21CSL55: DBMS LABORATORY WITH MINI PROJECT

Course objectives: This course will enable students to

- ➢ Foundation knowledge in database concepts, technology and practice to groom students into well-informed database application developers.
- ➢ Strong practice in SQL programming through a variety of database problems.
- ➢ Develop database applications using front-end tools and back-end DBMS.

Database: A Database is a collection of interrelated data and a Database Management System is a a software system that enables users to define, create and maintain the database and which provides controlled access to the database

SQL: It is structured query language, basically used to pass the query to retrieve and manipulate the information from database. Depending upon the nature of query, SQL is divided into different components:

- **DDL**(Data Definition Language)
- **DML**(Data Manipulation Language)
- **DCL**(Data Control Language)

DDL: The Data Definition Language (DDL) is used to create the database (i.e. tables, keys, relationships etc), maintain the structure of the database and destroy databases and database objects.

Eg. Create, Drop, Alter, Describe, Truncate

1. **CREATE** statements: It is used to create the table.

Syntax:
CREATE TABLE table_name(columnName1 datatype(size), columnName2 datatype(size), ...);

2. **DROP statements:** To destroy an existing database, table, index, or view. If a table is dropped all records held within it are lost and cannot be recovered.

Syntax:
DROP TABLE table_name;

3. **ALTER statements:** To modify an existing database object.

• **Adding new columns:**
Syntax:

Alter table table_name Add(New_columnName1 datatype(size), New_columnName2 datatype(size),........)

- **Dropping a columns from a table : Syntax:**

  Alter table table_name DROP column columnName:

- **Modifying Existing columns:**

Syntax:

Alter table table_name Modify (columnName1 Newdatatype(Newsize));

4. **Describe statements:** To describe the structure (column and data types) of an existing database, table, index, or view.

Syntax:

DESC table_name;

5. **Truncate statements:** To destroy the data in an existing database, table, index, or view. If a table is truncated all records held within it are lost and cannot be recovered but the table structure is maintained.

Syntax :

TRUNCATE TABLE table_name;

Data Manipulation Language (DML):

- A Data Manipulation Language enables programmers and users of the database to retrieve insert, delete and update data in a database. e.g. INSERT, UPDATE, DELETE, SELECT.

INSERT: INSERT statement adds one or more records to any single table in a relational database.
Syntax:

INSERT INTO tablename VALUES (expr1,expr2. ......);

UPDATE: UPDATE statement that changes the data of one or more records in a table. Either all the rows can be updated, or a subset may be chosen using a condition.

Syntax:
UPDATE table_name SET column_name = value [, column_name = value....] [WHERE condition]

DELETE: DELETE statement removes one or more records from a table. A subset may be defined for deletion using a condition, otherwise all records are removed.

Syntax:
DELETE FROM tablename WHERE condition:

SELECT: SELECT statement returns a result set of records from one or more tables.

The select statement has optional clauses:

- WHERE specifies which rows to retrieve
- GROUP BY groups rows sharing a property so that an aggregate function can be applied to each group having group.

- HAVING selects among the groups defined by the GROUP BY clause.
- ORDER BY specifies an order in which to return the rows.

Syntax:

SELECT<attribute list> FROM<table list>
WHERE<condition> Where

- Attribute list is a list of attribute name whose values to be retrieved by the query.
- Table list is a list of table name required to process query.
- Condition is a Boolean expression that identifies the tuples to be retrieved by query.

Data Constraints are the business Rules which are enforced on the data being stored in a table are called Constraints.

Types of Data Constraints

1. I/O Constraint This type of constraint determines the speed at which data can be inserted or extracted from an Oracle table. I/O Constraints is divided into two different types
   - The Primary Key Constraint
   - The Foreign Key Constraint
2. Business rule Constraint This type of constraint is applied to data prior the data being Inserted into table columns.
   - Column level
   - Table level

The PRIMARY KEY defined at column level

Syntax:

CREATETABLEtablename
(Columnname1DATATYPE
CONSTRAINT <constraintname1>
PRIMARY KEY, Columnname2
DATATYPE, columnname3
DATATYPE, ...);

The PRIMARY KEY defined at table level

Syntax:

CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, **PRIMARY KEY (columnname1, columnname2));**

The FOREIGN KEY defined at column level

Syntax

CREATE TABLE tablename (Columnname1
tablename[(columnname)] [ON DELETE CASCADE],
columnname3 DATATYPE , ...);

DATATYPE columnname2 REFERENCES DATATYPE , The table in which FOREIGN KEY is defined is called FOREIGN TABLE or DETAIL TABLE. The table in which PRIMARY KEY is defined and referenced by FOREIGN KEY is called PRIMARY TABLE or MASTER TABLE.

ON DELETE CASCADE is set then DELETE operation in master table will trigger the DELETE operation for corresponding records in the detail table.

The FOREIGN KEY defined at table level

Syntax:
CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, PRIMARY KEY (columnname1, columnname2), FOREIGN KEY (columnname2) REFERENCES tablename2;

A CONSTRAINT can be given User Defined Name, the syntax is:
CONSTRAINT < constraint name><constraint definition>

The CHECK Constraint defined at column level
Syntax:
CREATE TABLE tablename (Columnname1 DATATYPE CHECK (logical expression), columnname2 DATATYPE, columnname3 DATATYPE,...);

The CHECK Constraint defined at table level
Syntax:
CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, CHECK (logical expression1), CHECK (logical expression2));

The UNIQUE Constraint defined at the column level
Syntax:
CREATE TABLE tablename (Columnname1 DATATYPE UNIQUE, columnname2 DATATYPE UNIQUE, columnname3 DATATYPE ...);

The UNIQUE Constraint defined at the the table level
Syntax:
CREATE TABLE tablename (Columnname1 DATATYPE, columnname2 DATATYPE, columnname3 DATATYPE, UNIQUE(columnname1));

NOT NULL constraint defined at column level :
Syntax:
CREATE TABLE tablename (Columnname1 DATATYPE NOT NULL, columnname2 DATATYPE NOT NULL, columnname3 DATATYPE,...);

Note:
The NOT NULL constraint can only be applied at column level.

ER- Diagram: It is an Entity –Relationship diagram which is used to represent the relationship between different entities. An entity is an object in the real world which is distinguishable from other objects. The overall logical structure of a database can be expressed graphically by an ER diagram, which is built up from following components.

- Rectangles: represent entity sets.
- Ellipses: represent attributes.
- Diamonds: represent relationships among entity sets.
- Lines: link attribute to entity sets and entity sets to relationships.

Mapping Cardinalities: It expresses the number of entities to which another entity can be associated via a relationship set. For a binary relationship set R between entity sets A and B. The Mapping Cardinalities must be one of the following.

