

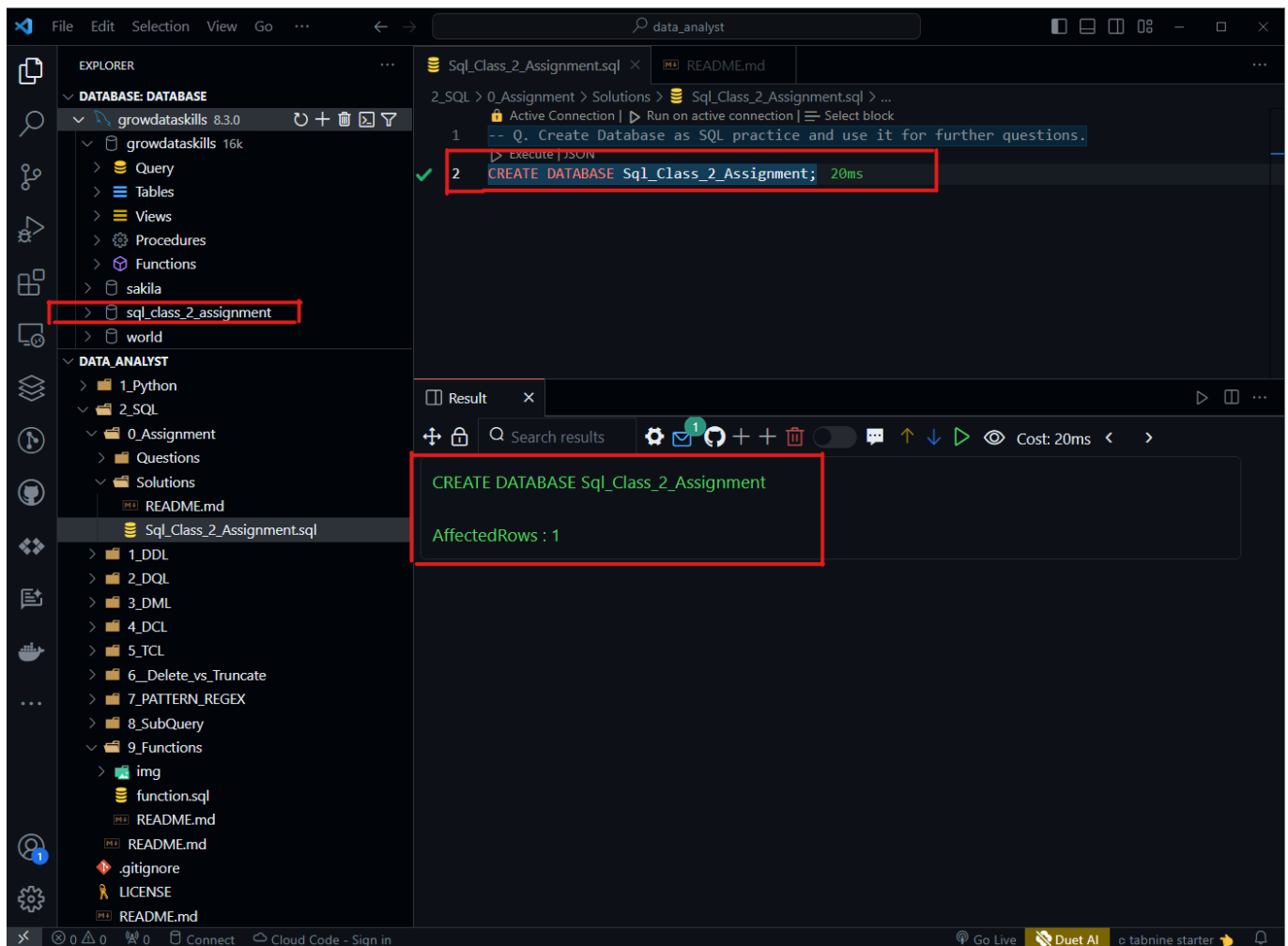
Q1. Create Database as SQL practice and use it for further questions.

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

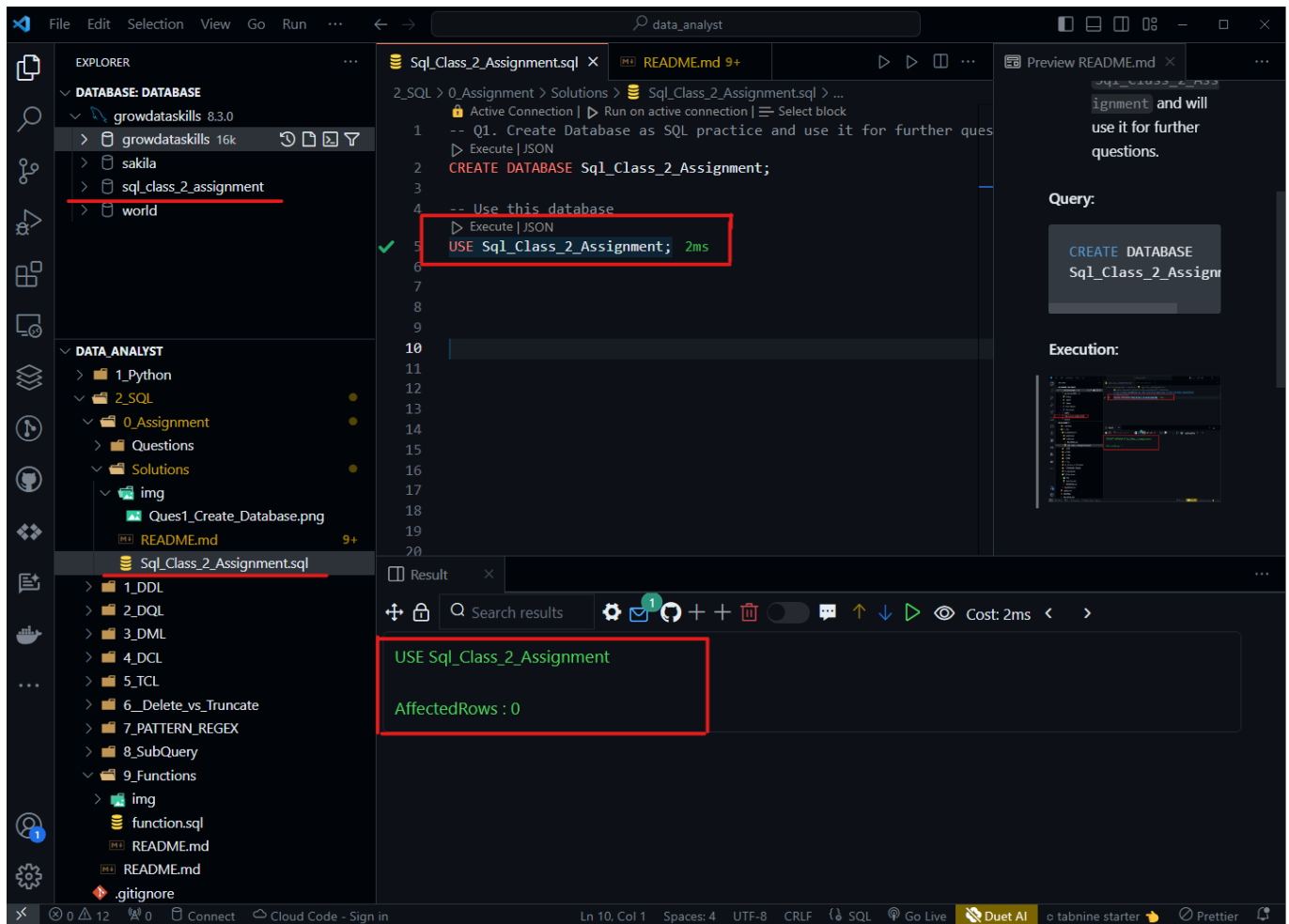
Query:

```
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:

The screenshot shows a SQL IDE interface with the following components:

- Explorer:** A sidebar on the left showing the project structure. The 'Tables' folder is expanded, showing a table named 'Students' with columns: StudentID (int), FirstName (varchar(60)), LastName (varchar(60)), and Age (int).
- Query Editor:** The main area contains a SQL script. The first part creates a database 'Sql_Class_2_Assignment'. The second part creates a table 'Students' with the specified columns. A handwritten red bracket and the word 'Query' are drawn over the table creation code.
- Execution Results:** Below the query editor, the results of the execution are shown. It displays the 'CREATE TABLE Students' statement and indicates that 0 rows were affected.
- Table Structure:** At the bottom, a preview of the 'Students' table structure is shown, listing the columns: StudentID (int), FirstName (varchar(60)), LastName (varchar(60)), and Age (int).

