**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.

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

v\_count NUMBER := 0;

v\_total\_interest NUMBER := 0;

BEGIN

SELECT COUNT(\*), NVL(SUM(Balance \* 0.01), 0)

INTO v\_count, v\_total\_interest

FROM Accounts

WHERE UPPER(AccountType) = 'SAVINGS';

DBMS\_OUTPUT.PUT\_LINE('PROCESSING MONTHLY INTEREST');

DBMS\_OUTPUT.PUT\_LINE('Found ' || v\_count || ' savings accounts');

IF v\_count > 0 THEN

UPDATE Accounts

SET Balance = Balance \* 1.01,

LastModified = SYSDATE

WHERE UPPER(AccountType) = 'SAVINGS';

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Interest rate applied: 1%');

DBMS\_OUTPUT.PUT\_LINE('Total interest added: $' || ROUND(v\_total\_interest, 2));

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

ELSE

DBMS\_OUTPUT.PUT\_LINE('No savings accounts found to process');

END IF;

END ProcessMonthlyInterest;

/

*-- Test the procedure*

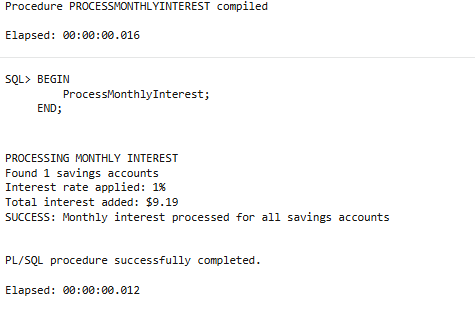
BEGIN

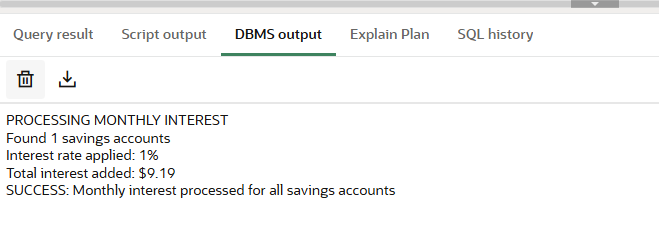
ProcessMonthlyInterest;

END;

/

**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 each department by adding a bonus percentage passed as a parameter.

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_percentage IN NUMBER

) AS

v\_count NUMBER := 0;

v\_total\_bonus NUMBER := 0;

v\_old\_salary\_total NUMBER := 0;

v\_new\_salary\_total NUMBER := 0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('PROCESSING EMPLOYEE BONUS');

DBMS\_OUTPUT.PUT\_LINE('Department: ' || p\_department);

DBMS\_OUTPUT.PUT\_LINE('Bonus Percentage: ' || p\_bonus\_percentage || '%');

SELECT COUNT(\*), NVL(SUM(Salary), 0), NVL(SUM(Salary \* (p\_bonus\_percentage/100)), 0)

INTO v\_count, v\_old\_salary\_total, v\_total\_bonus

FROM Employees

WHERE UPPER(Department) = UPPER(p\_department);

DBMS\_OUTPUT.PUT\_LINE('Employees found in ' || p\_department || ': ' || v\_count);

IF v\_count > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Total current salary: $' || v\_old\_salary\_total);

DBMS\_OUTPUT.PUT\_LINE('Total bonus amount: $' || ROUND(v\_total\_bonus, 2));

UPDATE Employees

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

WHERE UPPER(Department) = UPPER(p\_department);

v\_new\_salary\_total := v\_old\_salary\_total + v\_total\_bonus;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('SUCCESS: Bonus applied to ' || v\_count || ' employees');

DBMS\_OUTPUT.PUT\_LINE('New total salary: $' || ROUND(v\_new\_salary\_total, 2));

ELSE

DBMS\_OUTPUT.PUT\_LINE('No employees found in department: ' || p\_department);

END IF;

END UpdateEmployeeBonus;

/

-- Test the procedure with HR department (10% bonus)

BEGIN

UpdateEmployeeBonus('HR', 10);

END;

/

-- Test with IT department (5% bonus)

BEGIN

UpdateEmployeeBonus('IT', 5);

END;

