

CROWDSOURCED DISASTER MANAGEMENT RESPONSE SYSTEM

Group ID – 16

Student – Pansuriya Devangi (202412057)

Pandya Dhruvisha (202412019)

Institute – DAIICT, Gandhinagar

Date of submission – 12-11-2024

INDEX

No.	Title	Page no.
	SOFTWARE REQUIREMENT SPECIFICATION (SRS)	5
1	Problem Description	6
1.1	Introduction	6
1.2	Product Scope	8
2	Requirement Collection	9
2.1	Detailed Description of background reading	9
2.1(A)	Purpose	9
2.1(B)	Role of Technology in Disaster Management	11
2.1(C)	Key challenges in Crowdsourced disaster management response	16
2.1(D)	Real time decision making in crowdsourced disaster response coordination	18
2.1(E)	Case study – Kerala floods- 2018 – A technology driven response	20
2.1(F)	List of requirement gathered from background reading	22
2.1(G)	References	23
2.2	Interview	24
2.3	Questionnaires	27
2.3.1	Responses received from survey analysis	29

2.3.2	List of requirement gathered from responses	38
2.4	List of requirement gathered by observation	39
3	Fact finding chart	40
	DATABASE DESIGN	41
1	Noun Analysis	42
2	Schema and ER diagram design	57
3	ER Diagram Analysis	61
	Final ER Diagram	65
4	Mapping ER Model to Relational Model	66
5	DDL Scripts	71
	NORMALIZATION OF DATABASE	80
1	Normalization and Schema Refinement	81
1.1	Original Design of Database	81
2	Dependency, Redundancy, Anomalies Analysis	84
3	Normalization Process	97
3.1	First normal form(1NF)	97
3.2	Second normal form(2NF)	101
3.3	Third normal form(3NF)	106
3.4	BCNF	107
	Final Schema	108
	IMPLEMENTATION OF DATABASE	110
1	Revised DDL Scripts	111

2	Database Population	122
3	SQL Queries	258
	INTERFACE IMPLEMENTATION	294
1	Setup JDBC and Basic GUI	295
2	CRUD Operation in GUI	297
3	Code	316
	TECHNICAL ISSUES AND SOLUTION	330
1	Technical Issues	331
2	Solution	331

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1. PROBLEM DESCRIPTION

1.1 INTRODUCTION

In a disaster management response system, the need for real-time, reliable, and large-scale information is paramount. A crowdsourced disaster management response system leverages data provided by the public, first responders, and automated systems (like sensors and drones) to gather, process, and act upon critical information during a disaster. The success of such a system heavily relies on a well-designed and scalable database that can accommodate a wide range of inputs, process them in real-time, and make them accessible to the right authorities for decision-making and action.

Disaster management involves a systematic approach to preparing for, responding to, and recovering from disasters. It encompasses several phases, including mitigation, preparedness, response, and recovery. Each phase plays a critical role in minimizing the impact of disasters and ensuring the safety and well-being of affected communities. However, traditional disaster management approaches often face significant challenges, particularly in rapidly changing and complex environments.

A crowdsourced disaster management response system is a collaborative platform that leverages information from the public and other decentralized sources to enhance disaster response efforts. It enables individuals, volunteers, and organizations to provide real-time data such as reports, images, videos, and social media updates during natural or man-made disasters. By integrating these crowdsourced inputs with official data from authorities, sensors, and response teams, the system creates a comprehensive, real-time overview of the disaster

situation. This facilitates more effective coordination, resource allocation, and decision-making for emergency services and government agencies. The system aims to improve situational awareness, speed up response times, and ultimately minimize damage and loss of life.

The Emergence of Crowdsourcing in Disaster Management

In recent years, crowdsourcing has emerged as a transformative approach to disaster response. It involves leveraging the collective intelligence and resources of the public to enhance situational awareness, coordinate response efforts, and facilitate recovery.

Crowdsourcing can take various forms, including social media campaigns, mobile applications, and online platforms that allow individuals to report incidents, share information, and mobilize resources.

The effectiveness of crowdsourcing in disaster management is evident in several high-profile cases. For instance, during the 2010 Haiti earthquake, volunteers from around the world used social media and mapping tools to provide real-time information about affected areas, name and other information of victims, location, helping aid organizations to allocate resources more effectively. Similarly, in the aftermath of Typhoon Haiyan in the Philippines, crowdsourced data facilitated better coordination among NGOs, government agencies, and local communities, ultimately improving the response and recovery process. Members of agencies in contacts with government to give proper information about disaster and location.

1.2 PRODUCT SCOPE

The product scope of the crowdsourced disaster management response system database encompasses the development of a centralized platform designed to collect, store, and manage data from various sources during a disaster. The database will aggregate real-time inputs from crowdsourced reports, social media, IoT sensors, and official channels, facilitating efficient disaster response. It will include functionalities for data validation, geolocation tracking, and real-time updates, ensuring that first responders and authorities can access accurate and actionable information. The system will support large-scale data handling, integrate with mapping and resource management tools, and offer a user-friendly interface for quick decision-making.

While the database will focus on crowdsourced inputs and coordination of response efforts, it will not include advanced predictive analytics or long-term disaster recovery planning. This scope ensures that the system remains scalable, reliable, and optimized for immediate disaster response, without introducing unnecessary complexity that could delay critical functions.

2. REQUIREMENT COLLECTION

2.1 DETAILED DESCRIPTION OF BACKGROUND READING

(A) PURPOSE

Crowdsourcing, together with the widespread use of mobile technology and social media, is transforming how we handle disaster response. Crowdsourcing means gathering information or getting tasks done by relying on contributions from a large group of people instead of just a few. This method is now being used more in disaster situations to gather vital information, locate trouble spots, and coordinate tasks efficiently.

It helps responders, decision makers, and researchers work together better during emergencies. Access to data is crucial for responding to disasters, but many organizations struggle with poor coordination and a lack of real-time information. Crowdsourcing bridges this gap by using input from the public, including people affected by the disaster and volunteers worldwide. This method gives responders up-to-date information on rapidly changing conditions and helps them prioritize their efforts. Another benefit is that crowdsourcing captures feedback directly from the people affected, which can sometimes be overlooked.

For example, after the 2010 Haiti earthquake, many locals and volunteers used phones and laptops to create maps, translate messages, and report emergencies at almost no cost. Their contributions helped humanitarian workers respond more quickly and effectively. This success highlights the shift away from only using traditional methods to relying on mass collaboration.

Since then, organizations like Médecins Sans Frontières and the UN have started using crowdsourced data in response efforts, as seen during Typhoon Haiyan in the Philippines. Crowdsourcing allows non-experts to contribute as "digital humanitarians" and "crisis mappers," working alongside professionals like space agencies to improve disaster response.

Major aspects of disaster management includes:

- ◆ Disaster Prevention
- ◆ Disaster Preparedness
- ◆ Disaster Response
- ◆ Disaster Recovery

(B) ROLE OF TECHNOLOGY IN DISASTER MANAGEMENT

While technology may not replace the vital resources people need during disasters – food, shelter, water, and proximity with loved ones – technology is still changing disaster management preparedness and smoothness.

Technology plays a pivotal role in enhancing the effectiveness of crowdsourced disaster management. With the rise of smartphones and internet connectivity, individuals can quickly share critical information about emergencies in real time. Social media platforms like Twitter and Facebook serve as powerful tools for disseminating updates, alerts, and instructions during crises.

The accessibility of these platforms allows citizens to report incidents, share situational awareness, and communicate needs, thereby creating a robust network of information that can supplement official disaster response efforts. Furthermore, Geographic Information Systems (GIS) and mapping technologies are integral to analyzing and visualizing crowdsourced data. These tools enable responders to interpret spatial data and identify areas most affected by disasters. For example, during floods or earthquakes, crowdsourced geospatial data can help authorities pinpoint where aid is needed most urgently. Additionally, the integration of technologies such as satellite imagery and drones enhances situational awareness, providing real-time views of affected areas and allowing for more efficient resource allocation and response strategies.

- ♦ Data analytics also plays a crucial role in processing and validating the vast amounts of information generated through

crowdsourcing. Advanced algorithms and machine learning techniques can help filter out noise, identify patterns, and assess the credibility of user

generated content. By analyzing social media posts, for instance, disaster management agencies can quickly gauge public sentiment and gather insights into the evolving needs of affected populations. This analytical capability empowers decision-makers to respond more effectively and tailor their actions to the unique challenges posed by each disaster.

◆ Moreover, mobile applications and platforms designed specifically for disaster management facilitate better coordination among volunteers, emergency responders, and affected communities. These technologies can streamline the reporting process, enabling users to submit real-time information about hazards, resource availability, and individual needs. By fostering communication and collaboration, technology strengthens the collective response to disasters, allowing for a more organized and efficient effort in managing crises.

◆ In conclusion, technology serves as a crucial enabler in crowdsourced disaster management, transforming how information is gathered, analyzed, and shared. By leveraging social media, GIS, data analytics, and dedicated applications, disaster response efforts can be significantly enhanced, leading to better outcomes for communities affected by crises. As technology continues to evolve, its integration into disaster management strategies will undoubtedly play an increasingly important role in building resilience and improving emergency responses.

Tools and Platforms:

In the realm of crowdsourced disaster management, various tools and platforms have emerged to facilitate data collection, analysis, and communication among citizens, volunteers, and emergency responders. These tools enhance situational awareness, improve response coordination, and empower communities to actively participate in disaster management efforts.

a. Social Media Platforms:

Social media plays a crucial role in crowdsourced disaster management by enabling real-time information sharing and communication. Platforms like Twitter, Facebook, and Instagram allow individuals to report incidents, share updates, and communicate needs during emergencies. Hashtags, geotags, and location-based features help organize and categorize information, making it easier for emergency responders to access relevant data. Social media also serves as a valuable tool for disseminating official alerts and safety information to the public.

b. Geographic Information Systems (GIS):

GIS tools are essential for visualizing and analyzing spatial data collected during disasters. These tools help map the locations of incidents, assess damage, and identify areas in need of assistance. Platforms like ArcGIS and QGIS allow users to create detailed maps that integrate crowdsourced data, such as reports of damage or shelter availability. GIS technologies enable responders to visualize the extent of disasters and make informed decisions regarding resource allocation and response strategies.

c. Mobile Applications:

Various mobile applications are designed specifically for disaster management, allowing users to report incidents, access real-time information, and connect with emergency services. For example, applications like Disaster Alert provide users with alerts and updates on ongoing emergencies, while platforms like Ushahidi enable crowdsourced reporting of incidents through mobile devices. These apps often include features for mapping incidents, sharing resources, and facilitating communication among community members and responders.

d. Crowdsourcing Platforms:

Dedicated crowdsourcing platforms such as Ushahidi, OpenStreetMap, and Crisis Mapping play a vital role in collecting and visualizing data during disasters. Ushahidi allows users to submit reports via SMS, email, or web forms, which are then mapped for analysis. OpenStreetMap enables volunteers to contribute to mapping efforts by adding details about infrastructure, resources, and hazards. These platforms foster collaboration among citizens, NGOs, and government agencies, improving the overall response to disasters.

e. Data Analytics Tools:

Advanced data analytics tools and machine learning algorithms are increasingly used to process and validate crowdsourced information. Tools like Tableau, R, and Python libraries enable analysts to explore patterns, assess data credibility, and extract insights from large datasets. These tools help emergency responders filter out noise, identify trends, and prioritize actions based on the most critical needs during a disaster.

f. Drones and Remote Sensing Technologies:

Drones equipped with cameras and sensors provide real-time aerial imagery and data during disasters, enabling responders to assess damage and monitor evolving situations. Remote sensing technologies, including satellite imagery, allow for large-scale observations of disaster-affected areas, providing valuable insights into damage assessment and recovery planning. Combining drone and remote sensing data with crowdsourced information enhances situational awareness and improves decision-making.

g. Communication Tools:

Instant messaging and collaboration platforms like WhatsApp, Slack, and Signal facilitate communication among volunteers, emergency responders, and affected communities. These tools enable quick information exchange, coordination of response efforts, and dissemination of critical updates during emergencies.

(C) KEY CHALLENGES IN CROWDSOURCED DISASTER MANAGEMENT RESPONSE

1. Data Collection:

- ◆ Technology advancements have made it easier to collect data from various platforms. Social media, especially Twitter, is often used due to its open API, allowing researchers to gather real-time data during disasters.
- ◆ However, some organizations still rely primarily on text searches rather than geographic searches to collect relevant information.

2. Data Credibility and Quality:

- ◆ The immediacy of social media can lead to quick information sharing, but this raises concerns about data reliability. Misinformation can spread rapidly, making it essential to develop tools to assess the quality of crowdsourced information.
- ◆ Proposed methods include verifying data against official sources and using a protocol to enhance data quality.

3. Privacy Issues:

- ◆ Using social media data raises ethical concerns regarding user privacy. While Twitter's public data can be accessed more easily than Facebook's, researchers must still ensure that individuals' privacy is respected and that consent is understood.

4. Participant Engagement:

- ◆ Citizens are often motivated to share information during crises to feel a sense of control and to help others. Emotional responses and altruism also drive participation.
- ◆ Engaging volunteers in the data collection process can improve the quality and relevance of the information gathered.

5. Spatiotemporal Data Interpretation:

- ◆ Interpreting the collected data is challenging due to its spatial and temporal context. Mapping data accurately during crises is crucial for effective response.
- ◆ Advanced analysis methods, including the use of Big Data, can help create more precise maps and improve understanding of complex social and spatial relationships.

In conclusion, while crowdsourcing offers significant potential for disaster management, it also presents challenges related to data collection, quality, privacy, engagement, and interpretation that must be addressed to ensure effective responses to crises.

(D) REAL TIME DECISION MAKING IN CROWDSOURCED DISASTER RESPONSE COORDINATION

Real-time decision-making significantly improves the effectiveness of crowdsourced disaster response by ensuring that resources are deployed efficiently, volunteers are well-coordinated, and evolving risks are managed proactively.

By integrating real-time data from crowdsourced efforts, responders can make informed decisions that save lives and reduce the impact of disasters.

Situational Awareness: with the help of a crowdsource system Individuals on the ground continuously provide real-time information about the disaster's impact (e.g., infrastructure damage, flooded areas, or stranded populations). Real-time decision-making allows emergency teams to assess these reports instantly and prioritize responses.

During the 2018 Kerala floods, citizens used social media to report their locations. Volunteers and rescue teams responded quickly by tracking these reports and mapping areas needing urgent attention (Amrita Kripa app).

It helps us for,

- ◆ Rapid Adjustment of Resources: With real-time decision-making, crowdsourced data helps responders adjust their resources based on changing conditions. For example, if an area that was thought to be safe becomes flooded, the decision to redirect resources such as boats and relief supplies can be made immediately.

- ◆ Improved Coordination of Volunteers: Real-time decision-making enables platforms to assign tasks to volunteers based on immediate needs. If a large number of volunteers suddenly become available in one area, they can be directed to where they are most needed, avoiding duplication of effort and gaps in coverage.
- ◆ Rapid Response to Emerging Risks: With predictive analytics and real time monitoring, decision-makers can anticipate risks before they escalate. For instance, real-time flood models can help responders decide whether to issue evacuation orders or reroute resources to higher-risk areas.
- ◆ Enhanced Collaboration Between Agencies and government Real-time decision-making ensures that the data collected from the public is integrated into official disaster response systems. This improves collaboration between government agencies, NGOs, and volunteers, ensuring that efforts are not duplicated and resources are maximized.

With real-time situational awareness provided by crowdsourced inputs, authorities can identify hotspots, assess the severity of incidents, and detect new emerging risks or problem areas. For example, during the catastrophic Kerala floods of 2018, citizens utilized social media platforms to report their locations and conditions. This information enabled volunteers and rescue teams to respond quickly, mapping areas that required urgent attention using tools like the Amrita Kripa app. Such instances exemplify how real-time situational awareness can lead to timely and effective interventions.

(E) CASE STUDY : Kerala Floods 2018 - A Technology-Driven Response

In August 2018, the South Indian state of Kerala experienced its worst floods in nearly a century, caused by unusually heavy monsoon rainfall. The disaster resulted in the tragic loss of over 483 lives and the evacuation of approximately one million people. In response to this crisis, Chennai saw an unprecedented outpouring of relief efforts from volunteers and non-governmental organizations (NGOs). Various technologies were utilized extensively to facilitate communication and coordination during the disaster. Tools such as Google Spreadsheets, Google Crisis Map, Facebook Safety Check, and Twitter played crucial roles in sharing information about individuals in need of rescue, as well as requests and offers for essential supplies like food, clothing, and sanitary items.

One notable innovation was the development of the application Amrita Kripa, which supports a crowdsourced disaster relief model. This platform allows victims to request assistance, including rescue, medical attention, and essential supplies. Volunteers can also offer help through the app.

The interactive map interface of Amrita Kripa displays all requests and offers in real time, allowing users to filter and drill down into specific needs. Both mobile and web applications are available, with the web version particularly suited for government agencies and NGOs, providing administrative features to view detailed reports of disaster relief efforts.

The Amrita Kripa mobile app was utilized directly by victims as well as by call center volunteers. Victims could either enter their requests into

the app or call the call center for assistance. Upon receiving calls, personnel collected critical data, such as location, type of request, number of affected individuals, and contact information, which was then entered into the system.

The call center also established verification teams to confirm the legitimacy of requests and offers. For rescue requests, teams coordinated with local police or Army helplines, while medical and supply teams followed up with victims and volunteers to confirm details. Through this comprehensive approach, the call center aided over 100,000 victims by managing more than 25,000 phone calls. The app recorded over 5,117 rescue requests, 4,928 supply requests, and various service and medical requests. Overall, the Amrita Kripa platform demonstrated the power of technology in mobilizing community support and improving disaster response during the Kerala floods.

(F) REQUIREMENT GATHERED FROM BACKGROUND READING

1. Incident Data:

Location: Coordinates or addresses of the disaster areas.

Type of Disaster: Earthquake, flood, fire, etc.

Severity: Impact level or damage assessment.

2. Volunteer Information:

Contact Details: Names, phone numbers, email addresses.

Skills and Expertise: Medical training, firefighting experience, etc.

Availability: When users are available to help. Also include current assignment and deployment status.

3. Resource Information:

Supplies: Availability of food, water, medical supplies.

Shelter Locations: Places offering temporary accommodation.

Source: Donors and suppliers record

4. Victim Information:

Identification: Names, ages, and medical conditions.

Needs: Specific assistance required (e.g., medical attention, evacuation).

5. Alerts and Notifications:

Warnings: Updates on ongoing conditions or emerging threats.

Instructions: Guidelines for responders and affected individuals.

6. Agency and Organization data

government agencies and NGOs

Contact details and roles of agencies

Contact details of Ambulance facility and availability.

(G) REFERENCES

https://www.physio-pedia.com/Disaster_Management

<https://preparecenter.org/topic/crowdsourcing>

<https://link.springer.com/article/10.1007/s10796-017-9734-6>

<https://geoenvironmental-disasters.springeropen.com/articles/10.1186/s40677-021-00181-3>

[Mobile app saves 12,000 flood victims in Kerala \(livemint.com\)](#)

https://en.wikipedia.org/wiki/2018_Kerala_floods

<https://www.sciencedirect.com/science/article/pii/S1877050920313077>

https://www.researchgate.net/publication/333311009_Kerala_Floods - A_Model_of_Rescue_and_Rehabilitation_using_Information_Technology_and_Social_Media_based_Crowdsourcing

<https://www.cdc.gov/surveillance/data-modernization/snapshot/2022-snapshot/stories/outbreak-response-emergencies.html>

<https://www.forbes.com/sites/trevornace/2017/12/15/how-technology-is-advancing-emergency-response-and-survival-during-natural-disasters/>

<https://safetyculture.com/topics/emergency-management/ai-in-emergency-management/>

2.2 INTERVIEW PLAN (Role play)

System : Crowdsourced Disaster Response Coordination System

Students : Devangi Pansuriya(Interviewer)
Dhruvisha Pandya (Interviewee)

Date : 09/09/2024

Time : 11:00 AM

Duration : 30 minutes

Place : Classroom

Purpose of Interview :

Gain insight and opinions on the design, functionality and real world application of a Crowdsourced Disaster Response Coordination System that will help to guide the technical and functional requirements of our project.

Agenda :

Identify the problems of traditional disaster management

system Initial ideas to improve disaster management

Gathering requirements

Interview Summary

System : Crowdsourced Disaster Response Coordination System

Participants : Dhruvisha Pandya

Devangi Pansuriya

Date : 09/09/2024

Duration : 30 minutes

Place : Classroom

Time : 11:00 AM

Purpose of Interview :

Gain insight and opinions on the design, functionality and real world application of a Crowdsourced Disaster Response Coordination System that will help to guide the technical and functional requirements of our project.

1. Real time and diverse data collection is essential for efficient disaster response.
2. The involvement of local populations can significantly speed up response times and improve accuracy.
3. Historical cases can offer valuable insights for system improvements and show effectiveness of crowdsourcing.
4. Accurate data are crucial for disaster management.
5. Data validation mechanism ensure reliability while balancing the need for fast information flow.
6. Real time updates are vital for efficient Disaster management.
7. User friendliness and real time decision making are critical to the success of system.

8. Quick decision making based on live data is essential in emergency situation.
9. Data security must be priority to maintain trust in the system. Security and Integrity are critical, especially when crowdsourced data is involved.
10. Requirement for database/ system are victim information, incident data, volunteer information, resources, agency and organizations data.

2.3 QUESTIONNARIES :

Survey for Crowdsourced disaster response Coordination system

1. How familiar are you with crowdsourced disaster response systems?

Very familiar / Somewhat familiar / Not familiar

2. Do you think that a system that gathers real time data from the public could improve disaster response efforts? Yes / No / Not sure

3. How likely are you to report disaster-related events through a crowdsourced system if such a system were available?

Very likely / Somewhat likely / Neutral / Unlikely

4. How effective do you think a system like this could be in saving lives during a disaster ?

Extremely effective / Moderately effective / Slightly effective / Not effective

5. Would you feel comfortable submitting real time data such as photos, videos or reports during a Disaster situation ?

Yes / No / Maybe

6. How reliable do you think crowdsourced data during a disaster ?

Very reliable / Somewhat reliable / Not reliable / Don't know

7. Which platform would you prefer to use to submit information during a disaster ?

Mobile app / Social media / Text message / Website form

8. What concerns do you have about using a crowdsourced disaster response system ?

Data privacy / Accuracy of information / Overload of information / Technical issues / Coordination difficulties / Lack of verification

9. How important is real time communication with other agencies, local government, NGOs and volunteers during a disaster ?

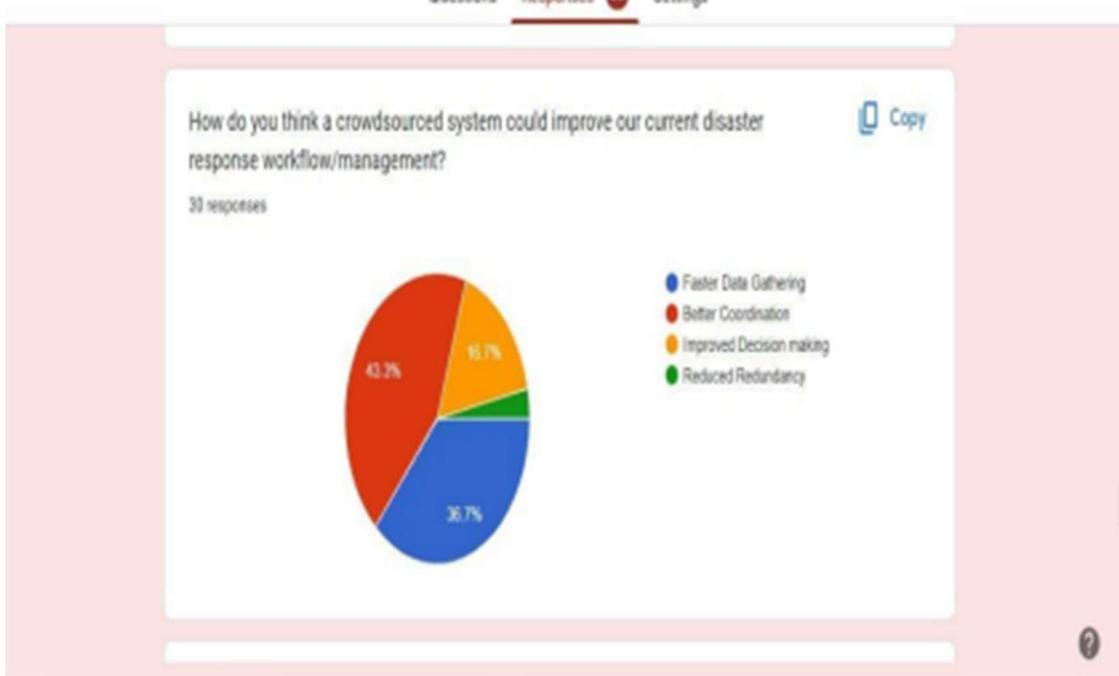
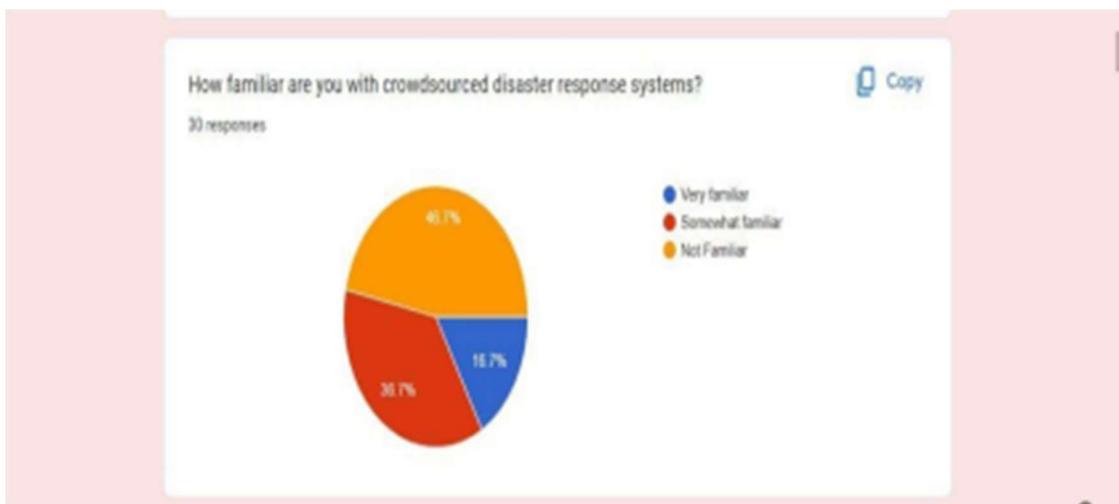
Extremely important / Very important / Somewhat important / Not important

10. How do you think a crowdsourced system could improve our current disaster response workflow/ management ? Faster data gathering / Better coordination / Improve decision making / Reduced redundancy

11. Note any suggestion to improve our current Disaster Response management.

12. Do you think such systems should be mandated for all companies and local governments ? Why or Why not ?

2.3.1 RESPONSES RECEIVED FROM GOOGLE FORMS

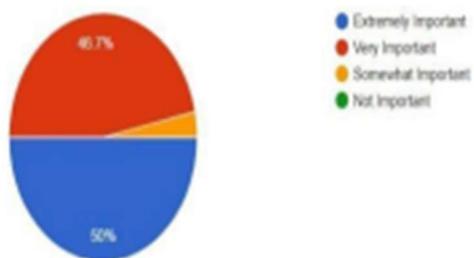


Questions Responses 30 Settings

How important is real-time communication with other agencies, local Government, NGOs, and volunteers during a disaster?

 Copy

30 responses

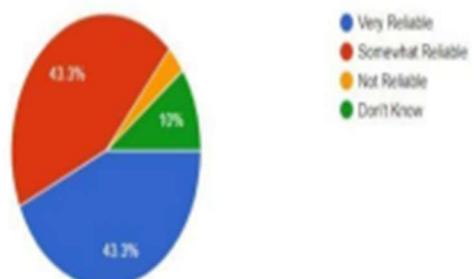


Questions Responses 30 Settings

How reliable do you think crowdsourced data is during a disaster?

 Copy

30 responses

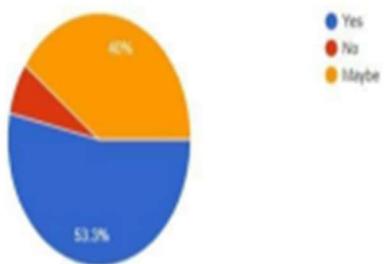


Questions Responses 30 Settings

Would you feel comfortable submitting real-time data such as photos, videos, or reports during a disaster situation?

30 responses

Copy

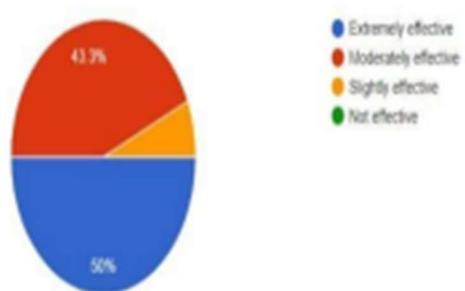


Questions Responses 30 Settings

How effective do you think a system like this could be in saving lives during a disaster?

30 responses

Copy

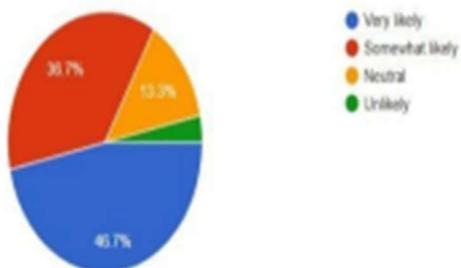


Questions Responses 30 Settings

How likely are you to report disaster-related events through a crowdsourced system if such a system were available?

 Copy

30 responses

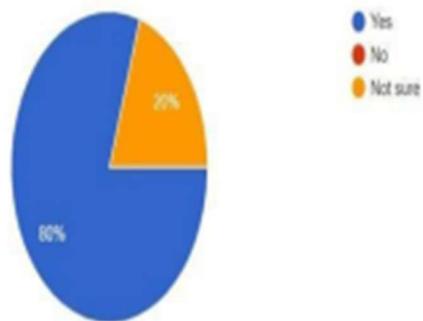


Questions Responses 30 Settings

Do you think that a system that gathers real-time data from the public could improve disaster response efforts?

 Copy

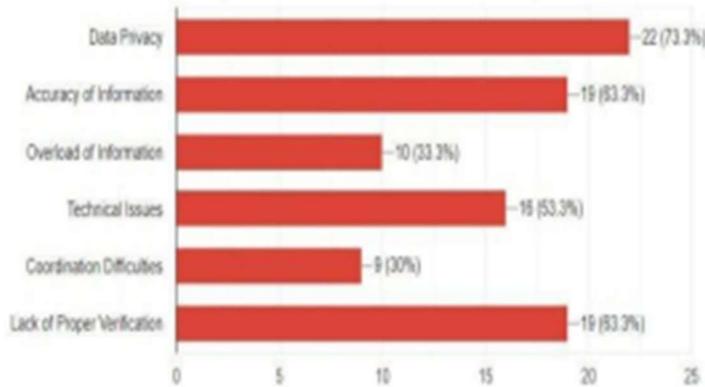
30 responses



Questions Responses 30 Settings

What concerns do you have about using a crowdsourced disaster response system?
(Check all that apply)

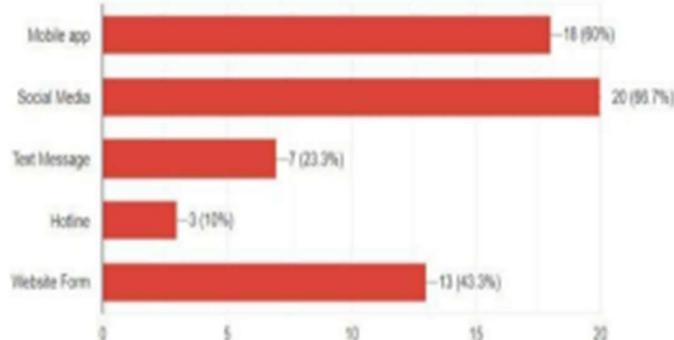
30 responses



Questions Responses 30 Settings

Which platform would you prefer to use to submit information during a disaster?
(Select all that apply)

30 responses



Questions Responses 30 Settings

Do you think such systems (Crowdsourced disaster response coordination) should be mandated for all companies and local governments? Why or why not?

30 responses

Because it helps to manage disasters

Such systems could be really helpful in taking faster and reliable data driven decisions and thus should be taken into deep and fulfilling consideration while ensuring that the data collected should be well protected.

Yes as it helps to cope up with disasters

Yes b'coz it helps people

Yes....for better and faster coordination

Yes.

2

Questions Responses 30 Settings

Note any suggestion to improve our current Disaster Response management.

30 responses

no suggestion

Accuracy of information is must

None.

To improve disaster response management, integrating advanced technologies like AI for predictive analysis and real-time data sharing could enhance decision-making. Additionally, strengthening public-private partnerships and investing in community training programs would increase preparedness and coordination during emergencies.

Early actions before disaster hits

2

Questions Responses 30 Settings

Note any suggestion to improve our current Disaster Response management.

30 responses

Add More feature

could probably add an chatbot that can give instant replies to certain common questions

Everything is good no need to improve anything

Better communication

Proper infrastructure and public awareness

Leverage technology for better post

Quick rescue of people and medical help should be provided as soon as possible. Finding lost people.



KEY INSIGHTS OF SURVEY RESULT

The survey results reveal significant insights into public familiarity, perceived value, and willingness to engage with crowdsourced disaster response systems. While most respondents express some familiarity with these systems, a notable portion remains less informed, indicating an opportunity for education and outreach. Encouragingly, 80% of participants believe that crowdsourced systems can enhance disaster response efforts, reflecting a positive perception of their potential benefits.

Engagement levels also show promise, with nearly 49% of respondents indicating a likelihood to report disaster related events through such platforms; however, the 44.8% of neutral respondents highlight the necessity for increased encouragement and demonstration of the system's advantages.

The overwhelming majority—93.1%—perceive these systems as effective in saving lives during disasters, with 43.3% rating their effectiveness as extremely high. Nevertheless, there are notable concerns surrounding data submission comfort, as only 53.3% feel comfortable providing real-time information, and 40% express uncertainty, signaling a critical need for robust privacy measures and transparent data handling practices. Reliability of the crowdsourced data itself also raises questions, with only 43.3% of respondents finding it very reliable, underscoring the necessity for mechanisms that ensure data accuracy and integrity.

Additionally, preferences for data submission platforms skew towards social media and mobile apps, suggesting that integrating these channels could enhance user engagement.

Respondents voiced major concerns regarding data privacy, information accuracy, and technical issues, all of which must be addressed to build user trust and ensure the system's effectiveness. The importance of real-time communication with agencies, local governments, NGOs, and volunteers is underscored by 96.6% of respondents, emphasizing the need for seamless integration and communication features within the system. Furthermore, suggestions for improvement focus on faster data gathering, better coordination, and enhanced decision-making, which should be prioritized in the system's design to maximize effectiveness.

2.3.2 LIST OF REQUIREMENTS GATHERED FROM RESPONSES

1. User Management:

User Profiles: Store user information including name, contact details, and preferences for submitting data.

Authentication: Secure login and authentication system to manage user access and ensure data privacy.

2. Disaster Information Management:

Manage data related to disasters, including location, severity, and types of emergencies.

3. Resource Management:

Manage data related to available resources (food, medical supplies, personnel).

4. Volunteer Management:

Maintain information on volunteers, including their location, availability, and expertise.

2.4 LIST OF REQUIREMENTS GATHERED BY OBSERVATION

1. Incident Data:

Location: Coordinates or addresses of the disaster areas.

Type of Disaster: Earthquake, flood, fire, etc.

Severity: Impact level or damage assessment.

2. Volunteer Information:

Contact Details: Names, phone numbers, email addresses.

Skills and Expertise: Medical training, firefighting experience, etc.

Availability: When users are available to help.

Also include current assignment and deployment status.

3. Resource Information:

Supplies: Availability of food, water, medical supplies. Shelter

Locations: Places offering temporary accommodation.

Source: Donors and suppliers record

4. Victim Information:

Identification: Names, ages, and medical conditions.

Needs: Specific assistance required (e.g., medical attention, evacuation).

5. Agency and Organization data

Participating government agencies

NGOs Contact details and roles of agencies

Contact details of Ambulance facility and availability.

3. FACT FINDING CHART

OBJECTIVE	TECHNIQUE	SUBJECT	TIME COMMITMENT
To get Background on the given Problem Statement and its Solutions	Background Reading	Articles, Journals Websites	2 days(4 hours a day)
To understand the requirements for the disaster management system	Interview(Role Play)	2 groupmates	30 Minutes
To determine People's Expectation and find requirement	Survey/Questionnaire	General Public	3 days(2 hours a day)

DATABASE DESIGN

1. Noun Analysis

Noun	Verb
Disaster	Strike
Consequences	Lead
Communities	Affect
World	Result
Earthquakes	Caused
Floods	Highlight
Hurricanes	Involves
Accidents	Preparing
Attacks	Responding
People	Recovering
Loss	Plays
Life	Minimizing
Disruption	Ensuring
Impacts	Face
Disasters	Emerged
Management	Leveraging
Strategies	collaborate
Approach	Coordinate
Preparation	Facilitate
Numbers	Affected by
Response	Take
Recovery	Allow
Phases	Report
Mitigation	Share
Preparedness	Mobilize
Safety	Helping
Well being	Allocate

Environment	Used
Emergence	Provide
Crowdsourcing	Improving
Intelligence	Fraught
Years	Lead
Resources	Ensuring
Public	Collected
Name	Assist
Role	Collaborate
Strategy	Used for
Awareness	Refers
Efforts	Lack
Forms	Limit
Campaigns	Exclude
Application	Poses
Platforms	Analyze
Individuals	Utilize
Incidents	Arise
Information	Collecting
Effectiveness	Sharing
Cases	Looking
Member	Present
Social media	Improving
Tools	Enhance
Aid organizations	Leveraged
Data	Improve
Coordination	Identifying
NGOs	Inform
Agencies	Fostering
Communities	Developing
Process	Addressing
Challenges	Improve

Obstacles	Increasing
Issue	Minimizing
Information	Adapt
Reliability	Lead
Situation	Address
Influx	Mapping
contact	Ensuring
Sources	Allocated
Misinformation	Coordinated
Confusion	Managed
Decision- making	Access
Divide	Implement
Disparity	Leverage
Access	Assess
Technology	Prioritize
Internet	Deploy
Areas	Report
Countries	Enabled
Portion	Respond
Population	Mapping
Smartphones	Exemplify
Connections	Enhancing
Efforts	Facilitates
Integration	Observed
Frameworks	Deemed
Organization	Faces
Capacity	Make
Analysis	Redirect
Utilization	Address
Privacy	Ensures
Security	Reaches
Collection	Need

Future	Mitigates
Advancements	Associated
Technology	Delayed
Opportunities	Improved
Crowdsourced	Allows
Smartphones	Allocate
Platforms	Becomes
Communication	Directed
Artificial Intelligence	Preventing
Patterns	Filling
Trends	Maximizing
Agencies	Enhances
Companies	Foresee
Communities	Escalate
Strategies	Provide
Challenges	Enabling
Advancements	Anticipating
Resilience	Take
Outcomes	Safeguarding
Data	Fosters
Decision making	Integrating
Crowdsourced data	Collected
Disaster response	Improved
Coordination	Minimizes
Context	Optimizes
Types	Ensuring
Intensity	Saving
Natural disaster	Reducing
Mechanism	Continue
Losses	Harnessing
Disaster management	Enable
Approaches	Act

Conditions	Saving
Delays	Speed up
Resource deployment	Improve
Misallocation	Offer
Efforts	Show
Challenges	Ensure
Integration	Balancing
Strategies	Making
Solution	Maintain
Information	Involved
Flow	Reveal
Effectiveness	Engage
Resources	Express
Volunteers	Remains
Risks	Indicating
Data	Believe
Integration	Involves
Responders	Reflecting
Situation	Show
Information	Highlight
Insights	Perceive
Effects	Saving
Disaster	Rating
Infrastructure damage	Providing
Flooded areas	Express
Population	Signaling
Emergency teams	Raises
Ground situation	Finding
Responses	Underscoring
Example	Ensure
Kerala floods	Suggesting
Citizens	Voiced

Social media platforms	Build
Location	Ensure
Condition	Emphasizing
Volunteers	Focus
Rescue teams	Prioritized
Areas	Maximize
Attention	Consider
Tools	Driven
Situational awareness	Highlight
Interventions	Realize
Resource management	Concerns
Decision making	Help
Adjustment	Contact
Resources	Involve
Conditions	Connect
Crowdsourced data	Used for
Scenarios	
Area	
Flooding	
Responders	
Decisions	
Resources	
Relief supplies	
Needs	
Adaptability	
Aid	
Risks	
Responses	
Volunteer coordination	
Volunteer	
Aspect	
Disaster response	

Platforms	
Tasks	
Needs	
Assistance	
Efforts	
Gaps	
Coverage	
Coordination	
Impact	
Contributions	
Crises	
Emerging risks	
Integration	
Predictive analytics	
Monitoring systems	
Capability	
Decision makers	
Flood modeling	
Responders	
Decisions	
Evacuation orders	
Rerouting	
Resources	
Agencies	
Actions	
Lives	
Property	
Collaborative disaster response	
Decision making	
Collaboration	
Agencies	
Government bodies	

Severity	
Crowdsourced data	
Public	
Disaster response frameworks	
Coordination	
Government agencies	
Approach	
Duplication	
Efforts	
Resource utilization	
Implementation	
Decision making	
Disaster response coordination	
Threats	
Communities	
Responders	
Lives	
Impact	
Catastrophic Events	
Real time	
Data collection	
Disaster response	
Involvement	
Local population	
Response times	
Accuracy	
Type	
Insights	
System improvements	
Effectiveness	
Crowdsourcing	
Data validation mechanism	

Reliability	
Information flow	
Updates	
User friendliness	
Decision making	
Success	
System	
Quick decision making	
Emergency situation	
Data security	
Priority	
Trust	
Name	
Security	
Integrity	
Requirement	
Database	
Location	
Victim information	
Resources	
Agency	
Organization's data	
Survey results	
Insights	
Public familiarity	
Perceived value	
Willingness	
Respondents	
Portion	
Opportunity	
Outreach	
Participants	

Systems	
Efforts	
Perception	
Benefits	
Date	
Respondents	
Likelihood	
Platforms	
Neutral respondents	
Encouragement	
Demonstration	
System's advantages	
Disasters	
Data submission	
Information	
Privacy measures	
Data handling practices	
Reliability	
Questions	
Mechanisms	
Accuracy	
Integrity	
Preferences	
Data submission platforms	
Social media	
Mobile apps	
Channels	
User engagement	
Data privacy	
Information accuracy	
Technical issues	
Trust	

Effectiveness	
Communication	
Agencies	
Governments	
NGOs	
Volunteers	
Communication features	
System	
Suggestions	
Improvement	
Data gathering	
Coordination	
Decision making	
Design	
Effectiveness	
Mandate	
Companies	
Disaster management	
Insights	
Promise	
Challenges	
Implementation	
Strategic planning	
Community engagement	
Potential	
Engagement levels	
Historical cases	
Availability	
Skillset	
Reliability	
Type	
Strategies	

Rejected Noun List

Noun	Reject Reason
World	Too general
Earthquake	Specific Instance of Disaster, Grouped under 'Disaster'
Hurricane	Specific Instance of Disaster, Grouped under 'Disaster'
Accident	Specific Instance of Disaster, Grouped under 'Disaster'
Attack	Specific Instance of Disaster, Grouped under 'Disaster'
Loss	Too vague
Life	Too general
Well being	Attribute of People/Victim
Consequences	Vague, can be attribute related to disaster
Year	Could be an attribute, not entity itself
Tools	Too general, can be attributed to technologies
Influx	Not an entity
Divide	Vague and general
Efforts	General, can be attribute of volunteers
Smartphone	Under 'technology'
Platform	Too general, attribute under 'social media platform'
Flow	Better treated as relationship
Communication Features	Too specific
Privacy	Attribute related to data
Security	Attribute related to data or system

Framework	Can be part of the system's structure, not a core entity
Conditions	General term
Future	Not an entity
Opportunities	Too general
Access	Can be attribute of resources
Reliability	Attribute of Data or System
Challenges	Attribute of Management or System
Success	Not an entity
Advancement	Too vague, can be attribute of technology
Trust	Attribute related to privacy, security, or the relationship between system and user
Ground Situation	Too specific
Example	General
Kerala Floods	Specific event, not entity
Citizens	Grouped under 'victim' or 'volunteer'
Infrastructure damage	Attribute, can be part of Disaster
Emergency teams	Already covered under 'teams'
Rescue teams	covered under 'emergency teams'
Connection	Attribute of 'communication'
Locatiion	Attribute rather than entity
Situation	Too vague
Information flow	Relationship
Communication	General term, can be relation
Interventions	Not an entity
Database	Part of system
Insight	Attribute, not an entity
Availability	Attribute of volunteers
Public	Vague
Victim Information	Attribute under victim
Perception	Not an entity
System Improvements	Can be part of system
Data collections	Process

System	General term
Communication features	specific
Questions	Not an entity
Mechanism	Not an entity, can be part of system
Requirements	Can be part of system
Encouragement	Not an aentity
Mobile apps	Grouped under 'technologies'
Data submission	Process, not an entity
User engagement	Part of user feedback
Technical Issue	Attribute under 'system'
Design	Attribute under 'system'
Mandate	Part of process, not an enity
Promise	Vague
Strategic planning	Process
Skill set	Attribute of volunteers

Accepted Entities

Candidate Entity Set	Candidate Attribute Set	Candidate Relationship Set
Disaster	Date, Type, Severity, Location	Involves, affects, impact, used for
Communities	Community_ID, Population, Location	Affected by, Supports, contact
Volunteers	Name, Contact no, Availability, Location, Skillset	Assists, Coordinates, connect, help
Agencies	ID, Name, Type, Role, contact_info	Collaborate, coordinate, involve, contact
Resources	Resource_ID, Type, Availability, Source, Location	Allocated, shared, used
Social Media Platform	Name, Type	Used for, collects, utilize
Response	ID, Disaster_date, Disaster_Location, start_date, End_date, count, status/phase	Enhance, useful, connect, involve
Emergency teams	Team_ID, Members, Availability	Collaborate, allocated by, collect
NGOs	ID, Name, contact	Coordinates, Involved, contact
Victims	Name, Contact_no, Location, Needs	Helped, Affected, supported by, impact, helped by

2. SCHEMA AND ER DIAGRAM DESIGN

1. Disaster (Disaster_ID, Date, Location, Type, Severity)

Cardinality:

Disaster – Communities(Many to Many)

Disaster – Volunteers(Many to Many)

Disaster – Agencies(Many to Many)

Disaster – victim (Many to Many)

Disaster – Social Media(Many to Many)

Disaster – Response(One to Many)

Disaster – NGOs(Many to Many)

2. Communities (pin code, Location, Population)

Cardinality:

communities – Volunteers (Many to Many)

communities – Agencies (Many to Many)

communities – NGOs (Many to Many)

communities – victim (one to Many)

3. Volunteer (Volunteer_ID, Name, Contact_no, age, location, availability, Skillset)

Cardinality:

Volunteer – Victims (Many to Many)

Volunteer – Emergency_team (One to Many)

4. Agencies (Agency_ID, Name, Type, Role, contact_info)

Cardinality:

Agencies – Volunteer(one to Many)

Agencies – NGOs(Many to Many)

Agencies – Emergency Teams(One to Many)

5. Resources (Resource_Id, Type, Availability, location, source)

Cardinality:

resources – Emergency teams(Many to many)

6. Social Media Platform (Name, Type)

Cardinality:

Social Media Platform – Disaster(Many to Many)

7. Response(Response_ID, start_date, end date, phase/status)

Cardinality:

Disaster – Response(one to many)

Response – Victims(Many to Many)

Response – Emergency teams(One to Many)

8. Emergency teams (Team_ID, no_of_members, Availability)**Cardinality:**

Emergency teams – Resources(Many to Many)

Emergency teams – Agencies(One to Many)

Emergency teams – Response(One to Many)

9. NGOs (NGO_ID, Name, Contact, service)**Cardinality:**

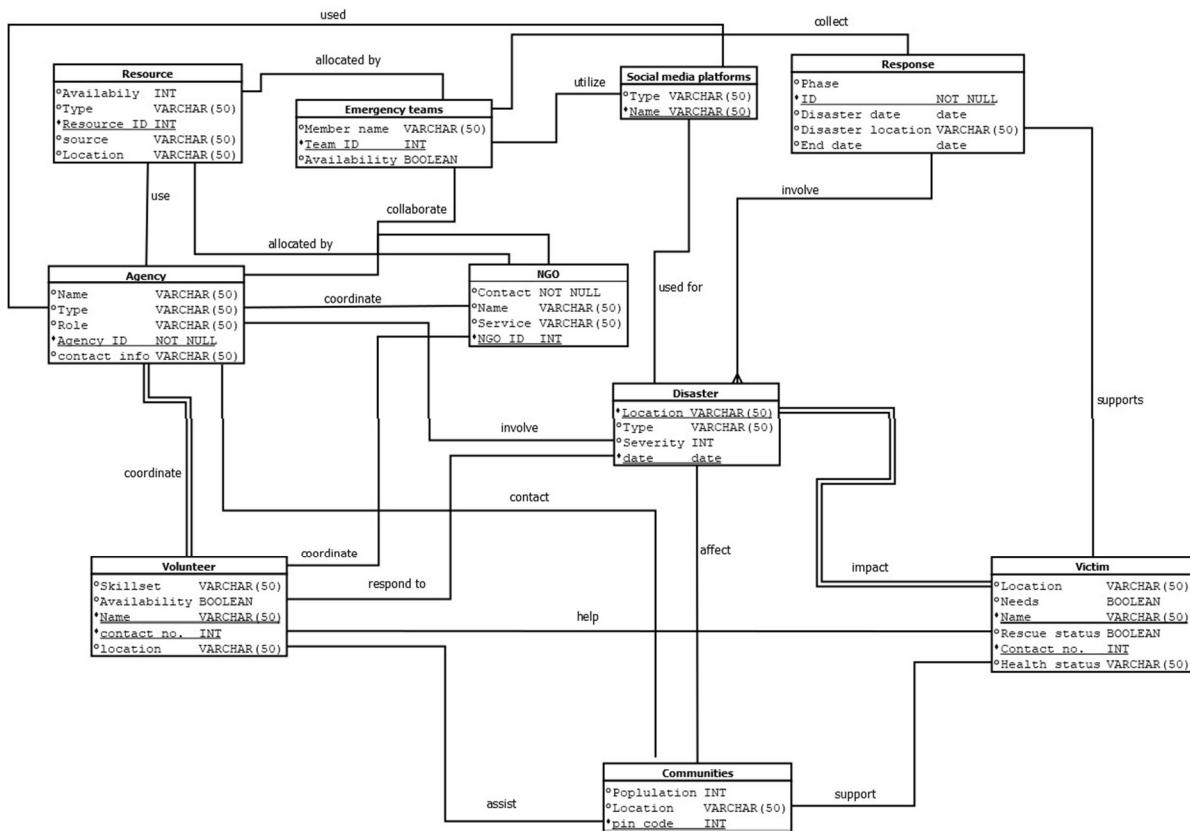
NGOs – Emergency Teams(One to Many)

10. Victims (Victim_ID, Name, Contact_no, Location, Health_status, Rescue_status, needs)**Cardinality:**

Victim – communities (One to many)

Victim – responses (Many to Many)

ER DIAGRAM



3. ER DIAGRAM ANALYSIS

- **Weak entities:**

As none of the entities rely on another entity for the Primary key and have their own unique Identifier, there is no weak entity. which means all entities are Strong entities.

