

Katelyn Wolberg
Lauren Watsky
Jamie Zuckerman

SI 206 Final Report

Link to GitHub: <https://github.com/lwatsky/final-project.git>

Project Goals and Goals Achieved:

Our original goal for this project was to test whether the top 20 restaurants in 5 different college towns are rated similarly across three different rating platforms. We found that out of the 250+ restaurants that we collected, only 32 restaurants were in the top 20 on two different rating sites and only two restaurants were on all three of the rating sites.

We used three APIs: Yelp, Google Places and Zomato. From each API we found the top 20 restaurants in Ann Arbor (University of Michigan), Austin (UT Austin), Bloomington (Indiana University), Gainesville (University of Florida) and Madison (University of Wisconsin).

Problems Faced:

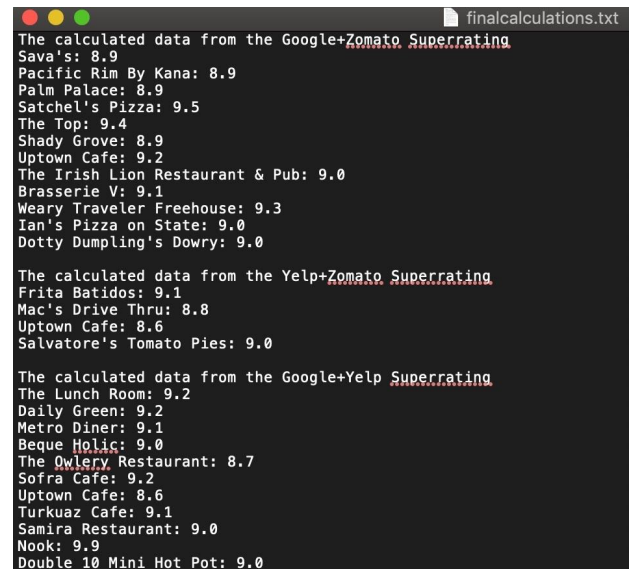
One problem we faced is that we originally wanted to use big cities, like New York and Chicago. However, since the cities are densely populated and filled with thousands of restaurants, there were barely any restaurants that were found in the top 20 list between the three APIs that we chose. We decided to switch to smaller cities where there are less restaurants in hopes for more crossover.

We also were originally planning on using the Foursquare API, but after research, found that this API was hard to access, so instead, we switched to using Google Places API.

Another challenge we face is the continual changing of our data. Google Places, Zomato, and Yelp are constantly updated, so our databases, and visualizations often change when we re-run them.

Calculations File:

This file puts our calculated “Super-Rate” data into a text file. It shows the restaurant and its associated rating. This file shows us if our calculations were correct.



