Spotify Data Exploration Project

CS126 - Database Systems

Anthony Sayre

Project Overview

- Two public datasets from Kaggle.
  - Spotify Data
  - US Artist Data
- Track data focused on artist songs, musical features like energy, valence and key.
- Artist data focused on the collection of songs and demographics like ranked popularity of artist.
- My goal was to explore patterns in music popularity and characteristics using SQL.

Why These Datasets?

- I chose them because they had:
  - Enough records for real analysis
  - Clean structure
  - Interesting musical attributes
- Some of the questions I wanted to answer:
  - Do popular artists make high-energy songs?
  - Which genres are happiest or most danceable?
  - Are there trends in musical keys or tempo?

Database Design and Schema

Database Design

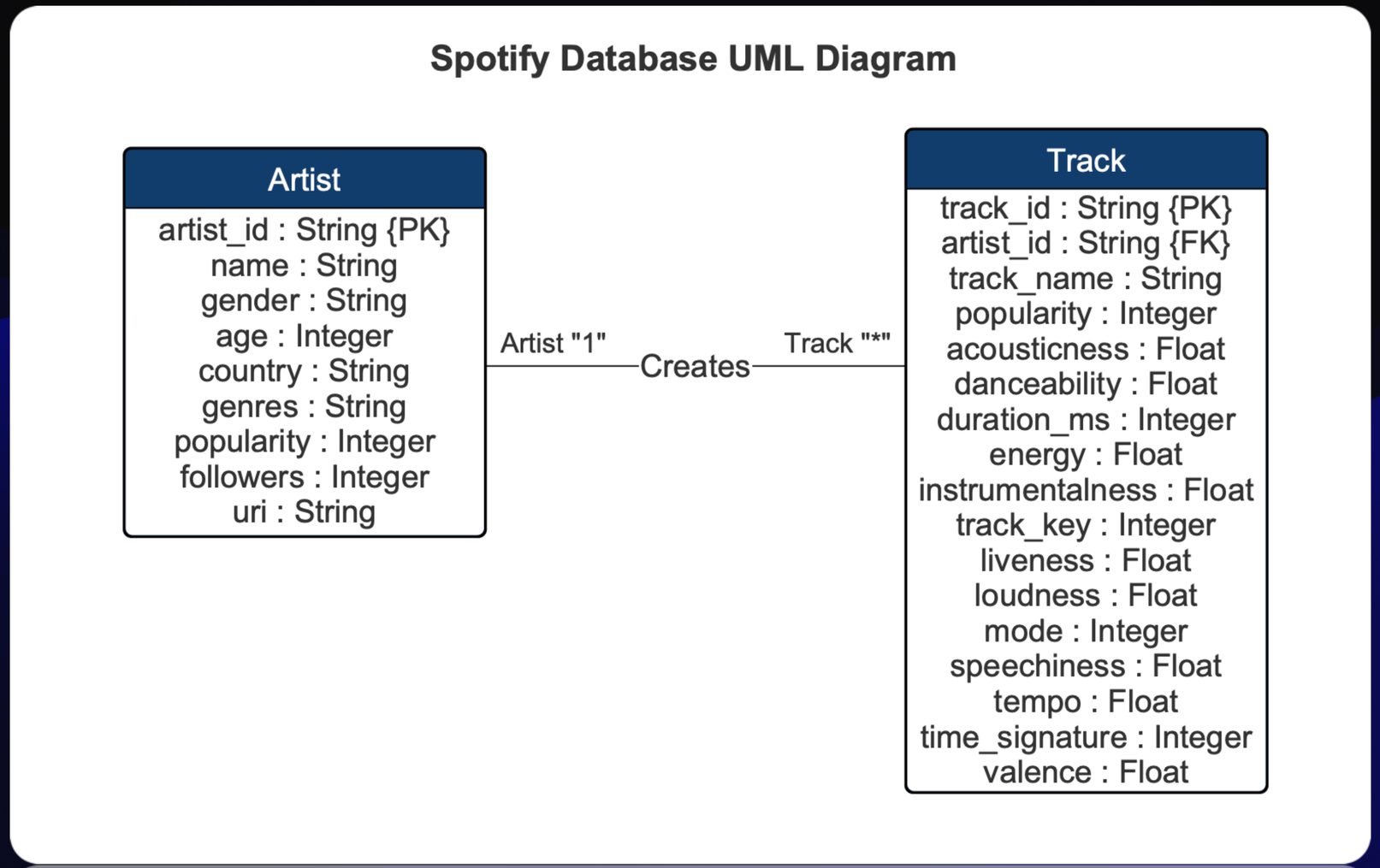
- I created a MySQL database called spotify_db
- Designed it in 3NF with two tables:
  - artists: artist info (name, genre, popularity, etc.)
  - tracks: track details + audio features
- Each track references an artist using artist_name
- I kept the structure simple and easy to query

```sql
-- Create artists table (columns in CSV order)
CREATE TABLE artists (
    Name VARCHAR(255),
    ID VARCHAR(100) PRIMARY KEY,
    Gender VARCHAR(50),
    Age INT,
    Country VARCHAR(100),
    Genres TEXT,
    Popularity INT,
    Followers INT,
    URI VARCHAR(255)
);

-- Create tracks table (columns in CSV order, renamed 'key' to 'track_key')
CREATE TABLE tracks (
    genre            VARCHAR(100),
    artist_name      VARCHAR(255),
    track_name       VARCHAR(255),
    track_id         VARCHAR(100) PRIMARY KEY,
    popularity       INT,
    acousticness     FLOAT,
    danceability     FLOAT,
    duration_ms      INT,
    energy           FLOAT,
    instrumentalness FLOAT,
    track_key        VARCHAR(10),
    liveness         FLOAT,
    loudness         FLOAT,
    mode             VARCHAR(10),
    speechiness      FLOAT,
    tempo            FLOAT,
    time_signature   INT,
    valence          FLOAT
);
```

A snippet of schema.sql

UML Diagram


Spotify Database UML Diagram
Artist
artist_id : String {PK}
name : String
gender : String
age : Integer
country : String
genres : String
popularity : Integer
followers : Integer
uri : String
Artist "1"
Creates
Track "*"
Track
track_id : String {PK}
artist_id : String {FK}
track_name : String
popularity : Integer
acousticness : Float
danceability : Float
duration_ms : Integer
energy : Float
instrumentalness : Float
track_key : Integer
liveness : Float
loudness : Float
mode : Integer
speechiness : Float
tempo : Float
time_signature : Integer
valence : Float

Data Exploration

Data Exploration

- Total of 14 SQL queries
  - Aggregations (counts, averages)
  - Joins between artists and tracks
  - Subqueries and a view
- Looked for trends like:
  - Most common genres and keys
  - Average valence (happiness) by genre
  - Popular artists and tracks

```sql
-- 1. Count of artists by country
SELECT Country, COUNT(*) AS artist_count
FROM artists
GROUP BY Country
ORDER BY artist_count DESC;

-- 2. Count of tracks by genre
SELECT genre, COUNT(*) AS track_count
FROM tracks
GROUP BY genre
ORDER BY track_count DESC;

-- 3. Top 10 most popular artists
SELECT Name, Popularity
FROM artists
ORDER BY Popularity DESC
LIMIT 10;

-- 4. Top 10 tracks by energy
SELECT track_name, energy
FROM tracks
ORDER BY energy DESC
LIMIT 10;

-- 5. Average danceability by artist
SELECT artist_name, ROUND(AVG(danceability), 3) AS avg_danceability
FROM tracks
GROUP BY artist_name
ORDER BY avg_danceability DESC
LIMIT 10;

-- 6. Tracks by U.S. artists (join)
SELECT t.track_name, a.Name AS artist, a.Country
FROM tracks AS t
JOIN artists AS a
  ON t.artist_name = a.Name
WHERE a.Country = 'US'
LIMIT 10;

-- 7. Number of tracks per artist (join & HAVING)
SELECT a.Name AS artist, COUNT(t.track_id) AS total_tracks
FROM artists a
JOIN tracks t ON a.Name = t.artist_name
GROUP BY a.Name
HAVING total_tracks > 5
ORDER BY total_tracks DESC
LIMIT 10;
```

A snippet of queries.sql

Interesting Insight 1

Query 5 – Average Danceability by Artist

```sql
-- 5. Average danceability by artist
SELECT artist_name, ROUND(AVG(danceability), 3) AS avg_danceability
FROM tracks
GROUP BY artist_name
ORDER BY avg_danceability DESC
LIMIT 10;
```

artist_name	avg_danceabil...
Vanilla Ice	0.978
Mr. Symarip	0.976
[dunkelbunt]	0.975
Bankroll Fresh	0.969
Biboulakis	0.964
Analogik	0.963
Ocho Drippin	0.96
Afro B	0.957
Goon Twinn	0.957
Sugababes	0.954

Interesting Insight 2

Query 8 – Average Valence by Genre

```sql
-- 8. Average valence per genre
SELECT genre, ROUND(AVG(valence), 3) AS avg_valence
FROM tracks
GROUP BY genre
ORDER BY avg_valence DESC;
```

genre	avg_valence
Reggae	0.68
Children's Music	0.676
Reggaeton	0.66
Ska	0.647
Blues	0.58
Country	0.535
Dance	0.518
Rock	0.514
Jazz	0.51
Pop	0.506
Soul	0.484
Hip-Hop	0.462
Indie	0.452
Alternative	0.449
Movie	0.448
R&B	0.445
Rap	0.442
Anime	0.442
Children’s Music	0.439
Folk	0.427
Comedy	0.413
Electronic	0.385
A Capella	0.329
World	0.295
Classical	0.216
Opera	0.19
Soundtrack	0.113

Interesting Insight 3

Query 14 – Average Energy by Musical Key

```sql
-- 14. Average energy by track key
SELECT t.track_key, ROUND(AVG(t.energy), 3) AS avg_energy
FROM tracks t
JOIN artists a ON t.artist_name = a.Name
GROUP BY t.track_key
ORDER BY avg_energy DESC;
```

track_key	avg_energy
B	0.657
F#	0.645
C#	0.642
G#	0.623
E	0.62
A	0.614
G	0.594
A#	0.592
D	0.588
F	0.58
C	0.579
D#	0.513

Takeaways

What I learned?

- The importance of normalizing data
- Further exploration into the music domain
- Even basic joins and filters lead to solid insights
- If I did it again, I’d break out the audio features into a separate table

The End. Thank you!