*

A visual exploration into listening data

www.github.com/jurdabos/canonfodder

For the course DLBDSEDAV01 – tutor: Lac Visieu

STUDENT: TORDA BALÁZS, 92128244 (Data Science Bsc.)

Finalized for submission on 31 May 2025 in València

Table of Contents

List of figures ii

Abbreviations and definitions iii

1. Introduction 1

2. Visual exploration of scrobble data 2

2.1 Preparatory tasks 2

2.2 Colour palette choice 2

2.3 EDA 3

2.4 Country-stat visuals 7

2.5 Comparison with literature 9

3. Conclusion and discussion 9

3.1 Methods recap 9

3.2 Key results 10

3.3 Discussion 10

Bibliography 11

APPENDIX: Extra information on CanonFodder 13

List of figures

FIGURE 1: TOP ARTISTS BEFORE AND AFTER RECORD LINKAGE OF BAND NAME VARIANTS FOR BOHREN &/UND DER CLUB OF GORE 1
FIGURE 2: CONTRAST VALUES OF CHOSEN MAIN COLOURS PLUS WHITE AND BLACK USING A TOOL AT HTTPS://CONTRASTGRID.COM/ 3
FIGURE 3: TOTAL NUMBER OF SCROBBLES PER EACH YEAR IN THE DATA SET 4
FIGURE 4: TIME SERIES DECOMPOSITION – SCROBBLES OVER THE YEARS, TREND, SEASONALITY AND NOISE (FROM TOP TO BOTTOM) 4
FIGURE 5: RIDGELINE PLOT OF MONTHLY DISTRIBUTION OF SEASONALITY 5
FIGURE 6: MOST ARTISTS ARE ONLY FEATURED IN THE SCROBBLE DATABASE WITH A FEW LISTENS 6
FIGURE 7: ARTISTS WITH MOST CONSISTENT LISTENING PATTERNS (2006-2025) 6
FIGURE 8: PUTTING SCROBBLES ON THE WORLD MAP 7
FIGURE 9: SCROBBLE COUNT PER COUNTRY POPULATION ON A LOG SCALE 8
FIGURE 10: BAR CHART OF SCROBBLES PER COUNTRY WHERE THE USER WAS LOCATED AT THE TIME OF SCROBBLING 8

Abbreviations and definitions

Abbrev/Term	Definition/Longform
al.	aliī = others
API	application programming interface
AVC	artist_variants_canonized (table in the DB)
BI	business intelligence (frontend)
CF	CanonFodder (project)
cf.	cōnfer = compare
CLI	command-line interface
DAG	Directed Acyclic Graph
db/DB	database
DBSCAN	density-based spatial clustering of applications with noise
DWH	data warehouse (schema in RDBMS)
e. g.	exemplī grātiā = for example
EDA	exploratory data analysis
etc.	et cetera
ETL	Extract Transform Load
Eval DB	Parquet star schema export for lightweight analytics
fm	frequency modulation
HTTP	Hypertext Transfer Protocol
i. e.	id est = that is
JSON	JavaScript Object Notation
MBID	MusicBrainz identification number
ML	machine learning
p./pp.	page/pages
REST	representational state transfer
RL	record linkage
scrobble	one track play event (artist, track, timestamp)
SRD	software requirement documentation
SQL	structured query language
URL	uniform resource locator
UTS	Unix Time Stamp provided by last.fm
Y/M/D	year/month/day

1. Introduction

CanonFodder is a reproducible data engineering pipeline that ingests music listening events (scrobbles), enriches them with metadata, stores them in a DB, and offers analytics and visualisation. For presentational purposes related to EDA in the current iteration of the project, we use last.fm scrobble data for a user created in 2006, which is publicly available at https://www.last.fm/user/jurda.

Scrobble data is a tabular list of records, with each row representing an event of the user having listened to a song. The top-priority group of such data services are musicologists and self-tracking enthusiasts. Researchers apply the terms *lifelogging*, *quantified past/*self and *personal informatics* to describe the phenomenon. Studies (e. g. Elsden et al., 2016) found enthusiasm among people tracking their listening habits with many getting actively involved in the data collection process.

As described in detail in Torda (2025, p. 1), household scrobble service providers such as last.fm and ListenBrainz have not managed to apply sufficient data cleaning to deal with quality issues. For our user, the top artist is incorrectly identified by these services. This has been the rationale for launching CanonFodder, with a mission of providing tools for canonization of artist name variants.

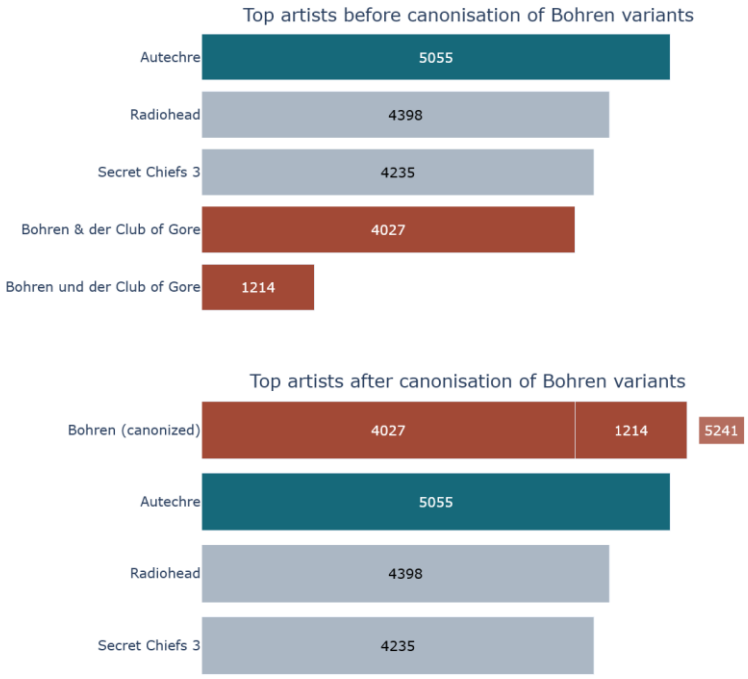


Figure 1: Top artists before and after record linkage of band name variants for Bohren &/und der Club of Gore

The aim of the current research is to explore data analytics and visualization techniques suitable for gathering insights into scrobbling data. In the main chapter, *(2.1)* the most important preparatory tasks will be delineated to set up the pipeline, then *(2.2)* choices on colour palettes will be explored. This will be followed by *(2.3)* exploring insights on scrobble data using EDA, then we *(2.4)* move on to end-user visuals, with putting scrobbles on the map as well. Out of scope for the current essay is

information on musical styles and genres, which is an interesting topic on its own right, but it is deferred to a later stage of CF to keep the focus on a limited set of attributes for now.

