**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.

-- Create or replace the ProcessMonthlyInterest procedure

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

-- Update the balance of all savings accounts by applying a 1% interest rate

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

-- Commit the transaction to save the changes

COMMIT;

-- Optionally, log the successful application of interest

DBMS\_OUTPUT.PUT\_LINE('Monthly interest of 1% applied to all savings accounts.');

EXCEPTION

WHEN OTHERS THEN

-- Log any errors encountered during the process

INSERT INTO TransactionErrors (ErrorDate, ErrorMessage)

VALUES (SYSDATE, 'Error processing monthly interest: ' || SQLERRM);

-- Rollback the transaction in case of an error

ROLLBACK;

-- Propagate the exception

RAISE;

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.

-- Create or replace the UpdateEmployeeBonus procedure

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department IN Employees.Department%TYPE,

p\_BonusPercentage IN NUMBER

) AS

BEGIN

-- Update the salary of employees in the given department by adding the bonus percentage

UPDATE Employees

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

WHERE Department = p\_Department;

-- Commit the transaction to save the changes

COMMIT;

-- Optionally, log the successful application of the bonus

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_BonusPercentage || '% applied to all employees in department ' || p\_Department || '.');

EXCEPTION

WHEN OTHERS THEN

-- Log any errors encountered during the process

INSERT INTO TransactionErrors (ErrorDate, ErrorMessage)

VALUES (SYSDATE, 'Error applying bonus: ' || SQLERRM);

-- Rollback the transaction in case of an error

ROLLBACK;

-- Propagate the exception

RAISE;

END UpdateEmployeeBonus;

/

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/

**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 the TransferFunds procedure

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccountID IN Accounts.AccountID%TYPE,

p\_ToAccountID IN Accounts.AccountID%TYPE,

p\_Amount IN NUMBER

) AS

v\_FromBalance Accounts.Balance%TYPE;

BEGIN

-- Lock the rows for update to ensure data integrity

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID

FOR UPDATE;

-- Check if the source account has sufficient funds

IF v\_FromBalance < p\_Amount THEN

-- Log the error and raise an exception if insufficient funds

INSERT INTO TransactionErrors (ErrorDate, ErrorMessage)

VALUES (SYSDATE, 'Insufficient funds in account ID ' || p\_FromAccountID);

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

ELSE

-- Deduct the amount from the source account

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

-- Add the amount to the destination account

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

-- Commit the transaction to save the changes

COMMIT;

-- Optionally, log the successful transfer

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || p\_Amount || ' from Account ' || p\_FromAccountID || ' to Account ' || p\_ToAccountID || ' completed successfully.');

END IF;

EXCEPTION

WHEN OTHERS THEN

-- Rollback the transaction in case of any error

ROLLBACK;

-- Log the error

INSERT INTO TransactionErrors (ErrorDate, ErrorMessage)

VALUES (SYSDATE, 'Error during fund transfer: ' || SQLERRM);

-- Propagate the exception

RAISE;

END TransferFunds;

/

BEGIN

TransferFunds(1, 2, 500); -- Transfer 500 from account ID 1 to account ID 2

END;

/