CIT244 Python II Program 3

Program 3: python & databases

`SQLITE3 and the WX LISTCTRL`

Please start early in the week that this program is due; this gives you plenty of time to ponder how is should work. Moreover, there will be more to ponder in these programs as the course progresses. I am glad to help troubleshoot programs, but I go to bed reasonably early and am unlikely to stay up until midnight of the due date. You will need the *tickets.csv* file from the course website. It contains the data that you will use to populate your table.

You need to do 2 things to make this program work.

1. Create a sqlite3 database with the exact name of *speeding_tickets.db*, create a table within that database with the exact name of *tickets*, then take the *tickets.csv* CSV file given in the same folder as this document and import that data into the tickets table.
2. Write a program that reads the data from the table and inserts it into a wx.ListCtrl. The program also uses a dialog box to allow adding new records to the table. You *do not* need to use an editable list control. The simpler, the better.

For me to run and grade your program using my database, our two database names need the have the same name, including the file extension, and have same column names, and the same column data types. So, please follow the given instructions carefully.

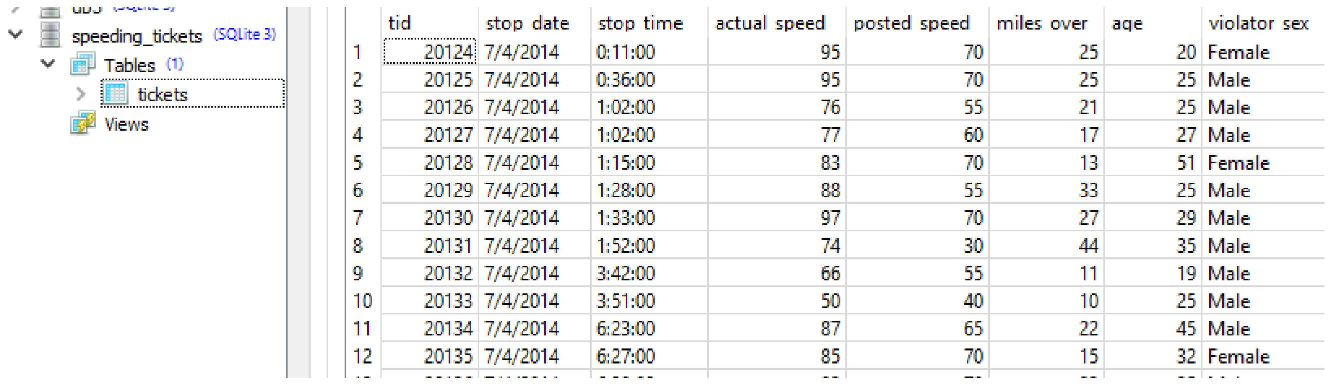
> The table holds real data for the first 49 speeding tickets issued during the early hours of July 4th, 2014, in or sort of near St. Paul, Minnesota.

Database name: `speeding_tickets.db`

Table name: `tickets`
column names: `tid, stop_data, stop_time, actual_speed, posted_speed, miles_over, age, violator_sex`
tid, the integer primary key, is *not* auto-incrementing. Here's a possible SQL statement to create the table. Note column names and data types:

```
CREATE TABLE tickets (
   tid INTEGER NOT NULL,
   stop_date TEXT NOT NULL,
   stop_time TEXT NOT NULL,
   actual_speed INTEGER NOT NULL,
   posted_speed INTEGER NOT NULL,
   miles_over INTEGER NOT NULL,
   age INTEGER NOT NULL,
   violator_sex TEXT NOT NULL,
   PRIMARY KEY (
      tid
   )
);
```

And here's a glimpse of the first few records, starting at 11 minutes into the morning of July 4th, 2014, as seen in SqliteStudio.



	tid	stop date	stop time	actual speed	posted speed	miles over	age	violator sex
1	20124	7/4/2014	0:11:00	95	70	25	20	Female
2	20125	7/4/2014	0:36:00	95	70	25	25	Male
3	20126	7/4/2014	1:02:00	76	55	21	25	Male
4	20127	7/4/2014	1:02:00	77	60	17	27	Male
5	20128	7/4/2014	1:15:00	83	70	13	51	Female
6	20129	7/4/2014	1:28:00	88	55	33	25	Male
7	20130	7/4/2014	1:33:00	97	70	27	29	Male
8	20131	7/4/2014	1:52:00	74	30	44	35	Male
9	20132	7/4/2014	3:42:00	66	55	11	19	Male
10	20133	7/4/2014	3:51:00	50	40	10	25	Male
11	20134	7/4/2014	6:23:00	87	65	22	45	Male
12	20135	7/4/2014	6:27:00	85	70	15	32	Female

Create the speeding_ticket.db Database

Prerequisite: In the most recently assigned set of lecture notes there was a video on how to download SqliteStudio and how to create a database and import data. SqliteStudio is a free GUI application for managing sqlte3 databases. If you missed that, or have not had time to work with SqliteStudio, you should review that video. We'll use SqliteStudio here, so if you have not done so, you need to first install SqliteStudio (or whatever software you want to use to manage sqlite3).

Open SqliteStudio. Click on the Database menu, choose Add a Database. This dialog box opens. Click the green plus sign to create a new database. Name the database '*speeding_tickets.db*'. Eventually this file should end up in the same folder as your Python program.

