Final Project Reflection

Yuna Takahashi, Yuri Chen, Abby Sypniewski

Link: https://github.com/abbysyp/si206_final_proj.git

Goals for the Project

The biggest objective for this project was to visualize, if any, the relationship between tempo, danceability, speechiness, and liveliness of a song and its popularity to understand what makes a song likable. In order to accomplish this, we looked at two music websites to pull the top 25 songs from 4 countries – US, UK, Canada, and France – before obtaining information on the music features for each song from the Spotify API. Then, we took the average of each music feature of the 8 songs in each ranking (e.g. average the tempo of rank #1 from US, UK, Canada, and France). This allowed our data to reflect the tendencies and preferences of music listeners around the world and have many data points.

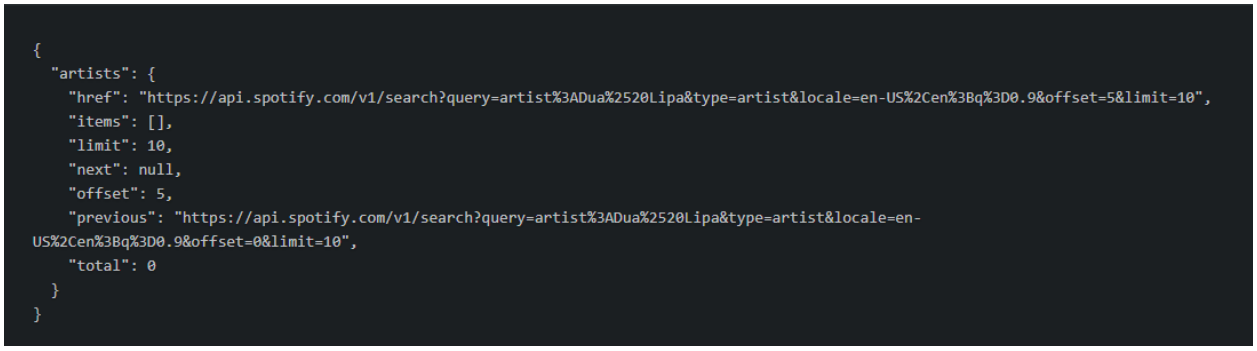
Goals Achieved

When visualizing the average Spotify statistics for each ranking in different formats—such as a scatter plot, line graph, and bar chart—we found that there was little to no correlation between the where the songs charted and their average tempo, speechiness, danceability, etc. For danceability, the line plot shows somewhat of a relationship: it appears as though songs with lower rankings (i.e., less popularity) have lower peaks for their danceability scores. However, this connection did not seem to be strong enough to draw any causal conclusions. We achieved the answer to our question by discovering that, surprisingly, there are more factors than what we considered which go into a hit or trending song. If we were to do this project again, however, we might collect the top 100 songs to see if there is a larger picture correlation.

Problems Faced

The first problem we ran into was finding an API source that provided the ranking information we needed while having an attainable token/key. For example, although our initial plan talked about using Apple Music API, after research we realized that it requires a paid membership in order to acquire the necessary token. As a solution, we were able to find an API from Deezer and figure out steps to acquire a token that is valid for one hour.

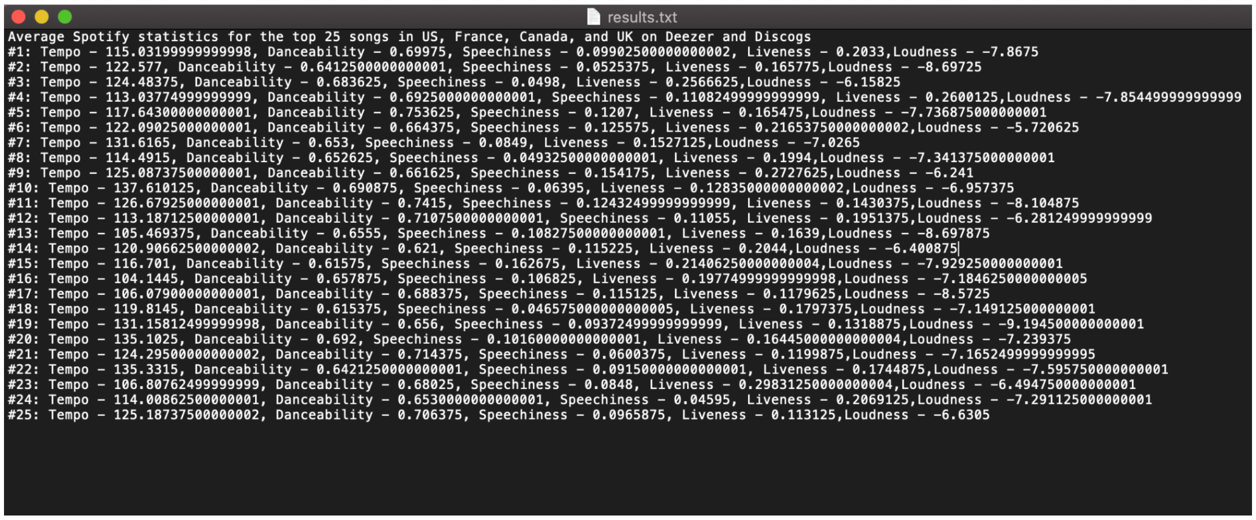
Another problem we faced while working with the Spotify API is using the database we had currently to search for the song features on the Spotify API. Spotify only allows users to look for songs by using their unique IDs. This presented a lot of constraints for our project, as we were unable to find these song IDs through the song titles and artist names stored in our database. We attempted many ways and researched different options to get Spotify song IDs. The only seemingly feasible option is using their "GET /search" request that gets Spotify catalog information by matching a keyword string. However, we then realized that this method only works when we search a song title or an artist name on its own, and does not work when we try to look up a specific song by a specific artist. When we attempted to search both at the same time, our results all ended up similar to the picture below.



```
{
  "artists": {
    "href": "https://api.spotify.com/v1/search?query=artist%3ADua%2520Lipa&type=artist&locale=en-US%2Cen%3Bq%3D0.9&offset=5&limit=10",
    "items": [],
    "limit": 10,
    "next": null,
    "offset": 5,
    "previous": "https://api.spotify.com/v1/search?query=artist%3ADua%2520Lipa&type=artist&locale=en-
US%2Cen%3Bq%3D0.9&offset=0&limit=10",
    "total": 0
  }
}
```

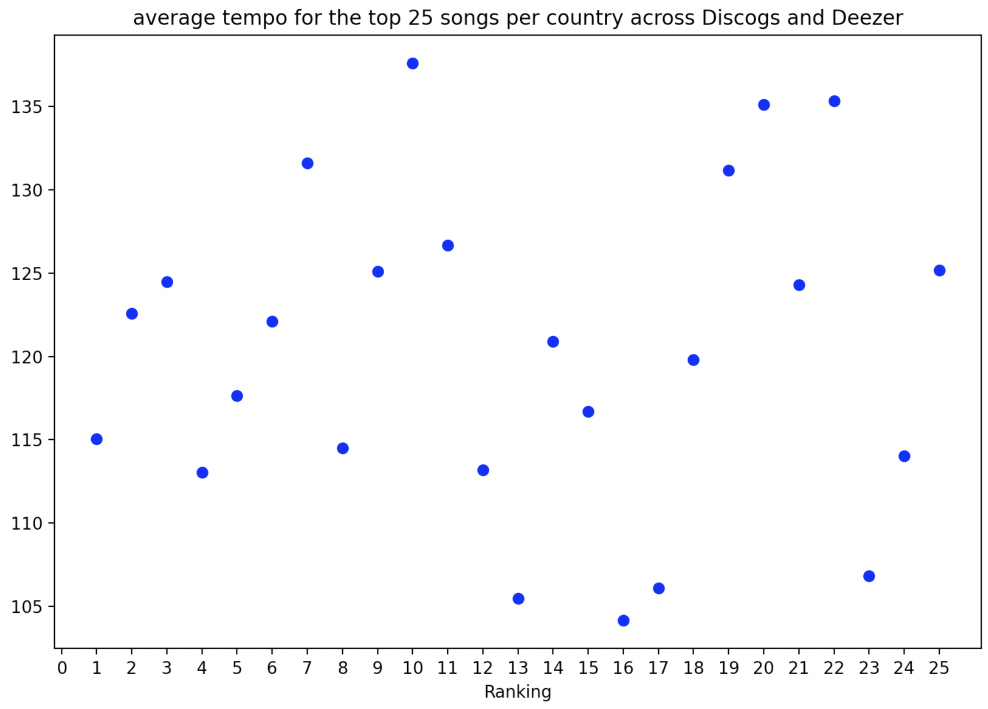
As shown, the result does not return the information about the songs or the song IDs. Because of this, we have to look into other ways to get these song IDs. We resorted to using playlist IDs to find individual songs' ID, then looked into their feature information. And in order to use these playlists, we used the Discogs and Deezer database to create playlists for top 25 songs in 4 different countries.

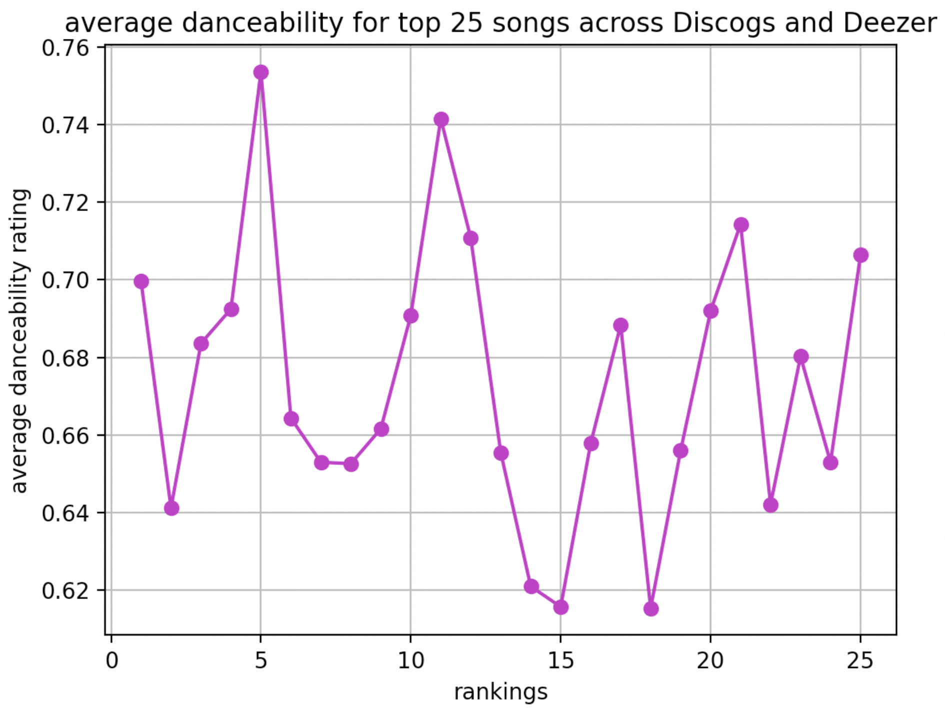
Calculations

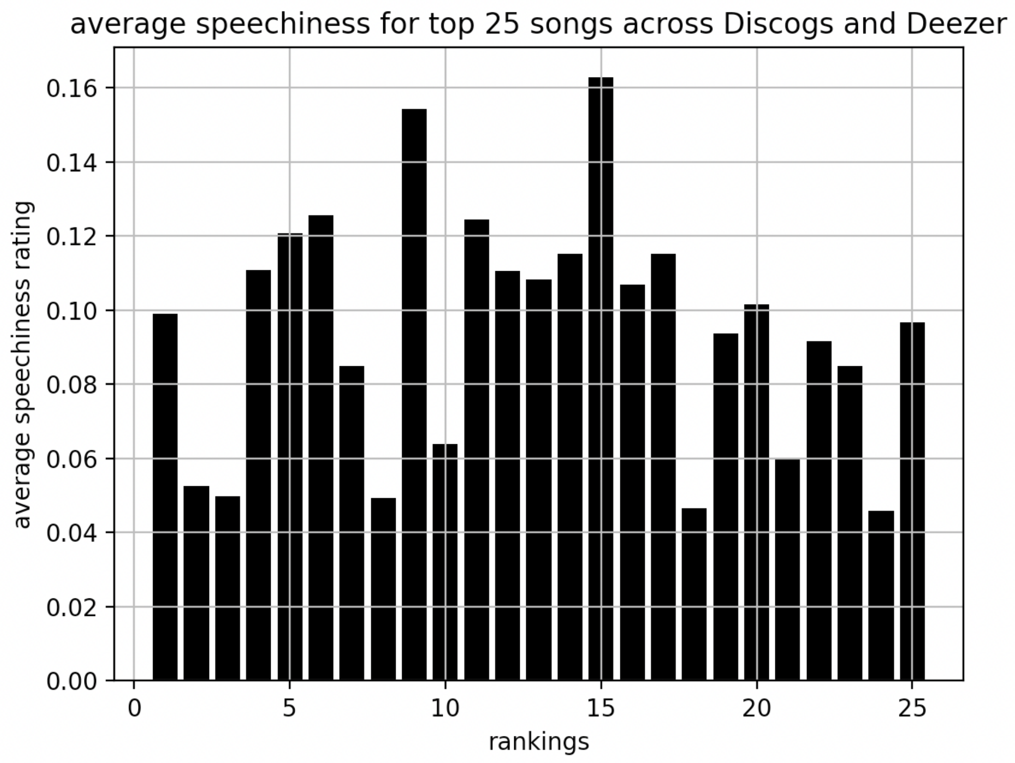


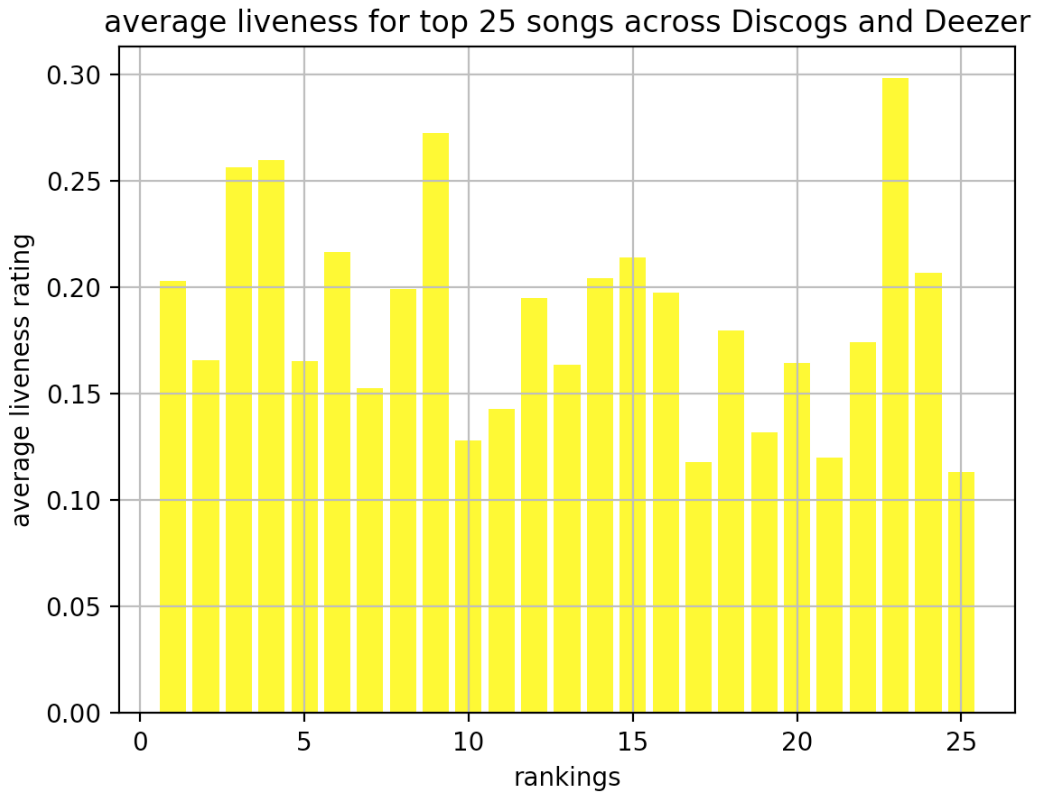
```
results.txt
Average Spotify statistics for the top 25 songs in US, France, Canada, and UK on Deezer and Discogs
#1: Tempo - 115.03199999999998, Danceability - 0.69975, Speechiness - 0.09902500000000002, Liveness - 0.2033,Loudness - -7.8675
#2: Tempo - 122.577, Danceability - 0.6412500000000001, Speechiness - 0.0525375, Liveness - 0.165775,Loudness - -8.69725
#3: Tempo - 124.48375, Danceability - 0.683625, Speechiness - 0.0498, Liveness - 0.2566625,Loudness - -6.15825
#4: Tempo - 113.03774999999999, Danceability - 0.6925000000000001, Speechiness - 0.11082499999999999, Liveness - 0.2600125,Loudness - -7.854499999999999
#5: Tempo - 117.64300000000001, Danceability - 0.753625, Speechiness - 0.1207, Liveness - 0.165475,Loudness - -7.736875000000001
#6: Tempo - 122.09025000000001, Danceability - 0.664375, Speechiness - 0.125575, Liveness - 0.21653750000000002,Loudness - -5.720625
#7: Tempo - 131.6165, Danceability - 0.653, Speechiness - 0.0849, Liveness - 0.1527125,Loudness - -7.0265
#8: Tempo - 114.4915, Danceability - 0.652625, Speechiness - 0.04932500000000001, Liveness - 0.1994,Loudness - -7.341375000000001
#9: Tempo - 125.08737500000001, Danceability - 0.661625, Speechiness - 0.154175, Liveness - 0.2727625,Loudness - -6.241
#10: Tempo - 137.610125, Danceability - 0.690875, Speechiness - 0.06395, Liveness - 0.12835000000000002,Loudness - -6.957375
#11: Tempo - 126.67925000000001, Danceability - 0.7415, Speechiness - 0.12432499999999999, Liveness - 0.1430375,Loudness - -8.104875
#12: Tempo - 113.18712500000001, Danceability - 0.7107500000000001, Speechiness - 0.11055, Liveness - 0.1951375,Loudness - -6.281249999999999
