

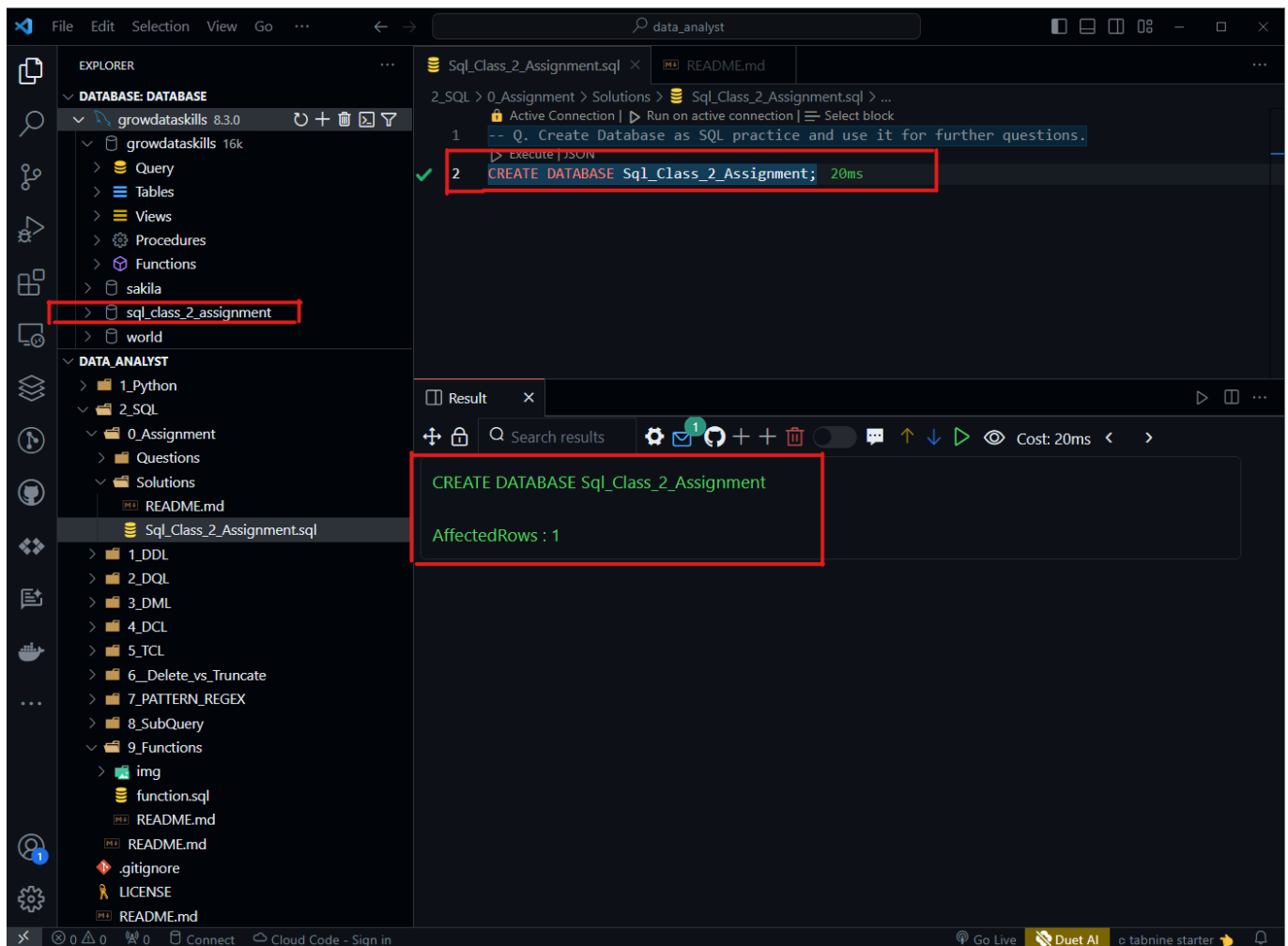
Q1. Create Database as SQL practice and use it for further questions.

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

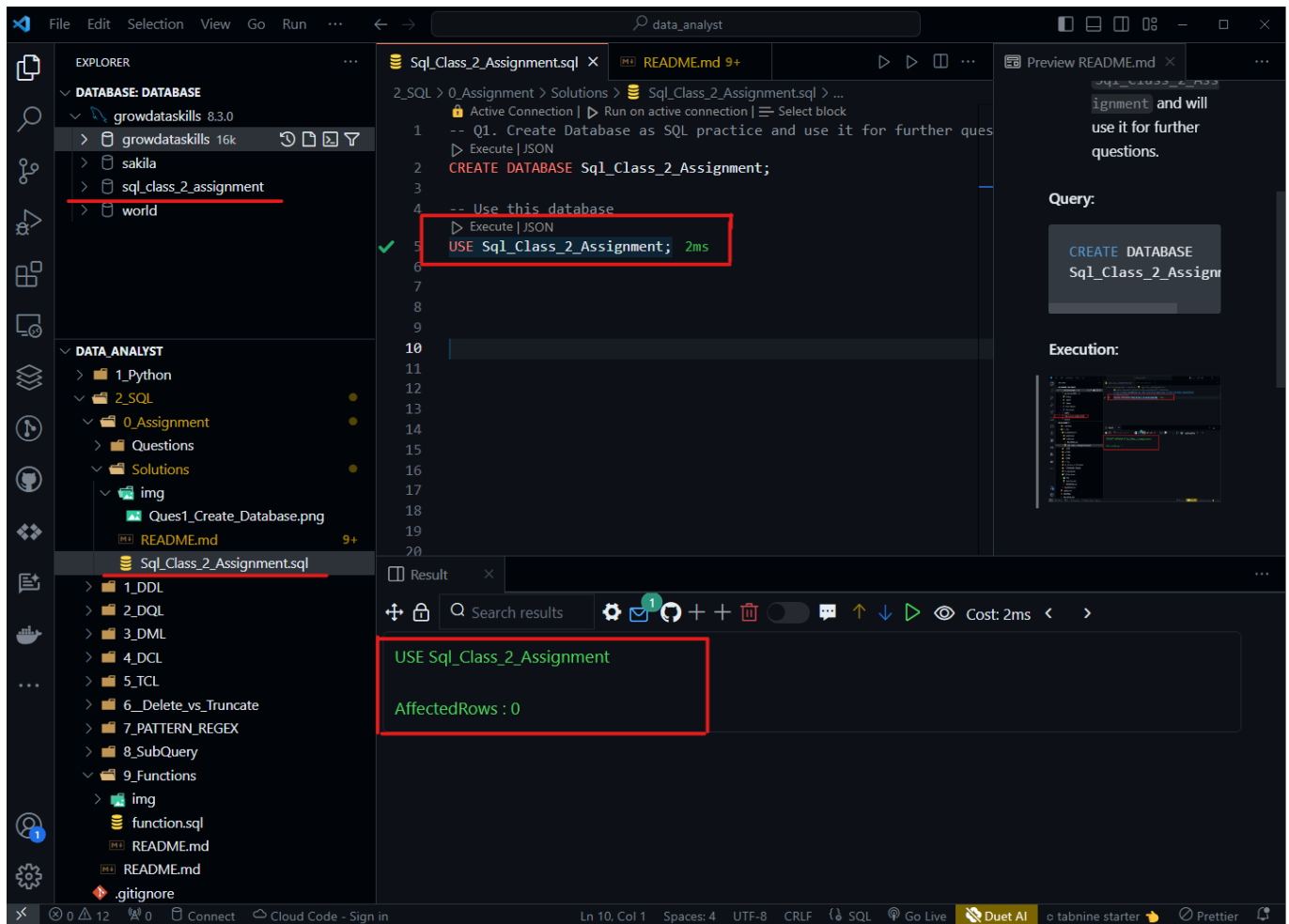
Query:

```
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 'DATABASE DATABASE' folder is expanded, showing 'sakila' and 'sql_class_2_assignment 16k'. The 'Tables' folder is expanded, showing 'students' with columns 'StudentID int', 'FirstName varchar(60)', 'LastName varchar(60)', and 'Age int'.
- SQL Editor:** The main workspace shows a SQL script in 'Sql_Class_2_Assignment.sql'. The script includes comments and SQL commands: 'CREATE DATABASE Sql_Class_2_Assignment;', 'USE Sql_Class_2_Assignment;', and 'CREATE TABLE Students(StudentID INT, FirstName VARCHAR(60), LastName VARCHAR(60), Age INT);'. A red box highlights the 'CREATE TABLE' statement, and a red bracket with the word 'Query' is drawn next to it.
- Query Results:** The 'Result' tab at the bottom shows the execution of the 'CREATE TABLE' statement. It displays 'AffectedRows: 0' and 'Cost: 32ms'.
- Table Structure:** A 'students' table structure is shown below the results, with 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 displays the SQL Server Enterprise Manager interface. In the Explorer pane on the left, the 'students' table is highlighted under the 'DATABASE' folder. The central query editor shows the following SQL code:

```
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:

SQL Class 2 Assignment.sql

```
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
36
```

Result

```
UPDATE Students SET Age = 21 WHERE StudentID = 1
AffectedRows : 1
```

students

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

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 the VS Code interface with the following components:

- Explorer:** Shows a project structure with folders like 'DATABASE: DATABASE', 'DATA ANALYST', and '1.SQL'. The 'students' table is selected under 'Tables (1)'.
- Editor:** Contains a SQL file 'Sql_Class_2_Assignment.sql' with the following 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
```
- Preview README.md:** Shows a table with columns 'StudentID', 'FirstName', 'LastName', and 'Age'. A red box highlights the row with StudentID 3, with a note 'Previously, it is 25'.
- Result:** Shows the executed SQL query: 'DELETE FROM Students WHERE StudentID = 3' and 'AffectedRows : 1'.
- students table:** A table with columns 'StudentID', 'FirstName', 'LastName', and 'Age'. It contains two rows: (1, Arpit, Dubey, 21) and (2, Shaija, Mishra, 23). A red box highlights the row with StudentID 2, with a note 'DELETED!! RECORD'.
- Execution:** Shows the executed SQL query: 'DELETE FROM Students WHERE StudentID = 3' and 'Cost: 2ms'.

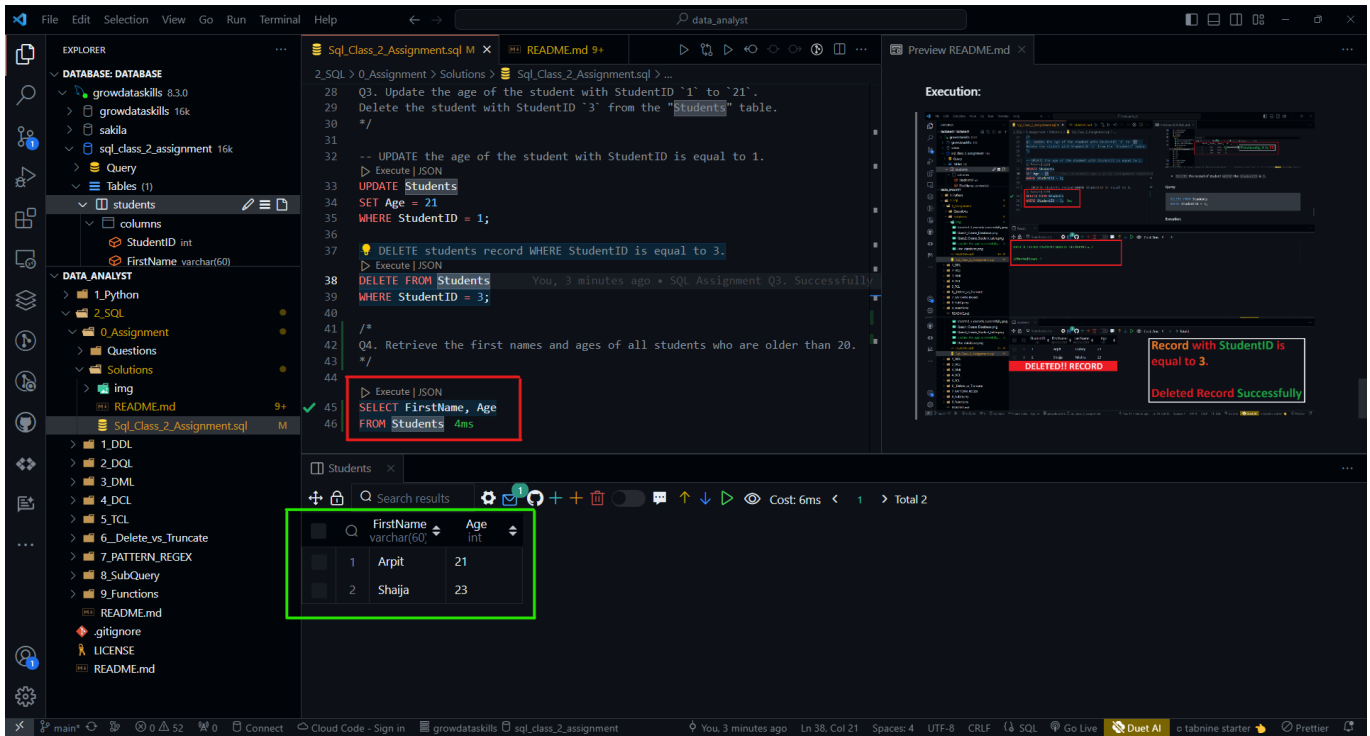
Q4. Retrieve the first names and ages of all students who are older than 20.

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

Query:

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

Execution:

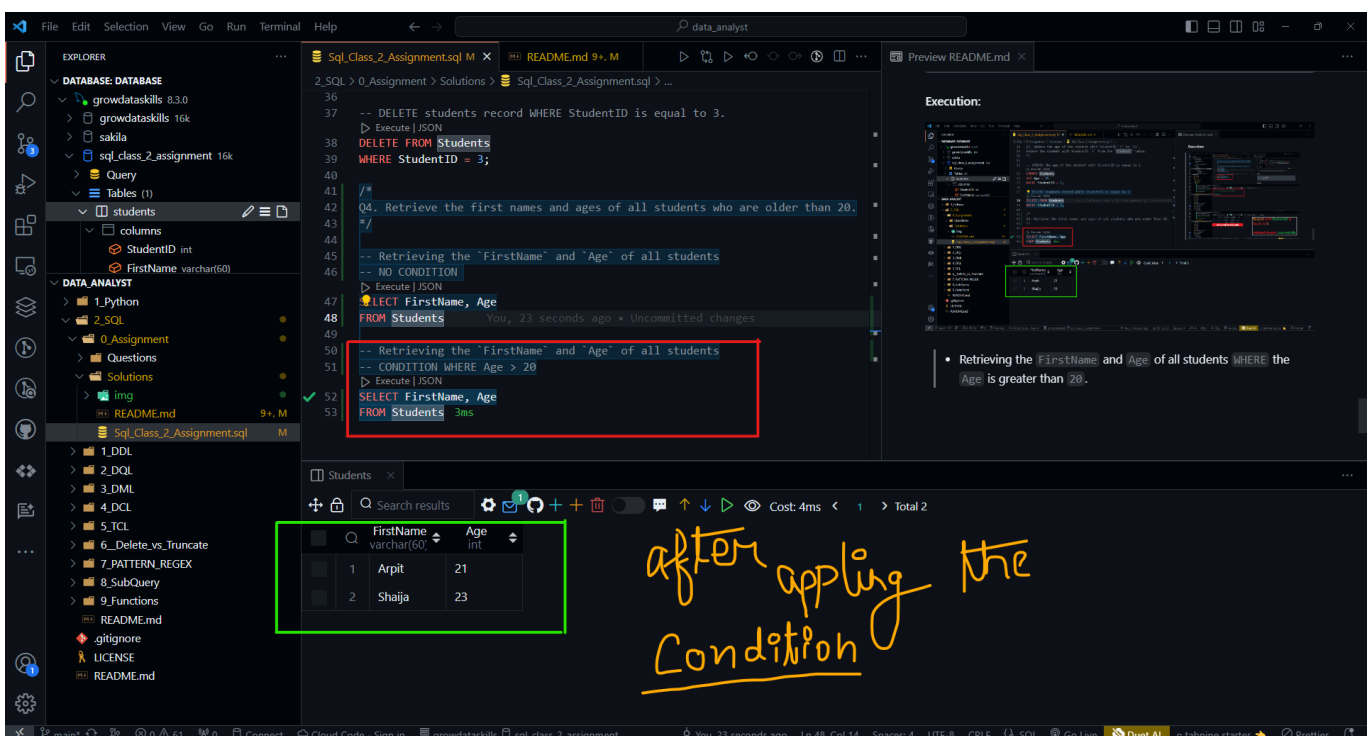


- Retrieving the **FirstName** and **Age** of all students **WHERE** the **Age** is greater than 20..

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 file explorer on the left, a central editor, and a results pane on the right. The editor contains a SQL script with an INSERT statement. The results pane shows the output of a SELECT query, displaying a table with 7 rows of student data. A handwritten yellow note with a bracket points to the last three rows of the table, indicating the newly inserted values.

Query:

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

Execution:

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

} Inserted more Values!!

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:

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:

```

52 SELECT FirstName, Age
53 FROM Students
54
55 -- Inserting more values
56 INSERT INTO Students(StudentID, FirstName, LastName, Age)
57 VALUES (4, "Rahul", "Kumar", 12),
58         (5, "Alice", "Smith", 25),
59         (6, "Bob", "Johnson", 15),
60         (7, "Eva", "Martinez", 16),
61         (8, "Hariharan", "S", 26);
62
63 -- Q5. Delete records from the same table where age < 18
64 DELETE FROM Students
65 WHERE Age < 18;
66

```

The results pane shows the execution of the DELETE query:

```

DELETE FROM Students
WHERE Age < 18;

Execution:

AffectedRows : 3

```

Below the results, a table view shows the remaining data in the 'students' table:

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:

