

Interim Project Documentation

How Trendy is your playlist

SG Beats

Nur Ain, Kin Meng, Jarrel, Teck Cheng

Introduction.....	2
Overview.....	2
Problem Statement.....	2
Data Sources.....	3
Data Dictionary.....	3
Methodology/Approach.....	4
Tech Stack/ Tools.....	4
ETL Process.....	4
Extraction.....	4
Transformations.....	4
Loading.....	4
Architecture Diagram.....	5
Database Schema/ ERD Diagram.....	5
Table Structure.....	6
Data Validation.....	6
Challenges(Solutions/Workarounds)/Limitations.....	7
Challenges.....	7
Limitations.....	7
Findings/Analysis.....	9
Key Results.....	9
Insights.....	10
Simplifications and Focus.....	10
Conclusion.....	11
What's Next.....	11
Reference.....	12

Introduction

Overview

The project aims to analyse the music preferences across team members by calculating a “trendy score” based on the average popularity metrics (0-100) of songs in their personal Spotify playlists. Using Spotify’s API data and web scraping techniques, the project leverages actionable metrics to compare scores within the group. The findings provide insight into whose music taste aligns more closely with mainstream trends and who favors niche preferences. Apart from this fun competition, this project showcases a successful ETL process, transforming raw Spotify data into meaningful conclusions.

Problem Statement

Music tastes often spark lively debates about mainstream vs niche preferences, but these discussions are subjective and lack quantitative backing. This project addresses the challenge by using Spotify’s popularity metrics to establish an objective framework for comparison.

Data Sources

The project utilizes the following data sources:

1. Spotify playlist: A personalised selection curated by each individual, showcasing their unique music taste. For this project, only publicly accessible playlists are utilized, ensuring that the data remains easily retrievable while preserving individual privacy
2. Spotify API: The API provides popularity metrics (0-100) for the songs in the playlists. These metrics quantify the mainstream appeal of each track.

Data Dictionary

Field Name	Description	Source
Song Title	Name of the song extracted from the Spotify playlist	Web Scraping, Spotify API
Album	Name of the artist(s) associated with the song	Web Scraping, Spotify API
Artist	The album in which the song is featured	Web Scraping, Spotify API
Spotify URL	A direct link to detailed track information retrieved from Spotify API	Spotify API
Popularity	A numerical value (0-100) indicating the song's popularity metric provided by Spotify API	Spotify API

Methodology/Approach

Tech Stack/ Tools

- Programming language: Python for data extraction processing and analysis
- Libraries:
 - **Web Scraping:** `Selenium` for extracting data from Spotify playlists.
 - **Spotify API:** `Spotipy` for accessing song popularity and metadata.
 - **Database Connection:** `psycopg2` and `psycopg2-binary` for interacting with PostgreSQL databases.
 - **Data Manipulation:** `Pandas`, `numpy` for cleaning and preparing data.
 - **SQL Management:** `SQLAlchemy` for database modeling and ORM (Object-Relational Mapping).
 - **Visualization:** `Plotly.graph_objects` for creating interactive graphs and charts.
 - **API Requests:** `requests` for making HTTP calls to gather necessary data.
 - **Time Management:** `time` for handling time-based operations like delays.
- Database: pgAdmin for storing the extracted and processed data

ETL Process

Extraction

- Web scraping is employed to collect song titles, artist names, and album details from publicly accessible Spotify playlists.
- Spotify API is used to retrieve song popularity metrics.

