

Analytical SQL Case Study Project

Datasets Used:

- 1) Songs Dataset
- 2) Events Dataset

Notes:

- 1) In Project Code Folder You Will Find Queries Code and Additional the Following:
 - a. Creation Code of Events and Songs Tables
 - b. Insertion Data Code of Events and Songs Tables
 - c. The Data Resulted from The Join of Two Tables by (Artist Name) Column and Another By (Song Name) Column
 - d. Selection of Data from Songs and Events Tables
- 2) Screen Shots of Each Code Result with Steps of Queries Building

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:

```
SELECT E.ARTIST_NAME, S.ARTIST_ID, ARTIST_LOCATION, ARTIST_LATITUDE, ARTIST_LONGITUDE
, COUNT(E.USER_ID) OVER(PARTITION BY E.ARTIST_NAME) USERS_NUMBER
FROM EVENTS E, SONGS S
WHERE E.ARTIST_NAME = S.ARTIST_NAME
```

Data Output		Explain	Messages	Notifications			
	artist_name character varying (200)	artist_id character varying (100)	artist_location character varying (100)	artist_latitude numeric (20,6)	artist_longitude numeric (20,6)	users_number bigint	
1	Blue Rodeo	ARD842G1187B997376	Toronto, Ontario, Canada	43.648560	-79.385330	2	
2	Blue Rodeo	ARD842G1187B997376	Toronto, Ontario, Canada	43.648560	-79.385330	2	
3	Elena	AR5KOSW1187FB35FF4	Dubai UAE	49.803880	15.474910	1	
4	Gob	ARXR32B1187FB57099	[null]	[null]	[null]	1	
5	Gwen Stefani	ARVBRGZ1187FB4675A	[null]	[null]	[null]	1	
6	Jimmy Wakely	ARQGY71187FB44566	Mineola, AR	34.311090	-94.029780	1	
7	Line Renaud	ARJIE2Y1187B994AB7	[null]	[null]	[null]	1	
8	Lionel Richie	ARIK43K1187B9AE54C	Beverly Hills, CA	[null]	[null]	8	
9	Lionel Richie	ARIK43K1187B9AE54C	Beverly Hills, CA	[null]	[null]	8	
10	Lionel Richie	ARIK43K1187B9AE54C	Beverly Hills, CA	[null]	[null]	8	
11	Lionel Richie	ARIK43K1187B9AE54C	Beverly Hills, CA	[null]	[null]	8	
12	Lionel Richie	ARIK43K1187B9AE54C	Beverly Hills, CA	[null]	[null]	8	
13	Lionel Richie	ARIK43K1187B9AE54C	Beverly Hills, CA	[null]	[null]	8	
14	Lionel Richie	ARIK43K1187B9AE54C	Beverly Hills, CA	[null]	[null]	8	

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

```
1 SELECT DISTINCT * FROM
2 (SELECT E.ARTIST_NAME, S.ARTIST_ID, ARTIST_LOCATION, ARTIST_LATITUDE, ARTIST_LONGITUDE
3 , COUNT(E.USER_ID) OVER(PARTITION BY E.ARTIST_NAME) USERS_NUMBER
4 FROM EVENTS E, SONGS S
5 WHERE E.ARTIST_NAME = S.ARTIST_NAME) USER_SUB_Q
6 ORDER BY USERS_NUMBER DESC;
```

```

1  -- THE ORDER OF ARTISTS WITH THEIR LOCATIONS ACCORDING TO THE NUMBER OF USERS HEAR THEIR SONGS
2  SELECT DISTINCT * FROM
3
4  (SELECT E.ARTIST_NAME, S.ARTIST_ID, ARTIST_LOCATION, ARTIST_LATITUDE, ARTIST_LONGITUDE
5   , COUNT(E.ARTIST_NAME) OVER(PARTITION BY E.ARTIST_NAME) USERS_NUMBER
6  FROM EVENTS E, SONGS S
7  WHERE E.ARTIST_NAME = S.ARTIST_NAME) USER_SUB_Q
8  ORDER BY USERS_NUMBER DESC;

