Database Management Systems
Final Submission

Students

1. Meshv Patel 202001006
2. Viral Barodia 202001007

1. Final SRS

Problem Description:

1. Description of the Case Study

In the real world, generally clubs employ websites for displaying all sorts of data which is supposed to be used by users to know more about the club or their own subscriptions. Clubs generally use third-party apps which take up the responsibility of managing the backend of the club workings such as staff etc. A system, which may result in data leak, since financial information can be abused by owners of the third party apps.
Clubs are of different kinds, some are meant only for dancing, drinking etc, while others are for more general purposes, like playing games for the entirety of a weekend, having good food etc. Some clubs are handled remotely, where the users need not be present at a particular location, but may enjoy activities at the comfort of their homes.
No matter what sector of the population the club serves, it will always need a management system, to better its efficiency, to make its working possible, even after handling a lot of, and various kinds of data.

A. Purpose

1. Online users (Address(near me)/ Events/ Menu/ USP)/members

- When any user (maybe a member) will search for clubs, they should be able to see the clubs based on their location/events/menu/any unique selling points like sound system/regular performers/specials in menus. The club information should be easily available to see and the location should be easily reachable.

2. Employees/Staff/Tracking of members, tracking events DJs/Performers
    - The club authorities would want to track its costs, employee details, member details, timeline of events of performers/events on their premises.
    - They would also want to categorize their staff, like cooks/security/maintenance and track salaries likewise.
    - All variable costs like the electricity costs, performance fees (if any), event costs should be present in a detailed manner to keep track of the finances.
    - Stocks of the food and drinks should be traceable.

3. Feedback/Ratings on the internet(filled through customer feedback and ratings)
    - Feedbacks and ratings will help the club get a better visibility of their software/name on the internet.

4. Customer complaints
    - The authorities should be able to track complaints by the customers, in case the details are required further by the police or upcoming staff.

5. Timings and capacity and facilities/Photos $\Rightarrow$ to make better choice among multiple clubs (for the customers)
    - The software should display the timings and capacity, with the photos of the facilities provided to attract the customers.

6. Types of membership on the basis of services offered/ discounts (seasonal)
    - The customers should be able to clearly understand the membership schemes based on types of services offered, and any discounts available at any point of time in the year. Group bookings could be discounted, and this information should be clearly defined and visible.
    - Members should be categorized on the basis of their plans.

7. Events ⇒ DJ/Dancing (Yes/No), Standup Comics/Live Bands
    - The database should properly define timelines of events and their time slots along with frequencies (Ex. every Sunday, etc).
    - This can be implemented using a calendar UI.

8. Franchisees across the city/country, different themes, different databases
    - Any franchisee should have a separate, independent database of its own, with references to every other database so that members can claim benefits in any branch.

9. Account (for later payments for members)
    - Outstanding and paid amounts for each member should be clearly defined.

10. No duplication
    - Ensuring no duplication of data/entries related to members/menus/events/employees etc is important since duplication can lead to serious financial damage to the treasury.

11. Online functionalities
    - All membership procedures and payment systems should be available online in the software itself for easy access. An easy and hassle-free membership process will result in an increased number of members.

B. Intended Audience and Reading Suggestions

1. Specific age group (majorly 18-35 years of age)
    - People of a certain age, people who are planning events/ looking for event spaces/holiday places (who can be non-members too)
2. Members
    - Members should be able to track their outstanding payments, see their history of bills usage of facilities with time and date of usage.

3. Club authorities
    - Should be able to efficiently use and track their finances, both incoming and outgoing.

4. Performers/DJs/Standup Comedians etc.
    - Performers should be able to properly see the event ground/space, crowd capacity, general theme/location for their benefit.

5. Municipal Corporation certificates (preparing food/drinks/crowd management)
    - The Municipality officers would also visit for inspections for cleanliness/food or certifications for the same.

6. Clubs in Mumbai, Clubs in Delhi⇒ Gives a general idea about clubs
7. About a particular club ⇒ Gives an idea about one particular club
8. About SRS ⇒ Gives an idea about the SRS document

9. Club management software ⇒ Tells us about the requirements of any general clubbing software.

C. Product Scope

1. Online users will see the information, who wish to visit the club.

People will be able to see the website/web-app over others with better use of SEO (Search Engine Optimization).
The system should provide all the information relevant for the users to choose a club, like menu, activities, timeline of events, timings of opening and closing etc. Moreover, the management system having a good UI and a user-compatible UX will help in bringing in more customers.

2. Club authorities to track their day-to-day expenses/events/finances.
   The management software will allow the administration of the club to perform mathematical and financial calculations for the costs/taxes/salary calculations.
   The club owners will also be able to maintain stock data of everything available in the club, and track maintenance services which are periodic in nature, and even send alerts to the relevant personnel.
   The administration should very efficiently be able to book their space(s) for any upcoming events.
   They can have a method of getting contact information of potential customers.

3. Ability to handle large amount of data
   - The database system can handle a large number of independent/dependent relations and data of a large number of people, whether they be members/non-members/staff etc.

4. Better UI/UX of the GUI required for more traffic of users.
    - A faster implementation of database system and interactive interface will ensure more and more traffic on the software, hence reducing physical queries (Ex. phone call queries etc)
    - Visualization of data of members will help in optimizing operations. Ex. If the swimming pool is being used less than the tennis court, timings of usage can be altered to save on maintenance costs.

D. Description

2. Multiple branches/franchisees ⇒ Multiple sets of databases required
    - Independent databases, while having references to each other and the databases being consistent, so that a customer can avail benefits in all branches.

3. Which relations are required? ⇒ Staff details/Complaints/Member details
    - Relations such as Staff details, electricity bills and usage, member details, events, food menu, account details etc.

4. DBMS and not File-type
    - A file system for such a huge database cannot be file-type, since it will result in data inconsistency, security issues, duplication etc.

5. Relational and not object oriented, since the size of the database is huge.
    - The database would be relational, since object-oriented wouldn't be able to hold such a large amount of data of all the users/members/staff etc.

6. Less subscription fees
    - Since we are only a developer and are providing a service to numerous clubs, we should charge a minimal, competitive rate so as to attract more and more business.

2. Requirements

1. Background readings:

i. WildApricot Clubbing Software

There already exist many softwares or software building sites for managing clubs. One of them is wildapricot.com, in which any club owner can appeal for services to manage the club.

This software has clearly defined tabs for each funcitonality, like member database, payments, mobile app etc.

A. For employing their services, one can first look at example sites they've already made and explore different pricing schemes for their service. Example sites can build confidence in a new client toward the service provider.

B. On top of this, "Features" tab gives information about all the features that a client can have in their software.

C. As we have already discussed, many different kinds of relations, which may or may not be independent, need to be constructed, and for that, the client can go to different tabs to explore different relations, like membership, payments, mobile app builder etc.

D. This software does not provide social media management, real-time analytics, etc. (Source)

E. The client can always build their own customisable website easily using features and tech provided by WildApricot (Source), or integrate a few new features if the client already has a working website of their own.

Ii. SENET E-Sports Club manager

Clubs can also be totally virtual, and one such example is that of an E-sports club. Here, gamers will be using the software for participating in the games, using a payments interface, shopping functionality, etc.

Functionalities:

A. For Owners:
- Cash Registers: Easy for owners to track the incoming finances integrated with the game playing.
- Financial Control: This helps with tracking the entire expenses and revenue scenario. This also includes making financial reports by monitoring the cash flow.
- Detailed statistics: Provides insights into the fact that from where is revenue being generated, popularity of certain games, etc.

B. For Gamers:
- Loyalty Program: This provides functionality of membership, discounts, reward systems etc.
- Reservations and Online payments: Gamers can book slots for playing, and make safe and secure payments online.
- Public Keys: One can use club accounts if one does not have an official Steam or Origins account. This helps in including all players.
- Shop Sell Module: People can order food, drinks and merchandise from the comfort of their homes.

References

Club management systems
WildApricot Features and reviews
Club management systems built by a particular service provider

Combined Requirements

Looking at the expanse of the word ‘club’, it is very clear that requirements of any club will heavily depend on what kind of public the club caters to, and what is the objective of the club. In spite of this, there are a few functionalities which every clubbing software should offer.

1. Offering pricing and subscription plans: Something that we got to learn is that according to different functionalities, we can provide different rates to clients to manage their software and database. These rates should be competitive and should come in the form of subscription plans, which makes it easy to maintain the softwares. These subscription plans could be monthly, quarterly, yearly etc.

2. Sorting functionality: Something that can be done is that a sorting functionality can be added to sort records to better understand the data.

2. Interviews

Interview Plan

1. With Club Staff

System: Presidency Club

Project Reference: SF/SJ/2022/12

Interviewee: 1) Mahepal Singh(Actual) **Designation**: Owner, Presidency Club

Interviewer: 1) Meshv Patel **Designation**: Student

2) Viral Barodia **Designation**: Student

Date: 25/9/2022 **Time:** 14:30

Duration: 25 minutes **Place:** Call

- **Purpose of Interview:**
  An interview to understand backend requirements of the club

- **Agenda**

  ➔ Problems with everyday cost calculation, stock maintenance

  ➔ Problems with salary calculation of staff

  ➔ Issues with data keeping of the staff, like waiters, cooks, bartenders, Bouncers *etc*

  ➔ Data keeping of members accounts, information, which may include

personal information like phone numbers etc.

- **Documents to be brought**:

Records of all employees
Register used for salary calculation, stock maintenance etc.

Interview Summary

System: Presidency Club

Interviewee: Mahepal Singh **Designation**: Manager, Operations

Contact Details: +91 8866067575
Organization Details: Presidency Club Employee

Interviewer:

1. Meshv Patel **Designation**: Student
2. Viral Barodia **Designation**: Student

Date: 25/9/2022 **Time**: 1430 Hrs

Purpose of Interview:

A. No problem with the big costs, but smaller bills (miscellaneous) are harder to track, since billing is not always done, many times paperwork is not available for all costs.
B. Stocks are to be replenished every few days, expiry date is to be taken care of.
C. Cannot give out rotten goods, refrigeration etc are abstract costs, and amount of stocks are variable, so amount sold is to be tracked.

D. Staff leaves are hard to keep track of, a calendar UI can help.
E. Membership details: Lot of data, in terms of which member utilized which facility for how much time, benefits used, to be used etc.

2. With User

Interview Plan

Project Reference: SF/SJ/2022/12

Interviewee: 1) Deep Patel(**Role Play**) **Designation**: Student, IITG

Interviewer: 1) Meshv Patel **Designation**: Student

2) Viral Barodia **Designation**: Student

Date: 25/9/2022 **Time:** 5:00 PM

Duration: 30 minutes **Place:** Phone call

- **Purpose of Interview:**
  An interview to understand requirements of functionalities in demand from the user's end.

- **Agenda**

  ➔ Membership details and plans.

➔ Timelines/Calendar maintenance.

➔ Logistics, features, facilities, parking, etc

Interview Summary

Interviewee: Deep Patel (Role Play) **Designation**: Student

Contact Details: +91 76895482213

Interviewer:

3. Meshv Patel **Designation**: Student
4. Viral Barodia **Designation**: Student

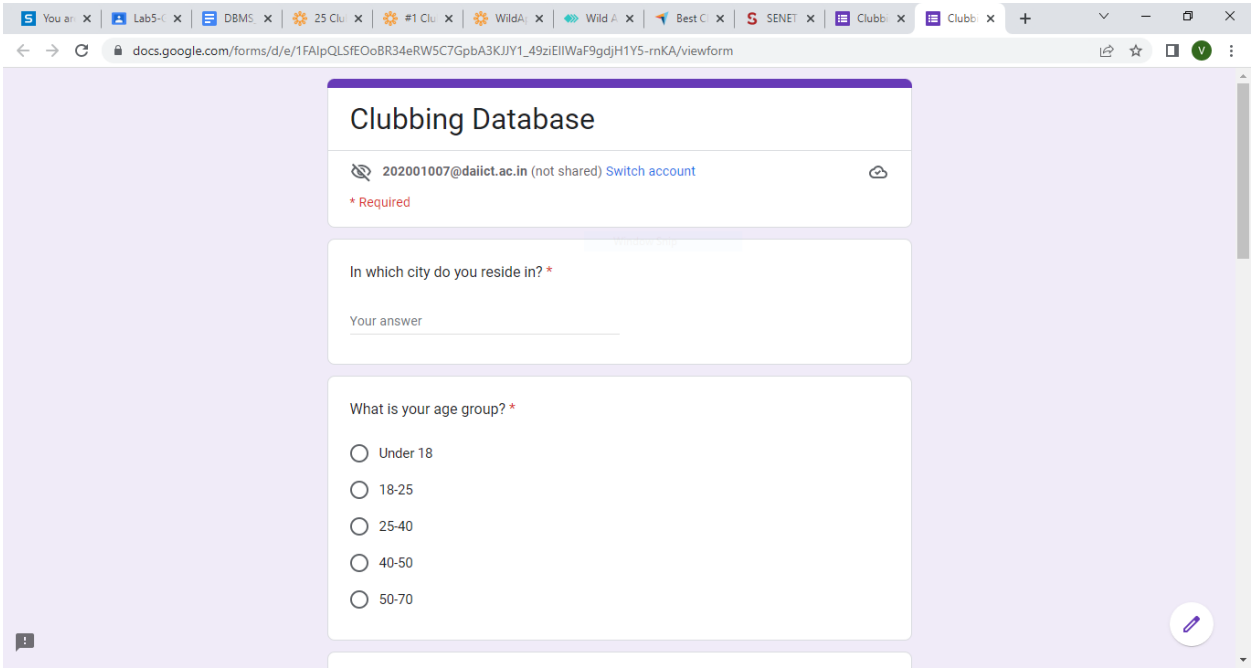
Date: 26/9/2022 **Time**: 1830 Hrs

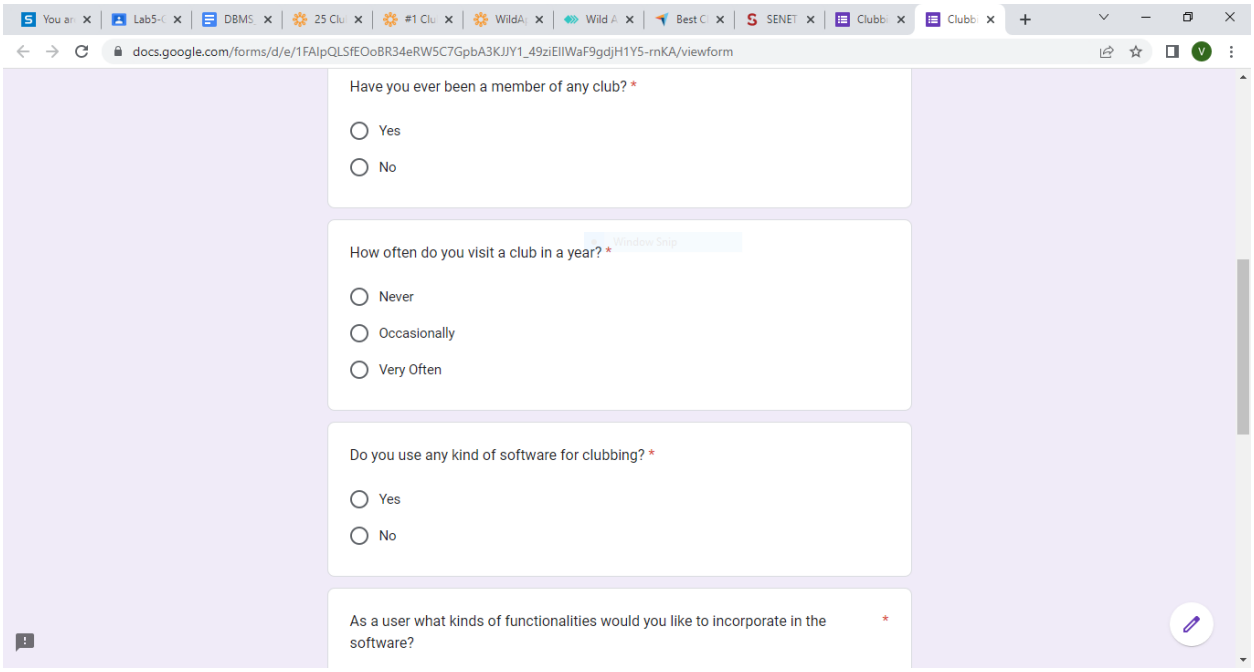
Purpose of Interview:

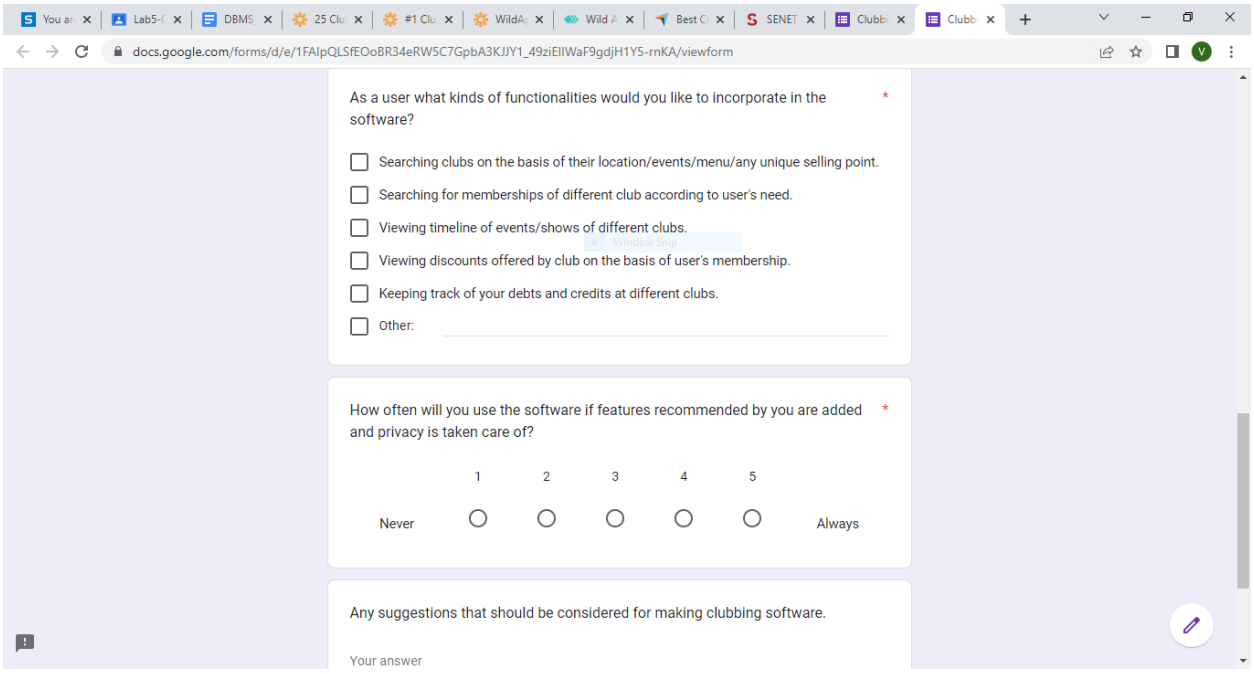
A. Very clearly, membership benefits should be clearly defined to avoid any misunderstanding.
B. Timelines should be clear and updated. Maybe consult all the members for some new activities and their dates and times.
C. Parking facilities, dance rooms/floors/ Dj, sports facilities, games, stag entry, etc all should be displayed with photos, so as to make an informed decision on selecting the club.
D. The club should be well-maintained, equipment and inventory should be usable at all times.
E. The club should adhere to standards of food, hygiene and cleanliness.

