DATABASE MANAGEMENT SYSTEM LABORATORY WITH MINI PROJECT - 21CSL55

1. Program 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.

```
CREATE TABLE PUBLISHER (
   Name VARCHAR(255) PRIMARY KEY,
   Address VARCHAR(255),
   Phone int
);

CREATE TABLE BOOK (
   Book_id INT PRIMARY KEY,
   Title VARCHAR(255),
   Publisher_Name VARCHAR(255),
   Pub_Year INT,
   FOREIGN KEY (Publisher_Name) REFERENCES PUBLISHER(Name) ON DELETE
CASCADE
);

CREATE TABLE BOOK_AUTHORS (
   Book_id INT,
   Author_Name VARCHAR(255),
   PRIMARY KEY (Book_id, Author_Name),
   FOREIGN KEY (Book_id) REFERENCES BOOK(Book_id) ON DELETE CASCADE
);
```

```
CREATE TABLE LIBRARY_PROGRAMME (
   Programme_id INT PRIMARY KEY,
   Programme_Name VARCHAR(255),
   Address VARCHAR(255)
);

CREATE TABLE BOOK_COPIES (
   Book_id INT,
   Programme_id INT,
   No_of_Copies INT,
   PRIMARY KEY (Book_id, Programme_id),
   FOREIGN KEY (Book_id) REFERENCES BOOK(Book_id) ON DELETE CASCADE,
   FOREIGN KEY (Programme_id) REFERENCES
LIBRARY_PROGRAMME(Programme_id) ON DELETE CASCADE
);

CREATE TABLE BOOK_LENDING (
   Book_id INT,
   Programme_id INT,
   Card_No INT,
   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
);

INSERT INTO PUBLISHER (Name, Address, Phone)
VALUES('Publisher A', 'Address A', 1234567858),
     ('Publisher B', 'Address B', 9876543245),
     ('Publisher C', 'Address C', 1112223312),
     ('Publisher D', 'Address C', 3568974155),
     ('Publisher E', 'Address C', 9886632471);

INSERT INTO BOOK (Book_id, Title, Publisher_Name, Pub_Year)
VALUES
   (1, 'Introduction to SQL', 'Publisher A', 2015),
   (2, 'Data Science Basics', 'Publisher B', 2018),
   (3, 'Programming in Python', 'Publisher C', 2020),
   (4, 'Web Development Guide', 'Publisher A', 2017),
   (5, 'Machine Learning Fundamentals', 'Publisher B', 2019);

INSERT INTO BOOK_AUTHORS (Book_id, Author_Name)
VALUES
   (1, 'Author1'),
   (1, 'Author2'),
   (2, 'Author3'),
   (3, 'Author4'),
   (4, 'Author5');

INSERT INTO LIBRARY_PROGRAMME (Programme_id, Programme_Name, Address)
VALUES
   (1, 'Programme A', 'Library A Address'),
   (2, 'Programme B', 'Library B Address'),
   (3, 'Programme C', 'Library C Address');
   (4, 'Programme D', 'Library D Address');
   (5, 'Programme E', 'Library E Address');

INSERT INTO BOOK_COPIES (Book_id, Programme_id, No_of_Copies)
VALUES
   (1, 1, 5),
   (2, 1, 3),
   (3, 2, 2),
   (4, 3, 7),
   (5, 1, 4);

INSERT INTO BOOK_LENDING (Book_id, Programme_id, Card_No, Date_Out,
Due_Date)
VALUES
   (1, 1, 101, '2017-02-15', '2017-03-15'),
   (2, 1, 101, '2017-04-20', '2017-05-20'),
   (3, 2, 101, '2017-01-10', '2017-02-10'),
   (4, 3, 104, '2017-06-05', '2017-07-05'),
   (5, 1, 105, '2019-04-12', '2019-05-12');
```