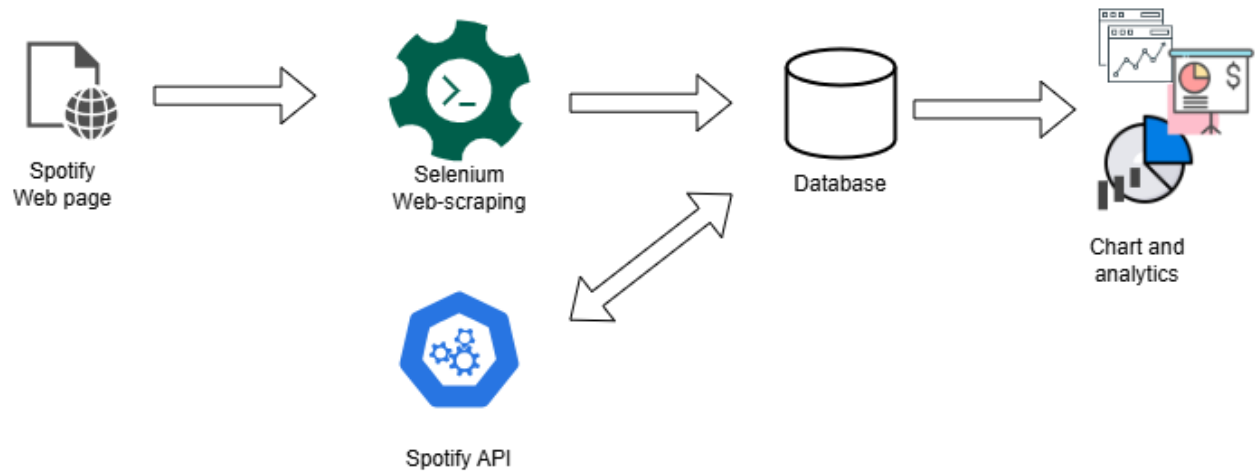
Transformations

- Data cleansing: Removing duplicates and handling missing values.
- Data normalization: Standardizing popularity scores for consistent comparison.

Loading

- All processed data is stored in the Postgres database for easy querying and analysis.

Architecture Diagram



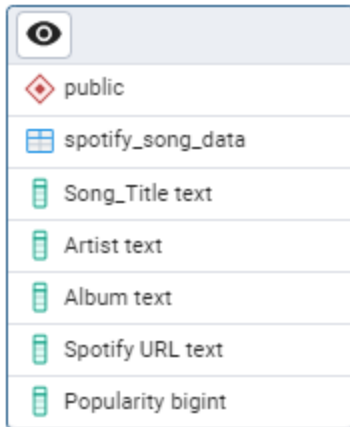
Database Schema/ ERD Diagram

Tables

- Playlist_data: Indexes, song_title, artist, album

public
playlist_data
Index bigint
Song_Title text
Artist text
Album text

- Spotify_song_data: song_title, artist, album, spotify_url, popularity



public
spotify_song_data
Song_Title text
Artist text
Album text
Spotify URL text
Popularity bigint

Table Structure

Our current approach utilizes a standalone table structure, where each table operates independently without relational links between them. While this method allowed us to simplify the project and focus on its main objective—showcasing a successful ETL process—it also came with certain limitations. The absence of relationships between tables meant we couldn't effectively leverage database principles like foreign keys to establish connections, which would have enabled streamlined querying and better data organization.

For example, if we had implemented relationships such as linking the **Users** table to the **Tracks** table via a foreign key, we could have easily identified which team member was associated with specific tracks. Similarly, connections between playlists and tracks could have facilitated deeper insights, such as grouping songs by popularity per individual playlist. However, due to time constraints and the priority of achieving actionable metrics, we opted for standalone tables as a practical solution.

Data Validation

- **Spotify API Validation:** Ensuring accurate popularity scores by handling API limitations and rate limits.
- **Web Scraped Data Validation:** Regular checks for completeness and integrity of extracted data (e.g., avoiding empty or mismatched fields).
- **Database Constraints:** NOT NULL constraints for critical fields.

Challenges(Solutions/Workarounds)/Limitations

Challenges

1. **Initial Focus on Top Genres:** The project originally aimed to identify the Top 3 popular genres of songs charting locally (Singapore) and globally. However, the lack of a **Genre** field in existing Kaggle datasets and Spotify API data extracts rendered this objective unfeasible.
 - a. **Solution/Workaround:** The scope was redefined to focus on analyzing song popularity metrics from personal playlists, ensuring the project could proceed with actionable data.
2. **Private Spotify Playlists:** Web scraping personal playlists posed a challenge due to restrictions on accessing private playlists. These required additional code to bypass login restrictions or significant delays to scrape after login.
 - a. **Solution/Workaround:** We shifted our focus to publicly accessible playlists, simplifying the implementation while respecting user privacy.
3. **Limitations to Obtainable Data:** The data we could retrieve was restricted, as only song titles, artist names, album details (via web scraping), and popularity metrics (via Spotify API) were available.
 - a. **Solution/Workaround:** The team decided to work within these constraints, prioritizing actionable metrics over broader dataset ambitions.
4. **Relational Data:** We initially intended to implement relationships between tables in the database (e.g., linking users to their playlists and tracks). However, time constraints prevented this, leading to the use of standalone tables.
 - a. **Solution/Workaround:** By adopting a standalone table structure, we successfully completed the ETL process and analysis while maintaining a simplified approach.
5. **Limited Song Extraction from Playlist URL (~50 Songs):** Despite scroll simulation and wait-time adjustments, the scraper only retrieved ~50 songs. This limitation is due to Spotify's dynamic content loading, which prevents Selenium from triggering full playlist loading reliably.
 - a. **Workaround:** While more advanced scraping techniques (e.g., headless browser automation, deeper API use) could improve this, they were outside the project scope. This limitation is acknowledged in the documentation to set expectations.

Limitations

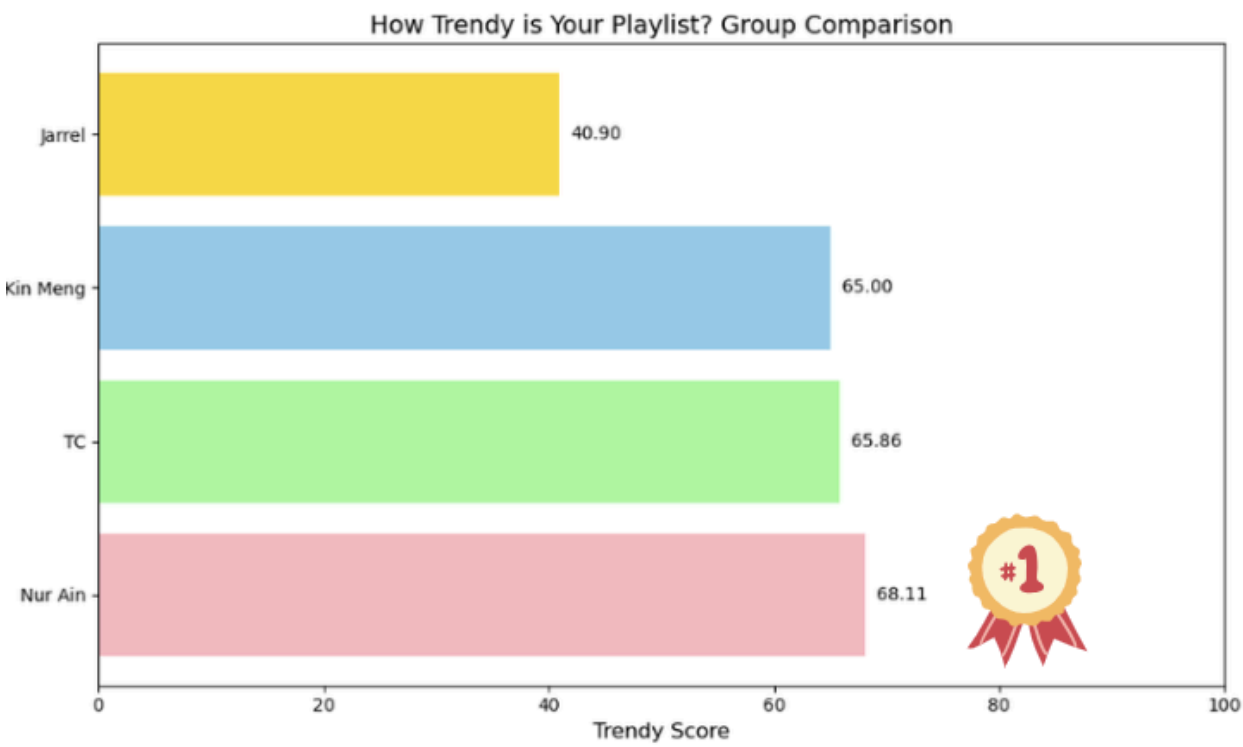
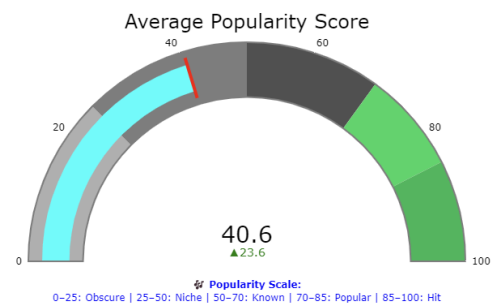
1. **Genre Analysis Excluded:** Without genre data, the project could not explore trends in popular music styles, limiting insights to popularity scores alone.
2. **Simplified Database Design:** The lack of relational structures in the database restricted the ability to perform advanced queries or derive insights based on connections (e.g., grouping tracks by user or playlist).

