SI 201 | Fall 2025
Final Project Report
Charlotte Norment, Emerson McKay

The goal for this project was to use data gathered from Last.FM and Ticketmaster APIs to find the correlation between artist popularity, the number of concerts they have, and capacity for each venue. From the Last.FM API we planned to gather data on the top artists and how many listeners they had. From Ticketmaster we planned to gather event details, including the artist, event, date, venue capacity, and location. Using this information, we then planned to find the correlation between artists' listener amounts and both the amount of concerts they have and how big the concerts are. We planned to gather this data in a SQL database and then do calculations to find the number of listeners for artists, and then visualize the relationship between artists popularity and concert size using charts.

We successfully completed our goal of using the Last.FM and Ticketmaster data to find out if there was any correlation between artist popularity and the number of concerts each artist has listed on Ticketmaster. We were unable to include any data on venue capacity as we found out that Ticketmaster doesn't actually provide that data about venues from their APIs, so we modified our original plan to not incorporate that. From the Last.FM API we collected artist names and total listener counts for each artist. From the Ticketmaster API we collected event IDs, event names, event dates, artist names, venue IDs, venue names, cities, and states for each event. All data was stored in a single SQLite database across three tables: artists, events, and venues. Data was inserted into the database incrementally with 25 items per run and we avoided duplicate artists in our tables. We then used SQL queries to calculate the number of concerts per artist and the number of unique venues per artist. After getting these numbers, we created visualizations to see how listener counts relate to the number of concerts.

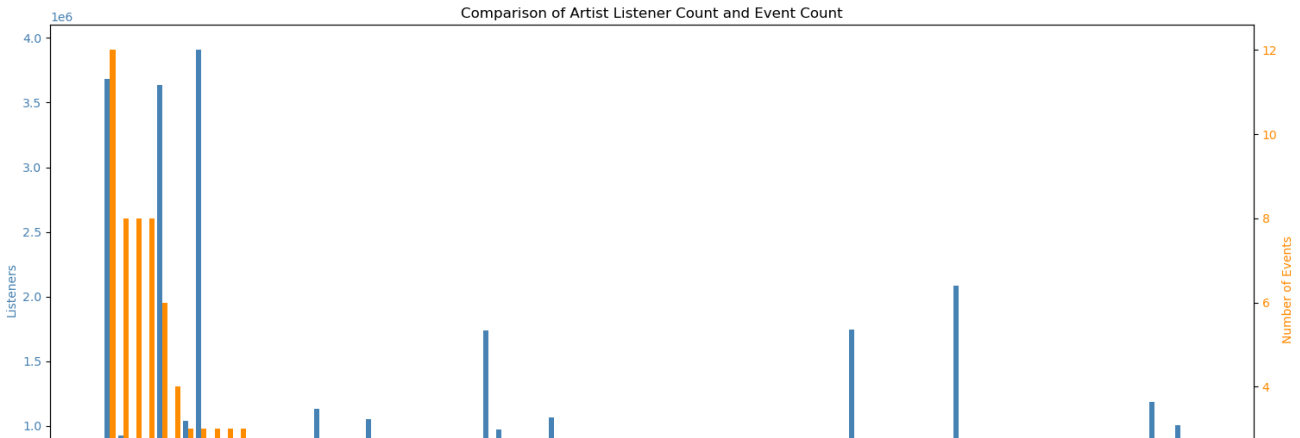
One major challenge we faced was designing a database scheme that made sense for not only the goal that we had in mind, but also the specific requirements of the class. While we avoided duplicate artists in our tables, we did have duplicate venues, cities, and states, leading to feedback about duplicate string data during our grading session. The reason for duplicate strings of venues, cities, and states is due to some artists having multiple shows at the same venue, like for residencies or larger cities requiring multiple shows to meet demand. We did not change those duplicates as that would change the whole objective of our project.

Another significant challenge was working collaboratively with GitHub. We encountered issues with merge conflicts, untracked files, and accidental overwrites which required us to learn how to properly align our files and synchronize our repository. Debugging Git issues was one of the most vital parts of the project to ensure we could work together easily. Despite these challenges, we were able to resolve these issues and build a fully functioning project. A final challenge faced was crafting readable and understandable visualizations. For the first few tries, the visualizations were messy and did not accurately represent our data. With some adjustments to the formatting, colors, labels, etc., we were able to cater the visualizations to best display the data in a way that the audience could easily interpret.