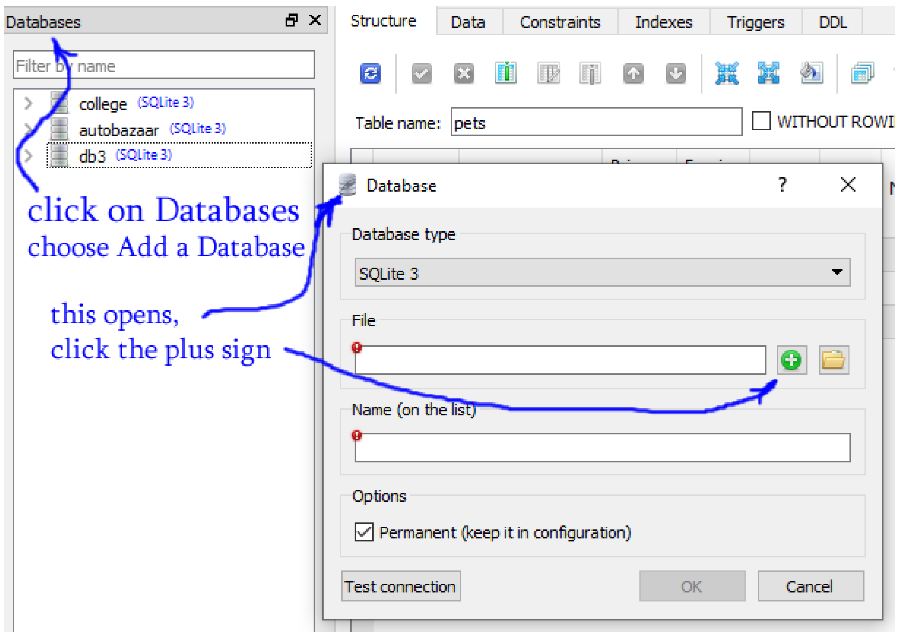


Table name: pets

Database type: SQLite 3

File:

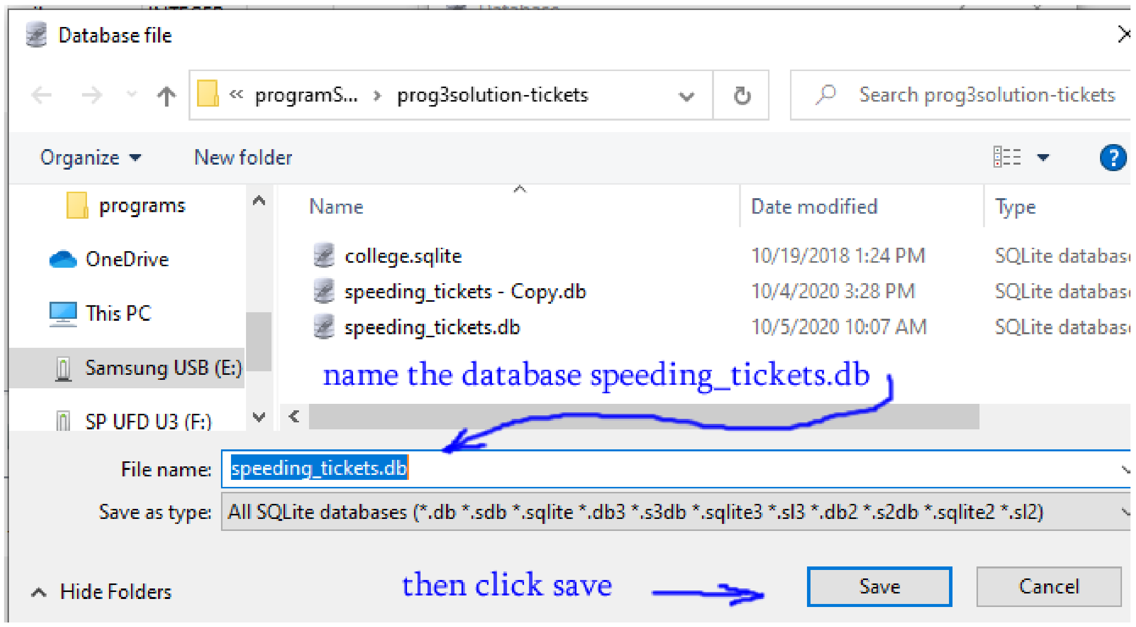
Name (on the list):

Options: ☑ Permanent (keep it in configuration)

click on Databases
choose Add a Database

this opens,
click the plus sign

Clicking on the green plus sign opens up the following, which allows you navigate to a place to save the file and give the db a name. Make sure to save the new database somewhere easy to find. Then click Save.



Name	Date modified	Type
college.sqlite	10/19/2018 1:24 PM	SQLite databas
speeding_tickets - Copy.db	10/4/2020 3:28 PM	SQLite databas
speeding_tickets.db	10/5/2020 10:07 AM	SQLite databas

