

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:

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists several databases and projects. A red box highlights the folder 'sql_class_2_assignment'. The main area contains a code editor with two lines of SQL. Line 1 is a comment: '-- Q. Create Database as SQL practice and use it for further questions.'. Line 2 is the command: 'CREATE DATABASE Sql_Class_2_Assignment;'. To the right of the code editor is a 'Result' panel showing the output of the query: 'CREATE DATABASE Sql_Class_2_Assignment' and 'AffectedRows : 1'. A red box highlights this output. The status bar at the bottom shows various icons and the text 'Cloud Code - Sign in'.

```
-- Q. Create Database as SQL practice and use it for further questions.  
CREATE DATABASE Sql_Class_2_Assignment;
```

```
USE Sql_Class_2_Assignment;
```

The screenshot shows a Visual Studio Code interface with the following details:

- Explorer View:** Shows a database connection to "growdataskills" containing databases "sakila", "sql_class_2_assignment", and "world". A red box highlights the "sql_class_2_assignment" database.
- Code Editor:** An SQL file named "Sql_Class_2_Assignment.sql" is open. Line 5 contains the command "USE Sql_Class_2_Assignment;". This line is also highlighted with a red box.
- Output Panel:** The "Result" tab shows the output of the executed query:


```
USE Sql_Class_2_Assignment
AffectedRows : 0
```
- Right-hand Side:** Includes sections for "Query" (showing the executed command) and "Execution" (showing the execution results in a terminal-like interface).

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 the Visual Studio Code interface with several tabs open:

- Sql_Class_2_Assignment.sql**: Contains the following SQL code:


```

1 -- Q1. Create Database as SQL practice and use it for further questions.
2 CREATE DATABASE Sql_Class_2_Assignment;
3
4 -- Use this database
5 USE Sql_Class_2_Assignment;
6
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 );
      
```
- README.md**: Preview tab shows the contents of the README.md file.
- Result**: Shows the output of the query:


```

CREATE TABLE Students( StudentID INT, FirstName VARCHAR(60), LastName VARCHAR(60), Age INT )
AffectedRows : 0
      
```
- students**: Shows the structure of the Students table:

	StudentID	FirstName	LastName	Age
	int	varchar(60)	varchar(60)	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:

`/*`

`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.`

`*/`

`CREATE TABLE Students(`

`StudentID INT,`

`FirstName VARCHAR(60),`

`LastName VARCHAR(60),`

`Age INT`

`);`

`-- Insert 3 records randomly`

`INSERT INTO Students(StudentID, FirstName, LastName, Age)`

`VALUES (1, "Arpit", "Dubey", 25),`

`(2, "Shajja", "Mishra", 23),`

`(3, "Spider", "Man", 29);`

`Cost: 7ms`

`AffectedRows : 3`

`StudentID FirstName LastName Age`

StudentID	FirstName	LastName	Age
1	Arpit	Dubey	25
2	Shajja	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 Visual Studio Code interface with several panes. The Explorer pane on the left shows a database named 'growdataskills' with a schema including 'sakila' and 'sql_class_2_assignment'. The 'Tables' section under 'sql_class_2_assignment' contains a single table 'students' with columns: StudentID (int), FirstName (varchar(60)), LastName (varchar(60)), and Age (int). The 'Sql Class 2 Assignment.sql' file in the Editor pane contains SQL code for inserting records into the 'Students' table. The 'Preview README.md' pane shows the results of the insert operation, displaying three new student records: Arpit Dubey (Age 21), Shajja Mishra (Age 23), and Spider Man (Age 29). A red box highlights the 'VALUES' part of the insert statement. The 'Result' pane shows the update command: 'UPDATE Students SET Age = 21 WHERE StudentID = 1;' and the output 'AffectedRows : 1'. The 'students' table in the bottom right shows the updated data, with the age for StudentID 1 now set to 21. A green box highlights the updated value '21' in the table.

```

19 );
20
21 -- Insert 3 records randomly
22 > Execute | JSON
23 INSERT INTO Students(StudentID, FirstName, LastName, Age)
24 VALUES (1, "Arpit", "Dubey", 25),
25 (2, "Shajja", "Mishra", 23),
26 (3, "Spider", "Man", 29);
27
28 /*
29 03. Update the age of the student with StudentID `1` to `21`.
30 Delete the student with StudentID `3` from the "Students" table.
31 */
32 > Execute | JSON
33 UPDATE Students
34 SET Age = 21
35 WHERE StudentID = 1; 6ms
36

```

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.

