DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

Udayapura, Kanakapura Road, Opp. Art of Living, Bangalore – 560082

(Affliated to VTU, Belagavi, Approved by AICTE, New Delhi)

Accredited by NBA and NAAC (A+)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2023-2024

DBMS LAB MANUAL (21CSL55)

Compiled by:

Prof. Lakshmi M.R
Prof. Manasa Sandeep
Prof. Shylaja B.

Dr. Kavitha C
HOD, CSE, DSATM

Dr. M Ravishankar
Principal, DSATM

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 Systemis 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 andmanipulate 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 isdropped 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 existingdatabase, 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 relationaldatabase.
Syntax:

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

UPDATE: UPDATE statement that changes the data of one or more records in a table. Eitherall 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 bedefined 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 tableare 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 table name (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 table name (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 relationshipbetween 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 beassociated 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

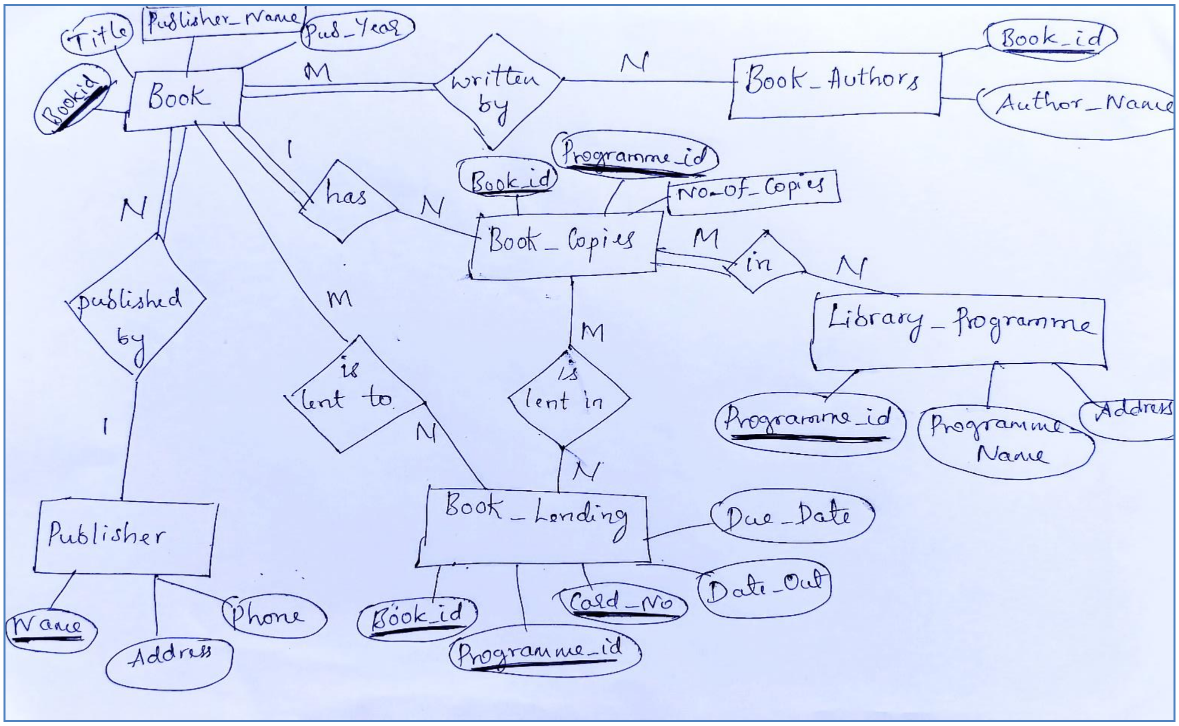
1. **Consider the following schema for a Library Database:**
BOOK(Book_id, Title, Publisher_Name, Pub_Year)
BOOK_AUTHORS(Book_id, Author_Name)
PUBLISHER(Name, Address, Phone)
BOOK_COPIES(Book_id, Programme_id, No-of_Copies)
BOOK_LENDING(Book_id, Programme_id, Card_No, Date_Out, Due_Date)
LIBRARY_PROGRAMME(Programme_id, Programme_Name, Address)

Write SQL queries to

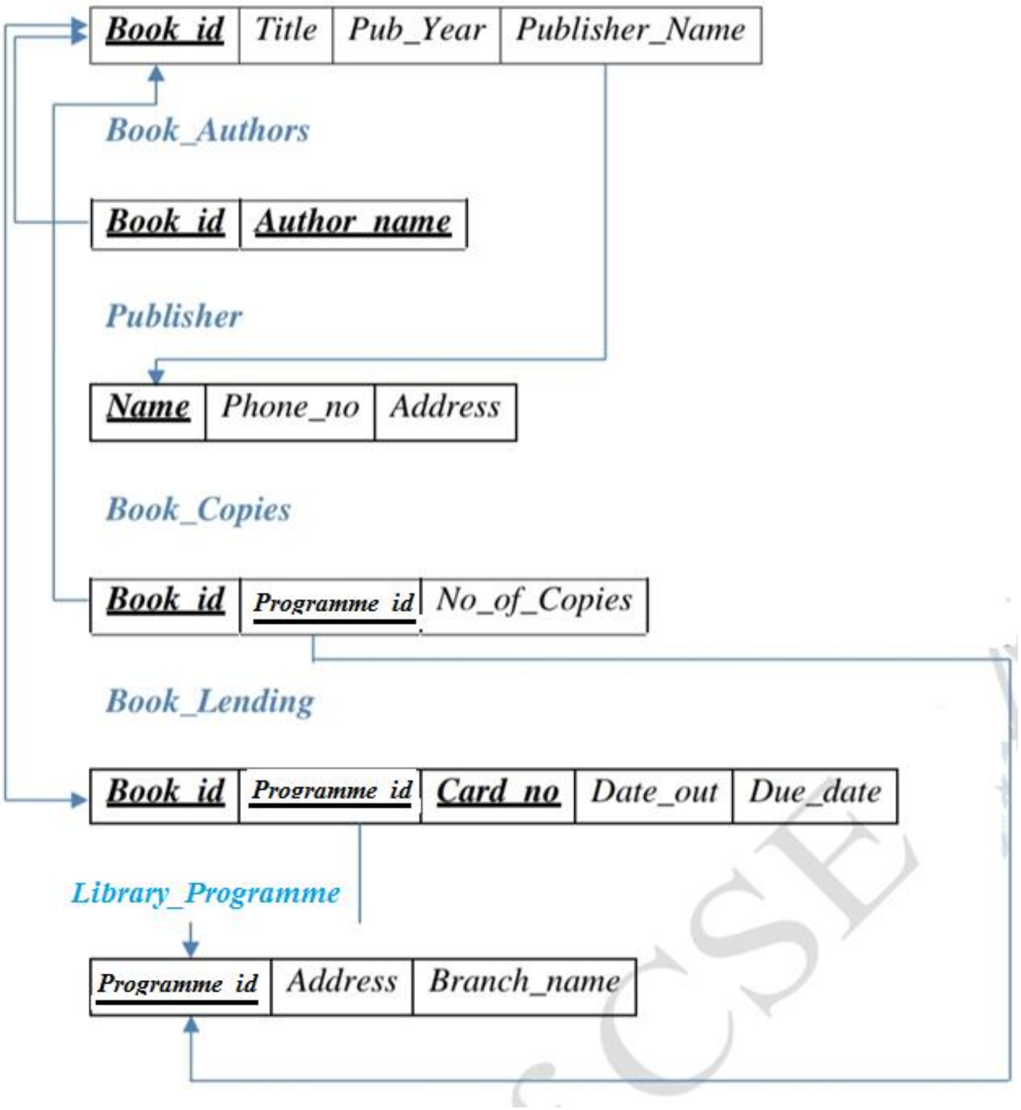
1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each Programme, etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.
3. Delete a book in BOOK table. Update the contents of other tables 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



Step 1: Create Database

```
create database Library;
use Library;
```

Step 2: Create Tables

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

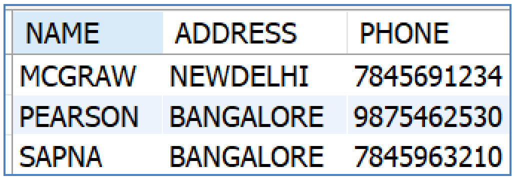
```
CREATE TABLE BOOK(
BOOK_ID INTEGER PRIMARY KEY,
TITLE VARCHAR(20),
PUBLISHER_NAME VARCHAR(20),
PUB_YEAR INT(4),
FOREIGN KEY(PUBLISHER_NAME) REFERENCES PUBLISHER(NAME) ON
DELETE CASCADE
);

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);

CREATE TABLE LIBRARY_PROGRAMME(
PROGRAMME_ID INTEGER PRIMARY KEY,
PROGRAMME_NAME VARCHAR(18),
ADDRESS VARCHAR(15));

CREATE TABLE BOOK_COPIES(
BOOK_ID INTEGER,
PROGRAMME_ID INTEGER,
NO_OF_COPIES INTEGER,
FOREIGN KEY(BOOK_ID) REFERENCES BOOK(BOOK_ID) ON DELETE CASCADE,
FOREIGN KEY(PROGRAMME_ID) REFERENCES LIBRARY_PROGRAMME(PROGRAMME_ID) ON
DELETE CASCADE,
PRIMARY KEY(BOOK_ID,PROGRAMME_ID));

CREATE TABLE BOOK_LENDING(
BOOK_ID INTEGER,
PROGRAMME_ID INTEGER,
CARD_NO INTEGER,
DATE_OUT DATE,
DUE_DATE DATE,
PRIMARY KEY(BOOK_ID,PROGRAMME_ID,CARD_NO),
FOREIGN KEY(BOOK_ID) REFERENCES BOOK(BOOK_ID) ON DELETE CASCADE,
FOREIGN KEY(PROGRAMME_ID) REFERENCES LIBRARY_PROGRAMME(PROGRAMME_ID) ON
DELETE CASCADE
);
```

Step 3: Insert Values into Tables

```
INSERT INTO PUBLISHER VALUES('PEARSON','BANGALORE','9875462530');
INSERT INTO PUBLISHER VALUES('MCGRAW','NEWDELHI','7845691234');
INSERT INTO PUBLISHER VALUES('SAPNA','BANGALORE','7845963210');

INSERT INTO BOOK VALUES(1111,'SE','PEARSON',2005);
INSERT INTO BOOK VALUES(2222,'DBMS','MCGRAW',2004);
INSERT INTO BOOK VALUES(3333,'ANOTOMY','PEARSON',2010);
INSERT INTO BOOK VALUES(4444,'ENCYCLOPEDIA','SAPNA',2010);

INSERT INTO BOOK_AUTHORS VALUES(1111,'SOMMERVILLE');
INSERT INTO BOOK_AUTHORS VALUES(2222,'NAVATHE');
INSERT INTO BOOK_AUTHORS VALUES(3333,'HENRY GRAY');
INSERT INTO BOOK_AUTHORS VALUES(4444,'THOMAS');

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

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

INSERT INTO BOOK_LENDING VALUES(2222,11,1,'2017-01-10','2017-08-20');
INSERT INTO BOOK_LENDING VALUES(3333,22,2,'2017-07-09','2017-08-12');
INSERT INTO BOOK_LENDING VALUES(4444,55,1,'2017-04-11','2017-08-09');
INSERT INTO BOOK_LENDING VALUES(2222,11,5,'2017-08-09','2017-08-19');
INSERT INTO BOOK_LENDING VALUES(4444,33,1,'2017-06-10','2017-08-15');
INSERT INTO BOOK_LENDING VALUES(1111,11,1,'2017-05-12','2017-06-10');
INSERT INTO BOOK_LENDING VALUES(3333,22,1,'2017-07-10','2017-07-15');
```

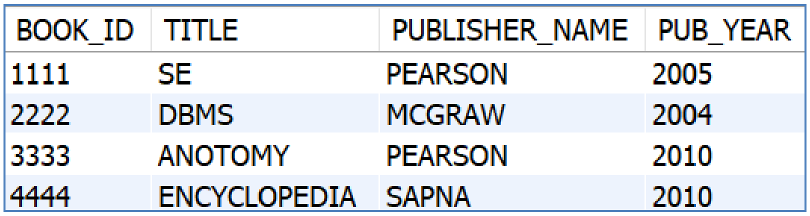
Step 4: Display table contents

```
SELECT * FROM BOOK;
```



