

Repo Link: [CLICK HERE](#)

# Report on Marvel APIs

Stan Leevasz



## **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)

<b>Date</b>	<b>Issue Description</b>	<b>Location of Resource</b>	<b>Result (did it solve the issue?)</b>
-------------	--------------------------	-----------------------------	---

# Planned: Project Goal, APIs Used, & Data Gathered

- ❖ The project's goal is to investigate the representation of different identities in superhero media.
- ❖ APIs Used:
  - Superhero API: <https://superheroapi.com/try-now.html>
  - Comic Vine API: <https://comicvine.gamespot.com/api/documentation>
  - Marvel Rivals API: <https://marvelrivalsapi.com/>
  - MCU Countdown: <https://github.com/DiljotSG/MCU-Countdown>
- ❖ Gathered Data:
  - Height & weight of superheroes, gender of superheroes, birthplace of superheroes, movies & comics appearances of superheroes.
  - Seeing representation in:
    - BMI
    - Gender
    - Birthplace
    - Comics
    - Movies
    - Games

# Achieved: Project Goal, APIs Used, & Data Gathered

- ❖ The project's goal is to investigate the representation of different identities in superhero media.
- ❖ APIs Used:
  - Superhero API: <https://superheroapi.com/try-now.html>
  - Comic Vine API: <https://comicvine.gamespot.com/api/documentation>
  - Marvel Rivals API: <https://marvelrivalsapi.com/>
- ❖ Gathered Data:
  - Height & weight of superheroes, gender of superheroes, birthplace of superheroes, & comics appearances of superheroes.
  - Seeing representation in:
    - BMI
    - Gender
    - Birthplace
    - Comics
    - Games

# Problems Faced

- ❖ Had to transition from using MCU Countdown to Marvel Rivals API
  - MCU Countdown only gave us data for the upcoming movie, not previous Marvel films
- ❖ No movie data
  - Comic Vine's documents weren't up to date
    - we couldn't find how many movies certain heroes were featured in
- ❖ Average Runtime: 15 min.
- ❖ 429 Errors
  - Largely occurred when problem solving data issues
- ❖ Excluding non-Marvel heroes from lists

```

5 def most_played_characters(file, filename):
6     """
7     Arguments: str file, str filename
8
9     Return: None
10
11     Iterating through the characters used and ordering them
12     through most played in a text file for later use
13     """
14     results = {"Male": 0, "Female": 0, "Unknown": 0}
15     cur, conn = marvel_rivals.set_up_database(file)
16
17     cur.execute(
18         """
19         SELECT mr_id, gender_id FROM superheroes
20         """
21     )
22
23     hero_lst = cur.fetchall()
24
25     for hero in hero_lst:
26         count = 0
27         id = hero[0]
28
29         if id != None:
30             for i in range(1,6):
31                 #iterating over a single match at a time so I'm not getting the whole row
32                 cur.execute(
33                     f"""
34                     SELECT character_by_match.match{i} FROM character_by_match
35                     WHERE character_by_match.match{i} = {id}
36                     """
37                 )
38
39                 #the length of fetchall should contain number of times character is used
40                 num = len(cur.fetchall())
41                 count += num
42
43                 cur.execute(
44                     f"""
45                     SELECT genders.gender FROM genders
46                     WHERE genders.gender_id = {hero[i]}
47                     """
48                 )
49
50                 gender = cur.fetchone()[0]
51
52                 results[gender] += count
53
54     try:
55         open(filename)
56     except:
57         #Only writing if the file doesn't already exist
58         with open(filename, "w") as f:
59             f.write("gender, number of times played")
60             for key in list(results.keys()):
61                 f.write(f"\n{key},{results[key]}")
62
63     conn.close()
64     pass

```

# Calculations from Data in Database - Blake

```

def gender_by_comics(db, filename):
    """
    Arguments: db, filename

    Returns: None

    Going through the superhero.db to grab how many issues different genders feature in
    """
    cur, conn = comicvine.set_up_database(db)
    results = {"Male": 0, "Female": 0, "Unknown": 0}

    cur.execute(
        """
        SELECT numcomics.comics, genders.gender FROM superheroes
        JOIN numcomics ON superheroes.num_comics = numcomics.id
        JOIN genders ON superheroes.gender_id = genders.gender_id
        """
    )

    heroes = cur.fetchall()

    for hero in heroes:
        if hero[0] != "NULL":
            num_comics = int(hero[0].split(" ")[0])
        else:
            num_comics = 0
        results[hero[1]] += num_comics

    try:
        open(filename)
    except:
        with open(filename, "w") as f:
            f.write("gender, number of issues")
            for key in list(results.keys()):
                f.write(f"\n{key},{results[key]}")

    conn.close()
    pass

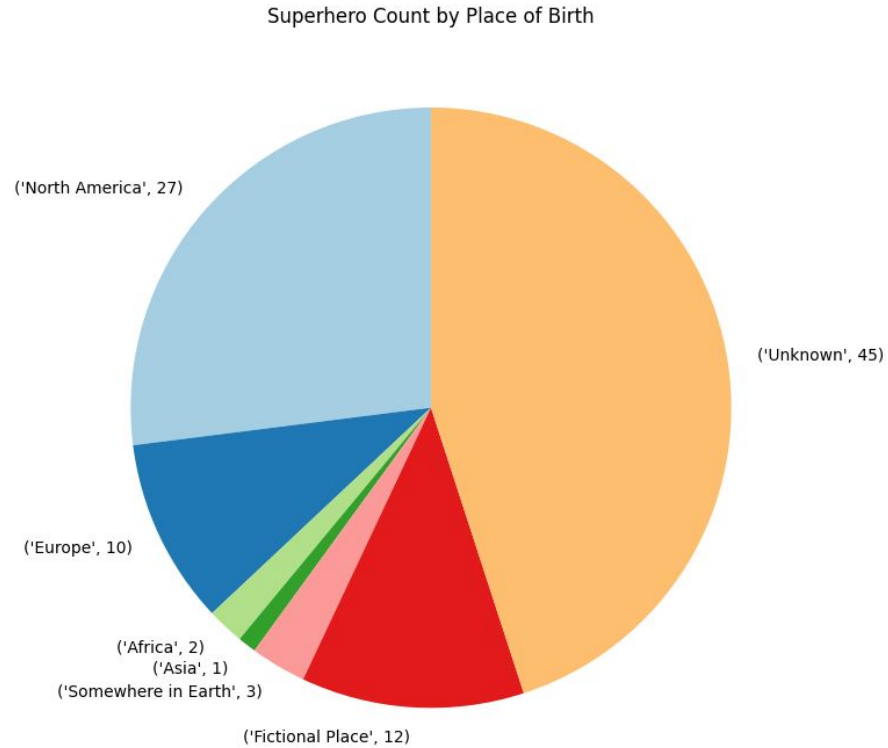
```

# Calculations from Data in Database - Stan

```
def superhero_api_calculations(db_file, filename, filename2):
    """
    Args: str db_file, str filename
    Out: none
    Calculates the count of superhero for each place or birth and average BMI for superhero by gender
    """
    conn, cur = superhero.setup_db(db_file)
    cur.execute("""
        SELECT places_of_birth.place_of_birth
        FROM superheros
        JOIN places_of_birth
        ON superheros.place_of_birth_id = places_of_birth.place_of_birth_id
    """)
    places_of_birth = cur.fetchall()
    places_count = {
        "North America": 0,
        "Europe": 0,
        "Africa": 0,
        "Asia": 0,
        "Somewhere in Earth": 0,
        "Fictional Place": 0,
        "Unknown": 0
    }
    for place in places_of_birth:
        if place[0] in places_count:
            places_count[place[0]] += 1
    try:
        open(filename)
    except:
        with open(filename, "w") as f:
            for origin, count in places_count.items():
                f.write(f"There are {count} superheroes from {origin}\n")
```

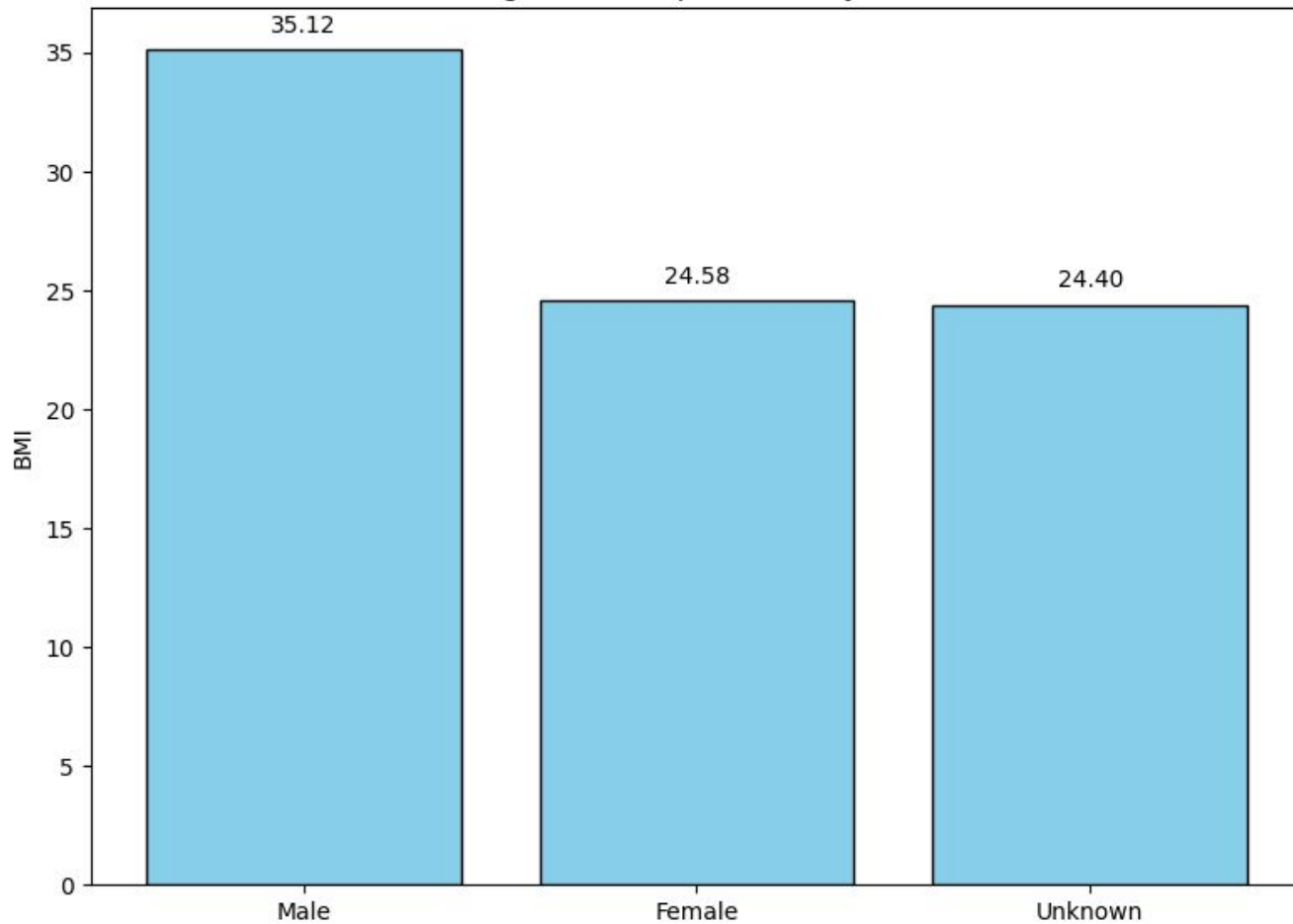
```
cur.execute("""
    SELECT genders.gender, superheros.height, superheros.weight
    FROM superheros
    JOIN genders
    ON superheros.gender_id = genders.gender_id
""")
gender_height_weight = cur.fetchall()
gender_counts = {"Male": 0, "Female": 0, "Unknown": 0}
gender_bmi = {"Male": 0, "Female": 0, "Unknown": 0}
for g, height, weight in gender_height_weight:
    if g in gender_counts and height and weight:
        try:
            bmi = weight / ((height / 100) ** 2)
            gender_counts[g] += 1
            gender_bmi[g] += bmi
        except Exception:
            continue
    try:
        open(filename2)
    except:
        with open(filename2, "a") as f: # append so both results are in same file
            for gender, total_bmi in gender_bmi.items():
                if gender_counts[gender] > 0:
                    avg_bmi = total_bmi / gender_counts[gender]
                    f.write(f"The average BMI for {gender} superheroes is {avg_bmi:.2f}\n")
                else:
                    f.write(f"No data for {gender} superheroes\n")
```

# Visualizations Created

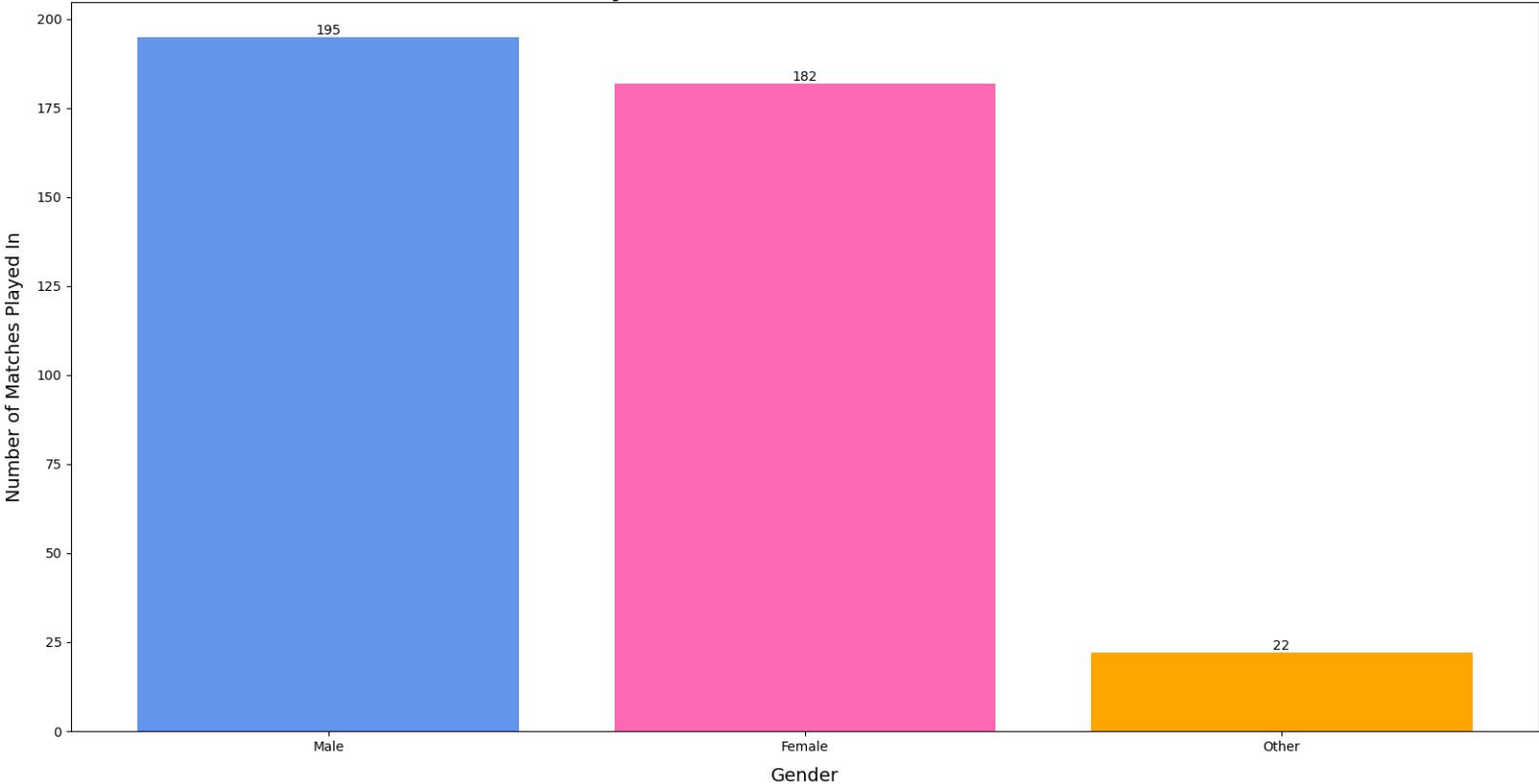




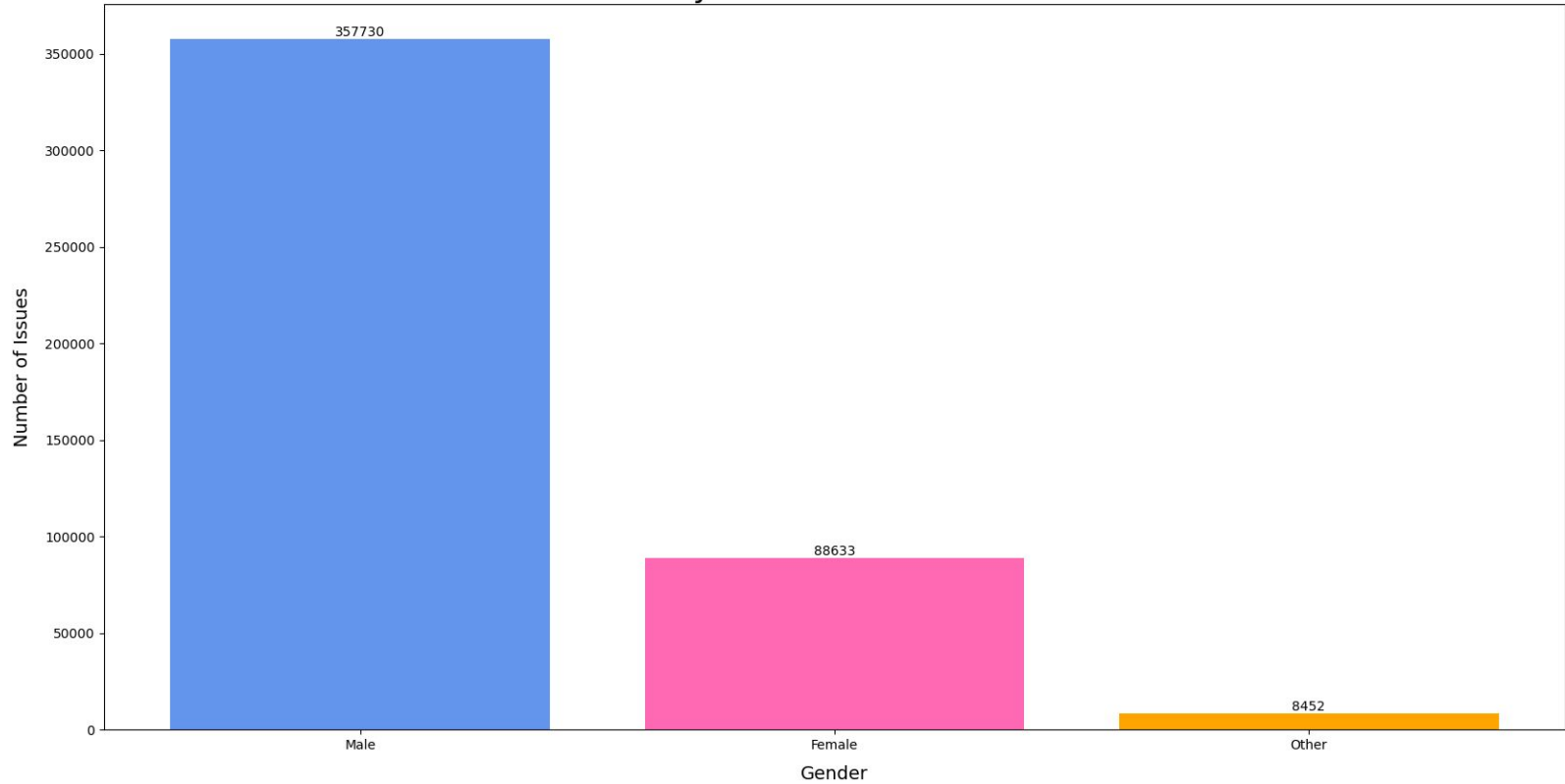
Average BMI of Superheroes by Gender



Heroes by Gender in Marvel Rival Matches



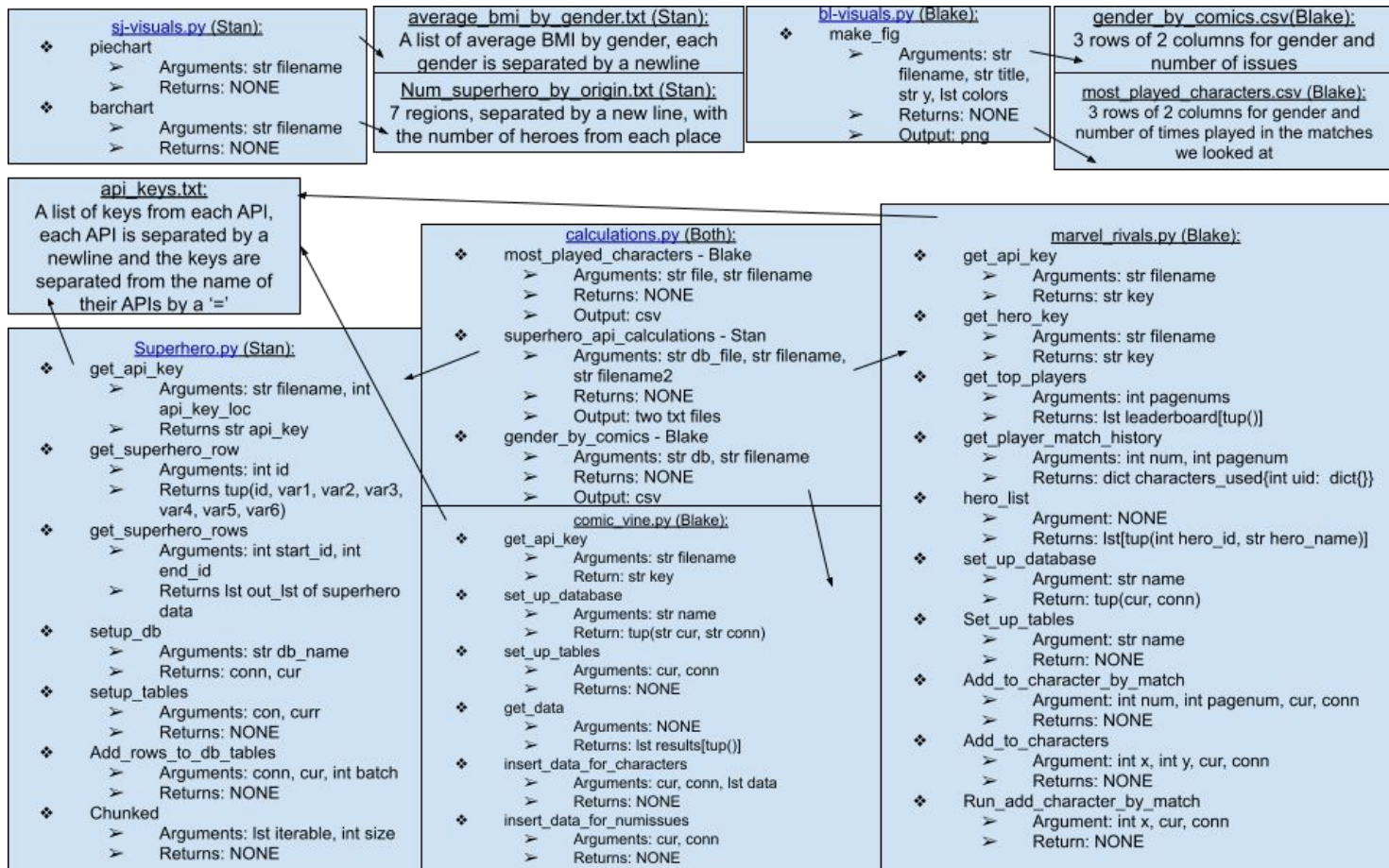
# Heroes by Gender in Marvel Comics



# Instructions for Running the Code

1. Create an API key from <https://superheroapi.com/>, <https://comicvine.gamespot.com/api/>, and <https://marvelrivalsapi.com/>.
2. Insert each API key into the “replace text with key” on each corresponding line in the api\_keys.txt file.
3. Run the superhero.py file first to create and populate the main database: [superhero.db](#).
4. Run the marvel\_rivals.py file second to create and populate its tables to the superhero.db database.
5. Run the superhero.py file third to create and populate its tables to the superhero.db database.
6. Run the calculations.py file to perform calculations using data from the superhero.db database to create the data calculations files: gender\_by\_comics.csv, most\_played\_characters.csv, num\_superhero\_by\_origin.txt, and average\_bmi\_by\_gender.txt.
7. Run the bl-visuals.py and sj-visuals.py files to create the visualizations from calculated data.

# Function Diagram



# Resources Used

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
11/18/2025	Difficulty in getting access to Marvel Rivals API key.	<a href="https://discord.com/invite/marvelrivalsapi">https://discord.com/invite/marvelrivalsapi</a>	Yes, the issue was solved.
11/24/2025	Long runtime in creating a database for Superhero API.	<a href="https://prismic.io/blog/api-response-times">https://prismic.io/blog/api-response-times</a>	Yes, the issue was solved.
12/1/2025	References for colors and text for matplotlib for customization of the charts	<a href="https://matplotlib.org/stable/api/matplotlib_configuration_api.html">https://matplotlib.org/stable/api/matplotlib_configuration_api.html</a>	Yes, the issue was solved.