AffectedRows : 1

	StudentID	FirstName	LastName	Age
1	1	Arpit	Dubey	21
2	2	Shajja	Mishra	23
3	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 a Visual Studio Code interface with several panes:

- Explorer:** Shows a file tree with a folder named "DATA ANALYST" containing subfolders like "1_Python", "2_SQL", and "Solutions".
- Editor:** Displays a SQL script named "Sql_Class_2_Assignment.sql" with the following content:


```

27 /* Q3. Update the age of the student with StudentID '1' to '21'.
28 Delete the student with StudentID '3' from the "Students" table.
29 */
30
31
32 -- UPDATE the age of the student with StudentID is equal to 1.
33 > Execute | JSON
34 UPDATE Students
35 SET Age = 21
36 WHERE StudentID = 1;
37
38 -- DELETE students record WHERE StudentID is equal to 3.
39 > Execute | JSON
40 DELETE FROM Students
41 WHERE StudentID = 3; 5ms
      
```
- Output:** Shows the execution results of the DELETE query:


```

DELETE FROM Students WHERE StudentID = 3
AffectedRows : 1
      
```
- Preview:** A separate window titled "Preview README.md" shows a table named "students" with two rows of data:

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

A red box highlights the message "DELETED!! RECORD" in the output pane, and another red box highlights the text "Record with StudentID is equal to 3." in the preview pane.

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:

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows the database structure under "DATABASE: DATABASE". A red box highlights the "students" table.
- SQL Class_2_Assignment.sql:** The current file being edited contains the following SQL code:


```

2_SQL > 0.Assignment > Solutions > Sql_Class_2_Assignment.sql > ...
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 -- UPDATE the age of the student with StudentID is equal to 1.
33 UPDATE Students
34 SET Age = 21
35 WHERE StudentID = 1;
36
37 /* DELETE students record WHERE StudentID is equal to 3.
38 DELETE FROM Students
39 WHERE StudentID = 3;
40 */
41 /*
42 Q4. Retrieve the first names and ages of all students who are older than 20.
43 */
44
45 /*
46 SELECT FirstName, Age
FROM Students
      
```
- Execution:** The execution tab shows the results of the query "SELECT FirstName, Age FROM Students". It displays two rows: Arpit (Age 21) and Shajja (Age 23). A red box highlights the table results.
- Terminal:** The terminal shows the command "Cloud Code - Sign in" and other system information.

- 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:

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows the database structure under "DATABASE: DATABASE". A red box highlights the "students" table.
- SQL Class_2_Assignment.sql:** The current file being edited contains the following SQL code:


```

-- DELETE students record WHERE StudentID is equal to 3.
DELETE FROM Students
WHERE StudentID = 3;

/*
Q4. Retrieve the first names and ages of all students who are older than 20.
*/
-- Retrieving the 'FirstName' and 'Age' of all students
-- NO CONDITION
SELECT FirstName, Age
FROM Students
      
```
- Execution:** The execution tab shows the results of the query "SELECT FirstName, Age FROM Students". It displays two rows: Arpit (Age 21) and Shajja (Age 23). A red box highlights the table results.
- Terminal:** The terminal shows the command "Cloud Code - Sign in" and other system information.

A handwritten note in the bottom right corner of the execution window says: "after applying the Condition".

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 development environment with the following details:

- Explorer:** Shows the database structure, including the `Students` table under `sql_class_2_assignment` with columns `StudentID`, `FirstName`, and `LastName`.
- SQL Editor:** Contains two queries:
 - The first query retrieves `FirstName` and `Age` for students where `Age > 20`.
 - The second query inserts new student records into the `Students` table.
- Results:** The results of the second query show 8 rows inserted into the `Students` table, with 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
- Handwritten Note:** A yellow bracket on the right side of the results table is annotated with the text "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 the Visual Studio Code interface with several panes:

- EXPLORER** pane on the left showing a file tree with various database and script files.
- Sql_Class_2_Assignment.sql** pane showing the following SQL code:


```

SELECT FirstName, Age
FROM Students

-- Inserting more values
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);
      
```
- Result** pane showing the output of the DELETE query:


```

DELETE FROM Students WHERE Age < 18
AffectedRows : 3
      
```
- Preview README.md** pane showing the content of the README.md file.
- Students** table preview pane showing the following data:

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

Query:

```

DELETE FROM Students WHERE Age < 18
      
```

Execution:

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:

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:

```
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)
);
```

AffectedRows : 0

customers

Successfully Executed!!

