SI206
Final Project Report
Team Edison Chiu, Jason Liang, Benjamin Wang [Github Repo]

Objectives:

- ***To create a program that:***
- Takes 25 songs from Billboard's top charts (e.g. Hot 100, Hot Dance, etc.) through Billboard API
- Retrieves respective song lengths by scraping Apple Music Web Browser
- Retrieves respective song lyrics from "lyrics.ovh" through lyrics.ovh API and counts number of words in said song
- Aggregates song length data and number of words in song data into one database
- Plots relationship between song length and genre, number of words in song and genre, and the ratio of words per song to song length by genre using matplotlib

Were all objectives achieved? Any problems encountered?

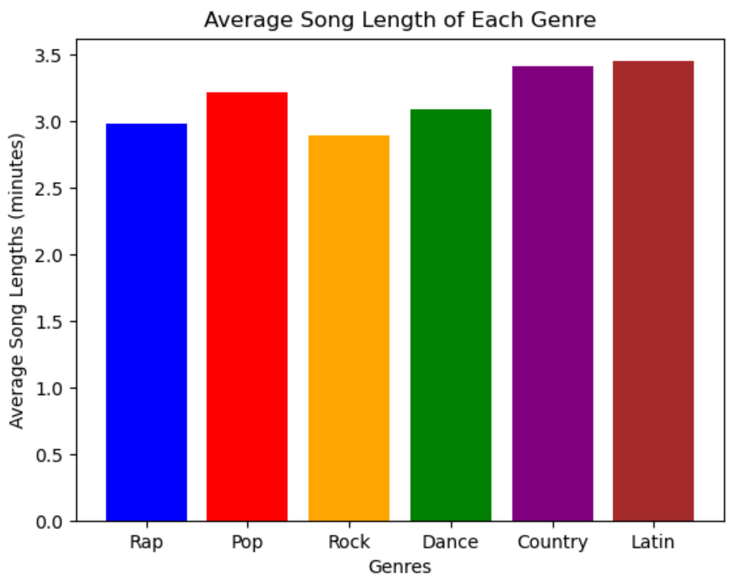
- Yes, all objectives outlined above were met.
- However, these objectives are not the original plan. The original plan was to scrape the Billboard website's top charts for songs and analyze play count on different streaming services such as Spotify and Apple Music. The following issues arose while executing our original idea: Spotify API did not provide data on play count for songs, and Apple Music API was only usable through a paid Developer account.
- Our team navigated through several revisions of objectives, including using Youtube API, Soundcloud API, and pytrends to compare song trends against play count over time.
- Finally we rested on creating a program that uses the Billboard API to retrieve songs, scrape the Apple Music web browser to retrieve these songs' lengths, and uses the lyrics.ovh API to count lyrics in each song.
- Certain problems involving the scraping of Apple Music arose, where the html of the webpage appeared to change after a few days, causing our code not to function properly, resulting in us not being able to retrieve the necessary data. Edits to our code fixed the problem, but this could be a potentially recurring problem if the html of the webpage is altered every few days.
- For data visualization, we did not change our goal of using matplotlib, however we did adjust what kind of graphs/charts we would be creating based on the relationship between the data that we collected.

Calculation file:

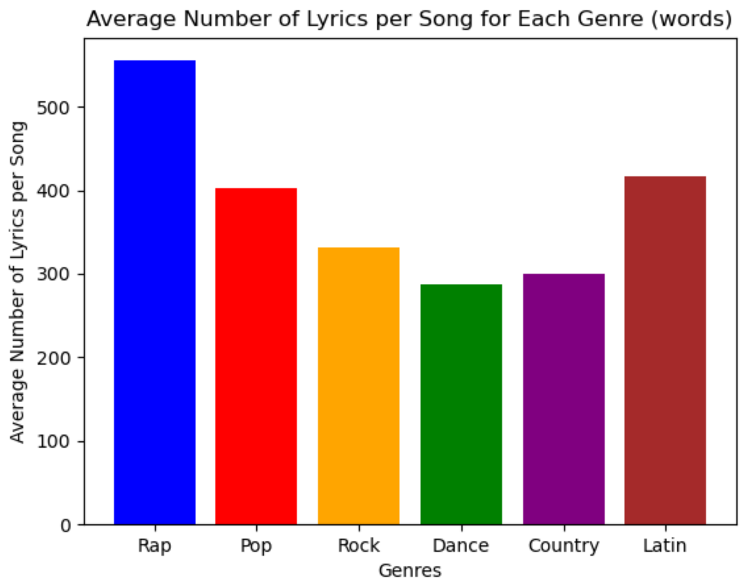
- calculations.py
- lyrics_to_time_ratios.txt

