**Exercise 3: Stored Procedures**

**Scenario 1: Process monthly interest for all savings accounts.**

SQL

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

IS

BEGIN

-- Update balance for all savings accounts

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

COMMIT;

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

EXCEPTION

WHEN OTHERS THEN

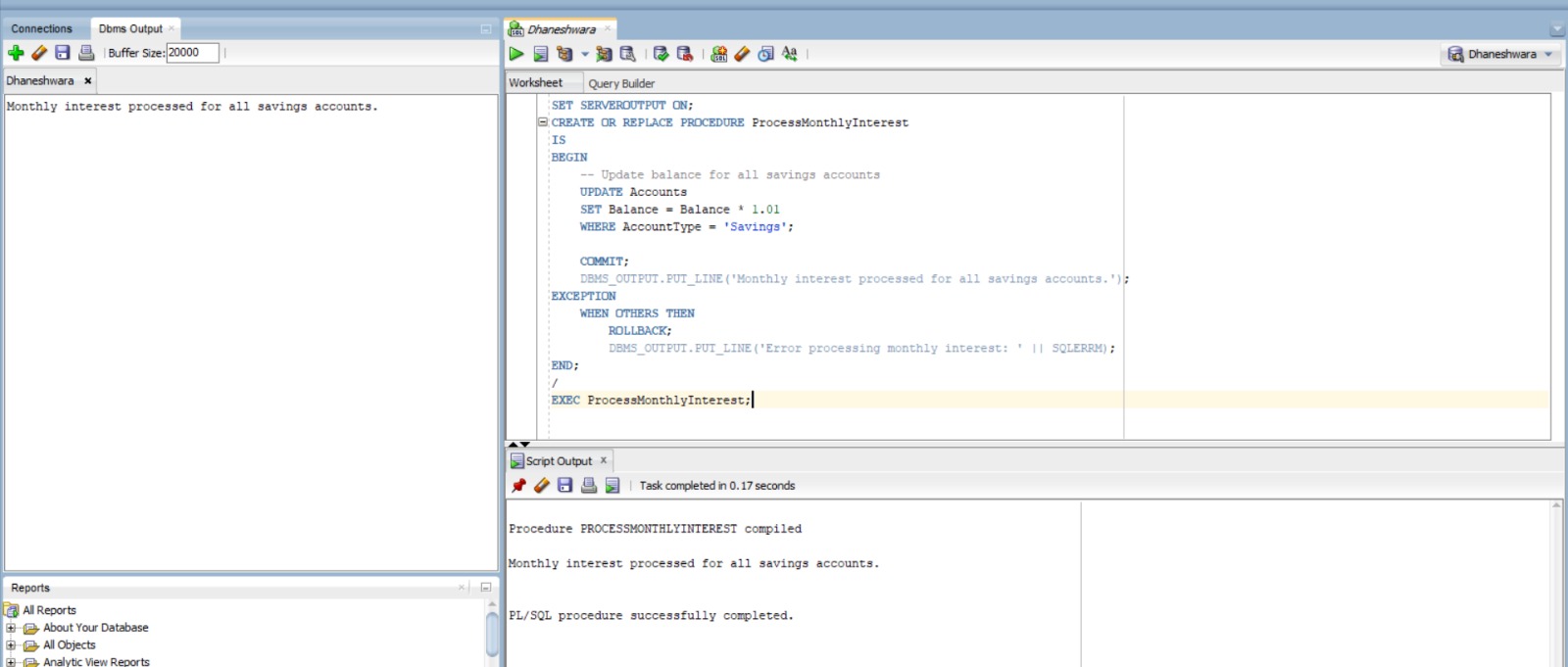
ROLLBACK;

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

END;

/

EXEC ProcessMonthlyInterest;



**Scenario 2: Implement a bonus scheme for employees.**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER -- e.g., pass 10 for 10%

) AS

v\_bonus\_amt NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Bonus Update for Department: ' || p\_department || ' ---');

FOR emp\_rec IN (

SELECT EmployeeID, Department, Salary

FROM Employees

WHERE Department = p\_department

) LOOP

-- Calculate bonus

v\_bonus\_amt := emp\_rec.Salary \* (p\_bonus\_pct / 100);

-- Display ID, Dept, and Bonus

DBMS\_OUTPUT.PUT\_LINE('ID: ' || emp\_rec.EmployeeID ||

' | Dept: ' || emp\_rec.Department ||

' | Bonus: ' || TO\_CHAR(v\_bonus\_amt, '999999.00'));

-- Update salary

UPDATE Employees

SET Salary = Salary + v\_bonus\_amt

WHERE EmployeeID = emp\_rec.EmployeeID;

END LOOP;

