ASSIGNMENT FOR BANKING SYSTEM

Step-by-Step Guide for Your Banking System Database Project

Let's go through each part of your project step by step.

Step 1: Plan Your Database Schema

Understand the entities and their relationships:

- **Customer** has multiple **Accounts**.
- **Account** has multiple **Transactions**.
- **Employee** works at one **Branch**.
- **Branch** has multiple **Employees**.

Step 2: Design the Schema

Define each table with its attributes and relationships:

1. **Customer**:
   - `CustomerID` INT, Primary Key, Auto Increment
   - `FirstName` VARCHAR(50), NOT NULL
   - `LastName` VARCHAR(50), NOT NULL
   - `DateOfBirth` DATE, NOT NULL
   - `Address` VARCHAR(255), NOT NULL
   - `Phone` VARCHAR(15), NOT NULL
   - `Email` VARCHAR(100), NOT NULL
2. **Account**:
   - `AccountID` INT, Primary Key, Auto Increment
   - `AccountNumber` VARCHAR(20), NOT NULL, UNIQUE
   - `CustomerID` INT, Foreign Key
   - `AccountType` ENUM('Checking', 'Savings'), NOT NULL
   - `Balance` DECIMAL(15, 2), NOT NULL
   - `OpenDate` DATE, NOT NULL
3. **Transaction**:
   - `TransactionID` INT, Primary Key, Auto Increment
   - `AccountID` INT, Foreign Key
   - `TransactionType` ENUM('Deposit', 'Withdrawal', 'Transfer'), NOT NULL
   - `Amount` DECIMAL(15, 2), NOT NULL
   - `Date` DATE, NOT NULL
   - `Description` VARCHAR(255)
4. **Employee**:
   - `EmployeeID` INT, Primary Key, Auto Increment
   - `FirstName` VARCHAR(50), NOT NULL
   - `LastName` VARCHAR(50), NOT NULL
   - `Position` VARCHAR(50), NOT NULL
   - `BranchID` INT, Foreign Key

   - `HireDate` DATE, NOT NULL
5. **Branch**:
   - `BranchID` INT, Primary Key, Auto Increment
   - `BranchName` VARCHAR(100), NOT NULL
   - `Location` VARCHAR(255), NOT NULL

Step 3: Implement the Schema in MySQL

Use the following SQL script to create the tables:

```
Enter password: **********

Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 25

Server version: 8.0.37 MySQL Community Server - GPL



Copyright (c) 2000, 2024, Oracle and/or its affiliates.



Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.



Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.



mysql> CREATE DATABASE Bankingsystem;

ERROR 1007 (HY000): Can't create database 'bankingsystem'; database exists

mysql> USE Bankingsystem;

Database changed

mysql> -- Create Customer table

mysql> CREATE TABLE Customer (

    ->     CustomerID INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each customer
```

ASSIGNMENT FOR BANKING SYSTEM

```
    ->     FirstName VARCHAR(50) NOT NULL,           -- Customer's first name
    ->     LastName VARCHAR(50) NOT NULL,            -- Customer's last name
    ->     DateOfBirth DATE NOT NULL,                -- Customer's date of birth
    ->     Address VARCHAR(255),                     -- Customer's address
    ->     Phone VARCHAR(15),                        -- Customer's phone number
    ->     Email VARCHAR(50)                         -- Customer's email address
    -> );
ERROR 1050 (42S01): Table 'customer' already exists
mysql>
mysql> -- Insert sample data into Customer table
mysql> INSERT INTO Customer (FirstName, LastName, DateOfBirth, Address, Phone, Email) VALUES
    -> ('John', 'Doe', '1980-01-01', '123 Main St', '1234567890', 'john.doe@example.com'),
    -> ('Jane', 'Smith', '1990-02-02', '456 Oak St', '2345678901', 'jane.smith@example.com'),
    -> ('Jim', 'Beam', '1975-03-03', '789 Pine St', '3456789012', 'jim.beam@example.com'),
    -> ('Jack', 'Daniels', '1985-04-04', '101 Maple St', '4567890123', 'jack.daniels@example.com'),
    -> ('Jill', 'Valentine', '1995-05-05', '202 Elm St', '5678901234', 'jill.valentine@example.com');
Query OK, 5 rows affected (0.13 sec)
Records: 5  Duplicates: 0  Warnings: 0


mysql> SELECT * FROM Customer;
+------------+-----------+-----------+-------------+--------------+------------+----------------------------+
| CustomerID | FirstName | LastName  | DateOfBirth | Address      | Phone      | Email                      |
+------------+-----------+-----------+-------------+--------------+------------+----------------------------+
|          1 | John      | Doe       | 1980-01-01  | 123 Main St  | 555-1234   | john.doe@example.com       |
|          2 | Jane      | Smith     | 1990-02-02  | 456 Oak St   | 555-5678   | jane.smith@example.com     |
```

ASSIGNMENT FOR BANKING SYSTEM

```
|          3 | Alice      | Johnson    | 1975-03-03  | 789 Pine St  | 555-9012    | alice.johnson@example.com  |
|          4 | Bob        | Brown      | 1985-04-04  | 101 Maple St | 555-3456    | bob.brown@example.com      |
|          5 | Carol      | Davis      | 1995-05-05  | 202 Birch St | 555-7890    | carol.davis@example.com    |
|          6 | John       | Doe        | 1980-01-01  | 123 Main St  | 1234567890 | john.doe@example.com       |
|          7 | Jane       | Smith      | 1990-02-02  | 456 Oak St   | 2345678901 | jane.smith@example.com     |
|          8 | Jim        | Beam       | 1975-03-03  | 789 Pine St  | 3456789012 | jim.beam@example.com       |
|          9 | Jack       | Daniels    | 1985-04-04  | 101 Maple St | 4567890123 | jack.daniels@example.com   |
|         10 | Jill       | Valentine | 1995-05-05  | 202 Elm St   | 5678901234 | jill.valentine@example.com |
+------------+-----------+-----------+-------------+--------------+------------+---------------------------+
10 rows in set (0.00 sec)
```

mysql> -- Create Account table

```
mysql> CREATE TABLE Account (
    ->     AccountID INT AUTO_INCREMENT PRIMARY KEY,  -- Unique identifier for each account
    ->     AccountNumber VARCHAR(20) NOT NULL,        -- Account number
    ->     CustomerID INT,                            -- Foreign key referencing Customer table
    ->     AccountType ENUM('Checking', 'Savings') NOT NULL,  -- Type of account (Checking or Savings)
    ->     Balance DECIMAL(10, 2) NOT NULL,           -- Current balance of the account
    ->     OpenDate DATE NOT NULL                     -- Date when the account was opened
    -> );
ERROR 1050 (42S01): Table 'account' already exists
mysql>
```

mysql> -- **Insert sample data into Account table**

```
mysql> INSERT INTO Account (AccountNumber, CustomerID, AccountType, Balance, OpenDate) VALUES
    -> ('1001', 1, 'Checking', 15000.00, '2020-01-01'),
```

ASSIGNMENT FOR BANKING SYSTEM

```
    -> ('1002', 1, 'Savings', 5000.00, '2020-02-01'),
    -> ('1003', 2, 'Checking', 20000.00, '2020-03-01'),
    -> ('1004', 3, 'Savings', 8000.00, '2020-04-01'),
    -> ('1005', 4, 'Checking', 12000.00, '2020-05-01'),
    -> ('1006', 5, 'Savings', 3000.00, '2020-06-01'),
    -> ('1007', 5, 'Checking', 6000.00, '2020-07-01'),
    -> ('1008', 4, 'Savings', 2500.00, '2020-08-01'),
    -> ('1009', 3, 'Checking', 7000.00, '2020-09-01'),
    -> ('1010', 2, 'Savings', 9000.00, '2020-10-01');
Query OK, 10 rows affected (0.12 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Account;
+-----------+---------------+------------+-------------+----------+------------+
| AccountID | AccountNumber | CustomerID | AccountType | Balance  | OpenDate   |
+-----------+---------------+------------+-------------+----------+------------+
|         1 | A12345        |          1 | Checking    |  1000.00 | 2023-01-01 |
|         2 | A12346        |          1 | Savings     |  5000.00 | 2023-02-01 |
|         3 | A12347        |          2 | Checking    |  1500.00 | 2023-03-01 |
|         4 | A12348        |          2 | Savings     |  2500.00 | 2023-04-01 |
|         5 | A12349        |          3 | Checking    |  3000.00 | 2023-05-01 |
|         6 | A12350        |          4 | Savings     |  4000.00 | 2023-06-01 |
|         7 | A12351        |          5 | Checking    |  2000.00 | 2023-07-01 |
|         8 | A12352        |          5 | Savings     |  6000.00 | 2023-08-01 |
|         9 | A12353        |          3 | Savings     |  3500.00 | 2023-09-01 |
```

ASSIGNMENT FOR BANKING SYSTEM

```
|        10 | A12354         |          4 | Checking     |  4500.00 | 2023-10-01 |
|        11 | 1001           |          1 | Checking     | 15000.00 | 2020-01-01 |
|        12 | 1002           |          1 | Savings      |  5000.00 | 2020-02-01 |
|        13 | 1003           |          2 | Checking     | 20000.00 | 2020-03-01 |
|        14 | 1004           |          3 | Savings      |  8000.00 | 2020-04-01 |
|        15 | 1005           |          4 | Checking     | 12000.00 | 2020-05-01 |
|        16 | 1006           |          5 | Savings      |  3000.00 | 2020-06-01 |
|        17 | 1007           |          5 | Checking     |  6000.00 | 2020-07-01 |
|        18 | 1008           |          4 | Savings      |  2500.00 | 2020-08-01 |
|        19 | 1009           |          3 | Checking     |  7000.00 | 2020-09-01 |
|        20 | 1010           |          2 | Savings      |  9000.00 | 2020-10-01 |
+-----------+---------------+------------+-------------+----------+------------+
20 rows in set (0.04 sec)
```

mysql> -- Create Transaction table

```
mysql> CREATE TABLE Transaction (
    ->     TransactionID INT AUTO_INCREMENT PRIMARY KEY,  -- Unique identifier for each transaction
    ->     AccountID INT,                          -- Foreign key referencing Account table
    ->     TransactionType ENUM('Deposit', 'Withdrawal', 'Transfer') NOT NULL,  -- Type of transaction
    ->     Amount DECIMAL(10, 2) NOT NULL,             -- Amount of the transaction
    ->     Date DATE NOT NULL,                         -- Date of the transaction
    ->     Description VARCHAR(255)                    -- Description of the transaction
    -> );
ERROR 1050 (42S01): Table 'transaction' already exists
mysql>
```

ASSIGNMENT FOR BANKING SYSTEM

mysql> **-- Insert sample data into Transaction table**

```
mysql> INSERT INTO Transaction (AccountID, TransactionType, Amount, Date, Description) VALUES
    -> (1, 'Deposit', 500.00, '2020-01-02', 'Initial deposit'),
    -> (1, 'Withdrawal', 200.00, '2020-01-03', 'ATM withdrawal'),
    -> (2, 'Deposit', 1000.00, '2020-02-02', 'Salary deposit'),
    -> (2, 'Withdrawal', 500.00, '2020-02-03', 'ATM withdrawal'),
    -> (3, 'Deposit', 1500.00, '2020-03-02', 'Salary deposit'),
    -> (3, 'Withdrawal', 700.00, '2020-03-03', 'ATM withdrawal'),
    -> (4, 'Deposit', 2000.00, '2020-04-02', 'Salary deposit'),
    -> (4, 'Withdrawal', 1000.00, '2020-04-03', 'ATM withdrawal'),
    -> (5, 'Deposit', 1200.00, '2020-05-02', 'Salary deposit'),
    -> (5, 'Withdrawal', 600.00, '2020-05-03', 'ATM withdrawal'),
    -> (6, 'Deposit', 300.00, '2020-06-02', 'Initial deposit'),
    -> (6, 'Withdrawal', 100.00, '2020-06-03', 'ATM withdrawal'),
    -> (7, 'Deposit', 600.00, '2020-07-02', 'Initial deposit'),
    -> (7, 'Withdrawal', 200.00, '2020-07-03', 'ATM withdrawal'),
    -> (8, 'Deposit', 250.00, '2020-08-02', 'Initial deposit'),
    -> (8, 'Withdrawal', 100.00, '2020-08-03', 'ATM withdrawal'),
    -> (9, 'Deposit', 700.00, '2020-09-02', 'Initial deposit'),
    -> (9, 'Withdrawal', 300.00, '2020-09-03', 'ATM withdrawal'),
    -> (10, 'Deposit', 900.00, '2020-10-02', 'Initial deposit'),
    -> (10, 'Withdrawal', 400.00, '2020-10-03', 'ATM withdrawal');
Query OK, 20 rows affected (0.14 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

ASSIGNMENT FOR BANKING SYSTEM

```
mysql> SELECT * FROM Transaction;
+---------------+-----------+-----------------+---------+------------+-----------------------+
| TransactionID | AccountID | TransactionType | Amount  | Date       | Description           |
+---------------+-----------+-----------------+---------+------------+-----------------------+
|             1 |         1 | Deposit         |  500.00 | 2023-01-05 | Initial deposit       |
|             2 |         2 | Withdrawal      |  200.00 | 2023-02-10 | ATM withdrawal        |
|             3 |         3 | Deposit         | 1000.00 | 2023-03-15 | Salary deposit        |
|             4 |         4 | Transfer        |  300.00 | 2023-04-20 | Transfer to savings   |
|             5 |         5 | Withdrawal      |  150.00 | 2023-05-25 | Grocery shopping      |
|             6 |         6 | Deposit         |  700.00 | 2023-06-30 | Freelance payment     |
|             7 |         7 | Transfer        |  400.00 | 2023-07-05 | Transfer to checking  |
|             8 |         8 | Deposit         |  800.00 | 2023-08-10 | Bonus deposit         |
|             9 |         9 | Withdrawal      |  500.00 | 2023-09-15 | Bill payment          |
|            10 |        10 | Transfer        |  600.00 | 2023-10-20 | Transfer to savings   |
|            11 |         1 | Withdrawal      |  100.00 | 2023-01-15 | Online shopping       |
|            12 |         2 | Deposit         |  300.00 | 2023-02-20 | Side job payment      |
|            13 |         3 | Withdrawal      |  200.00 | 2023-03-25 | Restaurant bill       |
|            14 |         4 | Deposit         |  500.00 | 2023-04-30 | Gift from friend      |
|            15 |         5 | Transfer        |  400.00 | 2023-05-05 | Transfer to checking  |
|            16 |         6 | Withdrawal      |  600.00 | 2023-06-10 | Home improvement      |
|            17 |         7 | Deposit         |  700.00 | 2023-07-15 | Part-time job payment |
|            18 |         8 | Transfer        |  200.00 | 2023-08-20 | Transfer to savings   |
|            19 |         9 | Deposit         |  900.00 | 2023-09-25 | Investment return     |
|            20 |        10 | Withdrawal      |  800.00 | 2023-10-30 | Vacation expenses     |
|            21 |         1 | Deposit         |  500.00 | 2020-01-02 | Initial deposit       |
|            22 |         1 | Withdrawal      |  200.00 | 2020-01-03 | ATM withdrawal        |
|            23 |         2 | Deposit         | 1000.00 | 2020-02-02 | Salary deposit        |
|            24 |         2 | Withdrawal      |  500.00 | 2020-02-03 | ATM withdrawal        |
|            25 |         3 | Deposit         | 1500.00 | 2020-03-02 | Salary deposit        |
|            26 |         3 | Withdrawal      |  700.00 | 2020-03-03 | ATM withdrawal        |
|            27 |         4 | Deposit         | 2000.00 | 2020-04-02 | Salary deposit        |
|            28 |         4 | Withdrawal      | 1000.00 | 2020-04-03 | ATM withdrawal        |
|            29 |         5 | Deposit         | 1200.00 | 2020-05-02 | Salary deposit        |
|            30 |         5 | Withdrawal      |  600.00 | 2020-05-03 | ATM withdrawal        |
|            31 |         6 | Deposit         |  300.00 | 2020-06-02 | Initial deposit       |
|            32 |         6 | Withdrawal      |  100.00 | 2020-06-03 | ATM withdrawal        |
|            33 |         7 | Deposit         |  600.00 | 2020-07-02 | Initial deposit       |
|            34 |         7 | Withdrawal      |  200.00 | 2020-07-03 | ATM withdrawal        |
|            35 |         8 | Deposit         |  250.00 | 2020-08-02 | Initial deposit       |
|            36 |         8 | Withdrawal      |  100.00 | 2020-08-03 | ATM withdrawal        |
|            37 |         9 | Deposit         |  700.00 | 2020-09-02 | Initial deposit       |
|            38 |         9 | Withdrawal      |  300.00 | 2020-09-03 | ATM withdrawal        |
|            39 |        10 | Deposit         |  900.00 | 2020-10-02 | Initial deposit       |
|            40 |        10 | Withdrawal      |  400.00 | 2020-10-03 | ATM withdrawal        |
+---------------+-----------+-----------------+---------+------------+-----------------------+
40 rows in set (0.00 sec)
```

-- **Employee table**

```
CREATE TABLE Employee (    EmployeeID INT PRIMARY KEY,    FirstName VARCHAR(50) NOT NULL,
LastName VARCHAR(50) NOT NULL,    Position VARCHAR(50),    BranchID INT,    HireDate DATE,
FOREIGN KEY (BranchID) REFERENCES Branch(BranchID));
```

```
mysql> --Sample data for Employee
```

ASSIGNMENT FOR BANKING SYSTEM

```
mysql> INSERT INTO Employee (EmployeeID, FirstName, LastName, Position, BranchID, HireDate)
    -> VALUES
    ->     (1, 'Michael', 'Smith', 'Manager', 1, '2010-08-01'),
    ->     (2, 'Emily', 'Johnson', 'Teller', 2, '2015-04-15'),
    ->     (3, 'David', 'Brown', 'Accountant', 1, '2018-02-20'),
    ->     (4, 'Sarah', 'Davis', 'Loan Officer', 2, '2017-09-10'),
    ->     (5, 'James', 'Wilson', 'Financial Advisor', 1, '2019-11-25');
Query OK, 5 rows affected (0.13 sec)
Records: 5  Duplicates: 0  Warnings: 0


mysql> SELECT * FROM Employee;
+------------+-----------+----------+-------------------+----------+------------+
| EmployeeID | FirstName | LastName | Position          | BranchID | HireDate   |
+------------+-----------+----------+-------------------+----------+------------+
|          1 | Michael   | Smith    | Manager           |        1 | 2010-08-01 |
|          2 | Emily     | Johnson  | Teller            |        2 | 2015-04-15 |
|          3 | David     | Brown    | Accountant        |        1 | 2018-02-20 |
|          4 | Sarah     | Davis    | Loan Officer      |        2 | 2017-09-10 |
|          5 | James     | Wilson   | Financial Advisor |        1 | 2019-11-25 |
+------------+-----------+----------+-------------------+----------+------------+
5 rows in set (0.00 sec)
mysql> -- Branch table
mysql> CREATE TABLE Branch (
    ->     BranchID INT PRIMARY KEY,
    ->     BranchName VARCHAR(100),
    ->     Location VARCHAR(255)
    -> );
mysql> -- Insert sample data into Branch table
mysql> INSERT INTO Branch (BranchID, BranchName, Location)
    -> VALUES
    ->     (1, 'Main Branch', '123 Center St, Citytown'),
    ->     (2, 'Downtown Branch', '456 Elm St, Othertown');
ERROR 1062 (23000): Duplicate entry '1' for key 'branch.PRIMARY'
mysql> SELECT * FROM Branch;
+----------+--------------+-------------+
| BranchID | BranchName   | Location    |
+----------+--------------+-------------+
|        1 | Main Branch  | 123 Main St |
|        2 | North Branch | 456 Oak St  |
+----------+--------------+-------------+
2 rows in set (0.00 sec)
```

SQL Queries:

1. **Retrieve all customers who have a balance greater than $10,000.**

```
mysql> -- Retrieve all customers who have a balance greater than $10,000
mysql> SELECT c.CustomerID, c.FirstName, c.LastName, SUM(a.Balance) AS TotalBalance
    -> FROM Customer c
    -> JOIN Account a ON c.CustomerID = a.CustomerID
    -> GROUP BY c.CustomerID, c.FirstName, c.LastName
    -> HAVING TotalBalance > 10000.00;
+------------+-----------+----------+--------------+
| CustomerID | FirstName | LastName | TotalBalance |
+------------+-----------+----------+--------------+
|          1 | John      | Doe      |     26000.00 |
|          2 | Jane      | Smith    |     33000.00 |
|          3 | Alice     | Johnson  |     21500.00 |
|          4 | Bob       | Brown    |     23000.00 |
|          5 | Carol     | Davis    |     17000.00 |
+------------+-----------+----------+--------------+
5 rows in set (0.07 sec)
```

2. **List all transactions for a specific account.**

```
mysql> -- List all transactions for a specific account
mysql> SELECT TransactionID, TransactionType, Amount, Date, Description
    -> FROM Transaction
    -> WHERE AccountID = 1; -- Replace 1 with the specific AccountID you want to query
+---------------+-----------------+--------+------------+-----------------+
| TransactionID | TransactionType | Amount | Date       | Description     |
+---------------+-----------------+--------+------------+-----------------+
|             1 | Deposit         | 500.00 | 2023-01-05 | Initial deposit |
|            11 | Withdrawal      | 100.00 | 2023-01-15 | Online shopping |
|            21 | Deposit         | 500.00 | 2020-01-02 | Initial deposit |
|            22 | Withdrawal      | 200.00 | 2020-01-03 | ATM withdrawal  |
+---------------+-----------------+--------+------------+-----------------+
4 rows in set (0.05 sec)
```

3.Find the total number of accounts for each branch.

```
-- Find the total number of accounts for each branch
-- Find the total number of accounts for each branch
mysql> SELECT b.BranchID, b.BranchName, COUNT(a.AccountID) AS TotalAccounts
    -> FROM Branch b
    -> LEFT JOIN Employee e ON b.BranchID = e.BranchID
    -> LEFT JOIN Account a ON e.EmployeeID = a.CustomerID -- Assuming Account is linked
directly to CustomerID
    -> GROUP BY b.BranchID, b.BranchName;
+----------+--------------+---------------+
| BranchID | BranchName   | TotalAccounts |
+----------+--------------+---------------+
|        1 | Main Branch  |            12 |
|        2 | North Branch |             8 |
+----------+--------------+---------------+
2 rows in set (0.04 sec)
```

4. **calculate total balance of each customer**

```
mysql> -- Calculate the total balance for each customer
mysql> SELECT c.CustomerID, c.FirstName, c.LastName, SUM(a.Balance) AS TotalBalance
    -> FROM Customer c
    -> LEFT JOIN Account a ON c.CustomerID = a.CustomerID
    -> GROUP BY c.CustomerID, c.FirstName, c.LastName;
+------------+-----------+-----------+--------------+
```

ASSIGNMENT FOR BANKING SYSTEM

```
| CustomerID | FirstName | LastName  | TotalBalance |
+------------+-----------+-----------+--------------+
|          1 | John      | Doe       |     26000.00 |
|          2 | Jane      | Smith     |     33000.00 |
|          3 | Alice     | Johnson   |     21500.00 |
|          4 | Bob       | Brown     |     23000.00 |
|          5 | Carol     | Davis     |     17000.00 |
|          6 | John      | Doe       |         NULL |
|          7 | Jane      | Smith     |         NULL |
|          8 | Jim       | Beam      |         NULL |
|          9 | Jack      | Daniels   |         NULL |
|         10 | Jill      | Valentine |         NULL |
+------------+-----------+-----------+--------------+
10 rows in set (0.00 sec)
```

5.list all employees who have been working for more than 5 years.

```
mysql> -- List all employees who have been working for more than 5 years
mysql> SELECT EmployeeID, FirstName, LastName, Position, BranchID, HireDate
    -> FROM Employee
    -> WHERE DATEDIFF(NOW(), HireDate) > 1825; -- 1825 days = 5 years
+------------+-----------+----------+--------------+----------+------------+
| EmployeeID | FirstName | LastName | Position     | BranchID | HireDate   |
+------------+-----------+----------+--------------+----------+------------+
|          1 | Michael   | Smith    | Manager      |        1 | 2010-08-01 |
|          2 | Emily     | Johnson  | Teller       |        2 | 2015-04-15 |
|          3 | David     | Brown    | Accountant   |        1 | 2018-02-20 |
|          4 | Sarah     | Davis    | Loan Officer |        2 | 2017-09-10 |
+------------+-----------+----------+--------------+----------+------------+
4 rows in set (0.07 sec)
```

6. **Find the branch with the highest number**

```
mysql> -- Find the branch with the highest number of employees
mysql> SELECT b.BranchID, b.BranchName, COUNT(e.EmployeeID) AS EmployeeCount
    -> FROM Branch b
    -> LEFT JOIN Employee e ON b.BranchID = e.BranchID
    -> GROUP BY b.BranchID, b.BranchName
    -> ORDER BY EmployeeCount DESC
    -> LIMIT 1;
+----------+-------------+---------------+
| BranchID | BranchName  | EmployeeCount |
+----------+-------------+---------------+
|        1 | Main Branch |             3 |
+----------+-------------+---------------+
1 row in set (0.00 sec)
```

ASSIGNMENT FOR BANKING SYSTEM

Advanced operations

```
mysql> -- Create stored procedure for money transfer
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE TransferMoney(
    ->     IN fromAccount INT,
    ->     IN toAccount INT,
    ->     IN amount DECIMAL(10,2)
    -> )
    -> BEGIN
    ->     DECLARE currentBalance DECIMAL(10,2);
    ->
    ->     -- Check if there's enough balance in the fromAccount
    ->     SELECT Balance INTO currentBalance
    ->     FROM Account
    ->     WHERE AccountID = fromAccount;
    ->
    ->     IF currentBalance >= amount THEN
    ->         -- Deduct amount from fromAccount
    ->         UPDATE Account
    ->         SET Balance = Balance - amount
    ->         WHERE AccountID = fromAccount;
    ->
    ->         -- Add amount to toAccount
    ->         UPDATE Account
    ->         SET Balance = Balance + amount
    ->         WHERE AccountID = toAccount;
    ->
    ->         -- Insert transaction record
    ->         INSERT INTO Transaction (AccountID, TransactionType, Amount, Date, Description)
    ->         VALUES (fromAccount, 'Transfer', -amount, CURDATE(), CONCAT('Transfer to
AccountID ', toAccount));
    ->
    ->         INSERT INTO Transaction (AccountID, TransactionType, Amount, Date, Description)
    ->         VALUES (toAccount, 'Transfer', amount, CURDATE(), CONCAT('Transfer from AccountID
', fromAccount));
    ->
    ->         SELECT 'Transfer successful' AS Message;
    ->     ELSE
    ->         SELECT 'Insufficient balance' AS Message;
    ->     END IF;
```

```
  ->
  -> END //
Query OK, 0 rows affected (0.46 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL TransferMoney(1, 2, 500.00); -- Transfer $500 from AccountID 1 to AccountID 2
+---------------------+
| Message             |
+---------------------+
| Transfer successful |
+---------------------+
1 row in set (0.46 sec)

Query OK, 0 rows affected (0.50 sec)
```

After calling the stored procedure, you will receive a result set based on the conditions within the procedure:

If the transfer is successful (i.e., `currentBalance` is sufficient), it will output:

```
sql
Copy code
+---------------------+
| Message             |
+---------------------+
| Transfer successful |
+---------------------+
1 row in set (0.00 sec)
```

If the transfer fails due to insufficient balance, it will output:

```
sql
Copy code
+----------------------+
| Message              |
+----------------------+
| Insufficient balance |
+----------------------+
1 row in set (0.00 sec)
```