DATABASE MANAGEMENT SYSTEM LABORATORY

List of practical programs

1. 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)

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.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017
3. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
4. Create a view of all books and its number of copies that are currently available in the Library.
5. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

2. 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:

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

3. 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)

Write SQL queries to:

1. List the titles of all movies directed by 'Hitchcock'.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation). 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.
4. Update rating of all movies directed by 'Steven Spielberg' to 5.

4. 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:

1. List all the student details studying in fourth semester 'C' section.
2. Compute the total number of male and female students in each semester and in each section.
3. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.
4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.
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.

5. 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:

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.
2. Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent raise.
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.

4.Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).

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.

PROGRAM 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)

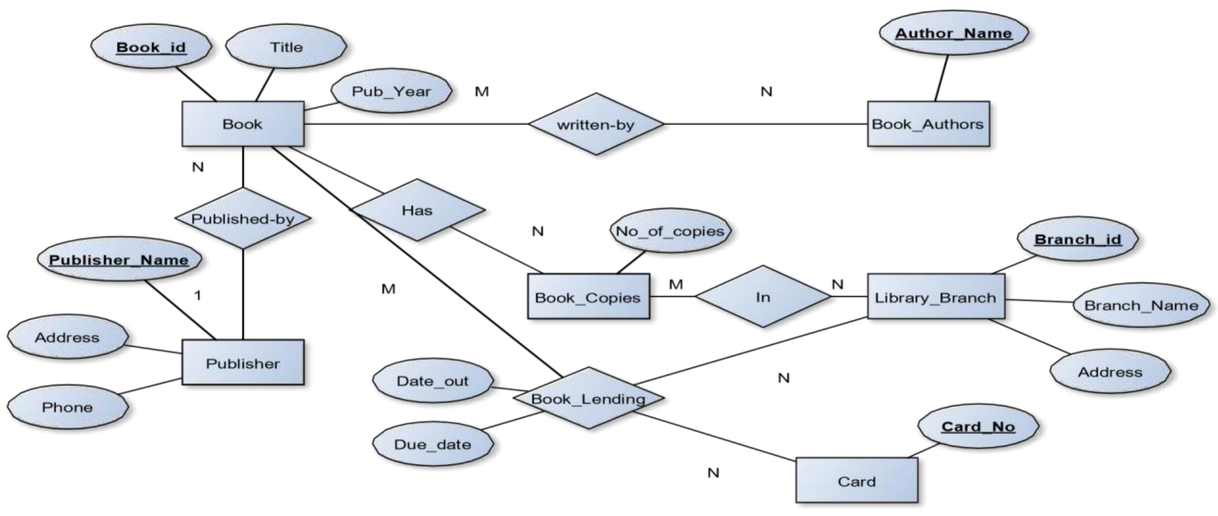
LIBRARY_BRANCH (Branch_id, Branch_Name, Address)

Write SQL queries to :

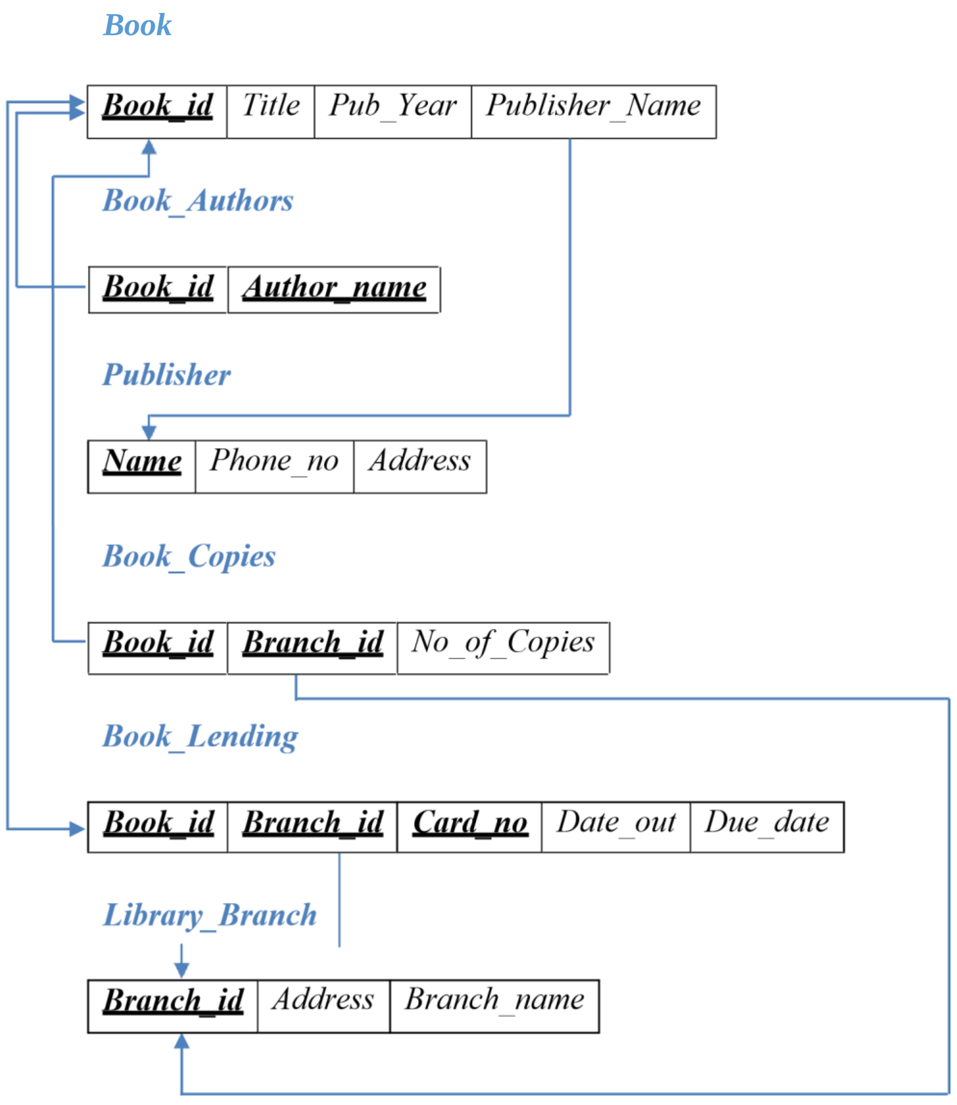
1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017
3. Partition the BOOK table based on year of publication. Demonstrate its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

SOLUTION:

Entity-Relationship Diagram



Schema Diagram



CREATION OF TABLES:

```
CREATE TABLE PUBLISHER
(
NAME VARCHAR2 (20) PRIMARY KEY,
PHONE INTEGER,
ADDRESS VARCHAR2 (20));


CREATE TABLE BOOK
(
BOOK_ID INTEGER PRIMARY KEY,
TITLE VARCHAR2 (20),
PUB_YEAR VARCHAR2 (20),
PUBLISHER_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);


CREATE TABLE BOOK_AUTHORS
(
AUTHOR_NAME VARCHAR2 (20),
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, AUTHOR_NAME));


CREATE TABLE LIBRARY_BRANCH
(
BRANCH_ID INTEGER PRIMARY KEY,
BRANCH_NAME VARCHAR2 (50),
ADDRESS VARCHAR2 (50));


CREATE TABLE BOOK_COPIES
(
NO_OF_COPIES INTEGER,
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE
CASCADE,
PRIMARY KEY (BOOK_ID, BRANCH_ID));


CREATE TABLE CARD
(
CARD_NO INTEGER PRIMARY KEY);
```

```
CREATE TABLE BOOK_LENDING
(
DATE_OUT DATE,
DUE_DATE DATE,
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE
CASCADE,
CARD_NO REFERENCES CARD (CARD_NO) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, BRANCH_ID, CARD_NO));
```

