PL/SQL programming – Hands-on

**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.

**Code:**

DECLARE

v\_age NUMBER;

BEGIN

FOR rec IN (SELECT CustomerID, InterestRate FROM Loans l)

JOIN Customers c ON l.CustomerID = c.CustomerID

LOOP

SELECT FLOOR(MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12) INTO v\_age FROM Customers c WHERE c.CustomerID = rec.CustomerID;

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate \* 0.99

WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

**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.

**Code:**

ALTER TABLE first: ALTER TABLE Customers ADD (IsVIP VARCHAR2(3));

UPDATE Customers SET IsVIP = 'FALSE';

BEGIN

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

IF rec.Balance > 10000 THEN

UPDATE Customers SET IsVIP = 'TRUE' WHERE CustomerID = rec.CustomerID;

ELSE

UPDATE Customers SET IsVIP = 'FALSE' WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

**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.

**Code:**

BEGIN

FOR rec IN (

SELECT c.Name, l.EndDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND ADD\_MONTHS(SYSDATE, 1)

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || rec.Name || ', your loan is due on ' || TO\_CHAR(rec.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.

**Code:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

COMMIT;

END;

EXECUTE ProcessMonthlyInterest;

**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.

**Code:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department VARCHAR2,

p\_BonusPercentage NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE Department = p\_Department;

COMMIT;

END;

EXECUTE UpdateEmployeeBonus('IT', 10);

**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.

**Code:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_SourceAccountID NUMBER,

p\_DestAccountID NUMBER,

p\_Amount NUMBER

) IS

v\_SourceBalance NUMBER;

BEGIN

SELECT Balance INTO v\_SourceBalance FROM Accounts WHERE AccountID = p\_SourceAccountID;

IF v\_SourceBalance >= p\_Amount THEN

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_SourceAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_DestAccountID;

COMMIT;

ELSE

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

END IF;

END;

EXECUTE TransferFunds(1, 2, 500);