**1.**

Question:

Consider the following schemas for "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 both a loan and an account at the bank.

ii. Find the average account balance at the “Perryridge” branch.

iii. Insert record into the account relation with the values of account number is "AC-101” at "Dhanmondi" branch and the balance is tk 30000.

Schema:

CREATE DATABASE IF NOT EXISTS bank;

USE bank;

CREATE TABLE branch (

  branch\_name VARCHAR(100),

  branch\_city VARCHAR(100),

  assets DECIMAL(10,2),

  PRIMARY KEY (branch\_name)

);

CREATE TABLE customer (

  customer\_name VARCHAR(100),

  customer\_street VARCHAR(255),

  customer\_city VARCHAR(255),

  PRIMARY KEY (customer\_name)

);

CREATE TABLE loan (

  loan\_number VARCHAR(100),

  branch\_name VARCHAR(100),

  amount DECIMAL(10,2),

  PRIMARY KEY (loan\_number),

  FOREIGN KEY (branch\_name) REFERENCES branch (branch\_name)

);

CREATE TABLE borrower (

  customer\_name VARCHAR(100),

  loan\_number VARCHAR(100),

  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 VARCHAR(100),

  branch\_name VARCHAR(100),

  balance DECIMAL(10,2),

  PRIMARY KEY (account\_number),

  FOREIGN KEY (branch\_name) REFERENCES branch (branch\_name)

);

CREATE TABLE depositor (

  customer\_name VARCHAR(100),

  account\_number VARCHAR(100),

  PRIMARY KEY (customer\_name, account\_number),

  FOREIGN KEY (customer\_name) REFERENCES customer (customer\_name),

  FOREIGN KEY (account\_number) REFERENCES account (account\_number)

);

INSERT INTO branch (branch\_name, branch\_city, assets)

VALUES ('Perryridge', 'CityA', 500000),

       ('Dhanmondi', 'CityB', 300000);

INSERT INTO customer (customer\_name, customer\_street, customer\_city)

VALUES ('John Doe', '123 Main Street', 'New York'),

       ('Jane Smith', '456 Elm Street', 'Los Angeles'),

       ('Mike Johnson', '789 Oak Street', 'Chicago');

INSERT INTO loan (loan\_number, branch\_name, amount)

VALUES ('LN-001', 'Perryridge', 5000),

       ('LN-002', 'Perryridge', 10000),

       ('LN-003', 'Dhanmondi', 7500);

INSERT INTO account (account\_number, branch\_name, balance)

VALUES ('AC-001', 'Perryridge', 10000),

       ('AC-002', 'Perryridge', 20000),

       ('AC-003', 'Dhanmondi', 15000);

INSERT INTO borrower (customer\_name, loan\_number)

VALUES ('John Doe', 'LN-001'),

       ('Jane Smith', 'LN-002'),

       ('Mike Johnson', 'LN-003');

INSERT INTO depositor (customer\_name, account\_number)

VALUES ('John Doe', 'AC-001'),

       ('Jane Smith', 'AC-002'),

       ('Mike Johnson', 'AC-003');

Query & Results:  
(i) Find all customers who have both a loan and an account at the bank.

**Query:**

SELECT DISTINCT c.customer\_name

FROM customer c

JOIN borrower b ON c.customer\_name = b.customer\_name

JOIN loan l ON b.loan\_number = l.loan\_number

JOIN branch br ON l.branch\_name = br.branch\_name

JOIN account a ON br.branch\_name = a.branch\_name;

**Result:**

A screenshot of a computer

Description automatically generated

(ii) Find the average account balance at the 'Perryridge' branch.

**Query:**

SELECT AVG(balance) AS average\_balance

FROM Account

WHERE branch\_name = 'Perryridge';

**Result:**



(iii) Insert a record into the account relation with the values of account number as "AC101" at "Dhanmondi" branch and the balance is tk 30000.

**Query:**

INSERT INTO account (account\_number, branch\_name, balance)

VALUES ('AC-101', 'Dhanmondi', 30000);

**Result:**



**2.**

Question:

Consider the following relational schema:

employee(emp-no, name, office, age)

books(isbn, title, author, publisher)

loan(emp-no, isbn, date)

Write down the SQL expression for the following queries:

i. Print the names of all employees who have borrowed any book published by "XYZ".

ii. Print the names of all employees who have borrowed all books published by "XYZ".

iii. For each publisher, print the names of employees who have borrowed more than five books of that publisher.

