**Week 2 (Pl/SQL Programming)**

**Exercise 1: Control Structures**

-- DROP & CREATE TABLES --

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE Transactions';

  EXECUTE IMMEDIATE 'DROP TABLE Accounts';

  EXECUTE IMMEDIATE 'DROP TABLE Loans';

  EXECUTE IMMEDIATE 'DROP TABLE Customers';

  EXECUTE IMMEDIATE 'DROP TABLE Employees';

EXCEPTION

  WHEN OTHERS THEN NULL;

END;

/

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

);

-- INSERT DATA --

INSERT INTO Customers VALUES (1, 'Amit Sharma', TO\_DATE('1950-05-20', 'YYYY-MM-DD'), 12000, SYSDATE);

INSERT INTO Customers VALUES (2, 'Sunita Verma', TO\_DATE('1985-10-14', 'YYYY-MM-DD'), 8500, SYSDATE);

INSERT INTO Customers VALUES (3, 'Ravi Kumar', TO\_DATE('1992-03-08', 'YYYY-MM-DD'), 9000, SYSDATE);

INSERT INTO Customers VALUES (4, 'Meena Joshi', TO\_DATE('1949-12-25', 'YYYY-MM-DD'), 15000, SYSDATE);

INSERT INTO Customers VALUES (5, 'Arjun Mehta', TO\_DATE('1978-09-17', 'YYYY-MM-DD'), 7000, SYSDATE);

INSERT INTO Accounts VALUES (101, 1, 'Savings', 5000, SYSDATE);

INSERT INTO Accounts VALUES (102, 2, 'Checking', 3000, SYSDATE);

INSERT INTO Accounts VALUES (103, 3, 'Savings', 2500, SYSDATE);

INSERT INTO Accounts VALUES (104, 4, 'Checking', 9000, SYSDATE);

INSERT INTO Accounts VALUES (105, 5, 'Savings', 1500, SYSDATE);

INSERT INTO Transactions VALUES (1001, 101, SYSDATE - 10, 1000, 'DEPOSIT');

INSERT INTO Transactions VALUES (1002, 102, SYSDATE - 7, 500, 'WITHDRAW');

INSERT INTO Transactions VALUES (1003, 103, SYSDATE - 5, 200, 'DEPOSIT');

INSERT INTO Transactions VALUES (1004, 104, SYSDATE - 3, 1200, 'WITHDRAW');

INSERT INTO Transactions VALUES (1005, 105, SYSDATE - 1, 800, 'DEPOSIT');

INSERT INTO Loans VALUES (201, 1, 100000, 0.08, SYSDATE - 365, SYSDATE + 5);

INSERT INTO Loans VALUES (202, 2, 75000, 0.07, SYSDATE - 365, SYSDATE + 10);

INSERT INTO Loans VALUES (203, 3, 50000, 0.09, SYSDATE - 365, SYSDATE + 90);

INSERT INTO Loans VALUES (204, 4, 125000, 0.065, SYSDATE - 365, SYSDATE + 60);

INSERT INTO Loans VALUES (205, 5, 30000, 0.1, SYSDATE - 365, SYSDATE + 120);

INSERT INTO Employees VALUES (301, 'Anjali Rao', 'Manager', 90000, 'Operations', TO\_DATE('2015-04-01', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (302, 'Rahul Sen', 'Teller', 45000, 'Retail', TO\_DATE('2018-10-12', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (303, 'Neha Gupta', 'Analyst', 60000, 'Finance', TO\_DATE('2020-03-05', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (304, 'Vikram Iyer', 'IT Officer', 75000, 'Technology', TO\_DATE('2016-07-30', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (305, 'Kiran Das', 'HR Executive', 50000, 'HR', TO\_DATE('2019-01-20', 'YYYY-MM-DD'));

**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.

BEGIN

    FOR cust IN (

        SELECT CustomerID

        FROM Customers

        WHERE MONTHS\_BETWEEN(SYSDATE, DOB) / 12 > 60

    )

    LOOP

        UPDATE Loans

        SET InterestRate = InterestRate - 0.01

        WHERE CustomerID = cust.CustomerID;

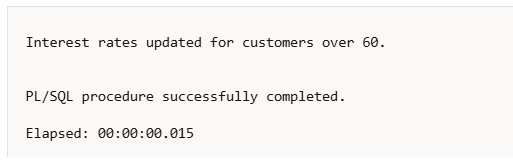
    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE('Interest rates updated for customers over 60.');

END;