- 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:

The screenshot shows the SQL Server Enterprise Manager interface. The Explorer pane on the left shows the database structure, including the 'Students' table. The central query editor displays the following SQL code:

```
2.SQL > 0.Assignment > Solutions > Sql_Class_2_Assignment.sql > ...  
7 /*  
8 Q2. Create a table named "Students" with the following columns:  
9 StudentID (int), FirstName (varchar), LastName (varchar),  
10 and Age (int).  
11 Insert at least three records into the table.  
12 */  
13  
14 -- Execute | JSON | Copy  
15 CREATE TABLE Students(  
16 StudentID INT,  
17 FirstName VARCHAR(60),  
18 LastName VARCHAR(60),  
19 Age INT  
20 );  
21  
22 -- Insert 3 records randomly  
23 -- Execute | JSON  
24 INSERT INTO Students(StudentID, FirstName, LastName, Age)  
25 VALUES (1, "Arpit", "Dubey", 25),  
26 (2, "Shaija", "Mishra", 23),  
27 (3, "Spider", "Man", 29); /ms
```

The Results pane on the right shows the execution output, including the SQL statement and the affected rows. A table view at the bottom displays the inserted data:

StudentID	FirstName	LastName	Age
1	Arpit	Dubey	25
2	Shaija	Mishra	23
3	Spider	Man	29

VALUES are INSERTED!! SUCCESSFULLY

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:

The screenshot shows a SQL IDE with the following components:

- Explorer:** Shows a database named 'DATABASE' with tables 'growdataskills 8.3.0', 'growdataskills 16k', 'sakila', and 'sql_class_2_assignment 16k'. The 'students' table is selected, showing columns: StudentID (int), FirstName (varchar(60)), LastName (varchar(60)), and Age (int).
- SQL Editor:** Contains the following SQL script:

```
20 -- Insert 3 records randomly
21 -- Execute JSON
22 INSERT INTO Students(StudentID, FirstName, LastName, Age)
23 VALUES (1, "Arpit", "Dubey", 25),
24         (2, "Shaija", "Mishra", 23),
25         (3, "Spider", "Man", 29);
26
27 /*
28 Q3. Update the age of the student with StudentID '1' to '21'.
29 Delete the student with StudentID '3' from the "Students" table.
30 */
31
32 -- Execute JSON
33 UPDATE Students
34 SET Age = 21
35 WHERE StudentID = 1; 6ms
```
- Results:** Shows the execution of the UPDATE statement: 'UPDATE Students SET Age = 21 WHERE StudentID = 1' with 'AffectedRows : 1'.
- Table View:** Displays the 'students' table with the following data:

StudentID	FirstName	LastName	Age
1	Arpit	Dubey	21
2	Shaija	Mishra	23
3	Spider	Man	29

An arrow points from the 'Age' column of the first row (21) to a text box that says 'Previously, it is 25'.

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

Query:

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

Execution:

The screenshot shows a SQL IDE with a query editor, a file explorer, and a results pane. The query editor contains the following SQL code:

```
27 /*
28 Q3. Update the age of the student with StudentID '1' to '21'.
29 Delete the student with StudentID '3' from the "Students" table.
30 */
31 -- UPDATE the age of the student with StudentID is equal to 1.
32 -- Execute | JSON
33 UPDATE Students
34 SET Age = 21
35 WHERE StudentID = 1;
36 -- DELETE students record WHERE StudentID is equal to 3.
37 -- Execute | JSON
38 DELETE FROM Students
39 WHERE StudentID = 3;
40
41
```

The results pane shows the execution of the DELETE query:

```
DELETE FROM Students WHERE StudentID = 3
AffectedRows: 1
```

The table view shows the remaining students:

StudentID	FirstName	LastName	Age
1	Arpit	Dubey	21
2	Shaija	Mishra	23

A callout box with a red border and text says: "Record with StudentID is equal to 3. Deleted Record Successfully".

