**Week2** (6397530 – Nayudu Praveen Kumar)

**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.

**CODE:**

CREATE TABLE customers (

customer\_id NUMBER,

name VARCHAR2(50),

age NUMBER,

loan\_interest\_rate NUMBER

);

INSERT INTO customers VALUES (1, 'Ravi', 65, 10.5);

INSERT INTO customers VALUES (2, 'Sita', 45, 11.0);

INSERT INTO customers VALUES (3, 'Mohan', 70, 9.5);

INSERT INTO customers VALUES (4, 'Geeta', 59, 10.0);

COMMIT;

DECLARE

CURSOR cust\_cursor IS

SELECT customer\_id, age, loan\_interest\_rate

FROM customers;

BEGIN

FOR cust\_rec IN cust\_cursor LOOP

IF cust\_rec.age > 60 THEN

-- Apply 1% discount

UPDATE customers

SET loan\_interest\_rate = loan\_interest\_rate - 1

WHERE customer\_id = cust\_rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('1% discount applied for Customer ID: ' || cust\_rec.customer\_id);

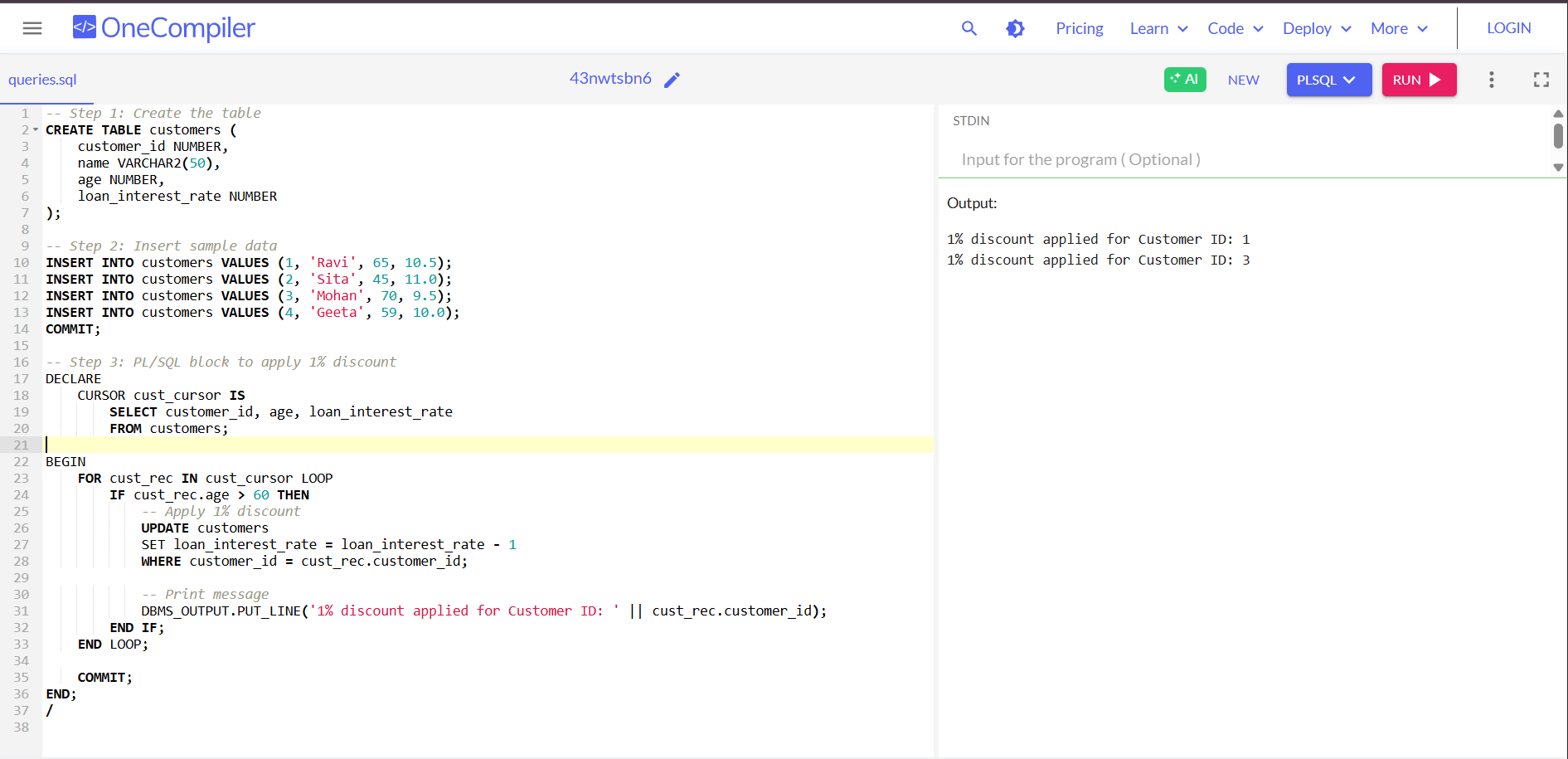
END IF;

