**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.

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.

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.

Here are the PL/SQL stored procedures for the given scenarios:

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

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

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

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'Savings';

COMMIT;

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

END ProcessMonthlyInterest;

/

**Scenario 2: Update Employee Bonus Based on Performance**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department\_id IN NUMBER,

p\_bonus\_percentage IN NUMBER

) IS

BEGIN

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

UPDATE employees

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

WHERE department\_id = p\_department\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Employee bonuses updated for department ' || p\_department\_id || '.');

END UpdateEmployeeBonus;

/

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

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) IS

insufficient\_funds EXCEPTION;

BEGIN

-- Check if the from account has sufficient funds

DECLARE

v\_from\_balance NUMBER;

BEGIN

SELECT balance INTO v\_from\_balance

FROM accounts

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

FOR UPDATE;

IF v\_from\_balance < p\_amount THEN

RAISE insufficient\_funds;

END IF;

-- Deduct the amount from the from account

UPDATE accounts

SET balance = balance - p\_amount

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

-- Add the amount to the to account

UPDATE accounts

SET balance = balance + p\_amount

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

COMMIT;

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

EXCEPTION

WHEN insufficient\_funds THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in account ' || p\_from\_account\_id);

WHEN OTHERS THEN

ROLLBACK;

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

END;

END TransferFunds;

/

Each of these stored procedures accomplishes the task described in the respective scenario by performing the necessary updates or checks, handling exceptions, and ensuring that the transactions are committed or rolled back appropriately.