PLSQL\_Exercises

Exercise 1: Control Structures

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

o 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.

o 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.

SOLUTION:

Scenario 1: Apply Interest Discount for Senior Customers

DECLARE

CURSOR cur\_senior\_loans IS

SELECT c.CustomerID, l.LoanID, l.InterestRate

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE c.Age > 60;

BEGIN

FOR rec IN cur\_senior\_loans LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

END LOOP;

COMMIT;

END;

/

Scenario 2: Promote Customers to VIP Based on Balance

DECLARE

CURSOR cur\_customers IS

SELECT CustomerID, Balance

FROM Customers;

BEGIN

FOR rec IN cur\_customers LOOP

IF rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

/

Scenario 3: Loan Payment Reminders for Upcoming Dues

DECLARE

CURSOR cur\_due\_loans IS

SELECT l.LoanID, c.CustomerName, l.DueDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR rec IN cur\_due\_loans LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || rec.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:** 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.

**SOLUTION:**

**Scenario 1: Monthly Interest Processing**

**Objective:** Apply a 1% interest to all savings account balances.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'SAVINGS';

COMMIT;

END;

/

**Scenario 2: Employee Bonus Update**

**Objective:** Update salaries by adding a bonus percentage for employees in a specific department.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_DepartmentId IN NUMBER,

p\_BonusPercent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE DepartmentId = p\_DepartmentId;

COMMIT;

END;

/

**Scenario 3: Fund Transfer Between Accounts**

**Objective:** Transfer funds between two accounts after checking sufficient balance.

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_SourceAccountId IN NUMBER,

p\_TargetAccountId IN NUMBER,

p\_Amount IN NUMBER

) AS

v\_SourceBalance NUMBER;

BEGIN

-- Check source account balance

SELECT Balance INTO v\_SourceBalance

FROM Accounts

WHERE AccountId = p\_SourceAccountId

FOR UPDATE;

IF v\_SourceBalance < p\_Amount THEN

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

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountId = p\_SourceAccountId;

-- Add to target

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountId = p\_TargetAccountId;

COMMIT;

END;

/