# IT-214

# Hackathon Management System

## Team ID: 5.11

**202001238 Aayush Brahmbhatt**
**202001240 Virat Chaudhari**

# Index

# Software Requirements Specification

# Table of contents

# 1.0 Introduction

## 1.1 Purpose

The purpose of this document is to present a detailed description of the Hackathon Management System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the restrictions under which it must operate, and the system's response to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the higher authorities for its approval.

## 1.2.1 Intended Audience

While the software requirement specification (SRS) document is written for a more general audience, this document is intended for individuals directly involved in developing the Hackathon Management System and event organizers. This includes software developers, company representatives, and event managers. This document need not be read sequentially; users are encouraged to jump to any section they find relevant.

## 1.2.2 Reading Suggestions

★ Part 1 (Introduction)

This section offers a summary of the Hackathon Management project, including goals and objectives, project scope, general system details, and some significant constraints associated with the intended platform.

★ Part 2 (Description)

The documentation describes what the software is doing, how it is expected to perform to external stimuli, and how the database is managed and integrity is maintained. This section also outlines the use cases for each system user separately.

★ Part 3 (Background Readings and References)

This section describes background information gathered about hackathons. Websites, books, and articles used are referenced in this section.

★ Part 4 (Interviews and Gathered Requirements)

The requirements of the company/client for the hackathon management system are collected from interviews. This section contains the information gathered from the interviews.

★ Part 5 (Surveys)

Conduction of various surveys and kind of information asked in surveys, multiple survey results, and essential points from surveys are presented in this section. The device will mediate between two parties (organizers and participants).

★ Part 6 (Observations)

Problems faced by various hackathon organizers and their suggestions about the functionalities of the Hackathon Management system are written in this section.

## 1.3 Scope of the Project

The system will be a management system for organizers of hackathons. The system will maximize the organizers' productivity by providing them with various tools to manage participants and rounds. By maximizing the organizer's work efficiency and production, the system will meet the organizer's needs while remaining easy to understand and use.

More specifically, the system allows organizers to manage and maintain the data while maintaining integrity constraints. The software facilitates communication between authorities and the participants. The database of participants is updated after every round. In the final round of the hackathon ( 24 hours of coding), transportation journey details and the contest venue are maintained. The system also contains a relational database that will ease the hackathon process.

## 1.4 References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software*

*Requirements Specifications*. IEEE Computer Society, 1998.0

## 1.5 Description

The different organizers will be able to announce numerous hackathon events on this site and reach their target audience. The hackathon's data will be stored in the system so that users can apply to contests according to their interests. The participants will register for the hackathons in which they are interested, and the organizers will use the information provided. The main objective is to streamline communication between businesses and developers. This platform will allow the different organizers to announce various Hackathon events to reach their intended participants.

**Functionalities:**

**Sign up:** The new users (Organizer or Participants) can fill in their asked information which will be stored in the system's database.

**Login:** The existing users(Organizer, Participants, or Judge) can enter their work environment after the information is entered into the login section.

# 2.0 Document the Requirements Collection/ Fact-Finding phase

## 2.1 Reading and Description:

- A hackathon (also known as a hack day, hackfest, datathon, or codefest; a portmanteau of the hacking marathon) is a sprint-like design event wherein computer programmers and others involved in software development, including graphic designers, interface designers, product managers, project managers, domain experts, and others collaborate intensively on software projects.

- The goal of a hackathon is to create functioning software or hardware by the end of the event. Hackathons tend to have a specific focus, which can include the programming language used, the operating system, an application, an API, or the subject and the demographic group of the programmers. In other cases, there is no restriction on the type of software being created or the design of the new system.
- **Common structure of hackathon:-**
    - Introductions (meet-and-greet).
    - Overview of the event (organizers explain hackathon rules and regulations and expectations).
    - Project pitches (participants can pitch ideas and form teams).
    - Hacking (collaboration on the project in a team format).
    - Presenting a finished product or unfinished work (this happens more often due to time constraints).
    - The jury decides who the winners are and hands out prizes.

- There are several types of hackathons
    1. Internal
    2. External
    3. Coding
    4. Industry

- Examples
  1. Google – Internal Hackathon
  2. Music Hack Day – Industry Hackathon
  3. HackUMass – External Hackathon
  4. TechCrunch Disrupt – Online Coding Hackathon

- Over the years hackathons have come under severe criticism, with multiple observers questioning the adequacy of hackathons to deliver impactful technological solutions. A major part of the reserve towards hackathons has to do with the lack of viability and sustainability of solutions they develop, as clearly shown by recent empirical research. Hackathons have been equally criticized for their failure to contemplate the complexity of issues that they seek to solve, developing technologies that do not address underlying societal and political causes of a problem.

**References**:-

1. Image link
2. https://en.wikipedia.org/wiki/Hackathon
3. https://www.quora.com/What-is-a-hackathon-What-do-you-do-in-it-Is-it-a-te am-event-If-yes-what-are-team-sizes
4. https://tips.hackathon.com/article/what-is-a-hackathon

### Description(HackerEarth):

- An organization must register on the website before it may host a hackathon.
- The hackathon's criteria, registration costs, prize money, and other details are all available to participants.
- Suppose the hackathon is now taking place or is upcoming. In that case, participants can sign up based on their interests. Participants can register after paying registration fees if they meet the eligibility requirements and agree to all the terms (if any).
- The organizers must choose the hackathon's subject, prize fund, mode, criteria, sponsors, and other details.
- To meet the hackathon's requirements, participants must submit their work throughout each phase (e.g., Powerpoint, Video presentation, Code File, Output Snippets, etc.).
- The hackathon's results are available to students on the website.
- Moreover, they will gain knowledge after planning a hackathon.
- An organization must register on the website before it may host a hackathon.
- The hackathon's criteria, registration costs, prize money, and other details are all available to participants.
- If the hackathon is now taking place or is upcoming, participants can sign up based on their interests. Participants can register after paying registration fees if they meet the eligibility requirements and agree to all the terms (if any).

The organizers must choose the hackathon's subject, prize fund, mode, criteria, sponsors, and other details.

To meet the hackathon's requirements, participants must submit their work throughout each phase (e.g., Powerpoint, Video presentation, Code File, Output Snippets, etc.).

## Background Reading:

- Designing the system's structure and operations should ensure that the system operates effectively.
- All the data regarding athletes, sports, events, leaderboards, etc., must be updated and maintained using a database management system.
- To ensure that a particular user can only access the relevant data and functionalities, distinct user interfaces are needed for the various user Entities (Organizers, Participants, and Judges).

The hackathon's results are available to students on the website.

Moreover, they will gain knowledge after planning a hackathon.

**Reference:** *https://link.springer.com/content/pdf/10.1007/978-3-030-58839-7.pdf*

*https://medium.com/earlybyte/what-is-a-hackathon-841a240e734a#:~:text=Hackathon%20is%20a%20word%20creation,1%E2%80%933%20days%20lasting%20event.*

## 2.2 Hackathon Management System: Interview Plan

### Interviewer:

**Aayush Brahmbhatt** (Co-Founder of Hackathon Management System)

**Virat Chaudhari** (Co-Founder Hackathon Management System)

### Invited Participants:

**Jeffery Bezos** (Sponsor 1: CEO Amazon Inc.)

**Billie Gates** (Sponsor 2: CEO Macro stiff Pvt. LTD.)

**Lionel Ronaldo** (Head - Hackathon Organization Committee DAIICT)

**Virat Sharma** (Vice Head - Hackathon Organization Committee DAIICT)

**Parth Malhotra** (Judge of the Hackathon & Professor DAIICT)

**Mukesh Adani** (Judge of the Hackathon & Managing Director Jio)

**Jay Ambani** (Convenor - Management Committee)

**Mark Juberburg** (Deputy Convenor - Management Committee)

**Date**: 9/30/2022 - 10/04/2022 **Time**: 4:00PM
**Venue**: HoR Men B102
**Duration**: 90 Minutes

### ➔ Purpose of Interview:

Discuss the organizer's and the sponsor's essential requirements from the hackathon management system and the timeline of the hackathon.

# ⋆ Participants:

## • Interviewee:

**Kiran Joshi** (Attended several Hackathons - Student at DAIICT)

**Mukesh Pandya** (Attended several Hackathons - Student at Stanford University)

## • Interviewer:

**Aayush Brahmbhatt** (Co-Founder of Hackathon Management System)

**Virat Chaudhari** (Co-Founder Hackathon Management System)

## ➔ Purpose of Interview:

Preliminary meeting to identify problems faced by participants during the hackathon and their suggestions.

**Date**: 9/29/2022 **Time**: 16:30

**Duration**: 90 minutes **Place**: HoR Men B102

## Interview Agenda / Questions:

1) Why are you drawn to participating in the hackathon?
2) How frequently do you attend hackathons?
3) Which mode is your favorite? Either online or off.
4) What issues do you run across during the process?

**Summary of Results and Requirements:**

- Current submission status in different phases.
- There should be accessible communication between the participants and the organizers.
- Comments regarding the judge's judgment (feedback), so one can enhance their performance.
- Details about the hackathons, events, and winners of various events.
- Project details of previous top submissions.
- Efficient, optimal, and reliable services.

## ★ Judge:

- **Interviewee:**

**Parth Malhotra** (Judge of the Hackathon & Professor DAIICT)

**Mukesh Adani** (Judge of the Hackathon & Managing Director Jio)

- **Interviewer:**

**Aayush Brahmbhatt** (Co-Founder Hackathon Management System)

**Virat Chaudhari** (Co-Founder Hackathon Management System)]

### ➔ Purpose of Interview:

Preliminary meeting to identify the judging criteria and requirements of the hackathon judges.

**Date:** 9/30/2022 **Time:** 16:30

**Duration:** 90 minutes **Place:** HoR Men B102

### Interview Agenda / Questions:

1) What commonalities do you notice in the participant submissions?
2) What characteristics do you observe in the submissions of participants?
3) How are the submissions classified?
4) What standards will you use to make your decisions?
5) Do you have any recommendations for us that would assist you in announcing the results?

**Summary of Results and Requirements:**

- Easy to access the submissions.
- Plagiarized work should be directly disqualified.
- The incomplete work and the complete work should be distinguished under provided constraints to make the judgment process faster

## ★ Sponsor:

### • Interviewee:

**Jeffery Bezos** (Sponsor 1: CEO Amazon Inc.)

**Billie Gates** (Sponsor 2: CEO Macro stiff Pvt. LTD.)

### • Interviewer:

**Aayush Brahmbhatt** (Co-Founder Hackathon Management System)

**Virat Chaudhari** (Co-Founder Hackathon Management System)

**Date:** 10/01/2022 **Time:** 16:30

**Duration:** 90 minutes **Place:** HoR Men B102

### ➔ Purpose of Interview:

Preliminary meeting to identify expectations and their expectations of the hackathon by the sponsors of the hackathon.

### Interview Agenda / Questions:

1) What is the hackathon's theme?
2) How can you tell which hackathons are profitable?
3) What is the main reason behind your sponsorship of the hackathon?
4) What is the most essential information you require?
5) What services, awards, or gifts do you sponsor?
6) What do you anticipate?

**Summary of Results and Requirements:**

- Sponsors get some requests to sponsor travel expenses for participants during the offline events up to some extent.
- The whole process should be transparent to the sponsors.
- The system should advertise the sponsors well enough to get recognition among the participants.

## ★ Organizer (Company, Government, Committee):

- **Interviewee:**

**Lionel Ronaldo** (Head - Hackathon Organization Committee DAIICT)

**Virat Sharma** (Vice Head - Hackathon Organization Committee DAIICT)

- **Interviewer:**

**Aayush Brahmbhatt** (Co-Founder Hackathon Management System)

**Virat Chaudhari** (Co-Founder Hackathon Management System)

### ➔ Purpose of Interview:

Preliminary meeting to identify expectations, expectations, and challenges faced while organizing the hackathon by the organization committee of the hackathon.

**Date:** 10/02/2022 **Time:** 16:30

**Duration:** 90 minutes **Place:** HoR Men B102

### Interview Agenda / Questions:

1) How many people would you anticipate attending?
2) What are the most
3) What kinds of platforms have you used?
4) What do you most anticipate from such systems?
5) What kinds of issues have you encountered thus far?
6) What are the participants' general eligibility requirements?
7) What additional features are you open to adding to your system?

**Summary of Results and Requirements:**

- ➢ Efficient, optimal, and reliable services.
- ➢ Security, consistency, and integrity of the data stored in the database.
- ➢ Analysis of the social and economic impact on the cities or countries hosting the Olympics events.
- ➢ Details about the viewership so that they can negotiate for better sponsorship and broadcasting rights.
- ➢ Efficient statistical analysis of the performance of each nation in order to get an idea about how favorable it could be for the nation hosting the Olympics over other nations.

## ★ Management team (Volunteers and Coordinators)

- **Interviewee:**

**Jay Ambani** (Convenor - Management Committee)

**Mark Juberburg** (Deputy Convenor - Management Committee)

- **Interviewer:**

**Aayush Brahmbhatt** (Co-Founder Hackathon Management System)

**Virat Chaudhari** (Co-Founder Hackathon Management System)

### ➔ Purpose of Interview:

Preliminary meeting to identify issues faced by the management team and their suggestions related to the hackathon.

**Date:** 10/04/2022 **Time:** 16:30

**Duration:** 90 minutes **Place:** HoR Men B102

### Interview Agenda / Questions:

1) What kind of information would you require about participants?
2) Which mode would be more suitable for hackathon management, online or offline?
3) What kind of behavior would you expect from participants?
4) How do you manage offline hackathons with traveling participants?
5) What were the main issues faced by you in previous hackathon management?

### Summary of Results and Requirements:

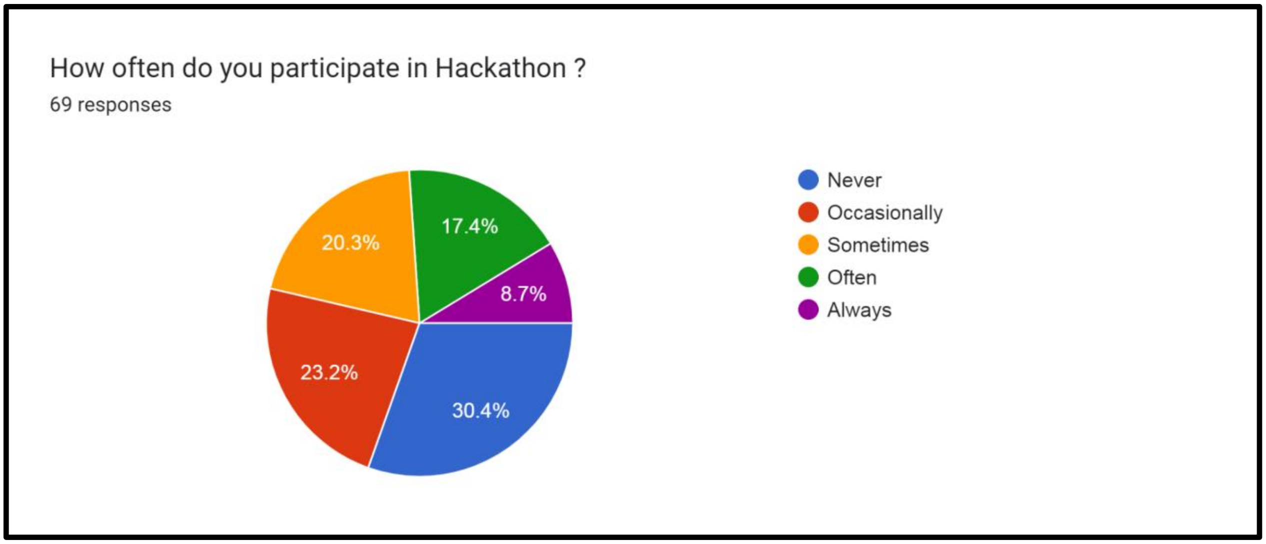
➢ All necessary information of team members to contact them.

- ➢ All the information related to the participants should be stored in well-mannered access.
- ➢ For offline hackathons user’s transport and accommodation details are required to be stored in well mannered format.

## Requirements gathered from the Interviews:

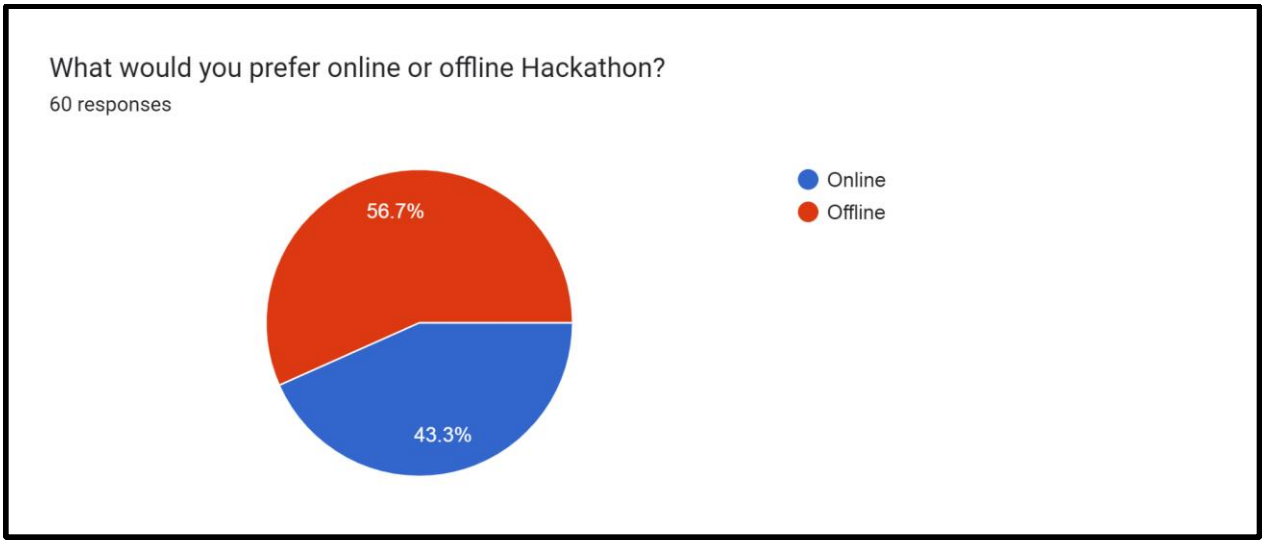
→ A well-functioning Database management system is required to maintain and update all the information about athletes, sports, events, leaderboards, etc.
→ The different user interfaces are required for the different user Entities(Organizers, Participants, Judges) so that the particular user can access only the corresponding data and functionalities.
→ System structure and functions should be designed in such a way that it ensures the efficient performance of the system.
→ The interface should be clean and without any redundant data.

## 2.3 Survey :



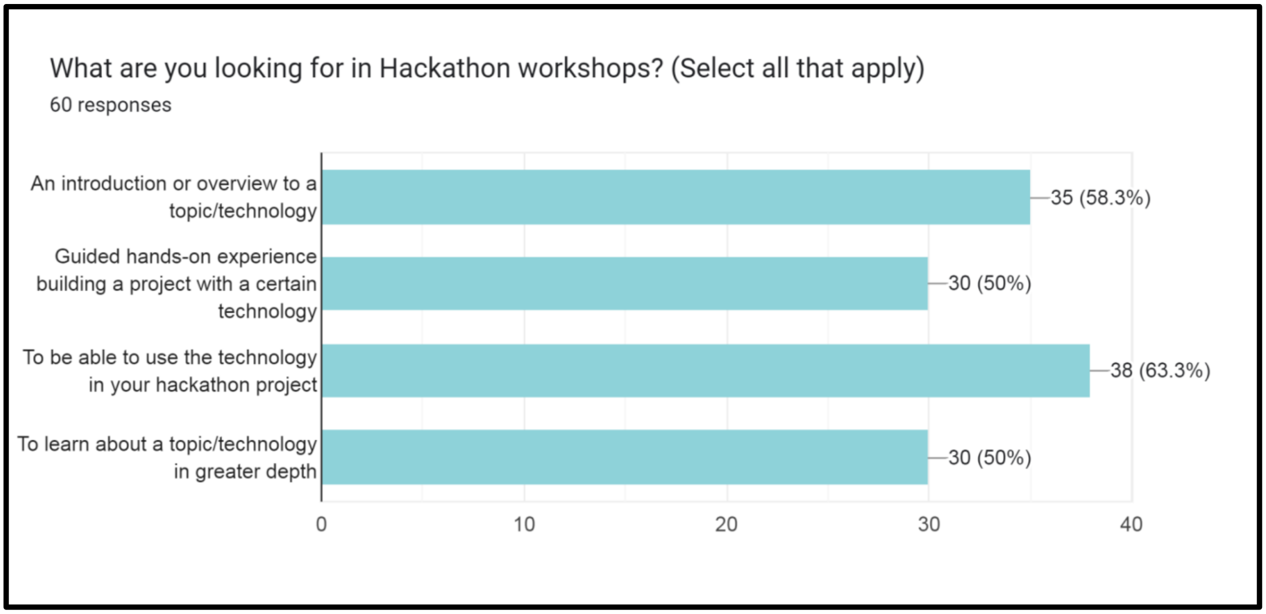
How often do you participate in Hackathon ?

69 responses



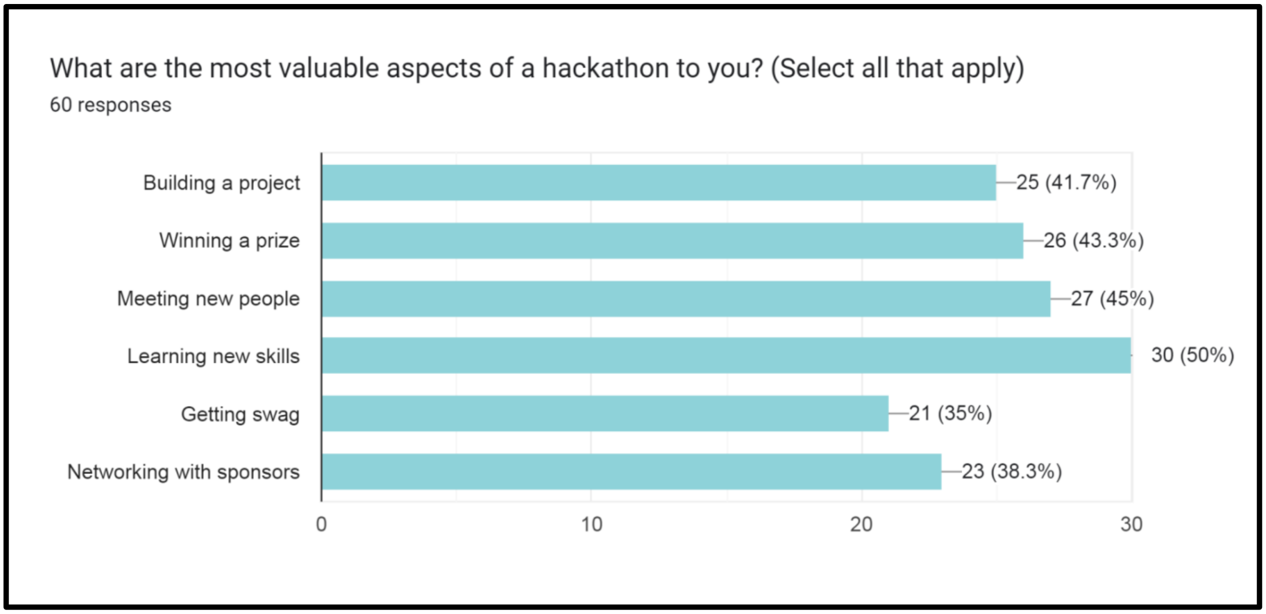
What would you prefer online or offline Hackathon?

60 responses



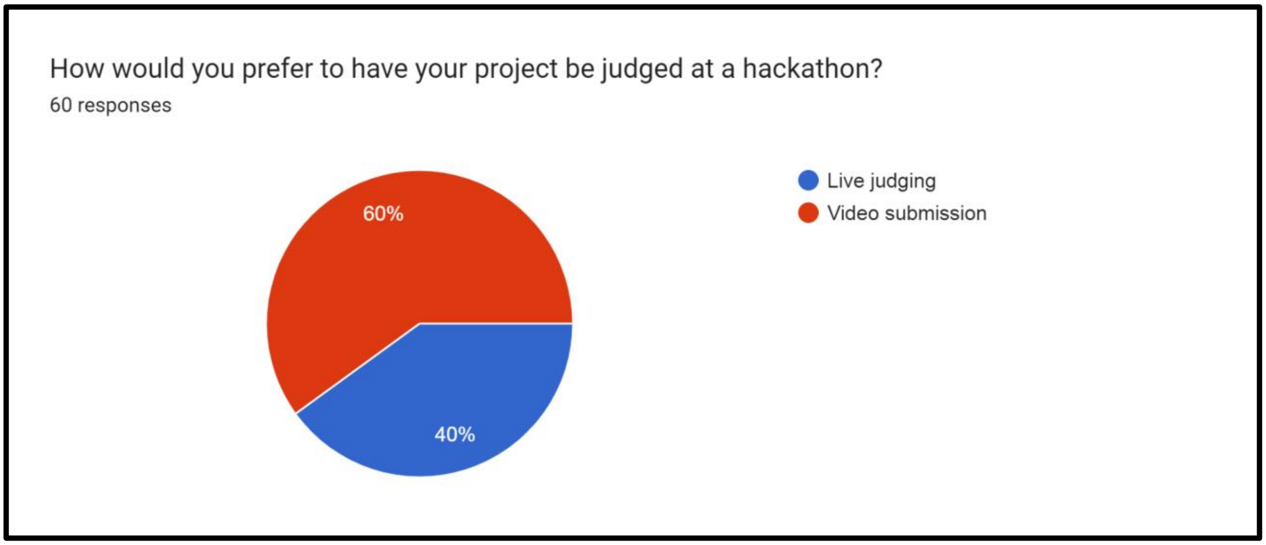
What are you looking for in Hackathon workshops? (Select all that apply)

60 responses



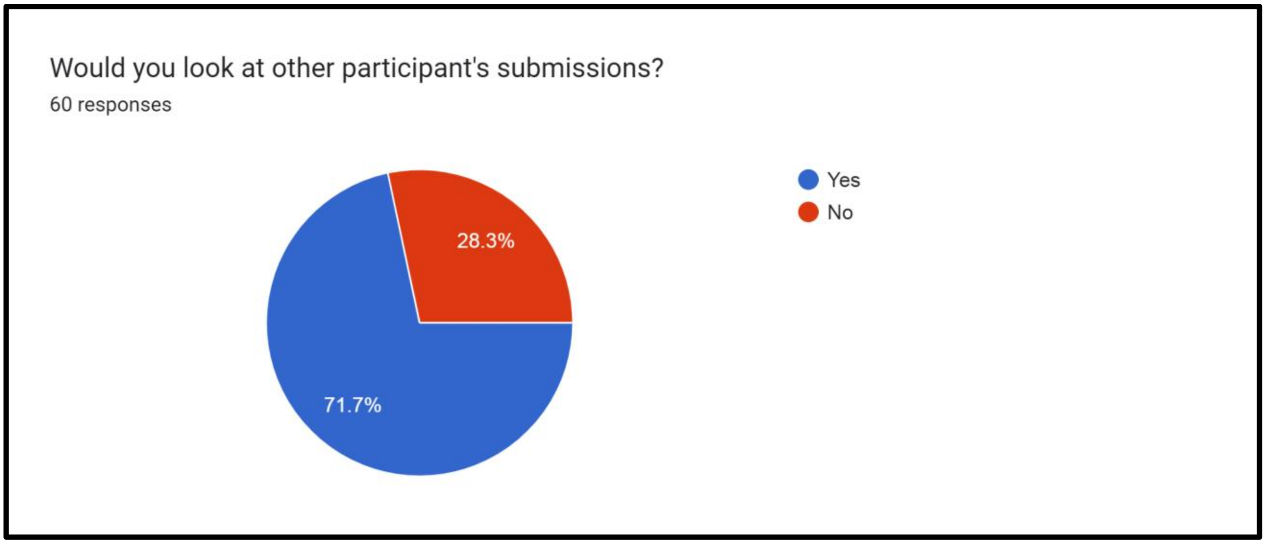
What are the most valuable aspects of a hackathon to you? (Select all that apply)

60 responses



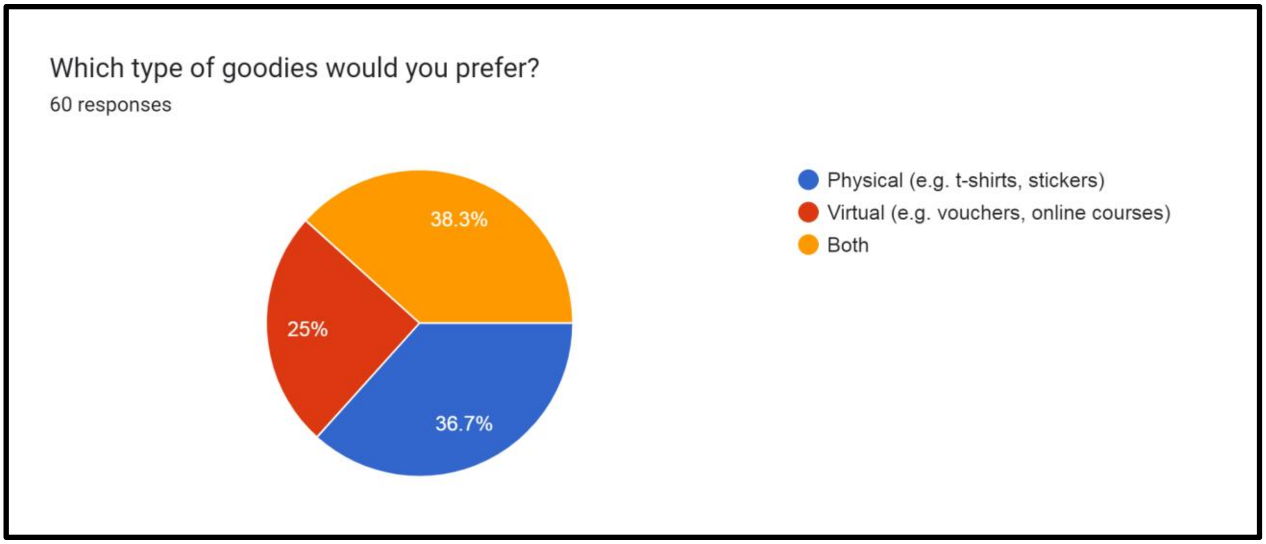
How would you prefer to have your project be judged at a hackathon?

60 responses



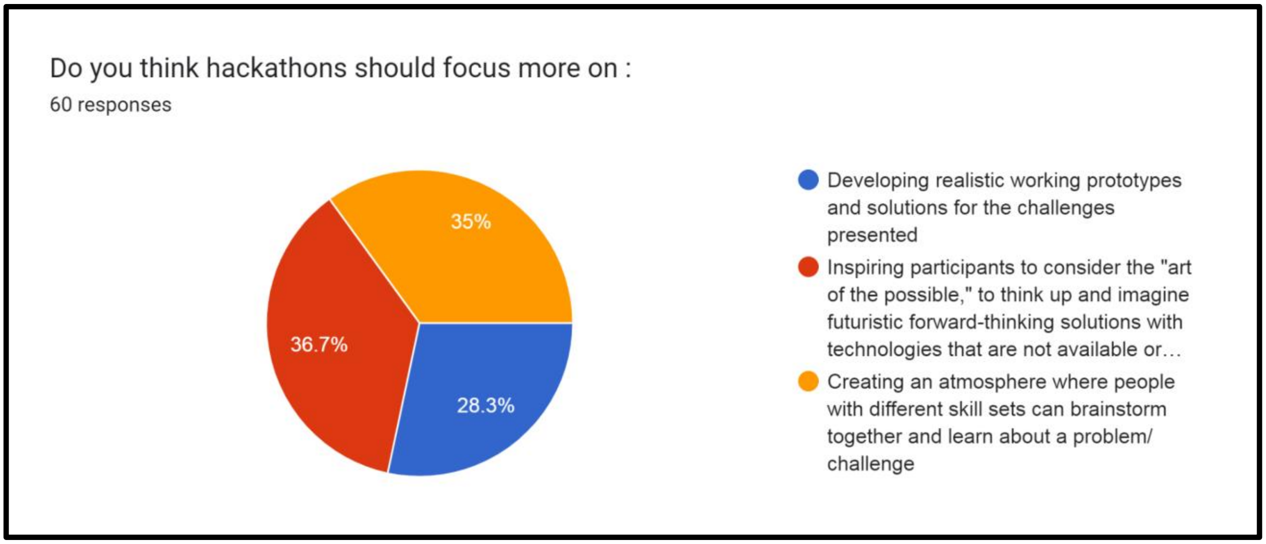
Would you look at other participant's submissions?

60 responses



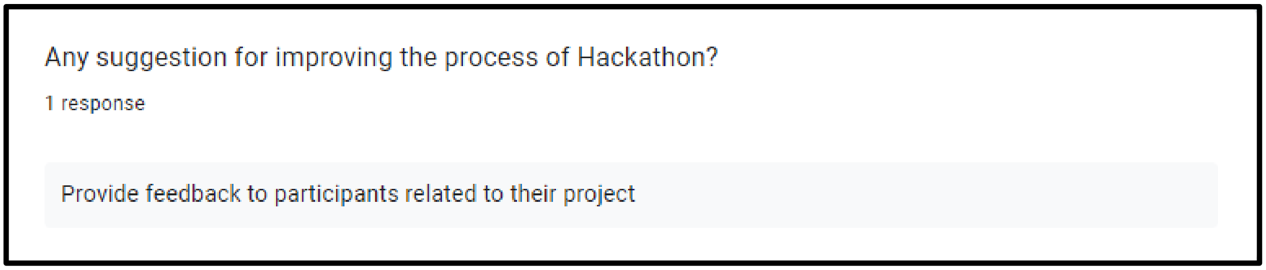
Which type of goodies would you prefer?