```

	artist_name character varying (200)	artist_id character varying (100)	artist_location character varying (100)	artist_latitude numeric (20,6)	artist_longitude numeric (20,6)	users_number bigint
1	Lionel Richie	ARIK43K1187B9AE54C	Beverly Hills, CA	[null]	[null]	8
2	Lupe Fiasco	ARPFHN61187FB575F6	Chicago, IL	41.884150	-87.632410	5
3	Tom Petty	ARBEBBY1187B9B43DB	Gainesville, FL	[null]	[null]	4
4	Blue Rodeo	ARD842G1187B997376	Toronto, Ontario, Canada	43.648560	-79.385330	2
5	Elena	AR5KOSW1187FB35FF4	Dubai UAE	49.803880	15.474910	1
6	Gob	ARXR32B1187FB57099	[null]	[null]	[null]	1
7	Gwen Stefani	ARVBRGZ1187FB4675A	[null]	[null]	[null]	1
8	Jimmy Wakely	ARQGY71187FB44566	Mineola, AR	34.311090	-94.029780	1
9	Line Renaud	ARJIE2Y1187B994AB7	[null]	[null]	[null]	1
10	Sophie B. Hawkins	ARNF6401187FB57032	New York, NY [Manhattan]	40.790860	-73.966440	1
11	Trafik	ARKULSX1187FB45F84	Utah	39.499740	-111.547320	1

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

```
SELECT * FROM
(
  SELECT SONG_ID, S.SONG_NAME, S.ARTIST_ID, E.ARTIST_NAME, E.SONG_LENGTH_IN_SECONDS,
  COUNT(USER_ID) OVER(PARTITION BY SONG_ID) USERS_NUMBER
  FROM EVENTS E, SONGS S
  WHERE E.SONG_NAME = S.SONG_NAME AND SONG_PLAYED = 'NextSong'
) SUB_QUERY ORDER BY USERS_NUMBER DESC;
```

Data Output Explain Messages Notifications						
	song_id character varying (100)	song_name character varying (100)	artist_id character varying (100)	artist_name character varying (200)	song_length_in_seconds numeric (26,6)	users_number bigint
1	SOGDBUF12A8C140FAA	Intro	AR558FS1187FB45658	Samy Deluxe	238.366890	3
2	SOGDBUF12A8C140FAA	Intro	AR558FS1187FB45658	Percubaba	124.786490	3
3	SOGDBUF12A8C140FAA	Intro	AR558FS1187FB45658	Calvin Richardson	76.250980	3
4	SOZCTXZ12AB0182364	Setanta matins	AR5KOSW1187FB35FF4	Elena	269.583220	1

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

```
1 SELECT SONG_NAME, ARTIST_NAME, SONG_LEVEL, COUNT(USER_ID) OVER (PARTITION BY SONG_NAME) USERS_NUMBER
2 FROM EVENTS
3 WHERE SONG_PLAYED = 'NextSong'
```

	song_name character varying (200)	artist_name character varying (200)	song_level character varying (200)	users_number bigint
6835	[null]	[null]	tree	950
6836	[null]	[null]	paid	950
6837	[null]	[null]	paid	950
6838	[null]	[null]	paid	950
6839	[null]	[null]	paid	950
6840	[null]	[null]	paid	950
6841	[null]	[null]	free	950
6842	[null]	[null]	free	950
6843	[null]	[null]	free	950
6844	[null]	[null]	free	950
6845	[null]	[null]	free	950
6846	[null]	[null]	free	950
6847	[null]	[null]	free	950
6848	[null]	[null]	paid	950

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

```
Query Editor  Query History

1  SELECT DISTINCT SONG_NAME, ARTIST_NAME, SONG_LEVEL, COUNT(USER_ID) OVER (PARTITION BY SONG_NAME) USERS_NUMBER
2  FROM EVENTS
3  WHERE SONG_PLAYED = 'NextSong'
```

	Data Output	Explain	Messages	Notifications
	song_name character varying (200)	artist_name character varying (200)	song_level character varying (200)	users_number bigint
5569	Put 'Em On	Timbaland & Magoo	paid	1
5570	Let Me (Amended Album Version)	Pleasure P	paid	1
5571	Mr. Sad	Marc Almond	paid	1
5572	The Gift	Angels and Airwaves	free	10
5573	Edge Hill	Groove Armada	paid	1
5574	Pop Is Dead	Radiohead	free	1
5575	Fix You	Coldplay	paid	6
5576	Bloody Well Right	Supertramp	paid	1
5577	Hurtful	Erik Hassle	paid	1
5578	..Come Around	Collie Buddz	free	1
5579	The Beat Is Rockin (Original Mix)	Ericke	free	1
5580	Lights	A Sunny Day In Glasgow	paid	1
5581	Crazy	Aerosmith	paid	2
5582	American Idiot [feat. Green Day & The Cast Of American Idiot...	Green Day	paid	3

