Briarre Johnson, Morgan Miller

Dr. Barbara Ericson

SI 206

27 April 2021

Final Project Report - SI 206 - MB Productions

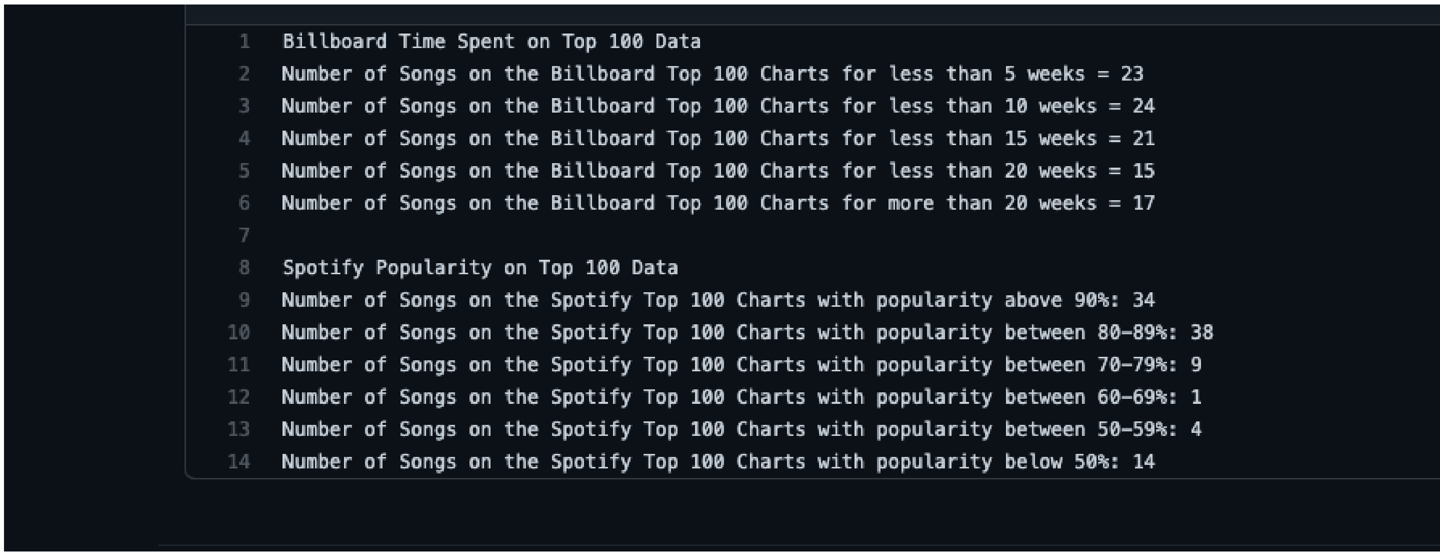
Link to GitHub Repository: https://github.com/briarrej/Final-Project.git

The goal of our project was to collect data from the Hot 100 charts and compare/rank how many weeks songs were on the top 100 charts. In order to achieve this goal, we had to use the Spotify(Spotipy) API and the Billboard website to scrap and collect data. Another goal for our project was to successfully use at least 2 API/websites. Also, we wanted to create 2 visualizations, one for Billboard and one for Spotify. After having the removal of a teammate in our group, our goals shifted to the ones stated above.

The goals that we achieved upon the completion of our project were that we used both 1 API and 1 website to scrap and collect data from. In addition, we created 2 visualizations that represented the average amount of time spent on BillBoard and Spotify Top 100.

One of the problems that we faced was having merge conflicts in our terminals. There would be times where we would get error messages in our terminals and would have to resolve those conflicts. Either, we wouldn't be able to 'git pull' or 'git commit, but we resolved this by quitting the terminal and opening a new terminal window after multiple tries. Another problem that we encountered with Spotify was not having enough data to pull from the playlist because they only had the Top 50 songs. We were able to resolve this issue by pulling data from the Top 50 USA playlists and the Top 50 UK playlists.

