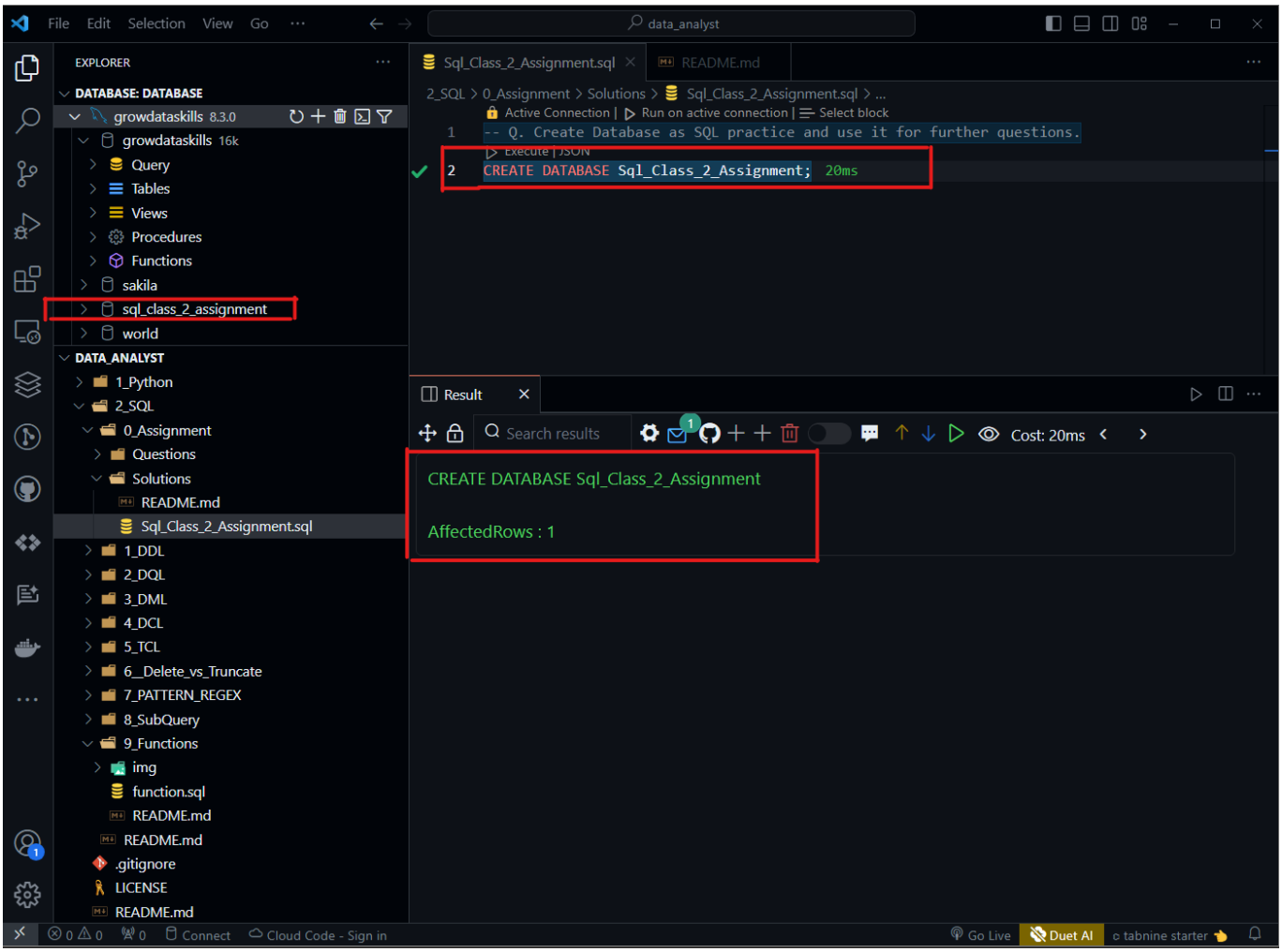
Q1. Create Database as SQL practice and use it for further questions.

- Creating a database name `Sql_Class_2_Assignment` and will use it for further questions.

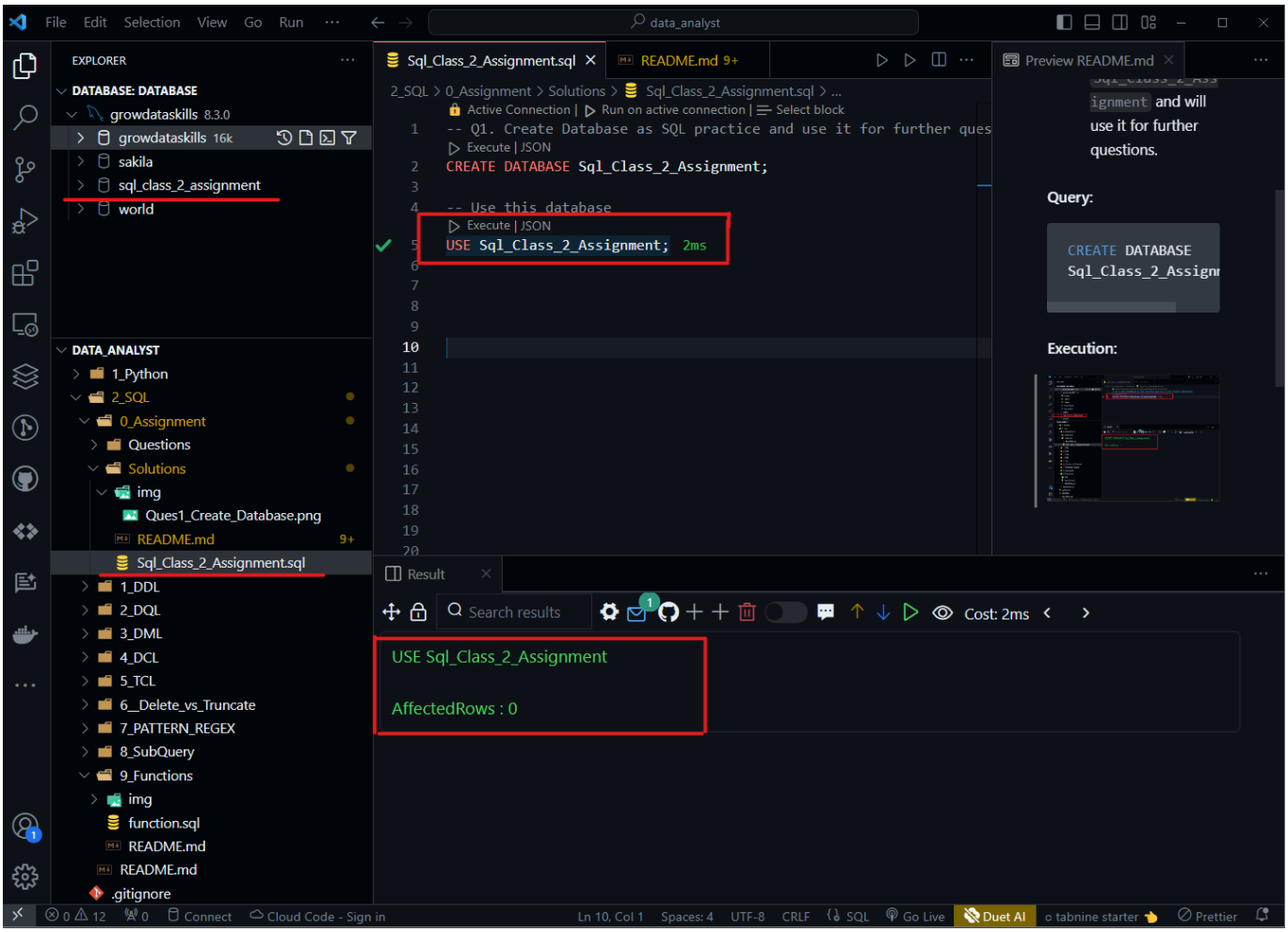
Query:

```
CREATE DATABASE Sql_Class_2_Assignment;
```

Execution:




```
USE Sql_Class_2_Assignment;
```



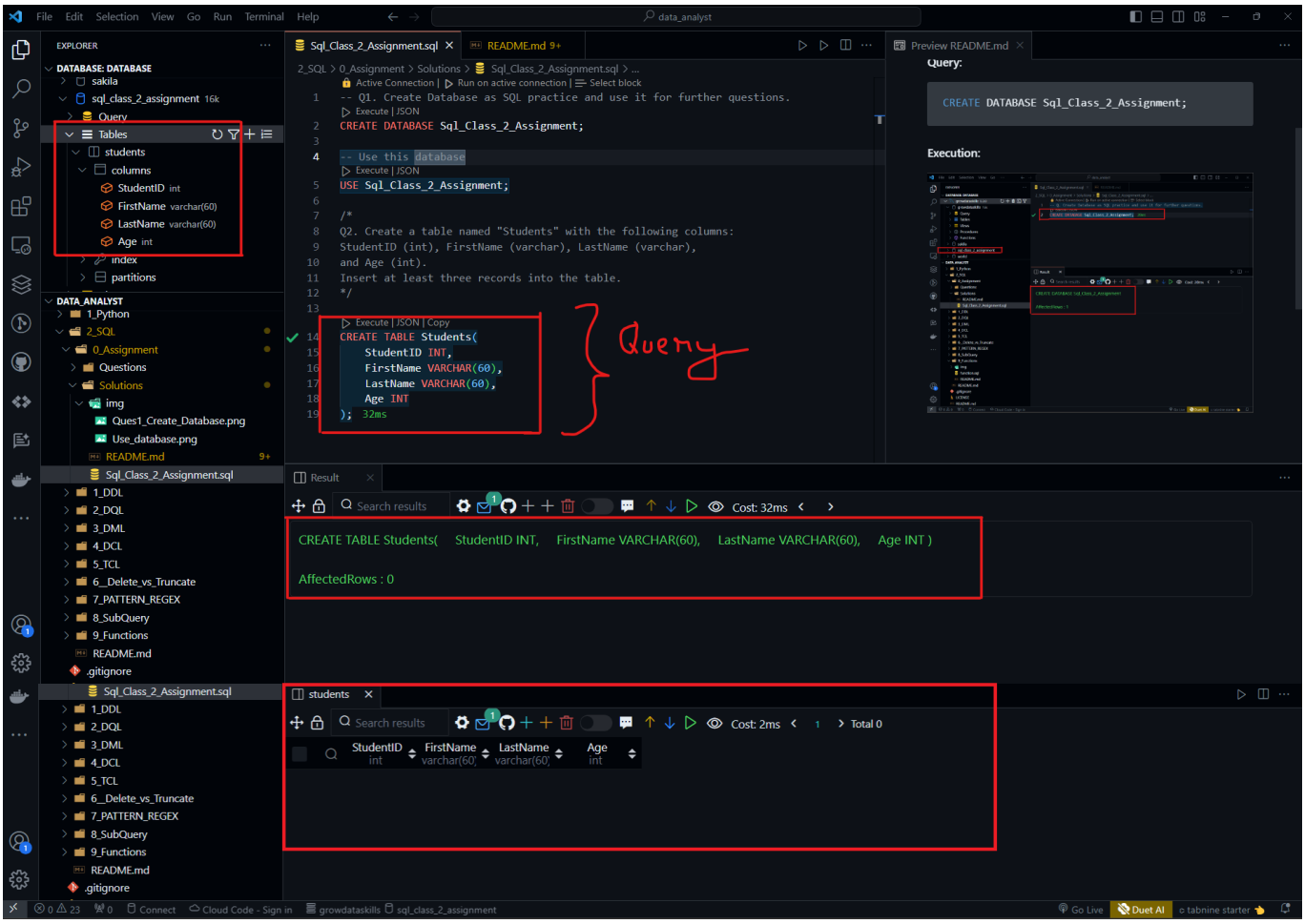
Q2. Create a table named "Students" with the following columns: StudentID (int), FirstName (varchar), LastName (varchar), and Age (int). Insert at least three records into the table.

- Creating a table named `Students` with the following columns: StudentID (int), FirstName (varchar), LastName (varchar), and Age (int)

Query:

```
CREATE TABLE Students(
    StudentID INT,
    FirstName VARCHAR(60),
    LastName VARCHAR(60),
    Age INT
);
```

Execution:

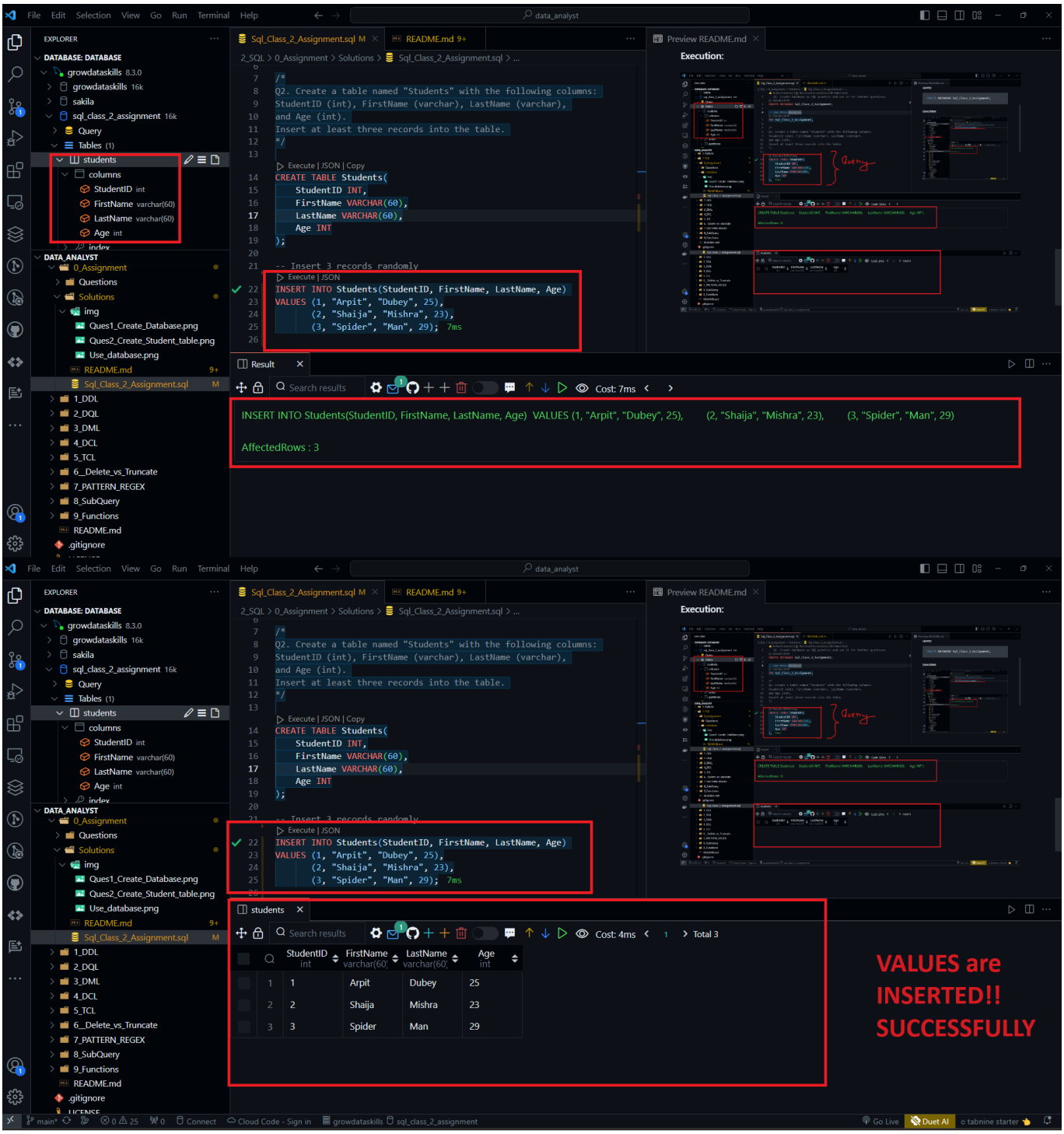


- Inserting 3 random records into the table `Students`

Query:

```
INSERT INTO Students(StudentID, FirstName, LastName, Age)
VALUES (1, "Arpit", "Dubey", 25),
       (2, "Shaija", "Mishra", 23),
       (3, "Spider", "Man", 29);
```

Execution:



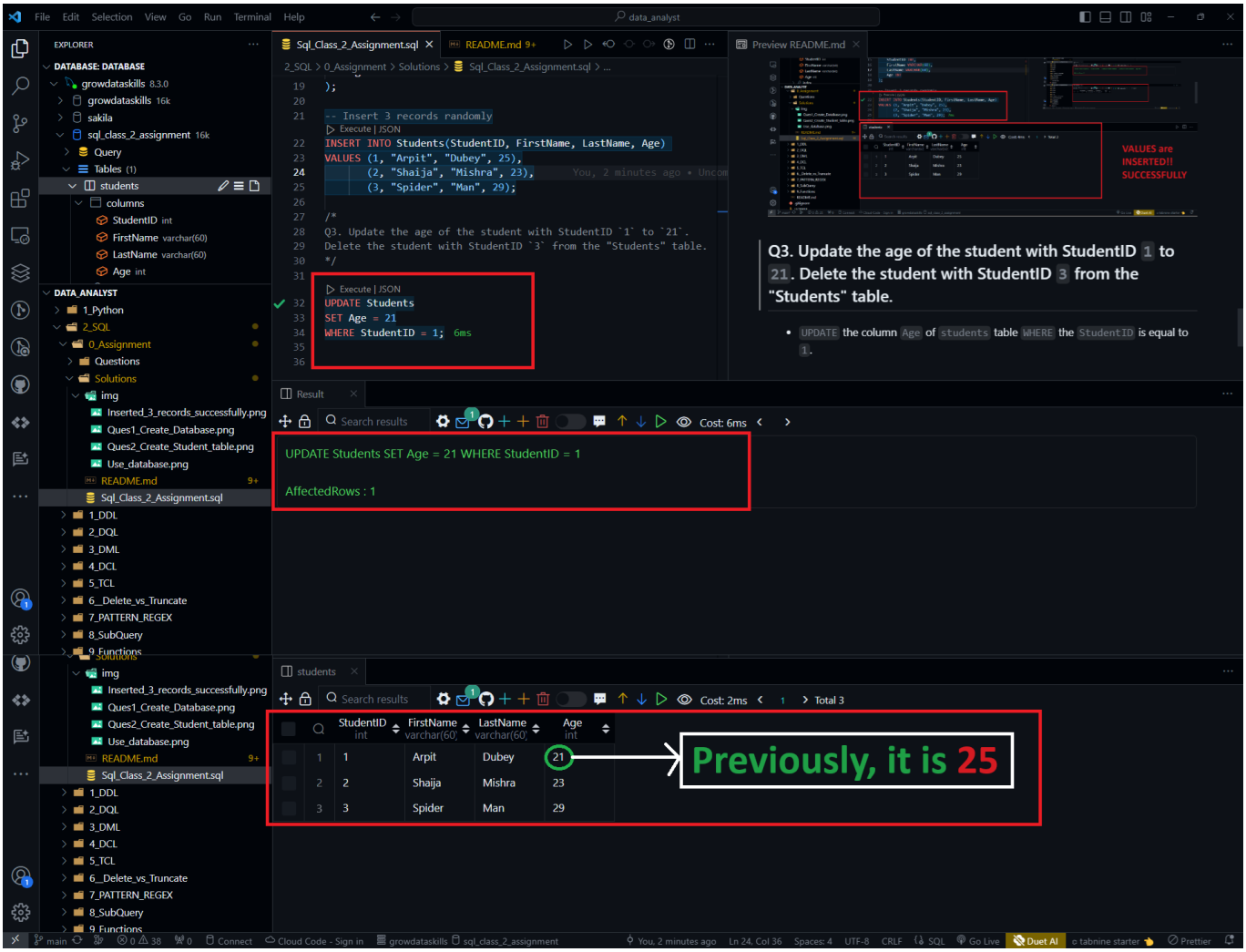
Q3. Update the age of the student with StudentID 1 to 21. Delete the student with StudentID 3 from the "Students" table.

- UPDATE the column Age of students table WHERE the StudentID is equal to 1.

Query:

```
UPDATE Students
SET Age = 21
WHERE StudentID = 1;
```

Execution:

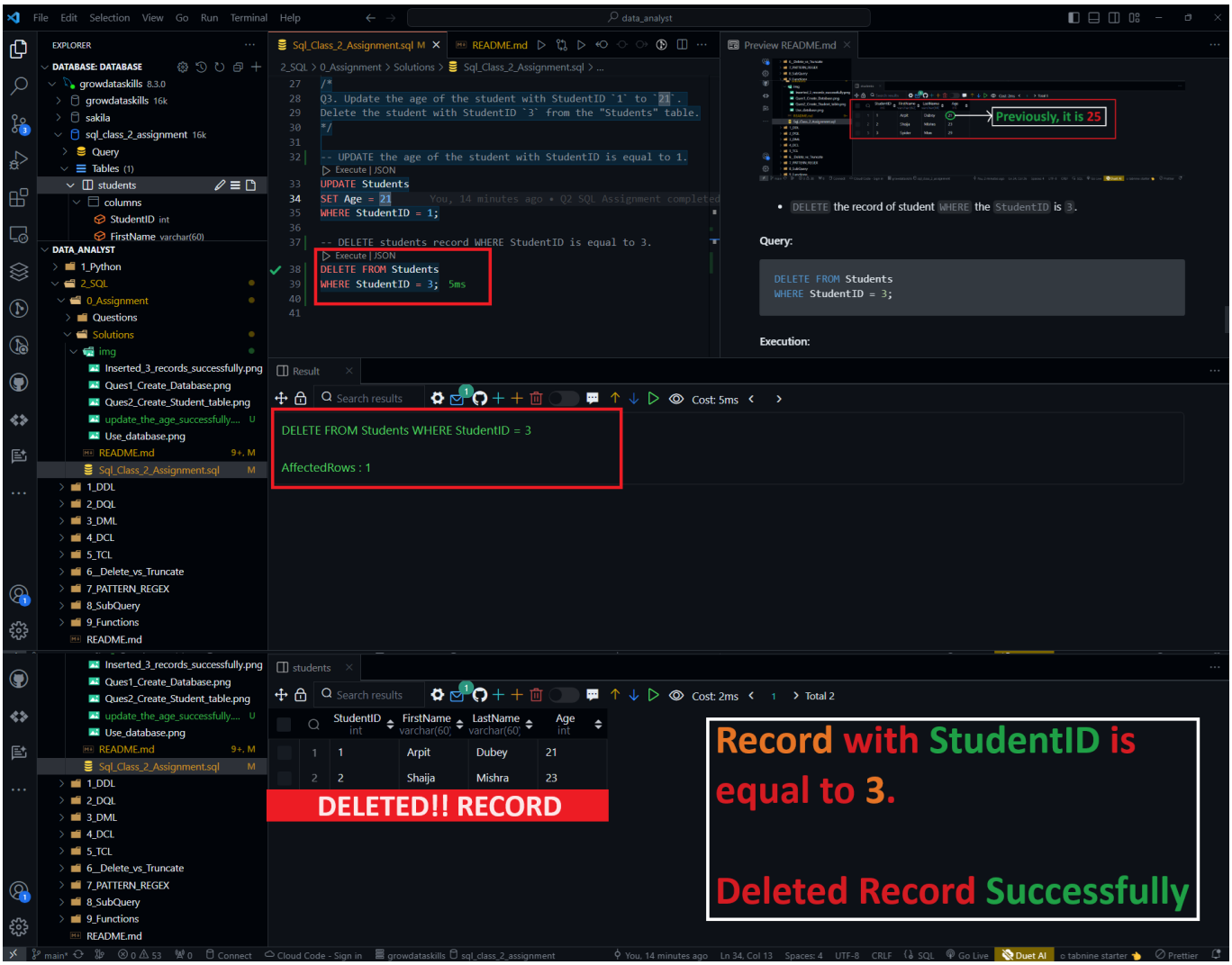


- `DELETE` the record of student `WHERE` the `StudentID` is `3`.

