**Scenario 1:** The bank needs to process monthly interest for all savings accounts. 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.

**Ans :**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

AS

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01 -- Assuming a 1% interest rate

WHERE AccountType = 'Savings';

END ProcessMonthlyInterest;

**Scenario 2 :** The bank wants to implement a bonus scheme for employees based on their performance. Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Ans :**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department VARCHAR2,

p\_bonus\_percentage NUMBER

)

AS

BEGIN

UPDATE Employees

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

WHERE Department = p\_department;

END UpdateEmployeeBonus**;**

**Scenario 3 :** Customers should be able to transfer funds between their accounts. 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.

**Ans :**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account\_id NUMBER,

p\_destination\_account\_id NUMBER,

p\_amount NUMBER

)

AS

v\_source\_balance NUMBER;

v\_source\_customer\_id NUMBER;

v\_destination\_customer\_id NUMBER;

BEGIN

-- Check if source and destination accounts exist and get their customer IDs

SELECT CustomerID INTO v\_source\_customer\_id FROM Accounts WHERE AccountID = p\_source\_account\_id;

SELECT CustomerID INTO v\_destination\_customer\_id FROM Accounts WHERE AccountID = p\_destination\_account\_id;

-- Check if both accounts belong to the same customer

IF v\_source\_customer\_id != v\_destination\_customer\_id THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Accounts must belong to the same customer');

END IF;

-- Check for sufficient balance in source account

SELECT Balance INTO v\_source\_balance FROM Accounts WHERE AccountID = p\_source\_account\_id;

IF v\_source\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Insufficient funds');

END IF;

-- Update account balances

UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_source\_account\_id;

UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_destination\_account\_id;

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Handle case where account does not exist

RAISE\_APPLICATION\_ERROR(-20003, 'One or both accounts do not exist');

WHEN OTHERS THEN

ROLLBACK;

RAISE;

END TransferFunds;