**PL/SQL Programming**

**Exercise 1: Control Structures**

**Create table Customers**

CREATE TABLE Customers (

CustomerID NUMBER,

Name VARCHAR2(50),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

**Create table Loans**

CREATE TABLE Loans (

LoanID NUMBER,

CustomerID NUMBER,

InterestRate NUMBER,

DueDate DATE

);

**Insert into Customers**

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

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

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

**Insert into Loans val1**

INSERT INTO Loans VALUES (101, 1, 10.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 9.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 11.0, SYSDATE + 25);

**Scenario 1 : Control Structures – Loan Discount For 60+ :**

BEGIN

-- Scenario 1: Apply 1% discount for customers above 60

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

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

-- Output line to show update

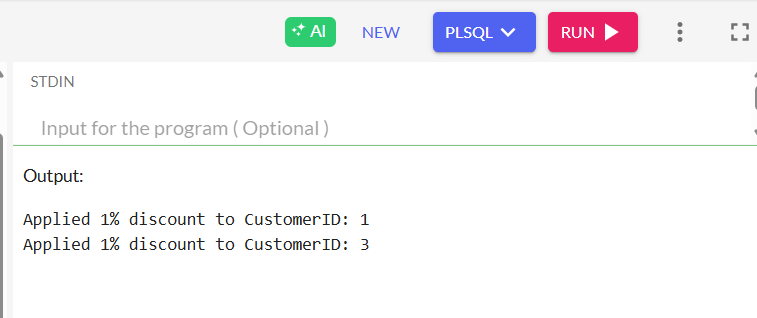
DBMS\_OUTPUT.PUT\_LINE('Applied 1% discount to CustomerID: ' || cust.CustomerID);

END LOOP;

END;

/

**Output:**

****

**Scenario 2 : Mark VIP Customers**

SET SERVEROUTPUT ON;

BEGIN

-- Loop through customers with balance > 10,000

FOR cust IN (SELECT CustomerID, Name FROM Customers WHERE Balance > 10000) LOOP

-- Update IsVIP to TRUE

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

-- Print customer name

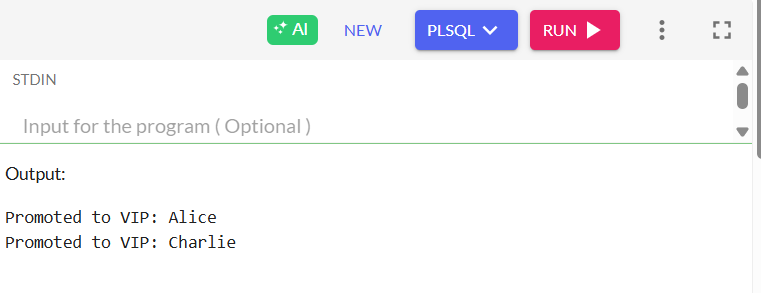
DBMS\_OUTPUT.PUT\_LINE('Promoted to VIP: ' || cust.Name);

END LOOP;

END;

/

**Output :**

****

**Scenario 3 : Loan Due To Be Reminded**

SET SERVEROUTPUT ON;

BEGIN

-- Loop through loans due within 30 days

FOR loan\_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: Loan ID ' || loan\_rec.LoanID ||

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

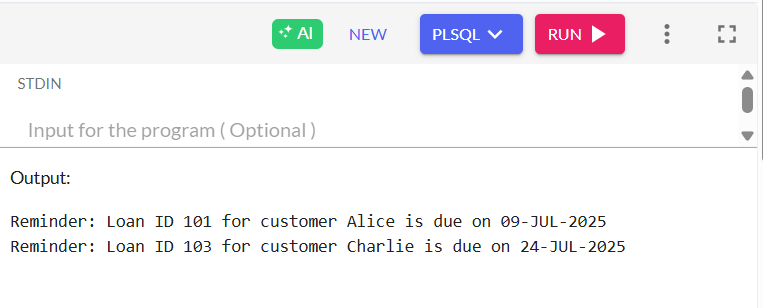
' is due on ' || TO\_CHAR(loan\_rec.DueDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

