1. **Consider the customer-sale scenario given below.**

CUSTOMER(Cust id : integer, cust_name: string)

ITEM(item_id: integer, item_name: string, price: integer)

SALE(bill_no: integer, bill_date: date, cust_id: integer, item_id: integer, qty_sold: integer)

For the above schema, perform the following:

a) Create the tables with the appropriate integrity constraints

b) Insert around 10 records in each of the tables

c) List all the bills for the current date with the customer names and item numbers

d) List the total Bill details with the quantity sold, price of the item and the final amount

e) List the details of the customer who have bought a product which has a price>200

f) Give a count of how many products have been bought by each customer

g) Give a list of products bought by a customer having cust_id as 5

h) List the item details which are sold as of today

i) Create a view which lists out the bill_no, bill_date, cust_id, item_id, price, qty_sold, amount.

1. create table customer1 (cust_id number(5) primary key, cust_name varchar2(15));
2. desc customer1
3. create table item(item_id number(4) primary key, item_name varchar2(15), price number);
4. create table sale(bill_no number(5) primary key,bill_date date, cust_id number(5) references customer1(cust_id), item_id number(4) references item(item_id),qty_sold number(4));
5. desc customer1;

5. dsec item;

6. dsec sale;

7. insert into customer1 values(&custid,'&custname');

8. select * from customer1;

9 .insert into item values(&item_id,'&item_name',&price)

10. select * from item;

11. insert into Sale values(1450,to_date('04-Jan-2008','dd-mm-yyyy'),100,2124.2)

12. select * from sale;

QUERIES

1. List all the bills for the current date with the customer names and item numbers

```
select c.cust_name, i.item_id, s.bill_no from customer1 c, item I, sale s
where c.cust_id=s.cust_id and s.bill_date=to_char(sysdate);
```

2. List the total Bill details with the quantity sold, price of the item and the final amount.

```
select i.price,s.item_id, (i.price*s.qty_sold) as total from item I, sale s where
i.item_id=s.item_id group by ( i.price ,s.item_id,s.qty_sold);
```

3. List the details of the customer who have bought a product which has a price>200

```
select c.cust_id, c.cust_name from customer1 c, sale s, item i where i.price>50 and
c.cust_id=s.cust_id and i.item_id=s.item_id;
```

4.Give a count of how many products have been bought by each customer

```
select cust_id, count(item_id) from sale group by cust_id;
```

5.Give a list of products bought by a customer having cust_id as 5

```
select i.item_name from item i, sale s where s.cust_id=101 and i.item_id=s.item_id;
```

6. List the item details which are sold as of today

```
select i.item_id, i.item_name from item I, sale s where i.item_id=s.item_id and
s.bill_date=to_char(sysdate);
```

7. Create a view which lists out the bill_no, bill_date, cust_id, item_id, price, qty_sold, amount

```
create view cust as (select s.bill_no, s.bill_date, c.cust_id, i.item_id, i.price, s.qty_sold from
customer1 c,sale s, item i where c.cust_id=s.cust_id and i.item_id=s.item_id);

select * from cust;
```

2.**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_BRANCH (Programme_id, Programme_Name, Address)

Write SQL queries to

a) Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each Programme, etc.
b) Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017
c) Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
d) Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
e) Create a view of all books and its number of copies that are currently available in the Library.

1. CREATE TABLE PUBLISHER
(NAME VARCHAR2 (20) PRIMARY KEY, PHONE INTEGER, ADDRESS VARCHAR2 (20));

2. CREATE TABLE BOOK
(BOOK_ID INTEGER PRIMARY KEY, TITLE VARCHAR2 (20), PUB_YEAR VARCHAR2 (20),
PUBLISHER_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);

3. CREATE TABLE BOOK_AUTHORS
(AUTHOR_NAME VARCHAR2 (20), BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE, PRIMARY KEY (BOOK_ID, AUTHOR_NAME));

4. CREATE TABLE LIBRARY_BRANCH
(BRANCH_ID INTEGER PRIMARY KEY, BRANCH_NAME VARCHAR2 (50),
ADDRESS VARCHAR2 (50));

5. CREATE TABLE BOOK_COPIES
(NO_OF_COPIES INTEGER, BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE CASCADE, PRIMARY KEY (BOOK_ID, BRANCH_ID));