Schema:

CREATE DATABASE IF NOT EXISTS library;

USE library;

CREATE TABLE employee (

  emp\_no INT PRIMARY KEY,

  name VARCHAR(50),

  office VARCHAR(50),

  age INT

);

CREATE TABLE books (

  isbn VARCHAR(20) PRIMARY KEY,

  title VARCHAR(100),

  author VARCHAR(50),

  publisher VARCHAR(50)

);

CREATE TABLE loan (

  emp\_no INT,

  isbn VARCHAR(20),

  date DATE,

  FOREIGN KEY (emp\_no) REFERENCES employee(emp\_no),

  FOREIGN KEY (isbn) REFERENCES books(isbn)

);

INSERT INTO `books` (`isbn`, `title`, `author`, `publisher`) VALUES

('ISBN001', 'Book 1', 'Author 1', 'XYZ'),

('ISBN002', 'Book 2', 'Author 2', 'ABC'),

('ISBN003', 'Book 3', 'Author 3', 'XYZ'),

('ISBN004', 'Book 4', 'Author 4', 'XYZ'),

('ISBN005', 'Book 5', 'Author 5', 'XYZ'),

('ISBN006', 'Book 6', 'Author 6', 'XYZ'),

('ISBN007', 'Book 7', 'Author 1', 'XYZ');

INSERT INTO `employee` (`emp\_no`, `name`, `office`, `age`) VALUES

(1, 'John Doe', 'Office A', 30),

(2, 'Jane Smith', 'Office B', 35),

(3, 'Mike Johnson', 'Office A', 28),

(4, 'Emily Brown', 'Office C', 32);

INSERT INTO `loan` (`emp\_no`, `isbn`, `date`) VALUES

(1, 'ISBN001', '2023-06-01'),

(1, 'ISBN002', '2023-06-02'),

(1, 'ISBN003', '2023-06-03'),

(2, 'ISBN001', '2023-06-04'),

(2, 'ISBN002', '2023-06-05'),

(3, 'ISBN001', '2023-06-06'),

(3, 'ISBN002', '2023-06-07'),

(3, 'ISBN003', '2023-06-08'),

(3, 'ISBN004', '2023-06-09'),

(4, 'ISBN001', '2023-06-10'),

(4, 'ISBN002', '2023-06-11'),

(4, 'ISBN003', '2023-06-12'),

(4, 'ISBN004', '2023-06-13'),

(4, 'ISBN005', '2023-06-14'),

(4, 'ISBN006', '2023-06-15'),

(1, 'ISBN001', '2023-06-01'),

(1, 'ISBN002', '2023-06-02'),

(1, 'ISBN003', '2023-06-03'),

(1, 'ISBN001', '2023-06-04'),

(1, 'ISBN002', '2023-06-05'),

(1, 'ISBN001', '2023-06-06'),

(1, 'ISBN002', '2023-06-07'),

(1, 'ISBN003', '2023-06-08'),

(1, 'ISBN004', '2023-06-09'),

(1, 'ISBN001', '2023-06-10'),

(1, 'ISBN002', '2023-06-11'),

(1, 'ISBN003', '2023-06-12'),

(1, 'ISBN004', '2023-06-13'),

(1, 'ISBN005', '2023-06-14'),

(1, 'ISBN006', '2023-06-15'),

(1, 'ISBN007', '2023-06-01');

Query & Results:

(i) Print the names of all employees who have borrowed any book published by "XYZ".

**Query:**

SELECT DISTINCT e.name

FROM employee e

JOIN loan l ON e.emp\_no = l.emp\_no

JOIN books b ON l.isbn = b.isbn

WHERE b.publisher = 'XYZ';

**Result:**



(ii) Print the names of all employees who have borrowed all books published by "XYZ".