3. **Questionnaire/s**

A. Screenshots:



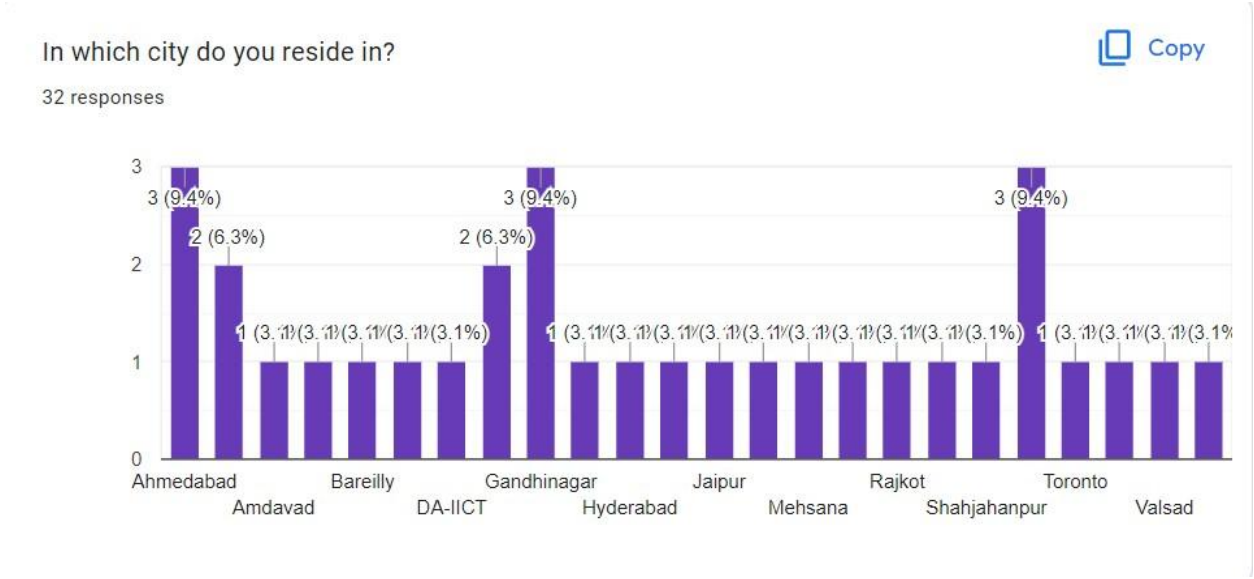

docs.google.com/forms/d/e/1FAIpQLSfEOoBR34eRW5C7GpbA3KJJY1_49ziEIIWaF9gdjH1Y5-rnKA/viewform
Have you ever been a member of any club? *
Yes
No
How often do you visit a club in a year? *
Never
Occasionally
Very Often
Do you use any kind of software for clubbing? *
Yes
No
As a user what kinds of functionalities would you like to incorporate in the software? *


docs.google.com/forms/d/e/1FAIpQLSfEOoBR34eRW5C7GpbA3KJJY1_49ziEIIWaF9gdjH1Y5-rnKA/viewform
As a user what kinds of functionalities would you like to incorporate in the software? *
Searching clubs on the basis of their location/events/menu/any unique selling point.
Searching for memberships of different club according to user's need.
Viewing timeline of events/shows of different clubs.
Viewing discounts offered by club on the basis of user's membership.
Keeping track of your debts and credits at different clubs.
Other:
How often will you use the software if features recommended by you are added and privacy is taken care of? *
1 2 3 4 5
Never
Always
Any suggestions that should be considered for making clubbing software.
Your answer

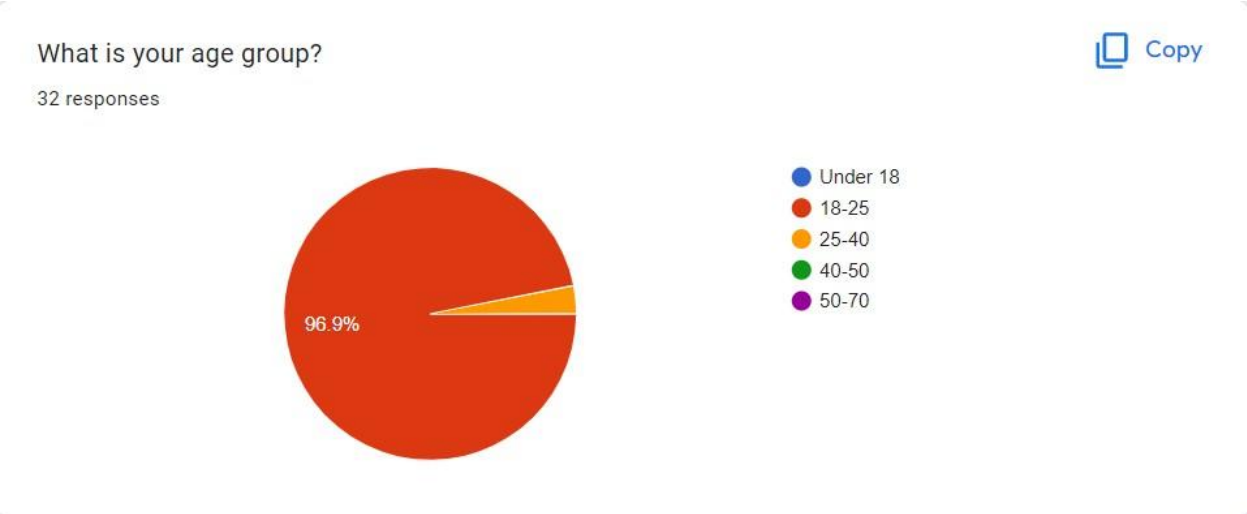
4. Summary

A. Most of the respondents are not a part of any club, and hence do not use any software for clubbing.
B. 95% of the respondents are between 18-25 years of age, letting us believe that it is this age group who are mostly interested in club activities
C. Most of the people would search clubs on the basis of their location and events.
D. Very few people would look at discounts while filling up membership forms, probably because this form was filed by individuals, but discounts are generally given to entire groups of people.
E. Clubs will mainly be used for enjoying events, since most of the people would specifically look at the event calendar before choosing one.

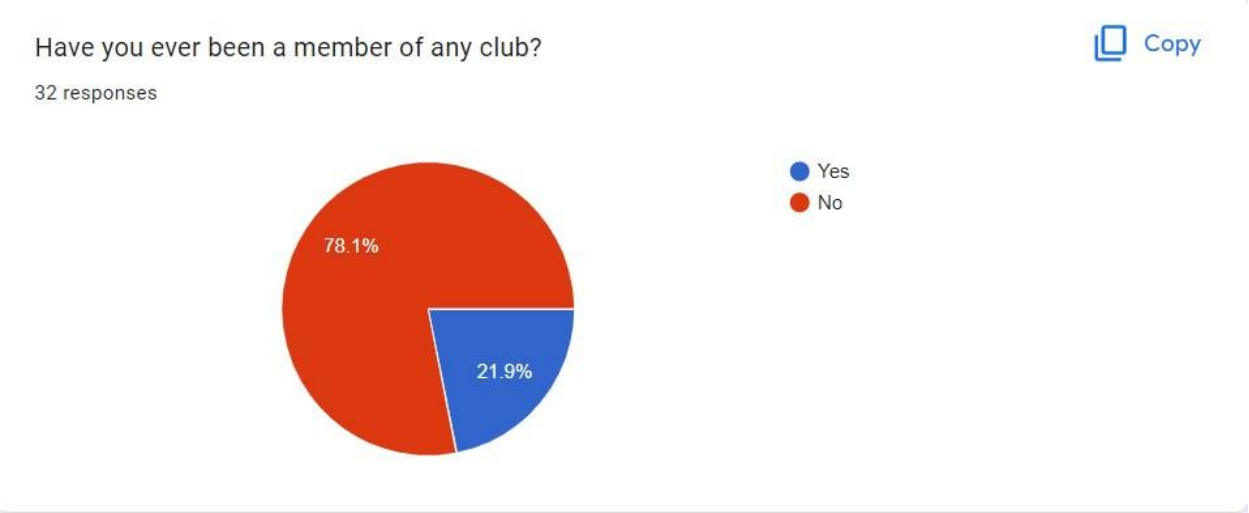
5. **Graphs and Short answers**:



Most of our respondents are from Gujarat



Most of our respondents are in the age group 18-25.

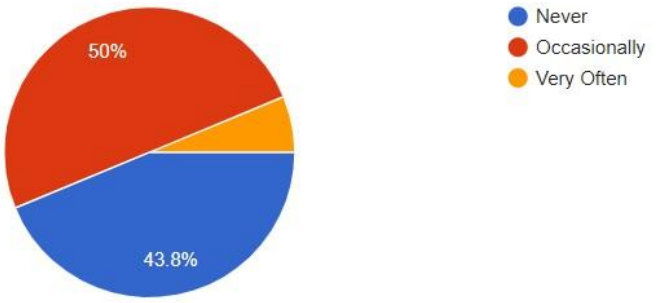


Most of our respondents haven't been part of any club.

How often do you visit a club in a year?

Copy

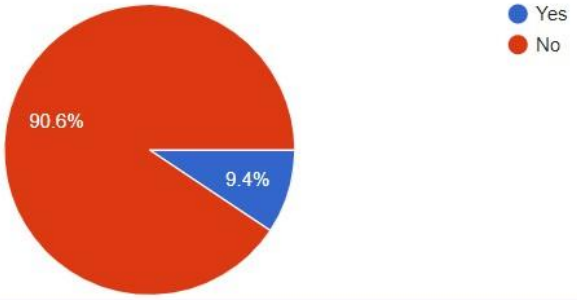
32 responses



Do you use any kind of software for clubbing?

Copy

32 responses

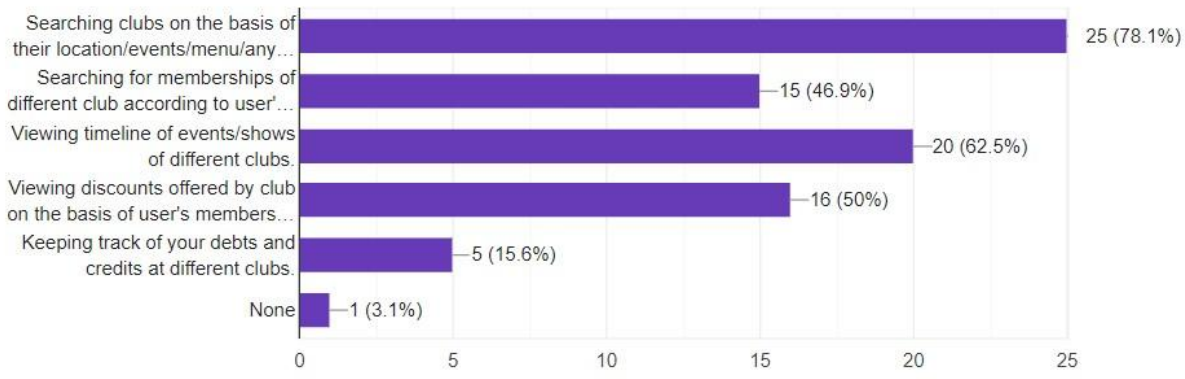


A very small percentage of our respondents are going to a club/using a clubbing software.

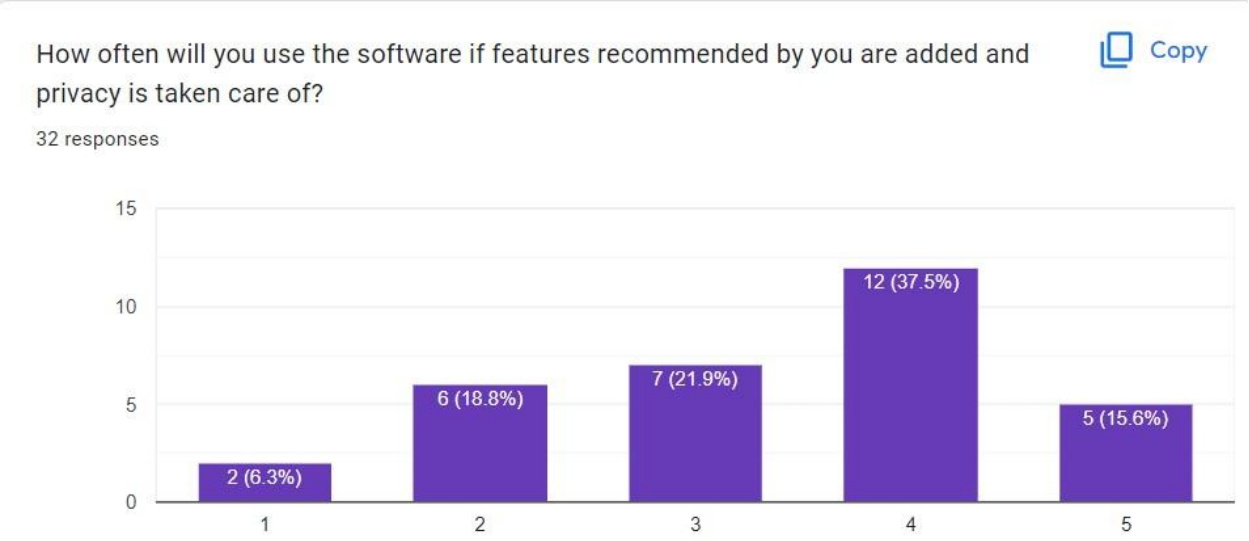
As a user what kinds of functionalities would you like to incorporate in the software?

Copy

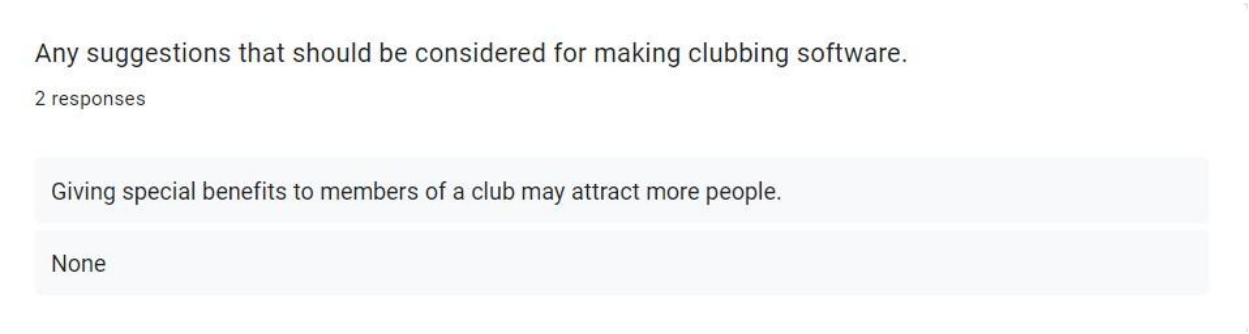
32 responses



Our users generally choose a club on the basis of their location and event timelines.



Most of our users share data security concerns and would want their own features included in the software, as is clearly visible from the above bar graph.



Distinguishing between members and non-members in terms of services provided will result in more people buying a membership card.

Observations

Observations by:- Viral Barodia

Date: 02/10/2022 **Time**: 10:30 AM
Duration: 60mins **Place**: Presidency Club

1. No one to guide for the parking, need more staff on ground, for both day and night shifts.
2. Lack of coordination among receptionists, need to optimize paperwork on the desk.
3. Ask for a lot of personal information like email-id, Aadhar number etc.
4. Lack of cleanliness in the washrooms. General ground/fountain not maintained well.
5. Unmaintained sports equipment.

A. Combined Requirements: The form responses help us include the following information in the functionalities that can be given to the users of the software system:

A. Data Security: The data will surely be given to the service provider (us). This prompts us to build trust with the consumers about their data not being abused. This can be done using encrypted data being stored in the database.
B. Definition of Members: We should clearly define and state the benefits of giving out memberships.
C. Proper staff: The staff should be adequate and well-trained in their job. They are expected to have good communication skills as well.
D. Quality: The route from where wasted food is taken out needs to be clean, as should be the kitchen. Sports equipment should be well maintained as well.

3. **Fact Finding Chart**

Objective	Technique	Subject	Time Commitment
Background on clubs, their events, features etc	Background reading	Articles on the internet	3-4 hrs
Administration and working of the club	Physical visit, interview	One operations manager, receptionist, club staff etc	1-2 hrs
User needs	Interview, readings	Role playing, internet	0.5-1 hrs
Why maintain a database	Interview, physical visit	Receptionist, staff, etc.	1-2 hrs

4. **List of Requirements**

A. A separate provision for maintenance and cleaning has to be there in the database, which can be updated from time to time.
B. Staff leave timeline has to be properly maintained.

5. **User Categories and Privileges**

A. Customer: The customer, who may or may not be a member of the club, uses the database to either avail the facilities provided by the club or check on them to make an informed decision. They mostly have a viewer access, with only ratings, feedback and a few other things such as bookings being their contribution to the database.
B. Administrators: They will have editor access to almost the entire database, and will be able to update, delete data which is visible to the

customers/general public. They shall use the software for keeping track of the club functioning.

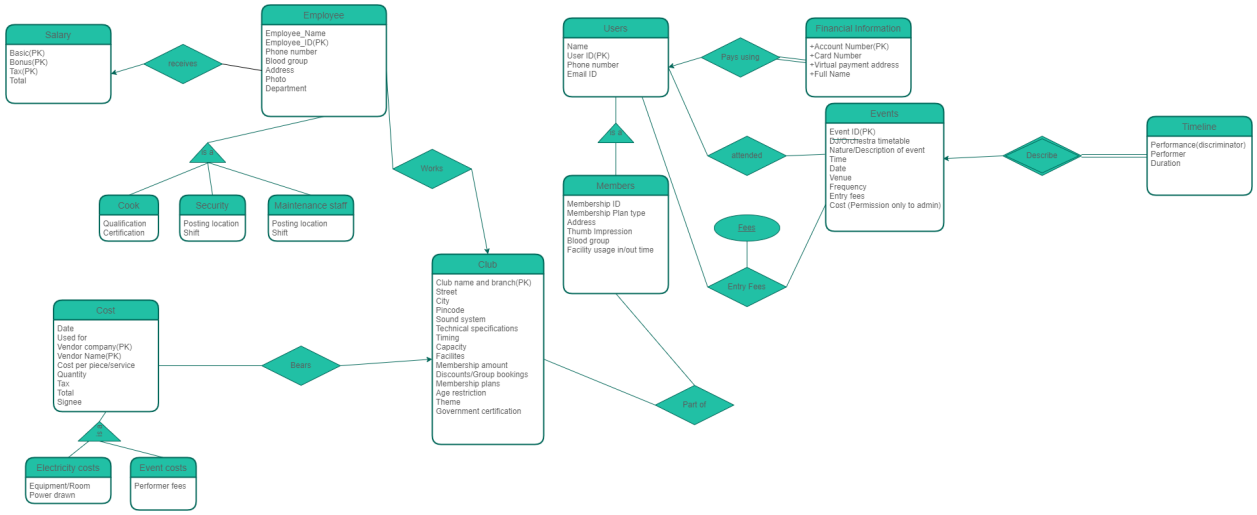
6. **Assumptions**

A. The software developer will not indulge in data manipulation/stealing.
B. The administrator will enter correct details about the financial transactions of the customers as well as their own costs and not cheat.
C. The administration will enter correct details of the certifications received (if any), from the government bodies, and updated data of inventory maintenance, salaries of employees, membership plans, members and their data etc.

