INDEX

1.

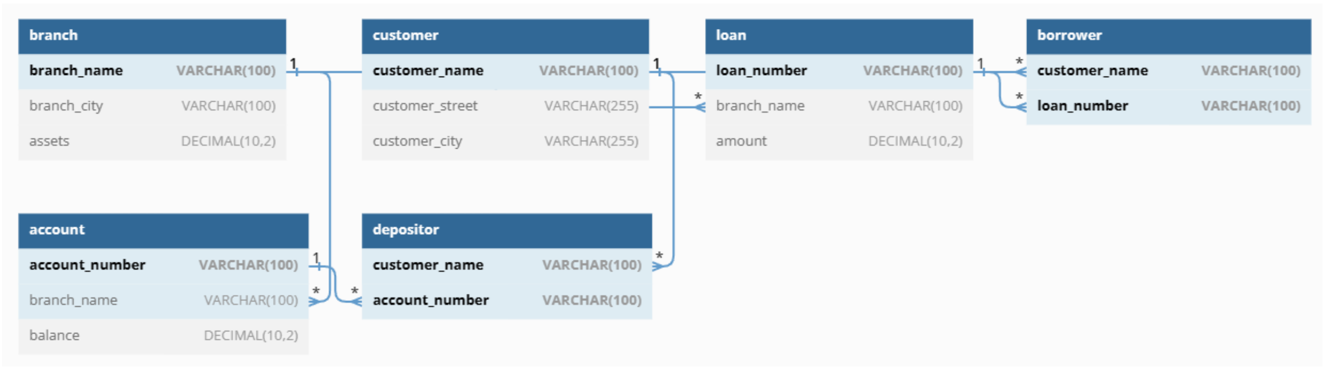
Question:

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

ER-Diagram:



Schema:

Data-Definition Language (DDL)

```
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)
);
```

Data-Manipulation Language (DML):

```
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');
```

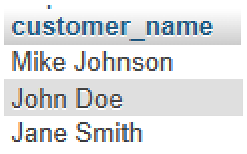
Query & Results:

(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
LIMIT 0, 25;
```

Result:



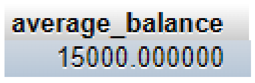
customer_name
Mike Johnson
John Doe
Jane Smith

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

Query:

```
SELECT AVG(balance) AS average_balance
FROM Account
WHERE branch_name = 'Perryridge';
```

Result:



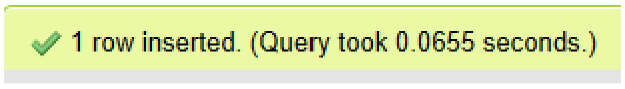
average_balance
15000.000000

(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:



```
1 row inserted. (Query took 0.0655 seconds.)
```

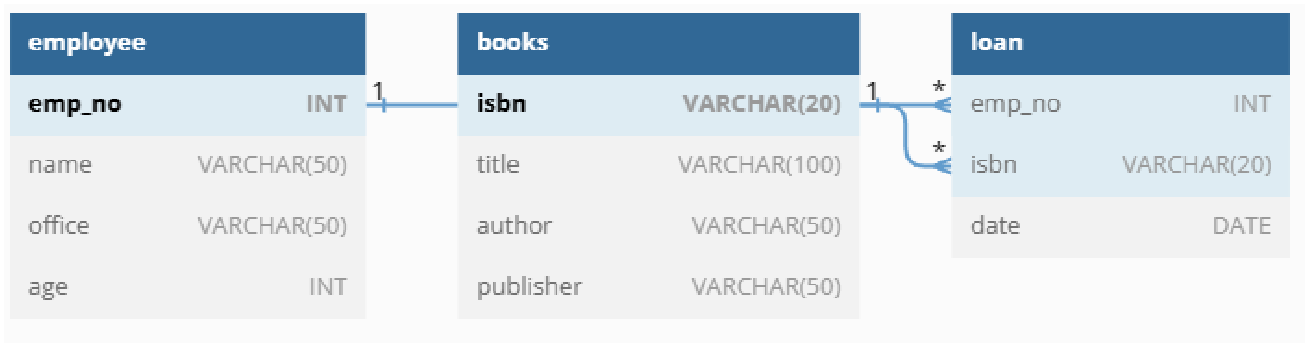
2.

Question:

2. 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.

ER-Diagram:



Schema:

Data-Definition Language (DDL)

