**Candidate Name**:- SHARANYA K

**Superset ID**:-6419887

**WEEK – 1 HANDS ON EXERCISE (JAVA FSE DEEPSKILLING)**

**(PL/SQL Programming)**

**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

-- Customers table

CREATE TABLE Customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER,

isvip VARCHAR2(5)

);

-- Loans table

CREATE TABLE Loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

due\_date DATE,

interest\_rate NUMBER,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

-- Inserting sample customers

INSERT INTO Customers VALUES (1, 'Rahul', 65, 12000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Priya', 45, 8000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Amit', 70, 10500, 'FALSE');

INSERT INTO Customers VALUES (4, 'Sneha', 30, 20000, 'FALSE');

-- Inserting sample loans

INSERT INTO Loans VALUES (101, 1, SYSDATE + 10, 8.5);

INSERT INTO Loans VALUES (102, 2, SYSDATE + 35, 9.0);

INSERT INTO Loans VALUES (103, 3, SYSDATE + 5, 7.8);

INSERT INTO Loans VALUES (104, 4, SYSDATE + 25, 8.0);

COMMIT;

DECLARE

-- Cursor to select loan info for customers aged above 60

CURSOR customer\_with\_loans IS

SELECT l.loan\_id, l.interest\_rate

FROM Loans l, Customers c

WHERE c.customer\_id = l.customer\_id

AND c.age > 60;

temp\_loan\_id Loans.loan\_id%TYPE;

original\_rate Loans.interest\_rate%TYPE;

BEGIN

OPEN customer\_with\_loans;

LOOP

FETCH customer\_with\_loans INTO temp\_loan\_id, original\_rate;

EXIT WHEN customer\_with\_loans%NOTFOUND;

UPDATE Loans

SET interest\_rate = original\_rate - 1

WHERE loan\_id = temp\_loan\_id;

END LOOP;

CLOSE customer\_with\_loans;

DBMS\_OUTPUT.PUT\_LINE('Interest rates updated for customers above age 60.');

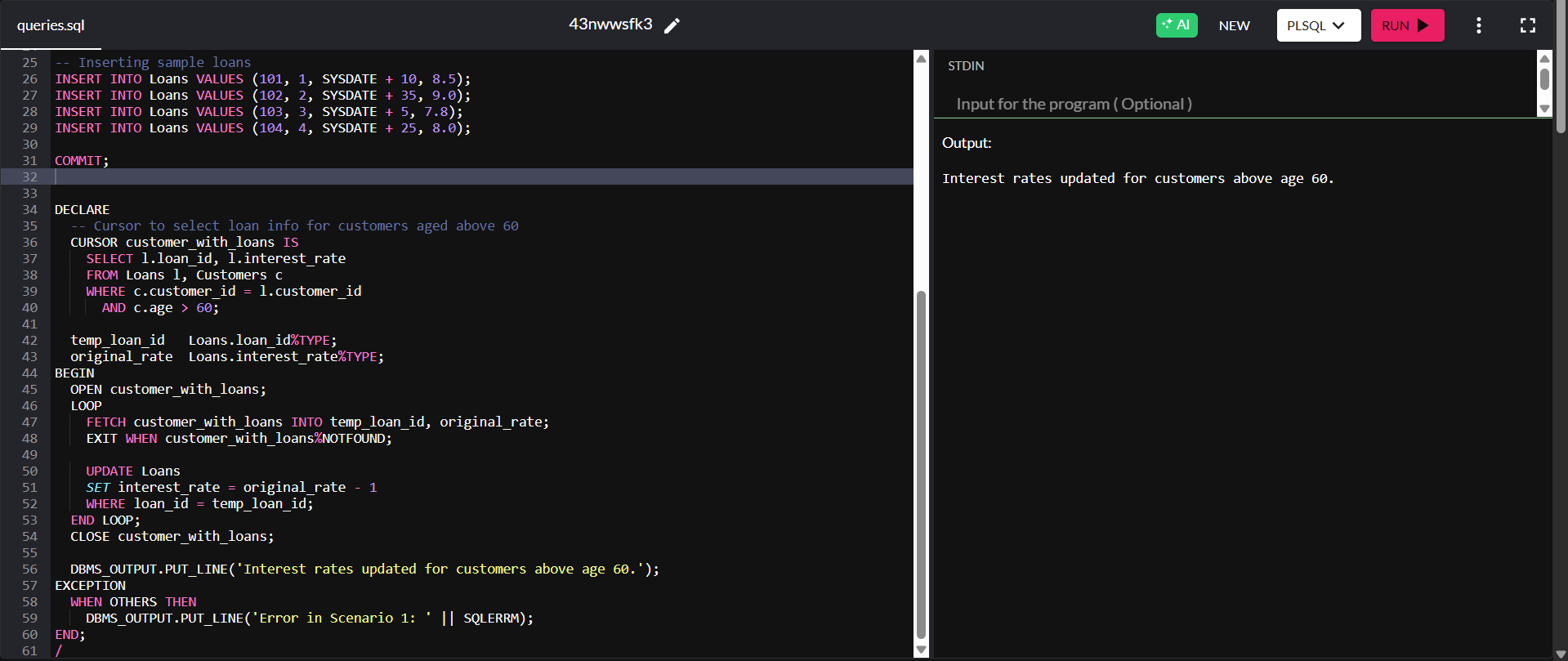
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error in Scenario 1: ' || SQLERRM);

