

# Lab 12: SQL **lab12.zip (lab12.zip)**

---

*Due by 11:59pm on Tuesday, November 17.*

## Starter Files

Download lab12.zip (lab12.zip). Inside the archive, you will find starter files for the questions in this lab, along with a copy of the Ok (ok) autograder.

## Topics

---

Consult this section if you need a refresher on the material for this lab, or if you're having trouble running SQL or SQLite on your computer. It's okay to skip directly to the questions and refer back here should you get stuck.

[SQL Basics](#)[Joins](#)[Troubleshooting](#)[SQLite Usage](#)

## The Survey Data!

---

On Tuesday, we asked you and your fellow students to complete a brief online survey through Google Forms, which involved relatively random but fun questions. In this lab, we will interact with the results of the survey by using SQL queries to see if we can find interesting things in the data.

First, take a look at `data.sql` and examine the table defined in it. Note its structure. You will be working with:

- `students` : The main results of the survey. Each column represents a different question from the survey, except for the first column, which is the time of when the result was submitted. This time is a unique identifier for each of the rows in the table.

Column Name	Question
time	The unique timestamp that identifies the submission
number	What's your favorite number between 1 and 100?
color	What is your favorite color?
seven	Choose the number 7 below. Options: <ul style="list-style-type: none"> <li>◦ 7</li> <li>◦ Choose this option instead</li> <li>◦ seven</li> <li>◦ the number 7 below.</li> <li>◦ I find this question condescending</li> </ul>
song	If you could listen to only one of these songs for the rest of your life, which would it be? Options: <ul style="list-style-type: none"> <li>◦ "Everytime we Touch" by Cascada</li> <li>◦ "The Middle" by Zedd</li> <li>◦ "Clair de Lune" by Claude Debussy</li> <li>◦ "Smells Like Teen Spirit" by Nirvana</li> <li>◦ "Dancing Queen" by ABBA</li> <li>◦ "All I Want for Christmas is you" by Mariah Carey</li> <li>◦ "Down With The Sick" by Disturbed</li> <li>◦ "thank u, next" by Ariana Grande</li> </ul>
date	Pick a day of the year!
pet	If you could have any animal in the world as a pet, what would it be?
instructor	Choose your favorite photo of John DeNero (Options shown under Question 3)
smallest	Try to guess the smallest unique positive INTEGER that anyone will put!

- numbers : The results from the survey in which students could select more than one option from the numbers listed, which ranged from 0 to 10 and included 2018, 9000, and 9001. Each row has a time (which is again a unique identifier) and has the value 'True' if the student selected the column or 'False' if the student did not. The column names in this table are the following strings, referring to each possible number: '0', '1', '2', '4', '5', '6', '7', '8', '9', '10', '2018', '9000', '9001'.

Since the survey was anonymous, we used the timestamp that a survey was submitted as a unique identifier. A time in `students` matches up with a time in `numbers`. For example, a row in `students` whose time value is "2019/08/06 4:19:18 PM CDT" matches up with the row in

numbers whose time value is "2019/08/06 4:19:18 PM CDT". These entries come from the same Google form submission and thus belong to the same student.

*Note:* If you are looking for your personal response within the data, you may have noticed that some of your answers are slightly different from what you had inputted. In order to make SQLite accept our data, and to optimize for as many matches as possible during our joins, we did the following things to clean up the data:

- `color` and `pet`: We converted all the strings to be completely lowercase.
- For some of the more "free-spirited" responses, we escaped the special characters so that they could be properly parsed.

You will write all of your solutions in the starter file `lab12.sql` provided. As with other labs, you can test your solutions with OK. In addition, you can use either of the following commands:

```
python3 sqlite_shell.py < lab12.sql
python3 sqlite_shell.py --init lab12.sql
```

## Questions

### Q1: What Would SQL print?

Note: there is no submission for this question

First, load the tables into sqlite3.

```
$ python3 sqlite_shell.py --init lab12.sql
```

Before we start, inspect the schema of the tables that we've created for you:

```
sqlite> .schema
```

This tells you the name of each of our tables and their attributes.

Let's also take a look at some of the entries in our table. There are a lot of entries though, so let's just output the first 20:

```
sqlite> SELECT * FROM students LIMIT 20;
```

If you're curious about some of the answers students put into the Google form, open up `data.sql` in your favorite text editor and take a look!

