**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.

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

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

VALUES (1, 'Amit Verma', TO\_DATE('1990-01-01', 'YYYY-MM-DD'), 0, SYSDATE);

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

VALUES (2, 'Riya Sen', TO\_DATE('1992-01-01', 'YYYY-MM-DD'), 0, SYSDATE);

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

VALUES (3, 'Suresh Reddy', TO\_DATE('1985-05-01', 'YYYY-MM-DD'), 0, SYSDATE);

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

VALUES (4, 'Pooja Patel', TO\_DATE('1988-08-08', 'YYYY-MM-DD'), 0, SYSDATE);

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

VALUES (99, 1, 'Savings', 5000, SYSDATE);

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

VALUES (100, 2, 'Savings', 2000, SYSDATE);

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

VALUES (101, 3, 'Savings', 200, SYSDATE);

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

VALUES (102, 4, 'Savings', 1500, SYSDATE);

COMMIT;

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

p\_FromAccountID NUMBER,

p\_ToAccountID NUMBER,

p\_Amount NUMBER

)

IS

v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance < p\_Amount THEN

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

END IF;

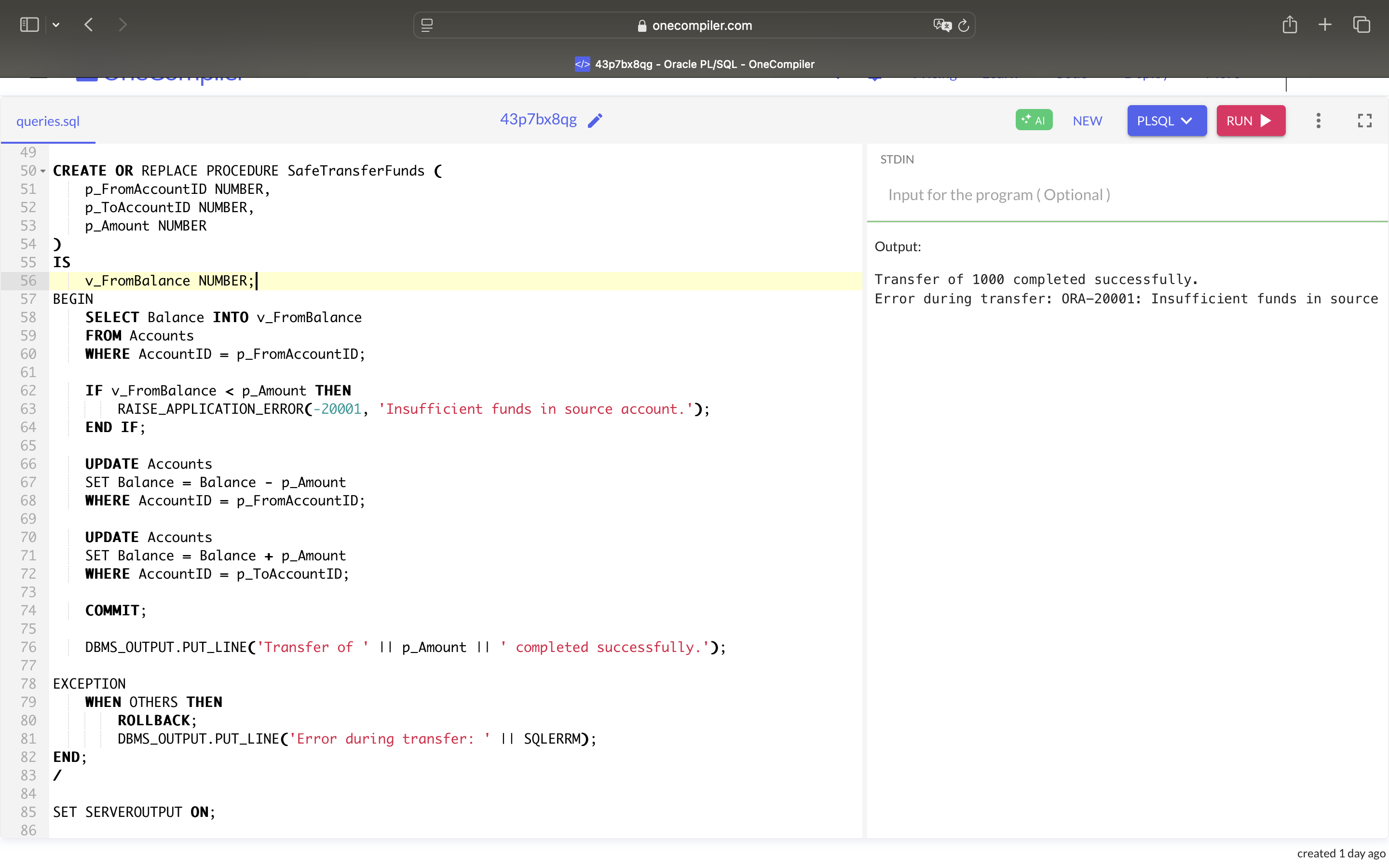
UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || p\_Amount || ' completed successfully.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

SET SERVEROUTPUT ON;

-- Successful Transfer

EXEC SafeTransferFunds(99, 100, 1000);

-- Insufficient Funds (Should give error message)

EXEC SafeTransferFunds(101, 102, 500);

**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.

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Rahul Mehra', 'Manager', 80000, 'Operations', TO\_DATE('2015-06-10', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Sneha Kapoor', 'Developer', 60000, 'IT', TO\_DATE('2018-02-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (3, 'Amit Joshi', 'Analyst', 50000, 'Finance', TO\_DATE('2020-01-01', 'YYYY-MM-DD'));

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateSalary (

p\_EmployeeID NUMBER,

p\_PercentIncrease NUMBER

)

IS

BEGIN

-- Attempt to update salary

UPDATE Employees

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

WHERE EmployeeID = p\_EmployeeID;

-- Check if any row was affected

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID does not exist.');

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salary updated successfully for Employee ID: ' || p\_EmployeeID);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

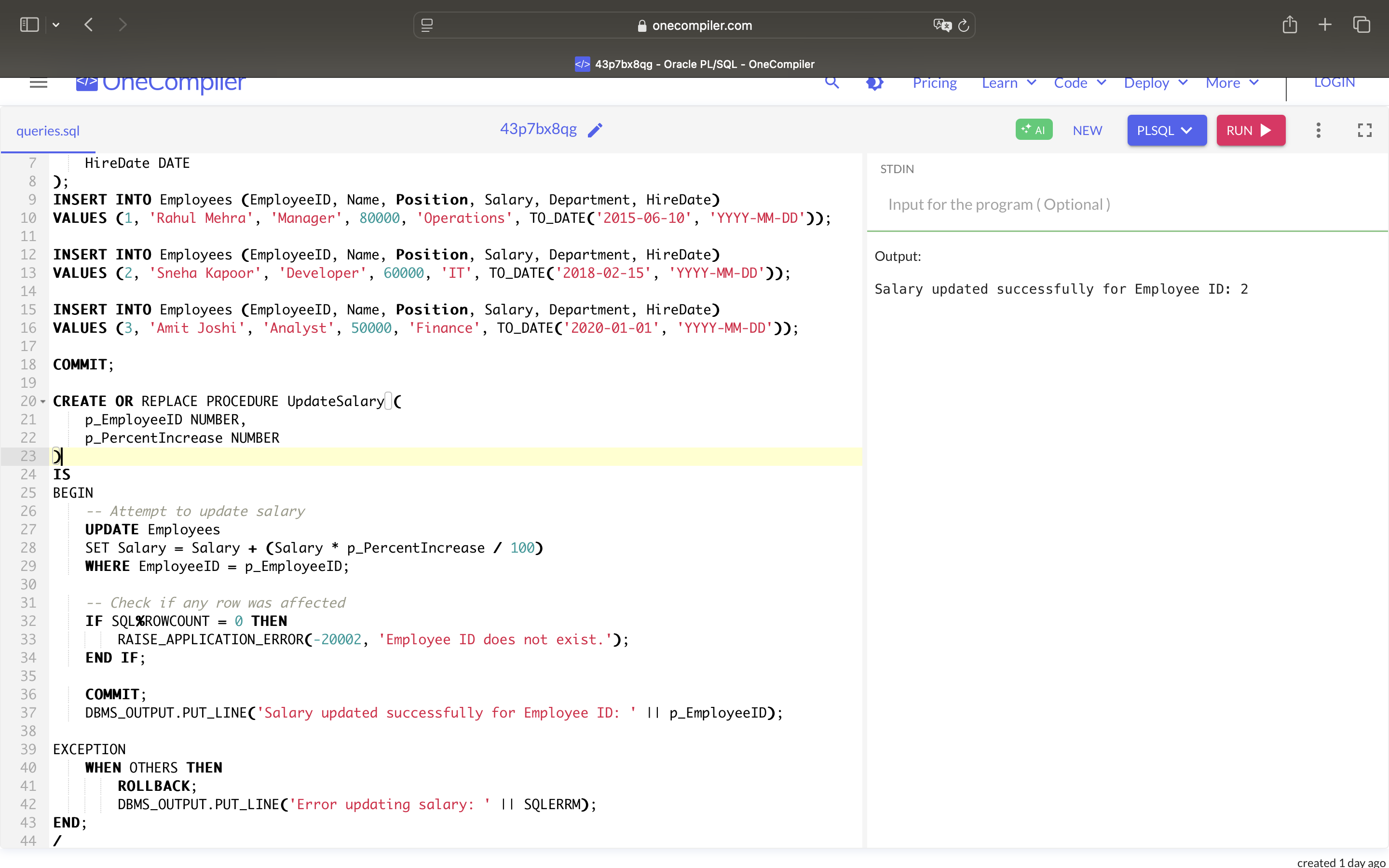
SET SERVEROUTPUT ON;

BEGIN

UpdateSalary(2, 10);

END;

/



**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

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

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

VALUES (1, 'Anjali Sharma', TO\_DATE('1995-03-10', 'YYYY-MM-DD'), 5000, SYSDATE);

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

VALUES (2, 'Rohit Mehra', TO\_DATE('1990-07-22', 'YYYY-MM-DD'), 10000, SYSDATE);

COMMIT;

CREATE OR REPLACE PROCEDURE AddNewCustomer (

p\_CustomerID NUMBER,

p\_Name VARCHAR2,

p\_DOB DATE,

p\_Balance NUMBER

)

IS

v\_Count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_Count

FROM Customers

WHERE CustomerID = p\_CustomerID;

IF v\_Count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Customer with this ID already exists.');

END IF;

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

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

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('New customer added successfully. Customer ID: ' || p\_CustomerID);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error adding customer: ' || SQLERRM);

END;

/

SET SERVEROUTPUT ON;

EXEC AddNewCustomer(3, 'Neha Gupta', TO\_DATE('1998-11-15', 'YYYY-MM-DD'), 7500);