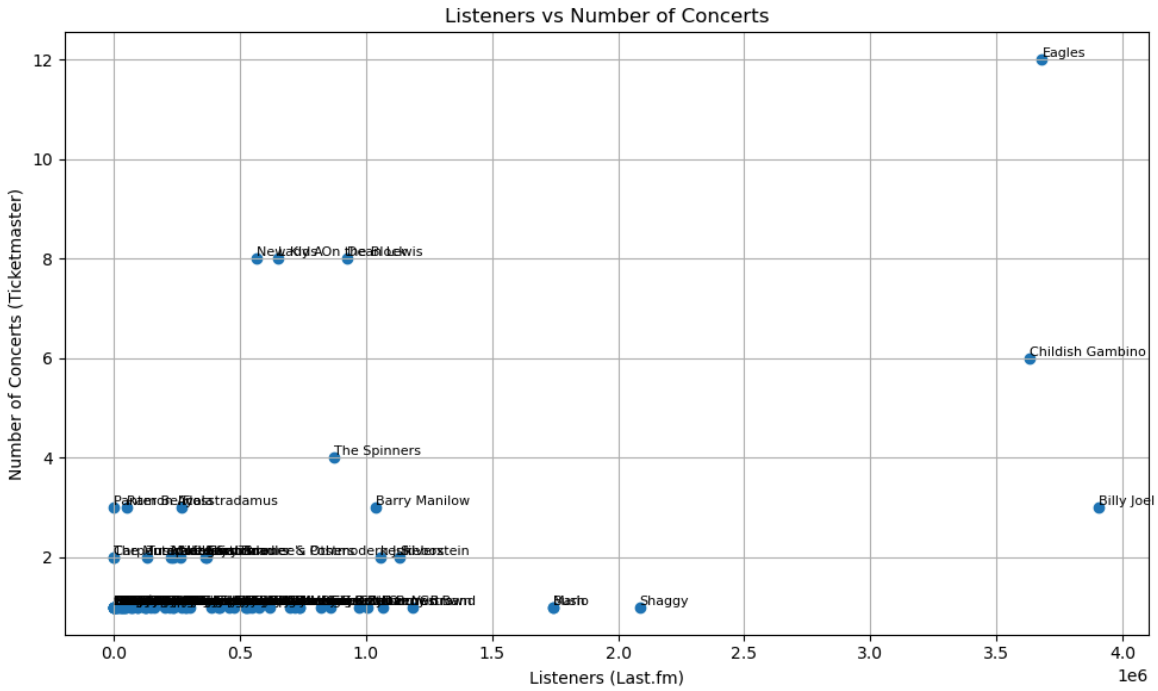
For our calculations, we used SQL queries that joined data from multiple tables in the database and used that to find the total number of concerts per artist and the number of unique venues per artist. These calculations were performed in our calculations.py file using SQL select, count, and join statements.

```
cur.execute("""
    SELECT
        artists.artist_name,
        artists.listeners,
        COUNT(events.event_id) AS num_concerts,
        COUNT(DISTINCT venues.venue_id) AS num_venues
    FROM artists
    LEFT JOIN events ON artists.artist_name = events.artist_name
    LEFT JOIN venues ON events.venue_id = venues.venue_id
    GROUP BY artists.artist_name
    HAVING num_concerts > 0 AND artists.listeners IS NOT NULL
    ORDER BY num_concerts DESC;
""")
```

We created two visualizations for our project using Matplotlib. The first visualization was a bar chart, comparing listener counts (in blue) and the number of events (in orange) for selected artists, allowing for direct visual comparison between popularity and concert activity. Our second visualization was a scatter plot, which



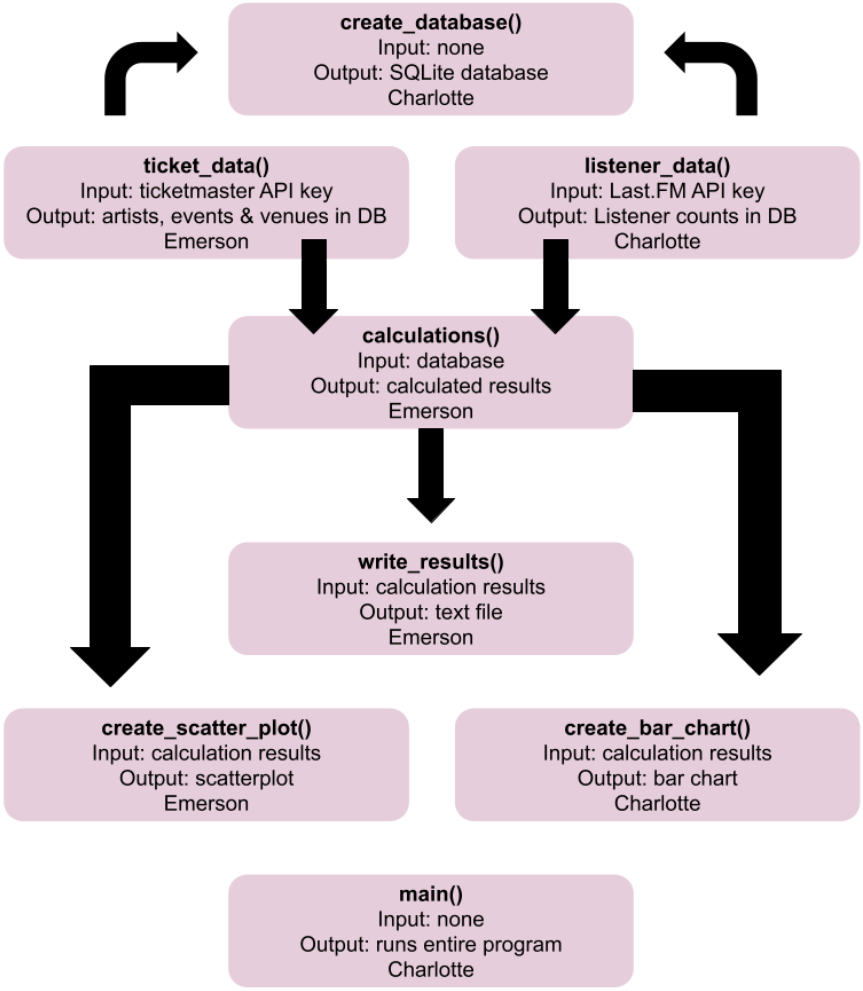
shows the relationship between the number of Last.FM listeners and the number of concerts listed on Ticketmaster. Each point represents an artist.



To run this project, the first step is to clone or download the project repository. Then, navigate to the project files directory. Run the database setup **python Database/create_database.py**, which

will allow for the database to be created. Then, run the Ticketmaster data collection script, **python Data_Collection/ticketmaster_data.py**. This will only generate 25 artists at a time, so it will need to be run multiple times to gather the 100+ rows. Next, run the Last.FM listener data, **python Data_collection/listener_data.py**. After that, run the main file, **python main.py**, to run the calculations, write calculation results to a file, and generate the visualizations.

Final function diagram:



SI 201 | Fall 2025
Final Project Report
Charlotte Norment, Emerson McKay

Resources:

Date:	Issue Description:	Location of Resource:	Result (did it solve the issue?):
11/20	Finding out more information of the Ticketmaster API	Get Started with the Ticketmaster API website	Yes! This provided a lot of useful information on the Ticketmaster API and also helped me get my API key
11/20	Understanding how to use Last.fm and where to find the base url	https://www.last.fm/api/intro	Yes! This introduction to Last.fm page clearly provided the base url that I needed as well as helpful notes and tips for how to work with the API.
12/4	Resolving GitHub merge issues and aligning our files	GitHub documentation and stack overflow	Yes! The GitHub documentation provided good basic information and the stack overflow provided more specific information to help us with the problems we were having
12/11	Getting our visualizations to look neat and organized with the amount of information in them	Matplotlib documentation	The matplotlib documentation helped us to find more ways to make our plots/charts look more appealing