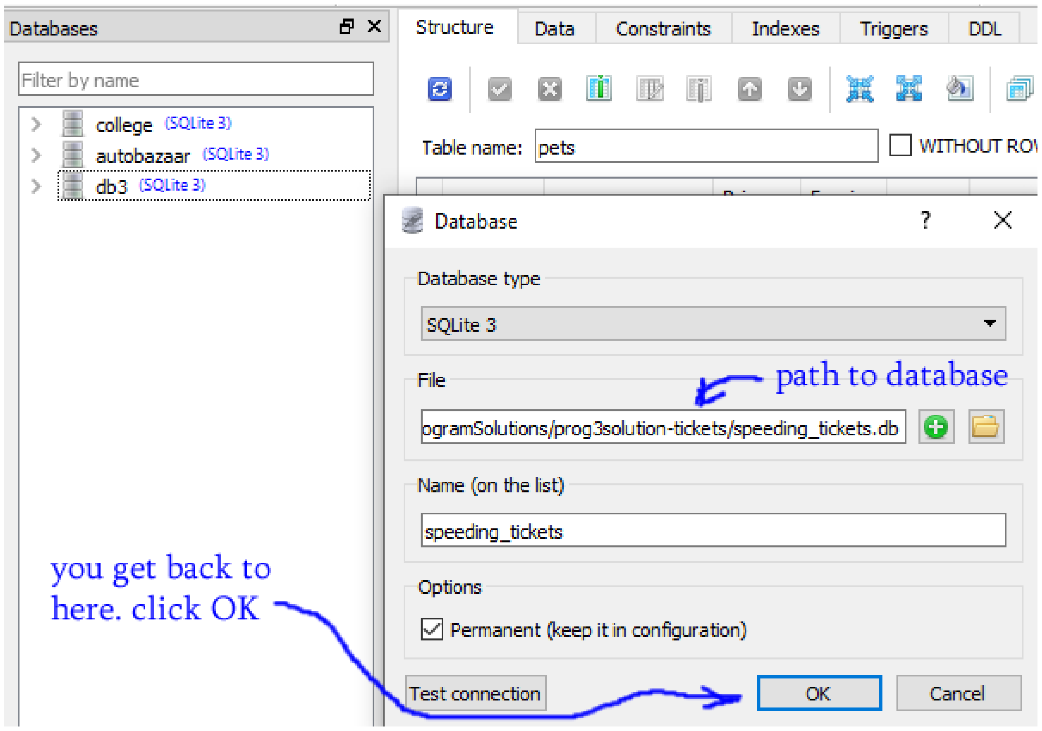
File name: speeding_tickets.db

Save as type: All SQLite databases (*.db *.sdb *.sqlite *.db3 *.s3db *.sqlite3 *.sl3 *.db2 *.s2db *.sqlite2 *.sl2)

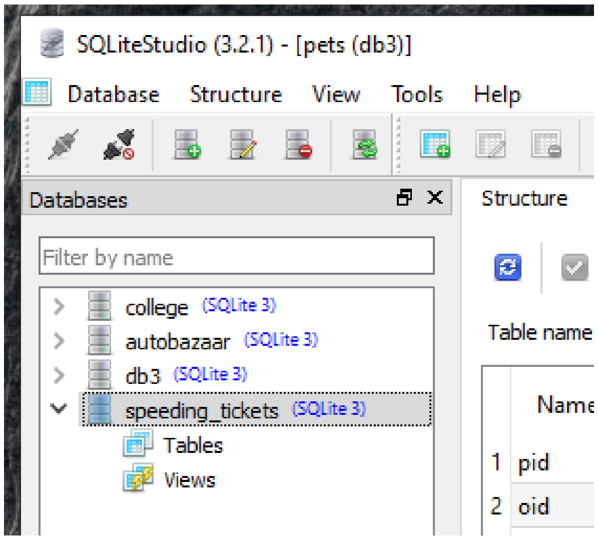
name the database speeding_tickets.db

then click save

Clicking Save in the above image gets you back to the following. *Shown in the image is my path to the db file. Your path will most likely be different.* If it all looks like it is right, OK.



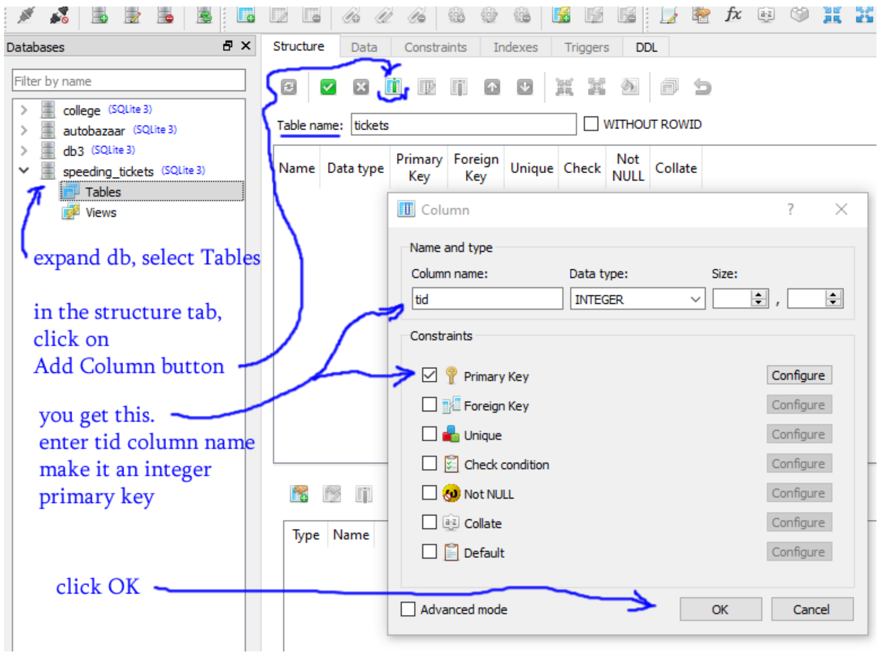
Expand the speeding_tickets db by clicking on the little arrow to the left of the db name.



Time to create a the table.

- Right click on Tables, and select *Create Table*. In the Table name textbox, name the table *tickets*.
- Then click on the Add Column icon, shown in the image below.

