Please list out changes in directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

The final project differs from the original proposal in the following ways:

- It features a recommendation for/against outdoor seating.
- It features a “Data Insights” tab, which has visualizations of the dataset we used.
- We decided against incorporating sentiment analysis into our application, as the rating already quantifies the review sentiment.
- We do not repeatedly call the weather API, as we realized that doing so would be costly.

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

In general, we were very satisfied with the outcome of the application. We were able to accomplish our goal of providing insights into businesses centered around major cities (like Philadelphia, Tampa, Tucson, Indianapolis, etc.). We hope that users can utilize the application to gain a greater understanding of businesses in their area through looking up information like common attributes, business hours, and top reviews. However, we believe that we could've found a larger dataset to increase the usefulness of the application. The dataset that we ended up using for the application only provided insights into a small subset of cities in the United States, rather than providing insights on every area code. Choosing a larger dataset would ensure that users all over the country will be able to gain insights on their area, rather than just those in major cities.

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

We largely did not change the schema or source of data of the application. As outlined in our proposal, our source of our data was ultimately both weather API data and the Yelp Kaggle dataset. The only change to our schema was that we added an additional table (LocalWeather), which our stored procedure creates.

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 primary change to the ER diagram was the addition of a table—LocalWeather—which has the postal code as its primary key. Its other columns are average temperature and average precipitation (averaged over a year). Before adding the LocalWeather table via a stored procedure, we needed to execute two advanced queries each time the user inputted a zip code into our “Outdoor Seating” tab. This new design is more suitable for our application; we can instead create a LocalWeather table when the page loads, and we then execute a single, far simpler query whenever the user inputs a zip code.

Discuss what functionalities you added or removed. Why?

We changed the weather-related functionality of our application. We added functionality which recommends whether business owners in a particular area should incorporate outdoor seating into their business model. One goal we had in creating this application was to allow businesses to better their business model based on cost considerations. This feature allows us to do this, as we can recommend against investing in outdoor seating in areas in which businesses would be able to use it for only a small number of months. However, we removed functionality to determine if there was a correlation between weather and ratings (as in, whether businesses might receive higher ratings in warmer months). We realized that this information may not be very helpful to businesses (as their general location is likely fixed).

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

Both our trigger and stored procedure were useful tools created to make life easier for businesses using our application. Our trigger made sure that our UI wasn’t too cluttered by having a default time (9-5) that new businesses would be assigned. For the majority of businesses we will have standard times and we thought that to save our users time we would use a trigger to update common information between all created businesses. In case a business needs to modify their hours later we can see that we have a way to do this manually. However making all businesses manually input hours when they don’t have to seems unnecessary.

For our stored procedure we wanted to give additional information to our users. For example if a business can open with outdoor seating they might be able to purchase a smaller building (because they don’t need indoor seating). We used a basic free weather API to check (for a specific postal code) if the average temperature was above 70 degrees and there was less than

5 inches of rain a month. With a better weather API (that costs money) we could get more detailed information but as a starting point we implemented this stored procedure to show that this concept was possible and works pretty well recommending outdoor seating for businesses farther south (where it is warm) and this procedure also returns the warmest month of the year. By finding the warmest month we can better target when a business might want to use outdoor seating

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.

Vrushhabh – A technical challenge I faced was parsing through the huge Yelp dataset and adding the contents to MySQL. It would take a long time to import from the buckets so I reduced the amount of data used. Another issue I faced when setting up the backend is that one of my partners was having an issue calling the endpoint from the flask server from the front end due to a CORS issue so we imported a library to solve that problem.

Maya – A technical challenge I had was in creating the CategoryCounter and ReviewCounter classes which, along with the API, connect the frontend and backend of our application. It required accounting for edge cases such as the deletion of the first review in the Reviews tab and invalid user input (e.g., the user inputted invalid indices to the get_review() function).

Michael – One technical challenge that I encountered was the implementation of user inputs on the frontend. With any web application, handling user inputs is extremely difficult as you have to consider cases like security risks and invalid inputs. However, it was especially difficult with this project as we wanted to make sure that we did not send in invalid inputs to the database and cause any undesirable side effects. Additionally, because our project was constantly changing, I had to keep flexibility in mind when creating the frontend. To create the user input fields, I utilized React state. Whenever the submit button is clicked on one of the forms, the user input is validated and then calls the appropriate backend route using Axios. This was especially tricky on the "update business hours" section of the site where we had many parsing issues with the business hours (because of the time format in the data set).

Zach – We spent a lot of time writing and debugging stored procedures during stage 5 because we did not have a lot of experience writing them in comparison to indexing or triggers. For this class we only have one GA example of a stored procedure (with a cursor). In addition, in our implementation we required 2 cursors and had to manage exit loops for both of them. For other groups I would recommend starting with a basic stored procedure only using one of your advanced queries before attempting to integrate the second query (and cursor). MySQLWorkbench was a great tool since you could insert the stored procedure in a test table and debug using the built-in error messages. By referencing the lecture notes, the error messages, and building slowly we were able to get the stored procedure done faster then just trying to do the whole thing at once.

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

Our user interface changed from the mockup in our project proposal. While the general layout of the home page is unchanged, we now have additional tabs and a different color scheme.

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

Although we are quite satisfied with the end application, we believe that there are a few areas that could be improved upon. Specifically, if we had unlimited resources and time, we would likely invest in a better weather API to obtain more data for the "Outdoor Seating" page. The current API that we are using only provides limited information like precipitation and average temperatures. If we had more data points, we could expand upon the "Outdoor Seating" algorithm that we developed to base it on more factors and make it more robust. Another way that we could improve the application is by obtaining more business data. The dataset that we used in the application did not contain every single area code, but rather focused on a few major cities. Expanding the data set would allow users to gain insight on even more areas around the country.

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

Fortunately, our team worked very well together and was able to communicate well. To accomplish the project, we created a group chat and regularly met to work on the project and delegate tasks. Everyone in the group took an equal portion of work, but each of us focused on areas of the project that we had the most experience with. Michael focused on the frontend, Vrushabh focused on the backend, and Maya/Zach worked on the database implementation and design for the project.