SPOTIFY TRACK DATA ANALYTICS

End-to-End Data Pipeline | Python + MySQL + Jupyter

Project Portfolio Documentation

Prepared By

Abdul Rahman

Aspiring Data Analyst

Technologies Used

- **Python (Pandas, Regex, Matplotlib)**
- **Spotify API – Spotipy**
- **MySQL (mysql-connector-python)**
- **Jupyter Notebook**
- **Excel / CSV**

Project Category:

ETL Pipeline | API Data Extraction | Data Analytics

December 2025

Project Structure

Spotify_Data_Analytic/

|

├── Code/

| ├── spotify_project.py

| ├── spotify_mysql_project.py

| ├── spotify_mysql_url_project.py

| ├── track_url

| └── spotify_track_data.csv

|

├── Data/

| └── spotify_10_tracks.csv

|

├── Exports/

| ├── Most popular song.csv

| ├── longest track.csv

| ├── shortest track.csv

| ├── unique artists.csv

| ├── average duration.csv

| ├── open_spotify_tracks.csv

| └── (All SQL query outputs)

|

├── Notebook/

| ├── spotify_track_analysis_Abdul.ipynb

| ├── spotify_track_analysis_Abdul.html

| └── spotify_track_analysis_Abdul.pdf

|

└── Spotify_Track_Project/ (Backup)

Project Overview

This project is an end-to-end data pipeline built using Python, Spotify Web API, MySQL, and Jupyter Notebook.
It automates the extraction of Spotify track metadata using track URLs, stores the processed data in a MySQL database, and performs analytical queries to generate meaningful insights.

The project demonstrates skills in:

- API data extraction
- Regex-based track ID processing
- Database storage & SQL analytics
- CSV exporting
- Data analysis using Python & Jupyter Notebook

This showcases a real-world ETL (Extract → Transform → Load) pipeline implemented with clean, reusable code.

Project Objectives

The main objectives of this project are:

- Extract metadata for multiple Spotify tracks using the Spotify Web API
- Automate processing of multiple track URLs
- Clean and transform metadata (name, artist, album, duration, popularity)
- Load structured data into MySQL for storage and querying
- Run analytical SQL queries to explore track patterns
- Export all insights as CSV files
- Create a summary analysis using Jupyter Notebook

This section clearly shows what the project achieves.

Tools & Technologies Used

This project uses a combination of programming, database, and analytical tools to build a complete end-to-end ETL and analytics workflow.

1. Python

Used for building the extraction and transformation logic.

- Data extraction from Spotify API
- Regex for track ID extraction
- Data cleaning & transformation
- Writing data into MySQL

Libraries Used:

- spotipy – Spotify Web API integration
- re – Regular expressions
- pandas – Data manipulation
- matplotlib – Visualization
- mysql-connector-python – Database connectivity

2. Spotify Web API (Spotipy)

- Fetches track metadata
- Provides artist, album, popularity, and duration information
- Used to automate retrieval for multiple track URLs

3. MySQL Database

Used to store the structured dataset.

- Table: **spotify_tracks**
- Stores cleaned metadata
- Enables SQL queries and analytical operations

4. SQL

Used for analytical queries such as:

- Most popular song
- Shortest & longest track
- Track popularity distribution
- Unique artists
- Average duration
- Songs above average popularity

5. Jupyter Notebook

Used for:

- Data analysis
- Visualizations
- Presentation-ready results
- Exporting HTML and PDF reports

6. Excel / CSV

Used for:

- Exporting SQL results
- Storing transformed datasets
- Delivering clean output files

7. PyCharm (IDE)

Used for coding and running:

- API scripts
- ETL pipeline
- Database operations

Workflow (ETL Pipeline)

This project follows a complete **ETL Pipeline** – Extract, Transform, Load – to process Spotify track data in a structured and automated manner.

1. Extract

This is the first step of the pipeline, where raw data is collected.

Steps Involved:

- Read multiple Spotify track URLs from a text file (track_url)
- Extract the **track ID** from each URL using **Regular Expressions (Regex)**
- Authenticate with Spotify API using **Client ID** and **Client Secret**
- Fetch track metadata for each URL using the Spotify Web API

Data Extracted from API:

- Track Name
- Artist Name
- Album Name
- Popularity Score
- Duration in milliseconds

This data is collected in JSON format and passed on to the transformation stage.

2.Transform

This stage converts the raw metadata into a clean and usable dataset.

Steps:

- Convert duration from milliseconds → **minutes**
- Keep only relevant fields
- Convert metadata into a **Python dictionary**
- Load the dictionary into a **Pandas DataFrame**
- Validate the data structure before loading into MySQL

Purpose:

- Cleans the metadata
- Ensures uniform structure
- Prepares data for storage and analysis

3. Load

In this step, the cleaned data is stored into a relational database.

Steps:

- Connect to **MySQL database** (spotify_db)
- Create a table: **spotify_tracks**
- Insert each track as a separate row using prepared SQL statements
- Use Jupyter Notebook to verify

Outcome

- Database now contains all the extracted & transformed tracks
- Ready for SQL analysis and reporting

4. Analysis (SQL + Jupyter Notebook)

Once data is stored, multiple SQL queries are run to generate business insights.

Insights Generated:

- Most popular track
- Longest / Shortest track
- Tracks above 5 minutes
- Tracks above average popularity
- Unique artists
- Track popularity distribution
- Full dataset export

Results are saved as individual **CSV files** under the **Exports** folder.

5. Reporting

Final stage of the pipeline, where insights are presented professionally.

Outputs:

- Jupyter Notebook (**.ipynb**)
- HTML report
- PDF report

These reports contain:

- SQL outputs
- Visualizations
- Summary insights

Python Code Snippets

This section includes important parts of the ETL pipeline implemented using Python. Only the key logic blocks are shown here. The full source code is available in the **Code** folder and on GitHub.

Extracting Track ID from Spotify URL (Regex)

This code extracts the Track ID from a Spotify URL using regular expression

```
pattern = r"track\/([A-Za-z0-9]+)"
match =   re.search(pattern, url)
track_id = match.group(1)
```

Fetching Track Metadata from Spotify API

This block authenticates with Spotify and retrieves metadata for each track.

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

sp = spotipy.Spotify(auth_manager=SpotifyClientCredentials(
   client_id=client_id,
   client_secret=client_secret
))

track_data = sp.track(track_id)
```

Transforming Data (Duration Conversion)

Converting duration from milliseconds to minutes.

```
duration_ms = track_data["duration_ms"]
duration_minutes = round(duration_ms / (1000 * 60), 2)
```

Loading Data into MySQL Database

This code inserts the cleaned metadata into the MySQL table.

```
import mysql.connector

query = """
INSERT INTO spotify_tracks (track_name, artist, album, popularity,
duration_minutes)
VALUES (%s, %s, %s, %s, %s)
"""

values = (name, artist, album, popularity, duration_minutes)

cursor.execute(query, values)
connection.commit()
```

Reading URLs From File

This section processes multiple URLs stored inside the track_url file.

```
with open("track_url", "r") as file:
  urls = file.readlines()

for url in urls:
  url = url.strip()
  # extract track_id and process each track
```

Creating DataFrame and Saving to CSV

```
import pandas as pd

df = pd.DataFrame(all_tracks)
df.to_csv("spotify_10_tracks.csv", index=False)
```

Jupyter Notebook Summary

In this project, I used a Jupyter Notebook to connect with the MySQL database, run SQL queries on the spotify_tracks table, and explore the results. The notebook contains both SQL outputs and small visualizations to understand the popularity and duration of different tracks.

Bullet list

Key steps done in the Jupyter Notebook:

- **Connected to MySQL using %load_ext sql and connection string**
- **Verified data insert using SELECT * FROM spotify_tracks;**
- **Calculated average popularity of all songs**
- **Identified the most popular and least popular tracks**
- **Found longest and shortest duration tracks**
- **Filtered tracks with duration > 5 minutes**
- **Categorised songs using a CASE statement based on popularity**
- **Exported the query results into CSV files**
- **Created summary statistics and visual insights**

Outputs from the Notebook:

Query result showing the most popular track

```
%%sql
/* Most popular song*/
select track_name, artist, album, popularity
from spotify_tracks
order by popularity desc
limit 1;
```

```
 * mysql+mysqlconnector://root:***@localhost/test
1 rows affected.
```

track_name	artist	album	popularity
Thalapathy Kacheri (From "Jana Nayagan")	Anirudh Ravichander	Thalapathy Kacheri (From "Jana Nayagan")	76

Project Results (Key Insights)

After running SQL queries and analyzing the data in Jupyter Notebook, the following key insights were discovered from the 10 Spotify tracks dataset. These insights summarize the popularity, duration patterns, and artist distribution.

1. Most Popular Track

The track with the highest popularity score:

- **Track Name:** *Nalavulae Mahane*
- **Popularity Score:** *76*
- **Artist:** *Hariharan*

This represents the top-performing song in the dataset.

2. Least Popular Track

The track with the lowest popularity score:

- **Track Name:** *Psycho Kanmani*
- **Popularity Score:** *37*

This shows a significant difference between the top and bottom songs in terms of listener engagement.

3. Average Popularity

The average popularity across all 10 tracks is:

- **Average Popularity:** *63.6*

This helps in classifying songs above or below the benchmark.

4. Tracks Above Average Popularity

Total number of tracks that performed above the average popularity:

- **Count:** *6 tracks*

These tracks can be categorized as *hit* or *well-performing*.

5. Longest Track

The longest track in the dataset:

- **Track Name:** *Pirai Thedum*
- **Duration:** *5.54 minutes*

6. Shortest Track

The shortest track:

- **Track Name:** *Ordinary Person (Leo)*
- **Duration:** *2.32 minutes*

Short songs usually target quicker engagement.

7. Tracks Longer Than 5 Minutes

Total tracks over 5 minutes:

- **Count:** *1 track* (Pirai Thedum)

This is useful for playlist duration planning.

8. Number of Unique Artists

The dataset contains:

- **Unique Artists:** *8*

This shows variety in music artists included in the dataset.

9. Popularity Category Classification (CASE Statement)

Based on popularity:

- **Very Popular (≥ 75):** 1 track
- **Popular (60–74):** 6 tracks
- **Less Popular (< 60):** 3 tracks

This gives a clear distribution of overall song performance.

10. Overall Dataset Summary

- **Total Tracks:** 10
- **Data Source:** Spotify Web API
- **Storage:** MySQL Database
- **Queries Executed:** 10+
- **Final Outputs:** CSV, HTML, PDF reports

What I Learned

This project helped me gain practical, hands-on experience across multiple areas of data engineering and analytics.

Through building this end-to-end pipeline, I developed the following key skills

1. Working with Real APIs (Spotify Web API)

- Learned how to authenticate using Client ID & Secret
- Used the Spotipy library to fetch track metadata
- Understood API response structures (JSON format)
- Automated extraction for multiple track URLs

2. Data Extraction Using Python

- Read URLs from a text file
- Used Regex to extract track IDs from Spotify links
- Implemented Python loops to process multiple records
- Converted raw API data into clean structured format

3. Data Transformation

- Converted duration from milliseconds to minutes
- Cleaned metadata fields
- Organized track information into dictionaries and DataFrames
- Prepared data for database insertion

4. Database Knowledge (MySQL)

- Created MySQL database and tables
- Inserted records using parameterized SQL queries
- Connected Python to MySQL using mysql-connector
- Verified results using SQL queries inside Jupyter Notebook

5. SQL Analysis & Insights

- Wrote analytical SQL queries
- Identified most/least popular songs
- Found longest/shortest tracks
- Calculated averages and categories
- Exported query results into CSV files

6. Jupyter Notebook Reporting

- Combined SQL magic commands with Python
- Displayed datasets, insights, and charts

- Exported notebook as HTML and PDF
- Learned to create clean analytical reports

7. Data Visualization

- Created bar charts for popularity and duration
- Understood how to present insights visually
- Used Matplotlib inside notebook

8. Project Structuring & Documentation

- Built a real, professional folder structure
- Wrote a detailed README.md
- Created portfolio documentation

Conclusion

- This project successfully demonstrates an end-to-end data pipeline using Python, Spotify Web API, MySQL, and Jupyter Notebook.
  I was able to extract real-world music metadata, transform it into a clean dataset, load it into a relational database, and run analytical SQL queries to gain insights.
- Through this project, I practiced:
- API integration
- ETL development
- SQL analytics
- Visualization
- Project structuring
- Portfolio documentation

The final outputs—including CSV files, SQL results, visualizations, and a structured PDF report—showcase my ability to work on real-world data engineering and analytics tasks.

This project has given me confidence to build similar pipelines for larger datasets and more complex scenarios.

Contact Details

If you would like to reach me:

- **Name: Abdul Rahman**
- **Email: abdulrahmanrafiq200@gmail.com**
- **GitHub: https://github.com/abdul-data-analyst/Spotify-Track-Analytics**
- **Linkedin: https://www.linkedin.com/in/abdulrahman-m-b32280335**