s1:
Consider following Relation
Account (Acc_no, branch_name,balance)
Branch(branch_name,branch_city,assets)
Customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)
Create above tables with appropriate constraints like primary key, foreign key, not null etc.

1. Find the names of all branches in loan relation.

```
SELECT DISTINCT branch_name
FROM Loan;
```

2. Find all loan numbers for loans made at ‘Wadia College’ Branch with loan amount > 12000.

```
SELECT loan_no
FROM Loan
WHERE branch_name = 'Wadia College'
AND amount > 12000;
```

3. Find all customers who have a loan from bank. Find their names,loan_no and loan amount.

```
SELECT Customer.cust_name, Loan.loan_no, Loan.amount
FROM Borrower
JOIN Customer ON Borrower.cust_name = Customer.cust_name
JOIN Loan ON Borrower.loan_no = Loan.loan_no;
```

4. List all customers in alphabetical order who have loan from ‘Wadia College’ branch.

```
SELECT Customer.cust_name
FROM Borrower
JOIN Customer ON Borrower.cust_name = Customer.cust_name
JOIN Loan ON Borrower.loan_no = Loan.loan_no
WHERE Loan.branch_name = 'Wadia College'
ORDER BY Customer.cust_name ASC;
```

5. Display distinct cities of branch.

```
SELECT DISTINCT branch_city
FROM Branch;
```

---

s2:
Consider following Relation
Account (Acc_no, branch_name,balance)
Branch(branch_name,branch_city,assets)
Customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)
Create above tables with appropriate constraints like primary key, foreign key, not null etc.

1. Find all customers who have both account and loan at bank.

```
SELECT DISTINCT c.cust_name
FROM Customer c
JOIN Depositor d ON c.cust_name = d.cust_name
JOIN Borrower b ON c.cust_name = b.cust_name;
```

2. Find all customers who have an account or loan or both at bank.

```
SELECT DISTINCT c.cust_name
FROM Customer c
LEFT JOIN Depositor d ON c.cust_name = d.cust_name
LEFT JOIN Borrower b ON c.cust_name = b.cust_name
```

```
WHERE d.cust_name IS NOT NULL OR b.cust_name IS NOT NULL;

3. Find all customers who have account but no loan at the bank.
SELECT DISTINCT c.cust_name
FROM Customer c
JOIN Depositor d ON c.cust_name = d.cust_name
LEFT JOIN Borrower b ON c.cust_name = b.cust_name
WHERE b.cust_name IS NULL;

4. Find average account balance at ‘Wadia College’ branch.
SELECT AVG(balance) AS avg_balance
FROM Account
WHERE branch_name = 'Wadia College';

5. Find no. of depositors at each branch
SELECT a.branch_name, COUNT(DISTINCT d.cust_name) AS no_of_depositors
FROM Account a
JOIN Depositor d ON a.Acc_no = d.acc_no
GROUP BY a.branch_name;
```

---

s3:

Consider following Relation

Account (Acc_no, branch_name,balance)
Branch(branch_name,branch_city,assets)
Customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)

Create above tables with appropriate constraints like primary key, foreign key, not null etc.

```
1. Find the branches where average account balance > 15000.
SELECT branch_name
FROM Account
GROUP BY branch_name
HAVING AVG(balance) > 15000;

2. Find number of tuples in customer relation.
SELECT COUNT(*) AS customer_count
FROM Customer;

3. Calculate total loan amount given by bank.
SELECT SUM(amount) AS total_loan_amount
FROM Loan;

4. Delete all loans with loan amount between 1300 and 1500.
DELETE FROM Loan
WHERE amount BETWEEN 1300 AND 1500;

5. Find the average account balance at each branch
SELECT branch_name, AVG(balance) AS avg_balance
FROM Account
GROUP BY branch_name;

6. Find name of Customer and city where customer name starts with
Letter P.
SELECT cust_name, cust_city
FROM Customer
WHERE cust_name LIKE 'P%';
```

---

s4:

Create following tables with suitable constraints (primary key, foreign key, not null etc).
Insert record and solve the following queries:

```
Create table Cust_Master(Cust_no, Cust_name, Cust_addr)
Create table Order(Order_no, Cust_no, Order_date, Qty_Ordered)
Create Product (Product_no, Product_name, Order_no)

1. List names of customers having 'A' as second letter in their
name.
SELECT Cust_name
FROM Cust_Master
WHERE Cust_name LIKE '_A%';

2. Display order from Customer no C1002, C1005, C1007 and C1008
SELECT *
FROM Order
WHERE Cust_no IN ('C1002', 'C1005', 'C1007', 'C1008');

3. List Clients who stay in either 'Banglore or 'Manglore'
SELECT Cust_name
FROM Cust_Master
WHERE Cust_addr IN ('Banglore', 'Manglore');

4. Display name of customers& the product_name they have purchase
SELECT cm.Cust_name, p.Product_name
FROM Cust_Master cm
JOIN Order o ON cm.Cust_no = o.Cust_no
JOIN Product p ON o.Order_no = p.Order_no;

5. Create view View1 consisting of Cust_name, Product_name.
CREATE VIEW View1 AS
SELECT cm.Cust_name, p.Product_name
FROM Cust_Master cm
JOIN Order o ON cm.Cust_no = o.Cust_no
JOIN Product p ON o.Order_no = p.Order_no;

6. Disply product_name and quantity purchase by each customer
SELECT cm.Cust_name, p.Product_name, o.Qty_Ordered
FROM Cust_Master cm
JOIN Order o ON cm.Cust_no = o.Cust_no
JOIN Product p ON o.Order_no = p.Order_no;

7. Perform different joint operation.
SELECT cm.Cust_name, p.Product_name
FROM Cust_Master cm
INNER JOIN Order o ON cm.Cust_no = o.Cust_no
INNER JOIN Product p ON o.Order_no = p.Order_no;

SELECT cm.Cust_name, p.Product_name
FROM Cust_Master cm
LEFT JOIN Order o ON cm.Cust_no = o.Cust_no
LEFT JOIN Product p ON o.Order_no = p.Order_no;

SELECT cm.Cust_name, p.Product_name
FROM Cust_Master cm
RIGHT JOIN Order o ON cm.Cust_no = o.Cust_no
RIGHT JOIN Product p ON o.Order_no = p.Order_no;

SELECT cm.Cust_name, p.Product_name
FROM Cust_Master cm
LEFT JOIN Order o ON cm.Cust_no = o.Cust_no
LEFT JOIN Product p ON o.Order_no = p.Order_no
UNION
SELECT cm.Cust_name, p.Product_name
FROM Cust_Master cm
RIGHT JOIN Order o ON cm.Cust_no = o.Cust_no
RIGHT JOIN Product p ON o.Order_no = p.Order_no;
```

```
-------------------------------------------------------
s5:
Consider following Relation
Employee(emp_id,employee_name,street,city)
Works(employee_name,company_name,salary)
Company(company_name,city)
Manages(employee_name,manager_name)
Create above tables with appropriate constraints like primary key,
foreign key, not null etc.

1. Find the names of all employees who work for ‘TCS’.
SELECT employee_name
FROM Works
WHERE company_name = 'TCS';

2. Find the names and company names of all employees sorted in
ascending order of company name and descending order of employee
names of that company.
SELECT employee_name, company_name
FROM Works
ORDER BY company_name ASC, employee_name DESC;

3. Change the city of employee working with InfoSys to ‘Bangalore’
UPDATE Employee
SET city = 'Bangalore'
WHERE employee_name IN (
    SELECT employee_name
    FROM Works
    WHERE company_name = 'InfoSys'
);

4. Find the names, street address, and cities of residence for all
employees who work for 'TechM' and earn more than $10,000.
SELECT e.employee_name, e.street, e.city
FROM Employee e
JOIN Works w ON e.employee_name = w.employee_name
WHERE w.company_name = 'TechM' AND w.salary > 10000;

5. Add Column Asset to Company table.
ALTER TABLE Company ADD Asset DECIMAL(15, 2);
-------------------------------------------------------
s6:
Consider following Relation
Employee(emp_id,employee_name,street,city)
Works(employee_name,company_name,salary)
Company(company_name,city)
Manages(employee_name,manager_name)
Create above tables with appropriate constraints like primary key,
foreign key, not null etc.

1. Change the city of employee working with InfoSys to ‘Bangalore’
UPDATE Employee
SET city = 'Bangalore'
WHERE employee_name IN (SELECT employee_name FROM Works WHERE company_name =
'InfoSys');

2. Find the names of all employees who earn more than the average
salary of all employees of their company. Assume that all people
work for at most one company.
SELECT employee_name
FROM Works AS W1
WHERE salary > (SELECT AVG(salary) FROM Works AS W2 WHERE W1.company_name =
W2.company_name);
```

