**Spotify ETL using PySpark and PostgreSQL**

**Table of Contents**

[Abstract 3](#_Toc134532827)

[Introduction 3](#_Toc134532828)

[a. Project Goals & Objectives: 3](#_Toc134532829)

[b. Project Scope: 4](#_Toc134532830)

[c. Project Limitations & Constraints: 4](#_Toc134532831)

[d. Feasibility Study: 4](#_Toc134532832)

[System Requirement Specifications 5](#_Toc134532834)

[a. Hardware Specifications 5](#_Toc134532835)

[b. Software Specifications 5](#_Toc134532836)

[c. Be-Spoke 6](#_Toc134532837)

[System Design 7](#_Toc134532838)

[a. Architecture Diagram 7](#_Toc134532839)

[b. Database Schema 7](#_Toc134532840)

[c. Use-Case Diagram 8](#_Toc134532841)

[d. Sequence Diagram 8](#_Toc134532842)

[Data Design 9](#_Toc134532843)

[a. Data Extraction 9](#_Toc134532844)

[b. Data Transformation 9](#_Toc134532845)

[c. Data Loading 9](#_Toc134532846)

[d. Summary generation 10](#_Toc134532847)

[e. Automation 12](#_Toc134532848)

[Result 12](#_Toc134532849)

### 

Conclusion  [19](#_Toc134532849)

References ……………………………………………………………………………………… [20](#_Toc134532849)

### Abstract

Spotify is a popular music streaming service that provides its users access to millions of songs and podcasts. Being a data-driven business, Spotify collects and analyzes user data to provide personalized suggestions and insights. One of the features that Spotify offers is a periodic summary, which presents users with a playlist of songs based on their listening history for the week. To generate personalized playlists for its users, Spotify must collect and process data using an ETL (Extract, Transform, Load) process. This process involves extracting data from multiple sources, formatting it for analysis, and loading it into a data warehouse. In this project, the ETL process for creating the periodic summary will be implemented using PySpark, PostgreSQL and Celery. PySpark is a Python-based tool that enables distributed data processing on large datasets using the Apache Spark architecture. In this project, we will use PySpark to extract data from the Spotify API and transform it into a format suitable for analysis. Celery is a distributed task queue that allows scheduling and executing tasks asynchronously. It provides a simple way to schedule tasks to be executed in the future or at a specific time using its built-in task scheduling feature. In this project, Celery, along with Redis broker will be used to program and schedule the ETL process, as well as monitor it for any errors and ensure it runs consistently. Spotipy API will be used to extract data, which will then be transformed using PySpark before being loaded into PostgreSQL. The goal of this project is to leverage PySpark and Celery to automate the ETL process and create a dependable system for generating the periodic summary playlist for Spotify users.

### Introduction

This project involves building an ETL (Extract, Transform, Load) process using the Spotify API to extract listening data, transform it using Python and Pyspark, and load it into a Postgresql database. Additionally, a periodic summary email is generated using SQL and Python, which includes metrics like total amount of time used, top artists user listened to, songs that user frequently listened to etc.

#### a. Project Goals & Objectives:

The main goal of this project is to create a system that automatically extracts listening data from Spotify API called Spotipy, transforms it into spark dataframes using python, and loads it into a PostgreSQL database. This database can then be used to generate a summary email with metrics related to the user's listening habits. The objectives of this project are to:

* Successfully extract listening data using Spotipy API.
* Transform the data into a format suitable for analysis.
* Load the transformed data into a Postgresql database.
* Generate a summary email with relevant metrics.
* Automate the entire process using Celery.

#### b. Project Scope:

The scope of this involves extraction of data from the Spotify API, transformation of the data using Python and PySpark, and loading of the data into a Postgresql database. Additionally, a periodic summary email is generated using SQL and Python to show user specific listening metrics and this data can be engineered and analyzed to provide better listening experience to users using machine learning. This can also be extended as a monthly or yearly summary where users can look back on what they listened.

#### c. Project Limitations & Constraints:

The limitations of the project include is that only user specific data can be retrieved using this API. It also stores data in Postgres database, which means the memory is limited and the application is not scalable.

#### d. Feasibility Study:

Based on the available resources and technology, this project is deemed feasible. The Spotify API provides an easy-to-use interface for extracting listening data, and Python and Pandas provide powerful tools for transforming and loading the data into a database. The use of Celery enables the automation of the entire process, which makes it more efficient and reliable.

### System Requirement Specifications

#### Hardware Specifications

* Operating System – Windows 11, 64-bit operating system
* RAM – 8gb

#### Software Specifications

* Software
* Java SE 8 or above
* Python 3.8 or above
* PostgreSQL 15
* pgAdmin 4
* Spark 3.3.2
* Hadoop (winutil.exe)
* Redis 3.0.5
* VS Code
* PostgreSQL JDBC Driver (postgresql-42.6.0.jar)
* Python libraries
* pyspark
* spotipy (Spotify API)
* celery
* psycopg2
* tabulate
* datetime
* tabulate

#### Be-Spoke

* The weekly generated summary by Spotify can be utilized to develop a variety of bespoke applications with use cases both within and outside of Spotify's platform.
* This summary can be leveraged by Spotify to create machine learning models that can provide music recommendations tailored to the individual user.
* In addition, the data can be analyzed to identify music trends based on various factors such as geography, language, and age of the listener.
* To achieve this, individual user listening histories can be extracted, and relevant features can be selected for specific recommendations. ML models can be developed using these features, and individual user data can be inputted to predict songs that the user may enjoy in the future.
* This data also helps Spotify to identify niche content and niche audiences easily.
* Third-party content creators can access this data to determine which type of content is preferred by users based on their geography, age group, language, and other factors.
* This provides valuable insights for content creators to develop more targeted and engaging content.

### 

### System Design

#### Architecture Diagram

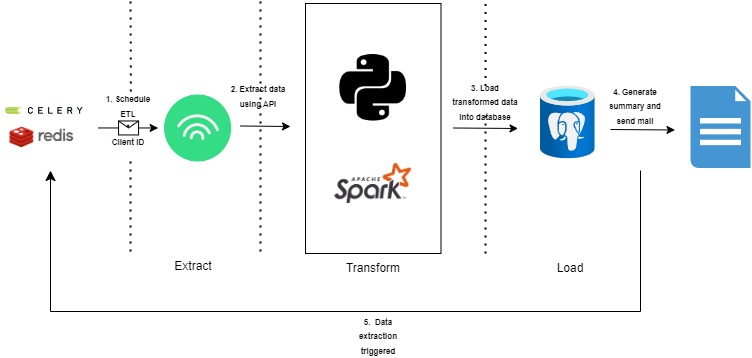
****

Fig 1. Architecture Diagram

#### Database Schema

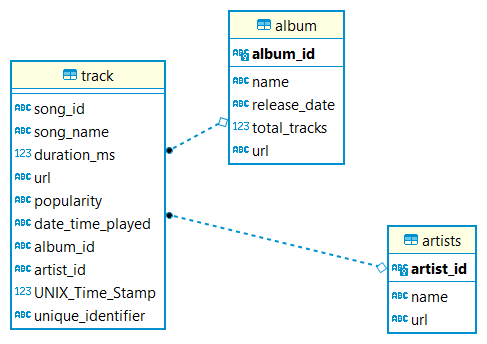


Fig 2. Database schema

#### Use-Case Diagram

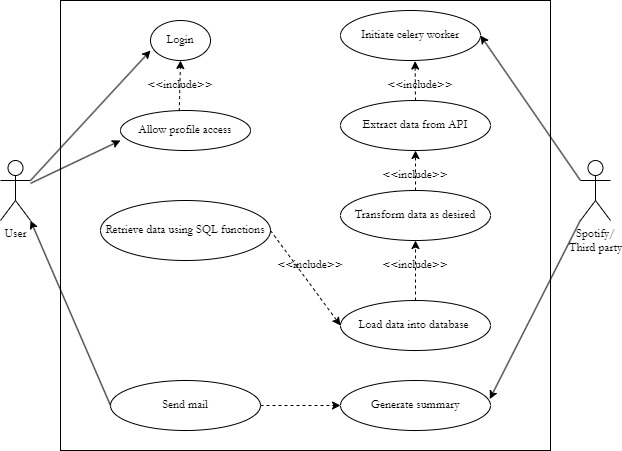


Fig 2. Use-Case Diagram

#### Sequence Diagram

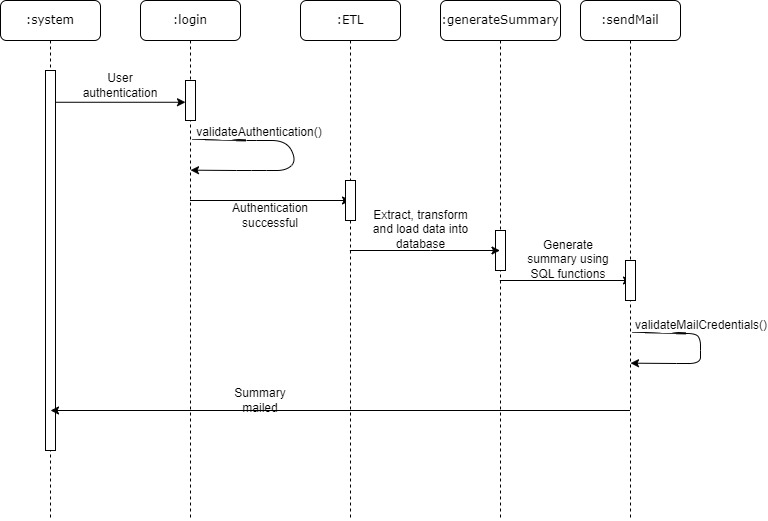


Fig 3. Sequence Diagram

### 

### Data Design

#### Data Extraction

* Spotipy API is used for extracting user specific data from Spotify.
* Spotipy uses oauth2, which expects client ID and client secret ID of user created Spotify developer application.
* Upon providing these, Spotify can be authorized and a json response will be provided.
* This response is further modified into three different dataframes which include data of songs played, respective albums and artists.

#### Data Transformation

* The json response is transformed into python dictionaries for creating spark dataframes out of them.
* Spark dataframes are created in such a way that we can load the dataframe directly into PostgreSQL tables using write() method.
* Album dataframe contains album id, album name and URL of all the albums of songs listened in last one week.
* Artist dataframe contains artist id, artist name and URL of all the artists associated with the songs listened in past week.
* Track dataframe contains details of all the songs listened in the past week along with the exact timestamp when the song was listened.
* In order to prevent two different instances of the same song listened at the same time, an unique identifier variable is added to the Track table.
* The dataframes are further loaded into PostgreSQL tables.

#### Data Loading

* To load the dataframes into database tables, a connection has to be established between the python code and PostgreSQL.
* JDBC driver for PostgreSQL is downloaded and configured within python script such that spark dataframes can be directly loaded into tables.
* Three tables are created inside a schema as shown in Fig 2.
* To achieve this, spark requires
  + URL – *jdbc:postgresql://{databaseServer} /{databaseName}*
  + Table name
  + Username – Database username
  + Password – Database password
  + Mode – Specify how the data has to be added into the table.

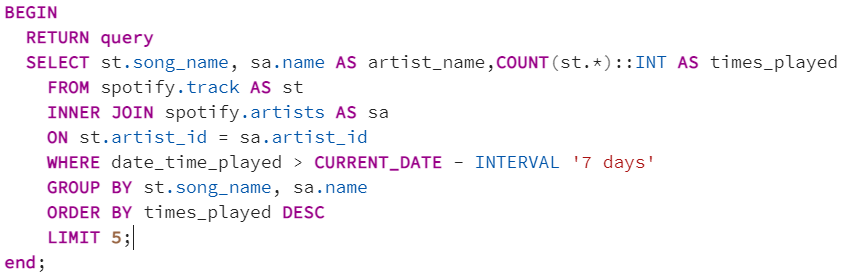
#### Summary generation

* Summary consists of the following insights from the listening history of the user.
  + Most listened tracks.
  + Most listened to artists.
  + Tracks that a user spent most time on.
* To achieve these results, SQL functions are used and called upon.
  + Function to calculate total time spent listening to Spotify.

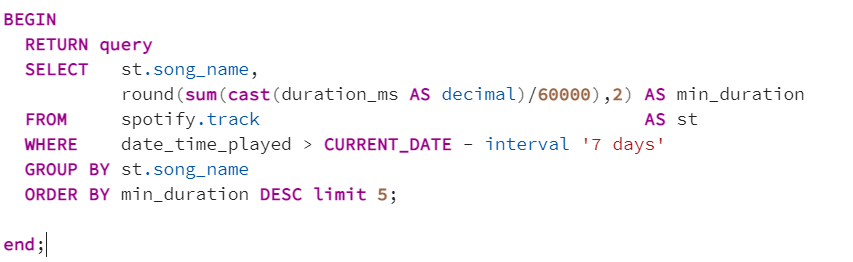
A picture containing text, screenshot, font

Description automatically generated

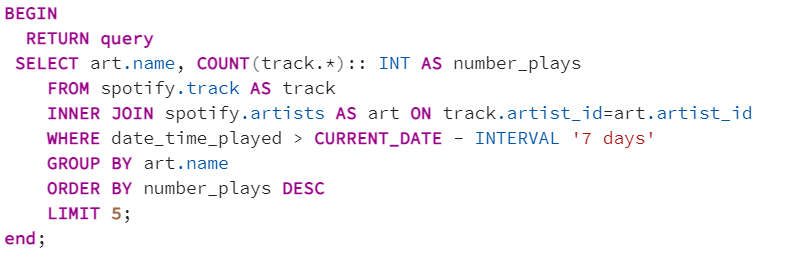
* + Function to find most listened to songs and respective artists.



* + Function to find top songs based on time spent.



* + Function to find top artists listened.



* Post connecting to the database, desired summarization is achieved using these functions.
* To send mail, python’s basic smtplib is used.
* The body of the mail is written in HTML format and upon specifying sender and recipient mail address, mail will be sent accordingly.

#### Automation

* Since the ETL process has been designed, it has to be automated to complete the pipeline.
* In order to automate the pipeline, Celery, an asynchronous queue is employed along with a broker Redis server.
* Two celery tasks are initiated to perform the tasks –
  + ETL process task, where data is extracted, transformed, and loaded into PostgreSQL database for generating summary.
  + Summary generation task, where loaded data is manipulated to generate a report and send it periodically.

### Result

To extract outputs, Daily summary has been scheduled, instead of Weekly summary, where user will get summary of last day’s listening. The period can be customized.

Celery run logs on **2023-05-07**:

2023-05-07 10:00:17,059 - tasks - INFO - User data successfully retrieved

2023-05-07 10:10:00,054 - tasks - INFO - Successfully connected to postgres database

2023-05-07 10:10:00,065 - tasks - INFO - Successfully retrieved top 5 songs listened wrt to minutes played

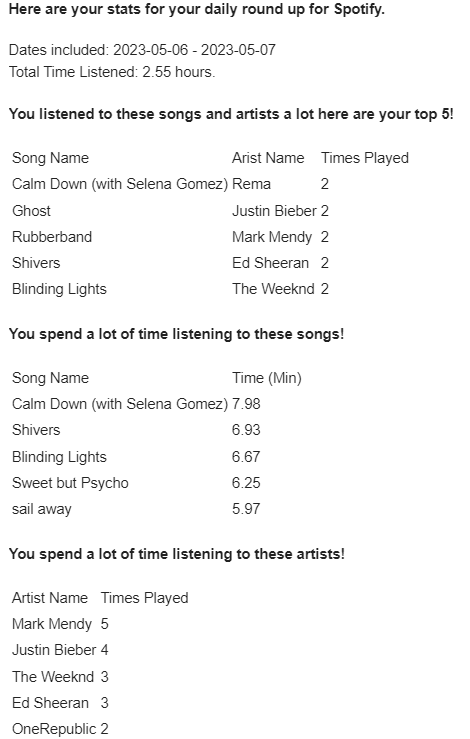
2023-05-07 10:10:00,066 - tasks - INFO - Successfully retrieved total time spent

2023-05-07 10:10:00,068 - tasks - INFO - Successfully retrieved top 5 songs and artists

2023-05-07 10:10:00,071 - tasks - INFO - Successfully retrieved top artists

2023-05-07 10:10:01,305 - tasks - INFO - Successfully sent the mail

Sample mail as shown below will be sent as a summary of user’s Spotify listening history.



Celery run logs on **2023-05-08**:

2023-05-08 10:00:01,073 - tasks - INFO - User data successfully retrieved

2023-05-08 10:10:00,061 - tasks - INFO - Successfully connected to postgres database

2023-05-08 10:10:00,092 - tasks - INFO - Successfully retrieved top 5 songs listened wrt to minutes played

2023-05-08 10:10:00,092 - tasks - INFO - Successfully retrieved total time spent

2023-05-08 10:10:00,092 - tasks - INFO - Successfully retrieved top 5 songs and artists

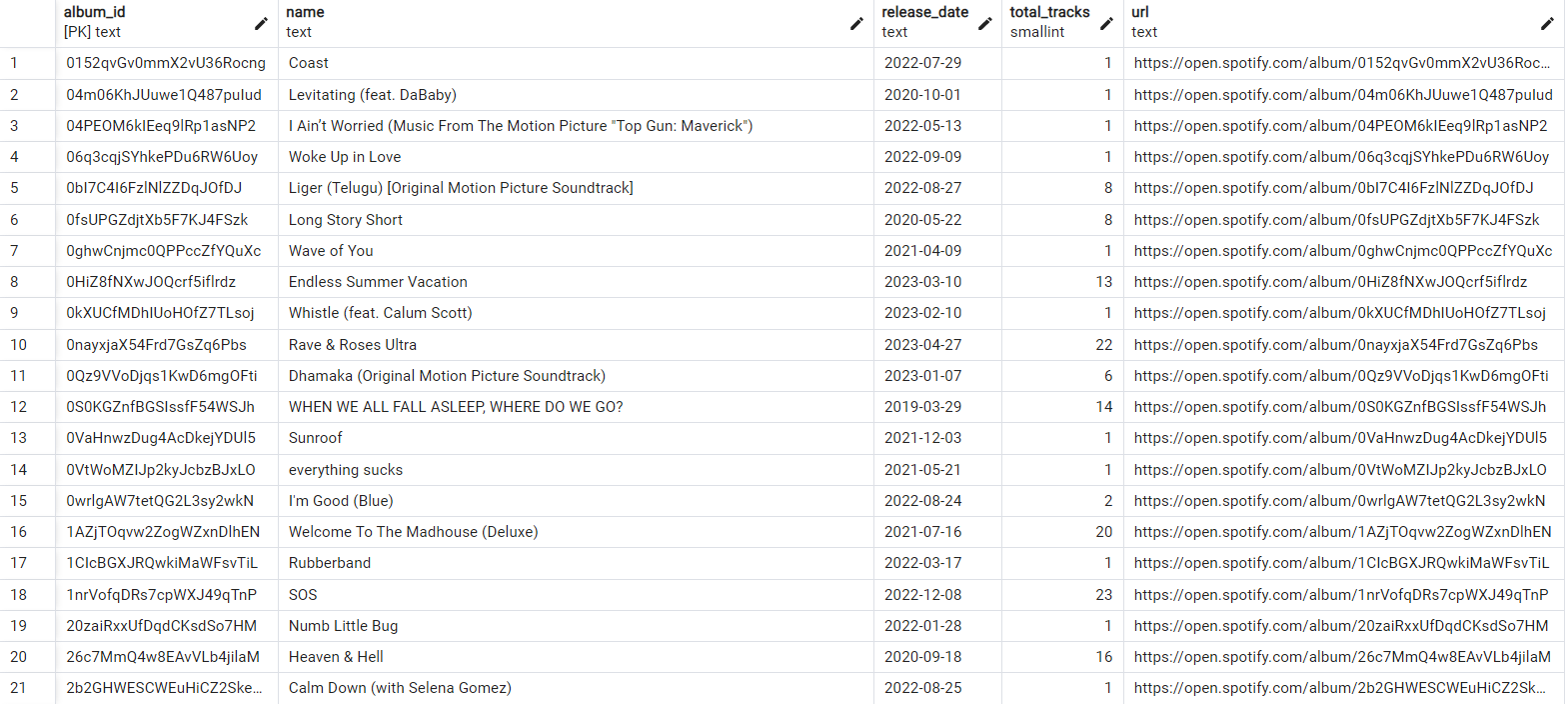
2023-05-08 10:10:00,092 - tasks - INFO - Successfully retrieved top artists