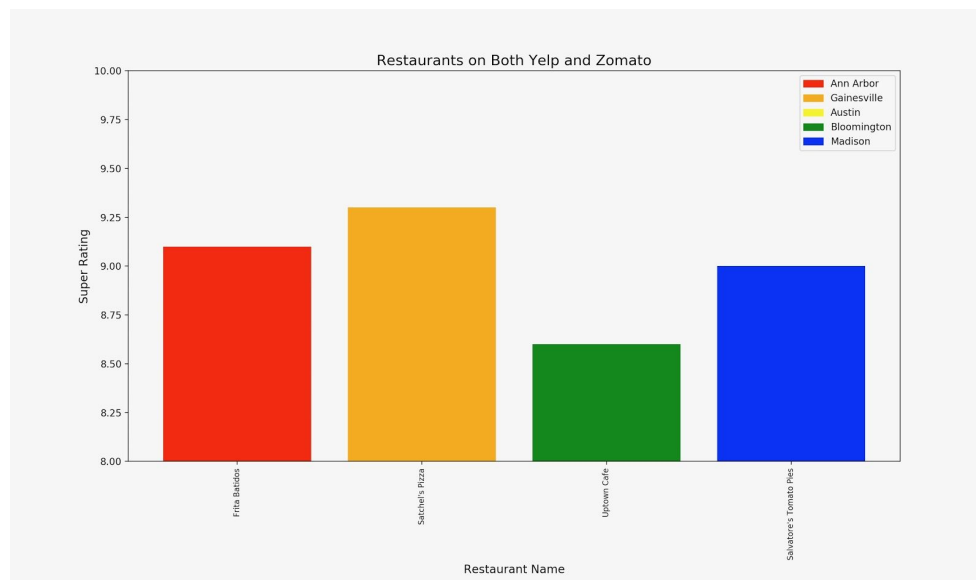
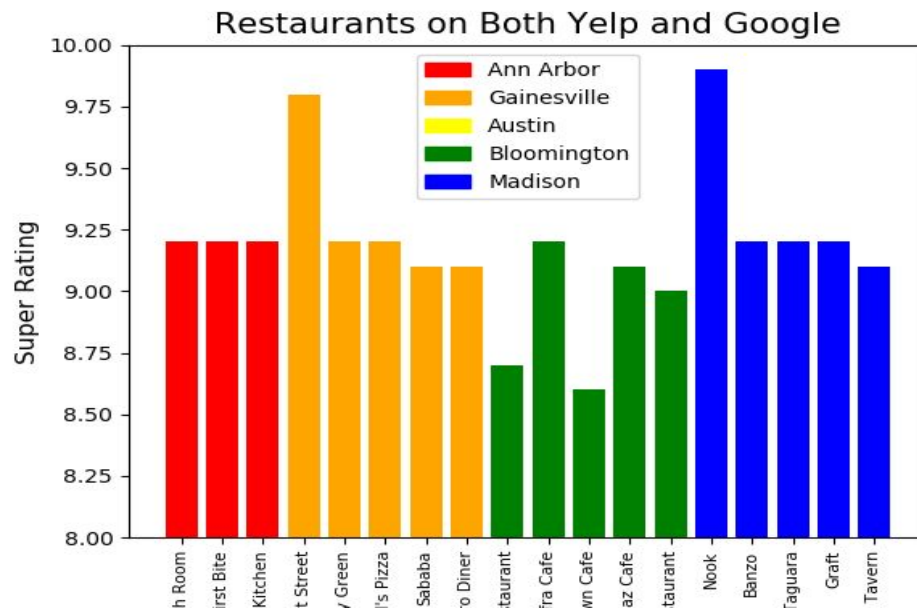
```
finalcalculations.txt
The calculated data from the Google+Zomato Superrating
Sava's: 8.9
Pacific Rim By Kana: 8.9
Palm Palace: 8.9
Satchel's Pizza: 9.5
The Top: 9.4
Shady Grove: 8.9
Uptown Cafe: 9.2
The Irish Lion Restaurant & Pub: 9.0
Brasserie V: 9.1
Weary Traveler Freehouse: 9.3
Ian's Pizza on State: 9.0
Dotty Dumpling's Dowry: 9.0

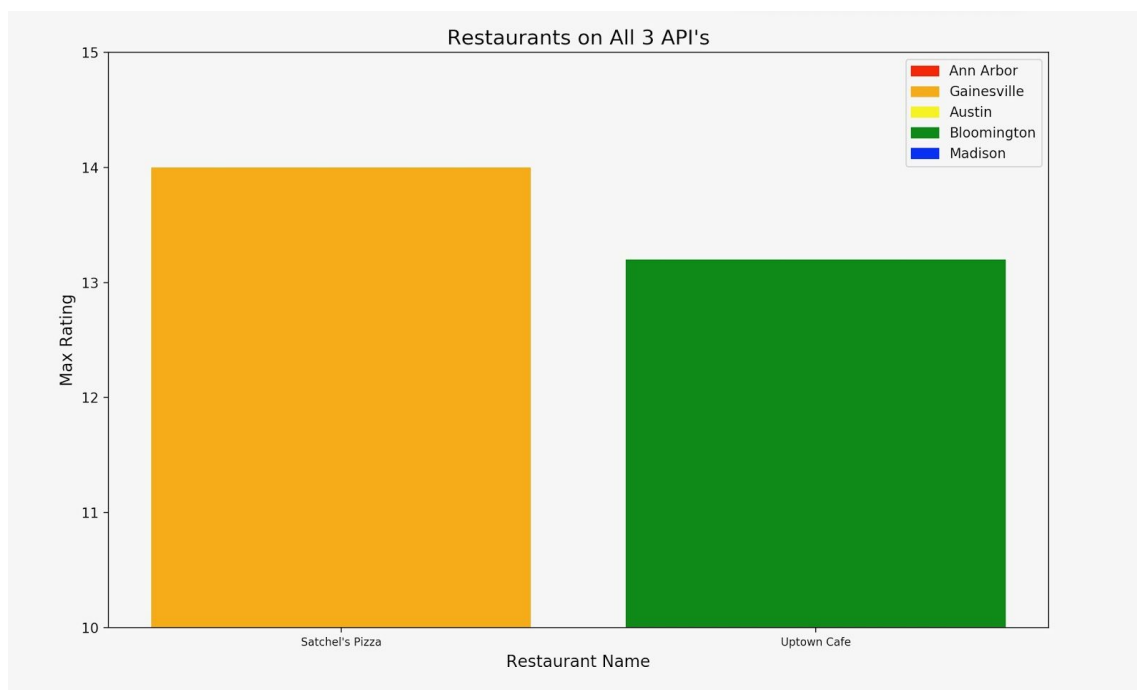
