**Group Name:** Wilkins Shelton Records

**Members**:

Caleb Shelton: sheltocr@umich.edu

Nakia Wilkins: [nakiasw@umich.edu](mailto:nakiasw@umich.edu)

**GitHub Repository:**

<https://github.com/nakiasw/SI206FinalProject>

**Project Goals:**

* Predict the most popular genre in BillBoard Top 100 for 2020 based on user input 3 year range
* Retrieve and store the song titles, artists, and ranking from BillBoard’s Top 100 list for any given range of 3 years
* Request Last FM API for genres and store into same database
* Count the number of times a genre appeared on the Top 100 list and provide visuals of our findings

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Achieved Goals:**

We successfully achieved all of the goals stated in the previous section, as our program will output a prediction for the 2020 BillBoard Top 100 top genre. Our database also stores all the information we needed to make the calculations, and the API is properly requested. However, there were a lot of features that we weren’t able to implement as in depth as we wanted to. The problems with these will be described in the next section.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Problems:**

The biggest roadblock in this project was finding a useful API. When we first read the documentation for a couple of them, we thought they suited our needs. However, once we started implementing functions and trying to get proper requests from the Genius API, we realized it wouldn’t give us the specific data that we needed. The Spotify API required data like unique IDs for each track and artist that only they have access to, which made it complicated to get requests from that API.

Once we found the Last FM API, we could finally get access to the specific data that we wanted: genres. However, we wanted to also get a record label and release date so our predictor could have more features, but those weren’t accessible. Also, the genres from Last FM didn’t seem too reliable, as some seemed to just be tags for that song like the artist name or something about the song. Some of the API requests returned inconsistent data, requiring us to implement error handling and a lot of micromanaging.

Although we were able to retrieve most of the songs genres, a lot went unaccounted for, and this will hurt our predictor’s accuracy in the long run.

In terms of some of our more complicated features, we just didn’t have time to teach ourselves how to perform linear regression on python, or enough data to find more specific correlations.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Calculation File:**

Our file that holds the data calculated from our database is called ‘data.txt’ and can be found in our GitHub repository. It is a simple text file that gets written to once we find the frequency of every single genre stored into our database. Attached is a screenshot.

As the image shows, this is the data collected for the frequency of each genre (found using Last FM API) in the BillBoard Top 100 for the years 2017, 2018, and 2019. “None” denotes the fact that our API couldn’t find a genre tag for it. This file shows that we were dealing with “fake” genres and not completely accurate data.

This data is then shortened to only display the top 4 genres from each year because we figured those would be the most important when it came down to making a prediction.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Visualizations:**

Attached are the visuals we created using Matplotlib.

