**Exercise 1: Control Structures**

**Code:**

-- Create Customers table

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(50),

    Age NUMBER,

    InterestRate NUMBER,

    Balance NUMBER,

    IsVIP VARCHAR2(5)

);

-- Create Loans table

CREATE TABLE Loans (

    LoanID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    DueDate DATE

);

-- Insert sample data

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

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

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

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

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

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

COMMIT;

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

BEGIN

    FOR rec IN (SELECT CustomerID, Age FROM Customers)

    LOOP

        IF rec.Age > 60 THEN

            UPDATE Customers

            SET InterestRate = InterestRate - 1

            WHERE CustomerID = rec.CustomerID;

        END IF;

    END LOOP;

    COMMIT;

END;

/

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

BEGIN

    FOR rec IN (SELECT CustomerID, Balance FROM Customers)

    LOOP

        IF rec.Balance > 10000 THEN

            UPDATE Customers

            SET IsVIP = 'TRUE'

            WHERE CustomerID = rec.CustomerID;

        END IF;

    END LOOP;

    COMMIT;

END;

/

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

BEGIN

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

    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

        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:***

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE SavingsAccounts';

  EXECUTE IMMEDIATE 'DROP TABLE Employees';

  EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXCEPTION

  WHEN OTHERS THEN NULL;

END;

/

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, 'Naveen', 9000);

INSERT INTO SavingsAccounts VALUES (2, 'Divya', 12000);

INSERT INTO SavingsAccounts VALUES (3, 'Rajesh', 18000);

INSERT INTO Employees VALUES (201, 'Meena', 'Finance', 45000);

INSERT INTO Employees VALUES (202, 'Arjun', 'IT', 52000);

INSERT INTO Employees VALUES (203, 'Lavanya', 'IT', 48000);

INSERT INTO Accounts VALUES (301, 'Prakash', 7000);

INSERT INTO Accounts VALUES (302, 'Harini', 10500);

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);

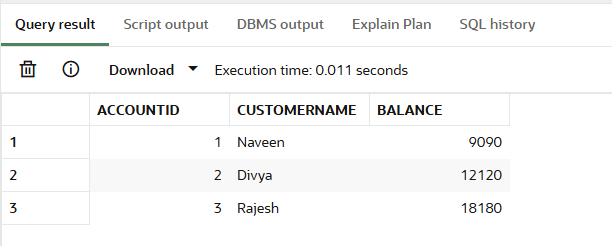
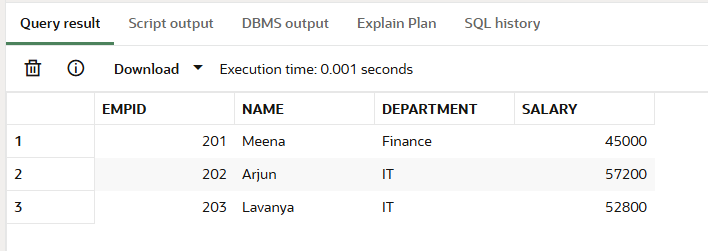
-- Scenario 3 Test

EXEC TransferFunds(301, 302, 2000);

SELECT \* FROM SavingsAccounts;

SELECT \* FROM Employees;

SELECT \* FROM Accounts;

***Result***:

