**Database And Management Systems**

**Lab File**

**Submitted by**

**Atharva Milind Deshmukh**

**(2124UCEM1035)**

**Subject**

**Database And Management Systems 24UCSES103**

**Under the Guidance of**

**Mr. Utkarsh Varshney Sir**

****

**Sanjivani Rural Education Society’s**

**“ SANJIVANI UNIVERSITY “**

**KOPARGAON – 423603, DIST : Ahilyanagar 2024-2025**

| **Sr.No** | **List Of Experiments** | **Page No** |
| --- | --- | --- |
| **1.** | Create a database and a table to store employee details. Perform basic operations like INSERT,UPDATE, and DELETE using SELECT queries. | **2** |
| **2.** | Create a table for storing student information. Insert sample data and perform basic operations: INSERT, UPDATE, DELETE, and SELECT. | **4** |
| **3.** | Create a table with columns for Employee lD, Name, Salary, Joining Date, and ActiveStatus using different data types. Insert sample data and perform queries to manipulate and retrieve data. | **6** |
| **4.** | Create a table to store employee information with constraints like Primary Key, Foreign Key, and Unique. Insert valid and invalid data to test the constraints. | **9** |
| **5.** | Testing Employee Constraints. To test constraints like PRIMARY KEY, UNIQUE, and CHECK by inserting invalid data into the Employee table. | **12** |
| **6.** | Using DDL commands to create tables and DML commands to insert, update, and delete data. Write SELECT queries to retrieve and verify data changes. | **14** |
| **7.** | Create a Sales table and use aggregate functions like COUNT, SUM, AVG, MIN, and MAX to summarize sales data and calculate statistics. | **16** |
| **8.** | Given Customers and Orders tables, write SQL queries to perform INNER JOIN, LEFT JOIN, and RIGHT JOIN to retrieve combined data for customer orders. | **19** |

**PRACTICAL NO 1**

Install and set up MySQL. Create a database and a table to store employee details. Perform basic operations like INSERT & DELETE.

**AIM**: To install and set up MySQL, create a database and a table to store employee details, and perform basic SQL operations such as INSERT, DELETE, SELECT, and USE to manage and manipulate data in a relational

database.

**CODE**:

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Department VARCHAR(50),

Salary DECIMAL(10, 2)

);

INSERT INTO Employee (EmpID, FirstName, LastName, Department, Salary)

VALUES

(100, 'Atharva', 'Deshmukh', 'Software', 85000.00),

(101, 'Rohan', 'Patil', 'HR', 50000.00),

(102, 'Mehul', 'Verma', 'Finance', 60000.00),

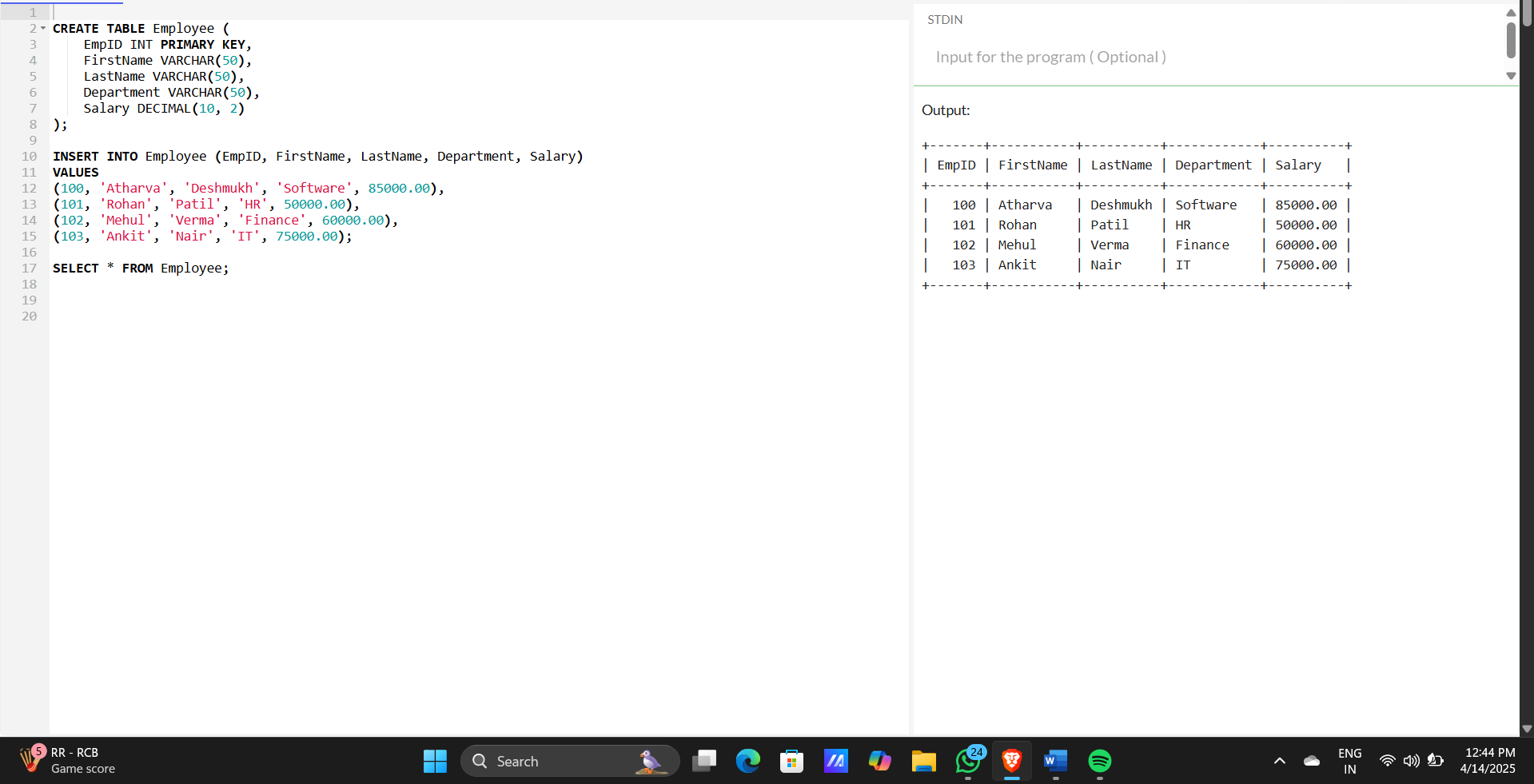
(103, 'Ankit', 'Nair', 'IT', 75000.00);

SELECT \* FROM Employee;

