

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

1. Introduction
 - 1.1. Purpose
 - 1.2. Intended Audience and Reading Suggestions:
 - 1.3. Product Scope
 - 1.4. Description
 - 1.5. Real World Working Flow
 - 1.5.1. The current flow of the system
 - 1.5.2. Difficulties or problems encountered in the current flow
2. Data collection
 - 2.1. Background Readings
 - 2.1.1. The current flow of the system
 - 2.1.1.1. Functionalities clubs provide to their customers
 - 2.2. Interviews
 - 2.3. Questionnaire
 - 2.4. Observations
3. Create Fact-Finding Chart
4. List Requirements
5. User categories and Privileges
6. Assumptions
7. Business Constraints

ER Diagram

8. Problem Description
9. Noun (& Verb) Analysis
 - 9.1. All Extracted Nouns & Verbs from Problem Description
 - 9.2. Accepted Noun & Verbs list
 - 9.3. Rejected Noun & Verbs list
10. Develop the ER Diagram (ERD)

Final Normalization and DDL Scripts SQL Queries

- 11. Mapping E-R Model to Relational Model
 - 11.1. Final relations
- 12. DDL Scripts
- 13. 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_subscribed, 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.

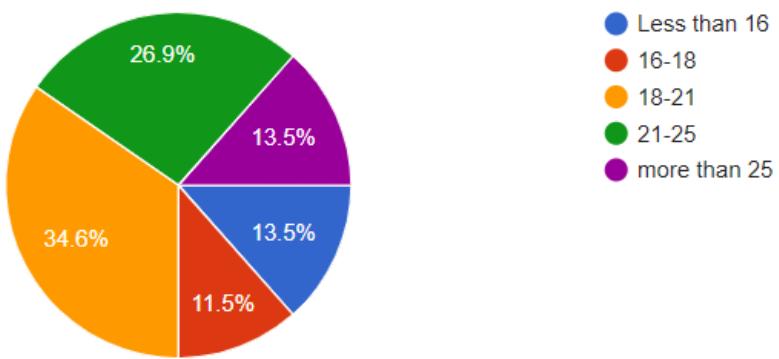
- **Combined Requirements gathered from the Interviews:**

- 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:

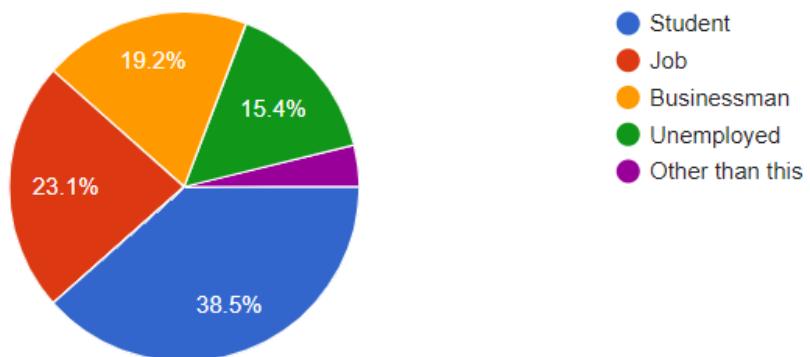
What is your age?

52 responses



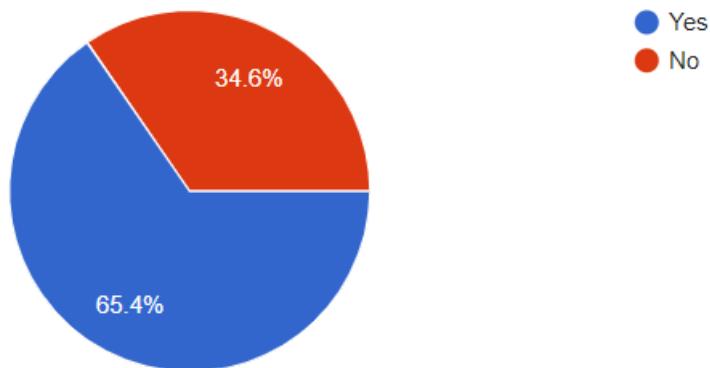
What is your occupation?

52 responses



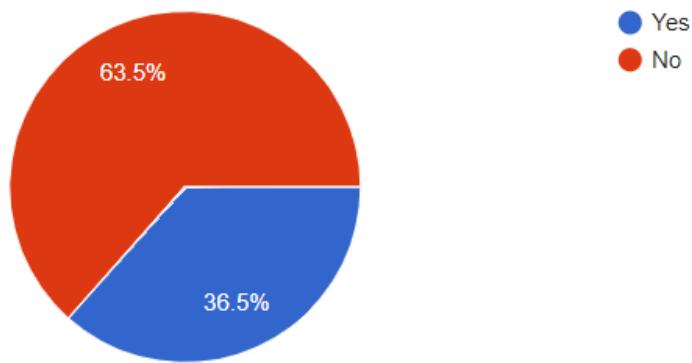
Have you visited any club?

52 responses



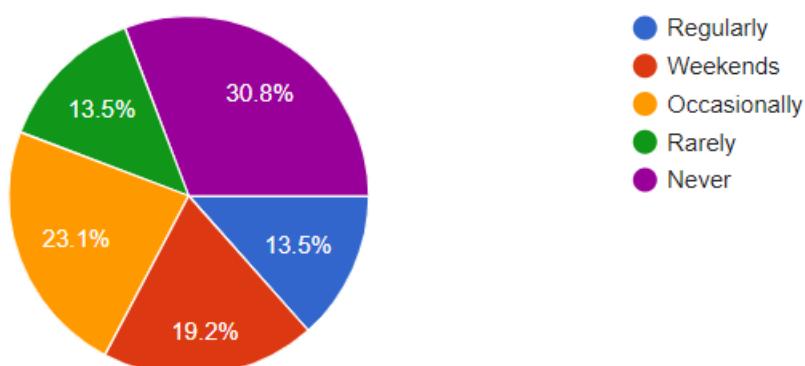
Are you member of any club?

52 responses



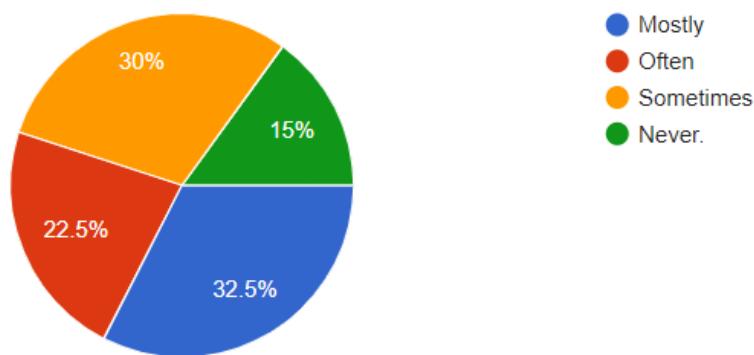
how often did you attended club?

52 responses



You often take drink at club?

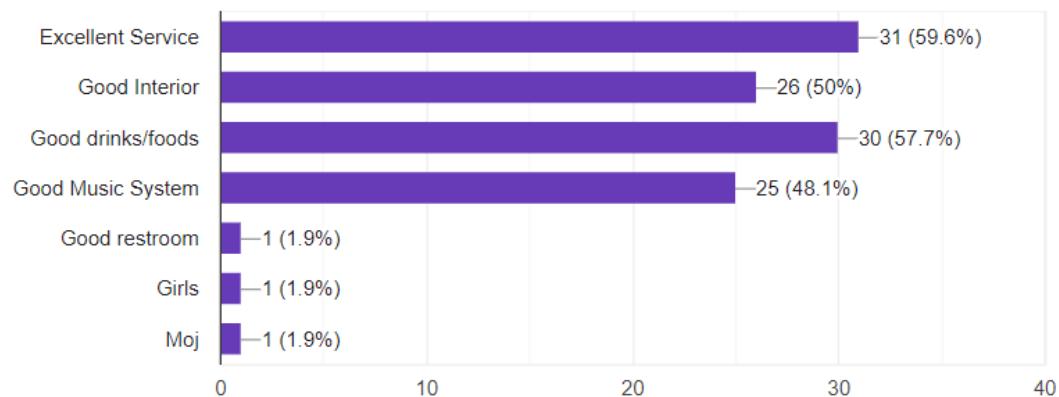
40 responses



What do you expect from a good Club?

[Copy](#)

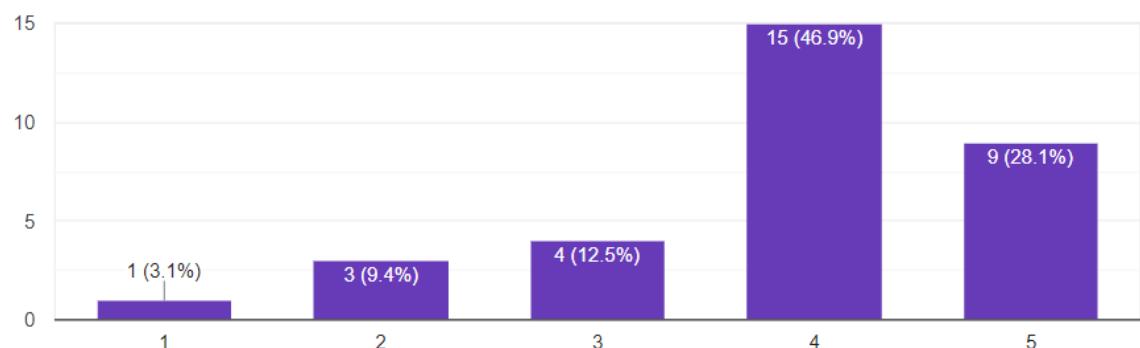
52 responses



What is Your experience of Restroom.

[Copy](#)

32 responses

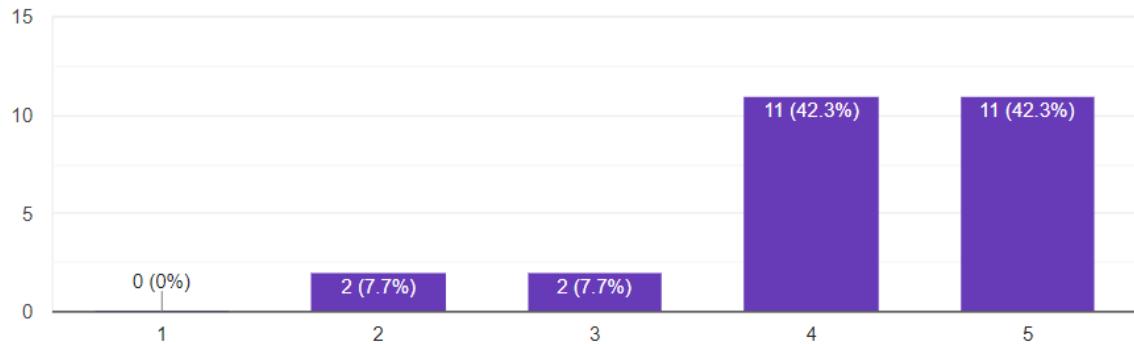


Are you satisfied with premium subscription (Membership)?

 Copy

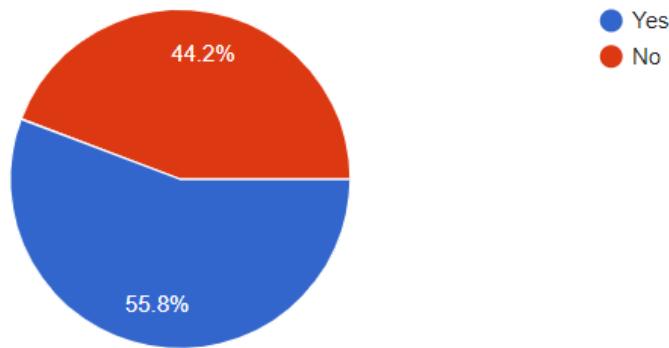
Please Rate your experience.

