

SYSC 3010 Systems Development Lab

Lab 3: Databases

LAB OBJECTIVES

- Setup and use a small example relational database using SQLite
- Deepen your understanding of JSON messages and test your knowledge integration skills
- [TBD: Experiment with Google sheets as a database]

SYNCHRONOUS VERSUS ASYNCHRONOUS

The TAs will be online for the entire lab period. Instructions will be posted at the time. If you have trouble connecting, please use SLACK to initiate contact.

There will be no signup for this lab. Please contact any of TAs as/when you need them for technical support.

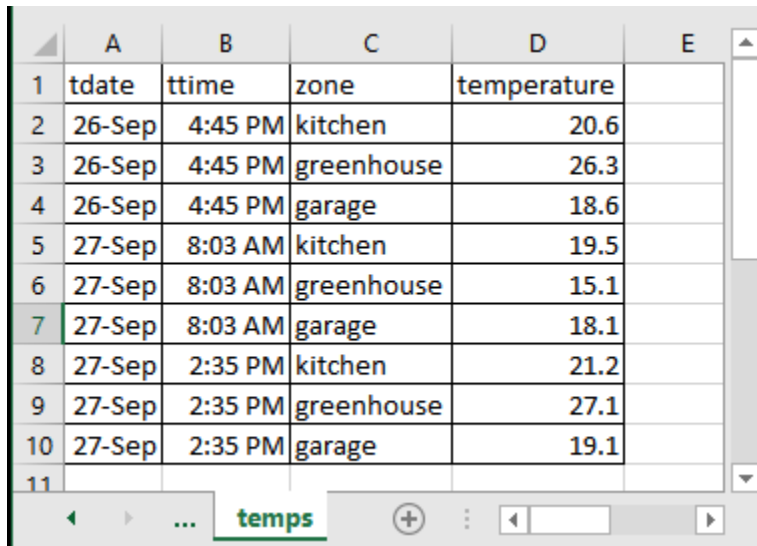
SUBMISSION

There will be no marked submission at this time. During the individual lab exam (Lab 4), a student must be able to perform any task involved in this lab. Additionally, the student's Github repository will be checked to ensure that (1) the content for this lab is present and (2) the timestamp for that content falls within this lab period.

All programs for this lab must be in a GitHub folder called **Lab-3**.

1 – THE SQLITE DATABASE

SQL is a language for relational databases, meaning in simplest terms, that all data are stored in tables made up of rows and columns; just like an Excel spreadsheet.



	A	B	C	D	E
1	tdate	ttime	zone	temperature	
2	26-Sep	4:45 PM	kitchen	20.6	
3	26-Sep	4:45 PM	greenhouse	26.3	
4	26-Sep	4:45 PM	garage	18.6	
5	27-Sep	8:03 AM	kitchen	19.5	
6	27-Sep	8:03 AM	greenhouse	15.1	
7	27-Sep	8:03 AM	garage	18.1	
8	27-Sep	2:35 PM	kitchen	21.2	
9	27-Sep	2:35 PM	greenhouse	27.1	
10	27-Sep	2:35 PM	garage	19.1	
11					

Figure 1 A relational table called temps stored in an Excel file

In the sample Excel file in Figure 1, Row 1 is the header, giving names to the data in each column. Each subsequent row contains the data reading. Notice how each column has an expected data type: *tdate* is a date, *ttime* is a AM/PM time, *zone* is a string and *temperature* is a real number. Finally, look at the bottom where you see the name of the “sheet”: **temps**. Excel files can contain multiple *sheets*, which correspond to multiple database *tables*.

There are many SQL tools. In this lab, we will experiment with SQLite which, as its name suggests, is a version that does not require a server installation.

Exercise 1

Install SQLite on your Pi and create your first table. [Follow this online tutorial](#). As you follow along, you can copy-paste the SQL code to save on typing.

The tutorial is going to have you install SQLite on your Pi. At home, if you want to install SQLite on your own [Windows] machine, I recommend this tutorial video:

<https://www.youtube.com/watch?v=w7-oOIRlhiQ>

[*Here is a short-and-sweet version of this tutorial*](#)

Exercise 2

Use this tutorial to install [SQLite manager for Raspberry Pi](#). Using the *manager* in this course is only an intermediate stage, one that is useful for testing, for learning and for knowing things are working.