2. Visual exploration of scrobble data

The research question is as follows: What type of analysis techniques and visualizations are appropriate for the exploration of music listening data?

2.1 Preparatory tasks

Data is gathered through the API last.fm offers. The process can be validated via the public code base in lfAPI.py, which sends HTTP GET requests to the provider's REST endpoint. SQLAlchemy and Alembic are utilized as part of a robust DWH-centred approach with backend-agnostic build-up of models and operations, while a Parquet file store guarantees quick load for lake-style analytics. The goal is to incorporate orchestration for weekly autofetch of new data, plus merge with previous information with robust policies in place for conflict handling. In addition to gathering scrobble data, logic is being designed for fetching artist information from MusicBrainz as well as incorporating user-provided data to be consolidated in a harmonious database. For the automated side of artist name variant canonization, we have trained an XGBoost model that is served as an internal predictor via a Flask app. For a detailed description of the project setup, see Appendix A.

CF is a pull-based pipeline triggered manually or via Airflow. It converts raw scrobbles into a star schema, normalising artist aliases and enriching artist entities with country metadata. Users can launch canonicalisation of artist name variants and visual exploration from a CLI menu.

In scope for the CF project is automated data ingestion from last.fm, artist info enrichment through MusicBrainz, manual/automatic canonisation, storage in MySQL/SQLite/PostgreSQL, parquet export, BI dashboard, workflow orchestration with Airflow, and Docker packaging. Out of scope is streaming playback, real time recommendation engine, paid cloud hosting.

The EDA and preliminary visualizations can be followed in the notebook-style profile.py file.

2.2 Colour palette choice

To achieve a consistent look, a font (Lucida Sans Unicode) and main colours have been chosen with a tool at https://coolors.co. For easy reusability, a .JSON colour object have been created containing a set of palettes with n number of distinct hues ($1 < n < 11$). As discrete sequential palettes, shades of Caribbean Current have been set up by setting how much white is mixed in to the main hue (both in the direction from dark to light as well as from light to dark). When necessary, steps between shades have been adjusted manually to incorporate only those neighbouring tints that are easier to tell apart for the human eye.

Contrast checking has also been performed for thoroughness.

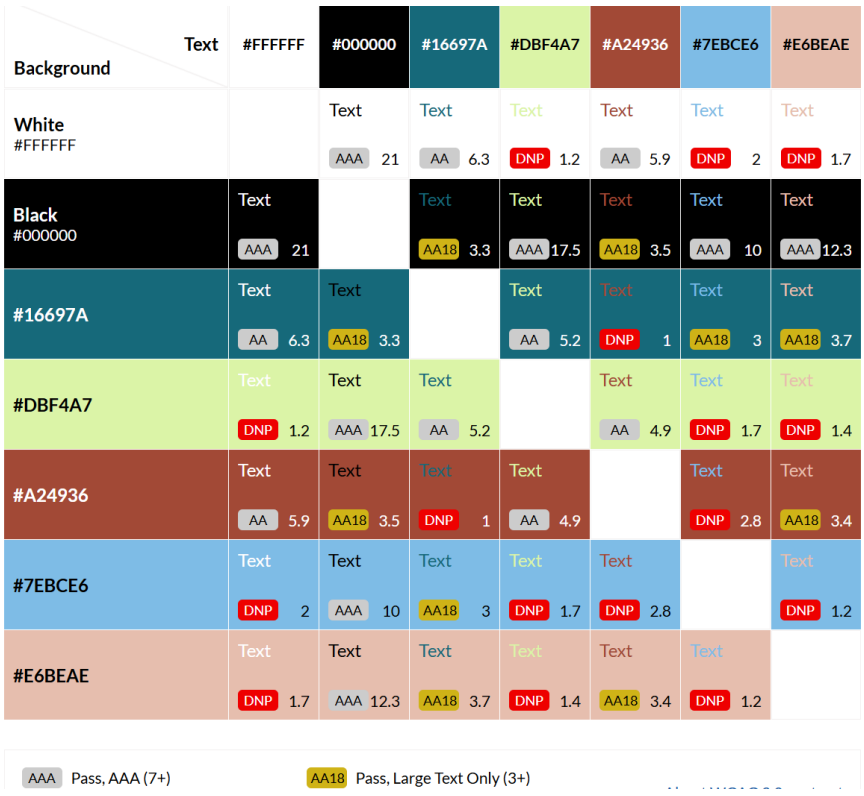


Figure 2: Contrast values of chosen main colours plus white and black using a tool at https://contrastgrid.com/

2.3 EDA

Exploratory data analysis refers to the process of probing into a data set with the intention to gain insight about its general shape, tendencies and attributes. The methods applied can include statistical calculations, data visualizations to identify outliers, and checks for the distribution of data points (Kaur et al., 2025, Section D). **EDA** overlaps with a related term **data profiling**, with some researches considering the two process orthogonal to each other (Abedjan et al., 2017).

In this section, we are going to illustrate the central tendency and the spread of attributes in the data set.

2.3.1 Scrobbles through the years/months

To get an initial bearing of the distribution of data points through time, we plot the number of scrobbles per year. As there is only one variable here to be visualized through time, a simple, clean bar chart is sufficient for this purpose. In the spirit of balancing cognitive load with complete information, the axis label (year, scrobbles) are dropped, since the plot title already contains the necessary reference point.

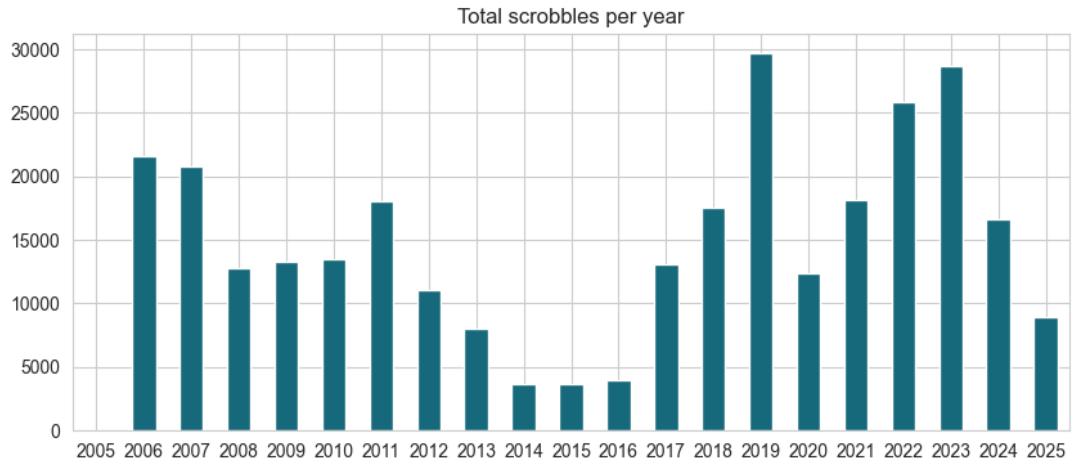


Figure 3: Total number of scrobbles per each year in the data set

Date feature construction from scrobble time stamps is performed in step 4 of the profile.py demonstration workflow. The raw data contains the uts value of historical scrobbles. Extra columns are created to include separate attributes for year, month and day. At this point, another plot is built identifying trends, seasonality and random noise contained in the observed data.