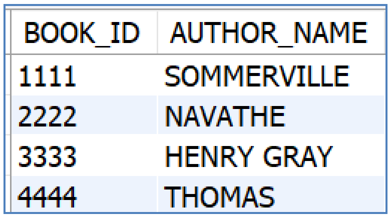
NAME	ADDRESS	PHONE
MCGRAW	NEWDELHI	7845691234
PEARSON	BANGALORE	9875462530
SAPNA	BANGALORE	7845963210

```
SELECT * FROM BOOK;
```



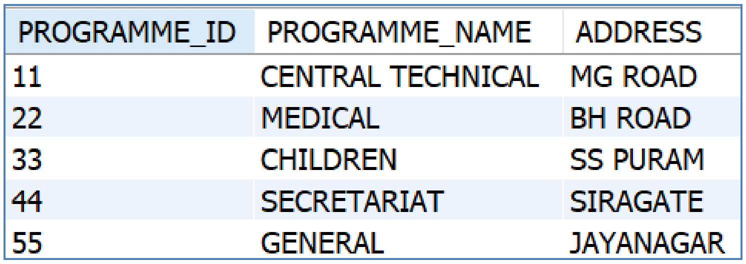
BOOK_ID	TITLE	PUBLISHER_NAME	PUB_YEAR
1111	SE	PEARSON	2005
2222	DBMS	MCGRAW	2004
3333	ANOTOMY	PEARSON	2010
4444	ENCYCLOPEDIA	SAPNA	2010

```
SELECT * FROM BOOK_AUTHORS;
```



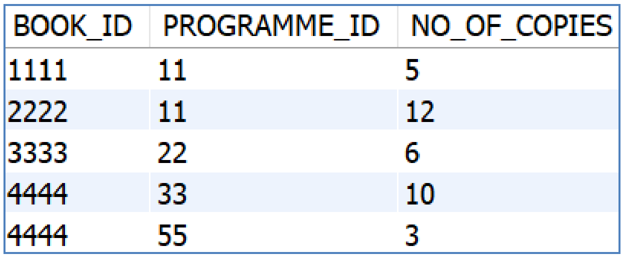
BOOK_ID	AUTHOR_NAME
1111	SOMMERVILLE
2222	NAVATHE
3333	HENRY GRAY
4444	THOMAS

```
SELECT * FROM LIBRARY_PROGRAMME;
```



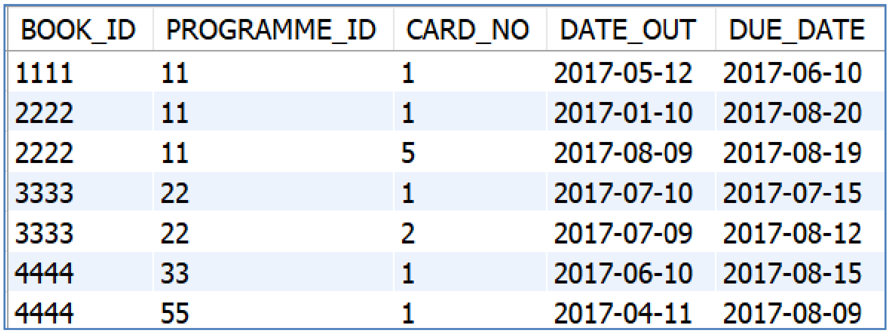
PROGRAMME_ID	PROGRAMME_NAME	ADDRESS
11	CENTRAL TECHNICAL	MG ROAD
22	MEDICAL	BH ROAD
33	CHILDREN	SS PURAM
44	SECRETARIAT	SIRAGATE
55	GENERAL	JAYANAGAR

```
SELECT * FROM BOOK_COPIES;
```



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

```
SELECT * FROM BOOK_LENDING;
```



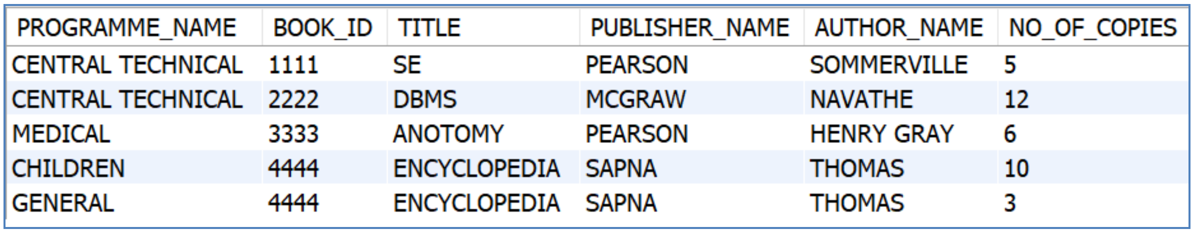
BOOK_ID	PROGRAMME_ID	CARD_NO	DATE_OUT	DUE_DATE
1111	11	1	2017-05-12	2017-06-10
2222	11	1	2017-01-10	2017-08-20
2222	11	5	2017-08-09	2017-08-19
3333	22	1	2017-07-10	2017-07-15
3333	22	2	2017-07-09	2017-08-12
4444	33	1	2017-06-10	2017-08-15
4444	55	1	2017-04-11	2017-08-09

Step 5: Execute Queries:

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

```
SELECT PROGRAMME_NAME, B.BOOK_ID,TITLE,
PUBLISHER_NAME,AUTHOR_NAME, NO_OF_COPIES
FROM BOOK B, BOOK_AUTHORS BA, BOOK_COPIES BC,
LIBRARY_PROGRAMME LB WHERE B.BOOK_ID = BA.BOOK_ID AND
BA.BOOK_ID = BC.BOOK_ID AND
BC.PROGRAMME_ID = LB.PROGRAMME_ID;
```

Output:

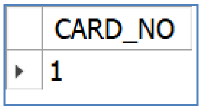


PROGRAMME_NAME	BOOK_ID	TITLE	PUBLISHER_NAME	AUTHOR_NAME	NO_OF_COPIES
CENTRAL TECHNICAL	1111	SE	PEARSON	SOMMERVILLE	5
CENTRAL TECHNICAL	2222	DBMS	MCGRAW	NAVATHE	12
MEDICAL	3333	ANOTOMY	PEARSON	HENRY GRAY	6
CHILDREN	4444	ENCYCLOPEDIA	SAPNA	THOMAS	10
GENERAL	4444	ENCYCLOPEDIA	SAPNA	THOMAS	3

/* 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 '2017-01-01' AND '2017-06-30'
GROUP BY CARD_NO
HAVING COUNT(*) > 3;
```

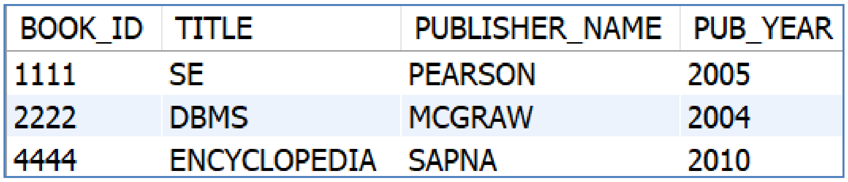
Output:



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';

SELECT * FROM BOOK;
```

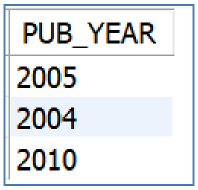
Output:



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

```
/* 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_PUBLICATION;
```

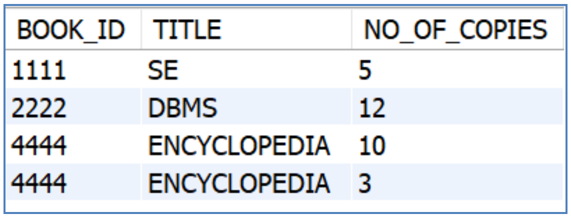
Output:



PUB_YEAR
2005
2004
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_PROGRAMME L, BOOK B, BOOK_COPIES C
WHERE B.BOOK_ID = C.BOOK_ID AND L.PROGRAMME_ID=C.PROGRAMME_ID;

SELECT * FROM BOOKS_AVAILABLE;
```

Output:



BOOK_ID	TITLE	NO_OF_COPIES
1111	SE	5
2222	DBMS	12
4444	ENCYCLOPEDIA	10
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*)

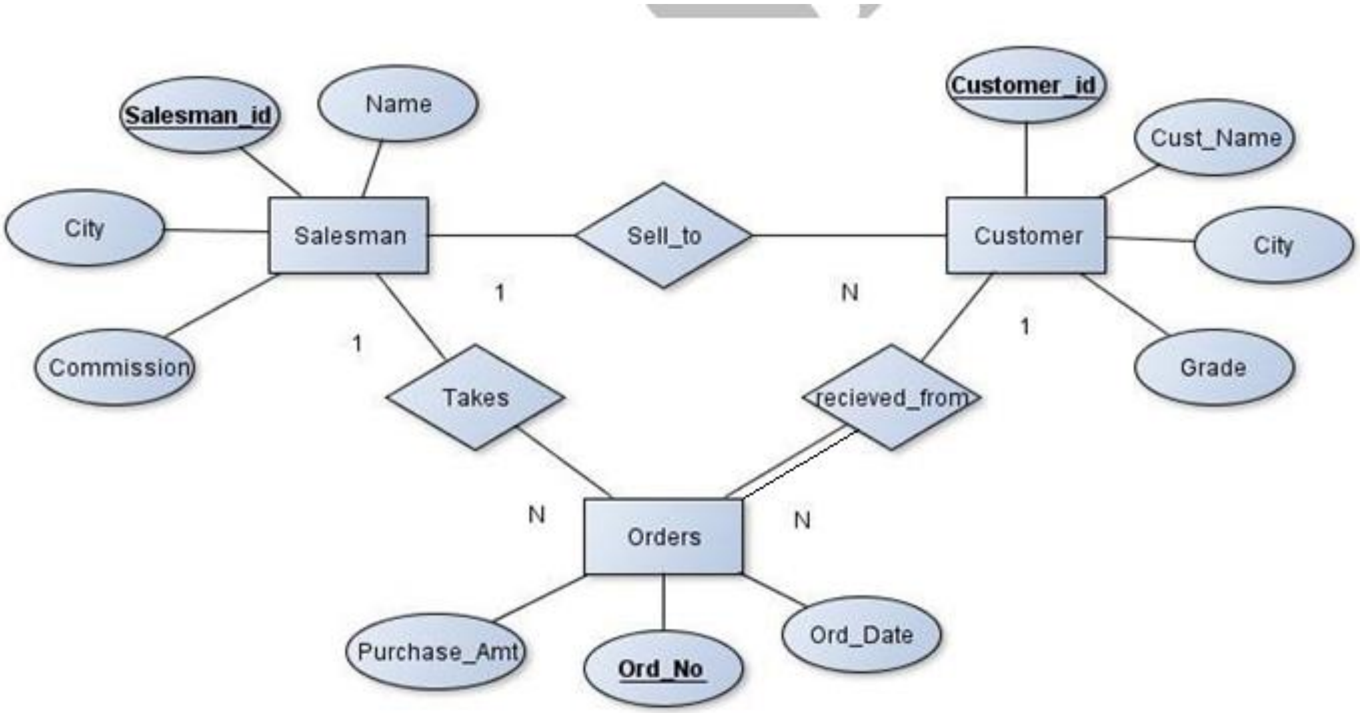
ORDERS (*Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id*)

Write SQL queries to

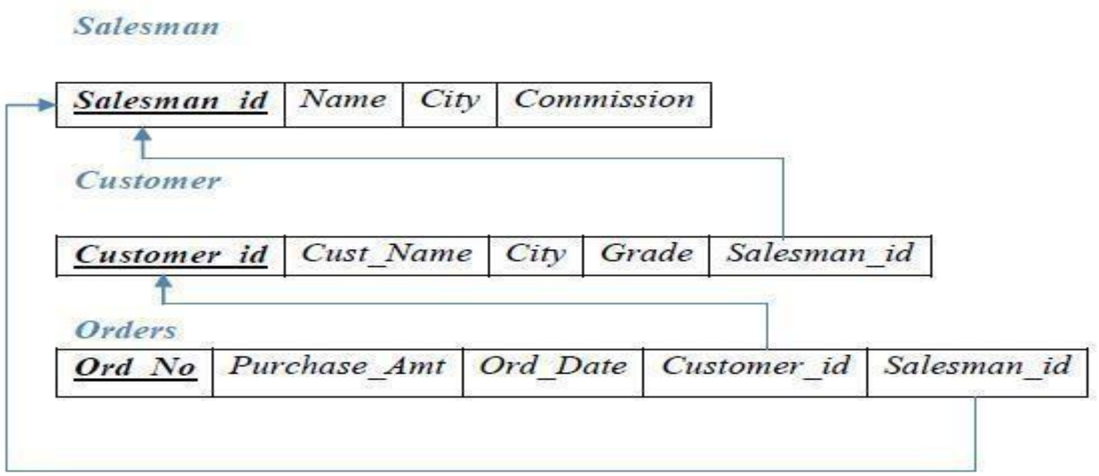
1. Count the customers with grades above Bangalore's average.
2. Find the name and numbers of all salesmen who had more than one customer.
3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)
4. Create a view that finds the salesman who has 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



Step 1: Create Database

