**WEEK -2**

**SUPERSET ID: 6384042**

**PL/SQL programming:**

**Exercise 1: Control Structures**

**Scenario 1: Apply 1% Interest Discount for Senior Citizens**

**Objective:** Apply a 1% discount on loan interest rates for customers over 60 years of age.

**Code:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

InterestRate NUMBER,

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers VALUES (1, 'Alice', 65, 12000, 'FALSE');

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

INSERT INTO Customers VALUES (3, 'Charlie', 70, 15000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 9.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 10.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 8.5, SYSDATE + 20);

COMMIT; DECLARE

CURSOR customer\_cursor IS

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

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE c.Age > 60;

BEGIN

FOR rec IN customer\_cursor LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

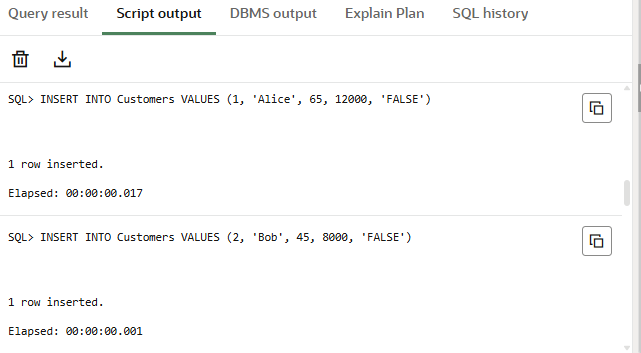
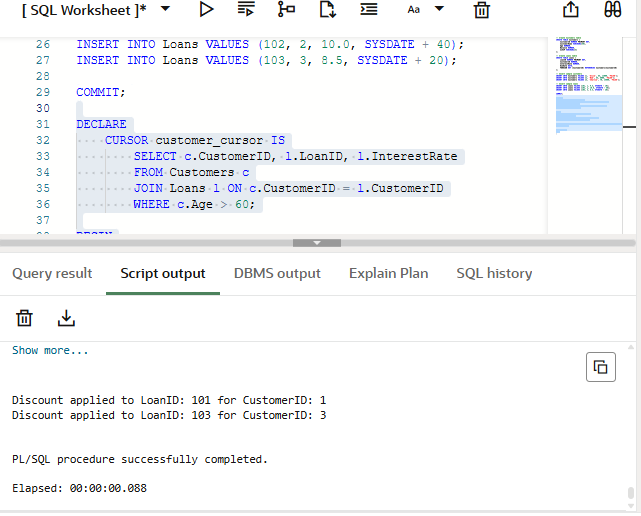
WHERE LoanID = rec.LoanID;

DBMS\_OUTPUT.PUT\_LINE('Discount applied to LoanID: ' || rec.LoanID || ' for CustomerID: ' || rec.CustomerID);

END LOOP;

COMMIT;

END;

**OUTPUT:**

**Scenario 2: Mark Customers as VIP Based on Balance**

**Objective:**Set IsVIP = 'TRUE' for customers whose Balance > 10,000.

**Precondition:** Add the IsVIP column

**Code:**

DECLARE

CURSOR vip\_cursor IS

SELECT CustomerID, Balance

FROM Customers

WHERE Balance > 10000;

BEGIN

FOR rec IN vip\_cursor LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

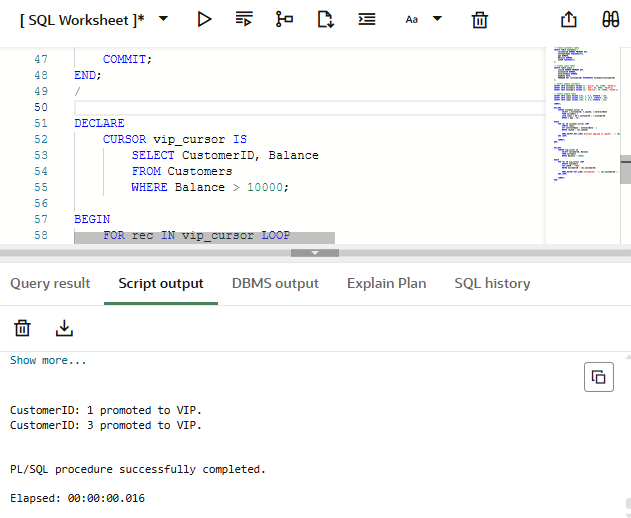
DBMS\_OUTPUT.PUT\_LINE('CustomerID: ' || rec.CustomerID || ' promoted to VIP.');

END LOOP;

COMMIT;

END;

**OUTPUT:**

****

**Scenario 3: Remind Customers of Loans Due in Next 30 Days**

**Objective:**Print reminders for loans due within the next 30 days.

**Code:**

DECLARE

CURSOR due\_loan\_cursor IS

SELECT l.LoanID, l.DueDate, c.CustomerName

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR rec IN due\_loan\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: LoanID ' || rec.LoanID || ' for Customer ' || rec.CustomerName ||

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

END LOOP;

END;

**OUTPUT:**

****

**Exercise 3: Stored Procedures (PL/SQL)**

**Scenario 1: Process Monthly Interest for Savings Accounts**

**Objective:**Create a stored procedure ProcessMonthlyInterest to apply 1% monthly interest to all accounts of type 'Savings'.

**Code:**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER

);

INSERT INTO Accounts VALUES (1, 101, 'SAVINGS', 1000);

INSERT INTO Accounts VALUES (2, 102, 'CURRENT', 5000);

INSERT INTO Accounts VALUES (3, 103, 'SAVINGS', 2000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'SAVINGS';

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

END;

/

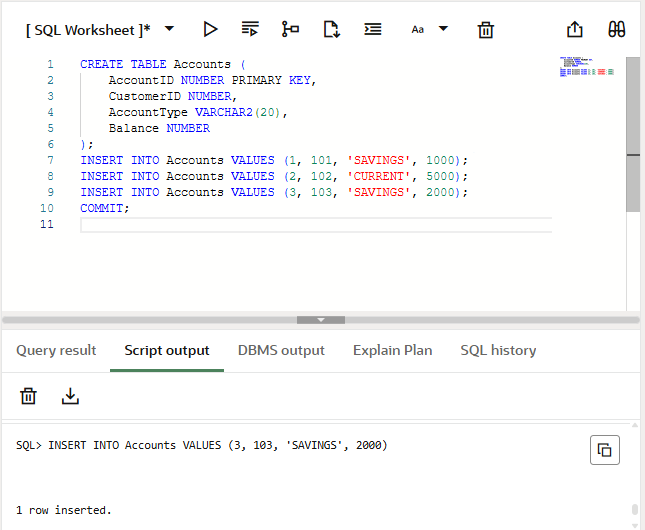
SET SERVEROUTPUT ON;

BEGIN

ProcessMonthlyInterest;

END;

/

**OUTPUT:**

**Scenario 2: Apply Bonus to Employees by Department**

**Objective:** Create a stored procedure UpdateEmployeeBonus to apply a bonus percentage to employees in a given department.

**Code:**

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

EmpName VARCHAR2(50),

DepartmentID NUMBER,

Salary NUMBER

);

INSERT INTO Employees VALUES (1, 'Alice', 101, 50000);

INSERT INTO Employees VALUES (2, 'Bob', 101, 55000);

INSERT INTO Employees VALUES (3, 'Charlie', 102, 60000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_DepartmentID IN NUMBER,

p\_BonusPercent IN NUMBER) AS

BEGIN

UPDATE Employees

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

WHERE DepartmentID = p\_DepartmentID;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_BonusPercent || '% applied to Department ID: ' || p\_DepartmentID);

END;

/

SET SERVEROUTPUT ON;

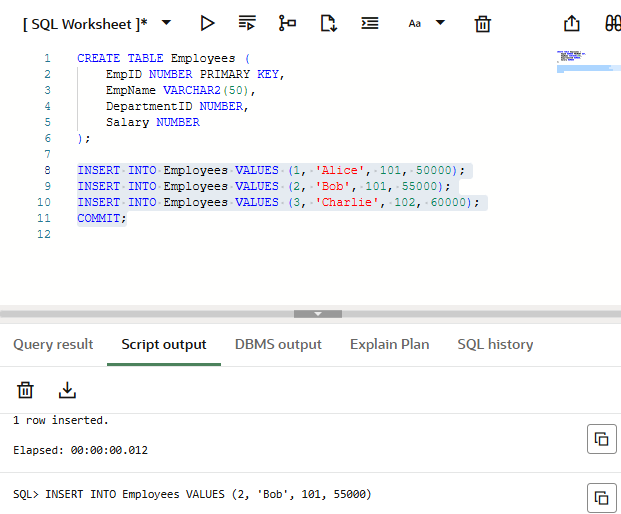
BEGIN

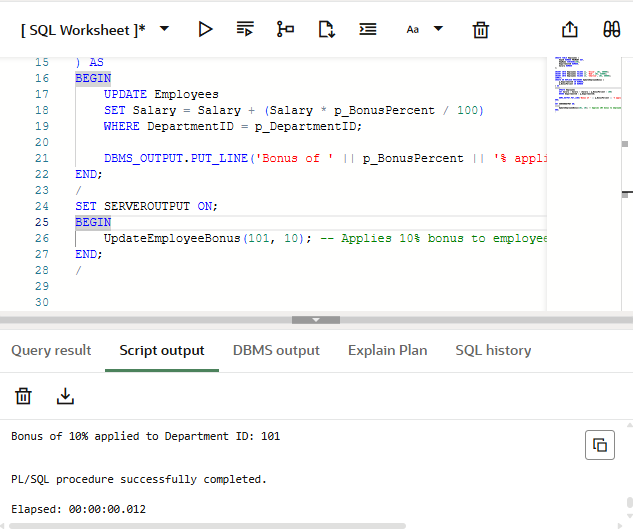
UpdateEmployeeBonus(101, 10); -- Applies 10% bonus to employees in department 101

END;

/

**OUTPUT:**

****

****

**Scenario 3: Transfer Funds Between Accounts**

**Objective:** Create a stored procedure TransferFunds that transfers a specified amount from one account to another with a check for sufficient balance.

**Code:**

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;

IF v\_FromBalance < p\_Amount THEN

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

END IF;

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_Amount || ' from Account ' || p\_FromAccountID || ' to ' || p\_ToAccountID);

END;

/

BEGIN

TransferFunds(1, 2, 500);

END;

/

**OUTPUT:**