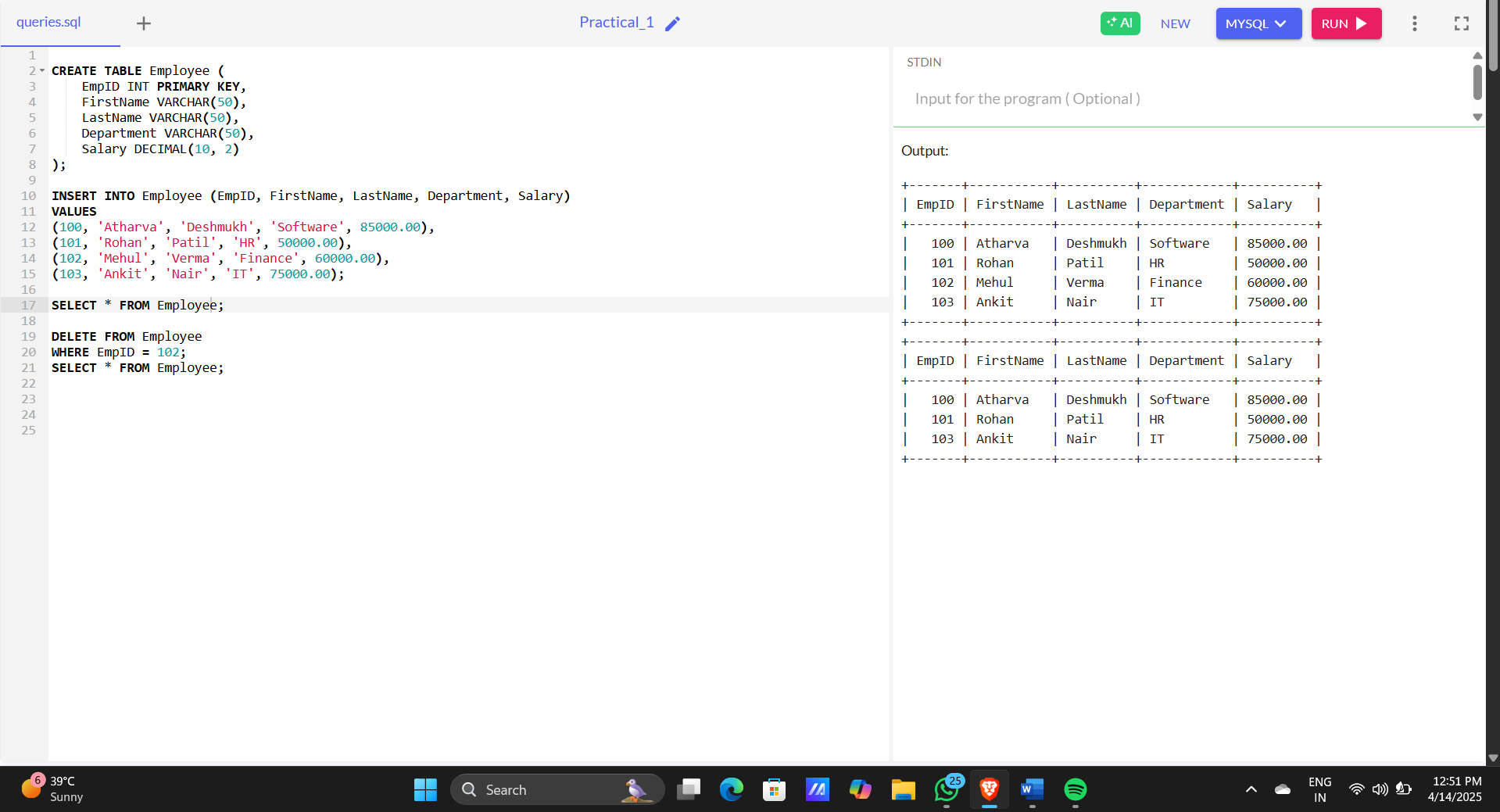
DELETE FROM Employee WHERE EmpID = 102;

SELECT \* FROM Employee;

The basic employees table



The updated table after the query is performed



**PRACTICAL NO 2**

**AIM**: Create a table for storing student information. Insert sample data and perform basic operations: INSERT, UPDATE, DELETE, and SELECT.

**CODE**:

CREATE TABLE students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

age INT,

email VARCHAR(100)

);

INSERT INTO students (student\_id, first\_name, last\_name, age, email)

VALUES

(1, 'Atharva', 'Deshmukh', 20, 'atharva.deshmukh@gmail.com'),

(2, 'Ram', 'Patil', 19, 'ram.patil@gmail.com'),

(3, 'Arjun', 'Raut', 18, 'arjun.raut@gmail.com'),

(4, 'Dhruv', 'Gore', 19, 'dhruv.gore@gmail.com');

SELECT \* FROM students;

UPDATE students

SET age = 20

WHERE first\_name = 'Ram' AND last\_name = 'Patil';

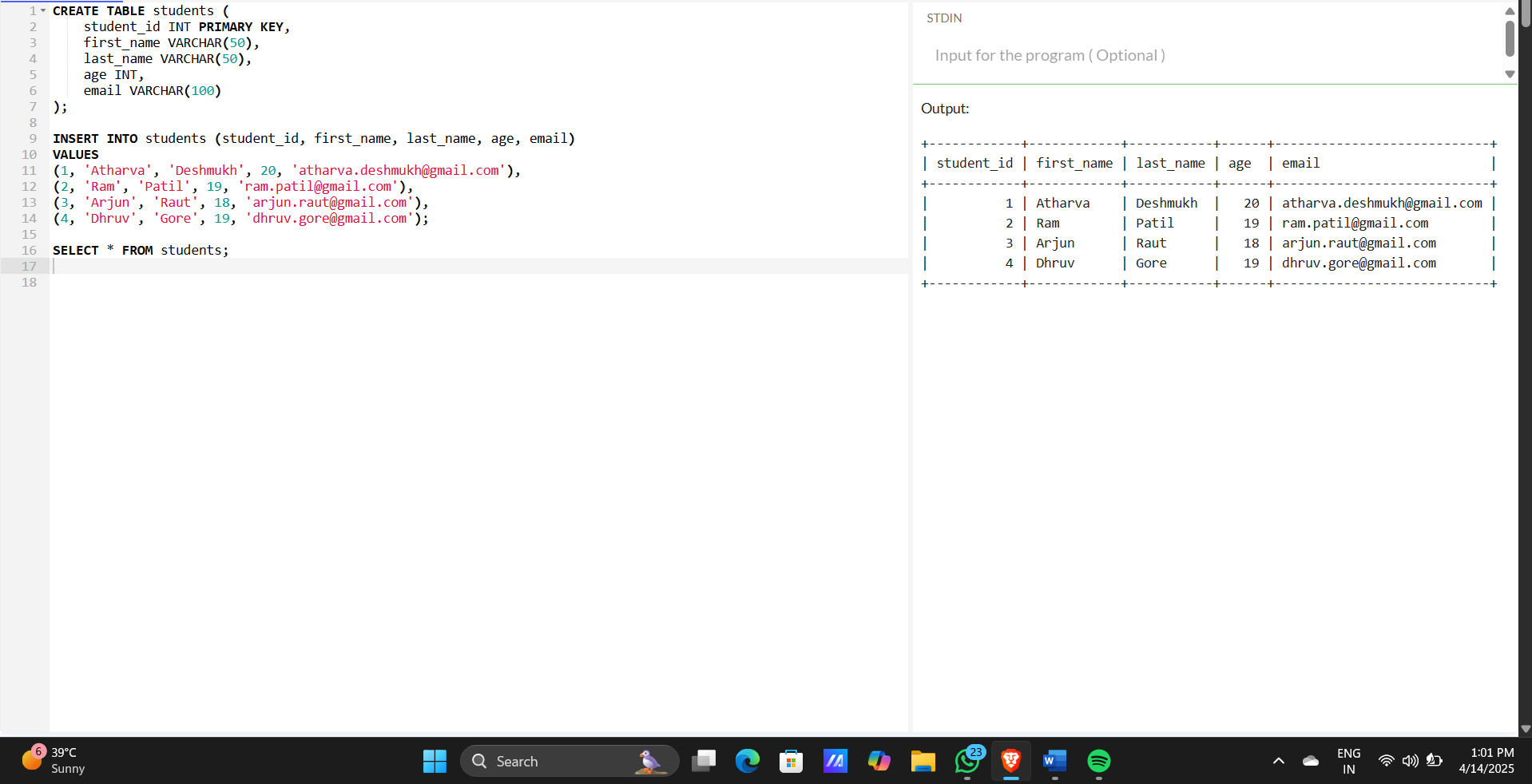
DELETE FROM students

WHERE first\_name = 'Arjun' AND last\_name = 'Raut';

