# Exercise 1: Control Structures

**Scenario 1: Apply 1% Discount to Customers Above 60**

Assumes a table Customers with columns: CustomerID, Age, and LoanInterestRate.

**Code:**

BEGIN

FOR rec IN (SELECT CustomerID, Age, LoanInterestRate FROM Customers) LOOP

IF rec.Age > 60 THEN

UPDATE Customers

SET LoanInterestRate = LoanInterestRate - 1

WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

/

**Scenario 2: Promote Customers with Balance > $10,000 to VIP**

Assumes a table Customers with columns: CustomerID, Balance, and IsVIP (as a BOOLEAN or CHAR/Y/N).

**Code:**

BEGIN

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

IF rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y' -- or TRUE depending on your DB design

WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

/

**Scenario 3: Reminders for Loans Due in Next 30 Days**

Assumes a table Loans with columns: LoanID, CustomerID, and DueDate, and a Customers table with CustomerName.

**Code:**

DECLARE

v\_customerName Customers.CustomerName%TYPE;

BEGIN

FOR rec IN (

SELECT LoanID, CustomerID, DueDate

FROM Loans

WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

SELECT CustomerName INTO v\_customerName

FROM Customers

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || v\_customerName ||

', your loan (ID: ' || rec.LoanID ||

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

END LOOP;

END;

/

# Exercise 3: Stored Procedures

**Scenario 1: Process Monthly Interest on Savings Accounts**

**Assumptions**:

* Table: SavingsAccounts
* Columns: AccountID, Balance

**Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

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

WHERE AccountID = rec.AccountID;

END LOOP;

COMMIT;

END;

/

**Scenario 2: Update Employee Bonus by Department**

**Assumptions**:

* Table: Employees
* Columns: EmployeeID, DepartmentID, Salary
* Input Parameters: p\_DepartmentID, p\_BonusPercentage (as a number like 5 for 5%)

**Code:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_DepartmentID IN NUMBER,

p\_BonusPercentage IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE DepartmentID = p\_DepartmentID;

COMMIT;

END;

/

**Scenario 3: Transfer Funds Between Accounts**

**Assumptions**:

* Table: Accounts
* Columns: AccountID, Balance
* Input Parameters: p\_FromAccountID, p\_ToAccountID, p\_Amount

**Code:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check source account balance

SELECT Balance INTO v\_balance FROM Accounts

WHERE AccountID = p\_FromAccountID

FOR UPDATE;

IF v\_balance < p\_Amount THEN

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

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

COMMIT;

END;

/