

For given tables and query task, give the query statement, execute the query and show the result.

(All questions have to actually execute query and include the execution results or screen shot.)

### Question 1: Show orders (orderId and orderDate) and customers who placed them.

SELECT orders.ORDERID, orders.OrderDate, customers.CustomerName  
FROM orders JOIN customers ON orders.CustomerID = customers.CustomerID;

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
1 -- 1. Show orders (orderId and orderDate) and customers who placed them.
2 SELECT orders.orderID, orders.OrderDate, customers.CustomerName
3 FROM orders JOIN customers ON orders.CustomerID = customers.CustomerID;
```

The results pane displays a table with the following data:

| OrderID | OrderDate | CustomerName           |
|---------|-----------|------------------------|
| 10248   | 7/4/1996  | Wilman Kala            |
| 10249   | 7/5/1996  | Tradição Hipermercados |
| 10250   | 7/8/1996  | Hanari Carnes          |
| 10251   | 7/8/1996  | Victualies en stock    |
| 10252   | 7/9/1996  | Suprêmes délices       |
| 10253   | 7/10/1996 | Hanari Carnes          |
| 10254   | 7/11/1996 | Chop suey Chinese      |

The status bar indicates "10254 rows returned in 34ms".

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
1 -- 1. Show orders (orderId and orderDate) and customers who placed them.
2 SELECT orders.orderID, orders.OrderDate, customers.CustomerName
3 FROM orders JOIN customers ON orders.CustomerID = customers.CustomerID;
```

The results pane displays a table with the following data:

| OrderID | OrderDate | CustomerName                       |
|---------|-----------|------------------------------------|
| 10248   | 7/4/1996  | Consolidated Holdings              |
| 10248   | 7/4/1996  | Drachenblut Delikatessend          |
| 10248   | 7/4/1996  | Du monde entier                    |
| 10248   | 7/4/1996  | Eastern Connection                 |
| 10248   | 7/4/1996  | Ernst Handel                       |
| 10248   | 7/4/1996  | Familia Arquibaldo                 |
| 10248   | 7/4/1996  | FISSA Fabrica Inter. Salchichas... |

The status bar indicates "10248 rows returned in 171ms".

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
1 -- 1. Show orders (orderId and orderDate) and customers who placed them.
2 SELECT orders.orderID, orders.OrderDate, customers.CustomerName
3 FROM orders JOIN customers ON orders.CustomerID = customers.CustomerID;
```

The results pane displays a table with the following data:

| OrderID | OrderDate | CustomerName           |
|---------|-----------|------------------------|
| 10437   | 2/5/1997  | Wartian Herkku         |
| 10438   | 2/6/1997  | Toms Spezialitäten     |
| 10439   | 2/7/1997  | Mère Poularde          |
| 10440   | 2/10/1997 | Savealot Markets       |
| 10441   | 2/10/1997 | Old World Delicatessen |
| 10442   | 2/11/1997 | Ernst Handel           |
| 10443   | 2/12/1997 | Reggiani Caseifici     |

The status bar indicates "10443 rows returned in 34ms".

## Question 2

### Show product and its category.

SELECT products.ProductName, categories.CategoryName, categories.Description  
FROM products JOIN categories ON products.CategoryID = categories.CategoryID;

The screenshot displays two instances of the SQL Developer interface. The top instance shows a query that joins the 'orders' and 'customers' tables, followed by a query that joins the 'products' and 'categories' tables. The bottom instance shows the same 'products' and 'categories' join query, but with a different set of results displayed in the grid.

**Top SQL Query Results:**

| ProductName                     | CategoryName | Description                         |
|---------------------------------|--------------|-------------------------------------|
| Chais                           | Beverages    | Soft drinks, coffees, teas, beer... |
| Chang                           | Beverages    | Soft drinks, coffees, teas, beer... |
| Aniseed Syrup                   | Condiments   | Sweet and savory sauces, ...        |
| Chef Anton's Cajun Seasoning    | Condiments   | Sweet and savory sauces, ...        |
| Chef Anton's Gumbo Mix          | Condiments   | Sweet and savory sauces, ...        |
| Grandma's Boysenberry Spread    | Condiments   | Sweet and savory sauces, ...        |
| Uncle Bob's Organic Dried Pears | Produce      | Dried fruit and bean curd           |

**Bottom SQL Query Results:**

| ProductName                     | CategoryName   | Description                         |
|---------------------------------|----------------|-------------------------------------|
| Flotemysost                     | Dairy_Products | Cheeses                             |
| Mozzarella di Giovanni          | Dairy_Products | Cheeses                             |
| Röd Kaviar                      | Seafood        | Seaweed and fish                    |
| Longlife Tofu                   | Produce        | Dried fruit and bean curd           |
| Rhônebäu Klosterbier            | Beverages      | Soft drinks, coffees, teas, beer... |
| Lakkalikööri                    | Beverages      | Soft drinks, coffees, teas, beer... |
| Original Frankfurter grüne Soße | Condiments     | Sweet and savory sauces, ...        |

### Question 3

Show all customers names , and any orders they might have.

