**Objectives:**

1. Design a simple database  (15 points)
2. Practice using SQLite (and optionally Peewee ORM) with Python (35 points)
3. **Writing modular code (25 points)**
4. **Testing database code (25 points)**

**Don't forget parts 3 and 4!**

**Task:**

Your customer is an art store who sell art from many different artists. The artists create unique artworks, which are sold in the store.

Build a prototype app to store data about the artists and their artworks. Focus on the required features. You'll be graded on application design and writing testable code.

Your app needs to save this data about each artist:

* name
* email address

and data about the artworks:

* artist
* name of artwork - each artist's artwork has a unique name
* price
* whether the artwork is available, or if it has been sold

Your application should have these features

* add a new artist
* search for all the artwork by an artist (everything - available and sold)
* display for all the available artwork by an artist
* add a new artwork. Make sure the artwork is associated with an artist. If needed, create an artist first.
* delete an artwork
* change the availability status of an artwork, for example, change from available to sold

Your application should use a SQLite database to persistently store data.

You should use foreign key relationships to help ensure that (for example) your user can't create a record of an artwork for an artist that doesn't exist.

You can build this as a command line app.   (*Optional: create a simple Tkinter GUI for your application;*[*https://likegeeks.com/python-gui-examples-tkinter-tutorial/*](https://likegeeks.com/python-gui-examples-tkinter-tutorial/)*OR build a web app, perhaps with Flask.*[*http://flask.pocoo.org/*](http://flask.pocoo.org/) )

In SQLite, foreign keys will be created, but won't be enforced. To make SQLite respect your foreign key relationships, configure your DB as described here: <https://sqlite.org/foreignkeys.html>

As you build your application, remember to write functions, and create modular, focused modules and classes.

Use parameterized queries with your database if you use Python's sqlite3 module.

Make sure you handle recoverable database errors (example: adding a artwork with no artist, duplicate artist name...) in a user-friendly way.

**Write unit tests. Separate the UI from the DB code and focus your energy on testing application logic. Your tests should focus on  unhappy paths, and logic that is difficult or time-consuming to test from your user interface.**

Comment your code!

**To submit:**

1. Zip all of the files in your project and upload to the Project 3 dropbox
2. Push your project code to GitHub and submit the URL to the Project 3 dropbox