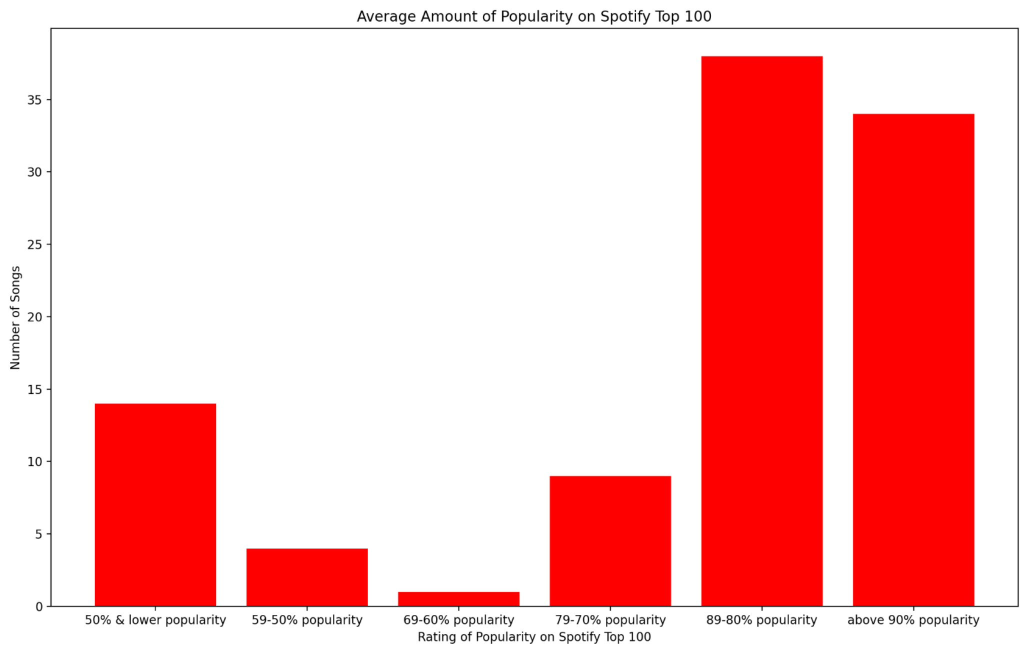
Here is the file that contains our calculations:

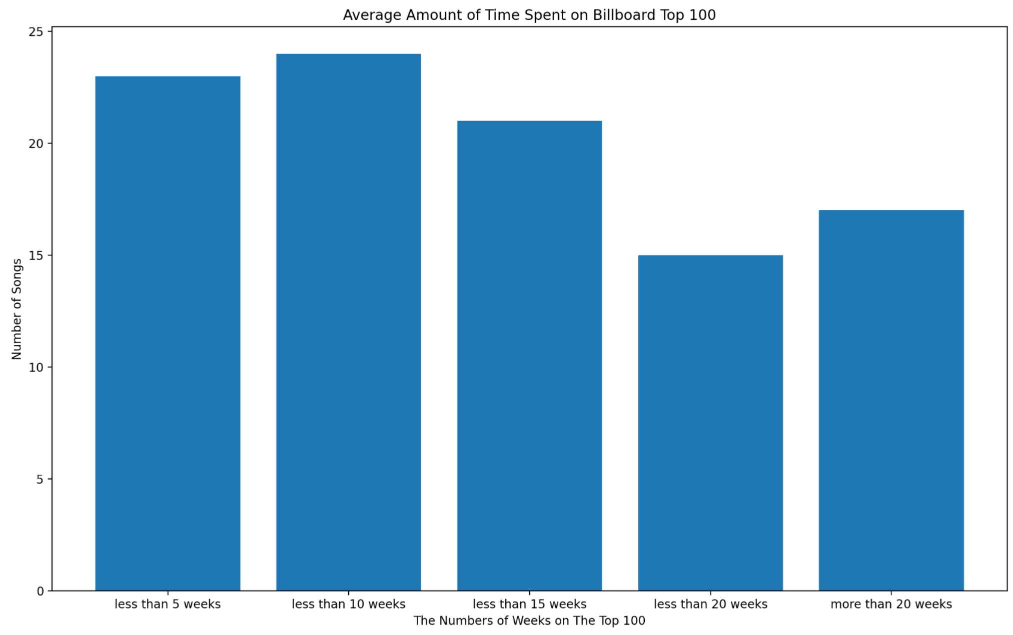


```
Billboard Time Spent on Top 100 Data
Number of Songs on the Billboard Top 100 Charts for less than 5 weeks = 23
Number of Songs on the Billboard Top 100 Charts for less than 10 weeks = 24
Number of Songs on the Billboard Top 100 Charts for less than 15 weeks = 21
Number of Songs on the Billboard Top 100 Charts for less than 20 weeks = 15
Number of Songs on the Billboard Top 100 Charts for more than 20 weeks = 17

Spotify Popularity on Top 100 Data
Number of Songs on the Spotify Top 100 Charts with popularity above 90%: 34
Number of Songs on the Spotify Top 100 Charts with popularity between 80-89%: 38
Number of Songs on the Spotify Top 100 Charts with popularity between 70-79%: 9
Number of Songs on the Spotify Top 100 Charts with popularity between 60-69%: 1
Number of Songs on the Spotify Top 100 Charts with popularity between 50-59%: 4
Number of Songs on the Spotify Top 100 Charts with popularity below 50%: 14
```

Here is a photo of our visualizations:





To run our code, follow these instructions:

1. Open the zip file in Visual Studio Code
2. In finder, make sure that the 'Billboard.db' does not exist. If it does exist, delete the 'Billboard.db' by dragging the file to the trash, so that it can be recreated when the code runs.
3. Make sure that Spotipy (API) is installed on your computer. If it is not installed, you can do so by typing "pip install spotipy" in your terminal and pressing entering.
4. Proceed the open the zip file in VS Code
    a. The first python file that you should open and run is BillBoard_Data.py. This file should be **run 4 times** because each time you run it the database will add 25 songs to the BillBoardSongs table in the database. It will also create the WeeksID table.
    b. The next python file that you should open is Spotify_Data.py. This file should be **run 4 times** because each time you run it the database will add 25 songs to the Spotify table in the database.
    c. Then, you can open the visualization.py file to see the charts we made.
    d. To see our calculations, open the calculations.txt file.

Documentation for our code:
Spotify_Data.py

- Def getSpotifyObject(username, scope):
  - This function allows me to have access to the Spotify data by giving me a token, clientID, and secretID which I acquired from the Developer Spotify website.
- Def user_info:
  - This function enables me to get the information from my current Spotify account, which would later give me access to the playlists.
- Def create_playlist:
  - This function serves as the main function that pulls the data from Spotify. The first for loop in this function gathers the data from the top 50 USA playlist by using the API reference. The second for loop does the same thing but with the top 50 UK playlist API reference. In this function, there are different variables such as song name, artist name, song popularity, and song release date. This
- Def join_table(cur, conn):
  - This function takes in the database connection and cursor as inputs. Also, it returns a list of the results of the songs/artists.
- Def setUpDatabase(db_name):
  - This function takes in the name of a database and a string as input. It takes in and returns the cursor and connection to the database.
- Def createDatabase(cur, conn, spotify):
  - This function takes in the database cursor and connection as inputs. It creates the table called Spotify. It takes all the variables from the create_playlist function and inserts them into the Spotify table in the database.
  - It also adds a counter so that each time the code runs, another 25 songs will be added to the database.
- Def main():
  - This function does not take any inputs and it returns nothing. It calls the functions spotify = getSpotifyObject, cur, conn = setUpDatabase('Billboard.db'), createDatabase(cur, conn, spotify), playlist_songs = create_playlist(spotify), spotify_viz_chart(playlist_songs). Then it closes the database connection by using conn.close().

3Billboard.py

- Def getBillBoardLink():
  - This function uses BeautifulSoup to scrape the Billboard Top 100 songs website for the week of April 17, 2021. The function grabs the song title, artist name, ranking of the song, and the number of weeks it has been on the Top 100 chart. This function also creates a system for the number of weeks the songs have been on the charts. For example, if the song has been on the chart for less than 5 weeks, the song category for that song would be 1. This function returns a list of song titles, artist names, song rankings, the number of weeks on the Top 100 chart, and the category that the number of weeks on the Top 100 chart corresponds to.
- Def setUpDatabase(db_name):
  - This function takes in the name of a database, which will be a string as input. It connects to sqlite3 and returns the cursor and connection to the database.
- Def createDatabase (cur, conn, startIndex):
  - This function takes in cur, conn, and startIndex. It calls on getBillBoardLink() and assigns it to the variables song, artist, ranking, weeksOnTop100, songCategory. Then it runs a for loop for items in the range of the startIndex - startIndex+25 and it inserts the song name, artist, ranking, and weeks_id into a new row for each item in the range.
- Def createDatabase2 (cur, conn):
  - This function takes in cur and conn as parameters. A list called stringWeeks is created to serve as the description for the song categories. There are only 5 categories that weeks in the top 100 can fall into which are less than 5 weeks (week_id = 1), less than 10 weeks (week_id = 2), less than 15 weeks (week_id = 3), less than 20 weeks (week_id = 4), and more than 20 weeks (weeks_id = 5). This table sets the maxNum to 5 so if any row is over 5, the function will pass and not create more rows. If the maxNum is lower than 5, then a row with the category number and the category meaning is created.
- Def main():
  - This function does not take in any inputs, but it is used to create the database, BillBoard.db. It is also used to create the BillBoardSongs table, the WeeksID table, and to join those two tables together (BillBoardSongs and WeeksID). The parameter startIndex is also assigned in this function. When it is called, this function creates our database.

Visualizations.py

- Def set_connection(db_file):
  - This function takes in a db_file, and it sets up the connection to SQLite and returns conn.
- Def get_weeks_popularity(conn):

    - This function takes in conn and it sets up the week dictionary for how many songs fall into a certain song category based on the number of weeks it was on the chart.
- Def viz(data):
    - This function displays the Billboard chart data into a bar graph with the amount of time (in weeks) that the songs were on the top100 chart for.
- Def get_song_pop(con):
    - This function takes in conn and it creates a dictionary based on the song popularity percentage on the Spotify Top 100 Playlist. If a song is above 90% popularity, the dictionary value for the 'above 90% popularity' key will increase by 1. This will repeat for the other values listed. This goes on for each song's popularity and the function returns song_pop_dict, which is the dictionary with all of the song's popularity level.
- Def spotify_viz_chart(spot_data):
    - This function displays the Spotify chart data into a bar graph with the range of the percentage of popularity that the songs have in the Top 100 Playlist.

Date	**Issue Description**	**Location of Resource**	**Result**
4/19/2021	**Having merge issue conflicts with our git repositories**	**https://stackoverflow.com/questions/161813/how-to-resolve-merge-conflicts-in-git-repository**	**We were able to resolve our merge conflict issue within the terminal**
Multiple Dates 4/17-4/20/2021	**Unfamiliar with the Spotipy API**	**https://spotipy.readthedocs.io/en/2.18.0/#api-reference**	**Yes, we were able to learn more about the Spotipy API**
4/18/2021	**I didn't know how to get access to spotify**	**https://stmorse.github.io/journal/spotify-api.html**	**Yes, I was able to get my token access and use Developer Spotify**
4/27/2021	**Struggling to grab all of the data from a specific row in the table.**	**https://pynative.com/python-cursor-fetchall-fetchmany-fetchone-to-read-rows-from-table/**	**This website helped me better understand fetchall() and use it in my visualization.py to grab all of the items for my graphs and calculations.**