SI 206 Final Report

Yang Meng, Ted Yuan, Haochen Yu

**Goals**

**Original Goals:**

Our initial goals including:

①collecting the information (artist, name, category/genre, price) of a song and to compare how many songs that each music platform has.

②Using three APIs/websites (iTunes API and YouTube Music API).

③Creating three visualizations to help us better understanding our results intuitively.

**Goals Achieved:**

Ultimately, we reached on the following steps:

①Utilizing 2 API (iTunes and LastFM) and 1 website (Billboard Top 100 hottest music)

②Differentiating each song based on the “price” criteria

③Creating graphs to exhibit how prices will affect a song’s popularity

**Problems**

We also encountered following problems during the project:

①We have some difficulties when accessing the YouTube Music API at the beginning, so we switched to the LastFM API later.

②We met issues with limiting import to the database.

③We failed to modify the font size when drawing our graphs.

**Calculation**

Table

Description automatically generated

The maximum, minimum, and average price are calculated both using Excel math functions and the python (in draw.py). The results are exactly the same.

**Visualization**

The following first graph illustrates the price occurrence frequency at the different given price level.

Chart, box and whisker chart

Description automatically generated

The following second graph showed the average price of an individual song and the price range, including minimum and maximum.

Chart, bar chart

Description automatically generated

The following third graph used a pie chart to exhibit percentage of the price distribution with “1 dollar” partition.

Chart, pie chart

Description automatically generated

The following fourth bar showed the top 5 most popular tracks and the corresponding numbers of listeners.

Chart, bar chart

Description automatically generated

The following fifth histogram illustrates the five artists who have the least numbers of listeners.

Chart, bar chart

Description automatically generated

**Running Code Instructions**

If you want to get all the diagrams, you first have to install matplotlib by typing

“pip3 install matplotlib” in the terminal.

Then do

“python3 driver.py”

To run download all the data into the database.

Then do

“python3 draw.py” to get all the visualization.

**Code Documentation**

**api.py file:**

**get\_songs\_from\_itunes(song\_name):**

“Takes in a song\_name and formats it with the item that matches from iTunes API. Returns a list of strings of each song\_name from the given API.”

**get\_artists\_info(artist):**

“Takes in an artist and formats it with the item that matches from iTunes API. Returns a list of strings of each artist from the given API.”

**get\_song\_from\_last\_fm(artist):**

“Takes in an artist and formats it with the item that matches from LastFM API. Returns a list of strings of each artist from the given API.”

**top\_100\_songs\_from\_billboard():**

“No inputs. Uses BeautifulSoup to read the top 100 songs. Returns a dictionary in the format (song’s name: artist).”

**database.py file:**

**set\_up\_data\_base(file\_name):**

“Takes the name of a database, a string, as an input. Returns the cursor and connection to the database.”

**create\_tables(cur, conn):**

“Takes the database cursor and connection as inputs. Returns nothing. Create four tables: Artist, Genre, Track, LastFM.”

**add\_data\_to\_artist(cur, conn):**

“Takes the database cursor and connection as inputs. Add artists info into the database.”

**add\_data\_to\_genre\_track(cur, conn):**

“Takes the database cursor and connection as inputs. Add genre info into the database.” **add\_data\_to\_LastFM(cur, conn):**

“Takes the database cursor and connection as inputs. Add LastFM info into the database.”

**draw.py file:**

**import\_database():**

“No inputs. Returns artist\_list, track\_list, genre\_list, price\_list.”

**plotting\_price(price\_list):**

“Inputs the price\_list. Sets up the x-y axis info and plots the price graph.”

**maximum(price\_list):**

“Inputs the price\_list. Returns the maximum value item from the price\_list.”

**minimum(price\_list):**

“Inputs the price\_list. Returns the minimum value item from the price\_list.”

**average(price\_list):**

“Inputs the price\_list. Returns the average value item from the price\_list.”

**plotting\_min\_max\_avg(price\_list):**

“Inputs the price\_list. Sets up the x-y axis info and plots the price bar graph.”

**sorting(price\_list):**

“Inputs the price\_list. Appends the price items that larger or smaller than 1.”

**plotting\_pie(price\_list):**

“Inputs the price\_list. Plots the pie graph.”

**sorting\_data(track\_list, listeners):**

“Inputs the track\_list and listeners. Returns the dictionary that stores the five largest numbers of listeners.”

**plotting\_listener(track\_list, listeners):**

“Inputs the track\_list and listeners. Sets up the x-y axis info and plots the histogram.”

**sorting\_data\_1(artist\_list, listeners):**

“Inputs the artist\_list and listeners. Returns the dictionary that stores the five smallest numbers of listeners.”

**plotting\_artist(artist\_list, listeners):**

“Inputs the artist\_list and listeners. Sets up the x-y axis info and plots the histogram.”

**main():**

“Takes nothing as an input and return nothing. Calls the functions plotting\_price(price\_list), plotting\_min\_max\_avg(price\_list), plotting\_pie(price\_list), plotting\_listener(track\_list, listeners), plotting\_artist(artist\_list, listeners).”

**driver.py:**

**download\_database():**

“No inputs. Calls the functions create\_tables(cur, conn), add\_data\_to\_artist(cur, conn), add\_data\_to\_genre\_track(cur, conn), add\_data\_to\_LastFM(cur, conn).”

**get\_data():**

“No inputs. Returns artist\_list, track\_list, genre\_list, price\_list.”  **write\_csv():**

“No inputs. Returns nothing. Opens the file and writes return values of the functions to the file.”

Tables for Artist & Genres, created from dataBase.py: Text

Description automatically generatedGraphical user interface, text

Description automatically generated

Tables for LastFM, created from database.py:Text

Description automatically generated

Tables for Track, created from dataBase.py:Graphical user interface, text

Description automatically generated

**Resources**

**Resources we have used,**

**iTunes API, LastFM API, billboard:** [**https://www.billboard.com/charts/hot-100**](https://www.billboard.com/charts/hot-100)**,**

**Tools we used, Python3 and sqlite3 is all main platforms. We also used requests, json to grab our data from the API.**

**Beautifulsoup is also used to scraping data from the internet specifically the billboard website.**

**Finally, we use matplotlib to make graphs of our results.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Issue Description** | **Location of Resources** | **Result** |
| April 19th | Couldn’t add 25 items into the database | Office hour | GSI pointed out the mistake we made in our code. |
| April 19th | Didn’t know where to start when building database | Talk to my teammates | Yes, they helped me to start after looking up online resources together. |
| April 19th | Didn’t know where to start with LastFM API | Looked up the tutorial of LastFM API | Yes |
| April 20th | Matplotlib couldn’t install on my laptop | Looked up online matplotlib official sites | No, still not working right now |
| April 21th | Code takes too long to run | Looked up online and talked to the GSI | Sort of, it still takes some time to download the data, but better than before. |