**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 account\_type = 'Savings';

COMMIT;

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

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred during monthly interest processing. Transaction rolled back.');

END ProcessMonthlyInterest;

**Explanation:**

We update the balance of all savings accounts by multiplying the current balance by 1.01, which represents a 1% interest rate increase. The accounts updated are specifically those with the account\_type 'Savings'. Error handling is included to handle any unexpected issues during the interest calculation process. The transaction is committed if the update is successful, ensuring data consistency, and rolled back in case of errors to maintain data integrity.

**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 (department\_id IN NUMBER, bonus\_percentage IN NUMBER) AS

BEGIN

UPDATE Employees

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

WHERE department\_id = department\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Employee bonuses updated successfully for the specified department.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred while updating employee bonuses. Transaction rolled back.');

END UpdateEmployeeBonus;

**Explanation:**

We update the salary of employees in the specified department by multiplying the current salary by (1 + bonus\_percentage/100) to add the bonus percentage. The update is applied to employees in the department identified by the department\_id parameter. Error handling is implemented to manage any unexpected issues during the bonus update process. The transaction is committed if the update is successful to maintain data consistency, and rolled back in case of errors to preserve data integrity.

**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 (source\_account\_id IN NUMBER, target\_account\_id IN NUMBER, amount IN NUMBER) AS

source\_balance NUMBER;

BEGIN

-- Check if the source account has sufficient balance

SELECT balance INTO source\_balance

FROM Accounts

WHERE account\_id = source\_account\_id;

IF source\_balance >= amount THEN

-- Deduct the amount from the source account

UPDATE Accounts

SET balance = balance - amount

WHERE account\_id = source\_account\_id;

-- Add the amount to the target account

UPDATE Accounts

SET balance = balance + amount

WHERE account\_id = target\_account\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Funds transferred successfully from Account ' || source\_account\_id || ' to Account ' || target\_account\_id);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in the source account for the transfer.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One or both account IDs are invalid.');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred during fund transfer. Transaction rolled back.');

END TransferFunds;

**Explanation:**

We first check if the source account has sufficient balance to cover the transfer amount. If the source account has enough balance, the specified amount is deducted from the source account and added to the target account. Error handling is included to address scenarios such as invalid account IDs or unexpected errors during the transfer process. The transaction is committed if the transfer is successful, ensuring data consistency, and rolled back in case of errors to maintain data integrity.