

Programming Project 3: A First Flask-sqlalchemy Program

You'll need to acquire the [cit109.tar](#) database as provided in this assignment. For me to be able to grade your program, we will both need the exact same database; I am gonna run your program by accessing my copy of that database, so the two dbs need to be the same.

Before laying out the program requirements, you should give a good look at the following video overview. It will save you some time.

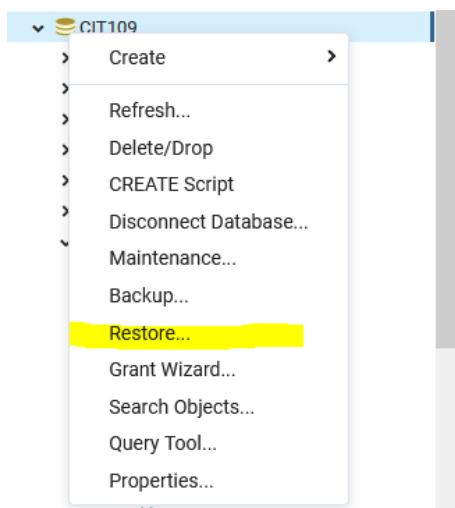


Download this tar backup copy of the CIT109 database, filename `cit109.tar`. `.tar` files are compressed. Download the file using this link given in the folder with this assignment.

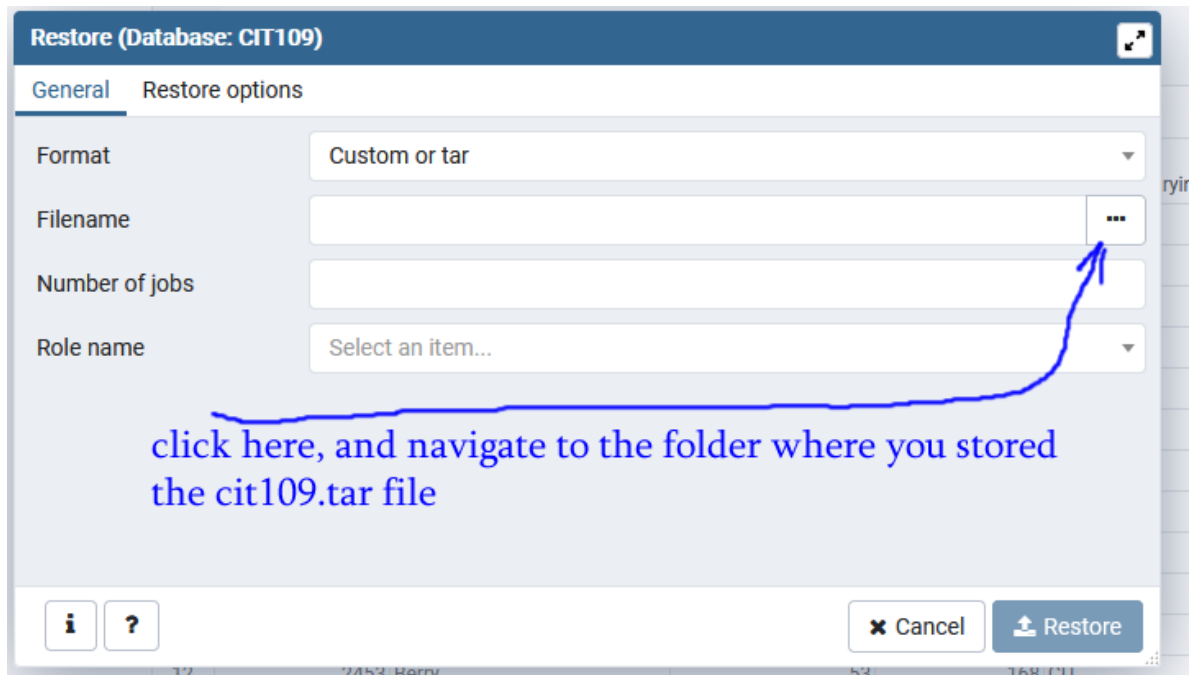
Restoring the Database

The database is named CIT109, the table that holds the data is named *ruby_on_rails*. In order to restore this backup copy on your machine in Postgresql, you will need to do the following.

