

## Lab 2

---

## Table of Contents

<b>Overview .....</b>	<b>3</b>
<b>Create some example table schema.....</b>	<b>3</b>
<b>Customer Table (a table to hold some data).....</b>	<b>3</b>
Task: Customers .....	3
<b>Task: SQL scripts .....</b>	<b>4</b>
Text editors (get your workflow well organised).....	5
Conditional table creation .....	6

## Overview

In this lab, continuing on the command line in sqlite3, you will:

- Create some tables, based on a *schema*, and populate them with some data
- Populate the tables with will data

## Create some example table schema

Following on from last week, from sqlite3, create 3 tables based on the following schema. Refer back to last week lab to see the general structure for a table, but don't just rush ahead and code; think about the **data types** you are using to represent your *attributes*.

### PersonTable

*/ Name / CreditCard / Email / Age / Phone / Address /*

### Address

*/ Postcode / House Number / Street /*

### Company

*/ Name / RegCode /*

Once you have created the tables, ensure that they are in lab2.db (you will have used this as an argument to sqlite3, remember, when you entered the sqlite shell... if you forgot to do this [or if you forgot to cd into the '2' folder], exit from sqlite3:

**<.exit>**

... and start again by referring to the instructions in the previous lab. If you rush ahead and don't follow the instructions carefully, you are putting yourself at a disadvantage. So, before you create anything, make sure you cd to folder '2', and call sqlite3 from there with the lab2.db argument (a useful think to always do is **<.databases>** to force the lab2.db to be written to disk...if you do that and see it in folder '2', your ready to go forwards...

## Customer Table (a table to hold some data)

When we model data in a relational way, we know that we will have tabular data, which is determined by the schema of our design. Therefore, when we create tables in a database, we will do so with a given schema in mind:

## Task: Customers

Look at Table 1. It contains data according to a schema (first row).

id	name	address	telno	email
NW001	DISKS 'N STUFF	5 MAIN ROAD, NEWTOWN, ST56 34ER	0142-432-5768	t.banks@abc.co.uk
NW002	CARRAGHER & SON	235 WAGNER WAY, ROMANVILLE, ST54 6WR	0134-223-5637	joe@bloggs.123.co.uk
NE001	TONY KENNEDY	55 BARNTON ROAD, MEADOW PARK, DUNWICH, D1 3TR	0162-444-1434	
SE001	THE MUSIC SHOP	CULLEN WYND, EASTPORT, FR54 3WX	0122-385-9028	music_shop@eastport.com
SW001	THREADS ETC.	11 CARNABY AVENUE, MAINTOWN, M1 2MM	0111-111-2222	

Table 1: Customer

Create this table schema in sqlite3. Look at the nature of the data in table 1...for example, you will again need to think which data types you will use (the 'TEXT' choices I included below aren't necessarily the best choices, but don't worry about that too much...we will look at how sqlite3 data types work in another part of the course), like you did when you did Programming 1.

```
--
C:\Users\rholden>C:\Apps\sqlite3.exe lab3.db
SQLite version 3.25.2 2018-09-25 19:08:10
Enter ".help" for usage hints.
sqlite> .databases
main: C:\Users\rholden\lab3.db
sqlite> CREATE TABLE Customer(
...> id TEXT,
...> name TEXT,
...> address TEXT,
...> telno TEXT,
...> email TEXT
...> );
sqlite> .tables
Customer
sqlite>
```

At this point you have created the table schema. Now you need to populate the table with data rows in table 1, i.e. the rows that are not filled in with grey background.

Here is some example cmd input/output to help you do this for the first row of data.

```
--
sqlite> INSERT INTO Customer (customer_no, name, address, telno, email )
...> VALUES
...> ("NW001", "DISKS 'N STUFF", "5 MAIN ROAD, NEWTON, ST56 34ER", "0142-432-5768", "t.banks@abc.co.uk");
sqlite> SELECT * FROM Customer;
NW001|DISKS 'N STUFF|5 MAIN ROAD, NEWTON, ST56 34ER|0142-432-5768|t.banks@abc.co.uk
sqlite>
--
```

I know this is a lot of information to type into the CMD/Terminal, but stick with it. Its honestly good practice. Each time you punch something in and it doesn't work, just look carefully at you SQL. Also (you might have noticed already) you will notice that if you enter a line of SQL in sqlite3, if it is not terminated by ';' then the interpreter assumes you want to continue with the command on the following line. In effect, a single command can take up several lines. Be careful to enter each character carefully.