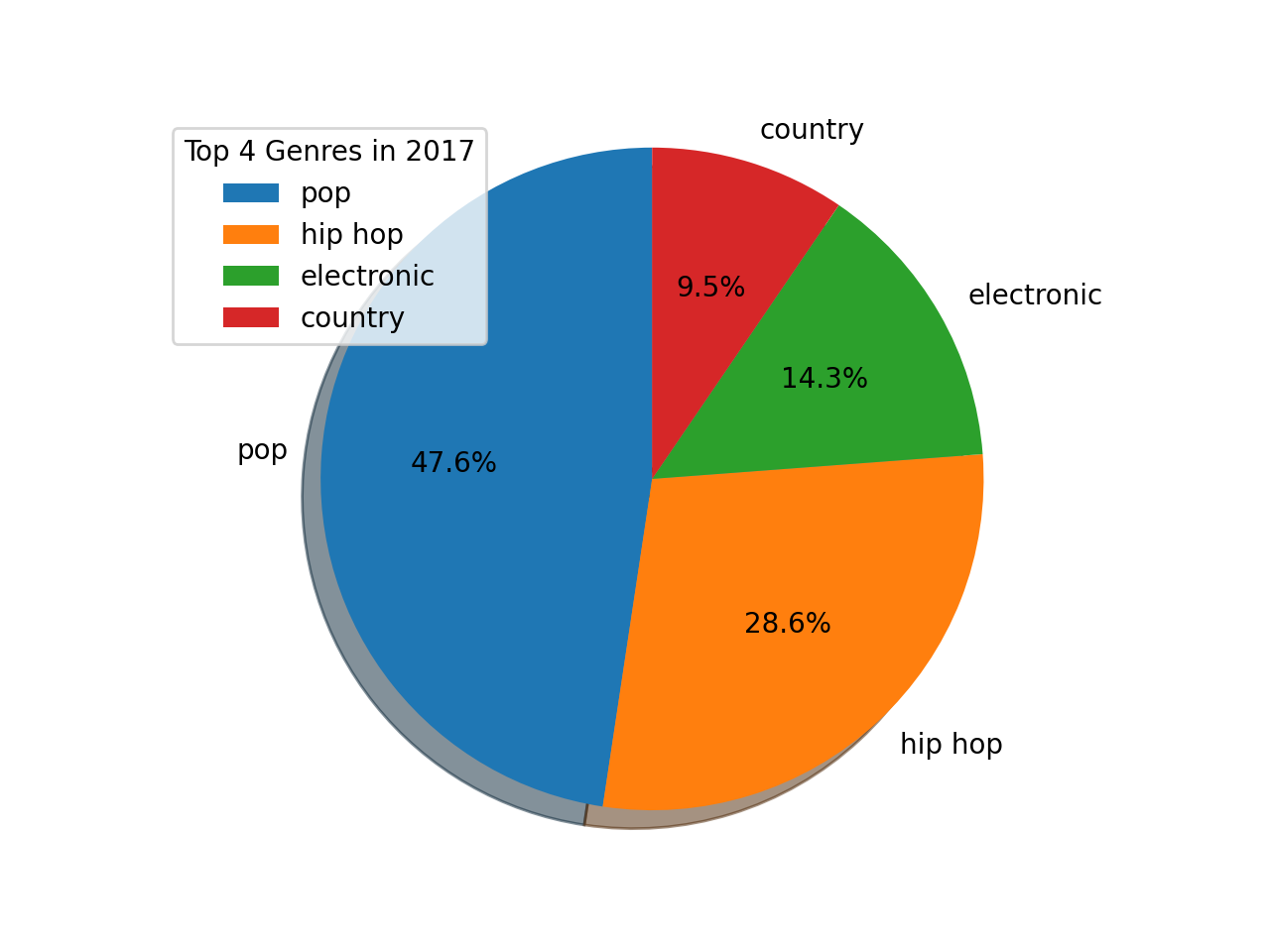


Figure 1. Top 4 Genres from 2017

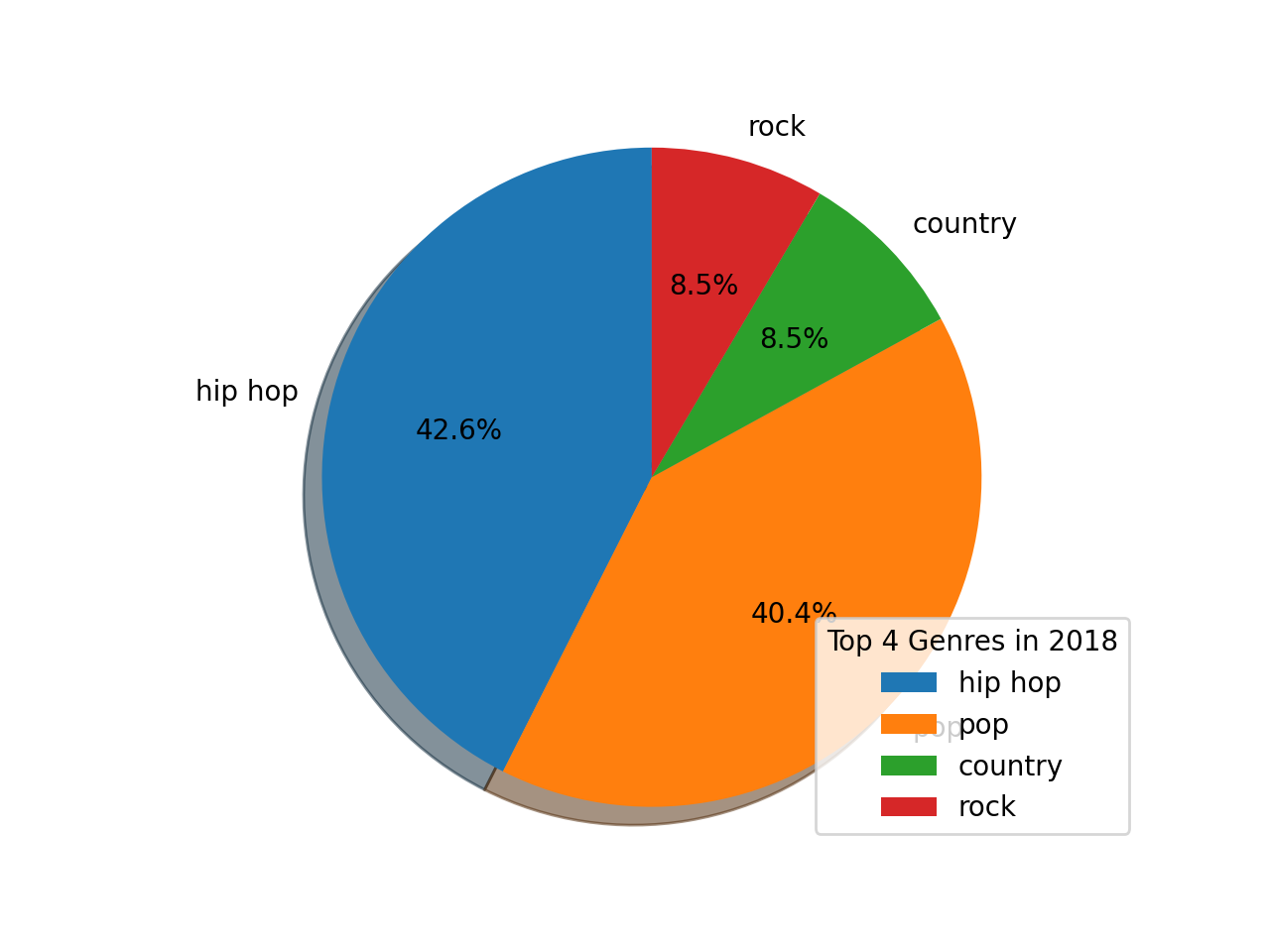


Figure 2. Top 4 Genres from 2018

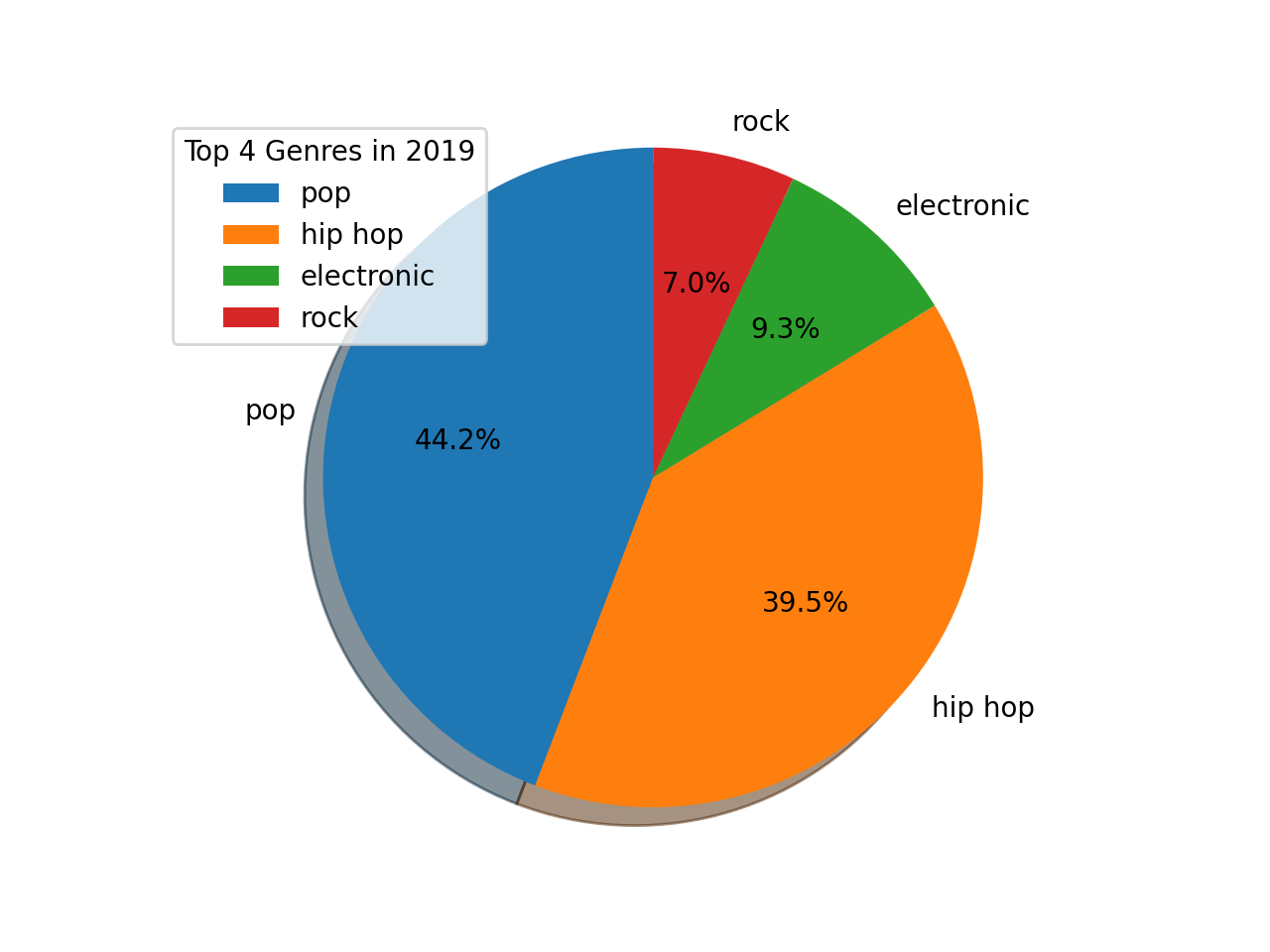


Figure 3. Top 4 Genres from 2019

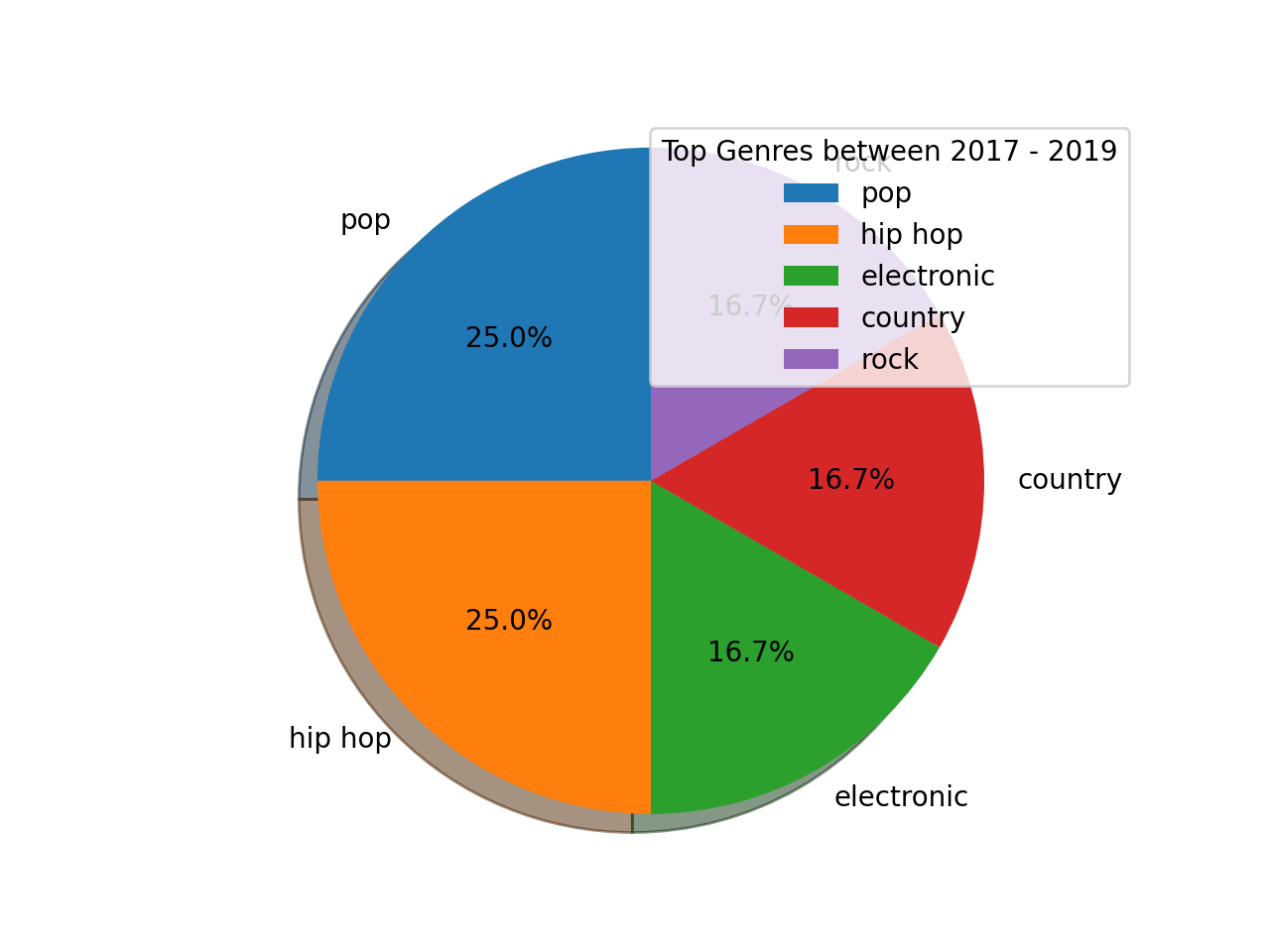


Figure 4. Top Genres between 2017 - 2019. This is basically a summary of the first 3 charts and is used to show the overall breakdown for the year range. Note that there are more than 4 genres listed because not every year between 2017 - 2019 had the same top 4 genres.

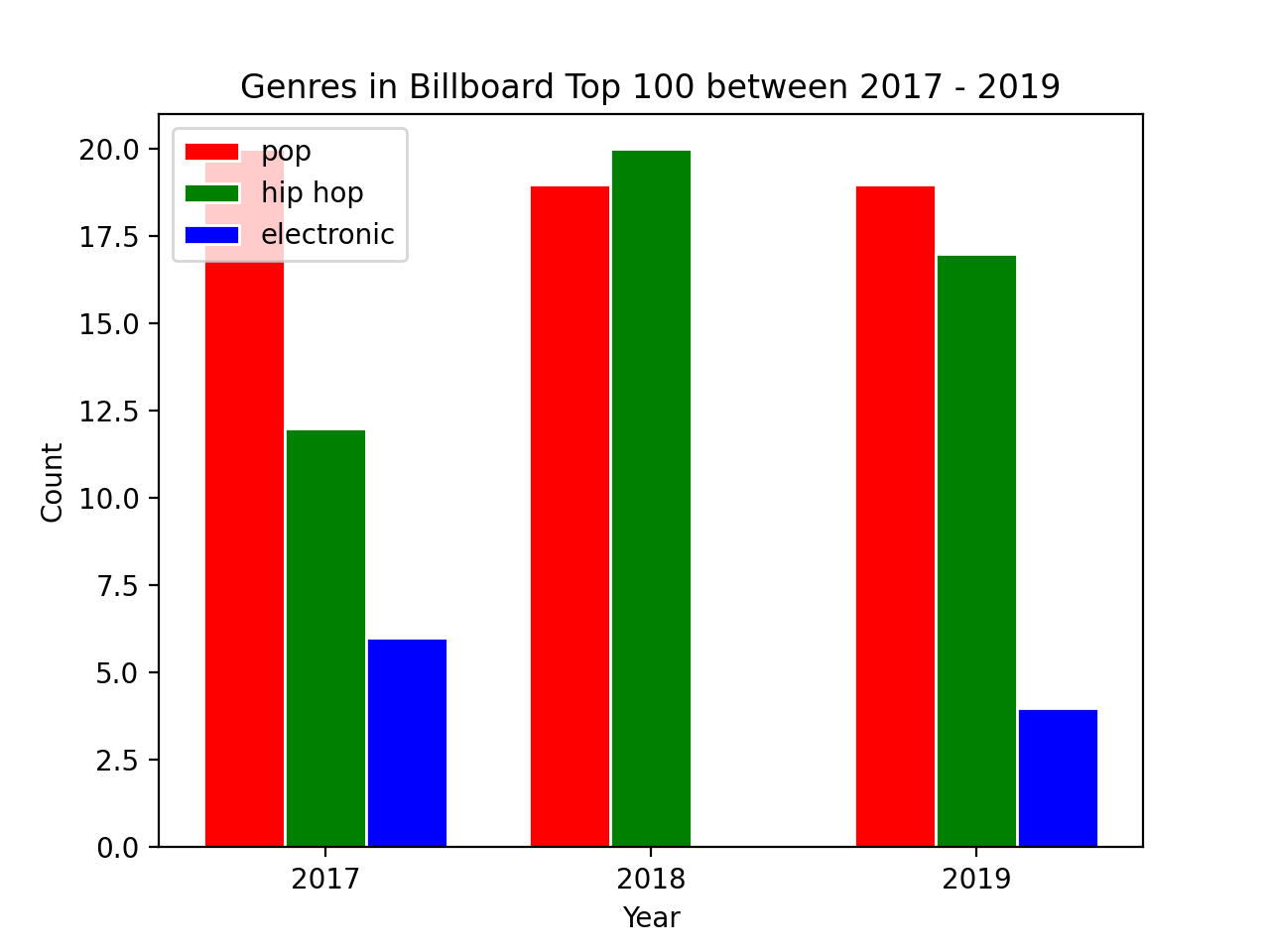


Figure 5. Frequency of Genres in the Billboard Top 100 between 2017 - 2019. This chart takes the top 3 overall genres in the given range and compares them side by side chronologically. The main purpose of this chart is to show the growth over time and help us predict the numbers for 2020. Ideally, we would use linear regression. Note in 2018, there were 0 electronic songs in the top 100.

Since our program is customizable based on user input for a given year range, it was quite a challenge to make the plots account for all the flexibility. That is why we could only display top 3 genres in the bar graph, instead of all of them.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to Run the Code:**

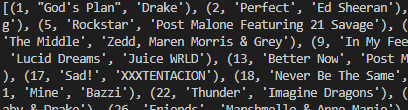
1. Fork the repository from <https://github.com/nakiasw/SI206FinalProject>
2. Run first file with “python3 billboard.py”. You will be prompted to enter your desired 3 year range.
3. Next, run “python3 lastfm.py”. The first run will request all of the data from the API, and since we can only store 20 entries into our database at a time, run “python lastfm.py” until the terminal prints “Last call needed.” That means everything has been stored.
4. Run the last file “python3 visuals.py”. The first figure will appear and won’t let you see the next one until you exit the window. Keep exiting to see each successive figure.
5. Look at your terminal for the 2020 Billboard Top 100 top genre prediction!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Function Documentations:**

**billboard.py**

* titles\_and\_ranking():
  + Prompts the user to input a 3 year range between 2006 and 2019
  + Uses beautiful soup to scrape the charts from Billboard Top 100 for the years requested
  + Returns a list of tuples in the form of (rank, title, artist(s))



* billboardData(info):
  + Inputs the information retrieved from the titles\_and\_ranking function into the database WilkinsSheltonRecords
  + Creates the table ‘Top100’ into the database and store the information from the ‘info’ parameter
  + Does not return anything
* genre\_table():
  + Created a table into the database titled ‘Genres’
  + This table will be the one that stores the information from our API requests

**lastfm.py**

* lastfm\_info():
  + This function accesses the Top100 table in our database to form the requests we send to the LastFM API to get the genres
  + Creates a file called “genres.json” that stores all the API responses in one place so we don’t have to overload the API when we store into our database
  + Returns nothing
* genre\_table():
  + This function keeps track of how many entries are in ‘Genres’ in our database so it knows which 20 to insert next
  + Accesses the API responses from “genres.json”
  + Stores 20 entries into the database

**visuals.py**

* calc\_freq():
  + This function is our main data calculation
  + It joins our two tables in the database, matching the id for a song and genre
  + Then, it runs through the Genre table and groups genres together so our data is more substantial
  + After fetching the joined data, it creates a dictionary that has 3 separate dictionaries for each year, and in each year calculates the frequency of each genre.
  + Stores the data in “data.txt”
  + Returns the aforementioned dictionary
* finalize\_data(dict):
  + Uses the dictionary returned from calc\_freq() to narrow down each year to only the top 4 most popular.
  + Returns a new dictionary that holds top 4 genres for each year
* pie(data, year):
  + This function takes in the dictionary returned from finalize\_data() and a specific year we want to make the pie chart for.
  + Using matplotlib, the function shows the individual year breakdown for top genres (gets called 3 times)
* pie1(data):
  + This function takes in the dictionary returned from finalize\_data() and shows a summary of the overall breakdown of top genres throughout the 4 year range.
* bar1(data):
  + Takes in the dictionary from finalize\_data()
  + This function displays the top 3 genres over the course of the 3 year range to show the growth/trend
  + At the end of the function, it calculates the average number of songs categorized in the top 3 genres over the 3 year span. Whichever is the highest is predicted to be the 2020 top genre.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**External Resources:**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Issue | Resource | Result |
| April 25 | Couldn’t find API that fit our needs | <https://stackoverflow.com/questions/8167322/music-genre-api/8167460> | The Last FM API suggested in the post is ultimately the one we used. Although the code in the post wasn’t very helpful, it at least pointed us towards that API. |
| April 25 | No idea how to use Last FM API | <https://www.last.fm/api/intro> | After reading through the documentation and how to set up the api key, we were still a little lost, but definitely in the right direction. We knew what method we needed to use. |
| April 25 | Not sure how to request from API in python | <https://www.dataquest.io/blog/last-fm-api-python/> | This webpage basically gave a step by step tutorial on how to use Last FM API with python. Super helpful. |
| April 25 | Get size of table in database | <https://www.brentozar.com/archive/2014/02/count-number-rows-table-sql-server/> | This laid out how to use SELECT COUNT(\*) to get the number of entries in a give table. |
| April 26 | Making grouped bar plot | <https://python-graph-gallery.com/11-grouped-barplot/> | Awesome tutorial on how to group bars in matplotlib. Helped us produce the output for figure 5, a main component of the results. |