Final project checkpoint

**1, Project code**

github repo: <https://github.com/wyjessica860/final_project_fiveoseven>

Python package: Ploty, datetime, statistics, collections, requests, bs4, pandas, Flask

**2, Data sources**

* All the access requires caching. The API key is applied on the website:

<https://genius.com/api-clients>

Therefore, all the crawling except searching for artist requires a headers with parameter 'Authorization' = 'Bearer ' +  GENIUS\_API\_KEY

* The data crawling refers to the instruction website:

<https://docs.genius.com/#/getting-started-h1>

* The data we obtained can be seen on the github and how we get those data can be seen by sample\_text.py
* The detailed process of each new crawling is as follows:

1. We provide two choice for searching, either a song or an artist.

For the song, we use search\_url = ‘http://api.genius.com/search’

For the artist, we use search\_url = ‘https://genius.com/api/search/artist’

The parameter is the search item

The response structure can be seen in **response\_of\_search\_songs.json** and **response\_of\_search\_artist.json.** The songs of the artists or the songs with similar name will be returned in the response.

1. From the response, we can get the song\_id of each song, which is used in the following lyrics crawling. We did this by searching the ‘api\_path’ in the response

The response is stored in **response\_of\_song\_id.json**, where we can extract a series of information of this song

song\_url = ‘http://api.genius.com{api\_path}’

Meanwhile, we can also get the artist’s api\_path and return the response of the information of this artist, stored in **response\_of\_artist\_id.json**

Using **config.py** we extract the attribute we want from those two json file.

**The instance will be inserted into database, discussed in the following section.**

1. One of the result from the response of song\_id is the url of lyrics page, which is used as the search url for the lyrics. The lyrics\_url = 'http://genius.com{lyric\_path}'

The response is saved in **response\_of\_lyric\_path.htm**l. We will use beautiful soup to extract the lyrics from this html.

[4] The result lyrics is saved in **lyric\_sample.txt**

**The resulting data and code can be seen in the sample\_data and sample\_text.py on the github website.**

On the other hand, we add the caching process. The detailed code can be seen in data\_crawling.py, and the cache file is ccache.json.

The data and attributes will be discussed below.

**3, Database**

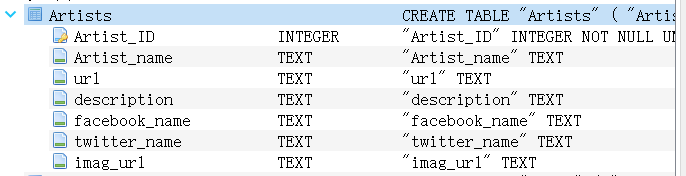
We’ve created 2 table in the dataframe **genius\_artists.db.**

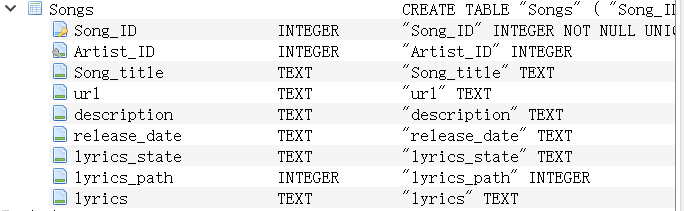
The command can be seen in **database.py.**

The primary key for Artists is Artist\_ID, which is unique for all data in this table

The primary key for Songs is Song\_ID, which is unique for all data in this table

The foreign key in Songs is Artist\_ID, connect the two table. Since an artist may have a lot of songs, we create the foreign key in songs table.





I’ve already inserted some data into this database: 12 for Artists and 30 for Songs

In the following time, I may insert around 100 artists and around 1000 songs.

The artist\_id is used to define the record and the url of crawling. Artist\_name is the name of the artists. Url is the official webpage that displays all the information of this artist and can be displayed on the web browser by this url link. Description briefly describe the artist and facebook\_name, twitter name is for user to better follow the artist. Image\_url is the link of a picture of this artist.

The song\_ID ,url,descriptionserves for songs the same function as artist\_id for artists. ARtist\_Id is the artist that this song belongs to. Song\_title is the title of thie song. Release date is when the song is released. Lyrics\_state if it’s complete then we can get the lyrics from the lyrics\_path, which contributes to our crawling. Lyrics is the lyrics of this song.

**4, Data Presentation**

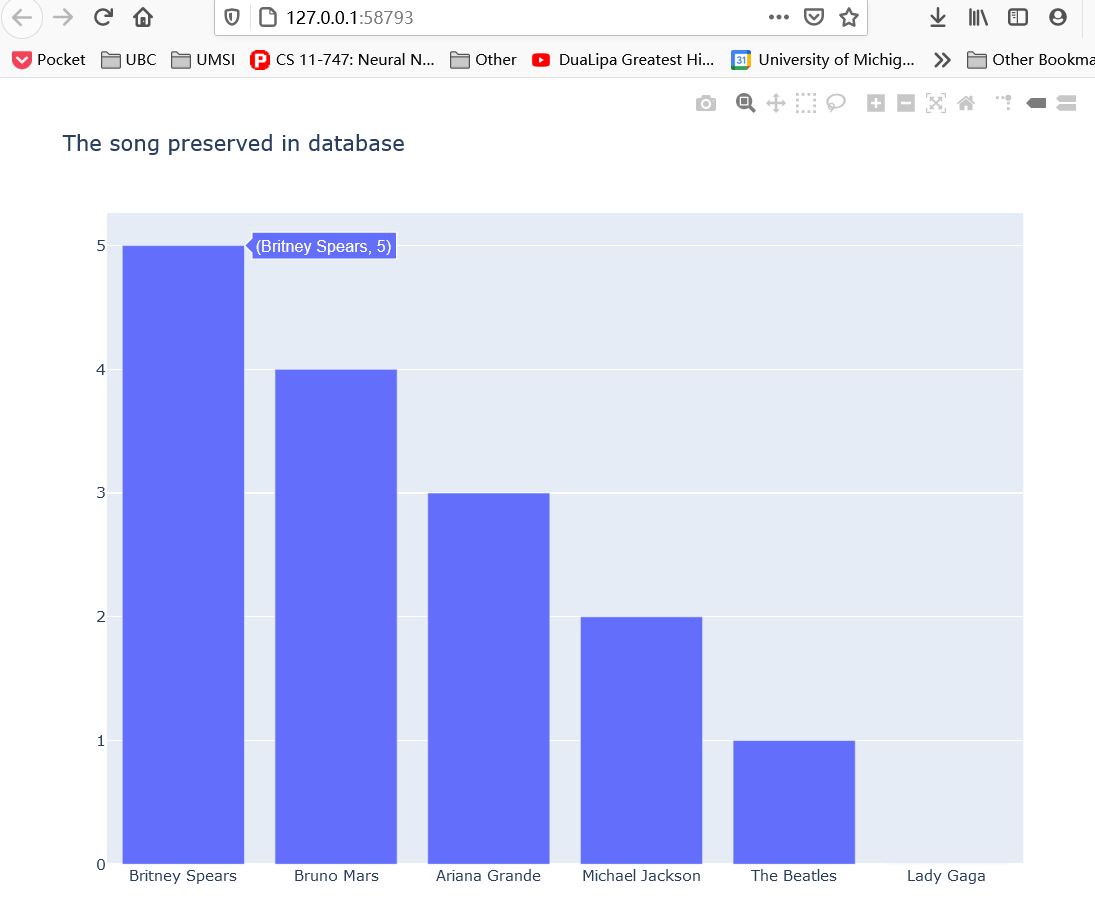
It is said in comment of previous final project proposal that the data presentation is not explained clear enough. I will try my best to explain it using the sample example, preserved in the **data\_pre\_display.py**. I hope I can get some advice and the extra points in my proposal.

First, we insert some data into the database, which can be seen in the main function of **database.py**

Our date presentation is as follows:

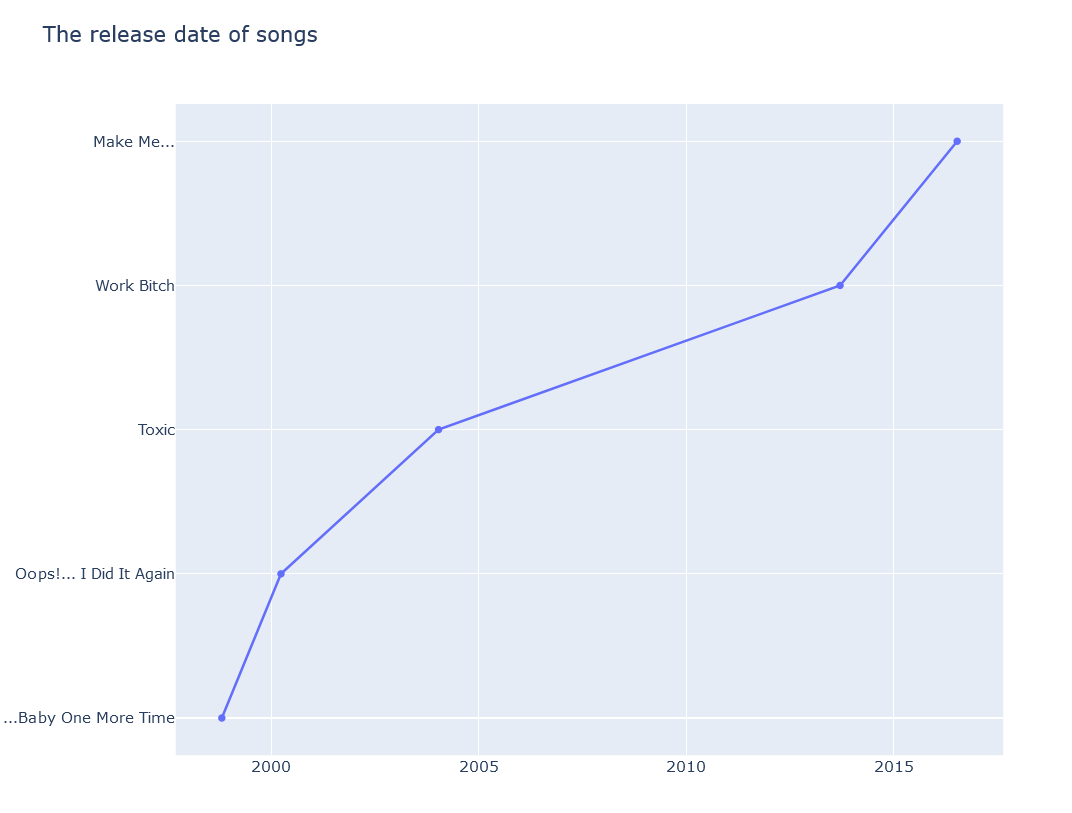
First, we can print out the information of the artist or the song your want. In the future, we can format those information to make output more readable. For example: song\_id: ....

Second, we can make a bar plot to display the number of songs each artist has. The artist can either be the selected one like what we did in the demo or all the artist in the database by selecting the unique artist id( demos will be done in the future).

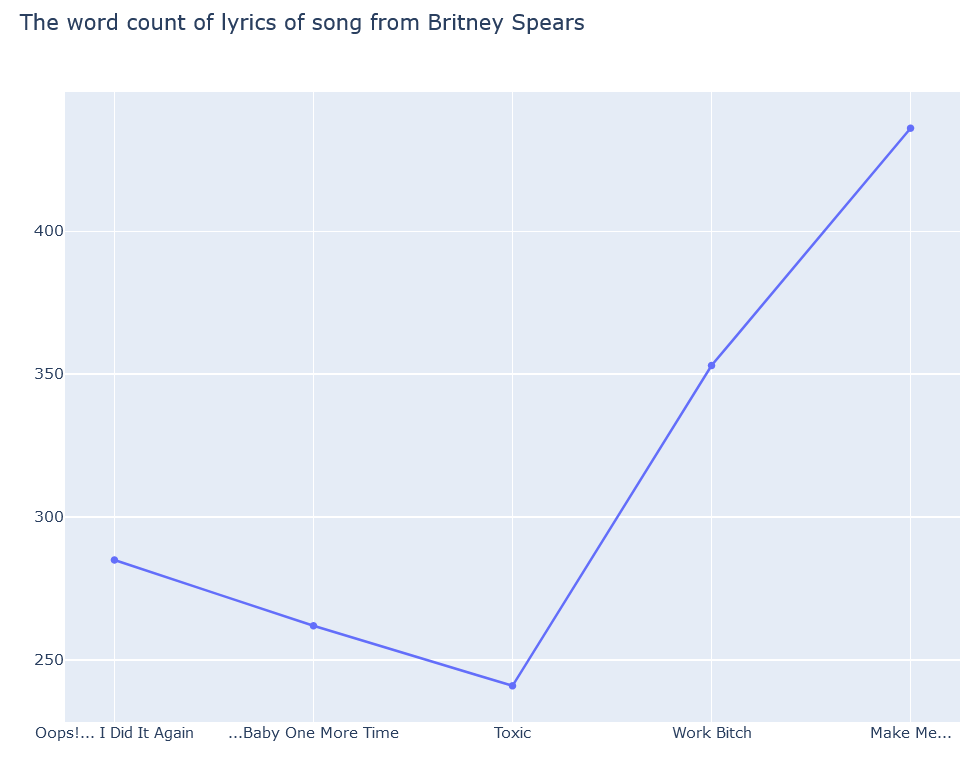


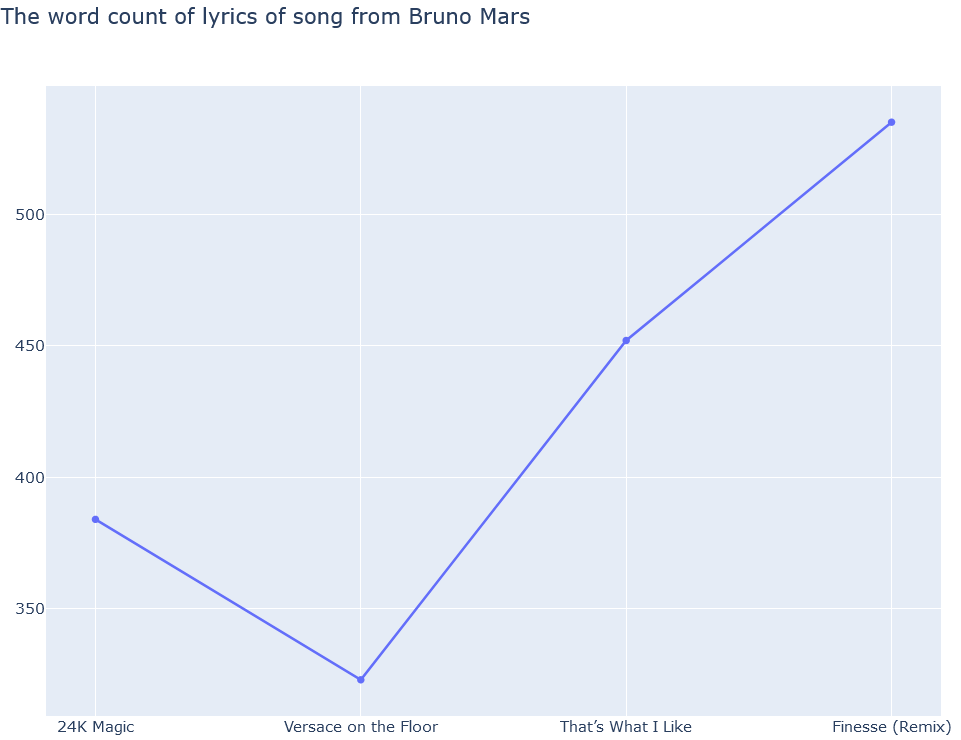
Third, we can trace the release of the songs of a singer. The following picture displays the song released by 'Britney Spears', it is sorted by the release time.

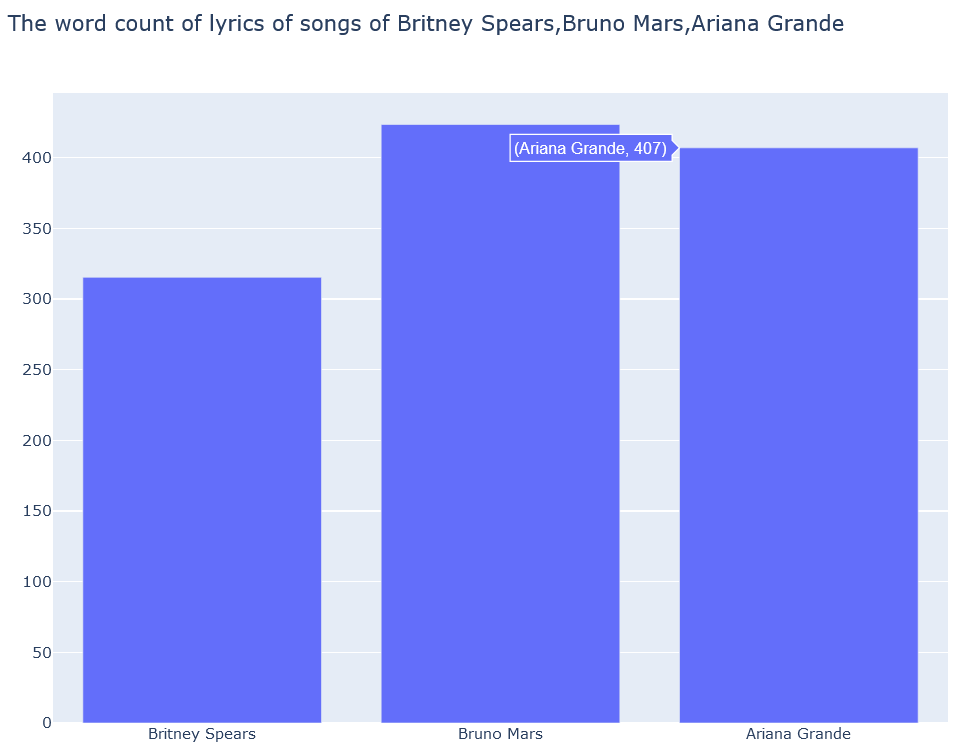
Similarly, if correct input set, we can sort whatever songs.



