Scenario 1: ProcessMonthlyInterest - Calculate and Update Balance of All Savings Accounts

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

IS

v\_AccountID Accounts.AccountID%TYPE;

v\_Balance Accounts.Balance%TYPE;

BEGIN

FOR account\_rec IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

v\_AccountID := account\_rec.AccountID;

v\_Balance := account\_rec.Balance;

-- Calculate the new balance with 1% interest

UPDATE Accounts

SET Balance = v\_Balance \* 1.01,

LastModified = SYSDATE

WHERE AccountID = v\_AccountID;

END LOOP;

COMMIT;

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

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

Scenario 2: UpdateEmployeeBonus - Update Salary of Employees Based on Bonus Percentage

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_Department IN Employees.Department%TYPE,

p\_BonusPercentage IN NUMBER

)

IS

v\_EmployeeID Employees.EmployeeID%TYPE;

v\_Salary Employees.Salary%TYPE;

BEGIN

FOR employee\_rec IN (SELECT EmployeeID, Salary FROM Employees WHERE Department = p\_Department) LOOP

v\_EmployeeID := employee\_rec.EmployeeID;

v\_Salary := employee\_rec.Salary;

-- Update the salary with the bonus percentage

UPDATE Employees

SET Salary = v\_Salary + (v\_Salary \* p\_BonusPercentage / 100)

WHERE EmployeeID = v\_EmployeeID;

END LOOP;

COMMIT;

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

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

Scenario 3: TransferFunds - Transfer Funds Between Accounts

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_FromAccountID IN Accounts.AccountID%TYPE,

p\_ToAccountID IN Accounts.AccountID%TYPE,

p\_Amount IN NUMBER

)

IS

v\_FromBalance Accounts.Balance%TYPE;

v\_ToBalance Accounts.Balance%TYPE;

e\_InsufficientFunds EXCEPTION;

BEGIN

-- Check if the source account exists and fetch the balance

SELECT Balance INTO v\_FromBalance FROM Accounts WHERE AccountID = p\_FromAccountID FOR UPDATE;

-- Check for sufficient funds

IF v\_FromBalance < p\_Amount THEN

RAISE e\_InsufficientFunds;

END IF;

-- Perform the fund transfer

UPDATE Accounts

SET Balance = Balance - p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_FromAccountID;

-- Check if the destination account exists and fetch the balance

SELECT Balance INTO v\_ToBalance FROM Accounts WHERE AccountID = p\_ToAccountID FOR UPDATE;

UPDATE Accounts

SET Balance = Balance + p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_ToAccountID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: One or both accounts not found');

WHEN e\_InsufficientFunds THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in the source account');

WHEN OTHERS THEN

ROLLBACK;

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

END;

/