**Candidate Name :- Abhishek Gowda M**

**Superset ID :- 6431229**

**WEEK – 2 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.

**Code for the above question:-**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE loan\_customers';

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

/

CREATE TABLE loan\_customers (

customer\_id NUMBER,

interest\_rate NUMBER,

age NUMBER

);

INSERT INTO loan\_customers VALUES (101, 10.5, 69);

INSERT INTO loan\_customers VALUES (102, 12.0, 65);

INSERT INTO loan\_customers VALUES (103, 9.5, 57);

INSERT INTO loan\_customers VALUES (104, 11.5, 77);

COMMIT;

SET SERVEROUTPUT ON;

DECLARE

CURSOR customer\_cursor IS

SELECT customer\_id, interest\_rate, age FROM loan\_customers;

v\_new\_rate NUMBER;

BEGIN

FOR rec IN customer\_cursor LOOP

IF rec.age > 60 THEN

v\_new\_rate := rec.interest\_rate \* 0.99;

UPDATE loan\_customers

SET interest\_rate = v\_new\_rate

WHERE customer\_id = rec.customer\_id;

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

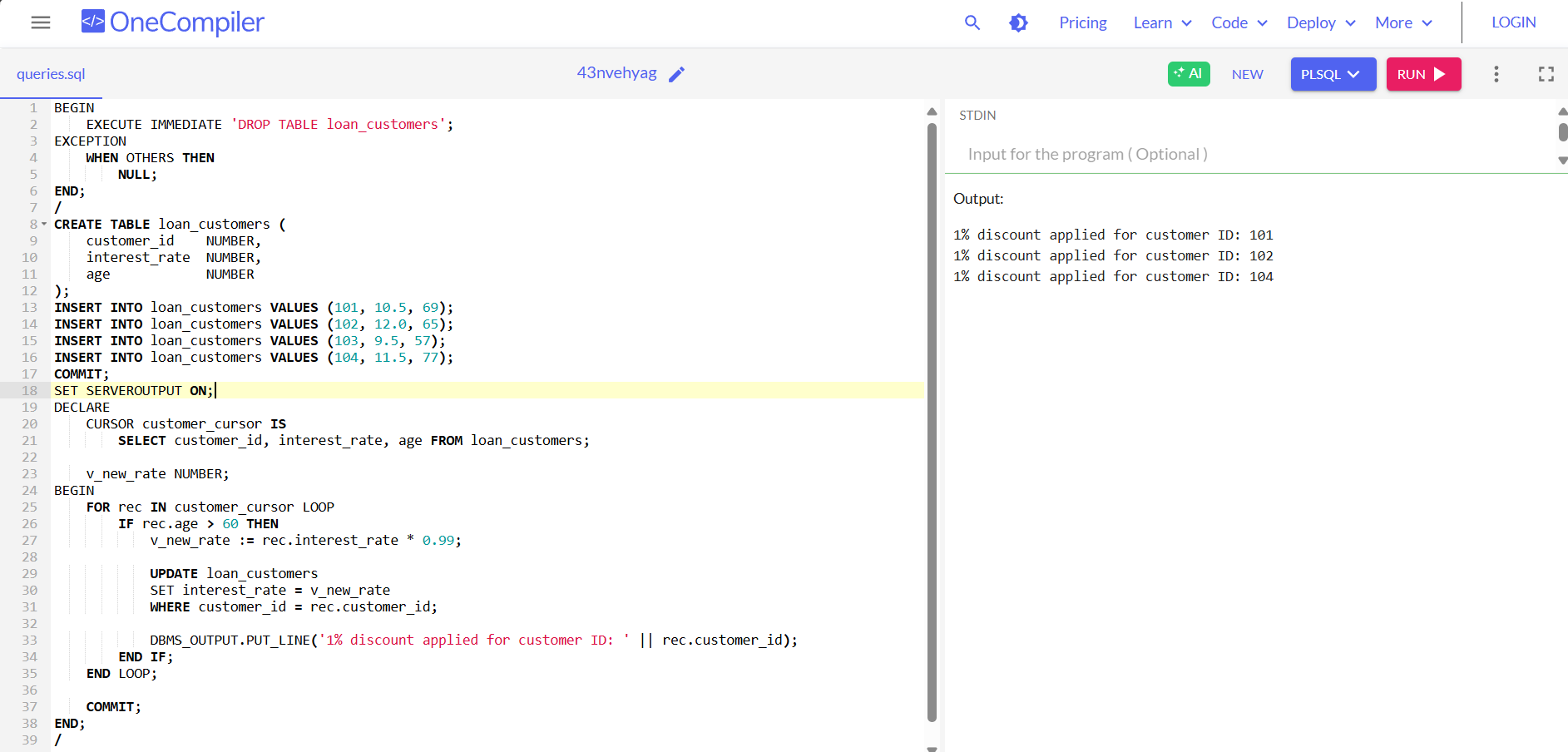
END IF;

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 IsVIP to TRUE for those with a balance over $10,000..

**Code for the above question:-**

CREATE TABLE bank\_customers (

customer\_id NUMBER,

customer\_name VARCHAR2(100),

balance NUMBER(10, 2),

is\_vip VARCHAR2(5) DEFAULT 'FALSE'

);

INSERT INTO bank\_customers VALUES (301, 'YASH', 92090.00, 'FALSE');

INSERT INTO bank\_customers VALUES (302, 'ABHI', 8000.00, 'FALSE');

INSERT INTO bank\_customers VALUES (303, 'MANJA', 78000.00, 'FALSE');

INSERT INTO bank\_customers VALUES (304, 'NEHA', 25000.00, 'FALSE');

INSERT INTO bank\_customers VALUES (305, 'RAGU', 8900.00, 'FALSE');

COMMIT;

SET SERVEROUTPUT ON;

DECLARE

CURSOR cust\_cursor IS

SELECT customer\_id, balance FROM bank\_customers;

BEGIN

FOR rec IN cust\_cursor LOOP

IF rec.balance > 10000 THEN

UPDATE bank\_customers

SET is\_vip = 'TRUE'

WHERE customer\_id = rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || rec.customer\_id || ' promoted to VIP.');

END IF;

END LOOP;

COMMIT;

END;

/

**OUTPUT:**A screenshot of a computer

AI-generated content may be incorrect.

**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 for the above question:-**

CREATE TABLE loan\_accounts (

loan\_id NUMBER,

customer\_name VARCHAR2(100),

due\_date DATE

);

INSERT INTO loan\_accounts VALUES (401, 'MANJA', SYSDATE + 10);

INSERT INTO loan\_accounts VALUES (402, 'NEHA', SYSDATE + 35);

INSERT INTO loan\_accounts VALUES (403, 'YASH', SYSDATE + 5);

INSERT INTO loan\_accounts VALUES (404, 'ASHA', SYSDATE - 1);

COMMIT;

SET SERVEROUTPUT ON;

DECLARE

CURSOR loan\_cursor IS

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

FROM loan\_accounts

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR rec IN loan\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || rec.loan\_id ||

