**Exercise 2: Error Handling**

**Scenario 1:** Handle exceptions during fund transfers between accounts.

* + **Question:** Write a stored procedure **SafeTransferFunds** that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

**Scenario 2:** Manage errors when updating employee salaries.

* + **Question:** Write a stored procedure **UpdateSalary** that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

**Scenario 3:** Ensure data integrity when adding a new customer.

* + **Question:** Write a stored procedure **AddNewCustomer** that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.

**Scenario 1: SafeTransferFunds — Fund Transfer Between Accounts**

**Requirements:**

* Transfer amount from fromAccountId to toAccountId.
* Rollback on any error (e.g., insufficient funds).
* Log appropriate messages using DBMS\_OUTPUT.

**CODE:-**

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

    fromAccountId IN NUMBER,

    toAccountId IN NUMBER,

    amount IN NUMBER

)

IS

    fromBalance NUMBER;

BEGIN

    -- Get balance of source account

    SELECT Balance INTO fromBalance FROM Accounts WHERE AccountID = fromAccountId FOR UPDATE;

    IF fromBalance < amount THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in the source account.');

    END IF;

    -- Deduct amount from source

    UPDATE Accounts

    SET Balance = Balance - amount,

        LastModified = SYSDATE

    WHERE AccountID = fromAccountId;

    -- Add amount to target

    UPDATE Accounts

    SET Balance = Balance + amount,

        LastModified = SYSDATE

    WHERE AccountID = toAccountId;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Transfer completed successfully.');

EXCEPTION

    WHEN OTHERS THEN

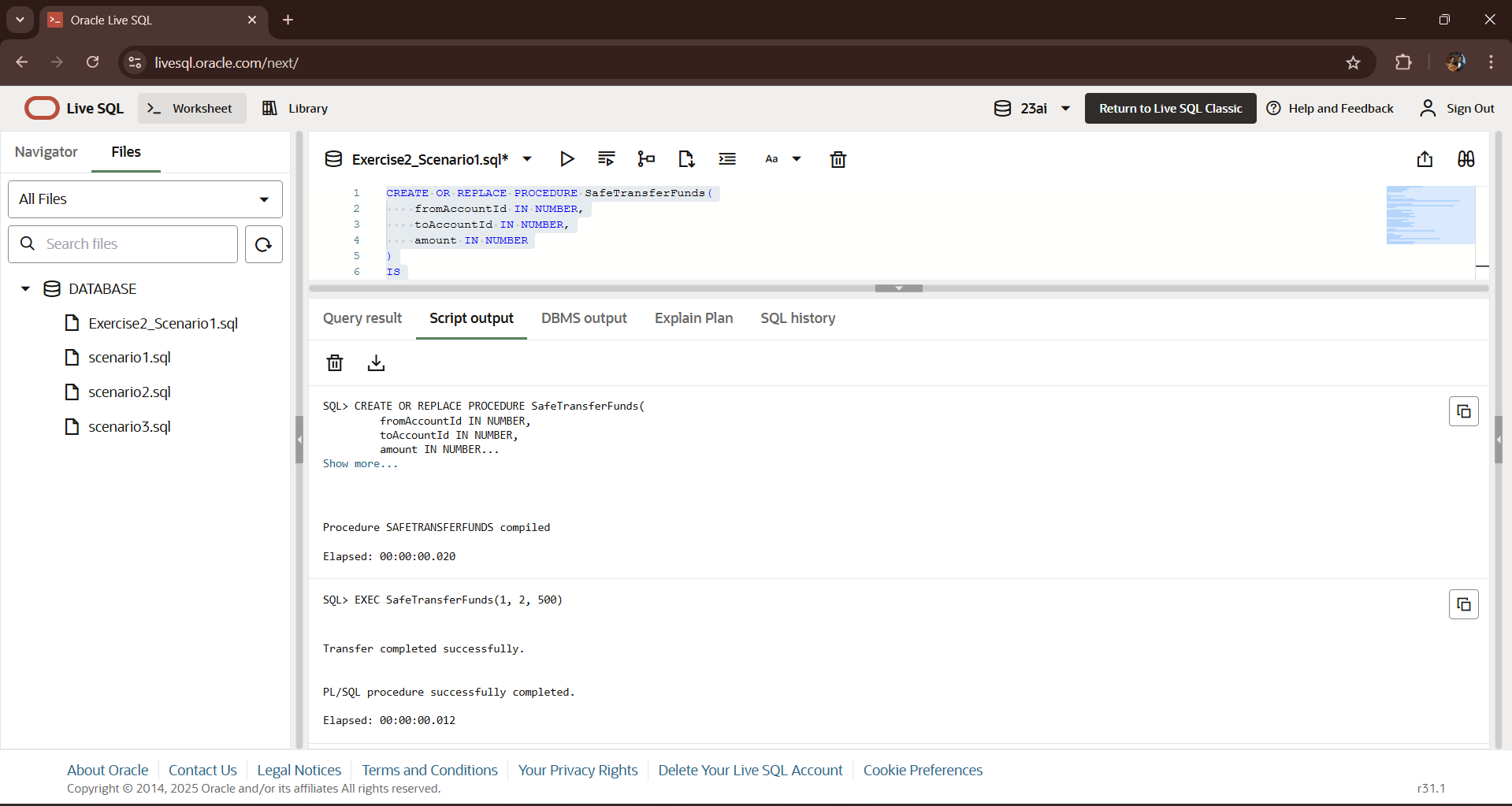
        ROLLBACK;

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

END;

--Example to test SafeTransferFunds:

EXEC SafeTransferFunds(1, 2, 500);

**OUTPUT:-**

**Scenario 2: UpdateSalary — Increase Salary with Error Handling**

**Requirements:**

* Increase salary by a given percentage.
* If employee does not exist, handle it and log error.