7. **Business Constraints**

7. Such a business won't cater to people below a certain financial level.
8. Will only attract consumers who are in search of leisure, since the services provided are mostly not a necessity.
9. The taxes on services will hence be higher than most products, resulting in lower margins initially.

3. Final ER Diagram and Noun Analysis



Noun and Verb Analysis

Sr. No.	Noun	Verb
1	Staff	
2	Staff_Id	
3	Name	
4	Phone number	
5	Blood group	
6	Address	
7	Financial information (users)	Hide
8	Account number	
9	Card Number	
10	Virtual Payment Address	

	(VPA)	
11	Club Information	
11	Full Name	
12	Location	reachable
13	City	
14	Street	
15	Pincode	
16	Management system	manages
17	Users	Use
18	User ID	
18	Name	
	Email ID	
19	Phone number	
20	Menu	Attract
21	Item name	
22	Item price	
23	Members	Use (facilities)
25	Membership ID	
26	Membership plan type	
27	Address	
29	Thumb impression	
30	Blood group	
31	Events	

32	Time	
33	Date	
34	Nature/Description of event	
35	Specials (in menu)	Attracts
36	Frequency	
37	Name	
38	Entry Fees	
39	Sound system	
40	Technical specifications	
41	DJ/Orchestra	
42	Name	
43	Cost (only to be viewed by admin)	
44	Costs	
45	Vendor company	
46	Vendor name	
47	Function performed	
48	Breakdown of costs	
49	Cost per service	
50	Quantity	
51	Total	
52	Employee details	
53	Employee_name	
54	Employee_ID	

55	Contact number	
56	Employee_department	
57	Salary	
58	Timeline	manages
59	performance	
60	Performer	
61	Duration	
62	Cooks	
63	Qualification	
64	Certification	
65	Security	
66	Posting location	
67	Maintenance staff	
68	Posting location	
69	Salaries	
70	Electricity cost	
71	Room/Equipment	
73	Power drawn	
74	Performance fees	
75	Event costs	
76	Stocks (food items)	maintained
77	Feedbacks	given
78	Ratings	given

79	Complaints	given
80	Timings	
81	Capacity	
82	Facilities	provided
83	Membership amount	
84	Discounts	
85	Group bookings	
86	Membership plan	
87	Age restrictions	
88	Event frequency	
89	Franchisee(s)	
90	Theme	
91	Personal member account	
92	Calendar	Implemented
93	Treasury	
94	Payments	
95	Bills	
96	Facility usage details	
97	Government certification	
98	Contact information of potential members	maintain
99	Bookings	
	Manager	manages
	Salary	

	Basic salary	
	Bonus	
	Tax	
	Total salary	

10. Final Normalization and DDL scripts and Queries

- Employee_ID
- Club Name and branch (FK referencing to entity)
- Basic(FK referencing to Salary)
- Bonus(FK referencing to Salary)
- Tax(FK referencing to Salary)
- Employee_Name
- Blood Group
- Department
- Photo

-> PK dependencies are: Employee_ID -> Employee_works_receives
-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only the decider is an insert candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

A.1. Cooks

- Employee_ID(FK referencing to Employee_works)
- Qualification
- Certification

-> PK dependencies are: Employee_ID -> Cooks
-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

A.2. Security

- Employee_ID(FK referencing to Employee_works)
- Posting location
- Shift

-> PK dependencies are: Employee_ID -> Security
-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

A.3. Maintenance staff

- Employee_ID(FK referencing to Employee_works)
- Posting location
- Shift

→ PK dependencies are: Employee_ID -> Maintenance staff
→ No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
→ No update, insert and delete anomalies exist as the relation is already in BCNF.

B. Costs_bears

- Date
- Used for
- Vendor company
- Vendor Name
- Cost per unit
- Quantity
- Tax
- Total
- Signee
- Club name and branch(FK referencing to Club).

-> PK dependencies are: (Vendor Company, vendor name) -> Costs_bears
-> Dependencies: (Cost Per Unit,Quantity,Tax) -> Total
-> Here there aren't any partial dependencies, so the relation is already in 2NF.
→ There exists one transitive dependency((Cost Per Unit,Quantity,Tax) -> Total) which prevents this table from being in 3NF. To convert this table to

3NF form, we can remove the Total attribute(because it can be easily derived from the other three attributes as Total is a derived attribute).
→ Update anomaly: If there is a change in the cost per unit or the quantity of equipment used, changes might not be reflected in all the tuples.

Updated Cost_bears is:

- Date
- Used for
- Vendor company
- Vendor Name
- Cost per unit
- Quantity
- Tax
- Signee
- Club name and branch(FK referencing to Club).

B.1. Electricity costs

- Equipment
- Room
- Power drawn
- Vendor company(FK referencing to Costs_bears)
- Vendor name(FK referencing to Costs_bears)

-> PK dependencies are: (Vendor Company, vendor name) -> Electricity costs
-> No functional dependencies(i.e partial and transitive) other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

B.2. Event costs

- Performer fees
- Vendor company(FK referencing to Costs_bears)
- Vendor name(FK referencing to Costs_bears)

-> PK dependencies are: (Vendor Company, vendor name) -> Event costs
-> No functional dependencies(i.e partial and transitive) other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

C. Club

- Club name and branch(PK)
- Street
- City
- Pincode
- Sound system
- Timing
- Capacity
- Facilities
- Discounts/Group bookings
- Age restriction
- Theme
- Government certification

-> PK dependency is: Club name and branch(PK) -> Club
-> Other functional dependencies: Pincode -> City
-> There aren't any partial dependencies, so the relation is already in 2NF.
-> Update anomalies: If the city name associated with a pin code changes then we need to change the city in every tuple.
→ Delete anomalies: If there is only one club in a city, then deletion of that tuple will lead to loss of information of pincode,city tuple.
→ Insert anomalies: If we want to store the city name for a given pincode, then it can be stored unless there is a club present in that city.
→ Pincode -> city is a transitive dependency which prevents this to be in 3NF. So this table can be further decomposed into

C.1: Club_1

- Club name and branch(PK)
- Street
- Pincode
- Sound system
- Timing
- Capacity
- Facilities
- Discounts/Group bookings
- Age restriction
- Theme
- Government certification

C.2: Club_2

- Pincode
- city

-> Above two relations are now in BCNF(as only determinants are candidate key)

D. Users

- Name
- User ID
- Phone number
- Email ID

→ PK Dependency is: (User_ID → Users)
→ Other functional dependencies are: ‘None’
→ No functional dependencies(i.e partial and transitive) other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
→ No update, insert and delete anomalies exist as the relation is already in BCNF.

D.1 Members

- User_ID(FK referencing to users)
- Membership ID
- Membership Plan type
- Address
- Thumb Impression
- Blood group

→ PK dependencies: (User_ID → Members)
→ Other functional dependencies: Membership ID → Members, Thumb Impression → Members
→ All the determinants, i.e User_ID, Thumb impression and Membership_ID are CKs, and hence the relation is in BCNF.

E. FinancialInformation_PaysUsing

- Account number
- Card number
- Virtual Payments Address
- Full name
- User_ID(Foreign key referencing to Users)(NOT null constraint).

→ PK Dependencies: Account number,Card number, VPA, Full name → Financialinformation_PaysUsing
→ No other functional dependency exists

F. Events

- Event ID(PK)
- DJ/Orchestra timetable
- Nature/Description of event
- Time
- Date
- Venue
- Frequency
- Entry fees
- Cost (Permission only to admin)

-> PK dependencies are: Event_ID -> Events
-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only decider are candidate key).

-> No update, insert and delete anomalies exist as the relation is already in BCNF.

G. Salary

- Basic(PK)
- Bonus(PK)
- Tax(PK)
- Total

-> PK dependencies are: (Basic,Bonus,Tax) -> Salary.
-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only decider are candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

H. Timeline_Describe

- Event_ID (FK referencing to Events)(Not NULL constraint)(On delete cascade)
- Performance
- Performer
- Duration

-> PK dependencies are: (Event_ID,Performance) -> Timeline_Describe.
-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only decider are candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

I. Attended

- User_ID(FK referencing to User)
- Event_ID (FK referencing to Event)

-> PK dependencies are: (Event_ID,Performance) -> Timeline_Describe.
-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only decider are candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

J. Part of

- Membership ID (FK referencing to Members)
- Club name and branch (FK referencing to Club)

-> PK dependencies are: (Membership_id,Club name and branch) -> Part of.

-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

K. EntryFees

- User_ID (FK referencing to User)
- Event_ID (FK referencing to Event)
- Fees

-> PK dependencies are: (User_id,Event_ID) -> Entry fees.
-> No functional dependencies other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
-> No update, insert and delete anomalies exist as the relation is already in BCNF.

L. PhoneNumber_Employee

- Employee ID(FK referencing to Employee_works)
- PhoneNumber

→ PK Dependencies: Employee_ID, PhoneNumber → PhoneNumber_Employee
→ No functional dependencies(i.e partial and transitive) other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
→ No update, insert and delete anomalies exist as the relation is already in BCNF.

M. Membership Amount and Plan and Discounts/Group bookings:

- Club name and branch (FK referencing to club relation)
- Membership plan
- Membership amount
- Discounts and group bookings

→ PK dependencies: Club name and branch, membership plan → Membership Amount and Plan and Discounts/Group bookings
→ No functional dependencies(i.e partial and transitive) other than PK dependency exist. So, the relation is already in BCNF(as only the decider is a candidate key).
→ No update, insert and delete anomalies exist as the relation is already in BCNF.

DDL script:
Users:

```
CREATE TABLE IF NOT EXISTS clubbing_db."Users"
(
    "Name" character varying,
    "User_ID" character varying NOT NULL,
    "Phone number" character varying,
    "Email ID" character varying,
    CONSTRAINT "Users_pkey" PRIMARY KEY ("User_ID")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing_db."Users"
    OWNER to postgres;
```

Members:

```
CREATE TABLE IF NOT EXISTS clubbing_db."Members"
(
```

```
    "User_ID" character varying NOT NULL,
    "Membership ID" character varying NOT NULL,
    "Membership plan type" character varying NOT NULL,
    "Address" character varying,
    "Thumb Impression" character varying,
    "Blood group" character varying,
    CONSTRAINT "Members_pkey" PRIMARY KEY ("User_ID"),
    CONSTRAINT u1 UNIQUE ("Membership ID"),
    CONSTRAINT u2 UNIQUE ("Thumb Impression"),
    CONSTRAINT "FK1" FOREIGN KEY ("User_ID")
        REFERENCES clubbing_db."Users" ("User_ID") MATCH SIMPLE
        ON UPDATE cascade
        ON DELETE cascade
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing_db."Members"
    OWNER to postgres;
```

DDL script:

Users:

```
CREATE TABLE clubbing_db."Users"
(
    "Name" character varying,
    "User_ID" character varying NOT NULL,
    "Phone number" character varying,
    "Email ID" character varying,
    PRIMARY KEY ("User_ID")
);

ALTER TABLE IF EXISTS clubbing_db."Users"
    OWNER to postgres;
```

Members:

```
CREATE TABLE IF NOT EXISTS clubbing_db."Members"
(
    "User_ID" character varying NOT NULL,
    "Membership ID" character varying NOT NULL,
    "Membership plan type" character varying NOT NULL,
    "Address" character varying,
    "Thumb Impression" character varying,
    "Blood group" character varying,
    CONSTRAINT "Members_pkey" PRIMARY KEY ("User_ID"),
    CONSTRAINT u1 UNIQUE ("Membership ID"),
    CONSTRAINT u2 UNIQUE ("Thumb Impression"),
    CONSTRAINT "FK1" FOREIGN KEY ("User_ID")
        REFERENCES clubbing_db."Users" ("User_ID") MATCH SIMPLE
        ON UPDATE cascade
        ON DELETE cascade
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing_db."Members"
    OWNER to postgres;
```

FinancialInformation_PaysUsing:

```
CREATE TABLE clubbing_db."FinancialInformation_PaysUsing"
(
    "Account number" character varying NOT NULL,
    "Card number" character varying NOT NULL,
    "Virtual Payment Address" character varying NOT NULL,
    "Full Name" character varying NOT NULL,
    "User_ID" character varying NOT NULL,
    PRIMARY KEY ("Account number", "Card number", "Virtual Payment
Address", "Full Name"),
    CONSTRAINT "FK1" FOREIGN KEY ("User_ID")
        REFERENCES clubbing_db."Users" ("User_ID") MATCH SIMPLE
        ON UPDATE cascade
        ON DELETE cascade
        NOT VALID
);
```

```
ALTER TABLE IF EXISTS clubbing_db."FinancialInformation_PaysUsing"
    OWNER to postgres;
```

EVENTs:

```
CREATE TABLE clubbing_db."Events"
(
    "Event_ID" character varying NOT NULL,
    "DJ/orchestra timetable" character varying,
    "Nature of event" character varying,
    "Time" character varying,
    "Date" character varying,
    "Venue" character varying,
    "Frequency" character varying,
    "Entry Fees" character varying,
    "Cost" character varying,
    PRIMARY KEY ("Event_ID")
);

ALTER TABLE IF EXISTS clubbing_db."Events"
    OWNER to postgres;
```

Salary:

```
CREATE TABLE clubbing_db."Salary"
(
    "Basic" character varying NOT NULL,
    "Bonus" character varying NOT NULL,
    "Tax" character varying NOT NULL,
    "Total" character varying,
    PRIMARY KEY ("Basic", "Bonus", "Tax")
);

ALTER TABLE IF EXISTS clubbing_db."Salary"
    OWNER to postgres;
```

Timeline_Describe:

```
CREATE TABLE clubbing_db."Timeline_Describe"
(
    "Event_ID" character varying NOT NULL,
    "Performance" character varying NOT NULL,
    "Performer" character varying,
    "Duration" character varying,
```

```
    PRIMARY KEY ("Event_ID", "Performance"),
    CONSTRAINT "FK1" FOREIGN KEY ("Event_ID")
        REFERENCES clubbing_db."Events" ("Event_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."Timeline_Describe"
    OWNER to postgres;
```

Attended:

```
CREATE TABLE clubbing_db."Attended"
(
    "User_ID" character varying NOT NULL,
    "Event_ID" character varying NOT NULL,
    PRIMARY KEY ("User_ID", "Event_ID"),
    CONSTRAINT "FK1" FOREIGN KEY ("User_ID")
        REFERENCES clubbing_db."Users" ("User_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT "FK2" FOREIGN KEY ("Event_ID")
        REFERENCES clubbing_db."Events" ("Event_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."Attended"
    OWNER to postgres;
```

CLUB:

```
CREATE TABLE IF NOT EXISTS clubbing_db."Club"
(
    "Club name and branch" character varying NOT NULL,
    street character varying,
    "Pincode" character varying,
    "Sound System" character varying,
    "Timing" character varying,
```

```
    "Capacity" character varying,
    "Facilities" character varying,
    "Discount" character varying,
    "Age restriction" character varying,
    "Theme" character varying,
    "Governement Certification" character varying,
    CONSTRAINT "Club_pkey" PRIMARY KEY ("Club name and branch")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing_db."Club"
    OWNER to postgres;
```

CLUB_2:

```
CREATE TABLE IF NOT EXISTS clubbing_db."Club_2"
(
    "Pincode" bigint NOT NULL,
    "City" character varying,
    CONSTRAINT "Club_2_pkey" PRIMARY KEY ("Pincode")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing_db."Club_2"
    OWNER to postgres;
```

Employee_works_recieves:

```
CREATE TABLE clubbing_db."Employee_works_receives"
(
    "Employee_ID" character varying NOT NULL,
    "Club Name and brach" character varying,
    "Basic" bigint,
    "Bonus" bigint,
    "Tax" bigint,
    "Employee Name" character varying,
    "Blood Group" character varying,
    "Photo" character varying,
    PRIMARY KEY ("Employee_ID"),
    CONSTRAINT "FK1" FOREIGN KEY ("Club Name and brach")
```

```
        REFERENCES clubbing_db."Club" ("Club name and branch") MATCH
SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT "FK2" FOREIGN KEY ("Basic", "Bonus", "Tax")
        REFERENCES clubbing_db."Salary" ("Basic", "Bonus", "Tax") MATCH
SIMPLE
        ON UPDATE cascade
        ON DELETE cascade
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."Employee_works_receives"
    OWNER to postgres;
```

COOKS:

```
CREATE TABLE clubbing_db."Cooks"
(
    "Employee_ID" character varying NOT NULL,
    "Qualification" character varying,
    "Certification" character varying,
    PRIMARY KEY ("Employee_ID"),
    CONSTRAINT "FK1" FOREIGN KEY ("Employee_ID")
        REFERENCES clubbing_db."Employee_works_receives" ("Employee_ID")
MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."Cooks"
    OWNER to postgres;
```

SEcurity:

```
CREATE TABLE clubbing_db."Security"
(
    "Employee_ID" character varying NOT NULL,
    "Posting location" character varying,
    "Shift" character varying,
```

```
    PRIMARY KEY ("Employee_ID"),
    CONSTRAINT "FK1" FOREIGN KEY ("Employee_ID")
        REFERENCES clubbing_db."Employee_works_receives" ("Employee_ID")
MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."Security"
    OWNER to postgres;
```

Maintenance STAff:

```
CREATE TABLE clubbing_db."Maintaince staff"
(
    "Employee_ID" character varying NOT NULL,
    "Posting location" character varying,
    "Shift" character varying,
    PRIMARY KEY ("Employee_ID"),
    CONSTRAINT "FK1" FOREIGN KEY ("Employee_ID")
        REFERENCES clubbing_db."Employee_works_receives" ("Employee_ID")
MATCH SIMPLE
        ON UPDATE cascade
        ON DELETE cascade
        NOT VALID
);
ALTER TABLE IF EXISTS clubbing_db."Maintaince staff"
    OWNER to postgres;
```

Costs_bears:

```
CREATE TABLE IF NOT EXISTS clubbing_db."Costs_bears"
(
    "Vendor Company" character varying NOT NULL,
    "Vendor name" character varying NOT NULL,
    "Date" character varying,
    "Used for" character varying,
    "Cost per unit" character varying ,
    "Quantity" character varying ,
    "Tax" character varying ,
    "Signee" character varying,
```

```
    "Club name and branch" character varying,
    CONSTRAINT "Costs_bears_pkey" PRIMARY KEY ("Vendor Company", "Vendor
name"),
    CONSTRAINT "FK1" FOREIGN KEY ("Club name and branch")
        REFERENCES clubbing_db."Club" ("Club name and branch") MATCH SIMPLE
        ON UPDATE cascade
        ON DELETE cascade
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing_db."Costs_bears"
    OWNER to postgres;
```

