**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **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.

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **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.

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

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

**Scenario 1**

BEGIN

FOR customer\_rec IN (

SELECT CustomerID, LoanID, InterestRate

FROM Customers JOIN Loans ON Customers.CustomerID = Loans.CustomerID

WHERE Age > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = customer\_rec.LoanID;

END LOOP;

COMMIT;

END;

### **Scenario 2**

BEGIN

FOR customer\_rec IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = customer\_rec.CustomerID;

END LOOP;

COMMIT;

END;

### **Scenario 3**

BEGIN

FOR loan\_rec IN (

SELECT Loans.LoanID, Customers.CustomerName, Loans.DueDate

FROM Loans JOIN Customers ON Loans.CustomerID = Customers.CustomerID

WHERE DueDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || loan\_rec.CustomerName ||

', your loan (ID: ' || loan\_rec.LoanID ||

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

END LOOP;

END;

**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all 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.

**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* + **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.

**Scenario 3:** Customers should be able to transfer funds between their 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.

### **Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'SAVINGS'

) LOOP

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc.AccountID;

END LOOP;

COMMIT;

END;

### **Scenario 2**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE DepartmentID = dept\_id;

COMMIT;

END;

### **Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_acc\_id IN NUMBER,

to\_acc\_id IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

-- Get balance of source account

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = from\_acc\_id

FOR UPDATE;

IF from\_balance < amount THEN

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

END IF;

-- Deduct from source account

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_acc\_id;

-- Add to destination account

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_acc\_id;

COMMIT;

END;