Database Schema:

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

DOB DATE,

Balance DECIMAL(10,2),

LastModified DATETIME

);

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY,

CustomerID INT,

AccountType VARCHAR(20),

Balance DECIMAL(10,2),

LastModified DATETIME,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID INT PRIMARY KEY,

AccountID INT,

TransactionDate DATETIME,

Amount DECIMAL(10,2),

TransactionType VARCHAR(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID INT PRIMARY KEY,

CustomerID INT,

LoanAmount DECIMAL(10,2),

InterestRate DECIMAL(5,2),

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

Position VARCHAR(50),

Salary DECIMAL(10,2),

Department VARCHAR(50),

HireDate DATE

);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1, 'John Doe', '1985-05-15', 1000.00, NOW());

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (2, 'Jane Smith', '1990-07-20', 1500.00, NOW());

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 1000.00, NOW());

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 1500.00, NOW());

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

VALUES (1, 1, NOW(), 200.00, 'Deposit');

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

VALUES (2, 2, NOW(), 300.00, 'Withdrawal');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000.00, 5.00, CURDATE(), DATE\_ADD(CURDATE(), INTERVAL 60 MONTH));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Alice Johnson', 'Manager', 70000.00, 'HR', '2015-06-15');

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Bob Brown', 'Developer', 60000.00, 'IT', '2017-03-20');

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (3, 'Robert Wilson', '1960-02-10', 8000.00, NOW());

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (4, 'Emily Davis', '1955-11-23', 12000.00, NOW());

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (5, 'Michael Brown', '1995-03-12', 600.00, NOW());

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (3, 3, 'Savings', 8000.00, NOW());

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (4, 4, 'Savings', 12000.00, NOW());

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (5, 5, 'Checking', 0.00, NOW());

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

VALUES (3, 3, NOW(), 5500.00, 'Deposit');

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

VALUES (4, 4, NOW(), 250.00, 'Withdrawal');

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

VALUES (5, 5, NOW(), 50.00, 'Deposit');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 3, 10000.00, 6.5, CURDATE(), DATE\_ADD(CURDATE(), INTERVAL 36 MONTH));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (3, 4, 20000.00, 7.5, CURDATE(), DATE\_ADD(CURDATE(), INTERVAL 12 MONTH));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (4, 5, 3000.00, 5.0, CURDATE(), DATE\_ADD(CURDATE(), INTERVAL 6 MONTH));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (3, 'Carol Miller', 'Analyst', 45000.00, 'Finance', '2018-09-01');

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (4, 'David Lee', 'System Admin', 48000.00, 'IT', '2019-01-20');

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (5, 'Rachel Adams', 'HR Assistant', 40000.00, 'HR', '2020-07-15');

Exercise 1: Control Structures

Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60

years old.

o Question: Write a PL/SQL block that loops through all customers, checks their age, and if

they are above 60, apply a 1% discount to their current loan interest rates.

DELIMITER $$

CREATE PROCEDURE ApplyInterestDiscount()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE v\_customer\_id INT;

DECLARE v\_dob DATE;

DECLARE v\_age INT;

DECLARE cur CURSOR FOR SELECT CustomerID, DOB FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_customer\_id, v\_dob;

IF done THEN

LEAVE read\_loop;

END IF;

SET v\_age = TIMESTAMPDIFF(YEAR, v\_dob, CURDATE());

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = v\_customer\_id;

END IF;

END LOOP;

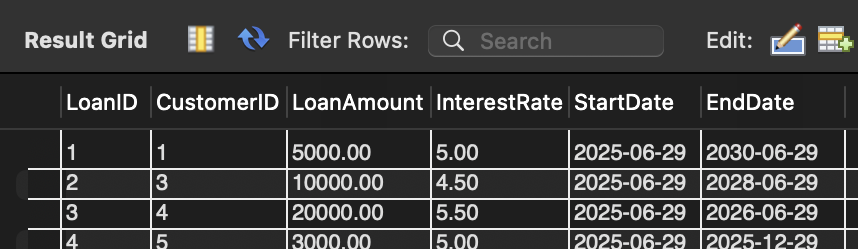
CLOSE cur;

END$$

DELIMITER ;

CALL ApplyInterestDiscount();

SELECT \* FROM Loans;

Output:

Scenario 2: A customer can be promoted to VIP status based on their balance.

o Question: Write a PL/SQL block that iterates through all customers and sets a flag IsVIP

to TRUE for those with a balance over $10,000.

DELIMITER $$

CREATE PROCEDURE PromoteToVIP()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE v\_customer\_id INT;

DECLARE v\_balance DECIMAL(10,2);

DECLARE cur CURSOR FOR SELECT CustomerID, Balance FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_customer\_id, v\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

IF v\_balance > 10000 THEN

UPDATE Customers

SET IsVIP = TRUE

WHERE CustomerID = v\_customer\_id;

END IF;

END LOOP;

CLOSE cur;

END$$

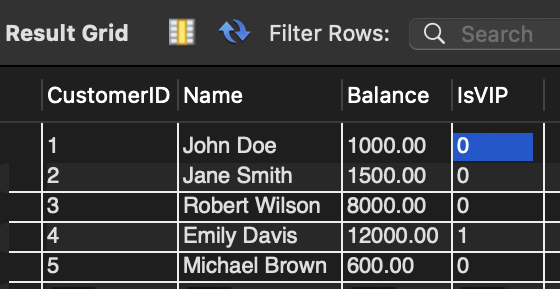
DELIMITER ;

ALTER TABLE Customers ADD COLUMN IsVIP BOOLEAN DEFAULT FALSE;

CALL PromoteToVIP();

SELECT CustomerID, Name, Balance, IsVIP

FROM Customers;



Output:

Scenario 3: The bank wants to send reminders to customers whose loans are due within the

next 30 days.

o Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a

reminder message for each customer.

DELIMITER $$

CREATE PROCEDURE SendLoanReminders()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE v\_loan\_id INT;

DECLARE v\_customer\_id INT;

DECLARE v\_due\_date DATE;

DECLARE cur CURSOR FOR

SELECT LoanID, CustomerID, EndDate

FROM Loans

WHERE EndDate BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_loan\_id, v\_customer\_id, v\_due\_date;

IF done THEN

LEAVE read\_loop;

END IF;

INSERT INTO LoanReminders (ReminderMessage)

VALUES (

CONCAT('Reminder: Customer ', v\_customer\_id,

', your loan ID ', v\_loan\_id,

' is due on ', DATE\_FORMAT(v\_due\_date, '%d-%b-%Y'))

);

END LOOP;

CLOSE cur;

-- Select all the messages at once

SELECT \* FROM LoanReminders;

END$$

DELIMITER ;

Exercise 3: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.

o Question: Write a stored procedure ProcessMonthlyInterest that calculates and updates

the balance of all savings accounts by applying an interest rate of 1% to the current

balance.

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

-- Apply 1% interest to all Savings accounts

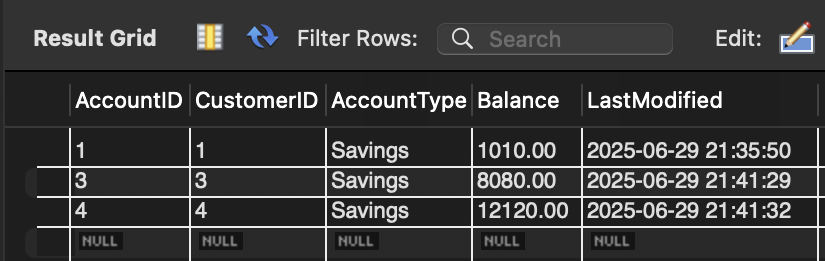
UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

END $$

DELIMITER ;

Output:

Scenario 2: The bank wants to implement a bonus scheme for employees based on their

performance.

o Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of

employees in a given department by adding a bonus percentage passed as a parameter.

DELIMITER $$

CREATE PROCEDURE UpdateEmployeeBonus(

IN deptName VARCHAR(50),

IN bonusPercent DECIMAL(5,2)

)

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonusPercent / 100)

WHERE Department = deptName;

END $$

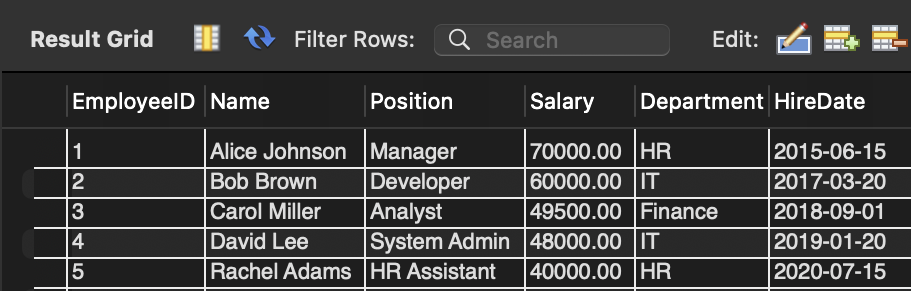
DELIMITER ;

SET SQL\_SAFE\_UPDATES = 0;

CALL UpdateEmployeeBonus('Finance', 10);

SET SQL\_SAFE\_UPDATES = 1;

SELECT \* FROM Employees;

Output:

Scenario 3: Customers should be able to transfer funds between their accounts.

Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

DELIMITER $$

CREATE PROCEDURE TransferFunds(

IN fromAccount INT,

IN toAccount INT,

IN amount DECIMAL(10,2)

)

BEGIN

DECLARE fromBalance DECIMAL(10,2);

SELECT Balance INTO fromBalance

FROM Accounts

WHERE AccountID = fromAccount;

IF fromBalance >= amount THEN

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = fromAccount;

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = toAccount;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient balance in source account';

END IF;

END $$

DELIMITER ;

SET SQL\_SAFE\_UPDATES = 0;

CALL TransferFunds(1, 2, 1000.00);

SET SQL\_SAFE\_UPDATES = 1;

SELECT \* FROM Accounts;

Output: