir\_Name varchar(20),

Dir\_Phone int,

Primary key(Dir\_id));

Create table **Movies**

( Mov\_id int,

Mov\_Title varchar(20),

Mov\_Year int,

Mov\_Lang varchar(20),

Dir\_id int,

Primary key(Mov\_id),

Foreign key(Dir\_id) references Director(Dir\_id));

Create table **Movie\_Cast**

( Act\_id int,

Mov\_id int,

Role varchar(20),

Primary key(Act\_id,Mov\_id),

Foreign key(Act\_id) references Actor(Act\_id),

Foreign key(Mov\_id) references Movies(Mov\_id));

Create table **Rating**

( Mov\_id int,

Rev\_Stars int,

Primary key(Mov\_id),

Foreign key(Mov\_id) references Movies(Mov\_id));

SQL> insert into Actor values(&act\_id,'&act\_name','&act\_gender');

SQL> select \* from actor;

**Act\_id Act\_Name Act\_Gender**

301 anushka F

302 prabhas M

303 punith M

304 jermy M

SQL> insert into Director values(&dir\_id,'&dir\_name',&dir\_phone);

SQL>select \* from director;

**Dir\_id Dir\_Name Dir\_Phone**

60 rajmouli 8899112200

61 hitchcock 7760508015

62 farhan 7619195833

63 steven spielberg 9379679136

SQL>insert into Movies values(&mov\_id,'&mov\_title',&mov\_year,'&mov\_lang',&dir\_id);

SQL> select \* from movies;

**Mov\_id Mov\_Title Mov\_Year Mov\_Lang Dir\_id**

1001 bahubali-2 2017 telugu 60

1002 bahubali-1 1998 telugu 60

1003 akash 2008 kannada 61

1004 war horse 2011 english 63

SQL>insert into Movie\_Cast values(&act\_id,&mov\_id,'&role');

SQL> select \* from movie\_cast;

**Act\_id Mov\_id Role**

301 1002 heroine

301 1001 heroine

303 1003 hero

303 1002 guest

304 1004 hero

SQL>insert into Rating values(&mov\_id,&rev\_stars);

SQL> select \* from rating;

**Mov\_id Rev\_Stars**

1001 4

1002 2

1003 5

1004 4

Write SQL queries to

1. List the titles of all movies directed by ‘Hitchcock’.

Select mov\_title

From movies

Where dir\_id=

(select dir\_id

From director

Where dir\_name=’Hitchcock’);

**OUTPUT**

MOV\_TITLE

--------------------

akash

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

Select mov\_title

From movies m ,movie\_cast mc

Where m.mov\_id=mc.mov\_id and mc.act\_id in(select act\_id

From movie\_cast

Group by act\_id having count(\*)>1)

Group by m.mov\_title having count(\*)>1;

**OUTPUT**

MOV\_TITLE

--------------------

bahubali-1

3. List all actors who acted in a movie before 2000 and also in a movie after 2015

(use JOIN operation).

Select a.act\_name

From actor a,movies m,movie\_cast mc

Where a.act\_id=mc.act\_id and m.mov\_id=mc.mov\_id and m.mov\_year<2000

INTERSECT

Select a.act\_name

From actor a,movies m,movie\_cast mc

Where a.act\_id=mc.act\_id and m.mov\_id=mc.mov\_id and m.mov\_year<2015;

**OUTPUT**

ACT\_NAME

--------------------

anushka

punith

4. 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 m.mov\_title, r.rev\_stars

From movies m,rating r

Where m.mov\_id=r.mov\_id and r.rev\_stars>0 and r.rev\_stars=(select max(rev\_stars)

From rating)

Order by m.mov\_title;

**OUTPUT**

MOV\_TITLE REV\_STARS

-------------------- ----------

akash 5

5. 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 where dir\_id in (select dir\_id

From director

Where dir\_name=’Steven Spielberg’));

**OUTPUT**

1 row updated.

