Sarah Hubbard, Sarah Penrose, Jennifer Kim

SI 206

Barbara Ericson

12 December 2023

A Data Analysis of the Music Industry

GitHub Repository: https://github.com/sarah-pen/jss-final-project.git

Planned Goals

Our goal for the project was to analyze an individual's music habits using data from various sources. At the start of our project process, we planned to use the LastFm API, Musixmatch API, and Ticketmaster API, as they are all related to music. While these APIS offer a vast amount of data, we narrowed it down to a few data points that we wanted to gather. For LastFm (http://ws.audioscrobbler.com/2.0), we hoped to retrieve top artists and songs from the last month from one of us. For Ticketmaster (https://app.ticketmaster.com/), we wanted to find information on recent concerts and events. For Musixmatch (https://api.musixmatch.com/ws/1.1/), we wanted to gather additional information regarding an artist's rating and genres. Using the data that we collected, we planned on making possible calculations on top songs and artists (possibly top 10) for the latest time frame, average cost of a concert or most common location to visit, overall rating of your songs based on Musixmatch, and most common genres based on Musixmatch. To visualize the data and calculations, we anticipated creating pie charts of most common genres, bar graphs for top songs, bar graphs for top artists, bar graphs for most common locations to visit, as well as a histogram of average costs of concerts (to see distribution).

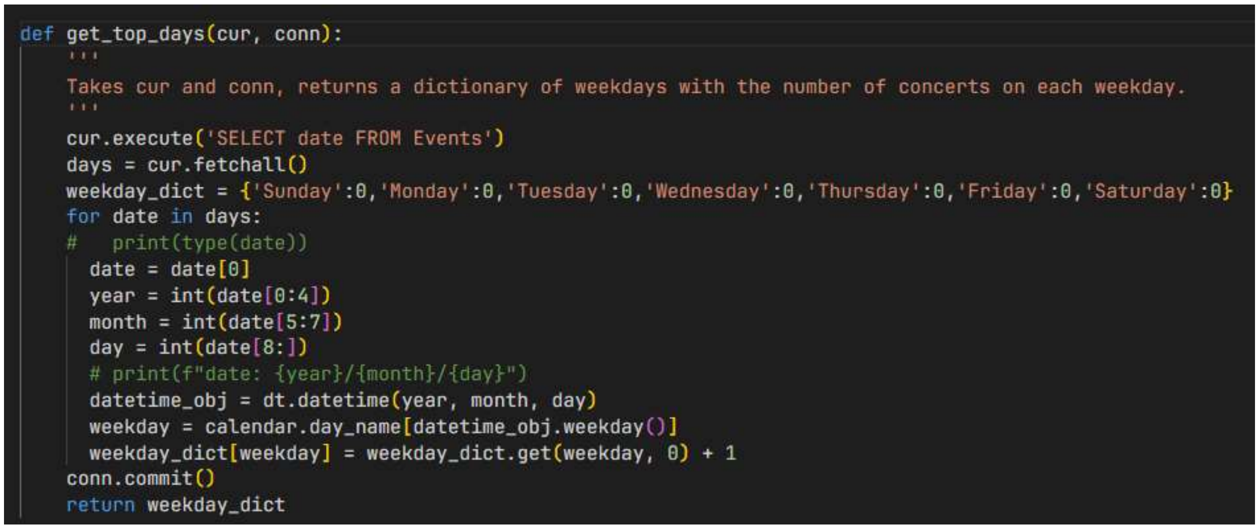
Achieved Goals

When implementing our planned goals into motion, we had decided to use the APIs we had originally planned for (http://ws.audioscrobbler.com/2.0, https://app.ticketmaster.com/, https://api.musixmatch.com/ws/1.1/). However, we did make changes as to the type of data we collected from each API, the calculations we wanted to implement, and the visualizations for these calculations. For instance, instead of getting data on artist genres for Musixmatch, we had decided to find the respective artist countries as specific artist genres were unavailable. We also increased the number of artists and songs gathered from LastFM. For Ticketmaster, we gathered data on events, cities, and venues for each artist. Using this information, we calculated the top artist countries, top days of the week for concerts, top artists, and concerts in the midwest. We made corresponding visualizations ranging from pie charts to bar graphs.

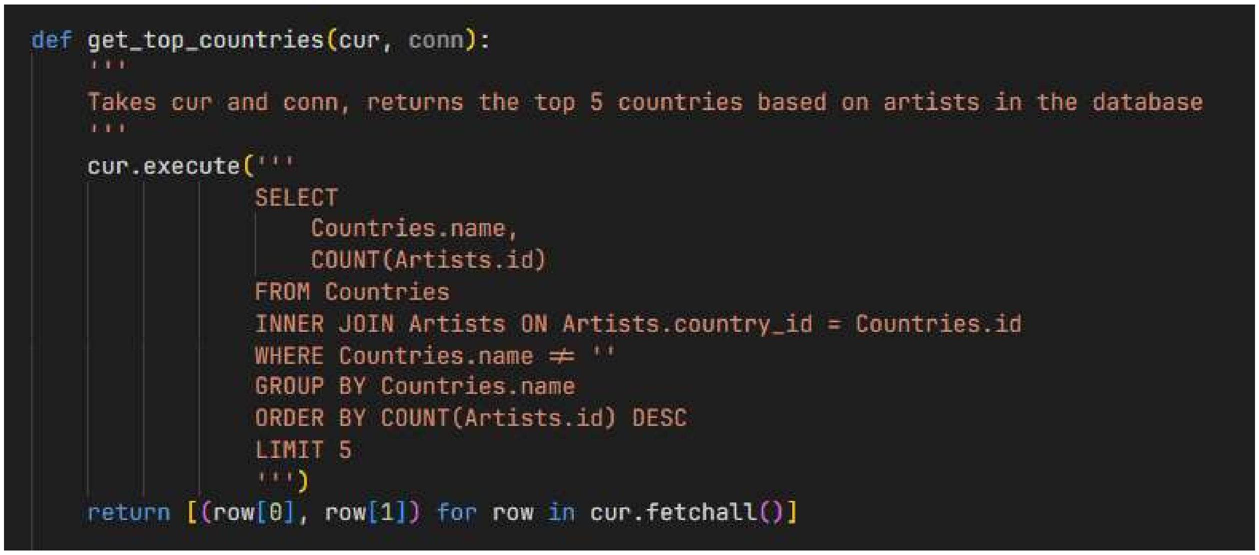
Problems

- Problem 1: Ticketmaster has a quota limit for each API key, which lead us to create multiple API keys to access their API
- Problem 2: Ticketmaster's event data has a lot of missing information, which leads to incomplete data collection
- Problem 3: Issues finding a overarching genre for Artists using Musixmatch API, used another data point (countries) instead

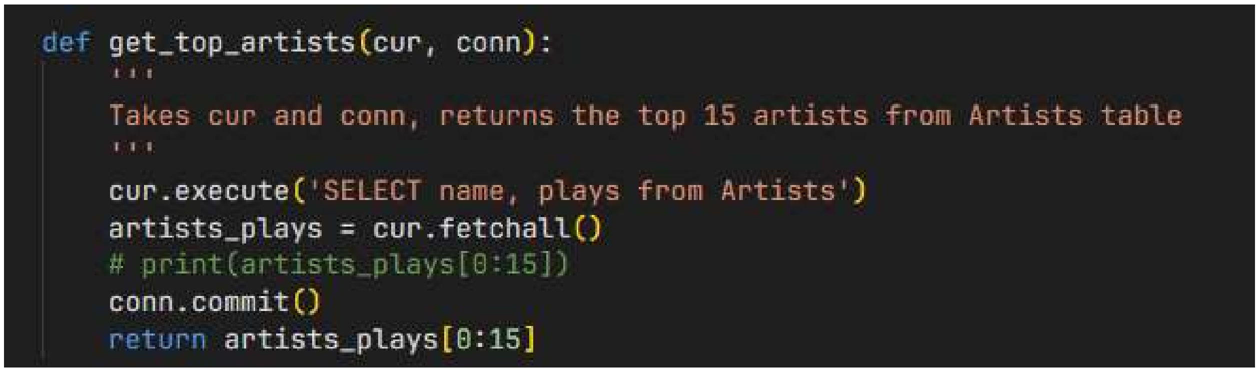
Calculations



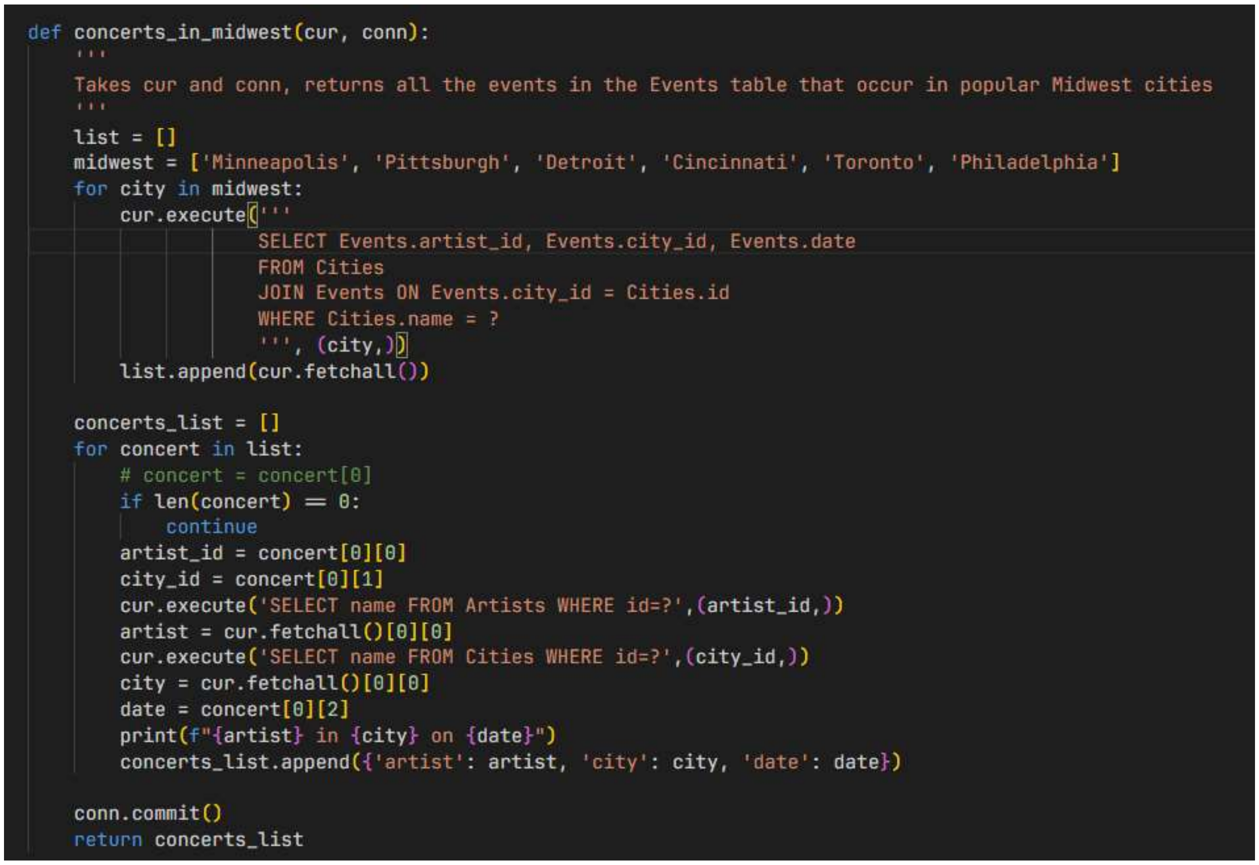
```python
def get_top_days(cur, conn):
    '''
    Takes cur and conn, returns a dictionary of weekdays with the number of concerts on each weekday.
    '''
    cur.execute('SELECT date FROM Events')
    days = cur.fetchall()
    weekday_dict = {'Sunday':0,'Monday':0,'Tuesday':0,'Wednesday':0,'Thursday':0,'Friday':0,'Saturday':0}
    for date in days:
    #   print(type(date))
      date = date[0]
      year = int(date[0:4])
      month = int(date[5:7])
      day = int(date[8:])
      # print(f"date: {year}/{month}/{day}")
      datetime_obj = dt.datetime(year, month, day)
      weekday = calendar.day_name[datetime_obj.weekday()]
      weekday_dict[weekday] = weekday_dict.get(weekday, 0) + 1
    conn.commit()
    return weekday_dict
```



```python
def get_top_countries(cur, conn):
    '''
    Takes cur and conn, returns the top 5 countries based on artists in the database
    '''
    cur.execute('''
                SELECT
                    Countries.name,
                    COUNT(Artists.id)
                FROM Countries
                INNER JOIN Artists ON Artists.country_id = Countries.id
                WHERE Countries.name ≠ ''
                GROUP BY Countries.name
                ORDER BY COUNT(Artists.id) DESC
                LIMIT 5
                ''')
    return [(row[0], row[1]) for row in cur.fetchall()]
```

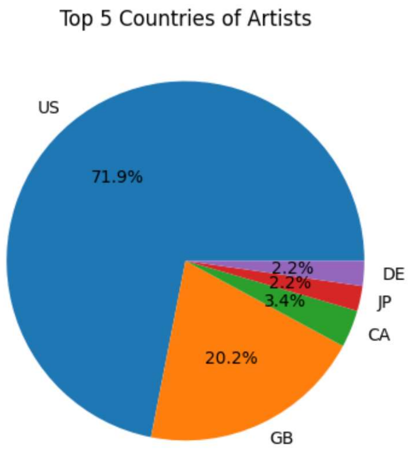


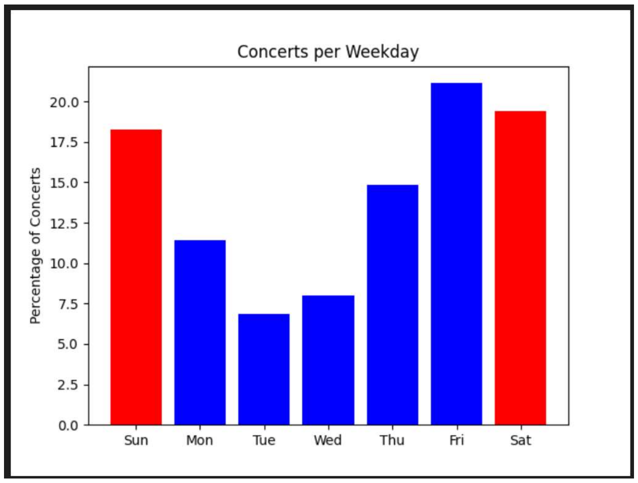
```python
def get_top_artists(cur, conn):
    '''
    Takes cur and conn, returns the top 15 artists from Artists table
    '''
    cur.execute('SELECT name, plays from Artists')
    artists_plays = cur.fetchall()
    # print(artists_plays[0:15])
    conn.commit()
    return artists_plays[0:15]
```

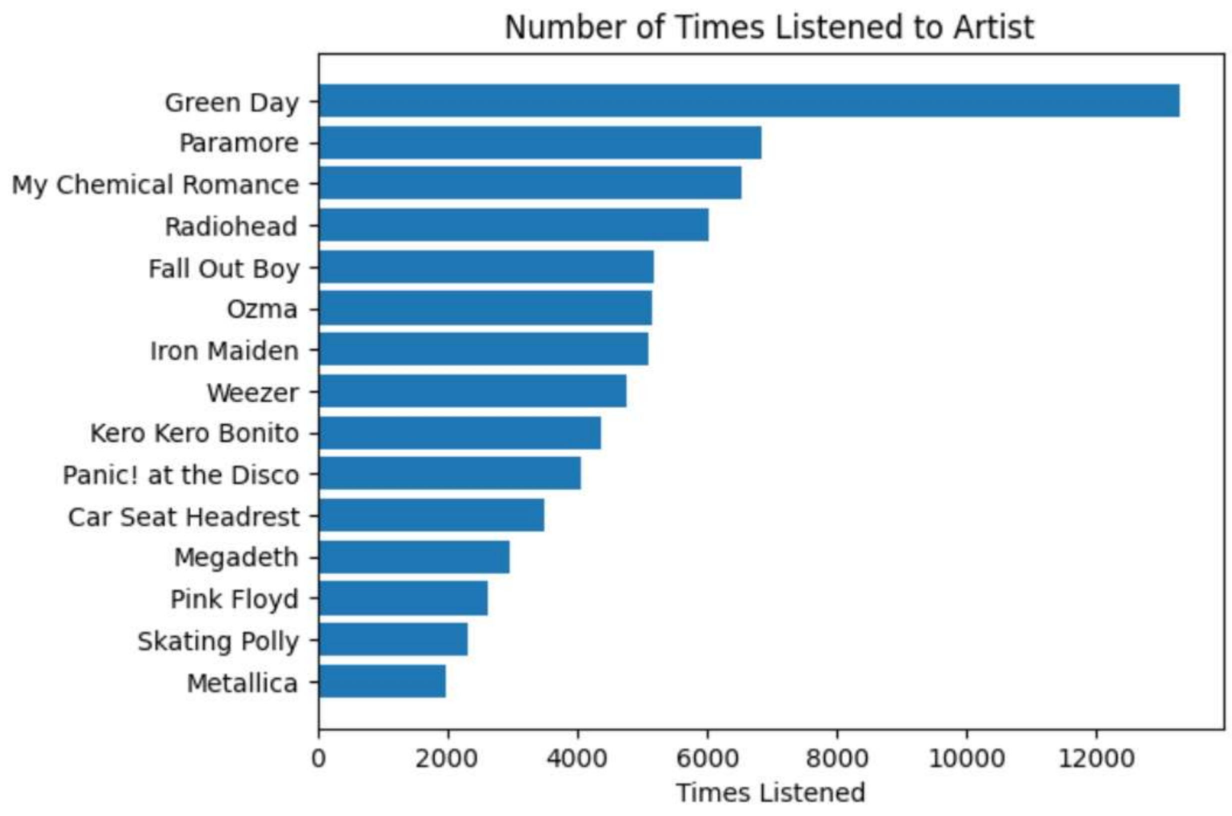


```
def concerts_in_midwest(cur, conn):
    '''
    Takes cur and conn, returns all the events in the Events table that occur in popular Midwest cities
    '''
    list = []
    midwest = ['Minneapolis', 'Pittsburgh', 'Detroit', 'Cincinnati', 'Toronto', 'Philadelphia']
    for city in midwest:
        cur.execute('''
                    SELECT Events.artist_id, Events.city_id, Events.date
                    FROM Cities
                    JOIN Events ON Events.city_id = Cities.id
                    WHERE Cities.name = ?
                    ''', (city,))
        list.append(cur.fetchall())

    concerts_list = []
    for concert in list:
        # concert = concert[0]
        if len(concert) == 0:
            continue
        artist_id = concert[0][0]
        city_id = concert[0][1]
        cur.execute('SELECT name FROM Artists WHERE id=?',(artist_id,))
        artist = cur.fetchall()[0][0]
        cur.execute('SELECT name FROM Cities WHERE id=?',(city_id,))
        city = cur.fetchall()[0][0]
        date = concert[0][2]
        print(f"{artist} in {city} on {date}")
        concerts_list.append({'artist': artist, 'city': city, 'date': date})

    conn.commit()
    return concerts_list
```