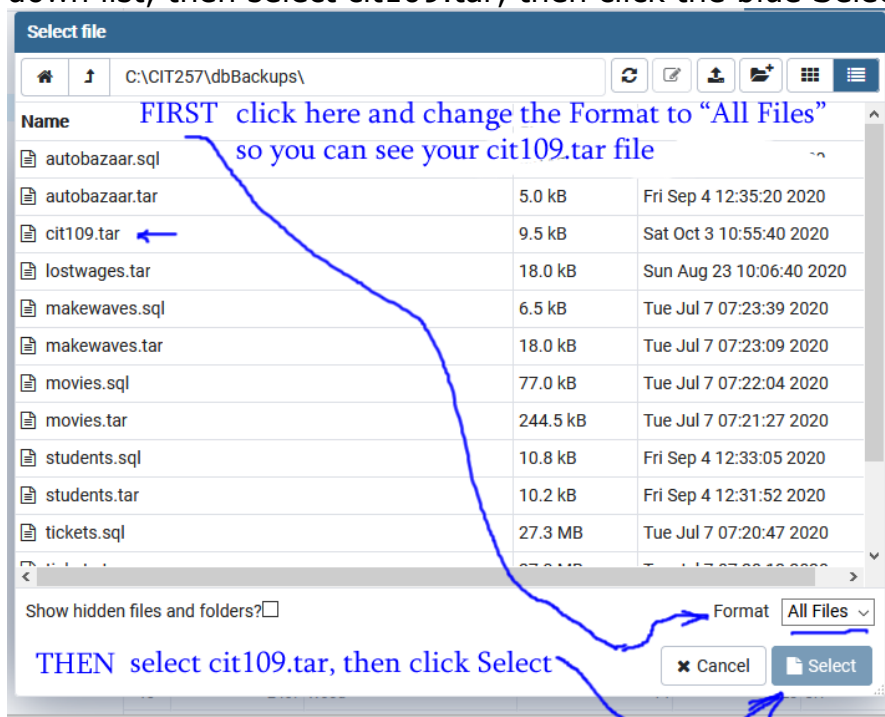
1. Open Postgresql, right-click on *Databases* in the menu on the left side, and choose *Create*. Create a database named CIT109
2. Once that db exists, right-click the CIT109 database name and select *Restore*, as shown in the image.



3. The following dialog should open. Click the 3 dots in the Filename row to navigate to your backup file



4. Once you navigate to the proper folder, initially the file may or may not be visible. So, on the bottom right choose *All Files* from the *Format* drop down list, then select cit109.tar, then click the blue Select button.



5. Things should now look like this, except the file path should be your file path, not mine. If so, click the Restore button

Restore (Database: CIT109)

General Restore options

Format: Custom or tar

Filename: C:\CIT257\dbBackups\cit109.tar

Number of jobs:

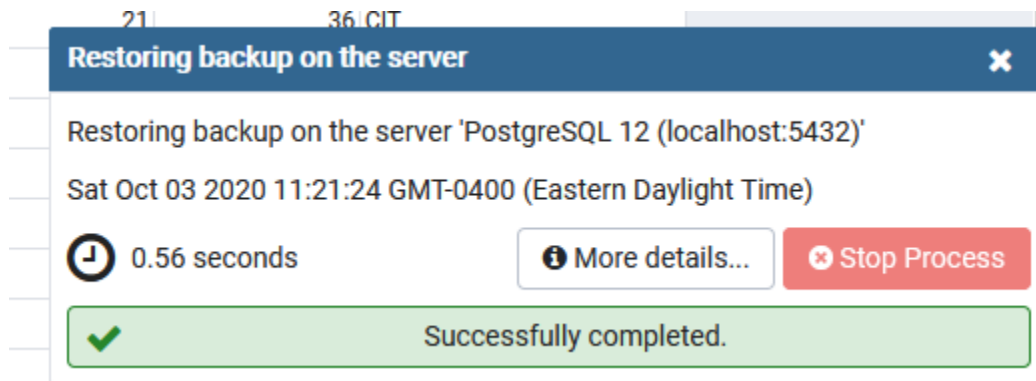
Role name: Select an item...

your file path is likely different from mine, so except for the path to the file, things should look like this...

