LAB 10
Art Gallery App
Final Submission

ID:202001169 Name:Thakor Harshal Dipaksinh
ID:202001172 Name: Patel Vedant

Group:3
Section:10
Team:6

- **Final SRS:**

1. Purpose

The primary purpose of an art gallery app is to showcase artists, promote their art, and expose or present that to the public, collectors, media, and institutions.

To get information about art and an artist and more connected to the artwork. And make the art gallery more accessible and promote it to buyers. To make available art galleries can be accessed from anywhere at any time.

In the real World, an Art Gallery app can be very useful.It can be a mediocre between artist and the buyer and viewer to represent the art of the artist.We can also design virtual reality art Gallery by making software for that.

2. Intended Audience and Reading Suggestions

- **The document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers and Users who want information or want to buy or rent public art like music,video,drawing, sculptures.**
- **Users care about art**
- **Users want to see public art**

- **Users like to comment on public art and also want to debate on art.**
- **Users who want to show their art publicly.**
- **Users who want to get information about different art exhibitions and participate in it.**

This document is a prototype of an art gallery app project. This document describes the different services provided by the art gallery app and how users can use the art gallery efficiently.

3. Product Scope

The art gallery is basically updating the manual Art gallery into an internet-based application so that the users can upload their art works . It also provides details about different types of arts, artists, exhibition information and gives options to customers or users if they want to buy or rent the art. If two or more users or customers want to buy any art we will conduct a bidding for them.

4. Description

Art gallery apps enable us to discover and buy various types of art from anywhere at any time because of the internet.
Art galleries contain huge collections of various kinds of art pieces made by seasoned and world-famous artists. Online art galleries are good places to discover art pieces of various types, belonging to different genres and made by both famous artists.Especially for young talent, online art galleries are helpful. They can register themselves on the site and make their works available to the public. Users need to provide Name , Phone number

, Address , email ,Gender , Age during registration also user set the password. If a user registers as an artist he can upload his artwork on the app.We generate distinct folders for picture, video, and audio data on the server side.These directories on the server are where the client-uploaded files are kept. We need file details like the file location, file name, and file type whenever the client requests a certain file the server can provide it efficiently. Artists can make his artwork available for buyers. Buyer can buy or take artwork as rent which is available and also he/she has enough app wallet. Before making a purchase, a user must first register in the app by entering information such as their name, email address, phone number, and so forth. Additionally, the user account has a password. The user must then pick a product and a method of payment. Based on the chosen payment method, the user completes the transaction. The security of online art gallery apps is quite effective. They give accurate information about transactions, such as the price, time, and date as well as details on the works of art and the artists who are the owners of those works.Bidding can take place on an artwork if numerous users wish to purchase it. Following that, the shipment is completed using the user and artist-provided information. (like: address , email address, phone number). But the viewer type of user cannot buy or rent artwork also cannot upload any artwork. Users can write feedback to any artwork and also an online art gallery app's comment section allows users to express their thoughts and read those of other users. Chat with other users in the comment section on subjects related to art.

- **Art gallery apps enable us to discover and buy various types of art from anywhere at any time because of the internet.**
- **Art galleries contain huge collections of various kinds of art pieces made by seasoned and world-famous artists. Online**

art galleries are good places to discover art pieces of various types, belonging to different genres and made by both famous artists.Especially for young talent, online art galleries are helpful.

- **They can register themselves on the site and make their works available to the public. Users need to provide a Name(first name , middle name ,last name etc.),User_id/User name , Phone number , Address(street ,street name, street number ,apt name , city, state, zip.) , email,qualification in art ,Gender , date of birth(DD/MM/YY), Age during registration also user set the password. By default the app wallet is set to zero value.**

- **If a user registers as an artist he can upload his artwork on the app.Through an online art gallery, User can effectively display his/her work and connect with a sizable audience. Users can list the title and the year an artwork was created. They can also decide on an art purchase price and the daily cost of an exhibition.**

- **Also there are different sections for different types of drawing like, Caricature drawing , Cartoon drawing, Figure drawing, Gesture drawing, Perspective drawing, Line drawing, Scratchboard drawing etc.**

- **Also there are different sections for different types of music like, pop music , rock music , classical , Disco, Opera etc.**

- **Also there are different sections for different types of Dance like Ballet , Ballroom , Contemporary , Hip Hop , Jazz , Tap Dance , Folk Dance.**

- **Distinct folders are made for picture, video, and audio data on the server side.These directories on the server are where the client-uploaded files are kept. We need file details like the file location, file name, and file type whenever the client requests a certain file the server can provide it efficiently. Also the server provides a number/Art id to all uploaded art to identify each art piece uniquely.**

- **Artists can make his artwork available for buyers. Buyer can buy or take artwork as rent which is available and also he/she has enough app wallet. Before making a purchase, a user must first register in the app by entering information such as their name, email address, phone number, and so forth. Additionally, the user account has a password.**

- **Different types of payment methods are available for users like cash on delivery, credit card, bank transfer, direct debit, mobile payment ,QR payment ,UPI etc. The user must then pick a product and a method of payment. Based on the chosen payment method, the user completes the transaction.**

- **The security of online art gallery apps is quite effective. They give accurate information about transactions, such as the price, time(hh:mm:ss),transaction/rent number,shipment date(DD/MM/YY),shipment time(hh:mm:ss) ,shipment from , shipment to,payment method and date as well as details on the works of art and the artists who are the owners of those works.**

- **Bidding can take place on an artwork if numerous users wish to purchase it. Employees of the gallery give the user information regarding the bidding, such as the bidding number/bidding id, user name, start price, and end price, when the auction has concluded. Additionally, they give information on the transition after it occurs. The viewer type of user cannot buy or rent artwork also cannot upload any artwork.**

- **Employees of the app give users a date on which their product will arrive at their residence. Using the information they provided upon registration, the item is then delivered to their home.**

- **Users can write feedback to any artwork and also an online art gallery app's comment description section allows users to express their thoughts and read those of other users. Chat with other users in the comment section on subjects related to art. Users also can give ratings to artists in the form of rating numbers in range 1 to 10 about artist performance.**

- **Users can also get in touch with artists using the app's provided phone number and email address.**

- **Artwork is categorized by artist or art form in an art gallery app. Additionally, users can utilize an app's search feature to locate specific works of art by entering the title of the piece and the artist.**

- **Artists are welcome to offer their works that are available for exhibition as well as their contact information, which**

includes his/her address and phone number. Therefore, they transfer art from his or her home to an exhibition place. Employees provide exhibition details like exhibition place , exhibition number/exhibition id, start date(DD/MM/YY) , end date(DD/MM/YY) etc.

- **Apps present their information through an art gallery's "about us" section. With this knowledge, a user might get in touch with an app employee.**

- **Artists can give lectures or talk related to their work, process, and influences. The lecture topic, location, start time(hh:mm:ss), conclusion time(hh:mm:ss), and other details must be disclosed by the artists and information about artist talks offered to users.**

Schemas:-

1.Artist

Attributes: Artist ID, Field ,User ID,Qualification
Primary Key:Artist ID
Relation: User ID from User table (UserId)

2.Artwork

Attributes: Art ID, Art type , Artist ID , Exhibition ID,Year, Title, Price,Available_for_sell,Available_for_exhibition, exhibition_cost.
Primary Key:Art ID
Relation: Artist ID From table Artist(Artist ID)

Exhibition ID From table Exhibition(Exhibition ID)

3. Singing_Art

Attributes:Singing_Art_ID, Art ID, Music_type,File name , File type,Path.
Primary Key:Singing_Art_ID
Relation: Art ID From table Artwork(Art_ID)

4.Drawing_Art

Attributes:Drawing_Art_ID,Art ID, Drawing_type,File name , File type,Path.
Primary Key:Drawing_Art_ID
Relation: Art ID From table Artwork(Art_ID)

5.Dancing_Art

Attributes:Dancing_Art_ID, Art ID,Dance_type,File name , File type,Path.
Primary Key:Dancing_Art_ID
Relation: Art ID From table Artwork(Art_ID)

6.Order_to_Buy

Attributes: OrderID ,Date, Art_ID, User_ID,time,shipment_date , shipment_time,shipment_from ,shipment_to ,payment_method.
Primary Key:OrderID
Relation: Art ID From table Artwork(Art_ID),
User_ID From table User(User_ID)

7.Rental

Attributes: RentalID,Rental Start date, Rental End Date,Art ID,User_ID,time,shipment_date , shipment_time,shipment_from ,shipment_to ,payment_method.
Primary Key:RentalID
Relation: Art ID From table Artwork(Art_ID)
, User_ID From table User(User_ID)

8. User

Attributes: UserId , Name , Phone number , Address , email ,Gender , Age, Password, app wallet.
Primary Key:UserId

9. Bidding

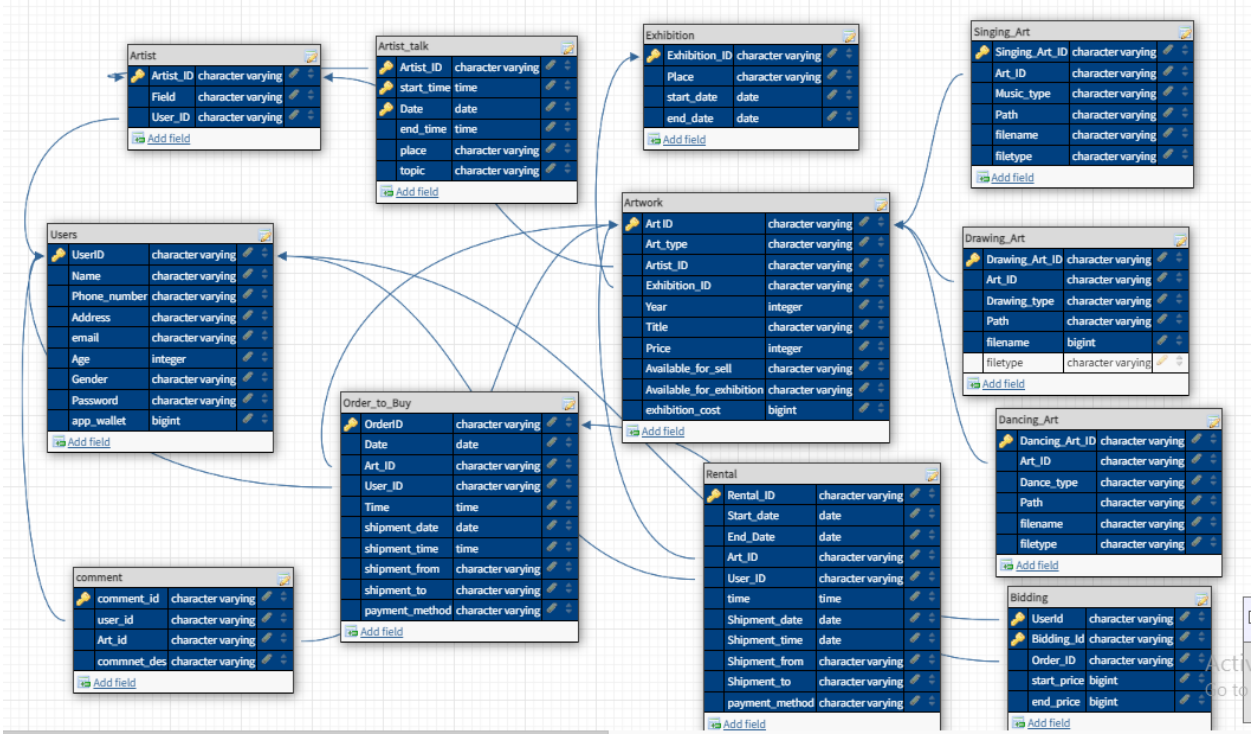
Attributes : Bidding_Number, User_ID,Bidding_Id,Order ID,Start price,End Price.
Primary Key:Bidding_Number
Relation: OrderID From table Order_to_Buy(OrderID)
User_ID Form User(UserId)

10. Exhibition

Attributes :Exhibition ID, Place , start date , End date
Primary Key:Exhibition ID

11. Comment
Attributes : comment_id , user_id , art_id , comment_des
Primary key: comment_id
Relation: user_id Form User(UserId)
Art_id from table Artwork(Art_ID)

12. Artist_talk
Attributes : Artist_id , start_time, Date , end_time, place, topic.
Primary key: Artist_id , start_time, Date
Relation: Artist_iD From table Artist(Artist ID)



2. Document the Requirements Collection/ Fact Finding Phase

1)Background Reading/s

We have read some books online and accessed online materials regarding art galleries to get information about the art gallery system.

1. Description of each reading done

- **Some of the books we have gone through and learned that:**
  - **Book Arts: Beautiful Bindings for Handmade Books**

- **From this book we have found that how can we make section with the different types of arts**

❖ **Form some of the websites we have learned that ,**
We can make different types of sections like:

- **Customer Login**
- **Admin login**
- **Art in sale**
- **New art**

We have found that we can give access to our art gallery for different users like admin , customer(buyer) and viewer. We can make sections like lists of art for sale. Add new art in the system etc.

❖ **From some of the Videos we have learned that,**
We can also design virtual reality art galleries by making software for that and give access to viewers of the art gallery App.

❖ **From that article we learn about how video , image and audio files are stored using databases. At the server side we create different folders for images , videos and audio files. The files uploaded by the client are stored at the server side in those folders. When a client requests a particular file we require file information like file path, file name and file type. So we need to store file path , name and type in the database for all files.**

2. References

Some of the books we have gone through to get reference:

- **Book Arts: Beautiful Bindings for Handmade Books**

Some of the websites we have visited to get reference:

https://www.lovelycoding.org/art-gallery-management-system/

https://www.sourcecodester.com/php/14503/art-gallery-management-system-using-phpmysql-source-code.html

https://www.geeksforgeeks.org/how-to-upload-image-into-database-and-display-it-using-php/

Virtual art gallery video:
https://youtu.be/3ewO-gy1VKo

3. List the combined Requirements gathered from Background Readings.

- **Categorizing arts with its type**
- **Types of users we have to create for our app.**
- **Add another software like a virtual art gallery inbuilt in our app.**
- **File path, name, and type for video, image, and audio files must be stored in a database.**

2)Interview

Owner:

System : Art Gallery Application
Participants: Subhasis Roy(Manager)
Date: 24/09/2022
Time : 5:30 PM
Duration : 45 minutes
Place: Archer Art Gallery,Navrangpura,Ahmedabad

Purpose of Interview : Preliminary meeting to identify problems and requirements regarding Art Gallery System.

1. **Which types of services do you provide in the art gallery?**
   Users can upload image , audio and video files and show their art work publically. We also provide exhibition's details, users can also buy and take an art piece as a rent. We also provide artist details to users.

2. **Which types of users visit the art gallery?**
   Buyer,Artist and Viewer.

3. **How do you manage many Users for one art piece?**
   If Two or more Users want to buy an art then we will conduct bidding for them. User with the maximum bid will take that art for that bid price.

4. **How do you categorize arts?**
   By Art types

5. **Which kind of art do you collect?**
   Arts like Drawing,Singing,dance and Sculpture.

6. **How do you manage transactions?**
   We keep track of users' email, phone number and address and also information of art pieces(like : art owner , price, availability). Using that information we communicate with users.

7. **How do you manage exhibitions?**

We keep track of exhibitions date, place, included arts and we provide this information to interested users.

8. **On an average how many users use your application at a time?**
   Approximately 10000

9. **How can you communicate with users?**
   We will take data from the user like Name, Phone no., address, email, gender.

10. **How many Users register in your application?**
    Approximately 20 lakhs

11. **Is there any option for users to return already bought art and return money to the user?**
    No.

12. **Do you have any plan for modification of the art gallery system in future?**
    Not decided yet.

Summary:

From the above interview, we have got an idea about how we can categorize arts with its types and what different kinds of users will be there in the art gallery app system like viewer, artist, buyer etc. How they communicate with buyer and seller for transections. How they invite users to the exhibitions.

Buyer:

System : Art Gallery Application
Participants: Jai shahrukh
Date: 24/09/2022
Time : 6:30 PM
Duration : 45 minutes
Place: Online mode
Purpose of Interview : Preliminary meeting to identify problems and requirements regarding Art Gallery System.

Question:

1. **What do you prefer: an online art gallery app or offline art gallery and Why?**
   I prefer online art gallery apps because online art gallery systems have many benefits over offline galleries. We can effectively use online galleries and also we can use them at any place at any time. A large amount of stuff is also present in an online gallery. We can easily access information about art ,artists and exhibitions. Also we can read people's thoughts using comment sections.

2. **Tell your thoughts about security.**
   Security is very efficient in online art gallery apps. They provide us proper information about transactions like price , time, date , information of art pieces , information of artists who own that art.

3. **How can you communicate with artists?**
   Apps contain information about artists like, name ,phone number , email address , address etc. Using that information I communicate with artists.

4. **Which additional services do you need from the online art gallery app?**
   Restoration of a Damaged Artwork.

5. **How can you communicate with other users?**
   I share my ideas and read those of other users in the comment section of an online art gallery app. Use the comment box to chat with other users about topics pertaining to art.

6. **How can you communicate with an art gallery app employee?**
   They give their details via the app for an art gallery's about us section. With the help of such details, I got in touch with an app worker.

7. **How can you do transection?**
   To purchase a product first I register in the app and provide details about my email id, name ,address , phone number etc. Also my account is password protected. Then I need to select a product and choose a payment option. I complete the transaction based on the selected payment method.
8. **How do art pieces deliver to your home?**
   They provide me a date on which product will arrive at my home. The item is then delivered to my house using the details I provide on registration.

Summary:
From the above interview, We have got an idea about how communication happens between user - artist , user - user and user - art gallery app employee. How they manage transaction

efficiently and with proper security and what kind of information needed for that. How shipment of artwork happens.

Viewer:

System : Art Gallery Application
Participants: kunal Mahinder
Date: 24/09/2022
Time : 7:30 PM
Duration : 45 minutes
Place: Online mode
Purpose of Interview : Preliminary meeting to identify problems and requirements regarding Art Gallery System.

1. **What do you prefer: an online art gallery app or offline art gallery and Why?**
   I prefer physical art galleries. The online art gallery systems include a lot of content, it is difficult to view it properly in an online mode.

2. **How can you communicate with other users?**
   In the comments part of an online art gallery app, I post my ideas and read those of other users. Talk with other users about topics relating to art by using the comment section.

3. **How is the artwork arranged in the art gallery app ?**
   Artwork is grouped both by the artist and by the category of art.

4. **How do you find particular art or a particular artist?**
   Using the search bar of an app.

5. **How can you tell your thoughts about particular art?**

Using the comment section of the art gallery app. Al so I can talk to an artist via email address.

6. How much time do you spend in one months in an art gallery app?

Approximately 30 hours.

Summary:
From the above interview, We have got an idea about how communication happens between user - artist , user - user. How are the pieces of art shown in the art gallery app. We also get a sense of how an app's searching feature works. Additionally, we can determine how long users typically spend using an app.

Artist :

System : Art Gallery Application
Participants: Suman rohan
Date: 25/09/2022
Time : 7:30 PM
Duration : 45 minutes
Place: Online mode
Purpose of Interview : Preliminary meeting to identify problems and requirements regarding Art Gallery System.

1. **What do you prefer: an online art gallery app or offline art gallery and Why?**
   I prefer online art gallery apps because I can effectively display my artwork and reach a sizable audience through an online art gallery. I can use it anytime and anywhere I want.

2. **What is your field of interest?**

Painting.

3. **How can you show your artwork to the public using an art gallery app?**
   I register in the app and provide details about my email id, name ,address , phone number , art piece etc. Then I upload my art piece demo file on the art gallery app and make it available to the public.

4. **Do you provide your art for exhibition? If yes, how?**
   Yes. I first make my artwork available for exhibition and give out my contact information, including my address and phone number. Then they take artwork from my house .

5. **How can you communicate with users?**
   I share my thoughts about my artwork in the comments section of an online art gallery app and read responses of other users.

6. **How can you communicate with an art gallery app employee?**
   They provide their information via the app for the "about us" part of an art gallery. Such information allowed me to contact an app employee.

7. **How does shipment happen of art pieces?**
   I provide my contact information, which includes my address,name,email and mobile number. Then they take artwork from my home and pay the price of the artwork. They give information about the person who purchases or rents my artwork.

8. **Tell your thoughts about security.**
   The security of online art gallery apps is quite effective. They give us accurate information about transactions, such as the price, time, and date as well as details of the person who purchases or rents my artwork.

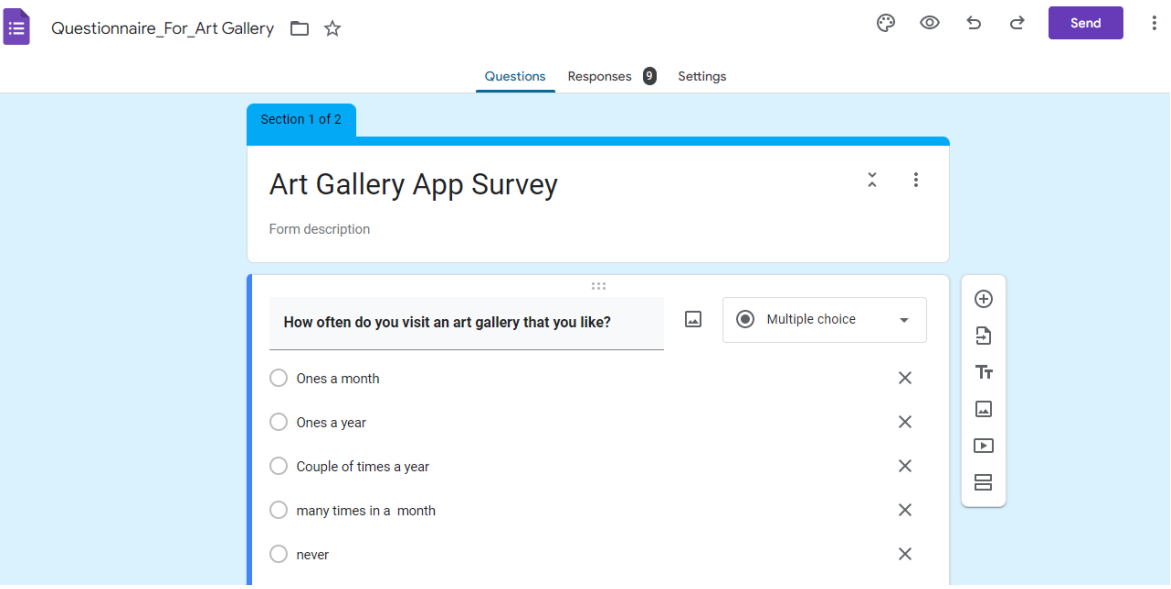
Summary:
From the above interview, We have got an idea about how communication happens between artist-users ,artist - art gallery app employees. How can artists use an art gallery app to display their works to the public. We got an insight on how they manage exhibitions.

List the combined Requirements gathered from all above interviews.

- **Store user name, email address , address and phone number.**
- **Need to store exhibition details.**
- **Need to store transaction information.**
- **Need to store shipment information.**
- **Categories art by its types.**
- **Keeping comments on art is necessary.**

3)Questionaier:

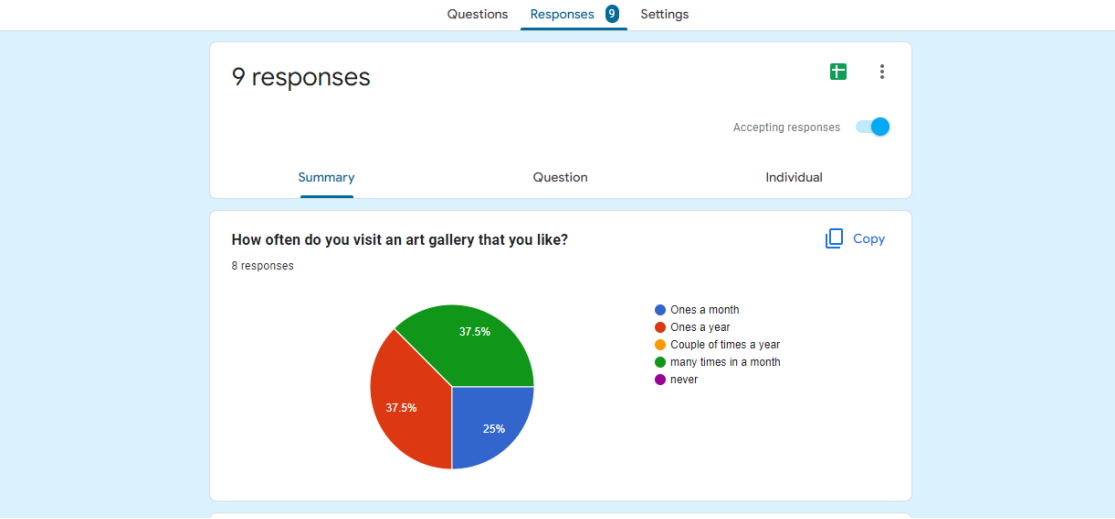
User Form:

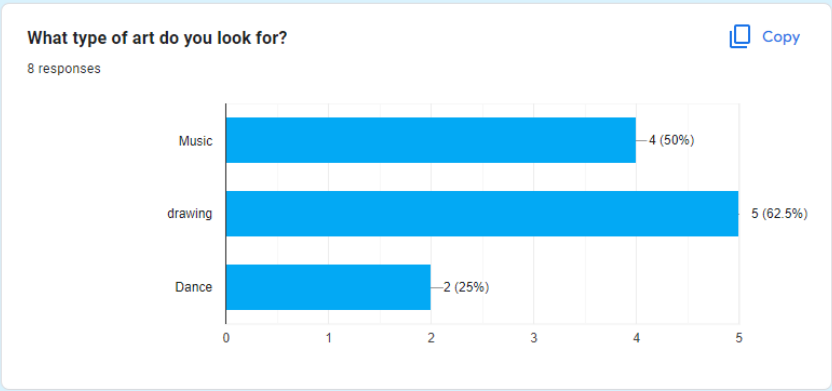


Form Link:

https://docs.google.com/forms/d/e/1FAIpQLSfNiOeW2vawyHV2gfUrDKd8wz-AWx3Y-fw3gLwVZuh3L39UKQ/viewform?usp=sf_link

Responses:





Answers Received from google form:
https://docs.google.com/spreadsheets/d/1oR_NrdIlcH3WYGK1sJ5DKQ78z9PhYnjX9wq6McbE4f0/edit?resourcekey#gid=1186199577

4)Observation:

System : Art Gallery Application
Observation By: Harshal Thakor and Vedant Patel
Date: 25/09/2022
Time : 5:30 PM
Duration : 1 hour
Place: Archer Art Gallery,Navrangpura,Ahmedabad

Observations:

- **We have seen there The art gallery has been divided into sections by the art types like the drawing ,sculpture , painting etc. has different sections.**
- **For each art there has been a glass wall in which art has been stored.**
- **Below each of the arts, there has been the artist name and creation date of the art.**

- ➢ **If someone wants to buy the art there has been another section to buy art if the artist wants to buy it.They will share art with price decided by the artist which can be buyable.**
- ➢ **If two or more people want to buy an art at a time they will conduct an auction for them. Whoever has the highest bid will take the art.**
- ➢ **Very little security at art galleries, both in building and on premises.**

Summary:

From the above observation, we have got an idea about how we can categorize arts with its types and what different kinds of users will be there in the art gallery app system like viewer, artist, buyer etc. What kind of information is required for an art piece.

3.Create Fact Finding Chart

Objective	Technique	Subject	Time commitment
To get background on the Art gallery system	Background reading	Art gallery's arts registered data	1 hour
To get understanding About the roles of each department in art gallery	Interview	Art gallery manager	45 minutes
To get understanding	interview	Art gallery manager	45 minutes

about the interaction and dealing with the artist to show art in the art gallery			
To get idea about how the art galley sections organized	Observation	2 people	0.5 day
To get survey from the users of art gallery	Questionnaire	Few persons to fill form for survey	1 day

4. List Requirements

List of Common requirements in art gallery app:

- We will need to maintain the Art and the details of its artist and date of art.
- We have to divide the art gallery in sections by the art types.Where viewers can view art by its types.
- If Someone wants to buy art we have to manage the transaction details.
- If two or more users want to buy art at the same time we have to conduct bidding for them.
- If an Artist gives permission to present(Exhibition) art offline we have to manage the place and start date ,end date etc for the exhibition.

5. User Categories and Privileges

In our Art Gallery App system three types of users will be there:

- **Artist - can upload art,artist also can be an art buyer so he can also access apps as buyer of art.**
- **Buyer - can buy art online**
- **Viewer- can just view art online and if wants visit art exhibition offline**

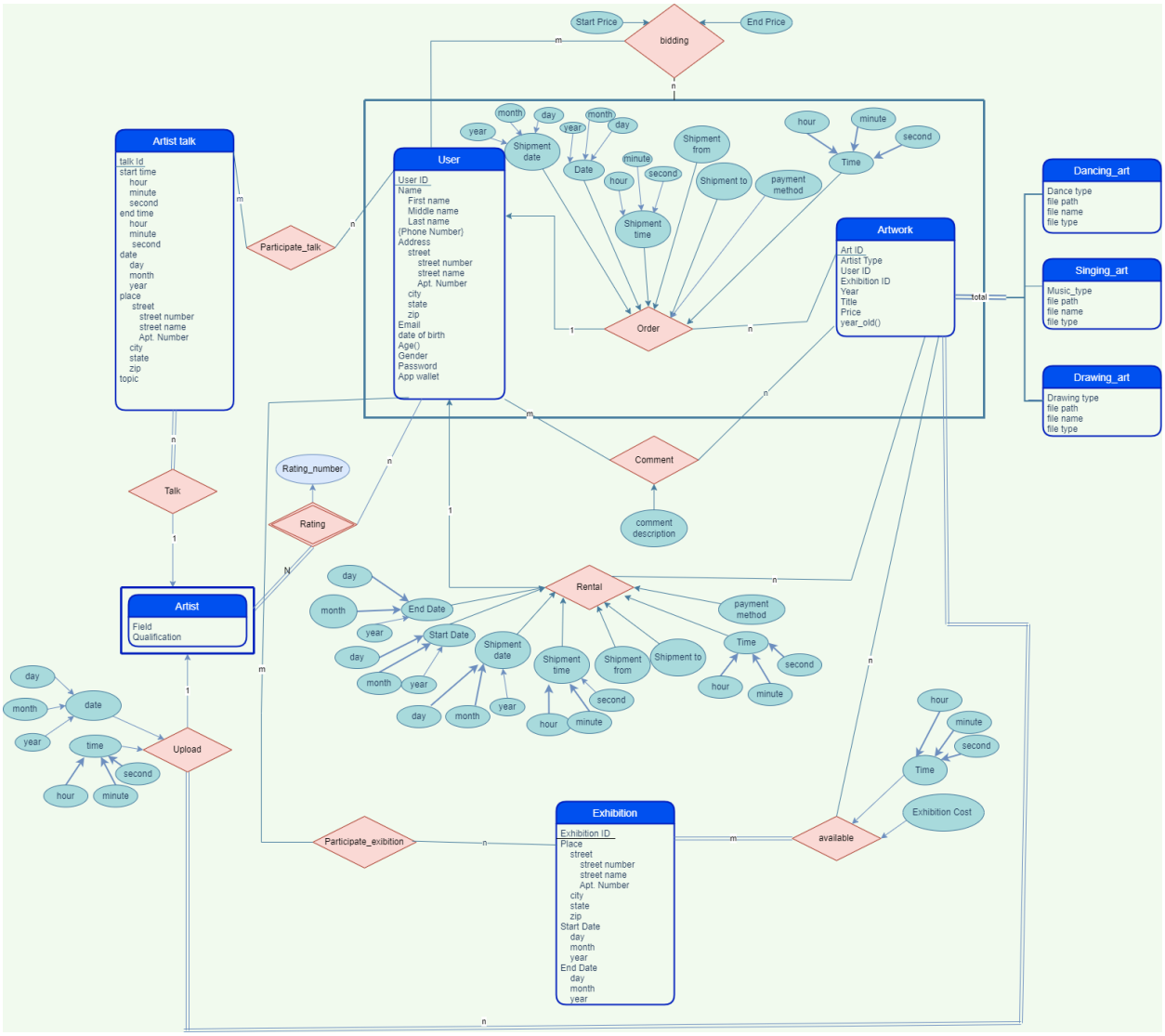
6. Assumptions

We have assumed that bidding will only happen if two or more users want to buy the same art.
We have assumed that the Art gallery App server remains online all time.

7. Business Constraints

Buyer cannot access the art gallery app as artist section.If buyer wants to upload art as artist he has to go to artist section.
Viewers also cannot upload art pieces and also cannot buy or take art as rent.

- **Final ER Diagram and Noun analysis**
- **Final ER diagram**



Problem Description

- **Art gallery apps enable us to discover and buy various types of art from anywhere at any time because of the internet.**

- **Art galleries contain huge collections of various kinds of art pieces made by seasoned and world-famous artists. Online**

art galleries are good places to discover art pieces of various types, belonging to different genres and made by both famous artists.Especially for young talent, online art galleries are helpful.

- **They can register themselves on the site and make their works available to the public. Users need to provide a Name(first name , middle name ,last name etc.),User_id/User name , Phone number , Address(street ,street name, street number ,apt name , city, state, zip.) , email,qualification in art ,Gender , date of birth(DD/MM/YY), Age during registration also user set the password. By default the app wallet is set to zero value.**

- **If a user registers as an artist he can upload his artwork on the app.Through an online art gallery, User can effectively display his/her work and connect with a sizable audience. Users can list the title and the year an artwork was created. They can also decide on an art purchase price and the daily cost of an exhibition.**

- **Also there are different sections for different types of drawing like, Caricature drawing , Cartoon drawing, Figure drawing, Gesture drawing, Perspective drawing, Line drawing, Scratchboard drawing etc.**

- **Also there are different sections for different types of music like, pop music , rock music , classical , Disco, Opera etc.**

- **Also there are different sections for different types of Dance like Ballet , Ballroom , Contemporary , Hip Hop , Jazz , Tap Dance , Folk Dance.**

- **Distinct folders are made for picture, video, and audio data on the server side.These directories on the server are where the client-uploaded files are kept. We need file details like the file location, file name, and file type whenever the client requests a certain file the server can provide it efficiently. Also the server provides a number/Art id to all uploaded art to identify each art piece uniquely.**

- **Artists can make his artwork available for buyers. Buyer can buy or take artwork as rent which is available and also he/she has enough app wallet. Before making a purchase, a user must first register in the app by entering information such as their name, email address, phone number, and so forth. Additionally, the user account has a password.**

- **Different types of payment methods are available for users like cash on delivery, credit card, bank transfer, direct debit, mobile payment ,QR payment ,UPI etc. The user must then pick a product and a method of payment. Based on the chosen payment method, the user completes the transaction.**

- **The security of online art gallery apps is quite effective. They give accurate information about transactions, such as the price, time(hh:mm:ss),transaction/rent number,shipment date(DD/MM/YY),shipment time(hh:mm:ss) ,shipment from , shipment to,payment method and date as well as details on the works of art and the artists who are the owners of those works.**

- **Bidding can take place on an artwork if numerous users wish to purchase it. Employees of the gallery give the user information regarding the bidding, such as the bidding number/bidding id, user name, start price, and end price, when the auction has concluded. Additionally, they give information on the transition after it occurs. The viewer type of user cannot buy or rent artwork also cannot upload any artwork.**

- **Employees of the app give users a date on which their product will arrive at their residence. Using the information they provided upon registration, the item is then delivered to their home.**

- **Users can write feedback to any artwork and also an online art gallery app's comment description section allows users to express their thoughts and read those of other users. Chat with other users in the comment section on subjects related to art. Users also can give ratings to artists in the form of rating numbers in range 1 to 10 about artist performance.**

- **Users can also get in touch with artists using the app's provided phone number and email address.**

- **Artwork is categorized by artist or art form in an art gallery app. Additionally, users can utilize an app's search feature to locate specific works of art by entering the title of the piece and the artist.**

- **Artists are welcome to offer their works that are available for exhibition as well as their contact information, which**

includes his/her address and phone number. Therefore, they transfer art from his or her home to an exhibition place. Employees provide exhibition details like exhibition place , exhibition number/exhibition id, start date(DD/MM/YY) , end date(DD/MM/YY) etc.

- **Apps present their information through an art gallery's "about us" section. With this knowledge, a user might get in touch with an app employee.**

- **Artists can give lectures or talk related to their work, process, and influences. The lecture topic, location, start time(hh:mm:ss), conclusion time(hh:mm:ss), and other details must be disclosed by the artists and information about artist talks offered to users.**

1. Noun (& Verb) Analysis

Nouns	Verbs
User	register
Name	enroll
Phone Number	upload
Address	request
Email	provide
Gender	create/make
Age	upload
Password	Enter
Artist	buy
server	rent
Artwork	upload
File name	purchase
File Type	write
File Path	read
File Location	chat
ArtWork	upload
Artist	Enter
Buyer	buy
picture	communicate
video	express

Audio	rent
directory	buy
Artist	upload
files	touch
Buyer	categorize
user	utilize
uploader	Register
payment method	speak
Transaction	comment
Security	search
price	participate
person	available
time	Talk
date	rating
details	
shipment	
Bidding	
Email address	
ArtType	
Artist	
viewer	

user	
Artist	
title	
Art	
Exhibition	
Contact	
address	
Phone number	
Payment Method	
Debit card	
Credit card	
UPI	
Cash	
Net Banking	
app wallet	
UserId	
Exhibition ID	
Art ID	
Bidding_Id	
Order Id	
Transaction ID	

Artist_talk	
Field	
qualification	
year	
Exhibition cost	
Start Price	
End Price	
place	
Topic	
comment description	
Start time	
End time	
topic	
Shipment date	
Shipment time	
Shipment from	
Shipment to	
Dance type	
Music Type	
Drawing Type	

Singing Art	
Dancing Art	
Drawing Art	
rental	
Order	
Start date	
End date	
Caricature drawing	
Cartoon drawing	
Figure drawing	
Gesture drawing	
Perspective drawing	
Line drawing	
Scratchboard drawing	
pop music	
rock music	
classical	
Disco	
Opera	

Ballet	
Ballroom	
Contemporary	
Hip Hop	
Jazz	
Tap Dance	
Folk Dance	
street	
street name	
street number	
apt name	
city	
state	
zip	
Day	
Month	
Year	
Hour	
Minute	
Second	

Date of Birth	
rating number	

2.Accepted Noun & Verbs list

Candidate entity set	Candidate attribute set	Candidate relationship set
User	User ID , Name , Phone number , Address , email ,Gender , Age, Password, app wallet.	order ,Rent ,comment,bidding, participate,rating
Artist	User ID, Field ,qualification	upload order,Rent,comment,bidding,talk,rating
Artwork	Art ID, Art type ,Exhibition ID,Year, Title, Price.	Order,rent, bidding, Comment,available for exhibition,upload
Exhibition	Exhibition ID, Place , start date , End date	Participate, available

Artist Talk	**Talk id , Start time, Date , end time, place, topic**	**Participate,talk**
Drawing_Art	**art_id,drawing type, file path ,file name file path**	
Singing_Art	**art_id,Music type,file path ,file name file path**	
Dancing_Art	**art_id,dancing type, file path ,file name file path**	

3.Rejected Noun & Verbs list

Noun	Reject Reason
server	Irrelevant
person	Vague
Debit card	Irrelevant
Credit card	Irrelevant

UPI	**Irrelevant**
Cash	**Irrelevant**
Net Banking	**Irrelevant**
Express	**Vague**
touch	**vague**
search	**Irrelevant**
utilize	**Vague**
Chat	**Irrelevant**
details	**general**
File location	**Duplicate**
Art	**Duplicate**
Buyer	**Duplicate**
picture	**Duplicate**
directory	**Duplicate**
uploader	**Duplicate**
register	**Irrelevant**
transaction	**Duplicate**
security	**Vague**
Details	**general**
viewer	**Duplicate**
Contact	**Duplicate**

Transaction ID	Duplicate
User ID	Attribute
Name	Attribute
Phone number	Attribute
Address	Attribute
email	Attribute
Gender	Attribute
Age	Attribute
Password	Attribute
app wallet	Attribute
Field	Attribute
qualification	Attribute
Art ID	Attribute
Art type	Attribute
Exhibition ID	Attribute
Year	Attribute
Title	Attribute
Price	Attribute
Exhibition cost	Attribute
Place	Attribute

start date	Attribute
End date	Attribute
Start time	Attribute
Talk id	Attribute
end time	Attribute
Date	Attribute
place	Attribute
topic	Attribute
drawing type	Attribute
file path	Attribute
file name	Attribute
file path	Attribute
Dancing type	Attribute
Music type	Attribute
register	Irrelevant
enroll	Irrelevant
request	Irrelevant
provide	Irrelevant
create/make	Vague
enter	Vague

buy	duplicate
purchase	duplicate
write	Vague
read	Vague
categorize	Vague
speak	duplicate
order	Association
Rent	Association
comment	Association
bidding	Association
participate	Association
talk	Association
upload	Association
available for exhibition	Association
available	Association
street	Attribute
street name	Attribute
street number	Attribute
apt name	Attribute
city	Attribute
state	Attribute

zip	**Attribute**
Day	**Attribute**
Month	**Attribute**
Year	**Attribute**
Hour	**Attribute**
Minute	**Attribute**
Second	**Attribute**
Date of Birth	**Attribute**
rating numbers	**Attribute**

- **ER Model:**

  https://app.diagrams.net/#G1HQ-m7c7CI8uTPA8XrITvwNrfG0dxP4Z0
  https://drive.google.com/file/d/1HQ-m7c7CI8uTPA8XrITvwNrfG0dxP4Z0/view?usp=sharing

1) Identify Entity types.

Entity sets	Type of entity	Type of relationship/s

User	strong	Simple Association Link, Aggregation,Hierarchy
Artwork	strong	Simple Association Link, Aggregation, Hierarchy
Artist	weak	Simple Association Link
Exhibition	strong	Simple Association Link
Artist Talk	strong	Simple Association Link
Drawing_Art	strong	Hierarchy
Singing_Art	strong	Hierarchy
Dance_Art	strong	Hierarchy

2) Identify Relationship types.

Relationship	**Degree of a Relationship Set**	**Mapping Cardinality**	**Associate Entity Set**	**Associate Attributes**
Order	Binary	one-many	User-Artwork	Date,Shipment date,Shipment timeShipment from,Shipment to,payment method , time
Comment	Binary	Many-many	User-Artwork	Comment _description

Rental	Binary	one-many	User-Artwork	Start date , end date, Shipment date,Shipment timeShipment from,Shipment to,payment method , time
participate_talk	Binary	many-many	User - Artist_talk	
available	Binary	many-many	Artwork-Exhibition	Time , Exhibition cost
talk	Binary	many-one	Artist_talk-Artist	
upload	Binary	one-many	Artist-Artwork	time,date
participate_exibition	Binary	many-many	Exhibition-User	

Aggregation Relationship:
bidding

- User(entity set) - Order (Relation ship)
- Associate attributes: Start Price , End Price.

Entity set who have Total participation:

- Entity Artist has total participation in relationship rating.
- Entity Artwork has total participation in the relationship upload.
- Entity Artist_talk has total participation in the relationship talk.
- Entity Exhibition has total participation in the relationship available.

Specialization :

drawing_art , dancing_art and singing_art are inherited from the artwork entity.

Identifying Relationship:

Relationship rating is Identifying relationship for entity artist. And the discriminator is user_id which uniquely identifies the entity artist .

3)Analyze ERD for any other missing information.

See SRS edited document.

- **Final Normalization and DDL Scripts:**

- **Relations with Schema Before Normalization**

1. **User**

   Attributs: <u>UserID</u>, Name ,Address, Email, date_of_birth , age , Gender , Password , App_wallet.

   Primary key: UserID

   Functional dependency :

   UserID -> { Name ,Address, Email, date_of_birth , age , Gender , Password , App_wallet }

date_of_birth -> age

Redundancies: age
Because age can be found from date_of_birth.

2. Artwork

Attributs: <u>ArtID</u> , Art type , ExhibitionID , Year , Title , Price, year_old , user_ID,Upload_date , Upload_time

Primary key: Art ID

Foreign key: userID,ExhibitionID.
userID reference from table Artist(userID)
ExhibitionID reference from table Exhibition(ExhibitionID)

Artwork and Artist entities have many to one relationship upload. So Upload relationship attributes to Artwork table and foreign key userID.

Functional dependency :
ArtID -> { Art type , ExhibitionID , Year , Title , Price, years_old , user_ID,Upload_date , Upload_time }
Year -> years_old
Redundancies: years_old
Because years_old can be found from the year it has been made.

3. Dancing Art

Attributs:Dance Type, File path, File name,File Type , <u>Art ID</u>
Primary key:Art ID

Foreign key:Art ID
ArtID reference from table Artwork (ArtID)
DancingArt is a specialization of Artwork

Functional dependency :
ArtID -> {Dance Type, File path, File name,File Type}
File path -> file name
Redundancy: File name
Some ArtID have same file name

4.Singing Art

Attributs:singing Type, File path, File name,File Type,Art ID
Primary key:Art ID

Foreign key:Art ID
ArtID reference from table Artwork (ArtID)
Singing Art is a specialization of Artwork

Functional dependency :
ArtID-> {singing Type, File path, File name,File Type}
File path -> file name
Redundancy: File name
Some ArtID have same file name

5.Drawing Art

Attributs:Drawing Type, File path, File name,File Type,Art ID
Primary key:Art ID

Foreign key:Art ID
ArtID reference from table Artwork (ArtID)
Drawing Art is a specialization of Artwork

Functional dependency :
ArtID -> {Drawing Type, File path, File name,File Type}
File path -> file name
Redundancy: File name
Some ArtID have same file name

6. Exhibition

Attributs: ExhibitionID , place , Start Date , End Date
Primary key: ExhibitionID

Functional dependency :
ExhibitionID -> { place , Start Date , End Date}

7. Artist_talk

Attributs: talkID , start time , end time , date , place , topic , UserID
Primary key: talkID
Foreign key: UserID
UserID reference from table Artist(userID)

Artist_talk and Artist have many to one relationship talks. So talk relationship attributes added to Artist_talk table and foreign key userID.

Functional dependency :
talkID -> { start time , end time , date , place , topic , UserID}

8. Artist

Attributs: Field , Qualification, User ID
Primary key: UserID
Foreign key: UserID
UserID reference from table User(UserID)

Functional dependency :
UserID -> {Field , Qualification}

9. Comment

Attributs: comment description , UserID , ArtID
Primary key: UserID , ArtID

Foreign key: UserID , ArtID
UserID reference from table User(UserID)
ArtID reference from table Artwork (ArtID)

Comment is a many to many relationship between entity User and Artwork

Functional dependency :
{UserID , ArtID} -> comment description

10. Available

Attributs:Time , Exhibition cost , <u>ArtID , ExhibitionID</u>
Primary key: ArtID , ExhibitionID

Foreign key: ArtID , ExhibitionID
ArtID reference from table Artwork (ArtID)
ExhibitionID reference from table Exhibition(ExhibitionID)

Available is a many to many relationship between entity Exhibition and Artwork

Insert Anomaly : We cannot insert Exhibition cost for Art without Art is in Exhibition
Update Anomaly:If we change Exhibition cost for particular art we need to update Exhibition cost for all entry of particular ArtID
Delete Anomaly: If we delete last entry of particular art we also lost information of Exhibition cost of Art

Functional dependency :
{ArtID , ExhibitionID} -> {Time , Exhibition cost}

11. Participate_exhibition

Attributes: <u>UserID , ExhibitionID</u>
Primary key:UserID , ExhibitionID

Foreign key: UserID , ExhibitionID
ExhibitionID reference from table Exhibition(ExhibitionID)

UserID reference from table User(UserID)
Participate_exhibition is a many to many relationship between entity Exhibition and User

Functional dependency :
{UserID , ExhibitionID} -> {UserID , ExhibitionID}

12. Participate_talk
Attributes: UserID , talkID
Primary key:UserID ,talkID

Foreign key: UserID , talkID
UserID reference from table User(UserID)
talkID reference from table Artist_talk(talkID)
Participate_talk is a many to many relationship between entity Artist_talk and User

Functional dependency :
{UserID,talkID}->UserID , talkID

13. Bidding
Attributes: Start price, End Price, ArtID ,UserID,Order_user_ID
Primary key:ArtID ,UserID,
Foreign key: ArtID ,UserID,Order_user_ID

ArtID reference from table Artwork (ArtID)
UserID reference from table User(UserID)
Order_user_ID reference from table Order(UserID)

Functional Dependencies:
{ArtID,UserID}->Start price, End Price,Order_user_ID
{ArtID} -> {Order_user_ID}

Redundancy : Order_user_id is repeated for ArtID. And we also can find Order_user_id form using ArtID of this relation and ArtID of order relation using join operation.

Insert Anomaly : We cannot insert Start price and End Price for Art without bidding is done on Art
Update Anomaly : If we changeStart price or End Price for particular art we need to update Start price and End Price for all entry of particular ArtID
Delete Anomaly : If we delete the last entry of particular art we also lose information of Start price and End Price for Art.

14. Order

Attributes: Shipment_date , Date , Shipment time , Shipment from , Shipment to , Time , payment method , ArtID , UserID

Primary key: ArtID
Foreign key: ArtID , UserID
ArtID reference from table Artwork (ArtID)
UserID reference from table User(UserID)
Order is a many to one relationship between entity Artwork and User.

Functional Dependencies:
ArtID->{Shipment_date , Date , Shipment time , Shipment from , Shipment to , Time , payment method,UserID}
ArtID- > UserID

15. Rental

Attributes: Shipment date , startdate,enddate ,Shipment time , Shipment from ,Shipment to , Time , payment method , UserID , ArtID

Primary key: ArtID
Foreign key: ArtID , UserID
ArtID reference from table Artwork (ArtID)
UserID reference from table User(UserID)

Rental is a many to one relationship between entity Artwork and User.

Functional dependency :

ArtID-> {Shipment date , startdate,enddate ,Shipment time , Shipment from ,Shipment to , Time , payment method , UserID}
ArtID- > UserID

16 User_phone_no

Attributs: UserID , Phone_no
Primary key: UserID , Phone_no
Foreign key: UserID
UserID reference from table User(UserID)

iv. Normalize the database up to 1NF (scalar values)

1.User

In our database,in table User, User_phone_no is a multivalued attribute.Therefore, we will create a separate table for it.
Its schema has been shown in 16 th table

All other relations are already in 1NF.

v. Normalize the database further to 2NF (Remove Partial Dependencies)

1.User

User Table is not in 1NF so,it is not in 2NF
Except this all the tables in 2NF

All other relations are already in 2NF.

vi. Identify (and document) a List of redundancies existing for the schema in 2NF

1. **Dancing Art**

   File path -> file name
   Redundancy: File name
   File name is repeated for ArtID.

2. **Singing Art**

   File path -> file name
   Redundancy: File name
   File name is repeated for ArtID.

3. **Drawing Art**
   File path -> file name
   Redundancy: File name
   File name is repeated for ArtID.

vii. Normalize it further to 3NF/BCNF (Remove Transitive Dependencies)

1.User

User table is also not in 1NF and 2NF also have transitive relation
UserID -> date_of_brith
Date_of_brith -> age so UserID -> age (transitivity)

In relation User non prime attribute age is transitively dependent on primary key attribute UserID.

2. Artwork

ArtID -> year
Year -> year_old so ArtLD -> year_old (transitivity)

In relation Artwork non prime attribute year_old is transitively dependent on primary key attribute ArtID.

Not in 3NF.

3.Dancing_Art

ArtID -> file_path
file_path -> file_name so ArtID-> file_name (transitivity)

In relation Dancing_Art non prime attribute file_name is transitively dependent on primary key attribute ArtID.
Not in 3NF.

4.Singing_Art

ArtID -> file_path
file_path -> file_name so ArtID-> file_name (transitivity)

In relation singing_Art non prime attribute file_name is transitively dependent on primary key attribute ArtID.
Not in 3NF.

5.Drawing_Art

ArtID -> file_path
file_path -> file_name so ArtID-> file_name (transitivity)

In relation drawing_Art non prime attribute file_name is transitively dependent on primary key attribute ArtID.
Not in 3NF.

All other relations are already in 3NF/BCNF.

After Normalization all tables' schema with DDL:

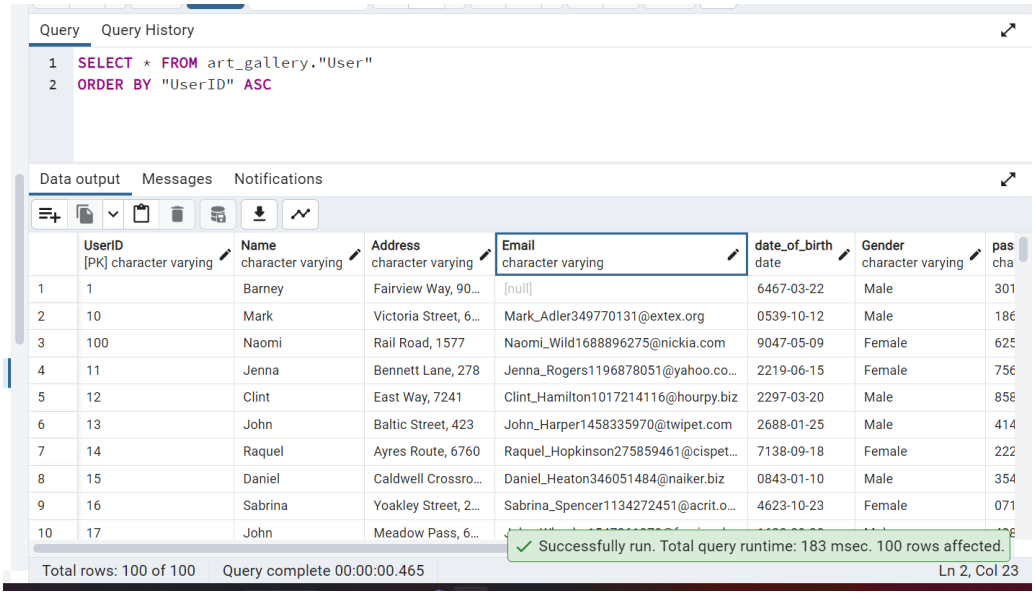
1. **User**

Attributes: <u>UserID</u>, Name ,Address, Email, date_of_birth , Gender , Password , App_wallet.

Primary key: UserID

DDL:

```
-- Table: art_gallery.User

-- DROP TABLE IF EXISTS art_gallery."User";

CREATE TABLE IF NOT EXISTS art_gallery."User"
(
    "UserID" character varying COLLATE pg_catalog."default" NOT
NULL,
    "Name" character varying COLLATE pg_catalog."default",
    "Address" character varying COLLATE pg_catalog."default",
    "Email" character varying COLLATE pg_catalog."default",
    date_of_birth date,
    "Gender" character varying COLLATE pg_catalog."default",
    password character varying COLLATE pg_catalog."default",
    "App_wallet" bigint,
    CONSTRAINT "User_pkey" PRIMARY KEY ("UserID")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."User"
    OWNER to postgres;
```



```
SELECT * FROM art_gallery."User"
ORDER BY "UserID" ASC
```

Query | Query History

Data output | Messages | Notifications

	UserID [PK] character varying	Name character varying	Address character varying	Email character varying	date_of_birth date	Gender character varying	pas cha
1	1	Barney	Fairview Way, 90...	[null]	6467-03-22	Male	301
2	10	Mark	Victoria Street, 6...	Mark_Adler349770131@extex.org	0539-10-12	Male	186
3	100	Naomi	Rail Road, 1577	Naomi_Wild1688896275@nickia.com	9047-05-09	Female	625
4	11	Jenna	Bennett Lane, 278	Jenna_Rogers1196878051@yahoo.co...	2219-06-15	Female	756
5	12	Clint	East Way, 7241	Clint_Hamilton1017214116@hourpy.biz	2297-03-20	Male	858
6	13	John	Baltic Street, 423	John_Harper1458335970@twipet.com	2688-01-25	Male	414
7	14	Raquel	Ayres Route, 6760	Raquel_Hopkinson275859461@cispet...	7138-09-18	Female	222
8	15	Daniel	Caldwell Crossro...	Daniel_Heaton346051484@naiker.biz	0843-01-10	Male	354
9	16	Sabrina	Yoakley Street, 2...	Sabrina_Spencer1134272451@acrit.o...	4623-10-23	Female	071
10	17	John	Meadow Pass, 6...				

Successfully run. Total query runtime: 183 msec. 100 rows affected.

Total rows: 100 of 100 | Query complete 00:00:00.465 | Ln 2, Col 23

2. Artwork

Attributs: ArtID , Art type , ExhibitionID , Year , Title , Price, user_ID,Upload_date , Upload_time

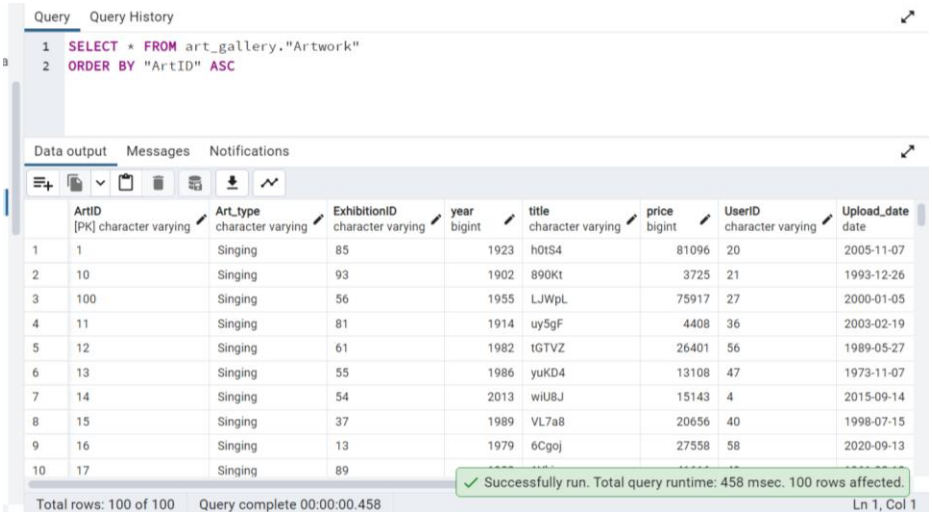
Primary key: Art ID

Foreign key: userID,ExhibitionID.
userID reference from table Artist(userID)
ExhibitionID reference from table Exhibition(ExhibitionID)

DDL:

```
-- Table: art_gallery.Artwork

-- DROP TABLE IF EXISTS art_gallery."Artwork";

CREATE TABLE IF NOT EXISTS art_gallery."Artwork"
(
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
    "Art_type" character varying COLLATE pg_catalog."default",
    "ExhibitionID" character varying COLLATE pg_catalog."default",
    year bigint,
    title character varying COLLATE pg_catalog."default",
```

```
    price bigint,
    "UserID" character varying COLLATE pg_catalog."default",
    "Upload_date" date,
    "Upload_time" time with time zone,
    CONSTRAINT "Artwork_pkey" PRIMARY KEY ("ArtID"),
    CONSTRAINT fk1 FOREIGN KEY ("UserID")
        REFERENCES art_gallery."User" ("UserID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk2 FOREIGN KEY ("ExhibitionID")
        REFERENCES art_gallery."Exhibition" ("ExhibitionID") MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Artwork"
    OWNER to postgres;
```



Query | Query History

```
SELECT * FROM art_gallery."Artwork"
ORDER BY "ArtID" ASC
```

Data output | Messages | Notifications

	ArtID [PK] character varying	Art_type character varying	ExhibitionID character varying	year bigint	title character varying	price bigint	UserID character varying	Upload_date date
1	1	Singing	85	1923	h0tS4	81096	20	2005-11-07
2	10	Singing	93	1902	890Kt	3725	21	1993-12-26
3	100	Singing	56	1955	LJWpL	75917	27	2000-01-05
4	11	Singing	81	1914	uy5gF	4408	36	2003-02-19
5	12	Singing	61	1982	tGTVZ	26401	56	1989-05-27
6	13	Singing	55	1986	yuKD4	13108	47	1973-11-07
7	14	Singing	54	2013	wiU8J	15143	4	2015-09-14
8	15	Singing	37	1989	VL7a8	20656	40	1998-07-15
9	16	Singing	13	1979	6Cgoj	27558	58	2020-09-13
10	17	Singing	89					

✓ Successfully run. Total query runtime: 458 msec. 100 rows affected.

Total rows: 100 of 100 | Query complete 00:00:00.458 | Ln 1, Col 1

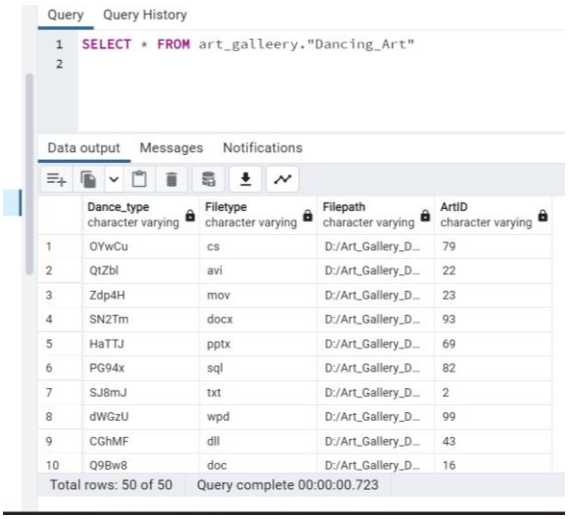
3.Dancing Art

Attributes: Dance Type, File path,File Type , Art ID
Primary key:Art ID

Foreign key:Art ID , File path
ArtID reference from table Artwork (ArtID)
DancingArt is a specialization of Artwork

DDL:

```
-- Table: art_gallery.Dancing_Art

-- DROP TABLE IF EXISTS art_gallery."Dancing_Art ";

CREATE TABLE IF NOT EXISTS art_gallery."Dancing_Art "
(
    "Dance_type" character varying COLLATE pg_catalog."default",
    "Filetype" character varying COLLATE pg_catalog."default",
    "Filepath" character varying COLLATE pg_catalog."default",
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Dancing_Art _pkey" PRIMARY KEY ("ArtID"),
    CONSTRAINT fk1 FOREIGN KEY ("ArtID")
        REFERENCES art_gallery."Artwork" ("ArtID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk2 FOREIGN KEY ("Filepath")
        REFERENCES art_gallery."Dancing_file" (filepath) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Dancing_Art "
    OWNER to postgres;
```



Query Query History

```
1  SELECT * FROM art_galleery."Dancing_Art"
2
```

Data output Messages Notifications

	Dance_type character varying	Filetype character varying	Filepath character varying	ArtID character varying
1	OYwCu	cs	D:/Art_Gallery_D...	79
2	QtZbl	avi	D:/Art_Gallery_D...	22
3	Zdp4H	mov	D:/Art_Gallery_D...	23
4	SN2Tm	docx	D:/Art_Gallery_D...	93
5	HaTTJ	pptx	D:/Art_Gallery_D...	69
6	PG94x	sql	D:/Art_Gallery_D...	82
7	SJ8mJ	txt	D:/Art_Gallery_D...	2
8	dWGzU	wpd	D:/Art_Gallery_D...	99
9	CGhMF	dll	D:/Art_Gallery_D...	43
10	Q9Bw8	doc	D:/Art_Gallery_D...	16

Total rows: 50 of 50 Query complete 00:00:00.723

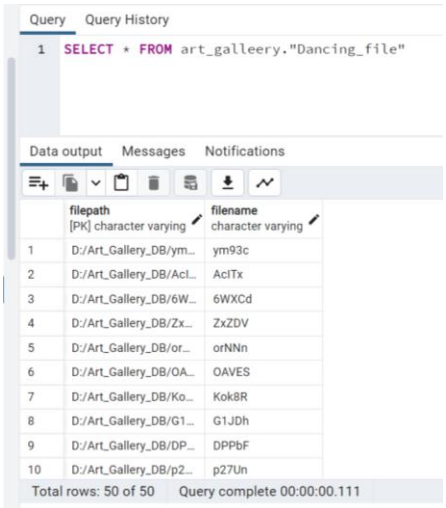
4.Dancing_file

Attributes:file path,file name

Primary Key: file path

DDL:

```
-- Table: art_gallery.Dancing_file

-- DROP TABLE IF EXISTS art_gallery."Dancing_file";

CREATE TABLE IF NOT EXISTS art_gallery."Dancing_file"
(
    filepath character varying COLLATE pg_catalog."default" NOT NULL,
    filename character varying COLLATE pg_catalog."default",
    CONSTRAINT "Dancing_file_pkey" PRIMARY KEY (filepath)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Dancing_file"
    OWNER to postgres;
```



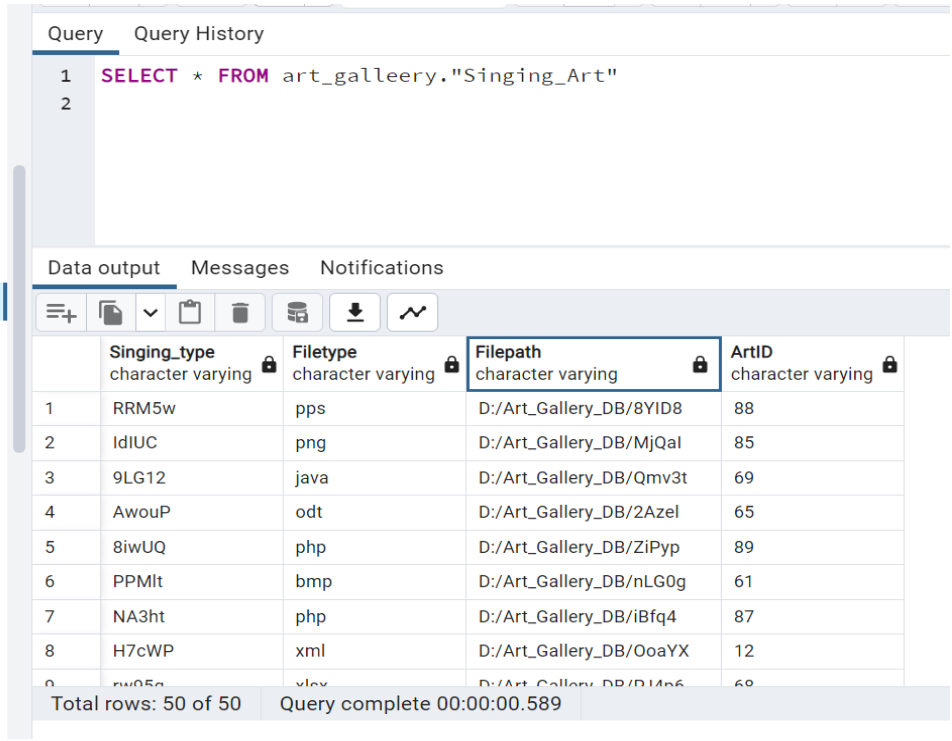
5.Singing Art

Attributs:singing Type, File path,File Type,Art ID
Primary key:Art ID

Foreign key:Art ID
ArtID reference from table Artwork (ArtID)
Singing Art is a specialization of Artwork

```
-- Table: art_gallery.Singing_Art

-- DROP TABLE IF EXISTS art_gallery."Singing_Art ";

CREATE TABLE IF NOT EXISTS art_gallery."Singing_Art "
(
    "Singing_type" character varying COLLATE pg_catalog."default",
    "Filetype" character varying COLLATE pg_catalog."default",
    "Filepath" character varying COLLATE pg_catalog."default",
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Singing_Art_pkey" PRIMARY KEY ("ArtID"),
    CONSTRAINT fk1 FOREIGN KEY ("ArtID")
        REFERENCES art_gallery."Artwork" ("ArtID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk2 FOREIGN KEY ("Filepath")
        REFERENCES art_gallery."Singing_file" (filepath) MATCH SIMPLE
        ON UPDATE NO ACTION
```

```
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Singing_Art "
    OWNER to postgres;
```



Query | Query History

```
1  SELECT * FROM art_galleery."Singing_Art"
2
```

Data output | Messages | Notifications

	Singing_type character varying	Filetype character varying	Filepath character varying	ArtID character varying
1	RRM5w	pps	D:/Art_Gallery_DB/8YID8	88
2	IdlUC	png	D:/Art_Gallery_DB/MjQaI	85
3	9LG12	java	D:/Art_Gallery_DB/Qmv3t	69
4	AwouP	odt	D:/Art_Gallery_DB/2Azel	65
5	8iwUQ	php	D:/Art_Gallery_DB/ZiPyp	89
6	PPMlt	bmp	D:/Art_Gallery_DB/nLG0g	61
7	NA3ht	php	D:/Art_Gallery_DB/iBfq4	87
8	H7cWP	xml	D:/Art_Gallery_DB/OoaYX	12

Total rows: 50 of 50 | Query complete 00:00:00.589

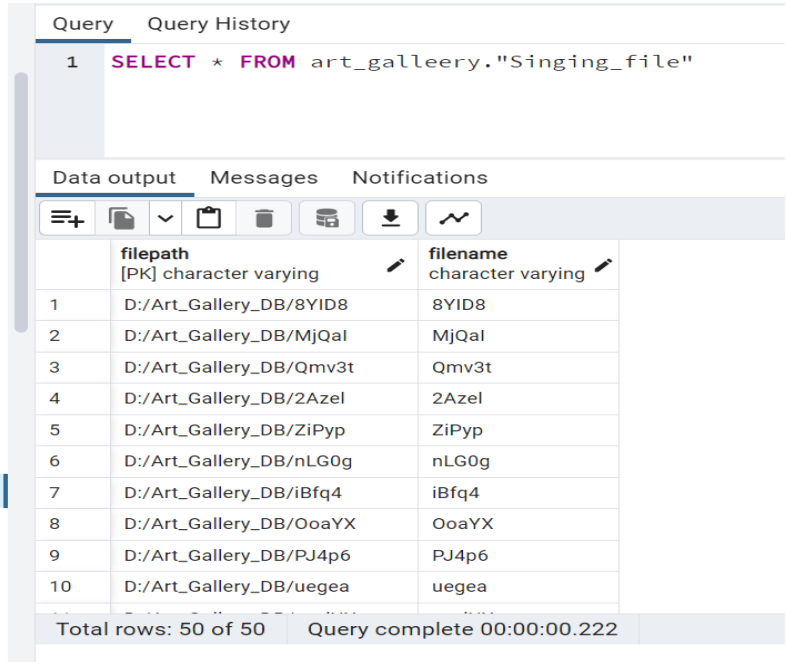
6.Singing_file

Attributes:<u>file path</u>,file name

Primary Key: file path

```
-- Table: art_gallery.Singing_file

-- DROP TABLE IF EXISTS art_gallery."Singing_file";

CREATE TABLE IF NOT EXISTS art_gallery."Singing_file"
(
    filepath character varying COLLATE pg_catalog."default" NOT NULL,
```

```
    filename character varying COLLATE pg_catalog."default",
    CONSTRAINT "Singing_file_pkey" PRIMARY KEY (filepath)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Singing_file"
    OWNER to postgres;
```



Query | Query History

```
1  SELECT * FROM art_galleery."Singing_file"
```

Data output | Messages | Notifications

	filepath [PK] character varying	filename character varying
1	D:/Art_Gallery_DB/8YID8	8YID8
2	D:/Art_Gallery_DB/MjQaI	MjQaI
3	D:/Art_Gallery_DB/Qmv3t	Qmv3t
4	D:/Art_Gallery_DB/2Azel	2Azel
5	D:/Art_Gallery_DB/ZiPyp	ZiPyp
6	D:/Art_Gallery_DB/nLG0g	nLG0g
7	D:/Art_Gallery_DB/iBfq4	iBfq4
8	D:/Art_Gallery_DB/OoaYX	OoaYX
9	D:/Art_Gallery_DB/PJ4p6	PJ4p6
10	D:/Art_Gallery_DB/uegea	uegea

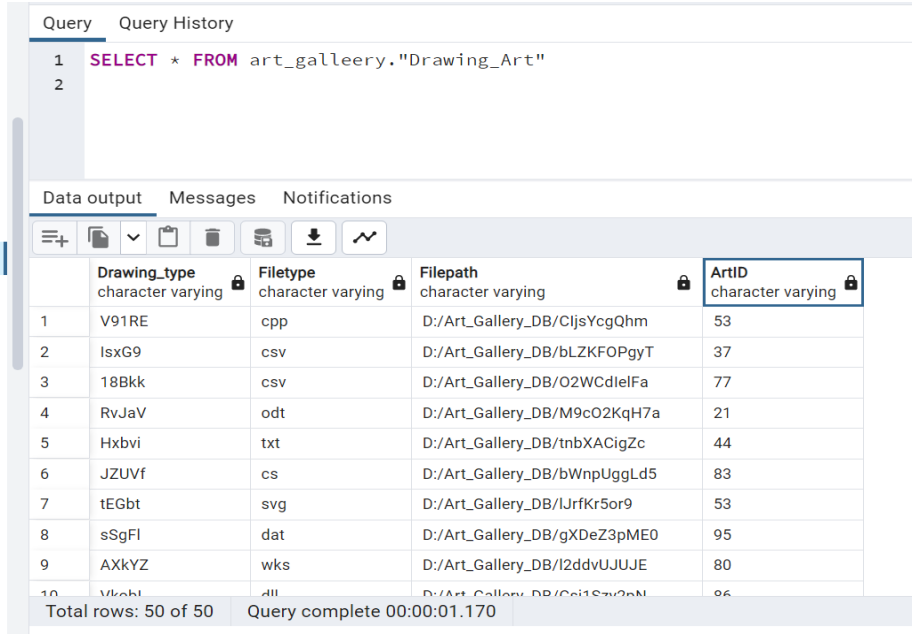
Total rows: 50 of 50 | Query complete 00:00:00.222

7.Drawing Art

Attributes:Drawing Type, File path, File Type,Art ID
Primary key:Art ID
Foreign key:Art ID
ArtID reference from table Artwork (ArtID)
Drawing Art is a specialization of Artwork

```
CREATE TABLE IF NOT EXISTS art_gallery."Drawing_Art "
(
    "Drawing_type" character varying COLLATE pg_catalog."default",
    "Filetype" character varying COLLATE pg_catalog."default",
    "Filepath" character varying COLLATE pg_catalog."default",
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
```

```
    CONSTRAINT "Drawing_Art_pkey" PRIMARY KEY ("ArtID"),
    CONSTRAINT fk1 FOREIGN KEY ("ArtID")
        REFERENCES art_gallery."Artwork" ("ArtID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk2 FOREIGN KEY ("Filepath")
        REFERENCES art_gallery."Drawing_file" (filepath) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Drawing_Art "
    OWNER to postgres;
```



Query | Query History

```
1  SELECT * FROM art_galleery."Drawing_Art"
2
```

Data output | Messages | Notifications

	Drawing_type character varying	Filetype character varying	Filepath character varying	ArtID character varying
1	V91RE	cpp	D:/Art_Gallery_DB/CIjsYcgQhm	53
2	IsxG9	csv	D:/Art_Gallery_DB/bLZKFOPgyT	37
3	18Bkk	csv	D:/Art_Gallery_DB/O2WCdIelFa	77
4	RvJaV	odt	D:/Art_Gallery_DB/M9cO2KqH7a	21
5	Hxbvi	txt	D:/Art_Gallery_DB/tnbXACigZc	44
6	JZUVf	cs	D:/Art_Gallery_DB/bWnpUggLd5	83
7	tEGbt	svg	D:/Art_Gallery_DB/lJrfKr5or9	53
8	sSgFl	dat	D:/Art_Gallery_DB/gXDeZ3pME0	95
9	AXkYZ	wks	D:/Art_Gallery_DB/l2ddvUJUJE	80
10	VkobI	dll	D:/Art_Gallery_DB/Gci1Szv2pN	86

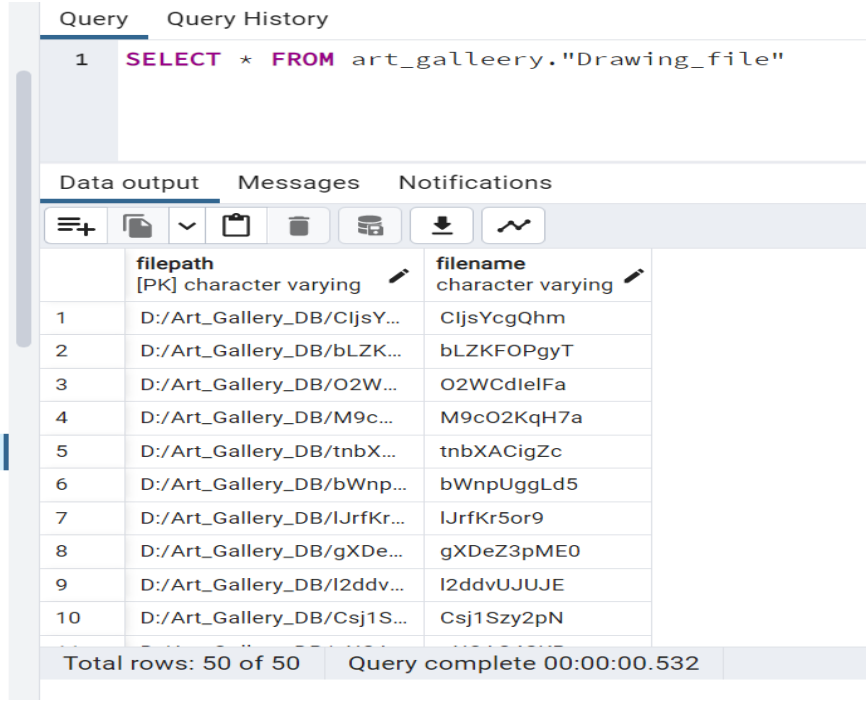
Total rows: 50 of 50 | Query complete 00:00:01.170

8.Drawing_file

Attributes:<u>file path</u>,file name

Primary Key: file path

```
CREATE TABLE IF NOT EXISTS art_gallery."Drawing_file"
(
    filepath character varying COLLATE pg_catalog."default" NOT
NULL,
    filename character varying COLLATE pg_catalog."default",
    CONSTRAINT "Drawing_file_pkey" PRIMARY KEY (filepath)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Drawing_file"
    OWNER to postgres;
```



Query | Query History

```
1  SELECT * FROM art_galleery."Drawing_file"
```

Data output | Messages | Notifications

	filepath [PK] character varying	filename character varying
1	D:/Art_Gallery_DB/CljsY...	CljsYcgQhm
2	D:/Art_Gallery_DB/bLZK...	bLZKFOPgyT
3	D:/Art_Gallery_DB/O2W...	O2WCdIelFa
4	D:/Art_Gallery_DB/M9c...	M9cO2KqH7a
5	D:/Art_Gallery_DB/tnbX...	tnbXACigZc
6	D:/Art_Gallery_DB/bWnp...	bWnpUggLd5
7	D:/Art_Gallery_DB/lJrfKr...	lJrfKr5or9
8	D:/Art_Gallery_DB/gXDe...	gXDeZ3pME0
9	D:/Art_Gallery_DB/l2ddv...	l2ddvUJUJE
10	D:/Art_Gallery_DB/Csj1S...	Csj1Szy2pN

Total rows: 50 of 50 | Query complete 00:00:00.532

9. Exhibition

Attributs: ExhibitionID , place , Start Date , End Date
Primary key: ExhibitionID

Functional dependency :
ExhibitionID -> { place , Start Date , End Date}

DDL:

```
-- Table: Art_gallery_db.Exhibition

-- DROP TABLE IF EXISTS "Art_gallery_db"."Exhibition";

CREATE TABLE IF NOT EXISTS "art_gallery"."Exhibition"
(
    "ExhibitionID" character varying COLLATE pg_catalog."default" NOT
NULL,
    start_date date,
    end_date date,
    place character varying COLLATE pg_catalog."default",
    CONSTRAINT "Exhibition_pkey" PRIMARY KEY ("ExhibitionID")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS "Art_gallery_db"."Exhibition"
    OWNER to postgres;
```

```
SELECT * FROM art_gallery."Exhibition"
ORDER BY "ExhibitionID" ASC
```

Data output | Messages | Notifications

	ExhibitionID [PK] character varying	start_date date	end_date date	place character varying
1	1	2009-06-24	1970-05-29	Oakland
2	10	1964-04-03	1974-05-29	Laredo
3	100	1955-01-04	1982-09-19	Jersey City
4	11	2016-07-30	1959-05-06	Valetta
5	12	2016-02-23	1976-01-28	Pittsburgh
6	13	2017-06-18	2012-02-09	San Jose
7	14	2016-11-13	1975-09-29	London
8	15	2009-05-24	1963-05-15	Salem
9	16	2014-04-11	2012-01-03	Otawa
10	17	1962-12-14	1986-02-01	Dallas

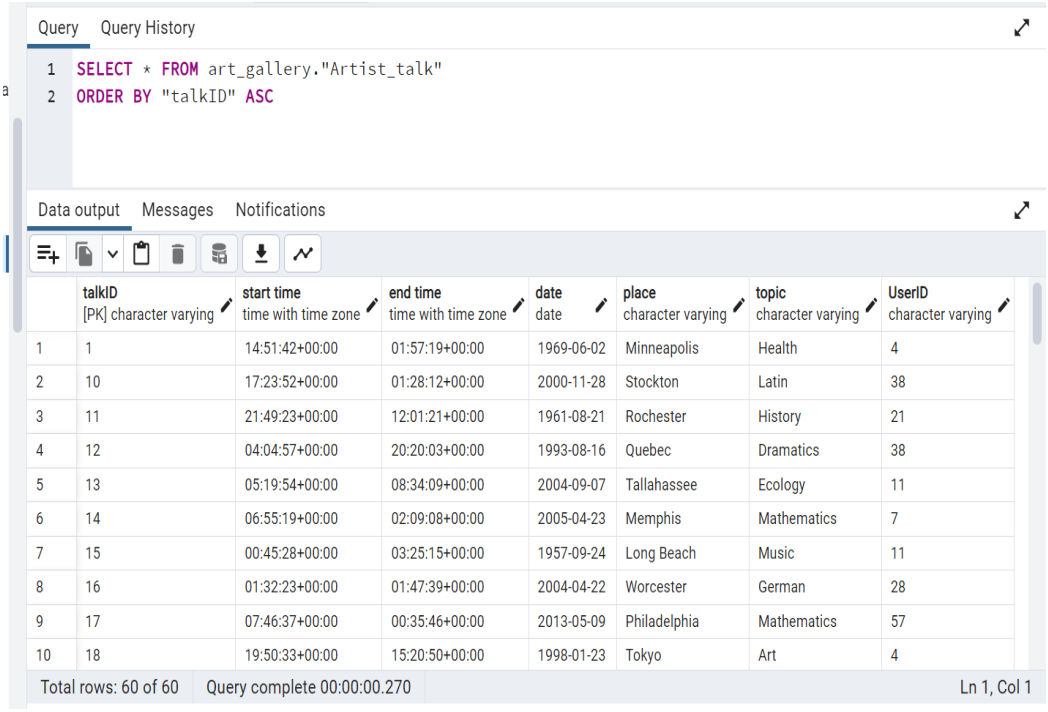
Total rows: 100 of 100 | Query complete 00:00:00.241

10. Artist_talk

Attributs: talkID , start time , end time , date , place , topic , UserID
Primary key: talkID
Foreign key: UserID
UserID reference from table Artist(userID)

-- Table: art_gallery.Artist_talk

```
-- DROP TABLE IF EXISTS art_gallery."Artist_talk";

CREATE TABLE IF NOT EXISTS art_gallery."Artist_talk"
(
    "talkID" character varying COLLATE pg_catalog."default" NOT NULL,
    "start time" time with time zone,
    "end time" time with time zone,
    date date,
    place character varying COLLATE pg_catalog."default",
    topic character varying COLLATE pg_catalog."default",
    "UserID" character varying COLLATE pg_catalog."default",
    CONSTRAINT "PK1" PRIMARY KEY ("talkID"),
    CONSTRAINT "FK1" FOREIGN KEY ("UserID")
        REFERENCES art_gallery."User" ("UserID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Artist_talk"
    OWNER to postgres;
```



```
SELECT * FROM art_gallery."Artist_talk"
ORDER BY "talkID" ASC
```

Query | Query History

Data output | Messages | Notifications

	talkID [PK] character varying	start time time with time zone	end time time with time zone	date date	place character varying	topic character varying	UserID character varying
1	1	14:51:42+00:00	01:57:19+00:00	1969-06-02	Minneapolis	Health	4
2	10	17:23:52+00:00	01:28:12+00:00	2000-11-28	Stockton	Latin	38
3	11	21:49:23+00:00	12:01:21+00:00	1961-08-21	Rochester	History	21
4	12	04:04:57+00:00	20:20:03+00:00	1993-08-16	Quebec	Dramatics	38
5	13	05:19:54+00:00	08:34:09+00:00	2004-09-07	Tallahassee	Ecology	11
6	14	06:55:19+00:00	02:09:08+00:00	2005-04-23	Memphis	Mathematics	7
7	15	00:45:28+00:00	03:25:15+00:00	1957-09-24	Long Beach	Music	11
8	16	01:32:23+00:00	01:47:39+00:00	2004-04-22	Worcester	German	28
9	17	07:46:37+00:00	00:35:46+00:00	2013-05-09	Philadelphia	Mathematics	57
10	18	19:50:33+00:00	15:20:50+00:00	1998-01-23	Tokyo	Art	4

Total rows: 60 of 60 | Query complete 00:00:00.270 | Ln 1, Col 1

11. Artist

Attributs: Field , Qualification, User ID
Primary key: UserID
Foreign key: UserID
UserID reference from table User(UserID)

```
-- Table: art_gallery.Artist

-- DROP TABLE IF EXISTS art_gallery."Artist";

CREATE TABLE IF NOT EXISTS art_gallery."Artist"
(
    "UserID" character varying COLLATE pg_catalog."default",
    "Field" character varying COLLATE pg_catalog."default",
    "Qualification" character varying COLLATE pg_catalog."default",
    CONSTRAINT "FK1" FOREIGN KEY ("UserID")
        REFERENCES art_gallery."User" ("UserID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Artist"
OWNER to postgres;

```
1  SELECT * FROM art_gallery."Artist"
2
```

Data output | Messages | Notifications

	UserID character varying	Field character varying	Qualification character varying
1	1	DHvPbuDNpB	Ancient Civilizati…
2	2	9Xl9keO1Ua	Earth Science
3	3	CP4YfBuST6	Music
4	4	YFlBxWERb6	Ecology
5	5	4w2TOO4EpX	History
6	6	fFgUDPBcWr	Physical Education
7	7	q8gXir1wD9	Trigonometry
8	8	gVIfLxgXPR	Design and techn…
9	9	ljgfcTzP8j	Instrumental Mu…

Total rows: 50 of 50 | Query complete 00:00:00.246

12. Comment

Attributs: comment description , <u>UserID , ArtID</u>
Primary key: UserID , ArtID

Foreign key: UserID , ArtID
UserID reference from table User(UserID)
ArtID reference from table Artwork (ArtID)

Comment is a many to many relationship between entity User and Artwork

Functional dependency :
{UserID , ArtID} -> comment description

DDL:

-- Table: art_gallery.Comment

```
-- DROP TABLE IF EXISTS art_gallery."Comment";

CREATE TABLE IF NOT EXISTS art_gallery."Comment"
(
    "UserID" character varying COLLATE pg_catalog."default" NOT NULL,
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
    comment_description character varying COLLATE pg_catalog."default",
    CONSTRAINT "Comment_pkey" PRIMARY KEY ("ArtID", "UserID"),
    CONSTRAINT "FK1" FOREIGN KEY ("UserID")
        REFERENCES art_gallery."User" ("UserID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "FK2" FOREIGN KEY ("ArtID")
        REFERENCES art_gallery."Artwork" ("ArtID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Comment"
    OWNER to postgres;
```

Query | Query History

```
1  SELECT * FROM art_gallery."Comment"
2  ORDER BY "UserID" ASC, "ArtID" ASC
```

Data output | Messages | Notifications

	UserID [PK] character varying	ArtID [PK] character varying	comment_description character varying
1	1	70	nlVMWyDcymKP4h4
2	10	27	KxyriXquoemkb50
3	11	11	Sd4VLJmbVXZKY1Y
4	12	87	u7WptycXipdfa2K
5	13	40	fo2lOEFBKWVL7fV
6	14	78	FuzeiLWeAhxfpHB
7	15	47	MApu9xnq1VeXcRZ
8	16	5	UrZpDq0URYkrQkV
9	17	6	V2860lxSGImNq9B
10	18	73	HuL8rVZgKUdlM6a

Total rows: 50 of 50 | Query complete 00:00:00.341

13.Available

Attributes:Time , Exhibition cost , ArtID , ExhibitionID
Primary key: ArtID , ExhibitionID
Foreign key: ArtID , ExhibitionID
ArtID reference from table Artwork (ArtID)
ExhibitionID reference from table Exhibition(ExhibitionID)

Available is a many to many relationship between entity Exhibition and Artwork

Functional dependency :
{ArtID , ExhibitionID} -> {Time , Exhibition cost}

DDL:

```
-- Table: art_gallery.Available

-- DROP TABLE IF EXISTS art_gallery."Available";

CREATE TABLE IF NOT EXISTS art_gallery."Available"
(
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
    "ExhibitionID" character varying COLLATE pg_catalog."default" NOT
NULL,
    "Time" time with time zone,
    "Exhibition_cost" bigint,
    CONSTRAINT "Available_pkey" PRIMARY KEY ("ExhibitionID", "ArtID"),
    CONSTRAINT fk1 FOREIGN KEY ("ArtID")
        REFERENCES art_gallery."Artwork" ("ArtID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk2 FOREIGN KEY ("ExhibitionID")
        REFERENCES art_gallery."Exhibition" ("ExhibitionID") MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;
```

ALTER TABLE IF EXISTS art_gallery."Available"
OWNER to postgres;

```
SELECT * FROM art_gallery."Available"
ORDER BY "ArtID" ASC, "ExhibitionID" ASC
```

Data output | Messages | Notifications

	ArtID [PK] character varying	ExhibitionID [PK] character varying	Time time with time zone	Exhibition_cost bigint
1	1	66	09:27:15+00:00	5533
2	10	40	22:51:31+00:00	8848
3	15	19	22:16:52+00:00	5980
4	15	41	11:13:04+00:00	3955
5	15	58	09:31:30+00:00	6479
6	16	47	12:59:23+00:00	5395
7	17	90	18:46:49+00:00	3140
8	18	61	13:23:33+00:00	4601
9	19	53	03:17:49+00:00	3117
10	20	35	16:54:53+00:00	7409

Total rows: 100 of 100 Query complete 00:00:00.306

14. Participate_exhibition

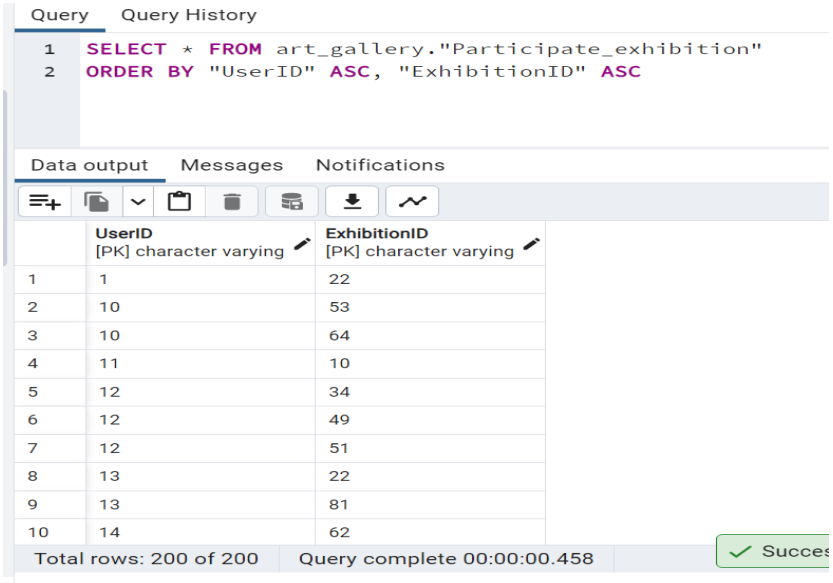
Attributes: <u>UserID , ExhibitionID</u>
Primary key:UserID , ExhibitionID

Foreign key: UserID , ExhibitionID
ExhibitionID reference from table Exhibition(ExhibitionID)
UserID reference from table User(UserID)
Participate_exhibition is a many to many relationship between entity Exhibition and User

Functional dependency :
{UserID , ExhibitionID} -> {UserID , ExhibitionID}

```
-- Table: art_gallery.Participate_exhibition

-- DROP TABLE IF EXISTS art_gallery."Participate_exhibition";

CREATE TABLE IF NOT EXISTS art_gallery."Participate_exhibition"
(
    "UserID" character varying COLLATE pg_catalog."default" NOT NULL,
```

```
    "ExhibitionID" character varying COLLATE pg_catalog."default" NOT
NULL,
    CONSTRAINT "Participate_exhibition_pkey" PRIMARY KEY ("UserID",
"ExhibitionID"),
    CONSTRAINT fk1 FOREIGN KEY ("UserID")
        REFERENCES art_gallery."User" ("UserID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk2 FOREIGN KEY ("ExhibitionID")
        REFERENCES art_gallery."Exhibition" ("ExhibitionID") MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Participate_exhibition"
    OWNER to postgres;
```



15. Participate_talk

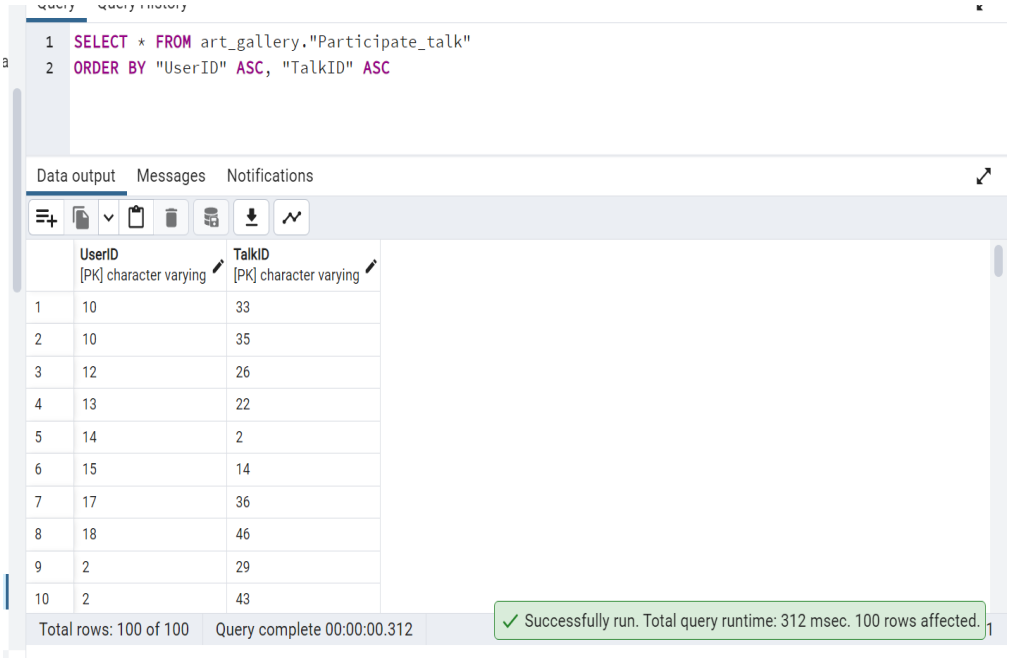
Attributes: <u>UserID , talkID</u>

Primary key:UserID ,talkID

Foreign key: UserID , talkID
UserID reference from table User(UserID)
talkID reference from table Artist_talk(talkID)

Participate_talk is a many to many relationship between entity Artist_talk and User

```
-- Table: art_gallery.Participate_talk

-- DROP TABLE IF EXISTS art_gallery."Participate_talk";

CREATE TABLE IF NOT EXISTS art_gallery."Participate_talk"
(
    "UserID" character varying COLLATE pg_catalog."default" NOT NULL,
    "TalkID" character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Participate_talk_pkey" PRIMARY KEY ("UserID",
"TalkID"),
    CONSTRAINT fk1 FOREIGN KEY ("UserID")
        REFERENCES art_gallery."User" ("UserID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk2 FOREIGN KEY ("TalkID")
        REFERENCES art_gallery."Artist_talk" ("talkID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Participate_talk"
    OWNER to postgres;
```



16. Bidding

Attributes: Start price, End Price, <u>ArtID ,UserID,</u>

Primary key:ArtID ,UserID,
Foreign key: ArtID ,UserID

ArtID reference from table Artwork (ArtID)
UserID reference from table User(UserID)

DDL:

```
-- Table: art_gallery.Bidding

-- DROP TABLE IF EXISTS art_gallery."Bidding";

CREATE TABLE IF NOT EXISTS art_gallery."Bidding"
(
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
    "UserID" character varying COLLATE pg_catalog."default" NOT NULL,
    start_price bigint,
    end_price bigint,
    CONSTRAINT "Bidding_pkey" PRIMARY KEY ("ArtID", "UserID"),
    CONSTRAINT fk1 FOREIGN KEY ("ArtID")
```

```
    REFERENCES art_gallery."Artwork" ("ArtID") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION,
  CONSTRAINT fk2 FOREIGN KEY ("UserID")
    REFERENCES art_gallery."User" ("UserID") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Bidding"
  OWNER to postgres;
```

```
1  SELECT * FROM art_gallery."Bidding"
2  ORDER BY "ArtID" ASC, "UserID" ASC
```

Data output | Messages | Notifications

	ArtID [PK] character varying	UserID [PK] character varying	start_price bigint	end_price bigint
1	1	77	1628	5473
2	10	35	2404	15931
3	11	26	2017	14499
4	11	30	1270	6813
5	12	40	1907	12565
6	12	96	4189	12459
7	13	82	2116	5895
8	14	51	2666	11879
9	15	47	1137	10299
10	15	96	4942	16608

Total rows: 150 of 150 | Query complete 00:00:00.389 | Successfully run. Total query runtime: 389 msec. 150 rows affected.

17. Order

Attributes: Shipment_date , Date , Shipment time , Shipment from , Shipment to , Time , payment method , <u>ArtID ,</u> UserID

Primary key: ArtID
Foreign key: ArtID , UserID
ArtID reference from table Artwork (ArtID)
UserID reference from table User(UserID)
Order is a many to one relationship between entity Artwork and User.

DDL:

```
-- Table: art_gallery.Order

-- DROP TABLE IF EXISTS art_gallery."Order";

CREATE TABLE IF NOT EXISTS art_gallery."Order"
(
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
    "Date" date,
    "Shipment_date" date,
    "Shipment_time" time with time zone,
    "Shipment_from" character varying COLLATE pg_catalog."default",
    "Shipment_to" character varying COLLATE pg_catalog."default",
    "Time" time with time zone,
    "payment_method" character varying COLLATE pg_catalog."default",
    "UserID" character varying COLLATE pg_catalog."default",
    CONSTRAINT "Order_pkey" PRIMARY KEY ("ArtID"),
    CONSTRAINT fk1 FOREIGN KEY ("ArtID")
        REFERENCES art_gallery."Artwork" ("ArtID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Order"
    OWNER to postgres;
```

```
SELECT * FROM art_gallery."Order"
ORDER BY "ArtID" ASC
```

Data output | Messages | Notifications

	ArtID [PK] character varying	Date date	Shipment_date date	Shipment_time time with time zone	Shipment_from character varying	Shipment_to character varying	Time time with time zone	pay cha
1	10	1999-05-17	1964-01-12	02:10:53+00:00	Long Beach	Fullerton	16:03:01+00:00	Off
2	11	1971-07-25	1997-01-07	21:49:54+00:00	Saint Paul	Otawa	06:47:34+00:00	Onl
3	12	1959-06-22	1980-01-10	05:33:51+00:00	Glendale	Reno	04:29:31+00:00	Off
4	13	1954-05-21	1960-07-30	15:28:14+00:00	Innsbruck	Lisbon	10:07:20+00:00	Onl
5	14	2015-02-20	1979-08-24	20:37:13+00:00	Phoenix	Laredo	05:43:08+00:00	Off
6	16	1957-06-30	2008-09-01	04:21:10+00:00	Tallahassee	San Diego	03:36:27+00:00	Onl
7	17	1992-12-11	1991-10-28	04:52:27+00:00	Santa Ana	Las Vegas	09:18:27+00:00	Off
8	23	1968-06-10	2014-08-14	13:54:32+00:00	San Bernardino	Colorado Springs	05:25:49+00:00	Onl
9	25	2015-06-26	2001-02-12	09:56:14+00:00	Pittsburgh	Washington	14:11:41+00:00	Off

Total rows: 50 of 50 | Query complete 00:00:00.354 | Successfully run. Total query runtime: 354 msec. 50 rows affected.

18. Rental

Attributes: Shipment date , startdate, enddate , Shipment time , Shipment from ,Shipment to , Time , payment method , UserID , ArtID

Primary key: ArtID
Foreign key: ArtID , UserID
ArtID reference from table Artwork (ArtID)
UserID reference from table User(UserID)
Rental is a many to one relationship between entity Artwork and User.

DDL:

```
-- Table: art_gallery.Order

-- DROP TABLE IF EXISTS art_gallery."Order";

CREATE TABLE IF NOT EXISTS art_gallery."Rental"
(
    "ArtID" character varying COLLATE pg_catalog."default" NOT NULL,
    "Date" date,
    "Shipment_date" date,
        "start_date" date,
        "end_date" date,
    "Shipment_time" time with time zone,
```

```
    "Shipment_from" character varying COLLATE pg_catalog."default",
    "Shipment_to" character varying COLLATE pg_catalog."default",
    "Time" time with time zone,
    "payment_method" character varying COLLATE pg_catalog."default",
    "UserID" character varying COLLATE pg_catalog."default",
    CONSTRAINT "Rental_pkey" PRIMARY KEY ("ArtID"),
    CONSTRAINT fk1 FOREIGN KEY ("ArtID")
        REFERENCES art_gallery."Artwork" ("ArtID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."Rental"
    OWNER to postgres;
```

```
1  SELECT * FROM art_gallery."Rental"
2  ORDER BY "ArtID" ASC
```

Data output | Messages | Notifications

	ArtID [PK] character varying	Date date	Shipment_date date	start_date date	end_date date	Shipment_time time with time zone	Shipment_from character varying	Shipment_to character varying
1	1	1980-06-06	2015-02-27	1995-06-30	2005-04-19	10:29:49+00:00	Orlando	Kansas City
2	13	1984-10-12	2019-12-12	1963-10-07	2013-03-19	12:23:22+00:00	Rome	Bakersfield
3	15	1964-07-30	2019-02-27	2001-02-19	1982-09-23	06:53:27+00:00	Madison	Salt Lake City
4	17	1971-09-08	2022-08-26	1995-09-14	1978-02-16	07:55:07+00:00	Berlin	Charlotte
5	2	1967-06-29	2008-01-31	1958-01-13	2016-01-03	17:08:42+00:00	Baltimore	Madison
6	20	1964-03-25	2015-03-03	1992-10-18	1988-10-12	02:15:37+00:00	Long Beach	Bellevue
7	22	1973-12-01	2009-08-07	1967-05-22	1988-05-16	06:26:58+00:00	Miami	Lakewood
8	25	2019-07-24	2008-03-11	1956-01-25	1979-02-23	02:31:50+00:00	Fullerton	El Paso
9	26	1994-11-05	2020-04-01	2022-02-04	1975-09-25	21:10:01+00:00	Wien	Worcester

Total rows: 50 of 50 | Query complete 00:00:00.524 | Successfully run. Total query runtime: 524 msec. 50 rows affected.

19. User_phone_no

Attributs: <u>UserID , Phone_no</u>

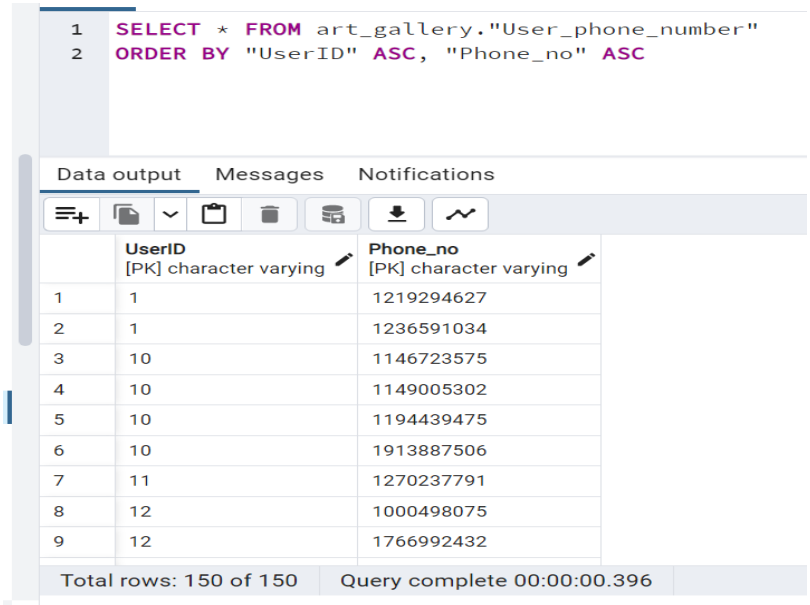
Primary key: UserID , Phone_no

Foreign key: UserID

UserID reference from table User(UserID)

DDL:

```
-- Table: art_gallery.User_phone_number

-- DROP TABLE IF EXISTS art_gallery."User_phone_number";

CREATE TABLE IF NOT EXISTS art_gallery."User_phone_number"
(
    "UserID" character varying COLLATE pg_catalog."default" NOT NULL,
    "Phone_no" character varying COLLATE pg_catalog."default" NOT
NULL,
    CONSTRAINT "User_phone_number_pkey" PRIMARY KEY ("UserID",
"Phone_no"),
    CONSTRAINT fk1 FOREIGN KEY ("UserID")
        REFERENCES art_gallery."User" ("UserID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS art_gallery."User_phone_number"
    OWNER to postgres;
```



```
1  SELECT * FROM art_gallery."User_phone_number"
2  ORDER BY "UserID" ASC, "Phone_no" ASC
```

Data output | Messages | Notifications

	UserID [PK] character varying	Phone_no [PK] character varying
1	1	1219294627
2	1	1236591034
3	10	1146723575
4	10	1149005302
5	10	1194439475
6	10	1913887506
7	11	1270237791
8	12	1000498075
9	12	1766992432

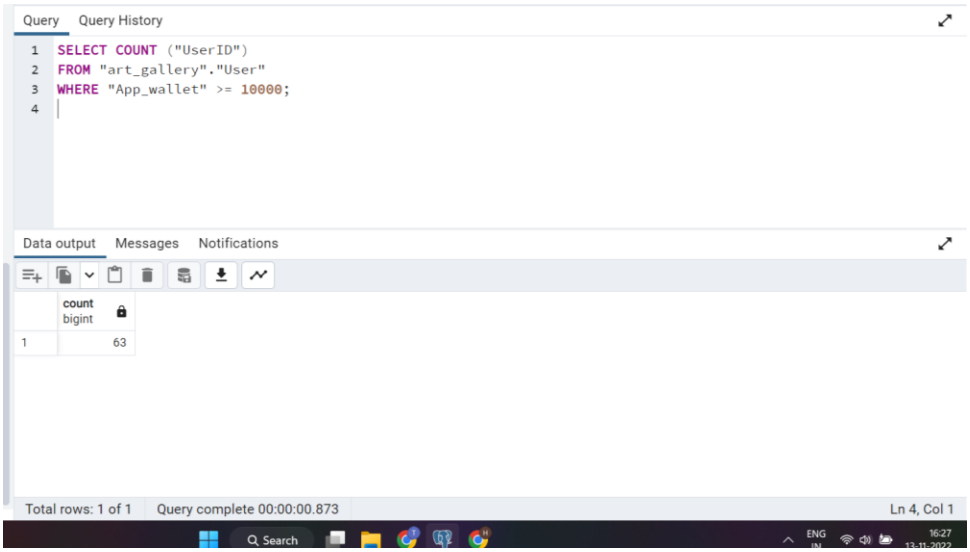
Total rows: 150 of 150 | Query complete 00:00:00.396

- **SQL Queries**

  - **Simple Query**

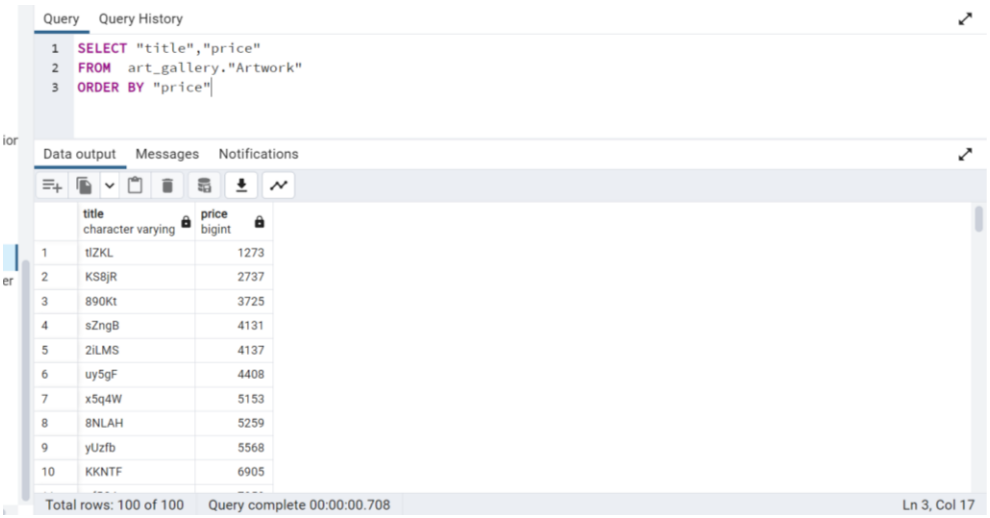
1. **Count the number of users whose app wallet is greater than or equal to 10000.**

```
SELECT COUNT ("UserID")
FROM "art_gallery"."User"
WHERE "App_wallet" >= 10000;
```

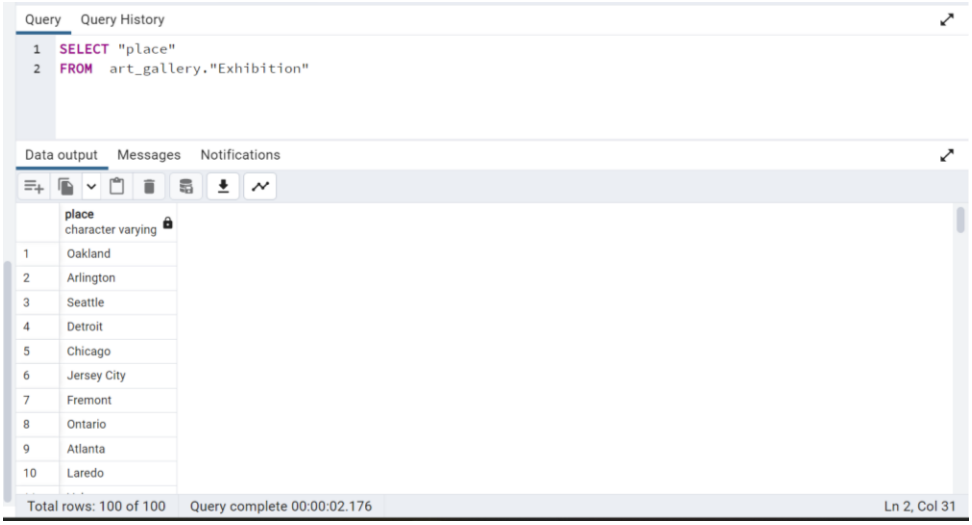


2. **Sort all the art in increasing order of price and display the title of art with price.**

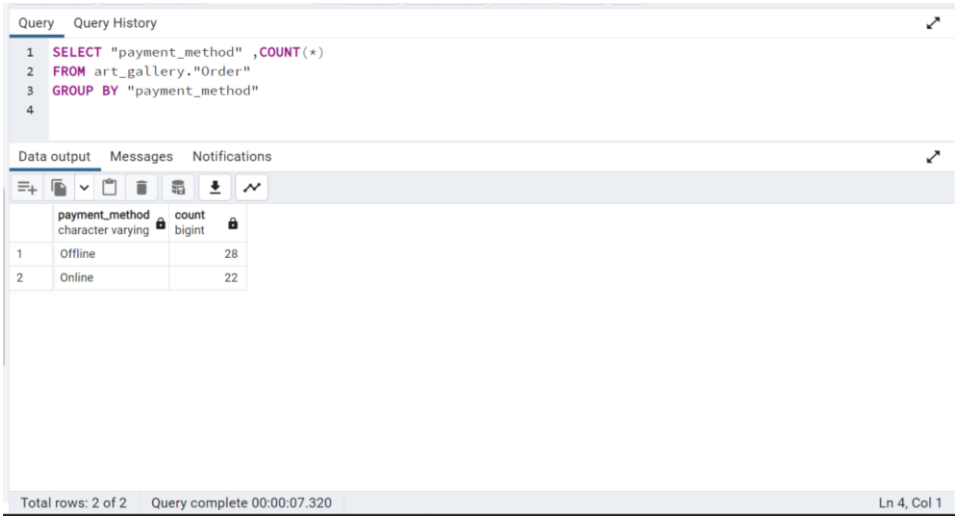
```
SELECT "title","price"
FROM  art_gallery."Artwork"
ORDER BY "price"
```



3. **Display the locations of the exhibitions.**
   SELECT "place"
   FROM art_gallery."Exhibition"

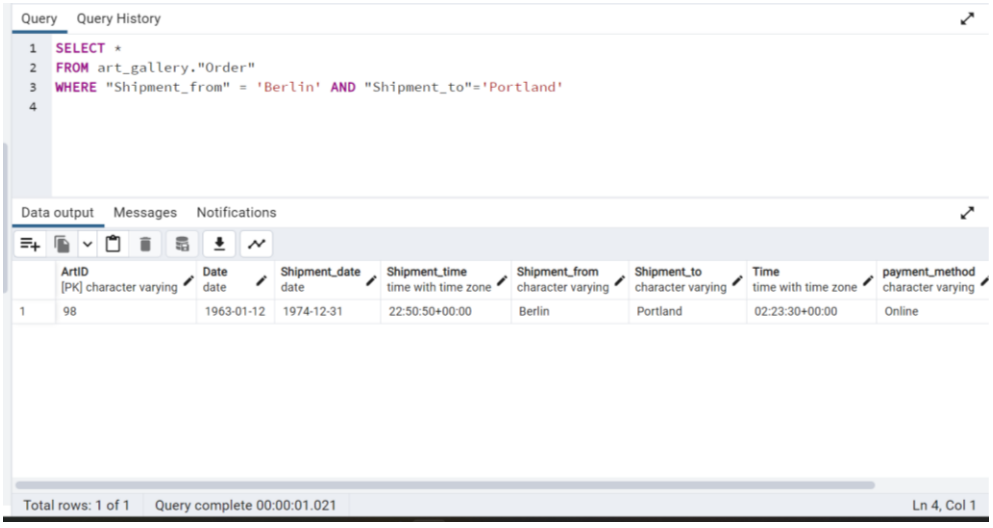


4. **Count the number of transactions that happen for each payment method.**
   SELECT "payment_method" ,COUNT(*)
   FROM art_gallery."Order"
   GROUP BY "payment_method"



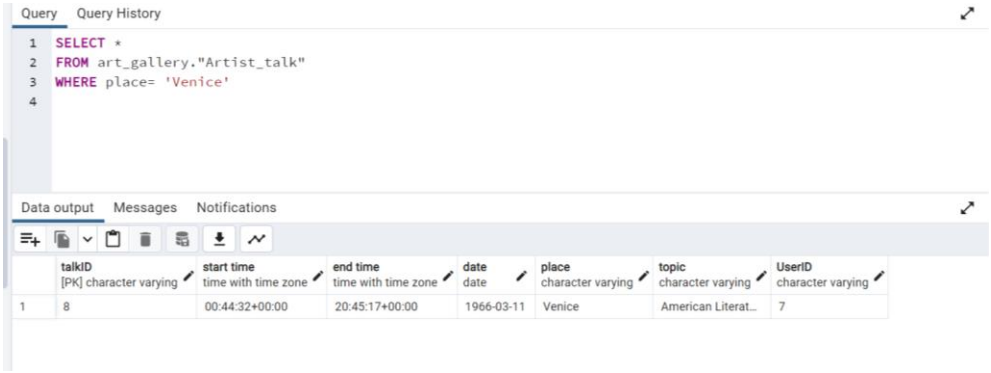
5. **Display the detail of order for which shipment happened from Berlin to Portland.**

```
SELECT *
FROM art_gallery."Order"
WHERE "Shipment_from" = 'Berlin' AND "Shipment_to"='Portland'
```



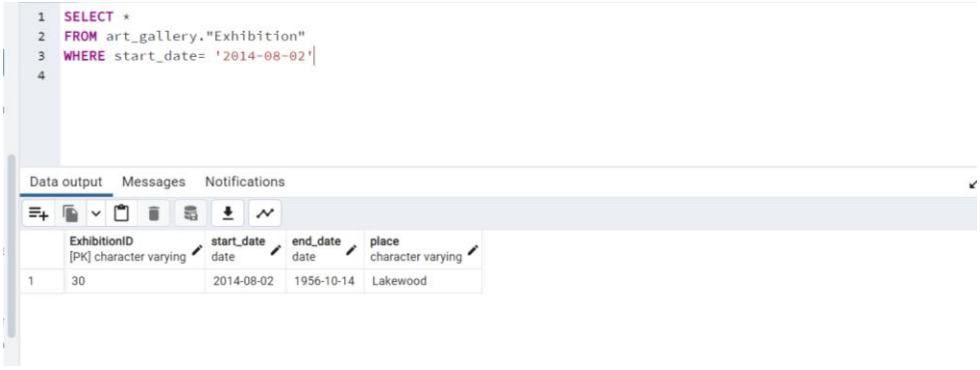
6. **Show information about the artist lecture that is held in Venice.**

```
SELECT *
FROM art_gallery."Artist_talk"
WHERE place= 'Venice'
```



7. **Show the exhibition details which is held on date mm/dd/yy.**

```
SELECT *
FROM art_gallery."Exhibition"
WHERE start_date= '2014-08-02'
```

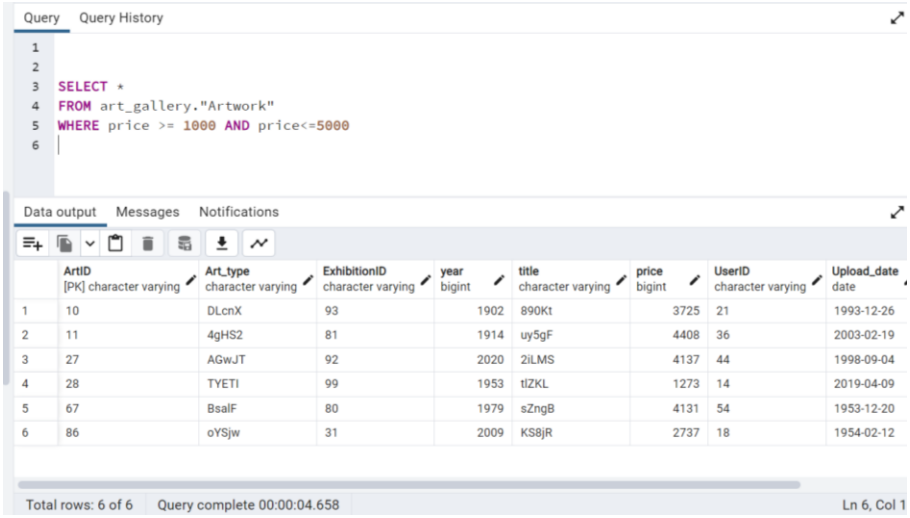


8. **Count the number of male and female users.**

```
SELECT  "Gender" ,  COUNT(*)
FROM art_gallery."User"
GROUP BY "Gender";
```

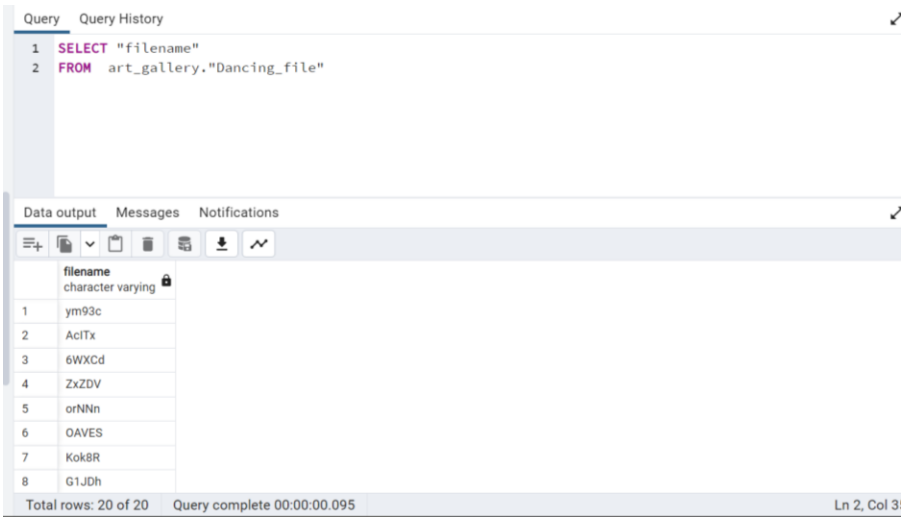
9. **Show the details of artwork whose price is in the range 1000-5000.**

```
SELECT *
FROM art_gallery."Artwork"
WHERE price >= 1000 AND price<=5000
```



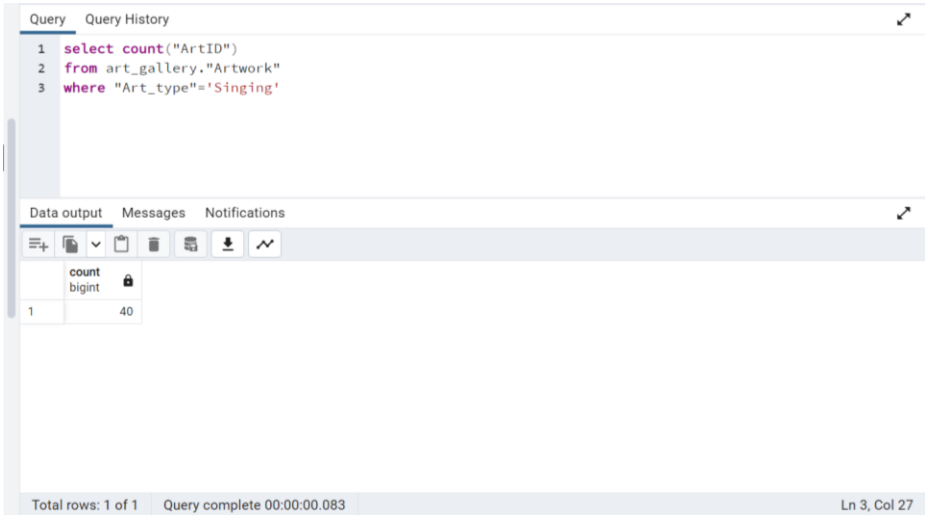
10. **Show the file names of all video files.**

```
SELECT "filename"
FROM  art_gallery."Dancing_file"
```



11. Count the number of 'Singing' related pieces of art.

```
select count("ArtID")
from art_gallery."Artwork"
where "Art_type"='Singing'
```

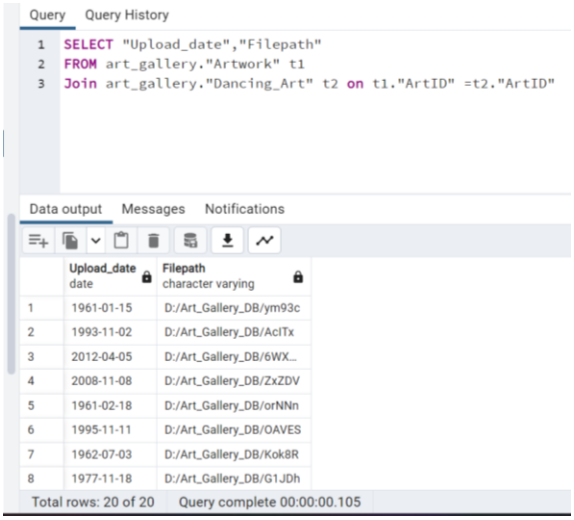


- **Complex Query**

12. Display the Upload date and file path of artwork which type is Dancing.

```
SELECT "Upload_date","Filepath"
FROM art_gallery."Artwork" t1
```

Join art_gallery."Dancing_Art" t2 on t1."ArtID" =t2."ArtID"



Query | Query History

```
SELECT "Upload_date","Filepath"
FROM art_gallery."Artwork" t1
Join art_gallery."Dancing_Art" t2 on t1."ArtID" =t2."ArtID"
```

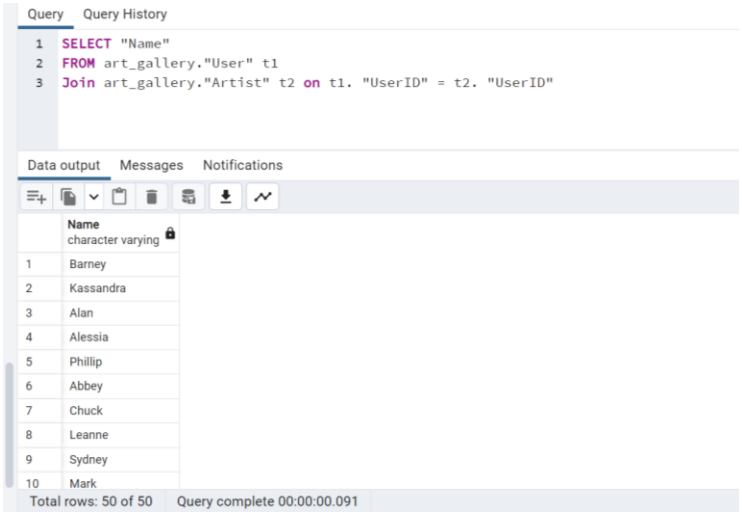
Data output | Messages | Notifications

	Upload_date date	Filepath character varying
1	1961-01-15	D:/Art_Gallery_DB/ym93c
2	1993-11-02	D:/Art_Gallery_DB/AcITx
3	2012-04-05	D:/Art_Gallery_DB/6WX...
4	2008-11-08	D:/Art_Gallery_DB/ZxZDV
5	1961-02-18	D:/Art_Gallery_DB/orNNn
6	1995-11-11	D:/Art_Gallery_DB/OAVES
7	1962-07-03	D:/Art_Gallery_DB/Kok8R
8	1977-11-18	D:/Art_Gallery_DB/G1JDh

Total rows: 20 of 20 | Query complete 00:00:00.105

13. Display the name of the artist .

SELECT "Name"
FROM art_gallery."User" t1
Join art_gallery."Artist" t2 on t1. "UserID" = t2. "UserID"



Query | Query History

```
SELECT "Name"
FROM art_gallery."User" t1
Join art_gallery."Artist" t2 on t1. "UserID" = t2. "UserID"
```

Data output | Messages | Notifications

	Name character varying
1	Barney
2	Kassandra
3	Alan
4	Alessia
5	Phillip
6	Abbey
7	Chuck
8	Leanne
9	Sydney
10	Mark

Total rows: 50 of 50 | Query complete 00:00:00.091

14. Count the number of art which shows in the exhibition which is held in Chicago.

```
SELECT COUNT("ArtID")
FROM art_gallery."Artwork" t1
Join  art_gallery."Exhibition" t2 on t1."ExhibitionID" =
t2."ExhibitionID"
WHERE t2."place" = 'Chicago'
```

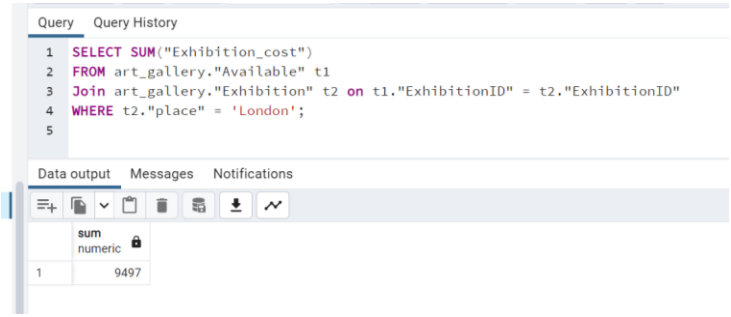
15. Display the artist names who give the comment on art number 3.

```
SELECT  "Name"
FROM art_gallery."User" t1
Join  art_gallery."Comment" t2 on t2."UserID" = t1."UserID"
WHERE t2. "ArtID" = '3'
```

16. Find the total money spent for the exhibition which is held in ‘London’.

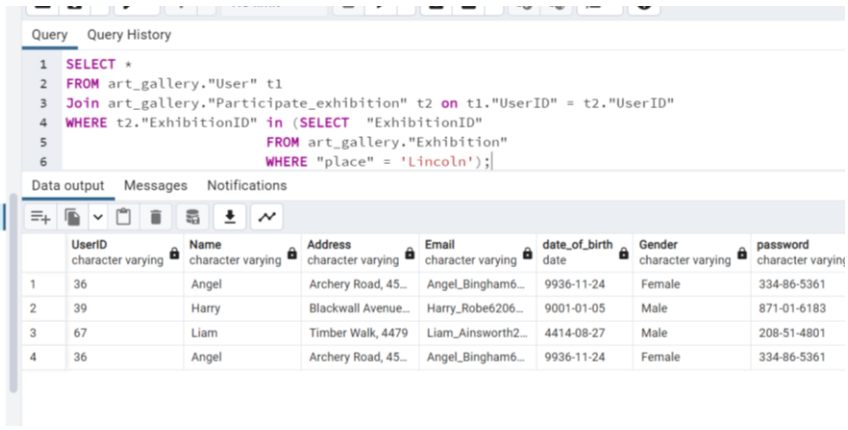
```
SELECT SUM("Exhibition_cost")
```

```
FROM art_gallery."Available" t1
Join art_gallery."Exhibition" t2 on t1."ExhibitionID" =
t2."ExhibitionID"
WHERE t2."place" = 'London';
```



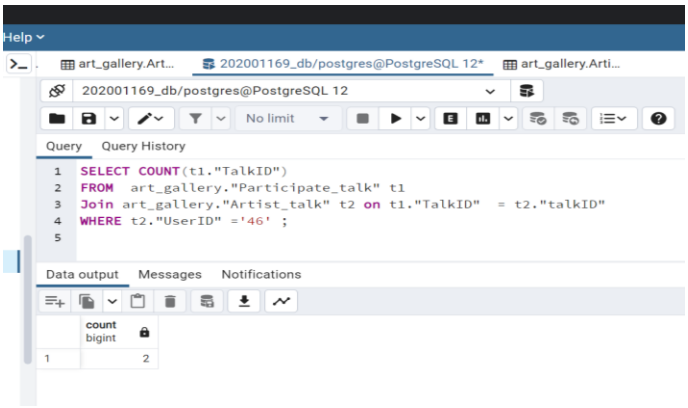
17. Display the details of users who take part in the exhibition which is held in Lincoln.

```
SELECT *
FROM art_gallery."User" t1
Join art_gallery."Participate_exhibition" t2 on t1."UserID" =
t2."UserID"
WHERE t2."ExhibitionID" in (SELECT  "ExhibitionID"
                            FROM art_gallery."Exhibition"
                            WHERE "place" = 'Lincoln');
```



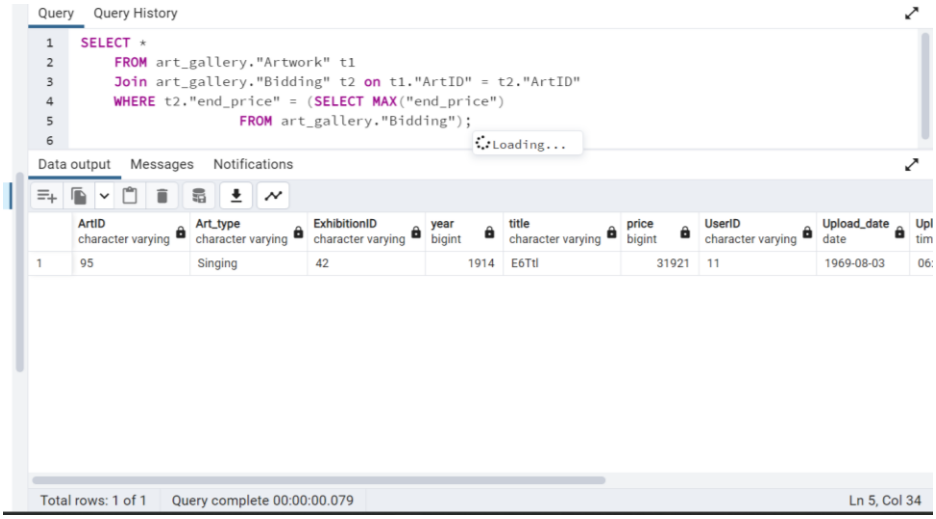
18. Count the number of users who take part in lectures given by artist xyz on topic music.

```
SELECT COUNT(t1."TalkID")
FROM  art_gallery."Participate_talk" t1
Join art_gallery."Artist_talk" t2 on t1."TalkID"  = t2."talkID"
WHERE t2."UserID" ='46' ;
```



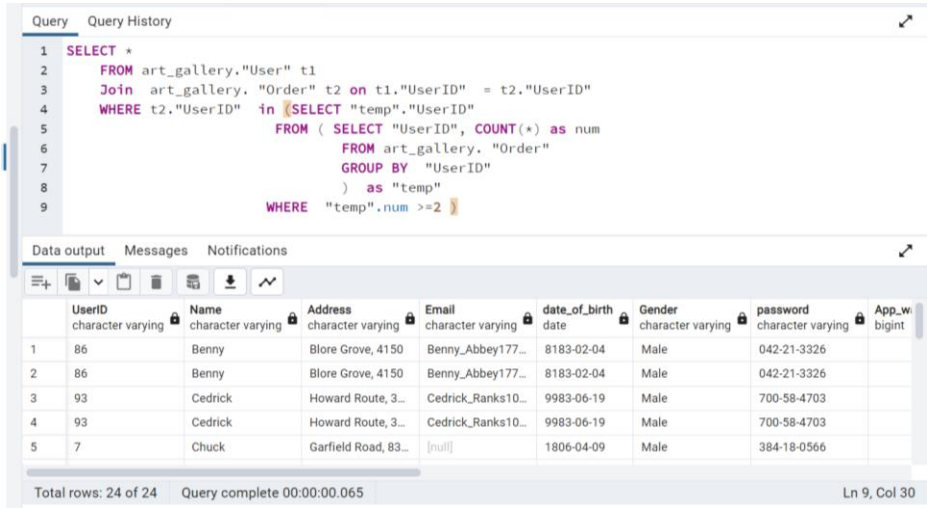
19. Display the detail of art which bidding last price is highest.

```
SELECT *
FROM art_gallery."Artwork" t1
Join art_gallery."Bidding" t2 on t1."ArtID" = t2."ArtID"
WHERE t2."end_price" = (SELECT MAX("end_price")
                        FROM art_gallery."Bidding");
```



20. Provide the details of users who buy 10 or more artwork.

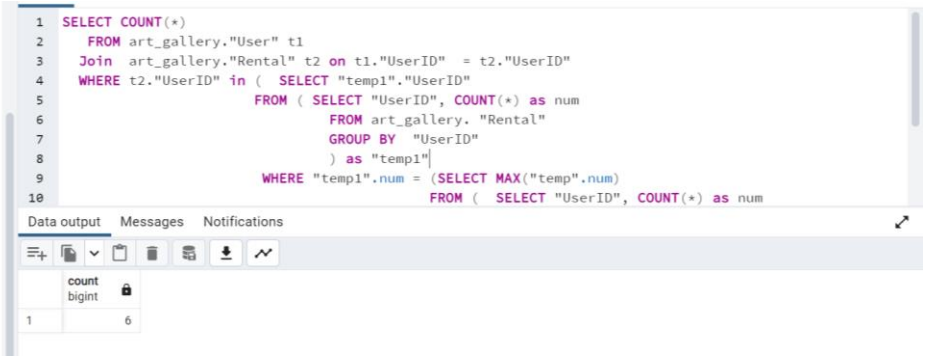
```
SELECT *
FROM art_gallery."User" t1
Join  art_gallery. "Order" t2 on t1."UserID"  = t2."UserID"
WHERE t2."UserID"  in (SELECT "temp"."UserID"
            FROM ( SELECT "UserID", COUNT(*) as num
            FROM art_gallery. "Order"
            GROUP BY  "UserID")        as "temp"
            WHERE  "temp".num >=2 )
```



21. Count the number of users who take the highest number of art as a Rental.

```
SELECT COUNT(*)
 FROM art_gallery."User" t1
Join  art_gallery."Rental" t2 on t1."UserID"  = t2."UserID"
WHERE t2."UserID" in (  SELECT "temp1"."UserID"
                FROM ( SELECT "UserID", COUNT(*) as num
                FROM art_gallery. "Rental"
                GROUP BY  "UserID") as "temp1"
                WHERE "temp1".num = (SELECT
MAX("temp".num)
FROM ( SELECT "UserID", COUNT(*) as num
```

FROM art_gallery. "Rental"
GROUP BY "UserID") as "temp"));



22. Trigger

- **Create Trigger:**

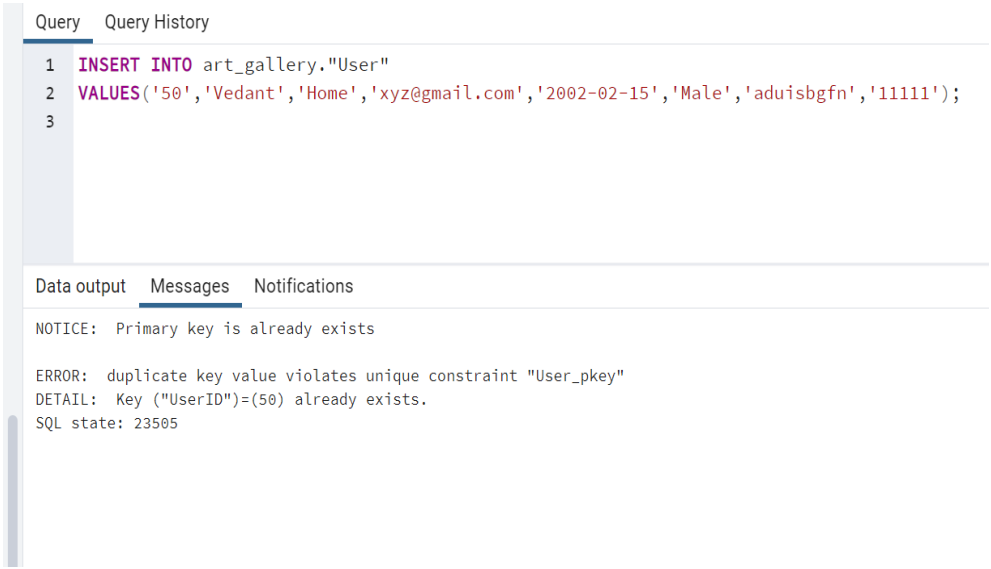
CREATE TRIGGER "Primary_key_check"
BEFORE INSERT
ON art_gallery."User"
FOR EACH ROW
EXECUTE PROCEDURE art_gallery."PrimaryKeyChecker"();

- **Trigger Function:**

DECLARE
KeyCount integer;
BEGIN
IF NEW."UserID" IS NULL THEN
RAISE NOTICE 'Primary key cannot be NULL';
END IF;
SELECT COUNT(*)
INTO KeyCount
FROM art_gallery."User"
WHERE "UserID" = NEW."UserID";
IF KeyCount >= 1 THEN
RAISE NOTICE 'Primary key is already exists';

END IF;
RETURN NEW;
END;

Insert New Tuple:
INSERT INTO art_gallery."User"
VALUES('50','Vedant','Home','xyz@gmail.com','02-15-2002','Male','aduisbgfn','11111');



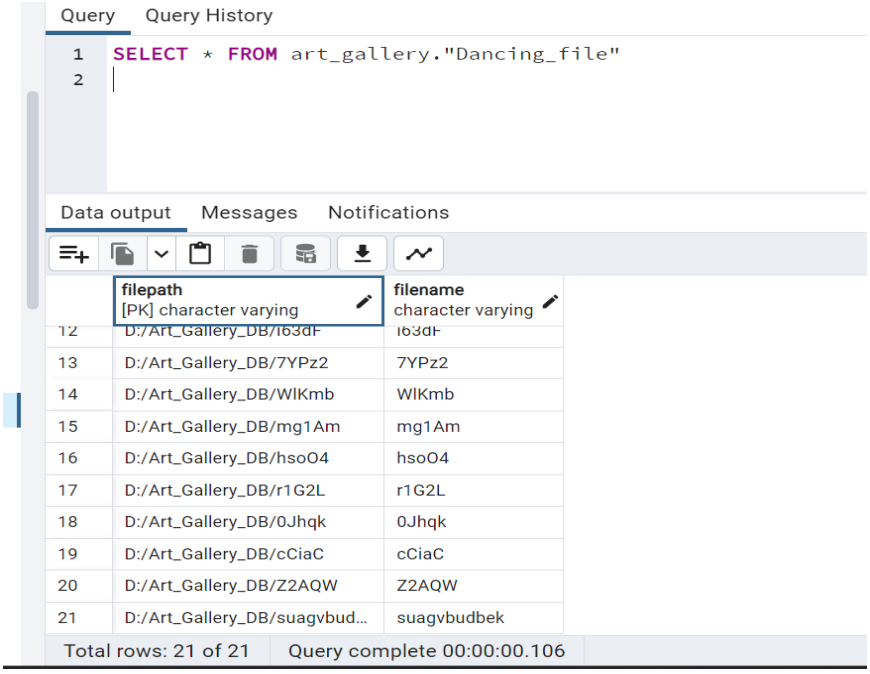
23. Procedure

- **Create Procedure:**
  CREATE OR REPLACE PROCEDURE art_gallery.procedure2(in fpath character varying,in fname character varying)
  LANGUAGE SQL
  as $$
  INSERT INTO art_gallery."Dancing_file" VALUES (fpath,fname);
  $$;

- **Call Procedure:**

Call art_gallery.procedure2('D:/Art_Gallery_DB/suagvbudbek','suagvbudbek');

After calling procedure2 Dancing_file.



- **Front-end Development**

Frontend code

1.index.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="CSS/utils.css">
```

```
    <link rel="stylesheet" href="CSS/style.css">
    <link rel="stylesheet" href="CSS/mobile.css">
    <title>Document</title>
    <style>
        a:hover {
            color: yellow;
        }
    </style>
</head>

<body>
    <nav class="navigation max-width-1 margin-auto">
        <div class="nav-left">
            <b>Art gallery app</b>
            <ul>
                <li><a href="/">Home</a></li>
                <!-- <li><a href="/query1">query</a></li> -->
                <li><a href="/signin">SignIn</a></li>
                <li><a href="/Pricefilter">Price Filter</a></li>
                <li><a href="/exhibition">Exhibition</a></li>
            </ul>
        </div>


        <div class="nav-right ">
            <form method="post" action="/searchArt">

                <input type="text" class="form-input" id="textarea"
name="title" placeholder="Search Art">
                <button class="button-71">Search</button>
            </form>


        </div>
    </nav>

    <div class="max-width-1 margin-auto">
        <hr>
    </div>
```

```
    <div class="content max-width-1 margin-auto my-2">
        <div class="content-left">
            <h2>Art Gallery</h2>
            <p>Welcome to our online art gallery.One place for all Types
of arts. where you scan see arts and also buy the arts </p>
        </div>

        <div class="content-right">

            <img src="IMG/Arts.jpg" alt="">
        </div>

    </div>

    <div class="max-width-1">
        <hr>
    </div>

    <div class="home-articles max-width-1 margin-auto font2">

        <h2>Featured Arts</h2>
        <div class="home-article">

            <div class="home-article-img">
                <img src="https://artlogic-
res.cloudinary.com/w_750,c_limit,f_auto,fl_lossy,q_auto/artlogicstorage/el
izabethmossgalleries/images/view/01d4159250fe34948862464a5c55fd9fj/elizabe
thmossgalleries-lynne-drexler-celestial-division-1966.jpg" alt="">
            </div>

            <div class="home-article-content font2">
                <a href="/blogpost.html">
                    <h3>Painting Art</h3>
                </a>
                <div>Author nmae : John Weasley</div>
                <span>Date : 24/10/2021</span><br>
                <span>required time : 1 day</span>

            </div>
        </div>
```

```
        <div class="home-article">

            <div class="home-article-img">
                <img src="https://artlogic-
res.cloudinary.com/w_750,c_limit,f_auto,fl_lossy,q_auto/ws-
artlogicwebsite0033/usr/images/exhibitions/main_image_override/65/peterson
_3-1-.jpg" alt="">
            </div>

            <div class="home-article-content font2">
                <a href="/blogpost.html">
                    <h3>Nature</h3>
                </a>
                <div>Author nmae : John Weasley</div>
                <span>Date : 20/7/2018</span><br>
                <span>required time : 3 days</span>

            </div>
        </div>

    </div>

    <div class="footer max-width-2 margin-auto">


        <svg width="30" height="30" viewBox="0 0 30 30" fill="none"
class="mf mg"><path fill-rule="evenodd"
            clip-rule="evenodd" d="M15 27a12 12 0 1 0 0-24 12 12 0 0 0 0 0
24zm4.95-16.17a2.67 2.67 0 0
            0-4.6 1.84c0 .2.03.41.05.62a7.6 7.6 0 0 1-5.49-2.82 3 3 0 0 0-
.38 1.34c.02.94.49 1.76 1.2
            2.23a2.53 2.53 0 0 1-1.2-.33v.04c0 1.28.92 2.36 2.14 2.62-
.23.05-.46.08-.71.11-.21-.02-.27
            -.03a2.68 2.68 0 0 0 2.48 1.86A5.64 5.64 0 0 1 9 19.38a7.62
7.62 0 0 0 4.1 1.19c4.9 0 7.58
            -4.07 7.57-7.58v-.39c.52-.36.97-.83 1.33-1.38-.48.23-1 .37-
1.53.43.56-.33.96-.86 1.15-1.48-
            .5.31-1.07.53-1.67.66z" fill="#292929"></path></svg>
```

```
        <svg width="30" height="30" viewBox="0 0 30 30" fill="none"
class="mf mg">
          <path fill-rule="evenodd" clip-rule="evenodd" d="M15 27a12 12 0
1 0 0-24
          12 12 0 0 0 0 24zm-1.23-6.03V15.6H12v-2.15h1.77v-1.6C13.77 10
14.85 9 16.42
          9c.75 0 1.4.06 1.58.08v1.93h-1.09c-.85 0-1.02.43-1.02
1.05v1.38h2.04l-.27
           2.15H15.9V21l-2.13-.03z" fill="#292929"></path></svg>

        <svg width="30" height="30" viewBox="0 0 30 30" fill="none"
class="mf mg">
           <path fill-rule="evenodd" clip-rule="evenodd" d="M27 15a12 12
0 1
           1-24 0 12 12 0 0 1 24 0zm-14.61 5v-7.42h-2.26V20h2.26zm-1.13-
8.44c.79 0 1.28-.57
           1.28-1.28-.02-.73-.5-1.28-1.26-1.28-.78 0-1.28.55-1.28 1.28 0
.71.49 1.28 1.25
           1.28h.01zM15.88 20h-2.5s.04-6.5 0-7.17h2.5v1.02l-.02.02h.02v-
.02a2.5 2.5 0 0 1
            2.25-1.18c1.64 0 2.87 1.02 2.87 3.22V20h-2.5v-3.83c0-.97-.36-
1.62-1.26-1.62-.69
             0-1.1.44-1.28.87-.06.15-.08.36-.08.58v4z"
fill="#292929"></path></svg>
    </div>
    <div class="social">

    </div>
</body>

</html>
```

2.exhibition.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="CSS/utils.css">
    <link rel="stylesheet" href="CSS/style.css">
    <link rel="stylesheet" href="CSS/contact.css">
    <link rel="stylesheet" href="CSS/mobile.css">
    <title>Document</title>
</head>

<script>
    function myfunc() {
        document.getElementById('frm').submit();
    }
</script>


<body>
    <nav class="navigation max-width-1 margin-auto">
        <div class="nav-left">
            <b>Exhibition</b>
            <ul>
                <li><a href="/">Home</a></li>
                <li><a href="/EXplace">place</a></li>
                <li><a href="/Signin">Signin</a></li>
                <li><a href="#">about</a></li>
            </ul>
        </div>


        <div class="nav-right ">
            <form method="post" action="/searchArt">
                <input type="text" class="form-input" id="textarea"
name="title" placeholder="Search Art">
                <button class="button-71">Search</button>
            </form>
        </div>
    </nav>

    <div class="max-width-1 margin-auto">
        <hr>
    </div>
```

```
    <div class="contact-content">

        <form method="post" action="/insertEx">
            <div class="home-articles max-width-1 margin-auto font2">
                <h2>Enter Exhibition Information</h2>
                <div class="contact-form">
                    <div class="form-box">
                        <input type="text" name="place" placeholder="Enter
Place">
                    </div>

                    <div class="form-box">
                        <input type="text" name="SD" placeholder="Enter
Start Date">
                    </div>

                    <div class="form-box">
                        <input type="text" name="ED" placeholder="Enter
End Date">
                    </div>

                    <div class="form-box">
                        <button class="button-71"
type="submit">Submit</button>
                    </div>
                </div>

        </form>

        </div>

        <div class="contact-content">

            <form method="post" action="/ArtExPlace">
                <div class="home-articles max-width-1 margin-auto font2">
                    <h2 style="text-align: center">Find Arts at particular
Exhibition</h2>
                    <h2> Enter Exhibition Place Information</h2>
                    <div class="contact-form">
```

```
                        <div class="form-box">
                            <input type="text" name="place"
placeholder="Enter  Place">
                        </div>

                        <div class="form-box">
                            <button class="button-71"
type="submit">Submit</button>
                        </div>
                    </div>

            </form>

            </div>
</body>


</html>
```

3.signin.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="CSS/utils.css">
    <link rel="stylesheet" href="CSS/style.css">
    <link rel="stylesheet" href="CSS/contact.css">
    <link rel="stylesheet" href="CSS/mobile.css">
    <title>Document</title>
</head>

<script>
    function myfunc() {
        document.getElementById('frm').submit();
    }
</script>
```

```
<body>
    <nav class="navigation max-width-1 margin-auto">
        <div class="nav-left">
            <b>Sign In</b>
            <ul>
                <li><a href="/">Home</a></li>
                <li><a href="/Signin">Signin</a></li>
                <li><a href="#">about</a></li>
            </ul>
        </div>


        <div class="nav-right ">
            <form method="post" action="/searchArt">

                <input type="text" class="form-input" id="textarea"
name="title" placeholder="Search Art">
                <button class="button-71">Search</button>
            </form>


        </div>
    </nav>

    <div class="max-width-1 margin-auto">
        <hr>
    </div>

    <div class="contact-content">
        <div class="home-articles max-width-1 margin-auto font2">
            <h2>Enter your Details here</h2>

            <form method="post" id="frm" action="/insertUser">
                <div class="contact-form">

                    <div class="form-box">
                        <input type="text" name="name" id="name"
placeholder="Enter your name">
                    </div>
```

```
                    <div class="form-box">
                        <input type="text" name="address"
placeholder="Enter your address">
                    </div>

                    <div class="form-box">
                        <input type="text" name="email" placeholder="Enter
your email">
                    </div>

                    <div class="form-box">
                        <input type="date" name="DOB" placeholder="Enter
your Date of Birth">
                    </div>

                    <div class="form-box">
                        <input type="text" name="gender"
placeholder="Enter your gender">
                    </div>

                    <div class="form-box">
                        <input type="text" name="password"
placeholder="Enter your password">
                    </div>

                    <div class="form-box">
                        <button class="button-71" onclick="myfunc();">
Submit </button>
                    </div>

                </div>
            </form>

        </div>
    </div>


    </div>
```

```
</body>



</html>
```

4.showArtDetails.ejs

```
<link href="/CSS/table.css" rel="stylesheet" />
<div class="para">Art details</div>
<% var Length = detail.length%>
    <% if(Length!=0) { %>
        <table>
            <thead>
                <tr class="firstt">
                    <th scope="col" class="firstt">Art Title</th>
                    <th scope="col" class="firstt">Art_Type</th>
                    <th scope="col" class="firstt">Year</th>
                    <th scope="col" class="firstt">Price</th>
                </tr>
            </thead>

            <tbody class="firstt">
                <tr class="firstt">
                    <td class="firstt">
                        <%= detail[0].title %>
                    </td>
                    <td class="firstt">
                        <%= detail[0].Art_type %>
                    </td>
                    <td class="firstt">
                        <%= detail[0].year %>
                    </td>
                    <td class="firstt">
                        <%= detail[0].price %>
                    </td>
                </tr>
                <br>
            </tbody>
        </table>
```

5.PriceFIlter.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="CSS/utils.css">
    <link rel="stylesheet" href="CSS/style.css">
    <link rel="stylesheet" href="CSS/contact.css">
    <link rel="stylesheet" href="CSS/mobile.css">
    <title>Document</title>
</head>


<body>
    <nav class="navigation max-width-1 margin-auto">
        <div class="nav-left">
            <b>Price Filter</b>
            <ul>
                <li><a href="/">Home</a></li>
            </ul>
        </div>
    </nav>

    <div class="max-width-1 margin-auto">
        <hr>
    </div>

    <div class="contact-content">

        <form method="post" action="/ArtFilter">
            <div class="home-articles max-width-1 margin-auto font2">
                <h2>Enter Price Range</h2>

                <div class="contact-form">
                    <div class="form-box">
                        <input type="text" name="SP" placeholder="Start
price">
```

```
                    </div>

                    <div class="form-box">
                        <input type="text" name="EP" placeholder="End
price">
                    </div>

                    <div class="form-box">
                        <button class="button-71"
type="submit">Submit</button>
                    </div>
                </div>

        </form>

        </div>

</body>


</html>
```

6.placeshow.ejs

```
<!-- <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css" rel="stylesheet" /> -->
<!-- <link href="https://getbootstrap.com/docs/5.2/assets/css/docs.css"
rel="stylesheet" /> -->
<link href="/CSS/table.css" rel="stylesheet" />
<div class="para">Exhibition Available at this Places:</div>
<% var Length = places.length%>

    <table id="tb">
        <thead>
            <tr class="firstt">
                <th scope="col" class="firstt">Art Title</th>
            </tr>
        </thead>
        <% for(var i=0; i<Length; i++){ %>
```

```
            <tbody class="firstt">
                <tr class="firstt">
                    <td class="firstt">
                        <%= places[i].place %>
                    </td>
                </tr>
                <br>
            </tbody>
            <% } %>
    </table>
```

7.FilteredArt.ejs

```
<!-- <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css" rel="stylesheet" /> -->
<!-- <link href="https://getbootstrap.com/docs/5.2/assets/css/docs.css"
rel="stylesheet" /> -->
<link href="/CSS/table.css" rel="stylesheet" />
<div class="para">Arts list for the selected range of price is :</div>
<% var Length = Art.length%>
    <table>
        <thead>
            <tr class="firstt">
                <th scope="col" class="firstt">Art Title</th>
                <th scope="col" class="firstt">Price</th>
            </tr>
        </thead>
        <% for(var i=0; i<Length; i++){ %>

            <tbody class="firstt">
                <tr class="firstt">
                    <td class="firstt">
                        <%= Art[i].title %>
                    </td>
                    <td class="firstt">
                        <%= Art[i].price %>
                    </td>
                </tr>
```

```
                <br>
            </tbody>
            <% } %>
    </table>
```

8.FilteredArt1.ejs

```
<!-- <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css" rel="stylesheet" /> -->
<!-- <link href="https://getbootstrap.com/docs/5.2/assets/css/docs.css"
rel="stylesheet" /> -->
<link href="/CSS/table.css" rel="stylesheet" />
<div class="para">Arts list for entered place</div>
<% var Length = Art.length%>
    <table>
        <thead>
            <tr class="firstt">
                <th scope="col" class="firstt">Art Title</th>
            </tr>
        </thead>
        <% for(var i=0; i<Length; i++){ %>

            <tbody class="firstt">
                <tr class="firstt">
                    <td class="firstt">
                        <%= Art[i].title %>
                    </td>
                </tr>
                <br>
            </tbody>
            <% } %>
    </table>
```

CSS

1. contact.css

```
.contact-content {
    padding: 10px;
```

```
    height: 100vh;
    overflow: hidden;
    /* background-color: rgb(138, 138, 255); */
}

.contact-content::after {
    content: "";
    background-image: url('IMG/Arts.jpg');
    position: absolute;
    width: 100%;
    height: inherit;
    opacity: 0.15;
    z-index: -1;
}

.form-box input,
textarea {
    width: 50vw;
    padding: 10px 7px;
    margin: 7px 0px;
    font-family: var(--font2);
    border: 2px solid rgb(12, 59, 248);
    border-radius: 5px;
}

.contact-form {}
```

2. mobile.css

```
@media screen and (max-width: 800px) {
    .navigation {
        flex-direction: column;
        margin-bottom: 23px;
    }
    .nav-left {
        text-align: center;
        flex-direction: column;
    }
    .content-right {
```

```
        display: none;
    }
    .home-article {
        flex-direction: column;
    }
    .home-article-img {
        text-align: center;
    }
    .form-input {
        width: 50%;
    }
    .form-box input,
    textarea {
        width: 66vw;
    }
    .social {
        padding: 0;
    }
    .post-img {
        height: auto;
    }
}
```

3. style.css

```
* {
    margin: 0;
    padding: 0;
}

body {
    background-color: rgb(143, 209, 221);
}

a:hover {
    color: yellow;
}

.navigation {
```

```
    font-family: var(--font2);
    height: 74px;
    /* background-color: aqua; */
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: rgb(79, 174, 230);
    padding-left: 120px;
    padding-right: 120px;
    border: solid 3px white;
    border-radius: 30px;
    margin-bottom: 15px;
    /* position: fixed; */
    z-index: 3;
    margin-top: 5px;
}

.nav-left {
    font-size: 20px;
    display: flex;
}

.nav-left ul {
    display: flex;
    align-items: center;
    margin: 0 50px;
    font-size: 15px;
}

.nav-left ul li {
    list-style: none;
    margin: 0 7px;
    font-family: var(--font1);
}

.nav-left ul li a {
    text-decoration: none;
    color: black;
}
```

```css
.nav-left ul li a:hover {
    color: rgb(250, 252, 250);
    text-decoration: underline;
}

.content {
    padding: 10px;
    display: flex;
    height: 100%;
    position: relative;
    /* background-color: rgb(138, 138, 255); */
}

.content::after {
    content: "";
    background-image: url('../Img/Home.jpg');
    position: absolute;
    width: 100%;
    height: inherit;
    opacity: 0.15;
}

.content-left {
    padding: 30px;
    font-family: var(--font2);
    /* width: 50%; */
    display: flex;
    justify-content: center;
    flex-direction: column;
    z-index: 1;
}

.content-left h2 {
    font-family: var(--font1);
}

.content-right {
    /* width: 50%; */
    display: flex;
    align-items: center;
```

```
    justify-content: center;
}

.content-right img {
    height: 300px;
    border: 2px solid black;
    border-radius: 50px;
}

.home-articles {
    padding: 20px;
    margin-top: 17px;
    /* height: 500px; */
    background-color: var(--main-bg-color);
}

.home-articles h2 {
    font-family: var(--font1);
}

.home-article {
    display: flex;
    padding: 0px;
}

.home-article-content {
    align-self: center;
    padding: 25px;
}

.home-article-content a {
    text-decoration: none;
    color: black;
}

.home-article img {
    width: 297px;
    padding: 20px;
}
```

```css
.footer {
    height: 50px;
    display: flex;
    align-items: center;
    justify-content: center;
    background-color: rgb(43, 159, 226);
    width: 60vw;
    flex-direction: column;
}

.link {
    align-items: center;
}

.social {
    display: flex;
    align-items: center;
}


/* CSS */

.button-71 {
    background-color: #0957c5;
    border: 0;
    border-radius: 56px;
    color: #fff;
    cursor: pointer;
    display: inline-block;
    font-family: system-ui, -apple-system, system-ui, "Segoe UI",
Roboto, Ubuntu, "Helvetica Neue", sans-serif;
    font-size: 13px;
    font-weight: 600;
    outline: 0;
    padding: 0px 0px;
    position: relative;
    text-align: center;
    text-decoration: none;
    transition: all .3s;
    user-select: none;
```

```
    -webkit-user-select: none;
    touch-action: manipulation;
}

.button-71:before {
    background-color: initial;
    background-image: linear-gradient(#fff 0, rgba(255, 255, 255, 0)
100%);
    border-radius: 125px;
    content: "";
    height: 50%;
    left: 4%;
    opacity: .5;
    position: absolute;
    top: 0;
    transition: all .3s;
    width: 92%;
}

.button-71:hover {
    box-shadow: rgba(255, 255, 255, .2) 0 3px 15px inset, rgba(0, 0,
0, .1) 0 3px 5px, rgba(0, 0, 0, .1) 0 10px 13px;
    transform: scale(1.05);
}

@media (min-width: 768px) {
    .button-71 {
        padding: 16px 48px;
    }
}

#textarea {
    background-color: rgb(231, 235, 229);
    color: rgb(100, 100, 100);
    -webkit-border-radius: 8.5px;
    -moz-border-radius: 8.5px;
    border-radius: 8.5px;
    width: 220px;
    height: 30px;
}
```

4. utils.css

```
@import
url('https://fonts.googleapis.com/css2?family=Mochiy+Pop+P+One&family=Roboto:wght@300&display=swap');


:root{
   --main-bg-color : rgb(92, 182, 241);
   --font1: 'Mochiy Pop P One', sans-serif;
   --font2: 'Roboto', sans-serif;

}

.font1{
    font-family: var(--font1);
}

.font2{
    font-family: var(--font2);
}

.max-width-1{
    max-width: 80vw;
    margin: auto;
}

.max-width-2{
    max-width: 60vm;
}

.margin-auto{
    margin: auto;
}

.my-2{
    margin-top: 30px;
    margin-bottom: 30px;
```

```
}

.btn{
    font-family: var(--font2);
    padding: 3px 10px;
    border :3px solid black;
    border-radius: 4px;
    font-size: 17px;
    cursor: pointer;
    transition: all 0.3s ease-in-out;
}

.btn:hover{
    color: black;
    background-color: var(--main-bg-color);
}

.form-input{
    padding: 3px 5px;
    font-size: 16px;
    border: 1px solid black;
    border-radius: 4px;
    margin: 0 12px;
    font-family: var(--font2);
}
```

5.table.css

```
body {
    background-color: rgb(143, 209, 221);
}

table {
    width: 40%;
    /* to put in centre table */
    margin: auto;
    font-size: 20px;
    border: 4px solid rgb(33, 33, 34);
```

```
    border-radius: 10px;
    /* spacing between border and letter  */
    padding: 5px;
    /* spacing between two borders of cells */
    border-spacing: 7px;
    /* table-layout: auto; */
}

th,
td {
    background-color: #96D4D4;
    padding: 5px;
    empty-cells: hide;
    text-align: center;
}

th {
    border: 5px solid rgb(5, 58, 231)!important;
}

.firstt {
    /* width: 25px; */
    border: solid rgb(224, 80, 40);
    border-width: 3px;
    border-radius: 10px;
}

.para {
    font-size: 50px;
    text-align: center;
    font-weight: 1000;
    color: rgb(23, 23, 239);
}
```

Backend:

1. index.js

```
var express = require("express");
var path = require("path");
```

```
var routes = require("./routes");
const client = require("./server/database");
const bodyParser = require("body-parser");

var app = express();

app.set("port", process.env.PORT || 3000);

//paths
app.set("views", path.join(__dirname, "views"));
app.use('/CSS', express.static(path.resolve(__dirname,
"views/CSS")))
app.use('/IMG', express.static(path.resolve(__dirname,
"views/IMG")))
app.use('/images', express.static(path.resolve(__dirname,
"views/images")))
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.set("view engine", "ejs");

app.use(routes);

app.listen(app.get("port"), function() {
    console.log("Server started on port " + app.get("port"));
})

client.connect();
app.get('/query1', (req, res) => {

    client.query('SELECT * FROM public.people', (err, result)
=> {
        console.log("query1");

        if (!err) {
            console.log(result.rows);
            res.send(result.rows);
        } else {
            console.log("query1 Err");
            console.log(err.message);
        }
```

```
    });
    console.log("query1 End");

    // res.render("exhibition.ejs");
})


var id = 0;
app.post('/insertUser', (req, res) => {
    const user = req.body;

    id++;
    let insertQuery = `insert into "User"
        values( '${4}', '${user.name}', '${user.address}',
'${user.email}' ,

'${user.DOB}','${user.gender}','${user.password}',${0} )`

    client.query(insertQuery, (err, result) => {

            if (!err) {
                res.send('Insertion was successful')
            } else { console.log(err.message) }
        })
        // res.render("index.ejs");
})

app.post('/insertEx', (req, res) => {
    const EX = req.body;
    console.log(req.body);
    console.log(EX.place);
    console.log(EX.ED);

    let insertQuery = `insert into "Exhibition"
        values( '${101}','${EX.SD}','${EX.ED}','${EX.place}'
)`

    client.query(insertQuery, (err, result) => {

            if (!err) {
```

```
                res.send('Insertion was successful')
            } else {
                console.log(err.message)
            }
        })
        // res.render("index.ejs");
})

app.get('/EXplace', (req, res) => {

    client.query('SELECT "Exhibition".place from
"Exhibition"', (err, result) => {
        if (!err) {
            console.log(result.rows);
            // res.send(result.rows);s
            res.render('placesShow.ejs', { places: result.rows
});
        }
    });
})

app.post('/searchArt', (req, res) => {
    const EX = req.body;
    console.log(req.body);


    client.query(`SELECT * from artwork where
"title"='${EX.title}'`, (err, result) => {
        if (!err) {
            console.log(result.rows);
            res.render('showArtDetails.ejs', { detail:
result.rows });
        }
    });
})

app.post('/ArtFilter', (req, res) => {

    const Filter = req.body;
```

```
    let insertQuery = `SELECT title,price from artwork WHERE
price >= '${Filter.SP}' AND price <= '${Filter.EP}' `
    client.query(insertQuery, (err, result) => {

        if (!err) {
            res.render('FilteredArt.ejs', { Art: result.rows
});
            // res.send(result.rows);
        } else { console.log(err.message) }
    });
})

app.post('/ArtExPlace', (req, res) => {

    const Loc = req.body;
    let insertQuery = `SELECT t1.title
    FROM art_gallery."Artwork" t1
    Join  art_gallery."Exhibition" t2 on t1."ExhibitionID" =
t2."ExhibitionID"
    WHERE t2."place" = '${Loc.place}' `


    client.query(insertQuery, (err, result) => {

        if (!err) {
            res.render('FilteredArt1.ejs', { Art: result.rows
});
        } else { console.log(err.message) }
    });
})
client.end;
```

2. routes.js

```
var express = require("express");
var router = express.Router();


router.get("/", function(req, res) {
    res.render("index.ejs");
});
```

```
router.get("/signin", function(req, res) {
    res.render("signin.ejs");
});

router.get("/exhibition", function(req, res) {
    res.render("exhibition.ejs");
});

router.get("/Pricefilter", function(req, res) {
    res.render("Pricefilter.ejs");
});
module.exports = router;
```

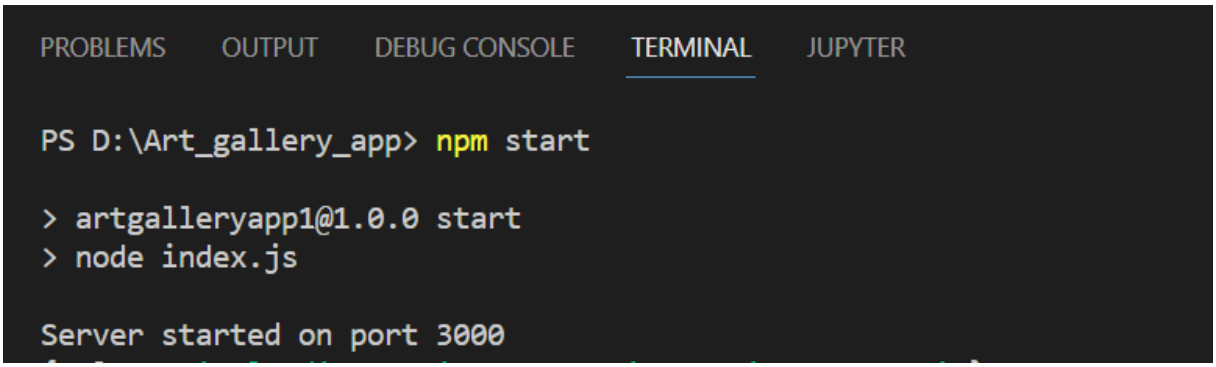
Connection to pgadmin:

1.database.js

```
const { Client } = require('pg');

const client = new Client({
    user: 'postgres',
    host: 'localhost',
    database: '202001169_db',
    password: 'harshal@003',
    port: 5432
})

module.exports = client
```

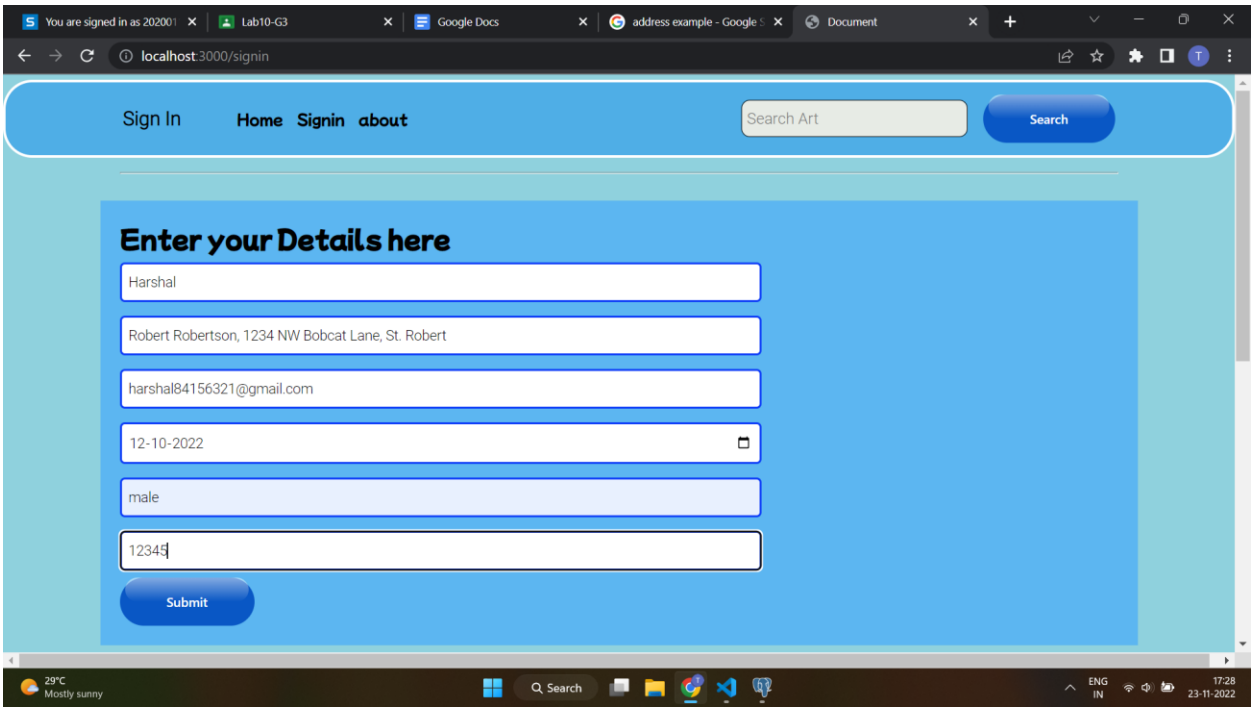
Web page Screenshots:

- **Starting server:**

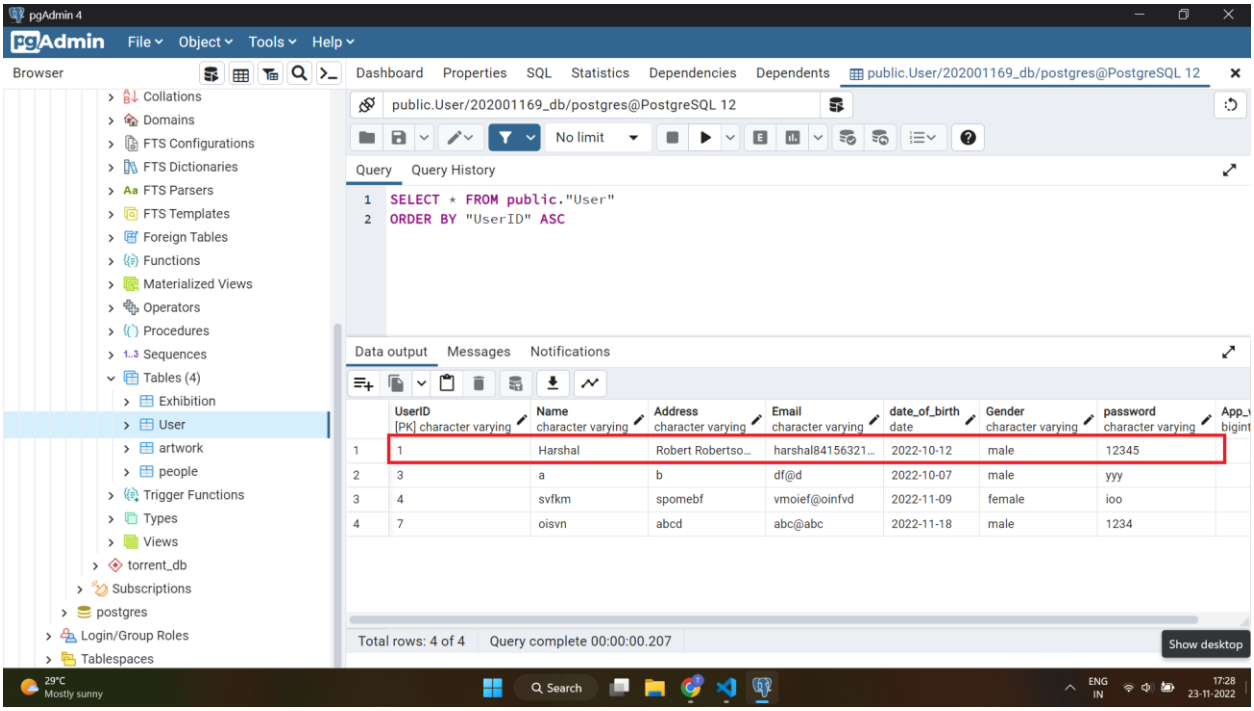


- **Insert data into tables**

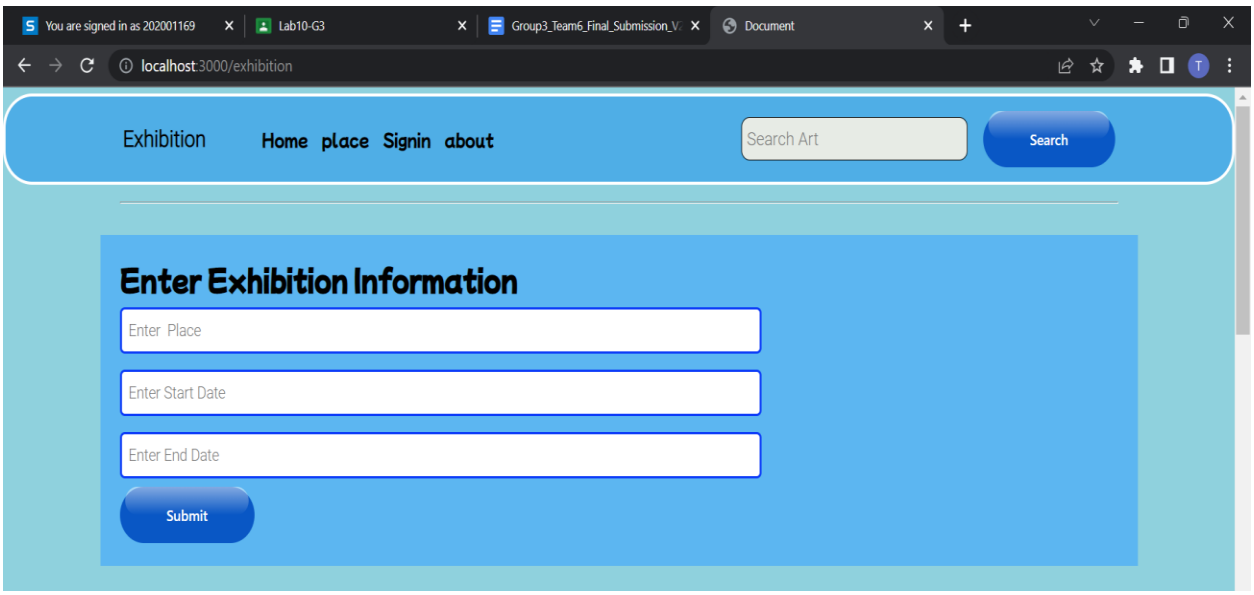
1.Sign in User

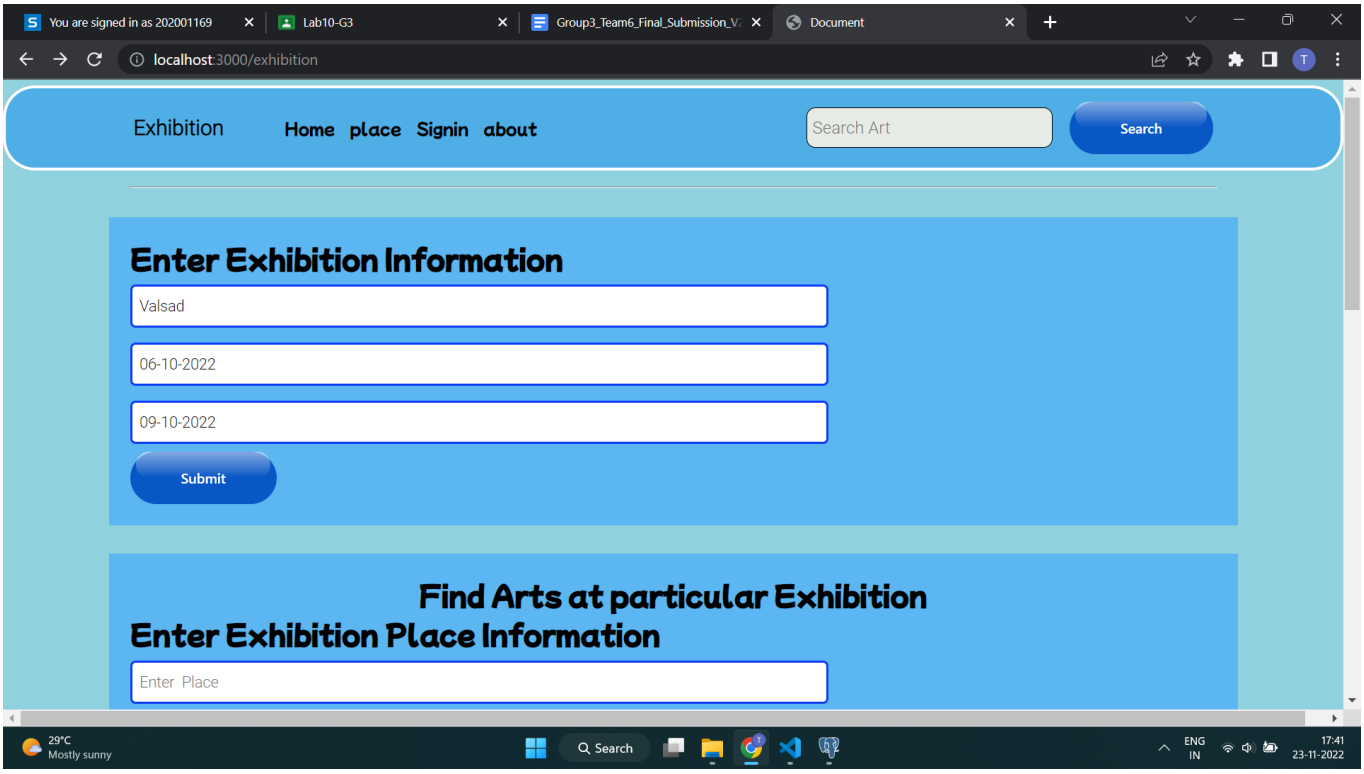


Change in User Table:

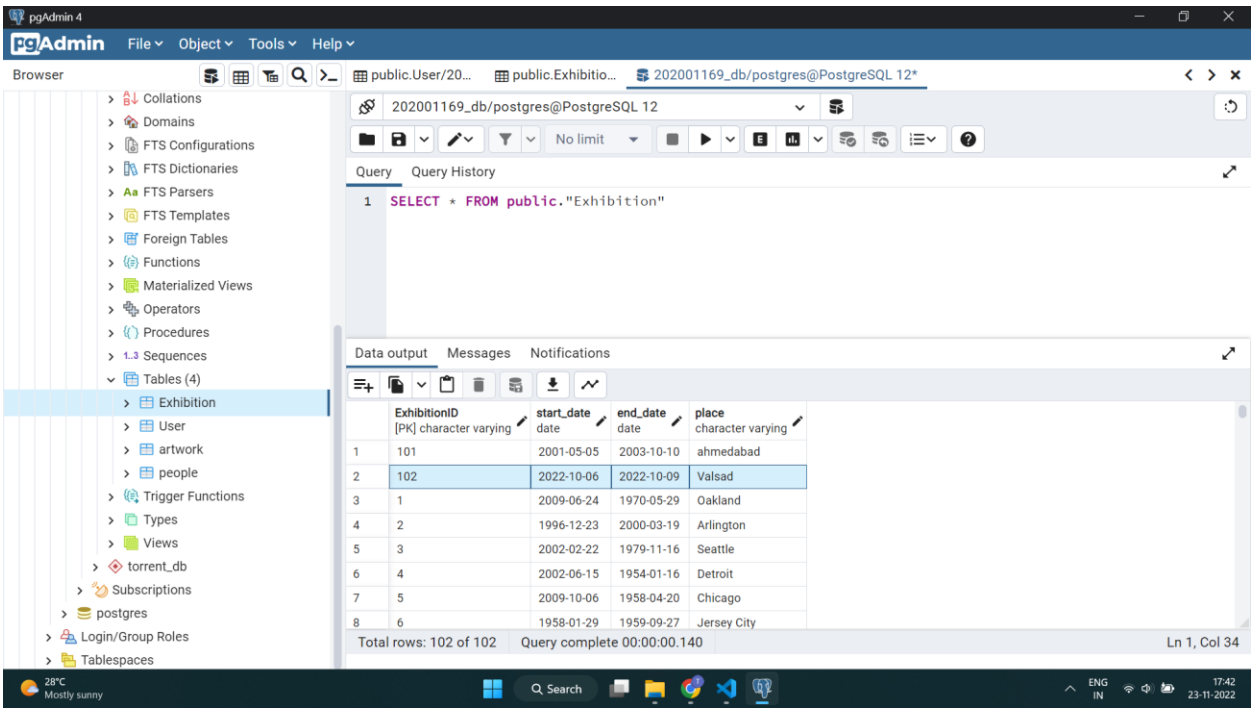


2. Add Exhibition



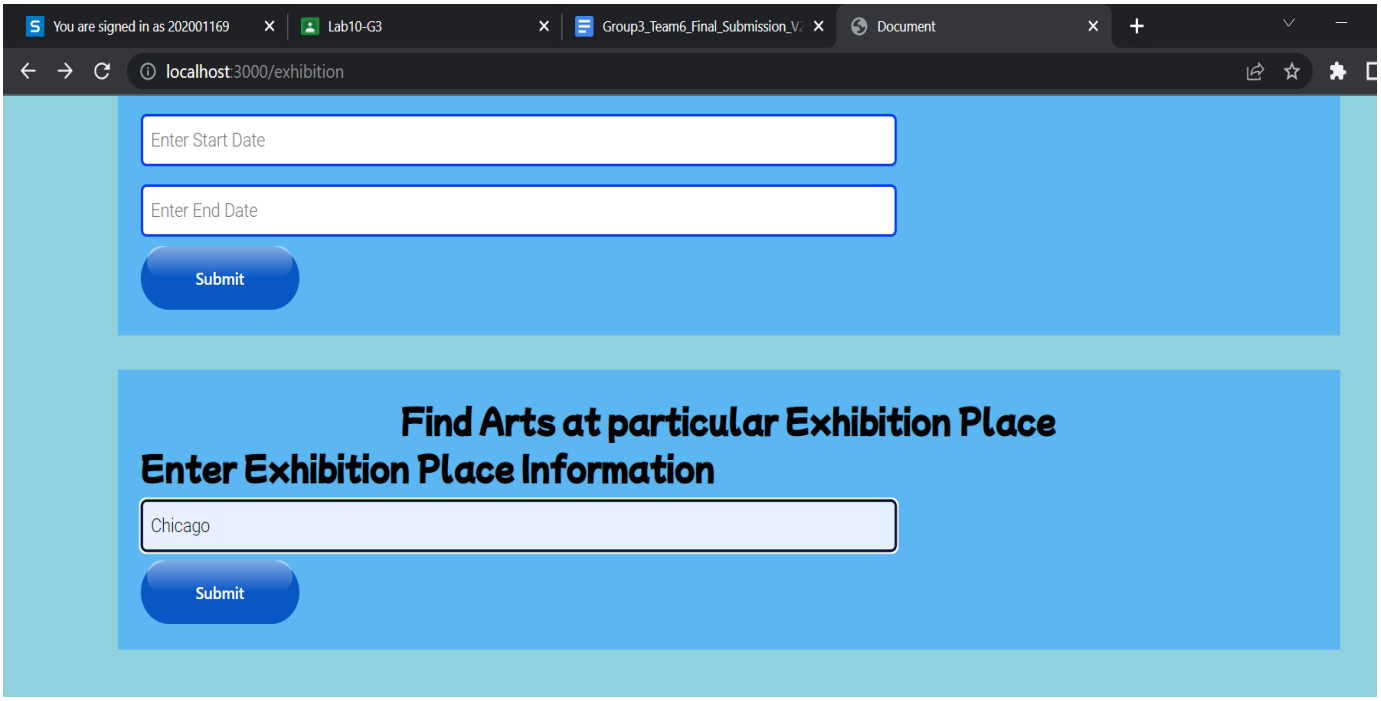


Change in Exhibition Table:

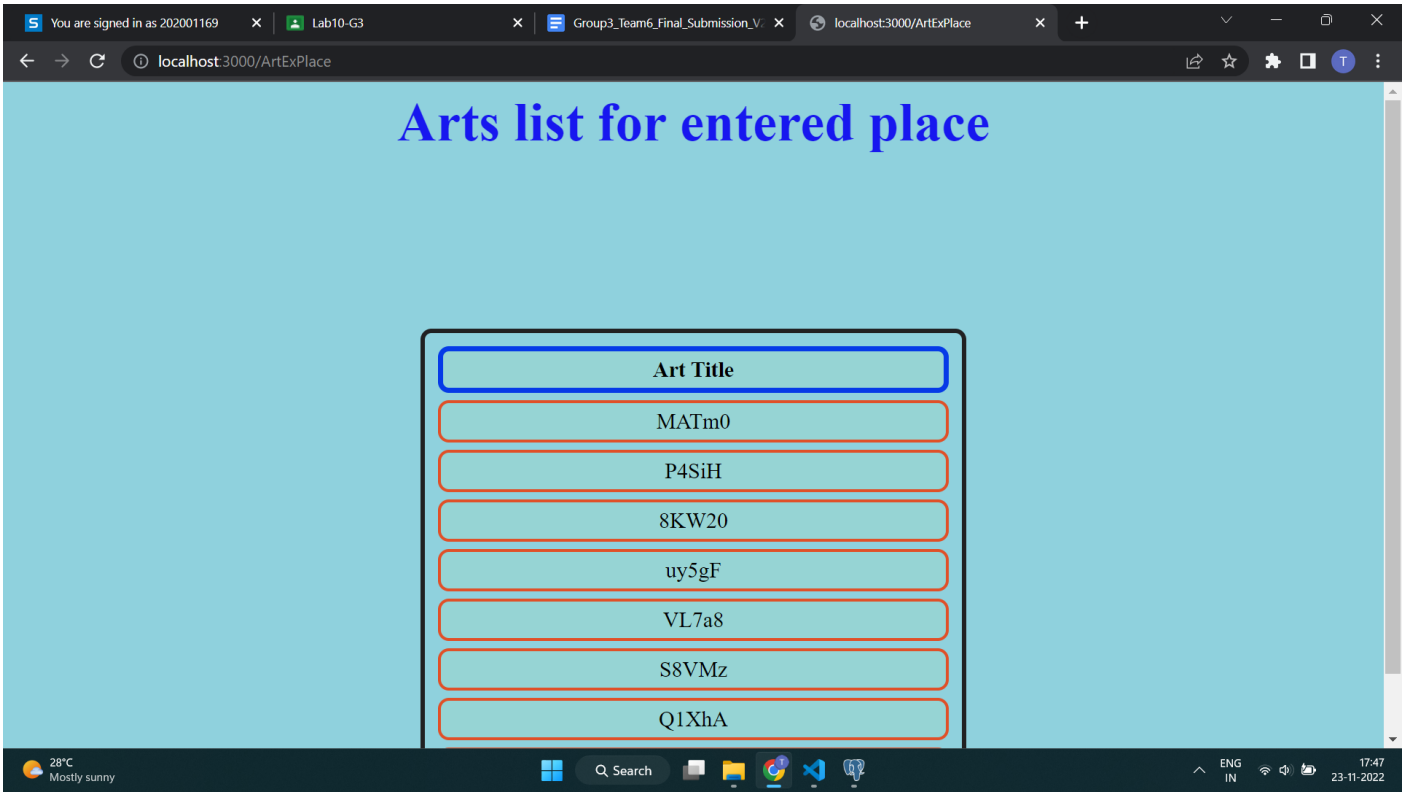


Queries:

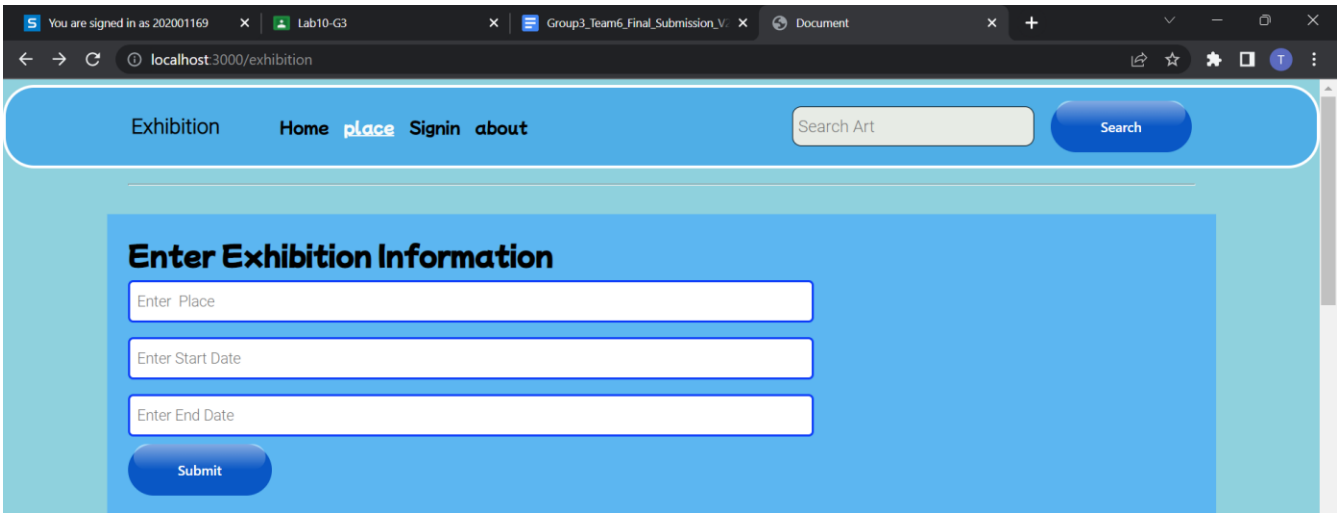
- **Find Arts at particular Exhibition place**



Result:



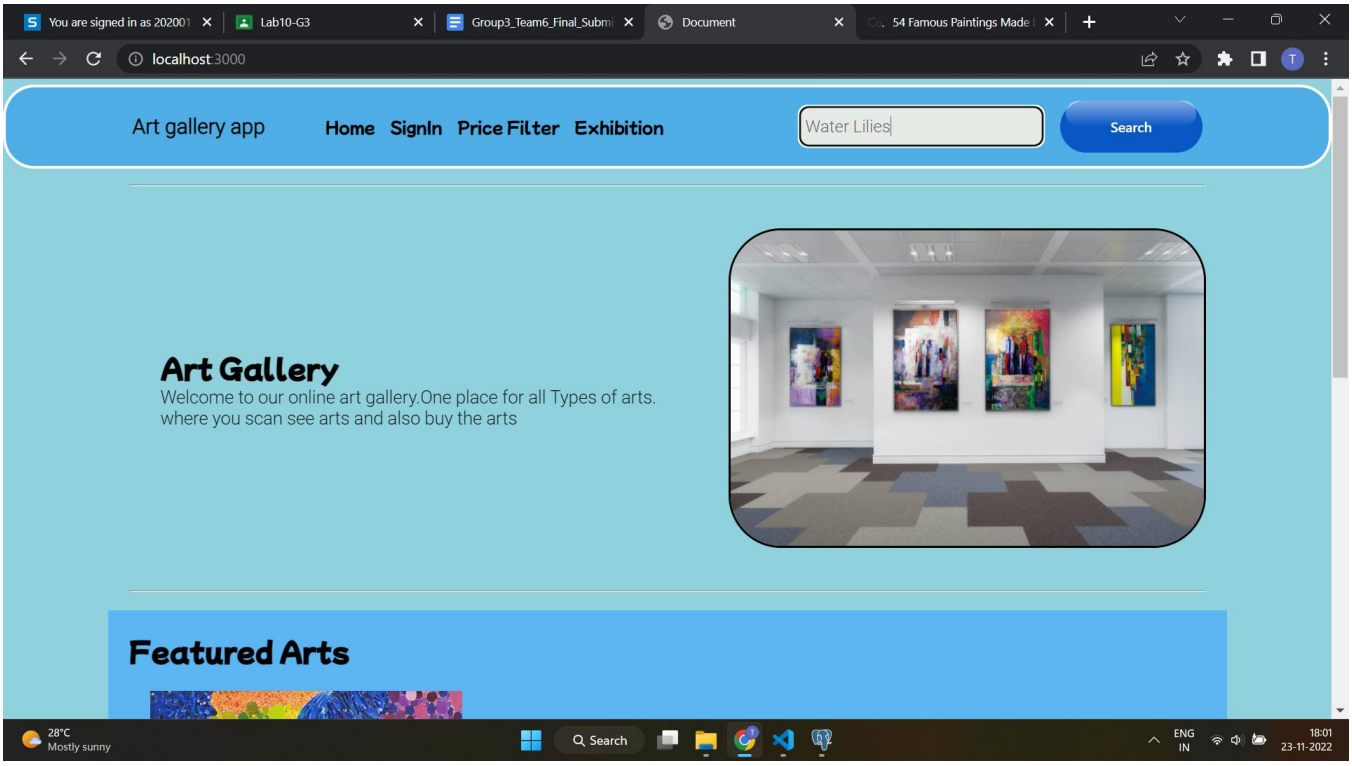
- **Exhibition available at this places:**

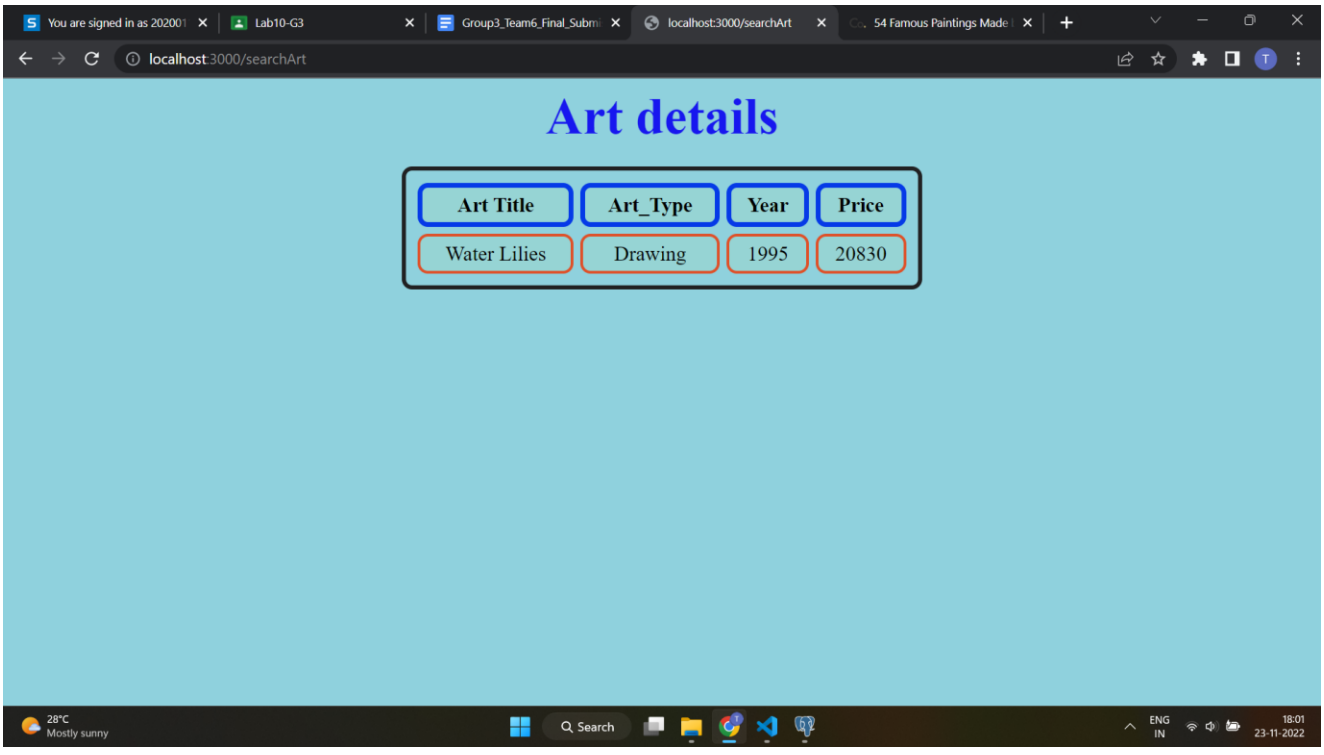


Result:

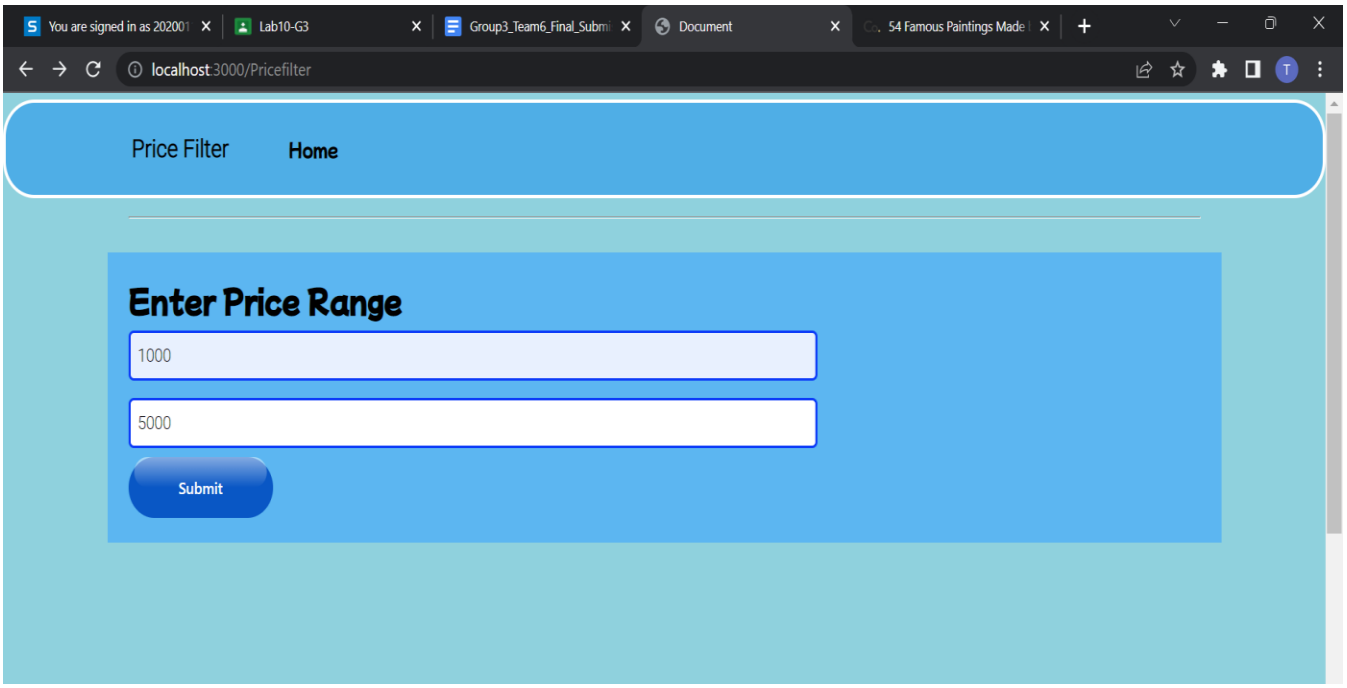
- **Search Art Details By searching art Name**

Search result:





- **Price Filter to search Arts whose price is between given range:**



Result: