**Exercise 1: Control Structures**

***Code*:**

-- Create Customers and Loans tables

-- Create Customers table to store customer details

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,      -- Unique identifier for each customer

    Name VARCHAR2(50),                  -- Customer name

    Age NUMBER,                         -- Age of the customer

    InterestRate NUMBER,                -- Interest rate on customer's loan

    Balance NUMBER,                     -- Account balance

    IsVIP VARCHAR2(5)                   -- Flag to mark VIP customers (TRUE/FALSE)

);

-- Create Loans table to store loan information

CREATE TABLE Loans (

    LoanID NUMBER PRIMARY KEY,          -- Unique identifier for each loan

    CustomerID NUMBER,                  -- Foreign key reference to customer

    DueDate DATE                        -- Loan due date

);

-- STEP 2: Insert sample data into Customers and Loans tables

INSERT INTO Customers VALUES (1, 'Kumar', 63, 10.0, 13000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Babu', 48, 11.0, 9000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Suresh', 67, 9.5, 11000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, SYSDATE + 10); -- Kumar

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

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

COMMIT;

-- Scenario 1 - Apply 1% interest discount for customers age > 60

BEGIN

    -- Loop through all customers

    FOR rec IN (SELECT CustomerID, Age FROM Customers)

    LOOP

        -- Check if customer is above 60

        IF rec.Age > 60 THEN

            -- Apply 1% discount to interest rate

            UPDATE Customers

            SET InterestRate = InterestRate - 1

            WHERE CustomerID = rec.CustomerID;

        END IF;

    END LOOP;

    COMMIT; -- Save changes

END;

/

-- Scenario 2 - Set IsVIP = TRUE for balance > 10000

BEGIN

    -- Loop through all customers

    FOR rec IN (SELECT CustomerID, Balance FROM Customers)

    LOOP

        -- Check if balance is greater than $10,000

        IF rec.Balance > 10000 THEN

            -- Set IsVIP flag to TRUE

            UPDATE Customers

            SET IsVIP = 'TRUE'

            WHERE CustomerID = rec.CustomerID;

        END IF;

    END LOOP;

    COMMIT; -- Save changes

END;

/

-- Scenario 3 - Show loan reminders for loans due within 30 days

BEGIN

    -- Print header for output

    DBMS\_OUTPUT.PUT\_LINE('--- Loan Due Reminders ---');

    -- Fetch loans due within the next 30 days and join with customer names

    FOR 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

        -- Print reminder message for each loan

        DBMS\_OUTPUT.PUT\_LINE(

            'Reminder: ' || rec.Name ||

            ' has loan ' || rec.LoanID ||

            ' due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY')

        );

    END LOOP;

END;

A screenshot of a computer error message

AI-generated content may be incorrect./

**Exercise 3: Stored Procedures**

***Code:***

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Balance NUMBER

);

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Department VARCHAR2(50),

Salary NUMBER

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Balance NUMBER

);

INSERT INTO SavingsAccounts VALUES (1, 'Kumar', 10000);

INSERT INTO SavingsAccounts VALUES (2, 'Babu', 15000);

INSERT INTO Employees VALUES (101, 'Anita', 'HR', 50000);

INSERT INTO Employees VALUES (102, 'Ravi', 'IT', 60000);

INSERT INTO Employees VALUES (103, 'Priya', 'IT', 55000);

INSERT INTO Accounts VALUES (201, 'Suresh', 8000);

INSERT INTO Accounts VALUES (202, 'Geetha', 12000);

COMMIT;

-- Scenario 1: Process Monthly Interest

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (SELECT AccountID, Balance FROM SavingsAccounts)

LOOP

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = rec.AccountID;

END LOOP;

COMMIT;

END;

/

-- Scenario 2: Update Employee Bonus

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;

/

-- Scenario 3: Transfer Funds Between Accounts

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_account;

IF v\_balance < p\_amount THEN

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

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

COMMIT;

END;

/

-- Scenario 1 Test

EXEC ProcessMonthlyInterest;

-- Scenario 2 Test

EXEC UpdateEmployeeBonus('IT', 10);

-- Scenario 3 Test

EXEC TransferFunds(201, 202, 2000);

SELECT \* FROM SavingsAccounts;

SELECT \* FROM Employees;

SELECT \* FROM Accounts;

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.Result**