```
CREATE DATABASE ORDERS11;
USE ORDERS11;
```

Step 2: Create Tables

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

CREATE TABLE CUSTOMER (
CUSTOMER_ID INT (4),
CUST_NAME VARCHAR (20),
CITY VARCHAR (20),
GRADE INT (3),
SALESMAN_ID INT (4),
PRIMARY KEY (CUSTOMER_ID),
FOREIGN KEY(SALESMAN_ID) REFERENCES SALESMAN (SALESMAN_ID) ON DELETE SET
NULL);
```

```
CREATE TABLE ORDERS (
ORD_NO INT(5),
PURCHASE_AMT FLOAT(10, 2),
ORD_DATE DATE,
CUSTOMER_ID INT (4),
SALESMAN_ID INT (4),
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);
```

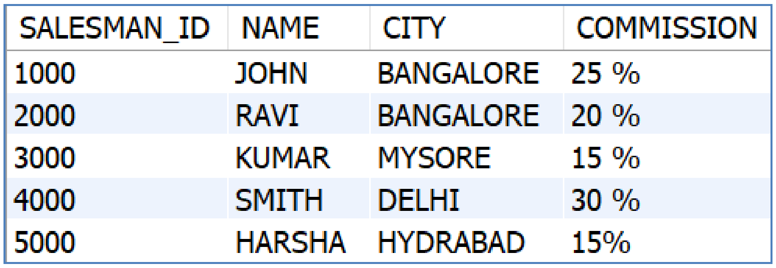
Step 3: Insert Values into Tables

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

INSERT INTO ORDERS VALUES (50, 5000, '2017-05-04', 10, 1000);
INSERT INTO ORDERS VALUES (55, 1000, '2017-05-04', 10, 1000);
INSERT INTO ORDERS VALUES (56, 300, '2017-05-04', 10, 2000);
INSERT INTO ORDERS VALUES (51, 450, '2017-01-20', 10, 2000);
INSERT INTO ORDERS VALUES (52, 1000, '2017-02-24', 13, 2000);
INSERT INTO ORDERS VALUES (53, 3500, '2017-04-13', 14, 3000);
INSERT INTO ORDERS VALUES (54, 550, '2017-03-09', 12, 2000);
INSERT INTO ORDERS VALUES (57, 450, '2017-03-09', 12, 2000);
INSERT INTO ORDERS VALUES (58, 350, '2017-03-09', 12, 2000);
INSERT INTO ORDERS VALUES (60, 150, '2017-03-09', 12, 1000);
```

INSERT INTO ORDERS VALUES (61, 200, '2017-03-09', 12, 3000);

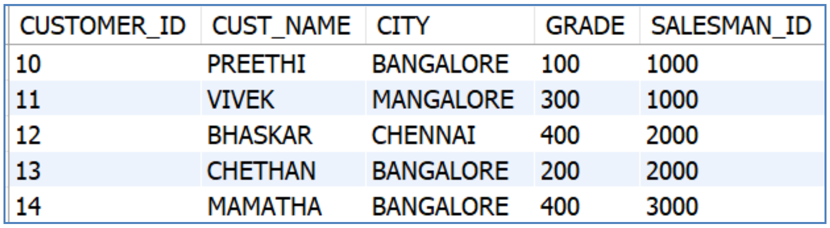
Step 4: Display table contents

select * from SALESMAN;



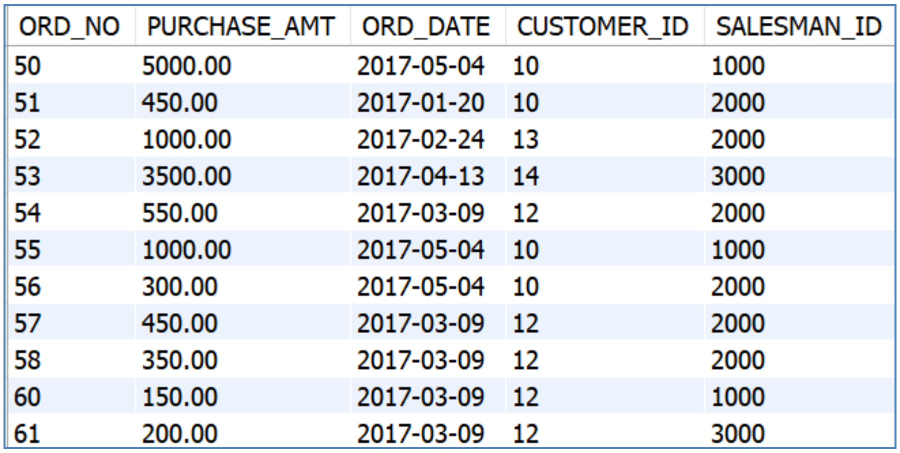
SALESMAN_ID	NAME	CITY	COMMISSION
1000	JOHN	BANGALORE	25 %
2000	RAVI	BANGALORE	20 %
3000	KUMAR	MYSORE	15 %
4000	SMITH	DELHI	30 %
5000	HARSHA	HYDRABAD	15%

select * from 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

select * from ORDERS;



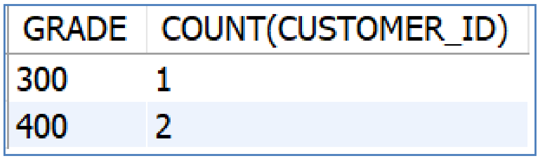
ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
50	5000.00	2017-05-04	10	1000
51	450.00	2017-01-20	10	2000
52	1000.00	2017-02-24	13	2000
53	3500.00	2017-04-13	14	3000
54	550.00	2017-03-09	12	2000
55	1000.00	2017-05-04	10	1000
56	300.00	2017-05-04	10	2000
57	450.00	2017-03-09	12	2000
58	350.00	2017-03-09	12	2000
60	150.00	2017-03-09	12	1000
61	200.00	2017-03-09	12	3000

Step 5: Execute Queries:

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

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

Output:

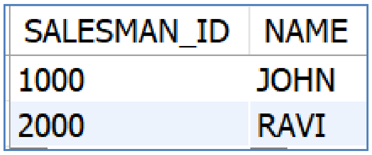


GRADE	COUNT(CUSTOMER_ID)
300	1
400	2

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

```
SELECT SALESMAN_ID, NAME
FROM SALESMAN A
WHERE 1 < (SELECT COUNT(*) FROM CUSTOMER WHERE SALESMAN_ID=A.SALESMAN_ID);
```

Output:

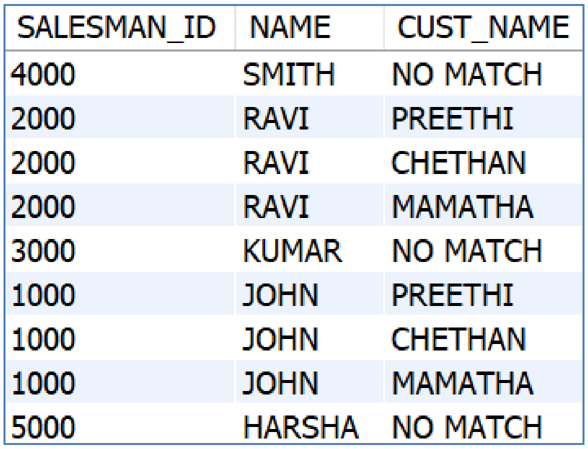


SALESMAN_ID	NAME
1000	JOHN
2000	RAVI

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

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

Output:

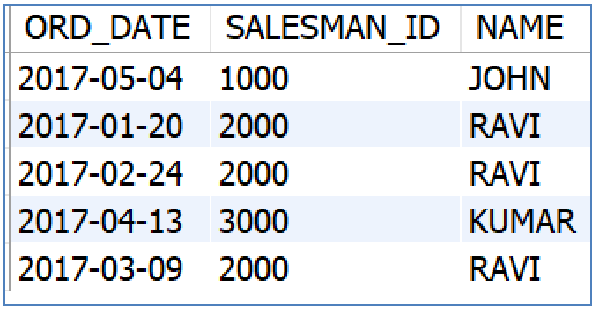


SALESMAN_ID	NAME	CUST_NAME
4000	SMITH	NO MATCH
2000	RAVI	PREETHI
2000	RAVI	CHETHAN
2000	RAVI	MAMATHA
3000	KUMAR	NO MATCH
1000	JOHN	PREETHI
1000	JOHN	CHETHAN
1000	JOHN	MAMATHA
5000	HARSHA	NO MATCH

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

```
CREATE VIEW ELITESALESMAN 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);


