J. Wilton
Mykiah Dupree
https://github.com/Wilton16/JMProject

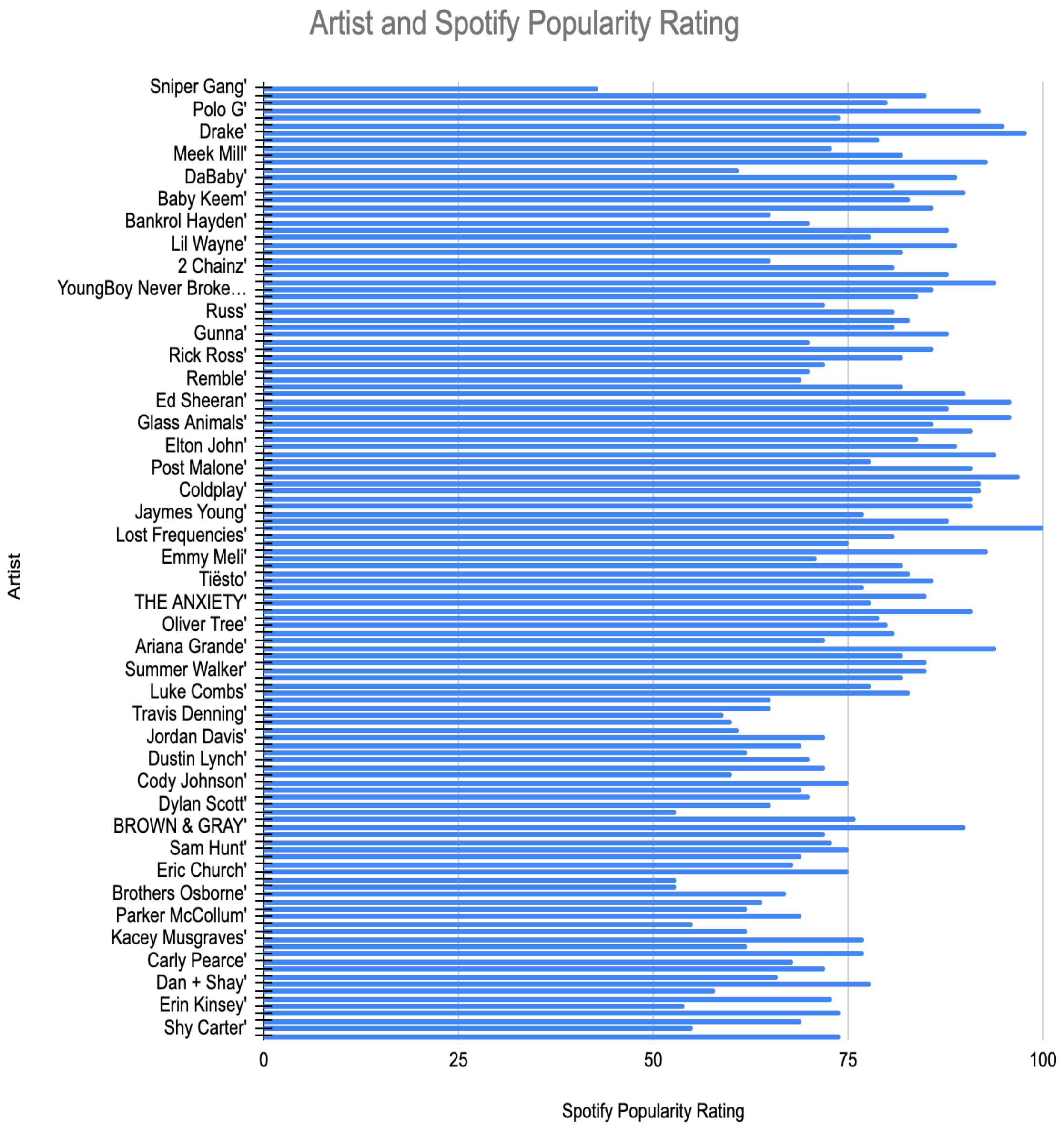
1) **Goals**
    a) Our main goal of the project was to see how an artists' popularity on Spotify corresponds with the amount of followers they have on social media, specifically Twitter.
2) **Goals that were achieved**
    a) We were able to successfully pull and store a group of artist's Popularity from Spotify and their amount of followers on Twitter. Then, we were able to calculate an aggregate ratio of the Spotify Popularity for artists to the Twitter Followers of artists, i.e. we averaged the ratios of all of the artists' data.
3) **Problems faced**
    a) Had to shift away from using the instagram api because there were too many problems we were experiencing.
    b) There were also alot of environmental and authentication errors.
    c) We weren't able to effectively limit database insertion to 25 at a time. We got close, but it was messing up our data.
    d) Some artists don't use Twitter, so some of the accounts that popped up were either fake accounts or irrelevant. That said, they were still included because officially we had no way to parse them (a "verified account" parameter would be great for the Twitter API for the purposes of this project).
    e) Time, heavy workloads, the holidays, and other projects and homeworks.
    f) Ensuring that the right data was being inserted into the functions in the right ways. Specifically, when using lists as arguments the way the code was set up kept parsing the argument indexes as strings, so instead of using something like "Argument1" and continuing in the code, it would slice it to say "A". This was resolved though.
    g) I had to quarantine because a "Close Contact" tested positive for Covid, and because I live on Campus and need the dining halls, I had to go back home the weekend we were trying to finish this up:/.
4) **File that contains calculations from data in database**
    a) "Writtenfile" - Below is a Copy of it.
    Here is a combination of spotify popularities and twitter followings for selected artists.
    [(43, 2854), (85, 2938), (80, 6518), (92, 2712809), (74, 476386), (95, 1792260), (98, 39395287), (79, 3576), (73, 747), (82, 11358), (93, 11444229), (61, 2768), (89, 4403303), (81, 1102164), (90, 5745725), (83, 186916), (86, 772031), (65, 697), (70, 11142), (88, 2195120), (78, 404665), (89, 34857801), (82, 2208283), (65, 23893), (81, 4465088), (88, 5798206), (94, 30628600), (86, 344), (84, 370550), (72, 412031), (81, 2207556), (83, 14289083), (81, 5484993), (88, 2039749), (70, 409056), (86, 7295210), (82, 4675301), (72, 155871), (70, 1572781), (69, 6829), (82, 21256), (90, 55137), (96, 40826), (88, 5642607), (96,

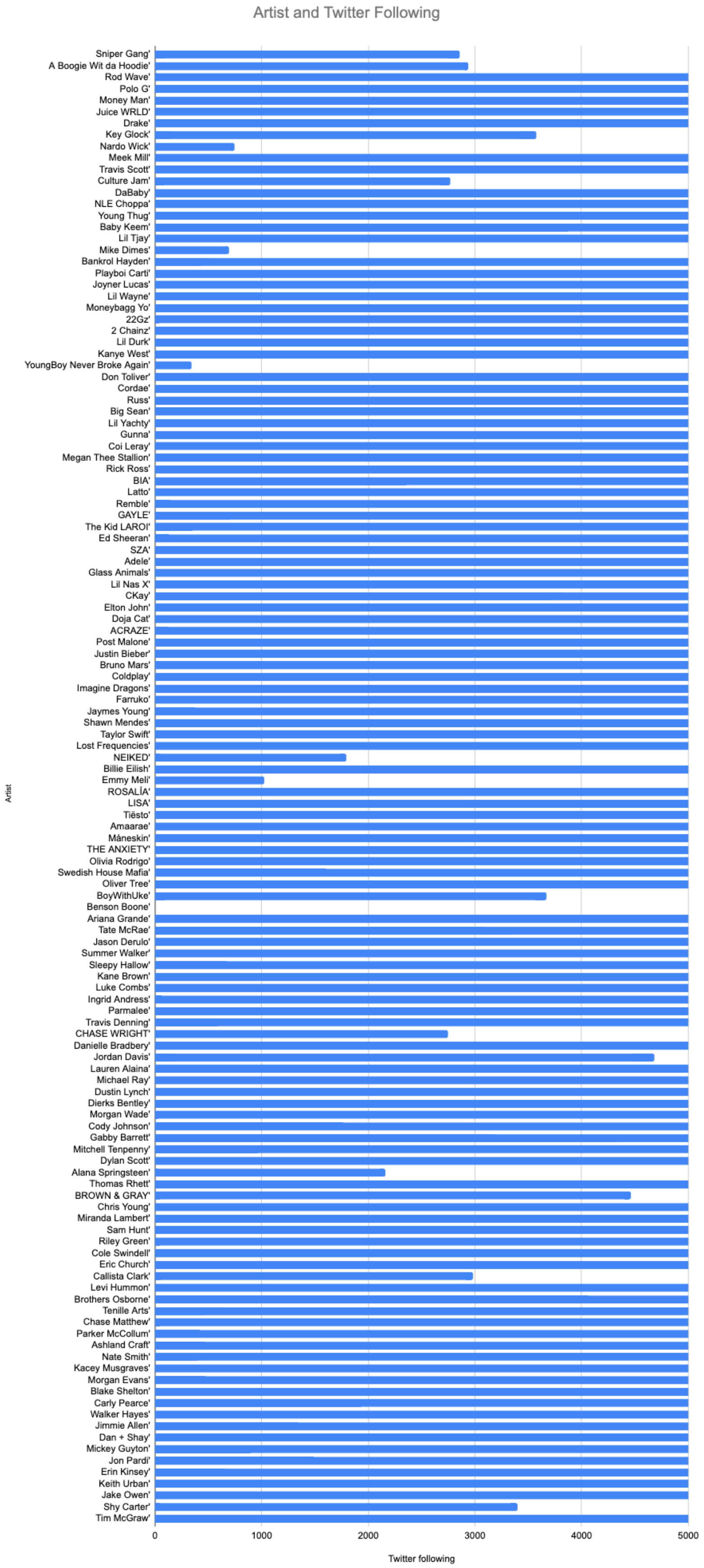
```
27054943), (86, 166944), (91, 7572216), (84, 90725), (89, 1078807), (94,
4007823), (78, 5347), (91, 6938584), (97, 114217412), (92, 43175713), (92,
23552363), (91, 5680957), (91, 4971606), (77, 22883), (88, 26731745), (100,
89654181), (81, 58246), (75, 1795), (93, 6456148), (71, 1023), (82, 2866458),
(83, 1598639), (86, 4794658), (77, 52807), (85, 452526), (78, 141012), (91,
1616256), (79, 2007622), (80, 233716), (81, 3672), (72, 1), (94, 85185953), (82,
77743), (85, 3740798), (85, 1537364), (82, 30826), (78, 1754995), (83, 888840),
(65, 17556), (65, 166991), (59, 24733), (60, 2747), (61, 349706), (72, 4683), (69,
668103), (62, 95506), (70, 884521), (72, 2593068), (60, 10506), (75, 276844),
(69, 91092), (70, 45723), (65, 57612), (53, 2163), (76, 2415343), (90, 4459), (72,
34972), (73, 6789667), (75, 599857), (69, 126409), (68, 713511), (75, 2038885),
(53, 2978), (53, 15412), (67, 204364), (64, 13303), (62, 6044), (69, 266399), (55,
15605), (62, 15582), (77, 976037), (62, 39467), (77, 19887646), (68, 124812),
(72, 41513), (66, 59963), (78, 584137), (58, 173959), (73, 340820), (54, 19864),
(74, 4165936), (69, 2123992), (55, 3397), (74, 3028576)]
```