```
SELECT customers.CustomerName, orders.OrderID, orders.OrderDate
FROM customers LEFT JOIN orders ON customers.CustomerID = orders.CustomerID;
```

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
3 FROM orders JOIN customers ON orders.CustomerID = customers.CustomerID;
4 -- 2. Show product and its category
5 SELECT products.ProductName, categories.CategoryName, categories.Description
6 FROM products JOIN categories ON products.CategoryID = categories.CategoryID;
7 -- 3. Show all customers na--mes, and any orders they might have
8 SELECT customers.CustomerName, orders.OrderID, orders.OrderDate
9 FROM customers LEFT JOIN orders ON customers.CustomerID = orders.CustomerID;
```

The results pane displays the following data:

| CustomerName                   | OrderID | OrderDate  |
|--------------------------------|---------|------------|
| Alfreds Futterkiste            | NULL    | NULL       |
| Ana Trujillo Emparedados y ... | 10308   | 9/18/1996  |
| Antonio Moreno Taquería        | 10365   | 11/27/1996 |
| Around the Horn                | 10355   | 11/15/1996 |
| Around the Horn                | 10383   | 12/16/1996 |
| Berglunds snabbköp             | 10278   | 8/12/1996  |
| Berglunds snabbköp             | 10280   | 8/14/1996  |

Execution finished without errors. Result: 213 rows returned in 28ms. At line 7: -- 3. Show all customers na--mes, and any orders they might have SELECT customers.CustomerName, orders.OrderID, orders.OrderDate FROM customers LEFT JOIN orders ON customers.CustomerID = orders.CustomerID;

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the same query as the first screenshot. The results pane displays the following data:

| CustomerName       | OrderID | OrderDate  |
|--------------------|---------|------------|
| Du monde entier    | 10311   | 9/20/1996  |
| Eastern Connection | 10364   | 11/26/1996 |
| Eastern Connection | 10400   | 1/1/1997   |
| Ernst Handel       | 10258   | 7/17/1996  |
| Ernst Handel       | 10263   | 7/23/1996  |
| Ernst Handel       | 10351   | 11/11/1996 |
| Ernst Handel       | 10368   | 11/29/1996 |

Execution finished without errors. Result: 213 rows returned in 28ms. At line 7: -- 3. Show all customers na--mes, and any orders they might have SELECT customers.CustomerName, orders.OrderID, orders.OrderDate FROM customers LEFT JOIN orders ON customers.CustomerID = orders.CustomerID;

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the same query as the previous screenshots. The results pane displays the following data:

| CustomerName           | OrderID | OrderDate |
|------------------------|---------|-----------|
| Wartian Merku          | 10437   | 2/5/1997  |
| Wellington Importadora | 10256   | 7/15/1996 |
| Wellington Importadora | 10420   | 1/21/1997 |
| White Clover Markets   | 10269   | 7/31/1996 |
| White Clover Markets   | 10344   | 11/1/1996 |
| Wilman Kala            | 10248   | 7/4/1996  |
| Wolski                 | 10374   | 12/5/1996 |

Execution finished without errors. Result: 213 rows returned in 28ms. At line 7: -- 3. Show all customers na--mes, and any orders they might have SELECT customers.CustomerName, orders.OrderID, orders.OrderDate FROM customers LEFT JOIN orders ON customers.CustomerID = orders.CustomerID;

## Question 4

Show all employees, and any orders they might have.

```
SELECT employees.FirstName || " " || employees.LastName AS Employee_Name, orders.OrderID,
orders.OrderDate
FROM employees LEFT JOIN orders ON employees.EmployeeID = orders.EmployeeID;
```

