**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.

**PL/SQL Block:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR r IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = r.Balance \* 1.01, -- 1 % interest

LastModified = SYSDATE

WHERE AccountID = r.AccountID;

DBMS\_OUTPUT.PUT\_LINE(

'Account ' || r.AccountID ||

' balance ' || TO\_CHAR(r.Balance) ||

' -> ' || TO\_CHAR(r.Balance \* 1.01)

);

END LOOP;

END;

/

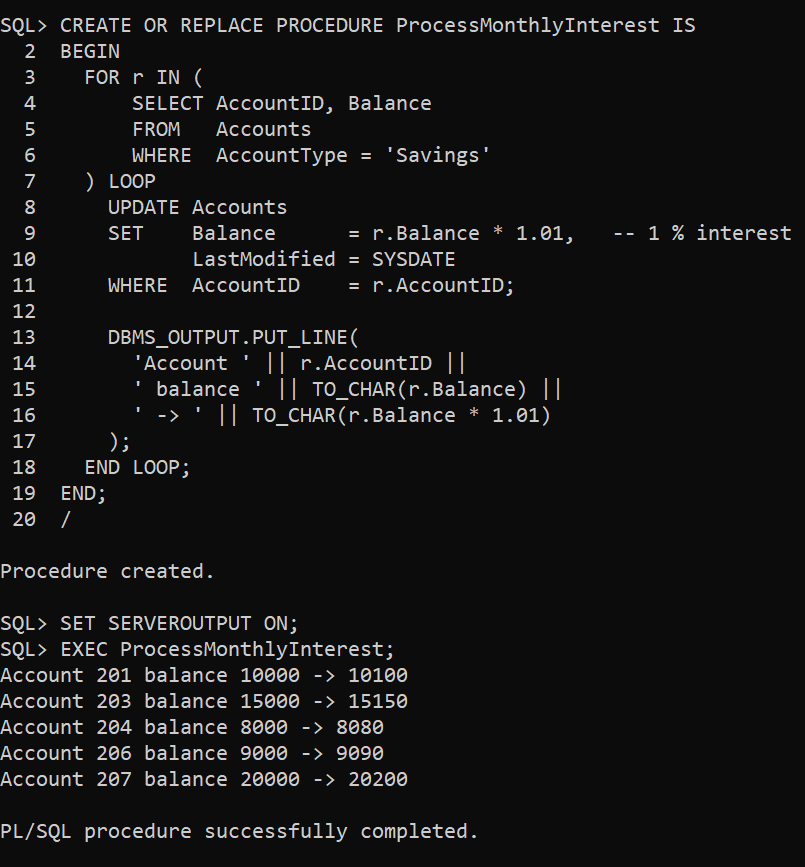
EXEC ProcessMonthlyInterest;

**Table:**

A screen shot of a computer screen

AI-generated content may be incorrect.

**Output:**



**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.

PL/SQL Block:

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

FOR r IN (

SELECT EmployeeID, Salary

FROM Employees

WHERE Department = p\_dept

) LOOP

UPDATE Employees

SET Salary = r.Salary \* (1 + p\_bonus\_pct / 100)

WHERE EmployeeID = r.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE(

'Employee ' || r.EmployeeID ||

' salary ' || TO\_CHAR(r.Salary) ||

' -> ' || TO\_CHAR(r.Salary \* (1 + p\_bonus\_pct / 100))

);

END LOOP;

END;

/

**Table:**

A screenshot of a computer screen

AI-generated content may be incorrect.

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

**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.

**PL/SQL Block:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_source\_acct IN NUMBER,

p\_dest\_acct IN NUMBER,

p\_amount IN NUMBER

) IS

v\_source\_bal NUMBER;

BEGIN

-- Step 1: Get balance from source account

SELECT Balance

INTO v\_source\_bal

FROM Accounts

WHERE AccountID = p\_source\_acct

FOR UPDATE;

-- Step 2: Check for sufficient balance

IF v\_source\_bal < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: insufficient balance in Account ' || p\_source\_acct);

RETURN;

END IF;

-- Step 3: Deduct from source account

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_source\_acct;

-- Step 4: Add to destination account

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_dest\_acct;

-- Step 5: Insert into Transactions table

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES ((SELECT NVL(MAX(TransactionID), 0) + 1 FROM Transactions), p\_source\_acct, SYSDATE, -p\_amount, 'Transfer');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES ((SELECT NVL(MAX(TransactionID), 0) + 1 FROM Transactions), p\_dest\_acct, SYSDATE, p\_amount, 'Transfer');

-- Step 6: Show success

DBMS\_OUTPUT.PUT\_LINE('Transferred ₹' || p\_amount || ' from Account ' || p\_source\_acct || ' to Account ' || p\_dest\_acct);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: invalid account ID(s).');

WHEN OTHERS THEN

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

END;

/

**Table:**

**A screen shot of a computer screen

AI-generated content may be incorrect.**

**Output:**

**A computer screen shot of a program

AI-generated content may be incorrect.**