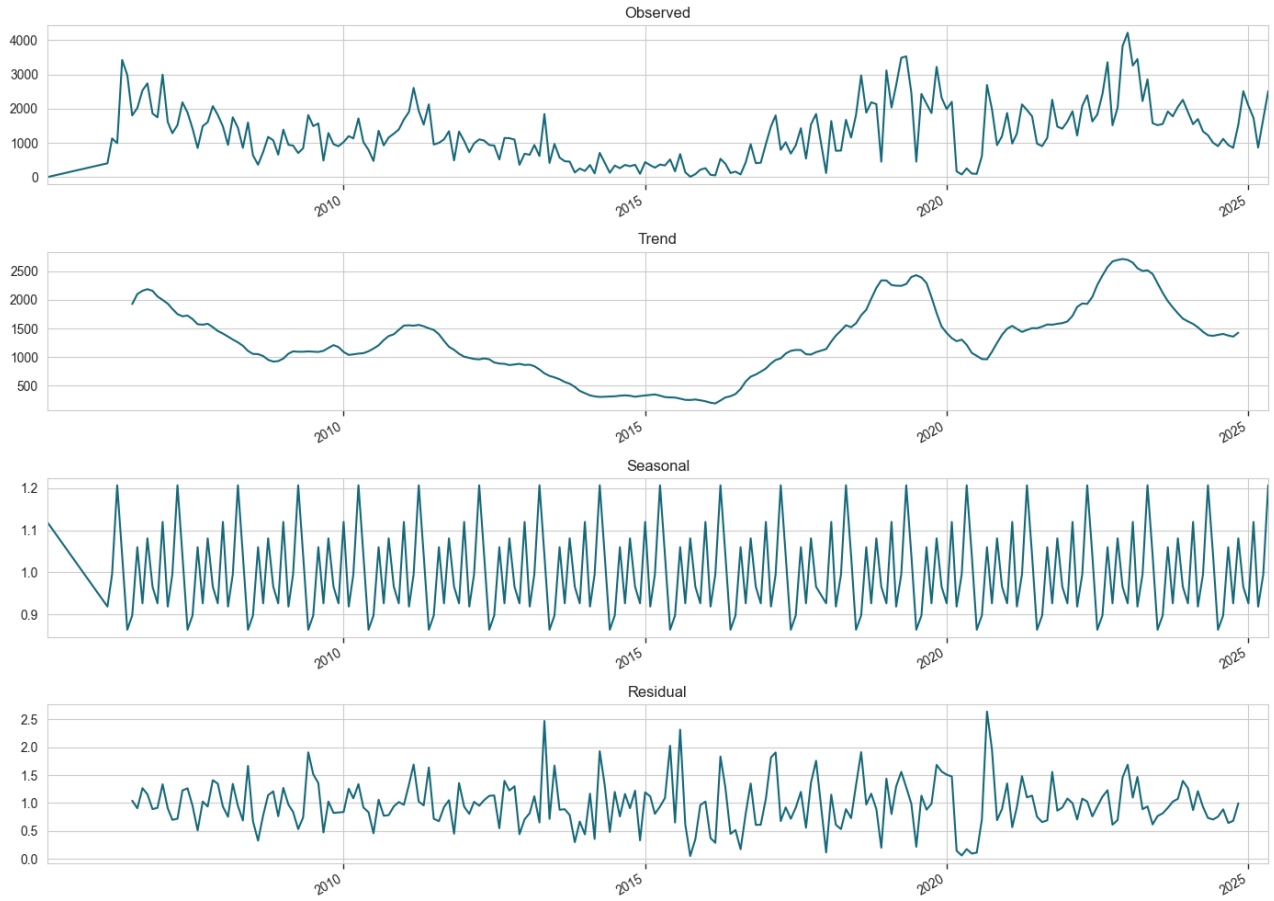


Figure 4: Time series decomposition – scrobbles over the years, trend, seasonality and noise (from top to bottom)

A very stable annual seasonality (≈ ±10-12 % on a multiplicative scale) persists throughout the entire window, signalling predictable listening highs and lows each calendar year. Except for a brief

negative spike in mid-2020 (likely a data gap or listening hiatus) and a few positive bursts during the 2021 surge, residual variance stays modest, so the trend-plus-seasonal model explains almost all systematic structure in the user's scrobble history.

Going one step further into the exploration of seasonality, monthly distribution of scrobble counts is visualized with a ridgeline plot (joyplot) that excels in demonstrating the probability density of variables through time. (cf. e. g. Abdul-Rahman et al., 2024, p. 5)