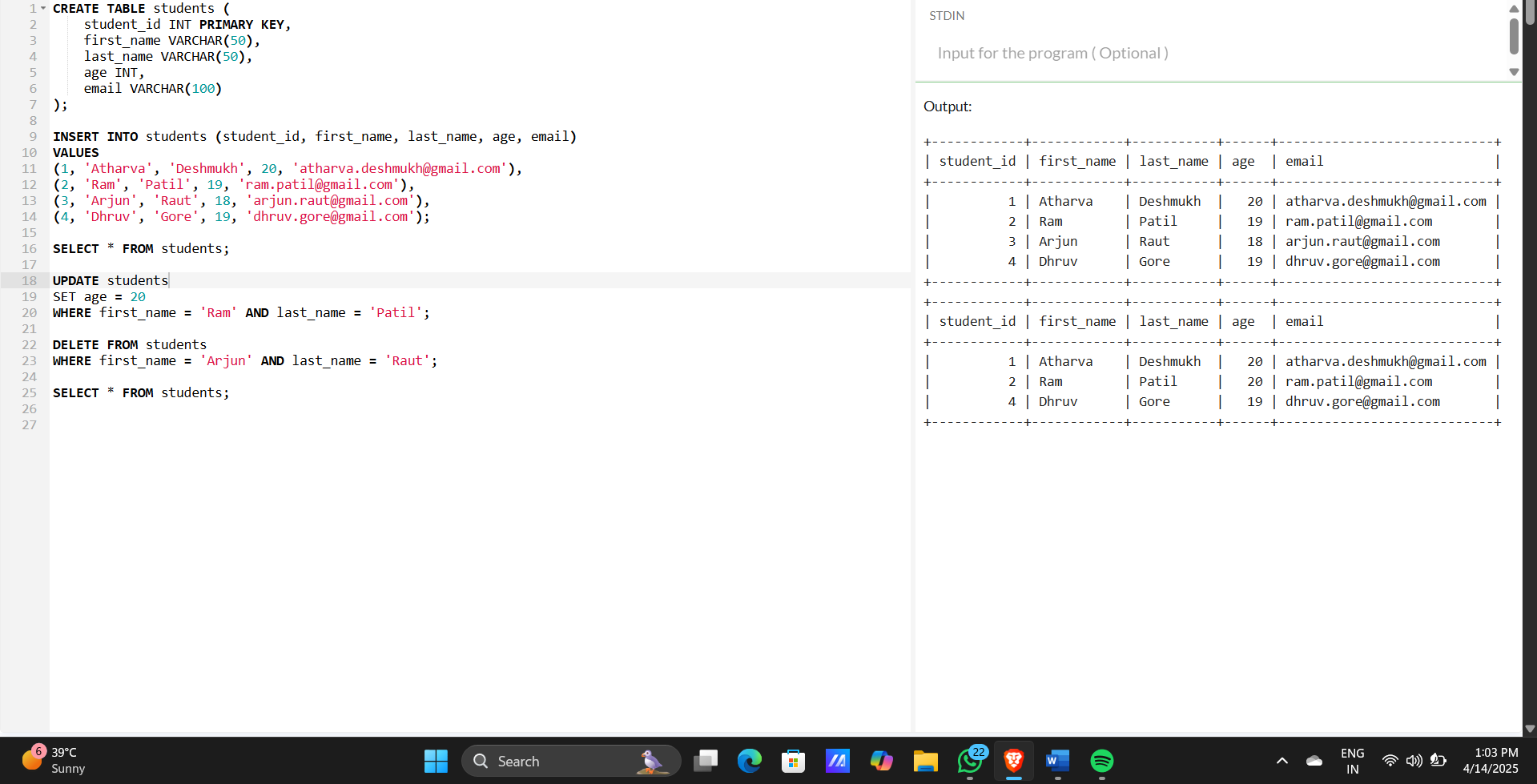
SELECT \* FROM students;

**IMAGE**:

Basic students table



Students table after the query is performed



**PRACTICAL NO 3**

**AIM**: Create a table with columns for Employee lD, Name, Salary, Joining Date, and ActiveStatus using different data types. Insert sample data and perform queries to manipulate and retrieve data.

**CODE**:

CREATE TABLE employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

Salary DECIMAL(10, 2),

JoiningDate DATE,

ActiveStatus BOOLEAN

);

INSERT INTO employees (EmployeeID, Name, Salary, JoiningDate, ActiveStatus)

VALUES

(100, 'Atharva Deshmukh', 75000.00, '2022-03-01', TRUE),

(101, 'Sneha Kapoor', 58000.50, '2021-07-12', TRUE),

(102, 'Karan Patel', 67000.00, '2019-11-23', FALSE),

(103, 'Riya Sharma', 61000.75, '2020-01-15', TRUE);

SELECT \* FROM employees;

UPDATE employees

SET Salary = 69000.00

WHERE Name = 'Karan Patel';

UPDATE employees

SET ActiveStatus = FALSE

WHERE Name = 'Riya Sharma';

DELETE FROM employees

WHERE EmployeeID = 101;

SELECT \* FROM employees;

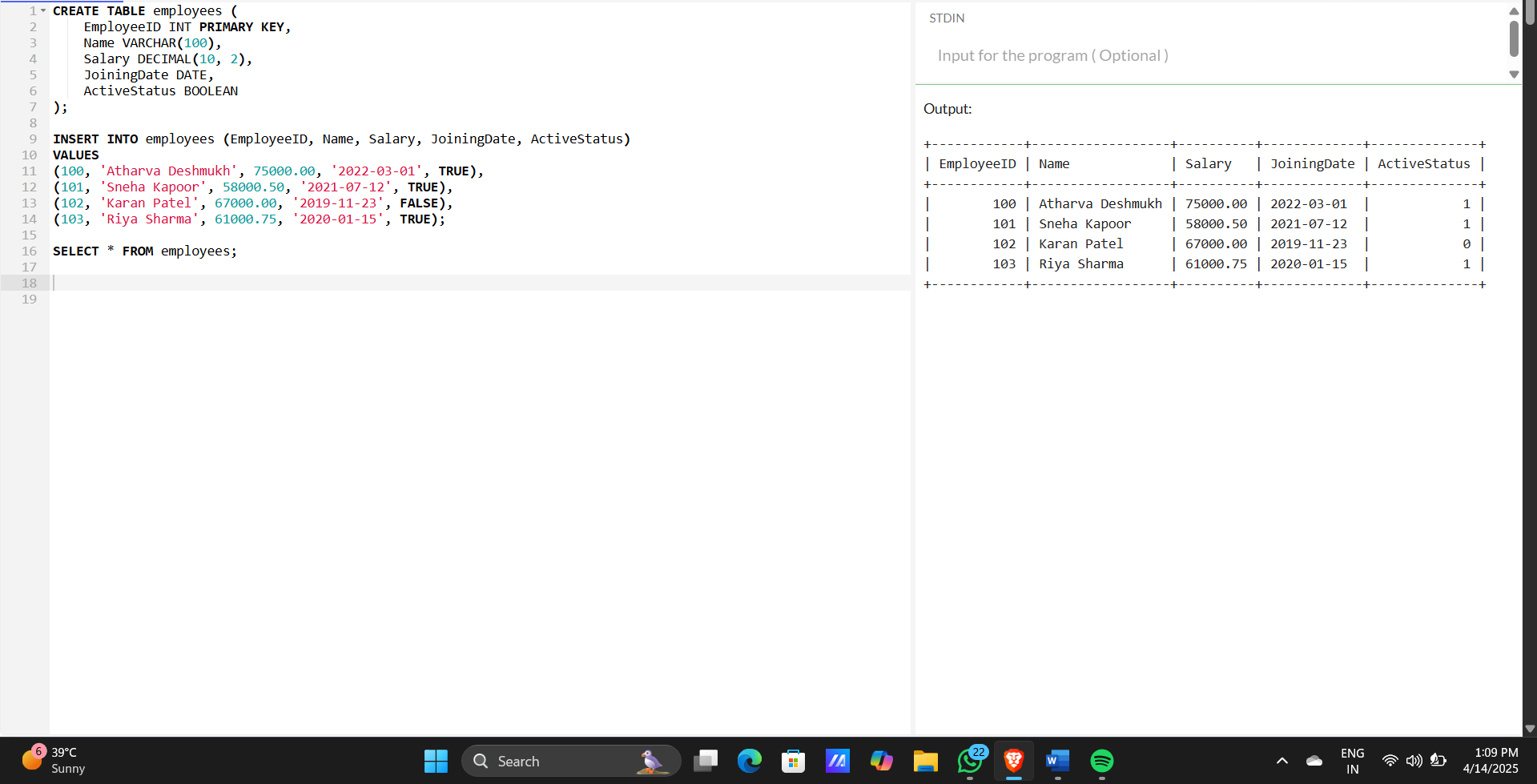
SELECT \* FROM employees

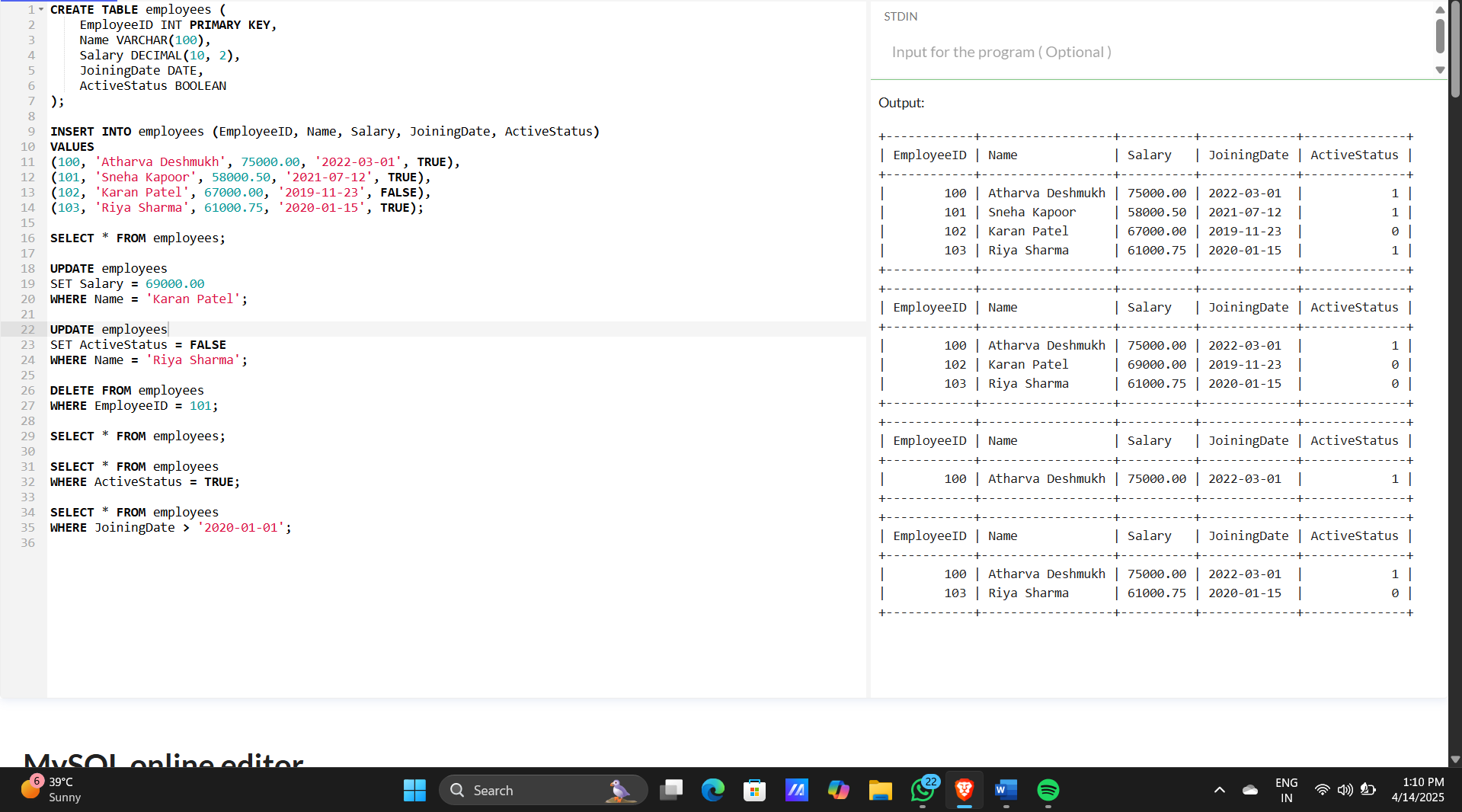
WHERE ActiveStatus = TRUE;

SELECT \* FROM employees

WHERE JoiningDate > '2020-01-01';

**IMAGE**:





**PRACTICAL NO 4**

Creating Employee Table with Constraints

**AIM**: Create a table to store employee information with constraints like Primary Key, Foreign Key, and Unique. Insert valid and invalid data to test the constraints.

**CODE**:

CREATE TABLE Department (

DeptID INT PRIMARY KEY,

DeptName VARCHAR(50) UNIQUE

);

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE,

Salary DECIMAL(10,2) CHECK (Salary > 0),

DeptID INT,

FOREIGN KEY (DeptID) REFERENCES Department(DeptID)

);

INSERT INTO Department (DeptID, DeptName)

VALUES

(1, 'HR'),

(2, 'IT');

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)

VALUES

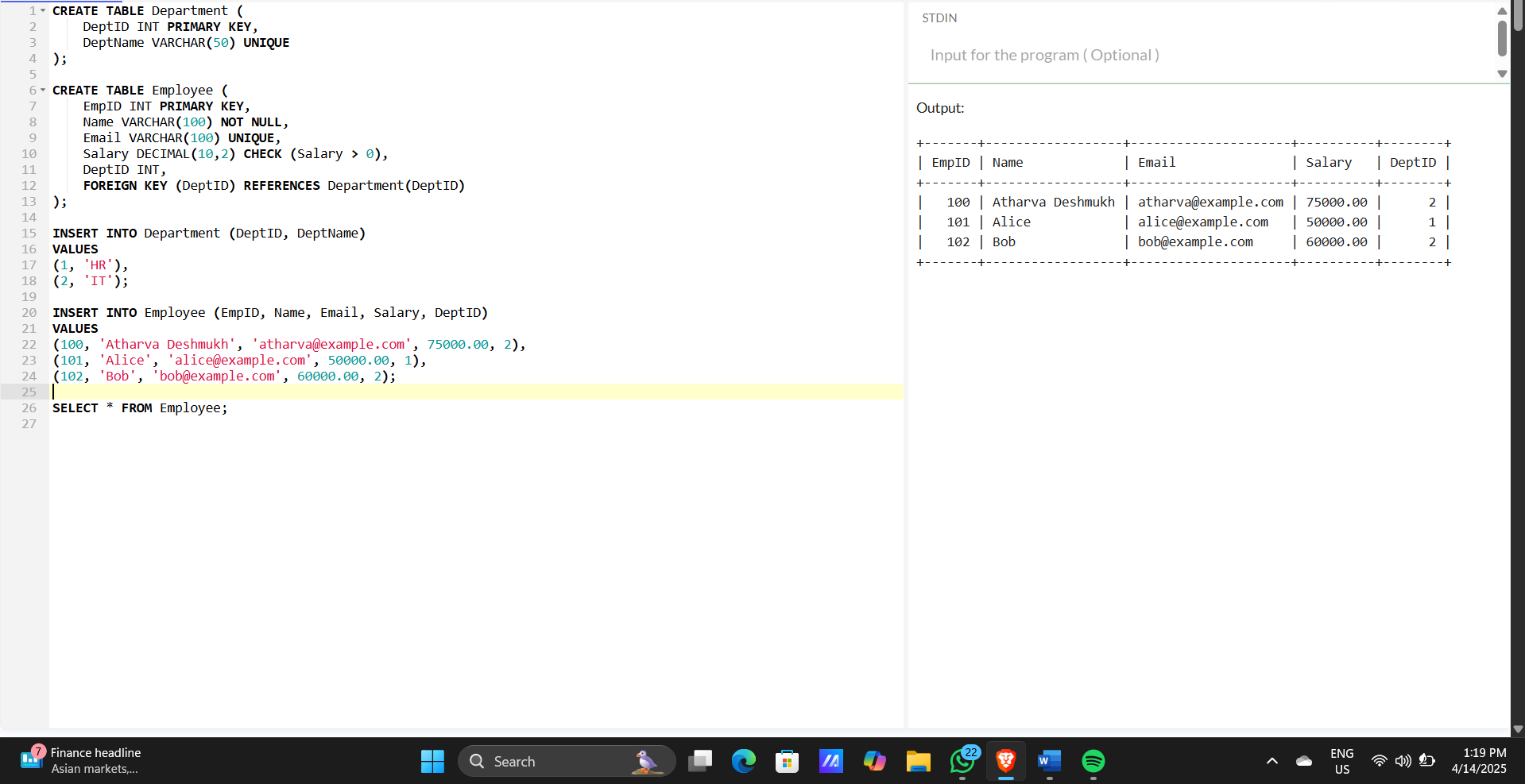
(100, 'Atharva Deshmukh', 'atharva@example.com', 75000.00, 2),

