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).**

   In our initial project proposal, we mentioned adding pictures of birds and tools for visualizing bird sightings. We ran out of time and could not get to adding pictures for each bird. We also came to an agreement that instead of adding visualization for bird sightings, we will have a "Bird Stats Explorer" where a user has the ability to discover unique species in a specific region, a most frequently seen species in a region, overlapping species in two regions that a user selected, and rarity of birds.
   In terms of datasets, we ended up using two out of the four that we had. We all agreed that from that the two datasets we chose had enough information to grab from the datasets to populate our data tables.
   The layout of our website was also changed. We now have an initial log in page and once a user logs in it takes them to the "Stats Explorer" tab. We have four tabs in total, "Stats Explorer", "Add Sighting", "User Info", and "Map".
   Lastly, we added a feature where a user is able to delete themself and their sightings from the database.

2. **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**

   Regarding usefulness, I think our application achieved what we wanted. We wanted a website where a user is able to view/add bird sightings and learn information about the bird. Also, we wanted an actual map UI on the website as a visual component, which we successfully added.

3. **Discuss if you changed the schema or source of the data for your application**

   We did not change the source of the data. We used two out of the four sources of data that we initially listed in Stage 1.

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?**

   We had made a few changes to the ER diagram/table. In our Location table, we got rid of country code because we felt it was unnecessary. All of the sightings we had in our data were from the US and we did not feel like we needed it for any display in the frontend or queries. In our Location table, we got rid of the life_stage and sex. Similarly to the Location table, we felt that life_stage and sex were not useful for the queries we had so we removed them. Also, we changed taxon_id to species_id. For our TAXON, we also changed taxon_id to species_id and removed scientific_name. For the User table, we removed name and added email and password. This addition was necessary because of our decision to add a log in page.

5. **Discuss what functionalities you added or removed. Why?**

   One functionality that we got rid of is the Info tab. Our idea beforehand was that a user could search up a bird and look at the taxonomy based on that. Instead of that, a user is now able to see information about the bird that they recently added. For the Info tab, we substituted a Stats Explorer page where a user can see unique species for a region and

overlapping species in two regions.We removed the search feature because we felt it would be more important for a user to know information about where the bird was found, rather than being able to search up the taxonomy of a bird, especially since that could easily be done with a simple Google search.
Another functionality was the ability for a user to see pictures of the bird. We simply did not have enough time to implement this function.

6. **Explain how you think your advanced database programs complement your application.**

   Our advanced database programs really helped support our application. For example, we used stored procedures to handle more complex queries, like finding all the unique species in a region or finding which species overlapped between two regions. Instead of running those queries from Django every time, having them in the database made things faster and cleaner. We also used a transaction for retrieving the user info, stats, and the leaderboard table (2 advanced queries we presented in stage 3) and set the isolation level to read committed to avoid any dirty reads that might transpire while retrieving the data. That helped us keep the data consistent. Lastly, we added a trigger to insert new records to the habitat table if those records do not already exist on the table. That way, we didn't have to write extra logic in our backend to manually insert records to the habitat table, it happens automatically. Overall, these features made our app run better and made sure the data stayed accurate even when users were interacting with it a lot.

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.**

Viki: We had a few challenges in connecting Django to our MySQL database hosted on GCP. Our first issue began with the actual connection to the database since it kept on failing. We then realized we needed to change items in settings.py, specifically under DATABASES. We updated the engine, name, user, password, host, and port fields with the correct information and that fixed it. Now, we were able to establish a connection, but we could not access anything from the database. For example, querying something simple such as getting all the users would not work. We then realized that we needed to add our IP address to GCP as authorized IP addresses. Once this was done, we were able to access our database.

