***Initial Goals of Project***

1. Retrieve data from Spotify, Wikipedia, and Billboard
    a. From Spotify, we wanted to get playlist data from any user that was logged in on a computer while our code ran
    b. From Wikipedia, we wanted to get all the birthplaces of the artists in the given user's playlists
    c. From Billboard, we wanted to get the top 100 songs and see how many songs in the top 100 were in the user's playlist
        i. We also wanted to get top songs per musical genre
2. We wanted to store the information in a database(SIProject.sqlite) with 3 tables, one for Spotify, Wiki, and Billboard
    a. Spotify table
        i. 3 columns, first is playlist name, second is playlist ID number, and third is song
    b. Wiki table
        i. 3 columns, first is source (Spotify or Billboard), second is artist, and third is artist location
    c. Billboard table
        i. 3 columns, first is artist name, then rank, then song title

3. We wanted to create three visualizations
    a. One choropleth chart to map the artists in a user's playlist to where they live
    b. One bar graph to map how many songs in a user's playlist were in the top 100
        i. X-axis would be playlist name
        ii. Y-axis would be the amount of songs
    c. One bar graph to map how many artists live in a given list of cities
        i. X-axis would be city names
        ii. Y-axis would be how many artists in a user's playlist live in given city

***Goals that were achieved***

1. We were able to use the Spotify API, and use Beautiful Soup for Wiki and Billboard
    a. Used Spotipy for Spotify → the API that allows developers to retrieve information from Spotify website
        i. However, ended up only using data from one spotify user instead of any spotify user to have more consistent data
    b. We were able to get *some* artist locations from Wiki (will explain more in problems that we faced)
    c. We were able to get the Top 100 from Billboard
        i. We got the top songs per genre, but decided not to use that data because it made the tables too complicated

2. We stored information in three tables
    a. Spotify table
        i. Ended up having 4 columns instead of three, as we noticed that we needed to also get artist name from each song
        ii. Columns are playlist name, playlist ID, songs in each playlist, artist of each song in playlist
    b. Wiki table
        i. We were able to retrieve source, artist, and *some* locations (again, will explain below)
    c. Billboard table
        i. We correctly got the Top 100 songs, artists, and rank in the database
3. Visualizations
    a. We created three bar graphs
        i. One mapping how many songs on each user's playlist were in the top 100
        ii. Mapping birthplace of Billboard artists in the top 100
        iii. Mapping birthplace of artists in each Spotify playlist

***Problems that we faced***

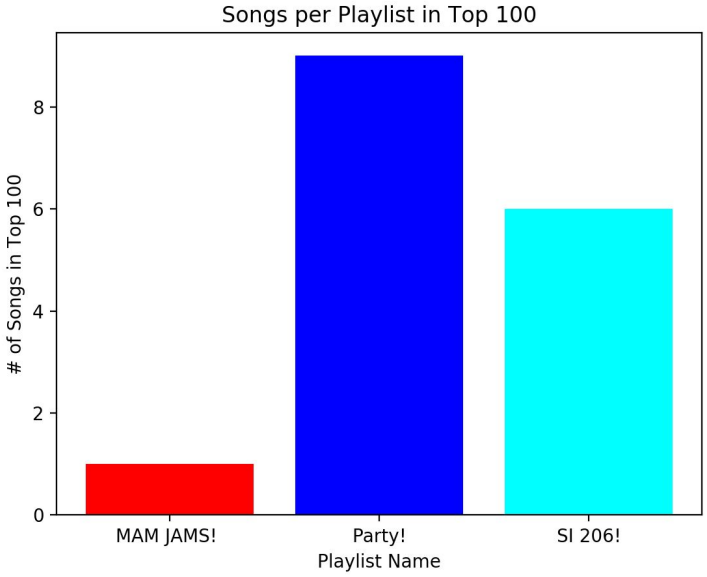
1. The first problem that we faced involved scraping an artist's birthplace from Wiki using their name that we found on Billboard and Spotify
    a. Since Wiki gives the birthplace, origin, or neither, we had to retrieve data from the website tags that included both birthplace and origin
        i. We made a for loop to encompass both tags
    b. In order to retrieve information from each artist's page, we changed the URL by adding their given name to the end of the link. This became problem as many artists were born with one name, however go by a different "show name," therefore the billboard names did not match up with the Wiki links.
        i. We used a try and except block and if the program could not find the page, we would mark the birthplace as "unknown"
    c. Lastly, some artist names match other people or objects. Wiki would then add "(singer)" or "(musician)" to the end of the URL. This again became an issue because we could not anticipate this change.
        i. At first we tried to make a regex that would allow characters after the name. This did not not work because it would grab the URL of many other pages. We again used a try and except block and if the program was unable to find the page, we marked it as "unknown"
2. The second problem that we faced was with our third visualization, where we mapped artist location on each playlist

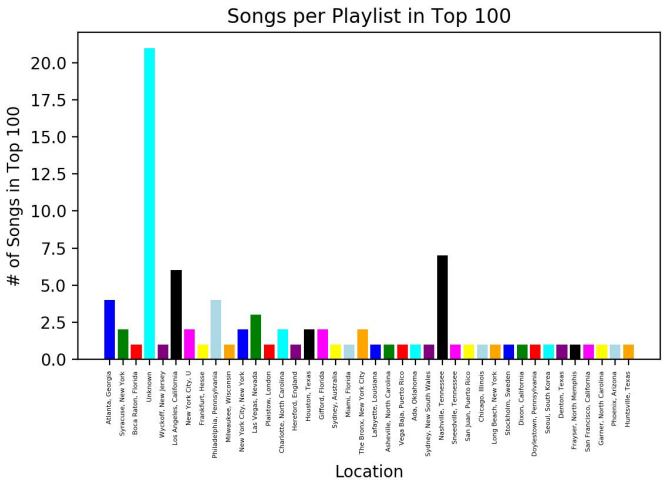
a. We were able to put the information on each playlist on a separate bar graph, however when the graphs were created, the bottom graph cut off the label of the x-axis for the graph above it
b. For this issue, we tried altering the location of the graphs, however the names of the cities on the x-axis were too long for the margins of the figure

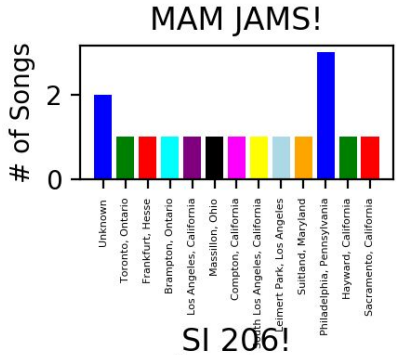
***File that Contains the Calculations from Database***

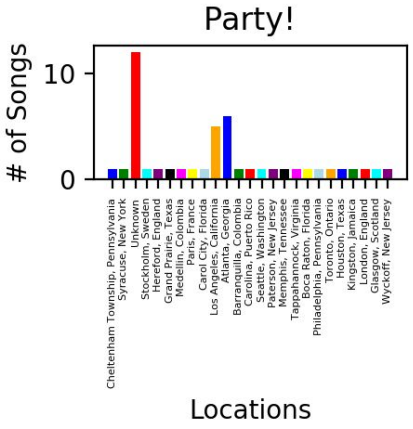
- In Zip file labeled as SIProject.json!
    - It is a String, but formatted as a dictionary. The dictionary has keys of each visualization and the values are the dictionary calculations that were used to make each visualization.

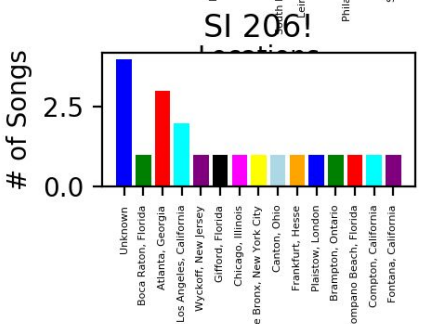
***Visualizations* (5)**



***Instructions for running the code***

*Run the code in this order:*

*Github Repository : [https://github.com/alexarathi/MAM](https://github.com/alexarathi/MAM)*

1. Spotify.py
    a. Instructions for running code
        i. Run spotify first
        ii. It will redirect you to google, you must copy and paste the google URL into terminal
        iii. Run again, and it should give you the spotify database
    b. Description of code (*each line of code is also commented out in the py file, since we did not use specific functions for spotipy)*
        i. Lines 1-13 import all necessary files and applications
        ii. Lines 19-21 use the file "spotify_info", which has client id's and access tokens, authorize our use of spotipy
        iii. Lines 24-31 establish the username of the user that we are retrieving information from, and finish authorizing our use of the data
        iv. Line 44 gets a large dictionary (results) of all the users playlist information
        v. Lines 46-59 gets the user's playlist names(playlist_name_list) and playlist ID numbers (playlist_list) from results, and puts them in a dictionary(playlist_id_name) where the key is the ID number, and the value is the actual playlist name
        vi. Lines 62-66 makes a dictionary(playlist_result_dict) where the key is the ID number, and the value is all the songs in the playlist
        vii. Lines 69-84 greats a dictionary(songs_in_playlist) where the key is the playlist's actual name, and the value is a list of songs in the playlist
        viii. Lines 86-97 writes playlist_results_dict to a json file and checks to make sure that there is no duplicates in it
        ix. Lines 100-110 creates a database and inputs the data to the database
2. Billboard.py
    a. Instructions for running code
        i. Run Billboard second
        ii. This will open one database with three tables for Hot 100, Hot 200, and EDM top 40
        iii. These will all have artist name, rank of song, and title of song in order
    b. Description of Code - all of the code is commented out in the file as well
        i. Lines 1-7 import all the necessary functions and applications
        ii. Lines 8-18 create three different soup objects for the different databases
        iii. Lines 20 - 22 create a database in sqlite called 'Billboard'
        iv. Line 25 creates the first table with the artist, rank and song respectively

v. Lines 27 - 34 inputs values taken from the (soup.find_all) function and inputs them into the first table
vi. Line 42 creates the second table within Billboard of the artist, rank and song respectively
vii. Lines 44 - 50 inputs values taken from the (soup.find_all) function and inputs them into the second table
viii. Line 57 creates the third table within Billboard of the artist, rank and song respectively
ix. Lines 59 - 65 inputs values taken from the (soup.find_all) function and inputs them into the third table
x. Lines 68 - 71 commits all the databases and closes the file

3. Wikipediascrape.py
    a. Instructions for running code:
        i. Run Wikipediascrape.py third in the terminal. It has different functions that pull from both the data created with the Spotify.py and Billboard.py.
        ii. When ran, it will create a new Table in the SIProject.sqlite database labeled "Wikipedia" containing the source (Billboard or Spotify), the Artist, and the Location.
        iii. The main goal of the code is to link artists with the location that they are from and to write that to the Database.
    b. Description of the functions:
        i. *ALL OF THE CODE IS COMMENTED OUT FOR SPECIFIC FUNCTION*
        ii. getartistURL
            1. Input -- A string with the name of an Artist
            2. Output -- A string of the correlating URL of the Artist
        iii. getartistLocation
            1. Input -- A string URL of the Artist
            2. Output -- A string Location of the Artist
        iv. allArtistInfo
            1. Input -- A list of strings, the strings being the Artists
            2. Output -- A dictionary with artist as key, location as value
        v. BillboardWriteToDatabase
            1. Input -- The artist dictionary
            2. Output -- Source (billboard), artist, and location to the Wikipedia Table for each key in the artist dictionary
        vi. SpotifyWriteToDatabase
            1. Input -- The Artist Dictionary

2. Output -- Adds to the Wikipedia Table the source(spotify), artist, and location for each key in the artist dictionary

4. Visualization.py
    a. Instructions for running code
        i. Run Visualization.py fourth in the terminal. It pulls from the SIProject.sqlite database and runs functions to visualize the calculations
    b. Description of the functions
        i. Get_spotify_dict
            1. Input -- Database
            2. Output -- Dictionary containing playlist as key and count of songs on billboard 100 as value
        ii. Spotify_billboard_visualization
            1. Input -- Spotify Dictionary
            2. Output -- Bar graph showing the count of songs on top 100 for each one of the spotify playlists
        iii. Get_billboard_artist_location
            1. Input -- Database
            2. Output -- Dictionary containing billboard artist location as key and count of how many are from there as value
        iv. Billboard_location_visualization
            1. Input -- billboard location dictionary
            2. Output -- Bar graph showing the amount of artists in each location on the billboard top 100
        v. Get_spotify_location_dict
            1. Input -- Database
            2. Output -- Dictionary with spotify playlist as key and a dictionary (key location|value artist count) as the value
        vi. Spotify_location_visualization
            1. Input -- Spotify location dictionary
            2. Output -- Makes 3 bar graphs with the location count of all the artists for each spotify playlist

***Documentation***

| ***Date*** | ***Issue Description*** | ***Location of Resource*** | ***Result*** |
|---|---|---|---|
| Friday, April 13th | Trouble with the Spotipy API | https://www.youtube.com/watch?v=tmt5SdvTqUI | These videos helped a lot! Since we previously did not know about Spotipy, |

| | | | |
|---|---|---|---|
| | | https://www.youtube.com/watch?v=vipVEWe86Lg&t=261s<br><br>https://www.youtube.com/watch?v=Tc5xnI6c8b0 | it was very helpful to get step by step instruction on how to get access tokens and set up a developers account. |
| Friday, April 13th | Spotipy functions | https://spotipy.readthedocs.io/en/latest/ | This website helped us learn the functions of Spotify and pull out the ones that we needed to get the playlists and song names |
| Monday, April 15 | How to insert a dictionary with values that are long lists | https://stackoverflow.com/questions/9336270/using-a-python-dict-for-a-sql-insert-statement | Helped! We successfully got our information in the database |
| Wednesday, April 24 | How to change font size on a visualization using matplotlib | https://stackoverflow.com/questions/3899980/how-to-change-the-font-size-on-a-matplotlib-plot | Helped! We changed the font size of our x and y axis labels |
| Wednesday, April 24th | How to add a column to a table | Office hours! (Katie) | Helped! We ended up having 4 rows in Spotify table. |