```
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)
);
```

Data-Manipulation Language (DML):

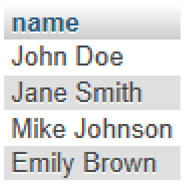
```
INSERT INTO `books` (`isbn`, `title`, `author`, `publisher`) VALUES
('ISBN001', 'Book 1', 'Author 1', 'XYZ'),
('ISBN002', 'Book 2', 'Author 2', 'ABC'),
('ISBN003', 'Book 3', 'Author 3', 'XYZ'),
('ISBN004', 'Book 4', 'Author 4', 'XYZ'),
('ISBN005', 'Book 5', 'Author 5', 'XYZ'),
('ISBN006', 'Book 6', '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:

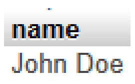


name
John Doe
Jane Smith
Mike Johnson
Emily Brown

(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:



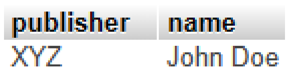
name
John Doe

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



publisher	name
XYZ	John Doe

3.

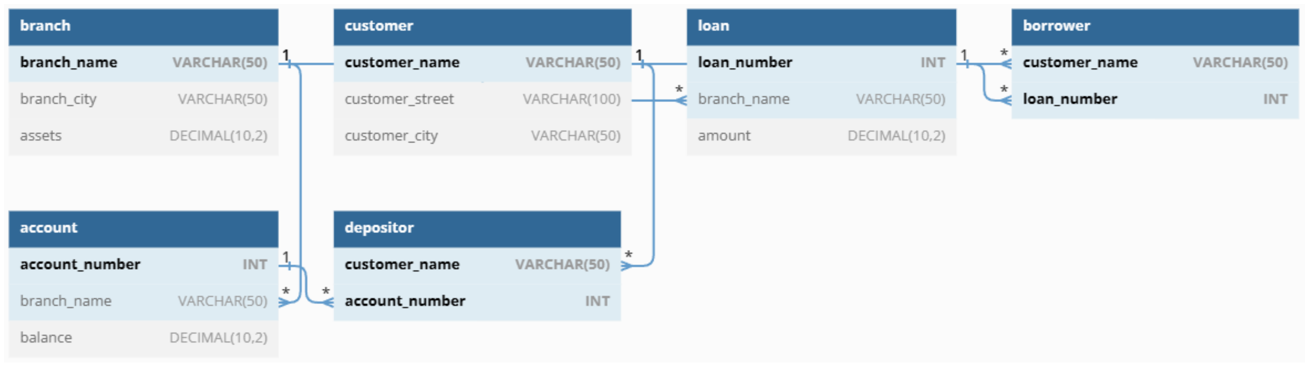
Question:

3. Consider the following schemas for "bank" database relations, where the primary keys are underlined.

branch (<u>branch-name</u>, branch_city, assets)
customer (<u>customer-name</u>, customer_street, customer_city)
loan (<u>loan-number</u>, branch_name, amount)
borrower (<u>customer-name</u>, <u>loan-number</u>)
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 the number of depositor at each branch.
ii. List in alphabetic order all customers who have a loan at the 'Perryridge' branch.
iii. Update database to change karim's street to a new one.

