Project Report:

1. Please list out changes in your project directions if the final project is different from your original proposal (based on your stage 1 proposal submission).
    a. **The only major change we made (compared to the stage 1 proposal) was to drop the "media" table, since we found that we could store media differently or live-grab images through our frontend. We also chose to drop a few relational tables as our project progressed, because we found our database no longer required them.**
2. Discuss what you think your application achieved or failed to achieve regarding its usefulness.
    a. **I think our application achieved its original goal of informing users of upcoming games, but it does fall short in some regard. We don't currently have a function to sort games, so you're stuck viewing the top rated or anticipated games unless you know the name of the unreleased game. Beyond viewing the data, I think our application succeeds in every field it set out to do.**
3. Discuss if you changed the schema or source of the data for your application
    a. **We ended up staying fairly close to the original schema, other than minor changes we made to organization, and we stayed with our original source of data.**
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?
    a. **The change we talked about earlier (dropping the media table and eliminating some relational tables) were the main changes we made to our ER diagram, and we believe they're for the better. Our newest iteration works the best for our goals, since SQL was not needed to handle media, and we could make it even better by implementing a NoSql database to handle media.**
5. Discuss what functionalities you added or removed. Why?
    a. **We added some functionality, like moderator-submitted games being instantly approved instead of awaiting mod approval, or changing up the layout of our UI to allow the user to view the most anticipated games as soon as they click onto the site. We actually didn't remove any functionality, because we attempted to stay as faithful as possible to our schema.**
6. Explain how you think your advanced database programs complement your application.
    a. **They provide the user with interesting stats they might use to decide what games to play. For example, our company grading program**

allows users to see letter grades attributed to companies, so maybe they can decide to play games by "B" ranked companies. Or, we have a trending users list, so users can shoot towards getting on the list or find games submitted by those users.

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.
    a. **Nick: I had some problems importing our dataset into GCP, because the CSV files were formatted incorrectly (some strings didn't have quotes, so separate words were treated differently and sentences couldn't be imported), so I had to manually extract the data from the CSV and import it into GCP differently. Some of the CSV documents worked as intended, but 2 or 3 didn't. I ended up using VSCode to generate a few thousand insert statements and pasted them all into the terminal at once because the CSV import issue couldn't be resolved.**
    b. **Anthany: One major challenge we faced was dealing with our way of handling submissions, which was designed to be separated from most of the code and be able to store the schemas of all other "submittable" tables. This made it very difficult to regulate what the users submitted automatically since the submitted data was stored in a string to be more versatile and so we could not access the information that the users submitted easily from the database without parsing the data in frontend first. To avoid revising all of our submissions code, we had to modify other tables in order to make our ideas for automation work, such as adding a "company name" attribute to a "worked on" relational table in order to make create a trigger that creates a new company if the "developer" of the newly submitted game does not exist. While adding one trigger did not require too much modification, if we wanted to implement more triggers, we would have to make even more changes to our tables to get around the issue of not being able to access the data the users submitted. Creating submissions in such a way did not really give us a noticeable benefit, except maybe for organizing the database a bit better, while there are other ways to set up submissions that would have made creating triggers, queries and stored procedures much easier. This may be an example of SQL's inflexibility, since "Submissions" schema had to be flexible, but SQL could not**

accommodate this very well, and other languages like mongodb may have worked better in this case.

c. **Kendrit: One technical issue that we faced was the creation & implementation of the Stage 5 Stored Procedure. Specifically, designing a stored procedure which respected all requirements given by the teaching staff turned it into a ~7 second query time. When implementing this in our NodeJS + ReactJS web app, it made the entire web page have random bugs. Our stored procedure would automatically run on the load of the page, and since the backend get & post requests using Axios were not parallel, the rest of the backend operations would hang until the stored procedure was finished running. Since the information was not essential to the operation of the app, we were able to change it from autorunning to a button that would run the procedure on click so that it would only be run on user request. This removed all the bugs we saw with the sequenced backend operations, and gave the user more context on what was happening in the app. In the long run, this would also optimize query and database read/write counts if that would be a concern. We would also eventually optimize the SQL commands in the stored procedure.**

d. **Sean: Some of the challenges we faced were that designing stored procedure and inserting csv file into MySQL. Since I focused more on backend side, I didn't know much about how the frontend interacted with database which made it hard for me to design the stored procedure. I had to think about the usefulness of the stored procedure and how the users would be able to interact with it. The way we solved this issue was by communicating more with our frontend developer and building off his idea. The second issue I met was inserting csv file into MySQL in my terminal. We ran into many issues including duplicate primary keys and unknown characters which was not (UTF-8). The way we solved this issue was just manually going over the csv file and changing/deleting broken entries.**

8. Are there other things that changed comparing the final application with the original proposal?
   a. **Generally, nothing changed that dramatically besides what was outlined in question 2.**
9. Describe future work that you think, other than the interface, that the application can improve on

    a. **We want to allow users to be able to search via more terms, as well as implement a tagging system for game genres to allow users to toggle tags while searching. Additionally, we'd like to implement a comment system, optimize our advanced queries to improve time, and generally clean up the database (get the whole project on gcp).**
10. Describe the final division of labor and how well you managed teamwork.
    a. **We ended up dividing labor stage-by-stage. It was generally volunteer-based, and whoever did more work on the previous stage would take a backseat for the current stage. Throughout the project, all of us actually commented on how much we liked the work distribution.**