6. CREATE TABLE CARD
(CARD_NO INTEGER PRIMARY KEY);

7. CREATE TABLE BOOK_LENDING
(DATE_OUT DATE, DUE_DATE DATE, BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,

```
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE CASCADE,
CARD_NO REFERENCES CARD (CARD_NO) ON DELETE CASCADE, PRIMARY KEY (BOOK_ID,
BRANCH_ID, CARD_NO));

desc*all;

INSERT INTO PUBLISHER VALUES (_MCGRAW-HILL', 9989076587, _BANGALORE');
INSERT INTO PUBLISHER VALUES (_PEARSON', 9889076565, _NEWDELHI');
INSERT INTO PUBLISHER VALUES (_RANDOM HOUSE', 7455679345, _HYDRABAD');
INSERT INTO PUBLISHER VALUES (_HACHETTE LIVRE', 8970862340, _CHENAI');
INSERT INTO PUBLISHER VALUES (_GRUPO PLANETA', 7756120238, _BANGALORE');

INSERT INTO BOOK VALUES (1,'DBMS','JAN-2017', _MCGRAW-HILL');
INSERT INTO BOOK VALUES (2,'ADBMS','JUN-2016', _MCGRAW-HILL');
INSERT INTO BOOK VALUES (3,'CN','SEP-2016', _PEARSON');
INSERT INTO BOOK VALUES (4,'CG','SEP-2015', _GRUPO PLANETA');
INSERT INTO BOOK VALUES (5,'OS','MAY-2016', _PEARSON');

INSERT INTO BOOK_AUTHORS VALUES ('NAVATHE', 1);
INSERT INTO BOOK_AUTHORS VALUES ('NAVATHE', 2);
INSERT INTO BOOK_AUTHORS VALUES ('TANENBAUM', 3);
INSERT INTO BOOK_AUTHORS VALUES ('EDWARD ANGEL', 4);
INSERT INTO BOOK_AUTHORS VALUES ('GALVIN', 5);

INSERT INTO LIBRARY_BRANCH VALUES (10,'RR NAGAR','BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (11,'RNSIT','BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (12,'RAJAJI NAGAR', 'BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (13,'NITTE','MANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (14,'MANIPAL','UDUPI');

INSERT INTO BOOK_COPIES VALUES (10, 1, 10);
INSERT INTO BOOK_COPIES VALUES (5, 1, 11);
INSERT INTO BOOK_COPIES VALUES (2, 2, 12);
INSERT INTO BOOK_COPIES VALUES (5, 2, 13);
INSERT INTO BOOK_COPIES VALUES (7, 3, 14);
INSERT INTO BOOK_COPIES VALUES (1, 5, 10);
INSERT INTO BOOK_COPIES VALUES (3, 4, 11);

INSERT INTO CARD VALUES (100);
INSERT INTO CARD VALUES (101);
INSERT INTO CARD VALUES (102);
INSERT INTO CARD VALUES (103);
INSERT INTO CARD VALUES (104);

INSERT INTO BOOK_LENDING VALUES ('01-JAN-17','01-JUN-17', 1, 10, 101);
INSERT INTO BOOK_LENDING VALUES ('11-JAN-17','11-MAR-17', 3, 14, 101);
INSERT INTO BOOK_LENDING VALUES ('21-FEB-17','21-APR-17', 2, 13, 101);
INSERT INTO BOOK_LENDING VALUES ('15-MAR-17','15-JUL-17', 4, 11, 101);
INSERT INTO BOOK_LENDING VALUES (_12-APR-17','12-MAY-17', 1, 11, 104);
SELECT * FROM PUBLISHER;
Select * from all;
```

QUERIES

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, A.AUTHOR_NAME, C.NO_OF_COPIES,
L.BRANCH_ID  FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C, LIBRARY_BRANCH L
WHERE B.BOOK_ID=A.BOOK_ID  AND B.BOOK_ID=C.BOOK_ID  AND
L.BRANCH_ID=C.BRANCH_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-JUL-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;
```

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;
```

