**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.

DECLARE

CURSOR senior\_customers IS

SELECT c.CustomerID

FROM Customers c

WHERE TRUNC(MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12) > 60;

BEGIN

FOR cust\_rec IN senior\_customers LOOP

UPDATE Loans

SET InterestRate = InterestRate \* 0.99

WHERE CustomerID = cust\_rec.CustomerID;

END LOOP;

COMMIT;

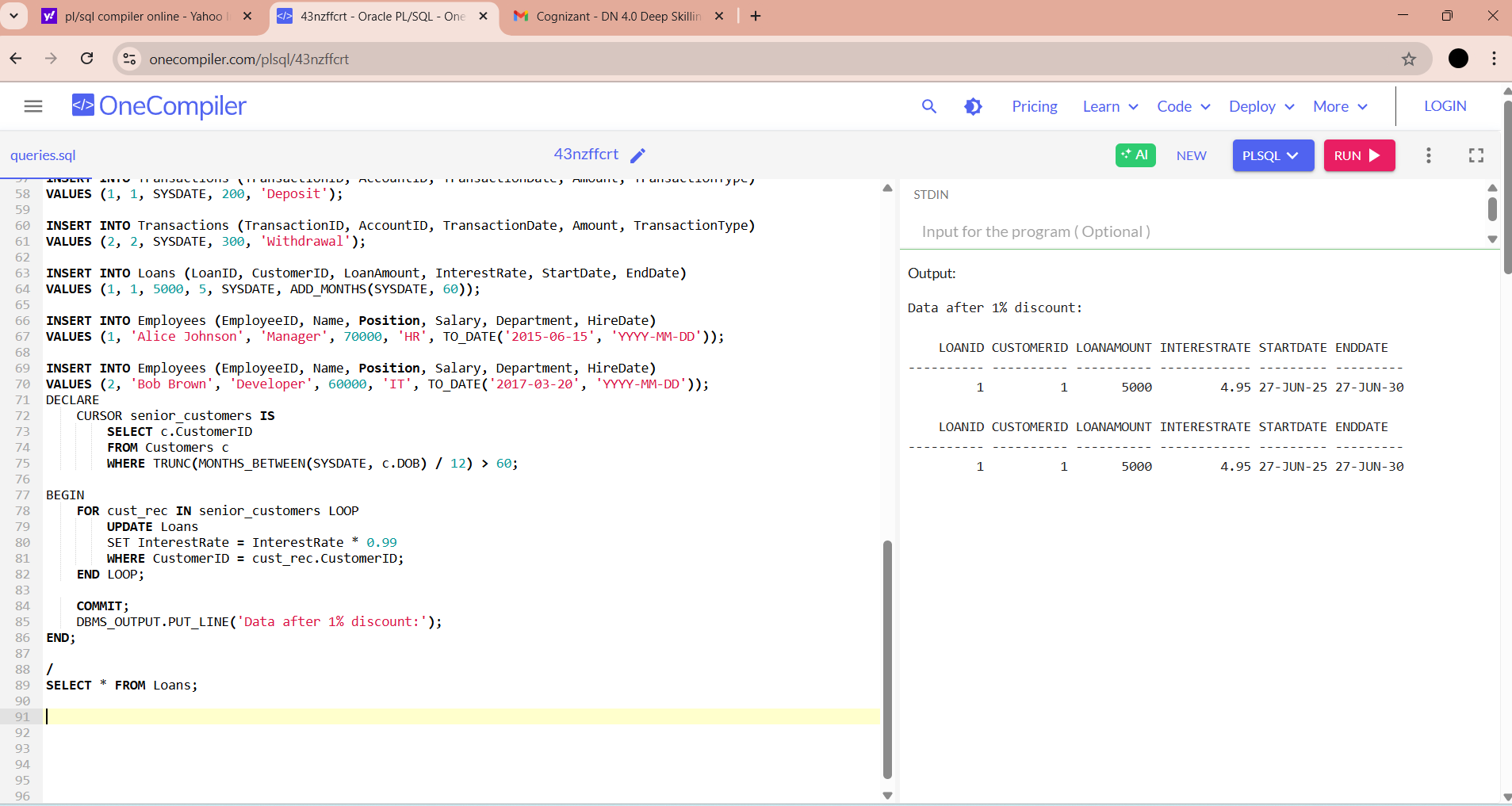
DBMS\_OUTPUT.PUT\_LINE('Data after 1% discount:');

END;

/

SELECT \* FROM Loans;

