Sabyena Ahmed
Nada El-Fakharany

Final Project Report (SI 201)

GitHub Repository Link: https://github.com/nelfakh-del/si201_final_project.git

1. Our goals for this project included gathering a wide variety of information, storing it all in multiple tables, being able to filter out any unnecessary or repetitive information, and ultimately creating visualizations that can accurately represent our data. We wanted to extract specific information from each of the three APIs and calculate something different within each one. Also, making sure that all of the information is correct, cleanly displayed, and making sure all the separate files collaborate seamlessly.

APIs we worked with:

- PokeAPI - https://pokeapi.co/docs/v2
- Spotify/Spotipy - https://api.spotify.com
- IMDB/OMDB - https://www.omdbapi.com/

Data we planned to gather:

PokeAPI

- Pokémon ID
- Name
- Height
- Weight
- Types

Spotify API

- Track ID
- Track name
- Artist name
- Associated artist ID

OMDB API

- Movie title
- Release year
- Movie genre
- IMDB rating

2. Over the course of the project, we successfully worked with and gathered data from three different APIs:

Sabyena Ahmed
Nada El-Fakharany

- PokeAI - https://pokeapi.co/docs/v2
  Used to collect structured Pokémon data, including names, heights, weights, and type relationships.
- Spotify/Spotipy - https://api.spotify.com
  Used to gather track information such as track IDs, titles, and artist metadata.
- IMDB/OMDB - https://www.omdbapi.com/
  Used to retrieve detailed movie metadata, including genre, release year, and IMDB ratings.

Each API required separate authentication steps, structured requests, and error-handling to gather consistent datasets.

We collected 100 Pokémon entries in batches of 25. For each Pokémon, we gathered:

- id
- name
- height
- weight
- type(s)

We created and populated three relational tables:

- pokemon
- types
- pokemon_types

This data enabled analysis such as:

- average weight per Pokémon type
- number of Pokémon per type

Both of these were also visualized in bar charts.

For Spotify, we collected 100 clean tracks (25 per batch), filtered to remove:

- remixes
- versions
- DJ edits
- playlist/radio hits

The collected track data included:

- track id

Sabyena Ahmed
Nada El-Fakharany

- track title
- artist name (linked via artist_id)

Stored in artists and Spotify. Using this, we produced visuals such as:

- number of collected tracks per artist

This required OAuth authentication and careful filtering to avoid duplicates.

We also gathered 100+ movies from a manually curated list expanded to exceed the minimum requirement.

For each movie, we retrieved:

- title
- year
- genre (primary genre extracted)
- IMDb/OMDb rating

Stored into genres and movies. This allowed us to calculate statistics such as:

- movie counts by genre
- genre distribution (visualized using a pie chart)

We created a full processing pipeline that:

1. Constructed the SQLite database with proper foreign keys and relationships.
2. Populated all tables with clean, deduplicated data across three APIs.
3. Performed calculations such as:
    - average Pokémon weight by type
    - track counts per artist
    - number of movies per genre
4. Generated four visualizations, including:
    - Pokémon avg weight by type (bar chart)
    - Pokémon count by type (bar chart)
    - Spotify: number of tracks per artist (bar chart)
    - Movie genre distribution (pie chart)

All visualizations successfully ran using matplotlib after debugging.

3. Problems we encountered:

Sabyena Ahmed
Nada El-Fakharany

- Expanded the movie and Spotify lists to at least 100 items each. Both lists were updated from around 25 entries to meet the required minimum, as there weren't enough initially.

- There were some issues with running it on Sabyena's end, but eventually, it was sorted out. Mostly, it was issues with committing and running up-to-date information.

- The fourth diagram was implemented and debugged, and all the others were fixed, as there were numerous issues regarding their information, the type of chart showing up, and how only certain APIs had graphs, while others didn't. For example, we had numerous charts with Pokémon information, but none about Spotify/Spotipy.

- Reviewed the OMDB homework and API homework for reference—to remember structure, API handling, and expectations. There were some issues with the formatting and files, but after looking back at previous work, we figured it out.

- The code couldn't read the keys from an external file. After moving keys, we implemented and confirmed a clean loading method.

- Fixed duplicate string entries in the movie list. Repeated movie data was cleaned and deduped.

- Fixed duplicate string entries in the Pokémon type_name field. Type names that appeared multiple times were cleaned up and deduped.

- Pokémon type table showed repeated Pokémon IDs (e.g., 1,1 and 2,2). We looked into the cause of the repeated pairs of IDs and addressed the issue.

- Duplicate string data appeared in the Spotify dataset. Repeated entries were removed and cleaned up.

- We needed to make sure that the PokeAPI tables correctly shared an integer key. We confirmed that the relational structure met the project requirements.

- The database kept returning a "Pokemon table not found" error. We tracked down the issue and resolved the missing table problem.

- Reviewed JOIN usage in the code. We ensured that the required table joins existed and worked correctly.

Sabyena Ahmed
Nada El-Fakharany

- Visualizations needed updates and polishing. All plots were corrected and aligned with the final report.

- The Spotify table included a mixed-letter ID that needed clarification. We investigated the non-numeric ID and determined its meaning and whether it needed cleanup.

- We fixed the manual batching issue by adding a batch-tracking system in the database that automatically determines which set of data to collect next, so each API file continues where it left off without re-collecting duplicate data.

4. Calculations from the data in the database:

```
--- DATABASE CALCULATIONS ---

Average Pokémon weight by type:
('rock', 1587.5)
('ice', 1262.5)
('ground', 832.6)
('fighting', 556.67)
('water', 515.33)
('fire', 496.44)
('psychic', 495.0)
('flying', 363.08)
('steel', 330.0)
('poison', 250.55)
('normal', 233.14)
('electric', 224.8)
('grass', 193.91)
('bug', 164.9)
('fairy', 162.5)
('ghost', 135.67)
```

```
Number of Pokémon per type:
('poison', 31)
('water', 18)
('normal', 14)
('flying', 12)
('grass', 11)
('ground', 10)
('bug', 10)
('fire', 9)
('psychic', 7)
('fighting', 6)
('electric', 5)
('rock', 4)
('fairy', 4)
('ghost', 3)
('steel', 2)
('ice', 2)
```

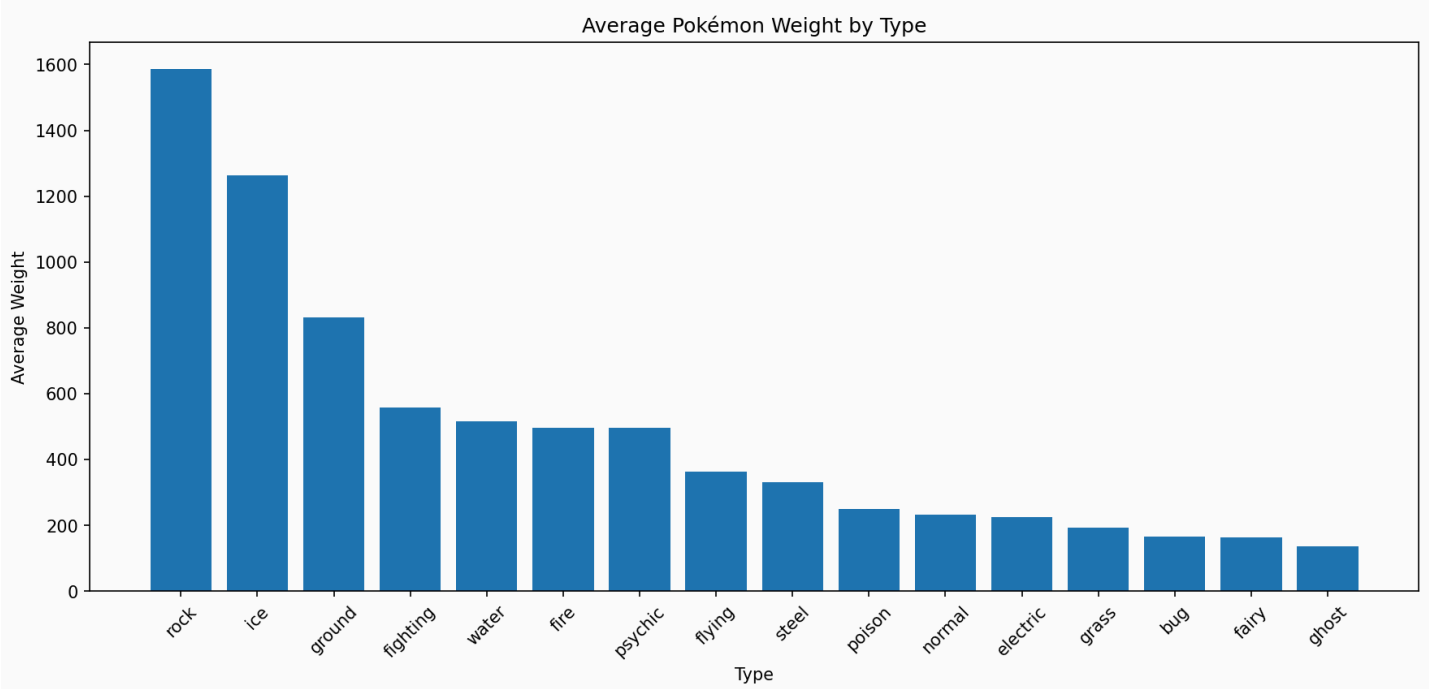
Sabyena Ahmed
Nada El-Fakharany

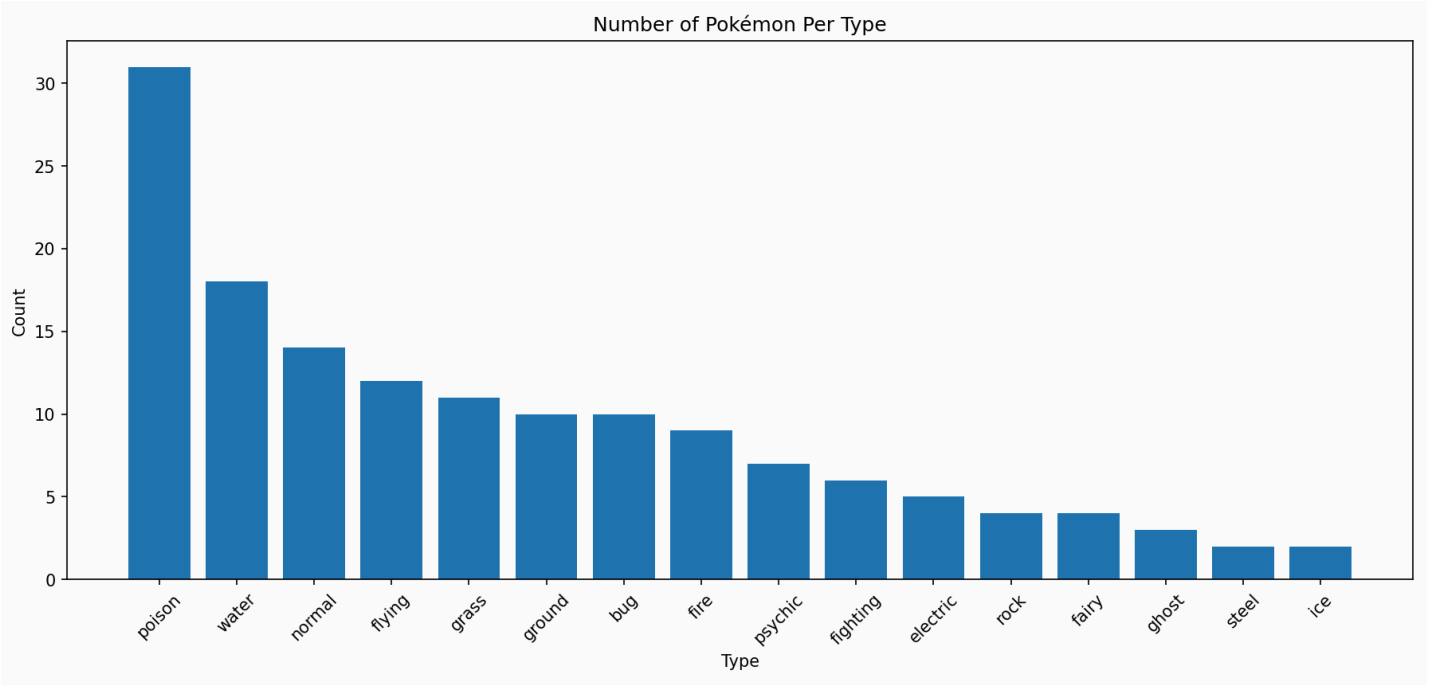
```
Average IMDb rating by genre:
('Crime', 8.77)
('Horror', 8.5)
('Drama', 8.14)
('Action', 7.94)
('Biography', 7.92)
('Adventure', 7.67)
('Animation', 7.65)
('Comedy', 7.52)
('Documentary', 7.45)
('Short', 7.1)

Number of movies per genre:
('Animation', 38)
('Action', 29)
('Drama', 8)
('Adventure', 7)
('Comedy', 5)
('Biography', 5)
('Crime', 3)
('Documentary', 2)
('Short', 1)
('Horror', 1)
```

```
Top 10 artists by number of tracks collected:
('2Pac', 24)
('AMORE', 12)
('Walid Toufic', 8)
('Vijay Yesudas', 4)
('Minh Tuyết', 4)
('Haisam', 4)
('Zeca Baleiro', 3)
('Jason Derulo', 3)
('Coldplay', 3)
('Charli xcx', 3)
```

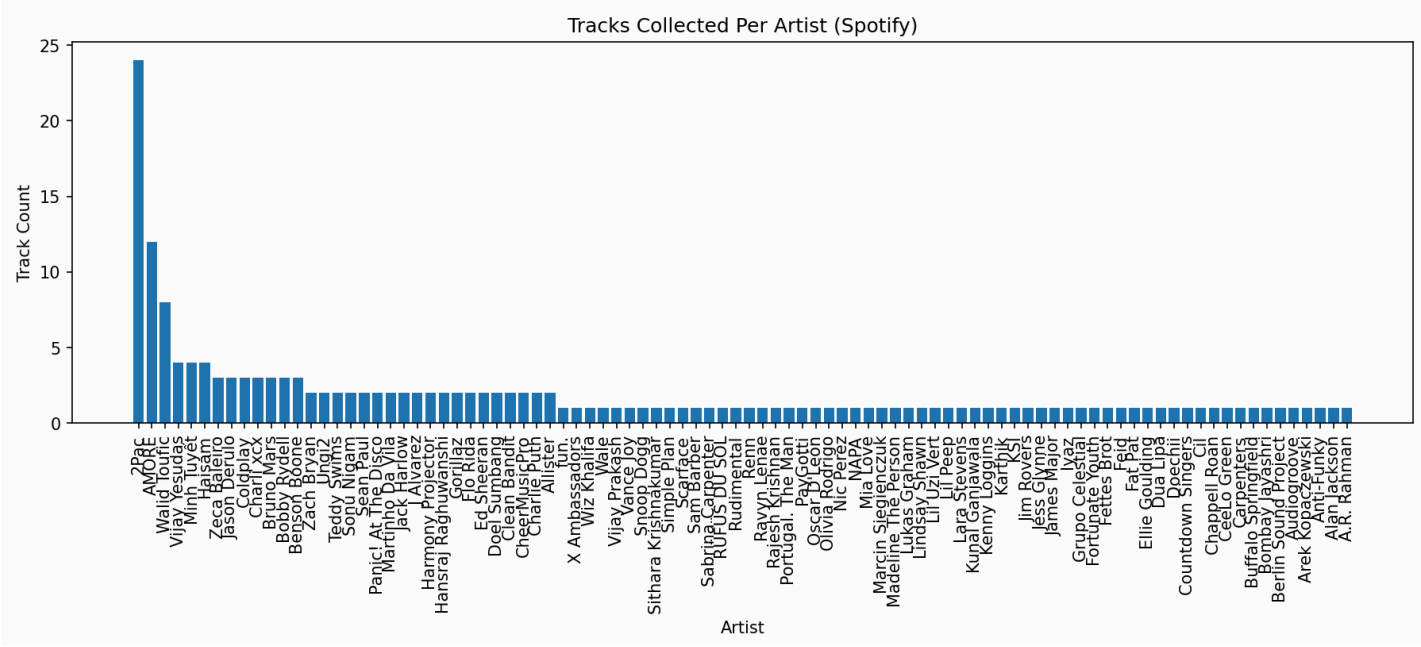
5. Visualizations (in order):

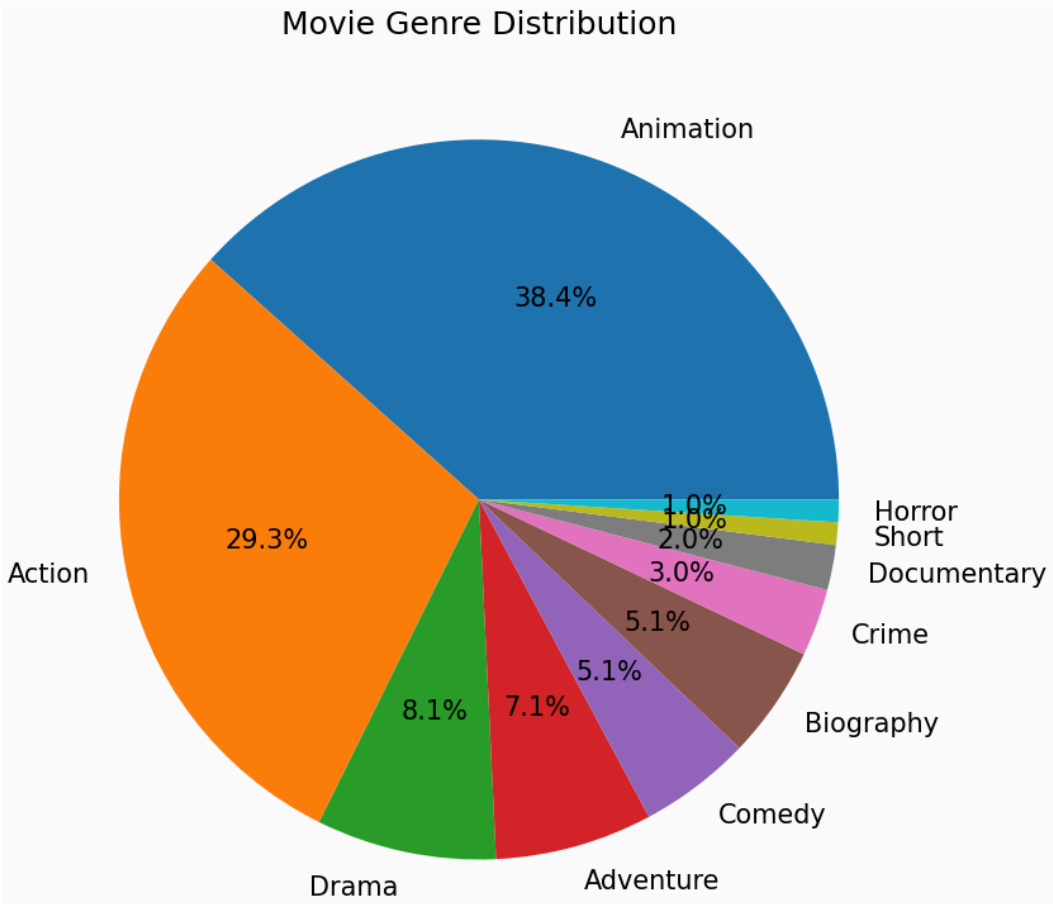
Sabyena Ahmed
Nada El-Fakharany





Sabyena Ahmed
Nada El-Fakharany



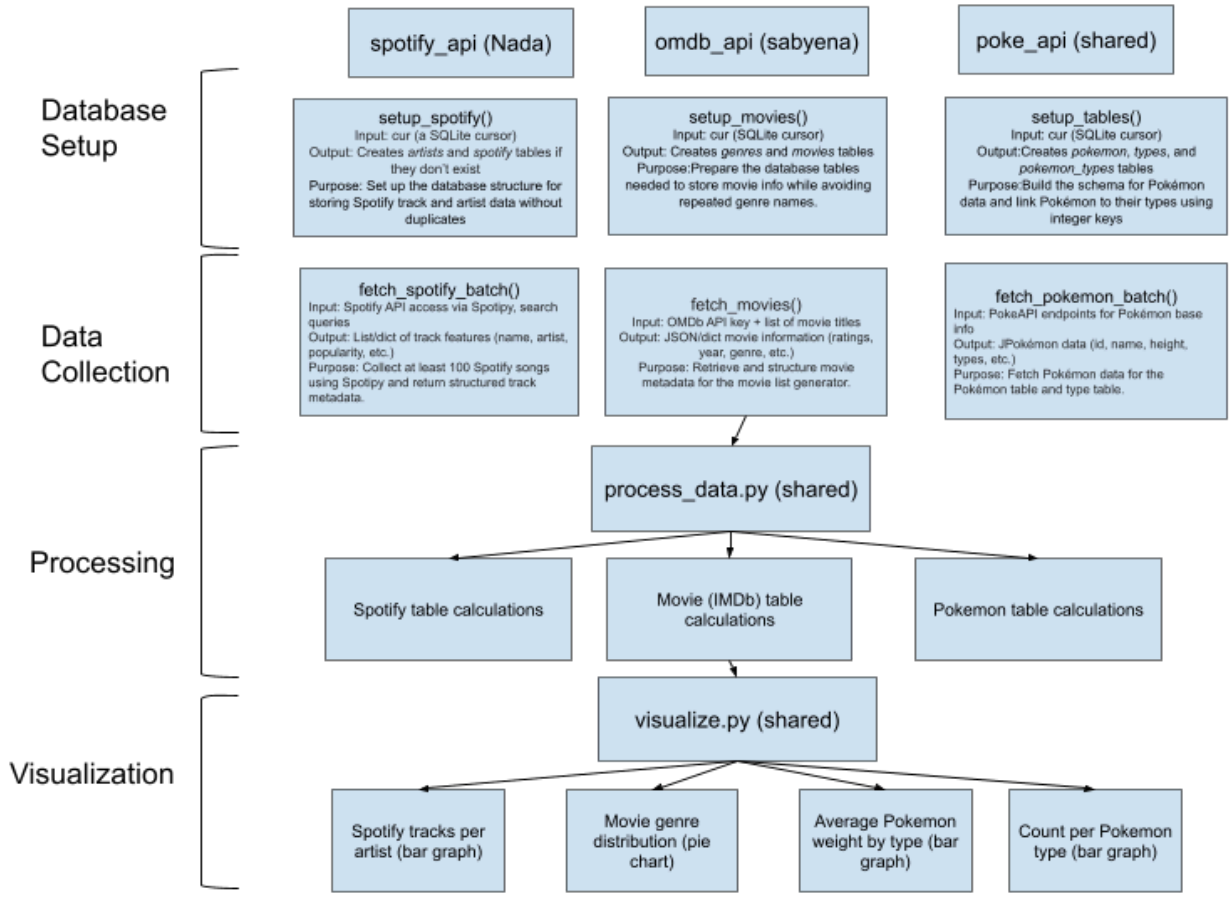


Sabyena Ahmed
Nada El-Fakharany

6. Instructions for running our code:

1. Make sure all files are in the same project folder, and that final.db is in the root directory.
2. Run the API data collection files:
    - omdb_api.py
    - poke_api.py
    - spotify_api.py
3. After all API files have finished running, run:
    - calculations_file.py
4. Next, run:
    - process_data.py
5. Finally, run:
    - visualize.py

7. Updated Function Diagram:



Sabyena Ahmed
Nada El-Fakharany

8. Resource Documentation:

Date (from chat)	Issue Description	Location of Resource	Result (Did it solve the issue?)
2025-12-03	Step-by-step breakdown needed for complex Python/database instructions	ChatGPT	Yes. Instructions were simplified and rewritten with explanations.
2025-12-07	Spotify API returning duplicate strings and non-song data (playlists/mixes)	Spotify Web API docs, ChatGPT	Yes. Filtering and normalized artist tables removed duplicates.
2025-12-07	OMDb API was crashing due to N/A values and missing data	OMDb API documentation, ChatGPT	Yes. Conditional checks prevented invalid inserts.
2025-12-07	Pokémon API confusion regarding repeated Pokémon IDs	ChatGPT	Yes. Clarified many-to-many relationships between Pokémon and types.
2025-12-08	Clarifications needed surrounding visualizations and setup	Matplotlib documentation, ChatGPT	Yes. Helped with understanding the visualization setup.
2025-12-08	Confusion about the purpose of process_data.py and the project requirements	ChatGPT	Yes. The role of calculations, JOINs, and the separation of concerns was clarified.
2025-12-11	Organizing layout and summarizing descriptions of the report	ChatGPT	Yes. The layout was organized nicely and clearly, and descriptions were made concise.

Sabyena Ahmed
Nada El-Fakharany

2025-12-13	Confusion about the calculations file and its contents	ChatGPT	Yes. Removed confusion and helped with the setup of code.
2025-12-13	Clarifications on how to change the code (in api files) to remove the manual aspect from it in each run	ChatGPT	Yes. Clarified how to set up and change code to run each batch of 25 without manually changing anything.