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.

```
ALTER TABLE Orders
ADD CONSTRAINT FK_Customer
FOREIGN KEY (CustomerID)
REFERENCES Customers(CustomerID);
```

Execution:

The screenshot shows a code editor interface with several panes. On the left, the 'EXPLORER' pane shows a database structure with a 'customers' table containing an 'orders' table. The 'DATA ANALYST' pane shows various assignment files and a 'Sql Class 2 Assignment.sql' file which contains the SQL script for creating the 'Orders' table. The main code editor pane displays the SQL code:

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

The 'PREVIEW' pane shows the resulting table structure:

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

The 'EXECUTION' pane shows the results of the execution:

Successfully Executed!!

The bottom pane shows the table 'orders' with its columns: OrderID, CustomerID, OrderDate, and TotalAmount.

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.

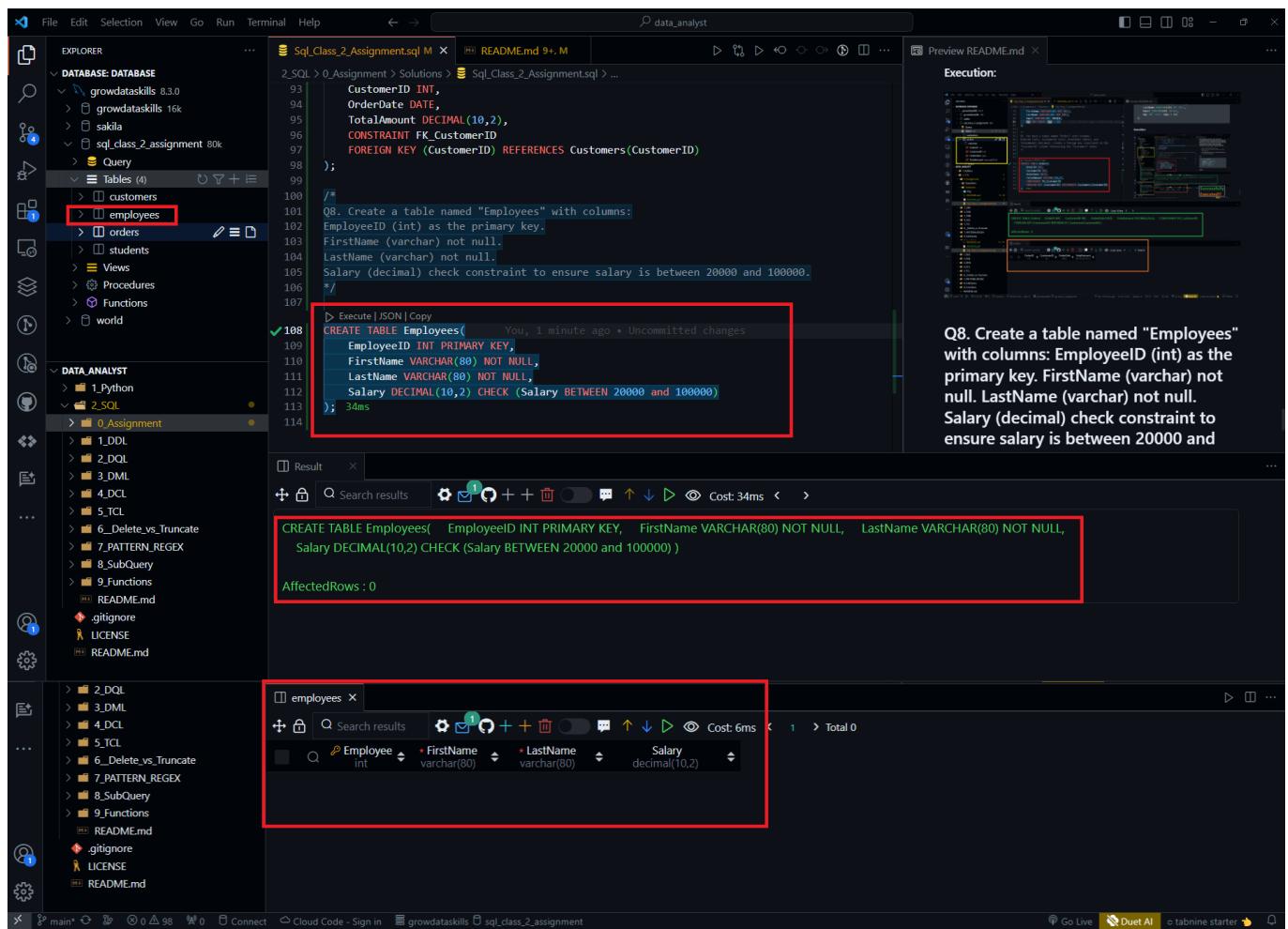
- **CREATE TABLE Employees:** This line initiates the creation of the "Employees" table.
- **(EmployeeID int PRIMARY KEY):** Defines the "EmployeeID" column as an integer primary key.
- **(FirstName varchar(80) NOT NULL):** Specifies the "FirstName" column as a non-null varchar datatype.
- **(LastName 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 shows the Visual Studio Code interface with the following details:

- Explorer:** Shows the database structure with a red box around the 'employees' table under the 'Tables (4)' section.
- Editor:** Displays the SQL code for creating the 'Employees' table. The line `Salary DECIMAL(10,2) CHECK (Salary BETWEEN 20000 and 100000)` is highlighted with a red box.
- Output:** Shows the result of the execution: "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))". The status bar indicates "AffectedRows : 0".
- SQL Server Object Explorer:** Shows the 'employees' table with its columns: EmployeeID, FirstName, LastName, and Salary.

Q9. Create a table named "Books" with columns: BookID (int) as the primary key. Title (varchar) not null. ISBN (varchar) unique.

- **CREATE TABLE Books:** Initiates the creation of a table named "Books" in the database.
- **(BookID INT PRIMARY KEY):** Defines the "BookID" column as an integer primary key, ensuring its uniqueness and serving as the identifier for each book.

- (**Title VARCHAR(80) NOT NULL**): Specifies the "Title" column as a non-null varchar data type, allowing up to 80 characters for the book title.
- (**ISBN VARCHAR(80) UNIQUE**): Defines the "ISBN" column as a varchar data type with a maximum length of 80 characters, and adds a constraint to ensure that each ISBN value is unique.

Query:

```
CREATE TABLE Books(
    BookID INT PRIMARY KEY,
    Title VARCHAR(80) NOT NULL,
    ISBN VARCHAR(80) UNIQUE
);
```

Execution:

The screenshot shows a SQL development environment with the following details:

- Left Panel (EXPLORER):** Shows the database structure. A folder named "books" is expanded, revealing "columns" (BookID int, Title varchar(80), ISBN varchar(80)), "index", "partitions", and "customers".
- Middle Panel (SQL Editor):** Displays the SQL code for creating the Books table. The entire code block is highlighted with a red box.
- Right Panel (Results):** Shows the execution results. The output includes the created table definition: `CREATE TABLE Books(BookID INT PRIMARY KEY, Title VARCHAR(80) NOT NULL, ISBN VARCHAR(80) UNIQUE)`. Below it, the message `AffectedRows : 0` is displayed.
- Bottom Status Bar:** Shows connection information, including the database name `sql_class_2_assignment`.