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

3 Consider the Employee-pay scenario given below.

EMPLOYEE(**emp_id : integer**, emp_name: string)
DEPARTMENT(**dept_id: integer**, dept_name:string)
PAYDETAILS(**emp_id : integer**, **dept_id: integer**, basic: integer, deductions: integer, additions: integer, DOJ: date)
PAYROLL**(emp_id : integer**, pay_date: date)
For the above schema, perform the following:

a) Create the tables with the appropriate integrity constraints
b) Insert around 10 records in each of the tables
c) List the employee details department wise

d) List all the employee names who joined after particular date
e) List the details of employees whose basic salary is between 10,000 and 20,000
f) Give a count of how many employees are working in each department
g) Give a names of the employees whose netsalary>10,000

1. create table employee(emp_id int(5) primary key,emp_name varchar2(25));
2. create table department(dept_id int(5) primary key,dept_name varchar2(20));
3. create table paydetails(emp_id int(5) references employee(emp_id),dept_id int(5) reerences department(dept_id),basic int(7,2),deductions int(5,2),additions int(5,2),doj date);

4. create table payroll(emp_id int(5)references employee(emp_id),pay_date date);

5. desc employee;

6. desc department;

7. desc paydetails;

8. desc payroll;

9. insert into employee values(&emp_id,'&emp_name');

select * from employee;

10. insert into department values(&dept_id,'&dept_name');

select * from department;

11. insert into paydeatils values(&emp_id,&dept_id,
&basic,&deductions,&additions,&doj);

select * from paydeatils;

12. insert into payroll values(&emp_id,'&date');

```
select * from payroll;
```

QUERIES

13. List all the employee names who joined after particular date

```
select e,empname from employee e,paydet p where e.empid=p.empid
and p.doj>='05-mar-06';
```

14.List the details of employees whose basic salary is between 10k and 20k

```
select e.emp_id , e.emp_name,d.dept_id , d.dept_name , pd.basic from
employee e , department d , paydetails pd , payroll pr where
e.emp_id=pd.emp_id and d.dept_id=pd.dept_id and e.emp_id=pr.emp_id and
pd.basic between 600 and 1000;
```

15. Give a count of how many employees are working in each department

```
select count(empid),deptid from paydet group by deptid;
```

16. Give a names of the employees whose netsalary>10,000

```
select empname from employee where empid in(select empid from
paydet where basic-deduction>10000);
```

17. List the details for an employee_id=5

```
select * from employee where empid=5;
```

18. Create a view which lists out the emp_name, department, basic, dedeuctions, netsalary

```
create view vw as select e.emp_name , d.dept_name , pd.basic,pd.deductions ,
(pd.basic+pd.additions-pd.deductions) netsalary from employee e, department d,
paydetails pd,payroll pr where e.emp_id=pd.emp_id and d.dept_id=pd.dept_id
and e.emp_id=pr.emp_id  ;

select * from vw ;
```

19. Create a view which lists the emp_name and his netsalary

```
create view vew as select e.emp_name , (pd.basic+pd.additions-pd.deductions)
netsalary from employee e, department d, paydetails pd,payroll pr where
e.emp_id=pd.emp_id and d.dept_id=pd.dept_id and e.emp_id=pr.emp_id  ;

select * from vew ;
```

4. Question 4

1. CREATE TABLE DEPARTMENT (DNO NUMBER (2), DNAME VARCHAR2 (20));
2. ALTER TABLE DEPARTMENT ADD PRIMARY KEY (DNO);
3. CREATE TABLE BRANCH (BCODE NUMBER (3), BNAME VARCHAR2 (25), DNO NUMBER (2));
4. ALTER TABLE BRANCH ADD PRIMARY KEY (BCODE);
5. ALTER TABLE BRANCH ADD FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO);
6. CREATE TABLE BRANCH_COURSE (BCODE NUMBER(3),
CCODE NUMBER(4),
SEMESTER NUMBER(2));
7. ALTER TABLE BRANCH_COURSE ADD PRIMARY KEY (BCODE, CCODE);
8. ALTER TABLE BRANCH_COURSE ADD FOREIGN KEY (BCODE) REFERENCES BRANCH (BCODE);
9. ALTER TABLE BRANCH_COURSE ADD FOREIGN KEY (CCODE) REFERENCES COURSE (CCODE);
10. CREATE TABLE STUDENT (ROLLNO NUMBER (5),
NAME VARCHAR2 (20),
DOB DATE, GENDER CHAR(2),
DOA DATE, BCODE NUMBER(3));