#13: Tempo - 105.469375, Danceability - 0.6555, Speechiness - 0.10827500000000001, Liveness - 0.1639,Loudness - -8.697875
#14: Tempo - 120.90662500000002, Danceability - 0.621, Speechiness - 0.115225, Liveness - 0.2044,Loudness - -6.400875
#15: Tempo - 116.701, Danceability - 0.61575, Speechiness - 0.162675, Liveness - 0.21406250000000004,Loudness - -7.929250000000001
#16: Tempo - 104.1445, Danceability - 0.657875, Speechiness - 0.106825, Liveness - 0.19774999999999998,Loudness - -7.184625000000005
#17: Tempo - 106.07900000000001, Danceability - 0.688375, Speechiness - 0.115125, Liveness - 0.1179625,Loudness - -8.5725
#18: Tempo - 119.8145, Danceability - 0.615375, Speechiness - 0.046575000000000005, Liveness - 0.1797375,Loudness - -7.149125000000001
#19: Tempo - 131.15812499999998, Danceability - 0.656, Speechiness - 0.09372499999999999, Liveness - 0.1318875,Loudness - -9.194500000000001
#20: Tempo - 135.1025, Danceability - 0.692, Speechiness - 0.10160000000000001, Liveness - 0.16445000000000004,Loudness - -7.239375
#21: Tempo - 124.29500000000002, Danceability - 0.714375, Speechiness - 0.0600375, Liveness - 0.1199875,Loudness - -7.165249999999995
#22: Tempo - 135.3315, Danceability - 0.6421250000000001, Speechiness - 0.09150000000000001, Liveness - 0.1744875,Loudness - -7.595750000000001
#23: Tempo - 106.80762499999999, Danceability - 0.68025, Speechiness - 0.0848, Liveness - 0.29831250000000004,Loudness - -6.494750000000001
#24: Tempo - 114.00862500000001, Danceability - 0.6530000000000001, Speechiness - 0.04595, Liveness - 0.2069125,Loudness - -7.291125000000001
#25: Tempo - 125.18737500000002, Danceability - 0.706375, Speechiness - 0.0965875, Liveness - 0.113125,Loudness - -6.6305
```

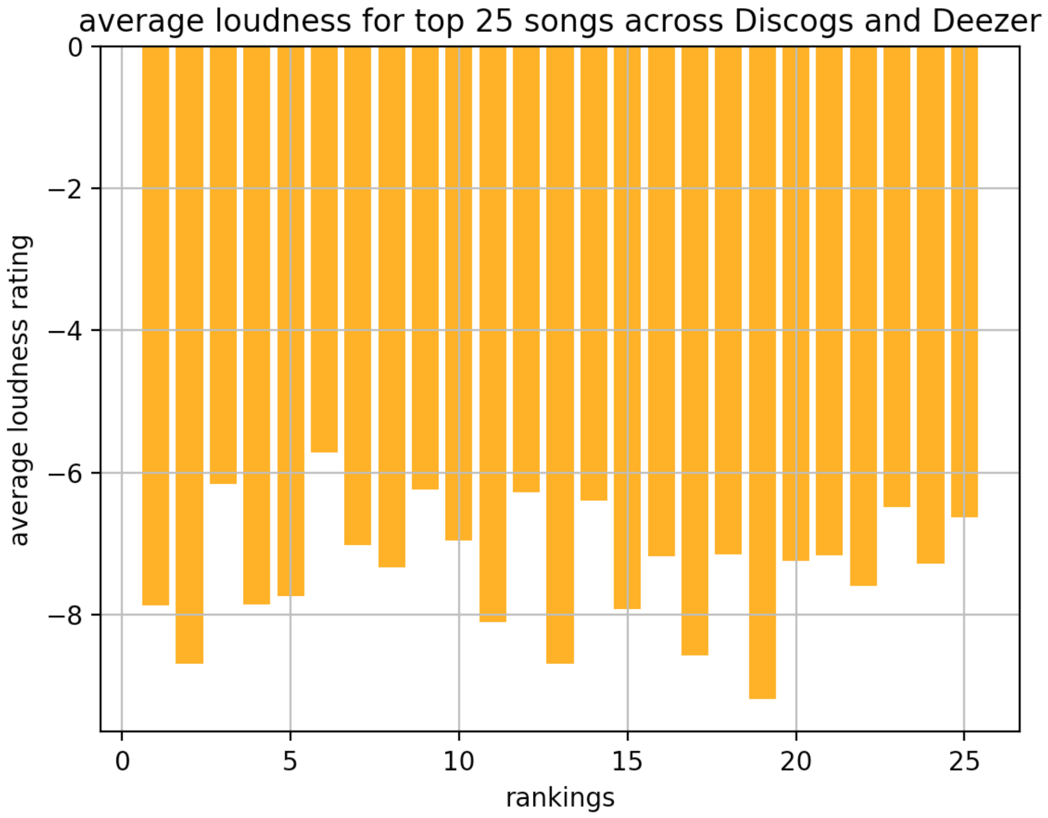
Visualizations






average speechiness for top 25 songs across Discogs and Deezer
average speechiness rating
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0
5
10
15
20
25
rankings


average liveness for top 25 songs across Discogs and Deezer
average liveness rating
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
5
10
15
20
25
rankings


average loudness for top 25 songs across Discogs and Deezer
average loudness rating
rankings
0
−2
−4
−6
−8
0
5
10
15
20
25

Instructions for Running the Code

Deezer25.py

1. Start by creating necessary tokens/codes.
    a. Go to https://connect.deezer.com/oauth/auth.php?app_id=YOUR_APP_ID&redirect_uri=YOUR_REDIRECT_URI&perms=basic_access,email
    b. User needs to create his/her own appId and redirectURL and enter them into YOUR_APP_ID and YOUR_REDIRECT_URL in the above url before pressing enter; however, for convenience I have provided my code if necessary.

        (Yuna's appId:537442, redirectURL: https://theappreciationengine.com/DeezerAuthenticator_Controller)

    c. Click "continue" if prompted yunat account
    d. On the url bar, find the code="**"
        i. (** is the auth code)
    e. On the url bar, type

        https://connect.deezer.com/oauth/access_token.php?app_id=537442&secret=4d5f70ae842906d48ae6521d946d5654&code=**

        i. (For the **, enter the auth code from the step above)
    f. You will get the access_token. This is valid for 3600 seconds. Copy and paste the access token onto line 28.
2. Run the code 4 times to get 100 total songs from 4 different countries in Deezer API inserted in the db

Discogs25.py

1. Run the code 4 times to get 100 total songs from 4 different countries in Discogs API inserted in the db

Project.py

1. Install Spotipy (a python library for Spotify Web API) in terminal by typing “pip3 install spotipy”
2. Receive Client ID and Client Secret from “Spotify for developers” account, and replace current Client ID and Client Secret with your own on line 11 and 12
3. **Uncomment** Additional Section in main if necessary to clear existing database
4. **Comment** out Additional Section
5. Run the code 8 times to get all the song features from all 200 songs in the database inserted in the db

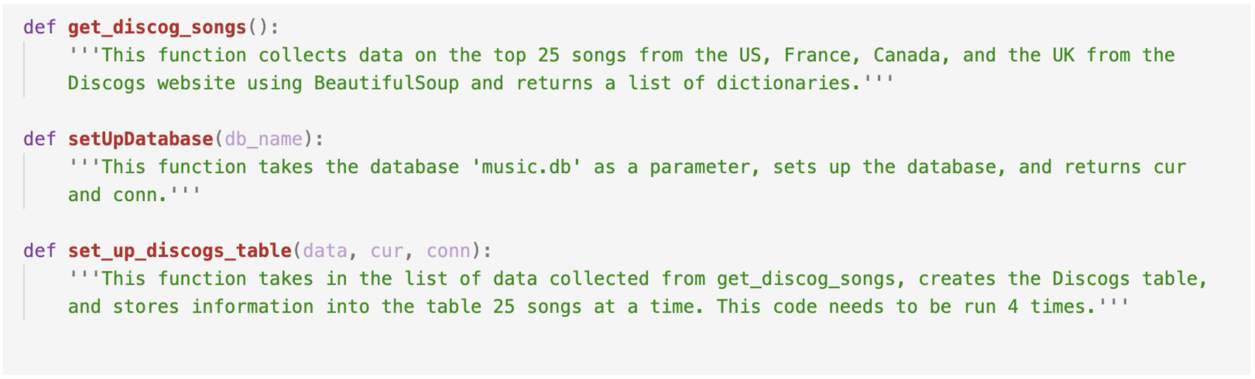
6. **Uncomment** Section 2 (lines 165 - 168) to join all 3 database, calculate the averages, and insert the averages to “Averages” table in db and write a text file of calculated data