Visualizations:

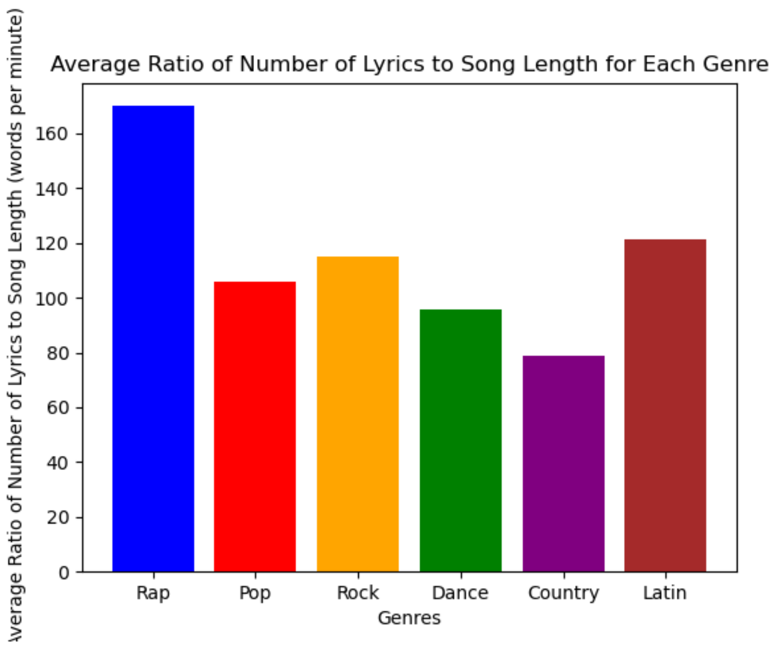
- Average song length of each genre



- Average number of lyrics per song in each genre



- Average ratio of number of lyrics to song length for each genre



Instructions for executing the program:

(1) Unzip folder
(2) Find main.py file
(3) Run the main.py Python program

Code documentation (per function listed below):
billboardapi.py:

- **def songs_list**
    - Takes in a specific category of a song (pop, rap, etc.) as the input. Uses billboard api to find matching songs, and returns a list of tuples where each tuple contains the song's title and artist.
- **def get_all_songs**
    - Does not take in any inputs. Calls songs_list on the categories pop, rock, rap, dance, country, latin to get a list of tuples for each of these categories. Creates and returns a dictionary where each key is the category and the value is the list returned from songs_list.

web.py:

- **def get_song_lengths**
    - Takes in a list (one of the values from dictionary returned by get_all_songs) and string URL (apple music) as the inputs. Scrapes the website for all the songs and its information. Iterates through each song in the inputted list; if a matching song is found on the web scraper, finds the length of the specific song. Returns a list

    consisting of the title, artist, and song length for each song that was found on the website.
- **def get_data**
    - Takes in a dictionary (the dictionary from get_all_songs) as the input. Creates keys for the categories Pop, Rock, Rap, Dance, Country, and Latin. Calls get_song_lengths for each category and sets the dictionary value to the list returned by get_song_lengths. Returns the newly created dictionary.
- **def unique_songs**
    - Takes in a dictionary (the dictionary from get_data). Creates a new list, then iterates through every song in each category. Adds the song if it has not already been added to the list. Returns the list, consisting of a tuple of title, artist, category, and song length.

lyrics.py:
- **def get_lyrics**
    - Takes artist and title of song as parameters and feeds it to lyrics.ovh API, which outputs the lyrics. Then counts the number of words in lyrics, and finally outputs a tuple in the format of (artist, title, num), along with a tuple containing the song and the first 200 characters of its lyrics.
- **def web_songs**
    - Takes in a list of song information and finds the lyrics info for each one. Returns a new list of tuples consisting of title, artist, category, and # of lyrics. Also returns list of song title and its lyrics.

database.py
- **def setUpDatabase**
    - Takes in the name of the database to create, and then returns the cursor and con for it
- **def create_bb_tables**
    - Takes in the cur and con for a specific database, and then sets up tables Categories, Lengths, Num_Lyrics, and Lyrics. Does not return anything
- **def insert_song_categories**
    - Inputs are the cursor and connection, a list of song information, and a counter variable to limit data collection to 25. First checks if the database has already been fully filled with all songs, then inserts each song’s appropriate info from the inputted list into the Categories table as long as the data collected is under 25. Returns the counter variable.
- **def insert_song_lengths**
    - Inputs are the cursor and connection, a list of song information, and a counter variable to limit data collection to 25. First checks if the database has already been fully filled with all songs, then inserts each song’s appropriate info from the inputted list into the Lengths table as long as the data collected is under 25. Returns the counter variable.
- **def insert_song_num_lyrics**