Electricity Costs:

```
CREATE TABLE clubbing_db."Electricity Costs"
(
    "Vendor Company" character varying NOT NULL,
    "Vendor name" character varying NOT NULL,
    "Equipment" character varying,
    "Room" character varying,
    "Power drawn" character varying,
    PRIMARY KEY ("Vendor Company", "Vendor name"),
    CONSTRAINT "FK1" FOREIGN KEY ("Vendor Company", "Vendor name")
        REFERENCES clubbing_db."Costs_bears" ("Vendor Company", "Vendor name")
MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."Electricity Costs"
    OWNER to postgres;
```

Event Costs:

```
CREATE TABLE clubbing_db."Event Costs"
(
    "Vendor Company" character varying NOT NULL,
    "Vendor name" character varying NOT NULL,
    "Performer fees" character varying,
    PRIMARY KEY ("Vendor Company", "Vendor name"),
```

```
    CONSTRAINT "FK1" FOREIGN KEY ("Vendor Company", "Vendor name")
        REFERENCES clubbing_db."Costs_bears" ("Vendor Company", "Vendor name")
MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."Event Costs"
    OWNER to postgres;
```

Part of:

```
CREATE TABLE clubbing_db."Part of"
(
    "Membership ID" character varying NOT NULL,
    "Club name and branch" character varying NOT NULL,
    PRIMARY KEY ("Membership ID", "Club name and branch"),
    CONSTRAINT "FK1" FOREIGN KEY ("Membership ID")
        REFERENCES clubbing_db."Members" ("Membership ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT "FK2" FOREIGN KEY ("Club name and branch")
        REFERENCES clubbing_db."Club" ("Club name and branch") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."Part of"
    OWNER to postgres;
```

Entry Fees:

```
CREATE TABLE IF NOT EXISTS clubbing_db."Entry Fees"
(
    "User_ID" character varying NOT NULL,
    "Event_ID" character varying NOT NULL,
    "Fees" character varying,
    CONSTRAINT "Entry Fees_pkey" PRIMARY KEY ("User_ID", "Event_ID"),
```

```
    CONSTRAINT "FK1" FOREIGN KEY ("User_ID")
        REFERENCES clubbing_db."Users" ("User_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT "FK2" FOREIGN KEY ("Event_ID")
        REFERENCES clubbing_db."Events" ("Event_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS clubbing_db."Entry Fees"
    OWNER to postgres;
```

PhoneNumber_Employee:

```
CREATE TABLE clubbing_db."PhoneNumber_Employee"
(
    "Employee_ID" character varying NOT NULL,
    "Phone Number" character varying NOT NULL,
    PRIMARY KEY ("Employee_ID", "Phone Number"),
    CONSTRAINT "FK1" FOREIGN KEY ("Employee_ID")
        REFERENCES clubbing_db."Employee_works_receives" ("Employee_ID")
MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."PhoneNumber_Employee"
    OWNER to postgres;
```

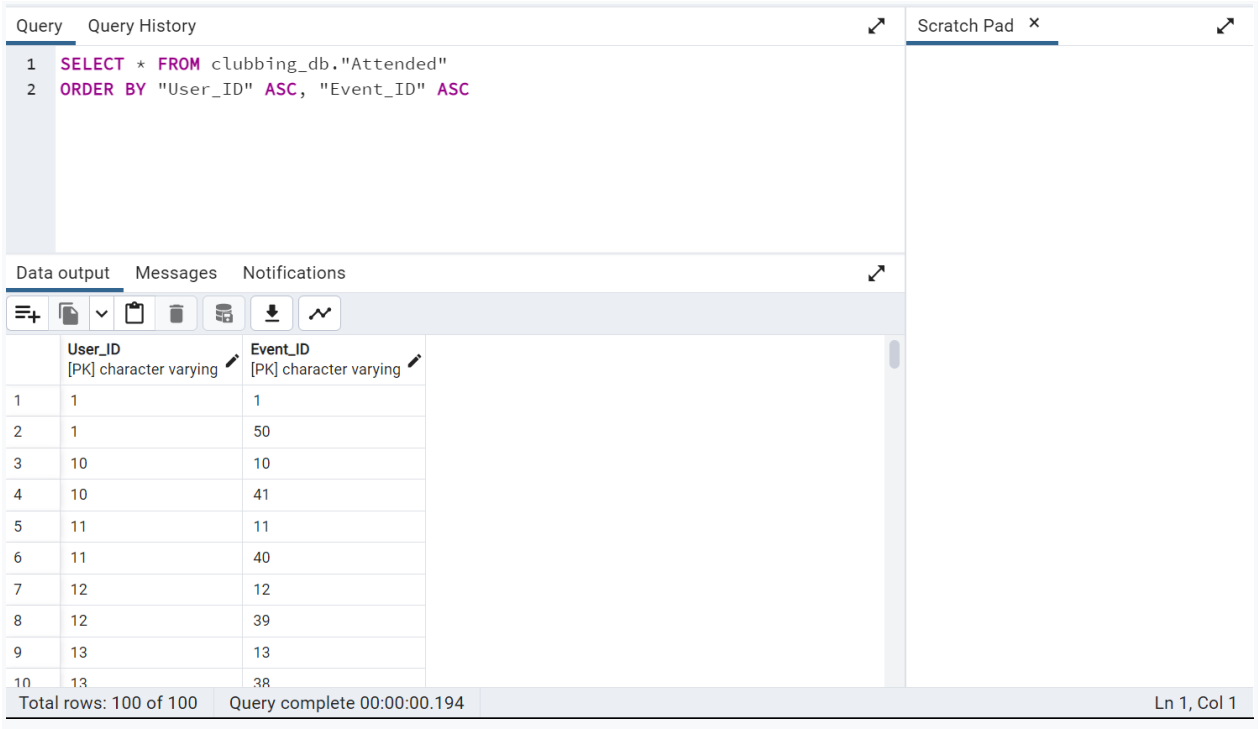
Membership amount and plan:

```
CREATE TABLE clubbing_db."MembershipAmount and Plan and Discounts"
(
    "Club name and branch" character varying NOT NULL,
    "Membership plan" character varying NOT NULL,
    "Membership amount" bigint NOT NULL,
    "Discounts" integer,
```

```
    PRIMARY KEY ("Club name and branch", "Membership plan"),
    CONSTRAINT fk1 FOREIGN KEY ("Club name and branch")
        REFERENCES clubbing_db."Club" ("Club name and branch") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE IF EXISTS clubbing_db."MembershipAmount and Plan and Discounts"
    OWNER to postgres;
```

DDL and data snapshots:



Query | Query History

```
1 SELECT * FROM clubbing_db."Club"
2 Loading... name and branch" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Club name and branch [PK] character varying	street character varying	Pincode character varying	Sound System character varying	Timing character varying	Cap cha
1	Aimbu	Norway Maple	643332	Pre Post Surgery	9:26 AM	852
2	Bluejam	Marcy	298270	daytime pe	4:01 PM	708
3	Bubblebox	Dryden	411666	OXYGEN	1:53 PM	517
4	Centizu	Burrows	533390	Daytime Sinus C...	8:30 PM	550
5	Demizz	Clove	108948	Ceftriaxone Sodi...	9:30 PM	380
6	Devify	Gerald	277180	Prednisone	3:28 PM	191
7	Dynabox	Lillian	426854	Haloperidol	4:40 AM	657
8	Eayo	Summit	698717	Trecator	10:20 PM	788
9	Fimbee	Briar Crest	845797	FSIKA	5:32 PM	490

Total rows: 20 of 20 | Query complete 00:00:00.146

Successfully run. Total query runtime: 146 msec. 20 rows affected.

clubbing_db.Club_2/202001006_db/postgres@PostgreSQL 14

No limit

Query | Query History

```
1 SELECT * FROM clubbing_db."Club_2"
2 Loading... code" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Pincode [PK] character varying	City character varying
1	010460	tbtdrd
2	015168	znrwey
3	038324	sjcwof
4	039069	rdvwjm
5	040646	aorvsi
6	045553	dhjxxh
7	057914	uxobto
8	073286	ilohnj
9	079384	osyyol

Total rows: 100 of 100 | Query complete 00:00:00.146

Successfully run. Total query runtime: 146 msec. 100 rows affected.

Query | Query History | Scratch Pad

```
SELECT * FROM clubbing_db."Cooks"
Loading... loyee_ID" ASC
```

Data output | Messages | Notifications

	Employee_ID [PK] character varying	Qualification character varying	Certification character varying
1	1	11th Pass	7882082341
2	10	B.tech	2505696756
3	11	B.E	6751839723
4	12	M.E	202750442
5	13	12th Pass	5028383138
6	14	B.tech	9838576646
7	15	B.E	2774579791
8	16	M.E	8146370705
9	17	12th Pass	8798329227

Total rows: 20 of 20 | Query complete 00:00:00.156

Successfully run. Total query runtime: 156 msec. 20 rows affected.

Query | Query History | Scratch Pad

```
SELECT * FROM clubbing_db."Costs_bears"
Loading... dor Company" ASC, "Vendor name" ASC
```

Data output | Messages | Notifications

	Vendor Company [PK] character varying	Vendor name [PK] character varying	Date character varying	Used for character varying	Cost per unit character varying
1	pR	Evita	31-01-1932	j	456
2	AUM	Daphne	29-10-1944	yg	865
3	AWT	Yolane	30-03-1932	RH	139
4	cbn	Helena	23-01-1903	Uh	974
5	cJfK	Jennica	08-09-2000	og	703
6	Crz	Devina	20-10-2008	Kq	216
7	DcTT	Sarette	26-08-1920	SG	288
8	dl	Leontine	21-01-1957	qU	127
9	Dn	Fina	06-02-1967	zO	374

Total rows: 40 of 40 | Query complete 00:00:00.190

Successfully run. Total query runtime: 190 msec. 40 rows affected.

Query | Query History

```
SELECT * FROM clubbing_db."Electricity Costs"
Loading...dor Company" ASC, "Vendor name" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Vendor Company character varying	Vendor name [PK] character varying	Equipment character varying	Room character varying	Power drawn character varying
1	[redacted]	Daphne	krah	ui MSbLp	399
2	cbn	Helena	mPTeh	lQvQ	708
3	cJfK	Jennica	OtUWs	mIAaicn	291
4	DcTT	Sarette	xhaxJ	bXWbIW	328
5	Dn	Fina	nuCXF	rrtedBX	9
6	fE	Malina	RwWNf	kTQoMEvD	668
7	JaPr	Brianna	APoi	ZPObRXKD	393
8	kN	Eolanda	nQfj	ISOq	634
9	kTcr	Ginnie	CSBvb	gREmex	102

Paste
Accesskey P

Total rows: 20 of 20 | Query complete 00:00:00.198

Successfully run. Total query runtime: 198 msec. 20 rows affected.

Query | Query History

```
SELECT * FROM clubbing_db."Employee_works_receives"
Loading...loyee_ID" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Employee_ID [PK] character varying	Club Name and brach character varying	Basic character varying	Bonus character varying	Tax character varying
1	1	Leenti	57422	5670	8283
2	10	Eayo	40174	4636	1604
3	100	Youspan	80285	545	895
4	11	Katz	67650	3906	2216
5	12	Centizu	78319	4659	185
6	13	Bubblebox	96038	2427	2473
7	14	Demizz	37534	5005	5384
8	15	Devify	23156	3247	7583
9	16	Skidoo	53660	5570	8465

Total rows: 100 of 100 | Query complete 00:00:00.186

Successfully run. Total query runtime: 186 msec. 100 rows affected.

Query | Query History

```
1  SELECT * FROM clubbing_db."Entry Fees"
2  Loading...  _ID" ASC, "Event_ID" ASC
```

Scratch Pad

Data output | Messages | Notifications

	User_ID [PK] character varying	Event_ID [PK] character varying	Fees character varying
1	1	28	2234
2	16	1	100
3	17	2	2234
4	18	3	345
5	19	4	3445
6	20	5	100
7	21	6	2234
8	22	7	345
9	23	8	3445

Total rows: 28 of 28 | Query complete 00:00:00.246 | Ln 1, Col 1

Query | Query History

```
1  SELECT * FROM clubbing_db."Event Costs"
2  Loading...  dor Company" ASC, "Vendor name" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Vendor Company [PK] character varying	Vendor name [PK] character varying	Performer fees character varying
1	pR	Evita	3445
2	AWT	Yolane	2234
3	Crz	Devina	100
4	dI	Leontine	2234
5	dU	Teddie	345
6	dZna	Tina	345
7	fN	Luci	100
8	Gvgl	Sindee	100
9	hVQF	Deane	3445

Total rows: 20 of 20 | Query complete 00:00:00.184

Successfully run. Total query runtime: 184 msec. 20 rows affected.

Query | Query History

```
1 SELECT * FROM clubbing_db."Events"
2 Loading... nt_ID" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Event_ID [PK] character varying	DJ/orchestra timetable character varying	Nature of event character varying	Time character varying	Date character varying
1	1	6:26 PM	96757oybl	11:32 PM	4/8/2022
2	10	5:44 PM	33315dybl	8:57 AM	5/26/2022
3	11	6:09 AM	61926aybl	1:25 PM	10/9/2022
4	12	1:47 AM	87527zybl	12:29 AM	8/16/2022
5	13	6:38 PM	74391cybl	9:41 AM	1/17/2022
6	14	9:02 AM	68855rybl	8:35 AM	7/31/2022
7	15	2:35 PM	42053cybl	9:35 AM	9/5/2022
8	16	4:44 PM	78906nybl	2:38 AM	8/12/2022
9	17	5:18 AM	48444rybl	10:04 AM	6/16/2022

Total rows: 50 of 50 | Query complete 00:00:00.182

Successfully run. Total query runtime: 182 msec. 50 rows affected.

Query | Query History

```
1 SELECT * FROM clubbing_db."FinancialInformation_PaysUsing"
2 Loading... ount number" ASC, "Card number" ASC, "Virtual Payment Address" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Account number [PK] character varying	Card number [PK] character varying	Virtual Payment Address [PK] character varying	Full Name [PK] character varying	User_ID character
1	00139362	9953916255	44718mybl	Madelene MacMearty	25
2	03541244	4811054145	27655kybl	Kalila O'Carney	7
3	04157451	8743740186	56221rybl	Dorthea Davinet	12
4	06070580	3836990643	40871zybl	Joane Kolakovic	31
5	06728041	4672810003	77074dybl	Adaline Dumbrill	26
6	07208359	5991174224	13830wybl	Vincents Dyers	32
7	07570483	9146107367	65702pybl	Brod Clyburn	5
8	09104452	5880808666	71514oybl	Allis Bugdale	30
9	10496898	0221580543	82979nybl	Allin Vescovo	16

Total rows: 50 of 50 | Query complete 00:00:00.159

Successfully run. Total query runtime: 159 msec. 50 rows affected.

Query | Query History

```
1 SELECT * FROM clubbing_db."Maintaince staff"
2 Loading... loyee_ID" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Employee_ID [PK] character varying	Posting location character varying	Shift character varying
1	41	Waywood	8:32 AM
2	42	Hollow Ridge	8:15 AM
3	43	Golf	2:26 AM
4	44	Arrowood	11:41 PM
5	45	Muir	2:11 PM
6	46	Vermont	10:13 AM
7	47	Elmside	11:46 AM
8	48	Dawn	4:33 AM
9	49	Summit	4:42 AM

Total rows: 20 of 20 | Query complete 00:00:00.195

Successfully run. Total query runtime: 195 msec. 20 rows affected.

Query | Query History

```
1 SELECT * FROM clubbing_db."Members"
2 Loading... r_ID" ASC
```

Scratch Pad

Data output | Messages | Notifications

	User_ID [PK] character varying	Membership ID character varying	Membership plan type character varying	Address character varying	Thumb Impression character varying
1	1	f82838d2-fb77-4f...	Mauv	7833 Roth Pass	GT97 IUXC BR0M...
2	10	ab858521-16f7-4...	Goldenrod	73472 Colorado ...	HR12 3362 8636 ...
3	11	8914484f-db16-4...	Indigo	911 Stone Corner...	LT59 2807 3054 ...
4	12	04dd5872-700f-4...	Orange	97483 Bonner Cr...	RO81 ABME MF8...
5	13	6eb941db-9f70-4...	Khaki	3 Fair Oaks Court	LT91 4714 5786 ...
6	14	8cc90e00-e135-4...	Indigo	693 Jana Center	FR78 5150 9354 ...
7	15	53781dae-601c-4...	Red	78 Springs Parkw...	IL74 6366 8744 9...
8	16	1dc96381-47c7-4...	Indigo	9 Ilene Parkway	EE91 6256 7864 ...
9	17	fe1bb898-e826-4...	Goldenrod	53515 Manitowis...	CZ76 2674 8446 ...

Total rows: 40 of 40 | Query complete 00:00:00.221

Successfully run. Total query runtime: 221 msec. 40 rows affected.

Query | Query History

```
1 SELECT * FROM clubbing_db."MembershipAmount and Plan and Discounts"
2 Loading... name and branch" ASC, "Membership plan" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Club name and branch [PK] character varying	Membership plan [PK] character varying	Membership amount character varying	Discounts character varying
1	Aimbu	Red	232	23
2	Bluejam	Orange	9987	12
3	Bubblebox	Khaki	2342	56
4	Centizu	Orange	9987	34
5	Demizz	Indigo	300	21
6	Devify	Red	232	23
7	Dynabox	Indigo	300	234
8	Eayo	Goldenrod	200	99
9	Eimbee	Purple	100	2

Total rows: 19 of 19 | Query complete 00:00:00.452

Successfully run. Total query runtime: 452 msec. 19 rows affected.

Query | Query History

```
1 SELECT * FROM clubbing_db."Part of"
2 Loading... ership ID" ASC, "Club name and branch" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Membership ID [PK] character varying	Club name and branch [PK] character varying
1	014dd2e0-a3f2-4ef2-...	Kazio
2	04dd5872-700f-4ae8-...	Centizu
3	0ab04d32-5b4d-4cb7-...	Aimbu
4	10ae7ffb-c0d4-476b-...	Trudeo
5	198c6dbc-c8ce-4381-...	Bubblebox
6	1c186821-eb55-4d42-...	Kazio
7	1cc74344-63c3-4902-...	Twimm
8	1d6e2de6-f0ff-4396-8...	Twimm
9	1dc96381-47c7-41cf-...	Skidoo

Total rows: 40 of 40 | Query complete 00:00:00.236

Successfully run. Total query runtime: 236 msec. 40 rows affected.

Query | Query History | Scratch Pad

```
SELECT * FROM clubbing_db."PhoneNumber_Employee"
Loading... loyee_ID" ASC, "Phone Number" ASC
```

Data output | Messages | Notifications

	Employee_ID [PK] character varying	Phone Number [PK] character varying
1	1	820-266-9718
2	1	820-266-9720
3	10	292-590-0584
4	10	292-590-0586
5	100	139-464-6849
6	100	139-464-6851
7	11	840-727-8435
8	11	840-727-8437
9	12	220-627-1966

Total rows: 200 of 200 | Query complete 00:00:00.240

Successfully run. Total query runtime: 240 msec. 200 rows affected.