Graph.py

1. Run the code to see the visualizations

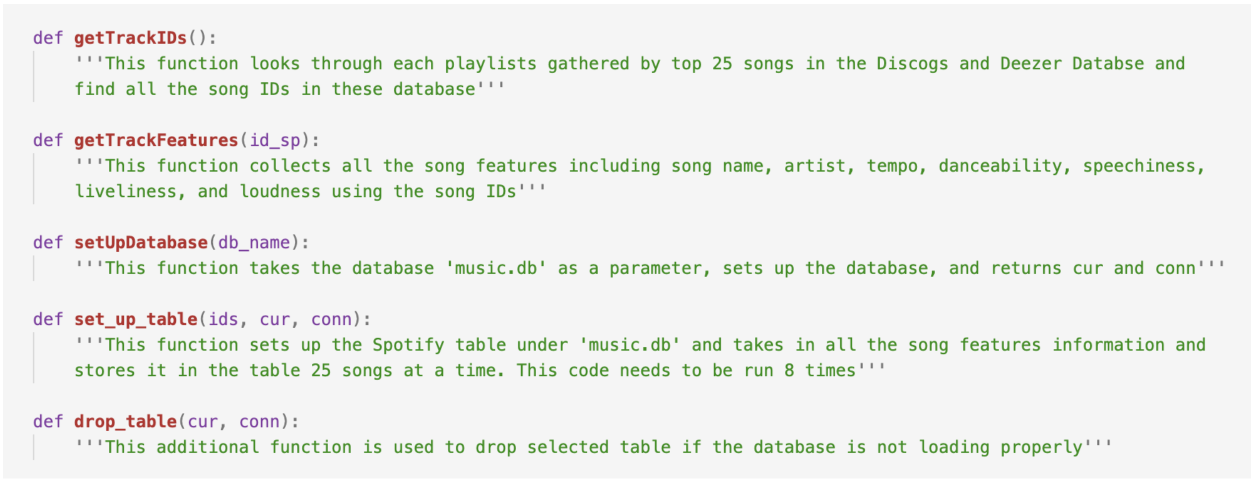
Documentation

Discogs25.py

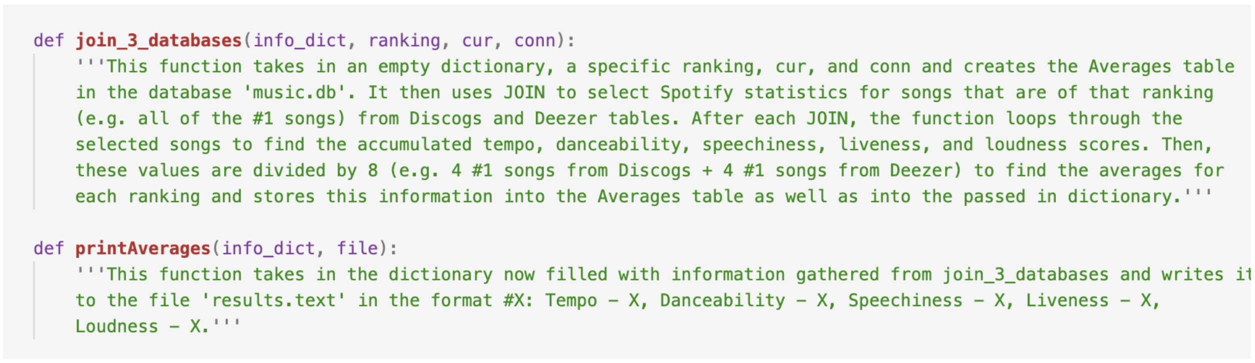


```
def get_discog_songs():
    '''This function collects data on the top 25 songs from the US, France, Canada, and the UK from the
    Discogs website using BeautifulSoup and returns a list of dictionaries.'''

def setUpDatabase(db_name):
    '''This function takes the database 'music.db' as a parameter, sets up the database, and returns cur
    and conn.'''

def set_up_discogs_table(data, cur, conn):
    '''This function takes in the list of data collected from get_discog_songs, creates the Discogs table,
    and stores information into the table 25 songs at a time. This code needs to be run 4 times.'''
```

Project.py

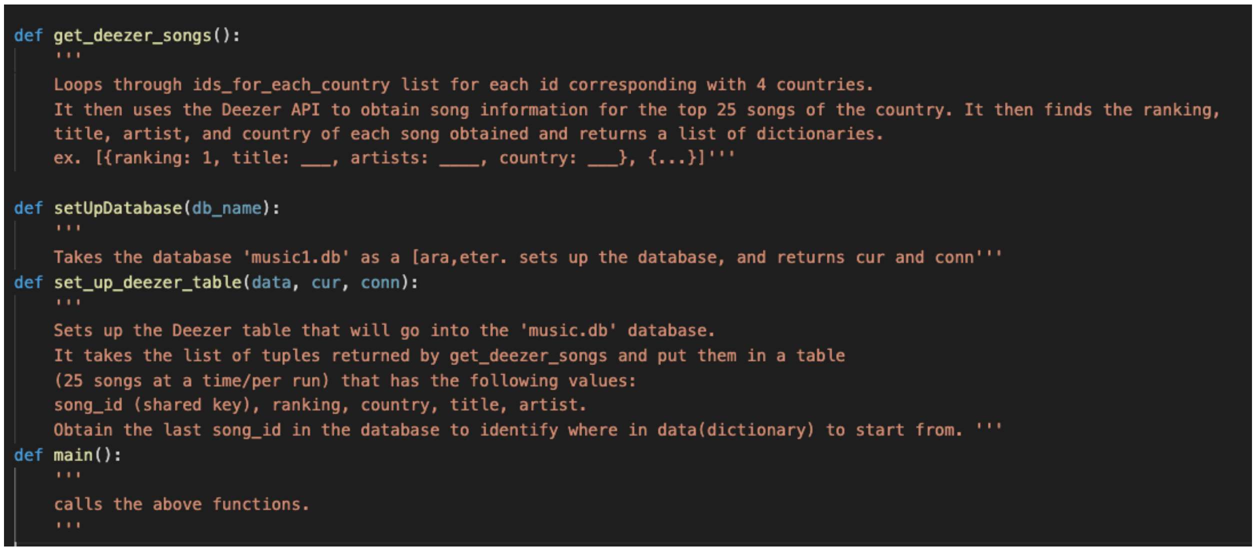


```
def getTrackIDs():
    '''This function looks through each playlists gathered by top 25 songs in the Discogs and Deezer Databse and
    find all the song IDs in these database'''

def getTrackFeatures(id_sp):
    '''This function collects all the song features including song name, artist, tempo, danceability, speechiness,
    liveliness, and loudness using the song IDs'''

def setUpDatabase(db_name):
    '''This function takes the database 'music.db' as a parameter, sets up the database, and returns cur and conn'''

def set_up_table(ids, cur, conn):
    '''This function sets up the Spotify table under 'music.db' and takes in all the song features information and
    stores it in the table 25 songs at a time. This code needs to be run 8 times'''

def drop_table(cur, conn):
    '''This additional function is used to drop selected table if the database is not loading properly'''
```



```
def join_3_databases(info_dict, ranking, cur, conn):
    '''This function takes in an empty dictionary, a specific ranking, cur, and conn and creates the Averages table
    in the database 'music.db'. It then uses JOIN to select Spotify statistics for songs that are of that ranking
    (e.g. all of the #1 songs) from Discogs and Deezer tables. After each JOIN, the function loops through the
    selected songs to find the accumulated tempo, danceability, speechiness, liveness, and loudness scores. Then,
    these values are divided by 8 (e.g. 4 #1 songs from Discogs + 4 #1 songs from Deezer) to find the averages for
    each ranking and stores this information into the Averages table as well as into the passed in dictionary.'''

def printAverages(info_dict, file):
    '''This function takes in the dictionary now filled with information gathered from join_3_databases and writes it
    to the file 'results.text' in the format #X: Tempo - X, Danceability - X, Speechiness - X, Liveness - X,
    Loudness - X.'''
```

Deezer25.py



```
def get_deezer_songs():
    '''
    Loops through ids_for_each_country list for each id corresponding with 4 countries.
    It then uses the Deezer API to obtain song information for the top 25 songs of the country. It then finds the ranking,
    title, artist, and country of each song obtained and returns a list of dictionaries.
    ex. [{ranking: 1, title: ___, artists: ____, country: ___}, {...}]'''

def setUpDatabase(db_name):
    '''
    Takes the database 'music1.db' as a [ara,eter. sets up the database, and returns cur and conn'''
def set_up_deezer_table(data, cur, conn):
    '''
    Sets up the Deezer table that will go into the 'music.db' database.
    It takes the list of tuples returned by get_deezer_songs and put them in a table
    (25 songs at a time/per run) that has the following values:
    song_id (shared key), ranking, country, title, artist.
    Obtain the last song_id in the database to identify where in data(dictionary) to start from. '''
def main():
    '''
    calls the above functions.
    '''
```

Graph.py

```
def create_vis(db_filename):

   '''This function loops through each of the 25 rows in Average and obtains the ranking,
avg_tempo, avg_danceability,avg_speechiness, avg_liveness, avg_loudness and stores them in a
dictionary. Creates a list of the 25 dictionaries made.'''

def create_tempo_graph(lst):

   '''creating graph 1 using 'music.db' database that takes the rankings as x-values and the
corresponding average tempo as y-values'''

def create_danceability_graph(lst):

   '''creating graph 2 using 'music.db' database that takes the rankings as x-values and the
corresponding average danceability as y-values'''



def create_speechiness_graph(lst):

   '''creating graph 3 using 'music.db' database that takes the rankings as x-values and the
corresponding average speechiness as y-values'''

def create_liveness_graph(lst):

   '''creating graph 4 using 'music.db' database that takes the rankings as x-values and the
corresponding average liveness as y-values'''

def create_loudness_graph(lst):

   '''creating graph 5 using 'music.db' database that takes the rankings as x-values and the
corresponding average loudness as y-values'''
```

Resources Used

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
4/4	Identify top songs in the US on the Discogs website	https://www.discogs.com/search/?sort=have%2Cdesc&ev=em_tr&format_exact=Single&country_exact=US	Understood expected result for dictionary
4/4	Identify top songs in France on the Discogs website	https://www.discogs.com/search/?sort=have%2Cdesc&ev=em_tr&format_exact=Single&country_exact=France	Understood expected result for dictionary
4/4	Identify top songs in Canada on the Discogs website	https://www.discogs.com/search/?sort=have%2Cdesc&ev=em_tr&format_exact=Single&country_exact=Canada	Understood expected result for dictionary
4/4	Identify top songs in the UK on the Discogs website	https://www.discogs.com/search/?sort=have%2Cdesc&ev=em_tr&format_exact=Single&country_exact=UK	Understood expected result for dictionary
4/7	Identifying songs on Deezer UK playlists and the id used to access	https://www.deezer.com/us/playlist/7241549564	Understood expected result for dictionary
4/7	Identifying songs on Deezer France playlists and the id used to access	https://www.deezer.com/us/playlist/1109890291	Understood expected result for dictionary

4/7	Identifying songs on Deezer US playlists and the id used to access	https://www.deezer.com/us/playlist/1313621735	Understood expected result for dictionary
4/7	Identifying songs on Deezer Canada playlists and the id used to access	https://www.deezer.com/us/playlist/1652248171	Understood expected result for dictionary
4/7	Understanding authentication process for Deezer API	https://developers.deezer.com/api/oauth	Understood steps, did not know how to find app_id
4/7	Understand how to obtain necessary info for authentication process	https://support.appreciationengine.com/article/o06djLDGXU-creating-a-deezer-app	Successfully created auth code
4/16	Install Spotipy	https://spotipy.readthedocs.io/en/2.19.0/	Successfully installed Spotipy on terminal
4/16	Spotify API set-up	https://towardsdatascience.com/extracting-song-data-from-the-spotify-api-using-python-b1e79388d50	Successfully authenticate Spotify
4/22	Implement Spotify Playlist IDs	https://morioh.com/p/31b8a607b2b0	Extracted individual song IDs from selected playlist IDs