**LIBRARY DATABASE**

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

```
SELECT B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME,
A.AUTHOR_NAME,C.NO_OF_COPIES,L.PROGRAMME_ID
FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C, LIBRARY_PROGRAMME L
WHERE B.BOOK_ID=A.BOOK_ID
AND B.BOOK_ID=C.BOOK_ID
AND L.PROGRAMME_ID=C.PROGRAMME_ID;
```

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

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

```
DELETE FROM BOOK
WHERE BOOK_ID=3;
SELECT * FROM BOOK;
SELECT * FROM BOOK_AUTHORS;
```

**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;
```

**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, C.NO_OF_COPIES
FROM
BOOK B, BOOK_COPIES C, LIBRARY_PROGRAMME L
WHERE B.BOOK_ID=C.BOOK_ID
AND C.PROGRAMME_ID=L.PROGRAMME_ID;
SELECT * FROM V_BOOKS;
```

**ORDER DATABASE**

**I) Count the customers with grades above Bangalore’s average**

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

**II) Find the name and numbers of all salesman who had more than one customer**

```
SELECT SALESMAN_ID, NAME
FROM SALESMAN S
WHERE (SELECT COUNT(*)
FROM CUSTOMER C
WHERE C.SALESMAN_ID=S.SALESMAN_ID) > 1;
```

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

```
SELECT S.SALESMAN_ID, S.NAME, C.CUST_NAME, S.COMMISSION
FROM SALESMAN S, CUSTOMER C
WHERE S.CITY=C.CITY
UNION
SELECT S.SALESMAN_ID,S.NAME,'NO MATCH',S.COMMISSION
FROM SALESMAN S
WHERE CITY NOT IN
(SELECT CITY
FROM CUSTOMER)
ORDER BY 1 ASC;
```

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

```
CREATE VIEW V_SALESMAN AS
SELECT O.ORDER_DATE, S.SALESMAN_ID, S.NAME
FROM SALESMAN S,ORDERS O
WHERE S.SALESMAN_ID = O.SALESMAN_ID
AND O.PURCHASE_AMOUNT= (SELECT MAX(PURCHASE_AMOUNT)
FROM ORDERS C
WHERE C.ORDER_DATE=O.ORDER_DATE);
SELECT * FROM V_SALESMAN;
```

**V) 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;
SELECT * FROM ORDERS;
```

**MOVIE DATABASE**

**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');
```

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

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

```
SELECT MOV_TITLE
FROM MOVIES M,MOVIE_CAST MC
WHERE M.MOV_ID=MC.MOV_ID AND 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
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;
```

**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 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'));
```

**COLLEGE DATABASE**

**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';
```

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

**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;
```

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

**Create a view of Test1 marks of student USN ‘1NC20CS101’ in all Courses.**

```
CREATE VIEW STUDENT_TEST1_MARKS_V
AS
SELECT TEST1, SUBCODE
FROM IAMARKS
WHERE USN = '1NC20IS101';
SELECT * FROM STUDENT_TEST1_MARKS_V;
```

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

**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;
```

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

**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;
```

**COMPANY DATABASE**

**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';
```

**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';
```

**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.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));
```

**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;
```