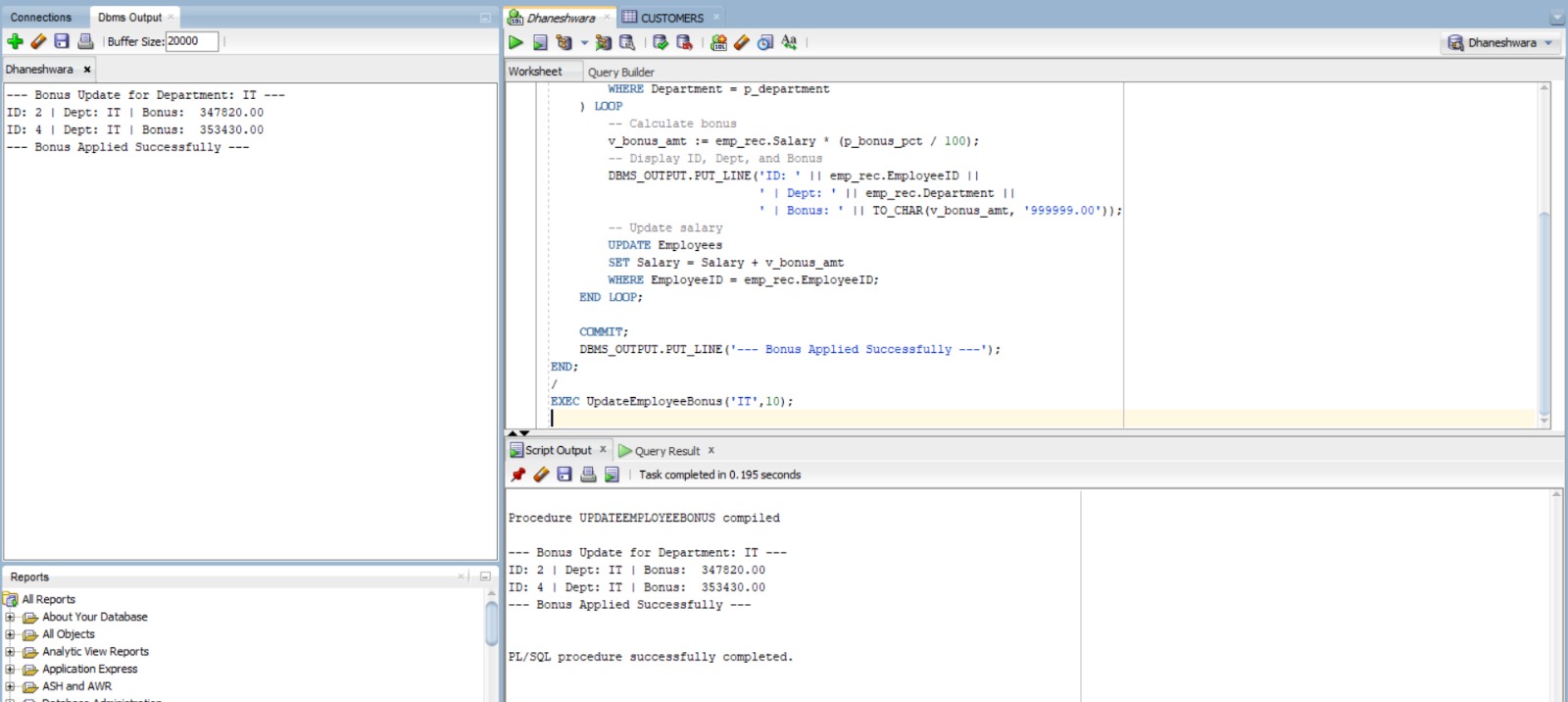
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('--- Bonus Applied Successfully ---');

END;

/

EXEC UpdateEmployeeBonus('IT',10);



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

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account\_id IN NUMBER,

p\_destination\_account\_id IN NUMBER,

p\_amount IN NUMBER

)

IS

v\_source\_balance NUMBER;

e\_insufficient\_funds EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_insufficient\_funds, -20001); -- Custom exception for insufficient funds

BEGIN

-- Check if source account has sufficient balance

SELECT Balance

INTO v\_source\_balance

FROM Accounts

WHERE AccountID = p\_source\_account\_id

FOR UPDATE; -- Lock the row to prevent concurrent updates

IF v\_source\_balance < p\_amount THEN

RAISE e\_insufficient\_funds;

END IF;

-- Deduct from source account

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_source\_account\_id;

-- Add to destination account

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_destination\_account\_id;

-- Record transactions

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

VALUES (TRANSACTIONS\_SEQ.NEXTVAL, p\_source\_account\_id, SYSDATE, p\_amount, 'Withdrawal');

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

VALUES (TRANSACTIONS\_SEQ.NEXTVAL, p\_destination\_account\_id, SYSDATE, p\_amount, 'Deposit');

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Successfully transferred ' || p\_amount || ' from Account ' || p\_source\_account\_id || ' to Account ' || p\_destination\_account\_id || '.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

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

WHEN e\_insufficient\_funds THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in source account ' || p\_source\_account\_id || '.');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred during fund transfer: ' || SQLERRM);

END;

/

BEGIN

TransferFunds(2,3,500);

END;

/