**Output :**

****

**Exercise 3: Stored Procedures**

**Create table accounts**

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY,

CustomerID INT,

Balance NUMBER(10,2),

AccountType VARCHAR2(20)

);

**Create table employees**

CREATE TABLE Employees (

EmpID INT PRIMARY KEY,

Name VARCHAR2(100),

Department VARCHAR2(50),

Salary NUMBER(10,2)

);

**Insert val into accounts**

INSERT INTO Accounts VALUES (1, 101, 5000.00, 'Savings');

INSERT INTO Accounts VALUES (2, 102, 7000.00, 'Current');

INSERT INTO Accounts VALUES (3, 103, 12000.00, 'Savings');

INSERT INTO Accounts VALUES (4, 104, 3000.00, 'Savings');

**Insert val into employees**

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

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

INSERT INTO Employees VALUES (3, 'Charlie', 'HR', 55000);

INSERT INTO Employees VALUES (4, 'David', 'Finance', 58000);

**Scenario : 1**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

newBalance NUMBER;

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

newBalance := acc.Balance \* 1.01;

UPDATE Accounts

SET Balance = newBalance

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to AccountID ' || acc.AccountID ||

' | New Balance: ' || TO\_CHAR(newBalance, '99999.99'));

END LOOP;

END;

/

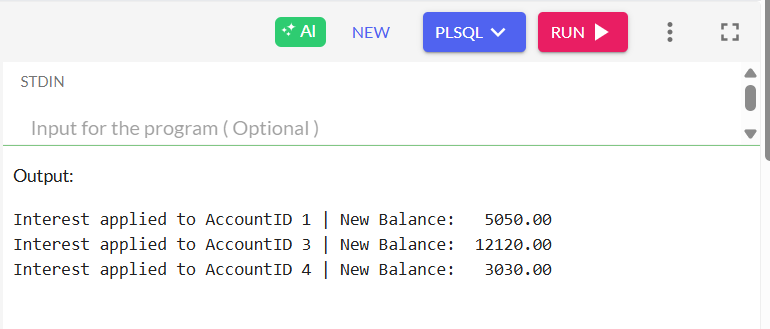
BEGIN

ProcessMonthlyInterest;

END;

/

**Output:**



**Scenario : 2**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

deptName IN VARCHAR2,

bonusPercent IN NUMBER

) IS

newSalary NUMBER;

BEGIN

FOR emp IN (SELECT EmpID, Name, Salary FROM Employees WHERE Department = deptName) LOOP

newSalary := emp.Salary + (emp.Salary \* bonusPercent / 100);

UPDATE Employees

SET Salary = newSalary

WHERE EmpID = emp.EmpID;

DBMS\_OUTPUT.PUT\_LINE('Updated Salary for ' || emp.Name || ' in ' || deptName || ': ' || newSalary);

END LOOP;

END;

/

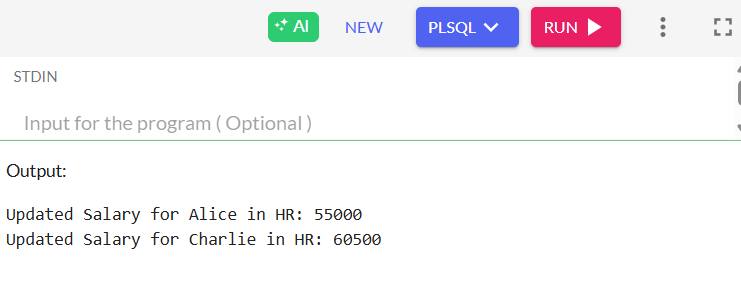
BEGIN

UpdateEmployeeBonus('HR', 10);

END;

/

**Output:**



**Scenario : 3**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_FromAccountID IN INT,

p\_ToAccountID IN INT,

p\_Amount IN NUMBER

) IS

v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance < p\_Amount THEN

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

ELSE

-- Deduct from source

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('Transfer successful from AccountID ' || p\_FromAccountID ||

' to AccountID ' || p\_ToAccountID ||

' | Amount: ' || p\_Amount);

END IF;

END;

/

BEGIN

TransferFunds(3, 1, 2000); -- Transfer $2000 from AccountID 3 to AccountID 1

END;

/

**Output:**

