685.652, Data Engineering Principles & Practice, Spring 2025

Group 6

Devavrath Hemantha Kumara – dhemant1@jh.edu

Shiv Palit – spalit2@jh.edu

Yashank Dasrapuria – ydasrap1@jh.edu

Zachary Barrett – zbarret1@jh.edu

Final Project – Cross Platform Music Analytics

April 22, 2025

Table of Contents

Group 6 – Final Project

Note on Instructions

See the section called "Instructions - Getting API Keys, Building the Container, Running the Application" for detailed instructions on which API keys need to be generated and how to do so.

A demonstration video is also located at: https://youtu.be/fa22fU5Ab-U

If you have any issue building the container, or running the DAGs, please send an e-mail to the group and one of us will get in touch as soon as possible. Our e-mails are listed on the cover sheet.

Project Overview

Our project explores how data from several widely used data sources – Spotify, Last.fm, Kaggle, and AcousticBrainz – can be integrated to provide a comprehensive view of track-level insights. Each dataset offers distinct types of information: Last.fm contributes listener behavior and genre tagging data via its public API; Spotify provides excellent track metadata and popularity metrics; the Kaggle dataset provides Billboard Hot 100 chart track information spanning back several decades; and AcousticBrainz supplies more detailed data on audio characteristics for the tracks. While each dataset is valuable on its own, combining them enables a deeper analysis of trends across genres, artists, and track attributes like date released, peak chart position, duration, danceability, and so on.

The primary goal of the project is to clean, normalize, and join these sources to enable meaningful cross-platform comparisons and analysis. This includes identifying and resolving inconsistencies in key fields, such as artist names and track titles, and implementing strategies for joining datasets using ISRC codes, MusicBrainz ID's (MBID's), or composite keys based on normalized text fields.

To support this, we developed a fully containerized data pipeline using Apache Airflow, PostgreSQL, and a Flask based web API. Airflow orchestrates the ingestion of data from the Last.fm API and manages scheduled data transformation tasks. PostgreSQL serves as the central relational database, storing all processed and normalized data across multiple tables. The Flask API allows users to access and interact with the data through simple, readable endpoints. The pipeline is designed to run end to end with minimal manual intervention and is fully reproducible through Docker.

File Organization

All scripts are contained within the project folder (652-group6). Feel free to explore the folder and file structure. Python scripts related to the DAGs are located in the airflow/dags folder. PDF and/or HTML versions of this writeup and some of the scripts are located in the documentation folder.

As the DAGs are run and data is retrieved from the APIs, CSVs get automatically generated and placed into the airflow/data folder with a timestamp. Examples of the data are placed into that folder already for your review. New files will be generated as the DAGs are run.

Datasets

This project integrates data from four distinct sources: the Last.fm API, the Spotify API, the AcousticBrainz API, and a specific Kaggle dataset through the Kaggle API. Each dataset was selected based on its relevance to track-level insights and its compatibility with other sources through shared identifiers such as identifier codes, artist names, and track titles. A brief overview of the data retrieved is below.

1. Last.fm Top Tags (API)

- **Source:** Last.fm API – https://www.last.fm/api
- **Access Method:** HTTP requests using a registered API key
- **Endpoint(s) used:** "tag.getTopTags" – (https://www.last.fm/api/show/tag.getTopTags)
- **Description:** Provides a list of the most-used tags on the platform. Users are able to tag tracks. Provides the top 50 most used user-generated tags (e.g., "rock", "pop", "electronic"). These tags are used to later retrieve the top tracks for that tag.

2. Last.fm Top Tag Tracks (API)

- **Source:** Last.fm API – https://www.last.fm/api
- **Access Method:** HTTP requests using a registered API key
- **Endpoint(s) used:** "tag.getTopTracks" – (https://www.last.fm/api/show/tag.getTopTracks), "track.getInfo" – (https://www.last.fm/api/show/track.getInfo)
- **Description:** A list of top tracks associated with a specific tag (e.g., "rock", "pop", "electronic") is retrieved via tag.getTopTracks. Not a perfect measure for popularity, but a good way to get basic metadata for many popular tracks in batches. This track info is then used to query the track.getInfo endpoint to get more detailed info for each track. Note that track.getInfo only allows retrieval of detailed info for one track per request. No better batch retrieval is available for the last.fm API.

3. Spotify Track Features (API)

- **Source:** Spotify Web API – (https://developer.spotify.com/documentation/web-api)
- **Access Method:** HTTP requests using registered credentials (OAuth)
- **Endpoint(s) Used:** Search – (https://developer.spotify.com/documentation/web-api/reference/search), Playlists – (https://developer.spotify.com/documentation/web-api/reference/get-playlists-tracks)
- **Description:** Playlist IDs are retrieved using the search endpoint, and then detailed information for each of the tracks within the playlists are retrieved using the playlists endpoint. Retrieves detailed track metadata such as artist names, track titles, and ISRC codes.

Note from the group: For years, Spotify provided an endpoint "audio-features" which included detailed audio features from each track (time signature, energy, tempo, etc.). Disappointingly, this endpoint was deprecated to new users in November 2024. See (https://developer.spotify.com/documentation/web-api/reference/get-audio-features). This certainly would provide a lot of interesting data quickly, but it is no longer available from Spotify. There are datasets (like from Kaggle) where people have stored and made available this data, and there are also other paid sites making this kind of data available. Instead, we opted to query the AcousticBrainz database, which has similar data, though not all data is included for all

tracks necessarily. Interestingly, AcousticBrainz also stopped accepting new data in 2022, but the API is still available to be called.

4. Billboard Hot 100 Chart Data (Kaggle)

- **Source:** Kaggle API
- **Access Method:** API call to a specific dataset, retrieves a zip file. (https://www.kaggle.com/datasets/elizabethearhart/billboard-hot-1001958-2024, sourced from https://github.com/utdata/rwd-billboard-data/tree/main)
- **Description:** The CSV retrieved from this API call contains information about tracks in the Billboard Hot 100 chart from 1958 to 2024. Song and artist names are included, along with current and historical chart positions.

5. AcousticBrainz High-Level Audio Features (API)

- **Source:** AcousticBrainz Web API – (https://acousticbrainz.org/data#api-reference, https://acousticbrainz.readthedocs.io/api.html)
- **Access Method:** HTTP requests
- **Endpoint:** High-level recording data – (https://acousticbrainz.org/api/v1/high-level)
- **Description:** Retrieves track metadata and more detailed audio features. These batch requests use the MBID's generated from the last.fm track retrieval.

Instructions - Getting API Keys, Building the Container, Running the Application

Instructions are contained herein and also within the README.md inside the project directory. The general steps to build the container and run the application are as follows:

1. Extract the repository zip to a location of your choosing.
2. Navigate to the data sources' websites to register accounts and get API keys to input into variables.json within the airflow directory in the project folder.
3. Build the container using docker-compose.
4. Open Airflow on the designated port, and trigger the DAGs.
5. Optionally, open pgAdmin to query some of the tables.
6. Open the flask application and query the database.

Detailed instructions for steps are outlined below.

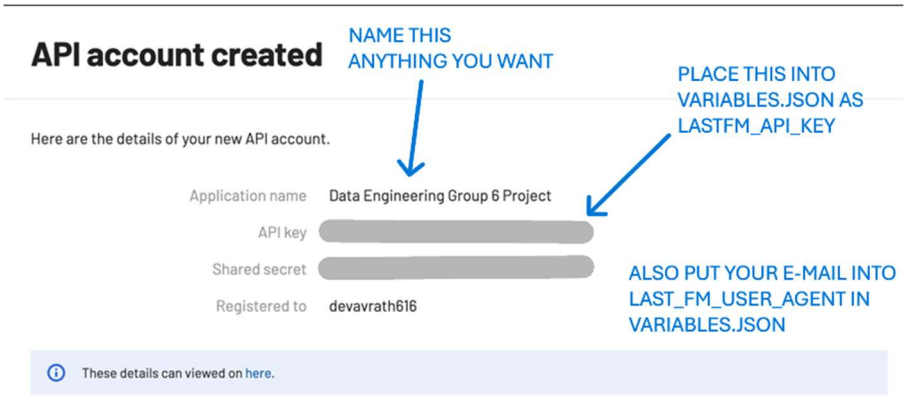
Step 1 - Getting API Credentials

Before running the pipeline, you will need to retrieve API credentials and ensure all services are properly configured via Docker. This section outlines the steps required to get the project running end to end. While our team developed the project using GitHub for version control and collaboration, this submission is designed to be self-contained and reproducible by the reviewer.

