Week2\_Control Structures

# Exercise 1: Control Structures

## 🔧 Database Setup

### 📌 1. Create Tables

-- Table: Customers  
CREATE TABLE Customers (  
 CustomerID NUMBER PRIMARY KEY,  
 Name VARCHAR2(100),  
 Age NUMBER,  
 Balance NUMBER(10, 2),  
 IsVIP VARCHAR2(5)  
);  
  
-- Table: Loans  
CREATE TABLE Loans (  
 LoanID NUMBER PRIMARY KEY,  
 CustomerID NUMBER,  
 InterestRate NUMBER(5, 2),  
 DueDate DATE,  
 FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
);

### 📌 2. Insert Sample Data

-- Customers  
INSERT INTO Customers VALUES (1, 'John Smith', 65, 8000.00, 'FALSE');  
INSERT INTO Customers VALUES (2, 'Jane Doe', 45, 15000.00, 'FALSE');  
INSERT INTO Customers VALUES (3, 'Robert Brown', 70, 11000.00, 'FALSE');  
INSERT INTO Customers VALUES (4, 'Emily Davis', 30, 5000.00, 'FALSE');  
  
-- Loans  
INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 15); -- Due in 15 days  
INSERT INTO Loans VALUES (102, 2, 6.0, SYSDATE + 45); -- Due in 45 days  
INSERT INTO Loans VALUES (103, 3, 8.2, SYSDATE + 5); -- Due in 5 days  
INSERT INTO Loans VALUES (104, 4, 5.9, SYSDATE + 60); -- Due in 60 days  
  
COMMIT;

## 🧪 Scenario 1: Apply Senior Citizen Loan Discount

### 🔷 Question:

Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

### ✅ PL/SQL Code:

BEGIN  
 FOR cust IN (  
 SELECT c.CustomerID, l.LoanID  
 FROM Customers c  
 JOIN Loans l ON c.CustomerID = l.CustomerID  
 WHERE c.Age > 60  
 )  
 LOOP  
 UPDATE Loans  
 SET InterestRate = InterestRate - 1  
 WHERE LoanID = cust.LoanID;  
  
 DBMS\_OUTPUT.PUT\_LINE('Discount applied to Customer ID: ' || cust.CustomerID || ', Loan ID: ' || cust.LoanID);  
 END LOOP;  
END;  
/

### 📸 Output

Discount applied to Customer ID: 1, Loan ID: 101  
Discount applied to Customer ID: 3, Loan ID: 103

## 🧪 Scenario 2: Promote to VIP Status

### 🔷 Question:

Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

### ✅ PL/SQL Code:

BEGIN  
 FOR cust IN (  
 SELECT CustomerID FROM Customers WHERE Balance > 10000  
 )  
 LOOP  
 UPDATE Customers  
 SET IsVIP = 'TRUE'  
 WHERE CustomerID = cust.CustomerID;  
  
 DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || cust.CustomerID || ' promoted to VIP.');  
 END LOOP;  
END;  
/

### 📸 Output

Customer ID 2 promoted to VIP.  
Customer ID 3 promoted to VIP.

## 🧪 Scenario 3: Loan Due Reminders

### 🔷 Question:

Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

### ✅ PL/SQL Code:

BEGIN  
 FOR due\_loan IN (  
 SELECT l.LoanID, c.Name, l.DueDate  
 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 ' || due\_loan.LoanID ||  
 ' for ' || due\_loan.Name ||  
 ' is due on ' || TO\_CHAR(due\_loan.DueDate, 'DD-MM-YYYY'));  
 END LOOP;  
END;  
/

### 📸 Output

Reminder: Loan ID 101 for John Smith is due on 12-07-2025  
Reminder: Loan ID 103 for Robert Brown is due on 02-07-2025

Week2\_StoredProcedures

## Exercise 3 : Stored Procedures

## 🔧 Database Setup

### 📌 1. Create Tables

-- Table: Accounts  
CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER(10, 2)

);

-- Table: Employees  
CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DepartmentID NUMBER,

Salary NUMBER(10, 2)

);

### 📌 2. Insert Sample Data

-- Accounts  
INSERT INTO Accounts VALUES (201, 1, 'Savings', 5000.00);

INSERT INTO Accounts VALUES (202, 2, 'Savings', 10000.00);

INSERT INTO Accounts VALUES (301, 3, 'Checking', 7000.00);

INSERT INTO Accounts VALUES (302, 4, 'Checking', 4000.00);

COMMIT;

--Employees

INSERT INTO Employees VALUES (101, 'Alice Green', 10, 50000);

INSERT INTO Employees VALUES (102, 'Bob White', 10, 55000);

INSERT INTO Employees VALUES (103, 'Charlie Black', 20, 48000);COMMIT;

## 🧪 Scenario 1: Monthly Interest Calculation

### 🔷 Question:

Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

### ✅ PL/SQL Procedure Code:

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS  
BEGIN  
 FOR acct IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP  
 UPDATE Accounts  
 SET Balance = Balance + (Balance \* 0.01)  
 WHERE AccountID = acct.AccountID;  
  
 DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID: ' || acct.AccountID);  
 END LOOP;  
END;  
/

### 📸 Output

Interest applied to Account ID: 201  
Interest applied to Account ID: 202

## 🧪 Scenario 2: Employee Bonus Update

### 🔷 Question:

Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

### ✅ PL/SQL Procedure Code:

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (  
 dept\_id IN NUMBER,  
 bonus\_percent IN NUMBER  
) AS  
BEGIN  
 FOR emp IN (SELECT EmployeeID, Salary FROM Employees WHERE DepartmentID = dept\_id) LOOP  
 UPDATE Employees  
 SET Salary = Salary + (Salary \* bonus\_percent / 100)  
 WHERE EmployeeID = emp.EmployeeID;  
  
 DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Employee ID: ' || emp.EmployeeID);  
 END LOOP;  
END;  
/

### 📸 Output

Bonus applied to Employee ID: 101  
Bonus applied to Employee ID: 102

## 🧪 Scenario 3: Fund Transfer Between Accounts

### 🔷 Question:

Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

### ✅ PL/SQL Procedure Code:

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 FOR UPDATE;  
  
 IF from\_balance >= amount THEN  
 UPDATE Accounts  
 SET Balance = Balance - amount  
 WHERE AccountID = from\_account;  
  
 UPDATE Accounts  
 SET Balance = Balance + amount  
 WHERE AccountID = to\_account;  
  
 DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || amount || ' completed from Account ID ' || from\_account || ' to Account ID ' || to\_account);  
 ELSE  
 DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in Account ID ' || from\_account);  
 END IF;  
END;  
/

### 📸 Output

Transfer of 1000 completed from Account ID 301 to Account ID 302