Query:

```
DELETE FROM Students
WHERE StudentID = 3;
```

Execution:



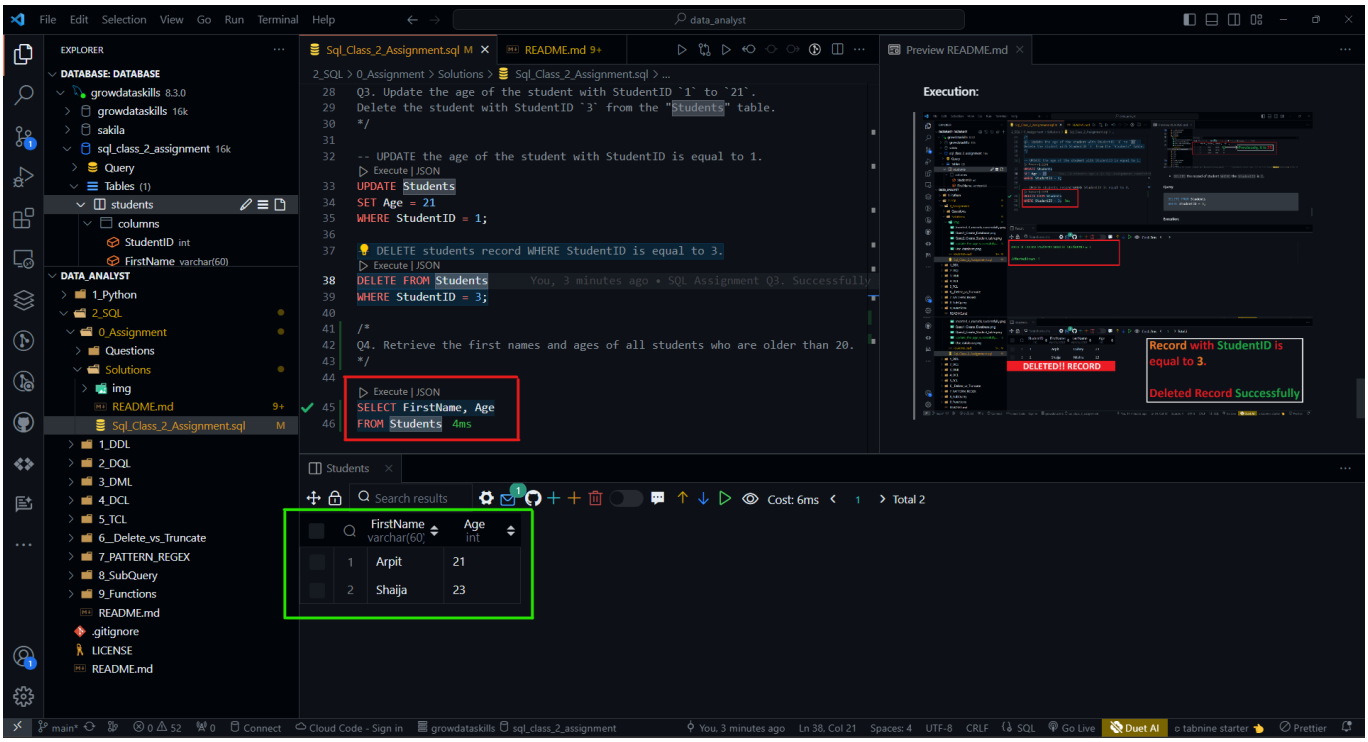
Q4. Retrieve the first names and ages of all students who are older than 20.

- Retrieving the `FirstName` and `Age` of all students without any condition.

Query:

```
SELECT FirstName, Age
FROM Students
```

Execution:

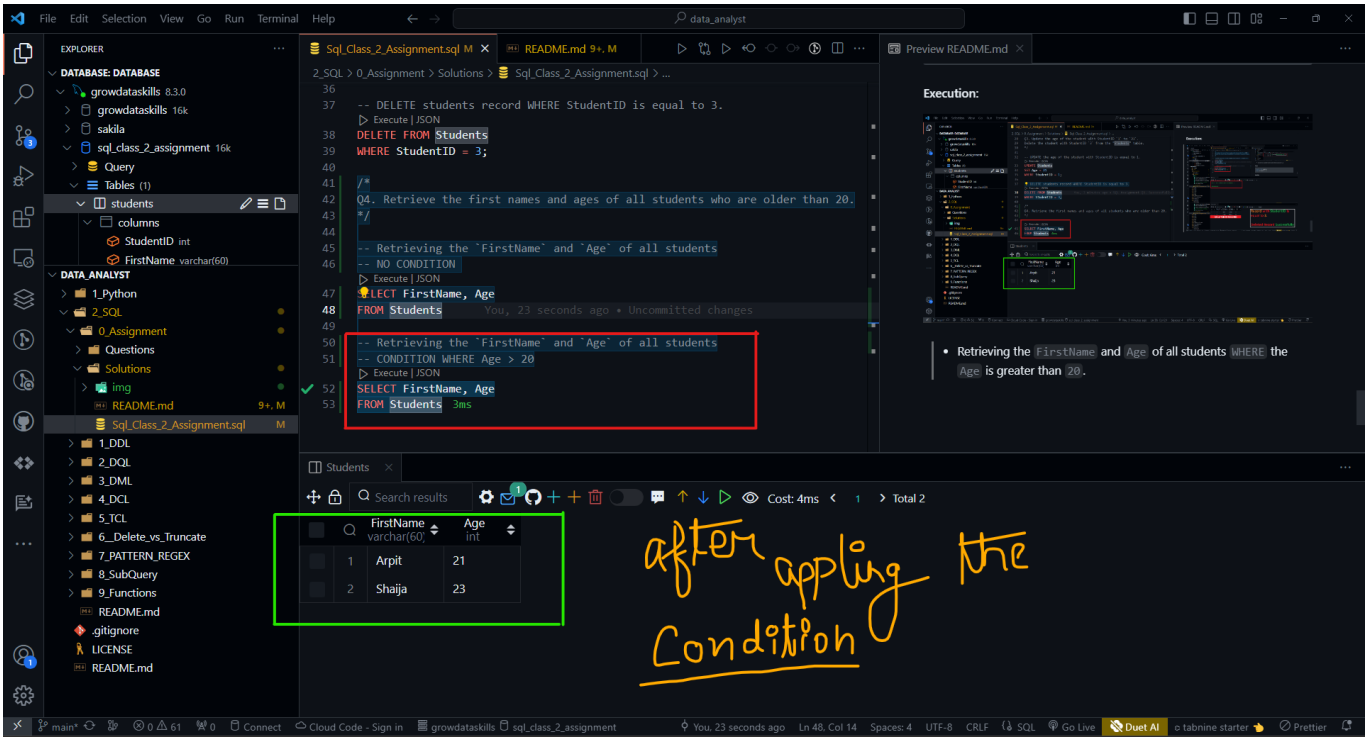


- Retrieving the `FirstName` and `Age` of all students `WHERE` the `Age` is greater than `20`..

