1. Consider the following bank database relations, where the primary keys are underlined.

Branch (branch-name, branch-city, assets)

Customer (customer-name, customer-street, customer-city)

Loan (loan-number, branch-name, amount)

Borrower (customer-name, loan-number)

Account (account-number, branch-name, balance)

Depositor (customer-name, account number)

Write down the SQL expressions for the following queries.

- **i.** Find all customers who have an account but no loan in the bank.
- **ii.** Delete all loan amount between 10000/- and 25000/-
- **iii.** Find the names of all customers who have a loan at Perryridge branch.
- **iv.** Delete all loans with amounts in the range 0 to 500.

DDL:

```
CREATE DATABASE IF NOT EXISTS q1_bank_database;
USE q1_bank_database;
CREATE TABLE Branch (
   branch_name VARCHAR(50),
   branch_city VARCHAR(50),
   assets DECIMAL(12, 2),
   PRIMARY KEY (branch_name)
);
CREATE TABLE Customer (
   customer_name VARCHAR(50),
   customer_street VARCHAR(100),
   customer_city VARCHAR(50),
   PRIMARY KEY (customer_name)
);

CREATE TABLE Loan (
   loan_number INT,
   branch_name VARCHAR(50),
   amount DECIMAL(12, 2),
   PRIMARY KEY (loan_number),
   FOREIGN KEY (branch_name) REFERENCES Branch(branch_name)
);
CREATE TABLE Borrower (
   customer_name VARCHAR(50),
   loan_number INT,
   PRIMARY KEY (customer_name, loan_number),
   FOREIGN KEY (customer_name) REFERENCES Customer(customer_name),
   FOREIGN KEY (loan_number) REFERENCES Loan(loan_number)
);
CREATE TABLE Account (
   account_number INT,
   branch_name VARCHAR(50),
   balance DECIMAL(12, 2),
   PRIMARY KEY (account_number),
   FOREIGN KEY (branch_name) REFERENCES Branch(branch_name)
);
CREATE TABLE Depositor (
   customer_name VARCHAR(50),
   account_number INT,
   PRIMARY KEY (customer_name, account_number),
   FOREIGN KEY (customer_name) REFERENCES Customer(customer_name),
   FOREIGN KEY (account_number) REFERENCES Account(account_number)
);
```

DML:

```
INSERT INTO Branch (branch_name, branch_city, assets)
VALUES ('Perryridge', 'Brooklyn', 150000),
            ('Brighton', 'New York', 200000),
            ('Downtown', 'Brooklyn', 100000);
INSERT INTO Customer (customer_name, customer_street, customer_city)
VALUES ('Alice', '123 Main St', 'Brooklyn'),
            ('Bob', '456 Elm St', 'New York'),
            ('Charlie', '789 Oak St', 'Brooklyn'),
            ('David', '101 Pine St', 'Brooklyn');
```

```
INSERT INTO Loan (loan_number, branch_name, amount)
VALUES (1, 'Perryridge', 20000),
       (2, 'Brighton', 30000),
       (3, 'Downtown', 15000),
       (4, 'Perryridge', 1000);
INSERT INTO Borrower (customer_name, loan_number)
VALUES ('Alice', 1),
       ('Bob', 2),
       ('Charlie', 3),
       ('David', 4);
INSERT INTO Account (account_number, branch_name, balance)
VALUES (101, 'Perryridge', 5000),
       (102, 'Brighton', 10000),
       (103, 'Downtown', 7500),
       (104, 'Perryridge', 200);
INSERT INTO Depositor (customer_name, account_number)
VALUES ('Alice', 101),
       ('Bob', 102),
       ('Charlie', 103),
       ('David', 104);
```

Write down the SQL expressions for the following queries.