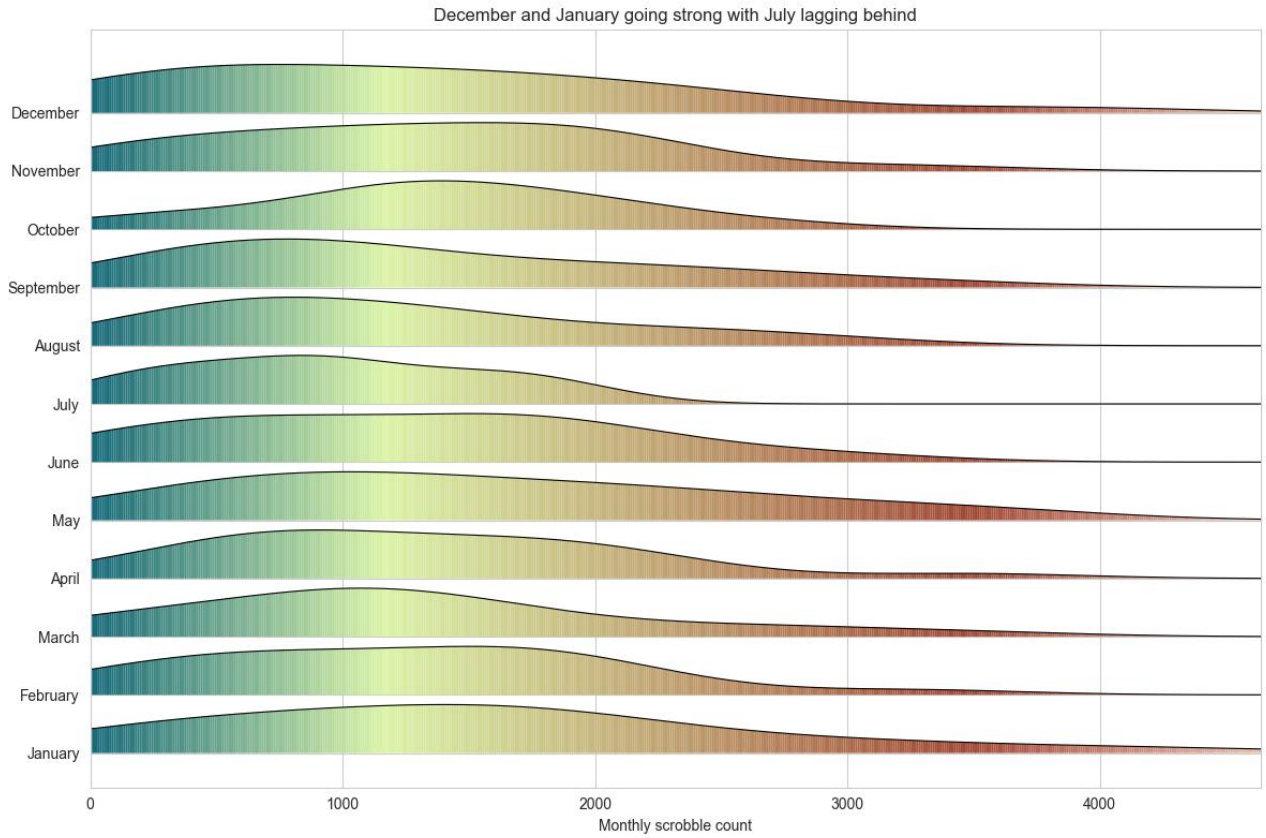


Figure 5: Ridgeline plot of monthly distribution of seasonality

The plot clearly indicates a downward trend for scrobbles during July, and a stronger overall listening activity for December and January.

2.3.2 Overall distribution of artist counts

The bird's eye stats for the number of tracks each artist is featured with in the data set might also prove to be interesting. Calling value_counts() on the relevant column in the dataframe, we see a total artist count of approximately 20000, with a mean scrobble count of close to 15, and low values for the 25%, 50%, and 75% quartile barriers. It might be appropriate to visualize this distribution with a violin plot extended with a strip plot to achieve an easier read of the overall spread of values. As the minimum is 1, while the maximum is above 5000, it is reasonable to set a log scale for the x axis to aid the visualization.

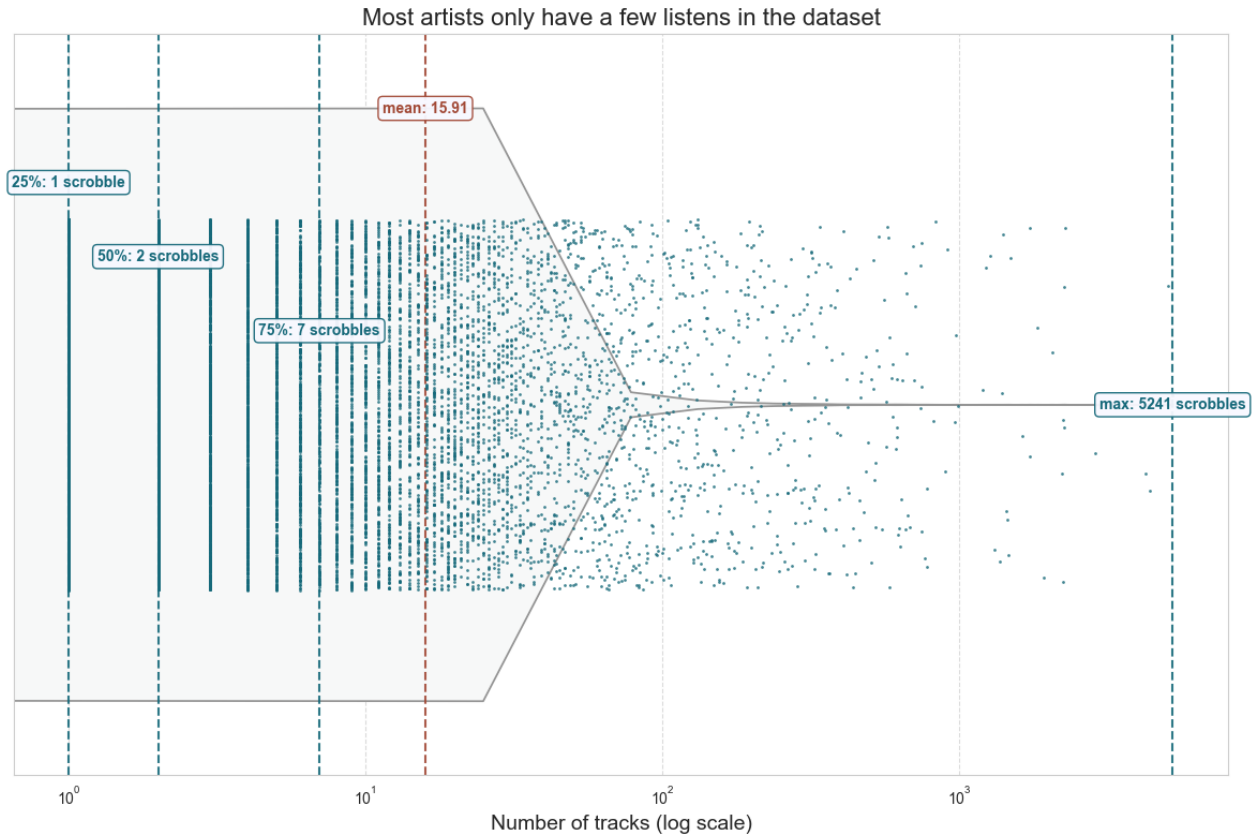


Figure 6: Most artists are only featured in the scrobble database with a few listens

2.3.3 Trusted companions

It might be interesting for the user to see if there are any artists present in every year of their scrobble data. We check if there are artists above the 25th/50th/75th etc. percentile in each and every year of the scrobble data, and try to answer the question: which artists have the lowest variance among these?