26 responses



Have you attended any Dj Party at clubs?

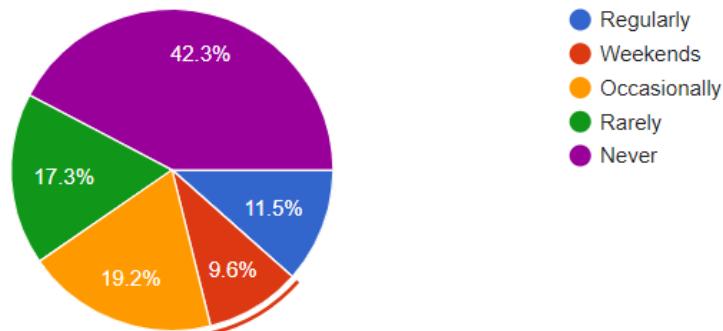
52 responses



how often did you attended DJ Party at clubs?



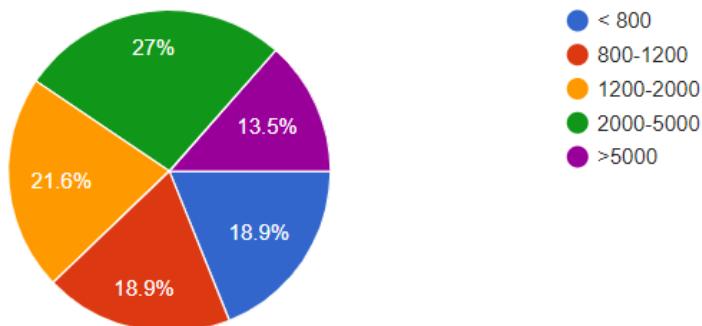
52 responses

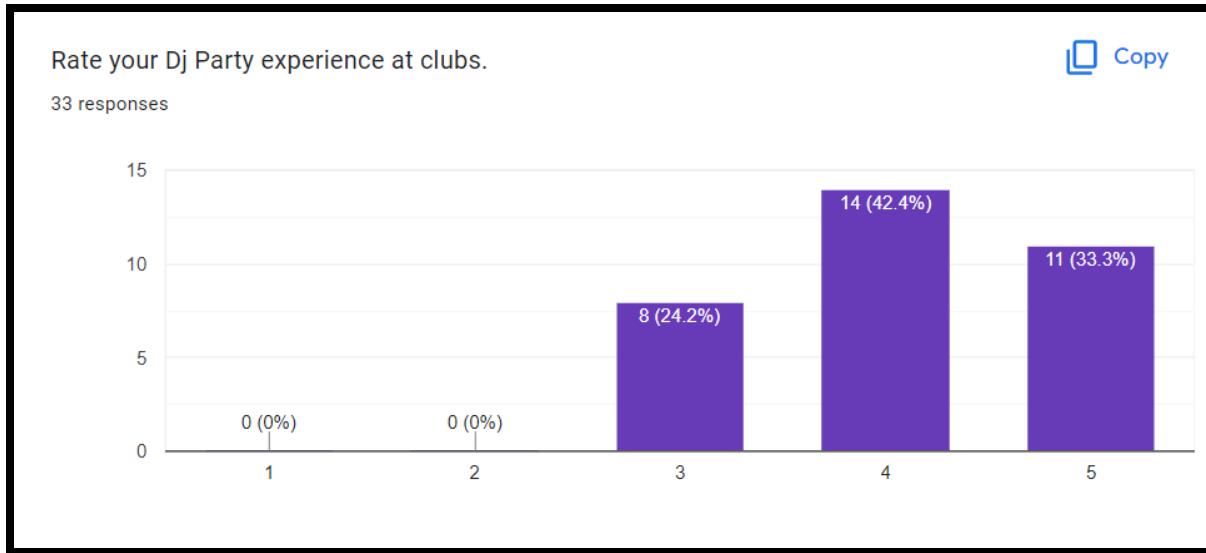


Please select your average cost of single visit at any club (in INR).



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-oriented).
- 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 membeber_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

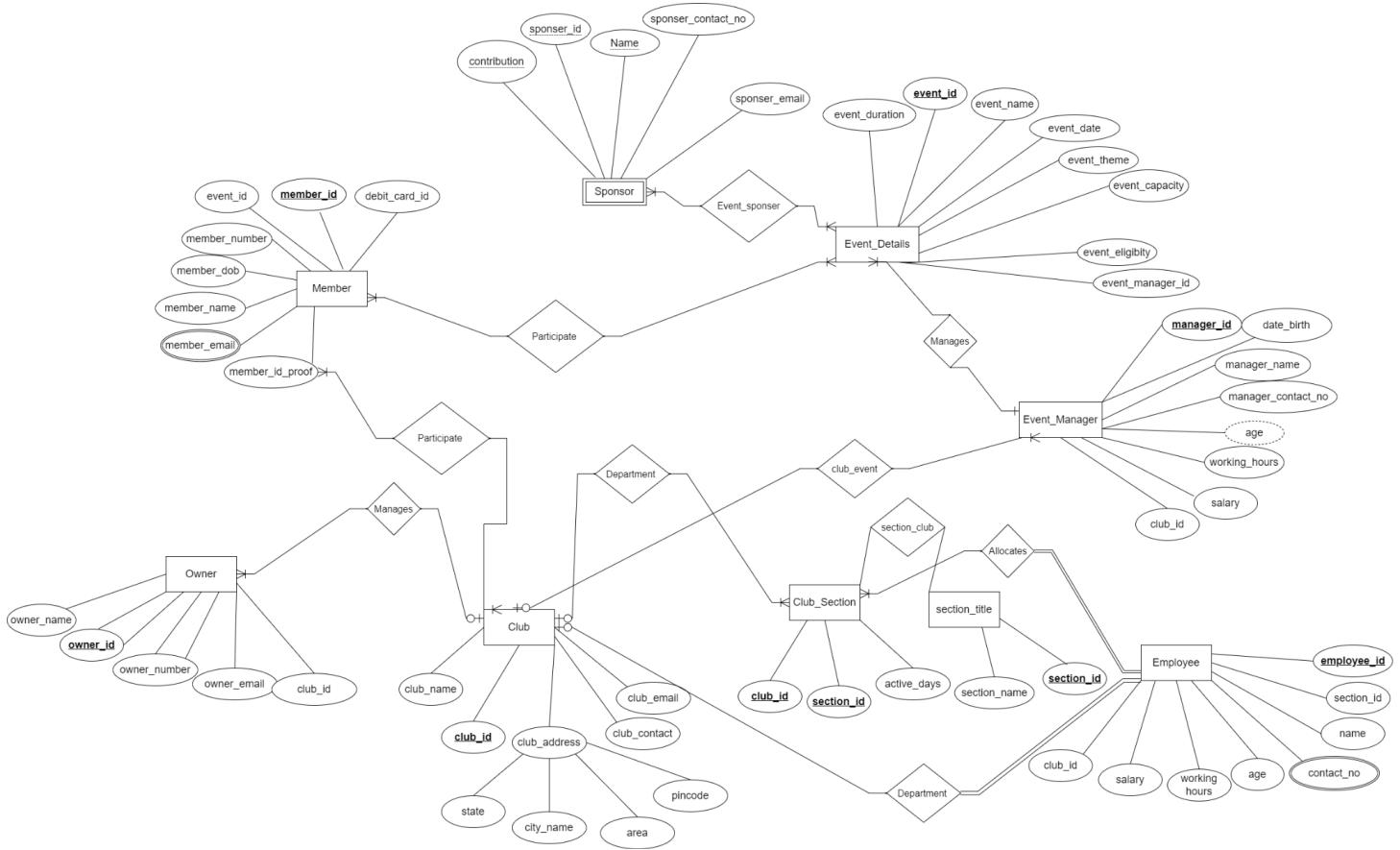
	Member_total_expense member_debit_card_id	
Event details	Event_id Event_manager_id Event_name Event_theme Event_type Event_date Event_start_time end_time Event_price_rates Event_eligibility Event_capacity Event_contact_detail Event_sponsors	
Employee	Employee_id Section_id Employee_role Employee_name 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	
worker	
Sweeper	
Bartender	
DJ party	They are all different type of worker required to run the club. We can group them in single attribute named employee_type
drink party	
user	It can be covered in party_type, no need of different attribute
customer	
cost	duplicate
price	
capacity	duplicate
Address	Attribute
Employee experience	irrelevant
Joining date	irrelevant
Event date	Attribute
Payment mode	Vague
Past experience	irrelevant
Details	general
Amount	Association
thing	Attributes
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 club_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:
 - $(\text{section_id}, \text{club_id}) \rightarrow (\text{active_days})$
 - 1NF \Rightarrow Yes as no multivalued/composite attribute
 - 2NF \Rightarrow Yes as no partial key is used
 - 3NF \Rightarrow Yes as no transitive dependency
 - BCNF \Rightarrow Yes as all determinants are super key
- This is lossless join decomposition as the common attribute Section_id in both tables is CK for section_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 \rightarrow (name, age, working_hours, salary, section_id)
- 1NF \Rightarrow Yes as no multivalued/composite attribute
- 2NF \Rightarrow Yes as no partial key is used
- 3NF \Rightarrow Yes as no transitive dependency
- BCNF \Rightarrow 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_eligibility, event_manager_id, contribution)
- PK is event_id

Constraints:

- Here sponsor_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_eligibility, event_manager_id, contribution)
 - 1NF ⇒ No as multivalued attribute sponsor
- ❖ Decomposition into
 - event_details(event_id, event_duration, event_name, event_date, event_theme, event_capacity, event_eligibility, event_manager_id)
 - Functional dependency:
 - Event_id → (event_duration, event_name, event_date, event_theme, event_capacity, event_eligibility, 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_sponsor(event_id, sponsor_id, 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(manager_id, event_id, date_birth, manager_name, manager_contact_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 \Rightarrow Yes as no partial key is used
- 3NF \Rightarrow Yes as no transitive dependency
- BCNF \Rightarrow 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:

- $\text{member_id} \rightarrow (\text{member_name}, \text{member_email}, \text{member_id_proof}, \text{debit_card_id}, \text{event_id}, \text{member_number}, \text{member_dob})$
- 1NF \Rightarrow 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:
 - $\text{member_id} \rightarrow (\text{member_name}, \text{member_email}, \text{member_id_proof}, \text{debit_card_id}, \text{member_number}, \text{member_dob})$
 - 1NF \Rightarrow Yes as no multivalued/composite attribute
 - 2NF \Rightarrow Yes as no partial key is used
 - 3NF \Rightarrow Yes as no transitive dependency
 - BCNF \Rightarrow Yes as all determinants are super key

- event_member(event_id, member_id)

- Functional dependency:
 - $(\text{event_id}, \text{member_id}) \rightarrow (\text{event_id}, \text{member_id})$
 - 1NF \Rightarrow Yes as no multivalued/composite attribute
 - 2NF \Rightarrow Yes as no partial key is used
 - 3NF \Rightarrow Yes as no transitive dependency