i. Find all customers who have an account but no loan in the bank.

```
SELECT DISTINCT D.customer_name
FROM Depositor D
LEFT JOIN Borrower B ON D.customer_name = B.customer_name
WHERE B.customer_name IS NULL;
```

ii. Delete all loan amount between 10000/- and 25000/-

```
DELETE FROM Loan
WHERE amount BETWEEN 10000 AND 25000;
```

iii. Find the names of all customers who have a loan at Perryridge branch.

```
SELECT DISTINCT B.customer_name
FROM Borrower B
JOIN Loan L ON B.loan_number = L.loan_number
WHERE L.branch_name = 'Perryridge';
```

iv. Delete all loans with amounts in the range 0 to 500.

```
DELETE FROM Loan
WHERE amount BETWEEN 0 AND 500;
```

2. Consider the employee database consisting of the following relations, where the primary keys are underlined.

Employee (<u>employee-id</u>, employee-name, street, city)
Works (<u>employee-id</u>, company-name, salary)
Company (<u>company-name</u>, city)
Manager (<u>employee-id</u>, manager-name)

Write down the SQL expressions for the following queries:

i. Find the company that has the most employees.
ii. Find the average salaries at each company.
iii. Find all employees who live in Barisal city, but their company is not in Barisal.
v. Find the names of all employees who work for First Bank Corporation.

DDL:

```
CREATE DATABASE IF NOT EXISTS q2_company_database;
USE q2_company_database;

CREATE TABLE Employee (
  employee_id INT PRIMARY KEY,
  employee_name VARCHAR(100),
  street VARCHAR(100),
  city VARCHAR(50)
);
CREATE TABLE Works (
  employee_id INT,
  company_name VARCHAR(100),
  salary DECIMAL(10, 2),
  FOREIGN KEY (employee_id) REFERENCES Employee(employee_id)
);
CREATE TABLE Company (
  company_name VARCHAR(100) PRIMARY KEY,
  city VARCHAR(50)
);
CREATE TABLE Manager (
    employee_id INT PRIMARY KEY,
    manager_name VARCHAR(100)
);
```

DML:

```
INSERT INTO Employee (employee_id, employee_name, street, city)
VALUES (1, 'John Doe', '123 Main St', 'New York'),
    (2, 'Jane Smith', '456 Elm St', 'Los Angeles'),
    (3, 'Alice Johnson', '789 Oak St', 'Chicago'),
    (4, 'Bob Brown', '101 Pine St', 'Barisal');
INSERT INTO Works (employee_id, company_name, salary)
VALUES (1, 'ABC Corporation', 60000.00),
    (2, 'XYZ Company', 70000.00),
    (3, 'ABC Corporation', 80000.00),
    (4, 'First Bank Corporation', 65000.00);
INSERT INTO Company (company_name, city)
VALUES ('ABC Corporation', 'New York'),
    ('XYZ Company', 'Los Angeles'),
    ('First Bank Corporation', 'Chicago');
INSERT INTO Manager (employee_id, manager_name)
VALUES (1, 'Michael Johnson'),
    (2, 'Emily Davis'),
    (3, 'Michael Johnson'),
    (4, 'Alice Johnson');
```

Write down the SQL expressions for the following queries:

i. Find the company that has the most employees.

```
SELECT company_name
FROM Works
GROUP BY company_name
ORDER BY COUNT(employee_id) DESC
LIMIT 1;
```

ii. Find the average salaries at each company.

```
 SELECT company_name, AVG(salary) AS average_salary
 FROM Works
GROUP BY company_name;
```

iii. Find all employees who live in Barisal city, but their company is not in Barisal.

```
SELECT e.employee_name
FROM Employee e
JOIN Works w ON e.employee_id = w.employee_id
JOIN Company c ON w.company_name = c.company_name
WHERE e.city = 'Barisal' AND c.city != 'Barisal';
```

iv. Find the names of all employees who work for First Bank Corporation.

```
SELECT employee_name
FROM Employee
JOIN Works ON Employee.employee_id = Works.employee_id
WHERE company_name = 'First Bank Corporation';
```

3. Consider the banking database consisting of the banking database consisting of the following tables, where the primary keys are underlined.

Branch (<u>branch-name</u>, branch-city, assets)

Customer (<u>customer-name</u>, customer-street, customer-city)

Loan-account (<u>loan-number</u>, branch-name, amount)

Borrower (<u>customer-name</u>, loan-number)

Saving-account (<u>account number</u>, branch-name, balance)

Depositor (<u>customer-name</u>, <u>account number</u>)

Write down the SQL expressions for the following queries:

i. Find all customers of the bank who have both loan and a saving account.
ii. Find all average account balance at each branch.

iii. Deduct 3% service charge from saving account balance that have both loan and a saving account otherwise deduct 5% service charge from saving account balance.

DDL:

```
CREATE DATABASE IF NOT EXISTS q3_bank_database;
USE q3_bank_database;


CREATE TABLE Branch (
  branch_name VARCHAR(50) PRIMARY KEY,
  branch_city VARCHAR(50),
  assets DECIMAL(12, 2)
);
CREATE TABLE Customer (
  customer_name VARCHAR(50) PRIMARY KEY,
  customer_street VARCHAR(100),
  customer_city VARCHAR(50)
);
CREATE TABLE Loan_account (
  loan_number INT PRIMARY KEY,
  branch_name VARCHAR(50),
  amount DECIMAL(12, 2),
  FOREIGN KEY (branch_name) REFERENCES Branch(branch_name)
);
CREATE TABLE Borrower (
  customer_name VARCHAR(50),
  loan_number INT,
  PRIMARY KEY (customer_name, loan_number),
  FOREIGN KEY (customer_name) REFERENCES Customer(customer_name),
  FOREIGN KEY (loan_number) REFERENCES Loan_account(loan_number)
);
CREATE TABLE Saving_account (
  account_number INT PRIMARY KEY,
  branch_name VARCHAR(50),
  balance DECIMAL(12, 2),
  FOREIGN KEY (branch_name) REFERENCES Branch(branch_name)
);
CREATE TABLE Depositor (
  customer_name VARCHAR(50),
  account_number INT,
  PRIMARY KEY (customer_name, account_number),
  FOREIGN KEY (customer_name) REFERENCES Customer(customer_name),
  FOREIGN KEY (account_number) REFERENCES Saving_account(account_number)
 );
```

DML:

```
INSERT INTO Branch (branch_name, branch_city, assets)
VALUES ('Main Branch', 'New York', 1000000.00),
       ('Downtown Branch', 'New York', 750000.00),
       ('Uptown Branch', 'New York', 500000.00),
       ('Westside Branch', 'Los Angeles', 900000.00),
       ('Eastside Branch', 'Los Angeles', 600000.00);
INSERT INTO Customer (customer_name, customer_street, customer_city)
VALUES ('John Smith', '123 Main St', 'New York'),
       ('Jane Doe', '456 Elm St', 'Los Angeles'),
       ('Alice Johnson', '789 Oak St', 'New York'),
       ('Bob Williams', '101 Pine St', 'Los Angeles');
INSERT INTO Loan_account (loan_number, branch_name, amount)
VALUES (1, 'Main Branch', 50000.00),
       (2, 'Downtown Branch', 30000.00),
       (3, 'Uptown Branch', 20000.00),
       (4, 'Westside Branch', 40000.00),
       (5, 'Eastside Branch', 25000.00);
INSERT INTO Borrower (customer_name, loan_number)
VALUES ('John Smith', 1),
       ('Jane Doe', 2),
       ('Alice Johnson', 3),
       ('Bob Williams', 4),
       ('Alice Johnson', 5);
INSERT INTO Saving_account (account_number, branch_name, balance)
VALUES (101, 'Main Branch', 10000.00),
       (102, 'Downtown Branch', 15000.00),
       (103, 'Uptown Branch', 20000.00),
       (104, 'Westside Branch', 30000.00),
       (105, 'Eastside Branch', 5000.00);
INSERT INTO Depositor (customer_name, account_number)
VALUES ('John Smith', 101),
       ('Jane Doe', 102),
       ('Alice Johnson', 103),
       ('Bob Williams', 104),
       ('Alice Johnson', 105);
```