### Task: SQL scripts

You should have noticed, already, that entering data from the command line isn't the most efficient process, but we have learned to get some instant feedback from the CMD/Terminal when we are typing, and that is useful. However, as you get more familiar with SQL, and as the SQL you will use becomes more complex (more SQL lines per table, more tables per

database) it is unreasonable to use the CMD/Terminal to enter SQL code. We will still use the sqlite3 shell, but only to call scripts that have SQL code in them.

### Text editors (get your workflow well organised)

When you are working with a computer generally, it is critical that you are well organised. There can be a strong temptation to rush forward into coding just to *'just get on with it'*. However, if you are not well organised with your workflow/environment, then you are putting yourself at a major disadvantage compared with others who are – they will most likely be able to think more clearly, make less mistakes, trace more easily the mistakes they do make etc. So, you can see that taking the time in the early stages of your degree to be well organised will most likely pay off in the long run.

Please see the content area on GCULearn, named **'Software/Text editors'**. In there you will find some very useful information about how to set up a nice and simple arrangement of your windows (command line, file explorer, text editor) for working with your scripts. Please take your time to follow all the suggestions.

Create a new file called **<load\_customer\_data.sql>** and save it to your current *working directory* (if you have followed previous instructions carefully, you will know what that is, and as a reminder, for this lab it should be the '2' folder in LABS\_HOME). To check where you are: type in a windows CMD (mac instructions are below, too):

**<.system cd>**

...which should show you the path to the current directory (notice what we did there...we called the 'Change Directory' command with no arguments, which just 'changes' the current directory to the directory we are already in! That's a bit like leaving a room to go into another room, which ends up being the same room you were in before. Completely pointless! Its best to think of the 'c' in cd as either 'change' or 'current' depending on how you are using it.

On a mac, use the **'print working directory'** command:

**<.system pwd>**

...at the sqlite3 shell. Notice what's going on here. You are calling the system shell from the sqlite shell, courtesy of sqlite3 'dot' notation **<.system>**. So, if your console output is getting a bit messy, type **<.system cls>** on Windows or on Mac OS **<.system clear>** to tidy it up. Good programmers get into the habit of keeping things nice and tidy!

So, verify that you have saved the load\_customer\_data.sql with the correct extension type:

**<.system DIR \*.sql>** on Windows

**<.system ls \*.sql>** on Mac OS

... you should see the file listed in your CMD/Terminal, of course.

Write the following information into the script:

```
INSERT INTO Customer (customer_no, name, address, telno, email )
VALUES
("NW001", "DISKS 'N STUFF", "5 MAIN ROAD, NEWTON, ST56 34ER", "0142-432-5768", "t.banks@abc.co.uk"),
("NW002", "CARRAGHER & SON", "235 WAGNER WAY, ROMANVILLE, ST54 6WR", "0134-223-5637", "joe@carragher.freemove.co.uk"),
("NE001", "TONY KENNEDY", "55 BARNTON ROAD, MEADOW PARK, DUNWICH, D1 3TR", "0122-385-9028", "music_shop@eastport.com"),
("SE001", "THE MUSIC SHOP", "CULLEN WYND, EASTPORT, FR54 3WX", "0122-385-9028", "music_shop@eastport.com"),
("SW001", "THREADS ETC", "11 CARNABY AVENUE, MAINTOWN", "M1 2MM", "0111-111-2222");
```

Yes, I deliberately included this 'text' as an image, so that you have to write it yourself and arrange it the way it is arranged, yourself, using whitespaces etc. The probability you will make a mistake is high (all those commas, and the semi colon etc.). But this is fine, you need to get used to writing out simple scripts. Tip: keep in mind how the data is *structured* by the use of *commas and brackets*, and how the data itself is enclosed in *double quotes*, rather than over-focusing on the actual values of the data (e.g. *NW001*) at this point.

