**Spotify Data Analysis using Postgre SQL**

**Exploratory Data Analysis**

1. Count the total number of rows

**SELECT COUNT(\*) FROM spotify;**

A screenshot of a computer

AI-generated content may be incorrect.

1. Count the total number of unique artists

**SELECT COUNT(DISTINCT artist) FROM spotify;**

A screenshot of a computer

AI-generated content may be incorrect.

1. Count the total number of unique albums

**SELECT COUNT(DISTINCT album) FROM spotify;**

A screenshot of a computer

AI-generated content may be incorrect.

1. What are different album types?

**SELECT DISTINCT album\_type FROM spotify;**

A screenshot of a computer

AI-generated content may be incorrect.

1. Max duration of a song

**SELECT MAX(duration\_min) FROM spotify;**

A grey and black text

AI-generated content may be incorrect.

1. Min duration of a song

**SELECT MIN(duration\_min) FROM spotify;**

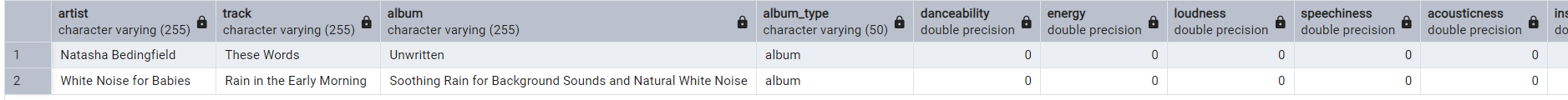
A screenshot of a computer

AI-generated content may be incorrect.

1. Min duration = 0 in Query 6 seems impossible, so find the tracks that have duration 0

**SELECT \* FROM spotify**

**WHERE duration\_min = 0;**



1. Delete the rows that have min\_duration = 0 because that is dirty data and then reprint the list to check if the tracks are actually deleted

**DELETE FROM spotify**

**WHERE duration\_min = 0;**

**SELECT \* FROM spotify**

**WHERE duration\_min = 0;**

A close up of a box

AI-generated content may be incorrect.

1. Check the types of channels

A screenshot of a computer

AI-generated content may be incorrect.

**Data Analysis**

1. Retrieve the names of all tracks that have more than 1 billion streams

**SELECT \* FROM spotify**

**WHERE stream > 1000000000;**

A screenshot of a computer

AI-generated content may be incorrect.

1. List all the albums along with their respective artists

**SELECT DISTINCT album, artist FROM spotify;**

A screenshot of a computer

AI-generated content may be incorrect.

1. Get the total number of comments for the licensed tracks

**SELECT SUM(comments) AS total\_comments FROM spotify**

**WHERE licensed = 'true';**

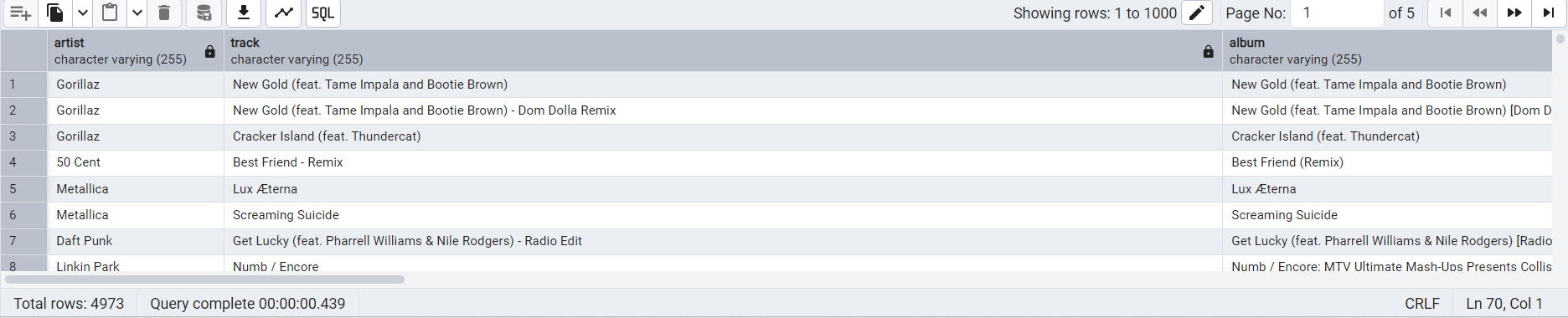
A screenshot of a computer

AI-generated content may be incorrect.