(101, 'Alice', 'alice@example.com', 50000.00, 1),

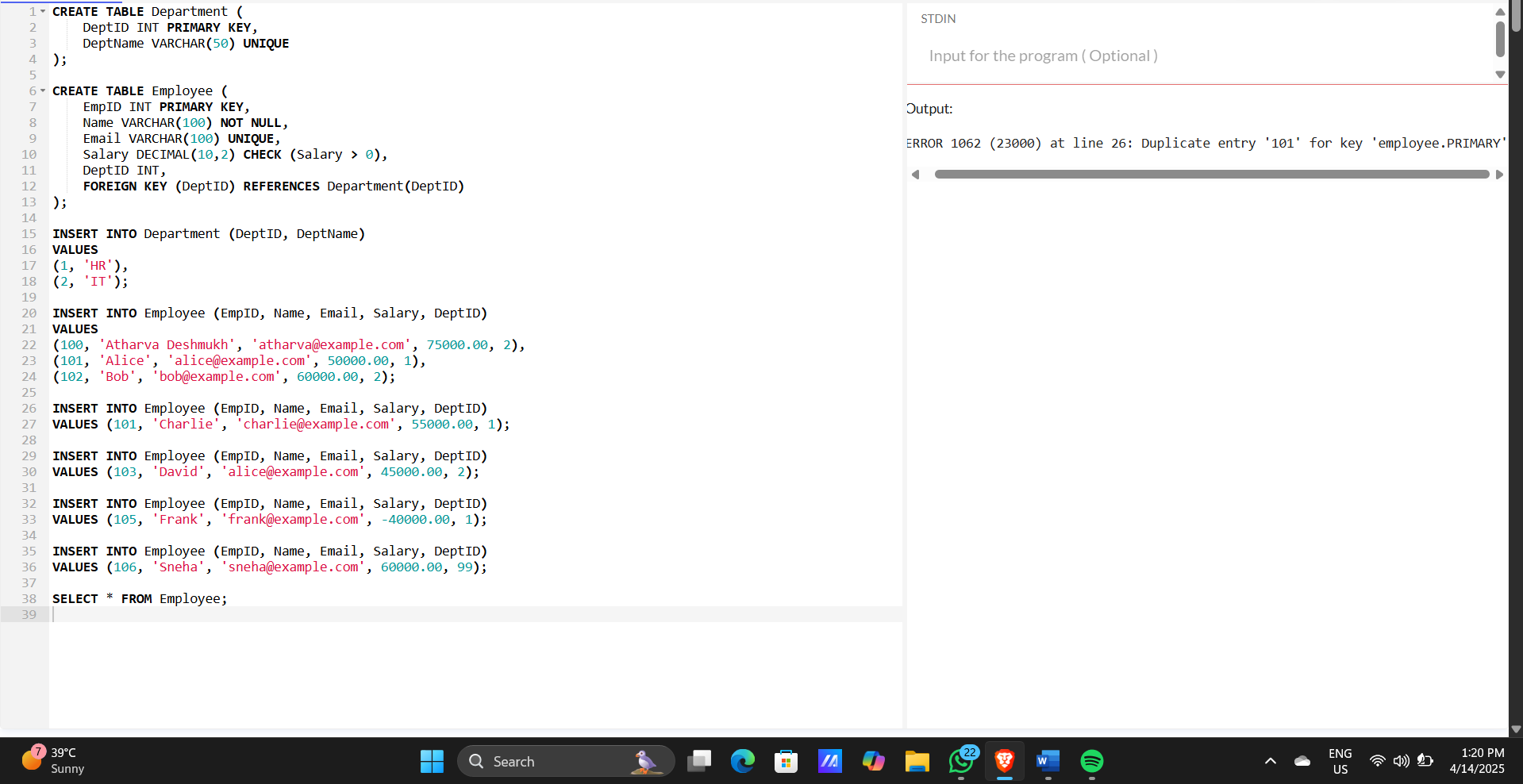
(102, 'Bob', 'bob@example.com', 60000.00, 2);

SELECT \* FROM Employee;

**IMAGE:**

****

2. Now code to check the Constraints like Primary, Foreign and Unique Key is working properly or not

****

CREATE TABLE Department (

DeptID INT PRIMARY KEY,

DeptName VARCHAR(50) UNIQUE

);

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE,

Salary DECIMAL(10,2) CHECK (Salary > 0),

DeptID INT,

FOREIGN KEY (DeptID) REFERENCES Department(DeptID)

);

INSERT INTO Department (DeptID, DeptName)

VALUES

(1, 'HR'),

(2, 'IT');

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)

VALUES

(100, 'Atharva Deshmukh', 'atharva@example.com', 75000.00, 2),

(101, 'Alice', 'alice@example.com', 50000.00, 1),

(102, 'Bob', 'bob@example.com', 60000.00, 2);

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)

VALUES (101, 'Charlie', 'charlie@example.com', 55000.00, 1);

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)

VALUES (103, 'David', 'alice@example.com', 45000.00, 2);

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)

VALUES (105, 'Frank', 'frank@example.com', -40000.00, 1);

INSERT INTO Employee (EmpID, Name, Email, Salary, DeptID)

VALUES (106, 'Sneha', 'sneha@example.com', 60000.00, 99);

SELECT \* FROM Employee;

**PRACTICAL NO 5**

Testing Employee Constraints

**AIM**: To test constraints like PRIMARY KEY, UNIQUE, and CHECK by inserting invalid data into the Employee table.

**CODE:**

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(100) NOT NULL,

LastName VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

Age INT CHECK (Age >= 18),

IsActive BOOLEAN DEFAULT TRUE

);

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age, IsActive)

VALUES

(1, 'Atharva', 'Deshmukh', 'john.doe@example.com', '1234567890', 25, TRUE),

(2, 'John', 'Smith', 'jane.smith@example.com', '0987654321', 30);

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age)

VALUES

(3, 'Emma', 'Taylor', 'taylor@example.com', '5551234567', 20);

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age)

VALUES

(4, 'Alice', 'Johnson', 'alice.johnson@example.com', '6669876543', 16);

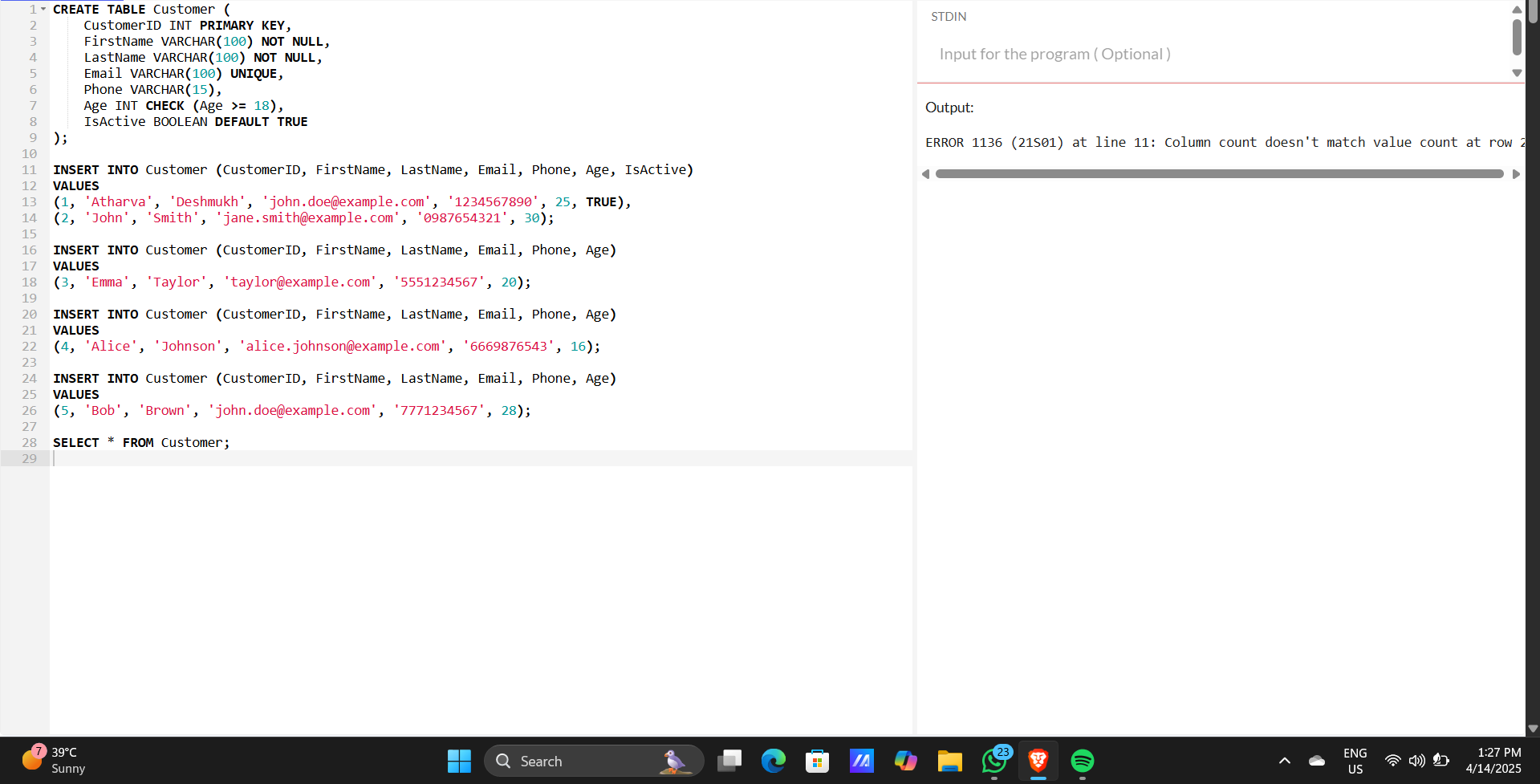
INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone, Age)

VALUES

(5, 'Bob', 'Brown', 'john.doe@example.com', '7771234567', 28);

SELECT \* FROM Customer;

**IMAGE**:



**Conclusion**: The Constraints are working properly because when I tried to enter the null value for NOT NULL constraints or enter duplicate value at UNIQUE constraint it throws error

**PRACTICAL NO 6**

Using DDL commands

**AIM**: Use DDL commands to create tables and DML commands to insert, update, and delete data. Write SELECT queries to retrieve and verify data changes.

**CODE**:

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Age INT,

Department VARCHAR(50),

Salary DECIMAL(10, 2)

);

INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Department, Salary)

VALUES

(1, 'John', 'Doe', 28, 'HR', 50000.00),

(2, 'Jane', 'Smith', 35, 'IT', 65000.00),

(3, 'Michael', 'Johnson', 40, 'Finance', 75000.00),

(4, 'Atharva', 'Deshmukh', 25, 'Marketing', 45000.00),

(5, 'Emma', 'Taylor', 30, 'IT', 60000.00),

(6, 'Oliver', 'Williams', 38, 'HR', 55000.00);

UPDATE Employees

SET Salary = 70000.00

WHERE EmployeeID = 2;

UPDATE Employees

SET FirstName = 'Janet', LastName = 'Williams', Salary = 75000.00

WHERE EmployeeID = 2;

UPDATE Employees

SET FirstName = 'Michael', LastName = 'Brown', Age = 45, Department = 'Management', Salary = 80000.00

WHERE EmployeeID = 3;

DELETE FROM Employees

WHERE EmployeeID = 1;

SELECT \* FROM Employees;

SELECT FirstName, LastName, Salary

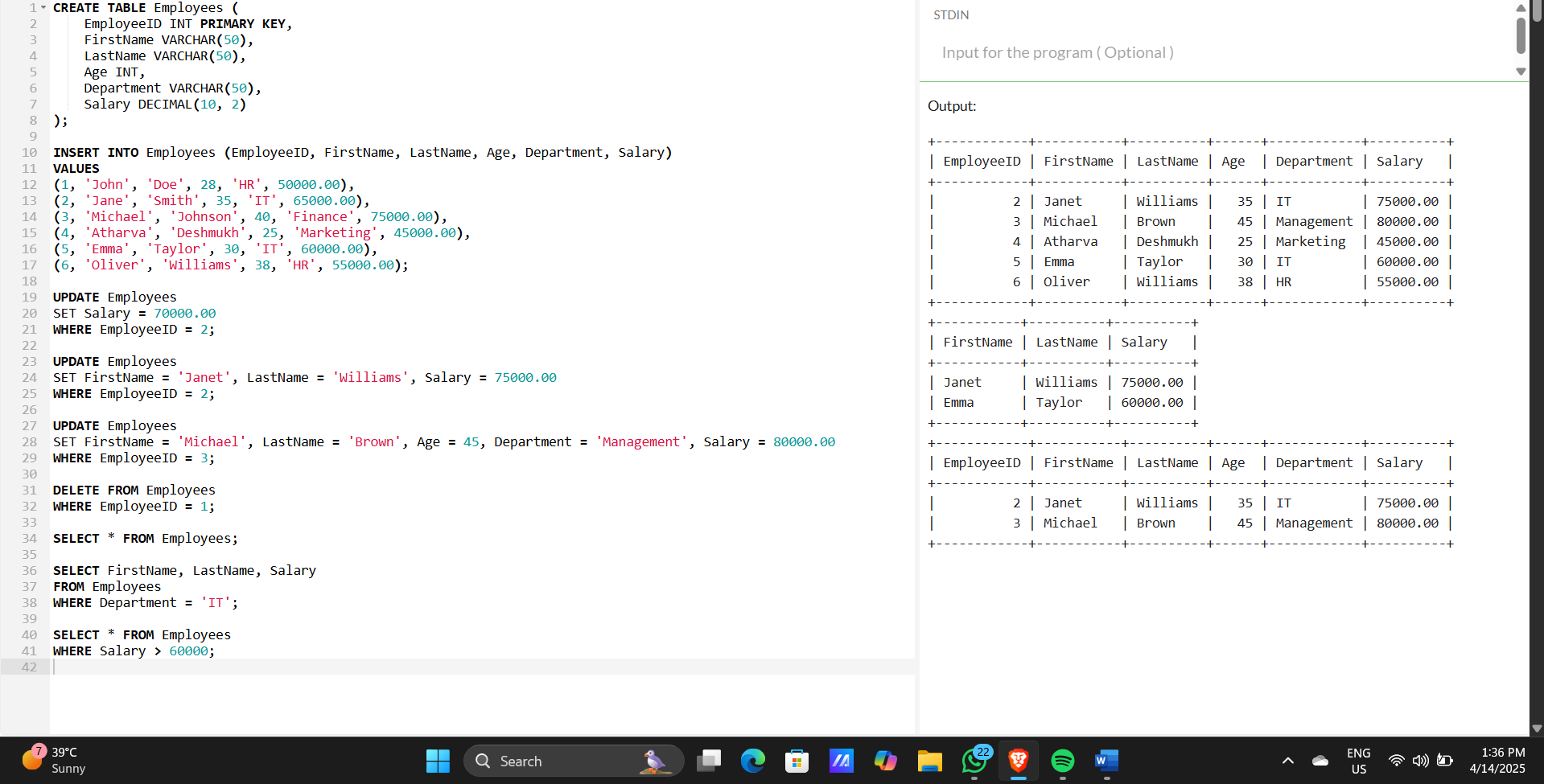
FROM Employees

WHERE Department = 'IT';

SELECT \* FROM Employees

WHERE Salary > 60000;

**IMAGE:**

****

**PRACTICAL NO 7**

**AIM:** Create a Sales table and use aggregate functions like COUNT, SUM, AVG, MIN, and MAX to summarize sales data and calculate statistics.

**CODE:**

CREATE TABLE Sales (

SaleID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Product VARCHAR(50),

Quantity INT,

Price DECIMAL(10,2),

SaleDate DATE

);

INSERT INTO Sales (FirstName, LastName, Product, Quantity, Price, SaleDate) VALUES

('Atharva', 'Deshmukh', 'Laptop', 2, 75000.00, '2025-02-01'),

('Riya', 'Shah', 'Mobile', 5, 20000.00, '2025-02-02'),

('Ankit', 'Verma', 'Tablet', 3, 30000.00, '2025-02-03'),

('Sneha', 'Kulkarni', 'Laptop', 1, 78000.00, '2025-02-04'),

('Karan', 'Yadav', 'Mobile', 4, 22000.00, '2025-02-05'),

('Neha', 'Rane', 'Tablet', 2, 32000.00, '2025-02-06');

SELECT COUNT(\*) AS Total\_Sales FROM Sales;

SELECT SUM(Quantity \* Price) AS Total\_Revenue FROM Sales;

SELECT AVG(Price) AS Average\_Price FROM Sales;

SELECT MIN(Price) AS Min\_Price, MAX(Price) AS Max\_Price FROM Sales;

SELECT COUNT(\*) AS Total\_Sales FROM Sales;

SELECT COUNT(DISTINCT Product) AS Unique\_Products FROM Sales;

SELECT Product, COUNT(\*) AS Sales\_Count

FROM Sales

GROUP BY Product;

SELECT SaleDate, COUNT(\*) AS Sales\_Per\_Day

FROM Sales

GROUP BY SaleDate;

SELECT COUNT(\*) AS High\_Quantity\_Sales

FROM Sales

WHERE Quantity > 2;

SELECT COUNT(\*) AS Sales\_This\_Month

FROM Sales

WHERE MONTH(SaleDate) = MONTH(CURRENT\_DATE)

AND YEAR(SaleDate) = YEAR(CURRENT\_DATE);

SELECT COUNT(\*) AS High\_Value\_Sales

FROM Sales

WHERE (Quantity \* Price) > 50000;

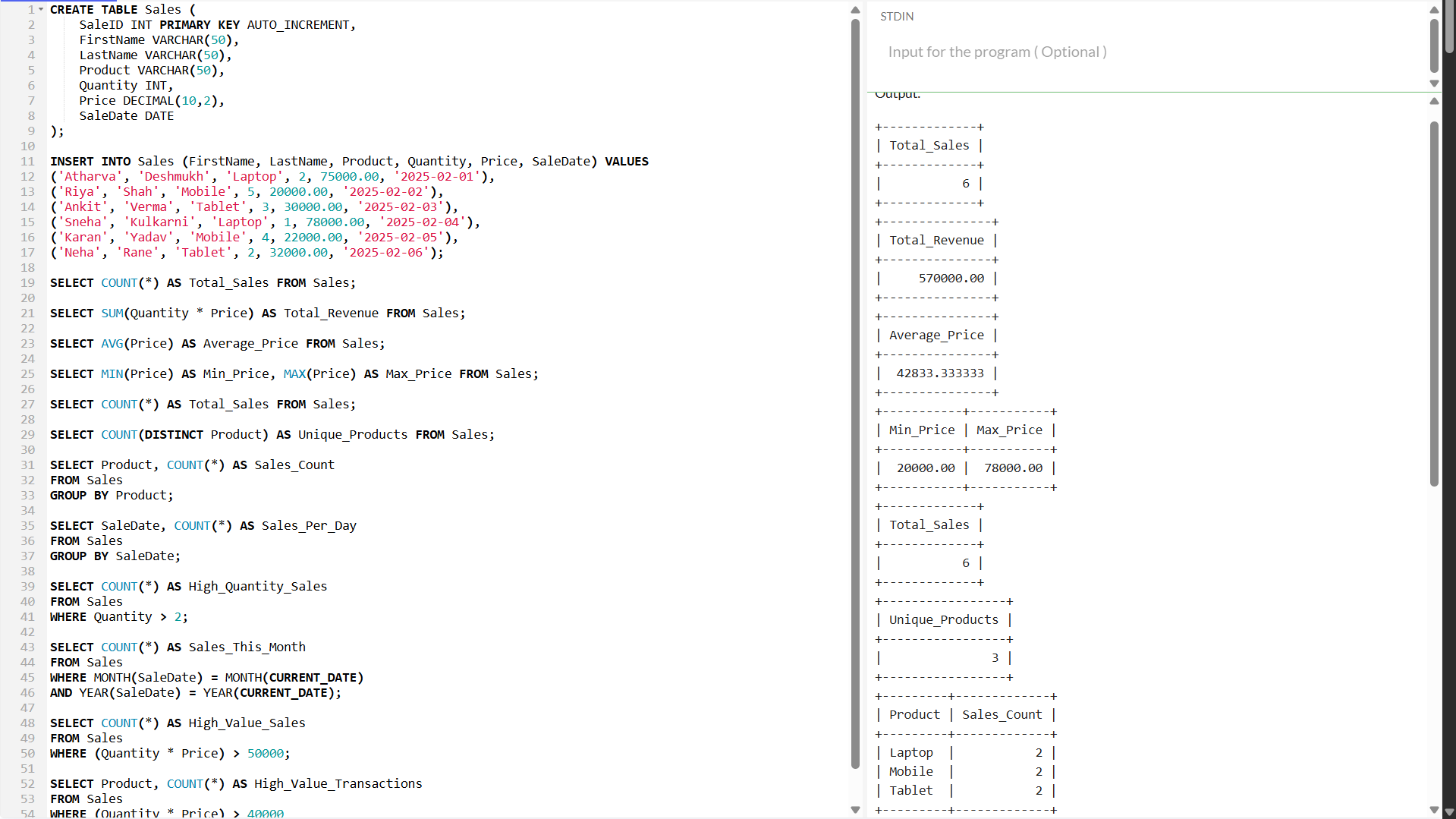
SELECT Product, COUNT(\*) AS High\_Value\_Transactions

