**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.

**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.

**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.

**Scenario 1: ProcessMonthlyInterest — Apply 1% to All Savings Accounts**

**Requirements:**

* Apply 1% interest to **Savings** accounts only.
* Update the balance in the Accounts table.

**CODE:-**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

IS

BEGIN

  FOR acc\_rec IN (

    SELECT AccountID, Balance

    FROM Accounts

    WHERE AccountType = 'Savings'

  ) LOOP

    UPDATE Accounts

    SET Balance = acc\_rec.Balance + (acc\_rec.Balance \* 0.01),

        LastModified = SYSDATE

    WHERE AccountID = acc\_rec.AccountID;

  END LOOP;

  COMMIT;

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

EXCEPTION

  WHEN OTHERS THEN

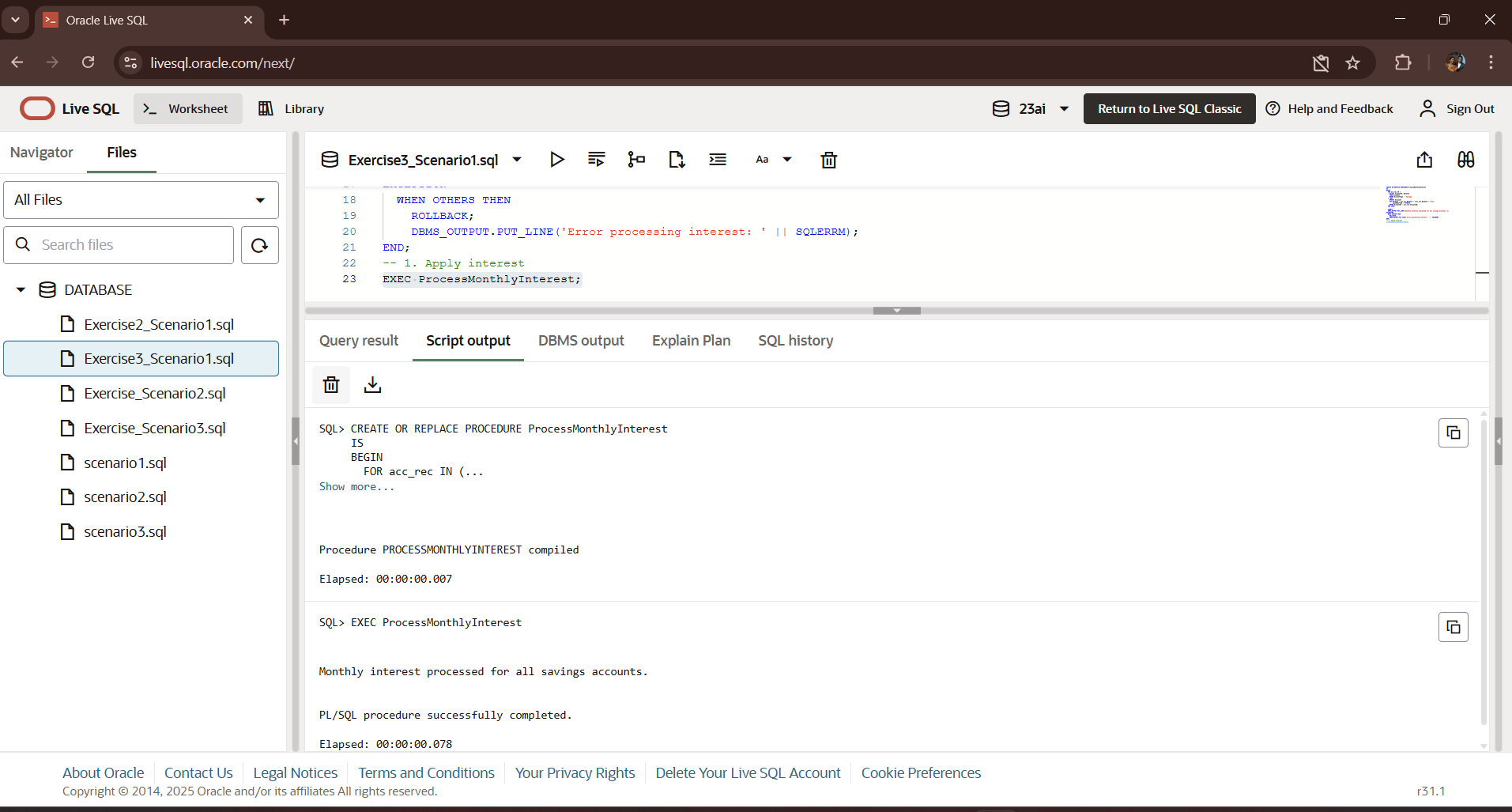
    ROLLBACK;

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

END;

-- 1. Apply interest

EXEC ProcessMonthlyInterest;

**OUTPUT:-**

**Scenario 2: UpdateEmployeeBonus — Bonus Based on Department**

**Requirements:**

* Take department name and bonus percentage as parameters.
* Apply bonus to all employees in that department.