if so, click Restore

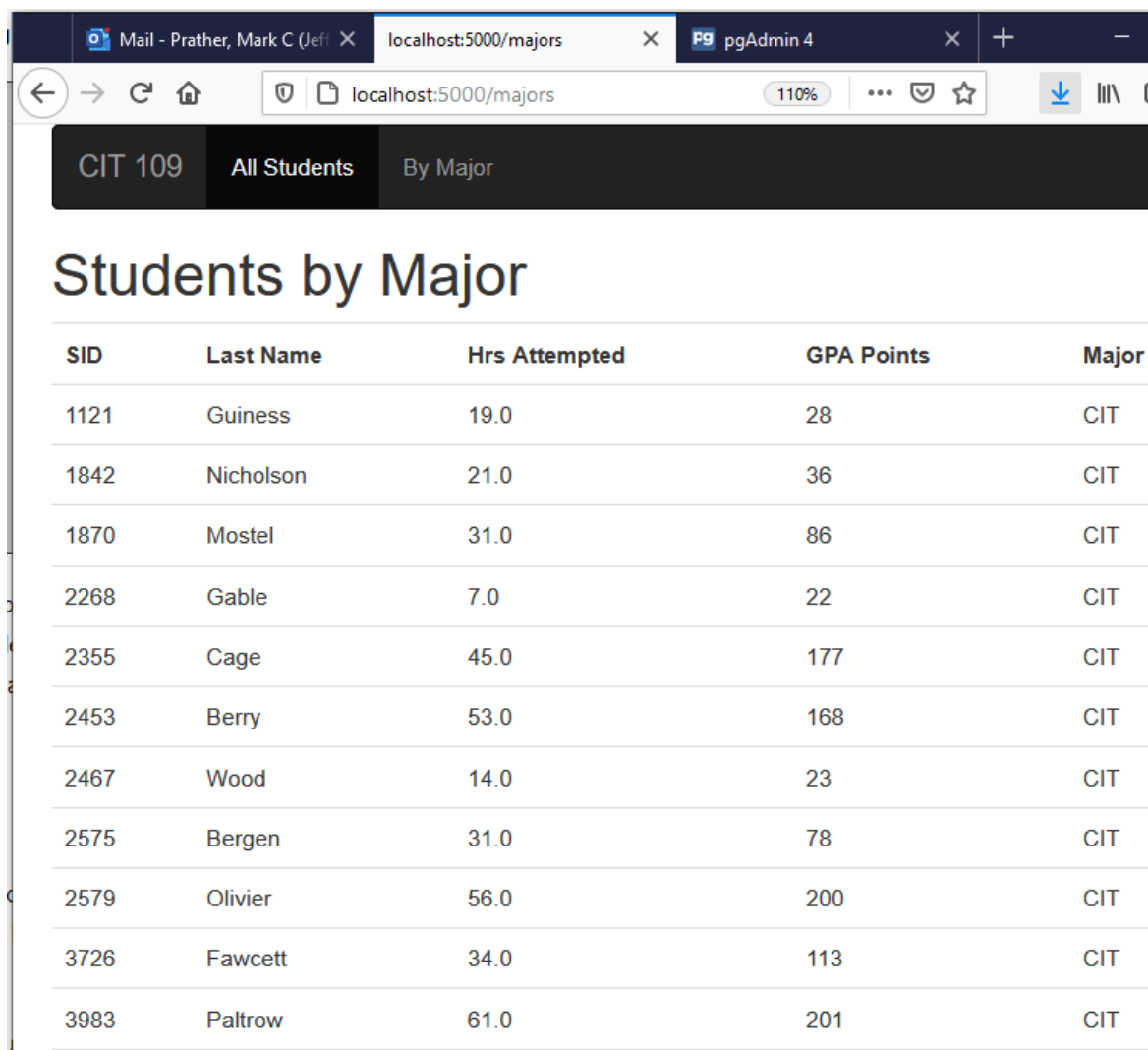
Cancel Restore

6. If successful, postgresql should show the following message.



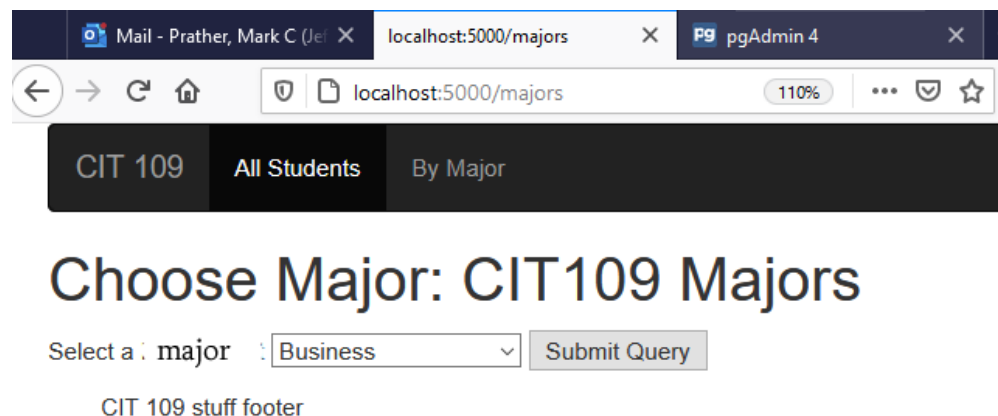
7. Now you're in business. You should have the `ruby_on_rails` table and its data.

You'll need to use Bootstrap and a main layout page that holds the links for the nav bar. I used the same template for both the *All Students* display and the *By Major* data display, the difference being that the *By Major* query filtered the data before sending it to the template. Your default page should display all the students and their data, and look similar to this. The image only shows the first 10 of the 26 students. The template displays this data as a Bootstrap-styled HTML table.



SID	Last Name	Hrs Attempted	GPA Points	Major
1121	Guinness	19.0	28	CIT
1842	Nicholson	21.0	36	CIT
1870	Mostel	31.0	86	CIT
2268	Gable	7.0	22	CIT
2355	Cage	45.0	177	CIT
2453	Berry	53.0	168	CIT
2467	Wood	14.0	23	CIT
2575	Bergen	31.0	78	CIT
2579	Olivier	56.0	200	CIT
3726	Fawcett	34.0	113	CIT
3983	Paltrow	61.0	201	CIT

Clicking on the By Major link in the nav bar should bring up a page with an HTML form that looks similar to this. Doesn't have to be exact, but needs to contain a drop down list of the majors and a submit button.



Choose Major: CIT109 Majors

Select a : major :

CIT 109 stuff footer

After selecting a major and clicking the submit button, the display should look very similar to this. In this case we selected the EET majors.

SID	Last Name	Hrs Attempted	GPA Points	Major
1398	Wahlberg	41.0	103	EET
3968	Tracy	19.0	45	EET
4991	Streep	44.0	135	EET
3117	Torn	24.0	90	EET

CIT 109 stuff footer

As mentioned in the video, the structure of this assignment is very similar to the `sqlalchemy-work1` program that is available from the downloads folder for this lesson. Before getting very far on this program, you should have studied `sqlalchemy-work1` thoroughly. It will save you time in the long run.

You will need to set up a virtual environment that includes installing Flask, `psycopg2`, and Flask-Sqlalchemy.

For full credit, your program needs to have the following features.

1. Use Bootstrap for styling; have a main layout page and a regular template that displays the data, where the regular template inherits from the main layout. Any data displayed should be in a Bootstrap-styled HTML table.
2. The main layout should have a nav bar with 2 links, one for the default page that displays all the students in an HTML table, and a second link that when clicked displays an HTML form allowing students to be filtered by major. The drop-down list in the HTML form should allow choosing from among the 4 different majors.
3. The HTML form that allows selecting by major should post to a route that sends to the template the `ruby_on_rails` table data that has been filtered

by major. The selection should be done in the route code, using the SQLAlchemy object-oriented approach rather than using SQL.

4. Your Python code needs to use Flask-Sqlalchemy for this, and you need a separate models.py file that contains the Flask-Sqlalchemy model for the *ruby_on_rails* table.
5. So, in summary, use Flask-Sqlalchemy, have a models.py file and a regular app.py file, and have a main layout template with a nav bar, and a second template that allows displaying the data in neat columns as an HTML table. The HTML form should have a drop-down list and a submit button.

I used the same template for displaying all the students and for displaying the students by major. You don't have to if you don't want to, but the columns displayed are identical, and the only different is what data is sent to them.

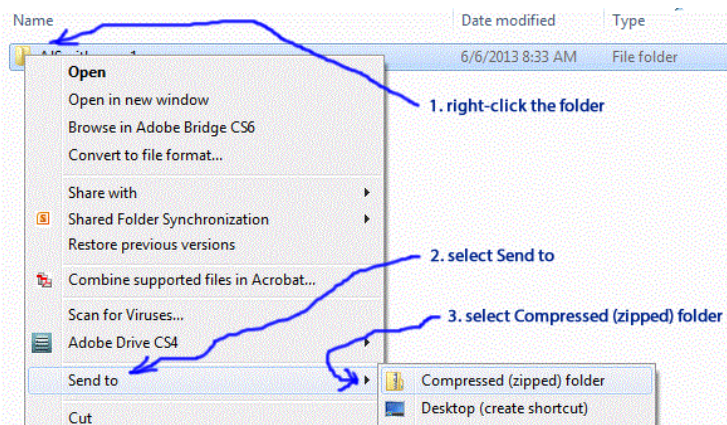
Please start early in the week this is due. The sqlalchemy-work1 example is your best guide for how this should work.

If you get good and stuck email me your zipped code and we'll get it working.

How To Submit Your Program

When you're finished, put the program in a new folder and name that folder using your name and project number. For example the folder might be named `AJSmith-prog1`. ***Please put your name on the folder as well as having your name in a comment in the file itself.*** I get lots of programs; putting your name on the folder help keep things organized and shortens the amount of time it takes to get things graded.

After creating the folder, Zip (compress) the folder by right-clicking on it, and then choosing **Send to** from the pop-up menu, then select **Compressed folder**. This will crate a new zipped folder identifiable by the little zipper on the folder icon.



Then, attach the zipped folder to an email and send it to me.

Thanks...

Again, if there are questions about this, let me know: mark.prather@kctcs.edu..