Table of contents:

I. Scenario:

The bank is organized into branches, and each branch has a unique identifier, a name and an address. A branch serves an arbitrary number of customers.

Each customer has a unique identifier, a first and last name, the date of birth and their gender. A customer also administers one or more bank accounts.

A bank account has a unique identifier and its current balance. Each account is associated with one active card, and the card contains its unique number, the expiration date and whether or not the card has been blocked. An account can also initiate loans and transactions.

The database also keeps track of loans; and each loan has a unique identifier, the type of loan which can be for example: personal loans, student loans, etc., the amount of money that the customer has already paid back and in addition the start and the due dates of the loan. Other than that, each type of loan has an identifier, a name, a brief description, a base amount and a base interest rate.

Furthermore, the database holds records of transactions, each having a unique identifier, a description of the transaction and the amount and the date of the transfer.

II. Design choices:

1. Customer table

We thought about having the gender in a separate lookup table, which would have had two columns, id and name, and we would reference the id of the gender in our customer's gender column as a foreign key. In the end, we decided to simplify the

gender field of the customer to just contain “male” or “female”, due to the small and constant size of the proposed table of genders.

2. **Loan table**

We decided to add an extra field, called “loan_type_id” that would reference one of the entries from the Loan Type table, because a loan usually has several constant fields, such as a description, a base amount and a base interest rate (while at the same time removing repetitive data from the Loan table, such as the amount and the interest rate), and it would be more appropriate to store them in a lookup table and have a many-to-one relationship between Loan and Loan Type.

3. **Simplify the database**

We decided to simplify the database by removing some fields, in order to only have the basic banking system implemented.
Here are some of the fields that were removed from each table:

- **Branch table**
  - branch_code
- **Customer table**
  - ssn
  - middle_name
  - nationality
  - address
- **Account table**
  - iban
- **Card table**
  - security_code
- **Loan table**
  - amount
  - interest_rate

III. Normalization

1st form of normalization
In our situation, all the tables respect the 1st form of normalization, having no multiple values in a field. From the first instance of the database, we had already the 1st form normalization implemented, so no adjustments were needed.

2nd form of normalization:
In this case, our database respects the 2nd form normalization, meaning that all the non-primary keys depend on a primary key attribute. For example: in the Branch table, the name field depends on the id, and the address field is dependent on the id.

3rd form of normalization:
Our database follows the 3rd form normalization, because all non-primary keys are not dependent on another non-primary key. For example: In the Customer table, the gender field does not depend on branch_id, first_name, last_name, date_of_birth.

IV. SQL Queries

Create Schema and Tables

```
CREATE SCHEMA Bank;
```

```
USE Bank;
CREATE TABLE Branch (
    id INT,
    name CHAR(50) UNIQUE,
    address CHAR(50),
    PRIMARY KEY (id)
);
```

```
USE Bank;
CREATE TABLE Card (
    id INT,
    number CHAR(50) UNIQUE,
    expiration_date DATE,
    is_blocked BOOL,
    PRIMARY KEY (id)
```

```
);
```

```
USE Bank;
CREATE TABLE Loan_type (
    id INT,
    type CHAR(10) UNIQUE,
    description CHAR(100),
    base_amount DECIMAL,
    base_interest_rate DECIMAL,
    PRIMARY KEY (id)
);
```

```
USE Bank;
CREATE TABLE Loan_type (
    id INT,
    type CHAR(50) UNIQUE,
    description CHAR(100),
    base_amount DECIMAL(10, 3),
    base_interest_rate DECIMAL(10, 3),
    PRIMARY KEY (id)
);
```

```
USE Bank;
CREATE TABLE Customer (
    id INT,
    branch_id INT,
    first_name CHAR(50),
    last_name CHAR(50),
    date_of_birth DATE,
    gender CHAR(6),
    PRIMARY KEY (id),
    FOREIGN KEY (branch_id) REFERENCES Branch(id)
           ON UPDATE CASCADE
           ON DELETE SET NULL
);
```

```
USE Bank;
CREATE TABLE Account (
    id INT,
    customer_id INT,
    card_id INT,
```

```
    balance CHAR(50),
    PRIMARY KEY (id),
    FOREIGN KEY (customer_id) REFERENCES Customer(id)
           ON UPDATE CASCADE
           ON DELETE SET NULL,
     FOREIGN KEY (card_id) REFERENCES Card(id)
           ON UPDATE CASCADE
           ON DELETE SET NULL
);
```

```
USE Bank;
CREATE TABLE Loan (
    id INT,
    account_id INT,
    loan_type_id INT,
    amount_paid DECIMAL(10, 3),
    start_date DATE,
    due_date DATE,
    PRIMARY KEY (id),
    FOREIGN KEY (account_id) REFERENCES Account(id)
           ON UPDATE CASCADE
           ON DELETE SET NULL,
     FOREIGN KEY (loan_type_id) REFERENCES Loan_type(id)
           ON UPDATE CASCADE
           ON DELETE SET NULL
);
```

```
USE Bank;
CREATE TABLE Transaction (
    id INT,
    account_id INT,
    description CHAR(100),
    amount DECIMAL(10, 3),
    date DATE,
    PRIMARY KEY (id),
    FOREIGN KEY (account_id) REFERENCES Account(id)
           ON UPDATE CASCADE
           ON DELETE SET NULL
);
```