For each of the SQL queries below, think about what the query is looking for, then try running the query yourself and see!

```
sqlite> SELECT * FROM students LIMIT 30; -- This is a comment. * is shorthand for all colu
-----

sqlite> SELECT color FROM students WHERE number = 7;
-----

sqlite> SELECT song, pet FROM students WHERE color = "blue" AND date = "12-25";
-----
```

Remember to end each statement with a `;` ! To exit out of SQLite, type `.exit` or `.quit` or hit `Ctrl-C`.

## Q2: Go Bears! (And Dogs?)

Now that we have learned how to select columns from a SQL table, let's filter the results to see some more interesting results!

It turns out that 61A students have a lot of school spirit: the most popular favorite color was 'blue'. You would think that this school spirit would carry over to the pet answer, and everyone would want a pet bear! Unfortunately, this was not the case, and the majority of students opted to have a pet 'dog' instead. That is the more sensible choice, I suppose...

Write a SQL query to create a table that contains both the column `color` and the column `pet`, using the keyword `WHERE` to restrict the answers to the most popular results of color being 'blue' and pet being 'dog'.

You should get the following output:

[illegible]

```
CREATE TABLE blue dog AS
SELECT "REPLACE THIS LINE WITH YOUR SOLUTION";
```

This isn't a very exciting table, though. Each of these rows represents a different student, but all this table can really tell us is how many students both like the color blue and want a dog as a pet, because we didn't select for any other identifying characteristics. Let's create another table, `bluedog_songs`, that looks just like `bluedog` but also tells us how each student answered the `song` question.

You should get the following output:

```
sqlite> SELECT * FROM blue_dog_songs;  
blue|dog|Smells like Teen Spirit  
blue|dog|The Middle  
blue|dog|Clair De Lune  
blue|dog|Smells like Teen Spirit  
blue|dog|Everytime We Touch  
blue|dog|Dancing Queen  
blue|dog|Clair De Lune  
blue|dog|Clair De Lune  
blue|dog|The Middle  
blue|dog|Down With The Sickness  
blue|dog|thank u, next  
blue|dog|All I want for Christmas is you  
blue|dog|Everytime We Touch  
blue|dog|Clair De Lune  
blue|dog|Everytime We Touch  
blue|dog|thank u, next  
blue|dog|All I want for Christmas is you  
blue|dog|Smells like Teen Spirit  
blue|dog|The Middle  
blue|dog|Clair De Lune  
blue|dog|Dancing Queen  
blue|dog|All I want for Christmas is you  
blue|dog|Dancing Queen  
blue|dog|Clair De Lune  
blue|dog|Dancing Queen  
blue|dog|Dancing Queen
```

```
CREATE TABLE blue_dog_songs AS  
SELECT "REPLACE THIS LINE WITH YOUR SOLUTION";
```

This distribution of songs actually largely represents the distribution of song choices that the total group of students made, so perhaps all we've learned here is that there isn't a correlation between a student's favorite color and desired pet, and what song they could spend the rest of their life listening to. Even demonstrating that there is no correlation still reveals facts about our data though!

Use Ok to test your code:

```
python3 ok -q blue_dog
```

### Q3: The Smallest Unique Positive Integer

Who successfully managed to guess the smallest unique positive integer value? Let's find out!

While we could find out the smallest unique integer using aggregation, for now let's just try hand-inspecting the data. An anonymous elf has informed us that the smallest unique positive value is greater than 2.

Write an SQL query to create a table with the columns `time` and `smallest` that we can inspect to determine what the smallest integer value is. In order to make it easier for us to inspect these values, use `WHERE` to restrict the answers to numbers greater than 2, `ORDER BY` to sort the numerical values, and `LIMIT` your result to the first 20 values that are greater than the number 2.

**Note:** Unfortunately, the smallest unique integer doesn't exist in the first 20 values greater than 2. For fun, you can try some new SQL queries (for example, looking at values larger than 10) to see if you can find the smallest unique integer!

```
CREATE TABLE smallest_int AS
SELECT "REPLACE THIS LINE WITH YOUR SOLUTION";
```

Use Ok to test your code:

```
python3 ok -q smallest-int
```

After you've successfully passed the Ok test, take a look at the table `smallest_int` that you just created and manually find the smallest unique integer value!

To do this, try the following:

```
$ python3 sqlite_shell.py --init lab12.sql
sqlite> SELECT * FROM smallest_int; -- No LIMIT this time!
```

## Q4: Matchmaker, Matchmaker

Did you take 61A with the hope of finding your quarantine romance? Well you're in luck! With all this data in hand, it's easy for us to find your perfect match. If two students want the same pet and have the same taste in music, they are clearly meant to be together! In order to provide some more information for the potential lovebirds to converse about, let's include the favorite colors of the two individuals as well!

In order to match up students, you will have to do a join on the `students` table with itself. When you do a join, SQLite will match every single row with every single other row, so make sure you do not match anyone with themselves, or match any given pair twice!

**Important Note:** When pairing the first and second person, make sure that the first person responded first (i.e. they have an earlier `time`). This is to ensure your output matches our tests.

*Hint:* When joining table names where column names are the same, use dot notation to distinguish which columns are from which table: `[table_name].[column name]`. This sometimes may get verbose, so it's stylistically better to give tables an alias using the `AS` keyword. The syntax for this is as follows:

```
SELECT <[alias1].[column name1], [alias2].[columnname2]...>
      FROM <[table_name1] AS [alias1],[table_name2] AS [alias2]...> ...
```

The query in the football example from earlier uses this syntax.

Write a SQL query to create a table that has 4 columns:

- The shared preferred pet of the couple
- The shared favorite song of the couple
- The favorite color of the first person
- The favorite color of the second person

```
CREATE TABLE matchmaker AS
SELECT "REPLACE THIS LINE WITH YOUR SOLUTION";
```

Use Ok to test your code:

```
python3 ok -q matchmaker
```

## Q5: Sevens

Let's take a look at data from both of our tables, `students` and `numbers`, to find out if students that really like the number 7 also chose '7' for the obedience question. Specifically, we want to look at the students that fulfill the below conditions and see if they also chose '7' in the question that asked students to choose the number 7 (column `seven` in `students`).

Conditions:

- reported that their favorite number (column `number` in `students`) was 7
- have 'True' in column '7' in `numbers`, meaning they checked the number 7 during the survey

In order to examine rows from both the `students` and the `numbers` table, we will need to perform a join.

How would you specify the `WHERE` clause to make the `SELECT` statement only consider rows in the joined table whose values all correspond to the same student? If you find that your output is massive and overwhelming, then you are probably missing the necessary condition in your `WHERE` clause to ensure this.



*Note:* The columns in the `numbers` table are strings with the associated number, so you must put quotes around the column name to refer to it. For example if you alias the table as `a`, to get the column to see if a student checked 9001, you must write `a.'9001'`.

**Write a SQL query to create a table with just the column `seven` from `students`, filtering first for students who said their favorite number (column `number`) was 7 in the `students` table and who checked the box for seven (column `'7'`) in the `numbers` table.**

```
CREATE TABLE sevens AS  
SELECT "REPLACE THIS LINE WITH YOUR SOLUTION";
```

Use Ok to test your code:

```
python3 ok -q sevens
```

## **CS 61A (/~cs61a/fa20/)**

[Weekly Schedule \(/~cs61a/fa20/weekly.html\)](/~cs61a/fa20/weekly.html)

[Office Hours \(/~cs61a/fa20/office-hours.html\)](/~cs61a/fa20/office-hours.html)

[Staff \(/~cs61a/fa20/staff.html\)](/~cs61a/fa20/staff.html)

## **Resources (/~cs61a/fa20/resources.html)**

[Studying Guide \(/~cs61a/fa20/articles/studying.html\)](/~cs61a/fa20/articles/studying.html)

[Debugging Guide \(/~cs61a/fa20/articles/debugging.html\)](/~cs61a/fa20/articles/debugging.html)

[Composition Guide \(/~cs61a/fa20/articles/composition.html\)](/~cs61a/fa20/articles/composition.html)

## **Policies (/~cs61a/fa20/articles/about.html)**

[Assignments \(/~cs61a/fa20/articles/about.html#assignments\)](/~cs61a/fa20/articles/about.html#assignments)

[Exams \(/~cs61a/fa20/articles/about.html#exams\)](/~cs61a/fa20/articles/about.html#exams)

[Grading \(/~cs61a/fa20/articles/about.html#grading\)](/~cs61a/fa20/articles/about.html#grading)