```
-- DROP TABLES IF NEEDED
DROP TABLE ENROLLED;
DROP TABLE CLASS;
DROP TABLE STUDENT;
DROP TABLE FACULTY;

-- CREATE TABLES
CREATE TABLE STUDENT (
  snum NUMBER PRIMARY KEY,
  sname VARCHAR2(50),
  major VARCHAR2(30),
  levels VARCHAR2(10),
  age NUMBER
);

CREATE TABLE FACULTY (
  fid NUMBER PRIMARY KEY,
  fname VARCHAR2(50),
  deptid NUMBER
);

CREATE TABLE CLASS (
  cname VARCHAR2(50) PRIMARY KEY,
  meetsat VARCHAR2(20),
  room VARCHAR2(10),
  fid NUMBER,
  FOREIGN KEY (fid) REFERENCES FACULTY(fid)
);

CREATE TABLE ENROLLED (
  snum NUMBER,
  cname VARCHAR2(50),
  FOREIGN KEY (snum) REFERENCES STUDENT(snum),
  FOREIGN KEY (cname) REFERENCES CLASS(cname)
```

```
);

-- INSERT SAMPLE DATA
-- STUDENTS
INSERT INTO STUDENT VALUES (1, 'Alice', 'CS', 'JR', 20);
INSERT INTO STUDENT VALUES (2, 'Bob', 'EE', 'SR', 22);
INSERT INTO STUDENT VALUES (3, 'Charlie', 'CS', 'JR', 21);
INSERT INTO STUDENT VALUES (4, 'David', 'ME', 'FR', 18);
INSERT INTO STUDENT VALUES (5, 'Eve', 'CS', 'SO', 19);

-- FACULTY
INSERT INTO FACULTY VALUES (10, 'Harshith', 1);
INSERT INTO FACULTY VALUES (11, 'Suman', 2);
INSERT INTO FACULTY VALUES (12, 'Geeta', 3);

-- CLASSES
INSERT INTO CLASS VALUES ('DBMS', '9AM', 'R128', 10);
INSERT INTO CLASS VALUES ('OS', '10AM', 'R129', 10);
INSERT INTO CLASS VALUES ('DSA', '9AM', 'R130', 11);
INSERT INTO CLASS VALUES ('COA', '11AM', 'R131', 12);
INSERT INTO CLASS VALUES ('Math', '10AM', 'R128', 11);

-- ENROLLED
INSERT INTO ENROLLED VALUES (1, 'DBMS');
INSERT INTO ENROLLED VALUES (1, 'DSA');
INSERT INTO ENROLLED VALUES (3, 'DBMS');
INSERT INTO ENROLLED VALUES (3, 'DSA');
INSERT INTO ENROLLED VALUES (2, 'OS');
INSERT INTO ENROLLED VALUES (4, 'COA');
INSERT INTO ENROLLED VALUES (5, 'Math');

-- QUERIES
-- 1. Juniors enrolled in a class taught by Prof. Harshith
SELECT DISTINCT S.sname
```

```
FROM STUDENT S
JOIN ENROLLED E ON S.snum = E.snum
JOIN CLASS C ON E.cname = C.cname
JOIN FACULTY F ON C.fid = F.fid
WHERE S.levels = 'JR' AND F.fname = 'Harshith';

-- 2. Class names that meet in R128 or have 5+ students enrolled
SELECT cname FROM CLASS WHERE room = 'R128'
UNION
SELECT cname FROM ENROLLED GROUP BY cname HAVING COUNT(*) >= 5;

-- 3. Students enrolled in two classes that meet at the same time
SELECT DISTINCT S1.sname
FROM STUDENT S1
JOIN ENROLLED E1 ON S1.snum = E1.snum
JOIN CLASS C1 ON E1.cname = C1.cname
JOIN ENROLLED E2 ON S1.snum = E2.snum AND E1.cname <> E2.cname
JOIN CLASS C2 ON E2.cname = C2.cname
WHERE C1.meetsat = C2.meetsat;

-- 4. Faculty teaching in every room where any class is taught
SELECT fname
FROM FACULTY F
WHERE NOT EXISTS (
  SELECT room FROM CLASS
  MINUS
  SELECT room FROM CLASS C WHERE C.fid = F.fid
);

-- 5. Faculty whose combined course enrollments are less than 5
SELECT F.fname
FROM FACULTY F
JOIN CLASS C ON F.fid = C.fid
LEFT JOIN ENROLLED E ON C.cname = E.cname
```

```
GROUP BY F.fname
HAVING COUNT(E.snum) < 5;
```

2. The following relations keep track of Airline Flight information:

FLIGHTS(no: integer, fromPlace: string, toPlace: string, distance: integer, Departs: date, arrives: date,price: real)
AIRCRAFT(aid: integer, aname: string, cruisingrange: integer)

CERTIFIED(eid: integer, aid: integer)

EMPLOYEES(eid: integer, ename: string, salary: integer)

```
CREATE TABLE FLIGHTS (
   no INTEGER PRIMARY KEY,
   fromPlace VARCHAR2(50),
   toPlace VARCHAR2(50),
   distance INTEGER,
   departs DATE,
   arrives DATE,
   price NUMBER(10, 2)
);

CREATE TABLE AIRCRAFT (
   aid INTEGER PRIMARY KEY,
   aname VARCHAR2(50),
   cruisingrange INTEGER
);

CREATE TABLE CERTIFIED (
   eid INTEGER,
   aid INTEGER,
   PRIMARY KEY (eid, aid)
);

CREATE TABLE EMPLOYEES (
   eid INTEGER PRIMARY KEY,
```

```
   ename VARCHAR2(50),
   salary INTEGER
);
-- AIRCRAFT
INSERT INTO AIRCRAFT VALUES (1, 'Boeing 737', 5600);
INSERT INTO AIRCRAFT VALUES (2, 'Airbus A320', 6100);
INSERT INTO AIRCRAFT VALUES (3, 'Boeing 777', 9700);
INSERT INTO AIRCRAFT VALUES (4, 'Embraer 190', 4000);
INSERT INTO AIRCRAFT VALUES (5, 'Bombardier CRJ', 3100);

-- EMPLOYEES (pilots)
INSERT INTO EMPLOYEES VALUES (101, 'John Smith', 90000);
INSERT INTO EMPLOYEES VALUES (102, 'Alice Brown', 75000);
INSERT INTO EMPLOYEES VALUES (103, 'Bob Johnson', 85000);
INSERT INTO EMPLOYEES VALUES (104, 'Carol White', 82000);
INSERT INTO EMPLOYEES VALUES (105, 'David Lee', 70000);

-- CERTIFIED (which pilots certified for which aircraft)
INSERT INTO CERTIFIED VALUES (101, 1);
INSERT INTO CERTIFIED VALUES (101, 3);
INSERT INTO CERTIFIED VALUES (102, 2);
INSERT INTO CERTIFIED VALUES (103, 1);
INSERT INTO CERTIFIED VALUES (103, 2);
INSERT INTO CERTIFIED VALUES (103, 3);
INSERT INTO CERTIFIED VALUES (103, 4);
INSERT INTO CERTIFIED VALUES (104, 5);
INSERT INTO CERTIFIED VALUES (105, 1);

-- FLIGHTS
INSERT INTO FLIGHTS VALUES (1001, 'Bengaluru', 'Frankfurt', 7000, TO_DATE('2025-06-01 08:00', 'YYYY-MM-DD HH24:MI'), TO_DATE('2025-06-01 15:00', 'YYYY-MM-DD HH24:MI'), 85000);
INSERT INTO FLIGHTS VALUES (1002, 'Bengaluru', 'New Delhi', 2150, TO_DATE('2025-06-02 09:00', 'YYYY-MM-DD HH24:MI'), TO_DATE('2025-06-02 11:30', 'YYYY-MM-DD HH24:MI'), 25000);
INSERT INTO FLIGHTS VALUES (1003, 'New Delhi', 'Frankfurt', 6000, TO_DATE('2025-06-03 10:00', 'YYYY-MM-DD HH24:MI'), TO_DATE('2025-06-03 17:00', 'YYYY-MM-DD HH24:MI'), 80000);
```

INSERT INTO FLIGHTS VALUES (1004, 'Bengaluru', 'Mumbai', 980, TO_DATE('2025-06-04 14:00', 'YYYY-MM-DD HH24:MI'), TO_DATE('2025-06-04 15:30', 'YYYY-MM-DD HH24:MI'), 10000);

INSERT INTO FLIGHTS VALUES (1005, 'Mumbai', 'New Delhi', 1400, TO_DATE('2025-06-05 07:00', 'YYYY-MM-DD HH24:MI'), TO_DATE('2025-06-05 08:45', 'YYYY-MM-DD HH24:MI'), 15000);

Write the following queries in SQL.