Write down the SQL expressions for the following queries:

i. Find all customers of the bank who have both loan and a saving account.

```
SELECT DISTINCT c.customer_name
FROM Customer c
INNER JOIN Borrower b ON c.customer_name = b.customer_name
INNER JOIN Loan_account la ON b.loan_number = la.loan_number
INNER JOIN Depositor d ON c.customer_name = d.customer_name
INNER JOIN Saving_account sa ON d.account_number = sa.account_number;
```

ii. Find all average account balance at each branch.

```
SELECT branch_name, AVG(balance) AS average_balance
FROM Saving_account
GROUP BY branch_name;
```

iii. Deduct 3% service charge from saving account balance that have both loan and a saving account otherwise deduct 5% service charge from saving account balance.

```
UPDATE Saving_account
SET balance = CASE
  WHEN EXISTS (
    SELECT 1
    FROM Depositor d
    INNER JOIN Borrower b ON d.customer_name = b.customer_name
    WHERE d.account_number = Saving_account.account_number
  ) THEN balance * 0.97
  ELSE balance * 0.95
END
WHERE account_number IN (
  SELECT account_number
  FROM Depositor
);
```

4. Consider the employee database consisting of the following tables, where the primary keys are underlined.

Employee (<u>employee-name</u>, street, city)

Works (<u>employee-name</u>, <u>company-name</u>, salary)

Company (<u>company-name</u>, city)

Manages (<u>employee-name</u>, manages-name)

Write down the SQL expressions for the following queries:

i. Find the names, cities and salaries of all employees who work for IFIC Bank Ltd.
ii. Find the total salaries of each company.
iii. Give all employees of First Bank Corporation a 20 percent salary raise.
iv. Find the names of all employees in this database who do not work for First Bank Corporation.

DDL:

```
CREATE DATABASE IF NOT EXISTS q4_company_database;

USE q4_company_database;


CREATE TABLE Employee (

   employee_name VARCHAR(50),

   street VARCHAR(100),

   city VARCHAR(50),

   PRIMARY KEY (employee_name)

);

CREATE TABLE Company (

   company_name VARCHAR(100),

   city VARCHAR(50),

   PRIMARY KEY (company_name)

);

CREATE TABLE Works (

   employee_name VARCHAR(50),

   company_name VARCHAR(100),

   salary DECIMAL(10,2),

   PRIMARY KEY (employee_name, company_name),

   FOREIGN KEY (employee_name) REFERENCES Employee(employee_name),

   FOREIGN KEY (company_name) REFERENCES Company(company_name)

);

CREATE TABLE Manages (

   employee_name VARCHAR(50),

   manages_name VARCHAR(50),
   PRIMARY KEY (employee_name),
   FOREIGN KEY (employee_name) REFERENCES Employee(employee_name),
   FOREIGN KEY (manages_name) REFERENCES Employee(employee_name)
 );
```

DML:

```
INSERT INTO Employee (employee_name, street, city)
VALUES ('John Doe', '123 Main St', 'New York'),
    ('Alice Smith', '456 Elm St', 'Los Angeles'),
    ('Bob Johnson', '789 Oak St', 'Chicago');
INSERT INTO Company (company_name, city)
VALUES ('IFIC Bank Ltd', 'New York'),
    ('First Bank Corporation', 'Los Angeles'),
    ('Second Company', 'Chicago');
INSERT INTO Works (employee_name, company_name, salary)
VALUES ('John Doe', 'IFIC Bank Ltd', 50000),
    ('Alice Smith', 'First Bank Corporation', 60000),
    ('Bob Johnson', 'Second Company', 55000);
INSERT INTO Manages (employee_name, manages_name)
VALUES ('Alice Smith', 'Bob Johnson');
```

Write down the SQL expressions for the following queries:

i. Find the names, cities and salaries of all employees who work for IFIC Bank Ltd.

```
SELECT E.employee_name, E.city, W.salary
FROM Employee E
JOIN Works W ON E.employee_name = W.employee_name
WHERE W.company_name = 'IFIC Bank Ltd';
```

ii. Find the total salaries of each company.

```
SELECT W.company_name, SUM(W.salary) AS total_salary
FROM Works W
GROUP BY W.company_name;
```

iii. Give all employees of First Bank Corporation a 20 percent salary raise.

```
UPDATE Works
SET salary = salary * 1.20
```

```
WHERE company_name = 'First Bank Corporation';
```

iv. Find the names of all employees in this database who do not work for First Bank Corporation.

```
SELECT employee_name

FROM Employee

WHERE employee_name NOT IN (

   SELECT employee_name

   FROM Works

   WHERE company_name = 'First Bank Corporation'

);
```

