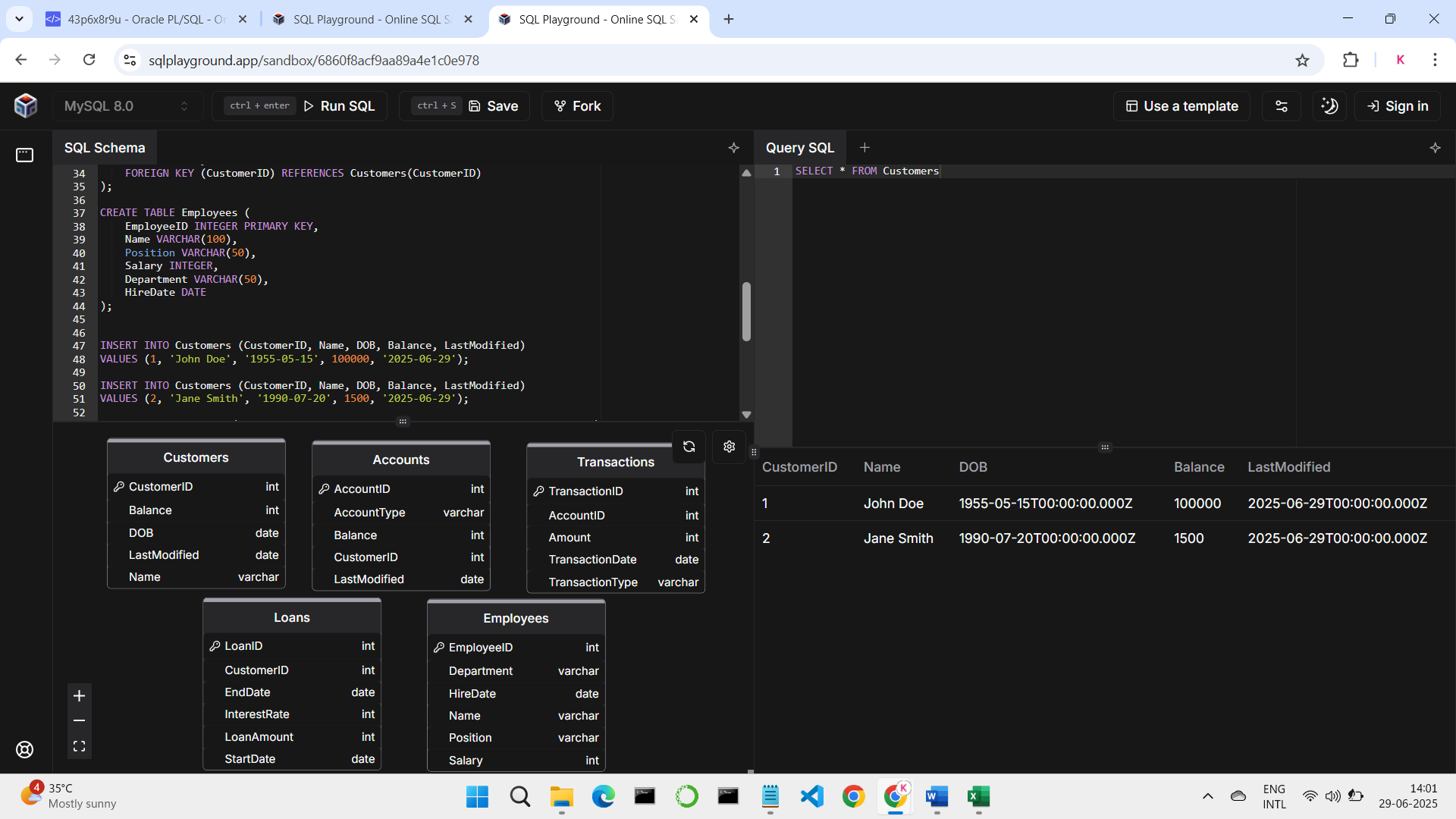
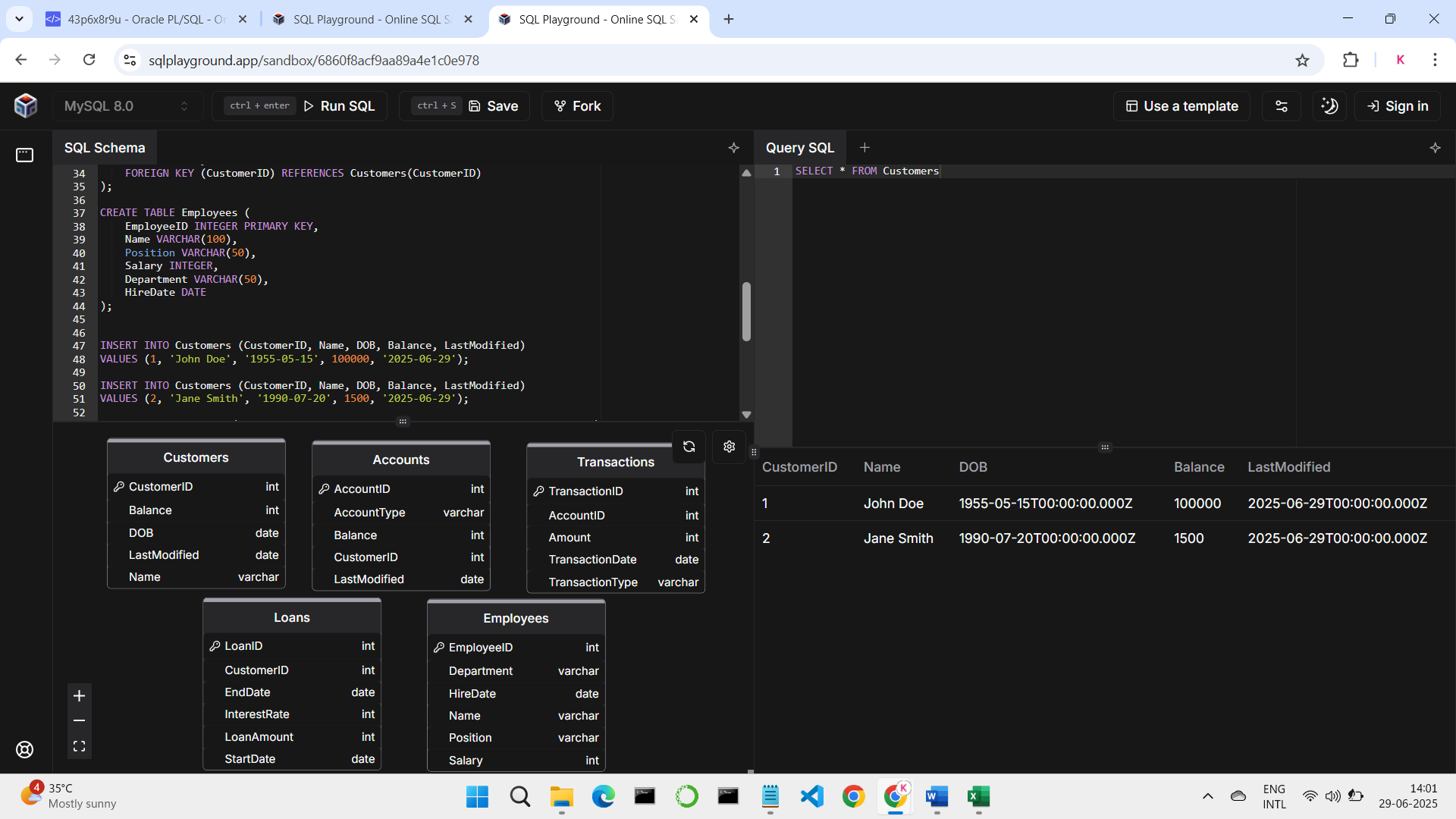
**Superset ID: 6385480**

**Java FSE Mandatory hands-on Assignment Week - 2 PL/SQL**

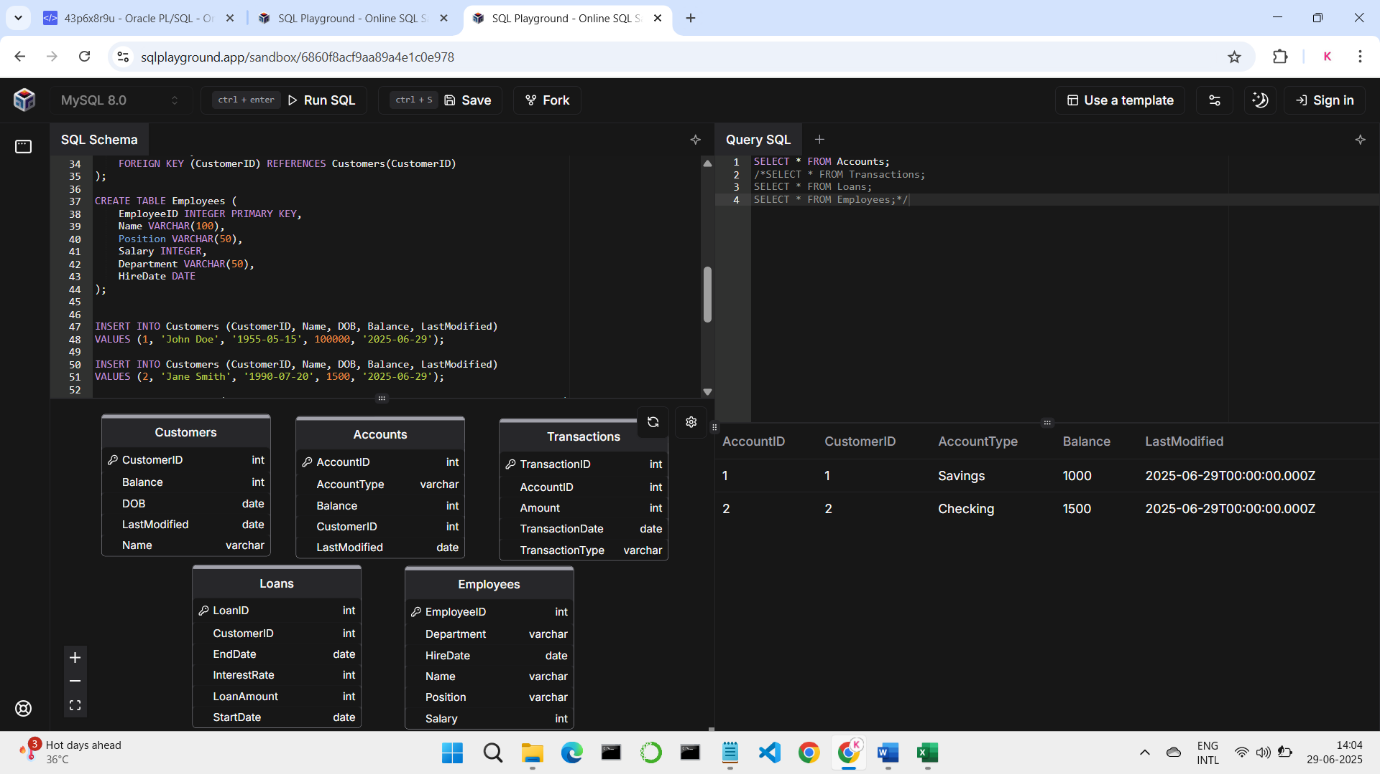
**Database Schema:**

****

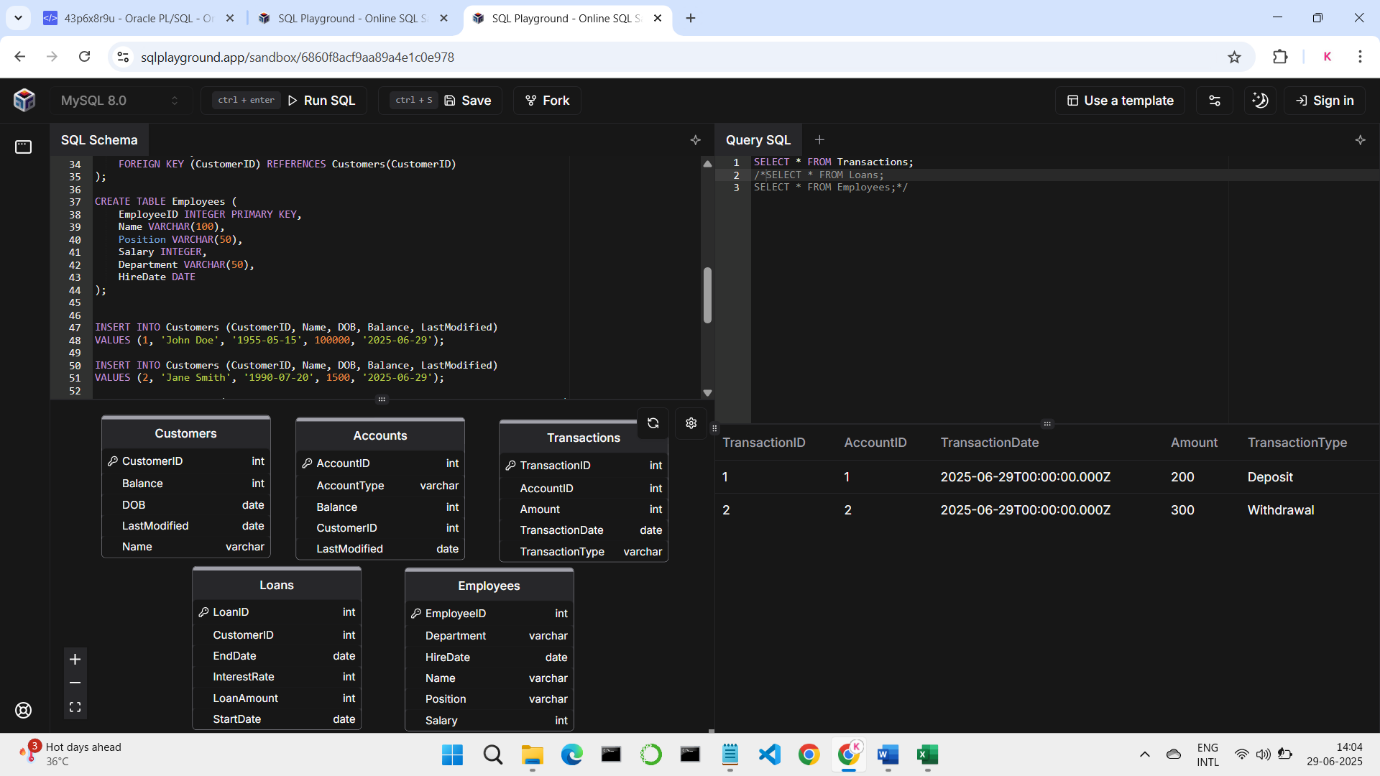