1.Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

```
SQL> SELECT aname
  2  FROM AIRCRAFT a
  3  WHERE NOT EXISTS (
  4     SELECT 1 FROM CERTIFIED c
  5     JOIN EMPLOYEES e ON c.eid = e.eid
  6     WHERE c.aid = a.aid AND e.salary <= 80000
  7  );

ANAME
--------------------------------------------------
Boeing 777
Embraer 190
Bombardier CRJ
```

2. For each pilot who is certified for more than three aircrafts,find the eid and the maximum cruisingrange of the aircraft for which she or he is certified.

```
SELECT c.eid, MAX(a.cruisingrange) AS max_cruisingrange
  2  FROM CERTIFIED c
  3  JOIN AIRCRAFT a ON c.aid = a.aid
  4  GROUP BY c.eid
  5  HAVING COUNT(DISTINCT c.aid) > 3;

     EID MAX_CRUISINGRANGE
---------- -----------------
     103            9700
```

3.Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.

```
SELECT ename
  2  FROM EMPLOYEES
  3  WHERE salary < (
```

```
  4     SELECT MIN(price)
  5     FROM FLIGHTS
  6     WHERE fromPlace = 'Bengaluru' AND toPlace = 'Frankfurt'
  7  );
```

```
ENAME
--------------------------------------------------
Alice Brown
Carol White
David Lee
```

4.Find the names of pilots certified for some Boeing aircraft.

```
SQL> SELECT DISTINCT e.ename
  2  FROM EMPLOYEES e
  3  JOIN CERTIFIED c ON e.eid = c.eid
  4  JOIN AIRCRAFT a ON c.aid = a.aid
  5  WHERE a.aname LIKE 'Boeing%';
```

```
ENAME
--------------------------------------------------
John Smith
Bob Johnson
David Lee
```

5.Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

```
SQL> SELECT DISTINCT a.aid
  2  FROM AIRCRAFT a
  3  JOIN FLIGHTS f ON f.fromPlace = 'Bengaluru' AND f.toPlace = 'New Delhi'
  4  WHERE a.cruisingrange >= f.distance;

      AID
----------
         1
         2
         3
         4
```

3. The following tables are maintained by a Book dealer.

AUTHOR(author_id:int, name:string, city:string, country:string)

PUBLISHER(publisher_id:int, name:string, city:string,country:string)

CATALOG(book_id:int, title:string, author_id:int, publisher_id:int, category_id:int, year:int, price:int)
CATEGORY(category_id:int, description:string)

ORDER_DETAILS(order_no:int, book_id:int, quantity:int)

Write the following queries in SQL.

```
-- AUTHOR
CREATE TABLE AUTHOR (
  author_id NUMBER PRIMARY KEY,
  name VARCHAR2(50),
  city VARCHAR2(50),
  country VARCHAR2(50)
);

-- PUBLISHER
CREATE TABLE PUBLISHER (
  publisher_id NUMBER PRIMARY KEY,
  name VARCHAR2(50),
  city VARCHAR2(50),
  country VARCHAR2(50)
);

-- CATEGORY
CREATE TABLE CATEGORY (
  category_id NUMBER PRIMARY KEY,
  description VARCHAR2(100)
);

-- CATALOG
CREATE TABLE CATALOG (
```

```
  book_id NUMBER PRIMARY KEY,
  title VARCHAR2(100),
  author_id NUMBER,
  publisher_id NUMBER,
  category_id NUMBER,
  year NUMBER,
  price NUMBER,
  FOREIGN KEY (author_id) REFERENCES AUTHOR(author_id),
  FOREIGN KEY (publisher_id) REFERENCES PUBLISHER(publisher_id),
  FOREIGN KEY (category_id) REFERENCES CATEGORY(category_id)
);

-- ORDER_DETAILS
CREATE TABLE ORDER_DETAILS (
  order_no NUMBER,
  book_id NUMBER,
  quantity NUMBER,
  FOREIGN KEY (book_id) REFERENCES CATALOG(book_id)
);

-- AUTHOR
INSERT INTO AUTHOR VALUES (1, 'R.K. Narayan', 'Chennai', 'India');
INSERT INTO AUTHOR VALUES (2, 'Chetan Bhagat', 'Mumbai', 'India');
INSERT INTO AUTHOR VALUES (3, 'George Orwell', 'London', 'UK');

-- PUBLISHER
INSERT INTO PUBLISHER VALUES (101, 'Penguin', 'Mumbai', 'India');
INSERT INTO PUBLISHER VALUES (102, 'HarperCollins', 'New Delhi', 'India');
INSERT INTO PUBLISHER VALUES (103, 'Bloomsbury', 'London', 'UK');
INSERT INTO PUBLISHER VALUES (104, 'UnpublishedHouse', 'Paris', 'France'); -- no books

-- CATEGORY
INSERT INTO CATEGORY VALUES (1, 'Fiction');
INSERT INTO CATEGORY VALUES (2, 'Politics');
```

```
-- CATALOG
INSERT INTO CATALOG VALUES (201, 'Malgudi Days', 1, 101, 1, 1943, 300);
INSERT INTO CATALOG VALUES (202, 'Five Point Someone', 2, 101, 1, 2004, 250);
INSERT INTO CATALOG VALUES (203, '1984', 3, 103, 2, 1949, 400);
INSERT INTO CATALOG VALUES (204, 'Half Girlfriend', 2, 101, 1, 2014, 350);
INSERT INTO CATALOG VALUES (205, 'Animal Farm', 3, 103, 2, 1945, 300);

-- ORDER_DETAILS
INSERT INTO ORDER_DETAILS VALUES (501, 201, 5);
INSERT INTO ORDER_DETAILS VALUES (502, 202, 10);
INSERT INTO ORDER_DETAILS VALUES (503, 203, 8);
INSERT INTO ORDER_DETAILS VALUES (504, 204, 12); -- highest sales: 12
INSERT INTO ORDER_DETAILS VALUES (505, 205, 3);
```

