**PL/SQL PROGRAMMING EXERCISES**

-- Enable output

SET SERVEROUTPUT ON;

-- DROP existing tables

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE Transactions CASCADE CONSTRAINTS';

  EXECUTE IMMEDIATE 'DROP TABLE Accounts CASCADE CONSTRAINTS';

  EXECUTE IMMEDIATE 'DROP TABLE Loans CASCADE CONSTRAINTS';

  EXECUTE IMMEDIATE 'DROP TABLE Employees CASCADE CONSTRAINTS';

  EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS';

EXCEPTION

  WHEN OTHERS THEN NULL;

END;

/

-- CREATE TABLES

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    DOB DATE,

    Balance NUMBER,

    LastModified DATE,

    IsVIP VARCHAR2(5)

);

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

);

-- INSERT SAMPLE DATA

INSERT INTO Customers VALUES (1, 'Alice', TO\_DATE('1960-05-10','YYYY-MM-DD'), 15000, SYSDATE, NULL);

INSERT INTO Customers VALUES (2, 'Bob', TO\_DATE('1985-07-20','YYYY-MM-DD'), 9500, SYSDATE, NULL);

INSERT INTO Customers VALUES (3, 'Charlie', TO\_DATE('1950-02-15','YYYY-MM-DD'), 12000, SYSDATE, NULL);

INSERT INTO Loans VALUES (101, 1, 50000, 8.0, SYSDATE - 100, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 30000, 7.5, SYSDATE - 200, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 45000, 9.0, SYSDATE - 300, SYSDATE + 5);

COMMIT;

**Exercise 1: Control Structures**

BEGIN

**Scenario 1: Apply 1% Interest Discount to Customers Above 60**

  FOR rec IN (

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

    FROM Loans L JOIN Customers C ON L.CustomerID = C.CustomerID

  ) LOOP

    DECLARE

      v\_age NUMBER;

    BEGIN

      v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, rec.DOB)/12);

      IF v\_age > 60 THEN

        UPDATE Loans

        SET InterestRate = InterestRate - (InterestRate \* 0.01)

        WHERE LoanID = rec.LoanID;

        DBMS\_OUTPUT.PUT\_LINE(' 1% Discount Applied: ' || rec.Name || ' (Age: ' || v\_age || ')');

      END IF;

    END;

  END LOOP;

**Scenario 2: Promote to VIP if Balance > 10000**

  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;

**Scenario 3: Send Loan Due Reminders (next 30 days)**

  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'));

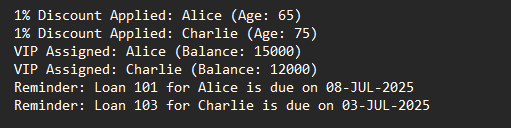
  END LOOP;

  COMMIT;

END;

/

**Output:**



**Exercise 3: Stored Procedures**

-- Enable DBMS Output

SET SERVEROUTPUT ON;

-- Drop tables for clean rerun

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE Transactions';

  EXECUTE IMMEDIATE 'DROP TABLE Loans';

  EXECUTE IMMEDIATE 'DROP TABLE Accounts';

  EXECUTE IMMEDIATE 'DROP TABLE Employees';

  EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

  WHEN OTHERS THEN NULL;

END;

/

--Create 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 Employees (

    EmployeeID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    Position VARCHAR2(50),

    Salary NUMBER,

    Department VARCHAR2(50),

    HireDate DATE

);

-- Insert Sample Data

INSERT INTO Customers VALUES (1, 'Alice', TO\_DATE('1950-05-10', 'YYYY-MM-DD'), 15000, SYSDATE);

INSERT INTO Customers VALUES (2, 'Bob', TO\_DATE('1985-08-21', 'YYYY-MM-DD'), 8000, SYSDATE);

INSERT INTO Accounts VALUES (101, 1, 'Savings', 10000, SYSDATE);

INSERT INTO Accounts VALUES (102, 1, 'Current', 3000, SYSDATE);

INSERT INTO Accounts VALUES (103, 2, 'Savings', 5000, SYSDATE);

INSERT INTO Employees VALUES (1, 'John', 'Manager', 50000, 'Sales', SYSDATE);

INSERT INTO Employees VALUES (2, 'Jane', 'Clerk', 30000, 'Sales', SYSDATE);

INSERT INTO Employees VALUES (3, 'Max', 'Analyst', 40000, 'IT', SYSDATE);

**Scenario 1: Process Monthly Interest for Savings Accounts**

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: Update Employee Bonus**

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: Transfer Funds**

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;

/

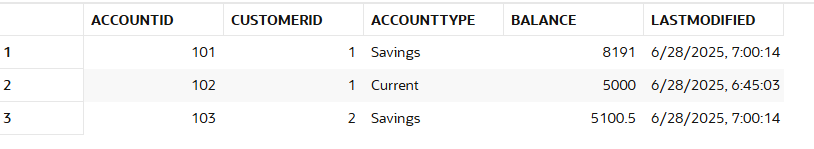
-- Final view of tables

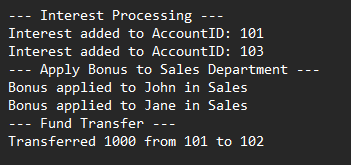
SELECT \* FROM Accounts;

SELECT \* FROM Employees;

**Output:**

**Query Result:**

****

****