PART 5 – Report (100 points)
In addition to your API activity results, you will be creating a report for your overall project. The report must include:

1. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)
2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)
3. The problems that you faced (10 points)
4. The calculations from the data in the database (i.e. a screen shot) (10 points)
5. The visualization that you created (i.e. screen shot or image file) (10 points)
6. Instructions for running your code (10 points)
7. An updated function diagram with the names of each function, the input, and output and who was responsible for that function (20 points)
8. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

Date	**Issue Description**	**Location of Resource**	**Result (did it solve the issue?**

Public Repository link: https://github.com/bmlhodge17/SI201-Final-Project.git

Team Name: JAB
Members: Asiah Bays, Brianna Hodge, Jasmine Abu

1. Goals/APIs/websites

Our main goal: Is biking networks and biking activity in cities related to weather conditions and cost of living? Which one has a stronger influence or correlation?

Other goals:

- Analyzing city data that relates to biking networks, weather, and cost related information

2. Goals achieved:

- APIs we actually used (different then our project plan):
    - Citybikes API
    - WeatherStack API
    - Cities Living Cost API
- Creating new database and tables
- Creating our visualizations/graphs

- Another goal we achieved was learning how to use Github for a group project. Learning how to pull, push, and make commits without messing up each other's code and progress.
- We were able to successfully gather data about city biking networks and their relationship to the city's weather description

3. Challenges faced:

- Finding new API's than what we submitted for our project plan. After this we had to change the main focus of our project as well.
- Deciding what API's to work with that would allow us to draw thoughtful conclusions and compare data with our calculations/graphs.
- Creating a new database without any duplicate string was a challenge for our project. These coding concepts are still fairly new to us as novice programmers. This took time for us to work around.
- Having to re-read the rubric and project instructions. Project instructions seemed contradictory and confusing.
- Working together on vs code and messing up anyone else's work was very hard

4. Calculations used from the database (screenshots)

Jasmine's calculation:

```
#calculating the average salary in the first 25 cities in the cost index table
def average_salary_first_25():
    conn, cur = get_connection()
    query = """
        SELECT AVG(monthly_salary)
        FROM cost_index
        WHERE city_id IN (
            SELECT city_id
            FROM cost_index
            ORDER BY city_id
            LIMIT 25
        )
        AND monthly_salary IS NOT NULL;
    """
    cur.execute(query)
    avg_salary = cur.fetchone()[0]
    conn.close()
    print(avg_salary)
```

```python
#jasmines calculation
# joined table calculation


def plot_join_table(conn: sqlite3.Connection) -> None:
    cur = conn.cursor()

    # Select data from join table
    cur.execute("""
        SELECT monthly_salary, gasoline_price
        FROM joined_table
        WHERE monthly_salary IS NOT NULL
          AND gasoline_price IS NOT NULL;
    """)

    rows = cur.fetchall()

    if not rows:
        print("No data available for scatter plot.")
        return

    salaries = [row[0] for row in rows]
    gas_prices = [row[1] for row in rows]

    # Create scatter plot
    plt.figure()
    plt.scatter(salaries, gas_prices)
    plt.xlabel("Average Monthly Salary")
    plt.ylabel("Gasoline Price (per liter)")
    plt.title("Monthly Salary vs Gasoline Price by City")
    plt.show()
```

Brianna's Calculation

```python
# Get connection
def get_connection():
    conn = sqlite3.connect(DB_PATH)
    cur = conn.cursor()

    return conn, cur


# Bri CALCULATION 1: Networks per country
from SQL_Data_base import connect_to_database

def networks_per_country():
    """
    Returns two lists:
    - countries
    - network counts
    sorted from most to least networks
    """
    conn, cur = connect_to_database()

    cur.execute("""
        SELECT country, COUNT(*) AS num_networks
        FROM cities
        WHERE country IS NOT NULL
        GROUP BY country
        ORDER BY num_networks DESC;
    """)

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

    # separate into two lists (same pattern as other calculations)
    countries = [row[0] for row in rows]
    counts = [row[1] for row in rows]

    return countries, counts
```

```python
# Function to plot the top countries
def plot_networks(countries, counts, top_n=15):
    countries = countries[:top_n]
    counts = counts[:top_n]

    plt.figure(figsize=(12, 6))
    bars = plt.bar(countries, counts)
    for bar in bars:
        bar.set_color("green")

    plt.xticks(rotation=45, ha="right")
    plt.xlabel("Country")
    plt.ylabel("Number of Networks")
    plt.title(f"Top {top_n} Countries by Number of CityBike Networks")
    plt.tight_layout()
    plt.show()
```

Asiah's Calculations:

```python
def plot_uv_index_histogram():
    conn, cur = connect_to_database()
    # get uv index values from weather table
    cur.execute("""
        SELECT uv_index
        FROM weather
        WHERE uv_index IS NOT NULL;
    """)
    rows = cur.fetchall()
    conn.close()

    uv_values = [row[0] for row in rows]

    if not uv_values:
        print("no uv index data to plot")
        return
    # uv index bins grouped by 2
    bins = [0, 2, 4, 6, 8, 10, 12]

    plt.figure(figsize=(8, 5))

    # create histogram and capture bars
    counts, bin_edges, patches = plt.hist(
        uv_values,
        bins=bins,
        edgecolor="black"
    )
    # colors for each uv range
    colors = [
        "#7FB3D5",  # low
        "#76D7C4",  # moderate
        "#F7DC6F",  # high
        "#F5B041",  # very high
        "#EB984E",  # extreme
        "#E74C3C"   # extreme+
    ]
    # apply colors to each bin
    for patch, color in zip(patches, colors):
        patch.set_facecolor(color)
    plt.xlim(0, 12)
    plt.xlabel("uv index")
```

```python
def plot_weather_description_dotplot():
    conn, cur = connect_to_database()

    # get city names and weather descriptions
    cur.execute("""
        SELECT city_name, weather_description
        FROM weather
        WHERE weather_description IS NOT NULL;
    """)

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

    if not rows:
        print("no weather description data to plot")
        return

    # group cities by weather description
    grouped = {}
    for city, desc in rows:
        grouped.setdefault(desc, []).append(city)

    x_vals = []
    y_vals = []
    labels = []

    # build dot positions
    for desc, cities in grouped.items():
        for i, city in enumerate(cities):
            x_vals.append(desc)
            y_vals.append(i + 1)
            labels.append(city)

    plt.figure(figsize=(16, 7))
    plt.scatter(x_vals, y_vals)

    # add city labels next to each dot
    for x, y, label in zip(x_vals, y_vals, labels):
        plt.text(
            x,
            y + 0.05,
```

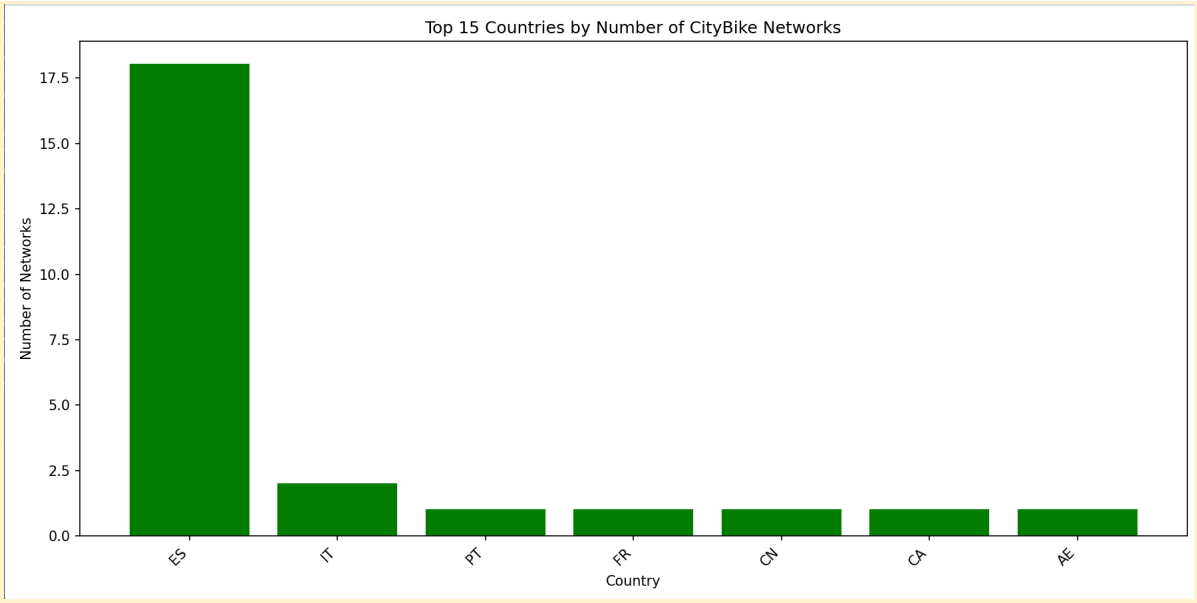
```python
def plot_top_10_hottest():
    conn, cur = connect_to_database()

    cur.execute("""
        SELECT city_name, temperature
        FROM weather
        WHERE temperature IS NOT NULL
        ORDER BY temperature DESC
        LIMIT 10;
    """)

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

    if not rows:
        print("no temperature data for hottest cities")
        return

    cities = [row[0] for row in rows]
    temps = [row[1] for row in rows]

    plt.figure(figsize=(10, 6))
    plt.barh(cities, temps)
    plt.xlabel("temperature (celsius)")
    plt.ylabel("city")
    plt.title("top 10 hottest cities (celsius)")
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
```

```python
# top 10 lowest temperatures
def plot_top_10_coldest():
    conn = sqlite3.connect(DB_PATH)
    cur = conn.cursor()

    cur.execute("""
        SELECT city_name, temperature
        FROM weather
        WHERE temperature IS NOT NULL
        ORDER BY temperature ASC
        LIMIT 10;
    """)

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

    if not rows:
        print("no temperature data for coldest cities")
        return

    cities = [row[0] for row in rows]
    temps = [row[1] for row in rows]

    plt.figure(figsize=(10, 6))
    plt.barh(cities, temps)
    plt.xlabel("temperature (celsius)")
    plt.ylabel("city")
    plt.title("top 10 coldest cities (celsius)")
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
```