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
6 FROM products JOIN categories ON products.CategoryID = categories.CategoryID;
7 -- 3. Show all customers names, and any orders they might have
8 SELECT customers.CustomerName, orders.OrderID, orders.OrderDate
9 FROM customers LEFT JOIN orders ON customers.CustomerID = orders.CustomerID;
10 -- 4. Show all employees, and any orders they might have
11 SELECT employees.FirstName || " " || employees.LastName AS Employee_Name, orders.OrderID, orders.OrderDate
12 FROM employees LEFT JOIN orders ON employees.EmployeeID = orders.EmployeeID;
```

The results pane displays the following data:

|   | Employee_Name | OrderID | OrderDate |
|---|---------------|---------|-----------|
| 1 | Nancy Davolio | 10258   | 7/17/1996 |
| 2 | Nancy Davolio | 10270   | 8/1/1996  |
| 3 | Nancy Davolio | 10275   | 8/7/1996  |
| 4 | Nancy Davolio | 10285   | 8/20/1996 |
| 5 | Nancy Davolio | 10292   | 8/28/1996 |
| 6 | Nancy Davolio | 10293   | 8/29/1996 |
| 7 | Nancy Davolio | 10304   | 9/12/1996 |

Execution finished without errors. Result: 107 rows returned in 35ms. At line 10: -- 4. Show all employees, and any orders they might have. SELECT employees.FirstName || " " || employees.LastName AS Employee\_Name, orders.OrderID, orders.OrderDate FROM employees LEFT JOIN orders ON employees.EmployeeID = orders.EmployeeID;

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
6 FROM products JOIN categories ON products.CategoryID = categories.CategoryID;
7 -- 3. Show all customers names, and any orders they might have
8 SELECT customers.CustomerName, orders.OrderID, orders.OrderDate
9 FROM customers LEFT JOIN orders ON customers.CustomerID = orders.CustomerID;
10 -- 4. Show all employees, and any orders they might have
11 SELECT employees.FirstName || " " || employees.LastName AS Employee_Name, orders.OrderID, orders.OrderDate
12 FROM employees LEFT JOIN orders ON employees.EmployeeID = orders.EmployeeID;
```

The results pane displays the following data:

|     | Employee_Name  | OrderID | OrderDate  |
|-----|----------------|---------|------------|
| 191 | Anne Dodsworth | 10255   | 7/12/1996  |
| 192 | Anne Dodsworth | 10263   | 7/23/1996  |
| 193 | Anne Dodsworth | 10324   | 10/8/1996  |
| 194 | Anne Dodsworth | 10331   | 10/16/1996 |
| 195 | Anne Dodsworth | 10386   | 12/18/1996 |
| 196 | Anne Dodsworth | 10411   | 1/10/1997  |
| 197 | Adam West      | NULL    | NULL       |

Execution finished without errors. Result: 107 rows returned in 35ms. At line 10: -- 4. Show all employees, and any orders they might have. SELECT employees.FirstName || " " || employees.LastName AS Employee\_Name, orders.OrderID, orders.OrderDate FROM employees LEFT JOIN orders ON employees.EmployeeID = orders.EmployeeID;

## Question

Match all customers that are from the same city (duplicate pairs is fine, for example, (customerA, customerB, cityA) and (customerB, customerA, cityA) can be both in result).

```
SELECT c1.CustomerName AS Customer_A, c2.CustomerName AS Customer_B, c1.city
FROM customers AS c1 JOIN customers AS c2 ON c1.City = c2.City
WHERE Customer_A != Customer_B;
```

The screenshot displays the DB Browser for SQLite interface. The main window shows a SQL query being executed, which selects pairs of customers from the same city. The query is as follows:

```
11 SELECT employees.FirstName || " " || employees.LastName AS Employee_Name, orders.OrderID, orders.OrderDate
12 FROM employees LEFT JOIN orders ON employees.EmployeeID = orders.EmployeeID;
13 -- 5. Match all customers that are from the same city (duplicate pairs is fine)
14 SELECT c1.CustomerName AS Customer_A, c2.CustomerName AS Customer_B, c1.City
15 FROM customers AS c1 JOIN customers AS c2 ON c1.City = c2.City
16 WHERE Customer_A != Customer_B;
```

The result of the query is displayed in a table with three columns: Customer\_A, Customer\_B, and City. The results are as follows:

| Customer_A                       | Customer_B                 | City        |
|----------------------------------|----------------------------|-------------|
| 1 Ana Trujillo Emparedados y ... | Centro comercial Moctezuma | México D.F. |
| 2 Ana Trujillo Emparedados y ... | Pericles Comidas clásicas  | México D.F. |
| 3 Ana Trujillo Emparedados y ... | Tortuga Restaurante        | México D.F. |
| 4 Around the Horn                | B's Beverages              | London      |
| 5 Around the Horn                | Consolidated Holdings      | London      |
| 6 Around the Horn                | Eastern Connection         | London      |
| 7 Around the Horn                | North South                | London      |

The interface also shows the DB Schema on the right, listing tables (8), indices (0), views (0), and triggers (0). The status bar at the bottom indicates the connection is to a local database and the encoding is UTF-8.