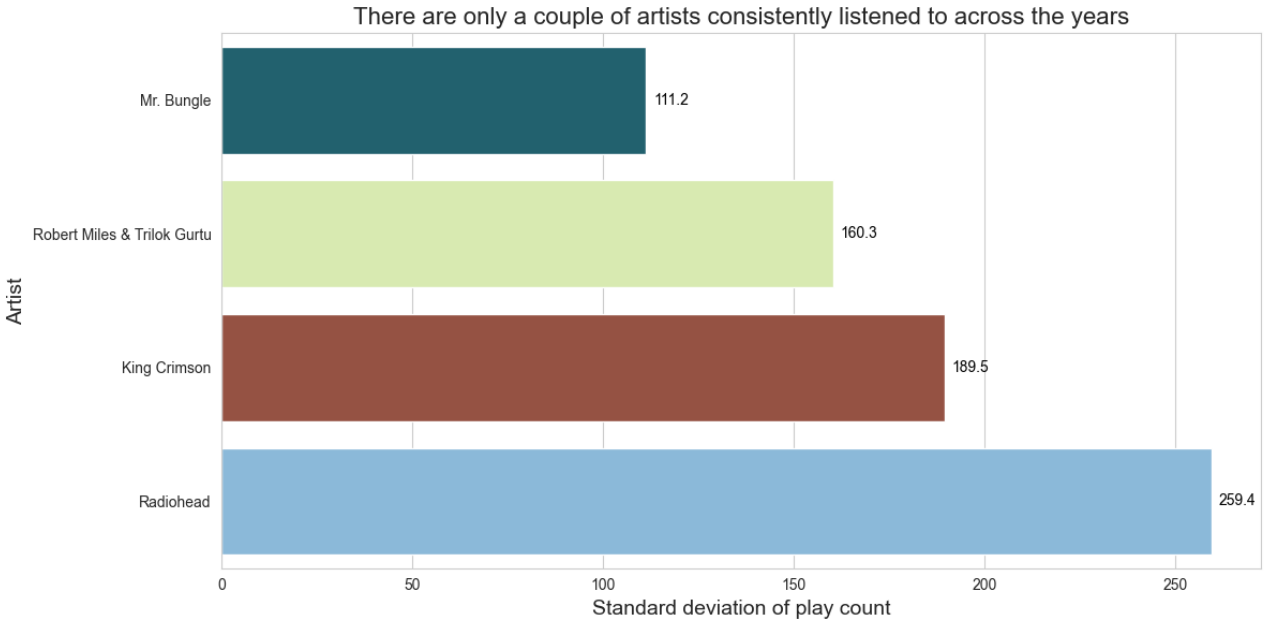


Figure 7: Artists with most consistent listening patterns (2006-2025)

2.4 Country-stat visuals

Data visualization, sitting "at the intersection of science and art" (Nussbaumer Knaflic, 2015, p. 17), aims to present information in a visual form that makes it easier for people to grasp the data that is presented.

Having gained an initial insight into the general tendencies of the data set, we are going further and explore extra visualization types users might be interested in.

2.4.1 Geoplotting of artist countries

Looking at an extensive set of publicly available music APIs (/github.com/public-apis), MusicBrainz is identified as the most ambitious and thorough contender in the field, with a clear intention to serve as a "universal lingua franca" (MusicBrainz, n.d.) for music identification with the MBID, a 36-char universally unique identifier at the heart of the operations. CanonFodder is querying MusicBrainz for the main country an artist is associated with. In enrich.py, a function is built to populate the artist_info table with ISO-2-style country values, also including MBIDs and any disambiguation comment used in record linkage/decoupling workflows.

Having gathered enough artist country data to put on the map, the Folium interface and a publicly available .geojson is used to build up a global map of artist countries with scrobble count values appearing interactively on hover.

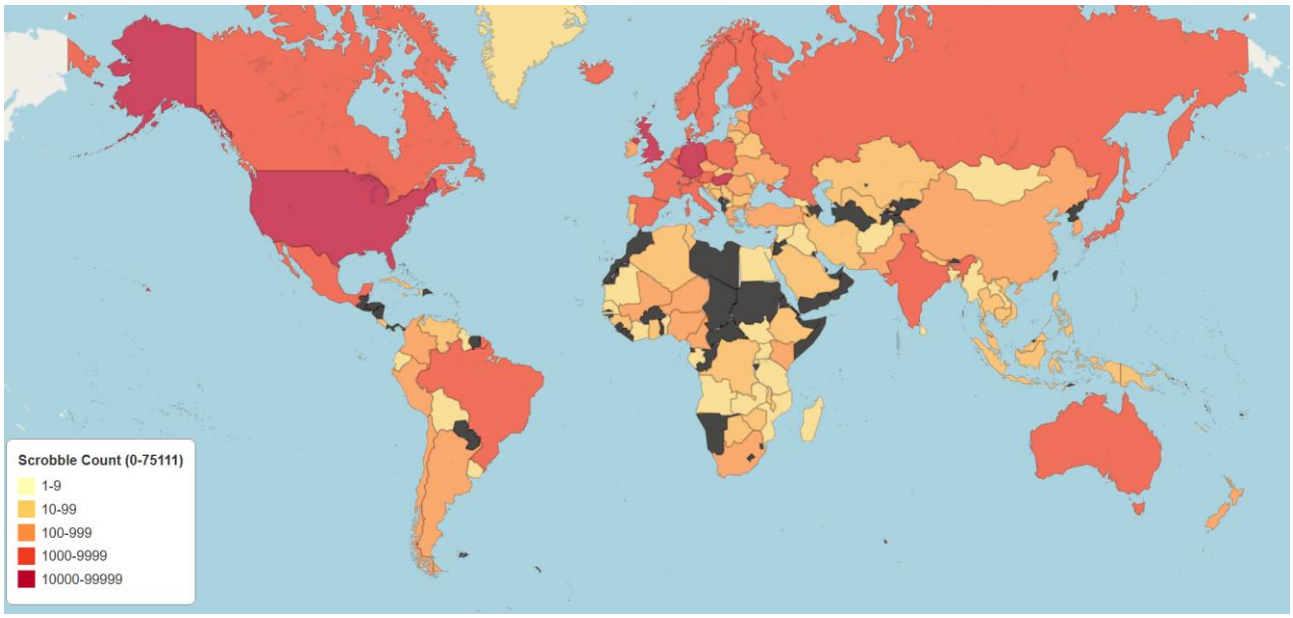


Figure 8: Putting scrobbles on the world map

A new df is built containing the number of scrobbles per country as well as the population of each country, and a scatter plot is constructed representing the covariance of 2 variables, country population, and scrobble no. for artist associated with the country.

For power users, a weighted per capita global map is created as well, shaded darker when the value is high, and shaded lighter when low with the same colour.

