**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 cust IN (

SELECT CustomerID, DOB

FROM Customers

) LOOP

IF MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12 > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

**/**

**OUTPUT:**

| **CustomerID** | **Name** | **DOB** | **Age** | **LoanID** | **InterestRate** |
| --- | --- | --- | --- | --- | --- |
| 1 | John Doe | 1950-05-15 | 74 | 1 | 5.00 |
| 2 | Jane Smith | 1990-07-20 | 34 | NULL | NULL |

**SCENARIO 2:**

BEGIN

FOR cust IN (

SELECT CustomerID, Balance

FROM Customers

) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

ELSE

UPDATE Customers

SET IsVIP = 'FALSE'

WHERE CustomerID = cust.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

/

**OUTPUT:**

| **CustomerID** | **Name** | **IsVIP** |
| --- | --- | --- |
| 1 | John Doe | FALSE |
| 2 | Jane Smith | TRUE |

**SCENARIO 3:**

DECLARE

v\_name Customers.Name%TYPE;

BEGIN

FOR loan IN (

SELECT CustomerID, EndDate, LoanID

FROM Loans

WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

SELECT Name INTO v\_name

FROM Customers

WHERE CustomerID = loan.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan.LoanID ||

' for customer ' || v\_name ||

' is due on ' || TO\_CHAR(loan.EndDate, 'YYYY-MM-DD'));

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 IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountType = 'Savings';

COMMIT;

END;

/

**SCENARIO 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

deptName IN VARCHAR2,

bonusPercent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonusPercent / 100)

WHERE Department = deptName;

COMMIT;

END;

/

**SCENARIO 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

fromAccountID IN NUMBER,

toAccountID IN NUMBER,

amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check source account balance

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = fromAccountID

FOR UPDATE;

IF v\_balance < amount THEN

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

END IF;

-- Deduct from source account

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = fromAccountID;

-- Add to destination account

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = toAccountID;

COMMIT;

END;

/