# Week 2 - PL/SQL Programming - Hands-On

## Exercise 1: Control Structures

### Scenario 1: Interest Discount for Senior Customers

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.

DECLARE  
 CURSOR senior\_customers IS  
 SELECT c.CustomerID, c.DOB, l.LoanID, l.InterestRate  
 FROM Customers c  
 JOIN Loans l ON c.CustomerID = l.CustomerID;  
 v\_age NUMBER;  
BEGIN  
 FOR cust IN senior\_customers LOOP  
 v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12);  
 IF v\_age > 60 THEN  
 UPDATE Loans  
 SET InterestRate = InterestRate - 1  
 WHERE LoanID = cust.LoanID;  
 DBMS\_OUTPUT.PUT\_LINE('Discount applied for customer ID: ' || cust.CustomerID);  
 END IF;  
 END LOOP;  
END;

### Scenario 2: VIP Status Assignment

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.

Note: Assume an additional column `IsVIP` (CHAR(1)) exists in the `Customers` table.

BEGIN  
 FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP  
 IF cust.Balance > 10000 THEN  
 UPDATE Customers  
 SET IsVIP = 'Y'  
 WHERE CustomerID = cust.CustomerID;  
 DBMS\_OUTPUT.PUT\_LINE('VIP status granted to customer ID: ' || cust.CustomerID);  
 END IF;  
 END LOOP;  
END;

### Scenario 3: Loan Reminder Notifications

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

BEGIN  
 FOR loan IN (  
 SELECT l.LoanID, l.CustomerID, c.Name, l.EndDate  
 FROM Loans l  
 JOIN Customers c ON l.CustomerID = c.CustomerID  
 WHERE l.EndDate <= SYSDATE + 30  
 ) LOOP  
 DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan.LoanID || ' for customer ' ||  
 loan.Name || ' is due on ' || TO\_CHAR(loan.EndDate, 'YYYY-MM-DD'));  
 END LOOP;  
END;

## **Exercise 3: Stored Procedures**

### **Scenario 1: Monthly Interest Processing for Savings Accounts**

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.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS  
 BEGIN  
 UPDATE Accounts  
 SET Balance = Balance \* 1.01,  
 LastModified = SYSDATE  
 WHERE AccountType = 'Savings';  
   
 DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');  
 END;

### **Scenario 2: Bonus Scheme for Employees**

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.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(  
 p\_Department IN VARCHAR2,  
 p\_BonusPercent IN NUMBER  
 ) IS  
 BEGIN  
 UPDATE Employees  
 SET Salary = Salary + (Salary \* p\_BonusPercent / 100)  
 WHERE Department = p\_Department;  
   
 DBMS\_OUTPUT.PUT\_LINE('Bonus updated for department: ' || p\_Department);  
 END;

### **Scenario 3: Funds 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.

CREATE OR REPLACE PROCEDURE TransferFunds(  
 p\_SourceAccountID IN NUMBER,  
 p\_TargetAccountID IN NUMBER,  
 p\_Amount IN NUMBER  
 ) IS  
 v\_SourceBalance NUMBER;  
 BEGIN  
 SELECT Balance INTO v\_SourceBalance  
 FROM Accounts  
 WHERE AccountID = p\_SourceAccountID FOR UPDATE;  
   
 IF v\_SourceBalance >= p\_Amount THEN  
 UPDATE Accounts  
 SET Balance = Balance - p\_Amount,  
 LastModified = SYSDATE  
 WHERE AccountID = p\_SourceAccountID;  
   
 UPDATE Accounts  
 SET Balance = Balance + p\_Amount,  
 LastModified = SYSDATE  
 WHERE AccountID = p\_TargetAccountID;  
   
 DBMS\_OUTPUT.PUT\_LINE('Funds transferred successfully.');  
 ELSE  
 DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in source account.');  
 END IF;  
 END;