**PL/SQL**

**Exercise 1: Control Structures**

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

**Query:**

BEGIN

FOR rec IN (

SELECT l.LoanID, c.Name, l.InterestRate

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

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

)

LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

DBMS\_OUTPUT.PUT\_LINE('1% discount applied to ' || rec.Name ||

' (Loan ID: ' || rec.LoanID || ')');

END LOOP;

END;

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

**Query:**

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

BEGIN

FOR rec IN (

SELECT CustomerID, Name

FROM Customers

WHERE Balance > 10000

)

LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ' || rec.Name || ' is marked as VIP');

END LOOP;

END;

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

**Query:**

BEGIN

FOR rec IN (

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

FROM Customers c

JOIN Loans l ON c.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

)

LOOP

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

', your loan (ID: ' || 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**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

)

LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('1% interest added to Account ID: ' || acc.AccountID);

END LOOP;

END;

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

**Query:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

FOR emp IN (

SELECT EmployeeID, Salary

FROM Employees

WHERE Department = p\_department

)

LOOP

UPDATE Employees

SET Salary = Salary + (Salary \* p\_bonus\_percent / 100)

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Employee ID: ' || emp.EmployeeID);

END LOOP;

END;

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

**Query:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_from\_balance NUMBER;

BEGIN

-- Lock and check the source account balance

SELECT Balance INTO v\_from\_balance

FROM Accounts

WHERE AccountID = p\_from\_account

FOR UPDATE;

IF v\_from\_balance < p\_amount THEN

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

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_to\_account;

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (3, p\_from\_account, SYSDATE, p\_amount, 'Transfer Out');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (4, p\_to\_account, SYSDATE, p\_amount, 'Transfer In');

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_amount || ' from Account ' || p\_from\_account || ' to Account ' || p\_to\_account);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One or both account IDs do not exist.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

ROLLBACK;

END;