Project By Micah Santow, Daniel Sun, Amy Boeving

GitHub Repository: **https://github.com/daniel-city/final_project.git**

Goals Before Project

We planned to use three APIs for this project, including Air Quality Index, Walk Score, and Traffic Flow. Our goals were to use one universal dataset with the locations in around one hundred of the largest cities in the US and Canada, and answer the question of whether the most walkable cities had the least congestion and best air quality or not. We planned to do this through first gathering the walk scores and its category names, the AQI values and their corresponding categories, and the traffic flow (speed) values, then first calculate the percent averages within each category before creating percentages that found the averages across the different data too.

Goals After Project

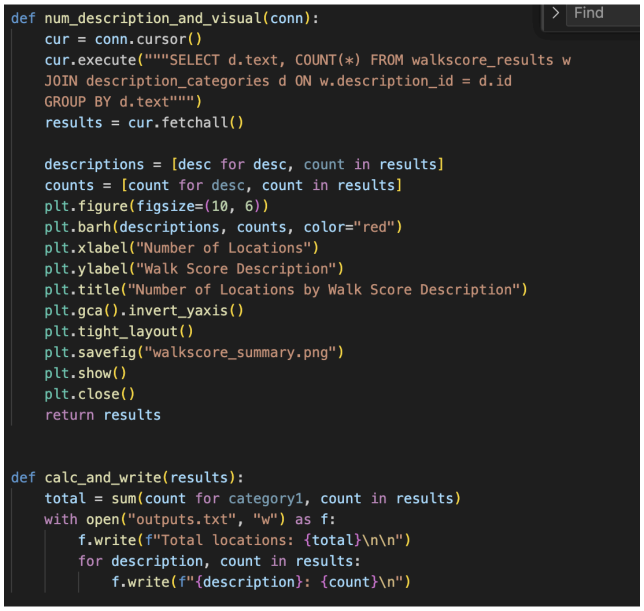
We were luckily able to accomplish a good amount of our goals, as we were able to keep the same APIs and open them with only minimal problems. We also did in fact calculate the averages when cross calculating from API to API, but for the calculations within, we instead simplified in a few of them, by just counting the proportions based on the category that each city's scores, indexes, or freeflow times/speed fell under.

Problems We Encountered

We faced a handful of problems throughout the project, including the initial struggle to open the APIs and properly load them into database tables that actually functioned correctly. Particularly for locations, that was our chosen column data type that we had chosen to be across all of our tables, but some of our tables were originally not compatible with some of the coordinates, and different variables were created from it. We were able to solve this by making our table that combined the same data from all other tables and datasets, particularly about the coordinate locations, then creating a location ID based on an integer. An additional problem we faced was getting at least one hundred lines in the tables, as some coordinates did not get past all of the code and requirements for all of the APIs, so it was adding only eighty-nine originally, until we fixed what was not working, and used an even larger set of coordinates.

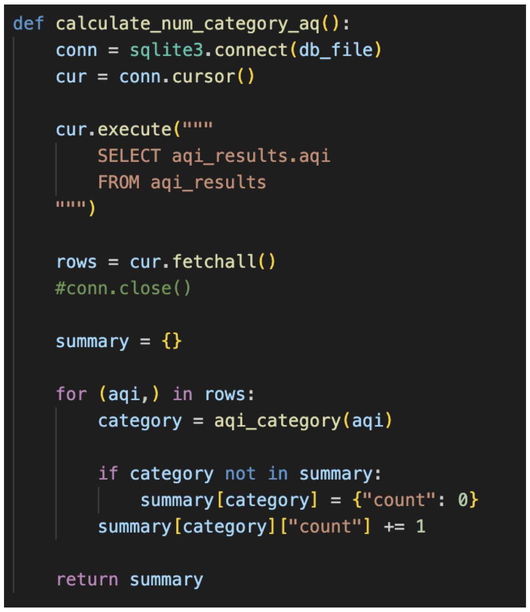
Calculations:

Walk Score:



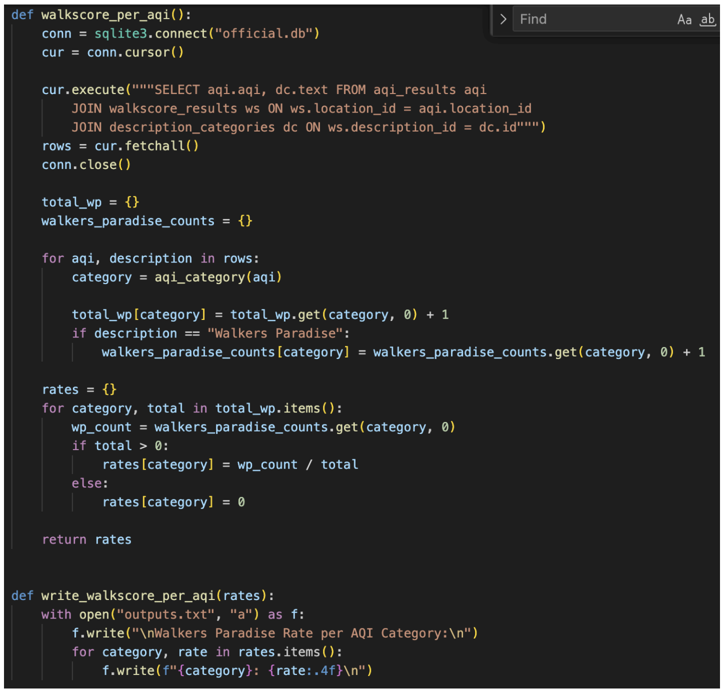
```python
def num_description_and_visual(conn):
    cur = conn.cursor()
    cur.execute("""SELECT d.text, COUNT(*) FROM walkscore_results w
    JOIN description_categories d ON w.description_id = d.id
    GROUP BY d.text""")
    results = cur.fetchall()

    descriptions = [desc for desc, count in results]
    counts = [count for desc, count in results]
    plt.figure(figsize=(10, 6))
    plt.barh(descriptions, counts, color="red")
    plt.xlabel("Number of Locations")
    plt.ylabel("Walk Score Description")
    plt.title("Number of Locations by Walk Score Description")
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.savefig("walkscore_summary.png")
    plt.show()
    plt.close()
    return results


def calc_and_write(results):
    total = sum(count for category1, count in results)
    with open("outputs.txt", "w") as f:
        f.write(f"Total locations: {total}\n\n")
        for description, count in results:
            f.write(f"{description}: {count}\n")
```

AQI:



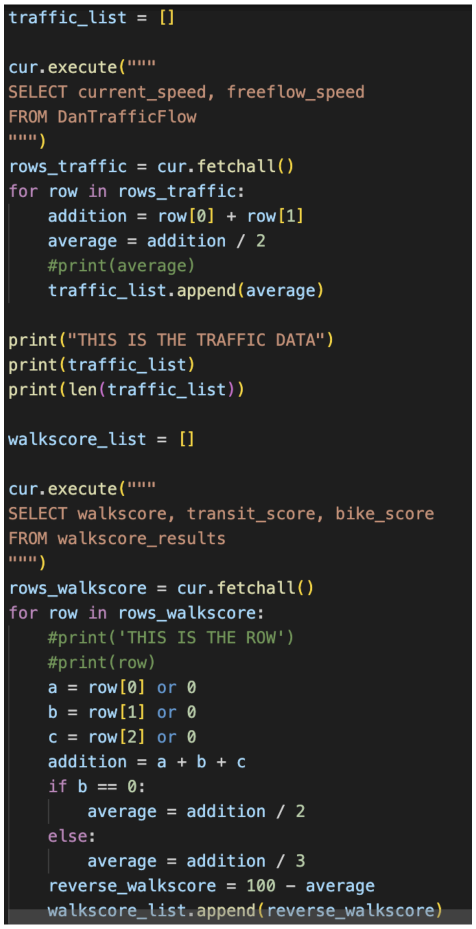
```python
def calculate_num_category_aq():
    conn = sqlite3.connect(db_file)
    cur = conn.cursor()

    cur.execute("""
        SELECT aqi_results.aqi
        FROM aqi_results
    """)

    rows = cur.fetchall()
    #conn.close()

    summary = {}

    for (aqi,) in rows:
        category = aqi_category(aqi)

        if category not in summary:
            summary[category] = {"count": 0}
        summary[category]["count"] += 1

    return summary
```

Walk Score + AQI:

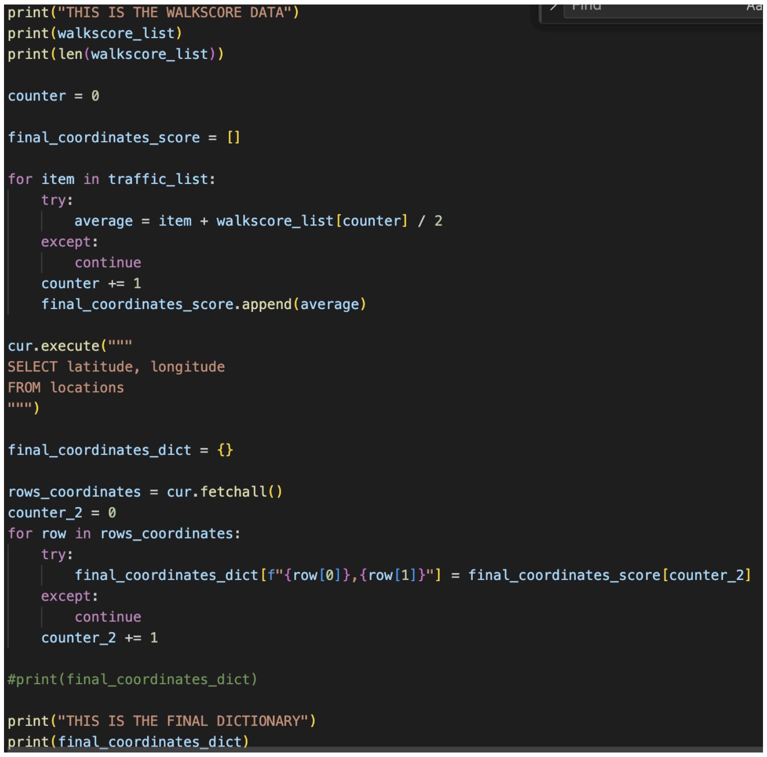


```python
def walkscore_per_aqi():
    conn = sqlite3.connect("official.db")
    cur = conn.cursor()

    cur.execute("""SELECT aqi.aqi, dc.text FROM aqi_results aqi
        JOIN walkscore_results ws ON ws.location_id = aqi.location_id
        JOIN description_categories dc ON ws.description_id = dc.id""")
    rows = cur.fetchall()
    conn.close()

    total_wp = {}
    walkers_paradise_counts = {}

    for aqi, description in rows:
        category = aqi_category(aqi)

        total_wp[category] = total_wp.get(category, 0) + 1
        if description == "Walkers Paradise":
            walkers_paradise_counts[category] = walkers_paradise_counts.get(category, 0) + 1

    rates = {}
    for category, total in total_wp.items():
        wp_count = walkers_paradise_counts.get(category, 0)
        if total > 0:
            rates[category] = wp_count / total
        else:
            rates[category] = 0

    return rates


def write_walkscore_per_aqi(rates):
    with open("outputs.txt", "a") as f:
        f.write("\nWalkers Paradise Rate per AQI Category:\n")
        for category, rate in rates.items():
            f.write(f"{category}: {rate:.4f}\n")
```

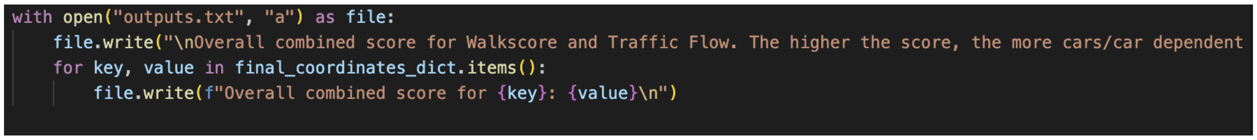
Traffic Flow:



```
traffic_list = []

cur.execute("""
SELECT current_speed, freeflow_speed
FROM DanTrafficFlow
""")
rows_traffic = cur.fetchall()
for row in rows_traffic:
    addition = row[0] + row[1]
    average = addition / 2
    #print(average)
    traffic_list.append(average)

print("THIS IS THE TRAFFIC DATA")
print(traffic_list)
print(len(traffic_list))

walkscore_list = []

cur.execute("""
SELECT walkscore, transit_score, bike_score
FROM walkscore_results
""")
rows_walkscore = cur.fetchall()
for row in rows_walkscore:
    #print('THIS IS THE ROW')
    #print(row)
    a = row[0] or 0
    b = row[1] or 0
    c = row[2] or 0
    addition = a + b + c
    if b == 0:
        average = addition / 2
    else:
        average = addition / 3
    reverse_walkscore = 100 - average
    walkscore_list.append(reverse_walkscore)
```

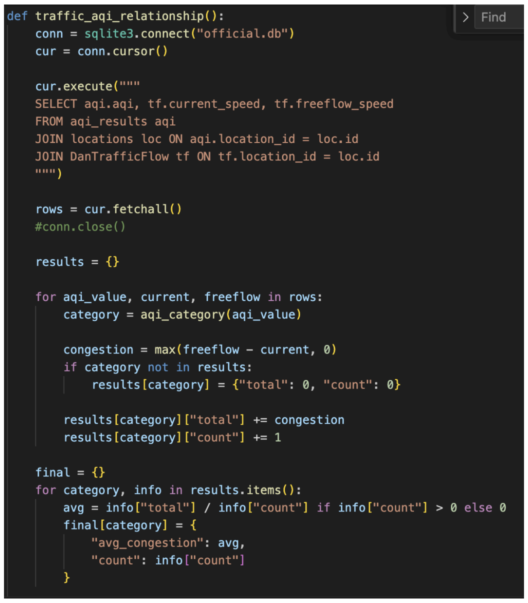


```
print("THIS IS THE WALKSCORE DATA")
print(walkscore_list)
print(len(walkscore_list))

counter = 0

final_coordinates_score = []

for item in traffic_list:
    try:
        average = item + walkscore_list[counter] / 2
    except:
        continue
    counter += 1
    final_coordinates_score.append(average)

cur.execute("""
SELECT latitude, longitude
FROM locations
""")

final_coordinates_dict = {}

rows_coordinates = cur.fetchall()
counter_2 = 0
for row in rows_coordinates:
    try:
        final_coordinates_dict[f"{row[0]},{row[1]}"] = final_coordinates_score[counter_2]
    except:
        continue
    counter_2 += 1

#print(final_coordinates_dict)

print("THIS IS THE FINAL DICTIONARY")
print(final_coordinates_dict)
```

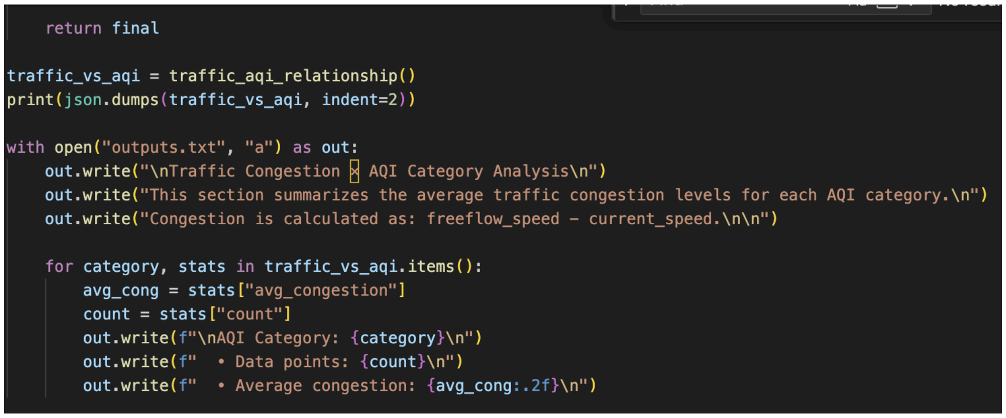


```
with open("outputs.txt", "a") as file:
    file.write("\nOverall combined score for Walkscore and Traffic Flow. The higher the score, the more cars/car dependent
    for key, value in final_coordinates_dict.items():
        file.write(f"Overall combined score for {key}: {value}\n")
```

Traffic Flow + AQI:

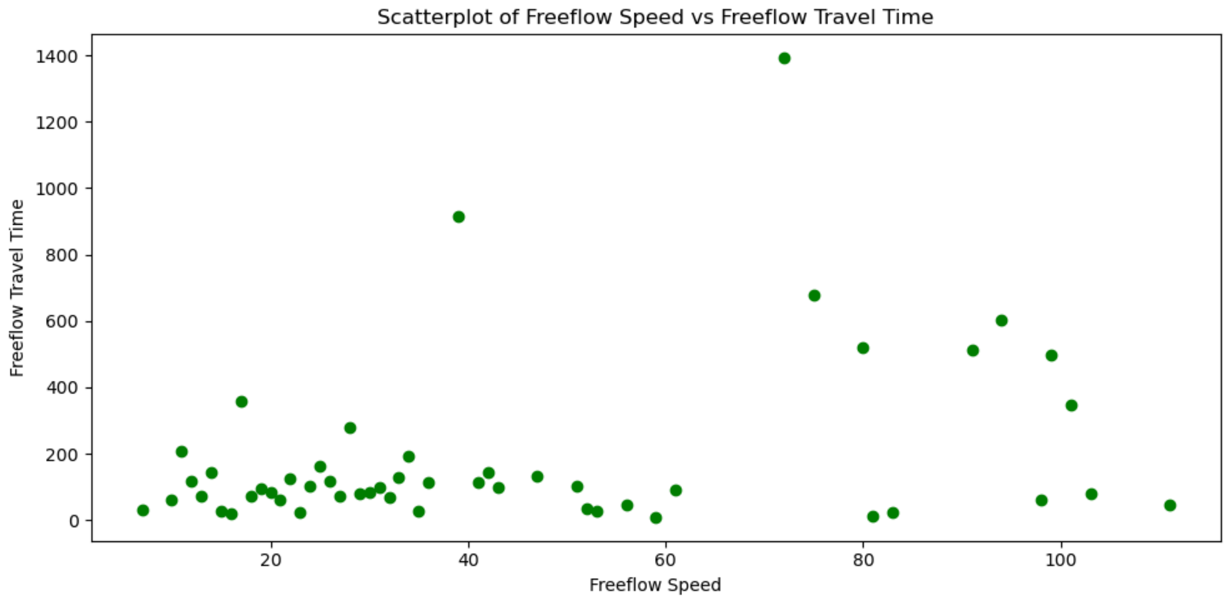


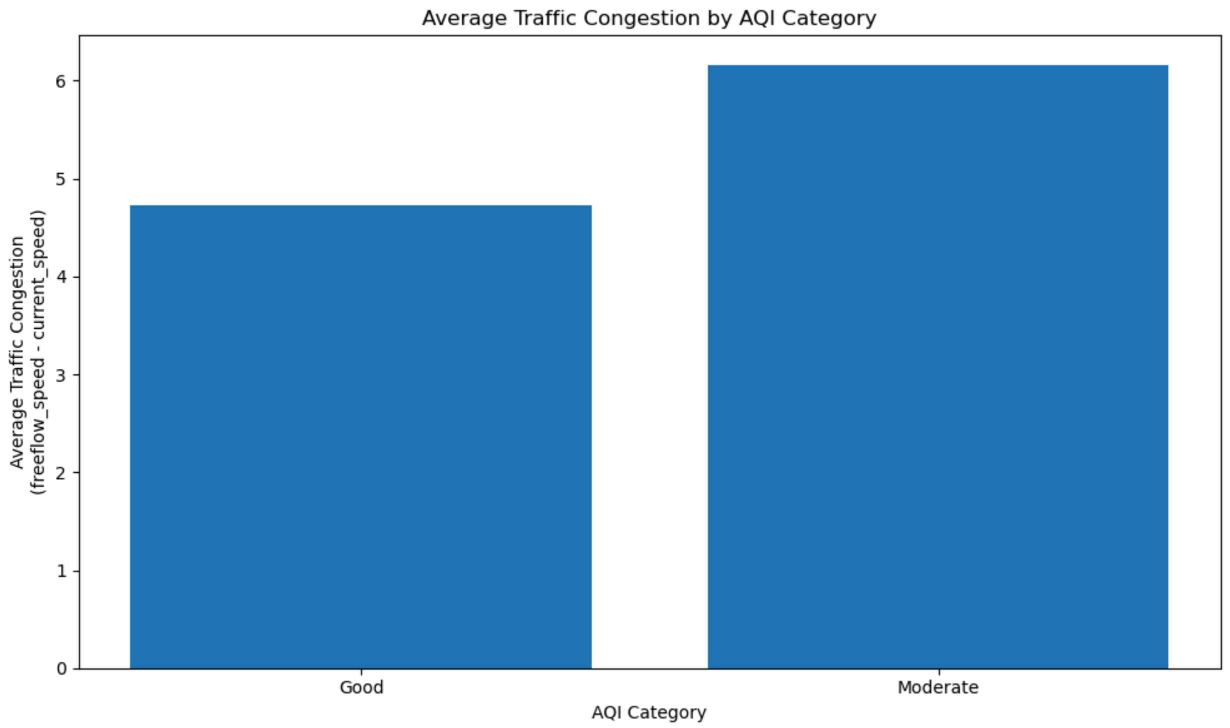
```python
def traffic_aqi_relationship():
    conn = sqlite3.connect("official.db")
    cur = conn.cursor()

    cur.execute("""
    SELECT aqi.aqi, tf.current_speed, tf.freeflow_speed
    FROM aqi_results aqi
    JOIN locations loc ON aqi.location_id = loc.id
    JOIN DanTrafficFlow tf ON tf.location_id = loc.id
    """)

    rows = cur.fetchall()
    #conn.close()

    results = {}

    for aqi_value, current, freeflow in rows:
        category = aqi_category(aqi_value)

        congestion = max(freeflow - current, 0)
        if category not in results:
            results[category] = {"total": 0, "count": 0}

        results[category]["total"] += congestion
        results[category]["count"] += 1

    final = {}
    for category, info in results.items():
        avg = info["total"] / info["count"] if info["count"] > 0 else 0
        final[category] = {
            "avg_congestion": avg,
            "count": info["count"]
        }
```

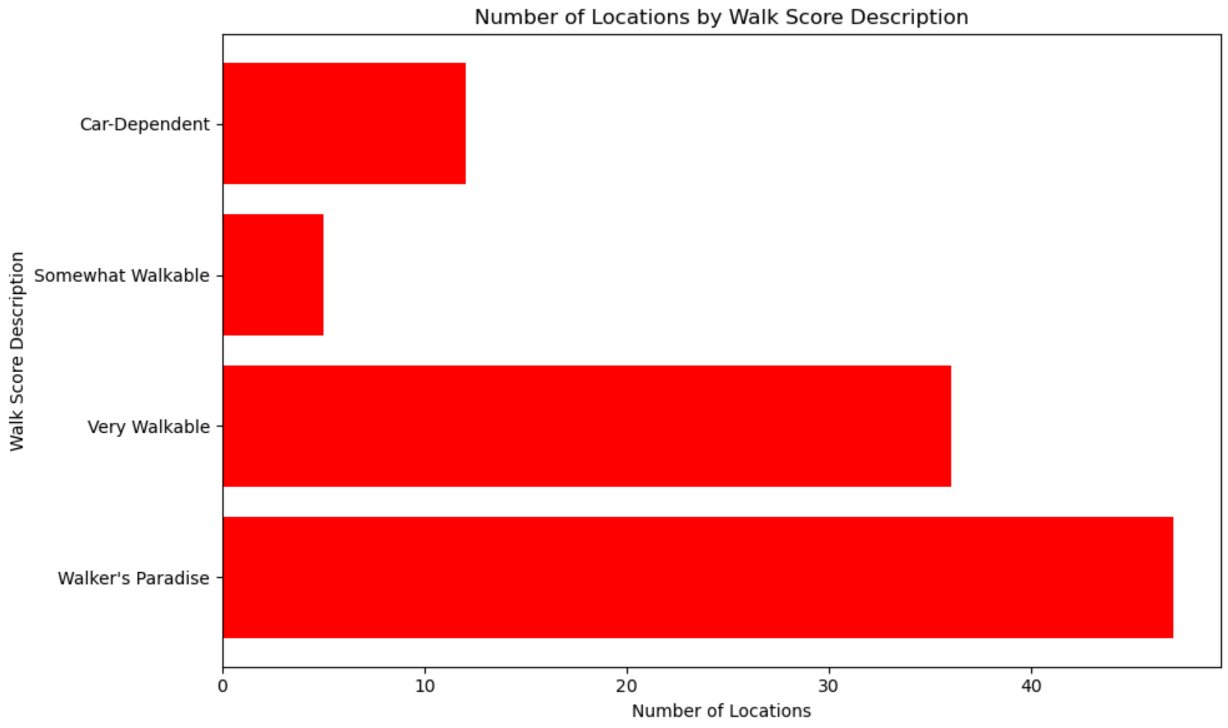


```python
    return final

traffic_vs_aqi = traffic_aqi_relationship()
print(json.dumps(traffic_vs_aqi, indent=2))

with open("outputs.txt", "a") as out:
    out.write("\nTraffic Congestion ⊠ AQI Category Analysis\n")
    out.write("This section summarizes the average traffic congestion levels for each AQI category.\n")
    out.write("Congestion is calculated as: freeflow_speed − current_speed.\n\n")

    for category, stats in traffic_vs_aqi.items():
        avg_cong = stats["avg_congestion"]
        count = stats["count"]
        out.write(f"\nAQI Category: {category}\n")
        out.write(f"  • Data points: {count}\n")
        out.write(f"  • Average congestion: {avg_cong:.2f}\n")
```

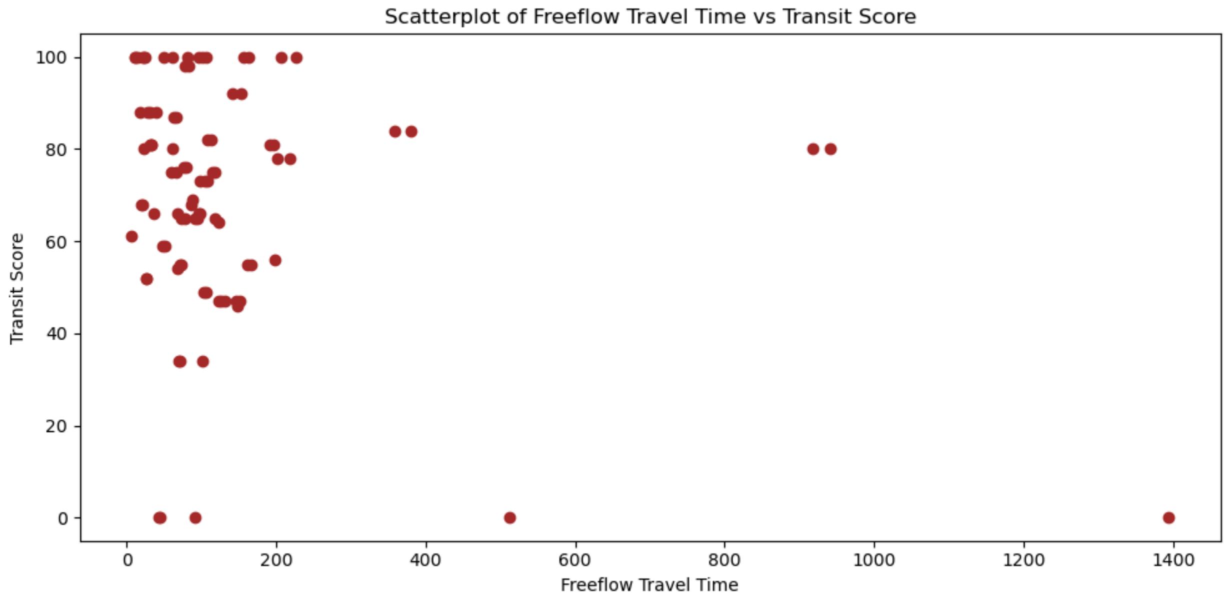
Visualizations:

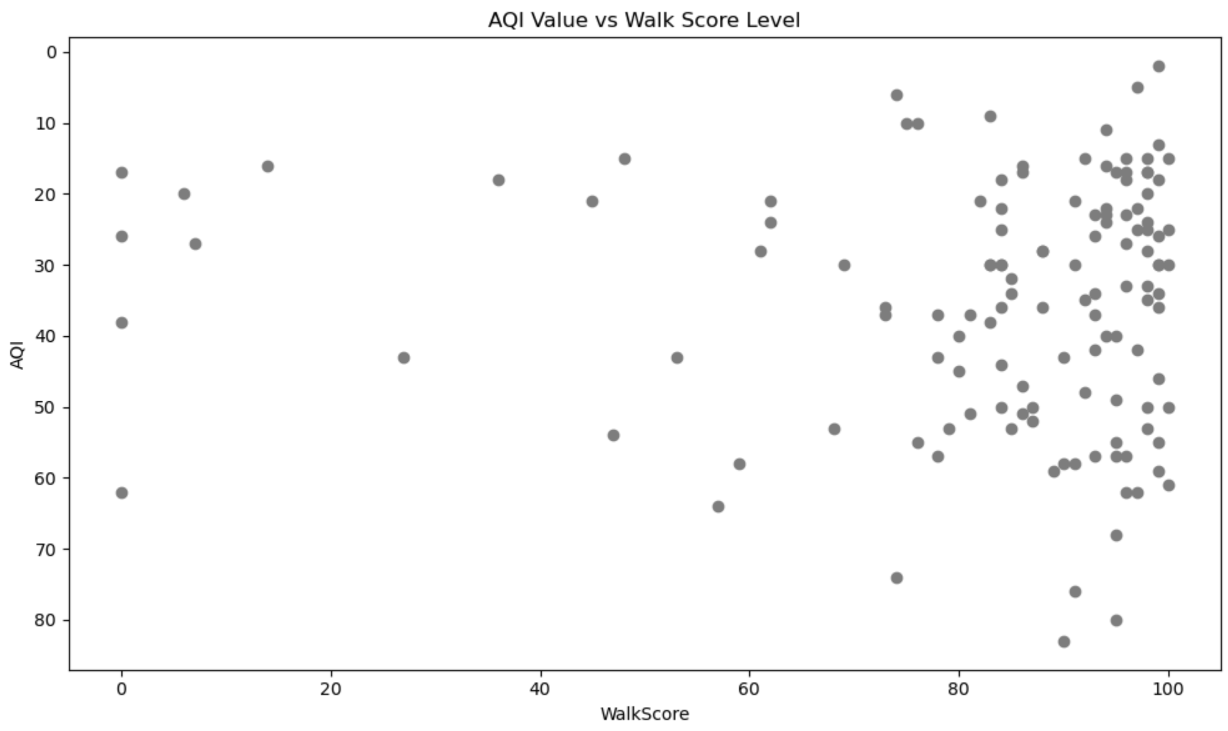

Scatterplot of Freeflow Speed vs Freeflow Travel Time
Freeflow Travel Time
0
200
400
600
800
1000
1200
1400
20
40
60
80
100
Freeflow Speed


Average Traffic Congestion by AQI Category
Average Traffic Congestion
(freeflow_speed - current_speed)
0
1
2
3
4
5
6
Good
Moderate
AQI Category



Extra Credit Visualisations





Instructions For Running Code:

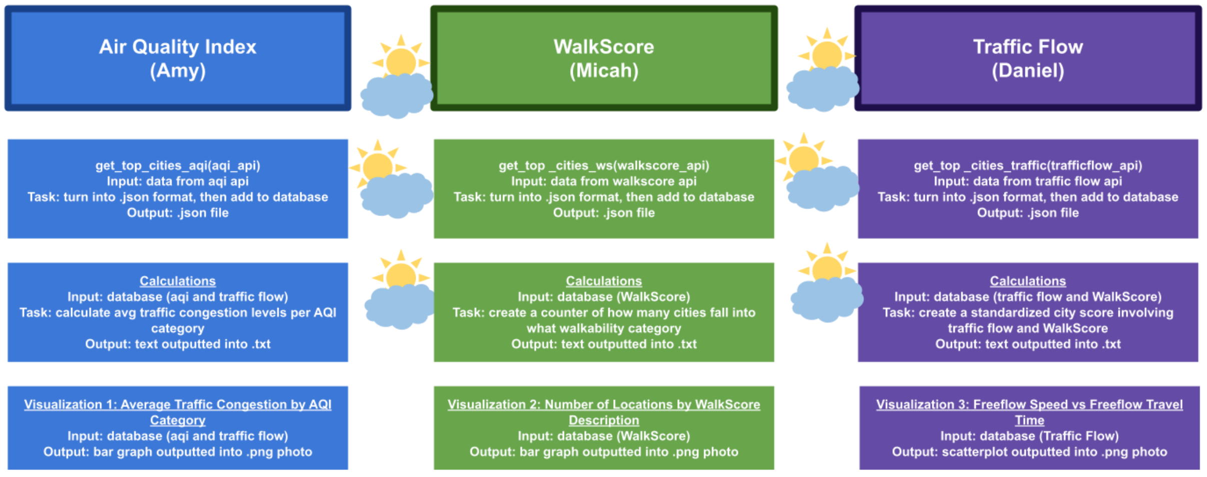
- Open ‘combinedcode.py’ file
- Hit run, it will take 2-3 minutes for everything to finish outputting/loading
- The visualizations will pop up, you can also view them updated in the 3 .png files in the folder, named accordingly to the API data used.
- To see the database, open ‘official.db’ in DB Browser. All tables are present in that specific file.
- To see all printed outputs of calculations, go to ‘outputs.txt’
- All other tabs/files were places in which we wrote code asynchronously and tested it without crashing/overriding each other's work.

Who Used It	**Date**	**Issue description**	**Location of Resource**	**Result(did it solve the issue?)**
Daniel, Micah	Dec 5	Had trouble loading in API data and putting it into a JSON	ChatGPT	Yes, we were able to successfully load in our data from our API and input it into a .json file which is also in the folder
Daniel	Dec 6	Had trouble looping through the JSON file (was looping through	ChatGPT	Yes, I was able to successfully include multiple data points (data of multiple

		the file name string, not the file itself). Also, was having issues with having more than one value in the JSON		coordinates) into my json. I was also able to loop through the JSON file as I had hoped.
Daniel	Dec 7	DB Database input is being loaded incorrectly. The column titles are right, but the data values are incorrect	ChatGPT	Yes, I was able to successfully load my data into SQL without any issues in the end.
Micah	Dec 7	I couldn't figure out why my data was not going to the database	ChatGPT	Yes, they had me place print statements throughout the code to see where the problem is and whether or not the data properly loads.
Micah	Dec 7	I needed the coordinates of 100 cities to use for locations.	ChatGPT	Yes, they gave me a list which saved time.
Daniel	Dec 7	I needed help changing the formatting of Micah's 100 city coordinates above	ChatGPT	Yes, it was able to change the formatting for each coordinate entry, which saved me time.
Amy	Dec 7	I was getting errors raised about undefined variables/missing names	ChatGPT	Yes, had mistakenly changed some variable names throughout my code + mistyped (conn, cur, init_db instead of conn, cur = init_db)
Amy	Dec 7	Code not executing and database not updating	ChatGPT	Yes, I had added new values to my database and had to delete the original file as it errored out when I would run the code with the updated database rows.
Daniel	Dec 7	Not being able to successfully do calculations because of formatting of SELECT SQL. Also, having issues with	ChatGPT	Mostly. I'm still running issues with my database table output suddenly disappearing.

		general structure of writing to a .txt file. Also, general debugging and issues with my code		
Micah	Dec 7	Dealt with numerous bugs and confusion points in SQL and matplotlib, including the number of issues with table row counts.	Chat GPT	Mostly, the visualization (bar graph) issues were resolved smoothly, but the problems with line counts persist/vary from file to file.
Amy	Dec 8	When doing cross calculation between traffic/AQI, visualization isn't showing any data	ChatGPT	No, initially I was encountering a problem where an image wouldn't print at all but that was fixed, now the issue is no data is showing up in the dantrafficflow table in test.db and therefore nothing is printed in the image. Tried to debug with ChatGPT assistance but no change.
Amy	Dec 8	When running code, the max number of rows in test.db tables is 68	ChatGPT	Yes! Our current location list has 68 unique values, and otherwise the coordinates/cities are duplicates. Need to replace with 32 unique alternate cities. Had ChatGPT generate another list with no duplicates.
Daniel	Dec 8	Had trouble loading in the visualisation and changing colors.	ChatGPT	Yes, ChatGPT was a bit helpful with helping me troubleshoot but I mainly was able to figure it out myself. ChatGPT was helpful in reminding me for how to change the colors.
Daniel	Dec 8	Text wasn't writing to .txt file	ChatGPT	Yes, it pointed out that I had defined "as file:" and then did f.write instead of file.write
Daniel	Dec 8 and	Had issues/bugs across my code	ChatGPT	Yes, ChatGPT was able to help me with fixing issues I

	prior			had across my code
Daniel	Dec 13	Had trouble implementing my insert 25 rows at a time code. It wasn't actually inputting 25 at a time properly.	ChatGPT	Yes, my code is now able to input 25 rows at a time.
Daniel	Dec 13	Changed one of my table names and had trouble removing the old table	ChatGPT	Yes, it was able to help me with that
Daniel	Dec 13	Had trouble getting data for extra credit visualisation from two different tables	ChatGPT	Yes, in the end I was able to format the cur.execute request correctly to get my visualization to work


Air Quality Index
(Amy)
WalkScore
(Micah)
Traffic Flow
(Daniel)
get_top_cities_aqi(aqi_api)
Input: data from aqi api
Task: turn into .json format, then add to database
Output: .json file
get_top _cities_ws(walkscore_api)
Input: data from walkscore api
Task: turn into .json format, then add to database
Output: .json file
get_top _cities_traffic(trafficflow_api)
Input: data from traffic flow api
Task: turn into .json format, then add to database
Output: .json file
Calculations
Input: database (aqi and traffic flow)
Task: calculate avg traffic congestion levels per AQI category
Output: text outputted into .txt
Calculations
Input: database (WalkScore)
Task: create a counter of how many cities fall into what walkability category
Output: text outputted into .txt
Calculations
Input: database (traffic flow and WalkScore)
Task: create a standardized city score involving traffic flow and WalkScore
Output: text outputted into .txt
Visualization 1: Average Traffic Congestion by AQI Category
Input: database (aqi and traffic flow)
Output: bar graph outputted into .png photo
Visualization 2: Number of Locations by WalkScore Description
Input: database (WalkScore)
Output: bar graph outputted into .png photo
Visualization 3: Freeflow Speed vs Freeflow Travel Time
Input: database (Traffic Flow)
Output: scatterplot outputted into .png photo

Copy of Text from Outputs.txt:

Total locations: 100

Car-Dependent: 12
Somewhat Walkable: 5
Very Walkable: 36
Walker's Paradise: 47

Walkers Paradise Rate per AQI Category:
Good: 0.0000

```
Moderate: 0.0000

Overall combined score for Walkscore and Traffic Flow. The higher the score, the more cars/car
dependent the location is:
Overall combined score for 40.7128,-74.006: 25.5
Overall combined score for 34.0522,-118.2437: 28.5
Overall combined score for 41.8781,-87.6298: 17.166666666666664
Overall combined score for 29.7604,-95.3698: 25.5
Overall combined score for 33.4484,-112.074: 30.833333333333336
Overall combined score for 39.9526,-75.1652: 13.0
Overall combined score for 29.4241,-98.4936: 31.166666666666664
Overall combined score for 32.7157,-117.1611: 22.833333333333336
Overall combined score for 32.7767,-96.797: 96.5
Overall combined score for 37.3382,-121.8863: 28.166666666666664
Overall combined score for 30.2672,-97.7431: 20.0
Overall combined score for 30.3322,-81.6557: 54.5
Overall combined score for 32.7555,-97.3308: 23.0
Overall combined score for 39.9612,-82.9988: 39.0
Overall combined score for 35.2271,-80.8431: 26.5
Overall combined score for 37.7749,-122.4194: 10.333333333333336
Overall combined score for 39.7684,-86.1581: 21.5
Overall combined score for 47.6062,-122.3321: 18.333333333333336
Overall combined score for 39.7392,-104.9903: 43.833333333333336
Overall combined score for 38.9072,-77.0369: 22.166666666666664
Overall combined score for 42.3601,-71.0589: 18.5
Overall combined score for 31.7619,-106.485: 55.0
Overall combined score for 36.1627,-86.7816: 18.166666666666664
Overall combined score for 42.3314,-83.0458: 24.666666666666664
Overall combined score for 35.4676,-97.5164: 37.5
Overall combined score for 45.5051,-122.675: 88.83333333333334
Overall combined score for 36.1699,-115.1398: 35.666666666666664
Overall combined score for 35.1495,-90.049: 28.833333333333336
Overall combined score for 38.2527,-85.7585: 26.666666666666664
Overall combined score for 39.2904,-76.6122: 19.833333333333336
Overall combined score for 43.0389,-87.9065: 25.5
Overall combined score for 35.0844,-106.6504: 25.166666666666664
Overall combined score for 32.2226,-110.9747: 43.166666666666664
Overall combined score for 36.7378,-119.7871: 38.0
Overall combined score for 38.5816,-121.4944: 34.833333333333336
Overall combined score for 33.4152,-111.8315: 41.833333333333336
Overall combined score for 33.749,-84.388: 39.0
Overall combined score for 41.2565,-95.9345: 28.833333333333336
Overall combined score for 38.8339,-104.8214: 34.0
Overall combined score for 35.7796,-78.6382: 28.833333333333336
Overall combined score for 36.8529,-75.978: 51.333333333333336
Overall combined score for 25.7617,-80.1918: 13.0
Overall combined score for 33.7701,-118.1937: 35.166666666666664
Overall combined score for 37.8044,-122.2711: 21.333333333333336
Overall combined score for 44.9778,-93.265: 19.5
Overall combined score for 39.1031,-84.512: 43.833333333333336
Overall combined score for 43.0731,-89.4012: 38.0
Overall combined score for 35.0456,-85.3097: 40.5
Overall combined score for 39.7397,-75.539: 53.166666666666667
Overall combined score for 32.6781,-83.222: 141.0
Overall combined score for 29.9511,-90.0715: 21.833333333333336
Overall combined score for 30.6954,-88.0399: 78.75
Overall combined score for 40.4406,-79.9959: 19.166666666666664
Overall combined score for 43.6532,-79.3832: 12.333333333333336
Overall combined score for 40.7357,-74.1724: 21.833333333333336
Overall combined score for 42.1015,-72.5898: 31.0
Overall combined score for 33.5207,-86.8025: 39.666666666666664
Overall combined score for 32.7765,-79.9311: 42.5
Overall combined score for 38.627,-90.1994: 56.5
Overall combined score for 36.0972,-79.7745: 92.75
Overall combined score for 40.7306,-73.9352: 50.666666666666667
Overall combined score for 41.6005,-93.6091: 69.5
Overall combined score for 36.7783,-119.4179: 105.25
Overall combined score for 40.6501,-73.9496: 25.0
Overall combined score for 34.0007,-81.0348: 41.333333333333336
Overall combined score for 36.7468,-119.7726: 61.666666666666667
Overall combined score for 48.7491,-122.4787: 36.166666666666664
Overall combined score for 36.1539,-95.9928: 43.666666666666664
Overall combined score for 27.3364,-82.5307: 39.0
Overall combined score for 40.014,-105.2705: 45.333333333333336
Overall combined score for 44.3148,-85.6024: 146.25
Overall combined score for 35.7796,-76.55: 137.75
Overall combined score for 32.3513,-87.02: 90.25
Overall combined score for 33.8333,-116.5453: 48.5
Overall combined score for 46.8772,-96.7898: 95.5
Overall combined score for 44.9537,-93.09: 32.5
Overall combined score for 39.7686,-94.8466: 65.5
Overall combined score for 36.1745,-86.7679: 81.5
Overall combined score for 40.8258,-73.2307: 85.0
Overall combined score for 35.3733,-119.0187: 55.666666666666667
Overall combined score for 44.0805,-103.231: 111.75
Overall combined score for 43.1242,-77.63: 42.0
Overall combined score for 42.8864,-78.8784: 35.166666666666664
Overall combined score for 32.2988,-90.1848: 69.5
```

Overall combined score for 39.5403,-119.7486: 31.25
Overall combined score for 27.9506,-82.4572: 32.166666666666664
Overall combined score for 30.4383,-84.2807: 64.66666666666666
Overall combined score for 45.7833,-108.5007: 34.0
Overall combined score for 41.4993,-81.6944: 45.666666666666664
Overall combined score for 33.4942,-111.9261: 115.33333333333334
Overall combined score for 46.7296,-94.6859: 57.0
Overall combined score for 40.7673,-111.8902: 83.0
Overall combined score for 39.195,-106.837: 58.25
Overall combined score for 48.0518,-122.1771: 43.0
Overall combined score for 30.2241,-92.0198: 90.75
Overall combined score for 29.9511,-95.0584: 154.75
Overall combined score for 37.947,-122.0652: 38.33333333333333
Overall combined score for 40.7608,-111.891: 50.666666666666664
Overall combined score for 44.9637,-92.9594: 57.33333333333333
Overall combined score for 39.739,-75.564: 53.83333333333333

Traffic Congestion × AQI Category Analysis

This section summarizes the average traffic congestion levels for each AQI category.
Congestion is calculated as: freeflow_speed - current_speed.

AQI Category: Good
- Data points: 75
- Average congestion: 4.73

AQI Category: Moderate
- Data points: 26
- Average congestion: 6.15