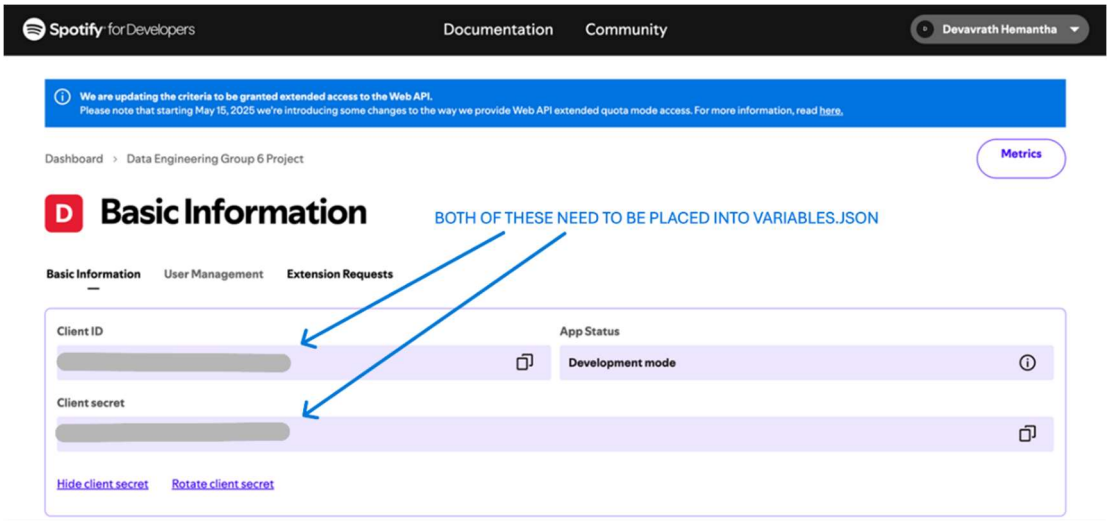
You will need to get API credentials for 3 data source API's – Last.fm, Spotify, and Kaggle. This will require registering an account and creating an "application" in some cases.

1. Last.fm API Key
    a. Register for a Last.fm account at (https://www.last.fm/api)

b. Click on "Get an API account" (https://www.last.fm/api/account/create)
c. Name your application something suitable and register it.
d. After registering an application, you will receive an API key. It can later be viewed at https://www.last.fm/api/accounts. See image below for what to put into variables.json in the airflow folder before composing the docker file.



2. Spotify API Credentials
   a. Create an account as required and log in to https://developer.spotify.com
   b. Click your account in the top right and go to "Dashboard" (https://developer.spotify.com/dashboard). Click "Create App".
   c. Put something reasonable into "App name" and "App description". Also put something into "Redirect URIs", like https://www.spotify.com. Check the "Web API" box below that, then agree to the terms and save.
   d. You will be brought to the "Basic Information" page. Click on "View client secret".
   e. Note your Client ID and Client Secret - these will be used to authenticate and retrieve access tokens. Only basic OAuth tokens are used for the endpoints we are using. Place these credentials into variables.json as "SPOTIFY_CLIENT_ID" and "SPOTIFY_CLIENT_SECRET".



3. Kaggle Username and Kaggle Key
    a. Create an account as required at (https://www.kaggle.com/account/login). Make a note of the username that you create. It can be viewed later at (https://www.kaggle.com/settings).
    b. Once you are registered (following verification code), go the settings link noted above, scroll down to "API", and click "Create New Token". A file called **'kaggle.json'** will be downloaded to your computer. View that file and place the username and key into variables.json as "KAG_USERNAME" and "KAG_KEY".

For each of the credentials mentioned above, place them into the appropriate field of the variables.json file as directed.

Step 2 – Build and Launch the Project

Docker Desktop is recommended to be used. Open Docker Desktop and stop the "jhu_docker" being used for other parts of the course if you have it running. Then, within a terminal in Docker Desktop (e.g. Windows Powershell or Command Prompt, etc., navigate to the location that you extracted the "652-group6" project repository to. For example, "cd Downloads/652-group6" or whatever path is required.

Once within the project directory, launch all services by running the command:

```
docker-compose up --build –d
```

This will:

- Set up the PostgreSQL database (used for both Airflow metadata and pipeline storage)
- Initialize Airflow with your DAGs and variables
- Launch the Flask API on port 5001
- Start all containers in the background

If you have issues at any point with the Docker container, it is recommended to compose down:

docker-compose down -v

And then compose up:

docker-compose up --build –d

Interim Step – Access All Container Services

1. Airflow UI: Within a browser, go http://localhost:8081. Or simply click on it within Docker.
    a. Username: group6
    b. Password: group6
2. PostgreSQL Access via pgAdmin
    a. Open pgAdmin. Within "Object Explorer", right-click "Servers" and then "Register" and "Server". You can name the server "group6" and ensure under "Connection" that it is on port 5432.
    b. Host name: localhost
    c. Maintenance database: group6
    d. Username: group6

   e. Password: group6
3. Flask API:
   a. Within your browser (or just clicking from Docker), go to http://localhost:5001/merged-tracks or http://localhost:5001/cross-platform-consensus
   b. For the JSON view of the data in the browser, go to http://localhost:5001/merged-tracks-data or http://localhost:5001/cross-platform-consensus-data
   c. Using the flask_api/flask_api_demo.ipynb jupyter notebook.

Step 3 – Trigger DAGs in the Pipeline

Once Airflow is running, the DAGs can be triggered from Airflow. Note that some of the DAGs will take some time due to the rate limiting requirements for the Spotify and Last.fm API's. The general expected runtimes are mentioned below.

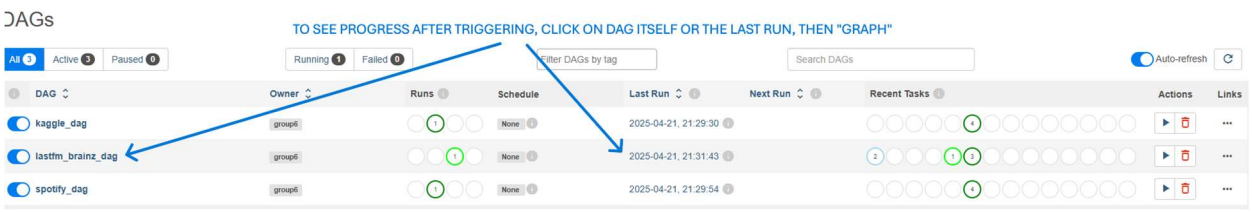
Once Airflow is running:

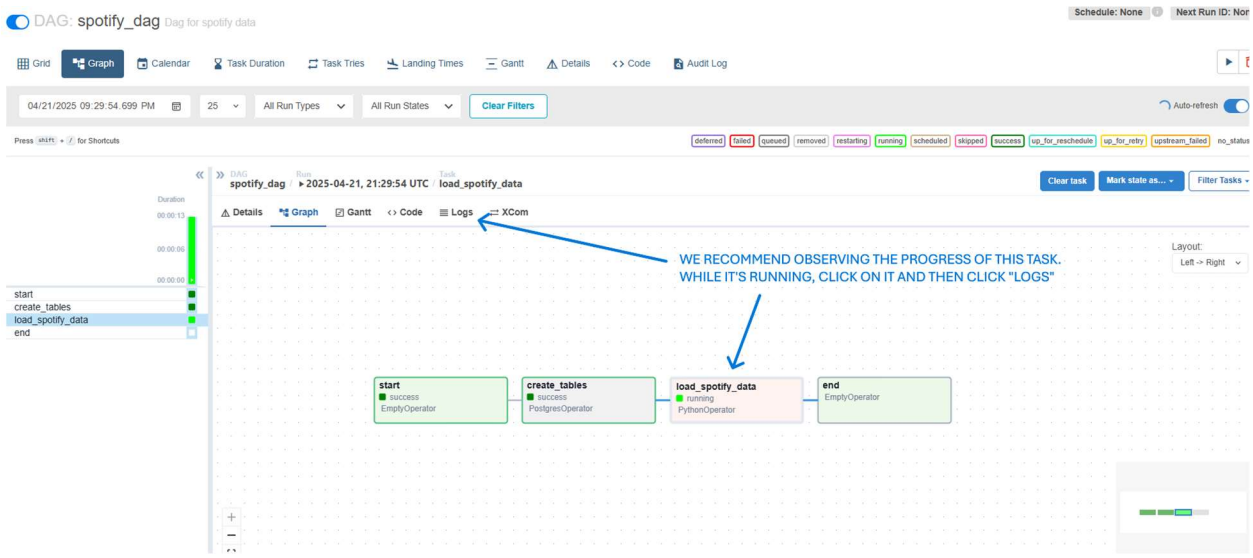
1. Log in to the Airflow UI (https://localhost:8081)
2. Locate the DAGs for data collection and transformation. Trigger each of the following DAGs by clicking the play button. It is recommended that they be triggered sequentially (wait until one is finished then trigger the next) in this order.
   a. kaggle_dag – Trigger 1st - Expected runtime – quick, maybe a few seconds.
   b. spotify_dag – Trigger 2nd - Expected runtime – About a minute.
   c. lastfm_brainz_dag – Trigger 3rd - Expected runtime – About 7 minutes.

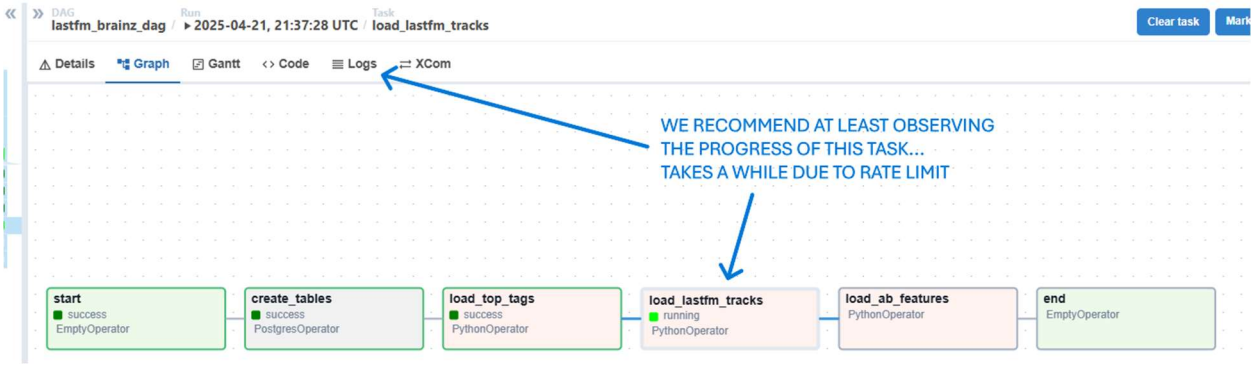
Each DAG will extract data from the associated API(s), transform the data through cleaning, and then load the data into the PostgreSQL database. Delays are introduced into the code to comply with either published rate limits, or in the case of Last.fm, the expected rate limit based on research. The last.fm DAG is the biggest constraint on time, due to the rate limit and the fact that retrieval of detailed track information can only occur through individual rather than batch retrieval.

It is recommended to view the logs for tasks within each DAG during or after the execution of the DAG. Specifically, it is recommended to view the logs for "load_spotify_data" (within the spotify_dag) and the "get_and_load_lastfm_tracks" (within the lastfm_brainz_dag) while they are executing, to view the progress, especially since the last.fm DAG takes a while.

Within Airflow, click on "DAGS", and then trigger each DAG sequentially per the instructions. View progress as you like by clicking on the DAG and viewing the graph. A few Operators we recommend watching the logs for are shown below:







Step 4 – Interact with the Database via API and/or pgAdmin

After running the DAGs, the transformed and matched data will be available in the PostgreSQL database.

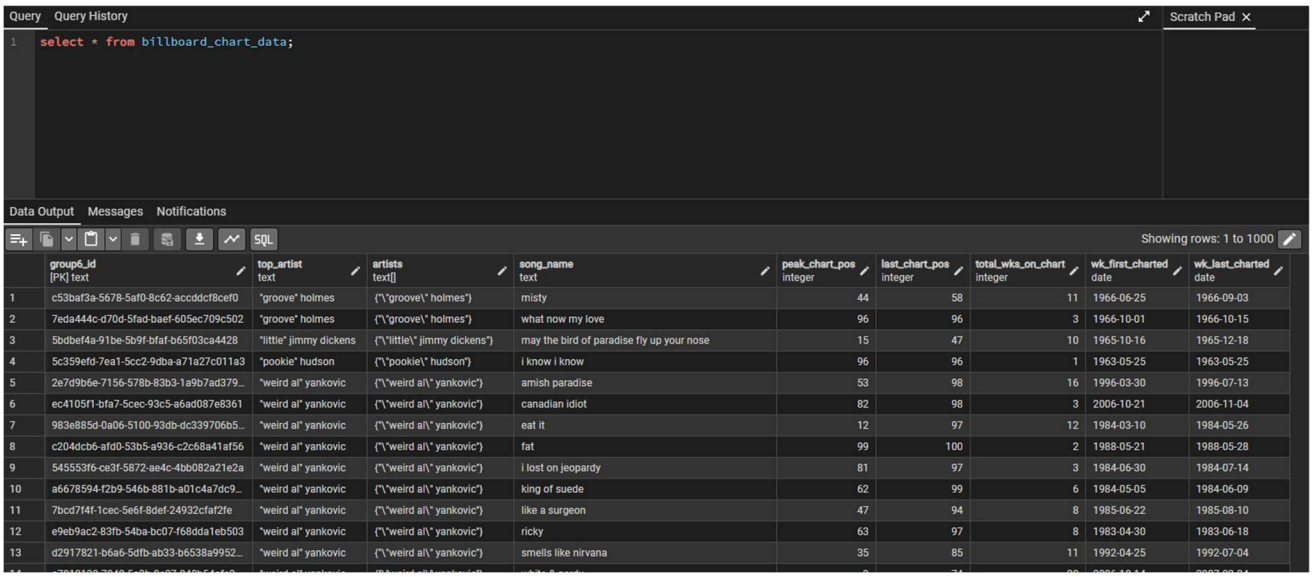
1. To interact with the data, Flask API provides browser-accessible endpoints.
    a. Open the flask API in your browser at http://localhost:5001/merged-tracks
    b. You should see a table of joined data from Spotify, Last.fm and Kaggle.
    c. Filter by artist dynamically to refine your search.
2. To see the JSON data in the terminal output:
    a. Run the command docker-compose logs -f flask_api
    b. This prints the raw JSON data (used in the HTML tables) to the terminal for debugging or inspection.

Data Transformation and Cleaning

A brief overview of the transformations made to the data before loading into the database is discussed here. First, we made the decision not to keep every attribute from every endpoint. We transform the data as needed to generate columns that we generally find interesting for inclusion in each table. A few of the high-level decisions we made for cleaning and transforming the data retrieved from the APIs are noted below.

1. Kaggle – Billboard Hot 100 Chart Data – Table "billboard_chart_data"
    a. De-Duplication – With 300,000+ rows, we decided to de-duplicate by song name and artist, resulting in just over 30,000 rows.
    b. New columns – Instead of keeping a given track's position every time it is on the chart, the transformation determines the first time and last time it is on the chart, and columns are renamed as such.
    c. Artist and song names are stripped and made lowercase.
    d. Other columns are made into an integer or date as appropriate to insert into the database.
    e. A unique key is given to the track by hashing the Song and Artist name, which acts as the Primary Key and is the joining key across all song tables

Sample records from the table are shown below.



Query | Query History | Scratch Pad

```
select * from billboard_chart_data;
```

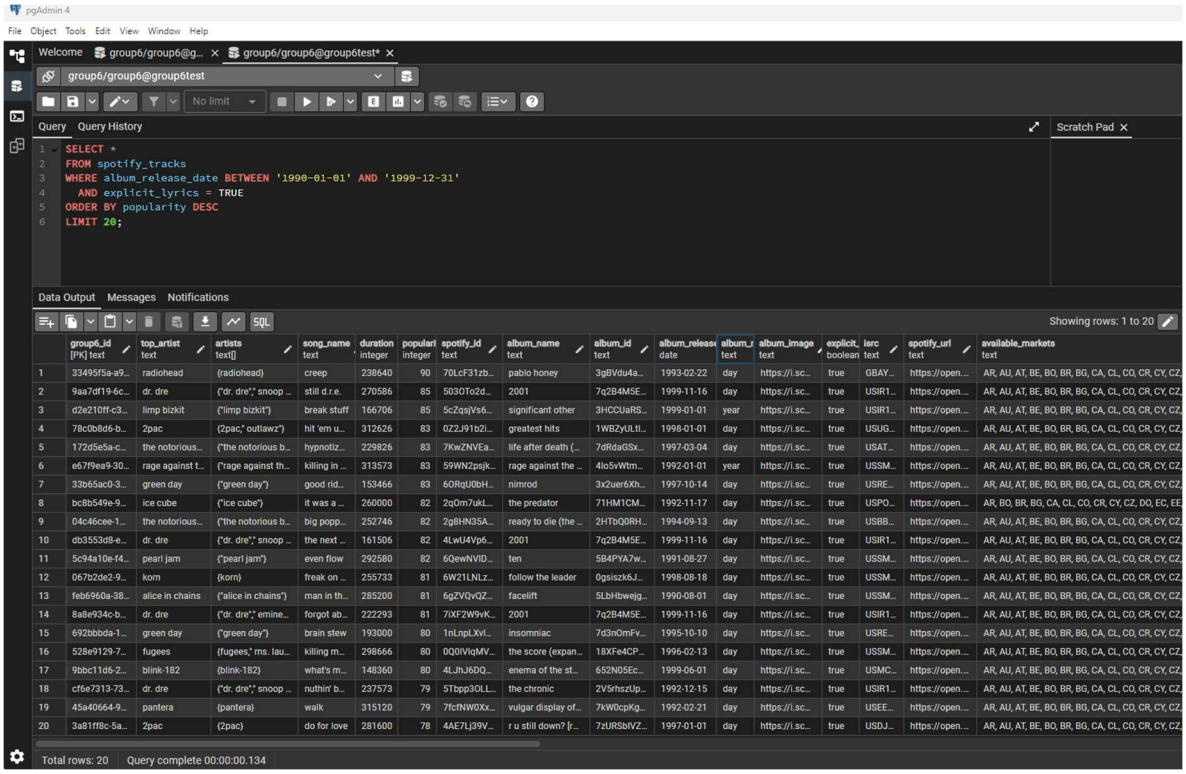
Data Output | Messages | Notifications

Showing rows: 1 to 1000

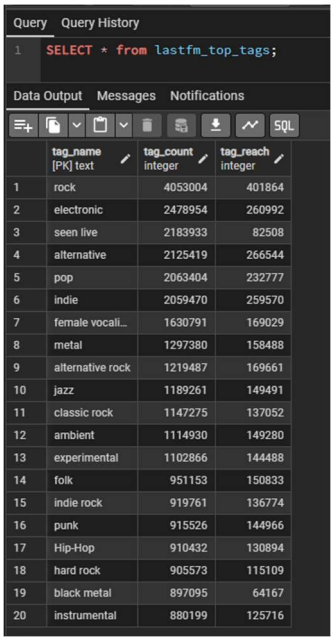
	group6_id [PK] text	top_artist text	artists text[]	song_name text	peak_chart_pos integer	last_chart_pos integer	total_wks_on_chart integer	wk_first_charted date	wk_last_charted date
1	c53baf3a-5678-5af0-8c62-accddcf8cef0	"groove" holmes	{"\"groove\" holmes"}	misty	44	58	11	1966-06-25	1966-09-03
2	7eda444c-d70d-5fad-baef-605ec709c502	"groove" holmes	{"\"groove\" holmes"}	what now my love	96	96	3	1966-10-01	1966-10-15
3	5bdbef4a-91be-5b9f-bfaf-b65f03ca4428	"little" jimmy dickens	{"\"little\" jimmy dickens"}	may the bird of paradise fly up your nose	15	47	10	1965-10-16	1965-12-18
4	5c359efd-7ea1-5cc2-9dba-a71a27c011a3	"pookie" hudson	{"\"pookie\" hudson"}	i know i know	96	96	1	1963-05-25	1963-05-25
5	2e7d9b6e-7156-578b-83b3-1a9b7ad379...	"weird al" yankovic	{"\"weird al\" yankovic"}	amish paradise	53	98	16	1996-03-30	1996-07-13
6	ec4105f1-bfa7-5cec-93c5-a6ad087e8361	"weird al" yankovic	{"\"weird al\" yankovic"}	canadian idiot	82	98	3	2006-10-21	2006-11-04
7	983e885d-0a06-5100-93db-dc339706b5...	"weird al" yankovic	{"\"weird al\" yankovic"}	eat it	12	97	12	1984-03-10	1984-05-26
8	c204dcb6-afd0-53b5-a936-c2c68a41af56	"weird al" yankovic	{"\"weird al\" yankovic"}	fat	99	100	2	1988-05-21	1988-05-28
9	545553f6-ce3f-5872-ae4c-4bb082a21e2a	"weird al" yankovic	{"\"weird al\" yankovic"}	i lost on jeopardy	81	97	3	1984-06-30	1984-07-14
10	a6678594-f2b9-546b-881b-a01c4a7dc9...	"weird al" yankovic	{"\"weird al\" yankovic"}	king of suede	62	99	6	1984-05-05	1984-06-09
11	7bcd7f4f-1cec-5e6f-8def-24932cfaf2fe	"weird al" yankovic	{"\"weird al\" yankovic"}	like a surgeon	47	94	8	1985-06-22	1985-08-10
12	e9eb9ac2-83fb-54ba-bc07-f68dda1eb503	"weird al" yankovic	{"\"weird al\" yankovic"}	ricky	63	97	8	1983-04-30	1983-06-18
13	d2917821-b6a6-5dfb-ab33-b6538a9952...	"weird al" yankovic	{"\"weird al\" yankovic"}	smells like nirvana	35	85	11	1992-04-25	1992-07-04

2. Spotify Tracks – Table "spotify_tracks"
    a. De-Duplication – We are pulling tracks from multiple playlists that sometimes have the same tracks. In the case of Spotify, we found that the tracks very consistently came with only one International Standard Recording Code (ISRC), and that was a good way to de-duplicate for Spotify.
    b. Not all attributes are kept for database (e.g. only one image album image URL is kept).
    c. Artist, song, and album names are stripped and made lowercase.
    d. A unique key is given to the track by hashing the Song and Artist name, which acts as the Primary Key and is the joining key across all song tables

A sample query for popular explicit songs from the 1990's is shown below:

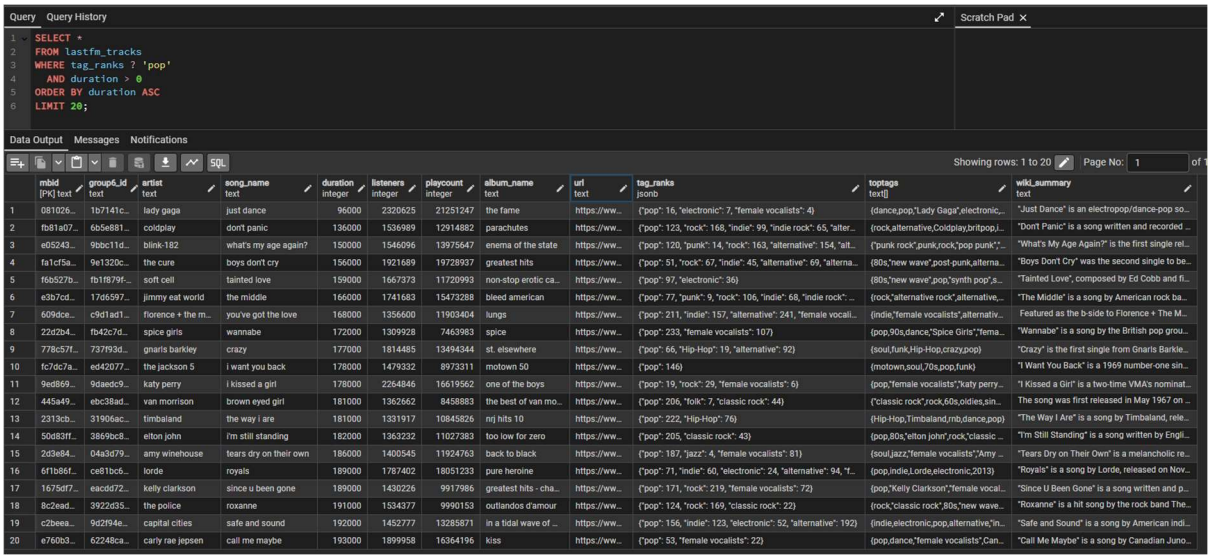


3. Last.fm Top Tags – Table "lastfm_top_tags"
    a. Consistent enough data that little to no transformation is required.



	tag_name [PK] text	tag_count integer	tag_reach integer
1	rock	4053004	401864
2	electronic	2478954	260992
3	seen live	2183933	82508
4	alternative	2125419	266544
5	pop	2063404	232777
6	indie	2059470	259570
7	female vocali...	1630791	169029
8	metal	1297380	158488
9	alternative rock	1219487	169661
10	jazz	1189261	149491
11	classic rock	1147275	137052
12	ambient	1114930	149280
13	experimental	1102866	144488
14	folk	951153	150833
15	indie rock	919761	136774
16	punk	915526	144966
17	Hip-Hop	910432	130894
18	hard rock	905573	115109
19	black metal	897095	64167
20	instrumental	880199	125716

4. Last.fm Track Info – Table “lastfm_tracks”
    a. The MBID (MusicBrainz ID) acts as the Primary Key for this data.
    b. De-Duplication – In the case of last.fm, de-duplication by MBID was determined to be excellent for removing duplicate songs.
    c. Tag ranks – Since songs often appear in the top rankings for multiple tags, a column tag_ranks is created, which shows the ranking for each rag where that tag appears in the top tracks for the tag.
    d. Again, not all attributes are kept for the database.
    e. Artist, song, and album names are stripped and made lowercase.
    f. A unique key is given to the track by hashing the Song and Artist name, which is the joining key across all song tables



Query Query History

```
SELECT *
FROM lastfm_tracks
WHERE tag_ranks ? 'pop'
  AND duration > 0
ORDER BY duration ASC
LIMIT 20;
```

