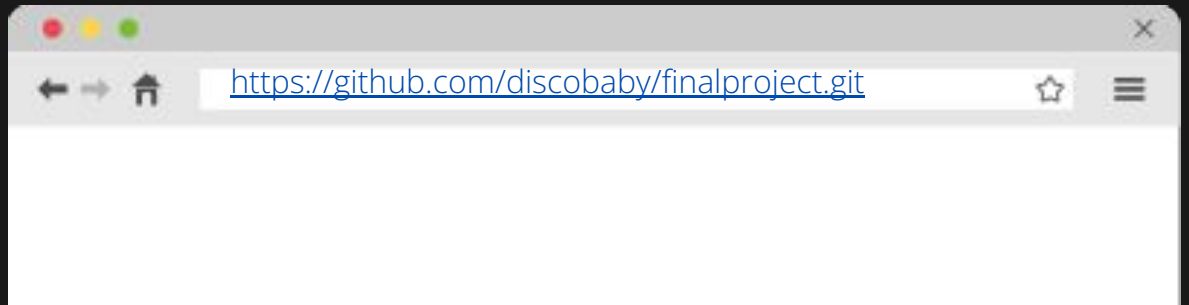
SI206 Final Project

By: Lizzie Potocsky, Ember Shan, and Ben Decker

Github Link



Original Goals

- Analyze the relationship between a celebrity's net worth and their average number of fans
- Analyze the relationship between gender and net worth, as well as gender and number of followers on both Spotify and Twitter
- Create data visualizations to answer the following questions:
  - Do certain music genres lead artists to having more followers on social media?
  - As artists gain popularity from their music, do they also gain popularity (i.e. followers) on subsequent social media platforms?

Achieved Goals

- We were able to collect data to see if an artist's number of Spotify followers and Twitter followers correlate in some way
- We customized a database consisting of data related to the top 200 artist's Twitter/Spotify followers, average number of fans, gender, genre, and net worth.
- We created visualizations that illustrated the potential correlations between gender and net worth and also between gender and follower count.
- Successfully created a bar chart that shows the average number of artist Twitter followers for a corresponding Spotify music genre

Problems Faced

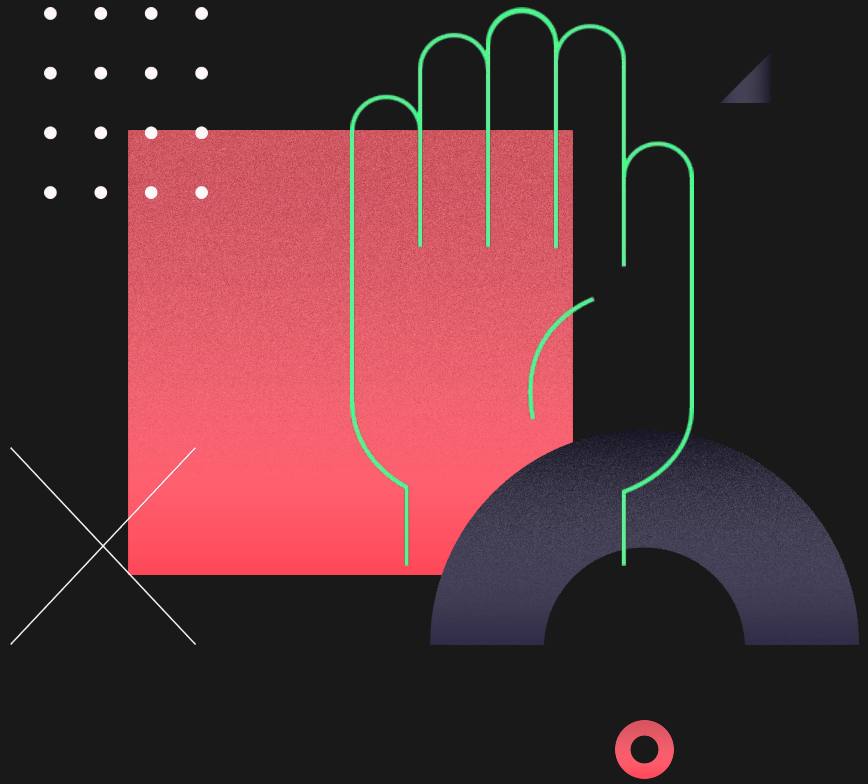
- Originally, we wanted to get the celebrity's number of followers from the Tiktok API
  - Tiktok required developers to host a website in order to get this data
  - Switched to Twitter API
- Some artist names were band names, but our net worth API could only deal with one person
- Artists had the same names
  - Solved by using comparison operators to see if the name returned was the exact same as the artist name retrieved from the artist table
- Celebrity API did not have all information; empty fields represented with a value of -1 or N/A

Problems Faced

- Spotify API required an account that gives a “client id” and “client secret”
  - We had to add these values in the code and create a variable in order to get an access token that could be used in the functions that follow

```
client_id = '88d2a36f58ac44b89789ef724bfe42a3'
client_secret = '1edb3b2408b2485ba355482c0960637b'
#get authorization with POST request
auth_url = 'https://accounts.spotify.com/api/token'
data = {
    'grant_type': 'client_credentials',
    'client_id': client_id,
    'client_secret': client_secret,
}
auth_response = requests.post(auth_url, data=data)
#get access token
access_token = auth_response.json().get('access_token')
#print(access_token)
```

CALCULATIONS



Calculations From Database

Used function **getTwitterFollowersByGenre**

```python
#Creating my main function to iterate through each genre, get the number of twitter followers for each artist that has the specific genre_id, and then get the average.
# Then I plot the data.
def getTwitterFollowersByGenre(cur, conn):
    #Calling my function to get the list of genre id's
    genre_id_list = createGenrelist(cur, conn)
    #Creating an empty list to add the average followers to
    avg_list = []
    #Calling the function to get the list of genre id names
    genre_name_list = createGenrelistname(cur, conn)

    # Iterating through the genre_ids to get my follower count data, which is the first item in a tupple.
    for i in genre_id_list[0]:
        cur.execute('SELECT follower_count FROM TwitterData JOIN SpotifyArtistData ON TwitterData.id = SpotifyArtistData.id WHERE SpotifyArtistData.genre_id=?', (i[0],)
        conn.commit()
        rows = cur.fetchall()
        #Creating empty follower list and adding the artists' followers to that for.
        follower_list = []
        follower_list.append(rows)
        count = 0
        artists = 0
        for num in follower_list:
            #Doing this to avoid the divide by zero or any index error, because some genres have 0 twitter followers averages.
            if len(num) == 0:
                pass
            else:
                count += int(num[0][0])
        #Getting the length of follower list to divide by and get the averages
        artists = len(follower_list)
        average_num = (int(count) / int(artists))
        #Appending my averages to a list, for all of the genres.
        avg_list.append(average_num)
```

Output:

73 lines (73 sloc) | 1.62 KB

```
1. genre, 2. average twitter followers
barbadian pop, 103706043.0
canadian hip hop, 39369362.0
permanent wave, 23537346.0
detroit hip hop, 22518439.0
pop, 14258529.0
chicago rap, 30623236.0
dance pop, 43172959.0
irish rock, 1591868.0
british soul, 27004587.0
canadian contemporary r&b, 15277855.0
canadian pop, 114158582.0
hip hop, 2746402.0
conscious hip hop, 11580537.0
alternative rock, 4796893.0
modern rock, 5681263.0
art pop, 83952770.0
big room, 20276193.0
atl hip hop, 12002504.0
classic rock, 302244.0
```

