- **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).**
    - We added a table to the database schema to include the players in the fantasy team which basically links the FantasyTeam to its roster.
    - We create the pages we stated for analysis and the 4 advanced queries we implemented allow the player to gather analytics and insights.

- **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**
    - I think our application achieved what we were aiming to do. It makes it easy to add players to your team, view their stats, and get useful information like the highest scoring players or the most consistent players.
    - Our application also makes it very user friendly for creating teams on the fly for someone to analyze quickly and efficiently.
    -
- **Discuss if you changed the schema or source of the data for your application**
    - We added a table to the database schema to include the players in the fantasy team. This was necessary to add players to our database. We ensured that the table would get deleted when a team would get deleted.
      ```
      | id              | int  | NO   | PRI | NULL    | auto_increment
      | fantasy_team_id | int  | NO   | MUL | NULL
      | player_id       | int  | NO   | MUL | NULL
      ```

- **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?
    - Our change to the ER diagram was just adding that new table mentioned above. I think we needed this for the code to work easier because having a table that stored the fantasy team roster associated with the fantasy team Id was helpful for us. Design wise it is a very minor change but it helped us actually code the database easier.

- **Discuss what functionalities you added or removed. Why?**
    - Search bar
        - Now when a user goes to manage their team, instead of manually looking for their players, they can use a search bar to quickly filter who they're looking for

    - Login/Logout
        - Created functionality for different users to sign up, sign in to the app, and sign out
    - Create Fantasy Team
        - Letting users create a fantasy team, this is then stored as a unique Fantasy Team in our database so they can add players to it.
    - Delete Fantasy Team
        - Being able to delete teams if the user wishes.
    - Highest Scoring Player
        - We have functionality for users to find the best players by points.
    - Most Consistent Player
        - We have functionality for users to assess the most consistent players week to week
    - Total Points
        - Users can see a players total points cumulatively for the season
    - Manager page where you can see all your players.
        - Prints out teams with clickable link where you can view all players
        - You can click on a player name to view the players stats as well which runs our transaction

- **Explain how you think your advanced database programs complement your application.**
- Transaction: We included a transaction that checks to see if a player and a game exists when obtaining the player statistics in a game. This transaction ensures that we are not pulling garbage data, and that our database is performing the function we want.
- Stored Procedure: We created a procedure to obtain the player statistics per game. In our front end, we have a page that displays player statistics. This page obtains the data through our stored procedure. The stored procedure is efficient and is run every time the user tries to get statistics for different players.
- Trigger: We incorporated a trigger that checks if the number of players in a team is less than the roster size. When the user creates a fantasy team, they choose the number of players on their roster size. When adding players to their team, the trigger does not allow the user to add more players than the trigger size. If the user tries to add more players

than their roster size, the page displays a warning message and does not allow the user to add more players.

- Constraints: We have defined primary keys and foreign keys in our tables. These constraints help maintain our database schema and obtain data when tables are related to each other.

- **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.**
  - **Antanas:** One challenge I faced was figuring out how to layout the pages and clean the data to be inserted into the database. Our group wanted to avoid generating data whenever possible, so we focused on cleaning existing datasets. I found that Kaggle was a great resource for locating large, competition-grade datasets that were both diverse and detailed enough to suit our needs. To address this challenge, I created helper scripts in Python using Pandas to clean and preprocess the data, ensuring its consistency and integrity. I then exported the cleaned data into CSV files, which could be easily inserted into our database. This process not only helped maintain accurate analytics but also reduced redundancy and improved the overall reliability of the system. My advice to future teams would be to dedicate time early on to data cleaning, as it's crucial for ensuring high-quality result.
  -
  - **Varshitha**: One big challenge our team faced was getting GCP to work and setting up a shared database. At first, we had trouble figuring out how GCP works and which services we actually needed for the project. Setting up the database so everyone on the team could access it was tricky because we had to deal with permissions and make sure it was both secure and easy to use. On top of that, connecting the database to the front-end was hard because we weren't sure how to make everything work together and show the data correctly. For future teams, we recommend starting by watching GCP tutorials provided through the class or online to understand the basics. When setting up the database, have one team member create the database on their account and share the details with the rest of the team. In addition, to access the front end connection, you need to add your IP to the GCP console under SQL, in Connections, under Networking.

- ◦ **Naman**: When uploading the data to GCP, we encountered issues when uploading datasets that relied on the foreign key first. We first had to upload the data that had only primary keys and then the data with the foreign keys since it relied on the previously uploaded data. This process ensured that we had the correct references between our datasets. We also had to make sure that the schema in GCP matched our data's schema. Since we kept making changes to our data, we needed to make sure that our schema was consistent across all platforms. For future reference, make sure that the data upload order is established beforehand to avoid errors.

  ◦ **Ashrith:** We initially had trouble getting the git repository to work correctly for all four of us. The main issue was that we initialized it wrong by dragging files into it and not enabling the ssh keys. For future teams, thoroughly reading the correct way to initialize the repository can help avoid issues like this when starting the project.

- **Are there other things that changed comparing the final application with the original proposal?**
  - Besides adding some basic functionalities and creating that new table for our backend to work easier, we didn't really make any other changes besides that. Compared to our original proposal, we are missing one feature. We didn't include functionality to directly compare different players with each other. Instead users are able to search each player up individually and compare them themselves.
- **Describe future work that you think, other than the interface, that the application can improve on**
  - Ideas: The player stats could be more detailed. We could use AI to suggest ideal teams. In the future, we can use AI to update the statistics on the players, based on team changes, injuries, and historical performance.
- **Describe the final division of labor and how well you managed teamwork.**
  - Each of us implemented one of the advanced queries and ran analysis on GCP.
  - We implemented the front end of the database together, making changes to the database when needed. We set up times throughout the week and got together in person to work on the project.
  - We met together to write the final report and clearly communicated how to split up the final document. Our division of labor s
  - We used iMessage to communicate and stay alert.