Mid Term Assignment

The schema definition of some relations are given as the following:

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

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

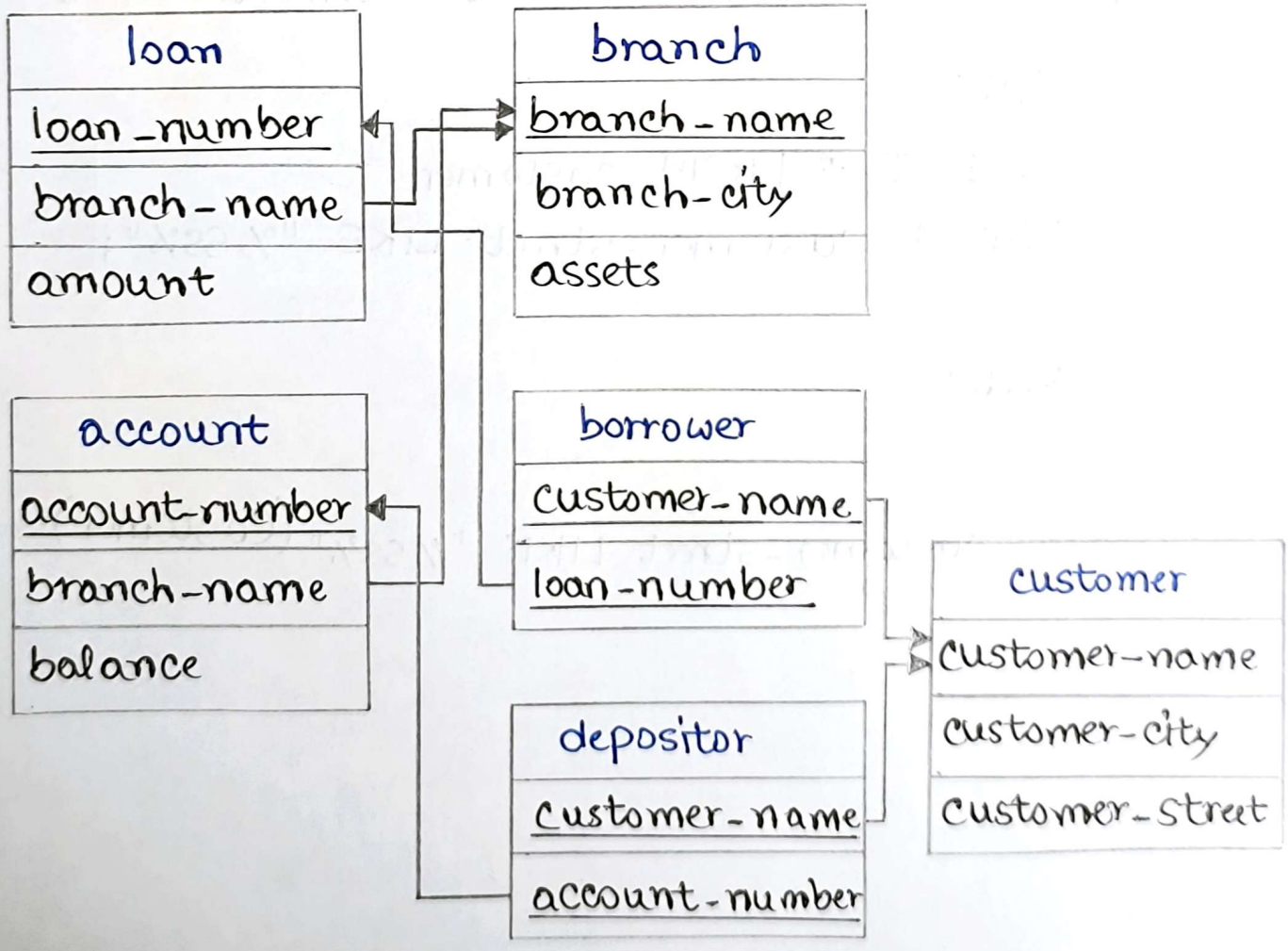
borrower (customer-name, loan-number)

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

depositor (customer-name, account-number)

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

Q1. Draw the schema diagram from the given relations.



Q2. Write down the SQL and relational algebra expressions for the questions below:

① Find the average loan amount from each branch.

SQL:

```
SELECT branch-name, AVG(amount)
FROM loan
GROUP BY branch-name;
```

R.A.:

$$branch\text{-}name\ \mathcal{G}_{AVG(amount)}\ (Loan)$$

② Write a query to show the details of a customer whose street name has two consecutive "s"

SQL:

```
SELECT * FROM customer
WHERE customer-street LIKE "%ss%";
```

R.A.:

$$\sigma_{customer\text{-}street\ LIKE\ "\%ss\%"}\ (Customer)$$

③ Find all customers who have a loan from the bank, find their names and loan_numbers.

SQL:

```
SELECT customer_name, loan_number
FROM borrower;
```

R.A.:

$\Pi_{customer\_name,\ loan\_number}(Borrower)$

④ Find the list of all customers in alphabetic order who have a loan in the "Perryridge" branch.

SQL:

```
SELECT DISTINCT customer_name
FROM borrower NATURAL JOIN loan
WHERE branch_name = "Perryridge"
ORDER BY customer_name ASC;
```

R.A.:

$\tau_{customer\_name}(\Pi_{customer\_name}(\sigma_{branch\_name="Perryridge"}(Borrower \bowtie Loan)))$

⑤ Find all customers having a loan, an account or both at the bank.

SQL:

```
SELECT customer-name FROM borrower
UNION SELECT customer-name FROM depositor;
```

R.A.:

$\Pi_{customer-name}(Borrower) \cup \Pi_{customer-name}(Depositor)$

⑥ Find the names of all customers whose street address includes the substring "Main"

SQL:

```
SELECT customer-name FROM customer
WHERE customer-street LIKE "%Main%";
```

R.A.:

$\Pi_{customer-name}(\sigma_{customer-street\ LIKE\ "\%Main\%"}(Customer))$

⑦ Find the average loan amount from each branch where the average loan amount is greater than 1500.

SQL:

```
SELECT branch_name, AVG(amount) FROM loan
GROUP BY branch_name
HAVING AVG(amount) > 1500;
```

R.A.:

$\gamma_{branch-name,\ AVG(amount)}(Loan)\ \sigma_{AVG(amount) > 1500}$

(8) Count the number of tuples in customer relation.

SQL:

```
SELECT COUNT(customer-name)
FROM customer ;
```

R.A.:

$$G_{COUNT(customer\text{-}name)}(Customer)$$

(9) Find average and maximum account balance at each branch.

SQL:

```
SELECT branch-name, AVG(balance),
MAX(balance) FROM account
GROUP BY branch-name;
```

R.A.:

$$\gamma_{branch\text{-}name,\ AVG(balance),\ MAX(balance)}(Account)$$

(10) Find the names of all those customers who have a loan at Perryridge branch.

SQL:

```
SELECT customer-name
FROM borrower NATURAL JOIN loan
WHERE branch-name = "Perryridge"
```

R.A.:

$$\Pi_{customer\text{-}name}(\sigma_{branch\text{-}name = "Perryridge"}(Borrower \bowtie Loan))$$

(11) Delete the records of all accounts with balances below average at that bank.

SQL:

```
DELETE FROM account
WHERE balance < (
        SELECT AVG(balance)
        FROM account
);
```

R.A.:

$$Account \leftarrow Account - {}^{(Account)}\sigma_{balance < AVG(balance)}$$

(12) Consider the following tables: loan and borrower.

loan-number	branch-name	amount
L-11	RoundHill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

customer-name	loan-number
Adams	L-16
Curry	L-93
Hayes	L-15
Johnson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Perform Right Outer Join, Left Outer Join, Inner Join, and Natural Join and show the results.

Performing a Right Outer Join:

```
SELECT * FROM loan NATURAL RIGHT OUTER JOIN
        borrower;
```

loan-number	customer-name	branch-name	amount
L-16	Adams	Perryridge	1300
L-93	Curry	Mianus	500
L-15	Hayes	Perryridge	1500
L-14	Johnson	Downtown	1500
L-17	Jones	Downtown	1000
L-11	Smith	Round Hill	900
L-23	Smith	Red Woord	2000
L-17	Williams	Downtown	1000

Performing a left Outer Join:

```
SELECT * FROM loan NATURAL LEFT OUTER JOIN
        borrower
```

loan_number	branch-name	amount	customer-name
L-16	Perryridge	1300	Adams
L-93	Mianus	500	Curry
L-15	Perryridge	1500	Hayes
L-14	Downtown	1500	Johnson
L-17	Downtown	1000	Jones
L-11	Round Hill	900	Smith
L-23	Redwood	2000	Smith
L-17	Downtown	1000	Williams

Performing an Inner Join:

```
SELECT * FROM loan INNER JOIN borrower
```

loan-number	branch-name	amount	customer-name	loan-number
L-11	Round Hill	900	Adams	L-16
L-14	Downtown	1500	Adams	L-16
L-15	Perryridge	1500	Adams	L-16
L-16	Perryridge	1300	Adams	L-16
L-17	Downtown	1000	Adams	L-16
L-23	Redwood	2000	Adams	L-16
L-93	Mianus	500	Adams	L-16
L-11	Round Hill	900	Curry	L-93
L-14	Downtown	1500	Curry	L-93
L-15	Perryridge	1500	Curry	L-93
L-16	Perryridge	1300	Curry	L-93
L-17	Downtown	1000	Curry	L-93
L-23	Redwood	2000	Curry	L-93
L-93	Mianus	500	Curry	L-93
L-11	Round Hill	900	Hayes	L-15
L-14	Downtown	1500	Hayes	L-15
L-15	Perryridge	1500	Hayes	L-15
L-16	Perryridge	1300	Hayes	L-15
L-17	Downtown	1000	Hayes	L-15
L-23	Redwood	2000	Hayes	L-15
L-93	Mianus	500	Hayes	L-15
L-11	Round Hill	900	Johnson	L-14
L-14	Downtown	1500	Johnson	L-14
L-15	Perryridge	1500	Johnson	L-14
L-16	Perryridge	1300	Johnson	L-14

Performing a Natural Join:

```
SELECT * FROM loan NATURAL JOIN borrower
```

loan-number	branch-name	amount	customer-name
L-16	Perryridge	1300	Adams
L-93	Mianus	500	Curry
L-15	Perryridge	1500	Hayes
L-14	Downtown	1500	Johnson
L-17	Downtown	1000	Jones
L-11	Round Hill	900	Smith
L-23	Redwood	2000	Smith
L-17	Downtown	1000	Williams

Q3. Demonstrate the output of the previous questions from Q2 (1-10).

① Find the average loan amount from each branch.

```
SELECT branch-name, AVG(amount) "Average Loan"
FROM loan GROUP BY branch-name;
```

branch-name	Average Loan
Downtown	1250.00
Mianus	500.00
Perryridge	1400.00
Redwood	2000.00
Round Hill	900.00

② Write a query to show the details of a customer whose street name has two consecutive "s".

```
SELECT * FROM customer
WHERE customer_street LIKE "%ss%"
```

customer-name	customer-street	customer-city
Williams	Nassau	Princeton

③ Find all customers who have a loan from the bank, find their names and loan numbers.

```
SELECT customer_name, loan-number
FROM borrower;
```

customer-name	loan-number
Adams	L-16
Curry	L-93
Hayes	L-15
Johnson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

④ Find the list of all customers in alphabetic order who have a loan in the Perryridge branch.

```
SELECT DISTINCT customer-name
FROM borrower NATURAL JOIN loan
WHERE branch-name = "Perryridge"
ORDER BY customer-name ASC
```

customer-name
Adams
Hayes

⑤ Find all customers having a loan, an account, or both at the bank.

```
SELECT customer-name FROM borrower
UNION SELECT customer-name FROM depositor
```

customer-name
Adams
Curry
Hayes
Johnson
Jones
Smith
Williams
Lindsay
Turner

⑥ Find the names of all customers whose street address includes the substring "Main"

```
SELECT customer-name FROM customer
WHERE customer-street LIKE "%Main%"
```

customer-name
Hayes
Jones

⑦ Find the average loan amount from each branch where the average loan amount is greater than 1500.

```
SELECT branch-name, AVG(amount) FROM loan
GROUP BY branch-name
HAVING AVG(amount) > 1500
```

branch-name	AVG(amount)
Redwood	2000.00

⑧ Count the number of tuples in customer relation.

```
SELECT COUNT(customer-name)
FROM customer
```

COUNT(customer-name)
12

⑨ Find the average and maximum account balance at each branch.

```
SELECT branch-name, AVG(balance), MAX(balance)
FROM account GROUP BY branch-name
```

branch-name	AVG(balance)	MAX(balance)
Brighton	825.00	900
Downtown	500.00	500
Mianus	700.00	700
Perryridge	400.00	400
Redwood	700.00	700
Round Hill	350.00	350

⑩ Find the names of all those customers who have a loan at Perryridge branch.

```
SELECT customer-name FROM borrower NATURAL JOIN
loan WHERE branch-name = "Perryridge"
```

Customer-name
Adams
Hayes

Q4. Identify all possible superkeys and candidate keys for the given relation:

Employee (EmpID, Name, Email, Phone, Department)

In the Employee relation:

- EmpID is unique (potential Primary key)
- Email and Phone are usually unique
- Name and Department are not unique because two employees can have the same name or work in the same department.

Therefore, potential candidate keys are:

{EmpID}, {Email}, {Phone}

We know that, a superkey is any set of attributes that uniquely identifies a tuple. Any superset of a candidate key is also a superkey.

Possible superkeys are:

- {EmpID}
- {Email}
- {Phone}
- {EmpID, Name}, {EmpID, Email}, {EmpID, Phone}, {EmpID, Department}, ...
- {Email, Name}, {Email, Phone}, {Email, Department}, ...
- {Phone, Name}, {Phone, Department}, ...
- {EmpID, Name, Email, Phone, Department}

Basically, all subsets of the relation containing at least one superkey.

Since candidate keys are minimal superkeys, the candidate keys are:

{EmpID}, {Email}, {Phone}