**Query:**

SELECT e.name

FROM employee e

WHERE NOT EXISTS (

    SELECT b.isbn

    FROM books b

    WHERE b.publisher = 'XYZ'

    AND NOT EXISTS (

        SELECT l.isbn

        FROM loan l

        WHERE l.emp\_no = e.emp\_no

        AND l.isbn = b.isbn

    )

);

**Result:**



(iii) For each publisher, print the names of employees who have borrowed more than five books from that publisher.

**Query:**

SELECT b.publisher, e.name

FROM books b

JOIN loan l ON b.isbn = l.isbn

JOIN employee e ON l.emp\_no = e.emp\_no

GROUP BY b.publisher, e.name

HAVING COUNT(DISTINCT b.isbn) > 5;

**Result:**



**3.**

Question:

3. Consider the following schemas for "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 the number of depositors at each branch.

ii. List in alphabetical order all customers who have a loan at the 'Perryridge' branch.

iii. Update database to change Karim's street to a new one.

Schema:

CREATE DATABASE IF NOT EXISTS bank;

USE bank;

CREATE TABLE branch (

  branch\_name VARCHAR(50),

  branch\_city VARCHAR(50),

  assets DECIMAL(10, 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(10, 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(10, 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)

);

INSERT INTO branch (branch\_name, branch\_city, assets)

VALUES ('Perryridge', 'City A', 100000),

       ('Brighton', 'City B', 150000),

       ('Weston', 'City C', 200000);

INSERT INTO customer (customer\_name, customer\_street, customer\_city)

VALUES ('John Smith', '123 Main St', 'City A'),

       ('Alice Johnson', '456 Oak St', 'City B'),

       ('Robert Brown', '789 Elm St', 'City C'),

       ('Karim', '987 Maple St', 'City D');

INSERT INTO loan (loan\_number, branch\_name, amount)

VALUES (1, 'Perryridge', 5000),

       (2, 'Perryridge', 10000),

       (3, 'Brighton', 7500),

       (4, 'Weston', 12000);

INSERT INTO borrower (customer\_name, loan\_number)

VALUES ('John Smith', 1),

       ('Alice Johnson', 2),

       ('Robert Brown', 3),

       ('Karim', 4);

INSERT INTO account (account\_number, branch\_name, balance)

VALUES (1001, 'Perryridge', 5000),

       (1002, 'Perryridge', 10000),

       (1003, 'Brighton', 7500),

       (1004, 'Weston', 12000);

INSERT INTO depositor (customer\_name, account\_number)

VALUES ('John Smith', 1001),

       ('Alice Johnson', 1002),

       ('Robert Brown', 1003),

       ('Karim', 1004);

Query & Results:

(i) Find the number of depositors at each branch.

**Query:**

SELECT account.branch\_name, COUNT(\*) AS depositor\_count

FROM depositor

JOIN account ON depositor.account\_number = account.account\_number

GROUP BY account.branch\_name;

**Result:**

A screenshot of a computer

Description automatically generated

(ii) List in alphabetical order all customers who have a loan at the 'Perryridge' branch.

**Query:**

SELECT customer\_name

FROM customer

WHERE customer\_name IN (

  SELECT customer\_name

  FROM borrower

  WHERE loan\_number IN (

    SELECT loan\_number

    FROM loan

    WHERE branch\_name = 'Perryridge'

  )

)

ORDER BY customer\_name;

**Result:**



(iii) Update database to change karim's street to a new one.

**Query:**

UPDATE customer

SET customer\_street = 'New Street'

WHERE customer\_name = 'Karim';

**Result:**



**4.**

Question:

4. Consider the following schemas for "company" database relations, where the primary keys are underlined.

employee (employee-name, street, city)

works (employee-name, company-name, salary)

company (company-name, city)

manages (employee-name, manager-name)

Write down the SQL expressions for the following queries:

i. Find the total salary of each company.

ii. Find all employees in the database who do not work for ACI Ltd.

iii. Insert record into the employee table with proper values.