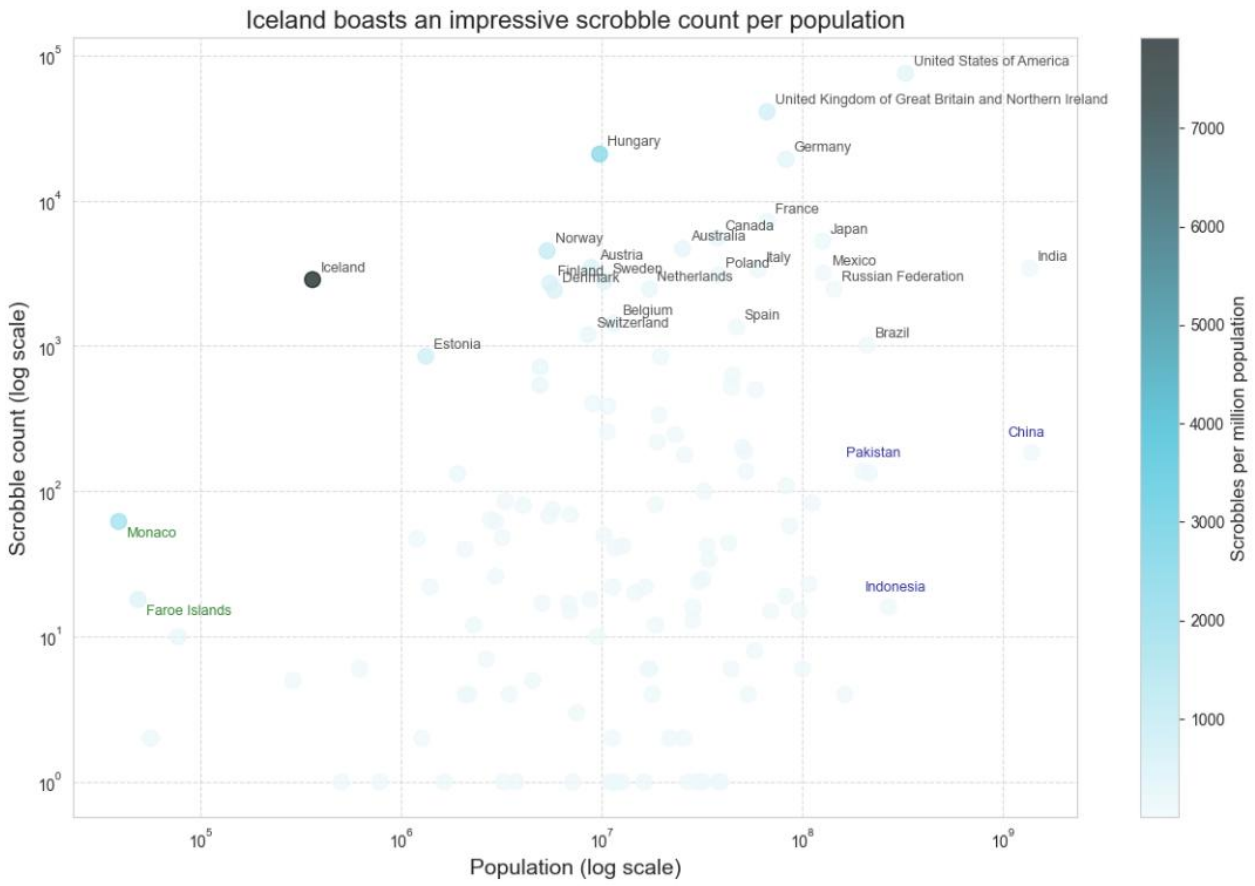


Figure 9: Scrobble count per country population on a log scale

2.4.2 UserCountry

Data can be added by the user registering the country they were located at on different dates, captured as a table with rows of non-overlapping date intervals. A typical use case is to filter statistics per user country location.

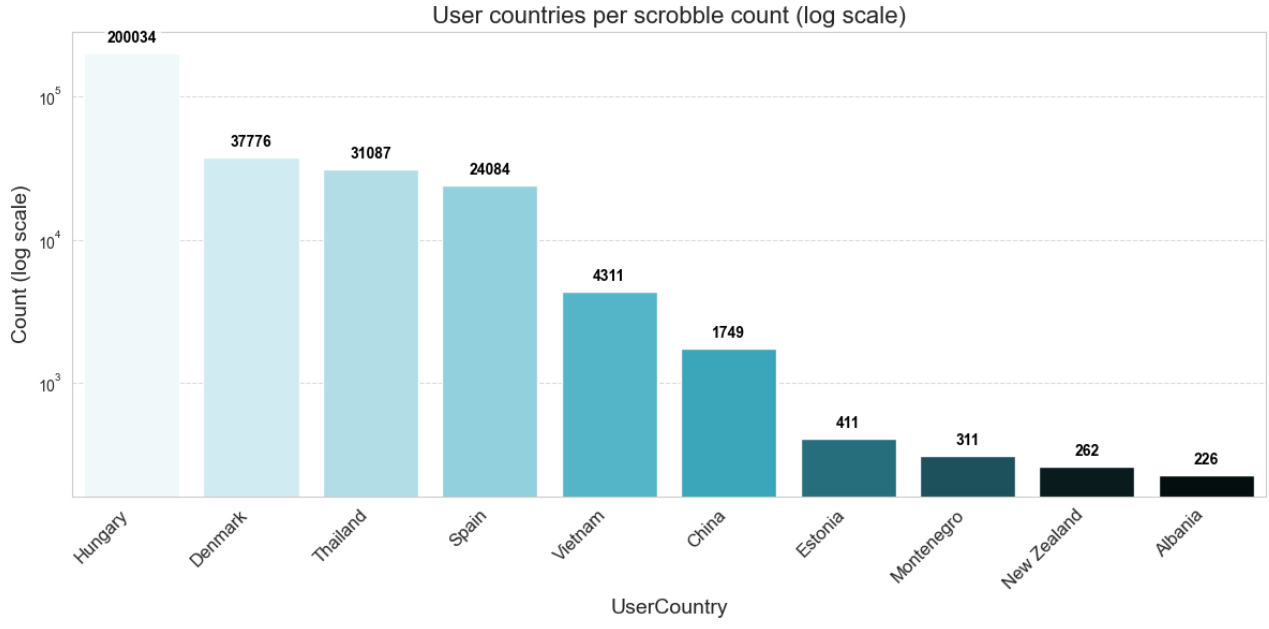


Figure 10: Bar chart of scrobbles per country where the user was located at the time of scrobbling

2.5 Comparison with literature

The exploratory workflow and visual repertoire developed for CanonFodder follow many of the recommendations that have matured in the EDA and data-visualisation literature, yet a few deliberate deviations—our palette system and the DB-centred tool chain—also shaped the insights we were able to produce. Tukey's infamous work (cf. e. g. Mueller, 2020) framed EDA as an iterative, question-raising exercise rather than a single go at descriptive statistics (David & Tukey, 1977). Our pipeline follows that spirit by going through lightweight profiling steps (null checks, duplicate detection, frequency scans) with richer diagnostics such as seasonality plots and variance summaries.

