**HandsOn Mandatory Excercises**

**Module 3 - PL/SQL Programming**

**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.

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

-- Schema to be Created

CREATE TABLE CUSTOMERS (

CUSTOMERID NUMBER PRIMARY KEY,

NAME VARCHAR2(100),

DOB DATE,

BALANCE NUMBER,

LASTMODIFIED DATE

);

CREATE TABLE ACCOUNTS (

ACCOUNTID NUMBER PRIMARY KEY,

CUSTOMERID NUMBER,

ACCOUNTTYPE VARCHAR2(20),

BALANCE NUMBER,

LASTMODIFIED DATE,

FOREIGN KEY ( CUSTOMERID )

REFERENCES CUSTOMERS ( CUSTOMERID )

);

CREATE TABLE TRANSACTIONS (

TRANSACTIONID NUMBER PRIMARY KEY,

ACCOUNTID NUMBER,

TRANSACTIONDATE DATE,

AMOUNT NUMBER,

TRANSACTIONTYPE VARCHAR2(10),

FOREIGN KEY ( ACCOUNTID )

REFERENCES ACCOUNTS ( ACCOUNTID )

);

CREATE TABLE LOANS (

LOANID NUMBER PRIMARY KEY,

CUSTOMERID NUMBER,

LOANAMOUNT NUMBER,

INTERESTRATE NUMBER,

STARTDATE DATE,

ENDDATE DATE,

FOREIGN KEY ( CUSTOMERID )

REFERENCES CUSTOMERS ( CUSTOMERID )

);

CREATE TABLE EMPLOYEES (

EMPLOYEEID NUMBER PRIMARY KEY,

NAME VARCHAR2(100),

POSITION VARCHAR2(50),

SALARY NUMBER,

DEPARTMENT VARCHAR2(50),

HIREDATE DATE

);

**SOLUTION:**

**Scenario 1:**

DECLARE

CURSOR c\_senior\_loans IS

SELECT l.LoanID, l.InterestRate

FROM Loans l

JOIN Customers c

ON l.CustomerID = c.CustomerID

WHERE FLOOR(MONTHS\_BETWEEN(SYSDATE, c.DOB)/12) > 60;

BEGIN

FOR rec IN c\_senior\_loans LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

END LOOP;

COMMIT;

END;

/

**Scenario 2:**

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

DECLARE

CURSOR c\_senior\_loans IS

SELECT l.LoanID, l.InterestRate

FROM Loans l

JOIN Customers c

ON l.CustomerID = c.CustomerID

WHERE FLOOR(MONTHS\_BETWEEN(SYSDATE, c.DOB)/12) > 60;

BEGIN

FOR rec IN c\_senior\_loans LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

END LOOP;

COMMIT;

END;

/

**Scenario 3:**

DECLARE

CURSOR c\_due\_loans IS

SELECT l.LoanID, l.EndDate, c.Name

FROM Loans l

JOIN Customers c

ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR rec IN c\_due\_loans LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Dear ' || rec.Name ||

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

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

);

END LOOP;

END;

/

**Exercise 3: Stored Procedures**

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

o 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.

o 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.

o 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:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01,

LastModified = SYSDATE

WHERE AccountType = 'SAVINGS';

COMMIT;

END ProcessMonthlyInterest;

/

**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_Department IN VARCHAR2,

p\_BonusPct IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE Department = p\_Department;

COMMIT;

END UpdateEmployeeBonus;

/

**SCENARIO 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_SourceAccountID IN NUMBER,

p\_DestinationAccountID IN NUMBER,

p\_Amount IN NUMBER

) IS

v\_SourceBalance NUMBER;

BEGIN

-- check available 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.');

ELSE

-- debit source account

UPDATE Accounts

SET Balance = Balance - p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_SourceAccountID;

-- credit destination account

UPDATE Accounts

SET Balance = Balance + p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_DestinationAccountID;

COMMIT;

END IF;

END TransferFunds;

/