## Question

Show customers and suppliers that are from the same city.

```
SELECT customers.City, customers.CustomerName, suppliers.SupplierName
FROM customers JOIN suppliers ON customers.City = suppliers.City;
```

The screenshot shows the DB Browser for SQLite interface. The main window displays a SQL query in the 'Execute SQL' tab. The query is as follows:

```
14 SELECT c1.CustomerName AS Customer_A, c2.CustomerName AS Customer_B, c1.City
15 FROM customers AS c1 JOIN customers AS c2 ON c1.City = c2.City
16 WHERE Customer_A != Customer_B;
17 -- 6. Show customers and suppliers that are from the same city
18 SELECT customers.City, customers.CustomerName, suppliers.SupplierName
19 FROM customers JOIN suppliers ON customers.City = suppliers.City;
```

Below the query, the results of the execution are displayed in a table with 3 columns: City, CustomerName, and SupplierName. The results are as follows:

|   | City      | CustomerName           | SupplierName                |
|---|-----------|------------------------|-----------------------------|
| 1 | Berlin    | Alfreds Futterkiste    | Heli Süßwaren GmbH & Co. KG |
| 2 | São Paulo | Comércio Mineiro       | Refrescos Americanas LTDA   |
| 3 | São Paulo | Familia Arquibaldo     | Refrescos Americanas LTDA   |
| 4 | Montréal  | Mère Paillard          | Ma Maison                   |
| 5 | Paris     | Paris spécialités      | Aux joyeux ecclésiastiques  |
| 6 | São Paulo | Queen Cozinha          | Refrescos Americanas LTDA   |
| 7 | Paris     | Spécialités du monde   | Aux joyeux ecclésiastiques  |
| 8 | São Paulo | Tradição Hipermercados | Refrescos Americanas LTDA   |

Below the table, the execution status is shown: "Execution finished without errors. Result: 8 rows returned in 16ms. At line 17: -- 6. Show customers and suppliers that are from the same city SELECT customers.City, customers.CustomerName, suppliers.SupplierName FROM customers JOIN suppliers ON customers.City = suppliers.City;".

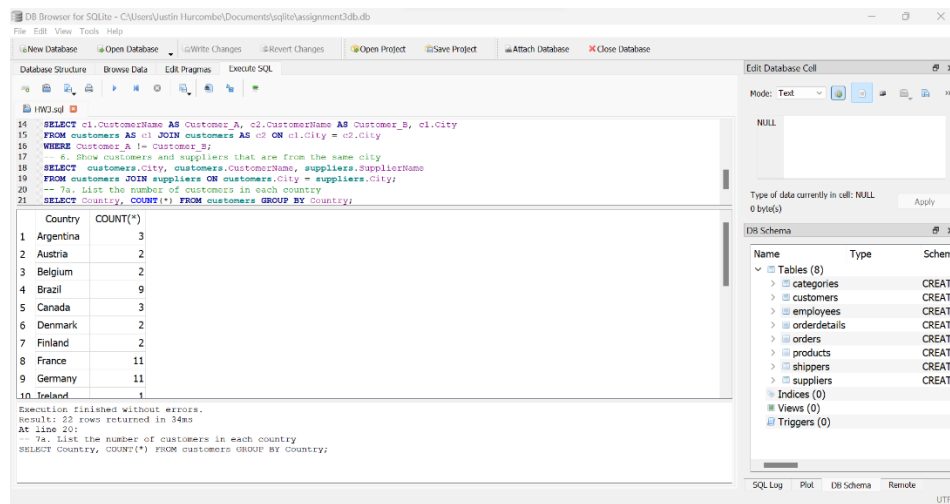
On the right side, the 'Edit Database Cell' window shows 'Mode: Text' and 'Type of data currently in cell: NULL'. Below it, the 'DB Schema' window shows a list of tables (8) and their types (CREATE).

At the bottom, the 'SQL Log' tab is selected, showing the executed SQL query.