Once (and while) you type the script **save** it! Always make sure you do that before running a script, otherwise you are just going to call the script saved to disk, which does not contain the information you can see in your editor (until you save it).

...then type:

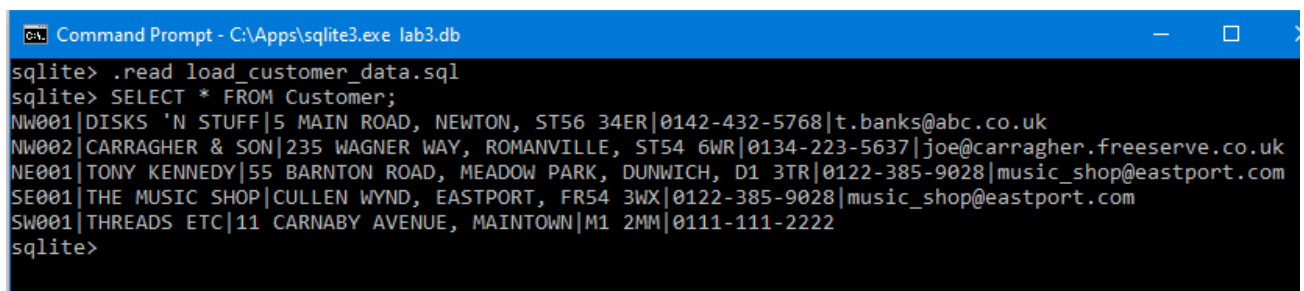
**<.read load\_customer\_data.sql>**

In the sqlite3 shell. If you have copied the above correctly, calling the script will run *the* command in the script; yes, that's just one SQL command, terminated by ';'.

And then you will be able to select all of the data from the Customer table by typing the SQL command/query:

**<SELECT \* FROM Customer;>**

... if you have structured your SQL correctly, then you should see something like this:



```

C:\Apps\sqlite3.exe lab3.db
sqlite> .read load_customer_data.sql
sqlite> SELECT * FROM Customer;
NW001|DISKS 'N STUFF|5 MAIN ROAD, NEWTON, ST56 34ER|0142-432-5768|t.banks@abc.co.uk
NW002|CARRAGHER & SON|235 WAGNER WAY, ROMANVILLE, ST54 6WR|0134-223-5637|joe@carragher.freemove.co.uk
NE001|TONY KENNEDY|55 BARNTON ROAD, MEADOW PARK, DUNWICH, D1 3TR|0122-385-9028|music_shop@eastport.com
SE001|THE MUSIC SHOP|CULLEN WYND, EASTPORT, FR54 3WX|0122-385-9028|music_shop@eastport.com
SW001|THREADS ETC|11 CARNABY AVENUE, MAINTOWN|M1 2MM|0111-111-2222
sqlite>

```

Conditional table creation

Look at the top of the sqlite tutorial here:

<https://www.sqlitetutorial.net/sqlite-drop-table/>

Create a script named `<create_customer.sql>`, which drops a table named Customers (if it exists) before creating it again using the syntax you already have. Then save your script and run the following sqlite3 commands:

<code>&lt;.read create_customer.sql&gt;</code>	<b>1</b>
<code>&lt;SELECT * FROM Customer;&gt;</code>	<b>2</b>
<code>&lt;.read load_customer_data.sql&gt;</code>	<b>3</b>
<code>&lt;SELECT * FROM Customer;&gt;</code>	<b>4</b>

This will:

1. Create a fresh customer table, i.e. create it after “dropping” (deleting) it.
2. Output nothing from the table because there is no data in it.
3. Load data into the table.
4. Display the data that was just loaded.

So, you can see how we are starting to develop higher-level functionality, hidden in scripts, which we can very easily from the sqlite3 shell using the `<.read>` command.

~~~~~