In this assignment, you’ll gain experience using Web APIs to gather data, use regular expressions to filter the data, and SQL queries to store and retrieve this data. Specifically, you’ll extract information the the MCommunity API about Arthur F. Thurnau Professors, put the data into a relational database, and retrieve a proper subset of data that meet specific criteria.

**Step 1 (20 points): Retrieving the data**

To access the MCommunity data you will need to access the MCommunity beta API at

https://mcommunity-beta.dsc.umich.edu/miPeople/services/person/search.json

by sending it an POST request with the following data payload:

data = {"searchCriteria":{"givenNameSearchType":"starts with","snSearchType":"starts with","uniqnameSearchType":"is equal","emailSearchType":"is equal","titleSearchType":"starts with","affiliationSearchType":"starts with","phoneSearchType":"ends with","cnSearchType":"starts with","ownerSearchType":"is equal","title":"Arthur F Thurnau Professor","searchForm":"People"}}

HINT: use json.dumps() to encode the data as a JSON string

HINT: create a urllib2 request that specifies the content-type and content length with something like:

request = urllib2.Request(url, json\_data, {'Content-Type': 'application/json', 'Content-Length': len(json\_data)})

then pass that request to urllib2.urlopen()

**Step 2 (20 points): Creating the professor and affiliation tables**

Create two sqlite3 databases, one to hold professor data and one to hold affiliation data.  The professor table should have the following fields:

id  
name  
title  
  
The affiliation tables should have the following fields:

id  
description

The data types for each field are up to you.

HINT: use INTEGER PRIMARY KEY AUTOINCREMENT for the data type for the affiliation id column (and possibly for the professor id column)

**Step 3 (10 points): Populating the professor and affiliation tables**

You should populate the professor table with appropriate data you retrieved from the MCommunity API.  There should be one row for each Thurnau professor.

You should also populate the affiliation table, but there should be no duplicate descriptions in the table.  You should use SQL to check for duplication as much as possible, rather than doing the duplicate checking in python.

**Step 4 (10 points): Create the link table**

The relationship between professors and affiliations is a many-to-many relationship.  That is, a professor can have 1 or more affiliation, and an affiliation can have one or more professors.  In these cases, we need to define a link table to allow us to traverse these relationships.

Create a link table called professor\_affiliation that consists of two fields:

professor\_id  
affiliation\_id

The data types will depend on what you chose for the data types of your id columns in Step 2.

**Step 5 (10 points): Populate the link table**

Insert data into the professor\_affiliation table that maps the id of the professor to the id of their affiliations.

**Step 6 (15 points): Create a chart based on a simple SQL query**

Create a bar chart that shows, in descending order, the number of Thurnau professors in every department, grouped by school or college, based on data you retrieve ***via a single SQL query*** from your sqlite3 database. You can use any charting package or software you like, although the result must be a PNG, JPG, GIF or PDF file.

Note: "LSA UG: Residential College" should be treated as a department and not a College or School.

**Step 7 (15 points): Answer a question based on a complex JOIN in SQL**

How many Thurnau professors are cross-appointed to multiple departments? (i.e. they are not retired and are affiliated with more than one department) Your answer should use **a single SQL query** from your sqlite3 database.

In this step, you will need to join data from the professor, professor\_affiliation and affiliation tables. We’ll introduce some new SQL keywords here to help with this.

(1) You can use the keyword LEFT JOIN to return all rows from the left table (table1) with the matching rows in the right table (table2), where the matching rule is specified by the ON condition. If there is no record in the right table with a matching value, the result is NULL. The syntax of the LEFT JOIN statement is:

SELECT column\_name(s)

FROM table1

LEFT JOIN table2

ON table1.column\_name=table2.column\_name;

The final LEFT JOIN syntax that you generate will be moderately complex, so I recommend you build this up in stages.  If you can't do this in one SQL statement to start with don't panic.  It's not an all-or-nothing thing:  try your best!

**Files to include in the submission:**

si-601-hw4\_{uniqname}.py   (Your code)

si-601-hw4\_{uniqname}.png (or whatever) (Your chart)

si-601-hw4\_{uniqname}.db   (Your database)

As usual, create a zip file containing these files.