Visualizations







Code Instructions

Install VSCode or a similar code editing program, and install DB Browser for SQLite. Open the zip file in the code editing program, and run main.py file at least four times. Make sure to wait until all the returned data and print statements are finished outputting before running again.

Functions Documentation

<u>LastFm File:</u>

- get_and_insert_top_artists(database, counter, period='alltime', limit='125')
    - Input: Database filename, counter variables (initiates artist and song count as 0), period (default is ‘alltime’), limit (default is ‘125’)
    - Output: None. Inserts top artists into database table *Artists*
- get_and_insert_top_songs(database, counter, period='alltime', limit='125')

- Input: Database filename, counter variables (initiates artist and song count as 0), period (default is ‘alltime’), limit (default is ‘125’)
- Output: None. Inserts top songs into database table *Songs*

<u>Musixmatch File</u>

- artist_populated(artist_name, database):
  - Input: artist_name (name of artist), database (path to SQLite database i.e music.db)
  - Output: returns a boolean value where True if there is an existing record in Artists table for country_id and rating for given artist name and False otherwise
- add_rating(artist_name, rating, database):
  - Input: artist_name (name of artist), rating (rating of artist), database (path to SQLite database i.e music.db)
  - Output: no return value for this function, however, it updates the Artists table in the database with provided rating for specified artist name
- add_artist_country(artist_name, country, database):
  - Input: artist_name (name of artist), country (country of artist), database (path to SQLite database i.e music.db)
  - Output: no return value for this function but inserts country into the Countries table if it doesn’t exist and updates the Artists table by setting country_id to th eID of the inserted/found country for specified artist name
- get_artists_from_db(database):
  - Input: database (path to SQLite database i.e music.db)
  - Output: returns a list of artist names retrieved from Artist table in the database

- get_artist_rating_from_musixmatch(artist_name, api_key):
    - Input: artist_name (name of artist), api_key (API key for accessing MusixmatchAPI)
    - Output: returns rating of the specified artist fetched from Musixmatch API. If the artist is not found or an error occurs, returns a message of "Error or artist not found"
- get_artist_country_from_musixmatch(artist_name, api_key, database):
    - Input: artist_name (name of artist), api_key (API key for accessing MusixmatchAPI), database (path to SQLite database i.e music.db)
    - Output: returns the country of specified artist fetched from Musixmatch API. If artist is not found or an error occurs, returns a message of "Error or artist not found"
- main():
    - Input: none
    - Output: does not have a return value but fetches artist names from the database and checks if their rating and country data is there (data is fetched from Musixmatch API if not), updates the database

Ticketmaster File

- get_artists(conn, cur)
    - Input: Connection and cursor to SQLite database
    - Output: Returns a list of artist names from *Artists* table.
- get_url(root, artist)
    - Input: Root URL to Ticketmaster API and one artist's name

    - Output: Returns the full URL with the artist as a “keyword” parameter.
- get_data(url)
    - Input: Full URL of API request for one artist
    - Output: Returns API response in JSON format. If it’s unable to retrieve data, it returns “Exception!”
- write_json(filename, dict)
    - Input: JSON filename and dictionary of API response data, with the page numbers as the keys and each page’s data as the values
    - Output: None. It writes the dictionary to the file.
- load_json(filename)
    - Input: JSON filename containing retrieved API data
    - Output: Returns file contents. If it’s unable to find and open the file, it returns an empty dictionary.
- cache_all_pages(url, filename)
    - Input: Full URL of API request for one artist, and JSON filename containing retrieved API data
    - Output: None. It writes retrieved data from passed URL to file, or “No data” if the artist has no events.
- event_info(filename)
    - Input: JSON filename containing retrieved API data
    - Output: Returns simplified dictionary with the keys being artist names and the values being a list of dictionaries. Each dictionary contains the city, date, minimum ticket price, and maximum ticket price for one event.

- insert_data(conn, cur, artists)
    - Input: Connection and cursor to SQLite database, and list of artists from the *Artists* table
    - Output: None. It inserts data from event_info() into database tables Events and Cities. If the table has enough data, it'll print "You don't need to add anything more!"
- join_tables(conn, cur)
    - Input: Connection and cursor to SQLite database
    - Output: None. It creates a new table *Events_Final* that joins the main *Events* table with *Cities* and *Artists* to prevent duplicate string data.
- main()
    - Input: None
    - Output: None. It runs get_artists(), insert_data(), join_tables(). It then loops through the data in *Events_Final* and prints each row in sentence format.

<u>Calculations File</u>

- get_top_days(cur, conn)
    - Input: Connection and cursor to SQLite database
    - Output: Returns a dictionary of weekdays with the number of concerts on each weekday.
- get_top_countries(cur, conn)
    - Input: Connection and cursor to SQLite database
    - Output: Returns the top 5 countries based on artists in the database.
- get_top_artists(cur, conn)

  - Input: Connection and cursor to SQLite database
  - Output: Returns the top 15 artists from the *Artists* table.
- concerts_in_midwest(cur, conn)
  - Input: Connection and cursor to SQLite database
  - Output: Returns a list of dictionaries representing events in the *Events_Final* table that occur in popular Midwest cities.

Main File

- build_database()
  - Input: Nothing
  - Output: None. It runs Musixmatch, LastFM, and Ticketmaster files.
- do_calculations()
  - Input: Nothing
  - Output: None. It converts calculations into visual plots.

Resources

Date	Issue Description	Location of Resource	Result
12/3/23	Had trouble indexing into nested retrieved data	https://jsonformatter.org/	JSON formatter displayed data in a hierarchical dictionary with the option to collapse/expand keys
12/9/23	Had trouble inserting only 25 items at a time into a table	https://piazza.com/class/llkxxst2asi496/post/501	Successfully implemented code to insert only 25 at a time
12/11/23	Didn't know how to join 3 SQLite tables	https://stackoverflow.com/questions/11321354/join-3-tables-in-sqlit	Used AND operator in SQlite command to join multiple tables

		e-database	
12/11/23	Exceeded Ticketmaster API quota	https://developer.ticketmaster.com/products-and-docs/apis/getting-started/	We used different emails to get new API keys and make more requests
12/11/23	Trouble Managing Dates	Stack Overflow	Used StackOverflow to learn how to use the datetime library to convert dates into their days of the week (Monday, Tuesday, etc.)
12/12/2023	Trouble managing various SQL databases and writing SQL Queries	Chat GPT	Chat GPT helped write better SQL queries that got the right data and made sure it was stored in the most efficient fashion.