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 PROCEDURE ProcessMonthlyInterest AS

BEGIN

-- Start a transaction

BEGIN

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

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'savings';

-- Commit the transaction

COMMIT;

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

EXCEPTION

WHEN OTHERS THEN

-- Handle any exceptions

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

-- Rollback in case of error

ROLLBACK;

END;

END;

/

**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 PROCEDURE UpdateEmployeeBonus(

p\_department\_id IN NUMBER,

p\_bonus\_percentage IN NUMBER

) AS

BEGIN

-- Start a transaction

BEGIN

-- Update salaries for employees in the given department with a bonus

UPDATE employees

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

WHERE department\_id = p\_department\_id;

-- Commit the transaction

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to employees in department ' || p\_department\_id);

EXCEPTION

WHEN OTHERS THEN

-- Handle any exceptions

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

-- Rollback in case of error

ROLLBACK;

END;

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 PROCEDURE TransferFunds(

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) AS

v\_from\_balance NUMBER;

v\_to\_balance NUMBER;

BEGIN

-- Start a transaction

BEGIN

-- Lock the accounts to prevent concurrent updates

SELECT balance INTO v\_from\_balance

FROM accounts

WHERE account\_id = p\_from\_account\_id

FOR UPDATE;

SELECT balance INTO v\_to\_balance

FROM accounts

WHERE account\_id = p\_to\_account\_id

FOR UPDATE;

-- Check for sufficient funds

IF v\_from\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in account ' || p\_from\_account\_id);

END IF;

-- Perform the transfer

UPDATE accounts

SET balance = v\_from\_balance - p\_amount

WHERE account\_id = p\_from\_account\_id;

UPDATE accounts

SET balance = v\_to\_balance + p\_amount

WHERE account\_id = p\_to\_account\_id;

-- Commit the transaction

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Funds transferred from account ' || p\_from\_account\_id || ' to account ' || p\_to\_account\_id);

EXCEPTION

WHEN OTHERS THEN

-- Rollback in case of error

ROLLBACK;

-- Log the error message

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

-- Optionally, re-raise the exception if needed

RAISE;

END;

END;

/