SI 206 Final Project Report
Jean Gamboa Fuentes (jcgamboa)
Henry Collins-Thompson (henryct)
December 16, 2024

Github link: https://github.com/HenryCT64/SI206_Project3

1. Goals for the project including APIs we planned on working with and what data we planned to gather.

For this project, our original goal was to take census data from the Census Bureau API both nationally and globally and compare it to Wikipedia's information. The types of data we intended to collect and compare included population data and potentially other demographic data.

Our original plan for APIs/websites to scrape data from was as follows:
1. https://www.wikipedia.org/
2. https://www.census.gov/
3. Potentially other countries census APIs
4. Potentially other language wikipedias (seeing access to accurate data in other languages)

However, we ultimately decided to scrap this plan, given that we found an idea we were far more interested in and it gave us experience working with two different types of data. This we found would benefit our understanding of working with APIs and databases as well as having a better database schema that fit the requirements of the project better.

2. Goals that were achieved including what APIs we actually worked with and what data we actually gathered.

Given our original plan, we still wanted to obtain information from the Census API. What we decided to do with this information was get median average income for zip codes, and compare it to the average Yelp reviews in the zipcode. We figured that this would give us a really interesting view of how wealthiness and income in a given area affects the quality of food and establishments, and also how people perceive and respond to service.

The APIs we actually worked with are as follows:
Yelp Fusion API: https://api.yelp.com/v3
Census Bureau API (ACS 5- Year Data): https://api.census.gov/data/

The data we collected from each API and stored in the database

Yelp Fusion API:

- Business name
- Zip Code
- Average Rating (1-5 scale)
- Number of Reviews
- Business Category

Census Bureau API:

- Zip Code
- Median Household Income
- Population

The information gathered from these APIs were stored each in their own table in the database. However, we created a third table for mapping a business category ID to the corresponding type of business. Then we used the business category ID in our Yelp Data table, to meet the constraints of the project and to avoid string duplication.

3. Problems that we faced during the project.

We faced many challenges throughout our project. I think that our biggest challenge was not having a clearly defined database schema and overall plan for storing our data before fetching it. This led to realizing that our initial plan would likely not meet the constraints for what the final project must include, and also would make it difficult to draw meaningful conclusions from the data itself.

Our second issue came from scraping information from the international version of wikipedia. The problem here was that the country names were in foreign languages, and there was not an easy way for us to manipulate this data such that we had direct translations from English to other languages. There was also not a one to one mapping in terms of the information found from two different regional variations of wikipedia, which made it difficult to clean that data.

Finally, our last major issue that we ran into stemmed from the Yelp API and the limits imposed on us from the free trial version. Given that the Yelp Fusion API is a paid API, the free trial version only allows for 300 API calls daily. I used a lot of these calls accidentally in testing, and this meant that by submission, I was not able to get nearly as much data as I would have liked in our database to get better results from our calculations and visualizations. While technically not

part of the grading or scope for this project, I wish we could have drawn more meaningful conclusions from our data.

4. Calculations from the data in the database.

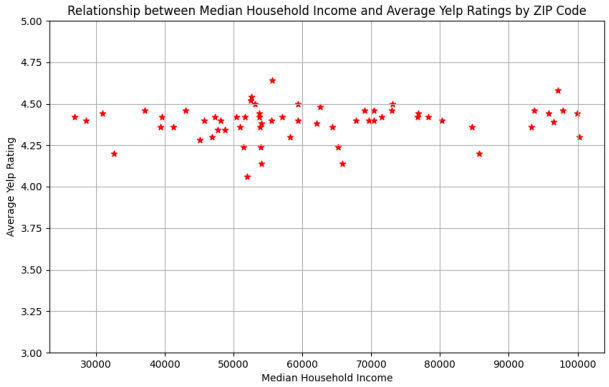
The calculations we performed from the data in the database are as follows:

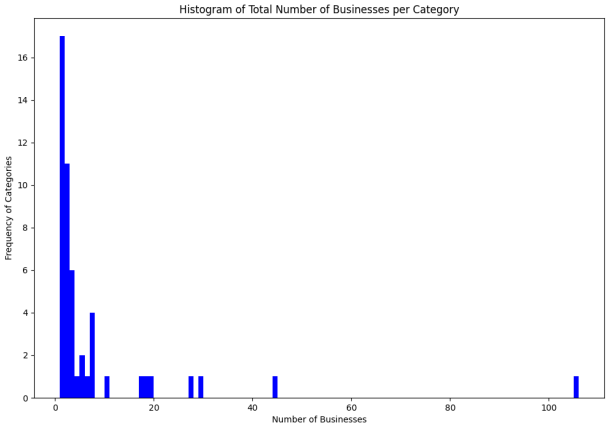
1. We took the average Yelp rating for businesses in the same zip code
2. We found the count of every type of business in the dataset and plotted it on a histogram to show the distribution of frequencies

```
def ave_rating_income_join():
    conn = sqlite3.connect(DATABASE_NAME)
    cursor = conn.cursor()
    cursor.execute("""
    SELECT yd.zip_code, AVG(yd.rating) as avg_rating, id.median_income
    FROM YelpData yd
    JOIN IncomeData id ON yd.zip_code = id.zip_code
    GROUP BY yd.zip_code
    """)
    result = cursor.fetchall()
    conn.close()
    return result

def create_scatter_plot1(joined_data):
    df = pd.DataFrame(joined_data, columns=['zip_code', 'avg_rating', 'median_income'])

    plt.figure(figsize=(10, 6))
    plt.scatter(df['median_income'], df['avg_rating'], marker = "*", color = "red")
    plt.title('Relationship between Median Household Income and Average Yelp Ratings by ZIP Code')
    plt.xlabel('Median Household Income')
    plt.ylabel('Average Yelp Rating')
    plt.ylim(3,5)
    plt.grid(True)
    plt.savefig('Income_Ratings_Scatterplot.png')
```

```python
def business_count():
    conn = sqlite3.connect(DATABASE_NAME)
    cursor = conn.cursor()

    cursor.execute("""
    SELECT COUNT(yd.id) as business_count
    FROM YelpData yd
    JOIN BusinessCategory bc ON yd.category_id = bc.id
    GROUP BY bc.category
    """)

    result = cursor.fetchall()
    conn.close()
    return [row[0] for row in result]

def create_histogram(business_data):
    plt.figure(figsize=(12, 8))
    plt.hist(business_data, bins=range(min(business_data), max(business_data) + 1), color='blue')
    plt.title('Histogram of Total Number of Businesses per Category')
    plt.xlabel('Number of Businesses')
    plt.ylabel('Frequency of Categories')
    plt.savefig('Business_Category_Histogram.png')
```

5. Visualizations





6. Instructions for running our code.

Running our code is pretty straightforward. Running the script will create the database with the three tables that we use to store our data. It will then make a call to the Census Bureau API to get 25 rows of information, with each row containing the zip code, the median income in that zip code, and the population of the zip code. From there, it will then randomly grab 5 zip codes that exist in the Census and Income data table and for each zip code make a call to the yelp API to get 5 rows each representing a business in the corresponding zip code. These rows contain the business name, the zip code of the business location, the rating on Yelp, the number of reviews, and the category (which is stored as an ID mapped from our third table).

Therefore, each time the script is run, the zip code and income table filled with census data will be populated with 25 more rows, and the Yelp data will also be populated with 25 more businesses from zip codes in our database. The business category table gets updated each time a new business type enters our database.

Calculations are then performed on the data from the database, such as the average Yelp rating by ZIP code for finding the relationship between median household income and average yelp review. Calculations are also done to create our second visualization, the count of every type of business in the dataset is found to create a histogram to show the distribution of frequencies. The results of both of those calculations are saved to own text files.

From here, the script will create and save the visualizations.

7. Documentation for each function written.

Setup Database Functions

a. setup_database()

- Purpose: Initialize the SQLite and sets up three tables: IncomeData, BusinessCategory, and YelpData, to ensure the database is always initialized and ready to store data.
- Inputs: None
- Outputs: Creates or updates the tables in project_data.db (No return value)

Census Data Functions

a. get_existing_zip_codes()

- Purpose: Fetches all zip codes currently stored in the IncomeData table to ensure no duplicate data is added.
- Inputs: None.
- Outputs: returns a set of ZIP codes that already exist in the database.

b. fetch_census_data()

- Purpose: Fetches Census Bureau data for ZIP codes, including median household income and population from API.
- Inputs: None.
- Outputs: Returns a list of tuples containing zip code, median household income, and population.

c. save_census_data_to_database()

- Purpose: Saves the fetched Census data into the IncomeData table.
- Inputs: data - a list of tuples containing ZIP codes, median household income, and population.
- Outputs: Inserts new rows into the IncomeData table (no return value).

Yelp Data Functions

a. fetch_and_store_yelp_data()

- Purpose: Fetches Yelp data for 5 random ZIP codes in IncomeData and saves it into the database.
- Inputs: None.
- Outputs: Adds up to 25 rows of Yelp data into the YelpData table of the database (no return value).

b. fetch_yelp_data()

- Purpose: Fetches Yelp Business data for a given ZIP code, limited to 5 businesses per ZIP code.
- Inputs: zip_code - The ZIP code to fetch data for.
- Outputs: Returns a list of tuples containing: Business name, ZIP code, Yelp rating, Number of reviews, Business category.

c. get_or_create_category_id()

- Purpose: Ensures a business category exists in the BusinessCategory table and fetches its corresponding ID.
- Inputs: category - A string representing the business category.
- Outputs: returns the id of the category in the BusinessCategory table.

d. save_yelp_data_to_database()

- Purpose: Saves Yelp business data into the YelpData table while ensuring proper category IDs are referenced.
- Inputs: data - a list of tuples containing: Business name, ZIP code, Yelp rating, Number of reviews, Business category.

- Outputs: Inserts data into the YelpData table (no return value).

Calculation and Visualization Functions

a. ave_rating_income_join()
- Purpose: Find the average yelp rating of restaurants pertaining to each zip code and join that with income data from that zipcode.
- Inputs: No inputs besides the already existing database.
- Outputs: returns a list of tuples that can be made into a dataframe in create_scatter_plot1().

b. create_scatter_plot1()
- Purpose: create a scatterplot png showing the relationship between average restaurant yelp rating and ZIP code income.
- Inputs: the outputs from ave_rating_income_join.
- Outputs: doesn't return anything, but creates a scatterplot png.

c. business_count()
- Purpose: to get a list of business count by frequency to make a histogram mapping the commonality of types of businesses.
- Inputs: No inputs besides the already existing database.
- Outputs: a list of the counts of business type

d. create_histogram()
- Purpose: to take data from business_count() and make a histogram to show the frequencies of business counts.
- Inputs: the list of counts from business_count().
- Outputs: a histogram png.

8. Resources Used

Date Issue	**Description**	**Location of Resource**	**Result (did it solve the issue?**
12/15/2024	This is an api url to grab data from the US census	https://api.census.gov/data/2020/acs/acs5	We decided to not use some of the data (population) and were able to get zipcode data from it to go in a new direction on our project.

12/16/2024	This is an api url to get data from Yelp	https://docs.developer.yelp.com/docs/getting-started	It was a little finicky as you needed a paid membership and had limits on data pulls, but we were able to get the information we needed.
12/16/2024	UMGPT	https://umgpt.umich.edu/	Helped figure out sql queries that we had trouble with.