END;

/



**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**The table creation and inserting value is same as mentioned above in scenario1 after that this code should be executed.**

DECLARE

CURSOR vip\_eligible\_customers IS

SELECT customer\_id

FROM Customers

WHERE balance > 10000;

vip\_id Customers.customer\_id%TYPE;

BEGIN

OPEN vip\_eligible\_customers;

LOOP

FETCH vip\_eligible\_customers INTO vip\_id;

EXIT WHEN vip\_eligible\_customers%NOTFOUND;

UPDATE Customers

SET isvip = 'TRUE'

WHERE customer\_id = vip\_id;

END LOOP;

CLOSE vip\_eligible\_customers;

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

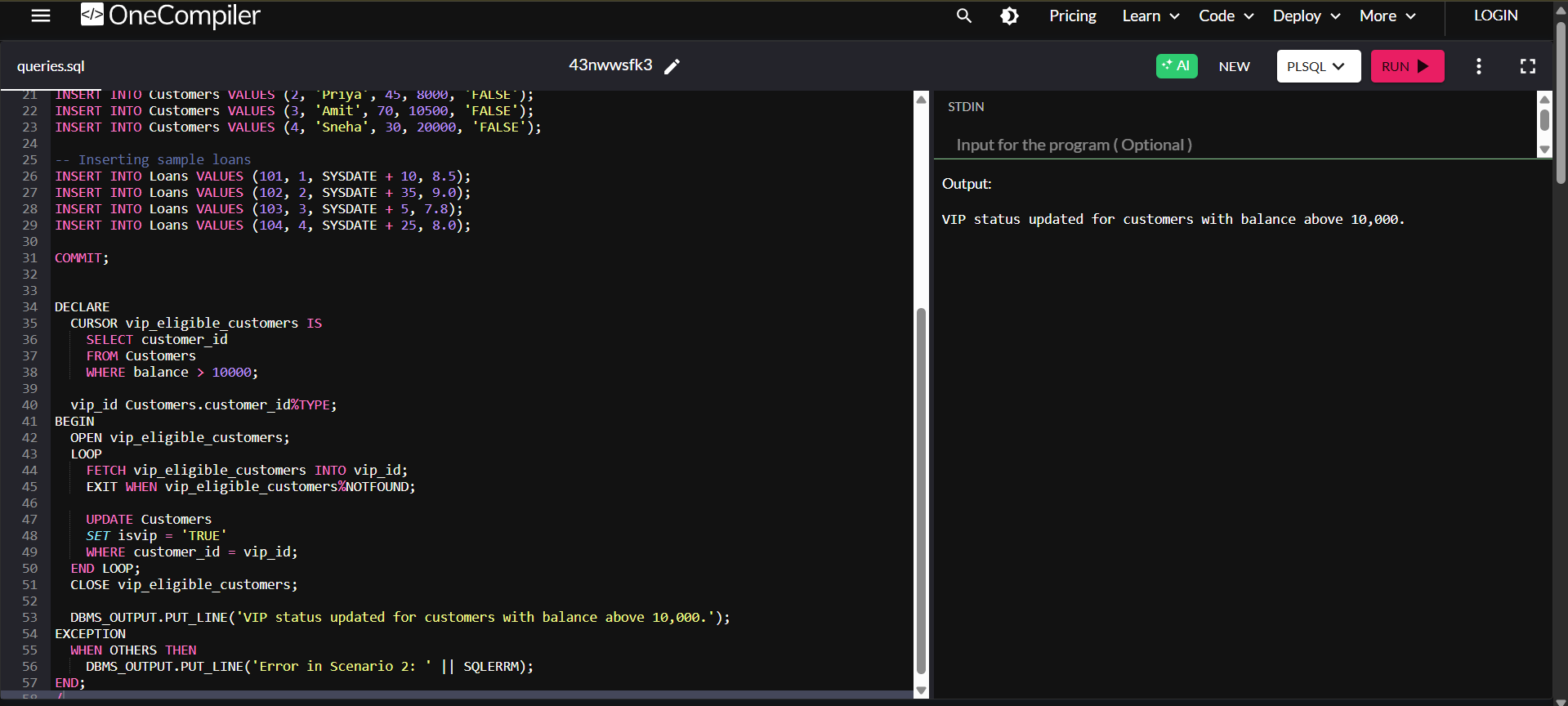
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error in Scenario 2: ' || SQLERRM);