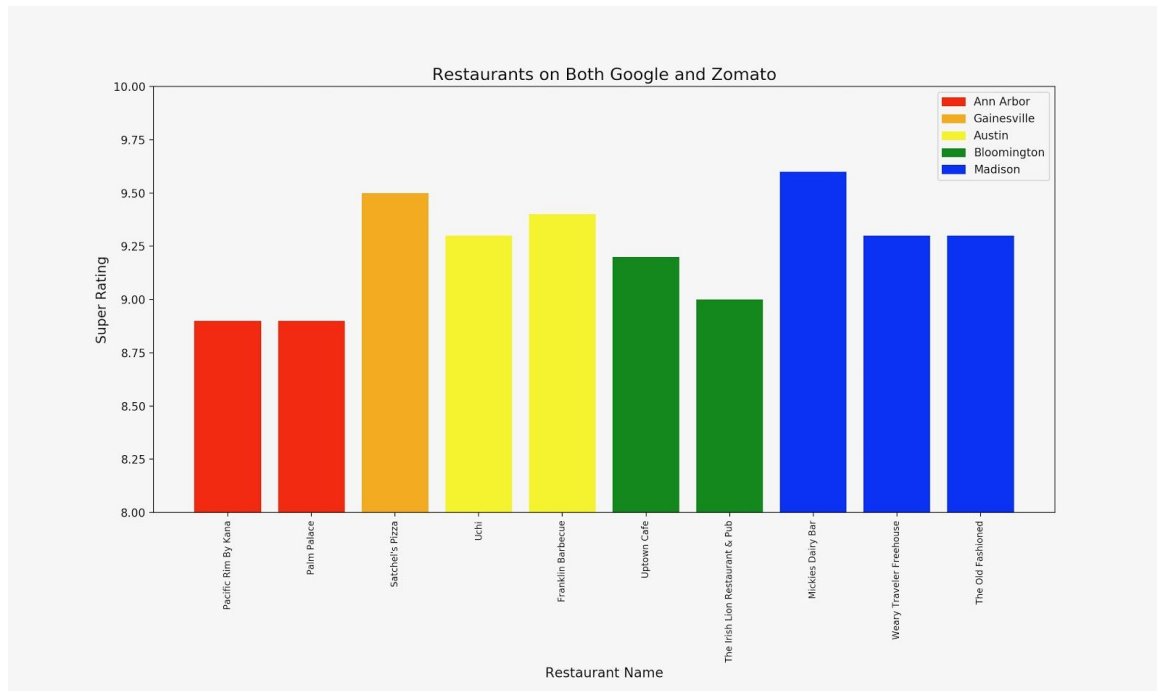
The calculated data from the Yelp+Zomato Superrating
Frita Batidos: 9.1
Mac's Drive Thru: 8.8
Uptown Cafe: 8.6
Salvatore's Tomato Pies: 9.0

The calculated data from the Google+Yelp Superrating
The Lunch Room: 9.2
Daily Green: 9.2
Metro Diner: 9.1
Beque Holic: 9.0
The Owlery Restaurant: 8.7
Sofra Cafe: 9.2
Uptown Cafe: 8.6
Turkuaz Cafe: 9.1
Samira Restaurant: 9.0
Nook: 9.0
Double 10 Mini Hot Pot: 9.0
```