END LOOP;

COMMIT;

END;/

**OUTPUT:**



**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 Is VIP to TRUE for those with a balance over $10,000.

**CODE:**

CREATE TABLE customers (

customer\_id NUMBER,

name VARCHAR2(50),

balance NUMBER,

isvip VARCHAR2(5) -- Using VARCHAR2 to represent TRUE/FALSE

);

INSERT INTO customers VALUES (1, 'Arjun', 5000, 'FALSE');

INSERT INTO customers VALUES (2, 'Lakshmi', 12000, 'FALSE');

INSERT INTO customers VALUES (3, 'Rohit', 15000, 'FALSE');

INSERT INTO customers VALUES (4, 'Anu', 8000, 'FALSE');

COMMIT;

DECLARE

CURSOR cust\_cursor IS

SELECT customer\_id, balance

FROM customers;

BEGIN

FOR cust\_rec IN cust\_cursor LOOP

IF cust\_rec.balance > 10000 THEN

UPDATE customers

SET isvip = 'TRUE'

WHERE customer\_id = cust\_rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated for Customer ID: ' || cust\_rec.customer\_id);

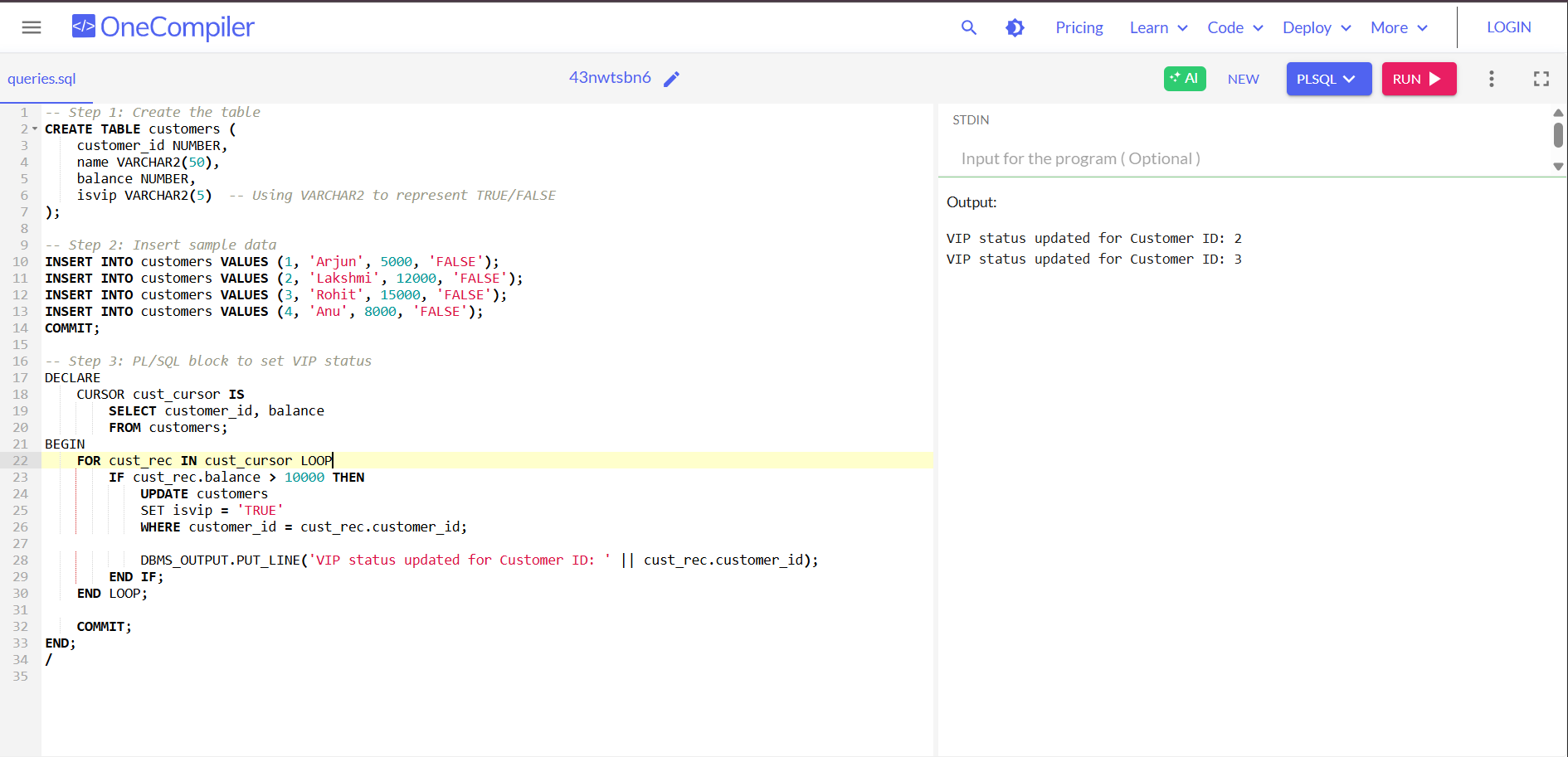
END IF;

END LOOP;

COMMIT;

END;/

**OUTPUT:**



**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.

**CODE:**

CREATE TABLE loans (

loan\_id NUMBER,

customer\_name VARCHAR2(50),

due\_date DATE

);

INSERT INTO loans VALUES (101, 'Ravi', SYSDATE + 10);

INSERT INTO loans VALUES (102, 'Sita', SYSDATE + 40);

INSERT INTO loans VALUES (103, 'Mohan', SYSDATE + 25);

INSERT INTO loans VALUES (104, 'Geeta', SYSDATE - 5);

COMMIT;

DECLARE

CURSOR due\_soon\_cursor IS

SELECT loan\_id, customer\_name, due\_date

FROM loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR loan\_rec IN due\_soon\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Loan ID ' || loan\_rec.loan\_id ||

' for customer ' || loan\_rec.customer\_name ||

' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-MON-YYYY')

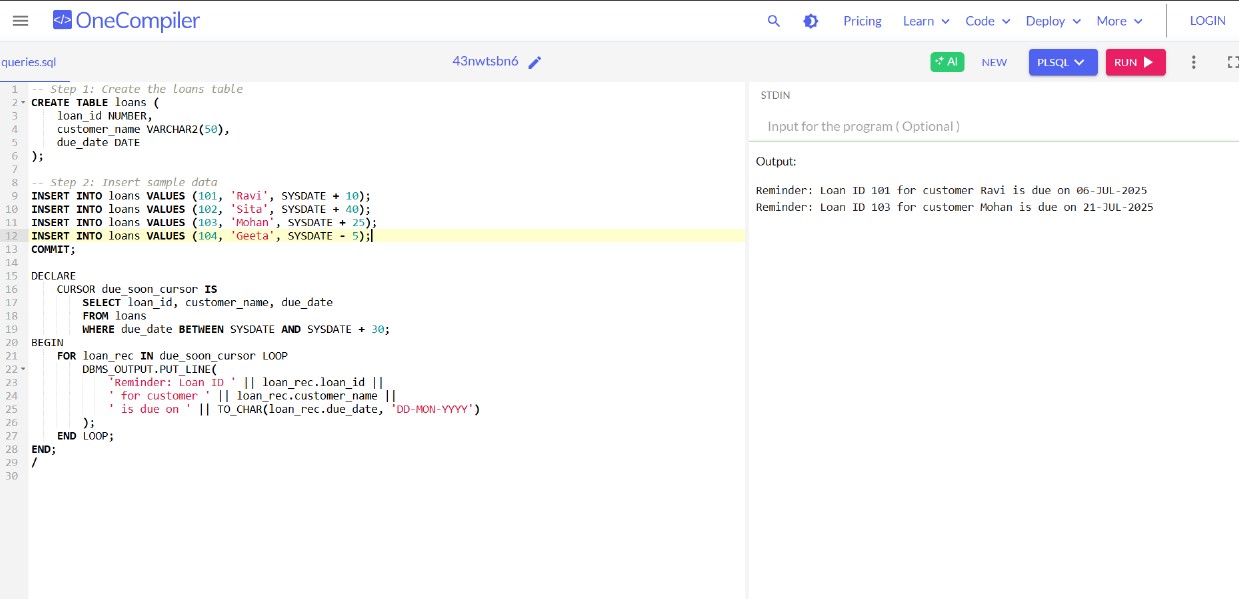
);

END LOOP;

END;

/

**OUTPUT:**



**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.

**CODE:**

CREATE TABLE accounts (

account\_id NUMBER,

account\_holder VARCHAR2(50),

account\_type VARCHAR2(20),

balance NUMBER

);

INSERT INTO accounts VALUES (1, 'Ravi', 'savings', 10000);

INSERT INTO accounts VALUES (2, 'Sita', 'current', 20000);

INSERT INTO accounts VALUES (3, 'Arjun', 'savings', 15000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

CURSOR savings\_cursor IS

SELECT account\_id, balance

FROM accounts

WHERE LOWER(account\_type) = 'savings';

BEGIN

FOR acc\_rec IN savings\_cursor LOOP

UPDATE accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_id = acc\_rec.account\_id;

DBMS\_OUTPUT.PUT\_LINE('Interest applied for Account ID: ' || acc\_rec.account\_id);

END LOOP;

COMMIT;

