# Coding Info

# MainDialog

* creates the app dialog. It is a shell that lists the functions (queries) that we want to call, has a title, subtitle (function we are in) , a display area for the last SQL call issued to the db and a content area where each of the queries are displayed
* the controller and fxml shells for each of the current content areas have been been created.
* each content area controller shells has 2 initial functions: initialize() which is called when the app starts and activate() which is called when the user selects that function from the main dialog
* initialize()
  + this is a javafx function. It is used to set up the content. Generally anything done here can also be done in the fxml but somethings are much easier here. You can create the all the content and layout here if you want but I find the general layout is easier in the fxml but that's just a preference. The fxml generally contains at least the parent container. Don't put any database access in this function since this is called when the apps starts, it will slow down the startup if all the content areas are loading up data.
* activate()
  + this is a method I added. It is called from the main dialog when the user selects a function to execute and is run just as the content area becomes visible. This is where any initial database calls should go. Thinks like populating drop down selections etc. Remember that you need to clear any drop downs etc before you populate them because they are not cleared when a new content area is shown.
* The SQL area of the main dialog is bound to a property of the main dialog called formattedDisplayText. There are getters and setters define for it. I intend to pretty up the display of the SQL later.

# ProjectionContent

* I've created a sample database access to populate a SearchableComboBox in the Projection content area. There is also code to show how to get at the value the user selected.
  + The combobox is populated with data objects for the artists. The data object is just a list of the fields in the artist table plus a couple of calculated fields for display purposes
  + In order to control what the user actually sees (the artist name) from the object, you define a CellFactory for the control and override the updateItem method. [lest you think I'm brilliant, it actually took quite a bit of research to understand this, fortunately I'd done something similar in another app].
    - Basically when the cell is updated (data is loaded) two values are passed into the updateItem method, the object (in this case ArtistDO) and a boolean (I don't know what this is for).
      * I added a method on the ArtistDO to return the displayNameByLast when the toString() method is called. That is what is displayed in the field the user sees.
  + The getValue() method of the SearchableComboBox returns the full object (ArtistDO). I added a listener on the control to know when the value changes. Once you have the object, you can access any of the fields. I got the artistID and put it in a text field for the demo.