[4]lyric words counting: using scatter plot for the single artist and bar plot for multiple artist.

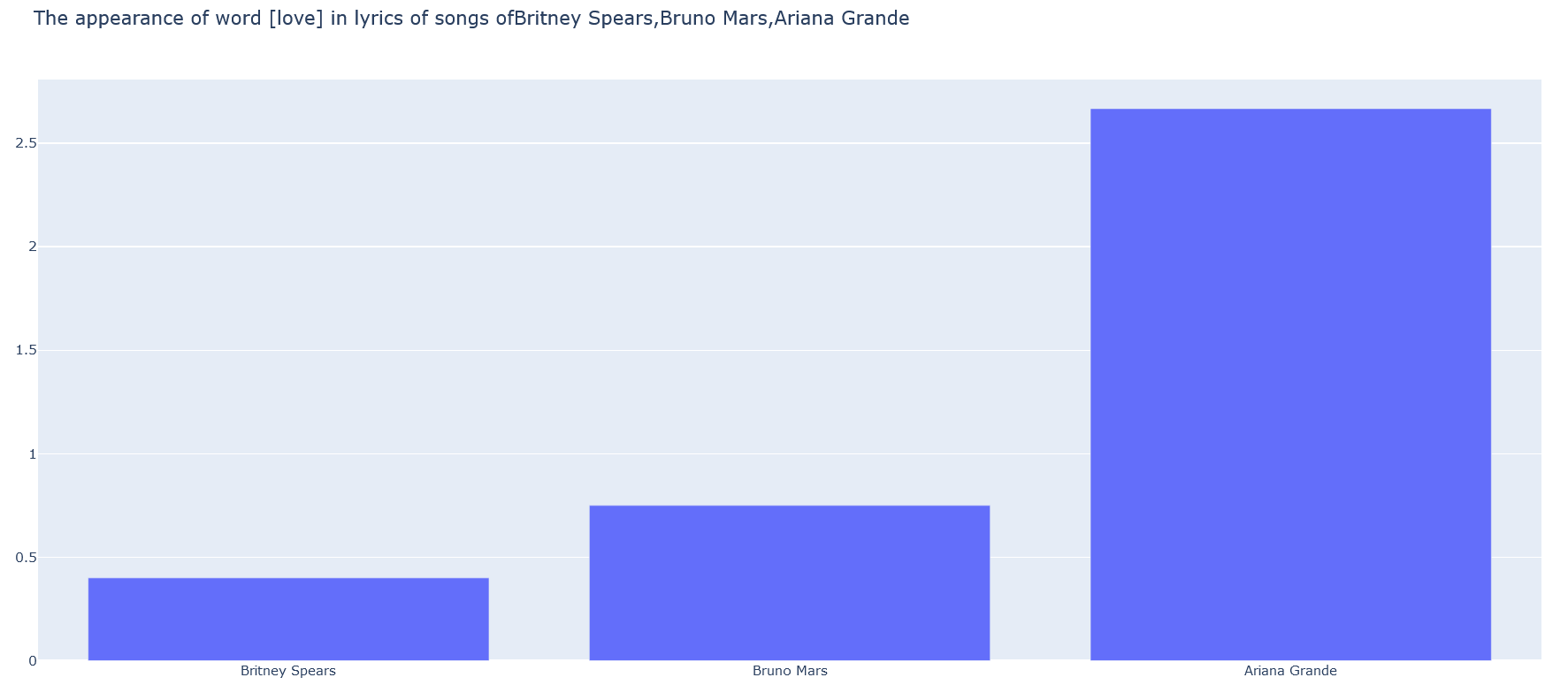






The plotting requires the processing of text. In the future, we can display more amazaing graphs if the processing is advanced. For exampl,e we can count how many times does ‘love’ appears in each artist’s songs on average.

The result is shown below.



Finally, we want to build an interface to better display our results. This will be done in the following time. We may use command line or the flask.