/

**Output:**

****

**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.

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

BEGIN

    FOR cust IN (

        SELECT CustomerID

        FROM Customers

        WHERE Balance > 10000

    )

    LOOP

        UPDATE Customers

        SET IsVIP = 'TRUE'

        WHERE CustomerID = cust.CustomerID;

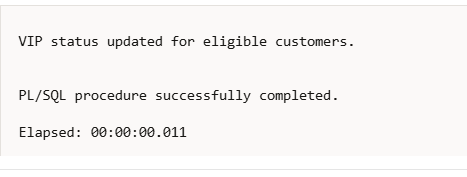
    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE('VIP status updated for eligible customers.');

END;

/

**Output:**

****

**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.

-- SCENARIO 3: LOAN REMINDERS --

BEGIN

    FOR loan\_rec IN (

        SELECT c.Name, 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.Name ||

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

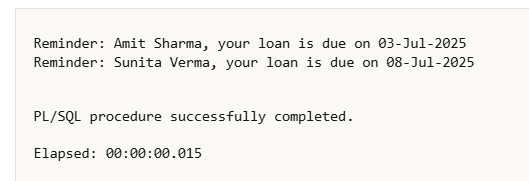
        );

    END LOOP;

END;

/

**Output:**

****

**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.

-- SCENARIO 1: Monthly Interest for Savings Accounts

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    FOR acc IN (

        SELECT AccountID, Balance

        FROM Accounts

        WHERE AccountType = 'Savings'

    ) LOOP

        UPDATE Accounts

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

        WHERE AccountID = acc.AccountID;

    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

/

**Output:**



-- TEST 1: Process Monthly Interest

BEGIN

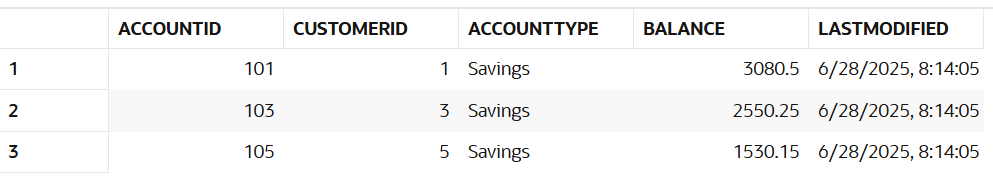
    ProcessMonthlyInterest;

END;

/

-- Check updated savings accounts

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



**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.

-- SCENARIO 2: Bonus for Employees by Department

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

    p\_Department IN VARCHAR2,

    p\_BonusPercent IN NUMBER

) IS

BEGIN

    UPDATE Employees

    SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

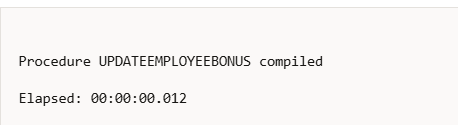
    WHERE Department = p\_Department;

    DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || p\_Department);

END;

/

**Output:**

****

-- TEST 2: Apply 10% Bonus to 'Finance' Department

BEGIN

    UpdateEmployeeBonus('Finance', 10);

END;

/

-- Check updated salaries

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



**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.

-- SCENARIO 3: Transfer Funds Between Accounts

CREATE OR REPLACE PROCEDURE TransferFunds (

    p\_FromAccountID IN NUMBER,

    p\_ToAccountID IN NUMBER,

    p\_Amount IN NUMBER

) IS

    v\_FromBalance NUMBER;

BEGIN

    -- Check balance of source account

    SELECT Balance INTO v\_FromBalance

    FROM Accounts

    WHERE AccountID = p\_FromAccountID;

    IF v\_FromBalance < p\_Amount THEN

        DBMS\_OUTPUT.PUT\_LINE('Insufficient funds for transfer.');

    ELSE

        -- Deduct from source

        UPDATE Accounts

        SET Balance = Balance - p\_Amount

        WHERE AccountID = p\_FromAccountID;

        -- Add to destination

        UPDATE Accounts

        SET Balance = Balance + p\_Amount

        WHERE AccountID = p\_ToAccountID;

        DBMS\_OUTPUT.PUT\_LINE('Transfer of Rs.' || p\_Amount || ' completed successfully.');

    END IF;

END;

/

**Output:**

****

-- TEST 3: Transfer Rs. 2000 from Account 101 to Account 102

BEGIN

    TransferFunds(101, 102, 2000);

END;

/

-- Check updated account balances

SELECT \* FROM Accounts WHERE AccountID IN (101, 102);