A dialog box opens allowing you to add column names and specify their data types and any constraints on that data. You will click this icon for each column you want to add. Here add the *tid* integer primary key first.



expand db, select Tables

in the structure tab, click on Add Column button

you get this. enter tid column name make it an integer primary key

click OK

Table name: tickets
WITHOUT ROWID ☐

Column

Name and type

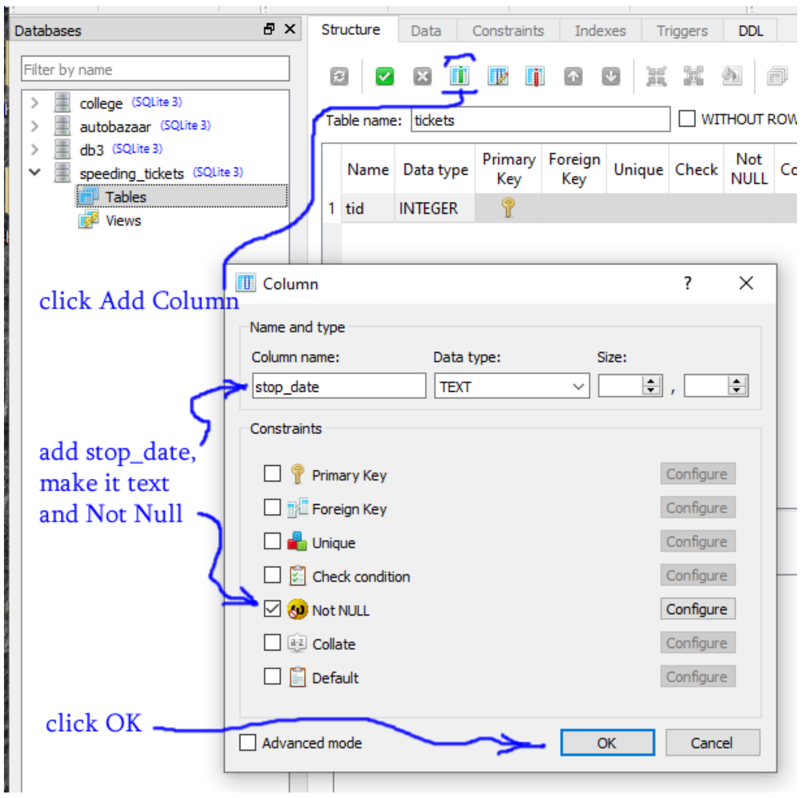
Column name:	Data type:	Size:
tid	INTEGER	

Constraints

- ☑ Primary Key — Configure
- ☐ Foreign Key — Configure
- ☐ Unique — Configure
- ☐ Check condition — Configure
- ☐ Not NULL — Configure
- ☐ Collate — Configure
- ☐ Default — Configure

☐ Advanced mode — OK — Cancel

Next, click the Add Column icon, and add the *stop_date* column which will be of type TEXT. Be sure to check the Not NULL checkbox.



click Add Column

add stop_date, make it text and Not Null

click OK

Table name: tickets
WITHOUT ROW ☐

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL
1	tid	INTEGER	🔑				

Column

Name and type

Column name:	Data type:	Size:
stop_date	TEXT	

Constraints

- ☐ Primary Key — Configure
- ☐ Foreign Key — Configure
- ☐ Unique — Configure
- ☐ Check condition — Configure
- ☑ Not NULL — Configure
- ☐ Collate — Configure
- ☐ Default — Configure

☐ Advanced mode — OK — Cancel

After adding the remaining columns, setting their data types, and checking the Not NULL checkboxes, it should look like the following. The square checkmark icon (which will be green for you) shown in the image should be clicked when you've added all the field names in order to commit and save the changes. If you forget to click the checkmark, your changes may not be saved. It will be less fun if you have to do this twice.

After the columns are set up, right-click the *tickets* table and choose *Import into the table*.

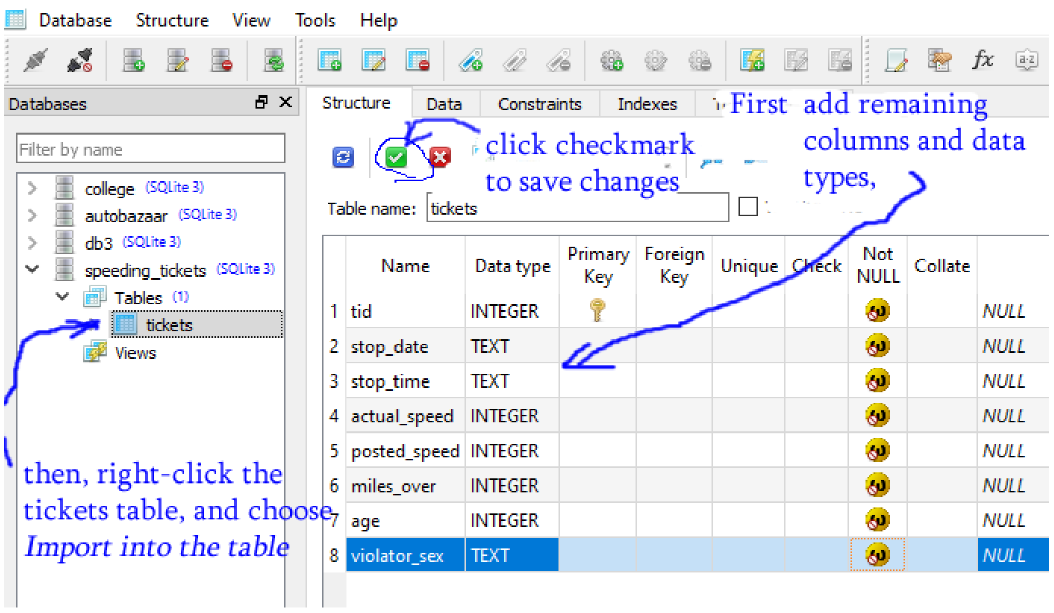
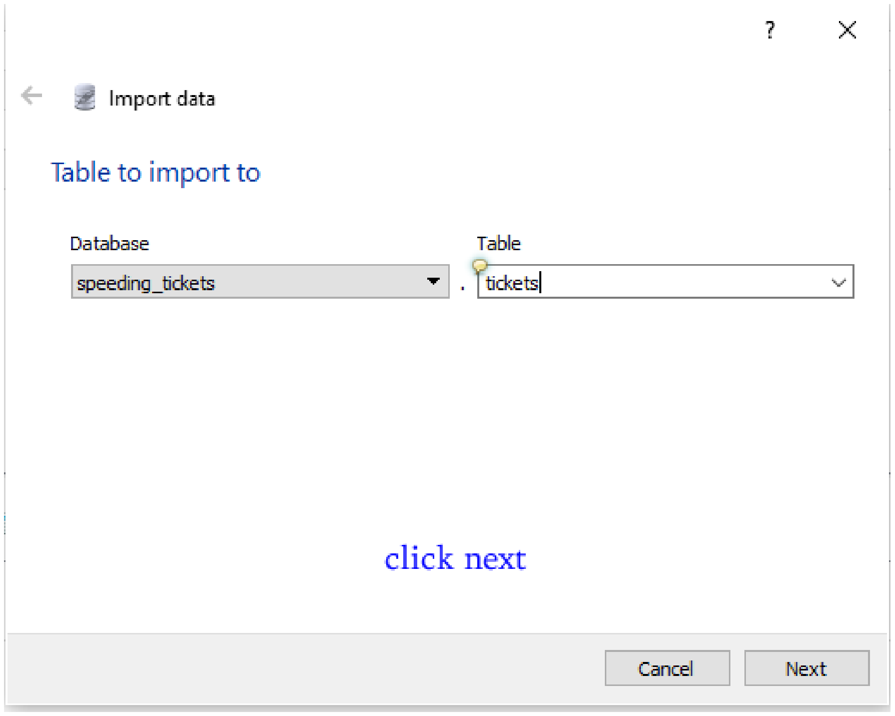

Database
Structure
View
Tools
Help
Databases
Filter by name
college (SQLite 3)
autobazaar (SQLite 3)
db3 (SQLite 3)
speeding_tickets (SQLite 3)
Tables (1)
tickets
Views
Structure
Data
Constraints
Indexes
First add remaining columns and data types,
click checkmark to save changes
then, right-click the tickets table, and choose Import into the table

Table name: tickets

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	
1	tid	INTEGER	🔑				✓		NULL
2	stop_date	TEXT					✓		NULL
3	stop_time	TEXT					✓		NULL
4	actual_speed	INTEGER					✓		NULL
5	posted_speed	INTEGER					✓		NULL
6	miles_over	INTEGER					✓		NULL
7	age	INTEGER					✓		NULL
8	violator_sex	TEXT					✓		NULL

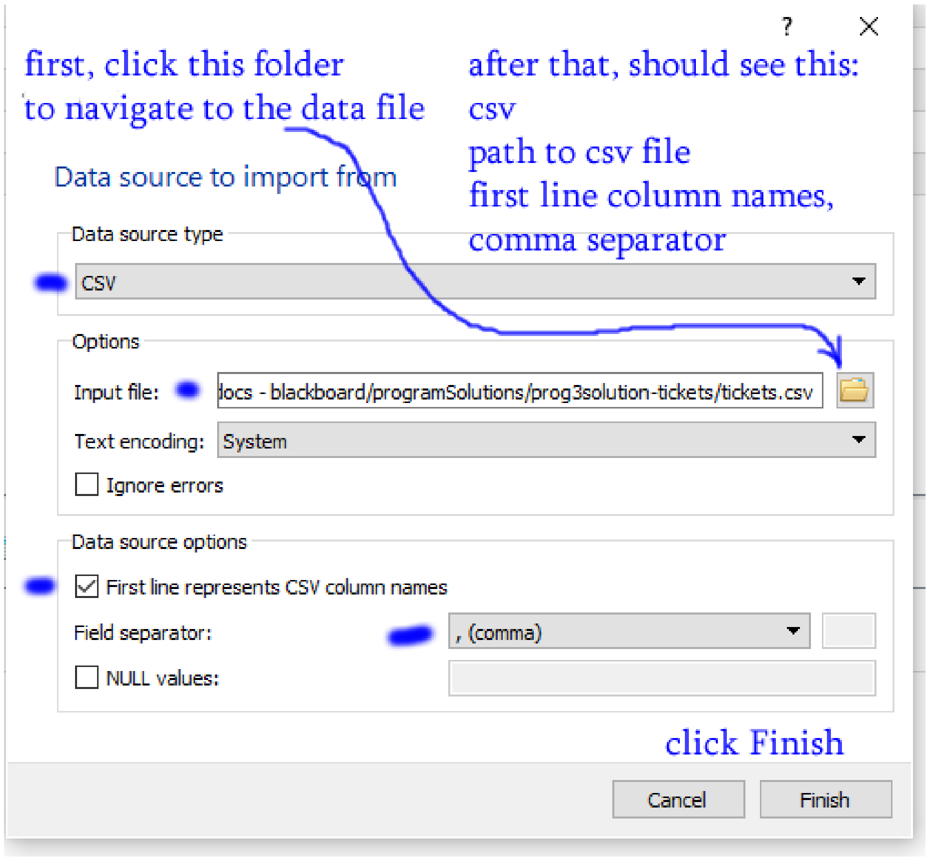
After the import into table command, this should appear, with DB name and table to receive the imported data. Click Next.



Database: speeding_tickets

Table: tickets

You should see this dialog box with this info: CSV file, proper path to your tickes.cvs file, check the box for *First line represents CSV columns names*, and make sure comma is selected as the field separator. The first line of our CSV file does indeed contain the column names. So it is important that you check the box shown below before you click Finish. Otherwise, you will get a type mismatch error. Click Finish.


first, click this folder to navigate to the data file

after that, should see this:
csv
path to csv file
first line column names,
comma separator

Data source to import from

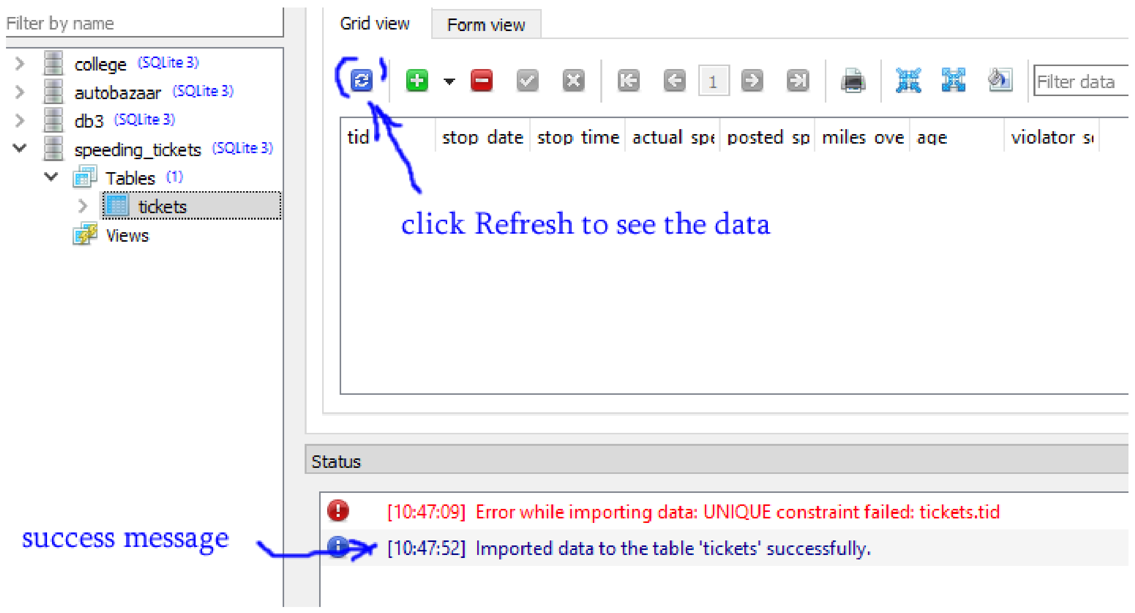
Data source type: CSV

Options
Input file: locs - blackboard/programSolutions/prog3solution-tickets/tickets.csv
Text encoding: System
☐ Ignore errors

Data source options
☑ First line represents CSV column names
Field separator: , (comma)
☐ NULL values:

click Finish

If successful, you will get a success message at the bottom. To see the data itself click the icon shown to refresh the page. Don't forget to to click the icon that saves the changes to the table. *After the data appears*, be sure to click either the blue icon or else the icon with the check mark to save the added data. No use doing this twice.



```
[10:47:09] Error while importing data: UNIQUE constraint failed: tickets.tid
[10:47:52] Imported data to the table 'tickets' successfully.
```

With that done...

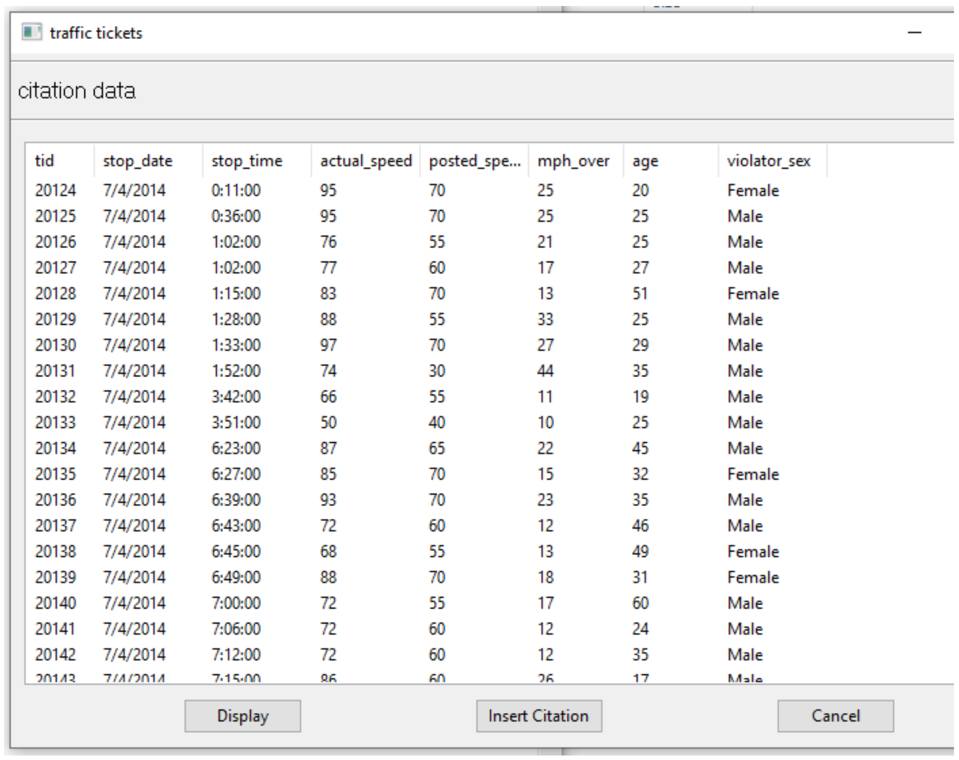
The Program Assignment

There is an example program in the lecture notes that is pretty similar in structure to this assignment. Studying that example and watching the associated video before starting this program will save you time in the long run. Just sayin'.

Write a program that reads table data from the database into a wx.ListCtrl. The database will be named *speeding_tickets* and the table is to be named *tickets*. The CSV file contains a row of 8 heading and then 49 rows of records.

Your interface will need a wx Frame and a wx ListCtrl and 3 buttons. A display button loads all of the ticket data into the list control. An insert button will open a wx Dialog that allows the user to enter the 8 data items used for a singe record. When the user closes the dialog the new record will be SQL INSERTED to the tickets table and the list control re-populated so as to display the inserted record. You also need a close button to close the program.

A possible interface might look like this; you may design your own layout as long is it works like that shown. The list control should be wide enough that you don't need a horizontal scroll bar. A vertical scroll bar is fine.

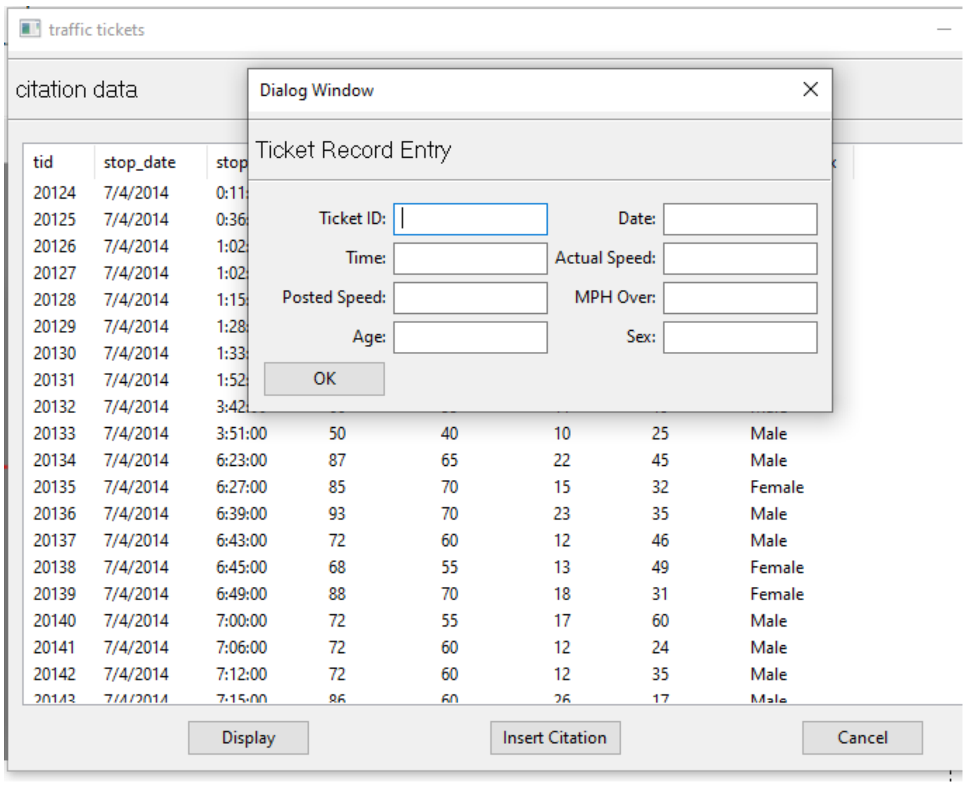


traffic tickets

citation data

tid	stop_date	stop_time	actual_speed	posted_spe...	mph_over	age	violator_sex
20124	7/4/2014	0:11:00	95	70	25	20	Female
20125	7/4/2014	0:36:00	95	70	25	25	Male
20126	7/4/2014	1:02:00	76	55	21	25	Male
20127	7/4/2014	1:02:00	77	60	17	27	Male
20128	7/4/2014	1:15:00	83	70	13	51	Female
20129	7/4/2014	1:28:00	88	55	33	25	Male
20130	7/4/2014	1:33:00	97	70	27	29	Male
20131	7/4/2014	1:52:00	74	30	44	35	Male
20132	7/4/2014	3:42:00	66	55	11	19	Male
20133	7/4/2014	3:51:00	50	40	10	25	Male
20134	7/4/2014	6:23:00	87	65	22	45	Male
20135	7/4/2014	6:27:00	85	70	15	32	Female
20136	7/4/2014	6:39:00	93	70	23	35	Male
20137	7/4/2014	6:43:00	72	60	12	46	Male
20138	7/4/2014	6:45:00	68	55	13	49	Female
20139	7/4/2014	6:49:00	88	70	18	31	Female
20140	7/4/2014	7:00:00	72	55	17	60	Male
20141	7/4/2014	7:06:00	72	60	12	24	Male
20142	7/4/2014	7:12:00	72	60	12	35	Male
20143	7/4/2014	7:15:00	86	60	26	17	Male

Clicking the Insert Citation button should open a dialog box that allows adding a new record to the table. You do not necessarily need to have the dialog widgets in 2 columns as shown, a single column would work just as well.



traffic tickets

citation data

Dialog Window

Ticket Record Entry

Ticket ID:		Date:	
Time:		Actual Speed:	
Posted Speed:		MPH Over:	
Age:		Sex:	

tid	stop_date	stop					
20124	7/4/2014	0:11:					
20125	7/4/2014	0:36:					
20126	7/4/2014	1:02:					
20127	7/4/2014	1:02:					
20128	7/4/2014	1:15:					
20129	7/4/2014	1:28:					
20130	7/4/2014	1:33:					
20131	7/4/2014	1:52:					
20132	7/4/2014	3:42:					
20133	7/4/2014	3:51:00	50	40	10	25	Male
20134	7/4/2014	6:23:00	87	65	22	45	Male
20135	7/4/2014	6:27:00	85	70	15	32	Female
20136	7/4/2014	6:39:00	93	70	23	35	Male
20137	7/4/2014	6:43:00	72	60	12	46	Male
20138	7/4/2014	6:45:00	68	55	13	49	Female
20139	7/4/2014	6:49:00	88	70	18	31	Female
20140	7/4/2014	7:00:00	72	55	17	60	Male
20141	7/4/2014	7:06:00	72	60	12	24	Male
20142	7/4/2014	7:12:00	72	60	12	35	Male
20143	7/4/2014	7:15:00	86	60	26	17	Male

After entering data for a new record, clicking the OK button in the dialog box should insert the record into the tickets table. Your code should then repopulate the list control so as to include the new record.

If clicking the link opens the file, just save it to a convenient location.

Here's what you need to do to get 100% on this one.

- create a sqlite3 database named `speeding_tickets` and create a table named `tickets` with the field names spelled exactly like those given in the CSV file. I recommend using the SQliteStudio, but you may use whatever program or method you want. There is a video on using Sqlite Studio in the lecture notes. You'll need to import the CSV file into SqliteStudio (or whatever) to generate the database.
- Your program should work with my database. For simplicity, as you write your program keep your database file in the same folder as your program so that we don't have any path issues.
- import the CSV file into the students table with the exact field names as given in the file. You should have 49 records.
- your program should properly connect to the database you created using techniques from the lecture notes.
- the id field should be an integer primary key. it does not need to be auto-incrementing.
- actual_speed, posted_speed, miles_over, and age should all be integers.
- stop_date, stop_time, and violator_sex should be of type TEXT.
- the Display button should load the whole table into the list control
- the Insert Citation button should open a custom dialog that allows entering a new, complete record
- when the dialog is closed the new record is added to the tickets table using an SQL INSERT statement and the updated table should then be re-displayed in the list control
- the close button closes the program.
- you may use sizers or absolute positioning, your choice, but all widgets need to be visible to the user without having to resize the window.

If you get good and stuck attach a compressed version of your code and email it to me; I'll be glad to help you troubleshoot it. I have a copy of the database, so you don't need to send that.

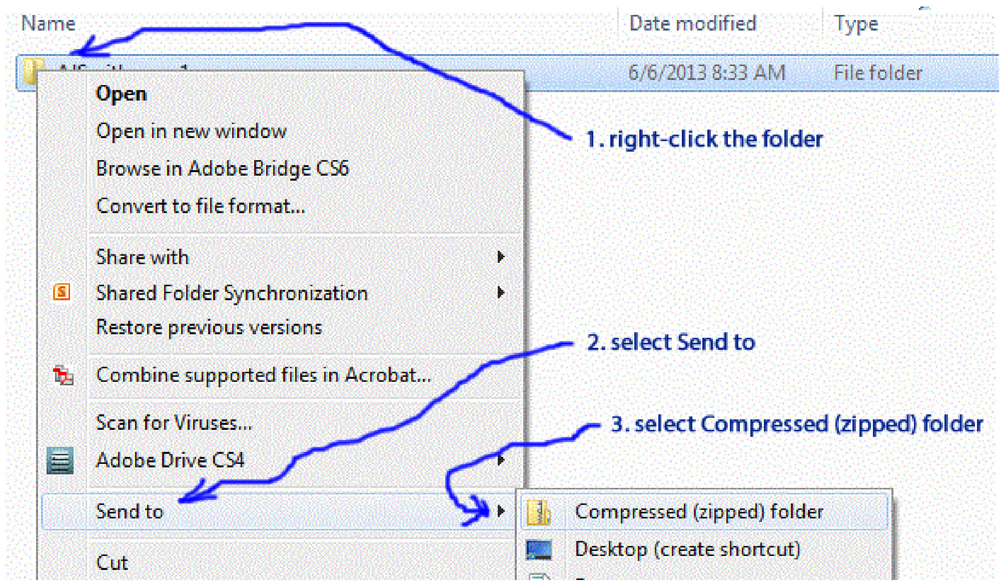
Grading will follow this approach.

- 100% program runs perfectly with all required items
- 90% program meets all requirement but doesn't always produce the correct output
- 80% program runs without errors, but is missing some requirement.

- 70% program had a good start, most but not all of the code was correct, had one or more fatal errors.
- 40-60% or below. Not very close, many parts missing, would not run, probably a late start.
- 0% no submission or code was not the student’s work.

How To Submit Your Program

You need to compress the folder that holds your program file or files before you email it to me. Uncompressed code is likely to be filtered out by a suspicious email server. You probably know this, but to compress an existing folder, *right click it*, choose *Send to*, then choose *Compressed (zipped) folder*, as shown below. The zipped folder has a little zipper on it.


Name
Date modified
Type
6/6/2013 8:33 AM
File folder
Open
Open in new window
Browse in Adobe Bridge CS6
Convert to file format...
Share with
Shared Folder Synchronization
Restore previous versions
Combine supported files in Acrobat...
Scan for Viruses...
Adobe Drive CS4
Send to
Cut
Compressed (zipped) folder
Desktop (create shortcut)
1. right-click the folder
2. select Send to
3. select Compressed (zipped) folder

If you email me your finished program, unless it comes in late at night, I will usually grade it that same day. Also, if you get stuck on something you can also send me your code and I'll be happy to help you get it working. The earlier in the week the program is due that you start, the more time we have to sort out any problems there may be. So, please start on these program many days before the due date.

> Please clearly name the folder with your code, with *your name, the class, and the program number*. For example, `jsmith-pythonprogram3`.

I get lots and lots of programs from several different classes, so it helps me a great deal in keeping things straight if I can tell what's in the folder by the folder label.

Once everything is working (or even if it isn't) attach the zipped folder to an email and send it to me..

If you have questions let me know: mark.prather@kctcs.edu.