**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.

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

DECLARE

CURSOR senior\_customers IS

SELECT c.CustomerID,c.Balance

FROM Customers c;

BEGIN

FOR cust\_rec IN senior\_customers LOOP

IF cust\_rec.Balance>10000 then

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust\_rec.CustomerID;

else

UPDATE Customers

SET IsVIP='FALSE'

WHERE CustomerID=cust\_rec.CustomerID;

END IF;

END LOOP;

COMMIT;

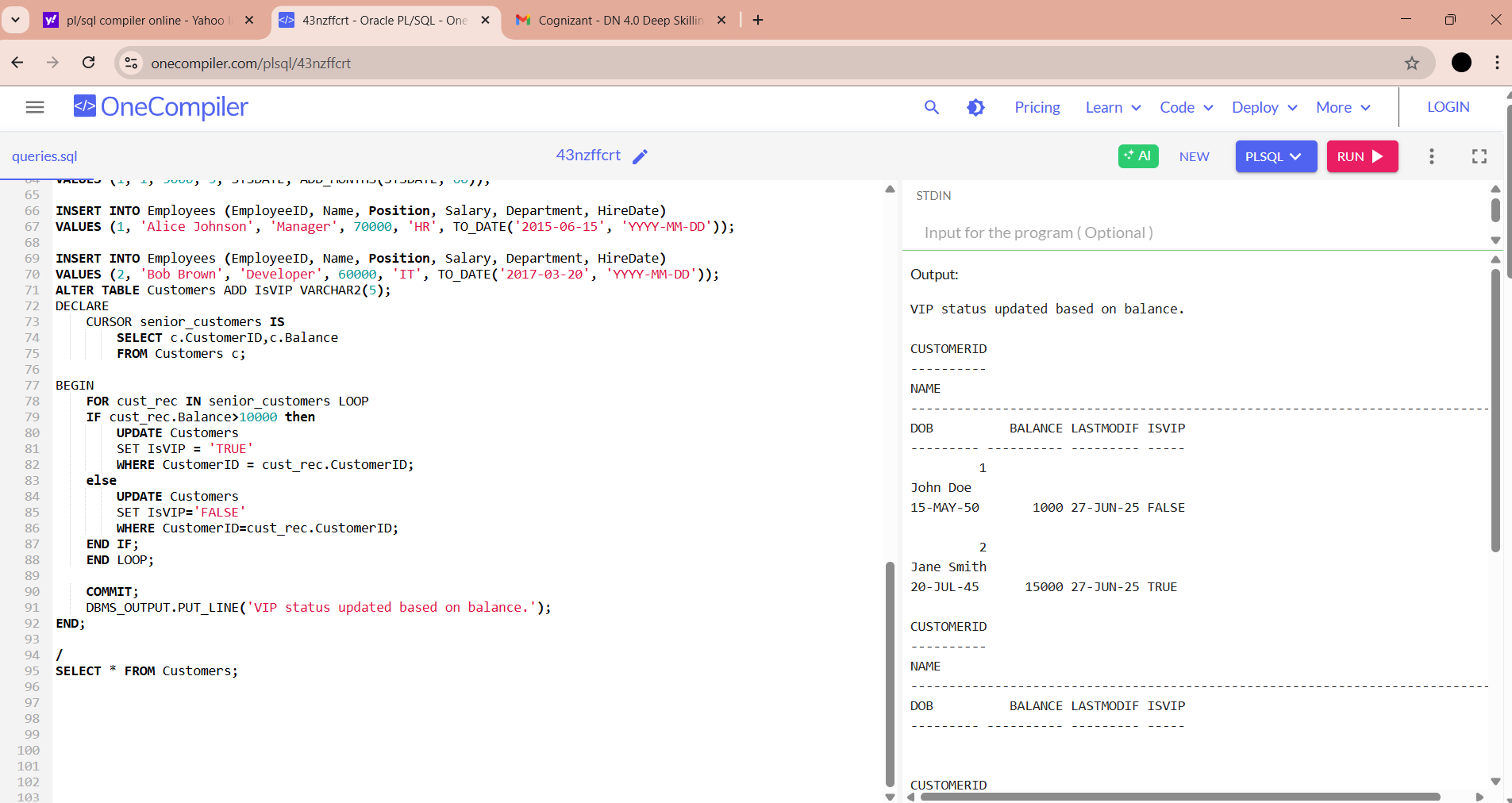
DBMS\_OUTPUT.PUT\_LINE('VIP status updated based on balance.');

END;

/

SELECT \* FROM Customers;

**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.**

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 1));

SELECT\* FROM Loans;

BEGIN

FOR loan\_rec IN ( SELECT l.LoanID,l.EndDate,c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID=c.CustomerID

WHERE l.EndDate between SYSDATE AND SYSDATE+30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Remainder: Dear' || loan\_rec.Name|| ',your loan(LoanId: '||loan\_rec.LoanId||') is due on '|| TO\_CHAR(loan\_rec.EndDate, 'DD-MON-YYYY')

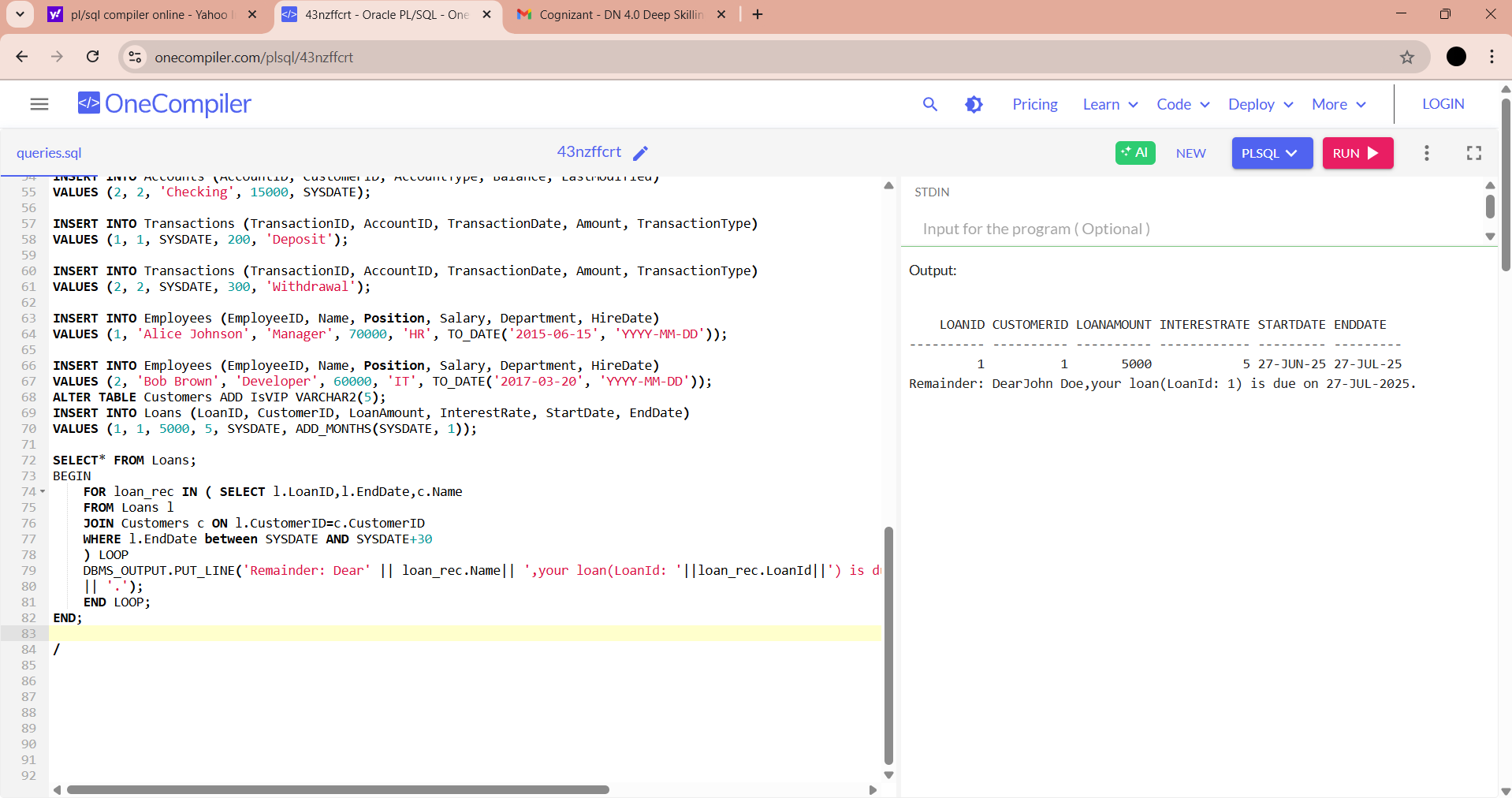
|| '.');

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.**

SELECT \* FROM Accounts;

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 for all savings accounts.');

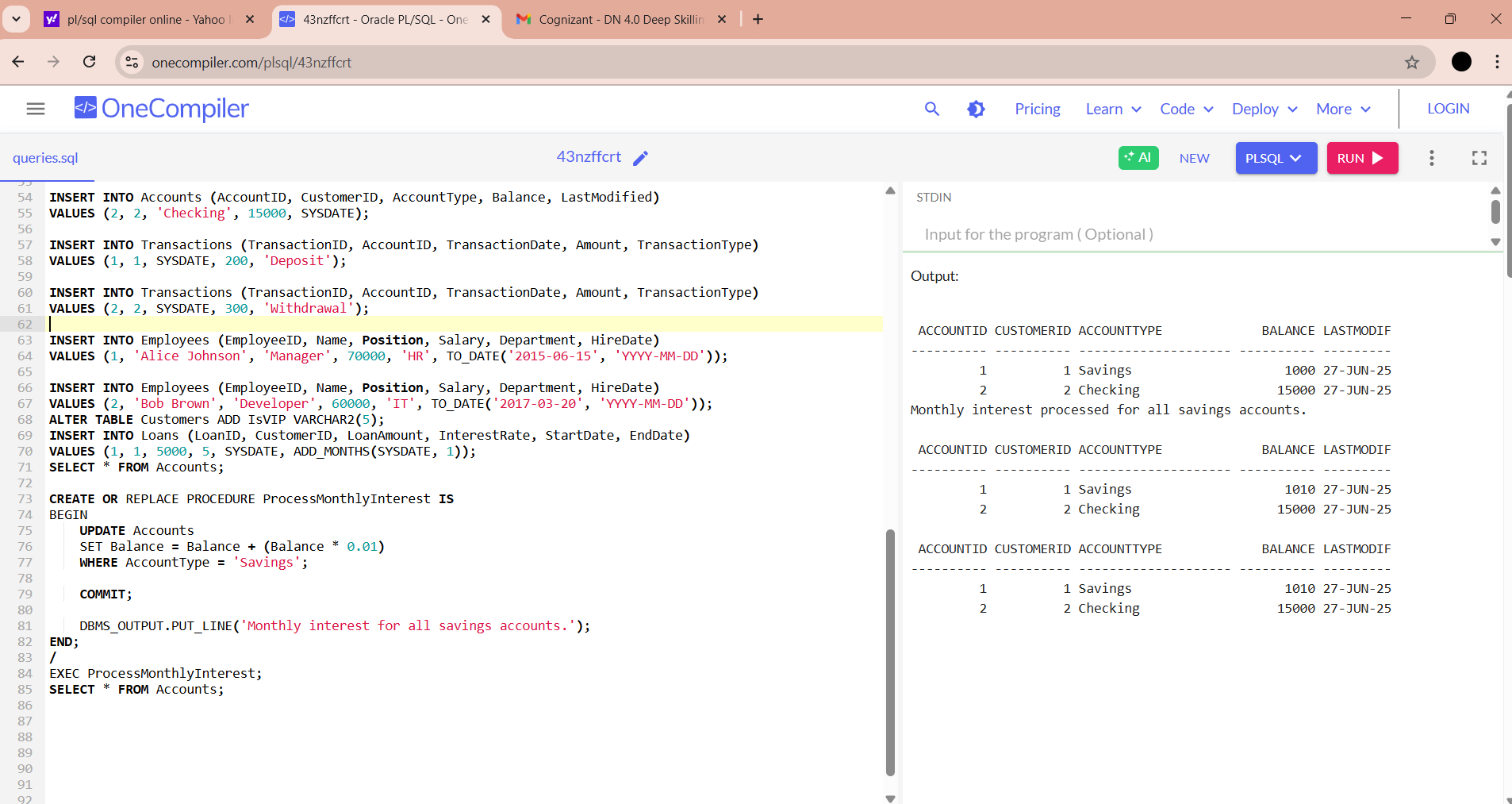
END;

/

EXEC ProcessMonthlyInterest;

SELECT \* FROM Accounts;

**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.**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department IN VARCHAR2,

p\_BonusPercent IN NUMBER

)

IS

BEGIN

UPDATE Employees

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

WHERE Department = p\_Department;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bonus of '||p\_BonusPercent||'% applied to department: '||p\_Department);

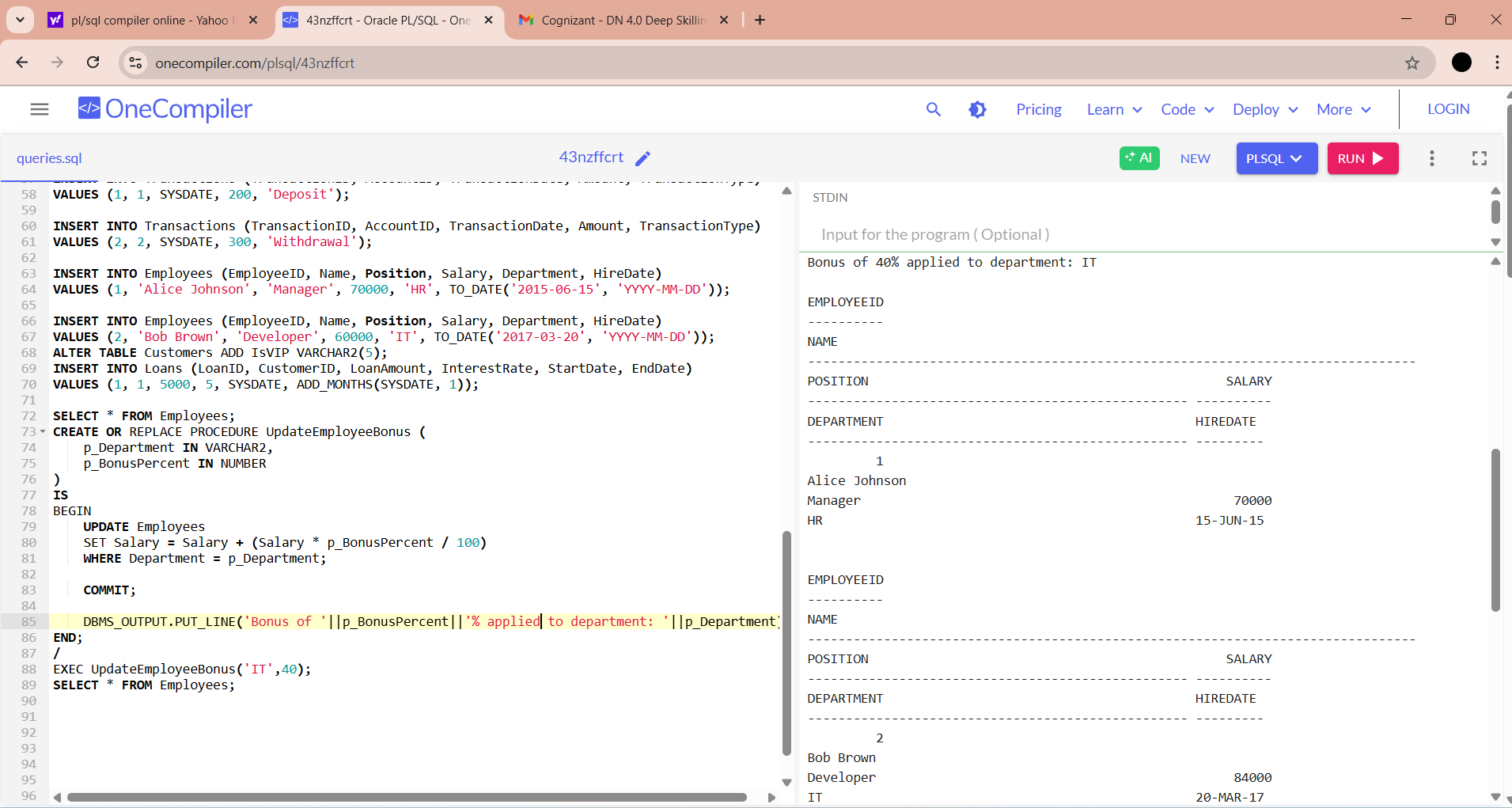
END;

/

EXEC UpdateEmployeeBonus('IT',40);

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.

SELECT \* FROM Accounts;

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_SourceID IN NUMBER,

p\_DestinationID IN NUMBER,

p\_amount IN NUMBER

)

IS

v\_SourceBalance NUMBER;

BEGIN

SELECT Balance INTO v\_SourceBalance

FROM Accounts

WHERE AccountID=p\_SourceID;

IF v\_SourceBalance<p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001,'INSUFFICIENT BALANCE IN SOURCE ACCOUNT.');

END IF;

UPDATE Accounts

SET Balance=Balance-P\_amount

WHERE AccountID=p\_SourceID;

UPDATE Accounts

SET Balance=Balance+P\_amount

WHERE AccountID=p\_DestinationID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('TRANSFER OF '||p\_amount||' FROM ACCOUNT'||p\_SourceID||' TO ACCOUNT'||p\_DestinationID||'COMPLETED SUCCESSFULLY.');

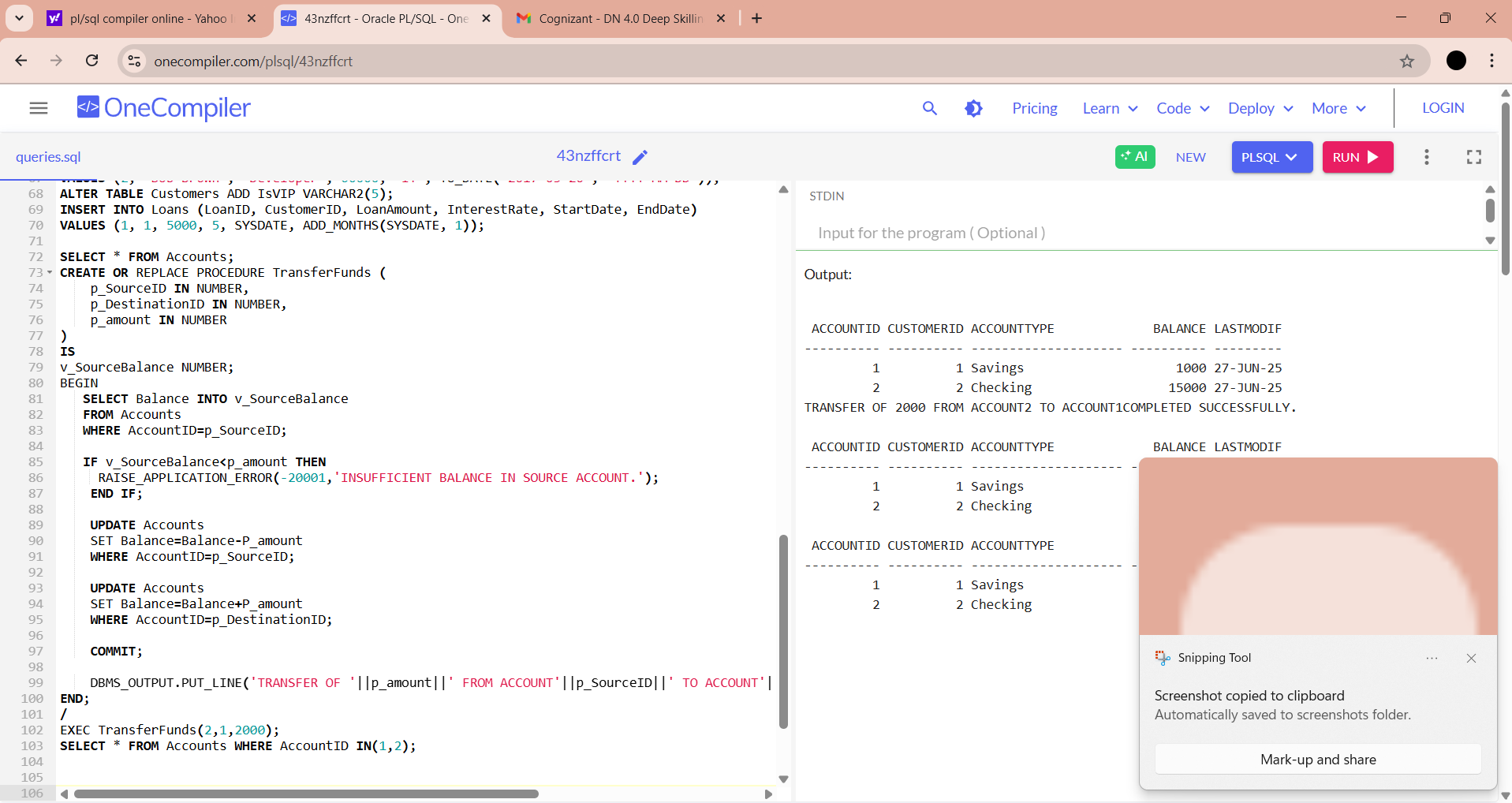
END;

/

EXEC TransferFunds(2,1,2000);

SELECT \* FROM Accounts WHERE AccountID IN(1,2);

**OUTPUT:**

****