- **Foreign keys**

1. Communities

Disaster_ID (from Disaster)

2. Volunteers

Disaster_ID (from Disaster)

Pincode,Location (Communities)

Agency_ID (Agencies)

NGO_ID (NGOs)

3. Agencies

Disaster_ID (Disaster)

NGO_ID (NGOs)

4. Resources

Agency_ID (Agencies)

NGO_ID (NGOs)

Team_ID (Emergency_teams)

5. Social Media Platform

Disaster_ID (Disaster)

6. Response

Disaster_ID (Disaster)

Team_ID (Emergency_team)

7. Emergency Teams

Agency_ID (agencies)

Ngo_ID (NGOS)

8. NGOs

Disaster_ID (Disaster)

9. Victims

Disaster_ID (Disaster)

Volunteer_Name,contact_no (Volunteers)

Response_ID (Responses)

- **Binary Relationships:**

-All the relations defined in the above schema have binary relationship.

-One possible ternary relationship was Agencies- Emergency teams- Resources but it is modelled as Binary relationship for simplicity.

-Other than that Volunteers – NGOs/Agencies – EmergencyTeams can have ternary relationship.

- **Aggregation**

No aggregation used in this model.

- **Recursive Relationships**

Recursive relationships occur when an entity is related to itself, indicating a relationship within the same entity type. But there is no such scenario for this model.

- **Participation**

- 1) Disaster:

Total participation with Victims (every disaster has victims).

Total participation with Response (every disaster must have a response).

Partial participation with Communities, Volunteers, Agencies, NGOs, and Social Media (not every community, volunteer, agency, or NGO needs to be involved in every disaster).

- 2) Communities:

Total participation with Victims (if every victim is from a community).

Partial participation with Volunteers, Agencies, and NGOs (not all communities will have volunteers or agencies).

- 3) Volunteers:

Partial participation with Disaster (not all volunteers will participate in every disaster).

Partial participation with Victims (only some volunteers may assist victims).

- 4) Agencies:

Partial participation with Disaster (not all agencies are involved in every disaster).

Partial participation with Volunteers (not all agencies will have volunteers).

5) Resources:

Partial participation with Emergency Teams (not all resources will be allocated to emergency teams).

6) Social Media Platform:

Partial participation with Disaster (not all disasters will be associated with every social media platform).

7) Response:

Total participation with Disaster (every response corresponds to a disaster).

Partial participation with Victims and Emergency Teams (not all responses will involve every victim or team).

8) Emergency Teams:

Partial participation with Response (not every emergency team is involved in every response).

9) NGOs:

Partial participation with Disaster (not all NGOs are involved in every disaster).

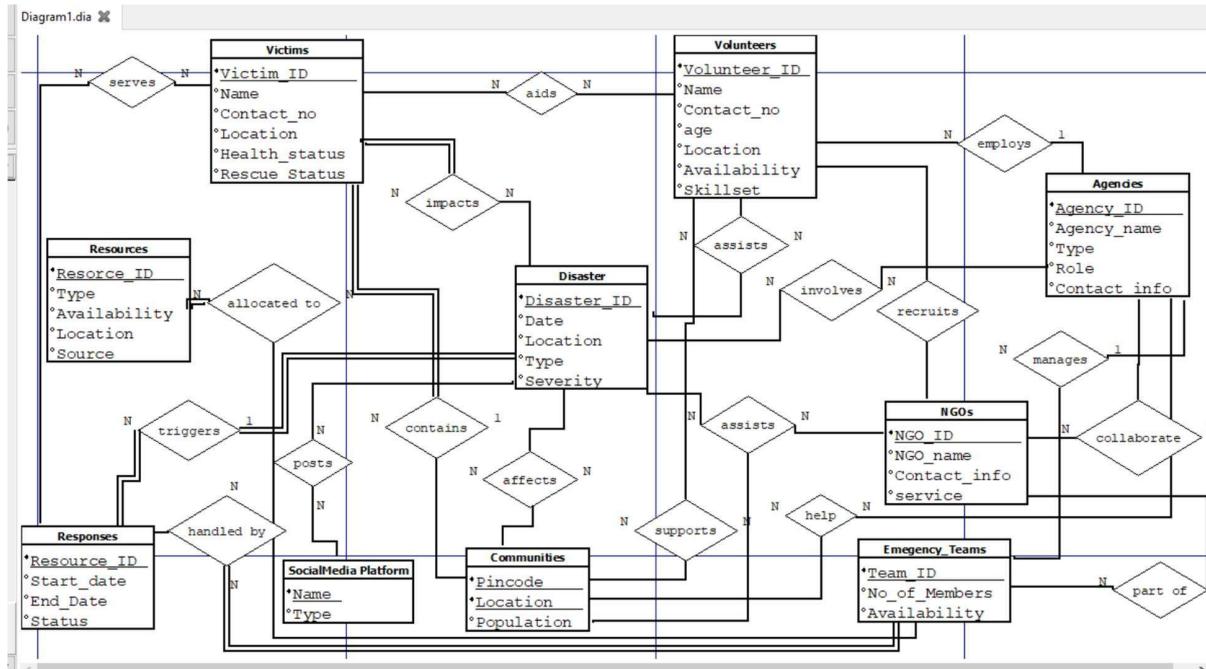
Partial participation with Volunteers (not every NGO will have volunteers).

10) Victims:

Total participation with Disaster (every victim corresponds to a disaster).

Partial participation with Volunteers (not every victim will have a volunteer assigned).

FINAL ER DIAGRAM



4. MAPPING ER MODEL TO RELATIONAL MODEL

Entity , Attributes and Relation

1. Disaster (Disaster_ID, Date, Location, Type, Severity)

Disaster – Communities(Many to Many)

Disaster – Volunteers(Many to Many)

Disaster – Agencies(Many to Many)

Disaster – victim (Many to Many)

Disaster – Social Media(Many to Many)

Disaster – Response(One to Many)

Disaster – NGOs(Many to Many)

2. Communities (pin_code, Location, Population)

communities – Volunteers (Many to Many)

communities – Agencies (Many to Many)

communities – NGOs (Many to Many)

communities – victim (one to Many)

3. Volunteer (Volunteer_ID, Name, Contact_no, age, location, availability, Skillset)

Volunteer – Victims (Many to Many)

Volunteer – Emergency_team (One to Many)

Volunteer – NGO (many to Many)

4. Agencies (Agency_ID, Name, Type, Role, contact_info)

Agencies – Volunteer(one to Many)

Agencies – NGOs(Many to Many)

Agencies – Emergency Teams(One to Many)

5. Resources (Resource_Id, Type, Availability, location, source)
resources – Emergency teams(Many to many)

6. Social Media Platform (Name, Type)
Social Media Platform – Disaster(Many to Many)

7. Response(Response_ID, start_date, end date, phase/status)
Disaster – Response(one to many)
Response – Victims(Many to Many)
Response – Emergency teams(One to Many)

8. Emergency teams (Team_ID, no_of_members, Availability)
Emergency teams – Resources(Many to Many)
Emergency teams – Agencies(One to Many)
Emergency teams – Response(One to Many)

9. NGOs (NGO_ID, Name, Contact, service)
NGOs – Emergency Teams(One to Many)

10. Victims (, Name, Contact_no, Location,
Health_status, Rescue_status)
Victim – communities (One to many)
Victim – responses (Many to Many)

SCHEMA

1. Disaster(Disaster_ID, date, location, type, severity)
2. Community(Pincode,Location, population)
3. Volunteer(Volunteer_ID, Name, contact_no, age, location, availability)
FK – Agency_ID, NGO_ID
4. Agency(Agency_ID, name, type, role, contact_info)
5. Resources(Resource_ID, type, availability, location, source)
6. Social_Media_Platform(name, type)
7. Response(Response_ID, start_date, end_date, status)
FK – Disaster_ID, EmergencyTeam_ID
8. Emergency_team(team_ID, no_of_members, Availability)
FK – Agency_ID, NGO_ID
9. NGO(NGO_ID, name, contact, service)
10. Victim(Victim_ID, name, contact_no, location, health_status, rescue_status)
FK – community_pincode, community_location
11. Disaster_Communities(Disaster_ID, Community_pincode, community_location)
FK - Disaster_ID, Community_pincode, community_location

12. Disaster_Volunteer(Disaster_ID, Volunteer_ID)

FK - Disaster_ID, Volunteer_ID

13. Disaster_Agencies(Disaster_ID, Agency_ID)

FK - Disaster_ID, Agency_ID

14. Disaster NGO(Disaster_ID, NGO_ID)

FK - Disaster_ID, NGO_ID

15. Disaster_Victim(Disaster_ID, victim_ID)

FK - Disaster_ID, victim_ID

16. Disaster_SocialMedia(Disaster_ID, PlatformName)

FK - Disaster_ID, PlatformName

17. Communities_Volunteer(Community_pincode, community_location,
Volunteer_ID)

FK - Community_pincode, community_location, Volunteer_ID

18. Communities_Agency(Community_pincode, community_location,
Agency_ID)

FK - Community_pincode, community_location,
Volunteer_ID

19. Communities NGO(Community_pincode, community_location,
NGO_ID)

FK - Community_pincode, community_location, NGO_ID

20. Volunteer_Victim(Volunteer_ID, Victim_ID)

FK - Volunteer_ID, Victim_ID

21. Agency NGO(Agency_ID, NGO_ID)

FK - Agency_ID, NGO_ID

22. EmergencyTeam_Resources(Team_ID, Resource_ID)

FK - Team_ID, Resource_ID

23. Response_Victim(Victim_ID, Response_ID)

FK - Victim_ID, Response_ID

5. DDL SCRIPTS

```
CREATE DATABASE crowdsourcedresponse;
```

```
CREATE TABLE Disaster (
    Disaster_ID INT PRIMARY KEY,
    Date DATE NOT NULL,
    Location VARCHAR(50) NOT NULL,
    Type VARCHAR(50) NOT NULL,
    Severity INT CHECK(1 TO 5));
```

```
CREATE TABLE Communities(
    Pincode INT ,
    Location VARCHAR(50) ,
    Population INT NOT NULL,
    PRIMARY KEY(Pincode, Location));
```

```
CREATE TABLE Volunteers(
    Volunteer_ID INT PRIMARY KEY,
    Volunteer_name VARCHAR(50) NOT NULL,
    Contact_no INT NOT NULL, UNIQUE,
    Age INT NOT NULL, CHECK(AGE>0),
```

```
Location VARCHAR(50) NOT NULL,  
Availability BOOLEAN DEFAULT TRUE,  
Skillset TEXT NOT NULL,  
NGO_ID INT,  
Agency_ID INT,  
FOREIGN KEY (ng_id) REFERENCES NGO(NGO_ID),  
FOREIGN KEY (ag_id) REFERENCES Agency(Agency_ID));
```

```
CREATE TABLE Agencies(  
Agency_ID INT PRIMARY KEY,  
Name VARCHAR(50) ,  
Type VARCHAR,  
Role TEXT NOT NULL,  
Contact_info VARCHAR NOT NULL, UNIQUE);
```

```
CREATE TABLE Resources(  
Resource_ID INT PRIMARY KEY,  
Type VARCHAR,  
Availability BOOLEAN DEFAULT TRUE,  
Location VARCHAR,  
Source VARCHAR);
```

```
CREATE TABLE SocialMediaPlatform(  
    Name VARCHAR PRIMARY KEY,  
    Type VARCHAR);
```

```
CREATE TABLE Response(  
    Response_ID INT PRIMARY KEY,  
    Start_date DATE NOT NULL,  
    End_date DATE NOT NULL,  
    Status VARCHAR ,  
    Disaster_ID INT,  
    EmergencyTeam_ID INT,  
    FOREIGN KEY (Disaster_ID) REFERENCES Disaster(Disaster_ID),  
    FOREIGN KEY (EmergencyTeam_ID) REFERENCES  
    EmergencyTeam(EmergencyTeam_ID));
```

```
CREATE TABLE EmergenctTeams(  
    Team_ID INT PRIMARY KEY,  
    No_of_Members INT NOT NULL, CHECK(>0),  
    Availability BOOLEAN DEFAULT TRUE,  
    NGO_ID INT,  
    Agency_ID INT,  
    FOREIGN KEY (NGO_ID) REFERENCES NGO(NGO_ID),
```

```
FOREIGN KEY (Agency_ID) REFERENCES Agency(Agency_ID));
```

```
CREATE TABLE NGOs(  
    NGO_ID INT PRIMARY KEY,  
    NGO_name VARCHAR NOT NULL,  
    Contact_info VARCHAR ,  
    Service VARCHAR);
```

```
CREATE TABLE victims(  
    Victim_ID INT PRIMARY KEY,  
    Victim_name VARCHAR,  
    Contact_no INT ,  
    Location VARCHAR NOT NULL,  
    Health_status VARCHAR NOT NULL,  
    Rescue_status VARCHAR NOT NULL,  
    Community_pincode INT ,  
    Community_Location VARCHAR,  
    FOREIGN KEY (Community_pincode) REFERENCES  
    Communities(pincode),  
    FOREIGN KEY (Community_Location) REFERENCES  
    Communities(Location));
```

MANY TO MANY RELATIONSHIP TABLES

```
CREATE TABLE Disaster_Communities(  
Disaster_ID INT PRIMARY KEY,  
Community_pincode INT,  
Community_Location INT,  
PRIMARY KEY (Community_pincode, Community_Location),  
FOREIGN KEY (Disaster_ID) REFERENCES Disaster(Disaster_ID),  
FOREIGN KEY (Community_pincode) REFERENCES  
Communities(Community_pincode),  
FOREIGN KEY (Community_Location) REFERENCES Communities  
(Community_Location));
```

```
CREATE TABLE Disaster_Volunteer(  
Disaster_ID INT ,  
Volunteer_ID INT,  
PRIMARY KEY(Disaster_ID, Volunteer_ID),  
FOREIGN KEY ( Disaster_ID ) REFERENCES Disaster (Disaster_ID),  
FOREIGN KEY (Volunteer_ID) REFERENCES Volunteer (Volunteer_ID));
```

```
CREATE TABLE Disaster_Agencies(  
Disaster_ID INT ,
```

```
Agency_ID INT,  
PRIMARY KEY (Disaster_ID, Agency_ID),  
FOREIGN KEY ( Disaster_ID) REFERENCES Disaster (Disaster_ID),  
FOREIGN KEY (Agency_ID) REFERENCES Agency (Agency_ID));
```

```
CREATE TABLE Disaster_NGO(  
Disaster_ID INT ,  
NGO_ID INT,  
PRIMARY KEY(Disaster_ID,NGO_ID),  
FOREIGN KEY ( Disaster_ID) REFERENCES Disaster (Disaster_ID),  
FOREIGN KEY (NGO_ID) REFERENCES NGOs(NGO_ID));
```

```
CREATE TABLE Disaster_Victim(  
Disaster_ID INT ,  
Victim_ID INT,  
PRIMARY KEY(Disaster_ID, Victim_ID),  
FOREIGN KEY ( Disaster_ID) REFERENCES Disaster (Disaster_ID),  
FOREIGN KEY (Victim_ID) REFERENCES Victim(Victim_ID));
```

```
CREATE TABLE Disaster_SocialMediaPlatform(  
Disaster_ID INT ,  
Platformname VARCHAR,
```

PRIMARY KEY(Disaster_ID,Platformname),
FOREIGN KEY (Disaster_ID) REFERENCES Disaster (Disaster_ID),
FOREIGN KEY (Platformname) REFERENCES SocialMediaPlatform
(Platformname));

CREATE TABLE Communities_Volunteer(
Volunteer_ID INT PRIMARY KEY,
Community_pincode INT,
Community_Location INT,
PRIMARY KEY (Community_pincode, Community_Location),
FOREIGN KEY (Volunteer_ID) REFERENCES Volunteer(Volunteer_ID),
FOREIGN KEY (Community_pincode) REFERENCES
Communities(Community_pincode),
FOREIGN KEY (Community_Location) REFERENCES Communities
(Community_Location));

CREATE TABLE Communities_Agency(
Agency_ID INT PRIMARY KEY,
Community_pincode INT,
Community_Location INT,
PRIMARY KEY (Community_pincode, Community_Location),
FOREIGN KEY (Agency_ID) REFERENCES Agency(Agency_ID),

```
FOREIGN KEY (Community_pincode) REFERENCES  
Communities(Community_pincode),  
FOREIGN KEY (Community_Location) REFERENCES Communities  
(Community_Location));
```

```
CREATE TABLE Communities_NGO(  
NGO_ID INT PRIMARY KEY,  
Community_pincode INT,  
Community_Location INT,  
PRIMARY KEY (Community_pincode, Community_Location), FOREIGN  
KEY (NGO_ID) REFERENCES NGO(NGO_ID),  
FOREIGN KEY (Community_pincode) REFERENCES  
Communities(Community_pincode),  
FOREIGN KEY (Community_Location) REFERENCES Communities  
(Community_Location));
```

```
CREATE TABLE Volunteer_Victim(  
Volunteer_ID INT,  
Victim_ID INT,  
PRIMARY KEY(Volunteer_ID, Victim_ID),  
FOREIGN KEY (Volunteer_ID) REFERENCES Volunteer(Volunteer_ID),  
FOREIGN KEY (Victim_ID) REFERENCES Victim(Victim_ID));
```

```
CREATE TABLE Agencies_NGOs(  
    NGO_ID INT,  
    Agency_ID INT ,  
    PRIMARY KEY(NGO_ID, Agency_ID),  
    FOREIGN KEY (Agency_ID) REFERENCES Agency(Agency_ID),  
    FOREIGN KEY (NGO_ID) REFERENCES NGO(NGO_ID));
```

```
CREATE TABLE EmergencyTeams_Resources(  
    Team_ID INT,  
    Resource_ID INT,  
    PRIMARY KEY(Team_ID, Resource_ID),  
    FOREIGN KEY(Team_ID) REFERENCES EmergencyTeams(Team_ID),  
    FOREIGN KEY(Resource_ID) REFERENCES Resources(Resource_ID));
```

```
CREATE TABLE Response_Victims(  
    Response_ID INT,  
    Victim_ID INT  
    PRIMARY KEY(Response_ID, Victim_ID),  
    FOREIGN KEY(Victim_ID) REFERENCES Victims(Victim_ID),  
    FOREIGN KEY(Response_ID) REFERENCES Response(Response_ID));
```

NORMALIZATION OF DATABASE

1. NORMALIZATION AND SCHEMA REFINEMENT

1.1 ORIGINAL DESIGN OF DATABASE

1. Disaster(Disaster_ID, date, location, type, severity)
2. Community(Pincode,Location, population)
3. Volunteer(Volunteer_ID, Name, contact_no, age, location, availability)
FK – Agency_ID, NGO_ID
4. Agency(Agency_ID, name, type, role, contact_info)
5. Resources(Resource_ID, type, availability, location, source)
6. Social_Media_Platform(name, type)
7. Response(Response_ID, start_date, end_date, status)
FK – Disaster_ID, EmergencyTeam_ID
8. Emergency_team(team_ID, no_of_members, Availability)
FK – Agency_ID, NGO_ID
9. NGO(NGO_ID, name, contact, service)
10. Victim(Victim_ID, name, contact_no, location, health_status, rescue_status)
FK – community_pincode, community_location

11. Disaster_Communities(Disaster_ID, Community_pincode,
community_location)

FK - Disaster_ID, Community_pincode, community_location

12. Disaster_Volunteer(Disaster_ID, Volunteer_ID)

FK - Disaster_ID, Volunteer_ID

13. Disaster_Agencies(Disaster_ID, Agency_ID)

FK - Disaster_ID, Agency_ID

14. Disaster_NGO(Disaster_ID, NGO_ID)

FK - Disaster_ID, NGO_ID

15. Disaster_Victim(Disaster_ID, victim_ID)

FK - Disaster_ID, victim_ID

16. Disaster_SocialMedia(Disaster_ID, PlatformName)

FK - Disaster_ID, PlatformName

17. Communities_Volunteer(Community_pincode, community_location,
Volunteer_ID)

FK - Community_pincode, community_location, Volunteer_ID

18. Communities_Agency(Community_pincode, community_location,
Agency_ID)

FK - Community_pincode, community_location,
Volunteer_ID

19. Communities_NGO(Community_pincode, community_location,
NGO_ID)

FK - Community_pincode, community_location, NGO_ID

20. Volunteer_Victim(Volunteer_ID, Victim_ID)

FK - Volunteer_ID, Victim_ID

21. Agency_NGO(Agency_ID, NGO_ID)

FK - Agency_ID, NGO_ID

22. EmergencyTeam_Resources(Team_ID, Resource_ID)

FK - Team_ID, Resource_ID

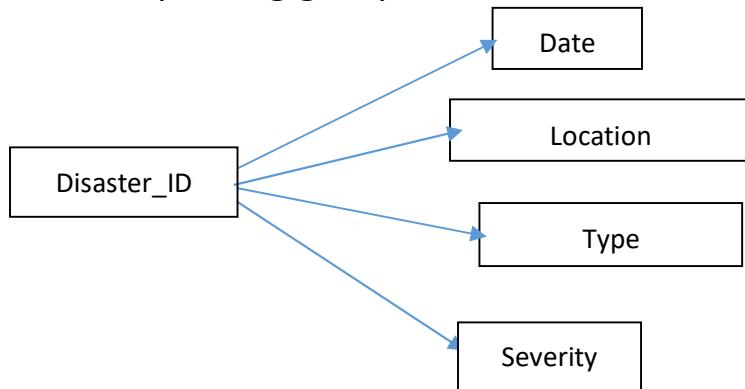
23. Response_Victim(Victim_ID, Response_ID)

FK - Victim_ID, Response_ID

2. DEPENDENCY, REDUNDANCIES, ANOMALIES ANALYSIS

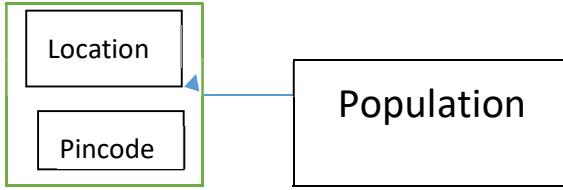
1. Disaster

- **Dependencies:**
 - PKD: Disaster_ID
 - FD: Disaster_ID → (Date, Location, Type, Severity) ◦ No partial, transitive, or multivalued dependencies.
- **Anomalies:** None. The structure is straightforward, and there are no repeating groups



2. Communities

- **Dependencies:**
 - PKD: (Pincode, Location)
 - FD: (Pincode, Location) → Population/ None
No partial, transitive, or multivalued dependencies.
- **Anomalies:** None. Each community is uniquely identified, and there are no multivalued attributes.



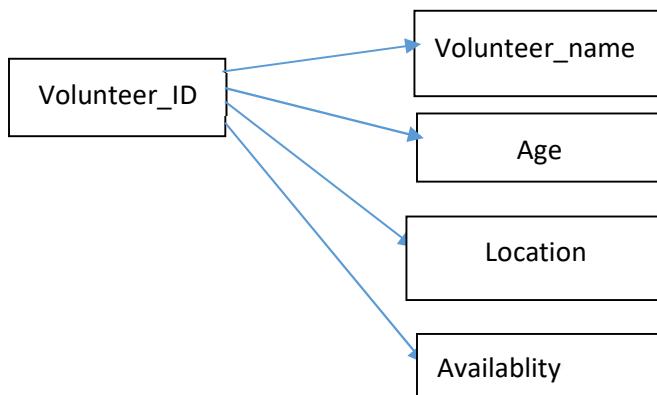
3. Volunteers

- **Dependencies:**

- PKD, FD: $\text{Volunteer_ID} \rightarrow (\text{Volunteer_name}, \text{Age}, \text{Location}, \text{Availability}, \text{NGO_ID}, \text{Agency_ID})$

Anomalies:

- Insertion Anomaly: A volunteer can be added without providing Agency_ID and NGO_ID since they can be NULL. Nonetheless if a volunteer is not associated with any agency or NGO, it is fine.

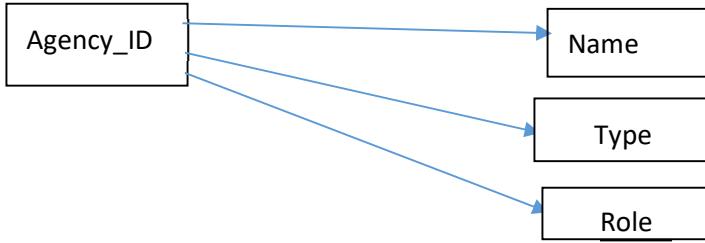


4. Agencies

- **Dependencies:**

PKD, FD: $\text{Agency_ID} \rightarrow (\text{Name}, \text{Type}, \text{Role})$

- **Anomalies:** None. Agency is uniquely identified, and there are no multivalued attributes.



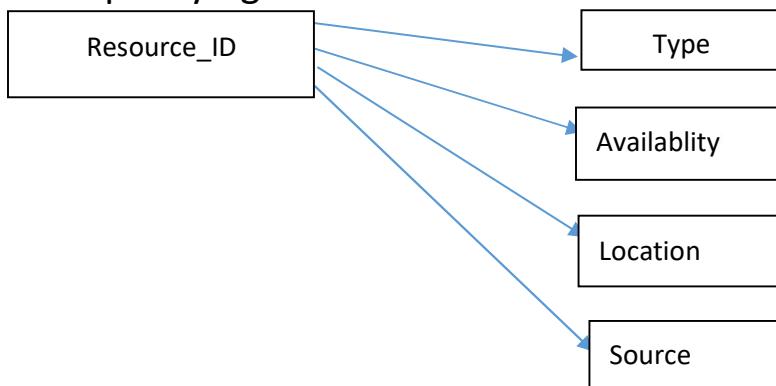
5. Resources

- **Dependencies:**

- PKD, FD: $\text{Resource_ID} \rightarrow (\text{Type}, \text{Availability}, \text{Location}, \text{Source})$

Anomalies:

- Insertion Anomaly: Cannot add resources without specifying all attributes **if it has not null constraint.**



6. Social Media Platform

- **Dependencies:** PKD, FD: $\text{Name} \rightarrow \text{Type}$ (None)

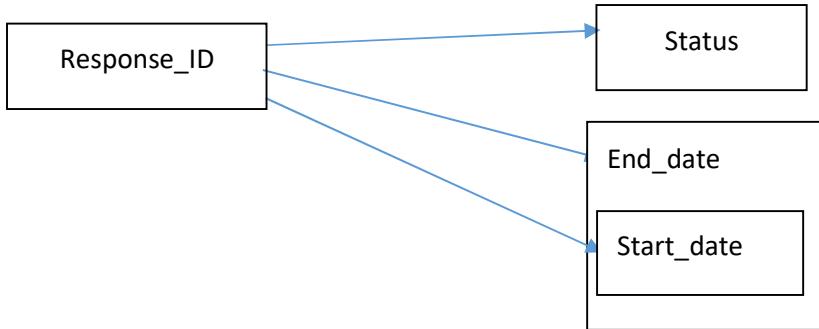
Anomalies:

None



7. Response

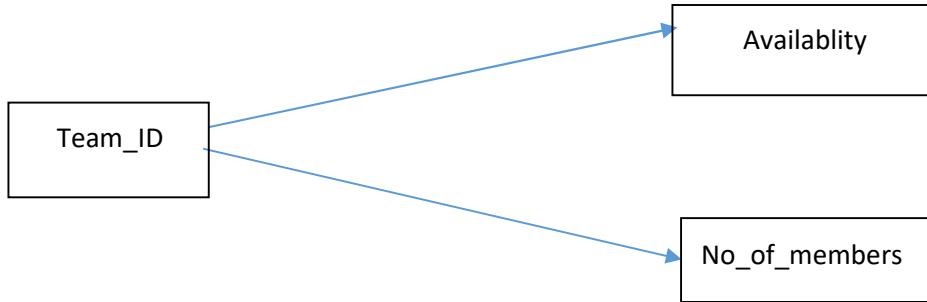
- **Dependencies:** PKD: Response_ID → (Disaster_ID, EmergencyTeam_ID, Start_date, End_date, Status)
- **Anomalies:**
 - Insertion Anomaly: A response cannot be added without linking it to a disaster and emergency team.
 - Update Anomaly: If the status of a response changes, it must be updated for all instances.
 - Deletion Anomaly: Removing a response will also remove links to related disasters and teams.



8. Emergency Teams

- **Dependencies:**

PKD, FD: Team_ID → (Agency_ID, NGO_ID, No_of_Members, Availability)
- **Anomalies:**
 - Insertion Anomaly: A team cannot be created without specifying an agency and/or NGO.
 - Update Anomaly: Changing the number of members requires updating all relevant team records.
 - Deletion Anomaly: Deleting a team removes associated data.

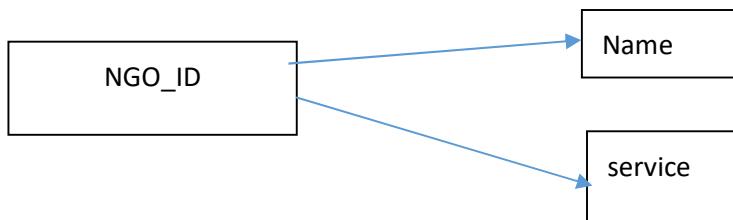


9. NGOs

- **Dependencies:** PKD, FD: $\text{NGO_ID} \rightarrow (\text{NGO_name}, \text{Service})$
- **Anomalies:**

Insertion Anomaly: Cannot add an NGO without all attributes.

Deletion Anomaly: Removing an NGO(ID) deletes all associated data.



10. Victims

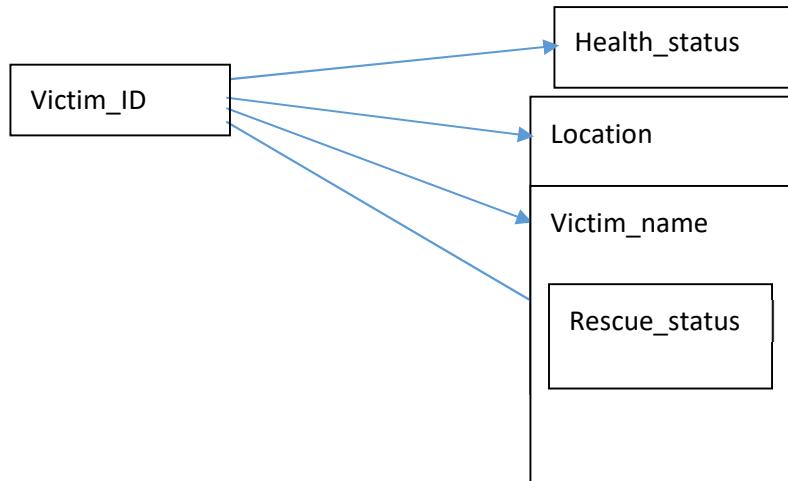
- **Dependencies:**

PKD, FD: $\text{Victim_ID} \rightarrow (\text{Victim_name}, \text{Location}, \text{Health_status}, \text{Rescue_status}, \text{Community_Pincode}, \text{Community_Location})$

Anomalies:

Insertion Anomaly: Cannot add a victim without specifying all attributes.

Deletion Anomaly: Deleting a victim may lead to loss of associated data.



11. Disaster Communities

- **Dependencies:**

PKD, FD: (Disaster_ID, Community_Pincode,
 Community_Location) \rightarrow NULL

- **Anomalies:**

Insertion Anomaly: Cannot add a disaster-community relation without all attributes.

Update Anomaly: Changes to disaster or community details require updating all relevant records.

Deletion Anomaly: Removing a community relation deletes all associated data.

12. Disaster Volunteers

- **Dependencies:** PKD, FD: (Disaster_ID, Volunteer_ID) → NULL

Anomalies:

Insertion Anomaly: Cannot link a volunteer to a disaster without specifying both IDs.

Update Anomaly: Updating volunteer or disaster details requires careful consideration of all links.

Deletion Anomaly: Deleting this relation may remove important volunteer records.

13. Disaster Agencies

- **Dependencies:** PKD: (Disaster_ID, Agency_ID) → NULL
- **Anomalies:**

Insertion Anomaly: Cannot link an agency to a disaster without specifying both IDs.

Update Anomaly: Changes in agency or disaster details require updating all relevant links.

Deletion Anomaly: Deleting an agency link may impact disaster records.

14. Disaster NGOs

- **Dependencies:** PKD, FD: (Disaster_ID, NGO_ID) → NULL
- **Anomalies:**
 - Insertion Anomaly: Cannot link an NGO to a disaster without specifying both IDs.
 - Update Anomaly: Changes in NGO details require careful updating of all links.

- Deletion Anomaly: Deleting an NGO link may affect disaster data.

15. Disaster Victims

- **Dependencies:**

PKD, FD: (Disaster_ID, Victim_ID) → NULL

Anomalies:

- Insertion Anomaly: Cannot link a victim to a disaster without both IDs.
- Update Anomaly: Changes to victim details require updating all links.

16. Disaster Social Media

- **Dependencies:** PKD, FD: (Disaster_ID, PlatformName) → NULL

Anomalies:

- Insertion Anomaly: Cannot link a social media platform to a disaster without both IDs.
- Update Anomaly: Changes in platform details require updating all links.
- Deletion Anomaly: Deleting this link may remove relevant disaster information.

17. Communities Volunteers

- **Dependencies:**

PKD, FD: (Community_Pincode, Community_Location, Volunteer_ID) → NULL

Anomalies:

- Insertion Anomaly: Cannot link a volunteer to a community without all attributes.
- Update Anomaly: Changes in volunteer or community details need careful updating.

18. Community_Agency

- **Dependencies:**
- PKD, FD: (Community_Pincode, Community_Location, Agency_ID) → NULL

Anomalies:

Insertion Anomaly: Cannot link an agency to a community without all attributes.

Update Anomaly: Changes to agency or community details require updates.

19. Community_NGO

- **Dependencies:**
 - PKD, FD: (Community_Pincode, Community_Location, NGO_ID) → NULL
- **Anomalies:**

Insertion Anomaly: Cannot link an NGO to a community without specifying all attributes.

Update Anomaly: Changes in NGO or community details need to be updated.

20. Volunteer_Victim

- **Dependencies:** PKD, FD: (Volunteer_ID, Victim_ID) → NULL

Anomalies:

Insertion Anomaly: Cannot link a volunteer to a victim.

Update Anomaly: Changes in volunteer or victim details require updating all links.

21. Agencies NGOs

- **Dependencies:**

PKD: (Agency_ID, NGO_ID)

FD: (Agency_ID, NGO_ID) → NULL

Anomalies: Insertion Anomaly: Cannot link an agency to an NGO without both IDs.

Update Anomaly: Changes in agency or NGO details require careful updating.

Deletion Anomaly: Deleting an agency-NGO link may result in loss of information.

Emergency Teams Resources

Dependencies: PKD, FD: (Team_ID, Resource_ID) → NULL

Anomalies: Insertion Anomaly: Cannot link a resource to a team without both IDs.

Update Anomaly: Changes in resource or team details require updates.

22. Response Victims

- **Dependencies:** PKD, FD: (Victim_ID, Response_ID) → NULL
- **Anomalies:**
 - Insertion Anomaly: Cannot link a victim to a response without both IDs.
 - Update Anomaly: Changes in victim or response details require careful updates.

23. Victim_Contacts

Dependencies: PKD, FD : (Victim_ID, contact) -> NULL

- **Anomalies:**
 - Insertion Anomaly: Cannot link a victim to contact if it is not a NOT NULL attribute
 - Update Anomaly: Changes in victim or contact details require careful updates.

24. Volunteer_Contacts

Dependencies: PKD, FD: (Volunteer_ID, Contact)-> NULL

Anomalies: Insertion Anomaly: Cannot link a volunteer to contact if it is not a NOT NULL attribute

Update Anomaly: Changes in volunteer or contact details require careful updates.

25. Volunteer_Skills

Dependencies: PKD, FD : (Volunteer_ID, Skillset)
->NULL

Anomalies:

- Insertion Anomaly: Cannot link a volunteer to skillset if it is not a NOT NULL attribute
- Update Anomaly: Changes in volunteers or skillset require careful updates.

26. Agency_Contacts

Dependencies: PKD, FD : (Agency_ID, Contact)-> NULL

Anomalies

- Insertion Anomaly: Cannot link an agency to contact if it is not a NOT NULL attribute
- Update Anomaly: Changes in agency or contact details require careful updates.

27. NGO_Contacts

Dependencies : PKD, FD : (NGO_ID, contact)-> NULL

Anomalies: Insertion Anomaly: Cannot link
a NGO to contact if it is not a NOT NULL
attribute

3. NORMALIZATION PROCESS

3.1 FIRST NORMAL FORM (1NF)

1. Disaster

- No multivalued or composite attributes exist in this table.
- **1NF Disaster Relation:**

Disaster(Disaster_ID, Date, Location, Type, Severity)

2. Communities

- No multivalued or composite attributes exist here.
- **1NF Communities Relation:**

Communities(Pincode, Location, Population)

3. Volunteers

- **Multivalued Attributes:** Skillset and Contact_no.
- **1NF Volunteers Relation:**

Volunteers(Volunteer_ID, Volunteer_name, Age, Location, Availability, NGO_ID, Agency_ID)

- New 1NF Relations for Skillset and Contact Info:

Volunteer_Skills(Volunteer_ID, Skill)

Volunteer_Contacts(Volunteer_ID, Contact_no)

4. Agencies

- **Multivalued Attribute:** Contact_info may contain multiple contact details.

- **1NF Agencies Relation:**

Agencies(Agency_ID, Name, Type, Role)

- New 1NF Relation for Contact Info:

Agency_Contacts(Agency_ID, Contact_Info)

5. Resources

- No multivalued or composite attributes exist here.

- **1NF Resources Relation:**

Resources(Resource_ID, Types, Availability, Location, Source)

6. SocialMediaPlatform

- No multivalued or composite attributes exist here.

- **1NF SocialMediaPlatform Relation:**

SocialMediaPlatform(Name, Type)

7. Response

- No multivalued or composite attributes exist here.

- **1NF Response Relation:**

Response(Response_ID, Disaster_ID, EmergencyTeam_ID, Start_date, End_date, Status)

8. EmergencyTeams

- No multivalued or composite attributes exist here.
- **1NF EmergencyTeams Relation:**

EmergencyTeams(Team_ID, Agency_ID, NGO_ID, No_of_Members, Availability)

9. NGOs

- **Multivalued Attribute:** Contact_info
- **1NF NGOs Relation:**

NGOs(NGO_ID, NGO_name, Service)

- New 1NF Relation for Contact Info:

NGO_Contacts(NGO_ID, Contact_Info)

10. Victims

- **Multivalued Attribute:** Contact_no.
- **1NF Victims Relation:**

Victims(Victim_ID, Victim_name, Location, Health_status, Rescue_status, Community_Pincode, Community_Location)

- New 1NF Relation for Contact Info:

Victim_Contacts(Victim_ID, Contact_no)

11-23. Remaining Relations (Associative Entities)

- These tables (**Disaster_Communities**, **Disaster_Volunteers**, **Disaster_Agencies**, **Disaster_NGO**, **Disaster_Victims**, **Disaster_SocialMedia**, **Communities_Volunteer**, **Communities_Agency**, **Communities_NGO**, **Volunteer_Victim**, **Agency_NGO**, **EmergencyTeams_Resources**, and **Response_Victims**) are associative entities with only foreign keys or atomic attributes, meaning they do not contain any multivalued or composite attributes.
- Each relation already satisfies 1NF.

3.2 SECOND NORMAL FORM (2NF)

A table is in 2NF if it meets the following criteria:

1. It is already in First Normal Form (1NF): This means that the table has no repeating groups or arrays. Every column contains atomic (indivisible) values, and each record is unique.
2. It has no partial dependency: Partial dependency occurs when a non-prime attribute (an attribute that is not part of a candidate key) depends on part of a composite primary key. In simpler terms, in a table with a composite key (a primary key made up of more than one column), every non-prime attribute must depend on the whole composite key, not just a part of it.

1. Disaster

Primary Key: Disaster_ID

2NF Compliance: Already in 2NF because there is a single primary key and no partial dependencies.

2. Communities

Primary Key: (Pincode, Location)

2NF Compliance: Already in 2NF, as all attributes are fully dependent on the composite primary key.

3. Volunteers

Primary Key: Volunteer_ID

2NF Compliance: Already in 2NF. Since we created separate tables for multivalued attributes Contact_no and Skillset in 1NF, there are no partial dependencies.

4. Volunteer_Contacts

- Primary Key: (Volunteer_ID, Contact_no)
- 2NF Compliance: Already in 2NF, as there are no partial dependencies.

5. Volunteer_Skills

- Primary Key: (Volunteer_ID, Skillset)
- 2NF Compliance: Already in 2NF, with no partial dependencies.

6. Agencies

- Primary Key: Agency_ID
- 2NF Compliance: Already in 2NF, as each attribute is fully dependent on the single primary key.

7. Agency_Contacts

- Primary Key: (Agency_ID, Contact_info)
- 2NF Compliance: Already in 2NF, with no partial dependencies.

8. Resources

- Primary Key: Resource_ID
- 2NF Compliance: Already in 2NF, as there are no partial dependencies.

9. Social Media Platform

- Primary Key: Name
- 2NF Compliance: Already in 2NF, as there are no partial dependencies.

10. Response

- Primary Key: Response_ID
- 2NF Compliance: Already in 2NF, as each attribute is fully dependent on the primary key.

11. Emergency Teams

- Primary Key: Team_ID
- 2NF Compliance: Already in 2NF.

12. NGOs

- Primary Key: NGO_ID
- 2NF Compliance: Already in 2NF.

13. NGO_Contacts

- Primary Key: (NGO_ID, Contact_info)
- 2NF Compliance: Already in 2NF, as there are no partial dependencies.

14. Victims

- Primary Key: Victim_ID
- 2NF Compliance: Already in 2NF, with no partial dependencies.

15. Victim_Contacts

- Primary Key: (Victim_ID, Contact_no) • 2NF Compliance: Already in 2NF.

16. Disaster_Communities

- Primary Key: (Disaster_ID, Community_Pincode,

Community_Location)

- 2NF Compliance: Already in 2NF.

17. Disaster_Volunteers

- Primary Key: (Disaster_ID, Volunteer_ID) • 2NF Compliance: Already in 2NF.

18. Disaster_Agencies

- Primary Key: (Disaster_ID, Agency_ID) • 2NF Compliance: Already in 2NF.

19. Disaster_NGOs

- Primary Key: (Disaster_ID, NGO_ID) • 2NF Compliance: Already in 2NF.

20. Disaster_Victims

- Primary Key: (Disaster_ID, Victim_ID) • 2NF Compliance: Already in 2NF.

21. Disaster_SocialMedia

- Primary Key: (Disaster_ID, PlatformName) • 2NF Compliance: Already in 2NF.

22. Communities_Volunteers

- Primary Key: (Community_Pincode, Community_Location, Volunteer_ID)
- 2NF Compliance: Already in 2NF.

23. Community_Agencies

- Primary Key: (Community_Pincode, Agency_ID) Community_Location,
- 2NF Compliance: Already in 2NF.

24. Community_NGOs

- Primary Key: (Community_Pincode, NGO_ID) Community_Location,
- 2NF Compliance: Already in 2NF.

25. Volunteer_Victim

- Primary Key: (Volunteer_ID, Victim_ID)
- 2NF Compliance: Already in 2NF.

26. Agencies_NGOs

- Primary Key: (Agency_ID, NGO_ID)
- 2NF Compliance: Already in 2NF.

27. EmergencyTeams_Resources

2. Primary Key: (Team_ID, Resource_ID)
3. 2NF Compliance: Already in 2NF.

28. Response_Victims

- Primary Key: (Victim_ID, Response_ID)
- 2NF Compliance: Already in 2NF.

3.3 Third Normal Form (3NF)

A relation is in **3NF** if:

1. It is in 2NF.
2. It has no **transitive dependencies** (no non-key attribute is transitively dependent on the primary key).

Since each relation is already in 2NF and no transitive dependencies exist, all relations are in 3NF.

3.4 BCNF

A relation is in **BCNF** if:

1. It is in 3NF.
2. For every functional dependency ($X \rightarrow Y$), X must be a superkey.

All relations adhere to BCNF, as all functional dependencies involve superkeys.

FINAL SCHEMA

- 1 **Disaster** (Disaster_ID, Date, Location, Type, Severity)
- 2 **Agency** (Agency_ID, Name, Type, Contact, Location, Resources)
- 3 **Communities** (Pincode, Location, Population, Resources, Vulnerability_Level)
- 4 **Volunteers** (Volunteer_ID, Name, Age, Skill, Contact)
- 5 **Victims** (Victim_ID, Name, Age, Gender, Medical_Condition)
- 6 **SocialMediaPlatform** (Name, Type)
- 7 **NGOs** (NGO_ID, Name, Location)
- 8 **EmergencyTeams** (Team_ID, Name,)
- 9 **Resources** (Resource_ID, Name, Type, Quantity)
- 10 **Response** (Response_ID, Start_date, End_date, Status, Disaster_ID, EmergencyTeam_ID)
- 11 **Agencies_Contacts**(agency_ID, Contact_no)
- 12 **Volunteer_contacts** (Volunteer_ID, Contact_no)
- 13 **Volunteer_skills** (Volunteer_ID, Skill)
- 14 **Disaster_Communities** (Disaster_ID, Community_Pincode, Community_Location)
- 15 **Disaster_Volunteers** (Disaster_ID, Volunteer_ID)
- 16 **Disaster_Agencies** (Disaster_ID, Agency_ID)
- 17 **Disaster_NGO** (Disaster_ID, NGO_ID)
- 18 **Disaster_Victims** (Disaster_ID, Victim_ID)

- 19 Disaster_SocialMedia** (Disaster_ID, PlatformName)
- 20 Communities_Volunteers** (Community_Pin_Code,
Community_Location, Volunteer_ID)
- 21 Community_Agency** (Community_Pin_Code, Community_Location,
Agency_ID)
- 22 Community_NGO** (Community_Pin_Code, Community_Location,
NGO_ID)
- 23 Volunteer_Victim** (Volunteer_ID, Victim_ID)
- 24 Agencies_NGOs** (Agency_ID, NGO_ID)
- 25 NGO_Contacts** (NGO_ID, Contact_no)
- 26 Response_Victims** (Victim_ID, Response_ID)
Victims_Contacts (Victim_ID, Contact_no)
- 27 EmergencyTeams_Response** (Team_ID, Response_ID)

IMPLEMENTATION OF DATABASE

1.REVISED DDL SCRIPTS

```
CREATE DATABASE crowdsourcedresponse;
```

1.Disaster

```
CREATE TABLE Disaster (
    Disaster_ID INT PRIMARY KEY,
    Date DATE NOT NULL,
    Location VARCHAR(50) NOT NULL,
    Type VARCHAR(50) NOT NULL,
    severity INT CHECK (severity BETWEEN 1 AND 5));
```

2.Communities

```
CREATE TABLE Communities(
    Pincode INT ,
    Location VARCHAR (50),
    Population INT NOT NULL,
    PRIMARY KEY(Pincode, Location));
```

3.Volunteers

```
CREATE TABLE Volunteers(
    Volunteer_ID INT PRIMARY KEY,
```

```
Volunteer_name VARCHAR(50) NOT NULL,  
Age INT NOT NULL, CHECK(AGE>0),  
Location VARCHAR(50) NOT NULL,  
Availability BOOLEAN DEFAULT TRUE,  
NGO_ID INT,  
Agency_ID INT,  
FOREIGN KEY (NGO_ID) REFERENCES NGOs(NGO_ID),  
FOREIGN KEY (Agency_ID) REFERENCES Agency(Agency_ID));
```

4. Volunteer_contacts

```
CREATE TABLE Volunteer_contacts(  
Volunteer_ID INT,  
Contact_no BIGINT NOT NULL,  
PRIMARY KEY(Contact_no, Volunteer_ID),  
FOREIGN KEY (Volunteer_ID) REFERENCES Volunteers (Volunteer_ID));
```

5. Volunteer_skills

```
CREATE TABLE Volunteer_skills(  
Volunteer_ID INT,  
skillset TEXT NOT NULL,  
PRIMARY KEY(skillset, Volunteer_ID),  
FOREIGN KEY (Volunteer_ID) REFERENCES Volunteers (Volunteer_ID));
```

6. Agency

```
CREATE TABLE Agency(  
    Agency_ID INT PRIMARY KEY,  
    Name VARCHAR ,  
    Type VARCHAR,  
    Role TEXT NOT NULL,  
);
```

7. Agency_Contacts

```
CREATE TABLE Agency_Contacts (  
    Agency_ID INT,  
    Contact_info VARCHAR,  
    PRIMARY KEY (Agency_ID, Contact_type, Contact_info),  
    FOREIGN KEY (Agency_ID) REFERENCES Agency(Agency_ID)  
);
```

8. Resources

```
CREATE TABLE Resources(  
    Resource_ID INT PRIMARY KEY,  
    Type VARCHAR,  
    Availability BOOLEAN DEFAULT TRUE,  
    Location VARCHAR,
```

Source VARCHAR);

9. SocialMediaPlatform

```
CREATE TABLE SocialMediaPlatform(  
    Name VARCHAR PRIMARY KEY,  
    Type VARCHAR);
```

10. Response

```
CREATE TABLE Response(  
    Response_ID INT PRIMARY KEY,  
    Start_date DATE NOT NULL,  
    End_date DATE NOT NULL,  
    Status VARCHAR ,  
    Disaster_ID INT,  
    EmergencyTeam_ID INT,  
    FOREIGN KEY (Disaster_ID) REFERENCES Disaster(Disaster_ID),  
    FOREIGN KEY (EmergencyTeam_ID) REFERENCES  
    EmergencyTeams(Team_ID));
```

11. EmergencyTeams

```
CREATE TABLE EmergencyTeams(  
    Team_ID INT PRIMARY KEY,
```

```
No_of_Members INT NOT NULL CHECK (No_of_Members > 0),  
Availability BOOLEAN DEFAULT TRUE,  
NGO_ID INT,  
Agency_ID INT,  
FOREIGN KEY (NGO_ID) REFERENCES NGOs(NGO_ID),  
FOREIGN KEY (Agency_ID) REFERENCES Agency(Agency_ID));
```

12. NGOs

```
CREATE TABLE NGOs(  
    NGO_ID INT PRIMARY KEY,  
    NGO_name VARCHAR NOT NULL,  
    Service VARCHAR);
```

13. NGO_Contacts

```
CREATE TABLE NGO_Contacts (  
    NGO_ID INT,  
    Contact_info VARCHAR,  
    PRIMARY KEY (NGO_ID, Contact_info),  
    FOREIGN KEY (NGO_ID) REFERENCES NGOs(NGO_ID)  
);
```

14. victims

```
CREATE TABLE victims(  
    Victim_ID INT PRIMARY KEY,  
    Victim_name VARCHAR,  
    Location VARCHAR NOT NULL,  
    Health_status VARCHAR NOT NULL,  
    Rescue_status VARCHAR NOT NULL,  
    Community_pincode INT ,  
    Community_Location VARCHAR(50),  
    FOREIGN KEY (Community_pincode, Community_Location) REFERENCES  
    Communities(Pincode, Location));
```

15. Volunteer_Communities

```
CREATE TABLE Volunteer_Communities  
( Volunteer_ID INT, Community_pincode INT,  
Community_Location VARCHAR(50),  
PRIMARY KEY (Volunteer_ID, Community_pincode,  
Community_Location),  
FOREIGN KEY (Volunteer_ID) REFERENCES Volunteers(Volunteer_ID),  
FOREIGN KEY (Community_pincode, Community_Location) REFERENCES  
Communities(Pincode, Location) );
```

16. Community_agencies

```
Community_Pin_Code INT,
```

```
Community_Location VARCHAR,  
Agency_ID INT REFERENCES Agency(Agency_ID),  
FOREIGN KEY (Community_Pin_Code, Community_Location)  
REFERENCES Communities(Pincode, Location),  
PRIMARY KEY (Community_Pin_Code, Community_Location,  
Agency_ID)  
);
```

17. Disaster_Victims

```
CREATE TABLE Disaster_Victims (  
Disaster_ID INT,  
Victim_ID INT,  
PRIMARY KEY (Disaster_ID, Victim_ID),  
FOREIGN KEY (Disaster_ID) REFERENCES Disaster(Disaster_ID),  
FOREIGN KEY (Victim_ID) REFERENCES Victims(Victim_ID)  
);
```

18. Disaster_Volunteers

```
CREATE TABLE Disaster_Volunteers (  
Disaster_ID INT,  
Volunteer_ID INT,  
PRIMARY KEY (Disaster_ID, Volunteer_ID),  
FOREIGN KEY (Disaster_ID) REFERENCES Disaster(Disaster_ID),
```

```
FOREIGN KEY (Volunteer_ID) REFERENCES Volunteers(Volunteer_ID)
);
```

19. Disaster_Agencies

```
CREATE TABLE Disaster_Agencies (
    Disaster_ID INT,
    Agency_ID INT,
    PRIMARY KEY (Disaster_ID, Agency_ID),
    FOREIGN KEY (Disaster_ID) REFERENCES Disaster(Disaster_ID),
    FOREIGN KEY (Agency_ID) REFERENCES Agency(Agency_ID)
);
```

20. Disaster_NGOs

```
CREATE TABLE Disaster_NGOs (
    Disaster_ID INT,
    NGO_ID INT,
    PRIMARY KEY (Disaster_ID, NGO_ID),
    FOREIGN KEY (Disaster_ID) REFERENCES Disaster(Disaster_ID),
    FOREIGN KEY (NGO_ID) REFERENCES NGOs(NGO_ID)
);
```

21. Disaster_Communities

```
CREATE TABLE Disaster_Communities (
    Disaster_ID INT REFERENCES Disaster(Disaster_Id),
    Community_Pincode INT,
    Community_Location VARCHAR,
    FOREIGN KEY (Community_Pincode, Community_Location) REFERENCES
    Communities(Pincode, Location),
    PRIMARY KEY (Disaster_ID, Community_Pincode, Community_Location)
);
```

22. Victim_Contact

```
CREATE TABLE Victim_EmergencyContact (
    Victim_ID INT,
    Emergency_Contact_no BIGINT,
    PRIMARY KEY (Victim_ID, Emergency_Contact_no),
    FOREIGN KEY (Victim_ID) REFERENCES Victims(Victim_ID)
);
```

23. Disaster_SocialMedia

```
CREATE TABLE Disaster_SocialMedia (
    Disaster_ID INT REFERENCES Disaster(Disaster_Id),
```

```
PlatformName VARCHAR REFERENCES SocialMediaPlatform(Name),
PRIMARY KEY (Disaster_ID, PlatformName)
);
```

24. Community_NGO

```
CREATE TABLE Community_NGO (
Community_Pin_Code INT,
Community_Location VARCHAR,
NGO_ID INT REFERENCES NGOs(NGO_ID),
FOREIGN KEY (Community_Pin_Code, Community_Location)
REFERENCES Communities(Pincode, Location),
PRIMARY KEY (Community_Pin_Code, Community_Location, NGO_ID)
);
```

25. Volunteers_Victims

```
CREATE TABLE Volunteer_Victim (
Volunteer_ID INT REFERENCES Volunteers(Volunteer_ID),
Victim_ID INT REFERENCES Victims(Victim_ID),
PRIMARY KEY (Volunteer_ID, Victim_ID)
);
```

26. Agencies_NGOs

```
CREATE TABLE Agencies_NGOs (
    Agency_ID INT REFERENCES Agency(Agency_ID),
    NGO_ID INT REFERENCES NGOs(NGO_ID),
    PRIMARY KEY (Agency_ID, NGO_ID)
);
```

27. Team_resources

```
CREATE TABLE EmergencyTeams_Resources (
    Team_ID INT REFERENCES EmergencyTeams(Team_ID),
    Resource_ID INT REFERENCES Resources(Resource_ID),
    PRIMARY KEY (Team_ID, Resource_ID)
);
```

28. Response_Victims

```
CREATE TABLE Response_Victims (
    Victim_ID INT REFERENCES Victims(Victim_ID),
    Response_ID INT REFERENCES Response(Response_ID),
    PRIMARY KEY (Victim_ID, Response_ID) );
```

2.DATABASE POPULATION

INSERT STATEMENTS

1. Disaster

```
INSERT INTO Disaster (Disaster_ID, Date, Location, Type, Severity)
```

```
VALUES
```

```
(1, '2003-06-15', 'Mumbai', 'Flood', 4),  
(2, '2004-07-25', 'Guwahati', 'Earthquake', 5),  
(3, '2005-08-10', 'Delhi', 'Tornado', 3),  
(4, '2006-04-15', 'Jaipur', 'Heatwave', 2),  
(5, '2007-05-05', 'Chennai', 'Cyclone', 4),  
(6, '2008-06-12', 'Hyderabad', 'Flood', 3),  
(7, '2009-07-01', 'Varanasi', 'Flood', 5),  
(8, '2010-08-20', 'Srinagar', 'Landslide', 2),  
(9, '2011-09-15', 'Ludhiana', 'Flood', 4),  
(10, '2012-10-10', 'Patna', 'Flood', 3),  
(11, '2013-11-30', 'Pune', 'Tornado', 2),  
(12, '2014-12-15', 'Ahmedabad', 'Earthquake', 5),  
(13, '2015-01-05', 'Kolkata', 'Cyclone', 4),  
(14, '2016-02-10', 'Bhubaneswar', 'Flood', 3),  
(15, '2017-03-01', 'Indore', 'Heatwave', 2),  
(16, '2018-04-15', 'Gurgaon', 'Earthquake', 4),
```

- (17, '2019-05-05', 'Nashik', 'Flood', 5),
- (18, '2020-06-01', 'Surat', 'Flood', 1),
- (19, '2021-07-10', 'Lucknow', 'Cyclone', 3),
- (20, '2010-08-05', 'Bhopal', 'Tornado', 4),
- (21, '2005-09-15', 'Coimbatore', 'Landslide', 2),
- (22, '2006-10-01', 'Mysore', 'Flood', 3),
- (23, '2007-11-15', 'Bengaluru', 'Earthquake', 5),
- (24, '2008-12-10', 'Visakhapatnam', 'Cyclone', 4),
- (25, '2009-01-20', 'Kanpur', 'Flood', 2),
- (26, '2010-02-15', 'Ranchi', 'Heatwave', 3),
- (27, '2011-03-10', 'Nagpur', 'Tornado', 4),
- (28, '2012-04-05', 'Raipur', 'Earthquake', 5),
- (29, '2013-05-20', 'Jodhpur', 'Flood', 1),
- (30, '2014-06-10', 'Agra', 'Heatwave', 2),
- (31, '2015-07-25', 'Chandigarh', 'Cyclone', 4),
- (32, '2016-08-01', 'Vadodara', 'Earthquake', 5),
- (33, '2017-09-15', 'Indore', 'Flood', 3),
- (34, '2018-10-20', 'Dehradun', 'Heatwave', 2),
- (35, '2019-11-25', 'Shimla', 'Landslide', 4),
- (36, '2020-12-05', 'Guwahati', 'Flood', 5),
- (37, '2021-01-15', 'Gwalior', 'Tornado', 2),
- (38, '2013-02-10', 'Kochi', 'Cyclone', 4),

- (39, '2014-03-15', 'Puducherry', 'Flood', 3),
- (40, '2015-04-10', 'Patiala', 'Earthquake', 5),
- (41, '2016-05-20', 'Dhanbad', 'Flood', 1),
- (42, '2017-06-15', 'Tirupati', 'Heatwave', 2),
- (43, '2018-07-10', 'Trivandrum', 'Landslide', 4),
- (44, '2019-08-25', 'Nagapattinam', 'Cyclone', 5),
- (45, '2020-09-10', 'Mysore', 'Flood', 3),
- (46, '2021-10-05', 'Ahmedabad', 'Heatwave', 2),
- (47, '2014-11-01', 'Ludhiana', 'Tornado', 4),
- (48, '2015-12-15', 'Kolkata', 'Flood', 2),
- (49, '2013-01-25', 'Bhuj', 'Flood', 4),
- (50, '2009-02-10', 'Agartala', 'Tornado', 2),
- (51, '2006-03-05', 'Raipur', 'Earthquake', 5),
- (52, '2004-04-15', 'Guwahati', 'Flood', 3),
- (53, '2007-05-10', 'Dibrugarh', 'Cyclone', 4),
- (54, '2008-06-20', 'Nashik', 'Heatwave', 2),
- (55, '2009-07-30', 'Vadodara', 'Earthquake', 5),
- (56, '2010-08-15', 'Surat', 'Flood', 1),
- (57, '2011-09-01', 'Patna', 'Tornado', 3),
- (58, '2012-10-10', 'Lucknow', 'Flood', 4),
- (59, '2013-11-20', 'Bhopal', 'Earthquake', 5),
- (60, '2014-12-30', 'Chennai', 'Cyclone', 2),

- (61, '2015-01-25', 'Kochi', 'Flood', 4),
(62, '2016-02-10', 'Guwahati', 'Heatwave', 3),
(63, '2017-03-15', 'Gurgaon', 'Tornado', 4),
(64, '2018-04-20', 'Nagpur', 'Earthquake', 5),
(65, '2019-05-10', 'Gwalior', 'Flood', 1),
(66, '2020-06-25', 'Mangaluru', 'Cyclone', 4),
(67, '2021-07-15', 'Bengaluru', 'Landslide', 5),
(68, '2019-08-30', 'Jodhpur', 'Heatwave', 2),
(69, '2020-09-10', 'Kanpur', 'Flood', 3),
(70, '2021-10-05', 'Kolkata', 'Earthquake', 5),
(71, '2005-06-10', 'Bhuj', 'Cyclone', 4),
(72, '2006-07-15', 'Mysore', 'Flood', 2),
(73, '2007-08-20', 'Agartala', 'Earthquake', 5),
(74, '2008-09-25', 'Srinagar', 'Landslide', 3),
(75, '2009-10-15', 'Delhi', 'Flood', 4),
(76, '2010-11-20', 'Bhopal', 'Tornado', 2),
(77, '2011-12-10', 'Indore', 'Heatwave', 3),
(78, '2012-01-20', 'Lucknow', 'Flood', 5),
(79, '2013-02-15', 'Nagpur', 'Earthquake', 4),
(80, '2014-03-05', 'Surat', 'Cyclone', 5),
(81, '2015-04-25', 'Puducherry', 'Flood', 1),
(82, '2016-05-15', 'Nashik', 'Tornado', 3),

```
(83, '2017-06-01', 'Patiala', 'Flood', 4),
(84, '2018-07-10', 'Visakhapatnam', 'Earthquake', 5),
(85, '2019-08-15', 'Gurgaon', 'Cyclone', 2);
```



The screenshot shows a database interface with a toolbar at the top containing icons for new, open, save, delete, and search, followed by a SQL button. Below the toolbar is a table with the following columns: disaster_id [PK] integer, date date, location character varying (50), type character varying (50), and severity integer. The table contains 85 rows of data. The last row is highlighted in yellow. At the bottom left, it says 'Total rows: 85 of 85'. At the bottom right, it says 'Ln 1, Col 22'.

	disaster_id [PK] integer	date date	location character varying (50)	type character varying (50)	severity integer
1	1	2003-06-15	Mumbai	Flood	4
2	2	2004-07-25	Guwahati	Earthquake	5
3	3	2005-08-10	Delhi	Tornado	3
4	4	2006-04-15	Jaipur	Heatwave	2
5	5	2007-05-05	Chennai	Cyclone	4
6	6	2008-06-12	Hyderabad	Flood	3
7	7	2009-07-01	Varanasi	Flood	5
8	8	2010-08-20	Srinagar	Landslide	2
9	9	2011-09-15	Ludhiana	Flood	4

Tuple count – 85

2. Communities

```
INSERT INTO Communities (Pincode, Location, Population) VALUES
(110001, 'Connaught Place', 50),
(110002, 'Chandni Chowk', 70),
(110003, 'Karol Bagh', 60),
(110004, 'Paharganj', 80),
(110005, 'Lajpat Nagar', 300),
(110006, 'Rohini', 1000),
(110007, 'Dwarka', 1500),
(110008, 'Janakpuri', 750),
(110009, 'Preet Vihar', 450),
(110010, 'Kalkaji', 650),
```

- (400001, 'Marine Lines', 30),
- (400002, 'Dadar', 200),
- (400003, 'Bandra', 700),
- (400004, 'Andheri', 2500),
- (400005, 'Thane', 5000),
- (400006, 'Navi Mumbai', 10000),
- (400007, 'Borivali', 5000),
- (400008, 'Malad', 6000),
- (400009, 'Versova', 2500),
- (400010, 'Vashi', 3000),
- (500001, 'Banjara Hills', 600),
- (500002, 'Jubilee Hills', 800),
- (500003, 'Secunderabad', 1500),
- (500004, 'Gachibowli', 900),
- (500005, 'Hitech City', 1200),
- (500006, 'Madhapur', 700),
- (500007, 'LB Nagar', 500),
- (500008, 'Dilsukhnagar', 400),
- (500009, 'Kukatpally', 300),
- (500010, 'Mehdipatnam', 450),
- (600001, 'T. Nagar', 500),
- (600002, 'Mylapore', 400),

- (600003, 'Nungambakkam', 600),
- (600004, 'Anna Nagar', 700),
- (600005, 'Velachery', 800),
- (600006, 'Kotturpuram', 300),
- (600007, 'Adyar', 700),
- (600008, 'Besant Nagar', 250),
- (600009, 'Kottivakkam', 300),
- (600010, 'Tambaram', 600),
- (700001, 'Salt Lake City', 1200),
- (700002, 'Garia', 800),
- (700003, 'Behala', 900),
- (700004, 'Dum Dum', 700),
- (700005, 'New Town', 1000),
- (700006, 'Howrah', 1500),
- (700007, 'Bidhannagar', 300),
- (700008, 'Kolkata', 4000),
- (700009, 'Baranagar', 600),
- (700010, 'Tollygunge', 500),
- (800001, 'Shahjahanpur', 1200),
- (800002, 'Bareilly', 1500),
- (800003, 'Moradabad', 2000),
- (800004, 'Aligarh', 1000),

(800005, 'Agra', 5000),
(800006, 'Lucknow', 9000),
(800007, 'Kanpur', 7000),
(800008, 'Noida', 6000),
(800009, 'Ghaziabad', 8000),
(800010, 'Meerut', 3000),
(900001, 'Kathmandu', 10000),
(900002, 'Biratnagar', 4000),
(900003, 'Lalitpur', 3000),
(900004, 'Bhaktapur', 2000),
(900005, 'Pokhara', 3000),
(900006, 'Nepalgunj', 1000),
(900007, 'Janakpur', 800),
(900008, 'Birgunj', 1500),
(900009, 'Bhairahawa', 500),
(900010, 'Itahari', 600),
(950001, 'Dhaka', 50000),
(950002, 'Chittagong', 40000),
(950003, 'Khulna', 25000),
(950004, 'Rajshahi', 10000),
(950005, 'Sylhet', 12000),
(950006, 'Barisal', 8000),

(950007, 'Comilla', 9000),
 (950008, 'Narayanganj', 7000),
 (950009, 'Bogra', 6000),
 (950010, 'Tangail', 5000),
 (110011, 'Saket', 450),
 (500011, 'KPHB Colony', 800),
 (600011, 'Perungudi', 600);



The screenshot shows a database interface with a toolbar at the top containing icons for file operations and SQL. Below the toolbar is a table with three columns: pincode, location, and population. The data consists of 83 rows, each representing a tuple. The population values range from 450 to 100000. At the bottom of the table, it says 'Total rows: 83 of 83' and 'Query complete 00:00:00.200'. In the bottom right corner of the interface, there is a small note 'Ln 3, Col 13'.

	pincode [PK] integer	location [PK] character varying (50)	population integer
1	110001	Connaught Place	5000
2	110002	Chandni Chowk	7000
3	110003	Karol Bagh	6000
4	110004	Paharganj	8000
5	110005	Lajpat Nagar	30000
6	110006	Rohini	100000
7	110007	Dwarka	150000
8	110008	Janakpuri	75000
9	110009	Preet Vihar	45000

Tuple count – 83

3. Agency

INSERT INTO Agency (Agency_ID, Name, Type, Role) VALUES
 (1001, 'National Disaster Response Force', 'Government', 'Disaster response and rescue operations'),
 (1002, 'State Disaster Response Force', 'Government', 'State-level disaster response'),
 (1003, 'Indian Meteorological Department', 'Government', 'Weather forecasting and disaster warning'),

- (1004, 'National Remote Sensing Centre', 'Government', 'Satellite data for disaster management'),
- (1005, 'Ministry of Home Affairs', 'Government', 'Coordination of disaster management policies'),
- (1006, 'National Disaster Management Authority', 'Government', 'Policy making for disaster management'),
- (1007, 'Indian Army', 'Military', 'Disaster rescue and relief operations'),
- (1008, 'Indian Navy', 'Military', 'Humanitarian assistance and disaster relief'),
- (1009, 'Indian Air Force', 'Military', 'Airlift and rescue operations'),
- (1010, 'Indian Coast Guard', 'Military', 'Coastal disaster response'),
- (1011, 'World Health Organization', 'International', 'Health support during disasters'),
- (1012, 'International Federation of Red Cross', 'International', 'Global humanitarian response'),
- (1013, 'United Nations Office for the Coordination of Humanitarian Affairs', 'International', 'Coordination of disaster relief efforts'),
- (1014, 'Food and Agriculture Organization', 'International', 'Food security in emergencies'),
- (1015, 'International Organization for Migration', 'International', 'Migration support during crises'),
- (1016, 'World Bank', 'International', 'Financial assistance for disaster recovery'),
- (1017, 'Asian Development Bank', 'International', 'Development assistance for disaster recovery'),

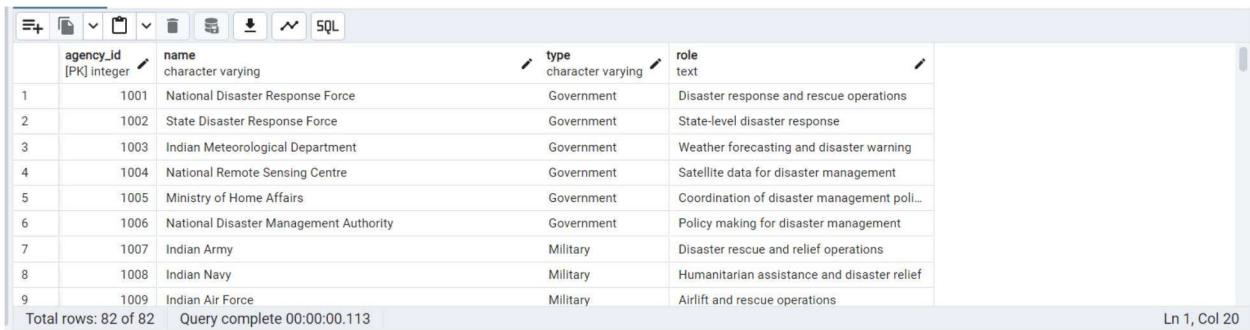
- (1018, 'Red Cross Society', 'International', 'Humanitarian aid'),
- (1019, 'CARE International', 'International', 'Emergency relief and recovery'),
- (1020, 'Oxfam International', 'International', 'Humanitarian assistance'),
- (1021, 'Doctors Without Borders', 'International', 'Emergency medical assistance'),
- (1022, 'Save the Children', 'Government', 'Child protection in emergencies'),
- (1023, 'ActionAid', 'International', 'Social justice and humanitarian aid'),
- (1024, 'Plan International', 'International', 'Child welfare support'),
- (1025, 'World Wildlife Fund', 'International', 'Environmental support in disasters'),
- (1026, 'United Nations Development Programme', 'International', 'Development aid in disaster recovery'),
- (1027, 'UNICEF', 'International', 'Child welfare in emergencies'),
- (1028, 'Global Fund', 'International', 'Health funding during crises'),
- (1029, 'Habitat for Humanity', 'International', 'Housing support in disasters'),
- (1030, 'Mercy Corps', 'International', 'Economic recovery support'),
- (1031, 'Humanitarian Accountability Partnership', 'International', 'Accountability in humanitarian response'),
- (1032, 'European Civil Protection and Humanitarian Aid Operations', 'International', 'Humanitarian aid coordination'),
- (1033, 'International Committee of the Red Cross', 'International', 'Humanitarian support in conflict areas'),

- (1034, 'World Food Programme', 'International', 'Food assistance during emergencies'),
- (1035, 'Child Rights International Network', 'International', 'Child rights advocacy'),
- (1036, 'African Union', 'International', 'Support for African nations during disasters'),
- (1037, 'Asian Disaster Preparedness Center', 'International', 'Disaster preparedness and training'),
- (1038, 'International Network for Small Hydro Power', 'International', 'Support for renewable energy in recovery'),
- (1039, 'Institute for the Study of Human Rights', 'International', 'Research on human rights in crises'),
- (1040, 'National Aeronautics and Space Administration', 'International', 'Satellite data for disaster response'),
- (1041, 'International Institute for Environment and Development', 'International', 'Sustainable development initiatives'),
- (1042, 'Global Green Growth Institute', 'International', 'Support for sustainable recovery'),
- (1043, 'Asian Disaster Risk Reduction Center', 'International', 'Disaster risk reduction strategies'),
- (1044, 'Environmental Defense Fund', 'International', 'Environmental recovery initiatives'),
- (1045, 'Earthquake Engineering Research Institute', 'International', 'Engineering support in recovery'),
- (1046, 'Institute of International Education', 'International', 'Educational support in crises'),

- (1047, 'The International Rescue Committee', 'International', 'Emergency response and recovery'),
- (1048, 'Global Disaster Preparedness Center', 'International', 'Disaster preparedness training'),
- (1049, 'World Vision', 'International', 'Child welfare in crises'),
- (1050, 'International Federation of Red Cross and Red Crescent Societies', 'International', 'Global humanitarian response'),
- (1051, 'International Disaster Emergency Committee', 'International', 'Emergency response coordination'),
- (1052, 'World Resources Institute', 'International', 'Environmental research and recovery'),
- (1053, 'International Fund for Agricultural Development', 'International', 'Support for food security'),
- (1054, 'International Labour Organization', 'International', 'Labor rights in emergencies'),
- (1055, 'Asian Development Fund', 'International', 'Financial assistance for Asian nations'),
- (1056, 'Global Fund for Women', 'International', 'Support for women in disasters'),
- (1057, 'Greenpeace', 'International', 'Environmental advocacy and recovery'),
- (1058, 'Human Rights Watch', 'International', 'Advocacy for human rights in crises'),
- (1059, 'International Organization for Migration', 'International', 'Migration support in crises'),

- (1060, 'Mennonite Central Committee', 'International', 'Disaster response and support'),
- (1061, 'Relief International', 'International', 'Emergency response in disasters'),
- (1062, 'ShelterBox', 'International', 'Emergency shelter support'),
- (1063, 'Catholic Relief Services', 'International', 'Emergency relief efforts'),
- (1064, 'NetHope', 'International', 'Technology support in crises'),
- (1065, 'Friends of the Earth', 'International', 'Environmental advocacy and recovery'),
- (1066, 'Institute for Disaster Management', 'International', 'Research on disaster management'),
- (1067, 'International Council of Voluntary Agencies', 'International', 'Coordination of NGOs in disasters'),
- (1068, 'InterAction', 'International', 'Coalition of NGOs for humanitarian aid'),
- (1069, 'Save the Children', 'International', 'Child welfare and humanitarian aid'),
- (1070, 'Women for Women International', 'International', 'Support for women in conflict zones'),
- (1071, 'Global Network for Disaster Reduction', 'International', 'Disaster risk reduction strategies'),
- (1072, 'Foundation for Environmental Education', 'International', 'Environmental education in crises'),

- (1073, 'ActionAidindia', 'Government', 'Humanitarian aid and social justice'),
- (1074, 'World Federation of United Nations Associations', 'International', 'Advocacy for global cooperation'),
- (1075, 'International Institute for Disaster Risk Reduction', 'International', 'Disaster risk reduction strategies'),
- (1076, 'World Disaster Preparedness Center', 'International', 'Disaster preparedness training centre'),
- (1077, 'International Society for Disaster Risk Reduction', 'International', 'Research on disaster management'),
- (1078, 'International Council for Science', 'International', 'Science for global challenges'),
- (1079, 'Global Disaster Risk Reduction Network', 'International', 'Promoting disaster risk reduction'),
- (1080, 'European Network for Humanitarian Action', 'International', 'Coordination of humanitarian response'),
- (1081, 'International Disaster Committee', 'International', 'Emergency response'),
- (1082, 'Global Health Security Agenda', 'International', 'Strengthening health security');



The screenshot shows a database interface with a toolbar at the top containing icons for file operations and SQL. Below the toolbar is a table with four columns: agency_id, name, type, and role. The data is as follows:

	agency_id [PK] integer	name character varying	type character varying	role text
1	1001	National Disaster Response Force	Government	Disaster response and rescue operations
2	1002	State Disaster Response Force	Government	State-level disaster response
3	1003	Indian Meteorological Department	Government	Weather forecasting and disaster warning
4	1004	National Remote Sensing Centre	Government	Satellite data for disaster management
5	1005	Ministry of Home Affairs	Government	Coordination of disaster management poli...
6	1006	National Disaster Management Authority	Government	Policy making for disaster management
7	1007	Indian Army	Military	Disaster rescue and relief operations
8	1008	Indian Navy	Military	Humanitarian assistance and disaster relief
9	1009	Indian Air Force	Military	Airlift and rescue operations

Total rows: 82 of 82 Query complete 00:00:00.113 Ln 1, Col 20

Tuple count – 82

4. Agency_Contacts

```
INSERT INTO Agency_Contacts (Agency_ID, Contact_info) VALUES  
(1001, '+91-9876543210'),  
(1001, 'ndrf.inquiry@ndma.gov.in'),  
(1002, 'sdrf@state.gov.in'),  
(1003, '+91-1123456789'),  
(1003, 'imd@imd.gov.in'),  
(1004, '+91-9876543210'),  
(1004, 'nrsc@isro.gov.in'),  
(1005, '+91-1987654321'),  
(1005, 'mha@nic.in'),  
(1006, 'ndma@ndma.gov.in'),  
(1007, '+91-3456789012'),  
(1007, 'army@mod.gov.in'),  
(1008, '+91-4567890123'),  
(1008, 'navy@mod.gov.in'),  
(1009, '+91-5678901234'),  
(1009, 'airforce@mod.gov.in'),  
(1010, '+91-6789012345'),  
(1010, 'coastguard@nic.in'),
```

(1011, '+41-1234567890'),
(1011, 'who@who.int'),
(1012, '+41-0987654321'),
(1012, 'ifrc@ifrc.org'),
(1013, '+1-234567890'),
(1013, 'ocha@un.org'),
(1014, '+39-1234567890'),
(1014, 'fao@fao.org'),
(1015, '+41-234567890'),
(1015, 'iom@iom.int'),
(1016, '+1-345678901'),
(1016, 'info@worldbank.org'),
(1017, '+63-123456789'),
(1017, 'adb@adb.org'),
(1018, '+41-876543210'),
(1018, 'redcross@redcross.org'),
(1019, '+1-6543210987'),
(1019, 'care@care.org'),
(1020, '+44-1234567890'),
(1020, 'oxfam@oxfam.org'),
(1021, '+33-1234567890'),
(1021, 'msf@msf.org'),

(1022, '+44-0987654321'),
(1022, 'info@savethechildren.org'),
(1023, '+91-2223456789'),
(1023, 'info@actionaid.org'),
(1024, '+49-123456789'),
(1024, 'info@plan-international.org'),
(1025, '+41-999999999'),
(1025, 'info@wwf.org'),
(1026, '+1-234567890'),
(1026, 'info@undp.org'),
(1027, '+1-345678901'),
(1027, 'info@unicef.org'),
(1028, '+41-876543210'),
(1028, 'info@globalfund.org'),
(1029, '+1-123456789'),
(1029, 'info@habitat.org'),
(1030, '+1-234567890'),
(1030, 'info@mercycorps.org'),
(1031, '+1-345678901'),
(1031, 'info@hapinternational.org'),
(1032, '+32-123456789'),
(1032, 'info@ec.europa.eu'),

(1033, '+41-234567890'),
(1033, 'icrc@icrc.org'),
(1034, '+1-4567890123'),
(1034, 'info@wfp.org'),
(1035, '+41-5678901234'),
(1035, 'info@crin.org'),
(1036, '+27-1234567890'),
(1036, 'info@africa-union.org'),
(1037, 'info@adpc.net'),
(1038, '+41-6543210987'),
(1038, 'info@smallhydropower.org'),
(1039, 'info@ishr.org'),
(1040, '+1-800-1234567'),
(1040, 'info@nasa.gov'),
(1041, '+44-2223334445'),
(1041, 'info@iied.org'),
(1042, '+82-1234567890'),
(1042, 'info@gggi.org'),
(1043, '+86-1234567890'),
(1043, 'info@adrin.org'),
(1044, '+33-2345678901'),
(1044, 'info@edf.org'),

(1045, '+1-7654321987'),
(1045, 'info@eeri.org'),
(1046, '+1-4325678901'),
(1046, 'info@iie.org'),
(1047, '+41-5678901234'),
(1047, 'info@rescue.org'),
(1048, '+44-2223456789'),
(1048, 'info@preparedness.org'),
(1049, '+61-1234567890'),
(1049, 'info@wvi.org'),
(1050, 'info@ifrc.org'),
(1051, '+81-2345678901'),
(1051, 'info@idec.org'),
(1052, '+41-876543210'),
(1052, 'info@wri.org'),
(1053, '+1-234567890'),
(1053, 'info@ifad.org'),
(1054, '+41-2223456789'),
(1054, 'info@ilo.org'),
(1055, 'info@adb.org'),
(1056, '+1-9876543210'),
(1056, 'info@globalfundforwomen.org'),

(1057, 'info@greenpeace.org'),
(1058, '+1-1112223333'),
(1058, 'info@hrw.org'),
(1059, '+41-234567890'),
(1060, '+1-5556667777'),
(1060, 'info@mcc.org'),
(1061, '+1-8765432109'),
(1061, 'info@ri.org'),
(1062, '+44-7890123456'),
(1062, 'info@shelterbox.org'),
(1063, '+1-0987654321'),
(1063, 'info@crs.org'),
(1064, '+1-8881234567'),
(1064, 'info@nethope.org'),
(1065, '+44-1112223334'),
(1065, 'info@foe.org'),
(1066, '+91-1234567890'),
(1066, 'info@idm.org'),
(1067, '+41-5678901234'),
(1067, 'info@icva.org'),
(1068, '+1-9876543210'),
(1068, 'info@interaction.org'),

(1069, '+1-8765432109'),
(1069, 'info@savethechildren.org'),
(1070, '+1-0987654321'),
(1070, 'info@womenforwomen.org'),
(1071, '+44-5556667777'),
(1071, 'info@gndr.org'),
(1072, '+1-2223334445'),
(1072, 'info@fee.org'),
(1073, '+91-234567890'),
(1073, 'info@actionaidindia.org'),
(1074, '+41-2223456789'),
(1074, 'info@wfuna.org'),
(1075, '+44-234567890'),
(1075, 'info@iidrr.org'),
(1076, '+1-5678901234'),
(1076, 'info@wdpc.org'),
(1077, '+44-6543210987'),
(1078, '+1-8765432109'),
(1078, 'info@icsu.org'),
(1079, '+1-1112223334'),
(1079, 'info@gdrr.org'),
(1080, '+32-3456789012'),

```
(1080, 'info@enha.eu'),
(1081, '+41-2345678901'),
(1081, 'info@idc.org'),
(1082, '+1-3456789012'),
(1082, 'info@ghsa.org');
```



	agency_Id [PK] integer	contact_type [PK] character varying	contact_info [PK] character varying
1	1001	Phone	+91-9876543210
2	1001	Email	ndrf.inquiry@ndma.gov.in
3	1002	Email	sdrf@state.gov.in
4	1003	Phone	+91-1123456789
5	1003	Email	imd@imd.gov.in
6	1004	Phone	+91-9876543210
7	1004	Email	nrcs@isro.gov.in
8	1005	Phone	+91-1987654321
9	1005	Email	mha@nlic.in

Total rows: 155 of 155 Query complete 00:00:00.073 Ln 3, Col 29

Tuple count – 155

5. Resources

```
INSERT INTO Resources (Resource_ID, Type, Availability, Location, Source) VALUES
```

```
(501, 'Water Tanker', TRUE, 'Mumbai', 'BMC'),
(502, 'Ambulance', TRUE, 'Delhi', 'AIIMS'),
(503, 'Fire Truck', TRUE, 'Chennai', 'Tamil Nadu Fire Service'),
(504, 'Helicopter', FALSE, 'Bangalore', 'Karnataka Aviation'),
(505, 'Rescue Boat', TRUE, 'Kochi', 'Kerala Coast Guard'),
(506, 'Medicines', TRUE, 'Hyderabad', 'Apollo Pharmacy'),
(507, 'Food Supplies', TRUE, 'Kolkata', 'West Bengal Food Corporation'),
```

- (508, 'Shelter Tents', TRUE, 'Pune', 'NDRF'),
- (509, 'Water Tanker', TRUE, 'Jaipur', 'Rajasthan Municipal Corporation'),
- (510, 'Medical Kits', TRUE, 'Lucknow', 'King George's Medical College'),
- (511, 'Rescue Boat', TRUE, 'Visakhapatnam', 'Andhra Pradesh Coastal Services'),
- (512, 'Fire Truck', TRUE, 'Nagpur', 'Maharashtra Fire Service'),
- (513, 'Helicopter', TRUE, 'Surat', 'Gujarat Aviation'),
- (514, 'Ambulance', TRUE, 'Indore', 'MP Medical Services'),
- (515, 'Medicines', TRUE, 'Patna', 'Bihar Health Department'),
- (516, 'Water Tanker', TRUE, 'Ranchi', 'Jharkhand Municipal Corporation'),
- (517, 'Food Supplies', TRUE, 'Kanpur', 'UP Food Corporation'),
- (518, 'Shelter Tents', TRUE, 'Thiruvananthapuram', 'NDRF'),
- (519, 'Medical Kits', TRUE, 'Vadodara', 'Gujarat Medical Supplies'),
- (520, 'Ambulance', TRUE, 'Ludhiana', 'Punjab Health Services'),
- (521, 'Rescue Boat', FALSE, 'Panaji', 'Goa Coast Guard'),
- (522, 'Fire Truck', TRUE, 'Bhubaneswar', 'Odisha Fire Service'),
- (523, 'Helicopter', TRUE, 'Guwahati', 'Assam Aviation'),
- (524, 'Water Tanker', TRUE, 'Agra', 'Agra Water Works'),
- (525, 'Food Supplies', TRUE, 'Varanasi', 'UP Food Corporation'),
- (526, 'Medicines', TRUE, 'Faridabad', 'Haryana Health Department'),
- (527, 'Shelter Tents', TRUE, 'Amritsar', 'Punjab Disaster Relief'),
- (528, 'Medical Kits', TRUE, 'Meerut', 'Meerut Medical Supplies'),

