1. **Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**
2. **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**
3. **Discuss if you change the schema or source of the data for your application**
4. **Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**
5. **Discuss what functionalities you added or removed. Why?**
6. **Explain how you think your advanced database programs complement your application.**
7. **Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.**
8. **Are there other things that changed comparing the final application with the original proposal?**
9. **Describe future work that you think, other than the interface, that the application can improve on**
10. **Describe the final division of labor and how well you managed teamwork.**

Stage 6 Report
Team 044: LowerDecks
05/03/2023

1. Based on our initial proposal, the final frontend design is what changed the most. Since many of the features from our initial proposal do not meet the credit requirements, we added some basic functions(CRUD) for the database in the frontend interface. We had planned on making it look more similar to google maps, with a google maps api as the front page and a sort of dropdown menu to do all the tasks we implemented. While we did implement all the tasks, implementing the api like that proved too difficult and not that useful, so we made a basic landing page and an option to show the map. We did, however, get the map to show lines between coordinates, as we had hoped. Our creative component listed in the proposal also proved too lofty. We had planned on making a sort of timeline to view changes of the trading communications over time, but we instead opted to implement the map and the tracings of the communications as our creative part.

2. In the end, our project seems to have succeeded in providing its functionality. It allows users to view the trends of high-frequency trading and their locations, frequency bands, and times. It also allows users to add and delete entries for trading events, which was listed in our goals. What we didn't do was what either changed or turned out to be more difficult, such as creating a timeline or filtering by frequencies, but these aren't detrimental to the usefulness of the application.

3. Our data source never changed; we used the same FCC database publicly available the whole time to work on our project. The schema did change, though, once or twice. Especially during the earlier stages of database design. This was mainly due to us misunderstanding the dataset. For example, the writeup available with the dataset would say one data type, but the type that actually appeared in the data was something different. Another problem we ran into that made us change things up was that we had to

get 5 unique relations, and figuring out how to get 5 relations out of the dataset in a way that would be useful.

4. The biggest problems with our ER diagrams and the biggest changes we made because of them was due to the requirements from the earlier design stages, which is connected directly with the problems we experienced with the schemas in question 3. Other changes we made were due to us either misinterpreting the datasets or because we designed the schema wrong. We have these changes detailed in a `Changes Report.pdf` file listed in the `doc` folder of our repository.
   Another big change we made is that initially, we had many insert statements to add our data into the database. We changed this so that we instead had a single query that inserted hundreds of thousands of lines at once, which sped up our database construction and performance significantly.

5. As mentioned in question 1, we had a few functional changes that never made it into the final version of the project. One was a timeline of high-frequency trades that would show trades on the map over a specified period of time. We decided against this due to the trade-off between usefulness and difficulty to implement. The schema we ended up with does not contain any date or time information so this would not have been possible, anyway. Another thing we changed was the frontend design. The plan was to have the “homepage” be the google earth API, but we opted instead for a more plain landing page with the google maps coming in later. What we added were different, advanced, queries in order to fit the requirements of the project.

6. We made an additional advanced query that returned the geographic center of a collection of communications paths. This could be useful in determining the geographic center of certain communications activities. We also made an advanced query for the google maps section where we selected the coordinates of various paths that allowed us to print them

7.
   a. Amritesh: The transaction query sent from the front end didn’t work at the beginning, while all the other queries worked. And it turns out that we need to set multipleStatements to true for the mysql connection.
   b. Cheng-Han: The Google Earth API is no longer working, so we just use a normal Google Map API, and it is kind of complicated to directly input coordinates’ information right into the API from the backend. But we managed to transfer the data into a temporary KML file and feed it to the frontend.
   c. John: One technical challenge that we encountered was getting Node and React to work on all of our machines. The biggest part of this was that the documentation online was never really that accurate or helpful in getting it set up or running. The docs say run `npm run install`, or something, but what worked best was just `npm i`, which installed everything very easily. From that point, we

could either run a node command `node server.js` or use the react/npm command `npm run start` and the interface would be up and going on `localhost`

d. Justin: We had issues getting the data quickly into our table. When we sometimes loaded things into the table, we used a single query that inserted hundreds of thousands of lines quickly, instead of multiple independent insert statements.

8. Nothing else changed other than what we've already mentioned.

9. Future work that would be interesting is adding the timeline mentioned in the proposal and also in earlier questions, but this might require a rework of the database design to include time data. Also, some filters of the dataset could be applied on the frontend to update the maps API, which might show some interesting results.

10. The final work distribution came out to this:
    a. Amritesh managed the GCP server and worked on the frontend
    b. Justin worked on entering data into the database, database design, and frontend
    c. Cheng-han worked on the logic for the backend and building the frontend
    d. John worked on backend and database design
    e. Everyone worked on all the reports, such as the proposal and ER diagram explanation

    When working together as a team, the work was often actually very smooth. We found success in discord calls and streaming screens, which allowed us to work on the same thing from almost the same screen. Also, we'd assign ourselves parts of the project to do once a week or every two weeks, and work on that, often in pairs.

Here is the final video demonstration:
https://drive.google.com/drive/folders/1XfemxeiRY89dCccCzG4RjZk19d1b7jfF?usp=sharing