11. ALTER TABLE STUDENT ADD PRIMARY KEY (ROLLNO);
ALTER TABLE STUDENT ADD FOREIGN KEY (BCODE) REFERENCES BRANCH (BCODE);
ALTER TABLE ADD CONSTRAINT CHK CHECK (GENDER IN ('M','F'));
ALTER TABLE ADD CONSTRAINT CHK2 CHECK (DOA < TO_DATE('31-4-2016,'DD-MM-YYYY');

12. CREATE TABLE COURSE (CCODE NUMBER (4), CNAME VARCHAR2 (25), CREDITS NUMBER (2), DNO NUMBER (2));

13. ALTER TABLE COURSE ADD PRIMARY KEY (CCODE);
14. ALTER TABLE COURSE ADD FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO));
15. CREATE TABLE ENROLLS (ROLLNO NUMBER (5), CCODE NUMBER (4), SESS VARCHAR2 (15), GRADE CHAR (2));

16. ALTER TABLE ENROLLS ADD PRIMARY KEY (ROLLNO, CCODE, SESS);
17. ALTER TABLE ENROLLS ADD FOREIGN KEY ROLLNO) REFERENCES STUDENT (ROLLNO);
18. ALTER TABLE ENROLLS ADD FOREIGN KEY (CCODE) REFERENCES COURSE (CCODE);

19. INSERT INTO COURSE VALUES (1011, 'LINEAR ALGEBRA', 2,1);

20. INSERT INTO STUDENT VALUES (12001, 'RAMESH KAUSHIK', TO_DATE('3-4-1989','DD-MM-YYYY') ,'M' , TO_DATE('24-4-2016','DD-MM-YYYY'), 110);
21. INSERT INTO ENROLLS VALUES(12001, 1112, 'APRIL2013','D');

QUERIES

22. Develop a SQL query to list details of Departments that offer more than 3 branches.

SELECT * FROM DEPARTMENT D WHERE D.DNO IN (SELECT B.DNO FROM BRANCH B GROUP BY B.DNO HAVING COUNT (B.DNO) > 3);

23. Develop a SQL query to list the details of Departments that offer more than 6 courses.

SELECT * FROM DEPARTMENT D WHERE D.DNO IN (SELECT C.DNO FROM COURSE C GROUP BY C.DNO HAVING COUNT (C.CCODE) > 6);

24. Develop a SQL query to list the details of courses that are common for more than 3 branches.

SELECT * FROM COURSE C WHERE C.CCODE IN (SELECT B.CCODE FROM BRANCH_COURSE B GROUP BY B.CCODE HAVING COUNT (B.BCODE) > 3);

25. Develop a SQL query to list students who got 'S' in more than 2 courses during single enrollment.

SELECT * FROM STUDENT S WHERE S.ROLLNO IN (SELECT E.ROLLNO FROM ENROLLS E WHERE E.GRADE = 'S' GROUP BY E.ROLLNO HAVING COUNT (E.GRADE) > 2);

26. Create a view that will keep track of the roll number, name and number of courses, a student has completed successfully.
CREATE VIEW STUDATA AS SELECT E.ROLLNO, S.NAME, COUNT (E.CCODE) AS CC FROM STUDENT S, ENROLLS E WHERE E.ROLLNO = S.ROLLNO AND E.GRADE ! = 'U' GROUP BY E.ROLLNO, S.NAME;

5. 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

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.

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

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

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

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

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

desc*all;
```

Insertion of Values to Tables

```
INSERT INTO ACTOR VALUES (301,‘ANUSHKA‘,‘F‘);
INSERT INTO ACTOR VALUES (302,‘PRABHAS‘,‘M‘);
INSERT INTO ACTOR VALUES (303,‘PUNITH‘,‘M‘);
INSERT INTO ACTOR VALUES (304,‘JERMY‘,‘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, _TELAGU‘, 60);
INSERT INTO MOVIES VALUES (1002,‘BAHUBALI-1‘, 2015, _TELAGU‘, 60);
INSERT INTO MOVIES VALUES (1003,‘AKASH‘, 2008, _KANNADA‘, 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);

SELECT * FROM all;
```

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’);
```

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;c
```

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'));
```