1.Find the publisher id who have not published any books.

```
SELECT publisher_id
FROM PUBLISHER
WHERE publisher_id NOT IN (
  SELECT DISTINCT publisher_id FROM CATALOG
);
```

2.Find the author of the book which has maximum sales.

```
SELECT A.name AS author_name
FROM AUTHOR A
JOIN CATALOG C ON A.author_id = C.author_id
JOIN ORDER_DETAILS O ON C.book_id = O.book_id
WHERE O.quantity = (
  SELECT MAX(quantity) FROM ORDER_DETAILS
);
```

3.Demonstrate how you increase the price of books published by a specific publisher by 10%.

```
UPDATE CATALOG
```

```
SET price = price * 1.10
WHERE publisher_id = 101;
```

4.Find the author name and publisher name who live in the same city.

```
SELECT A.name AS author_name, P.name AS publisher_name
FROM AUTHOR A
JOIN PUBLISHER P ON A.city = P.city;
```

5.Find the publisher name who have published at least 4 books.

```
SELECT P.name
FROM PUBLISHER P
JOIN CATALOG C ON P.publisher_id = C.publisher_id
GROUP BY P.name
HAVING COUNT(*) >= 4;
```

**

```
-- Check updated prices
SELECT * FROM CATALOG;

-- Check all publishers
SELECT * FROM PUBLISHER;

-- Check sales
SELECT * FROM ORDER_DETAILS;
```

5. Consider the following Insurance database:

PERSON(driver_id,name ,address)

CAR(regno,model,year)

ACCIDENT(report_number,accd_date,location)

OWNS(driver_id,regno)

PARTICIPATED(driver_id,regno,report_number, damage_amount)

Write the following queries in SQL.

```
-- PERSON tables
CREATE TABLE PERSON (
   driver_id NUMBER PRIMARY KEY,
   name VARCHAR2(50),
   address VARCHAR2(100)
);

-- CAR table
CREATE TABLE CAR (
   regno VARCHAR2(15) PRIMARY KEY,
   model VARCHAR2(30),
   year NUMBER
);

-- ACCIDENT table
CREATE TABLE ACCIDENT (
```

```
    report_number NUMBER PRIMARY KEY,
    accd_date DATE,
    location VARCHAR2(50)
);

-- OWNS table
CREATE TABLE OWNS (
    driver_id NUMBER,
    regno VARCHAR2(15),
    PRIMARY KEY (driver_id, regno),
    FOREIGN KEY (driver_id) REFERENCES PERSON(driver_id),
    FOREIGN KEY (regno) REFERENCES CAR(regno)
);

-- PARTICIPATED table
CREATE TABLE PARTICIPATED (
    driver_id NUMBER,
    regno VARCHAR2(15),
    report_number NUMBER,
    damage_amount NUMBER,
    FOREIGN KEY (driver_id) REFERENCES PERSON(driver_id),
    FOREIGN KEY (regno) REFERENCES CAR(regno),
    FOREIGN KEY (report_number) REFERENCES ACCIDENT(report_number)
);

-- PERSON
INSERT INTO PERSON VALUES (1, 'Ravi', 'Belgaum');
INSERT INTO PERSON VALUES (2, 'Anita', 'Gokak');
INSERT INTO PERSON VALUES (3, 'Suresh', 'Hubli');
INSERT INTO PERSON VALUES (4, 'Divya', 'Belgaum');
INSERT INTO PERSON VALUES (5, 'Rahul', 'Dharwad');

-- CAR
INSERT INTO CAR VALUES ('KA01AB1234', 'Hyundai i20', 2018);
```

```
INSERT INTO CAR VALUES ('KA02CD2345', 'Maruti Swift', 2019);
INSERT INTO CAR VALUES ('KA03EF3456', 'Honda City', 2020);
INSERT INTO CAR VALUES ('KA04GH4567', 'Hyundai i20', 2021);
INSERT INTO CAR VALUES ('KA05IJ5678', 'Tata Nexon', 2020);

-- ACCIDENT
INSERT INTO ACCIDENT VALUES (1, TO_DATE('2020-03-15','YYYY-MM-DD'), 'Belgaum');
INSERT INTO ACCIDENT VALUES (2, TO_DATE('2020-08-10','YYYY-MM-DD'), 'Hubli');
INSERT INTO ACCIDENT VALUES (3, TO_DATE('2021-01-20','YYYY-MM-DD'), 'Gokak');
INSERT INTO ACCIDENT VALUES (4, TO_DATE('2020-06-25','YYYY-MM-DD'), 'Dharwad');
INSERT INTO ACCIDENT VALUES (5, TO_DATE('2022-02-12','YYYY-MM-DD'), 'Belgaum');

-- OWNS
INSERT INTO OWNS VALUES (1, 'KA01AB1234');
INSERT INTO OWNS VALUES (2, 'KA02CD2345');
INSERT INTO OWNS VALUES (3, 'KA03EF3456');
INSERT INTO OWNS VALUES (4, 'KA04GH4567');
INSERT INTO OWNS VALUES (5, 'KA05IJ5678');

-- PARTICIPATED
INSERT INTO PARTICIPATED VALUES (1, 'KA01AB1234', 1, 12000);
INSERT INTO PARTICIPATED VALUES (2, 'KA02CD2345', 2, 18000);
INSERT INTO PARTICIPATED VALUES (3, 'KA03EF3456', 3, 15000);
INSERT INTO PARTICIPATED VALUES (4, 'KA04GH4567', 4, 22000);
INSERT INTO PARTICIPATED VALUES (5, 'KA05IJ5678', 5, 30000);
```

1. Find the driver name and the model of the car which is own by them.
   ```
   SELECT P.name, C.model
   FROM PERSON P
   JOIN OWNS O ON P.driver_id = O.driver_id
   JOIN CAR C ON O.regno = C.regno;
   ```

2. Update the damage amount for the car with specific regno in the accident with report number 4 to 25000.
   ```
   UPDATE PARTICIPATED
   SET damage_amount = 25000
   WHERE regno = 'KA04GH4567' AND report_number = 4;
   ```

   **

```
SELECT * FROM PARTICIPATED
WHERE report_number = 4;
```

3. Find the total number of people who owned cars that were involved in accidents in the year 2020.
```
SELECT COUNT(DISTINCT O.driver_id) AS Total_Owners_2020
FROM PARTICIPATED P
JOIN ACCIDENT A ON P.report_number = A.report_number
JOIN OWNS O ON P.regno = O.regno
WHERE TO_CHAR(A.accd_date, 'YYYY') = '2020';
```

4. Find the number of accidents in which cars belonging to a specific model were involved.
```
SELECT COUNT(DISTINCT P.report_number) AS Total_Accidents
FROM PARTICIPATED P
JOIN CAR C ON P.regno = C.regno
WHERE C.model = 'Hyundai i20';
```

5. Find the driver id and names of the drivers who stays in either Belgaum or Gokak.
```
SELECT driver_id, name
FROM PERSON
WHERE address IN ('Belgaum', 'Gokak');
```

4. Consider the following database for a Banking enterprise:

BRANCH(branch_name:string, branch_city:string, assets:real)

ACCOUNT(accno:int, branch_name:string, balance:real)

DEPOSITOR(customer_name:string, accno:int)

CUSTOMER(customer_name:string, customer_street:string,customer_city:string)

LOAN(Loan_number:int, branch_name:string, amount:real)

BORROWER(customer_name:string, Loan_number:int)

Write the following queries in SQL.

```
CREATE TABLE BRANCH1 (
  branch_name VARCHAR2(30) PRIMARY KEY,
  branch_city VARCHAR2(30),
  assets NUMBER
);

CREATE TABLE ACCOUNT1 (
  accno NUMBER PRIMARY KEY,
  branch_name VARCHAR2(30),
  balance NUMBER,
  FOREIGN KEY (branch_name) REFERENCES BRANCH1(branch_name)
);

CREATE TABLE CUSTOMER1 (
  customer_name VARCHAR2(50) PRIMARY KEY,
  customer_street VARCHAR2(50),
```

```
  customer_city VARCHAR2(50)
);

CREATE TABLE DEPOSITOR1 (
  customer_name VARCHAR2(50),
  accno NUMBER,
  FOREIGN KEY (customer_name) REFERENCES CUSTOMER1(customer_name),
  FOREIGN KEY (accno) REFERENCES ACCOUNT1(accno)
);

CREATE TABLE LOAN1 (
  Loan_number NUMBER PRIMARY KEY,
  branch_name VARCHAR2(30),
  amount NUMBER,
  FOREIGN KEY (branch_name) REFERENCES BRANCH1(branch_name)
);

CREATE TABLE BORROWER1 (
  customer_name VARCHAR2(50),
  Loan_number NUMBER,
  FOREIGN KEY (customer_name) REFERENCES CUSTOMER1(customer_name),
  FOREIGN KEY (Loan_number) REFERENCES LOAN1(Loan_number)
);

-- BRANCHES
INSERT INTO BRANCH1 VALUES ('Main', 'Mumbai', 1000000);
INSERT INTO BRANCH1 VALUES ('South', 'Bangalore', 500000);
INSERT INTO BRANCH1 VALUES ('East', 'Kolkata', 400000);

-- CUSTOMERS
INSERT INTO CUSTOMER1 VALUES ('Amit', 'MG Road', 'Mumbai');
INSERT INTO CUSTOMER1 VALUES ('Neha', 'Church Street', 'Bangalore');
INSERT INTO CUSTOMER1 VALUES ('Ravi', 'Park Street', 'Kolkata');
```

```
-- ACCOUNTS
INSERT INTO ACCOUNT1 VALUES (101, 'Main', 8000);
INSERT INTO ACCOUNT1 VALUES (102, 'Main', 12000);
INSERT INTO ACCOUNT1 VALUES (103, 'South', 7000);
INSERT INTO ACCOUNT1 VALUES (104, 'Main', 5000);
INSERT INTO ACCOUNT1 VALUES (105, 'East', 6000);

-- DEPOSITORS
INSERT INTO DEPOSITOR1 VALUES ('Amit', 101);
INSERT INTO DEPOSITOR1 VALUES ('Amit', 102); -- 2 accounts at Main
INSERT INTO DEPOSITOR1 VALUES ('Neha', 103);
INSERT INTO DEPOSITOR1 VALUES ('Ravi', 105);

-- LOANS
INSERT INTO LOAN1 VALUES (201, 'Main', 20000);
INSERT INTO LOAN1 VALUES (202, 'South', 30000);
INSERT INTO LOAN1 VALUES (203, 'East', 15000);

-- BORROWERS
INSERT INTO BORROWER1 VALUES ('Amit', 201); -- Same branch as account (Main)
INSERT INTO BORROWER1 VALUES ('Neha', 202);
INSERT INTO BORROWER1 VALUES ('Ravi', 203);
```

1. Find all the customers who have at least two accounts at the main branch.

```
SELECT d.customer_name
FROM DEPOSITOR1 d
JOIN ACCOUNT1 a ON d.accno = a.accno
WHERE a.branch_name = 'Main'
GROUP BY d.customer_name
HAVING COUNT(*) >= 2;
```

2. Demonstrate how you delete all account tuples at every branch located in a specific city.

```
DELETE FROM ACCOUNT1
WHERE branch_name IN (
  SELECT branch_name FROM BRANCH1 WHERE branch_city = 'Bangalore'
);
```

**

```
SELECT * FROM ACCOUNT1;
```

3. Find the bank that has having highest average balance.

```
SELECT branch_name
FROM ACCOUNT1
GROUP BY branch_name
HAVING AVG(balance) = (
  SELECT MAX(AVG(balance))
  FROM ACCOUNT1
  GROUP BY branch_name
);
```

4. Find all the customers who have both account and loan at specific branch.

```
SELECT DISTINCT d.customer_name
FROM DEPOSITOR1 d
JOIN ACCOUNT1 a ON d.accno = a.accno
JOIN BORROWER1 b ON d.customer_name = b.customer_name
JOIN LOAN1 l ON b.Loan_number = l.Loan_number
WHERE a.branch_name = 'Main' AND l.branch_name = 'Main';
```

6.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:

```
-- SALESMAN table
CREATE TABLE SALESMAN (
   Salesman_id NUMBER PRIMARY KEY,
   Name VARCHAR2(50),
   City VARCHAR2(50),
```

```
    Commission NUMBER(5,2)
);

-- CUSTOMER table
CREATE TABLE CUSTOMER (
    Customer_id NUMBER PRIMARY KEY,
    Cust_Name VARCHAR2(50),
    City VARCHAR2(50),
    Grade NUMBER,
    Salesman_id NUMBER,
    FOREIGN KEY (Salesman_id) REFERENCES SALESMAN(Salesman_id)
        ON DELETE CASCADE
);

-- ORDERS table
CREATE TABLE ORDERS (
    Ord_No NUMBER PRIMARY KEY,
    Purchase_Amt NUMBER(10,2),
    Ord_Date DATE,
    Customer_id NUMBER,
    Salesman_id NUMBER,
    FOREIGN KEY (Customer_id) REFERENCES CUSTOMER(Customer_id),
    FOREIGN KEY (Salesman_id) REFERENCES SALESMAN(Salesman_id)
        ON DELETE CASCADE
);

-- SALESMAN
INSERT INTO SALESMAN VALUES (1000, 'Ravi Kumar', 'Bangalore', 0.15);
INSERT INTO SALESMAN VALUES (1001, 'Anita Sharma', 'Delhi', 0.12);
INSERT INTO SALESMAN VALUES (1002, 'Suresh Mehta', 'Mumbai', 0.10);
INSERT INTO SALESMAN VALUES (1003, 'Rahul Joshi', 'Chennai', 0.13);
INSERT INTO SALESMAN VALUES (1004, 'Pooja Patel', 'Bangalore', 0.11);

-- CUSTOMER
```

```
INSERT INTO CUSTOMER VALUES (201, 'Akash', 'Bangalore', 200, 1000);
INSERT INTO CUSTOMER VALUES (202, 'Neha', 'Delhi', 300, 1001);
INSERT INTO CUSTOMER VALUES (203, 'Karan', 'Mumbai', 250, 1002);
INSERT INTO CUSTOMER VALUES (204, 'Divya', 'Chennai', 400, 1001);
INSERT INTO CUSTOMER VALUES (205, 'Tarun', 'Bangalore', 150, 1000);

-- ORDERS
INSERT INTO ORDERS VALUES (301, 4500.00, TO_DATE('2024-12-01', 'YYYY-MM-DD'), 201, 1000);
INSERT INTO ORDERS VALUES (302, 3800.00, TO_DATE('2024-12-05', 'YYYY-MM-DD'), 202, 1001);
INSERT INTO ORDERS VALUES (303, 6200.00, TO_DATE('2024-12-10', 'YYYY-MM-DD'), 204, 1001);
INSERT INTO ORDERS VALUES (304, 2000.00, TO_DATE('2024-12-15', 'YYYY-MM-DD'), 205, 1000);
INSERT INTO ORDERS VALUES (305, 5100.00, TO_DATE('2024-12-20', 'YYYY-MM-DD'), 203, 1002);
```

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

```
SELECT COUNT(*) AS Customers_Above_Bangalore_Avg
FROM CUSTOMER
WHERE Grade > (
   SELECT AVG(Grade)
   FROM CUSTOMER
   WHERE City = 'Bangalore'
);
```

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

```
SELECT S.Name, S.Salesman_id
FROM SALESMAN S
JOIN CUSTOMER C ON S.Salesman_id = C.Salesman_id
GROUP BY S.Name, S.Salesman_id
HAVING COUNT(C.Customer_id) > 1;
```

3. List all salesmen names and customer names for whom order amount is more than 4000.

```
SELECT S.Name AS Salesman_Name, C.Cust_Name AS Customer_Name
FROM ORDERS O
JOIN SALESMAN S ON O.Salesman_id = S.Salesman_id
JOIN CUSTOMER C ON O.Customer_id = C.Customer_id
WHERE O.Purchase_Amt > 4000;
```

