**PL/SQL**

**Exercise 1: Control Structures**

**--Customers Table:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER (10,2),

IsVIP VARCHAR2(5) DEFAULT 'FALSE'

);

**--Loans Table:**

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

InterestRate NUMBER (5,2),

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers (CustomerID)

);

**-- Inserting customers data**

INSERT INTO Customers VALUES (1, 'John Doe', 65, 15000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Jane Smith', 45, 8000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Robert Lee', 70, 12000, 'FALSE');

**-- Inserting loans data**

INSERT INTO Loans VALUES (101, 1, 5.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 6.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 5.0, SYSDATE + 20);

COMMIT;

**Scenario 1:**

BEGIN

FOR cust\_rec IN (

SELECT c. CustomerID, l. LoanID, l. InterestRate

FROM Customers c

JOIN Loans l ON c. CustomerID = l. CustomerID

WHERE c. Age > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

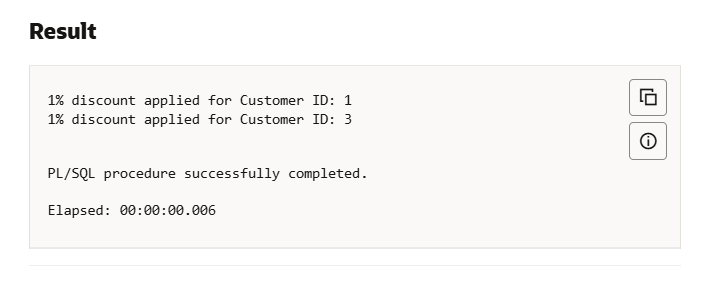
WHERE LoanID = cust\_rec. LoanID;

DBMS\_OUTPUT.PUT\_LINE ('1% discount applied for Customer ID: ' || cust\_rec. CustomerID);

END LOOP;

END;

/

****

**Scenario 2:**

BEGIN

FOR vip\_rec IN (

        SELECT CustomerID FROM Customers WHERE Balance > 10000

    ) LOOP

        UPDATE Customers

        SET IsVIP = 'TRUE'

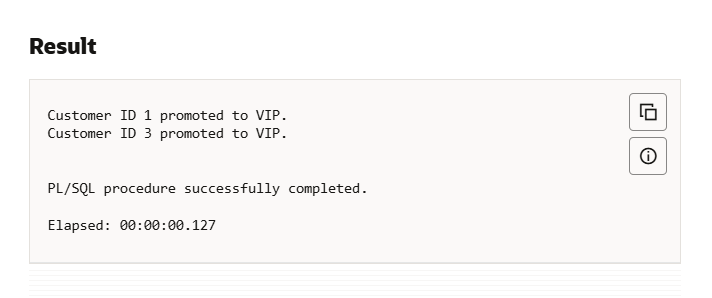
        WHERE CustomerID = vip\_rec.CustomerID;

        DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || vip\_rec.CustomerID || ' promoted to VIP.');

    END LOOP;

END;

/



**Scenario 3:**

BEGIN

    FOR due\_rec IN (

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

        FROM Loans l

        JOIN Customers c ON l. CustomerID = c. CustomerID

        WHERE l. DueDate <= SYSDATE + 30

    ) LOOP

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

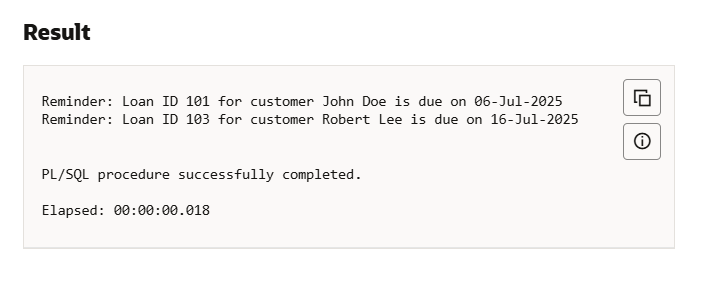
                             ' for customer ' || due\_rec. Name ||

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

    END LOOP;

END;

/



**Exercise 3: Stored Procedures**

**Scenario 1:**

**-- Savings accounts table**

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER (10,2)

);

**-- Employees table**

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Department VARCHAR2(50),

Salary NUMBER (10,2)

);

**-- General accounts table (for transfers)**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER (10,2)

);

**-- Insert sample savings accounts data**

INSERT INTO SavingsAccounts VALUES (101, 1, 1000);

INSERT INTO SavingsAccounts VALUES (102, 2, 2000);

**-- Insert sample employee’s data**

INSERT INTO Employees VALUES (1, 'Alice', 'HR', 5000);

INSERT INTO Employees VALUES (2, 'Bob', 'IT', 6000);

INSERT INTO Employees VALUES (3, 'Charlie', 'IT', 5500);

**-- Insert sample accounts data**

INSERT INTO Accounts VALUES (201, 1, 1000); -- Source account

INSERT INTO Accounts VALUES (202, 1, 500); -- Destination account

COMMIT;

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest added to Account ID: ' || acc.AccountID);

END LOOP;

END;

/

**--Call Procedure 1**

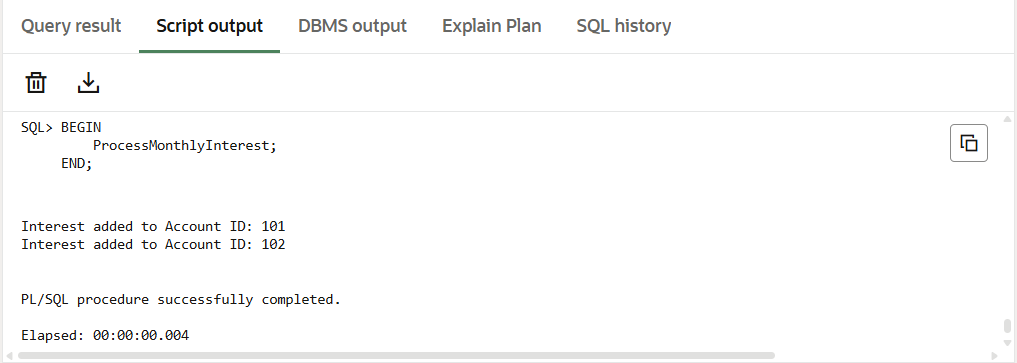
BEGIN

ProcessMonthlyInterest;

END;

/

**Result:**

****

**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

    p\_Department IN VARCHAR2,

    p\_BonusPercent IN NUMBER

) AS

BEGIN

    FOR emp IN (

        SELECT EmpID FROM Employees WHERE Department = p\_Department

    ) LOOP

        UPDATE Employees

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

        WHERE EmpID = emp.EmpID;

        DBMS\_OUTPUT.PUT\_LINE('Bonus added to Employee ID: ' || emp.EmpID);

    END LOOP;

END;

/

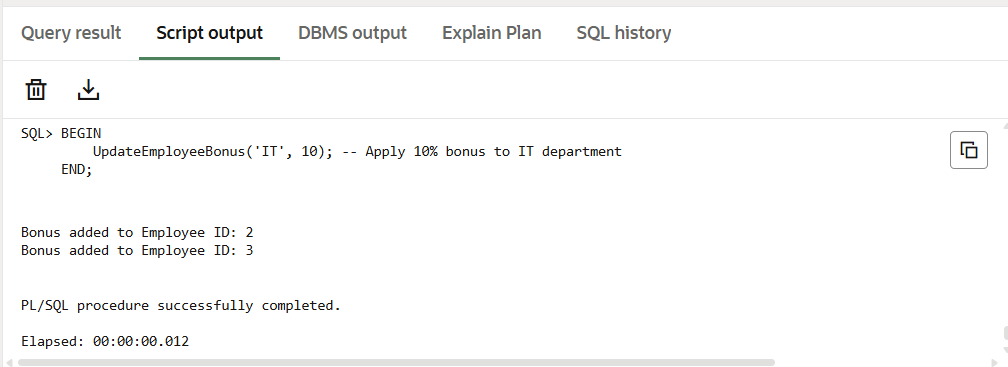
BEGIN

    UpdateEmployeeBonus('IT', 10); -- Apply 10% bonus to IT department

END;

/

**Result:**

****

**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) AS

v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance FROM Accounts WHERE AccountID = p\_FromAccountID FOR UPDATE;

IF v\_FromBalance >= p\_Amount THEN

-- 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 ('Transferred $' || p\_Amount || ' from Account ' || p\_FromAccountID || ' to Account ' || p\_ToAccountID);

ELSE

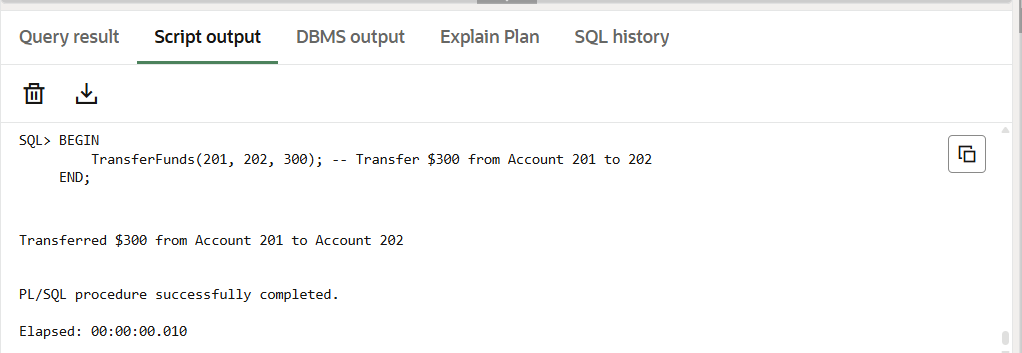
DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient funds in Account ' || p\_FromAccountID);

END IF;

END;

/

**Result:**

****