**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.

BEGIN

FOR cust IN (

SELECT CustomerID,

TRUNC(MONTHS\_BETWEEN(SYSDATE, DOB)/12) AS age

FROM Customers

) LOOP

IF cust.age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE(

'Customer '||cust.CustomerID

||' age '||cust.age

||': 1% discount applied.'

);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Customer '||cust.CustomerID ||' age '||cust.age ||': no discount.');

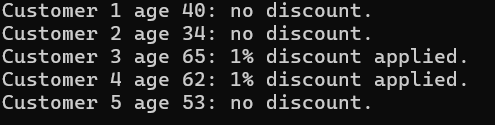
END IF;

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.

BEGIN

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

IF c.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = c.CustomerID;

DBMS\_OUTPUT.PUT\_LINE( 'Customer '||c.CustomerID||' balance '||c.Balance ||': marked VIP.'

);

ELSE

UPDATE Customers

SET IsVIP = 'N'

WHERE CustomerID = c.CustomerID;

DBMS\_OUTPUT.PUT\_LINE(

'Customer '||c.CustomerID

||' balance '||c.Balance

||': not VIP.'

);

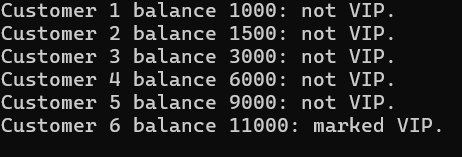
END IF;

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.

SET SERVEROUTPUT ON SIZE UNLIMITED

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Generating reminders: '||TO\_CHAR(SYSDATE,'YYYY-MM-DD HH24:MI:SS')||' ---');

DECLARE

v\_count PLS\_INTEGER := 0;

BEGIN

FOR r IN (

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

FROM Loans l

JOIN Customers c ON c.CustomerID = l.CustomerID

WHERE TRUNC(l.EndDate) BETWEEN TRUNC(SYSDATE) AND TRUNC(SYSDATE)+30

) LOOP

v\_count := v\_count + 1;

DBMS\_OUTPUT.PUT\_LINE('['||v\_count||'] Reminder → '||'Cust#'||r.CustomerID ||' "'||r.Name||'"'||', Loan#'||r.LoanID ||', due on '||TO\_CHAR(r.EndDate,'YYYY-MM-DD')

);

END LOOP;

IF v\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('No loans due in the next 30 days.');

ELSE

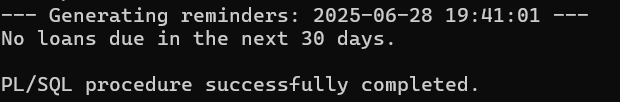
DBMS\_OUTPUT.PUT\_LINE('Total reminders sent: '||v\_count);

END IF;

END;

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.

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01,

LastModified = SYSDATE

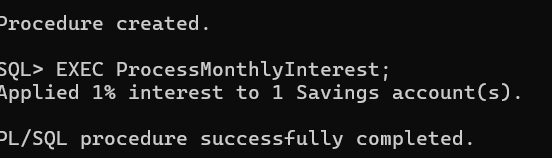
WHERE AccountType = 'Savings';

DBMS\_OUTPUT.PUT\_LINE('Applied 1% interest to '|| SQL%ROWCOUNT|| ' Savings account(s).' );

COMMIT;

END 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.

SET SERVEROUTPUT ON SIZE UNLIMITED

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept IN Employees.Department%TYPE, p\_bonusPct IN NUMBER) IS v\_count PLS\_INTEGER;

BEGIN

UPDATE Employees

SET Salary = Salary \* (1 + p\_bonusPct/100)

WHERE Department = p\_dept;

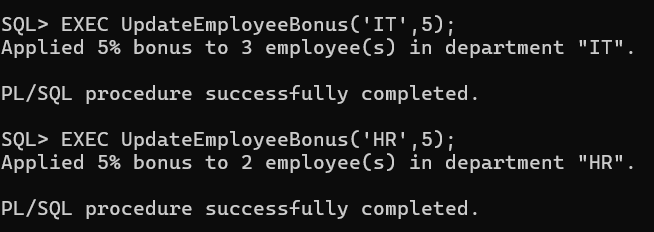
v\_count := SQL%ROWCOUNT;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Applied ' || p\_bonusPct || '% bonus to '|| v\_count || ' employee(s) in department "'|| p\_dept ||' ".');

END UpdateEmployeeBonus;

/



**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.

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_srcAcct IN NUMBER,

p\_dstAcct IN NUMBER,

p\_amount IN NUMBER

) IS

v\_srcBal NUMBER;

BEGIN

SELECT Balance

INTO v\_srcBal

FROM Accounts

WHERE AccountID = p\_srcAcct

FOR UPDATE;

IF v\_srcBal < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20003,'Insufficient funds in source account.');

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_srcAcct;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_dstAcct;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || p\_amount || ' moved from Account ' || p\_srcAcct || ' to Account ' || p\_dstAcct );

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

RAISE;

END TransferFunds;

/