2023-05-08 10:10:01,559 - tasks - INFO - Successfully sent the mail

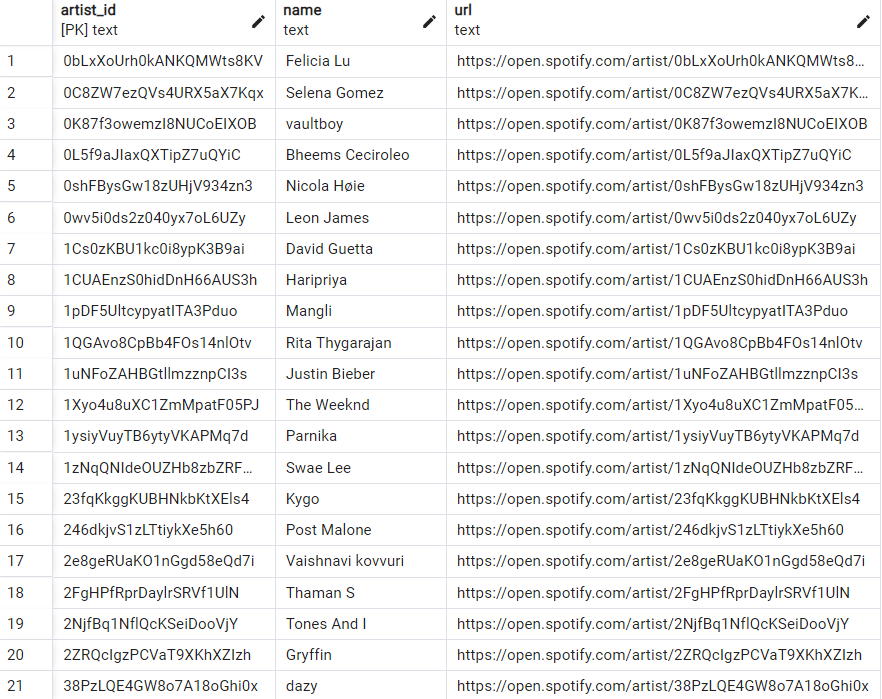
Sample mail as shown below will be sent as a summary of user’s Spotify listening history.



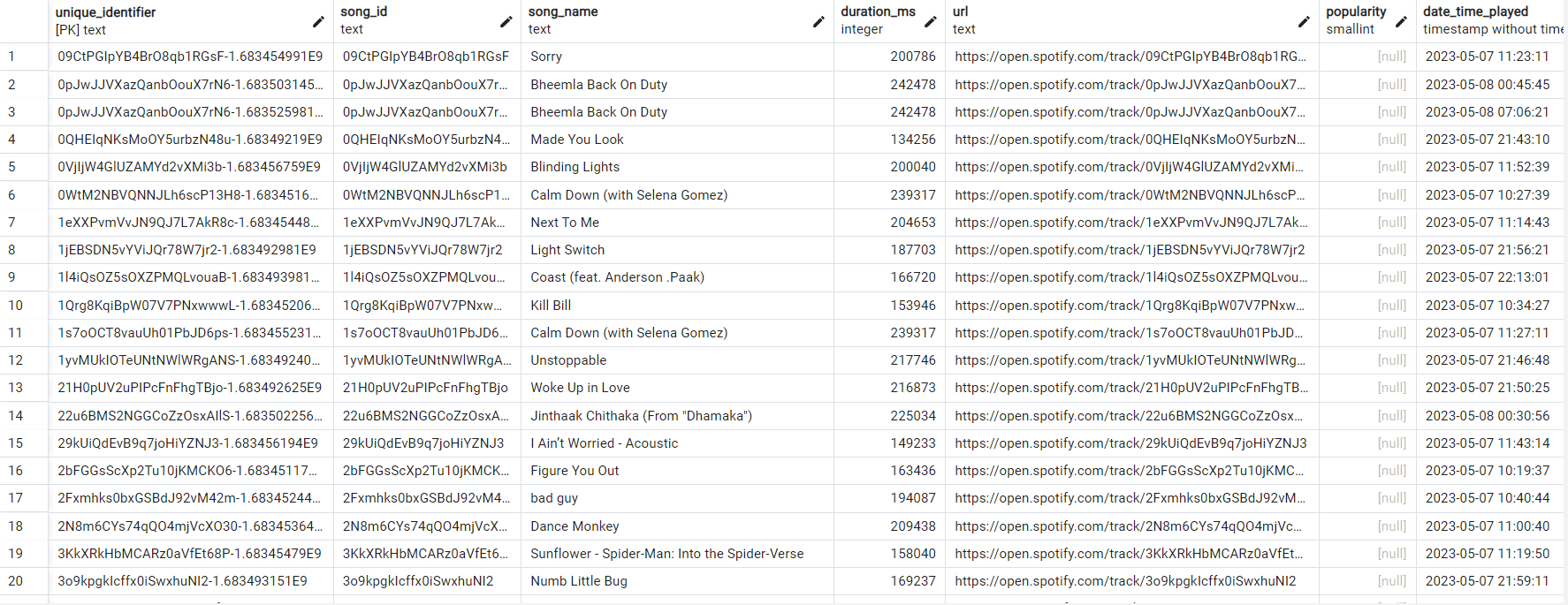
**Album table after run 2:**



**Artist table after run 2:**



**Track table after run 2:**



Celery run logs on **2023-05-09**:

2023-05-08 10:00:01,073 - tasks - INFO - User data successfully retrieved

2023-05-08 10:10:00,061 - tasks - INFO - Successfully connected to postgres database

2023-05-08 10:10:00,092 - tasks - INFO - Successfully retrieved top 5 songs listened wrt to minutes played

2023-05-08 10:10:00,092 - tasks - INFO - Successfully retrieved total time spent

2023-05-08 10:10:00,092 - tasks - INFO - Successfully retrieved top 5 songs and artists

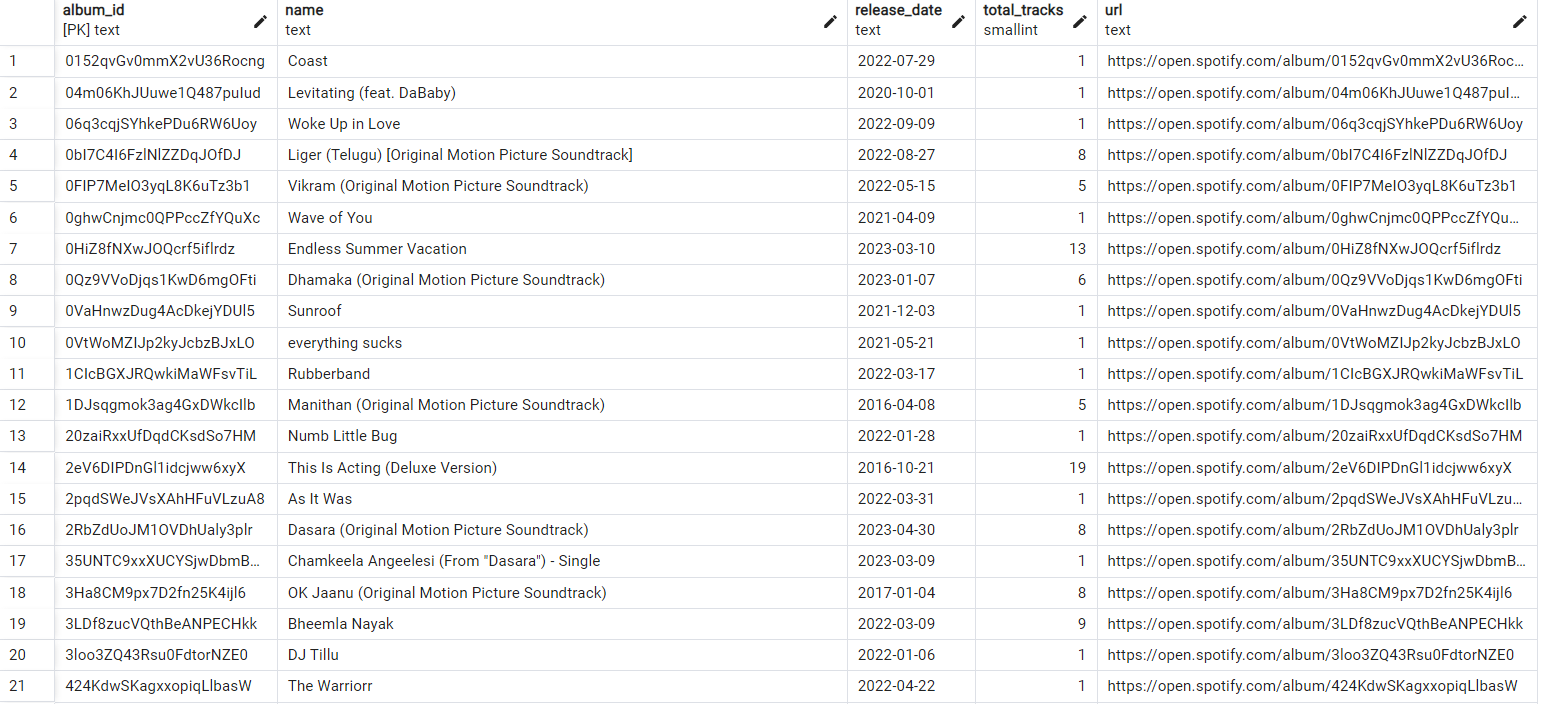
2023-05-08 10:10:00,092 - tasks - INFO - Successfully retrieved top artists