A recurring theme in modern textbooks is fit-for-purpose graph selection. For example, Wilke (2019) argues that stacked bars are for parts-of-a-whole across one categorical dimension, whereas line charts better express continuous change. We adopted Bohren-variant artist count share as stacked bars so readers can compare total volume and internal composition at once. The ridgeline plot used for the month-by-month distribution of listening activity adapts the "joyplot" popularised by Bryon & Wickham. Abdul-Rahman et al. (2024) quantifies how ridgelines outperform small-multiples when the number of facets is > 8—an empirical justification for our choice.

Several authors stress that hue should encode categories and lightness an ordered metric (Heer & Stone, 2012). Our palettes follow exactly that: the 5-hue base palette from Caribbean Current through Pale Dogwood in JSON/palettes.json give five perceptually-balanced colours, while sequential shades of the same base are reserved for density bands. Contrast-ratio checks were run to remain WCAG-AA compliant, matching guidelines proposed by Nussbaumer Knaflic (2015).

3. Conclusion and discussion

CanonFodder started out as a weekend hack to tidy up a listener's messy last.fm archive; two months later it has matured into a reproducible mini–data-platform that can ingest, clean, enrich and visualise millions of scrobbles on demand. Throughout the essay we tried to answer the simple question of which analysis techniques and visual encodings best illuminate long-term listening behaviour. It is a relatively new, and, thus, uncharted territory, but our end-to-end build now lets us answer the above question with evidence rather than opinion.

3.1 Methods recap

User-initiated workflow or an Airflow-driven pull from the last.fm API lands raw JSON in a lake layer, which is then normalised into a starlike schema via SQLAlchemy/Alembic migrations. Alias resolution through a manual CLI—backed by an XGBoost similarity model and manual overrides—fixed > 7 % of artist names, including the "incorrect top artist" bug cited in the introduction. MusicBrainz look-ups added country, MBID and disambiguation fields; user-provided "home country" intervals completed the context for a truly cosy data navigation experience.

3.2 Key results

Listening follows a steady annual groove with only a couple of larger dips and binges punctuating the time series data. July is consistently the quietest month, while December–January peaks hint at holiday listening rituals, or seasonal sequestration hinting at a Northern-hemisphere-based user. Only four “trusted companions”—artists consistently featured with high scrobble count every single year—survived the close-to-20-year span, confirming the canonicalisation workflow’s stability. When mapped per capita, Scandinavian countries punch far above their population weight, a nuance that a plain count map would have missed.

3.3 Discussion

The findings validate the palette-plus-plot mix. Stacked bars made relative artist shares readable at a glance, and a restrained hue scheme kept comparisons perceptually fair.

What could be better?

We postponed style analysis to stay focused on artist and time, but joining the pipeline with genre analysis, maybe providing a heatmap analysis of top 10 countries and top 10 tags. Weekly Airflow runs are fine for retrospection; a Kafka-based streaming ingest would support live “now playing” dashboards, or it is also possible to opt for a near-real-time strategy with Spark Streaming. Our WCAG-AA contrast checks are a start, yet colour-blind simulators and keyboard-only user tests remain on the to-do list.

Back in the Introduction we argued that musicologists and self-trackers need cleaner scrobble data and clearer visuals. CanonFodder now delivers both: canonised, query-ready tables on the back end and an opinionated—but proven—set of charts on the front. The project therefore completes the full circle from harvest → harmonise → highlight, turning a personal listening log into an evidence-rich narrative that others can replicate or extend. To sum up, we have shown that thoughtful engineering can turn raw scrobbles into musical data gems.

Bibliography

Abdul-Rahman, A., Morgan, W., Vukmirovic, A., & Yu, D.-Y. (2024). Probability density and information entropy of machine learning derived intracranial pressure predictions. *PLoS ONE*, *19*(7), 1–20. https://doi.org/10.1371/journal.pone.0306028

Abedjan, Z., Golab, L., & Naumann, F. (2017, January 1). *Data Profiling ; A Tutorial*. Proceedings of the 2017 ACM International Conference on Management of Data ; page 1747-1751, United States, North America. https://doi.org/10.1145/3035918.3054772

David, F. N., & Tukey, J. W. (1977). Exploratory Data Analysis. *Biometrics ; volume 33, issue 4, page 768 ; ISSN 0006-341X*. https://doi.org/10.2307/2529486

Elsden, C., Kirk, D. S., & Durrant, A. C. (2016). A Quantified Past: Toward Design for Remembering With Personal Informatics. *Human-Computer Interaction*, *31*(6), 518–557. https://doi.org/10.1080/07370024.2015.1093422

Heer, J., & Stone, M. (2012, January 1). *Color naming models for color selection, image editing and palette design*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems ; page 1007-1016, United States, North America. https://doi.org/10.1145/2207676.2208547

Kaur, J., Jain, K., Maana, M. S., & Singh, A. (2025). A Multi-Feature Analysis of Electric Vehicle Performance Using Statistical Techniques. *2025 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Electrical, Electronics and Computer Science (SCEECS), 2025 IEEE International Students' Conference On*, 1–6. https://doi.org/10.1109/SCEECS64059.2025.10940985

Mueller, J. (2020). *Data science programming all-in-one for dummies*. John Wiley & Sons, Inc. https://research.ebsco.com/linkprocessor/plink?id=b92c6f2c-8794-335f-a6e8-62bfb4e4c432

MusicBrainz. (n.d.). *MusicBrainz Identifier—MusicBrainz*. Retrieved April 29, 2025, from https://musicbrainz.org/doc/MusicBrainz_Identifier

Nussbaumer Knaflic, C. (2015). *Storytelling with Data [electronic resource]: A Data Visualization Guide for Business Professionals*.

https://research.ebsco.com/linkprocessor/plink?id=690778b9-e96f-3741-95df-1859664858dc

Torda, B. (2025). *CanonFodder* [Unpublished case study]. IU International University of Applied Sciences.