3. Find the names, street address, and cities of residence for all employees who work for 'TechM' and earn more than $10,000.

```
SELECT E.employee_name, E.street, E.city
FROM Employee AS E
JOIN Works AS W ON E.employee_name = W.employee_name
WHERE W.company_name = 'TechM' AND W.salary > 10000;
```

4. Change name of table Manages to Management.

```
RENAME TABLE Manages TO Management;
```

5. Create Simple and Unique index on employee table.

```
CREATE INDEX idx_emp_id ON Employee(emp_id);
CREATE UNIQUE INDEX idx_employee_name_unique ON Employee(employee_name);
```

6. Display index Information

```
SHOW INDEX FROM Employee;
```

---

s7:

Consider following Relation

Account (Acc_no, branch_name,balance)
Branch(branch_name,branch_city,assets)
Customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)

Execute the following query:

1. Create a View1 to display List all customers in alphabetical order who have loan from Pune_Station branch

```
CREATE VIEW View1 AS
SELECT DISTINCT B.cust_name
FROM Borrower B
JOIN Loan L ON B.loan_no = L.loan_no
WHERE L.branch_name = 'Pune_Station'
ORDER BY B.cust_name;
```

2. Create View2 on branch table by selecting any two columns and perform insert update delete operations.

```
CREATE VIEW View2 AS
SELECT branch_name, branch_city
FROM Branch;
```

3. Create View3 on borrower and depositor table by selecting any one column from each table perform insert update delete operations.

```
CREATE VIEW View3 AS
SELECT D.cust_name, B.loan_no
FROM Depositor D
JOIN Borrower B ON D.cust_name = B.cust_name;
```

4. Create Union of left and right joint for all customers who have an account or loan or both at bank

```
SELECT D.cust_name
FROM Depositor D
LEFT JOIN Borrower B ON D.cust_name = B.cust_name
UNION
SELECT B.cust_name
FROM Borrower B
RIGHT JOIN Depositor D ON B.cust_name = D.cust_name;
```

5. Create Simple and Unique index.

```
CREATE INDEX idx_cust_name ON Customer(cust_name);
CREATE UNIQUE INDEX idx_acc_no ON Account(acc_no);
```

6. Display index Information.

```
SHOW INDEX FROM Customer;
```

---

s8:
Consider following Relation:
 Companies (comp_id, name, cost, year)
 Orders (comp_id, domain, quantity)
Execute the following query:

1. Find names, costs, domains and quantities for companies using
inner join.

```
SELECT C.name, C.cost, O.domain, O.quantity
FROM Companies C
INNER JOIN Orders O ON C.comp_id = O.comp_id;
```

2. Find names, costs, domains and quantities for companies using
left outer join.

```
SELECT C.name, C.cost, O.domain, O.quantity
FROM Companies C
LEFT OUTER JOIN Orders O ON C.comp_id = O.comp_id;
```

3. Find names, costs, domains and quantities for companies using
right outer join.

```
SELECT C.name, C.cost, O.domain, O.quantity
FROM Companies C
RIGHT OUTER JOIN Orders O ON C.comp_id = O.comp_id;
```