2023-05-08 10:10:01,559 - tasks - INFO - Successfully sent the mail

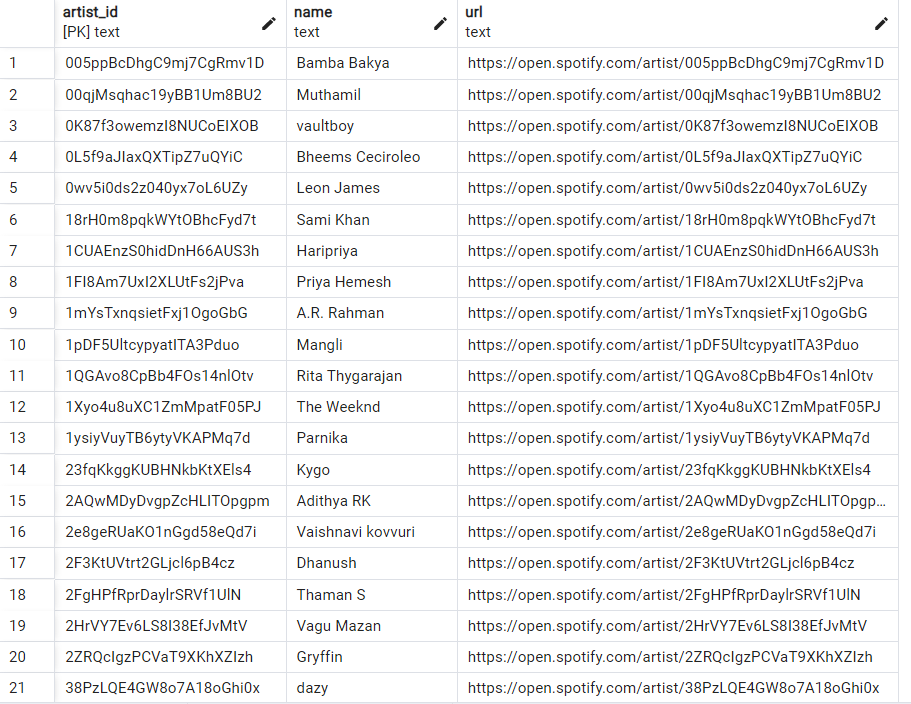
Sample mail as shown below will be sent as a summary of user’s Spotify listening history.



**Album table after run 3:**



**Artist table after run 3:**



**Track table after run 3:**



**Conclusion:**

Extracting insights from the spotify data after performing ETL can enable the creation of customized weekly summary emails for users this process involves extracting data from spotifys apis or database shaping it for analysis and loading it into a data warehouse or data lake user listening history top tracks top artists and user-generated playlists are examples of the data that can be extracted analytical techniques can then be applied to the transformed data to create personalized summaries for users showcasing their listening trends and preferences new releases recommended playlists and upcoming events in their location therefore spotifys ETL process can deliver valuable insights that can enhance users engagement and knowledge about their music listening habits.

**References**

[https://spotipy.readthedocs.io/en/latest/#](https://spotipy.readthedocs.io/en/latest/)

<https://docs.celeryq.dev/en/stable/>

Sidharth Ramalingam Data Engineer | Data enthusiast | Entrepreneurship | Financial Markets. <https://medium.com/dev-genius/data-engineering-project-2-building-spotify-etl-using-python-and-airflow-432dd8e4ffa3>