-- Creating the Customers table

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(15),

Address VARCHAR(255)

);

-- Creating the Branches table

CREATE TABLE Branches (

BranchID INT PRIMARY KEY,

BranchName VARCHAR(100),

Location VARCHAR(100)

);

-- Creating the Accounts table

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY,

CustomerID INT,

AccountType VARCHAR(50),

Balance DECIMAL(15, 2),

CreatedDate DATE,

BranchID INT, -- Added BranchID to link to Branches

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (BranchID) REFERENCES Branches(BranchID) -- Foreign key constraint linking to Branches

);

-- Creating the Transactions table

CREATE TABLE Transactions (

TransactionID INT PRIMARY KEY,

AccountID INT,

TransactionType VARCHAR(50),

Amount DECIMAL(15, 2),

TransactionDate DATETIME,

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID) -- Foreign key linking to Accounts

);

-- Creating the Employees table

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

BranchID INT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Role VARCHAR(50),

Salary DECIMAL(15, 2),

FOREIGN KEY (BranchID) REFERENCES Branches(BranchID) -- Foreign key linking to Branches

);

-- Inserting data into Customers table

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address)

VALUES

(1, 'John', 'Doe', 'john.doe@example.com', '1234567890', '123 Elm St'),

(2, 'Jane', 'Smith', 'jane.smith@example.com', '9876543210', '456 Oak St'),

(3, 'Michael', 'Brown', 'michael.brown@example.com', '5678901234', '789 Pine St'),

(4, 'Emily', 'Davis', 'emily.davis@example.com', '1357924680', '101 Maple St');

-- Inserting data into Branches table

INSERT INTO Branches (BranchID, BranchName, Location)

VALUES

(1, 'Downtown Branch', 'Downtown'),

(2, 'Uptown Branch', 'Uptown'),

(3, 'Eastside Branch', 'Eastside');

-- Inserting data into Accounts table

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, CreatedDate, BranchID)

VALUES

(1, 1, 'Savings', 5000.00, '2023-01-15', 1),

(2, 1, 'Checking', 2000.00, '2023-02-10', 1),

(3, 2, 'Savings', 10000.00, '2023-03-05', 2),

(4, 3, 'Savings', 7000.00, '2023-04-20', 3),

(5, 4, 'Checking', 3000.00, '2023-05-10', 1);

-- Inserting data into Transactions table

INSERT INTO Transactions (TransactionID, AccountID, TransactionType, Amount, TransactionDate)

VALUES

(1, 1, 'Deposit', 1000.00, '2023-01-20 10:00:00'),

(2, 1, 'Withdrawal', 500.00, '2023-01-25 14:30:00'),

(3, 2, 'Deposit', 2000.00, '2023-02-15 09:15:00'),

(4, 3, 'Withdrawal', 1000.00, '2023-04-25 16:45:00'),

(5, 4, 'Deposit', 1500.00, '2023-05-15 11:30:00'),

(6, 5, 'Withdrawal', 1000.00, '2023-06-01 08:45:00');

-- Inserting data into Employees table

INSERT INTO Employees (EmployeeID, BranchID, FirstName, LastName, Role, Salary)

VALUES

(1, 1, 'Alice', 'Johnson', 'Manager', 60000.00),

(2, 1, 'Bob', 'Williams', 'Teller', 40000.00),

(3, 2, 'Charlie', 'Miller', 'Manager', 65000.00),

(4, 3, 'David', 'Martinez', 'Teller', 35000.00),

(5, 3, 'Eva', 'Lopez', 'Manager', 70000.00);

USE BankDB;

**-- 1. List all customers and their accounts with balances.**

SELECT c.FirstName, c.LastName, a.AccountID, a.AccountType, a.Balance

FROM Customers c

JOIN Accounts a ON c.CustomerID = a.CustomerID;

**-- 2. List all employees who manage branches where the total account balances exceed $20,000.**

SELECT e.FirstName, e.LastName, e.Role, b.BranchName

FROM Employees e

JOIN Branches b ON e.BranchID = b.BranchID

JOIN Accounts a ON b.BranchID = a.BranchID

GROUP BY e.EmployeeID, b.BranchID

HAVING SUM(a.Balance) > 20000;

**-- 3. Identify accounts whose balance is higher than the average balance of accounts within their branch.**

SELECT a.AccountID, a.AccountType, a.Balance, a.BranchID

FROM Accounts a

JOIN Branches b ON a.BranchID = b.BranchID

WHERE a.Balance > (

SELECT AVG(a2.Balance)

FROM Accounts a2

WHERE a2.BranchID = a.BranchID

);

**-- 4. Find customers who have at least one transaction of more than $1,000.**

SELECT DISTINCT c.FirstName, c.LastName

FROM Customers c

JOIN Accounts a ON c.CustomerID = a.CustomerID

JOIN Transactions t ON a.AccountID = t.AccountID

WHERE t.Amount > 1000;

**-- 5. Get the total deposits and total withdrawals for each account, along with the account type.**

SELECT a.AccountID, a.AccountType,

SUM(CASE WHEN t.TransactionType = 'Deposit' THEN t.Amount ELSE 0 END) AS TotalDeposits,

SUM(CASE WHEN t.TransactionType = 'Withdrawal' THEN t.Amount ELSE 0 END) AS TotalWithdrawals

FROM Accounts a

JOIN Transactions t ON a.AccountID = t.AccountID

GROUP BY a.AccountID, a.AccountType;

**-- 6. Find pairs of customers who have accounts with the same account type and belong to the same branch.**

SELECT c1.FirstName AS Customer1\_FirstName, c1.LastName AS Customer1\_LastName,

c2.FirstName AS Customer2\_FirstName, c2.LastName AS Customer2\_LastName,

a1.AccountType

FROM Accounts a1

JOIN Accounts a2 ON a1.AccountType = a2.AccountType AND a1.BranchID = a2.BranchID

JOIN Customers c1 ON a1.CustomerID = c1.CustomerID

JOIN Customers c2 ON a2.CustomerID = c2.CustomerID

WHERE c1.CustomerID < c2.CustomerID; -- Ensures each pair is listed once

**-- 7. Find customers who do not have any transactions recorded.**

SELECT c.FirstName, c.LastName

FROM Customers c

LEFT JOIN Accounts a ON c.CustomerID = a.CustomerID

LEFT JOIN Transactions t ON a.AccountID = t.AccountID

WHERE t.TransactionID IS NULL;

**-- 8. Rank customers based on their total balance across all accounts.**

SELECT c.FirstName, c.LastName, SUM(a.Balance) AS TotalBalance,

RANK() OVER (ORDER BY SUM(a.Balance) DESC) AS RankCustomers

FROM Customers c

JOIN Accounts a ON c.CustomerID = a.CustomerID

GROUP BY c.CustomerID;

**-- 9. List employees whose salary is above the average salary of all employees in their branch.**

SELECT e.FirstName, e.LastName, e.Salary, e.Role, b.BranchName

FROM Employees e

JOIN Branches b ON e.BranchID = b.BranchID

WHERE e.Salary > (

SELECT AVG(e2.Salary)

FROM Employees e2

WHERE e2.BranchID = e.BranchID

);