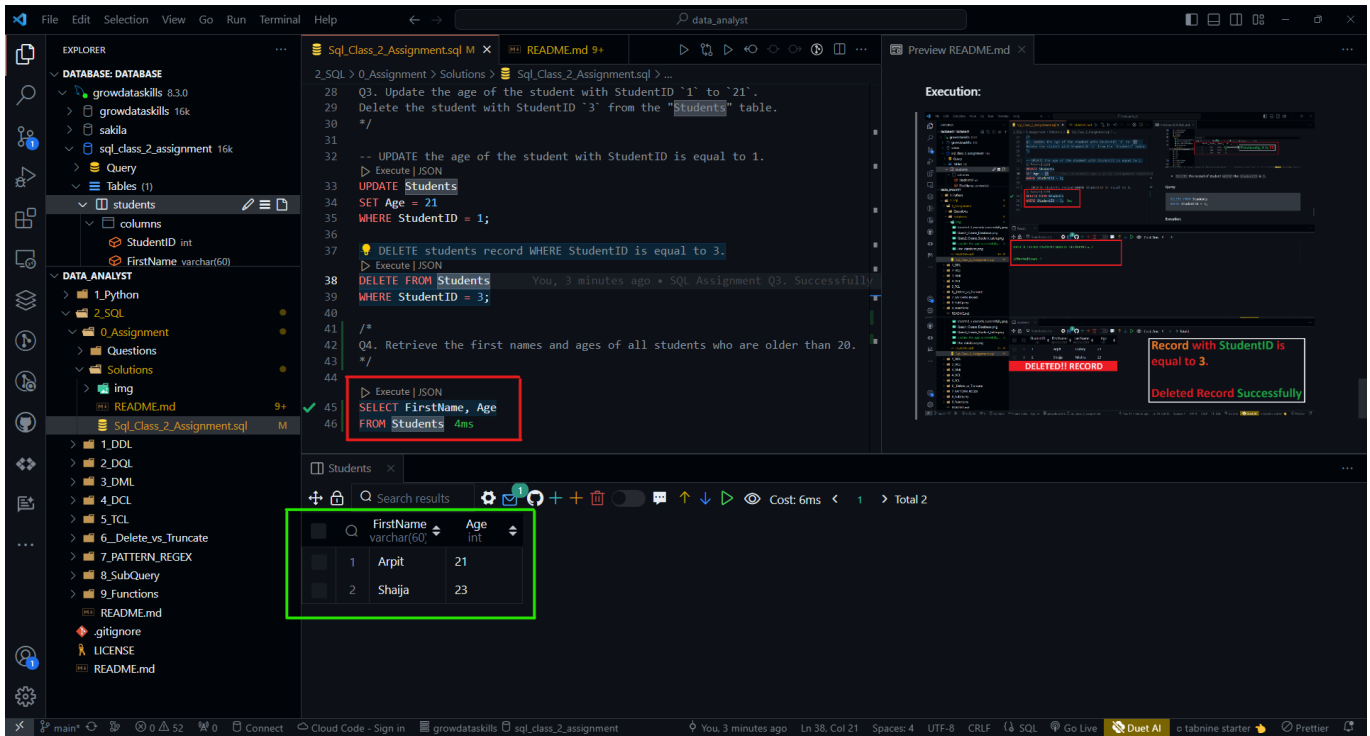
Q4. Retrieve the first names and ages of all students who are older than 20.