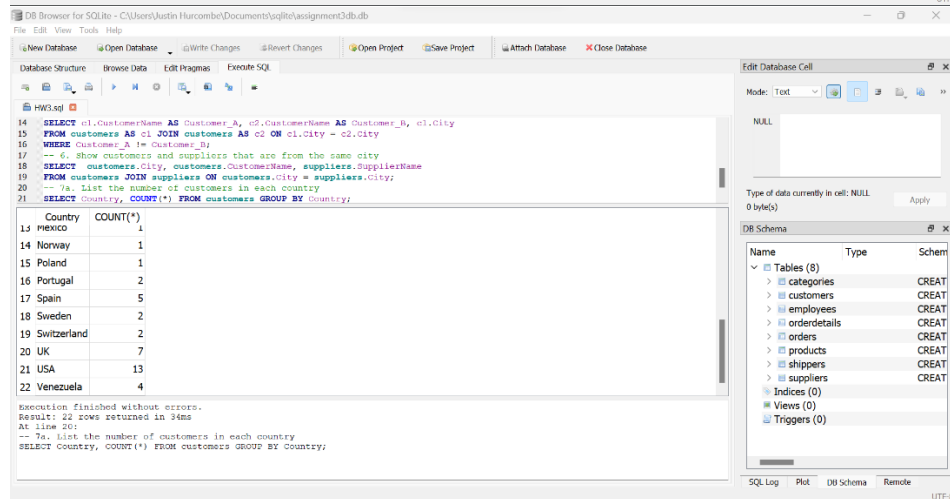
## Question

### a.) List the number of customers in each country.

SELECT Country, COUNT(\*) FROM customers GROUP BY Country;



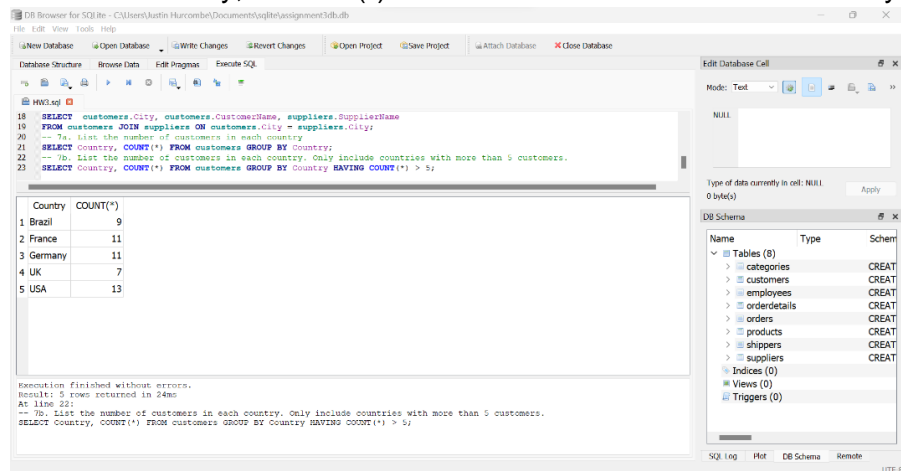
| Country     | COUNT(*) |
|-------------|----------|
| 1 Argentina | 3        |
| 2 Austria   | 2        |
| 3 Belgium   | 2        |
| 4 Brazil    | 9        |
| 5 Canada    | 3        |
| 6 Denmark   | 2        |
| 7 Finland   | 2        |
| 8 France    | 11       |
| 9 Germany   | 11       |
| 10 Ireland  | 1        |



| Country        | COUNT(*) |
|----------------|----------|
| 13 Mexico      | 1        |
| 14 Norway      | 1        |
| 15 Poland      | 1        |
| 16 Portugal    | 2        |
| 17 Spain       | 5        |
| 18 Sweden      | 2        |
| 19 Switzerland | 2        |
| 20 UK          | 7        |
| 21 USA         | 13       |
| 22 Venezuela   | 4        |

### b.) List the number of customers in each country. Only include countries with more than 5 customers.

SELECT Country, COUNT(\*) FROM customers GROUP BY Country HAVING COUNT(\*) > 5;