- One to one
- One to many
- Many to one
- Many to many

LAB EXPERIMENTS

PART A: SQL PROGRAMMING

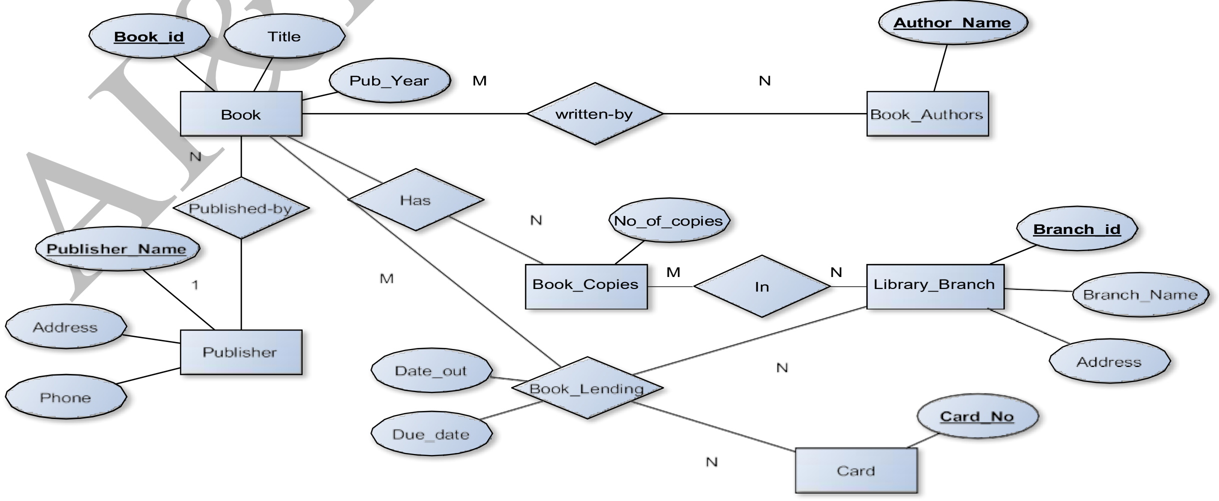
1.Consider the following schema for a Library Database:

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. **Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.**
4. **Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.**
5. **Create a view of all books and its number of copies that are currently available in the Library.**

Solution:
Entity-Relationship Diagram



Schema Diagram

Table Creation:

PUBLISHER

```
SQL> CREATE TABLE PUBLISHER(
     NAME VARCHAR(18) PRIMARY KEY,
     ADDRESS VARCHAR(10),
     PHONE VARCHAR(10));

Table created.
```

BOOK

```
SQL> CREATE TABLE BOOK(
     BOOK_ID INTEGER PRIMARY KEY,
     TITLE VARCHAR(20),
     PUBLISHER_NAME VARCHAR(20)
     PUB_YEAR NUMBER(4),
     FOREIGN KEY(PUBLISHER_NAME) REFERENCES PUBLISHER(NAME)ON DELETE
                                                               CASADE
     );

Table created.
```

BOOK_AUTHORS

```
SQL> CREATE TABLE BOOK_AUTHORS(
     BOOK_ID INTEGER,
     AUTHOR_NAME VARCHAR(20),
     PRIMARY KEY(BOOK_ID),
     FOREIGN KEY(BOOK_ID) REFERENCES BOOK(BOOK_ID) ON DELETE CASCADE);

Table created.
```

LIBRARY_BRANCH

```
SQL> CREATE TABLE LIBRARY_BRANCH(
     BRANCH_ID INTEGER PRIMARY KEY,
     BRANCH_NAME VARCHAR(18),
     ADDRESS VARCHAR(15));

Table created.
```

BOOK_COPIES

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

Table created.
```

BOOK_LENDING

```
SQL> CREATE TABLE BOOK_LENDING(
     BOOK_ID INTEGER,
     BRANCH_ID INTEGER,
     CARD_NO INTEGER,
     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,
     ); Table created.
```

Values for tables:

PUBLISHER

```
SQL>INSERT INTO PUBLISHER VALUES('PEARSON','BANGALORE','9875462530');

SQL> INSERT INTO PUBLISHER VALUES('MCGRAW','NEWDELHI','7845691234');

SQL> INSERT INTO PUBLISHER VALUES('SAPNA','BANGALORE','7845963210');
```

BOOK

```
SQL> INSERT INTO BOOK VALUES(1111,'SE','PEARSON',2005);

SQL> INSERT INTO BOOK VALUES(2222,'DBMS','MCGRAW',2004);

SQL> INSERT INTO BOOK VALUES(3333,'ANOTOMY','PEARSON',2010); SQL>

INSERT INTO BOOK VALUES(4444,'ENCYCLOPEDIA','SAPNA',2010);
```

BOOK_AUTHORS

```
SQL> INSERT INTO BOOK_AUTHORS VALUES(1111,'SOMMERVILLE');

SQL> INSERT INTO BOOK_AUTHORS VALUES(2222,'NAVATHE'); SQL>

INSERT INTO BOOK_AUTHORS VALUES(3333,'HENRY GRAY'); SQL>
INSERT INTO BOOK_AUTHORS VALUES(4444,'THOMAS');
```

LIBRARY_BRANCH

```
SQL> INSERT INTO LIBRARY_BRANCH VALUES(11,'CENTRAL TECHNICAL','MG
ROAD');
SQL> INSERT INTO LIBRARY_BRANCH VALUES(22,'MEDICAL','BH ROAD');
SQL> INSERT INTO LIBRARY_BRANCH VALUES(33,'CHILDREN','SS PURAM');
SQL> INSERT INTO LIBRARY_BRANCH VALUES(44,'SECRETARIAT','SIRAGATE');
SQL> INSERT INTO LIBRARY_BRANCH VALUES(55,'GENERAL','JAYANAGAR');
```

BOOK_COPIES

```
SQL> INSERT INTO BOOK_COPIES VALUES(1111,11,5);
SQL> INSERT INTO BOOK_COPIES VALUES(3333,22,6);
SQL> INSERT INTO BOOK_COPIES VALUES(4444,33,10);
SQL> INSERT INTO BOOK_COPIES VALUES(2222,11,12);
SQL> INSERT INTO BOOK_COPIES VALUES(4444,55,3);
```

BOOK_LENDING

```
SQL> INSERT INTO BOOK_LENDING VALUES(2222,11,1,'10-JAN-2017','20-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(3333,22,2,'09-JUL-2017','12-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(4444,55,1,'11-APR-2017','09-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(2222,11,5,'09-AUG-2017','19-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(4444,33,1,'10-JUN-2017','15-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(1111,11,1,'12-MAY-2017','10-JUN-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(3333,22,1,'10-JUL-2017','15-JUL-2017');
```

```
SQL> SELECT * FROM BOOK;

   BOOK_ID   TITLE                PUBLISHER_NAME   PUB_YEAR
  ---------- -------------------- ---------------- ----------
        1111 SE                   PEARSON                2005
        2222 DBMS                 MCGRAW                 2004
        3333 ANOTOMY              PEARSON                2010
        4444 ENCYCLOPEDIA         SAPNA                  2010

 4 rows selected.
```

```
SQL> SELECT * FROM BOOK_AUTHORS;

  BOOK_ID AUTHOR_NAME
  ------- ------------
     1111 SOMMERVILLE
     2222 NAVATHE
     3333 HENRY GRAY
     4444 THOMAS

4 rows selected.

SQL> SELECT * FROM PUBLISHER;

NAME            ADDRESS         PHONE
--------------- --------------- ----------
PEARSON         BANGALORE       9875462530
MCGRAW          NEWDELHI        7845691234
SAPNA           BANGALORE       7845963210

3 rows selected.

SQL> SELECT * FROM BOOK_COPIES;

BOOK_ID BRANCH_ID NO_OF_COPIES
------- --------- ------------
   1111        11            5
   3333        22            6
   4444        33           10
   2222        11           12
   4444        55            3

5 rows selected.

SQL> SELECT * FROM BOOK_LENDING;

  BOOK_ID BRANCH_ID CARD_NO  DATE_OUT  DUE_DATE
  ------- --------- -------- --------- ---------
     2222        11        1 10-JAN-17 20-AUG-17
     3333        22        2 09-JUL-17 12-AUG-17
     4444        55        1 11-APR-17 09-AUG-17
     2222        11        5 09-AUG-17 19-AUG-17
     4444        33        1 10-JUL-17 15-AUG-17
     1111        11        1 12-MAY-17 10-JUN-17
     3333        22        1 10-JUL-17 15-JUL-17

7 rows selected.
SQL> SELECT * FROM LIBRARY_BRANCH;

     BRANCH_ID BRANCH_NAME          ADDRESS
     --------- -------------------- ----------
            11 CENTRAL TECHNICAL     MG ROAD
            22 MEDICAL               BH ROAD
            33 CHILDREN              SS PURAM
```

```
                44 SECRETARIAT            SIRAGATE
                55 GENERAL                JAYANAGAR

5 rows selected.
```

Queries:

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

```
SELECT LB.BRANCH_NAME, B.BOOK_ID,TITLE,
      PUBLISHER_NAME,AUTHOR_NAME, NO_OF_COPIES
FROM BOOK B, BOOK_AUTHORS BA, BOOK_COPIES BC,
LIBRARY_BRANCH LB WHERE B.BOOK_ID = BA.BOOK_ID AND
     BA.BOOK_ID = BC.BOOK_ID AND
     BC.BRANCH_ID = LB.BRANCH_ID
```

```
BRANCH_NAME       BOOK_ID TITLE         PUBLISHER_NAME  AUTHOR_NAME    NO_OF_COPIES
----------------- ------- ------------- --------------- -------------- ------------
GENERAL              4444 ENCYCLOPEDIA  SAPNA           THOMAS                    3

MEDICAL              3333 ANOTOMY       PEARSON         HENRY GRAY                6

CHILDREN             4444 ENCYCLOPEDIA  SAPNA           THOMAS                   10

CENTRAL TECHNICAL    1111 SE            PEARSON         SOMMERVILLE               5

CENTRAL TECHNICAL    2222 DBMS          MCGRAW          NAVATHE                  12
```

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 '01-JAN-2017' AND '30-JUN-2017'
GROUP BY CARD_NO
HAVING COUNT(*) > 3;

  CARD_NO
  -------
        1
```

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 = '3333';

1 row deleted.
```

```
SQL> SELECT * FROM BOOK;
```

```
  BOOK_ID TITLE                 PUBLISHER_NAME      PUB_YEAR
------------ ------------------------------------------------ ------------------------------------------------  ---------------
       - --                                                       --
     1111 SE                    PEARSON                  2005
     2222 DBMS                  MCGRAW                   2004
     4444 ENCYCLOPEDIA          SAPNA                    2010

          SQL> SELECT * FROM
                BOOK_COPIES;
            BOOK_ID BRANCH_ID
                 NO_OF_COPIES
-----------------  ----------------------  ---------------------------
                            -
     1111       11          5
     4444       33         10
     2222       11         12
     4444       55          3

          SQL> SELECT * FROM
                BOOK_LENDING;

  BOOK_ID BRANCH_ID  CARD_NO DATE_OUT DUE_DATE
-----------------  -----------------------  ----------------------  ------------------ -----------
                                   -- -
                                   20-AUG-
     2222       11        1 10-JAN-17 17
                                   09-AUG-
     4444       55        1 11-APR-17 17
                                   19-AUG-
     2222       11        5 09-AUG-17 17
                                   15-AUG-
     4444       33        1 10-JUN-17 17
                                   10-JUN-
     1111       11        1 12-MAY-17 17
```

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

```
CREATE VIEW V_PUBLICATION AS
SELECT PUB_YEAR
FROM BOOK;

SELECT * FROM V_PUBLICATIONS;



  PUB_YEAR
      2004
      2005
      2010
      2010
```

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

```
CREATE VIEW BOOKS_AVAILABLE AS
SELECT B.BOOK_ID, B.TITLE, C.NO_OF_COPIES
FROM LIBRARY_BRANCH L, BOOK B, BOOK_COPIES C
WHERE B.BOOK_ID = C.BOOK_ID AND
L.BRANCH_ID=C.BRANCH_ID;

View created.

SQL> SELECT * FROM BOOKS_AVAILABLE;

    BOOK_I
    D        TITLE              NO_OF_COPIES
    -------- ------------------ ------------
    1111     SE                 5
    3333     ANOTOMY            6
    4444     ENCYCLOPEDIA       10
    2222     DBMS               12
    4444     ENCYCLOPEDIA       3
```

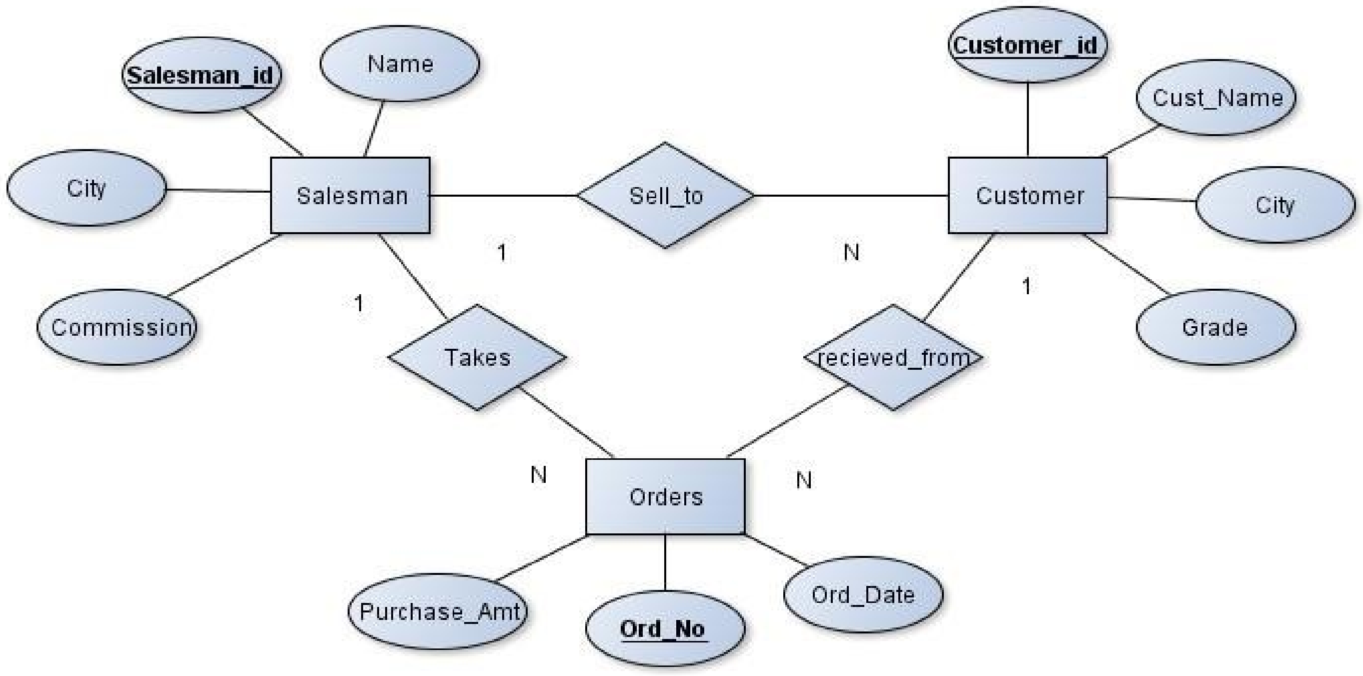
2.Consider the following schema for Order Database:

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

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.**

Solution:

Entity-Relationship Diagram



Schema Diagram

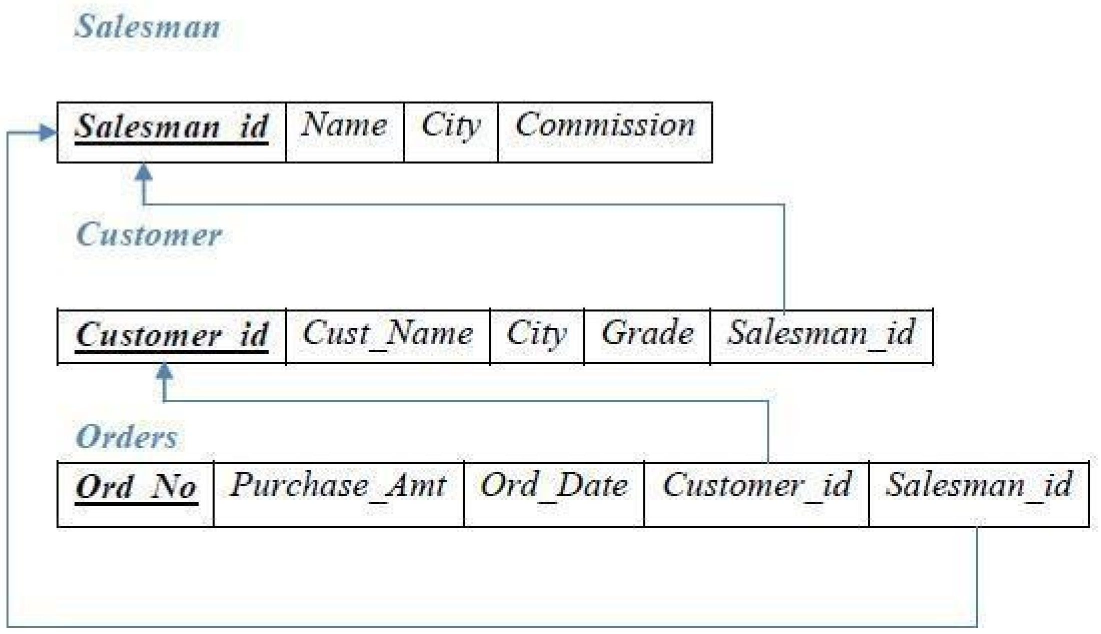
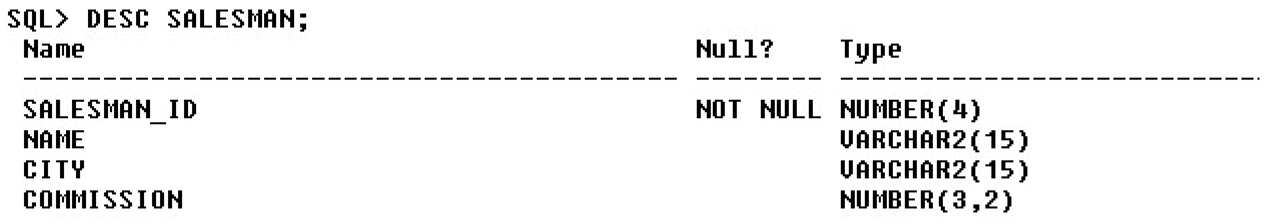


Table Creation

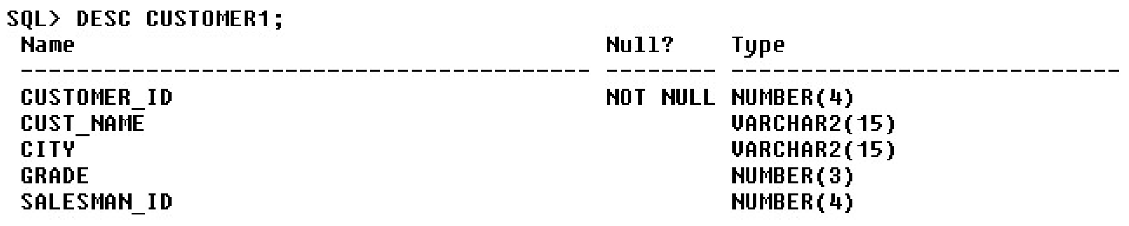
```
CREATE TABLE SALESMAN (SALESMAN_ID NUMBER (4),
NAME VARCHAR2 (20),
CITY VARCHAR2 (20),
COMMISSION VARCHAR2 (20), PRIMARY KEY(SALESMAN_ID));

CREATE TABLE CUSTOMER1 (CUSTOMER_ID NUMBER (4),
CUST_NAME VARCHAR2 (20),
CITY VARCHAR2 (20),
GRADE NUMBER (3),
SALESMAN_ID NUMBER (4),
PRIMARY KEY (CUSTOMER_ID),
FOREIGN KEY(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,
CUSTOMER_ID NUMBER (4),
SALESMAN_ID NUMBER (4),
 PRIMARY KEY (ORD_NO),
CUSTOMER_ID REFERENCES CUSTOMER1 (CUSTOMER_ID) ON DELETE
CASCADE, SALESMAN_ID REFERENCES SALESMAN (SALESMAN_ID) ON
DELETE CASCADE);
```

Table Descriptions

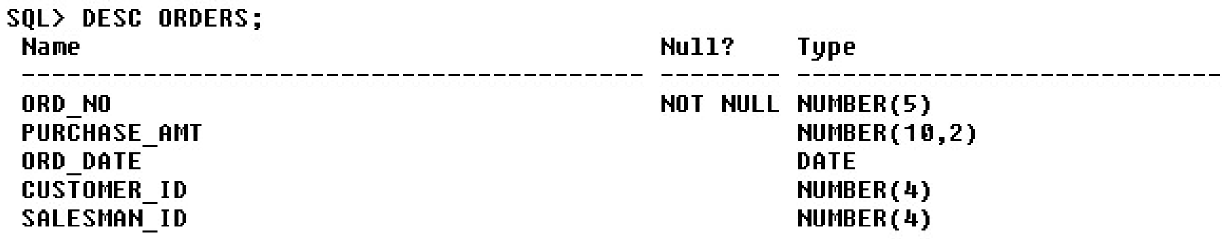
DESC SALESMAN;


