SI 206 Final Project Report
Music-Weather Analyzer

Aditya Bhatt (bhattadi), Anirudh Lagisetty (alagiset), Ashka Pujara (ashpuj)

Repository Link: https://github.com/bhattadi/Weather-Music-Analyzer

Project Goals

Our initial project goals included the following:

- Using the New York Times and Dark Sky API's to see what the weather was like during certain world events.
- Use the Spotify API to find what music was popular during the corresponding times
- In summary: Is there a trend in what music people listen to as the weather and world events change?

Goals Achieved

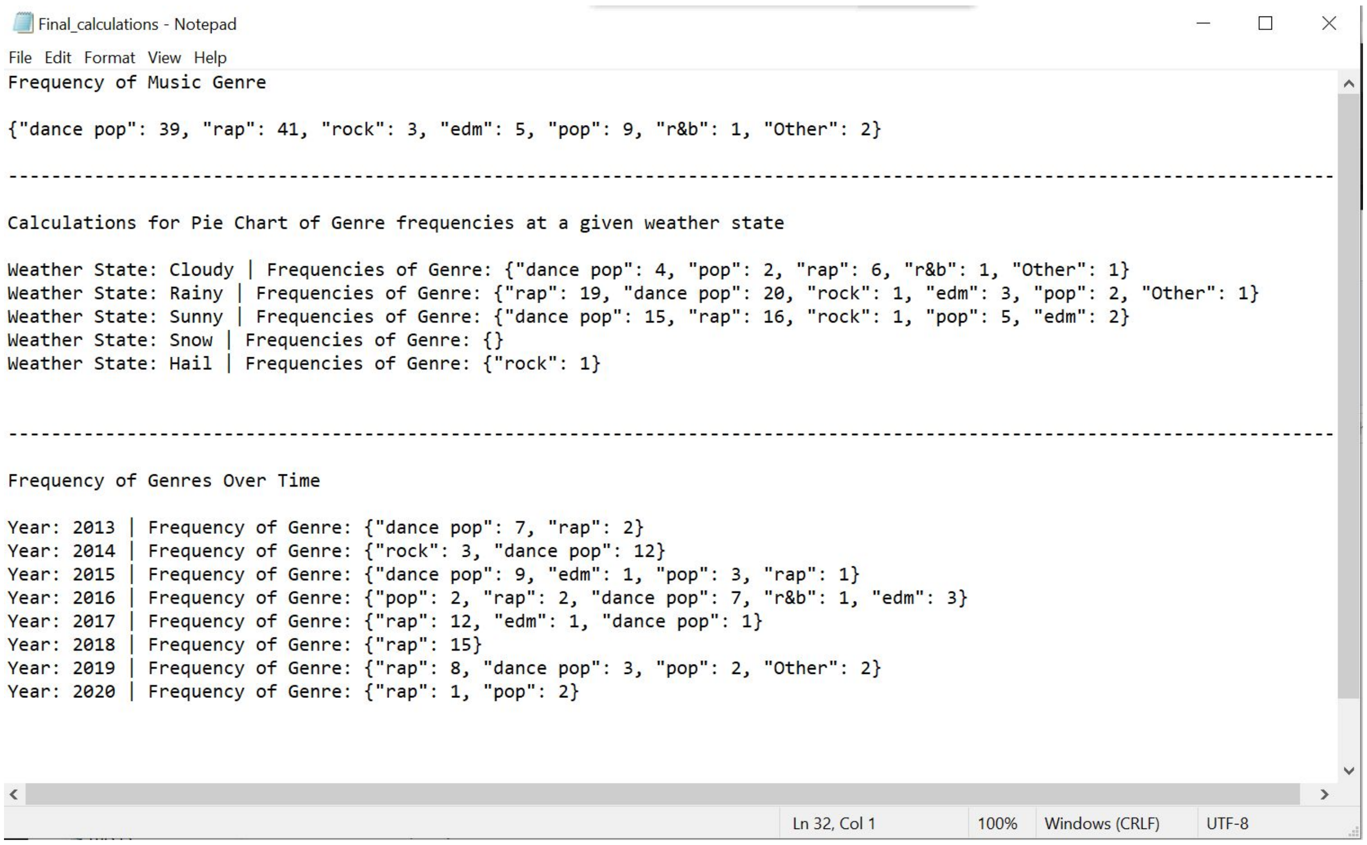
Due to the data available to retrieve from API's (specifically Spotify), we had to change our project plan slightly. Ultimately, we were able to achieve the following:

- Used the Spotipy, MetaWeather, and Billboard Charts API's to identify the genres of music that are most popular by date, as well as the corresponding weather temperature and condition
- Calculated the most common song genre per day accessed
- Created ___ visualizations to demonstrate the relationships observed between weather conditions (ex: cloudy, rainy, sunny), most popular genres of music, and time.

Problems Faced

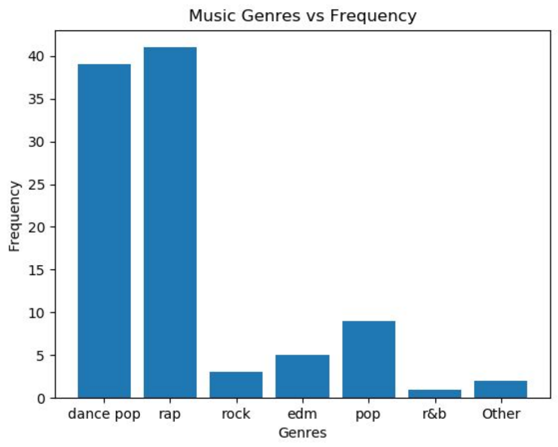
- As stated above, the Spotipy API did cause us some trouble. We had thought that we could see what users were listening to on a specific day, but that type of data was not available to retrieve. So, we instead used the Billboard API to find similar information which ultimately worked out quite well.
- We struggled a little bit with understanding how to only retrieve 20 items at a time to insert into our database. It didn't take too long to figure out how to do it, but the implementation ended up being quite tricky and it took us a while to debug our initial attempt. However, once we got it, it was quite easy to replicate in all the other functions.