<u>INSERTION OF VALUES TO TABLES:</u>

```
INSERT INTO PUBLISHER VALUES ('MCGRAW-HILL', 9989076587, 'BANGALORE');
INSERT INTO PUBLISHER VALUES ('PEARSON', 9889076565, 'NEWDELHI');
INSERT INTO PUBLISHER VALUES ('RANDOM HOUSE', 7455679345, 'HYDRABAD');
INSERT INTO PUBLISHER VALUES ('HACHETTE LIVRE', 8970862340, 'CHENAI');
INSERT INTO PUBLISHER VALUES ('GRUPO PLANETA', 7756120238, 'BANGALORE');


INSERT INTO BOOK VALUES (1,'DBMS','JAN-2017', 'MCGRAW-HILL');
INSERT INTO BOOK VALUES (2,'ADBMS','JUN-2016', 'MCGRAW-HILL');
INSERT INTO BOOK VALUES (3,'CN','SEP-2016', 'PEARSON');
INSERT INTO BOOK VALUES (4,'CG','SEP-2015', 'GRUPO PLANETA');
INSERT INTO BOOK VALUES (5,'OS','MAY-2016', 'PEARSON');


INSERT INTO BOOK_AUTHORS VALUES ('NAVATHE', 1);
INSERT INTO BOOK_AUTHORS VALUES ('NAVATHE', 2);
INSERT INTO BOOK_AUTHORS VALUES ('TANENBAUM', 3);
INSERT INTO BOOK_AUTHORS VALUES ('EDWARD ANGEL', 4);
INSERT INTO BOOK_AUTHORS VALUES ('GALVIN', 5);


INSERT INTO LIBRARY_BRANCH VALUES (10,'RR NAGAR','BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (11,'RNSIT','BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (12,'RAJAJI NAGAR', 'BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (13,'NITTE','MANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (14,'MANIPAL','UDUPI');


INSERT INTO BOOK_COPIES VALUES (10, 1, 10);
INSERT INTO BOOK_COPIES VALUES (5, 1, 11);
INSERT INTO BOOK_COPIES VALUES (2, 2, 12);
INSERT INTO BOOK_COPIES VALUES (5, 2, 13);
INSERT INTO BOOK_COPIES VALUES (7, 3, 14);
INSERT INTO BOOK_COPIES VALUES (1, 5, 10);
INSERT INTO BOOK_COPIES VALUES (3, 4, 11);


INSERT INTO CARD VALUES (100);
INSERT INTO CARD VALUES (101);
INSERT INTO CARD VALUES (102);
INSERT INTO CARD VALUES (103);
INSERT INTO CARD VALUES (104);


INSERT INTO BOOK_LENDING VALUES ('01-JAN-17','01-JUN-17', 1, 10, 101);
INSERT INTO BOOK_LENDING VALUES ('11-JAN-17','11-MAR-17', 3, 14, 101);
INSERT INTO BOOK_LENDING VALUES ('21-FEB-17','21-APR-17', 2, 13, 101);
INSERT INTO BOOK_LENDING VALUES ('15-MAR-17','15-JUL-17', 4, 11, 101);
INSERT INTO BOOK_LENDING VALUES ('12-APR-17','12-MAY-17', 1, 11, 104);
```

QUERIES:

1. **Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.**

```
SQL> SELECT B. BOOK_ID, B. TITLE, B. PUBLISHER_NAME, A. AUTHOR_NAME,
  2 C.NO_OF_COPIES, L. BRANCH_ID
  3 FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C, LIBRARY_BRANCH L
  4 WHERE B. BOOK_ID=A.BOOK_ID
  5 AND B. BOOK_ID=C.BOOK_ID
  6 AND L. BRANCH_ID=C.BRANCH_ID;
```

OUTPUT:

BOOK_ID	TITLE	PUBLISHER_NAME	AUTHOR_NAME	NO_OF_COPIES	BRANCH_ID
1	DBMS	MCGRAW-HILL	NAVATHE	10	10
1	DBMS	MCGRAW-HILL	NAVATHE	5	11
2	ADBMS	MCGRAW-HILL	NAVATHE	2	12
2	ADBMS	MCGRAW-HILL	NAVATHE	5	13
3	CN	PEARSON	TANENBAUM	7	14
5	OS	PEARSON	GALVIN	1	10
4	CG	GRUPO PLANETA	EDWARD ANGEL	3	11

2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.

```
SQL> SELECT CARD_NO
  2 FROM BOOK_LENDING
  3 WHERE DATE_OUT BETWEEN '01-JAN-2017' AND '01-JUL-2017'
  4 GROUP BY CARD_NO
  5 HAVING COUNT (*)>3;
```

OUTPUT:

CARD_NO
101

3. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

```
SQL> CREATE VIEW V_PUBLICATION AS
  2 SELECT PUB_YEAR
  3 FROM BOOK;

View created.

SQL> SELECT * FROM V_PUBLICATION;
```

OUTPUT:

PUB_YEAR
JAN-2017
JUN-2016
SEP-2016
SEP-2015
MAY-2016

4. Create a view of all books and its number of copies that are currently available in the Library.

```
SQL> CREATE VIEW V_BOOKS AS
  2 SELECT B. BOOK_ID, B. TITLE, C.NO_OF_COPIES
  3 FROM BOOK B, BOOK_COPIES C,
  4 LIBRARIES_BRANCH L WHERE
  5 B. BOOK_ID=C.BOOK_ID
  6 AND C.BRANCH_ID=L.BRANCH_ID;

View created.

SQL> SELECT * FROM V_BOOKS;
```

<u>OUTPUT:</u>

BOOK_ID	TITLE	NO_OF COPIES
1	DBMS	10
1	DBMS	5
2	ADBMS	2
2	ADBMS	5
3	CN	7
5	OS	1
4	CG	3

5. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

```
SQL> DELETE FROM BOOK WHERE BOOK_ID=3;

1 row deleted.

SQL> SELECT * FROM BOOK;
```

OUTPUT:

BOOK_ID	TITLE	PUB_YEAR	PUBLISHER_NAME
1	DBMS	JAN-2017	MCGRAW-HILL
2	ADBMS	JUN-2016	MCGRAW-HILL
4	CG	SEP-2015	GRUPO PLANETA
5	OS	MAY-2016	PEARSON

PROGRAM 2:

Consider the following schema for Order Database:

SALESMAN (Salesman_id, Name, City, Commission)

CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)

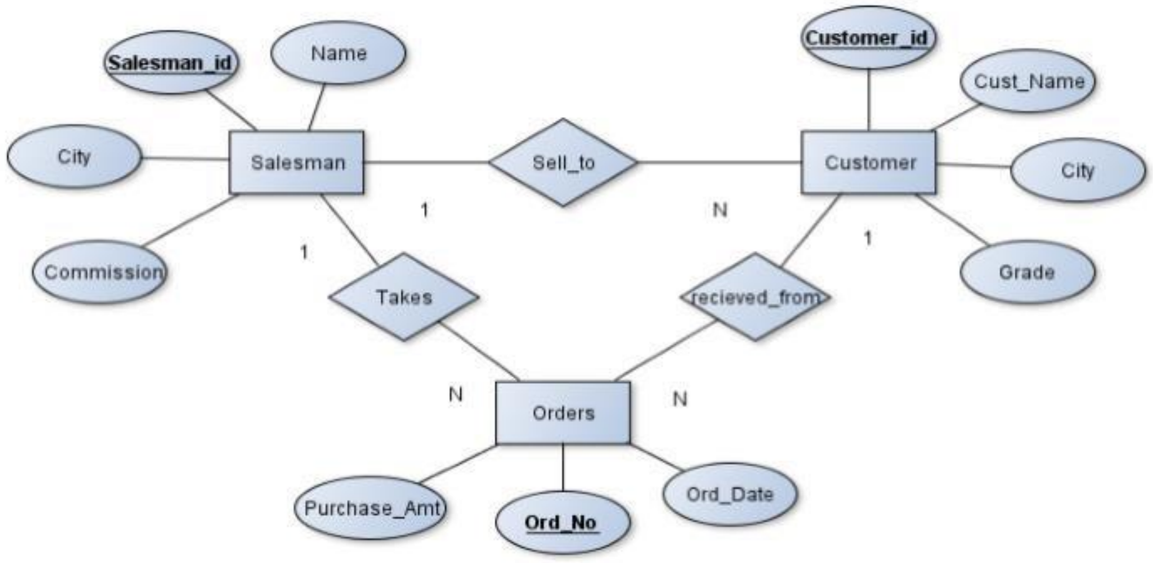
ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