Schema:

Data-Definition Language (DDL)

CREATE DATABASE IF NOT EXISTS company;

USE company;

CREATE TABLE employee (

  employee\_name VARCHAR(50),

  street VARCHAR(50),

  city VARCHAR(50),

  PRIMARY KEY (employee\_name)

);

CREATE TABLE company (

  company\_name VARCHAR(50),

  city VARCHAR(50),

  PRIMARY KEY (company\_name)

);

CREATE TABLE works (

  employee\_name VARCHAR(50),

  company\_name VARCHAR(50),

  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),

  manager\_name VARCHAR(50),

  PRIMARY KEY (employee\_name),

  FOREIGN KEY (employee\_name) REFERENCES employee (employee\_name),

  FOREIGN KEY (manager\_name) REFERENCES employee (employee\_name)

);

Data-Manipulation Language (DML):

INSERT INTO employee (employee\_name, street, city) VALUES

('John Smith', '123 Elm Street', 'New York'),

('Jane Doe', '456 Oak Avenue', 'Los Angeles'),

('Michael Johnson', '789 Maple Drive', 'Chicago');

INSERT INTO company (company\_name, city) VALUES

('ACI Ltd', 'New York'),

('XYZ Corp', 'Los Angeles'),

('ABC Inc', 'Chicago');

INSERT INTO works (employee\_name, company\_name, salary) VALUES

('John Smith', 'ACI Ltd', 5000),

('Jane Doe', 'XYZ Corp', 6000),

('Michael Johnson', 'ABC Inc', 5500),

('John Smith', 'XYZ Corp', 7000),

('Jane Doe', 'ABC Inc', 6500);

INSERT INTO manages (employee\_name, manager\_name) VALUES

('John Smith', 'Jane Doe'),

('Michael Johnson', 'Jane Doe');

Query & Results:

(i) Find the total salary of each company.

**Query:**

SELECT company.company\_name, SUM(works.salary) AS total\_salary

FROM company

LEFT JOIN works ON company.company\_name = works.company\_name

GROUP BY company.company\_name;

**Result:**



(ii) Find all employees in the database who do not work for ACI Ltd.

**Query:**

SELECT employee.employee\_name

FROM employee

LEFT JOIN works ON employee.employee\_name = works.employee\_name

WHERE works.company\_name IS NULL OR works.company\_name <> 'ACI Ltd';

**Result:**

A screenshot of a computer

Description automatically generated

(iii) Insert record into the employee table with proper values.

**Query:**

INSERT INTO employee (employee\_name, street, city)

VALUES ('John Doe', '123 Main Street', 'New York');

**Result:**



**5.**

Question:

5. Consider the following schemas for "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 the loan number of those loans with loan amounts between tk 10000 and tk 20000.

ii. Find all customers who have account but no loan at the bank.

iii. Add a record in "customer" table using a form.

Schema:

Data-Definition Language (DDL)

CREATE DATABASE IF NOT EXISTS bank;

USE bank;

CREATE TABLE `branch` (

  `branch\_name` VARCHAR(100) PRIMARY KEY,

  `branch\_city` VARCHAR(100),

  `assets` DECIMAL(10,2)

);

CREATE TABLE `customer` (

  `customer\_name` VARCHAR(100) PRIMARY KEY,

  `customer\_street` VARCHAR(100),

  `customer\_city` VARCHAR(100)

);

CREATE TABLE `loan` (

  `loan\_number` INT PRIMARY KEY,

  `branch\_name` VARCHAR(100),

  `amount` DECIMAL(10,2),

  FOREIGN KEY (`branch\_name`) REFERENCES `branch` (`branch\_name`)

);

CREATE TABLE `borrower` (

  `customer\_name` VARCHAR(100),

  `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 PRIMARY KEY,

  `branch\_name` VARCHAR(100),

  `balance` DECIMAL(10,2),

  FOREIGN KEY (`branch\_name`) REFERENCES `branch` (`branch\_name`)

);

CREATE TABLE `depositor` (

  `customer\_name` VARCHAR(100),

  `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`)

);

Data-Manipulation Language (DML):

INSERT INTO branch (branch\_name, branch\_city, assets)

VALUES ('Branch1', 'City1', 100000.00),

       ('Branch2', 'City2', 200000.00),

       ('Branch3', 'City3', 150000.00);

INSERT INTO customer (customer\_name, customer\_street, customer\_city)