Calculations From Database

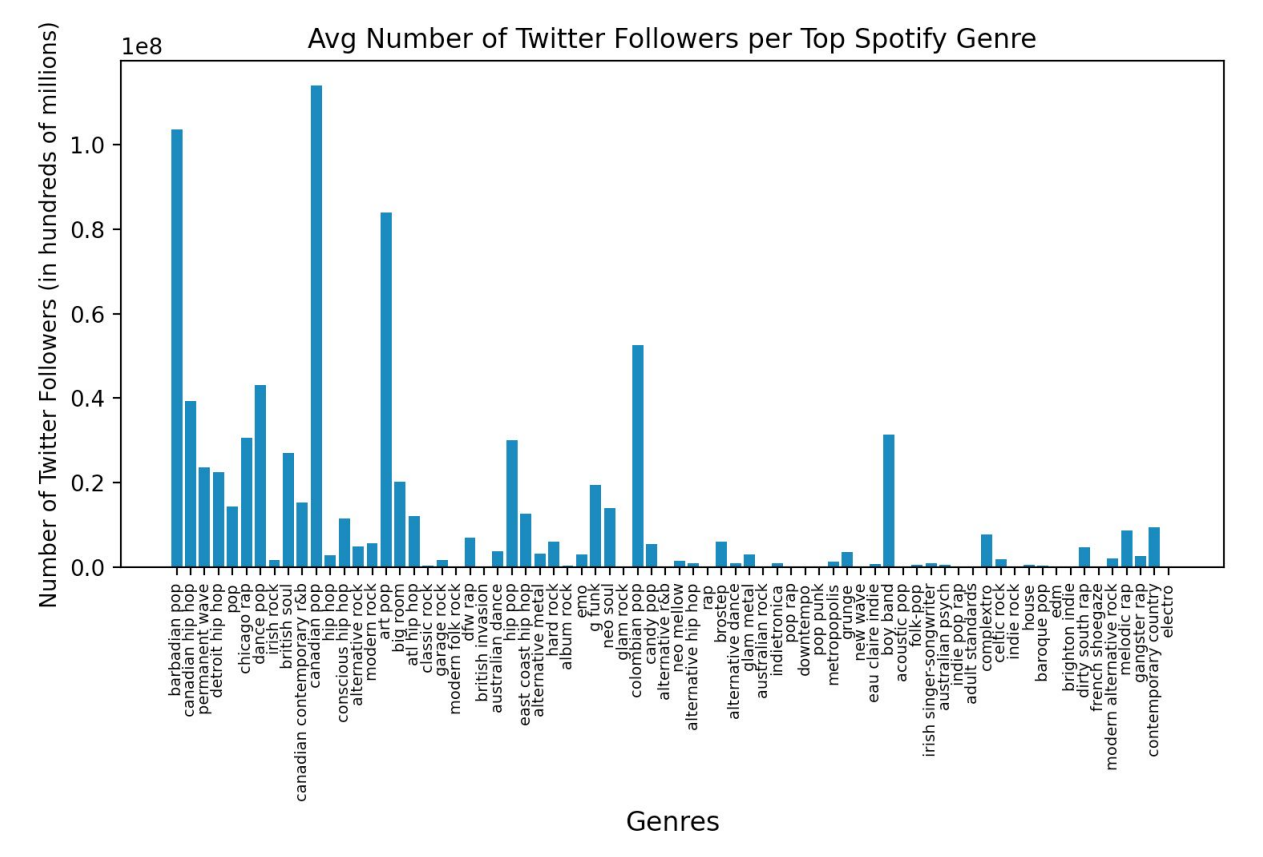
Used function **getAvgNetworthByGender**

```
def getAvgNetworthByGender(cur):
    # extracting the data
    cur.execute("SELECT net_worth FROM NetWorth WHERE gender = 'male'")
    male_networth = [x[0] for x in cur.fetchall()]
    cur.execute("SELECT net_worth FROM NetWorth WHERE gender = 'female'")
    female_networth = [x[0] for x in cur.fetchall()]
    # calculate the averages
    avg_m = sum(male_networth) / len(male_networth) / 1000000
    avg_f = sum(female_networth) / len(female_networth) / 1000000
```

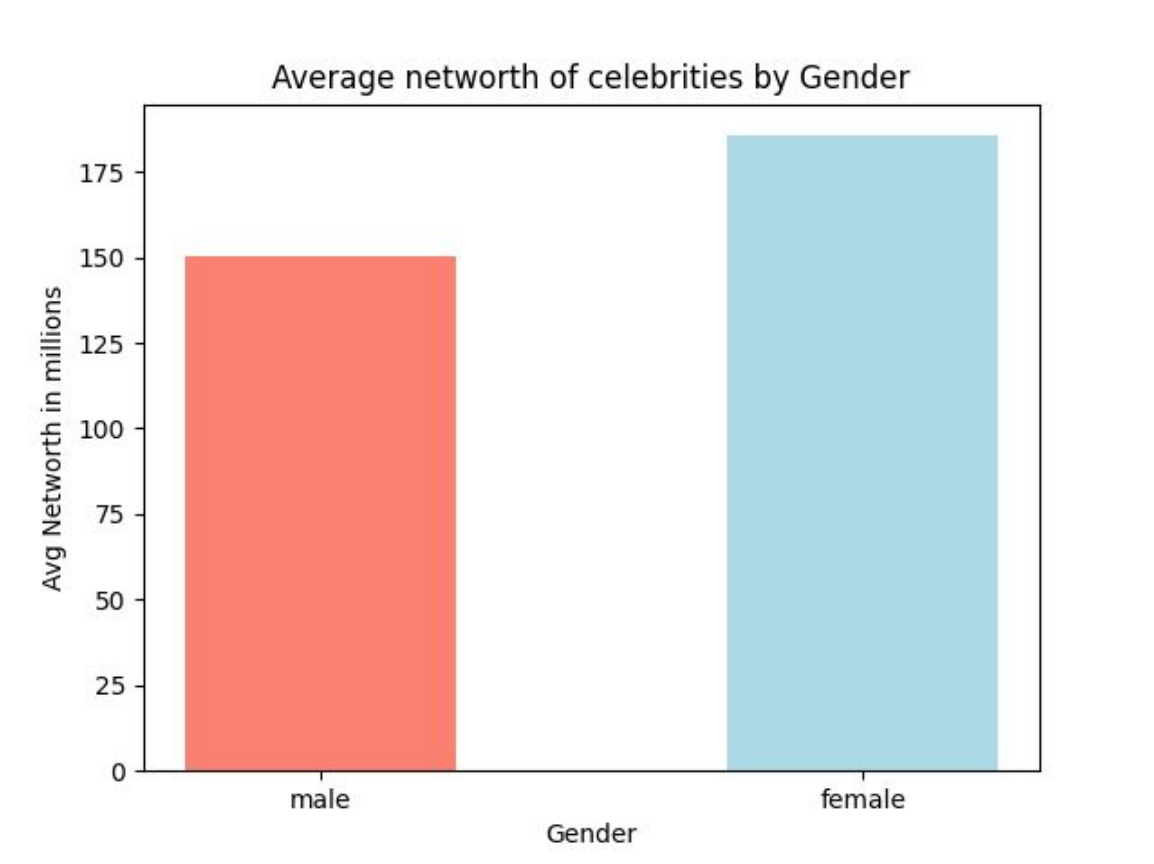
Output:

```
average networth of male celebrities: 150.49295774647888, average networth of female celebrities: 185.5
```