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

```

1 SELECT *, DENSE_RANK() OVER(ORDER BY USERS_NUMBER DESC) FROM
2 (SELECT DISTINCT SONG_NAME, ARTIST_NAME, SONG_LEVEL, COUNT(USER_ID) OVER (PARTITION BY SONG_NAME) USERS_NUMBER
3 FROM EVENTS
4 WHERE SONG_PLAYED = 'NextSong') SUB_QUERY_1
5 WHERE SONG_NAME IS NOT NULL ;

```

Data Output		Explain	Messages	Notifications		
	<div>song_name</div> <div>character varying (200)</div>	<div>artist_name</div> <div>character varying (200)</div>	<div>song_level</div> <div>character varying (200)</div>	<div>users_number</div> <div>bigint</div>	<div>dense_rank</div> <div>bigint</div>	
1	You're The One	Dwight Yoakam	paid	37	1	
2	You're The One	Dwight Yoakam	free	37	1	
3	Undo	Björk	free	28	2	
4	Undo	Björk	paid	28	2	
5	Revelry	Kings Of Leon	paid	27	3	
6	Revelry	Kings Of Leon	free	27	3	
7	Sehr kosmisch	Harmonia	free	21	4	
8	Sehr kosmisch	Harmonia	paid	21	4	
9	Horn Concerto No. 4 in E flat K495: II. Romance (Andante c...	Barry Tuckwell/Academy of St Martin-in-t...	free	19	5	
10	Horn Concerto No. 4 in E flat K495: II. Romance (Andante c...	Barry Tuckwell/Academy of St Martin-in-t...	paid	19	5	
11	Canada	Five Iron Frenzy	paid	17	6	
12	Secrets	OneRepublic	free	17	6	
13	Canada	Five Iron Frenzy	free	17	6	
14	Secrets	OneRepublic	paid	17	6	

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

```
SELECT SESSION_ID, SONG_NAME, COUNT(USER_ID) OVER(PARTITION BY SESSION_ID, SONG_NAME) USERS_NUMBER, ARTIST_NAME, SONG_LEVEL
FROM EVENTS
WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong'
```

Data Output	Explain	Messages	Notifications		
	<div>session_id</div> <div>numeric (5)</div> <div></div>	<div>song_name</div> <div>character varying (200)</div> <div></div>	<div>users_number</div> <div>bigint</div> <div></div>	<div>artist_name</div> <div>character varying (200)</div> <div></div>	<div>song_level</div> <div>character varying (200)</div> <div></div>
3158	589	Effervescing Elephant	1	Syd Barrett	paid
3159	589	El Golpe	1	Bebe	paid
3160	589	El Infeliz	1	Los Originales De San Juan	paid
3161	589	En Nat Bliver Det Sommer	1	Love Shop	paid
3162	589	Eton Boating Song/Wyoming Lullaby/The Wiffenpoof Song (Baa Baa ...	1	Reginald Dixon	paid
3163	589	Fast As I Can	1	Erin McKeown	paid
3164	589	Fools	1	The Dodos	paid
3165	589	From Head To Toe	1	Chris Clark	paid
3166	589	Gears	1	Future Rock	paid
3167	589	Getting Better (feat. The Mighty Diamonds)	1	Easy Star All-Stars	paid
3168	589	Halo	2	BeyoncÃ©	paid
3169	589	Halo	2	BeyoncÃ©	paid
3170	589	Haven't We Lost Enough? (LP Version)	1	Crosby_ Stills & Nash	paid
3171	589	Hello	1	Flying Lotus	paid
3172	589	Hello Walls	1	Faron Young	paid

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

```
SELECT DISTINCT SESSION_ID, SONG_NAME, USERS_NUMBER
, ARTIST_NAME, SONG_LEVEL, DENSE_RANK() OVER(PARTITION BY SESSION_ID ORDER BY USERS_NUMBER DESC) SONG_RANK FROM
(SELECT SESSION_ID, SONG_NAME, COUNT(USER_ID) OVER(PARTITION BY SESSION_ID, SONG_NAME) USERS_NUMBER, ARTIST_NAME, SONG_LEVEL
FROM EVENTS
WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong'
) SUB_QUERY
```