Furkan: Considering all the stages we have been through so far, I would say that setting up the MySql instance and our tables on GCP was very challenging. First, we had to load our datasets into a few tables and then make all the tables while satisfying all the requirements. We had a problem loading one of our datasets into a table because the way GCP's import data mechanism works is that it only accepts CSV format files. Our dataset was a TSV so we had to manually convert it into a CSV file. We also had to change our dataset a few times because there was a requirement that mandated at least 3 tables to have at least a 1000 records each; so, we had to maneuver to larger datasets. Also, similarly to what Viki said, connecting to our database became difficult from time to time, as we were travelling due to work\internship for some time and had to change our IP addresses a few times just to work on the project.

Sahil: One technical challenge I encountered was using Django's cursor object to execute queries. I was used to interacting with the databases through ORMs but since we were restricted from using them in this

project we had to use the cursor object to execute raw SQL queries and it was really different from what I was used to. Since we did not have a model/mapper we had to extract each attribute individually and process them in chunks. It was a great learning experience but took all of us a great amount of time getting used to it.

Roberts: A technical challenge I encountered was creating an SQL query that could dynamically change based on the presence or absence of optional filters, such as, species, locality, and limit, while displaying these markers on the map.

In the beginning, the backend query did not account for the user leaving certain fields blank. Therefore, leading to zero markers being displayed. This was since the backend was written with fixed "WHERE" clauses. Essentially, the user would filter for Locality Arlington (TX) expecting to see all the birds in this location - for example - but the database would be searched for "WHERE locality = Arlington (TX) AND Species = NULL AND limit = NULL". Since no sightings should have such parameters (based on our database design), the map returned zero markers.

The solution I implemented was as follows, I created a dynamic WHERE clause builder where the starting conditions would be "latitude IS NOT NULL AND longitude IS NOT NULL". Then I would append conditions that are specific to the filter parameters entered by the user. For example, if the user filtered for Arlington (TX), I would append "AND locality = Arlington (TX)" to the SQL query string. Repeating the process if other filter parameters were filled. Then, finally, querying the GDC SQL Database with this query and displaying the results.

The advice I would give future teams is that they should avoid hard coding SQL queries for cases where the inputs are optional. They should implement a dynamic approach to crafting the query like I described above.

8. **Are there other things that changed comparing the final application with the original proposal?**

   Another thing that was changed besides adding a new tab called “Birds Stats Explorer” was we added a rarity to the tab as well. This utilizes our rarity table that we implemented and it was not proposed initially.

9. **Describe future work that you think, other than the interface, that the application can improve on**

   Something that we can improve on is how we store the passwords from each user in the database. Right now, when a user signs up and enters in a password, it just goes straight to the database and gets saved as is. We can improve on this by creating a hash function, hashing the password first, and then putting it into the database. This would make our application more secure.
   Something else that we can improve on is adding a more user friendly error. Right now we just have it showing a red error, but in the future we could change it to be a bit more nice looking.

10. **Describe the final division of labor and how well you managed teamwork.**

    Regarding teamwork, we all feel that we managed it pretty well. We met outside of class to work on our project and if we ever had any questions they were answered right away.

The final division of labor was first we had a meeting to set up the initial website and connect it to the database. Once that was good, we divided the work amongst ourselves. Viki took on the role of adding stored procedures as well as making the “layout” of the webpage (which included just creating tabs and title), Sahil worked on the deleting user part and on the add sighting tab, Furkan worked on the transactions which was the User Info Tab, and Roberts worked on implementing the UI map on the Map tab. Additionally, we all helped each other if someone was stuck on their portion of the project.

11. **To get some points back in the earlier stages, you can add a section with two subsections for Stage 2 and 3 respectively. In each subsection, you should mention the state of your design/implementation before the feedback, the feedback itself and the change you have made as per the feedback. If there was no feedback for a stage (or if you have not addressed anything), you can leave the corresponding subsections empty. (Optional)**

    For our initial User table in stage 2, we had only user_id and name as the attributes. For our feedback, our TA said “User should have email, password.” For the change, we added to our User Table an email and password. Also, for stage 3, we got a feedback regarding how the user email has to be unique. So, we changed the constraint of user email to UNIQUE.