- (529, 'Ambulance', TRUE, 'Rajkot', 'Gujarat Health Services'),
(530, 'Rescue Boat', TRUE, 'Madurai', 'Tamil Nadu Coastal Services'),
(531, 'Fire Truck', TRUE, 'Nashik', 'Maharashtra Fire Service'),
(532, 'Helicopter', TRUE, 'Jamshedpur', 'Jharkhand Aviation'),
(533, 'Water Tanker', TRUE, 'Allahabad', 'UP Water Works'),
(534, 'Food Supplies', TRUE, 'Aurangabad', 'Maharashtra Food Corporation'),
(535, 'Shelter Tents', TRUE, 'Solapur', 'NDRF'),
(536, 'Medical Kits', TRUE, 'Srinagar', 'Jammu & Kashmir Medical Supplies'),
(537, 'Ambulance', TRUE, 'Tiruppur', 'Tamil Nadu Health Services'),
(538, 'Rescue Boat', TRUE, 'Shimla', 'HP Coastal Services'),
(539, 'Fire Truck', TRUE, 'Raipur', 'Chhattisgarh Fire Service'),
(540, 'Helicopter', TRUE, 'Dhanbad', 'Jharkhand Aviation'),
(541, 'Water Tanker', TRUE, 'Hubli', 'Karnataka Water Services'),
(542, 'Food Supplies', TRUE, 'Gwalior', 'MP Food Corporation'),
(543, 'Medicines', TRUE, 'Bikaner', 'Rajasthan Health Department'),
(544, 'Shelter Tents', TRUE, 'Udaipur', 'NDRF'),
(545, 'Medical Kits', TRUE, 'Jabalpur', 'MP Medical Supplies'),
(546, 'Ambulance', TRUE, 'Belgaum', 'Karnataka Health Services'),
(547, 'Rescue Boat', TRUE, 'Jodhpur', 'Rajasthan Coastal Services'),
(548, 'Fire Truck', TRUE, 'Kozhikode', 'Kerala Fire Service'),
(549, 'Helicopter', TRUE, 'Chandigarh', 'Punjab Aviation'),

(550, 'Water Tanker', TRUE, 'Dehradun', 'Uttarakhand Water Services'),
(551, 'Food Supplies', TRUE, 'Mysore', 'Karnataka Food Corporation'),
(552, 'Medicines', TRUE, 'Noida', 'UP Health Services'),
(553, 'Shelter Tents', TRUE, 'Asansol', 'West Bengal Disaster Relief'),
(554, 'Medical Kits', TRUE, 'Durgapur', 'West Bengal Medical Supplies'),
(555, 'Ambulance', TRUE, 'Nanded', 'Maharashtra Health Services'),
(556, 'Rescue Boat', TRUE, 'Bhavnagar', 'Gujarat Coastal Services'),
(557, 'Fire Truck', TRUE, 'Ajmer', 'Rajasthan Fire Service'),
(558, 'Helicopter', TRUE, 'Thrissur', 'Kerala Aviation'),
(559, 'Water Tanker', TRUE, 'Rourkela', 'Odisha Water Services'),
(560, 'Food Supplies', TRUE, 'Salem', 'Tamil Nadu Food Corporation'),
(561, 'Medicines', TRUE, 'Jhansi', 'UP Health Department'),
(562, 'Shelter Tents', TRUE, 'Kollam', 'Kerala Disaster Relief'),
(563, 'Medical Kits', TRUE, 'Karnal', 'Haryana Medical Supplies'),
(564, 'Ambulance', TRUE, 'Aligarh', 'UP Health Services'),
(565, 'Rescue Boat', TRUE, 'Siliguri', 'West Bengal Coastal Services'),
(566, 'Fire Truck', TRUE, 'Gorakhpur', 'UP Fire Service'),
(567, 'Helicopter', TRUE, 'Moradabad', 'UP Aviation'),
(568, 'Water Tanker', TRUE, 'Guntur', 'Andhra Pradesh Water Services'),
(569, 'Food Supplies', TRUE, 'Bardhaman', 'West Bengal Food Corporation'),
(570, 'Medicines', TRUE, 'Saharanpur', 'UP Health Department'),

(571, 'Shelter Tents', TRUE, 'Tirunelveli', 'Tamil Nadu Disaster Relief'),
(572, 'Medical Kits', TRUE, 'Kakinada', 'Andhra Pradesh Medical Supplies'),
(573, 'Ambulance', TRUE, 'Malegaon', 'Maharashtra Health Services'),
(574, 'Rescue Boat', TRUE, 'Akola', 'Maharashtra Coastal Services'),
(575, 'Fire Truck', TRUE, 'Latur', 'Maharashtra Fire Service'),
(576, 'Helicopter', TRUE, 'Dhule', 'Maharashtra Aviation'),
(577, 'Water Tanker', TRUE, 'Karimnagar', 'Telangana Water Services'),
(578, 'Food Supplies', TRUE, 'Ujjain', 'MP Food Corporation'),
(579, 'Medicines', TRUE, 'Ambala', 'Haryana Health Department'),
(580, 'Shelter Tents', TRUE, 'Bhilai', 'Chhattisgarh Disaster Relief'),
(581, 'Medical Kits', TRUE, 'Bhiwandi', 'Maharashtra Medical Supplies'),
(582, 'Ambulance', TRUE, 'Cuttack', 'Odisha Health Services'),
(583, 'Rescue Boat', TRUE, 'Kolhapur', 'Maharashtra Coastal Services'),
(584, 'Fire Truck', TRUE, 'Sangli', 'Maharashtra Fire Service'),
(585, 'Helicopter', TRUE, 'Bhatinda', 'Punjab Aviation'),
(586, 'Water Tanker', TRUE, 'Gandhinagar', 'Gujarat Water Services'),
(587, 'Food Supplies', TRUE, 'Sikar', 'Rajasthan Food Corporation'),
(588, 'Medicines', TRUE, 'Jalgaon', 'Maharashtra Health Department'),
(589, 'Shelter Tents', TRUE, 'Muzaffarpur', 'Bihar Disaster Relief'),
(590, 'Medical Kits', TRUE, 'Panipat', 'Haryana Medical Supplies');

	resource_id [PK] integer	type character varying	availability boolean	location character varying	source character varying
1	501	Water Tanker	true	Mumbai	BMC
2	502	Ambulance	true	Delhi	AllMS
3	503	Fire Truck	true	Chennai	Tamil Nadu Fire Service
4	504	Helicopter	false	Bangalore	Karnataka Aviation
5	505	Rescue Boat	true	Kochi	Kerala Coast Guard
6	506	Medicines	true	Hyderabad	Apollo Pharmacy
7	507	Food Supplies	true	Kolkata	West Bengal Food Corporation
8	508	Shelter Tents	true	Pune	NDRF
9	509	Water Tanker	true	Jaipur	Rajasthan Municipal Corporation

Total rows: 90 of 90 Query complete 00:00:00.075 Ln 3, Col 24

Tuple count – 90

6. SocialMediaPlatform

```
INSERT INTO SocialMediaPlatform (Name, Type) VALUES
('Facebook', 'Social Networking'),
('Twitter', 'Microblogging'),
('Instagram', 'Photo Sharing'),
('YouTube', 'Video Sharing'),
('WhatsApp', 'Messaging'),
('Snapchat', 'Multimedia Messaging'),
('TikTok', 'Short Video Sharing'),
('Reddit', 'Discussion Forum'),
('Telegram', 'Messaging'),
('LinkedIn', 'Professional Networking'),
('Pinterest', 'Image Sharing'),
('Nextdoor', 'Community Networking'),
('WeChat', 'Messaging'),
('Signal', 'Secure Messaging'),
```

('VK', 'Social Networking'),
('Line', 'Messaging'),
('Quora', 'Q&A Platform'),
('Clubhouse', 'Audio Chat'),
('Tumblr', 'Blogging'),
('Flickr', 'Photo Sharing'),
('Meetup', 'Event Networking'),
('Discord', 'Group Chatting'),
('Koo', 'Microblogging'),
('Viber', 'Messaging'),
('Hike', 'Messaging'),
('Medium', 'Publishing Platform'),
('Mix', 'Content Sharing'),
('Periscope', 'Live Video Streaming'),
('MeWe', 'Social Networking'),
('Parler', 'Microblogging'),
('Gab', 'Microblogging'),
('Blogger', 'Blogging'),
('Substack', 'Publishing Platform'),
('Hive Social', 'Social Networking'),
('Diaspora', 'Social Networking'),
('Mastodon', 'Microblogging'),

('Steemit', 'Content Sharing'),
('Rumble', 'Video Sharing'),
('Locals', 'Community Networking'),
('Vero', 'Photo Sharing'),
('Flipboard', 'Content Aggregation'),
('Zello', 'Walkie-Talkie App'),
('Citizen', 'Community Alerts'),
('Alert Media', 'Emergency Notifications'),
('Nixle', 'Public Safety Alerts'),
('PulsePoint', 'Emergency Notifications'),
('Earthquake Network', 'Earthquake Alerts'),
('ReliefWeb', 'Crisis Updates'),
('DisasterAlert', 'Disaster Monitoring'),
('SAFER Together', 'Preparedness Coordination'),
('AlertHub', 'Real-Time Alerts'),
('ReadyApp', 'Preparedness Information'),
('SkyAlert', 'Earthquake Notifications'),
('AlertNet', 'Humanitarian News'),
('LiveSafe', 'Safety Alerts'),
('MyShake', 'Earthquake Alerts'),
('Safety Check by Facebook', 'Crisis Check-In'),
('Zello Walkie Talkie', 'Real-Time Communication'),

('Eviate', 'Natural Disaster Alerts'),
('Weather Underground', 'Weather Alerts'),
('AccuWeather', 'Weather Updates'),
('The Weather Channel', 'Weather Broadcasting'),
('NOAA Weather App', 'Weather Monitoring'),
('Earthquake Tracker', 'Seismic Updates'),
('WeatherBug', 'Weather Notifications'),
('National Hurricane Center', 'Hurricane Monitoring'),
('Met Office Weather', 'Weather Forecasting'),
('Disaster Info', 'Emergency Updates'),
('American Red Cross App', 'Disaster Preparedness'),
('FEMA App', 'Disaster Information'),
('Alert Ready', 'Public Alerts'),
('SafetyNet', 'Emergency Notifications'),
('FindShelter', 'Shelter Locator'),
('Disaster Ready', 'Preparedness Information'),
('SafeCast', 'Crisis Information'),
('AlertReady Canada', 'Public Alerts'),
('Tsunami Alert', 'Tsunami Notifications'),
('QuakeFeed', 'Earthquake Alerts'),
('Hurricane Tracker', 'Storm Alerts'),
('WeatherNow', 'Weather Forecasts'),

```

('RescueNet', 'Rescue Coordination'),
('AlertMe', 'Emergency Alerts'),
('HelpLink', 'Crisis Assistance'),
('Global Disaster Alert', 'Disaster Information'),
('Shelter Connect', 'Shelter Finder'),
('CrisisConnect', 'Crisis Information'),
('GeoNet', 'Geological Alerts');

```

The screenshot shows a database interface with a toolbar at the top containing icons for file operations and SQL. Below the toolbar is a table with two columns: 'name' and 'type'. The table has 9 rows, each representing a platform and its category. The data is as follows:

	name [PK] character varying	type character varying
1	Facebook	Social Networking
2	Twitter	Microblogging
3	Instagram	Photo Sharing
4	YouTube	Video Sharing
5	WhatsApp	Messaging
6	Snapchat	Multimedia Messaging
7	TikTok	Short Video Sharing
8	Reddit	Discussion Forum
9	Telegram	Messaging

Total rows: 87 of 87 Query complete 00:00:00.120 Ln 1, Col 33

Tuple count – 87

7. NGOs

```

INSERT INTO NGOs (NGO_ID, NGO_name, Service) VALUES
(701, 'SEEDS', 'Disaster Response and Recovery'),
(702, 'Goonj', 'Disaster Relief and Rehabilitation'),
(703, 'HelpAge India', 'Emergency Support for the Elderly'),
(704, 'Oxfam India', 'Disaster Response and Livelihood Support'),
(705, 'CARE India', 'Emergency Response and Recovery'),

```

- (706, 'National Disaster Response Force (NDRF)', 'Disaster Management and Response'),
- (707, 'Indian Red Cross Society', 'Disaster Relief and Emergency Aid'),
- (708, 'The Akshaya Patra Foundation', 'Food Distribution During Disasters'),
- (709, 'SankalpTaru', 'Tree Plantation and Environmental Recovery'),
- (710, 'Save the Children India', 'Child Protection in Emergencies'),
- (711, 'CRY (Child Rights and You)', 'Support for Children in Disasters'),
- (712, 'Smile Foundation', 'Healthcare Support During Disasters'),
- (713, 'Disaster Emergency Committee', 'Emergency Response Coordination'),
- (714, 'Aga Khan Foundation', 'Community Resilience and Disaster Preparedness'),
- (715, 'Wildlife Trust of India', 'Wildlife Conservation and Disaster Recovery'),
- (716, 'Pratham', 'Education for Disaster-Affected Children'),
- (717, 'Shiksha Niketan', 'Education Support in Disaster Relief'),
- (718, 'Tata Institute of Social Sciences (TISS)', 'Research and Training for Disaster Management'),
- (719, 'Udyama', 'Youth Empowerment in Disaster Response'),
- (720, 'Relief India Trust', 'Disaster Relief and Rehabilitation'),
- (721, 'Youth for Seva', 'Volunteering for Disaster Relief'),
- (722, 'Nirmaan Organization', 'Capacity Building for Disaster Management'),

- (723, 'Hope Foundation', 'Disaster Relief and Rehabilitation'),
- (724, 'Amity Foundation', 'Disaster Response and Community Development'),
- (725, 'Sankalp', 'Skill Development for Disaster Response'),
- (726, 'Banyan Impact', 'Mental Health Support During Disasters'),
- (727, 'Rural Development Trust', 'Community Resilience Building'),
- (728, 'The Red Cross Society', 'Emergency Aid and Disaster Response'),
- (729, 'Citizen Rescue Team', 'Community Emergency Response'),
- (730, 'Bhumi', 'Volunteer Mobilization for Disaster Relief'),
- (731, 'Global Relief', 'International Disaster Response Coordination'),
- (732, 'Help Us Help', 'Relief for Disaster Victims'),
- (733, 'The Earthquake Network', 'Earthquake Awareness and Preparedness'),
- (734, 'Sahara Charitable Trust', 'Emergency Support for Vulnerable Communities'),
- (735, 'Indian Social Responsibility Network', 'Community Support in Disasters'),
- (736, 'Emergency Volunteers', 'Volunteer Coordination for Disaster Response'),
- (737, 'Lions Club International', 'Disaster Relief Initiatives'),
- (738, 'Gandhi Fellowship', 'Youth Training for Disaster Management'),
- (739, 'Reliefweb', 'Information Sharing for Disaster Response'),
- (740, 'ChildFund India', 'Support for Children in Disaster-Affected Areas'),

- (741, 'Care for You', 'Health Services During Disasters'),
- (742, 'All India Disaster Management Institute', 'Training and Capacity Building'),
- (743, 'Pahal', 'Women's Empowerment and Disaster Preparedness'),
- (744, 'Udyam', 'Entrepreneurship Development for Disaster Recovery'),
- (745, 'Vatsalya Foundation', 'Child Welfare in Emergencies'),
- (746, 'National Disaster Management Authority (NDMA)', 'Policy Development and Training'),
- (747, 'ActionAid India', 'Emergency Relief and Recovery Programs'),
- (748, 'Earthquake Engineering Research Institute', 'Disaster Risk Reduction'),
- (749, 'Pragati Abhiyan', 'Community Awareness for Disaster Management'),
- (750, 'Bharatiya Jain Sanghatana', 'Youth Development for Disaster Preparedness'),
- (751, 'Sankalp NGO', 'Immediate Response and Rehabilitation'),
- (752, 'Rescue and Relief', 'Search and Rescue Operations'),
- (753, 'Mahatma Gandhi National Rural Employment Guarantee Act (MGNREGA)', 'Disaster Recovery Employment'),
- (754, 'Greenpeace India', 'Environmental Awareness and Disaster Resilience'),
- (755, 'Swayam Shikshan Prayog', 'Women's Empowerment in Disaster Management'),
- (756, 'Dhananjay Foundation', 'Health Support in Disaster Areas'),

- (757, 'Uttar Pradesh Disaster Management Department', 'State-level Disaster Management'),
- (758, 'Vatsalya', 'Child Welfare in Disaster Situations'),
- (759, 'Krishna Foundation', 'Disaster Relief and Community Development'),
- (760, 'Maitri', 'Support for Disaster-Affected Families'),
- (761, 'Community Aid Abroad', 'International Disaster Relief'),
- (762, 'Seva International', 'Disaster Relief and Recovery'),
- (763, 'Himalaya Foundation', 'Environmental Recovery in Disasters'),
- (764, 'Operation Blessing India', 'Humanitarian Relief in Disasters'),
- (765, 'Yuva Unstoppable', 'Youth Initiatives for Disaster Preparedness'),
- (766, 'Aashray', 'Disaster Support for Vulnerable Communities'),
- (767, 'Gandhi Smriti and Darshan Samiti', 'Awareness Programs for Disaster Management'),
- (768, 'Network for Disaster Management', 'Research and Policy Advocacy'),
- (769, 'Association for Disaster Management', 'Training for Local Communities'),
- (770, 'Institute of Disaster Management', 'Research and Capacity Building'),
- (771, 'Earthquake Safety Initiative', 'Awareness Programs for Earthquake Safety'),
- (772, 'Relief Trust', 'Emergency Relief for Disaster Victims'),
- (773, 'Support Foundation', 'Community Rehabilitation and Support'),

- (774, 'Friends of the Earth India', 'Environmental Protection in Disasters'),
- (775, 'Save Life Foundation', 'Road Safety and Disaster Response'),
- (776, 'Tsunami Recovery Program', 'Support for Tsunami Affected Areas'),
- (777, 'National Disaster Relief Fund', 'Funding for Disaster Management'),
- (778, 'World Wildlife Fund (WWF) India', 'Wildlife Protection During Disasters'),
- (779, 'Akanksha Foundation', 'Education Support During Disasters'),
- (780, 'Seva Sadan', 'Community Support in Disasters'),
- (781, 'Sankalp Seva', 'Immediate Disaster Response'),
- (782, 'Volunteer Action Network', 'Volunteer Mobilization for Emergencies'),
- (783, 'Community Rehabilitation Organization', 'Disaster Recovery Support'),
- (784, 'Dr. K. R. Narayanan Foundation', 'Emergency Relief Support'),
- (785, 'Indian Network for Disaster Risk Reduction', 'Disaster Risk Management'),
- (786, 'Suraksha NGO', 'Safety Programs in Disasters'),
- (787, 'Hunger Relief India', 'Food Distribution in Emergencies'),
- (788, 'United Nations Development Programme (UNDP)', 'Disaster Management Initiatives'),
- (789, 'Aarambh', 'Support for Disaster-Affected Individuals'),

(790, 'Sanjeevani', 'Health Services During Emergencies');

	ngo_id [PK] integer	ngo_name character varying	service character varying
1	701	SEEDS	Disaster Response and Recovery
2	702	Goonj	Disaster Relief and Rehabilitation
3	703	HelpAge India	Emergency Support for the Elderly
4	704	Oxfam India	Disaster Response and Livelihood Support
5	705	CARE India	Emergency Response and Recovery
6	706	National Disaster Response Force (NDRF)	Disaster Management and Response
7	707	Indian Red Cross Society	Disaster Relief and Emergency Aid
8	708	The Akshaya Patra Foundation	Food Distribution During Disasters
9	709	SankalpTaru	Tree Plantation and Environmental Recovery

Tuple count – 90

8.NGO_Contacts

```
INSERT INTO NGO_Contacts (NGO_ID, Contact_info) VALUES  
(701, 'info@seedsindia.org'), (701, '+91 9876543210'),  
(702, 'contact@goonj.org'), (702, '+91 9765432101'),  
(703, 'helpline@helpageindia.org'), (703, '+91 9754312678'),  
(704, 'support@oxfamindia.org'), (704, '+91 9845123456'),  
(705, 'info@careindia.org'), (705, '+91 9432112345'),  
(706, 'contact@ndrf.gov.in'), (706, '+91 9212345678'),  
(707, 'info@indianredcross.org'), (707, '+91 9812345678'),  
(708, 'contact@akshayapatra.org'), (708, '+91 9832123456'),  
(709, 'info@sankalptaru.org'), (709, '+91 9876123456'),  
(710, 'support@savechildrenindia.org'), (710, '+91 9812156789'),  
(711, 'info@cry.org'), (711, '+91 9903123456'),  
(712, 'contact@smilefoundation.org'), (712, '+91 9783123456'),
```

(713, 'help@dec.org'), (713, '+91 9845126789'),
(714, 'info@akdn.org'), (714, '+91 9845612345'),
(715, 'support@wtifoundation.org'), (715, '+91 9754612345'),
(716, 'contact@pratham.org'), (716, '+91 9834512345'),
(717, 'info@shikshaniketan.org'), (717, '+91 9865132456'),
(718, 'contact@tiss.edu'), (718, '+91 9654213456'),
(719, 'info@udyama.org'), (719, '+91 9761234567'),
(720, 'contact@reliefindiatrust.org'), (720, '+91 9834123456'),
(721, 'info@youthforseva.org'), (721, '+91 9823412345'),
(722, 'support@nirmaan.org'), (722, '+91 9876543201'),
(723, 'contact@hopefoundation.org'), (723, '+91 9865432109'),
(724, 'info@amityfoundation.org'), (724, '+91 9845672101'),
(725, 'contact@sankalp.org'), (725, '+91 9823765432'),
(726, 'support@banyanimpact.org'), (726, '+91 9867541230'),
(727, 'info@ruraldevelopmenttrust.org'), (727, '+91 9756134023'),
(728, 'contact@redcross.org'), (728, '+91 9834213450'),
(729, 'help@citizenrescueteam.org'), (729, '+91 9876432100'),
(730, 'info@bhumi.org.in'), (730, '+91 9823765434'),
(731, 'support@globalrelief.org'), (731, '+91 9654321780'),
(732, 'contact@helpushelp.org'), (732, '+91 9753216457'),
(733, 'info@earthquakenetwork.org'), (733, '+91 9834261578'),
(734, 'support@saharacharity.org'), (734, '+91 9845123498'),

(735, 'contact@isrn.org'), (735, '+91 9867543289'),
(736, 'info@emergencyvolunteers.org'), (736, '+91 9834512789'),
(737, 'support@lionsclub.org'), (737, '+91 9812354678'),
(738, 'info@gandhifellowship.org'), (738, '+91 9865312457'),
(739, 'contact@reliefweb.org'), (739, '+91 9735124567'),
(740, 'info@childfundindia.org'), (740, '+91 9865432170'),
(741, 'contact@careforyou.org'), (741, '+91 9845612379'),
(742, 'support@aidmi.org'), (742, '+91 9732124567'),
(743, 'info@pahal.org'), (744, 'contact@udyam.org'),
(744, '+91 9732165408'), (745, 'info@vatsalyafoundation.org'),
(745, '+91 9823765410'), (746, 'contact@ndma.gov.in'),
(746, '+91 9867123456'), (747, 'support@actionaidindia.org'),
(747, '+91 9875132460'), (748, 'info@eeri.org'),
(748, '+91 9723612458'), (749, 'contact@pragatiabhiyan.org'),
(749, '+91 9864132578'), (750, 'support@bjsindia.org'),
(750, '+91 9875134260'), (751, '+91 9863124579'),
(752, 'help@rescueandrelief.org'), (752, '+91 9753412678'),
(753, 'info@mgnrega.gov.in'), (753, '+91 9835712460'),
(754, 'contact@greenpeace.org'), (754, '+91 9836214578'),
(755, 'info@sspindia.org'), (755, '+91 9765431280'),
(756, 'contact@dhananjayfoundation.org'), (756, '+91 9865124378'),
(757, 'info@updisastermanagement.org'), (757, '+91 9753124680'),

(758, 'contact@vatsalya.org'), (758, '+91 9836124579'),
(759, 'support@krishnafoundation.org'), (759, '+91 9865124738'),
(760, 'contact@maitri.org'), (760, '+91 9753124789'),
(761, 'info@caa.org'), (761, '+91 9865324178'),
(762, 'support@sevainternational.org'), (762, '+91 9732165480'),
(763, 'info@himalayafoundation.org'), (763, '+91 9823641258'),
(764, 'contact@operationblessing.org'), (764, '+91 9735124680'),
(765, 'info@yuvachange.org'), (765, '+91 9865134279'),
(766, 'support@aashray.org'), (766, '+91 9753624180'),
(767, 'info@gandhismriti.org'), (767, '+91 9835124789'),
(768, 'support@ndmp.org'), (768, '+91 9723124589'),
(769, 'info@associationfordisastermanagement.org'),
(769, '+91 9735124681'),
(770, 'contact@idm.org'), (770, '+91 9834712580'),
(771, 'info@earthquakesafety.org'), (771, '+91 9863124759'),
(772, 'contact@relieftrust.org'), (772, '+91 9764312509'),
(773, 'info@supportfoundation.org'), (773, '+91 9875124398'),
(774, 'contact@foei.org'), (774, '+91 9753126480'),
(775, 'support@savelifefoundation.org'),
(776, 'info@tsunamirecoveryprogram.org'),
(776, '+91 9753124687'), (777, 'support@ndrfindia.org'),
(777, '+91 9823641587'), (778, 'info@wwfindia.org'),

```
(778, '+91 9765312487'), (779, 'contact@akankshafoundation.org'),
(779, '+91 9832174569'), (780, 'info@sevasadan.org'),
(780, '+91 9753612479');
```

	ngo_id	contact_info
1	701	info@seedsindia.org
2	701	+91 9876543210
3	702	contact@goonj.org
4	702	+91 9765432101
5	703	helpline@helpageindia.org
6	703	+91 9754312678
7	704	support@oxfamindia.org
8	704	+91 9845123456
9	705	info@careindia.org

Total rows: 157 Query complete 00:00:00.074 Ln 1, Col 16

Tuple count – 157

9. Volunteers

```
INSERT INTO Volunteers (Volunteer_ID, Volunteer_name, Age, Location,
Availability, NGO_ID, Agency_ID) VALUES
```

```
(10001, 'Aarav Sharma', 24, 'Mumbai', TRUE, 701, 1001),
(10002, 'Vivaan Singh', 29, 'Delhi', TRUE, 702, 1060),
(10003, 'Anaya Gupta', 21, 'Chennai', FALSE, 703, 1003),
(10004, 'Diya Patel', 26, 'Bangalore', TRUE, 704, 1004),
(10005, 'Reyansh Mehta', 30, 'Hyderabad', FALSE, 705, 1065),
(10006, 'Saanvi Kumar', 23, 'Ahmedabad', TRUE, 706, 1006),
(10007, 'Vihaan Joshi', 35, 'Pune', TRUE, 707, 1005),
(10008, 'Aditi Reddy', 20, 'Kolkata', FALSE, 708, 1008),
(10009, 'Aryan Chaudhary', 28, 'Jaipur', TRUE, 709, 1009),
```

(10010, 'Kavya Rao', 22, 'Surat', TRUE, 710, 1010),
(10011, 'Advik Singh', 27, 'Lucknow', FALSE, 711, 1011),
(10012, 'Tara Naik', 32, 'Kanpur', TRUE, 712, 1012),
(10013, 'Sai Iyer', 25, 'Nagpur', TRUE, 713, 1013),
(10014, 'Dev Sharma', 30, 'Visakhapatnam', TRUE, 714, 1014),
(10015, 'Aarohi Patel', 24, 'Indore', FALSE, 715, 1015),
(10016, 'Ayush Gupta', 28, 'Thane', TRUE, 716, 1016),
(10017, 'Naira Kumar', 26, 'Bhopal', TRUE, 717, 1017),
(10018, 'Rudra Singh', 23, 'Patna', FALSE, 718, 1018),
(10019, 'Yashvi Joshi', 27, 'Vadodara', TRUE, 719, 1019),
(10020, 'Mihir Kapoor', 31, 'Ghaziabad', TRUE, 720, 1020),
(10021, 'Riya Choudhury', 22, 'Ludhiana', FALSE, 721, 1021),
(10022, 'Arjun Mehta', 30, 'Agra', TRUE, 722, 1022),
(10023, 'Ishaan Jain', 25, 'Nashik', FALSE, 723, 1023),
(10024, 'Myra Sethi', 24, 'Faridabad', TRUE, 724, 1024),
(10025, 'Kabir Batra', 27, 'Meerut', TRUE, 725, 1025),
(10026, 'Anvi Desai', 32, 'Rajkot', TRUE, 726, 1026),
(10027, 'Rishi Khanna', 28, 'Jabalpur', FALSE, 727, 1027),
(10028, 'Aahana Chauhan', 22, 'Jamshedpur', TRUE, 728, 1028),
(10029, 'Kiaan Verma', 26, 'Guwahati', TRUE, 729, 1029),
(10030, 'Advika Sen', 29, 'Coimbatore', FALSE, 730, 1030),
(10031, 'Dhruv Roy', 25, 'Kochi', TRUE, 731, 1031),

- (10032, 'Zara Aggarwal', 21, 'Dehradun', TRUE, 732, 1032),
(10033, 'Saanvi Prasad', 23, 'Mysore', TRUE, 733, 1033),
(10034, 'Ira Nair', 30, 'Raipur', FALSE, 734, 1034),
(10035, 'Reyansh Goswami', 27, 'Gwalior', TRUE, 735, 1035),
(10036, 'Myra Pathak', 26, 'Ranchi', TRUE, 736, 1036),
(10037, 'Parth Nambiar', 28, 'Chandigarh', TRUE, 737, 1037),
(10038, 'Shanaya Chopra', 24, 'Kota', FALSE, 738, 1038),
(10039, 'Vihaan Shetty', 30, 'Bareilly', TRUE, 739, 1007),
(10040, 'Ayaan Gokhale', 25, 'Allahabad', TRUE, 740, 1040),
(10041, 'Aditi Tripathi', 23, 'Howrah', TRUE, 741, 1041),
(10042, 'Rehan Kulkarni', 21, 'Hubli', FALSE, 742, 1042),
(10043, 'Aria Das', 26, 'Gorakhpur', TRUE, 743, 1043),
(10044, 'Devya Agrawal', 29, 'Bhiwandi', TRUE, 744, 1044),
(10045, 'Arnav Menon', 31, 'Siliguri', FALSE, 745, 1045),
(10046, 'Kavya Rathi', 28, 'Srinagar', TRUE, 746, 1046),
(10047, 'Pranav Gupta', 24, 'Belgaum', TRUE, 747, 1047),
(10048, 'Samaira Ahuja', 27, 'Jodhpur', FALSE, 748, 1048),
(10049, 'Shaurya Banerjee', 32, 'Erode', TRUE, 749, 1049),
(10050, 'Aadhya Bhalla', 23, 'Warangal', TRUE, 750, 1050),
(10051, 'Kian Kaur', 22, 'Amravati', TRUE, 751, 1051),
(10052, 'Kiara Malik', 29, 'Guntur', FALSE, 752, 1052),
(10053, 'Vivaan Mehra', 27, 'Ajmer', TRUE, 753, 1053),

- (10054, 'Anvi Bhattacharya', 25, 'Kolhapur', TRUE, 754, 1054),
(10055, 'Lakshya Sengupta', 30, 'Thiruvananthapuram', TRUE, 755, 1055),
(10056, 'Aarav Saxena', 28, 'Bikaner', FALSE, 756, 1056),
(10057, 'Aahana Bisht', 24, 'Jhansi', TRUE, 757, 1057),
(10058, 'Shaurya Pandey', 22, 'Udaipur', TRUE, 758, 1058),
(10059, 'Tia Kapoor', 23, 'Pondicherry', FALSE, 759, 1059),
(10060, 'Kabir Singhania', 31, 'Aligarh', TRUE, 760, 1077),
(10061, 'Aadhya Kulkarni', 27, 'Moradabad', TRUE, 761, 1061),
(10062, 'Reyansh Bhatia', 25, 'Gaya', TRUE, 762, 1062),
(10063, 'Arnav Chauhan', 23, 'Mathura', FALSE, 763, 1063),
(10064, 'Anaya Nambiar', 26, 'Panaji', TRUE, 764, 1064),
(10065, 'Krishna Shukla', 30, 'Nanded', TRUE, 765, 1079),
(10066, 'Arya Deshmukh', 22, 'Sagar', FALSE, 766, 1066),
(10067, 'Kiara Nair', 29, 'Bhavnagar', TRUE, 767, 1067),
(10068, 'Ayaan Sengupta', 26, 'Hisar', TRUE, 768, 1068),
(10069, 'Arjun Mukherjee', 24, 'Kharagpur', FALSE, 769, 1069),
(10070, 'Anvi Mishra', 27, 'Karnal', TRUE, 770, 1070),
(10071, 'Shaurya Grover', 31, 'Bilaspur', TRUE, 771, 1071),
(10072, 'Ira Seth', 25, 'Haridwar', TRUE, 772, 1072),
(10073, 'Nivaan Dey', 28, 'Tirupati', TRUE, 773, 1073),
(10074, 'Pia Yadav', 24, 'Ambala', FALSE, 774, 1074),

(10075, 'Ravina Khanna', 26, 'Aurangabad', TRUE, 775, 1075),
 (10076, 'Shivansh Mehta', 29, 'Roorkee', TRUE, 776, 1076),
 (10077, 'Tanvi Jain', 22, 'Guwahati', TRUE, 777, 1037),
 (10078, 'Veer Singh', 31, 'Haldwani', FALSE, 778, 1078),
 (10079, 'Siddhi Desai', 25, 'Solapur', TRUE, 779, 1070),
 (10080, 'Kunal Dutta', 30, 'Sambalpur', TRUE, 780, 1080),
 (10081, 'Riya Chawla', 22, 'Nellore', FALSE, 781, 1081),
 (10082, 'Aditya Malhotra', 28, 'Kottayam', TRUE, 782, 1002),
 (10083, 'Meera Iyer', 27, 'Jammu', TRUE, 783, 1082),
 (10084, 'Sana Kapur', 24, 'Alwar', TRUE, 784, 1074),
 (10085, 'Vikram Saini', 29, 'Rudrapur', TRUE, 785, 1065),
 (10086, 'Swara Soni', 22, 'Dharamshala', FALSE, 786, 1016),
 (10087, 'Anvi Chaudhary', 26, 'Jabalpur', TRUE, 787, 1037),
 (10088, 'Aaryan Kumar', 28, 'Bhopal', TRUE, 788, 1048),
 (10089, 'Meghna Singh', 30, 'Jodhpur', FALSE, 789, 1039),
 (10090, 'Harman Khurana', 25, 'Dehradun', TRUE, 790, 1002);



The screenshot shows a database interface with a table titled 'volunteer'. The columns are: volunteer_id [PK] integer, volunteer_name character varying (50), age integer, location character varying (50), availability boolean, ngo_id integer, and agency_id integer. The table contains 90 rows of data. The last row is highlighted in yellow.

	volunteer_id [PK] integer	volunteer_name character varying (50)	age integer	location character varying (50)	availability boolean	ngo_id integer	agency_id integer
1	10001	Aarav Sharma	24	Mumbai	true	701	1001
2	10002	Vivaan Singh	29	Delhi	true	702	1060
3	10003	Anaya Gupta	21	Chennai	false	703	1003
4	10004	Diya Patel	26	Bangalore	true	704	1004
5	10005	Reyansh Mehta	30	Hyderabad	false	705	1065
6	10006	Saanvi Kumar	23	Ahmedabad	true	706	1006
7	10007	Viihaan Joshi	35	Pune	true	707	1005
8	10008	Aditi Reddy	20	Kolkata	false	708	1008
9	10009	Aryan Chaudhary	28	Jaipur	true	709	1009
Total rows: 90 of 90							
Query complete 00:00:00.113							
Ln 1, Col 24							

Tuple count – 90

10. Volunteer contacts

(10001, 9876543210),(10002, 9123456789),
(10002, 9345678901),(10003, 8765432109),
(10004, 9988112233),(10004, 9900221144),
(10005, 9871234567),(10006, 8999000111),
(10006, 8912345678),(10007, 9775566432),
(10007, 9654321789),(10008, 8888776611),
(10008, 8822119988),(10009, 9191919191),
(10009, 9898989898),(10010, 9345678900),
(10010, 9123456780),(10011, 8112233445),
(10011, 8012345678),
(10012, 8765432190),(10012, 8901234567),
(10013, 7654321987),(10013, 7543210986),
(10014, 9034567891),(10015, 9765432109),
(10016, 8988776654),(10016, 8877665544),
(10017, 9657890123),(10017, 9494949494),
(10018, 9234567890),(10018, 9109876543),
(10019, 8309876543),(10020, 7654321098),
(10020, 7598765432),(10021, 9988775566),
(10021, 9776655443),(10022, 9456123456),
(10023, 8776655432),(10023, 8665544332),

(10024, 9901234567),(10024, 9887654321),
(10025, 9324567890),(10026, 9998765432),
(10026, 9777654320),(10027, 9087654320),
(10028, 8998765432),(10028, 8765432101),
(10029, 9234567810),(10029, 9323456781),
(10030, 9112345678),(10030, 9087654321),
(10031, 8765432100),(10032, 9345678902),
(10033, 9678901234),(10033, 9501234567),
(10034, 8991001100),(10034, 8877665544),
(10035, 9356789012),(10036, 9887654321),
(10037, 9654321098),(10037, 9745632180),
(10038, 9501234568),(10039, 9182736450),
(10039, 9234567801),(10040, 9098765432),
(10040, 9012345670),(10041, 8888776655),
(10042, 9109876540),(10042, 9198765432),
(10043, 9934567890),(10043, 9812345679),
(10044, 9654321980),(10044, 8765432190),
(10045, 9176543210),(10046, 9509876543),
(10046, 9654321091),(10047, 9876543211),
(10047, 9988776656),(10048, 9123456782),
(10048, 9345678903),(10049, 8765432110),
(10049, 8999888778),(10050, 9988112234),

(10051, 9871234568),(10051, 9567890124),
(10052, 8999000123),(10053, 9654321782),
(10054, 8888776612),(10054, 8822119989),
(10055, 9191919192),(10055, 9898989899),
(10056, 9345678901),(10056, 9123456781),
(10057, 8112233446),(10057, 8012345679),
(10058, 8765432191),(10058, 8901234568),
(10059, 7654321988),(10059, 7543210987),
(10060, 9034567892),(10060, 9087654322),
(10061, 9871234569),(10061, 9765432108),
(10062, 8988776655),(10062, 8877665545),
(10063, 9657890124),(10063, 9494949495),
(10064, 9234567891),(10064, 9109876544),
(10065, 8812345679),(10065, 8309876544),
(10066, 7654321099),(10066, 7598765433),
(10067, 9988775567),(10067, 9776655444),
(10068, 9456123457),(10068, 9567891235),
(10069, 8776655433),(10069, 8665544333),
(10070, 9901234568),(10070, 9887654323),
(10071, 9812345671),(10071, 9324567891),
(10072, 9998765433),(10072, 9777654321),
(10073, 9000123457),(10073, 9087654324),

(10074, 8998765433),(10074, 8765432111),
 (10075, 9234567811),(10075, 9323456782),
 (10076, 9112345679),(10076, 9087654325),
 (10077, 8765432101),(10077, 8686868687),
 (10078, 9212345679),(10079, 9501234568),
 (10080, 8991001101),(10080, 8877665545),
 (10081, 9356789013),(10081, 9276543211),
 (10082, 9812345679),(10082, 9887654322),
 (10083, 9654321099),(10083, 9745632181),
 (10084, 9501234569),(10084, 9409876544),
 (10085, 9182736451),(10085, 9234567802),
 (10086, 9098765433),(10086, 9012345671),
 (10087, 8888776656),(10087, 8775544333),
 (10088, 9198765433),(10089, 9934567892),
 (10089, 9812345680),(10090, 8765432109);

	volunteer_id [PK] integer	contact_no [PK] bigint
1	10001	9876543210
2	10002	9123456789
3	10002	9345678901
4	10003	8765432109
5	10004	9988112233
6	10004	9900221144
7	10005	9871234567
8	10006	8999000111
9	10006	8912345678

Total rows: 157 of 157 Query complete 00:00:00.331 Ln 1, Col 12

Tuple count – 157

11. Volunteer_skills

```
INSERT INTO Volunteer_skills (Volunteer_ID, skillset) VALUES  
(10001, 'Emergency Response Coordination'),  
(10002, 'Disaster Risk Assessment'),  
(10002, 'Emergency Operations Center (EOC) Setup'),  
(10003, 'Search and Rescue Operations'),  
(10003, 'Psychological First Aid'),  
(10004, 'First Aid and CPR'),  
(10005, 'Resource Management'),  
(10005, 'Logistics Planning'),  
(10006, 'Shelter Management'),  
(10006, 'Flood Response and Recovery'),  
(10007, 'Evacuation Planning'),  
(10008, 'Disaster Recovery Planning'),  
(10009, 'Crisis Intervention'),  
(10010, 'Public Safety Education'),  
(10010, 'Media Relations during Crises'),  
(10011, 'Psychosocial Support'),  
(10011, 'Damage Assessment'),  
(10012, 'Training and Capacity Building'),  
(10012, 'Logistics and Supply Chain Management'),  
(10013, 'Multi-Agency Coordination'),
```

- (10013, 'Community Resilience Planning'),
- (10014, 'Disaster Management Training'),
- (10014, 'Environmental Hazard Assessment'),
- (10015, 'Behavioral Health Support'),
- (10015, 'Rapid Needs Assessment'),
- (10016, 'Fire Safety and Evacuation'),
- (10016, 'Infrastructure Repair Planning'),
- (10017, 'Long-term Recovery Planning'),
- (10017, 'Building Back Better Principles'),
- (10018, 'Emergency Medical Services (EMS) Training'),
- (10018, 'Crisis Negotiation'),
- (10019, 'Community Health Initiatives'),
- (10019, 'Flood Mitigation Strategies'),
- (10020, 'Crisis Logistics Management'),
- (10020, 'Youth Engagement in Disaster Response'),
- (10021, 'Emergency Shelter Operations'),
- (10021, 'Restoration Planning'),
- (10022, 'Disaster Relief Fundraising'),
- (10022, 'Communication Technology in Emergencies'),
- (10023, 'Networking with NGOs'),
- (10023, 'Sustainability in Disaster Recovery'),
- (10024, 'Emergency Response Assessment'),

- (10024, 'Public Awareness Campaigns'),
- (10025, 'Disaster Simulation and Drills'),
- (10025, 'Case Management in Disaster Recovery'),
- (10026, 'Environmental Conservation in Recovery'),
- (10026, 'Volunteer Coordination during Crises'),
- (10027, 'Risk Analysis and Mitigation'),
- (10027, 'Health and Safety Regulations'),
- (10028, 'Crisis Resource Allocation'),
- (10028, 'Data Collection and Analysis'),
- (10029, 'Volunteer Outreach Programs'),
- (10029, 'Disaster Preparedness Workshops'),
- (10030, 'Emergency Incident Reporting'),
- (10030, 'Community Engagement and Support'),
- (10031, 'Disaster Risk Communication'),
- (10031, 'Post-Disaster Evaluation'),
- (10032, 'Urban Search and Rescue'),
- (10032, 'Crisis Social Work'),
- (10033, 'Agricultural Recovery Programs'),
- (10033, 'Mental Health First Aid'),
- (10034, 'Emergency Response Training'),
- (10034, 'Fire and Rescue Operations'),
- (10035, 'Community Disaster Planning'),

- (10035, 'Infrastructure Development Planning'),
- (10036, 'Health Emergency Preparedness'),
- (10036, 'Emergency Evacuation Planning'),
- (10037, 'Volunteer Training and Support'),
- (10037, 'Crisis Assessment and Reporting'),
- (10038, 'Community Resilience Workshops'),
- (10038, 'Flood Recovery Strategies'),
- (10039, 'Psychosocial Support in Emergencies'),
- (10039, 'Shelter Management Training'),
- (10040, 'Disaster Communications Planning'),
- (10040, 'Emergency Operations Training'),
- (10041, 'Health Risk Communication'),
- (10041, 'Disaster Logistics Management'),
- (10042, 'Environmental Impact Assessment'),
- (10042, 'Resource Allocation Planning'),
- (10043, 'Crisis Coordination with Local Authorities'),
- (10043, 'Risk Management Strategies'),
- (10044, 'Water and Sanitation Management'),
- (10044, 'Public Health Campaigns'),
- (10045, 'Long-Term Disaster Recovery'),
- (10045, 'Social Media Management during Crises'),
- (10046, 'Volunteer Recruitment Strategies'),

- (10046, 'Public Information Dissemination'),
- (10047, 'Community Safety Workshops'),
- (10047, 'Emergency Supply Distribution'),
- (10048, 'Wildfire Management'),
- (10048, 'Urban Resilience Planning'),
- (10049, 'Community Health Assessment'),
- (10049, 'Training for Disaster Response'),
- (10050, 'Interagency Collaboration Training'),
- (10050, 'Communication Strategies in Emergencies'),
- (10051, 'Building Community Partnerships'),
- (10051, 'Volunteer Impact Assessment'),
- (10052, 'Crisis Response Simulations'),
- (10052, 'Hazardous Materials Training'),
- (10053, 'Behavioral Health Support in Crises'),
- (10053, 'Disaster Recovery Planning'),
- (10054, 'Emergency Preparedness Training'),
- (10054, 'Disaster Research and Development'),
- (10055, 'Disaster Recovery Funding'),
- (10055, 'Community Needs Assessment'),
- (10056, 'Volunteer Supervision and Support'),
- (10056, 'Crisis Resource Management'),
- (10057, 'Community-Based Disaster Risk Reduction'),

- (10057, 'Emergency Plan Development'),
- (10058, 'Disaster Preparedness Education'),
- (10058, 'Volunteer Training Programs'),
- (10059, 'Emergency Management Simulation Exercises'),
- (10059, 'Environmental Health in Disasters'),
- (10060, 'Disaster Response Planning'),
- (10060, 'Community Resilience Initiatives'),
- (10061, 'Public Health Emergency Response'),
- (10061, 'Crisis Communication Planning'),
- (10062, 'Disaster Fundraising Strategies'),
- (10062, 'Volunteer Management in Emergencies'),
- (10063, 'Crisis Intervention Techniques'),
- (10063, 'Disaster Preparedness Planning'),
- (10064, 'Search and Rescue Training'),
- (10064, 'Emergency Management Best Practices'),
- (10065, 'Volunteer Coordination and Management'),
- (10065, 'Emergency Communications Systems'),
- (10066, 'Disaster Simulation Training'),
- (10066, 'Community Engagement Strategies'),
- (10067, 'Volunteer Recovery Planning'),
- (10067, 'Public Health Outreach during Disasters'),

- (10068, 'Behavioral Health Strategies'),
- (10069, 'Emergency Services Coordination'),
- (10069, 'Community Health Response'),
- (10070, 'Risk Reduction Strategies'),
- (10070, 'Training for First Responders'),
- (10071, 'Evacuation Drill Planning'),
- (10071, 'Community Safety Assessments'),
- (10072, 'Disaster Management Policy Development'),
- (10072, 'Resource Management in Crises'),
- (10073, 'Emergency Preparedness and Response'),
- (10073, 'Mental Health Support in Disasters'),
- (10074, 'Volunteer Skill Development'),
- (10074, 'Crisis Response Training'),
- (10075, 'Community Resilience and Recovery'),
- (10075, 'Psychological Support Training'),
- (10076, 'Logistics Planning for Emergencies'),
- (10077, 'Emergency Response Logistics'),
- (10078, 'Community Resource Management'),
- (10078, 'Disaster Recovery Support'),
- (10079, 'Emergency Response Coordination'),
- (10079, 'Crisis Evaluation Techniques'),
- (10080, 'Risk Communication and Community Engagement'),

(10080, 'Emergency Health Services Planning'),
(10081, 'Volunteer Training for Disaster Response'),
(10081, 'Safety Planning in Disasters'),
(10082, 'Incident Command Training'),
(10082, 'Disaster Management Research'),
(10083, 'Public Health Emergency Management'),
(10083, 'Volunteer Leadership in Disasters'),
(10084, 'Community-Based Response Planning'),
(10084, 'Disaster Training and Education'),
(10085, 'Emergency Response Team Coordination'),
(10086, 'Public Awareness and Education'),
(10086, 'Disaster Logistics Support'),
(10087, 'Crisis Assessment and Response'),
(10087, 'Emergency Management Protocols'),
(10088, 'Community Health Initiatives'),
(10088, 'Training for Crisis Response'),
(10089, 'Emergency Response Training'),
(10090, 'Long-Term Recovery and Support');

A screenshot of a database management system interface. At the top, there are various toolbar icons. Below the toolbar is a table with two columns: 'volunteer_Id' and 'skillset'. The 'volunteer_Id' column contains integers from 1 to 9, and the 'skillset' column contains corresponding text descriptions. The table has a total of 169 rows, as indicated by the status bar at the bottom which says 'Total rows: 169 of 169'. The status bar also shows 'Query complete 00:00:00.149' and 'Ln 1, Col 30'.

	volunteer_Id	skillset
	[PK] integer	[PK] text
1	10001	Emergency Response Coordination
2	10002	Disaster Risk Assessment
3	10002	Emergency Operations Center (EOC) Setup
4	10003	Search and Rescue Operations
5	10003	Psychological First Aid
6	10004	First Aid and CPR
7	10005	Resource Management
8	10005	Logistics Planning
9	10006	Shelter Management

Tuple count – 169

12. EmergencyTeams

INSERT INTO EmergencyTeams (Team_ID, No_of_Members, Availability, NGO_ID, Agency_ID) VALUES

```
(9001, 5, TRUE, 701, NULL),
(9002, 3, TRUE, 710, NULL),
(9003, 4, FALSE, NULL, 1021),
(9004, 6, TRUE, 730, NULL),
(9005, 7, TRUE, NULL, 1010),
(9006, 5, FALSE, 740, 1043),
(9007, 8, TRUE, 738, NULL),
(9008, 3, TRUE, NULL, 1065),
(9009, 4, FALSE, NULL, 1071),
(9010, 6, TRUE, NULL, 1019),
(9011, 7, TRUE, NULL, 1050),
(9012, 5, FALSE, NULL, 1003),
(9013, 3, TRUE, 747, 1027),
```

(9014, 4, FALSE, NULL, 1070),
(9015, 8, TRUE, 746, NULL),
(9016, 5, TRUE, NULL, 1046),
(9017, 3, TRUE, 712, 1011),
(9018, 6, FALSE, NULL, 1056),
(9019, 7, TRUE, NULL, 1034),
(9020, 5, FALSE, NULL, 1025),
(9021, 8, TRUE, NULL, 1068),
(9022, 4, FALSE, 708, 1077),
(9023, 6, TRUE, NULL, 1005),
(9024, 3, TRUE, NULL, 1041),
(9025, 5, FALSE, NULL, 1063),
(9026, 7, TRUE, NULL, 1079),
(9027, 8, FALSE, NULL, 1051),
(9028, 4, TRUE, 788, NULL),
(9029, 5, TRUE, 777, NULL),
(9030, 6, TRUE, NULL, 1031),
(9031, 3, FALSE, NULL, 1023),
(9032, 5, TRUE, 735, 1060),
(9033, 8, FALSE, 763, NULL),
(9034, 4, TRUE, NULL, 1082),
(9035, 7, TRUE, NULL, 1018),

(9036, 6, FALSE, NULL, 1022),
(9037, 3, TRUE, NULL, 1040),
(9038, 5, TRUE, 720, NULL),
(9039, 8, FALSE, 769, NULL),
(9040, 6, TRUE, 704, NULL),
(9041, 7, FALSE, NULL, 1033),
(9042, 5, TRUE, 705, NULL),
(9043, 8, TRUE, NULL, 1075),
(9044, 4, FALSE, NULL, 1062),
(9045, 6, TRUE, NULL, 1047),
(9046, 3, FALSE, NULL, 1006),
(9047, 5, TRUE, NULL, 1054),
(9048, 7, TRUE, 748, 1029),
(9049, 4, FALSE, 701, 1064),
(9050, 8, TRUE, NULL, 1014),
(9051, 5, TRUE, NULL, 1081),
(9052, 6, FALSE, NULL, 1074),
(9053, 7, TRUE, NULL, 1038),
(9054, 3, TRUE, NULL, 1059),
(9055, 8, FALSE, 708, 1044),
(9056, 5, TRUE, NULL, 1069),
(9057, 4, FALSE, 782, NULL),

(9058, 6, TRUE, NULL, 1080),
(9059, 5, FALSE, 715, 1026),
(9060, 7, TRUE, NULL, 1048),
(9061, 8, TRUE, 755, NULL),
(9062, 4, FALSE, NULL, 1061),
(9063, 6, TRUE, NULL, 1030),
(9064, 3, TRUE, 755, NULL),
(9065, 5, TRUE, NULL, 1052),
(9066, 7, FALSE, 776, 1002),
(9067, 8, TRUE, 787, 1012),
(9068, 5, TRUE, NULL, 1082),
(9069, 4, FALSE, NULL, 1001),
(9070, 6, TRUE, NULL, 1066),
(9071, 3, TRUE, 774, NULL),
(9072, 5, TRUE, 783, 1037),
(9073, 8, FALSE, 756, NULL),
(9074, 7, TRUE, NULL, 1008),
(9075, 4, FALSE, 779, 1071),
(9076, 6, TRUE, 701, 1049),
(9077, 3, FALSE, NULL, 1005),
(9078, 5, TRUE, NULL, 1066),
(9079, 8, FALSE, 735, NULL),

```
(9080, 7, TRUE, 787, NULL),
(9081, 5, TRUE, NULL, 1034),
(9082, 4, TRUE, NULL, 1078),
(9083, 3, FALSE, NULL, 1067),
(9084, 6, TRUE, 711, 1003),
(9085, 5, TRUE, NULL, 1057),
(9086, 4, FALSE, NULL, 1013),
(9087, 6, TRUE, 710, NULL);
```

	team_id [PK] integer ↴	no_of_members integer ↴	availability boolean ↴	ngo_id integer ↴	agency_id integer ↴
1	9001	5	true	701	1001
2	9002	3	true	710	1035
3	9003	4	false	725	1021
4	9004	6	true	730	1082
5	9005	7	true	712	1010
6	9006	5	false	740	1043
7	9007	8	true	738	1004
8	9008	3	true	709	1065
9	9009	4	false	706	1071

Total rows: 87 of 87 Query complete 00:00:00.089 Ln 1, Col 22

Tuple count – 87

13. Response

```
INSERT INTO Response (Response_ID, Start_date, End_date, Status,
Disaster_ID, EmergencyTeam_ID) VALUES
(234, '2021-01-15', '2021-01-20', 'In Progress', 85, 9001),
(235, '2021-02-10', '2021-02-15', 'Completed', 1, 9072),
(236, '2021-03-05', '2021-03-10', 'Pending', 2, 9053),
(237, '2021-04-12', '2021-04-16', 'In Progress', 3, 9004),
```

(238, '2021-05-20', '2021-05-25', 'Completed', 4, 9005),
(239, '2021-06-15', '2021-06-20', 'Pending', 5, 9006),
(240, '2021-07-01', '2021-07-06', 'In Progress', 6, 9007),
(241, '2021-08-10', '2021-08-15', 'Completed', 7, 9008),
(242, '2021-09-15', '2021-09-20', 'Pending', 8, 9011),
(243, '2021-10-05', '2021-10-10', 'In Progress', 9, 9010),
(244, '2021-11-10', '2021-11-15', 'Completed', 10, 9045),
(245, '2021-12-01', '2021-12-05', 'Pending', 11, 9012),
(246, '2022-01-10', '2022-01-15', 'In Progress', 12, 9013),
(247, '2022-02-20', '2022-02-25', 'Completed', 13, 9014),
(248, '2022-03-10', '2022-03-15', 'Pending', 14, 9015),
(249, '2022-04-15', '2022-04-20', 'In Progress', 15, 9016),
(250, '2022-05-20', '2022-05-25', 'Completed', 16, 9017),
(251, '2022-06-10', '2022-06-15', 'Pending', 17, 9018),
(252, '2022-07-01', '2022-07-05', 'In Progress', 18, 9019),
(253, '2022-08-10', '2022-08-15', 'Completed', 19, 9020),
(254, '2022-09-15', '2022-09-20', 'Pending', 20, 9021),
(255, '2022-10-05', '2022-10-10', 'In Progress', 21, 9022),
(256, '2022-11-10', '2022-11-15', 'Completed', 22, 9023),
(257, '2022-12-01', '2022-12-05', 'Pending', 23, 9024),
(258, '2023-01-10', '2023-01-15', 'In Progress', 24, 9025),
(259, '2023-02-20', '2023-02-25', 'Completed', 25, 9026),

(260, '2023-03-10', '2023-03-15', 'Pending', 26, 9027),
(261, '2023-04-15', '2023-04-20', 'In Progress', 27, 9028),
(262, '2023-05-20', '2023-05-25', 'Completed', 28, 9029),
(263, '2023-06-10', '2023-06-15', 'Pending', 29, 9030),
(264, '2023-07-01', '2023-07-05', 'In Progress', 30, 9031),
(265, '2023-08-10', '2023-08-15', 'Completed', 31, 9032),
(266, '2023-09-15', '2023-09-20', 'Pending', 32, 9033),
(267, '2023-10-05', '2023-10-10', 'In Progress', 33, 9034),
(268, '2023-11-10', '2023-11-15', 'Completed', 34, 9035),
(269, '2023-12-01', '2023-12-05', 'Pending', 35, 9036),
(270, '2023-01-10', '2023-01-15', 'In Progress', 36, 9037),
(271, '2023-02-20', '2023-02-25', 'Completed', 37, 9058),
(272, '2023-03-10', '2023-03-15', 'Pending', 38, 9039),
(273, '2023-04-15', '2023-04-20', 'In Progress', 39, 9040),
(274, '2023-05-20', '2023-05-25', 'Completed', 40, 9041),
(275, '2023-06-10', '2023-06-15', 'Pending', 41, 9048),
(276, '2023-07-01', '2023-07-05', 'In Progress', 42, 9043),
(277, '2023-08-10', '2023-08-15', 'Completed', 43, 9044),
(278, '2023-09-15', '2023-09-20', 'Pending', 44, 9011),
(279, '2023-10-05', '2023-10-10', 'In Progress', 45, 9046),
(280, '2023-11-10', '2023-11-15', 'Completed', 46, 9047),
(281, '2023-12-01', '2023-12-05', 'Pending', 47, 9042),

(282, '2023-01-10', '2023-01-15', 'In Progress', 48, 9049),
(283, '2023-02-20', '2023-02-25', 'Completed', 49, 9050),
(284, '2023-03-10', '2023-03-15', 'Pending', 50, 9051),
(285, '2023-04-15', '2023-04-20', 'In Progress', 51, 9056),
(286, '2023-05-20', '2023-05-25', 'Completed', 52, 9003),
(287, '2023-06-10', '2023-06-15', 'Pending', 53, 9054),
(288, '2023-07-01', '2023-07-05', 'In Progress', 54, 9055),
(289, '2023-08-10', '2023-08-15', 'Completed', 55, 9052),
(290, '2023-09-15', '2023-09-20', 'Pending', 56, 9057),
(291, '2023-10-05', '2023-10-10', 'In Progress', 57, 9038),
(292, '2023-11-10', '2023-11-15', 'Completed', 58, 9059),
(293, '2023-12-01', '2023-12-05', 'Pending', 59, 9060),
(294, '2023-01-10', '2023-01-15', 'In Progress', 60, 9061),
(295, '2023-02-20', '2023-02-25', 'Completed', 61, 9064),
(296, '2023-03-10', '2023-03-15', 'Pending', 62, 9063),
(297, '2023-04-15', '2023-04-20', 'In Progress', 63, 9062),
(298, '2023-05-20', '2023-05-25', 'Completed', 64, 9065),
(299, '2023-06-10', '2023-06-15', 'Pending', 65, 9066),
(300, '2023-07-01', '2023-07-05', 'In Progress', 66, 9067),
(301, '2023-08-10', '2023-08-15', 'Completed', 67, 9068),
(302, '2023-09-15', '2023-09-20', 'Pending', 68, 9069),
(303, '2023-10-05', '2023-10-10', 'In Progress', 69, 9070),

(304, '2023-11-10', '2023-11-15', 'Completed', 70, 9071),
 (305, '2023-12-01', '2023-12-05', 'Pending', 71, 9002),
 (306, '2023-01-10', '2023-01-15', 'In Progress', 72, 9073),
 (307, '2023-02-20', '2023-02-25', 'Completed', 73, 9074),
 (308, '2023-03-10', '2023-03-15', 'Pending', 74, 9075),
 (309, '2023-04-15', '2023-04-20', 'In Progress', 75, 9076),
 (310, '2023-05-20', '2023-05-25', 'Completed', 76, 9077),
 (311, '2023-06-10', '2023-06-15', 'Pending', 77, 9078),
 (312, '2023-07-01', '2023-07-05', 'In Progress', 78, 9079),
 (313, '2023-08-10', '2023-08-15', 'Completed', 79, 9084),
 (314, '2023-09-15', '2023-09-20', 'Pending', 80, 9081),
 (315, '2023-10-05', '2023-10-10', 'In Progress', 81, 9082),
 (316, '2023-11-10', '2023-11-15', 'Completed', 82, 9083),
 (317, '2023-12-01', '2023-12-05', 'Pending', 83, 9080),
 (318, '2023-01-10', '2023-01-15', 'In Progress', 84, 9085),
 (319, '2023-02-20', '2023-02-25', 'Completed', 85, 9086),
 (320, '2023-03-10', '2023-03-15', 'Pending', 18, 9023);



The screenshot shows a database interface with a toolbar at the top and a table below. The table has columns: response_id, start_date, end_date, status, disaster_id, and emergencyteam_id. The data consists of 87 rows, each with a unique response_id from 1 to 242, and various dates and statuses. The disaster_id column contains values ranging from 1 to 8, and the emergencyteam_id column contains values 9001 through 9011.

	response_id [PK] integer	start_date date	end_date date	status character varying	disaster_id integer	emergencyteam_id integer
1	234	2021-01-15	2021-01-20	In Progress	85	9001
2	235	2021-02-10	2021-02-15	Completed	1	9072
3	236	2021-03-05	2021-03-10	Pending	2	9053
4	237	2021-04-12	2021-04-16	In Progress	3	9004
5	238	2021-05-20	2021-05-25	Completed	4	9005
6	239	2021-06-15	2021-06-20	Pending	5	9006
7	240	2021-07-01	2021-07-06	In Progress	6	9007
8	241	2021-08-10	2021-08-15	Completed	7	9008
9	242	2021-09-15	2021-09-20	Pending	8	9011

Tuple count – 87

14. Victims

```
INSERT INTO victims (Victim_ID, Victim_name, Location, Health_status, Rescue_status, Community_pincode, Community_Location) VALUES  
(701, 'Aarav Singh', 'Karol Bagh', 'Injured', 'Pending', 110001, 'Connaught Place'),  
(702, 'Vivaan Sharma', 'Karol Bagh', 'Stable', 'Rescued', 110002, 'Chandni Chowk'),  
(703, 'Aditya Verma', 'Karol Bagh', 'Critical', 'Pending', 110004, 'Paharganj'),  
(704, 'Vihaan Reddy', 'Janakpuri', 'Stable', 'Rescued', 110005, 'Lajpat Nagar'),  
(705, 'Arjun Joshi', 'Lajpat Nagar', 'Injured', 'Pending', 110007, 'Dwarka'),  
(706, 'Krishna Nair', 'Marine Lines', 'Critical', 'Pending', 400003, 'Bandra'),  
(707, 'Sai Patel', 'Marine Lines', 'Stable', 'Rescued', 400002, 'Dadar'),  
(708, 'Ayaan Kapoor', 'Gachibowli', 'Injured', 'Pending', 500001, 'Banjara Hills'),  
(709, 'Anaya Bhatia', 'Gachibowli', 'Stable', 'Rescued', 110011, 'Saket'),  
(710, 'Saanvi Rao', 'Gachibowli', 'Critical', 'Pending', 500003, 'Secunderabad'),  
(711, 'Ishaan Khan', 'Gachibowli', 'Stable', 'Rescued', 500004, 'Gachibowli'),
```

- (712, 'Riya Malhotra', 'Kolkata', 'Injured', 'Pending', 700002, 'Garia'),
- (713, 'Kunal Menon', 'Bareilly', 'Critical', 'Pending', 800001, 'Shahjahanpur'),
- (714, 'Tanishq Mehta', 'Moradabad', 'Stable', 'Rescued', 800005, 'Agra'),
- (715, 'Maya Reddy', 'Lucknow', 'Injured', 'Pending', 600001, 'T. Nagar'),
- (716, 'Neha Joshi', 'Karol Bagh', 'Stable', 'Rescued', 600002, 'Mylapore'),
- (717, 'Advik Sharma', 'Janakpuri', 'Critical', 'Pending', 600003, 'Nungambakkam'),
- (718, 'Pooja Iyer', 'Lajpat Nagar', 'Stable', 'Rescued', 600004, 'Anna Nagar'),
- (719, 'Yash Verma', 'Marine Lines', 'Injured', 'Pending', 600005, 'Velachery'),
- (720, 'Kavya Dutta', 'Marine Lines', 'Critical', 'Pending', 600006, 'Kotturpuram'),
- (721, 'Rohan Choudhary', 'Gachibowli', 'Stable', 'Rescued', 600007, 'Adyar'),
- (722, 'Siddharth Yadav', 'Gachibowli', 'Injured', 'Pending', 600008, 'Besant Nagar'),
- (723, 'Anika Menon', 'Gachibowli', 'Critical', 'Pending', 600010, 'Tambaram'),
- (724, 'Nishant Patil', 'Kolkata', 'Stable', 'Rescued', 600002, 'Mylapore'),
- (725, 'Lakshmi Rao', 'Bareilly', 'Injured', 'Pending', 400001, 'Marine Lines'),
- (726, 'Rahul Agrawal', 'Karol Bagh', 'Critical', 'Pending', 400004, 'Andheri'),

- (727, 'Tanvi Khanna', 'Lajpat Nagar', 'Stable', 'Rescued', 110011, 'Saket'),
(728, 'Pranav Joshi', 'Dwarka', 'Injured', 'Pending', 500004, 'Gachibowli'),
(729, 'Meera Iyer', 'Janakpuri', 'Critical', 'Pending', 500003, 'Secunderabad'),
(730, 'Neeraj Kapoor', 'Marine Lines', 'Stable', 'Rescued', 110011, 'Saket'),
(731, 'Ritika Agarwal', 'Marine Lines', 'Injured', 'Pending', 400006, 'Navi Mumbai'),
(732, 'Raghav Shukla', 'Karol Bagh', 'Stable', 'Rescued', 400002, 'Dadar'),
(733, 'Jaya Malhotra', 'Lucknow', 'Critical', 'Pending', 400003, 'Bandra'),
(734, 'Shivani Verma', 'Gachibowli', 'Stable', 'Rescued', 500005, 'Hitech City'),
(735, 'Dev Sharma', 'Gachibowli', 'Injured', 'Pending', 500004, 'Gachibowli'),
(736, 'Aditi Kapoor', 'Bareilly', 'Critical', 'Pending', 500003, 'Secunderabad'),
(737, 'Ishaan Kumar', 'Moradabad', 'Stable', 'Rescued', 110011, 'Saket'),
(738, 'Niharika Patel', 'Agra', 'Injured', 'Pending', 400006, 'Navi Mumbai'),
(739, 'Yashvi Sharma', 'Karol Bagh', 'Stable', 'Rescued', 600001, 'T. Nagar'),
(740, 'Anshuman Singh', 'Lajpat Nagar', 'Critical', 'Pending', 600002, 'Mylapore'),
(741, 'Aditi Singh', 'Kolkata', 'Stable', 'Rescued', 600003, 'Nungambakkam'),

- (742, 'Shubham Agarwal', 'Bareilly', 'Injured', 'Pending', 600004, 'Anna Nagar'),
- (743, 'Rohan Kapoor', 'Gachibowli', 'Stable', 'Rescued', 600005, 'Velachery'),
- (744, 'Ishika Malhotra', 'Dwarka', 'Injured', 'Pending', 600006, 'Kotturpuram'),
- (745, 'Nikhil Chaudhary', 'Marine Lines', 'Critical', 'Pending', 600007, 'Adyar'),
- (746, 'Shivam Yadav', 'Janakpuri', 'Stable', 'Rescued', 600008, 'Besant Nagar'),
- (747, 'Tanisha Mehta', 'Moradabad', 'Injured', 'Pending', 600010, 'Tambaram'),
- (748, 'Devansh Joshi', 'Lucknow', 'Stable', 'Rescued', 600002, 'Mylapore'),
- (749, 'Suryansh Kumar', 'Agra', 'Injured', 'Pending', 700002, 'Garia'),
- (750, 'Ravi Sethi', 'Kolkata', 'Stable', 'Rescued', 700005, 'New Town'),
- (751, 'Simran Bansal', 'Gachibowli', 'Critical', 'Pending', 700004, 'Dum Dum'),
- (752, 'Harsh Sharma', 'Bareilly', 'Stable', 'Rescued', 700003, 'Behala'),
- (753, 'Kiran Iyer', 'Dwarka', 'Injured', 'Pending', 700006, 'Howrah'),
- (754, 'Rajiv Rao', 'Lajpat Nagar', 'Stable', 'Rescued', 700009, 'Baranagar'),
- (755, 'Aishwarya Gupta', 'Karol Bagh', 'Injured', 'Pending', 700010, 'Tollygunge'),

- (756, 'Yashasvi Mehta', 'Marine Lines', 'Critical', 'Pending', 600001, 'T. Nagar'),
- (757, 'Neha Bhardwaj', 'Janakpuri', 'Stable', 'Rescued', 600002, 'Mylapore'),
- (758, 'Pulkit Chaudhary', 'Karol Bagh', 'Injured', 'Pending', 600004, 'Anna Nagar'),
- (759, 'Avni Reddy', 'Gachibowli', 'Stable', 'Rescued', 600007, 'Adyar'),
- (760, 'Devraj Singh', 'Gachibowli', 'Critical', 'Pending', 600008, 'Besant Nagar'),
- (761, 'Naina Joshi', 'Marine', 'Stable', 'Rescued', 600010, 'Tambaram'),
- (762, 'Himanshu Verma', 'Lajpat Nagar', 'Injured', 'Pending', 600002, 'Mylapore'),
- (763, 'Samaira Agarwal', 'Bareilly', 'Stable', 'Rescued', 700002, 'Garia'),
- (764, 'Shivangi Gupta', 'Lucknow', 'Injured', 'Pending', 700005, 'New Town'),
- (765, 'Karan Yadav', 'Agra', 'Stable', 'Rescued', 700004, 'Dum Dum'),
- (766, 'Siddharth Patil', 'Moradabad', 'Critical', 'Pending', 700003, 'Behala'),
- (767, 'Tanvi Sharma', 'Kolkata', 'Stable', 'Rescued', 700006, 'Howrah'),
- (768, 'Ayush Singh', 'Gachibowli', 'Injured', 'Pending', 700010, 'Tollygunge'),
- (769, 'Maya Malhotra', 'Janakpuri', 'Stable', 'Rescued', 700002, 'Garia'),
- (770, 'Niharika Yadav', 'Karol Bagh', 'Injured', 'Pending', 700005, 'New Town'),

- (771, 'Aditya Kapoor', 'Marine Lines', 'Critical', 'Pending', 700004, 'Dum Dum'),
- (772, 'Tanish Sharma', 'Lucknow', 'Stable', 'Rescued', 700003, 'Behala'),
- (773, 'Rohan Gupta', 'Lajpat Nagar', 'Injured', 'Pending', 700006, 'Howrah'),
- (774, 'Divya Bansal', 'Bareilly', 'Stable', 'Rescued', 700010, 'Tollygunge'),
- (775, 'Dev Sethi', 'Agra', 'Critical', 'Pending', 600007, 'Adyar'),
- (776, 'Ananya Verma', 'Karol Bagh', 'Stable', 'Rescued', 110001, 'Connaught Place'),
- (777, 'Vikram Joshi', 'Janakpuri', 'Injured', 'Pending', 110002, 'Chandni Chowk'),
- (778, 'Kavya Reddy', 'Marine Lines', 'Stable', 'Rescued', 110004, 'Paharganj'),
- (779, 'Saanvi Iyer', 'Gachibowli', 'Injured', 'Pending', 600001, 'T. Nagar'),
- (780, 'Ishaan Bhatia', 'Bareilly', 'Stable', 'Rescued', 500003, 'Secunderabad'),
- (781, 'Shivani Singh', 'Lucknow', 'Injured', 'Pending', 110011, 'Saket'),
- (782, 'Tanushree Nair', 'Agra', 'Stable', 'Rescued', 400006, 'Navi Mumbai'),
- (783, 'Ritvik Sharma', 'Moradabad', 'Critical', 'Pending', 400003, 'Bandra'),
- (784, 'Anirudh Joshi', 'Kolkata', 'Stable', 'Rescued', 400004, 'Andheri'),
- (785, 'Priya Menon', 'Lajpat Nagar', 'Injured', 'Pending', 400002, 'Dadar'),

(786, 'Aarav Singh', 'Karol Bagh', 'Stable', 'Rescued', 400001, 'Marine Lines'),

(787, 'Vivaan Sharma', 'Gachibowli', 'Critical', 'Pending', 110002, 'Chandni Chowk'),

(788, 'Aditya Verma', 'Marine Lines', 'Stable', 'Rescued', 110001, 'Connaught Place'),

(789, 'Vihaan Reddy', 'Bareilly', 'Injured', 'Pending', 400006, 'Navi Mumbai'),

(790, 'Aarav Reddy', 'Agra', 'Stable', 'Rescued', 700006, 'Howrah');

INSERT INTO victims (Victim_ID, Victim_name, Location, Health_status, Rescue_status, Community_pincode, Community_Location) VALUES

(791, 'Zara Ali', 'Karol Bagh', 'Injured', 'Pending', 110001, 'Connaught Place'),

(792, 'Kabir Verma', 'Marine Lines', 'Stable', 'Rescued', 400002, 'Dadar'),

(793, 'Anika Reddy', 'Lajpat Nagar', 'Critical', 'Pending', 110007, 'Dwarka'),

(794, 'Samir Khan', 'Gachibowli', 'Stable', 'Rescued', 500001, 'Banjara Hills'),

(795, 'Riya Gupta', 'Kolkata', 'Injured', 'Pending', 700002, 'Garia'),

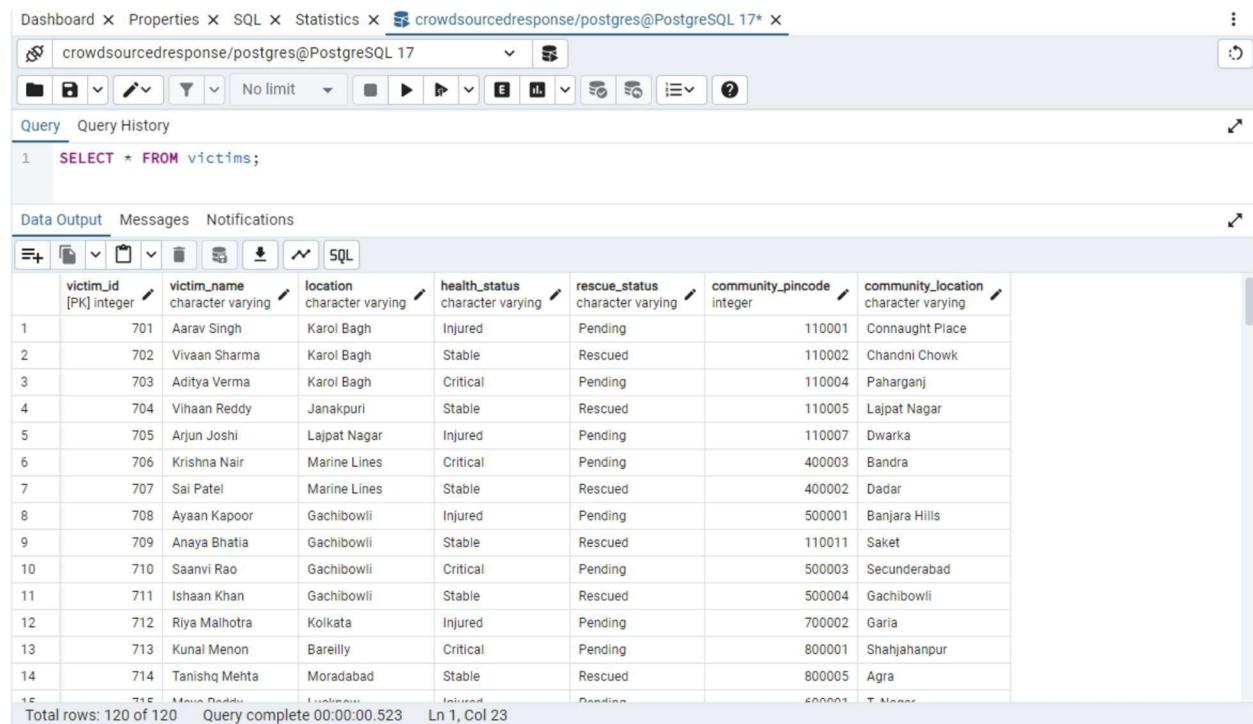
(796, 'Aarav Nair', 'Karol Bagh', 'Stable', 'Rescued', 110001, 'Connaught Place'),

(797, 'Rohan Singh', 'Bareilly', 'Critical', 'Pending', 800001, 'Shahjahanpur'),

(798, 'Diya Sharma', 'Moradabad', 'Stable', 'Rescued', 800005, 'Agra'),

- (799, 'Kiran Iyer', 'Dwarka', 'Injured', 'Pending', 500004, 'Gachibowli'),
(800, 'Maya Choudhary', 'Janakpuri', 'Stable', 'Rescued', 110005, 'Lajpat Nagar'),
(801, 'Tara Mehta', 'Gachibowli', 'Critical', 'Pending', 500003, 'Secunderabad'),
(802, 'Siddharth Rao', 'Karol Bagh', 'Injured', 'Pending', 110001, 'Connaught Place'),
(803, 'Kavya Yadav', 'Gachibowli', 'Stable', 'Rescued', 600007, 'Adyar'),
(804, 'Neha Patil', 'Lucknow', 'Critical', 'Pending', 600001, 'T. Nagar'),
(805, 'Vikrant Sharma', 'Bareilly', 'Stable', 'Rescued', 400006, 'Navi Mumbai'),
(806, 'Ishika Agarwal', 'Karol Bagh', 'Injured', 'Pending', 110001, 'Connaught Place'),
(807, 'Aarav Jain', 'Agra', 'Stable', 'Rescued', 700004, 'Dum Dum'),
(808, 'Nisha Verma', 'Marine Lines', 'Critical', 'Pending', 600001, 'T. Nagar'),
(809, 'Yashasvi Reddy', 'Janakpuri', 'Stable', 'Rescued', 600002, 'Mylapore'),
(810, 'Divya Sharma', 'Lucknow', 'Injured', 'Pending', 600003, 'Nungambakkam'),
(811, 'Amit Khanna', 'Gachibowli', 'Stable', 'Rescued', 500002, 'Jubilee Hills'),
(812, 'Shivam Patel', 'Bareilly', 'Critical', 'Pending', 700003, 'Behala'),
(813, 'Priya Singh', 'Lajpat Nagar', 'Stable', 'Rescued', 600005, 'Velachery'),

(814, 'Anaya Joshi', 'Karol Bagh', 'Injured', 'Pending', 110001, 'Connaught Place'),
 (815, 'Rahul Kumar', 'Gachibowli', 'Stable', 'Rescued', 600004, 'Anna Nagar'),
 (816, 'Ravi Nair', 'Marine Lines', 'Critical', 'Pending', 400005, 'Thane'),
 (817, 'Pooja Reddy', 'Kolkata', 'Stable', 'Rescued', 700002, 'Garia'),
 (818, 'Suryansh Verma', 'Dwarka', 'Injured', 'Pending', 500001, 'Banjara Hills'),
 (819, 'Kavya Gupta', 'Lucknow', 'Stable', 'Rescued', 600002, 'Mylapore'),
 (820, 'Tanisha Bhatia', 'Karol Bagh', 'Critical', 'Pending', 110001, 'Connaught Place');



The screenshot shows a PostgreSQL client interface with the following details:

- Toolbar:** Includes icons for file operations (New, Open, Save, Print), search, and various database management functions.
- Query Bar:** Shows the current connection as "crowdsourcedresponse/postgres@PostgreSQL 17*" and the query text: "SELECT * FROM victims;"
- Data Output Tab:** Active tab, showing the results of the query.
- Table Results:** A grid displaying 120 rows of data from the 'victims' table. The columns are: victim_Id [PK] integer, victim_name character varying, location character varying, health_status character varying, rescue_status character varying, community_pincode integer, and community_location character varying.
- Table Headers:** The first row of the table is labeled with column names and data types.
- Table Data:** Rows 1 through 120 are listed, showing various victim details such as name, location, and status.
- Bottom Status:** Shows "Total rows: 120 of 120" and "Query complete 00:00:00.523 Ln 1, Col 23".

Tuple count – 120

15. Victim_Contacts

```
INSERT INTO Victims_Contacts (Victim_ID, Contact_no) VALUES  
(701, 9876543210), (701, 9876543211), (702, 9876543212), (703,  
9876543213),  
(704, 9876543214), (705, 9876543215), (706, 9876543216), (707,  
9876543217),  
(708, 9876543218), (709, 9876543219), (710, 9876543220), (711,  
9876543221),  
(712, 9876543222), (712, 9876543223), (713, 9876543224), (714,  
9876543225),  
(715, 9876543226), (716, 9876543227), (717, 9876543228), (718,  
9876543229),  
(719, 9876543230), (720, 9876543231), (721, 9876543232), (722,  
9876543233),  
(723, 9876543234), (724, 9876543235), (725, 9876543236), (726,  
9876543237),  
(727, 9876543238), (728, 9876543239), (729, 9876543240), (730,  
9876543241),  
(731, 9876543242), (732, 9876543243), (733, 9876543244), (734,  
9876543245),  
(735, 9876543246), (736, 9876543247), (737, 9876543248), (738,  
9876543249),  
(739, 9876543250), (740, 9876543251), (741, 9876543252), (742,  
9876543253),
```

(743, 9876543254), (744, 9876543255), (745, 9876543256), (746, 9876543257),
(747, 9876543258), (748, 9876543259), (749, 9876543260), (750, 9876543261),
(751, 9876543262), (752, 9876543263), (753, 9876543264), (754, 9876543265),
(755, 9876543266), (756, 9876543267), (757, 9876543268), (758, 9876543269),
(759, 9876543270), (760, 9876543271), (761, 9876543272), (762, 9876543273),
(763, 9876543274), (764, 9876543275), (765, 9876543276), (766, 9876543277),
(767, 9876543278), (768, 9876543279), (769, 9876543280), (770, 9876543281),
(771, 9876543282), (772, 9876543283), (773, 9876543284), (774, 9876543285),
(775, 9876543286), (776, 9876543287), (777, 9876543288), (778, 9876543289),
(779, 9876543290), (780, 9876543291), (781, 9876543292), (782, 9876543293),
(783, 9876543294), (784, 9876543295), (785, 9876543296), (786, 9876543297),
(787, 9876543298), (788, 9876543299), (789, 9876543300), (790, 9876543301);

The screenshot shows a pgAdmin 4 interface with a query editor and a results grid. The query is:

```
1 SELECT * FROM victims_contacts;
```

The results grid displays the following data:

	victim_id	contact_no
1	701	9876543210
2	701	9876543211
3	702	9876543212
4	703	9876543213
5	704	9876543214
6	705	9876543215
7	706	9876543216
8	707	9876543217
9	708	9876543218
10	709	9876543219
11	710	9876543220
12	711	9876543221
13	712	9876543222
14	712	9876543223

Total rows: 92 of 92 Query complete 00:00:00.275 Ln 1, Col 31

Tuple count – 92

16. Disaster Communities

INSERT INTO Disaster_Communities (Disaster_ID, Community_Pincode, Community_Location) VALUES

(1, 110001, 'Connaught Place'),

(1, 110002, 'Chandni Chowk'),

(1, 110003, 'Karol Bagh'),

(1, 400001, 'Marine Lines'),

(2, 400002, 'Dadar'),

(2, 400003, 'Bandra'),

(2, 500001, 'Banjara Hills'),

(3, 110004, 'Paharganj'),

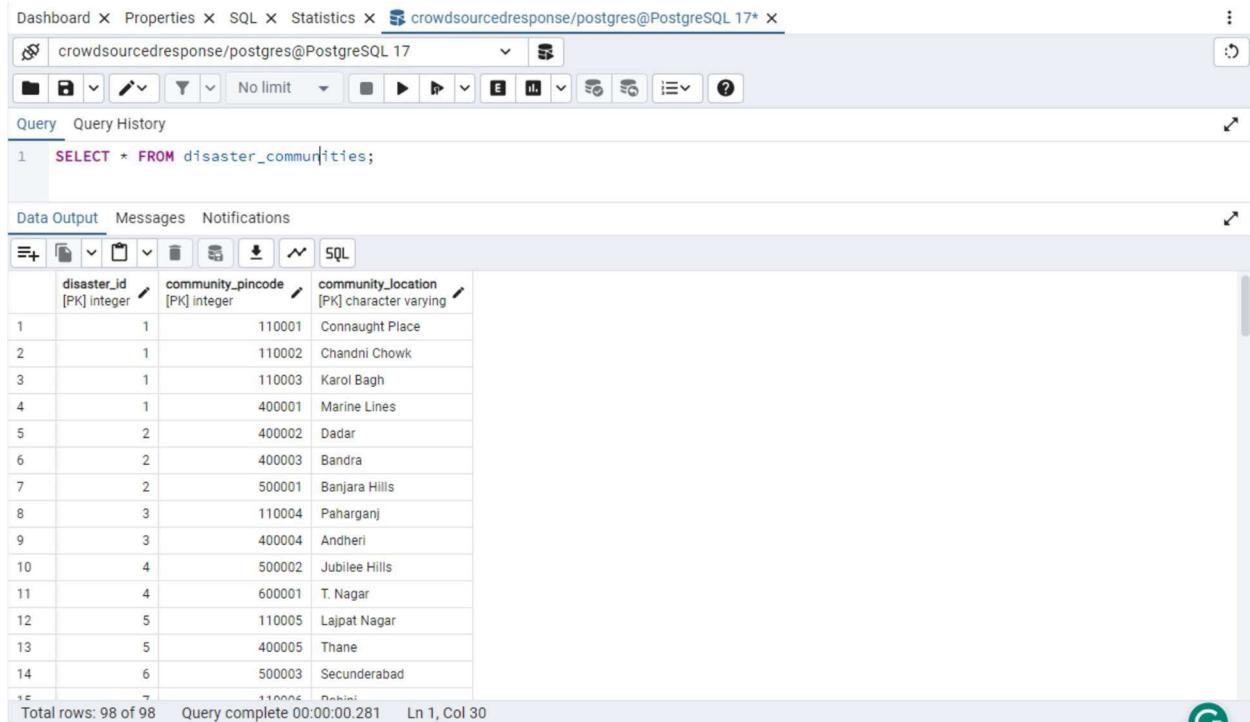
- (3, 400004, 'Andheri'),
- (4, 500002, 'Jubilee Hills'),
- (4, 600001, 'T. Nagar'),
- (5, 110005, 'Lajpat Nagar'),
- (5, 400005, 'Thane'),
- (6, 500003, 'Secunderabad'),
- (7, 110006, 'Rohini'),
- (7, 400006, 'Navi Mumbai'),
- (8, 500004, 'Gachibowli'),
- (9, 110007, 'Dwarka'),
- (9, 400007, 'Borivali'),
- (10, 500005, 'Hitech City'),
- (11, 110008, 'Janakpuri'),
- (11, 400008, 'Malad'),
- (12, 500006, 'Madhapur'),
- (13, 110009, 'Preet Vihar'),
- (13, 400009, 'Versova'),
- (14, 500007, 'LB Nagar'),
- (15, 110010, 'Kalkaji'),
- (15, 400010, 'Vashi'),
- (16, 600002, 'Mylapore'),
- (17, 600003, 'Nungambakkam'),

- (18, 600004, 'Anna Nagar'),
- (19, 600005, 'Velachery'),
- (20, 700001, 'Salt Lake City'),
- (21, 700002, 'Garia'),
- (22, 700003, 'Behala'),
- (23, 700004, 'Dum Dum'),
- (24, 700005, 'New Town'),
- (25, 700006, 'Howrah'),
- (26, 700007, 'Bidhannagar'),
- (27, 700008, 'Kolkata'),
- (28, 700009, 'Baranagar'),
- (29, 700010, 'Tollygunge'),
- (30, 800001, 'Shahjahanpur'),
- (31, 800002, 'Bareilly'),
- (32, 800003, 'Moradabad'),
- (33, 800004, 'Aligarh'),
- (34, 800005, 'Agra'),
- (35, 800006, 'Lucknow'),
- (36, 800007, 'Kanpur'),
- (37, 800008, 'Noida'),
- (38, 800009, 'Ghaziabad'),
- (39, 800010, 'Meerut'),

- (40, 900001, 'Kathmandu'),
- (41, 900002, 'Biratnagar'),
- (42, 900003, 'Lalitpur'),
- (43, 900004, 'Bhaktapur'),
- (44, 900005, 'Pokhara'),
- (45, 900006, 'Nepalgunj'),
- (46, 900007, 'Janakpur'),
- (47, 900008, 'Birgunj'),
- (48, 900009, 'Bhairahawa'),
- (49, 900010, 'Itahari'),
- (50, 950001, 'Dhaka'),
- (51, 950002, 'Chittagong'),
- (52, 950003, 'Khulna'),
- (53, 950004, 'Rajshahi'),
- (54, 950005, 'Sylhet'),
- (55, 950006, 'Barisal'),
- (56, 950007, 'Comilla'),
- (57, 950008, 'Narayanganj'),
- (58, 950009, 'Bogra'),
- (59, 950010, 'Tangail'),
- (60, 110011, 'Saket'),
- (61, 500011, 'KPHB Colony'),

- (62, 600011, 'Perungudi'),
- (63, 110001, 'Connaught Place'),
- (64, 110002, 'Chandni Chowk'),
- (65, 110003, 'Karol Bagh'),
- (66, 400001, 'Marine Lines'),
- (67, 400002, 'Dadar'),
- (68, 400003, 'Bandra'),
- (69, 500001, 'Banjara Hills'),
- (70, 110004, 'Paharganj'),
- (71, 400004, 'Andheri'),
- (72, 500002, 'Jubilee Hills'),
- (73, 600001, 'T. Nagar'),
- (74, 110005, 'Lajpat Nagar'),
- (75, 400005, 'Thane'),
- (76, 500003, 'Secunderabad'),
- (77, 110006, 'Rohini'),
- (78, 400006, 'Navi Mumbai'),
- (79, 500004, 'Gachibowli'),
- (80, 110007, 'Dwarka'),
- (81, 400007, 'Borivali'),
- (82, 500005, 'Hitech City'),
- (83, 110008, 'Janakpuri'),

```
(84, 400008, 'Malad'),
(85, 500006, 'Madhapur');
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar says "Dashboard × Properties × SQL × Statistics × crowdsourcedresponse/postgres@PostgreSQL 17*". The query tab contains the following SQL:

```
1  SELECT * FROM disaster_communities;
```

The results are displayed in a Data Output tab, showing a table with three columns: disaster_id, community_pincode, and community_location. The data consists of 98 rows, with the first few rows being:

	disaster_id [PK] integer	community_pincode [PK] integer	community_location [PK] character varying
1	1	110001	Connaught Place
2	1	110002	Chandni Chowk
3	1	110003	Karol Bagh
4	1	400001	Marine Lines
5	2	400002	Dadar
6	2	400003	Bandra
7	2	500001	Banjara Hills
8	3	110004	Paharganj
9	3	400004	Andheri
10	4	500002	Jubilee Hills
11	4	600001	T. Nagar
12	5	110005	Lajpat Nagar
13	5	400005	Thane
14	6	500003	Secunderabad

Total rows: 98 of 98 Query complete 00:00:00.281 Ln 1, Col 30

Tuple count – 98

17. Disaster_Volunteers

```
INSERT INTO Disaster_Volunteers (Disaster_ID, Volunteer_ID) VALUES
(1, 10001), (1, 10002), (1, 10003), (2, 10004), (2, 10005),
(2, 10006), (3, 10007), (3, 10008), (4, 10009), (4, 10010),
(5, 10011), (5, 10012), (5, 10013), (6, 10014), (6, 10015),
(7, 10016), (7, 10017), (8, 10018), (8, 10019), (9, 10020),
(9, 10021), (10, 10022), (10, 10023), (11, 10024), (11, 10025),
```

(12, 10026), (12, 10027), (13, 10028), (13, 10029), (14, 10030),
(15, 10031), (15, 10032), (16, 10033), (16, 10034), (17, 10035),
(18, 10036), (18, 10037), (19, 10038), (20, 10039), (20, 10040),
(21, 10041), (21, 10042), (22, 10043), (22, 10044), (23, 10045),
(24, 10046), (24, 10047), (25, 10048), (25, 10049), (26, 10050),
(26, 10051), (27, 10052), (27, 10053), (28, 10054), (29, 10055),
(30, 10056), (30, 10057), (31, 10058), (31, 10059), (32, 10060),
(32, 10061), (33, 10062), (33, 10063), (34, 10064), (35, 10065),
(36, 10066), (36, 10067), (37, 10068), (37, 10069), (38, 10070),
(39, 10071), (39, 10072), (40, 10073), (41, 10074), (41, 10075),
(42, 10076), (42, 10077), (43, 10078), (43, 10079), (44, 10080),
(45, 10081), (45, 10082), (46, 10083), (46, 10084), (47, 10085),
(48, 10086), (48, 10087), (49, 10088), (50, 10089), (51, 10090),
(52, 10001), (53, 10002), (54, 10003), (55, 10004), (56, 10005),
(57, 10006), (58, 10007), (59, 10008), (60, 10009), (61, 10010),
(62, 10011), (63, 10012), (64, 10013), (65, 10014), (66, 10015),
(67, 10016), (68, 10017), (69, 10018), (70, 10019), (71, 10020),
(72, 10021), (73, 10022), (74, 10023), (75, 10024), (76, 10025),
(77, 10026), (78, 10027), (79, 10028), (80, 10029), (81, 10030),
(82, 10031), (83, 10032), (84, 10033), (85, 10034);

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar indicates the connection is to 'crowdsourcedresponse/postgres@PostgreSQL 17*'. The query entered is 'SELECT * FROM disaster_volunteers;'. The results pane displays a table with two columns: 'disaster_id' and 'volunteer_id'. The data consists of 124 rows, with disaster_ids ranging from 1 to 15 and volunteer_ids ranging from 10001 to 1014. A success message at the bottom right states 'Successfully run. Total query runtime: 323 msec. 124 rows affected.'

	disaster_id [PK] integer	volunteer_id [PK] integer
1	1	10001
2	1	10002
3	1	10003
4	2	10004
5	2	10005
6	2	10006
7	3	10007
8	3	10008
9	4	10009
10	4	10010
11	5	10011
12	5	10012
13	5	10013
14	6	10014

Total rows: 124 of 124 Query complete 00:00:00.323 Ln 1, Col 35

Tuple count – 124

18. Disaster Agencies

```
INSERT INTO Disaster_Agencies (Disaster_ID, Agency_ID) VALUES
(1, 1001), (1, 1003), (1, 1012), (2, 1005),
(2, 1009), (3, 1010), (3, 1015), (3, 1018),
(4, 1025), (5, 1002), (5, 1013), (6, 1016),
(6, 1027), (7, 1005), (8, 1007), (8, 1008),
(8, 1024), (8, 1035), (9, 1041), (10, 1002),
(10, 1009), (10, 1014), (11, 1028), (11, 1043),
(12, 1003), (12, 1017), (13, 1004), (13, 1008),
(13, 1037), (14, 1032), (14, 1045), (15, 1006),
```

(16, 1047), (17, 1003), (17, 1015), (17, 1050),
(18, 1004), (18, 1029), (18, 1051), (19, 1014),
(19, 1020), (20, 1002), (20, 1053), (21, 1054),
(22, 1007), (22, 1055), (23, 1006), (23, 1026),
(23, 1056), (24, 1004), (24, 1010), (24, 1024),
(24, 1057), (25, 1001), (25, 1008), (25, 1023),
(25, 1058), (26, 1002), (26, 1011), (26, 1040),
(26, 1059), (27, 1003), (27, 1015), (28, 1032),
(28, 1061), (29, 1005), (29, 1013), (29, 1043),
(29, 1062), (30, 1063), (31, 1002), (31, 1009),
(31, 1019), (31, 1064), (32, 1003), (32, 1065),
(33, 1066), (34, 1004), (34, 1067), (35, 1002),
(35, 1015), (36, 1003), (36, 1006), (37, 1007),
(37, 1011), (38, 1005), (38, 1008), (38, 1033),
(38, 1071), (39, 1004), (39, 1017), (39, 1029),
(39, 1072), (40, 1001), (40, 1002), (40, 1018),
(40, 1073), (41, 1041), (41, 1074), (42, 1004),
(42, 1010), (42, 1025), (42, 1075), (43, 1008),
(43, 1019), (43, 1036), (43, 1076), (44, 1007),
(44, 1005), (44, 1022), (45, 1039), (45, 1078),
(46, 1002), (46, 1012), (46, 1023), (46, 1079),
(47, 1006), (47, 1080), (48, 1004), (48, 1014),

(48, 1021), (48, 1081), (49, 1026), (49, 1082),
(50, 1001), (50, 1005), (50, 1013), (50, 1024),
(51, 1002), (51, 1009), (51, 1017), (51, 1028),
(52, 1003), (52, 1006), (52, 1018), (52, 1030),
(53, 1004), (53, 1007), (53, 1022), (53, 1039),
(54, 1001), (54, 1011), (54, 1023), (54, 1045),
(55, 1005), (55, 1014), (55, 1034), (55, 1047),
(56, 1006), (56, 1008), (56, 1025), (56, 1051),
(57, 1002), (57, 1010), (57, 1026), (57, 1053),
(58, 1003), (58, 1012), (58, 1027), (58, 1055),
(59, 1004), (59, 1015), (59, 1040), (59, 1056),
(60, 1001), (60, 1009), (60, 1016), (60, 1057),
(61, 1002), (61, 1018), (61, 1031), (61, 1061),
(62, 1003), (62, 1008), (62, 1035), (62, 1063),
(63, 1004), (63, 1019), (63, 1042), (63, 1064),
(64, 1005), (64, 1007), (64, 1021), (64, 1065),
(65, 1006), (65, 1003), (65, 1023), (65, 1066),
(66, 1001), (66, 1002), (66, 1029), (66, 1067),
(67, 1004), (67, 1011), (67, 1030), (67, 1068),
(68, 1002), (68, 1008), (68, 1041), (68, 1069),
(69, 1003), (69, 1014), (69, 1033), (69, 1070),
(70, 1005), (70, 1007), (70, 1028), (70, 1071),

(71, 1006), (71, 1009), (71, 1012), (71, 1072),
(72, 1004), (72, 1016), (72, 1034), (72, 1073),
(73, 1001), (73, 1015), (73, 1045), (73, 1074),
(74, 1002), (74, 1013), (74, 1024), (74, 1075),
(75, 1003), (75, 1008), (75, 1036), (75, 1076),
(76, 1004), (76, 1010), (76, 1025), (76, 1077),
(77, 1001), (77, 1018), (77, 1040), (77, 1078),
(78, 1002), (78, 1017), (78, 1029), (78, 1079),
(79, 1003), (79, 1006), (79, 1032), (79, 1080),
(80, 1004), (80, 1011), (80, 1042), (80, 1081),
(81, 1005), (81, 1012), (81, 1028), (81, 1082),
(82, 1001), (82, 1014), (82, 1034), (83, 1002),
(83, 1009), (83, 1027), (83, 1031), (84, 1003),
(84, 1008), (84, 1013), (85, 1004), (85, 1010),
(85, 1015), (85, 1016);

The screenshot shows a PostgreSQL client interface with the following details:

- Connection:** crowdsourcedresponse/postgres@PostgreSQL 17
- Query Bar:** SELECT * FROM disaster_agencies;
- Data Output:** The results are displayed in a table with two columns: disaster_id and agency_id.
- Table Data:**

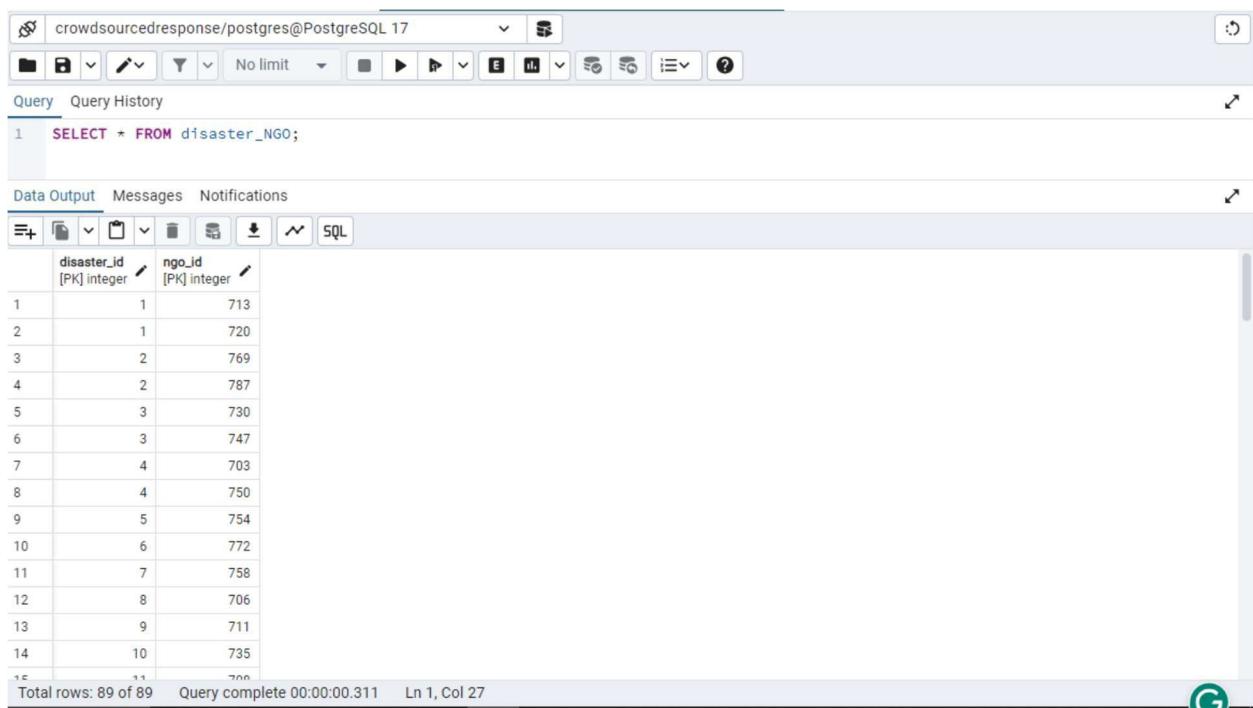
	disaster_id [PK] integer	agency_id [PK] integer
1	1	1001
2	1	1003
3	1	1012
4	2	1005
5	2	1009
6	3	1010
7	3	1015
8	3	1018
9	4	1025
10	5	1002
11	5	1013
12	6	1016
13	6	1027
14	7	1005
15	8	1007
- Message Bar:** Successfully run. Total query runtime: 268 msec. 266 rows affected.
- Status:** Total rows: 266 of 266 Query complete 00:00:00.268 Ln 1, Col 31

Tuple count – 266

19. Disaster_NGOs

```
INSERT INTO Disaster_NGO (Disaster_ID, NGO_ID) VALUES
(1, 713),(1, 720),(2, 769),(2, 787),(3, 730),
(3, 747),(4, 703),(4, 750),(5, 754),(6, 772),
(7, 758),(8, 706),(9, 711),(10, 735),(11, 708),
(12, 775),(13, 762),(14, 787),(15, 785),(16, 732),
(17, 778),(18, 718),(19, 742),(20, 764),(21, 717),
(22, 725),(23, 721),(24, 712),(25, 779),(26, 752),
(27, 740),(28, 707),(29, 710),(30, 736),(31, 782),
(32, 763),(33, 725),(34, 774),(35, 746),(36, 762),
(37, 743),(38, 758),(39, 769),(40, 775),(41, 780),
```

(42, 740), (43, 742), (44, 753), (45, 754), (46, 784), (47, 726),
 (48, 777), (49, 771), (50, 737), (51, 715), (52, 706), (53, 788),
 (54, 782), (55, 709), (56, 773), (57, 771), (58, 735), (59, 786),
 (60, 787), (61, 714), (62, 766), (63, 759), (64, 766), (65, 772),
 (66, 764), (67, 761), (68, 720), (69, 721), (70, 725), (71, 740),
 (72, 755), (73, 762), (74, 774), (75, 748), (76, 715), (77, 783),
 (78, 772), (79, 713), (80, 780), (81, 706), (82, 738), (83, 731),
 (84, 754), (85, 715);



The screenshot shows a PostgreSQL database client interface. The top bar displays the connection information: 'crowdsourcedresponse/postgres@PostgreSQL 17'. Below the toolbar, the 'Query' tab is selected, showing the SQL command: 'SELECT * FROM disaster_NGO;'. The main area displays the 'Data Output' tab, which contains a table with two columns: 'disaster_id' and 'ngo_id'. The table has 14 rows, each containing a unique pair of disaster and NGO IDs. At the bottom of the interface, it says 'Total rows: 89 of 89' and 'Query complete 00:00:00.311 Ln 1, Col 27'.

disaster_id	ngo_id
1	713
2	720
3	769
4	787
5	730
6	747
7	703
8	750
9	754
10	772
11	758
12	706
13	711
14	735

Tuple count – 89

20. Disaster_Victims

```
INSERT INTO Disaster_Victims (Disaster_ID, Victim_ID) VALUES  
(1, 712),(2, 715),(3, 804),(4, 788),(5, 719),  
(6, 794),(7, 735),(8, 709),(9, 718),(10, 791),  
(11, 775),(12, 805),(13, 773),(14, 766),(15, 722),  
(16, 810),(17, 724),(18, 784),(19, 702),(20, 815),  
(21, 788),(22, 801),(23, 715),(24, 711),(25, 792),  
(26, 820),(27, 713),(28, 707),(29, 799),(30, 793),  
(31, 746),(32, 776),(33, 786),(34, 710),(35, 706),  
(36, 762),(37, 704),(38, 707),(39, 768),(40, 748),  
(41, 725),(42, 798),(43, 790),(44, 703),(45, 797),  
(46, 755),(47, 702),(48, 721),(49, 767),(50, 817),  
(51, 755),(52, 715),(53, 713),(54, 803),(55, 785),  
(56, 785),(57, 709),(58, 762),(59, 758),(60, 715),  
(61, 712),(62, 804),(63, 790),(64, 815),(65, 707),  
(66, 774),(67, 758),(68, 724),(69, 809),(70, 730),  
(71, 795),(72, 715),(73, 783),(74, 755),(75, 703),  
(76, 781),(77, 701),(78, 772),(79, 802),(80, 820),  
(81, 773),(82, 787),(83, 712),(84, 760),(85, 792);
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar says "Dashboard × Properties × SQL × Statistics × crowdsourcedresponse/postgres@PostgreSQL 17*". The query tab contains the SQL command: "SELECT * FROM disaster_victims;". Below the query is a table titled "Data Output" showing the results of the query. The table has two columns: "disaster_id" and "victim_id". The data consists of 14 rows, each with a disaster_id from 1 to 14 and a corresponding victim_id. A green success message at the bottom right of the table area says "Successfully run. Total 85 rows".

disaster_id	victim_id
1	712
2	715
3	804
4	788
5	719
6	794
7	735
8	709
9	718
10	791
11	775
12	805
13	773
14	766

Tuple count – 85

21. Disaster_SocialMedia

```
INSERT INTO Disaster_SocialMedia (Disaster_ID, PlatformName) VALUES
(1, 'Facebook'), (1, 'Twitter'), (1, 'Instagram'),
(1, 'YouTube'), (1, 'WhatsApp'),
(2, 'Snapchat'), (2, 'TikTok'), (2, 'Reddit'),
(2, 'Telegram'), (2, 'LinkedIn'),
(3, 'Pinterest'), (3, 'Nextdoor'), (3, 'WeChat'),
(3, 'Signal'), (3, 'VK'),
(4, 'Line'), (4, 'Quora'), (4, 'Clubhouse'),
(4, 'Tumblr'), (4, 'Flickr'),
```

(5, 'Meetup'), (5, 'Discord'), (5, 'Koo'),
(5, 'Viber'), (5, 'Hike'),
(6, 'Medium'), (6, 'Mix'), (6, 'Periscope'),
(6, 'MeWe'), (6, 'Parler'),
(7, 'Gab'), (7, 'Blogger'), (7, 'Substack'),
(7, 'Hive Social'), (7, 'Diaspora'),
(8, 'Mastodon'), (8, 'Steemit'), (8, 'Rumble'),
(8, 'Locals'), (8, 'Vero'),
(9, 'Flipboard'), (9, 'Zello'), (9, 'Citizen'),
(9, 'Alert Media'), (9, 'Nixle'),
(10, 'PulsePoint'), (10, 'Earthquake Network'), (10, 'ReliefWeb'),
(10, 'DisasterAlert'), (10, 'SAFER Together'),
(11, 'AlertHub'), (11, 'ReadyApp'), (11, 'SkyAlert'),
(11, 'AlertNet'), (11, 'LiveSafe'),
(12, 'MyShake'), (12, 'Safety Check by Facebook'), (12, 'Zello Walkie Talkie'),
(12, 'Eviate'), (12, 'Weather Underground'),
(13, 'AccuWeather'), (13, 'The Weather Channel'), (13, 'NOAA Weather App'),
(13, 'Earthquake Tracker'), (13, 'WeatherBug'),
(14, 'National Hurricane Center'), (14, 'Met Office Weather'), (14, 'Disaster Info'),
(14, 'American Red Cross App'), (14, 'FEMA App'),

(15, 'Alert Ready'), (15, 'SafetyNet'), (15, 'FindShelter'),
(15, 'Disaster Ready'), (15, 'SafeCast'),
(16, 'AlertReady Canada'), (16, 'Tsunami Alert'), (16, 'QuakeFeed'),
(16, 'Hurricane Tracker'), (16, 'WeatherNow'),
(17, 'RescueNet'), (17, 'AlertMe'), (17, 'HelpLink'),
(17, 'Global Disaster Alert'), (17, 'Shelter Connect'),
(18, 'CrisisConnect'), (18, 'GeoNet'),
(19, 'Facebook'), (19, 'Twitter'), (19, 'Instagram'),
(19, 'YouTube'), (19, 'WhatsApp'),
(20, 'Snapchat'), (20, 'TikTok'), (20, 'Reddit'),
(20, 'Telegram'), (20, 'LinkedIn'),
(21, 'Pinterest'), (21, 'Nextdoor'), (21, 'WeChat'),
(21, 'Signal'), (21, 'VK'),
(22, 'Line'), (22, 'Quora'), (22, 'Clubhouse'),
(22, 'Tumblr'), (22, 'Flickr'),
(23, 'Meetup'), (23, 'Discord'), (23, 'Koo'),
(23, 'Viber'), (23, 'Hike'),
(24, 'Medium'), (24, 'Mix'), (24, 'Periscope'),
(24, 'MeWe'), (24, 'Parler'),
(25, 'Gab'), (25, 'Blogger'), (25, 'Substack'),
(25, 'Hive Social'), (25, 'Diaspora'),
(26, 'Mastodon'), (26, 'Steemit'), (26, 'Rumble'),

(26, 'Locals'), (26, 'Vero'),
(27, 'Flipboard'), (27, 'Zello'), (27, 'Citizen'),
(27, 'Alert Media'), (27, 'Nixle'),
(28, 'PulsePoint'), (28, 'Earthquake Network'), (28, 'ReliefWeb'),
(28, 'DisasterAlert'), (28, 'SAFER Together'),
(29, 'AlertHub'), (29, 'ReadyApp'), (29, 'SkyAlert'),
(29, 'AlertNet'), (29, 'LiveSafe'),
(30, 'MyShake'), (30, 'Safety Check by Facebook'), (30, 'Zello Walkie Talkie'),
(30, 'Eviate'), (30, 'Weather Underground'),
(31, 'AccuWeather'), (31, 'The Weather Channel'), (31, 'NOAA Weather App'),
(31, 'Earthquake Tracker'), (31, 'WeatherBug'),
(32, 'National Hurricane Center'), (32, 'Met Office Weather'), (32, 'Disaster Info'),
(32, 'American Red Cross App'), (32, 'FEMA App'),
(33, 'Alert Ready'), (33, 'SafetyNet'), (33, 'FindShelter'),
(33, 'Disaster Ready'), (33, 'SafeCast'),
(34, 'AlertReady Canada'), (34, 'Tsunami Alert'), (34, 'QuakeFeed'),
(34, 'Hurricane Tracker'), (34, 'WeatherNow'),
(35, 'RescueNet'), (35, 'AlertMe'), (35, 'HelpLink'),
(35, 'Global Disaster Alert'), (35, 'Shelter Connect'),
(36, 'CrisisConnect'), (36, 'GeoNet'), (37, 'Facebook'),

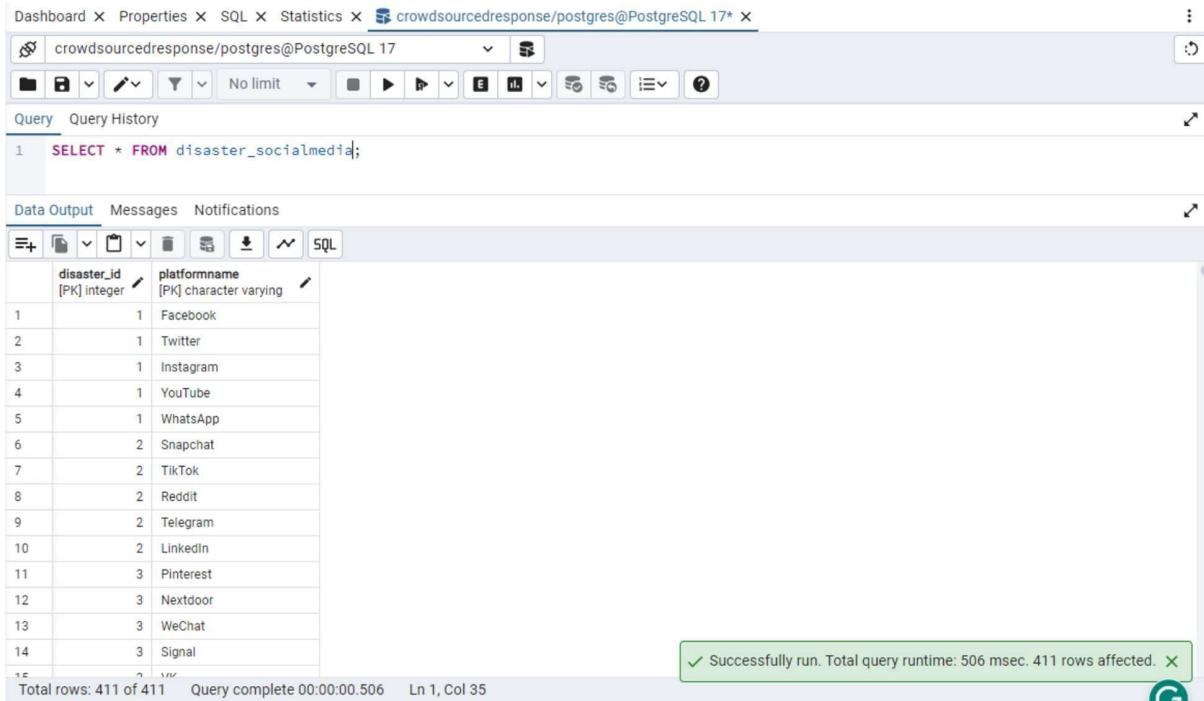
(37, 'Twitter'), (37, 'Instagram'),
(37, 'YouTube'), (37, 'WhatsApp'),
(38, 'Snapchat'), (38, 'TikTok'), (38, 'Reddit'),
(38, 'Telegram'), (38, 'LinkedIn'),
(39, 'Pinterest'), (39, 'Nextdoor'), (39, 'WeChat'),
(39, 'Signal'), (39, 'VK'),
(40, 'Line'), (40, 'Quora'), (40, 'Clubhouse'),
(40, 'Tumblr'), (40, 'Flickr'),
(41, 'Meetup'), (41, 'Discord'), (41, 'Koo'),
(41, 'Viber'), (41, 'Hike'),
(42, 'Medium'), (42, 'Mix'), (42, 'Periscope'),
(42, 'MeWe'), (42, 'Parler'),
(43, 'Gab'), (43, 'Blogger'), (43, 'Substack'),
(43, 'Hive Social'), (43, 'Diaspora'),
(44, 'Mastodon'), (44, 'Steemit'), (44, 'Rumble'),
(44, 'Locals'), (44, 'Vero'),
(45, 'Flipboard'), (45, 'Zello'), (45, 'Citizen'),
(45, 'Alert Media'), (45, 'Nixle'),
(46, 'PulsePoint'), (46, 'Earthquake Network'), (46, 'ReliefWeb'),
(46, 'DisasterAlert'), (46, 'SAFER Together'),
(47, 'AlertHub'), (47, 'ReadyApp'), (47, 'SkyAlert'),
(47, 'AlertNet'), (47, 'LiveSafe'),

(48, 'MyShake'), (48, 'Safety Check by Facebook'), (48, 'Zello Walkie Talkie'),
(48, 'Eviate'), (48, 'Weather Underground'),
(49, 'AccuWeather'), (49, 'The Weather Channel'), (49, 'NOAA Weather App'),
(49, 'Earthquake Tracker'), (49, 'WeatherBug'),
(50, 'National Hurricane Center'), (50, 'Met Office Weather'), (50, 'Disaster Info'),
(50, 'American Red Cross App'), (50, 'FEMA App'),
(51, 'Alert Ready'), (51, 'SafetyNet'), (51, 'FindShelter'),
(51, 'Disaster Ready'), (51, 'SafeCast'),
(52, 'AlertReady Canada'), (52, 'Tsunami Alert'), (52, 'QuakeFeed'),
(52, 'Hurricane Tracker'), (52, 'WeatherNow'),
(53, 'RescueNet'), (53, 'AlertMe'), (53, 'HelpLink'),
(53, 'Global Disaster Alert'), (53, 'Shelter Connect'),
(54, 'CrisisConnect'), (54, 'GeoNet'),
(55, 'Facebook'), (55, 'Twitter'), (55, 'Instagram'),
(55, 'YouTube'), (55, 'WhatsApp'),
(56, 'Snapchat'), (56, 'TikTok'), (56, 'Reddit'),
(56, 'Telegram'), (56, 'LinkedIn'),
(57, 'Pinterest'), (57, 'Nextdoor'), (57, 'WeChat'),
(57, 'Signal'), (57, 'VK'),
(58, 'Line'), (58, 'Quora'), (58, 'Clubhouse'),

(58, 'Tumblr'), (58, 'Flickr'),
(59, 'Meetup'), (59, 'Discord'), (59, 'Koo'),
(59, 'Viber'), (59, 'Hike'),
(60, 'Medium'), (60, 'Mix'), (60, 'Periscope'),
(60, 'MeWe'), (60, 'Parler'),
(61, 'Gab'), (61, 'Blogger'), (61, 'Substack'),
(61, 'Hive Social'), (61, 'Diaspora'),
(62, 'Mastodon'), (62, 'Steemit'), (62, 'Rumble'),
(62, 'Locals'), (62, 'Vero'),
(63, 'Flipboard'), (63, 'Zello'), (63, 'Citizen'),
(63, 'Alert Media'), (63, 'Nixle'),
(64, 'PulsePoint'), (64, 'Earthquake Network'), (64, 'ReliefWeb'),
(64, 'DisasterAlert'), (64, 'SAFER Together'),
(65, 'AlertHub'), (65, 'ReadyApp'), (65, 'SkyAlert'),
(65, 'AlertNet'), (65, 'LiveSafe'),
(66, 'MyShake'), (66, 'Safety Check by Facebook'), (66, 'Zello Walkie Talkie'),
(66, 'Eviate'), (66, 'Weather Underground'),
(67, 'AccuWeather'), (67, 'The Weather Channel'), (67, 'NOAA Weather App'),
(67, 'Earthquake Tracker'), (67, 'WeatherBug'),
(68, 'National Hurricane Center'), (68, 'Met Office Weather'), (68, 'Disaster Info'),

(68, 'American Red Cross App'), (68, 'FEMA App'),
(69, 'Alert Ready'), (69, 'SafetyNet'), (69, 'FindShelter'),
(69, 'Disaster Ready'), (69, 'SafeCast'),
(70, 'AlertReady Canada'), (70, 'Tsunami Alert'), (70, 'QuakeFeed'),
(70, 'Hurricane Tracker'), (70, 'WeatherNow'),
(71, 'RescueNet'), (71, 'AlertMe'), (71, 'HelpLink'),
(71, 'Global Disaster Alert'), (71, 'Shelter Connect'),
(72, 'CrisisConnect'), (72, 'GeoNet'), (73, 'Facebook'),
(73, 'Twitter'), (73, 'Instagram'),
(73, 'YouTube'), (73, 'WhatsApp'),
(74, 'Snapchat'), (74, 'TikTok'), (74, 'Reddit'),
(74, 'Telegram'), (74, 'LinkedIn'),
(75, 'Pinterest'), (75, 'Nextdoor'), (75, 'WeChat'),
(75, 'Signal'), (75, 'VK'),
(76, 'Line'), (76, 'Quora'), (76, 'Clubhouse'),
(76, 'Tumblr'), (76, 'Flickr'),
(77, 'Meetup'), (77, 'Discord'), (77, 'Koo'),
(77, 'Viber'), (77, 'Hike'),
(78, 'Medium'), (78, 'Mix'), (78, 'Periscope'),
(78, 'MeWe'), (78, 'Parler'),
(79, 'Gab'), (79, 'Blogger'), (79, 'Substack'),
(79, 'Hive Social'), (79, 'Diaspora'),

(80, 'Mastodon'), (80, 'Steemit'), (80, 'Rumble'),
 (80, 'Locals'), (80, 'Vero'),
 (81, 'Flipboard'), (81, 'Zello'), (81, 'Citizen'),
 (81, 'Alert Media'), (81, 'Nixle'),
 (82, 'PulsePoint'), (82, 'Earthquake Network'), (82, 'ReliefWeb'),
 (82, 'DisasterAlert'), (82, 'SAFER Together'),
 (83, 'AlertHub'), (83, 'ReadyApp'), (83, 'SkyAlert'),
 (83, 'AlertNet'), (83, 'LiveSafe'),
 (84, 'MyShake'), (84, 'Safety Check by Facebook'), (84, 'Zello Walkie Talkie'),
 (84, 'Eviate'), (84, 'Weather Underground'),
 (85, 'AccuWeather'), (85, 'The Weather Channel'), (85, 'NOAA Weather App');



The screenshot shows a PostgreSQL database interface with the following details:

- Toolbar:** Includes tabs for Dashboard, Properties, SQL, Statistics, and the current connection (crowdsourcedresponse/postgres@PostgreSQL 17*).
- Query Bar:** Shows the query: `SELECT * FROM disaster_socialmedia;`
- Data Output:** A table showing the results of the query. The table has two columns: `disaster_id` and `platformname`.
- Table Data:**

	disaster_id	platformname
1	1	Facebook
2	1	Twitter
3	1	Instagram
4	1	YouTube
5	1	WhatsApp
6	2	Snapchat
7	2	TikTok
8	2	Reddit
9	2	Telegram
10	2	LinkedIn
11	3	Pinterest
12	3	Nextdoor
13	3	WeChat
14	3	Signal
- Message Bar:** Shows a green message: "Successfully run. Total query runtime: 506 msec. 411 rows affected."
- Bottom Status:** Shows "Total rows: 411 of 411" and "Query complete 00:00:00.506 Ln 1, Col 35".

Tuple count – 411

22. Communities_Volunteers

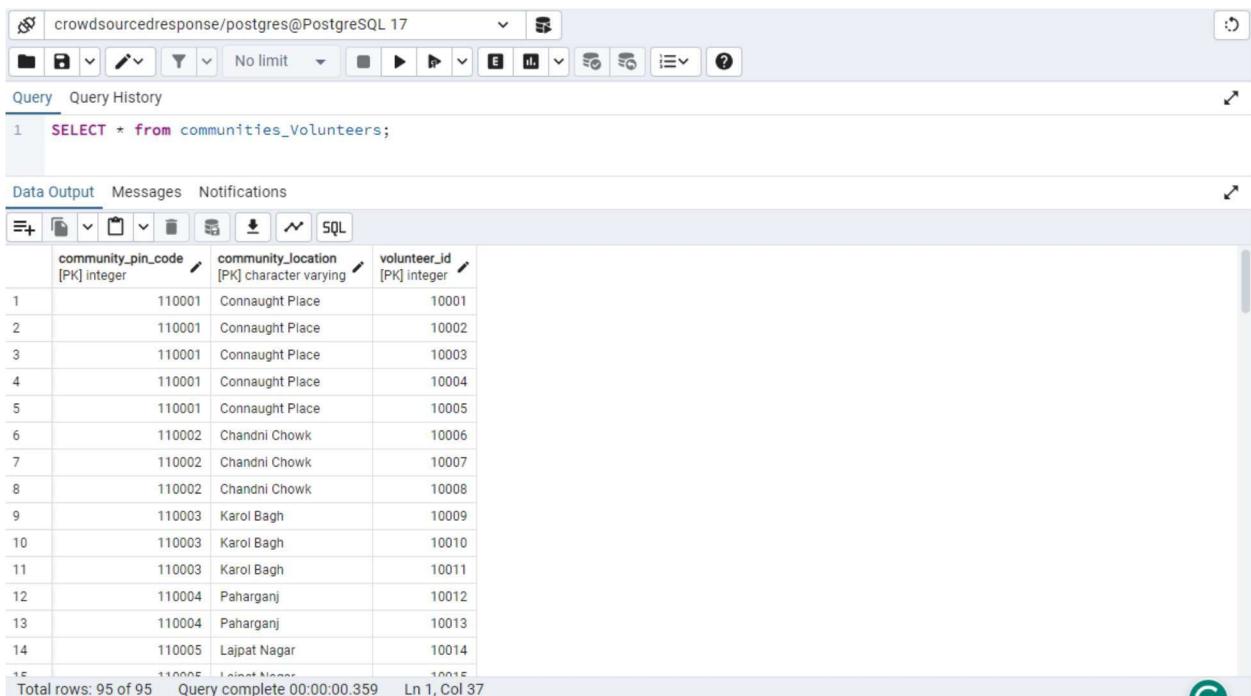
```
INSERT INTO Communities_Volunteers (Community_Pin_Code,  
Community_Location, Volunteer_ID) VALUES  
(110001, 'Connaught Place', 10001),  
(110001, 'Connaught Place', 10002),  
(110001, 'Connaught Place', 10003),  
(110001, 'Connaught Place', 10004),  
(110001, 'Connaught Place', 10005),  
(110002, 'Chandni Chowk', 10006),  
(110002, 'Chandni Chowk', 10007),  
(110002, 'Chandni Chowk', 10008),  
(110003, 'Karol Bagh', 10009),  
(110003, 'Karol Bagh', 10010),  
(110003, 'Karol Bagh', 10011),  
(110004, 'Paharganj', 10012),  
(110004, 'Paharganj', 10013),  
(110005, 'Lajpat Nagar', 10014),  
(110005, 'Lajpat Nagar', 10015),  
(110005, 'Lajpat Nagar', 10016),  
(110006, 'Rohini', 10017),
```

(110006, 'Rohini', 10018),
(110007, 'Dwarka', 10019),
(110007, 'Dwarka', 10020),
(110008, 'Janakpuri', 10021),
(110009, 'Preet Vihar', 10022),
(110010, 'Kalkaji', 10023),
(400001, 'Marine Lines', 10024),
(400002, 'Dadar', 10025),
(400003, 'Bandra', 10026),
(400004, 'Andheri', 10027),
(400005, 'Thane', 10028),
(400006, 'Navi Mumbai', 10029),
(400007, 'Borivali', 10030),
(400008, 'Malad', 10031),
(400009, 'Versova', 10032),
(400010, 'Vashi', 10033),
(500001, 'Banjara Hills', 10034),
(500002, 'Jubilee Hills', 10035),
(500003, 'Secunderabad', 10036),
(500004, 'Gachibowli', 10037),
(500005, 'Hitech City', 10038),
(500006, 'Madhapur', 10039),

- (500007, 'LB Nagar', 10040),
(500008, 'Dilsukhnagar', 10041),
(500009, 'Kukatpally', 10042),
(500010, 'Mehdipatnam', 10043),
(500011, 'KPHB Colony', 10044),
(600001, 'T. Nagar', 10045),
(600002, 'Mylapore', 10046),
(600003, 'Nungambakkam', 10047),
(600004, 'Anna Nagar', 10048),
(600005, 'Velachery', 10049),
(600006, 'Kotturpuram', 10050),
(600007, 'Adyar', 10051),
(600008, 'Besant Nagar', 10052),
(600009, 'Kottivakkam', 10053),
(600010, 'Tambaram', 10054),
(600011, 'Perungudi', 10055),
(700001, 'Salt Lake City', 10056),
(700002, 'Garia', 10057),
(700003, 'Behala', 10058),
(700004, 'Dum Dum', 10059),
(700005, 'New Town', 10060),
(700006, 'Howrah', 10061),

(700007, 'Bidhannagar', 10062),
(700008, 'Kolkata', 10063),
(700009, 'Baranagar', 10064),
(700010, 'Tollygunge', 10065),
(800001, 'Shahjahanpur', 10066),
(800002, 'Bareilly', 10067),
(800003, 'Moradabad', 10068),
(800004, 'Aligarh', 10069),
(800005, 'Agra', 10070),
(800006, 'Lucknow', 10071),
(800007, 'Kanpur', 10072),
(800008, 'Noida', 10073),
(800009, 'Ghaziabad', 10074),
(800010, 'Meerut', 10075),
(900001, 'Kathmandu', 10076),
(900002, 'Biratnagar', 10077),
(900003, 'Lalitpur', 10078),
(900004, 'Bhaktapur', 10079),
(900005, 'Pokhara', 10080),
(900006, 'Nepalgunj', 10081),
(900007, 'Janakpur', 10082),
(900008, 'Birgunj', 10083),

(900009, 'Bhairahawa', 10084),
 (900010, 'Itahari', 10085),
 (950001, 'Dhaka', 10086),
 (950002, 'Chittagong', 10087),
 (950003, 'Khulna', 10088),
 (950004, 'Rajshahi', 10089),
 (950005, 'Sylhet', 10090),
 (110011, 'Saket', 10001),
 (110011, 'Saket', 10006),
 (110011, 'Saket', 10002),
 (110011, 'Saket', 10009),
 (110011, 'Saket', 10003);



The screenshot shows a PostgreSQL client interface with the following details:

- Connection:** crowdsourcedresponse/postgres@PostgreSQL 17
- Toolbar:** Includes icons for file operations, search, and various database functions.
- Query Bar:** Shows "Query History" and a dropdown for "No limit".
- SQL Editor:** Contains the query: `SELECT * from communities_Volunteers;`
- Data Output:** A table showing the results of the query. The table has three columns: `community_pin_code`, `community_location`, and `volunteer_id`. The data consists of 95 rows, with the first few rows being:

	community_pin_code [PK] integer	community_location [PK] character varying	volunteer_id [PK] integer
1	110001	Connaught Place	10001
2	110001	Connaught Place	10002
3	110001	Connaught Place	10003
4	110001	Connaught Place	10004
5	110001	Connaught Place	10005
6	110002	Chandni Chowk	10006
7	110002	Chandni Chowk	10007
8	110002	Chandni Chowk	10008
9	110003	Karol Bagh	10009
10	110003	Karol Bagh	10010
11	110003	Karol Bagh	10011
12	110004	Paharganj	10012
13	110004	Paharganj	10013
14	110005	Lajpat Nagar	10014

- Bottom Status:** Total rows: 95 of 95 | Query complete 00:00:00.359 | Ln 1, Col 37

Tuple count – 95

23. Community NGO

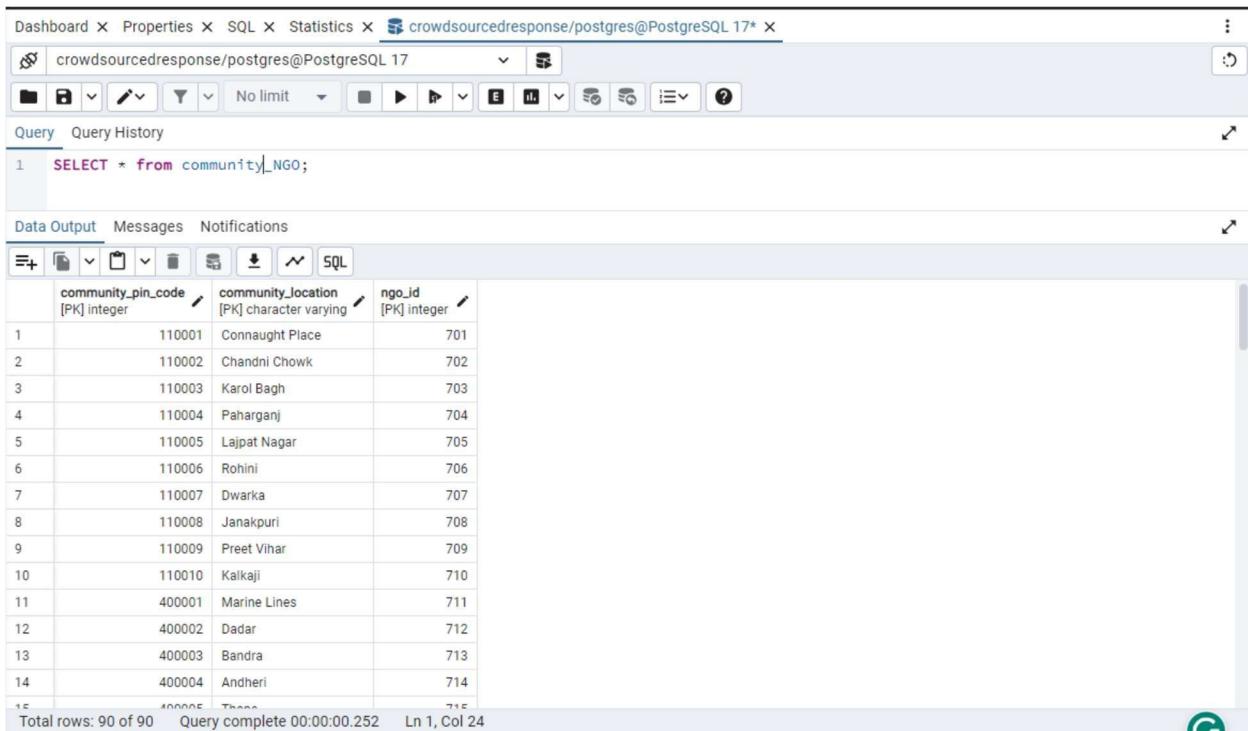
```
INSERT INTO Community_NGO (Community_Pin_Code,  
Community_Location, NGO_ID) VALUES  
(110001, 'Connaught Place', 701),  
(110002, 'Chandni Chowk', 702),  
(110003, 'Karol Bagh', 703),  
(110004, 'Paharganj', 704),  
(110005, 'Lajpat Nagar', 705),  
(110006, 'Rohini', 706),  
(110007, 'Dwarka', 707),  
(110008, 'Janakpuri', 708),  
(110009, 'Preet Vihar', 709),  
(110010, 'Kalkaji', 710),  
(400001, 'Marine Lines', 711),  
(400002, 'Dadar', 712),  
(400003, 'Bandra', 713),  
(400004, 'Andheri', 714),  
(400005, 'Thane', 715),  
(400006, 'Navi Mumbai', 716),  
(400007, 'Borivali', 717),  
(400008, 'Malad', 718),
```

- (400009, 'Versova', 719),
(400010, 'Vashi', 720),
(500001, 'Banjara Hills', 721),
(500002, 'Jubilee Hills', 722),
(500003, 'Secunderabad', 723),
(500004, 'Gachibowli', 724),
(500005, 'Hitech City', 725),
(500006, 'Madhapur', 726),
(500007, 'LB Nagar', 727),
(500008, 'Dilsukhnagar', 728),
(500009, 'Kukatpally', 729),
(500010, 'Mehdipatnam', 730),
(600001, 'T. Nagar', 731),
(600002, 'Mylapore', 732),
(600003, 'Nungambakkam', 733),
(600004, 'Anna Nagar', 734),
(600005, 'Velachery', 735),
(600006, 'Kotturpuram', 736),
(600007, 'Adyar', 737),
(600008, 'Besant Nagar', 738),
(600009, 'Kottivakkam', 739),
(600010, 'Tambaram', 740),

- (700001, 'Salt Lake City', 741),
- (700002, 'Garia', 742),
- (700003, 'Behala', 743),
- (700004, 'Dum Dum', 744),
- (700005, 'New Town', 745),
- (700006, 'Howrah', 746),
- (700007, 'Bidhannagar', 747),
- (700008, 'Kolkata', 748),
- (700009, 'Baranagar', 749),
- (700010, 'Tollygunge', 750),
- (800001, 'Shahjahanpur', 751),
- (800002, 'Bareilly', 752),
- (800003, 'Moradabad', 753),
- (800004, 'Aligarh', 754),
- (800005, 'Agra', 755),
- (800006, 'Lucknow', 756),
- (800007, 'Kanpur', 757),
- (800008, 'Noida', 758),
- (800009, 'Ghaziabad', 759),
- (800010, 'Meerut', 760),
- (900001, 'Kathmandu', 761),
- (900002, 'Biratnagar', 762),

- (900003, 'Lalitpur', 763),
- (900004, 'Bhaktapur', 764),
- (900005, 'Pokhara', 765),
- (900006, 'Nepalgunj', 766),
- (900007, 'Janakpur', 767),
- (900008, 'Birgunj', 768),
- (900009, 'Bhairahawa', 769),
- (900010, 'Itahari', 770),
- (950001, 'Dhaka', 771),
- (950002, 'Chittagong', 772),
- (950003, 'Khulna', 773),
- (950004, 'Rajshahi', 774),
- (950005, 'Sylhet', 775),
- (950006, 'Barisal', 776),
- (950007, 'Comilla', 777),
- (950008, 'Narayanganj', 778),
- (950009, 'Bogra', 779),
- (950010, 'Tangail', 780),
- (110011, 'Saket', 781),
- (500011, 'KPHB Colony', 782),
- (600011, 'Perungudi', 783),
- (110001, 'Connaught Place', 784),

(400007, 'Borivali', 785),
 (500002, 'Jubilee Hills', 786),
 (600004, 'Anna Nagar', 787),
 (700005, 'New Town', 788),
 (800006, 'Lucknow', 789),
 (900001, 'Kathmandu', 790);



The screenshot shows the pgAdmin 4 interface. The title bar says "Dashboard × Properties × SQL × Statistics × crowdsourcedresponse/postgres@PostgreSQL 17*". The main area has a toolbar with various icons. Below the toolbar, it says "Query History" and shows a single query:

```
1 SELECT * from community_NGO;
```

Underneath the query, there are tabs for "Data Output", "Messages", and "Notifications". The "Data Output" tab is selected and displays a table with the following data:

	community_pin_code [PK] integer	community_location [PK] character varying	ngo_Id [PK] integer
1	110001	Connaught Place	701
2	110002	Chandni Chowk	702
3	110003	Karol Bagh	703
4	110004	Paharganj	704
5	110005	Lajpat Nagar	705
6	110006	Rohini	706
7	110007	Dwarka	707
8	110008	Janakpuri	708
9	110009	Preet Vihar	709
10	110010	Kalkaji	710
11	400001	Marine Lines	711
12	400002	Dadar	712
13	400003	Bandra	713
14	400004	Andheri	714

At the bottom of the results pane, it says "Total rows: 90 of 90" and "Query complete 00:00:00.252 Ln 1, Col 24".