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

```
SELECT B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME, BA.AUTHOR_NAME,
BC.NO_OF_COPIES, LP.PROGRAMME_ID
FROM BOOK B
JOIN BOOK_AUTHORS BA ON B.BOOK_ID = BA.BOOK_ID
JOIN BOOK_COPIES BC ON B.BOOK_ID = BC.BOOK_ID
JOIN LIBRARY_PROGRAMME LP ON BC.PROGRAMME_ID = LP.PROGRAMME_ID;
```

```
+---------+----------------------------+----------------+-------------+--------------+--------------+
| BOOK_ID | TITLE                      | PUBLISHER_NAME | AUTHOR_NAME |
NO_OF_COPIES | PROGRAMME_ID |
+---------+----------------------------+----------------+-------------+--------------+--------------+
| 1       | Introduction to SQL        | Publisher A    | Author1     | 5            | 1            |
| 1       | Introduction to SQL        | Publisher A    | Author2     | 5            | 1            |
| 2       | Data Science Basics         | Publisher B    | Author3      | 3            | 1            |
| 3       | Programming in Python       | Publisher C    | Author4      | 2            | 2            |
| 4       | Web Development Guide       | Publisher A    | Author5      | 7            | 3            |
| 5       | Machine Learning Fundamentals| Publisher B    | Author1      | 4            | 1            |
+---------+----------------------------+----------------+-------------+--------------+--------------+
```

2. Get the particulars of borrowers who have borrowed more than 1 books, but from Jan 2017 to Jun 2017.

```
SELECT CARD_NO
FROM BOOK_LENDING
WHERE DATE_OUT BETWEEN '2017-01-01' AND '2017-07-30'
GROUP BY CARD_NO
HAVING COUNT(*) > 1;
+--------------+
| CARD_NO |
+--------------+
| 101            |
+--------------+
```

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

```
Delete from Book where Book_id=4;
```

Display content of BOOK table

```
SELECT * FROM BOOK;
+-----------+----------------------------+----------------+--------+
| Book_id | Title                       | Publisher_Name | Pub_Year |
+---------+----------------------------+----------------+--------+
| 1       | Introduction to SQL         | Publisher A    | 2015    |
| 2       | Data Science Basics         | Publisher B    | 2018    |
| 3       | Programming in Python       | Publisher C    | 2016    |
| 5       | Machine Learning Fundamentals| Publisher B    | 2017    |
+---------+----------------------------+----------------+--------+
```

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;

+--------+
| Pub_Year|
+--------+
| 2015   |
| 2018   |
| 2016   |
| 2017   |
| 2017   |
+--------+
```

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

```
CREATE VIEW V_BOOKS AS
SELECT B.BOOK_ID, B.TITLE, BC.NO_OF_COPIES
FROM BOOK B, BOOK_COPIES BC, LIBRARY_PROGRAMME LP
WHERE B.BOOK_ID=BC.BOOK_ID
AND BC.PROGRAMME_ID=LP.PROGRAMME_ID;

SELECT * FROM V_BOOKS;

+---------+----------------------------+-------------+
| BOOK_ID | TITLE                      | NO_OF_COPIES|
+---------+----------------------------+-------------+
| 1       | Introduction to SQL        | 5           |
| 2       | Data Science Basics        | 3           |
| 3       | Programming in Python      | 2           |
| 5       | Machine learning Fundamentals |4         |
+---------+----------------------------+-------------+
```

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.

```
CREATE TABLE SALESMAN (
    Salesman_id INT PRIMARY KEY,
    Name VARCHAR(255),
    City VARCHAR(255),
    Commission VARCHAR(255)
);

CREATE TABLE CUSTOMER (
    Customer_id INT PRIMARY KEY,
    Cust_Name VARCHAR(255),
    City VARCHAR(255),
    Grade INT,
    Salesman_id INT,
    FOREIGN KEY (Salesman_id) REFERENCES SALESMAN(Salesman_id) ON DELETE
CASCADE
);

CREATE TABLE ORDERS (
    Ord_No INT PRIMARY KEY,
    Purchase_Amt INT,
    Ord_Date DATE,
    Customer_id INT,
    Salesman_id INT,
    FOREIGN KEY (Customer_id) REFERENCES CUSTOMER(Customer_id) ON DELETE
CASCADE,
    FOREIGN KEY (Salesman_id) REFERENCES SALESMAN(Salesman_id) ON DELETE
CASCADE
);

INSERT INTO SALESMAN (Salesman_id, Name, City, Commission)
VALUES
  (1, 'John Doe', 'Bangalore', '10%'),
  (2, 'Jane Smith', 'Mumbai', '12%'),
  (3, 'Bob Johnson', 'Chennai', '8%'),
  (4, 'Alice Brown', 'Bangalore', '15%'),
  (5, 'Charlie Wilson', 'Mumbai', '10%');

INSERT INTO CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)
VALUES
  (101, 'Customer1', 'Bangalore', 5, 1),
  (102, 'Customer2', 'Mumbai', 8, 2),
  (103, 'Customer3', 'Chennai', 6, 3),
  (104, 'Customer4', 'Bangalore',7, 5),
  (105, 'Customer5', 'Mumbai', 9, 5);

INSERT INTO ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)
VALUES
  (1001, 500, '2023-01-01', 101, 1),
  (1002, 700, '2023-01-02', 102, 2),
  (1003, 300, '2023-01-03', 103, 3),
  (1004, 1000, '2023-01-01', 104, 4),
  (1005, 600, '2023-01-02', 105, 5);
```

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

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

```
GRADE         CustomerCount
8         1
9         1
```

2. Find the name and numbers of all salesman 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
);
```

```
SALESMAN_ID     NAME
5              Charlie wilson
```

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

```
SELECT Name, City
FROM SALESMAN
WHERE Salesman_id IN (SELECT DISTINCT Salesman_id FROM CUSTOMER)
UNION
SELECT Name, City
FROM SALESMAN
WHERE Salesman_id NOT IN (SELECT DISTINCT Salesman_id FROM CUSTOMER);
```

```
| Name          | City      |
| -------------- | --------- |
| John Doe       | Bangalore |
| Jane Smith     | Mumbai    |
| Bob Johnson    | Chennai   |
| Alice Brown    | Bangalore |
| Charlie Wilson | Mumbai    |
```

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

```
SELECT B.ORD_DATE, A.SALESMAN_ID, A.NAME
FROM SALESMAN A
JOIN ORDERS B ON A.SALESMAN_ID = B.SALESMAN_ID
WHERE B.PURCHASE_AMT = (
   SELECT MAX(PURCHASE_AMT)
   FROM ORDERS C
   WHERE C.ORD_DATE = B.ORD_DATE
);
```

```
| ORD_DATE   | SALESMAN_ID | NAME          |
|------------|-------------|---------------|
| 2023-01-01 | 4           | Alice Brown   |
| 2023-01-02 | 2           | Jane Smith    |
| 2023-01-03 | 3           | Bob Johnson   |
```

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

```
DELETE FROM SALESMAN WHERE Salesman_id = 5;
```

