**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.

**CODE:**

-- Create the stored procedure

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc\_rec IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountID = acc\_rec.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID: ' || acc\_rec.AccountID);

END LOOP;

COMMIT;

END;

/

-- Execute the stored procedure

BEGIN

ProcessMonthlyInterest;

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.

**CODE:**

-- Create the stored procedure

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

FOR emp\_rec IN (

SELECT EmployeeID, Salary

FROM Employees

WHERE Department = dept\_name

) LOOP

UPDATE Employees

SET Salary = Salary + (Salary \* (bonus\_percent / 100))

WHERE EmployeeID = emp\_rec.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Employee ID: ' || emp\_rec.EmployeeID);

END LOOP;

COMMIT;

END;

/

-- Execute the stored procedure for 'IT' department with 10% bonus

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.

**CODE:**

-- Create the stored procedure

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

transfer\_amount IN NUMBER

) AS

insufficient\_funds EXCEPTION;

src\_balance NUMBER;

BEGIN

-- Get current balance of source account

SELECT Balance INTO src\_balance

FROM Accounts

WHERE AccountID = from\_account\_id;

-- Check if sufficient funds are available

IF src\_balance < transfer\_amount THEN

RAISE insufficient\_funds;

END IF;

-- Deduct from source account

UPDATE Accounts

SET Balance = Balance - transfer\_amount,

LastModified = SYSDATE

WHERE AccountID = from\_account\_id;

-- Add to destination account

UPDATE Accounts

SET Balance = Balance + transfer\_amount,

LastModified = SYSDATE

WHERE AccountID = to\_account\_id;

-- Log success

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || transfer\_amount ||

' from Account ID ' || from\_account\_id ||

' to Account ID ' || to\_account\_id);

COMMIT;

EXCEPTION

WHEN insufficient\_funds THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in Account ID ' || from\_account\_id);

WHEN OTHERS THEN

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

ROLLBACK;

END;

/

-- Execute the stored procedure (Example: transfer 200 from account 1 to account 2)

BEGIN

TransferFunds(1, 2, 200);

END;

/