5. Consider the following schemas for "car_insurance" database relations, where the primary keys are underlined.

Person (driver-id, name, address)

Car (license, model, year)

Accident (report-number, date, location)

Owns (driver-id, license)

Participate (driver-id, car, report-number, damage amount)

Write down the SQL expressions for the following queries:

i. Add a new accident to the database (assume any values for required attributes).

ii. Delete the Toyota belonging to "Simanto".

iii. Find the total number of people who owned cars that were involved in accidents in 2020.

iv. Update the damage amount for the car with license number "DHAKA 4000" in the accident with report number "AR 2197" to 30,000/-

DDL:

```
CREATE DATABASE IF NOT EXISTS q5_car_insurance_database;

USE q5_car_insurance_database;


CREATE TABLE Person (

   driver_id INT PRIMARY KEY,

   name VARCHAR(100),

   address VARCHAR(200)

);

CREATE TABLE Car (

   license VARCHAR(20) PRIMARY KEY,
   model VARCHAR(50),
   year INT
);
CREATE TABLE Accident (
   report_number INT PRIMARY KEY,
   date DATE,
   location VARCHAR(100)
);
CREATE TABLE Owns (
   driver_id INT,
   license VARCHAR(20),
   PRIMARY KEY (driver_id, license),
   FOREIGN KEY (driver_id) REFERENCES Person(driver_id),
   FOREIGN KEY (license) REFERENCES Car(license)
);
CREATE TABLE Participate (
   driver_id INT,
   car VARCHAR(20),
   report_number INT,
   damage_amount DECIMAL(10, 2),
   PRIMARY KEY (driver_id, car, report_number),
   FOREIGN KEY (driver_id) REFERENCES Person(driver_id),
   FOREIGN KEY (car) REFERENCES Car(license),
   FOREIGN KEY (report_number) REFERENCES Accident(report_number)
);
```

DML:

```
INSERT INTO Person (driver_id, name, address)
VALUES (1, 'Simanto', '123 Main St'),
         (2, 'John Doe', '456 Elm St');
INSERT INTO Car (license, model, year)
VALUES ('DHAKA 1000', 'Toyota', 2018),
         ('DHAKA 2000', 'Honda', 2019);
INSERT INTO Accident (report_number, date, location)
```

```
VALUES (100, '2020-05-20', 'Downtown'),
       (101, '2021-07-15', 'Uptown');
INSERT INTO Owns (driver_id, license)
VALUES (1, 'DHAKA 1000'),
       (2, 'DHAKA 2000');
INSERT INTO Participate (driver_id, car, report_number, damage_amount)
VALUES (1, 'DHAKA 1000', 100, 5000.00),
       (2, 'DHAKA 2000', 101, 3000.00);
```

Write down the SQL expressions for the following queries:

i. Add a new accident to the database (assume any values for required attributes).

```
INSERT INTO Accident (report_number, date, location)
VALUES (102, '2024-05-25', 'City Center');
```

ii. Delete the Toyota belonging to "Simanto".

```
DELETE FROM Car
WHERE license = (
  SELECT license
  FROM Owns
  WHERE driver_id = (
    SELECT driver_id
    FROM Person
    WHERE name = 'Simanto'
  ) AND model = 'Toyota'
);
```

iii. Find the total number of people who owned cars that were involved in accidents in 2020.

```
SELECT COUNT(DISTINCT P.driver_id) AS total_owners
FROM Participate P
JOIN Accident A ON P.report_number = A.report_number
WHERE YEAR(A.date) = 2020;
```

iv. Update the damage amount for the car with license number "DHAKA 4000" in the accident with report number "AR 2197" to 30,000/-.

```
UPDATE Participate
SET damage_amount = 30000.00
WHERE car = 'DHAKA 4000'
```

```
AND report_number = 'AR 2197';
```