**WEEK – 2(Hands-on)**

**PLSQL\_Exercises**

**Exercise 1: Control Structures**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

InterestRate NUMBER(5,2),

Balance NUMBER(10,2),

IsVIP VARCHAR2(5)

);

INSERT INTO Customers VALUES (1, 'John Doe', 65, 7.5, 12000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Jane Smith', 45, 6.0, 8000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Emily Johnson', 70, 8.0, 15000, 'FALSE');

COMMIT;

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

DueDate DATE

);

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

BEGIN

FOR cust\_rec IN (SELECT CustomerID FROM Customers WHERE Age > 60) LOOP

UPDATE Customers

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust\_rec.CustomerID;

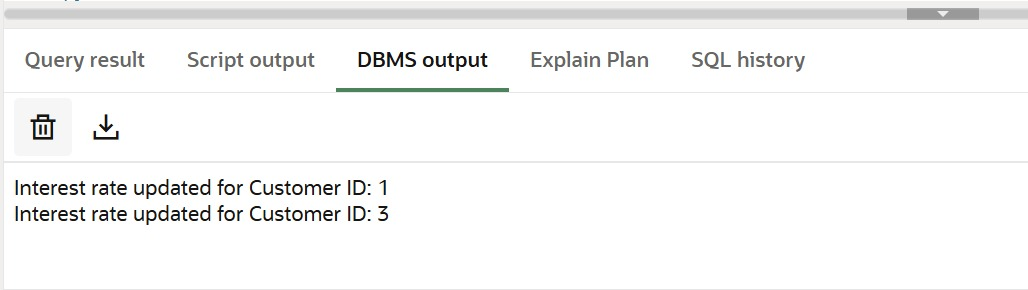
DBMS\_OUTPUT.PUT\_LINE('Interest rate updated for customer ID: ' || cust\_rec.CustomerID);

END LOOP;

COMMIT;

END;

**Output:**



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

BEGIN

FOR vip\_rec IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = vip\_rec.CustomerID;

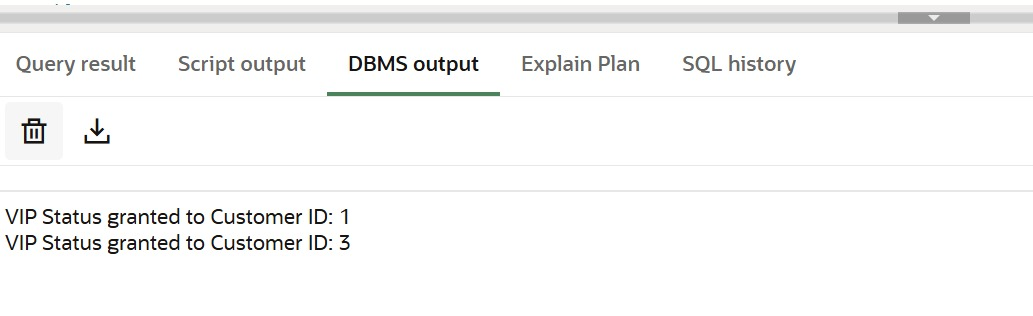
DBMS\_OUTPUT.PUT\_LINE ('Customer ID ' || vip\_rec. CustomerID || ' promoted to VIP.');

END LOOP;

COMMIT;

END;