VISUALIZATIONS

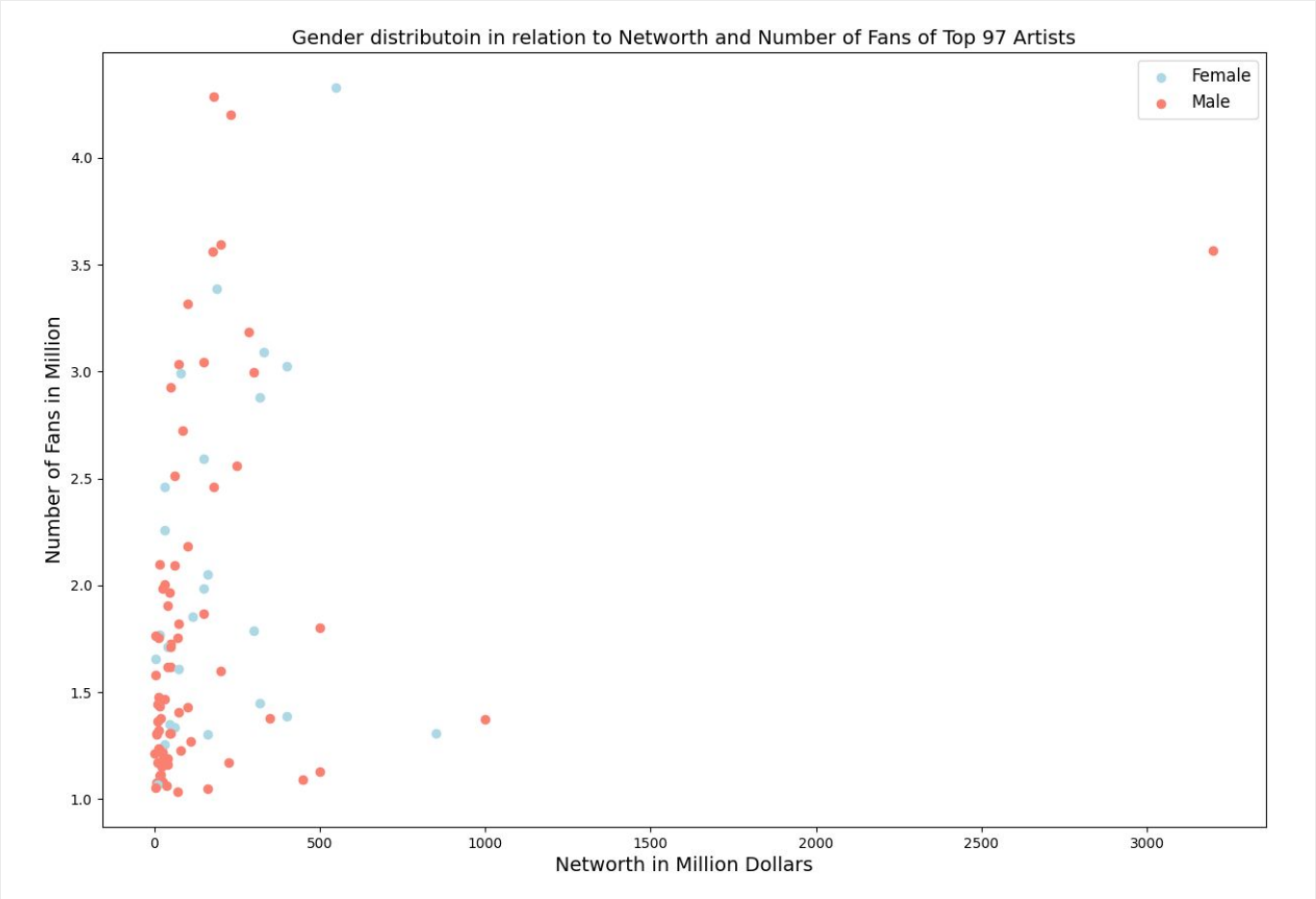


- Each bar represents a genre and the average number of Twitter followers
- Here, it appears that artists in “pop” genres have significantly more followers on Twitter than artists of other music types

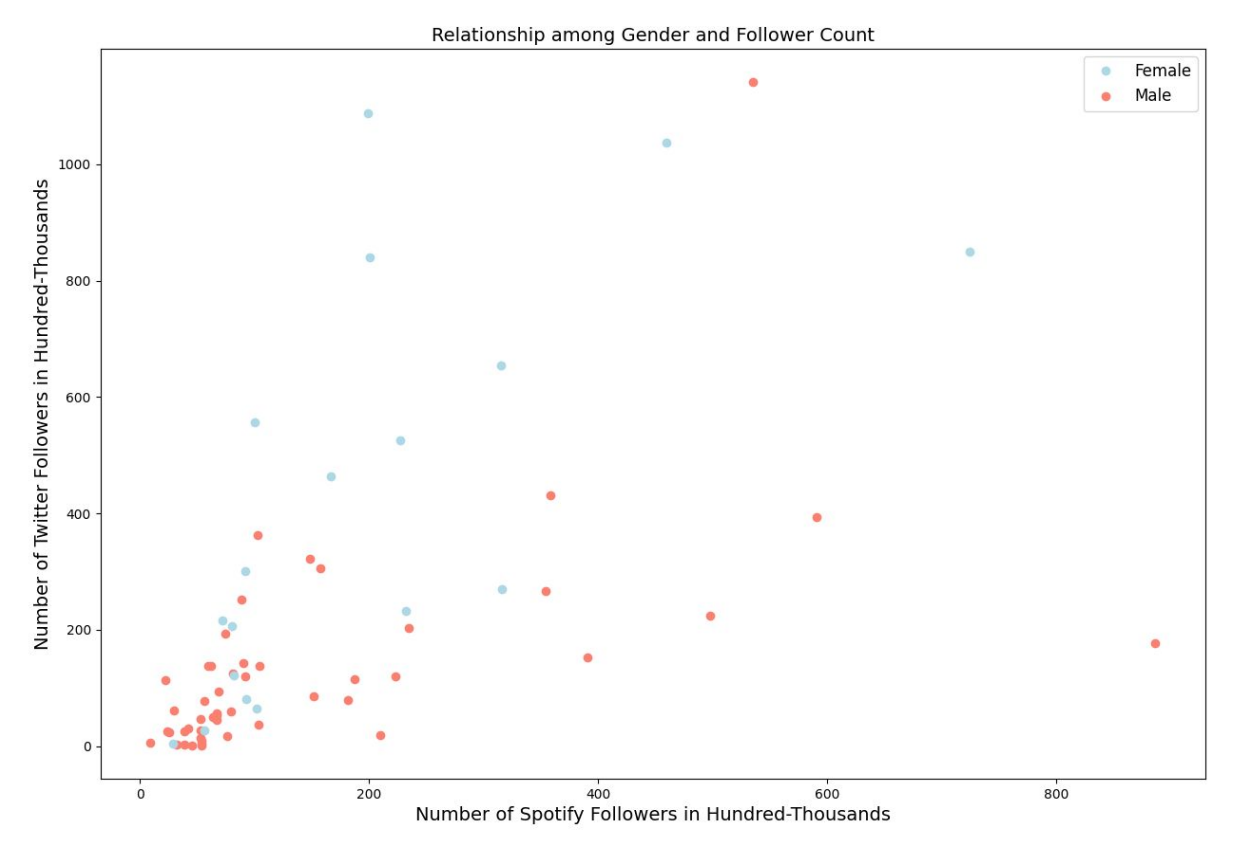


- Bar graph represents the average of net worth of female vs male celebrities

- Males appear to have a lower net worth. However, data is only collected from 97 artists, so we wonder if this is truly representative.



- Scatterplot of networth and number of fans for celebrities, categorized into male and female
- Those with higher net worth appear to have a higher number of fans
- Does not appear to be a significant difference between male and female artists, but data is taken from more males

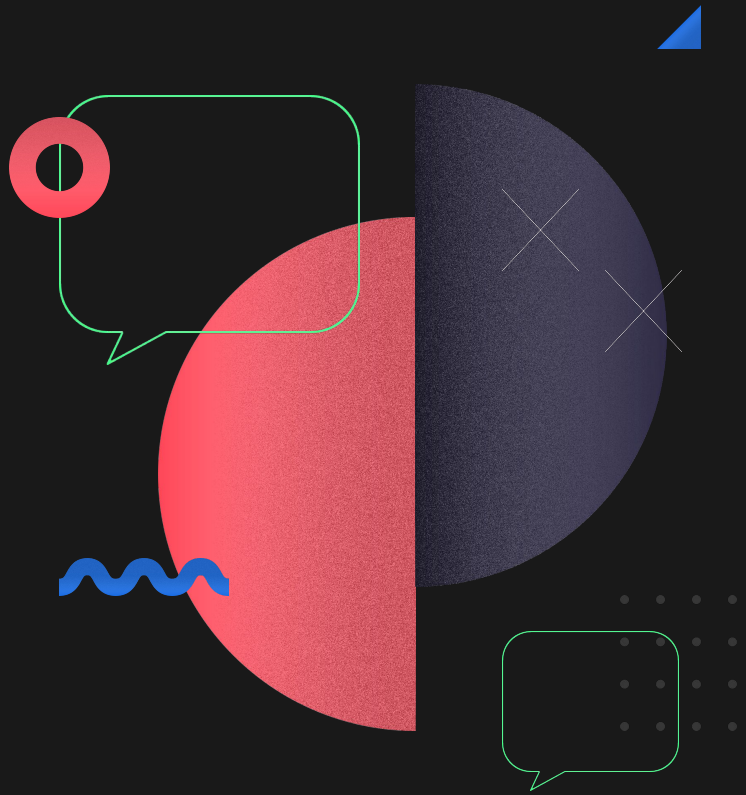


- Scatterplot to see if correlation between an artist's number of Spotify Followers and their corresponding Twitter followers distributed by gender

- Those with more Spotify followers also seem to have more Twitter followers. There is a moderately strong, positive increasing correlation.