60 responses



Do you think hackathons should focus more on :

60 responses



Any suggestion for improving the process of Hackathon?

1 response

Provide feedback to participants related to their project

Observation:

- The evaluation process took a long time since some participants' proposals did not adhere to the guidelines; these entries had to be eliminated right away.
- During the hackathon, the participants couldn't get in touch with the organizers.
- Participants who wanted to improve their talents were unable to receive comments and feedback from judges regarding their submissions.
- The privacy and security of the data were prioritized, so not everyone had access to it.
- Different hackathon recommendations could be presented on each participant's home page according to their activities (past participation, profile, and search history.

- To make sure that everyone has access to the most recent information, the leaderboard was updated in real-time by the system.

## 2.4 Survey & Observation:

An overview of the variety of experiences with learning that are expressed by participants is provided using a method of descriptive statistics to present data acquired by surveying participants via online questionnaires. A total of forty-seven (60) survey responses, including the results of the pilot study for the modified questionnaire, were gathered.

A pilot study based on a reviewed questionnaire will be carried out for these software requirement requirements. The primary goal of the pilot project is to determine how much the survey instrument can be utilized to investigate issues and improve the prerequisites for the hackathon process. This pilot research is a crucial step to improve the validity, effectiveness, and transparency of the features that are going to be put into action. A real-time hackathon management system may be implemented by using the survey to gather key data for analysis.

# 3.0 Fact Finding Chart:

| Objective | Technique | Subject(s) | Time Commitment |
|---|---|---|---|
| To get the background of the Hackathon Management System and format of the SRS | Background Reading | Website, E-Book | 1 Day |
| To find out the roles of organizer | Interview | Hackout Organizers | 2 hour |
| To gain an understanding of the roles of Sponsors | Interview | Organizers | 90 mins |
| To gain an understanding of the roles of Judge | Interview | Judge | 30 mins |
| To gain an understanding of the real world Hackathon Management database | Background Readings, Observation | Hackerearth Website | 1 hour |

| | | | |
|---|---|---|---|
| To determine the difficulty faced by the participant's | Questionnaire | Hackout User | 2 hour |

# 4.0 Requirements

- A properly working database must update and maintain all the stored data's related information.
- The various user interfaces are necessary for the other user entities (Organizers, Participants, Judges) so that the specific user may access just the related data and capabilities. The database should be consistent and free of any redundant data.
- The dashboard exhibiting recent and prior occurrences should be visible to the average visitor.
- The organizers should find it simple to sign up for their next hackathons.
- All participant information should be accessible to the organizers.
- The participants must be able to view the pertinent specifics of recent and prior occurrences.
- The status of the participants' contributions needs to be visible to them.
- Participants should be able to view the judges' critiques and remarks on their entries.
- After the conclusion of the hackathon, the participants should be able to provide comments and ideas.
- The hackathon organizer and participants should be able to communicate to address any issues and easily share any problems.
- Judges ought to have access to every bit of participant data. (As well as submissions)
- Participants should be able to view leaderboards and other people's performances.
- The user interface needs to be tidy and devoid of extraneous information
- continuously updating the data in the database system.

★ Real-time integrity maintenance is required for many system components.
★ Real-time updates and integrity maintenance should be performed on the database. Additionally, a backup should be offered for inconsistency/crashes.
★ System structure and functions should be designed in such a way that it ensures the fast performance of the system.
★ There should be FAQs and a discussion section.

# 5.0 User Categories and Privileges

**List of user categories and their roles:**

★ **Participants:**

The participants' job is to look for upcoming hackathons and find ones that interest them.

★ **Organizers:**

The organizers' job is to plan numerous Hackathon events and post all relevant information about them on the platform.

★ **Judges:**

On the first visit, all users will be regarded as regular visitors. The entire public will be able to view both past and upcoming activities. They won't have any special rights.

★ **General Visitor:**

On the first visit, all users will be regarded as regular visitors. The entire public will be able to view both past and upcoming activities. They won't have any special rights.

★ **Database system manager:**

The manager of the database system has three key responsibilities: managing traffic via the database system, preserving data consistency and integrity, and managing technical issues that users may encounter.

# List of privileges/functions that can be accessed by different user classes

## 1) General visitor:

- ➔ At the first visit, all users will be regarded as regular visitors.
- ➔ On their interface, they can view both recent and previous occurrences.
- ➔ They will be divided into **three groups**, including judges, participants, and organizers. To continue, each of these users will have the choice to Sign up or Login.
- ➔ The **Sign-up/Log-in** area allows users to choose the category that best fits their needs before continuing with the required information.

## 2) Organizer:

- ➔ They will be guided in organizing the hackathon using the organizer's user interface.
- ➔ Organizers can access their existing account by entering their password and the requested details.
- ➔ The hackathons' information, themes, and other specifics can be released by the organizers.
- ➔ They have access to all the data pertaining to the attendees of their hackathons.
- ➔ They have the authority to **accept or reject** a candidate's application.
- ➔ The sponsors' names, contact information, and judge information can all be set.
- ➔ They will have access to the participants' communications.
- ➔ To manage the data, they can search, update, delete, insert, and more.

## 3) Participants:

- ➔ The participant's user interface will direct them to listings of upcoming events and events that have already taken place.
- ➔ They will receive their profile, which they can edit with new information.
- ➔ Data from previous participants will be available for participants to keep.
- ➔ A search tool will be provided to participants so they can find the hackathon of their choice.
- ➔ Check out the entries from previous hackathons.
- ➔ The judges will be able to provide participants with feedback and comments on their contributions.

- During the hackathon, participants will be able to ask the organisers questions to get answers to their issues.
- Additionally, there will be a few **user-friendly general functionalities**.

## 4) Judges:

- The judges' User Interface will direct them to the hackathon submissions that are relevant to them.
- They will have access to the participants' activities.
- To make the process of judging simple under the circumstances, the system will have the ability to discriminate between submissions that are valid and those that are invalid.
- The opportunity to review the participants' contributions will be given to the judges. They can provide feedback and make suitable comments about them.
- They will be able to inform the participants of the results.

.

# 6.0 Assumptions:

- It is assumed that the users of this database system own all necessary hardware and software to use this programme.
- Furthermore, it is assumed that the database's information is constantly updated.
- Users of the database system are taken to have dependable internet connections.
- Database consistency is present.
- The database constantly upholds the accuracy of the data.
- Rarely, the users will have access to alternative resources.
- The server's internet connection is stable enough to handle all user requests without crashing, and no data loss due to technical issues ever happens.

# 7.0 Business Constraints:

- The database system has a finite amount of computing capacity.
- The database's storage space is constrained.
- The database system just needs a small amount of hardware and software.
- Funding for the entire system is constrained.
- The number of users who can access the database at once is likewise restricted.

# Noun Analysis and ER diagram

# 1. Description:

Hackathon is a competition where you propose your ideas in the tech world. A Hackathon management system will behave as a middleware between company or event/hackathon organizer and participants to communicate or do some kind of operations, the path between businesses and development.

**USERS**

The hackathon management system offers features for both coordinating and participating in hackathons. It also offers tools to make the procedure simple and trouble-free.
Between the businesses that host hackathons and the participants, the hackathon management system offers a seamless link.
The system's UI/UX is accessible to all users, who may browse past and forthcoming hackathons up to a certain point. They'll have two possibilities.

**Sign-up**
**Log-in**

Using sign-up choices and entering the necessary information, new users may establish an account.

By selecting the log-in option and supplying the necessary information, including a special key attribute that will set them apart from the other current users, the existing users will be able to access their work environment.

In the work environment of existing users they will be given three options to proceed further,

1. Participate Hackathon
2. Organize Hackathon
3. Evaluate Hackathon

As per the users role they will select an option.

## ORGANIZERS

- if an already-registered user decides to activate the hackathon option. He or she will be required to fill out the hackathon's information.
    - ➔ Hackathon format (offline/online)
    - ➔ Location (if offline)
    - ➔ Details about sponsors
    - ➔ Submission format
    - ➔ Time and date
    - ➔ Duration of the hackathon
    - ➔ Prizes (if any)
    - ➔ Details about judges
    - ➔ Theme of hackathon

- They will be guided in organizing the hackathon using the organizer's user interface.
- Organizers can access their existing account by entering their password and the requested details.
- The hackathons' information, topics, and other specifics can be released by the organisers.
- They have access to all the data pertaining to the attendees of their hackathons.
- They have the authority to accept or reject a candidate's application.
- The sponsors' names, contact information, and judging information can all be set.
- They will have access to the participants' communications.
- They can search, update, delete, insert, etc. to regulate the data.

## Judges

- The User Interface for the judges will navigate them toward the submissions for the corresponding hackathons.
- They will be given access to the activities of participants.
- The system will provide the functionality to distinguish between valid or invalid submissions to make the process of judgment easy under given conditions.
- Judges will be given the privilege to go through the submissions of the participants. They can make appropriate comments on them as well as give feedback.
- They will be able to announce results to the participants.

## PARTICIPANTS

If an existing user selects the option to participate in a hackathon he will be navigated to the page where he/she will be able to see upcoming events and past events.

- ❖ Upcoming events:
    - ➔ Details about the hackathon
    - ➔ Details about the sponsors
    - ➔ Details about the judges
    - ➔ Registration
        - Email id
        - Password
        - User name for hackathon
        - Team size (if any)
        - Team name (if any)
        - Details about the team

- ❖ Past events:
    - ➔ Details about the hackathon
    - ➔ Details about the sponsors
    - ➔ Details about the judges
    - ➔ Registration
        - Email id
        - Password
        - User name for hackathon
        - Team size (if any)
        - Team name (if any)
        - Details about the team

- The participant's user interface will direct them to listings of upcoming events and events that have already taken place.
- Additionally, there will be a few user-friendly general functionalities.
- They will receive their profile, which they can edit with new information.
- The judges will be able to provide participants with feedback and comments on their contributions.
- Data from previous participants will be available for participants to keep.
- A search tool will be provided to participants so they can find the hackathon of their choice.
- Check out the entries from previous hackathons.
- During the hackathon, participants will be able to ask the organizers questions to get answers to their issues.

On the basis of judging criteria, teams will be judged after each round and leaderboard will be updated. Judges will provide feedback on submissions. Also Companies hire different skilled persons and get the benefit of a hackathon.

## 2. Noun (& Verb) Analysis.

| Nouns | Verbs |
|---|---|
| You | designing (Hackathon Management system) |
| Sponsors | Sponsor the hackathon |
| Participants | Participate in hackathon |
| Judges | Evaluates the work |
| system | Provides the system |
| sponser_id | Identifies sponser_id |
| hackathon_id | Identifies hackathon_id |
| type_of_company | Identifies companies |
| name | Provides name |
| end_time | Provides end_time |
| no_of_member | Provides number of members |
| leader | Provides name of leader |
| Information | Provides data about participants |
| location | Provides location about participants |
| account | Account info users |
| password | Password of users |
| price | Available price money for winners |
| connection | Provides connections |
| companies | Companies with problem statement |
| mode | Offline / online |

| | |
|---|---|
| UI/UX | Presents User interface |
| Organizers | Manages the hackathon |
| backup | Provides Backuped data |
| feedback | Feedback from participants |
| Code | Provided Code by participants |
| FAQs | Provides frequently asked questions |
| data | Provides data of users |
| cash | Indicates cash payment |
| section | differentiates |
| conditions | restricts |
| structure | constructs |
| mode | Online / offline |
| Snippets | Pieces of code |
| Video | Video streaming |
| Duration | Defines time length |
| environment | Provides work area |
| work | Provides the work |
| Theme | Defines subject |
| Submission | Provides work |
| format | Defines format of submission |
| options | Provides choices |
| users | Uses system |
| account | Keeps user info |

| | |
|---|---|
| Details | Provides user info |
| Powerpoint | Presents final work |
| requirement | identifies constraints |
| information | Provides info |
| Log-in | Provides way to log-in |
| criteria | Provides constraints |
| options | Indicates various choices |
| demonstration | Demonstrates the work |
| queries | Inquires |
| time | Provides time |

# Selected Nouns

| Candidate entity set | Candidate attribute set | Candidate relationship set |
|---|---|---|
| team | team_id, team_name, no_member, leader, hackathon_id, submission | hackathon_management |
| sponsors | Name , sponsor id , hackathon id, amount | sponser_hackathon |
| participant | name, p_id, domain, phone_number, company_id, email_id | hackathon_company |
| hackathon | Start_time, name , end_time, hackathon id | Judge_hackathon , sponser_hackathon , hackathon_company , Hackathon_management, team_hackathon |
| judges | judge_id, name, hackathon_id | judge_hackathon |
| company | company_id, name, type_of_company | team_participant |
| management_team | team_id, team_name, no_of_member, contact_no | hackathon_management |

# Rejected Nouns

| **Noun** | **Reject Reason** |
|---|---|
| price | Irrelevant |
| connection | vague |
| options | redundancy |
| mode | duplicate |
| UI/UX | Irrelevant |
| Organizers | Irrelevant |
| backup | vague |
| feedback | redundancy |
| Code | duplicate |
| FAQs | General |
| data | redundancy |
| cash | duplicate |
| section | redundancy |
| conditions | Irrelevant |
| structure | vague |
| mode | duplicate |
| Snippets | irrelevant |
| Video | vague |
| Duration | redundancy |
| environment | redundancy |

| **Noun** | **Reject Reason** |
|---|---|
| price | Irrelevant |
| connection | vague |
| options | redundancy |
| mode | duplicate |
| UI/UX | Irrelevant |
| work | Irrelevant |
| Theme | vague |
| Submission | vague |
| format | redundancy |
| options | duplicate |
| users | redundancy |
| account | vague |
| Details | redundancy |
| Powerpoint | duplicate |
| requirement | General |
| information | redundancy |
| Log-in | vague |
| criteria | redundancy |
| options | duplicate |
| demonstration | irrelevant |
| queries | redundancy |

# ER-Diagram

## 1) Identify Entity types.

- In the relationship between hackathon and sponsors, this weak entity relation sponsor is the weak entity, and the name is the partial key.

- Hackathon is a strong entity in this weak relation, and Hack_id is identifying.

- Also, about hackathons, Judge is a weak relation, and Judge is a weak entity. Also, the name is a partial key to this weak relation.

- In the above weak relation, Hack_id is the identifying key, whereas Hackathon is the strong entity.

## 2) Identify Relationship types.

**Entity vs. Attribute:**

| **Entity** | **Attribute** |
|---|---|
| User | Age, User_Id, DOB, Password, Contact_Info, Email_Id, User_name |

| | |
|---|---|
| Organizer | Company_Id, Name |
| Team | Leader, Team_Id, Current_stat, Team_Size |
| Participant | Domain |
| Judge | Judge_id, Name |
| Submission | Time,  eval_status |
| Sponsors | Name |
| Hackathon | Hack_id,  company_id, date, start_time, end_time, duration, theme, Location |

**Binary vs. Ternary Relationships:**

All relations between all the entities are binary in the ER diagram.

Team_mates(relationship between Team and Participant)

- This is a one-to-many relationship.

Judes(relationship between Judge and Hackathon)

- This is a many-to-many relationship.

Organizes(relationship between Organizers and Hackathon)

- This is a one-to-many relationship.

Sponsors(relationship between Sponsor and Hackathon)

- This is a many-to-many relationship.

Evaluate(relationship between Judge and Submission)

- This is a one-to-many relationship.

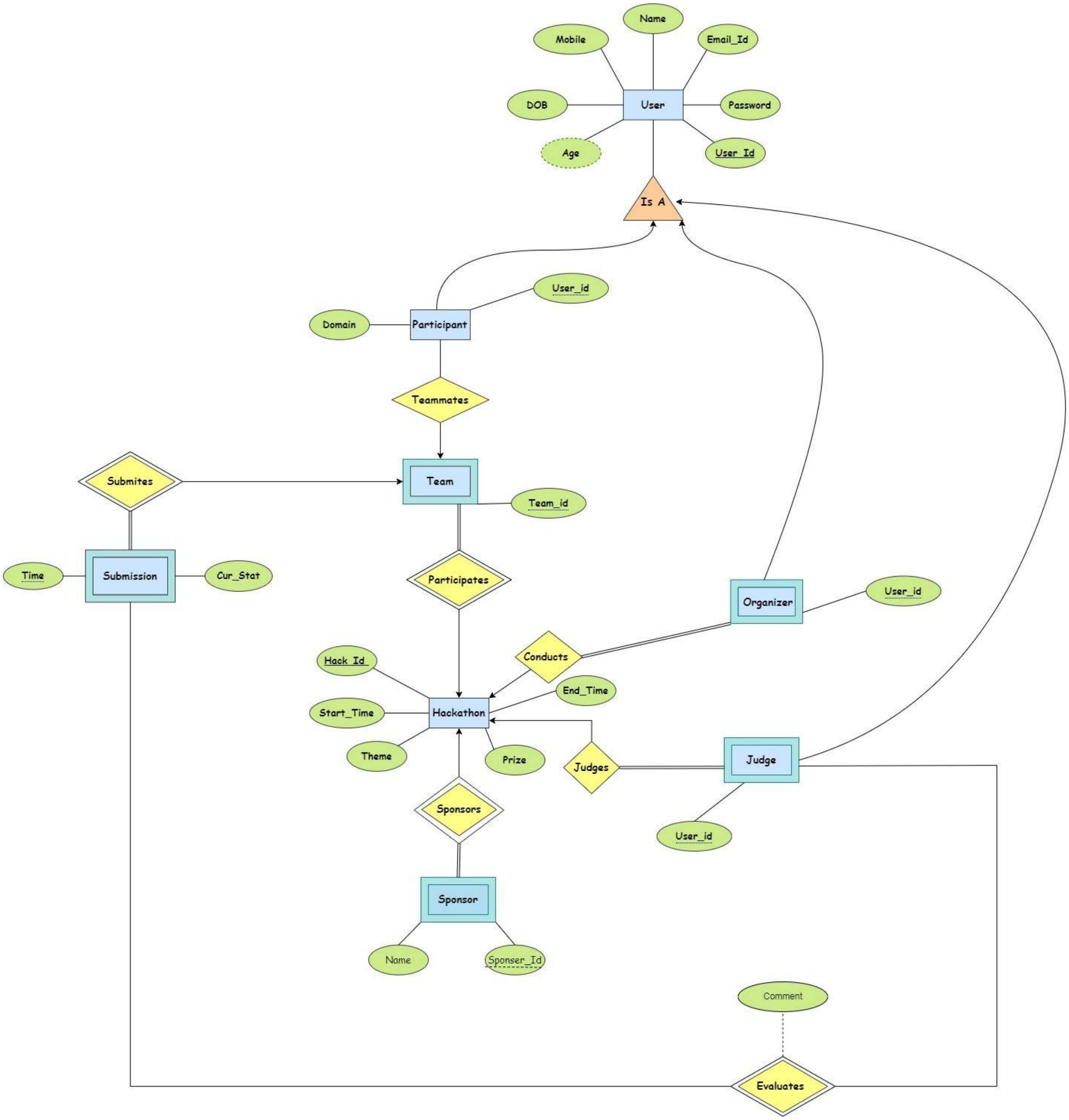
Submit(relationship between Team and submission)

- This is a one-to-many relationship.

## 3) Analyze ERD for any other missing information.

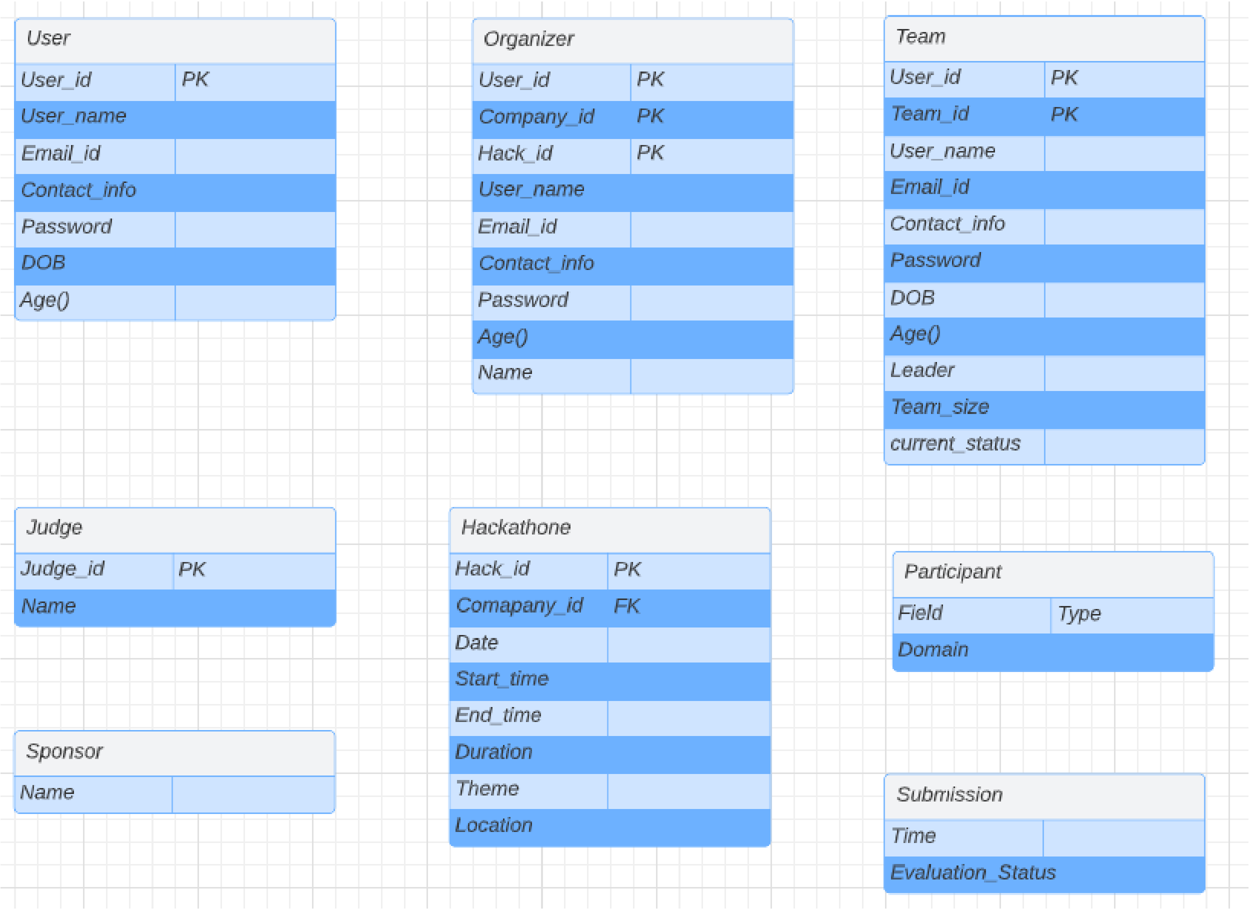
- All entities are the same as before in the latest ER diagram.
- We added a **derived** attribute(Duration).
- We have added some attributes like team_size and removed some redundant attributes like email, contact, etc.
- Instead of extra attributes, we generalized that attribute and removed redundancy.
- We added a weak entity relationship between two relations and defined partial and identifying keys for this relation.
- Also, we add ISA **specialization** from regular users to participants, organizers, judges, and sponsors.
- We make Team entity **aggregation** as all other entities are related to the Team entity.
- Similarly, the organizer entity has more company_id and Name attributes than the user entity.
- The Hackathon entity holds various information related to the ongoing hackathon in our system (i.e., hack_id, date, start time, end time, duration, etc.).
- ER diagram also includes **weak entities** such as organizer, judge, participant, submission, and sponsor.
- The team and participants combined create a separate entity about a hackathon entity. The concept used in this procedure is aggregation.

❖ **ER Diagram:**


Name
Mobile
Email_Id
DOB
User
Password
Age
User_Id
Is A
User_id
Domain
Participant
Teammates
Submites
Team
Team_id
Submission
Time
Cur_Stat
Participates
Organizer
User_id
Hack_Id
Conducts
End_Time
Start_Time
Hackathon
Theme
Prize
Judges
Judge
User_id
Sponsors
Sponsor
Name
Sponser_Id
Comment
Evaluates

# Normalization and DDL script

# Relational Model



| User | |
|---|---|
| User_id | PK |
| User_name | |
| Email_id | |
| Contact_info | |
| Password | |
| DOB | |
| Age() | |

| Organizer | |
|---|---|
| User_id | PK |
| Company_id | PK |
| Hack_id | PK |
| User_name | |
| Email_id | |
| Contact_info | |
| Password | |
| Age() | |
| Name | |

| Team | |
|---|---|
| User_id | PK |
| Team_id | PK |
| User_name | |
| Email_id | |
| Contact_info | |
| Password | |
| DOB | |
| Age() | |
| Leader | |
| Team_size | |
| current_status | |

| Judge | |
|---|---|
| Judge_id | PK |
| Name | |

| Hackathone | |
|---|---|
| Hack_id | PK |
| Comapany_id | FK |
| Date | |
| Start_time | |
| End_time | |
| Duration | |
| Theme | |
| Location | |

| Participant | |
|---|---|
| Field | Type |
| Domain | |

| Sponsor | |
|---|---|
| Name | |

| Submission | |
|---|---|
| Time | |
| Evaluation_Status | |

## i. List all the Relations &amp; Schemas with all details (Original Design of Database)

## → Schema Before Refinement:

- **User**(User_Id , User_name , Email_Id , Contact_info , Password , DOB , Age)

- **Organizer**(Hack_id , Company_id,User_Id , User_name , Email_Id , Contact_info , Password , DOB , Age, Name )

- **Team**(Hack_id , Team_Id , Team_size , Leader , Current_Status)
    - FK Team_id references to **Participant**

- **Participant**(Tean_id, User_Id , User_name , Email_Id , Contact_info , Password , DOB , Age , Domain)

- **Submission**(Hack_id, Time , Evaluation_Status)

- **Sponsor**(Hack_id , Name)

- **Judge**(Hack_id , Judge_Id ,User_I d , User_name , Email_Id , Contact_info , Password , DOB , Age, Name)

- **Hackathon**(Hack_id , Comapny_id , date , start_time , end_time , duration , theme , location)
    - FK Company_id references to **Organizer**

- **User:**
  - **User_id** → User_id, User_name , Email_Id , Contact_info , Password , DOB , Age
  - **DOB** → Age

- **Organizer**
  - **Hack_id , Company_id,User_Id** → User_name , Email_Id , Contact_info , Password , DOB , Age, Name
  - **DOB** → Age

- **Team**
  - **Hack_id , Team_Id** → Team_size , Leader , Current_Status
  - FK Team_id references to **Participant**

- **Participant**
  - **Team_id, User_Id** → User_name , Email_Id , Contact_info , Password , DOB , Age , Domain
  - **DOB** → Age

- **Submission**
  - **Hack_id, Team_id, Time** → Evaluation_Status

- **Sponsor**
  - **Hack_id , Name ->** Hack_id , Name

- **Hackathon**
  - **Hack_id** → Comapny_id , date , start_time , end_time , duration , theme , location
  - FK Company_id references to **Organizer**

- **Judge**

- ➢ **Hack_id , Judge_Id ,User_Id** → User_name , Email_Id , Contact_info , Password , DOB , Age, Name
- ➢ **DOB** → Age

**ii.Dependencies and Normal forms**

**1. Hackathon** (Hack_id , Comapny_id , date , start_time , end_time , duration , theme , location)

**PK dependency:**
**Hack_id** → Comapny_id , date , start_time , end_time , duration , theme , location

**Functional Dependencies:**

Hack_id→ Hack_id
Hack_id→ Comapny_id
Hack_id→ date
Hack_id→ start_time
Hack_id→ end_time
Hack_id→ duration
Hack_id→ theme
Hack_id→ location

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.
For a relation to be in BCNF,
a. It should be in the third normal form (3NF).

**2. Organizer** (Hack_id , Company_id, User_Id , User_name , Email_Id , Contact_info , Password , DOB , Age, Name)

**PK dependency:**

Hack_id, Company_id, User_Id → User_name, Email_Id, Contact_info , Password , DOB , Age, Name

**Functional Dependencies:**

User_id→ User_name
User_id→ Email_id
User_id→ Contact_info
User_id→ password
User_id→ DOB
User_id→ Age
User_id→ User_id
DOB→ Age

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.
For a relation to be in BCNF,

a. It should be in the third normal form (3NF).

### 3. Participants(Team_id, User_Id, User_name , Email_Id , Contact_info , Password , DOB , Age , Domain)

**PK dependency:**

Team_id,User_Id → User_name, Email_Id, Contact_info , Password , DOB , Age, Name

**Functional Dependencies:**

User_id→ User_name
User_id→ Email_id
User_id→ Contact_info
User_id→ password
User_id→ DOB
User_id→ Age
User_id→ User_id
DOB→ Age

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.
For a relation to be in BCNF,
a. It should be in the third normal form (3NF).

**4. User** (User_Id , User_name , Email_Id , Contact_info , Password , DOB , Age)

**PK dependency:**
User_id → User_name, Email_Id , Contact_info , Password , DOB , Age

**Functional Dependencies:**
User_id→ User_id
User_id→ User_name
User_id→ Email_id
User_id→ mobile
User_id→ password
User_id→ DOB
User_id→ Age
DOB→ Age

**Partial Key Dependency**: None
**Transitive Dependency**: None
**Redundancies**: None

**Anomalies**:
Insert – None.

Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary

key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF. There is no partial dependency here, so it is in 2NF as well.

2NF Redundancies: None

Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.
For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.
Hence, this relation is in BCNF as well.

## 5. Evaluate(hack_id , team_id , judge_id , comment)

**PK dependency:**
Hack_id , team_id → judge_id ,  comment

**Functional Dependencies:**
Hack_id , Team_id → Judge_id , comment

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.
Hence, this relation is in BCNF as well.

### 6. Submission (Hack_id, Team_id, Time, Evaluation_Status)

**PK dependency:**
Hack_id, Team_id, Time → Evaluation_Status

**Functional Dependencies:**
Hack_id, Team_id, Time → Hack_id, Team_id, Time, Evaluation_Status

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.
Hence, this relation is in BCNF as well.

**7. Judge** (Hack_id , Judge_Id ,User_Id , User_name , Email_Id , Contact_info , Password , DOB , Age, Name)

**PK dependency:**
**Hack_id , Judge_Id ,User_Id** → User_name , Email_Id , Contact_info , Password , DOB , Age, Name

**Functional Dependencies:**
**Hack_id , Judge_Id ,User_Id** → Hack_id , Judge_Id ,User_Id **,** User_name , Email_Id , Contact_info , Password , DOB , Age, Name

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.
Hence, this relation is in BCNF as well.

### 8. Team_info(Hack_id ,user_id, Team_id)

**PK dependency:**
Hack_id ,user_id, Team_id -> Hack_id ,user_id, Team_id

**Functional Dependencies:**
Hack_id ,user_id, Team_id -> Hack_id ,user_id, Team_id

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.
Hence, this relation is in BCNF as well.

**9. Team** (Hack_id , Team_Id , Team_size , Leader , Current_Status)

**PK dependency:**
**Hack_id , Team_id**→ Team_size , Leader , Current_Status

**Functional Dependencies:**
**Hack_id , Team_id** → Hack_id , Team_id , Team_size , Leader , Current_Status

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.
Hence, this relation is in BCNF as well.

**10. Sponsor**(Hack_id ,Sponser_id, Name)

**PK dependency:**
Hack_id , Name → Hack_id , Name

**Functional Dependencies:**
Hack_id , Team_id → Hack_id , Name

**Partial Key Dependency:** None
**Transitive Dependency:** None
**Redundancies:** None

**Anomalies:**
Insert – None.
Update – Updating this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
Delete – Deleting from this table will create problems sometimes as its primary key is being accessed as a foreign key in other tables.
In this schema, every attribute is single-valued (scalar) making it already in 1NF.
There is no partial dependency here, so it is in 2NF as well.
2NF Redundancies: None
Since there is no transitive dependency, it is also in 3NF.
Thus, our final schema remains the same.

For a relation to be in BCNF,
a. It should be in the third normal form (3NF).
b. For any dependency A → B, A must be a super key.
Hence, this relation is in BCNF as well.

## ❖ Final Schema:

- **User**(user_Id , Email_Id , name, Password , DOB , Age , mobile)

- **Organizer**(user_id , hack_id )
  - FK hack_id references to **Hackathon**
  - FK user_id references to **User**

- **Team**(Hack_id , Team_Id)
  - FK Hack_id references to **Hackathon**

- **Team_info**(user_id , Hack_id , Team_id)
  - FK user_id references to **User**
  - FK Hack_id references to **Hackathon**
  - FK Team_id references to **Team**

- **Participant**(User_Id, Hack_id , domain)
  - FK Hack_id references to **Hackathon**
  - FK user_id references to **User**

- **Submission**(Hack_id, team_id , Time , Curr_Status)
  - FK Hack_id references to **Hackathon**
  - FK team_id references to **Team**

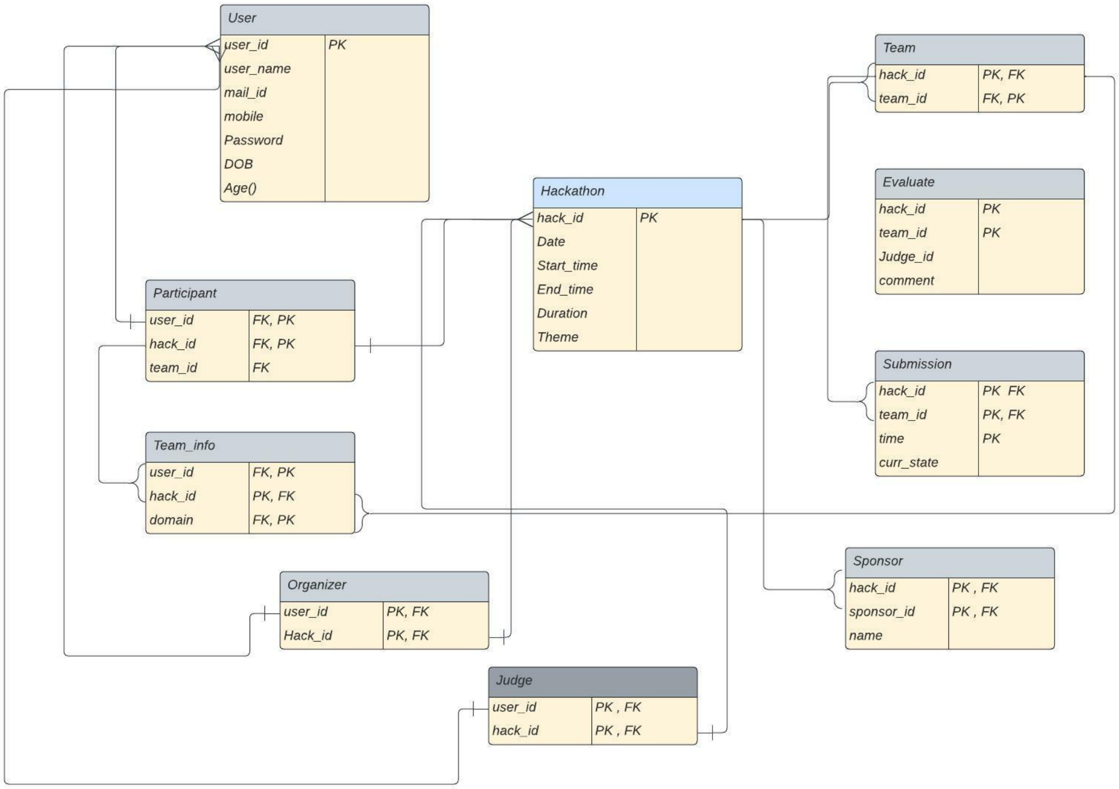
- **Sponsor**(Hack_id , Sponsor_id , Name)
  - FK Hack_id references to **Hackathon**

- **Hackathon**(Hack_id , date , start_time , end_time , duration , theme)

- **Evaluate**(hack_id , team_id , judge_id , comment)
  - FK Hack_id references to **Hackathon**
  - FK team_id references to **Team**

- **Judge**(Hack_id , Judge_Id )
  - FK Hack_id references to **Hackathon**
  - FK Judge_id references to **User**

# Relational Model



# DDL Script

## Hackathon:

```
-- Table: hm.Hackathon

-- DROP TABLE IF EXISTS hm."Hackathon";

CREATE TABLE IF NOT EXISTS hm."Hackathon"
(
   hack_id character varying COLLATE pg_catalog."default" NOT NULL,
   date date NOT NULL,
   s_time time without time zone NOT NULL,
   e_time time without time zone NOT NULL,
   duration character varying COLLATE pg_catalog."default" NOT NULL,
   theme character varying COLLATE pg_catalog."default" NOT NULL,
   CONSTRAINT "Hackathon_pkey" PRIMARY KEY (hack_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hm."Hackathon"
   OWNER to postgres;
```

## User:

```
-- Table: hm.User

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

CREATE TABLE IF NOT EXISTS hm."User"
(
   user_id character varying COLLATE pg_catalog."default" NOT NULL,
   email_id character varying COLLATE pg_catalog."default" NOT NULL,
   name character varying COLLATE pg_catalog."default" NOT NULL,
   password character varying COLLATE pg_catalog."default" NOT NULL,
   dob date NOT NULL,
   age integer NOT NULL,
   CONSTRAINT "User_pkey" PRIMARY KEY (user_id)
)

TABLESPACE pg_default;

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

## Participant:

```
-- Table: hm.Participant

-- DROP TABLE IF EXISTS hm."Participant";

CREATE TABLE IF NOT EXISTS hm."Participant"
(
   user_id character varying COLLATE pg_catalog."default" NOT NULL,
   hack_id character varying COLLATE pg_catalog."default" NOT NULL,
   domain character varying COLLATE pg_catalog."default" NOT NULL,
   CONSTRAINT "Participant_pkey" PRIMARY KEY (user_id, hack_id),
   CONSTRAINT fk_hack FOREIGN KEY (hack_id)
      REFERENCES hm."Hackathon" (hack_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   CONSTRAINT fk_user FOREIGN KEY (user_id)
      REFERENCES hm."User" (user_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hm."Participant"
   OWNER to postgres;
```

## Judge:

```
-- Table: hm.Judge

-- DROP TABLE IF EXISTS hm."Judge";

CREATE TABLE IF NOT EXISTS hm."Judge"
(
    user_id character varying COLLATE pg_catalog."default" NOT NULL,
    hack_id character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Judge_pkey" PRIMARY KEY (user_id, hack_id),
    CONSTRAINT fk_hack FOREIGN KEY (hack_id)
        REFERENCES hm."Hackathon" (hack_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_user FOREIGN KEY (user_id)
        REFERENCES hm."User" (user_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hm."Judge"
    OWNER to postgres;
```

## Organizer:

```
-- Table: hm.Organizer

-- DROP TABLE IF EXISTS hm."Organizer";

CREATE TABLE IF NOT EXISTS hm."Organizer"
(
    user_id character varying COLLATE pg_catalog."default" NOT NULL,
    hack_id character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Organizer_pkey" PRIMARY KEY (user_id, hack_id),
    CONSTRAINT fk_hack FOREIGN KEY (hack_id)
        REFERENCES hm."Hackathon" (hack_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_user FOREIGN KEY (user_id)
        REFERENCES hm."User" (user_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hm."Organizer"
    OWNER to postgres;
```

## Sponsor:

```
-- Table: hm.Sponsor

-- DROP TABLE IF EXISTS hm."Sponsor";

CREATE TABLE IF NOT EXISTS hm."Sponsor"
(
    sponsor_id character varying COLLATE pg_catalog."default" NOT NULL,
    hack_id character varying COLLATE pg_catalog."default" NOT NULL,
    name character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Sponser_pkey" PRIMARY KEY (sponsor_id, hack_id),
    CONSTRAINT fk_hack FOREIGN KEY (hack_id)
        REFERENCES hm."Hackathon" (hack_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hm."Sponsor"
    OWNER to postgres;
```

## Team:

```
-- Table: hm_db.Team

-- DROP TABLE hm_db."Team";

CREATE TABLE hm_db."Team"
(
    "Curr_status" character varying COLLATE pg_catalog."default" NOT NULL,
    "Leader" character varying COLLATE pg_catalog."default" NOT NULL,
    "Team_id" character varying COLLATE pg_catalog."default" NOT NULL,
    "Hack_id" character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Team_pkey" PRIMARY KEY ("Team_id", "Hack_id"),
    CONSTRAINT fk_team1 FOREIGN KEY ("Hack_id")
        REFERENCES hm_db."Hackathon" ("Hack_id") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT fk_team2 FOREIGN KEY ("Leader")
        REFERENCES hm_db."Participant" ("User_id") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

TABLESPACE pg_default;

ALTER TABLE hm_db."Team"
    OWNER to postgres;
```

## Submission:

```
-- Table: hm.Submission

-- DROP TABLE IF EXISTS hm."Submission";

CREATE TABLE IF NOT EXISTS hm."Submission"
(
    hack_id character varying COLLATE pg_catalog."default" NOT NULL,
    team_id character varying COLLATE pg_catalog."default" NOT NULL,
    "time" time without time zone NOT NULL,
    curr_state character varying COLLATE pg_catalog."default",
    CONSTRAINT "Submission_pkey" PRIMARY KEY (hack_id, team_id, "time"),
    CONSTRAINT fk_team_hack FOREIGN KEY (hack_id, team_id)
        REFERENCES hm."Team" (hack_id, team_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hm."Submission"
    OWNER to postgres;

-- Trigger: curr
-- DROP TRIGGER IF EXISTS curr ON hm."Submission";
CREATE TRIGGER curr
    AFTER INSERT
    ON hm."Submission"
    FOR EACH ROW
    EXECUTE FUNCTION hm.func_state();

-- Trigger: present
-- DROP TRIGGER IF EXISTS present ON hm."Submission";

CREATE TRIGGER present
    AFTER INSERT
    ON hm."Submission"
    FOR EACH ROW
    EXECUTE FUNCTION hm.func_state();
```

## Team_info:

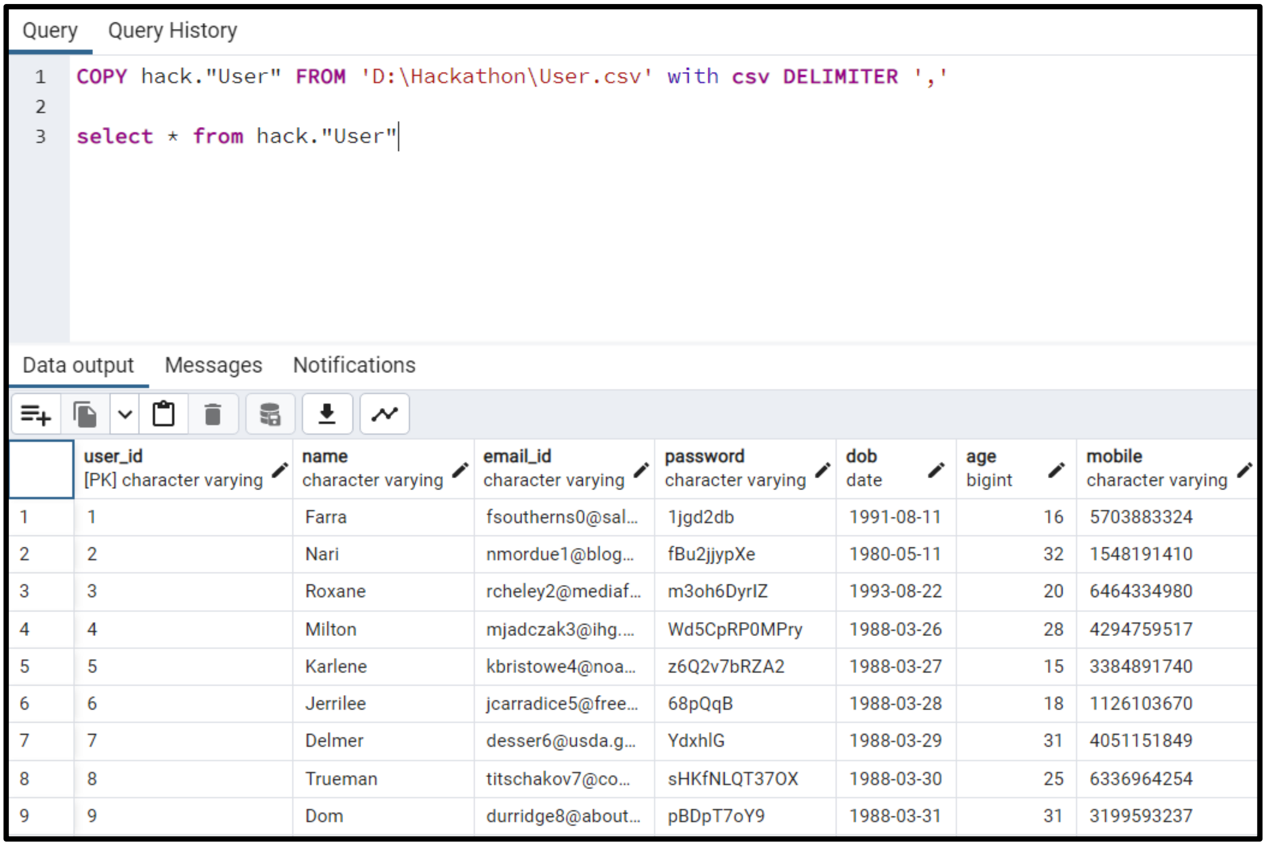
```
-- Table: hm.Team_info

-- DROP TABLE IF EXISTS hm."Team_info";

CREATE TABLE IF NOT EXISTS hm."Team_info"
(
    user_id character varying COLLATE pg_catalog."default" NOT NULL,
    hack_id character varying COLLATE pg_catalog."default" NOT NULL,
    team_id character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Team_info_pkey" PRIMARY KEY (user_id, hack_id, team_id),
    CONSTRAINT fk_team_hack FOREIGN KEY (hack_id, team_id)
        REFERENCES hm."Team" (hack_id, team_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_user_hack FOREIGN KEY (hack_id, user_id)
        REFERENCES hm."Participant" (hack_id, user_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hm."Team_info"
    OWNER to postgres;
```

## Evaluate:

```
-- Table: hm.Evaluate

-- DROP TABLE IF EXISTS hm."Evaluate";

CREATE TABLE IF NOT EXISTS hm."Evaluate"
(
   hack_id character varying COLLATE pg_catalog."default" NOT NULL,
   team_id character varying COLLATE pg_catalog."default" NOT NULL,
   judge_id character varying COLLATE pg_catalog."default" NOT NULL,
   "time" time without time zone NOT NULL,
   comment character varying COLLATE pg_catalog."default",
   CONSTRAINT "Evaluate_pkey" PRIMARY KEY (hack_id, team_id, judge_id,
"time"),
   CONSTRAINT fk_judge_hack FOREIGN KEY (hack_id, judge_id)
      REFERENCES hm."Judge" (hack_id, user_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   CONSTRAINT fk_team_hack_time FOREIGN KEY ("time", team_id, hack_id)
      REFERENCES hm."Submission" ("time", team_id, hack_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS hm."Evaluate"
   OWNER to postgres;
```

# SQL Queries

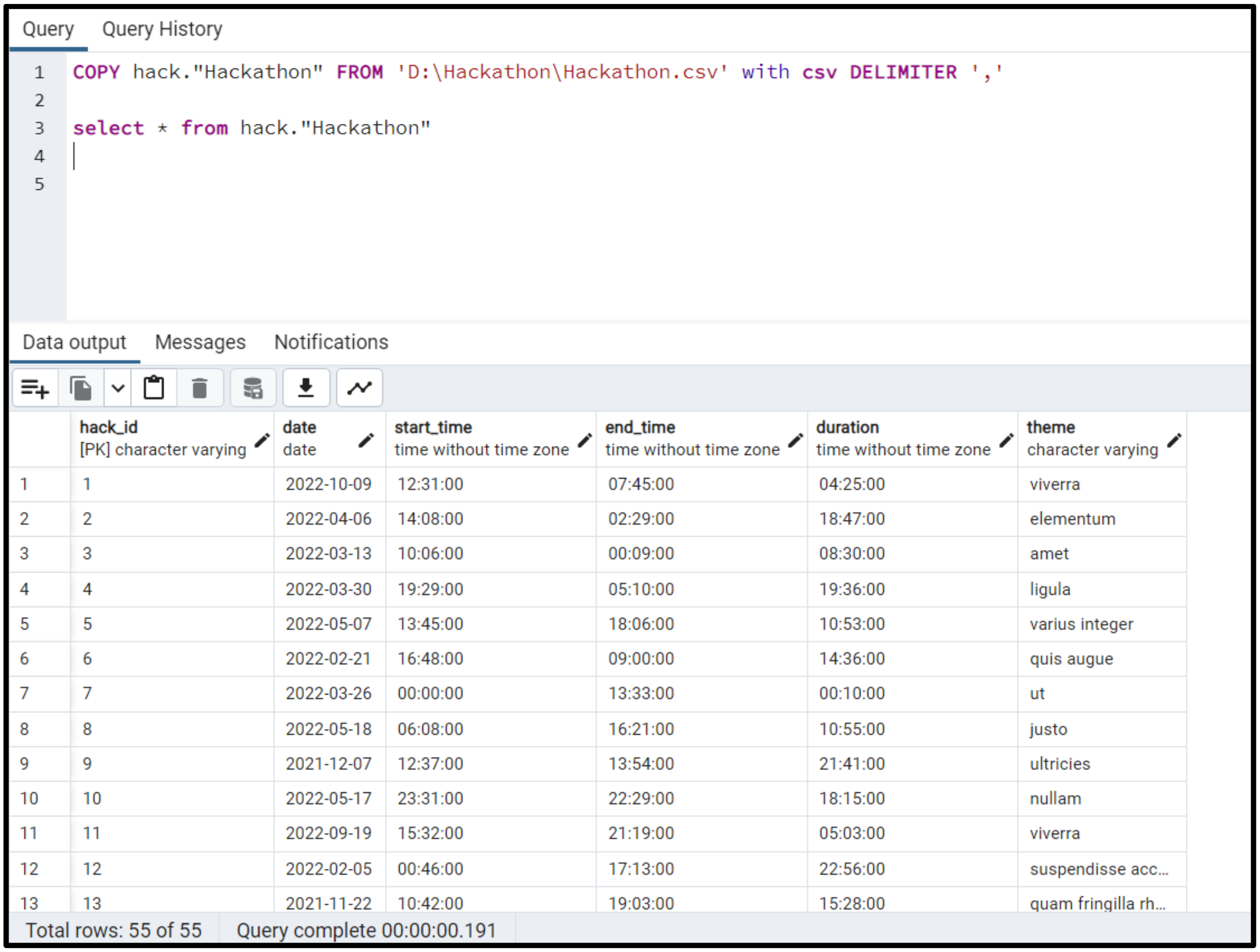
# Show User Table:



```
COPY hack."User" FROM 'D:\Hackathon\User.csv' with csv DELIMITER ','

select * from hack."User"
```

| | user_id | name | email_id | password | dob | age | mobile |
|---|---|---|---|---|---|---|---|
| | [PK] character varying | character varying | character varying | character varying | date | bigint | character varying |
| 1 | 1 | Farra | fsoutherns0@sal... | 1jgd2db | 1991-08-11 | 16 | 5703883324 |
| 2 | 2 | Nari | nmordue1@blog... | fBu2jjypXe | 1980-05-11 | 32 | 1548191410 |
| 3 | 3 | Roxane | rcheley2@mediaf... | m3oh6DyrIZ | 1993-08-22 | 20 | 6464334980 |
| 4 | 4 | Milton | mjadczak3@ihg.... | Wd5CpRP0MPry | 1988-03-26 | 28 | 4294759517 |
| 5 | 5 | Karlene | kbristowe4@noa... | z6Q2v7bRZA2 | 1988-03-27 | 15 | 3384891740 |
| 6 | 6 | Jerrilee | jcarradice5@free... | 68pQqB | 1988-03-28 | 18 | 1126103670 |
| 7 | 7 | Delmer | desser6@usda.g... | YdxhlG | 1988-03-29 | 31 | 4051151849 |
| 8 | 8 | Trueman | titschakov7@co... | sHKfNLQT37OX | 1988-03-30 | 25 | 6336964254 |
| 9 | 9 | Dom | durridge8@about... | pBDpT7oY9 | 1988-03-31 | 31 | 3199593237 |

# Show Hackathon Table:

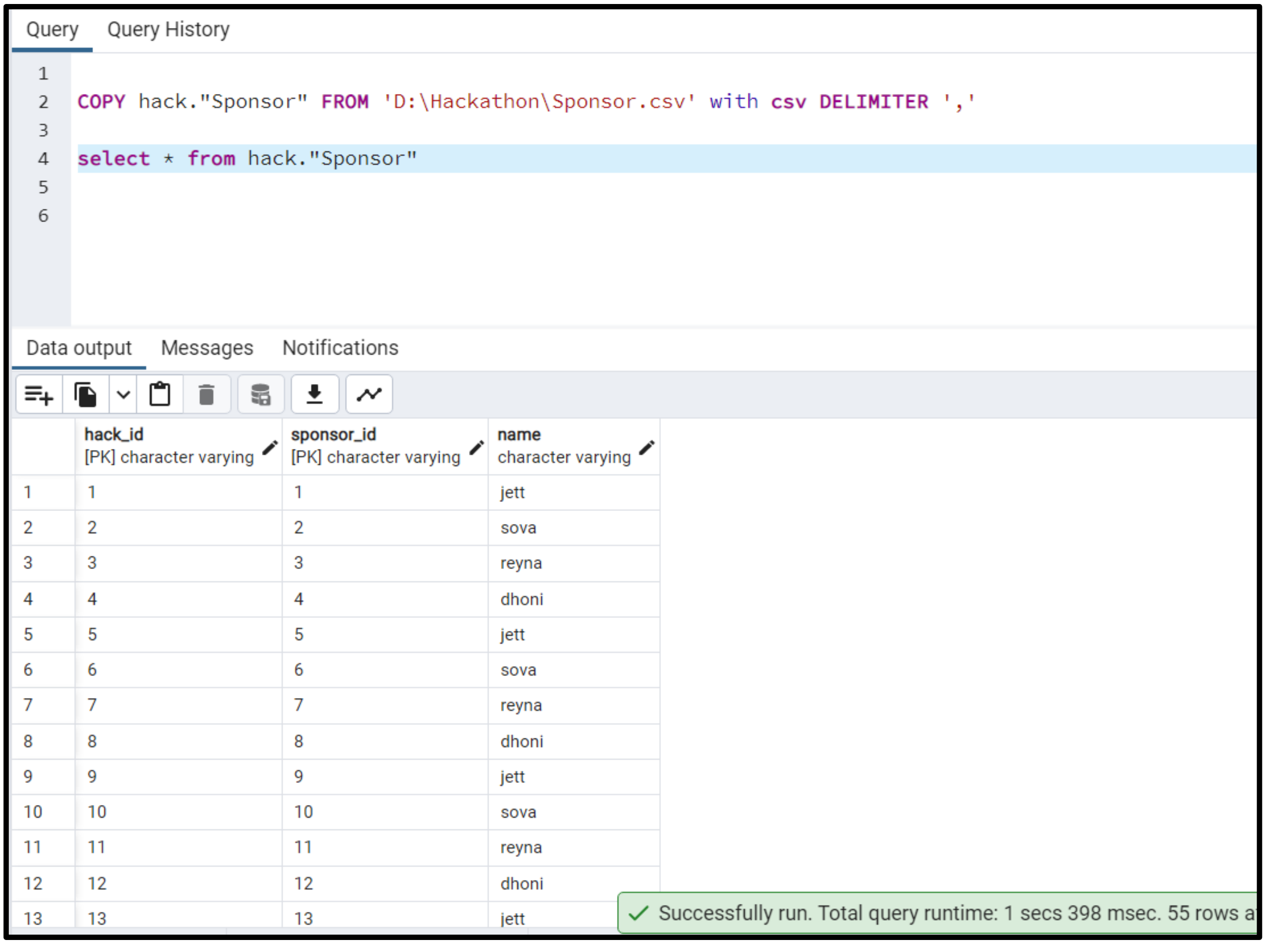


```
COPY hack."Hackathon" FROM 'D:\Hackathon\Hackathon.csv' with csv DELIMITER ','

select * from hack."Hackathon"

```

| | hack_id<br>[PK] character varying | date<br>date | start_time<br>time without time zone | end_time<br>time without time zone | duration<br>time without time zone | theme<br>character varying |
|---|---|---|---|---|---|---|
| 1 | 1 | 2022-10-09 | 12:31:00 | 07:45:00 | 04:25:00 | viverra |
| 2 | 2 | 2022-04-06 | 14:08:00 | 02:29:00 | 18:47:00 | elementum |
| 3 | 3 | 2022-03-13 | 10:06:00 | 00:09:00 | 08:30:00 | amet |
| 4 | 4 | 2022-03-30 | 19:29:00 | 05:10:00 | 19:36:00 | ligula |
| 5 | 5 | 2022-05-07 | 13:45:00 | 18:06:00 | 10:53:00 | varius integer |
| 6 | 6 | 2022-02-21 | 16:48:00 | 09:00:00 | 14:36:00 | quis augue |
| 7 | 7 | 2022-03-26 | 00:00:00 | 13:33:00 | 00:10:00 | ut |
| 8 | 8 | 2022-05-18 | 06:08:00 | 16:21:00 | 10:55:00 | justo |
| 9 | 9 | 2021-12-07 | 12:37:00 | 13:54:00 | 21:41:00 | ultricies |
| 10 | 10 | 2022-05-17 | 23:31:00 | 22:29:00 | 18:15:00 | nullam |
| 11 | 11 | 2022-09-19 | 15:32:00 | 21:19:00 | 05:03:00 | viverra |
| 12 | 12 | 2022-02-05 | 00:46:00 | 17:13:00 | 22:56:00 | suspendisse acc... |
| 13 | 13 | 2021-11-22 | 10:42:00 | 19:03:00 | 15:28:00 | quam fringilla rh... |

Total rows: 55 of 55 Query complete 00:00:00.191

# Show Sponsor Table:

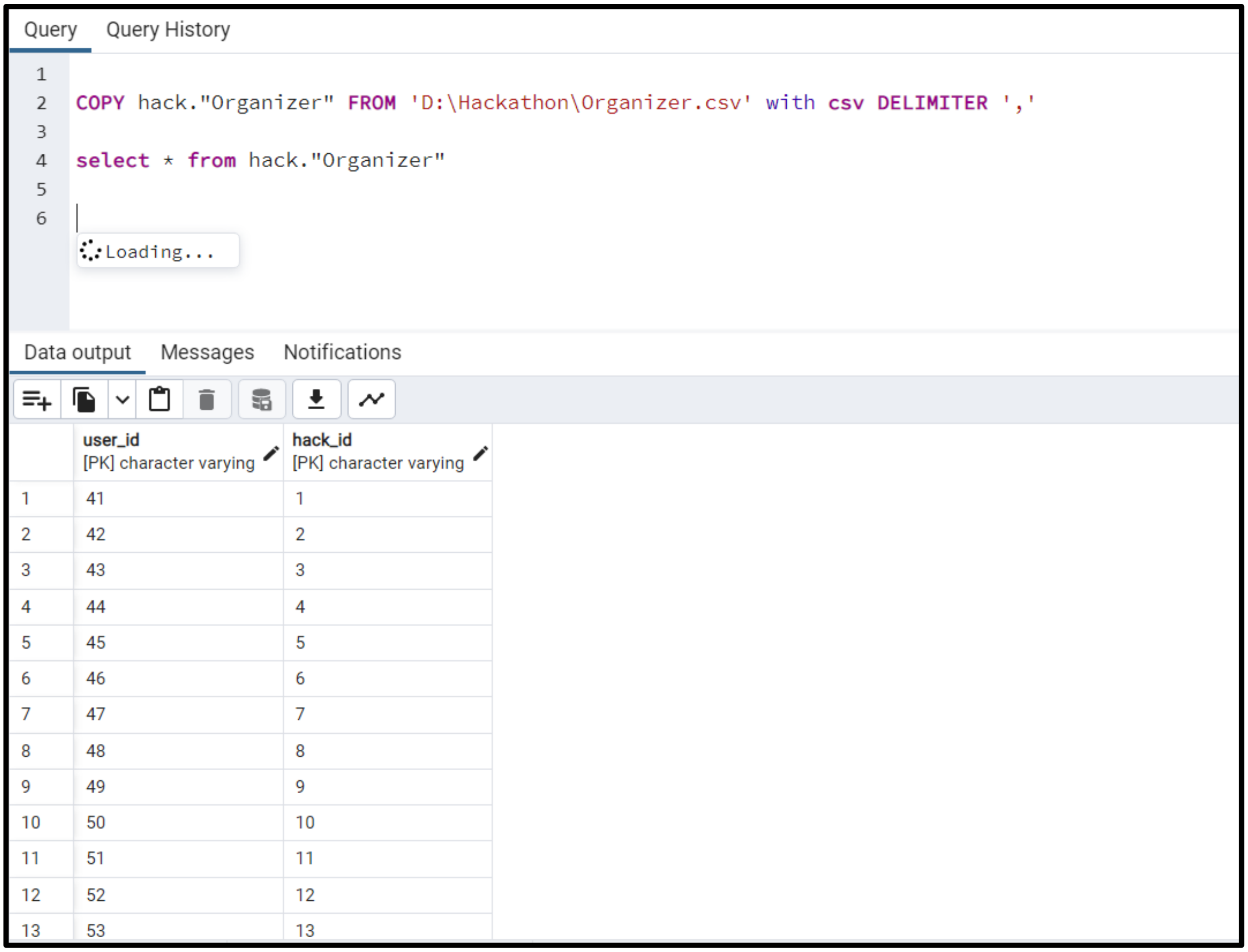


```
COPY hack."Sponsor" FROM 'D:\Hackathon\Sponsor.csv' with csv DELIMITER ','

select * from hack."Sponsor"
```

| | hack_id<br>[PK] character varying | sponsor_id<br>[PK] character varying | name<br>character varying |
|---|---|---|---|
| 1 | 1 | 1 | jett |
| 2 | 2 | 2 | sova |
| 3 | 3 | 3 | reyna |
| 4 | 4 | 4 | dhoni |
| 5 | 5 | 5 | jett |
| 6 | 6 | 6 | sova |
| 7 | 7 | 7 | reyna |
| 8 | 8 | 8 | dhoni |
| 9 | 9 | 9 | jett |
| 10 | 10 | 10 | sova |
| 11 | 11 | 11 | reyna |
| 12 | 12 | 12 | dhoni |
| 13 | 13 | 13 | jett |

✓ Successfully run. Total query runtime: 1 secs 398 msec. 55 rows a

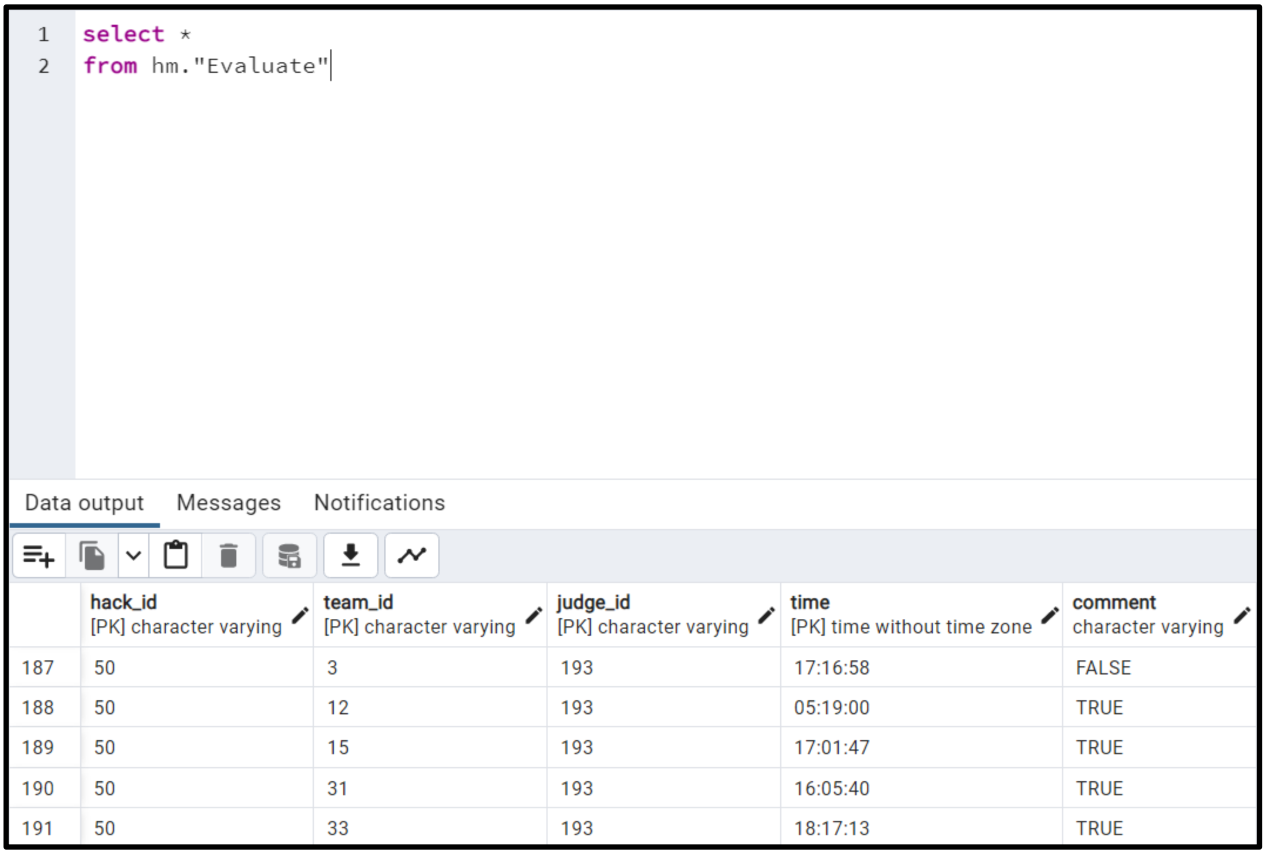
# Show Organizer Table:



```
COPY hack."Organizer" FROM 'D:\Hackathon\Organizer.csv' with csv DELIMITER ','

select * from hack."Organizer"
```

| | user_id<br>[PK] character varying | hack_id<br>[PK] character varying |
|---|---|---|
| 1 | 41 | 1 |
| 2 | 42 | 2 |
| 3 | 43 | 3 |
| 4 | 44 | 4 |
| 5 | 45 | 5 |
| 6 | 46 | 6 |
| 7 | 47 | 7 |
| 8 | 48 | 8 |
| 9 | 49 | 9 |
| 10 | 50 | 10 |
| 11 | 51 | 11 |
| 12 | 52 | 12 |
| 13 | 53 | 13 |

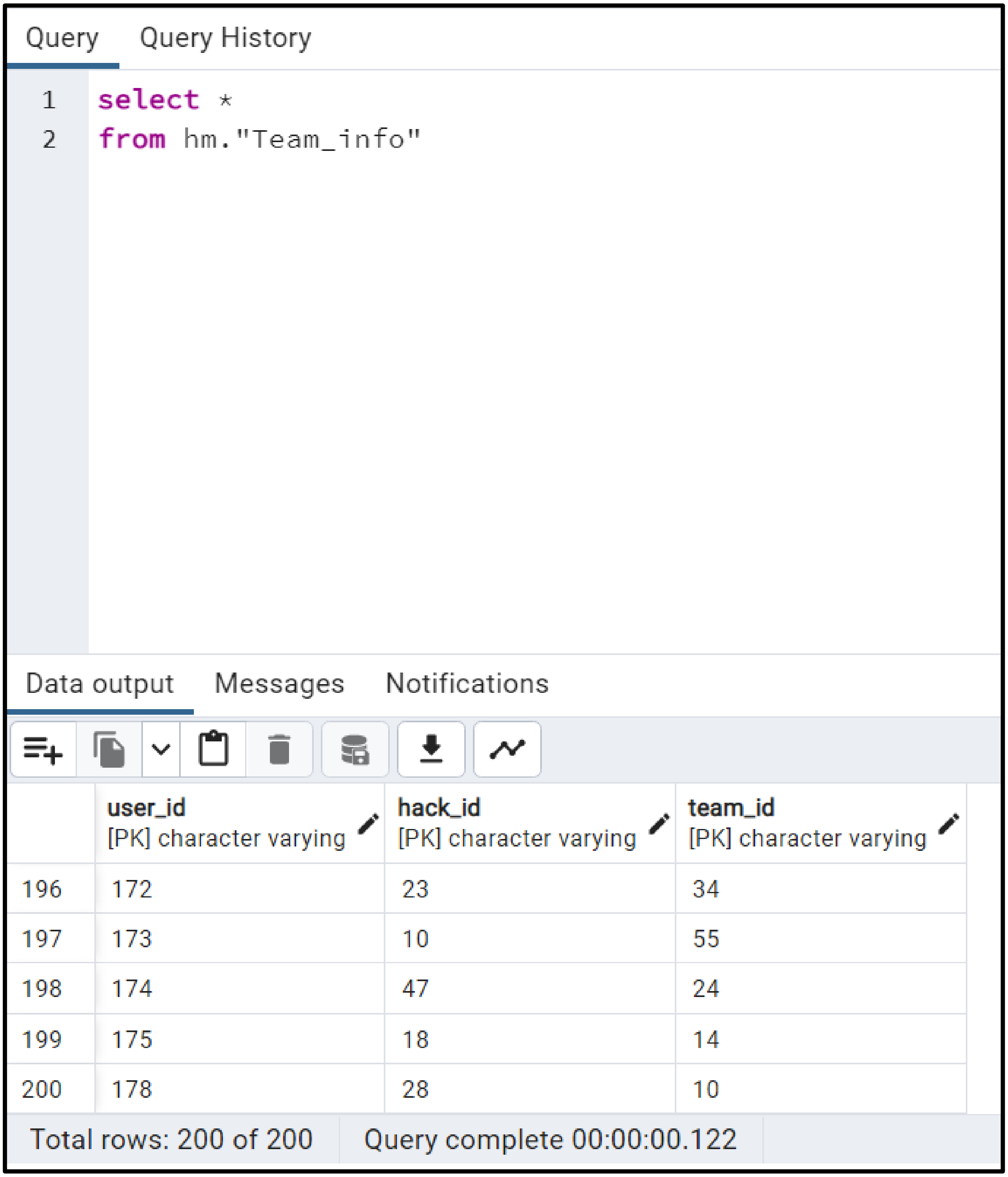
# Show Evaluate Table:



```
select *
from hm."Evaluate"
```

| | hack_id<br>[PK] character varying | team_id<br>[PK] character varying | judge_id<br>[PK] character varying | time<br>[PK] time without time zone | comment<br>character varying |
|---|---|---|---|---|---|
| 187 | 50 | 3 | 193 | 17:16:58 | FALSE |
| 188 | 50 | 12 | 193 | 05:19:00 | TRUE |
| 189 | 50 | 15 | 193 | 17:01:47 | TRUE |
| 190 | 50 | 31 | 193 | 16:05:40 | TRUE |
| 191 | 50 | 33 | 193 | 18:17:13 | TRUE |

## Show Team_info Table:

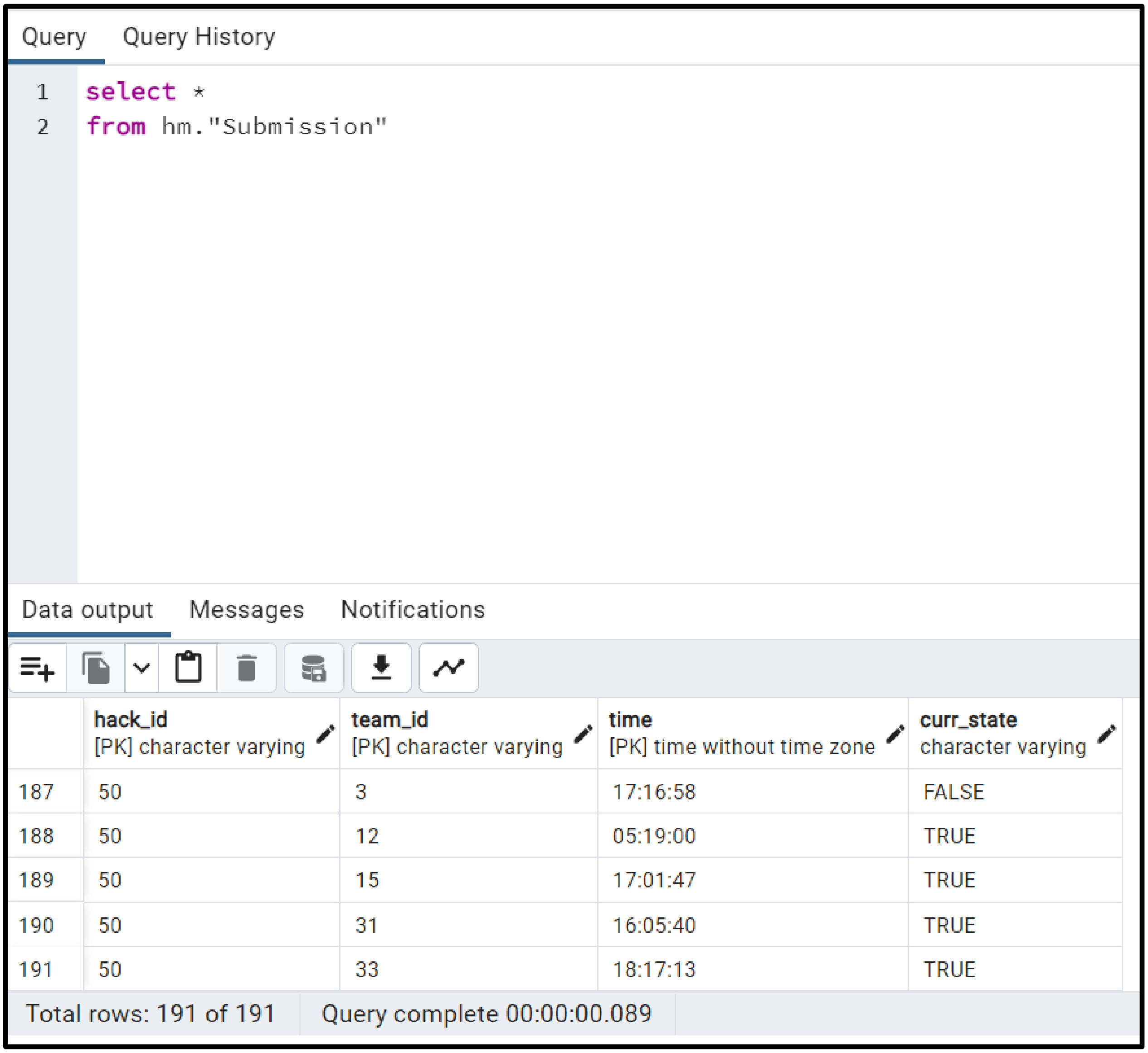


```
select *
from hm."Team_info"
```

| | user_id<br>[PK] character varying | hack_id<br>[PK] character varying | team_id<br>[PK] character varying |
|---|---|---|---|
| 196 | 172 | 23 | 34 |
| 197 | 173 | 10 | 55 |
| 198 | 174 | 47 | 24 |
| 199 | 175 | 18 | 14 |
| 200 | 178 | 28 | 10 |

Total rows: 200 of 200 Query complete 00:00:00.122

# Show Submission Table:

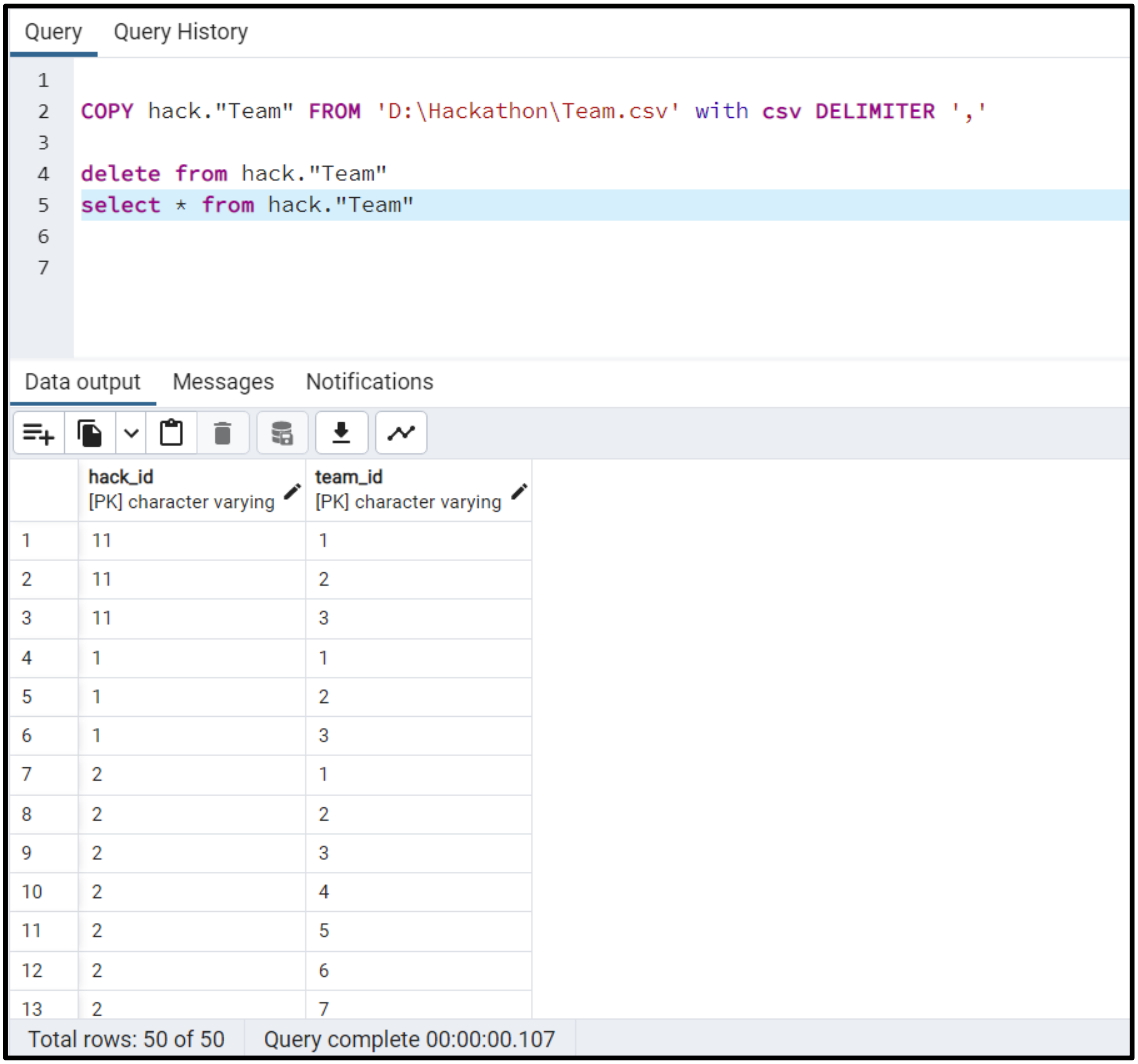


```sql
select *
from hm."Submission"
```

| | hack_id<br>[PK] character varying | team_id<br>[PK] character varying | time<br>[PK] time without time zone | curr_state<br>character varying |
|---|---|---|---|---|
| 187 | 50 | 3 | 17:16:58 | FALSE |
| 188 | 50 | 12 | 05:19:00 | TRUE |
| 189 | 50 | 15 | 17:01:47 | TRUE |
| 190 | 50 | 31 | 16:05:40 | TRUE |
| 191 | 50 | 33 | 18:17:13 | TRUE |

Total rows: 191 of 191 Query complete 00:00:00.089

## Show Team Table:

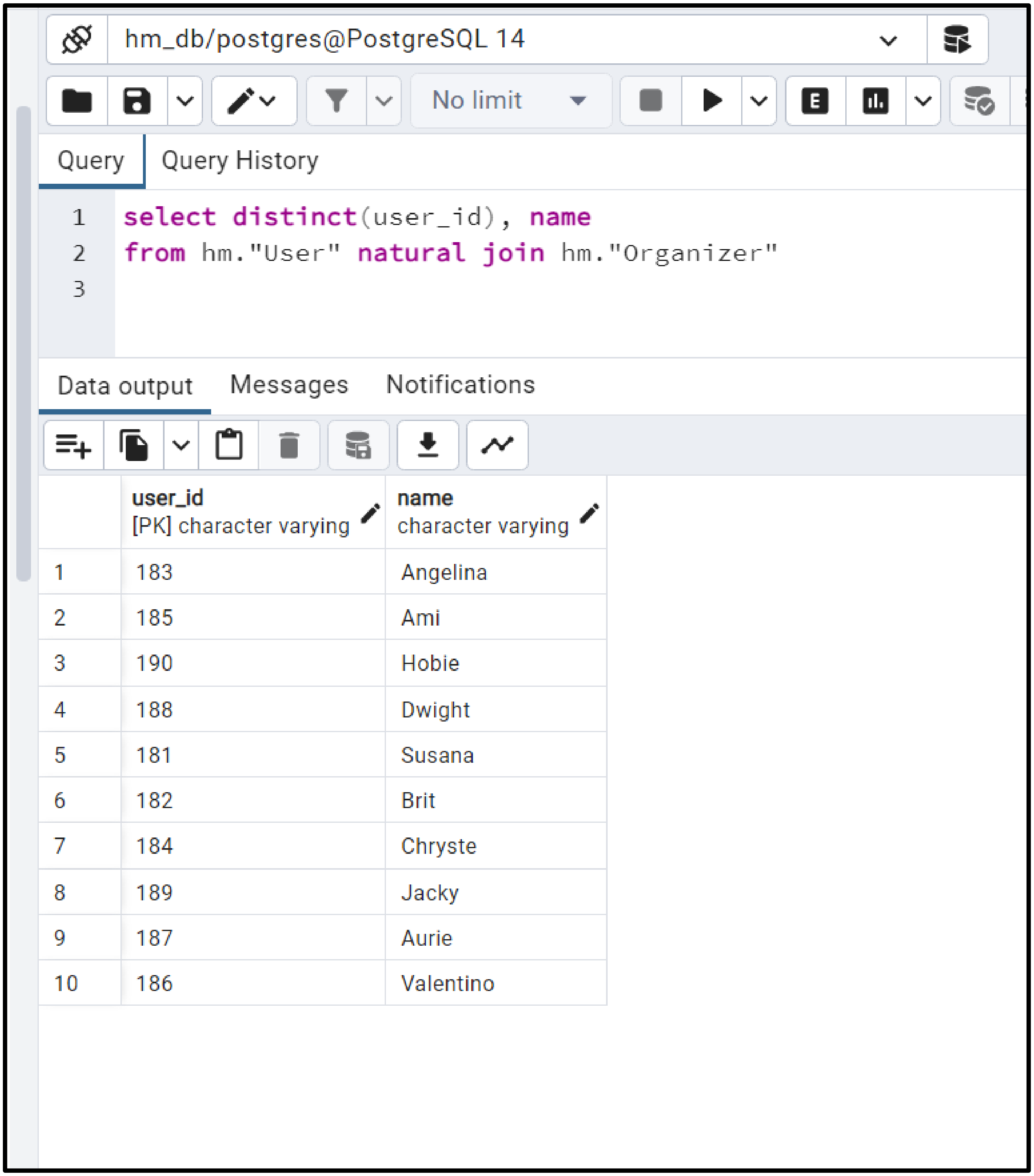


```

COPY hack."Team" FROM 'D:\Hackathon\Team.csv' with csv DELIMITER ','

delete from hack."Team"
select * from hack."Team"
```

| | hack_id<br>[PK] character varying | team_id<br>[PK] character varying |
|---|---|---|
| 1 | 11 | 1 |
| 2 | 11 | 2 |
| 3 | 11 | 3 |
| 4 | 1 | 1 |
| 5 | 1 | 2 |
| 6 | 1 | 3 |
| 7 | 2 | 1 |
| 8 | 2 | 2 |
| 9 | 2 | 3 |
| 10 | 2 | 4 |
| 11 | 2 | 5 |
| 12 | 2 | 6 |
| 13 | 2 | 7 |

```
Total rows: 50 of 50    Query complete 00:00:00.107
```

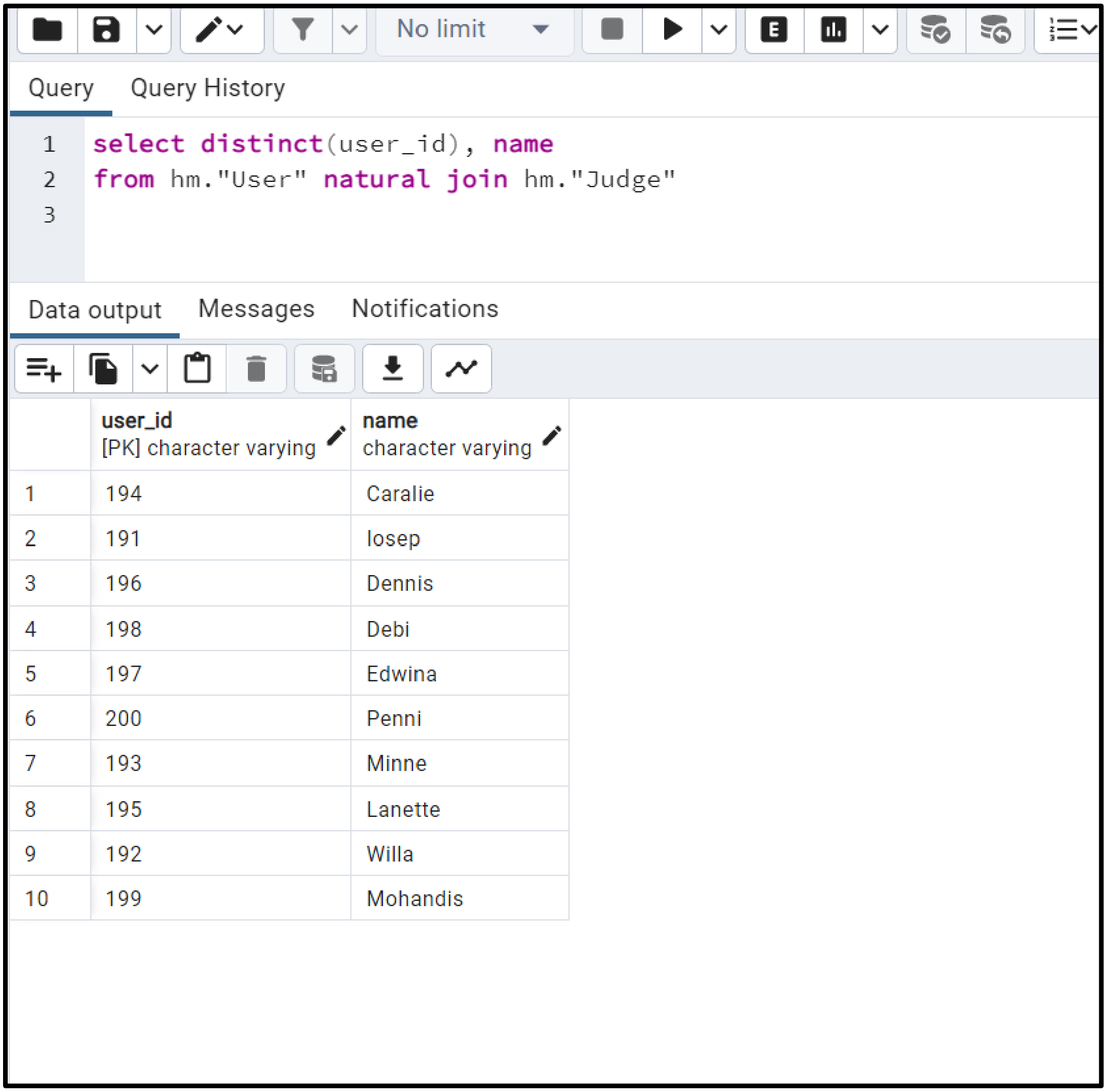
1) Show name and user_id of the users who have Organized the Hackathon :



```
select distinct(user_id), name
from hm."User" natural join hm."Organizer"
```

| | user_id<br>[PK] character varying | name<br>character varying |
|---|---|---|
| 1 | 183 | Angelina |
| 2 | 185 | Ami |
| 3 | 190 | Hobie |
| 4 | 188 | Dwight |
| 5 | 181 | Susana |
| 6 | 182 | Brit |
| 7 | 184 | Chryste |
| 8 | 189 | Jacky |
| 9 | 187 | Aurie |
| 10 | 186 | Valentino |

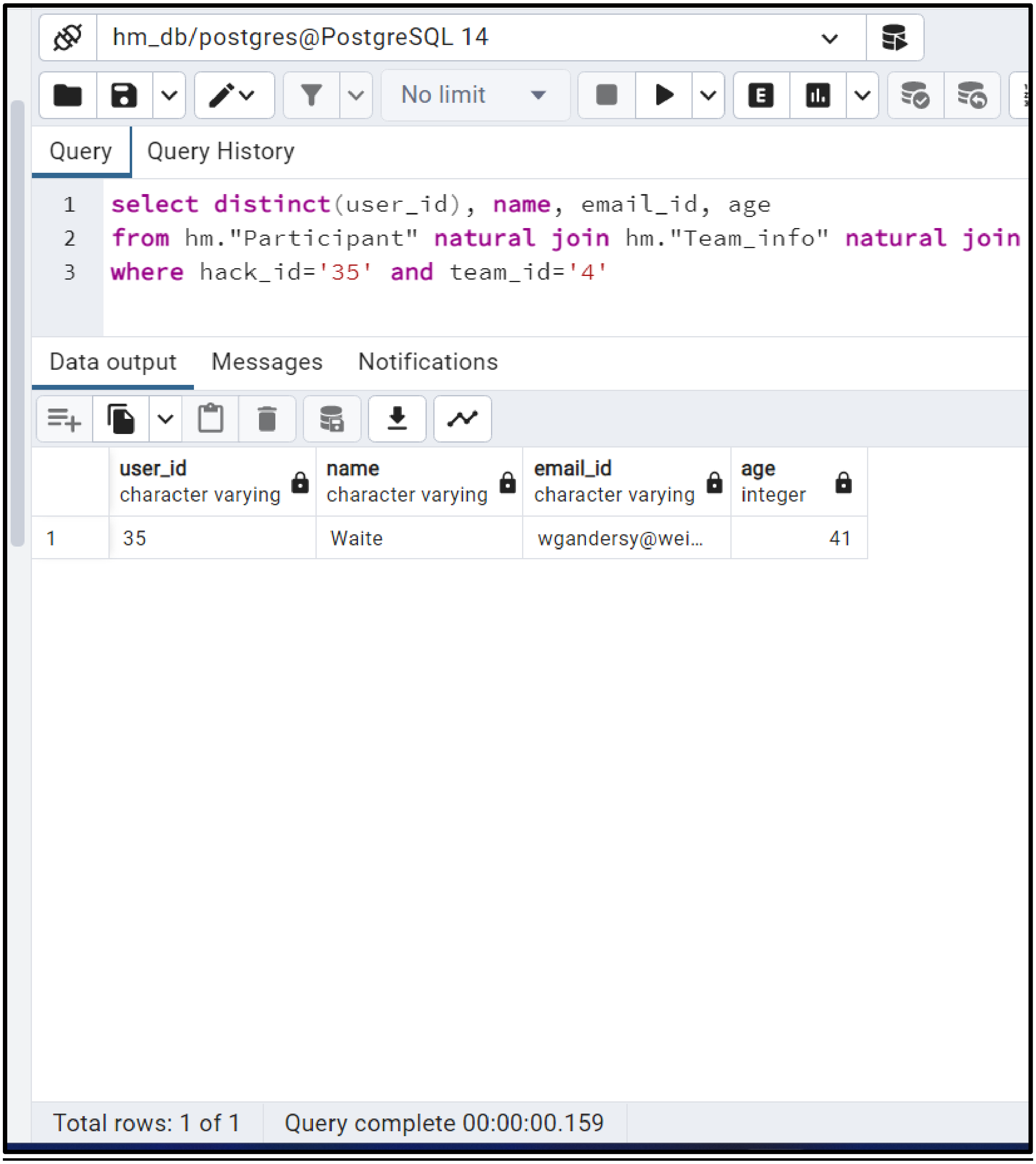
2) Show name and user_id of the users who have Evaluated the submissions in Hackathon :



```
select distinct(user_id), name
from hm."User" natural join hm."Judge"
```

| | user_id<br>[PK] character varying | name<br>character varying |
|---|---|---|
| 1 | 194 | Caralie |
| 2 | 191 | Iosep |
| 3 | 196 | Dennis |
| 4 | 198 | Debi |
| 5 | 197 | Edwina |
| 6 | 200 | Penni |
| 7 | 193 | Minne |
| 8 | 195 | Lanette |
| 9 | 192 | Willa |
| 10 | 199 | Mohandis |

3) Show the details of Participants in Hackathon with hack_id = 35 and team_id = 4 :

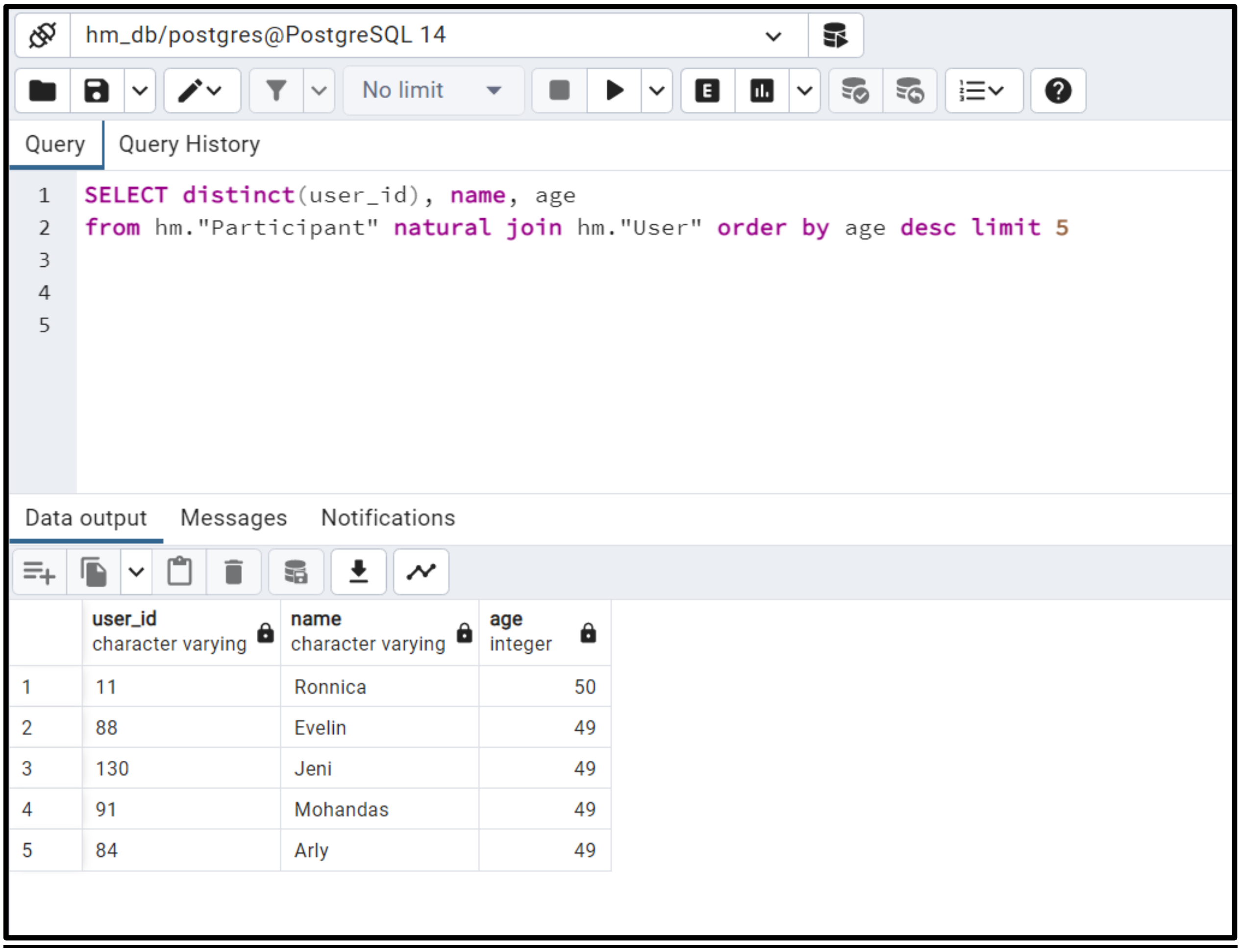


```
select distinct(user_id), name, email_id, age
from hm."Participant" natural join hm."Team_info" natural join
where hack_id='35' and team_id='4'
```

|  | user_id<br>character varying | name<br>character varying | email_id<br>character varying | age<br>integer |
|---|---|---|---|---|
| 1 | 35 | Waite | wgandersy@wei... | 41 |

Total rows: 1 of 1 Query complete 00:00:00.159

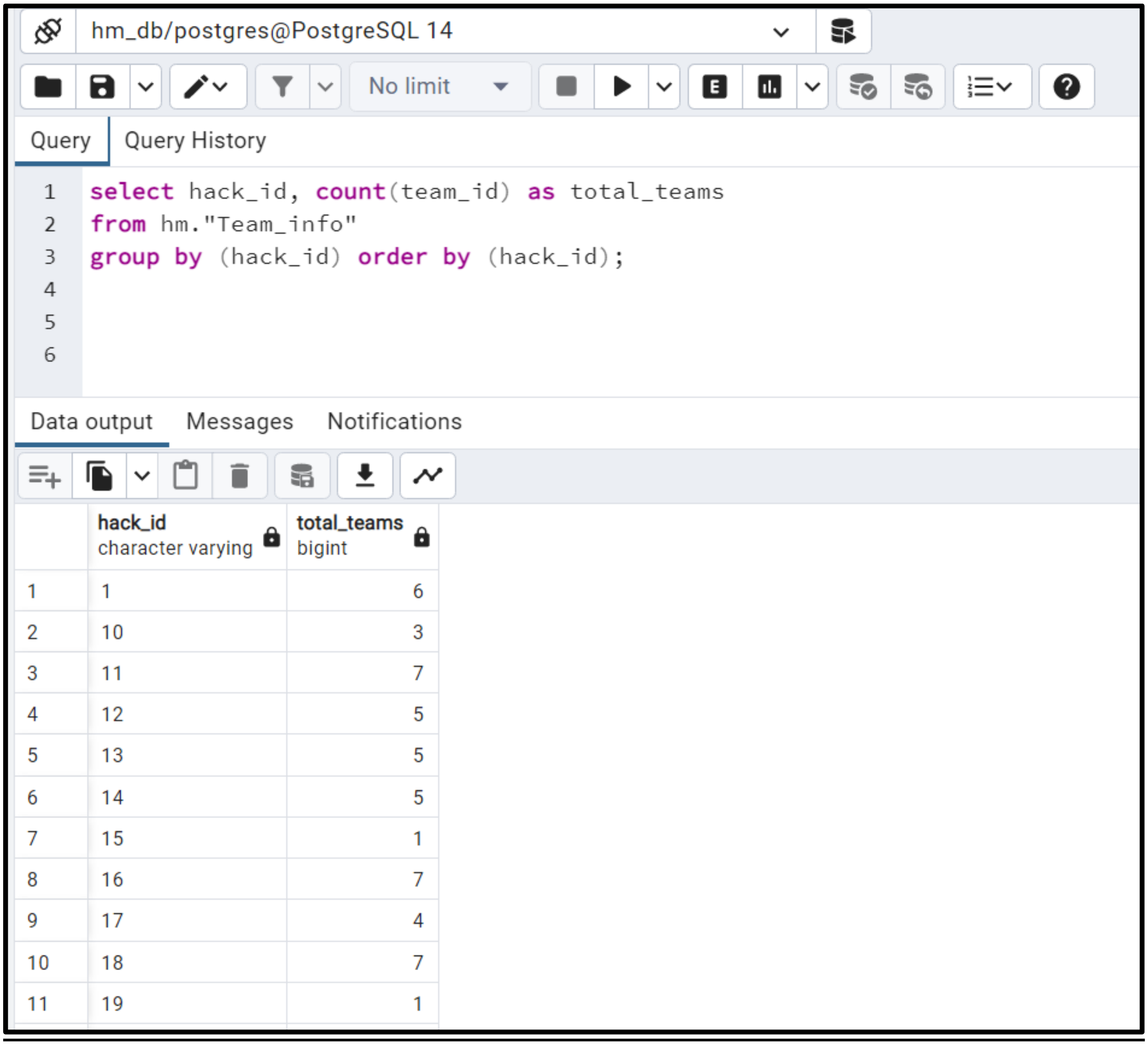
4) Show details of the first 5 oldest participants.



```
SELECT distinct(user_id), name, age
from hm."Participant" natural join hm."User" order by age desc limit 5
```

| | user_id<br>character varying | name<br>character varying | age<br>integer |
|---|---|---|---|
| 1 | 11 | Ronnica | 50 |
| 2 | 88 | Evelin | 49 |
| 3 | 130 | Jeni | 49 |
| 4 | 91 | Mohandas | 49 |
| 5 | 84 | Arly | 49 |

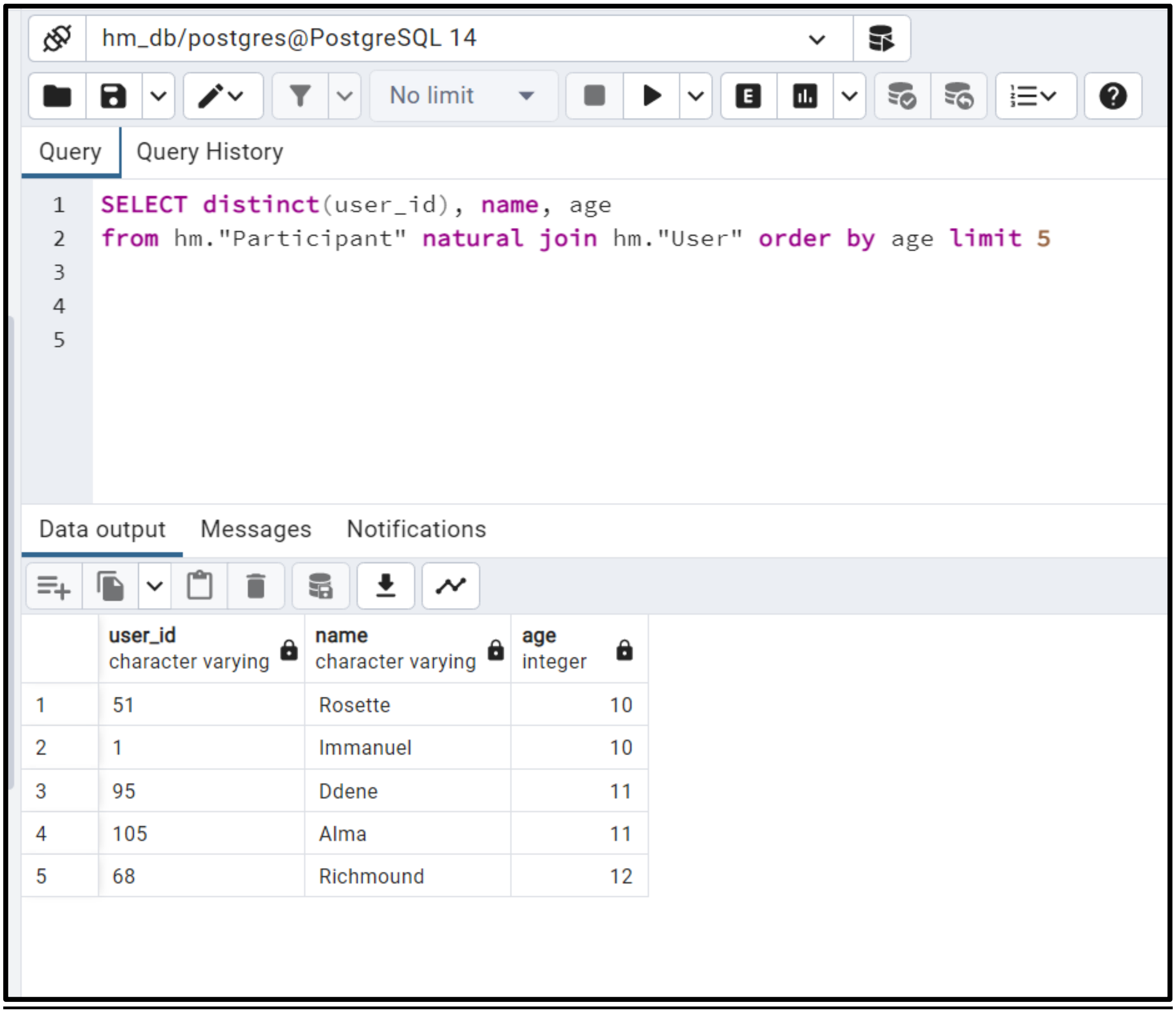
5) Show the number of the teams participating in each hackathon.



```
select hack_id, count(team_id) as total_teams
from hm."Team_info"
group by (hack_id) order by (hack_id);
```

| | hack_id<br>character varying | total_teams<br>bigint |
|---|---|---|
| 1 | 1 | 6 |
| 2 | 10 | 3 |
| 3 | 11 | 7 |
| 4 | 12 | 5 |
| 5 | 13 | 5 |
| 6 | 14 | 5 |
| 7 | 15 | 1 |
| 8 | 16 | 7 |
| 9 | 17 | 4 |
| 10 | 18 | 7 |
| 11 | 19 | 1 |

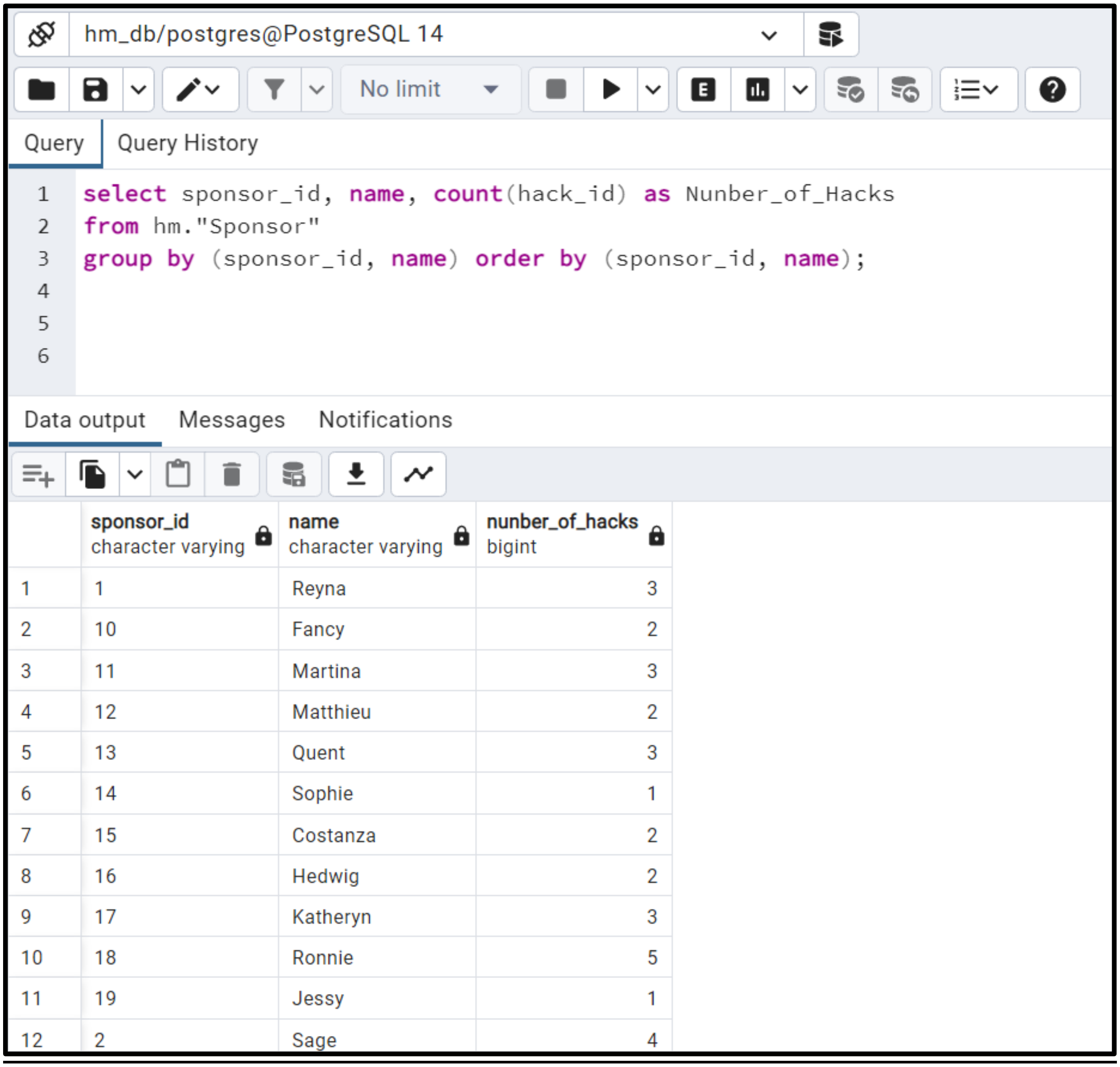
6) Show details of the first 5 youngest participants.



```
SELECT distinct(user_id), name, age
from hm."Participant" natural join hm."User" order by age limit 5
```

|  | user_id<br>character varying | name<br>character varying | age<br>integer |
|---|---|---|---|
| 1 | 51 | Rosette | 10 |
| 2 | 1 | Immanuel | 10 |
| 3 | 95 | Ddene | 11 |
| 4 | 105 | Alma | 11 |
| 5 | 68 | Richmound | 12 |

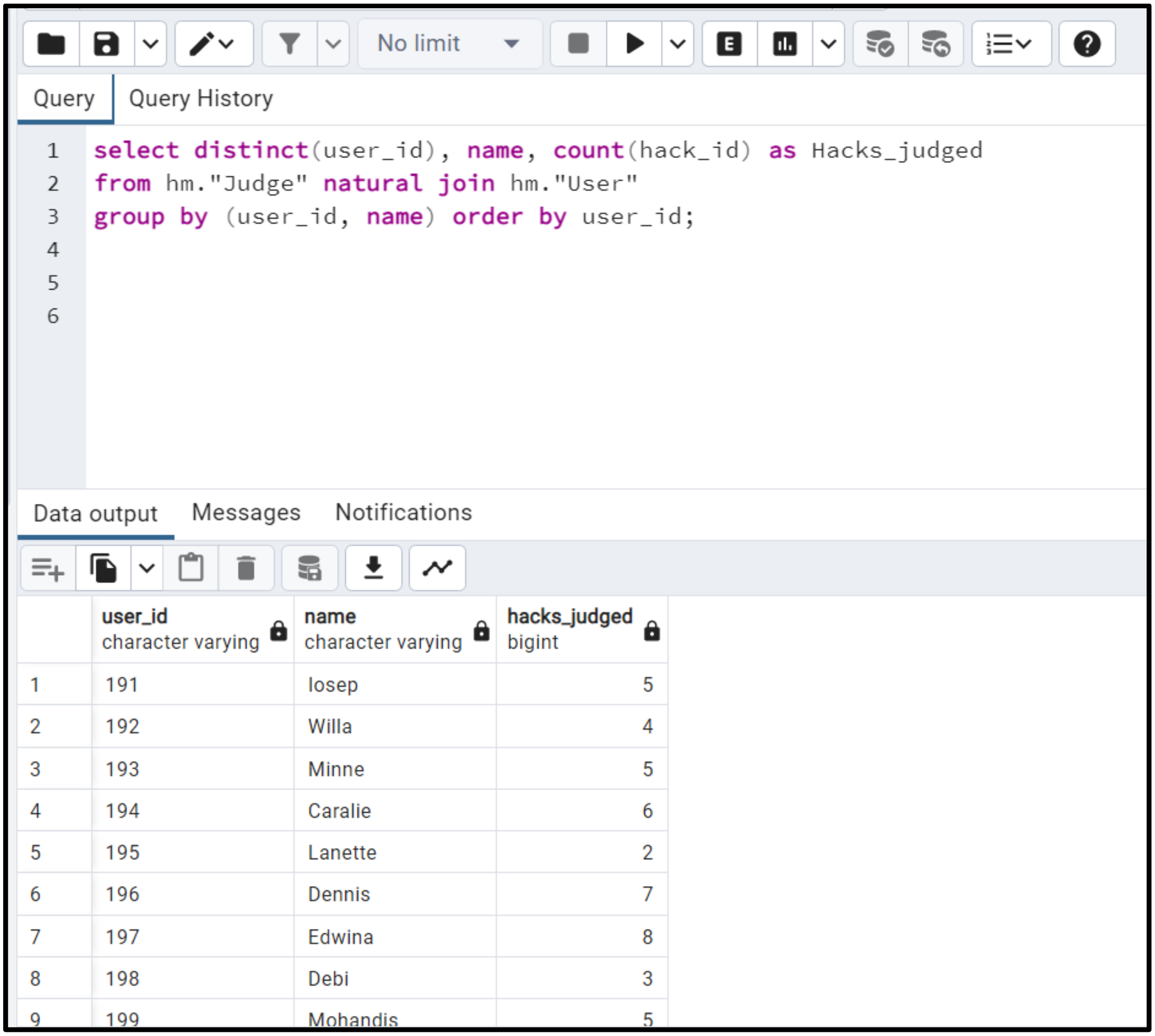
7) Show number of hackathons sponsored by each sponsor with their ids and name.



```
select sponsor_id, name, count(hack_id) as Nunber_of_Hacks
from hm."Sponsor"
group by (sponsor_id, name) order by (sponsor_id, name);
```

| | sponsor_id<br>character varying | name<br>character varying | nunber_of_hacks<br>bigint |
|---|---|---|---|
| 1 | 1 | Reyna | 3 |
| 2 | 10 | Fancy | 2 |
| 3 | 11 | Martina | 3 |
| 4 | 12 | Matthieu | 2 |
| 5 | 13 | Quent | 3 |
| 6 | 14 | Sophie | 1 |
| 7 | 15 | Costanza | 2 |
| 8 | 16 | Hedwig | 2 |
| 9 | 17 | Katheryn | 3 |
| 10 | 18 | Ronnie | 5 |
| 11 | 19 | Jessy | 1 |
| 12 | 2 | Sage | 4 |

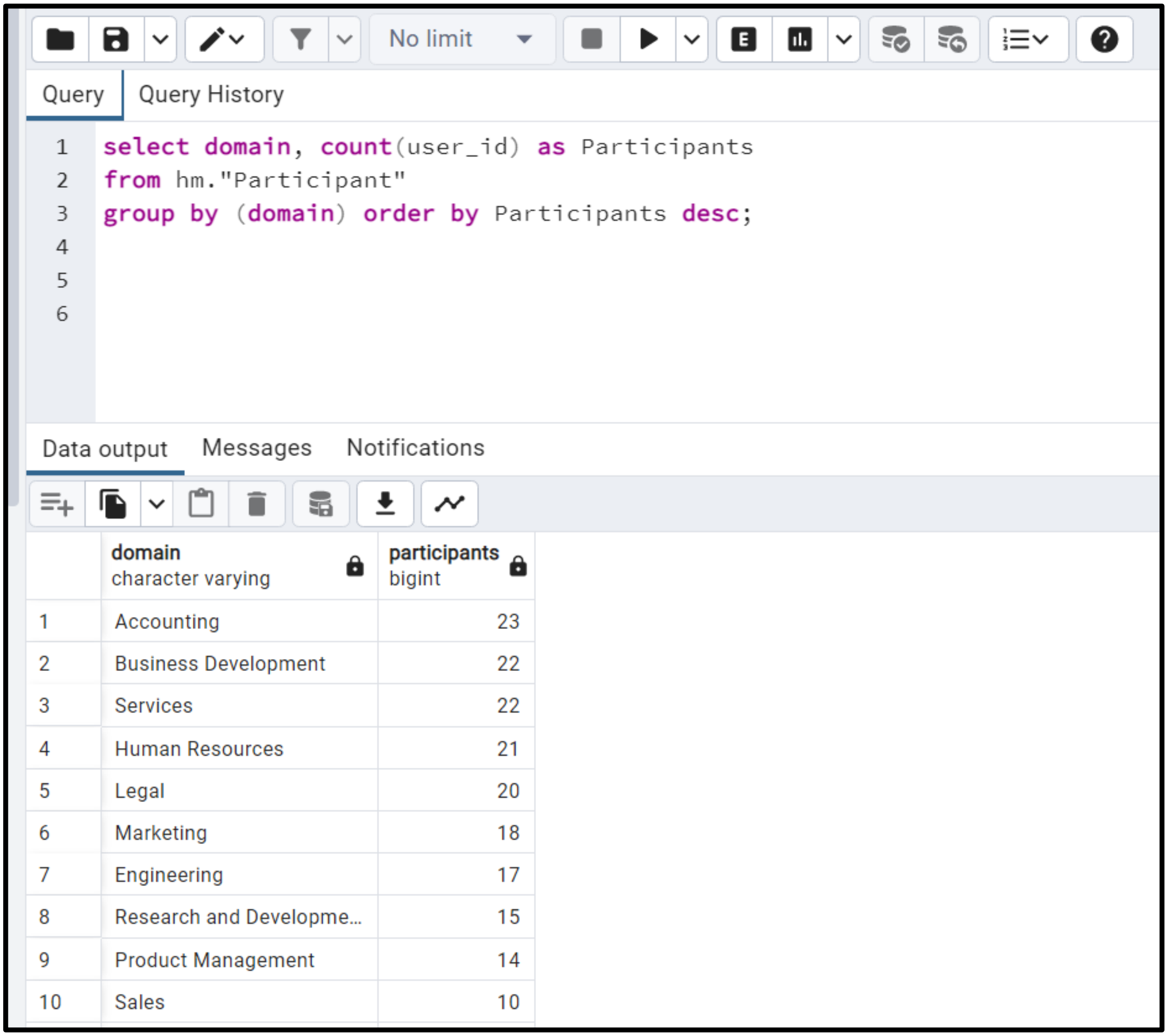
8) Show user_id and name of the users who have evaluated hackathons along with the number of hackathons they have evaluated.



```
select distinct(user_id), name, count(hack_id) as Hacks_judged
from hm."Judge" natural join hm."User"
group by (user_id, name) order by user_id;
```

| | user_id<br>character varying | name<br>character varying | hacks_judged<br>bigint |
|---|---|---|---|
| 1 | 191 | Iosep | 5 |
| 2 | 192 | Willa | 4 |
| 3 | 193 | Minne | 5 |
| 4 | 194 | Caralie | 6 |
| 5 | 195 | Lanette | 2 |
| 6 | 196 | Dennis | 7 |
| 7 | 197 | Edwina | 8 |
| 8 | 198 | Debi | 3 |
| 9 | 199 | Mohandis | 5 |

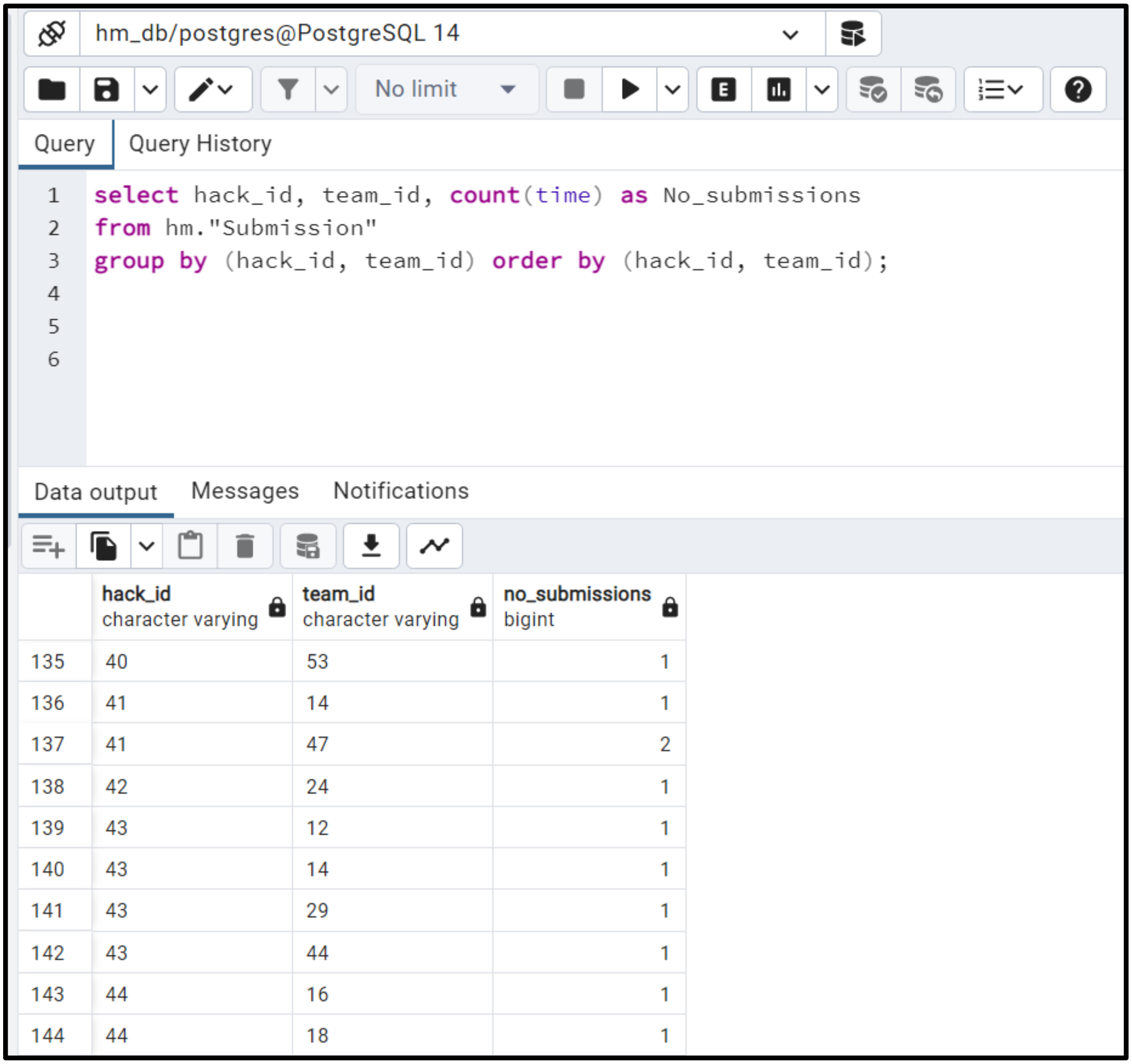
9) Show the number of participants corresponding to different domain. The domain with higher number of participants should be first.



```sql
select domain, count(user_id) as Participants
from hm."Participant"
group by (domain) order by Participants desc;
```

| | domain<br>character varying | participants<br>bigint |
|---|---|---|
| 1 | Accounting | 23 |
| 2 | Business Development | 22 |
| 3 | Services | 22 |
| 4 | Human Resources | 21 |
| 5 | Legal | 20 |
| 6 | Marketing | 18 |
| 7 | Engineering | 17 |
| 8 | Research and Developme... | 15 |
| 9 | Product Management | 14 |
| 10 | Sales | 10 |

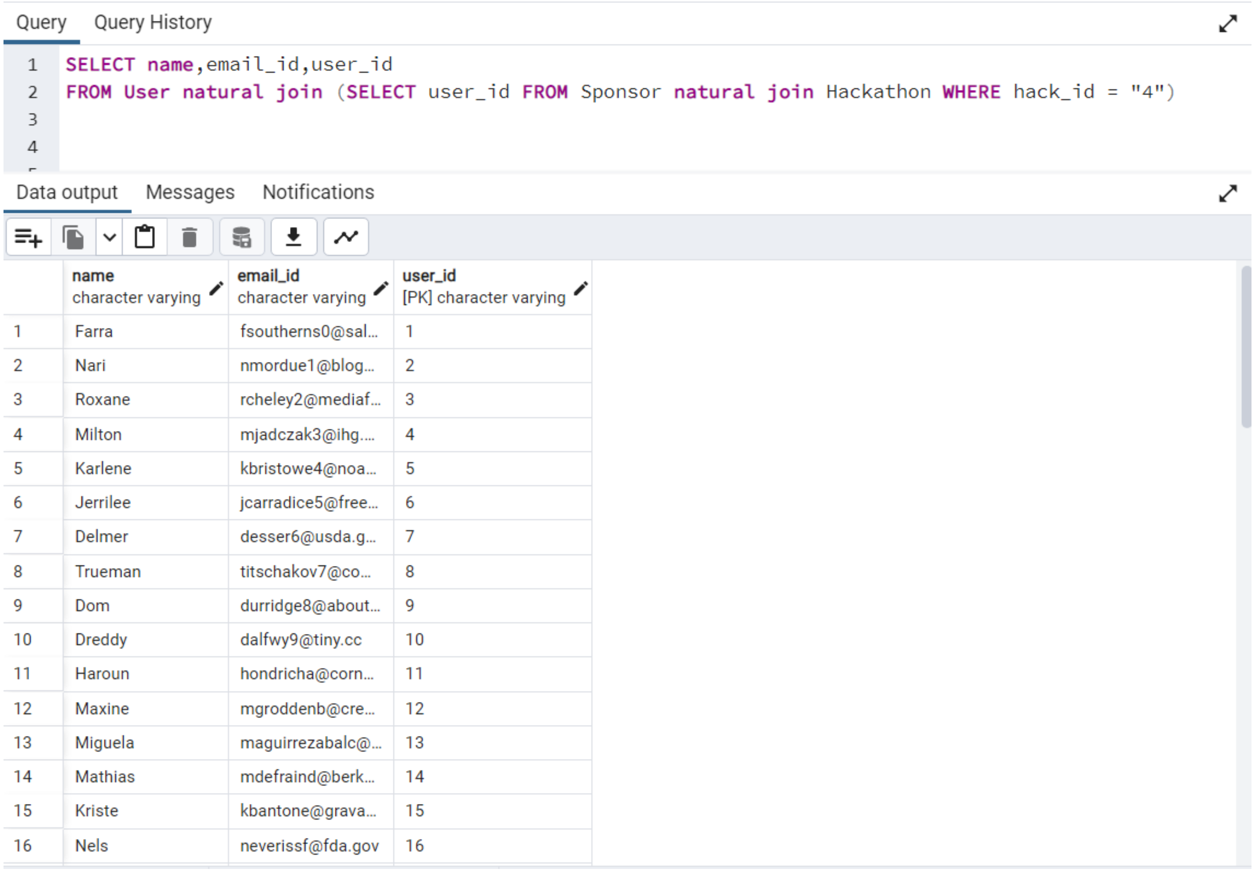
10) Show numbers of submissions made by each team corresponding to each hackathon.



```
select hack_id, team_id, count(time) as No_submissions
from hm."Submission"
group by (hack_id, team_id) order by (hack_id, team_id);
```

| | hack_id<br>character varying | team_id<br>character varying | no_submissions<br>bigint |
|---|---|---|---|
| 135 | 40 | 53 | 1 |
| 136 | 41 | 14 | 1 |
| 137 | 41 | 47 | 2 |
| 138 | 42 | 24 | 1 |
| 139 | 43 | 12 | 1 |
| 140 | 43 | 14 | 1 |
| 141 | 43 | 29 | 1 |
| 142 | 43 | 44 | 1 |
| 143 | 44 | 16 | 1 |
| 144 | 44 | 18 | 1 |

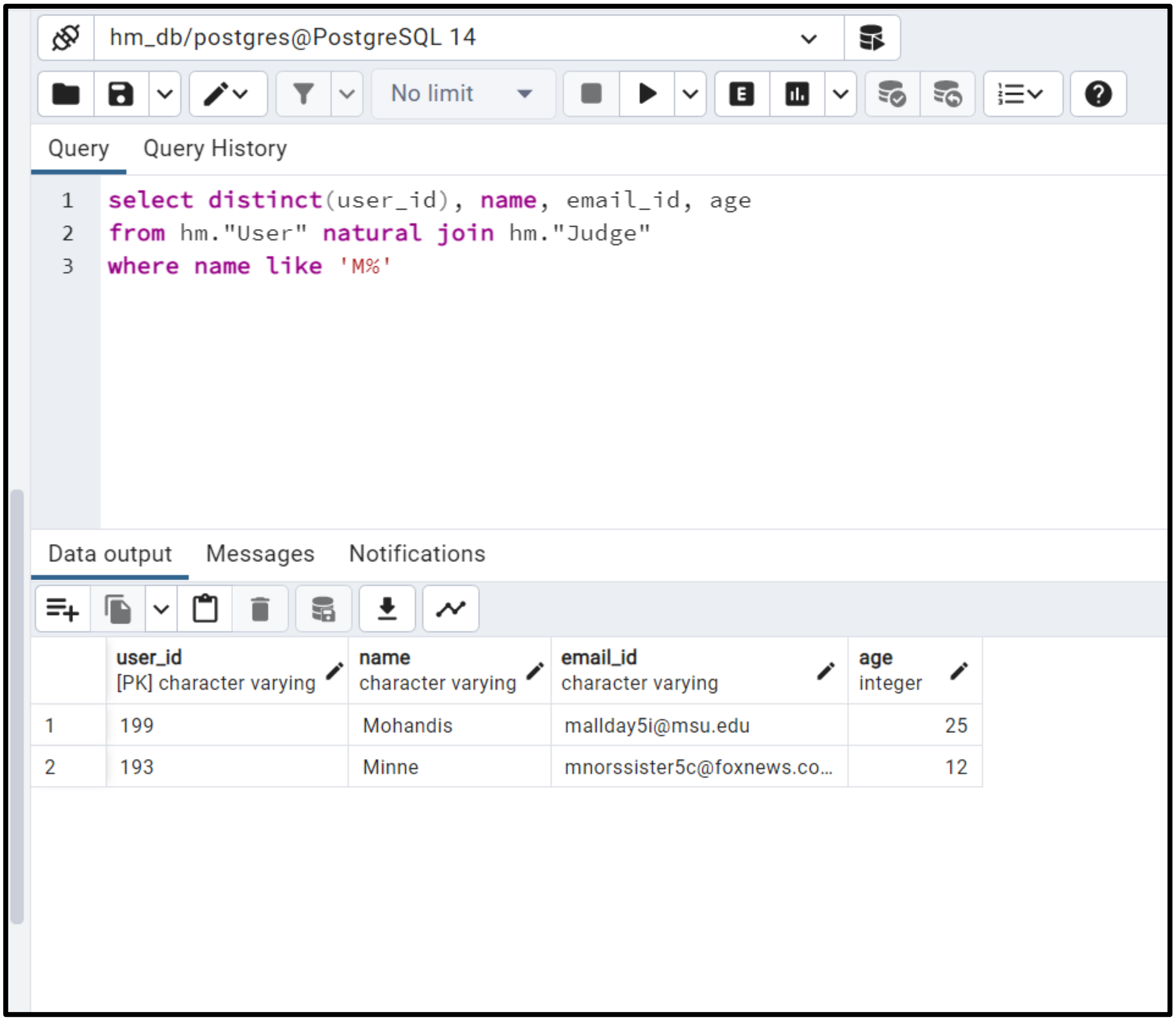
## 11) Show name and email address and user id of sponsors of hackathon - 4:



```
SELECT name,email_id,user_id
FROM User natural join (SELECT user_id FROM Sponsor natural join Hackathon WHERE hack_id = "4")
```

| | name<br>character varying | email_id<br>character varying | user_id<br>[PK] character varying |
|---|---|---|---|
| 1 | Farra | fsoutherns0@sal... | 1 |
| 2 | Nari | nmordue1@blog... | 2 |
| 3 | Roxane | rcheley2@mediaf... | 3 |
| 4 | Milton | mjadczak3@ihg.... | 4 |
| 5 | Karlene | kbristowe4@noa... | 5 |
| 6 | Jerrilee | jcarradice5@free... | 6 |
| 7 | Delmer | desser6@usda.g... | 7 |
| 8 | Trueman | titschakov7@co... | 8 |
| 9 | Dom | durridge8@about... | 9 |
| 10 | Dreddy | dalfwy9@tiny.cc | 10 |
| 11 | Haroun | hondricha@corn... | 11 |
| 12 | Maxine | mgroddenb@cre... | 12 |
| 13 | Miguela | maguirrezabalc@... | 13 |
| 14 | Mathias | mdefraind@berk... | 14 |
| 15 | Kriste | kbantone@grava... | 15 |
| 16 | Nels | neverissf@fda.gov | 16 |

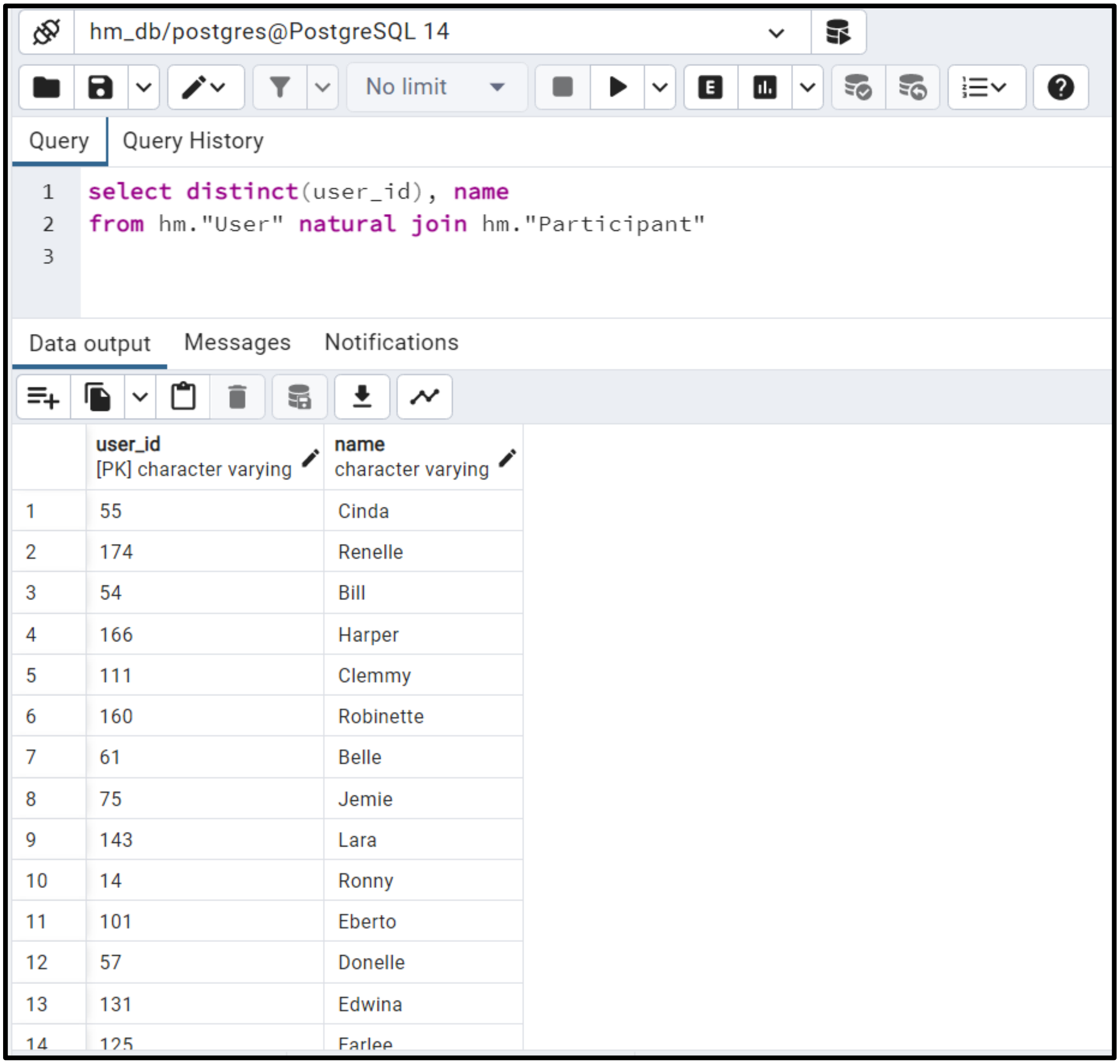
## 12) Find details of Judges whose name starts with letter M :



```
select distinct(user_id), name, email_id, age
from hm."User" natural join hm."Judge"
where name like 'M%'
```

| | user_id<br>[PK] character varying | name<br>character varying | email_id<br>character varying | age<br>integer |
|---|---|---|---|---|
| 1 | 199 | Mohandis | mallday5i@msu.edu | 25 |
| 2 | 193 | Minne | mnorssister5c@foxnews.co... | 12 |

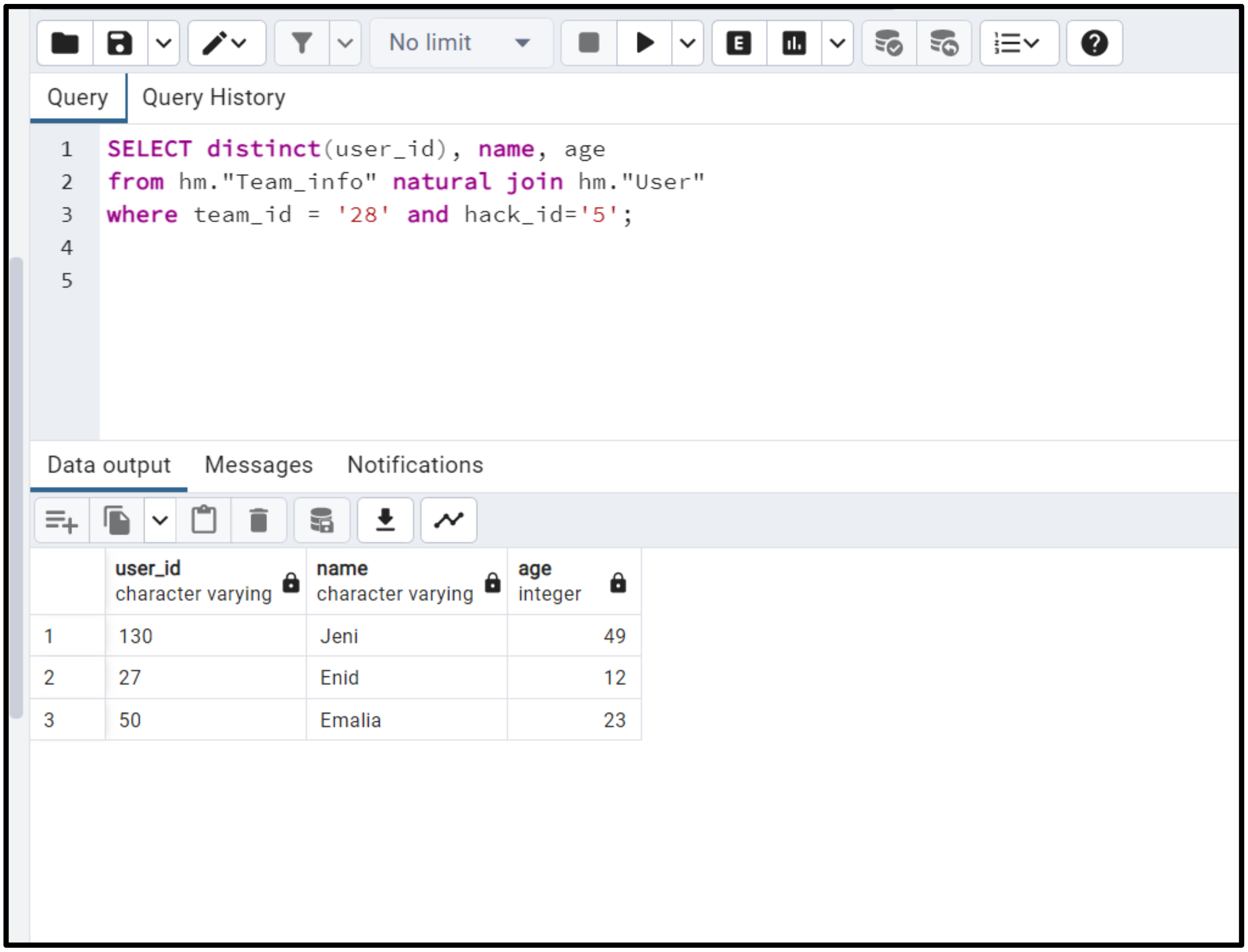
13) Show name and user_id of the users who have participated in Hackathon:



```
select distinct(user_id), name
from hm."User" natural join hm."Participant"
```

| | user_id<br>[PK] character varying | name<br>character varying |
|---|---|---|
| 1 | 55 | Cinda |
| 2 | 174 | Renelle |
| 3 | 54 | Bill |
| 4 | 166 | Harper |
| 5 | 111 | Clemmy |
| 6 | 160 | Robinette |
| 7 | 61 | Belle |
| 8 | 75 | Jemie |
| 9 | 143 | Lara |
| 10 | 14 | Ronny |
| 11 | 101 | Eberto |
| 12 | 57 | Donelle |
| 13 | 131 | Edwina |
| 14 | 125 | Farlee |

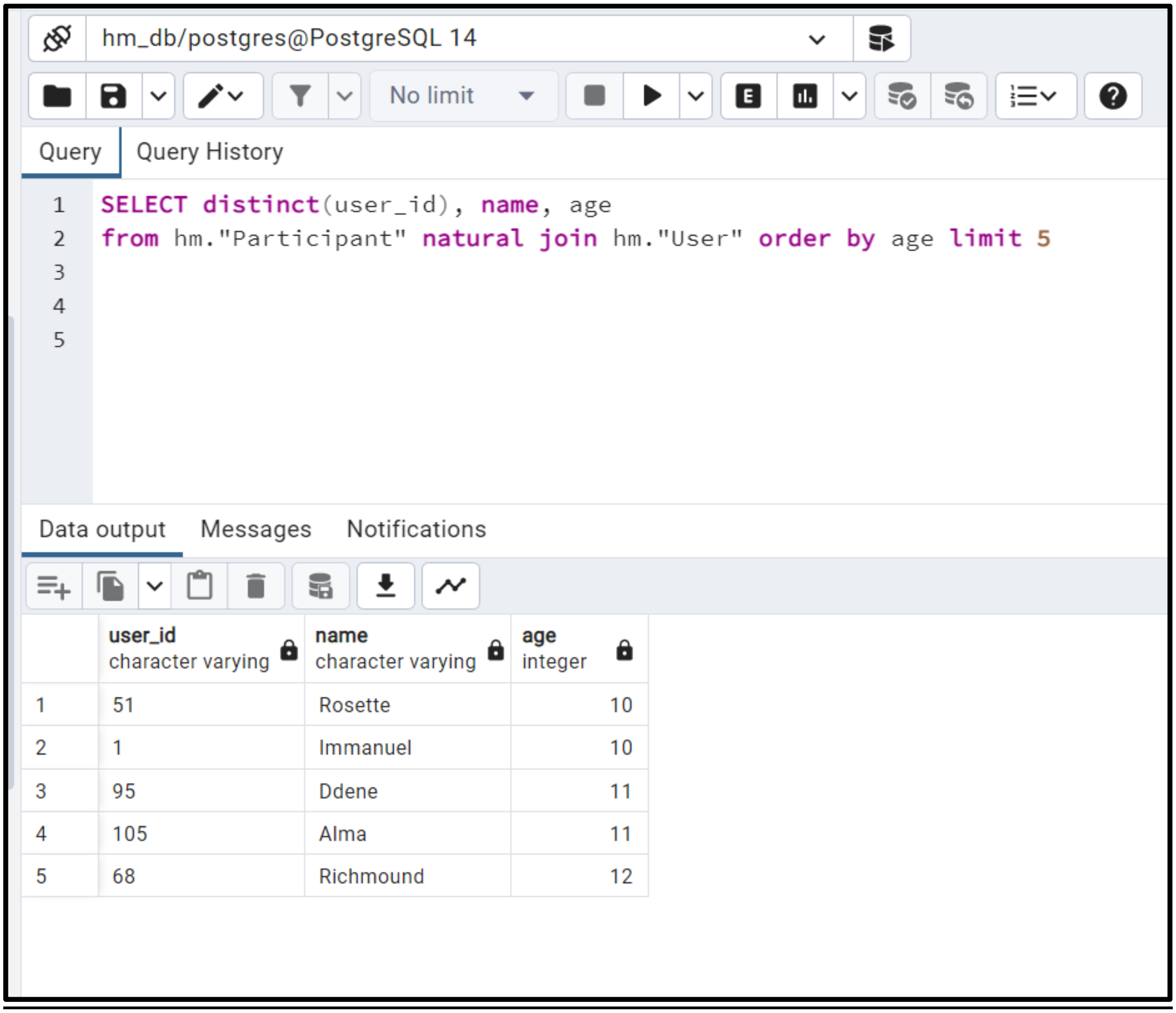
14) Show details of participants of hack_id 5 with team_id 28.



```
SELECT distinct(user_id), name, age
from hm."Team_info" natural join hm."User"
where team_id = '28' and hack_id='5';
```

| | user_id<br>character varying | name<br>character varying | age<br>integer |
|---|---|---|---|
| 1 | 130 | Jeni | 49 |
| 2 | 27 | Enid | 12 |
| 3 | 50 | Emalia | 23 |

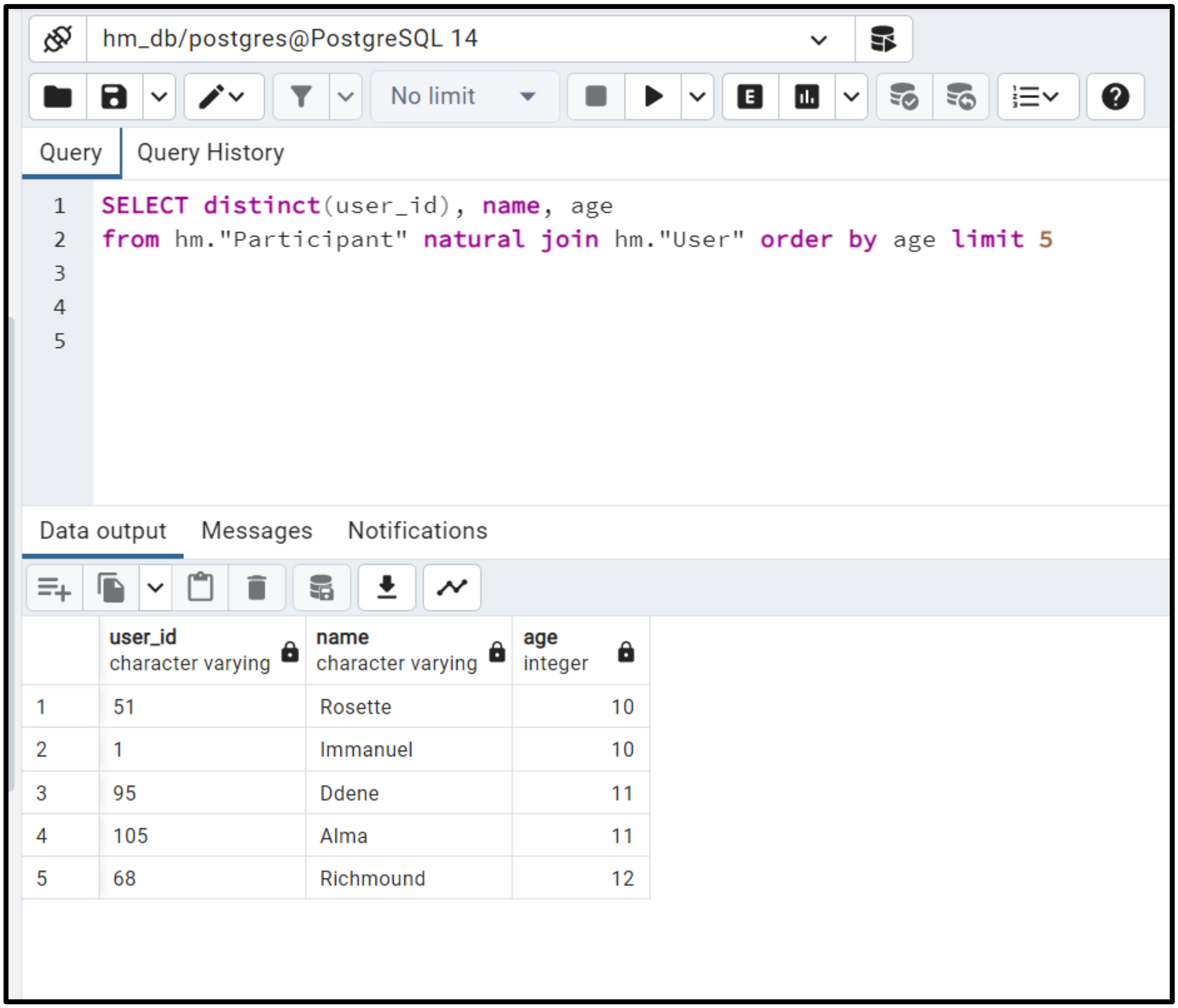
## 15) Show details of the first 5 youngest participants.



```
SELECT distinct(user_id), name, age
from hm."Participant" natural join hm."User" order by age limit 5
```

| | user_id<br>character varying | name<br>character varying | age<br>integer |
|---|---|---|---|
| 1 | 51 | Rosette | 10 |
| 2 | 1 | Immanuel | 10 |
| 3 | 95 | Ddene | 11 |
| 4 | 105 | Alma | 11 |
| 5 | 68 | Richmound | 12 |

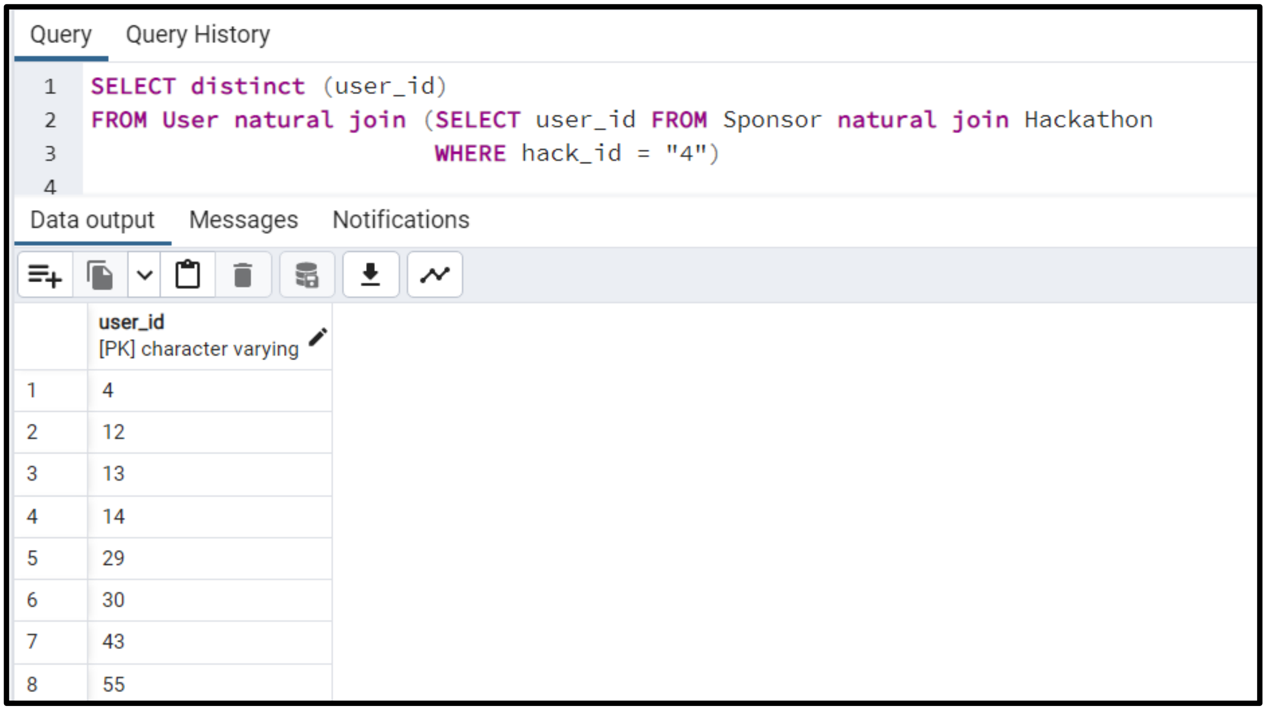
16) Show details of the first 5 youngest participants.



```
SELECT distinct(user_id), name, age
from hm."Participant" natural join hm."User" order by age limit 5
```

| | user_id<br>character varying | name<br>character varying | age<br>integer |
|---|---|---|---|
| 1 | 51 | Rosette | 10 |
| 2 | 1 | Immanuel | 10 |
| 3 | 95 | Ddene | 11 |
| 4 | 105 | Alma | 11 |
| 5 | 68 | Richmound | 12 |

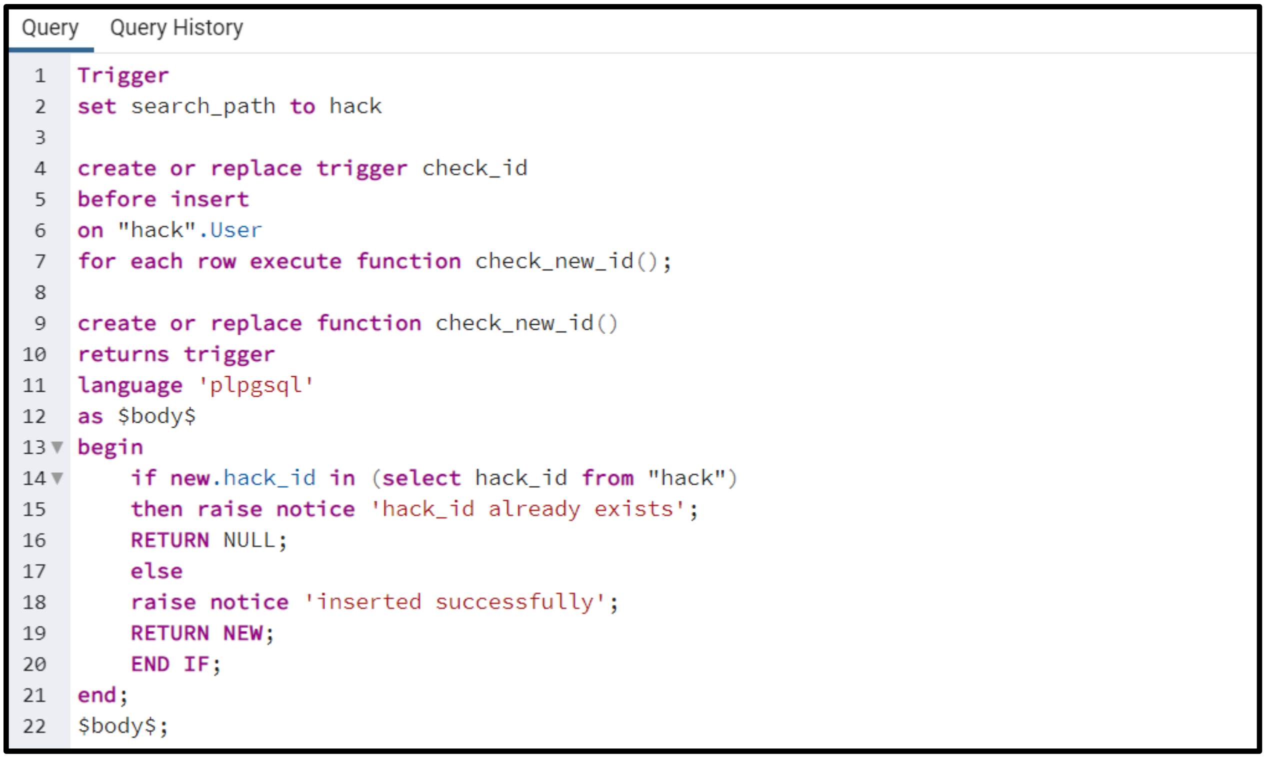
## 17) Show user_id of sponsors from hack_ id = 4:



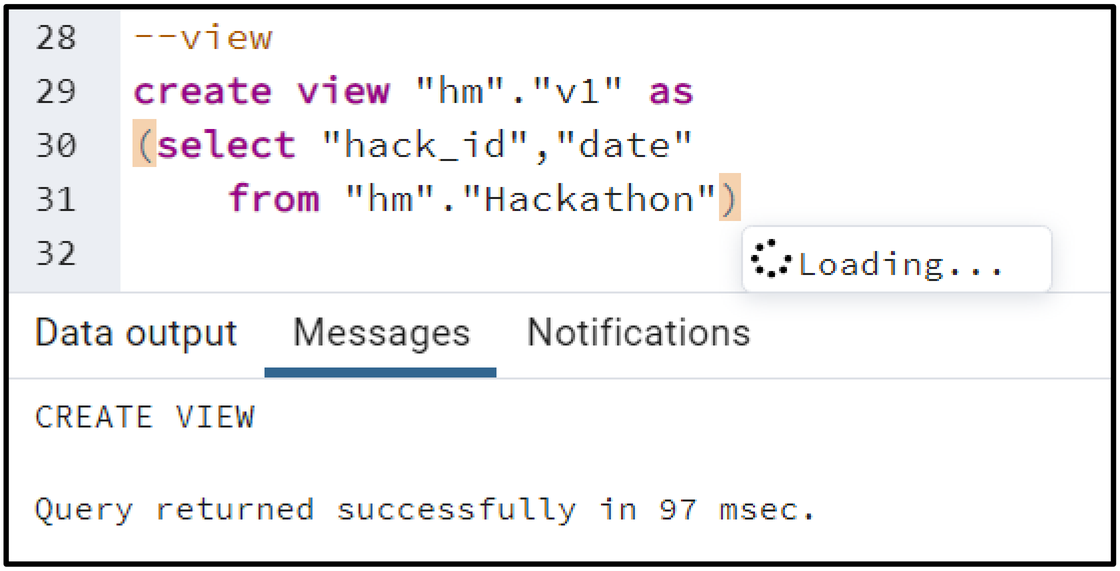
```
SELECT distinct (user_id)
FROM User natural join (SELECT user_id FROM Sponsor natural join Hackathon
                        WHERE hack_id = "4")

```

| | user_id<br>[PK] character varying |
|---|---|
| 1 | 4 |
| 2 | 12 |
| 3 | 13 |
| 4 | 14 |
| 5 | 29 |
| 6 | 30 |
| 7 | 43 |
| 8 | 55 |

## 18) Trigger for Hack_id:

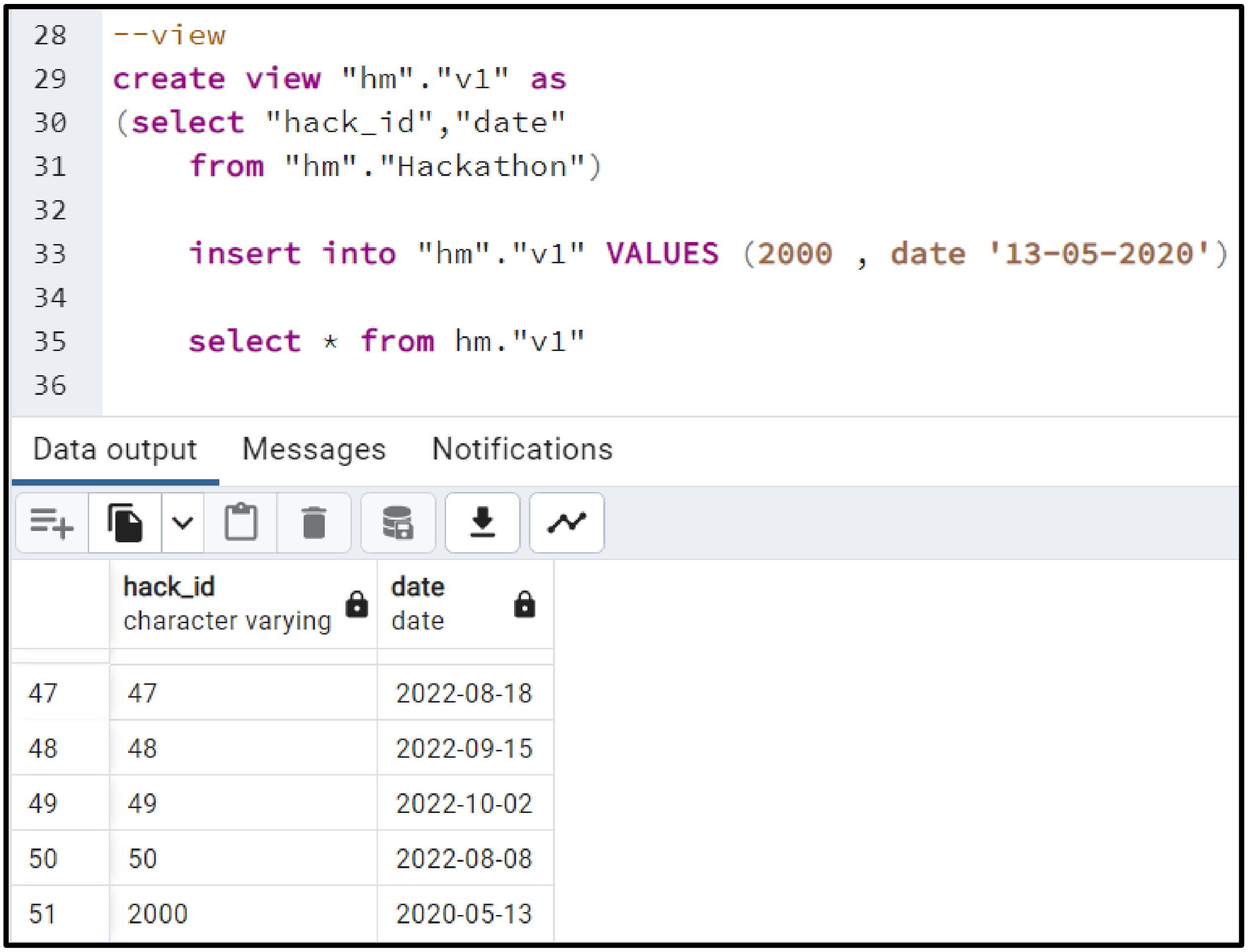


```
Trigger
set search_path to hack

create or replace trigger check_id
before insert
on "hack".User
for each row execute function check_new_id();

create or replace function check_new_id()
returns trigger
language 'plpgsql'
as $body$
begin
    if new.hack_id in (select hack_id from "hack")
    then raise notice 'hack_id already exists';
    RETURN NULL;
    else
    raise notice 'inserted successfully';
    RETURN NEW;
    END IF;
end;
$body$;
```



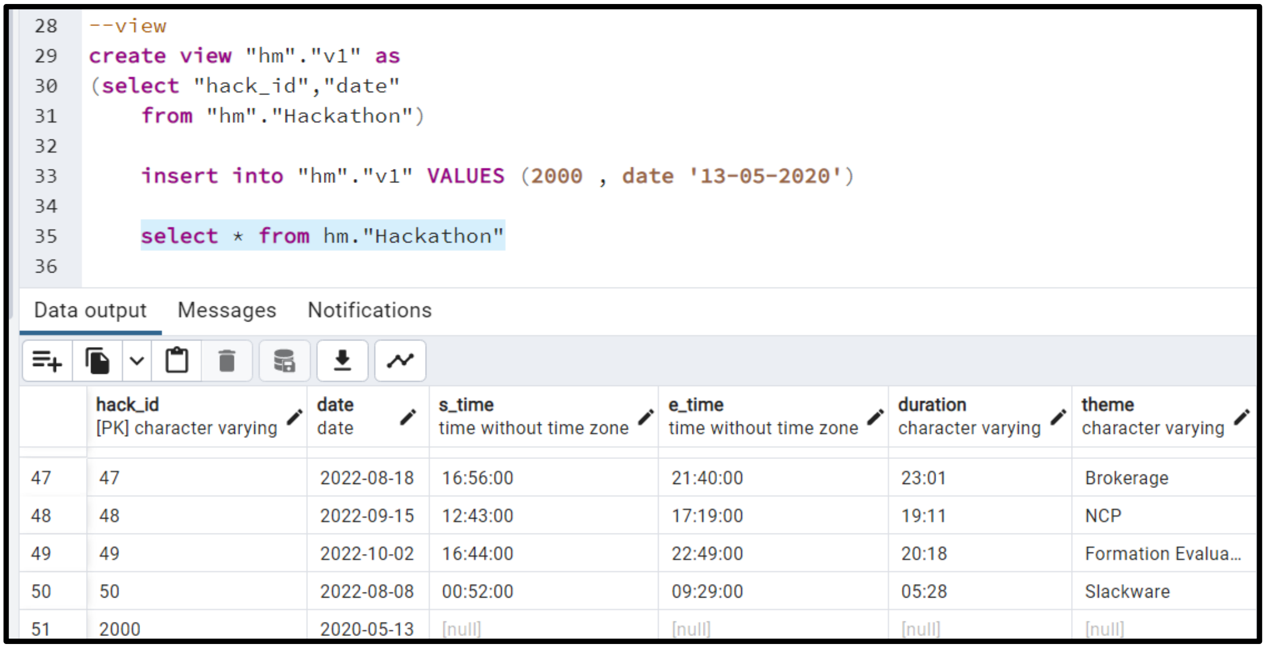
```
--view
create view "hm"."v1" as
(select "hack_id","date"
    from "hm"."Hackathon")
```

```
CREATE VIEW

Query returned successfully in 97 msec.
```



```
--view
create view "hm"."v1" as
(select "hack_id","date"
    from "hm"."Hackathon")

    insert into "hm"."v1" VALUES (2000 , date '13-05-2020')

    select * from hm."v1"

```

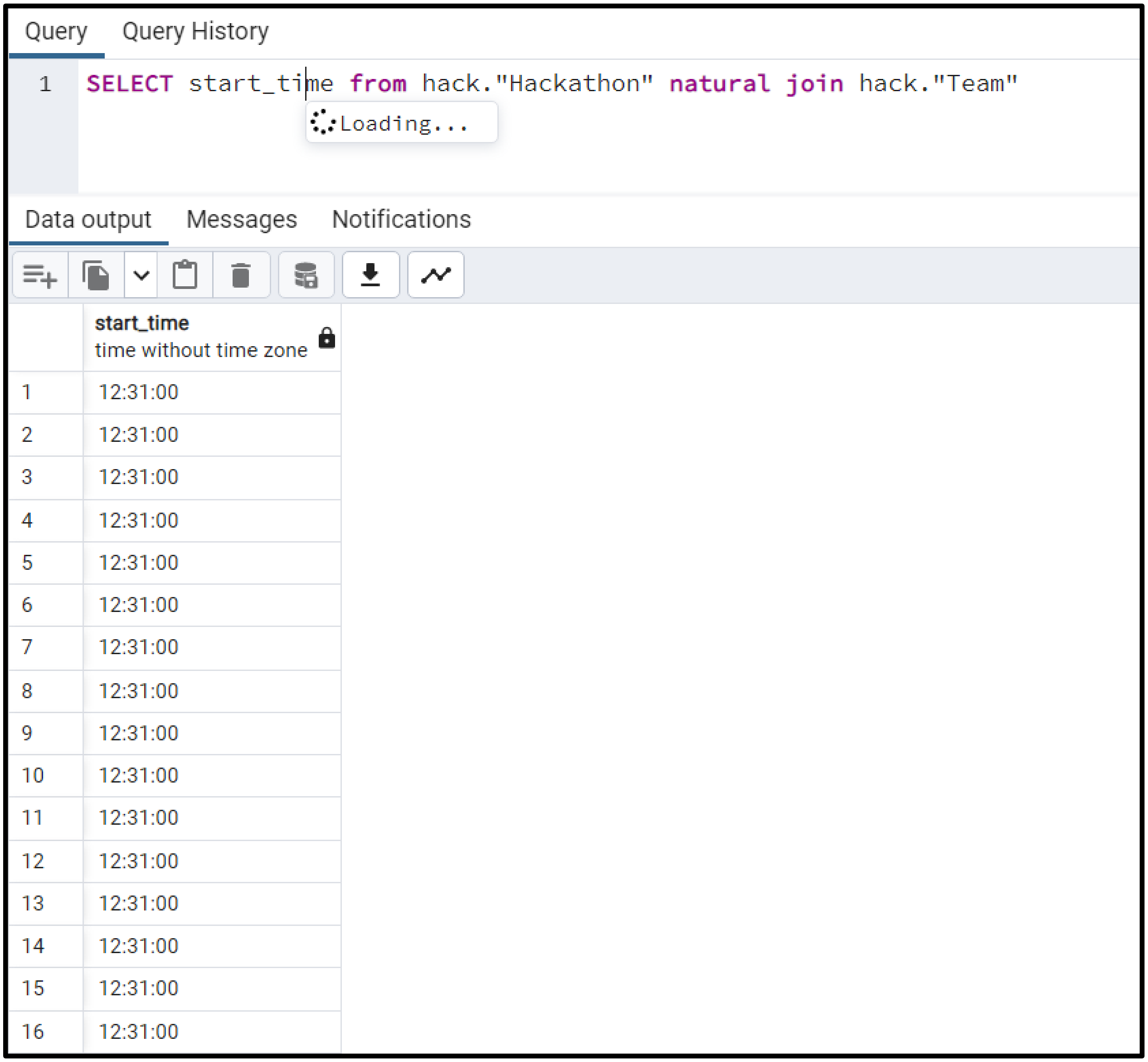
| | hack_id<br>character varying | date<br>date |
|---|---|---|
| 47 | 47 | 2022-08-18 |
| 48 | 48 | 2022-09-15 |
| 49 | 49 | 2022-10-02 |
| 50 | 50 | 2022-08-08 |
| 51 | 2000 | 2020-05-13 |