**CODE:-**

CREATE OR REPLACE PROCEDURE UpdateSalary(

    empId IN NUMBER,

    percent IN NUMBER

)

IS

BEGIN

    -- Try to update the employee's salary

    UPDATE Employees

    SET Salary = Salary + (Salary \* percent / 100)

    WHERE EmployeeID = empId;

    IF SQL%ROWCOUNT = 0 THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Employee ID ' || empId || ' not found.');

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('Salary updated for Employee ID ' || empId);

        COMMIT;

    END IF;

EXCEPTION

    WHEN OTHERS THEN

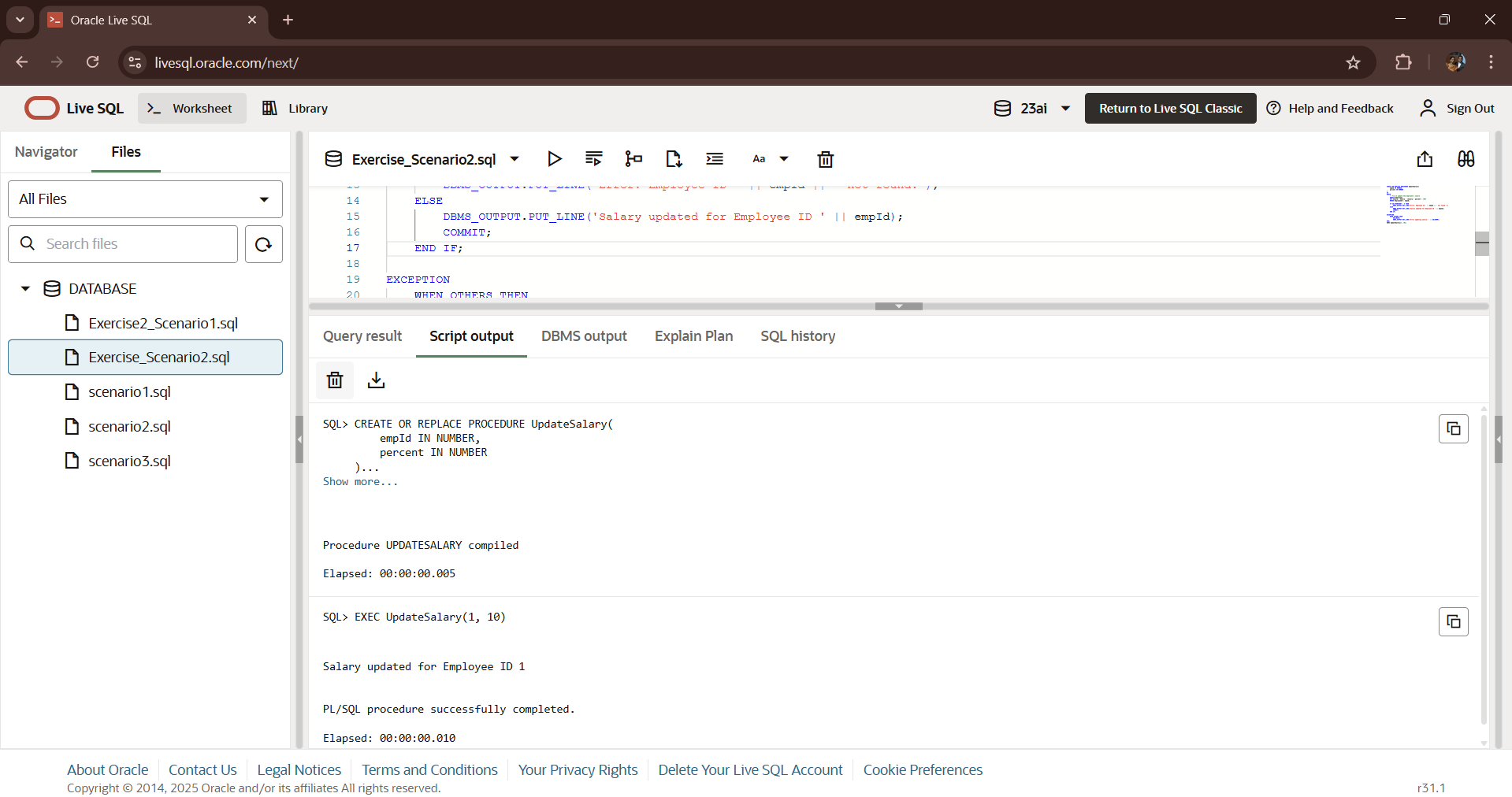
        ROLLBACK;

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

END;

EXEC UpdateSalary(1, 10);

**OUTPUT:-**

****

**Scenario 3: AddNewCustomer — Insert New Customer Safely**

**Requirements:**

* Add customer if ID doesn't exist.
* Handle primary key violation and log message.