Q10. Consider a table named "Employees" with columns: EmployeeID, FirstName, LastName, and Age. Write an SQL query to retrieve the first name and last name of employees who are older than 30

- Adding Age column to pre-exist Employees TABLE

Query:

```
ALTER TABLE Employees
ADD Age INT;
```

Execution:

The screenshot shows a Visual Studio Code interface with several panes. The Explorer pane on the left displays the database structure, including a table named 'employees'. The SQL pane in the center contains the following code:

```
ALTER TABLE Employees
ADD Age INT;
```

The Results pane below shows the execution results:

```
ALTER TABLE Employees ADD Age INT
AffectedRows : 0
```

- Inserting some random values to Employees TABLE

Query:

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary, Age)
VALUES
(1, 'Shivcharan', 'Das', 45000.00, 32),
(2, 'Krish', 'Naik', 60000.00, 35),
(3, 'Rohit', 'Kapoor', 75000.00, 42),
(4, 'Ekta', 'Dubey', 95000.00, 36),
(5, 'Shailja', 'Mishra', 80000.00, 23),
(6, 'Hariharan', 'S', 65000.00, 25),
(7, 'Aman', 'Gupta', 90000.00, 45),
(8, 'Arpit', 'Dubey', 90800.00, 26),
(9, 'Shreya', 'Richharia', 42000.00, 27),
(10, 'Sudhanshu', 'Kumar', 30000.00, 40);
```

Execution:

The screenshot shows the Data Analyst interface with two tabs open: 'Sql_Class_2_Assignment.sql' and 'README.md'. In the 'Sql_Class_2_Assignment.sql' tab, a query is being run to alter the 'Employees' table to add an 'Age' column. The output shows the execution of the ALTER TABLE command and the insertion of 10 rows of sample data. The 'AffectedRows : 10' message indicates the successful execution of the insert statement.

```

ALTER TABLE Employees
ADD Age INT;

-- Inserting some random values to Employees TABLE
INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary, Age)
VALUES
(1, 'Shivcharan', 'Das', 45000.00, 32),
(2, 'Krish', 'Naik', 60000.00, 35),
(3, 'Rohit', 'Kapoor', 75000.00, 42),
(4, 'Ekta', 'Dubey', 95000.00, 36),
(5, 'Shailja', 'Mishra', 80000.00, 23),
(6, 'Hariharan', 'S', 65000.00, 25),
(7, 'Aman', 'Gupta', 90000.00, 45),
(8, 'Arpit', 'Dubey', 90800.00, 26),
(9, 'Shreya', 'Richharia', 42000.00, 27),
(10, 'Sudhanshu', 'Kumar', 30000.00, 40);

```

A red box highlights the inserted data, and a green box highlights the resulting table view below.

EmployeeID	FirstName	LastName	Salary	Age
1	Shivcharan	Das	45000.00	32
2	Krish	Naik	60000.00	35
3	Rohit	Kapoor	75000.00	42
4	Ekta	Dubey	95000.00	36
5	Shailja	Mishra	80000.00	23
6	Hariharan	S	65000.00	25
7	Aman	Gupta	90000.00	45
8	Arpit	Dubey	90800.00	26
9	Shreya	Richharia	42000.00	27
10	Sudhanshu	Kumar	30000.00	40

- SQL query to retrieve the first name and last name of employees who are older than 30

Query:

```

SELECT FirstName, LastName
FROM Employees
WHERE Age > 30

```

Execution:

The screenshot shows a Visual Studio Code interface with several panes. On the left is the Explorer pane showing a database structure with tables like `employees`, `orders`, and `customers`. The main code editor pane contains the following SQL query:

```

-- SQL query to retrieve the first name
-- and last name of employees who are older than 30
SELECT FirstName, LastName
FROM Employees
WHERE Age > 30
  
```

The results of this query are displayed in a table below the code editor, showing six rows of employee data. A green curly brace on the right side of the table groups the results, with handwritten text next to it reading "Employees who are older than 30 years".

	FirstName	LastName
1	Shivcharan	Das
2	Krish	Naik
3	Rohit	Kapoor
4	Ekta	Dubey
5	Aman	Gupta
6	Sudhanshu	Kumar

Q11. Using the same "Employees" table, write an SQL query to retrieve the first name, last name, and age of employees whose age is between 20 and 30.

Query:

```

SELECT FirstName, LastName, Age
FROM Employees
WHERE Age BETWEEN 20 and 30;
  
```

Execution:

The screenshot shows a Visual Studio Code interface with several windows open. On the left, the Explorer sidebar shows a database connection to 'growdataskills 8.3.0' containing schemas like 'sakila' and 'sql_class_2_assignment'. In the center, a code editor window titled 'Sql_Class_2_Assignment.sql' contains the following SQL query:

```

154 -- SQL query to retrieve the first name
155 -- and last name of employees who are older than 30
156 SELECT FirstName, LastName
157 FROM Employees
158 WHERE Age > 30
159 /*
160 Q11. Using the same "Employees" table, write an SQL query to retrieve the f
161 /*
162 /*
163 /*
164 */
165 /*
166 */
167 WHERE Age BETWEEN 20 and 30; 2ms
  
```

A red box highlights the line 'WHERE Age BETWEEN 20 and 30;'. Below the code editor is a preview pane for 'READMEmd' showing the results of the query:

	FirstName	LastName	Age
1	Shalija	Mishra	23
2	Hariharan	S	25
3	Arpit	Dubey	26
4	Shreya	Richharia	27

A green curly brace groups the rows from 1 to 4, with the handwritten note 'Age b/w 20 and 30' written next to it.

Q12. Given a table named "Products" with columns: ProductID, ProductName, Price, and InStock (0- for out of stock, 1- for in stock). Write an SQL query to retrieve the product names and prices of products that are either priced above \$100 or are out of stock

- Create a table named "Products"

Query:

```

CREATE TABLE Products(
    ProductID INT,
    ProductName VARCHAR(80),
    Price DECIMAL(10, 2),
    InStock INT
);
  
```

Execution:

The screenshot shows a Visual Studio Code interface with several panes:

- Explorer:** Shows a tree view of the project structure, including a folder named "products" which is highlighted with a red box.
- Editor:** Displays a SQL file named "Sql_Class_2_Assignment.sql". A specific section of the code is highlighted with a red box:


```

CREATE TABLE Products(
    ProductID INT,
    ProductName VARCHAR(80),
    Price DECIMAL(10, 2),
    InStock INT
);
      
```
- Output:** Shows the result of running the code, indicating 26ms execution time and 0 affected rows.
- SQL Explorer:** Shows a table named "products" with columns: ProductID, ProductName, Price, and InStock. The table has 0 rows.
- Terminal:** Shows the command "You, 12 seconds ago" and other terminal logs.

- Inserting some random values to the table

Query:

```

INSERT INTO Products (ProductID, ProductName, Price, InStock)
VALUES
(1, 'Widget A', 120.00, 1),
(2, 'Gizmo X', 80.50, 0),
(3, 'Thingamajig', 150.00, 1),
(4, 'Doohickey', 90.00, 1),
(5, 'Whatchamacallit', 110.00, 0),
(6, 'Widget B', 95.00, 1),
(7, 'Gadget Y', 200.00, 0),
(8, 'Contraption Z', 180.00, 1),
(9, 'Doodad', 75.00, 0),
(10, 'Device Q', 105.00, 1);
      
```

Execution:

The screenshot shows two instances of the SQL Class_2_Assignment.sql file in VS Code. The left instance has a red box around the products table in the Explorer sidebar and the SQL code. The right instance has a red box around the preview window showing the execution results.

```

184 -- Inserting some random values
185 INSERT INTO Products (ProductID, ProductName, Price, InStock)
VALUES
(1, 'Widget A', 120.00, 1),
(2, 'Gizmo X', 80.50, 0),
(3, 'Thingamajig', 150.00, 1),
(4, 'Doohickey', 90.00, 1),
(5, 'Whatchamacallit', 110.00, 0),
(6, 'Widget B', 95.00, 1),
(7, 'Gadget Y', 200.00, 0),
(8, 'Contraption Z', 180.00, 1),
(9, 'Doodad', 75.00, 0),
(10, 'Device Q', 105.00, 1); 6ms

```

Execution:

```

INSERT INTO Products (ProductID, ProductName, Price, InStock) VALUES (1, 'Widget A', 120.00, 1), (2, 'Gizmo X', 80.50, 0), (3, 'Thingamajig', 150.00, 1), (4, 'Doohickey', 90.00, 1), (5, 'Whatchamacallit', 110.00, 0), (6, 'Widget B', 95.00, 1), (7, 'Gadget Y', 200.00, 0), (8, 'Contraption Z', 180.00, 1), (9, 'Doodad', 75.00, 0), (10, 'Device Q', 105.00, 1)

AffectedRows : 10

```

The right instance also shows a table view of the products data.

ProductID	ProductName	Price	InStock
1	Widget A	120.00	1
2	Gizmo X	80.50	0
3	Thingamajig	150.00	1
4	Doohickey	90.00	1
5	Whatchamacallit	110.00	0
6	Widget B	95.00	1
7	Gadget Y	200.00	0
8	Contraption Z	180.00	1
9	Doodad	75.00	0
10	Device Q	105.00	1

- SQL query to retrieve the product names and prices of products that are either priced above \$100 or are out of stock

Query:

```

SELECT ProductName, Price
FROM Products
WHERE Price > 100 OR InStock = 0;

```

Execution:

The screenshot shows the Visual Studio Code interface with several tabs open:

- EXPLORER**: Shows the file structure, including a database named "growdataskills" with tables like books, customers, employees, orders, and products.
- Sql_Class_2_Assignment.sql**: The active code editor tab contains an SQL query:


```

2_SQL > 0.Assignment > Solutions > Sql_Class_2_Assignment.sql ...
191 (5, 'Whatchamacallit', 110.00, 0),
192 (6, 'Widget B', 95.00, 1),
193 (7, 'Gadget Y', 200.00, 0),
194 (8, 'Contraption Z', 180.00, 1),
195 (9, 'Doodad', 75.00, 0),
196 (10, 'Device Q', 105.00, 1);

-- SQL query to retrieve the product names and prices of products
-- that are either priced above $100 or are out of stock
200 SELECT ProductName, Price
FROM Products
WHERE Price > 100 OR InStock = 0; 2ms
      
```
- Sql_Class_2_Assignment.sql**: A preview tab showing the results of the query:

	ProductName	Price
1	Widget A	120.00
2	Gizmo X	80.50
3	Thingamajig	150.00
4	Whatchamacallit	110.00
5	Gadget Y	200.00
6	Contraption Z	180.00
7	Doodad	75.00
8	Device Q	105.00
- Preview README.md**: A preview tab showing the contents of the README.md file.

Handwritten annotations on the right side of the screenshot explain the WHERE clause conditions:

- A box labeled "out of stock" points to the row for "Gizmo X" where InStock = 0.
- A box labeled "Price is greater than \$ 100" points to the rows for "Widget A", "Thingamajig", "Whatchamacallit", "Gadget Y", and "Contraption Z" where Price > 100.

Q13. Using the "Products" table, write an SQL query to retrieve the product names and prices of products that are in stock and priced between 50 and 150.

Query:

```

SELECT ProductName, Price
FROM Products
WHERE InStock = 1 AND Price BETWEEN 50 AND 150;
      
```

Execution:

The screenshot shows a code editor interface with several tabs and panes. On the left, there's a sidebar with 'EXPLORER' and 'DATA ANALYST' sections. The main area has a tab for 'Sql_Class_2_Assignment.sql' containing an SQL query:

```
2_SQL > 0.Assignment > Solutions > Sql_Class_2_Assignment.sql > ...
200
201
202 SELECT ProductName, Price
203 FROM Products
204 WHERE Price > 100 OR InStock = 0;      You, 2 seconds ago + Uncommitted changes
205
206 /*
207 Q13. Using the "Products" table, write an SQL query to retrieve the product
208 prices of products that are in stock and priced between 50 and 150.
209 */
210
211
212
```

Below the code editor is a preview pane showing the same SQL query. To the right is an 'Execution' pane with the output:

```
SELECT ProductName, Price
FROM Products
WHERE InStock = 1 AND Price BETWEEN 50 AND 150;
```

At the bottom, there's a table titled 'Products' with columns 'ProductName' and 'Price'. The data is:

	ProductName	Price
1	Widget A	120.00
2	Thingamajig	150.00
3	Doochickey	90.00
4	Widget B	95.00
5	Device Q	105.00

Handwritten annotations in yellow highlight the query in the code editor and the results table. A large bracket groups the results table with the text '0 / P results'. Another bracket groups the 'InStock' column with the text 'in-stocks'. A third bracket groups the 'Price' column with the text 'Prices are b/w 50 & 150'.