```
--view
create view "hm"."v1" as
(select "hack_id","date"
    from "hm"."Hackathon")

    insert into "hm"."v1" VALUES (2000 , date '13-05-2020')

    select * from hm."Hackathon"

```

| | hack_id<br>[PK] character varying | date<br>date | s_time<br>time without time zone | e_time<br>time without time zone | duration<br>character varying | theme<br>character varying |
|---|---|---|---|---|---|---|
| 47 | 47 | 2022-08-18 | 16:56:00 | 21:40:00 | 23:01 | Brokerage |
| 48 | 48 | 2022-09-15 | 12:43:00 | 17:19:00 | 19:11 | NCP |
| 49 | 49 | 2022-10-02 | 16:44:00 | 22:49:00 | 20:18 | Formation Evalua... |
| 50 | 50 | 2022-08-08 | 00:52:00 | 09:29:00 | 05:28 | Slackware |
| 51 | 2000 | 2020-05-13 | [null] | [null] | [null] | [null] |

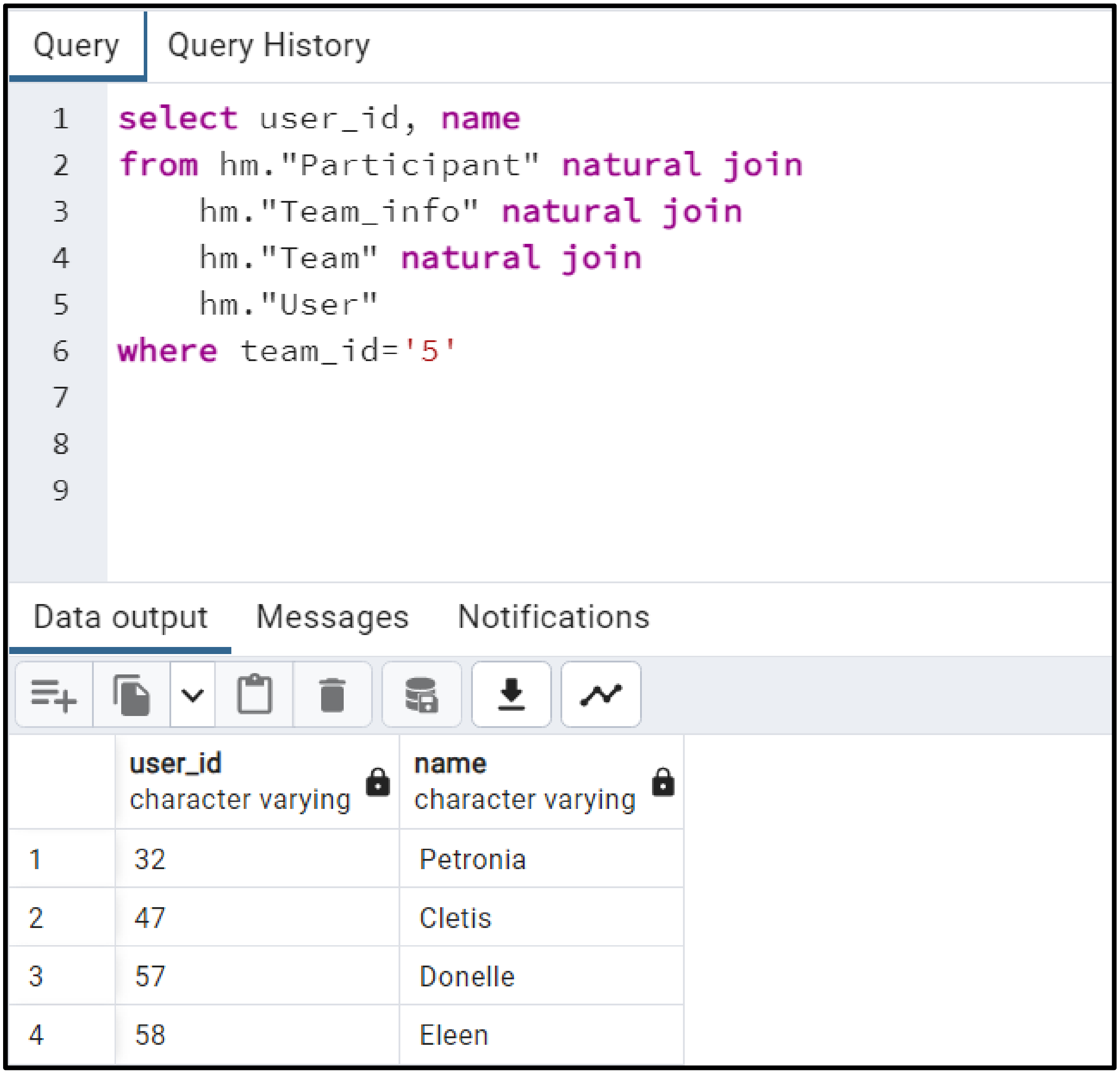
## 19) Show start time of different teams from Hackathon.



```sql
SELECT start_time from hack."Hackathon" natural join hack."Team"
```

| | start_time<br>time without time zone |
|---|---|
| 1 | 12:31:00 |
| 2 | 12:31:00 |
| 3 | 12:31:00 |
| 4 | 12:31:00 |
| 5 | 12:31:00 |
| 6 | 12:31:00 |
| 7 | 12:31:00 |
| 8 | 12:31:00 |
| 9 | 12:31:00 |
| 10 | 12:31:00 |
| 11 | 12:31:00 |
| 12 | 12:31:00 |
| 13 | 12:31:00 |
| 14 | 12:31:00 |
| 15 | 12:31:00 |
| 16 | 12:31:00 |

20)Find details of participants whose team id is 5 of every Hackathon.

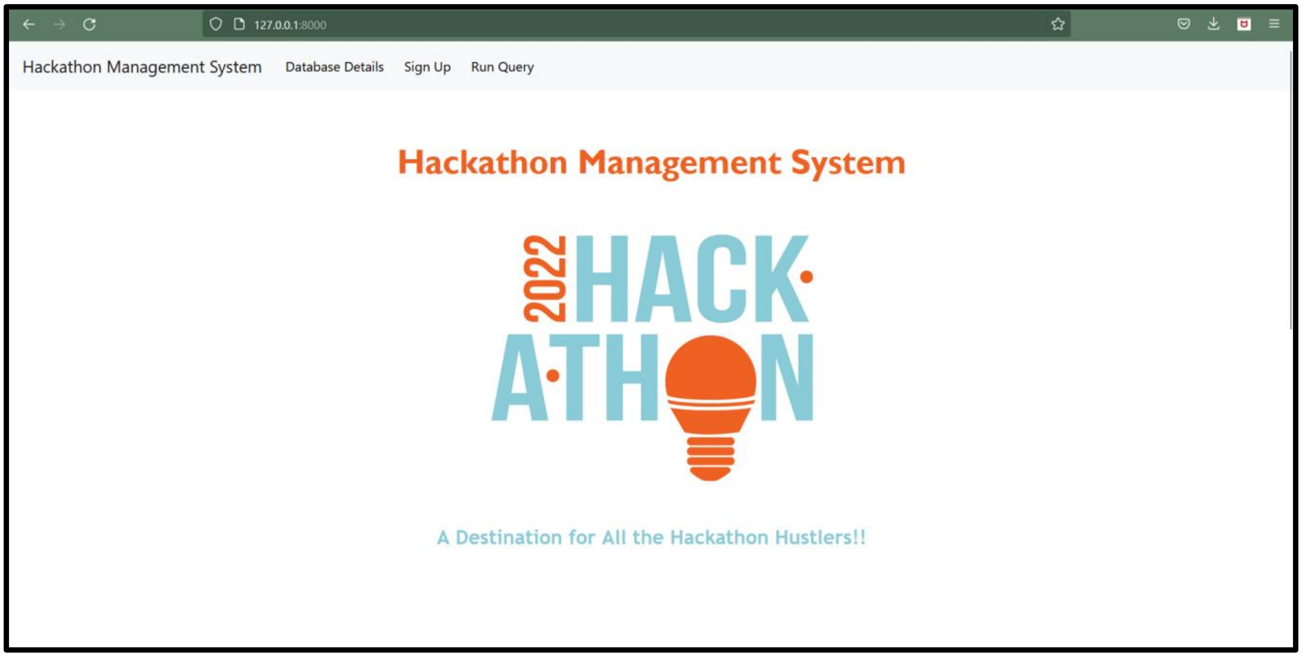