4. Find names, costs, domains and quantities for companies using
Union operator.

```
SELECT C.name, C.cost, O.domain, O.quantity
FROM Companies C
JOIN Orders O ON C.comp_id = O.comp_id
UNION
SELECT C.name, C.cost, NULL AS domain, NULL AS quantity
FROM Companies C
WHERE NOT EXISTS (SELECT 1 FROM Orders O WHERE O.comp_id = C.comp_id);
OR
SELECT C.name, C.cost, O.domain, O.quantity
FROM Companies C
LEFT JOIN Orders O ON C.comp_id = O.comp_id
UNION
SELECT C.name, C.cost, NULL AS domain, NULL AS quantity
FROM Companies C RIGHT JOIN Orders O ON C.comp_id = O.comp_id;
```

5. Create View View1 by selecting both tables to show company name
and quantities.

```
CREATE VIEW View1 AS
SELECT C.name, O.quantity
FROM Companies C
INNER JOIN Orders O ON C.comp_id = O.comp_id;
```

6. Create View View2 by selecting any two columns and perform
insert update delete operations.

```
CREATE VIEW View2 AS
SELECT C.name, O.quantity
FROM Companies C
LEFT JOIN Orders O ON C.comp_id = O.comp_id;
```

insert:

```
INSERT INTO View2 (name, quantity)
VALUES ('NewCompany', 100);
```

update:

```
UPDATE View2
SET quantity = 150
WHERE name = 'NewCompany';
```

```
delete: DELETE FROM View2
WHERE name = 'NewCompany';

7. Display content of View1, View2.
-- Display content of View1
SELECT * FROM View1;

-- Display content of View2
SELECT * FROM View2;
-----------------------------------------------------
s9:
Create following tables with suitable constraints. Insert data and
solve the following queries:
CUSTOMERS(CNo, Cname, Ccity, CMobile)
ITEMS(INo, Iname, Itype, Iprice, Icount)
PURCHASE(PNo, Pdate, Pquantity, Cno, INo)

CREATE TABLE CUSTOMERS (
    CNo INT PRIMARY KEY,
    Cname VARCHAR(50) NOT NULL,
    Ccity VARCHAR(50),
    CMobile VARCHAR(15) UNIQUE
);

CREATE TABLE ITEMS (
    INo INT PRIMARY KEY,
    Iname VARCHAR(50) NOT NULL,
    Itype VARCHAR(50) CHECK (Itype IN ('Stationary', 'Electronic', 'Other')),
    Iprice DECIMAL(10, 2) CHECK (Iprice > 0),
    Icount INT CHECK (Icount >= 0)
);

CREATE TABLE PURCHASE (
    PNo INT PRIMARY KEY,
    Pdate DATE NOT NULL,
    Pquantity INT CHECK (Pquantity > 0),
    CNo INT,
    INo INT,
    FOREIGN KEY (CNo) REFERENCES CUSTOMERS(CNo),
    FOREIGN KEY (INo) REFERENCES ITEMS(INo)
);

--insert
1. List all stationary items with price between 400/- to 1000/-
SELECT * FROM ITEMS
WHERE Itype = 'Stationary' AND Iprice BETWEEN 400 AND 1000;

2. Change the mobile number of customer “Gopal”
UPDATE CUSTOMERS
SET CMobile = '9876501234'
WHERE Cname = 'Gopal';

3. Display the item with maximum price
SELECT * FROM ITEMS
WHERE Iprice = (SELECT MAX(Iprice) FROM ITEMS);

4. Display all purchases sorted from the most recent to the oldest
SELECT * FROM PURCHASE
ORDER BY Pdate DESC;

5. Count the number of customers in every city
SELECT Ccity, COUNT(*) AS CustomerCount
FROM CUSTOMERS
```

```
GROUP BY Ccity;

6. Display all purchased quantity of Customer Maya
SELECT Pquantity
FROM PURCHASE
JOIN CUSTOMERS ON PURCHASE.CNo = CUSTOMERS.CNo
WHERE CUSTOMERS.Cname = 'Maya';

7. Create view which shows Iname, Price, and Count of all stationary items in
descending order of price
CREATE VIEW StationaryItemsView AS
SELECT Iname, Iprice, Icount
FROM ITEMS
WHERE Itype = 'Stationary'
ORDER BY Iprice DESC;
```