EXERCISE 1 AND EXERCISE 3

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    DOB DATE,

    Balance NUMBER,

    LastModified DATE,

    IsVIP CHAR(1) DEFAULT 'N'  -- For Scenario 2

);

CREATE TABLE Accounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    AccountType VARCHAR2(20),

    Balance NUMBER,

    LastModified DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

    TransactionID NUMBER PRIMARY KEY,

    AccountID NUMBER,

    TransactionDate DATE,

    Amount NUMBER,

    TransactionType VARCHAR2(10),

    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

    LoanID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    LoanAmount NUMBER,

    InterestRate NUMBER,

    OriginalInterestRate NUMBER, -- For keeping original rate

    StartDate DATE,

    EndDate DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

    EmployeeID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    Position VARCHAR2(50),

    Salary NUMBER,

    Department VARCHAR2(50),

    HireDate DATE

);

CREATE TABLE Loans (

    LoanID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    LoanAmount NUMBER,

    InterestRate NUMBER,

    OriginalInterestRate NUMBER,  -- For storing original value

    StartDate DATE,

    EndDate DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Insert Customers

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1955-05-15', 'YYYY-MM-DD'), 12000, SYSDATE, 'N');

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 8000, SYSDATE, 'N');

-- Insert Accounts

INSERT INTO Accounts VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Checking', 1500, SYSDATE);

-- Insert Transactions

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

-- Insert Loans

INSERT INTO Loans VALUES (1, 1, 5000, 7, NULL, SYSDATE, ADD\_MONTHS(SYSDATE, 20));

INSERT INTO Loans VALUES (2, 2, 3000, 6, NULL, SYSDATE, ADD\_MONTHS(SYSDATE, 1));

-- Insert Employees

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

-- Commit the changes

COMMIT;

BEGIN

  FOR cust IN (

    SELECT c.CustomerID, c.DOB, l.LoanID, l.InterestRate

    FROM Customers c

    JOIN Loans l ON c.CustomerID = l.CustomerID

  ) LOOP

    IF (MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12) > 60 THEN

      UPDATE Loans

      SET OriginalInterestRate = NVL(OriginalInterestRate, InterestRate),

          InterestRate = InterestRate - 1

      WHERE LoanID = cust.LoanID;

    END IF;

  END LOOP;

END;

/

SELECT \* FROM Loans;

BEGIN

  FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP

    IF cust.Balance > 10000 THEN

      UPDATE Customers

      SET IsVIP = 'Y'

      WHERE CustomerID = cust.CustomerID;

    END IF;

  END LOOP;

END;

/

SELECT CustomerID, Name, Balance, IsVIP FROM Customers;

BEGIN

  FOR loan IN (

    SELECT c.Name, l.EndDate

    FROM Customers c

    JOIN Loans l ON c.CustomerID = l.CustomerID

    WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE(

      'Reminder: Loan for ' || loan.Name ||

      ' is due on ' || TO\_CHAR(loan.EndDate, 'YYYY-MM-DD')

    );

  END LOOP;

END;

/

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  FOR acc IN (

    SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings'

  ) LOOP

    UPDATE Accounts

    SET Balance = Balance + (acc.Balance \* 0.01),

        LastModified = SYSDATE

    WHERE AccountID = acc.AccountID;

  END LOOP;

END;

/

BEGIN

  ProcessMonthlyInterest;

END;

/

SELECT \* FROM Accounts WHERE AccountType = 'Savings';

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

  p\_department IN VARCHAR2,

  p\_bonus\_percent IN NUMBER

) AS

BEGIN

  UPDATE Employees

  SET Salary = Salary + (Salary \* p\_bonus\_percent / 100)

  WHERE Department = p\_department;

END;

/

BEGIN

  UpdateEmployeeBonus('HR', 10);

END;

/

SELECT \* FROM Employees WHERE Department = 'HR';

CREATE OR REPLACE PROCEDURE TransferFunds(

  p\_from\_account IN NUMBER,

  p\_to\_account IN NUMBER,

  p\_amount IN NUMBER

) AS

  v\_balance NUMBER;

BEGIN

  -- Check if sufficient balance exists

  SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from\_account;

  IF v\_balance < p\_amount THEN

    RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account.');

  END IF;

  -- Debit from source

  UPDATE Accounts

  SET Balance = Balance - p\_amount,

      LastModified = SYSDATE

  WHERE AccountID = p\_from\_account;

  -- Credit to destination

  UPDATE Accounts

  SET Balance = Balance + p\_amount,

      LastModified = SYSDATE

  WHERE AccountID = p\_to\_account;

END;

/

BEGIN

  TransferFunds(1, 2, 200);

END;

/

SELECT \* FROM Accounts WHERE AccountID IN (1, 2);