Query:

```
SELECT FirstName, Age
FROM Students
WHERE Age > 20;
```

Execution:

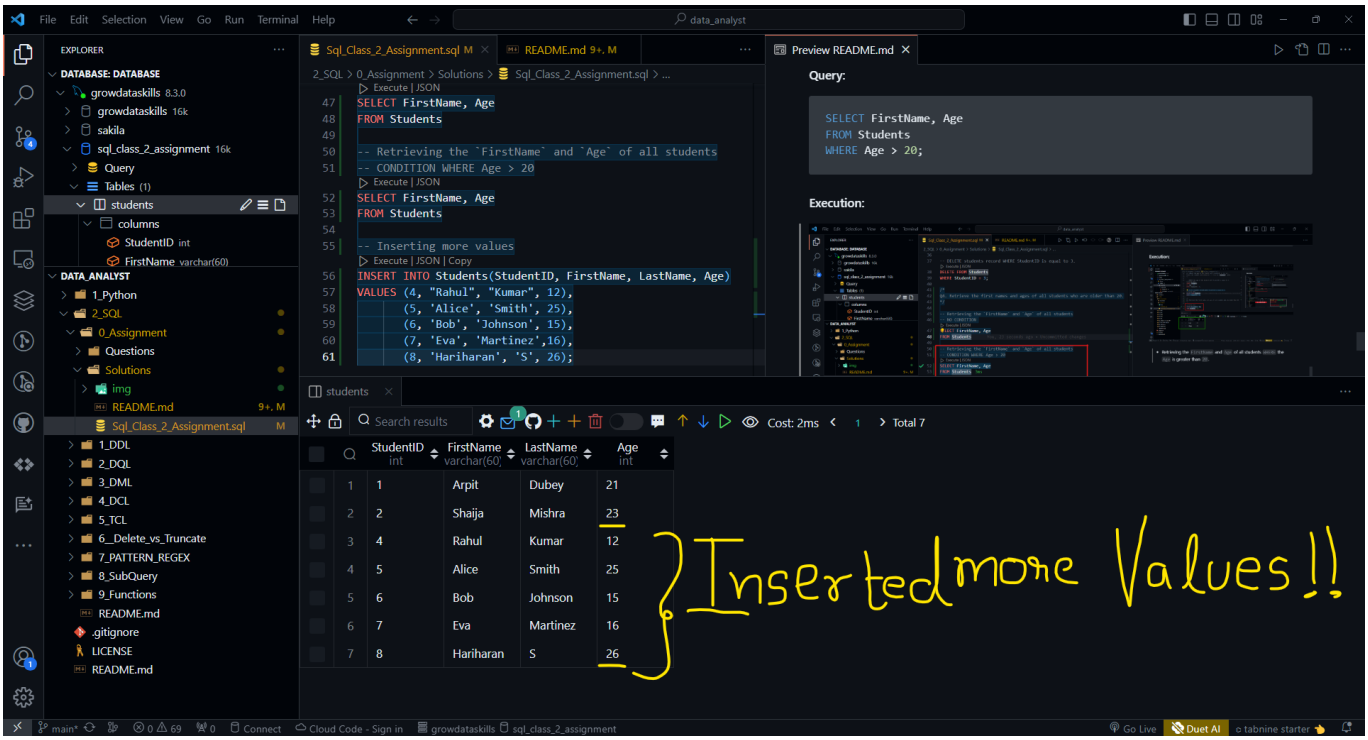


NOTE : To Perform more operation we required more values so I just inserted more values to the table.

Query:

```
INSERT INTO Students(StudentID, FirstName, LastName, Age)
VALUES (4, "Rahul", "Kumar", 12),
       (5, 'Alice', 'Smith', 25),
       (6, 'Bob', 'Johnson', 15),
       (7, 'Eva', 'Martinez',16),
       (8, 'Hariharan', 'S', 26);
```

Execution:



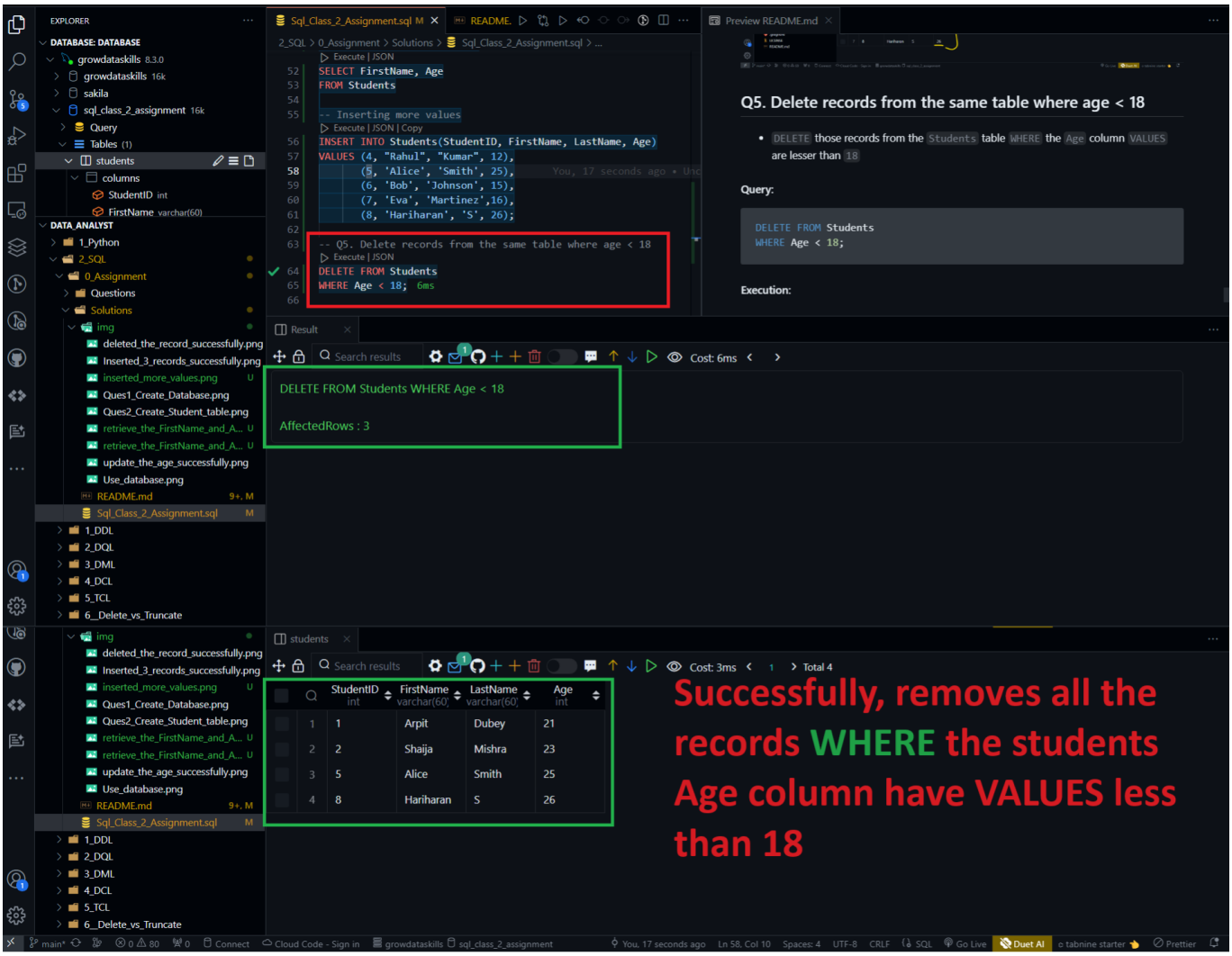
Q5. Delete records from the same table where age < 18

- `DELETE` those records from the `Students` table `WHERE` the `Age` column `VALUES` are lesser than `18`

Query:

```
DELETE FROM Students
WHERE Age < 18;
```

Execution:



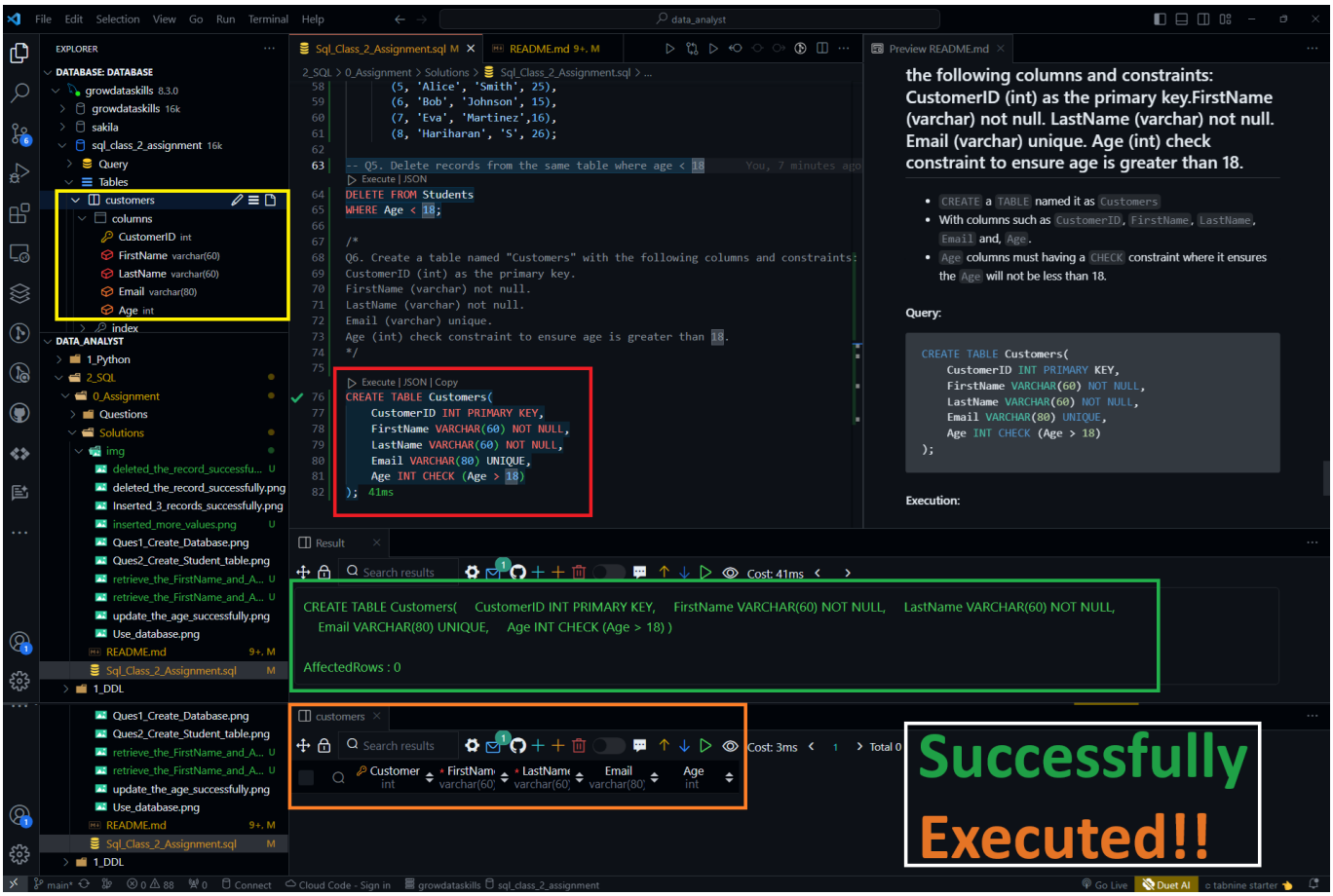
Q6. Create a table named "Customers" with the following columns and constraints: CustomerID (int) as the primary key.FirstName (varchar) not null. LastName (varchar) not null. Email (varchar) unique. Age (int) check constraint to ensure age is greater than 18.

- `CREATE` a `TABLE` named it as `Customers`
- With columns such as `CustomerID`, `FirstName`, `LastName`, `Email` and, `Age`.
- `Age` columns must having a `CHECK` constraint where it ensures the `Age` will not be less than 18.

