Team 26 Team Report Summary

- Plan for diving responsibilities: Our team didn't use a specific plan for dividing responsibilities. We based who got what responsibility off of who hadn't done very much up to that task in the project. For instance, Graham and Ryan mainly worked on creating the header files for checkpoint 1, so Sarah created the API documentation since she hadn't worked on anything yet.

- Actual division of responsibilities:
  - Graham Leslie:
    1. Helped with header file creation
    2. Created boost test file
    3. Helped finish the functions in the .cpp files
    4. Created helper functions for reading in the input files
    5. Edited query() and crossjoin()
    6. Edited various functions
    7. Helped with final debugging

  - Sarah Vance
    1. Created API documentation
    2. Helped with creating boost test cases
    3. Created query() and crossjoin()
    4. Made helper function to print tables
    5. Recreated .dll and .lib files for other team
    6. Created user menu in main
    7. Helped with final debugging

  - Ryan Finke
    1. Created a couple of simple .cpp functions
    2. **Did not contribute much at all before he Q-dropped without notifying us!**

  Graham and Sarah ended up doing all the work after the second checkpoint because Ryan q-dropped the class. We still used our original method of dividing responsibilities.

- How we would organize differently: I don't think either of us would have done anything differently. If we had known that Ryan dropped the class earlier, we would have been able to get our checkpoints done a bit faster since we were waiting on him to do parts of the project.
- Major problems: Using the boost library was a bit of a challenge because none of us had ever used it before. Also, having to collaborate with other teams was a challenge because of different ideas about what should be done, different functions, and code not working at times.

Team 26 Evaluation of Team 01's Library

- Communication:
  1. Rating: 5
  2. Team 01 sent all their files to us on time and responded to all questions we had. If there were any problems with our/their code, all questions and problems were fixed quickly and with no bitterness. Email replies were prompt and clear, so communication was perfect overall.

- API Stage:
  1. Rating: 5
  2. The API was clear and readable. We had no problems understanding their well-documented code, and it was a simple API to use.

- Test Cases:
  1. Rating: 5
  2. Their test case was very well written with only 2 or 3 copy-paste typos; they were easily fixed. Very clear, well commented when necessary, and thoroughly tested our database.

- Final Database Library:
  1. Rating: 3
  2. The final database library seemed to work smoothly until we encountered several memory access errors. We were unable to track down the cause with the aid of our TA, so we opted to switch to team 07's final database. We were not sure if it was a problem with our implementation or team 01's database. However, upon switching to a new database, our implementation worked correctly and the program ran smoothly.

- Problem Handling:
  1. Rating: 5
  2. Team 01 was really helpful whenever they had a problem with our code. They told us exactly what the issue was and even gave us some ideas/tips about how to go about fixing them. Whenever we had a problem with their code, they allowed us to make small changes to their code (for example, fixing the typos in their test cases) and anything larger, they tries to fix as soon as possible. Extremely helpful overall and would be pleased to work with them again in the future.