**Team members: Stella Young and Elizabeth Kim**
**Link to github repository:** https://github.com/stellahyoung/SI206Final

**APIs, SQLs, & Visualizations**

1. **The goals for your project**

Goals:

- Our goal for this project was to use the Spotipy API and a chart informing the amount of concerts planned for top artists this year to investigate a potential correlation between the two. Through the Spotipy API, we aimed to pull artist names, number of followers, as well as their defined popularity value. Our final code was intended to allow us to compare the numerous values provided by these two sources and draw any correlation that exists between the two.

2. **The goals that were achieved**

- Successfully web-scraped from the Spotipy API to get information about musical artists and their popularity and a website using beautiful soup to retrieve an artist and the number of their upcoming concerts
- Successfully calculated the correlation coefficient between artist popularity and the number of upcoming concerts.
- Successfully created two data visualizations that show the positive correlation between artist popularity and the number of upcoming concerts.

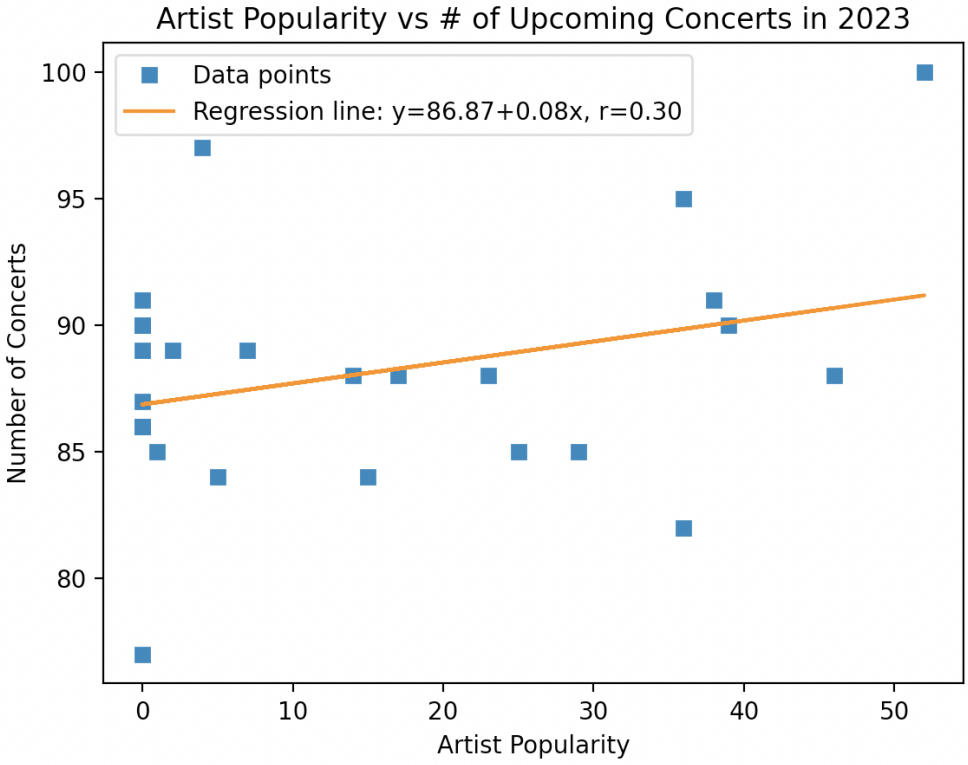
3. **The problems that you faced**

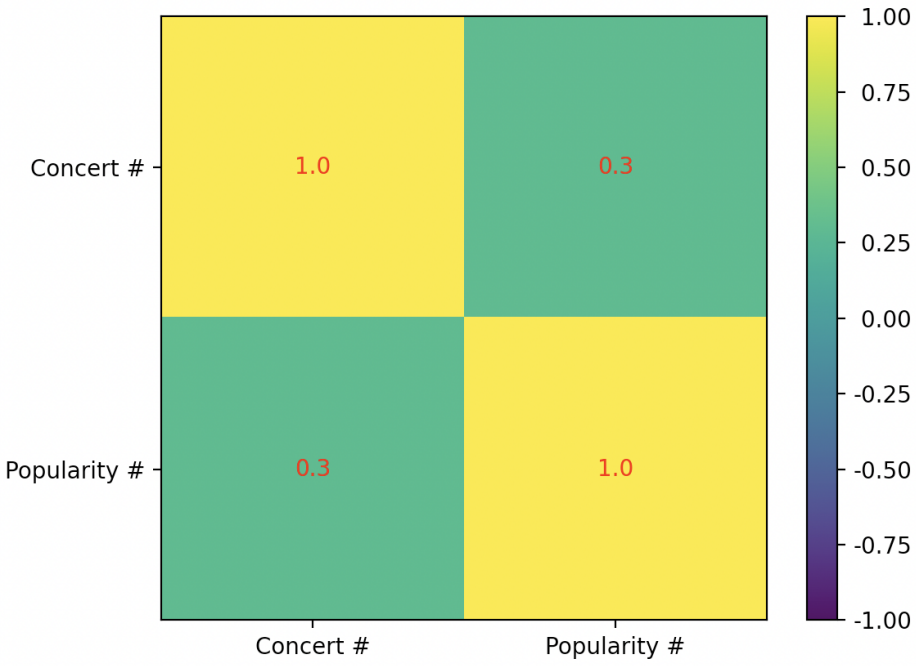
- Our biggest challenge was when we initially were reading in the first Spotify API instead of Spotipy, resulting in the need for a token and identification that automatically expired every hour. After transitioning to the Spotipy API to retrieve the same information, we were able to successfully overcome this obstacle.
- We also initially struggled with inputting our data into the data visualizations. We continually had issues with making sure the data was accurate and the visualization was actually representative of what we wanted to present.
- Additionally, making sure that our database was reading in 25 items each time was something we struggled to implement.