Tuple count – 90

24. Volunteer Victim

```
INSERT INTO Volunteer_Victim (Volunteer_ID, Victim_ID) VALUES
(10001, 701),(10002, 702),(10003, 703),
```

(10004, 704),(10005, 705),(10006, 706),
(10007, 707),(10008, 708),(10009, 709),
(10010, 710),(10011, 711),(10012, 712),
(10013, 713),(10014, 714),(10015, 715),
(10016, 716),(10017, 717),(10018, 718),
(10019, 719),(10020, 720),(10021, 721),
(10022, 722),(10023, 723),(10024, 724),
(10025, 725),(10026, 726),(10027, 727),
(10028, 728),(10029, 729),(10030, 730),
(10031, 731),(10032, 732),(10033, 733),
(10034, 734),(10035, 735),(10036, 736),
(10037, 737),(10038, 738),(10039, 739),
(10040, 740),(10041, 741),(10042, 742),
(10043, 743),(10044, 744),(10045, 745),
(10046, 746),(10047, 747),(10048, 748),
(10049, 749),(10050, 750),(10051, 751),
(10052, 752),(10053, 753),(10054, 754),
(10055, 755),(10056, 756),(10057, 757),
(10058, 758),(10059, 759),(10060, 760),
(10061, 761),(10062, 762),(10063, 763),
(10064, 764),(10065, 765),(10066, 766),
(10067, 767),(10068, 768),(10069, 769),

(10070, 770),(10071, 771),(10072, 772),
(10073, 773),(10074, 774),(10075, 775),
(10076, 776),(10077, 777),(10078, 778),
(10079, 779),(10080, 780),(10081, 781),
(10082, 782),(10083, 783),(10084, 784),
(10085, 785),(10086, 786),(10087, 787),
(10088, 788),(10089, 789),(10090, 790),
(10001, 791),(10002, 792),(10003, 793),
(10004, 794),(10005, 795),(10006, 796),
(10007, 797),(10008, 798),(10009, 799),
(10010, 800),(10011, 801),(10012, 802),
(10013, 803),(10014, 804),(10015, 805),
(10016, 806),(10017, 807),(10018, 808),
(10019, 809),(10020, 810),(10021, 811),
(10022, 812),(10023, 813),(10024, 814),
(10025, 815),(10026, 816),(10027, 817),
(10028, 818),(10029, 819),(10030, 820);

```

1  SELECT * from volunteer_victim;

Data Output Messages Notifications
SQL

volunteer_id [PK] integer victim_id [PK] integer
1 10001 701
2 10002 702
3 10003 703
4 10004 704
5 10005 705
6 10006 706
7 10007 707
8 10008 708
9 10009 709
10 10010 710
11 10011 711
12 10012 712
13 10013 713
14 10014 714
Total rows: 120 of 120 Query complete 00:00:00.345 Ln 1, Col 24

```

Successfully run. Total query runtime: 345 msec. 120 rows affected.

Tuple count – 120

25. Agencies NGO

```

INSERT INTO Agencies_NGOs (Agency_ID, NGO_ID) VALUES
(1001, 701), (1002, 702), (1003, 703), (1001, 704),
(1004, 705), (1005, 706), (1001, 707), (1006, 708),
(1007, 709), (1008, 710), (1004, 711), (1002, 712),
(1009, 713), (1010, 714), (1011, 715), (1012, 716),
(1013, 717), (1014, 718), (1015, 719), (1016, 720),
(1017, 721), (1018, 722), (1019, 723), (1020, 724),
(1021, 725), (1022, 726), (1023, 727), (1024, 728),
(1025, 729), (1026, 730), (1027, 731), (1028, 732),
(1029, 733), (1030, 734), (1031, 735), (1032, 736),

```

(1033, 737), (1034, 738), (1035, 739), (1036, 740),
(1037, 741), (1038, 742), (1039, 743), (1040, 744),
(1041, 745), (1042, 746), (1043, 747), (1044, 748),
(1045, 749), (1046, 750), (1047, 751), (1048, 752),
(1049, 753), (1050, 754), (1051, 755), (1052, 756),
(1053, 757), (1054, 758), (1055, 759), (1056, 760),
(1057, 761), (1058, 762), (1059, 763), (1060, 764),
(1061, 765), (1062, 766), (1063, 767), (1064, 768),
(1065, 769), (1066, 770), (1067, 771), (1068, 772),
(1069, 773), (1070, 774), (1071, 775), (1072, 776),
(1073, 777), (1074, 778), (1075, 779), (1076, 780),
(1077, 781), (1078, 782), (1079, 783), (1080, 784),
(1081, 785), (1082, 786), (1001, 787), (1002, 788),
(1003, 789), (1004, 790), (1005, 701), (1006, 702),
(1007, 703), (1008, 704), (1009, 705), (1010, 706),
(1011, 707), (1012, 708), (1013, 709), (1014, 710),
(1015, 711), (1016, 712), (1017, 713), (1018, 714),
(1019, 715), (1020, 716), (1021, 717), (1022, 718),
(1023, 719), (1024, 720), (1025, 721), (1026, 722),
(1027, 723), (1028, 724), (1029, 725), (1030, 726);

The screenshot shows a pgAdmin 4 interface with a query editor and a results table. The query is:

```
1  SELECT * from agencies_Ngos;
```

The results table has two columns: agency_id and ngo_id. The data is as follows:

	agency_id	ngo_id
1	1001	701
2	1002	702
3	1003	703
4	1001	704
5	1004	705
6	1005	706
7	1001	707
8	1006	708
9	1007	709
10	1008	710
11	1004	711
12	1002	712
13	1009	713
14	1010	714

Total rows: 116 of 116 Query complete 00:00:00.323 Ln 1, Col 28

Tuple count – 116

26. emergencyteams_resources

INSERT INTO EmergencyTeams_Resources (Team_ID, Resource_ID)
VALUES

(9001, 501),

(9001, 502),

(9001, 503),

(9001, 504),

(9002, 505),

(9002, 506),

(9002, 507),

(9002, 508),

(9003, 509),

(9003, 510),

(9003, 511),

(9004, 512),

(9005, 516),

(9006, 518),

(9006, 519),

(9007, 520),

(9007, 521),

(9007, 522),

(9008, 523),

(9008, 524),

(9009, 525),

(9009, 526),

(9010, 527),

(9011, 529),

(9011, 530),

(9012, 532),

(9013, 533),

(9013, 534),

(9014, 535),

(9015, 538),

(9016, 539),

(9016, 540),

(9017, 541),

(9018, 544),

(9019, 546),

(9020, 548),

(9021, 549),

(9022, 552),

(9023, 553),

(9024, 555),

(9024, 556),

(9025, 557),

(9026, 559),

(9026, 560),

(9027, 561),

(9027, 562),

(9028, 563),

(9028, 564),

(9029, 565),

(9029, 566),

(9030, 568),

(9031, 569),

(9031, 570),

(9032, 571),

(9032, 572),

(9033, 573),

(9033, 574),

(9034, 575),

(9034, 576),

(9035, 577),

(9035, 578),

(9036, 579),

(9036, 580),

(9037, 581),

(9037, 582),

(9038, 583),

(9038, 584),

(9039, 586),

(9040, 587),

(9040, 588),

(9041, 589),

(9041, 590),

(9042, 501),

(9043, 502),

(9044, 503),

(9045, 504),

(9046, 505),

(9047, 506),

(9048, 507),

(9049, 508),

(9050, 509),

(9051, 510),

(9052, 511),

(9053, 512),

(9054, 513),

(9055, 514),

(9056, 515),

(9057, 516),

(9058, 517),

(9059, 518),

(9060, 519),

(9061, 520),

(9062, 521),

(9063, 522),

(9064, 523),

(9065, 524),

(9066, 525),

(9067, 526),

(9068, 527),

(9069, 528),

(9070, 529),

(9071, 530),

(9072, 531),

(9073, 532),

(9074, 533),

(9075, 534),

(9076, 535),

(9077, 536),

(9078, 537),

(9079, 538),

(9080, 539),

(9081, 540),

(9082, 541),

(9083, 542),

(9084, 543),

(9085, 544),

(9086, 545),

(9087, 546);

The screenshot shows a PostgreSQL client interface with the following details:

- Connection:** crowdsourcedresponse/postgres@PostgreSQL 17
- Query Bar:** SELECT * from emergencyteams_resources;
- Data Output:** Shows a table with two columns: team_id and resource_id.
- Table Data:**

	team_id [PK] integer	resource_id [PK] integer
1	9001	501
2	9001	502
3	9001	503
4	9001	504
5	9002	505
6	9002	506
7	9002	507
8	9002	508
9	9003	509
10	9003	510
11	9003	511
12	9004	512
13	9005	516
14	9006	518
15	9006	519
- Message Bar:** Successfully run. Total query runtime: 290 msec. 118 rows affected.
- Bottom Status:** Total rows: 118 of 118 Query complete 00:00:00.290 Ln 1, Col 39

Tuple count – 118

27. Communities Agency

```
INSERT INTO Community_Agency (Community_Pin_Code,
Community_Location, Agency_ID) VALUES
(110001, 'Connaught Place', 1001),
(110001, 'Connaught Place', 1002),
(110002, 'Chandni Chowk', 1003),
(110002, 'Chandni Chowk', 1004),
(110003, 'Karol Bagh', 1005),
(110003, 'Karol Bagh', 1006),
(110004, 'Paharganj', 1007),
(110004, 'Paharganj', 1008),
```

- (110005, 'Lajpat Nagar', 1009),
- (110005, 'Lajpat Nagar', 1010),
- (110006, 'Rohini', 1011),
- (110006, 'Rohini', 1012),
- (110007, 'Dwarka', 1013),
- (110007, 'Dwarka', 1014),
- (110008, 'Janakpuri', 1015),
- (110008, 'Janakpuri', 1016),
- (110009, 'Preet Vihar', 1017),
- (110009, 'Preet Vihar', 1018),
- (110010, 'Kalkaji', 1019),
- (110010, 'Kalkaji', 1020),
- (400001, 'Marine Lines', 1021),
- (400001, 'Marine Lines', 1022),
- (400002, 'Dadar', 1023),
- (400002, 'Dadar', 1024),
- (400003, 'Bandra', 1025),
- (400003, 'Bandra', 1026),
- (400004, 'Andheri', 1027),
- (400004, 'Andheri', 1028),
- (400005, 'Thane', 1029),
- (400005, 'Thane', 1030),

(400006, 'Navi Mumbai', 1031),
(400006, 'Navi Mumbai', 1032),
(400007, 'Borivali', 1033),
(400007, 'Borivali', 1034),
(400008, 'Malad', 1035),
(400008, 'Malad', 1036),
(400009, 'Versova', 1037),
(400009, 'Versova', 1038),
(400010, 'Vashi', 1039),
(400010, 'Vashi', 1040),
(500001, 'Banjara Hills', 1041),
(500001, 'Banjara Hills', 1042),
(500002, 'Jubilee Hills', 1043),
(500002, 'Jubilee Hills', 1044),
(500003, 'Secunderabad', 1045),
(500003, 'Secunderabad', 1046),
(500004, 'Gachibowli', 1047),
(500004, 'Gachibowli', 1048),
(500005, 'Hitech City', 1049),
(500005, 'Hitech City', 1050),
(500006, 'Madhapur', 1051),
(500006, 'Madhapur', 1052),

- (500007, 'LB Nagar', 1053),
- (500007, 'LB Nagar', 1054),
- (500008, 'Dilsukhnagar', 1056),
- (500009, 'Kukatpally', 1057),
- (500009, 'Kukatpally', 1058),
- (500010, 'Mehdipatnam', 1059),
- (500010, 'Mehdipatnam', 1060),
- (600001, 'T. Nagar', 1061),
- (600001, 'T. Nagar', 1062),
- (600002, 'Mylapore', 1063),
- (600002, 'Mylapore', 1064),
- (600003, 'Nungambakkam', 1065),
- (600003, 'Nungambakkam', 1066),
- (600004, 'Anna Nagar', 1067),
- (600004, 'Anna Nagar', 1068),
- (600005, 'Velachery', 1069),
- (600005, 'Velachery', 1070),
- (600006, 'Kotturpuram', 1071),
- (600006, 'Kotturpuram', 1072),
- (600007, 'Adyar', 1073),
- (600007, 'Adyar', 1074),
- (600008, 'Besant Nagar', 1075),

- (600008, 'Besant Nagar', 1076),
- (600009, 'Kottivakkam', 1077),
- (600009, 'Kottivakkam', 1078),
- (600010, 'Tambaram', 1079),
- (600010, 'Tambaram', 1080),
- (700001, 'Salt Lake City', 1081),
- (700001, 'Salt Lake City', 1001),
- (700002, 'Garia', 1002),
- (700002, 'Garia', 1003),
- (700003, 'Behala', 1004),
- (700003, 'Behala', 1005),
- (700004, 'Dum Dum', 1006),
- (700004, 'Dum Dum', 1007),
- (700005, 'New Town', 1008),
- (700005, 'New Town', 1009),
- (700006, 'Howrah', 1010),
- (700006, 'Howrah', 1011),
- (700007, 'Bidhannagar', 1012),
- (700007, 'Bidhannagar', 1013),
- (700008, 'Kolkata', 1014),
- (700008, 'Kolkata', 1015),
- (700009, 'Baranagar', 1016),

- (700009, 'Baranagar', 1017),
- (700010, 'Tollygunge', 1018),
- (700010, 'Tollygunge', 1019),
- (800001, 'Shahjahanpur', 1020),
- (800001, 'Shahjahanpur', 1021),
- (800002, 'Bareilly', 1022),
- (800002, 'Bareilly', 1023),
- (800003, 'Moradabad', 1024),
- (800003, 'Moradabad', 1025),
- (800004, 'Aligarh', 1026),
- (800004, 'Aligarh', 1027),
- (800005, 'Agra', 1028),
- (800005, 'Agra', 1029),
- (800006, 'Lucknow', 1030),
- (800006, 'Lucknow', 1031),
- (800007, 'Kanpur', 1032),
- (800007, 'Kanpur', 1033),
- (800008, 'Noida', 1034),
- (800008, 'Noida', 1035),
- (800009, 'Ghaziabad', 1036),
- (800009, 'Ghaziabad', 1037),
- (800010, 'Meerut', 1038),

- (800010, 'Meerut', 1039),
- (900001, 'Kathmandu', 1040),
- (900001, 'Kathmandu', 1041),
- (900002, 'Biratnagar', 1042),
- (900002, 'Biratnagar', 1043),
- (900003, 'Lalitpur', 1044),
- (900003, 'Lalitpur', 1045),
- (900004, 'Bhaktapur', 1046),
- (900004, 'Bhaktapur', 1047),
- (900005, 'Pokhara', 1048),
- (900005, 'Pokhara', 1049),
- (900006, 'Nepalgunj', 1050),
- (900006, 'Nepalgunj', 1051),
- (900007, 'Janakpur', 1052),
- (900007, 'Janakpur', 1053),
- (900008, 'Birgunj', 1054),
- (900008, 'Birgunj', 1055),
- (900009, 'Bhairahawa', 1056),
- (900009, 'Bhairahawa', 1057),
- (900010, 'Itahari', 1058),
- (900010, 'Itahari', 1059),
- (950001, 'Dhaka', 1060),

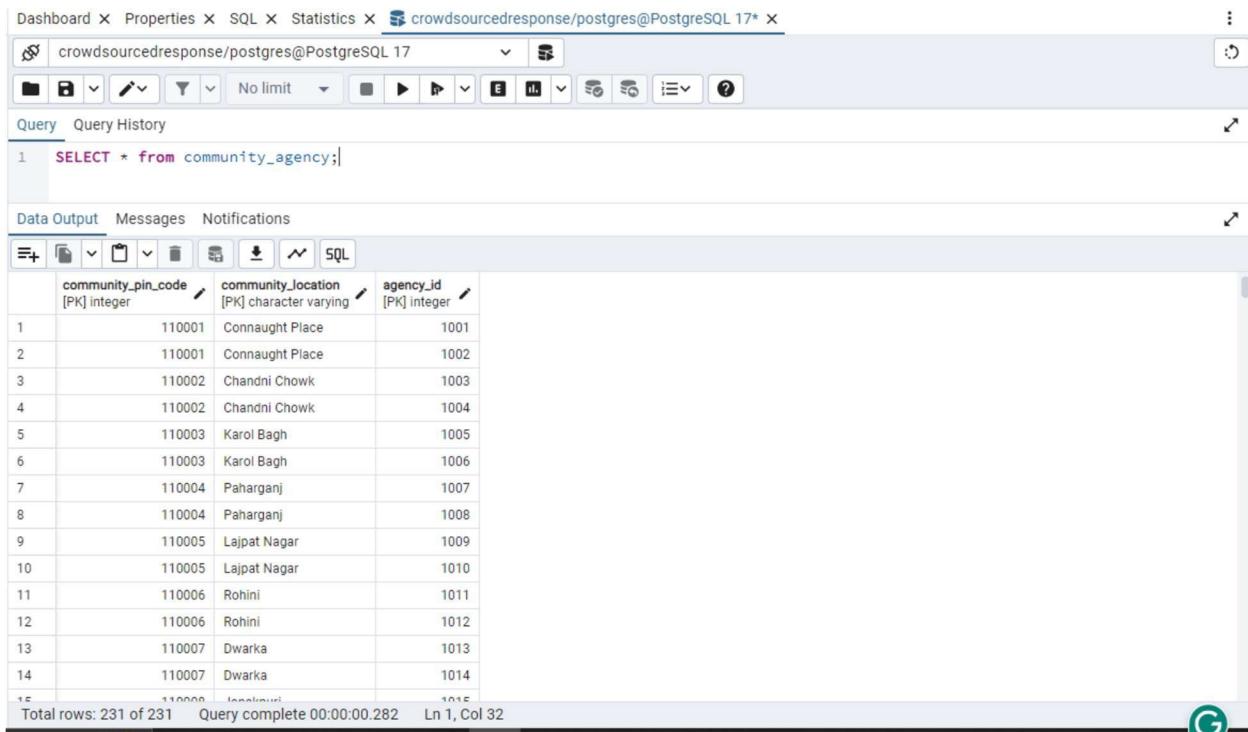
- (950001, 'Dhaka', 1061),
(950002, 'Chittagong', 1062),
(950002, 'Chittagong', 1063),
(950003, 'Khulna', 1064),
(950003, 'Khulna', 1065),
(950004, 'Rajshahi', 1066),
(950004, 'Rajshahi', 1067),
(950005, 'Sylhet', 1068),
(950005, 'Sylhet', 1006),
(950006, 'Barisal', 1070),
(950006, 'Barisal', 1071),
(950007, 'Comilla', 1072),
(950007, 'Comilla', 1073),
(950008, 'Narayanganj', 1074),
(950008, 'Narayanganj', 1075),
(950009, 'Bogra', 1076),
(950009, 'Bogra', 1077),
(950010, 'Tangail', 1078),
(950010, 'Tangail', 1079),
(110011, 'Saket', 1080),
(110011, 'Saket', 1081),
(500011, 'KPHB Colony', 1082),

- (500011, 'KPHB Colony', 1001),
(600011, 'Perungudi', 1002),
(600011, 'Perungudi', 1003),
(110001, 'Connaught Place', 1004),
(110002, 'Chandni Chowk', 1005),
(400001, 'Marine Lines', 1006),
(400002, 'Dadar', 1007),
(500001, 'Banjara Hills', 1008),
(600002, 'Mylapore', 1009),
(700001, 'Salt Lake City', 1010),
(800001, 'Shahjahanpur', 1011),
(900001, 'Kathmandu', 1012),
(950001, 'Dhaka', 1013),
(110010, 'Kalkaji', 1014),
(400005, 'Thane', 1015),
(500003, 'Secunderabad', 1016),
(600003, 'Nungambakkam', 1017),
(700003, 'Behala', 1018),
(800005, 'Agra', 1019),
(900003, 'Lalitpur', 1020),
(950002, 'Chittagong', 1021),
(110004, 'Paharganj', 1022),

(500002, 'Jubilee Hills', 1023),
(400008, 'Malad', 1024),
(700002, 'Garia', 1025),
(800003, 'Moradabad', 1026),
(600008, 'Besant Nagar', 1027),
(950004, 'Rajshahi', 1028),
(500004, 'Gachibowli', 1029),
(110009, 'Preet Vihar', 1030),
(400007, 'Borivali', 1031),
(600004, 'Anna Nagar', 1032),
(900002, 'Biratnagar', 1033),
(950005, 'Sylhet', 1035),
(700005, 'New Town', 1036),
(500005, 'Hitech City', 1037),
(400006, 'Navi Mumbai', 1038),
(600006, 'Kotturpuram', 1039),
(900006, 'Nepalgunj', 1040),
(800009, 'Ghaziabad', 1041),
(700008, 'Kolkata', 1042),
(110005, 'Lajpat Nagar', 1043),
(600007, 'Adyar', 1044),
(500007, 'LB Nagar', 1045),

(400010, 'Vashi', 1046),
(700006, 'Howrah', 1047),
(950008, 'Narayanganj', 1048),
(400004, 'Andheri', 1049),
(110006, 'Rohini', 1050),
(600009, 'Kottivakkam', 1051),
(950003, 'Khulna', 1052),
(700007, 'Bidhannagar', 1053),
(900007, 'Janakpur', 1054),
(500008, 'Dilsukhnagar', 1055),
(800007, 'Kanpur', 1056),
(400001, 'Marine Lines', 1057),
(600002, 'Mylapore', 1059),
(110007, 'Dwarka', 1060),
(800006, 'Lucknow', 1061),
(700009, 'Baranagar', 1062),
(400003, 'Bandra', 1063),
(500010, 'Mehdipatnam', 1064),
(950009, 'Bogra', 1065),
(600011, 'Perungudi', 1066),
(110008, 'Janakpuri', 1067),
(700004, 'Dum Dum', 1068),

(950001, 'Dhaka', 1069),
 (800010, 'Meerut', 1070),
 (110010, 'Kalkaji', 1071);



The screenshot shows a PostgreSQL database interface with the following details:

- Toolbar:** Includes icons for Dashboard, Properties, SQL, Statistics, and a connection named "crowdsourcedresponse/postgres@PostgreSQL 17*".
- Query Bar:** Shows the query "SELECT * from community_agency;".
- Data Output:** Displays the results of the query in a table format.
- Table Headers:** community_pin_code [PK] integer, community_location [PK] character varying, agency_id [PK] integer.
- Table Data:** 231 rows of data, listing various locations and their corresponding agency IDs.
- Bottom Status:** Total rows: 231 of 231, Query complete 00:00:00.282, Ln 1, Col 32.

Tuple count – 231

28. Response_Victims

INSERT INTO Response_Victims (Victim_ID, Response_ID) VALUES
 (701, 234), (702, 234), (703, 234),
 (704, 235), (705, 235), (706, 235),
 (707, 236), (708, 236), (709, 237),
 (710, 238), (711, 238), (712, 238),
 (713, 239), (714, 239), (715, 240),

(716, 240), (717, 241), (718, 241),
(719, 242), (720, 242), (721, 243),
(722, 244), (723, 244), (724, 245),
(725, 246), (726, 246), (727, 247),
(728, 247), (729, 248), (730, 248),
(731, 249), (732, 249), (733, 250),
(734, 251), (735, 251), (736, 252),
(737, 252), (738, 253), (739, 254),
(740, 254), (741, 255), (742, 255),
(743, 256), (744, 257), (745, 257),
(746, 258), (747, 258), (748, 259),
(749, 260), (750, 261), (751, 261),
(752, 262), (753, 263), (754, 263),
(755, 264), (756, 265), (757, 265),
(758, 266), (759, 267), (760, 267),
(761, 268), (762, 269), (763, 269),
(764, 270), (765, 271), (766, 271),
(767, 272), (768, 273), (769, 274),
(770, 275), (771, 275), (772, 276),
(773, 277), (774, 278), (775, 279),
(776, 279), (777, 280), (778, 281),
(779, 282), (780, 283), (781, 283),

(782, 284), (783, 285), (784, 286),
(785, 287), (786, 288), (787, 289),
(788, 290), (789, 291), (790, 292),
(791, 293), (792, 294), (793, 295),
(794, 296), (795, 297), (796, 298),
(797, 299), (798, 300), (799, 301),
(800, 302), (801, 303), (802, 304),
(803, 305), (804, 306), (805, 307),
(806, 308), (807, 309), (808, 310),
(809, 311), (810, 312), (811, 313),
(812, 314), (813, 315), (814, 316),
(815, 317), (816, 318), (817, 319),
(818, 320), (819, 320), (820, 320),
(725, 302), (726, 298), (727, 297),
(758, 256), (759, 247), (760, 313);

Dashboard × Properties × SQL × Statistics ×  crowdsourcedresponse/postgres@PostgreSQL 17*

crowdsourcedresponse/postgres@PostgreSQL 17 No limit

Query History

1 `SELECT * FROM response_victims;`

Data Output Messages Notifications

	victim_id [PK] integer	response_id [PK] integer
1	701	234
2	702	234
3	703	234
4	704	235
5	705	235
6	706	235
7	707	236
8	708	236
9	709	237
10	710	238
11	711	238
12	712	238
13	713	239
14	714	239

Total rows: 126 of 126 Query complete 00:00:00.361 Ln 1, Col 24

Tuple count - 126

3. SQL QUERIES

1. List all communities with population greater than 1000.

```
SELECT * FROM Communities WHERE Population > 1000;
```

The screenshot shows a SQL query interface with the following details:

- Query History:** SELECT * FROM Communities WHERE Population > 1000;
- Data Output:** A table showing the results of the query. The columns are pincode, location, and population. The data is as follows:

	pincode [PK] integer	location [PK] character varying (50)	population integer
1	110007	Dwarka	1500
2	400004	Andheri	2500
3	400005	Thane	5000
4	400006	Navi Mumbai	10000
5	400007	Borivali	5000
6	400008	Malad	6000
7	400009	Versova	2500
8	400010	Vashi	3000
9	500003	Secunderabad	1500

Total rows: 38 of 38 Query complete 00:00:00.115 Ln 1, Col 50

2. Find resources available in 'Mumbai'.

```
SELECT * FROM Resources WHERE Location = 'Mumbai';
```

```

1 SELECT * FROM Resources WHERE Location = 'Mumbai';
2

```

Data Output Messages Notifications ↗

SQL

	resource_id [PK] integer	type character varying	availability boolean	location character varying	source character varying
1	501	Water Tanker	true	Mumbai	BMC

Total rows: 1 of 1 Query complete 00:00:00.143 Ln 2, Col 1

3. List all disaster records since 2015.

SELECT * FROM Disaster WHERE Date >= '2015-01-01';

```

1 SELECT * FROM Disaster WHERE Date >= '2015-01-01';
2
3
4

```

Data Output Messages Notifications ↗

SQL

	disaster_id [PK] integer	date date	location character varying (50)	type character varying (50)	severity integer
1	13	2015-01-05	Kolkata	Cyclone	4
2	14	2016-02-10	Bhubaneswar	Flood	3
3	15	2017-03-01	Indore	Heatwave	2
4	16	2018-04-15	Gurgaon	Earthquake	4
5	17	2019-05-05	Nashik	Flood	5
6	18	2020-06-01	Surat	Flood	1
7	19	2021-07-10	Lucknow	Cyclone	3
8	31	2015-07-25	Chandigarh	Cyclone	4
9	32	2016-08-01	Vadodara	Earthquake	5

Total rows: 37 of 37 Query complete 00:00:00.074 Ln 2, Col 1

4. Display all available resources.

SELECT * FROM Resources WHERE Availability = TRUE;

```
1  SELECT * FROM Resources WHERE Availability = TRUE;
2
3
4
```

Data Output Messages Notifications

SQL

	resource_id [PK] integer	type character varying	availability boolean	location character varying	source character varying
1	501	Water Tanker	true	Mumbai	BMC
2	502	Ambulance	true	Delhi	AIIMS
3	503	Fire Truck	true	Chennai	Tamil Nadu Fire Service
4	505	Rescue Boat	true	Kochi	Kerala Coast Guard
5	506	Medicines	true	Hyderabad	Apollo Pharmacy
6	507	Food Supplies	true	Kolkata	West Bengal Food Corporation
7	508	Shelter Tents	true	Pune	NDRF
8	509	Water Tanker	true	Jaipur	Rajasthan Municipal Corporation
9	510	Medical Kits	true	Lucknow	King George's Medical College

Total rows: 88 of 88 Query complete 00:00:00.102 Ln 2, Col 1

5. Display all disasters by severity level 5.

SELECT * FROM Disaster WHERE Severity = 5;

```
1  SELECT * FROM Disaster WHERE Severity = 5;
2
3
4
5
```

Data Output Messages Notifications

SQL

	disaster_id [PK] integer	date date	location character varying (50)	type character varying (50)	severity integer
1	2	2004-07-25	Guwahati	Earthquake	5
2	7	2009-07-01	Varanasi	Flood	5
3	12	2014-12-15	Ahmedabad	Earthquake	5
4	17	2019-05-05	Nashik	Flood	5
5	23	2007-11-15	Bengaluru	Earthquake	5
6	28	2012-04-05	Raipur	Earthquake	5
7	32	2016-08-01	Vadodara	Earthquake	5
8	36	2020-12-05	Guwahati	Flood	5
9	40	2015-04-10	Patiala	Earthquake	5

Total rows: 20 of 20 Query complete 00:00:00.128 Ln 2, Col 1

6. Count volunteers aged over 30.

```
SELECT COUNT(*) FROM Volunteers WHERE Age > 30;
```

The screenshot shows a SQL query interface with the following details:

- Query code:

```
1 SELECT COUNT(*) FROM Volunteers WHERE Age > 30;
```
- Result table:

count	bigint
1	9
- Message bar: Total rows: 1 of 1 | Query complete 00:00:00.067
- Bottom right corner: Ln 2, Col 1

7. Display all NGOs with 'Disaster' in their name.

```
SELECT * FROM NGOs WHERE NGO_name LIKE '%Disaster%';
```

Query Query History

```

1  SELECT * FROM NGOs WHERE NGO_name LIKE '%Disaster%';
2
3
4
5
6
7

```

Data Output Messages Notifications

	ngo_id [PK] integer	ngo_name character varying	service character varying
1	706	National Disaster Response Force (NDRF)	Disaster Management and Response
2	713	Disaster Emergency Committee	Emergency Response Coordination
3	742	All India Disaster Management Institute	Training and Capacity Building
4	746	National Disaster Management Authority (NDMA)	Policy Development and Training
5	757	Uttar Pradesh Disaster Management Department	State-level Disaster Management
6	768	Network for Disaster Management	Research and Policy Advocacy
7	769	Association for Disaster Management	Training for Local Communities
8	770	Institute of Disaster Management	Research and Capacity Building
9	777	National Disaster Relief Fund	Funding for Disaster Management

Total rows: 10 of 10 Query complete 00:00:00.133 Ln 2, Col 1

8. Retrieve all disasters of type 'Flood'.

SELECT * FROM Disaster WHERE Type = 'Flood';

```

1  SELECT * FROM Disaster WHERE Type = 'Flood';
2
3
4
5
6
7
8

```

Data Output Messages Notifications

	disaster_id [PK] integer	date date	location character varying (50)	type character varying (50)	severity integer
1	1	2003-06-15	Mumbai	Flood	4
2	6	2008-06-12	Hyderabad	Flood	3
3	7	2009-07-01	Varanasi	Flood	5
4	9	2011-09-15	Ludhiana	Flood	4
5	10	2012-10-10	Patna	Flood	3
6	14	2016-02-10	Bhubaneswar	Flood	3
7	17	2019-05-05	Nashik	Flood	5
8	18	2020-06-01	Surat	Flood	1
9	22	2006-10-01	Mysore	Flood	3

Total rows: 29 of 29 Query complete 00:00:00.137 Ln 2, C

9. Display all available shelter tents.

```
SELECT * FROM Resources WHERE Type = 'Shelter Tents' AND Availability = TRUE;
```

The screenshot shows a SQL database interface with the following details:

- SQL Editor:** The code `SELECT * FROM Resources WHERE Type = 'Shelter Tents' AND Availability = TRUE;` is entered.
- Data Output:** The results are displayed in a table:

resource_id	[PK] integer	type	character varying	availability	boolean	location	character varying	source	character varying
1	508	Shelter Tents		true		Pune		NDRF	
2	518	Shelter Tents		true		Thiruvananthapuram		NDRF	
3	527	Shelter Tents		true		Amritsar		Punjab Disaster Relief	
4	535	Shelter Tents		true		Solapur		NDRF	
5	544	Shelter Tents		true		Udaipur		NDRF	
6	553	Shelter Tents		true		Asansol		West Bengal Disaster Relief	
7	562	Shelter Tents		true		Kollam		Kerala Disaster Relief	
8	571	Shelter Tents		true		Tirunelveli		Tamil Nadu Disaster Relief	
9	580	Shelter Tents		true		Bhilai		Chhattisgarh Disaster Relief	

- Message Bar:** "Total rows: 10 of 10" and "Query complete 00:00:00.148".
- Status Bar:** "Ln 2, Col 1".

10. Display all international agencies.

```
SELECT * FROM Agency WHERE Type = 'International';
```

Query History

```

1 SELECT * FROM Agency WHERE Type = 'International';
2
3
4
5
6
7
8
9
10

```

Data Output

	agency_id	name	type	role
1	1011	World Health Organization	International	Health support during disasters
2	1012	International Federation of Red Cross	International	Global humanitarian response
3	1013	United Nations Office for the Coordination of Humanitarian Affairs	International	Coordination of disaster relief efforts
4	1014	Food and Agriculture Organization	International	Food security in emergencies
5	1015	International Organization for Migration	International	Migration support during crises
6	1016	World Bank	International	Financial assistance for disaster recovery
7	1017	Asian Development Bank	International	Development assistance for disaster recovery
8	1018	Red Cross Society	International	Humanitarian aid
9	1019	CARE International	International	Emergency relief and recovery

Total rows: 70 of 70 Query complete 00:00:00.134 Ln 2, Col 1

11. Find all volunteers who are currently unavailable.

SELECT * FROM Volunteers WHERE availability = FALSE;

```

1 SELECT * FROM Volunteers WHERE availability = FALSE;
2
3
4
5
6
7
8
9
10
11

```

Data Output

	volunteer_id	volunteer_name	age	location	availability	ngo_id	agency_id
1	10003	Anaya Gupta	21	Chennai	false	703	1003
2	10005	Reyansh Mehta	30	Hyderabad	false	705	1065
3	10008	Aditi Reddy	20	Kolkata	false	708	1008
4	10011	Advik Singh	27	Lucknow	false	711	1011
5	10015	Aarohi Patel	24	Indore	false	715	1015
6	10018	Rudra Singh	23	Patna	false	718	1018
7	10021	Riya Choudhury	22	Ludhiana	false	721	1021
8	10023	Ishaan Jain	25	Nashik	false	723	1023
9	10027	Rishi Khanna	28	Jabalpur	false	727	1027

Total rows: 26 of 26 Query complete 00:00:00.104 Ln 8, Col 1

12. List all victims with a 'Critical' health status.

```
SELECT * FROM Victims WHERE Health_status = 'Critical';
```

The screenshot shows a SQL query execution interface. The top section is labeled "Query" and contains the SQL code: "SELECT * FROM Victims WHERE Health_status = 'Critical';". Below this is a "Data Output" section which displays a table of results. The table has columns: victim_id, victim_name, location, health_status, rescue_status, community_pincode, and community_location. The data shows 9 rows of victims, all with the health_status 'Critical'. The results are as follows:

	victim_id [PK] integer	victim_name character varying	location character varying	health_status character varying	rescue_status character varying	community_pincode integer	community_location character varying
1	703	Aditya Verma	Karol Bagh	Critical	Pending	110004	Paharganj
2	706	Krishna Nair	Marine Lines	Critical	Pending	400003	Bandra
3	710	Saanvi Rao	Gachibowli	Critical	Pending	500003	Secunderabad
4	713	Kunal Menon	Bareilly	Critical	Pending	800001	Shahjahanpur
5	717	Advik Sharma	Janakpuri	Critical	Pending	600003	Nungambakkam
6	720	Kavya Dutta	Marine Lines	Critical	Pending	600006	Kotturpuram
7	723	Anika Menon	Gachibowli	Critical	Pending	600010	Tambaram
8	726	Rahul Agrawal	Karol Bagh	Critical	Pending	400004	Andheri
9	729	Meera Iyer	Janakpuri	Critical	Pending	500003	Secunderabad

Total rows: 29 of 29 Query complete 00:00:00.147 Ln 1, Col 22

13. List all resources with type 'Ambulance'.

```
SELECT * FROM Resources WHERE Type = 'Ambulance';
```

Query Query History

```

1  SELECT * FROM Resources WHERE Type = 'Ambulance';
2
3
4
5
6
7
8
9
10
11
12

```

Data Output Messages Notifications

	resource_id [PK] integer	type character varying	availability boolean	location character varying	source character varying
1	502	Ambulance	true	Delhi	AIIMS
2	514	Ambulance	true	Indore	MP Medical Services
3	520	Ambulance	true	Ludhiana	Punjab Health Services
4	529	Ambulance	true	Rajkot	Gujarat Health Services
5	537	Ambulance	true	Tiruppur	Tamil Nadu Health Services
6	546	Ambulance	true	Belgaum	Karnataka Health Services
7	555	Ambulance	true	Nanded	Maharashtra Health Services
8	564	Ambulance	true	Aligarh	UP Health Services
9	573	Ambulance	true	Malegaon	Maharashtra Health Services

Total rows: 10 of 10 Query complete 00:00:00.131 Ln 2, Col 1

14. Display all communities with names starting with 'B'.

SELECT * FROM Communities WHERE Location LIKE 'B%';

Query Query History

```

1  SELECT * FROM Communities WHERE Location LIKE 'B%';
2
3
4
5
6
7
8
9
10
11
12

```

Data Output Messages Notifications

	pincode [PK] integer	location [PK] character varying (50)	population integer
1	400003	Bandra	700
2	400007	Borivali	5000
3	500001	Banjara Hills	600
4	600008	Besant Nagar	250
5	700003	Behala	900
6	700007	Bidhanagar	300
7	700009	Baranagar	600
8	800002	Bareilly	1500
9	900002	Biratnagar	4000

Total rows: 14 of 14 Query complete 00:00:00.150 Ln 2, Col 1

15. List all agencies with a role in 'rescue operations'.

```
SELECT * FROM Agency WHERE Role LIKE '%rescue operations%';
```

The screenshot shows a database query interface with the following details:

- Query History:** A list of numbered lines from 1 to 12, with line 1 containing the query: `SELECT * FROM Agency WHERE Role LIKE '%rescue operations%';`
- Data Output:** A table showing the results of the query. The table has four columns: `agency_id`, `name`, `type`, and `role`. The data is as follows:

	agency_id [PK] integer	name character varying	type character varying	role text
1	1001	National Disaster Response Force	Government	Disaster response and rescue operations
2	1009	Indian Air Force	Military	Airlift and rescue operations

- Total rows:** 2 of 2
- Query complete:** 00:00:00.119
- Ln 2, Col 1:** This is a status message indicating the current row and column being processed.

16. Find all victims located in 'Kolkata'.

```
SELECT * FROM Victims WHERE Location = 'Kolkata';
```

Query Query History

```

1  SELECT * FROM Victims WHERE Location = 'Kolkata';
2
3
4
5
6
7
8
9
10
11
12
13

```

Data Output Messages Notifications

	victim_id [PK] integer	victim_name character varying	location character varying	health_status character varying	rescue_status character varying	community_pincode integer	community_location character varying
1	712	Riya Malhotra	Kolkata	Injured	Pending	700002	Garia
2	724	Nishant Patil	Kolkata	Stable	Rescued	600002	Mylapore
3	741	Aditi Singh	Kolkata	Stable	Rescued	600003	Nungambakkam
4	750	Ravi Sethi	Kolkata	Stable	Rescued	700005	New Town
5	767	Tanvi Sharma	Kolkata	Stable	Rescued	700006	Howrah
6	784	Anirudh Joshi	Kolkata	Stable	Rescued	400004	Andheri
7	795	Riya Gupta	Kolkata	Injured	Pending	700002	Garia
8	817	Pooja Reddy	Kolkata	Stable	Rescued	700002	Garia

Total rows: 8 of 8 Query complete 00:00:00.076 Ln 1, Col 22

17. Display all victims with a rescue status of 'Pending'.

`SELECT * FROM Victims WHERE Rescue_status = 'Pending';`

Query Query History

```

1  SELECT * FROM Victims WHERE Rescue_status = 'Pending';
2
3
4
5
6
7
8
9
10
11
12
13
14

```

Data Output Messages Notifications

	victim_id [PK] integer	victim_name character varying	location character varying	health_status character varying	rescue_status character varying	community_pincode integer	community_location character varying
1	701	Aarav Singh	Karol Bagh	Injured	Pending	110001	Connaught Place
2	703	Aditya Verma	Karol Bagh	Critical	Pending	110004	Paharganj
3	705	Arjun Joshi	Lajpat Nagar	Injured	Pending	110007	Dwarka
4	706	Krishna Nair	Marine Lines	Critical	Pending	400003	Bandra
5	708	Ayaan Kapoor	Gachibowli	Injured	Pending	500001	Banjara Hills
6	710	Saanvi Rao	Gachibowli	Critical	Pending	500003	Secunderabad
7	712	Riya Malhotra	Kolkata	Injured	Pending	700002	Garia
8	713	Kunal Menon	Bareilly	Critical	Pending	800001	Shahjahanpur
9	715	Maya Reddy	Lucknow	Injured	Pending	600001	T. Nagar

Total rows: 66 of 66 Query complete 00:00:00.078 Ln 1, Col 22

18. Find the oldest disaster recorded in the database.

```
SELECT * FROM Disaster ORDER BY Date ASC LIMIT 1;
```

The screenshot shows a SQL query interface with the following details:

- Query History:** A list of numbered lines from 1 to 15. Line 1 contains the query: `SELECT * FROM Disaster ORDER BY Date ASC LIMIT 1;`
- Data Output:** A table showing the results of the query. The table has columns: `disaster_id`, `date`, `location`, `type`, and `severity`. There is one row with values: 1, 2003-06-15, Mumbai, Flood, 4.
- Status Bar:** Shows "Total rows: 1 of 1" and "Query complete 00:00:00.095".
- Toolbar:** Includes icons for file operations (New, Open, Save, Print, etc.) and a SQL button.

