**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

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest processed for all savings accounts.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END ProcessMonthlyInterest;

**To display the output:**

**Query:**

BEGIN

ProcessMonthlyInterest;

END;

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings';

**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\_percentage IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE Department = p\_department;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salaries updated with a bonus of ' || p\_bonus\_percentage || '% for department: ' || p\_department);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END UpdateEmployeeBonus;

**To display the output:**

**Query:**

BEGIN

UpdateEmployeeBonus('HR', 5);

END;

SELECT EmployeeID, Name, Salary

FROM Employees

WHERE Department = 'HR';

**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\_source\_account\_id IN NUMBER,

p\_target\_account\_id IN NUMBER,

p\_amount IN NUMBER

) AS

v\_source\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_source\_balance

FROM Accounts

WHERE AccountID = p\_source\_account\_id;

IF v\_source\_balance < p\_amount THEN

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

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_source\_account\_id;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_target\_account\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Successfully transferred ' || p\_amount || ' from account ' || p\_source\_account\_id || ' to account ' || p\_target\_account\_id);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END TransferFunds;

**To display the output:**

**Query:**

BEGIN

TransferFunds(1, 2, 100);

END;

SELECT AccountID, Balance

FROM Accounts

WHERE AccountID IN (1, 2);