Rubric for Databases Final Project

Deliverables:

- You will turn in all the code for the final website application, including Python files, schemas for each table (if they have changed from previous milestones), the initial data you used (CSV files, Excel files, or SQL INSERT statements).
- Your website can work in any way you'd like, but it should be logical from a user perspective. You do not need to have a login/logout system, but if you do, it should work appropriately (though the security of this will not be evaluated).
    - Your website, for full credit in the functionality section (see below) should go beyond basic display of tables that are pulled straight from the underlying data. In other words, there should be some functionality that requires you to write more sophisticated queries than just "`SELECT x, y, z FROM table t`." You should have some queries that include joins (hopefully multiple joins), as well as aggregation. The goal is not to write complicated queries for the sake of complicated queries, but to provide the most useful user experience.
    - The one specific requirement for your website is it should include at least one *dashboard*. This should be a central page that includes multiple (say, at least 5) visualizations (graphs or charts) illustrating something about the data you have used in your project. This is a good place where you can write more complicated queries that might not fit elsewhere. In other words, the dashboard can be a place to display an "overview" of the state of your database. In other words, imagine creating a class registration system for Rhodes (similar to our labs done in class with NiceGUI). From perspective of a "regular" user (a student), all they will need to do is browse, add, and drop classes. However, a dashboard will be useful from the registrar's perspective to provide a real-time overview of the course registration system that displays, for instance, what the most popular classes are, the most popular course timeslots, the most popular professors, whether there are any classes that have abnormally low or high enrollments, etc.
    - Alternatively, these charts or graphs for the dashboard can be spread around your website in other places, if that makes more sense from a functionality standpoint.
- You should include a document that tells me how to run your project. This can be as simple as a text file that states which Python file I should execute that starts the website. You can also include, in this document, any pieces of the project you did not accomplish and rationale for why you did not accomplish them.

(See specific grading criteria on next page.)

Category	Points	Criteria
Functionality and Completeness	25	Website application performs all major operations as described in previous milestones, or justification is provided. Application is stable, with well-tested core features. Partial credit for bugs or unimplemented functionality.
Database Design and Queries	25	At least 8 well-structured, normalized tables. Schema reflects good relational modeling. Appropriate use of SQL joins, aggregation, data type selection. At least one functional operation each for SELECT, INSERT, UPDATE, and DELETE (or justification for why we aren't using one or more of them).
Dashboard Implementation	10	Dashboard is present and meaningful. It should present aggregated or summarized data (counts, totals, etc.) and offer users a high-level overview. The layout should be clean and visualizations should be easy to understand.
Usability and Interface Design	10	Pages are logically structured; navigation is intuitive. No visual polish required, but the user should clearly understand how to access each feature and what each operation does. All operations needed to run the application are implemented through the web interface (don't have to do anything through the command line) and work smoothly.
Innovation and Ambition	10	Goes beyond minimum: interesting data features, non-obvious design decisions, useful extensions, or domain-specific cleverness. May include real-world data, unique workflows, or thoughtful user scenarios.
Code Quality	10	Code is modular, readable, organized, and documented with comments appropriately. Good use of functions, error handling, and file structure.
Oral Presentation	10	Presentation is well-organized and demonstration works. All team members present during demo or at least we know what each member's contributions are.