Here is the individual correlations between an artist's spotify popularity and their twitter following

```
[66.37209302325581, 34.56470588235294, 81.475, 29487.054347826088,
6437.648648648648, 18865.894736842107, 401992.7244897959,
45.265822784810126, 10.232876712328768, 138.5121951219512,
123056.2258064516, 45.377049180327866, 49475.31460674157,
13606.962962962964, 63841.38888888889, 2252.0, 8977.10465116279,
10.723076923076922, 159.17142857142858, 24944.545454545456,
5188.01282051282, 391660.6853932584, 26930.280487804877,
367.5846153846154, 55124.543209876545, 65888.70454545454,
325836.170212766, 4.0, 4411.309523809524, 5722.652777777777,
27253.777777777777, 172157.6265060241, 67715.96296296296,
23178.965909090908, 5843.657142857143, 84828.02325581395,
57015.865853658535, 2164.875, 22468.3, 98.97101449275362,
259.219512195122, 612.6333333333333, 425.2708333333333,
64120.53409090909, 281822.3229166667, 1941.2093023255813,
83211.16483516483, 1080.0595238095239, 12121.426966292134,
42636.414893617024, 68.55128205128206, 76248.17582417582,
1177499.092783505, 469301.22826086957, 256003.94565217392,
62428.0989010989, 54633.03296703297, 297.1818181818182,
303769.82954545453, 896541.81, 719.0864197530864, 23.933333333333334,
69420.94623655915, 14.408450704225352, 34956.80487804878,
19260.710843373494, 55751.83720930233, 685.8051948051948,
5323.835294117647, 1807.8461538461538, 17761.054945054944,
25412.936708860758, 2921.45, 45.333333333333336, 0.013888888888888888,
906233.5425531915, 948.0853658536586, 44009.388235294115,
18086.63529411765, 375.9268292682927, 22499.9358974359,
10708.915662650603, 270.0923076923077, 2569.0923076923077,
419.20338983050846, 45.78333333333333, 5732.88524590164,
65.04166666666667, 9682.652173913044, 1540.4193548387098,
12636.014285714286, 36014.833333333336, 175.1, 3691.2533333333336,
1320.1739130434783, 653.1857142857143, 886.3384615384615,
40.81132075471698, 31780.82894736842, 49.544444444444444,
485.72222222222223, 93009.13698630137, 7998.093333333333,
1832.0144927536232, 10492.808823529413, 27185.133333333335,
56.18867924528302, 290.79245283018867, 3050.208955223881, 207.859375,
97.48387096774194, 3860.855072463768, 283.72727272727275,
251.32258064516128, 12675.805194805194, 636.5645161290323,
258281.1168831169, 1835.4705882352941, 576.5694444444445,
908.530303030303, 7488.9358974358975, 2999.293103448276,
4668.767123287671, 367.85185185185185, 56296.43243243243,
30782.492753623188, 61.763636363636365, 40926.7027027027]
Here is the average correlation between an artist's spotify popularity and their
twitter following.
60816.86766591045
```

5) The visualization that you created


Artist and Spotify Popularity Rating


Artist and Twitter Following
Sniper Gang'
A Boogie Wit da Hoodie'
Rod Wave'
Polo G'
Money Man'
Juice WRLD'
Drake'
Key Glock'
Nardo Wick'
Meek Mill'
Travis Scott'
Culture Jam'
DaBaby'
NLE Choppa'
Young Thug'
Baby Keem'
Lil Tjay'
Mike Dimes'
Bankrol Hayden'
Playboi Carti'
Joyner Lucas'
Lil Wayne'
Moneybagg Yo'
22Gz'
2 Chainz'
Lil Durk'
Kanye West'
YoungBoy Never Broke Again'
Don Toliver'
Cordae'
Russ'
Big Sean'
Lil Yachty'
Gunna'
Coi Leray'
Megan Thee Stallion'
Rick Ross'
BIA'
Latto'
Remble'
GAYLE'
The Kid LAROI'
Ed Sheeran'
SZA'
Adele'
Glass Animals'
Lil Nas X'
CKay'
Elton John'
Doja Cat'
ACRAZE'
Post Malone'
Justin Bieber'
Bruno Mars'
Coldplay'
Imagine Dragons'
Farruko'
Jaymes Young'
Shawn Mendes'
Taylor Swift'
Lost Frequencies'
NEIKED'
Billie Eilish'
Emmy Meli'
ROSALÍA'
LISA'
Tiësto'
Amaarae'
Måneskin'
THE ANXIETY'
Olivia Rodrigo'
Swedish House Mafia'
Oliver Tree'
BoyWithUke'
Benson Boone'
Ariana Grande'
Tate McRae'
Jason Derulo'
Summer Walker'
Sleepy Hallow'
Kane Brown'
Luke Combs'
Ingrid Andress'
Parmalee'
Travis Denning'
CHASE WRIGHT'
Danielle Bradbery'
Jordan Davis'
Lauren Alaina'
Michael Ray'
Dustin Lynch'
Dierks Bentley'
Morgan Wade'
Cody Johnson'
Gabby Barrett'
Mitchell Tenpenny'
Dylan Scott'
Alana Springsteen'
Thomas Rhett'
BROWN & GRAY'
Chris Young'
Miranda Lambert'
Sam Hunt'
Riley Green'
Cole Swindell'
Eric Church'
Callista Clark'
Levi Hummon'
Brothers Osborne'
Tenille Arts'
Chase Matthew'
Parker McCollum'
Ashland Craft'
Nate Smith'
Kacey Musgraves'
Morgan Evans'
Blake Shelton'
Carly Pearce'
Walker Hayes'
Jimmie Allen'
Dan + Shay'
Mickey Guyton'
Jon Pardi'
Erin Kinsey'
Keith Urban'
Jake Owen'
Shy Carter'
Tim McGraw'
Artist
0
1000
2000
3000
4000
5000
Twitter following

6)

7) **Instructions for running your code**

   a) Run the Main.py File, and the database will populate.

8) **Documentation for each function that you wrote, including input and output for each function**

   a) **SPOTIFYAPI**

      i)
      ```
      makeDatabase(database)
      ```
      (1) **- Creates an Inital Database taking a database name, returns cur and conn for SQLite3 connection.**

      ii)
      ```
      makeSpotifytable(cur, conn):
      ```
      (1) **Creates the Spotify Artist Table using the cursor and connection, and doesn't return anything but does commit the Spotify table to the database.**

      iii)
      ```
      insertspotifydata(cur, conn, info):
      ```
      (1) **Inserts Spotify Data into Database using the cursor, the connection, and the passed info, doesn't return anything but does commit the added Spotify info to the database.**

      iv)
      ```
      artistlistfromplaylist(playlistid)
      ```
      (1) **Takes (a) Playlist ID(s) and searches for the playlist(s), returning a list of artists in that playlist**

      v)
      ```
      searchforartistpopularity(q, type = 'artist', limit = 10)
      ```
      (1) **Returns a list of a searched artist's popularity, taking a search query, a type of search, and a limit for results.**

      vi)
      ```
      makeartistpopularities(artistlist, artistdict= {})
      ```
      (1) **Creates a Dictionary of Artists and their Popularities using a passed artist list and a new or passed artist dictionary.**

   b) **Twitter**

      i)
      ```
      makeTwittertable(cur, conn):
      ```
      (1) **Creates the Twitter Table using the cursor and connection, and doesn't return anything but does commit the Twitter table to the database.**

      ii)
      ```
      inserttwitterdata(cur, conn, info):
      ```
      (1) **Inserts Twitter Data into Database using the cursor, the connection, and the passed info, doesn't return anything but does commit the added Twitter info to the database.**

      iii)
      ```
      artistsearchtwitter(artistname)
      ```
      (1) **Searches for an artist on twitter and pulls their follower count using a passed artist name as a search query**

      iv)
      ```
      spotifytwitterlookup(playlistids, followerdict = {}
      ```
      v) **Takes a list from Spotify of artists that appear on a playlist and Searches for individual artists on twitter, returning a dictionary of the artist and their follower count**

c) Main

   i)

   ```
   main()
   ```

   ii) Runs the whole program, which includes looking up artists on spotify from three popular playlists, looking them up on twitter, and storing the data pulled in the other programs in the database.

d) Calculation

   i) No function, but code does pull from JOINED database tables and calculates with the data.

9) Documentation of resources

   a) https://runestone.academy/runestone/books/published/py4e-int

   b) https://stackoverflow.com/questions

   c) https://docs.tweepy.org/en/stable/

   d) https://developer.spotify.com/documentation