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

```
DELETE FROM SALESMAN
WHERE Salesman_id = 1000;
**
-- Check if salesman 1000 is deleted
SELECT * FROM SALESMAN WHERE Salesman_id = 1000;

-- Check if related orders are deleted
SELECT * FROM ORDERS WHERE Salesman_id = 1000;
```

q. 7 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:

```
CREATE TABLE ACTOR (
   Act_id NUMBER PRIMARY KEY,
   Act_Name VARCHAR2(50),
```

```
    Act_Gender CHAR(1)
);

CREATE TABLE DIRECTOR (
    Dir_id NUMBER PRIMARY KEY,
    Dir_Name VARCHAR2(50),
    Dir_Phone VARCHAR2(15)
);

CREATE TABLE MOVIES (
    Mov_id NUMBER PRIMARY KEY,
    Mov_Title VARCHAR2(100),
    Mov_Year NUMBER(4),
    Mov_Lang VARCHAR2(20),
    Dir_id NUMBER,
    FOREIGN KEY (Dir_id) REFERENCES DIRECTOR(Dir_id)
);

CREATE TABLE MOVIE_CAST (
    Act_id NUMBER,
    Mov_id NUMBER,
    Role VARCHAR2(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 NUMBER,
    Rev_Stars NUMBER(1),
    FOREIGN KEY (Mov_id) REFERENCES MOVIES(Mov_id)
);

-- ACTOR data
```

```
INSERT INTO ACTOR VALUES (1, 'Ranveer Singh', 'M');
INSERT INTO ACTOR VALUES (2, 'Deepika Padukone', 'F');
INSERT INTO ACTOR VALUES (3, 'Ajay Devgn', 'M');
INSERT INTO ACTOR VALUES (4, 'Alia Bhatt', 'F');
INSERT INTO ACTOR VALUES (5, 'Amitabh Bachchan', 'M');

-- DIRECTOR data
INSERT INTO DIRECTOR VALUES (101, 'Sanjay Leela Bansali', '9876543210');
INSERT INTO DIRECTOR VALUES (102, 'Ram Gopal Verma', '9988776655');
INSERT INTO DIRECTOR VALUES (103, 'Rohit Shetty', '9012345678');
INSERT INTO DIRECTOR VALUES (104, 'Karan Johar', '9123456780');
INSERT INTO DIRECTOR VALUES (105, 'Zoya Akhtar', '9234567812');

-- MOVIES data
INSERT INTO MOVIES VALUES (201, 'Padmaavat', 2018, 'Hindi', 101);
INSERT INTO MOVIES VALUES (202, 'Gangubai Kathiawadi', 2022, 'Hindi', 101);
INSERT INTO MOVIES VALUES (203, 'Company', 2002, 'Hindi', 102);
INSERT INTO MOVIES VALUES (204, 'Shiva', 1990, 'Telugu', 102);
INSERT INTO MOVIES VALUES (205, 'Singham', 2011, 'Hindi', 103);

-- MOVIE_CAST data
INSERT INTO MOVIE_CAST VALUES (1, 201, 'Lead');
INSERT INTO MOVIE_CAST VALUES (2, 201, 'Lead');
INSERT INTO MOVIE_CAST VALUES (2, 202, 'Lead');
INSERT INTO MOVIE_CAST VALUES (3, 203, 'Villain');
INSERT INTO MOVIE_CAST VALUES (3, 205, 'Police');

-- RATING data
INSERT INTO RATING VALUES (201, 4);
INSERT INTO RATING VALUES (201, 5);
INSERT INTO RATING VALUES (202, 3);
INSERT INTO RATING VALUES (203, 5);
INSERT INTO RATING VALUES (204, 4);
```

1. List the titles of all movies directed by ‘Sanjay Leela Bansali’.

```
SELECT Mov_Title
FROM MOVIES M
JOIN DIRECTOR D ON M.Dir_id = D.Dir_id
WHERE D.Dir_Name = 'Sanjay Leela Bansali';
```

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

```
SELECT DISTINCT M.Mov_Title
FROM MOVIES M
JOIN MOVIE_CAST MC ON M.Mov_id = MC.Mov_id
WHERE MC.Act_id IN (
   SELECT Act_id
   FROM MOVIE_CAST
   GROUP BY Act_id
   HAVING COUNT(DISTINCT Mov_id) >= 2
);
```

3. 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 M.Mov_Title, R.Rev_Stars,
      (SELECT MAX(Rev_Stars) FROM RATING R2 WHERE R2.Mov_id = M.Mov_id) AS Max_Stars
FROM MOVIES M
JOIN RATING R ON M.Mov_id = R.Mov_id
ORDER BY M.Mov_Title;
```

4. Update rating of all movies directed by ‘Ram Gopal Verma’ 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 = 'Ram Gopal Verma'
```

```
    )
);
```

**

```
SELECT M.Mov_Title, R.Rev_Stars
FROM RATING R
JOIN MOVIES M ON R.Mov_id = M.Mov_id
WHERE M.Dir_id = (
    SELECT Dir_id FROM DIRECTOR WHERE Dir_Name = 'Ram Gopal Verma'
);
```