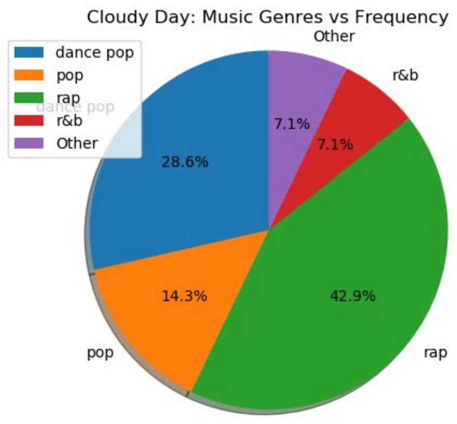
Calculation File

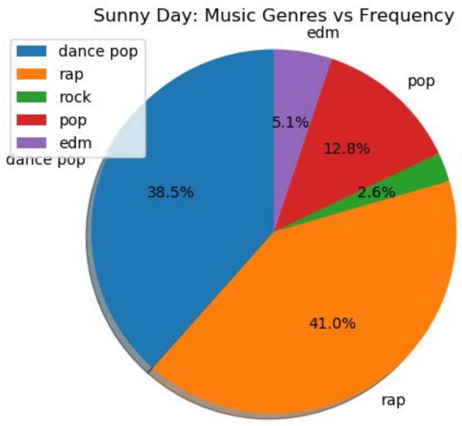


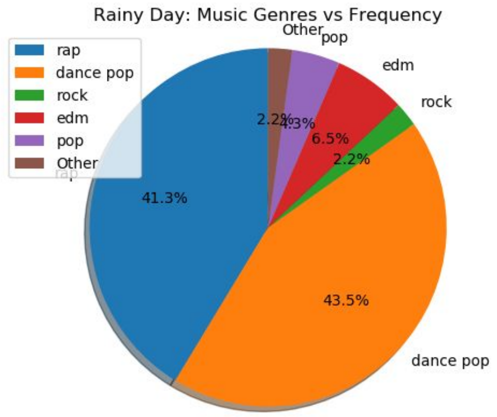
```
Frequency of Music Genre

{"dance pop": 39, "rap": 41, "rock": 3, "edm": 5, "pop": 9, "r&b": 1, "Other": 2}

------------------------------------------------------------------------------------------------------------------------

Calculations for Pie Chart of Genre frequencies at a given weather state

Weather State: Cloudy | Frequencies of Genre: {"dance pop": 4, "pop": 2, "rap": 6, "r&b": 1, "Other": 1}
Weather State: Rainy | Frequencies of Genre: {"rap": 19, "dance pop": 20, "rock": 1, "edm": 3, "pop": 2, "Other": 1}
Weather State: Sunny | Frequencies of Genre: {"dance pop": 15, "rap": 16, "rock": 1, "pop": 5, "edm": 2}
Weather State: Snow | Frequencies of Genre: {}
Weather State: Hail | Frequencies of Genre: {"rock": 1}


------------------------------------------------------------------------------------------------------------------------

Frequency of Genres Over Time

Year: 2013 | Frequency of Genre: {"dance pop": 7, "rap": 2}
Year: 2014 | Frequency of Genre: {"rock": 3, "dance pop": 12}
Year: 2015 | Frequency of Genre: {"dance pop": 9, "edm": 1, "pop": 3, "rap": 1}
Year: 2016 | Frequency of Genre: {"pop": 2, "rap": 2, "dance pop": 7, "r&b": 1, "edm": 3}
Year: 2017 | Frequency of Genre: {"rap": 12, "edm": 1, "dance pop": 1}
Year: 2018 | Frequency of Genre: {"rap": 15}
Year: 2019 | Frequency of Genre: {"rap": 8, "dance pop": 3, "pop": 2, "Other": 2}
Year: 2020 | Frequency of Genre: {"rap": 1, "pop": 2}
```

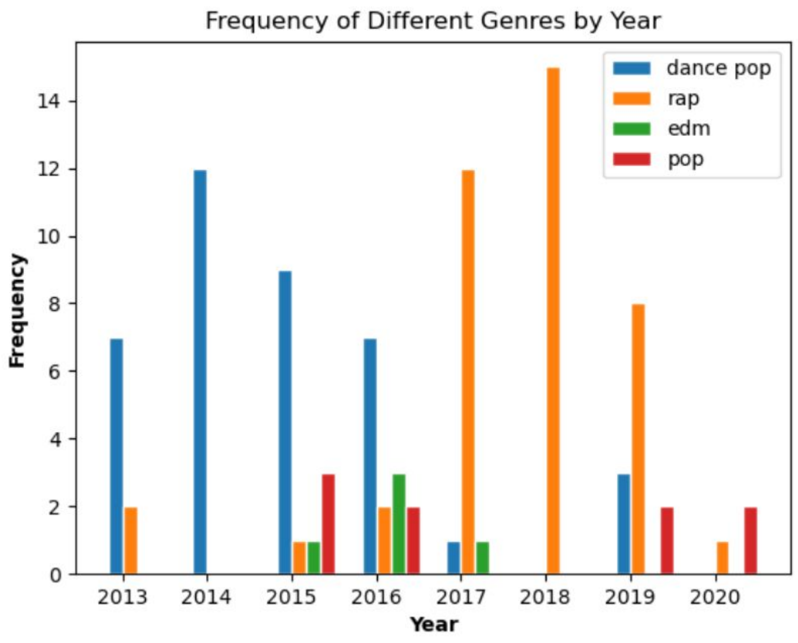
Visualizations











Instructions to Run Code
There are 2 files:

1. data_collection.py
2. driver.py

To run the code, run the data_collection.py file 5 times. Then, run the driver.py file once. Once both files have been run, open the bar graph and pie charts created that are saved in your folder (same file path as the two Python files).

