**Final Project Report**

**1, Project code**

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

**2, Data sources**

* 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 and sample\_data**
* 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.

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**

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 attributes will be extracted from data and then be inserted into database. Therefore, the attributed will be discussed in the Database section.**

**3, Database**

**3.1 Attributes summary:**

I've inserted 817 pieces of data in [Songs] and 87 in [Artists]

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 s**ong\_ID ,url,description** serves 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 the 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.

**3.2 Database schema:**

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

CREATE TABLE "Artists" (

"Artist\_ID" INTEGER NOT NULL UNIQUE,

"Artist\_name" TEXT,

"url" TEXT,

"description" TEXT,

"facebook\_name" TEXT,

"twitter\_name" TEXT,

"imag\_url" TEXT,

PRIMARY KEY("Artist\_ID")

CREATE TABLE "Songs" (

"Song\_ID" INTEGER NOT NULL UNIQUE,

"Artist\_ID" INTEGER,

"Song\_title" TEXT,

"url" TEXT,

"description" TEXT,

"release\_date" TEXT,

"lyrics\_state" TEXT,

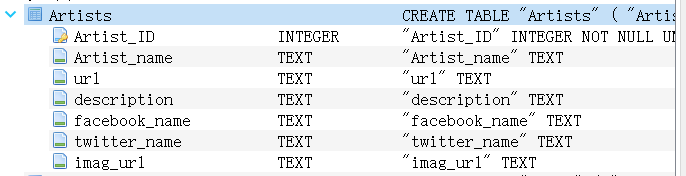
"lyrics\_path" INTEGER,

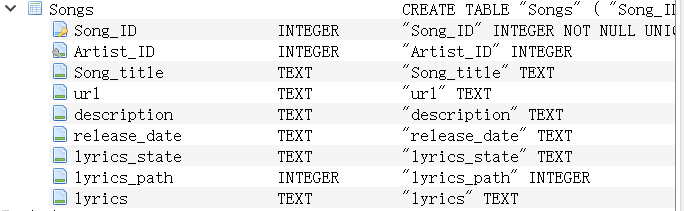
"lyrics" TEXT,

PRIMARY KEY("Song\_ID"),

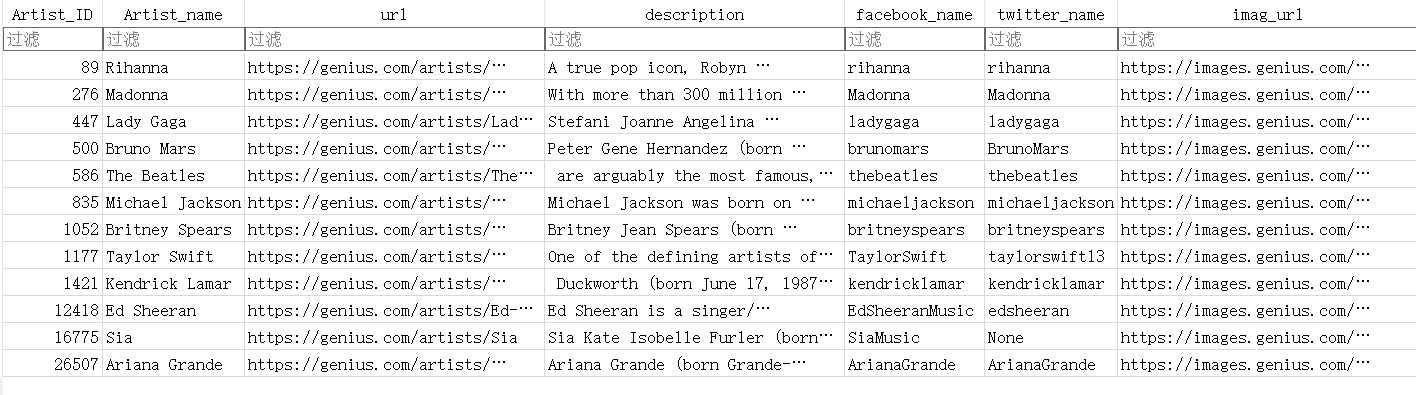
FOREIGN KEY("Artist\_ID") REFERENCES Artists("Artist\_ID")

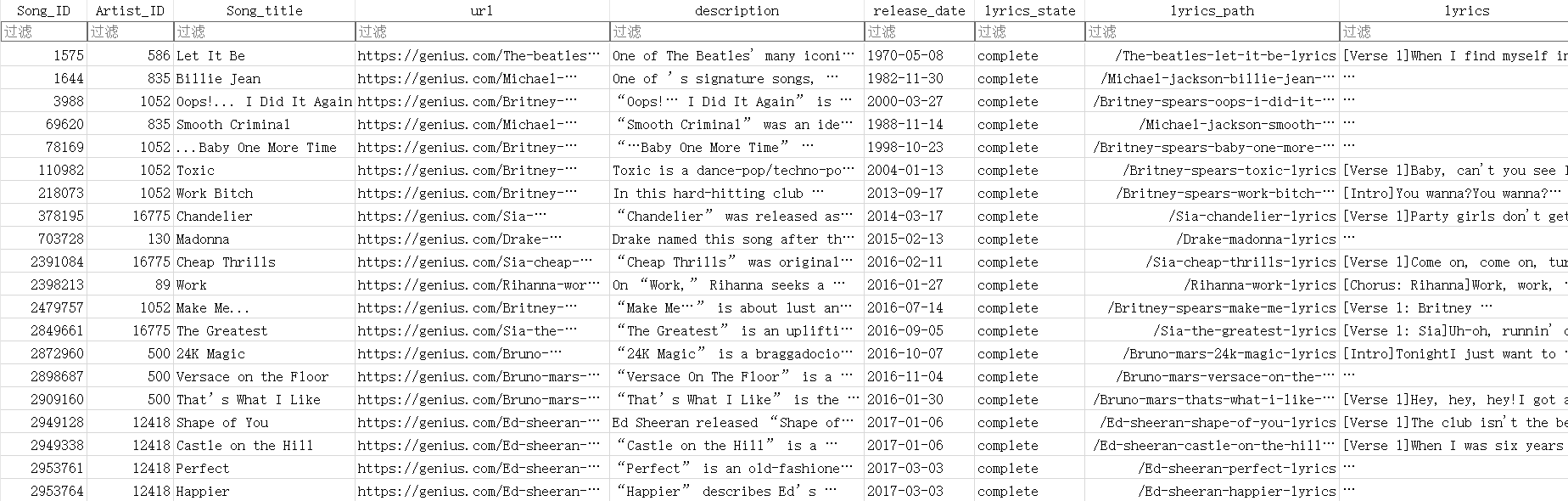
The foreign key in Songs is Artist\_ID, connect the two table. Since an artist may have a lot of songs, we create the foreign key in songs table.





**3.2 Examples:**





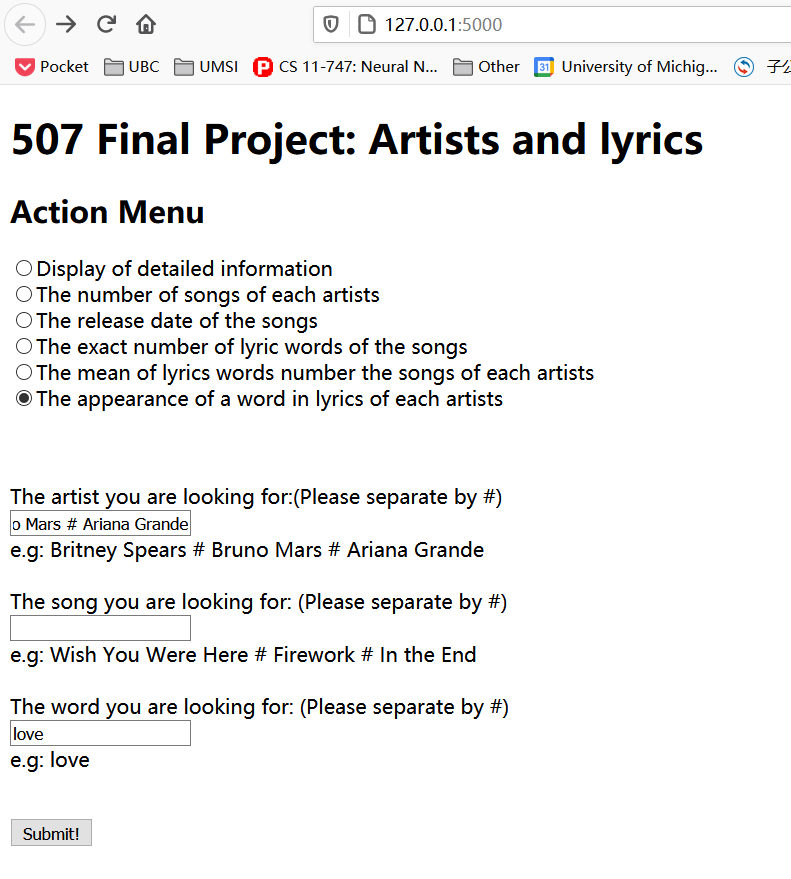
**4, Data Presentation**

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.

Note that the example is given depending on the status of the database. After the checkpoint, we have inserted more songs and artists into database. As more songs and artists are inserted, the number of songs or other information may present differently. Overall, we can get an idea of how the data can be plotted. If you want to interact with this project, you can run the interface.py. The property and pattern of plots will shown the same as what we presented.

We create a Flask app for interaction.

There are six main action choices:

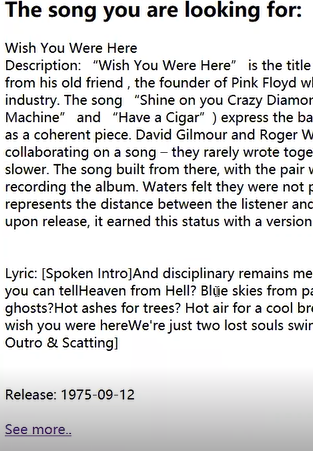


First, we insert some data into the database, which can be seen in the main function of **database.py.**

Our date presentation is as follows:

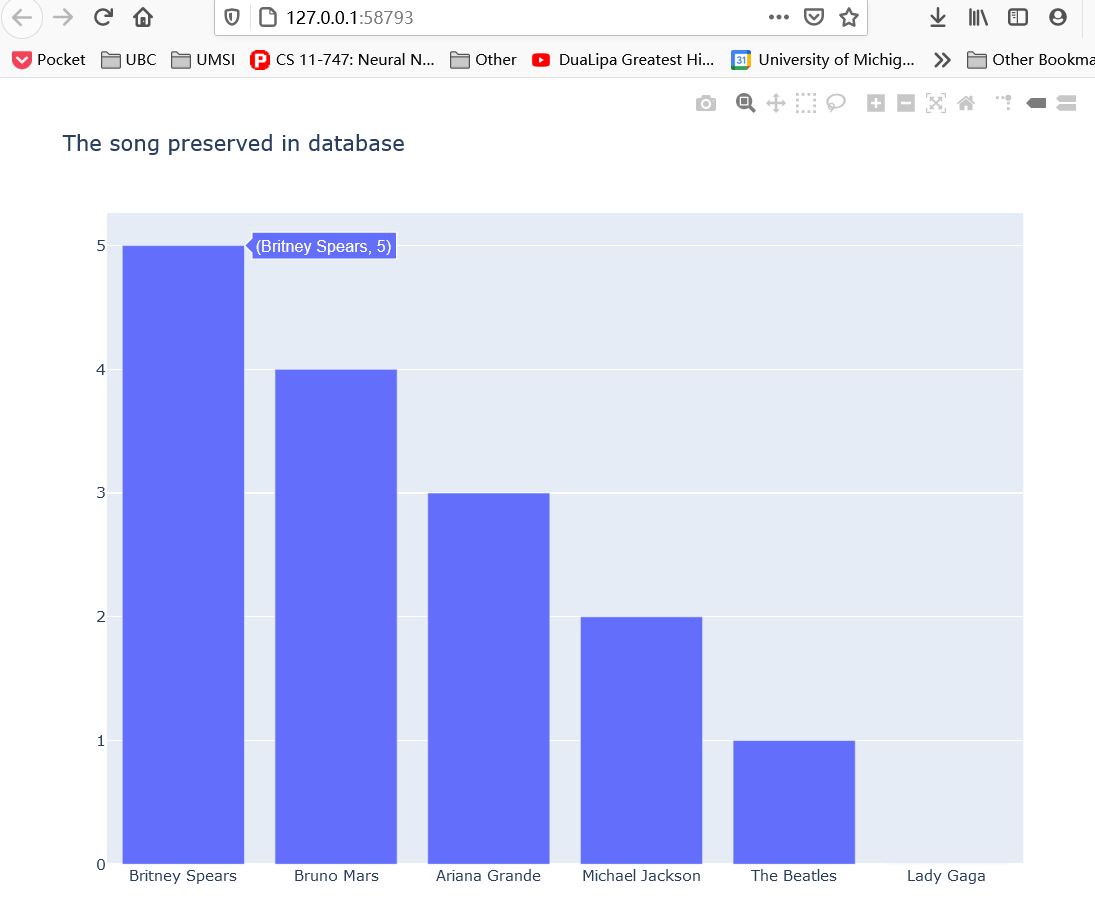
* [Display of detailed information]

It will transfer to a new website where all the information of the artists including a general description, facebook, twitter, a picture and a link of more detailed information as well of the songs including a general description, lyrics, release date will be displayed.



* [The number of songs of each artists]

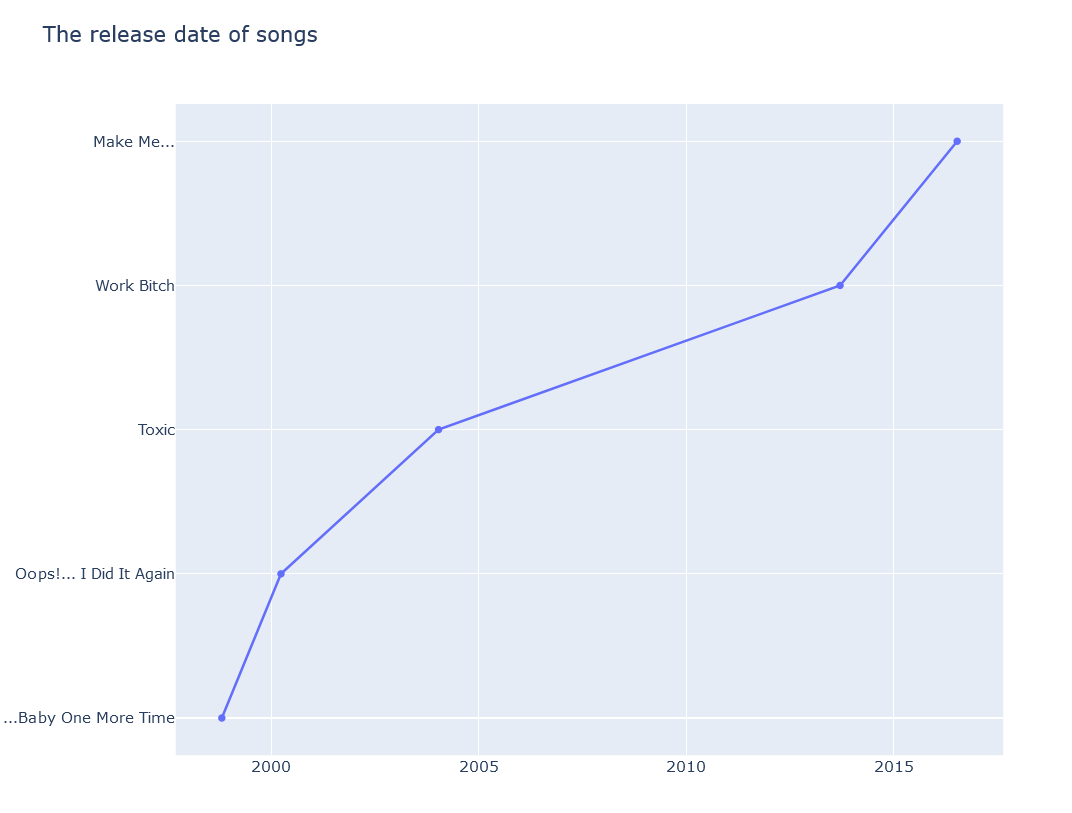
The second option is a bar plot to display the number of songs each artist has.



* [The release date of the songs]

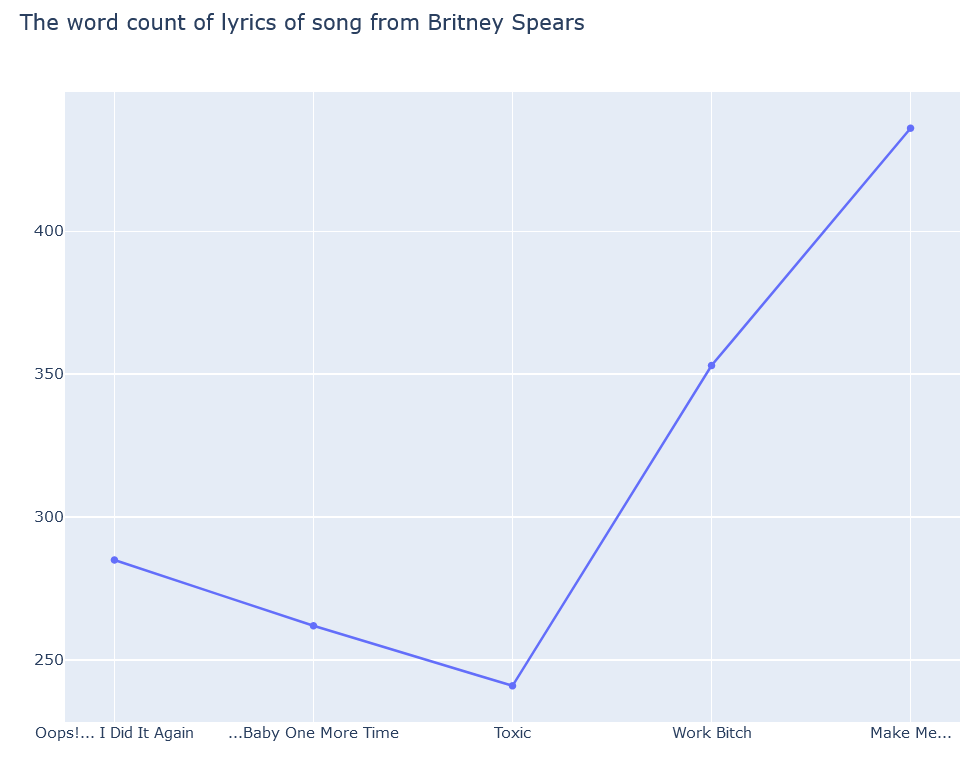
Third, we use a scatter plot to show the release of the songs of a singer. The following picture displays the song released by 'Britney Spears', it is sorted by the release time.