```
SQL> DESC SALESMAN;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SALESMAN_ID                               NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(15)
 CITY                                               VARCHAR2(15)
 COMMISSION                                         NUMBER(3,2)
```

DESC CUSTOMER1;


```
SQL> DESC CUSTOMER1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                               NOT NULL NUMBER(4)
 CUST_NAME                                          VARCHAR2(15)
 CITY                                               VARCHAR2(15)
 GRADE                                              NUMBER(3)
 SALESMAN_ID                                        NUMBER(4)
```

DESC ORDERS;


```
SQL> DESC ORDERS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORD_NO                                    NOT NULL NUMBER(5)
 PURCHASE_AMT                                       NUMBER(10,2)
 ORD_DATE                                           DATE
 CUSTOMER_ID                                        NUMBER(4)
 SALESMAN_ID                                        NUMBER(4)
```

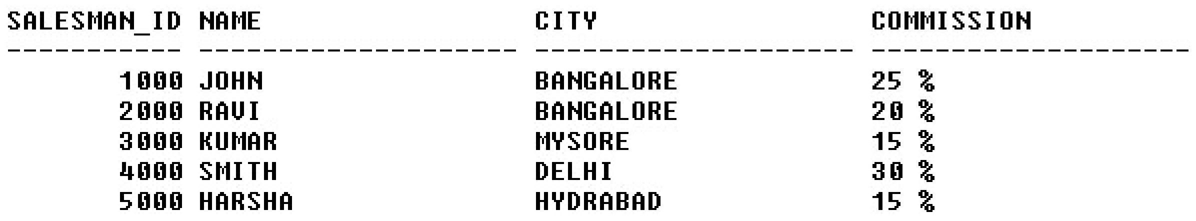
Insertion of Values to Tables

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’,’HYDRABAD’,’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);

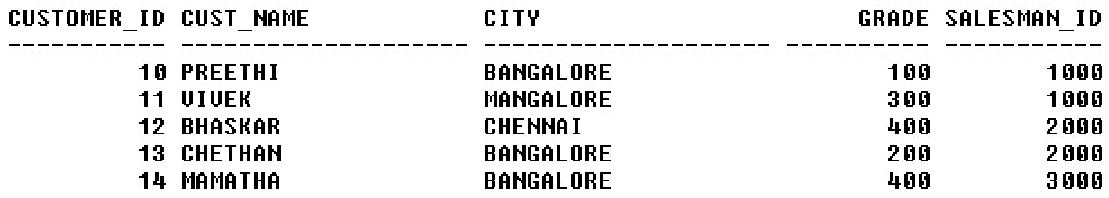
INSERT INTO ORDERS VALUES (50, 5000, ‘04-MAY-17’, 10, 1000);
INSERT INTO ORDERS VALUES (51, 450, ‘20-JAN-17’, 10, 2000);

```
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);

SELECT * FROM SALESMAN;
```



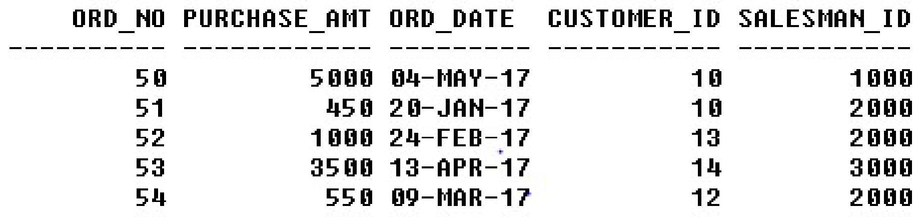
```
SALESMAN_ID NAME                 CITY                 COMMISSION
----------- -------------------- -------------------- --------------------
       1000 JOHN                 BANGALORE            25 %
       2000 RAVI                 BANGALORE            20 %
       3000 KUMAR                MYSORE               15 %
       4000 SMITH                DELHI                30 %
       5000 HARSHA               HYDRABAD             15 %
```

```
SELECT * FROM CUSTOMER1;
```



```
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
```

```
SELECT * FROM ORDERS;
```

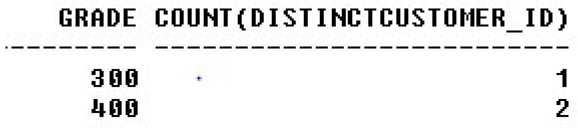


```
    ORD_NO PURCHASE_AMT ORD_DATE  CUSTOMER_ID SALESMAN_ID
---------- ------------ --------- ----------- -----------
        50         5000 04-MAY-17          10        1000
        51          450 20-JAN-17          10        2000
        52         1000 24-FEB-17          13        2000
        53         3500 13-APR-17          14        3000
        54          550 09-MAR-17          12        2000
```

<u>**Queries:**</u>

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

```
SELECT GRADE, COUNT (DISTINCT CUSTOMER_ID)
FROM CUSTOMER1
GROUP BY GRADE
HAVING GRADE > (SELECT AVG(GRADE)
FROM CUSTOMER1
WHERE CITY='BANGALORE');
```



```
   GRADE COUNT(DISTINCTCUSTOMER_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 A
 WHERE 1 < (SELECT COUNT (*)
           FROM CUSTOMER1
 WHERE SALESMAN_ID=A.SALESMAN_ID);
```

```
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.)**

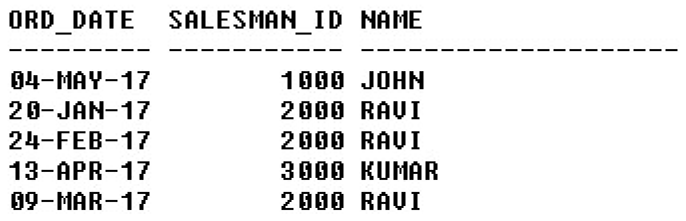
```
SELECT SALESMAN.SALESMAN_ID, NAME, CUST_NAME, COMMISSION
FROM SALESMAN, CUSTOMER1
WHERE SALESMAN.CITY = CUSTOMER1.CITY
UNION
SELECT SALESMAN_ID, NAME, 'NO MATCH', COMMISSION
FROM SALESMAN
WHERE NOT CITY = ANY
 (SELECT CITY
  FROM CUSTOMER1)
ORDER BY 2 DESC;
```

```
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.**