Function Documentation Documentation provided in comments throughout the two Python files (data_collection.py and driver.py)

data_collection.py

```
# Initial database setup function to allow the program to be
able to find the file in the OS
# Input: database name (string)
# Output: cur and conn (database cursor and connection)
def setUpDatabase(db_name)

# Creates the weather temperature table, weather conditions
table, the top Billboards table, and Genres table
# Input: database cursor and connection
# Output: none; creates database tables
def setUpTables(cur, conn)

# Call the weather API
# Create a table with dates and weather data
# Input: database cursor and connection
# Output: Sets up weather tables with temperature/condition by
date
def setUpWeather(cur, conn)

# Call the billboards API
# Create a table with dates and top hits
# Input: The cur and conn connections to the final database
# Output: Table filled with information about top number of
songs on a given range of dates
def setUpBillBoards(cur, conn)
```

```
# Find corresponding Genres for the songs
# Input: The cur and conn connections to the final database
# Output: Table filled with information about what the top
genre of the day is for a range of dates
def setUpGenres(cur, conn)
```

driver.py

```
#Using SQL joins to find corresponding genres for a particular
date
#Using matplotlib to produce bar graphs and pie charts
# -------------- Visualization #1---------------------------
# Input: The cur and conn connections to the final database
# Output: Bar Graphs for frequency of different genres
def barGraph(cur, conn)

# ------------------ Visualizations #2-#4 --------------------
# Input: Database cursor and connection
# Output: Produces 5 pie charts corresponding to each type of
weather condition and the frequency of genres on that type of
day
def pieCharts(cur, conn)

# ------------------ Visualization #5 ------------------------
# Input: Cur and conn of the database
# Output: A bar graph analyzing the change in music taste in
terms of genres across the years
def yearsAnalysisOfGenre(cur, conn)

# Input: None
# Output: Written file with calculations for all visualizations
def writeToFile()
```

Resources Used

Date	Issue Description	Location of Resource	Result
4/26/20	Trouble using JOIN sql command to bring genre	https://stackoverflow.com/questions/1	We successfully executed the JOIN

	and date tuples	9270259/update-with-join-in-sqlite	command and retrieved data from both tables
4/26/20	Had trouble figuring out how to format a pie chart.	https://matplotlib.org/3.1.1/gallery/pie_and_polar_charts/pie_features.html	Were able to create our pie chart and format as needed.
4/24/20	Sqlite3 module was failing to be recognized by one of our computers even though it had been installed.	https://stackoverflow.com/questions/54876404/unable-to-import-sqlite3-using-anaconda-python	Was able to use the instructions in the link to run the code without error.
4/25/20	Deciphering the genre from a Spotify object was difficult to find after parsing the returned JSON	https://developer.spotify.com/dashboard/	We were able to pull the genre field from the Album data of a song by using the Search function
4/23/20	Visual Studio Code Liveshare link crashed randomly and it was difficult to view the host's code edits live	https://docs.microsoft.com/en-us/visualstudio/liveshare/troubleshooting	The file path was undefined. Once that path was defined the liveshare worked normally
4/25/20	Calling Spotify API required authentication key but the key required a Spotify account and did not give guest access	https://developer.spotify.com/dashboard/	We created a new spotify account so we can use the Spotify authentication keys that allow user to call API functions
4/27/20	Had a lot of trouble trying to format our multiple bar graph so it looked neat without overlapping bars	https://python-graph-gallery.com/11-grouped-barplot/	We were able to create a multiple graph with all the bars aligned
	No issue- Spotipy API documentation	https://spotipy.readthedocs.io/en/2.12.0/	
	No issue- Billboard API documentation	https://github.com/guoguo12/billboard	

		-charts	
	No issue- MetaWeather API documentation	https://www.metaweather.com/api/	

Bonus B: Additional Visualizations

Refer to Visualizations section on page 2 for additional figures beyond the required 3.