**Compilation and Implementation Platform**

Compiling Python Code Requires the Anaconda Distribution and PyCharm.

Anaconda - <https://www.anaconda.com/download>

PyCharm - <https://www.jetbrains.com/pycharm/>

Install as stated by the Instructions.

Two Packages are required to be installed for the code to run tkinter and psycopg2. This can be done via PyCharm by opening the project with the python code going to View->Tool Windows->Python Packages.

Compiling the SQL Code Requires PostgreSQL 16 and PGAdmin 4.

All in One Package - <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

Install as stated by the Instructions. Use Port 5432 and localhost (if necessary). Username is Postgres and password is postgres1. Create a database called Books by right clicking and selecting database (no password).

**Compiling and Executing Code**

SQL Code should be run first.

Open PgAdmin 4 and login if necessary. Right Click on the Books Database select Query Tool. Select open file once in query tool, select the Books.sql and click run. Then minimize.

Python Code should be run second.

Open PyCharm and create a project, if necessary, selecting the Anaconda3 interpreter(this should be done automatically). Make sure packages are installed as stated above. Click the Green Run button.

**Architecture Style**

The two architecture styles that could have been chosen to implement this program was either Client-Server or Rule-Based.

In a Rule-Based system queries could be easily solved as rules and the program could implement them accordingly. The system is also straightforward and simple. Any new rules that needed to be implemented to deal with new queries could be easily entered. One the problems is that if too many rules are entered it could potentially affect the performance of the program.

But, in a client-server-based system the data is centralized and offers scalability. Because the data is centralized it allows for easy access by the user if manual changes need to be as well as server access being easily restored if there is any downtime. Users are also only able to enter or remove data and interact with the interface. The problem with this architecture is that any downtime in the server means the user cannot access their data.

For the purposes of the is project, the Client-Server architecture is the better software architecture in this case. The performance of the two architectures is very similar due to the low amount of queries the project will require. The client-Server Architecture is easier to manage and fix any problems as the Rule-Based will require a significant number of rules. Finally, Client-Server Architecture coding can be easily modified to include more clients and an online component compared to rule based.