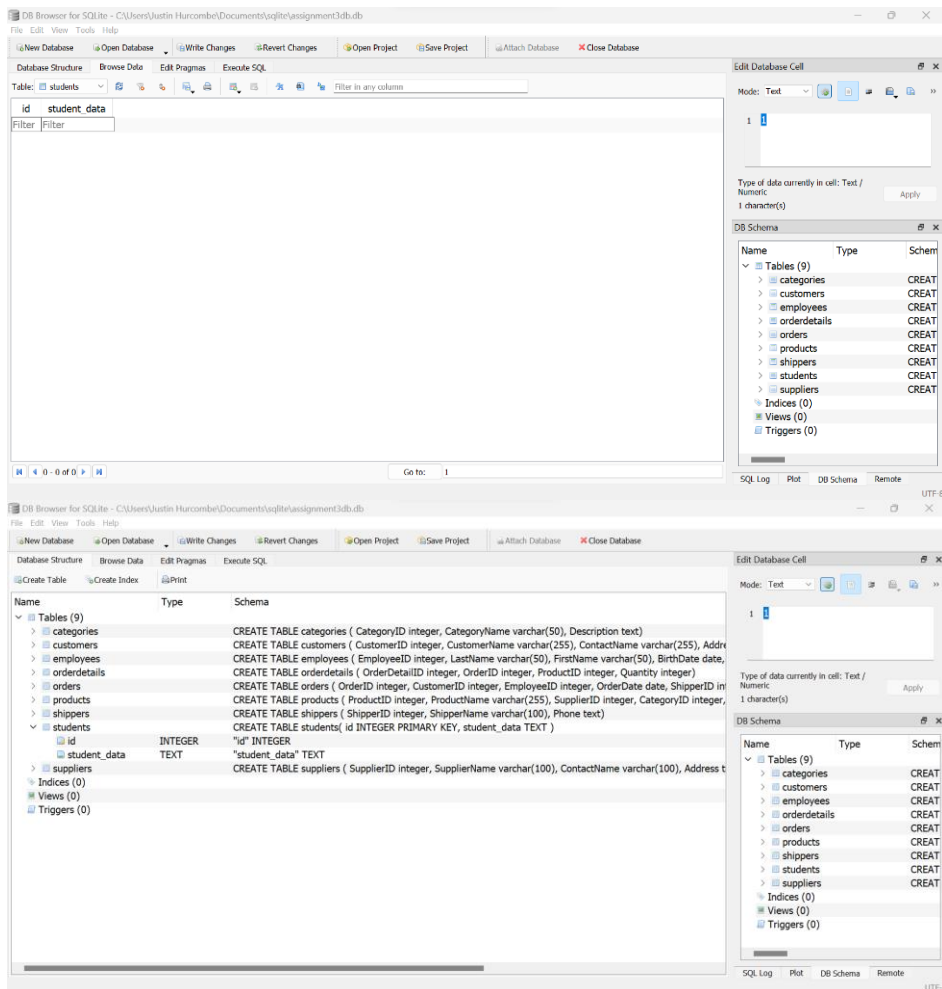
| Country   | COUNT(*) |
|-----------|----------|
| 1 Brazil  | 9        |
| 2 France  | 11       |
| 3 Germany | 11       |
| 4 UK      | 7        |
| 5 USA     | 13       |