- More female celebrities represent outliers, but hard to tell whether this is due to a reason or if our sample simply contained more female artists

TABLES



Artists

artist_id	name	fans_count
1	Rihanna	4,328,785
2	Drake	4,288,021
3	Coldplay	4,251,621
4	Eminem	4,202,731
5	Maroon 5	3,785,233
6	Ed Sheeran	3,592,631
7	Kanye West	3,566,946
8	Bruno Mars	3,561,948
9	U2	3,391,148
10	Adele	3,387,368
11	The Weeknd	3,317,123
12	Beyoncé	3,307,897
13	Justin Bieber	3,184,494
14	Katy Perry	3,092,396
15	Lil Wayne	3,043,928
16	Kendrick Lamar	3,035,636
17	Taylor Swift	3,024,415
18	Red Hot Chili Peppers	3,022,824
19	Calvin Harris	2,997,563
20	Nicki Minaj	2,991,397
21	Chris Brown	2,926,812
22	Imagine Dragons	2,914,040
23	Lady Gaga	2,881,099
24	David Guetta	2,722,256
25	Ariana Grande	2,590,414

NetWorth

id	name	net_worth	age	gender	nationality
1	Rihanna	550000000	33	female	bb
2	Drake	180000000	35	male	ca
4	Eminem	230000000	49	male	us
6	Ed Sheeran	200000000	30	male	gb
7	Kanye West	3200000000	44	male	us
8	Bruno Mars	175000000	36	male	us
10	Adele	190000000	33	female	gb
11	The Weeknd	100000000	31	male	ca
13	Justin Bieber	285000000	27	male	ca
14	Katy Perry	330000000	37	female	us
15	Lil Wayne	150000000	39	male	us
16	Kendrick Lamar	75000000	34	male	us
17	Taylor Swift	400000000	31	female	us
19	Calvin Harris	300000000	37	male	gb
20	Nicki Minaj	80000000	38	female	us
21	Chris Brown	50000000	32	male	us
23	Lady Gaga	320000000	35	female	us
24	David Guetta	85000000	54	male	fr
25	Ariana Grande	150000000	28	female	us

SpotifyArtistData

id	name	followers	genre_id
1	Rihanna	45978363	1
2	Drake	59075502	2
3	Coldplay	32895585	3
4	Eminem	49831592	4
5	Maroon 5	34063834	5
6	Ed Sheeran	88645737	5
7	Kanye West	15747836	6
8	Bruno Mars	35872525	7
9	U2	9052419	8
10	Adele	31607503	9
11	The Weeknd	39075301	10
12	Beyoncé	29593199	7
13	Justin Bieber	53563698	11
14	Katy Perry	19928018	7
15	Lil Wayne	11185683	12
16	Kendrick Lamar	18786582	13
17	Taylor Swift	46141722	5
18	Red Hot Chili Peppers	17128176	14
19	Calvin Harris	22317368	7
20	Nicki Minaj	23247012	7
21	Chris Brown	14853668	7
22	Imagine Dragons	36902239	15
23	Lady Gaga	20060736	16
24	David Guetta	23504442	17
25	Ariana Grande	72410286	7

SpotifyGenreData

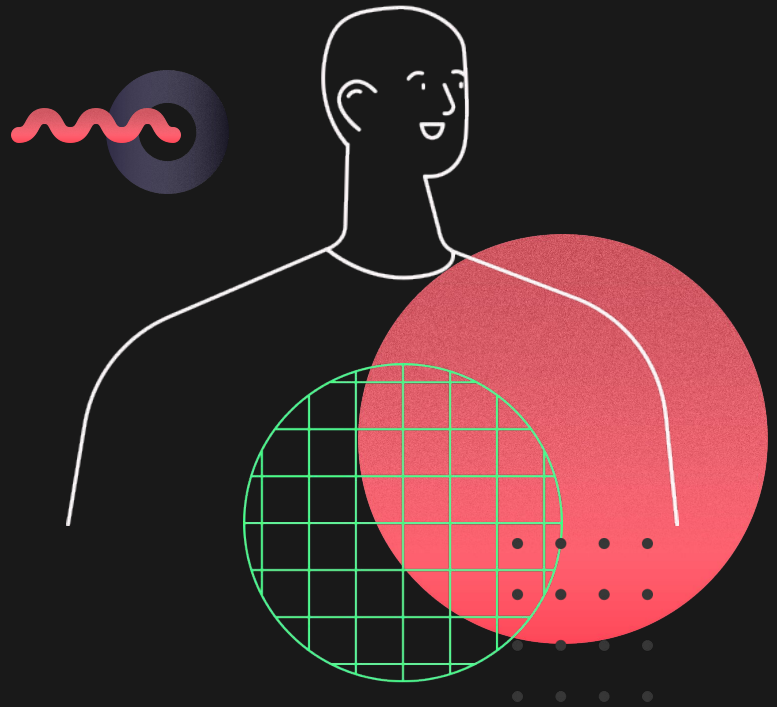
genre_id	genre
1	barbadian pop
2	canadian hip hop
3	permanent wave
4	detroit hip hop
5	pop
6	chicago rap
7	dance pop
8	irish rock
9	british soul
10	canadian contemporary r&b
11	canadian pop
12	hip hop
13	conscious hip hop
14	alternative rock
15	modern rock
16	art pop
17	big room
18	atl hip hop
19	classic rock
20	garage rock
21	modern folk rock
22	dfw rap
23	british invasion
24	australian dance
25	hip pop

- Shared key→ genre_id

TwitterData

id	name	screen_name	follower_count
1	Rihanna	rihanna	103706043
2	Drizzy	Drake	39369362
3	Coldplay	coldplay	23537346
4	Marshall Mathers	Eminem	22518439
5	Maroon 5	maroon5	14258529
6	Ed Sheeran HQ	edsheeran	17675434
7	ye	kanyewest	30623236
8	Bruno Mars	BrunoMars	43172959
9	U2	U2	1591868
10	Adele	Adele	27004587
11	The Weeknd	theweeknd	15277855
13	Justin Bieber	justinbieber	114158582
14	KATY PERRY	katyperry	108777260
16	Kendrick Lamar	kendricklamar	11580537
19	Calvin Harris	CalvinHarris	11950320
20	Nicki Minaj	NICKIMINAJ	23241542
21	Chris Brown	chrisbrown	32149399
22	Imagine Dragons	Imaginedragons	5681263
23	Lady Gaga	ladygaga	83952770
24	David Guetta	davidguetta	20276193
25	Ariana Grande	ArianaGrande	85067061

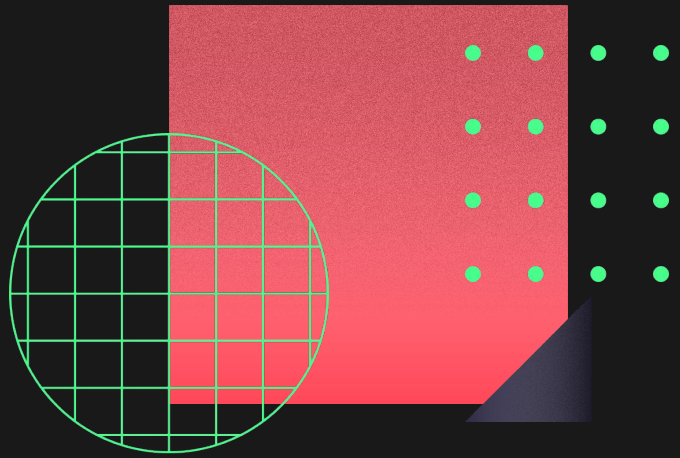
INSTRUCTIONS



Steps:

1. **get_artists_names.py** must be rune first to get the list of names which will be used later in other Python files
2. Then, can ruge **get_net_worth.py**, **get_spotify_info.py**, and **get_twitter_data.py** in any order. The order does not matter.
3. **visulization.py** must be ran last since it needs to use all of the data that has been processed in the previous files

DOCUMENTATION



get_artist_names.py

```python
def createSoup():
    url = "https://www.songkick.com/leaderboards/popular_artists"
    r = requests.get(url)
    soup = BeautifulSoup(r.content, "html.parser")
    return soup
```

createSoup() returns soup based on the url

```python
def createSoupFromFile():
    # in case the website changes or denied web scraping
    dir_path = os.path.dirname(os.path.realpath(__file__))
    path = dir_path + '/backup_website' + 'Most_Popular_Artists_Worldwide.html'
    with open(path) as pg:
        soup = BeautifulSoup(pg, "html.parser")
    return soup
```

createSoupFromFile() returns soup based on the html file (serves as a backup option if the actual website fails)

get_artist_names.py

```python
def getLastID(cur):
    # create the table if not exist
    cur.execute('CREATE TABLE IF NOT EXISTS Artists (artist_id INTEGER PRIMARY KEY, name TEXT, fans_count INTEGER)')
    # checking what have already been stored to the database
    # get the last row of the database
    # change the table name if expecting to get the id from other tables
    cur.execute('SELECT artist_id FROM Artists ORDER BY artist_id DESC LIMIT 1')
    row = cur.fetchone()
    # check if this is the first time inserting any data
    if row:
        # get the id of the last inserted item
        count = row[0]
        if count == 200: print('All 200 items already inserted')
    else:
        print('first time inserting the data')
        count = 0 # count for id
    return count
```

getLastID(cur) returns the id from the last row of the table

```python
def getData(soup, cur, conn):
    names = []
    fans = []
    start = getLastID(cur)

    table = soup.find('table')
    # check if the data from website is none
    if table != None:
        rows = table.find_all("tr")
        # scraping only 25 elements from the website
        for r in rows[start + 1: start + 21]:
            name_cols = r.find('td', class_ = "name")
            fan = r.find('td', class_ = 'count').text.split()[2]
            names.append(name_cols.text.strip('\n'))
            fans.append(fan)

    # create the artist database
    createArtistDatabase(cur, conn, names, fans, start)
```

getData(soup, cur, conn) returns nothing, but calls the CreateArtistDatabase function

get_artist_names.py

```
def createArtistDatabase(cur, conn, names, fans, start):
    # iteraing through the scraped data and insert them into the database
    for i in range(0, len(names)):
        cur.execute('INSERT OR IGNORE INTO Artists (artist_id, name, fans_count) VALUES (?, ?, ?)', \
            (start+1, names[i], fans[i]))
        start += 1
    conn.commit()
```

createArtistDatabase(cur, conn, names, fans, start) iterates through the data and inserts it into the database. Names and fans are lists, start is the id return by **getLastID**.

```
def main():
    # create the soup
    s = createSoup()
    # if cannot scrape from the website, then scrape it form the file
    # s = createSoupFromFile()

    # create the database
    dir_path = os.path.dirname(os.path.realpath(__file__))
    conn = sqlite3.connect(dir_path + '/' + "finalproject.db")
    cur = conn.cursor()

    # get the needed data from the website and create the artists database
    getData(s, cur, conn)
```

main() calls all necessary functions to get the 20 names from the website. Execute this 10 times to get all 200 artist names.

get_net_worth.py

```python
def getKey():
    # get my API key from file
    dir_path = os.path.dirname(os.path.realpath(__file__))
    path = dir_path + '/' + 'API_keys.txt'
    f = open(path, 'r')
    t = f.readline()
    f.close()
    return t
```

getKey() returns the personal key from the API_keys.txt file

```python
def getLastID(cur):
    # first create the table
    cur.execute('CREATE TABLE IF NOT EXISTS NetWorth (id INTEGER PRIMARY KEY, name TEXT, net_worth NUMBER, age INTEGER, gender TEXT, nationality TEXT)')
    # checking what have already been stored to the database
    # get the last row of the database
    # change the table name if expecting to get the id from other tables
    cur.execute('SELECT id FROM NetWorth ORDER BY id DESC LIMIT 1')
    row = cur.fetchone()
    # check if this is the first time inserting any data
    if row:
        # get the id of the last inserted item
        count = row[0]
        if count == 200: print('All 200 items already inserted')
    else:
        print('first time inserting the data')
        count = 0 # count for id
    return count
```

getLastID(cur) returns the id from the last row of a table

get_net_worth.py

```
def getNameList(cur):
    id = getLastID(cur)
    cur.execute("SELECT artist_id, name FROM Artists WHERE artist_id > ? AND artist_id <= ?", (id, id+20))
    rows = cur.fetchall()
    names = []
    # craete a list of tuples with id as the first element and name as the second element
    for r in rows:
        names.append(tuple([r[0], r[1]]))
    return names
```

getNameList(cur) returns a list of tuples in the format of [(id,name)]

get_net_worth.py

```python
def getNetWorth(names, cur, conn):
    # get the artist infortion from the celebrity API
    key = getKey()
    networth = []
    for n in names:
        # get the api request url
        api_url = 'https://api.api-ninjas.com/v1/celebrity?name={}'.format(n[1].lower())
        r = requests.get(api_url, headers={'X-Api-Key': key} )
        # check if the request succeeded
        if r.status_code == requests.codes.ok:
            d = json.loads(r.text)
            # check if the returned celebrity information is really what we are look for
            for dict in d:
                if dict['name'].lower() == n[1].lower():
                    # create a list of tuples with id, name, networth, age, and gender
                    networth.append(tuple([n[0], n[1], dict.get('net_worth', -1), dict.get('age', -1), dict.get('gender', 'NA'), dict.get('nationality', 'NA')]))
                    break
        else:
            print("Error:", r.status_code, r.text)

    # store the data into the table
    insertIntoDatabase(networth, cur, conn)
```

getNetWorth(names, cur, conn) returns nothing, but calls the insertIntoDatabase function

get_net_worth.py

```
def insertIntoDatabase(networth, cur, conn):
    # iterate through the list and add them to the table with the id that is the same as the artist_id in the Artists table
    for t in networth:
        cur.execute('INSERT OR IGNORE INTO NetWorth (id, name, net_worth, age, gender, nationality) VALUES (?, ?, ?, ?, ?, ?)', \
                (t[0], t[1], t[2], t[3], t[4], t[5]) )
    conn.commit()
```

insertIntoDatabase(networth, cur, conn) inserts the data into the NetWorth table

```
def main():
    # connect to the database
    dir_path = os.path.dirname(os.path.realpath(__file__))
    conn = sqlite3.connect(dir_path + '/' + "finalproject.db")
    cur = conn.cursor()
    # get the data and insert it into the database
    names = getNameList(cur)
    getNetWorth(names, cur, conn)
```

main() can be called 10 times to return all 200 items of data

get_twitter_data.py

```python
def getId(cur):
    cur.execute('CREATE TABLE IF NOT EXISTS TwitterData (id INTEGER PRIMARY KEY, name TEXT, screen_name TEXT, follower_count INTEGER)')
    cur.execute('SELECT id FROM TwitterData ORDER BY id DESC LIMIT 1')
    row = cur.fetchone()
    # check if this is the first time inserting any data
    if row:
        # get the id of the last inserted item
        count = row[0]
        if count == 200: print('All 200 items already inserted')
    else:
        print('first time inserting the data')
        count = 0 # count for id
    return count
```

getID(cur) gets the API keys for Twitter

```python
def getNameList(cur):
    id = getId(cur)
    cur.execute("SELECT artist_id, name FROM Artists WHERE artist_id > ? AND artist_id <= ?", (id, id+20))
    rows = cur.fetchall()
    names = []
    # craete a list of tuples with id as the first element and name as the second element
    for r in rows:
        names.append(tuple([r[0], r[1]]))
    return names
```

getNameList(cur) returns a list of Twitter names

get_twitter_data.py

```python
def getData (name, cur, conn):
    user_name = None
    screen_name = None
    follower_count = None

    try:
        new_name = name[1].lower().replace(" ", "")
        cursor = tweepy.Cursor(api.user_timeline, id = new_name, tweet_mode = 'extended').items(1)
        print(new_name)

        for i in cursor:
            # print(i)
            verified = i.author.verified
            # print(verified)
            user_name = i.author.name
            screen_name = i.author.screen_name
            follower_count = i.author.followers_count
            tweet_count = i.author.statuses_count
    except:
        print("Cannot find data for this artist")


    if user_name != None and verified == True:
        print(follower_count)

        cur.execute('INSERT OR IGNORE INTO TwitterData (id, name, screen_name, follower_count) VALUES (?,?,?,?)', (name[0], user_name, screen_name, follower_count))
        conn.commit()
```

getData(name, cur, conn) adds corresponding Twitter followers to the artist's names

```python
def main():
    # connect to the database
    dir_path = os.path.dirname(os.path.realpath(__file__))
    conn = sqlite3.connect(dir_path + '/' + "finalproject.db")
    cur = conn.cursor()
    name_list = getNameList(cur)
    # getData((23, 'ladygaga'), cur, conn)
    for t in name_list:
        getData(t, cur, conn)
```

main() runs the entire function and returns a database with twitter screen names, artist_ids, and the followers

get_spotify_info.py

```
def getLastID(cur, tablename, idname):
    # createa the table
    cur.execute('CREATE TABLE IF NOT EXISTS SpotifyArtistData (id INTEGER PRIMARY KEY, name TEXT, followers INTEGER, genre_id INTEGER)')
    # create another table with shared key
    cur.execute('CREATE TABLE IF NOT EXISTS SpotifyGenreData (genre_id INTEGER PRIMARY KEY, genre TEXT)')
    # checking what have already been stored to the database
    # get the last row of the database
    # change the table name if expecting to get the id from other tables
    cur.execute('SELECT {} FROM {} ORDER BY {} DESC LIMIT 1'.format(idname, tablename, idname))
    row = cur.fetchone()
    # check if this is the first time inserting any data
    if row:
        # get the id of the last inserted item
        count = row[0]
        if count == 200: print('All 200 items already inserted')
    else:
        print('first time inserting the data')
        count = 0 # count for id
    return count
```