Wilke, C. (2019). *Fundamentals of data visualization: A primer on making informative and compelling figures*. https://research.ebsco.com/linkprocessor/plink?id=90600a2a-f388-3de3-ab84-ea27ce20703f

APPENDIX: Extra information on CanonFodder

Stakeholder identification & prioritisation

Priority	Stakeholder	Motivation
P0	Core developers / maintainers	Build & operate CF
P0	Power users / Quantified-Self music enthusiasts	Analyse personal listening history
P1	Casual last.fm users	Occasional insight into listening habits
P1	Researchers (musicology, data science)	Reliable structured dataset
P2	Integration partners (ListenBrainz, streaming platforms)	Optional data exchange
P3	Business sponsors	Potential monetisation & brand presence

Assumptions, dependencies, constraints

- **Python ≥ 3.12**, PEP-8 compliance, 80 % test coverage
- Default RDBMS = MySQL 8; SQLite/Postgres supported
- External APIs – last.fm *(API key)*, MusicBrainz (rate-limited), public internet
- Local filesystem read/write for parquet cache
- Docker container run via `docker run canonfodder`

Software requirements document

System interfaces

ID	Interface	Direction	Protocol / Lib	Description
I-01	last.fm API	Inbound	HTTPS REST	user.getRecentTracks, user.getInfo
I-02	MusicBrainz API	Inbound	HTTPS REST	Artist lookup & search
I-03	RDBMS	Bidirectional	SQLAlchemy 2.0	MySQL / PostgreSQL / SQLite
I-04	File System	Outbound	Parquet (pyarrow)	Eval DB export

I-05	CLI	Inbound	argparse / questionary	User launches workflows
I-06	Airflow DAG	Inbound	Python API	Scheduled orchestration

Functional requirements

ID	Description
FR-01	**Fetch scrobbles** – Pull recent tracks for a user since the last stored timestamp and persist raw JSON.
FR-02	**Normalise scrobbles** – Rename columns, convert UTS→UTC datetime, and remove duplicates.
FR-03	**Bulk insert** – Load normalised rows into scrobble table with dialect-aware conflict ignore.
FR-04	**Artist enrichment** – For new MBIDs, fetch country & aliases; cache in artistcountry.
FR-05	**Canonisation** – Group artist name variants, store mapping in AVC, apply to scrobble history.
FR-06	**Parquet export** – Dump star schema to parquet files on demand or based on schedule.
FR-07	**BI frontend** – Provide menu to launch data gathering and open interactive dashboards using custom colour palettes.
FR-08	**Retry & back-off** – Network requests retry up to *8* times with exponential back-off.
FR-09	**Workflow orchestration** – One-click Airflow DAG (cf_ingest) covering FR-01-FR-07.
FR-10	**Docker distribution** – Build image with code, dependencies, docs, dashboards.

Performance requirements

- Capable of ingesting **10 000 scrobbles per minute** on consumer-grade hardware.
- End-to-end Airflow DAG completes within **15 min** for 1 million scrobbles.

Data requirements

- All timestamps stored in UTC with timezone awareness (TIMESTAMP WITH TIME ZONE)
- Primary keys, unique constraints and foreign keys per ERM
- Eval DB parquet partitioned by year for scrobble table

Reliability & availability

- **Retry logic** on HTTP 5xx and network errors (see FR-08)
- Alembic migrations version-control schema
- Unit + integration tests; CI fails if coverage < 80 % or linting errors

Security & privacy

- Secrets stored in .env; mounted via Docker secrets in production
- Only read-only last.fm API scope
- GDPR compliance: user can *purge* all data via CLI command

Quality attributes

- **Maintainability** – Docstrings, PEP-8, and consistent type hints
- **Portability** – Runs on Linux/Mac/Windows via Docker
- **Usability** – Guided CLI prompts; colour-blind-safe palettes

Constraints & limitations

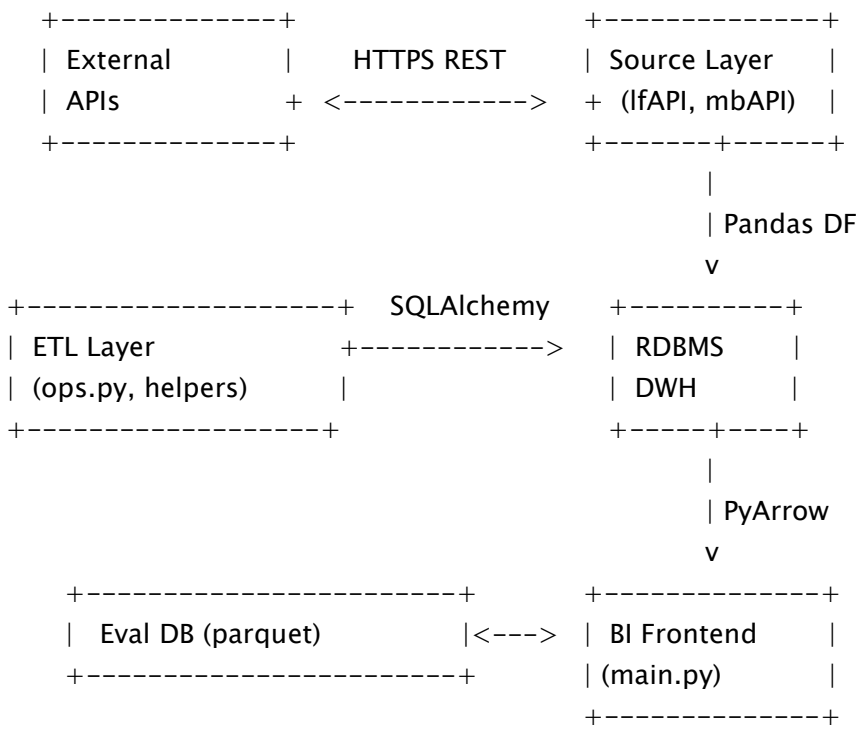
- Rate limits: last.fm ≈ 5 req/sec; MusicBrainz 1 req/sec. The pipeline must honour these to avoid bans.
- Offline mode requires cached parquet.

Acceptance criteria

- All functional requirements demonstrably fulfilled.
- Ingest 100 k scrobbles without duplication.
- Dashboard renders top-10 artists within 2 seconds.

High-Level design

Architectural overview



Key components

Component	Responsibilities
Source layer	Fetch scrobbles (lfAPI), fetch artist data (mbAPI), resilient client (HTTP/client.py)
ETL layer	Data normalisation, canonicalisation, enrichment, bulk inserts, parquet export
Persistence	RDBMS schema; Alembic migrations; evaluation parquet store
Orchestration	Airflow DAG & CLI glue commands
BI frontend	CLI menu & notebooks/dashboards using palettes.json

Data flow

1. Airflow triggers **fetch & ingest** task.
2. lfAPI pulls new scrobbles, returns JSON.
3. DB/ops.py converts to DataFrame → dedupe → bulk insert.
4. mbAPI enriches missing artist rows; bulk upsert.
5. Canonisation (helpers/cli) groups variants, updates AVC.
6. Parquet export updates Eval DB.
7. User opens dashboard to explore.