Exercise 3

The final step required in your final project is to access your database from a program (either Java or Python). Follow this tutorial for an introduction: <https://scienceprog.com/working-with-sqlite-in-raspberry-pi-using-python-3/>

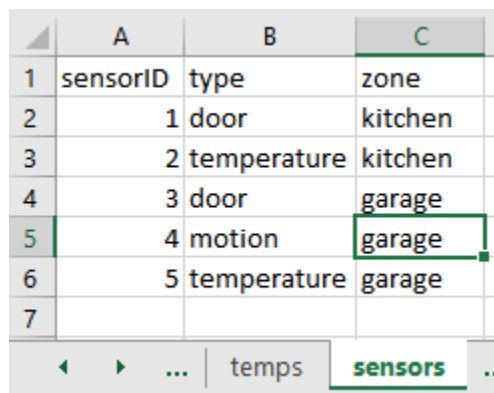
Warning: There is a slight misalignment in the versions of the tutorials for Exercise 1/2 and Exercise 3 – the name of the database is slightly different and the number of columns is slightly different. Use it as a challenge to figure out the changes to the tutorial that you need to make in order to get the job done.

Exercise 4

Can you prove that you learned something about databases? Demonstrate your new skills by changing the program from Exercise 3 to also create another table with the same contents as shown in Figure 2. After creating and filling this table, write Python code to (1) display all the sensors in the kitchen and (2) display all the door sensors.

Call your program **lab4-database-demo.py**. Make sure it is on your Github in the lab3 folder.

If you need help with the two database tasks, consult this reference: <http://www.sqltutorial.org/sql-where/>



	A	B	C
1	sensorID	type	zone
2	1	door	kitchen
3	2	temperature	kitchen
4	3	door	garage
5	4	motion	garage
6	5	temperature	garage
7			

Figure 2 A second table called sensors

Other Good References:

<https://iotbytes.wordpress.com/sqlite-db-on-raspberry-pi/>

<https://randomnerdtutorials.com/esp8266-publishing-dht22-readings-to-sqlite-database/>

MongoDB FYI (For your information)

No work is required here.

MongoDB is becoming very popular among students because it is a “free-form” database (you can store anything) and because you can store JSON objects directly. Yet, Mongo requires a 64-bit operating system while Raspbian O/S is not a 64-bit operating system. There are other versions of this Linux O/S that are 64-bits, but it is recommended that you choose a simpler path for this course (in the interest of time).

If you really wish to proceed, consider checking out the following postings: Both involve selecting a variant of the operating system.

Feb 28, 2019: <https://andyfelong.com/2019/01/mongodb-3-2-64-bit-running-on-raspberry-pi-3-with-caveats/>

Sept 2018: <https://medium.com/@mattalord/embedded-mongodb-4-0-on-raspberry-pi-e50fd1d65e43>

2: INTEGRATING LESSONS

The purpose of these labs is to bootstrap you in your learning so that you can easily go on a build a system from all the parts about which you are learning. For instance, ultimately, in your project, one member of your team will write some JSON data to a ThingSpeak channel, and another member will read that same JSON data from the ThingSpeak channel and put the data into a database.

In the final part of this last individual lab you will demonstrate whether you can do such integration.

Posted on the CULearn is a small Python program that shows how to “scrape data” from a website. The data is in JSON format. Your job is to

1. First run this program to watch how it works (demo-json.py)
2. Extend this program to store the data in a new table in your SQLite database
3. Extend this program to also print and store the wind speed.

Call your program **lab3-database-JSON.py** and **make sure it is in your Github folder lab3.**

The provided file is presented below, for safe keeping. More information about the site is: <https://www.meteomatics.com/api/getting-started/>

```

# The URL that is formatted:
http://api.openweathermap.org/data/2.5/weather?APPID=a808bbf30202728efca23e09
9a4eecc7&units=imperial&q=ottawa
# As of October 2015, you need an API key.
# I have registered under my Carleton email.
apiKey = "a808bbf30202728efca23e099a4eecc7" # If it doesn't work, get your
own.
# Query the user for a city
city = input("Enter the name of a city whose weather you want: ")

# Build the URL parameters
params = {"q":city, "units":"imperial", "APPID":apiKey }
arguments = urlencode(params)

# Get the weather information
address = "http://api.openweathermap.org/data/2.5/weather"
url = address + "?" + arguments

print (url)
webData = urlopen(url)
results = webData.read().decode('utf-8')
    # results is a JSON string
webData.close()

print (results)
#Convert the json result to a dictionary
# See http://openweathermap.org/current#current_JSON for the API

data = json.loads(results)

# Print the results

current = data["main"]
degreeSym = chr(176)

print ("Temperature: %d%sF" % (current["temp"], degreeSym ))
print ("Humidity: %d%" % current["humidity"])
print ("Pressure: %d" % current["pressure"] )

current = data["wind"]
print ("Wind : %d" % current["speed"])

```