select * from ELITESALESMAN;
```

Output:



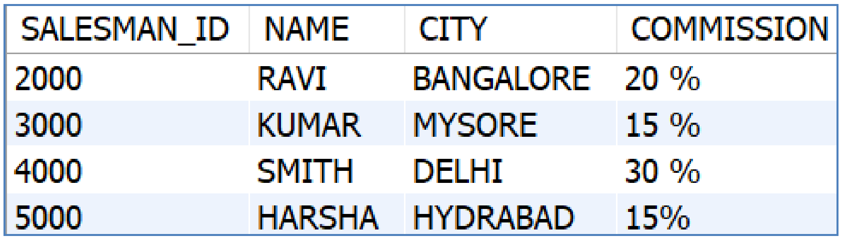
ORD_DATE	SALESMAN_ID	NAME
2017-05-04	1000	JOHN
2017-01-20	2000	RAVI
2017-02-24	2000	RAVI
2017-04-13	3000	KUMAR
2017-03-09	2000	RAVI

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

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

select * from SALESMAN;

Output:



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

****

3. Consider the schema for MovieDatabase:

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

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

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

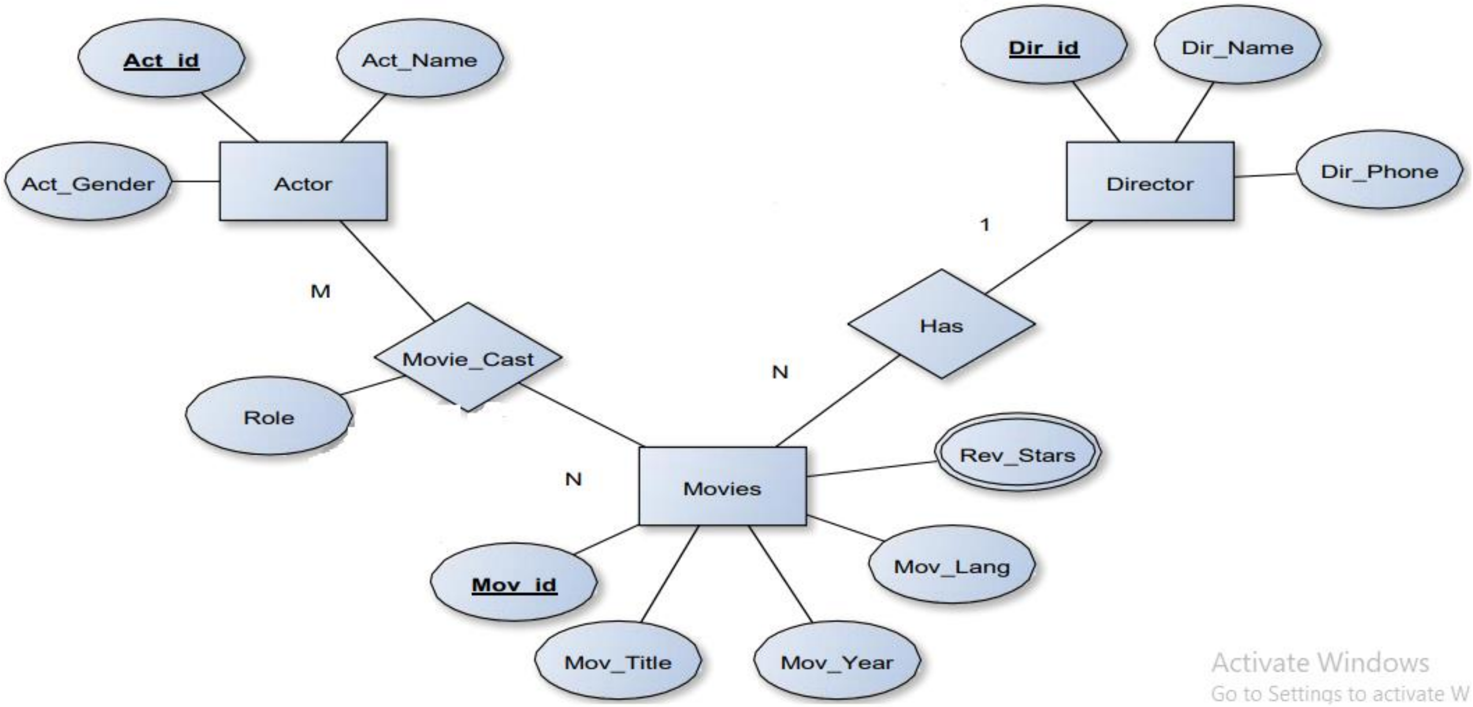
c. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

d. 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.

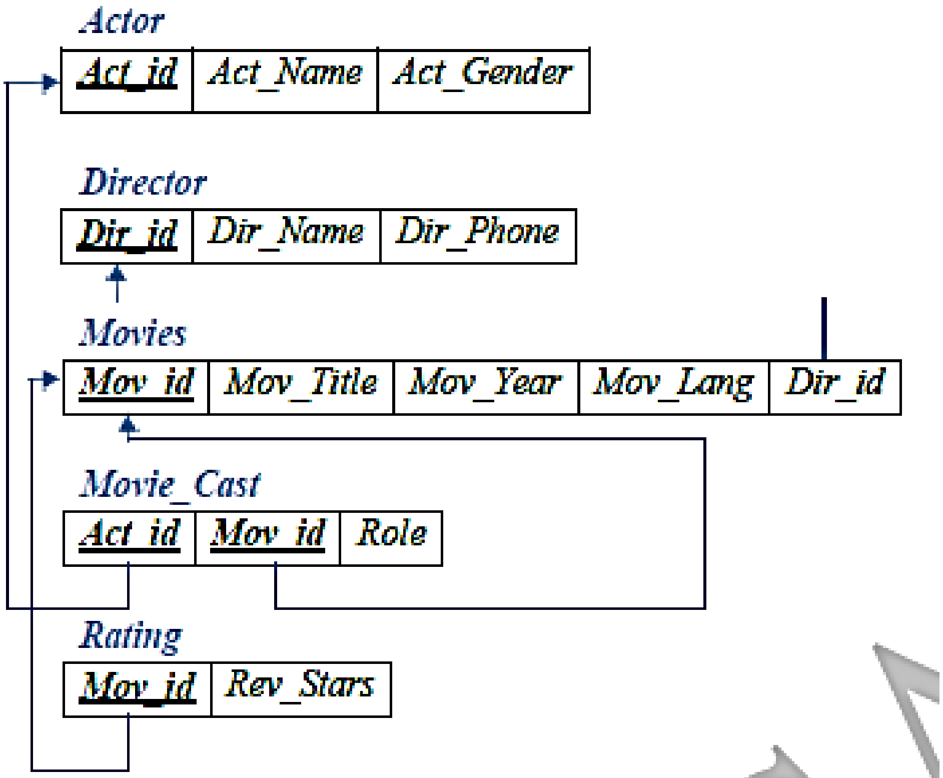
e. Update rating of all movies directed by ‘Steven Spielberg’ to 5.

Solution:

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

Step 1: Create Database

```
CREATE database movies;
use movies;
```

Step 2: Create Tables

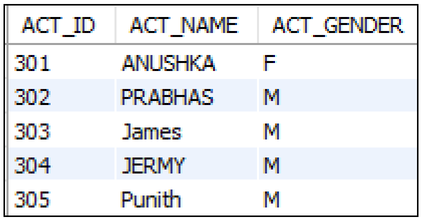
```
CREATE TABLE ACTOR (
ACT_ID int(3),
ACT_NAME VARCHAR(20),
ACT_GENDER CHAR(1),
PRIMARY KEY (ACT_ID));

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

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

CREATE TABLE MOVIE_CAST (
ACT_ID int(3),
MOV_ID int(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 int(4),
REV_STARS VARCHAR (25),
PRIMARY KEY (MOV_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

Step 3: Insert Values into Tables

```
INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');
INSERT INTO ACTOR VALUES (302,'PRABHAS','M');
INSERT INTO ACTOR VALUES (303,'James','M');
INSERT INTO ACTOR VALUES (304,'JERMY','M');
INSERT INTO ACTOR VALUES (305,'Punith','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, 'TELUGU', 60);
INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELUGU', 60);
INSERT INTO MOVIES VALUES (1003,'Vertigo', 1958, 'ENGLISH', 61);
INSERT INTO MOVIES VALUES (1005,'The Birds', 1963, 'ENGLISH', 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);

INSERT INTO RATING VALUES (1005, 3);
```

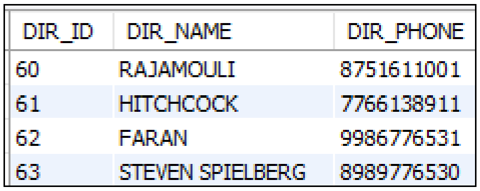
Step 4: Display table contents

```
SELECT * FROM ACTOR;
```



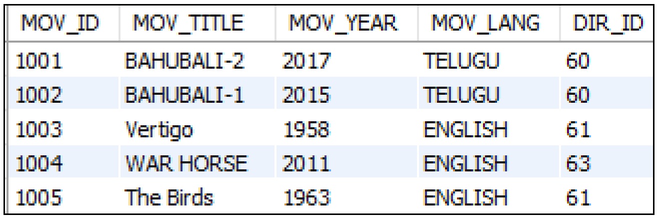
ACT_ID	ACT_NAME	ACT_GENDER
301	ANUSHKA	F
302	PRABHAS	M
303	James	M
304	JERMY	M
305	Punith	M

```
SELECT * FROM DIRECTOR;
```



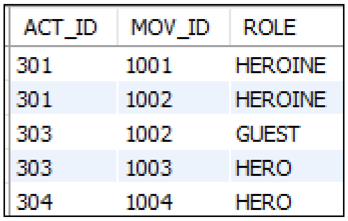
DIR_ID	DIR_NAME	DIR_PHONE
60	RAJAMOULI	8751611001
61	HITCHCOCK	7766138911
62	FARAN	9986776531
63	STEVEN SPIELBERG	8989776530

```
SELECT * FROM MOVIES;
```



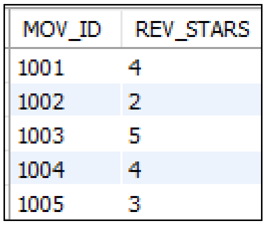
MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
1001	BAHUBALI-2	2017	TELUGU	60
1002	BAHUBALI-1	2015	TELUGU	60
1003	Vertigo	1958	ENGLISH	61
1004	WAR HORSE	2011	ENGLISH	63
1005	The Birds	1963	ENGLISH	61

```
SELECT * FROM MOVIE_CAST;
```



ACT_ID	MOV_ID	ROLE
301	1001	HEROINE
301	1002	HEROINE
303	1002	GUEST
303	1003	HERO
304	1004	HERO

```
SELECT * FROM RATING;
```

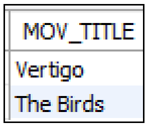


MOV_ID	REV_STARS
1001	4
1002	2
1003	5
1004	4
1005	3

Step 5: Execute Queries:

```
-- 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');
```

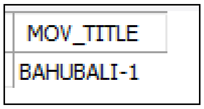
Output:



MOV_TITLE
Vertigo
The Birds

```
/* 2. Find the movie names where one or more actors acted in two or
moremovies.*/
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;
```

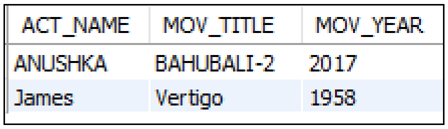
Output:



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). */
                              -- Method 1
SELECT ACT_NAME, MOV_TITLE, MOV_YEAR 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;

                              -- Method 2
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;
```

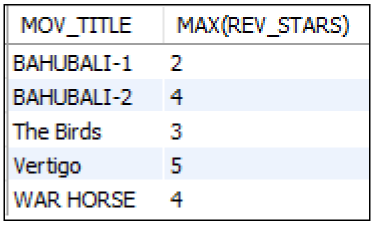
Output:



ACT_NAME	MOV_TITLE	MOV_YEAR
ANUSHKA	BAHUBALI-2	2017
James	Vertigo	1958

```
/* 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;
```

Output:

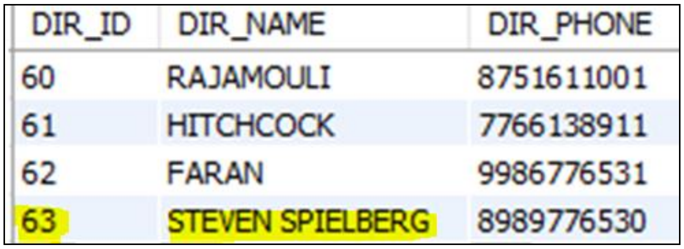


MOV_TITLE	MAX(REV_STARS)
BAHUBALI-1	2
BAHUBALI-2	4
The Birds	3
Vertigo	5
WAR HORSE	4

```
-- 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'));
```

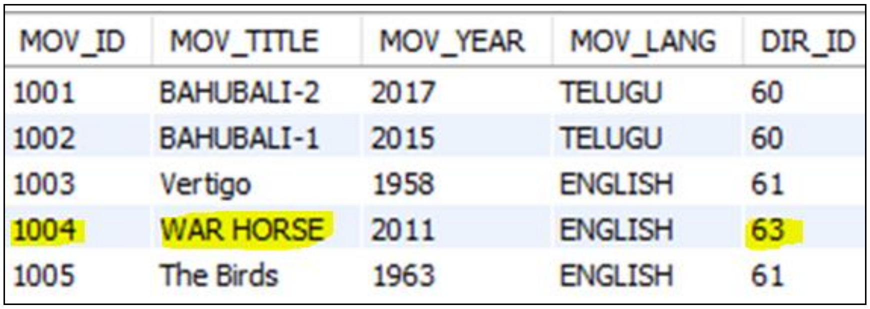
Output:

```
select * from DIRECTOR;
```



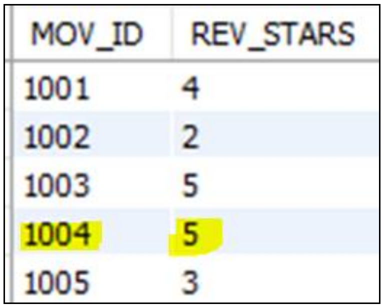
DIR_ID	DIR_NAME	DIR_PHONE
60	RAJAMOULI	8751611001
61	HITCHCOCK	7766138911
62	FARAN	9986776531
63	STEVEN SPIELBERG	8989776530

```
select * from MOVIES;
```



MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
1001	BAHUBALI-2	2017	TELUGU	60
1002	BAHUBALI-1	2015	TELUGU	60
1003	Vertigo	1958	ENGLISH	61
1004	WAR HORSE	2011	ENGLISH	63
1005	The Birds	1963	ENGLISH	61