' is due for ' || rec.customer\_name ||

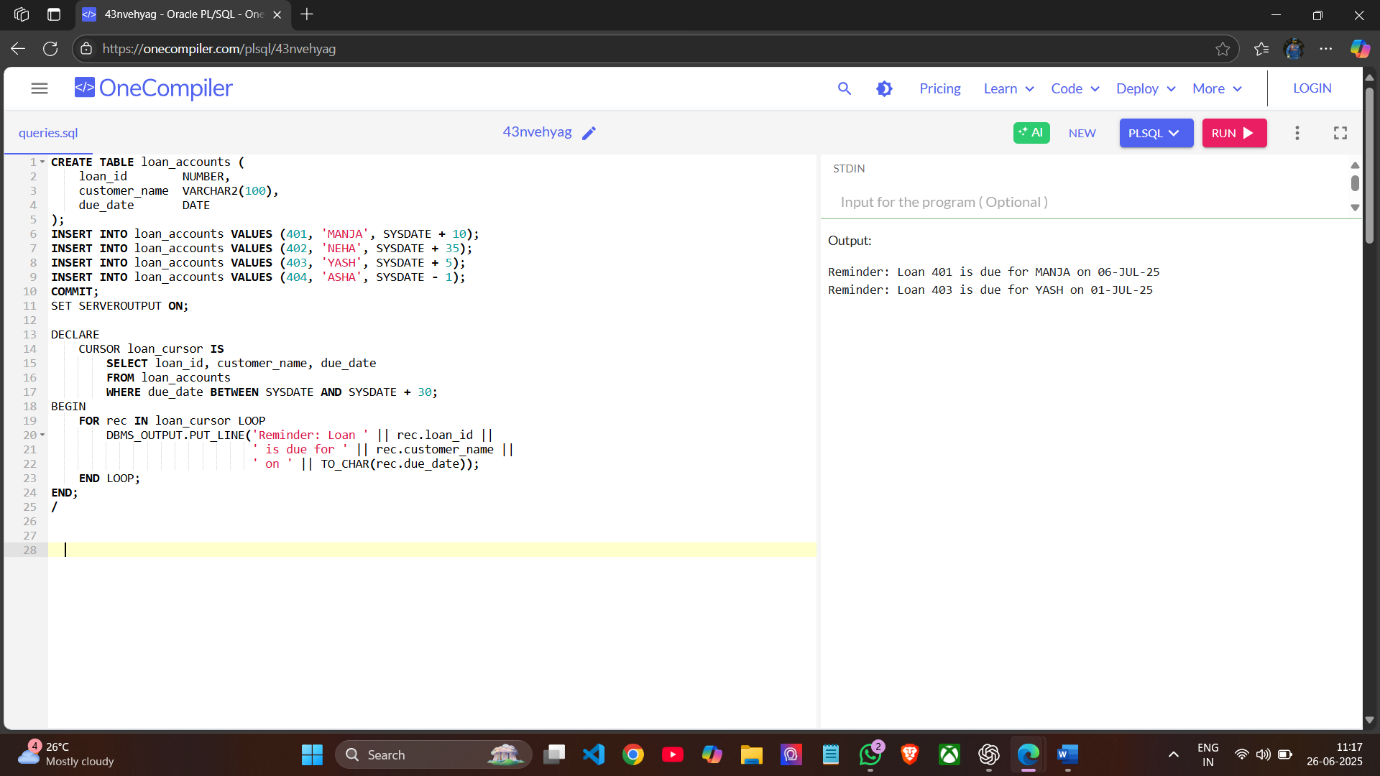
' on ' || TO\_CHAR(rec.due\_date));

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 for above question:-**

CREATE TABLE Accounts (

AccountID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER(10, 2)

);

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

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

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

INSERT INTO Accounts VALUES (104, 'CURRENT', 300000);

INSERT INTO Accounts VALUES (105, 'SAVINGS', 43000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

v\_rows\_updated NUMBER;

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'SAVINGS';

v\_rows\_updated := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE(v\_rows\_updated || ' savings account(s) updated with intrest.');

COMMIT;

END;

/

SET SERVEROUTPUT ON;

BEGIN

ProcessMonthlyInterest;

END;

/

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.  
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 for above question:-**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DepartmentID NUMBER,

Salary NUMBER

);

INSERT INTO Employees VALUES (1, 'Abhi', 101, 50000);

INSERT INTO Employees VALUES (2, 'manjunath', 101, 6000);

INSERT INTO Employees VALUES (3, 'sujay', 102, 55000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_DepartmentID IN NUMBER,

p\_BonusPercent IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE DepartmentID = p\_DepartmentID;

COMMIT;

END;

/

BEGIN

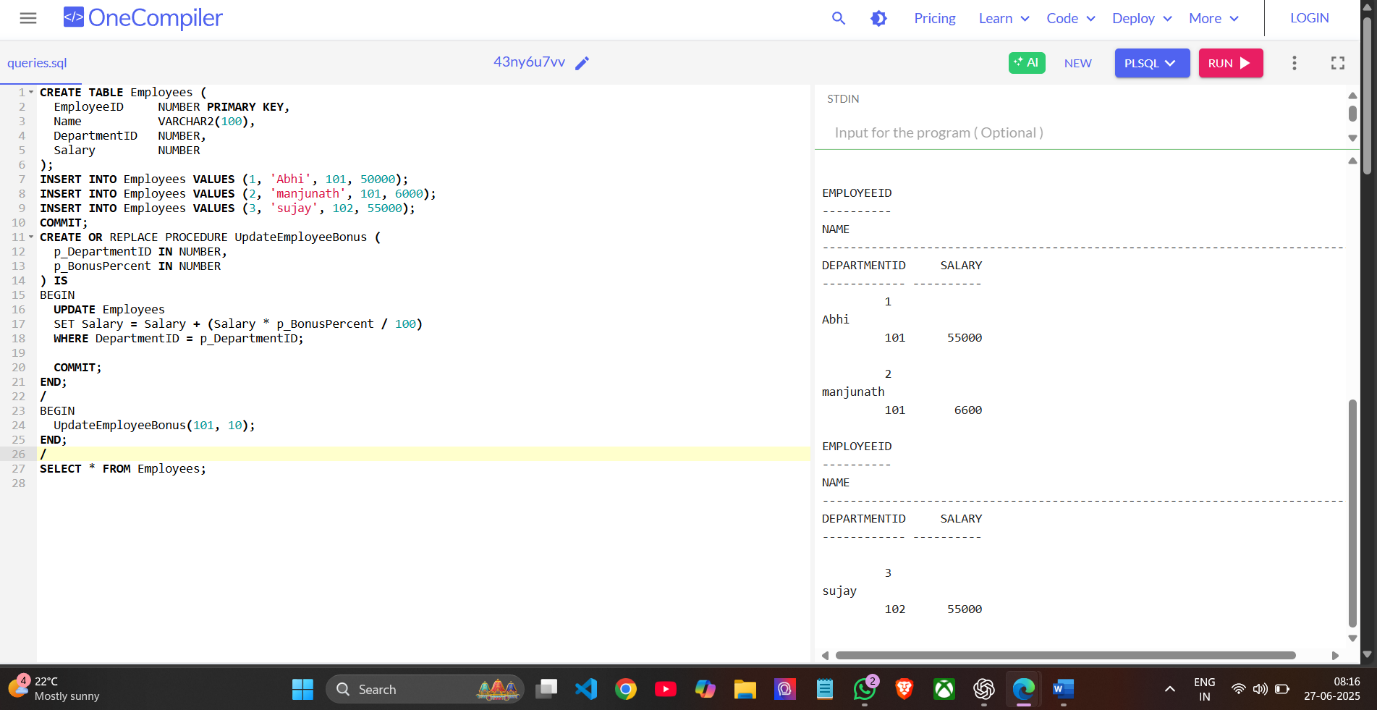
UpdateEmployeeBonus(101, 10);

END;

/

SELECT \* FROM Employees;

**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 for above question:-**

CREATE TABLE customer\_accounts (

acc\_id NUMBER PRIMARY KEY,

holder\_name VARCHAR2(50),

balance NUMBER

);

INSERT INTO customer\_accounts VALUES (1001, 'Ramesh', 6000);

INSERT INTO customer\_accounts VALUES (1002, 'Suresh', 4000);

INSERT INTO customer\_accounts VALUES (1003, 'Mahesh', 2000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (

sender\_id IN NUMBER,

receiver\_id IN NUMBER,

amount IN NUMBER

) IS

sender\_balance NUMBER;

BEGIN

SELECT balance INTO sender\_balance

FROM customer\_accounts

WHERE acc\_id = sender\_id;

IF sender\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Not enough balance in sender account.');

RETURN;

END IF;

UPDATE customer\_accounts

SET balance = balance - amount

WHERE acc\_id = sender\_id;

UPDATE customer\_accounts

SET balance = balance + amount

WHERE acc\_id = receiver\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(

'Transfer of Rs. ' || amount ||

' from account ' || sender\_id ||

' to account ' || receiver\_id || ' completed successfully.'

);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Account not found.');

ROLLBACK;

WHEN OTHERS THEN

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

ROLLBACK;

END;

/

BEGIN

TransferFunds(1001, 1002, 1000);

END;

/

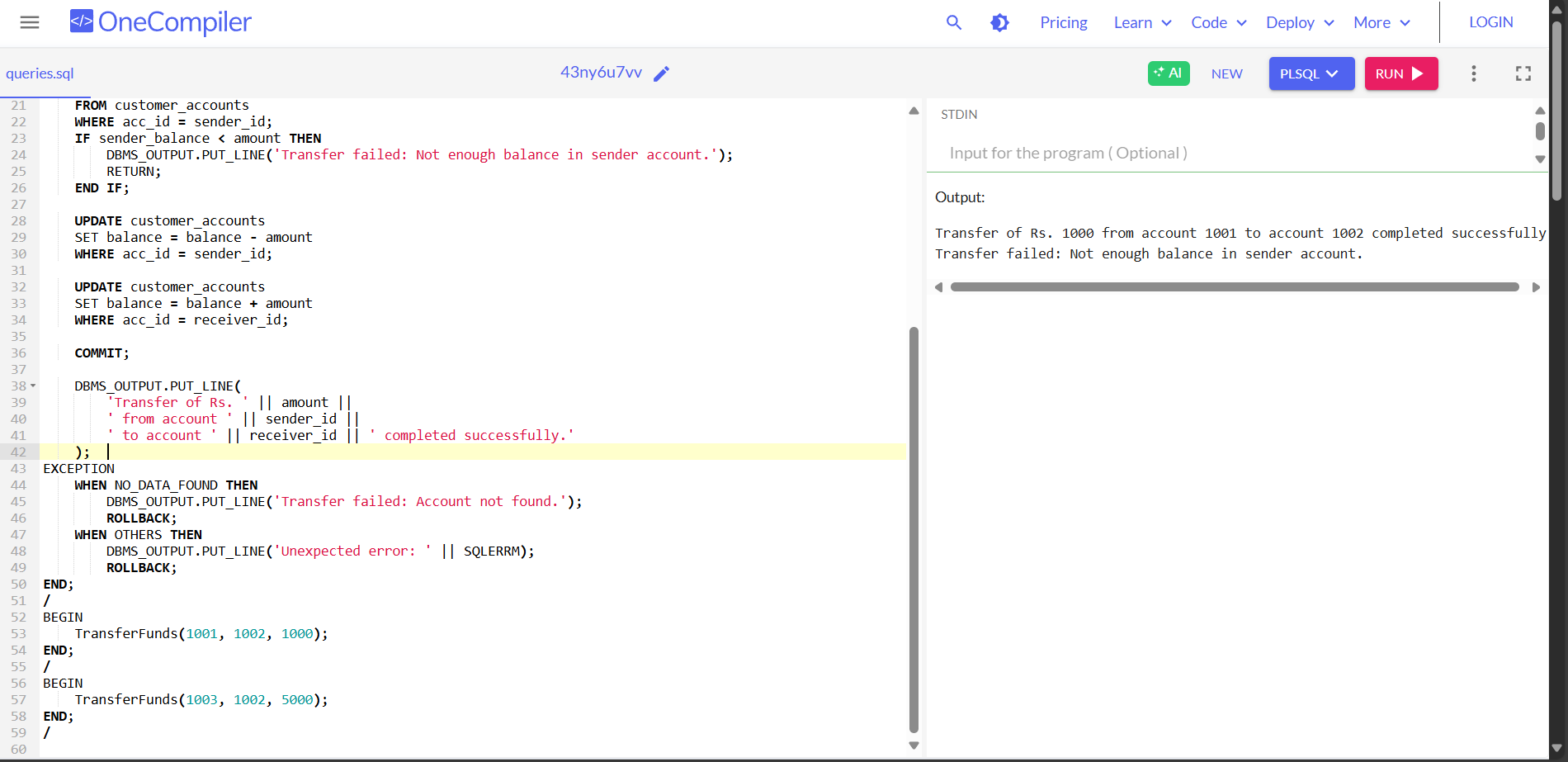
BEGIN

TransferFunds(1003, 1002, 5000);

END;

/

**OUTPUT:**

****