Data	Output	Explain	Messages	Notifications		
	session_id numeric (5)	song_name character varying (200)	users_number bigint	artist_name character varying (200)	song_level character varying (200)	song_rank bigint
1	537	Bulletproof	1	La Roux	paid	1
2	381	Benga's Off His Head	1	Benga	paid	1
3	999	Wild World	1	Cat Stevens	paid	1
4	691	Walk Away (Single/LP Version)	1	Epidemic	paid	1
5	620	Bend Over Beethoven	1	!!!	paid	1
6	987	Long Time Gone (LP Version)	1	Crosby_ Stills & Nash	paid	1
7	514	Ja I Ty	1	Fu	free	1
8	548	Better To Reign In Hell	1	Cradle Of Filth	paid	2
9	221	Everlong	1	Foo Fighters	paid	1
10	129	I Believe In You	1	Nancy Wilson	paid	1
11	891	Yellow	1	Coldplay	paid	1
12	764	Look What You've Done (LP Version)	1	Bread	paid	2
13	294	Hanging On By A Thread	1	The Letter Black	paid	2
14	255	Boots On	1	Randy Houser	paid	2
15	869	Oliver'S Army	1	Raimundos	free	1

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)

```

3 SELECT * From
4 (
5 SELECT DISTINCT SESSION_ID, SONG_NAME, USERS_NUMBER
6 , ARTIST_NAME, SONG_LEVEL, DENSE_RANK() OVER(PARTITION BY SESSION_ID ORDER BY USERS_NUMBER DESC) SONG_RANK FROM
7 (SELECT SESSION_ID, SONG_NAME, COUNT(USER_ID) OVER(PARTITION BY SESSION_ID, SONG_NAME) USERS_NUMBER, ARTIST_NAME, SONG
8 FROM EVENTS
9 WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong'
10 ) SUB_QUERY) SUB_QUERY_2
11 GROUP BY SESSION_ID, SONG_NAME, SONG_LEVEL, USERS_NUMBER, ARTIST_NAME, SONG_RANK
12 ORDER BY SESSION_ID, SONG_RANK;
13

```



```

1 SELECT DISTINCT ARTIST_NAME, SONG_NAME, COUNT(SONG_NAME) OVER(PARTITION BY ARTIST_NAME) SONGS_NUMBER
2 FROM EVENTS
3 WHERE SONG_NAME IS NOT NULL

```

	Data Output	Explain	Messages	Notifications
	artist_name character varying (200)		song_name character varying (200)	songs_number bigint
13	12 Stones		Adrenaline	3
14	12 Stones		The Way I Feel (Not Our Master)	3
15	1200 Mics		DNA	1
16	22-20s		Such A Fool	1
17	23 Skidoo		S-Matrix (Part 1 - A Winter Ritual)	2
18	23 Skidoo		Crossfire	2
19	2Mex		Making Money Off God feat. Bus Driver	1
20	3 Doors Down		Here Without You	13
21	3 Doors Down		Kryptonite	13
22	3 Doors Down		She Don't Want The World	13
23	3 Doors Down		Kryptonite	13
24	3 Doors Down		Kryptonite	13
25	3 Doors Down		Here Without You	13
26	3 Doors Down		Kryptonite	13
27	3 Doors Down		Here Without You	13
28	3 Doors Down		Here Without You	13

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

Query EditorQuery History

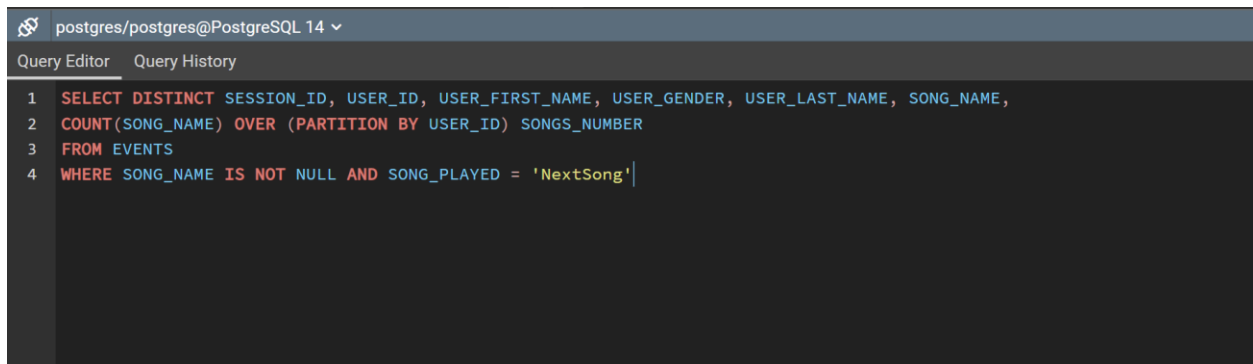
```
1 SELECT *, DENSE_RANK() OVER(ORDER BY SONGS_NUMBER DESC) FROM
2 (
3 SELECT DISTINCT ARTIST_NAME, SONG_NAME, COUNT(SONG_NAME) OVER(PARTITION BY ARTIST_NAME) SONGS_NUMBER
4 FROM EVENTS
5 WHERE SONG_NAME IS NOT NULL) SUB_QUERY;
```

	artist_name character varying (200)	song_name character varying (200)	songs_number bigint	dense_rank bigint
1	Coldplay	42	58	1
2	Coldplay	A Message	58	1
3	Coldplay	Fix You	58	1
4	Coldplay	Yellow	58	1
5	Coldplay	Speed Of Sound	58	1
6	Coldplay	In My Place	58	1
7	Coldplay	One I Love	58	1
8	Coldplay	Brothers & Sisters	58	1
9	Coldplay	Only Superstition	58	1
10	Coldplay	Til Kingdom Come	58	1
11	Coldplay	Shiver	58	1
12	Coldplay	Trouble	58	1
13	Coldplay	The Scientist	58	1
14	Coldplay	Life In Technicolor ii	58	1
15	Coldplay	Cemeteries Of London	58	1

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



```
postgres/postgres@PostgreSQL 14 ▾
Query Editor  Query History
1  SELECT DISTINCT SESSION_ID, USER_ID, USER_FIRST_NAME, USER_GENDER, USER_LAST_NAME, SONG_NAME,
2  COUNT(SONG_NAME) OVER (PARTITION BY USER_ID) SONGS_NUMBER
3  FROM EVENTS
4  WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong'|
```

Data Output		Explain	Messages	Notifications			
	session_id numeric (5)	user_id numeric (10)	user_first_name character varying (200)	user_gender character varying (5)	user_last_name character varying (200)	song_name character varying (200)	songs_number bigint
1	594	25	Jayden	M	Graves	Isla Mujeres	169
2	833	8	Kaylee	F	Summers	There's Your Trouble	27
3	898	29	Jacqueline	F	Lynch	From This Day Forward	346
4	525	66	Kevin	M	Arellano	Move The Crowd	37
5	293	97	Kate	F	Harrell	When You Were Young	557
6	491	26	Ryan	M	Smith	Jeanie Jeanie Jeanie	114
7	237	44	Aleena	F	Kirby	Summertime Clothes	397
8	72	73	Jacob	M	Klein	Kitty Kat	289
9	264	69	Anabelle	F	Simpson	Teach Me How To Dougie	29
10	806	97	Kate	F	Harrell	Livin' On A Prayer	557
11	129	42	Harper	M	Barrett	Ain't Nobody (Album Version)	140
12	548	80	Tegan	F	Levine	You Wanted It (Album Version)	665
13	935	37	Jordan	F	Hicks	Your Body's Callin'	34
14	128	25	Jayden	M	Graves	Coldest Winter	169
15	275	42	Harper	M	Barrett	I'm Like A Lawyer With The Wa...	140

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

```

1
2 SELECT *, DENSE_RANK() OVER(ORDER BY SONGS_NUMBER) USER_RANK
3 FROM
4 (
5 SELECT DISTINCT SESSION_ID, USER_ID, USER_FIRST_NAME, USER_GENDER, USER_LAST_NAME, SONG_NAME,
6 COUNT(SONG_NAME) OVER (PARTITION BY USER_ID) SONGS_NUMBER
7 FROM EVENTS
8 WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong') SUB_QUERY;|

```

Data Output	Explain	Messages	Notifications				
session_id numeric (5)	user_id numeric (10)	user_first_name character varying (200)	user_gender character varying (5)	user_last_name character varying (200)	song_name character varying (200)	songs_number bigint	user_rank bigint
486	29	Jacqueline	F	Lynch	Use Somebody	346	48
589	29	Jacqueline	F	Lynch	Howlin' For You	346	48
556	29	Jacqueline	F	Lynch	Danke Schoen	346	48
589	29	Jacqueline	F	Lynch	A Girl Named You	346	48
486	29	Jacqueline	F	Lynch	Time To Pretend	346	48
709	29	Jacqueline	F	Lynch	The Trance Is The Motion [Live]	346	48
963	29	Jacqueline	F	Lynch	8105	346	48
559	29	Jacqueline	F	Lynch	Enough To Get Away With	346	48
589	29	Jacqueline	F	Lynch	Pursuit Of Happiness (nightmare)	346	48
372	29	Jacqueline	F	Lynch	Swedish Purse	346	48
389	29	Jacqueline	F	Lynch	How Can You Buy Killarney (1992 Digital Rema...	346	48
486	29	Jacqueline	F	Lynch	Television Rules The Nation / Crescendolls	346	48
589	29	Jacqueline	F	Lynch	Never Say Never	346	48
486	29	Jacqueline	F	Lynch	No No No	346	48
589	29	Jacqueline	F	Lynch	Bust A Move	346	48

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.

Query Editor
Query History

```

1  -- GET THE LONGEST SONG and SHORTEST SONG IN EACH SESSION
2
3  SELECT SONG_NAME, ARTIST_NAME, SESSION_ID, SONG_LENGTH_IN_SECONDS, USER_ID,
4  FIRST_VALUE(SONG_NAME) OVER(PARTITION BY SESSION_ID ORDER BY SONG_LENGTH_IN_SECONDS DESC) LONGEST_SONG,
5  FIRST_VALUE(SONG_NAME) OVER(PARTITION BY SESSION_ID ORDER BY SONG_LENGTH_IN_SECONDS) SHORTEST_SONG
6  FROM EVENTS
7  WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong'
8  ORDER BY SESSION_ID;

```

	song_name character varying (200)	artist_name character varying (200)	session_id numeric (5)	song_length_in_seconds numeric (26,6)	user_id numeric (10)	longest_song character varying	shortest_song character varying
1	Read My Mind	The Killers	3	246.804440	4	Read My Mind	Read My Mind
2	Just Might (Make Me Believe)	Sugarland	5	247.770980	6	Absolution: Of Flight and Failure	Just Might (Make Me Believe)
3	Absolution: Of Flight and Failure	A Hope For Home	5	388.388120	6	Absolution: Of Flight and Failure	Just Might (Make Me Believe)
4	Dirtouse (Album Version)	Static-X	6	183.692610	7	Dirtouse (Album Version)	Dirtouse (Album Version)
5	SÄÄn't ÄÄr Livet (You Can Have ...	Anne-Lie RydÄÄ	8	176.900770	9	Master Of Puppets	SÄÄn't ÄÄr Livet (You Ca
6	Cuando	Juan Carlos Baglietto	8	221.805260	9	Master Of Puppets	SÄÄn't ÄÄr Livet (You Ca
7	There Is A Light That Never Goes Out	The Smiths	8	242.912200	9	Master Of Puppets	SÄÄn't ÄÄr Livet (You Ca
8	Era En Abril	Juan Carlos Baglietto	8	285.648530	9	Master Of Puppets	SÄÄn't ÄÄr Livet (You Ca
9	Master Of Puppets	Metallica	8	515.212610	9	Master Of Puppets	SÄÄn't ÄÄr Livet (You Ca
10	Pump It	Black Eyed Peas	9	214.935060	10	Pump It	Pump It
11	Rehab	Rihanna	10	293.824850	11	Rehab	Rehab
12	Tennessee Waltz (Live)	Billy J Kramer & The Da...	15	191.320360	16	All Summer Long (Album Version)	Tennessee Waltz (Live)
13	Can't Help But Wait (Album Version)	Trey Songz	15	205.818320	16	All Summer Long (Album Version)	Tennessee Waltz (Live)
14	When You Were Young	The Killers	15	220.890980	16	All Summer Long (Album Version)	Tennessee Waltz (Live)
15	Ghost Of A Good Thing	Dashboard Confessional	15	224.548120	16	All Summer Long (Album Version)	Tennessee Waltz (Live)

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

```
1 SELECT USER_ID, USER_FIRST_NAME, USER_LAST_NAME
2 ,COUNT(USER_ID) OVER(PARTITION BY USER_ID) NUMBER_OF_TRAILS
3 , COUNT (CASE WHEN SONG_STATUS = 200 THEN 1 END) OVER(PARTITION BY USER_ID) SUCCESSFUL_TRAILS
4 FROM EVENTS
5 WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong' |
```

Data Output		Explain	Messages	Notifications		
	user_id numeric (10)		user_first_name character varying (200)	user_last_name character varying (200)	number_of_trails bigint	successful_trails bigint
1	2	Jizelle	Benjamin	10	10	
2	2	Jizelle	Benjamin	10	10	
3	2	Jizelle	Benjamin	10	10	
4	2	Jizelle	Benjamin	10	10	
5	2	Jizelle	Benjamin	10	10	
6	2	Jizelle	Benjamin	10	10	
7	2	Jizelle	Benjamin	10	10	
8	2	Jizelle	Benjamin	10	10	
9	2	Jizelle	Benjamin	10	10	
10	2	Jizelle	Benjamin	10	10	
11	3	Isaac	Valdez	3	3	
12	3	Isaac	Valdez	3	3	
13	3	Isaac	Valdez	3	3	
14	4	Alivia	Terrell	5	5	
15	4	Alivia	Terrell	5	5	
16	4	Alivia	Terrell	5	5	
17	4	Alivia	Terrell	5	5	









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].

```

4 SELECT DISTINCT USER_ID, USER_FIRST_NAME, USER_LAST_NAME, NUMBER_OF_TRAILS, SUCCESSFUL_TRAILS,
5 CAST(SUCCESSFUL_TRAILS AS FLOAT) /NUMBER_OF_TRAILS SUCCESSFUL_PERCENTAGE
6 FROM
7 (
8 SELECT USER_ID, USER_FIRST_NAME, USER_LAST_NAME
9 ,COUNT(USER_ID) OVER(PARTITION BY USER_ID) NUMBER_OF_TRAILS
10 , COUNT (CASE WHEN SONG_STATUS = 200 THEN 1 END) OVER(PARTITION BY USER_ID) SUCCESSFUL_TRAILS
11 FROM EVENTS
12 WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong' ) SUB_QUERY
13 ORDER BY USER_ID;
14

```

Data Output		Explain	Messages	Notifications			
	user_id numeric (10) 	user_first_name character varying (200) 	user_last_name character varying (200) 	number_of_trails bigint 	successful_trails bigint 	successful_percentage double precision 	
1	2	Jizelle	Benjamin	10	10	1	 Successfully run. Total query runtime: 65 ms
2	3	Isaac	Valdez	3	3	1	
3	4	Alivia	Terrell	5	5	1	
4	5	Elijah	Davis	4	4	1	
5	6	Cecilia	Owens	23	23	1	
6	7	Adelyn	Jordan	5	5	1	
7	8	Kaylee	Summers	27	27	1	
8	9	Wyatt	Scott	16	16	1	
9	10	Sylvie	Cruz	28	28	1	
10	11	Christian	Porter	2	2	1	
11	12	Austin	Rosales	12	12	1	
12	13	Ava	Robinson	5	5	1	
13	14	Theodore	Harris	22	22	1	
14	15	Lily	Koch	463	463	1	
15	16	Rylan	George	223	223	1	
16	17	Makinley	Jones	7			
17	18	Jacob	Rogers	5			

✓ Successfully run. Total query runtime: 65 ms

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

```
SELECT DISTINCT USER_ID, USER_FIRST_NAME, USER_LAST_NAME,  
COUNT(CASE SONG_LEVEL WHEN 'paid' THEN 1 END) OVER(PARTITION BY USER_ID) PAID_SONGS_NUMBER,  
COUNT(CASE SONG_LEVEL WHEN 'free' THEN 1 END) OVER(PARTITION BY USER_ID) FREE_SONGS_NUMBER  
FROM EVENTS  
WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong' |
```









	user_id numeric (10)	user_first_name character varying (200)	user_last_name character varying (200)	paid_number bigint	free_number bigint
1	76	Jayden	Duffy	0	14
2	4	Alivia	Terrell	0	5
3	61	Samuel	Gonzalez	0	24
4	22	Sean	Wilson	0	2
5	50	Ava	Robinson	0	48
6	86	Aiden	Hess	0	45
7	30	Avery	Watkins	178	0
8	42	Harper	Barrett	140	0
9	29	Jacqueline	Lynch	341	5
10	9	Wyatt	Scott	0	16
11	82	Avery	Martinez	87	0
12	20	Aiden	Ramirez	9	0
13	68	Jordan	Rodriguez	0	3

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

```

2
3 SELECT USER_ID, USER_FIRST_NAME, USER_LAST_NAME, PAID_SONGS_NUMBER, FREE_SONGS_NUMBER,
4 CAST (PAID_SONGS_NUMBER AS FLOAT) / (PAID_SONGS_NUMBER + FREE_SONGS_NUMBER) PAID_SONGS_PERCENTAGE
5 FROM
6 (
7 SELECT DISTINCT USER_ID, USER_FIRST_NAME, USER_LAST_NAME,
8 COUNT(CASE SONG_LEVEL WHEN 'paid' THEN 1 END) OVER(PARTITION BY USER_ID) PAID_SONGS_NUMBER,
9 COUNT(CASE SONG_LEVEL WHEN 'free' THEN 1 END) OVER(PARTITION BY USER_ID) FREE_SONGS_NUMBER
10 FROM EVENTS
11 WHERE SONG_NAME IS NOT NULL) SUB_QUERY
12 ORDER BY USER_ID;

```

Data Output		Explain	Messages	Notifications			
	 user_id numeric (10)	 user_first_name character varying (200)	 user_last_name character varying (200)	 paid_songs_number bigint	 free_songs_number bigint	 paid_songs_percentage double precision	
8	9	Wyatt	Scott	0	16	0	
9	10	Sylvie	Cruz	0	28	0	
10	11	Christian	Porter	0	2	0	
11	12	Austin	Rosales	0	12	0	
12	13	Ava	Robinson	0	5	0	
13	14	Theodore	Harris	0	22	0	
14	15	Lily	Koch	462	1	0.9978401727861771	
15	16	Rylan	George	208	15	0.9327354260089686	
16	17	Makinley	Jones	0	7	0	
17	18	Jacob	Rogers	0	5	0	
18	19	Zachary	Thomas	0	9	0	
19	20	Aiden	Ramirez	9	0	1	
20	22	Sean	Wilson	0	 Successfully run. Total query run		

✓ Successfully run. Total query run

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)

```
1  -- HOW LONG EACH USER SPEND IN OUR WEB SITE
2
3  SELECT DISTINCT USER_ID, USER_FIRST_NAME, USER_LAST_NAME,
4  SUM(SONG_LENGTH_IN_SECONDS) OVER(PARTITION BY USER_ID) USER_DURATION_IN_SECONDS
5  FROM EVENTS
6  WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong';
7
8
9
```

	Data Output	Explain	Messages	Notifications
	user_id numeric (10)	user_first_name character varying (200)	user_last_name character varying (200)	user_duration_in_seconds numeric
1	71	Ayleen	Wise	1081.336510
2	9	Wyatt	Scott	4057.566630
3	39	Walter	Frye	387.943580
4	41	Brayden	Clark	2392.054710
5	24	Layla	Griffin	80918.476550
6	84	Shakira	Hunt	1669.091150
7	43	Jahiem	Miles	2699.802360
8	100	Adler	Barrera	5220.562810
9	22	Sean	Wilson	378.069290
10	49	Chloe	Cuevas	171592.386360
11	3	Isaac	Valdez	525.634560
12	10	Sylvie	Cruz	6660.506510
13	78	Chloe	Roth	3083.774870
14	86	Aiden	Hess	10558.046540
15	90	Andrea	Butler	713.089260

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

```

Query Editor  Query History
1  -- HOW LONG EACH USER SPEND IN OUR WEB SITE
2
3  SELECT USER_ID, DENSE_RANK() OVER(ORDER BY USER_DURATION_IN_SECONDS DESC) ,
4  USER_FIRST_NAME, USER_LAST_NAME, USER_DURATION_IN_SECONDS
5  FROM
6  (
7  SELECT DISTINCT USER_ID, USER_FIRST_NAME, USER_LAST_NAME,
8  SUM(SONG_LENGTH_IN_SECONDS) OVER(PARTITION BY USER_ID) USER_DURATION_IN_SECONDS
9  FROM EVENTS
10 WHERE SONG_NAME IS NOT NULL AND SONG_PLAYED = 'NextSong') SUB_QUERY;
11
12
13

```

Data Output		Explain	Messages	Notifications			
	user_id numeric (10)		dense_rank bigint	user_first_name character varying (200)	user_last_name character varying (200)	user_duration_in_seconds numeric	
1		49	1	Chloe	Cuevas	171592.386360	
2		80	2	Tegan	Levine	165578.042890	
3		97	3	Kate	Harrell	137032.184830	
4		15	4	Lily	Koch	118222.732070	
5		44	5	Aleena	Kirby	95128.714690	
6		29	6	Jacqueline	Lynch	82995.617850	
7		24	7	Layla	Griffin	80918.476550	
8		73	8	Jacob	Klein	70658.063190	
9		88	9	Mohammad	Rodriguez	67108.187710	
10		36	10	Matthew	Jones	63484.779040	
11		16	11	Rylan	George	56552.022950	
12		95	12	Sara	Johnson	48936.262670	

Thank You