Final relations

1. Club \Rightarrow (**club_id**, club_name, club_email)
2. club_contact \Rightarrow (**club_id**,**club_contact**)
3. club_address \Rightarrow (**club_id**, pincode, city_name, state, area)
4. owner \Rightarrow (**owner_id**, **club_id**, owner_name, owner_number, owner_email)
5. Section_title \Rightarrow (**Section_id**, section_name)
6. section_days \Rightarrow (**section_id**, **club_id**, active_days)
7. employee \Rightarrow (**employee_id**, name, age, working_hours, salary, **section_id**,**club_id**)
8. event_manager \Rightarrow (**manager_id**, date_birth, manager_name, manager_email, working_hours, salary, **club_id**)
9. event_details \Rightarrow (**event_id**, event_duration, event_name, event_date, event_theme, event_capacity, event_eligibility, **manager_id**)
10. event_sponser \Rightarrow (**sponsor_id**, contribution, sponser_name, sponser_email, sponser_contact_no)
11. sponser_with_event \Rightarrow (**event_id**,**sponser_id**)
12. Member \Rightarrow (**member_id**, member_name, member_contact,member_email, member_id_proof, debit_card_id, member_dob)
13. member_with_event \Rightarrow (**member_id**,**event_id**)
14. Member_with_club \Rightarrow (**member_id**, **club_id**,membership_type, no_event_attended, member_penalty, money_spent)

2. Create DDL Scripts :

Create Table

1. Club \Rightarrow (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 \Rightarrow (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 \Rightarrow (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 \Rightarrow (**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 \Rightarrow (**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. $\text{section_days} \Rightarrow (\underline{\text{section_id}}, \underline{\text{club_id}}, \text{active_days})$

```

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. $\text{employee} \Rightarrow (\underline{\text{employee_id}}, \text{name}, \text{age}, \text{working_hours}, \text{salary}, \underline{\text{section_id}}, \underline{\text{club_id}})$

```

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 \Rightarrow (**event_id**, event_duration, event_name, event_date, event_theme, event_capacity, event_eligibility, 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 \Rightarrow (**sponser_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

Format

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

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Properties SQL Statistics q*

Browser

- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- Tables (11)
 - > club
 - > club_address
 - > club_contact
 - > employee
 - > event_details
 - > event_manager
 - > event_sponsor
 - > owner
 - > section_days
 - > section_title
 - > sponsor_with_event
- > Trigger Functions
- > Types
- > Views
- > pg_toast
- > public
- > Subscriptions
- > postgres
- > template1
- > template0
- > Login/Group Roles
- > Tablespaces

club/postgres@PostgreSQL 14

Query Query History

```
1 select * from clubbing.club
2
3
4
```

Data output Messages Notifications

club_id	club_name	club_email
1	Energizer Person	lcalbaith0@colu...
2	Laboratoire der...	cdeville1@vk.co...
3	General Injectab...	cmanhire2@live...
4	Rebel Distributor...	bsones3@phoca...
5	Taro Pharmaceut...	hcarslake4@pay...
6	Cover FX Skin Ca...	kchristy5@cbsne...
7	Allermed Laborat...	hmckennan6@us...
8	Natural Health S...	moland7@virgini...

Total rows: 50 of 50 Query complete 00:00:00.071 Ln 4, Col 1

2. find all available contact_no of club_id=1

select * from clubbing.club_contact where club_id=1

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Properties SQL Statistics q*

Browser

- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- Tables (11)
 - > club
 - > club_address
 - > club_contact
 - > employee
 - > event_details
 - > event_manager
 - > event_sponsor
 - > owner
 - > section_days
 - > section_title
 - > sponsor_with_event
- > Trigger Functions
- > Types
- > Views
- > pg_toast
- > public
- > Subscriptions
- > postgres
- > template1
- > template0
- > Login/Group Roles
- > Tablespaces

club/postgres@PostgreSQL 14

Query Query History

```
1 select * from clubbing.club_contact where club_id=1
2
3
4
```

Data output Messages Notifications

club_id	club_contact
1	1139269094
2	6444114597
3	9929768980

Total rows: 3 of 3 Query complete 00:00:00.074 Ln 1, Col 52

3. find information of clubs which are located in Texas

select * from clubbing.club_address where stat='Texas'

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects: FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (11), and several specific tables like club, club_address, club_contact, employee, event_details, event_manager, event_sponsor, owner, section_days, section_title, sponsor_with_event, Trigger Functions, Types, Views, pg_toast, public, Subscriptions, postres, template1, template0, Login/Group Roles, and Tablespaces. The 'owner' table is currently selected. The top navigation bar includes File, Object, Tools, and Help. The main area has tabs for Properties, SQL, Statistics, and a search bar. A query editor window is open with the following SQL query:

```
1 select * from clubbing.club_address where stat='Texas'
```

The 'Data output' tab shows the results of the query:

club_id	[PK] integer	pincode	character varying	stat	character varying	area	character varying
1	18	54	San Antonio	Texas	2	Autumn Leaf P...	
2	20	27	Fort Worth	Texas	515	Northridge P...	
3	22	6240	Arlington	Texas	60225	Mayer Jun...	
4	48	4056	Austin	Texas	0	4th Court	

Total rows: 4 of 4 Query complete 00:00:00.074 Ln 2, Col 1

4. find owner information of club_id=1

```
select * from clubbing.owner where club_id=1
```

The screenshot shows the pgAdmin 4 interface, similar to the previous one but with a different query. The left sidebar is identical. The top navigation bar includes File, Object, Tools, and Help. The main area has tabs for Properties, SQL, Statistics, and a search bar. A query editor window is open with the following SQL query:

```
1 select * from clubbing.owner where club_id=1
```

The 'Data output' tab shows the results of the query:

owner_id	[PK] integer	owner_name	character varying	owner_number	bigint	owner_email	character varying	club_id	integer
1	1	Kial		7954368752		kogdell0@time...		1	

Total rows: 1 of 1 Query complete 00:00:00.045 Ln 2, Col 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
```

PgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

- > A FTS Parsers
- > B FTS Templates
- > C Foreign Tables
- > D Functions
- > E Materialized Views
- > F Operators
- > G Procedures
- > H Sequences
- > I Tables (11)
 - > club
 - > club_address
 - > club_contact
 - > employee
 - > event_details
 - > event_manager
 - > event_sponsor
 - > owner
 - > section_days
 - > section_title
 - > sponsor_with_event
- > J Trigger Functions
- > K Types
- > L Views
- > M pg_toast
- > N public
- > O Subscriptions
- > P postgres
- > Q template1
- > R template0

Properties SQL Statistics q*

club/postgres@PostgreSQL 14

Query Query History

```
1 select * from clubbing.section_title where section_id=1 or section_id =2 or section_id=3
2 |
```

Data output Messages Notifications

section_id	section_name
1	drink
2	music
3	dance

Total rows: 3 of 3 Query complete 00:00:08.919 Ln 2, Col 1

6. find club_id and section_id, which are active on only sunday

select * from clubbing.section_days where active_days='1000000'

PgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

- > A FTS Parsers
- > B FTS Templates
- > C Foreign Tables
- > D Functions
- > E Materialized Views
- > F Operators
- > G Procedures
- > H Sequences
- > I Tables (11)
 - > club
 - > club_address
 - > club_contact
 - > employee
 - > event_details
 - > event_manager
 - > event_sponsor
 - > owner
 - > section_days
 - > section_title
 - > sponsor_with_event
- > J Trigger Functions
- > K Types
- > L Views
- > M pg_toast
- > N public
- > O Subscriptions
- > P postgres
- > Q template1
- > R template0

Properties SQL Statistics q*

club/postgres@PostgreSQL 14

Query Query History

```
1 select * from clubbing.section_days where active_days='1000000'
```

Data output Messages Notifications

section_id	club_id	active_days
1	1	1000000
2	6	14
3	1	31
4	6	43

Total rows: 4 of 4 Query complete 00:00:00.114 Ln 1, Col 64

7. find employee information of club_id=1

select * from clubbing.employee where club_id=1

pgAdmin 4

File Object Tools Help

Properties SQL Statistics q*

club/postgres@PostgreSQL 14

No limit

Query History

```
1 select * from clubbing.employee where club_id=1
```

Data output Messages Notifications

	employee_id	name	age	working_hours	salary	section_id	club_id
1	92	Mar	35	9	32000	4	1
2	116	Dot	25	5	32000	4	1
3	119	Madelin	36	5	32000	9	1
4	147	Eina	23	3	50000	8	1
5	155	Gunther	29	8	23000	3	1
6	166	Kesley	52	5	23000	10	1
7	190	Ximenes	44	8	23000	9	1
8	210	Jeffrey	23	3	34000	7	1

Total rows: 17 of 17 Query complete 00:00:00.072 Ln 1, Col 48

8. find manager details of club_id=34

```
select * from clubbing.event_manager where club_id=34
```

pgAdmin 4

File Object Tools Help

Properties SQL Statistics q*

club/postgres@PostgreSQL 14

No limit

Query History

```
1 select * from clubbing.event_manager where club_id=34
2
```

club_id date_birth manager_id manager_name member_email working_hours salary club_id

	manager_id	date_birth	manager_id	manager_name	member_email	working_hours	salary	club_id
1	34	1992-12-02	Maud	Maud.Stilwell@y...	7	65000	34	
2	51	1986-06-23	Marcelline	Marcelline.Geora...	7	65000	34	
3	62	1995-06-06	Consuela	Consuela.Ovid@y...	9	65000	34	
4	74	1998-10-23	Cam	Cam.Cle@yopm...	7	76000	34	

Total rows: 4 of 4 Query complete 00:00:00.051 Ln 2, Col 1

9. find details of event which have event_eligibility>18

```
select * from clubbing.event_details where event_eligibility>18
```

event_id	event_duration	event_name	event_date	event_theme	event_capacity	event_eligibility	manager_id
1	8	2	2010-11-09	CCEAA	37	20	59
2	14	3	2017-12-13	CEGBB	72	20	44
3	15	3	2016-12-03	CAGCG	84	21	21
4	17	2	2014-12-06	FBGEB	40	21	38
5	18	1	2015-02-11	AEGBB	59	20	68
6	19	4	2017-07-14	FCDGE	40	19	29
7	22	4	2008-08-09	HFACE	95	20	12
8	23	1	2012-08-13	AGEGD	71	21	36

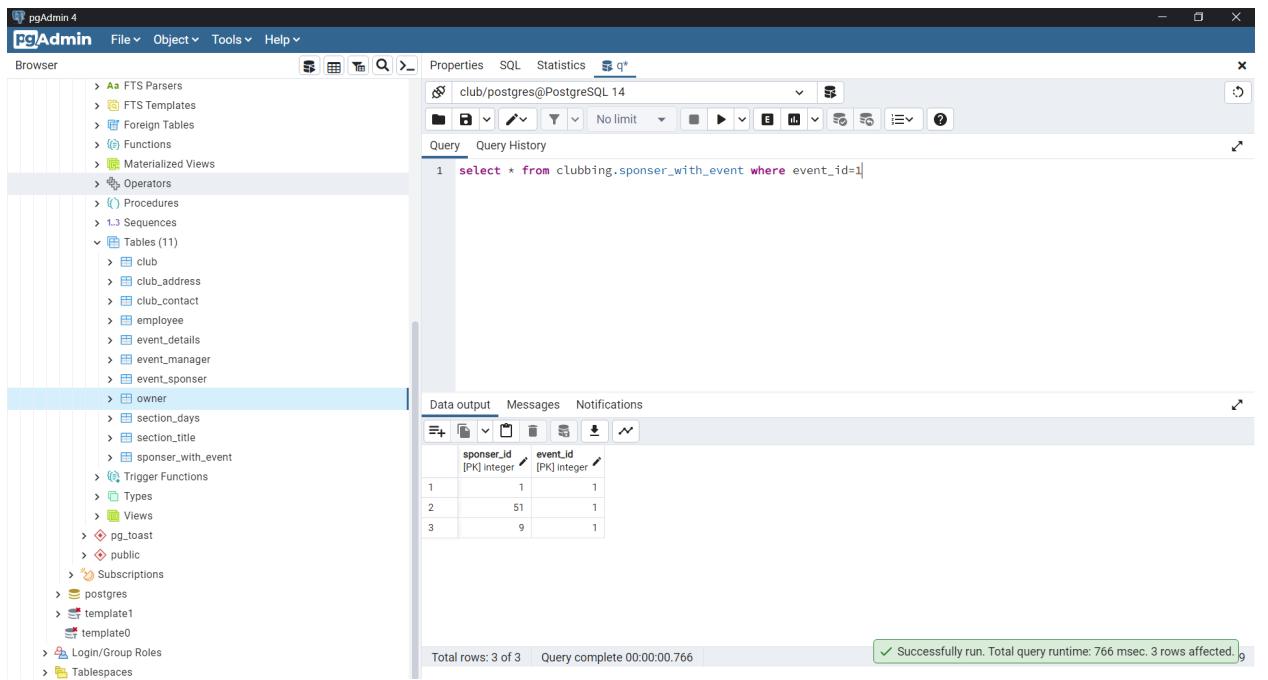
10. find information of event_sponser who have contribution>50000