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

Query:

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

Execution:

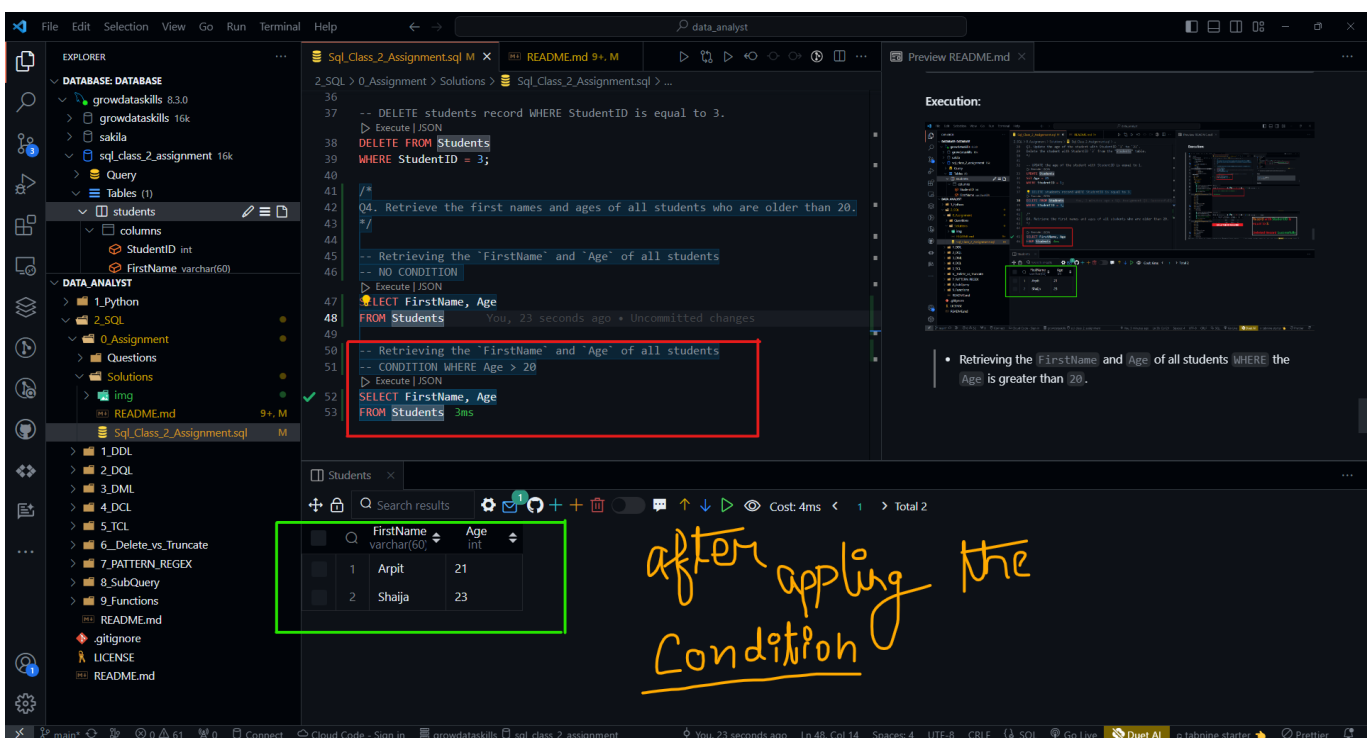


- Retrieving the **FirstName** and **Age** of all students **WHERE** the **Age** is greater than 20..

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:

The screenshot shows a SQL IDE interface with a query editor, a results pane, and a table explorer. The query editor contains the following SQL code:

```
2_SQL > 0.Assignment > Solutions > Sql_Class_2_Assignment.sql > ...
47 SELECT FirstName, Age
48 FROM Students
49
50 -- Retrieving the "FirstName" and "Age" of all students
51 -- CONDITION WHERE Age > 20
52
53 -- Execute | JSON
54 SELECT FirstName, Age
55 FROM Students
56
57 -- Inserting more values
58 -- Execute | JSON | Copy
59 INSERT INTO Students(StudentID, FirstName, LastName, Age)
60 VALUES (4, "Rahul", "Kumar", 12),
61         (5, 'Alice', 'Smith', 25),
62         (6, 'Bob', 'Johnson', 15),
63         (7, 'Eva', 'Martinez', 16),
64         (8, 'Hariharan', 'S', 26);
```

The results pane shows the execution of the query, and the table explorer displays the following data:

StudentID	FirstName	LastName	Age
1	Arpit	Dubey	21
2	Shaija	Mishra	23
3	Rahul	Kumar	12
4	Alice	Smith	25
5	Bob	Johnson	15
6	Eva	Martinez	16
7	Hariharan	S	26

A handwritten note in yellow ink says "Inserted more Values!!" with a bracket pointing to the new rows in the table.

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:

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:

```
DELETE FROM Students WHERE Age < 18
AffectedRows : 3
```

StudentID	FirstName	LastName	Age
1	Arpit	Dubey	21
2	Shaija	Mishra	23
3	Alice	Smith	25
4	Hariharan	S	26

Successfully, removes all the records WHERE the students Age column have VALUES less than 18

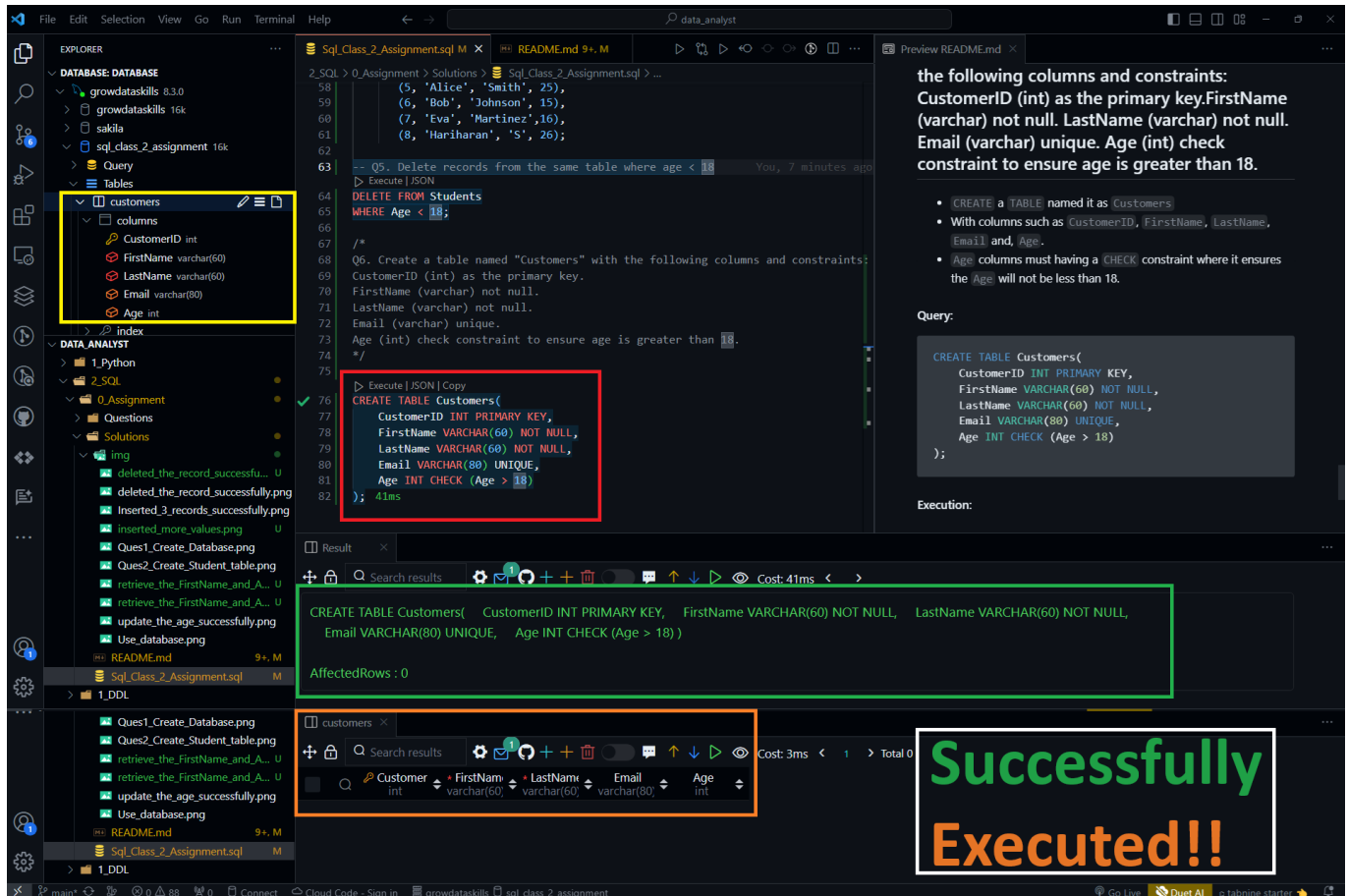
Q6. Create a table named "Customers" with the following columns and constraints: CustomerID (int) as the primary key. FirstName (varchar) not null. LastName (varchar) not null. Email (varchar) unique. Age (int) check constraint to ensure age is greater than 18.

- 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:

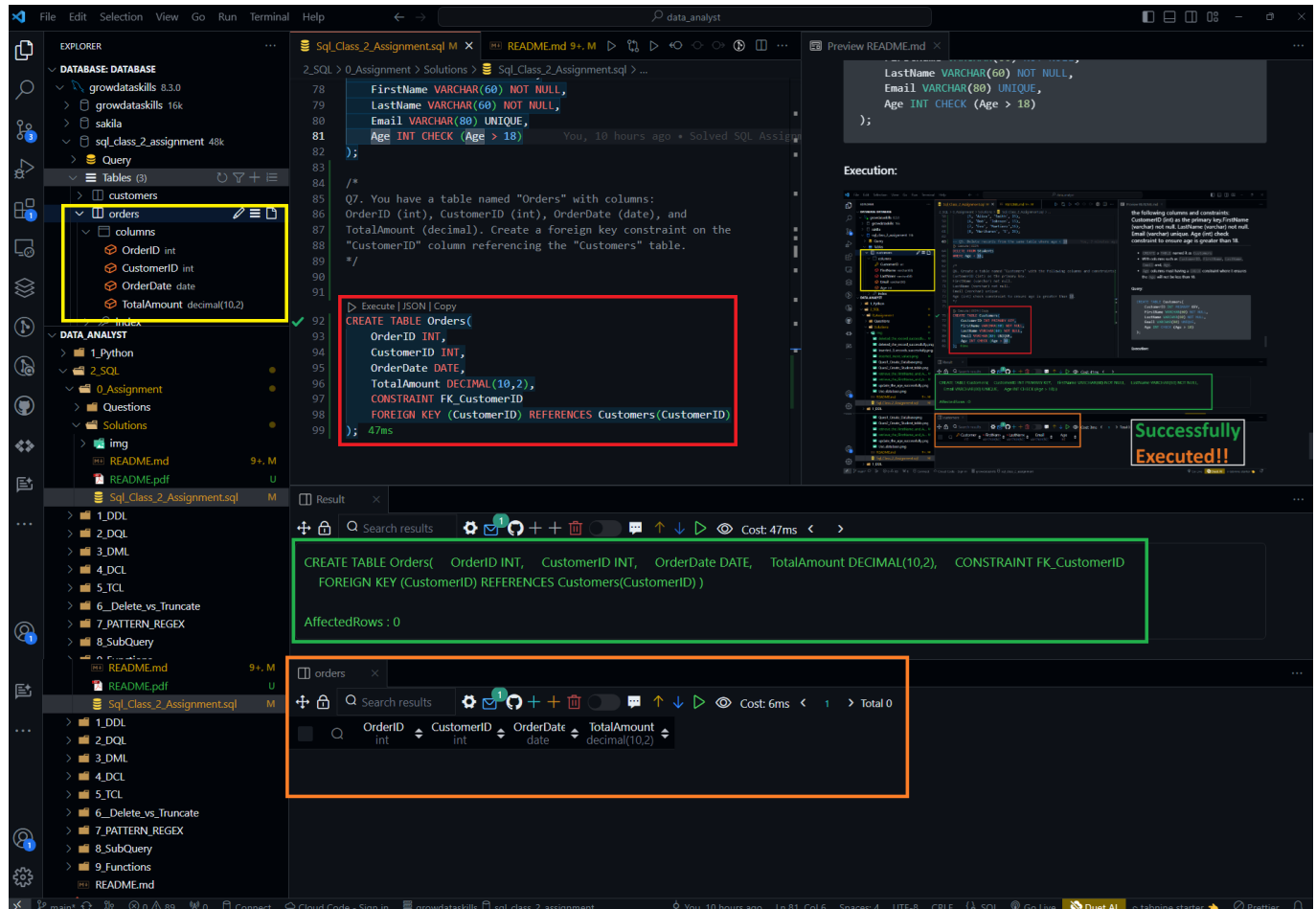
```
CREATE TABLE Orders(
  OrderID INT,
  CustomerID INT,
  OrderDate DATE,
  TotalAmount DECIMAL(10,2),
  CONSTRAINT FK_CustomerID
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

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:

```
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:

The screenshot displays a database IDE interface with the following components:

- EXPLORER:** Shows a project structure with a database named 'growdataSkills'. Under 'Tables (4)', the 'employees' table is highlighted.
- SQL Editor:** Contains the SQL script for creating the 'Employees' table. The script is highlighted with a red box. The script includes a comment: '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.'
- Execution Results:** Shows the successful execution of the SQL script. The results are highlighted with a red box, displaying the SQL statement and 'AffectedRows : 0'.
- Table Structure:** A preview of the 'employees' table structure is shown at the bottom, highlighting the columns: EmployeeID (int), FirstName (varchar(80)), LastName (varchar(80)), and Salary (decimal(10,2)).
- Preview README.md:** A sidebar on the right shows a preview of the README.md file, which contains the same SQL script and a comment: '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.'