**Output:**



**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

BEGIN

FOR loan\_rec IN (

SELECT LoanID, CustomerID, DueDate

FROM Loans

WHERE DueDate <= SYSDATE + 30

) LOOP

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

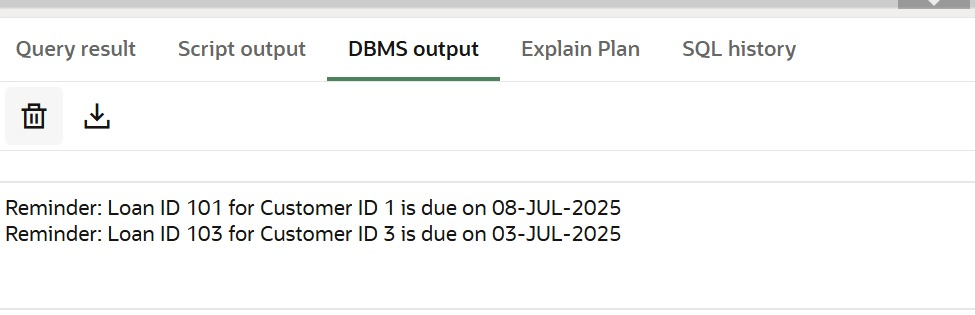
' for Customer ID ' || loan\_rec.CustomerID ||

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

END LOOP;

END;

**Output:**



**Exercise 3: Stored Procedures**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(10, 2)

);

INSERT INTO Accounts VALUES (101, 1, 5000);

INSERT INTO Accounts VALUES (102, 2, 3000);

INSERT INTO Accounts VALUES (103, 3, 10000);

COMMIT;

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Department VARCHAR2(30),

Salary NUMBER(10,2)

);

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

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

INSERT INTO Employees VALUES (3, 'Charlie', 'IT', 70000);

COMMIT;

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

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

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

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc.AccountID;

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

END LOOP;

COMMIT;

END;

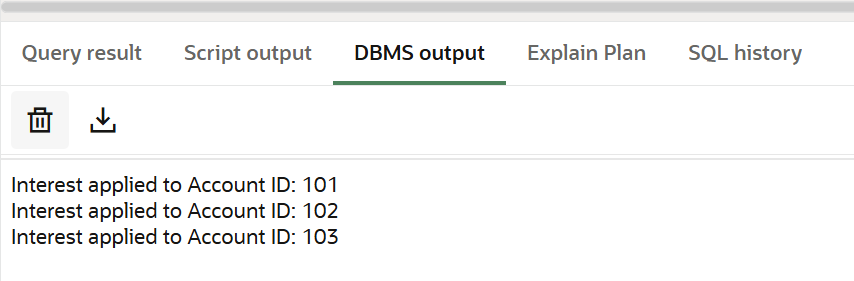
/

BEGIN

ProcessMonthlyInterest;

END;

**Output:**



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

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

FOR emp IN (SELECT EmpID FROM Employees WHERE Department = dept\_name) LOOP

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE EmpID = emp.EmpID;

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

END LOOP;

COMMIT;

END;

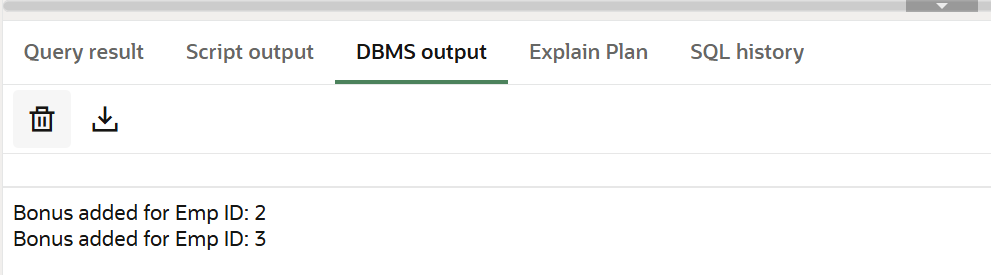
/

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

**Output:**



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

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

SELECT Balance INTO from\_balance FROM Accounts WHERE AccountID = from\_account;

IF from\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE(' Insufficient funds in Account ID: ' || from\_account);

ELSE

UPDATE Accounts SET Balance = Balance - amount WHERE AccountID = from\_account;

UPDATE Accounts SET Balance = Balance + amount WHERE AccountID = to\_account;

DBMS\_OUTPUT.PUT\_LINE(' Transferred ' || amount || ' from ' || from\_account || ' to ' || to\_account);

COMMIT;

END IF;

END;

/

BEGIN

TransferFunds(101, 102, 1000);

END;

**Output:**

