**Exercise 1: Control Structures**

CREATE TABLE Customers (

    CustomerID   NUMBER PRIMARY KEY,

    Name         VARCHAR2(100),

    DOB          DATE,

    Balance      NUMBER(12, 2),

    IsVIP        VARCHAR2(5) DEFAULT 'FALSE'

);

CREATE TABLE Loans (

    LoanID       NUMBER PRIMARY KEY,

    CustomerID   NUMBER REFERENCES Customers(CustomerID),

    InterestRate NUMBER(5, 2),

    DueDate      DATE

);

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

INSERT INTO Customers VALUES (2, 'Alice Green', TO\_DATE('1990-03-22','YYYY-MM-DD'), 8500, 'FALSE');

INSERT INTO Customers VALUES (3, 'Robert White', TO\_DATE('1945-09-15','YYYY-MM-DD'), 11000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 5.50, SYSDATE + 15);

INSERT INTO Loans VALUES (102, 2, 6.00, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 5.75, SYSDATE + 5);

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

**CODE:**

BEGIN

    FOR cust\_rec IN (SELECT CustomerID, DOB FROM Customers) LOOP

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

            UPDATE Loans

            SET InterestRate = InterestRate - 0.01

            WHERE CustomerID = cust\_rec.CustomerID;

        END IF;

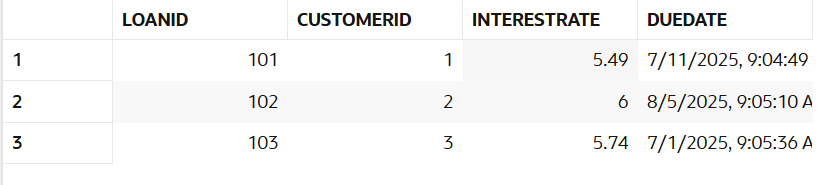
    END LOOP;

    COMMIT;

END;

SELECT \* FROM Loans;

**OUTPUT:**



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

**CODE:**

BEGIN

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

        IF cust\_rec.Balance > 10000 THEN

            UPDATE Customers

            SET IsVIP = 'TRUE'

            WHERE CustomerID = cust\_rec.CustomerID;

        END IF;

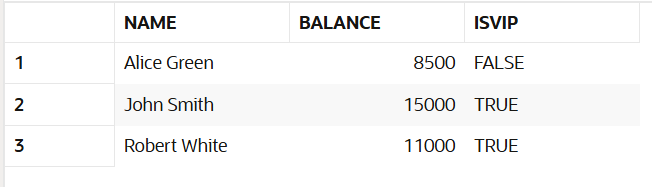
    END LOOP;

    COMMIT;

END;

SELECT 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.

**CODE:**

BEGIN

FOR loan\_rec IN (

SELECT l.LoanID, l.DueDate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.LoanID ||

' for ' || loan\_rec.Name ||

' is due on ' || TO\_CHAR(loan\_rec.DueDate, 'DD-MON-YYYY'));

END LOOP;

END;

**OUTPUT:**



**Exercise 3: Stored Procedures**

CREATE TABLE BankAccounts (

    AccountID     NUMBER PRIMARY KEY,

    CustomerID    NUMBER,

    Balance       NUMBER(12, 2)

);

CREATE TABLE Employees (

    EmpID        NUMBER PRIMARY KEY,

    Name         VARCHAR2(100),

    Department   VARCHAR2(50),

    Salary       NUMBER(10, 2)

);

INSERT INTO BankAccounts VALUES (1001, 201, 10000);

INSERT INTO BankAccounts VALUES (1002, 202, 20000);

INSERT INTO Employees VALUES (1, 'Alice', 'Sales', 50000);

INSERT INTO Employees VALUES (2, 'Bob', 'Sales', 60000);

INSERT INTO Employees VALUES (3, 'Charlie', 'HR', 55000);

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

**CODE:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE BankAccounts

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

    COMMIT;

END;

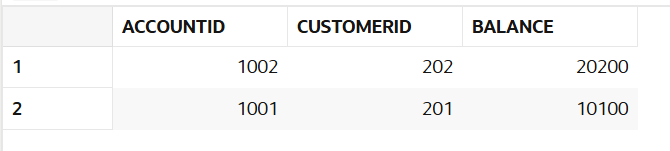
BEGIN

    ProcessMonthlyInterest;

END;

SELECT \* FROM BankAccounts;

**OUTPUT:**



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

**CODE:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

    p\_dept IN VARCHAR2,

    p\_bonus\_pct IN NUMBER

) IS

BEGIN

    UPDATE Employees

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

    WHERE Department = p\_dept;

    COMMIT;

END;

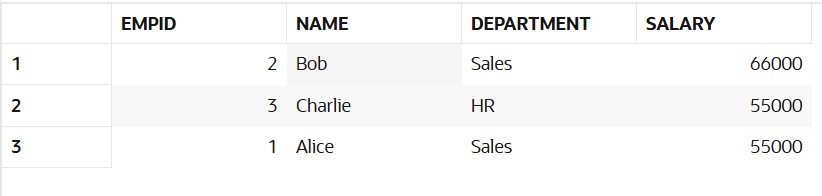
BEGIN

    UpdateEmployeeBonus('Sales', 10);

END;

SELECT \* FROM Employees;

**OUTPUT:**



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

**CODE:**

CREATE OR REPLACE PROCEDURE TransferFunds(

    p\_from\_acct IN NUMBER,

    p\_to\_acct IN NUMBER,

    p\_amount IN NUMBER

) IS

    v\_balance NUMBER;

BEGIN

    SELECT Balance INTO v\_balance

    FROM BankAccounts

    WHERE AccountID = p\_from\_acct;

    IF v\_balance < p\_amount THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance');

    END IF;

    -- Deduct from source

    UPDATE BankAccounts

    SET Balance = Balance - p\_amount

    WHERE AccountID = p\_from\_acct;

    -- Add to destination

    UPDATE BankAccounts

    SET Balance = Balance + p\_amount

    WHERE AccountID = p\_to\_acct;

    COMMIT;

END;

BEGIN

    TransferFunds(1001, 1002, 2000);

END;

SELECT \* FROM BankAccounts;

**OUTPUT:**