Query:

```
CREATE TABLE Customers(
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(60) NOT NULL,
    LastName VARCHAR(60) NOT NULL,
    Email VARCHAR(80) UNIQUE,
    Age INT CHECK (Age > 18)
);
```

Execution:



Q7. You have a table named "Orders" with columns: OrderID (int), CustomerID (int), OrderDate (date), and TotalAmount (decimal). Create a foreign key constraint on the "CustomerID" column referencing the "Customers" table.

- Create Orders Table with Foreign Key Constraint: Now, create the "Orders" table while referencing the "Customers" table by including a foreign key constraint. Here's an example SQL script to create the "Orders" table with the foreign key constraint:

Query:

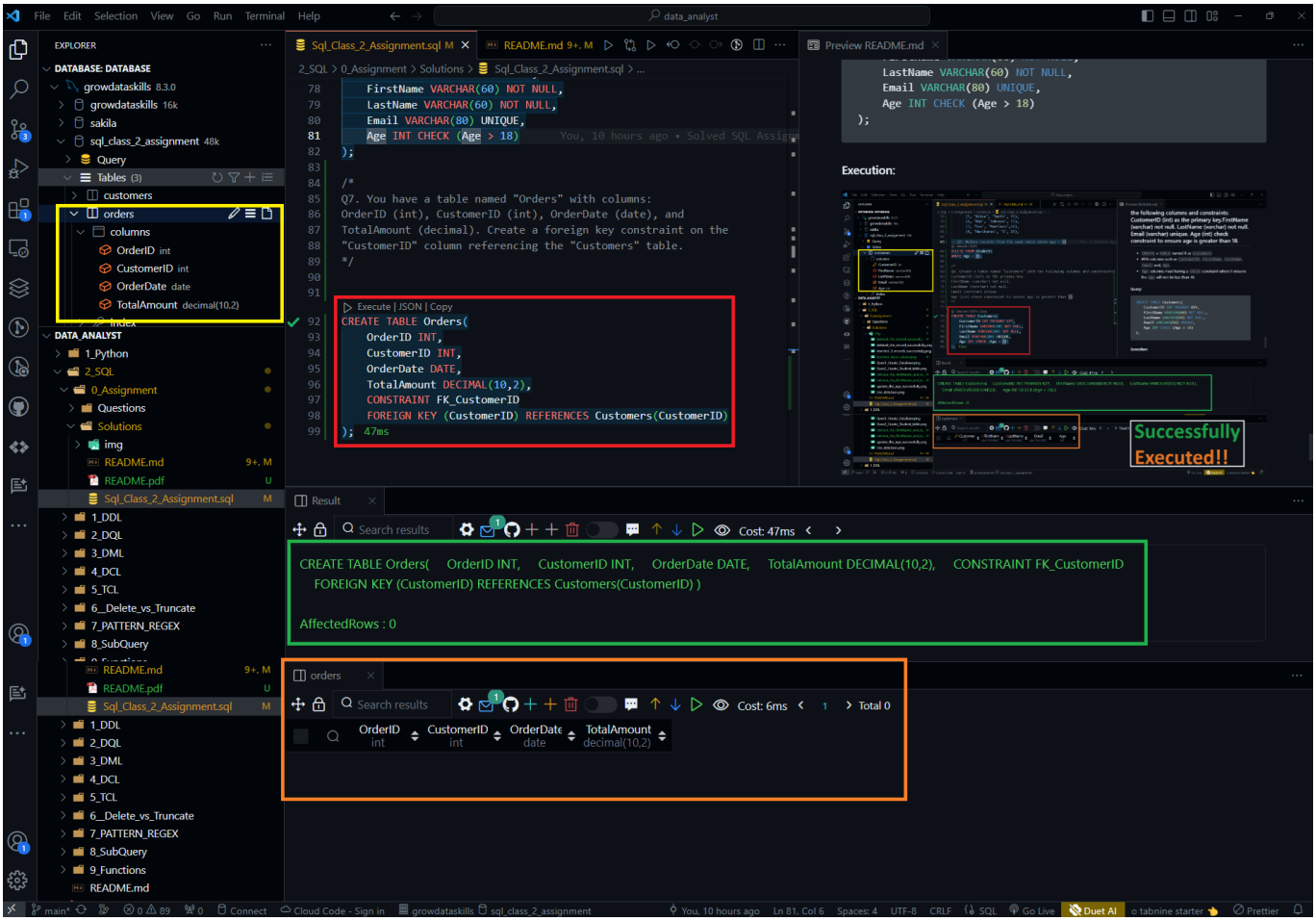
```
CREATE TABLE Orders(
    OrderID INT,
    CustomerID INT,
    OrderDate DATE,
    TotalAmount DECIMAL(10,2),
    CONSTRAINT FK_CustomerID
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Introduction: Provide an overview of the requirement to create the "Orders" table and to establish a relationship with the "Customers" table.

Step 1: Explain the creation of the "Customers" table with columns like CustomerID and CustomerName. Include the SQL script for creating the table.

Step 2: Describe the creation of the "Orders" table with columns like OrderID, CustomerID, OrderDate, and TotalAmount. Clarify the addition of a foreign key constraint on the "CustomerID" column, referencing the "CustomerID" column in the "Customers" table. Include the SQL script for creating the table along with the foreign key constraint.

Execution:



Q8. Create a table named "Employees" with columns: EmployeeID (int) as the primary key. FirstName (varchar) not null. LastName (varchar) not null. Salary (decimal) check constraint to ensure salary is between 20000 and 100000.

- `CREATE TABLE Employees`: This line initiates the creation of the "`Employees`" table.
- (`EmployeeID` int `PRIMARY KEY`): Defines the "`EmployeeID`" column as an integer primary key.
- (`FirstName` varchar(80) `NOT NULL`): Specifies the "`FirstName`" column as a non-null varchar datatype.
- (`LastName` varchar(80) `NOT NULL`): Specifies the "`LastName`" column as a non-null varchar datatype.
- `Salary DECIMAL(10,2) CHECK` (`Salary BETWEEN` 20000 `and` 100000): `SET` the "`Salary`" column as a decimal type with a precision of 10 digits and 2 decimal places and adds a check constraint to ensure that the salary is between 20000 and 100000.

Query:

```sql
CREATE TABLE Employees(
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(80) NOT NULL,
    LastName VARCHAR(80) NOT NULL,
    Salary DECIMAL(10,2) CHECK (Salary BETWEEN 20000 and 100000)
);
```

Execution: