- **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).**

The direction of our project has not changed from our original proposal. Our project, BBDB (Basketball Database), is a tool that harnesses the vast amount of data in professional basketball and allows the user to choose and make queries that suit their needs. We have data spanning 14 seasons, with thousands of rows for players and games, as well as over a million rows for shot data, which allows us to calculate things such as shot percentage or points scored for players as well as teams in a variety of time frames.

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

Our application certainly achieved being generally useful; we have a large amount of data spanning many seasons, with everything from team rosters to game scores to every shot taken over the course of each season (millions of database rows). This comprehensive data allows us to provide a good range of statistics to users; we can get different statistics for players, teams, games, and seasons, as well as related searches. The main aspect that our application "failed" to achieve (due to its large complexity) is the ability to parse any plain English input. This is discussed in more detail in a later part.

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

We did not change the source of the data for our application. We relied on Basketball Reference, and web scraped it with various Python scripts to get the data we needed for our different tables. Our schema did change slightly from the initial version to what it is now. We did not add or remove any entities (User, Player, Team, Game, Season, Shot). Instead, during our process of collecting all of the data, we realized that the data types of some columns needed to be changed to accommodate the web scraped data, and we also added some columns to give us more options when it came to functionality.

- **Discuss what you change to your ER diagram and/or your table implementations.**

Our ER diagram has not really changed since the original version. We've kept the same entities (User, Player, Team, Game, Season, Shot), and their foreign keys have remained the same, with simple dependencies that are needed to verify that data is accurate (User's favorite player and team must exist in Player and Team respectively, for example). Our database schema and the resulting table implementations were only

changed slightly to accommodate new columns, and this was all done in stage 3, so everything is up to date.

- **What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

Besides the slight schema changes, which don't really affect the user experience, the main differences between the original and final designs are in the functionalities. There's not really a more or less suitable design since the direction of our project hasn't changed.

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

The main functionality we removed was the visualizations and shot charts. This was unfortunately due to time constraints; we are all busy students, and with the various challenges we faced throughout the project (discussed later), we were unable to implement these visual elements. We chose to focus on providing the user with a wide variety of queries, since the data and statistics are still the core of the application, with the visual elements being an addition to enhance the user experience. However, other than these, we were able to deliver what we hoped to accomplish.

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

We have several advanced database features that help complement our application. For stored procedures, we have several different stored procedures that are used to make it easy to call queries that are often used. For example, we have stored procedures to add, delete, and update User information. For transactions, we can execute two advanced queries at the same time, and then store both of their outputs for the user to access. For triggers, we're adding checks to make sure that the user selects an existing favorite player and favorite team when they create a new account, as well as when they update their favorites. For constraints, we have our primary keys and foreign keys defined in our database schema that enforce data accuracy (for example, User's favorite player and team must exist in Player and Team). We have additional constraints on the User data, including minimum password length and maximum username length.

- **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.

Max: One technical challenge that we encountered was when trying to use SQLAlchemy to actually interact with our database from Python, which was necessary both for inserting our initial data as well as for the project's backend. There were many issues that arose with permissions (our database is on Google Cloud Platform), and so we ended up even having to mess with the IP addresses that were allowed when trying to connect to the database.
Abhi: One technical challenge that we faced was all of the difficulties involved with web scraping to get all of our data. For instance, rate limiting meant we had to add sleeps into our code so we could continue, which of course affected how quickly we could obtain the data we needed to test. Additionally, using an HTML parser wasn't completely straightforward due to many players being from foreign countries and having special characters in their names, which meant some names couldn't be recognized easily. Some web pages also had different structures, which further complicated things.
Michael: One technical challenge that we encountered was with Google Cloud Platform closing billing accounts for seemingly no reason. Even over fall break when we hadn't worked as much on the project and made very few calls to the database, our billing account would get closed and force us to figure out a new option. We've switched accounts three times without the credits being consumed or expired, and so trying to keep our database accessible has been an unexpected challenge.
Brian: One technical challenge that we faced was setting up the sql database in Google Cloud Platform. None of us had any experience using Google Cloud Platform so it was a bit difficult to find the right places to do the necessary setup. For example, at first, we started our database sql instance in the compute engine itself, instead of in sql with gcloud. Also, we had to change some local sql settings to allow some of our advanced queries to work, such as disabling sql_mode=only_full_group. Essentially, there was a bit of a learning curve around Google Cloud Platform that we had to go through.

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

There isn't really anything else that's changed when we compare our final product with the original proposal. The direction of the project has remained a comprehensive basketball database, and our changes in functionality have been covered. It was hard work obtaining millions of rows for our database, and we may not have been able to meet all of the ambitious ideas in our initial proposal, but in the end, we were able to create an application that works well.

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

Future work that comes to mind is increasing the complexity of user queries that we can handle. Ideally, the user would be able to input a plain English sentence that describes their desired query; for example, they might ask “How well does LeBron perform against the Golden State Warriors?”. This doesn’t exactly have a single correct interpretation of what statistics they’re looking for, so we might look through all of LeBron James’ past games again the Golden State Warriors and show his average statline (points scored, rebounds, assists, etc), as well as some visualizations of how his average stats change from season to season. This of course would be incredibly difficult; something like natural language processing or API calls to a GPT model might be necessary to parse the user input reasonably. Since we have data stretching back many seasons, and can update the database as new games are played, we have all the data any user might desire, and the main limitation is how well we can support these desires.

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

We worked very well as a team; we are friends and so there weren’t any team conflicts. We met regularly and made sure to communicate throughout the process of each stage so that we were always on the same page. Our division of labor looked a bit different for stage 3 compared to stage 4. For stage 3, where we had to actually get all of our data, set up our database, and create and test advanced queries, we each focused on a different area before coming back together to finish the stage. For instance, some of us worked on web scraping the data and using Python to actually update the databases with the data, while the rest of us worked on designing the advanced queries and testing them along with indexing optimizations. For stage 4, we often live shared so that we could code on the same file simultaneously, which allowed us to discuss different approaches and test how our frontend looked as we worked, which helped tremendously. Overall, we were a strong team that worked cohesively and were able to get everything done efficiently.