getLastID(cur, tablename, idname) creates the SpotifyArtistData table as well as the SpotifgyGenreData table (which have a shared key of genre_id) and returns the id from the last row of the table

```
def getListOfArtists(cur):
    id = getLastID(cur, 'SpotifyArtistData', 'id')
    cur.execute("SELECT artist_id, name FROM Artists WHERE artist_id > ? AND artist_id <= ?", (id, id+20))
    rows = cur.fetchall()
    names = []
    # craete a list of tuples with id as the first element and name as the second element
    for r in rows:
        names.append(tuple([r[0], r[1]]))
    #print(names)
    return names
```

getListofArtists(cur) returns a list of tuples in the format of [(id, artist name)]

get_spotify_info.py

```python
def getArtistInfo(names, cur, conn):
    artist_info= []
    # get the api request url
    for n in names:
        api_url = 'https://api.spotify.com/v1/search?q={}&type=artist&offset=0&limit=1'.format(n[1].lower())
        headers = {'Authorization': 'Bearer {}'.format(access_token)}
        response = requests.get(api_url,headers=headers)
        #check if request was successful
        if response.status_code == requests.codes.ok:
            d = json.loads(response.text)
            artist_info.append(tuple([n[0], n[1], d['artists']['items'][0]['followers']['total'], d['artists']['items'][0]['genres'][0]]))
        else:
            print("Error:", response.status_code, response.text)
    #print(artist_info)
    # store the data into the table
    insertIntoDatabase(artist_info, cur, conn)
```

getArtistInfo(names, cur, conn) returns nothing and calls insertIntoDatabase

```python
def insertIntoDatabase(artist_info, cur, conn):
    for t in artist_info:
        # iterate through the list and add them to the table with the id that is the same as the artist_id in the Artists table
        cur.execute('SELECT genre_id FROM SpotifyGenreData WHERE genre = ?', (t[-1],))
        row = cur.fetchone()
        # check if this is the first time inserting any data
        if row:
            id = row[0]
        else:
            id = getLastID(cur, 'SpotifyGenreData', 'genre_id') + 1
        print(id)

        cur.execute('INSERT OR IGNORE INTO SpotifyArtistData (id, name, followers, genre_id) VALUES (?, ?, ?, ?)', \
                (t[0], t[1], t[2], id) )
        cur.execute('INSERT OR IGNORE INTO SpotifyGenreData (genre_id, genre) VALUES (?, ?)', \
                (id, t[3]) )
    conn.commit()
```

insertIntoDatabase(artist_info, cur, conn) inserts data into the SpotifyArtistData table and the SpotifyGenreData table

get_spotify_info.py

```
def main():
    # connect to the database
    dir_path = os.path.dirname(os.path.realpath(__file__))
    conn = sqlite3.connect(dir_path + '/' + "finalproject.db")
    cur = conn.cursor()
    artists = getListOfArtists(cur)
    getArtistInfo(artists, cur , conn)
```

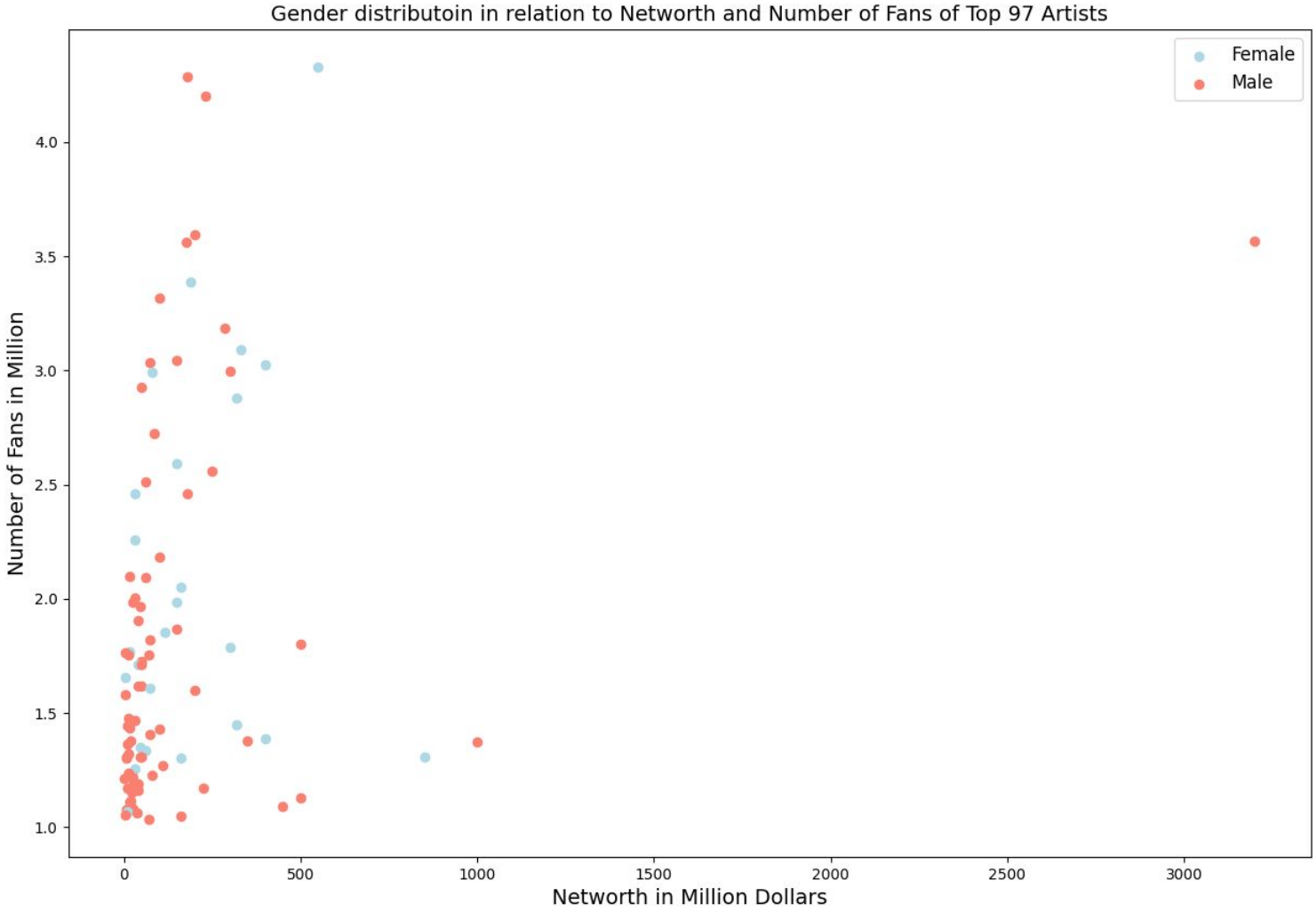
main() can be called 10 times to get all 200 items of data

visualization.py

```
def genderScatterPlot(cur, conn):
    # get the data from NetWorth and Artists
    cur.execute("SELECT Artists.name, NetWorth.gender, Artists.fans_count, NetWorth.net_worth FROM NetWorth JOIN Artists ON \
        Artists.artist_id = NetWorth.id WHERE gender != 'NA'")
    data = cur.fetchall()
    colors = ['salmon', 'lightblue']
    fig = plt.figure(figsize = (15, 10))
    for d in data:
        g = 1 if d[1] == 'female' else 0
        x = d[3]/1000000
        y = int(d[2].replace(',', ''))/1000000
        # begin plotting the data
        plt.scatter(x, y, label = d[0], color = colors[g])
        # plt.annotate(text=d[0], xy = (x+0.004, y+0.001), fontsize = 8)

    # customizing a little
    plt.xlabel('Networth in Million Dollars', fontsize=14)
    plt.ylabel('Number of Fans in Million', fontsize=14)
    plt.title('Gender distributoin in relation to Networth and Number of Fans of Top {} Artists'.format(len(data)), fontsize=14)
    plt.legend(['Female', 'Male'], fontsize = 12)
    # save the plot
    plt.savefig(dir_path + '/visualizations/gender_Scatter_Plot.png')
    plt.show()
```

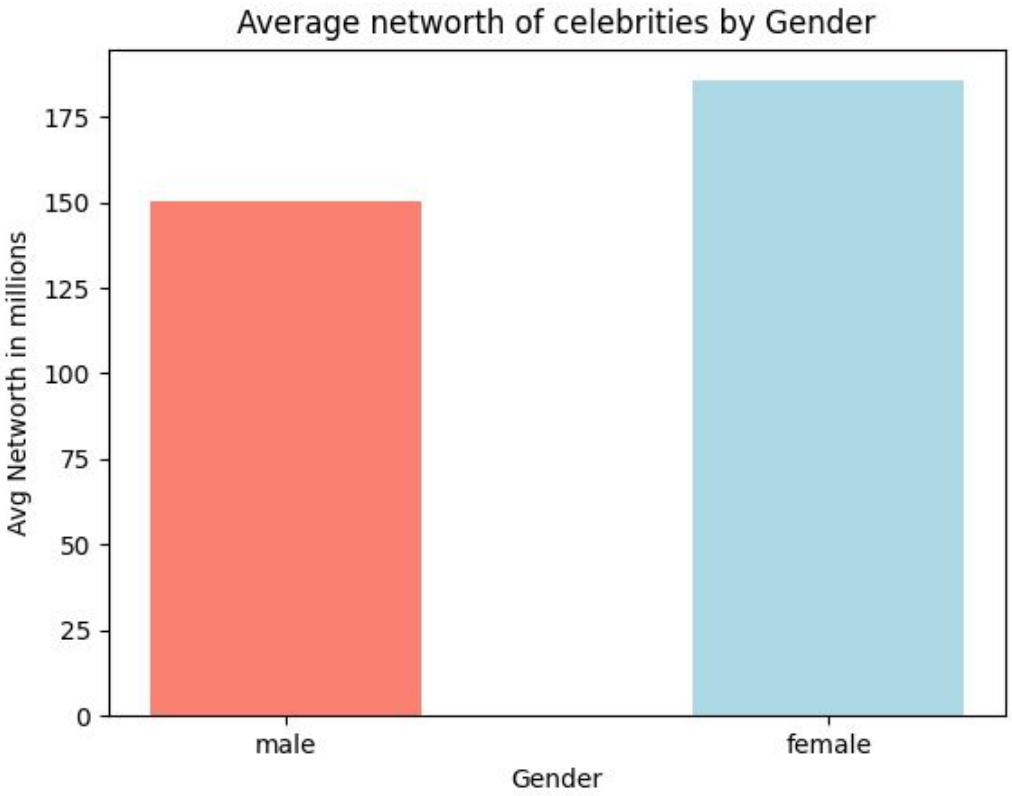
genderScatterPlot(cur, conn) creates the scatterplot illustrating the correlation between an artist's net worth and number of fans, by gender


Gender distributoin in relation to Networth and Number of Fans of Top 97 Artists
Female
Male
Number of Fans in Million
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
500
1000
1500
2000
2500
3000
Networth in Million Dollars

visualization.py

```python
def getAvgNetworthByGender(cur):
    # extracting the data
    cur.execute("SELECT net_worth FROM NetWorth WHERE gender = 'male'")
    male_networth = [x[0] for x in cur.fetchall()]
    cur.execute("SELECT net_worth FROM NetWorth WHERE gender = 'female'")
    female_networth = [x[0] for x in cur.fetchall()]
    # calculate the averages
    avg_m = sum(male_networth) / len(male_networth) / 1000000
    avg_f = sum(female_networth) / len(female_networth) / 1000000
    # save the data to a file
    f = open(dir_path + '/calculation_files/avg_networth_by_gender.txt', 'w')
    f.write(f'average networth of male celebrities: {avg_m}, average networth of female celebrities: {avg_f}')
    f.close()
    # plot the data
    plt.bar(['male', 'female'], [avg_m, avg_f], color = ['salmon', 'lightblue'], width = 0.5)
    plt.title('Average networth of celebrities by Gender')
    plt.xlabel("Gender")
    plt.ylabel('Avg Networth in millions')
    # save the plots
    plt.savefig(dir_path + '/visualizations/avg_NetWorth_By_Gender.png')
    plt.show()
```

getAvgNetworthByGender(cur) returns the bar graph illustrating average net worth of celebrities by gender

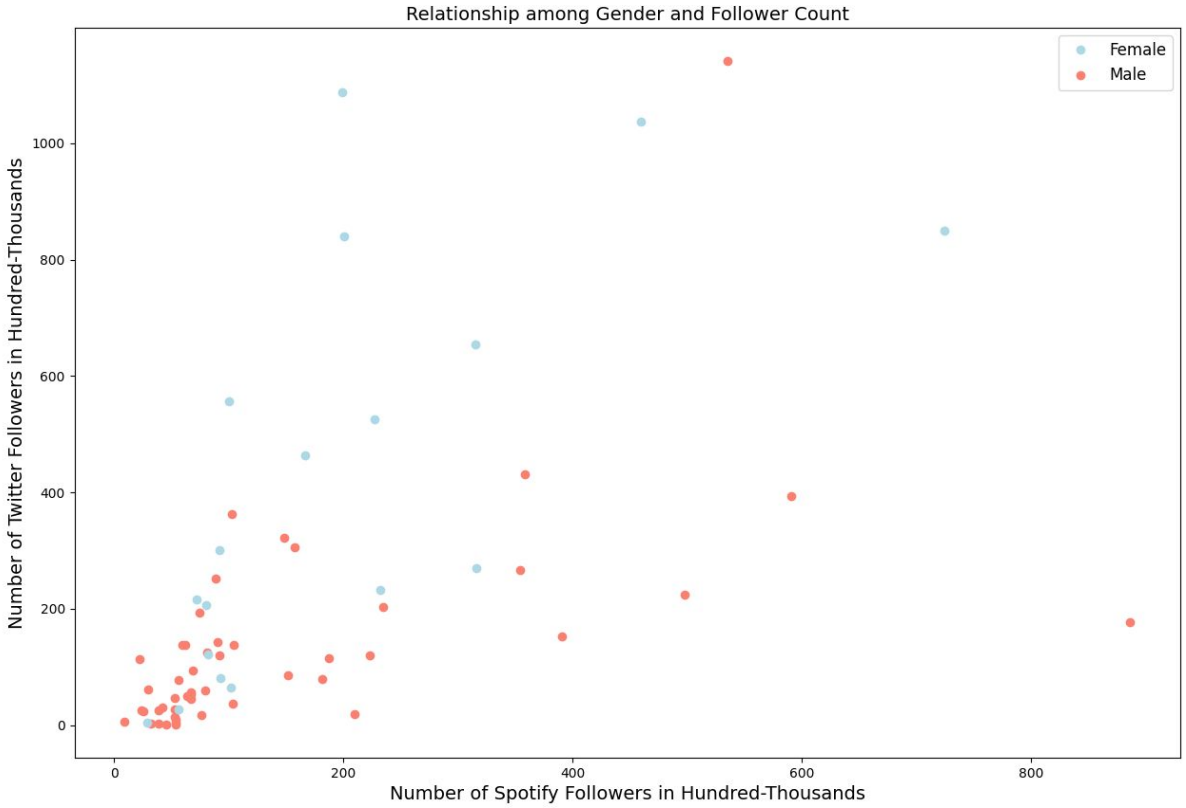

Average networth of celebrities by Gender
Avg Networth in millions
0
25
50
75
100
125
150
175
male
female
Gender

visualization.py

```
def genderFollowersScatterPlot(cur, conn):
    dir_path = os.path.dirname(os.path.realpath(__file__))
    # get the data from Spotify and NetWorth
    cur.execute("SELECT SpotifyArtistData.name, NetWorth.gender, SpotifyArtistData.followers, TwitterData.follower_count FROM NetWorth JOIN SpotifyArtistData /
                ON SpotifyArtistData.id = NetWorth.id JOIN TwitterData ON TwitterData.id = NetWorth.id WHERE gender != 'NA'")
    data = cur.fetchall()
    colors = ['salmon', 'lightblue']
    fig = plt.figure(figsize = (15, 10))
    for d in data:
        g = 1 if d[1] == 'female' else 0
        x = d[2]/100000
        y = d[3]/100000
        # begin plotting the data
        plt.plot(x, y, 'o', color = colors[g])
    plt.xlabel('Number of Spotify Followers in Hundred-Thousands', fontsize=14)
    plt.ylabel('Number of Twitter Followers in Hundred-Thousands', fontsize=14)
    plt.title('Relationship among Gender and Follower Count', fontsize=14)
    plt.legend(['Female', 'Male'], fontsize = 12)

    plt.savefig(dir_path + '/visualizations/gender_Followers_ScatterPlot.png')
    plt.show()
```

genderFollowersScatterPlot(cur, conn) returns the scatterplot illustrating the correlation between an artist's Spotify followers and Twitter followers, by gender


Relationship among Gender and Follower Count
Female
Male
Number of Twitter Followers in Hundred-Thousands
Number of Spotify Followers in Hundred-Thousands
0
200
400
600
800
1000
0
200
400
600
800

visualization.py

```
def createGenrelist(cur, conn):
    cur.execute('SELECT genre_id FROM SpotifyGenreData')
    conn.commit()
    data = cur.fetchall()
    genre_list = []
    genre_list.append(data)
    return genre_list
```

createGenrelist(cur, conn) returns a list of genre_ids to iterate through

```
def createGenrelistname(cur, conn):
    cur.execute('SELECT genre FROM SpotifyGenreData')
    conn.commit()
    data = [x[0] for x in cur.fetchall()]
    return data
```

createGenrelistname(cur, conn) returns a list of genre ID names to later be used on the x-axis

visualization.py

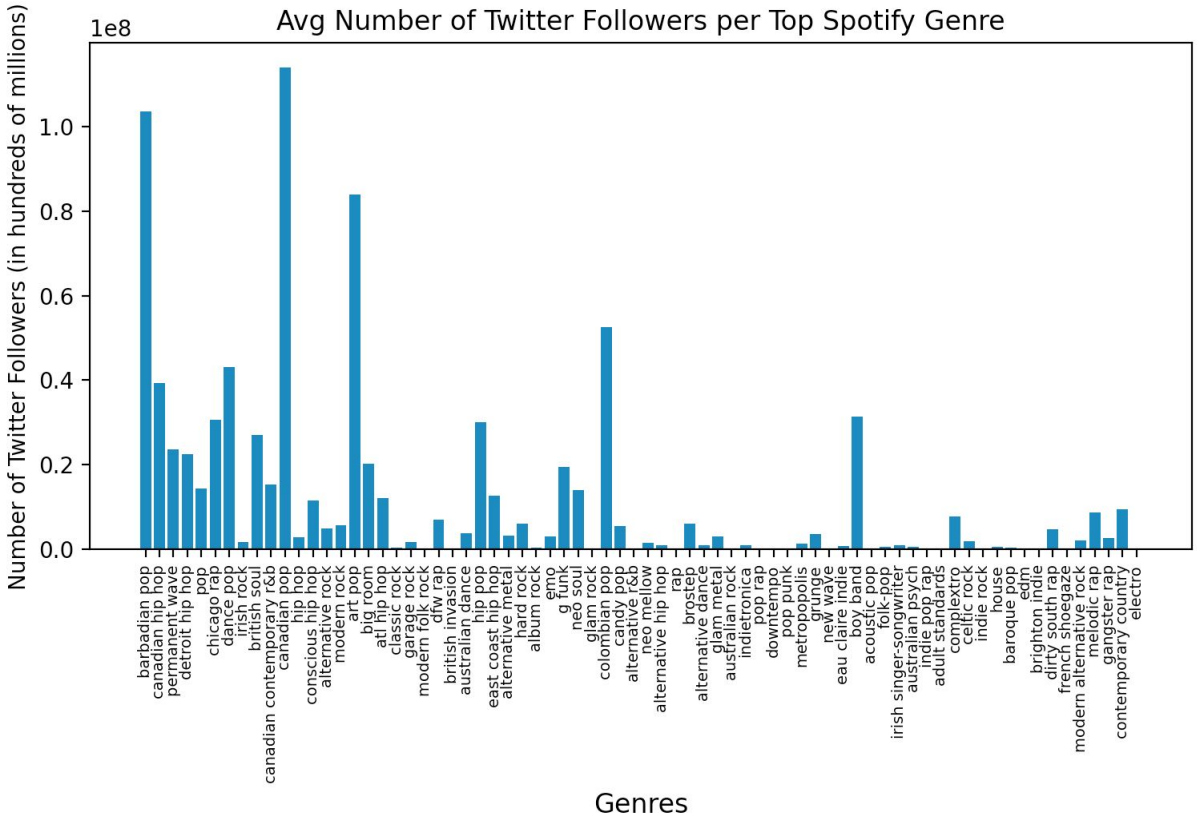
```python
def getTwitterFollowersByGenre(cur, conn):
    #Calling my function to get the list of genre id's
    genre_id_list = createGenrelist(cur, conn)
    #Creating an empty list to add the average followers to
    avg_list = []
    #Calling the function to get the list of genre id names
    genre_name_list = createGenrelistname(cur, conn)

    # Iterating through the genre_ids to get my follower count data, which is the first item in a tupple.
    for i in genre_id_list[0]:
        cur.execute('SELECT follower_count FROM TwitterData JOIN SpotifyArtistData ON TwitterData.id = SpotifyArtistData.id WHERE SpotifyArtistData.genre_id=?', (i[0],)
        conn.commit()
        rows = cur.fetchall()
        #Creating empty follower list and adding the artists' followers to that for.
        follower_list = []
        follower_list.append(rows)
        count = 0
        artists = 0
        for num in follower_list:
            #Doing this to avoid the divide by zero or any index error, because some genres have 0 twitter followers averages.
            if len(num) == 0:
                pass
            else:
                count += int(num[0][0])
        #Getting the length of follower list to divide by and get the averages
        artists = len(follower_list)
        average_num = (int(count) / int(artists))
         avg_list.append(average_num)

         # save the data to a file
    f = open(dir_path + '/calculation_files/avg_twitter_followers_by_genre.txt', 'w')
    f.write('1. genre, 2. average twitter followers'+ '\n')
    for i in range(len(avg_list)):
        f.write( f'{genre_name_list[i]}, {avg_list[i]}' + '\n')
    f.close()

    #Creating my plot, passing in genre_name_list and avg_list for my axes.
    fig = plt.figure(figsize = (15, 10))
    plt.bar(genre_name_list, avg_list)
    plt.xlabel('Genres', fontsize = 12)
    #Setting accurate labels because there is such a wide range of data, that it has to be (in hundres of millions) for it to all fit into on
    plt.ylabel('Number of Twitter Followers (in hundreds of millions)', fontsize = 10)
    plt.title('Avg Number of Twitter Followers per Top Spotify Genre')
    #Rotating my x ticks so they do not get overlapped
    plt.xticks(rotation = 90, fontsize = 7)
    plt.savefig(dir_path + '/visualizations/Avg_Twitter_Followers_Per_Spotify_Genre.png')
    plt.show()
```

getTwitterFollowerscur, conn) returns a bar graph of average Twitter followers by genre


Avg Number of Twitter Followers per Top Spotify Genre
1e8
Number of Twitter Followers (in hundreds of millions)
1.0
0.8
0.6
0.4
0.2
0.0
barbadian pop
canadian hip hop
permanent wave
detroit hip hop
pop
chicago rap
dance pop
irish rock
british soul
canadian contemporary r&b
canadian pop
hip hop
conscious hip hop
alternative rock
modern rock
art pop
bin r pop
atl hip hop
classic rock
garage rock
modern folk rap
british invasion
australian dance
hip hop
east coast hip hop
alternative metal
hard rock
album rock
emo
g funk
neo soul
glam rock
colombian pop
candy pop
alternative r&b
neo mellow
alternative hip hop
rap
brostep
alternative dance
glam metal
australian rock
indietronica
pop rap
downtempo
pop punk
metropopolis
grunge
new wave
eau claire indie
boy band
acoustic pop
folk-pop
irish singer-songwriter
australian psych
indie pop rap
adult standards
complextro
celtic rock
indie rock
house
baroque pop
edm
brighton indie
dirty south rap
french shoegaze
modern alternative rock
melodic rap
gangster rap
contemporary country
electro
Genres

visualization.py

```python
def main():
    conn = sqlite3.connect(dir_path + '/' + "finalproject.db")
    cur = conn.cursor()
    # first visualization
    genderScatterPlot(cur, conn)
    # second visualization
    getAvgNetworthByGender(cur)
    # third visualization
    genderFollowersScatterPlot(cur, conn)
    # fourth visualization
    getTwitterFollowersByGenre(cur, conn)
```

main() takes nothing and returns nothing. establishes the database connection, calls the listed functions, and closes the database connection

RESOURCES

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
11.27	Need to get the id of the last row to determine what next 20 items to scrape/get from website/API	https://stackoverflow.com/questions/4073923/select-last-row-in-mysql	Solved
12.2	Need to remove all comma from the fans_count which is a string	https://www.journaldev.com/23763/python-remove-spaces-from-string	Solved.
12.2	Need to annotate the points in scatter plot with artist names	https://www.geeksforgeeks.org/how-to-annotate-matplotlib-scatter-plots/	Solved
12.6	Cannot pull the files from the remote git repo	https://stackoverflow.com/questions/26281767/run-command-git-branch-set-upstream-to-produce-error/42305655	Need to set upstream first then git pull. Solved.
12.6	Change bar plot to have different colors of bars	https://www.python-graph-gallery.com/3-control-color-of-barplots	Solved