4. **Your file that contains the calculations from the data in the database** (10 points)

```
Correlation between number of concerts and popularity:
The correlation coefficient between number of concerts and popularity was r = 0.30311962298385026.
```

5. **The visualization that you created (i.e. screenshot or image file)**




## 6. **Instructions for running your code**

In order to successfully run our code, make sure to have a SQL reader installed on your computer so you are able to view the database and see the tables that we've created.

1) Run through the final.py file to fill the database for each of the tables.
2) Run code multiple (4) times to reach at least a 100 items - 25 will be accumulated each time
3) You may need to give the database time to load all of the information.
4) Once the code successfully runs, three tables, two visualizations, and our outputted csv file should be visible and accessible.The csv file will be outputted to your device and may need to be searched by title ("calculations.txt") to access.
5) Please note as the website is live, the number of concerts for each artist changes everyday; the final correlation coefficient may differ from the screenshotted images in this report.

## 7. **Documentation for each function that you wrote. This includes the input and output for each function**

**Database:**

- def setUpDatabase(db_name):
  - Sets up the database with an input(name of choice). Returns the cursor and the connection to the database

**Concert Data**

- def concert_data():
  - No inputs. Requests and scrapes songkick website using beautiful soup. Returns a dictionary that has the artist name and the number of their upcoming concerts

- def create_concert_table(cur, conn):
  - Takes the dictionary that we got from concert_web, the cursor, and the connection of the database as inputs. Creates the "Concerts" table. Returns nothing.

- def add_into_concert_table(cur,conn, add):

- Inserts artists' name and number of upcoming concerts into the table. Returns nothing.

**Spotify Data**

- def spotify_api():
  - Takes nothing as input. Uses request module to get artist name and their popularity from the Spotipy API. Returns data for a specified date range in JSON formatting.

- def create_spotify_table(cur, conn):
  - Takes in database cursor and connector. Returns nothing. Creates a new table called 'Spotify'.

- def add_into_spotify_table(cur, conn, add):
  - Takes in database cursor, database connector, and specified integer. The function calls the spotify_api() and loops through the data from a starting point to the limit and indexes the wanted data. Indexes the artist id and their popularity. It then appends the wanted data as a tuple to a list. Then, it will loop through the list of tuples and insert it into the Spotify table. Returns nothing.

- def create_artist_id_table(cur, conn):
  - Takes in database cursor and connector. Returns nothing. Creates a new table called 'Spotify_Followers'.

- def insert_id_data_table(cur, conn, add):
  - Takes in database cursor, database connector, and specified integer. The function calls the spotify_api() and loops through the data from a starting point to the limit and indexes the wanted data. Indexes the artist and the number of their followers. It then appends the wanted data as a tuple to a list. Then, it will loop through the list of tuples and insert it into the Spotify table. Returns nothing.

**Calculations**

- def join_tables(cur,conn):
  - Takes in database cursor and database connector as inputs. Uses JOIN to return a list of tuples in the format (artist number of concerts, artist popularity) where the dates are the same

- def correlation_calc(list_of_tuple):

- Takes in the list of tuples from the join_tables() function as input. Calculates the correlation coefficient between artist popularity and the number of upcoming concerts

- def write_correlation_calc(filename, correlation):
    - Takes in a filename and the correlation coefficient that is returned from the correlation_calc() function. Returns a text file that writes the correlation coefficient between artist popularity and number of upcoming concerts.

**Visualizations**

- def create_regression_line(list_of_tuple):
- Takes in the list of tuples returned from join_tables(). Uses Matplotlib to create and return a scatter plot with a regression line of the correlation between artist popularity and number of upcoming concerts.

- def create_heatmap(list_of_tuples):
- Takes in the list of tuples returned from join_tables(). Returns a heat map to show the correlation coefficient between artist popularity and number of upcoming concerts.

**Main**

- def main():
    - Takes in no input, returns nothing.
    - Calls setUpDatabase(), create_concert_table(), create_spotify_table(), create_artist_id_table, join_tables(), correlation_calc(), write_correlation_calc(), create_regression_line(), create_heatmap() functions

8. **Resource Documentation**

| Date | Issue Description | Location of resource | Result |
|---|---|---|---|
| 12.8.22 | How to use and read in Spotipy API | https://spotipy.readthedocs.io/en/2.21.0/ | Taught us how to retrieve data from the Spotify API |
| 12.8.22 | Lists artists' and the number of their upcoming concerts | https://www.songkick.com/leaderboards/popular_artists | Let us scrape data so we could use it as one of the |

| | | | |
|---|---|---|---|
| | | | variables in the correlation coefficient we found |
| 12.8.22 | How to calculate correlation coefficient in python utilizing NumPy | https://realpython.com/numpy-scipy-pandas-correlation-python/ | Provided us good instruction and outline on how to calculate correlation coefficient with two data sets |
| 12.9.22 | How to create correlation matrix | https://likegeeks.com/python-correlation-matrix/ | Gave clear step by step tutorial on hwo to create correlation matrix/heat map in Python |
| 12.9.22 | How to only take in a limit of 25 each time into database | https://piazza.com/class/l754aa3ib8z43s/post/481 | Piazza discussion helped in clarifying that 25 items were being added each time; gave us the idea of “add” and utilizing a starting & limit variable to achieve this |