**PL/SQL EXERCISES**

**Exercise 1: Control Structures**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

InterestRate NUMBER

);

INSERT INTO Customers VALUES (1, 'Ravi', 65, 7.5);

INSERT INTO Customers VALUES (2, 'Priya', 45, 8.0);

INSERT INTO Customers VALUES (3, 'John', 70, 6.5);

INSERT INTO Customers VALUES (4, 'Anu', 58, 9.0);

COMMIT;

**SCENARIO 1 :**

DECLARE

    CURSOR cur\_customers IS

        SELECT CustomerID, InterestRate, Age FROM Custome  WHERE Age > 60;

BEGIN

    FOR rec IN cur\_customers LOOP

        UPDATE Customers

        SET InterestRate = InterestRate - (InterestRate \* 0.01)

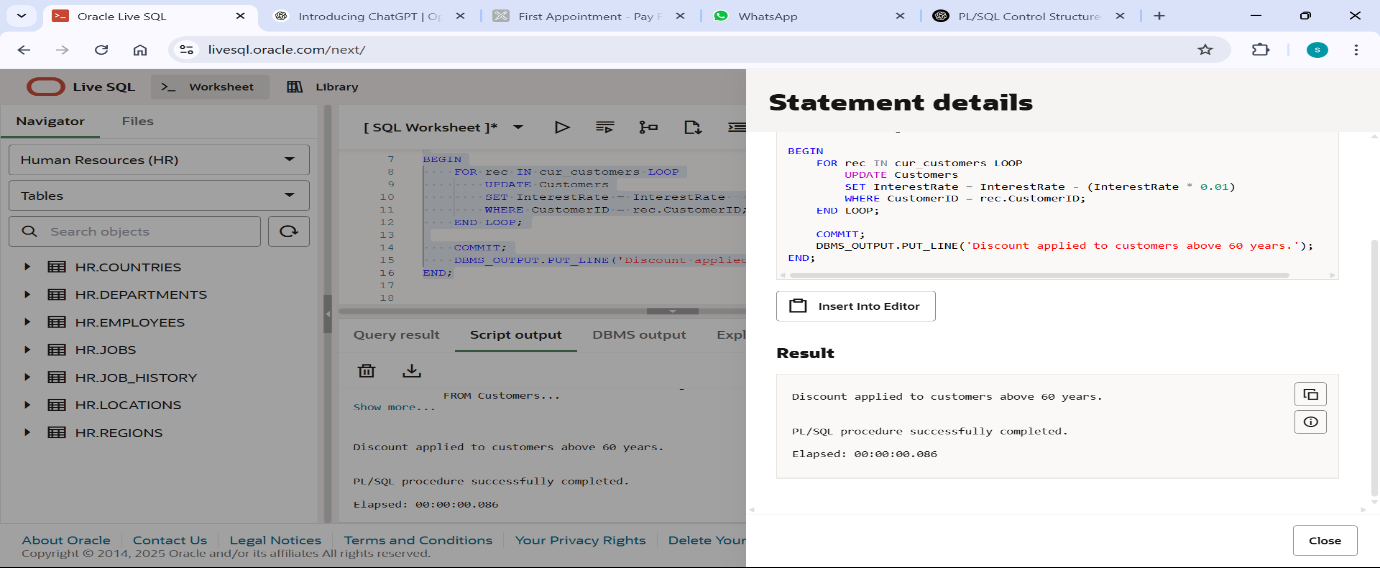
        WHERE CustomerID = rec.CustomerID;

    END LOOP;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Discount applied to customers above 60 years.');

END;

**OUTPUT :** 

**SCENARIO 2 :**

DECLARE

    CURSOR cur\_vip IS

        SELECT CustomerID

        FROM Customers

        WHERE Balance > 10000;

BEGIN

    FOR rec IN cur\_vip LOOP

        UPDATE Customers

        SET IsVIP = 'TRUE'

        WHERE CustomerID = rec.CustomerID;

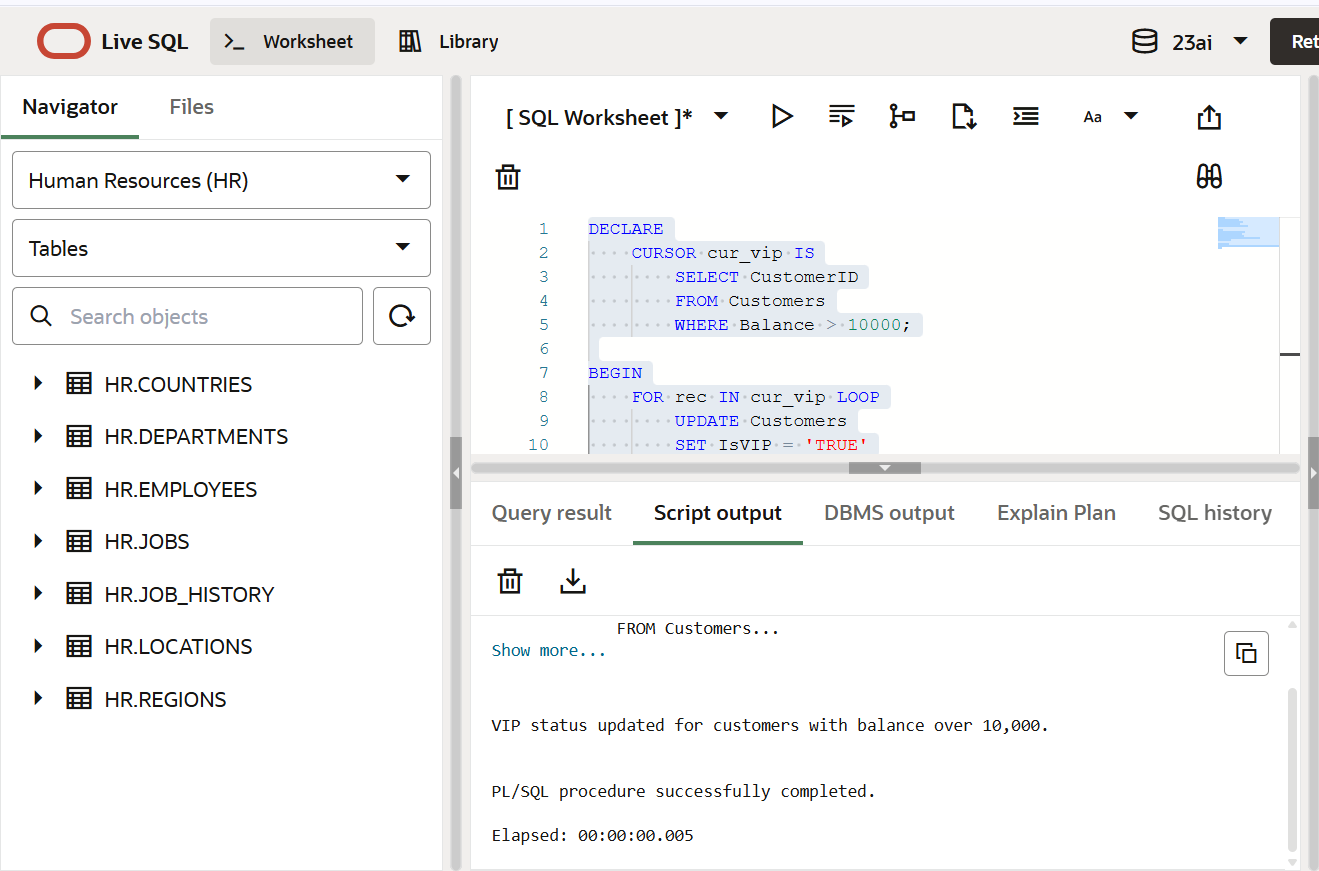
    END LOOP;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('VIP status updated for customers with balance over 10,000.');

END;

**OUTPUT :**

****

**SCENARIO 3 :**

DECLARE

    CURSOR cur\_vip IS

        SELECT CustomerID

        FROM Customers

        WHERE Balance > 10000;

BEGIN

    FOR rec IN cur\_vip LOOP

        UPDATE Customers

        SET IsVIP = 'TRUE'

        WHERE CustomerID = rec.CustomerID;

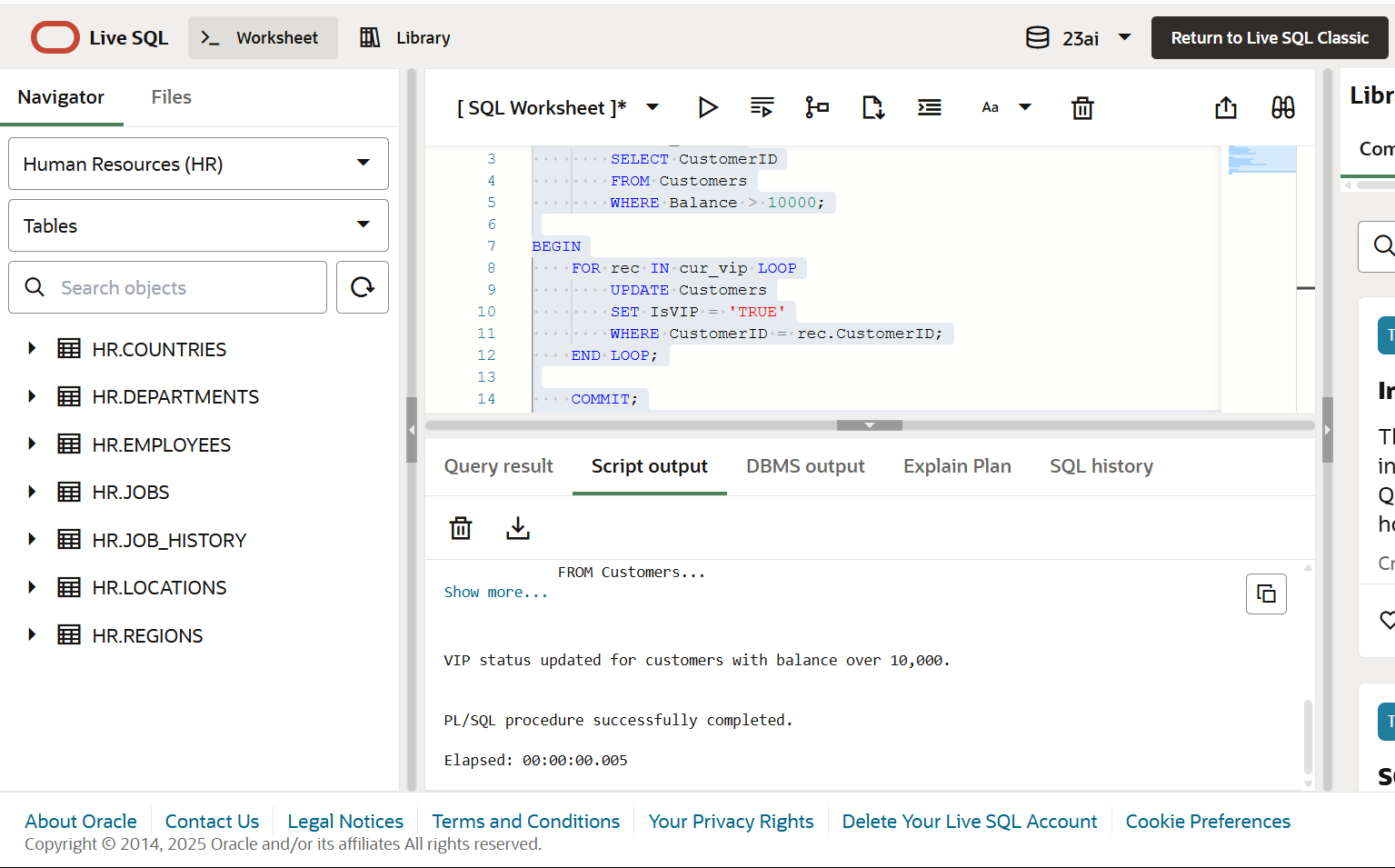
    END LOOP;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('VIP status updated for customers with balance over 10,000.');

END;

**OUTPUT :**

****

**Exercise 2: Stored Procedures**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

AccountType VARCHAR2(20),

Balance NUMBER

);

INSERT INTO Accounts VALUES (101, 'Savings', 1000);

INSERT INTO Accounts VALUES (102, 'Current', 2000);

INSERT INTO Accounts VALUES (103, 'Savings', 3000);

COMMIT;

**SCENARIO 1 :**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE Accounts

    SET Balance = Balance + (Balance \* 0.01)

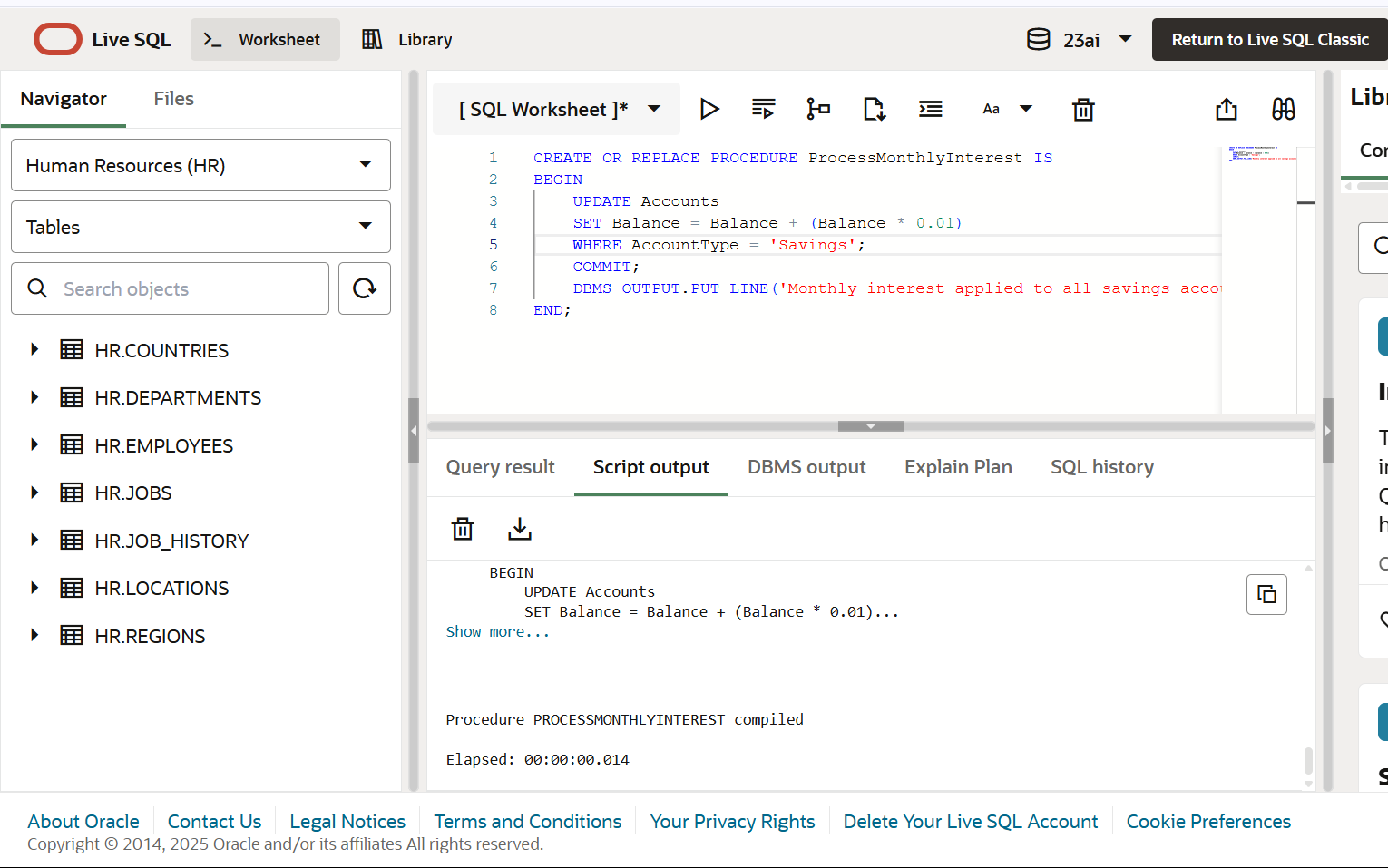
    WHERE AccountType = 'Savings';

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

**OUTPUT :**

****

**SCENARIO 2 :**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

DepartmentID NUMBER,

Salary NUMBER

);

INSERT INTO Employees VALUES (1, 'John', 10, 50000);

INSERT INTO Employees VALUES (2, 'Sara', 10, 52000);

INSERT INTO Employees VALUES (3, 'Mike', 20, 48000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

    dept\_id IN NUMBER,

    bonus\_pct IN NUMBER  -- e.g., 0.10 = 10%

) IS

BEGIN

    UPDATE Employees

    SET Salary = Salary + (Salary \* bonus\_pct)

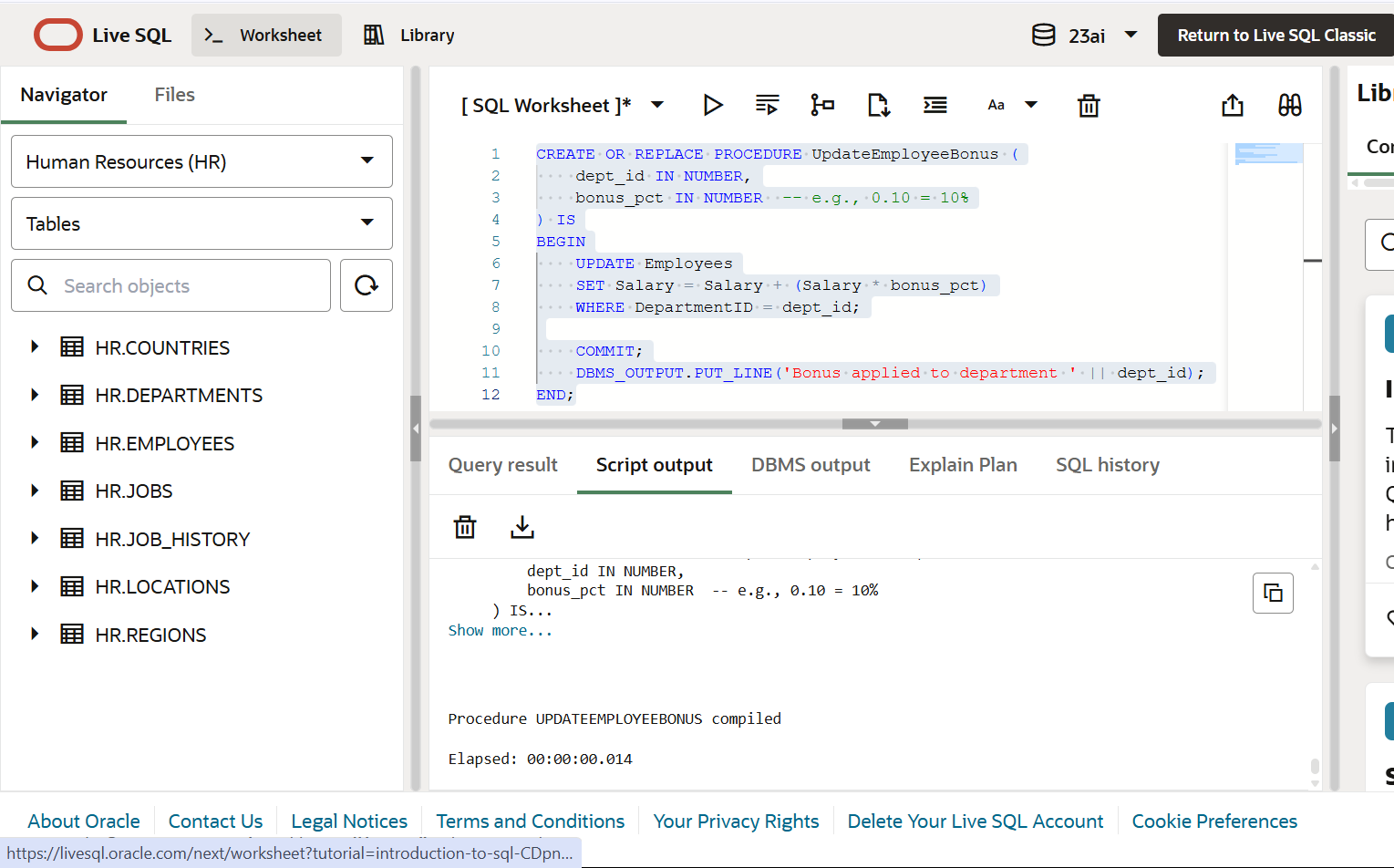
    WHERE DepartmentID = dept\_id;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department ' || dept\_id);

END;

**OUTPUT :**



**SCENARIO 3 :**

CREATE OR REPLACE PROCEDURE TransferFunds (

    from\_acc IN NUMBER,

    to\_acc IN NUMBER,

    amt IN NUMBER

) IS

    insufficient\_balance EXCEPTION;

    v\_balance NUMBER;

BEGIN

    -- Check balance of source account

    SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = from\_acc FOR UPDATE;

    IF v\_balance < amt THEN

        RAISE insufficient\_balance;

    END IF;

    UPDATE Accounts

    SET Balance = Balance - amt

    WHERE AccountID = from\_acc;

    UPDATE Accounts

    SET Balance = Balance + amt

    WHERE AccountID = to\_acc;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amt || ' from Account ' || from\_acc || ' to Account ' || to\_acc);

EXCEPTION

    WHEN insufficient\_balance THEN

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: insufficient balance in Account ' || from\_acc);

        ROLLBACK;

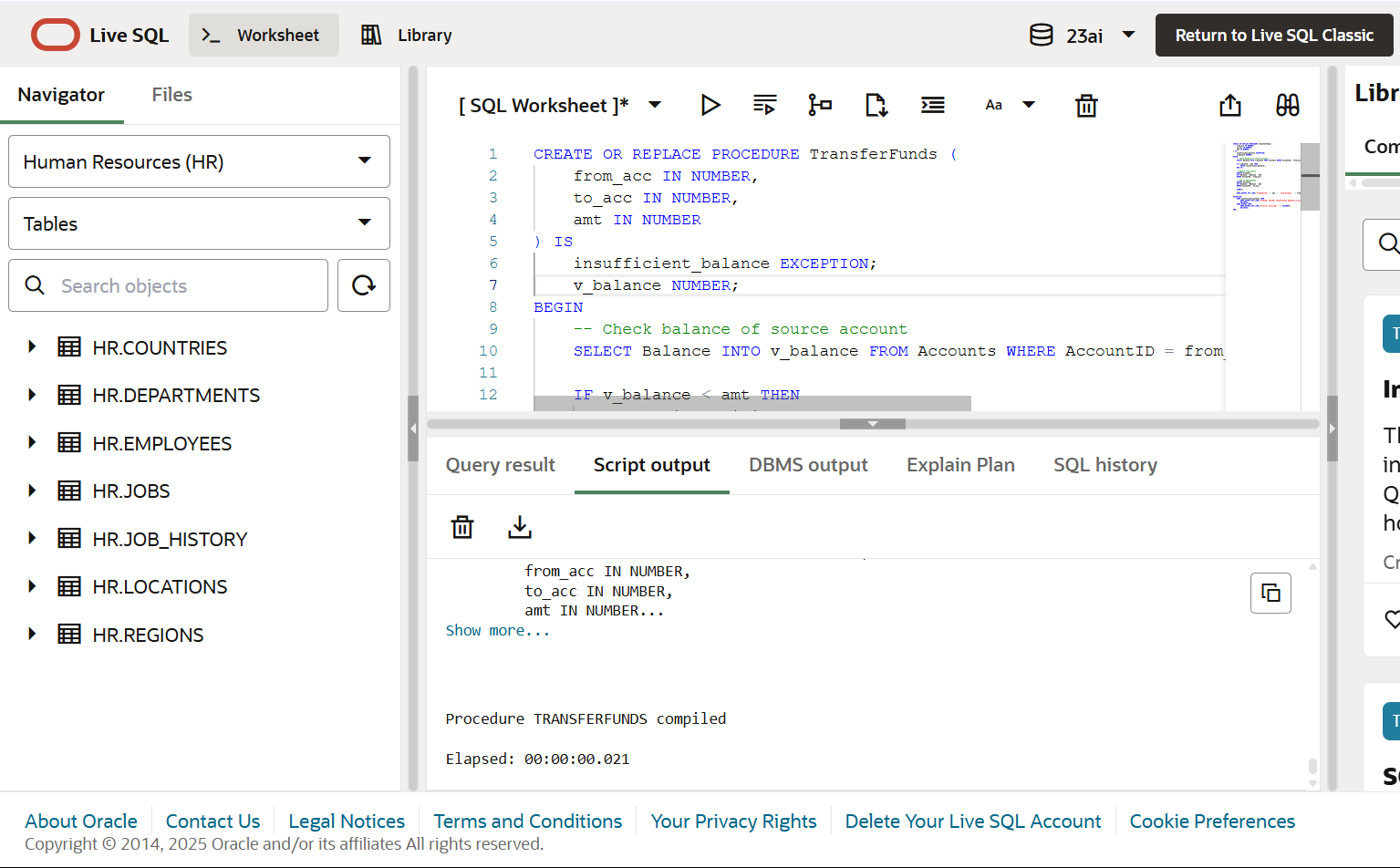
    WHEN OTHERS THEN

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

        ROLLBACK;

END;

**OUTPUT :**

****