VALUES ('Customer1', 'Street1', 'City1'),

       ('Customer2', 'Street2', 'City2'),

       ('Customer3', 'Street3', 'City3');

INSERT INTO loan (loan\_number, branch\_name, amount)

VALUES (1, 'Branch1', 15000.00),

       (2, 'Branch2', 25000.00),

       (3, 'Branch3', 10000.00);

INSERT INTO borrower (customer\_name, loan\_number)

VALUES ('Customer1', 1),

       ('Customer2', 2),

       ('Customer2', 3);

INSERT INTO account (account\_number, branch\_name, balance)

VALUES (1001, 'Branch1', 5000.00),

       (1002, 'Branch2', 10000.00),

       (1003, 'Branch3', 7500.00);

INSERT INTO depositor (customer\_name, account\_number)

VALUES ('Customer1', 1001),

       ('Customer2', 1002),

       ('Customer3', 1003);

Query & Results:

(i) Find the loan number of those loans with loan amounts between tk 10000 and tk 20000.

**Query:**

SELECT loan\_number

FROM loan

WHERE amount BETWEEN 10000 AND 20000;

**Result:**

A screenshot of a computer

Description automatically generated

(ii) Find all customers who have an account but no loan at the bank.

**Query:**

SELECT c.customer\_name

FROM customer c

JOIN depositor d ON c.customer\_name = d.customer\_name

LEFT JOIN borrower b ON c.customer\_name = b.customer\_name

WHERE b.customer\_name IS NULL;

**Result:**



(iii) Add a record in the "customer" table using a form.

**Query:**

INSERT INTO customer (customer\_name, customer\_street, customer\_city)

VALUES ('John Doe', '123 Main St', 'New York');

**Result:**



**6.**

Question:

6. Consider the following schemas for "emp" database relations, Where the primary keys are underlined.

employee (employee-name, street, city)

works (employee-name, company-name, salary)

company (company-name, city)

manages (employee-name, manager-name)

Write down the SQL expressions for the following queries:

i. Find the number of tuples in the work relation.

ii. Find all employees in the database who earn more than each employee of Trust Bank.

Schema:

Data-Definition Language (DDL)

CREATE DATABASE IF NOT EXISTS emp;

USE emp;

CREATE TABLE employee (

  employee\_name VARCHAR(100),

  street VARCHAR(255),

  city VARCHAR(255),

  PRIMARY KEY (employee\_name)

);

CREATE TABLE company (

  company\_name VARCHAR(100),

  city VARCHAR(255),

  PRIMARY KEY (company\_name)

);

CREATE TABLE works (

  employee\_name VARCHAR(100),

  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(100),

  manager\_name VARCHAR(100),

  PRIMARY KEY (employee\_name),

  FOREIGN KEY (employee\_name) REFERENCES employee(employee\_name)

);

Data-Manipulation Language (DML):

INSERT INTO employee (employee\_name, street, city)

VALUES

  ('John Doe', '123 Main Street', 'New York'),

  ('Jane Smith', '456 Elm Avenue', 'Los Angeles');

INSERT INTO company (company\_name, city)

VALUES

  ('Acme Corporation', 'New York'),

  ('Trust Bank', 'Los Angeles');

INSERT INTO manages (employee\_name, manager\_name)

VALUES

  ('John Doe', 'Jane Smith');

INSERT INTO works (employee\_name, company\_name, salary)

VALUES

  ('John Doe', 'Acme Corporation', 5000),

  ('John Doe', 'Trust Bank', 6000),

  ('Jane Smith', 'Acme Corporation', 5500);

Query & Results:

(i) Find the number of tuples in the works relation:

**Query:**

SELECT COUNT(\*) AS tuple\_count

FROM works;

**Result:**



(ii) Find all employees in the database who earn more than each employee of Trust Bank.

**Query:**

SELECT e.employee\_name

FROM employee e

WHERE NOT EXISTS (

  SELECT \*

  FROM works w

  JOIN company c ON w.company\_name = c.company\_name

  WHERE c.company\_name = 'Trust Bank' AND w.salary <= (SELECT MAX(salary) FROM works WHERE employee\_name = e.employee\_name)

);

**Result:**

A close up of a name

Description automatically generated