END;

/



**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**The table creation and inserting value is same as mentioned above in scenario1 after that this code should be executed.**

DECLARE

CURSOR due\_soon\_loans IS

SELECT c.name, l.due\_date

FROM Customers c, Loans l

WHERE c.customer\_id = l.customer\_id

AND l.due\_date BETWEEN SYSDATE AND SYSDATE + 30;

cust\_name Customers.name%TYPE;

due\_date\_value Loans.due\_date%TYPE;

BEGIN

OPEN due\_soon\_loans;

LOOP

FETCH due\_soon\_loans INTO cust\_name, due\_date\_value;

EXIT WHEN due\_soon\_loans%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || cust\_name || ', your loan is due on ' || TO\_CHAR(due\_date\_value, 'DD-MON-YYYY'));

END LOOP;

CLOSE due\_soon\_loans;

DBMS\_OUTPUT.PUT\_LINE('Loan reminders printed for upcoming due dates.');

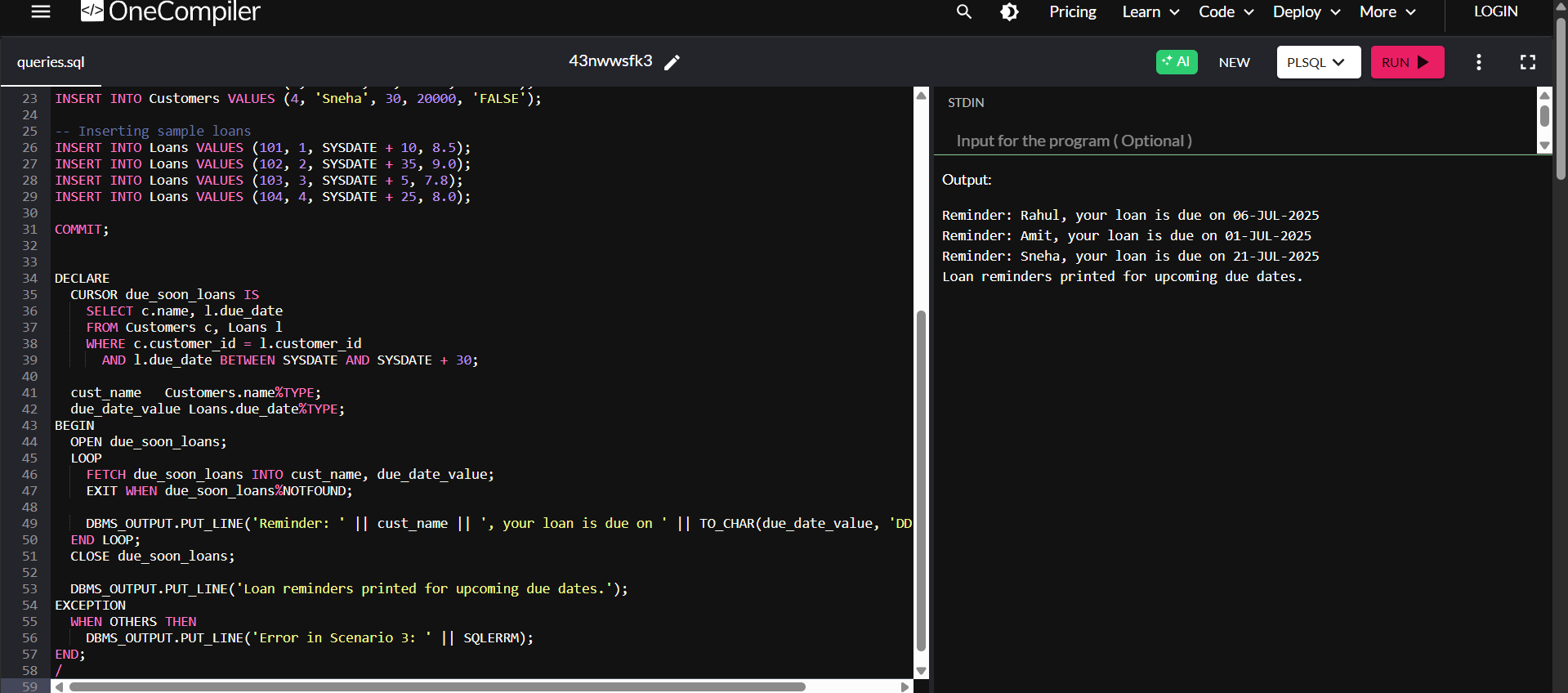
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error in Scenario 3: ' || SQLERRM);

END;

/



**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.

-- Enable output

SET SERVEROUTPUT ON;

-- Create table if not already created

CREATE TABLE Accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

account\_type VARCHAR2(20),

balance NUMBER

);

-- Insert sample data

INSERT INTO Accounts VALUES (101, 1, 'Savings', 10000);

INSERT INTO Accounts VALUES (102, 2, 'Current', 15000);

INSERT INTO Accounts VALUES (103, 3, 'Savings', 5000);

COMMIT;

-- Create the procedure

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

CURSOR savings\_cursor IS

SELECT account\_id, balance

FROM Accounts

WHERE account\_type = 'Savings';

v\_account\_id Accounts.account\_id%TYPE;

v\_balance Accounts.balance%TYPE;

v\_interest NUMBER;

BEGIN

OPEN savings\_cursor;

LOOP

FETCH savings\_cursor INTO v\_account\_id, v\_balance;

EXIT WHEN savings\_cursor%NOTFOUND;

v\_interest := v\_balance \* 0.01;

UPDATE Accounts

SET balance = balance + v\_interest

WHERE account\_id = v\_account\_id;

END LOOP;

CLOSE savings\_cursor;

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

EXCEPTION

WHEN OTHERS THEN

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

END;

/

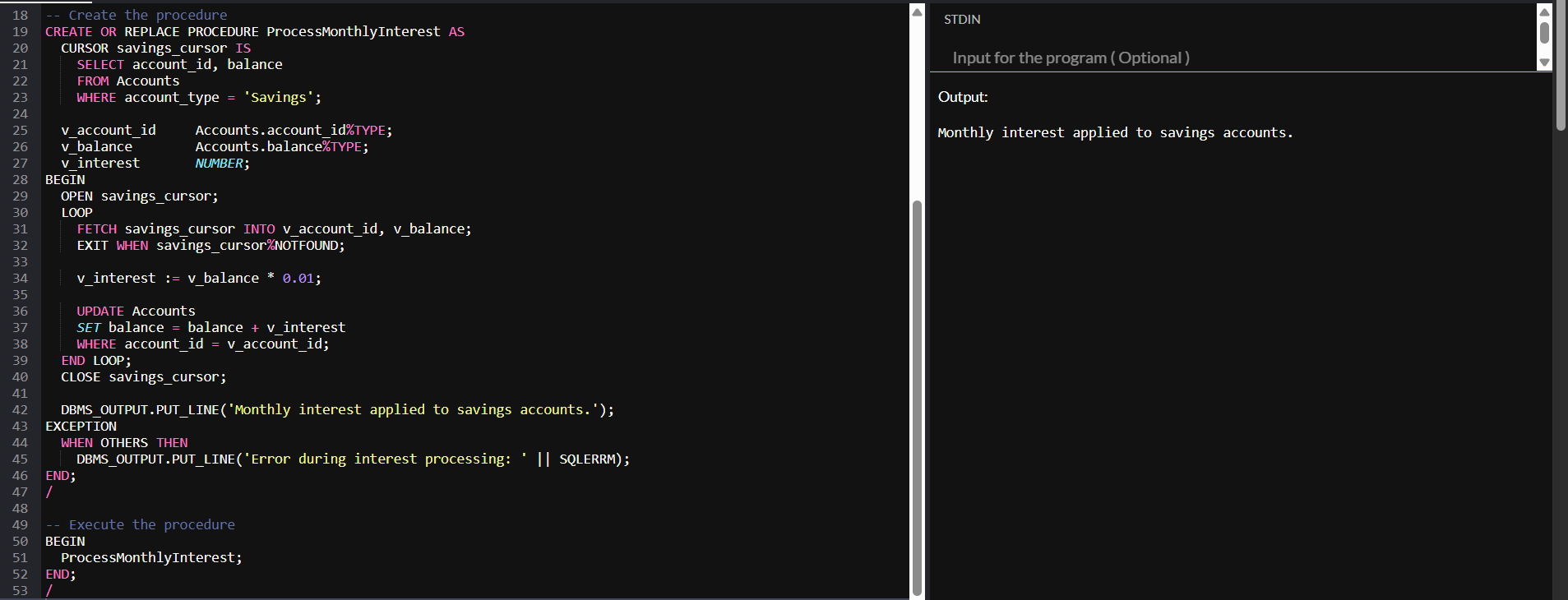
-- Execute the procedure

BEGIN

ProcessMonthlyInterest;

END;

/



**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.

-- Enable output

SET SERVEROUTPUT ON;

-- Create Employees table (if not exists)

CREATE TABLE Employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER

);

-- Sample data

INSERT INTO Employees VALUES (1, 'Ravi', 'Sales', 40000);

INSERT INTO Employees VALUES (2, 'Anita', 'Sales', 45000);

INSERT INTO Employees VALUES (3, 'John', 'HR', 30000);

COMMIT;

-- Procedure to update salary with bonus

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_pct IN NUMBER

) AS

BEGIN

UPDATE Employees

SET salary = salary + (salary \* bonus\_pct / 100)

WHERE department = dept\_name;

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

EXCEPTION

WHEN OTHERS THEN

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

END;

