Dhirubhai Ambani
Institute of Information and Communication Technology

Course: **Database Management System**

Course ID: **IT214**

Year: **2021-2022**

Online Music Management System

Section 06, Team 09

Mentor TA: Kavan

Student ID	Names
201901076	Ladani Utsav
201901090	Pandar Mayur
201901131	Solanki Bhavya
201901304	Joshi Dev

Section1: Final Version Of SRS

Table of content

1. Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the **Online Music Management System (OMMS)**. The OMMS (Online Music Management System) is a software that facilitates Good user experience with synchronization of both user experience and copyrights while providing services like online music management, legal downloads, artists' management. There are several other applications available in the market(Jio Savan, YT music Spotify etc.) that either provide some specific services or large scale integrated solutions. This Software differs from the rest in a way that we give more power to the users remaining within the copyrights circle and also some other functionality. This document is for both developers and users of this system. This software provides users to give review and rating to a song. It also provides copyright management for different songs and albums.

This document contains the basic functionality of the frontend and backend of OMM system, system flow or working principles, error handling, efficient database design, and common problems.

1.2 Document Conventions

This document is written in Latex. Whole document is written in Arial font with simple text having fontsize of 12. Important terms In this document are written in bold style.

1.3 Intended Audience and Reading Suggestions

This SRS doc is helpful for developers who want to develop a music application with an online database, Database architecture who want to make an efficient database system, Product managers, startups, etc.

This document is written in a few sections. Product sections are for product managers, startups, etc. These sections describe the use case of the product, user charactrics, etc. Design and architecture sections are helpful for developers to develop efficient online databases. This section gives the information about schema, data type and data. Other sections are written with simple language so easily understood by any readers who have basic knowledge of databases.

Any reader and developers are welcomed to use this document for their work. If a reader has any suggestions about this document or this OMMS system they can contact us on ommsproduct@gmail.com.
If you want to send lyrics in your language use the above email address to send lyrics to us. We will verify lyrics and add it to our system if it is correct. To appreciate this effort of the user we will provide a 2 month premium subscription to the user.

SRS Docs includes:

1. Description of OMMS
2. Details of user interface
3. System features
4. Efficient Database design
5. Privileges of Database
6. Business rules

1.4 **Product scope**

According to the survey 60-70% of youth listen to music, so there is always a need for a music system which can fulfill the requirements of people. They want to listen to music anywhere, anytime. To satisfy this requirement we develop OMMS, so that our collection of music is accessed by anyone, anywhere, anytime with internet connection and no need to fill device space with many songs, just select a song from the list and listen to it. We use a data stream to transfer music over the internet and it takes only a few KBs of space. Also the online database is very easy to maintain and update with the latest songs. This OMMS system touches various different Operating systems (Android to Microsoft). We are also providing the advanced facility of search in our large music database.

We are concerned that our system does not violate any copyright rules, so users can't download the song on his/her device without permission of the creator to preserve the copyright of music. So there is no issue related to piracy of songs. If a user wants to download the song, then a premium account is required. Only users with a premium account can download the song, but we store it in encrypted format so they can't distribute their own. And this encryption is different for each device. So there is total security.

To make money, the application contains ads and if a user wants to remove these ads, then he/she must buy a premium account, which provides an add-free user experience.
Till now our OMMS is a standalone system, We can make more money by integrating our OMMS system with telecom. Companies, and giving their users caller tunes from our Database. More details about this are given in the **Business constraints** section.

1.5 Description

The Online Music Management System will provide a large collection of songs to every user who is connected with our service over the internet. Users can search music and/or select from a given list of songs and that song will play in the appropriate app or system (web / android / desktop). As far as the IT214 subject is concerned, this project will be database oriented. Our goal is to make efficient database design, so that maximum users can connect with our service using minimum resources.

The system consist of three parts;

1. **User interface:** installed on device or accessed via browser
2. **Server:** connect application and database, handle user request, stream song, register user,etc.
3. **Database:** Store the songs and user information, provide some basic functions to modify and search into the database. It uses a PostgreSQL database.

Basic Workflows of Client Application

- **Register/SignUp:** Every user needs to register in application when he/she operates for the first time.Without registering users can listen to only 20 songs.

- **SignIn:** Every time when a user opens an application, he/she must have to sign in using their credentials.

- **Suggestion/Song List:** After signing in, users get the list of some song according to their preference and history.

- **Search Bar:** Search the song in our database using various filters like artist name, rating, views, type, date, movies, etc.

- **Music Player:** Play, pause, prev, next, queued, etc. functionalities are handled by the music player.
- **User Profile:** show the information about user, account type(premium/comman), update profile, etc.
- **Upload Song:** Users can upload songs on the server.
- **Money:** Give the money to artists according to views and rating.

Basic Workflows of Server

- **Authentication Handling:** all authentication related functionality like register user, check whether the user's credential is right or not, etc. are done by this part.
- **Search request:** Send the result of the search query to the application in JSON format.
- **Song request:** Start the streaming of songs on a particular user's device, download, upload can be done via this service.
- **Update:** Update song rating, views,etc. and user data also.
- **Money:** All money transactions controlled by this part.

Basic Workflow of Database

- **User:** Add, delete, update user and also update account to premium account.
- **Song:** Add, delete, update song.
- **Money:** Money transaction.

Popular Issues and Solutions

- **UI/UX:** Modern apps must have to provide a customization of theme and layout because bright light can harm eyes at night and UI with dark color can not be visible in day. So instance switching of themes was needed. Different layouts give the diversity of system usage.
- **Suggestion:** Users don't want the same type of suggestion all the time. They want to customize the suggestion based on their own requirements. No any existing system have this kind of feature to customize the suggestion for each user according to user requirements
- **Filter:** Only Suggestions are not enough, people must need some filter to get a particular song list. All music platforms are providing different filters, but it's not sufficient. Users want to filter the songs according to the group of attributes. So I need some sort of feature to apply more than one filter on any list of songs, even in a playlist, albums ,search result, etc.
- **Offline Availability:** Copyright concern prohibits the user to download songs. Many places like gym, car on highway, train, flight etc. have low connectivity due to closed infrastructure or mobility of devices. In

such cases users want the offline availability of songs. There is some smart system required so that users can download songs without violating copyrights.

- **Other:** Playlists are too important to feature in any music system. System must take care of this feature. Users should be able to share songs with some privileges. Users also want to gather online and organize a virtual party (party means one can play a song and other listen to it), podcast, etc. So the system adds these features so that users get a better experience.

Online Music Management System(OMMS) will help to categorize using song name, artist, albums, rating, type, time, type, number of listener, etc. Admin will be able to add songs. Users will be able to create their favourite playlists, Download and Share songs. This system also has features for singers and authors of albums. If some other singer wants to use a song or it's lyrics in his song, He will have to take permission from the owner of that song. This system will alert the owner of a song or album when someone publishes a song or album containing the song of the owner. Based on that if he has not taken permission from the owner he will be charged a penalty. We are providing **Repeat Current song, Repeat playlist, Shuffle playlist** like features for users. Users can make their own playlist by adding their favourite songs. The system also has a **play in background** feature so a user doesn't have to keep the application open while listening to music. To use this play in background feature the user has to buy premium.To set any song for caller tune(song for incoming call) the user has to buy premium.

There are several other applications in the market which are similar to our OMMS software.these applications/softwares provide several functionalities like search music,add different songs in the favourite song list,set caller tune,lyrics,etc.

Search Music: By this function you can search the music song whichever you want.

Favourite song list:some applications provide a Favourite song library/folder and in this library you can add the song so it notes the song to your favourite song.

Caller tune:this function provides the ability to set the song for incoming calls.

Lyrics:some other applications provide lyrics of songs.

2. Fact finding Phase

2.1 Background Readings

We have analyzed two products from the market.

2.1.1 Description of each reading done

2.1.1.1 YT Music

YT Music is a music streaming service developed by YouTube. YT Music provides a tailored interface for the service, oriented towards music streaming. It allows users to browse through songs and music videos on YouTube based genres, playlists and recommendations. YT music is available for android (application), Desktops (Website).YT Music service consists of three parts.

1. **User Interface:** Device application or website

2. **Database:** Stores songs and user information. This information is stored in Youtube's cloud storage.
3. **Server:** It connects an application to the database. It provides many features like search, share, download, sign in etc.

Basic workflows of YT Music application

- **Sign Up or Register :** Users don't need to register to listen to songs on YT music, But If a user wants to use his premium features he should have signed up in YT music.

- **Login or Sign in :** Again if users just want to listen to songs and don't want to use premium features they can do it so without log in. to use premium features one should have to login to his account.

- **Search bar :** Searches songs in Youtube database by using various filters like song name, artist name, movie or album name etc. It uses Google's Powerful algorithms for searching.

- **Suggestions/Recommendations :** It provides suggestions/recommendations based on various things like Trending songs, Popular, albums, Your past listened songs, most listened songs etc.

- **Library :** Stores recent activity of users. Users can also view history.
- **Account** : This has many features like switch account, create channel, settings, Your channel etc

- **Create Channel :** Users can create their own channel on YT music.

- **Upload Music :** Users can upload their music in YT music.

- **Upgrade to premium :** Users can buy premium from here

- **Help :** Gives Users help about application or website interface.

- **Terms and privacy policy :** Provides users terms of usage (for example a user must be 13 years old) and gives information about privacy that YT Music provides to users.

- **Feedback :** Users can send feedback about anything in YT Music.

- **Money :** Gives money to the channel owner according to certain rules.

- **Restrictions :** It has some restrictions for some premium songs. Users need a premium account to access those songs.

- **Connect to a device :** Users are able to connect YT Music with their smart TV.

- **Ratings :** Users can give likes or dislikes based on their choice.

- **Music Player :** Basic Music player that can play, pause, skip songs. It also gives features like repeat this song, shuffle etc.

Basic workflows of Database

- **Users :** Stores information about user account, also stores information about user's premium account, It adds and deletes user according to server requests.

- **Storage :** YT Music stores its data in YouTube's Cloud Storage.
.
- **Song :** Stores music, Add music, Delete Music.

- **Money :** Stores transaction information of its users.

Basic workflows of Server

- **Search request :** It finds Songs in Database using advanced algorithms of google or youtube and sends results to YT Music application or website.

- **Connect request :** It connects YT Music with other devices like Android TV, Desktop, Laptop etc.

- **Update request :** It updates songs, song views etc. based on request from application or website.

- **Song request :** It provides facilities like Streaming song in a particular device, share song, download song etc.

- **Money :** It does all transactions successfully without any errors.

- **Authentication Handling :** All tasks like signing in user, logging in user, checking premium of user etc. are done by this part.

Issues

- **UI/UX:** Many modern apps provide a preference option to customise UI according to user requirements, but YoutTube music does not provide such options to set presences like themes, layout, zen mode, etc.

- **Video quality:** We are not able to change the video quality on YouTube music, it may lead to high internet data consumption.

- **Suggestions:** YouTube Music some suggest most listen to songs by you. It may be good but after a long time users don't enjoy to listen same songs again and again. It keeps users far from the new experience of different types of songs.

- **Filter:** There is not enough filter eg. We are not able to apply filters on songs of particular artists like we can not sort songs of badshah by alphabets in youtube music.

- **Copyrights:** Anyone can upload a duplicate song, there is no copyright checking before publishing.

- **Other:** Users are not able to select multiple songs from the playlist to remove. Users have to remove songs one after another. Downloaded music shows blurry album pictures. etc.

This application also provides premium features which enables ad-free playback, background audio only playback and downloading songs for offline playback. This service is designed for those users who primarily listen to music through YouTube. This YT Music works on several devices having different operating systems. It recommends Popular songs to users. Searching on the base of Lyrics, Song name, Singer, Album name etc. is very fast in YT Music. It also provides a voice search feature in which we can say anything about the song and it will give us the best matching result of that song. It provides a selection of bit rate at which song is streaming or downloading to premium users. You can download YT Music in Smartphones from their appropriate app store. For other devices, below is the website of YT Music service.

https://music.youtube.com/

2.1.1.2 Spotify

Spotify is an application that provides the service to listen to music online. Spotify streams the songs over the internet so that people can listen to it everywhere. It allows users to stream music, share music with everyone who is using Spotify. This application provides premium features which allow users to play ad-free playback. It also provides digital copyright restrictions for recorded songs/music.it also supports searching the music based on artists, albums and genres. When you search for some music, it provides suggestions based on artists and songs and albums. On Spotify, you can create your own playlist and also stream it. Based on your music or playlist users can give you ratings and likes.on Spotify, some basic features are free with advertising but for the additional features like offline music listening and commercial-free listening, you need to buy a subscription.Spotify can be represented in three parts.

1. **User Interface:** two types of interfaces are possible 1. Application 2. website
2. **Database:** stores songs, user information, favourite song list, etc.
3. **Server:** server connects application or website to the database and handles some user requests like pause, play, change song, search song etc.

Basic Workflow of Spotify application.

- **Register or sign up:** Every user needs to register on Spotify when the user uses it for the first time. The listened songs, favourite songs and others are all linked with the registration.
- **Sign In:** when a user uses the website interface for Spotify then the user needs to sign in every time to fetch the user details but in the application sometimes you need to sign in.
- **Search song or search bar:** On Spotify, you can search songs based on the artist, song name, Lyrics, album name etc. this searching happens in the databases with various commands and algorithms.
- **Update account:** users can update their accounts on spotify. If a user uses a free account then He/she can update his/her account to a premium account.

- **Reset password:** in some cases, if the user doesn't know their account password then they can reset the by forgot password option.

- **Suggestion:** when the user searches for some song, album or anything then in the below search bar Spotify gives suggestions Based on trending, popular artists, albums, songs, podcasts, recommendations and also your past history.

- **Favourite songs list:** user can add the song in his favourite songs list. So, when a user needs to listen to his/her favourite song , he/she does not need to search for the song everytime. He/she can listen to it from his/her favourite songs list.

- **Create playlist:** users can create their own playlist and also can stream the song which is created by them.

- **Copyright restrictions:** if a user wants to use some other user's lyrics or song then he/she(user) needs to take permission from the lyrics's owner.if user use the song's lyrics without lyrics's owner permission then this will be called infringement and user break the rules of terms and conditions rule. Spotify can take legal action against him/her.

- **Connect to a device:**You can connect some other device with Spotify like TV, speaker etc.
.
- **Money :** gives the money to the playlist owner according to some specific rules.

- **Help:** users can take help about the application interfaces.

- **Feedback:** users can give their feedback related to the application.

Basic workflows of database

- **User:** it stores account information and user related details. It adds and deletes the user according to registration on spotify.

- **Money:** all the money transition of user's is stored in this area.

- **Store:** spotify uses all user's cache memory and also hard drive to store the information about listened songs history, liked song etc.

- **Song:** new songs will be added in this database and deleting will also happen in this database.

Basic workflows of server

- **Connect request:** when we connect Spotify with other devices like TV, speaker. Then it will connect via connection request from the server.also when the user enters in spotify then a connection request will arise.

- **update request :** it updates song's information like views, likes, lyrics and also updates the songs.for particular user, if user updates his/her profile then this is also handled by this update request.

- **Money related or transaction related request:** it handles all the errors related to the money transaction.

- **Search request:** all the searching related work will be handled by this request like searching suggestion. This searching will happen by some algorithms.

- **Authentication handling:** all user's premium plans remaining days, sign in or loging related information will be handled by this

- **Song request:** this handles the sharing song, download songs and streaming songs requests.

Issues

- **Sharing Privileges:** There is lack of admin privileges for collaborative sharing and creation of playlist. Either all can access or only one can access the playlist, there needs to be some good privilege system.

- **Suggestion:** There is no way to stop suggestions from some particular artist or production company, etc. Does not have any filter to customize suggestions like suggestions include 40% newly published songs, 20% from history, 50% from popular songs, etc.

- **Filter:** App is not able to create a radio playlist based on only selected artists.

- **Podcast:** Podcast is a very good feature of spotify, but users get bad experiences like bad podcast organization, ads, etc.

- **Other:** Users are not able to select multiple songs from the playlist to remove. users have to remove songs one after another. Few browser specific bugs. Allow multiple select playlists to allow for mass deletion.

On the Smartphone, you can download Spotify from the appropriate app store and for desktop, you use their website which is given below.

https://www.spotify.com/us/

2.1.2 References:

YT music :

- By using YT music website, Application.
- https://en.wikipedia.org/wiki/YouTube_Music
- https://www.pocket-lint.com/apps/news/youtube/144541-what-is-youtube-music-youtube-s-new-music-streaming-service-explained
- https://www.androidpolice.com/2020/06/09/hands-on-youtube-music-upload/

Spotify :

- https://en.wikipedia.org/wiki/Spotify
- https://pageflows.com/product/spotify/
- Using the spotify application we wrote some functions or features.

2.1.3 Combined Requirements

- To check information about users' premium accounts we should have a good authentication handler in our OMMs system.
- We should make our system platform Independent. (For example, our system should work on various Operating systems and various devices.)
- We should have better storage options to store songs in our Database of OMMs.
- We will need a strong server for our OMMs which can communicate with Database nicely.
- For Online transactions between system and users, we will need a better transaction management system.

2.2 Interviews

2.2.1 Interview plans and summaries

From the following Interviews of various roles are given. For each Interview, First Interview plan is written and then Interview summary is written.

OMMs: (Actual) Interview Plan 1

System: OMMs (Online Music Management System)

Project Reference:

Interviewee: 1) Dharmik Solanki(Actual)

Designation:User of OMMs

Contact Details: solbha9@gmail.com.

Organization Details: Student at DAIICT

Interviewer: 1) Mayur Pandar

Designation: OMMs admin

Interviewer: 2) Dev Joshi

Designation: OMMs admin

Date: 05/10/2021 **Time:** 21:00

Duration: 60 minutes **Place:** Online (Google meet)

Purpose of Interview:

Simple meeting to know user requirements for OMMs.

Agenda:

Users' expectations from an Online Music Management System.

Users' suggestions about improving these kinds of systems.

Follow-up actions

Documents to be brought to the interview:

Nothing

OMMs: (Actual) Interview Summary 1

System: OMMs (Online Music Management System)

Project Reference:

Interviewee: 1) Dharmik Solank(Actual)

Designation: User of OMMs

Contact Details: solbha9@gmail.com

Organization Details: Student at DAIICT

Interviewer: 1) Mayur Pandar
Designation:OMMs admin

Interviewer: 2) Dev Joshi

Designation: OMMs admin

Date: 05/10/2021 **Time:** 21:00

Duration: 60 minutes **Place:** Online(Google meet)

Purpose of Interview:

Simple meeting to know user requirements for OMMs.

1. System should have a beautiful UI and eye protecting theme.
2. All types of songs should be available in the system.
3. System should not face network related issues if the user has a good internet connection.

OMMs: (**Actual**) **Interview Plan 2**

System: OMMs(Online Music Management System)

Project Reference:

Interviewee: 1) Vrinda Mayatra(**Actual**)

Designation: User of OMMs

Contact Details: vrinda22mayatra@gmail.com

Organization Details: Student at DAIICT

Interviewer: 1) Bhavya solanki

Designation: OMMs admin

Interviewer: 2) Utsav Ladani

Designation: OMMs admin

Date: 06/10/2021 **Time:** 17:00

Duration: 60 minutes **Place:**Online(Google meet)

Purpose of Interview:

Simple meeting to know user requirements for OMMs.

Agenda:

Users' expectations from an Online Music Management System.

Users' suggestions about improving these kinds of systems.

Follow-up actions

Documents to be brought to the interview:

Nothing

OMMs: Mock/Actual Interview Summary 2

System: OMMs(Online Music Management System)

Project Reference:

Interviewee: 1) Vrinda Mayatra(**Actual**)

Designation:User of OMMs

Contact Details: vrinda22mayatra@gmail.com

Organization Details: Student at DAIICT

Interviewer: 1) Bhavya Solanki

Designation: OMMs admin

Interviewer: 2) Utsav Ladani

Designation:OMMs admin

Date: 06/10/2021 **Time:** 17:00

Duration: 60 minutes **Place:** Online(Google meet)

Purpose of Interview:

Simple meeting to know user requirements for OMMs.

1. System should have less ads.
2. System should have less price for premium subscription.
3. System should give discounts on the price of premium subscriptions.

OMMs: (Role Play) Interview Plan 3

System: OMMs(Online Music Management System)

Project Reference:

Interviewee: 1) Camila Cabello(Role Play)

Designation:Singer

Interviewer: 1) Mayur Pandar

Designation: OMMs admin

Interviewer: 2) Bhavya Solanki

Designation: OMMs admin

Date: 06/10/2021 **Time:** 21:00

Duration: 60 minutes **Place:** Online(Google meet)

Purpose of Interview:

Artists' Expectations from the System

Agenda:

Artists' Expectations from the System

Follow-up actions

Documents to be brought to the interview:

Nothing

OMMs : Mock Interview Summary 3

System:OMMs(Online Music Management Systems)

Project Reference:

Interviewee: 1) Camila Cabello(**Role Play**)

Designation: Singer

Interviewer: 1) Mayur Pandar **Designation**: OMMs admin

Interviewer: 2) Bhavya Solanki **Designation**: OMMs admin

Date: 06/10/2021 **Time:** 21:00

Duration: 60 minutes **Place:** Online(Google meet)

Purpose of Interview:

Artists' Expectations from the System

- According to Artists', The system should have a high user base who listens to songs.
- System should have high Production companies who make contracts with artists.

OMMs: (Role Play) Interview Plan 4

System: OMMs(Online Music Management System)

Project Reference:

Interviewee: 1) Daniel Ek(Role Play)

Designation: CEO at Spotify

Interviewer: 1) Utsav Ladani

Designation: OMMs admin

Interviewer: 2) Dev joshi

Designation: OMMs admin

Date: 07/10/2021 **Time:** 21:00

Duration: 60 minutes **Place:** Online(Google meet)

Purpose of Interview:

Reputation of business and robustness of the System

Agenda:

To develop Business between company and Owner of the system

Initial ideas

Follow-up actions

Documents to be brought to the interview:

Nothing

OMMs: Mock Interview Summary 4

System: OMMs(Online Music Management System)

Project Reference:

Interviewee: 1) Danieal Ek(**Role Play**)

Designation: CEO at Spotify

Interviewer: 1) Utsav Ladani

Designation: OMMs admin

Interviewer: 2) Dev joshi

Designation: OMMs admin

Date: 07/10/2021 **Time:** 21:00

Duration: 60 minutes **Place:**Online(Google meet)

Purpose of Interview:

Reputation of business and robustness of the System

1. System should provide better privacy to albums or songs of the product company.
2. System should be able to generate high profit for the Production company.
3. System should be able to do all transactions smoothly.

OMMs: (Actual) Interview Plan 5

System: OMMs(Online Music Management System)

Project Reference:

Interviewee: 1) Yash Vasani(Actual)

Designation:Enthusiastic Linux system administrator

Contact Details: 201901081@daiict.ac.in

Organization Details: Student at DAIICT

Interviewer: 1) Utsav Ladani

Designation: OMMs admin

Interviewer: 2) Mayur Pandar

Designation: OMMs admin

Date: 08/10/2021 **Time:** 17:00

Duration: 75 minutes **Place:** Online(Google meet)

Purpose of Interview:

To identify problems and requirements regarding security in the OMMS system.

Agenda:

Security problems in system and any other concerns

Current security procedures

Follow-up actions

Documents to be brought to the interview:

Nothing

OMMs:Actual Interview Summary 5

System: OMMs(Online Music Management System)

Project Reference:

Interviewee: 1) Yash Vasani(**Actual**)

Designation: Enthusiastic Linux System Administrator

Contact Details: 201901081@daiict.ac.in

Organization Details: Student at DAIICT

Interviewer: 1) Ustav Ladani

Designation: OMMs admin

Interviewer: 2) Mayur Panda

Designation: OMMs admin

Date: 08/10/2021 **Time:** 17:00

Duration: 75 minutes **Place:**Online(Google meet)

Purpose of Interview:

To identify problems and requirements regarding security in the OMMS system.

1. System should be robust and privacy oriented.

2. It should stay stable despite any failures.
3. System should be able to generate big profit for production company and Administrator
4. System should provide concurrent access to multiple users.
5. System should be secure and easy to use.

2.2.2 Combined Requirements

- For a good user interface, we should have better UI/UX designers or we should have good knowledge about UI/UX designing.
- We should have a better chief information security officer (CISO) to secure our system and serve good privacy to users.
- We will require a good testing team who can test our system before production.
- We should have good algorithm developers to provide optimized suggestions.

2.3 Questionnaires

2.3.1 Google Form

Questionnaire form is given in the next page. We have created a similar google form to analyze this questionnaire. Google form link is given below
https://forms.gle/qUaxtigovBRwgkQg8

OMMs– Online Music management system survey

Please circle your answers to the following questions:

1. Do you like to listen Music?

   Yes / No

2. How frequent do you listen to music?

   Always / Often / Sometimes / Not frequent / never

3. Which mode do you prefer to listen music?

   Online / Offilne

4. Which online platforms do you prefer to listen music? (Multiple choice available)

   YT Music / Spotify / Jio Saavan / Gaana / Other : _____

5. Have you ever bought premium in any music platform?

   Yes / No

6. Which kind of music do you prefer most? (Multiple choice available)

   Classical / Instrumental / Folk / Rock / Pop / EDM / Bhajan

7. In which languages do you listen the music most?(Multiple choice available)

   English / Hindi / Other : _____

8. Who is your favourite music artist?

---

9. What is your favourite song?

---

10. How much rating would you like to give to your favourite online music platform?

(Strongly Dissatisfied) **1** **2** **3** **4** **5** (Strongly Satisfied)

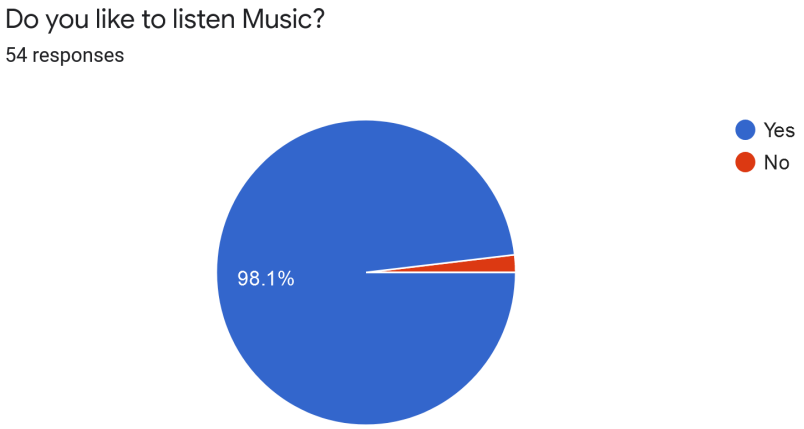
11. Any suggestion about Online Music Management System

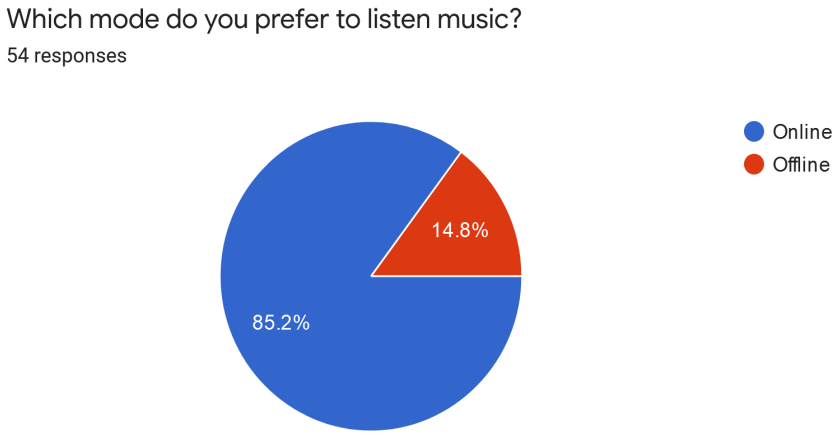
Your Name and Email address

Thank you for completing this questionnaire

2.3.2 Summary

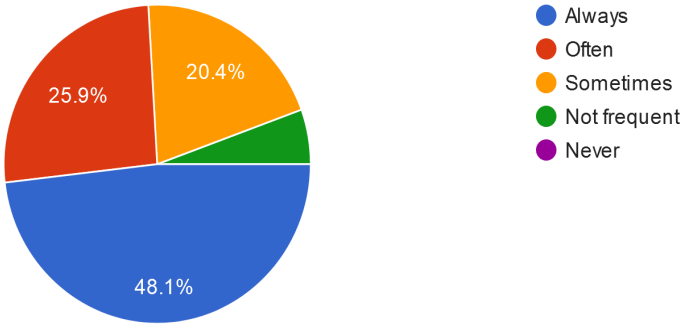
Summary of responses:





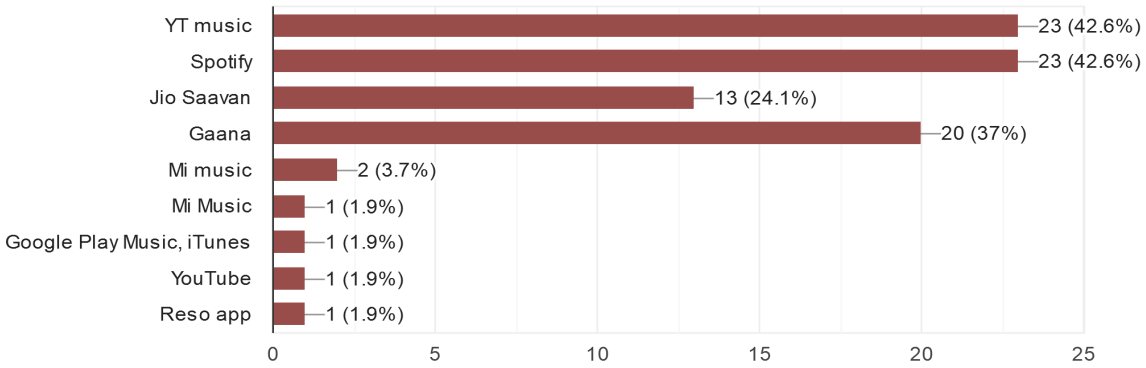
How frequent do you listen to music?

54 responses



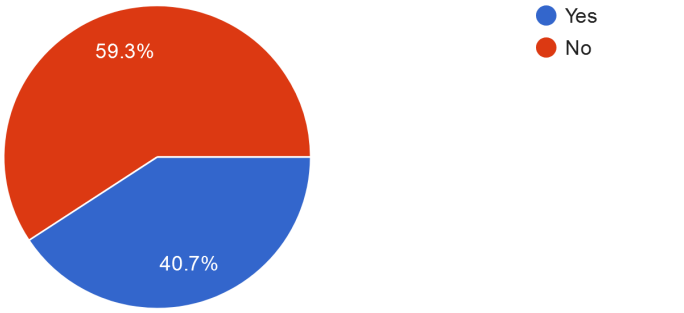
Which online platforms do you prefer to listen music?

54 responses



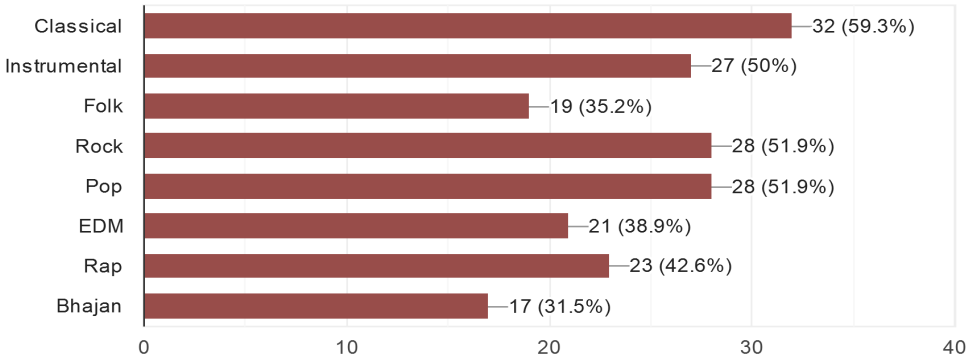
Have you ever bought premium in any music platform?

54 responses



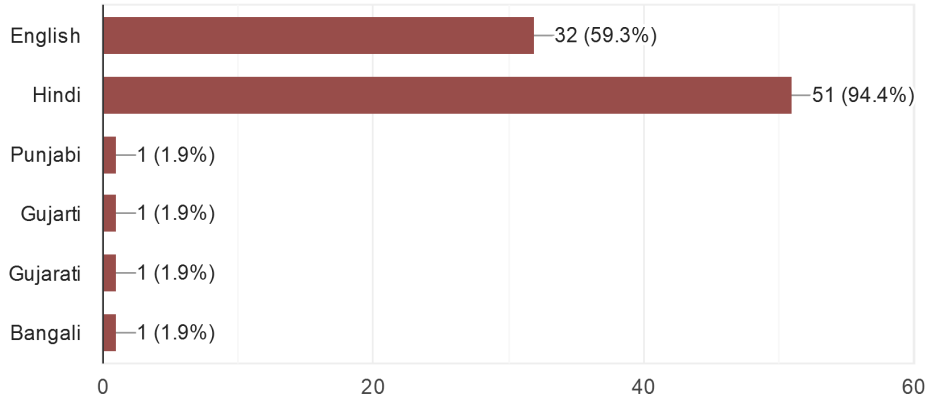
Which kind of music do you prefer most?

54 responses



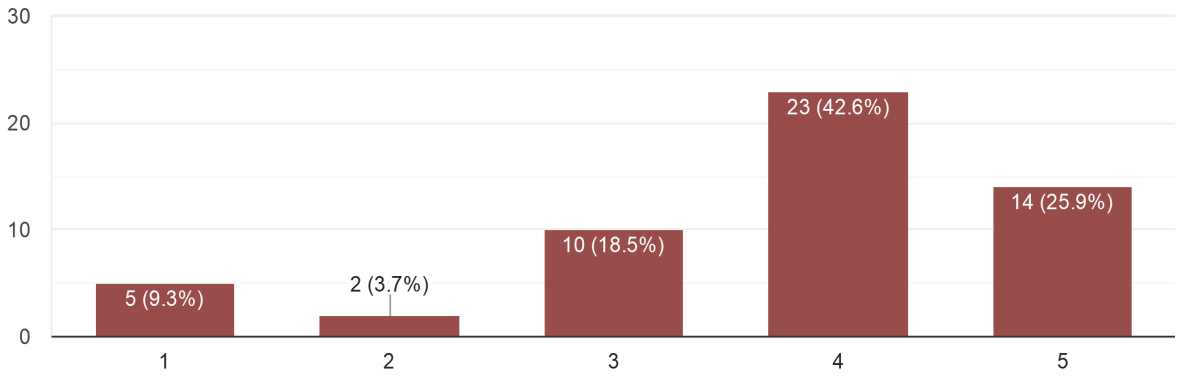
In which languages do you listen the music most?

54 responses



How much rating would you like to give to your favourite online music platform?

54 responses



That's it, These are the summaries for objective questions. For the short answer type question (8 and 9 eg. who is your favorite artist) different responses had different answers for that. For the Description type question (suggestion about Online Music Management System) common answers are written below.

1. There should be an option to make (and recommend) a playlist containing whole albums, instead of different singers.
2. It Should work more on collecting the music from smaller artists.
3. More than you think people love music You should try to reach them all.
4. There should be less ads in the Online Music Management System.
5. Online Music Management Systems should not have any ads.

2.3.3 Combined Requirements

- We will need better ad providing algorithms which provide related ads to users such that we can make profit and more important users will stay happy with the system by seeing related ads.

- We will need a good song suggesting algorithms such that users will use our service for more time.

- We will require some Networking features to stream music on different devices like android TV etc.

2.4 Observations

2.4.1 Observation Summary

Summary from all the Observations is given in the next page.

OMMs: Observations

System :OMMs(Online Music Management System)

Project Reference :

Observations by: Mayur Pandar,Bhavya Soanki, Utsav Ladani, Dev Joshi (All are OMMs admin).

Date: 09/10/2021 **Time:** 11:00

Duration: 60 minutes **Place:**Online(Google Meet)

Observations:

When we were designing our OMMs(Online Music Management System) and when we got a summary from all interviews and when we had seen all responses from google form we came to the following observations.

1. Many online music systems contain authentication handling.

2. Most systems are mainly divided into three parts. 1. User interface 2.Sever 3. Database.

3. To make money many Online Music Management systems are giving ads to users.

4. Some systems provide a feature to connect systems to other devices.

5. Many systems provide privacy to users and have copyright restrictions to premium songs.

2.4.2 Combined Requirements

Combined requirements are mostly written before in Background reading, Interviews, Questionnaires, Rest of the requirements are given below.

- We will need a Great UI design for the User Interface of our system.
- We need to develop a Good Music player which can work in various type of operating systems.
- We will need good hardwares (Good Storage devices to store data and Good server which optimizes queries) for smoothness of our system.
- A very good and robust copyright management system is needed to stop pirating of premium songs.
- Strong Firewalls or any other security providing devices will be needed to Keep our database secure from attackers.
- We have to find a reliable operating system (RHEL (Red Hat Enterprise Linux), CentOS etc.) which can act as a server for queries in our system.
- We will need a reliable and robust Transaction management system for handling transactions.
- At the end We will need a good audience who use our system and will need good investors who help us to improve our system by investing money in our system.

3 Fact FInding Chart

Objective	**Technique**	**subject(s)**	**Time commitment**
To understand various existing systems.	Background reading	Systems' workflows, manual pages, Documentations etc.	2 days
Simple meetings to know users' expectations from the OMMs	Interview	2 User	2 x 1 hours each
Simple meetings to know artists' expectations from the OMMs	Interview	1 artist	1 hours
To develop business between production company and system administrator	Interview	1 CEO	1 hours
To identify security problems in system and their solutions	Interview	1 Enthusiastic security officer / Linux system administrator	1.5 hours

To collect information about system related questions from the users for the system.	Questionnaires	4 information(data) analyzers	2 days to get data + 2 hours for the summary (total)
For continuation of the development of the system	Observation	2 creative admins	0.5 days
To make the SRS document look clear.	Document formatting	1 member	0.5 hours

4 List Requirements

- To check information about users' premium accounts we should have a good authentication handler in our OMMs system.
- For Online transactions between system and users, we will need a better transaction management system.
- We should make our system platform Independent. (For example, our system should work on various Operating systems and various devices.)
- We should have better storage options to store songs in our Database of OMMs.
- We will need a strong server for our OMMs which can communicate with Database nicely.
- We should have a better chief information security officer (CISO) to secure our system and serve good privacy to users.
- We should have good algorithm developers to provide optimized suggestions.
- We will need better ad providing algorithms which provide related ads to users such that we can make profit and more important users will stay happy with the system by seeing related ads.
- A very good and robust copyright management system is needed to stop pirating of premium songs.
- Strong Firewalls or any other security providing devices will be needed to Keep our database secure from attackers.
- For a good user interface, we should have better UI/UX designers or we should have good knowledge about UI/UX designing.
- We will require a good testing team who can test our system before production.
- We will require some Networking features to stream music on different devices like android TV etc.
- We have to find a reliable operating system (RHEL (Red Hat Enterprise Linux), CentOS etc.) which can act as a server for queries in our system.
- At the end We will need a good audience who use our system and will need good investors who help us to improve our system by investing money in our system.

5 User Classes and Characteristics

System has 4 user classes:

1. **Listener:** All users are listeners. They can play songs whatever they want.

2. **Artist:** Singer, lyrics writer, music composer, etc. are artists. They are similar to listeners, but the difference is that they have more information fields.

3. **Production Company:** Include all production companies. They can produce and publish the songs.

4. **Admin:** Owner of the OMMs. Admin approves the song, removes users, etc. But not all privileges. More details in the privileges section.

6 Operating Environment

Hardware: Any smartphone, tablet, pc, laptop which support internet connection

System requirements:

- All systems are capable of using the internet using TCP/IP protocol..
- System have to allot at least 10 KB bandwidth to this application

Platforms :

Windows

- Processor: 1 GHz or faster
- RAM: 1 GB or more
- Hard disk: 4GB or more
- Graphics: DirectX 9 or later
- Version: 7, 8, 8.1, 10, 11

Linux

- Processor: 1 GHz or faster
- RAM: 1 GB or more
- Hard disk: 4GB or more
- Graphics: X window server or similar
- Version: Ubuntu 8 or later, Android 6.2 or later

External interface requirements:

Hardware:

- Good quality of speaker

Software:

- Web browser or Web view with support of ECMAscript 5 or later
- Preinstalled .mp3 player in the operating system, if the application is native application, otherwise web browser if enough.

User Interface:

- Web application use JavaScript with react + material UI
- Android Application use Java or Kotlin to make GUI
- Desktop application use Electron framework with react + material UI
- Design : *TBD*

Communication Protocol:

- HTTP to transfer web pages over internet or handel the API request
- FTP to transfer music file over internet

APIs:

- Google,Facebook,Github API for *SignIn/Signup* purpose.
- Google cloud to host server.
- Google AdSense for ads.
- Google Pay, Paytm, etc. APIs for payment.

7 Product Function

OMMs has three major subsystems:

Client Application

Client application is installed on user's devices. I handle the HTTP request/response, music operations, etc.

Workflow:

- **Register/SignUp/SignIn:** Clients can create a new account using their email id or signup with google, facebook or github account. To create a user, the user sends a request to the server, server verifies information and creates a new user. SignIn can be done using social media accounts or email.

- **Music Player:** Music player uses .mp3 player to play music into the client application. The modern browser provides inbuilt functionality to play music, otherwise we have to add a library according to the platform.

  Music Player functions:
  - Play
  - Pause
  - Next
  - Previous
  - Seek slider

- **Search:** Search the music in the database and send the list of metadata of music to the user. Users select the music and that song plays in the music player. Search user different filters to get a list of specific types of songs. Rating, views, popularity, type, etc can be used as filters.

- **User Profile:** User profile provides the function to change their details, password, premium, etc.

- **Upload Song:** Song can upload only by production Company

- **Money Operations:** Payment transactions are done using this function with security. Check money information, give money to premium accounts,etc. Handled with Money function

Server Application

Server application is hosted on google cloud or any similar hosting platform. It handles the user request and sends responses, manipulating the data into the database, etc. It works as a controller between client and database.

Workflow:

- **Authentication:** Handle all *Register* and *SignIn* requests. Register new user, login user, etc. authentication related requests are handled by authentication function

- **Search Query:** Handle search request. According to the user's query and filter, the function returns the list of metadata of songs.

- **Music Operations:** All music related operations handled by Music Ops. Insert, delete, update music and it's metadata, etc.

- **User Operation:** User profile update and delete using this function.

- **Payment Transaction:** Payment transactions are done using this function with security.

Database System

Database system store and manipulate the music and client information, make the system secure, manage the user access rules, etc.

Workflow:

- **User:** Insert, update and delete user are done using this function

- **Music:** Insert, update and delete song are done using this function

- **Money:** All transection update done using this function

8 Privileges

1. Listener

   Privileges:
   - Create, update and delete their account
   - Search song with filters
   - Give rating to particular song
   - Switch between free and premium account

   ***Who*:** All user

2. Artist

   ***Privileges*:**

- Create, update and delete their account
- Search song with filters
- Give rating to particular song
- Switch between free and premium account

Who**:** All singer, lyrics writer, music composer, etc.

3. Production Company

Privileges**:**

- Update, delete and publish music
- Withdraw money

Who**:** Production companies who are producing music like T-series, Sony, etc.

4. Admin

Privileges**:** Usually admin have all privileges, but in our system, admin do not have all privileges.

- Insert, update, delete any user, any song and any detail except money, because money transactions are automated.
- Approve song publish request from production company

Who**:** Group of few people like system owner, manager and staff of company

All money transactions are automatic, so there is no need for any money related role. Some functions are not included because that operations are automated so no privileges required.

9 Assumption

- All money-related transactions are automated and done via an external payment gateway (APIs)
- Artist is not publishing music, they make music for a production company, only production company publishing music
- Admin has to approve all songs before publishing
- The user's device must fulfill all hardware and software requirements.

10 Business Constraints

- A great product always needs enough time, expert developers, and sufficient money. If the budget of a client company is low, then product quality is low because deduct in money explicitly leads to deduct in developer quality. So the application may or may not be scalable and too buggy.
- Time is an important factor. Good amount of time gives a good product. If a client company has less time to release then the product may not be perfect.

- If developers haven't good skills then hire the new developer with high expertises may costly, so company must give some time to learning new technology to their developers

- It is required to teach each function of the system to the client company so the system works perfectly, so it takes extra time.
  Hence a balanced combination of *money, time* and *expertise* is important.

- A production company is paid according to views of music

- System owners earn money in two popular way
  (1) Visual Ads between 2-3 music
  (2) Ads-free premium account

- Almost all functions are automated, so avoid hiring more staff.
  Financial condition for profit: *Staff salary + money pay to Production company + system hosting service change + other < Money earned by Ads + paid promotion of songs.*

- Song approval is approved as soon as possible.

- Any system failure must be resolved very quickly so user experience can not affect.

- Use social media marketing techniques to reach out to more users.

Section2: Noun Verb Analysis

1. Final Problem Description

The Online Music Management System will provide a large collection of songs to every user who is connected with our service over the internet. Users can search music and/or select from a given list of songs and that song will play in the appropriate app or system (web / android / desktop). As far as the IT214 subject is concerned, this project will be database oriented. Our goal is to make efficient database design, so that maximum users can connect with our service using minimum resources.

The system consist of three parts;

1. **User interface:** installed on device or accessed via browser
2. **Server:** connect application and database, handle user request, stream song, register user,etc.
3. **Database:** Store the songs and user information, provide some basic functions to modify and search into the database. It uses a PostgreSQL database.

Basic Workflows of Client Application

- **Register/SignUp:** Every user needs to register in application when he/she operates for the first time.Without registering users can listen to only 20 songs.

- **SignIn:** Every time when a user opens an application, he/she must have to sign in using their credentials.

- **Suggestion/Song List:** After signing in, users get the list of some song according to their preference and history.

- **Search Bar:** Search the song in our database using various filters like artist name, rating, views, type, date, movies, etc.

- **Music Player:** Play, pause, prev, next, queued, etc. functionalities are handled by the music player.

- **User Profile:** show the information about user, account type(premium/comman), update profile, etc.

- **Upload Song:** Users can upload songs on the server.

- **Money:** Give the money to artists according to views and rating.

Basic Workflows of Server

- **Authentication Handling:** all authentication related functionality like register user, check whether the user's credential is right or not, etc. are done by this part.

- **Search request:** Send the result of the search query to the application in JSON format.

- **Song request:** Start the streaming of songs on a particular user's device, download, upload can be done via this service.

- **Update:** Update song rating, views,etc. and user data also.

- **Money:** All money transactions controlled by this part.

Basic Workflow of Database

- **User:** Add, delete, update user and also update account to premium account.
- **Song:** Add, delete, update song.
- **Money:** Money transaction.

Popular Issues

- **UI/UX:** Modern apps must have to provide a customization of theme and layout because bright light can harm eyes at night and UI with dark color can not be visible in day. So instance switching of themes was needed. Different layouts give the diversity of system usage.
- **Suggestion:** Users don't want the same type of suggestion all the time. They want to customize the suggestion based on their own requirements. No any existing system have this kind of feature to customize the suggestion for each user according to user requirements
- **Filter:** Only Suggestions are not enough, people must need some filter to get a particular song list. All music platforms are providing different filters, but it's not sufficient. Users want to filter the songs according to the group of attributes. So I need some sort of feature to apply more than one filter on any list of songs, even in a playlist, albums ,search result, etc.
- **Offline Availability:** Copyright concern prohibits the user to download songs. Many places like gym, car on highway, train, flight etc. have low connectivity due to closed infrastructure or mobility of devices. In such cases users want the offline availability of songs. There is some smart system required so that users can download songs without violating copyrights.
- **Other:** Playlists are too important to feature in any music system. System must take care of this feature. Users should be able to share songs with some privileges. Users also want to gather online and organize a virtual party (party means one can play a song and other listen to it), podcast, etc. So the system adds these features so that users get a better experience.

Online Music Management System(OMMS) will help to categorize using song name, artist, albums, rating, type, time, type, number of listener, etc. Admin will be able to add songs. Users will be able to create their favourite playlists, Download and Share songs. This system also has features for singers and authors of albums. If some other singer wants to use a song or it's lyrics in his song, He will have to take permission from the owner of that song. This system will alert the owner of a song or album when someone publishes a song or album containing the song of the owner. Based on that if he has not taken permission from the owner he will be charged a penalty. We are providing **Repeat Current song, Repeat playlist, Shuffle playlist** like features for users. Users can make their own playlist by adding their favourite songs. The system also has a **play in background** feature so a user doesn't have to keep the application open while listening to music. To use this play in background feature the user has to buy premium.To set any song for caller tune(song for incoming call) the user has to buy premium.

There are several other applications in the market which are similar to our OMMS software.these applications/softwares provide several functionalities like search music,add different songs in the favourite song list,set caller tune,lyrics,etc.

Search Music: By this function you can search the music song whichever you want.

Favourite song list:some applications provide a Favourite song library/folder and in this library you can add the song so it notes the song to your favourite song.

Caller tune:this function provides the ability to set the song for incoming calls.

Lyrics:some other applications provide lyrics of songs.

To achieve this workflow in our system, We will need to design a database which includes several entities having relations internally. For users, we will need tables storing information about the user, about the premium accounts, about user credentials, past transection of users etc. For the song, we will need tables storing various information about the song like song type (pop, rock, rap, bhajan etc.), song language, album name, singer, whether the song is for premium users or not, artist name, premium price, views, likes, dislikes, rating etc. for the singers and artists we will need tables storing their song names, their albums, their details, production company name etc. For the production company we will need tables storing information about songs or albums they have released, artist or singer names of that song/album, price of that song/album etc. For the admin of the system we will need several functions for adding songs in system, deleting songs in system, updating details of songs etc. We will need transaction management in our system.

For the users, We will need different sets which store information about the user. User table will have attributes like User ID, User Name, Email ID, Mobile number, etc. We will also need to define a set which tells which users have premium accounts and which have not. It will have attributes like User ID, User Name, Premium Details, Premium time etc. We will also have to store the privileges (like view add, delete, update, edit etc.) for different users(not listener only) of the system. User and Premium user table will define a relationship named Premium user account.

For the Artists, We will need different sets which store information about the artists. We will need a table which will store information about the artists which will have attributes like Artist ID, Artist Name, Produced songs etc. We will define a set named album which will store albums present in the system which will have attributes like album name, artist name, artist ID, album ID, Count of songs present in a album etc. We will need a set which will store the rating of the album which will have attributes like Album ID, Album rating etc. We will define a relationship between artist and album which will give us a relation named Artist's Album.

We will define several sets to store the information about the production company. We will define a set named production company which stores information about the company. It will have attributes like company name, company ID, produced albums, employers(artists). We will define a relation between Artists and the production companies which will give us a relationship named Job details. We will also need to store information about the profit which each produced album made for the production company. It will have attributes like company name, company ID, album name, album ID, Profit etc.

Music is the main component of our system. We should handle music related things properly. To store the Music, we will need a set named Music. It will have attributes like Music name, Music ID, Artist name, Artist ID, company name, Company ID etc. To store different languages of the music we will need a set named Music language.It will have attributes like Music ID, Music name, Music language etc. To store the type of the music (rap, rock, pop, bhajan etc.) we will need a set named Music Type. It will have attributes like Music ID, Music name, Music Type etc. All these sets should be connected to each other for the query optimization because the listener of our system will search music based on various things. To store whether a song is for premium users or not, we will create a set named premium songs which will have attributes like Music ID, Music Name. To store views and likes of the songs

we will need a set which will store likes and views of the songs. It will have attributes like Music ID, Music name, Views, Likes, Ratings etc.

Users of our system expect different premium plans from the system. So we will also need sets which store information about plan money and plan start time, plan end time (in short plan duration). Premium details and the premium of the user will create a relationship named the premium duration of the user.

For the admins we will need a set which stores information about the admin which will have attributes like Admin name, Admin ID etc. Admin will be able to add, remove, edit albums and songs in our system. Admin will have total control over the whole system.

Privacy of the users is the main of our system. We will apply an authentication system in our system. So for that, we will need a set which will store credentials of the users. It will have attributes like User email ID, User name, User password, mobile number, etc. Whenever a user registers it will store his credentials in the system and whenever user logins into the system, it will try to match stored credentials and entered credentials.

Users need to make the transaction to purchase the premium. So we will apply a transaction management system. It will need a set which stores transactions in the system. We will provide our users various types of transactions because users may want to purchase the premium using a credit card, debit card, UPI etc. It will have attributes like Transaction ID, Transaction money, User ID, Company ID, Transaction type (Only online transactions are applicable) etc. Transaction management system will also be related to the production company because the transaction generally will happen between the user and production company.

We will also need to define some functions and some trigger functions for the system's correct and optimized working. For example when users login we will need to define a function which compares entered credentials with the stored credentials. We can apply a trigger function when a non premium user tries to access the premium songs.

2. Noun (& verb) analysis Tables

1. Noun & analysis

Nouns	**Verbs**
User	
Premium User	
Music	
Music language	Music
Music type	
Premium information	Premium songs
Artist	
Production company	

Admin	
Job	
Transaction	
Album	
Album Rating	
Premium time	
Premium detail	
online	System
large collection	Songs
users	User
maximum users	User
minimum resources	System
server	System
user request	Query
store	Database
user information	User
client application register/signup sign in	Authentication
search	Query, function
user profile	Details
Account type	Premium Accounts
update profile	Query
upload	Add
authentication handling	System
Password	Authentication
money transactions	Payments
update user	Query

update account	Query
premium account	Premium
update song	Query
Music Name	Music
past transaction	Past payments
ID	
bhajan	Music Type
Album name	Album
Artist name	Singer of Music
production company name	Production company
Plan Price	Money
User id	User
User name	User
Email ID	User
Mobile Number	User
edit	query
Count album	Songs in a album
Album id	Artist’s Album
Profit	Company’s profit
Artist ID	artist
Company ID	Album’s Company
Music ID	Music’s Album
query optimization	Optimization
Type	
premium songs	Premium
views	Music

likes	Music
ratings	Music
different premium plans	Plan details
plan money	Money
plan duration	Starting and ending time of a plan
premium duration	Plan duration
Admin id	Admin
edit albums	Query
total control	Admin
privacy	feature
authentication system	System
store credentials	database
user registers	Registration
transaction management system	system
credit card	Transaction Type
debit card	Transaction Type
Transaction ID	Payment details
online transactions	Mode
large collection	collection
appropriate app	application
web / android / desktop	platform
efficient database design	Designing
stream song	streaming
basic functions	functions
workflows	working
client application register/signup	

various filters	
music player	
json	format
postgresql	Tool
ui/ux	User interface
offline availability	
copyright	
omms	System
software	
singer names	
Plan Duration	
different users	
server authentication handling	
store information	
produced	
production companies	
Company name	
main component	
different languages	Languages
trigger function	
Plan Name	
Total song made	Artist
Base salary	Job
Admin Name	Admin
Plan ID	Premium plans
Transaction Mode	Transactions

Sender Type	Transactions
ReceiverType	Transactions
Price	

2. Accepted Noun and Verbs list

Candidate entity set	**Candidate attribute set**	**Candidate relationship set**
User	User ID	
	User Name	
	Password	
	Email ID	
	Mobile Number	
	User Type	
Artist	Artist ID	Job
	Artist Name	
	Password	
	Total song made	
Album	Album ID	Artist's Album
	Album Name	
	Artist ID	
	Album Rating	
	Company ID	
	Profit	Companies Profit
Production Company	Company ID	
	Company Name	
	Base salary	Job
Music	Music ID	Music's Album

	Music Name	
	Music Language	
	Music Type	
	Artist ID	Singer
	Company ID	
	Views	
	Likes	
Admin	Admin ID	
	Admin Name	
	Password	
Premium Details	Plan ID	Plan
	Plan Name	
	Plan Duration	
	Plan Price	
Transaction	Transaction ID	
	Transaction Mode	
	Sender ID	
	Receiver ID	
	Sender Type	
	Receiver Type	
	Price	

3. Rejected noun and verbs list

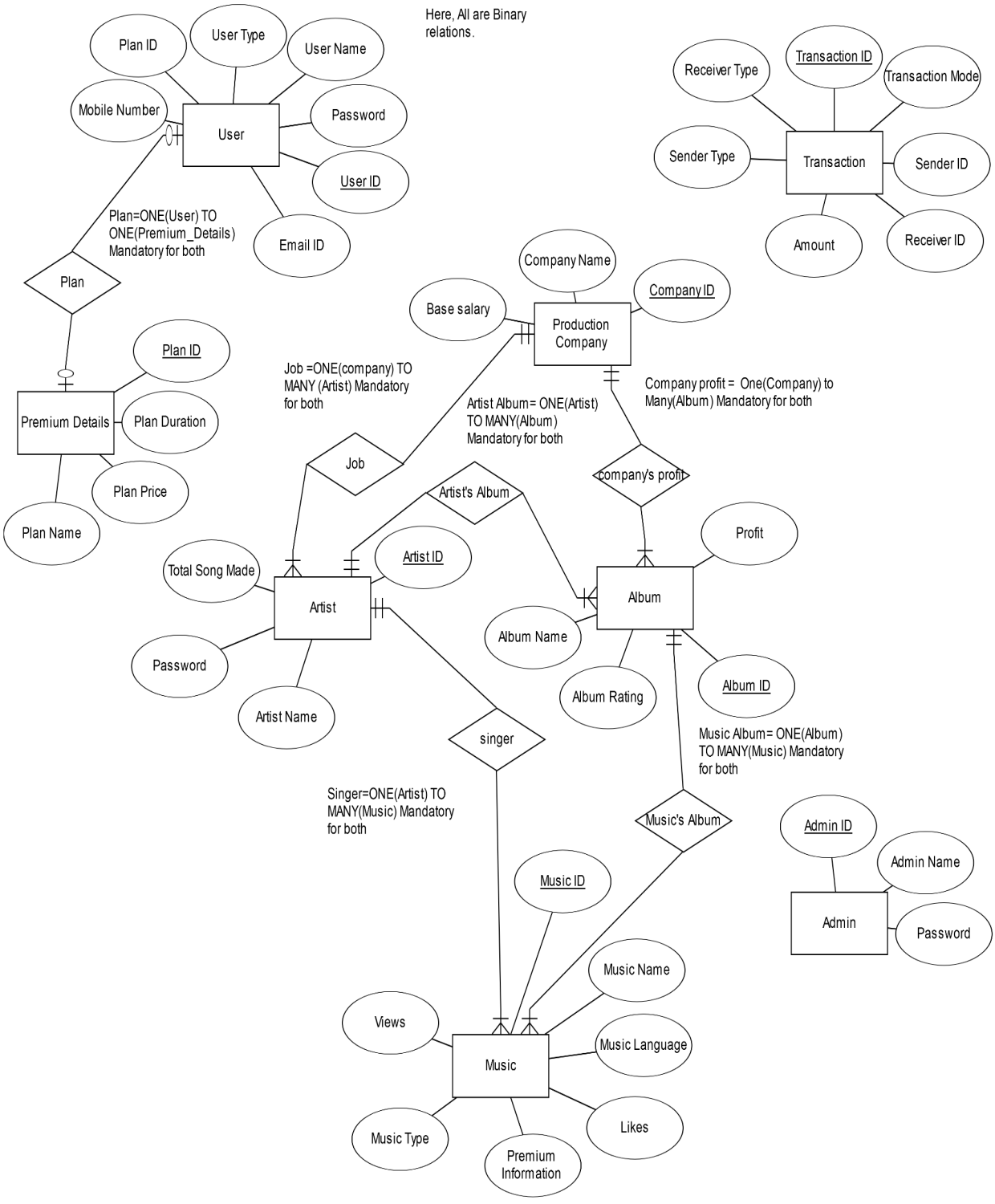
Noun	Reject Reason
trigger function	**Vague**
different languages	**General**
main component	**General**
company name	**Duplicate**

production companies	**Duplicate**
produced	**Vague**
store information	**General**
server authentication handling	**General**
different users	**Vague**
ID	**Attributes**
singer names	**Duplicate**
software	**General**
omms	**General**
copyright	**General**
Type	**Attributes**
offline availability	**Irrelevant**
ui/ux	**Irrelevant**
postgresql	**General**
json	**Irrelevant**
music player	**General**
various filters	**General**
client application register/signup	**General**
workflows	**General**
basic functions	**Vague**
stream song	**Irrelevant**
efficient database design	**General**
web / android / desktop	**General**
appropriate app	**General**
online transactions	**Duplicate**
debit card	**General**
credit card	**General**
transaction management system	**General**

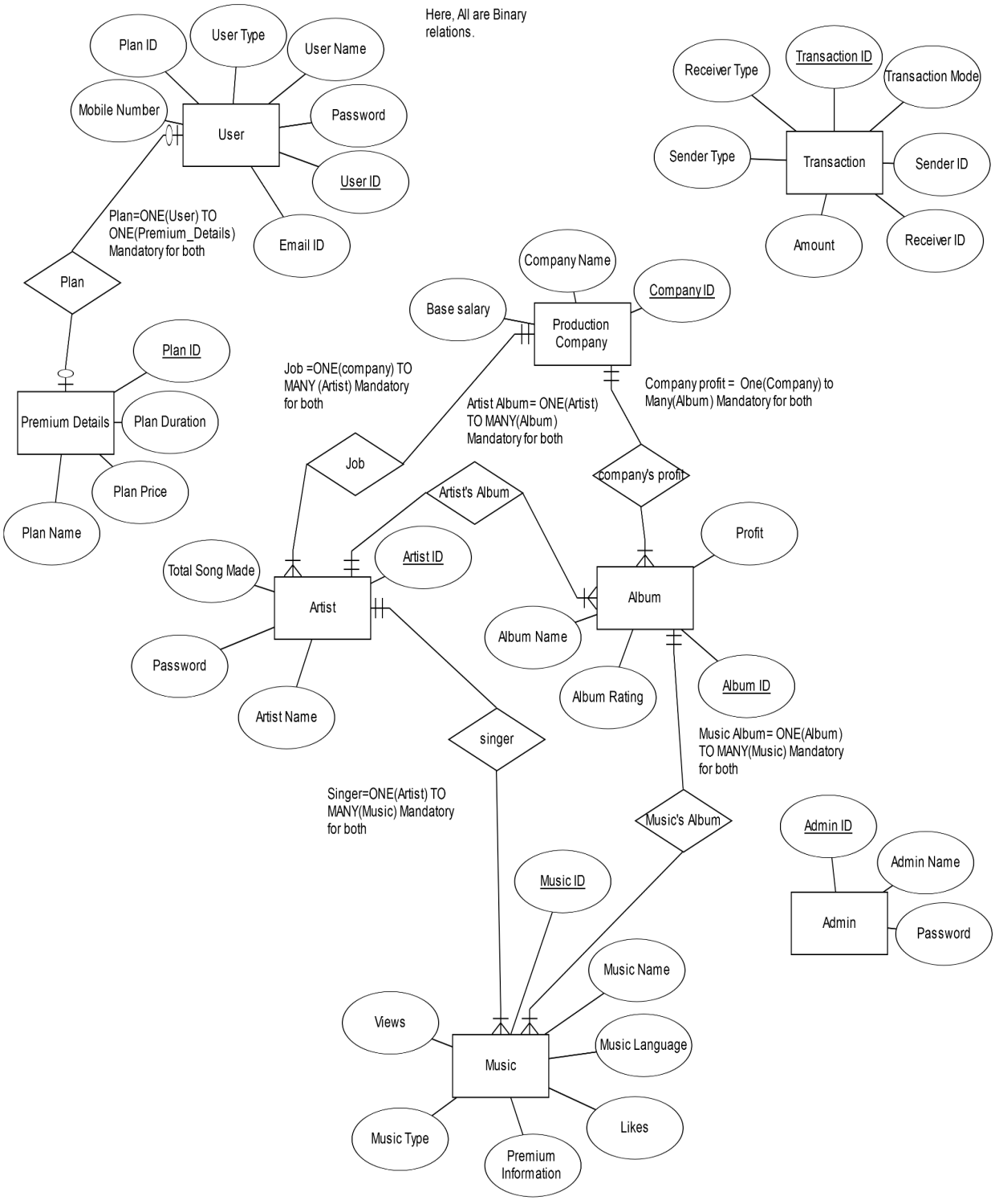
Premium time	**Duplicate**
online	**General**
large collection	**Vague**
users	**Duplicate**
maximum users	**Duplicate**
minimum resources	**General**
server	**General**
user request	**General**
store	**General**
user information	**Duplicate**
client application register/signup sign in	**Duplicate**
search	**General**
user profile	**Duplicate**
update profile	**General**
upload	**General**
authentication handling	**General**
money transactions	**General**
update user	**Duplicate**
update account	**Duplicate**
premium account	**Duplicate**
update song	**General**
past transaction	**Duplicate**
bhajan	**Vague**
production company name	**General**
edit	**General**
Count album	**Duplicate**

query optimization	**General**
premium songs	**Duplicate**
ratings	**Duplicate**
different premium plans	**General**
plan money	**Duplicate**
plan duration	**Duplicate**
premium duration	**Duplicate**
edit albums	**General**
total control	**Irrelevant**
privacy	**Association**
authentication system	**Duplicate**
store credentials	**General**
user registration	**Duplicate**
Authentication	**General**
Premium User	**Attribute**
Premium Song	**Attribute**