Query | Query History | Scratch Pad

```
SELECT * FROM clubbing_db."Salary"
Loading... ic" ASC, "Bonus" ASC, "Tax" ASC
```

Data output | Messages | Notifications

	Basic [PK] character varying	Bonus [PK] character varying	Tax [PK] character varying	Total character varying
1	10746	6695	8061	9380
2	14719	9969	4047	20641
3	16738	3987	1365	19360
4	17253	6280	4785	18748
5	21871	9983	5215	26639
6	23156	3247	7583	18820
7	24469	5674	4459	25684
8	27532	5998	9731	23799
9	29191	4577	4346	29422

Total rows: 50 of 50 | Query complete 00:00:00.437

Successfully run. Total query runtime: 437 msec. 50 rows affected.

Query | Query History

```
1 SELECT * FROM clubbing_db."Security"
2 Loading... loyee_ID" ASC
```

Scratch Pad

Data output | Messages | Notifications

	Employee_ID [PK] character varying	Posting location character varying	Shift character varying
1	21	Waywood	8:32 AM
2	22	Hollow Ridge	8:15 AM
3	23	Golf	2:26 AM
4	24	Arrowood	11:41 PM
5	25	Muir	2:11 PM
6	26	Vermont	10:13 AM
7	27	Elmside	11:46 AM
8	28	Dawn	4:33 AM
9	29	Summit	4:42 AM

Total rows: 20 of 20 | Query complete 00:00:00.218

Successfully run. Total query runtime: 218 msec. 20 rows affected.

Query | Query History

```
1 SELECT * FROM clubbing_db."Timeline_Describe"
2 Loading... nt_ID" ASC, "Performance" ASC
```

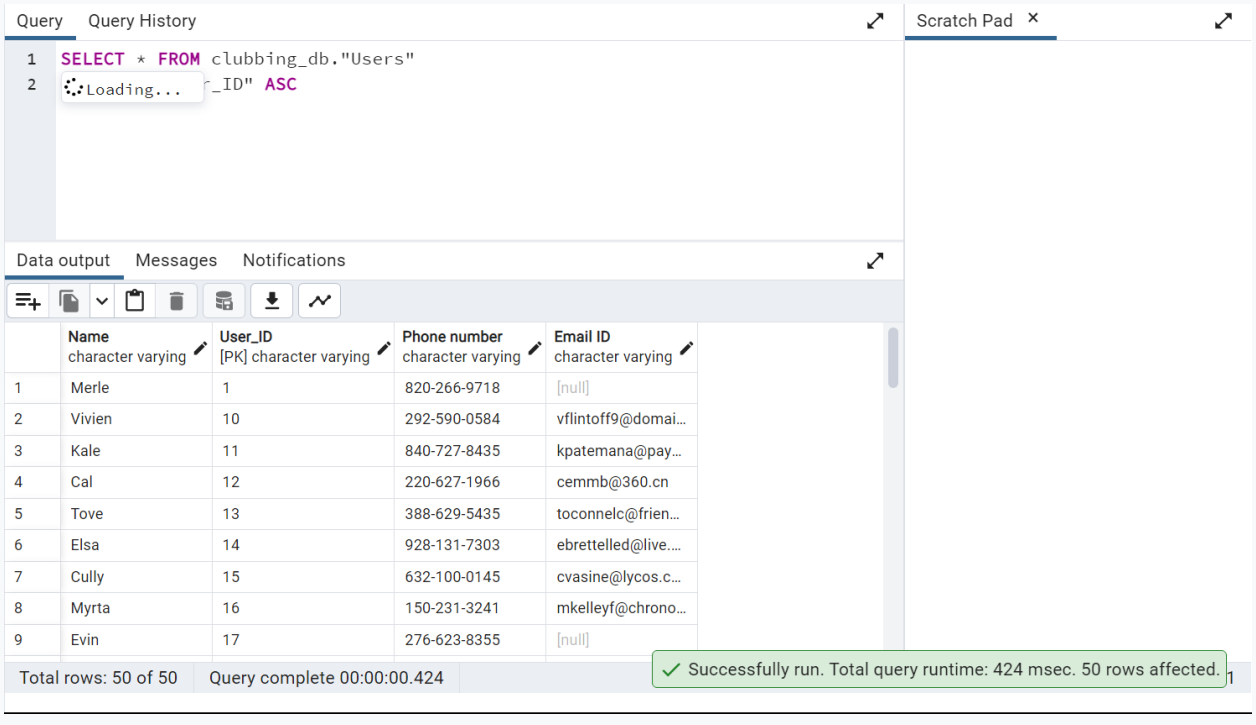
Scratch Pad

Data output | Messages | Notifications

	Event_ID [PK] character varying	Performance [PK] character varying	Performer character varying	Duration character varying
1	1	Matsoft	Grazia Crane	10:53 AM
2	1	Voltsillam	Bren Billsberry	4:20 PM
3	10	Asoka	Wanda Siemons	12:30 PM
4	10	Bytecard	Reagen Dimblebee	2:52 PM
5	11	Lotstring	Milt Layfield	4:43 AM
6	11	Redhold	Aluin Wickham	5:05 PM
7	12	Flowdesk	Zonda Fuggle	12:12 PM
8	12	Hatity	Isa Phelp	8:05 PM
9	13	Bamity	Burton Brookwell	10:57 PM

Total rows: 100 of 100 | Query complete 00:00:00.528

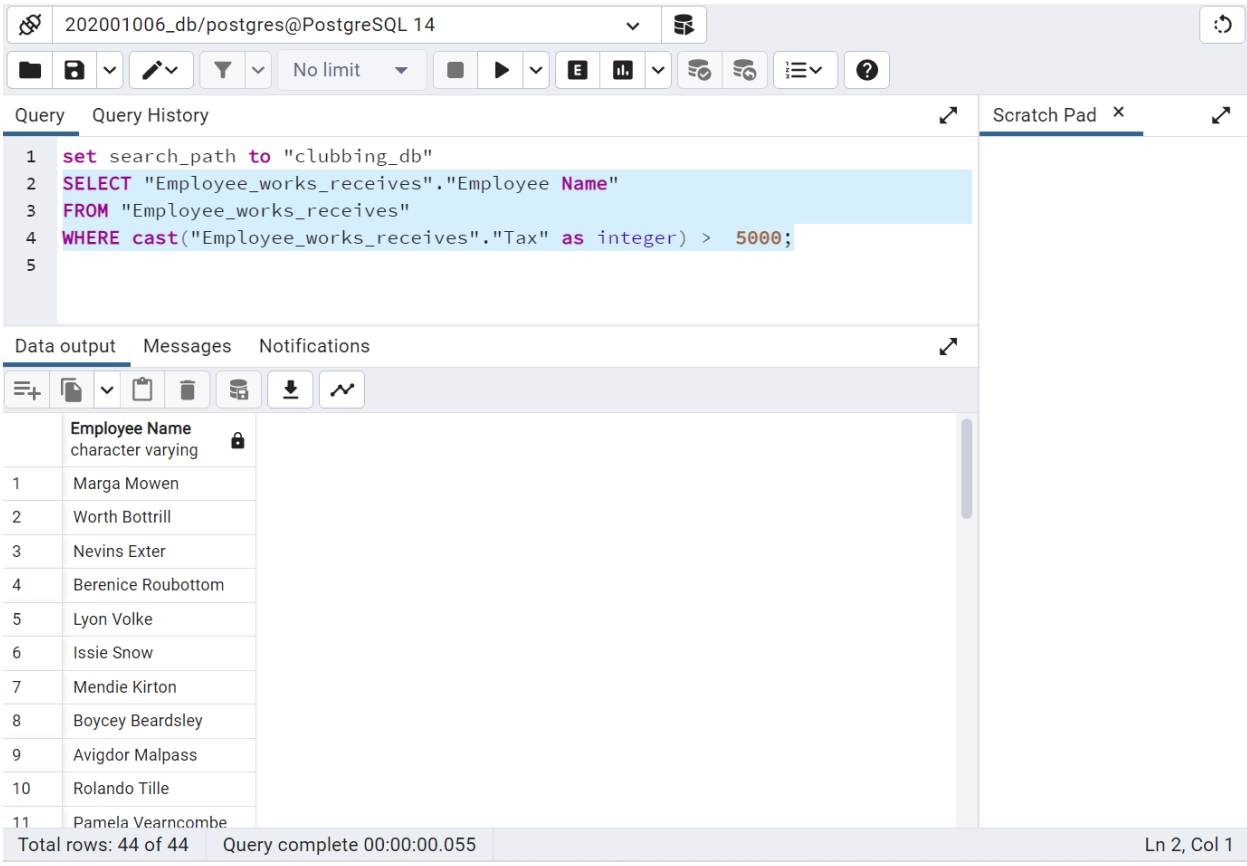
Successfully run. Total query runtime: 528 msec. 100 rows affected.



Queries

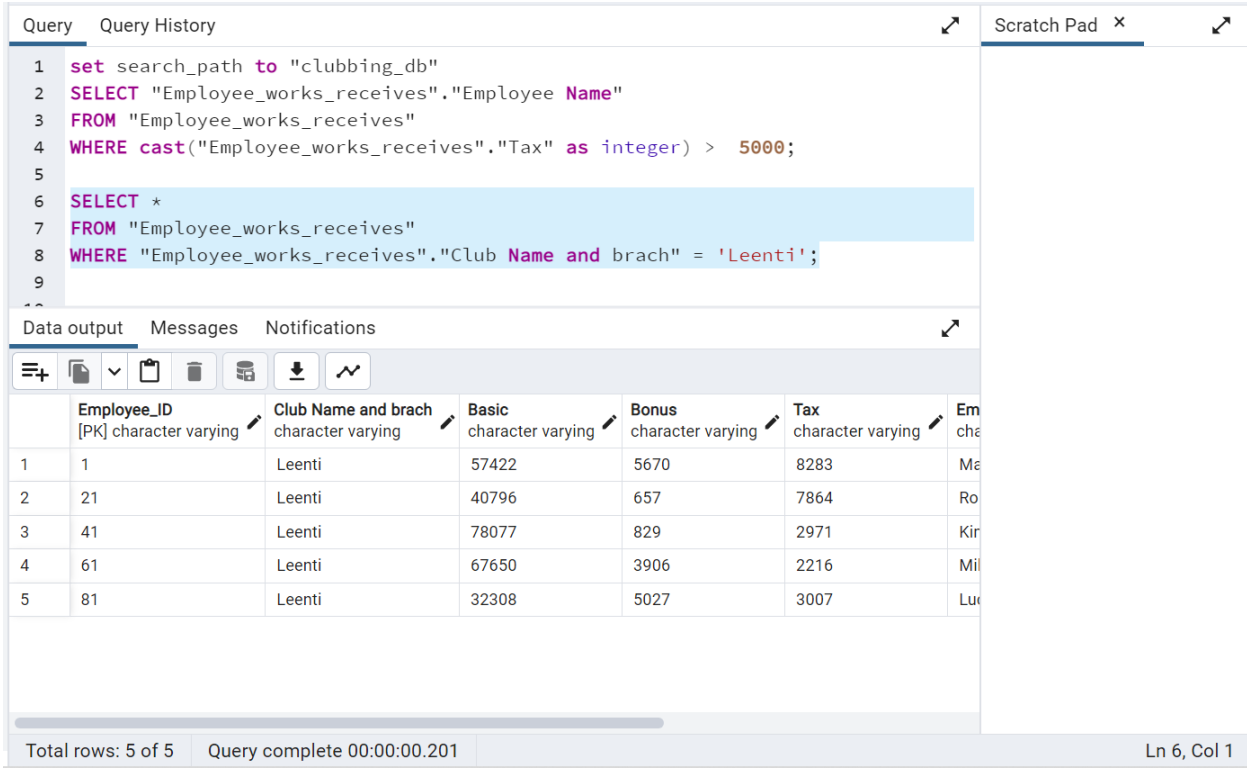
1. All employee names paying more than ‘k’ rupees tax

```
SELECT "Employee_works_receives"."Employee Name"
FROM "Employee_works_receives"
WHERE cast("Employee_works_receives"."Tax" as integer) >  5000;
```



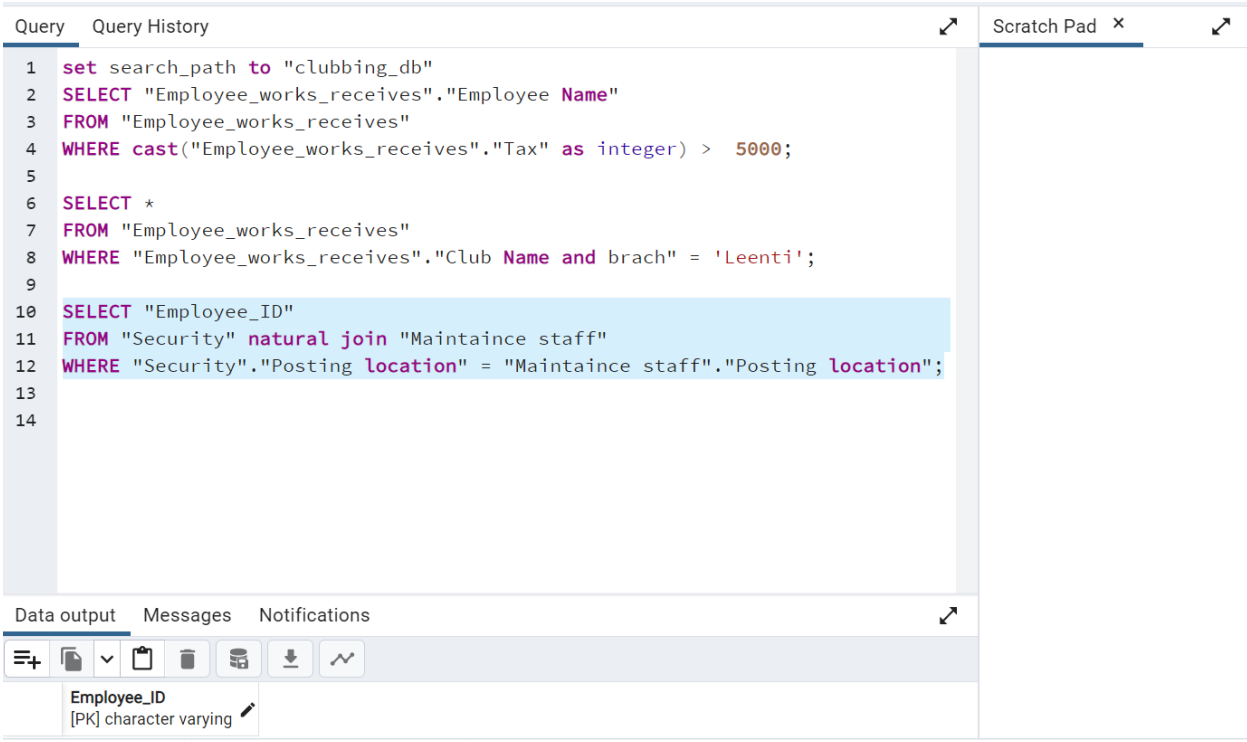
2. All employees in a certain club

```
SELECT *
FROM "Employee_works_receives"
WHERE "Employee_works_receives"."Club Name and brach" = 'Leenti';
```



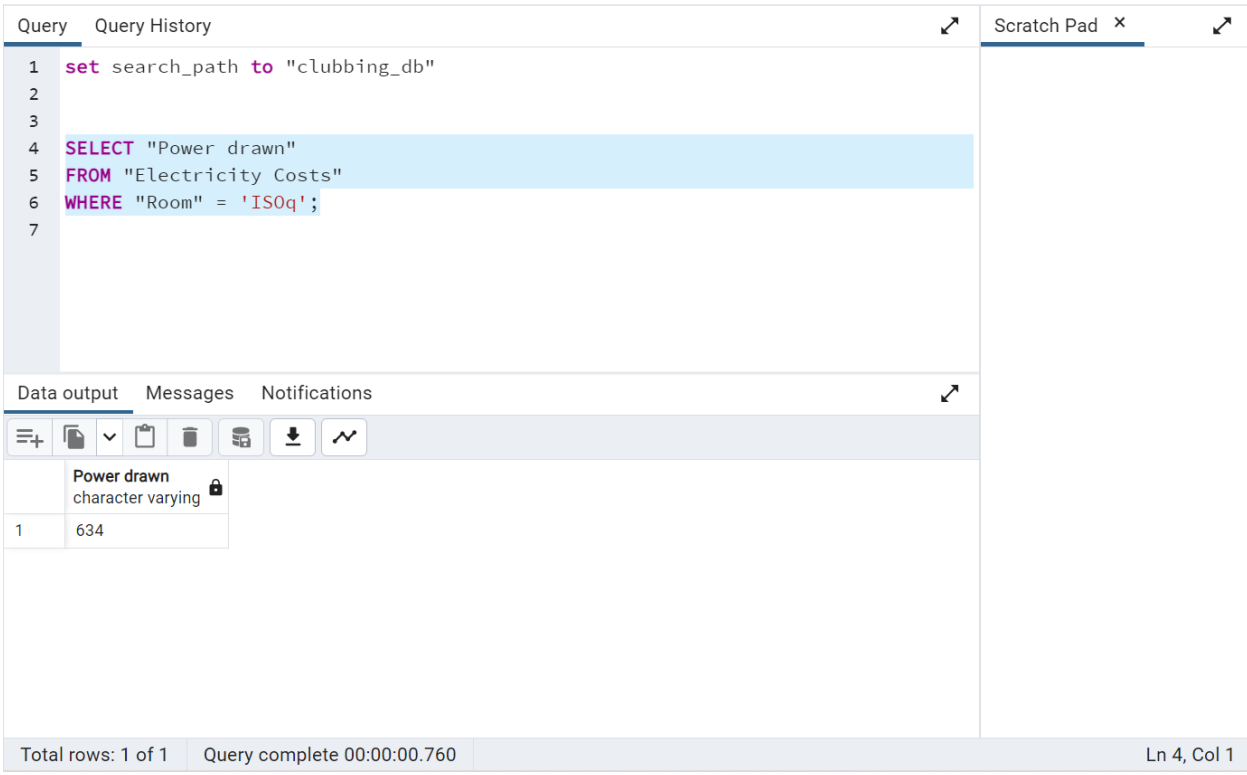
3. © All maintenance and security staff at a given (x) location

```
SELECT "Employee_ID"
FROM "Security" natural join "Maintaince staff"
WHERE "Security"."Posting location" = "Maintaince staff"."Posting location";
```



