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.

**TABLES USED**

CREATE TABLE Accounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    AccountType VARCHAR2(20),

    Balance NUMBER,

    LastModified DATE

);

**INSERTING VALUES INTO THE TABLES**

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 1500, SYSDATE);

**CREATE PROCEDURE**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

CURSOR savings\_cursor IS

SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings';

v\_account\_id Accounts.AccountID%TYPE;

v\_balance Accounts.Balance%TYPE;

v\_interest\_rate CONSTANT NUMBER := 0.01;

BEGIN

OPEN savings\_cursor;

LOOP

FETCH savings\_cursor INTO v\_account\_id, v\_balance;

EXIT WHEN savings\_cursor%NOTFOUND;

UPDATE Accounts

SET Balance = Balance + (v\_balance \* v\_interest\_rate),

LastModified = SYSDATE

WHERE AccountID = v\_account\_id;

END LOOP;

CLOSE savings\_cursor;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

**RUN THE PROCEDURE**

BEGIN

    ProcessMonthlyInterest;

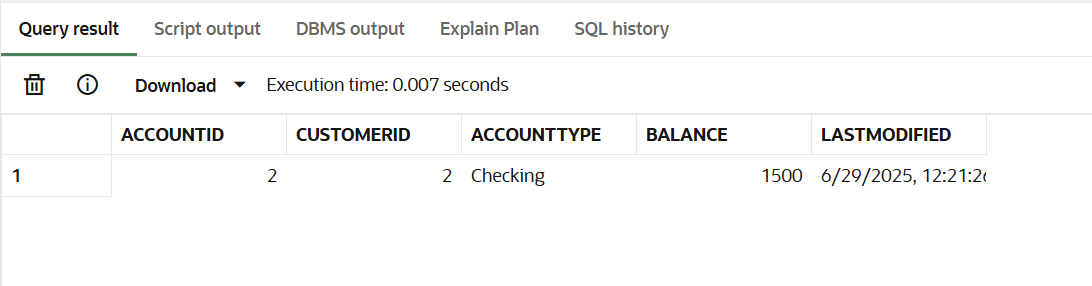
END;

/

**VIEW RESULT**

SELECT \* FROM Accounts;

**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.

**TABLES USED**

CREATE TABLE Employees (

    EmployeeID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    Position VARCHAR2(50),

    Salary NUMBER,

    Department VARCHAR2(50),

    HireDate DATE

);

**INSERTING VALUES**

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

**CREATE PROCEDURE**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

updated\_count NUMBER;

BEGIN

UPDATE Employees

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

WHERE Department = dept\_name;

updated\_count := SQL%ROWCOUNT;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(updated\_count || ' employee(s) received bonus in department ' || dept\_name);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

**RUN THE PROCEDURE**

BEGIN

    UpdateEmployeeBonus('IT', 10);

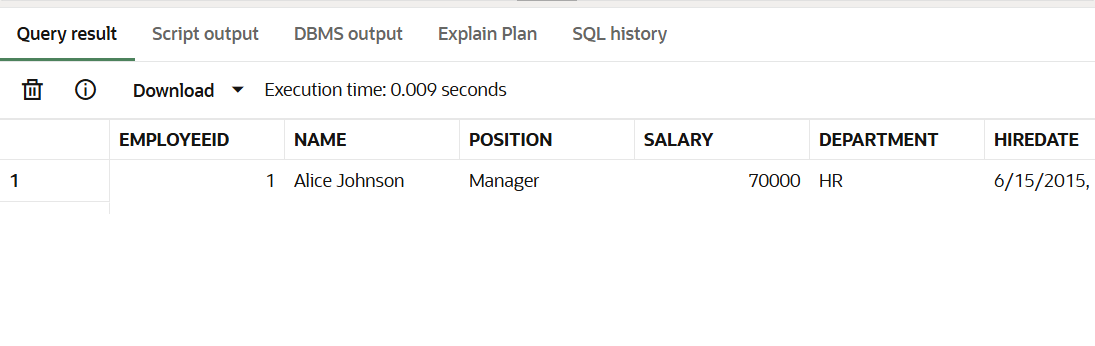
END;

/

**VIEW RESULT**

SELECT \* FROM Employees;

**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.

**TABLES USED**

CREATE TABLE Accounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    AccountType VARCHAR2(20),

    Balance NUMBER,

    LastModified DATE

);

CREATE TABLE Transactions (

    TransactionID NUMBER PRIMARY KEY,

    AccountID NUMBER,

    TransactionDate DATE,

    Amount NUMBER,

    TransactionType VARCHAR2(10)

);

**INSERTING VALUES**

-- Accounts

INSERT INTO Accounts VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Checking', 1500, SYSDATE);

-- Transactions

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

**CREATE PROCEDURE**

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) IS

insufficient\_balance EXCEPTION;

account\_not\_found EXCEPTION;

from\_balance NUMBER;

to\_count NUMBER;

BEGIN

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = from\_account\_id

FOR UPDATE;

SELECT COUNT(\*) INTO to\_count

FROM Accounts

WHERE AccountID = to\_account\_id;

IF to\_count = 0 THEN

RAISE account\_not\_found;

END IF;

IF from\_balance < amount THEN

RAISE insufficient\_balance;

END IF;

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = from\_account\_id;

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = to\_account\_id;

INSERT INTO Transactions

VALUES (Transactions\_seq.NEXTVAL, from\_account\_id, SYSDATE, amount, 'Debit');

INSERT INTO Transactions

VALUES (Transactions\_seq.NEXTVAL, to\_account\_id, SYSDATE, amount, 'Credit');

COMMIT;

EXCEPTION

WHEN insufficient\_balance THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: insufficient funds.');

WHEN account\_not\_found THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: destination account not found.');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Transfer failed due to an unexpected error: ' || SQLERRM);

END;

/

**RUN THE PROCEDURE**

BEGIN

    TransferFunds(1, 2, 300); -- Transfer $300 from account 1 to 2

END;

/

**VIEW RESULT**

SELECT \* FROM Accounts;

SELECT \* FROM Transactions;

**OUTPUT**

