Team: Jenya Patel and Tarush Nandrajog
Github Repository: https://github.com/jenya-patel/SI-206-Final-Project

Original Goals:
Our original goal was to utilize information from the Yelp API and the Census Bureau website. By using the Yelp API, we aimed to collect data about schools in Michigan regarding ratings, total ratings, number of schools in each state, and zip code. Through the Census Bureau, we wanted to collect data about median income levels from cities in Michigan with the schools we had chosen in the Yelp API, and join the data based on the Zip Code. Ultimately, we wanted to compare how income level impacted access to education and quality of education by understanding the relationship between income and the ratings we collected from Yelp. Our other main goals were to apply our learning in understanding how to access an API, extract relevant data, scrap a website with Beautiful Soup, and compile all collected into databases that we could then generate various visualizations from.

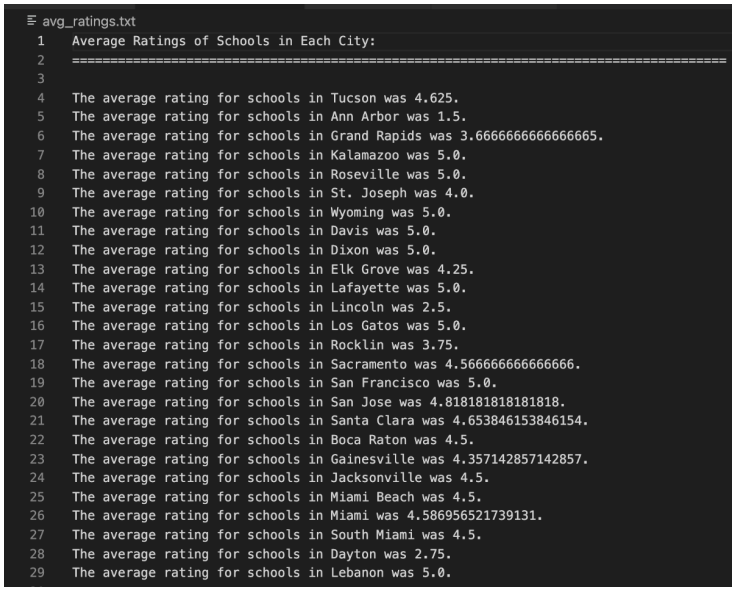
Achieved Goals:
Early into our data collection process, we realized that the Yelp data was limited for Michigan schools data and that the zip code was not available on the Census Bureau site. As a result, we shifted our focus to collecting data about schools in multiple states and decided to join the data on the cities instead of zip codes. We were able to collect data about schools in any city in the US from the Yelp API. Information on the schools included the number of reviews, ratings, city, zip code, and state. We were also able to scrape data from the Census Bureau regarding the median income for over 2000 cities in the US. In our database we created two tables for the Yelp API and one table for the Census Bureau data on median incomes. Our Yelp API tables were limited to input less than 25 rows at a time with over 100 rows in our main Yelp table and Census table. We processed the data collected through four unique calculations, for which we also formatted into text files. Using our calculations and collected data, we were able to complete four visualizations using Matplotlib.

Major Problems Faced:
An overarching problem we have faced is increasing the speed of our VScode browser. The Yelp API required us to make a unique API request for each region we are pulling data from which meant we had 5-6 API calls for each run of our code leading to a very slow debugging process. The solution we found to this was creating a list of cities that were observed in both the Yelp and Censes data and inputting the data from each of these regions one by one to the database as opposed to doing it all at once. This reduced the amount of repeated code we had to write while also increasing the speed of our browser. A separate issue we faced was in collecting income data to join with our Yelp findings. Our original plan was to extract median income data based on zip codes which could be easily compared to the zip code data found using the Yelp API. However, we have found that income data on the zip code level is very difficult to find and so are

instead joining the databases on the city-level income and review data. The problem with this is that the Government Census website and Yelp format their data differently and so the city names are not the same across both platforms. To solve this we had to clean our census data by splitting and replacing certain words/characters to ensure the data is consistent and can be joined further down the line.

Calculations:

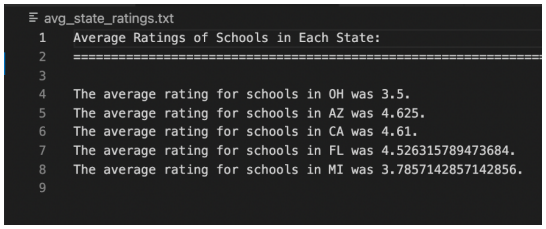
The Average Ratings of Schools in Each City



avg_ratings.txt

```
Average Ratings of Schools in Each City:
================================================================================

The average rating for schools in Tucson was 4.625.
The average rating for schools in Ann Arbor was 1.5.
The average rating for schools in Grand Rapids was 3.6666666666666665.
The average rating for schools in Kalamazoo was 5.0.
The average rating for schools in Roseville was 5.0.
The average rating for schools in St. Joseph was 4.0.
The average rating for schools in Wyoming was 5.0.
The average rating for schools in Davis was 5.0.
The average rating for schools in Dixon was 5.0.
The average rating for schools in Elk Grove was 4.25.
The average rating for schools in Lafayette was 5.0.
The average rating for schools in Lincoln was 2.5.
The average rating for schools in Los Gatos was 5.0.
The average rating for schools in Rocklin was 3.75.
The average rating for schools in Sacramento was 4.566666666666666.
The average rating for schools in San Francisco was 5.0.
The average rating for schools in San Jose was 4.818181818181818.
The average rating for schools in Santa Clara was 4.653846153846154.
The average rating for schools in Boca Raton was 4.5.
The average rating for schools in Gainesville was 4.357142857142857.
The average rating for schools in Jacksonville was 4.5.
The average rating for schools in Miami Beach was 4.5.
The average rating for schools in Miami was 4.586956521739131.
The average rating for schools in South Miami was 4.5.
The average rating for schools in Dayton was 2.75.
The average rating for schools in Lebanon was 5.0.
```

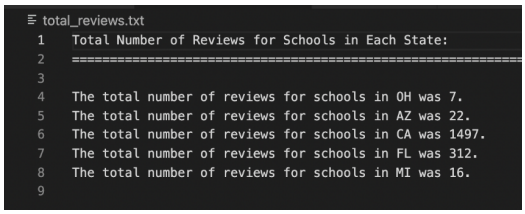
The Average Ratings of Schools in Each State



avg_state_ratings.txt

```
Average Ratings of Schools in Each State:
============================================================

The average rating for schools in OH was 3.5.
The average rating for schools in AZ was 4.625.
The average rating for schools in CA was 4.61.
The average rating for schools in FL was 4.526315789473684.
The average rating for schools in MI was 3.7857142857142856.

```

The Total Number of Educational-Related Reveiws for Each State



total_reviews.txt

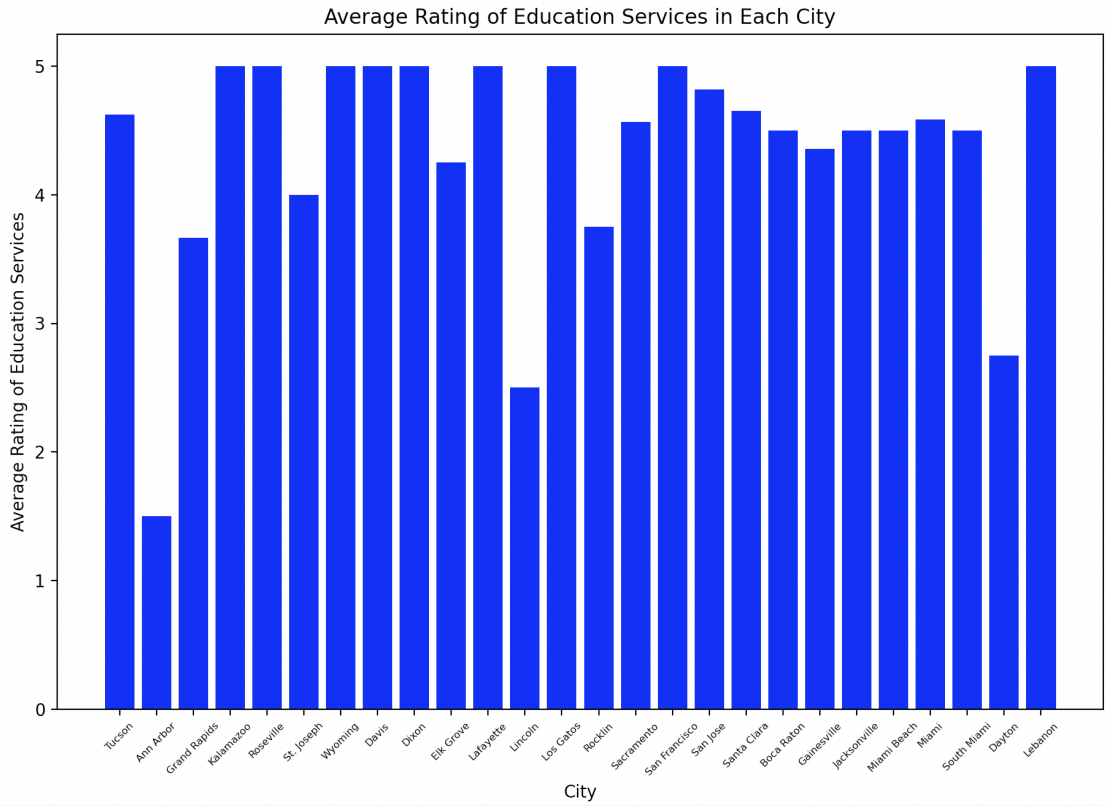
```
Total Number of Reviews for Schools in Each State:
============================================================

The total number of reviews for schools in OH was 7.
The total number of reviews for schools in AZ was 22.
The total number of reviews for schools in CA was 1497.
The total number of reviews for schools in FL was 312.
The total number of reviews for schools in MI was 16.

```

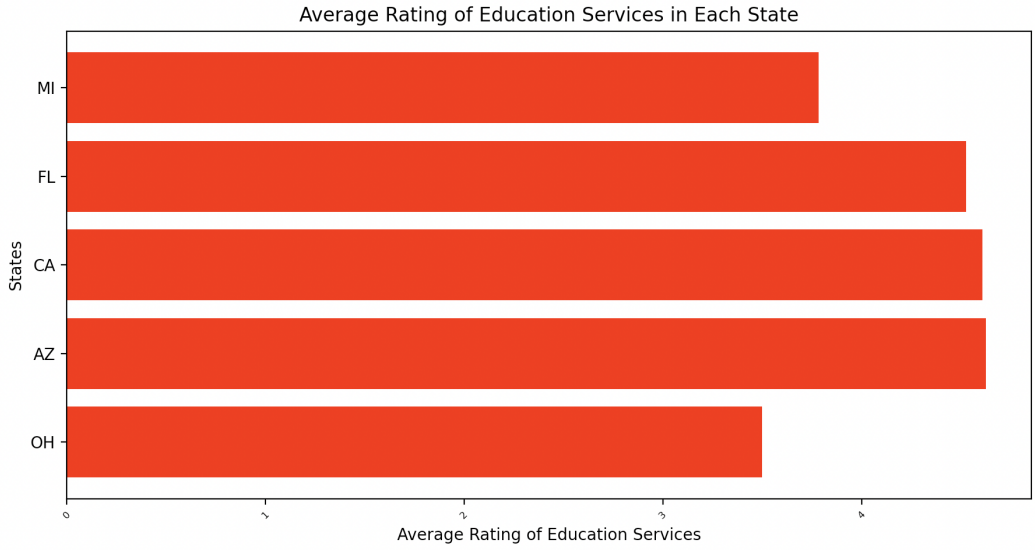
The Average Ratings of Schools Compared to the Median Income of the City the School is in

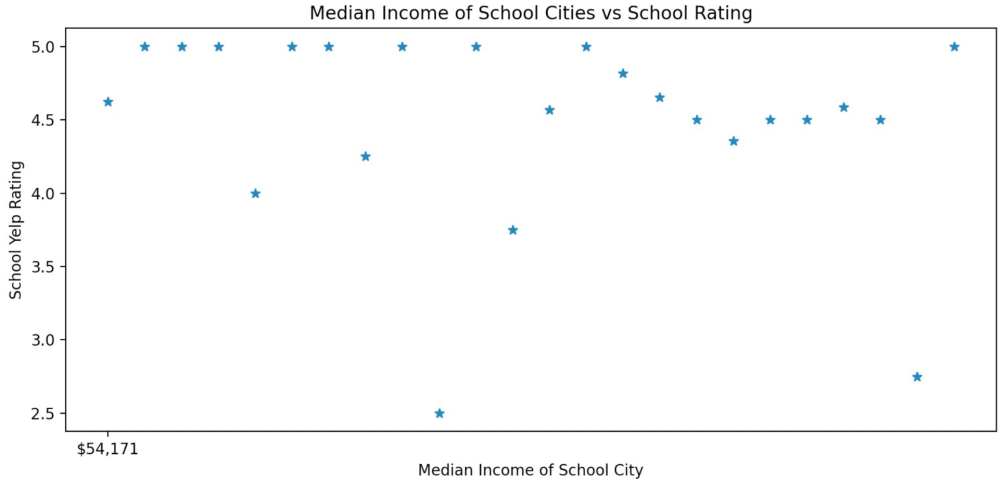
income_vs_ratins.txt

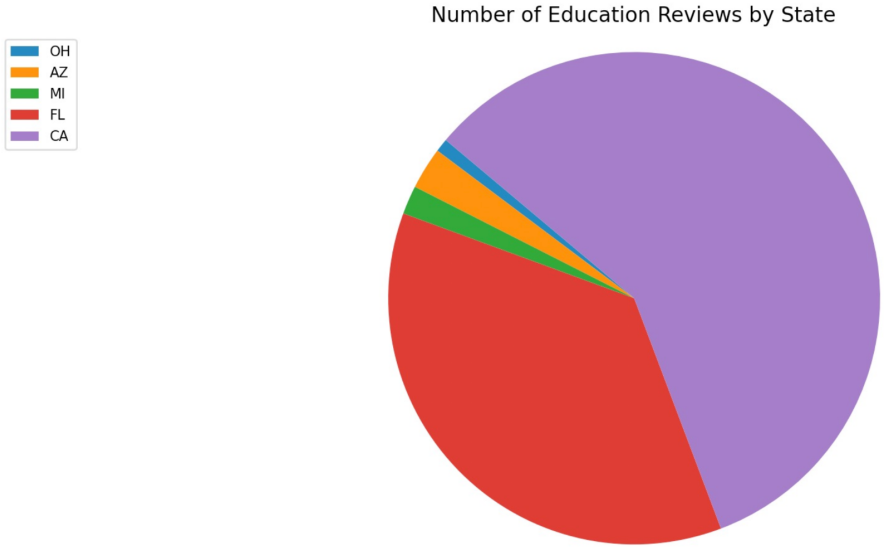
```
Average Ratings of Schools vs Median Income of the City:
================================================================================

The average rating for schools in Tucson with a median income of $54,171 was 4.625.
The average rating for schools in Ann Arbor with a median income of $35,670 was 1.5.
The average rating for schools in Grand Rapids with a median income of $54,270 was 3.6666666666666665.
The average rating for schools in Kalamazoo with a median income of $37,192 was 5.0.
The average rating for schools in Roseville with a median income of $86,694 was 5.0.
The average rating for schools in St. Joseph with a median income of $62,935 was 4.0.
The average rating for schools in Wyoming with a median income of $67,067 was 5.0.
The average rating for schools in Davis with a median income of $99,910 was 5.0.
The average rating for schools in Dixon with a median income of $56,899 was 5.0.
The average rating for schools in Elk Grove with a median income of $115,846 was 4.25.
The average rating for schools in Lafayette with a median income of $93,473 was 5.0.
The average rating for schools in Lincoln with a median income of $68,658 was 2.5.
The average rating for schools in Los Gatos with a median income of $250,000+ was 5.0.
The average rating for schools in Rocklin with a median income of $89,110 was 3.75.
The average rating for schools in Sacramento with a median income of $138,641 was 4.566666666666666.
The average rating for schools in San Francisco with a median income of $123,529 was 5.0.
The average rating for schools in San Jose with a median income of $119,136 was 4.818181818181818.
The average rating for schools in Santa Clara with a median income of $52,349 was 4.653846153846154.
The average rating for schools in Boca Raton with a median income of $54,922 was 4.5.
The average rating for schools in Gainesville with a median income of $58,566 was 4.357142857142857.
The average rating for schools in Jacksonville with a median income of $55,197 was 4.5.
The average rating for schools in Miami Beach with a median income of $60,804 was 4.5.
The average rating for schools in Miami with a median income of $51,934 was 4.586956521739131.
The average rating for schools in South Miami with a median income of $38,656 was 4.5.
The average rating for schools in Dayton with a median income of $47,685 was 2.75.
The average rating for schools in Lebanon with a median income of $61,974 was 5.0.

```

Visualizations:




Average Rating of Education Services in Each State
MI
FL
CA
AZ
OH
States
0
1
2
3
4
Average Rating of Education Services


Median Income of School Cities vs School Rating
5.0
4.5
4.0
3.5
3.0
2.5
School Yelp Rating
$54,171
Median Income of School City


Number of Education Reviews by State
OH
AZ
MI
FL
CA

Instructions to Run Code:
To run our code it is important that you have a SQL reader and matplotlib installed on your computer so you are able to see the database tables and visualizations that we have created.

Please run the code as many times as it takes to populate the database to at least 100 entries. Different cities return a different number of datapoints and so not every city returns 25 rows of data, although that is the max that each run of the program will return.

Function Documentation:
Yelp Education Data:

a) def YelpAPI(city)
   i) Takes city as a parameter which allows the program to run through a list of cities which the Yelp API will use to scrape data from a surrounding 1000 mile radius.
   ii) Sets up the API using our unique API Key, endpoint to clarify that we are only looking for 'business search' data, and the parameters for the search, specifying the categories of the data to only be education/ school information. Additionally, each request is limited to a maximum of 25 data points.
   iii) Returns business data which is all of the relevant data to schools within the specified city parameter.

b) def setUpDataBase(db_name)
   i) Takes the database name in as an input parameter which is chosen by the user.
   ii) Template function used to set up a database in which Yelp and Census data is stored.
   iii) Returns cur and conn variables to create the database and establish SQL functionality.

c) def setUpYelpTable(business_data, cur, conn)
   i) Takes business_data which is the data collected from Yelp, cur, and conn as input parameters to create the Yelp Table within our database.
   ii) Creates the relevant headers in the Yelp to categorize the data we collected and then inputs the data by selecting and storing the information using the for loop before inserting non-repeat information using SQL.
   iii) Also creates ids for Cities and States to avoid any repeat string data by selecting it from the States and Census tables
   iv) Does not return anything, but conn.commit() function updates the database

d) def YelpStatesTable(business_data, cur, conn):
   i) Takes business_data which is the data collected from Yelp, cur, and conn as input parameters to create the States Table within our database.

ii) Stores just the ids and State names into variables through the for loop before inserting non-repeat data into the newly created States table
iii) Does not return anything but conn.commit() function updates the database

Census Bureau Data:

e) def extract_income_data(income_data)
   i) Takes in income_data as a parameter which is a variable which stores the webpage from which the income_data is being scrapped
   ii) Sets up the BeautifulSoup call to parse through a web-page and store all Median house-hold income information
   iii) Cleans the data by stripping any extraneous characters before returning them in an income list

f) def get_income_data()
   i) Does not take in any inputs but does make a call to the previous extract_income_data function
   ii) Calls the previous function for 5 different states in order to collect the Median income information for every city in each of those states
   iii) Appends all of this information to the income_list before returning the list

g) def extract_city_data(state_data)
   i) Takes in state as a parameter which is a variable which stores the webpage from which the city_data is being scrapped
   ii) Sets up the BeautifulSoup call to parse through a web-page and store all city name information. Uses different tags to the income data and also cleans it differently to only take the name of the city, omitting extra words, commas, and the name of the state.
   iii) After the data has been cleaned and appended to the city_list, the list is returned.

h) def get_city_data()
   i) Does not take in any inputs but does make a call to the previous extract_city_data function
   ii) Calls the previous function for 5 different states in order to collect the city names for every city in each of those states
   iii) Appends all of this information to the city_list before returning the list

i) def setUpCensusTable(city_list, income_list, cur, conn)

i) Takes city_list and income_list in as parameters which contains all of the information which will be used to populate the Census table in the database. Also takes cur and conn in as parameters for SQL functionality and to update the database.
ii) Creates the table with the relevant headers to categorize the data before directly inputting the city name and median income data in for each datapoint (city) collected.
iii) Does not return anything, but conn.commit() function updates the database

Calculations:

j) def calculate_avg_rating(cur, conn)
   i) Takes the cursor and database connector as inputs in order to establish SQL capabilities.
   ii) Selects the average Yelp rating and associated city id while grouping by this city_id in order to calculate the average yelp rating of all schools within each city for which Yelp data was collected.
   iii) Returns this average rating for each city

k) def write_avg_rating_to_file(filename, avg_city_rating)
   i) Takes in the filename that the calculations are being outputted to and the data that contains the calculations from the previous function
   ii) The file is opened and the data along with contextual information is added to the file.
   iii) Nothing is returned but the file is closed and a .txt file is added to the program

l) def calculate_total_reviews(cur, conn)
   i) Takes the cursor and database connector as inputs in order to establish SQL capabilities.
   ii) Selects the sum of all Yelp reviews for all of the states present in the Yelp table. To do so, the Yelp and States table must be joined and the information is grouped into the state_ids.
   iii) Returns the sum of all of the reviews for each State

m) def write_total_reviews_file(filename, total_reviews)
   i) Takes in the filename that the calculations are being outputted to and the data that contains the calculations from the previous function
   ii) The file is opened and the data along with contextual information is added to the file.

iii) Nothing is returned but the file is closed and a .txt file is added to the program

n) def calc_scatter_plot(cur, conn)
   i) Takes the cursor and database connector as inputs in order to establish SQL capabilities.
   ii) Selects the average of all Yelp ratings for all of the cities present in the Yelp table and the Median income for each of those cities along with the city name. To do so, the Yelp and Census tables must be joined and the information is grouped into by the city_id.
   iii) Returns the average rating for schools in each city and the city's median income.
o) def write_scatter_data_to_file(filename, scatter_data)
   i) Takes in the filename that the calculations are being outputted to and the data that contains the calculations from the previous function
   ii) The file is opened and the data along with contextual information is added to the file.
   iii) Nothing is returned but the file is closed and a .txt file is added to the program

p) def calculate_avg_rating_by_state(cur, conn)
   i) Takes the cursor and database connector as inputs in order to establish SQL capabilities.
   ii) Selects the average Yelp rating and associated State names while grouping by this State in order to calculate the average Yelp rating of all schools within each State for which Yelp data was collected.
   iii) Returns this average rating for each State

q) def write_avg_rating_by_state_to_file(filename, avg_city_rating)
   i) Takes in the filename that the calculations are being outputted to and the data that contains the calculations from the previous function
   ii) The file is opened and the data along with contextual information is added to the file.
   iii) Nothing is returned but the file is closed and a .txt file is added to the program

<u>Visualizations:</u>

a) pie_chart_totals(cur, conn)
   i) Takes in the curson and connector as the only inputs in order to access sql and database functionality
   ii) Creates a piechart for the total number of reviews collected by our Yelp API for each state. Legend is created to outline all of the labels in a clear manner.
   iii) Nothing is outputted but the plot is displayed using .show()

b) bar_chart_avg_rating(cur, conn)
    i) Takes in the curson and connector as the only inputs in order to access sql and database functionality
    ii) Creates a barchart for the average rating of each of education services collected by our Yelp API for each city. The title, x, and y axis labels help explain the visualization.
    iii) Nothing is outputted but the plot is displayed using .show()

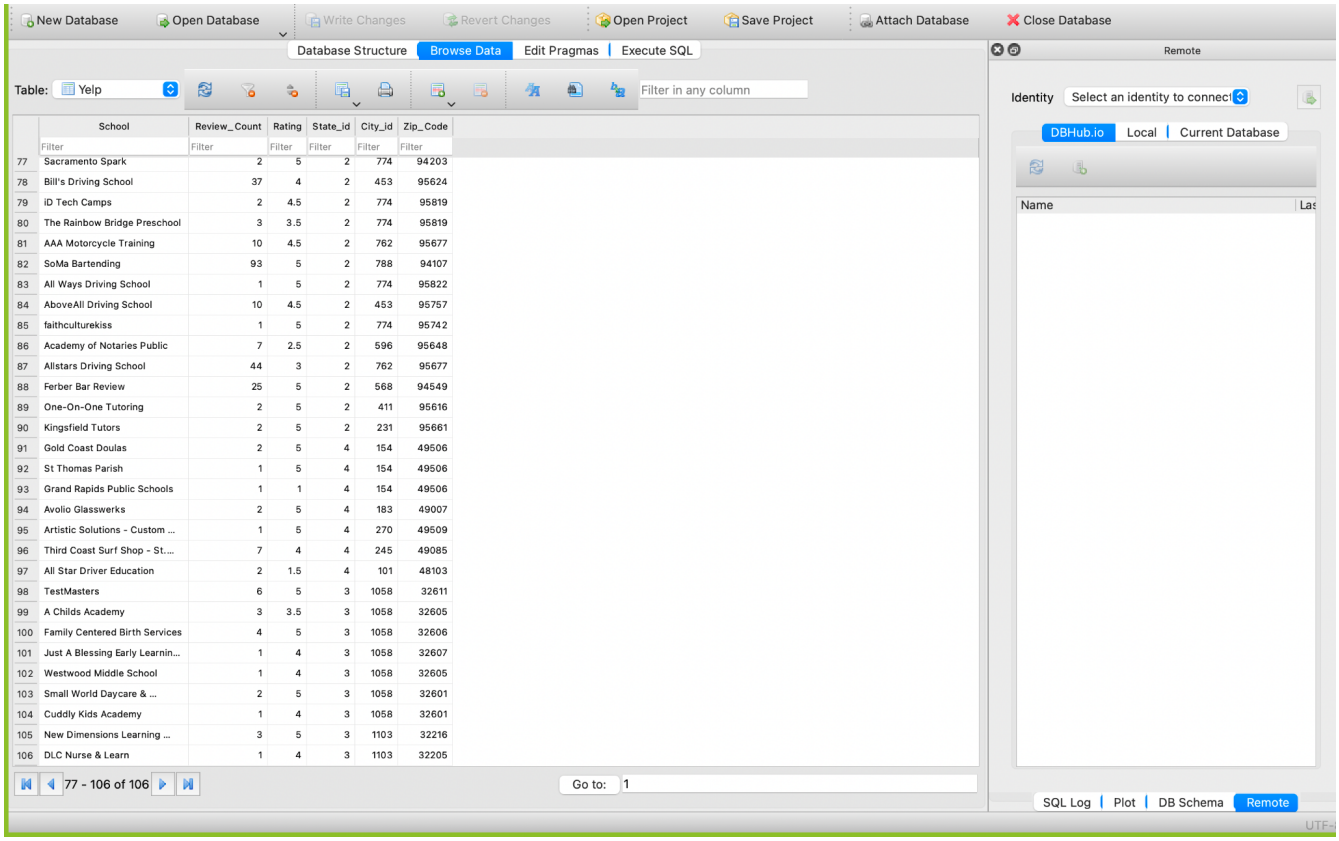
c) scatter_plot_rating_income(cur, conn)
    i) Takes in the curson and connector as the only inputs in order to access sql and database functionality
    ii) Creates a scatterplot displaying the average rating of schools within each city alongside the median income of the city the schools reside within. This helps us see if there are any correlations between income and the quality of schools.
    iii) Nothing is outputted but the plot is displayed using .show()

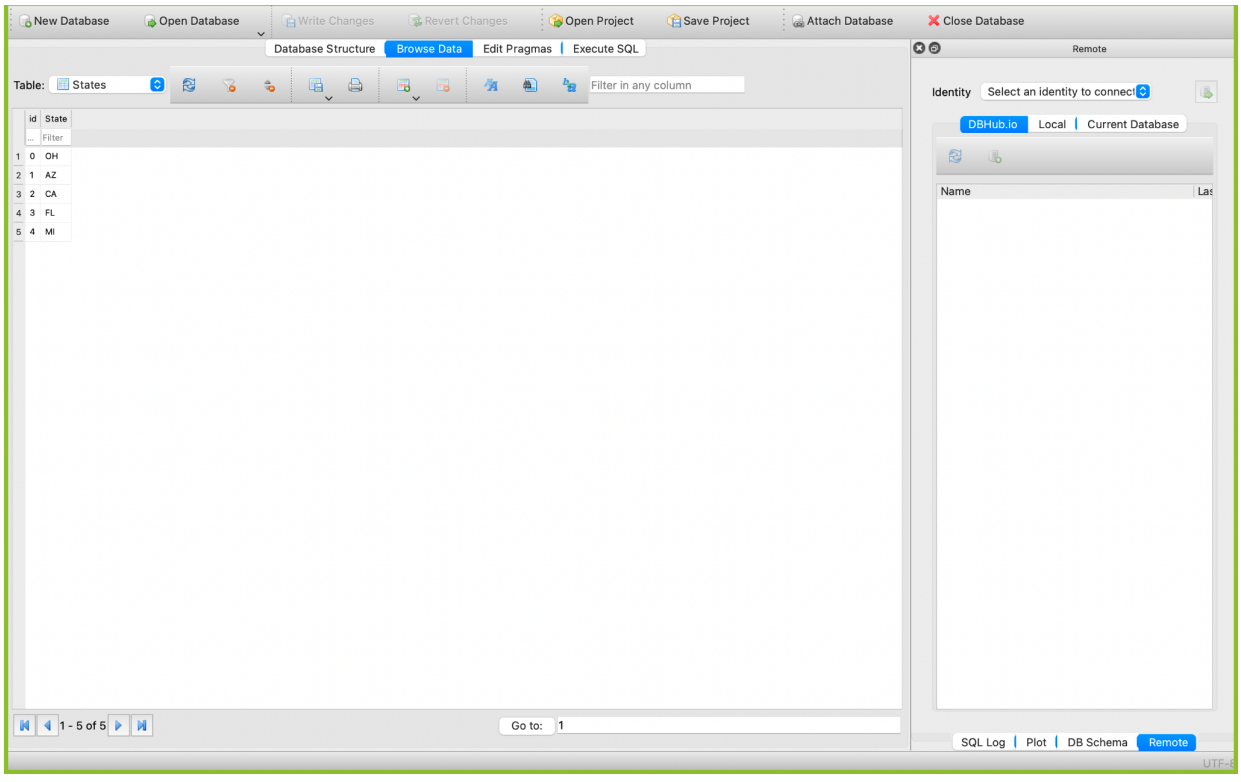
d) Horizontal_bar_chart_state_avg(cur, conn)
    i) Takes in the curson and connector as the only inputs in order to access sql and database functionality
    ii) Creates a horizontal barchart for the average rating of each of education services collected by our Yelp API for each state. The title, x, and y axis labels help explain the visualization.
    iii) Nothing is outputted but the plot is displayed using .show()

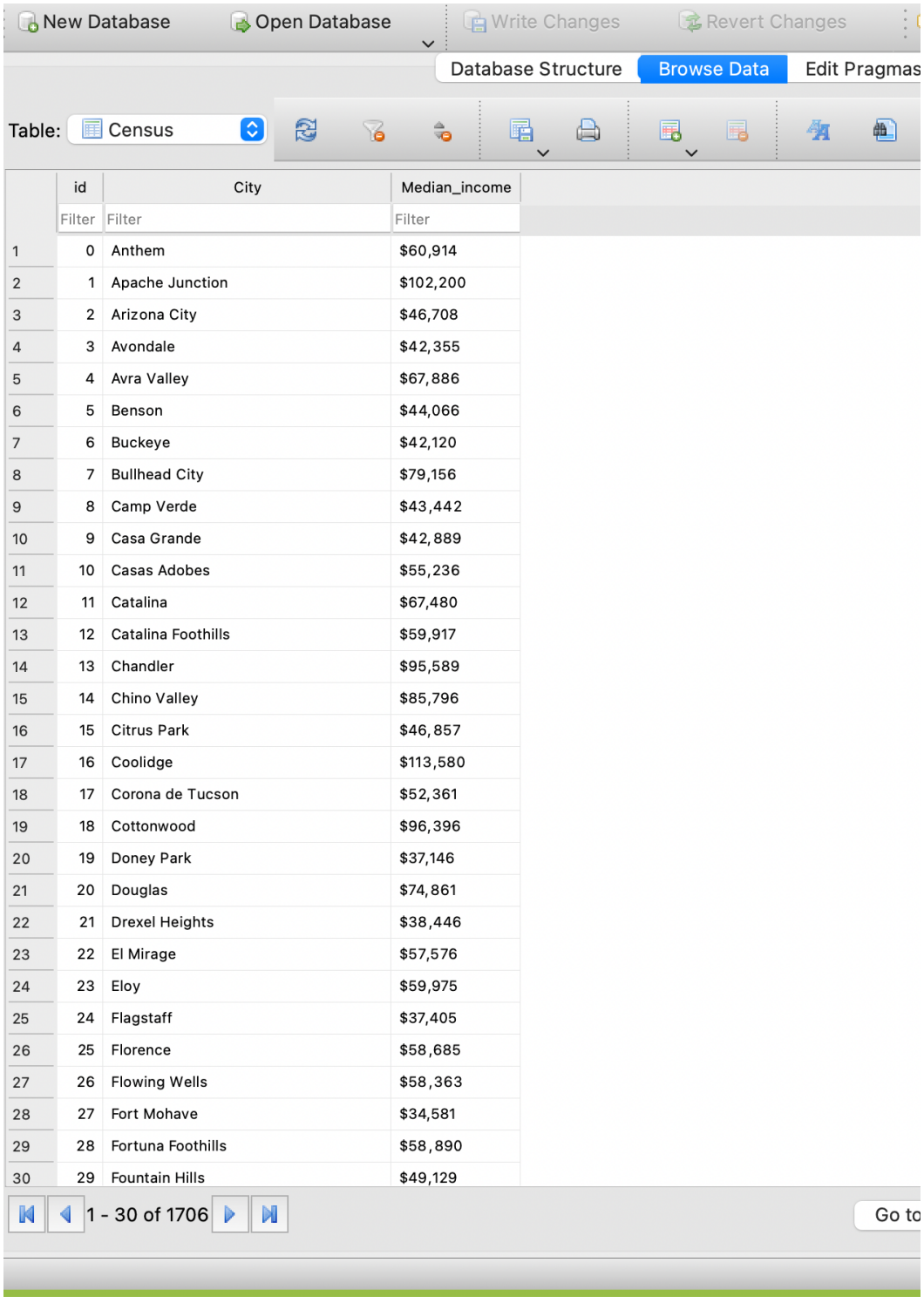
Main:

a) Cur, conn to set up the database with the given db name
b) Called get city and income data functions and store them to city and income lists which are used when setting up the Census table
c) List of cities which are present in both the Yelp and Census data are checked and if successful, the data in the regions are used to set up State ids.
d) Business_data variable is created storing the Yelp API data using the list of cities in order to set up the State and Yelp Tables
e) All of the calculation functions are called and their outputs are written into the relevant txt files
f) All of the visualization functions are also called so that they can be displayed
g) Connection is closed once all of the visualizations are outputted

Database Tables:







	id	City	Median_income
	Filter	Filter	Filter
1	0	Anthem	$60,914
2	1	Apache Junction	$102,200
3	2	Arizona City	$46,708
4	3	Avondale	$42,355
5	4	Avra Valley	$67,886
6	5	Benson	$44,066
7	6	Buckeye	$42,120
8	7	Bullhead City	$79,156
9	8	Camp Verde	$43,442
10	9	Casa Grande	$42,889
11	10	Casas Adobes	$55,236
12	11	Catalina	$67,480
13	12	Catalina Foothills	$59,917
14	13	Chandler	$95,589
15	14	Chino Valley	$85,796
16	15	Citrus Park	$46,857
17	16	Coolidge	$113,580
18	17	Corona de Tucson	$52,361
19	18	Cottonwood	$96,396
20	19	Doney Park	$37,146
21	20	Douglas	$74,861
22	21	Drexel Heights	$38,446
23	22	El Mirage	$57,576
24	23	Eloy	$59,975
25	24	Flagstaff	$37,405
26	25	Florence	$58,685
27	26	Flowing Wells	$58,363
28	27	Fort Mohave	$34,581
29	28	Fortuna Foothills	$58,890
30	29	Fountain Hills	$49,129

Resources:

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
13th April	**Needed to load Yelp API**	https://spectralops.io/blog/yelp-api-guide/#:~:text=To%20use%20Yelp%27s%20API%2C%20you,cap%20on%20daily%20API%20calls	**Yes - this helped us understand how the Yelp API worked in regards to specifying the information we wanted to pull from the site and also how to use our API Key**
15th April	**Were not able to pull the correct tags using BeautifulSoup from our web-page**	https://www.pluralsight.com/guides/web-scraping-with-beautiful-soup	**Helped - the guide provided some helpful insights into how to identify the correct tag but we ultimately needed some additional guidance in Office Hours to ensure we were selecting the appropriate tag as the website was very cluttered**
17th April	**We were struggling to understand how the AVG() and SUM() function worked**	https://www.w3schools.com/sql/sql_count_avg_sum.asp	**Yes - the guide showed us how to format the SQL and correctly select the data from the appropriate tables to ensure the calculations were done correctly**
20th April	**Ambiguous-column-name error when trying to join tables**	https://stackoverflow.com/questions/31193354/ambiguous-column-name-when-joining-tables	**Yes - we realized we had to specify the table we were pulling from to ensure SQL did not get confused when executing SELECT statements**

23rd April	**Needed to generate a unique ID to join our State and Yelp Tables on**	https://stackoverflow.com/questions/20674282/how-to-automatically-generate-unique-id-in-sql-like-uid12345678	**No - the method that was outlined in the thread did not work for us as each time we ran our code the ID would be reset**
23rd April	**Tried to use the NOT IN command to set up unique ids**	https://www.mssqltips.com/sqlservertip/6904/sql-not-in-operator	**Helped - we realized from the article that this was not the best way for us to go about setting up unique ids but were still struggling for the correct method and so reverted to setting up an Office Hours meeting for additional support which in the end allowed us to solve the issue using SELECT COUNT(*) function calls in SQL**
24th April	**Could not group our average rating data to make it categorical, allowing us to use it in our bar graph visualization**	https://stackoverflow.com/questions/5662545/how-do-i-group-on-continuous-ranges	**No - this did not help us as the form did not help us properly cast our NUMERIC data to TEXT format but once we did figure this out, the article helped us understand the logic of grouping data.**
24th April	**Needed to use a Round() function all to group our data**	**https://www.w3resource.com/sql/arithmetic-functions/floor-wth-positive-value.php**	**Yes - this article helped us identify CEILING() as the correct function to call which allowed us to successfully group our data to integer values before**

			casting those to be strings
26th April	**Did not understand what the subplot function did in matplotlib after seeing it in the code for other visualizations**	https://stackoverflow.com/questions/3584805/in-matplotlib-what-does-the-argument-mean-in-fig-add-subplot111	**Yes - we came to understand that the subplot function sets up a grid and specifies how much of the pop-up window the chart populates. Setting it to 111 meant it would be the only graph that could fit on the window.**
26th April	**The labels on our pie chart were overlapping leading to the data being difficult to read and interpret**	https://stackoverflow.com/questions/23577505/how-to-avoid-overlapping-of-labels-autopct-in-a-matplotlib-pie-chart https://www.tutorialspoint.com/how-to-avoid-overlapping-of-labels-and-autopct-in-a-matplotlib-pie-chart	**Yes - these two guides taught us how to create legends for our pie charts which separate the values from the chart itself making it far easier to read for a user.**