**PL/SQL Programming Exercise**

**TABLES:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

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,

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

);

**Sample Data Insertion**

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

VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

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

VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

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

VALUES (1, 1, 'Savings', 1000, SYSDATE);

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

VALUES (2, 2, 'Checking', 1500, SYSDATE);

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

VALUES (1, 1, SYSDATE, 200, 'Deposit');

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

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

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

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

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

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

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

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

**Exercise 1: Control Structures**

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

* + 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.

**Code:**

BEGIN

FOR loan\_rec IN (

SELECT L.LoanID, L.InterestRate, C.DOB

FROM Loans L

JOIN Customers C ON L.CustomerID = C.CustomerID

)

LOOP

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

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = loan\_rec.LoanID;

END IF;

END LOOP;

COMMIT;

END;

/

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

* + 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.

**Code:**

BEGIN

FOR rec IN (SELECT CustomerID, Name, Balance FROM Customers) LOOP

    IF rec.Balance > 10000 THEN

      UPDATE Customers

      SET IsVIP = 'TRUE'

      WHERE CustomerID = rec.CustomerID;

      DBMS\_OUTPUT.PUT\_LINE(' VIP Assigned: ' || rec.Name || ' (Balance: ' || rec.Balance || ')');

    END IF;

  END LOOP;

END

/

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**Code:**

  FOR rec IN (

    SELECT C.Name, L.LoanID, L.EndDate

    FROM Loans L

    JOIN Customers C ON C.CustomerID = L.CustomerID

    WHERE L.EndDate BETWEEN SYSDATE AND SYSDATE + 30

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE(' Reminder: Loan ' || rec.LoanID ||

                         ' for ' || rec.Name || ' is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

  END LOOP;

  COMMIT;

END;

/

**Exercise 3: Stored Procedures**

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

* + 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.

**Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

  FOR acc IN (SELECT \* FROM Accounts WHERE AccountType = 'Savings') LOOP

    UPDATE Accounts

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

        LastModified = SYSDATE

    WHERE AccountID = acc.AccountID;

    DBMS\_OUTPUT.PUT\_LINE('Interest added to AccountID: ' || acc.AccountID);

  END LOOP;

END;

/

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

* + 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.

**Code:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(dept IN VARCHAR2, bonusPercent IN NUMBER) IS

BEGIN

  FOR emp IN (SELECT \* FROM Employees WHERE Department = dept) LOOP

    UPDATE Employees

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

    WHERE EmployeeID = emp.EmployeeID;

    DBMS\_OUTPUT.PUT\_LINE('Bonus applied to ' || emp.Name || ' in ' || dept);

  END LOOP;

END;

/

**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.

**Code:**

CREATE OR REPLACE PROCEDURE TransferFunds(

    fromAccID IN NUMBER,

    toAccID IN NUMBER,

    amount IN NUMBER

) IS

  fromBalance NUMBER;

BEGIN

  SELECT Balance INTO fromBalance FROM Accounts WHERE AccountID = fromAccID FOR UPDATE;

  IF fromBalance < amount THEN

    DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in account: ' || fromAccID);

  ELSE

    UPDATE Accounts SET Balance = Balance - amount WHERE AccountID = fromAccID;

    UPDATE Accounts SET Balance = Balance + amount WHERE AccountID = toAccID;

    DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount || ' from ' || fromAccID || ' to ' || toAccID);

  END IF;

EXCEPTION

  WHEN NO\_DATA\_FOUND THEN

    DBMS\_OUTPUT.PUT\_LINE('One of the accounts not found.');

END;

/

-- Test the procedures

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('--- Interest Processing ---');

  ProcessMonthlyInterest;

  DBMS\_OUTPUT.PUT\_LINE('--- Apply Bonus to Sales Department ---');

  UpdateEmployeeBonus('Sales', 10);

  DBMS\_OUTPUT.PUT\_LINE('--- Fund Transfer ---');

  TransferFunds(101, 102, 1000);

END;

/