Create Users

```
CREATE USER 'paul2'@'%' IDENTIFIED BY 'password';
CREATE USER 'constantin2'@'%' IDENTIFIED BY 'password';
CREATE USER 'marius2'@'%' IDENTIFIED BY 'password';
GRANT ALL ON *.* TO 'paul2'@'%';
GRANT ALL ON *.* TO 'constantin2'@'%' WITH GRANT OPTION;
GRANT SELECT, UPDATE, DELETE ON *.* TO 'marius2'@'%';
SELECT * FROM mysql.user;

SHOW GRANTS for 'paul2'@'%';
```

Create View

```
USE Bank;
CREATE VIEW User_role_information AS
     SELECT User, Select_priv, Insert_priv, Update_priv,
Delete_priv, Create_priv
           FROM mysql.user
           WHERE Select_priv = 'Y' OR Insert_priv = 'Y' OR
Update_priv = 'Y' OR Delete_priv = 'Y' OR Create_priv = 'Y';
```

Populate the Database

```
USE Bank;
INSERT INTO Branch (id, name, address) VALUES ('1', 'Albertslund
Bank', 'Albertslund');
INSERT INTO Branch (id, name, address) VALUES ('2', 'Norrebro Bank',
'Albertslund');
INSERT INTO Branch (id, name, address) VALUES ('3', 'Kolding Bank',
'Kolding, Jutland');
INSERT INTO Branch (id, name, address) VALUES ('4', 'Glostrup Bank',
'Glostrup');
INSERT INTO Branch (id, name, address) VALUES ('5', 'Valby Bank',
'Valby');
```

```
USE Bank;
```

```
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES
('1', '1234567890123456', '2021-01-30', TRUE);
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES
('2', '1234567890123457', '2022-08-20', TRUE);
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES
('3', '1234567890123458', '2023-03-21', TRUE);
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES
('4', '1234567890123459', '2021-01-14', FALSE);
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES
('5', '1234567890123450', '2021-06-9', TRUE);
```

```
USE Bank;
INSERT INTO Loan_type (id, type, description, base_amount,
base_interest_rate) VALUES ('1', 'Mortgages loans', 'description1',
10000, 15);
INSERT INTO Loan_type (id, type, description, base_amount,
base_interest_rate) VALUES ('2', 'Car loans', 'description2', 5000,
20);
INSERT INTO Loan_type (id, type, description, base_amount,
base_interest_rate) VALUES ('3', 'Appliance loans', 'description3',
3000, 25);
INSERT INTO Loan_type (id, type, description, base_amount,
base_interest_rate) VALUES ('4', 'Payday loans', 'description4',
1000, 50);
INSERT INTO Loan_type (id, type, description, base_amount,
base_interest_rate) VALUES ('5', 'Small Business loans',
'description5', 7000, 35);
```

```
USE Bank;
INSERT INTO Customer (id, branch_id, first_name, last_name,
date_of_birth, gender) VALUES ('1', '1', 'Paul', 'Panaitescu',
'1996-10-7', 'male');
INSERT INTO Customer (id, branch_id, first_name, last_name,
date_of_birth, gender) VALUES ('2', '3', 'Constantin', 'Tarau',
'1998-09-15', 'male');
INSERT INTO Customer (id, branch_id, first_name, last_name,
date_of_birth, gender) VALUES ('3', '1', 'Marius', 'Munteanu',
'1998-07-31', 'male');
INSERT INTO Customer (id, branch_id, first_name, last_name,
date_of_birth, gender) VALUES ('4', '2', 'Dragos', 'Mocanasu',
'1998-12-31', 'female');
```

```
INSERT INTO Customer (id, branch_id, first_name, last_name,
date_of_birth, gender) VALUES ('5', '2', 'Daenerys', 'Targaryen',
'1895-10-7', 'female');
```

```
USE Bank;
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('1',
'1', '1', '1000');
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('2',
'2', '2', '100');
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('3',
'3', '3', '200');
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('4',
'5', '4', '50000');
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('5',
'5', '5', '1000000');
```

```
USE Bank;
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid,
start_date, due_date) VALUES ('1', '1', '3', '0', '2020-05-18',
'2023-05-18');
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid,
start_date, due_date) VALUES ('2', '5', '1', '0', '2019-08-12',
'2021-05-25');
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid,
start_date, due_date) VALUES ('3', '4', '2', '100', '2019-05-13',
'2024-05-14');
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid,
start_date, due_date) VALUES ('4', '2', '5', '1000', '2018-05-25',
'2021-05-21');
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid,
start_date, due_date) VALUES ('5', '1', '4', '5000', '2020-05-20',
'2023-05-07');
```

```
USE Bank;
INSERT INTO Transaction (id, account_id, description, amount, date)
VALUES ('1', '1', 'description 100', '1000.90', '2020-05-18');
INSERT INTO Transaction (id, account_id, description, amount, date)
VALUES ('2', '2', 'description 200', '500.80', '2019-12-07');
INSERT INTO Transaction (id, account_id, description, amount, date)
VALUES ('3', '5', 'description 300', '100.90', '2018-06-30');
```

```
INSERT INTO Transaction (id, account_id, description, amount, date)
VALUES ('4', '5', 'description 400', '5060.7', '2020-01-24');
INSERT INTO Transaction (id, account_id, description, amount, date)
VALUES ('5', '5', 'description 500', '500.67', '2018-01-24');
```

Ensure that every account contains a required minimum amount of money at any time.

```
USE Bank;
delimiter //
CREATE TRIGGER bal_limit_insert BEFORE INSERT ON Account FOR EACH ROW
    BEGIN
           DECLARE message varchar(50);
           IF NEW.balance < 100 THEN
                SET message= CONCAT('Insertion error: new balance too
low: ', NEW.balance);
                SIGNAL SQLSTATE '46000'
            SET MESSAGE_TEXT = message;
           END IF;
     END;
//

CREATE TRIGGER bal_limit_update BEFORE UPDATE ON Account FOR EACH ROW
    BEGIN
           DECLARE message varchar(50);
           IF NEW.balance < 100 THEN
                SET message= CONCAT('Update error: new balance too
low: ', NEW.balance);
                SIGNAL SQLSTATE '46000'
            SET MESSAGE_TEXT = message;
           END IF;
     END;
//
delimiter ;
```

Exercises:

1. List of customers that have accounts in two or more branches of the bank at the same time.

```
USE Bank;

SELECT c.first_name, c.last_name
      FROM Customer c
      WHERE c.id IN (SELECT customer_id
            FROM Customer_Branch cb
         GROUP BY customer_id
         HAVING COUNT(*) >= 2);
```

---

2. Statement showing who takes loans more often – men or women.

```
USE Bank;
SELECT gender, COUNT(*) AS count
FROM Customer AS c
WHERE c.id IN (
      SELECT customer_id
     FROM Account AS a
     WHERE a.id IN (
            SELECT account_id
            FROM Loan AS l))
GROUP BY gender
ORDER BY count DESC;
```

---

3. At the end of every year, a statement of all movements is generated for each account.

```
CREATE EVENT IF NOT EXISTS Account_transactions_every_year
ON SCHEDULE AT '2020-12-31' + INTERVAL 1 year
      DO SELECT *
      FROM Transaction t
```

---

4. List of customers that have never had a loan

```
USE Bank;
SELECT c.first_name, c.last_name
      FROM Customer c
      WHERE c.id IN (SELECT a.customer_id
            FROM Account a
            WHERE a.id NOT IN (SELECT l.account_id
                        FROM Loan l));
```

5. Custom: Find customers who have no open accounts.

```
USE Bank;
SELECT c.first_name, c.last_name
      FROM Customer c
      WHERE c.id NOT IN (SELECT customer_id
            FROM Account cb
         GROUP BY customer_id);
```

V. Screenshots of the database

Branch

#	id	name	address
1	1	Albertslund Bank	Albertslund
2	2	Norrebro Bank	Albertslund
3	3	Kolding Bank	Kolding, Jutland
4	4	Glostrup Bank	Glostrup
5	5	Valby Bank	Valby
*	NULL	NULL	NULL

Customer

#	id	branch_id	first_name	last_name	date_of_birth	gende
1	1	1	Paul	Panaitescu	1996-10-07	male
2	2	3	Constantin	Tarau	1998-09-15	male
3	3	1	Marius	Munteanu	1998-07-31	male
4	4	2	Dragos	Mocanasu	1998-12-31	female
5	5	2	Daenerys	Targaryen	1895-10-07	female
*	NULL	NULL	NULL	NULL	NULL	NULL

Account

#	id	customer_id	card_id	balance
1	1	1	1	1000
2	2	2	2	100
3	3	3	3	200
4	4	5	4	50000
5	5	5	5	1000000
*	NULL	NULL	NULL	NULL

Card

#	id	number	expiration_date	is_blocked
1	1	1234567890123456	2021-01-30	1
2	2	1234567890123457	2022-08-20	1
3	3	1234567890123458	2023-03-21	1
4	4	1234567890123459	2021-01-14	0
5	5	1234567890123450	2021-06-09	1
*	NULL	NULL	NULL	NULL

Transaction

#	id	account_id	description	amount	date
1	1	1	description 100	1000.900	2020-05-18
2	2	2	description 200	500.800	2019-12-07
3	3	5	description 300	100.900	2018-06-30
4	4	5	description 400	5060.700	2020-01-24
5	5	5	description 500	500.670	2018-01-24
*	NULL	NULL	NULL	NULL	NULL

Loan

#	id	account_id	loan_type_id	amount_paid	start_date	due_date
1	1	1	3	0.000	2020-05-18	2023-05-18
2	2	5	1	0.000	2019-08-12	2021-05-25
3	3	4	2	100.000	2019-05-13	2024-05-14
4	4	2	5	1000.000	2018-05-25	2021-05-21
5	5	1	4	5000.000	2020-05-20	2023-05-07
*	NULL	NULL	NULL	NULL	NULL	NULL

Customer_branch

	id	customer_id	branch_id
▶	1	1	1
	2	2	1
	3	3	2
	4	2	3
	5	5	2

Loan_type

#	id	type	description	base_amount	base_interest_rate
1	1	Mortgages loans	description1	10000.000	15.000
2	2	Car loans	description2	5000.000	20.000
3	3	Appliance loans	description3	3000.000	25.000
4	4	Payday loans	description4	1000.000	50.000
5	5	Small Business lo…	description5	7000.000	35.000
*	NULL	NULL	NULL	NULL	NULL