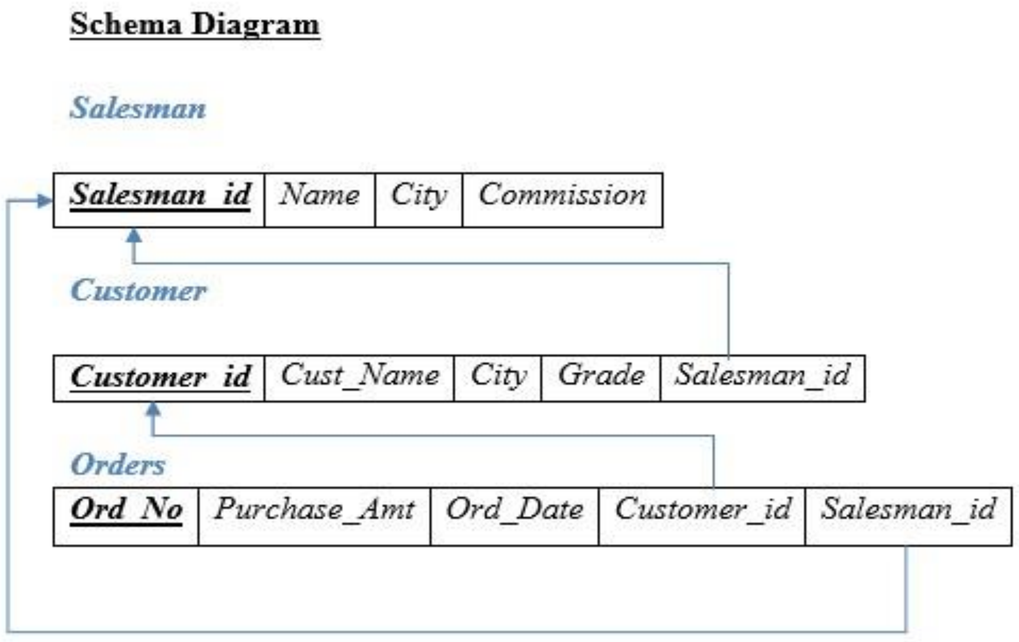
Write SQL queries to :

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

SOLUTION:

Entity-Relationship Diagram





CREATION OF TABLES:

```
CREATE TABLE SALESMAN
(
SALESMAN_ID NUMBER(4),
NAME VARCHAR2(20),
CITY VARCHAR2(20),
COMMISSSION VARCHAR2(20),
PRIMARY KEY (SALESMAN_ID));


CREATE TABLE CUSTOMER1
(
CUSTOMER_ID NUMBER(4),
CUST_NAME VARCHAR2(20),
CITY VARCHAR2(20),
GRADE NUMBER(3),
PRIMARY KEY (CUSTOMER_ID),
SALESMAN_ID REFERENCES SALESMAN (SALESMAN_ID) ON DELETE SET NULL);


CREATE TABLE ORDERS
(
ORD_NO NUMBER(5),
PURCHASE_AMT NUMBER(10,2),
ORD_DATE DATE,
PRIMARY KEY  (ORD_NO),
CUSTOMER_ID REFERENCES CUSTOMER1 (CUSTOMER_ID) ON DLETE CASCADE,
SALESMAN_ID REFERENCES SALESMAN (SALESMAN_ID) ON DELETE CASCADE);
```

INSERTION OF VALUES:

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


INSERT INTO CUSTOMER1 VALUES (10, ‘PREETHI’, ‘BANGALORE’,100, 1000);
INSERT INTO CUSTOMER1 VALUES (11, ‘VIVEK’, ‘MANGALORE’, 300, 1000);
INSERT INTO CUSTOMER1 VALUES (12, ‘BHASKAR’, ‘CHENNAI’, 400, 2000);
INSERT INTO CUSTOMER1 VALUES (13, ‘CHETHAN’, ‘BANGALORE’, 200, 2000);
INSERT INTO CUSTOMER1 VALUES (14, ‘MAMATHA’, ‘BANGALORE’, 400, 3000);


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

QUERIES:

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

```
SQL>SELECT GRADE,
  2 COUNT (DISTINCT CUSTOMER_ID) AS "CUSTOMER GRADE FOR BANGALORE"
  3 FROM CUSTOMER1 GROUP BY GRADE HAVING GRADE > (SELECT
    AVG(GRADE) FROM CUSTOMER1 WHERE
  4 CITY='BANGALORE');
```

OUTPUT:

GRADE	CUSTOMER GRADE FOR BANGALORE
300	1
400	2

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

```
SQL> SELECT SALESMAN_ID, NAME
  2 FROM SALESMAN A WHERE 1 < (SELECT COUNT (*)
  3 FROM CUSTOMER1
  4 WHERE SALESMAN_ID=A.SALESMAN_ID);
```

OUTPUT:

SALESMAN_ID	NAME
1000	JOHN
2000	RAVI

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

```
SQL> SELECT SALESMAN.SALESMAN_ID, NAME, CUST_NAME, COMMISSION
  2 FROM SALESMAN, CUSTOMER1
  3 WHERE SALESMAN.CITY= CUSTOMER1.CITY
  4 UNION
  5 SELECT SALESMAN_ID, NAME, ‘NO MATCH’, COMMISSION
  6 FROM SALESMAN WHERE NOT CITY= ANY(SELECT CITY FROM CUSTOMER1)
  7 ORDER BY 2 DESC;
```

OUTPUT:

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

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

```
SQL> CREATE VIEW ELITSALESMAN AS SELECT B.ORD_DATE, A.SALEMAN_ID,
     A.NAME
  2 FROM SALESMAN A, ORDERS B
  3 WHERE A.SALESMAN_ID= B.SALESMAN_ID
  4 AND B.PURCHASE_AMT=(SELECT MAX (PURCHASE_AMT)
  5 FROM ORDERS C
  6 WHERE C.ORD_DATE=B.ORD_DATE);

View created.

SQL>SELECT * FROM ELITSALESMAN;
```

OUTPUT:

ORD_DATE	SALESMAN_ID	NAME
04-MAY -17	1000	JOHN
20-JAN-17	2000	RAVI
24-FEB-17	2000	RAVI
13-APR-17	3000	KUMAR
09-MAR-17	2000	RAVI

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

Use ON DELETE CASCADE at the end of foreign key definitions while creating child table orders and then execute the following: Use ON DELETE SET NULL at the end of foreign key definitions while creating child table customers and then executes the following:

```
SQL> DELETE FROM SALESMAN
  2 WHERE SALESMAN_ID=1000;
```

OUTPUT:

SALESMAN_ID	NAME	CITY	COMMISSION
2000	RAVI	BANGALORE	20%
3000	KUMAR	MYSORE	15%
4000	SMITH	DELHI	30%
5000	HARSHA	HYDERABAD	15%

PROGRAM 3:

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)

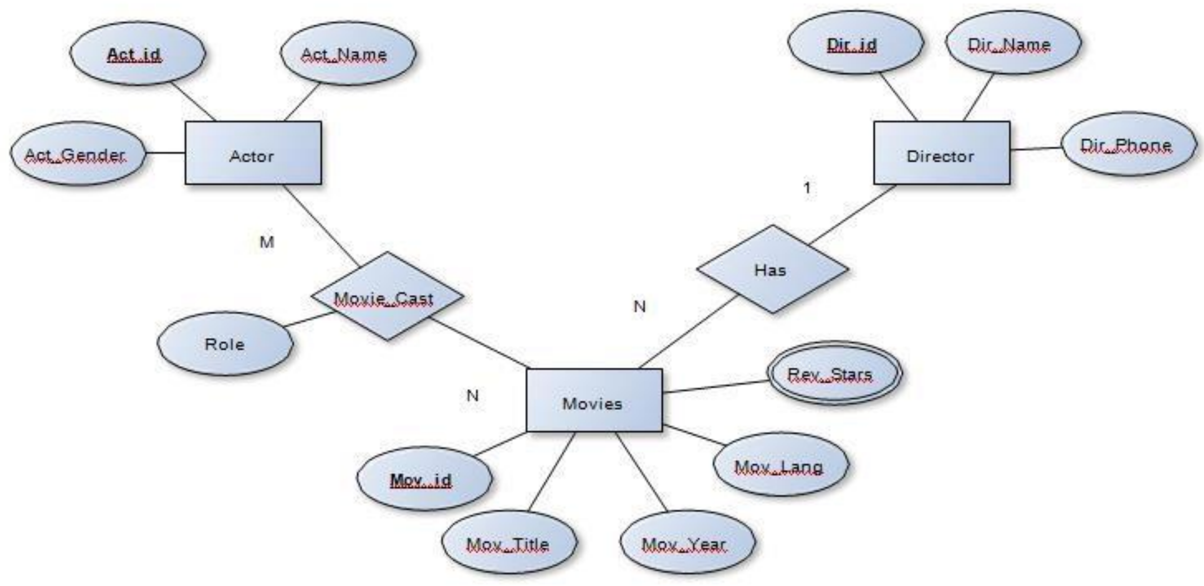
RATING (Mov_id, Rev_Stars)

Write SQL queries to:

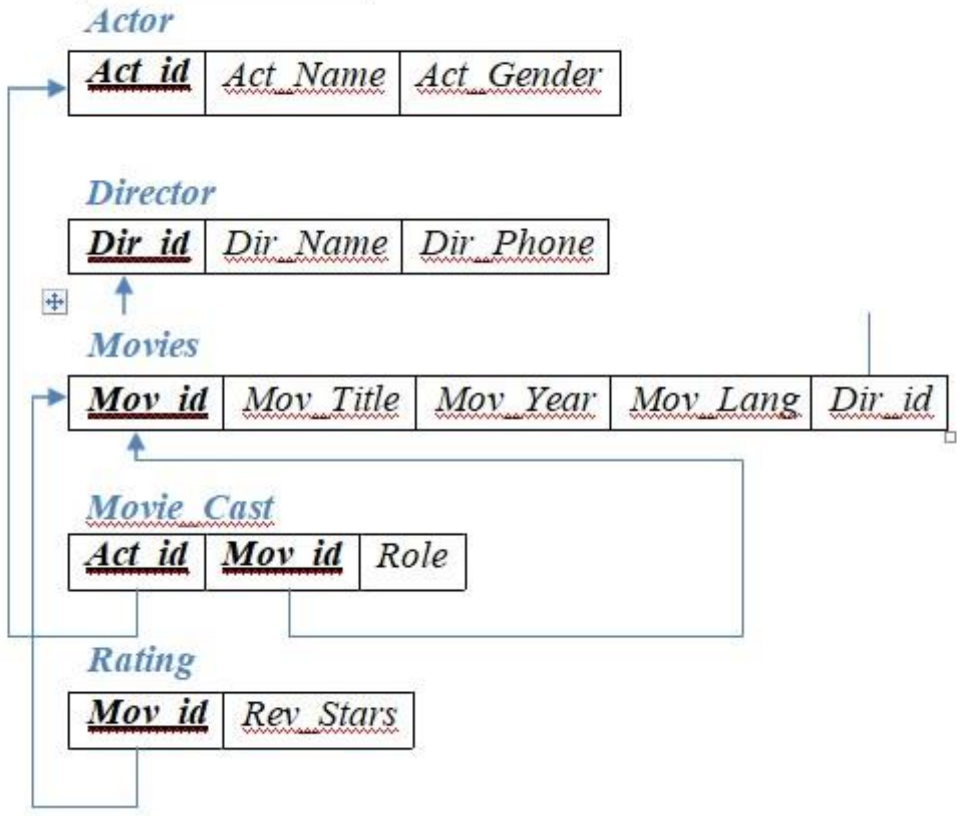
1. List the titles of all movies directed by ‘Hitchcock’.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.
5. Update rating of all movies directed by ‘Steven Spielberg’ to 5.

SOLUTION:

Entity-Relationship Diagram



Schema Diagram



CREATION OF TABLES:

```
CREATE TABLE ACTOR
 (
ACT_ID NUMBER (3),
ACT_NAME VARCHAR (20),
ACT_GENDER CHAR (1), PRIMARY KEY (ACT_ID));


CREATE TABLE DIRECTOR
(
DIR_ID NUMBER (3),
DIR_NAME VARCHAR (20),
DIR_PHONE NUMBER (10), PRIMARY KEY (DIR_ID));


CREATE TABLE MOVIES
 (
MOV_ID NUMBER (4),
MOV_TITLE VARCHAR (25),
MOV_YEAR NUMBER (4),
MOV_LANG VARCHAR (12),
DIR_ID NUMBER (3), PRIMARY KEY (MOV_ID),
FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR (DIR_ID));


CREATE TABLE MOVIE_CAST
 (
ACT_ID NUMBER (3),
MOV_ID NUMBER (4),
ROLE VARCHAR (10),

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 NUMBER (4),

REV_STARS VARCHAR (25), PRIMARY KEY (MOV_ID),

FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

INSERTION OF VALUES:

```
INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');

INSERT INTO ACTOR VALUES (302,'PRABHAS','M');

INSERT INTO ACTOR VALUES (303,'PUNITH','M');

INSERT INTO ACTOR VALUES (304,'JERMY','M');


INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 8751611001);

INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 7766138911);

INSERT INTO DIRECTOR VALUES (62,'FARAN', 9986776531);

INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG', 8989776530);


INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELAGU', 60);

INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015,'TELAGU', 60);

INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);

INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);
INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');
INSERT INTO MOVIE_CAST VALUES (301, 1001, 'HEROINE');
INSERT INTO MOVIE_CAST VALUES (303, 1003, 'HERO');
INSERT INTO MOVIE_CAST VALUES (303, 1002, 'GUEST');
INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');

INSERT INTO RATING VALUES (1001, 4);
INSERT INTO RATING VALUES (1002, 2);
INSERT INTO RATING VALUES (1003, 5);
INSERT INTO RATING VALUES (1004, 4);
```

QUERIES:

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

```
SQL> SELECT MOV_TITLE FROM MOVIES
  2    WHERE DIR_ID IN (SELECT DIR_ID
  3    FROM DIRECTOR
  4    WHERE DIR_NAME = 'HITCHCOCK');
```

OUTPUT:

MOV_TITLE
AKASH

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

```
SQL> SELECT MOV_TITLE
  2 FROM MOVIES M, MOVIE_CAST MV
  3 WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN (SELECT ACT_ID
  4 FROM MOVIE_CAST GROUP BY ACT_ID HAVING COUNT (ACT_ID)>1)
  5 GROUP BY 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).

```
SQL> SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
  2 FROM ACTOR A JOIN MOVIE_CAST C
  3 ON A.ACT_ID=C.ACT_ID JOIN MOVIES M
  4 ON C.MOV_ID=M.MOV_ID
  5 WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
```

OUTPUT:

ACT_NAME	MOV_TITLE	MOV_YEAR
ANUSHKA	BAHUBALI-2	2017

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.

```
SQL> SELECT MOV_TITLE, MAX (REV_STARS)
  2 FROM MOVIES
  3 INNER JOIN RATING USING (MOV_ID)
  4 GROUP BY MOV_TITLE
  5 HAVING MAX (REV_STARS)>0 ORDER BY MOV_TITLE;
```

OUTPUT:

MOV_TITLE	**MAX(REV_STARS)**
AKASH	5
BAHUBALI-1	2
BAHUBALI-2	4
WAR HORSE	4

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

```
SQL> UPDATE RATING SET REV_STARS=5
  2 WHERE MOV_ID IN (SELECT MOV_ID FROM MOVIES
  3 WHERE DIR_ID IN (SELECT DIR_ID
  4 FROM DIRECTOR
  5 WHERE DIR_NAME = 'STEVEN SPIELBERG'));

1 row updated.

SQL> SELECT * FROM RATING;
```

OUTPUT:

MOV_ID	REV_STARS
1001	4
1002	2
1003	5
1004	5

PROGRAM 4:

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:

1. List all the student details studying in fourth semester 'C' section.
2. Compute the total number of male and female students in each semester and in each section.
3. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.
4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.
5. Categorize students based on the following criterion:

If FinalIA = 17 to 20 then CAT = 'Outstanding'

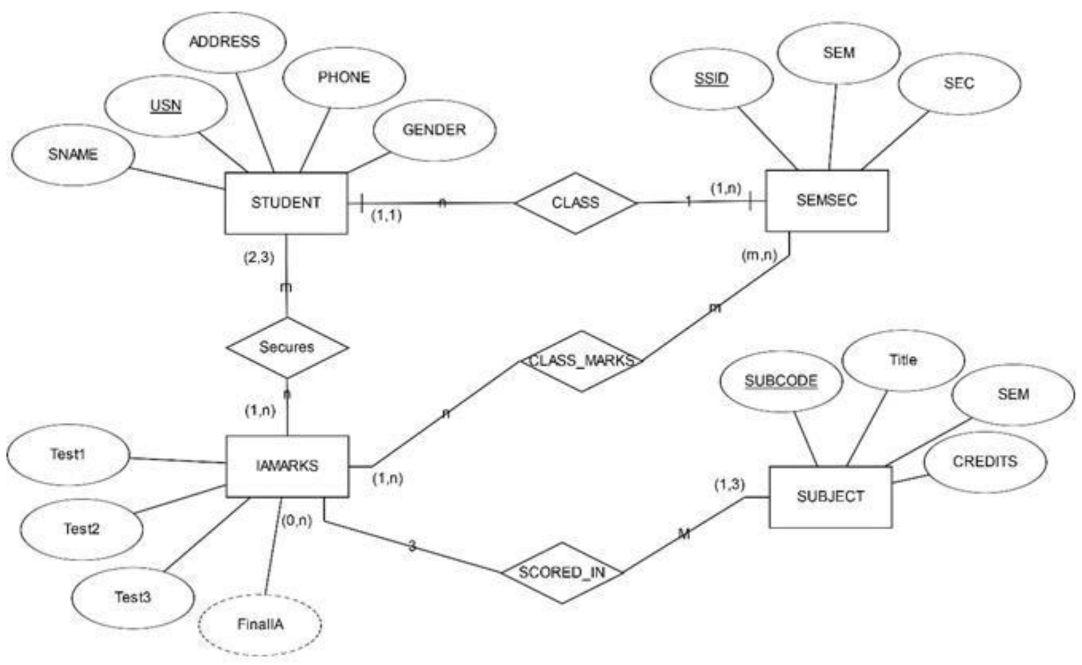
If FinalIA = 12 to 16 then CAT = 'Average'

If FinalIA< 12 then CAT = 'Weak'

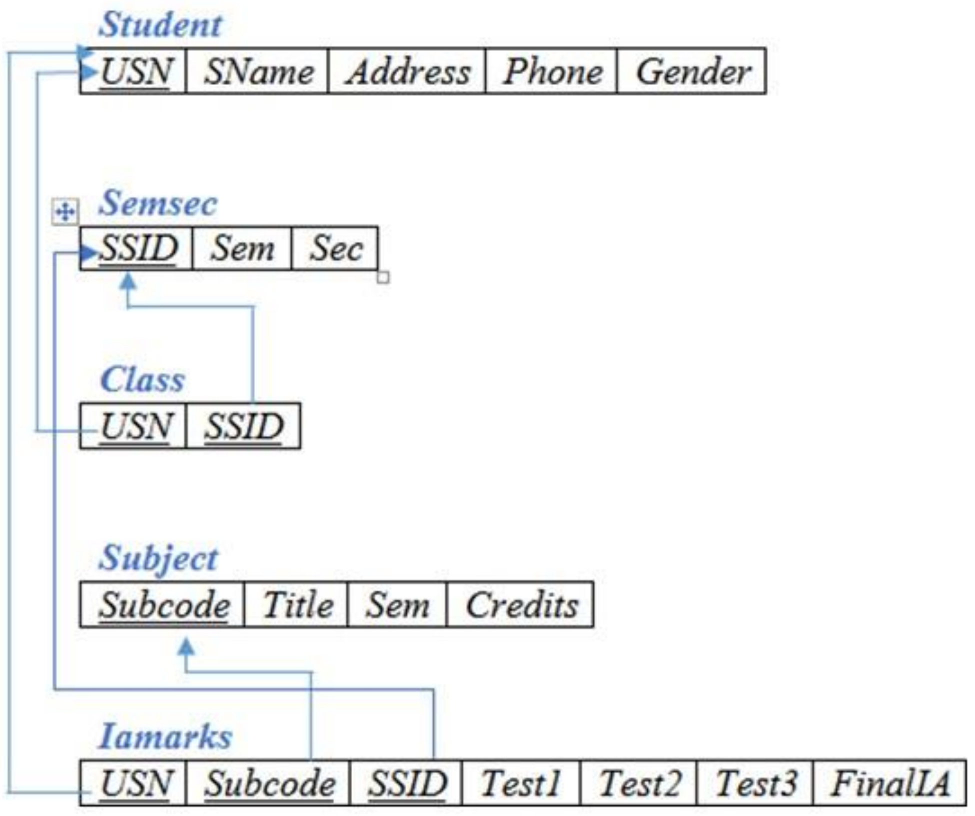
Give these details only for 8th semester A, B, and C section students.

SOLUTION:

Entity - Relationship Diagram



Schema Diagram



CREATION OF TABLES:

```
CREATE TABLE STUDENT (
USN VARCHAR (10) PRIMARY KEY, SNAME VARCHAR (25),
ADDRESS VARCHAR (25),
PHONE NUMBER (10),
GENDER CHAR (1));

CREATE TABLE SEMSEC (
SSID VARCHAR (5) PRIMARY KEY, SEM NUMBER (2),
SEC CHAR (1));

CREATE TABLE CLASS (
USN VARCHAR (10),
SSID VARCHAR (5), PRIMARY KEY (USN, SSID),
FOREIGN KEY (USN) REFERENCES STUDENT (USN), FOREIGN KEY (SSID)
REFERENCES SEMSEC (SSID));


CREATE TABLE SUBJECT (
SUBCODE VARCHAR (8),
TITLE VARCHAR (20),
SEM NUMBER (2),
CREDITS NUMBER (2), PRIMARY KEY (SUBCODE));
```

```
CREATE TABLE IAMARKS (
USN VARCHAR (10),
SUBCODE VARCHAR (8),
SSID VARCHAR (5),
TEST1 NUMBER (2),
TEST2 NUMBER (2),
TEST3 NUMBER (2),
FINALIA NUMBER (2),
PRIMARY KEY (USN, SUBCODE, SSID),
FOREIGN KEY (USN) REFERENCES STUDENT (USN),
FOREIGN KEY (SUBCODE) REFERENCES SUBJECT (SUBCODE), FOREIGN
KEY (SSID) REFERENCES
SEMSEC (SSID));
```

INSERTION OF VALUES:

```
INSERT INTO STUDENT VALUES ('1RN13CS020','AKSHAY','BELAGAVI',
8877881122,'M');
INSERT INTO STUDENT VALUES ('1RN13CS062','SANDHYA','BENGALURU',
7722829912,'F');
INSERT INTO STUDENT VALUES ('1RN13CS091','TEESHA','BENGALURU',
7712312312,'F');
INSERT INTO STUDENT VALUES ('1RN13CS066','SUPRIYA','MANGALURU',
8877881122,'F');
INSERT INTO STUDENT VALUES ('1RN14CS010','ABHAY','BENGALURU',
9900211201,'M');
INSERT INTO STUDENT VALUES ('1RN14CS032','BHASKAR','BENGALURU',
9923211099,'M');
INSERT INTO STUDENT VALUES ('1RN14CS025','ASMI','BENGALURU',
7894737377,'F');
INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR',
9845091341,'M');
```

```
INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE',
7696772121,'F');

INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY',
9944850121,'M');

INSERT INTO STUDENT VALUES ('1RN15CS091','SANTOSH','MANGALURU',
8812332201,'M');

INSERT INTO STUDENT VALUES ('1RN16CS045','ISMAIL','KALBURGI',
9900232201,'M');

INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA',
9905542212,'F');

INSERT INTO STUDENT VALUES ('1RN16CS122','VINAYAKA','CHIKAMAGALUR',
8800880011,'M');



INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');

INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B');

INSERT INTO SEMSEC VALUES ('CSE8C', 8,'C');

INSERT INTO SEMSEC VALUES ('CSE7A', 7,'A');

INSERT INTO SEMSEC VALUES ('CSE7B', 7,'B');

INSERT INTO SEMSEC VALUES ('CSE7C', 7,'C');

INSERT INTO SEMSEC VALUES ('CSE6A', 6,'A');

INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B');

INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');

INSERT INTO SEMSEC VALUES ('CSE5A', 5,'A');

INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B');

INSERT INTO SEMSEC VALUES ('CSE5C', 5,'C');

INSERT INTO SEMSEC VALUES ('CSE4A', 4,'A');

INSERT INTO SEMSEC VALUES ('CSE4B', 4,'B');

INSERT INTO SEMSEC VALUES ('CSE4C', 4,'C');

INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A');

INSERT INTO SEMSEC VALUES ('CSE3B', 3,'B');

INSERT INTO SEMSEC VALUES ('CSE3C', 3,'C');
INSERT INTO SEMSEC VALUES ('CSE2A', 2,'A');
INSERT INTO SEMSEC VALUES ('CSE2B', 2,'B');
INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C');
INSERT INTO SEMSEC VALUES ('CSE1A', 1,'A');
INSERT INTO SEMSEC VALUES ('CSE1B', 1,'B');
INSERT INTO SEMSEC VALUES ('CSE1C', 1,'C');


INSERT INTO CLASS VALUES ('1RN13CS020','CSE8A');
INSERT INTO CLASS VALUES ('1RN13CS062','CSE8A');
INSERT INTO CLASS VALUES ('1RN13CS066','CSE8B');
INSERT INTO CLASS VALUES ('1RN13CS091','CSE8C');
INSERT INTO CLASS VALUES ('1RN14CS010','CSE7A');
INSERT INTO CLASS VALUES ('1RN14CS025','CSE7A');
INSERT INTO CLASS VALUES ('1RN14CS032','CSE7A');
INSERT INTO CLASS VALUES ('1RN15CS011','CSE4A');
INSERT INTO CLASS VALUES ('1RN15CS029','CSE4A');
INSERT INTO CLASS VALUES ('1RN15CS045','CSE4B');
INSERT INTO CLASS VALUES ('1RN15CS091','CSE4C');
INSERT INTO CLASS VALUES ('1RN16CS045','CSE3A');
INSERT INTO CLASS VALUES ('1RN16CS088','CSE3B');
INSERT INTO CLASS VALUES ('1RN16CS122','CSE3C');
INSERT INTO SUBJECT VALUES ('10CS81','ACA', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS82','SSM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS83','NM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS84','CC', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS85','PW', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS71','OOAD', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS72','ECS', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS73','PTW', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS74','DWDM', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS75','JAVA', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS76','SAN', 7, 4);
INSERT INTO SUBJECT VALUES ('15CS51','ME', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS52','CN', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS53','DBMS', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS54','ATC', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS55','JAVA', 5, 3);
INSERT INTO SUBJECT VALUES ('15CS56','AI', 5, 3);
INSERT INTO SUBJECT VALUES ('15CS41','M4', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS42','SE', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS43','DAA', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS44','MPMC', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS45','OOC', 4, 3);
INSERT INTO SUBJECT VALUES ('15CS46','DC', 4, 3);
INSERT INTO SUBJECT VALUES ('15CS31','M3', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS32','ADE', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS33','DSA', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS34','CO', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS35','USP', 3, 3);
INSERT INTO SUBJECT VALUES ('15CS36','DMS', 3, 3);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS81','CSE8C', 15, 16, 18);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS82','CSE8C', 12, 19, 14);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS83','CSE8C', 19, 15, 20);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS84','CSE8C', 20, 16, 19);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS85','CSE8C', 15, 15, 12);
```

QUERIES:

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

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

OUTPUT:

USN	SNAME	ADDRESS	PHONE	G	SEM	S
1RN15CS091	SANTOSH	MANGALURU	8812332201	M	4	C

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

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

OUTPUT:

SEM	S	G	COUNT
3	A	M	1
3	B	F	1
3	C	M	1
4	A	F	1
4	A	M	1
4	B	M	1
4	C	M	1
7	A	F	1
7	A	M	2
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 ‘1BI15CS101’ in all subjects.

```
SQL> CREATE VIEW STU_TEST1_MARKS_VIEW AS
  2  SELECT TEST1, SUBCODE FROM IAMARKS
  3  WHERE USN = '1RN13CS091';

View created.

SQL> SELECT * FROM STU_TEST1_MARKS_VIEW;
```

OUTPUT:

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

4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

```
SQL> CREATE OR REPLACE PROCEDURE AVGMARKS IS
  2  CURSOR C_IAMARKS IS
  3  SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B, GREATEST(TEST3,TEST2)
  4  AS C
  5  FROM IAMARKS
  6  WHERE FINALIA IS NULL FOR UPDATE;
  7  C_A NUMBER;
  8  C_B NUMBER;
  9  C_C NUMBER;
 10  C_SM NUMBER;
 11  C_AV NUMBER;
 12  BEGIN
 13  OPEN C_IAMARKS;
 14  LOOP
 15  FETCH C_IAMARKS INTO C_A, C_B, C_C; EXIT WHEN C_IAMARKS%NOTFOUND;
 16  DBMS_OUTPUT.PUT_LINE(C_A || ' ' || C_B || ' ' || C_C); IF (C_A != C_B) THEN
 17  C_SM:=C_A+C_B; ELSE
 18  C_SM:=C_A+C_C;
 19  END IF;
 20  C_AV:=C_SM/2;
 21  DBMS_OUTPUT.PUT_LINE('SUM = '||C_SM);
 22  DBMS_OUTPUT.PUT_LINE('AVERAGE = '||C_AV);
 23  UPDATE IAMARKS SET FINALIA=C_AV WHERE CURRENT OF C_IAMARKS;
 24  END LOOP;
 25  CLOSE C_IAMARKS;
 26  END;
 27  /

Procedure created.

SQL> SELECT * FROM IAMARKS;
```

OUTPUT:

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1RN13CS091	10CS81	CSE8C	15	16	18	NULL
1RN13CS091	10CS82	CSE8C	12	19	14	NULL
1RN13CS091	10CS83	CSE8C	19	15	20	NULL
1RN13CS091	10CS84	CSE8C	20	16	19	NULL
1RN13CS091	10CS85	CSE8C	15	15	12	NULL

```
SQL> BEGIN
  2  AVGMARKS;
  3  END;
  4  /

PL/SQL procedure successfully completed.
```

SQL> SELECT * FROM IAMARKS;

OUTPUT:

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1RN13CS091	10CS81	CSE8C	15	16	18	17
1RN13CS091	10CS82	CSE8C	12	19	14	17
1RN13CS091	10CS83	CSE8C	19	15	20	20
1RN13CS091	10CS84	CSE8C	20	16	19	20
1RN13CS091	10CS85	CSE8C	15	15	12	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

```
SQL> SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER, (CASE
  2  WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING' WHEN
  IA.FINALIA BETWEEN 12
  3  AND 16 THEN 'AVERAGE' ELSE 'WEAK'
  4  END) AS CAT
  5  FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB WHERE S.USN =
  IA.USN AND
  6  SS.SSID = IA.SSID AND SUB.SUBCODE = IA.SUBCODE AND SUB.SEM = 8;
```

OUTPUT:

USN	SNAME	ADDRESS	PHONE	G	CAT
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OUTSTANDING
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OUTSTANDING
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OUTSTANDING
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OUTSTANDING
1RN13CS091	TEESHA	BENGALURU	7712312312	F	AVERAGE

PROGRAM 5:

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)

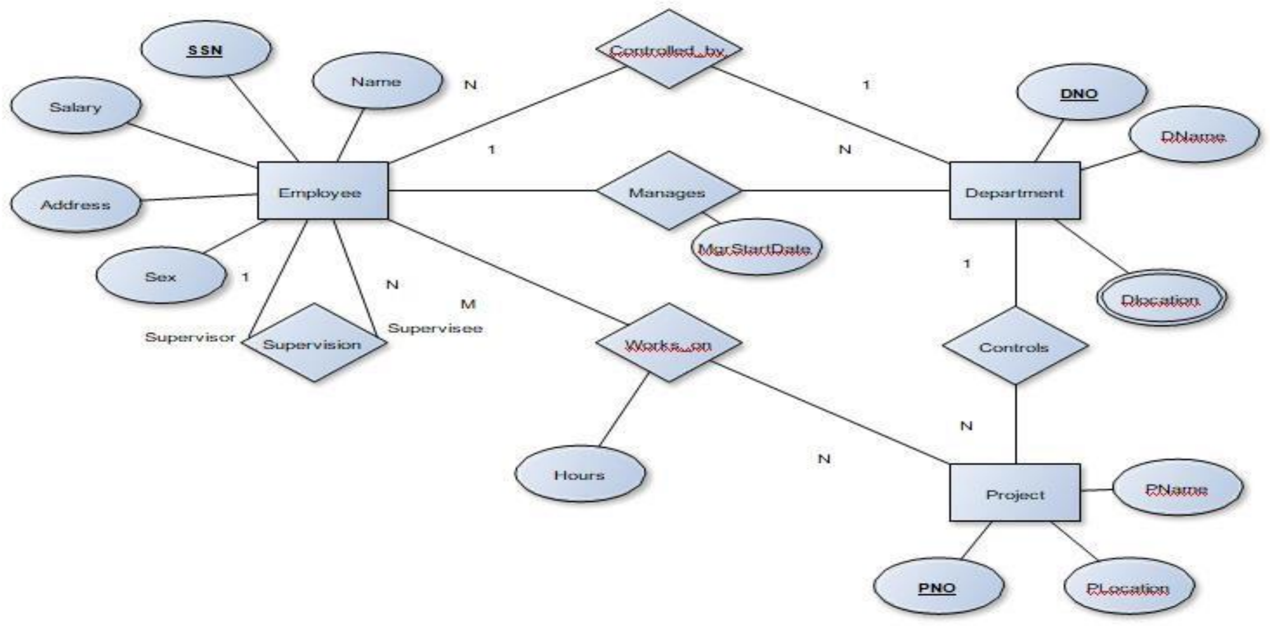
WORKS_ON (SSN, PNo, Hours)

Write SQL queries to :

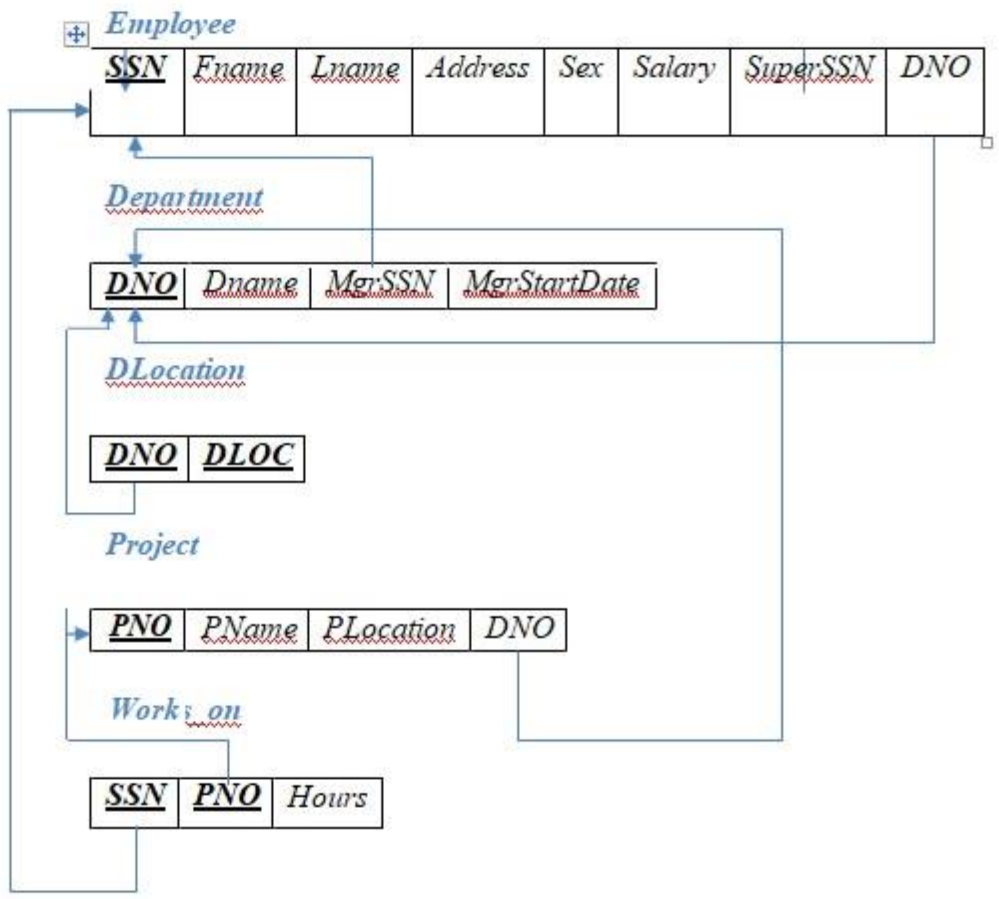
1. Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.
2. Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.
3. Find the 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.
4. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).
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.

SOLUTION:

Entity-Relationship Diagram



Schema Diagram



CREATION OF TABLES:

```
CREATE TABLE DEPARTMENT (
DNO VARCHAR2 (20) PRIMARY KEY,
DNAME VARCHAR2 (20),
MGRSTARTDATE DATE);


CREATE TABLE EMPLOYEE (
SSN VARCHAR2 (20) PRIMARY KEY,
FNAME VARCHAR2 (20),
LNAME VARCHAR2 (20),
ADDRESS VARCHAR2 (20),
SEX CHAR (1),
SALARY INTEGER,
SUPERSSN REFERENCES EMPLOYEE (SSN),
DNO REFERENCES DEPARTMENT (DNO));


CREATE TABLE DLOCATION (
DLOC VARCHAR2 (20),
DNO REFERENCES DEPARTMENT (DNO),
PRIMARY KEY (DNO, DLOC));


CREATE TABLE PROJECT (
PNO INTEGER PRIMARY KEY,
PNAME VARCHAR2 (20),
PLOCATION VARCHAR2 (20),
DNO REFERENCES DEPARTMENT (DNO));
```

```
CREATE TABLE WORKS_ON (
HOURS NUMBER (2),
SSN REFERENCES EMPLOYEE (SSN),
PNO REFERENCES PROJECT(PNO),
PRIMARY KEY (SSN, PNO));
```

```
ALTER TABLE DEPARTMENT
ADD MGRSSN REFERENCES EMPLOYEE (SSN);
```

INSERTION OF VALUES:

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSECE01','JOHN','SCOTT','BANGALORE','M', 450000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE01','JAMES','SMITH','BANGALORE','M', 500000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE02','HEARN','BAKER','BANGALORE','M', 700000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE03','EDWARD','SCOTT','MYSORE','M', 500000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE04','PAVAN','HEGDE','MANGALORE','M', 650000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE05','GIRISH','MALYA','MYSORE','M', 450000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE06','NEHA','SN','BANGALORE','F', 800000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSACC01','AHANA','K','MANGALORE','F', 350000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSACC02','SANTHOSH','KUMAR','MANGALORE','M', 300000);

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSISE01','VEENA','M','MYSORE','M', 600000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES
('RNSIT01','NAGESH','HR','BANGALORE','M', 500000);



INSERT INTO DEPARTMENT VALUES ('1','ACCOUNTS','01-JAN-01','RNSACC02');
INSERT INTO DEPARTMENT VALUES ('2','IT','01-AUG-16','RNSIT01');
INSERT INTO DEPARTMENT VALUES ('3','ECE','01-JUN-08','RNSECE01');
INSERT INTO DEPARTMENT VALUES ('4','ISE','01-AUG-15','RNSISE01');
INSERT INTO DEPARTMENT VALUES ('5','CSE','01-JUN-02','RNSCSE05');
```

Note: update entries of employee table to fill missing fields SUPERSSN and DNO.

```
UPDATE EMPLOYEE SET SUPERSSN=NULL, DNO='3' WHERE SSN='RNSCSE01';
UPDATE EMPLOYEE SET SUPERSSN='RNSCSE02', DNO='5' WHERE
SSN='RNSCSE01';
UPDATE EMPLOYEE SET SUPERSSN='RNSCSE03', DNO='5' WHERE
SSN='RNSCSE02';
UPDATE EMPLOYEE SET SUPERSSN='RNSCSE04', DNO='5' WHERE
SSN='RNSCSE03';
UPDATE EMPLOYEE SET DNO='5', SUPERSSN='RNSCSE05' WHERE
SSN='RNSCSE04';
 UPDATE EMPLOYEE SET DNO='5', SUPERSSN='RNSCSE06' WHERE
SSN='RNSCSE05';
UPDATE EMPLOYEE SET DNO='5', SUPERSSN=NULL WHERE SSN='RNSCSE06';
UPDATE EMPLOYEE SET DNO='1', SUPERSSN='RNSACC02' WHERE
SSN='RNSACC01';
UPDATE EMPLOYEE SET DNO='1', SUPERSSN=NULL WHERE SSN='RNSACC02';
UPDATE EMPLOYEE SET DNO='4', SUPERSSN=NULL WHERE SSN='RNSISE01';
UPDATE EMPLOYEE SET DNO='2', SUPERSSN=NULL WHERE SSN='RNSIT01';



INSERT INTO DLOCATION VALUES ('BANGALORE', '1');
INSERT INTO DLOCATION VALUES ('BANGALORE', '2');
INSERT INTO DLOCATION VALUES ('BANGALORE', '3');
INSERT INTO DLOCATION VALUES ('MANGALORE','4');
INSERT INTO DLOCATION VALUES ('MANGALORE', '5');


INSERT INTO PROJECT VALUES (100,'IOT','BANGALORE','5');
INSERT INTO PROJECT VALUES (101,'CLOUD','BANGALORE','5');
INSERT INTO PROJECT VALUES (102,'BIGDAT','BANGALORE','5');
INSERT INTO PROJECT VALUES (103,'SENSORS','BANGALORE','3');
INSERT INTO PROJECT VALUES (104,'BANK MANAGEMENT','BANGALORE','1');
INSERT INTO PROJECT VALUES (105,'SALARY MANAGEMENT','BANGALORE','1');
INSERT INTO PROJECT VALUES (106,'OPENSTACK','BANGALORE','4');
INSERT INTO PROJECT VALUES (107,'SMART CITY','BANGALORE','2');


INSERT INTO WORKS_ON VALUES (4, 'RNSCSE01', 100);
INSERT INTO WORKS_ON VALUES (6, 'RNSCSE01', 101);
INSERT INTO WORKS_ON VALUES (8, 'RNSCSE01', 102);
INSERT INTO WORKS_ON VALUES (10,'RNSCSE02', 100);
INSERT INTO WORKS_ON VALUES (3, 'RNSCSE04', 100);
INSERT INTO WORKS_ON VALUES (4, 'RNSCSE05', 101);
INSERT INTO WORKS_ON VALUES (5, 'RNSCSE06', 102);
INSERT INTO WORKS_ON VALUES (6, 'RNSCSE03', 102);
INSERT INTO WORKS_ON VALUES (7, 'RNSECE01', 103);
INSERT INTO WORKS_ON VALUES (5, 'RNSACC01', 104);
INSERT INTO WORKS_ON VALUES (6, 'RNSACC02', 105);
INSERT INTO WORKS_ON VALUES (4, 'RNSISE01', 106);
INSERT INTO WORKS_ON VALUES (10, 'RNSIT01', 107);
```

QUERIES:

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.**

```
SQL> (SELECT DISTINCT P.PNO
  2 FROM PROJECT P, DEPARTMENT D, EMPLOYEE E WHERE E.DNO=D.DNO
  3 AND D.MGRSSN=E.SSN AND E.LNAME='SCOTT') UNION
  4 (SELECT DISTINCT P1.PNO
  5 FROM PROJECT P1, WORKS_ON W, EMPLOYEE E1 WHERE P1.PNO=W.PNO
  6 AND E1.SSN=W.SSN
  7 AND E1.LNAME='SCOTT');
```

OUTPUT:

PNO
100
101
102
103
104
105
106
107

2. Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent raise.

```
SQL> SELECT E.FNAME, E.LNAME, 1.1*E.SALARY AS INCR_SAL FROM EMPLOYEE E,
  2 WORKS_ON W, PROJECT P
  3 WHERE E.SSN=W.SSN AND W.PNO=P.PNO AND P.PNAME='IOT';
```

OUTPUT:

FNAME	LNAME	INCR_SAL
JAMES	SMITH	550000
HEARN	BAKER	770000
PAVAN	HEGDE	715000

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.

```
SQL> SELECT SUM (E. SALARY), MAX (E. SALARY), MIN (E. SALARY), AVG
(E.SALARY)
  2  FROM EMPLOYEE E, DEPARTMENT D WHERE E.DNO=D.DNO
  3 AND D. DNAME='ACCOUNTS';
```

OUTPUT:

SUM(E.SALARY)	MAX(E.SALARY)	MIN(E.SALARY)	AVG(E.SALARY)
650000	350000	300000	325000

4. Retrieve the name of each employee who works on all the projects Controlled by department number 5 (use NOT EXISTS operator).

```
SQL> SELECT E. FNAME, E.LNAME FROM EMPLOYEE E
  2 WHERE NOT EXISTS ((SELECT PNO
  3 FROM PROJECT
  4 WHERE DNO='5')
  5 MINUS (SELECT PNO FROM WORKS_ON WHERE E.SSN=SSN));
```

OUTPUT:

FNAME	LNAME
JAMES	SMITH

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.

```
SQL> SELECT D.DNO, COUNT (*)
  2  FROM DEPARTMENT D, EMPLOYEE E WHERE D.DNO=E.DNO
  3  AND E.SALARY>600000
  4  AND D.DNO IN (SELECT E1.DNO FROM EMPLOYEE E1 GROUP BY E1.DNO
HAVING COUNT (*)>5)
  5  GROUP BY D.DNO;
```

OUTPUT:

DNO	COUNT(*)
5	3