```sql
select user_id, name
from hm."Participant" natural join
     hm."Team_info" natural join
     hm."Team" natural join
     hm."User"
where team_id='5'
```

| | user_id<br>character varying | name<br>character varying |
|---|---|---|
| 1 | 32 | Petronia |
| 2 | 47 | Cletis |
| 3 | 57 | Donelle |
| 4 | 58 | Eleen |

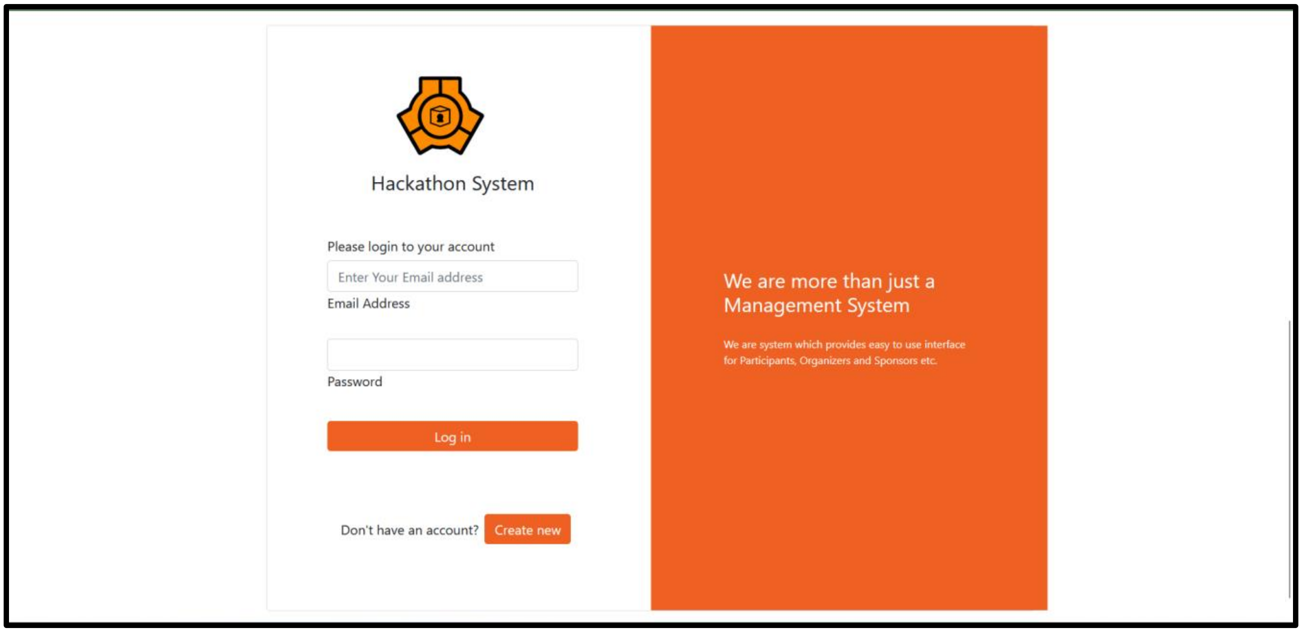
# Website Work

- Home page with navigation bar and log-in page.



Hackathon Management System

A Destination for All the Hackathon Hustlers!!



Please login to your account

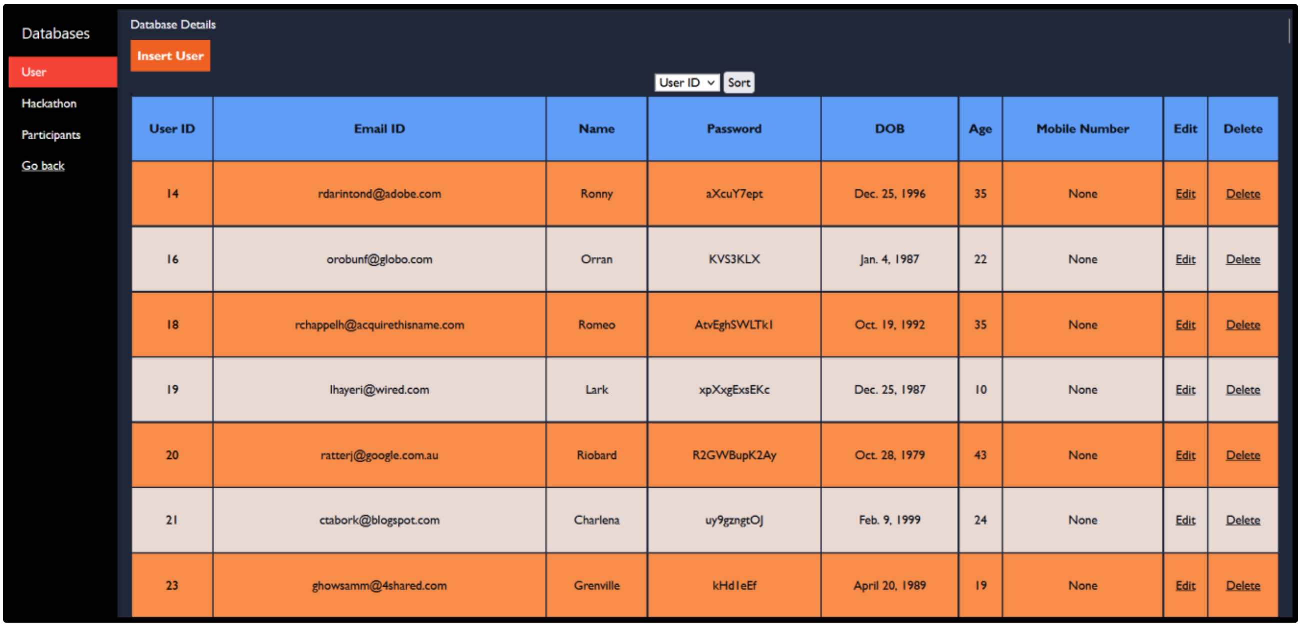
Don't have an account?

We are more than just a Management System

We are system which provides easy to use interface for Participants, Organizers and Sponsors etc.

## ❖ Database details:

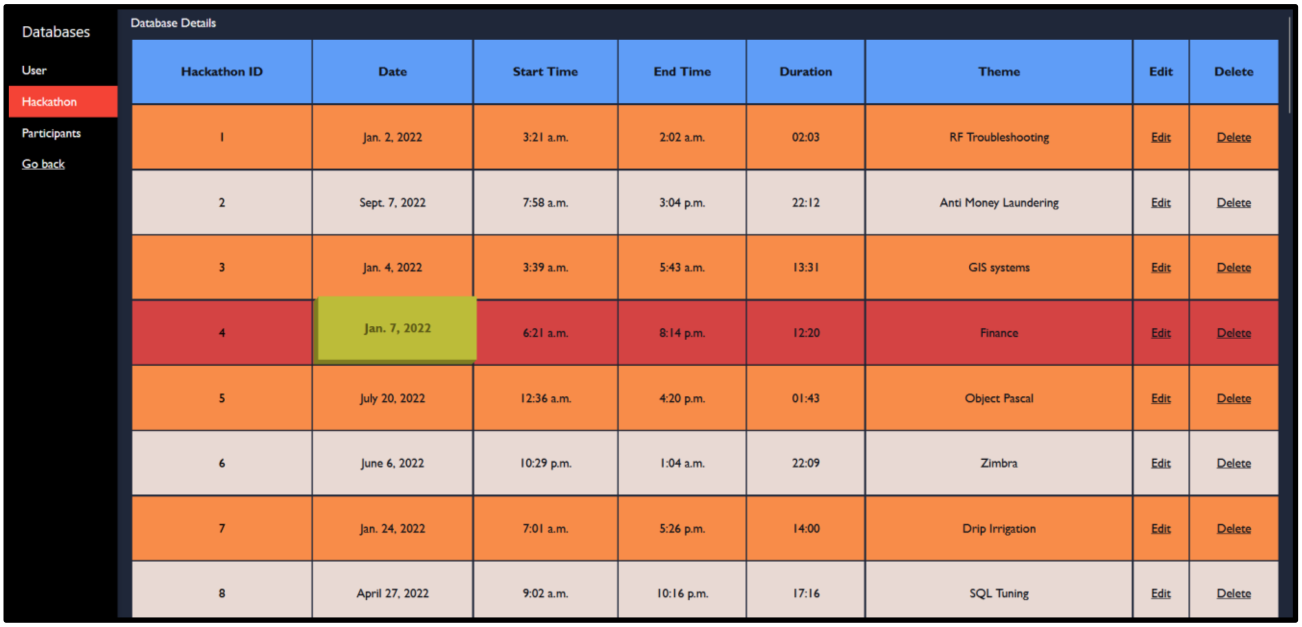
- User details:



Database Details

| User ID | Email ID | Name | Password | DOB | Age | Mobile Number | Edit | Delete |
|---|---|---|---|---|---|---|---|---|
| 14 | rdarintond@adobe.com | Ronny | aXcuY7ept | Dec. 25, 1996 | 35 | None | Edit | Delete |
| 16 | orobunf@globo.com | Orran | KVS3KLX | Jan. 4, 1987 | 22 | None | Edit | Delete |
| 18 | rchappelh@acquirethisname.com | Romeo | AtvEghSWLTk1 | Oct. 19, 1992 | 35 | None | Edit | Delete |
| 19 | lhayeri@wired.com | Lark | xpXxgExsEKc | Dec. 25, 1987 | 10 | None | Edit | Delete |
| 20 | ratterj@google.com.au | Riobard | R2GWBupK2Ay | Oct. 28, 1979 | 43 | None | Edit | Delete |
| 21 | ctabork@blogspot.com | Charlena | uy9gzngtOJ | Feb. 9, 1999 | 24 | None | Edit | Delete |
| 23 | ghowsamm@4shared.com | Grenville | kHd1eEf | April 20, 1989 | 19 | None | Edit | Delete |

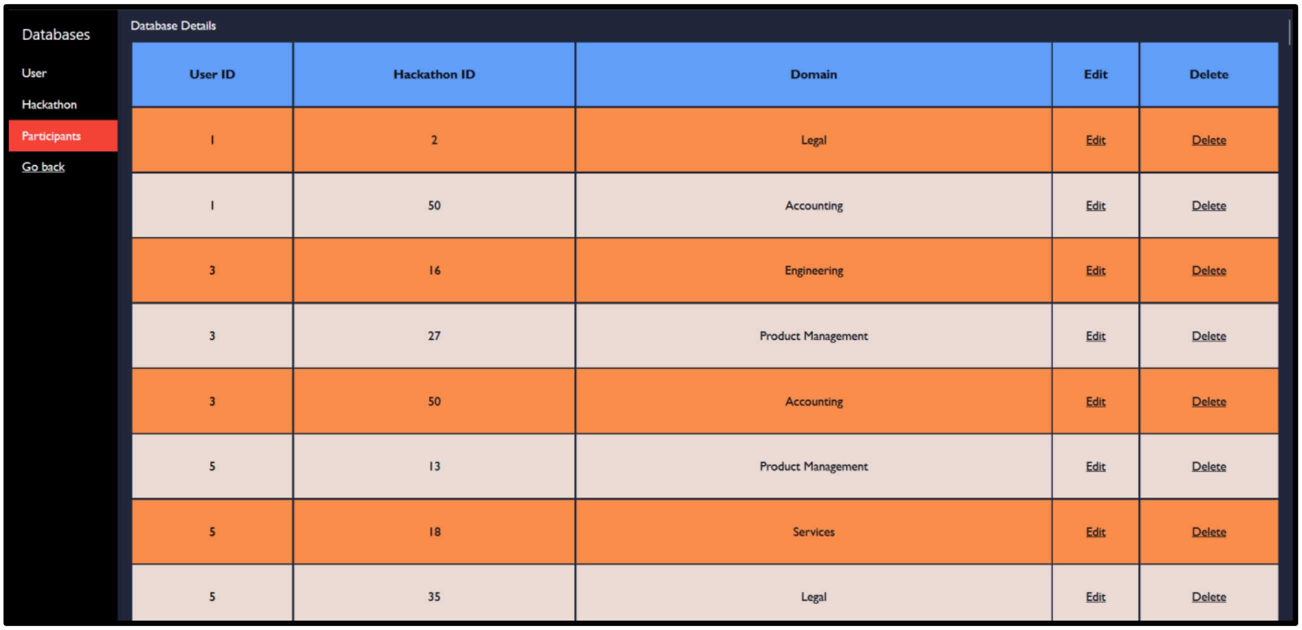
- Hackathon details:



Database Details

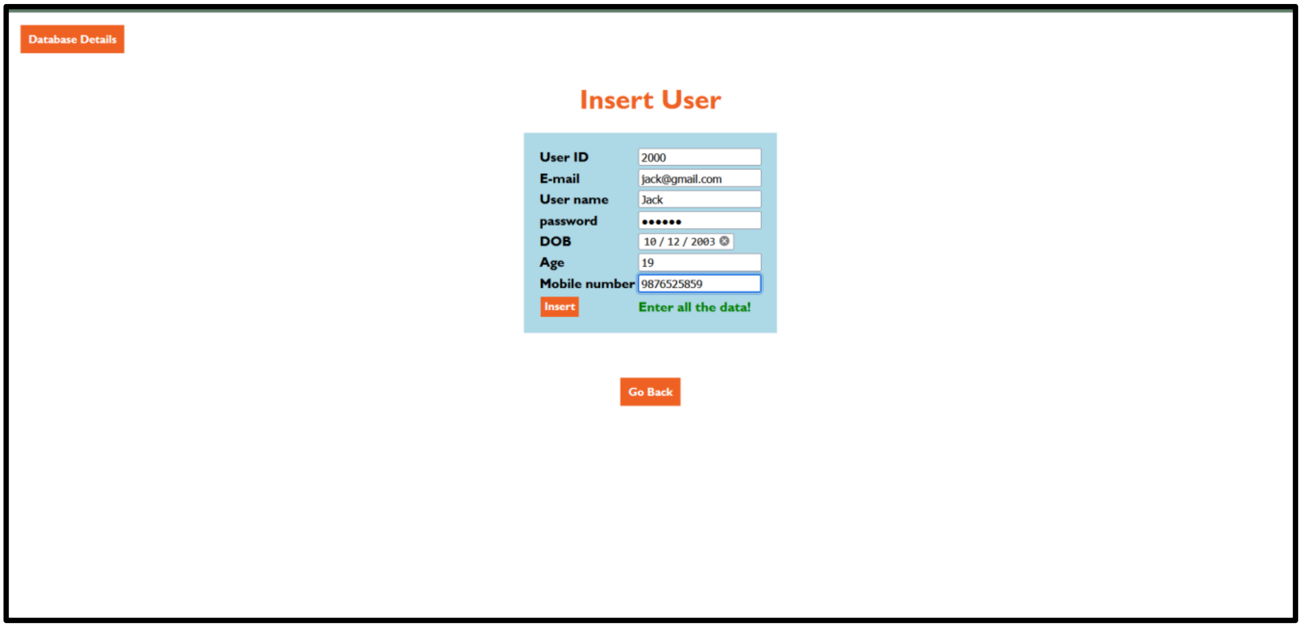
| Hackathon ID | Date | Start Time | End Time | Duration | Theme | Edit | Delete |
|---|---|---|---|---|---|---|---|
| 1 | Jan. 2, 2022 | 3:21 a.m. | 2:02 a.m. | 02:03 | RF Troubleshooting | Edit | Delete |
| 2 | Sept. 7, 2022 | 7:58 a.m. | 3:04 p.m. | 22:12 | Anti Money Laundering | Edit | Delete |
| 3 | Jan. 4, 2022 | 3:39 a.m. | 5:43 a.m. | 13:31 | GIS systems | Edit | Delete |
| 4 | Jan. 7, 2022 | 6:21 a.m. | 8:14 p.m. | 12:20 | Finance | Edit | Delete |
| 5 | July 20, 2022 | 12:36 a.m. | 4:20 p.m. | 01:43 | Object Pascal | Edit | Delete |
| 6 | June 6, 2022 | 10:29 p.m. | 1:04 a.m. | 22:09 | Zimbra | Edit | Delete |
| 7 | Jan. 24, 2022 | 7:01 a.m. | 5:26 p.m. | 14:00 | Drip Irrigation | Edit | Delete |
| 8 | April 27, 2022 | 9:02 a.m. | 10:16 p.m. | 17:16 | SQL Tuning | Edit | Delete |

- Participant details:



| User ID | Hackathon ID | Domain | Edit | Delete |
|---|---|---|---|---|
| 1 | 2 | Legal | Edit | Delete |
| 1 | 50 | Accounting | Edit | Delete |
| 3 | 16 | Engineering | Edit | Delete |
| 3 | 27 | Product Management | Edit | Delete |
| 3 | 50 | Accounting | Edit | Delete |
| 5 | 13 | Product Management | Edit | Delete |
| 5 | 18 | Services | Edit | Delete |
| 5 | 35 | Legal | Edit | Delete |

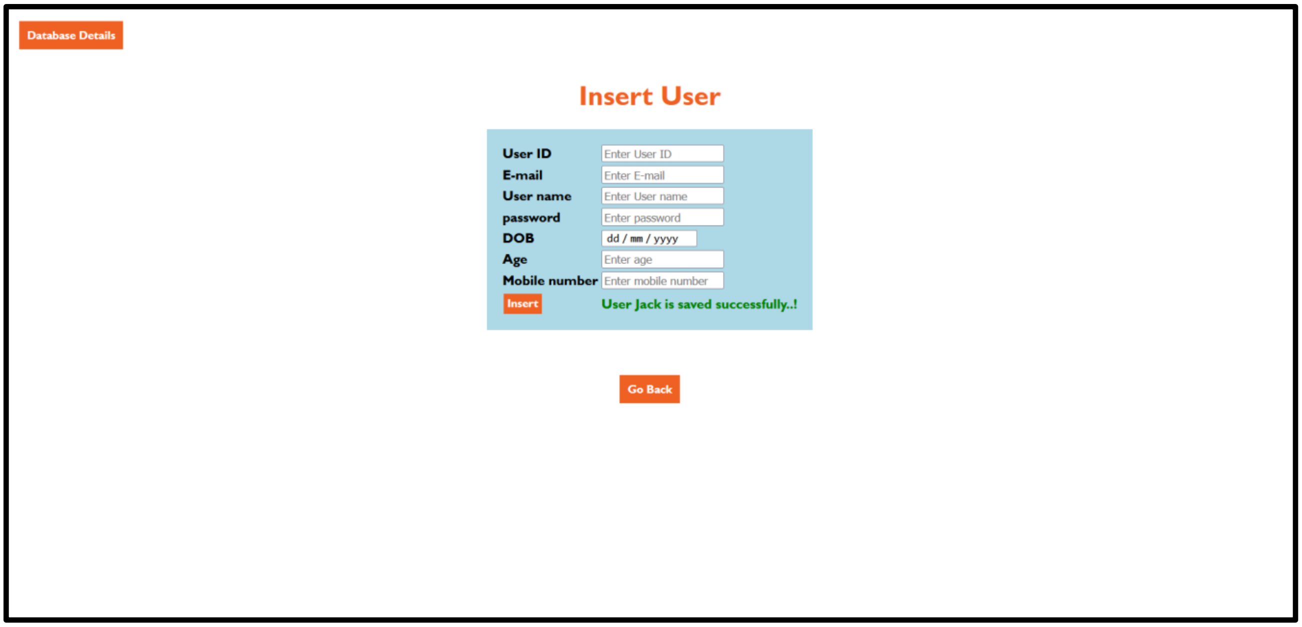
❖ Insert new user:



Insert User

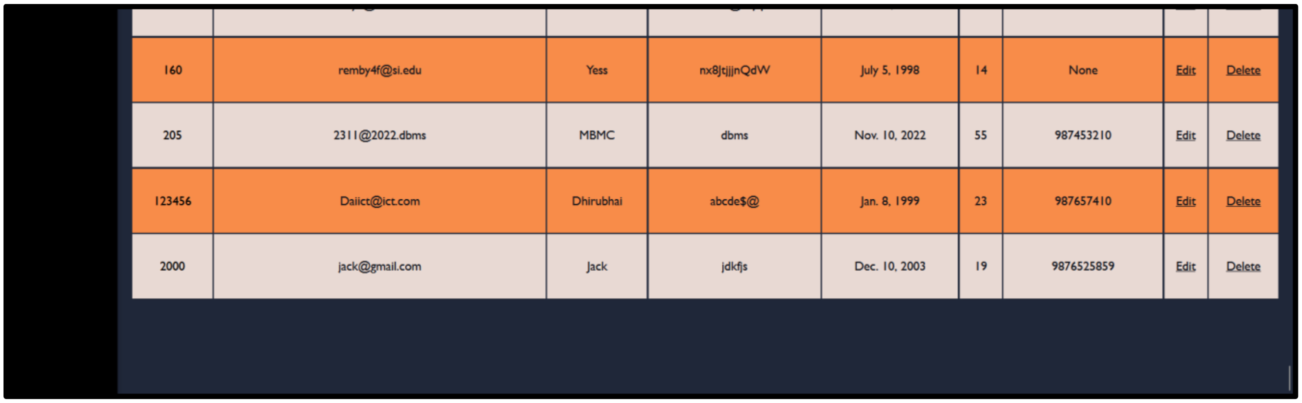
| | |
|---|---|
| User ID | 2000 |
| E-mail | jack@gmail.com |
| User name | Jack |
| password | •••••• |
| DOB | 10 / 12 / 2003 |
| Age | 19 |
| Mobile number | 9876525859 |

Enter all the data!



Insert User

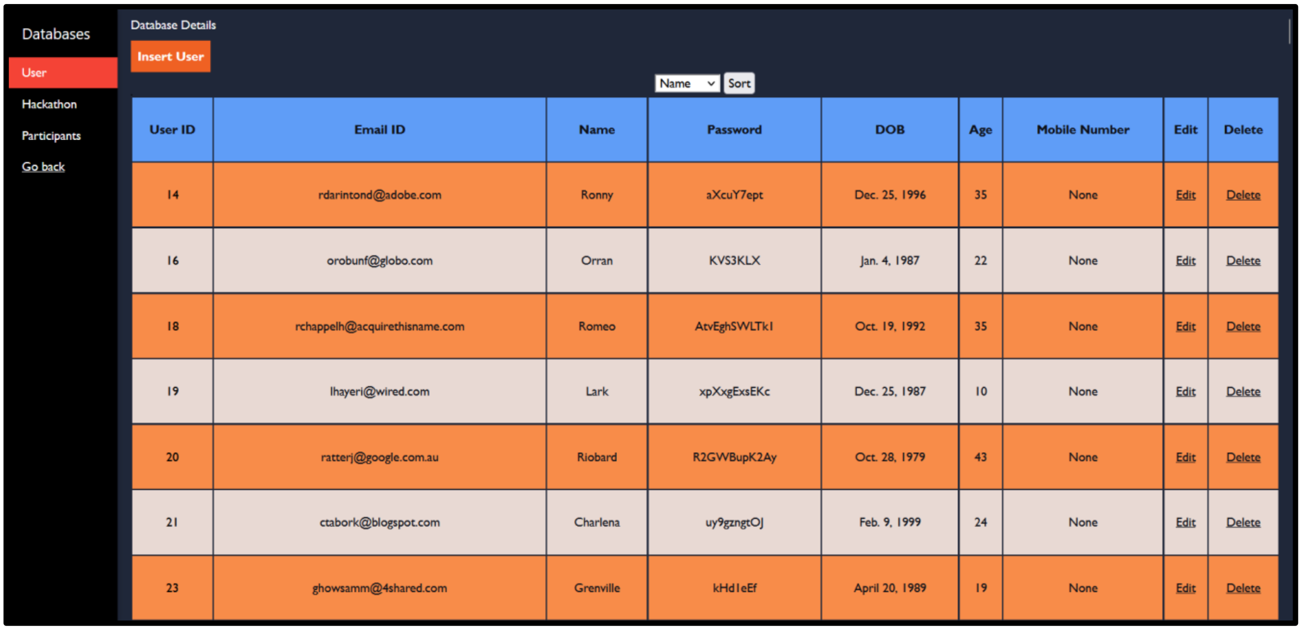
| | |
|---|---|
| User ID | Enter User ID |
| E-mail | Enter E-mail |
| User name | Enter User name |
| password | Enter password |
| DOB | dd / mm / yyyy |
| Age | Enter age |
| Mobile number | Enter mobile number |
| Insert | User Jack is saved successfully..! |



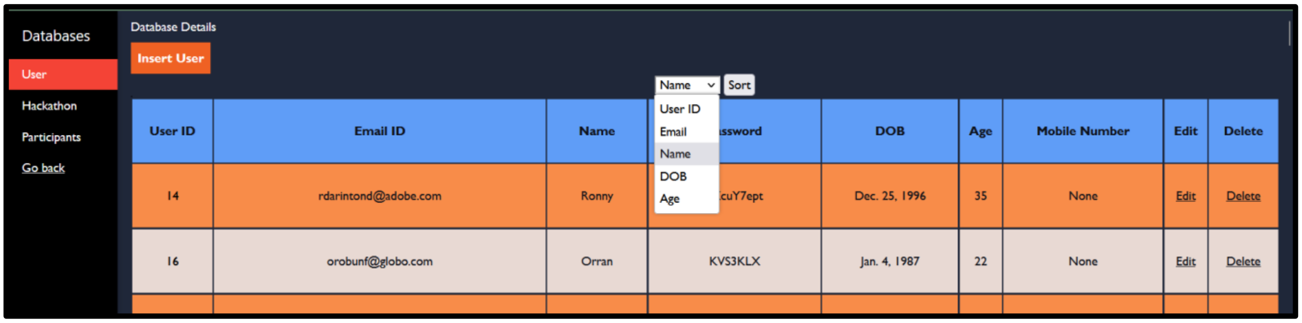
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 160 | remby4f@si.edu | Yess | nx8JtjjjnQdW | July 5, 1998 | 14 | None | Edit | Delete |
| 205 | 2311@2022.dbms | MBMC | dbms | Nov. 10, 2022 | 55 | 987453210 | Edit | Delete |
| 123456 | Daiict@ict.com | Dhirubhai | abcde$@ | Jan. 8, 1999 | 23 | 987657410 | Edit | Delete |
| 2000 | jack@gmail.com | Jack | jdkfjs | Dec. 10, 2003 | 19 | 9876525859 | Edit | Delete |

## ❖ Sort the details:

- Sort the details of the user record according to name.

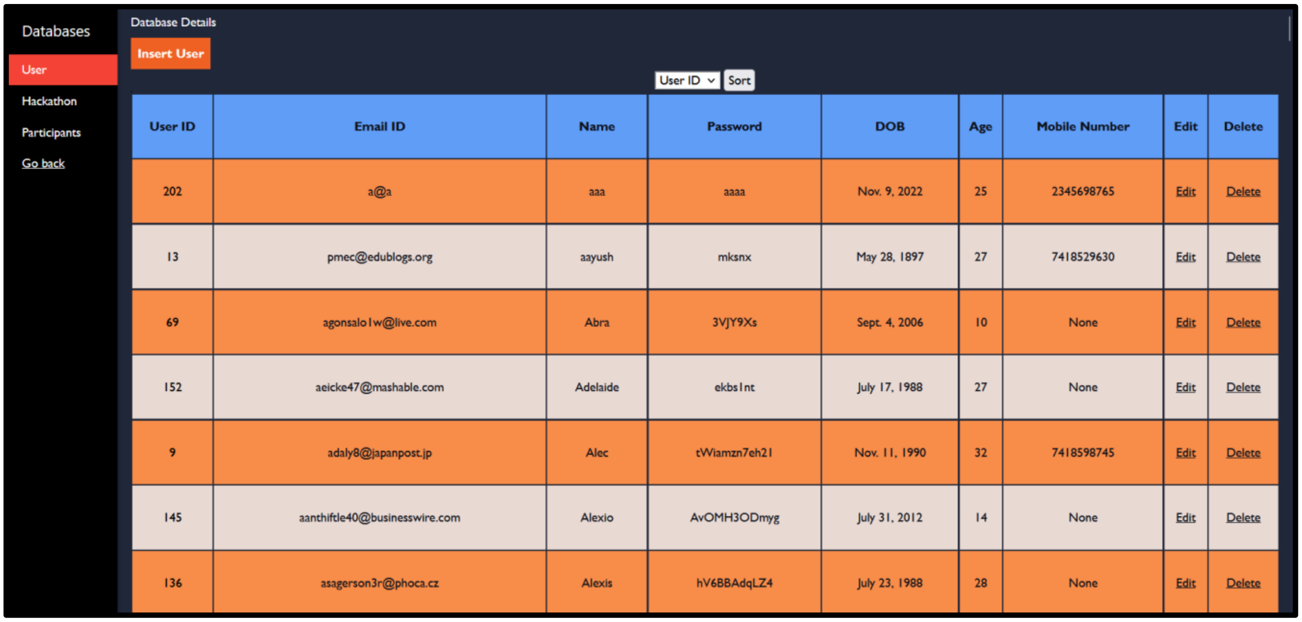


| User ID | Email ID | Name | Password | DOB | Age | Mobile Number | Edit | Delete |
|---|---|---|---|---|---|---|---|---|
| 14 | rdarintond@adobe.com | Ronny | aXcuY7ept | Dec. 25, 1996 | 35 | None | Edit | Delete |
| 16 | orobunf@globo.com | Orran | KVS3KLX | Jan. 4, 1987 | 22 | None | Edit | Delete |
| 18 | rchappelh@acquirethisname.com | Romeo | AtvEghSWLTk1 | Oct. 19, 1992 | 35 | None | Edit | Delete |
| 19 | lhayeri@wired.com | Lark | xpXxgExsEKc | Dec. 25, 1987 | 10 | None | Edit | Delete |
| 20 | ratterj@google.com.au | Riobard | R2GWBupK2Ay | Oct. 28, 1979 | 43 | None | Edit | Delete |
| 21 | ctabork@blogspot.com | Charlena | uy9gzngtOJ | Feb. 9, 1999 | 24 | None | Edit | Delete |
| 23 | ghowsamm@4shared.com | Grenville | kHd1eEf | April 20, 1989 | 19 | None | Edit | Delete |



| User ID | Email ID | Name | ssword | DOB | Age | Mobile Number | Edit | Delete |
|---|---|---|---|---|---|---|---|---|
| 14 | rdarintond@adobe.com | Ronny | cuY7ept | Dec. 25, 1996 | 35 | None | Edit | Delete |
| 16 | orobunf@globo.com | Orran | KVS3KLX | Jan. 4, 1987 | 22 | None | Edit | Delete |

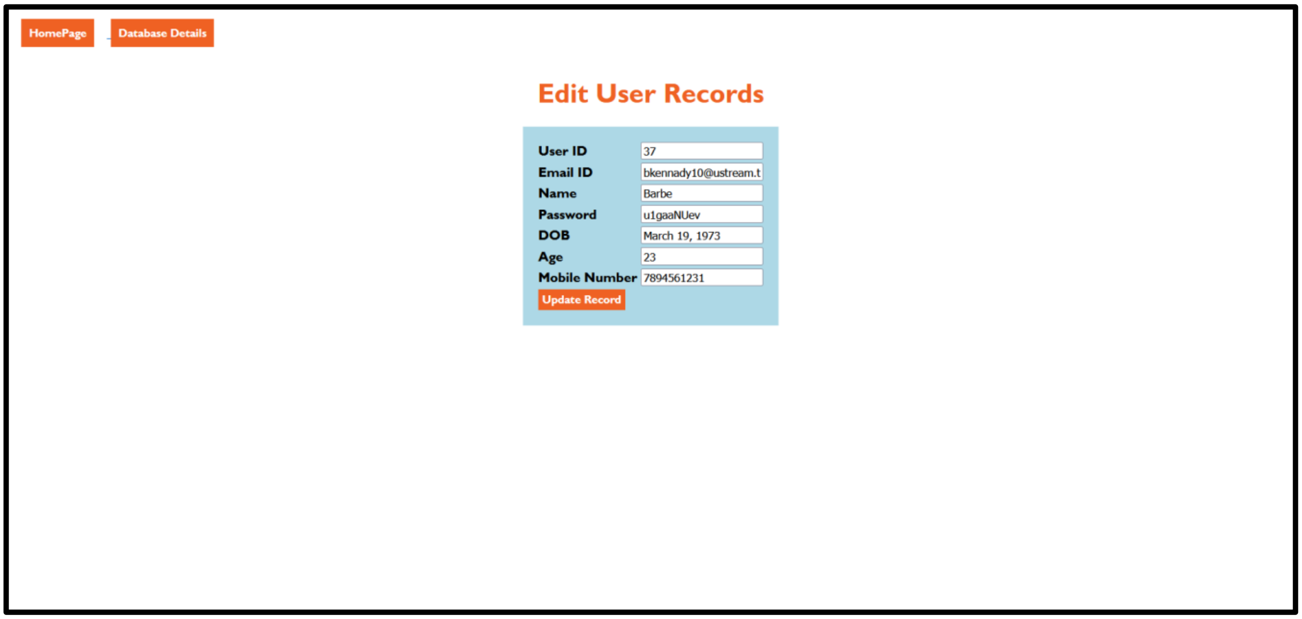
5.11



Database Details

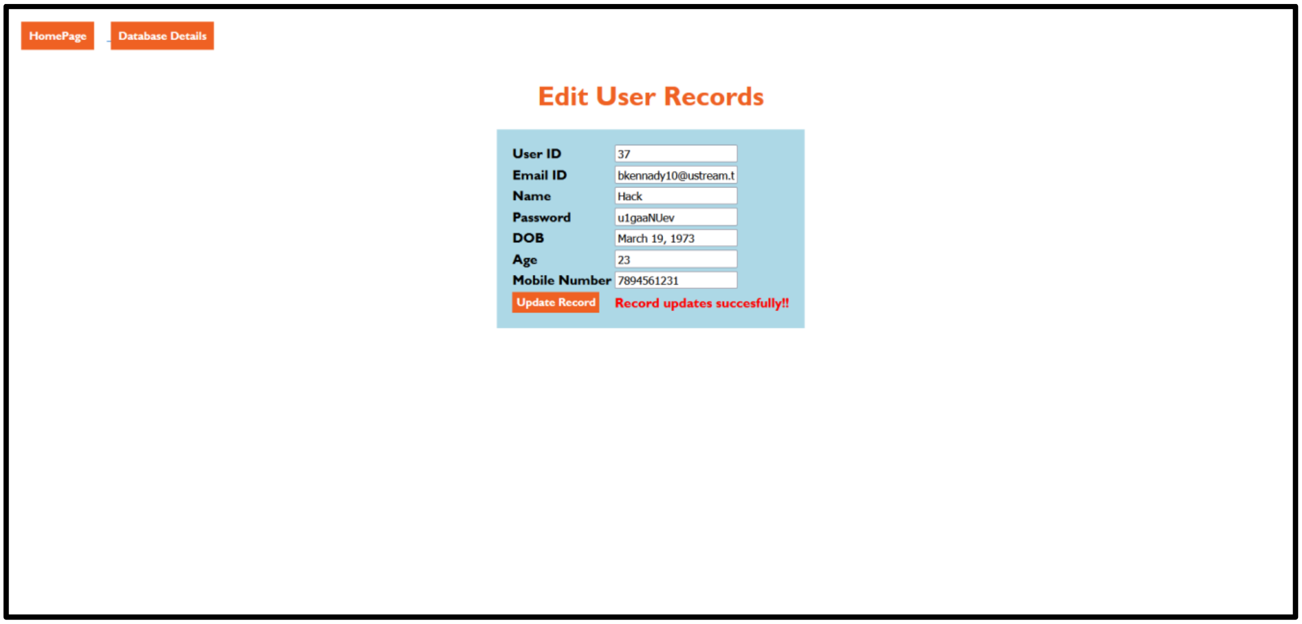
| User ID | Email ID | Name | Password | DOB | Age | Mobile Number | Edit | Delete |
|---|---|---|---|---|---|---|---|---|
| 202 | a@a | aaa | aaaa | Nov. 9, 2022 | 25 | 2345698765 | Edit | Delete |
| 13 | pmec@edublogs.org | aayush | mksnx | May 28, 1897 | 27 | 7418529630 | Edit | Delete |
| 69 | agonsalo1w@live.com | Abra | 3VJY9Xs | Sept. 4, 2006 | 10 | None | Edit | Delete |
| 152 | aeicke47@mashable.com | Adelaide | ekbs1nt | July 17, 1988 | 27 | None | Edit | Delete |
| 9 | adaly8@japanpost.jp | Alec | tWiamzn7eh21 | Nov. 11, 1990 | 32 | 7418598745 | Edit | Delete |
| 145 | aanthiftle40@businesswire.com | Alexio | AvOMH3ODmyg | July 31, 2012 | 14 | None | Edit | Delete |
| 136 | asagerson3r@phoca.cz | Alexis | hV6BBAdqLZ4 | July 23, 1988 | 28 | None | Edit | Delete |

## ❖ Edit the data of existing users:



Edit User Records

| | |
|---|---|
| User ID | 37 |
| Email ID | bkennady10@ustream.t |
| Name | Barbe |
| Password | u1gaaNUev |
| DOB | March 19, 1973 |
| Age | 23 |
| Mobile Number | 7894561231 |



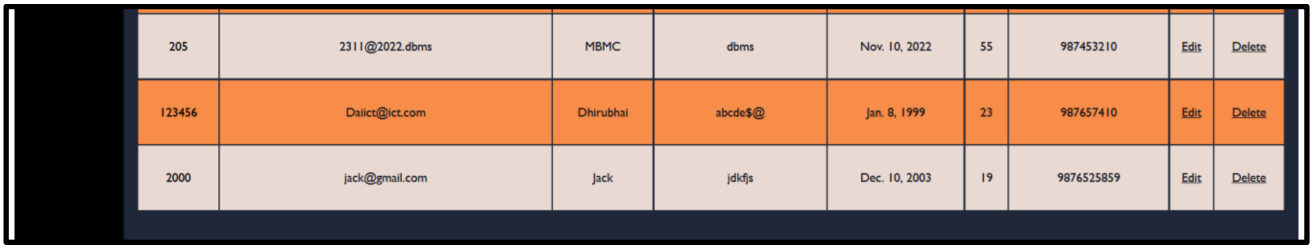
Edit User Records

| | |
|---|---|
| User ID | 37 |
| Email ID | bkennady10@ustream.t |
| Name | Hack |
| Password | u1gaaNUev |
| DOB | March 19, 1973 |
| Age | 23 |
| Mobile Number | 7894561231 |

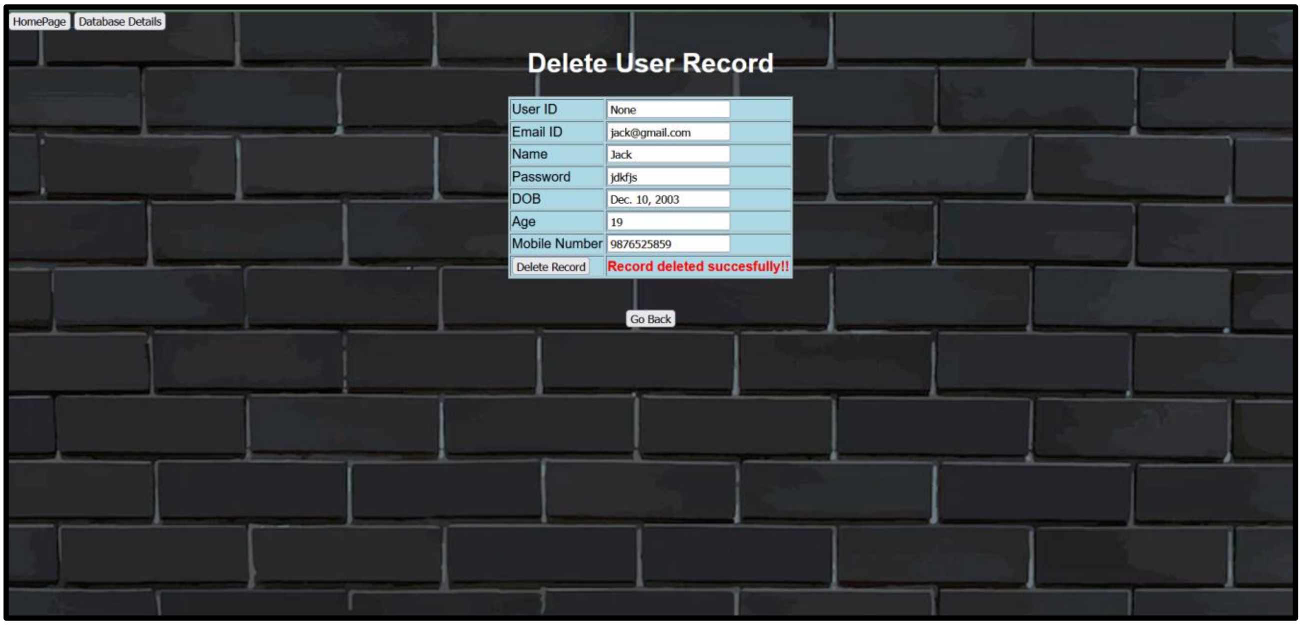
Record updates succesfully!!

- Details of user_id = 37 have been updated.

## ❖ Delete existing user:

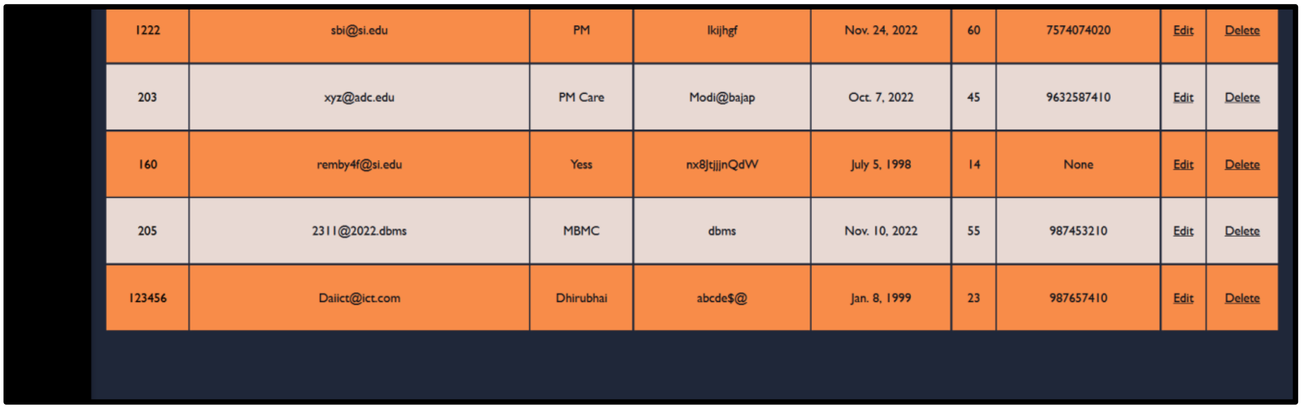


| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 205 | 2311@2022.dbms | MBMC | dbms | Nov. 10, 2022 | 55 | 987453210 | Edit | Delete |
| 123456 | Daiict@ict.com | Dhirubhai | abcde$@ | Jan. 8, 1999 | 23 | 987657410 | Edit | Delete |
| 2000 | jack@gmail.com | Jack | jdkfjs | Dec. 10, 2003 | 19 | 9876525859 | Edit | Delete |



Delete User Record

| | |
|---|---|
| User ID | None |
| Email ID | jack@gmail.com |
| Name | Jack |
| Password | jdkfjs |
| DOB | Dec. 10, 2003 |
| Age | 19 |
| Mobile Number | 9876525859 |
| Delete Record | Record deleted succesfully!! |

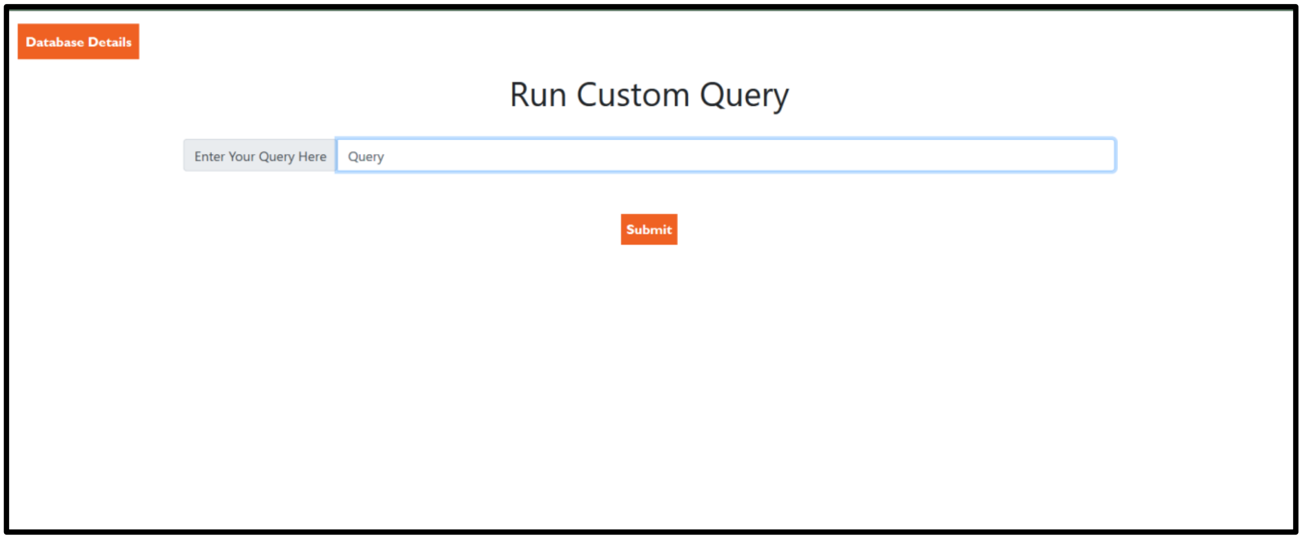


| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1222 | sbi@si.edu | PM | lkijhgf | Nov. 24, 2022 | 60 | 7574074020 | Edit | Delete |
| 203 | xyz@adc.edu | PM Care | Modi@bajap | Oct. 7, 2022 | 45 | 9632587410 | Edit | Delete |
| 160 | remby4f@si.edu | Yess | nx8JtjjjnQdW | July 5, 1998 | 14 | None | Edit | Delete |
| 205 | 2311@2022.dbms | MBMC | dbms | Nov. 10, 2022 | 55 | 987453210 | Edit | Delete |
| 123456 | Daiict@ict.com | Dhirubhai | abcde$@ | Jan. 8, 1999 | 23 | 987657410 | Edit | Delete |

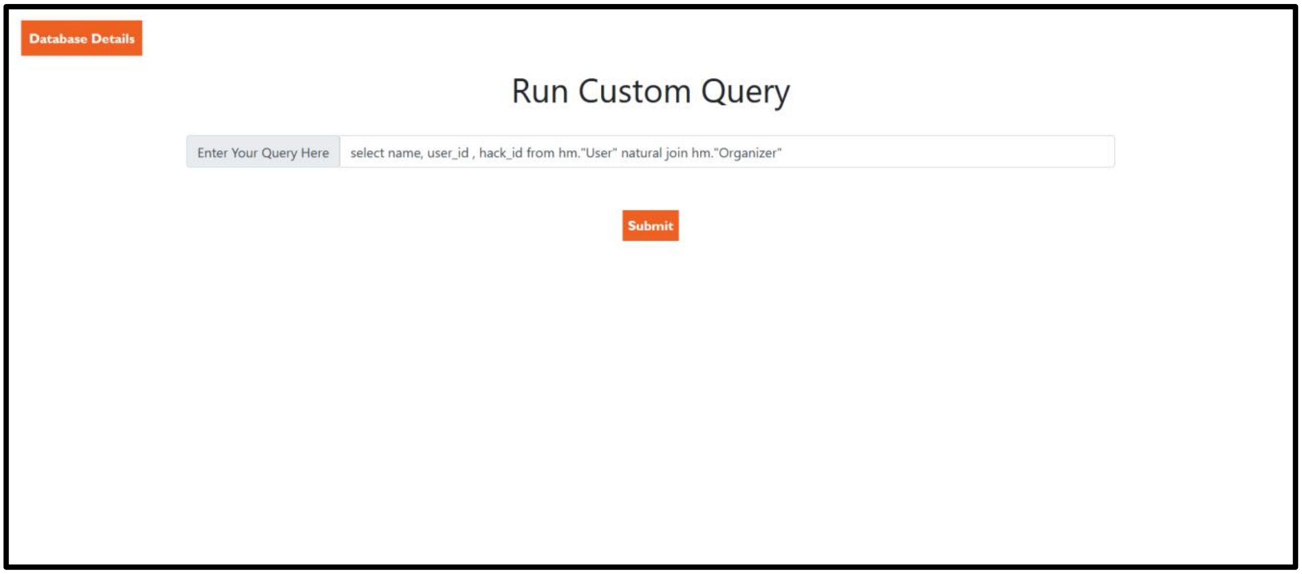
- Details of user_id = 2000 have been deleted.

## ❖ Run Query:

- Query page:



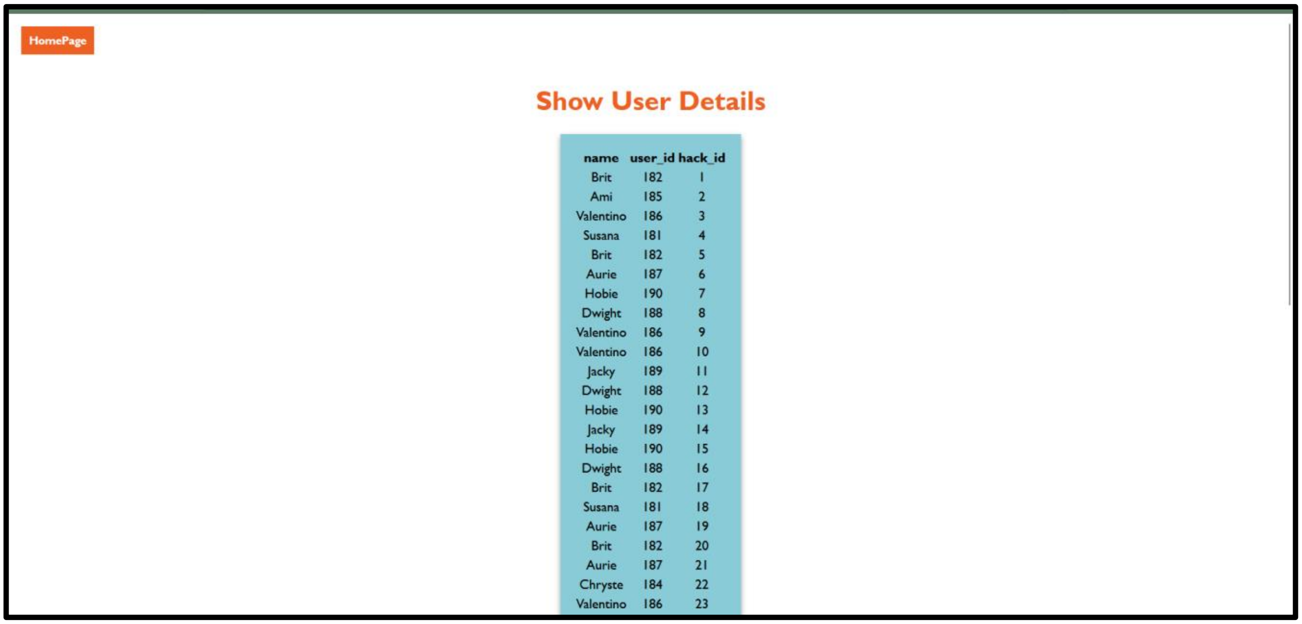
1. **Run query to get the user ids and names of the Organizers along with the hackathon ids of the hackathon they are Organizing.**



Run Custom Query

Enter Your Query Here: select name, user_id , hack_id from hm."User" natural join hm."Organizer"

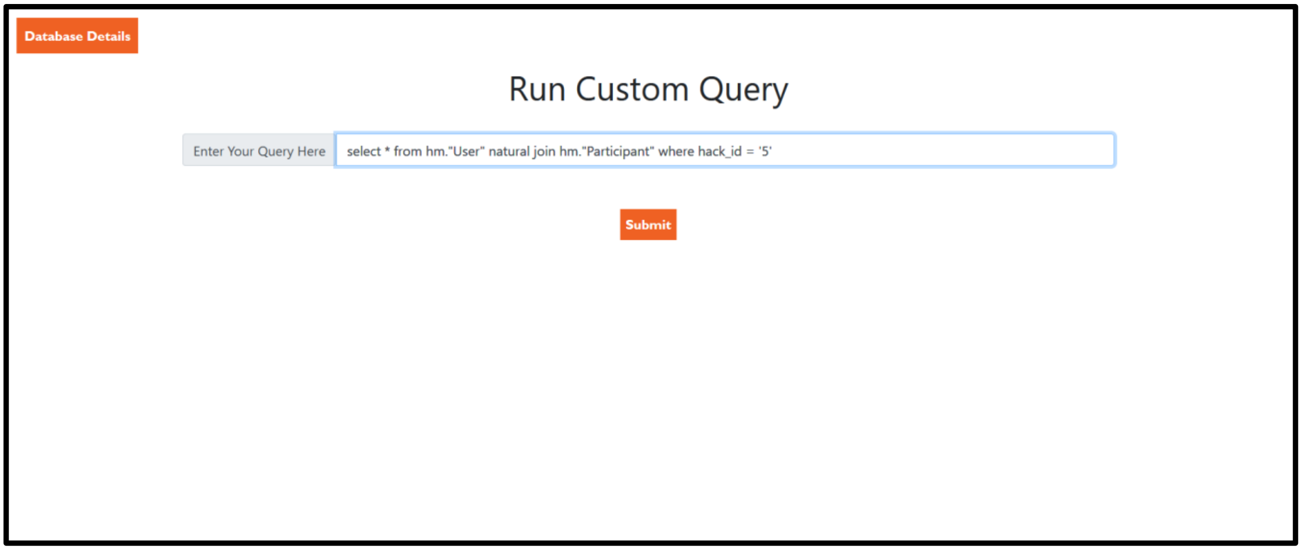
- **Output:**



Show User Details

| name | user_id | hack_id |
|---|---|---|
| Brit | 182 | 1 |
| Ami | 185 | 2 |
| Valentino | 186 | 3 |
| Susana | 181 | 4 |
| Brit | 182 | 5 |
| Aurie | 187 | 6 |
| Hobie | 190 | 7 |
| Dwight | 188 | 8 |
| Valentino | 186 | 9 |
| Valentino | 186 | 10 |
| Jacky | 189 | 11 |
| Dwight | 188 | 12 |
| Hobie | 190 | 13 |
| Jacky | 189 | 14 |
| Hobie | 190 | 15 |
| Dwight | 188 | 16 |
| Brit | 182 | 17 |
| Susana | 181 | 18 |
| Aurie | 187 | 19 |
| Brit | 182 | 20 |
| Aurie | 187 | 21 |
| Chryste | 184 | 22 |
| Valentino | 186 | 23 |

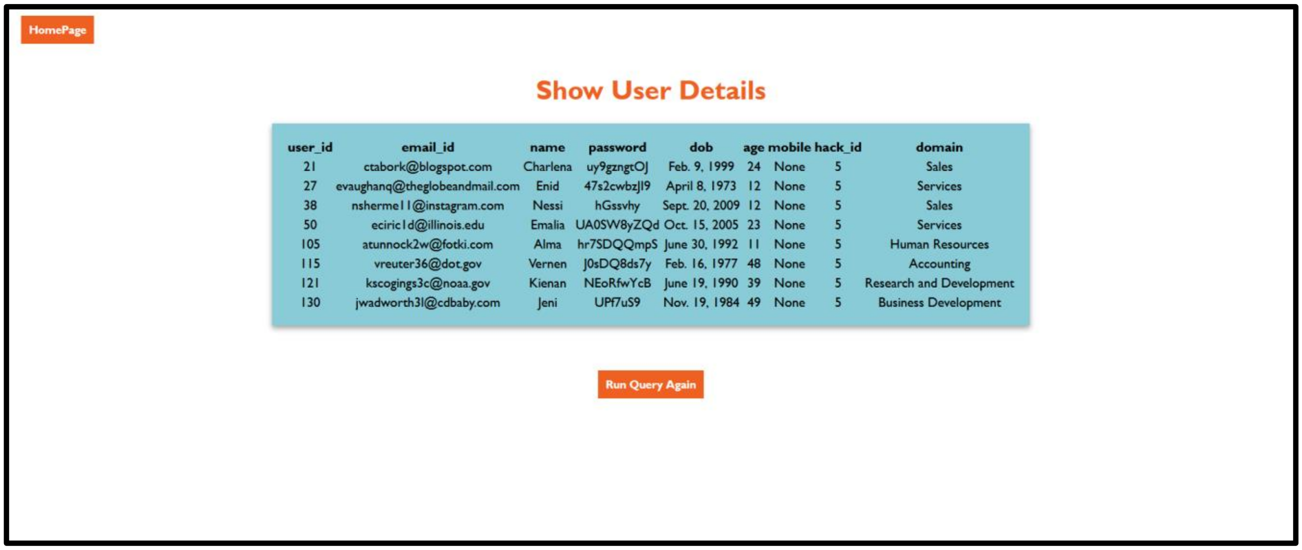
**2. Run query to get the details of participants who are participants in the hackathon with hack_id = 5.**



Run Custom Query

Enter Your Query Here: select * from hm."User" natural join hm."Participant" where hack_id = '5'

- **Output:**



Show User Details

| user_id | email_id | name | password | dob | age | mobile | hack_id | domain |
|---|---|---|---|---|---|---|---|---|
| 21 | ctabork@blogspot.com | Charlena | uy9gzngtOJ | Feb. 9, 1999 | 24 | None | 5 | Sales |
| 27 | evaughanq@theglobeandmail.com | Enid | 47s2cwbzJl9 | April 8, 1973 | 12 | None | 5 | Services |
| 38 | nsherme11@instagram.com | Nessi | hGssvhy | Sept. 20, 2009 | 12 | None | 5 | Sales |
| 50 | eciric1d@illinois.edu | Emalia | UA0SW8yZQd | Oct. 15, 2005 | 23 | None | 5 | Services |
| 105 | atunnock2w@fotki.com | Alma | hr7SDQQmpS | June 30, 1992 | 11 | None | 5 | Human Resources |
| 115 | vreuter36@dot.gov | Vernen | J0sDQ8ds7y | Feb. 16, 1977 | 48 | None | 5 | Accounting |
| 121 | kscogings3c@noaa.gov | Kienan | NEoRfwYcB | June 19, 1990 | 39 | None | 5 | Research and Development |
| 130 | jwadworth3l@cdbaby.com | Jeni | UPf7uS9 | Nov. 19, 1984 | 49 | None | 5 | Business Development |

## GitHub Repository link:

https://github.com/virat10/HackathonManagementSystem