**PLSQL\_Exercises (Mandatory Hands On)**

**Exercise 1: Control Structures**

**Code:**

**Scenario 1:**

set serveroutput on;

DECLARE

  CURSOR c\_customers IS

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

    FROM Customers c

    JOIN Loans l ON c.*CustomerID* **=** l.*CustomerID*

    WHERE MONTHS\_BETWEEN(SYSDATE, c.*DOB*) **/** 12 **>** 60;

  v\_customer\_id Customers.*CustomerID*%TYPE;

  v\_loan\_id Loans.*LoanID*%TYPE;

  v\_interest\_rate Loans.*InterestRate*%TYPE;

BEGIN

  OPEN c\_customers;

  LOOP

    FETCH c\_customers INTO v\_customer\_id, v\_loan\_id, v\_interest\_rate;

    EXIT WHEN c\_customers%NOTFOUND;

    UPDATE Loans

    SET InterestRate **=** v\_interest\_rate **-** 1

    WHERE LoanID **=** v\_loan\_id;

  END LOOP;

  CLOSE c\_customers;

  DBMS\_OUTPUT.*PUT\_LINE*('Operation Completed');

  COMMIT;

END;

**/**

**Scenario 2:**

ALTER TABLE Customers ADD (IsVIP VARCHAR2(3));

DECLARE

  CURSOR c\_customers IS

    SELECT CustomerID, Balance

    FROM Customers

    WHERE Balance **>** 10000;

  v\_customer\_id Customers.*CustomerID*%TYPE;

  v\_balance Customers.*Balance*%TYPE;

BEGIN

  OPEN c\_customers;

  LOOP

    FETCH c\_customers INTO v\_customer\_id, v\_balance;

    EXIT WHEN c\_customers%NOTFOUND;

    UPDATE Customers

    SET IsVIP **=** 'YES'

    WHERE CustomerID **=** v\_customer\_id;

  END LOOP;

  CLOSE c\_customers;

  COMMIT;

END;

**/**

**Scenario 3:**

set serveroutput on;

DECLARE

  CURSOR c\_loans IS

    SELECT c.*CustomerID*, c.*Name*, l.*LoanID*, l.*EndDate*

    FROM Loans l

    JOIN Customers c ON c.*CustomerID* **=** l.*CustomerID*

    WHERE l.*EndDate* BETWEEN SYSDATE AND SYSDATE **+** 30;

  v\_customer\_id Customers.*CustomerID*%TYPE;

  v\_customer\_name Customers.*Name*%TYPE;

  v\_loan\_id Loans.*LoanID*%TYPE;

  v\_end\_date Loans.*EndDate*%TYPE;

BEGIN

  OPEN c\_loans;

  LOOP

    FETCH c\_loans INTO v\_customer\_id, v\_customer\_name, v\_loan\_id, v\_end\_date;

    EXIT WHEN c\_loans%NOTFOUND;

    DBMS\_OUTPUT.*PUT\_LINE*('Reminder: Loan ' **||** v\_loan\_id **||** ' for customer :' **||** v\_customer\_name **||** ' is due on ' **||** TO\_CHAR(v\_end\_date, 'YYYY-MM-DD'));

  END LOOP;

  CLOSE c\_loans;

END;

**Exercise 3: Stored Procedures**

**Code:**

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

    UPDATE Accounts

    SET Balance **=** Balance **\*** 1.01,

        LastModified **=** SYSDATE

    WHERE AccountType **=** 'Savings';

END;

**/**

**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(p\_Department VARCHAR2, p\_BonusPercentage NUMBER) AS

BEGIN

    UPDATE Employees

    SET Salary **=** Salary **\*** (1 **+** p\_BonusPercentage **/** 100)

    WHERE Department **=** p\_Department;

END;

**/**

**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(p\_SourceAccountID NUMBER, p\_TargetAccountID NUMBER, p\_Amount NUMBER) AS

    v\_SourceBalance Accounts.*Balance*%TYPE;

BEGIN

*-- Checking if the account has sufficient balance or not*

    SELECT Balance INTO v\_SourceBalance

    FROM Accounts

    WHERE AccountID **=** p\_SourceAccountID

    FOR UPDATE;

    IF v\_SourceBalance **>=** p\_Amount THEN

*-- Deducting the amount from the source account*

        UPDATE Accounts

        SET Balance **=** Balance **-** p\_Amount,

            LastModified **=** SYSDATE

        WHERE AccountID **=** p\_SourceAccountID;

*-- Adding the amount to the destination account*

        UPDATE Accounts

        SET Balance **=** Balance **+** p\_Amount,

            LastModified **=** SYSDATE

        WHERE AccountID **=** p\_TargetAccountID;

    ELSE

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

    END IF;

END;

**/**

SELECT **\*** FROM Employees;

SELECT **\*** FROM Customers;

SELECT **\*** FROM Accounts;