ER-Diagram:



Schema:

Data-Definition Language (DDL)

```
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)
);
```

Data-Manipulation Language (DML):

```
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');
```

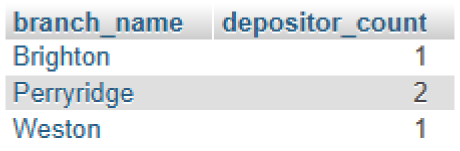
```
INSERT INTO loan (loan_number, branch_name, amount)
VALUES (1, 'Perryridge', 5000),
       (2, 'Perryridge', 10000),
       (3, 'Brighton', 7500),
       (4, 'Weston', 12000);

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 depositor 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:

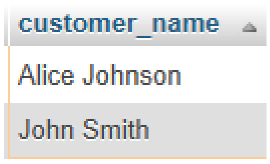


branch_name	depositor_count
Brighton	1
Perryridge	2
Weston	1

(ii) List in alphabetic 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:

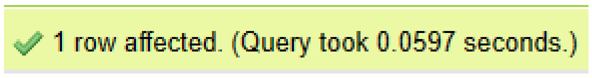


customer_name
Alice Johnson
John Smith

(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:



```
1 row affected. (Query took 0.0597 seconds.)
```

4.

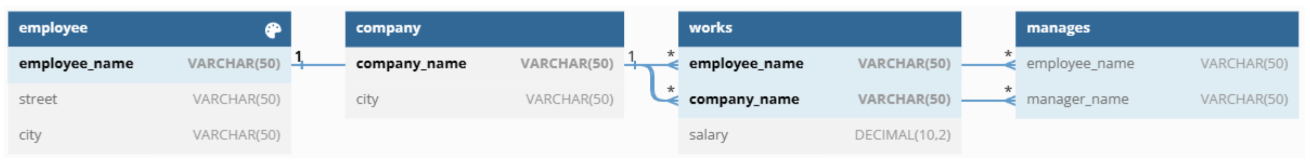
Question:

4. Consider the following schemas for "company" database relations, 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>, 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.

ER-Diagram:



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');
```

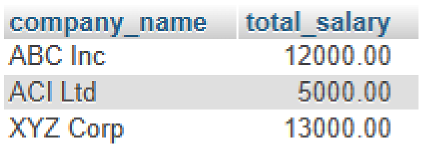
Query & Results:

(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:

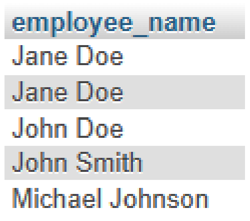


company_name	total_salary
ABC Inc	12000.00
ACI Ltd	5000.00
XYZ Corp	13000.00

(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:

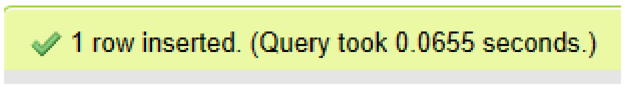


employee_name
Jane Doe
Jane Doe
John Doe
John Smith
Michael Johnson

(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:



```
1 row inserted. (Query took 0.0655 seconds.)
```

5.

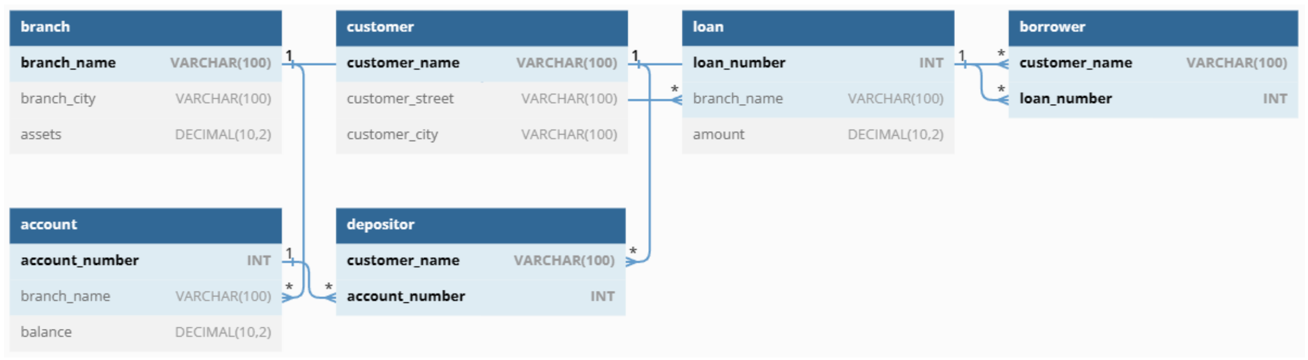
Question:

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

branch (<u>branch-name</u>, branch_city, assets)
customer (<u>customer-name</u>, customer_street, customer_city)
loan (<u>loan-number</u>, branch_name, amount)
borrower (<u>customer-name</u>, <u>loan-number</u>)
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 the loan number of those loans with loan amounts between tk 10000 and tk 20000.
i.. Find all customers who have account but no loan at the bank.
iii. Add a record in "customer" table using a form.

ER-Diagram:



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`)
);
```

<u>Data-Manipulation Language (DML):</u>

```
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)
```

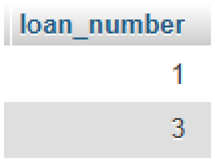
```
VALUES ('Customer1', 1),
       ('Customer2', 2),
       ('Customer2', 3);

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:

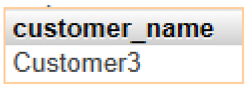


loan_number
1
3

(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:

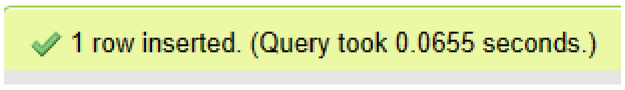


customer_name
Customer3

(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:



```
1 row inserted. (Query took 0.0655 seconds.)
```

6.

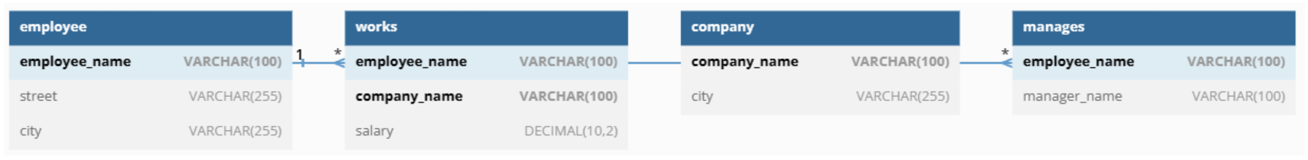
Question:

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

ER-Diagram:


employee
employee_name VARCHAR(100)
street VARCHAR(255)
city VARCHAR(255)
1
*
works
employee_name VARCHAR(100)
company_name VARCHAR(100)
salary DECIMAL(10,2)
company
company_name VARCHAR(100)
city VARCHAR(255)
*
manages
employee_name VARCHAR(100)
manager_name VARCHAR(100)

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 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)
);

CREATE TABLE company (
  company_name VARCHAR(100),
  city VARCHAR(255),
  PRIMARY KEY (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)
);
```

<u>Data-Manipulation Language (DML):</u>

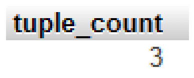
```
INSERT INTO employee (employee_name, street, city)
VALUES
  ('John Doe', '123 Main Street', 'New York'),
A  ('Jane Smith', '456 Elm Avenue', 'Los Angeles');

INSERT INTO works (employee_name, company_name, salary)
VALUES
  ('John Doe', 'Acme Corporation', 5000),
  ('John Doe', 'Trust Bank', 6000),
  ('Jane Smith', 'Acme Corporation', 5500);

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');
```

<u>Query & Results:</u>

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

```
SELECT COUNT(*) AS tuple_count
FROM works;
```

Result:

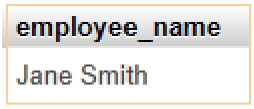


```
tuple_count
          3
```

(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:



employee_name
Jane Smith