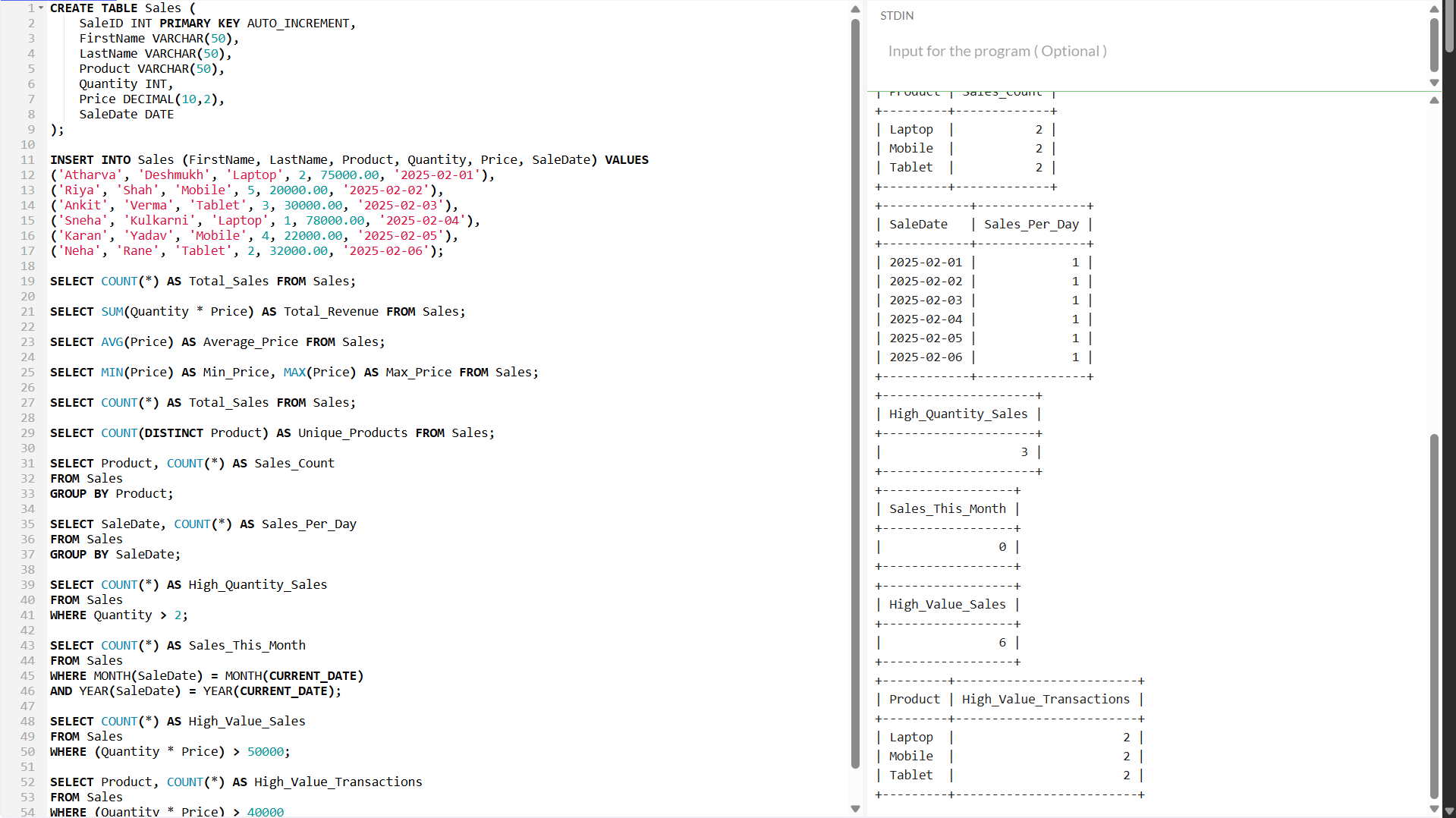
FROM Sales

WHERE (Quantity \* Price) > 40000

GROUP BY Product;

**IMAGE:**

****

****

**PRACTICAL NO 8**

**AIM:** Given Customers and Orders tables, write SQL queries to perform INNER JOIN, LEFT JOIN, and RIGHT JOIN to retrieve combined data for customer orders.

**CODE:**

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(100) NOT NULL

);

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

order\_date DATE NOT NULL,

customer\_id INT,

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

);

INSERT INTO Customers (customer\_id, customer\_name) VALUES

(1, 'Atharva Deshmukh'),

(2, 'Priya Shah'),

(3, 'Rohan Mehta'),

(4, 'Sneha Reddy');

INSERT INTO Orders (order\_id, order\_date, customer\_id) VALUES

(101, '2024-01-01', 1),

(102, '2024-01-02', 2),

(103, '2024-01-03', 4);

SELECT \* FROM Customers;

SELECT \* FROM Orders;

SELECT

c.customer\_id,

c.customer\_name,

o.order\_id,

o.order\_date

FROM

Customers c

INNER JOIN

Orders o ON c.customer\_id = o.customer\_id;

SELECT

c.customer\_id,

c.customer\_name,

o.order\_id,

o.order\_date

FROM

Customers c

LEFT JOIN

Orders o ON c.customer\_id = o.customer\_id;

SELECT

c.customer\_id,

c.customer\_name,

o.order\_id,

o.order\_date

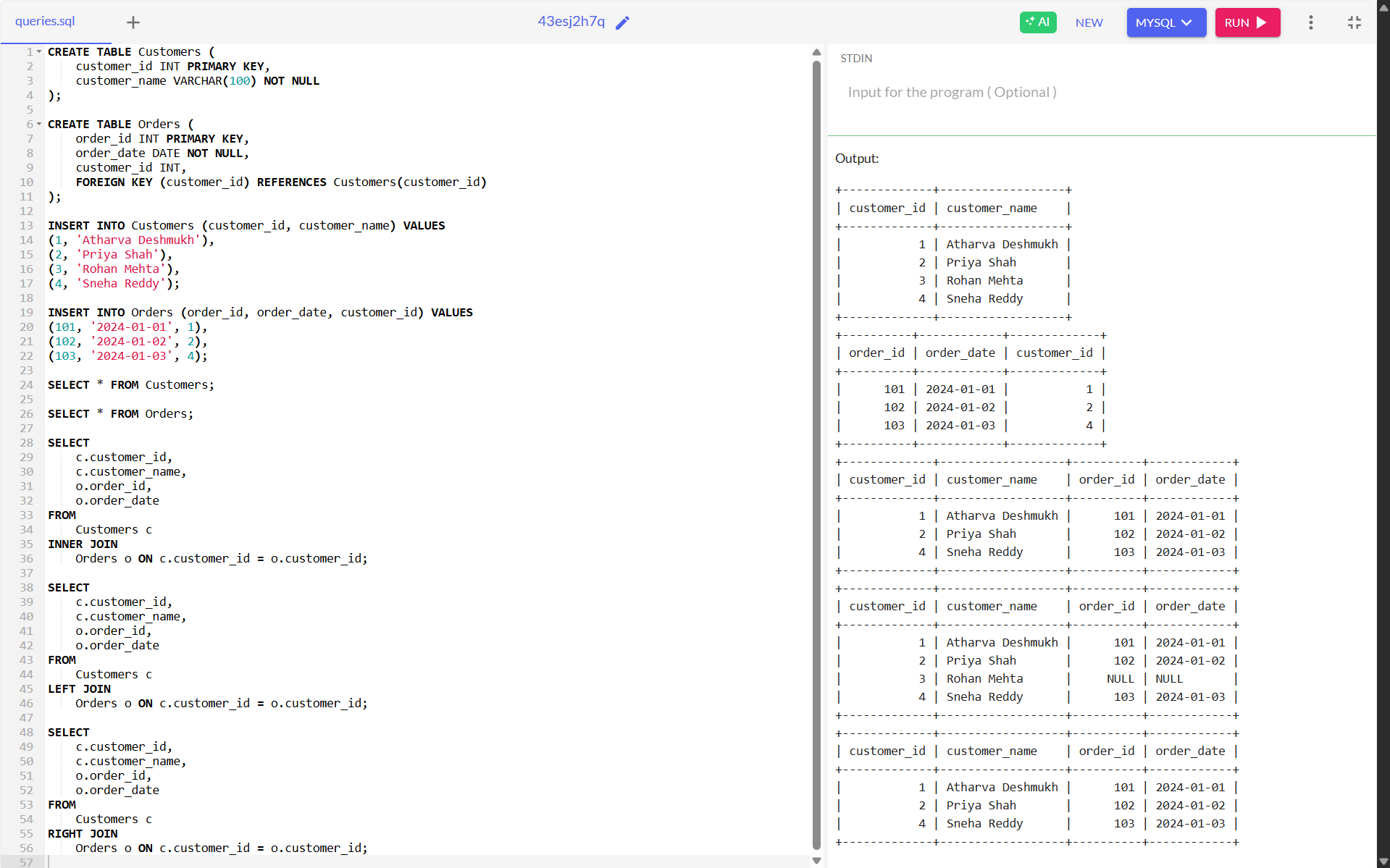
FROM

Customers c

RIGHT JOIN

Orders o ON c.customer\_id = o.customer\_id;

**IMAGE:**

****