/

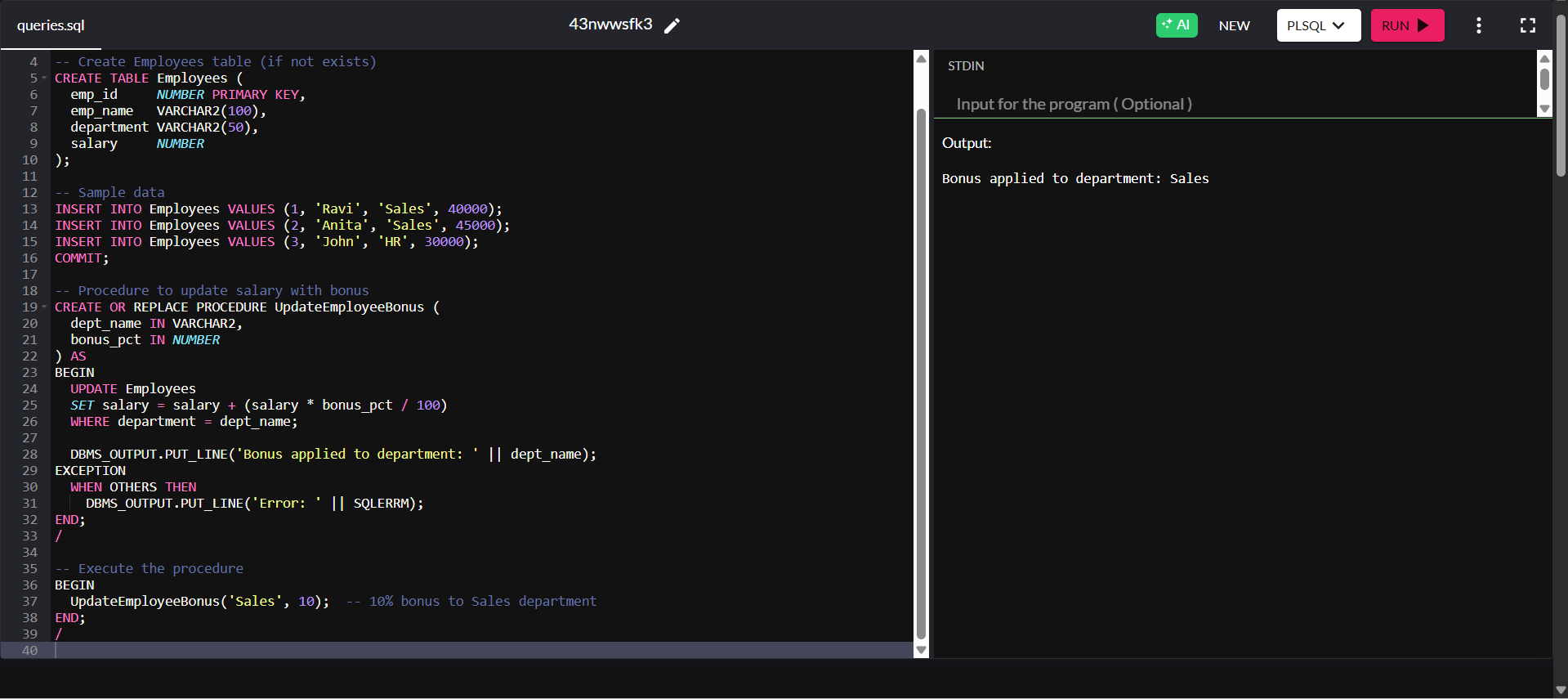
-- Execute the procedure

BEGIN

UpdateEmployeeBonus('Sales', 10); -- 10% bonus to Sales department

END;

/



**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.

-- Enable output

SET SERVEROUTPUT ON;

-- Create Accounts table (if not exists)

CREATE TABLE Accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

account\_type VARCHAR2(20),

balance NUMBER

);

-- Sample data

INSERT INTO Accounts VALUES (201, 10, 'Savings', 12000);

INSERT INTO Accounts VALUES (202, 11, 'Savings', 6000);

COMMIT;

-- Procedure to transfer funds

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amt IN NUMBER

) AS

current\_balance Accounts.balance%TYPE;

BEGIN

SELECT balance INTO current\_balance

FROM Accounts

WHERE account\_id = from\_acc

FOR UPDATE;

IF current\_balance < amt THEN

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

END IF;

UPDATE Accounts

SET balance = balance - amt

WHERE account\_id = from\_acc;

UPDATE Accounts

SET balance = balance + amt

WHERE account\_id = to\_acc;

DBMS\_OUTPUT.PUT\_LINE('Transfer completed from ' || from\_acc || ' to ' || to\_acc);

EXCEPTION

WHEN OTHERS THEN

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

END;

/

-- Execute the procedure

BEGIN

TransferFunds(201, 202, 3000); -- Transfer ₹3000 from account 201 to 202

END;

/

