Lab 10

Submitted in fulfillment of the course

IT214 (DBMS)

Lab 10 : Final

Name: Patel Kamal Hiteshbhai
Student ID: **202001018**

Name: Patel Aditya Dineshbhai
Student ID: **202001020**

Case Study: Clubbing
Group 1
Section 1
Team 1.9

Under the guidance of
Prof. Minal Bhise
Mentor TA: Ami

11 Nov 2022

Table of Content

SRS

ER Diagram

Final Normalization and DDL Scripts SQL Queries

Final Product

SRS

1. Introduction

1.1 Purpose:

The primary purpose of this document is to create a database for managing Data in a Club and build a system that will make the work of the Club employees and customers much more accessible. This system's purpose will be to track customers, their membership plans, eligibility, waiting list, etc.

This database will provide fast and convenient access to the required data, allow customers to view their profiles and see their previous visits, and allow employees to keep track of their customers, events details, and needs of events. This document serves as a guide for developers of this management system.

1.2 Intended Audience and Reading Suggestions:

The document is intended for:

- ➔ Club owner:
  - ◆ They are the admin of the database.
  - ◆ They have all the rights and can also add/remove managers.

- ➔ Club manager:
  - ◆ They are the first user or supreme users of the database.
  - ◆ They can add/remove employees and customers, update users' information, etc.

- ➔ Club employees:
  - ◆ They will be able to access basic entities like event_participants, waiting_list, etc.
  - ◆ They will have access to add or update customer details according to their position.

- ➔ Developers team at the club:
  - ◆ They are the mind behind this database system.
  - ◆ All the updates and functionalities have to be implemented by the developers. They are also responsible for technical glitches and have to solve them.

- ➔ Customers:
    - ◆ They can only view the information permitted by developers, such as their ID, Details, previous visit details, time spent, etc.

Other target audiences of this document are independent developers, and the primary example is database seekers for education purposes. Others are stakeholders for review purposes of club management system details.

1.3 The Scope of the project is as follows:-

This system provides a **user interface** through which the user can search for different clubs near them to search for upcoming events and register for them. They can also take a membership to a particular club and get an authentication id from clubs.

This system provides an **admin interface** to club owners and managers where they will be able to add details for upcoming events and can send the details/notifications of the forthcoming events to their premium members, membership benefits, and discount offers. This will facilitate the owners and stack owners with their club status and updates they require.

This system will also help create a **bridge** between customers and reviews where customers can see reviews of the club and chat using a link between the club manager and them. It eases the process of knowing club events to get past them and attend them using the system.

1.4 Description

- Club management system is designed to manage activities throughout the clubs and to manage events that are organized in clubs across India.
- In the past, there were two points of view on data management. The first is a demand-oriented administration (Policy-oriented), while the second is a public-oriented administration (Public-orientated).
- In a policy-oriented approach, the database is targeted to the company's goal and overall to manage customer behavior and study the improvements required to boost sales.
- When viewing the database in the public-oriented mode, users can use the site to read information about each club, its events, and previous activities anytime they wish to go out or take part in any activity.

- Clubs are required databases to manage all user data, employee data as well past event details.
- Whenever a user comes across any club or its details. He/She will analyze all the past events, rating given by their customers, foods, other things they are offering, price for particular events, etc.
- So for this thing there should be a database server from the club side that stores all these details and provides them to the customer when they ask about it.
- To design a Database we will have different perspectives required such as Club, managers, employees, customers etc. and relations/ tables that we are planning to create.

● The Perceptive that we are planning to implement for this system

1. Club (Includes owners, managers, employees, and Workers)
    - System will keep track of all clubs and show their upcoming event details, active days, reviews, etc.
    - It will store different tables like
        club, club_sections, club_type, club_detail, etc
    - It will store different attributes like
        club_id, club_name, city_name, Address, club_estab, email_id, phone_number, capacity, active_days, Active_hours, review_of_club, club_type.

2. Members
    - Our system will keep track of all customer data and previous events they have attended.
    - It will store different tables like
        member, club_name, club_role, etc.
    - It will store different attributes like
        memeber_detail: event_id, club_id, event_type (theme), age_limit, event_date, event_start_time, event_end_time, ony_subscripted, sponsors.

3. Events
    - We need a database of all events having information about the past events and upcoming events, ticket cost, event coordinator, etc.
    - It will store different tables like
        event, event_manager, event_section, event_serive, event_visitors
    - It will store different attributes like

event_detail:event_id, club_id, event_type (theme), age_limit, event_date, event_start_time, event_end_time, only_subscripted, sponsors.

4. Sponsors and Owners
    - It will store admin data with superior functionalities like updating or changing the data of employees.
    - It will store different tables like
      sponsors, owners, sponsor_service, etc.
    - it will store different attributes like
      sponser_id, club_holder, name,email_id, stack_percentage, phone_number.

● The relations that we are planning to implement for this system:-

1. Owner: This relation contains information about the owner of the club like Owner ID, Owner Name, etc.
2. Manager: This relation has manager ID, manager name, etc. details of the managers.
3. Sponsors: This contains information about sponsors and their services.
4. Employees: This relation has Employee id, Employee_section, Employee_history, joining date, etc.
5. Customer: This relation has customer_name, customer_id, attended_information, age, name, address of the customers.
6. Premium member: This relation has member_name, member_id, plan_type, age, and address.
7. Club sections: This contains section details like the food section, DJ section, etc.
8. Events: This relation has a list of all events with their event ID, age and height restrictions, duration of the event, the price for the event, etc.
9. Event manager: This contains information about the event manager and event details.
10. Event partners: This contains information about event partners and their contribution details.
11. Developers: This contains details of developers and the work they have done and their current project name.
12. Upcoming events: This relation has the information about the upcoming events, it's a date, time, capacity, name of the event manager, and other information.
13. Waiting list: This relation will have a waiting list of all the customers with their waiting number because there is a certain limit to each event.

1.5 Real-World working Flow:

1.5.1 Current working/flow.

They are currently following a basic **TwigaWorld** club system where they store basic information as text files and build a database focusing on basic functionalities and requirements that they encounter on a daily basis.
Refer point 2.1.1 for the Detailed system they are following.

1.5.2 Difficulties or Problems may be faced.

The current system has the following Problems and difficulties.

- ➔ Data redundancy
  - ◆ Data has less information, but still, it has repetition for different participants.
- ➔ Inconsistency:
  - ◆ There might be cases where data from different departments may be merged. Inconsistency might exist in event details as there is no comparison with previous event data.
- ➔ Data securities
  - ◆ There is no encapsulation or encryption of data which might cause the data leakage.
- ➔ Atomicity of updates
  - ◆ Sometimes due to technical failure, the database may be left in an inconsistent state with partial updates, which may cause significant problems for the club.
- ➔ Concurrent access
  - ◆ Concurrent access by multiple users to the database can lead to data inconsistency.

2. Data collection or Fact-finding phase

2.1 Background Readings

A) Reading and Description:

- ➔ We used one sample SRS Document from Sample SRS To get a basic idea of the introduction and what we have to follow to make an informative SRS.
- ➔ We have visited some club sites, twigaworld, and clubo7 , to get the information and interface required to manage the customer's view.
- ➔ We get the basic ideas from the templates provided at google classroom for this course.
- ➔ We get clear concepts of database systems like the E-R model and other information on design databases from " Database system concept by S. Sudarshan, Abraham Silberschatz, Henry F. Korth."

2.1.1 Existing Database System clubs following.

Currently, there is a basic **TwigaWorld** club system where they store basic information as text files and build a database focusing on basic functionalities and requirements they face regularly.

2.1.1.1 Existing Features and Flows:

To customer:

- ➔ Getting the latest news of events
- ➔ Getting information about new offers
- ➔ Customers can get their previous visit time and date
- ➔ Customers can get updates about the club and its newly added facilities.
- ➔ Customers can enroll in different events and take membership.
- ➔ Customers can get the details of a special discount on their birthday if there are any.

To Member:

- ➔ Getting the latest news of events
- ➔ Getting information about new offers
- ➔ Members can bring their previous visit time and date
- ➔ Members will be able to see discount pricing compared to regular customers.
- ➔ Members can get the details of a special discount on their birthday if there are any.
- ➔ Members can view the details of some special events, which are only limited to subscripted members.
- ➔ Members can see the buying and expiry date of their membership plan.
- ➔ Members can see the utilization of their subscriptions during the membership period.

➔ Members get unique website features that regular members can’t reach, like an activity dashboard.

To Employees:

➔ They will be allowed to see details of participants in upcoming events.
➔ Employees can see their bond period details and all.
➔ Employees can request their respective managers for a salary increment.
➔ Employees can request their respective managers for any changes to their profiles.
➔ Employees have access to ban or restrict any customer from any particular event.
➔ Employees can request their respective managers if they have any unique or creative ideas for the club.
➔ Employees can see and review their joining date and their old version of documents.

To Managers:

➔ They will have the authority to see and update the details of any employee or member or remove them.
➔ They will be able to add events and change event details.
➔ Managers can change the rules and regulations of club and club events.
➔ Managers can see and approve different requests of employees like salary increments, profile updating, etc.

- **Combined Requirements gathered from the readings:**

pla

➔ First, we understand that from the reader's or analyzer's viewpoint, the introduction plays an essential role in clarifying what SRS contains and what order the document follows.
➔ What objects are required to design and maintain the database system that covers all the information about clubs, club members, employees, etc.?
➔ How security is required while developing the database.
➔ How suitable structures and functions play essential roles in the performance.
➔ How good and clean is the user interface/Site required for customers' easy access to club details?

2.2 Interviews

1. Interview Plan

System: Club Management System (CMS)
Project Reference: SF/SJ/2003/12

Interviewee:
1) Stefano Lanzoni(Role Play) Designation: Director of Operations at TwigaWorld
Interviewer:
1) Kamal Patel Designation: Developer
2) Aditya Patel Designation: Developer

Date: 26/09/2022 Time: 12:30
Duration: 45 minutes Mode: Online Place: Google meet

Purpose of Interview:
Preliminary meeting to identify their services, requirements, policies, and problems at TwigaWorld.

Agenda:
- Service provided by the company to the customers.
- Current Policies, rules, and regulations to manage the system.
- Talk about the challenges that consumers have and how you resolve them.

Documents to be brought to the interview:
- Rough plan of Club building.
- Documents showing the company's past events and programs.
- Any documentation on the present system and client demands.

● Interview Summary

System: Club Management System (CMS)
Project Reference: SF/SJ/2003/12

Interviewee:
1) Stefano Lanzoni(Role Play) Designation: Director of Operations at TwigaWorld
Interviewer:
1) Kamal Patel Designation: Developer
2) Aditya Patel Designation: Developer

Date: 26/09/2022 Time: 12:30
Duration: 45 minutes Mode: Online Place: Google meet

Purpose of Interview:
Preliminary meeting to identify their services, requirements, policies, and problems at TwigaWorld.

Results of the interview:

- Security will be kept at ground level to check the authorized ID cards of all persons and the age limit.
- Strict rules will be implemented, so no underage customer gets unsuitable drinks.
- There will be different sections like only DJs, bars with DJs, food and drinks, and more so customers can get their choices.
- Customers' information will be stored in encrypted form, so non-authorized employees can’t illegally access customers' data, and security keys used in encryption will be changed from time to time.
- Preference and activity of subscripted members will also be stored to provide them better and faster service next time.
- Customers' input will be collected regularly, and the system will be improved as a result. In case of contrast, the majority will be taken into consideration.

2. Interview Plan

System: Club Management System (CMS)
Project Reference: SF/SJ/2003/12

Interviewee:
1) Joseph Jordan (Role Play) Designation: Developer at TwigaWorld
Interviewer:
1) Kamal Patel Designation: Developer
2) Aditya Patel Designation: Developer

Date: 27/09/2022 Time: 2:30
Duration: 40 minutes Mode: Offline Place: At his office

Purpose of Interview:
To understand their current database and how they manage it. How do they update to overcome upcoming challenges and improve service at TwigaWorld?

Agenda:
- Taking an overview of the current database.
- They update their database from time to time and update them according to new rules and conditions and improve system interfaces.
- Talk about problems faced during maintenance and how they resolve it.

Documents to be brought to the interview:
- Data showing their performance over the years.
- A document containing the database and security system followed by the company.

● Interview Summary

System: Club Management System (CMS)
Project Reference: SF/SJ/2003/12

Interviewee:
1) Joseph Jordan (Role Play) Designation: Developer at TwigaWorld
Interviewer:
1) Kamal Patel Designation: Developer
2) Aditya Patel Designation: Developer

Date: 27/09/2022 Time: 2:30
Duration: 40 minutes Mode: Offline Place: At his office

Purpose of Interview:
To understand their current database and how they manage it. How do they update to overcome upcoming challenges and improve service at TwigaWorld?

Results of the interview:

- After the Interview, we get basic information on ground-level database creation and management.
- How databases help increase performance and improve service by efficiently analyzing data using algorithms.
- How updating the Database daily is required to match up with the latest technologies.
- Websites play an essential role in increasing the reach of clubs.

3. Interview Plan

System: Club Management System (CMS)
Project Reference: SF/SJ/2003/12

Interviewee:
1) Jordan Ray(Role Play) Designation: Customer and Member at TwigaWorld
Interviewer:
1) Kamal Patel Designation: Developer
2) Aditya Patel Designation: Developer

Date: 28/09/2022 Time: 2:00
Duration: 30 minutes Mode: Offline Place: At Club

Purpose of Interview:
Understand his opinion about the club, his expectations, and complaints towards the club and system.

Agenda:
- Knowing their experience of our club till now.
- Features that are dissatisfactory or need improvement.
- Future events or things that were more likable to the customer.
- Difficulties faced by customers and how they got responded.
- Ask them separately about their experiences with staff and restrooms.

Documents to be brought to the interview:
- Data showing the customer satisfaction rate of the club.
- The document contains problems faced by the customer on a regular visit to clubs.

● Interview Summary

System: Club Management System (CMS)
Project Reference: SF/SJ/2003/12

Interviewee:
1) Jordan Ray(Role Play) Designation: Customer and Member at TwigaWorld
Interviewer:
1) Kamal Patel Designation: Developer
2) Aditya Patel Designation: Developer

Date: 28/09/2022 Time: 2:00
Duration: 30 minutes Mode: Offline Place: At Club

Purpose of Interview:
Understand his opinion about the club, expectations, and complaints towards the club and system.

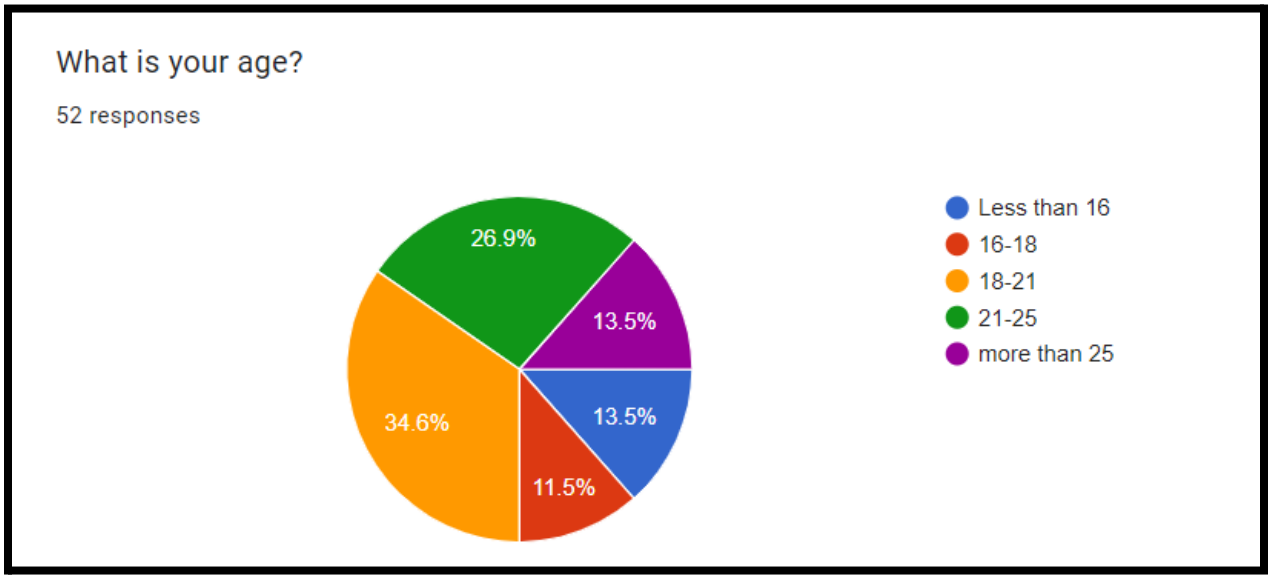
Results of the interview:

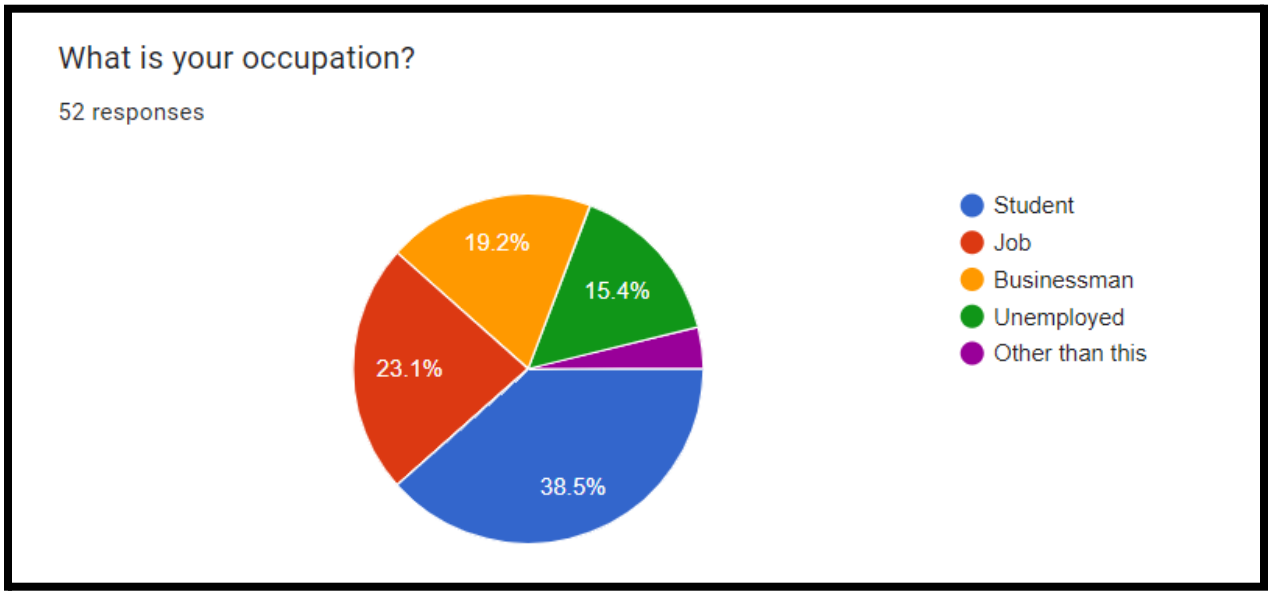
- The website interface should be clear and free of unnecessary info and advertisements.
- The system should have helpful features such as waiting list reminders, transfer balances, etc.
- The truthfulness of data we got from the club side and data collected from customers.
- The feedback system should be updated, and more high-level authorities should be directly involved.
- Users suggested frequent updates of membership programs and sites according to events programs.

- **Combined Requirements gathered from the Interviews:**

➔ It is necessary to create a front end so that users may access specific data.
➔ Membership programs and webpages are often updated in accordance with event programming.
➔ The system should have appealing features such as waiting list reminders, transfer balances, and so on.
➔ The database will be backed up at regular intervals so that lost data may be retrieved.
➔ We must ensure that other databases are not negatively affected. (integrity is maintained).
➔ Employees should be able to have more command of event management.
➔ Suggestions should be sent and solved by a higher level of management.
➔ System administrators and employees will be allocated separate roles to maintain the protection of sensitive client information.
➔ There should be a proper system to create suggestions for different members according to their previous experience at the club.

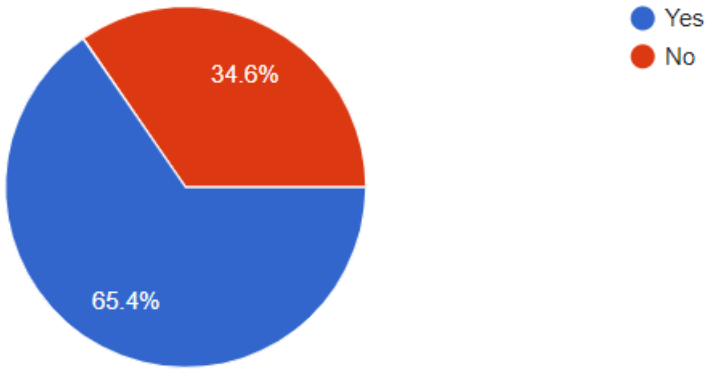
2.3 Questionnaires:





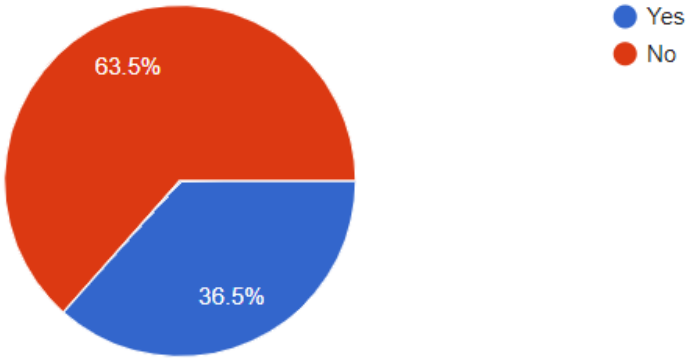
Have you visited any club?

52 responses



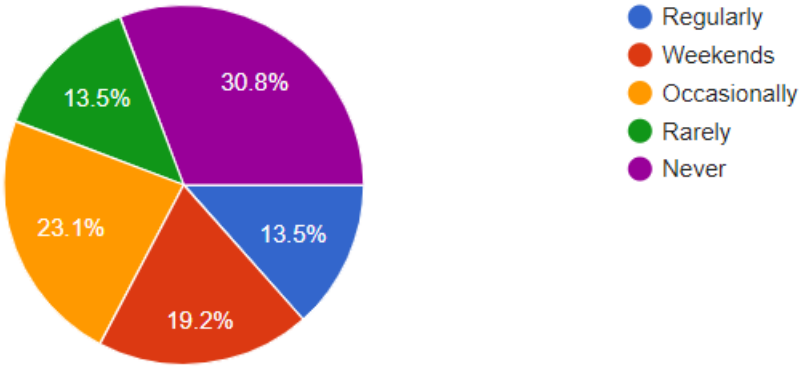
Are you member of any club?

52 responses



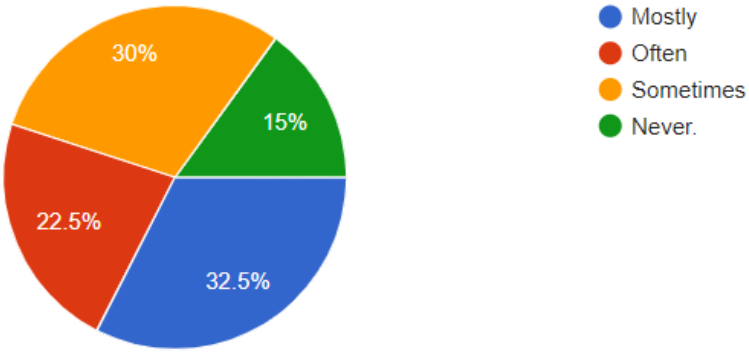
how often did you attended club?

52 responses



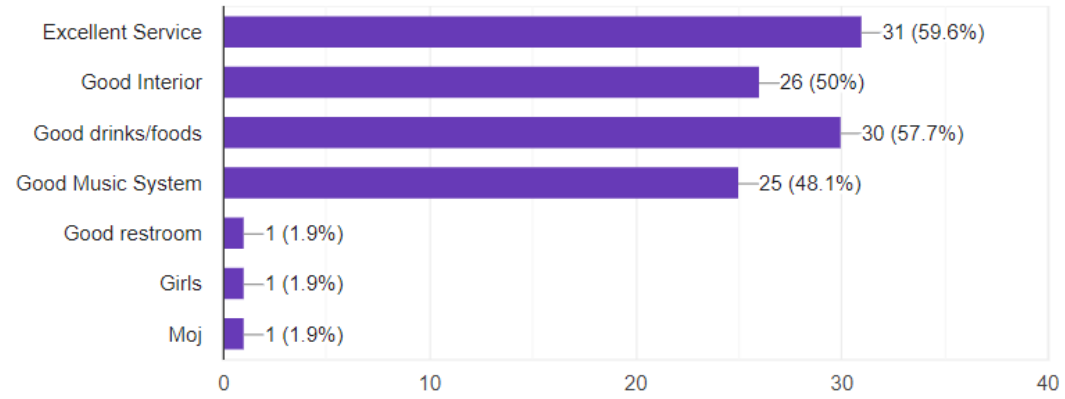
You often take drink at club?

40 responses



What do you expect from a good Club?

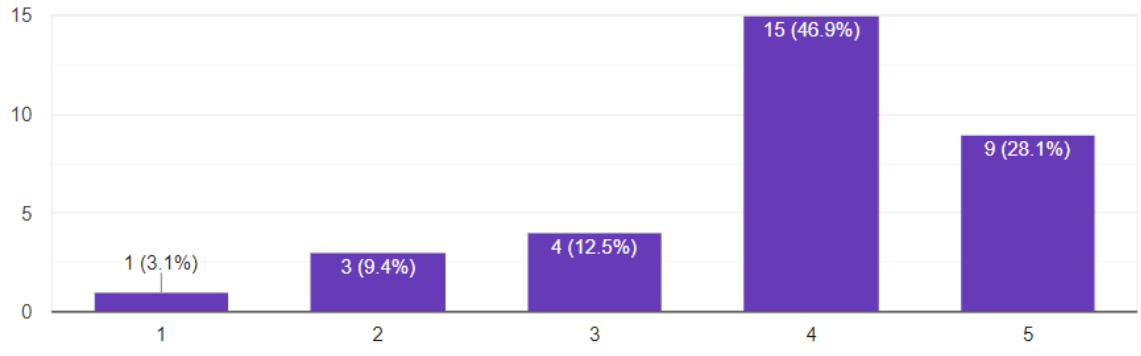
52 responses



What is Your experience of Restroom.

Copy

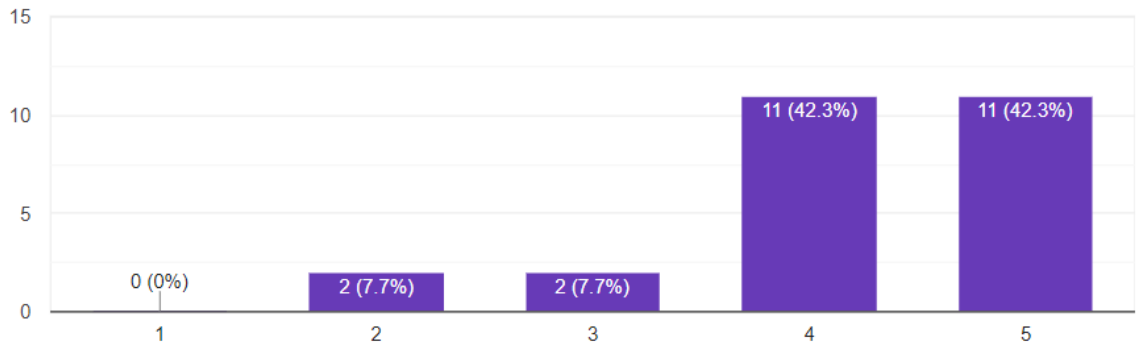
32 responses



Are you satisfied with premium subscription (Membership)?
Please Rate your experience.

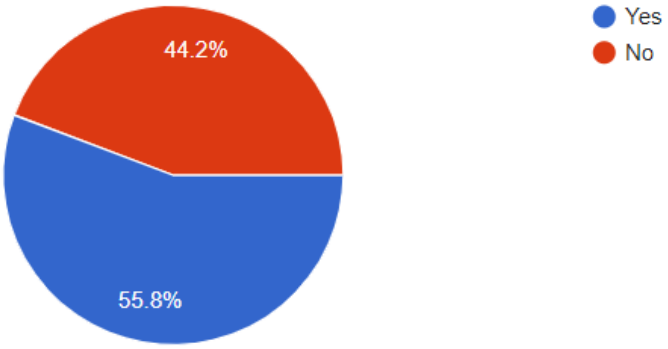
Copy

26 responses



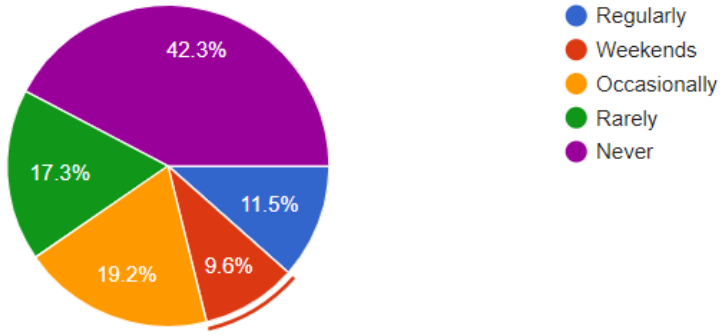
Have you attended any Dj Party at clubs?

52 responses



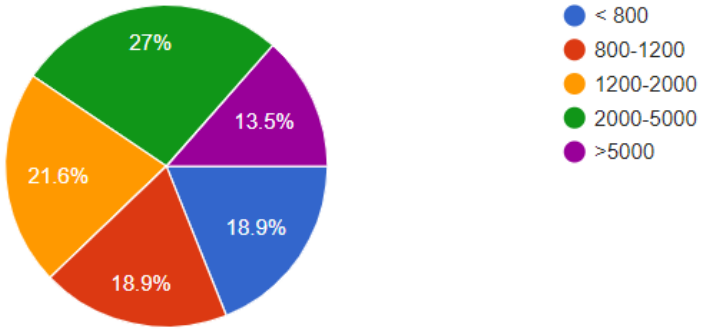
how often did you attended DJ Party at clubs?

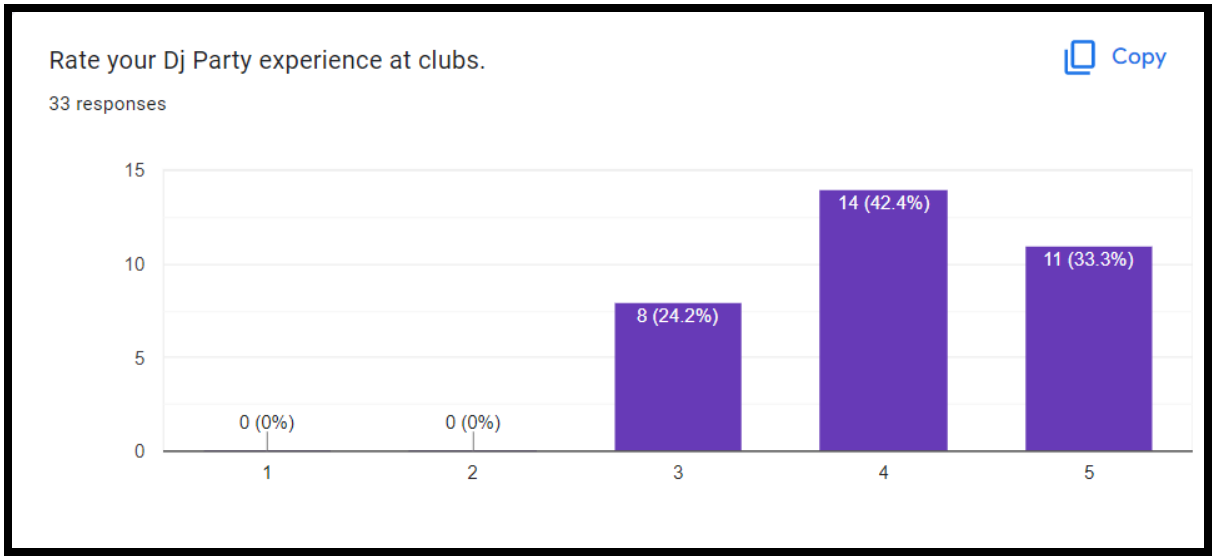
52 responses



Please select your average cost of single visit at any club (in INR).

37 responses





- **Combined Requirements gathered from the Questionnaires:**

➔ More people like to have good music experiences.
➔ Premium subscriptions should have more benefits compared to normal customers.
➔ A good bot system should be available to obtain rapid assistance on websites.
➔ Several data points must be observed in real-time.
➔ There should be a proper system to create suggestions for different members according to their previous experience at the club.

2.4 Observations:

System: TwigaWorld Club
Project Reference: SF/SJ/2021/10
Observations by: Kamal Patel, Aditya Patel
Date: 18/09/2022 Time: 14:30
Duration: 45 minutes Place: TwigaWorld Club at MoteCarlo

Observations:

- ➔ Currently using the basic database system with required functionalities.
- ➔ Lots of data redundancy observed.
- ➔ Lack of coordination between employees working in different sections such as the event management section, and decoration section.
- ➔ No real-time update of events like remaining slots for events.
- ➔ Server takes too much load when multiple users log in or when slot booking of the event starts.
- ➔ Modification of data is very hard for developers due to data inconsistency.
- ➔ Nobody else had access to one person's data. Data privacy and security were prioritized, such that even the owner or management could not change user data.
- ➔ Proper age limitations were enforced by requiring identification upon entry.
- ➔ The electricity and internet should be maintained 24/7.
- ➔ Lack of proper marking.
- ➔ There are no prior notifications to members about upcoming events.

- **Combined requirements from Observations:**

- ➔ There should be a good and clean user interface with an easy registration process for events and membership plans to ensure a good quality experience for the users.
- ➔ Different user classes like members and regular customers should be given different access to see events and prices.
- ➔ There should be proper id proof verification on entry to check user age.
- ➔ The user should be able to save any event as a favorite for future usage or reference.
- ➔ The system should optimize the user's work by saving past data and experience.
- ➔ There should be enough vital gadgets and party supplies to avoid long lines.
- ➔ There should be a real time update of the event.
- ➔ A help desk should be available at all times to handle user problems by guiding them via software.

3. Fact - Finding Chart

Objective	Technique	Subject	Time commitment
To get background knowledge on the Club	Background Readings and website	Few similar Projects and requirements of Club	½ Day
Preliminary meeting to get the background of the Club	Interview	Stefano Lanzoni (Director of Operations at TwigaWorld)	2 Hours
To get the information about their current database and their viewpoints on different sections	Interview	Joseph Jordan (Developer at TwigaWorld)	2 Hours
To know the customer point of view, their needs and their satisfaction	Interview	Jordan Ray (Customer and Member at TwigaWorld)	3 Hours
To follow up the development of business understanding	Observation	Creative employees of the Club (Total 2)	1 Day
To get knowledge of the real world Club environment	Observation	Members and Subscriptor	3 Hours
To understand the public opinion and experience	Questionnaire	Student Database Developer	1 Day

4. List Requirements

- First, we understand that from the reader's or analyzer's viewpoint, the introduction plays an essential role in clarifying what SRS contains and what order the document follows. **(Interview)**
- What objects are required to design and maintain the database system that covers all the information about clubs, club members, employees, etc.? **(Background Readings + Interviews)**
- How security is required while developing the database. **(Background Readings Interview)**
- How good and clean is the user interface/Site required for customers' easy access to club details? **(Background Readings + Observations)**
- The system should have appealing features such as waiting list reminders, transfer balances, and users should be able to save any event as a favorite for future usage or reference. **(Interview + Observation).**

➔ More people like to have good music experiences. **(Background Readings + questionnaire)**

➔ Premium subscriptions should have more benefits compared to normal customers. **(questionnaire)**

➔ A good bot system should be available to obtain rapid assistance on websites**. (interview + questionnaire)**

➔ Several data points must be observed in real-time**. (questionnaire + Observation)**

➔ There should be enough vital gadgets and party supplies to avoid long lines. **(Background Readings + Observations)**

➔ There should be a proper system to create suggestions for different members according to their previous experience at the club. **(Observation + Interview).**

➔ There should be a proper system to create suggestions for different members. **(Observation + Interview)**

- The database will be backed up at regular intervals so that lost data may be retrieved. **(Interview + Background Readings)**
- Employees should be able to have more command of event management. **(Interview)**
- Suggestions should be sent and solved by a higher level of management. **(Observation)**
- System administrators and employees will be allocated separate roles to maintain the protection of sensitive client information. **(Interview)**

5. User categories and Privileges

1. Database Administrator
   - ➔ Database Administrators create, modify, and manage databases. He is in charge of bug detection, security concerns, and root cause analysis.
   - ➔ Other user classes can be granted rights by the Database Administrator.
   - ➔ Database Administrators give memory and processing resources.
   - ➔ In short, Database Administrator controls all three levels of DBMS.

2. Database Designers
   - ➔ Work on DDL (data definition language), which includes tables, views, indexes, constraints, triggers, and procedures, among other things.
   - ➔ Determines how data should be connected to one another.

3. System Analyst
   - ➔ Gather the information about end requirements.
   - ➔ Check to see whether these prerequisites have been met.

4. Programmers
   - ➔ DDL is implemented by programmers creating code in many languages such as PostgreSQL, MySQL, COBOL, and others.

5. End Users:
   - ➔ They don't have technical knowledge about DBMS.
   - ➔ They only access functions allowed by designers using interfaces such as websites or software. (Here, End user can be a member who wants to find information about the club or its upcoming events)

6. Data Entry Professionals
   - ➔ Do data entry in the database.In clubs mostly data entry done by employees or event coordinators.
   - ➔ Sometimes data entry can be done by end-users as well.

Privileges :

1. Database Administrator: The role of the Database administrator is to design a database, find bugs, security issues, root cause analysis, and data recovery.They have privileges to perform tasks that no one else can do as they are in charge of the database.They can access all 3 layer of database.
2. Database designers, analysts, and programmers: They primarily design and implement efficient databases according to the need. They will have access to the logical level of the database.

3. End-users: End-users have frequent use of the database. They do not have access to inserting, modifying, or deleting in the database. They just can see the interface allowed by the database designer.

6. Assumptions

1. All required numerical information is in accurate form.
2. We anticipated that all required data inputs would be filled out correctly.
3. All consumers and employees have encrypted user IDs and passwords that cannot be exploited.
4. No customers are given priority on the waiting list of the event as employees have access to event details.
5. It is assumed that all things stated on event details are available and in working condition.
6. Once a user is enrolled, the user is not willing to deregister from the site.

7. Business Constraints

1. The amount of hardware, and stuff like drinks/foods available is limited.
2. The capacity of the club is limited.
3. Physical devices and components for storing database and processor should be latest and ful-filling requirements.This will be expensive.
4. The data must be saved on a server. Storing large quantities of data on cloud servers might be an expensive proposition.
5. Physical DBMS hardware and components like Memory and CPU should be kept up to date to reduce system crashes, but this will cost a lot of money.

References:

For sample srs Document
https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

To draw a reference view of DB.
https://vertabelo.com

Final ER Diagram and Noun analysis

1. Problem Description

- Club management system is designed to manage activities throughout the clubs and to manage events that are organized in clubs across India.
- In the past, there were two points of view on data management. The first is a demand-oriented administration (Policy-oriented), while the second is a public-oriented administration (Public-orientated).
- In a policy-oriented approach, the database is targeted to the company's goal and overall to manage customer behavior and study the improvements required to boost sales.
- When viewing the database in the public-oriented mode, users can use the site to read information about each club, its events, and previous activities anytime they wish to go out or take part in any activity.
- Clubs are required databases to manage all user data, employee data as well past event details.they also required different employees to run clubs such as cook, bartender, Sweeper, etc.
- Different clubs require event types like Only DJ Party, party_club, Drink Party, etc.
- Whenever a user comes across any club or its details. He/She will analyze all the past events, rating given by their customers, foods, other things they are offering, price for particular events, etc.
- So for this thing there should be a database server from the club side that stores all these details and provides them to the customer when they ask about it.
- To design a Database we will have different perspectives required such as Club, managers, employees, customers etc. and relations/ tables that we are planning to create.

The Perceptive that we are planning to implement for this system

1. Club (Includes owners, managers, employees, and Workers)
    - System will keep track of all clubs and show their upcoming event details, active days, reviews, etc.
    - It will store different tables like
        club, club_sections, club_type, club_detail, etc
    - It will store different attributes like

club_id, club_name, city_name, Address, club_estab, email_id, phone_number, capacity, active_days, Active_hours, review_of_club, club_type.

2. Members
    - Our system will keep track of all customer data and previous events they have attended.
    - It will store different tables like
        member, club_name, club_role, etc.
    - It will store different attributes like
        memeber_detail: event_id, club_id, event_type (theme), age_limit, event_date, event_start_time, event_end_time, ony_subscripted, sponsors.

3. Events
    - We need a database of all events having information about the past events and upcoming events, ticket cost, event coordinator, etc.
    - It will store different tables like
        event, event_manager, event_section, event_serive, event_visitors
    - It will store different attributes like
        event_detail:event_id, club_id, event_type (theme), age_limit, event_date, event_start_time, event_end_time, only_subscripted, sponsors.

4. Sponsors and Owners
    - It will store admin data with superior functionalities like updating or changing the data of employees.
    - It will store different tables like
        sponsors, owners, sponsor_service, etc.
    - it will store different attributes like
        sponser_id, club_holder, name,email_id, stack_percentage, phone_number.

● The relations that we are planning to implement for this system:-

1. Owner: This relation contains information about the owner of the club like Owner ID, Owner Name, etc.
2. Manager: This relation has manager ID, manager name, etc. details of the managers.
3. Sponsors: This contains information about sponsors and their services.
4. Employees: This relation has Employee id, Employee_section, Employee_history, joining date, etc.

5. Customer: This relation has customer_name, customer_id, attended_information, age, name, address of the customers.
6. Premium member: This relation has member_name, member_id, plan_type, age, and address.
7. Club sections: This contains section details like the food section, DJ section, etc.
8. Events: This relation has a list of all events with their event ID, age and height restrictions, duration of the event, the price for the event, etc.
9. Event manager: This contains information about the event manager and event details.
10. Event partners: This contains information about event partners and their contribution details.
11. Developers: This contains details of developers and the work they have done and their current project name.
12. Upcoming events: This relation has the information about the upcoming events, it's a date, time, capacity, name of the event manager, and other information.
13. Waiting list: This relation will have a waiting list of all the customers with their waiting number because there is a certain limit to each event.

2. Noun (& Verb) Analysis

Table 1 : All Extracted Nouns & Verbs from Problem Description

Nouns	Verbs
management	
System	Design
club	
Event	
Database	design
Purpose	
customer	drink
Access	
Site	design
User	Login
employee	work

Nouns	Verbs
Price	rise
Server	work
customer	buy
Details	fill
food	cook
Thing	
Store	sell
Owner	manage
Age	store
customer	pay
Information	fill
Id	give
Name	add
Manager	manage
Date	store
Member	pay
List	fill
Section	divide
height	restriction
duration	allow
Address	store
Work	do
Project	
time	
Capacity	

Nouns	Verbs
Number	
Limit	
Customer	Buy
Sponsors	give
Restriction	allow
Partner	give
Developer	design
Premium	buy
DJ party	dance
customer	pay
Admin	manage
Theme	design
Ticket	allocate
Cost	manage
Coordinator	arrange
Visitor	visit
Reviews	analyze
Worker	Work
cook	cook
Sweeper	Sweep
bartender	Serve

Table 2 : Accepted Noun & Verbs list

Candidate entity set	Candidate attribute set	Candidate relationship set
Club	club_id club_section club_type club_detail	Manage
Event_manager	Manager_id Manager_role Manager_name Manager_age Manager_phone_no Manager_working_hours Manager_section Manager_salary Event_id	Manage
Club section	Id club_id, club_name, section_id city_name, Email_id, phone_number, capacity	Manage
Member	member_id Event_id member_name member_dob member_id_proof member_contact_number member_visiting_time_date member_paid_amount member_payment_mode	Participant Assign Pay Wait

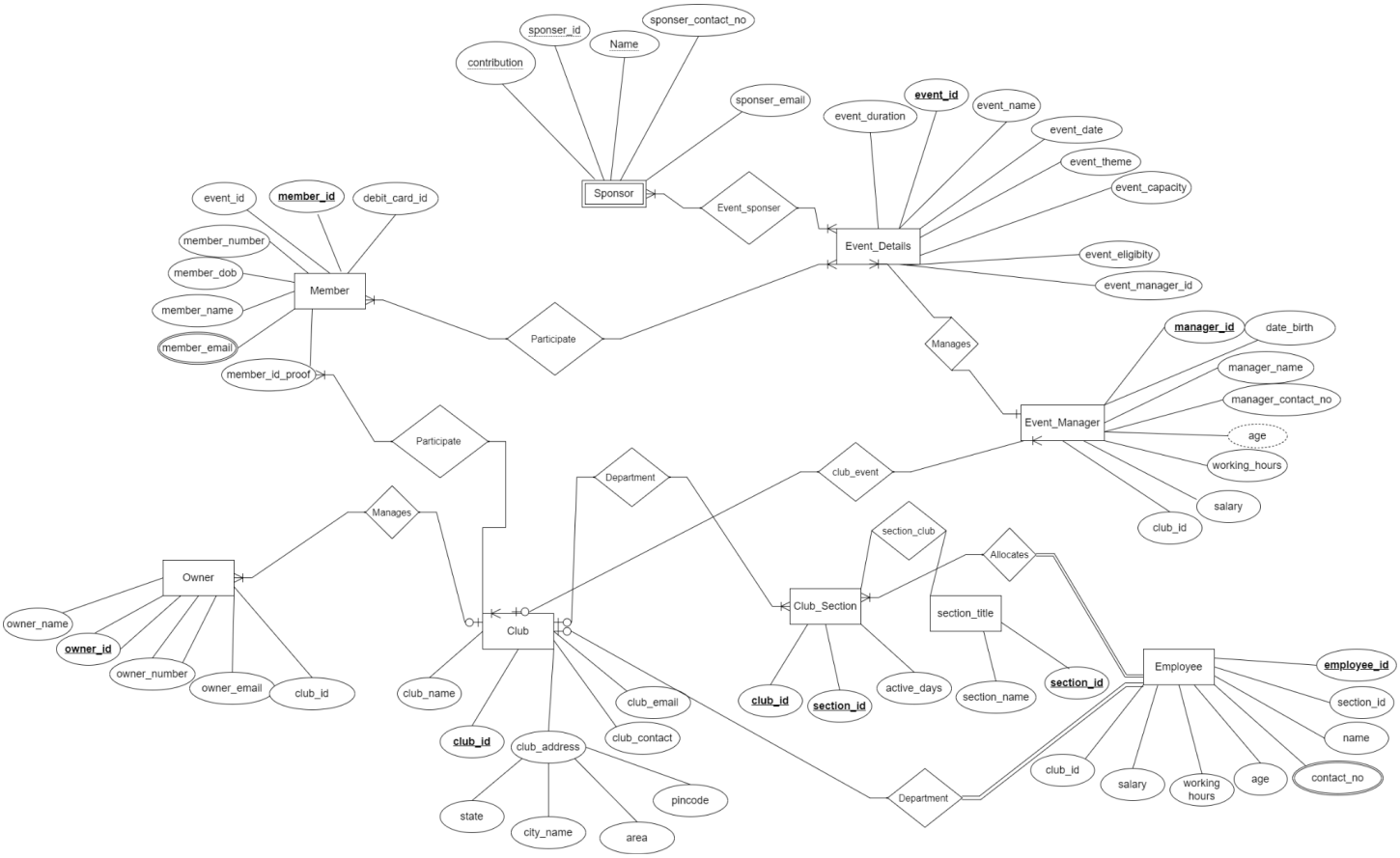
	Member_total_expense member_debit_card_id	
Event details	Event_id Event_manager_id Event_name Event_theme Event_type Event_date Event start_time end_time Event_price_rates Event_elegibilty Event_capacity Event_contact_detail Event_sponsers	
Employee	Employee_id Section_id Employee_role Employee_name Employee_age Employee_phone_no Employee_address Employee_joining_date Employee_salary	Help manage
Sponsor	Sponsor_id Sponsor_name Sponsor_contribution Event_id	Wait Dance Drink
owner	Owner_name Owner_id Owner_phone_no owner _email club_id	manage

Table 3 : Rejected Noun & Verbs list

Noun	Reject Reason
cook	They are all different type of worker required to run the club.We can group them in single attribute named employee_type
worker	
Sweeper	
Bartender	
DJ party	It can be covered in party_type, no need of different attribute
drink party	
user	duplicate
customer	
cost	duplicate
price	
capacity	Attribute
Address	Attribute
Employee experience	irrelevant
Joining date	irrelevant
Event date	Attribute
Payment mode	Vague
Past experience	irrelevant
Details	Association
Amount	Attributes
thing	general
list	general
Reviews	irrelevant
height	Vague

Things	Irrelevant
store	Unnecessary
number	Unnecessary
limit	duplicate
partner	duplicate
premium	duplicate
department	association
section	Attributes

3. Develop the ER Diagram (ERD)



Final Normalization and DDL Scripts, SQl Queries

1. Mapping E-R Model to Relational Model :

1. Club

- club(club_id, club_name, club_address, club_contact, club_email)
- PK is club_id
- Constraints:
    - club_contact and club_email may contain **null values**.

❖ Functional dependency:
- ➢ Club_id → (club_name, club_address, club_contact, club_email)
- ➢ 1NF ⇒ No as we have a multivalued attribute club_contact and composite attribute cub_address.

❖ Decomposition into
- ➢ club(club_id, club_name club_email)
    - ■ Functional dependency:
        - Club_id → (club_name, club_contact,club_email)
        - 1NF ⇒ Yes as no multivalued/composite attribute
        - 2NF ⇒ Yes as no partial key is used
        - 3NF ⇒ Yes as no transitive dependency
        - BCNF ⇒ Yes as all determinants are super key
- ➢ club_contact(club_id,contact_number)
    - ■ Functional dependency:
        - (Club_id,contact_number) → (Club_id,contact_number)
        - 1NF ⇒ Yes as no multivalued/composite attribute
        - 2NF ⇒ Yes as no partial key is used
        - 3NF ⇒ Yes as no transitive dependency
        - BCNF ⇒ Yes as all determinants are super key
- ➢ club_address(club_id, pincode, city_name, state, area)
    - ■ Functional dependency:
        - (Club_id) → (pincode, city_name, state, area)
        - 1NF ⇒ Yes as no multivalued/composite attribute
        - 2NF ⇒ Yes as no partial key is used
        - 3NF ⇒ Yes as no transitive dependency
        - BCNF ⇒ Yes as all determinants are super key
- ➢ This is lossless join decomposition as the common attribute club_id in first 2 tables is CK for club and common attribute club_id in last 2 tables is CK for club_address.

2. owner

- Owner(owner_id, club_id, owner_name, owner_number, owner_email)
- PK is owner_id
- FK is club_id referenced from club

Constraints:

- **club** relations are **many to one** so there may be many owner to single club
- **Owner** must participate in the **club** so FK club_id can't be empty

❖ Functional dependency:

- ➢ Owner_id → (club_id, owner_name, owner_number, owner_email)
- ➢ 1NF ⇒ Yes as no multivalued/composite attribute
- ➢ 2NF ⇒ Yes as no partial key is used
- ➢ 3NF ⇒ Yes as no transitive dependency
- ➢ BCNF ⇒ Yes as all determinants are super key

3. Club Section

- club_section(section_id, club_id, active_days, section_name)
- PK are section_id and club_id
- FK is club_id referenced from club

Constraints:

- **Club** relations are **many to one** so there may be many sections to a single club.
- **Club** section must participate in the **club** so FK club_id can't be empty.
- We choose **NO ACTION** as default on performing **updation or deletion**, because it might affect club relation as we have borrowed club_id as a FK.

❖ Functional dependency:

- ➢ Section_id → (section_name)
- ➢ (club_id, section_id) → (active_days)
- ➢ 1NF ⇒ Yes as no multivalued/composite attribute
- ➢ 2NF ⇒ No, as we have partial keys as determinants.

❖ Decomposition into

- ➢ section_title(Section_id, section_name)
  - Functional dependency:
    - (section_id) → (section_name)
    - 1NF ⇒ Yes as no multivalued/composite attribute
    - 2NF ⇒ Yes as no partial key is used
    - 3NF ⇒ Yes as no transitive dependency
    - BCNF ⇒ Yes as all determinants are super key

- ➢ club_contact(section_id, club_id, active_days)
  - Functional dependency:
    - (section_id, club_id) → (active_days)
    - 1NF ⇒ Yes as no multivalued/composite attribute
    - 2NF ⇒ Yes as no partial key is used
    - 3NF ⇒ Yes as no transitive dependency
    - BCNF ⇒ Yes as all determinants are super key
- ➢ This is lossless join decomposition as the common attribute Section_id in both tables is CK for secton_title.

4. Employee
   employee (employee_id, name, age, working_hours, salary, section_id,club_id)

- PK is employee_id.
- FK is section_id referenced from club_section.

Constraints:

- In **domain constraint** age, working_hours, salary must be integer and may be null. While section_id can't be null.
- club_section relations are **many to one** so there may be many employees to a single club section.
- club_section section must participate in the **club** so FK section_id can't be empty.
- **Referential Integrity Constraints** We choose **NO ACTION** as default on performing **updation or deletion**, because it might affect club relation as we have borrowed club_id as a FK.

❖ Functional dependency:
- ➢ Employee_id → (name, age, working_hours, salary, section_id)
- ➢ 1NF ⇒ Yes as no multivalued/composite attribute
- ➢ 2NF ⇒ Yes as no partial key is used
- ➢ 3NF ⇒ Yes as no transitive dependency
- ➢ BCNF ⇒ Yes as all determinants are super key

5. event_details

- event_details(event_id, event_duration, event_name, event_date, event_theme, event_capacity, event_sponsor_id, event_eligibity, event_manager_id,contribution)
- PK is event_id

Constraints:

- Here sponser_id is a multivalued attribute.
- **Club** section must participate in the **club** so FK club_id can't be empty.

- We choose **NO ACTION** as default on performing **updation or deletion**, because it might affect club relation as we have borrowed club_id as a FK.

❖ Functional dependency:
- ➢ Event_id → (event_duration, event_name, event_date, event_theme, event_capacity, event_sponsor_id, event_eligibity, event_manager_id,contribution)
- ➢ 1NF ⇒ No as multivalued attribute sponsor

❖ Decomposition into
- ➢ event_details(<u>event_id</u>, event_duration, event_name, event_date, event_theme, event_capacity, event_eligibity, event_manager_id)
  - Functional dependency:
    - Event_id → (event_duration, event_name, event_date, event_theme, event_capacity, event_eligibity, event_manager_id)
    - 1NF ⇒ Yes as no multivalued/composite attribute
    - 2NF ⇒ Yes as no partial key is used
    - 3NF ⇒ Yes as no transitive dependency
    - BCNF ⇒ Yes as all determinants are super key

- ➢ event_sponser(<u>event_id</u>, <u>sponsor_id</u>, contribution)
  - Functional dependency:
    - (event_id, sponsor_id) → (contribution)
    - 1NF ⇒ Yes as no multivalued/composite attribute
    - 2NF ⇒ Yes as no partial key is used
    - 3NF ⇒ Yes as no transitive dependency
    - BCNF ⇒ Yes as all determinants are super key
- ➢ This is lossless join decomposition as the common attribute event_id in both tables is CK for event_details.

6. Event_Manager

- Event_Manager(<u>manager_id</u>, event_id, date_birth, manager_name, manager_contatc_no, age, working_hours, salary, club_id)
- PK is section_id
- FK is club_id referenced from club

Constraints:
- **Club** relations are **many to one** so there may be many sections to a single club.
- **Club** section must participate in the **club** so FK club_id can't be empty.
- We choose **NO ACTION** as default on performing **updation or deletion**, because it might affect club relation as we have borrowed club_id as a FK.

❖ Functional dependency:

- ➢ manager_id→(event_id, date_birth, manager_name, manager_contact_no, age, working_hours, salary, club_id)
- ➢ 1NF ⇒ Yes as no multivalued/composite attribute
- ➢ 2NF ⇒ Yes as no partial key is used
- ➢ 3NF ⇒ Yes as no transitive dependency
- ➢ BCNF ⇒ Yes as all determinants are super key

7. Sponser

- Sponser(sponser_id, sponser_name, sponser_email, sponser_contact_no)
- PK is section_id
- FK is club_id referenced from club

Constraints:

- **Club** relations are **many to one** so there may be many sections to a single club.
- **Club** section must participate in the **club** so FK club_id can't be empty.
- We choose **NO ACTION** as default on performing **updation or deletion**, because it might affect club relation as we have borrowed club_id as a FK.

❖ Functional dependency:

- ➢ sponser_id →(sponser_name, sponser_email, sponser_contact_no)
- ➢ 1NF ⇒ Yes as no multivalued/composite attribute
- ➢ 2NF ⇒ Yes as no partial key is used
- ➢ 3NF ⇒ Yes as no transitive dependency
- ➢ BCNF ⇒ Yes as all determinants are super key

8. Transaction History

- Transaction_History(transaction_id, Debit_card_id, time, amount, club_id)
- PK is transaction_id

Constraints:

- **Club** relations are **many to one** so there may be many sections to a single club.
- **Club** section must participate in the **club** so FK club_id can't be empty.
- We choose **NO ACTION** as default on performing **updation or deletion**, because it might affect club relation as we have borrowed club_id as a FK.

❖ Functional dependency:

- ➢ transaction_id →(Debit_card_id, time, amount, club_id)
- ➢ 1NF ⇒ Yes as no multivalued/composite attribute

- ➢ 2NF ⇒ Yes as no partial key is used
- ➢ 3NF ⇒ Yes as no transitive dependency
- ➢ BCNF ⇒ Yes as all determinants are super key

9. Member

- member(member_id, member_name, member_email, member_id_proof, debit_card_id, event_id, member_number, member_dob)
- PK is member_id

Constraints:

- **Club** relations are **many to one** so there may be many sections to a single club.
- **Club** section must participate in the **club** so FK club_id can't be empty.
- We choose **NO ACTION** as default on performing **updation or deletion**, because it might affect club relation as we have borrowed club_id as a FK.

❖ Functional dependency:

- ➢ member_id →(member_name, member_email, member_id_proof, debit_card_id, event_id, member_number, member_dob)
- ➢ 1NF ⇒ No as multivalued attribute event_id

❖ Decomposition into

- ➢ member(member_id, member_name, member_email, member_id_proof, debit_card_id, member_number, member_dob)
  - ■ Functional dependency:
    - member_id →(member_name, member_email, member_id_proof, debit_card_id, member_number, member_dob)
    - 1NF ⇒ Yes as no multivalued/composite attribute
    - 2NF ⇒ Yes as no partial key is used
    - 3NF ⇒ Yes as no transitive dependency
    - BCNF ⇒ Yes as all determinants are super key

- ➢ event_member(event_id, member_id)
  - ■ Functional dependency:
    - (event_id, member_id) → (event_id, member_id)
    - 1NF ⇒ Yes as no multivalued/composite attribute
    - 2NF ⇒ Yes as no partial key is used
    - 3NF ⇒ Yes as no transitive dependency

Final relations

1. Club ⇒ (**club_id**, club_name, club_email)
2. club_contact ⇒ (**club_id,club_contact**)
3. club_address ⇒ (**club_id**, pincode, city_name, state, area)

4. owner ⇒ (**owner_id**, club_id, owner_name, owner_number, owner_email)

5. Section_title⇒ (**Section_id**, section_name)
6. section_days ⇒ (**section_id**, **club_id**, active_days)

7. employee ⇒ (**employee_id**, name, age, working_hours, salary, section_id,club_id)

8. event_manager ⇒ (**manager_id**, date_birth, manager_name, manager_email, working_hours, salary, club_id)

9. event_details ⇒ (**event_id**, event_duration, event_name, event_date, event_theme, event_capacity, event_eligibity, manager_id)

10. event_sponser ⇒ (**sponsor_id**, contribution, sponser_name, sponser_email, sponser_contact_no)
11. sponser_with_event ⇒ (**event_id,sponser_id**)

12. Member ⇒ (**member_id**, member_name, member_contact,member_email, member_id_proof, debit_card_id, member_dob)
13. member_with_event ⇒ (**member_id**,**event_id**)
14. Member_with_club ⇒ (**member_id**, **club_id**,membership_type, no_event_attended, member_penalty, money_spent)

2. Create DDL Scripts :

Create Table

1. Club ⇒ (**club_id**, club_name, club_email)

```
CREATE TABLE IF NOT EXISTS clubbing."club"
(
    club_id integer NOT NULL,
    club_name character varying COLLATE pg_catalog."default",
    club_email character varying COLLATE pg_catalog."default",
        CONSTRAINT club_pkey PRIMARY KEY (club_id)

)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."club"
    OWNER to postgres;
```

2. club_contact ⇒ (**club_id**,**contact_number**)

```
CREATE TABLE IF NOT EXISTS clubbing."club_contact"
(
    club_id integer NOT NULL,
    club_contact int Not NULL,
        CONSTRAINT club_contact_pkey PRIMARY KEY (club_id,club_contact),
    CONSTRAINT contacting FOREIGN KEY (club_id)
        REFERENCES clubbing."club" (club_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."club_contact"
    OWNER to postgres;
```

3. club_address ⇒ (**club_id**, pincode, city_name, state, area)

```
CREATE TABLE IF NOT EXISTS clubbing."club_address"
(
    club_id integer NOT NULL,
    pincode int Not NULL,
        city_name character varying COLLATE pg_catalog."default",
        stat character varying COLLATE pg_catalog."default" Not NULL,
        area character varying COLLATE pg_catalog."default",
        CONSTRAINT club_address_pkey PRIMARY KEY (club_id),
    CONSTRAINT addressing FOREIGN KEY (club_id)
      REFERENCES clubbing."club" (club_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."club_address"
    OWNER to postgres;
```

4. owner ⇒ (**owner_id**, club_id, owner_name, owner_number, owner_email)

```
CREATE TABLE IF NOT EXISTS clubbing."owner"
(
    owner_id integer NOT NULL,
    owner_name character varying COLLATE pg_catalog."default",
    owner_number bigint,
   owner_email character varying COLLATE pg_catalog."default",
   club_id integer Not NULL,
    CONSTRAINT owner_pkey PRIMARY KEY (owner_id),
        CONSTRAINT owner_manage FOREIGN KEY (club_id)
      REFERENCES clubbing."club" (club_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."owner"
    OWNER to postgres;
```

5. Section_title⇒ (**Section_id**, section_name)

```
CREATE TABLE IF NOT EXISTS clubbing."section_title"
(
    section_id INTEGER NOT NULL,
        section_name CHARACTER VARYING collate pg_catalog."default" not
NULL,
    CONSTRAINT section_title_pkey PRIMARY KEY (section_id)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."section_title"
    OWNER to postgres;
```

6. section_days ⇒ (**section_id**, **club_id**, active_days)

```
CREATE TABLE IF NOT EXISTS clubbing."section_days"
(
    section_id INTEGER NOT NULL,
    club_id INTEGER NOT NULL,
        active_days CHARACTER VARYING collate pg_catalog."default",
    CONSTRAINT section_days_pkey PRIMARY KEY (section_id,club_id),
        CONSTRAINT depart_sec FOREIGN KEY (club_id)
        REFERENCES clubbing."club" (club_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
        CONSTRAINT club_section FOREIGN KEY (section_id)
        REFERENCES clubbing."section_title" (section_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."section_days"
    OWNER to postgres;
```

7. employee ⇒ (**employee_id**, name, age, working_hours, salary, section_id,club_id)

```
CREATE TABLE IF NOT EXISTS clubbing."employee"
(
    employee_id integer NOT NULL,
    name character varying COLLATE pg_catalog."default",
    age int,
        working_hours int,
        salary int,
        section_id int not NULL,
        club_id int not NULL,
    CONSTRAINT employee_pkey PRIMARY KEY (employee_id),
        CONSTRAINT section_employee FOREIGN KEY (section_id)
      REFERENCES clubbing."section_title" (section_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
        CONSTRAINT club_employee FOREIGN KEY (club_id)
      REFERENCES clubbing."club" (club_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."employee"
    OWNER to postgres;
```

8. event_manager ⇒ (**manager_id**, date_birth, manager_name, manager_email, age, working_hours, salary, club_id)

```
CREATE TABLE IF NOT EXISTS clubbing."event_manager"
(
    manager_id integer NOT NULL,
        date_birth date,
        manager_name character varying COLLATE pg_catalog."default",
        member_email character varying COLLATE pg_catalog."default",
    working_hours character varying COLLATE pg_catalog."default",
        salary int,
        club_id int not NULL,

        CONSTRAINT club_manages_pkey PRIMARY KEY (manager_id),
    CONSTRAINT contacting FOREIGN KEY (club_id)
      REFERENCES clubbing."club" (club_id) MATCH SIMPLE
      ON UPDATE NO ACTION
```

```
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."event_manager"
    OWNER to postgres;
```

9. event_details ⇒ (**event_id**, event_duration, event_name, event_date, event_theme, event_capacity, event_eligibity, manager_id)

```
 CREATE TABLE IF NOT EXISTS clubbing."event_details"
(
    event_id integer NOT NULL,
        event_duration int ,
    event_name character varying COLLATE pg_catalog."default",
    event_date int,
        event_theme character varying COLLATE pg_catalog."default",
        event_capacity int ,
        event_eligibility int,
        manager_id int not NULL,
        CONSTRAINT event_details_pkey PRIMARY KEY (event_id),
        CONSTRAINT eventatclub_manager FOREIGN KEY (manager_id)
      REFERENCES clubbing."event_manager" (manager_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION

)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."event_details"
    OWNER to postgres;
```

10. event_sponser ⇒ (**sponsor_id**, sponser_name, sponser_contact_no, sponser_email, contribution)

```
 CREATE TABLE IF NOT EXISTS clubbing."event_sponser"
(
    sponser_id integer NOT NULL,
```

```
  sponser_name character varying COLLATE pg_catalog."default",
   sponser_contact_no bigint,
   sponser_email character varying COLLATE pg_catalog."default",
        contribution int not NULL,
        CONSTRAINT event_sponser_pkey PRIMARY KEY (sponser_id)
)
WITH (
   OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."event_sponser"
   OWNER to postgres;
```

11. sponser_with_event ⇒ (**event_id,sponser_id**)

```
CREATE TABLE IF NOT EXISTS clubbing."sponser_with_event"
(
   sponser_id integer NOT NULL,
        event_id int not NULL ,
        CONSTRAINT eventdetails_sponser_pkey PRIMARY KEY
(sponser_id,event_id),
        CONSTRAINT from_sponser FOREIGN KEY (sponser_id)
      REFERENCES clubbing."event_sponser" (sponser_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
        CONSTRAINT from_eventdetails FOREIGN KEY (event_id)
      REFERENCES clubbing."event_details" (event_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION

)
WITH (
   OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."sponser_with_event"
   OWNER to postgres;
```

12. Member ⇒ (**member_id**, member_name, member_contact,member_email, member_id_proof, member_dob)

```
 CREATE TABLE IF NOT EXISTS clubbing."member"
(
    member_id integer NOT NULL,
        member_name character varying COLLATE pg_catalog."default",
    member_contact bigint,
    member_email character varying COLLATE pg_catalog."default",
        member_id_proof int not NULL,
        member_dob date,
        CONSTRAINT member_pkey PRIMARY KEY (member_id)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."member"
    OWNER to postgres;
```

13. member_with_event ⇒ (**member_id**,**event_id**)

```
CREATE TABLE IF NOT EXISTS clubbing."member_with_event"
(
    member_id integer NOT NULL,
        event_id int not NULL ,
        CONSTRAINT member_with_event_pkey PRIMARY KEY
(member_id,event_id),
        CONSTRAINT from_member FOREIGN KEY (member_id)
      REFERENCES clubbing."member" (member_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
        CONSTRAINT from_eventdetails FOREIGN KEY (event_id)
      REFERENCES clubbing."event_details" (event_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."member_with_event"
    OWNER to postgres;
```

14. member_with_club ⇒ (**member_id**, **club_id**)

```
CREATE TABLE IF NOT EXISTS clubbing."member_with_club"
(
    member_id integer NOT NULL,
        club_id int not NULL ,
        membership_type character varying COLLATE pg_catalog."default" not NULL,
        no_event_attended int,
        member_penalty int not null,
        money_spent int,
        CONSTRAINT member_with_club_pkey PRIMARY KEY (member_id,club_id),
        CONSTRAINT from_member FOREIGN KEY (member_id)
      REFERENCES clubbing."member" (member_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
        CONSTRAINT from_club FOREIGN KEY (club_id)
      REFERENCES clubbing."club" (club_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing."member_with_club"
    OWNER to postgres;
```

Import

1. **club**
   COPY clubbing.club(club_id, club_name, club_email) FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\clubs.csv' DELIMITER ',' CSV HEADER;

2. **club_contact**
   COPY clubbing.club_contact(club_id, club_contact) FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\club_contact.csv' DELIMITER ',' CSV HEADER;

3. **club_address**
   COPY clubbing.club_address FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\club_address.csv' DELIMITER ',' CSV HEADER;

4. **owner**

COPY clubbing.owner FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\owner.csv' DELIMITER ',' CSV HEADER;

5. **Section_title**
   COPY clubbing.section_title FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\section_title.csv' DELIMITER ',' CSV HEADER;

6. **section_days**
   COPY clubbing.section_days FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\section_days.csv' DELIMITER ',' CSV HEADER;

7. **employee**
   COPY clubbing.employee FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\employee.csv' DELIMITER ',' CSV HEADER;

8. **event_manager**
   COPY clubbing.event_manager FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\event_manager.csv' DELIMITER ',' CSV HEADER;

9. **event_details**
   COPY clubbing.event_details FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\event_details.csv' DELIMITER ',' CSV HEADER;

10. **event_sponser**
    COPY clubbing.event_sponser FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\event_sponser.csv' DELIMITER ',' CSV HEADER;

11. **sponser_with_event**
    COPY clubbing.sponser_with_event FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\sponser_with_event.csv' DELIMITER ',' CSV HEADER;

12. **member**
    COPY clubbing.member FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\member.csv' DELIMITER ',' CSV HEADER;

13. **member_with_event**
    COPY clubbing.member_with_event FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\member_with_event.csv' DELIMITER ',' CSV HEADER;

14. **member_with_club**
    COPY clubbing.member_with_club FROM 'E:\Sem 5\Database management system\LAbs\Project\Files\member_with_club.csv' DELIMITER ',' CSV HEADER;

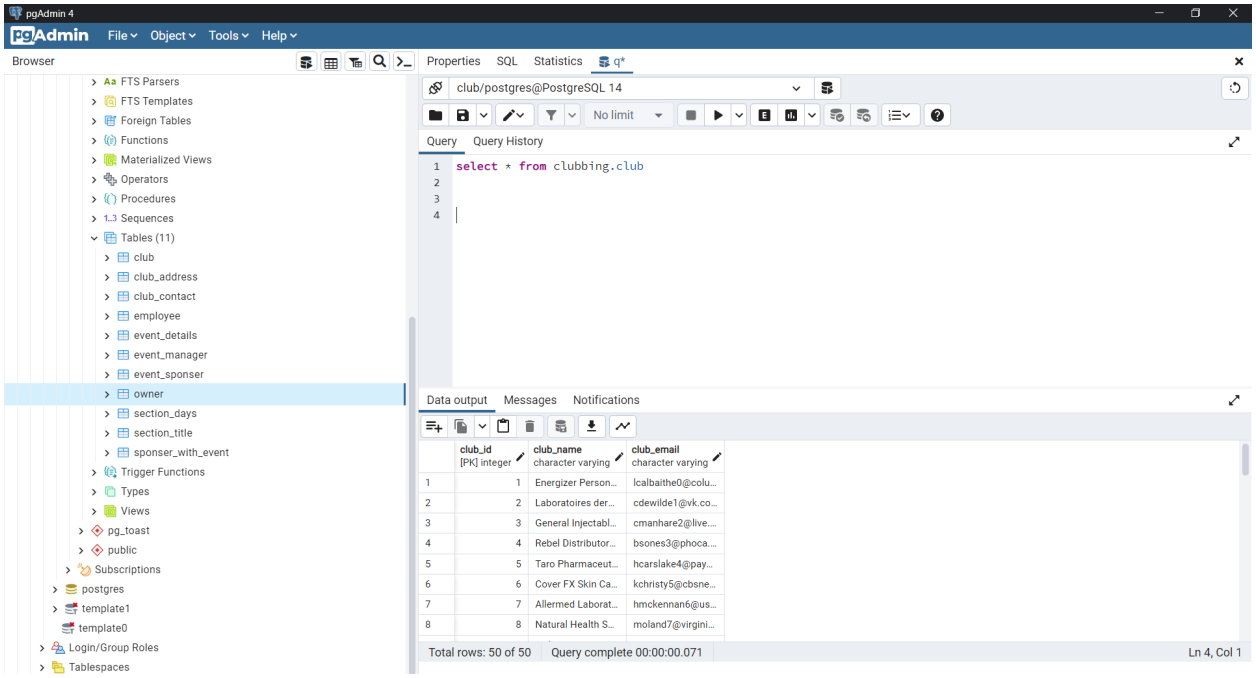
Queries

Formate

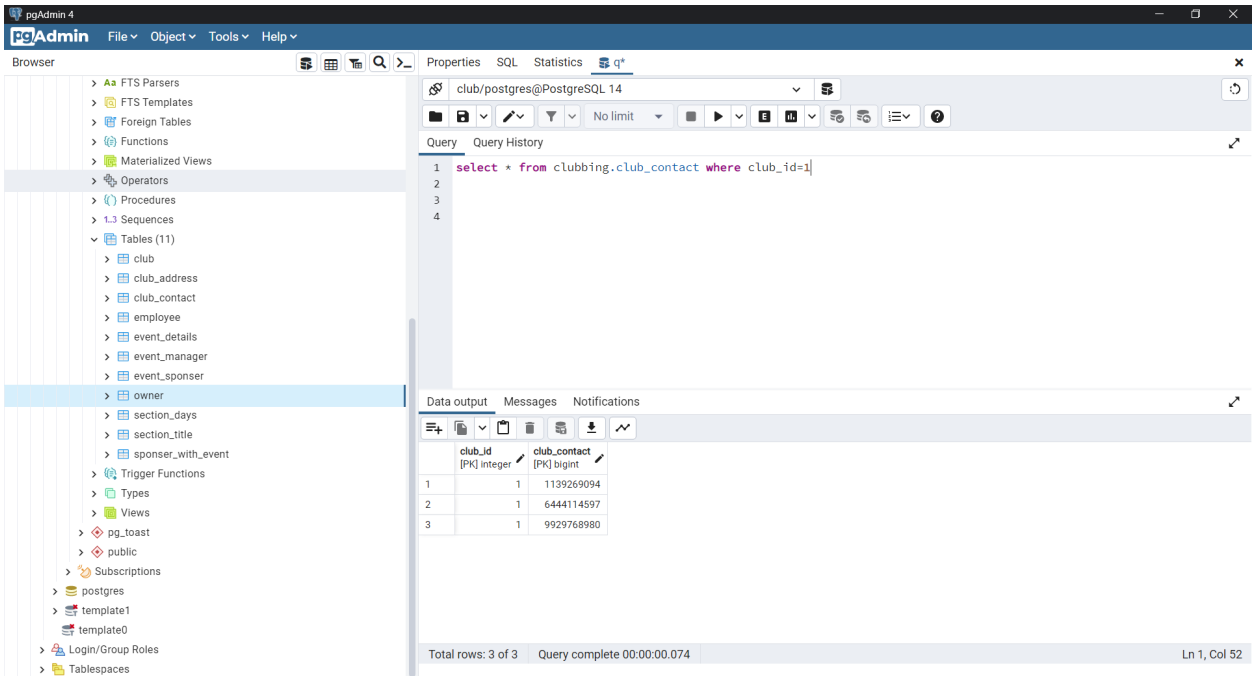
1. SQL Answer for that query.
2. Plain English query.
3. Snapshots of the SQL Queries and their answers ran on Postgres.

Simple Queries

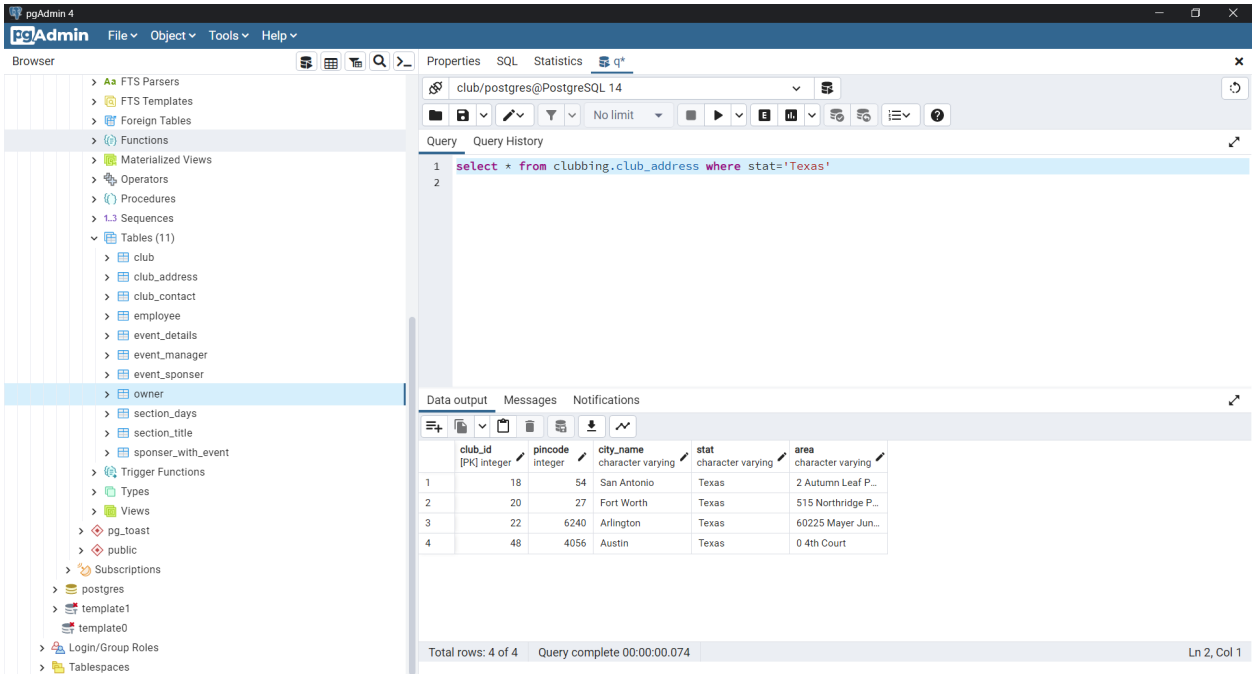
1. find information about all clubs
   select * from clubbing.club



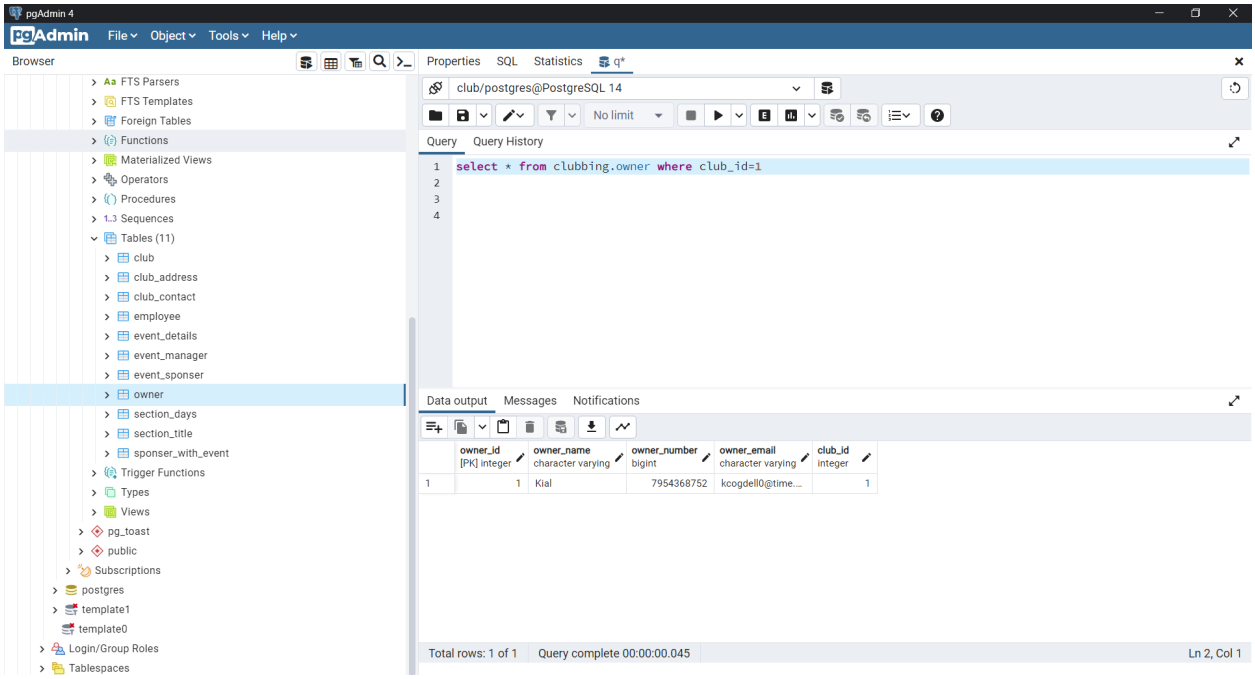
2. find all available contact_no of club_id=1
   select * from clubbing.club_contact where club_id=1



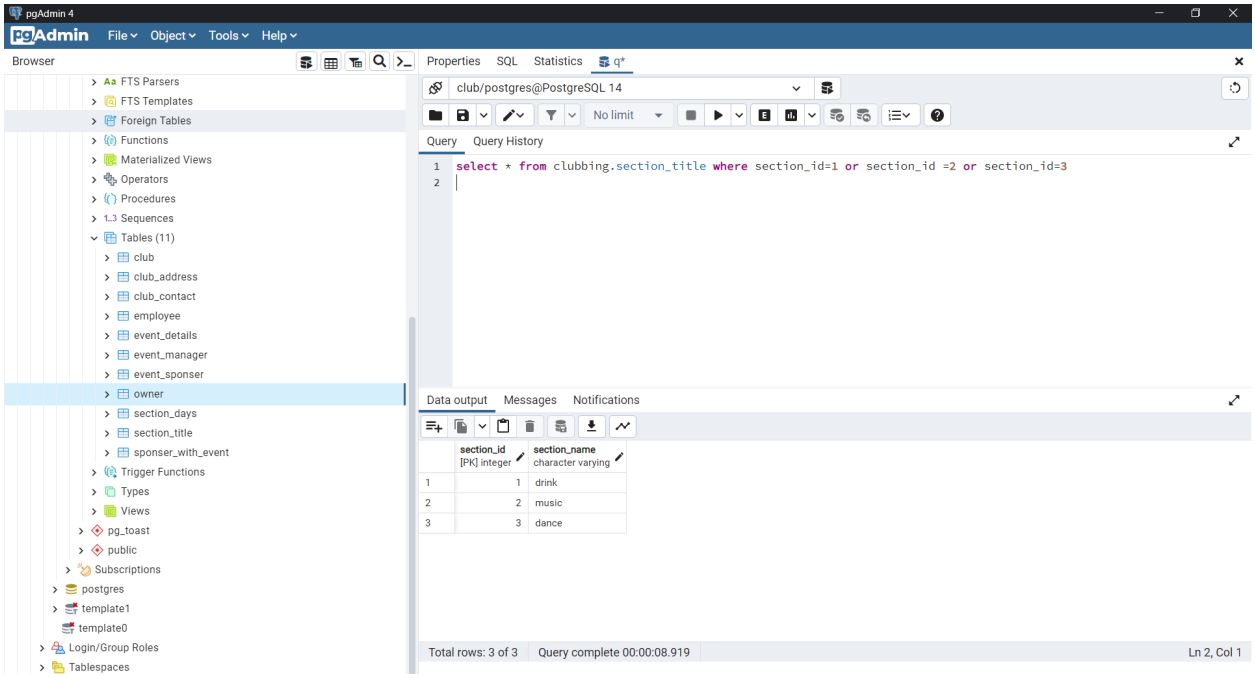
3. find information of clubs which are located in Texas
   select * from clubbing.club_address where stat='Texas'



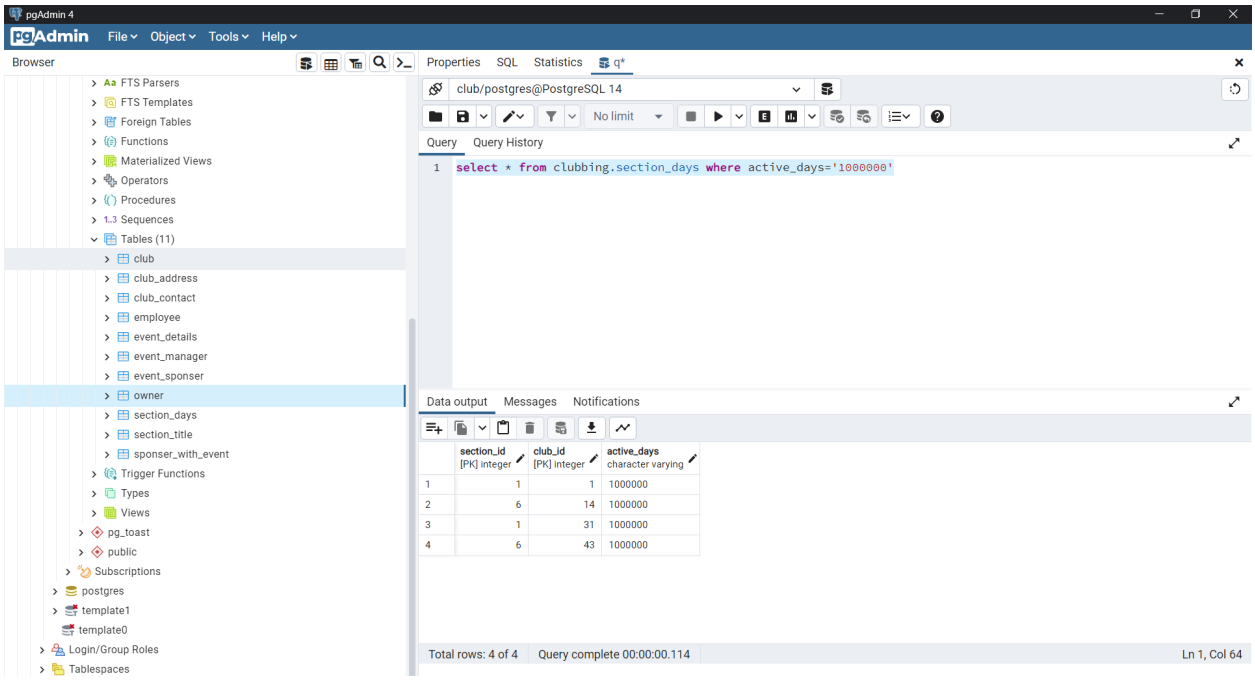
4. find owner information of club_id=1
   select * from clubbing.owner where club_id=1



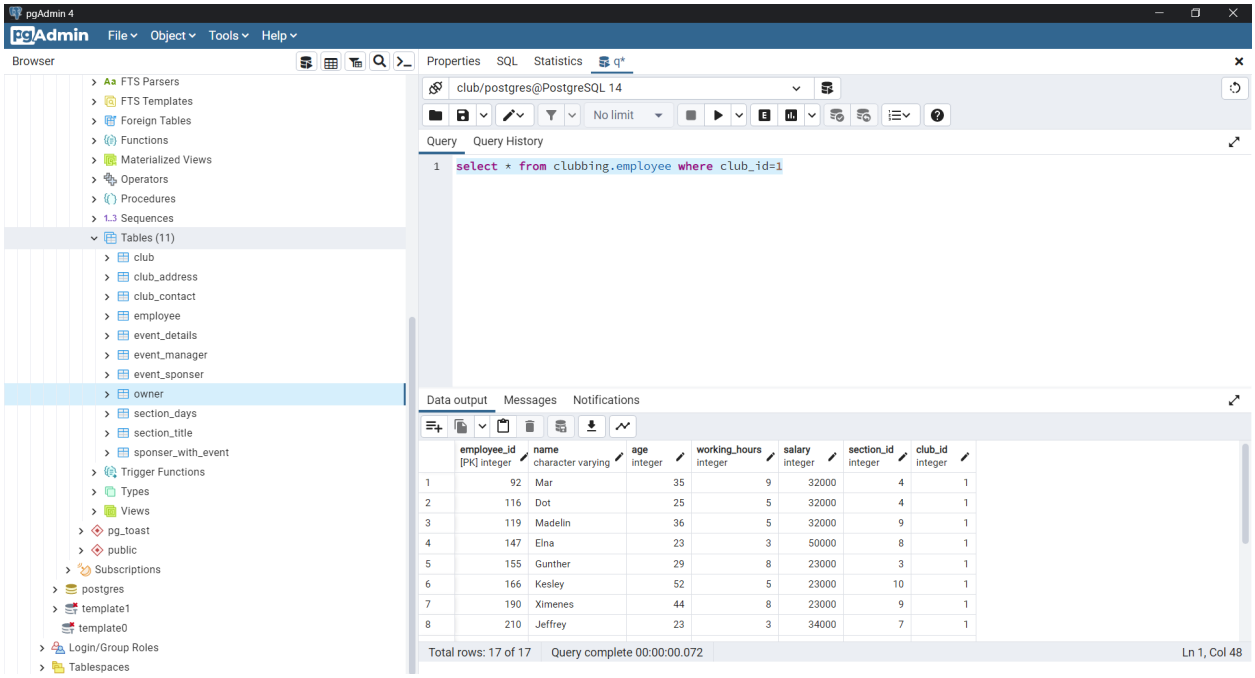
5. find section_name of section_id=1 or 2 or 3
   select * from clubbing.section_title where section_id=1 or section_id =2 or section_id=3



6. find club_id and section_id, which are active on only sunday
   select * from clubbing.section_days where active_days='1000000'

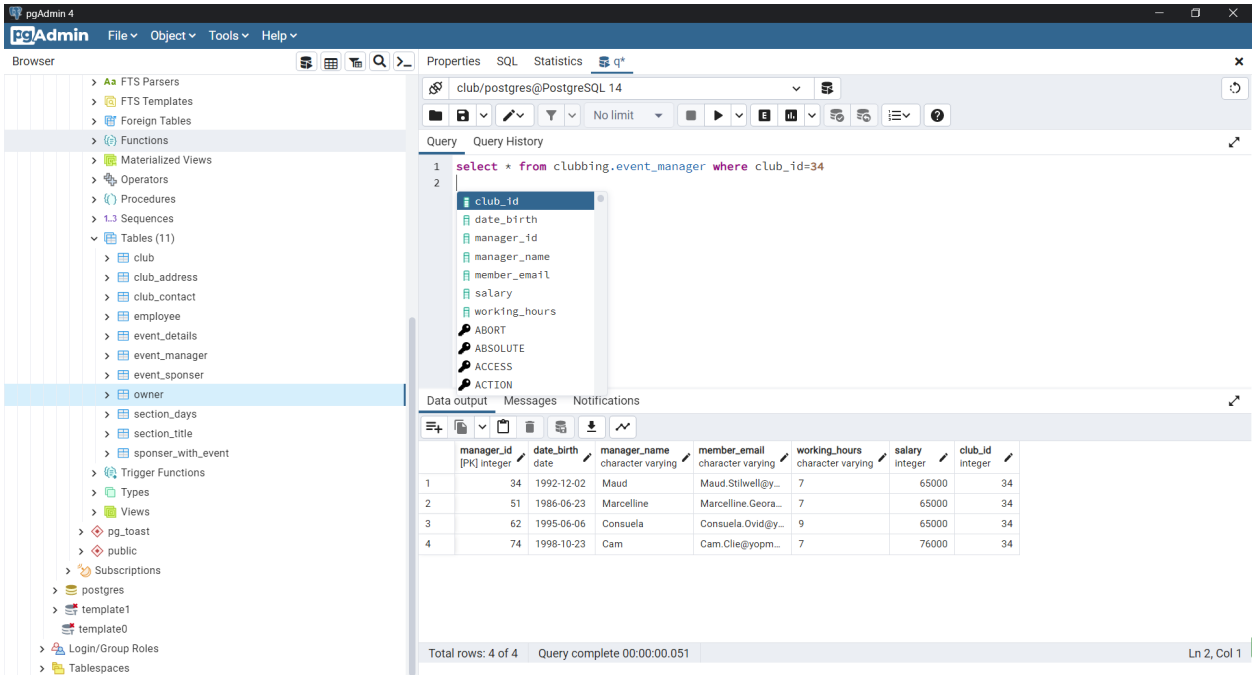


7. find employee information of club_id=1
   select * from clubbing.employee where club_id=1



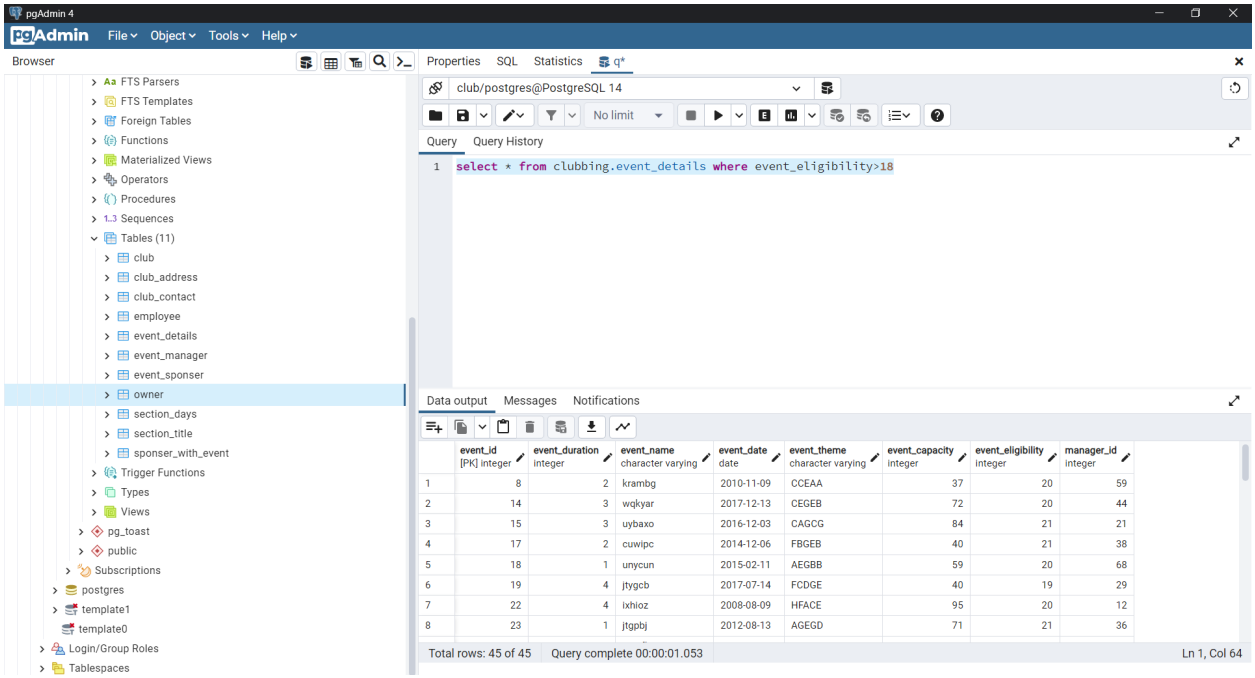
8. find manager details of club_id=34

select * from clubbing.event_manager where club_id=34



9. find details of event which have event_eligibility>18

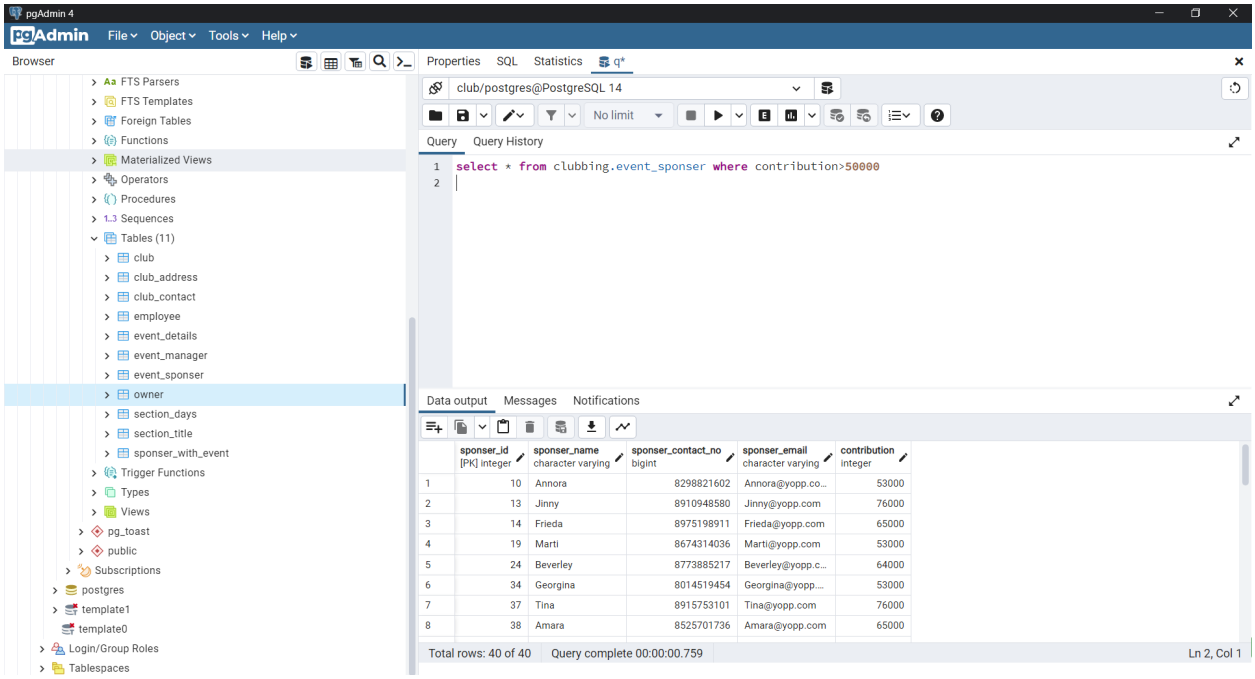
select * from clubbing.event_details where event_eligibility>18



	event_id [PK] integer	event_duration integer	event_name character varying	event_date date	event_theme character varying	event_capacity integer	event_eligibility integer	manager_id integer
1	8	2	krambg	2010-11-09	CCEAA	37	20	59
2	14	3	wqkyar	2017-12-13	CEGEB	72	20	44
3	15	3	uybaxo	2016-12-03	CAGCG	84	21	21
4	17	2	cuwipc	2014-12-06	FBGEB	40	21	38
5	18	1	unycun	2015-02-11	AEGBB	59	20	68
6	19	4	jtygcb	2017-07-14	FCDGE	40	19	29
7	22	4	ixhioz	2008-08-09	HFACE	95	20	12
8	23	1	jtgpbj	2012-08-13	AGEGD	71	21	36

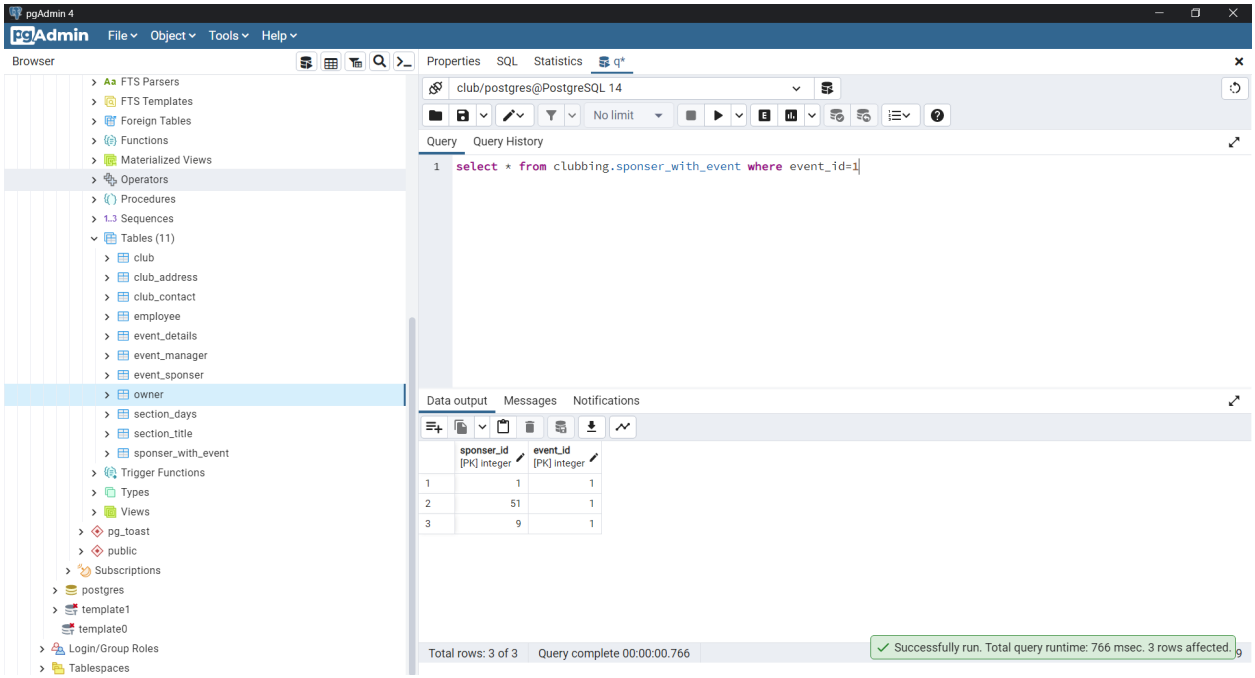
10. find information of event_sponser who have contribution>50000

select * from clubbing.event_sponser where contribution>50000

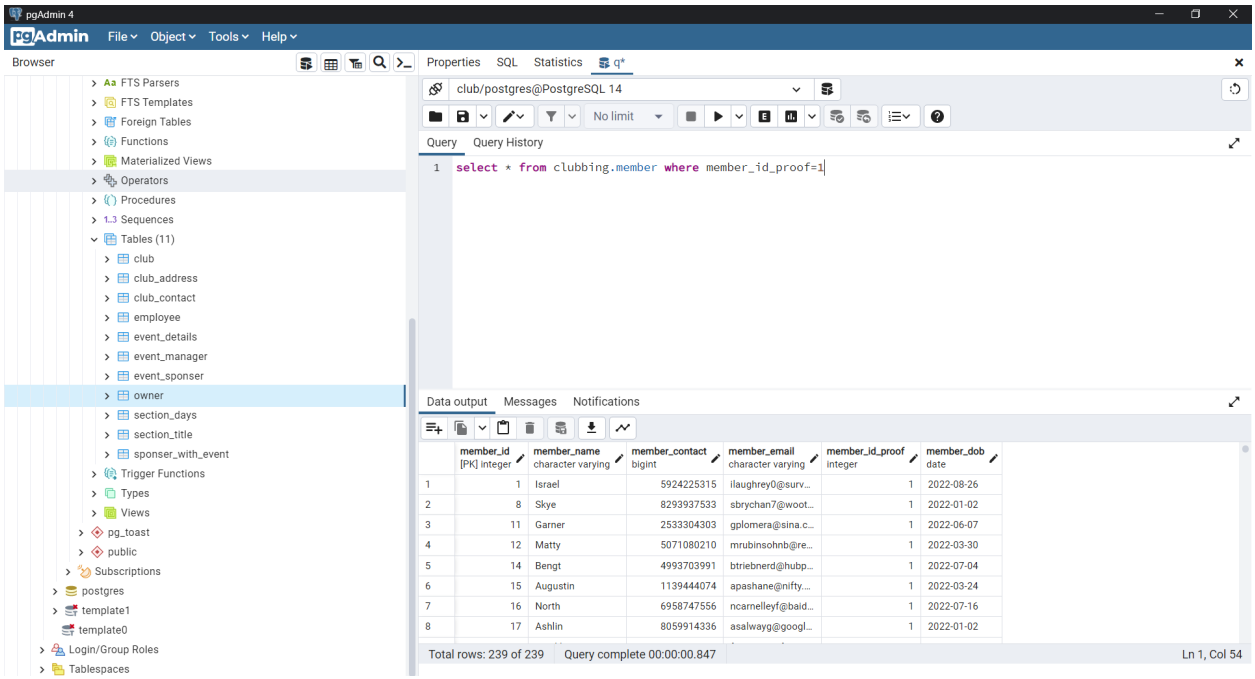


	sponser_id [PK] integer	sponser_name character varying	sponser_contact_no bigint	sponser_email character varying	contribution integer
1	10	Annora	8298821602	Annora@yopp.co...	53000
2	13	Jinny	8910948580	Jinny@yopp.com	76000
3	14	Frieda	8975198911	Frieda@yopp.com	65000
4	19	Marti	8674314036	Marti@yopp.com	53000
5	24	Beverley	8773885217	Beverley@yopp.c...	64000
6	34	Georgina	8014519454	Georgina@yopp....	53000
7	37	Tina	8915753101	Tina@yopp.com	76000
8	38	Amara	8525701736	Amara@yopp.com	65000

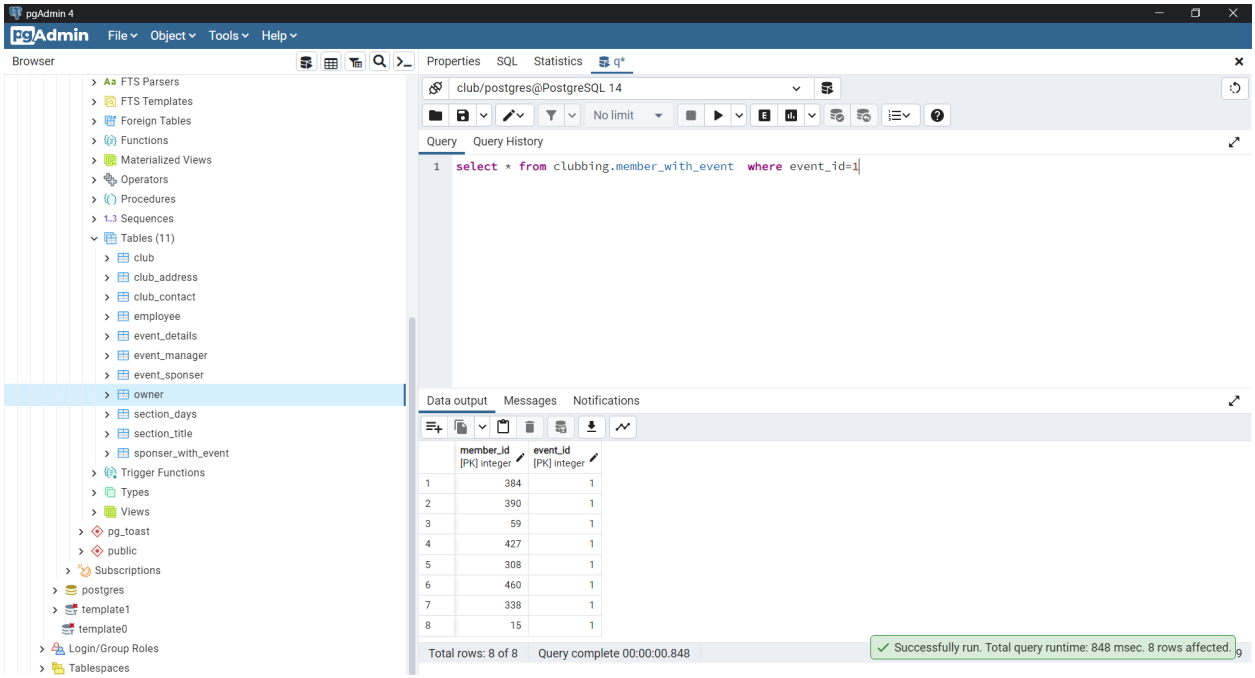
11. find the sponser_id of sponsors of event_id=1
    select * from clubbing.sponser_with_event where event_id=1



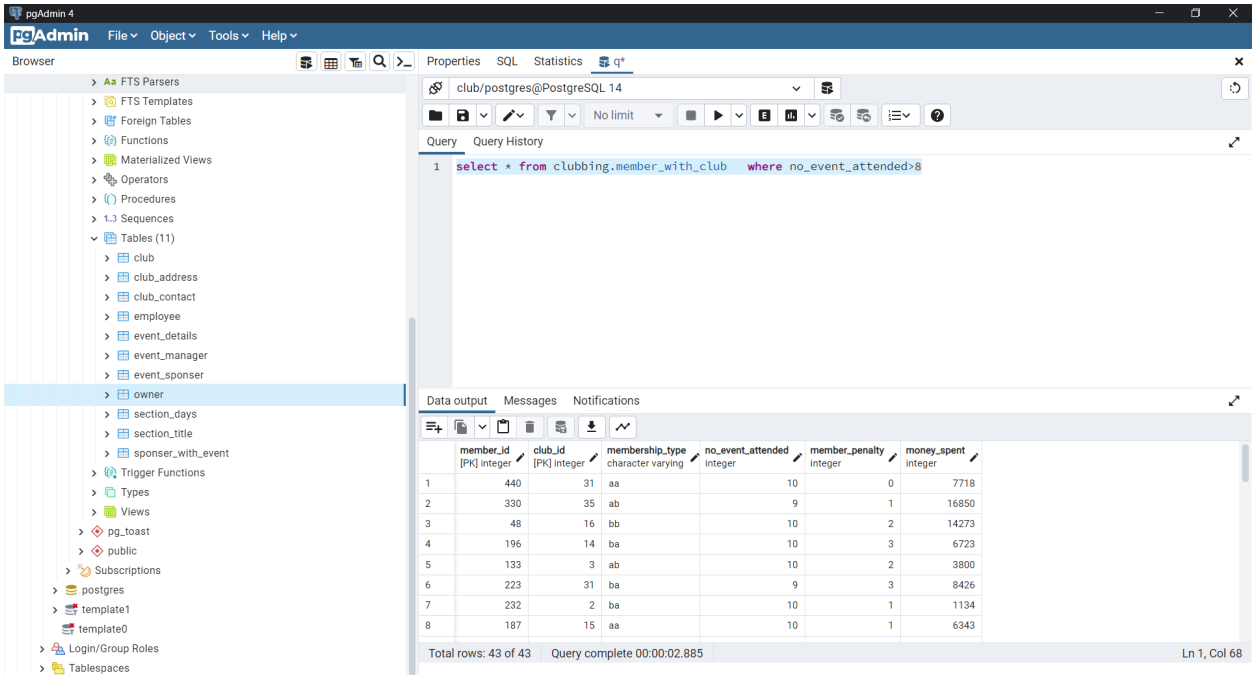
12. find the member list which have attached their id proofs
    select * from clubbing.member where member_id_proof=1



13. find the member who had attended event with event_id=1
    select * from clubbing.member_with_event where event_id=1

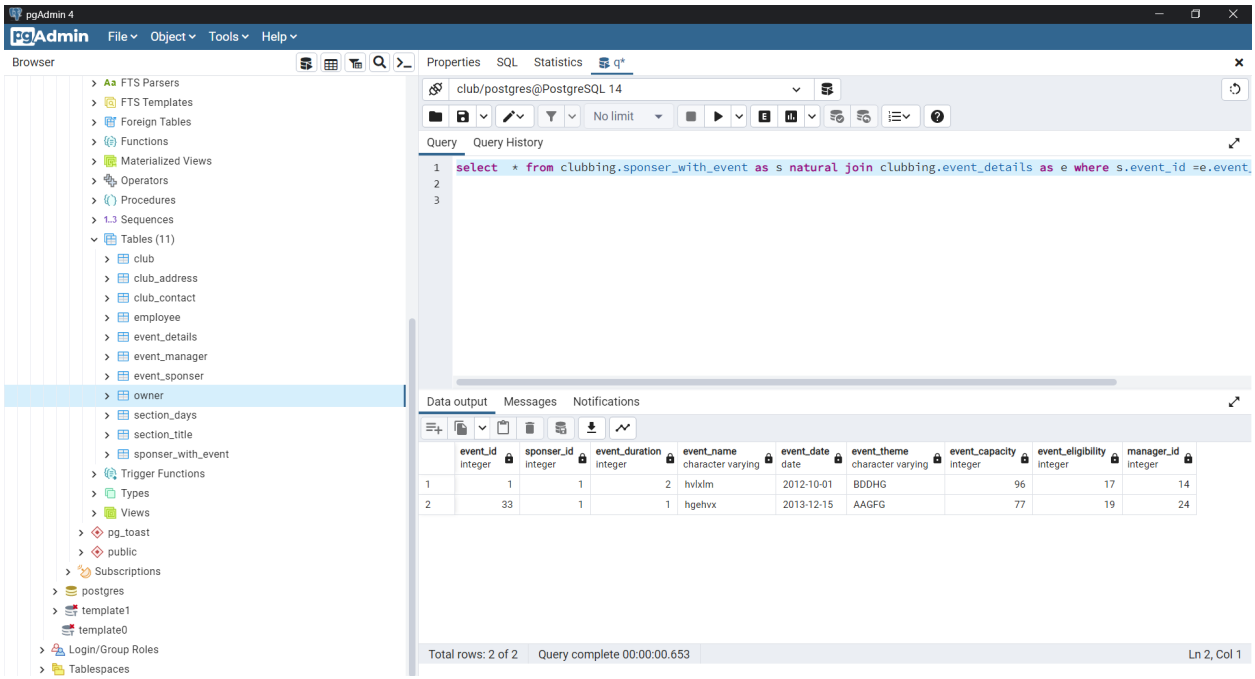


14. find the member details with no_event_attended>8
    select * from clubbing.member_with_club where no_event_attended>8

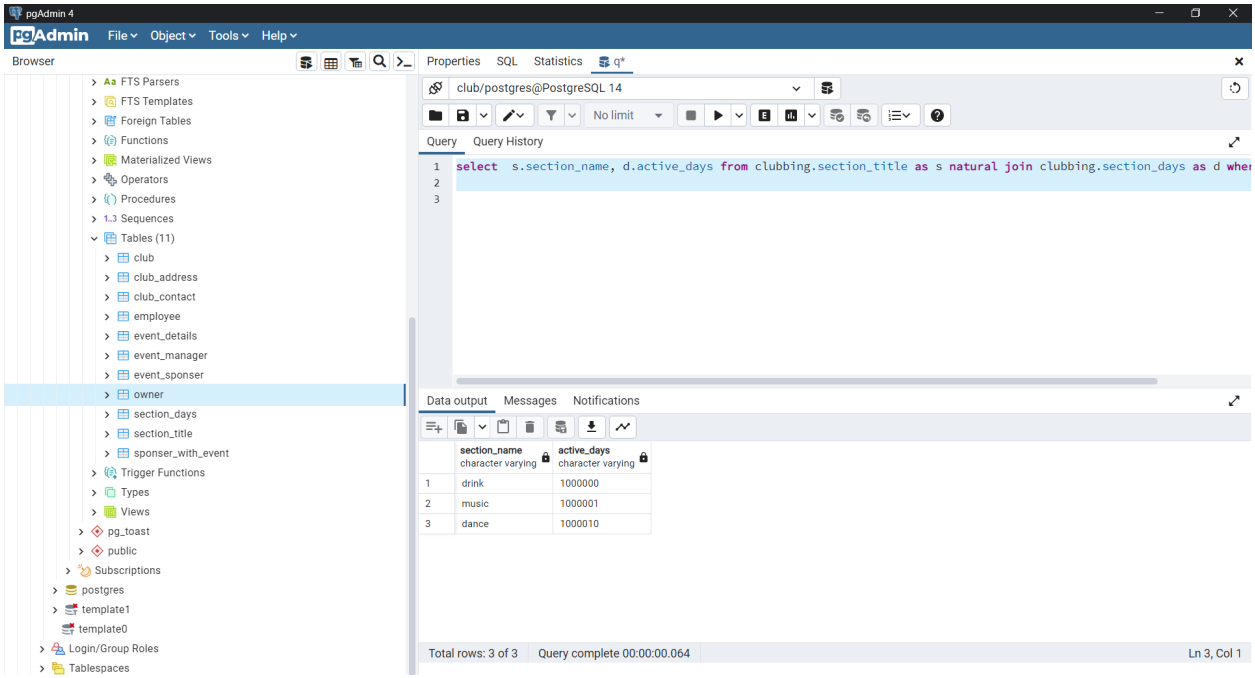


Complex Queries

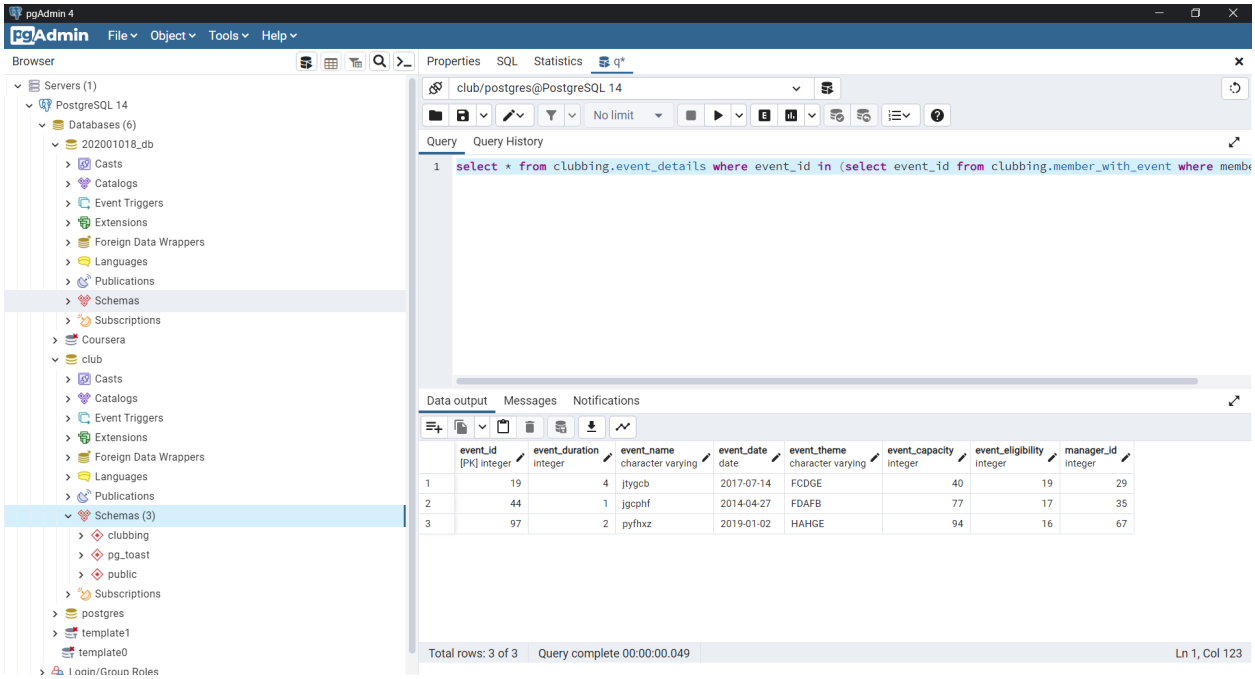
1. find the event details which is sponsored by sponsor with sponser_id=1
   select * from clubbing.sponser_with_event as s natural join clubbing.event_details as e where s.sponser_id=1



2. Find active_days and available section_name of club with club_id=1
   select s.section_name, d.active_days from clubbing.section_title as s natural join clubbing.section_days as d where d.club_id=1

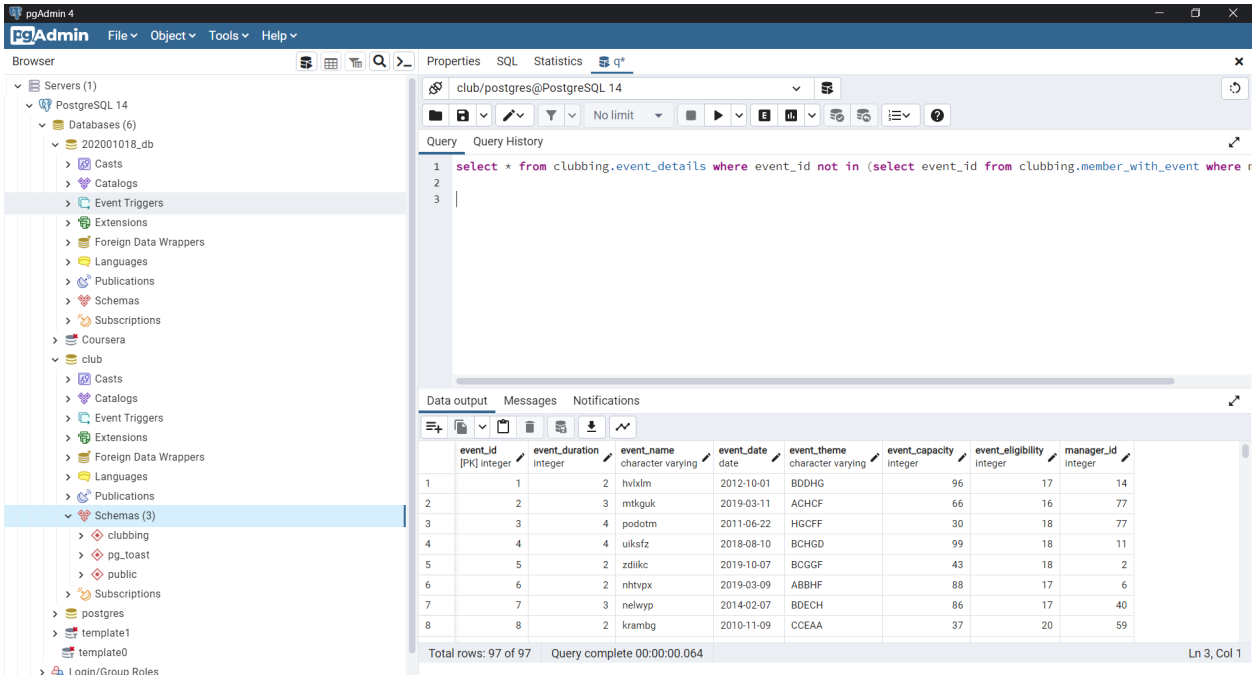


3. Find the all details of event, which events are attended by member with member_id=3
   select * from clubbing.event_details where event_id in (select event_id from clubbing.member_with_event where member_id=3)



4. Find the all details of event, which events are not attended by member with member_id=3

select * from clubbing.event_details where event_id not in (select event_id from clubbing.member_with_event where member_id=3)



5. Create a trigger on the table of event_details to check if the primary key id already exists or not before inserting a new record and send a custom reply instead of an error message(trigger)

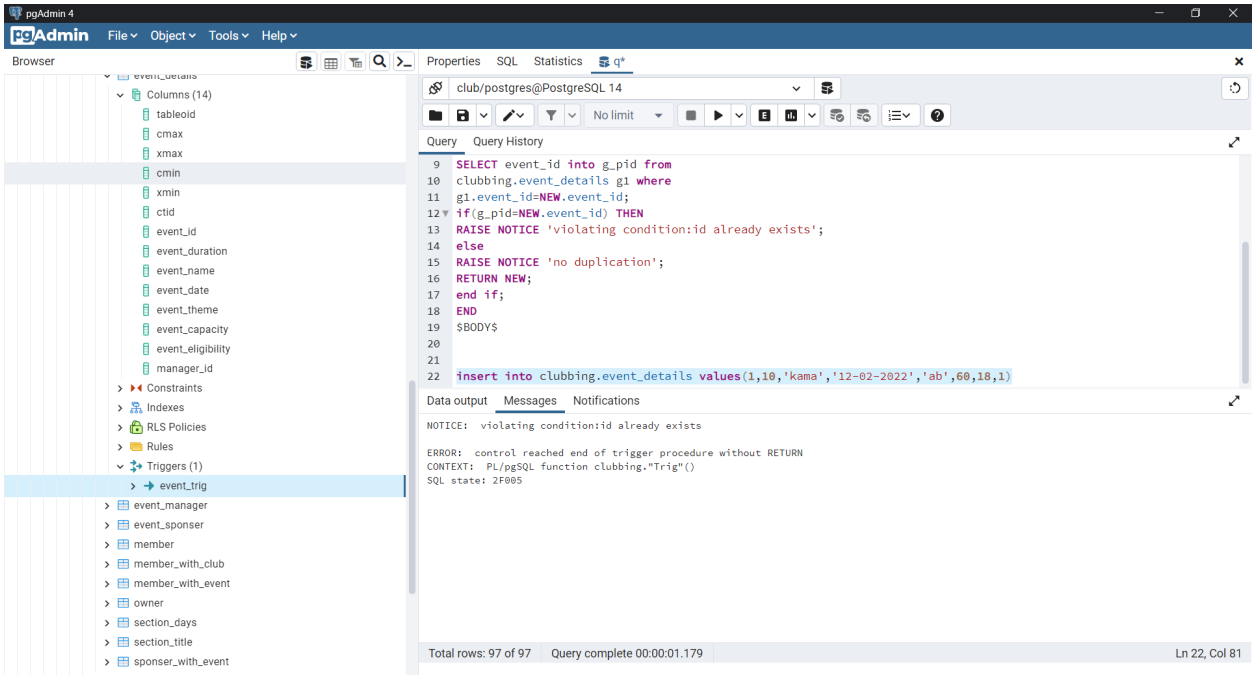
Trigger

```
CREATE OR REPLACE FUNCTION
clubbing."Trig"()
RETURNS TRIGGER
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
g_pid INTEGER;
BEGIN
SELECT event_id into g_pid from
clubbing.event_details g1 where
g1.event_id=NEW.event_id;
if(g_pid=NEW.event_id) THEN
RAISE NOTICE 'violating condition:id already exists';
else
RAISE NOTICE 'no duplication';
RETURN NEW;
end if;
END
```

$BODY$

Insert problem:

insert into clubbing.event_details values(1,10,'kama','12-02-2022','ab',60,18,1)

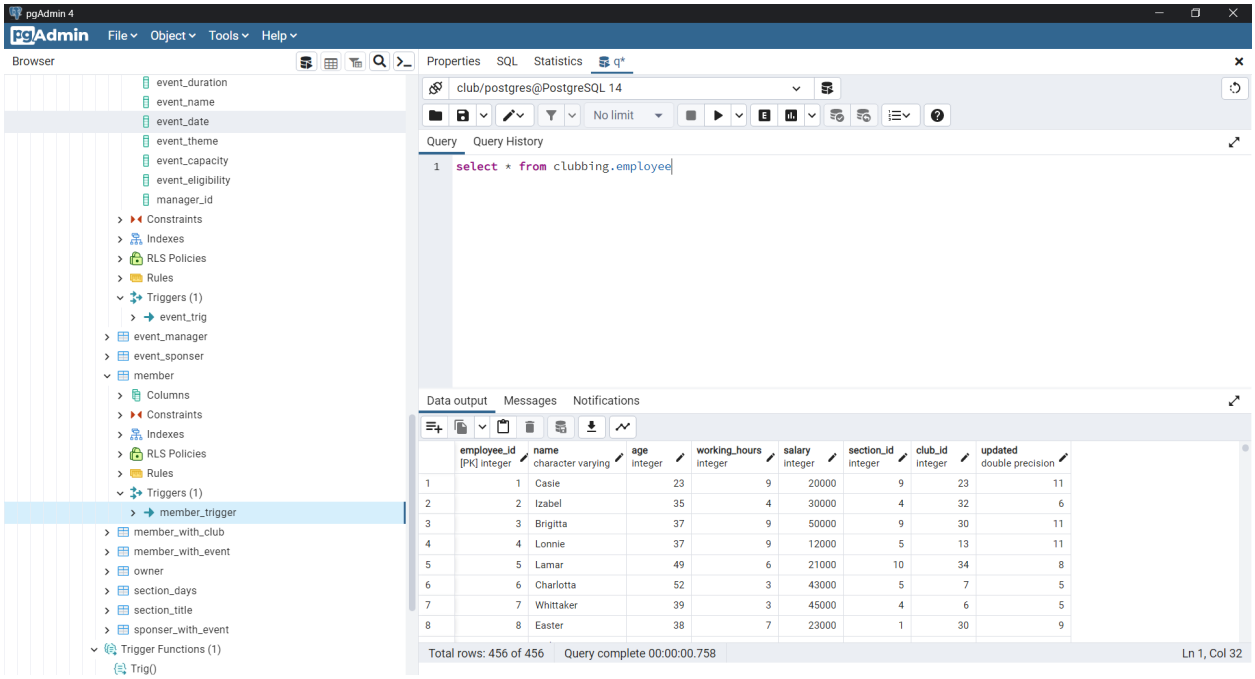


6. Add new column in employee table, which have updated working_hours (increase working hours by 2)
   ALTER table clubbing.employee
   add updated double precision

   CREATE OR REPLACE function clubbing.fun()
   RETURNS TABLE (a double precision)
   LANGUAGE 'plpgsql'
   AS $BODY$
   BEGIN
   update clubbing.employee set updated=working_hours+(2);
   RETURN QUERY EXECUTE format ('SELECT updated FROM clubbing.employee');
   END;
   $BODY$;

   select clubbing.fun()
   select * from clubbing.employee

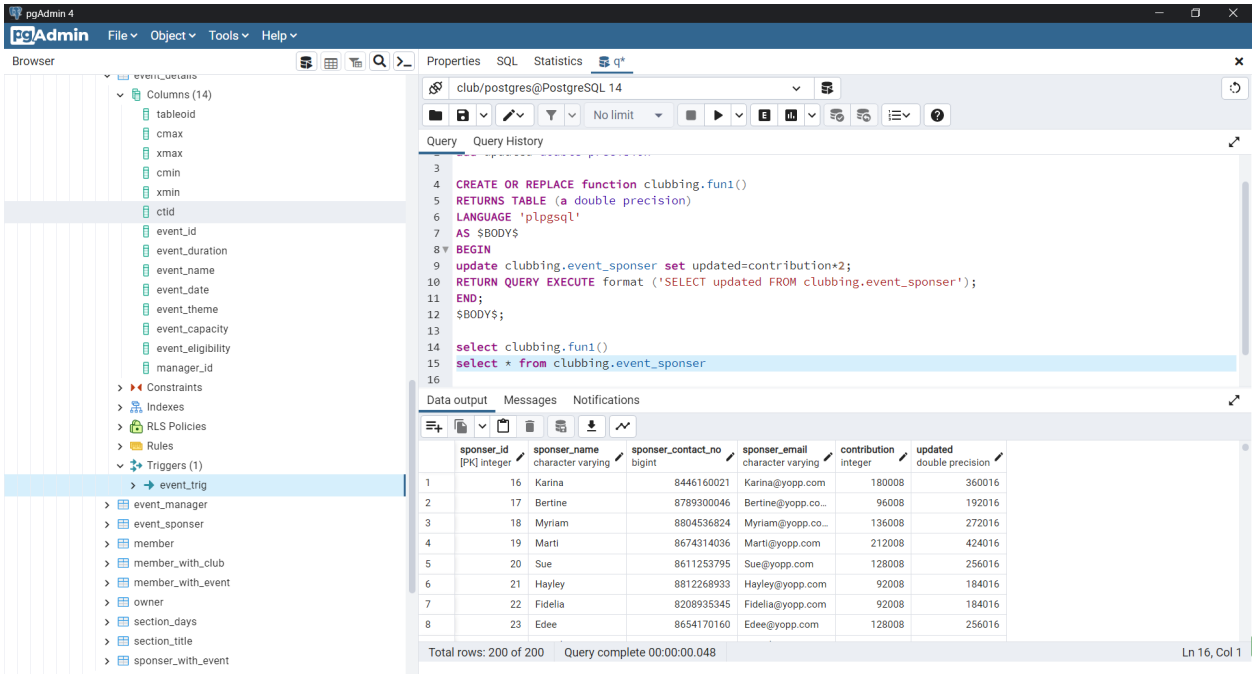
   --tables
   select * from clubbing.employee



7. Create a function which will add a new column in event_sponser table which will have an updated contribution column.
   Function
   ALTER table clubbing.event_sponser
   add updated double precision

   CREATE OR REPLACE function clubbing.fun1()
   RETURNS TABLE (a double precision)
   LANGUAGE 'plpgsql'
   AS $BODY$
   BEGIN
   update clubbing.event_sponser set updated=contribution*2;
   RETURN QUERY EXECUTE format ('SELECT updated FROM clubbing.event_sponser');
   END;
   $BODY$;

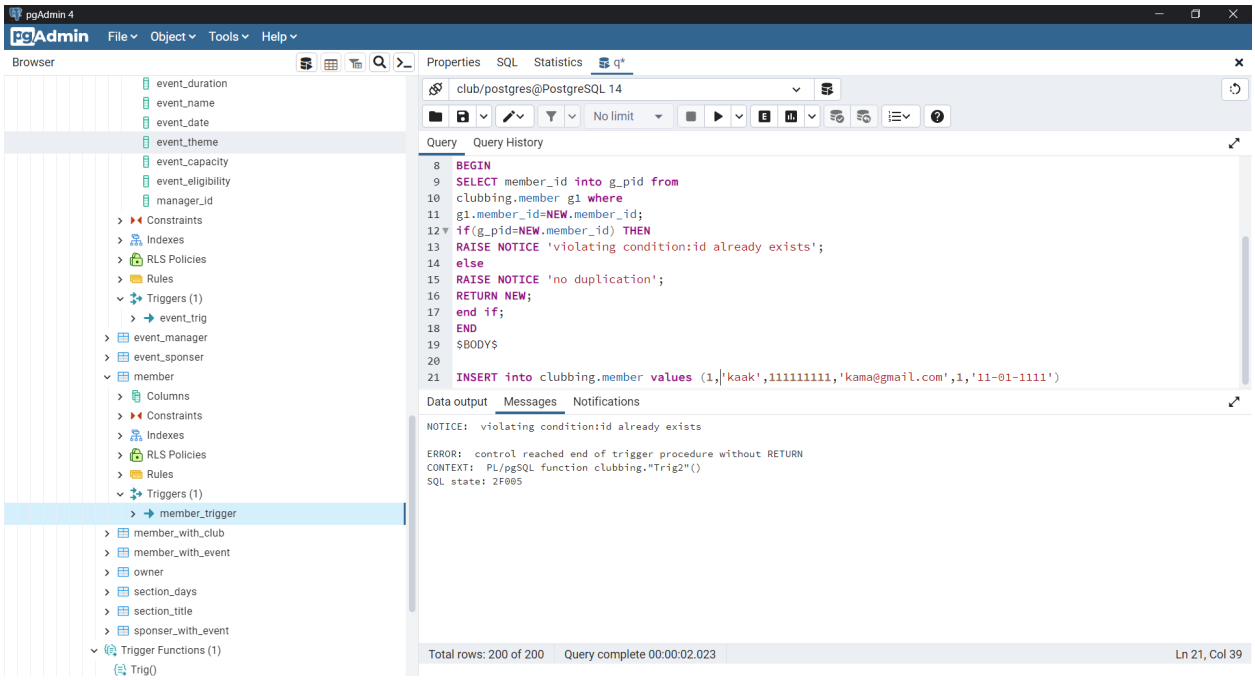
   select clubbing.fun1()
   select * from clubbing.event_sponser



8. Create a trigger on the table of member to check if the primary key id already exists or not before inserting a new record and send a custom reply instead of an error message(trigger)

Insertion problem
CREATE OR REPLACE FUNCTION
clubbing."Trig2"()
RETURNS TRIGGER
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
g_pid INTEGER;
BEGIN
SELECT member_id into g_pid from
clubbing.member g1 where
g1.member_id=NEW.member_id;
if(g_pid=NEW.member_id) THEN
RAISE NOTICE 'violating condition:id already exists';
else
RAISE NOTICE 'no duplication';
RETURN NEW;
end if;
END
$BODY$

INSERT into clubbing.member values
(1,'kaak',111111111,'kama@gmail.com',1,'11-01-1111')



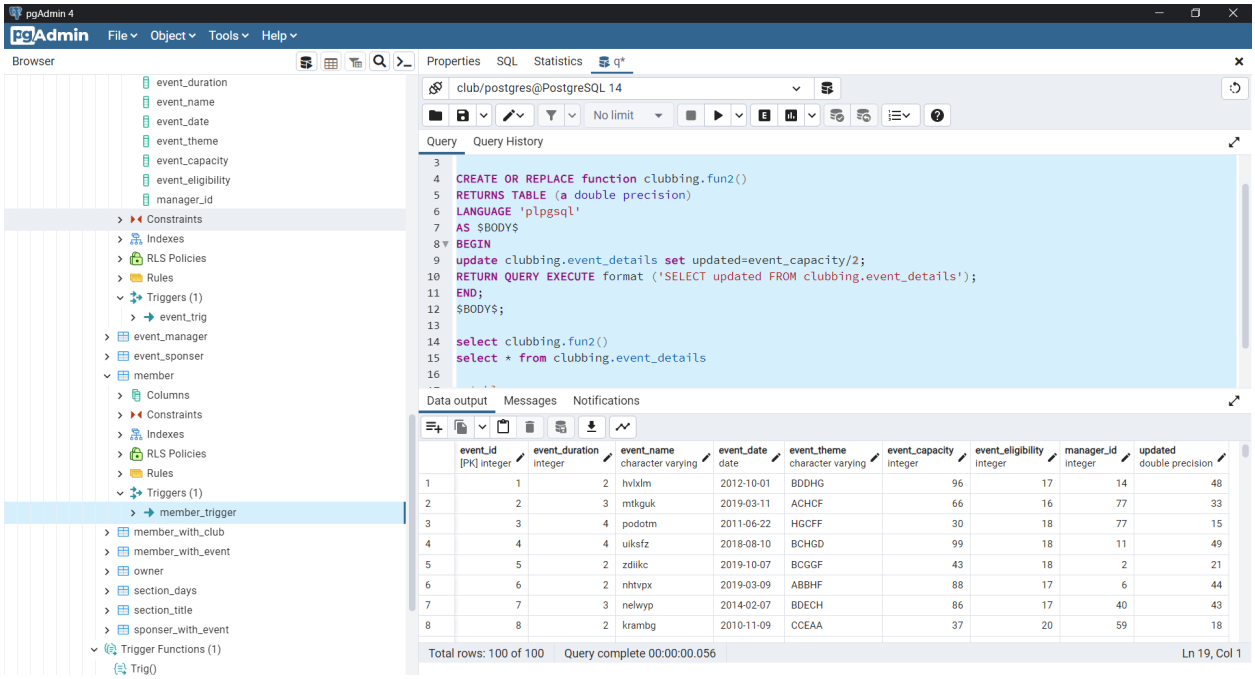
9. Create a function which will add a new column in event_details table which will have an updated event_capacity(event_capacity half)

Updated capacity:
ALTER table clubbing.event_details
add updated double precision

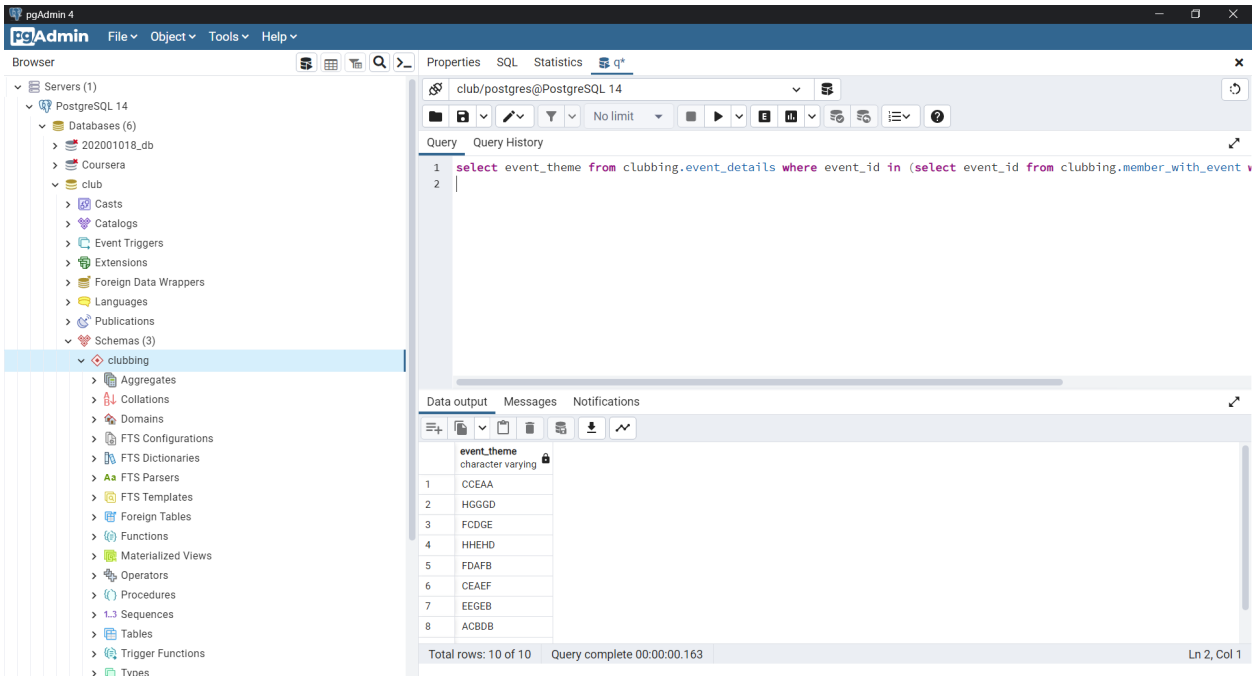
CREATE OR REPLACE function clubbing.fun2()
RETURNS TABLE (a double precision)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
update clubbing.event_details set updated=event_capacity/2;
RETURN QUERY EXECUTE format ('SELECT updated FROM
clubbing.event_details');
END;
$BODY$;

select clubbing.fun2()
select * from clubbing.event_details

--tables
select * from clubbing.event_details



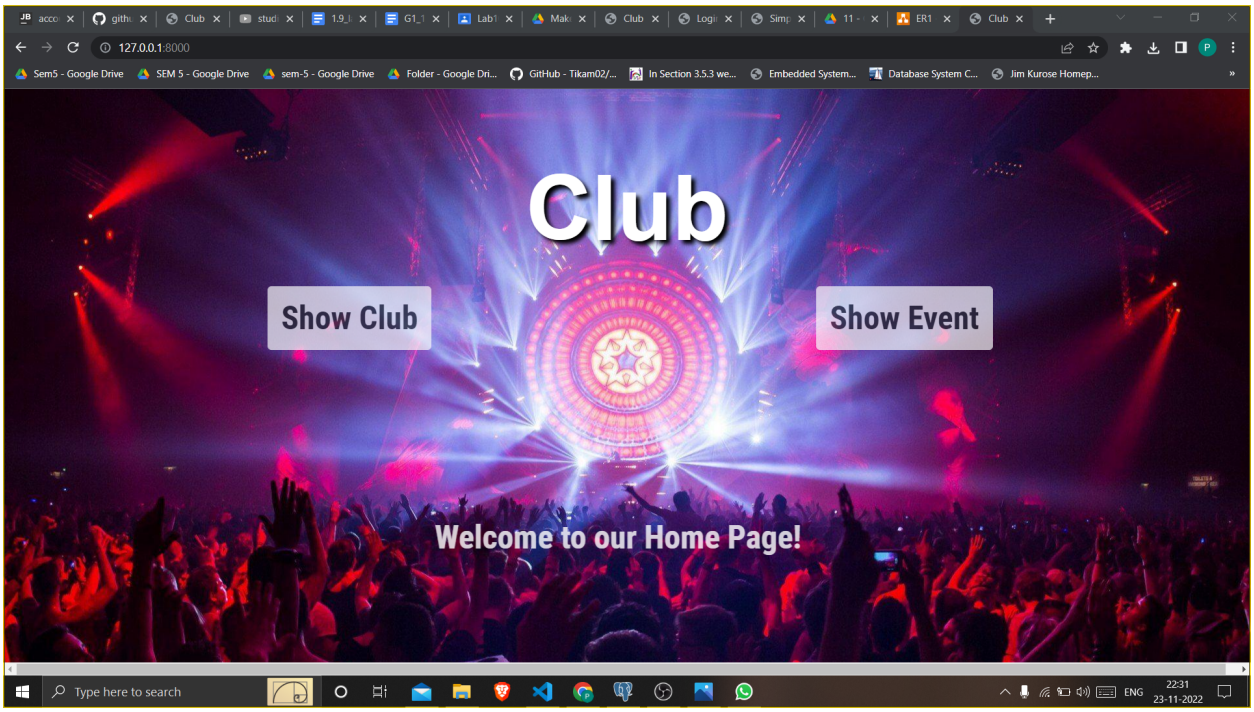
10. Find theme that attended by first 10 member
    select event_theme from clubbing.event_details where event_id in (select event_id from clubbing.member_with_event where member_id<10 and member_id>0)



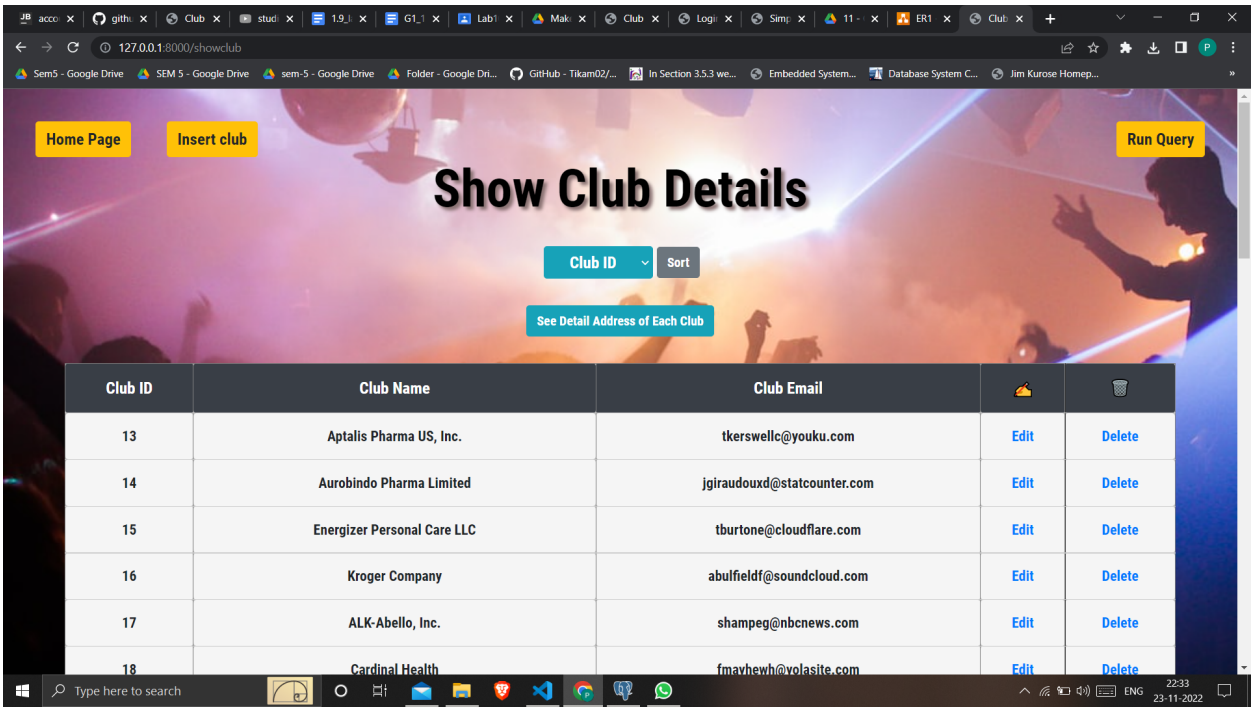
```
UPDATE clubbing.club
SET club_email= '1@columbia.edu'
WHERE club_id=1;

delete from clubbing.club_contact where club_id=1

select * from clubbing.club order by club_id
select * from clubbing.club_contact  order by club_id
```

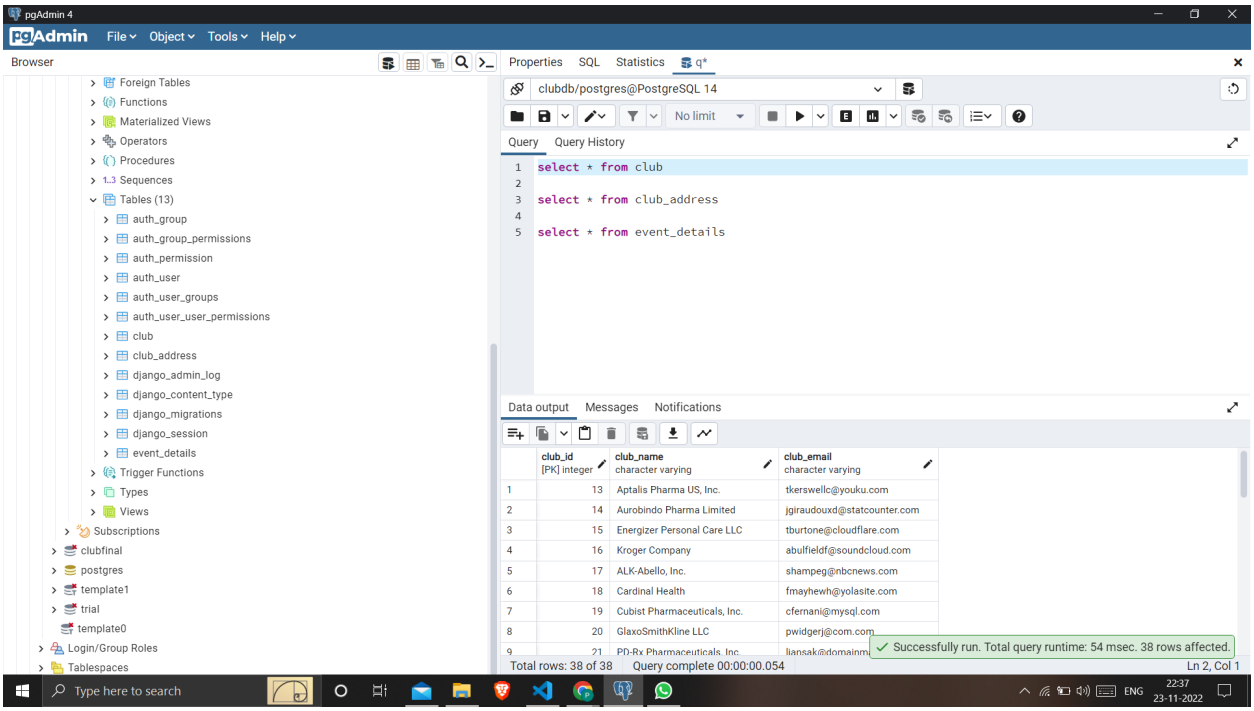
Final Product

1. Main Page

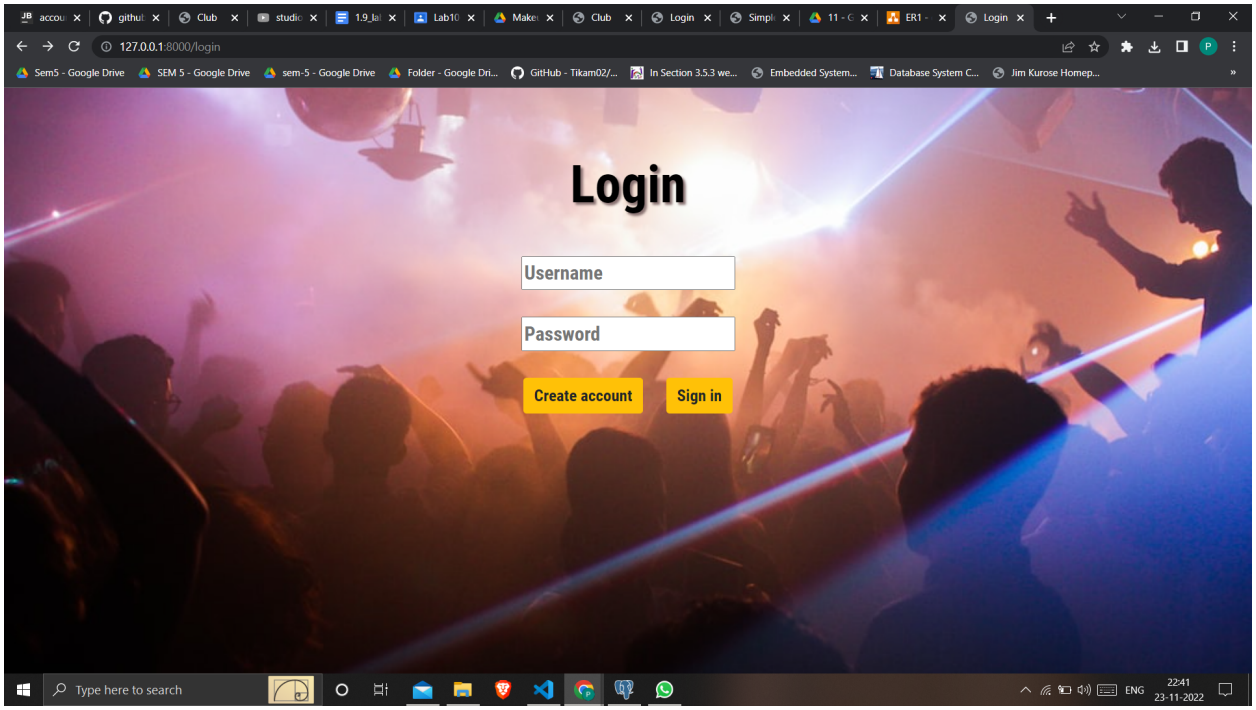


2. Club Details

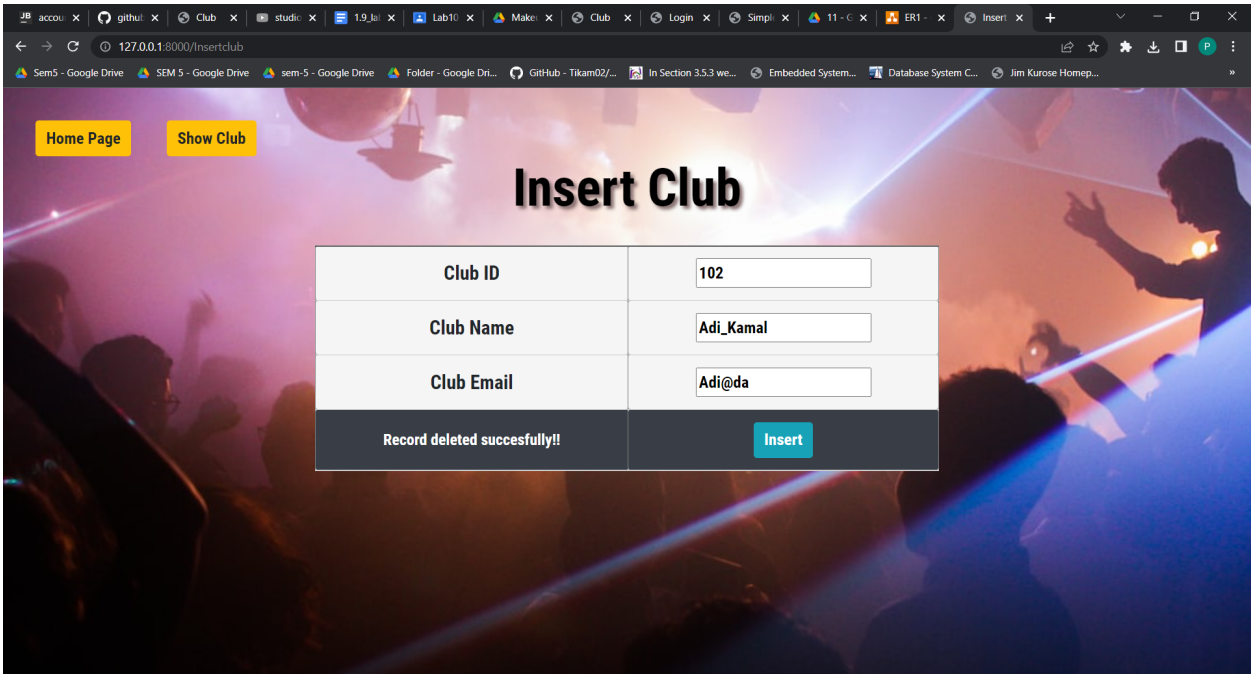


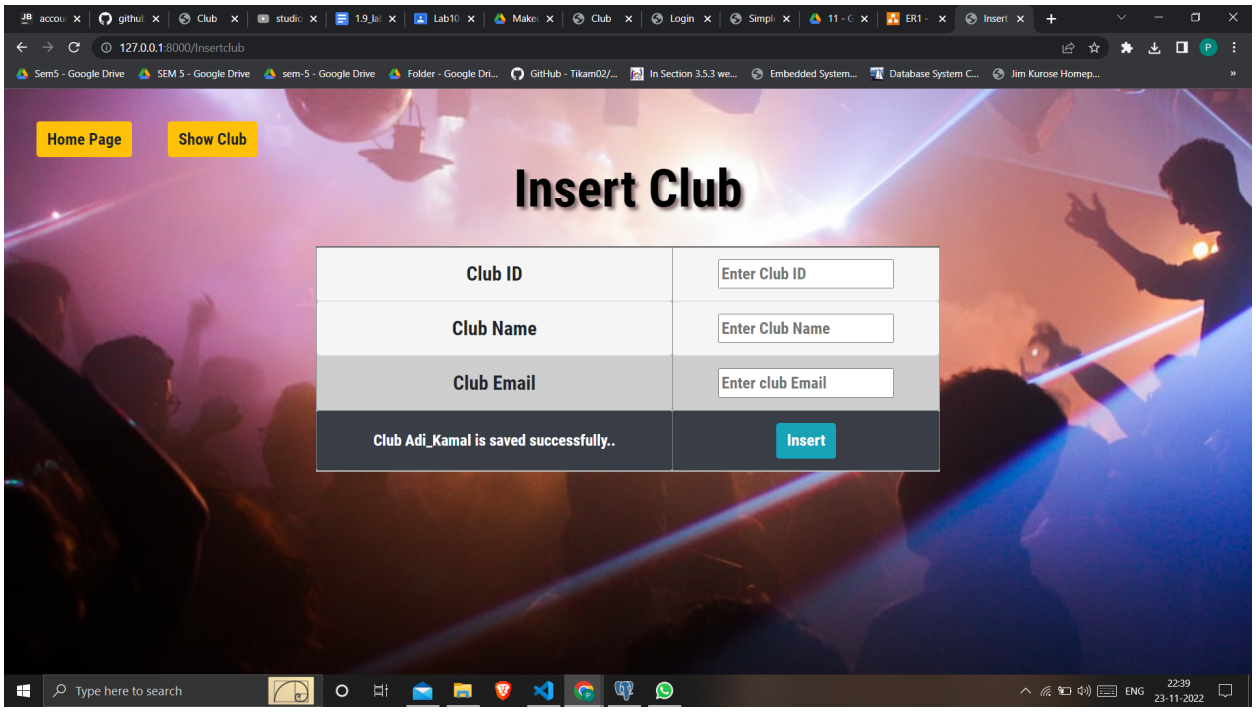


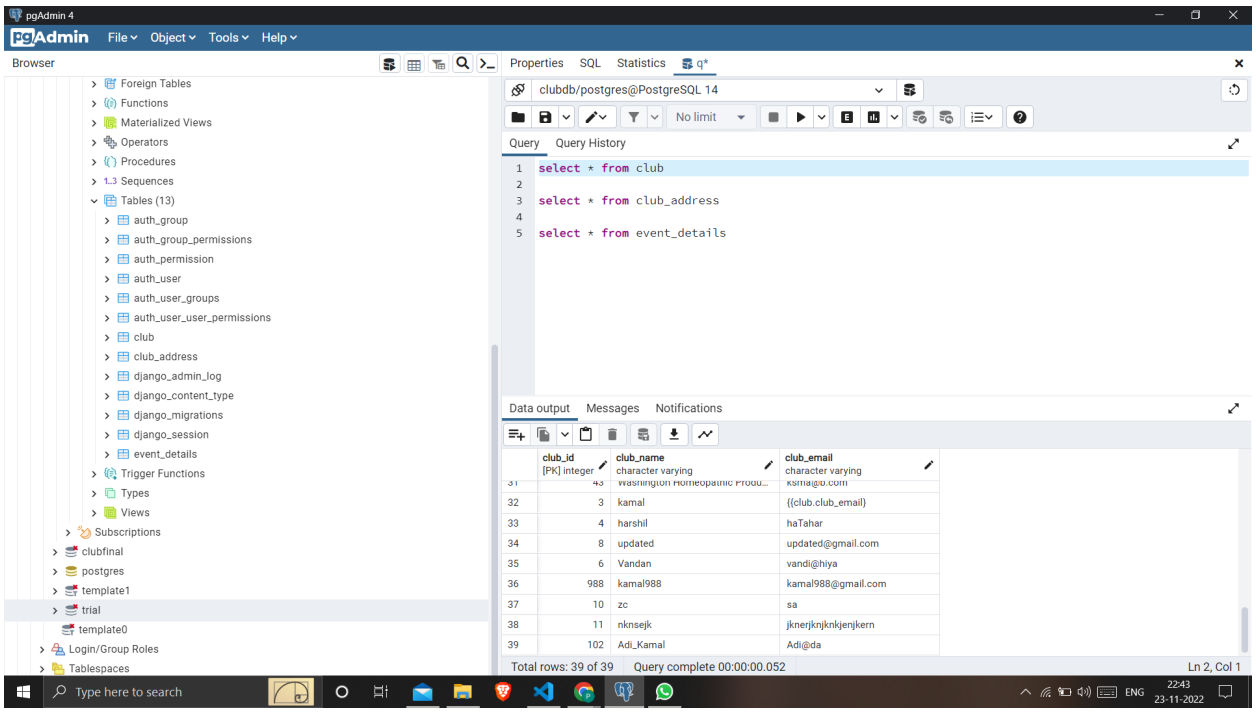
3. Login Page in Order to Insert New Club



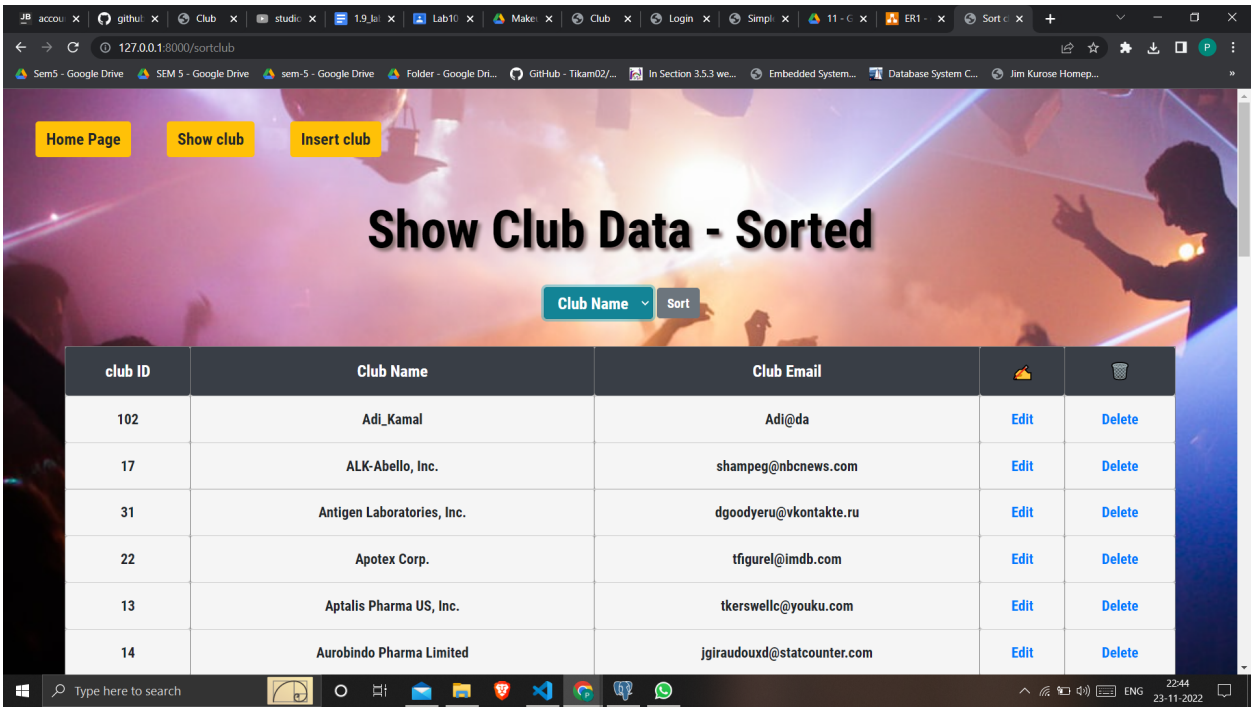
4. Insert Page



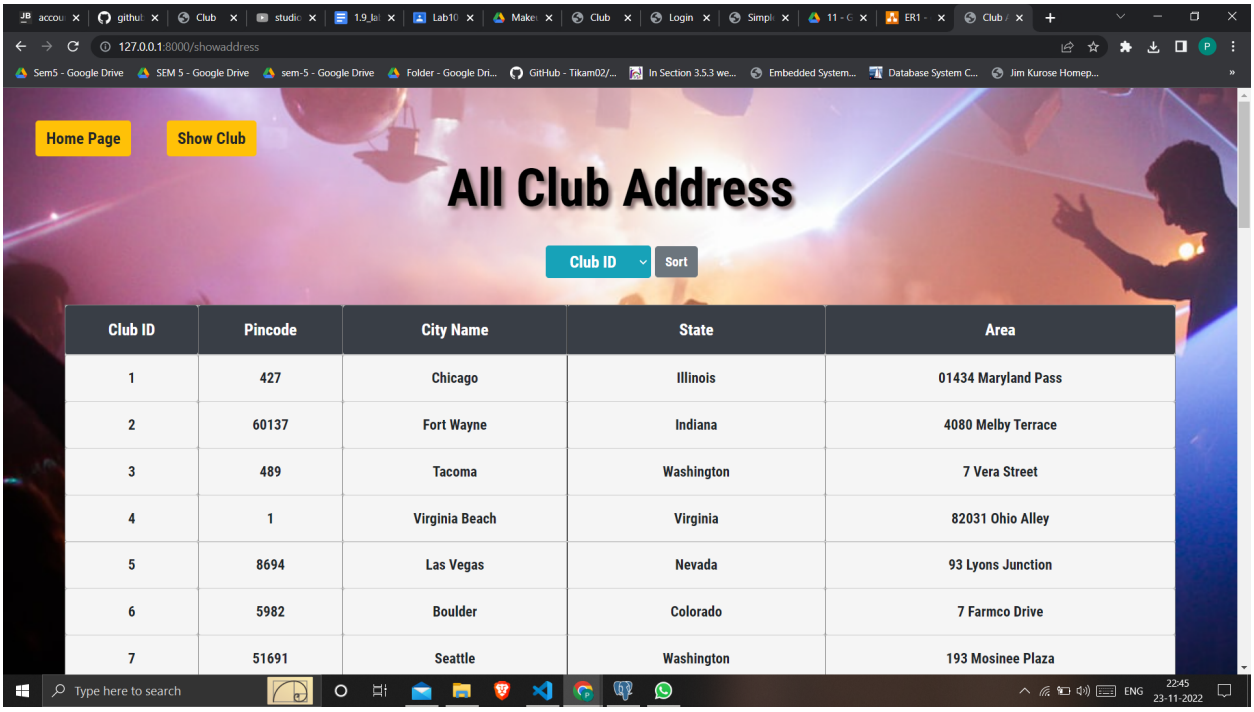

127.0.0.1:8000/Insertclub
Home Page
Show Club
Insert Club
Club ID
Enter Club ID
Club Name
Enter Club Name
Club Email
Enter club Email
Club Adi_Kamal is saved successfully..
Insert
Type here to search
ENG
22:39
23-11-2022


pgAdmin 4
File
Object
Tools
Help
Browser
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (13)
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
club
club_address
django_admin_log
django_content_type
django_migrations
django_session
event_details
Trigger Functions
Types
Views
Subscriptions
clubfinal
postgres
template1
trial
template0
Login/Group Roles
Tablespaces
Properties
SQL
Statistics
clubdb/postgres@PostgreSQL 14
No limit
Query
Query History
select * from club
select * from club_address
select * from event_details
Data output
Messages
Notifications
club_id [PK] integer
club_name character varying
club_email character varying
32 3 kamal {{club.club_email}}
33 4 harshil haTahar
34 8 updated updated@gmail.com
35 6 Vandan vandi@hiya
36 988 kamal988 kamal988@gmail.com
37 10 zc sa
38 11 nknsejk jknerjknjknkjenjkern
39 102 Adi_Kamal Adi@da
Total rows: 39 of 39
Query complete 00:00:00.052
Ln 2, Col 1
Type here to search
ENG
22:43
23-11-2022

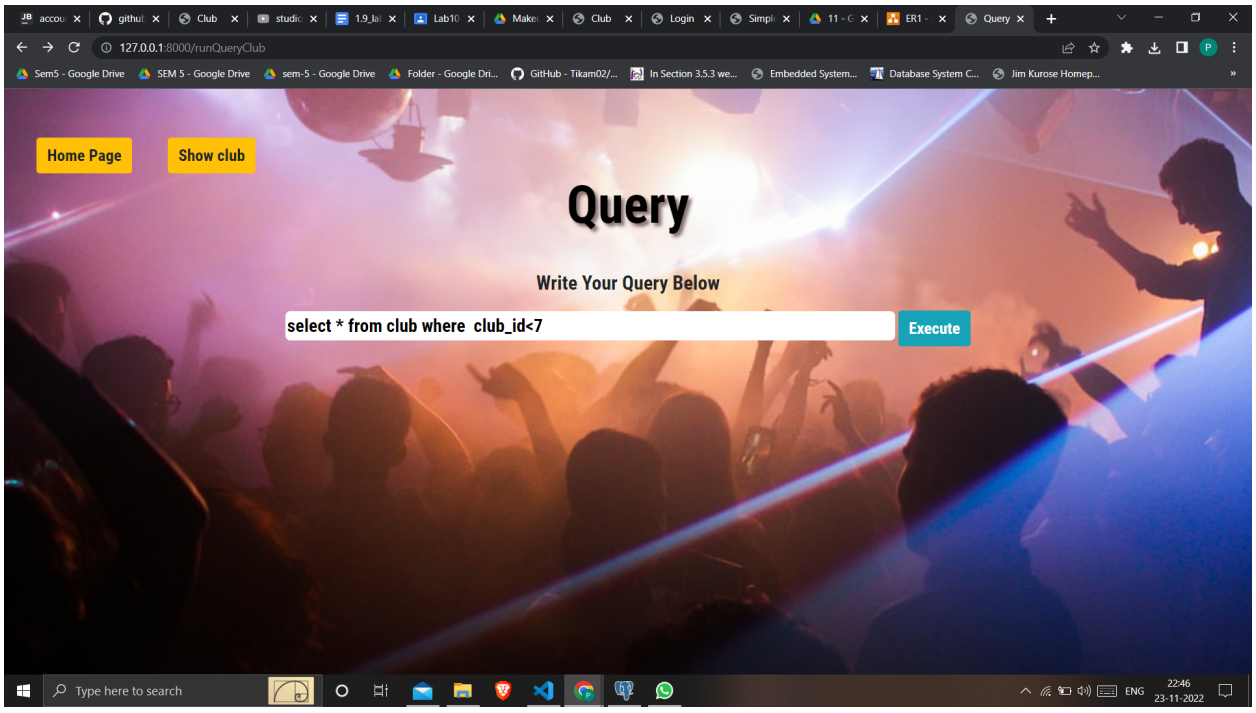
5. Club Details in sorted Order according to Name

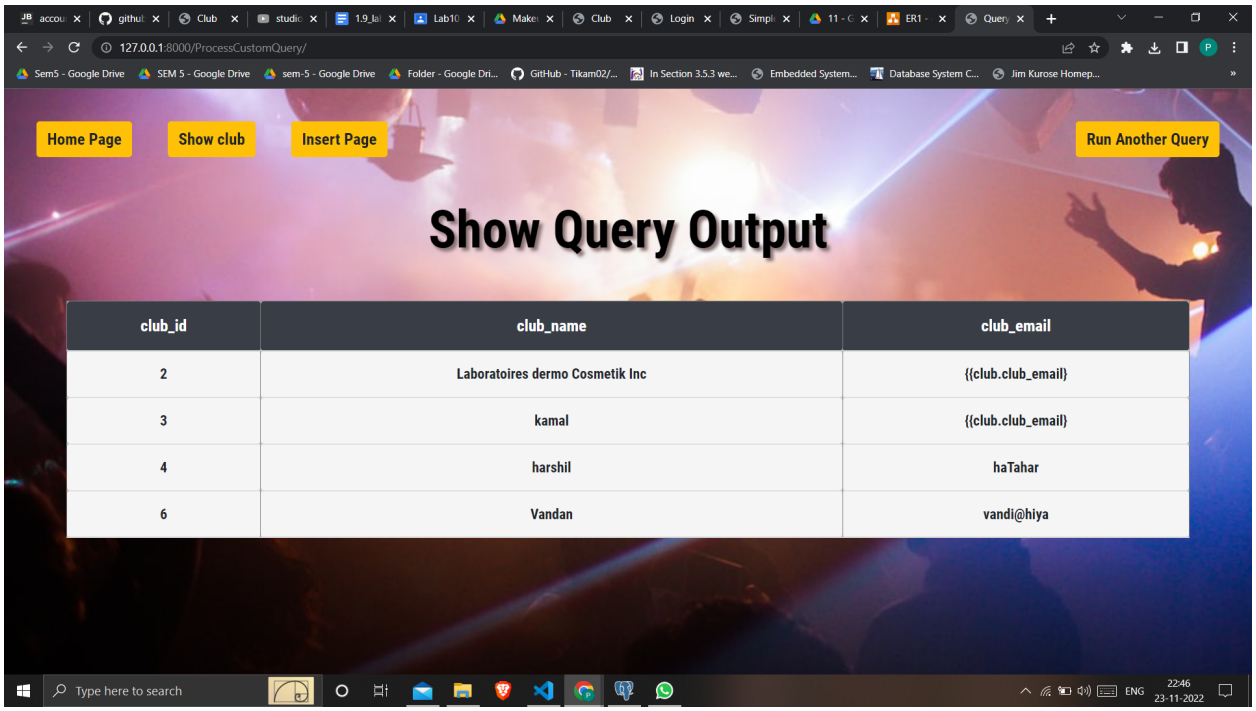


6. Club Address containing foreign key as Club ID from Club Details

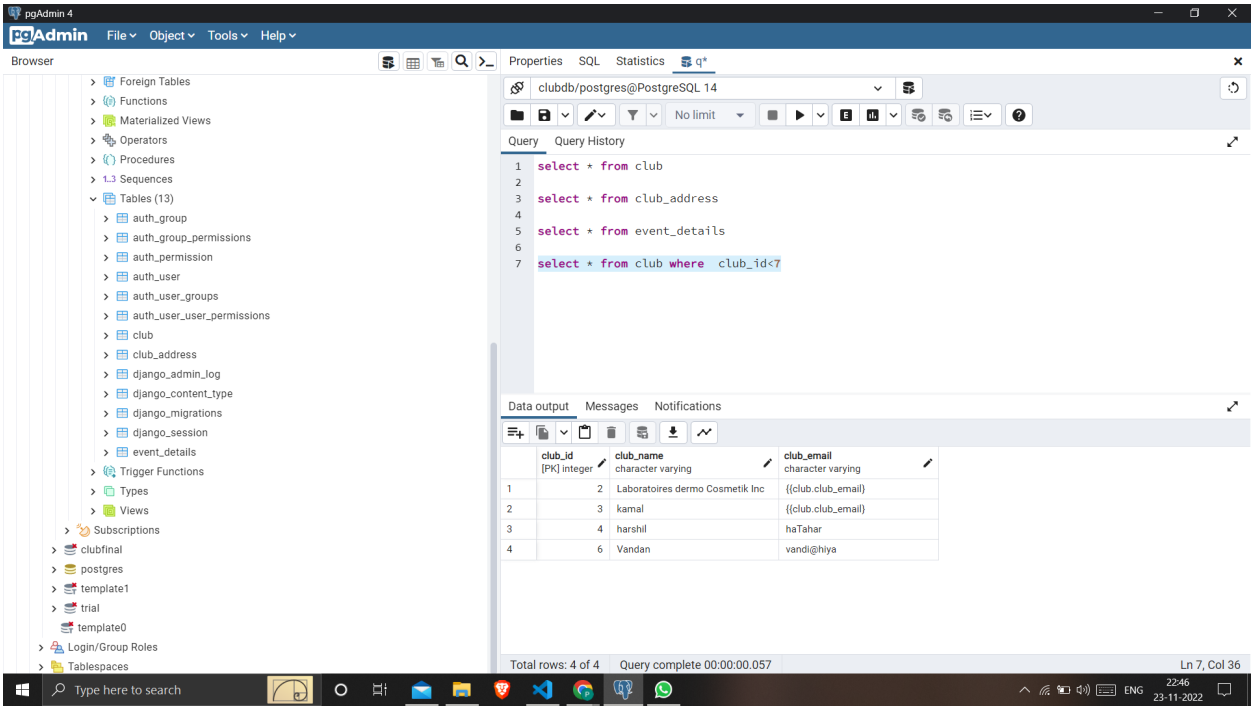


7. Custom Query

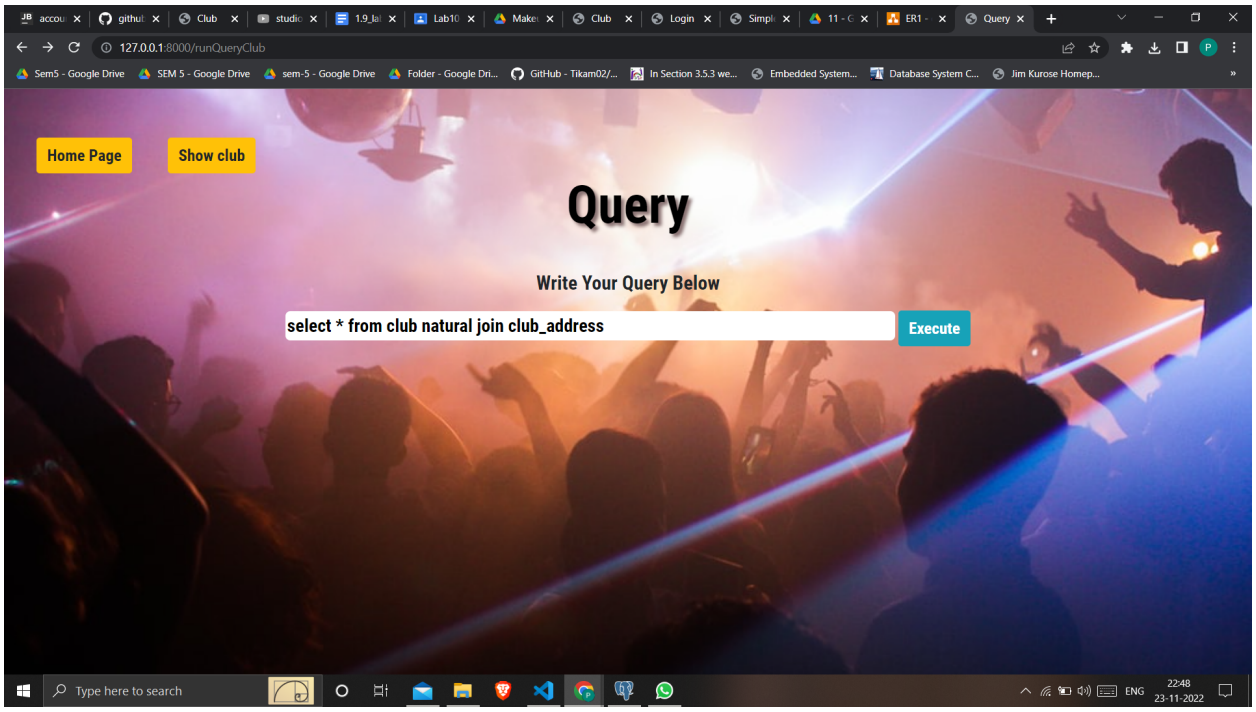


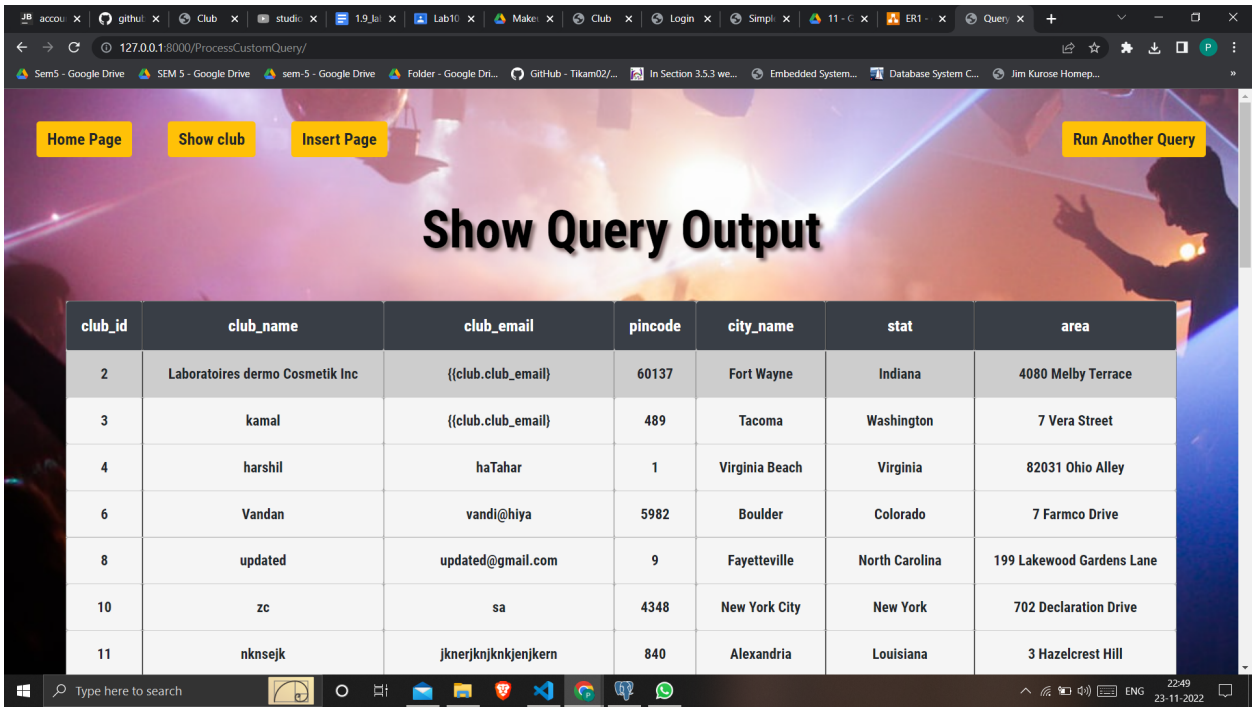


club_id	club_name	club_email
2	Laboratoires dermo Cosmetik Inc	{{club.club_email}
3	kamal	{{club.club_email}
4	harshil	haTahar
6	Vandan	vandi@hiya

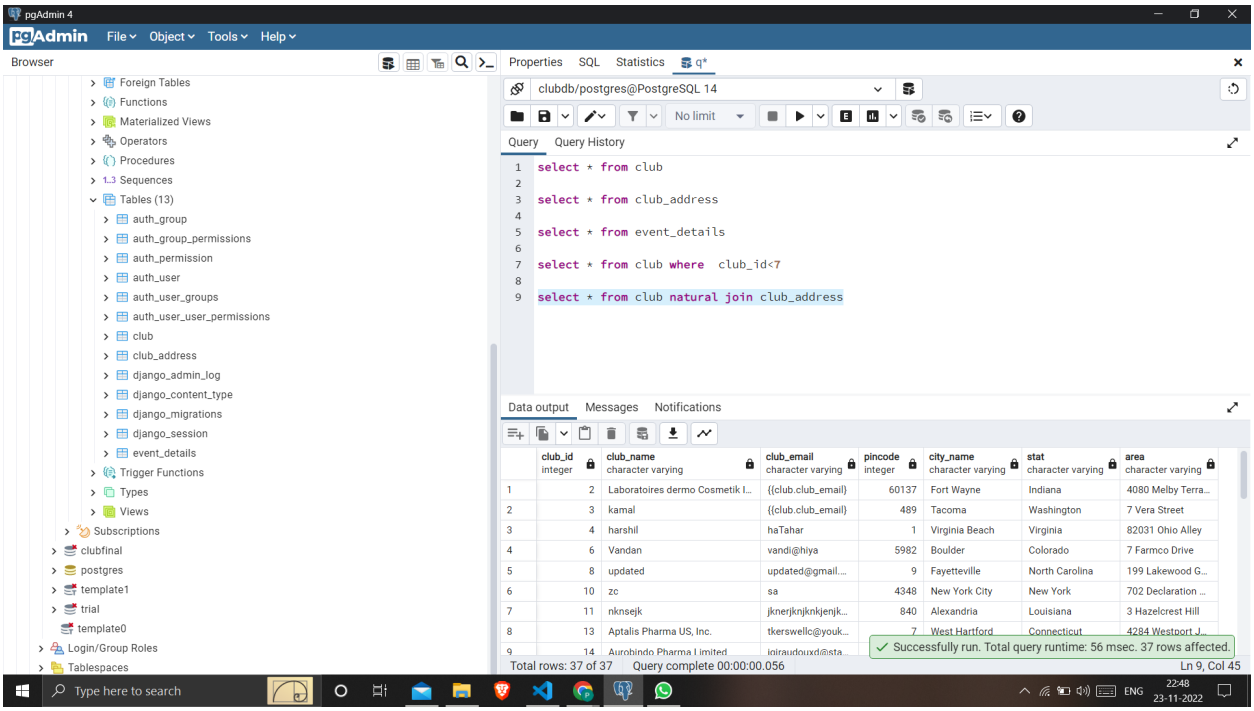

pgAdmin 4
File
Object
Tools
Help
Browser
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (13)
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
club
club_address
django_admin_log
django_content_type
django_migrations
django_session
event_details
Trigger Functions
Types
Views
Subscriptions
clubfinal
postgres
template1
trial
template0
Login/Group Roles
Tablespaces
Properties
SQL
Statistics
clubdb/postgres@PostgreSQL 14
No limit
Query
Query History
select * from club
select * from club_address
select * from event_details
select * from club where club_id<7
Data output
Messages
Notifications
club_id [PK] integer
club_name character varying
club_email character varying
1 2 Laboratoires dermo Cosmetik Inc {{club.club_email}}
2 3 kamal {{club.club_email}}
3 4 harshil haTahar
4 6 Vandan vandi@hiya
Total rows: 4 of 4
Query complete 00:00:00.057
Ln 7, Col 36
Type here to search
ENG
22:46
23-11-2022

8. Custom Query Containing natural join of 2 relation

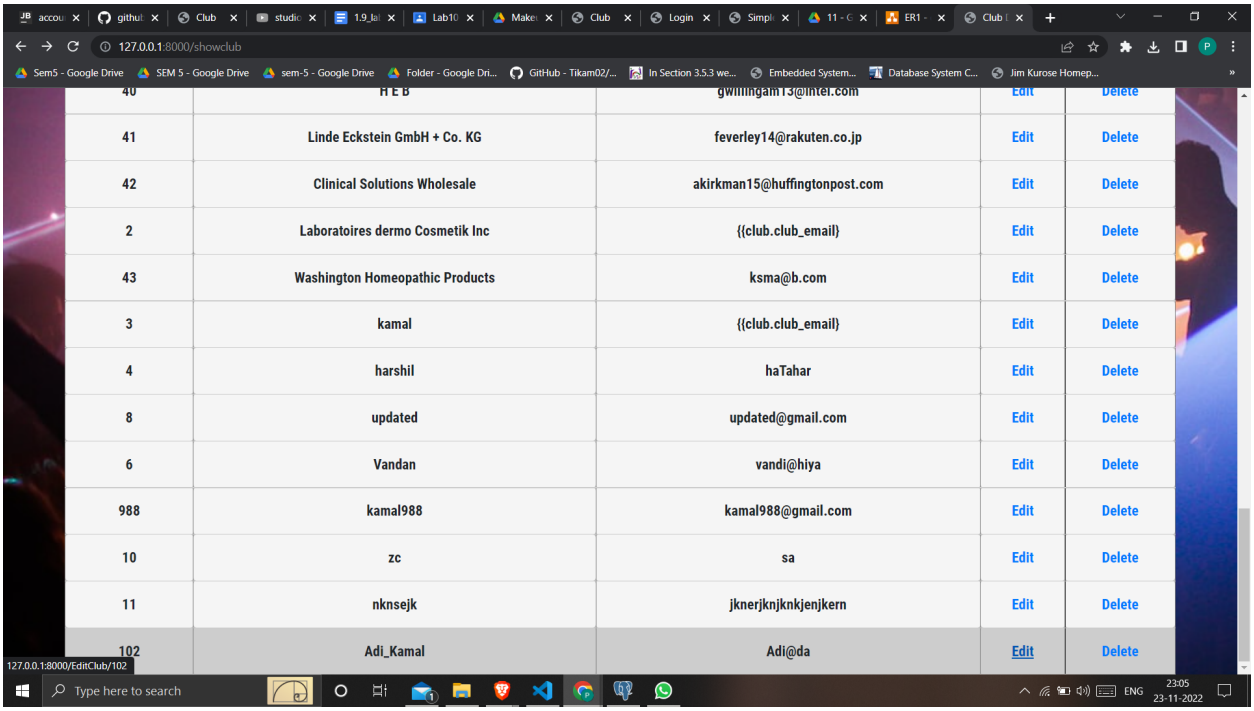


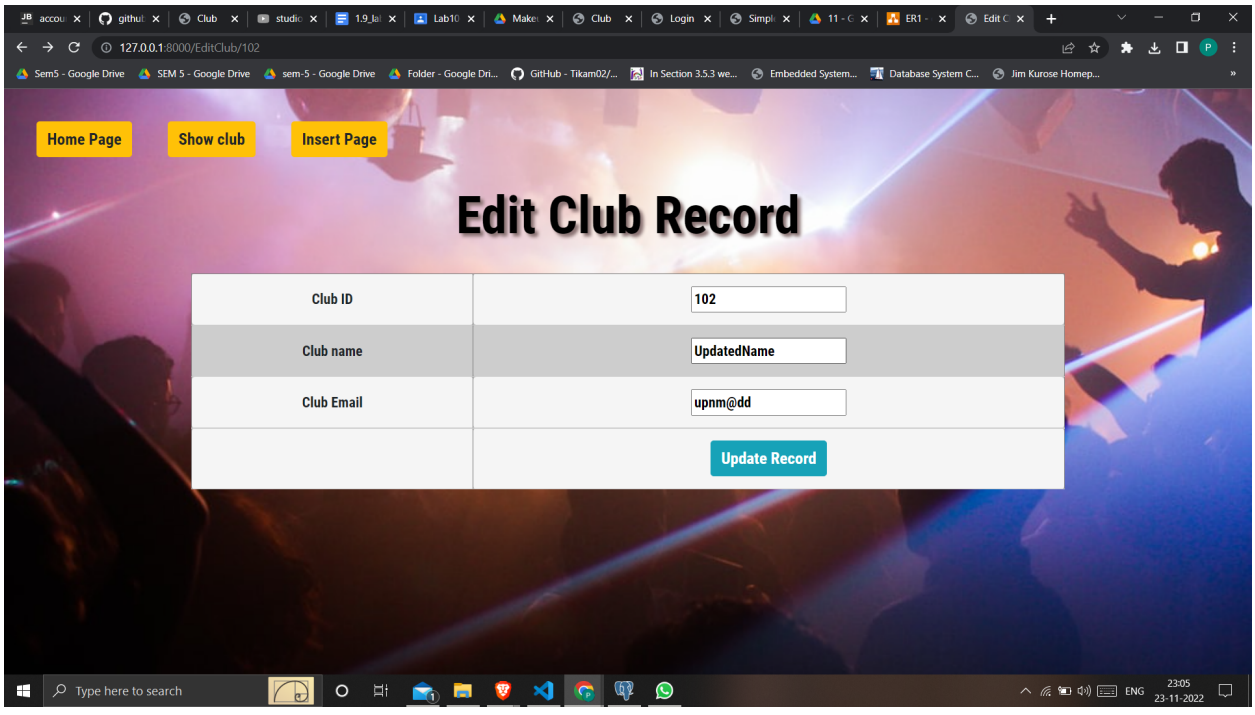


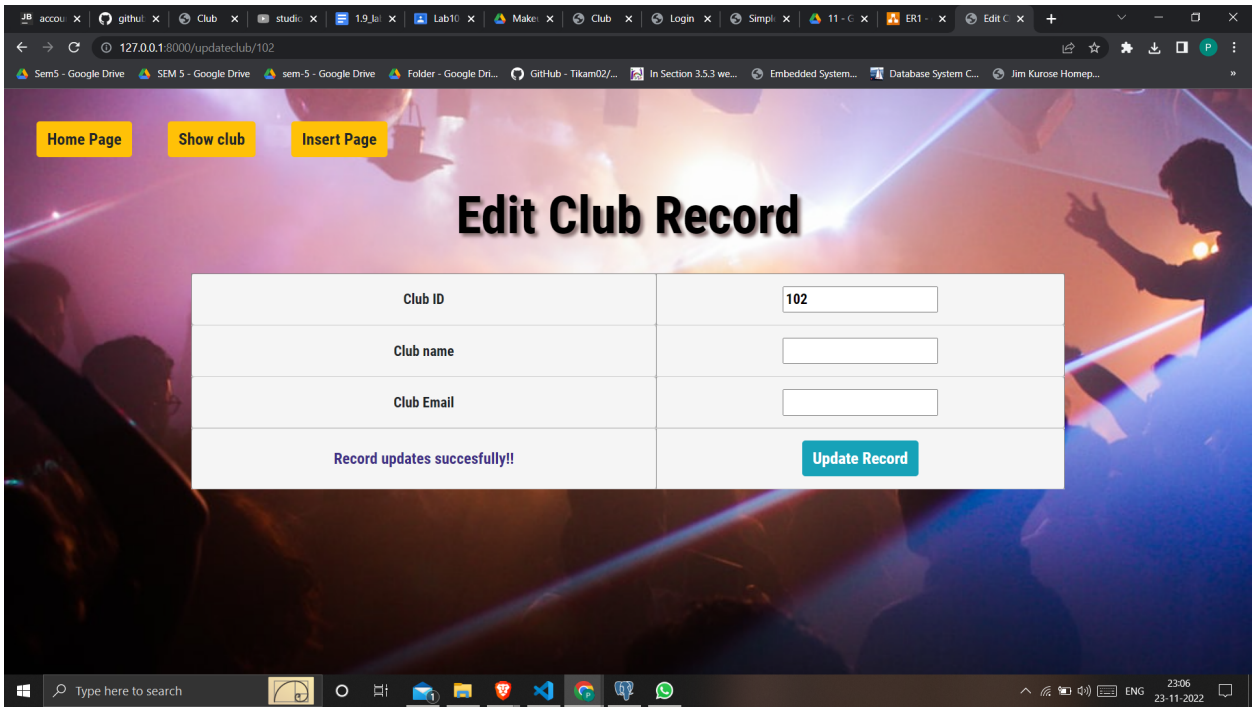
club_id	club_name	club_email	pincode	city_name	stat	area
2	Laboratoires dermo Cosmetik Inc	{{club.club_email}}	60137	Fort Wayne	Indiana	4080 Melby Terrace
3	kamal	{{club.club_email}}	489	Tacoma	Washington	7 Vera Street
4	harshil	haTahar	1	Virginia Beach	Virginia	82031 Ohio Alley
6	Vandan	vandi@hiya	5982	Boulder	Colorado	7 Farmco Drive
8	updated	updated@gmail.com	9	Fayetteville	North Carolina	199 Lakewood Gardens Lane
10	zc	sa	4348	New York City	New York	702 Declaration Drive
11	nknsejk	jknerjknjknkjenjkern	840	Alexandria	Louisiana	3 Hazelcrest Hill

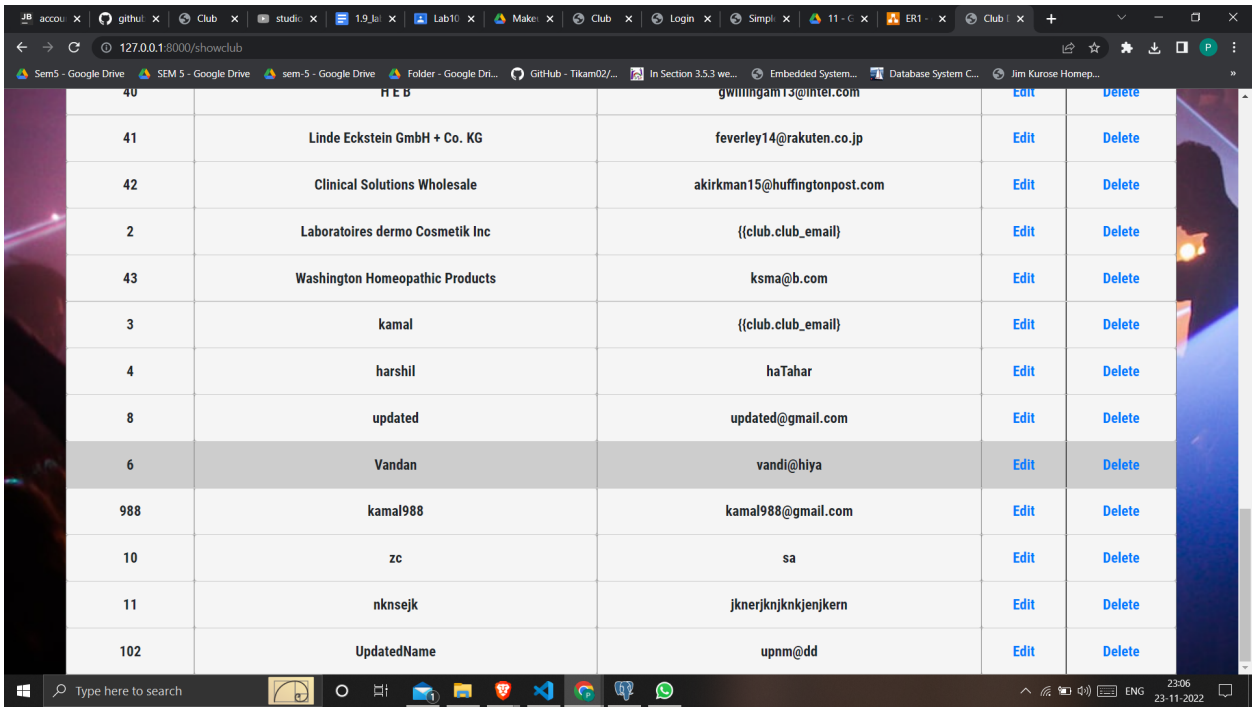

pgAdmin 4
File
Object
Tools
Help
Browser
Properties
SQL
Statistics
clubdb/postgres@PostgreSQL 14
No limit
Query
Query History
select * from club
select * from club_address
select * from event_details
select * from club where club_id<7
select * from club natural join club_address
Data output
Messages
Notifications
Successfully run. Total query runtime: 56 msec. 37 rows affected.
Total rows: 37 of 37
Query complete 00:00:00.056
Ln 9, Col 45
Type here to search
ENG
22:48
23-11-2022

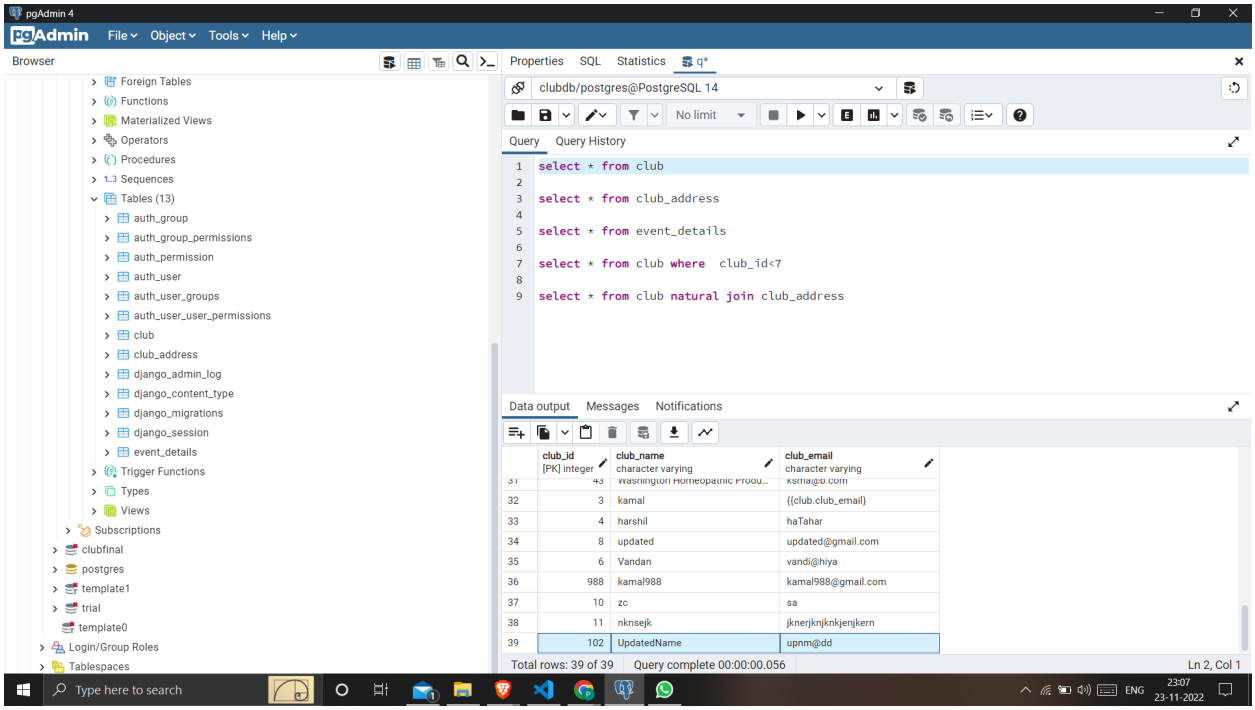
9. Updation of Club details



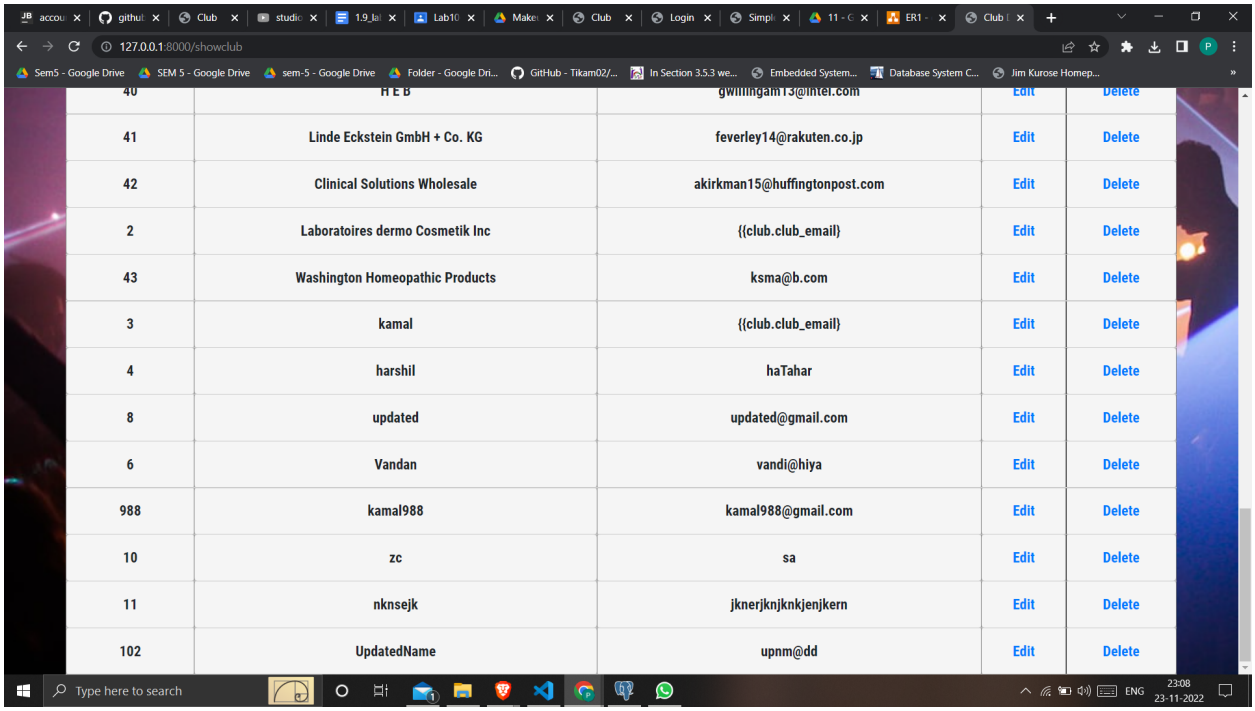



127.0.0.1:8000/updateclub/102
Home Page
Show club
Insert Page
Edit Club Record
Club ID
102
Club name
Club Email
Record updates succesfully!!
Update Record
Type here to search
ENG
23:06
23-11-2022

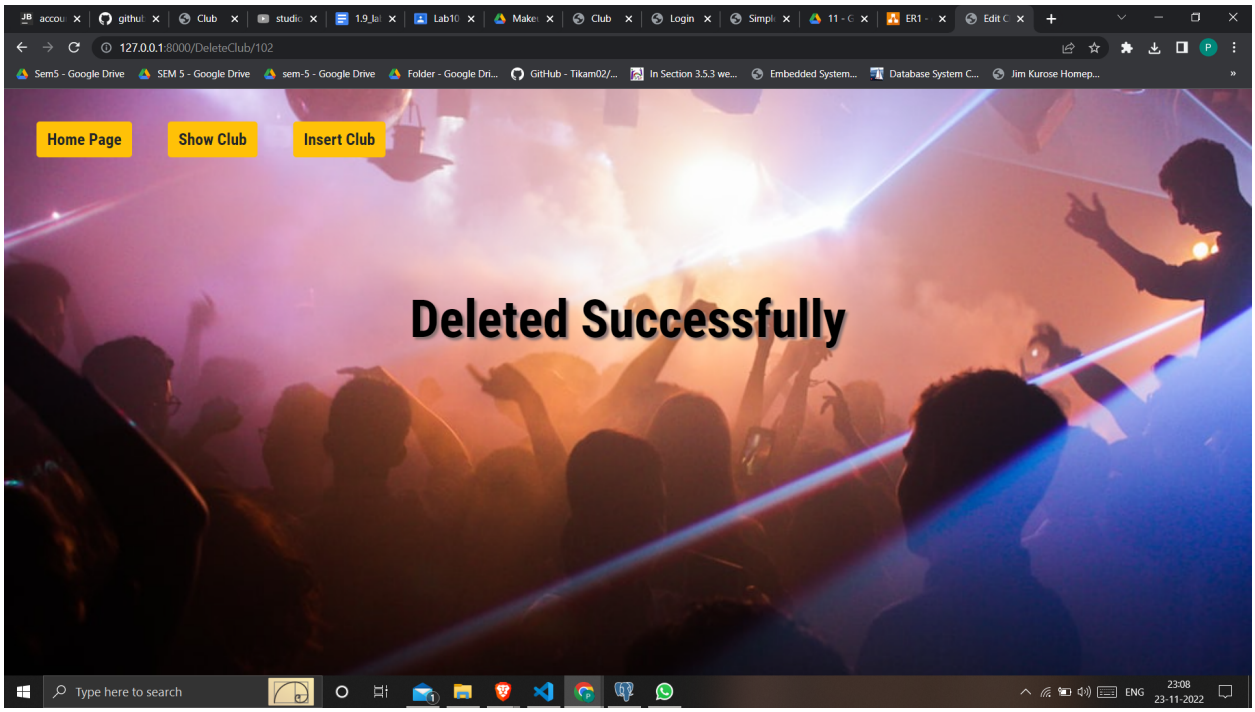

127.0.0.1:8000/showclub
41 Linde Eckstein GmbH + Co. KG feverley14@rakuten.co.jp Edit Delete
42 Clinical Solutions Wholesale akirkman15@huffingtonpost.com Edit Delete
2 Laboratoires dermo Cosmetik Inc {{club.club_email}} Edit Delete
43 Washington Homeopathic Products ksma@b.com Edit Delete
3 kamal {{club.club_email}} Edit Delete
4 harshil haTahar Edit Delete
8 updated updated@gmail.com Edit Delete
6 Vandan vandi@hiya Edit Delete
988 kamal988 kamal988@gmail.com Edit Delete
10 zc sa Edit Delete
11 nknsejk jknerjknjknkjenjkern Edit Delete
102 UpdatedName upnm@dd Edit Delete
Type here to search
ENG
23:06
23-11-2022

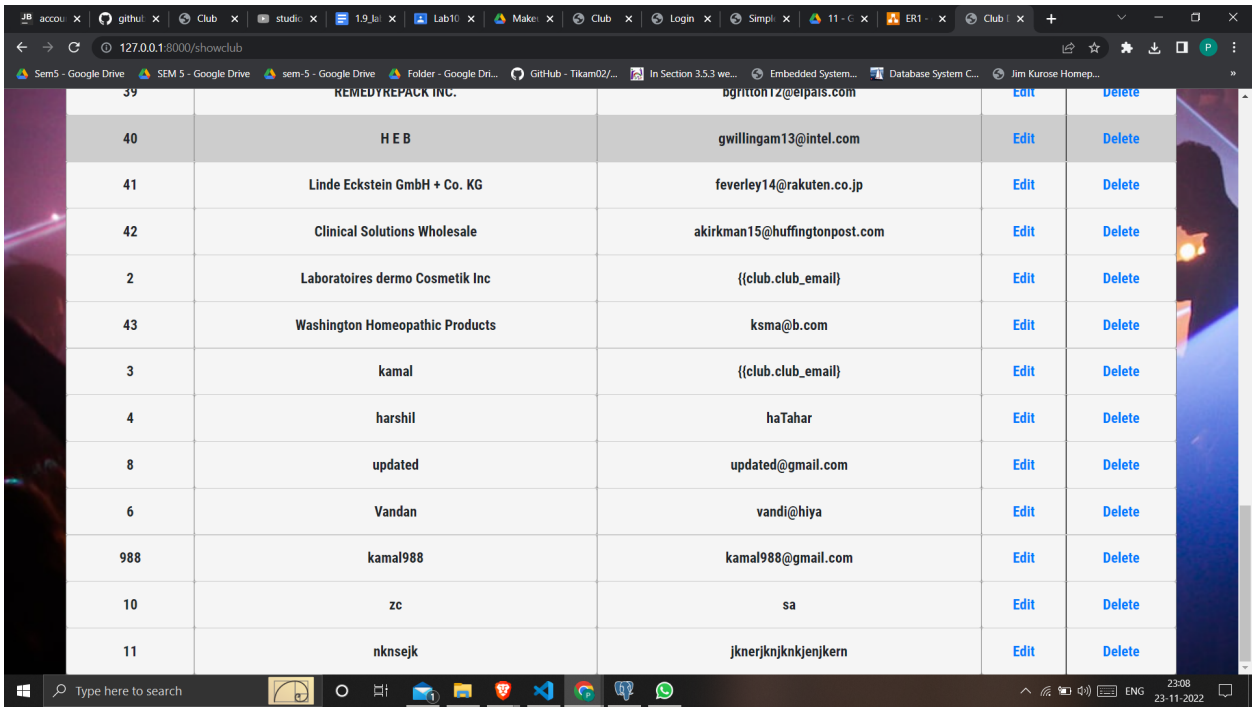

pgAdmin 4
File
Object
Tools
Help
Browser
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (13)
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
club
club_address
django_admin_log
django_content_type
django_migrations
django_session
event_details
Trigger Functions
Types
Views
Subscriptions
clubfinal
postgres
template1
trial
template0
Login/Group Roles
Tablespaces
Properties
SQL
Statistics
q*
clubdb/postgres@PostgreSQL 14
No limit
Query
Query History
1 select * from club
2
3 select * from club_address
4
5 select * from event_details
6
7 select * from club where club_id<7
8
9 select * from club natural join club_address
Data output
Messages
Notifications
club_id [PK] integer
club_name character varying
club_email character varying
32 3 kamal {{club.club_email}}
33 4 harshil haTahar
34 8 updated updated@gmail.com
35 6 Vandan vandi@hiya
36 988 kamal988 kamal988@gmail.com
37 10 zc sa
38 11 nknsejk jknerjknjknkjenjkern
39 102 UpdatedName upnm@dd
Total rows: 39 of 39
Query complete 00:00:00.056
Ln 2, Col 1
Type here to search
ENG
23:07
23-11-2022

10. Deletion of Data in Club details

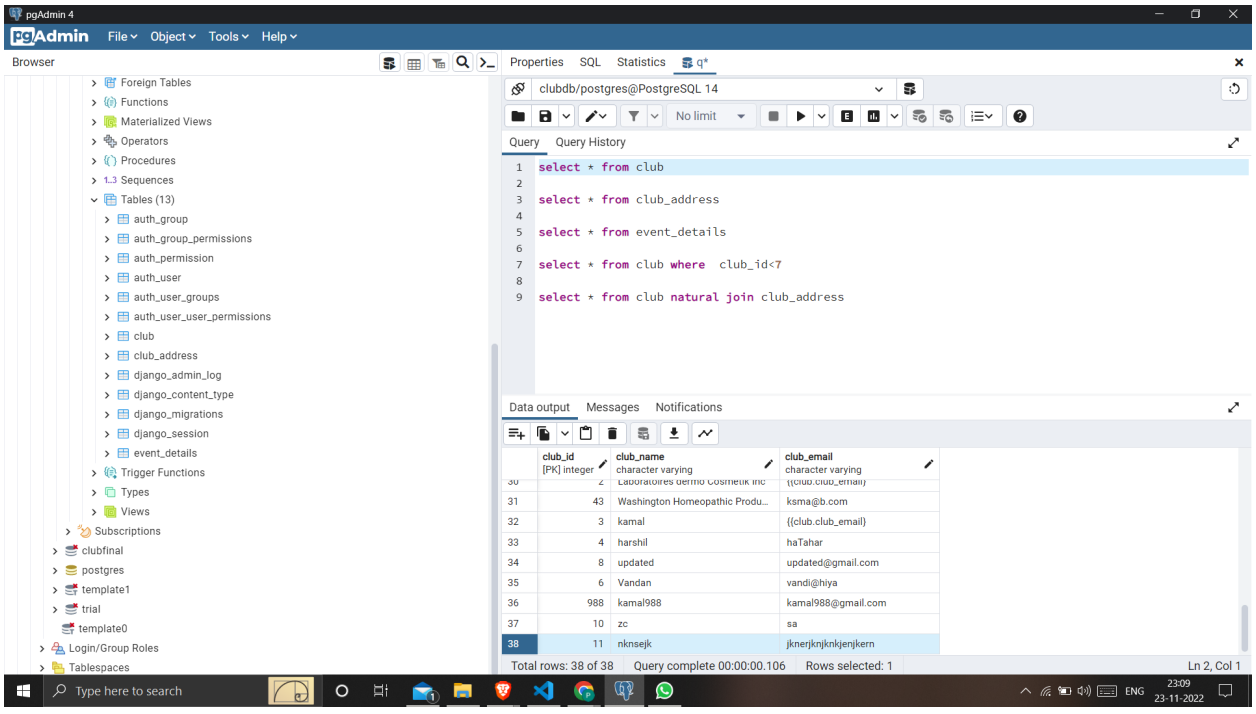


40	H E B	gwillingam13@intel.com	Edit	Delete
41	Linde Eckstein GmbH + Co. KG	feverley14@rakuten.co.jp	Edit	Delete
42	Clinical Solutions Wholesale	akirkman15@huffingtonpost.com	Edit	Delete
2	Laboratoires dermo Cosmetik Inc	{{club.club_email}	Edit	Delete
43	Washington Homeopathic Products	ksma@b.com	Edit	Delete
3	kamal	{{club.club_email}	Edit	Delete
4	harshil	haTahar	Edit	Delete
8	updated	updated@gmail.com	Edit	Delete
6	Vandan	vandi@hiya	Edit	Delete
988	kamal988	kamal988@gmail.com	Edit	Delete
10	zc	sa	Edit	Delete
11	nknsejk	jknerjknjknkjenjkern	Edit	Delete
102	UpdatedName	upnm@dd	Edit	Delete





39	REMEDYREPACK INC.	bgrittonl2@elpais.com	Edit	Delete
40	H E B	gwillingam13@intel.com	Edit	Delete
41	Linde Eckstein GmbH + Co. KG	feverley14@rakuten.co.jp	Edit	Delete
42	Clinical Solutions Wholesale	akirkman15@huffingtonpost.com	Edit	Delete
2	Laboratoires dermo Cosmetik Inc	{{club.club_email}	Edit	Delete
43	Washington Homeopathic Products	ksma@b.com	Edit	Delete
3	kamal	{{club.club_email}	Edit	Delete
4	harshil	haTahar	Edit	Delete
8	updated	updated@gmail.com	Edit	Delete
6	Vandan	vandi@hiya	Edit	Delete
988	kamal988	kamal988@gmail.com	Edit	Delete
10	zc	sa	Edit	Delete
11	nknsejk	jknerjknjknkjenjkern	Edit	Delete



```
select * from club

select * from club_address

select * from event_details

select * from club where  club_id<7

select * from club natural join club_address
```

club_id [PK] integer	club_name character varying	club_email character varying
31	43	Washington Homeopathic Produ...	ksma@b.com
32	3	kamal	{{club.club_email}
33	4	harshil	haTahar
34	8	updated	updated@gmail.com
35	6	Vandan	vandi@hiya
36	988	kamal988	kamal988@gmail.com
37	10	zc	sa
38	11	nknsejk	jknerjknjknkjenjkern

Link :

https://youtu.be/DQ_esLHSTzQ