```
select * from RATING;
```



MOV_ID	REV_STARS
1001	4
1002	2
1003	5
1004	5
1005	3

****

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

a. List all the student details studying in fourth semester ‘C’ section.

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

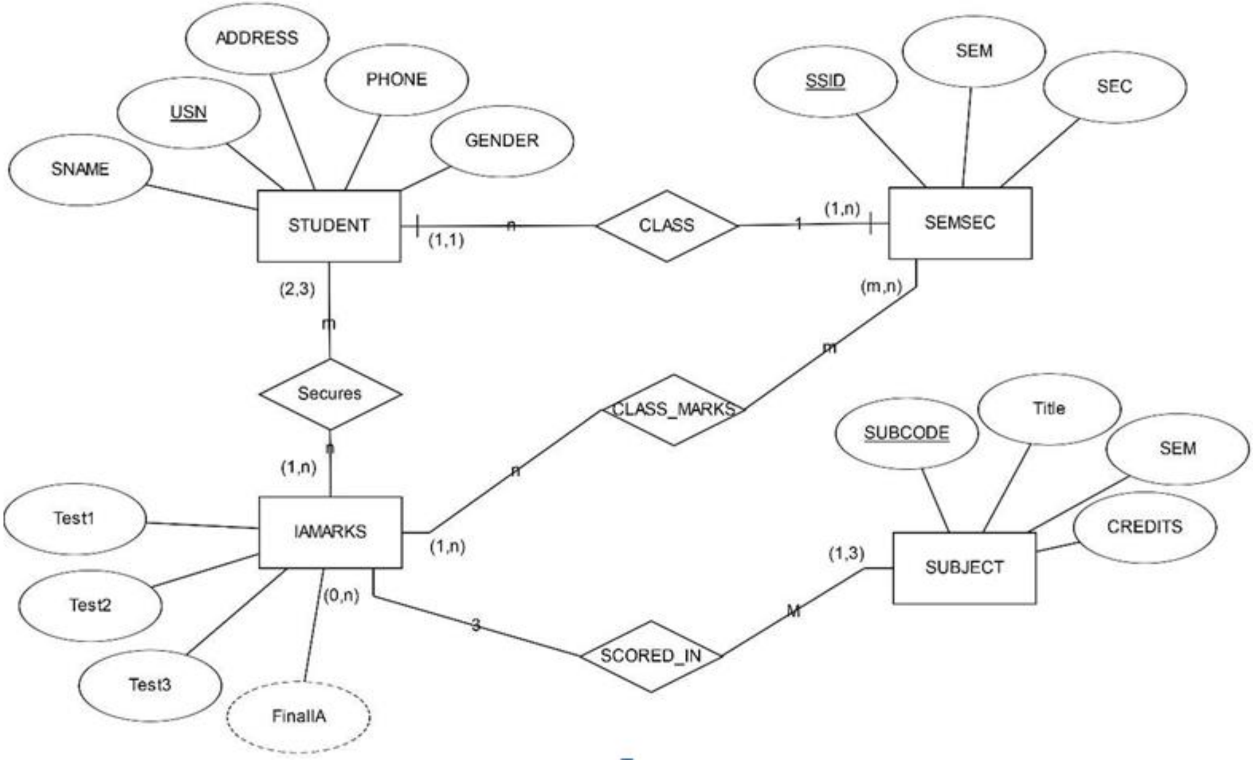
c. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.

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

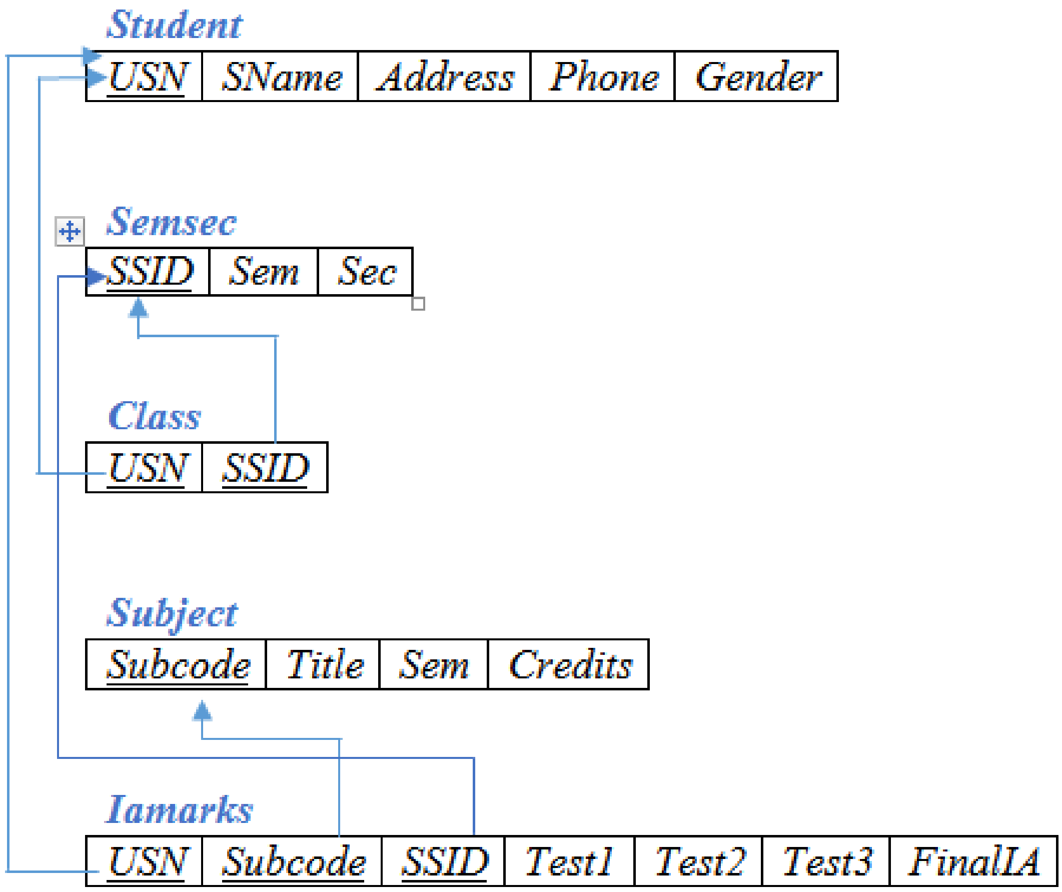
e. Categorize students based on the following criterion: If
FinalIA = 17 to 20 then CAT =‘Outstanding’
If FinalIA = 12 to 16 then CAT = ‘Average’ If
FinalIA< 12 then CAT = ‘Weak’
Give these details only for 8th semester A, B, and C section students.

Solution:

Entity - Relationship Diagram



Schema Diagram



Step 1: Create Database

```
create database collegedb;
use collegedb;
```

Step 2: Create Tables

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

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

DESC SEMSEC;

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

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

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

DESC IAMARKS;
```

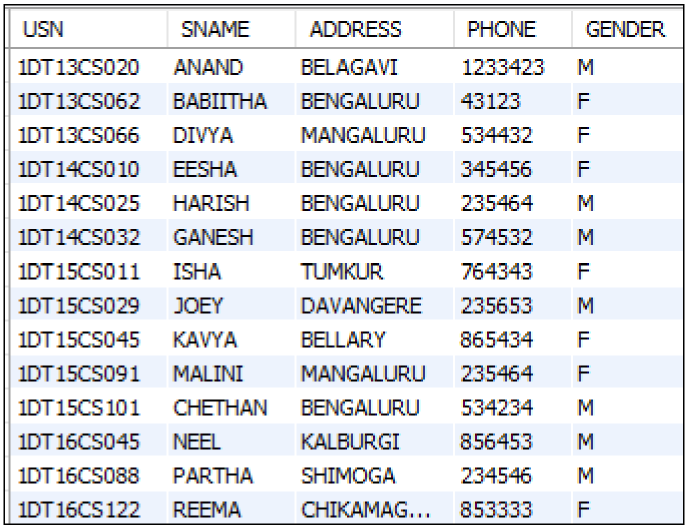
Step 3: Insert Values into Tables

```
INSERT INTO STUDENT VALUES ('1DT13CS020','ANAND','BELAGAVI',
1233423,'M');
INSERT INTO STUDENT VALUES
('1DT13CS062','BABIITHA','BENGALURU',43123,'F');
INSERT INTO STUDENT VALUES ('1DT15CS101','CHETHAN','BENGALURU',
534234,'M');
INSERT INTO STUDENT VALUES
('1DT13CS066','DIVYA','MANGALURU',534432,'F');
INSERT INTO STUDENT VALUES ('1DT14CS010','EESHA','BENGALURU',
345456,'F');
INSERT INTO STUDENT VALUES
('1DT14CS032','GANESH','BENGALURU',574532,'M');
INSERT INTO STUDENT VALUES ('1DT14CS025','HARISH','BENGALURU',
235464,'M');
INSERT INTO STUDENT VALUES ('1DT15CS011','ISHA','TUMKUR',
764343,'F');
INSERT INTO STUDENT VALUES ('1DT15CS029','JOEY','DAVANGERE',
235653,'M');
INSERT INTO STUDENT VALUES ('1DT15CS045','KAVYA','BELLARY',
865434,'F');
INSERT INTO STUDENT VALUES
('1DT15CS091','MALINI','MANGALURU',235464,'F');
INSERT INTO STUDENT VALUES ('1DT16CS045','NEEL','KALBURGI',
856453,'M');
```

```
INSERT INTO STUDENT VALUES ('1DT16CS088','PARTHA','SHIMOGA',
234546,'M');
INSERT INTO STUDENT VALUES ('1DT16CS122','REEMA','CHIKAMAGALUR',
853333,'F');
------------------------------------
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 ('1DT13CS020','CSE8A');
INSERT INTO CLASS VALUES ('1DT13CS062','CSE8A');
INSERT INTO CLASS VALUES ('1DT13CS066','CSE8B');
INSERT INTO CLASS VALUES ('1DT15CS101','CSE8C');
INSERT INTO CLASS VALUES ('1DT14CS010','CSE7A');
INSERT INTO CLASS VALUES ('1DT14CS025','CSE7A');
INSERT INTO CLASS VALUES ('1DT14CS032','CSE7A');
INSERT INTO CLASS VALUES ('1DT15CS011','CSE4A');
INSERT INTO CLASS VALUES ('1DT15CS029','CSE4A');
INSERT INTO CLASS VALUES ('1DT15CS045','CSE4B');
INSERT INTO CLASS VALUES ('1DT15CS091','CSE4C');
INSERT INTO CLASS VALUES ('1DT16CS045','CSE3A');
INSERT INTO CLASS VALUES ('1DT16CS088','CSE3B');
INSERT INTO CLASS VALUES ('1DT16CS122','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
('1DT15CS101','10CS81','CSE8C', 15, 16, 18);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1DT15CS101','10CS82','CSE8C', 12, 19, 14);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1DT15CS101','10CS83','CSE8C', 19, 15, 20);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1DT15CS101','10CS84','CSE8C', 20, 16, 19);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1DT15CS101','10CS85','CSE8C', 15, 15, 12);
```

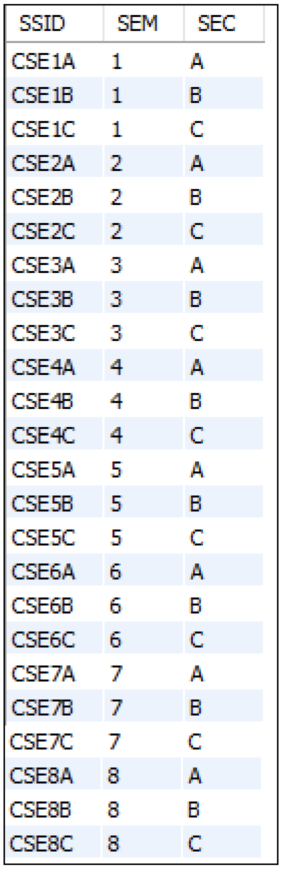
Step 4: Display table contents

```
SELECT * FROM STUDENT;
```



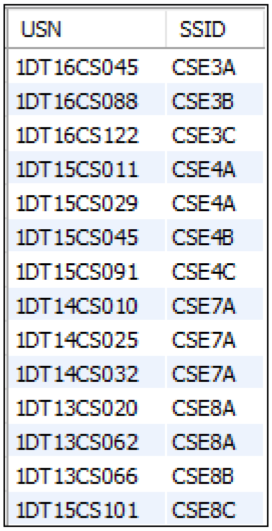
USN	SNAME	ADDRESS	PHONE	GENDER
1DT13CS020	ANAND	BELAGAVI	1233423	M
1DT13CS062	BABIITHA	BENGALURU	43123	F
1DT13CS066	DIVYA	MANGALURU	534432	F
1DT14CS010	EESHA	BENGALURU	345456	F
1DT14CS025	HARISH	BENGALURU	235464	M
1DT14CS032	GANESH	BENGALURU	574532	M
1DT15CS011	ISHA	TUMKUR	764343	F
1DT15CS029	JOEY	DAVANGERE	235653	M
1DT15CS045	KAVYA	BELLARY	865434	F
1DT15CS091	MALINI	MANGALURU	235464	F
1DT15CS101	CHETHAN	BENGALURU	534234	M
1DT16CS045	NEEL	KALBURGI	856453	M
1DT16CS088	PARTHA	SHIMOGA	234546	M
1DT16CS122	REEMA	CHIKAMAG...	853333	F

```
SELECT * FROM SEMSEC;
```



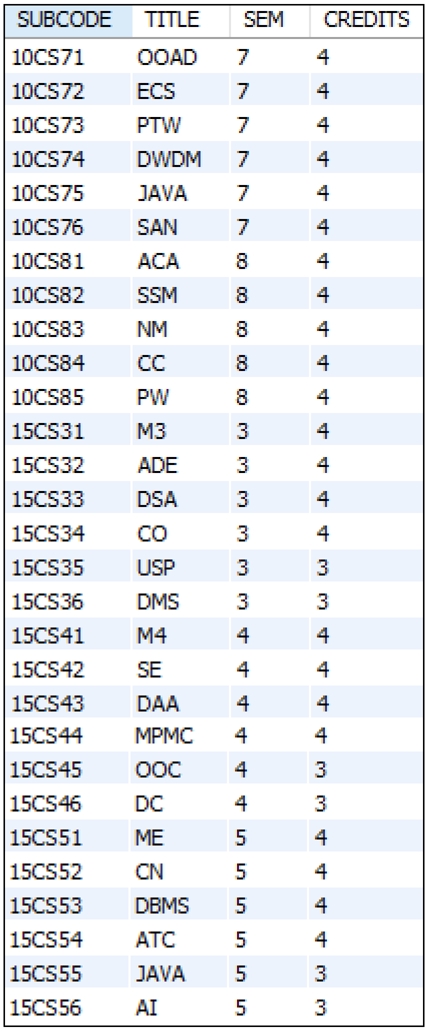
SSID	SEM	SEC
CSE1A	1	A
CSE1B	1	B
CSE1C	1	C
CSE2A	2	A
CSE2B	2	B
CSE2C	2	C
CSE3A	3	A
CSE3B	3	B
CSE3C	3	C
CSE4A	4	A
CSE4B	4	B
CSE4C	4	C
CSE5A	5	A
CSE5B	5	B
CSE5C	5	C
CSE6A	6	A
CSE6B	6	B
CSE6C	6	C
CSE7A	7	A
CSE7B	7	B
CSE7C	7	C
CSE8A	8	A
CSE8B	8	B
CSE8C	8	C

```
SELECT * FROM CLASS;
```



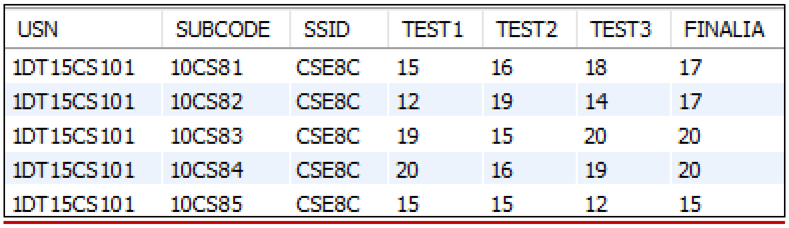
USN	SSID
1DT16CS045	CSE3A
1DT16CS088	CSE3B
1DT16CS122	CSE3C
1DT15CS011	CSE4A
1DT15CS029	CSE4A
1DT15CS045	CSE4B
1DT15CS091	CSE4C
1DT14CS010	CSE7A
1DT14CS025	CSE7A
1DT14CS032	CSE7A
1DT13CS020	CSE8A
1DT13CS062	CSE8A
1DT13CS066	CSE8B
1DT15CS101	CSE8C

```
SELECT * FROM SUBJECT;
```



SUBCODE	TITLE	SEM	CREDITS
10CS71	OOAD	7	4
10CS72	ECS	7	4
10CS73	PTW	7	4
10CS74	DWDM	7	4
10CS75	JAVA	7	4
10CS76	SAN	7	4
10CS81	ACA	8	4
10CS82	SSM	8	4
10CS83	NM	8	4
10CS84	CC	8	4
10CS85	PW	8	4
15CS31	M3	3	4
15CS32	ADE	3	4
15CS33	DSA	3	4
15CS34	CO	3	4
15CS35	USP	3	3
15CS36	DMS	3	3
15CS41	M4	4	4
15CS42	SE	4	4
15CS43	DAA	4	4
15CS44	MPMC	4	4
15CS45	OOC	4	3
15CS46	DC	4	3
15CS51	ME	5	4
15CS52	CN	5	4
15CS53	DBMS	5	4
15CS54	ATC	5	4
15CS55	JAVA	5	3
15CS56	AI	5	3

```
SELECT * FROM IAMARKS;
```

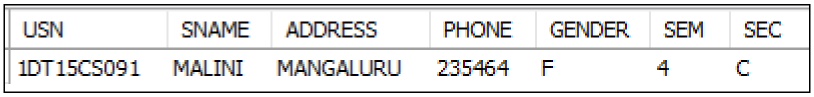


USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1DT15CS101	10CS81	CSE8C	15	16	18	17
1DT15CS101	10CS82	CSE8C	12	19	14	17
1DT15CS101	10CS83	CSE8C	19	15	20	20
1DT15CS101	10CS84	CSE8C	20	16	19	20
1DT15CS101	10CS85	CSE8C	15	15	12	15

Step 5: Execute 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';
```

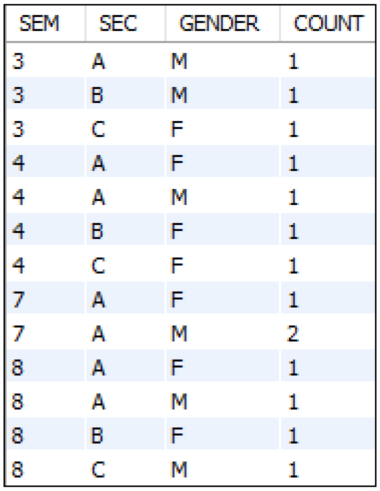
Output:



USN	SNAME	ADDRESS	PHONE	GENDER	SEM	SEC
1DT15CS091	MALINI	MANGALURU	235464	F	4	C

```
/* 2. Compute the total number of male and female students in each
semester and in each section.*/
SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT
FROM STUDENT S, SEMSEC SS, CLASS C
WHERE S.USN = C.USN AND SS.SSID = C.SSID
GROUP BY SS.SEM, SS.SEC, S.GENDER ORDER BY SEM;
```

Output:

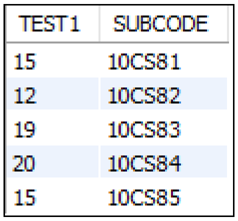


SEM	SEC	GENDER	COUNT
3	A	M	1
3	B	M	1
3	C	F	1
4	A	F	1
4	A	M	1
4	B	F	1
4	C	F	1
7	A	F	1
7	A	M	2
8	A	F	1
8	A	M	1
8	B	F	1
8	C	M	1

```
/* 3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in
all subjects.*/
CREATE VIEW STU_TEST1_MARKS_VIEW AS
```

```
SELECT TEST1, SUBCODE FROM IAMARKS WHERE USN = '1DT13CS091';


select * from STU_TEST1_MARKS_VIEW;
```

Output:



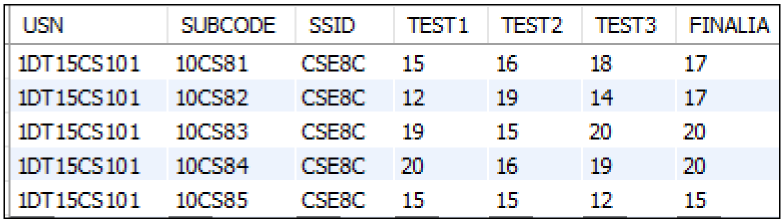
TEST1	SUBCODE
15	10CS81
12	10CS82
19	10CS83
20	10CS84
15	10CS85

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

```
DELIMITER //
CREATE PROCEDURE AVG_MARKS()
BEGIN
DECLARE C_A INTEGER;
DECLARE C_B INTEGER;
DECLARE C_C INTEGER;
DECLARE C_SUM INTEGER;
DECLARE C_AVG INTEGER;
DECLARE C_USN VARCHAR(10);
DECLARE C_SUBCODE VARCHAR(8);
DECLARE C_SSID VARCHAR(5);

DECLARE C_IAMARKS CURSOR FOR
SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B,
GREATEST(TEST3,TEST2) AS C, USN, SUBCODE, SSID
FROM IAMARKS
WHERE FINALIA IS NULL
FOR UPDATE;

OPEN C_IAMARKS;
LOOP

FETCH C_IAMARKS INTO C_A, C_B, C_C, C_USN, C_SUBCODE, C_SSID;

IF (C_A != C_B) THEN
     SET C_SUM=C_A+C_B;
ELSE
```

```
     SET C_SUM=C_A+C_C;
END IF;

SET C_AVG=C_SUM/2;

UPDATE IAMARKS SET FINALIA = C_AVG
WHERE USN = C_USN AND SUBCODE = C_SUBCODE AND SSID = C_SSID;

END LOOP;
CLOSE C_IAMARKS;
END;
//

CALL AVG_MARKS();

SELECT * FROM IAMARKS;
```

Output:

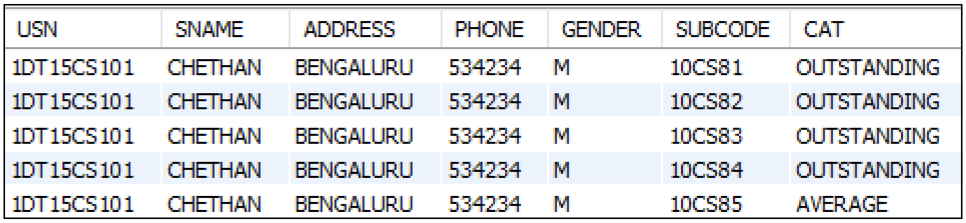


USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1DT15CS101	10CS81	CSE8C	15	16	18	17
1DT15CS101	10CS82	CSE8C	12	19	14	17
1DT15CS101	10CS83	CSE8C	19	15	20	20
1DT15CS101	10CS84	CSE8C	20	16	19	20
1DT15CS101	10CS85	CSE8C	15	15	12	15

```
/* 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. */
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER, IA.SUBCODE,
(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;
```

Output:



USN	SNAME	ADDRESS	PHONE	GENDER	SUBCODE	CAT
1DT15CS101	CHETHAN	BENGALURU	534234	M	10CS81	OUTSTANDING
1DT15CS101	CHETHAN	BENGALURU	534234	M	10CS82	OUTSTANDING
1DT15CS101	CHETHAN	BENGALURU	534234	M	10CS83	OUTSTANDING
1DT15CS101	CHETHAN	BENGALURU	534234	M	10CS84	OUTSTANDING
1DT15CS101	CHETHAN	BENGALURU	534234	M	10CS85	AVERAGE

****

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

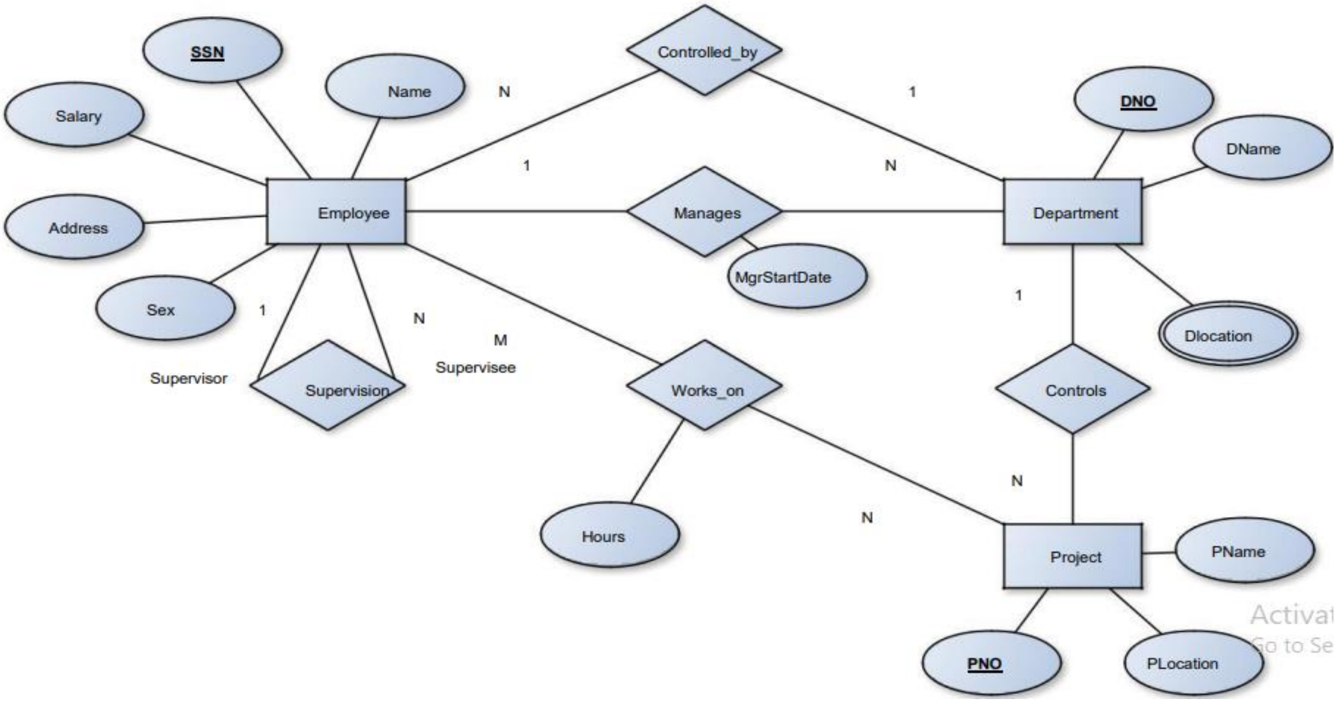
a. Make a list of all project numbers for projects that involve an employee whose last name is ‘Scott’, either as a worker or as a manager of the department that controls the project.

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

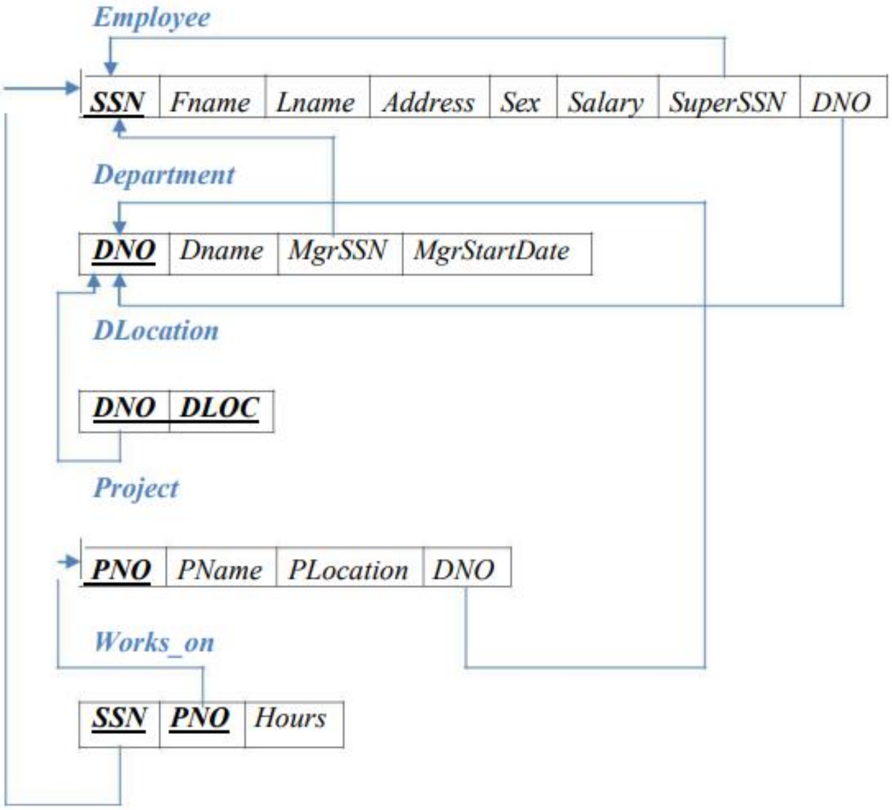
c. Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the maximum salary, the minimum salary, and the average salary in this department

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

Entity-Relationship Diagram



Schema Diagram



Step 1: Create Database

```
CREATE DATABASE COMPANY;
USE COMPANY;
```

Step 2: Create Tables

```
CREATE TABLE DEPARTMENT
(DNO VARCHAR(20) PRIMARY KEY,
DNAME VARCHAR(20),
MGR_SSN VARCHAR(20),
MGR_START_DATE DATE);

DESC DEPARTMENT;
-- --------------------------------
CREATE TABLE EMPLOYEE
(SSN VARCHAR(20) PRIMARY KEY,
NAME VARCHAR(20),
ADDRESS VARCHAR(20),
SEX CHAR(1),
SALARY INTEGER,
SUPERSSN VARCHAR(20),
DNO VARCHAR(20),
FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE (SSN),
FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO));

DESC EMPLOYEE;
-- --------------------------------
-- ADD FOREIGN KEY Constraint to DEPARTMENT table

ALTER TABLE DEPARTMENT
ADD FOREIGN KEY (MGR_SSN) REFERENCES EMPLOYEE(SSN);
-- --------------------------------
```

```
CREATE TABLE DLOCATION
(DLOC VARCHAR(20),
DNO VARCHAR(20),
FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO),
PRIMARY KEY (DNO, DLOC));

DESC DLOCATION;
-- --------------------------------
CREATE TABLE PROJECT
(PNO INTEGER PRIMARY KEY,
PNAME VARCHAR(20),
PLOCATION VARCHAR(20),
DNO VARCHAR(20),
FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO));

DESC PROJECT;
-- --------------------------------
CREATE TABLE WORKS_ON
(HOURS INTEGER,
SSN VARCHAR(20),
PNO INTEGER,
FOREIGN KEY (SSN) REFERENCES EMPLOYEE(SSN),
FOREIGN KEY (PNO) REFERENCES PROJECT(PNO),
PRIMARY KEY (SSN, PNO));

DESC WORKS_ON;
-- --------------------------------
```

Step 3: Insert Values into Tables

```
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC01','BEN SCOTT','BANGALORE','M', 450000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC02','HARRY SMITH','BANGALORE','M', 500000);
```

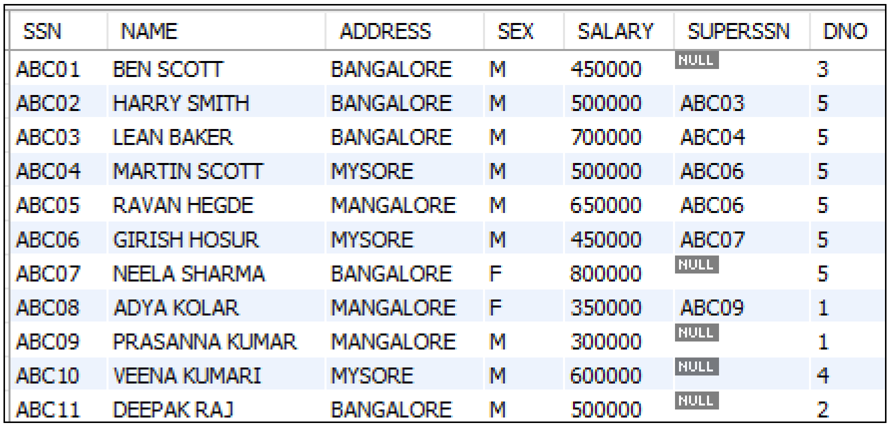
```
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC03','LEAN BAKER','BANGALORE','M', 700000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC04','MARTIN SCOTT','MYSORE','M', 500000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC05','RAVAN HEGDE','MANGALORE','M', 650000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC06','GIRISH HOSUR','MYSORE','M', 450000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC07','NEELA SHARMA','BANGALORE','F', 800000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC08','ADYA KOLAR','MANGALORE','F', 350000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC09','PRASANNA KUMAR','MANGALORE','M', 300000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC10','VEENA KUMARI','MYSORE','M', 600000);
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
 ('ABC11','DEEPAK RAJ','BANGALORE','M', 500000);

 INSERT INTO DEPARTMENT VALUES ('1','ACCOUNTS','ABC09', '2016-01-03');
INSERT INTO DEPARTMENT VALUES ('2','IT','ABC11', '2017-02-04');
INSERT INTO DEPARTMENT VALUES ('3','HR','ABC01', '2016-04-05');
INSERT INTO DEPARTMENT VALUES ('4','HELPDESK', 'ABC10', '2017-06-03');
INSERT INTO DEPARTMENT VALUES ('5','SALES','ABC06', '2017-01-08');
----------------------------------
-- Updating EMPLOYEE records

UPDATE EMPLOYEE SET
SUPERSSN=NULL, DNO='3'
WHERE SSN='ABC01';

UPDATE EMPLOYEE SET
```

```
SUPERSSN='ABC03', DNO='5'
WHERE SSN='ABC02';

UPDATE EMPLOYEE SET
SUPERSSN='ABC04', DNO='5'
WHERE SSN='ABC03';

UPDATE EMPLOYEE SET
SUPERSSN='ABC06', DNO='5'
WHERE SSN='ABC04';

UPDATE EMPLOYEE SET
DNO='5', SUPERSSN='ABC06'
WHERE SSN='ABC05';

UPDATE EMPLOYEE SET
DNO='5', SUPERSSN='ABC07'
WHERE SSN='ABC06';

UPDATE EMPLOYEE SET
DNO='5', SUPERSSN=NULL
WHERE SSN='ABC07';

UPDATE EMPLOYEE SET
DNO='1', SUPERSSN='ABC09'
WHERE SSN='ABC08';

UPDATE EMPLOYEE SET
DNO='1', SUPERSSN=NULL
WHERE SSN='ABC09';

UPDATE EMPLOYEE SET
```

```
DNO='4', SUPERSSN=NULL
WHERE SSN='ABC10';

UPDATE EMPLOYEE SET
DNO='2', SUPERSSN=NULL
WHERE SSN='ABC11';

SELECT * FROM EMPLOYEE;
-------------------------------
-- Inserting records into DLOCATION table

INSERT INTO DLOCATION VALUES ('BENGALURU', '1');
INSERT INTO DLOCATION VALUES ('BENGALURU', '2');
INSERT INTO DLOCATION VALUES ('BENGALURU', '3');
INSERT INTO DLOCATION VALUES ('MYSORE', '4');
INSERT INTO DLOCATION VALUES ('MYSORE', '5');

SELECT * FROM DLOCATION;
--------------------------------
-- Inserting records into PROJECT table
INSERT INTO PROJECT VALUES (1000,'IOT','BENGALURU','5');
INSERT INTO PROJECT VALUES (1001,'CLOUD','BENGALURU','5');
INSERT INTO PROJECT VALUES (1002,'BIGDATA','BENGALURU','5');
INSERT INTO PROJECT VALUES (1003,'SENSORS','BENGALURU','3');
INSERT INTO PROJECT VALUES (1004,'BANK MANAGEMENT','BENGALURU','1');
INSERT INTO PROJECT VALUES (1005,'SALARY MANAGEMENT','BANGALORE','1');
INSERT INTO PROJECT VALUES (1006,'OPENSTACK','BENGALURU','4');
INSERT INTO PROJECT VALUES (1007,'SMART CITY','BENGALURU','2');
------------------------------
INSERT INTO WORKS_ON VALUES (4, 'ABC02', 1000);
INSERT INTO WORKS_ON VALUES (6, 'ABC02', 1001);
INSERT INTO WORKS_ON VALUES (8, 'ABC02', 1002);
INSERT INTO WORKS_ON VALUES (10,'ABC03', 1000);
INSERT INTO WORKS_ON VALUES (3, 'ABC05', 1000);
INSERT INTO WORKS_ON VALUES (4, 'ABC06', 1001);
INSERT INTO WORKS_ON VALUES (5, 'ABC07', 1002);
INSERT INTO WORKS_ON VALUES (6, 'ABC04', 1002);
INSERT INTO WORKS_ON VALUES (7, 'ABC01', 1003);
INSERT INTO WORKS_ON VALUES (5, 'ABC08', 1004);
INSERT INTO WORKS_ON VALUES (6, 'ABC09', 1005);
INSERT INTO WORKS_ON VALUES (4, 'ABC10', 1006);
INSERT INTO WORKS_ON VALUES (10,'ABC11', 1007);
```

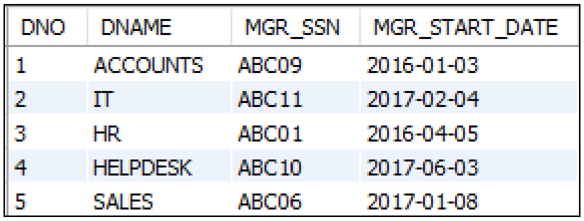
Step 4: Display table contents

```
SELECT * FROM EMPLOYEE;
```



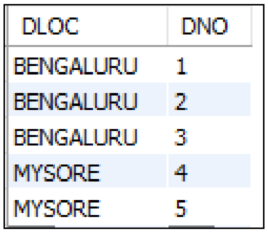
SSN	NAME	ADDRESS	SEX	SALARY	SUPERSSN	DNO
ABC01	BEN SCOTT	BANGALORE	M	450000	NULL	3
ABC02	HARRY SMITH	BANGALORE	M	500000	ABC03	5
ABC03	LEAN BAKER	BANGALORE	M	700000	ABC04	5
ABC04	MARTIN SCOTT	MYSORE	M	500000	ABC06	5
ABC05	RAVAN HEGDE	MANGALORE	M	650000	ABC06	5
ABC06	GIRISH HOSUR	MYSORE	M	450000	ABC07	5
ABC07	NEELA SHARMA	BANGALORE	F	800000	NULL	5
ABC08	ADYA KOLAR	MANGALORE	F	350000	ABC09	1
ABC09	PRASANNA KUMAR	MANGALORE	M	300000	NULL	1
ABC10	VEENA KUMARI	MYSORE	M	600000	NULL	4
ABC11	DEEPAK RAJ	BANGALORE	M	500000	NULL	2

```
SELECT * FROM DEPARTMENT;
```



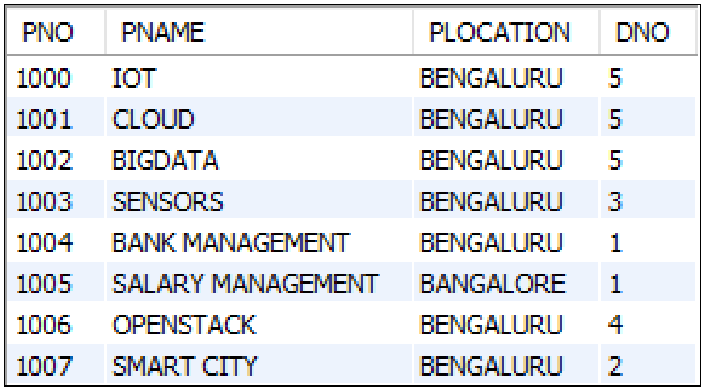
DNO	DNAME	MGR_SSN	MGR_START_DATE
1	ACCOUNTS	ABC09	2016-01-03
2	IT	ABC11	2017-02-04
3	HR	ABC01	2016-04-05
4	HELPDESK	ABC10	2017-06-03
5	SALES	ABC06	2017-01-08

SELECT * FROM DLOCATION;



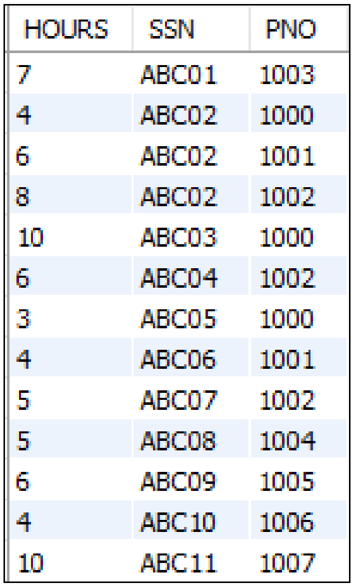
DLOC	DNO
BENGALURU	1
BENGALURU	2
BENGALURU	3
MYSORE	4
MYSORE	5

SELECT * FROM PROJECT;



PNO	PNAME	PLOCATION	DNO
1000	IOT	BENGALURU	5
1001	CLOUD	BENGALURU	5
1002	BIGDATA	BENGALURU	5
1003	SENSORS	BENGALURU	3
1004	BANK MANAGEMENT	BENGALURU	1
1005	SALARY MANAGEMENT	BANGALORE	1
1006	OPENSTACK	BENGALURU	4
1007	SMART CITY	BENGALURU	2

SELECT * FROM WORKS_ON;



HOURS	SSN	PNO
7	ABC01	1003
4	ABC02	1000
6	ABC02	1001
8	ABC02	1002
10	ABC03	1000
6	ABC04	1002
3	ABC05	1000
4	ABC06	1001
5	ABC07	1002
5	ABC08	1004
6	ABC09	1005
4	ABC10	1006
10	ABC11	1007

Step 5: Execute 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

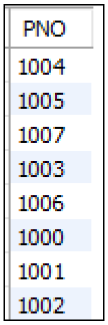
WHERE E.DNO=D.DNO AND D.MGR_SSN=E.SSN AND E.NAME LIKE '%SCOTT'

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.NAME LIKE '%SCOTT';

Output:



PNO
1004
1005
1007
1003
1006
1000
1001
1002

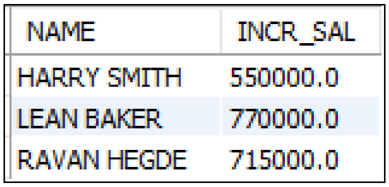
/* 2. Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent raise. */

SELECT E.NAME, 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';

Output:

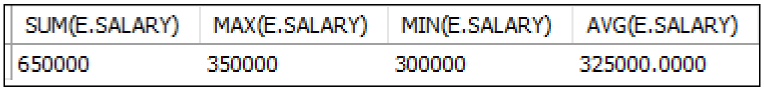


NAME	INCR_SAL
HARRY SMITH	550000.0
LEAN BAKER	770000.0
RAVAN HEGDE	715000.0

/* 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';
```

Output:

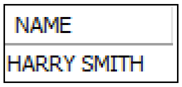


SUM(E.SALARY)	MAX(E.SALARY)	MIN(E.SALARY)	AVG(E.SALARY)
650000	350000	300000	325000.0000

/* 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' AND PNO NOT IN
(SELECT PNO FROM WORKS_ON WHERE E.SSN=SSN));
```

Output:

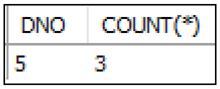


NAME
HARRY SMITH

/* 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. */

```
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;
```

Output:



DNO	COUNT(*)
5	3

****

Viva Questions

1. What is SQL?

Structured Query Language

2. What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

3. What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

4. What is a Database system?

The database and DBMS software together is called as Database system.

5. Advantages of DBMS?

- Redundancy is controlled.
- Unauthorized access is restricted.
- Providing multiple user interfaces.
- Enforcing integrity constraints.
- Providing backup and recovery.

6. Disadvantage in File Processing System?

- Data redundancy & inconsistency.
- Difficult in accessing data.
- Data isolation.
- Data integrity.
- Concurrent access is not possible.
- Security Problems.

7. Describe the three levels of data abstraction?

There are three levels of abstraction:

- Physical level: The lowest level of abstraction describes how data are stored.
- Logical level*:* The next higher level of abstraction, describes what data are stored in database and what relationship among those data.

- View level: The highest level of abstraction describes only part of entire database.

8. Define the "integrity rules"

There are two Integrity rules.

- Entity Integrity: States that "Primary key cannot have NULL value"
- Referential Integrity: States that "Foreign Key can be either a NULL value or should be Primary Key value of other relation.

9. What is extension and intension?

Extension - It is the number of tuples present in a table at any instance. This is time dependent.

Intension -It is a constant value that gives the name, structure of table and the constraints laid on it.

10. What is Data Independence?

Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

- Physical Data Independence: Modification in physical level should not affect the logical level.
- Logical Data Independence: Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

11. What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

12. What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

13. What is E-R model?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

14. What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

15. What is an Entity?

It is an 'object' in the real world with an independent existence.

16. What is an Entity type?

It is a collection (set) of entities that have same attributes.

17. What is an Entity set?

It is a collection of all entities of particular entity type in the database.

18. What is an Extension of entity type?

The collections of entities of a particular entity type are grouped together into an entity set.

19. What is an attribute?

It is a particular property, which describes the entity.

20. What is a Relation Schema and a Relation?

A relation Schema denoted by R(A1, A2, …, An) is made up of the relation name R and the list of attributes A_i that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1,t2,t3, ...,tn). Each tuple is an ordered list of n-values t=(v1,v2, ...,vn).

21. What is degree of a Relation?

It is the number of attribute of its relation schema.

22. What is Relationship?

It is an association among two or more entities.

23. What is Relationship set?

The collection (or set) of similar relationships.

24. What is Relationship type?

Relationship type defines a set of associations or a relationship set among a given set of entity types.

25. What is degree of Relationship type?

It is the number of entity type participating.

26. What is DDL (Data Definition Language)?

A data base schema is specified by a set of definitions expressed by a special language called DDL.

27. What is VDL (View Definition Language)?

It specifies user views and their mappings to the conceptual schema.

28. What is SDL (Storage Definition Language)?

This language is to specify the internal schema. This language may specify the mapping between two schemas.

29. What is Data Storage – Definition Language?

The storage structures and access methods used by database systemare specified by a set of definition in a special type of DDL called data storage- definition language.

30. What is DML (Data Manipulation Language)?

This language that enable user to access or manipulate data as organized by appropriate data model.

- Procedural DML or Low level: DML requires a user to specify what data are needed and how to get those data.
- Non-Procedural DML or High level: DML requires a user to specify what data are needed without specifying how to get those data.

31. What is DML Compiler?

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

32. What is Relational Algebra?

It is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

33. What is Relational Calculus?

It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL, ALPHA,QUEL.

34. What is normalization?

It is a process of analyzing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

- Minimizing redundancy
- Minimizing insertion, deletion and update anomalies.

35. What is Functional Dependency?

A Functional dependency is denoted by X Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y]. This means the value of X component of a tuple uniquely determines the value of component Y.

36. When is a functional dependency F said to be minimal?

- Every dependency in F has a single attribute for its right hand side.
- We cannot replace any dependency X A in F with a dependency Y A where Y is a proper subset of X and still have a set of dependency that is equivalent to F.
- We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

37. What is Multivalued dependency?

Multivalued dependency denoted by X Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties

- t3[x] = t4[X] = t1[X] = t2[X]
- t3[Y] = t1[Y] and t4[Y] = t2[Y]
- t3[Z] = t2[Z] and t4[Z] = t1[Z]

where [Z = (R-(X U Y))]

38. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

39. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

40. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency X Y is fully functional dependency if removal of any attribute A from X means that the dependency does not hold anymore.

41. What is2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

42. What is3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X A either of the following is true

- X is a Super-key of R.
- A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

43. What is BCNF (Boyce-Codd Normal Form)?

A relation schema R is in BCNF if it is in 3NF and satisfies additional constraints that for every FD X A, X must be a candidate key.

44. What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency X Y that holds over R, one of following is true

- X is subset or equal to (or) XY =R.
- X is a super key.

45. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2, ...,Rn} that holds R, one the following is true

- Ri = R for some i.
- The join dependency is implied by the set of FD, over R in which the left side is key of R.

46. What is Domain-Key Normal Form?

A relation is said to be in DKNF if all constraints and dependencies that should hold on the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.