    - Inputs are the cursor and connection, a list of song information, and a counter variable to limit data collection to 25. First checks if the database has already been fully filled with all songs, then inserts each song's appropriate info from the inputted list into the Num_Lyrics table as long as the data collected is under 25. Returns the counter variable.
- **def insert_lyrics**
    - Inputs are the cursor and connection, a list of song information, and a counter variable to limit data collection to 25. First checks if the database has already been fully filled with all songs, then inserts each song's appropriate info from the inputted list into the Lyrics table as long as the data collected is under 25. Returns the counter variable.

calculations.py:
- **def select_lengths**
    - Inputs a specific category of songs. Creates and returns a list of each matching song's title, artist, category, and lengths after joining the Categories and Lengths table based on song title.
- **def select_lyrics**
    - Inputs a specific category of songs. Creates and returns a list of each matching song from the Lyrics table.
- **def calc_ratio**
    - Inputs a time in the (xx:xx) format and a number of lyrics. Calculates and returns the ratio of these values.
- **def get_all_ratios**
    - Inputs a list of songs and their lengths and a list of songs and their # of lyrics. Calculates the ratio of time/# of lyrics for each song and returns a list of tuples consisting of song title and the calculated ratio.
- **def average_ratios**
    - Inputs a list of songs and their calculated ratios. Finds and returns the average of all the calculated ratios.
- **def write_to_file**
    - Inputs a number (the average ratio) and a song category. Creates/opens a file named "lyrics_to_time_ratios.txt" and writes the inputted information to it. Closes the file and doesn't return anything.
- **def average_ratio_of_category**
    - Inputs a database cursor, connection, and a song category. Writes the average song length to # of lyrics ratio for the category to a file. Doesn't return anything.

plot.py:
- **def lengths_only**
    - Inputs a list of song information (title, artist, category, lengths) and returns a list of just the lengths
- **def lyrics_only**

- Inputs a list of song information (title, artist, category, # lyrics) and returns a list of just the # lyrics

- **def category_avg_length**
    - Inputs a database cursor, connection, and song category. Returns the average lengths of all songs in the specified category.
- **def plot_lengths**
    - Inputs a database cursor and connection. Finds the average lengths of all given categories, and creates a bar graph visualizing the lengths of each category using matplotlib.
- **def category_avg_lyrics**
    - Inputs a database cursor, connection, and song category. Returns the average lyrics of all songs in the specified category.
- **def plot_lyrics**
    - Inputs a database cursor and connection. Finds the average # lyrics of all given categories, and creates a bar graph visualizing the # lyrics of each category using matplotlib.
- **def avg_ratio**
    - Inputs a database cursor, connection, and song category. Returns the average song length to # lyrics ratio of all songs in the specified category.
- **def plot_ratios**
    - Inputs a database cursor and connection. Finds the average song length to # lyrics ratio of all given categories, and creates a bar graph visualizing the ratios of each category using matplotlib.

RESOURCE DOCUMENTATION:

Date	**Issue Description**	**Location of Resource**	**Result**
4/21	API documentation for pytrends originally, how to use?	https://pypi.org/project/pytrends/	Could not get API to work, Python environment issues
4/22	Could not use originally planned APIs, need to come up with something else	Zoom Office Hours	Sought advice from TA and found new APIs to use
4/22	Looking for new APIs related to music	https://rapidapi.com/collection/music-apis	Was able to obtain Billboard API
4/23	Could not install Billboard API package	https://stackoverflow.com/questions/28436769/how-to-change-default-anaconda-python-environment	Successfully ran APIs in a default Python environment (pip and Anaconda did

			not work)
4/24	Didn't know how to stop inserting into database after 25 were inserted	https://www.kite.com/python/docs/sqlite3.Cursor.rowcount	Was able to track how many rows were affected and therefore stop at the right time
4/24	How to use lyrics.ovh API	https://lyricsovh.docs.apiary.io/#	Learned how to use lyrics.ovh API to extract lyrics
4/27	How to use matplotlib	https://matplotlib.org/stable/contents.html	Generated bar charts
4/27	Plot was blank when saved	https://stackoverflow.com/questions/30765455/why-is-my-plt-savefig-is-not-working	Was able to save plot as a png file
4/27	How to write to a file	https://www.w3schools.com/python/python_file_write.asp	Able to write to a text file