3. Consider the schema for Movie Database:
ACTOR(Act_id, Act_Name, Act_Gender)
DIRECTOR(Dir_id, Dir_Name, Dir_Phone)
MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)
MOVIE_CAST(Act_id, Mov_id, Role)
RATING(Mov_id, Rev_Stars) Write SQL queries to
1. List the titles of all movies directed by _Hitchcock‘.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.
5. Update rating of all movies directed by _Steven Spielberg‘ to 5.

```
-- Table creation
CREATE TABLE ACTOR (
   Act_id INT PRIMARY KEY,
   Act_Name VARCHAR(255),
   Act_Gender VARCHAR(10)
);
```

```sql
CREATE TABLE DIRECTOR (
   Dir_id INT PRIMARY KEY,
   Dir_Name VARCHAR(255),
   Dir_Phone VARCHAR(15)
);

CREATE TABLE MOVIES (
   Mov_id INT PRIMARY KEY,
   Mov_Title VARCHAR(255),
   Mov_Year INT,
   Mov_Lang VARCHAR(50),
   Dir_id INT,
   FOREIGN KEY (Dir_id) REFERENCES DIRECTOR(Dir_id)
);

CREATE TABLE MOVIE_CAST (
   Act_id INT,
   Mov_id INT,
   Role VARCHAR(50),
   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 PRIMARY KEY,
   Rev_Stars INT,
   FOREIGN KEY (Mov_id) REFERENCES MOVIES(Mov_id)
);

-- Data insertion
INSERT INTO ACTOR VALUES (1, 'Actor1', 'Male');
INSERT INTO ACTOR VALUES (2, 'Actor2', 'Female');
INSERT INTO ACTOR VALUES (3, 'Actor3', 'Male');
INSERT INTO ACTOR VALUES (4, 'Actor4', 'Female');
INSERT INTO ACTOR VALUES (5, 'Actor5', 'Male');

INSERT INTO DIRECTOR VALUES (1, 'Hitchcock', '123-456-7890');
INSERT INTO DIRECTOR VALUES (2, 'Spielberg', '987-654-3210');
INSERT INTO DIRECTOR VALUES (3, 'Director3', '111-222-3333');
INSERT INTO DIRECTOR VALUES (4, 'Director4', '444-555-6666');
INSERT INTO DIRECTOR VALUES (5, 'Director5', '777-888-9999');

INSERT INTO MOVIES VALUES (1, 'Movie1', 1998, 'English', 1);
INSERT INTO MOVIES VALUES (2, 'Movie2', 2005, 'Spanish', 2);
INSERT INTO MOVIES VALUES (3, 'Movie3', 2010, 'French', 3);
INSERT INTO MOVIES VALUES (4, 'Movie4', 2018, 'German', 4);
INSERT INTO MOVIES VALUES (5, 'Movie5', 2022, 'Italian', 5);

INSERT INTO MOVIE_CAST VALUES (1, 1, 'Lead');
INSERT INTO MOVIE_CAST VALUES (2, 1, 'Supporting');
INSERT INTO MOVIE_CAST VALUES (3, 2, 'Lead');
INSERT INTO MOVIE_CAST VALUES (4, 2, 'Supporting');
INSERT INTO MOVIE_CAST VALUES (5, 3, 'Lead');

INSERT INTO RATING VALUES (1, 4);
INSERT INTO RATING VALUES (2, 3);
INSERT INTO RATING VALUES (3, 5);
INSERT INTO RATING VALUES (4, 4);
INSERT INTO RATING VALUES (5, 2);
```

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

```
SELECT Mov_Title
FROM MOVIES
WHERE Dir_id = (SELECT Dir_id FROM DIRECTOR WHERE Dir_Name = 'Hitchcock');

+-----------+
| Mov_Title |
+-----------+
| Movie1    |
+-----------+
```

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

```
SELECT Mov_Title
FROM MOVIES
WHERE Mov_id IN (
   SELECT Mov_id
   FROM MOVIE_CAST
   GROUP BY Act_id
   HAVING COUNT(Mov_id) >= 2
);

+-----------+
| Mov_Title |
+-----------+
| Movie1    |
| Movie2    |
+-----------+
```

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

```
SELECT DISTINCT A.Act_Name
FROM ACTOR A
JOIN MOVIE_CAST MC ON A.Act_id = MC.Act_id
JOIN MOVIES M ON MC.Mov_id = M.Mov_id
WHERE M.Mov_Year < 2000 OR M.Mov_Year > 2015;
```

```
+---------+
| Act_Name|
+---------+
| Actor1  |
| Actor3  |
| Actor5  |
+---------+
```

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

```
SELECT M.Mov_Title, R.Rev_Stars
FROM MOVIES M
JOIN RATING R ON M.Mov_id = R.Mov_id
ORDER BY M.Mov_Title;
```

```
+-----------+-----------+
| Mov_Title | Rev_Stars |
+-----------+-----------+
| Movie1    | 4         |
| Movie2    | 3         |
| Movie3    | 5         |
| Movie4    | 4         |
| Movie5    | 2         |
+-----------+-----------+
```

5. **Update the 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 = (SELECT Dir_id FROM DIRECTOR WHERE Dir_Name = 'Spielberg')
);
```