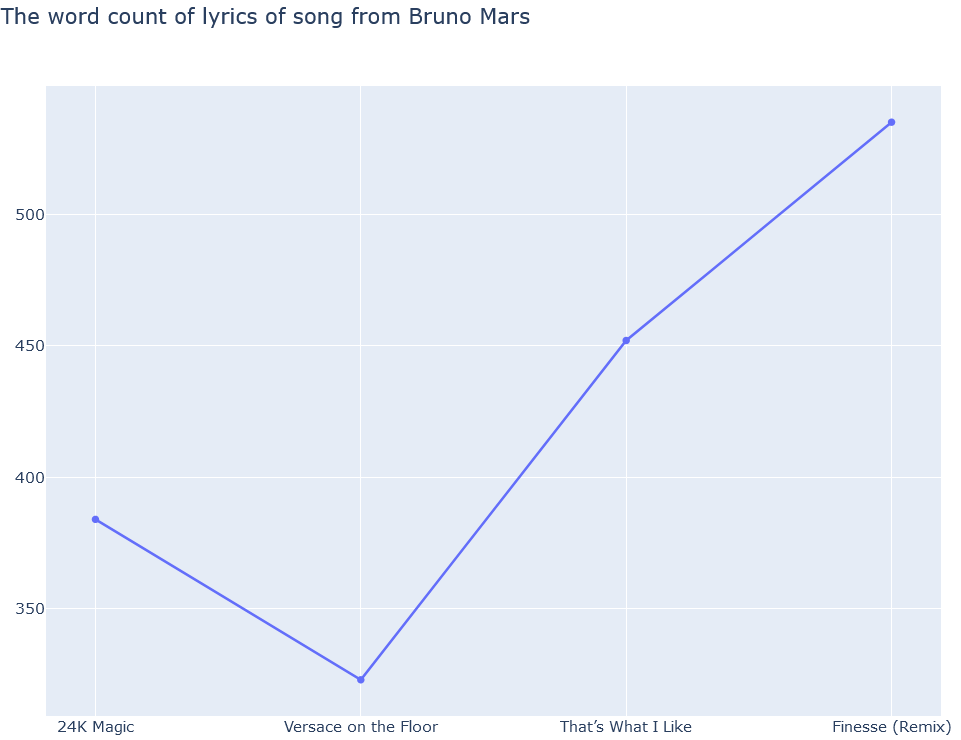
Similarly, if we input a number of songs, the y axis to turn into the songs we selected.



* [The exact number of lyric words of the songs]

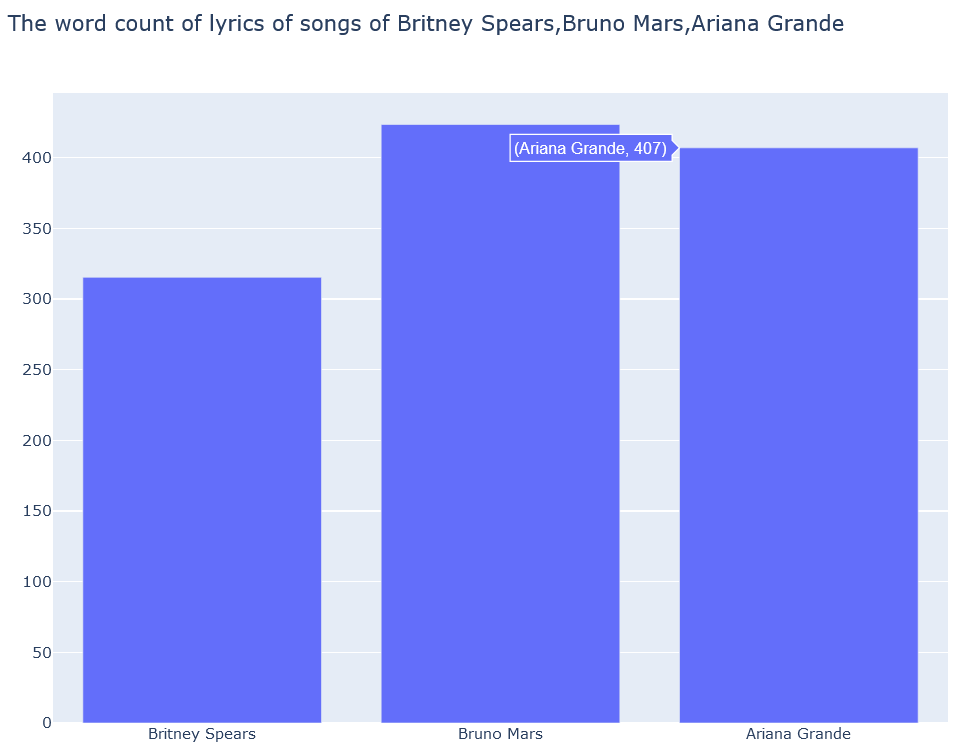
Using scatter plot, we count the number of lyric words for the single artist per plot. The xaxis is the name of songs, the y axis is the count.





* [The mean of lyrics words number the songs of each artists]

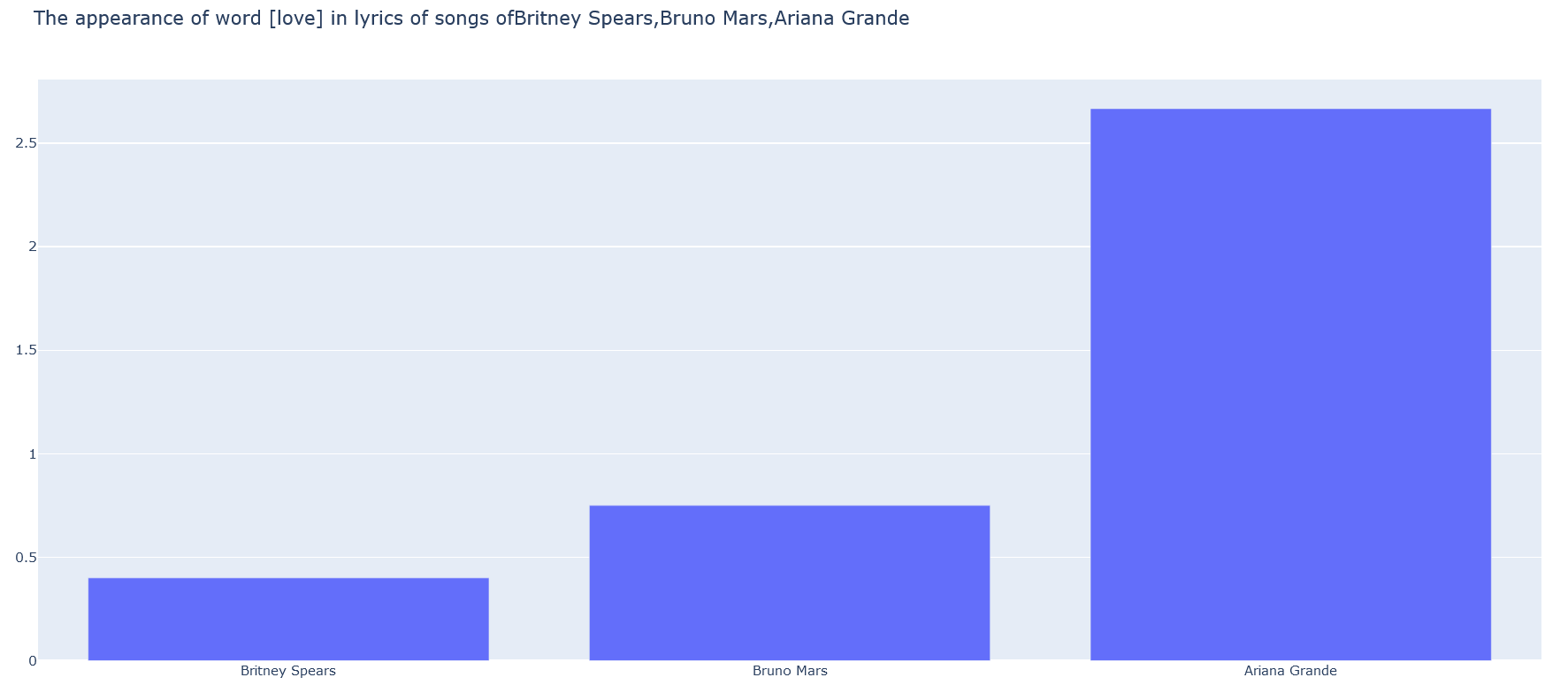
After get the word count of each lyrics, we can calculate the mean of the lyrics words of each artist and use a bar plot to plot it.



* [ The appearance of a word in lyrics of each artists]

The plotting requires the processing of text. In the future, we can display more amazing graphs if the processing is advanced. For the time being, we manage to count how many times certain word for example ‘love’ appears in each artist’s songs on average.

The result is shown below.



The technology applied in this process includes ploty, flask

**5, Demo vedio**

https://drive.google.com/file/d/1KBcowoqvcZO7jiBEegyTcwP4F9LkKqQG/view?usp=sharing