**CODE:-** CREATE OR REPLACE PROCEDURE AddNewCustomer(

    p\_CustomerID IN NUMBER,

    p\_Name IN VARCHAR2,

    p\_DOB IN DATE,

    p\_Balance IN NUMBER

)

IS

BEGIN

    INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

    VALUES (p\_CustomerID, p\_Name, p\_DOB, p\_Balance, SYSDATE);

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Customer added successfully.');

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || p\_CustomerID || ' already exists.');

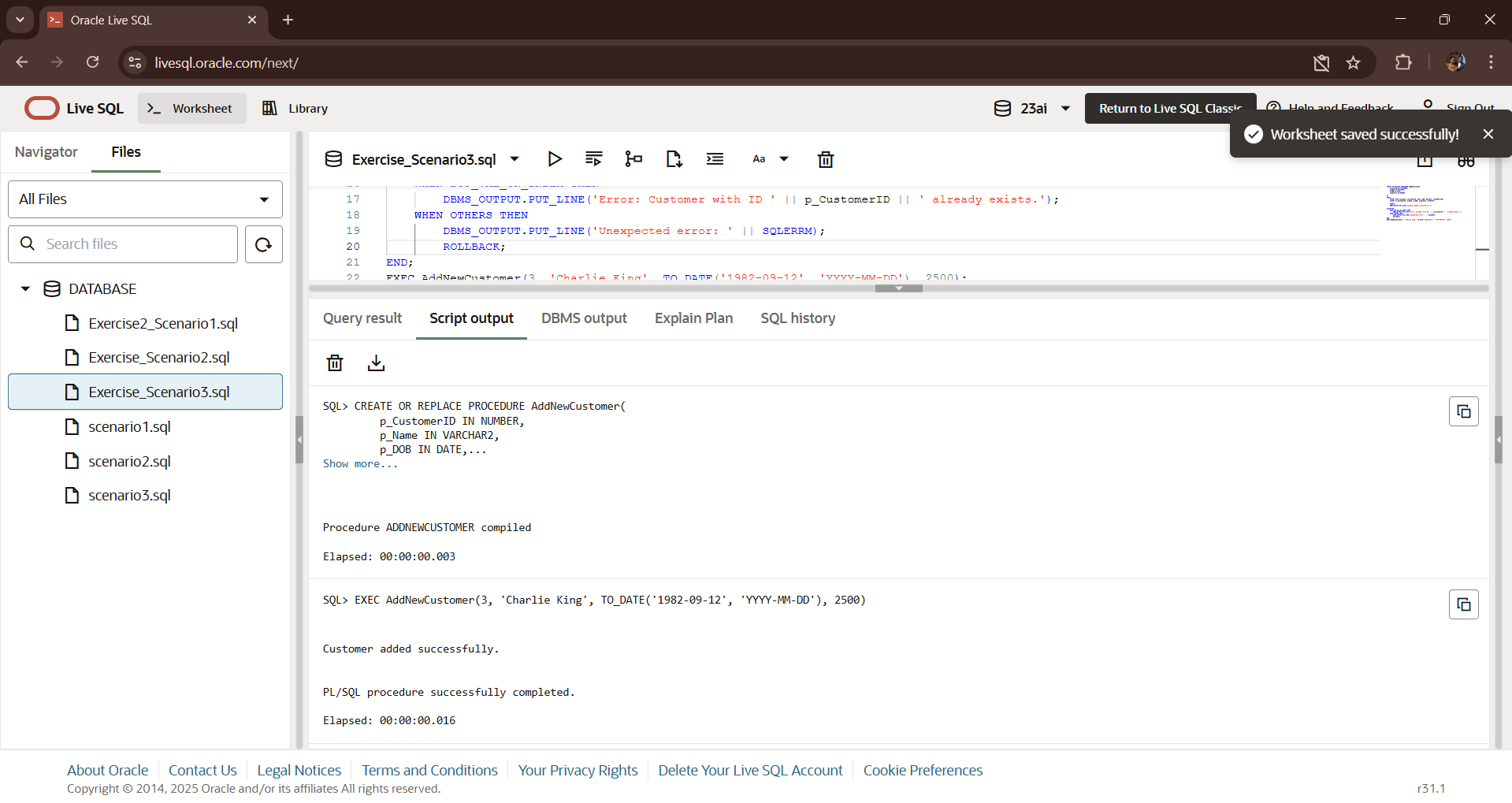
    WHEN OTHERS THEN

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

        ROLLBACK;

END;

EXEC AddNewCustomer(3, 'Charlie King', TO\_DATE('1982-09-12', 'YYYY-MM-DD'), 2500);

**OUTPUT:-**