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

select a.author_name,b.bookid,b.title, b.pub_name,bc.no_of_copies as branch_copies from book b, book_authors a, book_copies bc where a.bookid=bc.bookid and a.bookid=b.bookid and bc.programme_idin (select programme_idfrom library_branch group by progeamme_id)

2. **Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017**.

select card_no, count(*) from book_lending where date_out between '01-jan-2017' and '30-sep-2017' group by card_no having count(*) >=1;

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

delete from book where bookid=1004;

update book_lending set bookid=1002 where bookid=1004;

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

create table book_part (bookid int, title varchar(10), pub_name varchar(10), pub_year int) PARTITION BY RANGE (pub_year) (PARTITION p1 VALUES LESS THAN (1990), PARTITION p2 VALUES LESS THAN (2000), PARTITION p3 VALUES Less THAN (2010))

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

create view Display_books as select b.bookid,b.title, bc.no_of_copies, bc.progeamme_id from book b,book_copies bc where b.bookid=bc.bookid;

2(1).Count the customers with grades above Bangalore's average.

select count(*),GRADES from customer 2 where grades > (select avg(grades)from customer 3 where city='bangalore') group by GRADES;

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

select salesman_id,sname,count(*) from customer,salesman where sales_id=salesman_id group by salesman_id,sname having count(*)>2;

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

(select s.sname from salesman s, customer c where s.city=c.city and sales_id=salesman_id) UNION (select s.sname from salesman s, customer c where s.city!=c.city and sales_id=salesman_id)

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

select s.SALESMAN_ID,s.sname, s.city, count(*) as highest_orders 2 from customer c, salesman s 3 where s.SALESMAN_ID=c.SALES_ID 4 group by s.SALESMAN_ID,s.sname, s.city 5 having count(*) = (select max(count(sales_id)) from orders 6 where ord_date='&ord_date' 7 group by sales_id);

create view highest_order as select s.SALESMAN_ID,s.sname, s.city, count(*) as highest_orders 2 from customer c, salesman s 3 where s.SALESMAN_ID=c.SALES_ID 4 group by s.SALESMAN_ID,s.sname, s.city 5 having count(*) = (select max(count(sales_id)) from orders 6 where ord_date='&ord_date' 7 group by sales_id);

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;

1. **3(1) List the titles of all movies directed by 'Hitchcock'.**
   select m.mov_title 2 from movies m, director d 3 where m.dir_id=d.dir_id and dir_name='hitchcock';
2. **Find the movie names where one or more actors acted in two or more movies.**
   select m.mov_title from movies m where exists (select mc.mov_id, count(*) from movie_cast mc where m.mov_id=mc.mov_id and exists (select act_id, count(*) from movie_cast group by act_id having count(*) >1) group by mov_id having count(*)>2);
3. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**
   (select a.act_name from actor a, movie_cast mc,movies m where a.act_id=mc.act_id and m.mov_id=mc.mov_id and mov_year<2000); intersect (select a.act_name from actor a, movie_cast mc,movies m where a.act_id=mc.act_id and m.mov_id=mc.mov_id and mov_year>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,REV_STARS as highest_rating from movies m,rating r where m.mov_id=r.mov_id and rev_stars in (select max(REV_STARS) from rating)
   group by mov_title,REV_STARS;
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 natural join director where dir_name='stephen');

1. **4(1)List all the student details studying in fourth semester 'C' section.**

select s.usn,s.sname 2 from student s,semsec sc,classes c 3 where sc.ssid=c.ssid and s.usn=c.usn and sc.sem=5 and sc.sec='C';

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

select sc.ssid,sem,sec,gender,count(*) from semsec sc,classes c, student s where sc.ssid=c.ssid and s.usn=c.usn group by sc.ssid,sem,sec,gender;

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

create view stud_iamarks as select usn,sname,subcode, test1 from student natural join iamarks where usn='&usn';

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

alter table iamarks drop column finalia;

alter table iamarks add finalia int;

update iamarks set finalia= round((test1+test2+test3-least(test1,test2,test3))/2);

```
CREATE OR REPLACE PROCEDURE AVGMARKS
2 IS
3 CURSOR C_IAMARKS IS
4 SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B, GREATEST(TEST3,TEST2) AS C
5 FROM IAMARKS
 6 WHERE FINALIA IS NULL
7 FOR UPDATE;
 8 C_A NUMBER;
9 C_B NUMBER;
10 C_C NUMBER;
11 C_SM NUMBER;
12 C_AV NUMBER;
 13 BEGIN
14 OPEN C_IAMARKS;
15 LOOP
```

```
16 FETCH C_IAMARKS INTO C_A, C_B, C_C;
 17 EXIT WHEN C_IAMARKS%NOTFOUND;
18 --DBMS_OUTPUT.PUT_LINE(C_A || ' ' || C_B || ' ' || C_C);
19 IF (C_A >= C_B AND C_B >= C_C) THEN
 20 C_SM:=C_A+C_B; 21 ELSIF (C_A >= C_B AND C_C >= C_B) THEN
22 C_SM:=C_A+C_C;
23 ELSIF (C_B >= C_A AND C_A >= C_C) THEN
24 C_SM:=C_B+C_A; 25 ELSIF (C_B >= C_A AND C_C >= C_A) THEN
26 C_SM:=C_B+C_C;
27 ELSIF(C_C >= C_A AND C_A >= C_B) THEN
 28 C_SM:=C_C+C_A; 29 ELSIF(C_C >= C_A AND C_B >= C_A)THEN
30 C_SM:=C_C+C_B; 31 END IF;
32 C_AV:=C_SM/2;
 33 --DBMS_OUTPUT.PUT_LINE('SUM = '||C_SM);
34 --DBMS_OUTPUT.PUT_LINE('AVERAGE = '||C_AV);
35 UPDATE IAMARKS SET FINALIA = C_AV WHERE CURRENT OF C_IAMARKS;
 36 END LOOP;
37 CLOSE C_IAMARKS;
38 END;/
```

1. **5(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 pq.pno, pq.pname from project pq where pno in(2 (select p.pno 3 from project p, employee e, workson w 4 where w.pno=p.pno and e.ssn=w.ssn and e.ename='scott') 5 union 6 (select p.pno 7 from project p, employee e,department d 8 where e.ssn=d.mgrssn and p.dno=d.dno and ename='scott') 9);
2. **Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.**
   SELECT SALARY*1.1 2 FROM EMPLOYEE e, PROJECT p, WORKSON w 3 WHERE PNAME='rolls' and e.ssn=w.ssn and p.pno=w.pno;
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 max(salary),min(salary),avg(salary), sum(salary) from employee e, department d where e.dno=d.dno and DNAME='accounting';
4. **Retrieve the name of each employee who works on all the projects controlledby department number 5 (use NOT EXISTS operator).**
   select e.ename from employee e where exists (select * from workson w,project p where p.pno=w.pno and e.ssn=w.ssn and p.dno=2);
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 e.ename from employee e where not exists (select * from workson w,project p where p.pno=w.pno and e.ssn=w.ssn and p.dno=2);