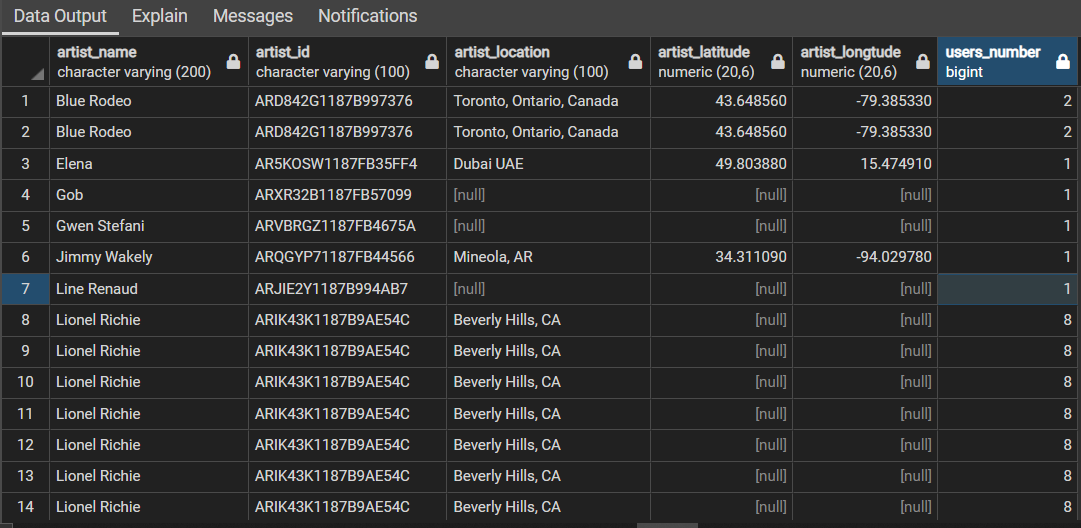
***Analytical SQL Case Study Project  
-----------------------------------------------------------------***

***Datasets Used:   
------------------------------------***

1. Songs Dataset
2. Events Dataset

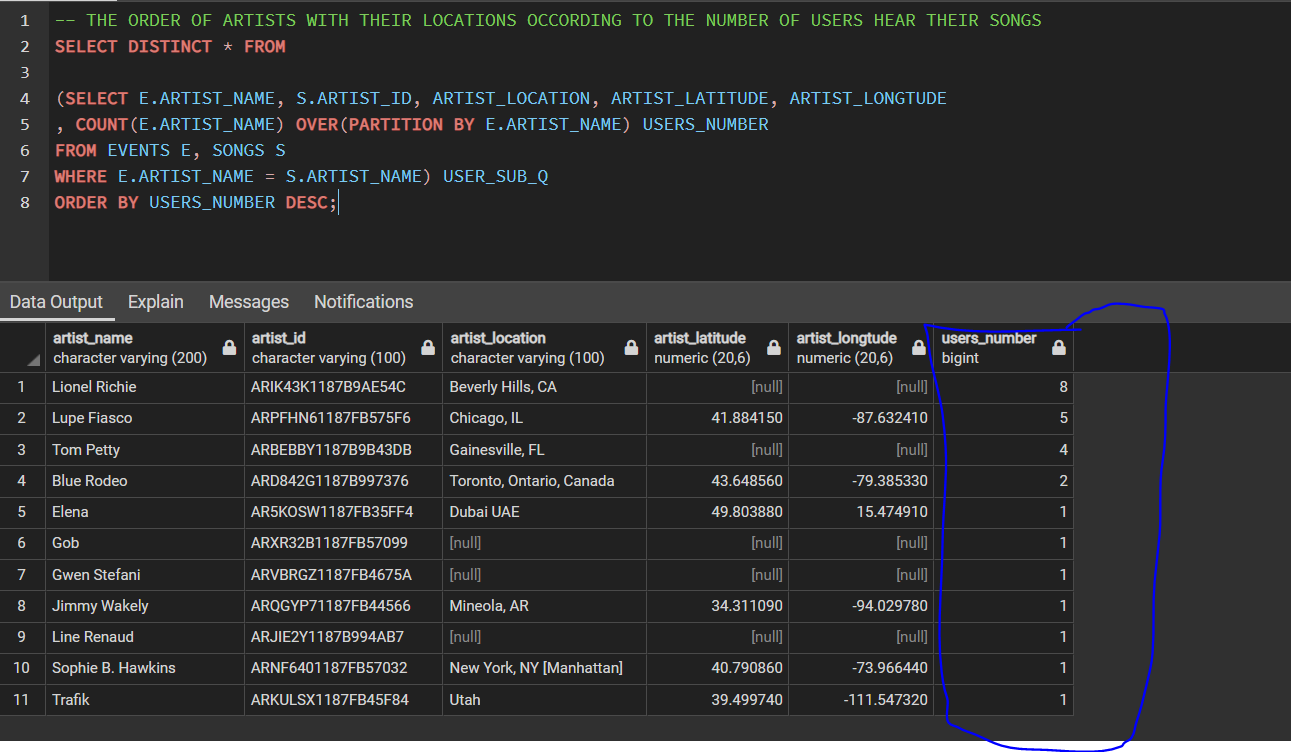
***Query 1:  
------------------***

We want to get the information of the artists from the two table to know the most famous one according to the number of users hear their songs and this will help us to improve the business income by adding additional songs for those artists   
  
first we get the number of users hear each artist by this:  
  
Text

Description automatically generated  
  


Now we want to order them, and git rid of duplicates by this:   
Text

Description automatically generated



***Query 2:  
------------------***

For improvement also we want to know the most famous songs and some other information by joining the two tables so we can attract more users and offer some promotions on these songs

We get the information we want by joining the two tables from songs and events and then order them according to the number of users who here them A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

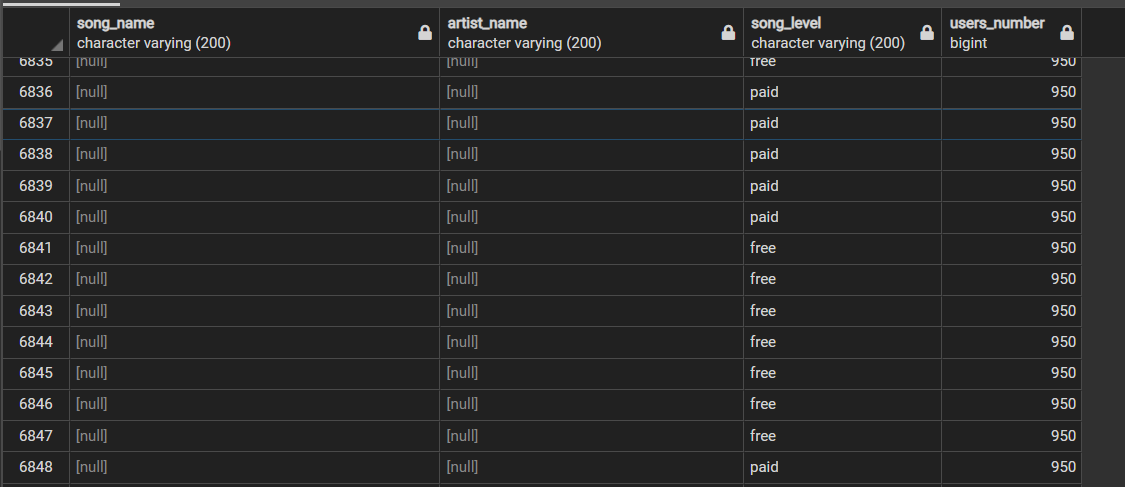
***Query 3:  
------------------***

We want to know now the most played songs in all sessions in events table so we can get some offers and promotions to users and some information of each song

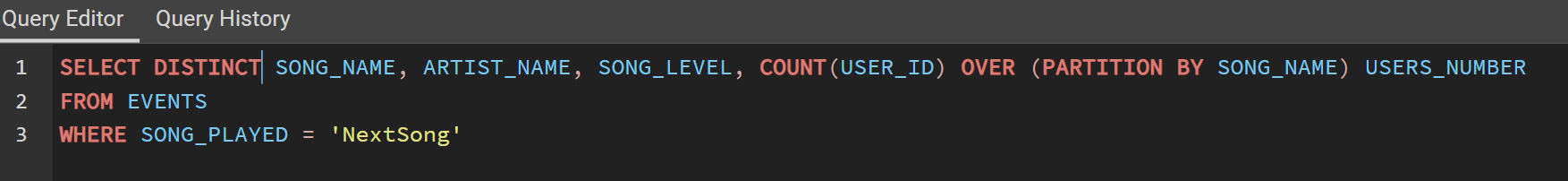
First, we get the information, and the count of users heard this song

Text

Description automatically generated



But there is a lot of nulls, so we add distinct



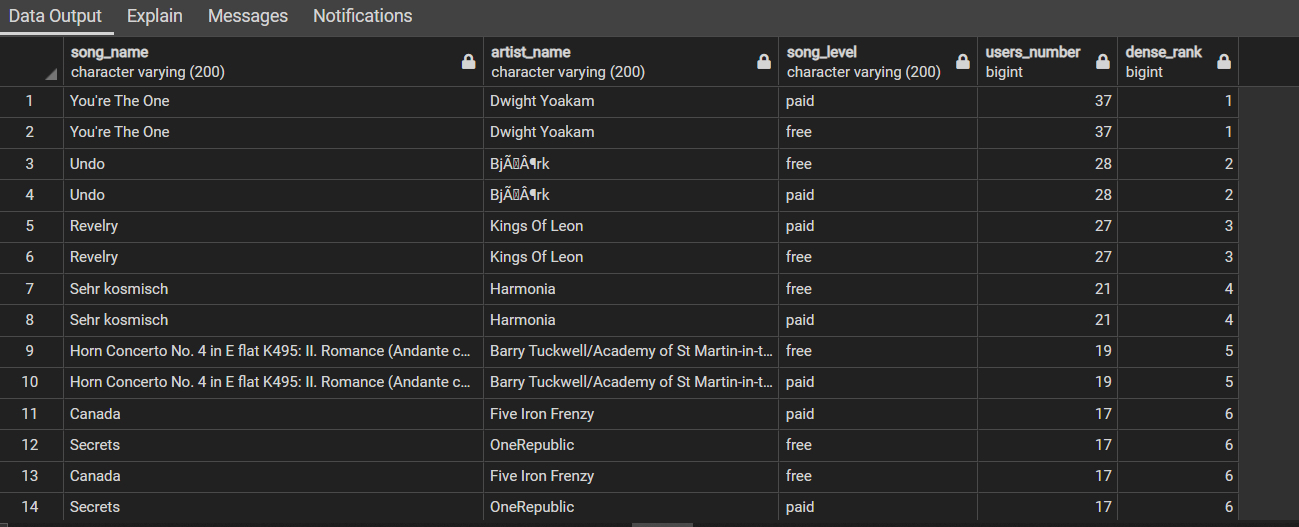
A screenshot of a computer

Description automatically generated with medium confidence

After this we get the rank of each song with dense rank and subquery

Text

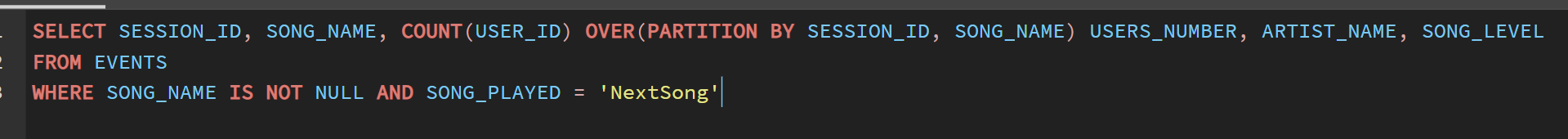
Description automatically generated

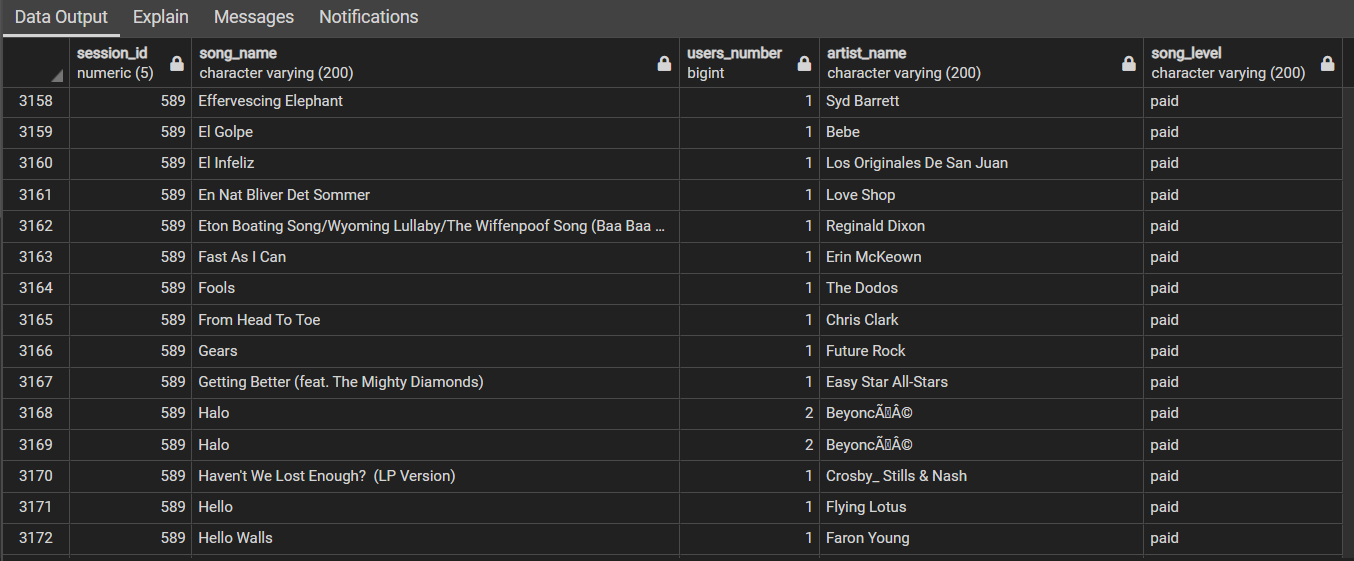


***Query 4:  
------------------***

We want to know the most famous song with the session distribution among users so we can improve the number of songs that is similar and attract more users

First, we get the number of users, session id and some information about each song in this session

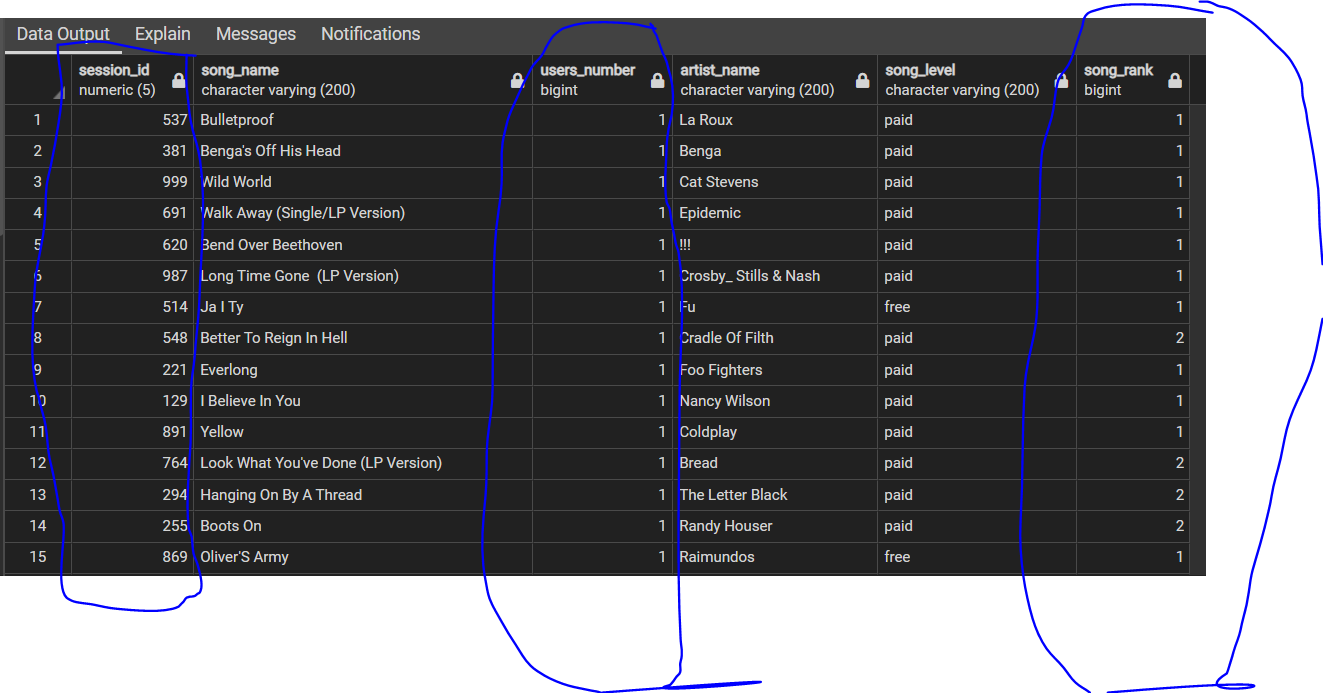




Second step, get the rank of each song in each partition but with distinct to prevent the duplication as the user may hear the same song in the same session so we count it but not to show it again

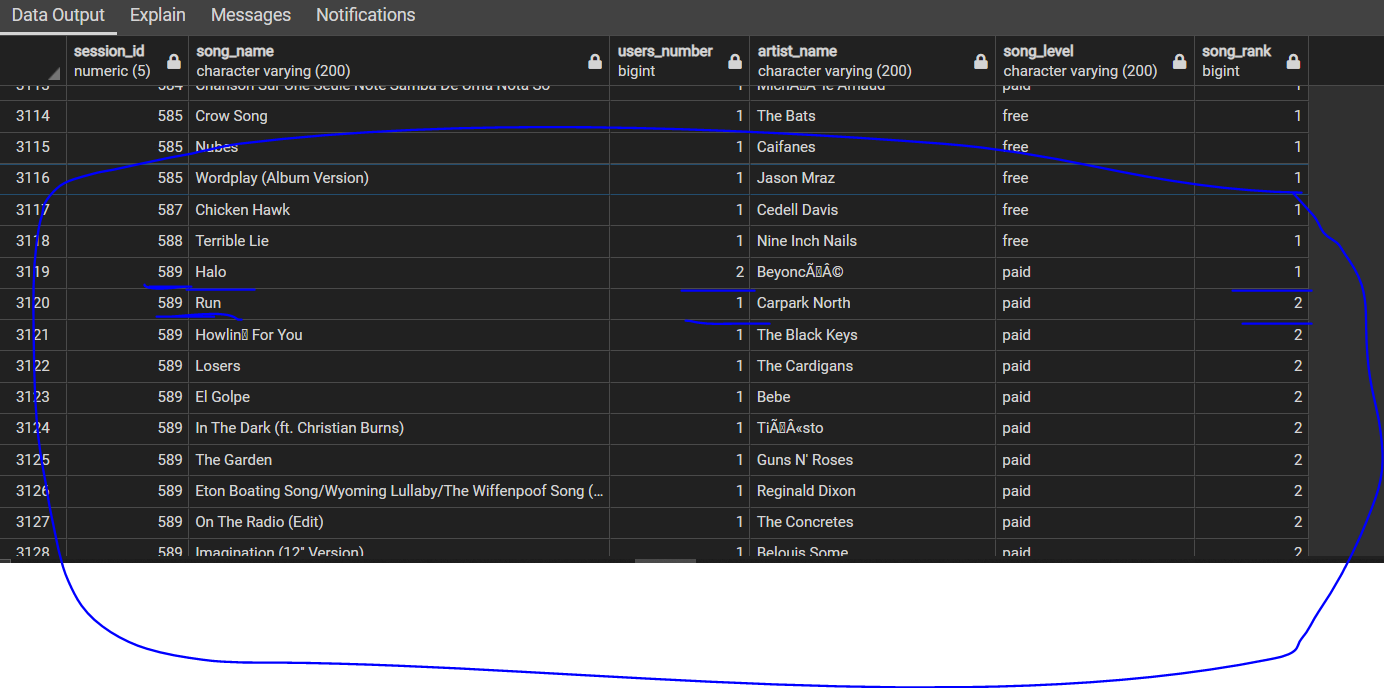
Text

Description automatically generated



Last thing, to group all those information by all selected columns and order them by session number and the rank number (rank number 1 has the maximum number of users) A screenshot of a computer

Description automatically generated with medium confidence



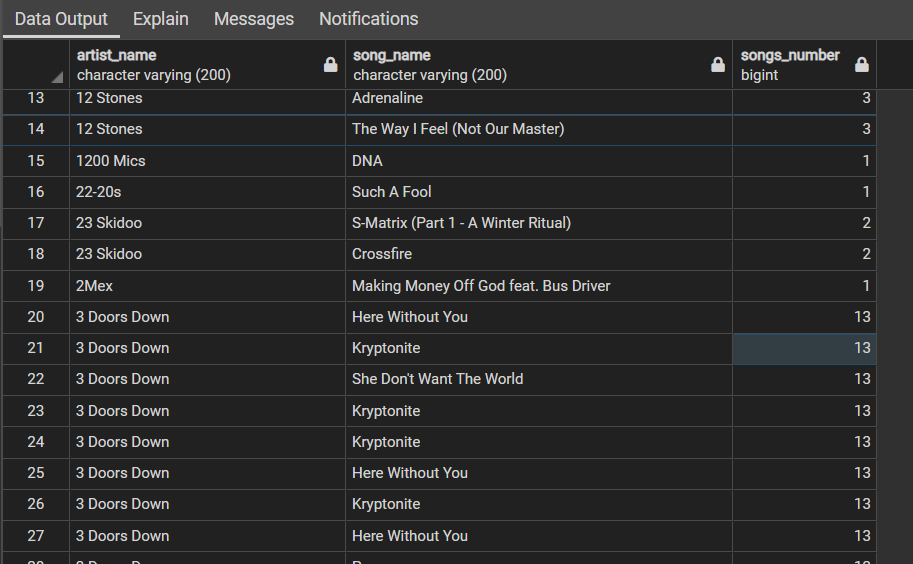
***Query 5:  
------------------***

Now we want to know the order of the famous artist in all sessions so we can add more of their songs and get more profit from users

First, we get the artist information and count of songs they made (distinct to prevent the duplication)

Text

Description automatically generated



Second step, to give each artist a rank according to the number of users and sort them in descending order

Text

Description automatically generated



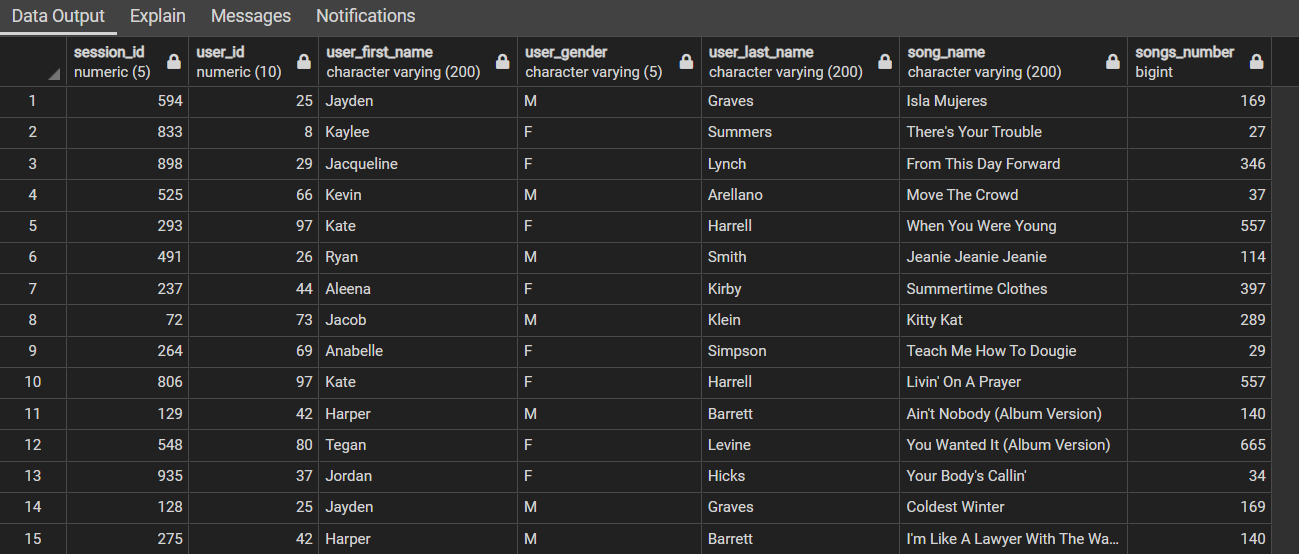
***Query 6:  
------------------***

We want to get the most contributed users in the system so we could make them more comfortable and give them additional offers which will attract other users to our web site

First step, get the users and the number of songs they heard all over the sessions

A screenshot of a computer screen

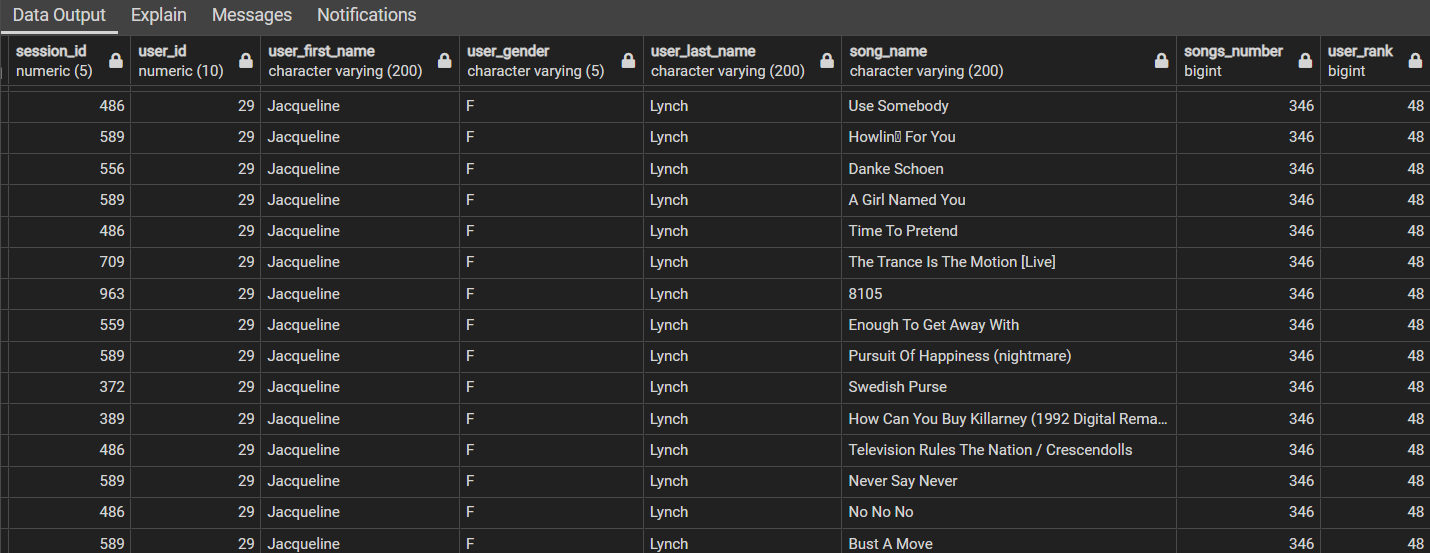
Description automatically generated with medium confidence



Second step, give each user a rank and order them in descending order according to the songs they hear

Text

Description automatically generated



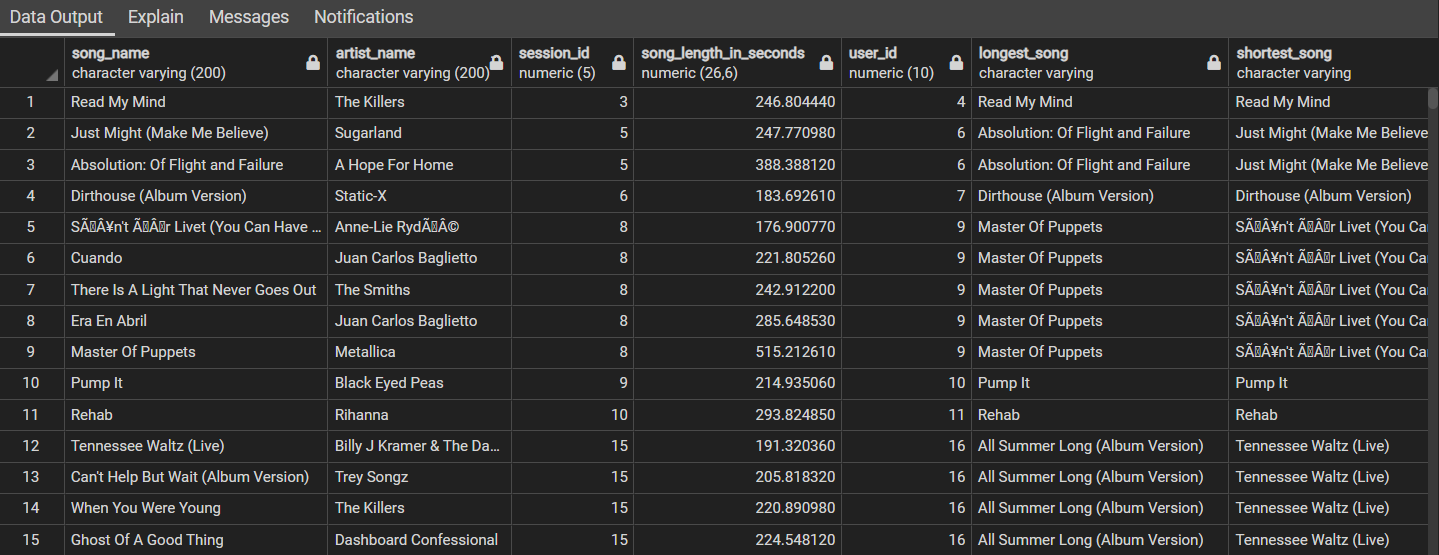
***Query 7:  
------------------***

We want to know the longest and shortest song the users heard in each session so we can put average for songs to not get the users feel boring.

We use here the first value function with desc for the windowing issues as to compare for each session the length of all songs.

Graphical user interface, text

Description automatically generated



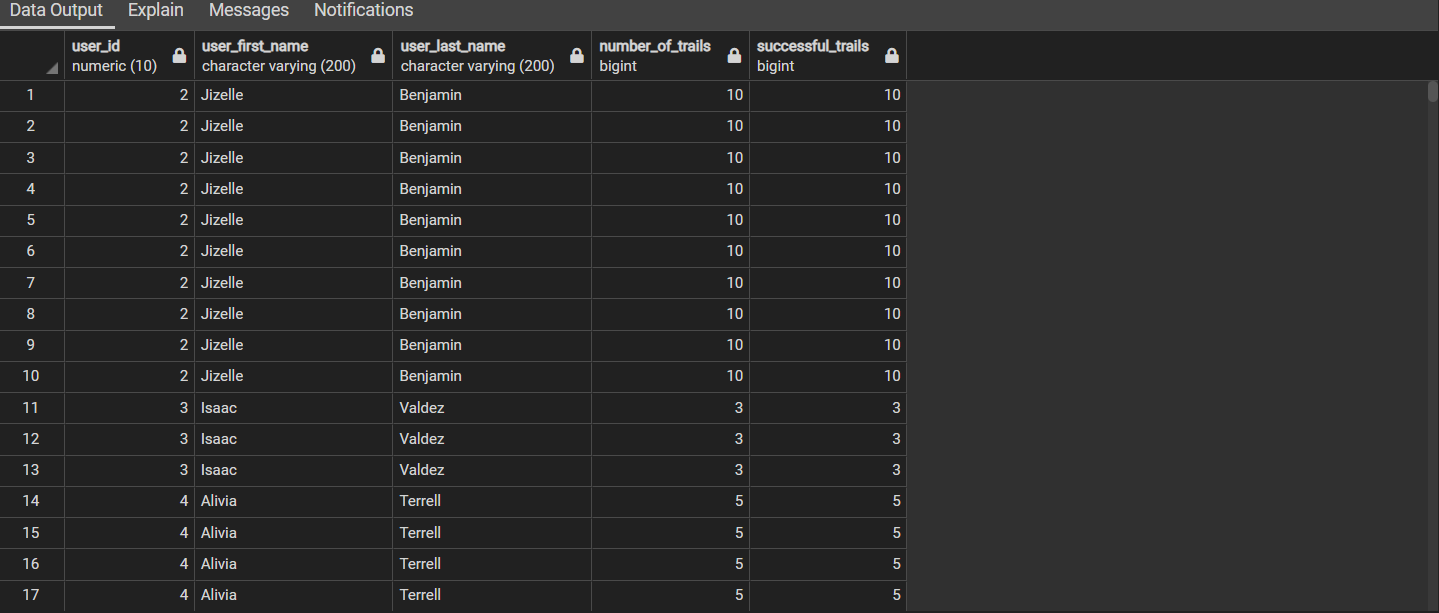
***Query 8:  
------------------***

Sometimes there can be issue in the connection from the user to our server so we need to analysis the number of successful trails the users get to the website.

First step, get the number of trails for each user and the number of successful trails.

Text

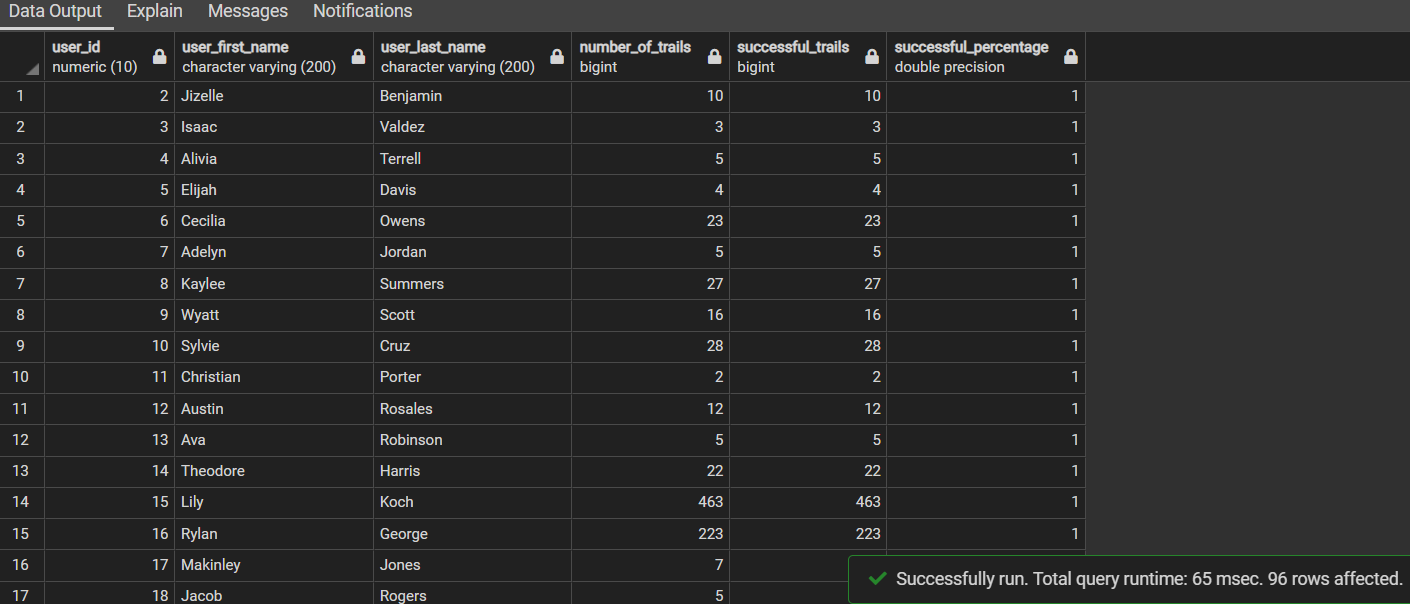
Description automatically generated



Second step is to calculate the percentage of successful connection to the website by successful trails by the number of trails.  
The result will be between [0, 1].

Text

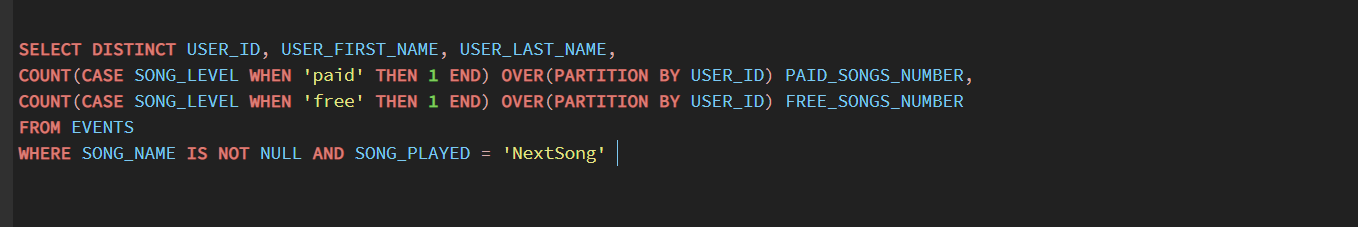
Description automatically generated

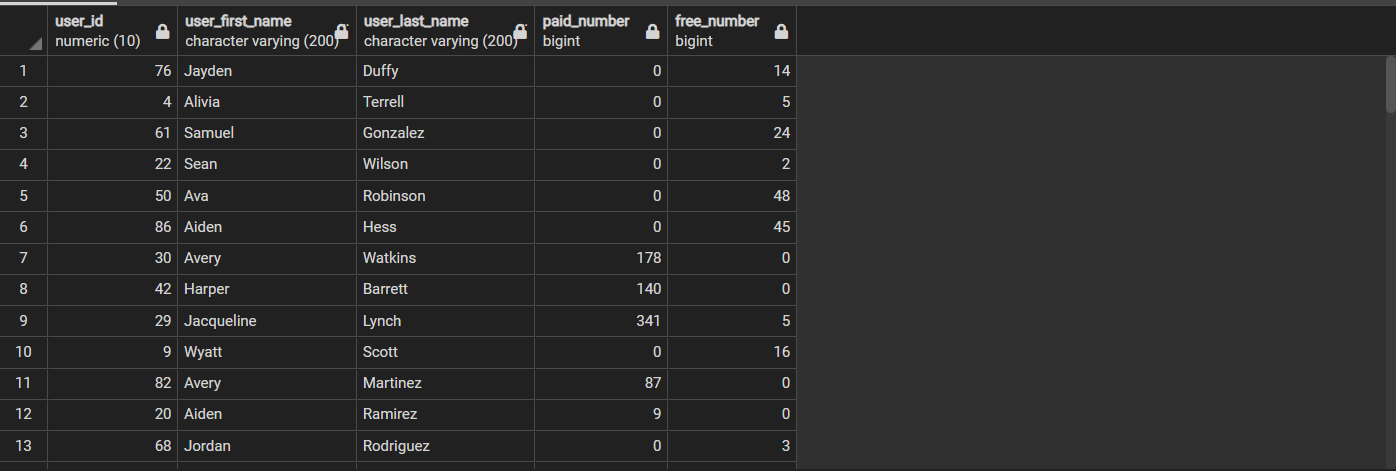


***Query 9:  
------------------***

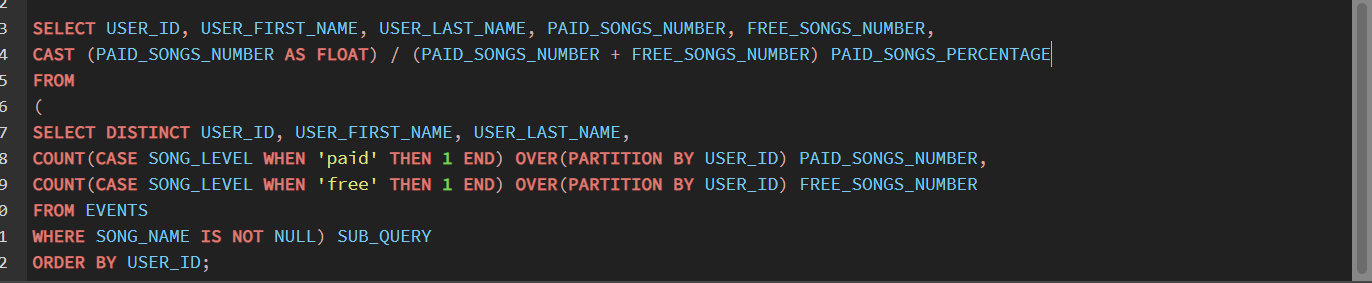
We want now to analysis the income channel as we can know the income from each user from the level of the song paid or free so we will get the number of paid and free songs for each user with the percentage of paid songs which will bet between [0, 1]

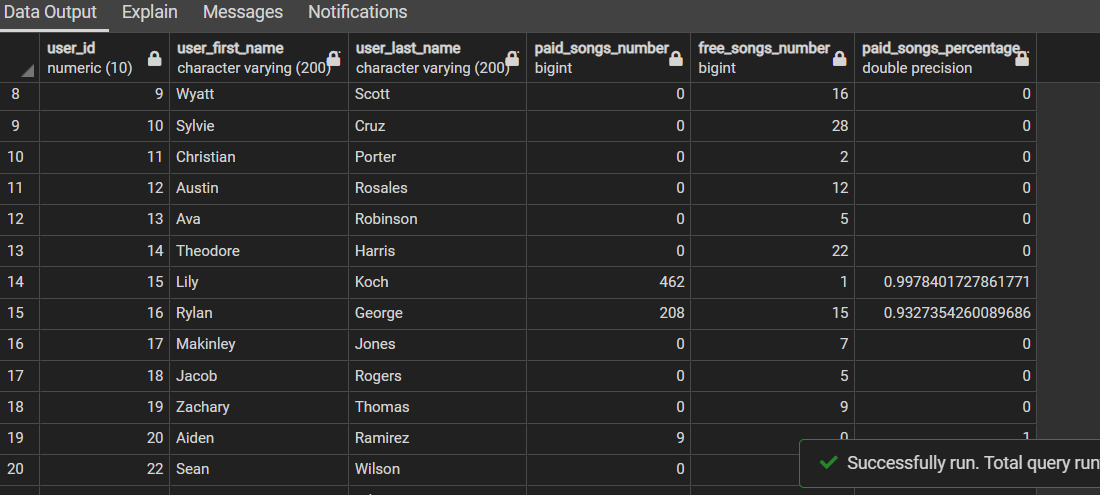
First step, get the information of each user, count of paid songs and count of free songs





Second step calculate the percentage of the paid songs from the total and then order by the user id

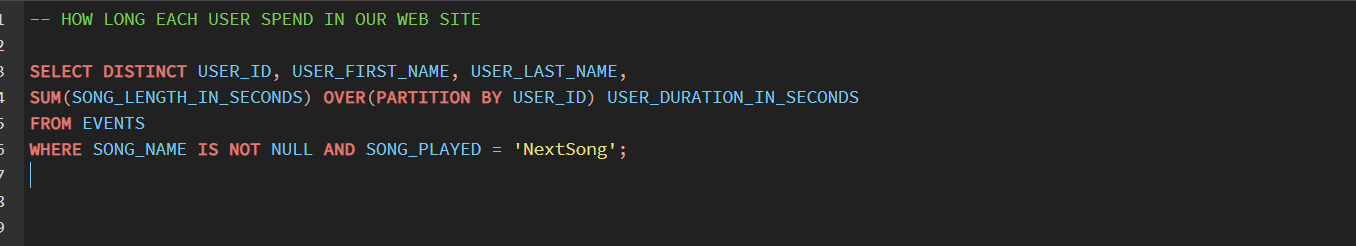


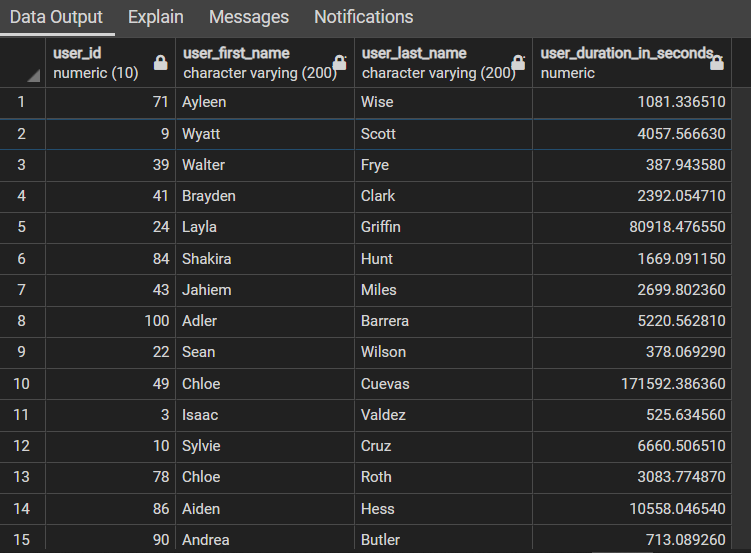


***Query 10:  
------------------***

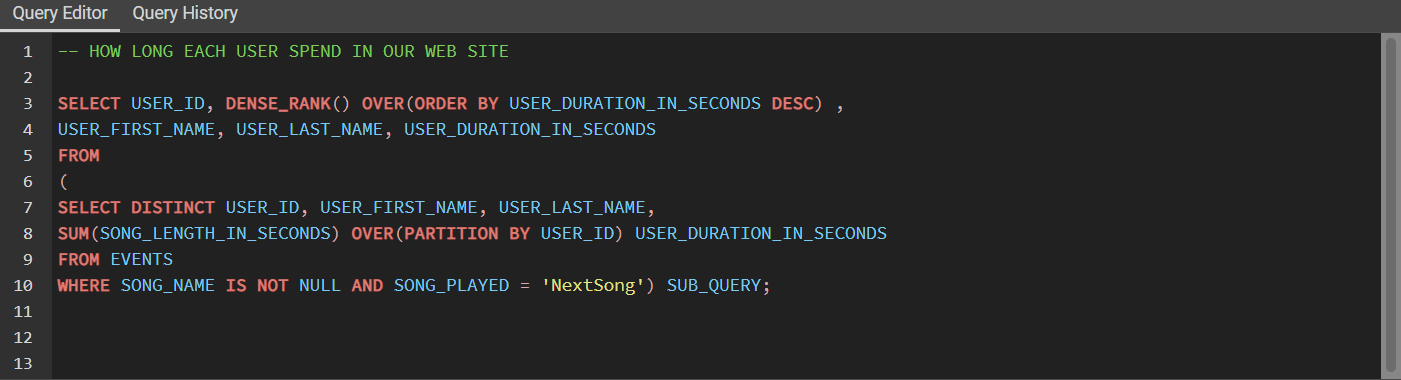
We need to know how long each user spent in the system website listening to songs and order the user to know the most one spent time to improve the website if some users don’t spend too much time.

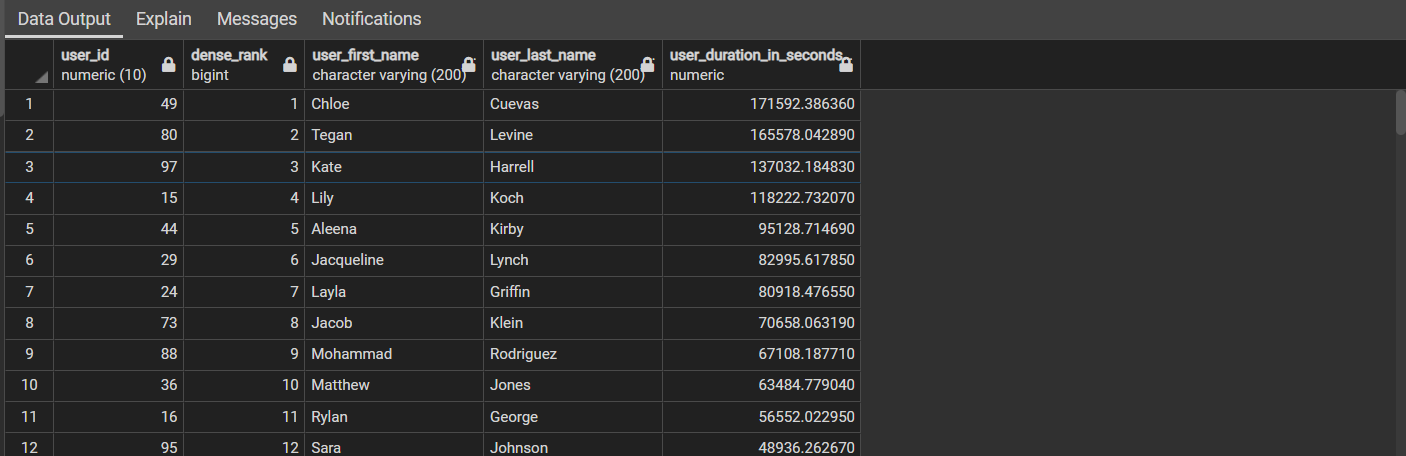
First get all users information and the summation of each song the user played in seconds (distinct to select the unique users without duplication)





Second step, to order the users and give them rank according to the duration in seconds descending order.





***Thank You   
--------------------------------------------------***