1. Find all the tracks that belong to album type 'single'

**SELECT \* FROM spotify**

**WHERE album\_type = 'single';**



1. Count the total number of tracks by each artist

**SELECT artist, COUNT(track) FROM spotify**

**GROUP BY artist;**

A screenshot of a computer

AI-generated content may be incorrect.

1. Calculate the average danceability of tracks in each album

**SELECT album, AVG(danceability) AS avg\_danceability**

**FROM spotify**

**GROUP BY album;**

A screenshot of a music album

AI-generated content may be incorrect.

1. Find the top 5 tracks with the highest energy values

**SELECT DISTINCT track, MAX(energy)**

**FROM spotify**

**GROUP BY track**

**ORDER BY 2 DESC**

**LIMIT 5;**

A screenshot of a computer

AI-generated content may be incorrect.

1. List all the tracks along with their views and likes where official\_video = TRUE

**SELECT track, SUM(views) AS total\_views, SUM(likes) AS total\_likes**

**FROM spotify**

**WHERE official\_video = 'true'**

**GROUP BY track;**

A screenshot of a computer

AI-generated content may be incorrect.

1. For each album calculate the total views of all associated tracks

**SELECT album, track, SUM(views) AS total\_views**

**FROM spotify**

**GROUP BY album, track**

**ORDER BY 3 DESC;**

A screenshot of a computer

AI-generated content may be incorrect.

1. Retrieve the track names that have been streamed more on Spotify than Youtube

**SELECT \* FROM**

**(SELECT track,**

**COALESCE(SUM(CASE WHEN most\_played\_on = 'Youtube' THEN stream END),0) AS streamed\_on\_youtube,**

**COALESCE(SUM(CASE WHEN most\_played\_on = 'Spotify' THEN stream END),0) AS streamed\_on\_spotify**

**FROM spotify**

**GROUP BY track)**

**AS t1**

**WHERE streamed\_on\_spotify > streamed\_on\_youtube**

**AND streamed\_on\_youtube <> 0;**

A screenshot of a computer

AI-generated content may be incorrect.

1. Find the top 3 most viewed tracks for each artist using window functions

**WITH ranking\_artist AS** -- CTE named as ranking\_artist

**(SELECT artist, track, SUM(views) as total\_views**, -- select the columns

**DENSE\_RANK() OVER (PARTITION BY artist ORDER BY SUM(views) DESC) AS rank** -- use DENSE\_RANK() to assign ranks to views

**FROM spotify**

**GROUP BY artist, track**

**ORDER BY artist, total\_views DESC)**

**SELECT \* FROM ranking\_artist WHERE rank <= 3;** -- SELECT from the CTE

A white and grey striped background

AI-generated content may be incorrect.

1. Write a query to find tracks where the liveness score is above average

**SELECT track, liveness FROM spotify WHERE liveness > (SELECT AVG(liveness) AS avg\_liveness FROM spotify)**

**ORDER BY liveness DESC;**

A screenshot of a computer

AI-generated content may be incorrect.

1. Use a WITH clause to calculate the difference between the highest and the lowest energy values for tracks in each album

**WITH minmaxenergy**

**AS**

**(SELECT album, MAX(energy) AS highest\_energy, MIN(energy) AS lowest\_energy**

**FROM spotify**

**GROUP BY album)**

**SELECT album, highest\_energy - lowest\_energy AS energy\_diff**

**FROM minmaxenergy**

**ORDER BY energy\_diff DESC;**

**A screenshot of a computer

AI-generated content may be incorrect.**