```
select * from clubbing.event_sponser where contribution>50000
```

sponser_id	sponser_name	sponser_contact_no	sponser_email	contribution
1	Annora	8299821602	Annora@yopp.co...	53000
2	Jinny	8910948580	Jinny@yopp.com	76000
3	Frieda	8975198911	Frieda@yopp.com	65000
4	Marti	8674314036	Marti@yopp.com	53000
5	Beverley	8773885217	Beverley@yopp.c...	64000
6	Georgina	8014519454	Georgina@yopp...	53000
7	Tina	8915753101	Tina@yopp.com	76000
8	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
```



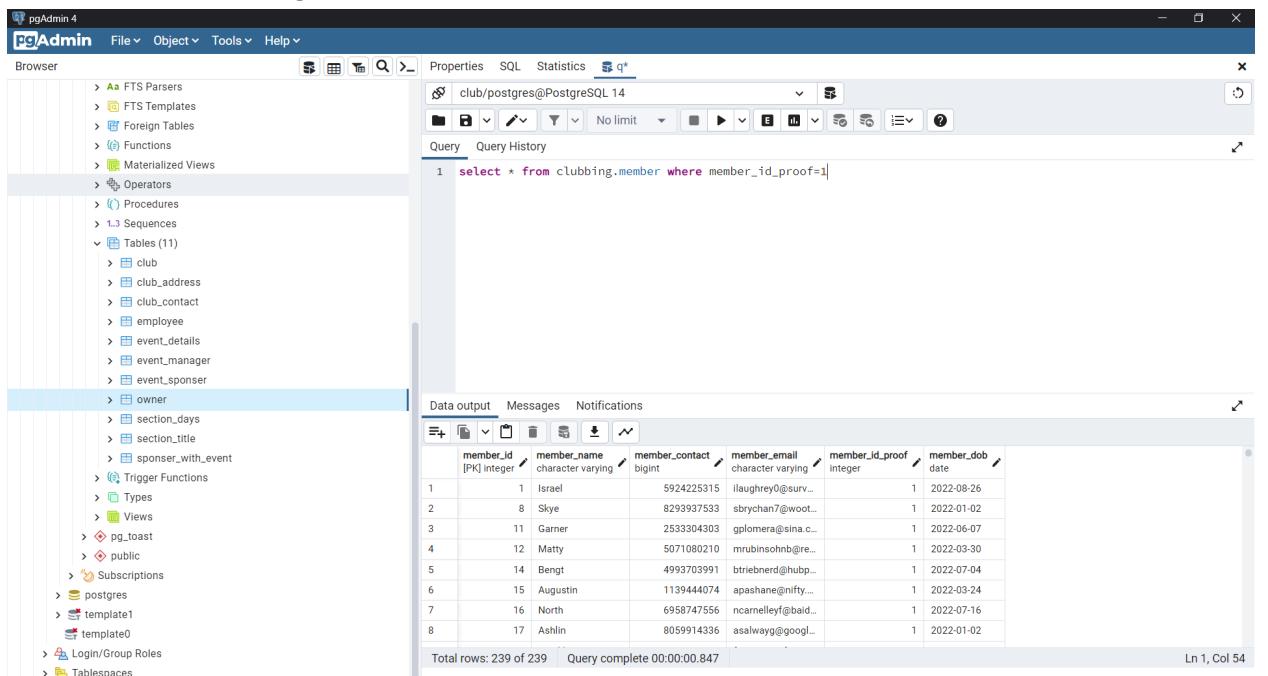
The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database schema with the **Tables (11)** node expanded, revealing tables like **club**, **club_address**, **club_contact**, etc.
- Properties:** Tab selected in the top bar.
- SQL:** Tab selected in the top bar.
- Statistics:** Tab selected in the top bar.
- Query:** The query `select * from clubbing.sponser_with_event where event_id=1` is entered.
- Data output:** Tab selected in the bottom bar.
- Table:** A result table with columns `sponser_id` and `event_id`. The data is:

	sponser_id	event_id
1	1	1
2	51	1
3	9	1
- Messages:** Tab selected in the bottom bar.
- Notifications:** Tab selected in the bottom bar.

12. find the member list which have attached their id proofs

```
select * from clubbing.member where member_id_proof=1
```



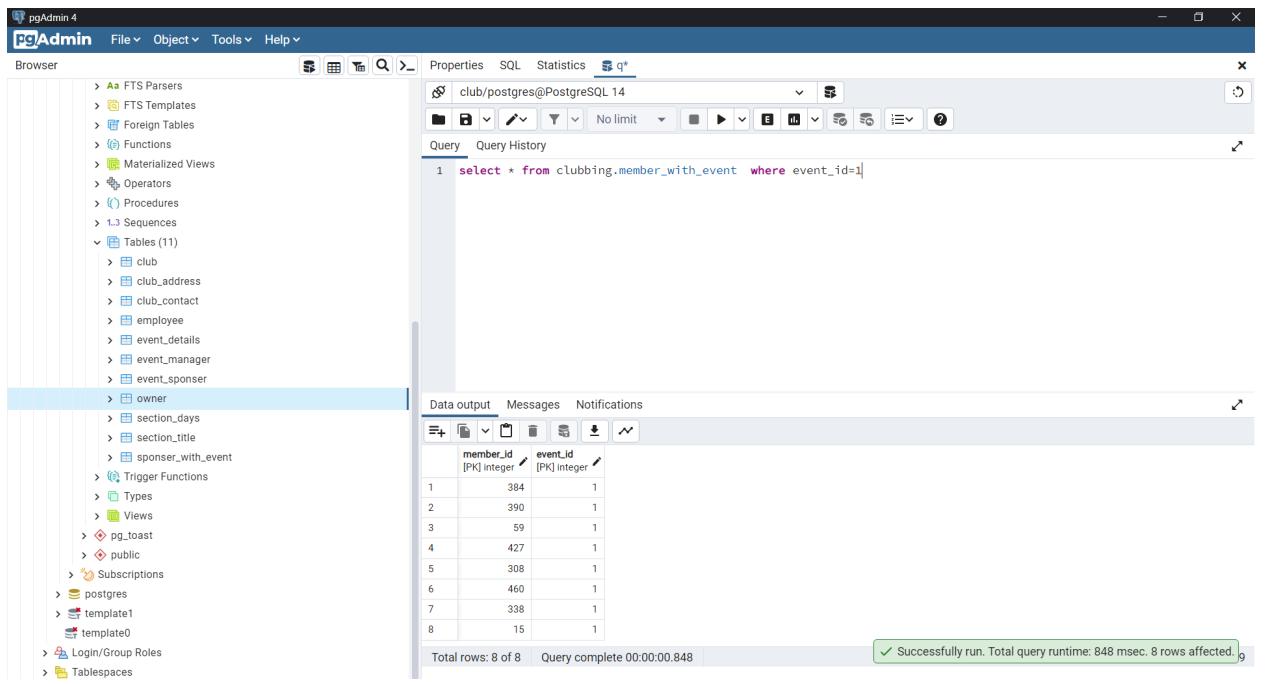
The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database schema with the **Tables (11)** node expanded, revealing tables like **club**, **club_address**, **club_contact**, etc.
- Properties:** Tab selected in the top bar.
- SQL:** Tab selected in the top bar.
- Statistics:** Tab selected in the top bar.
- Query:** The query `select * from clubbing.member where member_id_proof=1` is entered.
- Data output:** Tab selected in the bottom bar.
- Table:** A result table with columns `member_id`, `member_name`, `member_contact`, `member_email`, `member_id_proof`, and `member_dob`. The data is:

	member_id	member_name	member_contact	member_email	member_id_proof	member_dob
1	1	Israel	5924225315	ilaughrey0@surv...	1	2022-08-26
2	8	Skye	8293937533	sbrychan7@woot...	1	2022-01-02
3	11	Garner	2533304303	gplomera@sina.c...	1	2022-06-07
4	12	Matty	5071080210	mrubinsonhb@re...	1	2022-03-30
5	14	Bentg	4993703991	btriebnerd@hubp...	1	2022-07-04
6	15	Augustin	1139444074	apashane@nify...	1	2022-03-24
7	16	North	6958747556	ncarmelleyf@bald...	1	2022-07-16
8	17	Ashlin	8059914336	asalwayg@google...	1	2022-01-02
- Messages:** Tab selected in the bottom bar.
- Notifications:** Tab selected in the bottom bar.

13. find the member who had attended event with event_id=1

```
select * from clubbing.member_with_event where event_id=1
```



The screenshot shows the pgAdmin 4 interface with the following details:

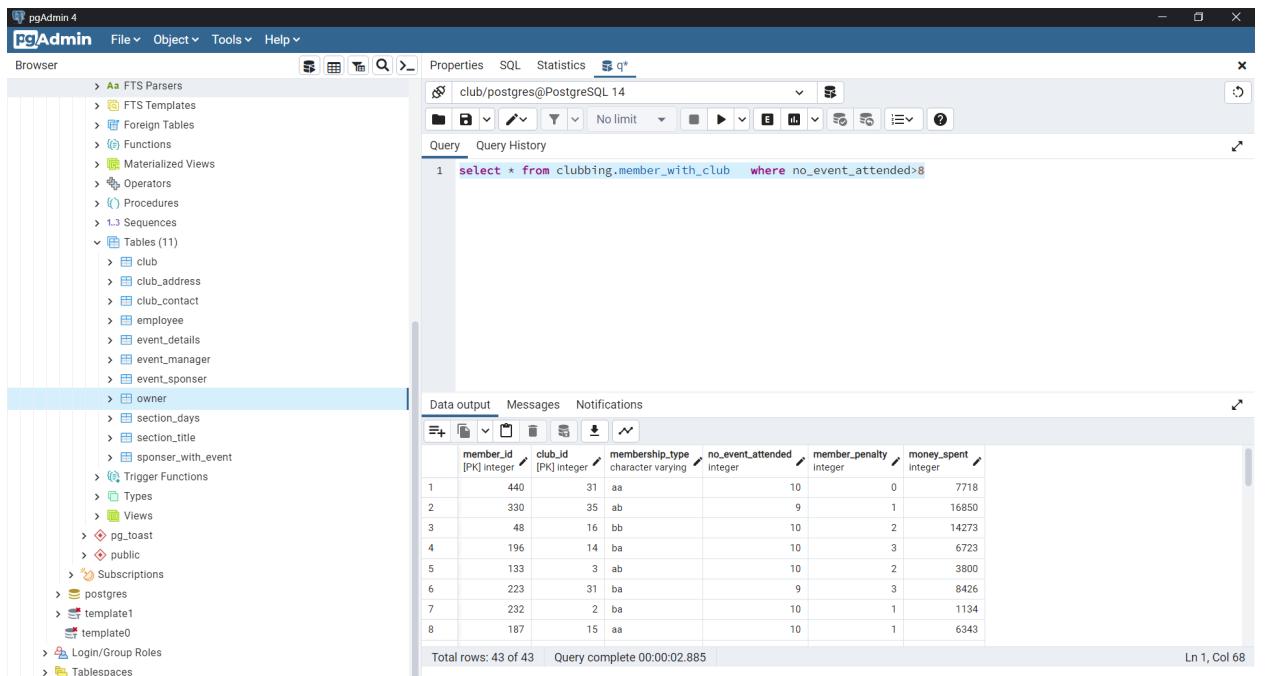
- Browser:** Shows the database schema with the "owner" table selected.
- Properties:** Shows the connection properties: club/postgres@PostgreSQL 14.
- SQL:** The query entered: `select * from clubbing.member_with_event where event_id=1`.
- Data output:** A table showing the results of the query. The columns are `member_id` and `event_id`. The data consists of 8 rows:

member_id	event_id
1	384
2	390
3	59
4	427
5	308
6	460
7	338
8	15

- Messages:** Shows a success message: "Successfully run. Total query runtime: 848 msec. 8 rows affected."

14. find the member details with no_event_attended>8

```
select * from clubbing.member_with_club where no_event_attended>8
```



The screenshot shows the pgAdmin 4 interface with the following details:

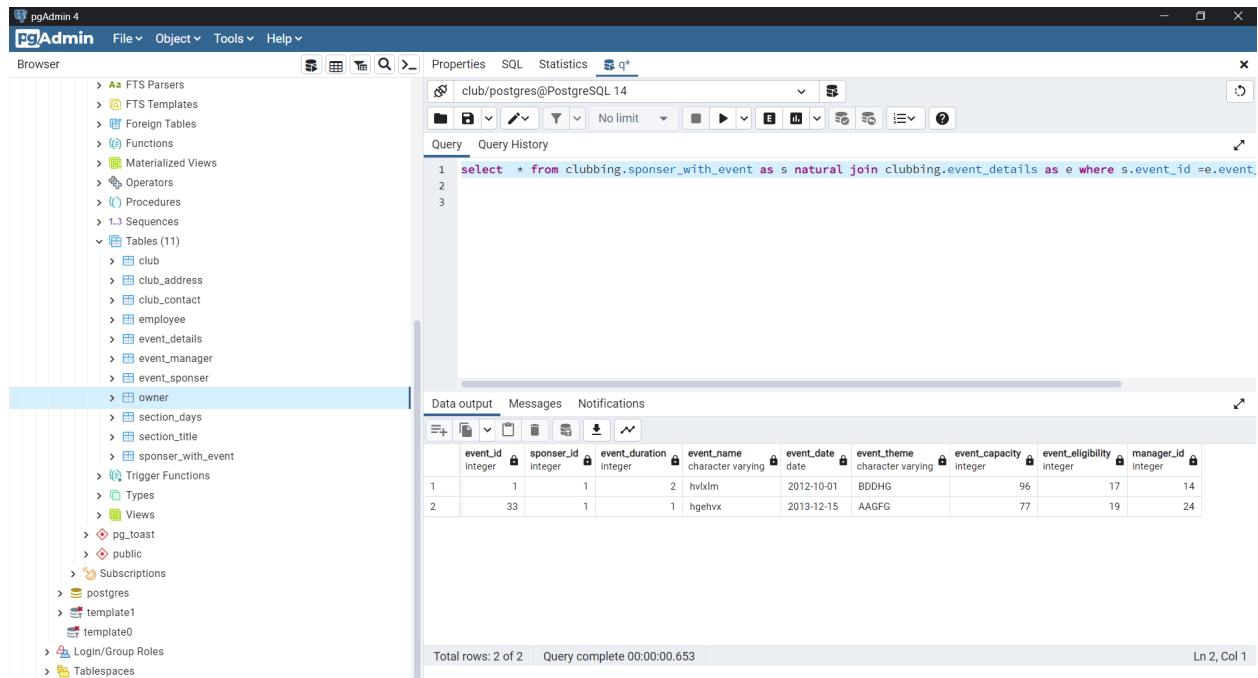
- Browser:** Shows the database schema with the "owner" table selected.
- Properties:** Shows the connection properties: club/postgres@PostgreSQL 14.
- SQL:** The query entered: `select * from clubbing.member_with_club where no_event_attended>8`.
- Data output:** A table showing the results of the query. The columns are `member_id`, `club_id`, `membership_type`, `no_event_attended`, `member_penalty`, and `money_spent`. The data consists of 8 rows:

member_id	club_id	membership_type	no_event_attended	member_penalty	money_spent
1	440	31 aa	10	0	7718
2	330	35 ab	9	1	16850
3	48	16 bb	10	2	14273
4	196	14 ba	10	3	6723
5	133	3 ab	10	2	3800
6	223	31 ba	9	3	8426
7	232	2 ba	10	1	1134
8	187	15 aa	10	1	6343

- Messages:** Shows a success message: "Successfully run. Total query runtime: 288 msec. 8 rows affected."

Complex Queries

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



The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, including FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (11), and various system and schema objects. The main window shows a query editor with the following SQL statement:

```
1 select * from clubbing.sponser_with_event as s natural join clubbing.event_details as e where s.event_id=e.event
```

The results pane displays a table with two rows of data:

event_id	sponser_id	event_duration	event_name	event_date	event_theme	event_capacity	event_eligibility	manager_id
1	1	1	2 hvifxm	2012-10-01	BDDHG	96	17	14
2	33	1	1 hgehv	2013-12-15	AAGFG	77	19	24

At the bottom of the results pane, it says "Total rows: 2 of 2 Query complete 00:00:00.653 Ln 2, Col 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
```

section_name	active_days
drink	1000000
music	1000001
dance	1000010

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)
```

event_id	event_duration	event_name	event_date	event_theme	event_capacity	event_eligibility	manager_id
19	4	jtgcgb	2014-07-14	FCDGE	40	19	29
44	1	jcpcfh	2014-04-27	FDABF	77	17	35
97	2	pyflxz	2019-01-02	HAHGE	94	16	67

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)
```

	event_id [PK] integer	event_duration	event_name	event_date	event_theme	event_capacity	event_eligibility	manager_id
1	1	2	hvixlm	2012-10-01	BDDHG	96	17	14
2	2	3	mrtguk	2011-03-11	ACHCF	66	16	77
3	3	4	podotm	2011-06-22	HGCCF	30	18	77
4	4	4	uiksfz	2018-08-10	BCHGD	99	18	11
5	5	2	zdiikc	2019-10-07	BCGGF	43	18	2
6	6	2	nhtvpx	2011-03-09	ABBFH	88	17	6
7	7	3	nelwyp	2014-02-07	BDECH	86	17	40
8	8	2	krambg	2010-11-09	CCEAA	37	20	59

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)
```

```
9  SELECT event_id into g_pid from
10 clubbing.event_details g1 where
11 g1.event_id=NEW.event_id;
12 if(g.pid=NEW.event_id) THEN
13 RAISE NOTICE 'Violating condition:id already exists';
14 else
15 RAISE NOTICE 'no duplication';
16 RETURN NEW;
17 end if;
18 END
19 $BODY$
20
21
22 insert into clubbing.event_details values(1,10,'kama','12-02-2022','ab',60,18,1)
```

NOTICE: violating condition:id already exists
ERROR: control reached end of trigger procedure without RETURN
CONTEXT: PL/pgSQL function clubbing."Trig"()
SQL state: 2F005

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
```

The screenshot shows the pgAdmin 4 interface. The left pane is the 'Browser' showing the schema structure of the 'clubbing' database, including tables like event_duration, event_name, event_date, and various triggers such as event_trig and member_trigger. The right pane is the 'Query' editor with the SQL command 'select * from clubbing.employee'. Below it is the 'Data output' pane displaying the results of the query:

employee_id	name	age	working_hours	salary	section_id	club_id	updated
1	Casie	23	9	20000	9	23	11
2	Izabel	35	4	30000	4	32	6
3	Brigitta	37	9	50000	9	30	11
4	Lonnie	37	9	12000	5	13	11
5	Lamar	49	6	21000	10	34	8
6	Charlotta	52	3	43000	5	7	5
7	Whittaker	39	3	45000	4	6	5
8	Easter	38	7	23000	1	30	9

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
```

```

3
4 CREATE OR REPLACE function clubbing.fun1()
5 RETURNS TABLE (a double precision)
6 LANGUAGE 'plpgsql'
7 AS $BODY$
8 BEGIN
9 update clubbing.event_sponser set updated=contribution*2;
10 RETURN QUERY EXECUTE format ('SELECT updated FROM clubbing.event_sponser');
11 END;
$BODY$;
13
14 select clubbing.fun1()
15 select * from clubbing.event_sponser
16

```

	sponsor_id [PK] integer	sponsor_name character varying	sponsor_contact_no bigint	sponsor_email character varying	contribution integer	updated double precision
1	16	Karina	8446160021	Karina@yopp.com	180008	360016
2	17	Bertine	8789300046	Bertine@yopp.co...	96008	192016
3	18	Myriam	8804536824	Myriam@yopp.co...	136008	272016
4	19	Marti	8674314036	Marti@yopp.com	212008	424016
5	20	Sue	8611253795	Sue@yopp.com	128008	256016
6	21	Hayley	8812268933	Hayley@yopp.com	92008	184016
7	22	Fidella	8208935345	Fidella@yopp.com	92008	184016
8	23	Edee	8654170160	Edee@yopp.com	128008	256016

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')

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a database browser with several tables listed under the schema 'club'. The table 'member' is currently selected. The right panel contains a query editor with the following SQL code:

```
8 BEGIN
9  SELECT member_id into g_pid from
10 clubbing.member g1 where
11 g1.member_id=NEW.member_id;
12 if(g_pid=NEW.member_id) THEN
13 RAISE NOTICE 'violating condition:id already exists';
14 else
15 RAISE NOTICE 'no duplication';
16 RETURN NEW;
17 end if;
18 END
19 $BODY$
```

Below the code, the 'Data output' tab is active, showing the message: "NOTICE: violating condition:id already exists". The 'Messages' tab shows an error: "ERROR: control reached end of trigger procedure without RETURN". The 'Notifications' tab is also present.

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
```

pgAdmin 4

File Object Tools Help

Browser

```

event_duration
event_name
event_date
event_theme
event_capacity
event_eligibility
manager_id
> Constraints
> Indexes
> RLS Policies
> Rules
> Triggers (1)
  > event_trig
> event_manager
> event_sponsor
member
  > Columns
  > Constraints
  > Indexes
  > RLS Policies
  > Rules
  > Triggers (1)
    > member_trigger
> member_with_club
> member_with_event
owner
section_days
section_title
sponser_with_event
Trigger Functions (1)
  (Trig0)

```

Properties SQL Statistics q*

club/postgres@PostgreSQL 14

No limit

Query History

```

3
4 CREATE OR REPLACE function clubbing.fun2()
5 RETURNS TABLE (a double precision)
6 LANGUAGE 'plpgsql'
7 AS $BODY$
8 BEGIN
9 update clubbing.event_details set updated=event_capacity/2;
10 RETURN QUERY EXECUTE format ('SELECT updated FROM clubbing.event_details');
11 END;
$BODY$;
12
13
14 select clubbing.fun2()
15 select * from clubbing.event_details
16

```

Data output Messages Notifications

event_id	event_duration	event_name	event_date	event_theme	event_capacity	event_eligibility	manager_id	updated
1	1	2	2012-10-01	BDDHG	96	17	14	48
2	2	3	2019-03-11	ACHCF	66	16	77	33
3	3	4	2011-06-22	HGCF	30	18	77	15
4	4	4	2018-08-10	BCHGD	99	18	11	49
5	5	2	2019-10-07	BCGGF	43	18	2	21
6	6	2	2019-03-09	ABBHF	88	17	6	44
7	7	3	2014-02-07	BDECH	86	17	40	43
8	8	2	2010-11-09	CCEAA	37	20	59	18

Total rows: 100 of 100 Query complete 00:00:00.056 Ln 19, Col 1

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)
```

pgAdmin 4

File Object Tools Help

Browser

```

Servers (1)
PostgreSQL 14
  Databases (6)
    2020010118_db
    Courses
    club
      Casts
      Catalogs
      Event Triggers
      Extensions
      Foreign Data Wrappers
      Languages
      Publications
      Schemas (3)
        clubbing
          Aggregates
          Collations
          Domains
          FTS Configurations
          FTS Dictionaries
          FTS Parsers
          FTS Templates
          Foreign Tables
          Functions
          Materialized Views
          Operators
          Procedures
          Sequences
          Tables
          Trigger Functions
          Types

```

Properties SQL Statistics q*

club/postgres@PostgreSQL 14

No limit

Query History

```

1 select event_theme from clubbing.event_details where event_id in (select event_id from clubbing.member_with_event v
2 |
```

Data output Messages Notifications

event_theme
CCEAA
HGCGD
FCDGE
HHEHD
FDAFB
CEAEF
EEGEB
ACBDB

Total rows: 10 of 10 Query complete 00:00:00.163 Ln 2, Col 1

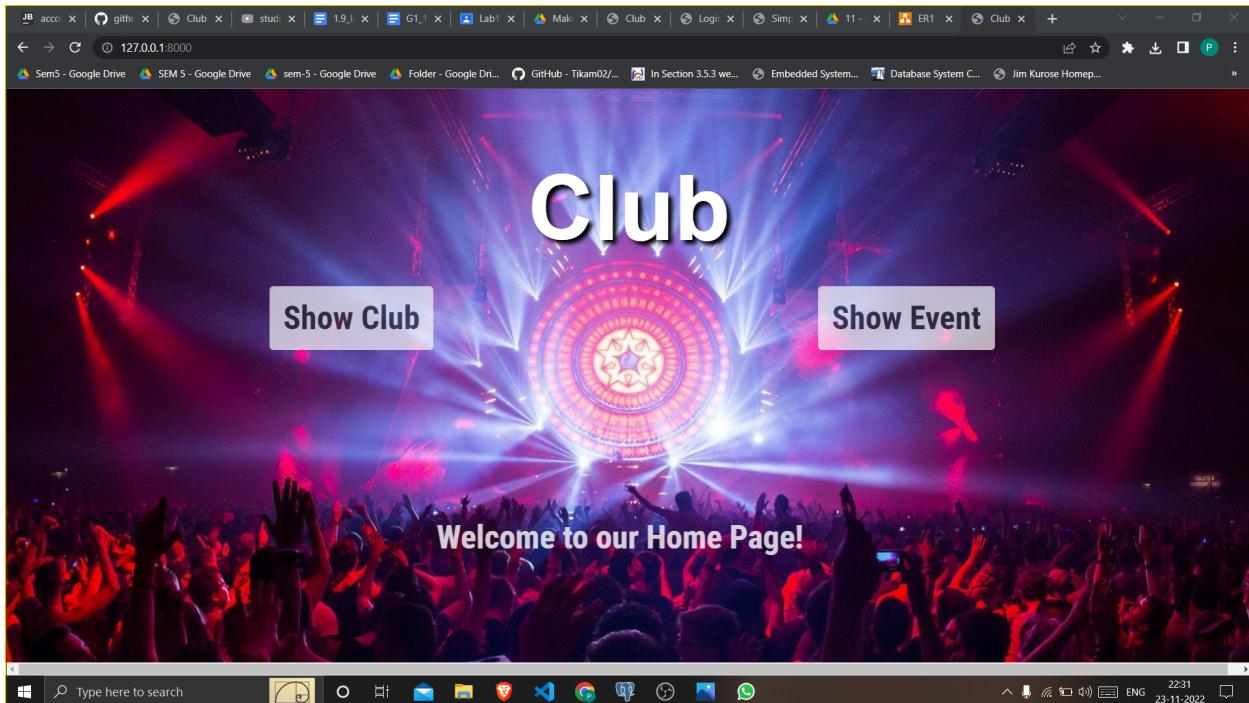
```
UPDATE clubbing.club  
SET club_email= '1@columbia.edu'  
WHERE club_id=1;
```

```
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

Show Club Details

Club ID	Club Name	Club Email	Edit	Delete
13	Aptalis Pharma US, Inc.	tkerswellc@youku.com	Edit	Delete
14	Aurobindo Pharma Limited	jgiraudoux@statcounter.com	Edit	Delete
15	Energizer Personal Care LLC	tburton@cloudflare.com	Edit	Delete
16	Kroger Company	abulfieldf@soundcloud.com	Edit	Delete
17	ALK-Abello, Inc.	shampeg@nbcnews.com	Edit	Delete
18	Cardinal Health	fmayhewh@yolasite.com	Edit	Delete

Properties **SQL** **Statistics**

```

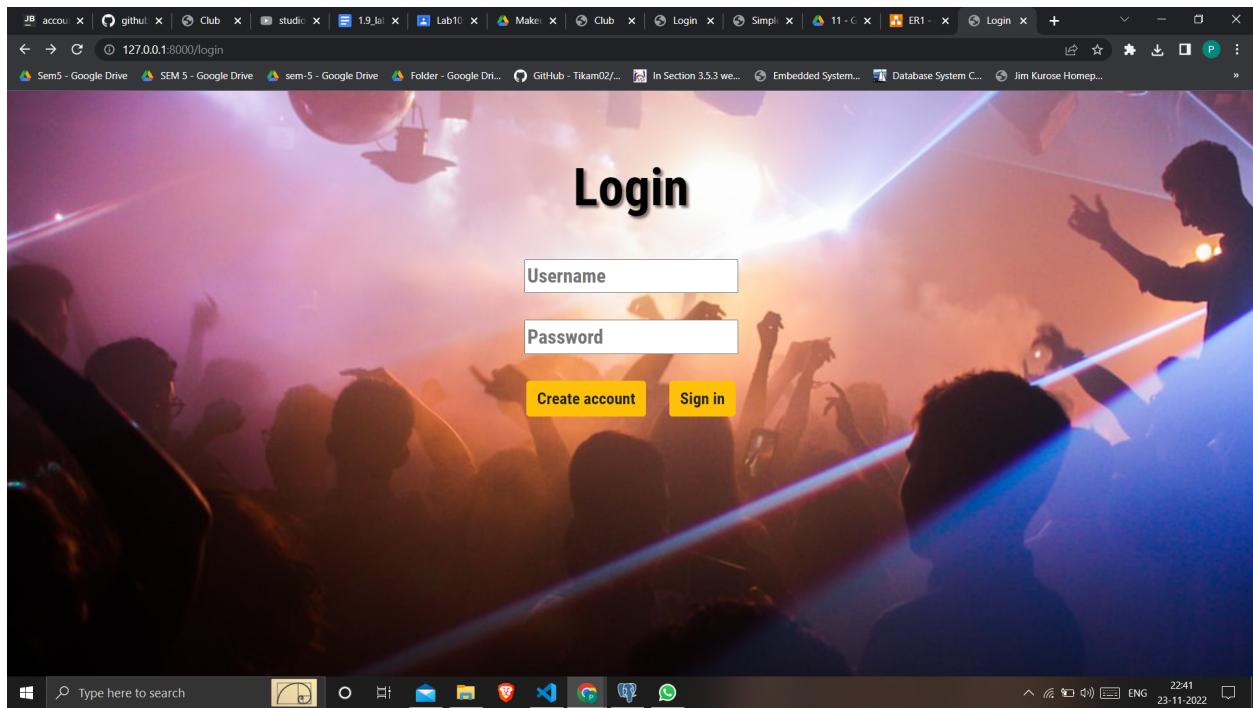
1 select * from club
2
3 select * from club_address
4
5 select * from event_details

```

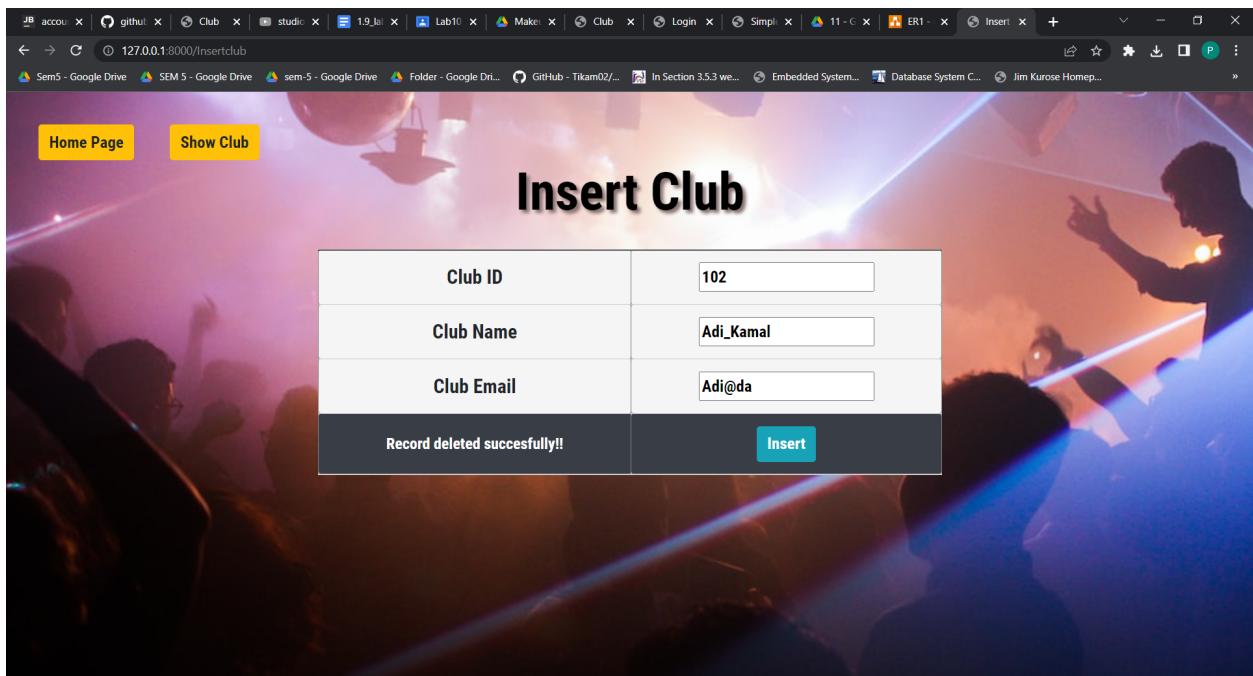
club_id	club_name	club_email
13	Aptalis Pharma US, Inc.	tkerswellc@youku.com
14	Aurobindo Pharma Limited	jgiraudoux@statcounter.com
15	Energizer Personal Care LLC	tburton@cloudflare.com
16	Kroger Company	abulfieldf@soundcloud.com
17	ALK-Abello, Inc.	shampeg@nbcnews.com
18	Cardinal Health	fmayhewh@yolasite.com
19	Cubist Pharmaceuticals, Inc.	cfermani@mysql.com
20	GlaxoSmithKline LLC	pwidger@com.com
21	PD-Rx Pharmaceuticals, Inc.	liansak@domaininn.com

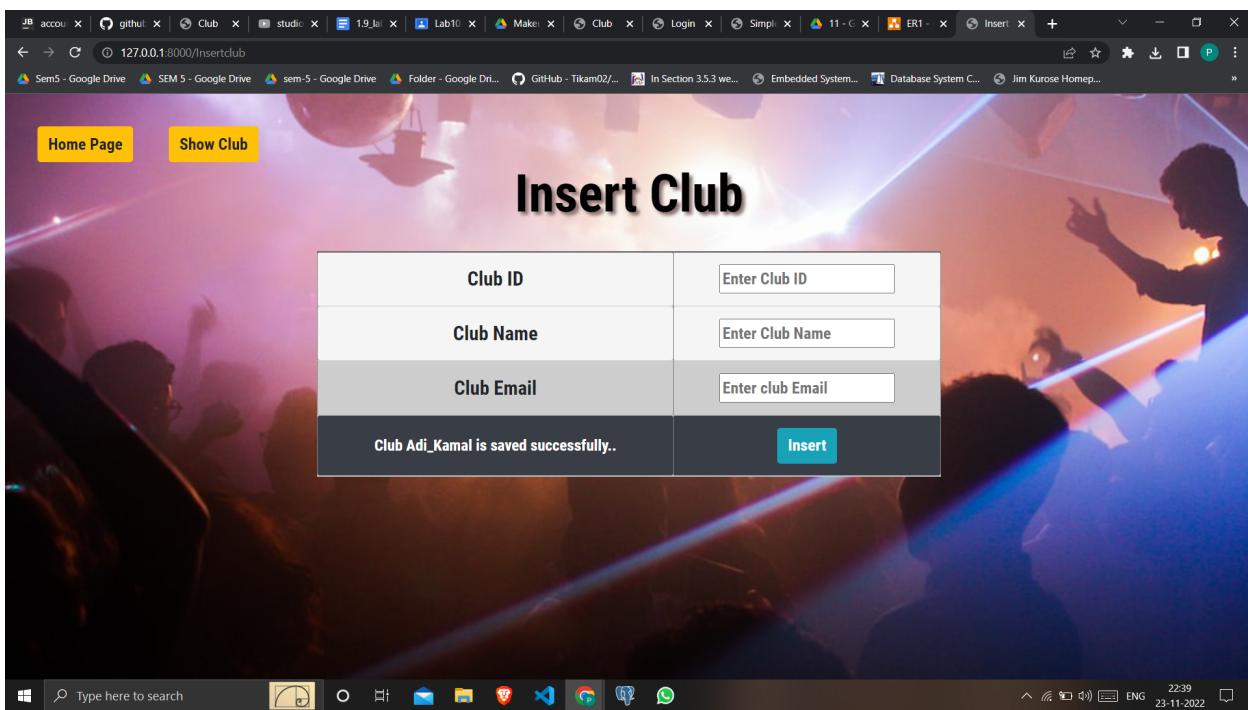
Total rows: 38 of 38 Query complete 00:00:00.054 Successfully run. Total query runtime: 54 msec. 38 rows affected.

3. Login Page in Order to Insert New Club



4. Insert Page





The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema, including tables like auth_group, 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, and others. The central pane shows a SQL query editor with the following code:

```
1 select * from club
2
3 select * from club_address
4
5 select * from event_details
```

The right pane shows the 'Data output' results for the first query, listing 39 rows of club data. The columns are club_id, club_name, and club_email. The data includes entries such as Kamal, harshil, Vandana, kamal988, zc, nknsejk, and Adi_Kamal. The club_email column contains various email addresses like Kamal@su.com, (club.club_email), haTahar, updated@gmail.com, vand@hiya, kamal988@gmail.com, sa, jknerjknjknjenjkern, and Adi@da.

5. Club Details in sorted Order according to Name

The screenshot shows a web browser window with multiple tabs open at the top. The active tab is titled "127.0.0.1:8000/sortclub". The page content features a banner with a night club scene and the text "Show Club Data - Sorted". Below the banner is a search bar with dropdown menus for "Club Name" and "Sort". A table lists club details with columns for club ID, club name, club email, edit, and delete options. The data is sorted by club name.

club ID	Club Name	Club Email	Edit	Delete
102	Adi_Kamal	Adi@da	Edit	Delete
17	ALK-Abello, Inc.	shampeg@nbcnews.com	Edit	Delete
31	Antigen Laboratories, Inc.	dgoodyeru@vkontakte.ru	Edit	Delete
22	Apotex Corp.	tfigurel@imdb.com	Edit	Delete
13	Aptalis Pharma US, Inc.	tkerswellc@youku.com	Edit	Delete
14	Aurobindo Pharma Limited	jgiraudoux@statcounter.com	Edit	Delete

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

The screenshot shows a web browser window with multiple tabs open at the top. The active tab is titled "127.0.0.1:8000/showaddress". The page content features a banner with a night club scene and the text "All Club Address". Below the banner is a search bar with dropdown menus for "Club ID" and "Sort". A table lists club addresses with columns for club ID, pincode, city name, state, and area. The data is sorted by club ID.

Club ID	Pincode	City Name	State	Area
1	427	Chicago	Illinois	01434 Maryland Pass
2	60137	Fort Wayne	Indiana	4080 Melby Terrace
3	489	Tacoma	Washington	7 Vera Street
4	1	Virginia Beach	Virginia	82031 Ohio Alley
5	8694	Las Vegas	Nevada	93 Lyons Junction
6	5982	Boulder	Colorado	7 Farmco Drive
7	51691	Seattle	Washington	193 Mosinee Plaza

7. Custom Query

Home Page Show club

Query

Write Your Query Below

```
select * from club where club_id<7
```

Execute

Home Page Show club Insert Page Run Another Query

Show Query Output

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

Properties SQL Statistics **SQL**

clubdb/postgres@PostgreSQL 14 No limit

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
```

Data output Messages Notifications

club_id	club_name	club_email
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

The screenshot shows the pgAdmin 4 interface. The left pane is a tree view of the database schema, including Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (13). The 'Tables' node is expanded, showing 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, and Tablespaces. The right pane has tabs for Properties, SQL, Statistics, and SQL (selected). A query window titled 'clubdb/postgres@PostgreSQL 14' shows the following SQL code:

```
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
```

The 'Data output' tab displays the results of the last query in a grid:

club_id	club_name	club_email
1	2 Laboratoires dermo Cosmetik Inc	((club.club_email))
2	3 kamal	((club.club_email))
3	4 harshil	haTahar
4	6 Vandan	vandi@hiya

Below the table, it says 'Total rows: 4 of 4 Query complete 00:00:00.057'. The status bar at the bottom right shows 'Ln 7, Col 36'.

8. Custom Query Containing natural join of 2 relation

Home Page Show club

Query

Write Your Query Below

```
select * from club natural join club_address
```

Execute

Home Page Show club Insert Page Run Another Query

Show Query Output

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	jknerjknjkjnjkern	840	Alexandria	Louisiana	3 Hazelcrest Hill

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 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	club_name	club_email	pincode	city_name	stat	area
1	Laboratoires dermo Cosmetik I...	{club.club_email}	60137	Fort Wayne	Indiana	4080 Melby Terra...
2	kamal		489	Tacoma	Washington	7 Vera Street
3	4	harshil	1	Virginia Beach	Virginia	82031 Ohio Alley
4	6	vandi@hiya	5982	Boulder	Colorado	7 Farmco Drive
5	8	updated	4348	Fayetteville	North Carolina	199 Lakewood G...
6	10	zc	840	New York City	New York	702 Declaration ...
7	11	nksejk	840	Alexandria	Louisiana	3 Hazelcrest Hill
8	13	Apaltis Pharma US, Inc.	7	West Hartford	Connecticut	4284 Westport J...
9	14	Aurobindo Pharma I limited				

Total rows: 37 of 37 Query complete 00:00:00.056

Successfully run. Total query runtime: 56 msec. 37 rows affected.

Ln 9, Col 45

22:48 23-11-2022

9. Updation of Club details

Club ID	Club name	Club Email	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	jknerjknjkjnjenjkern	Edit	Delete
102	Adi_Kamal	Adi@da	Edit	Delete

Home Page Show club Insert Page

Edit Club Record

Club ID	<input type="text" value="102"/>
Club name	<input type="text" value="UpdatedName"/>
Club Email	<input type="text" value="upnm@dd"/>
<input type="button" value="Update Record"/>	

JB account | GitHub | Club | studio | 19_Jel | Lab10 | Make | Club | Login | Simpli | 11 - C | ERT | Edit C | +

← → C 127.0.0.1:8000/updateclub/102

Sem5 - Google Drive SEM 5 - Google Drive sem-5 - Google Drive Folder - Google Dr... GitHub - Tikam02... In Section 3.5.3 we... Embedded System... Database System C... Jim Kurose Homep... »

[Home Page](#) [Show club](#) [Insert Page](#)

Edit Club Record

Club ID	<input type="text" value="102"/>
Club name	<input type="text"/>
Club Email	<input type="text"/>
Record updates successfully!!	
Update Record	

Type here to search

23:06 23-11-2022

JB account | GitHub | Club | studio | 19_Jel | Lab10 | Make | Club | Login | Simpli | 11 - C | ERT | Club | +

← → C 127.0.0.1:8000/showclub

Sem5 - Google Drive SEM 5 - Google Drive sem-5 - Google Drive Folder - Google Dr... GitHub - Tikam02... In Section 3.5.3 we... Embedded System... Database System C... Jim Kurose Homep... »

			Edit	Delete
40	HEB	gwillingham13@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	<code>{{club.club_email}}</code>	Edit	Delete
43	Washington Homeopathic Products	ksma@b.com	Edit	Delete
3	kamal	<code>{{club.club_email}}</code>	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	jknerjknjkjnjkern	Edit	Delete
102	UpdatedName	upnm@dd	Edit	Delete

Type here to search

23:06 23-11-2022

pgAdmin 4

File Object Tools Help

Browser

Properties SQL Statistics clubdb/postgres@PostgreSQL 14

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
31	3	kamal	washington Homeopathic PROU...	(club.club_email)	kam@o.com
32	4	harshil		haTahar	
33	8	updated		updated@gmail.com	
34	6	Vandan		vandi@hiya	
35	988	kamal988		kamal988@gmail.com	
36	10	zc		sa	
37	11	nknsejk		jknerjknkjnjern	
38	102	UpdatedName		upnm@dd	
39					

Total rows: 39 of 39 Query complete 00:00:00.056 Ln 2, Col 1

Type here to search

23:07 23-11-2022

10. Deletion of Data in Club details


| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 | 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 | 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 | 321 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 340 | 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 | 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 | 361 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 | 370 | 371 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 | 380 | 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 | 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 | 401 | 402 | 403 | 404 | 405 | 406 | 407 | 408 | 409 | 410 | 411 | 412 | 413 | 414 | 415 | 416 | 417 | 418 | 419 | 420 | 421 | 422 | 423 | 424 | 425 | 426 | 427 | 428 | 429 | 430 | 431 | 432 | 433 | 434 | 435 | 436 | 437 | 438 | 439 | 440 | 441 | 442 | 443 | 444 | 445 | 446 | 447 | 448 | 449 | 450 | 451 | 452 | 453 | 454 | 455 | 456 | 457 | 458 | 459 | 460 | 461 | 462 | 463 | 464 | 465 | 466 | 467 | 468 | 469 | 470 | 471 | 472 | 473 | 474 | 475 | 476 | 477 | 478 | 479 | 480 | 481 | 482 | 483 | 484 | 485 | 486 | 487 | 488 | 489 | 490 | 491 | 492 | 493 | 494 | 495 | 496 | 497 | 498 | 499 | 500 | 501 | 502 | 503 | 504 | 505 | 506 | 507 | 508 | 509 | 510 | 511 | 512 | 513 | 514 | 515 | 516 | 517 | 518 | 519 | 520 | 521 | 522 | 523 | 524 | 525 | 526 | 527 | 528 | 529 | 530 | 531 | 532 | 533 | 534 | 535 | 536 | 537 | 538 | 539 | 540 | 541 | 542 | 543 | 544 | 545 | 546 | 547 | 548 | 549 | 550 | 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 | 560 | 561 | 562 | 563 | 564 | 565 | 566 | 567 | 568 | 569 | 570 | 571 | 572 | 573 | 574 | 575 | 576 | 577 | 578 | 579 | 580 | 581 | 582 | 583 | 584 | 585 | 586 | 587 | 588 | 589 | 590 | 591 | 592 | 593 | 594 | 595 | 596 | 597 | 598 | 599 | 600 | 601 | 602 | 603 | 604 | 605 | 606 | 607 | 608 | 609 | 610 | 611 | 612 | 613 | 614 | 615 | 616 | 617 | 618 | 619 | 620 | 621 | 622 | 623 | 624 | 625 | 626 | 627 | 628 | 629 | 630 | 631 | 632 | 633 | 634 | 635 | 636 | 637 | 638 | 639 | 640 | 641 | 642 | 643 | 644 | 645 | 646 | 647 | 648 | 649 | 650 | 651 | 652 | 653 | 654 | 655 | 656 | 657 | 658 | 659 | 660 | 661 | 662 | 663 | 664 | 665 | 666 | 667 | 668 | 669 | 670 | 671 | 672 | 673 | 674 | 675 | 676 | 677 | 678 | 679 | 680 | 681 | 682 | 683 | 684 | 685 | 686 | 687 | 688 | 689 | 690 | 691 | 692 | 693 | 694 | 695 | 696 | 697 | 698 | 699 | 700 | 701 | 702 | 703 | 704 | 705 | 706 | 707 | 708 | 709 | 710 | 711 | 712 | 713 | 714 | 715 | 716 | 717 | 718 | 719 | 720 | 721 | 722 | 723 | 724 | 725 | 726 | 727 | 728 | 729 | 730 | 731 | 732 | 733 | 734 | 735 | 736 | 737 | 738 | 739 | 740 | 741 | 742 | 743 | 744 | 745 | 746 | 747 | 748 | 749 | 750 | 751 | 752 | 753 | 754 | 755 | 756 | 757 | 758 | 759 | 760 | 761 | 762 | 763 | 764 | 765 | 766 | 767 | 768 | 769 | 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | 778 | 779 | 780 | 781 | 782 | 783 | 784 | 785 | 786 | 787 | 788 | 789 | 790 | 791 | 792 | 793 | 794 | 795 | 796 | 797 | 798 | 799 | 800 | 801 | 802 | 803 | 804 | 805 | 806 | 807 | 808 | 809 | 810 | 811 | 812 | 813 | 814 | 815 | 816 | 817 | 818 | 819 | 820 | 821 | 822 | 823 | 824 | 825 | 826 | 827 | 828 | 829 | 830 | 831 | 832 | 833 | 834 | 835 | 836 | 837 | 838 | 839 | 840 | 841 | 842 | 843 | 844 | 845 | 846 | 847 | 848 | 849 | 850 | 851 | 852 | 853 | 854 | 855 | 856 | 857 | 858 | 859 | 860 | 861 | 862 | 863 | 864 | 865 | 866 | 867 | 868 | 869 | 870 | 871 | 872 | 873 | 874 | 875 | 876 | 877 | 878 | 879 | 880 | 881 | 882 | 883 | 884 | 885 | 886 | 887 | 888 | 889 | 890 | 891 | 892 | 893 | 894 | 895 | 896 | 897 | 898 | 899 | 900 | 901 | 902 | 903 | 904 | 905 | 906 | 907 | 908 | 909 | 910 | 911 | 912 | 913 | 914 | 915 | 916 | 917 | 918 | 919 | 920 | 921 | 922 | 923 | 924 | 925 | 926 | 927 | 928 | 929 | 930 | 931 | 932 | 933 | 934 | 935 | 936 | 937 | 938 | 939 | 940 | 941 | 942 | 943 | 944 | 945 | 946 | 947 | 948 | 949 | 950 | 951 | 952 | 953 | 954 | 955 | 956 | 957 | 958 | 959 | 960 | 961 | 962 | 963 | 964 | 965 | 966 | 967 | 968 | 969 | 970 | 971 | 972 | 973 | 974 | 975 | 976 | 977 | 978 | 979 | 980 | 981 | 982 | 983 | 984 | 985 | 986 | 987 | 988 | 989 | 990 | 991 | 992 | 993 | 994 | 995 | 996 | 997 | 998 | 999 | 1000 | 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 | 1008 | 1009 | 1010 | 1011 | 1012 | 1013 | 1014 | 1015 | 1016 | 1017 | 1018 | 1019 | 1020 | 1021 | 1022 | 1023 | 1024 | 1025 | 1026 | 1027 | 1028 | 1029 | 1030 | 1031 | 1032 | 1033 | 1034 | 1035 | 1036 | 1037 | 1038 | 1039 | 1040 | 1041 | 1042 | 1043 | 1044 | 1045 | 1046 | 1047 | 1048 | 1049 | 1050 | 1051 | 1052 | 1053 | 1054 | 1055 | 1056 | 1057 | 1058 | 1059 | 1060 | 1061 | 1062 | 1063 | 1064 | 1065 | 1066 | 1067 | 1068 | 1069 | 1070 | 1071 | 1072 | 1073 | 1074 | 1075 | 1076 | 1077 | 1078 | 1079 | 1080 | 1081 | 1082 | 1083 | 1084 | 1085 | 1086 | 1087 | 1088 | 1089 | 1090 | 1091 | 1092 | 1093 | 1094 | 1095 | 1096 | 1097 | 1098 | 1099 | 1100 | 1101 | 1102 | 1103 | 1104 | 1105 | 1106 | 1107 | 1108 | 1109 | 1110 | 1111 | 1112 | 1113 | 1114 | 1115 | 1116 | 1117 | 1118 | 1119 | 1120 | 1121 | 1122 | 1123 | 1124 | 1125 | 1126 | 1127 | 1128 | 1129 | 1130 | 1131 | 1132 | 1133 | 1134 | 1135 | 1136 | 1137 | 1138 | 1139 | 1140 | 1141 | 1142 | 1143 | 1144 | 1145 | 1146 | 1147 | 1148 | 1149 | 1150 | 1151 | 1152 | 1153 | 1154 | 1155 | 1156 | 1157 | 1158 | 1159 | 1160 | 1161 | 1162 | 1163 | 1164 | 1165 | 1166 | 1167 | 1168 | 1169 | 1170 | 1171 | 1172 | 1173 | 1174 | 1175 | 1176 | 1177 | 1178 | 1179 | 1180 | 1181 | 1182 | 1183 | 1184 | 1185 | 1186 | 1187 | 1188 | 1189 | 1190 | 1191 | 1192 | 1193 | 1194 | 1195 | 1196 | 1197 | 1198 | 1199 | 1200 | 1201 | 1202 | 1203 | 1204 | 1205 | 1206 | 1207 | 1208 | 1209 | 1210 | 1211 | 1212 | 1213 | 1214 | 1215 | 1216 | 1217 | 1218 | 1219 | 1220 | 1221 | 1222 | 1223 | 1224 | 1225 | 1226 | 1227 | 1228 | 1229 | 1230 | 1231 | 1232 | 1233 | 1234 | 1235 | 1236 | 1237 | 1238 | 1239 | 1240 | 1241 | 1242 | 1243 | 1244 | 1245 | 1246 | 1247 | 1248 | 1249 | 1250 | 1251 | 1252 | 1253 | 1254 | 1255 | 1256 | 1257 | 1258 | 1259 | 1260 | 1261 | 1262 | 1263 | 1264 | 1265 | 1266 | 1267 | 1268 | 1269 | 1270 | 1271 | 1272 | 1273 | 1274 | 1275 | 1276 | 1277 | 1278 | 1279 | 1280 | 1281 | 1282 | 1283 | 1284 | 1285 | 1286 | 1287 | 1288 | 1289 | 1290 | 1291 | 1292 | 1293 | 1294 | 1295 | 1296 | 1297 | 1298 | 1299 | 1300 | 1301 | 1302 | 1303 | 1304 | 1305 | 1306 | 1307 | 1308 | 1309 | 1310 | 1311 | 1312 | 1313 | 1314 | 1315 | 1316 | 1317 | 1318 | 1319 | 1320 | 1321 | 1322 | 1323 | 1324 | 1325 | 1326 | 1327 | 1328 | 1329 | 1330 | 1331 | 1332 | 1333 | 1334 | 1335 | 1336 | 1337 | 1338 | 1339 | 1340 | 1341 | 1342 | 1343 | 1344 | 1345 | 1346 | 1347 | 1348 | 1349 | 1350 | 1351 | 1352 | 1353 | 1354 | 1355 | 1356 | 1357 | 1358 | 1359 | 1360 | 1361 | 1362 | 1363 | 1364 | 1365 | 1366 | 1367 | 1368 | 1369 | 1370 | 1371 | 1372 | 1373 | 1374 | 1375 | 1376 | 1377 | 1378 | 1379 | 1380 | 1381 | 1382 | 1383 | 1384 | 1385 | 1386 | 1387 | 1388 | 1389 | 1390 | 1391 | 1392 | 1393 | 1394 | 1395 | 1396 | 1397 | 1398 | 1399 | 1400 | 1401 | 1402 | 1403 | 1404 | 1405 |<th
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

JB accou x github x Club x studio x 1.9.Jai x Lab10 x Make x Club x Login x Simpl x 11-c x ERT - x Club x +

127.0.0.1:8000/showclub

REMEDYREPACK INC.

39		REMEDYREPACK INC.	ogrittoh12@eipals.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	jknerjknjknjenjkern	Edit	Delete

Type here to search

23:08 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

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
31	43 Washington Homeopathic Prod...	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	jknerjknjknjenjkern

Total rows: 38 of 38 Query complete 00:00:00.106 Rows selected: 1

Ln 2, Col 1

Type here to search

23:09 23-11-2022

Link :

https://youtu.be/DQ_esLHSTzQ