5. Visualizations created (screenshots or files)

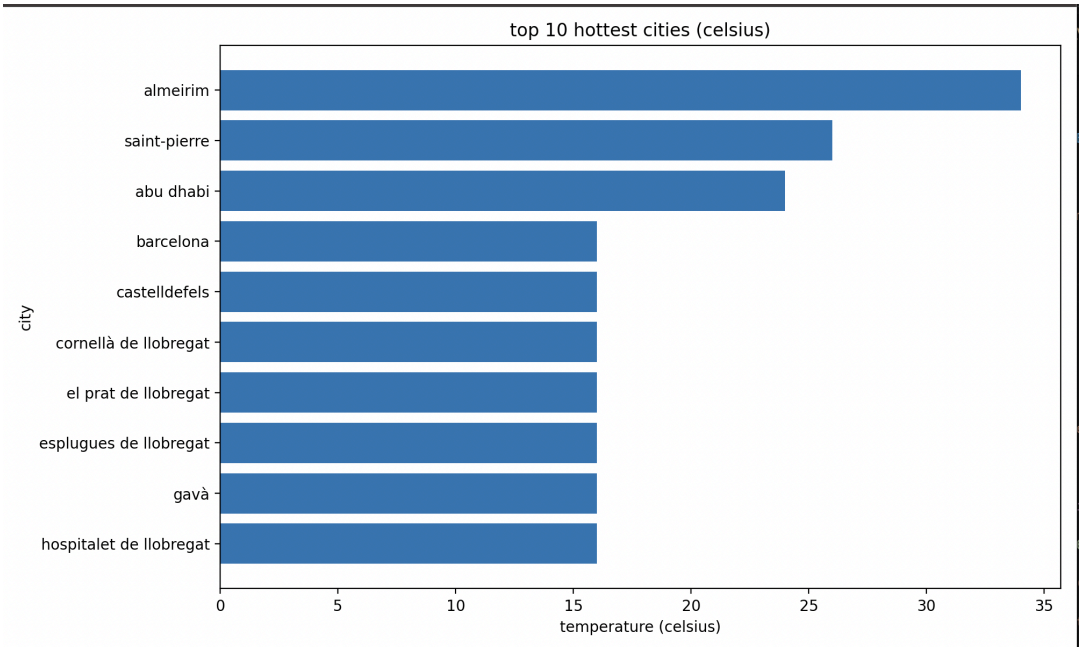
Bri CityBikes API Matplotlib Visualization:

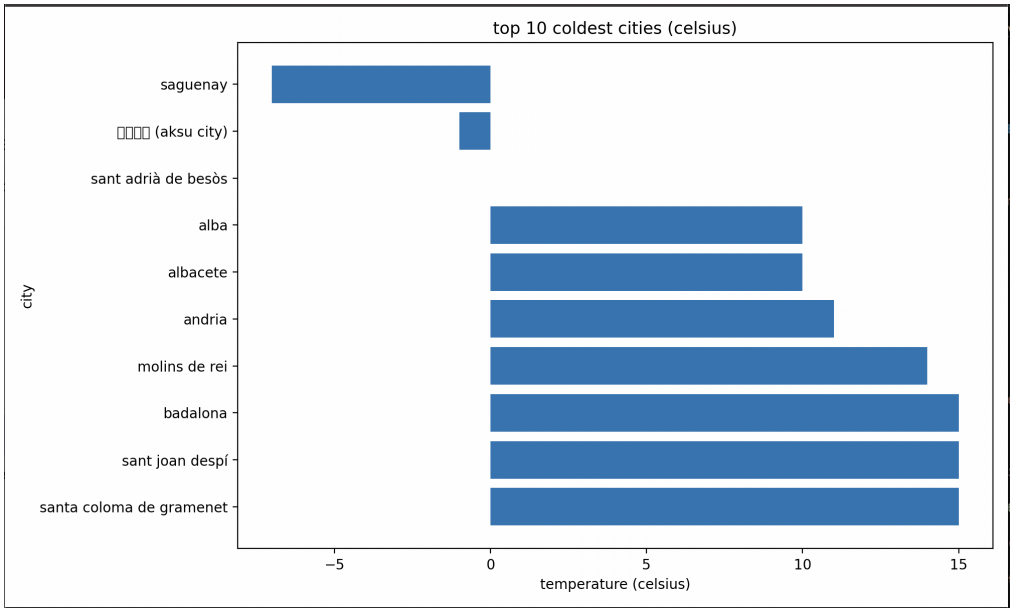
Top 15 Countries by the Number of CityBike Networks

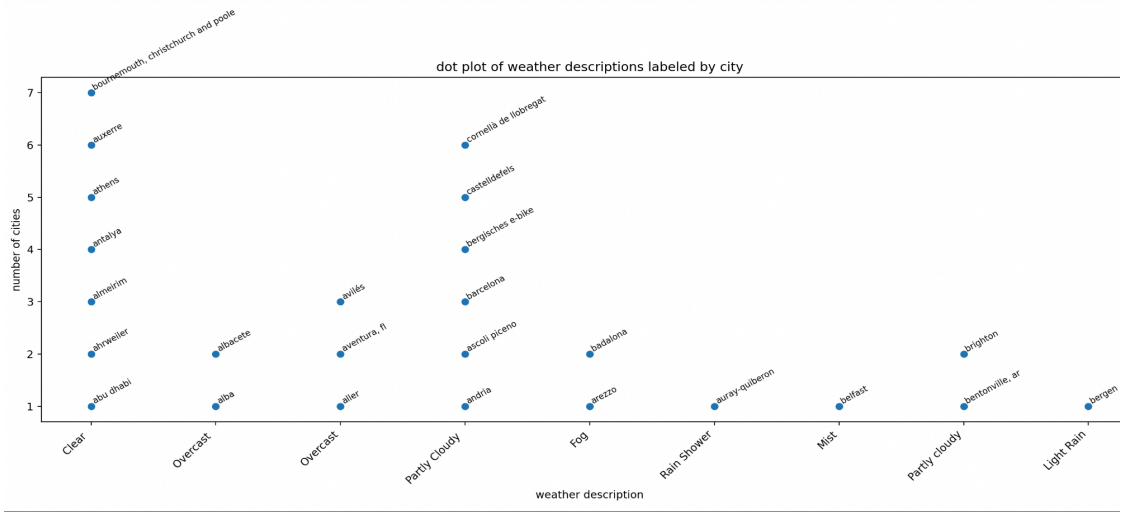


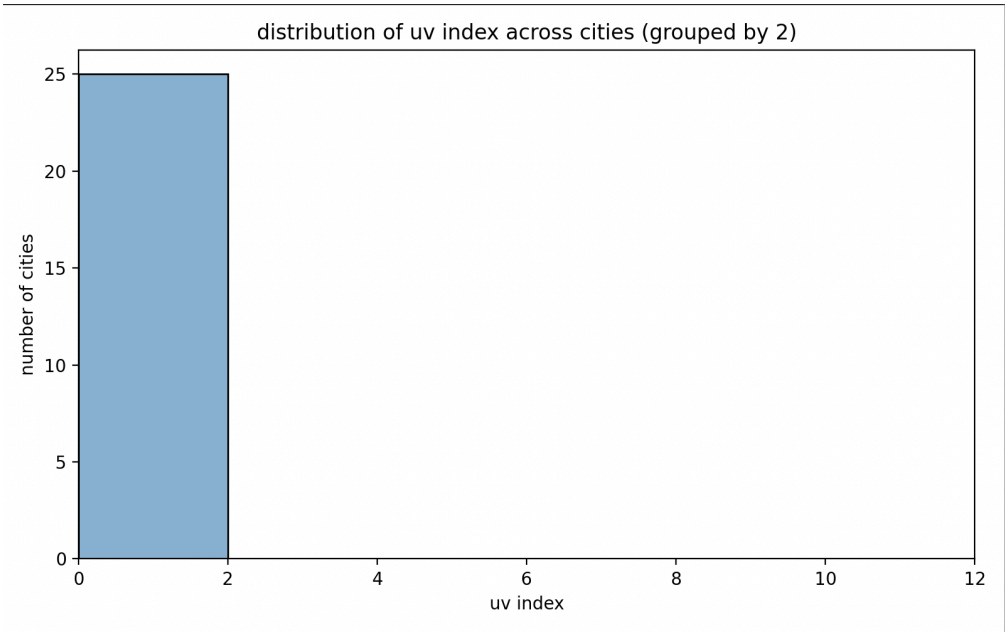
Jasmine's visualization

Asiah's Visualization




top 10 coldest cities (celsius)
saguenay
□□□□ (aksu city)
sant adrià de besòs
alba
albacete
andria
molins de rei
badalona
sant joan despí
santa coloma de gramenet
city
−5
0
5
10
15
temperature (celsius)


dot plot of weather descriptions labeled by city
bournemouth, christchurch and poole
auxerre
athens
antalya
almeirim
ahrweiler
abu dhabi
albacete
alba
aviléS
aventura, fl
aller
cornellà de llobregat
castelldefels
bergisches e-bike
barcelona
ascoli piceno
andria
badalona
arezzo
auray-quiberon
belfast
brighton
bentonville, ar
bergen
number of cities
1
2
3
4
5
6
7
Clear
Overcast
Overcast
Partly Cloudy
Fog
Rain Shower
Mist
Partly cloudy
Light Rain
weather description



only one bin full because the db was loaded at night

6. Instructions for running our code

How to run our code:

Each of us have our own API.py file that will use an API key (not for Citybikes since it doesn't require an API key) to access the API and return into our summary json files. (Note: For the JA.py that uses the Kaggle data set, the api key is in a JSON file. The Json file (API key) is accessed in the terminal to download the data set. A screenshot of how the API key was accessed is in a screenshot below)

To create our JAB_Database run the FINAL_DATABASE_CODE.py file

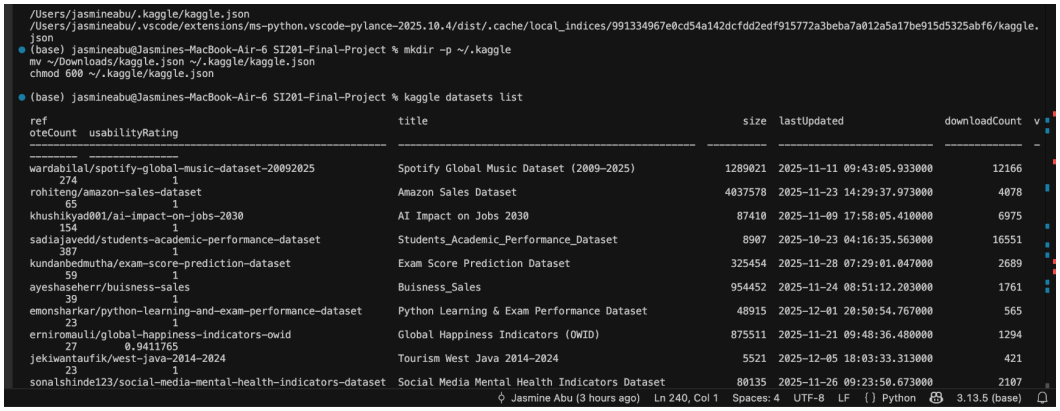
This should create the JAB_DATABASE and our tables (the weather table takes a bit longer than the rest, so you may have to refresh when you see the "weather populated print statement in terminal)

The Calc_Vis.py is our code that executes our calculations and makes our visualizations using Matplotlib (after running the graphs should pop up)

a. Note: click the X in the top corner of the screen to show the rest of the graphs

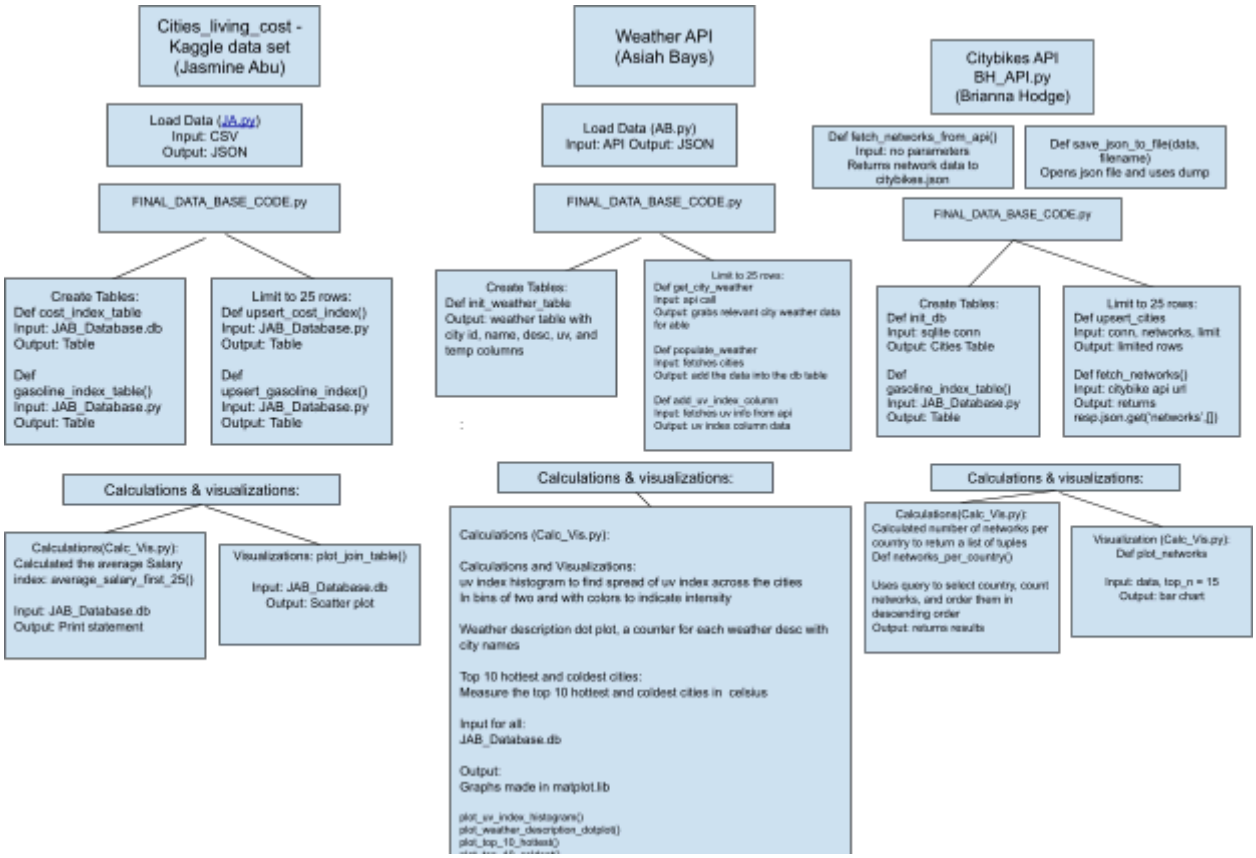
Summary file: Our SI201-Final-Project folder should also contain the summary json files for our APIs

Kaggle JSON/API key



(Accessing the my Kaggle API key and using the key to browse through public data sets)

7. Updated function diagram



8. Our Resources Used:

Date	Issue Description	Location of Resource	Result (did it solve the issue)?
11/22	Free API Github source, changed our APIs	https://github.com/public-apis/public-apis	Yes, we were able to pick new apis for our project
12/9	Accessing the cost of living in other cities	https://www.kaggle.com/datasets/georgeliarommatis/world-cities-living-cost	Finding new API
12/1-12/10	SQL Canvas Cheat Sheet	https://umich.instructure.com/courses/789559/files/41680991?module_item_id=4624505	Helped us code in SQL
Throughout the project	ChatGPT	ChatGPT.com	Help answer questions for errors and using class examples to explain coding concepts
Throughout the project	Tips for Final Project Slides	Lecture 22 Slides	Helped our understanding of duplicate string data and limiting 25 rows