## Question

### Question 8

Create table in your database

**CREATE TABLE students(  
    Id INTEGER PRIMARY KEY,  
    Student\_data TEXT  
);**



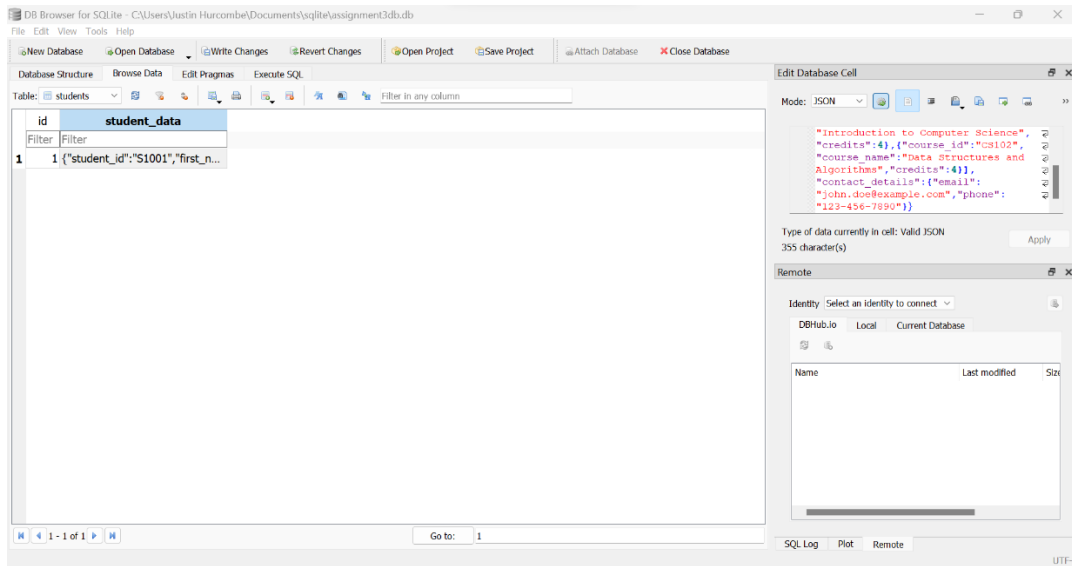


## Question

Finish the following tasks (use SQLITE, if you are able to other database, it is also ok)

a.) Insert your JSON object into this table

```
INSERT INTO students(student_data) VALUES(JSON(
'{
"student_id": "S1001", "first_name": "John", "last_name": "Doe", "age": 20,
"major": "Computer Science", "enrolled_courses": [
{
"course_id": "CS101",
"course_name": "Introduction to Computer Science", "credits": 4
},
{
"course_id": "CS102",
"course_name": "Data Structures and Algorithms", "credits": 4
}
],
"contact_details": {
"email": "john.doe@example.com", "phone": "123-456-7890"
}
}'));
```



## Question

### b.) Show the JSON object in the table

SELECT json\_valid(student\_data) FROM students

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following code:

```
26 10 INTEGER PRIMARY KEY,  
27 student_data TEXT  
28 );  
29 -- 6a. Insert your JSON object into this table.  
30 INSERT INTO students(student_data) VALUES (JSON(  
31 '{  
32   "student_id": "S1001", "first_name": "John", "last_name": "Doe", "age": 20,  
33   "major": "Computer Science", "enrolled_courses": [  
34     {  
35       "course_id": "CS101",  
36       "course_name": "Introduction to Computer Science", "credits": 4  
37     },  
38     {  
39       "course_id": "CS102",  
40       "course_name": "Data Structures and Algorithms", "credits": 4  
41     }  
42   ],  
43   "contact_details": {  
44     "email": "john.doe@example.com", "phone": "123-456-7890"  
45   }  
46 }'  
47 ));  
48 -- 6b. Show the JSON object in the table.  
49 SELECT json_valid(student_data) FROM students;
```

The results pane shows a single row with the value 1 in the column json\_valid(student\_data).

Execution finished without errors.  
Result: 1 rows returned in 39ms  
At line 47:  
-- 6b. Show the JSON object in the table.  
SELECT json\_valid(student\_data) FROM students;

### c.) Retrieve all students first\_name and email and phone number

SELECT json\_extract(student\_data, '\$.first\_name') AS first\_name,  
json\_extract(student\_data, '\$.contact\_details.email') AS email,  
json\_extract(student\_data, '\$.contact\_details.phone') AS phone  
FROM students;

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following code:

```
41 }  
42 }  
43 "contact_details": {  
44   "email": "john.doe@example.com", "phone": "123-456-7890"  
45 }  
46 }  
47 }  
48 -- 6b. Show the JSON object in the table.  
49 SELECT json_valid(student_data) FROM students;  
50 -- 6c. Retrieve all students first_name and email and phone number  
51 SELECT json_extract(student_data, '$.first_name') AS first_name,  
52 json_extract(student_data, '$.contact_details.email') AS email,  
53 json_extract(student_data, '$.contact_details.phone') AS phone  
54 FROM students;
```

The results pane shows a single row with the following data:

| first_name | email                | phone        |
|------------|----------------------|--------------|
| John       | john.doe@example.com | 123-456-7890 |

Execution finished without errors.  
Result: 1 rows returned in 28ms  
At line 50:  
-- 6c. Retrieve all students first\_name and email and phone number  
SELECT json\_extract(student\_data, '\$.first\_name') AS first\_name,

## Question

### d.) Extract all “computer science students”

```
SELECT json_extract(student_data, '$.first_name') AS first_name,  
json_extract(student_data, '$.contact_details.email') AS email,  
json_extract(student_data, '$.contact_details.phone') AS phone,  
json_extract(student_data, '$.major') AS major  
FROM students WHERE major = "Computer Science";
```

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
47 ();  
48 -- 8b. show the json object in the table.  
49 SELECT json_valid(student_data) FROM students;  
50 -- 8c. Retrieve all students first_name and email and phone number  
51 SELECT json_extract(student_data, '$.first_name') AS first_name,  
52 json_extract(student_data, '$.contact_details.email') AS email,  
53 json_extract(student_data, '$.contact_details.phone') AS phone  
54 FROM students;  
55 -- 8d. Extract all "computer science" students.  
56 SELECT json_extract(student_data, '$.first_name') AS first_name,  
57 json_extract(student_data, '$.contact_details.email') AS email,  
58 json_extract(student_data, '$.contact_details.phone') AS phone,  
59 json_extract(student_data, '$.major') AS major  
60 FROM students WHERE major = "Computer Science";  
61
```

The results pane shows the following data:

| first_name | email                | phone        | major            |
|------------|----------------------|--------------|------------------|
| 1 John     | john.doe@example.com | 123-456-7890 | Computer Science |

Execution finished without errors.  
Result: 1 rows returned in 22ms  
At line 55:  
-- 8d. Extract all "computer science" students.  
SELECT json\_extract(student\_data, '\$.first\_name') AS first\_name,

### e.) Display list of courses for each student

```
SELECT  
students.id,  
json_extract(student_data, '$.first_name') AS first_name,  
json_extract(student_data, '$.last_name') AS last_name,  
json_extract(value, '$.course_id') AS course_id,  
json_extract(value, '$.course_name') AS course_name,  
json_extract(value, '$.credits') AS credits  
FROM students, json_each(students.student_data, '$.enrolled_courses');
```

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
56 SELECT json_extract(student_data, '$.first_name') AS first_name,  
57 json_extract(student_data, '$.contact_details.email') AS email,  
58 json_extract(student_data, '$.contact_details.phone') AS phone,  
59 json_extract(student_data, '$.major') AS major  
60 FROM students WHERE major = "Computer Science";  
61 -- 8e. Display list of courses for each student.  
62 SELECT  
63 students.id,  
64 json_extract(student_data, '$.first_name') AS first_name,  
65 json_extract(student_data, '$.last_name') AS last_name,  
66 json_extract(value, '$.course_id') AS course_id,  
67 json_extract(value, '$.course_name') AS course_name,  
68 json_extract(value, '$.credits') AS credits  
69 FROM students, json_each(students.student_data, '$.enrolled_courses');  
70
```

The results pane shows the following data:

| id | first_name | last_name | course_id | course_name                    | credits |
|----|------------|-----------|-----------|--------------------------------|---------|
| 1  | 1 John     | Doe       | CS101     | Introduction to Computer ...   | 4       |
| 2  | 1 John     | Doe       | CS102     | Data Structures and Algorithms | 4       |

Execution finished without errors.  
Result: 2 rows returned in 32ms  
At line 61:  
-- 8e. Display list of courses for each student.  
SELECT

## Question

### f.) Display first course for all students

```
SELECT
students.id,
json_extract(student_data, '$.first_name') AS first_name,
json_extract(student_data, '$.last_name') AS last_name,
json_extract(student_data, '$.enrolled_courses[0].course_id') AS course_id,
json_extract(student_data, '$.enrolled_courses[0].course_name') AS course_name,
json_extract(student_data, '$.enrolled_courses[0].credits') AS credits
FROM students;
```

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following code:

```
64 json_extract(student_data, '$.first_name') AS first_name,
65 json_extract(student_data, '$.last_name') AS last_name,
66 json_extract(value, '$.course_id') AS course_id,
67 json_extract(value, '$.course_name') AS course_name,
68 json_extract(value, '$.credits') AS credits
69 FROM students, json_each(students.student_data, '$.enrolled_courses');
70 -- 8f. Display first course for all students
71 SELECT
72 students.id,
73 json_extract(student_data, '$.first_name') AS first_name,
74 json_extract(student_data, '$.last_name') AS last_name,
75 json_extract(student_data, '$.enrolled_courses[0].course_id') AS course_id,
76 json_extract(student_data, '$.enrolled_courses[0].course_name') AS course_name,
77 json_extract(student_data, '$.enrolled_courses[0].credits') AS credits
78 FROM students;
```

The results pane displays a single row of data:

| id | first_name | last_name | course_id | course_name                  | credits |
|----|------------|-----------|-----------|------------------------------|---------|
| 1  | John       | Doe       | CS101     | Introduction to Computer ... | 4       |

Execution finished without errors.  
Result: 1 rows returned in 26ms  
At line 70:  
-- 8f. Display first course for all students  
SELECT