PL SQL PROGRAMMING

**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.

**QUERY**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

v\_new\_balance NUMBER;

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

v\_new\_balance := acc.Balance \* 1.01;

UPDATE Accounts

SET Balance = v\_new\_balance,

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Updated AccountID ' || acc.AccountID || ' New Balance: ' ||

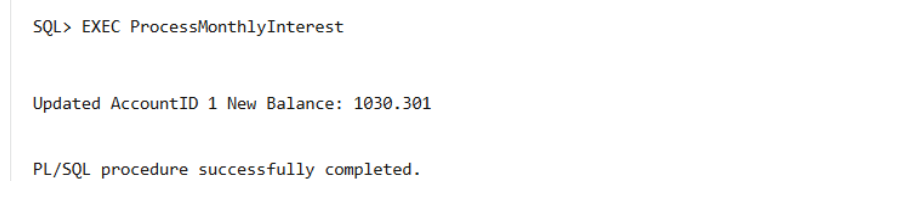
v\_new\_balance);

END LOOP;

END;

/

**OUTPUT**

****

**EXEC 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.

**QUERY**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

FOR emp IN (

SELECT EmployeeID, Name, Salary

FROM Employees

WHERE Department = p\_department

) LOOP

DECLARE

v\_new\_salary NUMBER;

BEGIN

v\_new\_salary := emp.Salary + (emp.Salary \* p\_bonus\_percent / 100);

UPDATE Employees

SET Salary = v\_new\_salary

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE(

'EmployeeID: ' || emp.EmployeeID ||

', Name: ' || emp.Name ||

', New Salary: ' || v\_new\_salary

);

END;

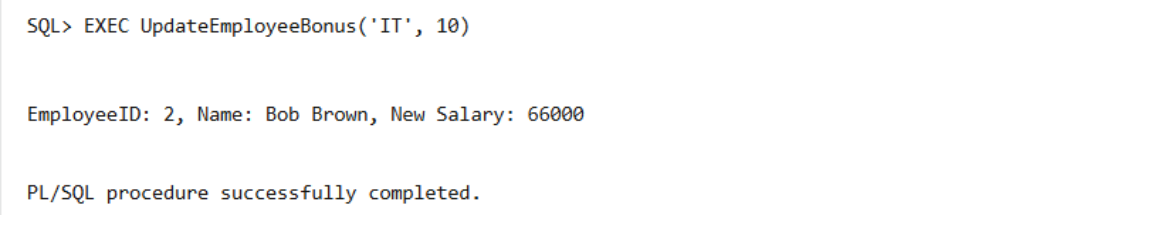
END LOOP;

END;

/

EXEC UpdateEmployeeBonus(‘IT’, 10);

**OUTPUT**

****

**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.

**QUERY**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER) AS

v\_from\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_from\_balance

FROM Accounts

WHERE AccountID = p\_from\_account\_id;

IF v\_from\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance. Transfer failed.');

RETURN;

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account\_id;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account\_id;

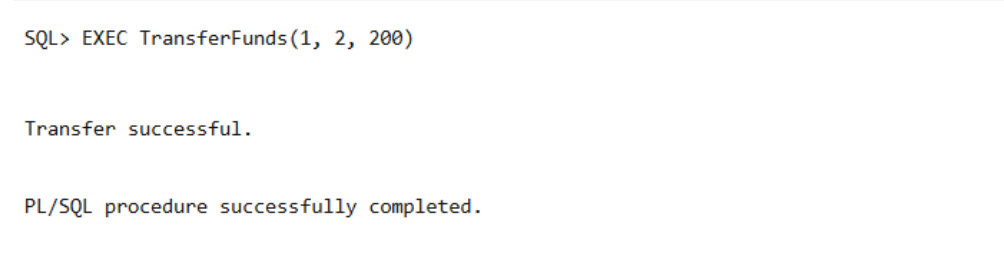
DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

END;

/

EXEC TransferFunds(1, 2, 200);

**OUTPUT**

****