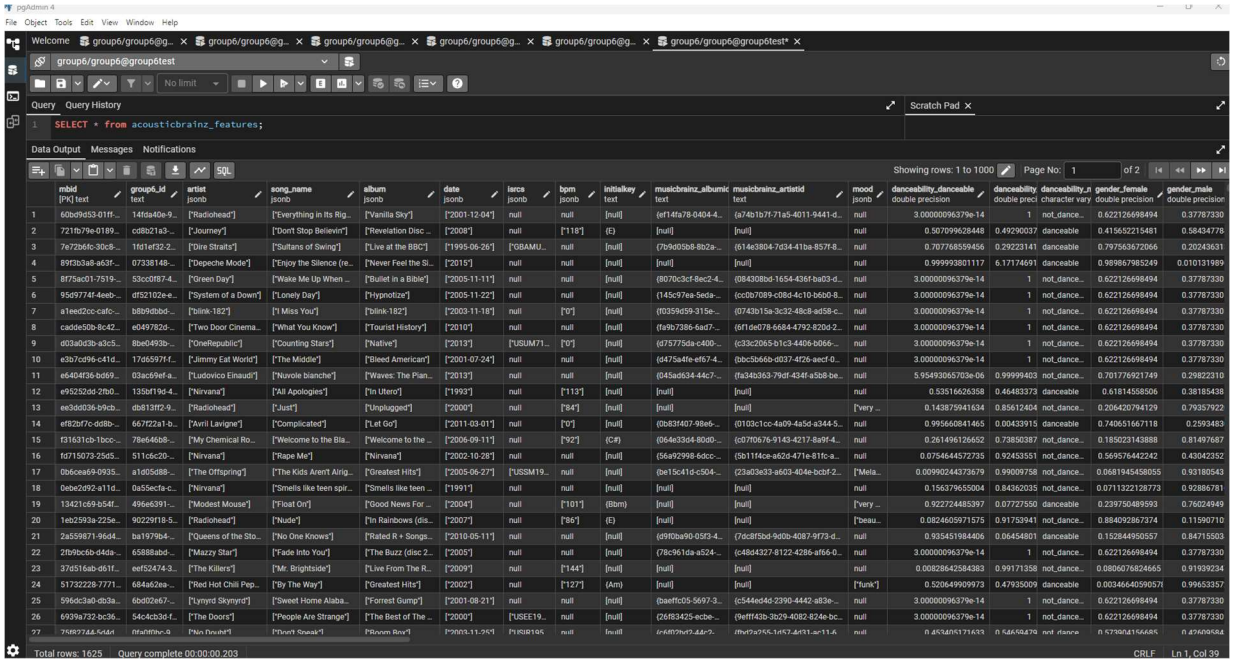
Scratch Pad

Data Output Messages Notifications

Showing rows: 1 to 20 Page No: 1

	mbid [PK] text	group6_id text	artist text	song_name text	duration integer	listeners integer	playcount integer	album_name text	url text	tag_ranks jsonb	toptags text[]	wiki_summary text
1	081026...	1b7141c...	lady gaga	just dance	96000	2320625	21251247	the fame	https://ww...	{"pop": 16, "electronic": 7, "female vocalists": 4}	{dance,pop,"Lady Gaga",electronic,...	"Just Dance" is an electropop/dance-pop so...
2	fb81a07...	6b5e881...	coldplay	don't panic	136000	1536989	12914882	parachutes	https://ww...	{"pop": 123, "rock": 168, "indie": 99, "indie rock": 65, "alter...	{rock,alternative,Coldplay,britpop,i...	"Don't Panic" is a song written and recorded ...
3	e05243...	9bbc11d...	blink-182	what's my age again?	150000	1546096	13975647	enema of the state	https://ww...	{"pop": 120, "punk": 14, "rock": 163, "alternative": 154, "alt...	{"punk rock",punk,rock,"pop punk",'...	"What's My Age Again?" is the first single rel...
4	fa1cf5a...	9e1320c...	the cure	boys don't cry	156000	1921689	19728937	greatest hits	https://ww...	{"pop": 51, "rock": 67, "indie": 45, "alternative": 69, "alterna...	{80s,"new wave",post-punk,alterna...	"Boys Don't Cry" was the second single to be...
5	f6b527b...	fb1f879f-...	soft cell	tainted love	159000	1667373	11720993	non-stop erotic ca...	https://ww...	{"pop": 97, "electronic": 36}	{80s,"new wave",pop,"synth pop",s...	"Tainted Love", composed by Ed Cobb and fi...
6	e3b7cd...	17d6597...	jimmy eat world	the middle	166000	1741683	15473288	bleed american	https://ww...	{"pop": 77, "punk": 9, "rock": 106, "indie": 68, "indie rock": ...	{rock,"alternative rock",alternative,...	"The Middle" is a song by American rock ba...
7	609dce...	c9d1ad1...	florence + the m...	you've got the love	168000	1356600	11903404	lungs	https://ww...	{"pop": 211, "indie": 157, "alternative": 241, "female vocali...	{indie,"female vocalists",alternativ...	Featured as the b-side to Florence + The M...
8	22d2b4...	fb42c7d...	spice girls	wannabe	172000	1309928	7463983	spice	https://ww...	{"pop": 233, "female vocalists": 107}	{pop,90s,dance,"Spice Girls",'fema...	"Wannabe" is a song by the British pop grou...
9	778c57f...	737f93d...	gnarls barkley	crazy	177000	1814485	13494344	st. elsewhere	https://ww...	{"pop": 66, "Hip-Hop": 19, "alternative": 92}	{soul,funk,Hip-Hop,crazy,pop}	"Crazy" is the first single from Gnarls Barkle...
10	fc7dc7a...	ed42077...	the jackson 5	i want you back	178000	1479332	8973311	motown 50	https://ww...	{"pop": 146}	{motown,soul,70s,pop,funk}	"I Want You Back" is a 1969 number-one sin...
11	9ed869...	9daedc9...	katy perry	i kissed a girl	178000	2264846	16619562	one of the boys	https://ww...	{"pop": 19, "rock": 29, "female vocalists": 6}	{pop,"female vocalists",'katy perry...	"I Kissed a Girl" is a two-time VMA's nominat...
12	445a49...	ebc38ad...	van morrison	brown eyed girl	181000	1362662	8458883	the best of van mo...	https://ww...	{"pop": 206, "folk": 7, "classic rock": 44}	{"classic rock",rock,60s,oldies,sin...	The song was first released in May 1967 on ...
13	2313cb...	31906ac...	timbaland	the way i are	181000	1331917	10845826	nrj hits 10	https://ww...	{"pop": 222, "Hip-Hop": 76}	{Hip-Hop,Timbaland,rnb,dance,pop}	"The Way I Are" is a song by Timbaland, rele...
14	50d83ff...	3869bc8...	elton john	i'm still standing	182000	1363232	11027383	too low for zero	https://ww...	{"pop": 205, "classic rock": 43}	{pop,80s,'elton john',rock,'classic ...	"I'm Still Standing" is a song written by Engli...
15	2d3e84...	04a3d79...	amy winehouse	tears dry on their own	186000	1400545	11924763	back to black	https://ww...	{"pop": 187, "jazz": 4, "female vocalists": 81}	{soul,jazz,"female vocalists",'Amy ...	"Tears Dry on Their Own" is a melancholic re...
16	6f1b86f...	ce81bc6...	lorde	royals	189000	1787402	18051233	pure heroine	https://ww...	{"pop": 71, "indie": 60, "electronic": 24, "alternative": 94, "f...	{pop,indie,Lorde,electronic,2013}	"Royals" is a song by Lorde, released on Nov...
17	1675df7...	eacdd72...	kelly clarkson	since u been gone	189000	1430226	9917986	greatest hits - cha...	https://ww...	{"pop": 171, "rock": 219, "female vocalists": 72}	{pop,'Kelly Clarkson',"female vocal...	"Since U Been Gone" is a song written and p...
18	8c2ead...	3922d35...	the police	roxanne	191000	1534377	9990153	outlandos d'amour	https://ww...	{"pop": 124, "rock": 169, "classic rock": 22}	{rock,"classic rock",80s,"new wave...	"Roxanne" is a hit song by the rock band The...
19	c2beea...	9d2f94e...	capital cities	safe and sound	192000	1452777	13285871	in a tidal wave of ...	https://ww...	{"pop": 156, "indie": 123, "electronic": 52, "alternative": 192}	{indie,electronic,pop,alternative,"in...	"Safe and Sound" is a song by American indi...
20	e760b3...	62248ca...	carly rae jepsen	call me maybe	193000	1899958	16364196	kiss	https://ww...	{"pop": 53, "female vocalists": 22}	{pop,dance,"female vocalists",Can...	"Call Me Maybe" is a song by Canadian Juno...

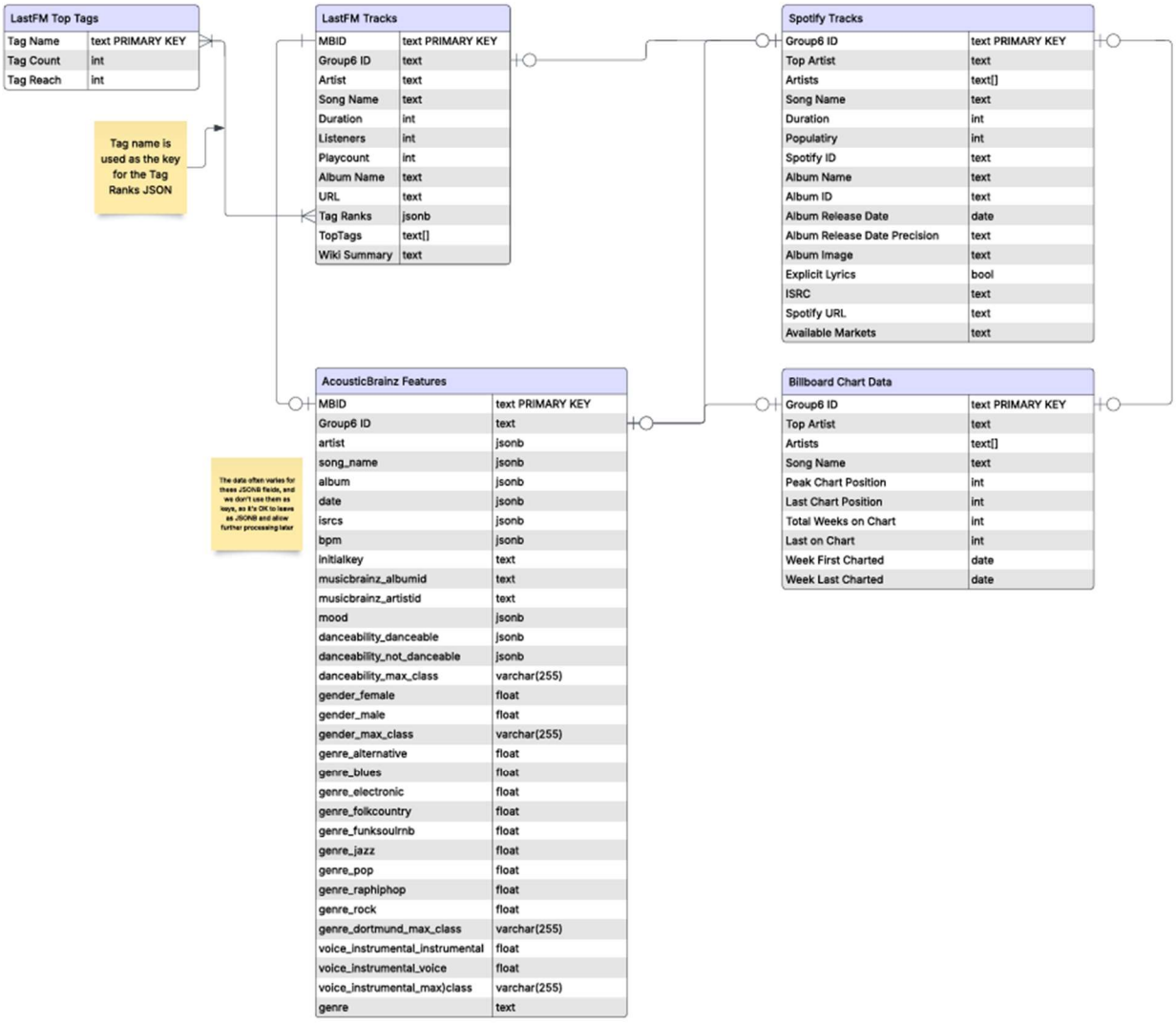
4. AcousticBrainz Track Info – Table “acousticbrainz_features”
    a. The MBID (MusicBrainz ID) acts as the Primary Key for this data.
    b. No de-duplication was necessary for AcousticBrainz since we queried the data using the distinct MBIDs from the LastFM dataset, so by default there was only one record per song.
    c. Not all attributes are kept. Much of the data is experimental in nature and many audio features are available. As an example, for genre data, we only kept the “dortmund” set.
    d. A unique key is given to the track by hashing the Song and Artist name, which is the joining key across all song tables



Entity – Relationship Diagram

This ERD is also shared at the end of the PDF. The keys are fairly self-explanatory but a few important note are as follows:

- The Group6 ID is created during processing as a universal ID, based on matches of the combination of the name of the song and the name of the song's top artist. Matches are carried across 4 of the 5 tables.
- The MusicBrainz ID (MBID) is used as a way to relate the LastFM Tracks and the AcousticBrainz Features, since that is retrieved from the API requests in each case.



See this attached at the end of the writeup PDF for a closer view.

Flask API

To make the processed music data accessible and easy to explore, the project includes a Flask API. This component allows users to interact with the database and retrieve views of the integrated dataset through a web browser or API calls.

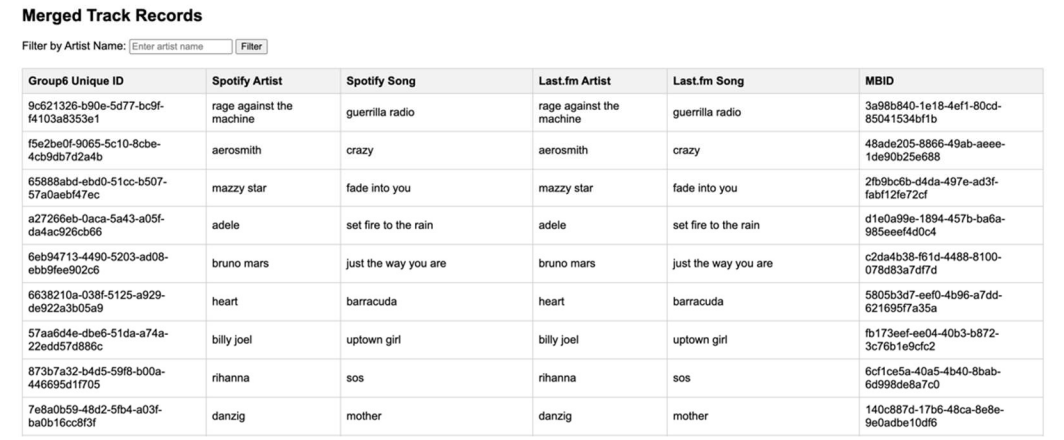
The Flask API is containerized and runs as part of the Docker pipeline. It connects to the group6 PostgreSQL database and exposes HTTP endpoints that allow users to query the data using predefined SQL statements. Each route is designed to return relevant results in a readable HTML format.

Architecture:

- API code is located in app.py
- Flask runs on port 5001 within its own Docker container
- Queries are executed using psycopg2 to connect to the PostgreSQL database

Available Endpoints:

1. merged-tracks
   a. HTML view: Displays the final merged dataset from Spotify, Last.fm and Kaggle where data from all three sources match based on the shared group6_id. Includes a filter box to narrow results by artist name.

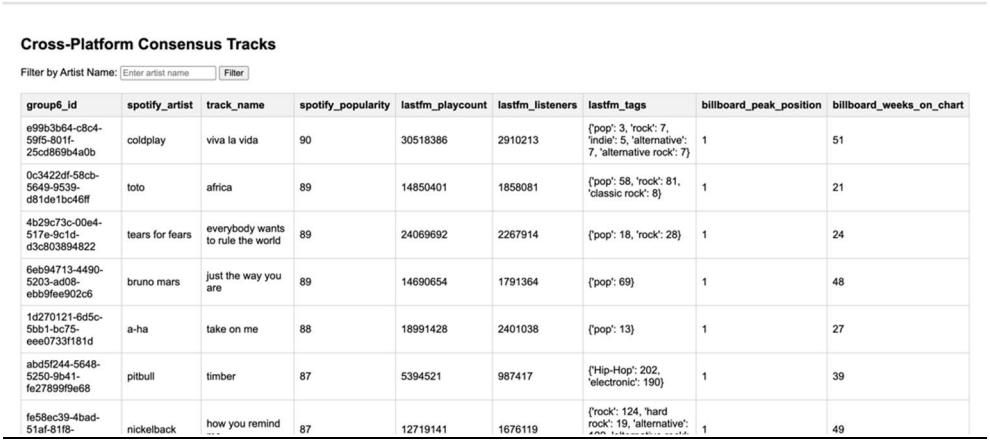


Merged Track Records

Filter by Artist Name: Enter artist name Filter

Group6 Unique ID	Spotify Artist	Spotify Song	Last.fm Artist	Last.fm Song	MBID
9c621326-b90e-5d77-bc9f-f4103a8353e1	rage against the machine	guerrilla radio	rage against the machine	guerrilla radio	3a98b840-1e18-4ef1-80cd-85041534bf1b
f5e2be0f-9065-5c10-8cbe-4cb9db7d2a4b	aerosmith	crazy	aerosmith	crazy	48ade205-8866-49ab-aeee-1de90b25e688
65888abd-ebd0-51cc-b507-57a0aebf47ec	mazzy star	fade into you	mazzy star	fade into you	2fb9bc6b-d4da-497e-ad3f-fabf12fe72cf
a27266eb-0aca-5a43-a05f-da4ac926cb66	adele	set fire to the rain	adele	set fire to the rain	d1e0a99e-1894-457b-ba6a-985eeef4d0c4
6eb94713-4490-5203-ad08-ebb9fee902c6	bruno mars	just the way you are	bruno mars	just the way you are	c2da4b38-f61d-4488-8100-078d83a7df7d
6638210a-038f-5125-a929-de922a3b05a9	heart	barracuda	heart	barracuda	5805b3d7-eef0-4b96-a7dd-621695f7a35a
57aa6d4e-dbe6-51da-a74a-22edd57d886c	billy joel	uptown girl	billy joel	uptown girl	fb173eef-ee04-40b3-b872-3c76b1e9cfc2
873b7a32-b4d5-59f8-b00a-446695d1f705	rihanna	sos	rihanna	sos	6cf1ce5a-40a5-4b40-8bab-6d998de8a7c0
7e8a0b59-48d2-5fb4-a03f-ba0b16cc8f3f	danzig	mother	danzig	mother	140c887d-17b6-48ca-8e8e-9e0adbe10df6

   b. JSON view: Programmatic version of the above data. Returns the same matched dataset, but in raw JSON format. You will see a JSON array of objects.
2. Cross-platform-consensus
   a. HTML view: Shows tracks that have achieved cross-platform success with Spotify popularity > 70, Last.fm playcount> 100,000, and charted on Billboard for more than 5 weeks. Includes a filter box to narrow results by artist name.

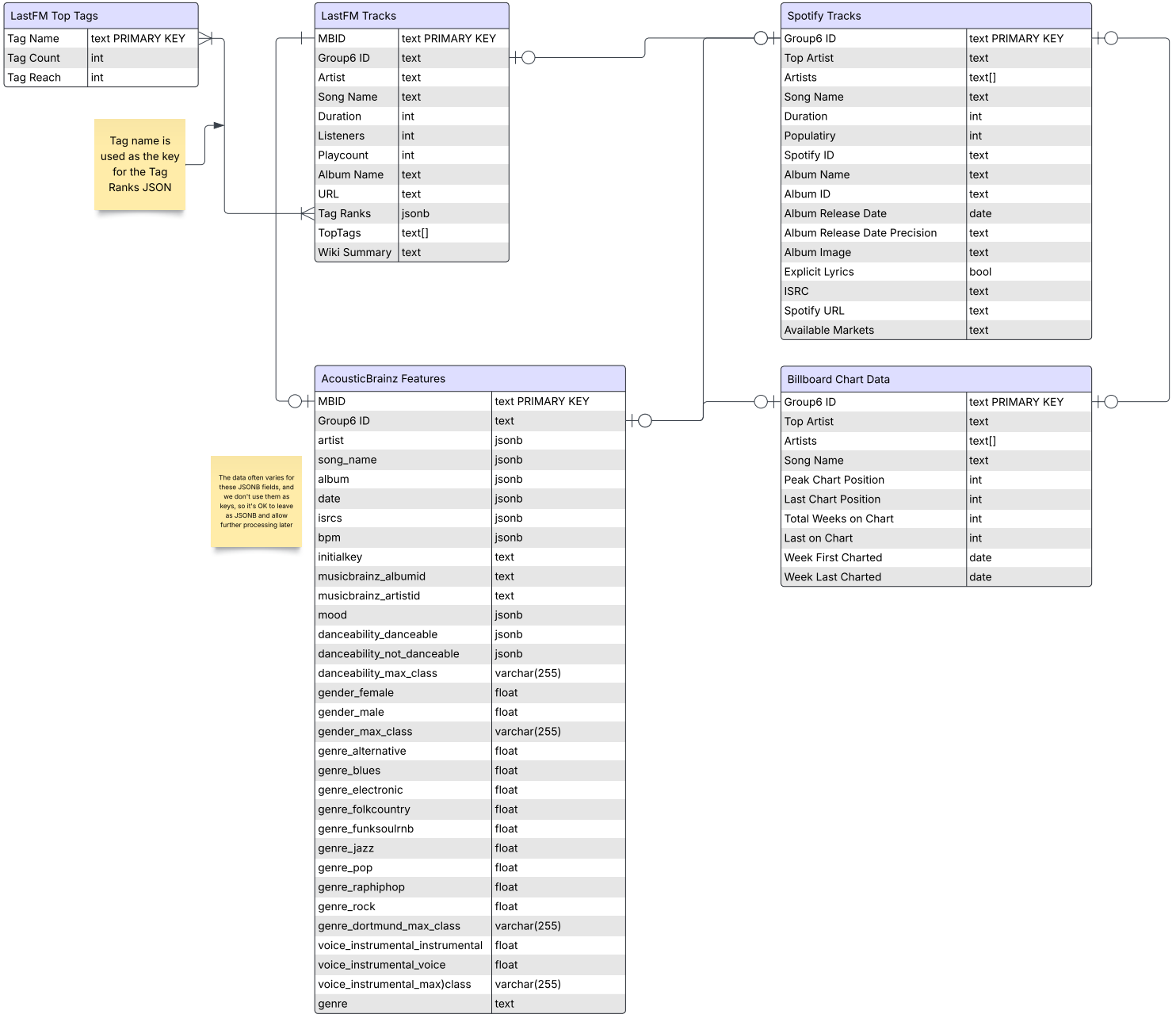


Cross-Platform Consensus Tracks

Filter by Artist Name: Enter artist name Filter

group6_id	spotify_artist	track_name	spotify_popularity	lastfm_playcount	lastfm_listeners	lastfm_tags	billboard_peak_position	billboard_weeks_on_chart
e99b3b64-c8c4-59f5-801f-25cd869b4a0b	coldplay	viva la vida	90	30518386	2910213	{'pop': 3, 'rock': 7, 'indie': 5, 'alternative': 7, 'alternative rock': 7}	1	51
0c3422df-58cb-5649-9539-d81de1bc46ff	toto	africa	89	14850401	1858081	{'pop': 58, 'rock': 81, 'classic rock': 8}	1	21
4b29c73c-00e4-517e-9c1d-d3c803894822	tears for fears	everybody wants to rule the world	89	24069692	2267914	{'pop': 18, 'rock': 28}	1	24
6eb94713-4490-5203-ad08-ebb9fee902c6	bruno mars	just the way you are	89	14690654	1791364	{'pop': 69}	1	48
1d270121-6d5c-5bb1-bc75-eee0733f181d	a-ha	take on me	88	18991428	2401038	{'pop': 13}	1	27
abd5f244-5648-5250-9b41-fe27899f9e68	pitbull	timber	87	5394521	987417	{'Hip-Hop': 202, 'electronic': 190}	1	39
fe58ec39-4bad-51af-81f8-	nickelback	how you remind	87	12719141	1676119	{'rock': 124, 'hard rock': 19, 'alternative':	1	49

b. JSON view: Programmatic version of the above data. Returns the same matched dataset, but in raw JSON format. You will see a JSON array of objects.


LastFM Top Tags
Tag Name text PRIMARY KEY
Tag Count int
Tag Reach int
Tag name is used as the key for the Tag Ranks JSON
LastFM Tracks
MBID text PRIMARY KEY
Group6 ID text
Artist text
Song Name text
Duration int
Listeners int
Playcount int
Album Name text
URL text
Tag Ranks jsonb
TopTags text[]
Wiki Summary text
Spotify Tracks
Group6 ID text PRIMARY KEY
Top Artist text
Artists text[]
Song Name text
Duration int
Populatiry int
Spotify ID text
Album Name text
Album ID text
Album Release Date date
Album Release Date Precision text
Album Image text
Explicit Lyrics bool
ISRC text
Spotify URL text
Available Markets text
AcousticBrainz Features
MBID text PRIMARY KEY
Group6 ID text
artist jsonb
song_name jsonb
album jsonb
date jsonb
isrcs jsonb
bpm jsonb
initialkey text
musicbrainz_albumid text
musicbrainz_artistid text
mood jsonb
danceability_danceable jsonb
danceability_not_danceable jsonb
danceability_max_class varchar(255)
gender_female float
gender_male float
gender_max_class varchar(255)
genre_alternative float
genre_blues float
genre_electronic float
genre_folkcountry float
genre_funksoulrnb float
genre_jazz float
genre_pop float
genre_raphiphop float
genre_rock float
genre_dortmund_max_class varchar(255)
voice_instrumental_instrumental float
voice_instrumental_voice float
voice_instrumental_max)class varchar(255)
genre text
The data often varies for these JSONB fields, and we don't use them as keys, so it's OK to leave as JSONB and allow further processing later
Billboard Chart Data
Group6 ID text PRIMARY KEY
Top Artist text
Artists text[]
Song Name text
Peak Chart Position int
Last Chart Position int
Total Weeks on Chart int
Last on Chart int
Week First Charted date
Week Last Charted date