**CODE:-**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

    p\_Department IN VARCHAR2,

    p\_BonusPercent IN NUMBER

)

IS

BEGIN

  UPDATE Employees

  SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

  WHERE Department = p\_Department;

  IF SQL%ROWCOUNT = 0 THEN

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

  ELSE

    DBMS\_OUTPUT.PUT\_LINE('Bonus applied to employees in department: ' || p\_Department);

  END IF;

  COMMIT;

EXCEPTION

  WHEN OTHERS THEN

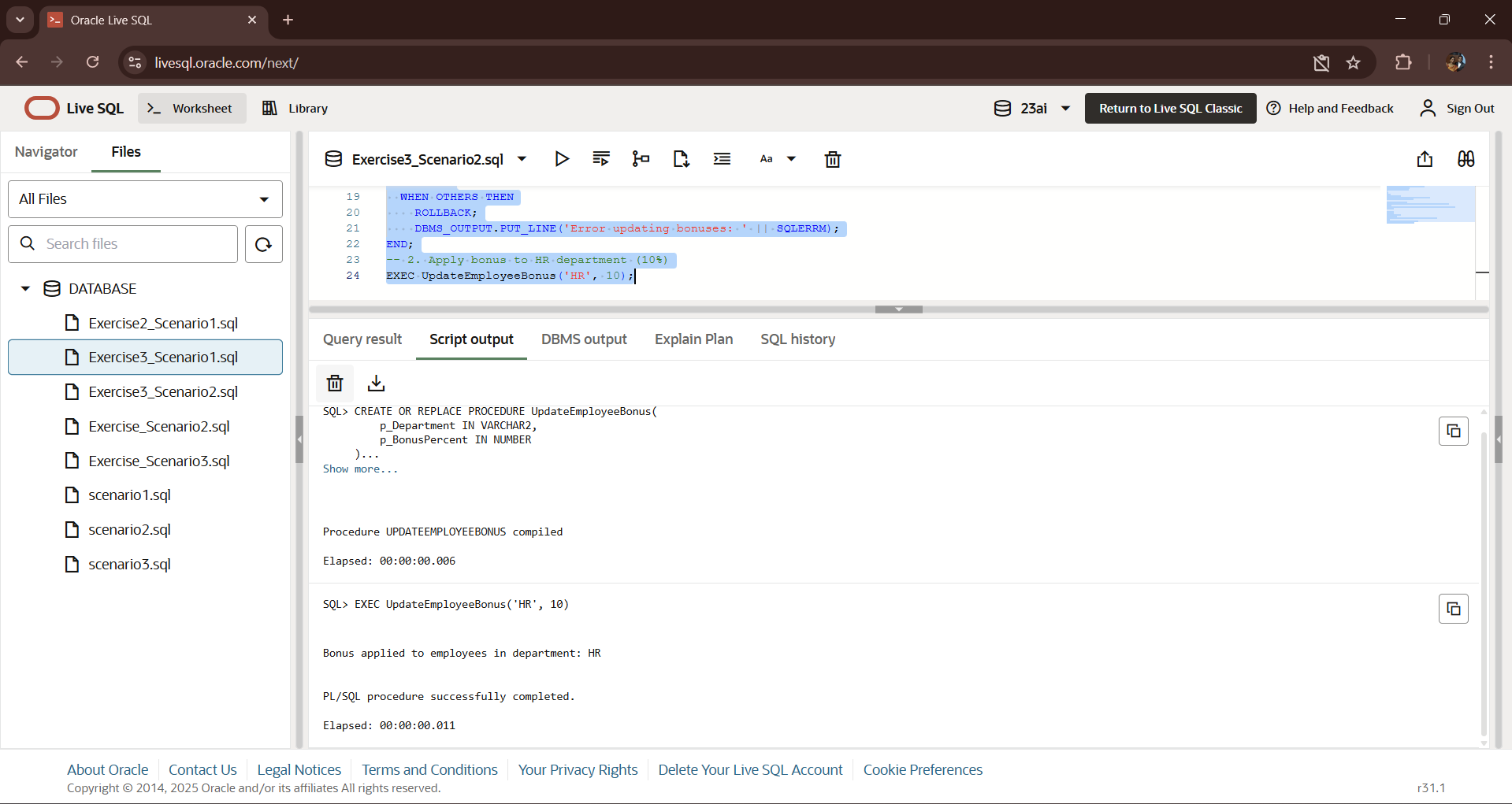
    ROLLBACK;

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

END;

-- 2. Apply bonus to HR department (10%)

EXEC UpdateEmployeeBonus('HR', 10);

**OUTPUT:-**

**Scenario 3: TransferFunds — Transfer Between Accounts with Validation**

Requirements:

* Transfer a given amount between accounts.
* Check if source account has enough balance.