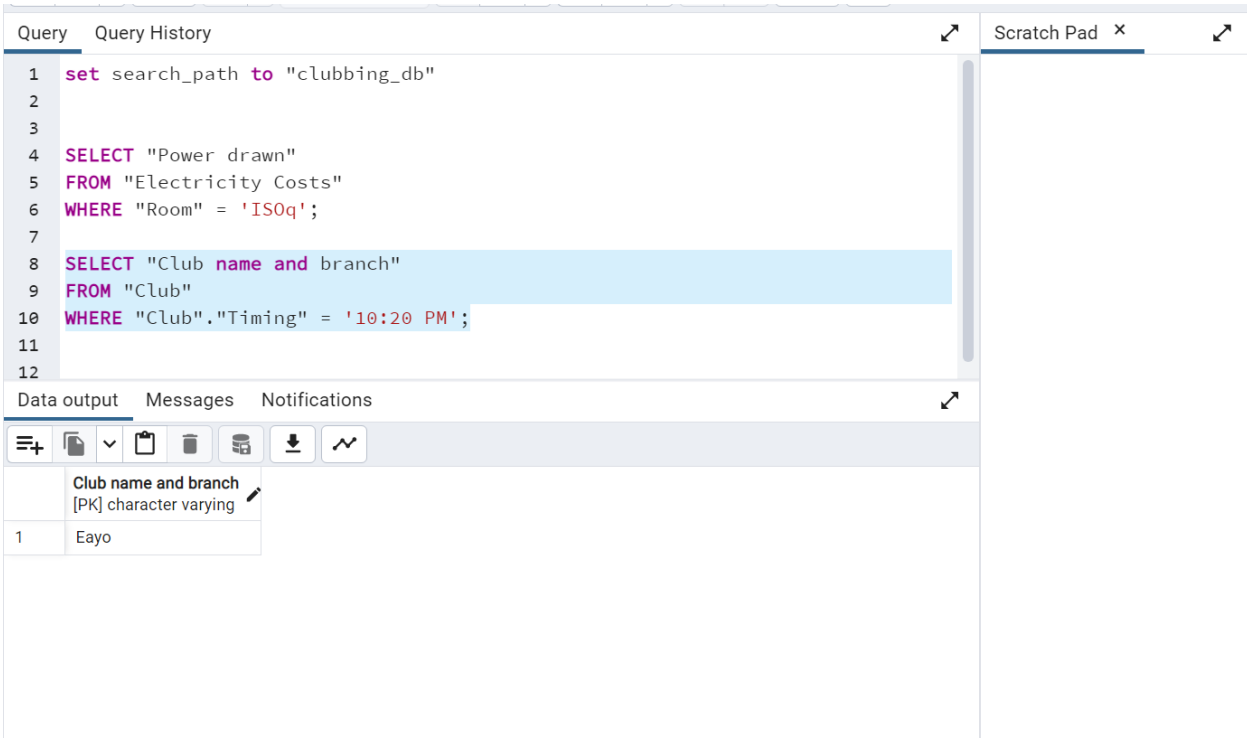
4. © Total money (= qty*cost_per_unit + tax) used by a particular branch
    - CREATE FUNCTION GetTotal (int quantity, int cost_per_unit, int Tax)
        RETURNS integer
        LANGUAGE SQL
        MODIFIES SQL
        BEGIN
            DECLARE tot DECIMAL
            tot = quantity*cost_per_unit+Tax
        RETURN tot;
    END

5. Total power drawn by a particular room
    SELECT "Power drawn"
    FROM "Electricity Costs"
    WHERE "Room" = 'ISOq';



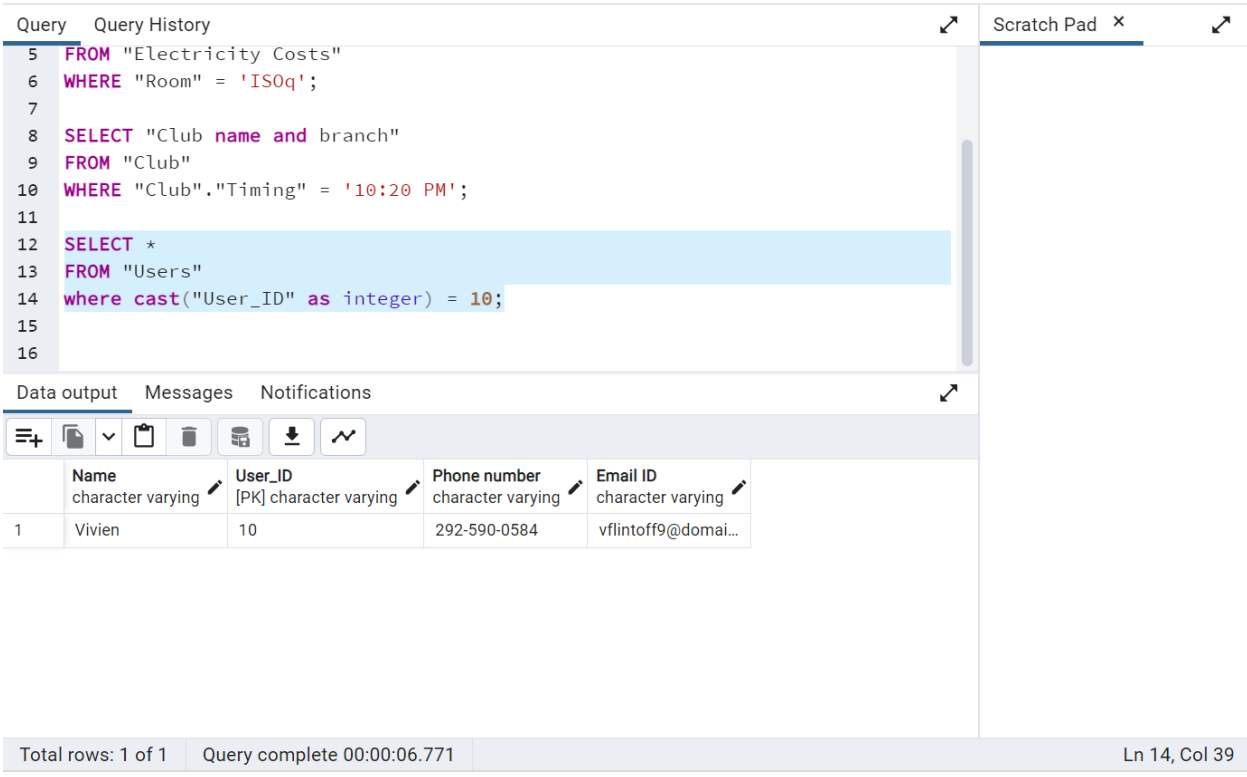
6. All clubs with club timing equal to 10:20PM.

```
SELECT "Club name and branch"
FROM "Club"
WHERE "Club"."Timing" = '10:20 PM';
```

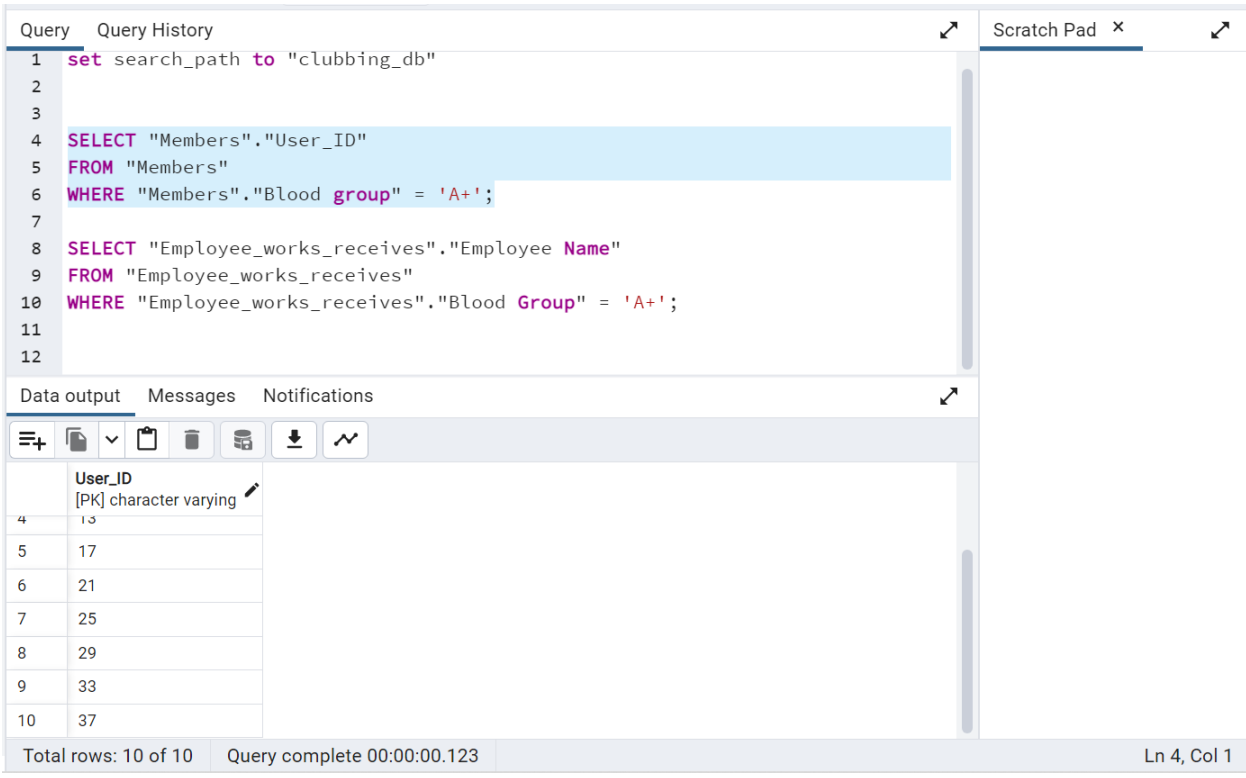


7. All details of a particular user

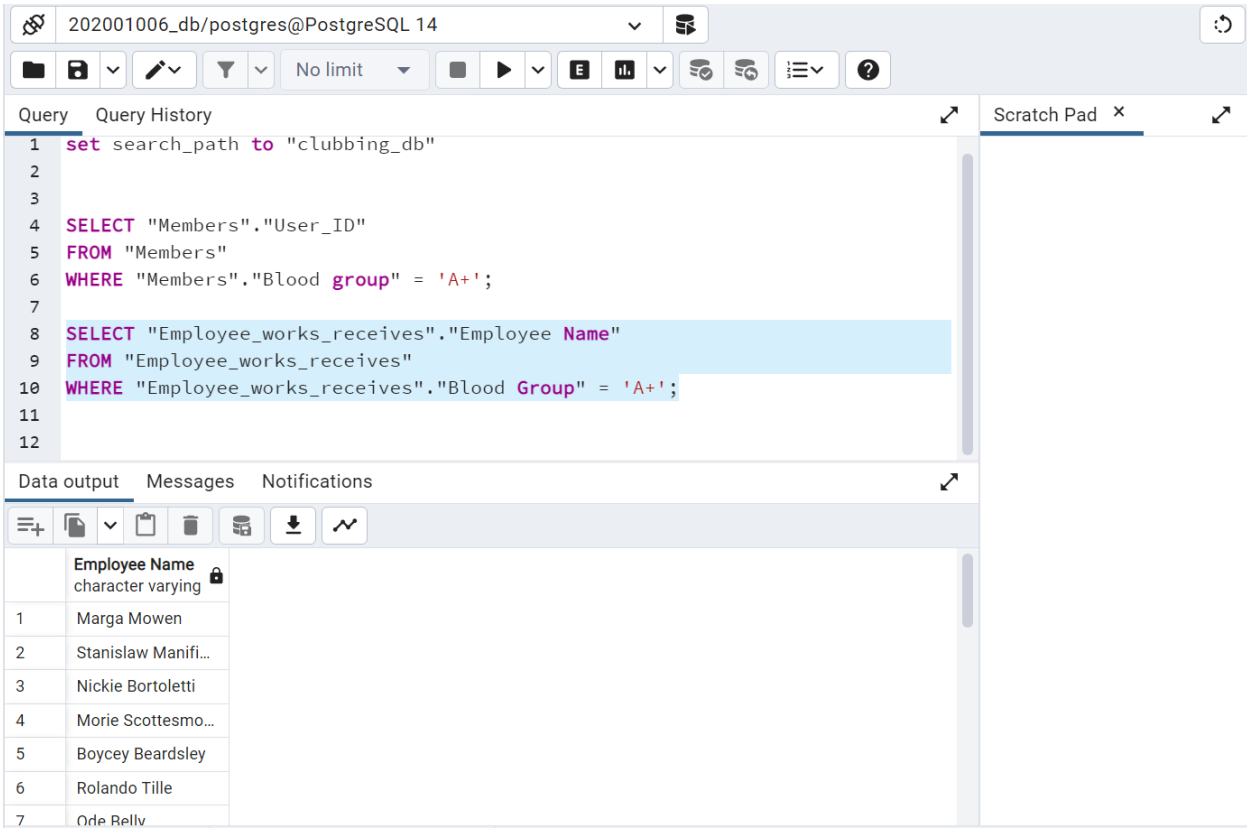
```
SELECT *
FROM "Users"
where cast("User_ID" as integer) = 10;
```



8. © All members and employees with a certain blood group (in case of emergencies, this might be helpful) (No join used since time used to run should be less)
SELECT "Members"."User_ID"
FROM "Members"
WHERE "Members"."Blood group" = 'A+';

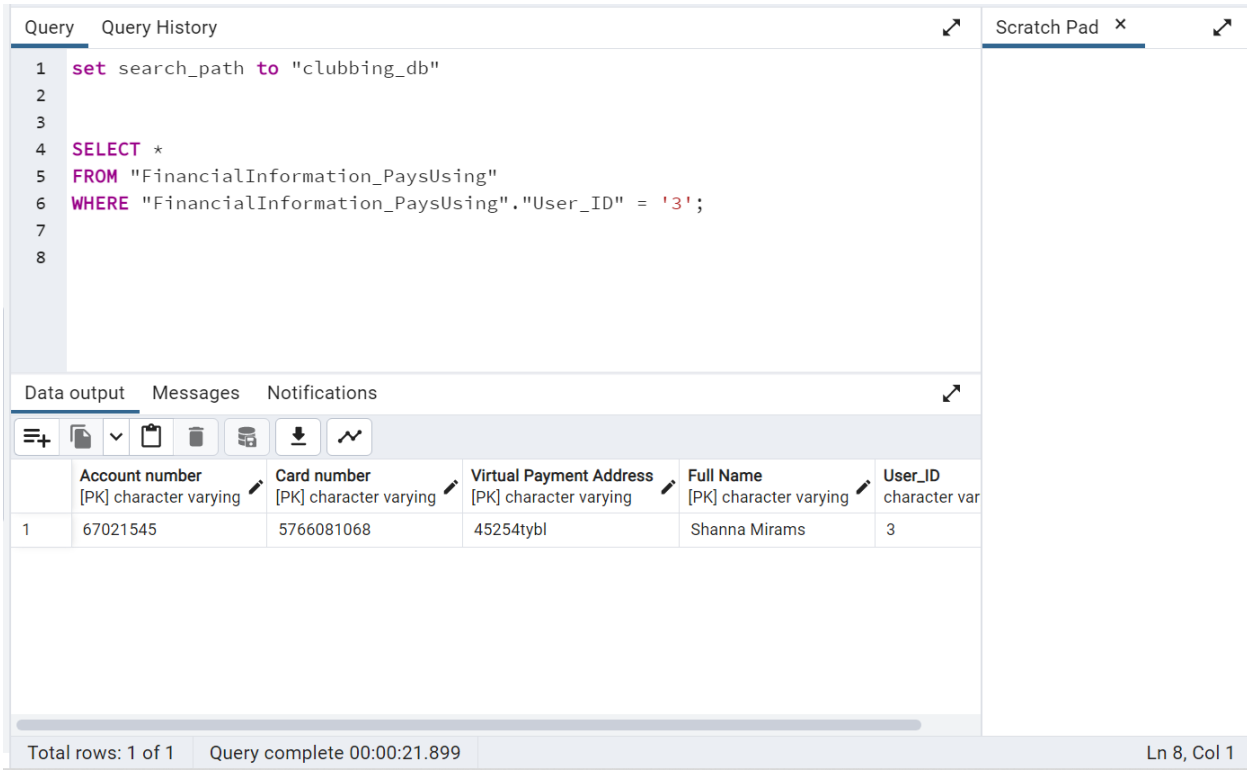


SELECT "Employee_works_receives"."Employee Name"
FROM "Employee_works_receives"
WHERE "Employee_works_receives"."Blood Group" = 'A+';



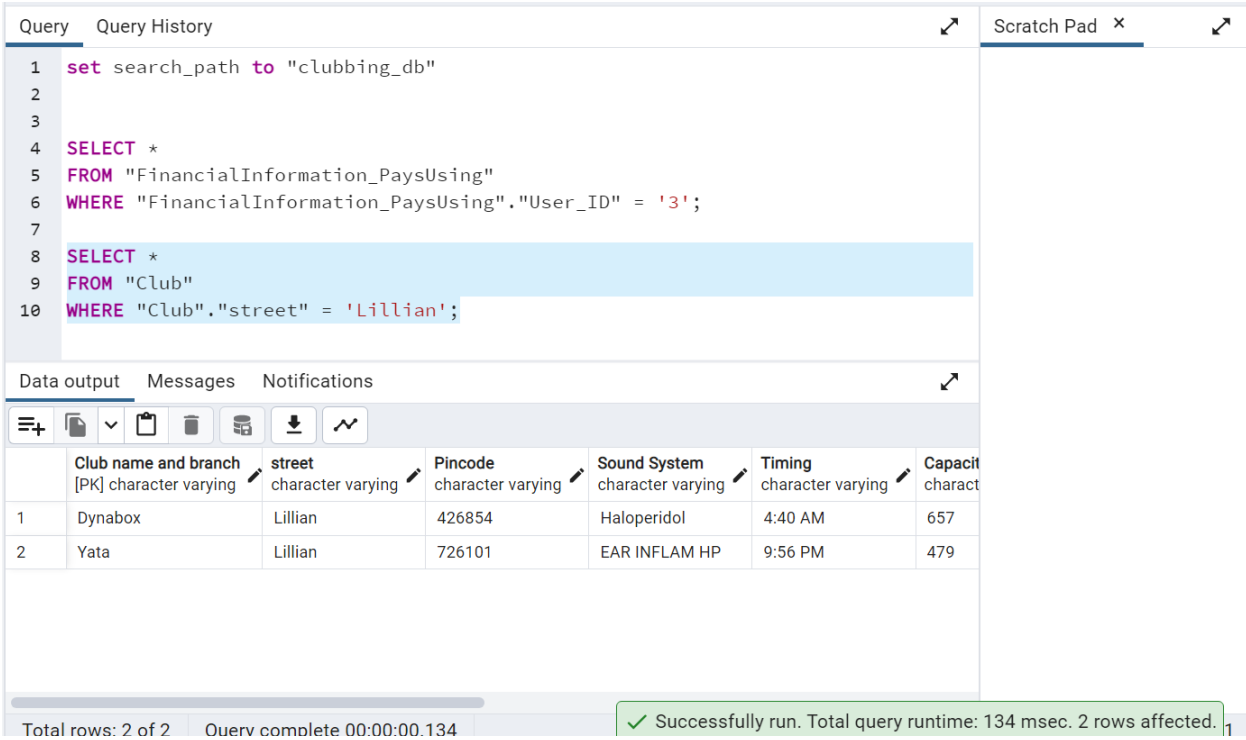
9. All financial information of a particular user.

```
SELECT *
FROM "FinancialInformation_PaysUsing"
WHERE "FinancialInformation_PaysUsing"."User_ID" = '3';
```



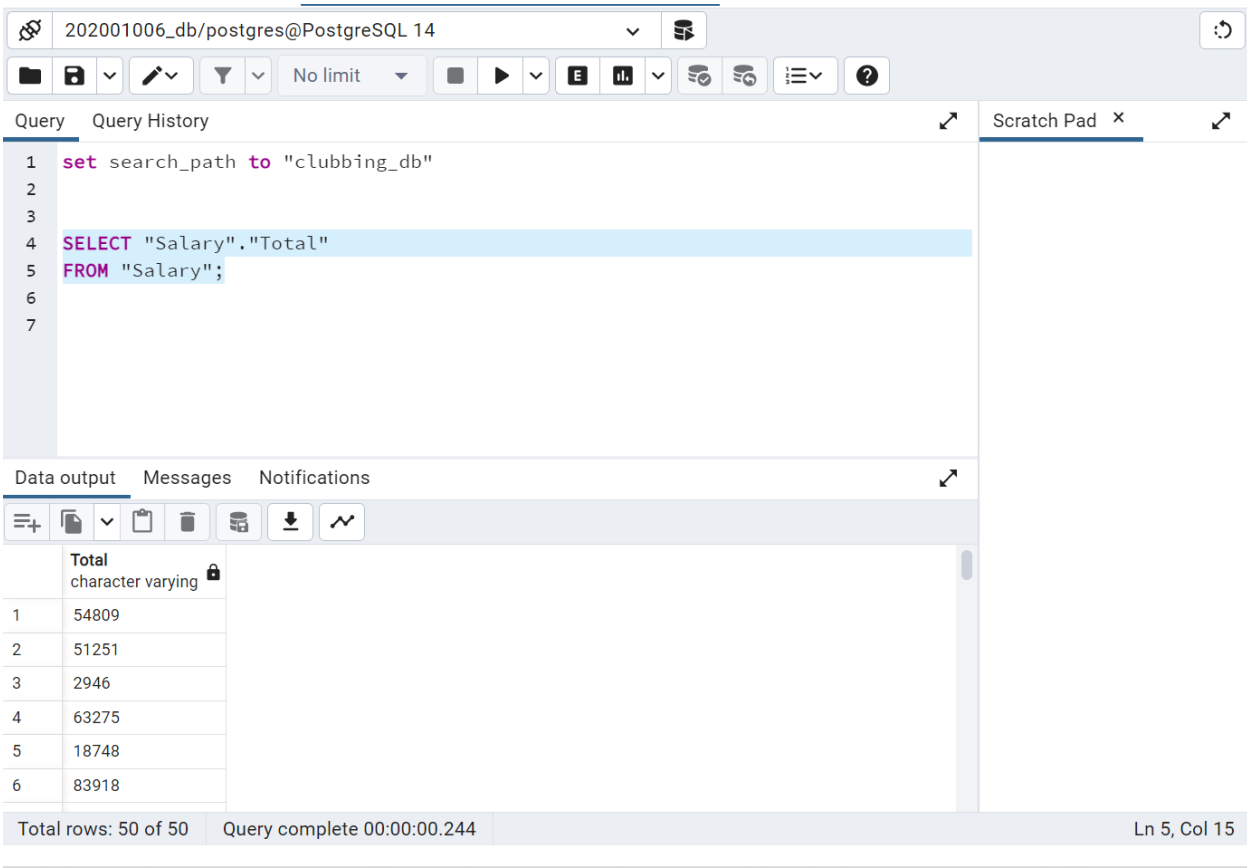
10. © All clubs in a particular street.

```
SELECT *
FROM "Club"
WHERE "Club"."street" = 'Lillian';
```



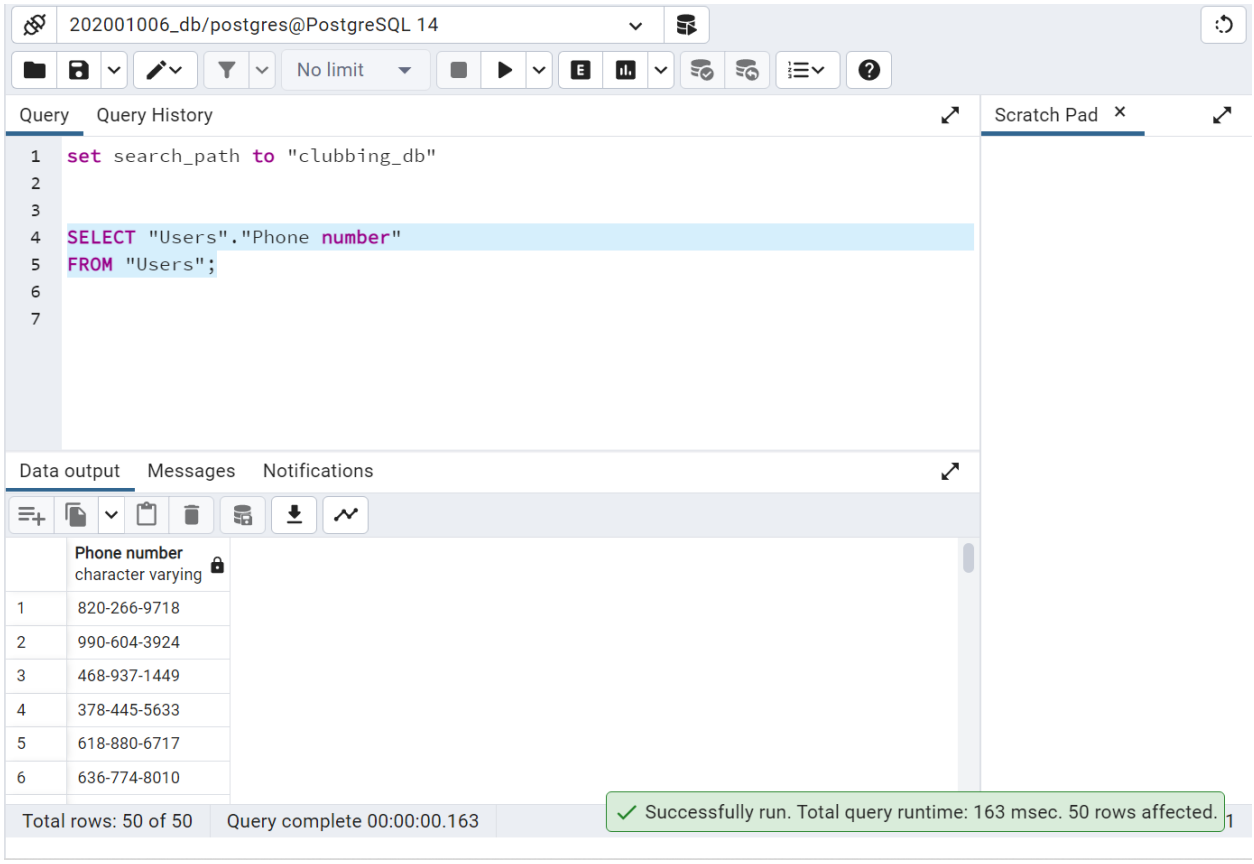
11. Salaries of all employees

```
SELECT "Salary"."Total"
FROM "Salary";
```



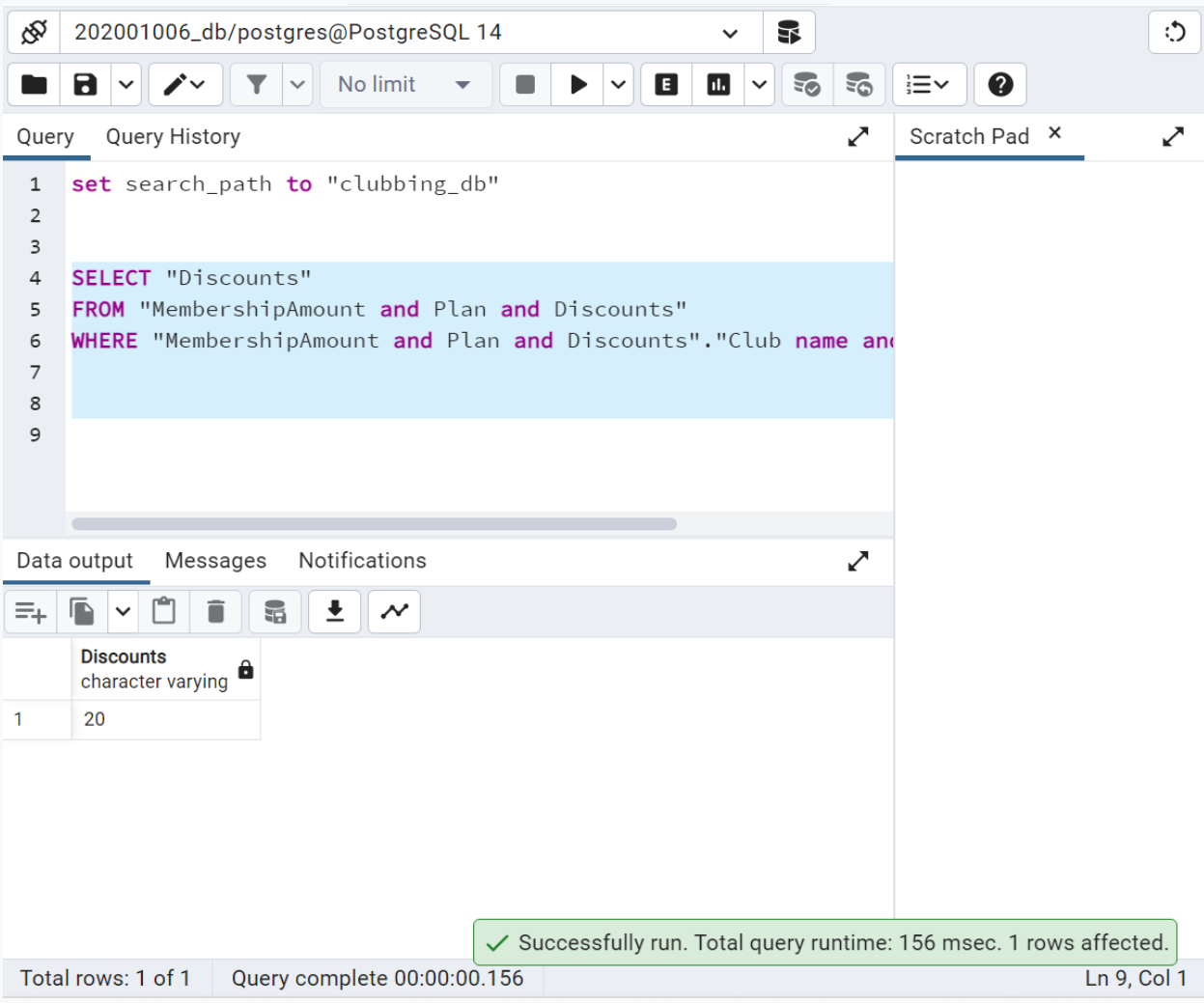
12. All contact numbers of all users

```
SELECT "Users"."Phone number"
FROM "Users";
```



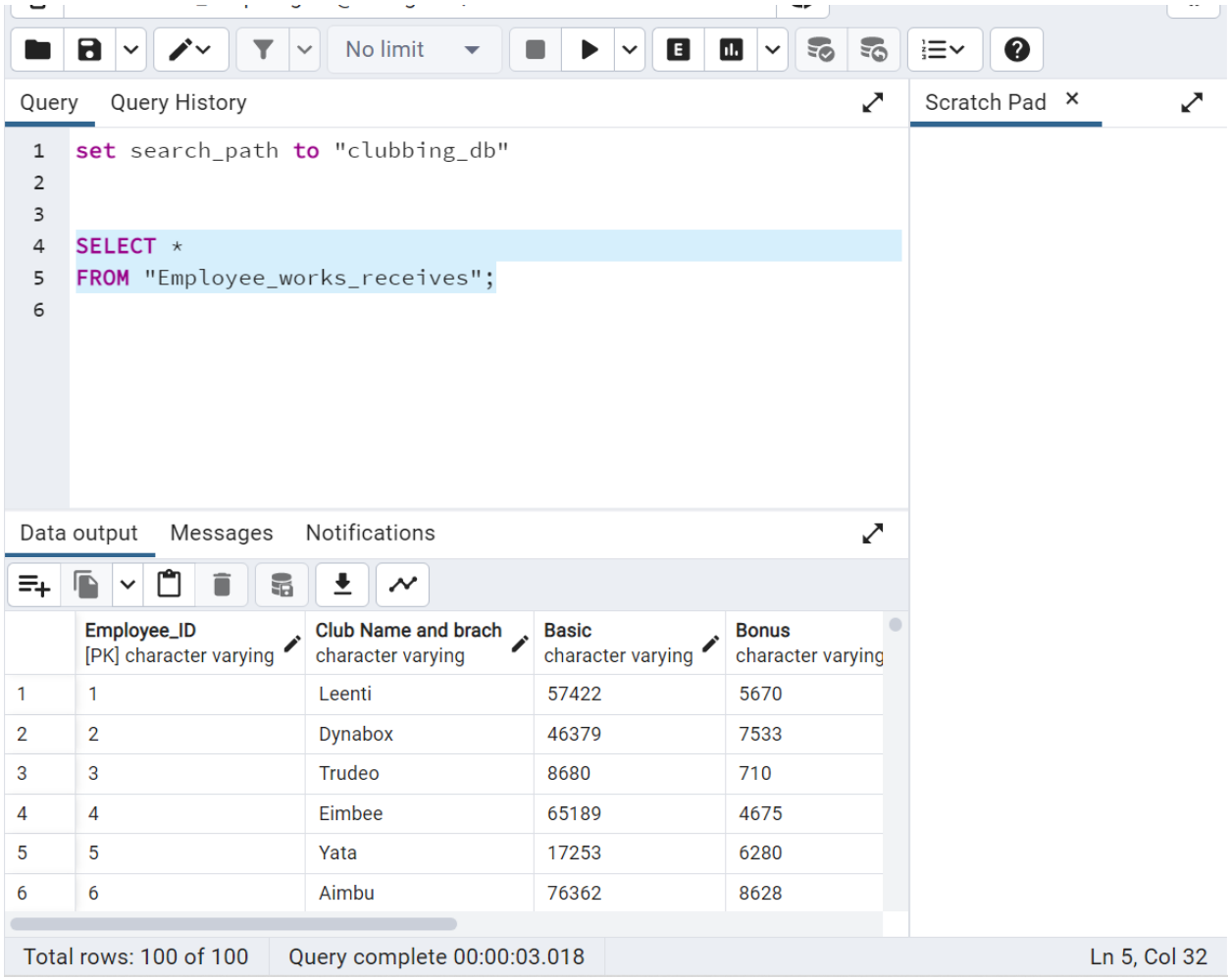
13. All discounts and group booking in a particular branch (using Membership Amount and Plan and Discounts/Group bookings)

```
SELECT "Discounts"
FROM "MembershipAmount and Plan and Discounts"
WHERE "MembershipAmount and Plan and Discounts"."Club name and branch"
= 'Leenti';
```



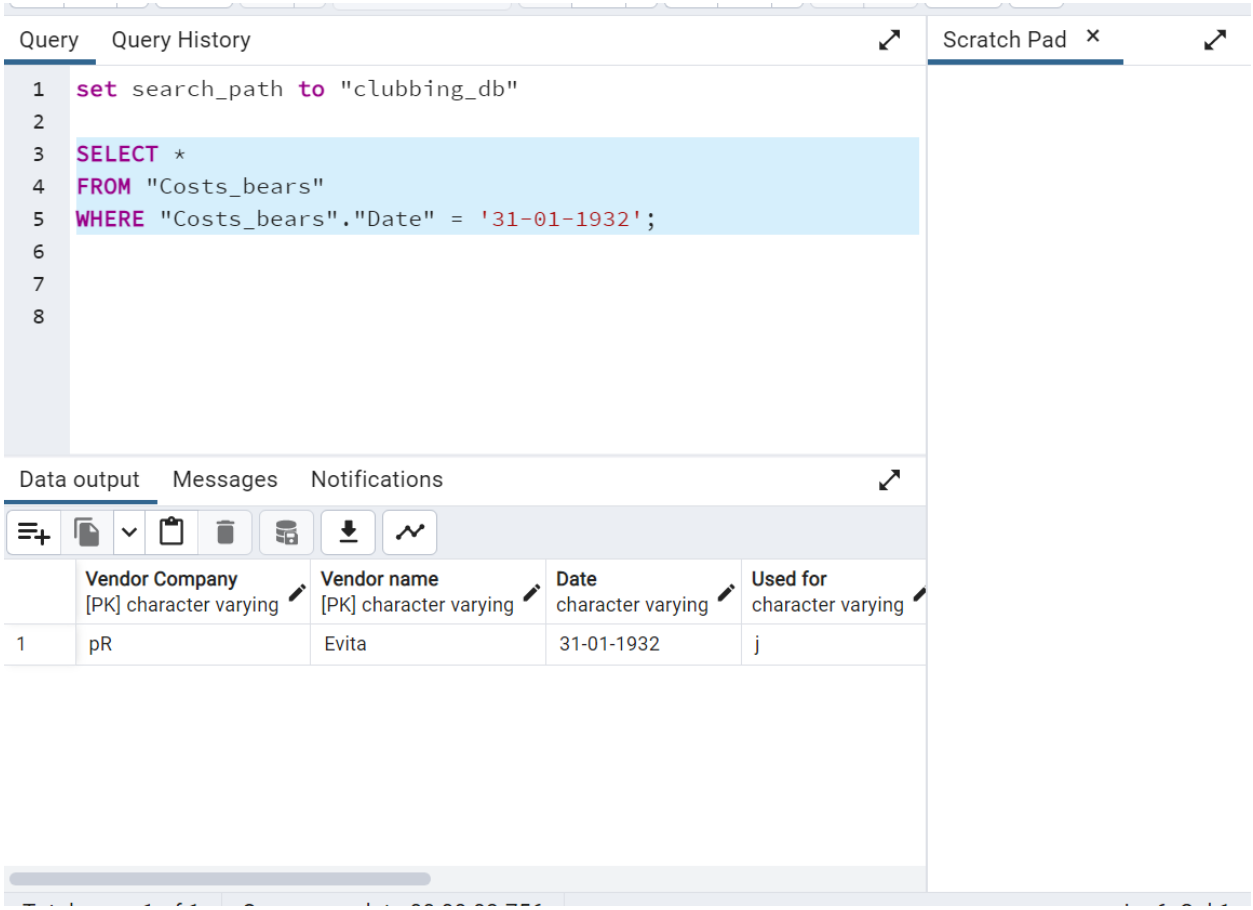
14. All information of all employees

```
SELECT *
FROM "Employee_works_receives";
```



