WEEK 2

PL/SQL Exercise Solutions

**Exercise 1: Control Structures**

**Scenario 1:**

SET SERVEROUTPUT ON;

DECLARE

    c\_age NUMBER;

BEGIN

  FOR rec IN (

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

        FROM Customers c

        JOIN Loans l ON c.CustomerID = l.CustomerID

    ) LOOP

        c\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, rec.DOB) / 12);

        IF c\_age > 60 THEN

            UPDATE Loans

            SET InterestRate = InterestRate - 1

            WHERE LoanID = rec.LoanID;

            DBMS\_OUTPUT.PUT\_LINE('Customer: ' || rec.Name ||

                                 ' | Age: ' || c\_age ||

                                 ' | New Interest Rate: ' || (rec.InterestRate - 1));

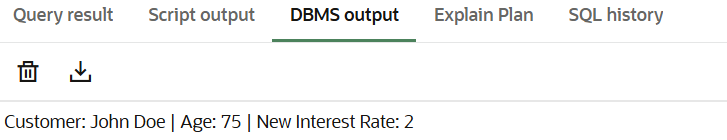
        END IF;

    END LOOP;

    COMMIT;

END;

OUTPUT



**Scenario 2:**

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

SET SERVEROUTPUT ON;

BEGIN

FOR rec IN (SELECT CustomerID, Name, Balance FROM Customers) LOOP

IF rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer: ' || rec.Name ||

' | Balance: ' || rec.Balance ||

' | IsVIP: TRUE');

END IF;

END LOOP;

COMMIT;

END;

OUTPUT



**Scenario 3:**

SET SERVEROUTPUT ON;

BEGIN

    FOR rec IN (

        SELECT c.Name, l.EndDate

        FROM Customers c

        JOIN Loans l ON c.CustomerID = l.CustomerID

        WHERE l.EndDate <= SYSDATE + 30

    ) LOOP

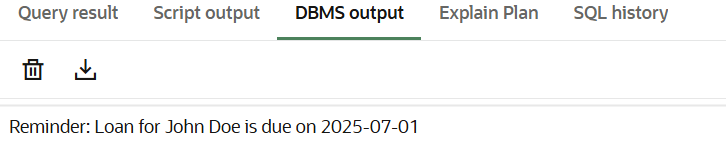
        DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan for ' || rec.Name ||

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

    END LOOP;

END;

OUTPUT



**Exercise 3: Stored Procedures**

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

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

UPDATE Accounts

SET Balance = Balance + (rec.Balance \* 0.01)

WHERE AccountID = rec.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Account ID: ' || rec.AccountID ||

' | New Balance: ' || TO\_CHAR(rec.Balance \* 1.01));

END LOOP;

COMMIT;

END;

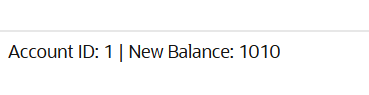
SET SERVEROUTPUT ON;

BEGIN

ProcessMonthlyInterest;

END;

OUTPUT



**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

FOR rec IN (SELECT EmployeeID, Salary FROM Employees WHERE Department = p\_department) LOOP

UPDATE Employees

SET Salary = Salary + (rec.Salary \* p\_bonus\_percent / 100)

WHERE EmployeeID = rec.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || rec.EmployeeID ||

' | New Salary: ' || TO\_CHAR(rec.Salary \* (1 + p\_bonus\_percent / 100)));

END LOOP;

COMMIT;

END;

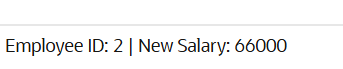
SET SERVEROUTPUT ON;

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

OUTPUT



**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

c\_balance NUMBER;

BEGIN

SELECT Balance INTO c\_balance FROM Accounts WHERE AccountID = p\_from\_account;

IF c\_balance >= p\_amount THEN

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_amount ||

' from Account ' || p\_from\_account ||

' to Account ' || p\_to\_account);

COMMIT;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in Account ' || p\_from\_account);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One of the accounts does not exist.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

SET SERVEROUTPUT ON;

BEGIN

TransferFunds(1, 2, 100);

END;

OUTPUT