**CODE:-**

CREATE OR REPLACE PROCEDURE TransferFunds(

    fromAccId IN NUMBER,

    toAccId IN NUMBER,

    amount IN NUMBER

)

IS

    fromBalance NUMBER;

BEGIN

  -- Lock and check source account balance

  SELECT Balance INTO fromBalance FROM Accounts

  WHERE AccountID = fromAccId FOR UPDATE;

  IF fromBalance < amount THEN

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

  END IF;

  -- Deduct from source

  UPDATE Accounts

  SET Balance = Balance - amount,

      LastModified = SYSDATE

  WHERE AccountID = fromAccId;

  -- Add to target

  UPDATE Accounts

  SET Balance = Balance + amount,

      LastModified = SYSDATE

  WHERE AccountID = toAccId;

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('Funds transferred successfully.');

EXCEPTION

  WHEN OTHERS THEN

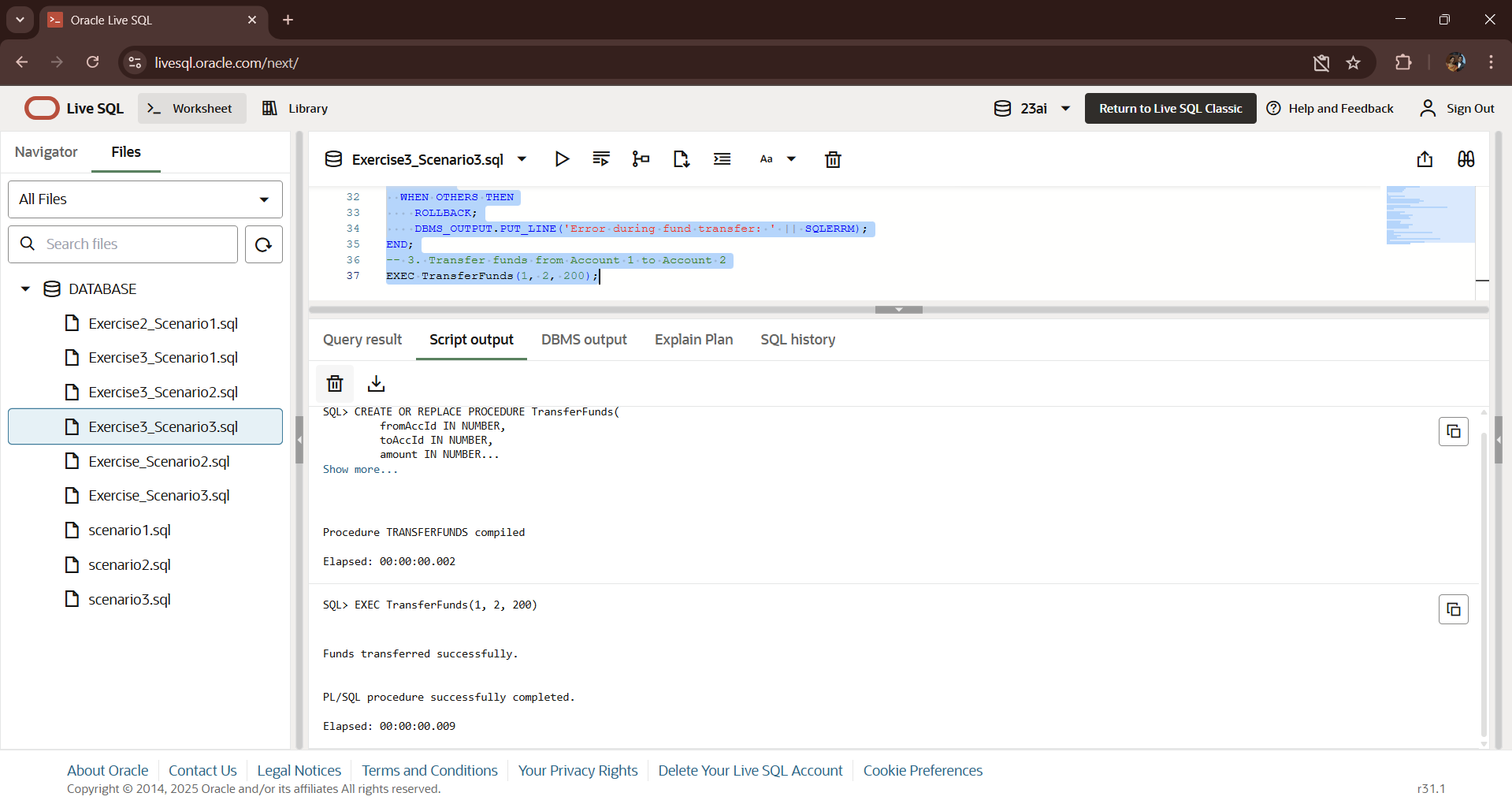
    ROLLBACK;

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

END;

-- 3. Transfer funds from Account 1 to Account 2

EXEC TransferFunds(1, 2, 200);

**OUTPUT:-**