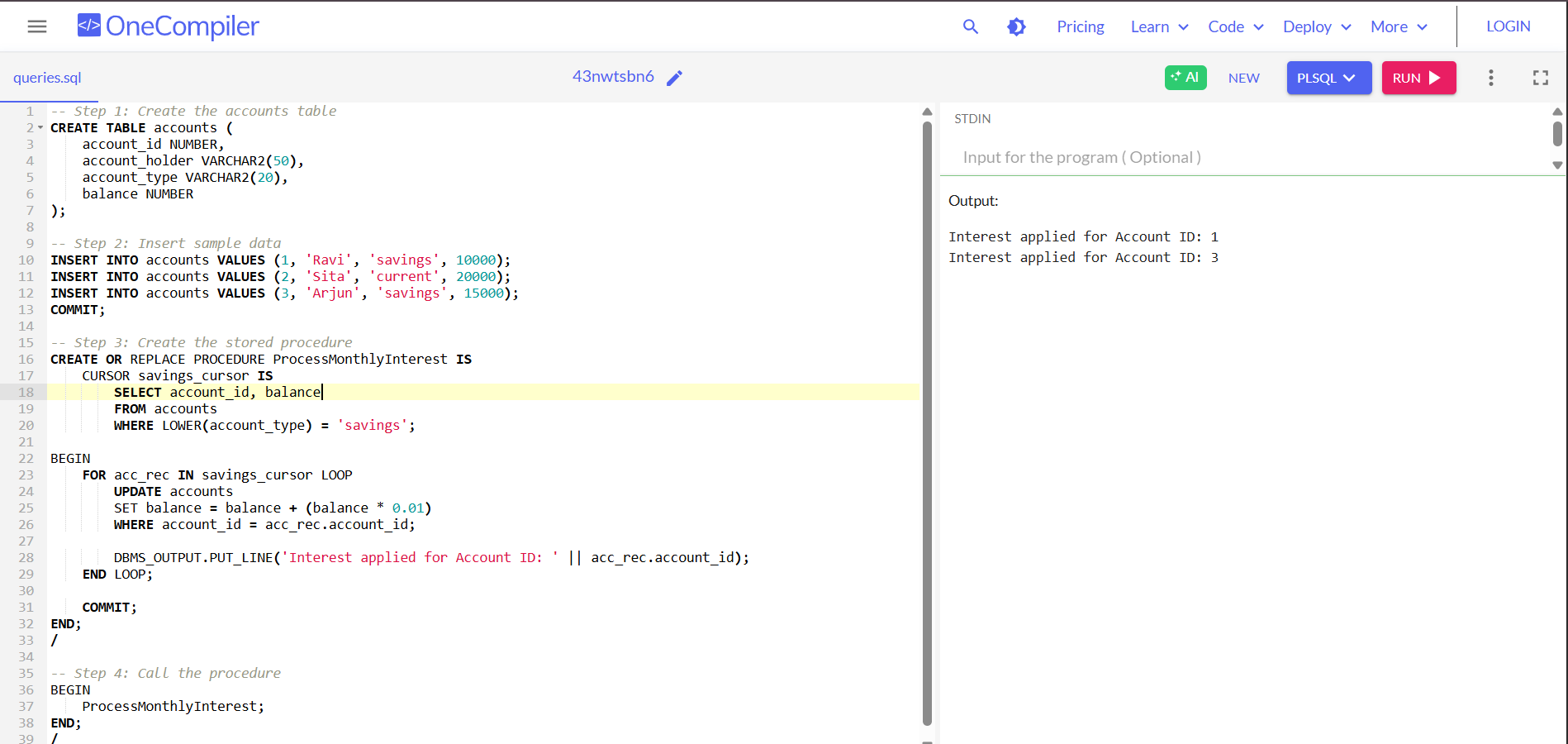
END;/

BEGIN

ProcessMonthlyInterest;

END;/

**OUTPUT:**



**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.

**CODE:**

CREATE TABLE employees (

emp\_id NUMBER,

emp\_name VARCHAR2(50),

department VARCHAR2(30),

salary NUMBER

);

INSERT INTO employees VALUES (1, 'Ravi', 'HR', 30000);

INSERT INTO employees VALUES (2, 'Sita', 'IT', 40000);

INSERT INTO employees VALUES (3, 'Mohan', 'IT', 50000);

INSERT INTO employees VALUES (4, 'Geeta', 'Finance', 45000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_pct IN NUMBER

) IS

CURSOR emp\_cursor IS

SELECT emp\_id, salary

FROM employees

WHERE LOWER(department) = LOWER(dept\_name);

BEGIN

FOR emp\_rec IN emp\_cursor LOOP

UPDATE employees

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

WHERE emp\_id = emp\_rec.emp\_id;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied for Employee ID: ' || emp\_rec.emp\_id);

END LOOP;

COMMIT;

END;

/

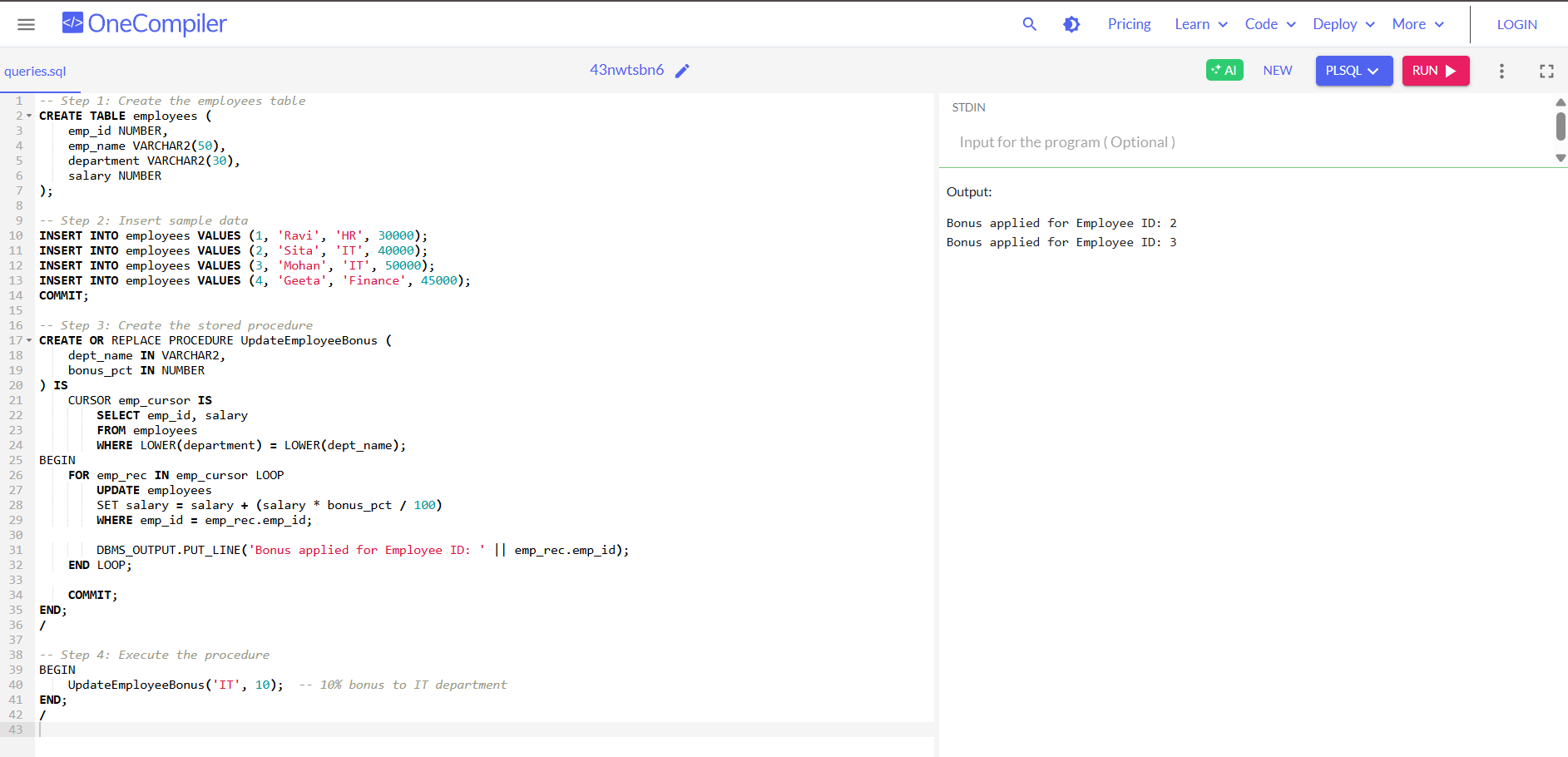
BEGIN

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

END;

/

**OUTPUT:**



**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.

**CODE:**

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

account\_holder VARCHAR2(50),

balance NUMBER

);

INSERT INTO accounts VALUES (1, 'Ravi', 10000);

INSERT INTO accounts VALUES (2, 'Sita', 8000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_acc\_id IN NUMBER,

to\_acc\_id IN NUMBER,

amount IN NUMBER

) IS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance

FROM accounts

WHERE account\_id = from\_acc\_id;

IF from\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in account ' || from\_acc\_id);

ELSE

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_acc\_id;

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_acc\_id;

DBMS\_OUTPUT.PUT\_LINE('Rs.' || amount || ' transferred from Account ' || from\_acc\_id || ' to Account ' || to\_acc\_id);

END IF;

COMMIT;

END;/

BEGIN

TransferFunds(1, 2, 3000); -- Transfer ₹3000 from account 1 to 2

END;/

**OUTPUT:**