SQL> Select \* from Rating;

MOV\_ID REV\_STARS

---------- ----------

1001 4

1002 2

1003 5

1004 5

**4th Problem**

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)

Create table **Student**

(usn varchar2(10),

sname varchar2(20),

address varchar2(20),

phone int,

gender char(1),

primary key(usn));

Create table **Semsec**

(ssid varchar2(5),

sem int,

sec char(1),

primary key(ssid));

Create table **Class**

(usn varchar2(10),

ssid varchar2(5),

primary key(usn),

foreign key(usn) references student(usn),

foreign key(ssid) references semsec(ssid));

Create table **Subject**

(subcode varchar2(10),

title varchar2(20),

sem int,

credits int,

primary key(subcode));

Create table **IAMarks**

(usn varchar2(10),

subcode varchar2(10),

ssid varchar2(5),

test1 int,

test2 int,

test3 int,

finalIA int,

primary key(usn,subcode,ssid),

foreign key(usn) references student(usn),

foreign key(subcode) references subject(subcode),

foreign key(ssid) references semsec(ssid));

SQL>insert into student values('&usn','&name','&address',&phone,'&gender');

SQL>select \* from student;

**USN SNAME ADDRESS PHONE G**

---------- -------------------- -------------------- ---------- -

1BI13CS020 Akshay Belagavi 8877881122 M

1BI13CS062 Sandhya Bangalore 7722829912 F

1BI13CS091 Teesha Bangalore 7712312312 F

1BI13CS066 Supriya Mangalore 8877881133 F

1BI15CS011 Ajay Tumakuru 9845091341 M

1BI15CS029 Chitra Davanagere 7696722121 F

1BI15CS045 Jeeva Bellary 9944850121 M

1BI15CS091 Santhosh Mangalore 8812332201 M

1BI15CS101 Varun Tumakuru 9900899072 M

SQL>insert into semsec values('&ssid',&sem,'&sec');

SQL>select \* from semsec;

**SSID SEM S**

----- ---------- -

CSE8A 8 A

CSE8B 8 B

CSE8C 8 C

CSE4A 4 A

CSE4B 4 B

CSE4C 4 C

SQL>insert into class values('&usn','&ssid');

SQL>select \* from class;

**USN SSID**

---------- -----

1BI13CS020 CSE8A

1BI13CS062 CSE8A

1BI13CS066 CSE8B

1BI13CS091 CSE8C

1BI15CS011 CSE4A

1BI15CS029 CSE4A

1BI15CS045 CSE4B

1BI15CS091 CSE4C

1BI15CS101 CSE4B

SQL>insert into subject values('&subcode','&title',&sem,&credits);

SQL>select \* from subject;

**SUBCODE TITLE SEM CREDITS**

---------- -------------------- ---------- ----------

10CS81 ACA 8 4

10CS82 SSM 8 4

15CS41 M4 4 4

15CS42 SE 4 4

15CS43 DAA 4 4

15CS44 MPMC 4 4

15CS45 OOC 4 3

15CS46 DC 4 3

SQL>insert into IAMarks values('&usn','&subcode','&ssid',&test1,&test2,&test3,null);

SQL>select \* from IAMarks;

**USN SUBCODE SSID TEST1 TEST2 TEST3 FINALIA**

---------- ---------- ----- ---------- ---------- ---------- ----------

1BI13CS020 10CS81 CSE8A 10 11 10

1BI13CS020 10CS82 CSE8A 10 11 11

1BI13CS066 10CS81 CSE8B 12 13 14

1BI13CS066 10CS82 CSE8B 13 14 15

1BI13CS091 10CS81 CSE8C 15 16 18

1BI13CS091 10CS82 CSE8C 12 19 14

1BI15CS101 15CS41 CSE4B 15 18 20

1BI15CS101 15CS42 CSE4B 20 20 19

1BI15CS101 15CS43 CSE4B 18 16 17

1BI15CS101 15CS44 CSE4B 16 18 20

1BI15CS101 15CS45 CSE4B 19 20 20

1BI15CS101 15CS46 CSE4B 17 18 19

Write SQL queries to

1. 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';

OUTPUT:

**USN SNAME ADDRESS PHONE G SEM S**

---------- -------------------- -------------------- ---------- - ---------- -

1BI15CS091 Santhosh Mangalore 8812332201 M 4 C

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

each section.

select sem,sec,gender,count(gender)

from student s,class c,semsec ss

where s.usn=c.usn and

ss.ssid=c.ssid

group by sem,sec,gender

order by sem;

OUTPUT:

**SEM S G COUNT(GENDER)**

---------- - - -------------

4 A F 1

4 A M 1

4 B M 2

4 C M 1

8 A F 1

8 A M 1

8 B F 1

8 C F 1

3. Create a view of Test1 marks of student USN ‘1BI17CS101’ in all subjects.

create view testmarks as

select test1,subcode

from IAMarks

where usn='1BI15CS101';

select \* from testmarks;

OUTPUT:

**TEST1 SUBCODE**

---------- ----------

15 15CS41

20 15CS42

18 15CS43

16 15CS44

19 15CS45

17 15CS46

4. Calculate the FinalIA (average of best two test marks) and update the

corresponding table for all students.

create or replace procedure avgmarks

is

cursor CIAMarks is

select greatest(test1,test2) as A, greatest(test1,test3) as B, greatest(test2,test3) as C

from IAMarks

where finalIA is null

for update;

C\_A int;

C\_B int;

C\_C int;

C\_SUM int;

C\_AVG int;

Begin

open CIAMarks;

loop

fetch CIAMarks into C\_A,C\_B,C\_C;

exit when CIAMarks%notfound;

if(C\_A != C\_B) then

C\_SUM := C\_A + C\_B;

else

C\_SUM := C\_A + C\_C;

end if;

C\_AVG := C\_SUM/2;

update IAMarks

set finalIA = C\_AVG

where current of CIAMarks;

end loop;

close CIAMarks;

end;

/

//-----TO EXECUTE PROCEDURE TYPE THE FOLLOWING CODE---//

begin

avgmarks;

end;

/

select \* from IAMarks;

OUTPUT:

**USN SUBCODE SSID TEST1 TEST2 TEST3 FINALIA**

---------- ---------- ----- ---------- ---------- ---------- ----------

1BI13CS091 10CS81 CSE8C 15 16 18 17

1BI13CS091 10CS82 CSE8C 12 19 14 17

1BI15CS101 15CS41 CSE4B 15 18 20 19

1BI15CS101 15CS42 CSE4B 20 20 19 20

1BI15CS101 15CS43 CSE4B 18 16 17 18

1BI15CS101 15CS44 CSE4B 16 18 20 19

1BI15CS101 15CS45 CSE4B 19 20 20 20

1BI15CS101 15CS46 CSE4B 17 18 19 19

1BI13CS020 10CS81 CSE8A 10 11 10 11

1BI13CS020 10CS82 CSE8A 10 11 11 11

1BI13CS066 10CS81 CSE8B 12 13 14 14

1BI13CS066 10CS82 CSE8B 13 14 15 15

5. Categorize students based on the following criterion:

If FinalIA = 17 to 20 then CAT = ‘Outstanding’

If FinalIA = 12 to 16 then CAT = ‘Average’

If FinalIA< 12 then CAT = ‘Weak’

Give these details only for 8th semester A, B, and C section students.

select s.\*,

( case

when finalIA between 17 and 20 then 'OUTSTANDING'

when finalIA between 12 and 16 then 'AVERAGE'

else 'WEAK'

end ) as category

from student s,IAMarks ia,semsec ss

where s.usn=ia.usn and

ss.ssid=ia.ssid and

ss.sem=8 and ss.sec in ('A','B','C');

OUTPUT:

**USN SNAME ADDRESS PHONE G CATEGORY**

---------- -------------------- -------------------- ---------- - -----------

1BI13CS020 Akshay Belagavi 8877881122 M WEAK

1BI13CS020 Akshay Belagavi 8877881122 M WEAK

1BI13CS091 Teesha Bangalore 7712312312 F OUTSTANDING

1BI13CS091 Teesha Bangalore 7712312312 F OUTSTANDING

1BI13CS066 Supriya Mangalore 8877881133 F AVERAGE

1BI13CS066 Supriya Mangalore 8877881133 F AVERAGE

**5th** **Problem**

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)

Create table Employee

(SSN int,

Name varchar(20),

Address varchar(20),

Sex char(1),

Salary int,

SuperSSN int,

DNo int,

Primary Key(SSN));

Create table Department

(DNo int,

DName varchar(20),

MgrSSN int,

MgrStartDate Date,

Primary Key(DNo));

Create table Dlocation

(DNo int,

DLoc varchar(20),

Primary Key(DNo,DLoc),

Foreign Key(DNo) references Department(DNo));

Create table Project

(PNo int,

PName varchar(20),

PLocation varchar(20),

DNo int,

Primary Key(PNo),

Foreign Key(DNo) references Department(DNo));

Create table Works\_On

(SSN int,

PNo int,

Hours int,

Primary Key(SSN,PNo),

Foreign Key(SSN) references Employee(SSN),

Foreign Key(PNo) references Project(PNo));

SQL> Alter table Employee

Add Foreign Key(DNo) references Department(DNo) Initially Deferred Deferrable;

SQL> Alter table Employee

Add Foreign Key(SuperSSN) references Employee(SSN) Initially Deferred Deferrable;

SQL> Alter table Department

Add Foreign Key(MgrSSN) references Employee(SSN) Initially Deferred Deferrable;

SQL>insert into Employee values(&ssn,’&name’,’&address’,’&sex’,&salary,&superssn,&dno);

SQL>Select \* from Employee;

**EMPLOYEE**

**SSN NAME ADDRESS SEX SALARY SUPERSSN DNO**

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

1 John Bangalore M 600000 NULL 11

2 Franklin Tumkur M 650000 1 11

3 Alicia Mysore F 670000 2 11

4 Jennifer Belagavi F 675000 3 11

5 Ramesh Delhi M 680000 4 11

6 Joyce Bellari F 690000 5 11

7 Ahmad Chennai M 700000 6 12

8 James Hyderabad M 500000 7 13

9 Bhaskar Mumbai M 800000 8 5

10 Girish Kolkata M 900000 9 5

11 Scott Mumbai M 800000 10 11

SQL>insert into Department values(&dno,’&dname’,&mgrssn,’&mgrstartdate’);

SQL>Select \* from Department;

**DEPARTMENT**

**DNO DNAME MGRSSN MGRSTART DATE**

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

11 Accounts 11 01-Jan-2005

12 Sales 2 02-Feb-2006

13 Marketing 3 03-Mar-2007

14 Research 4 04-Apr-2008

15 Administration 5 05-May-2009

SQL>insert into Dlocation values(&dno,’&dloc’);

SQL>Select \* from Dlocation;

**DLOCATION**

**DNO DLOC**

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

11 Delhi

12 Chennai

12 Bangalore

13 Hyderabad

14 Mumbai

SQL>insert into Project values(&pno,’&pname’,’&ploc’,&dno);

SQL>Select \* from Project;

**PROJECT**

**PNO PNAME PLOC DNO**

------- ------------ ---------- --------

100 IOT Bangalore 5

101 Cloud Mangalore 11

102 Bigdata Belagavi 12

103 Seniors Salem 13

104 Banking Delhi 13

105 Payroll Bangalore 5

106 Smartcity Delhi 5

SQL>insert into Works\_on values(&ssn,&pno,&hours);

SQL>Select \* from Works\_on;

**WORKS\_ON**

**SSN PNO HOURS**

------ -------- -----------

10 100 20

10 101 10

9 100 25

9 103 30

6 100 25.5

6 105 35.5

6 106 12.5

1 100 22

1 105 28

1 106 32

11 103 26

11 102 31

11 104 24

**Write SQL queries to:**

1. 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.

Select PNO

From Employee E, Department D,

Project P

Where P.DNO=D.DNO

and E.SSN=D.MgrSSN

and E.name=’Scott’

union

select

PNO

From Employee E, Works\_on W

Where E.SSn = W.SSN

And E.name=’Scott’;

**OUTPUT**

PNO

----------

101

102

103

104

2. Show the resulting salaries if every employee working on the ‘IoT’ project is

given a 10 percent raise.

Select Name,Salary\*1.1

From Employee E,Project P,

Works\_on w

Where E.SSN = W.SSN

and P.PNO = W.PNO

and PName = ’IOT’;

**OUTPUT**

NAME SALARY\*1.1

-------------------- ----------

John 660000

Joyce 759000

Bhaskar 880000

Girish 990000

3. 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.

Select Sum(Salary),Avg(salary),

Max(Salary),Min(Salary)

from Employee E,Department D

where E.DNO = D.DNO

and Dname = ’Accounts’;

**OUTPUT**

SUM(SALARY) AVG(SALARY) MAX(SALARY) MIN(SALARY)

----------- ----------- ----------- -----------

4765000 680714.286 800000 600000

4. Retrieve the name of each employee who works on all the projects

Controlled by department number 5 (use NOT EXISTS operator).

Select E.Name

from Employee E

where NOT EXISTS ((Select PNO

from project

where DNO = 5)

MINUS(select PNO from works\_on w where E.SSN = W.SSN));

**OUTPUT**

NAME

--------------------

John

Joyce

5. 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.

Select D.DNO, Count(\*)

from employee E, Department D

where E.DNO = D.DNO

and salary > 600000

and D.DNO IN(Select E1.DNO from employee E1 Group by E1.DNO Having count(\*) > 5 )

Group by D.DNO;

**OUTPUT**

DNO COUNT(\*)

---------- ----------

11 6