Music Artist Popularity Tracking Across Platforms

Data Acquisition Project

Rebecca El Chidiac Adrien Guittard Oscar Garnier

December 19, 2025

Outline

Motivation

During this project, we wanted to focus on several topics covered in the course that we found particularly interesting and build one whole coherent system around it.
In particular, we wanted to work with:

- **Interacting with external APIs**
- **Web scraping** and data extraction
- **Relational data modeling and storage**
- **Data provenance**, at both the workflow and fine-grained levels
- We also aimed to include an **aggregation step inspired by the MapReduce paradigm** introduced during the course.

Project Topic

- We chose to build a dataset that tracks the **popularity of artists over time**.
- The objective is to produce a **time-series dataset**, rather than a static snapshot.
- Popularity signals are **distributed across multiple platforms**, each providing different information:
  - **Spotify** (music-related popularity indicators)
  - **YouTube** (channel-level activity and audience)
  - **Wikipedia** (public interest through page views)
- The final goal is a **clean, structured dataset** that integrates these signals on a daily basis.

Pipeline Overview

The system is organized around two complementary pipelines with different roles.

Initialization pipeline

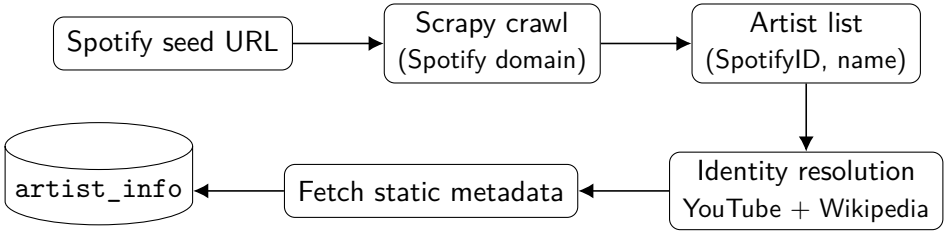
- **Static**, one-shot process
- Run once at initialization
- Builds the list of artists to track
- Resolves artist identities across platforms
- Collects stable artist metadata

Daily pipeline

- **Dynamic**, runs on a daily basis
- Tracks artist popularity over time
- Collects indicators from multiple data sources
- Updates the dataset incrementally

Outline

Initialization pipeline (static, one-shot)

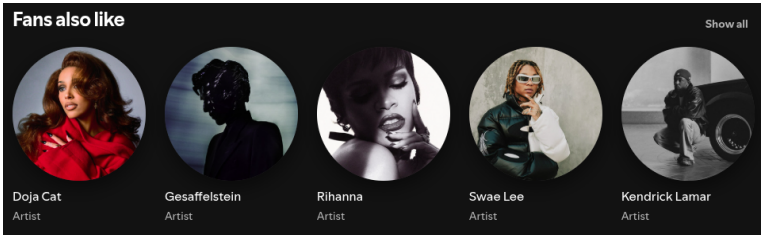


One-shot pipeline producing a clean reference table used by the daily pipeline.

Scrapping

Goal: Get a list of artists automatically.

- We start from a root page (default is The Weeknd), and follow spotify's "Fans also like".
- We iterate until we reach **max_artists** parameter (we ran for 1000 artists 20min)
- We include an option for filtering artists included with **min_artist_listeners** and **max_artist_listeners**



Scraper

- We build a **scrapy** spider
- Using Xpath to get info on the raw htlm
- We extract **artist name**, **monthly listeners**, and **Fans also like**

Idendity Resolution

Goal: Gather artist metadata that won´t be updated regularly (links, profile picture).
We map the spotify_id's from the scrapper to youtube and wikipedia id's thought calls to their API's search functionality.

local_artist_id	artist_name	spotify_artist_id	wiki_title	youtube_channel_id	genres_json	image_url
The Weeknd	The Weeknd	1Xyo4u8uXC1ZmMpatF05PJ	The Weeknd	UC0WP5P-ufpRfjbNrmOWwLBQ	[]	https://i.scdn.co/image/ab6761610000e5eb
Bruno Mars	Bruno Mars	0du5cEVh5yTK9QJze8zA8C	Bruno Mars	UCoUM-UJ7rirJYP8CQ8EIaHA	[]	https://i.scdn.co/image/ab6761610000e5eb
Lily-Rose Depp	Lily-Rose Depp	1pBLC0qVRTB5zVMuteQ9jJ	Lily-Rose Depp	UC-LN1pXo40K9j0iyFsv_GA	[]	https://i.scdn.co/image/ab6761610000e5eb
SZA	SZA	7tYKF4w9nC0nq9CsPZTHyP	SZA	UC0SIQ70V7l-XpHW40HwaGsw	["r&b"]	https://i.scdn.co/image/ab6761610000e5eb
Kendrick Lamar	Kendrick Lamar	2YZyLoL8N0Wb9xBtiNhZWg	Kendrick Lamar	UC3lBXcrKFnFAFkfVk5WuKcQ	["hip hop", "west coast hip hop"]	https://i.scdn.co/image/ab6761610000e5eb
Swae Lee	Swae Lee	1zNqQNIdeOUZHb8zbZRFMX	Swae Lee	UCUJTvb2boc_15y3DMNvZlEA	[]	https://i.scdn.co/image/ab6761610000e5eb
Rihanna	Rihanna	5pKCCKE2ajJHZ9KAiaK11H	Rihanna	UCcgqSM4YEo5vVQpgwN-MaNw	[]	https://i.scdn.co/image/ab6761610000e5eb
Gesaffelstein	Gesaffelstein	3hteYQfiMFbJY7w50xDymP	Gesaffelstein	UCneAnaIzzbA9MtTijP4UkpA	[]	https://i.scdn.co/image/ab6761610000e5eb
Doja Cat	Doja Cat	5cjOlLjcoR7Y0SnhnX0Po5	Doja Cat	UCzpl23pGTHVYqvKsgYBA-_w	[]	https://i.scdn.co/image/ab6761610000e5eb
Charlie Puth	Charlie Puth	6VuMaDnrHyPL1p4EHjYLi7	Charlie Puth	UCwppdrjsBPAZg5_cUwOjfHQ	["soft pop", "pop"]	https://i.scdn.co/image/ab6761610000e5eb
Shawn Mendes	Shawn Mendes	7n2WHs1TKAczGzO7Dd2rGr	Shawn Mendes	UCAvCL8hyXjSUHKEGUUPr1BA	[]	https://i.scdn.co/image/ab6761610000e5eb
Katy Perry	Katy Perry	6jJ0s89eD6GaHleKKya26X	Katy Perry	UCYvnuw-JtVrTZQ-7Y4kd63Q	["pop"]	https://i.scdn.co/image/ab6761610000e5eb
Sam Smith	Sam Smith	2wY79sveU1sp5g7SokKOiI	Sam Smith	UCvpDeGlR5wLP9Z3Tb6K8Xfg	["soft pop"]	https://i.scdn.co/image/ab6761610000e5eb
Coldplay	Coldplay	4gzpq5DPGxSnKTe4SA8HAU	Coldplay	UCDPM_n1atn2ijUwHd0NNROw	[]	https://i.scdn.co/image/ab6761610000e5eb
Adele	Adele	4dpARuHxo51G3z768sgnrY	Adele	UCsRM0YB_dabtEPGPTKo-gcw	["soft pop"]	https://i.scdn.co/image/ab6761610000e5eb
Maroon 5	Maroon 5	04gDigrS5kc9YWfZHwBETP	Maroon 5	UCBVjMGOIkavEAhyqpxJ73Dw	["pop"]	https://i.scdn.co/image/ab6761610000e5eb

artist_info — Reference Table

This table is the central reference produced by the initialization pipeline. This table is the entry point of the daily pipeline and need to be correctly traced

Identifiers

- local_artist_id (primary key)
- spotify_artist_id, youtube_channel_id, wiki_title

External resources

- spotify_url, youtube_channel_url, wikipedia_url
- image_url

Artist metadata

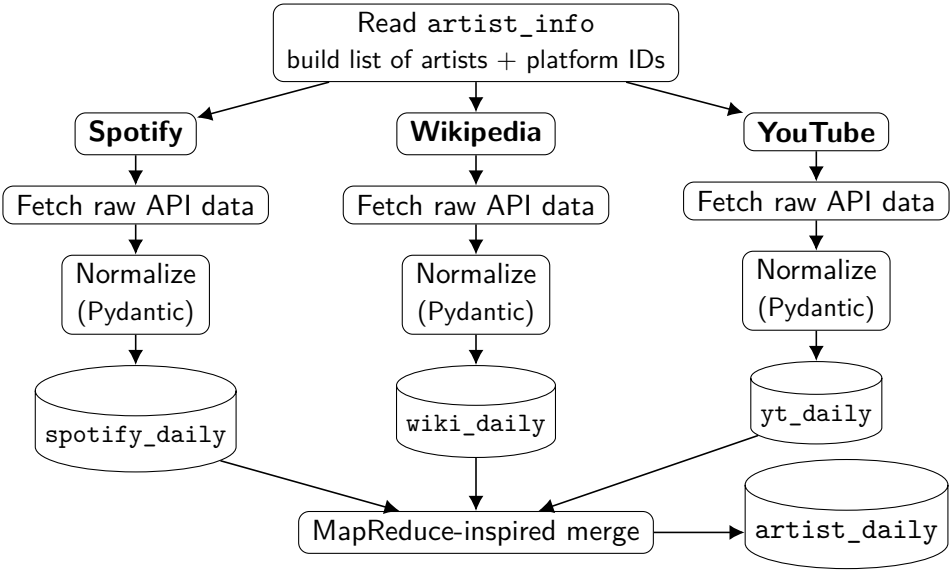
- artist_name
- country, debut_year
- genres_json

Traceability

- fetched_at
- job_run_id

Outline

Daily pipeline (dynamic, runs every day)



Normalization

Why normalization matters:

- APIs return heterogeneous JSON structures
- We need consistent naming, types, and missing-value handling
- Validation prevents silently corrupted data

Examples of normalized fields:

- `day_date` (YYYY-MM-DD)
- `fetched_at` (timestamp, UTC)
- `followers_total`, `popularity`
- `youtube_subs`, `wiki_pageviews`

MapReduce-style Merge

Why merge?

- Analysts want a single daily row per artist with all signals
- Enables correlations: Spotify popularity vs Wiki spikes vs YouTube growth

Approach

- **Map**: emit key-value pairs (k, v) from each source table, where $k = (\texttt{local\_artist\_id}, \texttt{day\_date})$ and v contains metrics + `fetched_at`
- **Shuffle**: group all values by key k
- **Reduce**: for each key, select the **latest per source** using `fetched_at`
- **Output**: upsert the merged result into `artist_day`

Provenance & Traceability

We record provenance so that each stored data point can be traced back to **how it was produced** and **which request(s)** generated it.

Workflow provenance (run & steps)

- `run`: one entry per daily execution
  - commit hash, start/end, status, error
- `run_step`: one entry per step inside a run
  - step name, counts (success/error), inputs/outputs

Fine-grained provenance (per request)

- `api_request`: one entry per API call
  - source, endpoint, params
  - timing, HTTP status, ok/error
  - artist + platform id linkage

Result: end-to-end traceability

For any daily record, we can identify the **run** $\rightarrow$ **step** $\rightarrow$ **API request(s)** that produced it, and diagnose failures (rate limits, invalid IDs, timeouts).

Demo / screenshot of the DB

spotify_followers_total	spotify_popularity	spotify_top_track_popularity_mean	wiki_pageviews
115575232	96	88.6	6111
76965691	92	84.7	4481
647869	76	58.1111111111111	3425
33864433	90	83.5	2431
45472722	91	85.0	5479
3140927	79	69.5	502
70190164	93	83.2	7857
1342438	72	63.3	312
35354244	88	80.2	3626
25582444	82	77.5	2017
45964242	83	78.9	3217
38896782	86	76.9	7161
26544975	83	79.5	1470
61663947	90	87.6	2783
67693187	85	80.5	4047
46252926	88	83.6	1445

Outline

Limitations & Next Steps

Current limitations

- Some metadata missing (e.g., genres sometimes empty)

Next steps

- Add more sources (Deezer, Instagram/TikTok signals, ticketing...)
- Monitoring: automated daily job + alerting on collector failures

Conclusions and Takeaways

This project taught us

- To structure an SQLite database
- To access information via an API
- To scrape a website
- To run a Map-Reduce operation
- To setup a clean data filtering step when collecting it

Thank you