**Table Customers:**

****

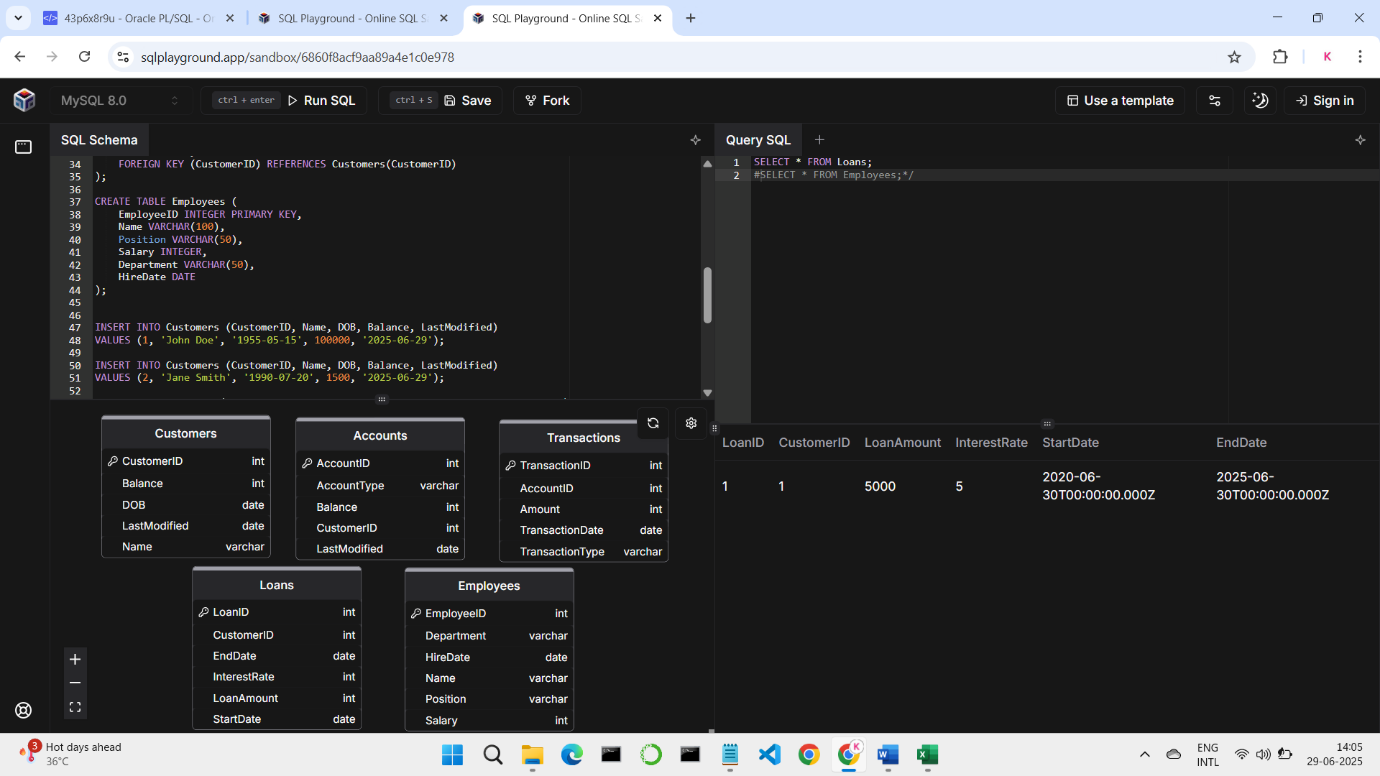
**Table Accounts:**

****

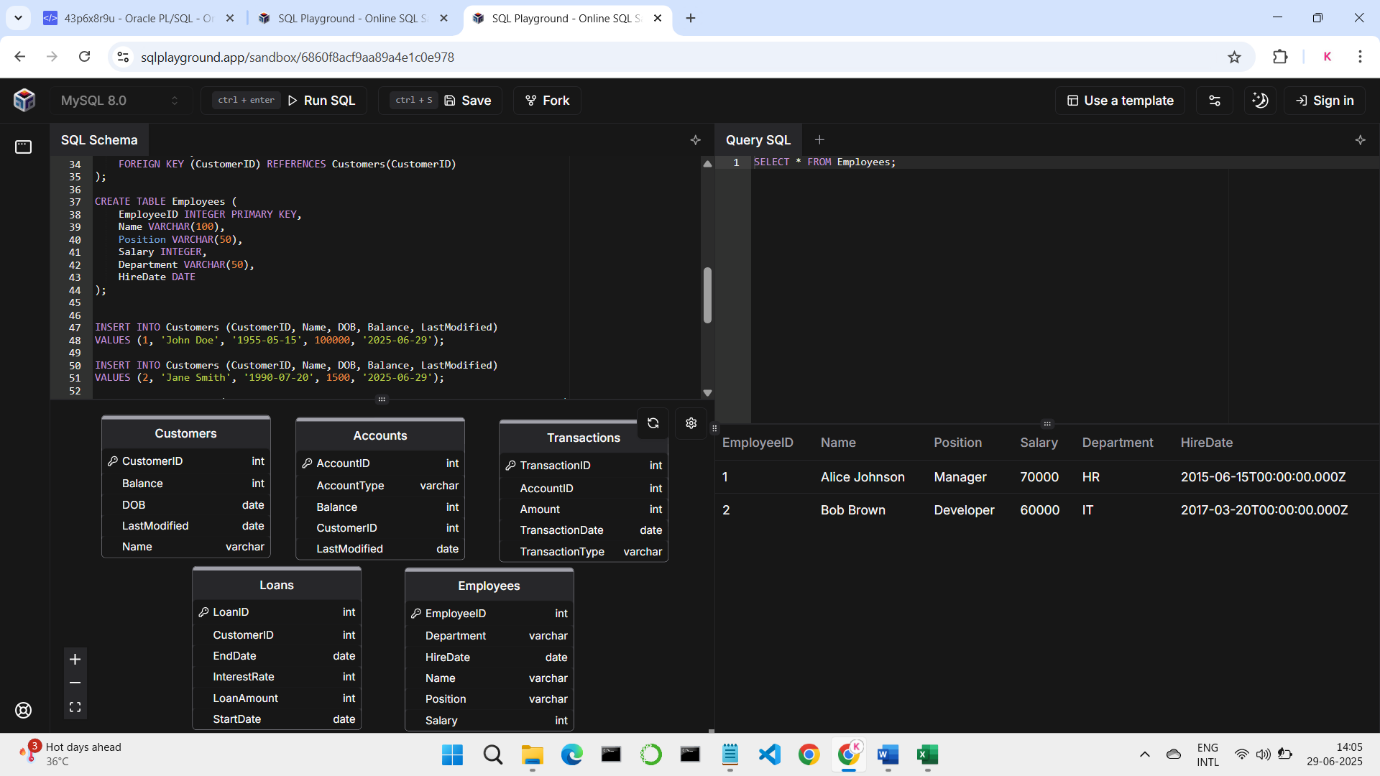
**Table Transactions:**

****

**Table Loans:**

****

**Table Employees:**

****

**Exercise 1: Control Structures**

**Question – 1**

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.

BEGIN

FOR c IN (

SELECT CustomerID, TRUNC(MONTHS\_BETWEEN(SYSDATE, DOB) / 12) AS Age

FROM Customers

) LOOP

IF c.Age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = c.CustomerID;

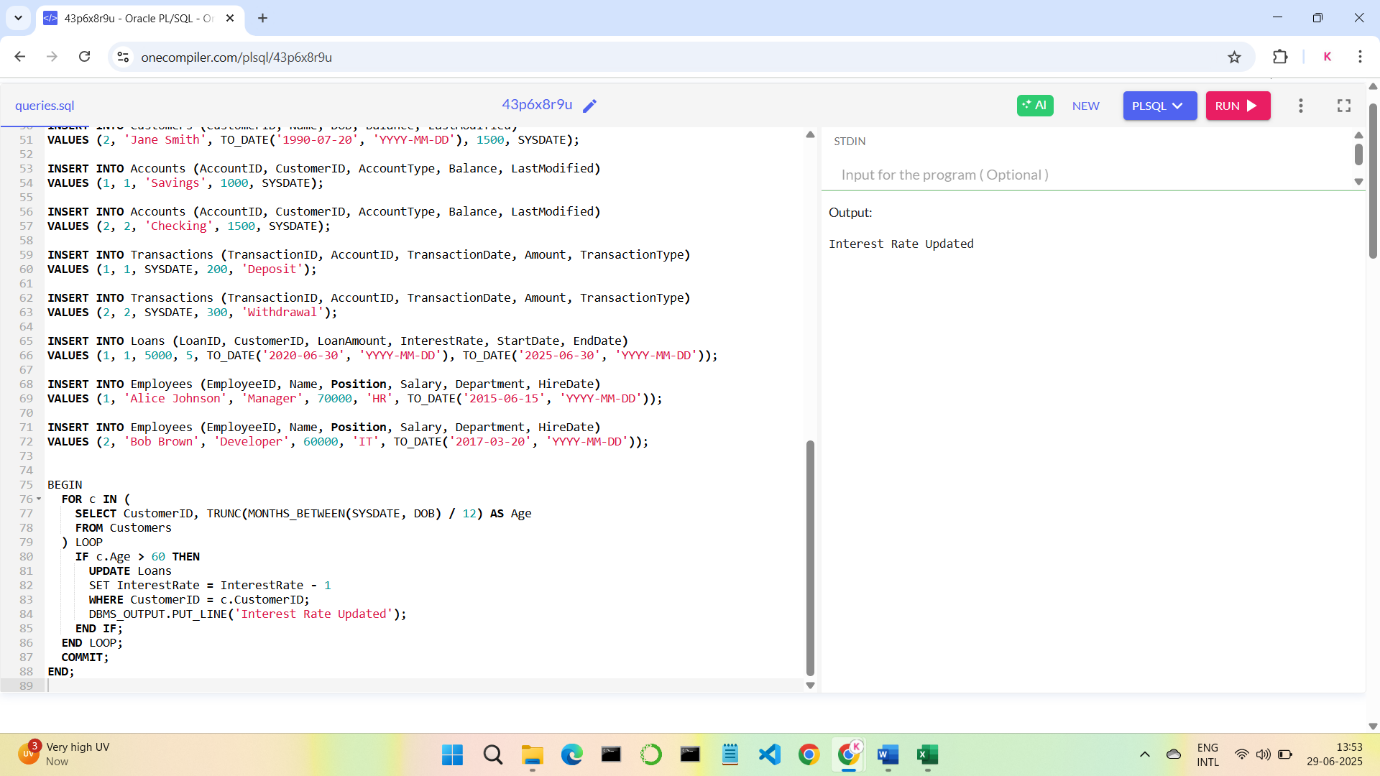
DBMS\_OUTPUT.PUT\_LINE('Interest Rate Updated');

END IF;

END LOOP;

COMMIT;

END;



**Question – 2**

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.

#To add a IsVIP column to the table Customers

ALTER TABLE Customers ADD IsVIP CHAR(1);

BEGIN

FOR c IN (

SELECT CustomerID, Balance

FROM Customers

) LOOP

IF c.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = c.CustomerID;

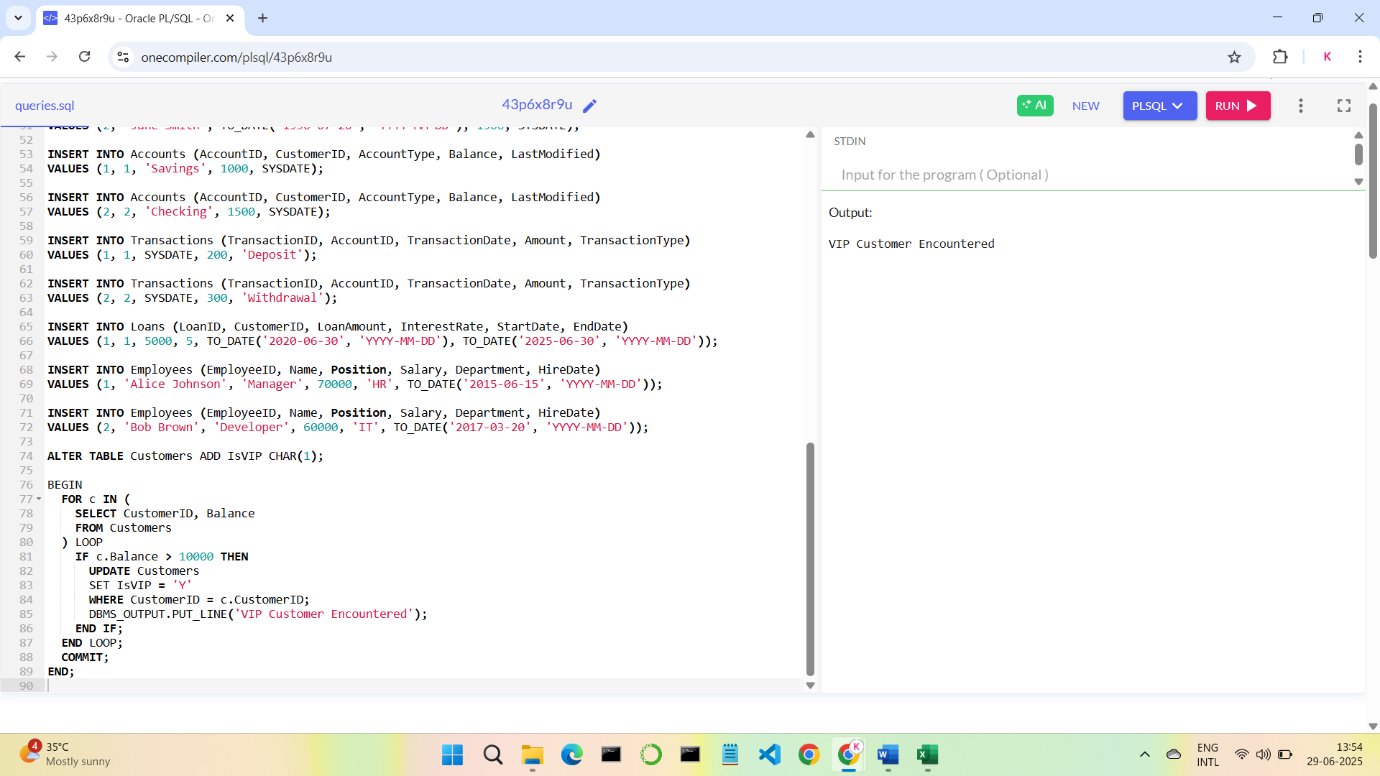
DBMS\_OUTPUT.PUT\_LINE('VIP Customer Encountered');

END IF;

END LOOP;

COMMIT;

END;



**Question – 3**

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

BEGIN

FOR l IN (

SELECT l.LoanID, l.CustomerID, l.EndDate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

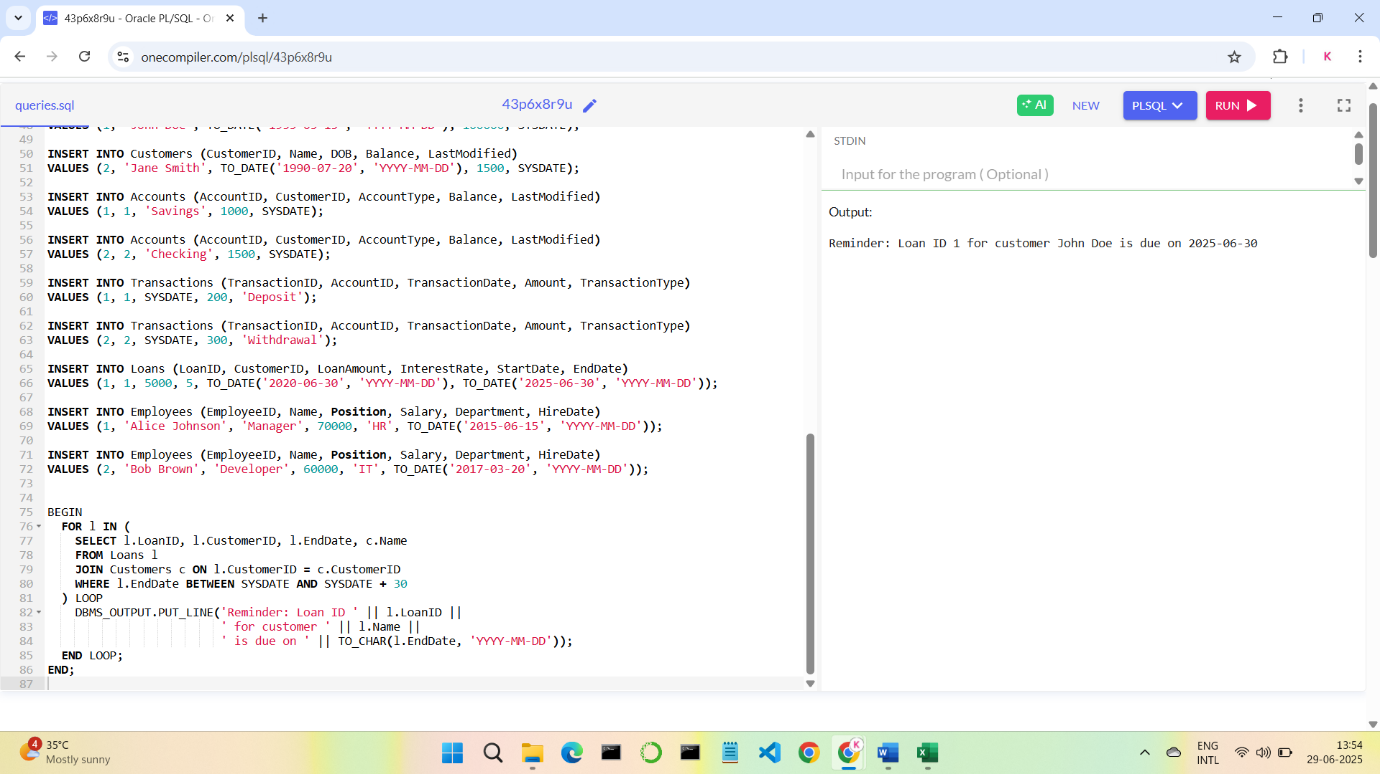
DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || l.LoanID ||

' for customer ' || l.Name ||

' is due on ' || TO\_CHAR(l.EndDate, 'YYYY-MM-DD'));

END LOOP;

END;



**Exercise 3: Stored Procedures**

**Question – 1**

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.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE Accounts

SET Balance = (Balance \* 1.01),LastModified = SYSDATE

WHERE AccountType = 'Savings';

DBMS\_OUTPUT.PUT\_LINE('Interest updated for all savings accounts.');

COMMIT;

END;

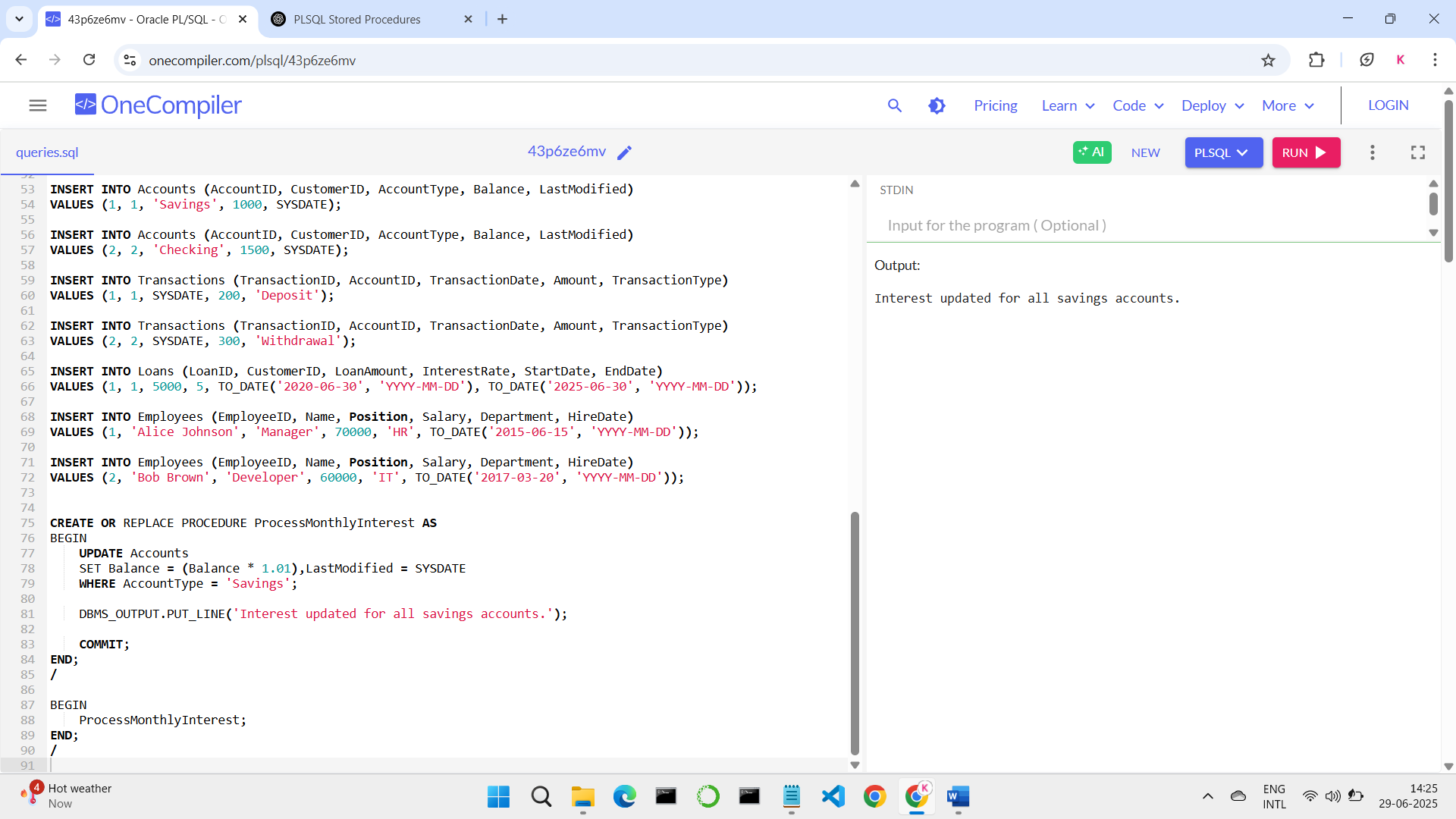
/

BEGIN

ProcessMonthlyInterest;

END;