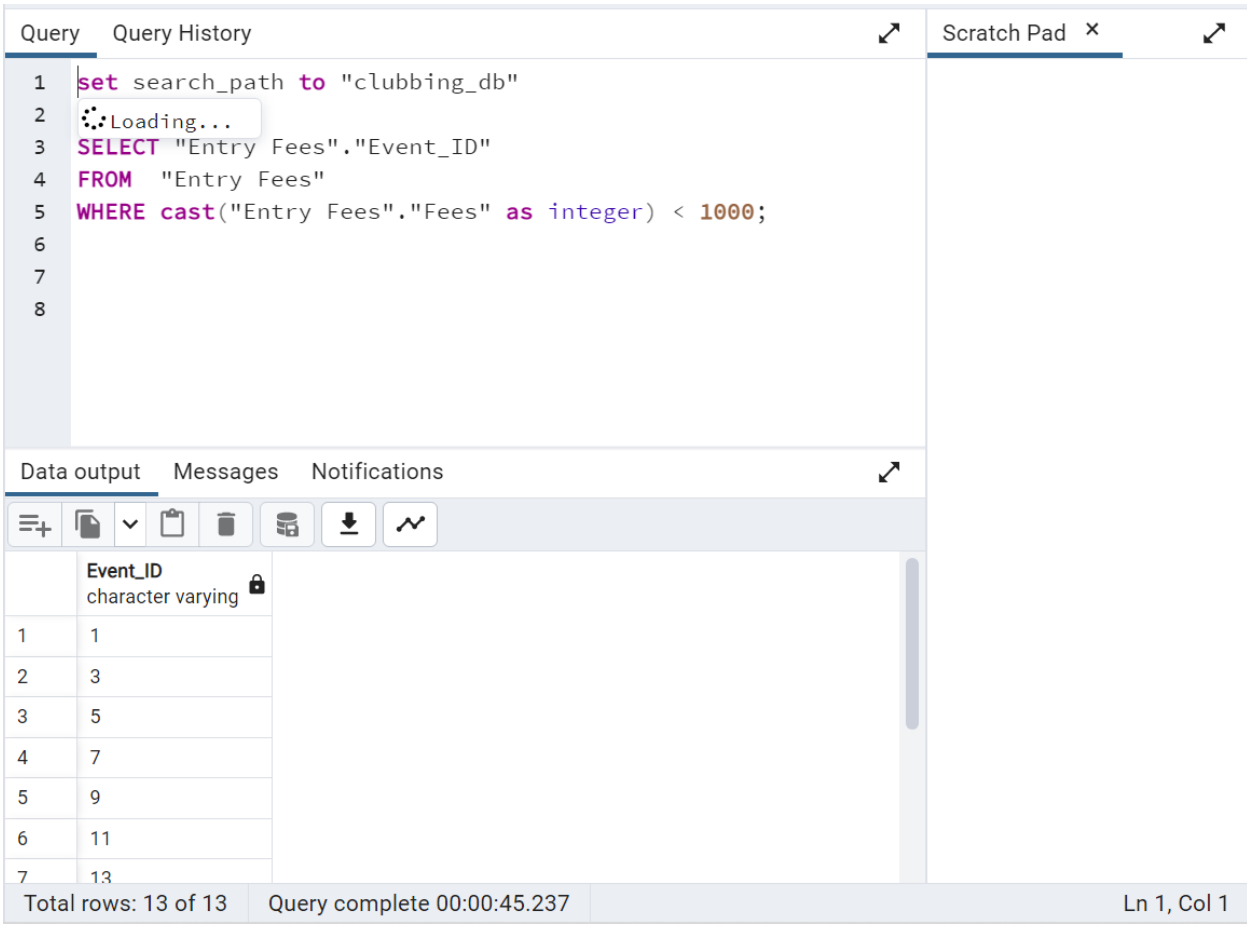
15. All costs incurred on a particular date.

```
SELECT *
FROM "Costs_bears"
WHERE "Costs_bears"."Date" = '31-01-1932';
```



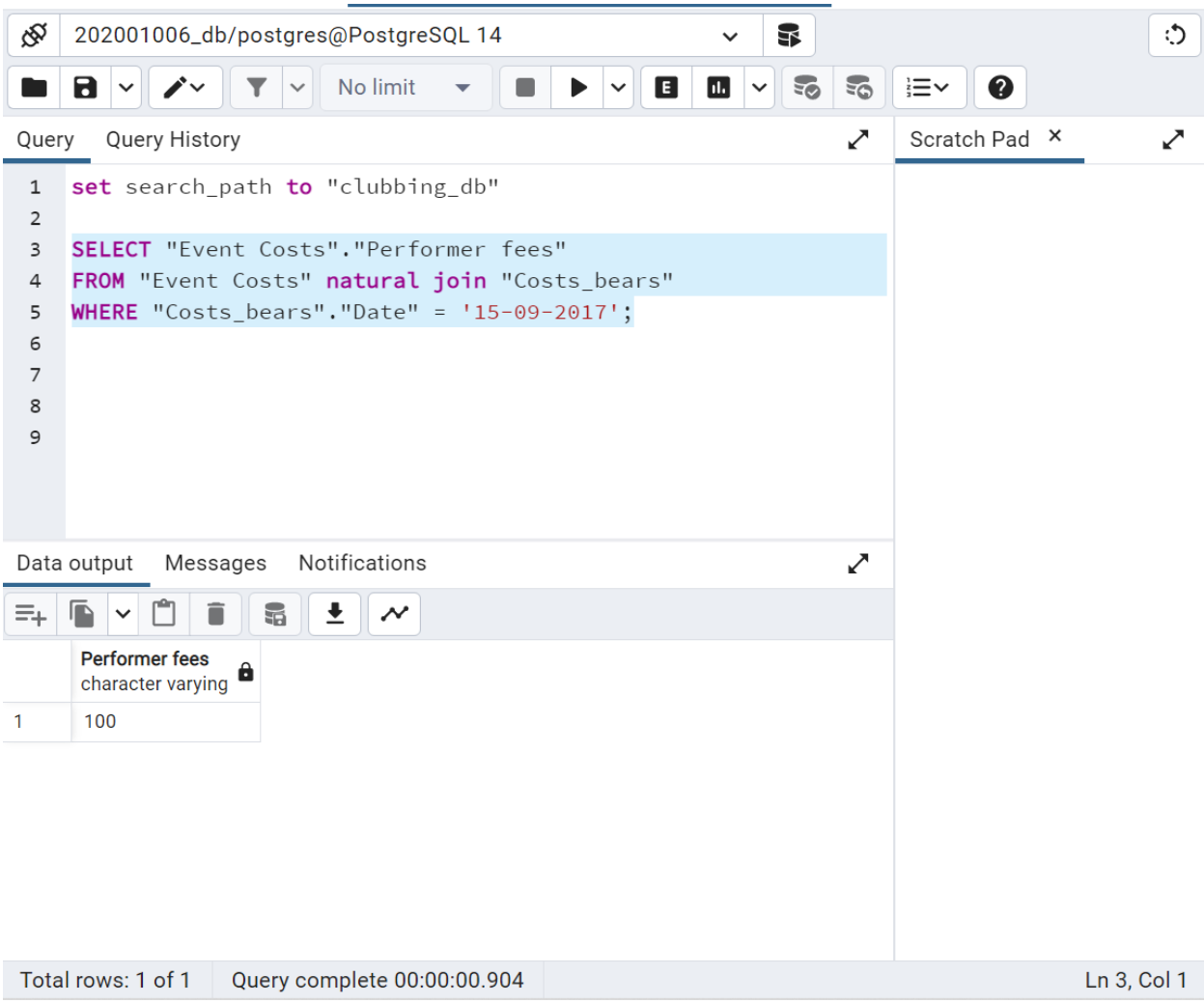
16. © All events with entry fees less than k

```
SELECT "Entry Fees"."Event_ID"
FROM  "Entry Fees"
WHERE cast("Entry Fees"."Fees" as integer) < 1000;
```



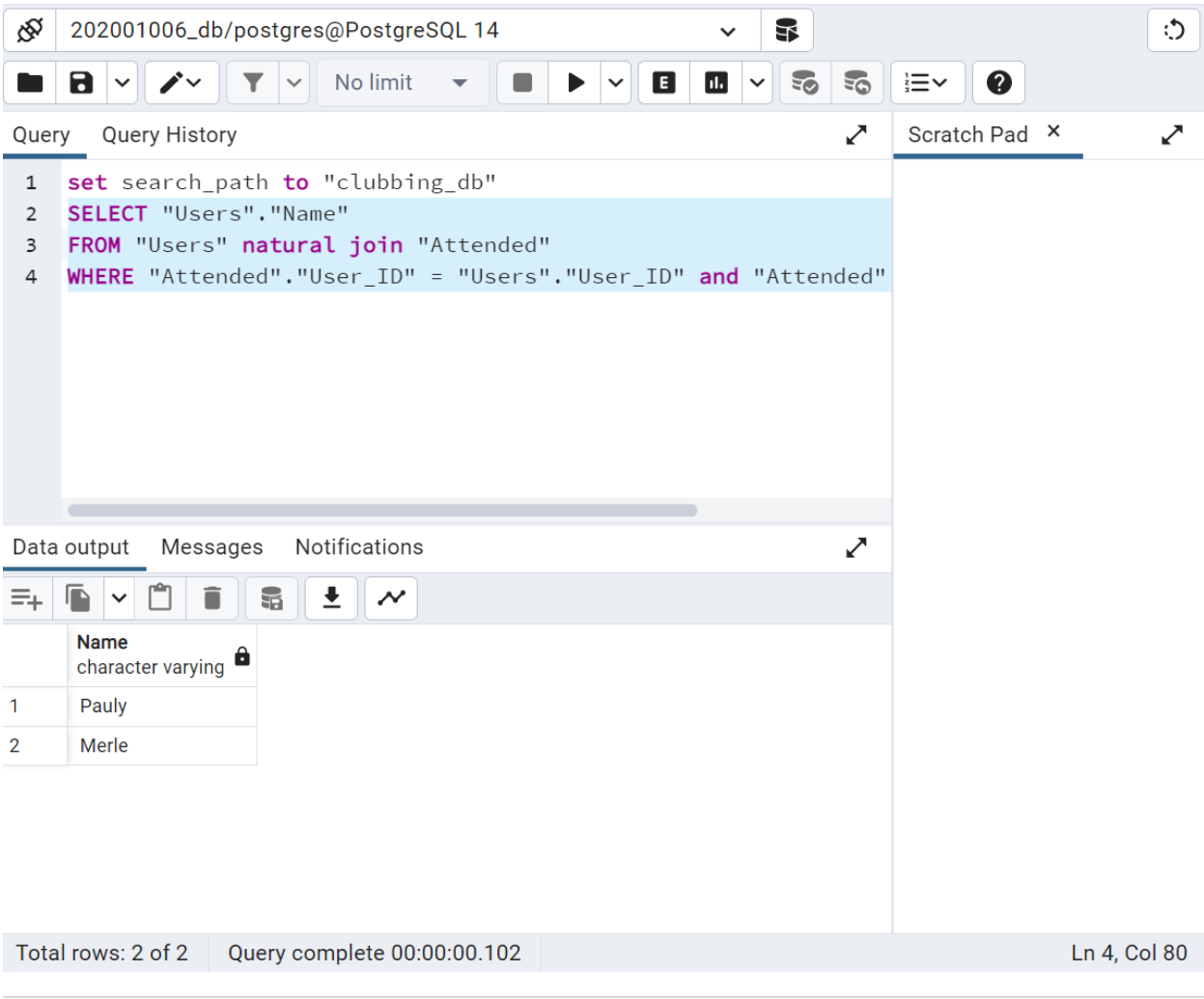
17. © Performer fees on a given date

```
SELECT "Event Costs"."Performer fees"
FROM "Event Costs" natural join "Costs_bears"
WHERE "Costs_bears"."Date" = '15-09-2017';
'
```



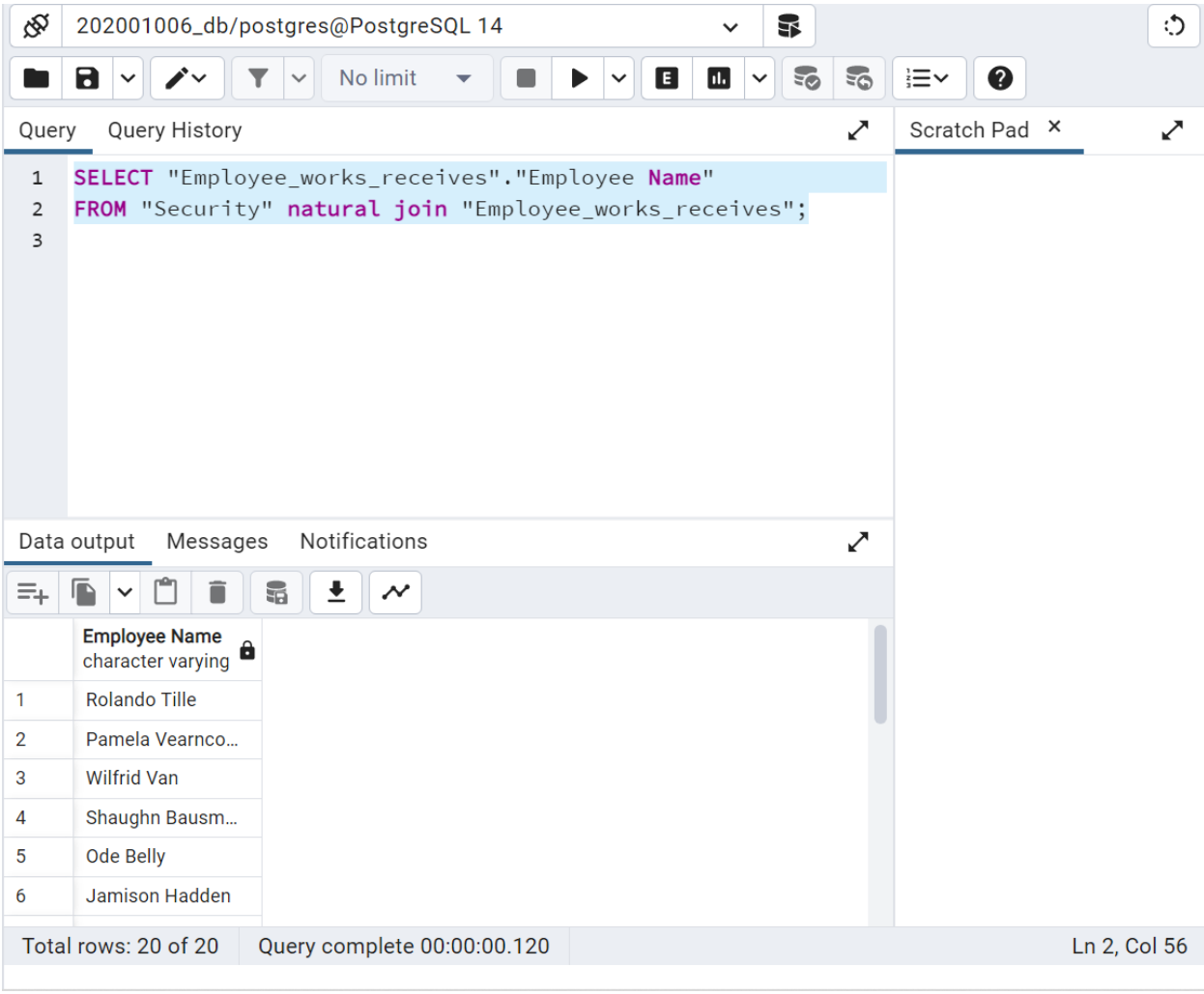
18. © Name of all guests in a particular event

```
SELECT "Users"."Name"
FROM "Users" natural join "Attended"
WHERE "Attended"."User_ID" = "Users"."User_ID" and "Attended"."Event_ID" =
'1';
```



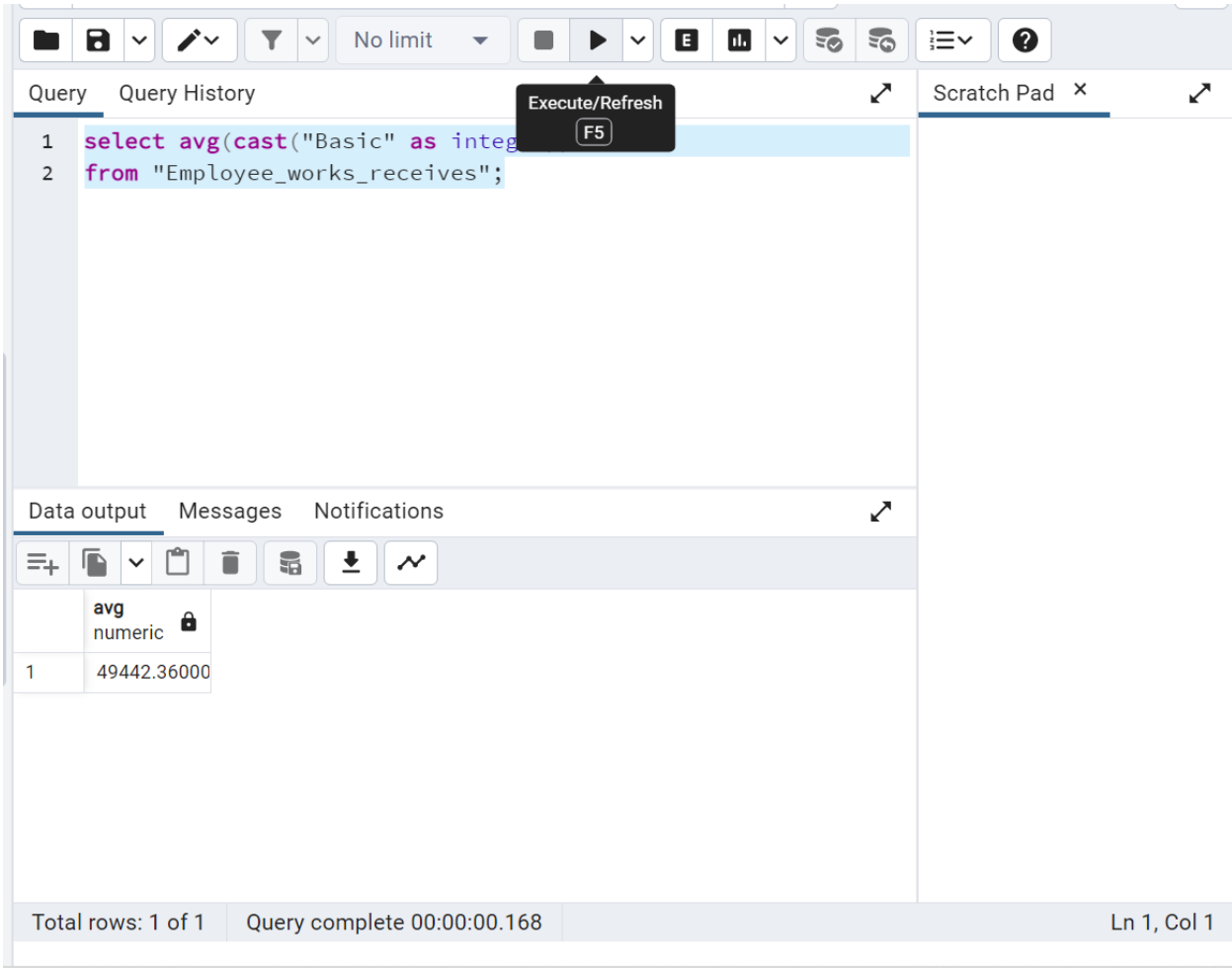
19. © Names of all Security staff

```
SELECT "Employee_works_receives"."Employee Name"
FROM "Security" natural join "Employee_works_receives";
```



20. © average basic given to employee

```
select avg(cast("Basic" as integer))
from "Employee_works_receives";
```



21. © Trigger: Minimum Salary should be 5,000

Trigger function:

```
create or replace function trigger_function_1()
returns TRIGGER
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE va character varying;
BEGIN
	if cast(new."Basic" as integer) < 100 then
		raise notice 'Basic less than 100 not allowed';
		return NULL;
	else
		return new;
	end if;
END;
$BODY$;
```

Trigger:

CREATE TRIGGER "Trigger_insert"
before INSERT
ON "clubbing_db"."Salary"
FOR EACH ROW
EXECUTE PROCEDURE "clubbing_db".trigger_function_1();