**Exercise 3: Stored Procedures**

**Scenario 1: Processing Monthly Interest for All Savings Accounts**

**Stored Procedure: ProcessMonthlyInterest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

WHERE accounttype = 'Savings';

COMMIT; -- Commit the changes to the database

END ProcessMonthlyInterest;

/

**Scenario 2: Implementing a Bonus Scheme for Employees**

**Stored Procedure: UpdateEmployeeBonus**

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; -- Commit the changes to the database

END UpdateEmployeeBonus;

/

**Scenario 3: Transferring Funds Between Customer Accounts**

**Stored Procedure: TransferFunds**

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 accounts.balance%TYPE;

BEGIN

-- Start transaction

SAVEPOINT sp\_transfer;

-- Check the balance of the source account

SELECT balance INTO v\_from\_balance

FROM accounts

WHERE accountid = p\_from\_account\_id

FOR UPDATE;

IF v\_from\_balance < p\_amount THEN

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

END IF;

-- Update the source account balance

UPDATE accounts

SET balance = balance - p\_amount

WHERE accountid = p\_from\_account\_id;

-- Update the destination account balance

UPDATE accounts

SET balance = balance + p\_amount

WHERE accountid = p\_to\_account\_id;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK TO sp\_transfer;

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

-- Log the error in a log table (optional)

-- INSERT INTO error\_log (message, created\_at) VALUES (SQLERRM, SYSDATE);

END TransferFunds;

/