3. **Narrow Data Focus:** The analysis relied solely on song popularity as the key metric, without additional context such as global or local chart comparisons, which were initially considered.
4. **Data Accessibility:** The dependence on public playlists excluded the music preferences of individuals using private playlists, reducing inclusivity within the dataset.

Findings/Analysis

Key Results

Through our ETL process, we transformed Spotify data into actionable insights and a fun competition within the team. By calculating the average popularity scores (0-100) of songs in the playlists, we successfully ranked our team members' playlists. This led to identifying a clear "trendiest" winner—someone whose musical taste aligned the most with mainstream preferences. Conversely, we also highlighted members with more niche tastes, demonstrating diverse music personalities in the group.



The diagram above shows who is the trendiest among our team

Insights

1. **Popularity Patterns:** Playlists with higher average scores revealed a preference for widely recognized songs, suggesting an affinity for trending music.
2. **Diversity in Tastes:** Lower average scores reflected more unconventional or niche choices, highlighting unique personal preferences that deviate from mainstream trends.
3. **Data-Driven Discussions:** By using measurable metrics like popularity scores, playful debates about music tastes became grounded in objective, data-backed comparisons.

Simplifications and Focus

The project's scope was adjusted to prioritize internal group comparisons rather than including broader metrics like Singapore or Global Top 50 charts. This decision streamlined the analysis, enabling us to achieve the primary objective of showcasing a successful ETL process without overcomplicating the methodology.

Conclusion

This project took raw Spotify data and turned it into something fun and insightful which is a way to compare our team's music tastes and see who has the "trendiest" playlist.

- We ranked playlists using Spotify's popularity scores
- One playlist clearly came out on top
- It showed how data can settle even playful debates and make it more engaging

What's Next

The team identified several ways to take this further:

- 1) **Track Trends over time** - see how song or artist popularity rises and falls over weeks or months or years
- 2) **Look at the Genres** - group songs by genre to find which styles are most popular in playlists
- 3) **Use song features** - break song down by things like energy, danceability or tempo to understand what kind of mood each playlist gives off
- 4) **Match playlist vibes** - compare playlist by their overall mood like chill vs upbeat
- 5) **Add Regional Trends** - use location data to show how music preferences vary by country
- 6) **Explore artist collaborations** - show how artists work together by mapping who appears on songs together

Reference

Spotify Tracks Dataset. (2022, October 22). Kaggle.

<https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset>

CoderzColumn. (2023, February 24). *Gauge Chart using Matplotlib | Python* [Video].

YouTube. <https://www.youtube.com/watch?v=loU6argFsQ0>

Home | Spotify for Developers. (n.d.). <https://developer.spotify.com/>

Wikipedia contributors. (2025, March 18). *Web colors*. Wikipedia.

https://en.wikipedia.org/wiki/Web_colors