/



**Question – 2**

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

) AS

BEGIN

UPDATE Employees

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

WHERE Department = p\_Department;

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Before Bonus Update:');

FOR emp IN (SELECT \* FROM Employees) LOOP

DBMS\_OUTPUT.PUT\_LINE(emp.Name||' - Department: '||emp.Department||' - Salary: '||emp.Salary);

END LOOP;

UpdateEmployeeBonus('IT', 10);

DBMS\_OUTPUT.PUT\_LINE('After Bonus Update:');

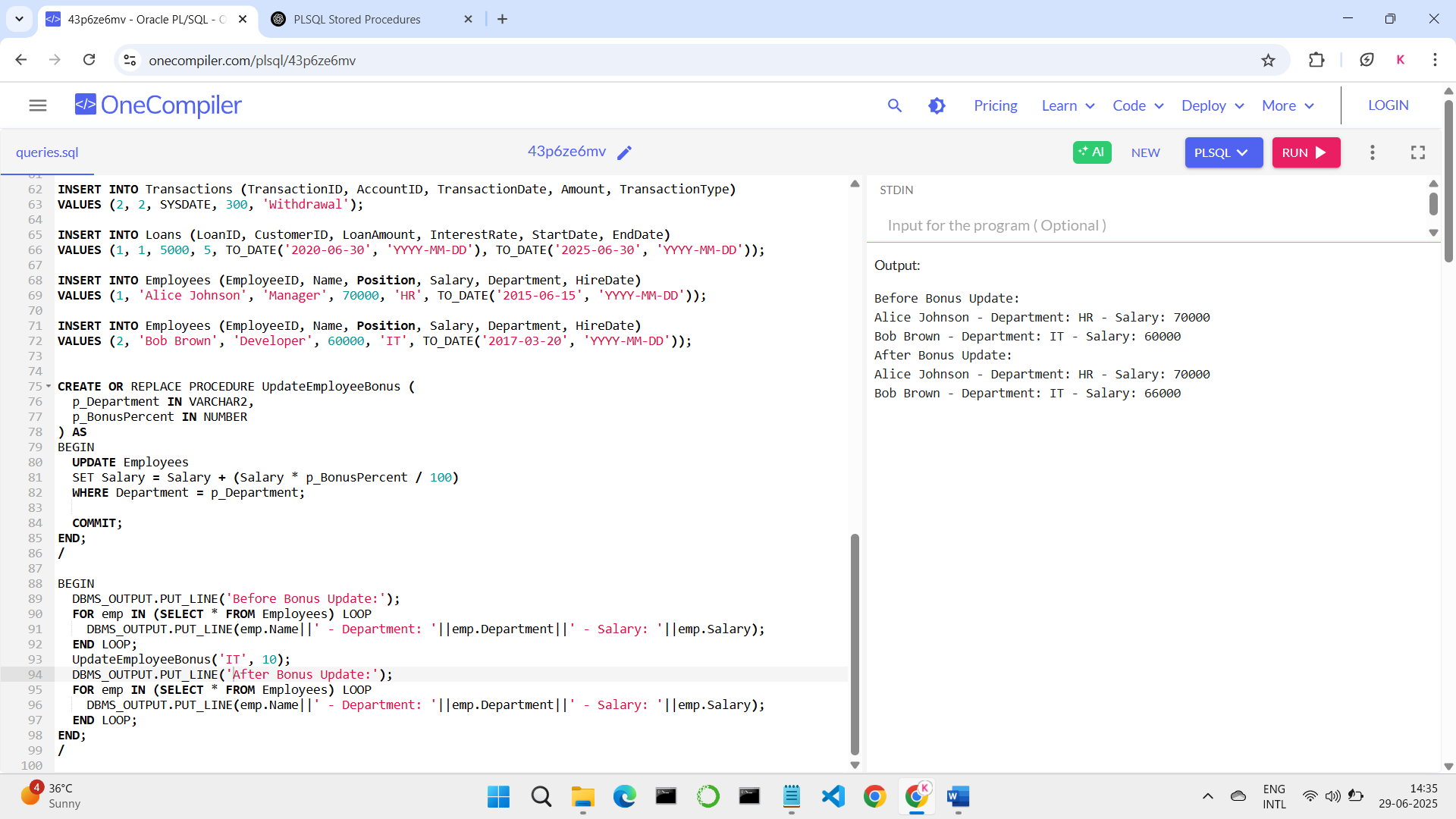
FOR emp IN (SELECT \* FROM Employees) LOOP

DBMS\_OUTPUT.PUT\_LINE(emp.Name||' - Department: '||emp.Department||' - Salary: '||emp.Salary);

END LOOP;

END;

/



**Question – 3**

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.

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) AS

v\_FromBalance NUMBER;

ex\_InsufficientBalance EXCEPTION;

BEGIN

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID

FOR UPDATE;

IF v\_FromBalance < p\_Amount THEN

RAISE ex\_InsufficientBalance;

END IF;

UPDATE Accounts

SET Balance = Balance - p\_Amount, LastModified = SYSDATE

WHERE AccountID = p\_FromAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount, LastModified = SYSDATE

WHERE AccountID = p\_ToAccountID;

DBMS\_OUTPUT.PUT\_LINE('Transaction Successful');

COMMIT;

EXCEPTION

WHEN ex\_InsufficientBalance THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in source account.');

ROLLBACK;

END;

/

BEGIN

TransferFunds(1, 2, 500);

TransferFunds(1, 2, 2000);

END;

/