/

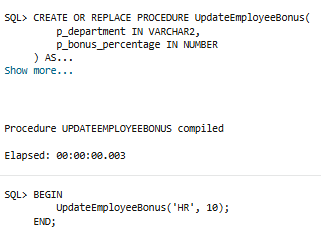
-- Test with non-existing department

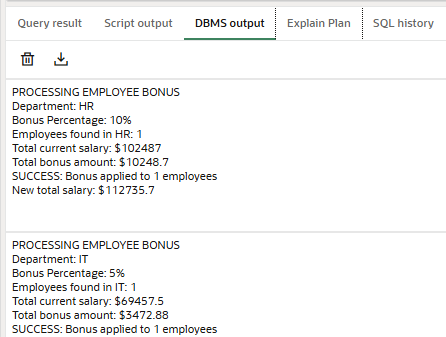
BEGIN

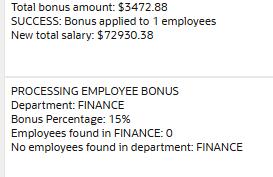
UpdateEmployeeBonus('FINANCE', 15);

END;/

**OUTPUT:**







**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.

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_from\_balance NUMBER;

v\_to\_balance NUMBER;

v\_transaction\_id NUMBER;

v\_from\_customer NUMBER;

v\_to\_customer NUMBER;

v\_from\_count NUMBER := 0;

v\_to\_count NUMBER := 0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('PROCESSING FUND TRANSFER ');

DBMS\_OUTPUT.PUT\_LINE('From Account ID: ' || p\_from\_account);

DBMS\_OUTPUT.PUT\_LINE('To Account ID: ' || p\_to\_account);

DBMS\_OUTPUT.PUT\_LINE('Transfer Amount: $' || p\_amount);

SELECT COUNT(\*) INTO v\_from\_count

FROM Accounts

WHERE AccountID = p\_from\_account;

SELECT COUNT(\*) INTO v\_to\_count

FROM Accounts

WHERE AccountID = p\_to\_account;

IF v\_from\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: Source account ' || p\_from\_account || ' does not exist');

RETURN;

END IF;

IF v\_to\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: Destination account ' || p\_to\_account || ' does not exist');

RETURN;

END IF;

SELECT Balance, CustomerID

INTO v\_from\_balance, v\_from\_customer

FROM Accounts

WHERE AccountID = p\_from\_account;

SELECT Balance, CustomerID

INTO v\_to\_balance, v\_to\_customer

FROM Accounts

WHERE AccountID = p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE('Source account balance: $' || v\_from\_balance);

DBMS\_OUTPUT.PUT\_LINE('Destination account balance: $' || v\_to\_balance);

IF v\_from\_balance >= p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Sufficient balance available. Processing transfer...');

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_to\_account;

SELECT NVL(MAX(TransactionID), 0) + 1 INTO v\_transaction\_id FROM Transactions;

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

VALUES (v\_transaction\_id, p\_from\_account, SYSDATE, p\_amount, 'Transfer Out');

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

VALUES (v\_transaction\_id + 1, p\_to\_account, SYSDATE, p\_amount, 'Transfer In');

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('SUCCESS: Transfer completed!');

DBMS\_OUTPUT.PUT\_LINE('New balance in Account ' || p\_from\_account || ': $' || (v\_from\_balance - p\_amount));

DBMS\_OUTPUT.PUT\_LINE('New balance in Account ' || p\_to\_account || ': $' || (v\_to\_balance + p\_amount));

DBMS\_OUTPUT.PUT\_LINE('Transaction IDs created: ' || v\_transaction\_id || ', ' || (v\_transaction\_id + 1));

ELSE

DBMS\_OUTPUT.PUT\_LINE('FAILED: Insufficient balance');

DBMS\_OUTPUT.PUT\_LINE('Required amount: $' || p\_amount);

DBMS\_OUTPUT.PUT\_LINE('Available balance: $' || v\_from\_balance);

DBMS\_OUTPUT.PUT\_LINE('Shortfall: $' || (p\_amount - v\_from\_balance));

END IF;

END TransferFunds;

/

-- Test the procedure

BEGIN

-- Test 1: Valid transfer

TransferFunds(1, 2, 100);

END;

/

BEGIN

-- Test 2: Insufficient balance

TransferFunds(1, 2, 50000);

END;

/

BEGIN

-- Test 3: Invalid source account

TransferFunds(999, 2, 100);

END;

/

BEGIN

-- Test 4: Invalid destination account

TransferFunds(1, 999, 100);

END;

/

**OUTPUT:**