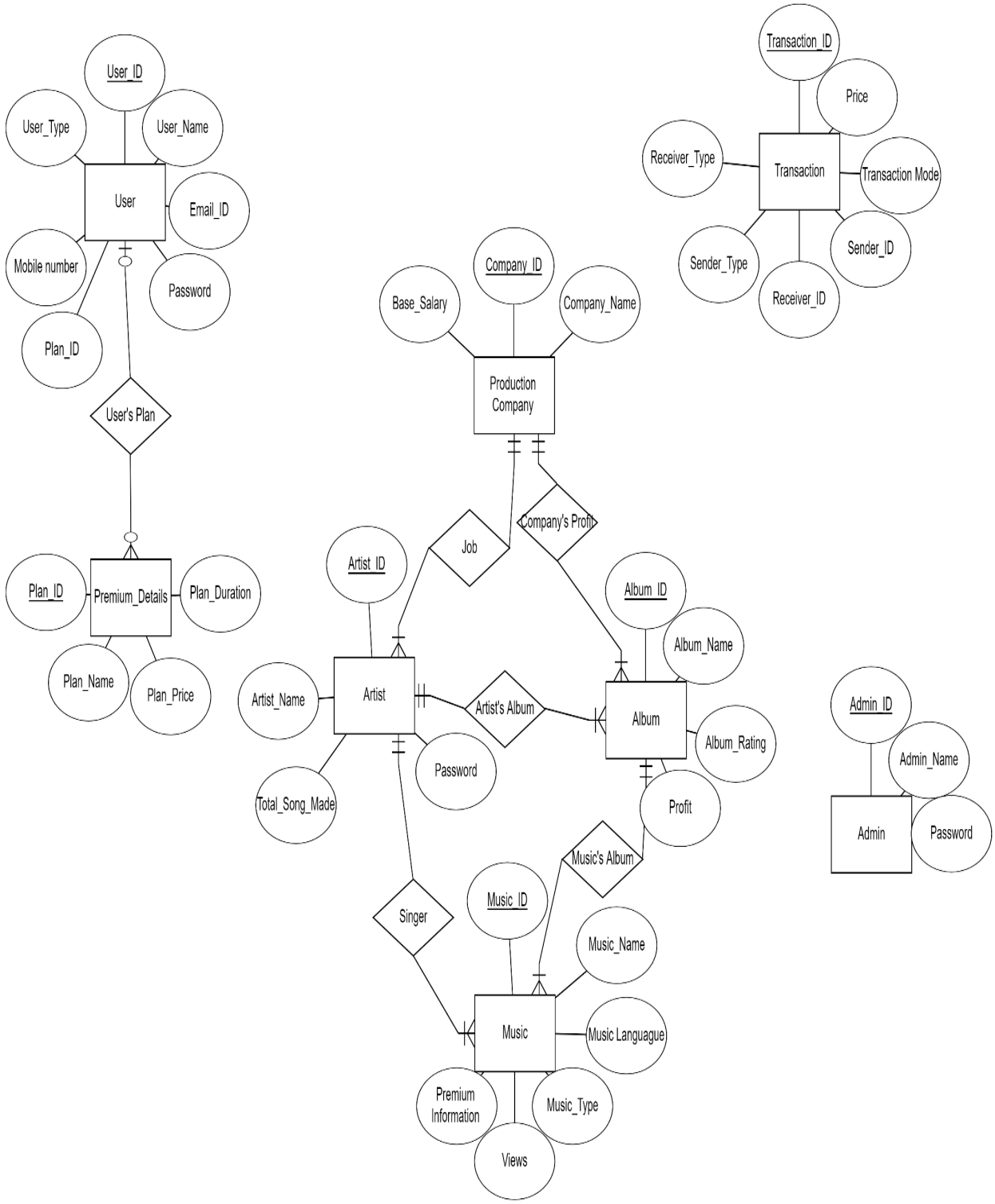
3. Version 1 of ER diagram



4. Version 2 of ER diagram

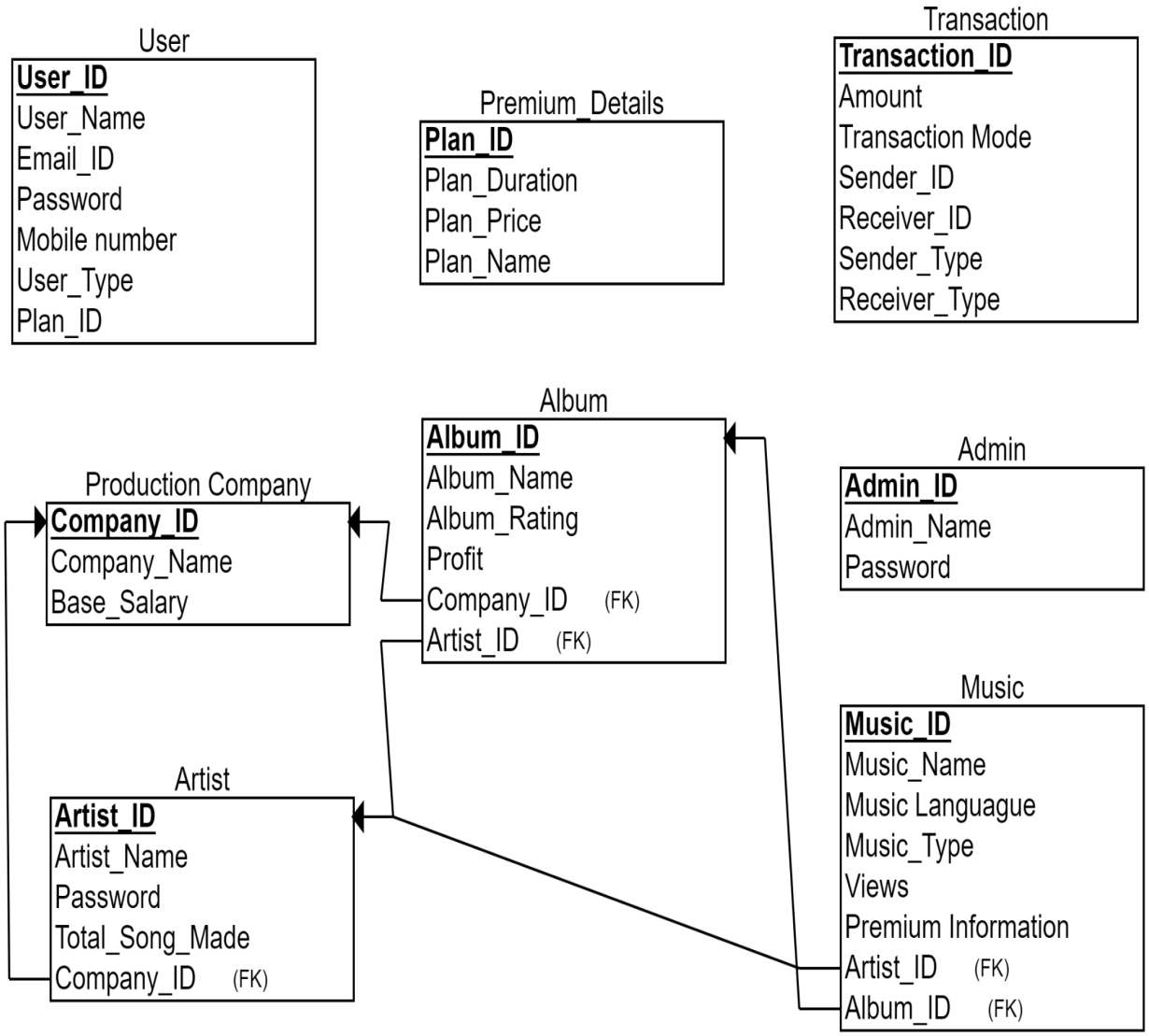


5. Final Version of ER Diagram

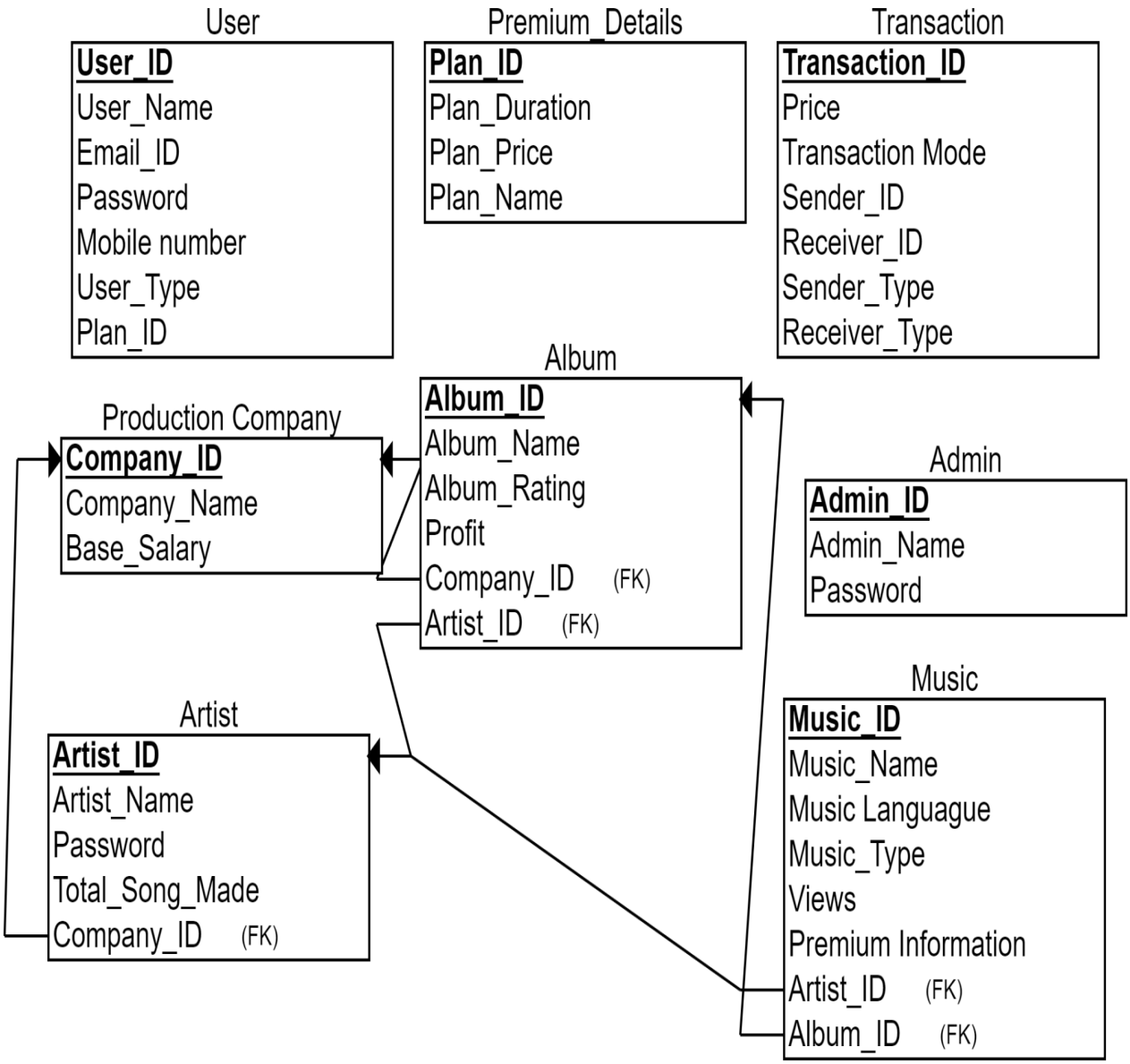


Section4: Conversation of Final ER-Diagram to Relational Model

1. Mapping ER-Diagram to Relational Model Version 1



2. Mapping ER-Diagram to Relational Model Final Version



3. Schema of Relation

1. **User** (***User ID***, User Name, Password, Email ID, Mobile Number, Plan ID, User Type)
2. **Music** (***Music ID***, Music Name, Music Language, Premium Information, Views,Likes, Music Type, Artist ID, Album ID)
3. **Artist (*Artist ID***, Artist Name, Password, Total song made, Company ID)
4. **Album** (***Album ID***, Album Name, Album rating, Profit, Artist ID, Company ID)
5. **Production Compnay** (***Company ID***, Company Name, Base Salary)
6. **Admin** (***Admin ID***, Admin Name, Password)
7. **Plan** (***Plan ID***, Plan Name, Plan Duration, Plan Price)
8. **Transaction** (***Transaction ID***, Transaction Mode, Sender ID, Receiver ID, Sender Type, Receiver Type,Price)

4. DDL Scripts

```
CREATE TABLE "User" (
 "User_ID" BIGINT NOT NULL,
 "User_Name" VARCHAR(100) NOT NULL,
 "Password" VARCHAR(128) NOT NULL,
 "Email_ID" VARCHAR(100) NOT NULL,
 "Mobile_Number" CHAR(10) NOT NULL,
 "Plan_ID" INT NOT NULL,
 "User_Type" VARCHAR(60) NOT NULL,
 PRIMARY KEY ("User_ID"),
 FOREIGN KEY ("Plan_ID") REFERENCES "Premium_Details"("Plan_ID")
);

CREATE TABLE "Production_Company"
(
 "Company_ID" BIGINT NOT NULL,
 "Company_Name" VARCHAR(100) NOT NULL,
 "Base_salary" BIGINT NOT NULL,
 PRIMARY KEY ("Company_ID")
```

```
);

CREATE TABLE "Admin"
(
  "Admin_ID" BIGINT NOT NULL,
  "Admin_Name" VARCHAR(100) NOT NULL,
  "Password" VARCHAR(128) NOT NULL,
  PRIMARY KEY ("Admin_ID")
);

CREATE TABLE "Premium_Details"
(
  "Plan_ID" BIGINT NOT NULL,
  "Plan_Name" VARCHAR(100) NOT NULL,
  "Plan_Duration" INT NOT NULL,
  "Plan_Price" INT NOT NULL,
  PRIMARY KEY ("Plan_ID")
);

CREATE TABLE "Artist"
(
  "Artist_ID" BIGINT NOT NULL,
  "Artist_Name" VARCHAR(100) NOT NULL,
  "Password" VARCHAR(128) NOT NULL,
  "Total_Song_Made" INT NOT NULL,
  "Company_ID" BIGINT NOT NULL,
  PRIMARY KEY ("Artist_ID"),
  FOREIGN KEY ("Company_ID") REFERENCES "Production_Company"("Company_ID")
);

CREATE TABLE "Album"
(
  "Album_ID" BIGINT NOT NULL,
  "Album_Name" VARCHAR(100) NOT NULL,
  "Album_Rating" BIGINT NOT NULL,
  "Profit" BIGINT NOT NULL,
  "Artist_ID" BIGINT NOT NULL,
  "Company_ID" BIGINT NOT NULL,
```

```
 PRIMARY KEY ("Album_ID"),
 FOREIGN KEY ("Artist_ID") REFERENCES "Artist"("Artist_ID"),
 FOREIGN KEY ("Company_ID") REFERENCES "Production_Company"("Company_ID")
);

CREATE TABLE "Music"
(
 "Music_ID" BIGINT NOT NULL,
 "Music_Name" VARCHAR(100) NOT NULL,
 "Music_Language" VARCHAR(100) NOT NULL,
 "Premium_Information" BOOLEAN NOT NULL,
 "Views" BIGINT NOT NULL,
 "Likes" BIGINT NOT NULL,
 "Music_Type" VARCHAR(60) NOT NULL,
 "Artist_ID" BIGINT NOT NULL,
 "Album_ID" BIGINT NOT NULL,
 PRIMARY KEY ("Music_ID"),
 FOREIGN KEY ("Artist_ID") REFERENCES "Artist"("Artist_ID"),
 FOREIGN KEY ("Album_ID") REFERENCES "Album"("Album_ID")
);

CREATE TABLE "Transaction"
(
 "Transaction_ID" BIGINT NOT NULL,
 "Transaction_Mode" VARCHAR(20) NOT NULL,
 "Sender_ID" BIGINT NOT NULL,
 "Receiver_ID" BIGINT NOT NULL,
 "Sender_Type" VARCHAR(20) NOT NULL,
 "Receiver_Type" VARCHAR(20) NOT NULL,
 "Price" BIGINT NOT NULL,

 PRIMARY KEY ("Transaction_ID")
);
```

Section5: Normalization and Schema Refinement

Original Design of the Database

1. **User** (***User_ID***, User_Name, Password, Email_ID, Mobile_Number, Plan_ID, User_Type)
2. **Music** (***Music_ID***, Music_Name, Music_Language, Premium_Information, Views,Likes, Music_Type, Artist_ID, Album_ID)
3. **Artist (*Artist_ID***, Artist_Name, Password, Total_song_made, Company_ID)
4. **Album** (***Album_ID***, Album_Name, Album_rating, Profit, Artist_ID, Company_ID)
5. **Production Company** (***Company_ID***, Company_Name, Base_Salary)
6. **Admin** (***Admin_ID***, Admin_Name, Password)
7. **Plan** (***Plan_ID***, Plan_Name, Plan_Duration, Plan_Price)
8. **Transaction** (***Transaction_ID***, Transaction_Mode, Sender_ID, Receiver_ID, Sender_Type, Receiver_Type ,Price)

List of All Dependencies

1. **User** (***User ID***, User_Name, Password, Email_ID, Mobile_Number, Plan_ID, User_Type)

   - ➔ **Primary Key : ** User_ID
   - ➔ **Foreign Key : ** Plan_ID

   - ➔ **Functional Dependencies:**

     - ❖ **User_ID→**
       User_Name, Password, Email_ID, Mobile_Number,Plan_ID, User_Type

     - ❖ **Email_ID→**
       User_ID, User_Name, Password, Mobile_Number, Plan_ID, User_Type.

     - ❖ **Mobile_Number→**
       User_ID, User_Name, Password, Email_ID, Plan_ID, User_Type

2. **Music** (***Music_ID***, Music_Name, Music_Language, Premium_Information, Views,Likes, Music_Type, Artist_ID, Album_ID)

   - ➔ **Primary Key : ** Music_ID
   - ➔ **Foreign Key : ** Artist_ID, Album_ID

➔ **Functional Dependencies:**

- **Music_ID→**
  Music_Name, Music_Language, Premium_Information, Views,Likes, Music_Type, Artist_ID, Album_ID

3. **Artist (*Artist_ID*, Artist_Name, Password, Total_song_made, Company_ID)**

➔ **Primary Key :** Artist_ID
➔ **Foreign Key :** Company_ID

➔ **Functional Dependencies:**

- **Artist_ID→**
  Artist_Name, Password, Total_song_made, Company_ID

4. **Album (*Album_ID*, Album_Name, Album_rating, Profit, Artist_ID, Company_ID)**

➔ **Primary Key :** Album_ID
➔ **Foreign Key :** Artist_ID, Company_ID

➔ **Functional Dependencies:**

- **Album_ID→**
  Album_Name, Album_rating, Profit, Artist_ID, Company_ID

5. **Production Company (*Company_ID*, Company_Name, Base_Salary)**

➔ **Primary Key :** Company_ID
➔ **Foreign Key :** ----------

➔ **Functional Dependencies:**

- **Company_ID→**
  Company_Name, Base Salary

6. **Admin (*Admin_ID*, Admin_Name, Password)**

➔ **Primary Key :** Admin_ID
➔ **Foreign Key :**

➔ **Functional Dependencies:**

- **Admin_ID→**
  Admin_Name, Password

7. **Plan** (***Plan_ID***, Plan_Name, Plan_Duration, Plan_Price)

   ➔ **Primary Key :** Plan_ID
   ➔ **Foreign Key :**

   ➔ **Functional Dependencies:**

   ❖ **Plan_ID→**
   Plan_Name, Plan_Duration, Plan_Price

8. **Transaction** (***Transaction_ID***, Transaction_Mode, Sender_ID, Receiver_ID, Sender_Type, Receiver_Type, Price)

   ➔ **Primary Key :** Transaction_ID
   ➔ **Foreign Key :** ----------

   ➔ **Functional Dependencies:**

   ❖ **Transaction_ID→**
   Transaction_Mode, Sender_ID, Receiver_ID, Sender_Type, Receiver_Type,Price

Anomalies and redundancy:

- There are no anomalies and redundancy in our schema.

Normalize the database up to 1NF

1. **User** (***User_ID***, User_Name, Password, Email_ID, Mobile_Numb9er, Plan_ID, User_Type)

- None of the attributes User_Name, User_ID, Password, Email_ID(assume Unique(1per User)), Mobile_Number(assume Unique(1per User)), Plan_ID,User_Type are multivalued. So, **User** is already normalized to 1NF.

2. **Music** (***Music_ID***, Music_Name, Music_Language, Premium_Information, Views,Likes, Music_Type, Artist_ID, Album_ID)

- **Music** is already normalized to 1NF because None of the attributes Music_ID, Music_Name, Music_Language, Premium_Information, Views,Likes, Music_Type, Artist_ID, Album_ID are multivalued.

3. **Artist (*Artist_ID***, Artist_Name, Password, Total_song_made, Company_ID)

- None of the attributes Artist_ID, Artist_Name, Password, Total_song_made, Company_ID are multivalued. So, **Artist** is already normalized to 1NF.

4. **Album** (***Album_ID***, Album_Name, Album_rating, Profit, Artist_ID, Company_ID)

- **Album** is already normalized to 1NF because None of the attributes Album_ID, Album_Name, Album_rating, Profit, Artist_ID, Company_ID are multivalued.

5. **Production Company** (***Company_ID***, Company_Name, Base_Salary)

- None of the attributes Company_ID, Company_Name, Base_Salary are multivalued. So, **Production Company** is already normalized to 1NF.

6. **Admin** (***Admin_ID***, Admin_Name, Password)

- **Admin** is already normalized to 1NF because None of the attributes Admin_ID, Admin_Name, Password are multivalued.

7. **Plan** (***Plan_ID***, Plan_Name, Plan_Duration, Plan_Price)

- None of the attributes Plan_ID, Plan_Name, Plan_Duration, Plan_Price are multivalued. So, **Plan** is already normalized to 1NF.

8. **Transaction** (***Transaction_ID***, Transaction_Mode, Sender_ID, Receiver_ID, Sender_Type, Receiver_Type, Price)

- **Transaction** is already normalized to 1NF because None of the attributes Transaction_ID, Transaction_Mode, Sender_ID, Receiver_ID, Sender_Type, Receiver_Type, Price are multivalued.

Normalize the database to 2NF

1. **User** (***User_ID***, User_Name, Password, Email_ID, Mobile_Number, Plan_ID, User_Type)

- **User** is already in 1NF and also no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table (Hence no Partial Dependency). So **User** is already normalized to 2NF.

2. **Music** (***Music_ID***, Music_Name, Music_Language, Premium_Information, Views,Likes, Music_Type, Artist_ID, Album_ID)

- **Music** is already normalized to 2NF because it is already normalized to 1NF and also no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table (Hence no Partial Dependency).

3. **Artist (*Artist_ID***, Artist_Name, Password, Total_song_made, Company_ID)

- **Artist** is already in 1NF and also no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table (Hence no Partial Dependency). So **Artist** is already normalized to 2NF.

4. **Album** (***Album_ID***, Album_Name, Album_rating, Profit, Artist_ID, Company_ID)

- **Album** is already normalized to 2NF because it is already normalized to 1NF and also no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table (Hence no Partial Dependency).

5. **Production Company** (***Company_ID***, Company_Name, Base_Salary)

- **Production Company** is already in 1NF and also no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table (Hence no Partial Dependency). So **Production Comapany** is already normalized to 2NF.

6. **Admin** (***Admin_ID***, Admin_Name, Password)

- **Admin** is already normalized to 2NF because it is already normalized to 1NF and also no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table (Hence no Partial Dependency).

7. **Plan** (***Plan_ID***, Plan_Name, Plan_Duration, Plan_Price)

- **Plan** is already in 1NF and also no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table (Hence no Partial Dependency). So **Plan** is already normalized to 2NF.

8. **Transaction** (***Transaction_ID***, Transaction_Mode, Sender_ID, Receiver_ID, Sender_Type, Receiver_Type, Price)

- **Transaction** is already normalized to 2NF because it is already normalized to 1NF and also no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table (Hence no Partial Dependency).

Normalize the database to 3NF/BCNF

1. **User** (***User_ID***, User_Name, Password, Email_ID, Mobile_Number, Plan_ID, User_Type)

- **User** is already normalized to 2NF and no non key attributes are transitively dependant on the primary key attribute (Hence no transitive dependencies) so **User** is already normalized to 3NF and also in all A → B type relation has A as Candidate Key So it is also in BCNF.

2. **Music** (***Music_ID***, Music_Name, Music_Language, Premium_Information, Views,Likes, Music_Type, Artist_ID, Album_ID)

- **Music** is already normalized to 3NF because it is already normalized to 2NF and no non key attributes are transitively dependent on the primary key attribute (Hence no transitive dependencies). It is also normalized to BCNF because all A → B type relations have A as Candidate key.

3. **Artist (*Artist_ID***, Artist_Name, Password, Total_song_made, Company_ID)

- **Artist** is already normalized to 2NF and no non key attributes are transitively dependant on the primary key attribute (Hence no transitive dependencies) so **Artist** is already normalized to 3NF and also in all A → B type relation has A as Candidate Key So it is also in BCNF.

4. **Album** (***Album_ID***, Album_Name, Album_rating, Profit, Artist_ID, Company_ID)

- **Album** is already normalized to 3NF because it is already normalized to 2NF and no non key attributes are transitively dependent on the primary key attribute (Hence no transitive dependencies). It is also normalized to BCNF because all A → B type relations have A as Candidate key.

5. **Production Company** (***Company_ID***, Company_Name, Base_Salary)

- **Production Company** is already normalized to 2NF and no non key attributes are transitively dependant on the primary key attribute (Hence no transitive dependencies) so **Production Company** is already normalized to 3NF and also in all A → B type relation has A as Candidate Key So it is also in BCNF.

6. **Admin** (***Admin_ID***, Admin_Name, Password)

- **Admin** is already normalized to 3NF because it is already normalized to 2NF and no non key attributes are transitively dependent on the primary key attribute (Hence no transitive dependencies). It is also normalized to BCNF because all A → B type relations have A as Candidate key.

7. **Plan** (***Plan_ID***, Plan_Name, Plan_Duration, Plan_Price)

- **Plan** is already normalized to 2NF and no non key attributes are transitively dependant on the primary key attribute (Hence no transitive dependencies) so **Plan** is already normalized to 3NF and also in all A → B type relation has A as Candidate Key So it is also in BCNF.

8. **Transaction** (***Transaction_ID***, Transaction_Mode, Sender_ID, Receiver_ID, Sender_Type, Receiver_Type, Price)

- **Transaction** is already normalized to 3NF because it is already normalized to 2NF and no non key attributes are transitively dependent on the primary key attribute (Hence no transitive dependencies). It is also normalized to BCNF because all A → B type relations have A as Candidate key.

Updated DDL Script

```
CREATE TABLE "User" (
 "User_ID" BIGINT NOT NULL,
 "User_Name" VARCHAR(100) NOT NULL,
```

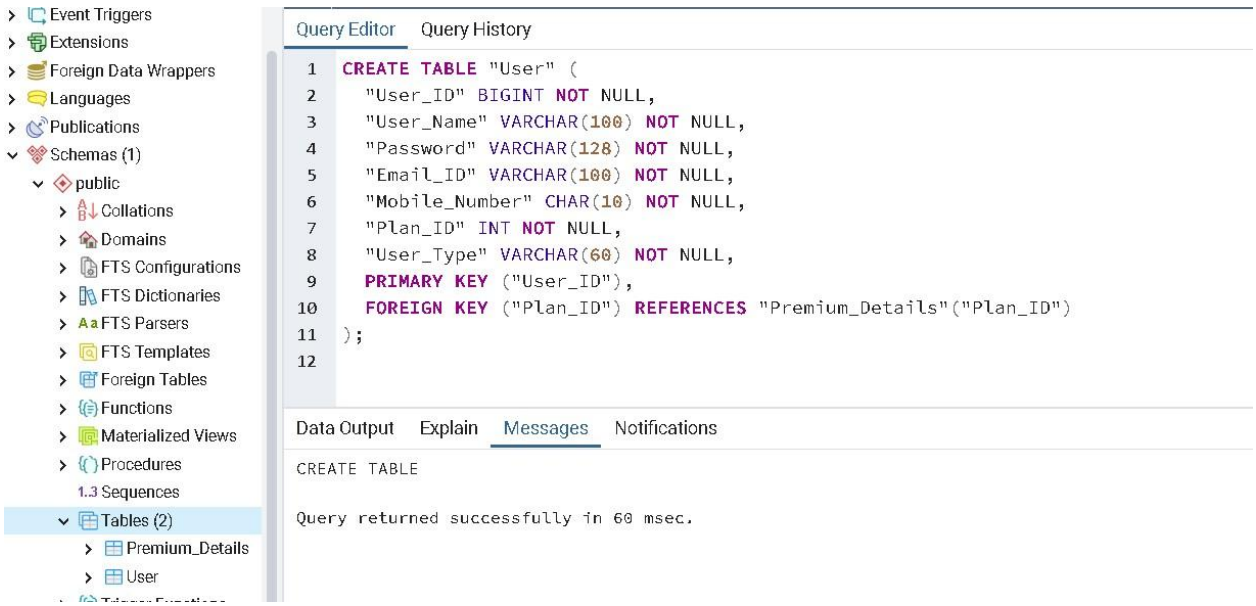
```
 "Password" VARCHAR(128) NOT NULL,
 "Email_ID" VARCHAR(100) NOT NULL,
 "Mobile_Number" CHAR(10) NOT NULL,
 "Plan_ID" INT NOT NULL,
 "User_Type" VARCHAR(60) NOT NULL,
 PRIMARY KEY ("User_ID"),
 FOREIGN KEY ("Plan_ID") REFERENCES "Premium_Details"("Plan_ID")
);


CREATE TABLE "Production_Company"
(
 "Company_ID" BIGINT NOT NULL,
 "Company_Name" VARCHAR(100) NOT NULL,
 "Base_salary" BIGINT NOT NULL,
 PRIMARY KEY ("Company_ID")
);

CREATE TABLE "Admin"
(
 "Admin_ID" BIGINT NOT NULL,
 "Admin_Name" VARCHAR(100) NOT NULL,
 "Password" VARCHAR(128) NOT NULL,
 PRIMARY KEY ("Admin_ID")
);

CREATE TABLE "Premium_Details"
(
 "Plan_ID" BIGINT NOT NULL,
 "Plan_Name" VARCHAR(100) NOT NULL,
 "Plan_Duration" INT NOT NULL,
 "Plan_Price" INT NOT NULL,
 PRIMARY KEY ("Plan_ID")
);

CREATE TABLE "Artist"
(
 "Artist_ID" BIGINT NOT NULL,
```

```
 "Artist_Name" VARCHAR(100) NOT NULL,
 "Password" VARCHAR(128) NOT NULL,
 "Total_Song_Made" INT NOT NULL,
 "Company_ID" BIGINT NOT NULL,
 PRIMARY KEY ("Artist_ID"),
 FOREIGN KEY ("Company_ID") REFERENCES "Production_Company"("Company_ID")
);

CREATE TABLE "Album"
(
 "Album_ID" BIGINT NOT NULL,
 "Album_Name" VARCHAR(100) NOT NULL,
 "Album_Rating" INT NOT NULL,
 "Profit" BIGINT NOT NULL,
 "Artist_ID" BIGINT NOT NULL,
 "Company_ID" BIGINT NOT NULL,
 PRIMARY KEY ("Album_ID"),
 FOREIGN KEY ("Artist_ID") REFERENCES "Artist"("Artist_ID"),
 FOREIGN KEY ("Company_ID") REFERENCES "Production_Company"("Company_ID")
);

CREATE TABLE "Music"
(
 "Music_ID" BIGINT NOT NULL,
 "Music_Name" VARCHAR(100) NOT NULL,
 "Music_Language" VARCHAR(100) NOT NULL,
 "Premium_Information" BOOLEAN NOT NULL,
 "Views" BIGINT NOT NULL,
 "Likes" BIGINT NOT NULL,
 "Music_Type" VARCHAR(60) NOT NULL,
 "Artist_ID" BIGINT NOT NULL,
 "Album_ID" BIGINT NOT NULL,
 PRIMARY KEY ("Music_ID"),
 FOREIGN KEY ("Artist_ID") REFERENCES "Artist"("Artist_ID"),
 FOREIGN KEY ("Album_ID") REFERENCES "Album"("Album_ID")
);

CREATE TABLE "Transaction"
```

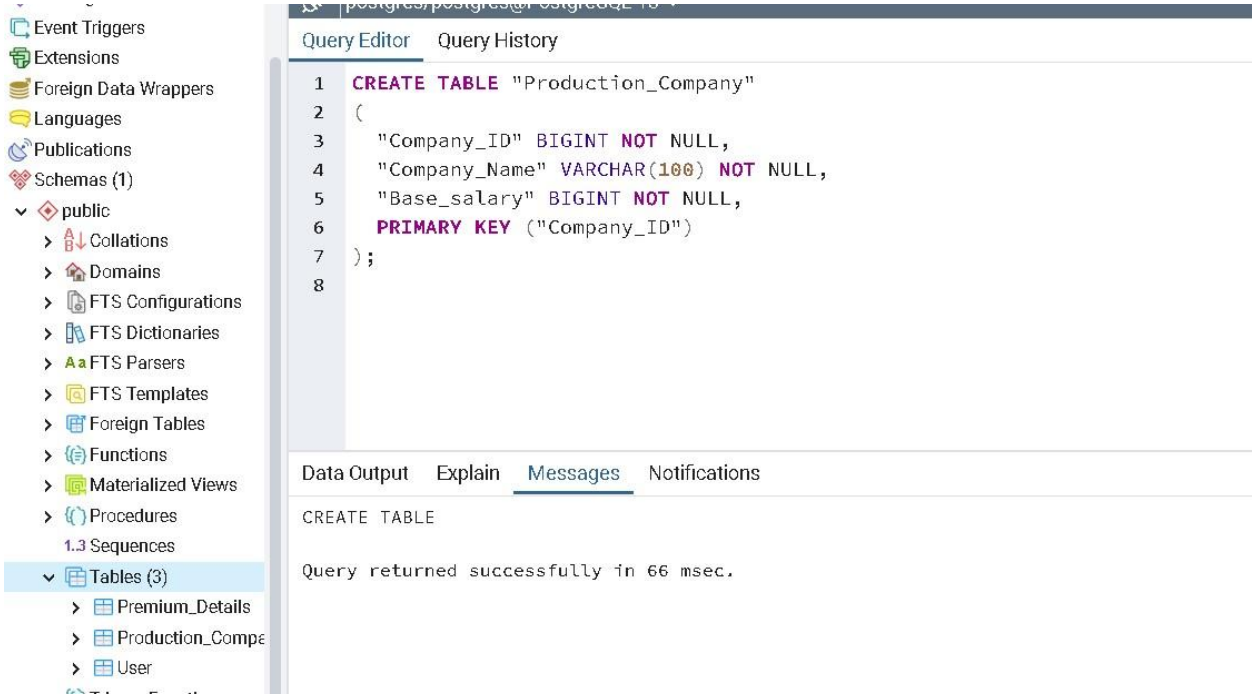
```
(
 "Transaction_ID" BIGINT NOT NULL,
 "Transaction_Mode" VARCHAR(20) NOT NULL,
 "Sender_ID" BIGINT NOT NULL,
 "Receiver_ID" BIGINT NOT NULL,
 "Sender_Type" VARCHAR(20) NOT NULL,
 "Receiver_Type" VARCHAR(20) NOT NULL,
 "Price" BIGINT NOT NULL,
 PRIMARY KEY ("Transaction_ID")
);
```

Snapshot of Create Table using DDL

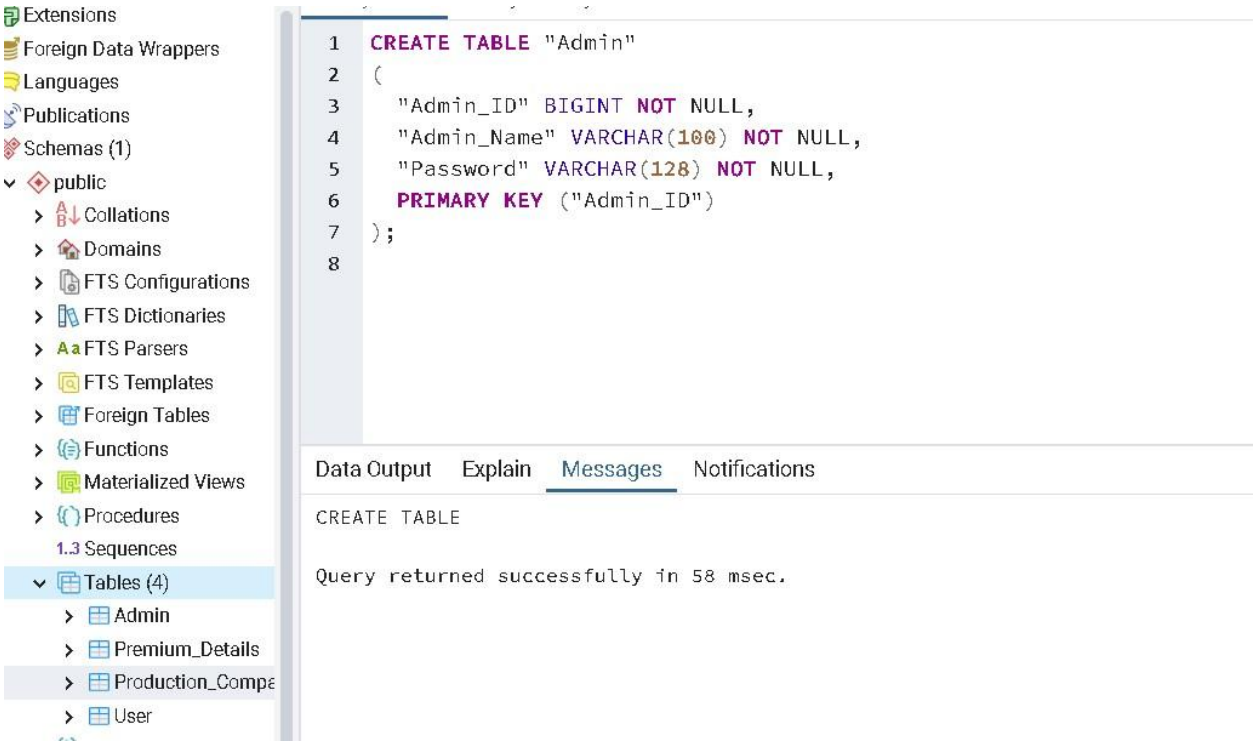
1. **User**



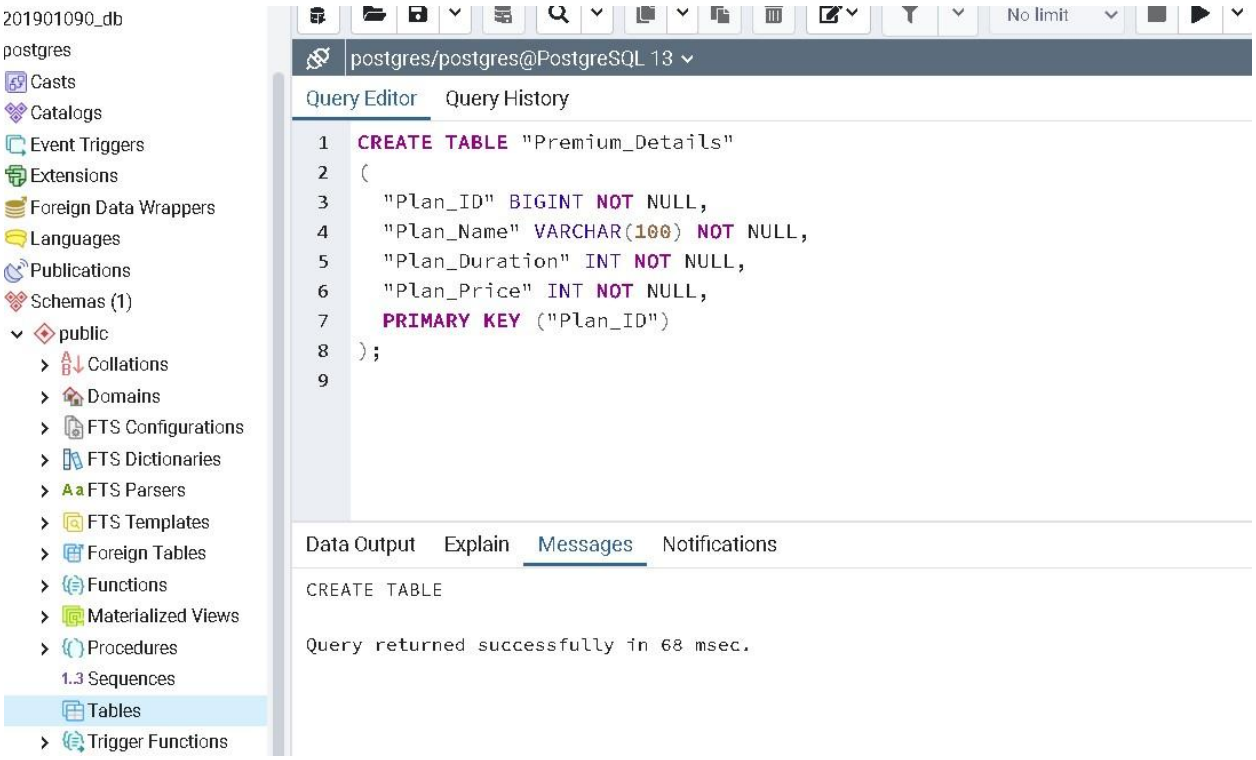
2. **Production_Company**



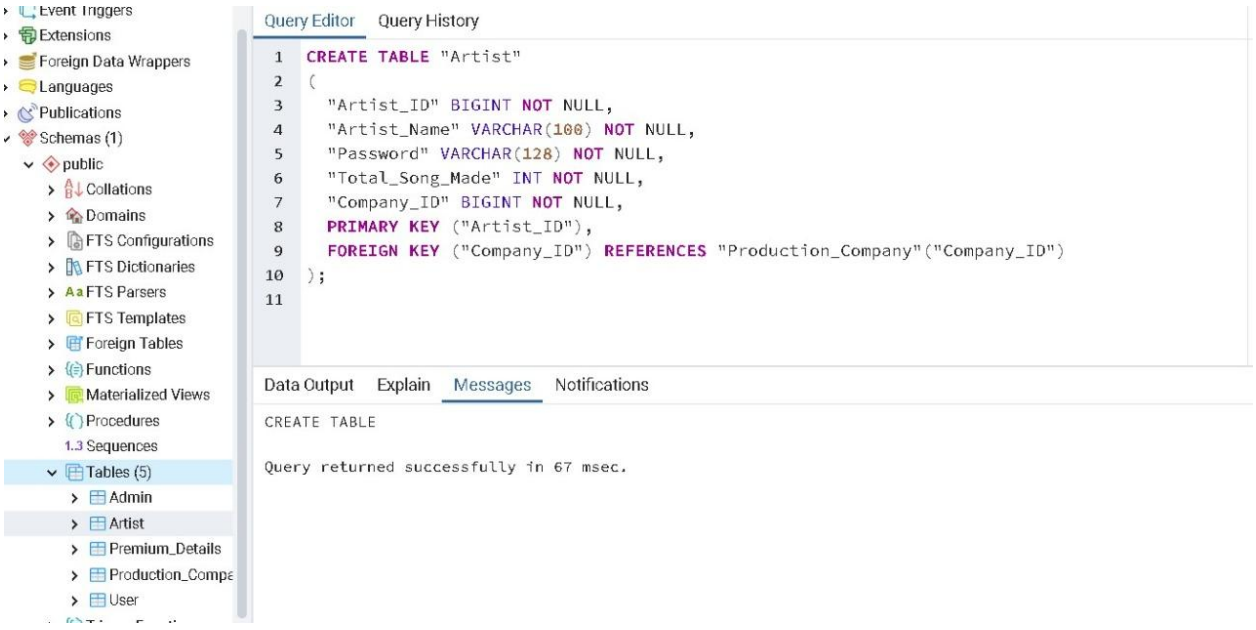
3. Admin



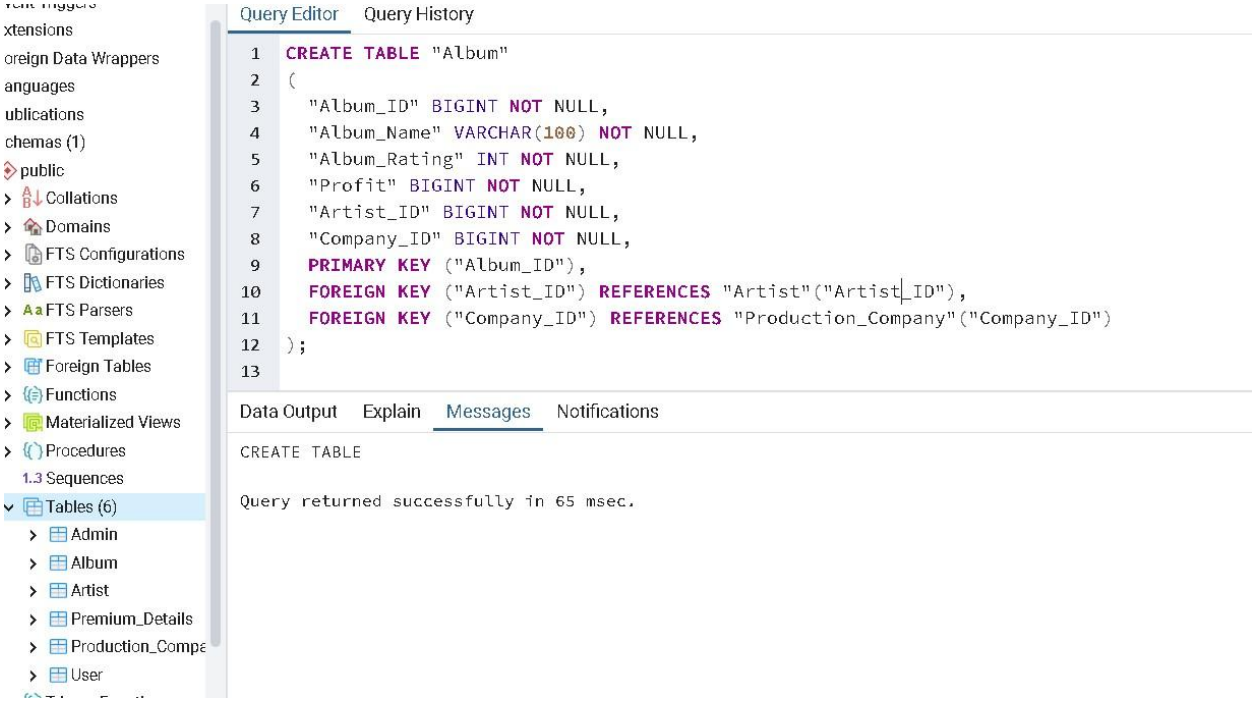
4. Premium_Details



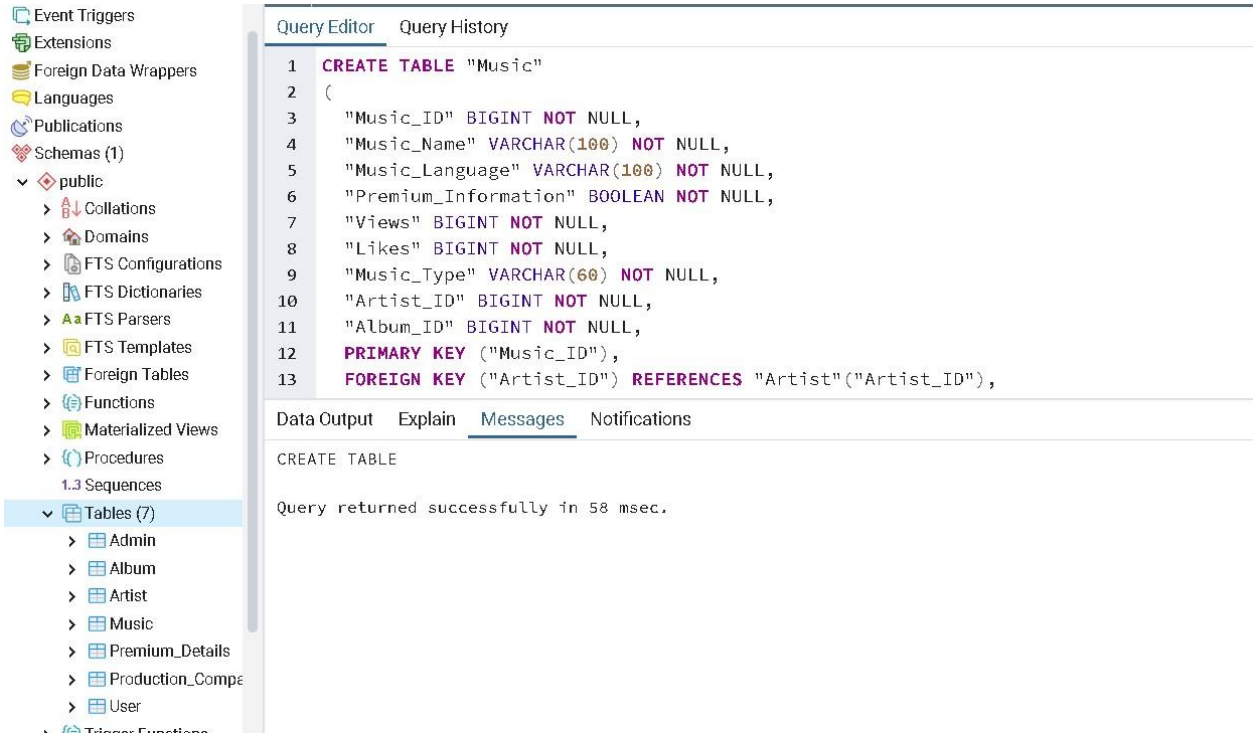
5. Artist



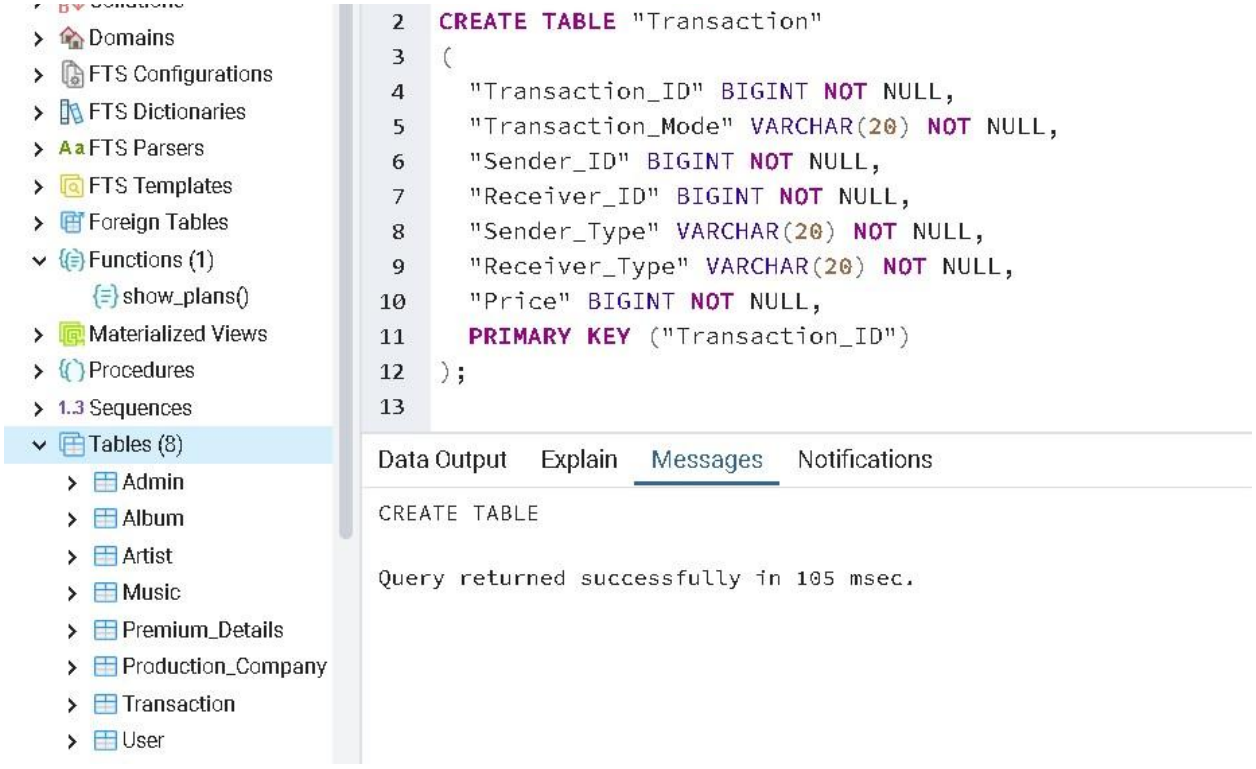
6. Album



7. Music

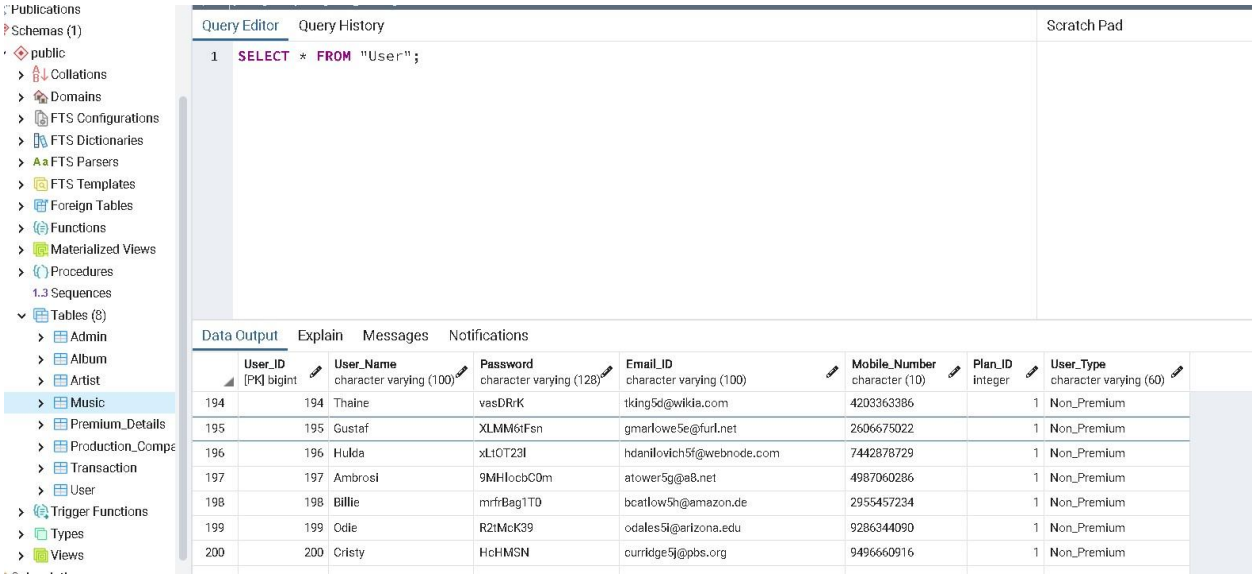


8. Transaction

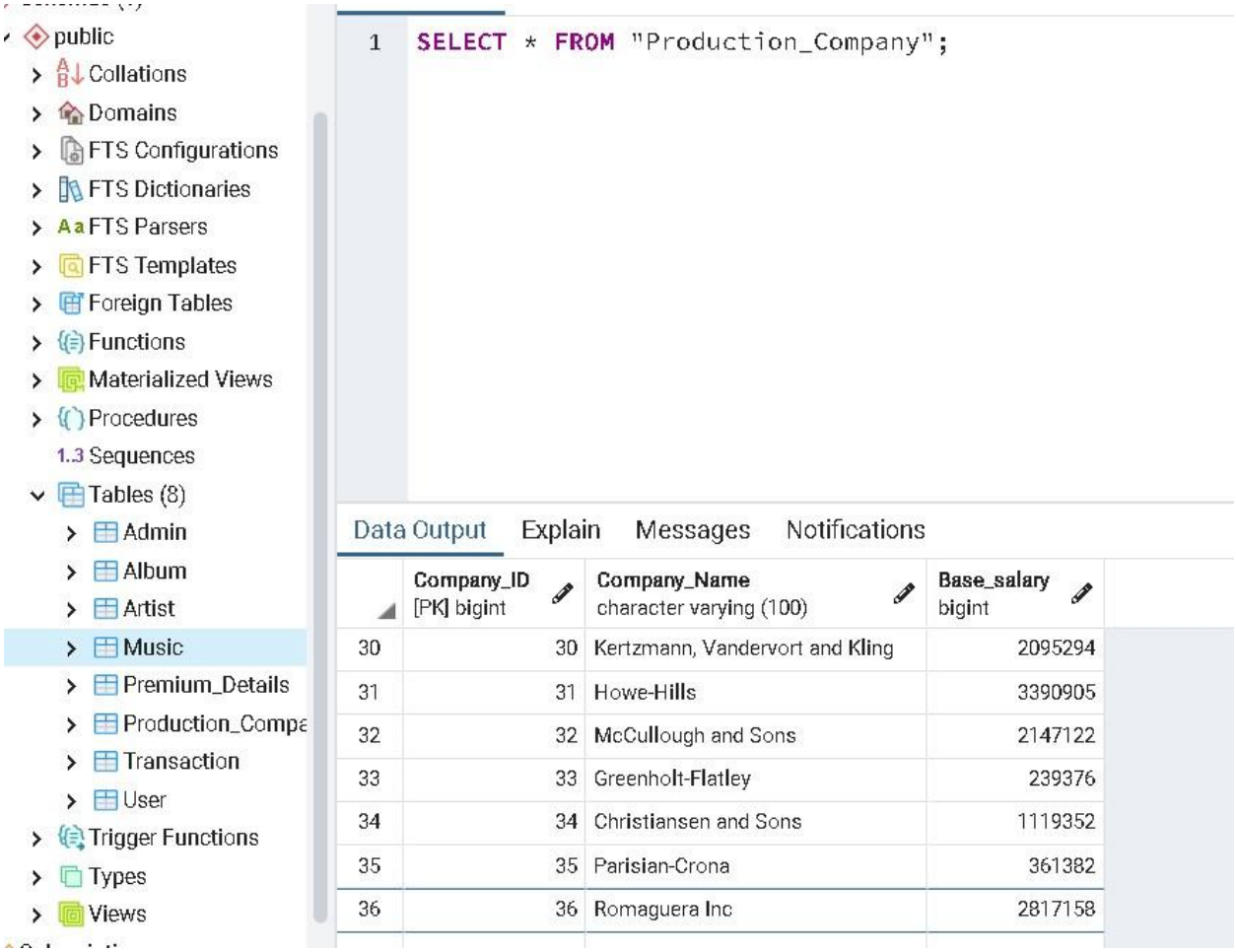


Data snapshots

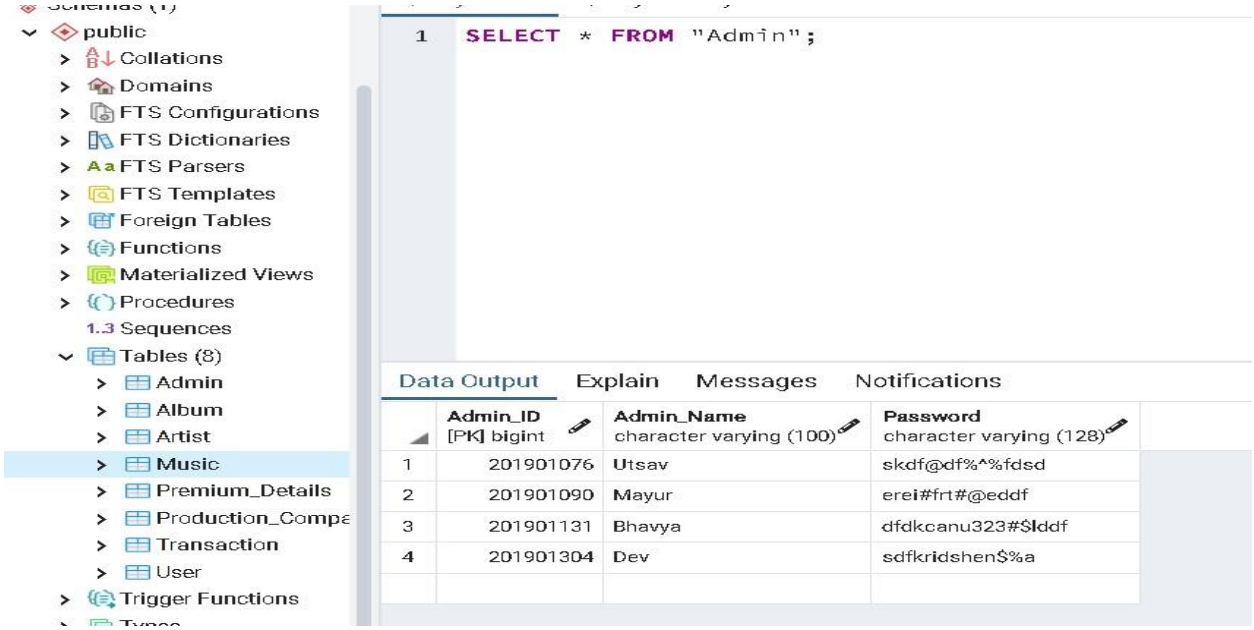
1. User



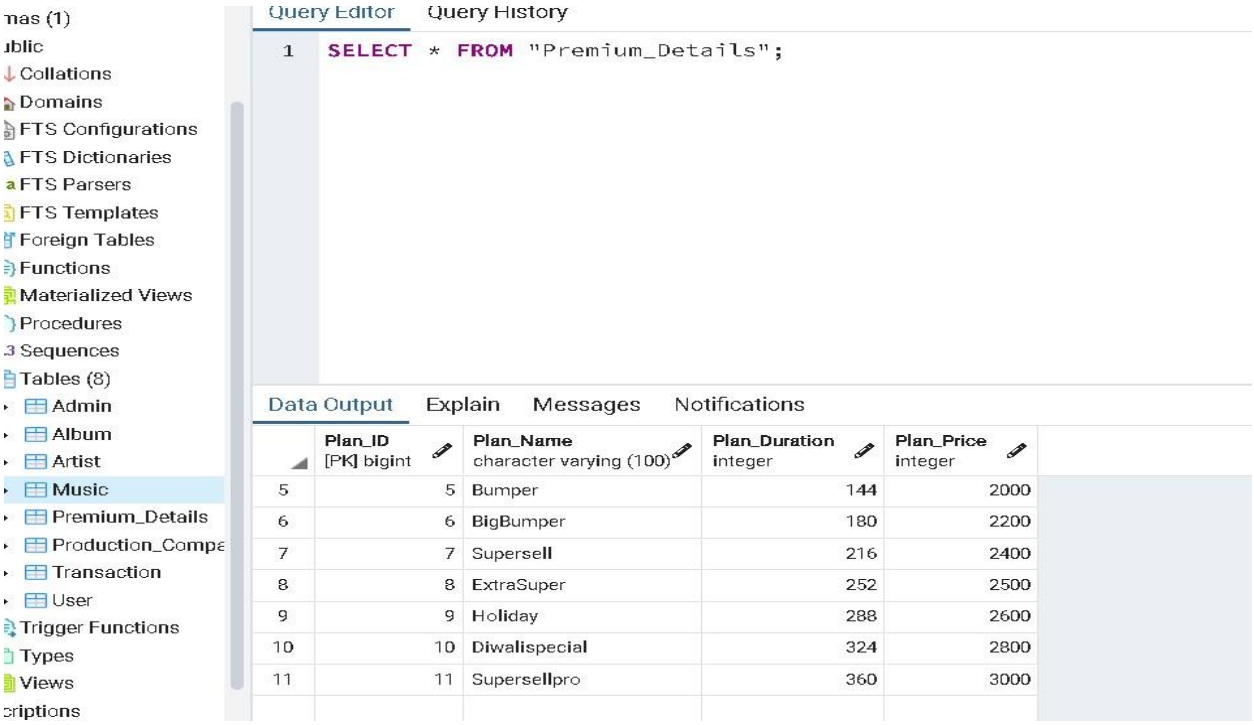
2. Production Company


public
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (8)
Admin
Album
Artist
Music
Premium_Details
Production_Compa
Transaction
User
Trigger Functions
Types
Views
1 SELECT * FROM "Production_Company";
Data Output
Explain
Messages
Notifications
Company_ID [PK] bigint
Company_Name character varying (100)
Base_salary bigint
30 30 Kertzmann, Vandervort and Kling 2095294
31 31 Howe-Hills 3390905
32 32 McCullough and Sons 2147122
33 33 Greenholt-Flatley 239376
34 34 Christiansen and Sons 1119352
35 35 Parisian-Crona 361382
36 36 Romaguera Inc 2817158

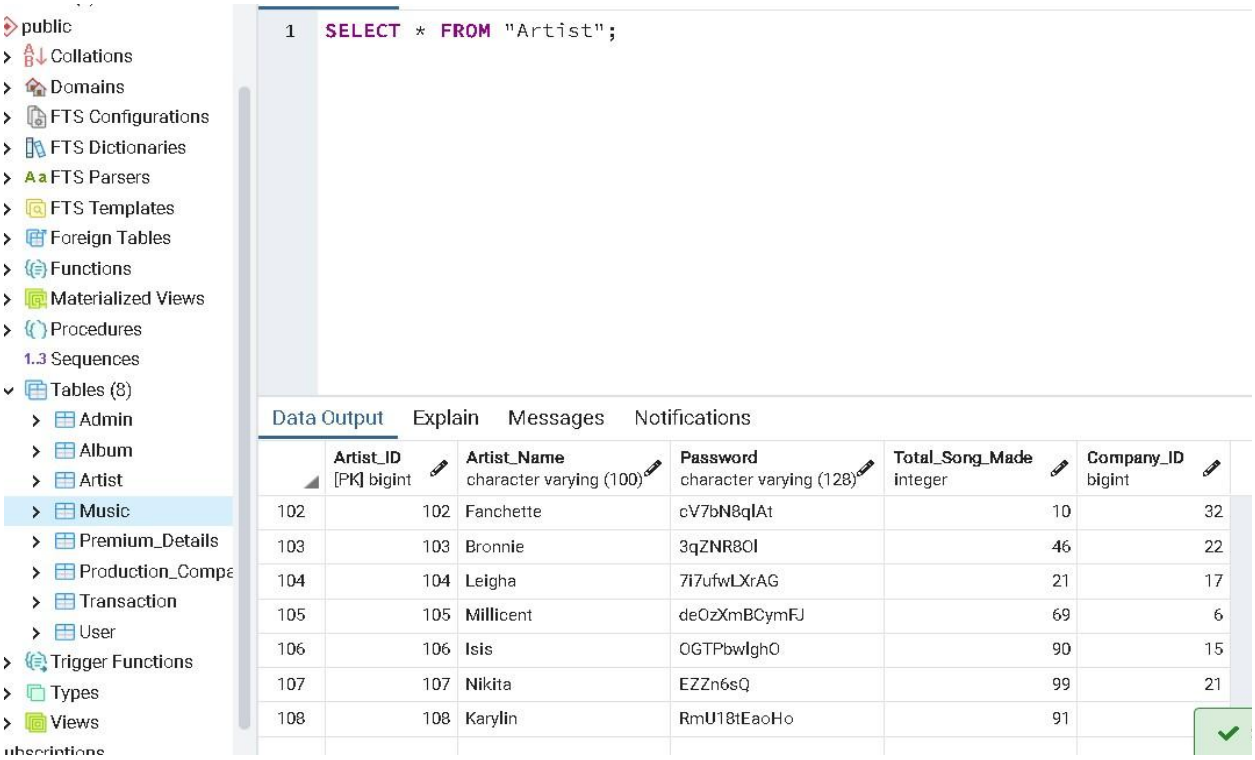
3. Admin


public
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (8)
Admin
Album
Artist
Music
Premium_Details
Production_Compa
Transaction
User
Trigger Functions
1 SELECT * FROM "Admin";
Data Output
Explain
Messages
Notifications
Admin_ID [PK] bigint
Admin_Name character varying (100)
Password character varying (128)
1 201901076 Utsav skdf@df%^%fdsd
2 201901090 Mayur erei#frt#@eddf
3 201901131 Bhavya dfdkcanu323#$lddf
4 201901304 Dev sdfkridshen$%a

4. Premium Details



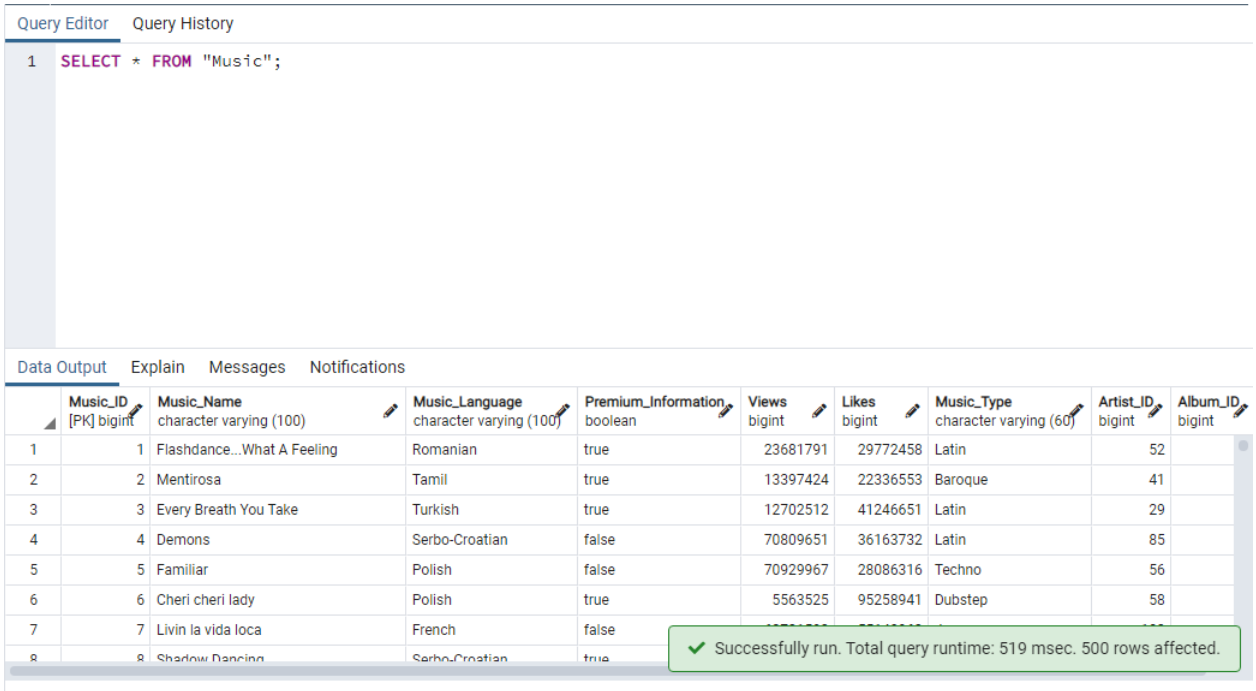
5. Artist



6. Album

	Album_ID [PK] bigint	Album_Name character varying (100)	Album_Rating integer	Profit bigint	Artist_ID bigint	Company_ID bigint
69	69	Thor : Ragnarok	2	7632	101	28
70	70	Fantastic Beasts and Where To Find Them	8	3196	7	4
71	71	Wonder Woman	1	4056	52	23
72	72	The Martian	9	3278	89	19
73	73	The Dark Tower	8	7303	67	3
74	74	Shrek	9	2271	4	18
75	75	Harry Potter and The Sorcerer's Stone	4	2111	51	13

7. Music



	Music_ID [PK] bigint	Music_Name character varying (100)	Music_Language character varying (100)	Premium_Information boolean	Views bigint	Likes bigint	Music_Type character varying (60)	Artist_ID bigint	Album_ID bigint
1	1	Flashdance...What A Feeling	Romanian	true	23681791	29772458	Latin	52	
2	2	Mentirosa	Tamil	true	13397424	22336553	Baroque	41	
3	3	Every Breath You Take	Turkish	true	12702512	41246651	Latin	29	
4	4	Demons	Serbo-Croatian	false	70809651	36163732	Latin	85	
5	5	Familiar	Polish	false	70929967	28086316	Techno	56	
6	6	Cheri cheri lady	Polish	true	5563525	95258941	Dubstep	58	
7	7	Livin la vida loca	French	false					
8	8	Shadow Dancing	Serbo-Croatian	true					

8. Transaction

Query Editor | Query History

```
1  SELECT * FROM "Transaction";
```

Data Output | Explain | Messages | Notifications

	Transaction_ID [PK] bigint	Transaction_Mode character varying (20)	Sender_ID bigint	Receiver_ID bigint	Sender_Type character varying (20)	Receiver_Type character varying (20)	Price bigint
1	1	offline	1	201901076	User	Admin	1500
2	2	offline	1	201901076	User	Admin	1500
3	3	offline	1	201901076	User	Admin	1500
4	4	offline	1	201901076	User	Admin	1500
5	5	offline	1	201901076	User	Admin	1500
6	6	offline	1002	201901076	User	Admin	0
7	7	offline	1003	201901076	User		
8	8	offline	1004	201901076	User		

Successfully run. Total query runtime: 527 msec. 26 rows affected.

Section 6: SQL: Final DDL Scripts, Insert statements, 40 SQL Queries with Snapshots of output of each query.

Final DDL Script

```
CREATE TABLE "User" (
  "User_ID" BIGINT NOT NULL,
  "User_Name" VARCHAR(100) NOT NULL,
  "Password" VARCHAR(128) NOT NULL,
  "Email_ID" VARCHAR(100) NOT NULL,
  "Mobile_Number" CHAR(10) NOT NULL,
  "Plan_ID" INT NOT NULL,
  "User_Type" VARCHAR(60) NOT NULL,
  PRIMARY KEY ("User_ID"),
  FOREIGN KEY ("Plan_ID") REFERENCES "Premium_Details"("Plan_ID")
);


CREATE TABLE "Production_Company"
(
  "Company_ID" BIGINT NOT NULL,
  "Company_Name" VARCHAR(100) NOT NULL,
  "Base_salary" BIGINT NOT NULL,
  PRIMARY KEY ("Company_ID")
);

CREATE TABLE "Admin"
(
  "Admin_ID" BIGINT NOT NULL,
  "Admin_Name" VARCHAR(100) NOT NULL,
  "Password" VARCHAR(128) NOT NULL,
  PRIMARY KEY ("Admin_ID")
);

CREATE TABLE "Premium_Details"
(
  "Plan_ID" BIGINT NOT NULL,
  "Plan_Name" VARCHAR(100) NOT NULL,
  "Plan_Duration" INT NOT NULL,
  "Plan_Price" INT NOT NULL,
```

```
 PRIMARY KEY ("Plan_ID")
);

CREATE TABLE "Artist"
(
 "Artist_ID" BIGINT NOT NULL,
 "Artist_Name" VARCHAR(100) NOT NULL,
 "Password" VARCHAR(128) NOT NULL,
 "Total_Song_Made" INT NOT NULL,
 "Company_ID" BIGINT NOT NULL,
 PRIMARY KEY ("Artist_ID"),
 FOREIGN KEY ("Company_ID") REFERENCES "Production_Company"("Company_ID")
);

CREATE TABLE "Album"
(
 "Album_ID" BIGINT NOT NULL,
 "Album_Name" VARCHAR(100) NOT NULL,
 "Album_Rating" INT NOT NULL,
 "Profit" BIGINT NOT NULL,
 "Artist_ID" BIGINT NOT NULL,
 "Company_ID" BIGINT NOT NULL,
 PRIMARY KEY ("Album_ID"),
 FOREIGN KEY ("Artist_ID") REFERENCES "Artist"("Artist_ID"),
 FOREIGN KEY ("Company_ID") REFERENCES "Production_Company"("Company_ID")
);

CREATE TABLE "Music"
(
 "Music_ID" BIGINT NOT NULL,
 "Music_Name" VARCHAR(100) NOT NULL,
 "Music_Language" VARCHAR(100) NOT NULL,
 "Premium_Information" BOOLEAN NOT NULL,
 "Views" BIGINT NOT NULL,
 "Likes" BIGINT NOT NULL,
 "Music_Type" VARCHAR(60) NOT NULL,
 "Artist_ID" BIGINT NOT NULL,
 "Album_ID" BIGINT NOT NULL,
```

```
 PRIMARY KEY ("Music_ID"),
 FOREIGN KEY ("Artist_ID") REFERENCES "Artist"("Artist_ID"),
 FOREIGN KEY ("Album_ID") REFERENCES "Album"("Album_ID")
);

CREATE TABLE "Transaction"
(
 "Transaction_ID" BIGINT NOT NULL,
 "Transaction_Mode" VARCHAR(20) NOT NULL,
 "Sender_ID" BIGINT NOT NULL,
 "Receiver_ID" BIGINT NOT NULL,
 "Sender_Type" VARCHAR(20) NOT NULL,
 "Receiver_Type" VARCHAR(20) NOT NULL,
 "Price" BIGINT NOT NULL,
 PRIMARY KEY ("Transaction_ID")
);
```

Insert into Scripts

```
INSERT INTO "T9_OMMS"."User"(
        "User_ID", "User_Name", "Password", "Email_ID", "Mobile_Number", "Plan_ID",
"User_Type")
        VALUES (?, ?, ?, ?, ?, ?, ?);

INSERT INTO "T9_OMMS"."Production_Company"(
        "Company_ID", "Company_Name", "Base_salary")
        VALUES (?, ?, ?);

INSERT INTO "T9_OMMS"."Admin"(
        "Admin_ID", "Admin_Name", "Password")
        VALUES (?, ?, ?);

INSERT INTO "T9_OMMS"."Premium_Details"(
        "Plan_ID", "Plan_Name", "Plan_Duration", "Plan_Price")
        VALUES (?, ?, ?, ?);

INSERT INTO "T9_OMMS"."Artist"(
        "Artist_ID", "Artist_Name", "Password", "Total_Song_Made", "Company_ID")
        VALUES (?, ?, ?, ?, ?);
```

```
INSERT INTO "T9_OMMS"."Album"(
        "Album_ID", "Album_Name", "Album_Rating", "Profit", "Artist_ID", "Company_ID")
        VALUES (?, ?, ?, ?, ?, ?);

INSERT INTO "T9_OMMS"."Music"(
        "Music_ID", "Music_Name", "Music_Language", "Premium_Information", "Views", "Likes",
"Music_Type", "Artist_ID", "Album_ID")
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);

INSERT INTO "T9_OMMS"."Transaction"(
        "Transaction_ID", "Transaction_Mode", "Sender_ID", "Receiver_ID", "Sender_Type",
"Receiver_Type", "Price")
        VALUES (?, ?, ?, ?, ?, ?, ?);
```

1)

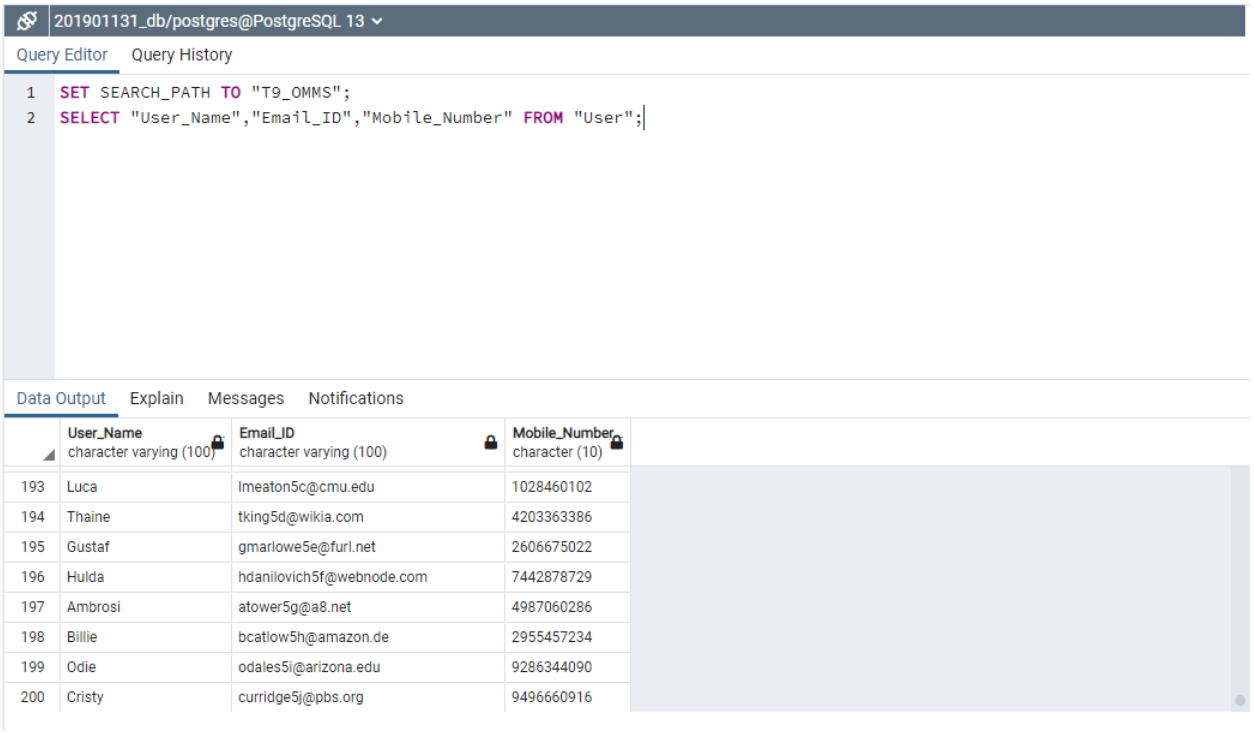
Plain English Query :

Show Details of Users (user name, email ID, phone number).

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "User_Name","Email_ID","Mobile_Number" FROM "User";
```

Snapshot :



Result Contains :

200 Tuples.

2)

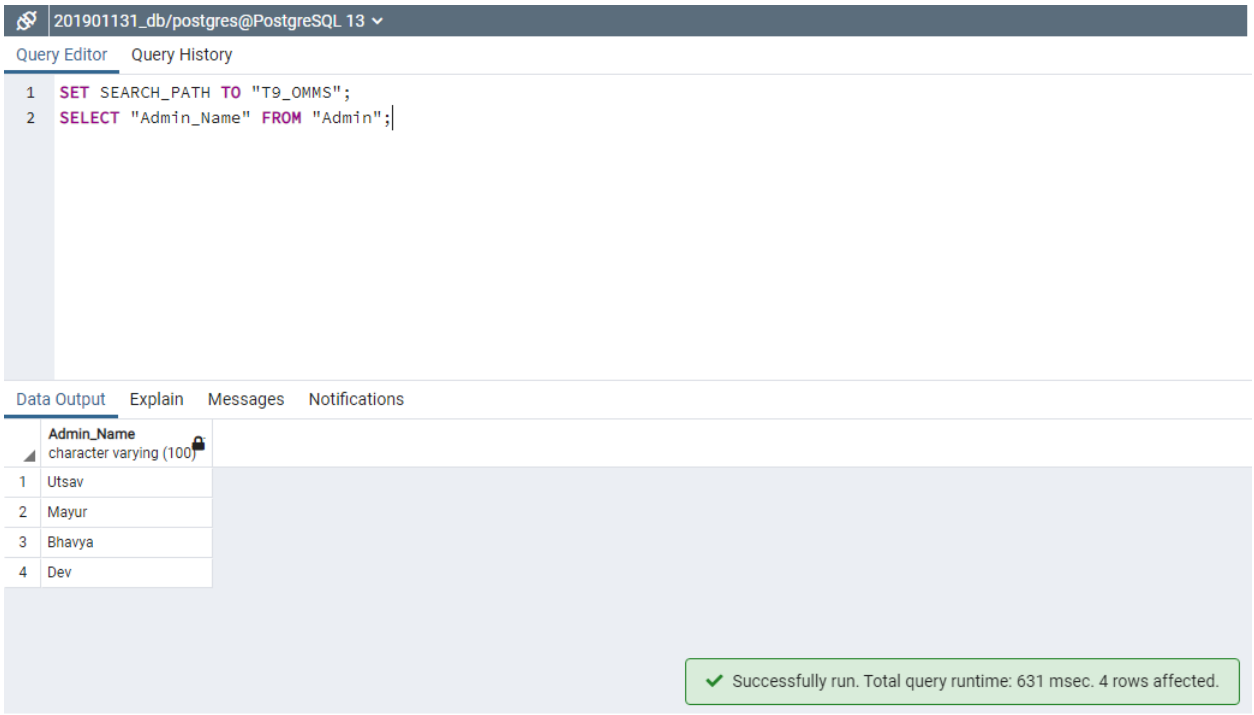
Plain English Query :

Show Names of the All Admins.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Admin_Name" FROM "Admin";
```

Snapshot :



Result Contains :

4 Tuples.

3)

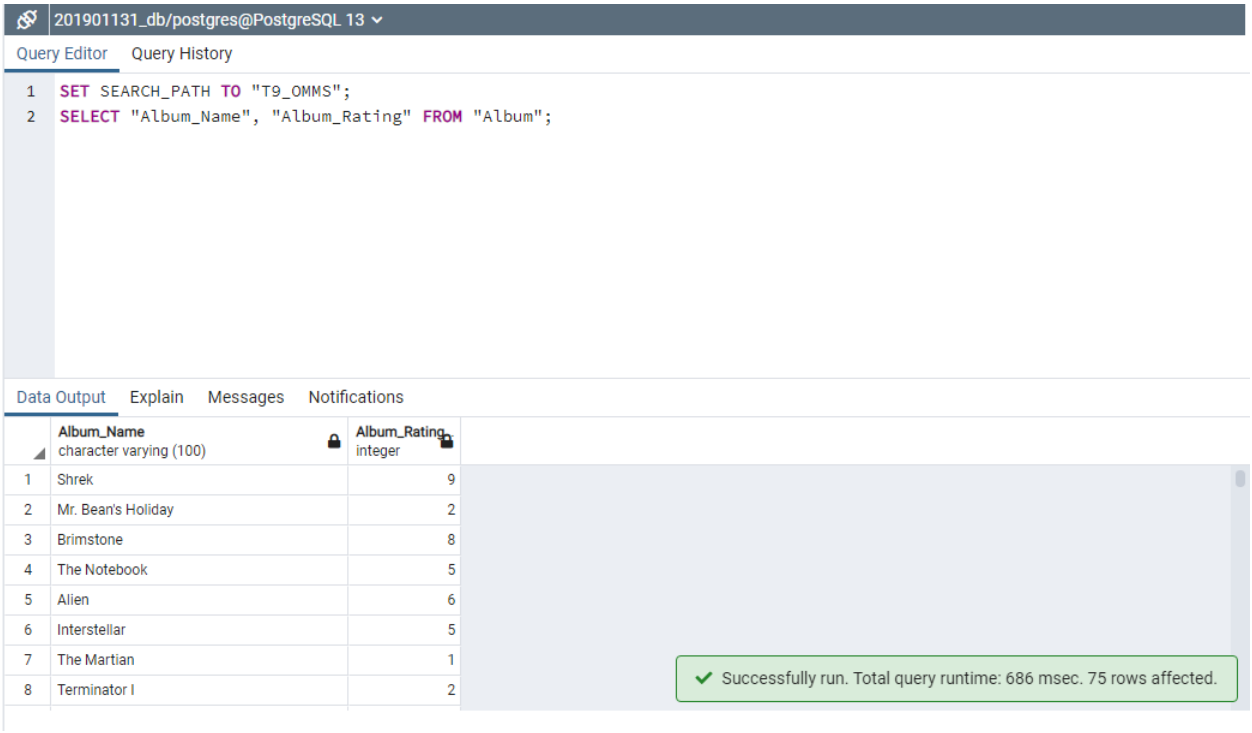
Plain English Query :

Show Rating of the Albums with Album Name.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Album_Name", "Album_Rating" FROM "Album";
```

Snapshot :



Result Contains :

75 Tuples.

4)

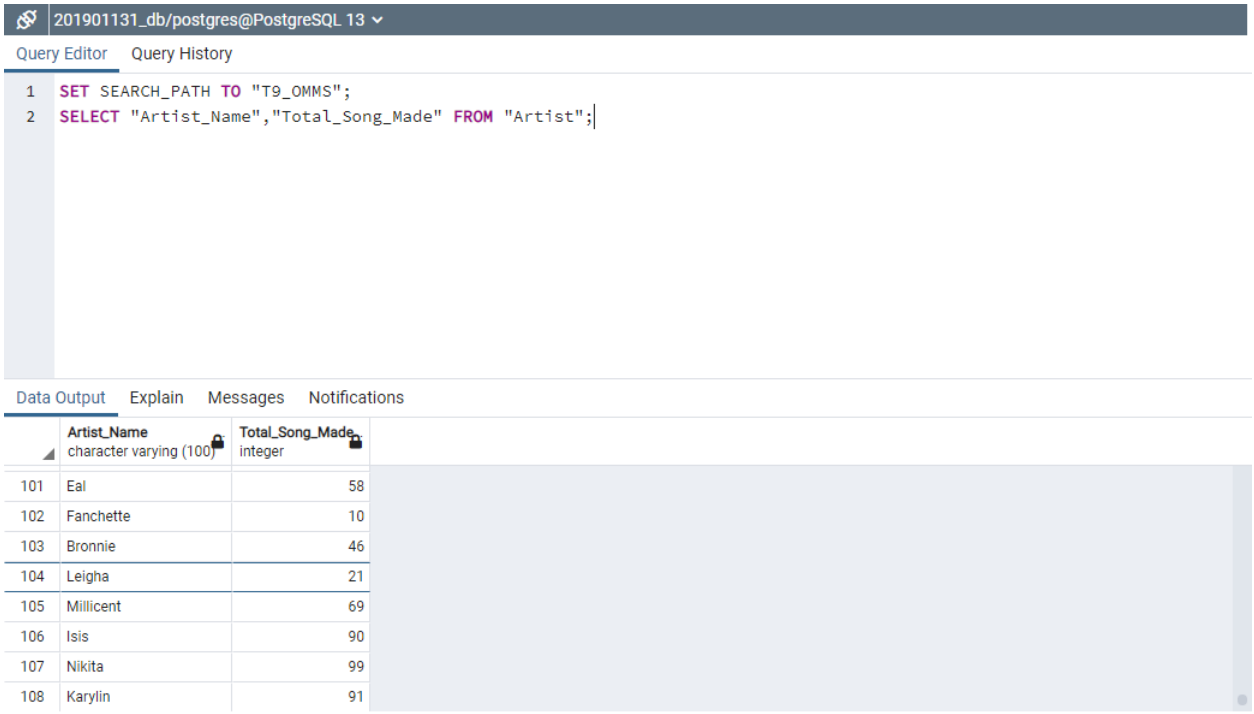
Plain English Query :

Show Count of How many songs an Artist has made along with Artist Name.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Artist_Name","Total_Song_Made" FROM "Artist";
```

Snapshot :



Result Contains :

108 Tuples.

5)

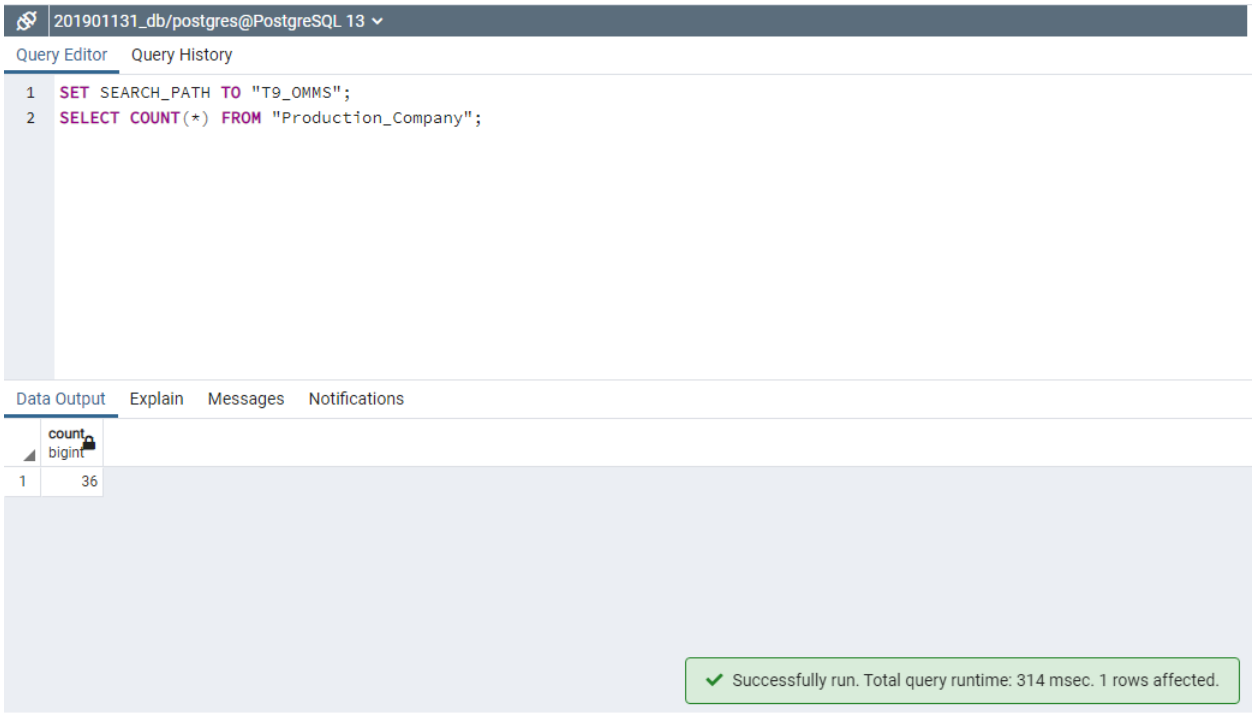
Plain English Query :

Count Total Number Of Production Companies.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT COUNT(*) FROM "Production_Company";
```

Snapshot :



Result Contains :

Count = 36 (1 Tuple).

6)

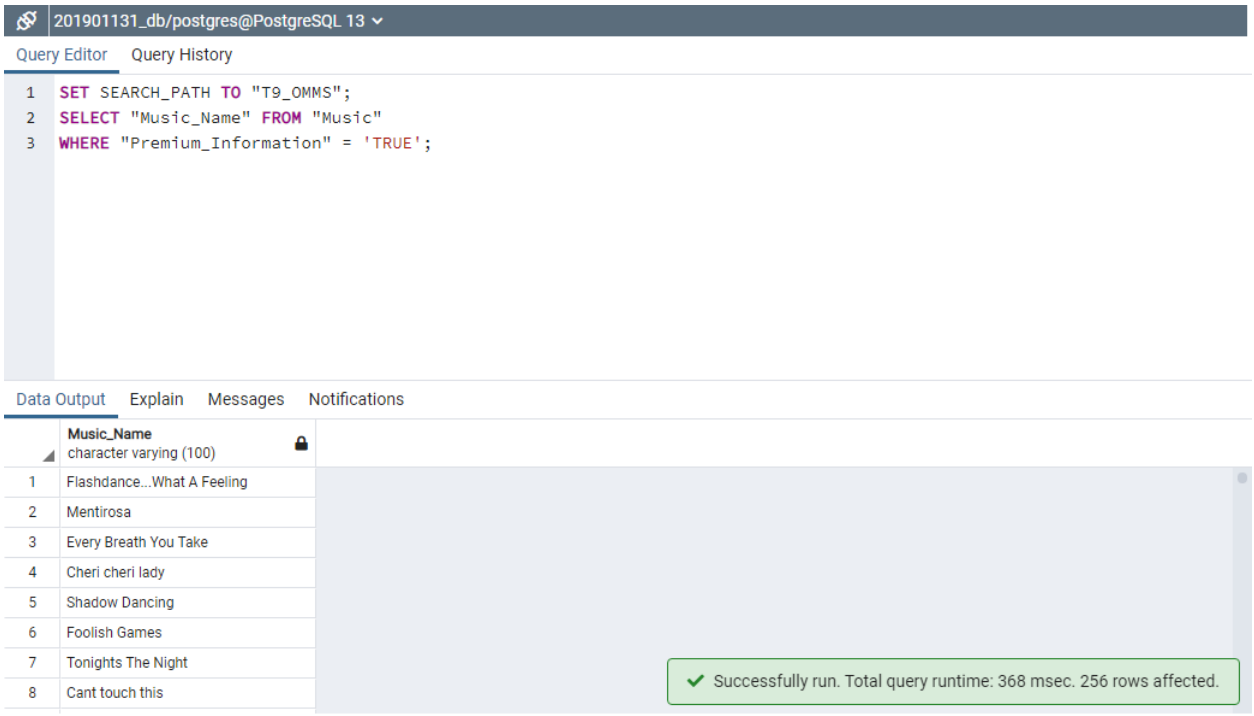
Plain English Query :

List Name of All Premium Songs

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Music_Name" FROM "Music"
WHERE "Premium_Information" = 'TRUE';
```

Snapshot :



Result Contains :

256 Tuples.

7)

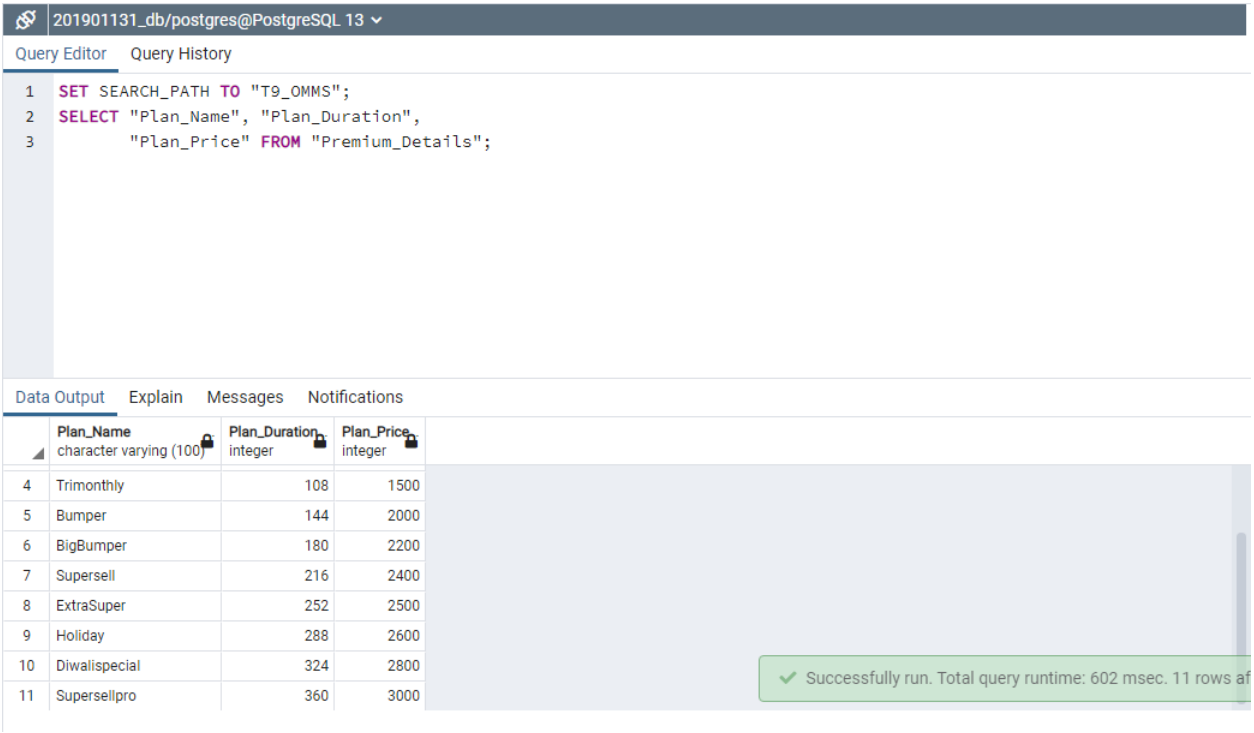
Plain English Query :

Show Details of all the Plans.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Plan_Name", "Plan_Duration",
    "Plan_Price" FROM "Premium_Details";
```

Snapshot :



Result Contains :

11 Tuples.

8)

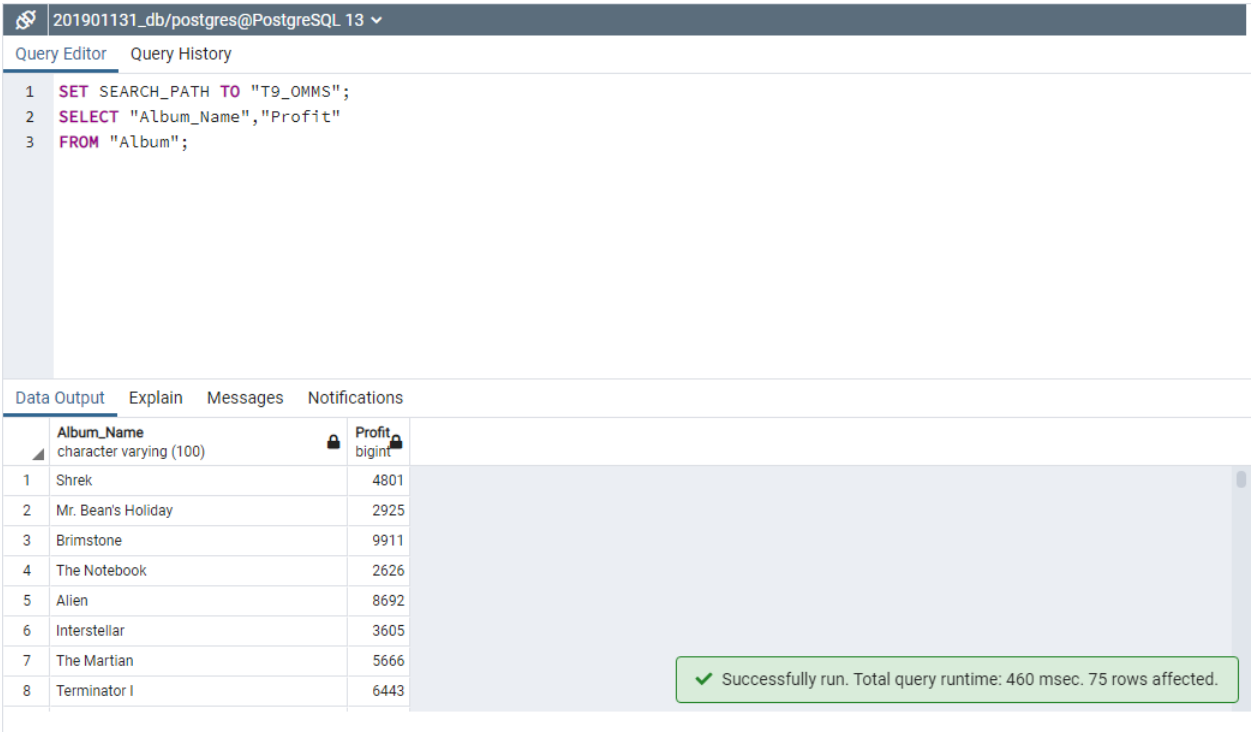
Plain English Query :

Show Money made by each Album.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Album_Name","Profit"
        FROM "Album";
```

Snapshot :



Result Contains :

75 Tuples.

9)

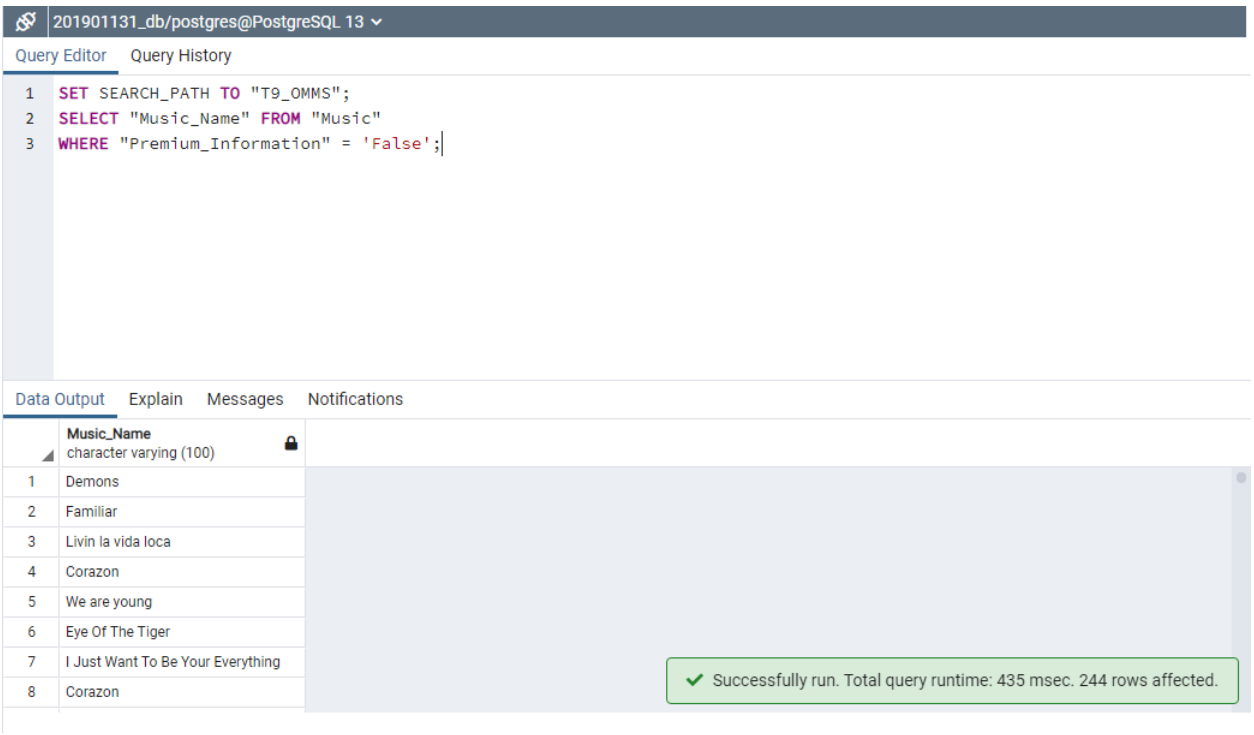
Plain English Query :

List all song names that non-premium users can listen.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Music_Name" FROM "Music"
WHERE "Premium_Information" = 'False';
```

Snapshot :



Result Contains :

244 Tuples.

10)

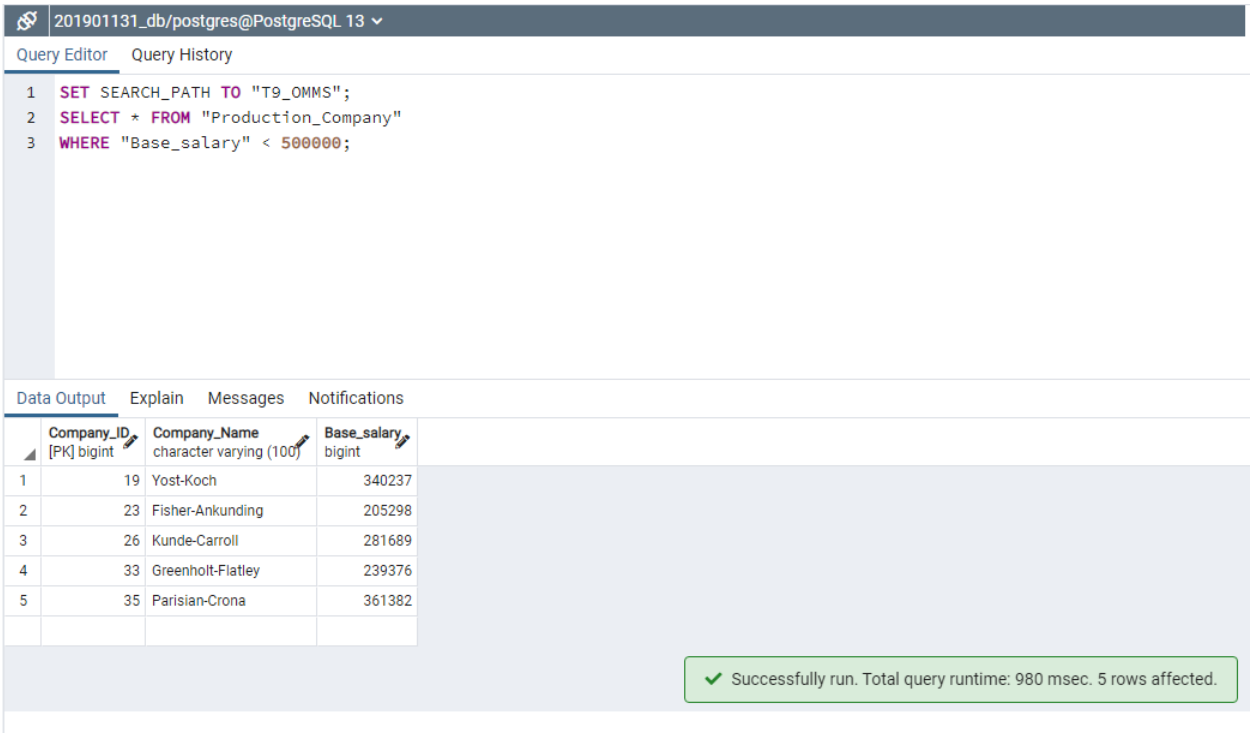
Plain English Query :

List all the Company names which give the base salary less than 500000 .

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT * FROM "Production_Company"
WHERE "Base_salary" < 500000;
```

Snapshot :



Result Contains :

5 Tuples.

11)

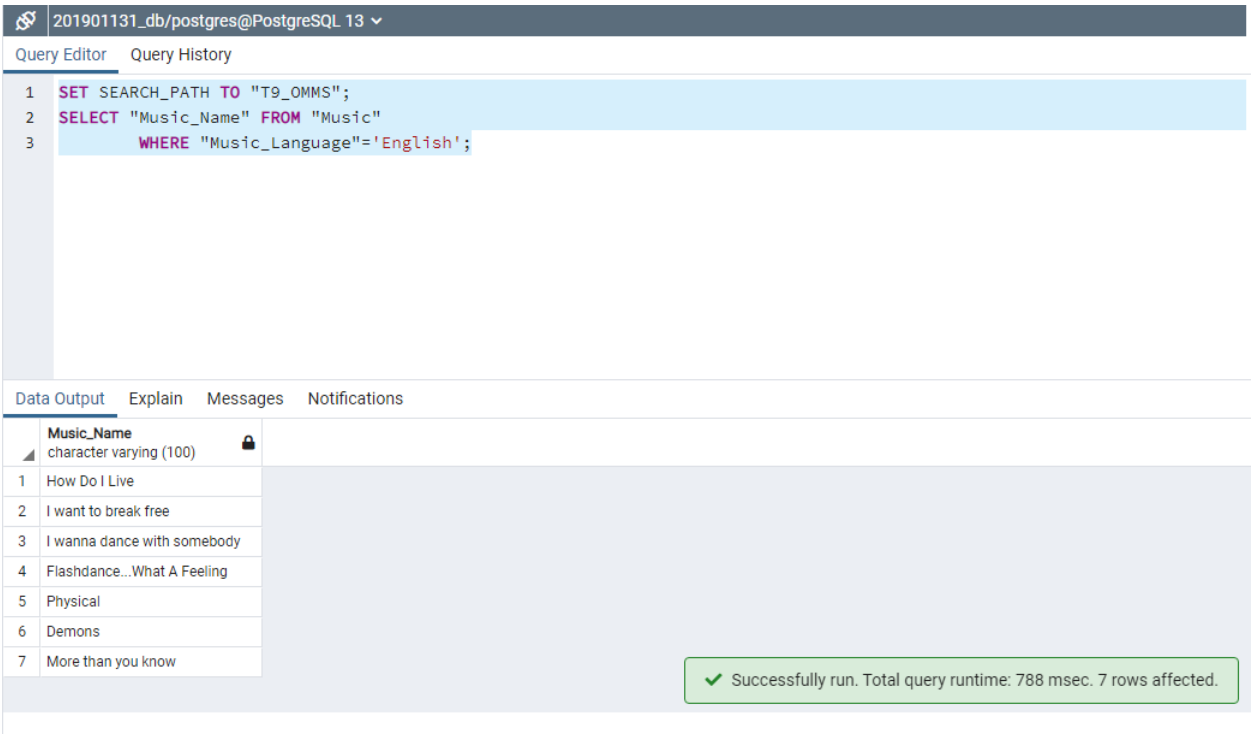
Plain English Query :

List all the Music names with English language.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Music_Name" FROM "Music"
        WHERE "Music_Language"='English';
```

Snapshot :



Result Contains : 7 Tuples.

12)

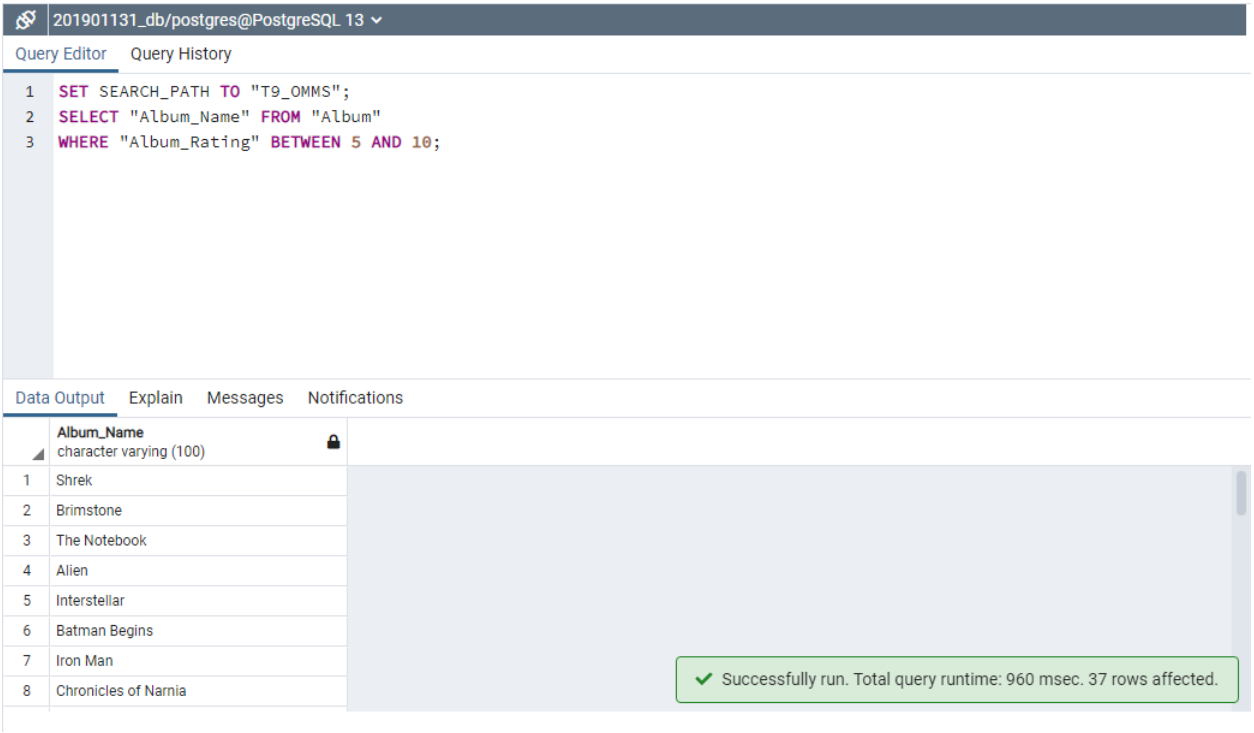
Plain English Query :

List all the album names which rating between 6 to 10.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Album_Name" FROM "Album"
WHERE "Album_Rating" BETWEEN 5 AND 10;
```

Snapshot :



Result Contains :

37 Tuples.

13)

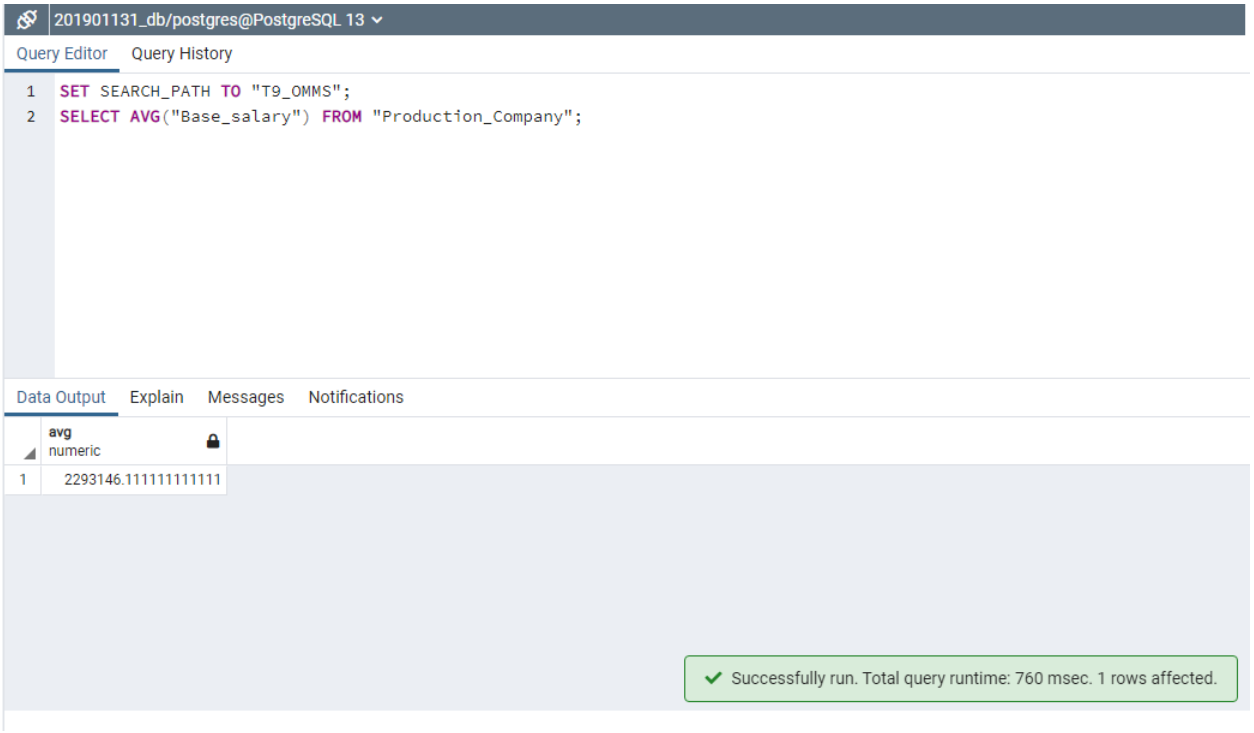
Plain English Query :

Find the average base salary of all the companies.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT AVG("Base_salary") FROM "Production_Company";
```

Snapshot :



Result Contains :

1 Tuple.

14)

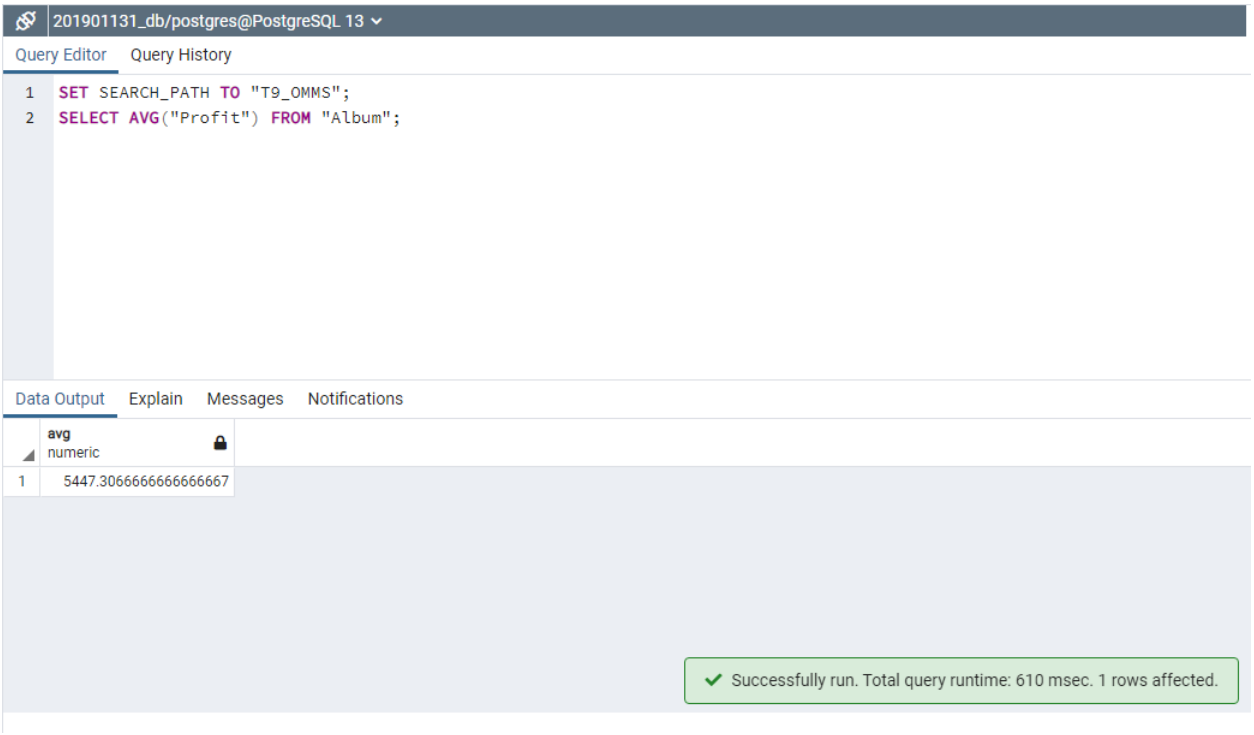
Plain English Query :

Find out the average profit of albums.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT AVG("Profit") FROM "Album";
```

Snapshot :



Result Contains :

1 Tuple.

15)

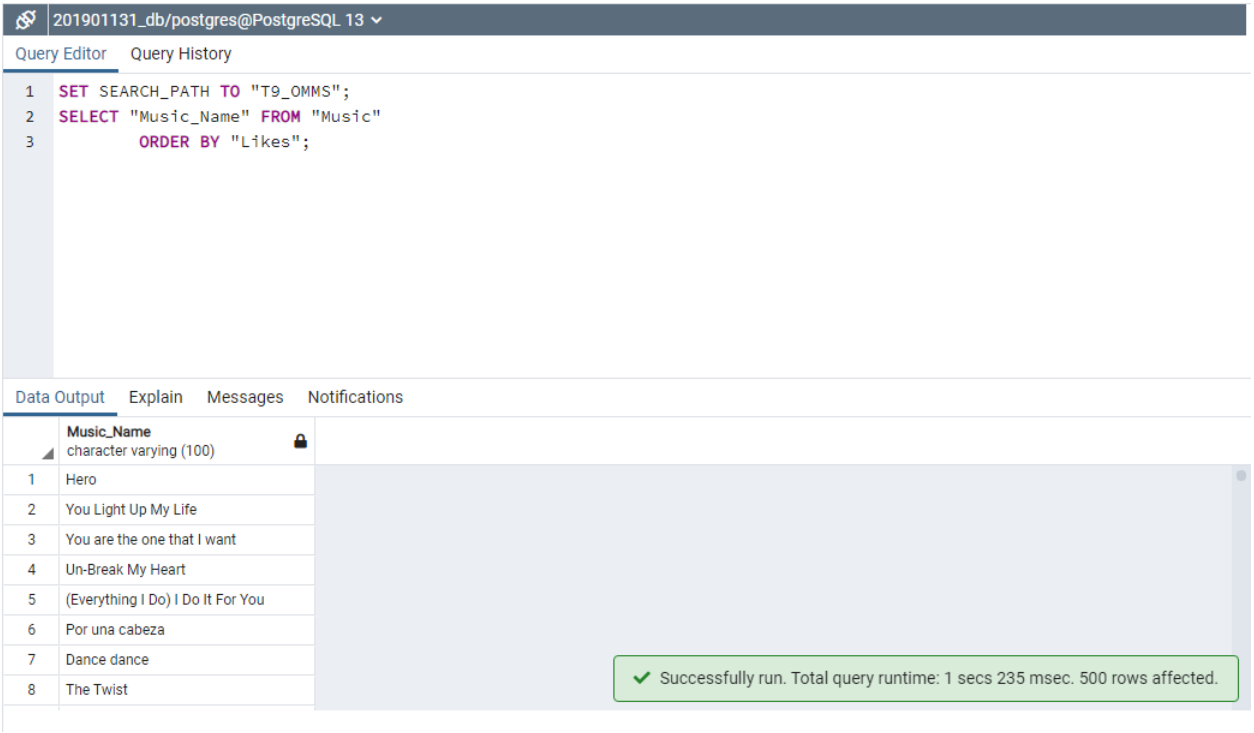
Plain English Query :

List the music names in ascending order of likes.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Music_Name" FROM "Music"
        ORDER BY "Likes";
```

Snapshot :



Result Contains :

500 Tuples.

16)

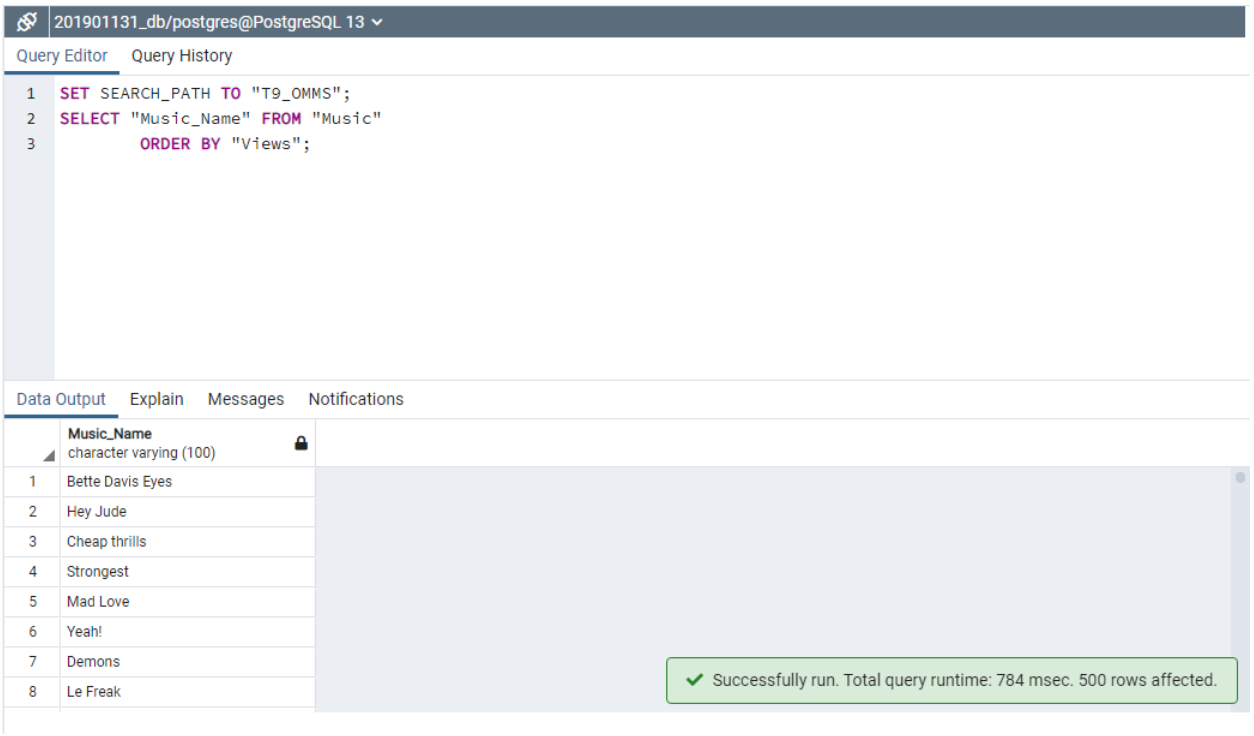
Plain English Query :

List the music names in ascending order of views.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Music_Name" FROM "Music"
        ORDER BY "Views";
```

Snapshot :



Result Contains :

500 Tuples.

17)

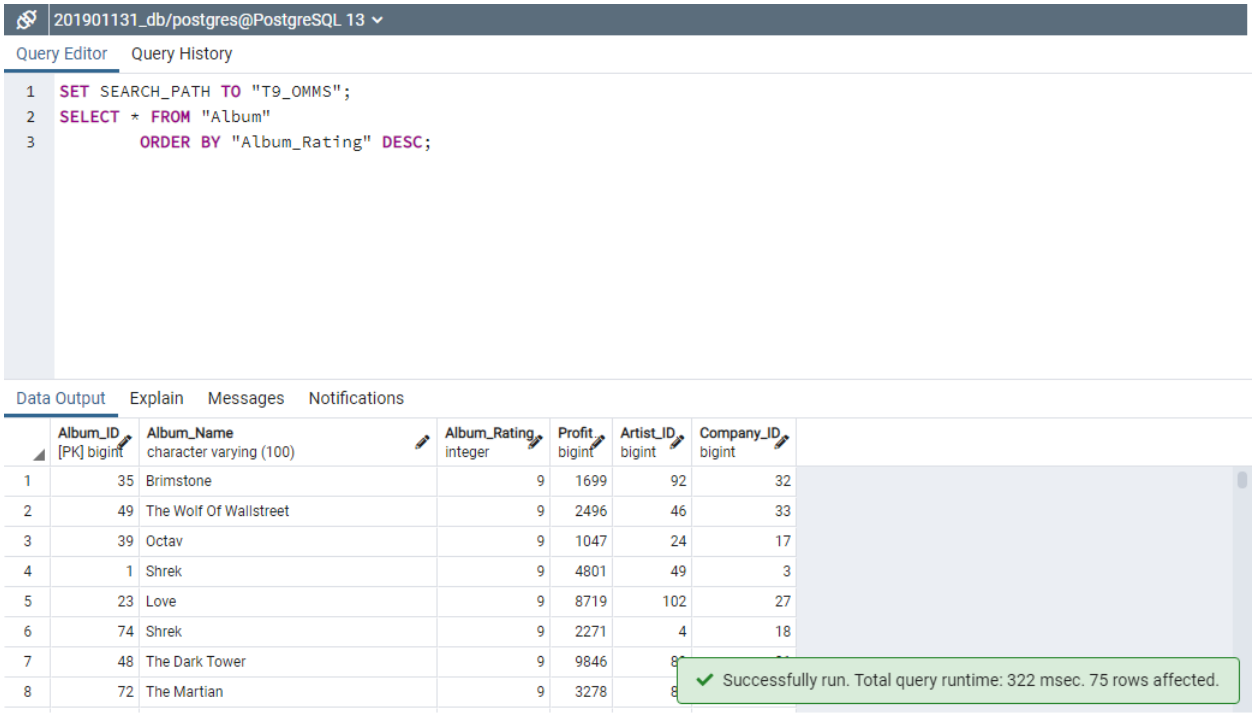
Plain English Query :

List the albums in descending order of rating.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT * FROM "Album"
        ORDER BY "Album_Rating" DESC;
```

Snapshot :



Result Contains :

75 Tuples.

18)

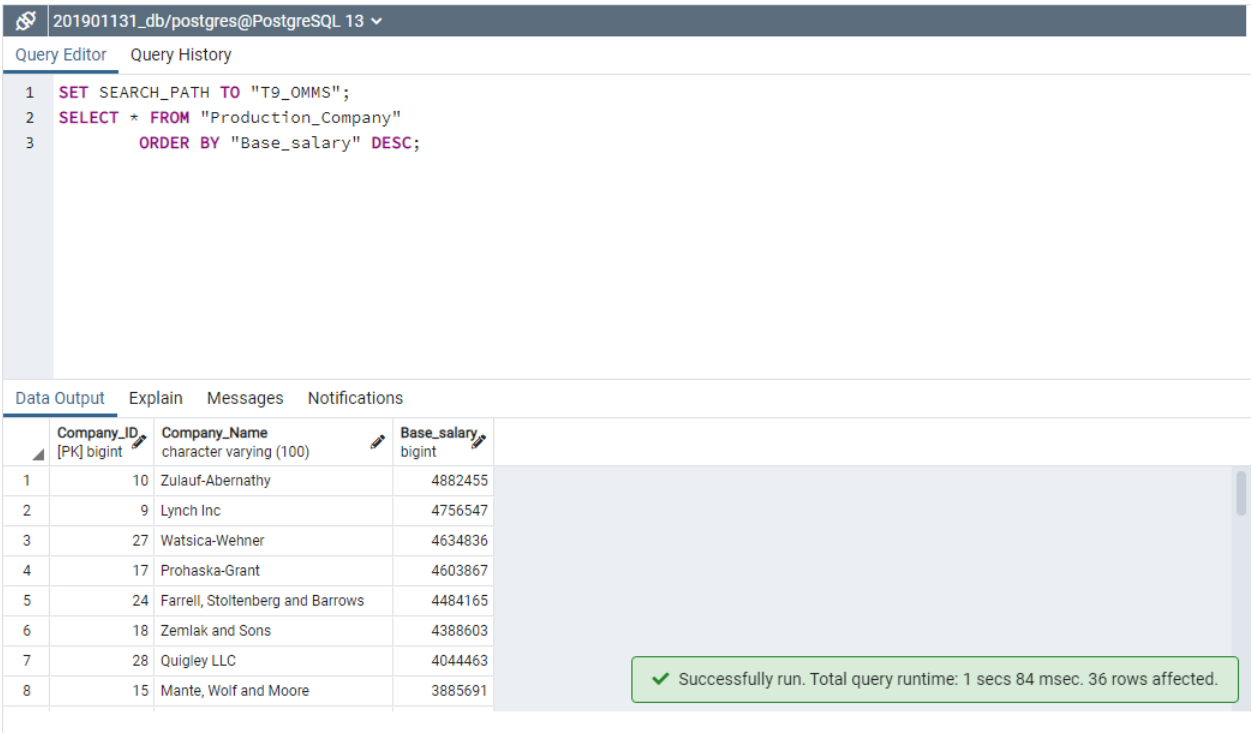
Plain English Query :

List the company in descending order of base salary.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT * FROM "Production_Company"
        ORDER BY "Base_salary" DESC;
```

Snapshot :



Result Contains :

36 Tuples.

19)

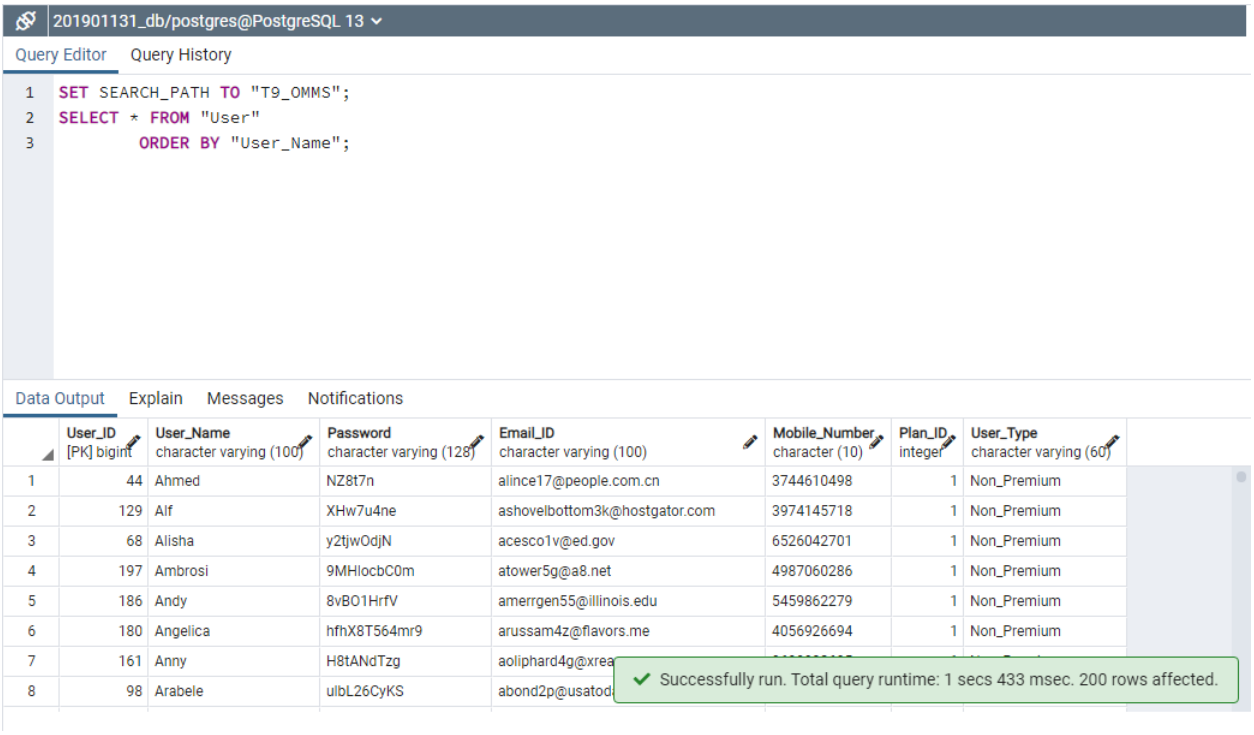
Plain English Query :

List the User in alphabetical order.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT * FROM "User"
        ORDER BY "User_Name";
```

Snapshot :



201901131_db/postgres@PostgreSQL 13

Query Editor | Query History

```
1  SET SEARCH_PATH TO "T9_OMMS";
2  SELECT * FROM "User"
3          ORDER BY "User_Name";
```

Data Output | Explain | Messages | Notifications

	User_ID [PK] bigint	User_Name character varying (100)	Password character varying (128)	Email_ID character varying (100)	Mobile_Number character (10)	Plan_ID integer	User_Type character varying (60)
1	44	Ahmed	NZ8t7n	alince17@people.com.cn	3744610498	1	Non_Premium
2	129	Alf	XHw7u4ne	ashovelbottom3k@hostgator.com	3974145718	1	Non_Premium
3	68	Alisha	y2tjwOdjN	acesco1v@ed.gov	6526042701	1	Non_Premium
4	197	Ambrosi	9MHlocbC0m	atower5g@a8.net	4987060286	1	Non_Premium
5	186	Andy	8vBO1HrfV	amerrgen55@illinois.edu	5459862279	1	Non_Premium
6	180	Angelica	hfhX8T564mr9	arussam4z@flavors.me	4056926694	1	Non_Premium
7	161	Anny	H8tANdTzg	aoliphard4g@xrea			
8	98	Arabele	ulbL26CyKS	abond2p@usatod			

Successfully run. Total query runtime: 1 secs 433 msec. 200 rows affected.

Result Contains :

200 Tuples.

20)

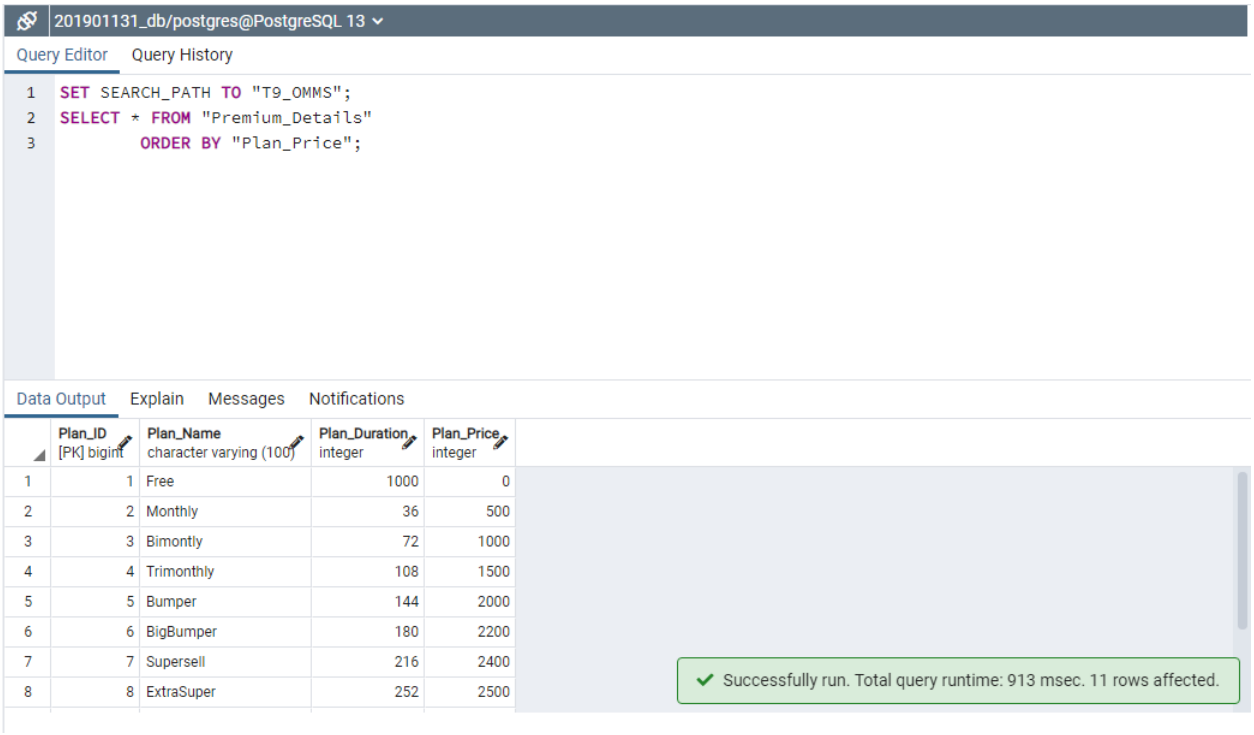
Plain English Query :

List the Plan in ascending order of plan price.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT * FROM "Premium_Details"
        ORDER BY "Plan_Price";
```

Snapshot :



	Plan_ID [PK] bigint	Plan_Name character varying (100)	Plan_Duration integer	Plan_Price integer
1	1	Free	1000	0
2	2	Monthly	36	500
3	3	Bimontly	72	1000
4	4	Trimonthly	108	1500
5	5	Bumper	144	2000
6	6	BigBumper	180	2200
7	7	Supersell	216	2400
8	8	ExtraSuper	252	2500

Result Contains :

11 Tuples.

21)

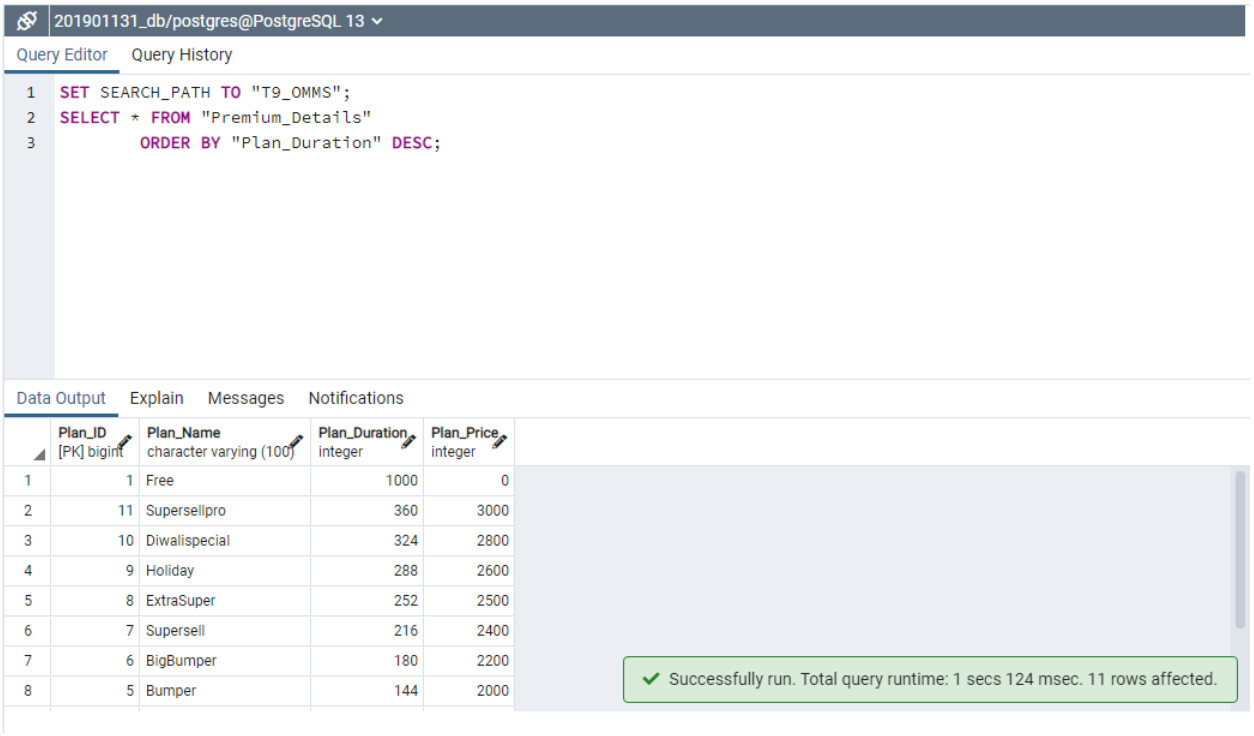
Plain English Query :

List the Plan in descending order of plan duration.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT * FROM "Premium_Details"
        ORDER BY "Plan_Duration" DESC
```

Snapshot :



Result Contains :

11 Tuples.

22)

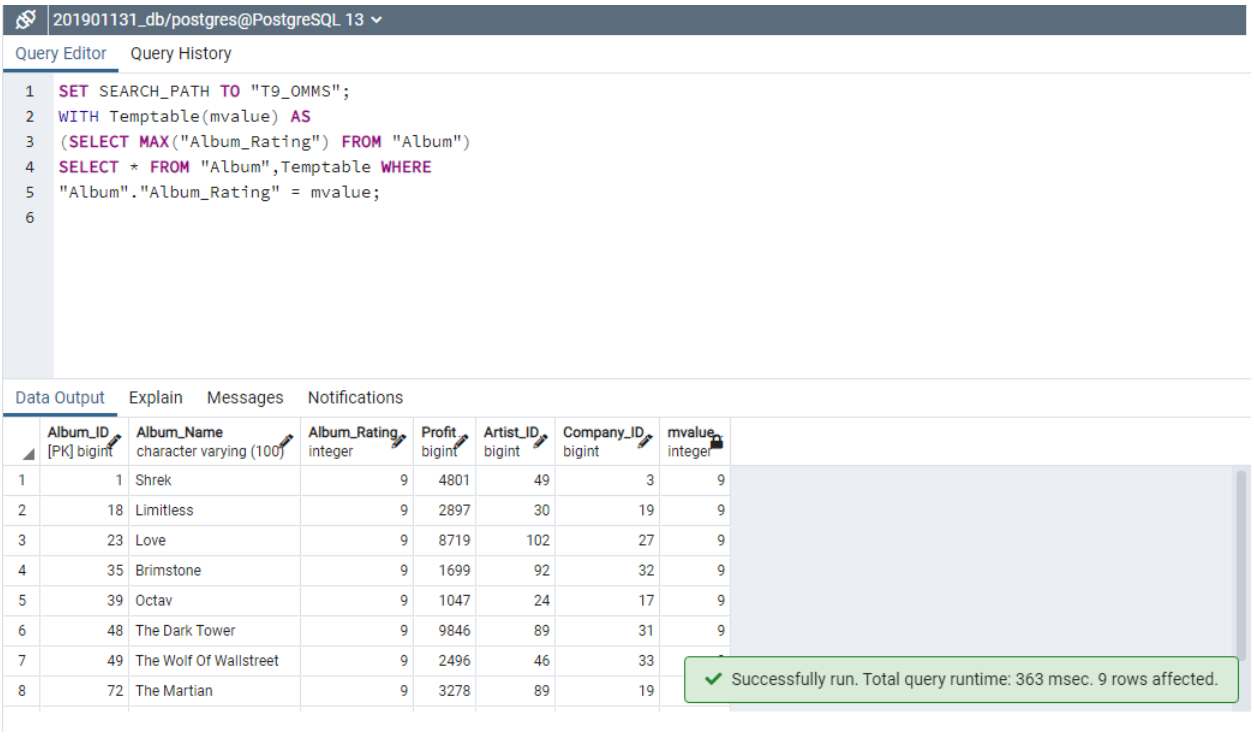
Plain English Query :

List the albums which have maximum rating.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
WITH Temptable(mvalue) AS
(SELECT MAX("Album_Rating") FROM "Album")
SELECT * FROM "Album",Temptable WHERE
"Album"."Album_Rating" = mvalue;
```

Snapshot :



Result Contains :

9 Tuples.

23)

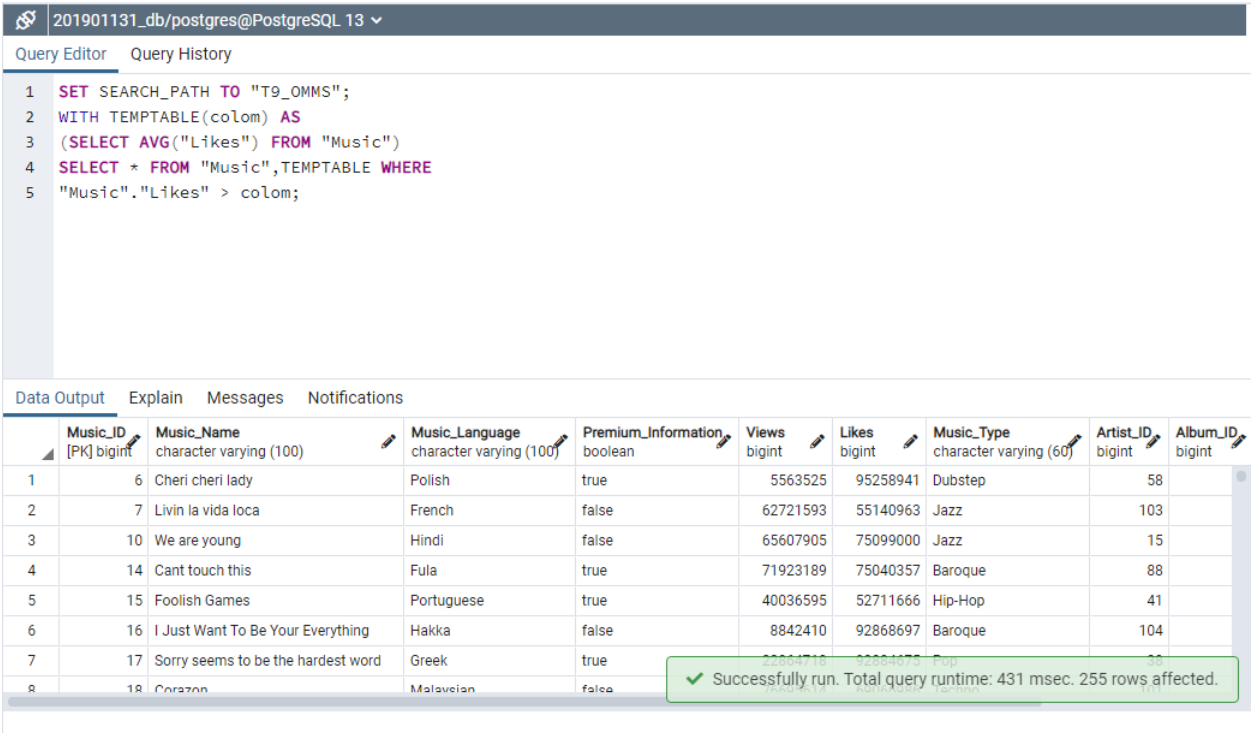
Plain English Query :

List songs which have more likes than average likes.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
WITH TEMPTABLE(colom) AS
(SELECT AVG("Likes") FROM "Music")
SELECT * FROM "Music",TEMPTABLE WHERE
"Music"."Likes" > colom;
```

Snapshot :



	Music_ID [PK] bigint	Music_Name character varying (100)	Music_Language character varying (100)	Premium_Information boolean	Views bigint	Likes bigint	Music_Type character varying (60)	Artist_ID bigint	Album_ID bigint
1	6	Cheri cheri lady	Polish	true	5563525	95258941	Dubstep	58	
2	7	Livin la vida loca	French	false	62721593	55140963	Jazz	103	
3	10	We are young	Hindi	false	65607905	75099000	Jazz	15	
4	14	Cant touch this	Fula	true	71923189	75040357	Baroque	88	
5	15	Foolish Games	Portuguese	true	40036595	52711666	Hip-Hop	41	
6	16	I Just Want To Be Your Everything	Hakka	false	8842410	92868697	Baroque	104	
7	17	Sorry seems to be the hardest word	Greek	true					
8	18	Corazon	Malaysian	false					

Result Contains :

255 Tuples.

24)

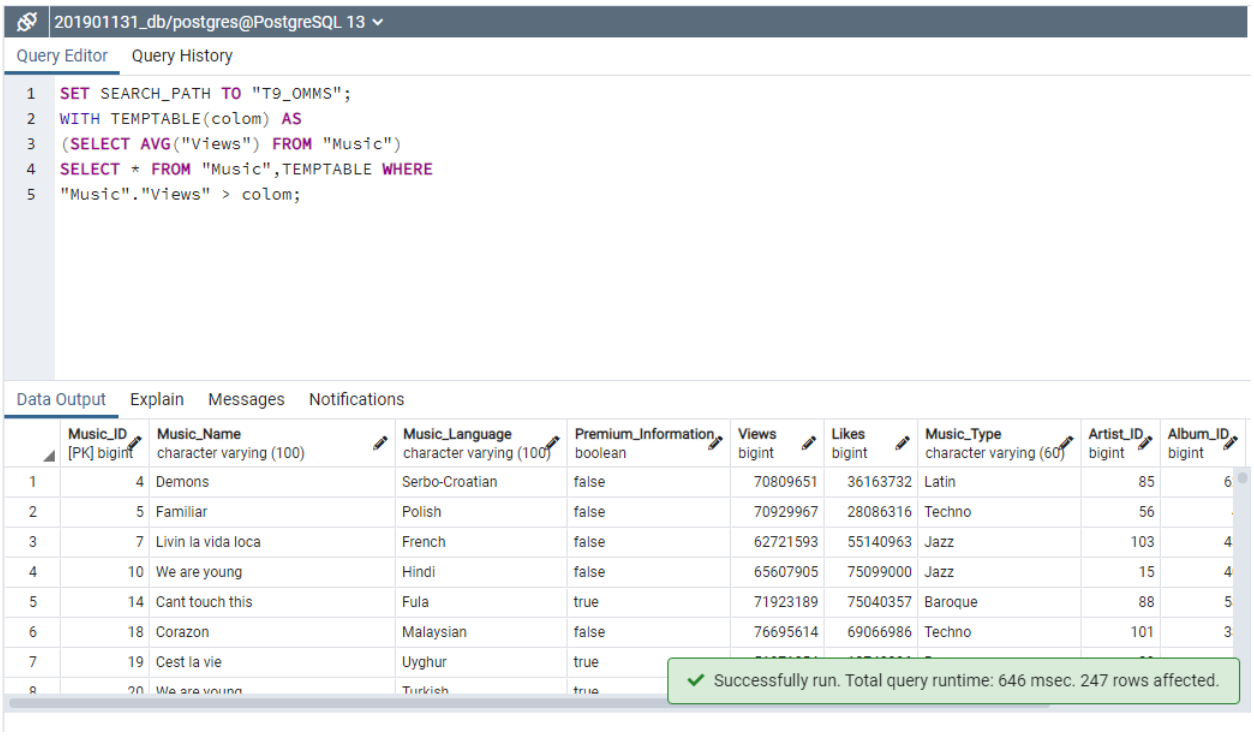
Plain English Query :

List songs which have more views than average views.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
WITH TEMPTABLE(colom) AS
(SELECT AVG("Views") FROM "Music")
SELECT * FROM "Music",TEMPTABLE WHERE
"Music"."Views" > colom;
```

Snapshot :



Result Contains :

247 Tuples.

25)

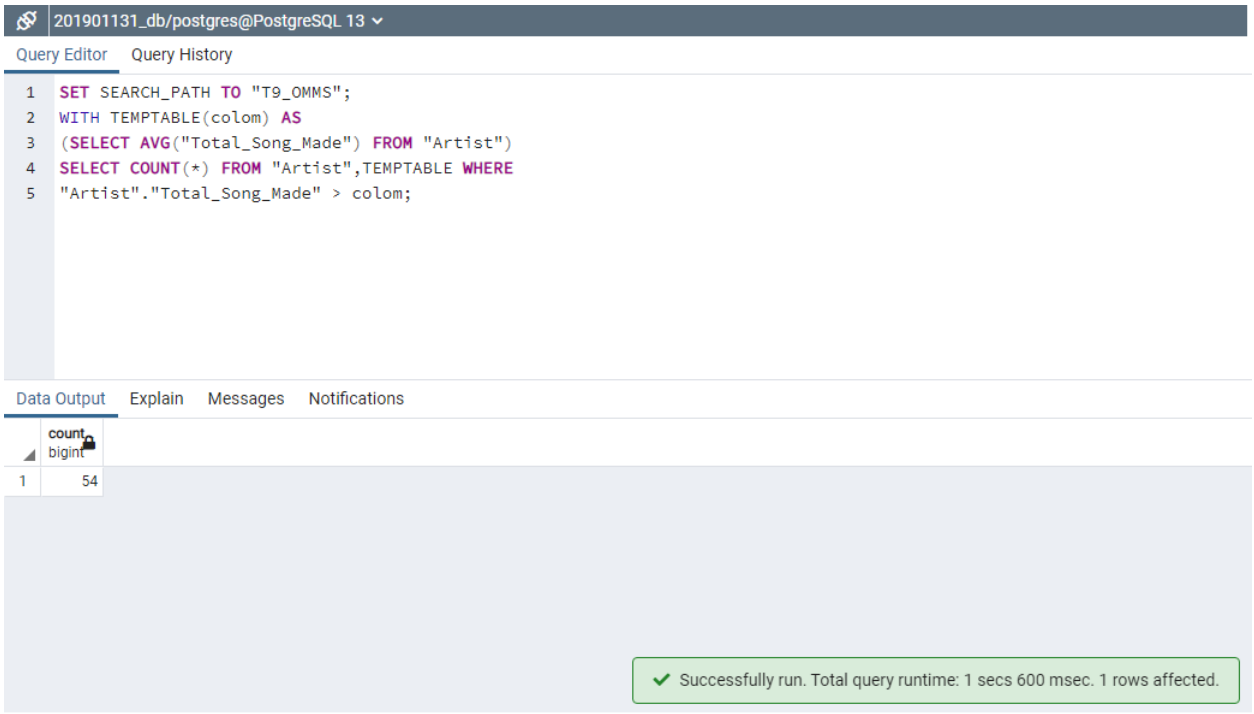
Plain English Query :

List the number of artists which total made the songs greater than average songs.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
WITH TEMPTABLE(colom) AS
(SELECT AVG("Total_Song_Made") FROM "Artist")
SELECT COUNT(*) FROM "Artist",TEMPTABLE WHERE
"Artist"."Total_Song_Made" > colom;
```

Snapshot :



Result Contains :

1 Tuple.

26)

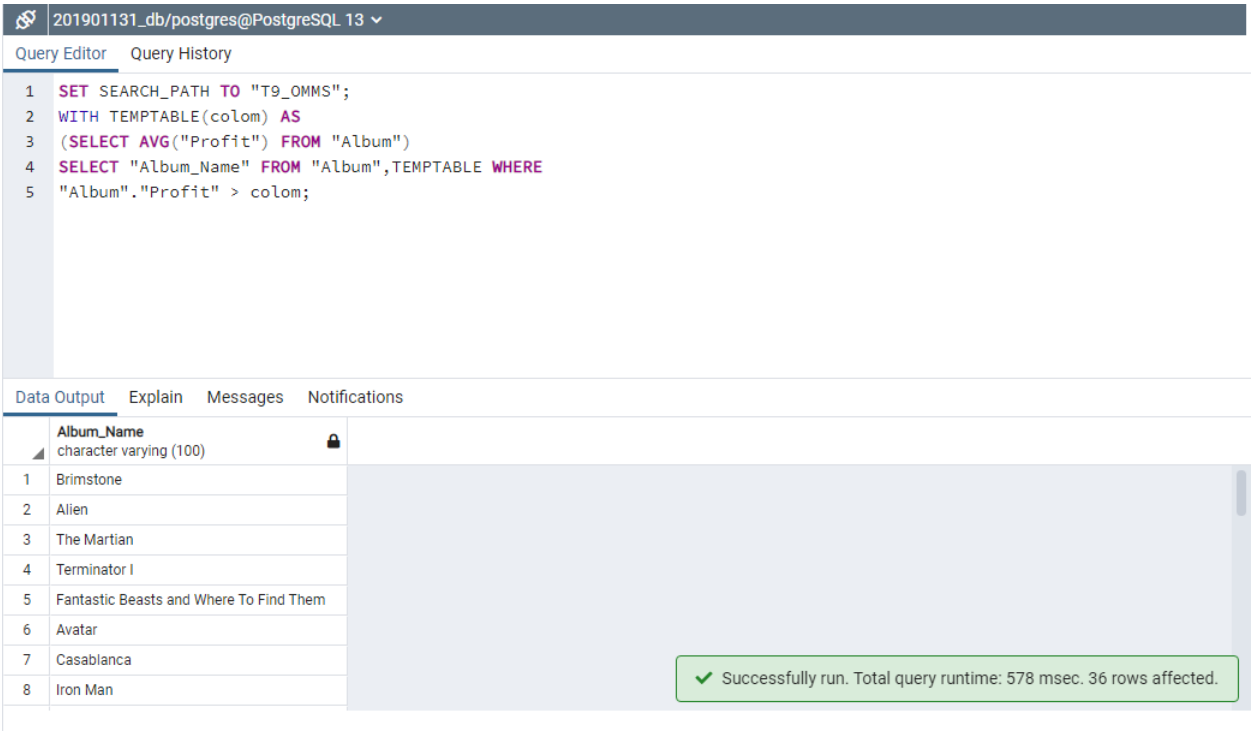
Plain English Query :

List the name of albums which have profit greater than average profit.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
WITH TEMPTABLE(colom) AS
(SELECT AVG("Profit") FROM "Album")
SELECT "Album_Name" FROM "Album",TEMPTABLE WHERE
"Album"."Profit" > colom;
```

Snapshot :



Result Contains :

36 Tuples.

27)

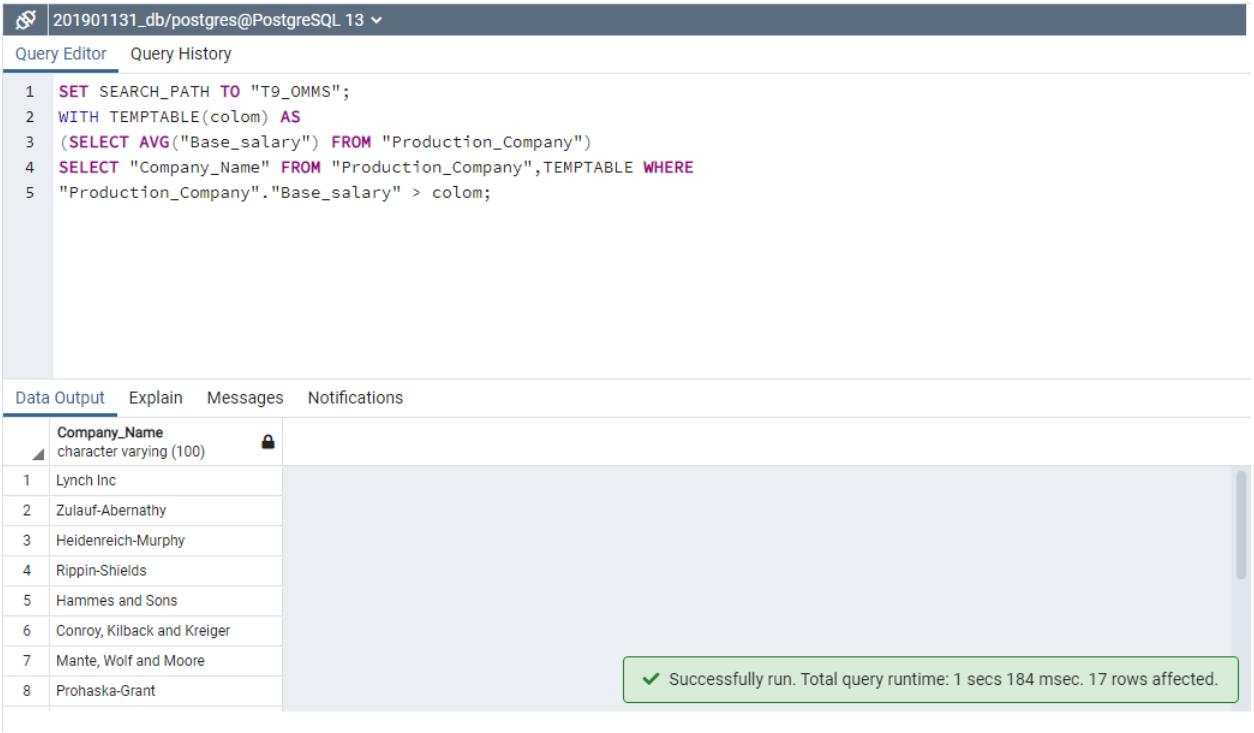
Plain English Query :

List the Companies which have a base salary greater than average base salary.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
WITH TEMPTABLE(colom) AS
(SELECT AVG("Base_salary") FROM "Production_Company")
SELECT "Company_Name" FROM "Production_Company",TEMPTABLE WHERE
"Production_Company"."Base_salary" > colom;
```

Snapshot :



Result Contains :

17 Tuples.

28)

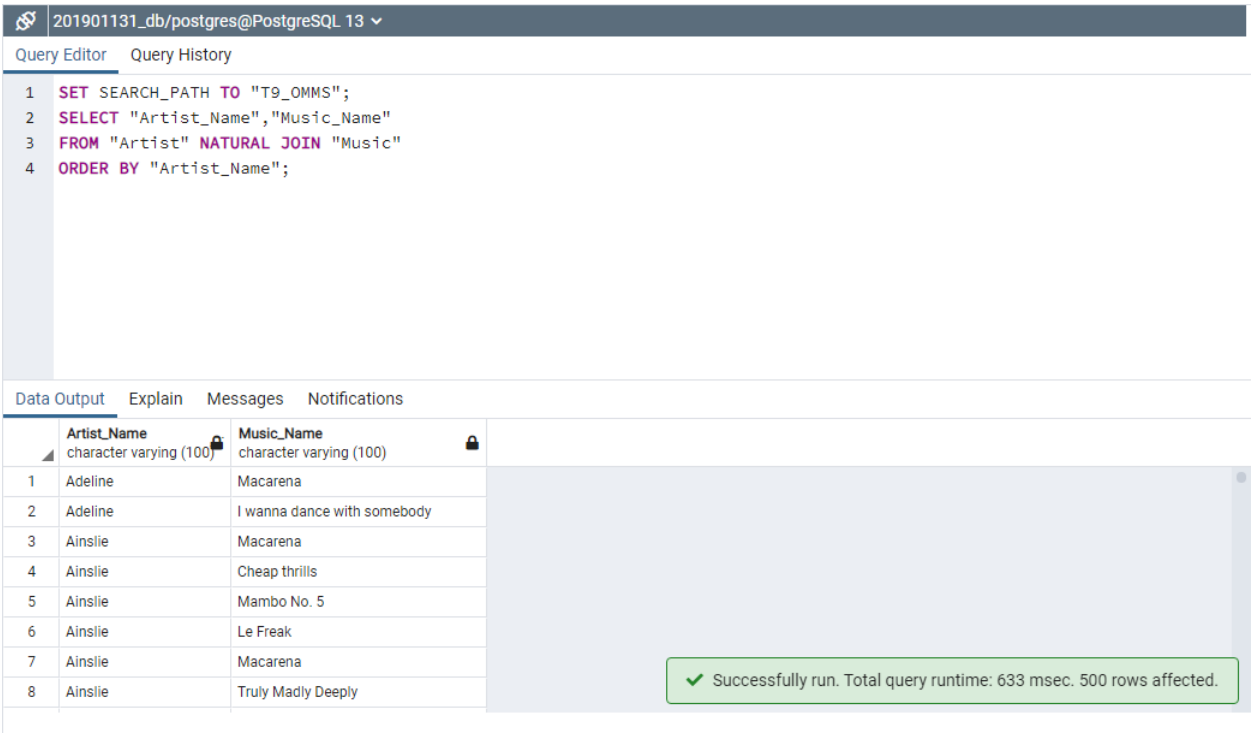
Plain English Query :

List the Artist name along with the song name he has made and order them by Artist name.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Artist_Name","Music_Name"
FROM "Artist" NATURAL JOIN "Music"
ORDER BY "Artist_Name";
```

Snapshot :



Result Contains :

500 Tuples.

29)

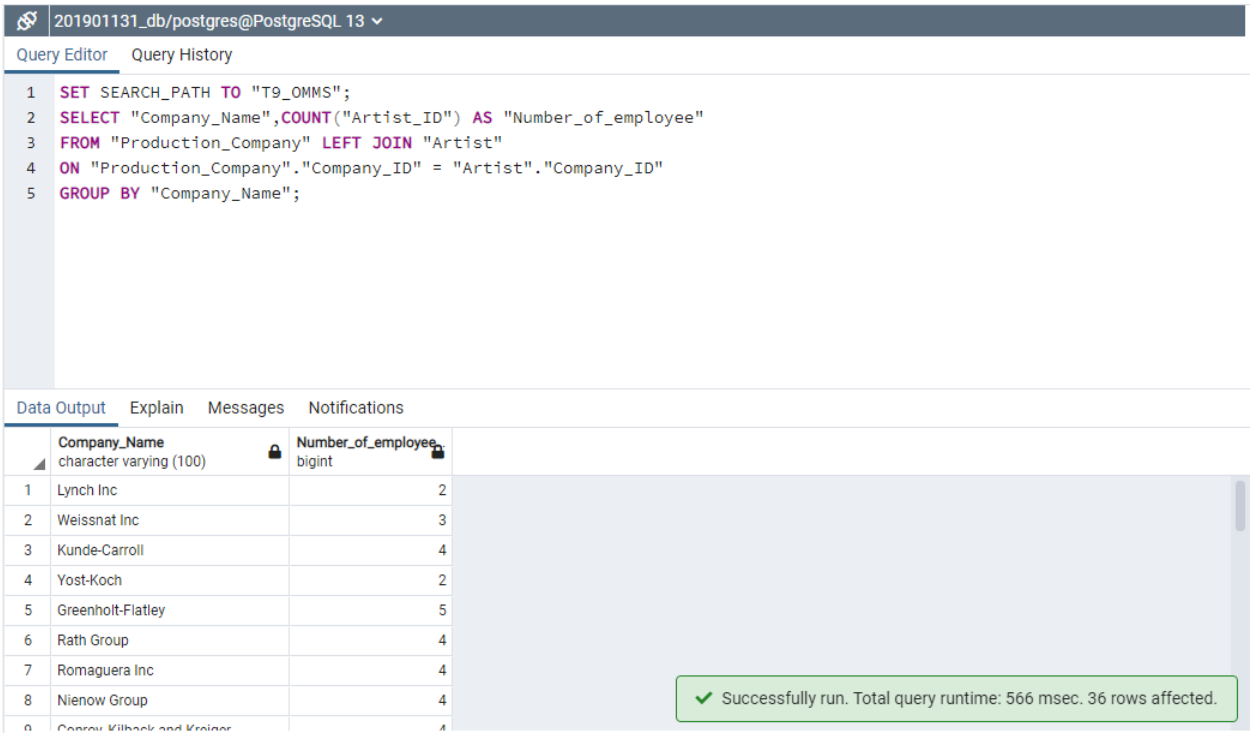
Plain English Query :

List the company name along with number of employees.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Company_Name",COUNT("Artist_ID") AS "Number_of_employee"
FROM "Production_Company" LEFT JOIN "Artist"
ON "Production_Company"."Company_ID" = "Artist"."Company_ID"
GROUP BY "Company_Name";
```

Snapshot :



Result Contains :

36 Tuples.

30)

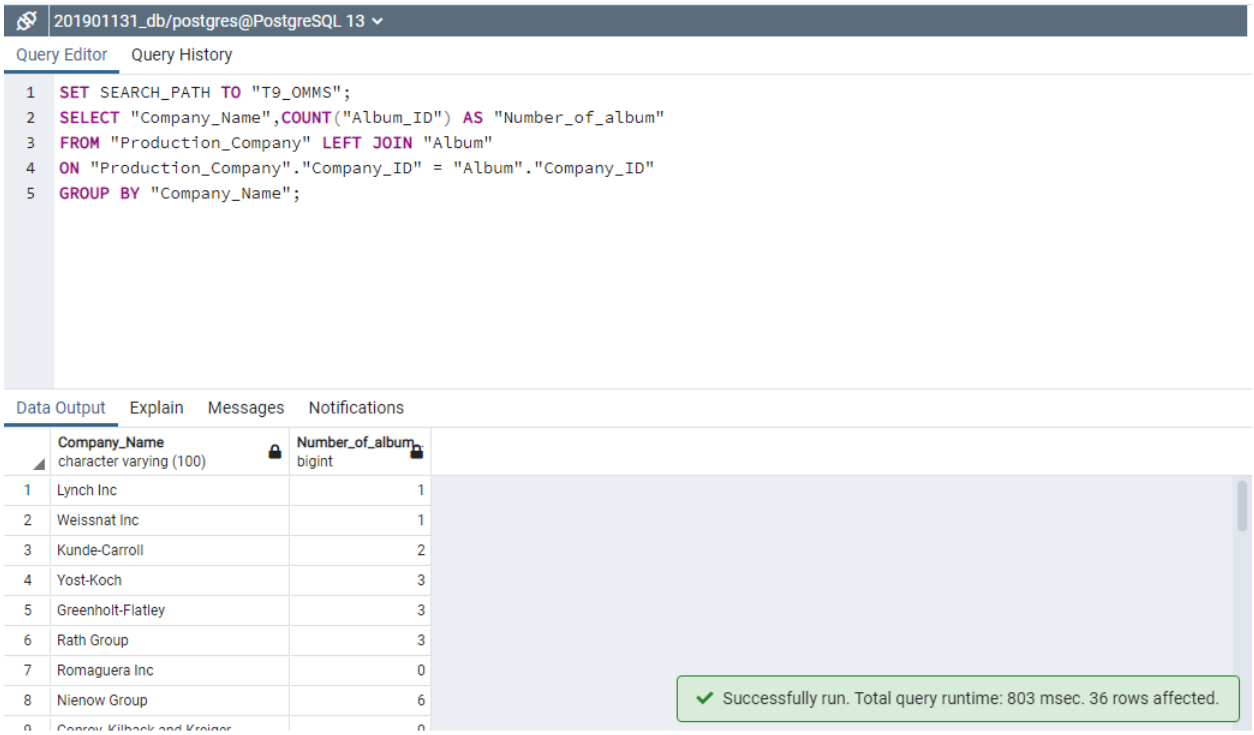
Plain English Query :

List the company name along with the number of albums.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Company_Name",COUNT("Album_ID") AS "Number_of_album"
FROM "Production_Company" LEFT JOIN "Album"
ON "Production_Company"."Company_ID" = "Album"."Company_ID"
GROUP BY "Company_Name";
```

Snapshot :



Result Contains :

36 Tuples.

31)

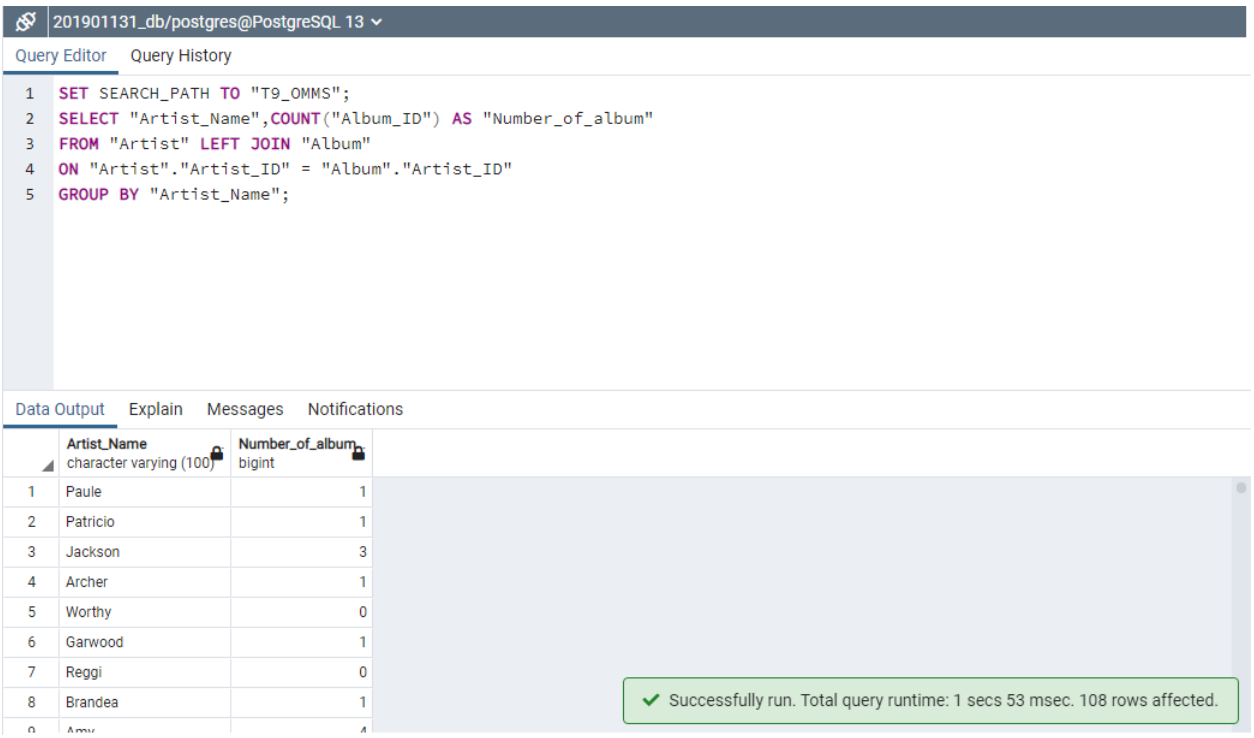
Plain English Query :

List artist along with number of albums.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Artist_Name",COUNT("Album_ID") AS "Number_of_album"
FROM "Artist" LEFT JOIN "Album"
ON "Artist"."Artist_ID" = "Album"."Artist_ID"
GROUP BY "Artist_Name";
```

Snapshot :



Result Contains :

108 Tuples.

32)

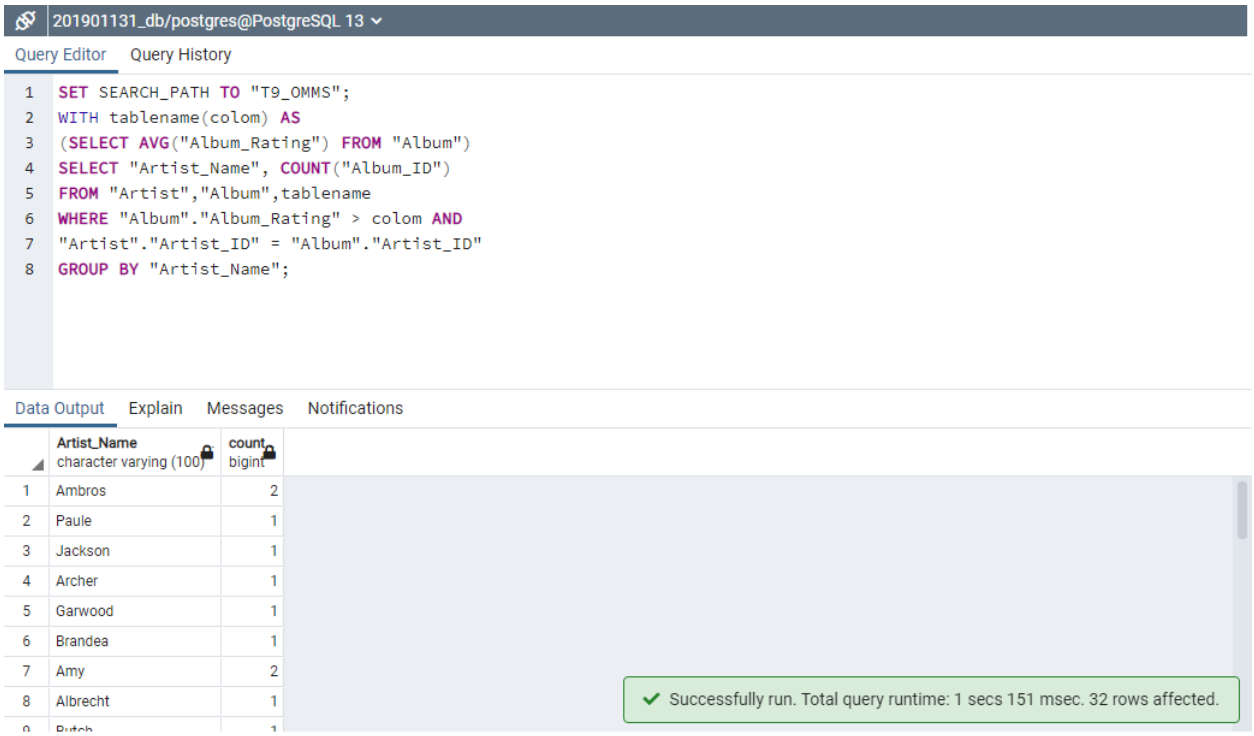
Plain English Query :

List artist along with number of albums which have ratings greater than average rating.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
WITH tablename(colom) AS
(SELECT AVG("Album_Rating") FROM "Album")
SELECT "Artist_Name", COUNT("Album_ID")
FROM "Artist","Album",tablename
WHERE "Album"."Album_Rating" > colom AND
"Artist"."Artist_ID" = "Album"."Artist_ID"
GROUP BY "Artist_Name";
```

Snapshot :



Result Contains :

32 Tuples.

33)

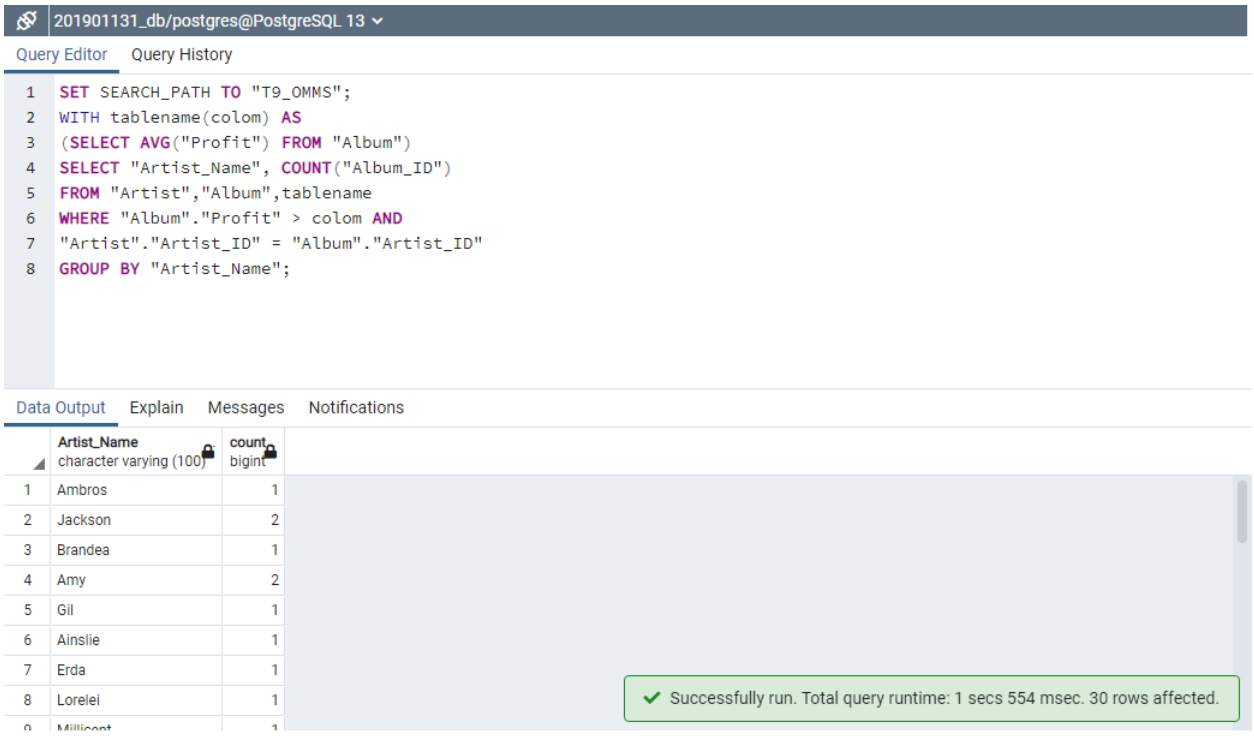
Plain English Query :

List artist along with number of profit which have profit greater than average profit.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
WITH tablename(colom) AS
(SELECT AVG("Profit") FROM "Album")
SELECT "Artist_Name", COUNT("Album_ID")
FROM "Artist","Album",tablename
WHERE "Album"."Profit" > colom AND
"Artist"."Artist_ID" = "Album"."Artist_ID"
GROUP BY "Artist_Name";
```

Snapshot :



Result Contains :

30 Tuples.

34)

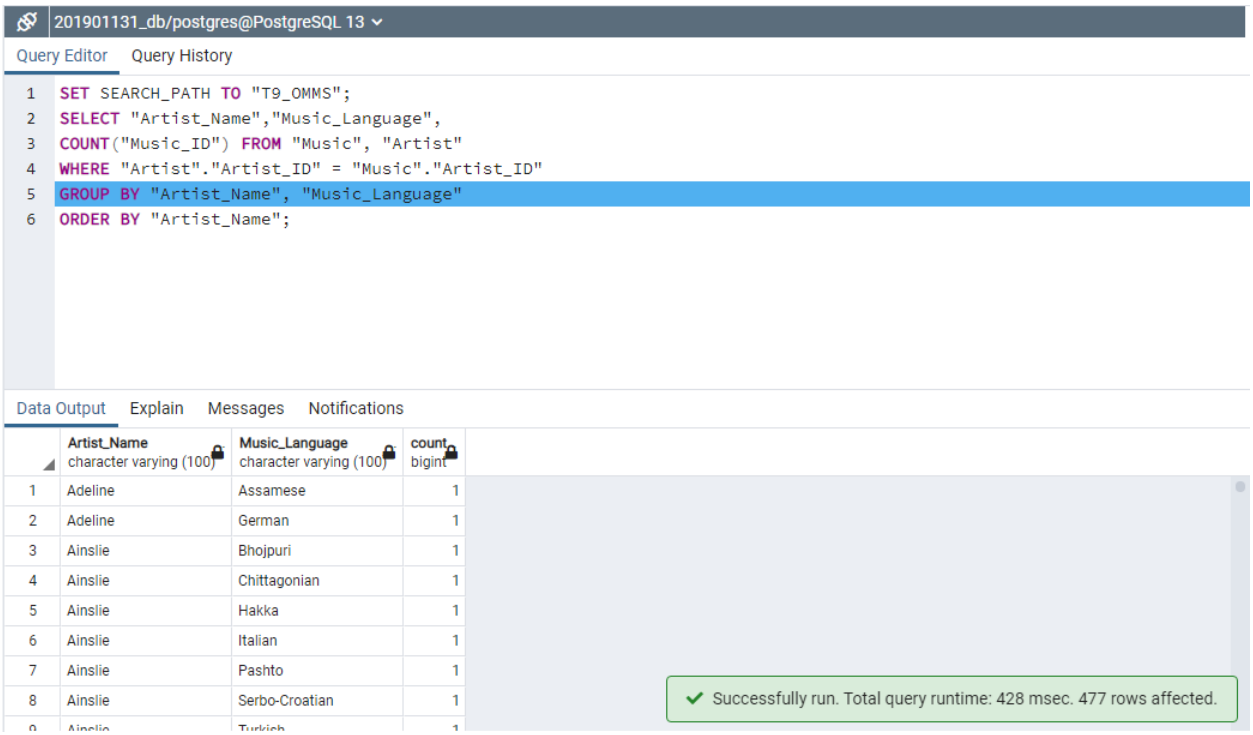
Plain English Query :

List artist along with number of music made in each language.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Artist_Name","Music_Language",
COUNT("Music_ID") FROM "Music", "Artist"
WHERE "Artist"."Artist_ID" = "Music"."Artist_ID"
GROUP BY "Artist_Name", "Music_Language"
ORDER BY "Artist_Name";
```

Snapshot :



Result Contains :

477 Tuples.

35)

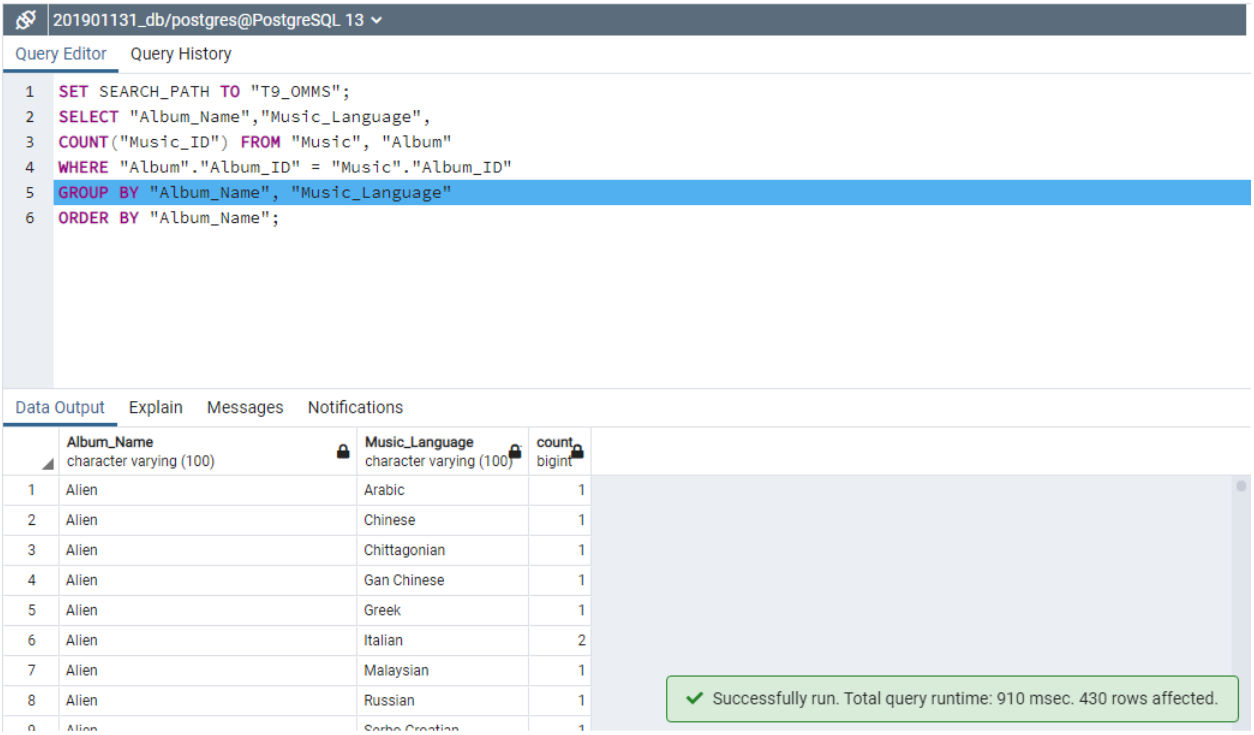
Plain English Query :

List album along with number of music in each language.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Album_Name","Music_Language",
COUNT("Music_ID") FROM "Music", "Album"
WHERE "Album"."Album_ID" = "Music"."Album_ID"
GROUP BY "Album_Name", "Music_Language"
ORDER BY "Album_Name";
```

Snapshot :



Result Contains :

430 Tuples.

36)

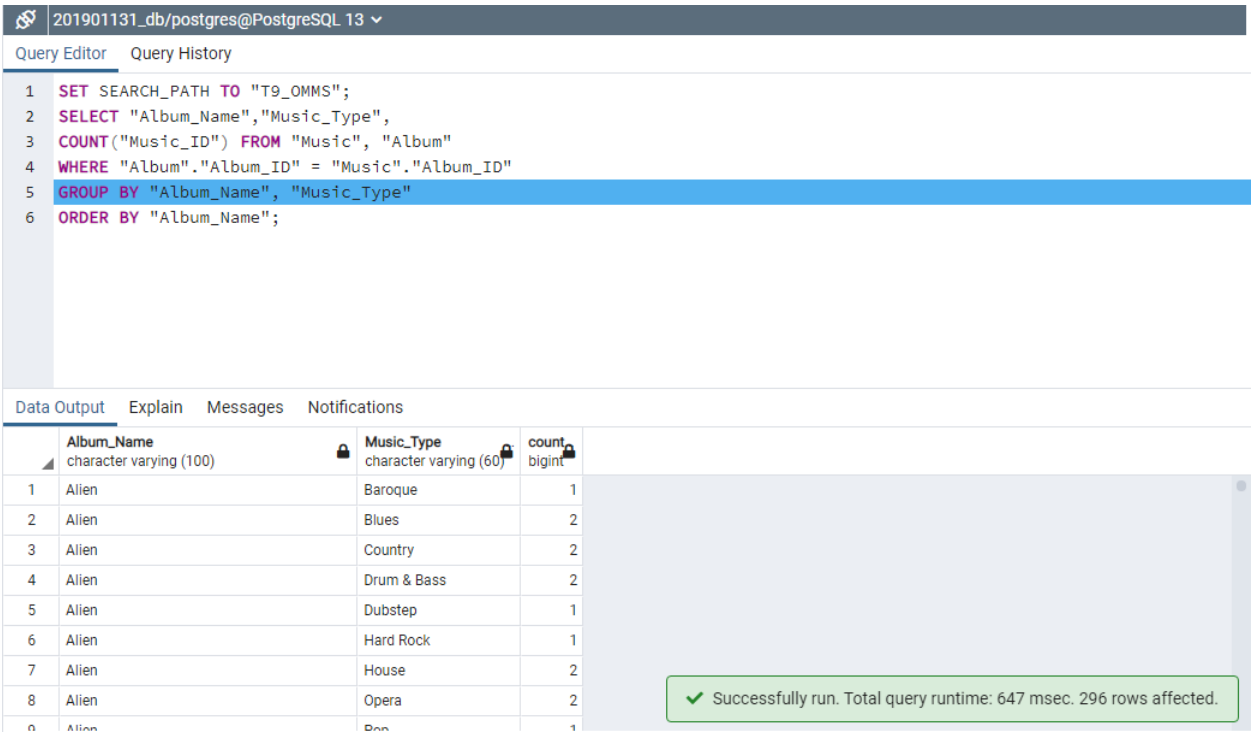
Plain English Query :

List albums along with a number of music types(pop, rope, HIp-Hop, etc).

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Album_Name","Music_Type",
COUNT("Music_ID") FROM "Music", "Album"
WHERE "Album"."Album_ID" = "Music"."Album_ID"
GROUP BY "Album_Name", "Music_Type"
ORDER BY "Album_Name";
```

Snapshot :



Result Contains :

296 Tuples.

37)

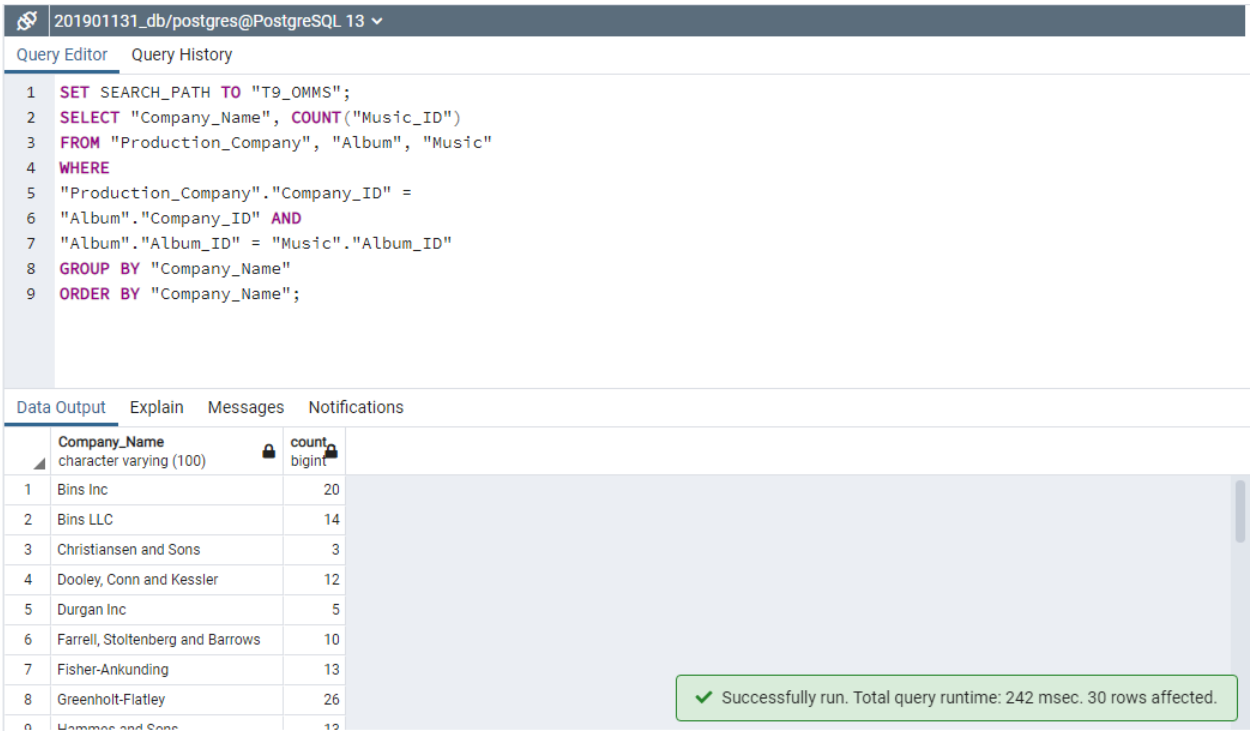
Plain English Query :

List company name along with number of produced songs.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Company_Name", COUNT("Music_ID")
FROM "Production_Company", "Album", "Music"
WHERE
"Production_Company"."Company_ID" =
"Album"."Company_ID" AND
"Album"."Album_ID" = "Music"."Album_ID"
GROUP BY "Company_Name"
ORDER BY "Company_Name";
```

Snapshot :



Result Contains :

30 Tuples.

38)

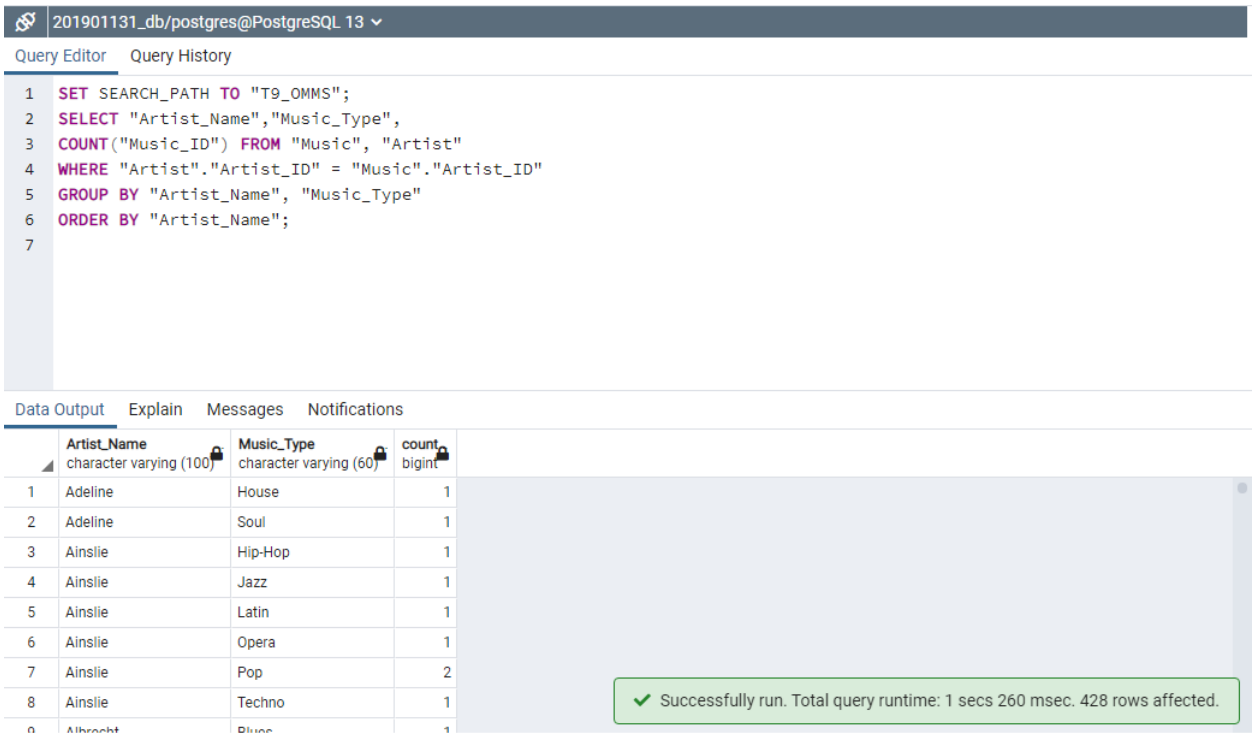
Plain English Query :

List artist along with a number of music types(pop, rope, HIp-Hop, etc).

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
SELECT "Artist_Name","Music_Type",
COUNT("Music_ID") FROM "Music", "Artist"
WHERE "Artist"."Artist_ID" = "Music"."Artist_ID"
GROUP BY "Artist_Name", "Music_Type"
ORDER BY "Artist_Name";
```

Snapshot :



Result Contains :

428 Tuples.

39)

Plain English Query :

Show plan details using a function.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
CREATE OR REPLACE FUNCTION "T9_OMMS".show_plans()
RETURNS TABLE(PID BIGINT, PName CHAR VARYING(100), Duration INT, Price INT)
LANGUAGE 'plpgsql'

AS $BODY$
DECLARE
tempvar record;

BEGIN

FOR tempvar in (
        SELECT "Plan_ID", "Plan_Name", "Plan_Duration", "Plan_Price" FROM
"T9_OMMS"."Premium_Details"
)
LOOP
PID := tempvar."Plan_ID";
PName := tempvar."Plan_Name";
Duration := tempvar."Plan_Duration";
Price := tempvar."Plan_Price";
Return next;
end loop;
END;
$BODY$;
SELECT * FROM show_plans();
```

Snapshot :

201901131_db/postgres@PostgreSQL 13

Query Editor | Query History

```
1  SET SEARCH_PATH TO "T9_OMMS";
2  CREATE OR REPLACE FUNCTION "T9_OMMS".show_plans()
3  RETURNS TABLE(PID BIGINT, PName CHAR VARYING(100), Duration INT, Price INT)
4  LANGUAGE 'plpgsql'
5
6  AS $BODY$
7  DECLARE
8  tempvar record;
9
10 BEGIN
11
12 FOR tempvar in (
```

Data Output | Explain | Messages | Notifications

	pid bigint	pname character varying	duration integer	price integer
1	1	Free	1000	0
2	2	Monthly	36	500
3	3	Bimontly	72	1000
4	4	Trimonthly	108	1500
5	5	Bumper	144	2000
6	6	BigBumper	180	2200
7	7	Supersell	216	2400
8	8	ExtraSuper	252	2500

Successfully run. Total query runtime: 521 msec. 11 rows affected.

Result Contains :

11 Tuples

40)

Plain English Query :

Create a Trigger function to do the transaction.

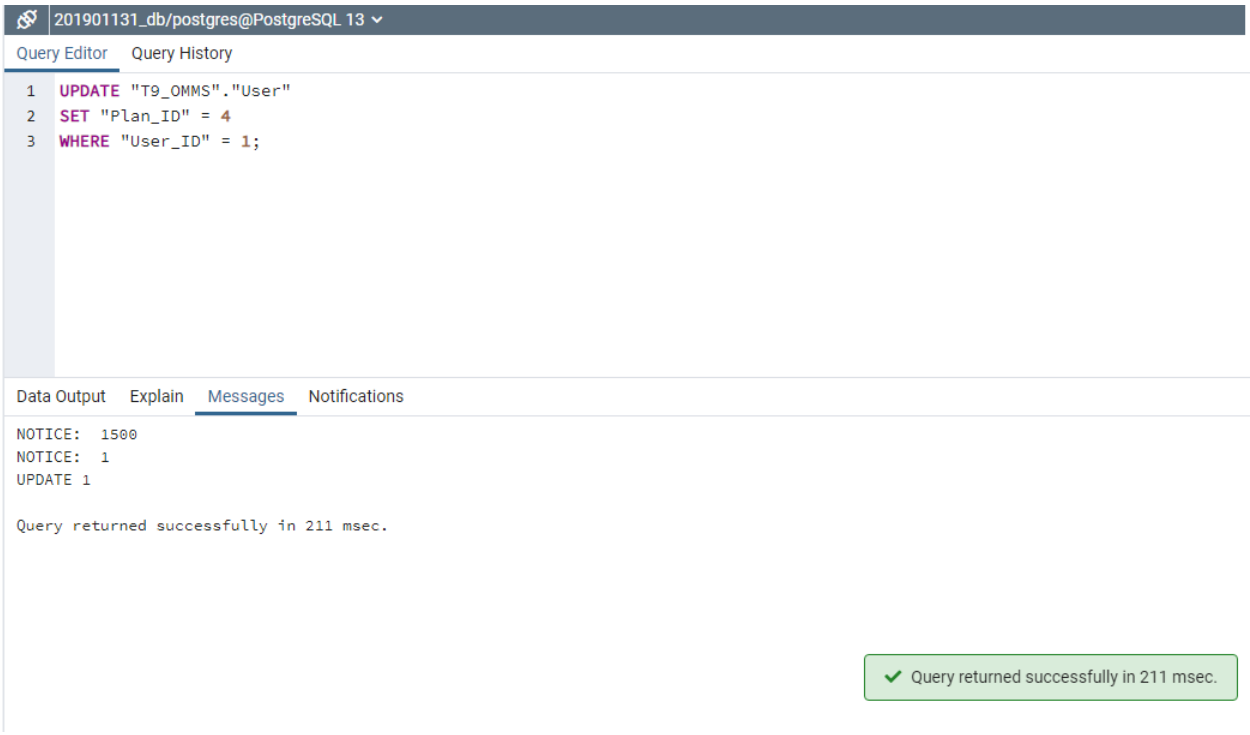
SQL Query :

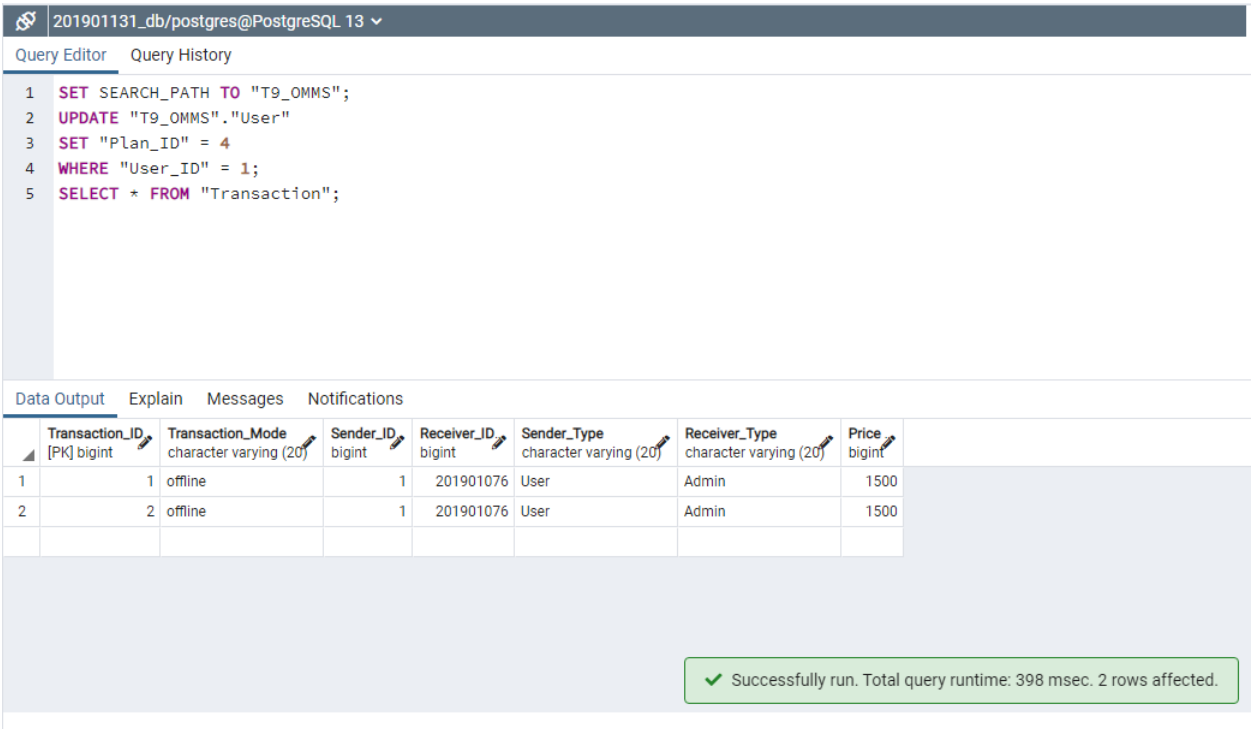
```
SET SEARCH_PATH TO "T9_OMMS";
CREATE OR REPLACE FUNCTION buy_plan()
RETURNS TRIGGER
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
price integer;
number_of_transaction integer;

BEGIN

SET SEARCH_PATH TO "T9_OMMS";
SELECT "Plan_Price" INTO price
FROM "Premium_Details"
WHERE "Plan_ID" = NEW."Plan_ID";
RAISE NOTICE '%', price;

SELECT COUNT("Transaction_ID") INTO
number_of_transaction FROM "Transaction";
RAISE NOTICE '%', number_of_transaction;

INSERT INTO "Transaction"(
"Transaction_ID","Transaction_Mode",
"Sender_ID", "Receiver_ID",
"Sender_Type","Receiver_Type", "Price")
VALUES (number_of_transaction+1,'offline',NEW."User_ID",201901076,'User','Admin',price);
RETURN NULL;
END;
$BODY$;

CREATE TRIGGER "Buy_Plan" AFTER INSERT OR UPDATE
ON "T9_OMMS"."User" FOR EACH ROW EXECUTE FUNCTION "T9_OMMS".buy_plan();
```

Check on update

```
SET SEARCH_PATH TO "T9_OMMS";
UPDATE "T9_OMMS"."User"
SET "Plan_ID" = 4
WHERE "User_ID" = 1;
SELECT * FROM "Transaction";
```

Snapshot :





Result Contains :

2 Rows affected.

41)

Plain English Query :

Create a trigger function to update User Type.

SQL Query :

```
SET SEARCH_PATH TO "T9_OMMS";
CREATE OR REPLACE FUNCTION set_user_type()
RETURNS TRIGGER
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN

IF (NEW."Plan_ID" > 1) THEN
        NEW."User_Type" = 'Premium';
ELSE
        NEW."User_Type" = 'Free';
END IF;

RETURN NEW;
END;
$BODY$;

CREATE TRIGGER "Set_User_Type"
BEFORE UPDATE ON "T9_OMMS"."User"
FOR EACH ROW EXECUTE FUNCTION "T9_OMMS".set_user_type();
```

Check on Update

```
SET SEARCH_PATH TO "T9_OMMS";
UPDATE "T9_OMMS"."User"
SET "User_Type" = 'Premium'
WHERE "User_ID" = 1;
```

Snapshot :

201901131_db/postgres@PostgreSQL 13

Query Editor | Query History

```
1  SET SEARCH_PATH TO "T9_OMMS";
2  UPDATE "T9_OMMS"."User"
3  SET "User_Type" = 'Premium'
4  WHERE "User_ID" = 1;
```

Data Output | Explain | Messages | Notifications

```
NOTICE:  1500
NOTICE:  4
UPDATE 1

Query returned successfully in 240 msec.
```

Query returned successfully in 240 msec.

201901131_db/postgres@PostgreSQL 13

Query Editor | Query History

```
1  SET SEARCH_PATH TO "T9_OMMS";
2  SELECT * FROM "User";
```

Data Output | Explain | Messages | Notifications

	User_ID [PK] bigint	User_Name character varying (100)	Password character varying (128)	Email_ID character varying (100)	Mobile_Number character (10)	Plan_ID integer	User_Type character varying (60)
194	195	Gustaf	XLMM6tFsn	gmarlowe5e@furl.net	2606675022	1	Non_Premium
195	196	Hulda	xLtOT23l	hdanilovich5f@webnode.com	7442878729	1	Non_Premium
196	197	Ambrosi	9MHlocbC0m	atower5g@a8.net	4987060286	1	Non_Premium
197	198	Billie	mrfrBag1T0	bcatlow5h@amazon.de	2955457234	1	Non_Premium
198	199	Odie	R2tMcK39	odales5i@arizona.edu	9286344090	1	Non_Premium
199	200	Cristy	HcHMSN	curridge5j@pbs.org	9496660916	1	Non_Premium
200	1	Melodee	7zj0m93xWj	martrick0@jigsy.com	2276321528	4	Premium

Result Contains :

200 Tuples..