Visualizations:

We created three visualizations based off of the crossover of restaurants between the three APIs. We used different colors to represent the different cities that the restaurants are found in and we created a legend to mark which color identifies each city. The x axis contains the names of the restaurants and the y axis shows the “super rating” that we calculated out of 10.





Also, we added a fourth visualization to show the two restaurants that were on all three API's. The x axis contains the names of the two restaurants and the y axis shows the "max rating" that we calculated out of 15.

Instructions for Running Code:

1. Run **read-data.py**:

- a. This will run the ZomatoAPI to retrieve information and put it into the database, final-database.db. Running this will prompt the user for user-input, (each user will have to start with the city, 'Ann Arbor,' in order to create the table, then we used 'Gainesville', 'Austin', 'Bloomington', and 'Madison') which would be the city they want to add to the database. Running this will populate the database with 100 items, 20 per city.

2. Run **yelp-read.py**:

- a. This will run the YelpAPI to retrieve information and put it into the database, final-database.db. Running this will prompt the user for user-input, (each user will have to start with the city, 'Ann Arbor,' in order to create the table) which would be the city they want to add to the database. Running this will populate the database with 100 items, 20 per city.

3. Run **google-read.py**:

- a. This will run the Google Places API to retrieve information and put it into the database, final-database.db. Running this will prompt the user for user-input, (each user will have to start with the city, 'Ann Arbor,' in order to create the table) which would be the city they want to add to the database. Running this will populate the database with 100 items, 20 per city.

4. Run **calculations.py**:

- a. This file will select data from each table in the database for each API. Then, using two comparisons at a time (Google Places v. Yelp, Yelp v. Zomato, Zomato v. Google Places), it calculates the "super-rating," and outputs the calculated data to a file.

5. Run **visualizations.py**:

- a. This will run the visualizations needed, selecting the data from the database and after it runs. It will run the matplotlib to create a bar graph, and matplotlib.patches to create a custom legend for the cities. It will create four visualizations, one for each comparison and a 'bonus' visualization that highlights the similarities between all three APIs.

Code Documentation:

read-data.py:

a. Def pull_data:

- i. Input: Zomato API key, Zomato entity ID (varies on city), entity type (geographical location). In our case, every entity type is a city.

- ii. Output: a list of tuples, each containing the name of a restaurant, its ID, its rating and its city. The tuples are sorted by each restaurant's rating, in order from highest to lowest.
- b. Def Name:**
 - i. Input: none
 - ii. Output: a list of restaurants from the tuples
- c. Def rest_id:**
 - i. Input: none
 - ii. Output: a list of restaurant IDs from the tuples
- d. Def rating:**
 - i. Input: none
 - ii. Output: a list of ratings from the tuples
- e. Def add_city:**
 - i. Input: none
 - ii. Output: a list of cities from the tuples
- f. Def convert:**
 - i. Input: none
 - ii. Output: a list of integers, 1-5, each representing a city.
- g. Def start_db:**
 - i. Input: none
 - ii. Output: created a table called "ZomatoData"
- h. Def write_db:**
 - i. Input: rest_data, which is the name of a restaurant, rest_id, the restaurant ID, rest_rate, the rating of the restaurant, and city_id, the city ID.
 - ii. Output: Inserted the data inputted into the "ZomatoData" table

yelp-read.py:

- a. Def pull_data:**
 - i. Input: Yelp API key and a city
 - ii. Output: a list of tuples, each containing the rating, ID, name and city of a restaurant, sorted by rating from highest to lowest.
- b. Def name:**
 - i. Input: none
 - ii. Output: a list of restaurants from the tuples
- c. Def rating:**
 - i. Input: none
 - ii. Output: a list of ratings from the tuples
- d. Def rest_id:**
 - i. Input: none

- ii. Output: a list of restaurant IDs from the tuples
- e. **Def add_city:**
 - i. Input: none
 - ii. Output: a list of cities from the tuples
- f. **Def convert:**
 - i. Input: none
 - ii. Output: a list of integers, 1-5, each representing a city.
- g. **Def start_db:**
 - i. Input: none
 - ii. Output: created two tables. One called “YelpData” and one called “CityIDConversion”.
- h. **Def write_db:**
 - i. Input: rest_data, which is the name of a restaurant, rest_id, the restaurant ID, rest_rate, the rating of the restaurant, and city_id, the city ID.
 - ii. Output: Inserted the data inputted into the “YelpData” table
- i. **Def new_table:**
 - i. Input: city ID and city
 - ii. Output: Inserted the data inputted into the “CityIDConversion” table

google-read.py:

- a. **Def pull_data:**
 - i. Input: Google API key and search, the input for the query of each city (for example the search query for Ann Arbor would be the string: “restaurants+in+Ann+Arbor”)
 - ii. Output: a list of tuples, each containing the rating, ID and name of a restaurant, sorted by rating from highest to lowest.
- b. **Def name:**
 - i. Input: none
 - ii. Output: a list of restaurants from the tuples
- c. **Def rest_id:**
 - i. Input: none
 - ii. Output: a list of restaurant IDs from the tuples
- d. **Def rating:**
 - i. Input: none
 - ii. Output: a list of ratings from the tuples
- e. **Def convert:**
 - i. Input: none
 - ii. Output: a list of integers, 1-5, each representing a city
- f. **Def start_db:**

- i. Input: none
- ii. Output: created table called “GoogleData”

g. Def write_db:

- i. Input: rest_data, which is the name of a restaurant, rest_id, the restaurant ID, rest_rate, the rating of the restaurant, and city_id, the city ID.
- ii. Output: Inserted the data inputted into the “GoogleData” table

calculations.py:

a. Def join_yelpgoogle:

- i. This function takes no input
- ii. This function uses ‘SELECT’ and ‘JOIN’ to add a new table to final-database.db, which calculations of the ‘super-rate,’ for Yelp and Google

b. Def join_yelpzomato:

- i. This function takes no input
- ii. This function uses ‘SELECT’ and ‘JOIN’ to add a new table to final-database.db, which calculations of the ‘super-rate,’ for Yelp and Zomato

c. Def join_zomatogoogle:

- i. This function takes no input
- ii. This function uses ‘SELECT’ and ‘JOIN’ to add a new table to final-database.db, which calculations of the ‘super-rate,’ for Google and Zomato

visualizations.py:

a. Def visualization1:

- i. The function inputs three parameters: tablename, title, savename
 - 1. Tablename = the name of the table as it is stored in the database, so the function knows which table to retrieve the data from
 - 2. Title = this input is the title of the visualization
 - 3. Savename = the name we want to save the visualization under
- ii. The function then creates the visualization and gives it the title we assign, saves it under the name we assign, and pulls the data from the table we choose as input. We colored the bar chart based off of the city and included a legend, which displays what color corresponds to which of the five cities (red = Ann Arbor, orange = Gainesville, yellow = Austin, green = Bloomington, blue = Madison).
- iii. The output of the function is the visualization
 - 1. We ran this function 3 times (1 for each of the cross-listed API’s) and each time inputted a different tablename, title, and savename for 3 total visualizations that show the restaurant names on the horizontal axis of the bar graph, and super rating on the vertical axis (score out of 5 from one API + score out of 5 from the second API that it is cross-listed on)

b. Def visualization2:

- i. This function does not input any parameters
 1. It pulls data from the YelpZomato table in the final-database.db and the GoogleData table, so we were able to add the 'max rating' for the two restaurants that were on all 3 API's.
 2. We color the bar graph by city and have max rating on the y axis and restaurant name on the x axis
- ii. The output of this function is the visualization
 1. We save it under "All3.png" and the visualization will appear when you run the function

Documents Used:

Date:	Issue Description:	Location of Resource:	Results:
4/7/20	Decided on which APIs we wanted to use - ZomatoAPI,YelpAPI, GooglePlaces API	API Websites/ API Documentation - Zomato, Yep, and Google Places	We were able to extract data from each API, and put the data into the database
4/21/20	Needed help with the visualizations, used the Lecture slides and the matplotlib website	Matplotlib-v3.pptx matplotlib.org	We were able to gain knowledge on how to create visualizations,and how to make a proper 'legend' so that our graphs were readable.
4/10/20 - 4/25/20	Various issues: how to access items in a table, how to make calls to an API, how to make sure there isn't duplicate data in a database, how to make a shared key	Piazza StackOverFlow Github W3Schools	Resources allowed us to solve existing problems and help move us forward with our project.