```
CREATE VIEW ELITSALESMAN AS
SELECT B.ORD_DATE, A.SALESMAN_ID, A.NAME
FROM SALESMAN A, ORDERS B
```

```
WHERE A.SALESMAN_ID = B.SALESMAN_ID
AND B.PURCHASE_AMT=(SELECT MAX (PURCHASE_AMT)
                              FROM ORDERS C
                                    WHERE C.ORD_DATE = B.ORD_DATE);
```



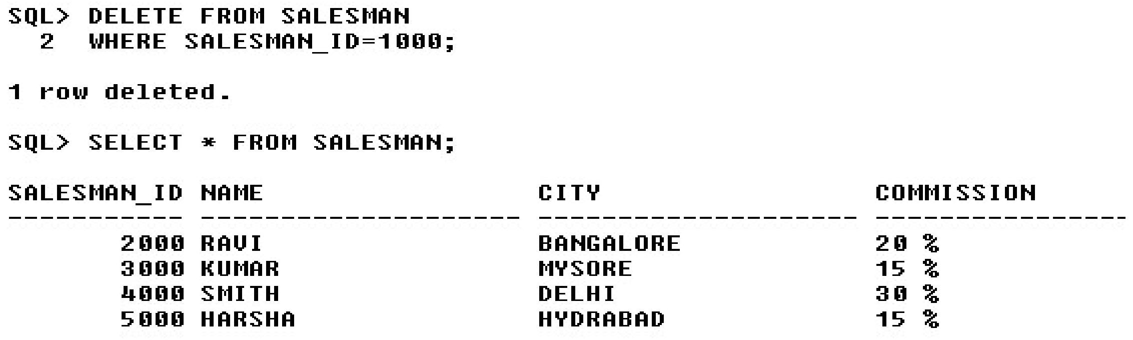
```
ORD_DATE  SALESMAN_ID NAME
--------- ----------- --------------------
04-MAY-17        1000 JOHN
20-JAN-17        2000 RAVI
24-FEB-17        2000 RAVI
13-APR-17        3000 KUMAR
09-MAR-17        2000 RAVI
```

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:

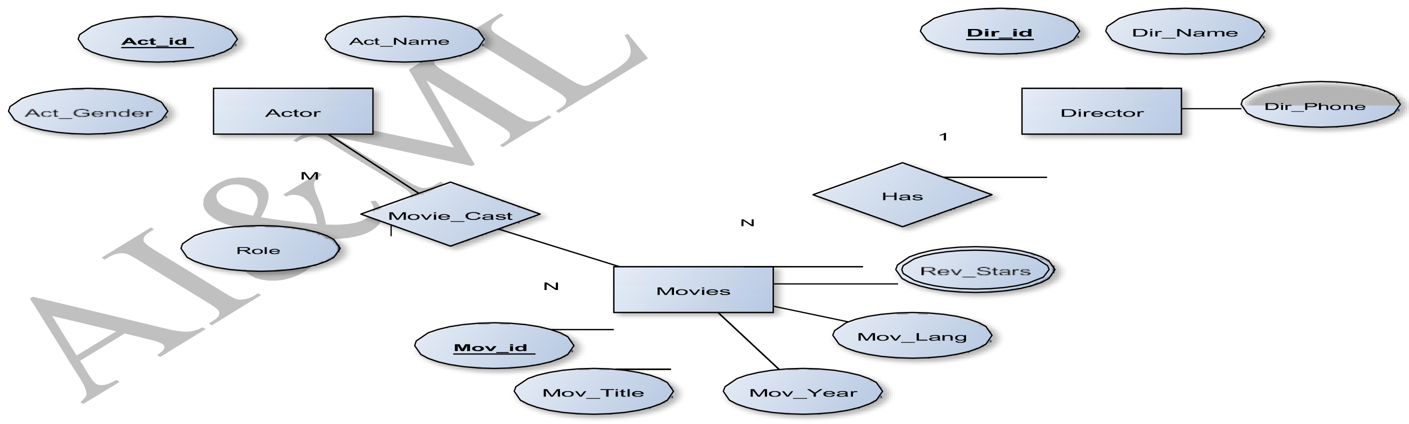
   Use ON DELETE SET NULL at the end of foreign key definitions while creating child table customers and then executes the following:

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

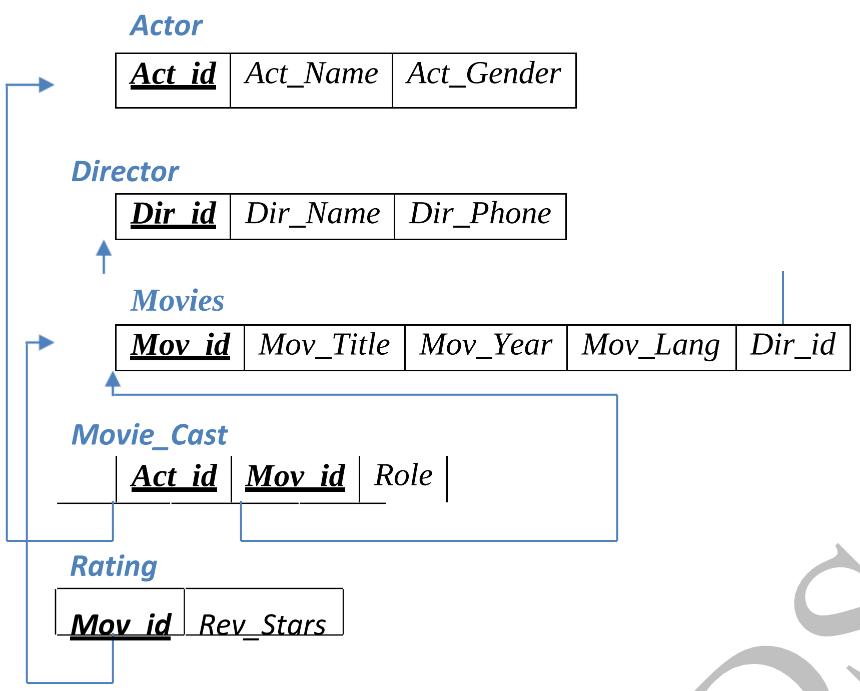


```
SQL> DELETE FROM SALESMAN
  2  WHERE SALESMAN_ID=1000;

1 row deleted.

SQL> SELECT * FROM SALESMAN;

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

Entity-Relationship Diagram



Schema Diagram



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*

```
-- CREATION OF TABLES


 -- ACTOR

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

        -- DIRECTOR

        CREATE TABLE DIRECTOR (
        DIR_ID INTEGER,
        DIR_NAME VARCHAR (20),
        DIR_PHONE INTEGER,

        PRIMARY KEY (DIR_ID));


        -- MOVIES


        CREATE TABLE MOVIES (

        MOV_ID INTEGER,

        MOV_TITLE VARCHAR (25),

        MOV_YEAR INTEGER,

        MOV_LANG VARCHAR (12),

        DIR_ID INTEGER,

        PRIMARY KEY (MOV_ID),

        FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR (DIR_ID));


        -- MOVIE_CAST


        CREATE TABLE MOVIE_CAST (

        ACT_ID INTEGER,

        MOV_ID INTEGER,

        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));


        -- RATING


        CREATE TABLE RATING (

        MOV_ID INTEGER,

        REV_STARS VARCHAR (25),
        PRIMARY KEY (MOV_ID),
        FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));



                -- VALUES FOR TABLES

        -- ACTOR

        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');

        -- DIRECTOR

        INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 875161100);
        INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 776613891);
        INSERT INTO DIRECTOR VALUES (62,'FARAN', 998677653);
        INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG', 898977653);

        -- MOVIES

        INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELUGU', 60);
        INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELUGU', 60);
        INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);
        INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);

        -- MOVIE_CAST
```

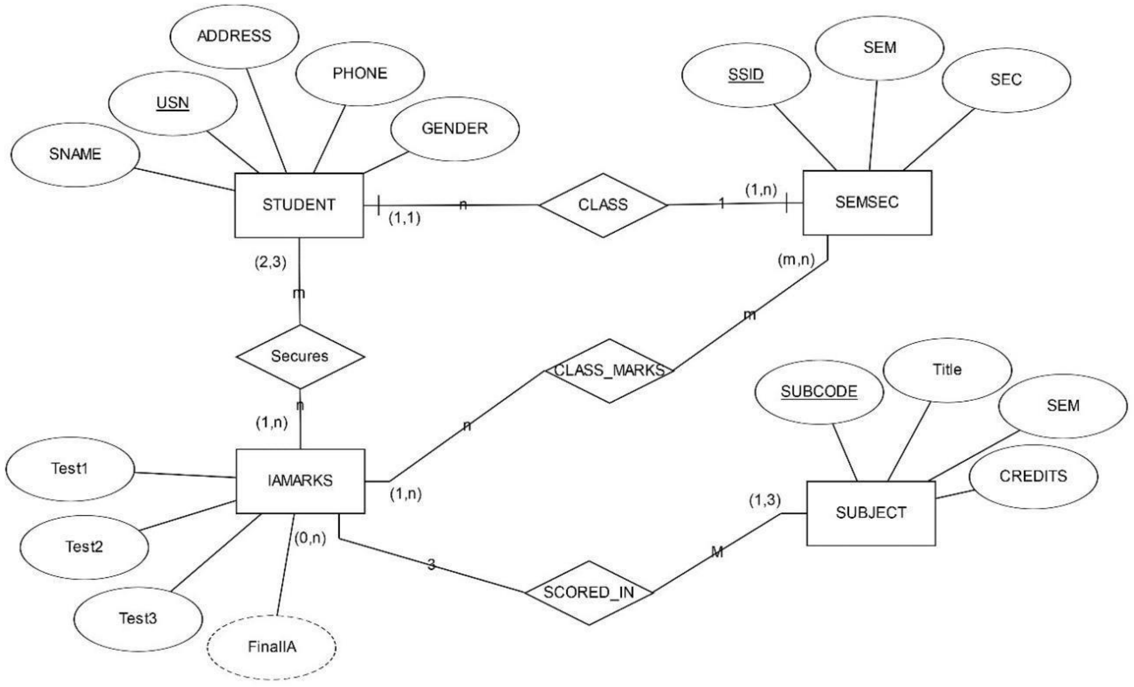
```
INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');

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');


-- RATING


INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4);



        -- DISPLAYING TABLES


-- ACTOR

SELECT * FROM ACTOR;


-- DIRECTOR

SELECT * FROM DIRECTOR;;


-- MOVIES

SELECT * FROM MOVIES;


-- MOVIE_CAST

SELECT * FROM MOVIE_CAST;


-- RATING
```

```
SELECT * FROM RATING;


        -- QUERIES


-- 1) 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');



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


SELECT MOV_TITLE

FROM MOVIES M, MOVIE_CAST MV

WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN(SELECT ACT_ID

FROM MOVIE_CAST GROUP BY ACT_ID

HAVING COUNT(ACT_ID)>1)

GROUP BY MOV_TITLE

HAVING COUNT(*) > 1;



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


SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
```

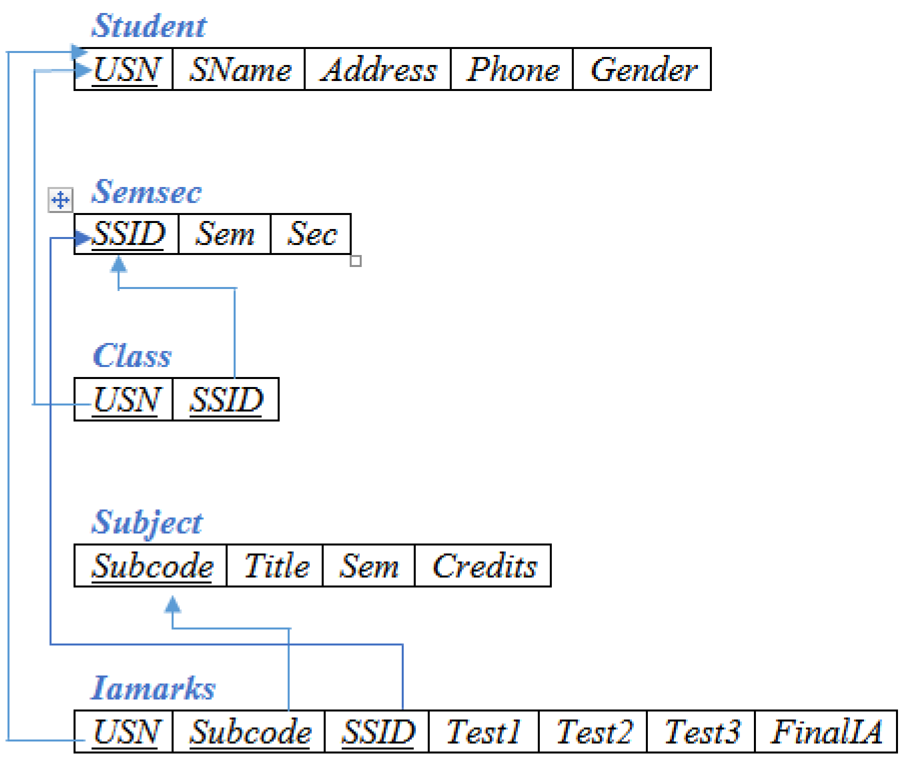
```sql
FROM ACTOR A
JOIN MOVIE_CAST C
ON A.ACT_ID=C.ACT_ID
JOIN MOVIES M
ON C.MOV_ID=M.MOV_ID
WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;

-- OR

SELECT A.ACT_NAME, A.ACT_NAME, C.MOV_TITLE, C.MOV_YEAR
FROM ACTOR A, MOVIE_CAST B, MOVIES C
WHERE A.ACT_ID=B.ACT_ID
AND B.MOV_ID=C.MOV_ID
AND C.MOV_YEAR NOT BETWEEN 2000 AND 2015;


-- 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 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;


-- 5) Update rating of all movies directed by 'Steven Spielberg' to 5KL
```

```
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'));
```

Entity - Relationship Diagram



Schema Diagram



```
-- CREATION OF TABLES

    -- STUDENT

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

    -- SEMSEC

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

    -- CLASS

    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));

    -- SUBJECT

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

    -- IAMARKS

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


            -- VALUES FOR TABLES

    -- STUDENT

    INSERT INTO STUDENT VALUES ('1RN13CS020','AKSHAY','BELAGAVI',887788112,'M');
    INSERT INTO STUDENT VALUES ('1RN13CS062','SANDHYA','BENGALURU',772282991,'F');
    INSERT INTO STUDENT VALUES ('1RN13CS091','TEESHA','BENGALURU',771231231,'F');
    INSERT INTO STUDENT VALUES ('1RN13CS066','SUPRIYA','MANGALURU',887788112,'F');
    INSERT INTO STUDENT VALUES ('1RN14CS010','ABHAY','BENGALURU',990021120,'M');
    INSERT INTO STUDENT VALUES
('1RN14CS032','BHASKAR','BENGALURU',992321109,'M');
    INSERT INTO STUDENT VALUES ('1RN14CS025','ASMI','BENGALURU', 789473737,'F');
    INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 984509134,'M');
    INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE',769677211,'F');
    INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 994485012,'M');
    INSERT INTO STUDENT VALUES
('1RN15CS091','SANTOSH','MANGALURU',881233220,'M');
    INSERT INTO STUDENT VALUES ('1RN16CS045','ISMAIL','KALBURGI',990023221,'M');
    INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA',990554221,'F');
    INSERT INTO STUDENT VALUES
('1RN16CS122','VINAYAKA','CHIKAMAGALUR',880088001,'M');
```

```
-- SEMSEC

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');

-- CLASS

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');

-- SUBJECT

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);

        -- IAMARKS

        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);


                -- DISPLAYING TABLES

        -- STUDENT
        SELECT * FROM STUDENT;

        -- SEMSEC
        SELECT * FROM SEMSEC;

        -- CLASS
        SELECT * FROM CLASS;

        -- SUBJECT
        SELECT * FROM SUBJECT;

        -- IAMARKS
```

```
SELECT * FROM IAMARKS;



        -- QUERIES

-- 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';

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

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

-- 3) Create a view of Test1 marks of student USN '1RN13CS091' in all subjects.

CREATE VIEW STU_TEST1_MARKS_VIEW
AS
SELECT TEST1, SUBCODE
FROM IAMARKS
WHERE USN = '1RN13CS091';

SELECT * FROM STU_TEST1_MARKS_VIEW;

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

UPDATE IAMARKS
SET FINALIA=GREATEST(TEST1+TEST2,TEST2+TEST3,TEST1+TEST3)/2;
SELECT * FROM IAMARKS;


-- 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 8 th semester A, B, and C section students.

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

Program 5

```
 -- CREATION OF TABLES

    -- EMPLOYEE

    CREATE TABLE EMPLOYEE (
    SSN VARCHAR(10) PRIMARY KEY,
    FNAME VARCHAR(20),
    LNAME VARCHAR(20),
    ADDRESS VARCHAR(25),
    SEX CHAR(1),
    SALARY INTEGER,
    SUPERSSN VARCHAR(10),
    DNO VARCHAR(10),
    FOREIGN KEY(SUPERSSN) REFERENCES EMPLOYEE(SSN));

    -- DEPARTMENT

    CREATE TABLE DEPARTMENT(
    DNO VARCHAR(10) PRIMARY KEY,
    DNAME VARCHAR(20),
    MGRSTARTDATE DATE,
    MGRSSN VARCHAR(10),
    FOREIGN KEY(MGRSSN) REFERENCES EMPLOYEE(SSN));



    -- ALTERING EMPLPOYEE
```

```
ALTER TABLE EMPLOYEE

ADD FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DNO);



-- DLOCATION


CREATE TABLE DLOCATION(

DLOC VARCHAR(20),

DNO VARCHAR(10) PRIMARY KEY,

FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DNO));


-- PROJECT


CREATE TABLE PROJECT(

PNO VARCHAR(20),

PNAME VARCHAR(20),

PLOCATION VARCHAR(20),

DNO VARCHAR(10),

PRIMARY KEY(PNO),

FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DNO));


-- WORKS_ON


CREATE TABLE WORKS_ON(

HOURS VARCHAR(10),

SSN VARCHAR(10),

PNO VARCHAR(20),

PRIMARY KEY(SSN,PNO),
        FOREIGN KEY(SSN) REFERENCES EMPLOYEE(SSN),
        FOREIGN KEY(PNO) REFERENCES PROJECT(PNO));

                -- VALUES FOR TABLES

        -- EMPLOYEE

        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);

        -- DEPARTMENT

        INSERT INTO DEPARTMENT VALUES ('1','ACCOUNTS','2001-01-01','RNSACC02');
INSERT INTO DEPARTMENT VALUES ('2','IT','2016-07-01','RNSIT01');

INSERT INTO DEPARTMENT VALUES ('3','ECE','2008-06-01','RNSECE01');

INSERT INTO DEPARTMENT VALUES ('4','ISE','2015-07-01','RNSISE01');

INSERT INTO DEPARTMENT VALUES ('5','CSE','2002-06-01','RNSCSE05');


-- UPDATING SUPERSSN AND DNO FOR EMPLPOYEE TABLE


UPDATE EMPLOYEE SET SUPERSSN = NULL      , DNO='3' WHERE SSN='RNSECE01';

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 SUPERSSN = 'RNSCSE05', DNO='5' WHERE SSN = 'RNSCSE04';

UPDATE EMPLOYEE SET SUPERSSN = 'RNSCSE06', DNO='5' WHERE SSN = 'RNSCSE05';

UPDATE EMPLOYEE SET SUPERSSN = NULL      , DNO='5' WHERE SSN = 'RNSCSE06';

UPDATE EMPLOYEE SET SUPERSSN = 'RNSACC02', DNO='1' WHERE SSN = 'RNSACC01';

UPDATE EMPLOYEE SET SUPERSSN = NULL      , DNO='1' WHERE SSN = 'RNSACC02';

UPDATE EMPLOYEE SET SUPERSSN = NULL      , DNO='4' WHERE SSN = 'RNSISE01';

UPDATE EMPLOYEE SET SUPERSSN = NULL      , DNO='2' WHERE SSN = 'RNSIT01';


-- DLOCATION


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');


-- PROJECT
```

```
INSERT INTO PROJECT VALUES (100,'IOT','BANGALORE','5');

INSERT INTO PROJECT VALUES (101,'CLOUD','BANGALORE','5');

INSERT INTO PROJECT VALUES (102,'BIGDATA','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');


-- WORKS_ON


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);


        -- DISPLAYING TABLES


-- EMPLOYEE

SELECT * FROM EMPLOYEE;
```

```
-- DEPARTMENT

SELECT * FROM DEPARTMENT ;

-- DLOCATION

SELECT * FROM DLOCATION ;

-- PROJECT

SELECT * FROM PROJECT ;

-- WORKS_ON

SELECT * FROM WORKS_ON;


      -- 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.


(SELECT DISTINCT P.PNO

FROM PROJECT P, DEPARTMENT D, EMPLOYEE E WHERE E.DNO=D.DNO

AND D.MGRSSN=E.SSN AND E.LNAME='SCOTT') UNION

(SELECT DISTINCT P1.PNO

FROM PROJECT P1, WORKS_ON W, EMPLOYEE E1 WHERE P1.PNO=W.PNO

AND E1.SSN=W.SSN

AND E1.LNAME='SCOTT');


-- 2) Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.


SELECT E.FNAME, E.LNAME, 1.1*E.SALARY AS INCR_SAL FROM EMPLOYEE E, WORKS_ON W, PROJECT P

WHERE E.SSN=W.SSN AND W.PNO=P.PNO AND P.PNAME='IOT';
```

```
-- 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(E.SALARY), MAX(E.SALARY), MIN(E.SALARY), AVG(E.SALARY)
FROM EMPLOYEE E, DEPARTMENT D WHERE E.DNO=D.DNO
AND D.DNAME='ACCOUNTS';


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

SELECT E.FNAME,E.LNAME FROM EMPLOYEE E WHERE NOT EXISTS
(SELECT PNO FROM PROJECT P WHERE DNO=5 AND PNO NOT IN
(SELECT PNO FROM WORKS_ON W WHERE E.SSN=SSN));

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