SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Final Project Report

Initial Goals:

Our initial goals centered largely around the use of the Spotify, Apple, and Genius API's. Originally we were going to try and retrieve data from these API's regarding the popularity of songs in different genres and regions. J was to use the Spotify API, Bernie was going to use the Apple API, and Jacob initially planned on using the Genius API. Between the three API's we wanted to compare what songs were most popular for an artist and then use a metric that we were going to determine at a later time in order to categorize whether a song was a breakout song for an artist or whether the artist themselves had just broken on to the scene.

Goals Achieved:

Our final product ended up being somewhat different than our original plan but the end goal of comparing the popularity between songs on different platforms was achieved. The first switch that occurred was a switch from the Apple API to the Shazam API. This was due in large part to the necessity of creating a developers account for the Apple API. This change did not impact the overall dynamics of the project as both Apple and Shazam let users view the most popular songs on their platforms. The next major shift from our original goals was the switch from using the Genius API to using BeautifulSoup to scrape data from the billboard hot 100 website. This swap allowed us to transition from using an API that did not offer much useful information regarding the popularity of songs or artists, to using a website whose sole purpose was to display information regarding the popularity of any given song in the top 100. From here we were able to stick to many of our original goals including comparing the popularity of different songs between the different platforms. We also were able to somewhat determine if a song was a breakout song for the artist, but we decided to limit how we used this data as we arbitrarily determined it without clear standards. We sadly were unable to compare the popularity of any given song between different regions; however as a group we determined that the popularity between platforms was in and of itself more than enough for a project of this size.

Problems that we faced:

- Bernie
    - One of the earliest problems I faced was being able to not access the Apple API. Apple was one of the API's that we were really excited to play around with, but we could not get access due to it needing a developer Id and a few other

Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

requirements that we did not have time to look into. Shazam became our second API, and it worked nicely, however my next problem was trying to get my data to look a certain way. I ended up just creating a list of tuples that included the song name, a list of the artists in the song, and the rank of the song. For some reason there was one song in the data gathered that had no artist name, for this one I created a case to just add the artist name as unknown. The most difficult problem that I faced was that of inputting 25 data points onto the database. I struggled for at least two days on this code. I went to office hours and watched a video that allowed me to solve this, but it really was a hassle to figure out. For the visual side it was difficult to create a function that built the shazam graph because some examples did not include how to name the x-axis or title. It took me three hours to find the right graph I wanted to use.

- Jacob
    - On my end my largest problem was in determining the best method by which to limit the data that was input into the table at one time. Originally I began by using a while loop, but for some reason this while loop continued inputting all 100 into the table. My next solution involved a method that was very similar to Bernies in which the SELECT statement was used to obtain that maximum value of rows in the table and then would use this value and would begin inputting 25 data points into the table from this starting point through the use of a for loop and a break statement. After this the largest difficulty I had was in the creation of our data processing file. This was supposed to look at all the songs that are shared between the three tables and would then find the average ranking of the shared songs. However this was easier said than done, as the data received from the apis and from Billboard were all formatted slightly differently. My solution to this problem was to go back and make sure that all the shared songs were entered into their tables as uppercase words, this way if a song was shared between the three tables it would be in the same format no matter where the data was retrieved from. My next issue came when I was creating a visualization for the billboard data. This was mostly an issue of trying to find the best way to model the data that I was using. After looking through the matplotlib tutorials I decided that the best format would be a 3d scatterplot which would allow a user to best visualize exactly what was going on with the data.

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

- J
    - One of my first problems was understanding how to workaround the Spotify API documentation. On the website, it tells you how to work with it, but it took a much different route than was necessary for this project. I spent a while experimenting with setting up what they say is a “Proper Environment” that included a mix of Javascript and HTML coding, which I knew was probably doing too much. I just couldn’t find how else to access the API, and after Bernie had to pivot to Shazam I really didn’t want us to pivot again. I actually ended up getting it somewhat working though, which I was quite elated with. It felt good to go through all that just to get it moving. I did realize what wasn’t necessary in accessing the Spotify API in order to access it somewhat, but there was still room for improvement. So I began working with the API to be able to retrieve the data that we wanted. Originally, we wanted to compare the top songs on each platform, however on Spotify their “charts” are playlists owned by Spotify. The documentation makes it clear that you can only access user owned playlists, and in trying to force it in the code the way you would with normal playlists wasn’t working. To my luck however, there was a key value Spotify provides that we deemed an appropriate pivot to my part of the data collection, namely song popularity which could be used to calculate an average artist popularity. These could serve as stand ins to us not having the charts that we wanted. Things were smooth sailing until my access token needed to be refreshed. I actually wasn’t sure how to actually get one, so I repeatedly generated temporary ones that I hoped would last in one of the Spotify API web consoles. That clearly became a problem though. In working through the Spotify API tutorial, I decided to use the Spotipy module to connect with their API instead. I would have done it before, but the documentation actually didn’t reference it so I wanted to try it otherwise. However, I learned that the Spotipy module is exactly appropriate for the scope of this project, as it is purely regarding Spotify data and not any user data. Thus, by shifting a couple statements around to use the module tools I was able to bypass the need for an authorization token as the module only required a client ID and a secret key. The data processing side ended up working out smoothly, as Bernie and Jacob were a few paces ahead of me after my back and forth with the documentation and we ended up being able to have the key characteristic of the Spotify data be the unique “Popularity” rating that I obtained earlier.

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Calculations File:

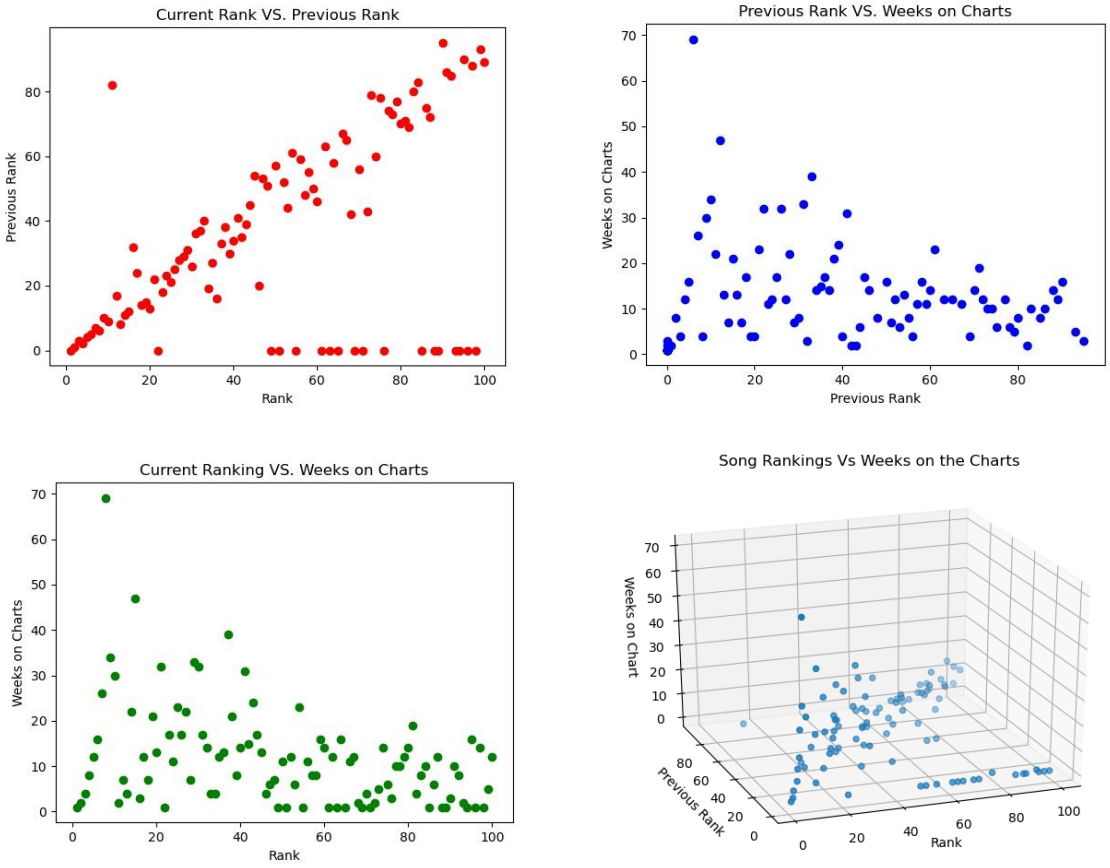
Below is the average "popularity" out of 100 total of a song between Spotify, Billboard, and Shazam.
A value of one is the most popular and a value of 100 is the least popular.
PEACHES ---- 2
LEAVE THE DOOR OPEN ---- 3
SAVE YOUR TEARS ---- 6
ASTRONAUT IN THE OCEAN ---- 6
DRIVERS LICENSE ---- 7
MONTERO (CALL ME BY YOUR NAME) ---- 10
CALLING MY PHONE ---- 11
LEVITATING ---- 13
UP ---- 14
HEARTBREAK ANNIVERSARY ---- 14
WITHOUT YOU ---- 16
WHAT YOU KNOW BOUT LOVE ---- 17
BEST FRIEND ---- 17
BEAUTIFUL MISTAKES ---- 18
MY EX'S BEST FRIEND ---- 21
WANTS AND NEEDS ---- 21
BLINDING LIGHTS ---- 23
BACK IN BLOOD ---- 25
BEAT BOX ---- 26
TRACK STAR ---- 26
WHAT'S NEXT ---- 28
GO CRAZY ---- 28
WE'RE GOOD ---- 29
MOOD ---- 30
HEAT WAVES ---- 30
ON ME ---- 31
PUT YOUR RECORDS ON ---- 32
THEREFORE I AM ---- 33
GOOD DAYS ---- 33
THE GOOD ONES ---- 34
TOMBSTONE ---- 41
FOR THE NIGHT ---- 44
QUICKSAND ---- 44
FOREVER AFTER ALL ---- 45
TIME TODAY ---- 45
THE BUSINESS ---- 45
STREETS ---- 46
CRY BABY ---- 47
MASTERPIECE ---- 47
HOLD ON ---- 48
WILLOW ---- 52
DAMAGE ---- 53
BREAKING UP WAS EASY IN THE 90'S ---- 57
YOU GOT IT ---- 57
PICK UP YOUR FEELINGS ---- 58
MADE FOR YOU ---- 59
LA NOCHE DE ANOCHE ---- 61
ONE TOO MANY ---- 68

Link to actual file: https://github.com/berniev9/SAG-Music/blob/master/ProcessedData.txt

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Visualizations:

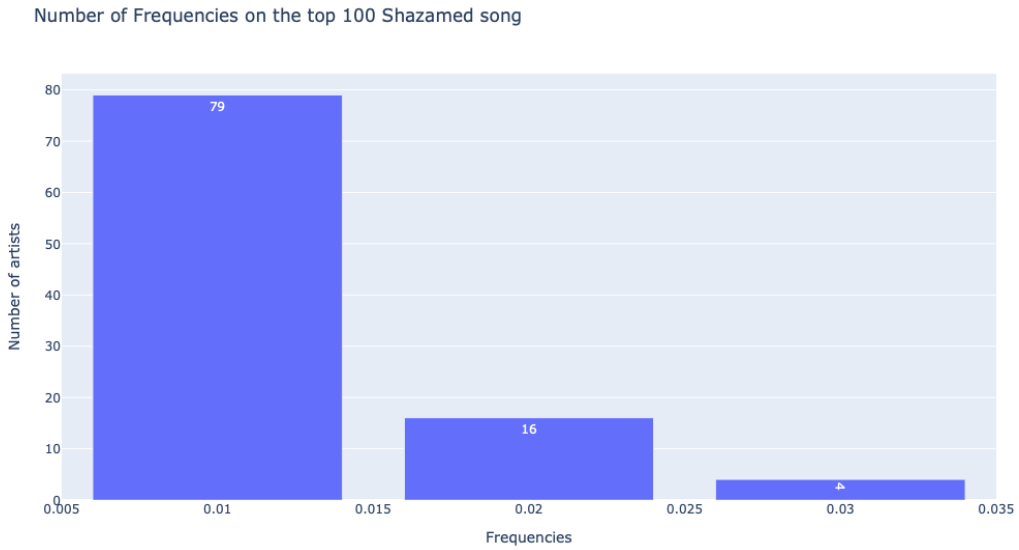
Visualization number 1: Using data from the Billboard table.

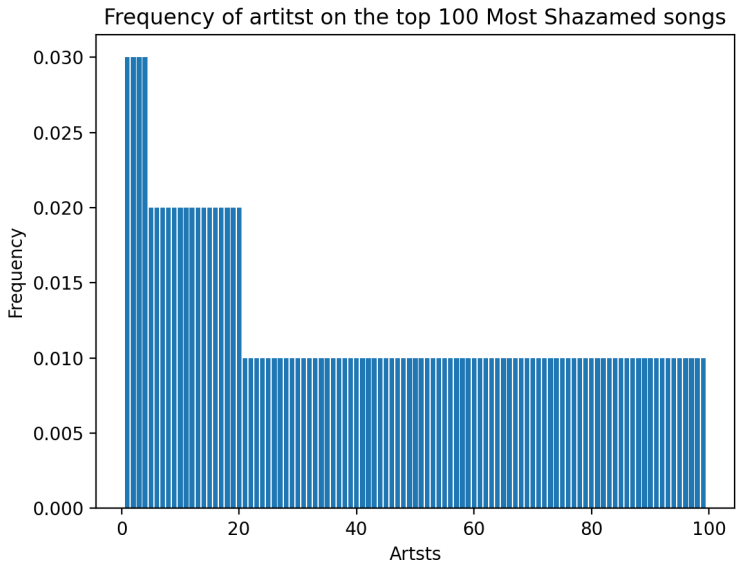


(In Current Rank vs. Previous Rank, the line of dots at previous rank = 0 represent the songs that weren't on the billboard charts the previous week.)

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Visualization number 2:





SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Tables Used to Aid in Comprehension of Visualization 2:

Rank	Artist	Frequency
1	the-weeknd	0.03
2	pop-smoke	0.03
3	young-thug	0.03
4	Unknown	0.03
5	bruno-mars	0.02
6	justin-bieber	0.02
7	doja-cat	0.02
8	megan-thee-stallion	0.02
9	olivia-rodrigo	0.02
10	drake	0.02
11	tate-mcrae	0.02
12	dua-lipa	0.02
13	young-stoner-life	0.02
14	gunna	0.02
15	bad-bunny	0.02
16	jhay-cortez	0.02
17	pooh-shiesty	0.02
18	ajr	0.02
19	ariana-grande	0.02
20	ava-max	0.02
21	anderson-paak	0.01
22	silk-sonic	0.01
23	masked-wolf	0.01
24	curtis-harding	0.01
25	lil-tjay	0.01
26	6lack	0.01
27	givēon	0.01
28	the-kid-laroi	0.01
29	maroon-5	0.01
30	mooski	0.01
31	kali-uchis	0.01
32	yung-bleu	0.01
33	polo-g	0.01
34	aurora	0.01
35	machine-gun-kelly	0.01
36	blackbear	0.01
37	morray	0.01
38	saweetie	0.01

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Instructions for running code

Introduction:

- There are five files, one to gather the Shazam data, one to gather billboard data, one to gather spotify data, one to create the visuals, and lastly one to input 25 data sets onto the database at a time.

Step 1: Run main.py four times

- This collects and inserts 25 datasets for each API/website onto their respective database

Step 2: Run Data Processing.py

Step 3: Run visual.py

- Once the data is gathered and stored you can run visual.py for the visuals to be created.

Step 4:

- Enjoy :)

Shazamapi.py Functions

```
def top_100_songs(Country_code):
```

// Requires

- Takes in a country code. In this case we took in the country code for the U.S.A.

// Effects

- First creates a connection to Shazam's API
- Creates a data set of the top 100 most shazamed songs in the U.S
- Iterates through the data set for it to become a list of tuples

// Output

- A list of tuples that holds information about each song. Song name, list of artists in the song, rank of the song on the chart. EX: ('Leave The Door Open', ['bruno-mars', 'anderson-paak', 'silk-sonic'], 1)

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def get_artists(artists):
```

// Requires
- Takes in a reference of a list of artists for a particular song

// Effects
- Iterates through the reference to see if the song has an artists
- If it doesn't then the artist will be set to unknown.

// Output
- Creates a list of all the artists in that specific song

```
def get_count_artists_appear(US_top100):
```

// Requires
- Takes in a list of tuples. Song name, list of artists in the song, rank of the song on the chart EX: ('Leave The Door Open', ['bruno-mars', 'anderson-paak', 'silk-sonic'], 1)

// Effects
- Iterates through the list to gather up all the artists
- Counts the amount of times the artist appears on a song
- creates a dictionary where the key is the artist name and the value is the amount of time they appear on the top 100 most shazamed songs

// Output
- Returns a sorted list of tuples that hold the artists name and amount of times they appear(artist name, number of times they appear)

```
def calculate_artist_frequency(artist_rank_list):
```

// Requires
- Takes in a list of tuples of all the artists in the top 100 most shazamed songs EX('Leave The Door Open', ['bruno-mars', 'anderson-paak', 'silk-sonic'], 1)

// Effects
- Calculates the frequency of the artist by dividing the number of times they appear by 100. Since we gathered up the top 100 most shazamed songs

// Output
- Returns a dictionary with the artist name as the key and the frequency as the value

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def get_artist_pop_song(artist,US_top100):
```

// Requires

- Takes in one artist name and the list of tuples containing song name, list of artists that appear for that song, and rank of the song tuple example: ('Leave The Door Open', ['bruno-mars', 'anderson-paak', 'silk-sonic'], 1)

// Effects

- Iterates through the list of tuples
- Finds the first time the artists appears on a top 100 most shazamed song

// Output

- Returns the artist's most popular song name

```
def create_Shazam_table(cur,conn, US_top100):
```

// Requires

- Curr, conn and a list of tuples, tuple ex: ('Leave The Door Open', ['bruno-mars', 'anderson-paak', 'silk-sonic'], 1)

// Effects

- Creates the datatable called Shazam_data that holds the data that was originally given to us by the API
- Inserts 25 data sets onto the table

// Output

- No output the functions inserts data onto the given datatable

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def create_calculate_shazam_table(cur,conn, dictionary, US_top100):
```

// Requires

- Curr, conn and a list of tuples, tuple ex: ('Leave The Door Open', ['bruno-mars', 'anderson-paak', 'silk-sonic'], 1), a dictionary where the key is the artist name, and the value if the frequency the artist appears on the top 100 most shazamed songs

// Effects

- Creates the datatable called Shazam_Calculated_data that the calculated date, artists most popular song and the artist name
- Inserts 25 data sets onto the table

// Output

- No output the functions inserts data onto the given datatable

Billboard top 100.py Functions

```
def setUpDatabase(db_name):
```

// Requires

- Db_name. This is the name of the database that we will be creating the tables within

// Effects

- Creates the database that all our functions will use throughout the project

// Output

- Creates a cursor and connector that are used later when creating, adding to, and selecting from tables

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def billboard_soup(year, month, day):
```

// Requires

- Year, month, and day. Each of these inputs are integers and are used in the creation of the url that BeautifulSoup will be scraping from

// Effects

- Creates a url using the year, month, and day that was input by the user
- Uses BeautifulSoup to scrape (Rank, Song, Artist, Previous Rank, Peak Rank, and Weeks on the Charts) from the website that the url links to
- Creates a list of these variables through the use of a for loop.

// Output

- A list in which each item has the form: (Rank, Song, Artist, Previous Rank, Peak Rank, and Weeks on the Charts)

```
def billboard_table(year, month, day, db_name):
```

// Requires

- Requires no inputs for itself but Billboard_soup is called within so requires year, month, and day to be input

// Effects

- Calls setUpDatabase within itself to set up the database that will be used.
- Creates a table titled Billboard with columns: (Rank, Song, Artist, Previous Rank, Peak Rank, and Weeks on the Charts)
- Uses data from Billboard_soup and inputs it into the table
- Inserts 25 data sets onto the table

// Output

- No output from the function as the data is directly input into the table within the database

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

SpotifyGas.py Functions

```
def create_Spotify_table(cur, conn):
```

// Requires
- Requires data from the Billboard Table

// Effects
- Creates a table titled Spotify with columns: (Rank, Song, Artist, Popularity, Popularity_Status)
- Uses data from Billboard_soup and combinedata and inputs it into the table
- Inserts 25 data sets onto the table

// Output
- No output from the function as the data is directly input into the table within the database

```
def songsearchinfo(song):
```

// Requires
- Requires a song name, in our code passed into it from Billboard

//Effects
- Accesses the Spotify API to retrieve data on the song

//Output
- A dictionary of the song's information from Spotify including information like Artist, ID, Album, Popularity, and so on

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def songsearchgetartist(song):
```

//Requires
- Requires the same song name

//Effects
- Calls the songsearchinfo function to access the passed song's spotify data
- Searches that data for the artist name

//Output
- Returns the name of the artist in a string

```
def songsearchgetid(song):
```

//Requires
- Requires the same song name

//Effects
- Calls the songsearchinfo function to access the passed song's spotify data
- Searches that data for the artist ID

//Output
- Returns the artists' ID

```
def songsearchgetpopularity(song):
```

//Requires
- Requires the same song name

//Effects
- Calls the songsearchinfo function to access the passed song's spotify data
- Searches that data for the song's spotify popularity rating

//Output
- Returns the song's popularity

Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def toptracksartistdata(song, market = "US"):
```

//Requires
- Requires the same song name and the market desired (defaulted as the US)

//Effects
- Calls the songsearchgetid function to access the passed song's Artists's id
- Searches that Spotify API for that artist's top tracks using the artist_top_tracks method

//Output
- Returns a list of dictionaries of the artist's top tracks and their data

```
def toptracksartistlist(song, market = "US"):
```

//Requires
- Requires the same song name and the market desired (defaulted as the US)

//Effects
- Calls the toptrackartistdata function with a passed song and market
- Creates a list for the artist's top song names
- Parses the dictionary for the song names only and adds them to the list

//Output
- Returns the list of the artist's top song names

```
def meanartistpopularity(song, market = "US", popularratings = False):
```

//Requires
- Requires the same song name and the market desired (defaulted as the US), along with a boolean that can enable the printing out of the artist's song's popularities immediately

//Effects
- Calls the toptrackartistdata function with a passed song and market
- Can print out the individual popularities of each of the artists' top songs
- Creates a list to contain the artist's top tracks' popularities
- Parses the dictionary from the toptrackartistdata function for the song popularities only and adds them to the list
- Calculates the average popularity of the artist's top songs

//Output
- Returns the average popularity of the artist's top songs

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def breakoutsongforartist(song, market = "US"):
```

//Requires

- Requires the same song name and the market desired (defaulted as the US)

//Effects

- Calls the meanartistpopularity function and the songsearchgetpopularity function for the passed song
- Determines if an artist is generally popular or not on spotify
- Then determines if the song is more popular than the average of the artist's other songs

//Output

- Returns a designation of if the artist is or is not popular AND if their song is uniquely popular or not. If the song is uniquely popular for the artist and the artist is not an already popular artist, it is deemed a breakout song

```
def combinedata(song):
```

//Requires

- Requires the same song name

//Effects

- Calls the songsearchgetartist, songsearchgetpopularity, and breakoutsongforartist functions with a passed song
- Isolates the song, artist, and popularity
- Creates a breakout rating for the song based off the results of the breakoutsongforartist function
- Creates a tuple of the song's key data

//Output

- Returns a tuple containing the song, artist, popularity, and breakout rating

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Data Processing.py functions

```
def setUpDatabase(db_name):
```

// Requires
- Name of the Database file that the datatables will be inserted into

// Effects
- Creates a path way to the given database

// Output
- Creates a cur and conn that will allowed us to access the database file

```
def dataprocess(cur, conn):
```

// Requires
- Input is a cursor and a connector

// Effects
- Uses select from with JOIN to gather data regarding the rankings of songs that are shared between the three main tables.
- Then it takes these ranking and averages them for each song
- Creates a new list of tuples with the first item in each tuple being the song name and the second being an integer version of the average taken from the rankings

// Output
- Outputs said list of tuples regarding song names and averaged ranking

```
def writetotext(cur, conn, text_name):
```

// Requires
- Cursor, connector, and a text file name

// Effects
- Creates text file
- Uses dataprocess() function to get list of tuples regarding average song ranking
- Writes each song name and ranking to the text file

// Output
- No output. Text file is either edited or created from function.

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Visual.py functions

```
def setUpDatabase(db_name):
```

// Requires
- Name of the Database file that the datatables will be inserted into

// Effects
- Creates a path way to the given database

// Output
- Creates a cur and conn that will allowed us to access the database file

```
def grabs_shazam_data(cur, conn):
```

// Requires
- Cur, conn to the database

// Effects
- Creates a list of all the data sets in the datatable Shazam_calculated_data
- Iterates through the list to check the different types of frequencies
- Creates a dictionary where the key is the frequency, and the value is the amount of artists that have that same frequency

// Output
- Returns a sorted list of tuples where the first value is the frequency and the second value is the number of artists with that frequency Ex: [(0.01, 81), (0.02, 15), (0.03, 3)]

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def create_shazam_visual(data):
```

// Requires
- Takes in a list of tuples where the first value is the frequency and the second value is the number of artists with that frequency Ex: [(0.01, 81), (0.02, 15), (0.03, 3)]
-

// Effects
- Creates the first visual for shazam
- Creates a bar graph where the x-axis is the type of frequency, the y- axis is the amount of artists who have that frequency

// Output
- Creates a graph that shows the number of frequencies on the top 100 shazamed songs

```
def grab_shazam_key_value(cur, conn):
```

// Requires
- Cur, conn to the database

// Effects
- Creates a list of all the data sets in the datatable Shazam_calculated_data
- Iterates through the list to create a list for artists, ranks, and frequencies

// Output
- Returns a list of artists, a list of frequencies, and a list for ranks

These visualizations work together to show the artists frequencies {

```
def creates_shazam_2nd_visual(values, ranks):
```

// Requires
- Takes in a list of frequencies, and a list of ranks

// Effects
- Creates a bar graph that shows the 100 artists and their frequencies.

// Output
- Creates a bar graph

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

```
def create_shazam_artist_table(keys,ranks, values):
```

// Requires
- Takes in a list of artists, list of frequencies, and a list of ranks

// Effects
- Creates a table that shows the artists, the artist rank, and the frequency they have when it comes to how much they appear on the top 100 most shazamed songs

// Output
- Creates a table

```
def billboardvisual(cur, conn):
```

// Requires
- Inputs are cur and conn in order to be able to connect to database/table

// Effects
- Selects the rank, previous rank, and number of weeks on charts for each song within the table titled 'Billboard'
- Then uses this data and creates a 3d scatter plot in which the x axis is current rank, the y is previous rank, and the z axis is number of weeks on the charts
- Then uses already created lists of data to create 3 scatter plots. One of each axis in order to aid in the clarification of the 3d plot.

// Output
- Creates a 3-dimensional scatter plot graph
- Creates 3 scatter plots. One for each axis of the 3-dimensional scatter plot

Main.py functions

```
def main():
```

We added this file to make it easier on the grader to gather 25 data sets for each datatable. You will have to run this file four times in total to get a total of 100 data sets.

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

Resources used:

Date	Issue Description	Location of Resource	Result. (Did it solve the issue)
April 10, 2021	Spotify API Tutorial	Web API Tutorial | Spotify for Developers	No, this is what had me extremely confused for a while. I was able to work with it a little, but it wasn't what I needed.
April 10, 2021	Spotify API Documentation	Web API Reference | Spotify for Developers	Yes, this showed what the API was and was not capable of.
April 10, 2021	Genius API Documentation	https://docs.genius.com/	No, after looking through the API documentation I determined that this API would not fit our project and moved onto Billboard.
April 10, 2021	Shazam API Documentation	https://rapidapi.com/tipsters/api/shazam-core?endpoint=apiendpoint_4e971f7c-ebc2-4e9b-8f43-5f2860dcdd65	Yes this showed me the types of data I could gather as well as how to do so.
April 12, 2021	How to limit the data being collected	https://rapidapi.com/tipsters/api/shazam-core?endpoint=apiendpoint_4e971f7c-ebc2-4e9b-8f43-5f2860dcdd65	Yes(sort of), limited the amount of data together by 100

SAG Music
SI 206
Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

April 13, 2021	Exploring the Spotify API in Python	Exploring the Spotify API in Python | Steven Morse (stmorse.github.io)	Not really. It did introduce me to the easier Spotipy package which I later used, so that was the real impact.
April 13, 2021	User Playlists and Access Token	Get a List of Current User's Playlists | Spotify for Developers	Yes. This indicated that only user playlists were accessible. Furthermore the web console provided a temporary token that I used until I later learned how to avert the constant need to hardcode it.
April 14, 2021	Having trouble visualizing the data given from the Shazam Api	https://jsonformatter.org/	No
April 16, 2021	Spotipy Package	Welcome to Spotipy! — spotipy 2.0 documentation	Yes. It was still a little dense but made it so much easier to work with the API. It did eventually resolve the token issue as well.
April 18th	Was under the impression that I had to encrypt my client id and secret to get an authorization token	Encoding and Decoding Base64 Strings in Python (stackabuse.com)	No. I later found out that this was unnecessary.
April 20, 2021	Basic Database Reference	21. Databases — Python for Everybody - Interactive	Somewhat. I just needed to check something specific in the formatting.

Students : Bernie Velasquez: berniev@umich.edu
Jacob Cleaver: cleaverj@umich.edu
J Wilton: wilton@umich.edu

		(runestone.academy)	
April 22, 2021	Having trouble inserting 25 datasets at a time into the datatable	https://umich.zoom.us/rec/play/stQGK7NMUYluy_VE8dLp2bQjpLbTm9MKo57VecVof9QzJ_UxwZu7ozkYQlFApWWTVfnTpBFnvukqkqwM.XQAuis0Em4Hhp0Q-	Yes
April 24, 2021	Naming the table, the x-axis, and y-axis	https://plotly.com/python/bar-charts/	Yes
April 24, 2021	Creating a 3 dimensional plot	https://matplotlib.org/stable/gallery/mplot3d/scatter3d.html	Yes, this allowed me to be able to better understand how matplotlib works and specifically what the best manner in which to plot a 3-d plot is.