1. **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-JUL-2017„
GROUP BY CARD_NO
HAVING COUNT (*)>3;
```

2. **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;
```

3. **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;
```

4. **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_BRANCH L
WHERE B.BOOK_ID=C.BOOK_ID
AND C.BRANCH_ID=L.BRANCH_ID;
```

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

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

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

```
SELECT SALESMAN_ID, NAME FROM
 SALESMAN A
WHERE 1 < (SELECT COUNT (*) FROM
 CUSTOMER1
 WHERE SALESMAN_ID=A.SALESMAN_ID);
```

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

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

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

```
CREATE VIEW ELITSALESMAN AS
SELECT B.ORD_DATE, A.SALESMAN_ID, A.NAME FROM
SALESMAN A, ORDERS B
WHERE A.SALESMAN_ID = B.SALESMAN_ID
AND B.PURCHASE_AMT=(SELECT MAX (PURCHASE_AMT)
FROM ORDERS C
WHERE C.ORD_DATE = B.ORD_DATE);
```

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

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

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

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

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

```
SELECT MOV_TITLE FROM
MOVIES
WHERE DIR_ID IN (SELECT DIR_ID
FROM DIRECTOR
WHERE DIR_NAME = _HITCHCOCK„);
```

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

```
SELECT MOV_TITLE
FROM MOVIES M, MOVIE_CAST MV
WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN (SELECT ACT_ID
FROM MOVIE_CAST GROUP BY ACT_ID HAVING COUNT (ACT_ID)>1)
GROUP BY MOV_TITLE HAVING
COUNT (*)>1;
```

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

```
SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
FROM ACTOR A JOIN
MOVIE_CAST C
        ON A.ACT_ID=C.ACT_ID JOIN
MOVIES M
ON C.MOV_ID=M.MOV_ID
WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015; OR

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

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

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

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

```
KL
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„));
```

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
```

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
WHERES.USN = C.USN AND
SS.SSID = C.SSID
GROUP BY SS.SEM, SS.SEC, S.GENDER
ORDER BY SEM;
```

3. **Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.**

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