19. Display the number of disasters in each location.

```
SELECT Location, COUNT(*) AS Disaster_Count FROM Disaster GROUP  
BY Location;
```

Query History

```

1 SELECT Location, COUNT(*) AS Disaster_Count FROM Disaster GROUP BY Location;
2
3
4
5
6
7
8
9
10
11
12
13
14
15

```

Data Output

	location	disaster_count
1	Jaipur	1
2	Patna	2
3	Visakhapatnam	2
4	Kochi	2
5	Gwalior	2
6	Hyderabad	1
7	Indore	3
8	Dibrugarh	1
9	Bhuj	2

Total rows: 46 of 46 Query complete 00:00:00.067 Ln 2, Col 1

20. Find all disasters with a type of 'Cyclone' or 'Tornado'.

SELECT * FROM Disaster WHERE Type IN ('Cyclone', 'Tornado');

Query History

```

1 SELECT * FROM Disaster WHERE Type IN ('Cyclone', 'Tornado');
2
3
4
5
6
7
8
9
10
11
12
13
14
15

```

Data Output

	disaster_id	date	location	type	severity
1	3	2005-08-10	Delhi	Tornado	3
2	5	2007-05-05	Chennai	Cyclone	4
3	11	2013-11-30	Pune	Tornado	2
4	13	2015-01-05	Kolkata	Cyclone	4
5	19	2021-07-10	Lucknow	Cyclone	3
6	20	2010-08-05	Bhopal	Tornado	4
7	24	2008-12-10	Visakhapatnam	Cyclone	4
8	27	2011-03-10	Nagpur	Tornado	4
9	31	2015-07-25	Chandigarh	Cyclone	4

Total rows: 24 of 24 Query complete 00:00:00.070 Ln 2, Col 1

21. List all volunteers associated with the 'NDRF' agency.

```
SELECT v.Volunteer_name  
FROM Volunteers v  
JOIN Agency a ON v.Agency_ID = a.Agency_ID  
WHERE a.Name = 'National Disaster Response Force';
```

The screenshot shows a SQL query editor interface. The query window displays the following code:

```
1 < SELECT v.Volunteer_name  
2   FROM Volunteers v  
3   JOIN Agency a ON v.Agency_ID = a.Agency_ID  
4 WHERE a.Name = 'National Disaster Response Force';  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15
```

The results pane shows a single row of data:

volunteer_name
Aarav Sharma

Below the results, status information is displayed:

Total rows: 1 of 1 Query complete 00:00:00.115 Ln 5, Col 1

22. List all international agencies and their contacts.

```
SELECT a.Name, ac.Contact_info  
FROM Agency a  
JOIN Agency_Contacts ac ON a.Agency_ID = ac.Agency_ID  
WHERE a.Type = 'International';
```

The screenshot shows a SQL query editor interface. At the top, there's a toolbar with icons for file operations like New, Open, Save, Print, and a magnifying glass. Below the toolbar, the title bar says "Query" and "Query History". The main area contains a numbered SQL query:

```

1 ✓ SELECT a.Name, ac.Contact_info
2   FROM Agency a
3   JOIN Agency_Contacts ac ON a.Agency_ID = ac.Agency_ID
4 WHERE a.Type = 'International';
5
6
7
8
9
10
11
12
13
14
15

```

Below the query, there are tabs for "Data Output", "Messages", and "Notifications". The "Data Output" tab is selected, displaying a table with two columns: "name" and "contact_info". The data consists of nine rows, each representing an organization and its contact information. The table has a header row with column names and data types. The footer of the table shows "Total rows: 133 of 133" and "Query complete 00:00:00.065". In the bottom right corner of the editor window, it says "Ln 5, Col 1".

	name character varying	contact_info character varying
1	World Health Organization	+41-1234567890
2	World Health Organization	who@who.int
3	International Federation of Red Cross	+41-0987654321
4	International Federation of Red Cross	ifrc@ifrc.org
5	United Nations Office for the Coordination of Humanitarian Affairs	+1-234567890
6	United Nations Office for the Coordination of Humanitarian Affairs	ocha@un.org
7	Food and Agriculture Organization	+39-1234567890
8	Food and Agriculture Organization	fao@fao.org
9	International Organization for Migration	+41-234567890

23. Display the top 5 most populated communities.

SELECT Location, Population

FROM Communities

ORDER BY Population DESC

LIMIT 5;

The screenshot shows a SQL query editor interface. At the top, there is a toolbar with icons for file operations like New, Open, Save, Print, and a magnifying glass. Below the toolbar is a menu bar with 'Query' and 'Query History'. The main area contains a code editor with the following SQL query:

```

1 v SELECT Location, Population
2   FROM Communities
3 ORDER BY Population DESC
4 LIMIT 5;
5
6
7
8
9
10
11
12
13
14
15

```

Below the code editor is a 'Data Output' tab, which is selected. The output area displays a table with the following data:

	location	population
	character varying (50)	integer
1	Dhaka	50000
2	Chittagong	40000
3	Khulna	25000
4	Sylhet	12000
5	Navi Mumbai	10000

At the bottom of the interface, there is a status bar with the text 'Total rows: 5 of 5' and 'Query complete 00:00:00.074' on the left, and 'Ln 5, Col 1' on the right.

24. Find the average population of communities affected by 'Flood'.

SELECT AVG(c.Population)

FROM Communities c

JOIN Disaster d ON c.Location = d.Location

WHERE d.Type = 'Flood';

```
1 ✓ SELECT AVG(c.Population)
2   FROM Communities c
3   JOIN Disaster d ON c.Location = d.Location
4   WHERE d.Type = 'Flood';
5
6
7
8
9
10
11
12
13
14
15
```

Data Output Messages Notifications

	avg	lock
1	7200.0000000000000000	

Total rows: 1 of 1 Query complete 00:00:00.117 Ln 5, Col 1

25. Display the details of disasters with a severity greater than the average.

```
SELECT *
FROM Disaster
WHERE Severity > (SELECT AVG(Severity) FROM Disaster);
```

The screenshot shows a SQL query editor interface. The top section contains the following SQL code:

```

1 v SELECT *
2 FROM Disaster
3 WHERE Severity > (SELECT AVG(Severity) FROM Disaster);
4
5
6
7
8
9
10
11
12
13
14
15

```

The bottom section displays the results of the query:

	disaster_id	date	location	type	severity
1	1	2003-06-15	Mumbai	Flood	4
2	2	2004-07-25	Guwahati	Earthquake	5
3	5	2007-05-05	Chennai	Cyclone	4
4	7	2009-07-01	Varanasi	Flood	5
5	9	2011-09-15	Ludhiana	Flood	4
6	12	2014-12-15	Ahmedabad	Earthquake	5
7	13	2015-01-05	Kolkata	Cyclone	4
8	16	2018-04-15	Gurgaon	Earthquake	4
9	17	2019-05-05	Nashik	Flood	5

Total rows: 43 of 43 Query complete 00:00:00.116 Ln 4, Col 1

26. Count the number of volunteers aged over 30 associated with each agency.

SELECT a.Name, COUNT(v.Volunteer_ID) AS Volunteer_Count

FROM Agency a

JOIN Volunteers v ON a.Agency_ID = v.Agency_ID

WHERE v.Age > 30

GROUP BY a.Name;

The screenshot shows a SQL query editor interface. At the top, there's a 'Query History' tab and a code editor area with the following SQL query:

```

1 v SELECT a.Name, COUNT(v.Volunteer_ID) AS Volunteer_Count
2 FROM Agency a
3 JOIN Volunteers v ON a.Agency_ID = v.Agency_ID
4 WHERE v.Age > 30
5 GROUP BY a.Name;
6
7
8
9
10
11
12
13
14
15

```

Below the code editor is a 'Data Output' tab, which displays the results of the query in a table:

	name	volunteer_count
1	International Federation of Red Cross	1
2	International Council for Science	1
3	United Nations Development Programme	1
4	International Society for Disaster Risk Reduction	1
5	Earthquake Engineering Research Institute	1
6	Ministry of Home Affairs	1
7	Global Network for Disaster Reduction	1
8	Oxfam International	1
9	World Vision	1

At the bottom of the table, it says 'Total rows: 9 of 9'. To the right of the table, it says 'Query complete 00:00:00.131' and 'Ln 6, Col 1'.

27. Find disaster types occurring in locations where 'Water Tanker' resources are available.

SELECT DISTINCT d.Type

FROM Disaster d

JOIN Resources r ON d.Location = r.Location

WHERE r.Type = 'Water Tanker' AND r.Availability = TRUE;

```

1 ✓ SELECT DISTINCT d.Type
2   FROM Disaster d
3   JOIN Resources r ON d.Location = r.Location
4 WHERE r.Type = 'Water Tanker' AND r.Availability = TRUE;
5
6
7
8
9
10
11
12
13
14
15

```

Data Output Messages Notifications

	type
1	Heatwave
2	Flood

Total rows: 2 of 2 Query complete 00:00:00.118 Ln 5, Col 1

28. List all disasters along with the total available resources at each disaster location.

```

SELECT d.Type, d.Location, COUNT(r.Resource_ID) AS
Total_Available_Resources
FROM Disaster d
JOIN Resources r ON d.Location = r.Location
WHERE r.Availability = TRUE
GROUP BY d.Type, d.Location;

```

The screenshot shows a SQL query being run in a database environment. The query is:

```

1 ✓ SELECT d.Type, d.Location, COUNT(r.Resource_ID) AS Total_Available_Resources
2   FROM Disaster d
3   JOIN Resources r ON d.Location = r.Location
4   WHERE r.Availability = TRUE
5   GROUP BY d.Type, d.Location;
6
7
8
9
10
11
12
13
14
15

```

The results are displayed in a table:

	type	location	total_available_resources
1	Flood	Indore	1
2	Flood	Kolkata	1
3	Flood	Jodhpur	1
4	Landslide	Shimla	1
5	Tornado	Delhi	1
6	Heatwave	Jaipur	1
7	Earthquake	Vadodara	2
8	Heatwave	Guwahati	1
9	Tornado	Ludhiana	1

Total rows: 49 of 49 Query complete 00:00:00.106 Ln 6, Col 1

29. Retrieve communities that have more than 5,000 people and have been affected by floods.

SELECT c.Location, c.Population

FROM Communities c

JOIN Disaster d ON c.Location = d.Location

WHERE d.Type = 'Flood' AND c.Population > 5000;

The screenshot shows a SQL query editor interface. At the top, there are tabs for 'Query' (which is selected) and 'Query History'. Below the tabs is a code editor area containing the following SQL query:

```

1 v SELECT c.Location, c.Population
2 FROM Communities c
3 JOIN Disaster d ON c.Location = d.Location
4 WHERE d.Type = 'Flood' AND c.Population > 5000;
5
6
7
8
9
10
11
12
13
14
15

```

Below the code editor is a 'Data Output' tab, followed by 'Messages' and 'Notifications' tabs. Under the 'Data Output' tab, there is a toolbar with icons for file operations (New, Open, Save, Print, Copy, Paste, Find, Refresh, SQL) and a dropdown menu. The main area displays a table with two columns: 'location' and 'population'. The data is as follows:

	location	population
1	Kanpur	7000
2	Lucknow	9000
3	Kanpur	7000
4	Lucknow	9000

At the bottom of the interface, there is a status bar with the text 'Total rows: 4 of 4' and 'Query complete 00:00:00.135' on the left, and 'Ln 4, Col 47' on the right.

30. Find the number of resources available per resource type for all resources located in Delhi.

```

SELECT r.Type, COUNT(*) AS Available_Count
FROM Resources r
WHERE r.Location = 'Delhi' AND r.Availability = TRUE
GROUP BY r.Type;

```

The screenshot shows a SQL query editor interface. At the top, there's a toolbar with icons for file operations like Open, Save, Print, and a magnifying glass. Below the toolbar is a menu bar with 'Query' and 'Query History'. The main area contains a code editor with the following SQL query:

```

1 ✓ SELECT r.Type, COUNT(*) AS Available_Count
2   FROM Resources r
3 WHERE r.Location = 'Delhi' AND r.Availability = TRUE
4 GROUP BY r.Type;
5
6
7
8
9
10
11
12
13
14
15

```

Below the code editor is a 'Data Output' tab, which is selected. The results are displayed in a table:

	type	available_count
1	Ambulance	1

At the bottom of the interface, there are status messages: 'Total rows: 1 of 1' and 'Query complete 00:00:00.092'.

31. Retrieve the names of volunteers, their ages, and the agencies they are assigned to for volunteers under 30.

SELECT v.Volunteer_name, v.Age, a.Name AS Agency

FROM Volunteers v

JOIN Agency a ON v.Agency_ID = a.Agency_ID

WHERE v.Age < 30;

The screenshot shows a SQL query editor interface. At the top, there is a 'Query' tab and a 'Query History' tab. Below the tabs is a code editor area containing the following SQL query:

```

1 ✓ SELECT v.Volunteer_name, v.Age, a.Name AS Agency
2   FROM Volunteers v
3   JOIN Agency a ON v.Agency_ID = a.Agency_ID
4 WHERE v.Age < 30;
5
6
7
8
9
10
11
12
13
14
15

```

Below the code editor is a 'Data Output' tab, which is currently selected. The data output area displays a table with three columns: 'volunteer_name', 'age', and 'agency'. The data is as follows:

	volunteer_name	age	agency
1	Aarav Sharma	24	National Disaster Response Force
2	Vivian Singh	29	Mennonite Central Committee
3	Anaya Gupta	21	Indian Meteorological Department
4	Diya Patel	26	National Remote Sensing Centre
5	Saanvi Kumar	23	National Disaster Management Authority
6	Aditi Reddy	20	Indian Navy
7	Aryan Chaudhary	28	Indian Air Force
8	Kavya Rao	22	Indian Coast Guard
9	Advik Singh	27	World Health Organization

At the bottom of the data output area, it says 'Total rows: 72 of 72' and 'Query complete 00:00:00.114'. To the right, it says 'Ln 5, Col 1'.

32. List all locations with available 'Medicines' resources, along with the disaster types at those locations.

```

SELECT d.Location, d.Type AS Disaster_Type, r.Type AS Resource_Type
FROM Disaster d
JOIN Resources r ON d.Location = r.Location
WHERE r.Type = 'Medicines' AND r.Availability = TRUE;

```

Query Query History

```

1 SELECT d.Location, d.Type AS Disaster_Type, r.Type AS Resource_Type
2 FROM Disaster d
3 JOIN Resources r ON d.Location = r.Location
4 WHERE r.Type = 'Medicines' AND r.Availability = TRUE;
5
6
7
8
9
10
11
12
13
14
15

```

Data Output Messages Notifications

	location character varying (50)	disaster_type character varying (50)	resource_type character varying
1	Hyderabad	Flood	Medicines
2	Patna	Flood	Medicines
3	Patna	Tornado	Medicines

Total rows: 3 of 3 Query complete 00:00:00.083 Ln 5, Col 1

33. Find all NGOs that have contacts ending in .org.

```

SELECT ngo.NGO_name, nc.Contact_info
FROM NGOs ngo
JOIN NGO_Contacts nc ON ngo.NGO_ID = nc.NGO_ID
WHERE nc.Contact_info LIKE '%.org';

```

Query History

```

1 ✓ SELECT ngo.NGO_name, nc.Contact_info
2   FROM NGOs ngo
3     JOIN NGO_Contacts nc ON ngo.NGO_ID = nc.NGO_ID
4   WHERE nc.Contact_info LIKE '%.org';
5
6
7
8
9
10
11
12
13
14
15

```

Data Output Notifications

	ngo_name	contact_info
	character varying	character varying
1	SEEDS	info@seedsindia.org
2	Goonj	contact@goonj.org
3	HelpAge India	helpline@helpageindia.org
4	Oxfam India	support@oxfamindia.org
5	CARE India	info@careindia.org
6	Indian Red Cross Society	info@indianredcross.org
7	The Akshaya Patra Foundation	contact@akshayapatra.org
8	SankalpTaru	info@sankalptaru.org
9	Save the Children India	support@savechildrenindia.org

Total rows: 74 of 74 Query complete 00:00:00.101 Ln 5, Col 1

34. List all disasters in communities with population greater than the average population of all communities.

SELECT d.*

FROM Disaster d

JOIN Communities c ON d.Location = c.Location

WHERE c.Population > (SELECT AVG(Population) FROM Communities);

```

1 ✓ SELECT d.*  

2   FROM Disaster d  

3   JOIN Communities c ON d.Location = c.Location  

4 WHERE c.Population > (SELECT AVG(Population) FROM Communities);  

5  

6  

7  

8  

9  

10  

11  

12  

13  

14  

15

```

Data Output Messages Notifications

	disaster_id [PK] integer	date date	location character varying (50)	type character varying (50)	severity integer
1	13	2015-01-05	Kolkata	Cyclone	4
2	19	2021-07-10	Lucknow	Cyclone	3
3	25	2009-01-20	Kanpur	Flood	2
4	30	2014-06-10	Agra	Heatwave	2
5	48	2015-12-15	Kolkata	Flood	2
6	58	2012-10-10	Lucknow	Flood	4
7	69	2020-09-10	Kanpur	Flood	3
8	70	2021-10-05	Kolkata	Earthquake	5
9	78	2012-01-20	Lucknow	Flood	5

Total rows: 9 of 9 Query complete 00:00:00.126 Ln 4, Col 21

35. Display all disasters that affected communities where the population exceeds 2000.

SELECT d.Type, d.Location

FROM Disaster d

JOIN Communities c ON d.Location = c.Location

WHERE c.Population > 2000;

Query Query History

```

1 ✓ SELECT d.Type, d.Location
2   FROM Disaster d
3   JOIN Communities c ON d.Location = c.Location
4   WHERE c.Population > 2000;
5
6
7
8
9
10
11
12
13
14
15

```

Data Output Messages Notifications



	type character varying (50)	location character varying (50)
1	Cyclone	Kolkata
2	Cyclone	Lucknow
3	Flood	Kanpur
4	Heatwave	Agra
5	Flood	Kolkata
6	Flood	Lucknow
7	Flood	Kanpur
8	Earthquake	Kolkata
9	Flood	Lucknow

Total rows: 9 of 9 Query complete 00:00:00.079 Ln 4, Col 26

36. Find the average age of volunteers for each NGO and display NGOs with an average age over 25.

SELECT ngo.NGO_name, AVG(v.Age) AS Avg_Age

FROM NGOs ngo

JOIN Volunteers v ON ngo.NGO_ID = v.NGO_ID

GROUP BY ngo.NGO_name

HAVING AVG(v.Age) > 25;

Query Query History

```

1 ✓ SELECT ngo.NGO_name, AVG(v.Age) AS Avg_Age
2   FROM NGOs ngo
3   JOIN Volunteers v ON ngo.NGO_ID = v.NGO_ID
4   GROUP BY ngo.NGO_name
5   HAVING AVG(v.Age) > 25;
6
7
8
9
10
11
12
13
14
15

```

Data Output Messages Notifications

SQL

	ngo.name	avg_age
	character varying	numeric
1	Udyam	29.000000000000000000000000000000
2	Sankalp	27.000000000000000000000000000000
3	Gandhi Smriti and Darshan Samiti	29.000000000000000000000000000000
4	World Wildlife Fund (WWF) India	31.000000000000000000000000000000
5	Operation Blessing India	26.000000000000000000000000000000
6	Tsunami Recovery Program	29.000000000000000000000000000000
7	National Disaster Management Authority (NDMA)	28.000000000000000000000000000000
8	Swayam Shikshan Prayog	30.000000000000000000000000000000
9	Emergency Volunteers	26.000000000000000000000000000000

Total rows: 52 Query complete 00:00:00.140 Ln 6, Col 1

37. Find the locations and population of communities affected by 'Cyclone' disasters.

```

SELECT c.Location, c.Population
FROM Communities c
JOIN Disaster d ON c.Location = d.Location
WHERE d.Type = 'Cyclone';

```

```
1 ✓ SELECT c.Location, c.Population
2   FROM Communities c
3   JOIN Disaster d ON c.Location = d.Location
4 WHERE d.Type = 'Cyclone';
5
6
7
8
9
10
11
12
13
14
15
```

Data Output Messages Notifications

location population

	location	population
1	Kolkata	4000
2	Lucknow	9000

Total rows: 2 of 2 Query complete 00:00:00.065 Ln 5, Col 1

38. Find all communities affected by disasters that occurred after the year 2015.

```
SELECT DISTINCT c.Location, c.Population
FROM Communities c
JOIN Disaster d ON c.Location = d.Location
WHERE d.Date > '2015-01-01';
```

```

1 v SELECT DISTINCT c.Location, c.Population
2 FROM Communities c
3 JOIN Disaster d ON c.Location = d.Location
4 WHERE d.Date > '2015-01-01';
5
6
7
8
9
10
11
12
13
14
15

```

Data Output Messages Notifications

	location character varying (50)	population integer
1	Kanpur	7000
2	Lucknow	9000
3	Kolkata	4000

Total rows: 3 of 3 Query complete 00:00:00.100 Ln 5, Col 1

39. Get the names of social media platforms used for disasters of a specified type "Earthquake".

```

SELECT DISTINCT sp.Name AS PlatformName
FROM Disaster d
INNER JOIN Disaster_SocialMedia ds ON d.Disaster_ID = ds.Disaster_ID
INNER JOIN SocialMediaPlatform sp ON ds.PlatformName = sp.Name
WHERE d.Type = 'Earthquake';

```

The screenshot shows a SQL query editor interface. The top section is labeled "Query History" and contains the following SQL code:

```
1 SELECT DISTINCT sp.Name AS PlatformName
2 FROM Disaster d
3 INNER JOIN Disaster_SocialMedia ds ON d.Disaster_ID = ds.Disaster_ID
4 INNER JOIN SocialMediaPlatform sp ON ds.PlatformName = sp.Name
5 WHERE d.Type = 'Earthquake';
6
```

The bottom section is labeled "Data Output" and displays the results of the query. The results are presented in a table with one column labeled "platformname". The data is as follows:

platformname
AlertReady Canada
Hurricane Tracker
Facebook
WhatsApp
DisasterAlert
SafetyNet
Diaspora
Tumblr
FindShelter

Total rows: 50 of 50 | Query complete 00:00:00.137 | Ln 6, Col 1

40. Retrieve all social media platforms associated with Disaster_ID = 1.

```
SELECT dsp.PlatformName
FROM Disaster_SocialMedia dsp
JOIN Disaster d ON dsp.Disaster_ID = d.Disaster_ID
WHERE d.Disaster_ID = 1;
```

The screenshot shows a SQL query editor interface. The top section contains a code editor with the following SQL query:

```
1 ✓ SELECT dsp.PlatformName  
2   FROM Disaster_SocialMedia dsp  
3   JOIN Disaster d ON dsp.Disaster_ID = d.Disaster_ID  
4 WHERE d.Disaster_ID = 1;  
5
```

The bottom section shows the results of the query in a data grid:

platformname
Facebook
Twitter
Instagram
YouTube
WhatsApp

Below the grid, the status bar displays "Total rows: 5 of 5" and "Query complete 00:00:00.144".

41. Find all disasters that used a specific platform "Twitter" and the location of those disasters.

SELECT d.Disaster_ID, d.Location

FROM Disaster d

JOIN Disaster_SocialMedia dsp ON d.Disaster_ID = dsp.Disaster_ID

WHERE dsp.PlatformName = 'Twitter';

The screenshot shows a SQL query editor interface. The top section is labeled "Query History" and contains the following SQL code:

```
1 SELECT d.Disaster_ID, d.Location
2 FROM Disaster d
3 JOIN Disaster_SocialMedia dsp ON d.Disaster_ID = dsp.Disaster_ID
4 WHERE dsp.PlatformName = 'Twitter';
5
```

The bottom section is labeled "Data Output" and displays the results of the query as a table:

	disaster_id [PK] integer	location character varying (60)
1	1	Mumbai
2	19	Lucknow
3	37	Gwalior
4	55	Vadodara
5	73	Agartala

Below the table, the status bar shows "Total rows: 5 of 5" and "Query complete 00:00:00.075". On the right side of the status bar, it says "Ln 5, Col 1".

42. Get a count of social media platforms used per disaster.

```
SELECT d.Disaster_ID, d.Location, COUNT(dsp.PlatformName) AS
PlatformCount
FROM Disaster d
JOIN Disaster_SocialMedia dsp ON d.Disaster_ID = dsp.Disaster_ID
GROUP BY d.Disaster_ID, d.Location;
```

```

1 v SELECT d.Disaster_ID, d.Location, COUNT(dsp.PlatformName) AS PlatformCount
2 FROM Disaster d
3 JOIN Disaster_SocialMedia dsp ON d.Disaster_ID = dsp.Disaster_ID
4 GROUP BY d.Disaster_ID, d.Location;
5

```

Data Output Messages Notifications

SQL

	disaster_id [PK] integer	location character varying (50)	platformcount bigint
1	74	Srinagar	5
2	54	Nashik	2
3	29	Jodhpur	5
4	71	Bhuj	5
5	68	Jodhpur	5
6	4	Jaipur	5
7	34	Dehradun	5
8	51	Raipur	5
9	80	Surat	5

Total rows: 85 of 85 Query complete 00:00:00.264 Ln 5, Col 1

43. Display each community's location, population, and the count of available resources.

SELECT c.Location, c.Population, COUNT(r.Resource_ID) AS Available_Resources

FROM Communities c

JOIN Resources r ON c.Location = r.Location

WHERE r.Availability = TRUE

GROUP BY c.Location, c.Population;

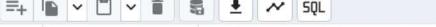
Query Query History

```

1 ✓ SELECT c.Location, c.Population, COUNT(r.Resource_ID) AS Available_Resources
2   FROM Communities c
3   JOIN Resources r ON c.Location = r.Location
4   WHERE r.Availability = TRUE
5   GROUP BY c.Location, c.Population;
6

```

Data Output Messages Notifications



	location character varying (50)	population integer	available_resources bigint
1	Noida	6000	1
2	Kanpur	7000	1
3	Aligarh	1000	1
4	Lucknow	9000	1
5	Kolkata	4000	1
6	Meerut	3000	1
7	Agra	5000	1
8	Moradabad	2000	1

Total rows: 8 of 8 Query complete 00:00:00.174 Ln 6, Col 1

INTERFACE IMPLEMENTATION

Tables:

1. Disaster (Disaster_ID, Date, Type, Location, severity)
2. Agency (Agency_ID, Agency_name, type, role)
3. Disaster_agencies (Disaster_ID, Agency_ID)

1.SETUP JDBC AND BASIC GUI

JDBC Setup: To connect the application to a PostgreSQL database, I used JDBC (Java Database Connectivity) to enable interaction with the database.

- **Step 1:** Download the PostgreSQL JDBC driver (postgresql-42.7.4.jar) and add it to the project classpath.
- **Step 2:** Establish a database connection using JDBC in the connectToDatabase() method.
- Code:

```
private void connectToDatabase() {  
    try {  
        // Database connection details  
        String url = "jdbc:postgresql://localhost:5432/disaster_agency";  
        String user = "postgres";  
        String password = "password";  
  
        // Establishing connection  
        connection = DriverManager.getConnection(url, user, password);  
        System.out.println("Connected to the database successfully.");  
    } catch (SQLException e) {  
        JOptionPane.showMessageDialog(this, "Database connection failed: " + e.getMessage());  
    }  
}
```

```
 }  
}
```

Explanation:

- The `connectToDatabase()` method uses the `DriverManager.getConnection()` method to establish a connection with the PostgreSQL database. If successful, it prints a success message. If not, it displays an error message through a dialog box.

2. CRUD OPERATIONS IN GUI

1. Overview of the Application

- The Disaster and Agency Management interface provides a user-friendly way to manage agencies and disasters, associate agencies with specific disasters, and display relevant information.
- Built using Java Swing for the GUI and JDBC for database interactions, it operates on Disaster and Agency tables along with a junction table disaster_agencies.
- Error handling, validation, and data integrity checks are implemented for reliable database operations.

2. JDBC Connection Setup

- A JDBC connection is established at the start of the program to allow interaction with a PostgreSQL database. The connection object is used across various operations.

```
Connection connection = DriverManager.getConnection (DB_URL,  
DB_USERNAME, DB_PASSWORD);
```

3. CRUD Operations for Agency

a. Add Agency

- Allows users to add a new agency by entering details in agencyNameField, agencyTypeField, and roleField.
- Data is then inserted into the Agency table.
- Code snippet :

```
String insertAgencySQL = "INSERT INTO Agency (name, type, role)  
VALUES (?, ?, ?);
```

b. Update Agency

- The system checks if an agency exists by agency_id before updating.
- Updated information (name, type, role) is entered, and the system updates the corresponding record in the Agency table.
- Validation: Alerts if the agency does not exist.
- Code Snippet:
- `String updateAgencySQL = "UPDATE Agency SET name = ?, type = ?, role = ? WHERE agency_id = ?";`

c. Delete Agency

- Deletes an agency by ID, first ensuring the record exists and then removing it from the disaster_agencies junction table to maintain referential integrity.
- Code Snippet:
- `ResultSet rs = statement.executeQuery("SELECT * FROM Agency");`

d. Show Agencies

- Retrieves and displays all records from the Agency table in a JTable.
- Dynamically builds the table model based on the table metadata, showing the flexibility of the showAgencies method.
- Code Snippet:
- `ResultSet rs = statement.executeQuery("SELECT * FROM Agency");`

4. CRUD Operations for Disaster

a. Add Disaster

- Adds a new disaster entry by allowing users to specify the date, location, and type for the disaster.
- Inserts the new record into the Disaster table.
- Example Code Snippet

```
String insertDisasterSQL = "INSERT INTO Disaster (date, location, type) VALUES (?, ?, ?);  
PreparedStatement pstmt =  
connection.prepareStatement(insertDisasterSQL);  
pstmt.setDate(1, java.sql.Date.valueOf(dateField.getText()));  
pstmt.setString(2, locationField.getText());  
pstmt.setString(3, typeField.getText());  
pstmt.executeUpdate();
```

b. Update Disaster

- Updates an existing disaster record in the Disaster table by disaster_id.
- Users enter the updated values for the fields they want to modify.
- If disaster_id does not exist, it alerts the user.
- Example Code Snippet

```
String updateDisasterSQL = "UPDATE Disaster SET date = ?,  
location = ?, type = ? WHERE disaster_id = ?";  
PreparedStatement pstmt =  
connection.prepareStatement(updateDisasterSQL);  
pstmt.setDate(1, java.sql.Date.valueOf(dateField.getText()));  
pstmt.setString(2, locationField.getText());  
pstmt.setString(3, typeField.getText());  
pstmt.setInt(4, Integer.parseInt(disasterIDField.getText()));  
pstmt.executeUpdate();
```

c. Delete Disaster

- Deletes a disaster entry based on disaster_id.
- Ensures related records in the disaster_agencies junction table are also handled to maintain data integrity.
- Example Code Snippet

```
String deleteDisasterSQL = "DELETE FROM Disaster WHERE  
disaster_id = ?";  
PreparedStatement pstmt =  
connection.prepareStatement(deleteDisasterSQL);  
pstmt.setInt(1, Integer.parseInt(disasterIDField.getText()));  
pstmt.executeUpdate();
```

d. Show Disasters

- Retrieves disaster records from the Disaster table and displays them in a JTable.
- Similar to the showAgencies method, it dynamically fetches column names and data.
- Code Snippet:

```
ResultSet rs = statement.executeQuery("SELECT * FROM  
Disaster");
```

5. Associating Agencies with Disasters

a. Add Agency for Disaster

- Users can associate an agency with a disaster by entering the disaster_id and agency_id.
- Inserts records into the disaster_agencies junction table.
- Code Snippet:

```
String sql = "INSERT INTO disaster_agencies (disaster_id,  
agency_id) VALUES (?, ?);
```

b. Delete Agency from Disaster

- Deletes a specific agency-disaster association by disaster_id and agency_id.
- Checks if the combination exists to prevent errors and provides feedback to the user.
- Code Snippet:

```
String deleteSql = "DELETE FROM disaster_agencies WHERE  
disaster_id = ? AND agency_id = ?";
```

6. Display Relationships (Join Operations)

a. Show Agencies for a Specific Disaster

- Displays a list of agencies associated with a selected disaster.
- Uses a join query to retrieve disaster and agency details, organizing the information in a JTable.
- Code Snippet:

```
String sql = "SELECT d.disaster_id, d.date, d.location, a.agency_id,  
a.name FROM Disaster d " + "JOIN disaster_agencies da ON  
d.disaster_id = da.disaster_id " + "JOIN Agency a ON a.agency_id =  
da.agency_id WHERE da.disaster_id=?";
```

b. Show Disasters for a Specific Agency

- Displays a list of disasters associated with a selected agency.
- Similar to the previous method but uses agency_id to filter.
- Code Snippet:

```
String sql = "SELECT a.agency_id, a.name, d.disaster_id, d.date,  
d.location " + "FROM Agency a JOIN disaster_agencies da ON  
a.agency_id = da.agency_id " + "JOIN Disaster d ON d.disaster_id =  
da.disaster_id WHERE da.agency_id=?";
```

7. Error Handling and Input Validation

- Every operation includes validation and error handling to provide user feedback on issues like missing fields, non-existent IDs, or SQL exceptions.
- For example, missing fields trigger JOptionPane dialogs to alert the user.
- Error handling example:
`JOptionPane.showMessageDialog(this, "Error updating agency: " +
e.getMessage());`

8. Demonstration of Application in Action

- Launch the application and showcase each functionality sequentially:
 - Adding, updating, deleting agencies
 - Displaying agencies and disasters
 - Associating agencies with disasters
 - Viewing agency-disaster relationships through joins

9. Conclusion

- This application demonstrates how JDBC facilitates seamless CRUD operations and relationships within a GUI.

- The interface manages disaster and agency records effectively, providing both data management and visualization.
- Proper validation, error handling, and use of SQL best practices ensure data consistency and reliable operations.

10. Snapshots

1. Add Disaster

The screenshot shows a window titled "Disaster and Agency Management". It has two main sections: "Disaster Details" and "Agency Details".

Disaster Details:

Disaster ID:	85
Date (YYYY-MM-DD):	2019-08-15
Location:	Gurgaon
Type:	Cyclone
Severity (1-5):	2

Agency Details:

Agency ID:	
Agency Name:	
Type:	
Role:	

Below these sections are four buttons: "Add Disaster", "Update Disaster", "Delete Disaster", and "Add Agency".

Underneath the buttons are two more buttons: "Update Agency" and "Delete Agency" on the left, and "Show Disasters" and "Show Agencies" on the right.

At the bottom is a table titled "Show Agencies for Disaster" and "Show Disasters for Agency". The table has columns: disaster_id, date, location, type, and severity. It lists various disaster entries with their details.

disaster_id	date	location	type	severity
51	2015-01-25	Kochi	Flood	4
62	2016-02-10	Guwahati	Heatwave	3
63	2017-03-15	Gurgaon	Tornado	4
64	2018-04-20	Nagpur	Earthquake	5
65	2019-05-10	Gwalior	Flood	1
66	2020-06-25	Mangaluru	Cyclone	4
67	2021-07-15	Bengaluru	Landslide	5
68	2019-08-30	Jodhpur	Heatwave	2
69	2020-09-10	Kanpur	Flood	3
70	2021-10-05	Kolkata	Earthquake	5
71	2005-06-10	Bhuj	Cyclone	4
72	2006-07-15	Mysore	Flood	2
73	2007-08-20	Agartala	Earthquake	5
74	2008-09-25	Srinagar	Landslide	3
75	2009-10-15	Delhi	Flood	4
76	2010-11-20	Bhopal	Tornado	2
77	2011-12-10	Indore	Heatwave	3
78	2012-01-20	Lucknow	Flood	5
79	2013-02-15	Nagpur	Earthquake	4
80	2014-03-05	Surat	Cyclone	5
81	2015-04-25	Puducherry	Flood	1
82	2016-05-15	Nashik	Tornado	3
83	2017-06-01	Patiala	Flood	4
84	2018-07-10	Visakhapatnam	Earthquake	5

Disaster and Agency Management

Disaster Details		Agency Details		
Disaster ID:		Agency ID:		
Date (YYYY-MM-DD):		Agency Name:		
Location:		Type:		
Type:		Role:		
Severity (1-5):				
Add Disaster		Update Disaster		
Update Agency		Delete Agency		
Show Agencies for Disaster		Show Disasters for Agency		
		Add Agency for Disaster		
		Delete Agency for Disaster		
disaster_id	date	location	type	severity
62	2016-02-10	Guwahati	Heatwave	3
63	2017-03-15	Gurgaon	Tornado	4
64	2018-04-20	Nagpur	Earthquake	5
65	2019-05-10	Gwalior	Flood	1
66	2020-06-25	Mangalore	Cyclone	4
67	2021-07-15	Bengaluru	Landslide	5
68	2019-08-30	Jodhpur	Heatwave	2
69	2020-09-10	Kanpur	Flood	3
70	2021-10-05	Kolkata	Earthquake	5
71	2005-06-10	Bhuj	Cyclone	4
72	2006-07-15	Mysore	Flood	2
73	2007-08-20	Agartala	Earthquake	5
74	2008-09-25	Srinagar	Landslide	3
75	2009-10-15	Delhi	Flood	4
76	2010-11-20	Bhopal	Tornado	2
77	2011-12-10	Indore	Heatwave	3
78	2012-01-20	Lucknow	Flood	5
79	2013-02-15	Nagpur	Earthquake	4
80	2014-03-05	Surat	Cyclone	5
81	2015-04-25	Puducherry	Flood	1
82	2016-05-15	Nashik	Tornado	3
83	2017-06-01	Patiala	Flood	4
84	2018-07-10	Visakhapatnam	Earthquake	5
85	2019-08-15	Gurgaon	Cyclone	2

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:	85	Agency ID:	
Date (YYYY-MM-DD):	2019-08-15	Agency Name:	
Location:	Gurgaon	Type:	
Type:	Cyclone	Role:	
Severity (1-5):	2		
Add Disaster		Update Disaster	
Update Agency		Delete Agency	
Show Agencies for Disaster		Show Disasters for Agency	
		Add Agency for Disaster	
		Delete Agency for Disaster	
disaster_id			severity
62	2016-02-10		3
63	2017-03-15		4
64	2018-04-20		5
65	2019-05-10		1
66	2020-06-25		4
67	2021-07-15	Bengaluru	5
68	2019-08-30	Jodhpur	2
69	2020-09-10	Kanpur	3
70	2021-10-05	Kolkata	5
71	2005-06-10	Bhuj	Cyclone
72	2006-07-15	Mysore	Flood
73	2007-08-20	Agartala	Earthquake
74	2008-09-25	Srinagar	Landslide
75	2009-10-15	Delhi	Flood
76	2010-11-20	Bhopal	Tornado
77	2011-12-10	Indore	Heatwave
78	2012-01-20	Lucknow	Flood
79	2013-02-15	Nagpur	Earthquake
80	2014-03-05	Surat	Cyclone
81	2015-04-25	Puducherry	Flood
82	2016-05-15	Nashik	Tornado
83	2017-06-01	Patiala	Flood
84	2018-07-10	Visakhapatnam	Earthquake
85	2019-08-15	Gurgaon	Cyclone

Message:

Error adding disaster: ERROR: duplicate key value violates unique constraint "disaster_pkey"
Detail: Key (disaster_id)=(85) already exists.

OK

2. Update Disaster

Disaster and Agency Management

Disaster Details				Agency Details			
Disaster ID:	85	Agency ID:		Date (YYYY-MM-DD):	2019-08-15	Agency Name:	
Location:	Ahmedabad	Type:	Cyclone	Severity (1-5):	2	Role:	

Add Disaster		Update Disaster		Delete Disaster		Add Agency	
Update Agency		Delete Agency		Show Disasters		Show Agencies	
Show Agencies for Disaster		Show Disasters for Agency		Add Agency for Disaster		Delete Agency for Disaster	

Message

i Disaster updated successfully!

OK

disaster_id	date	location	type	severity
62	2015-02-10	Gujarat	Heatwave	3
63	2017-03-15	Gurgaon	Tornado	4
64	2018-04-20	Nagpur	Earthquake	5
65	2019-05-10	Gwalior	Flood	1
66	2020-06-25	Mangalore	Cyclone	4
67	2021-07-15	Bengaluru	Landslide	5
68	2019-08-30	Jodhpur	Heatwave	2
69	2020-09-10	Kanpur	Flood	3
70	2021-10-05	Kolkata	Earthquake	5
71	2005-06-10	Bhuj	Cyclone	4
72	2006-07-15	Mysore	Flood	2
73	2007-08-20	Agartala	Earthquake	5
74	2008-09-25	Srinagar	Landslide	3
75	2009-10-15	Delhi	Flood	4
76	2010-11-20	Bhopal	Tornado	2
77	2011-12-10	Indore	Heatwave	3
78	2012-01-20	Lucknow	Flood	5
79	2013-02-15	Nagpur	Earthquake	4
80	2014-03-05	Surat	Cyclone	5
81	2015-04-25	Puducherry	Flood	1
82	2016-05-15	Nashik	Tornado	3
83	2017-06-01	Patiala	Flood	4
84	2018-07-10	Visakhapatnam	Earthquake	5
85	2019-08-15	Gurgaon	Cyclone	2

Windows Type here to search [Search icon] [File Explorer icon] [Word icon] [Excel icon] [PowerPoint icon] [Google Chrome icon] [Skype icon] [File icon] [Help icon] [Clock icon] [Battery icon] [Network icon] [User icon] 15:01 ENG 09-11-2024 [Feedback icon]

Disaster and Agency Management

Disaster Details				Agency Details			
Disaster ID:	85	Agency ID:		Date (YYYY-MM-DD):	2019-08-15	Agency Name:	
Location:	Ahmedabad	Type:	Cyclone	Severity (1-5):	2	Role:	

Add Disaster		Update Disaster		Delete Disaster		Add Agency	
Update Agency		Delete Agency		Show Disasters		Show Agencies	
Show Agencies for Disaster		Show Disasters for Agency		Add Agency for Disaster		Delete Agency for Disaster	

disaster_id	date	location	type	severity
62	2015-02-10	Gujarat	Heatwave	3
63	2017-03-15	Gurgaon	Tornado	4
64	2018-04-20	Nagpur	Earthquake	5
65	2019-05-10	Gwalior	Flood	1
66	2020-06-25	Mangalore	Cyclone	4
67	2021-07-15	Bengaluru	Landslide	5
68	2019-08-30	Jodhpur	Heatwave	2
69	2020-09-10	Kanpur	Flood	3
70	2021-10-05	Kolkata	Earthquake	5
71	2005-06-10	Bhuj	Cyclone	4
72	2006-07-15	Mysore	Flood	2
73	2007-08-20	Agartala	Earthquake	5
74	2008-09-25	Srinagar	Landslide	3
75	2009-10-15	Delhi	Flood	4
76	2010-11-20	Bhopal	Tornado	2
77	2011-12-10	Indore	Heatwave	3
78	2012-01-20	Lucknow	Flood	5
79	2013-02-15	Nagpur	Earthquake	4
80	2014-03-05	Surat	Cyclone	5
81	2015-04-25	Puducherry	Flood	1
82	2016-05-15	Nashik	Tornado	3
83	2017-06-01	Patiala	Flood	4
84	2018-07-10	Visakhapatnam	Earthquake	5
85	2019-08-15	Ahmedabad	Cyclone	2

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:	90	Agency ID:	
Date (YYYY-MM-DD):	2019-08-15	Agency Name:	
Location:	Ahmedabad	Type:	
Type:	Cyclone	Role:	
Severity (1-5):	2		

Add Disaster		Update Disaster	
Delete Disaster		Add Agency	
Update Agency		Delete Agency	
Show Agencies for Disaster		Show Disasters for Agency	
Show Disasters for Agency		Add Agency for Disaster	
Delete Agency for Disaster			

disaster_id	date	type	severity
62	2015-02-10	Heatwave	3
63	2017-03-15	Tornado	4
64	2018-04-20	Earthquake	5
65	2019-05-10	Flood	1
66	2020-06-25	Cyclone	4
67	2021-07-15	Landslide	5
68	2019-08-30	Heatwave	2
69	2020-09-10	Flood	3
70	2021-10-05	Earthquake	5
71	2005-06-10	Cyclone	4
72	2006-07-15	Flood	2
73	2007-08-20	Earthquake	5
74	2008-09-25	Agartala	
75	2009-10-15	Srinagar	
76	2010-11-20	Landslide	
77	2011-12-10	Flood	
78	2012-01-20	Tornado	
79	2013-02-15	Heatwave	
80	2014-03-05	Flood	
81	2015-04-25	Cyclone	
82	2016-05-15	Flood	
83	2017-06-01	Tornado	
84	2018-07-10	Heatwave	
85	2019-08-15	Cyclone	
	Ahmedabad		

Message: Disaster ID does not exist! OK

Windows Taskbar: Type here to search, File Explorer, Microsoft Edge, Google Chrome, File Manager, Task View, Taskbar icons, Date: 09-11-2024, Time: 15:02, Language: ENG

3. Delete Disaster

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:	85	Agency ID:	
Date (YYYY-MM-DD):	2019-08-15	Agency Name:	
Location:	Ahmedabad	Type:	
Type:	Cyclone	Role:	
Severity (1-5):	2		

Add Disaster		Update Disaster	
Delete Disaster		Add Agency	
Update Agency		Delete Agency	
Show Agencies for Disaster		Show Disasters for Agency	
Show Disasters for Agency		Add Agency for Disaster	
Delete Agency for Disaster			

disaster_id	date	type	severity
62	2015-02-10	Heatwave	3
63	2017-03-15	Tornado	4
64	2018-04-20	Earthquake	5
65	2019-05-10	Flood	1
66	2020-06-25	Cyclone	4
67	2021-07-15	Landslide	5
68	2019-08-30	Heatwave	2
69	2020-09-10	Flood	3
70	2021-10-05	Earthquake	5
71	2005-06-10	Cyclone	4
72	2006-07-15	Flood	2
73	2007-08-20	Earthquake	5
74	2008-09-25	Srinagar	
75	2009-10-15	Landslide	
76	2010-11-20	Flood	
77	2011-12-10	Tornado	
78	2012-01-20	Heatwave	
79	2013-02-15	Flood	
80	2014-03-05	Cyclone	
81	2015-04-25	Flood	
82	2016-05-15	Tornado	
83	2017-06-01	Heatwave	
84	2018-07-10	Cyclone	
85	2019-08-15	Ahmedabad	

Message: Disaster deleted successfully! OK

Windows Taskbar: Type here to search, File Explorer, Microsoft Edge, Google Chrome, File Manager, Task View, Taskbar icons, Date: 09-11-2024, Time: 15:03, Language: ENG

Disaster and Agency Management

Disaster Details			Agency Details		
Disaster ID:	85	Agency ID:			
Date (YYYY-MM-DD):	2019-08-15	Agency Name:			
Location:	Ahmedabad	Type:			
Type:	Cyclone	Role:			
Severity (1-5):	2				
Add Disaster		Update Disaster	Delete Disaster	Add Agency	
Update Agency		Delete Agency	Show Disasters	Show Agencies	
Show Agencies for Disaster		Show Disasters for Agency	Add Agency for Disaster	Delete Agency for Disaster	
disaster_id	date	location	type	severity	
61	2015-01-25	Kochi	Flood	4	
62	2016-02-10	Guwahati	Heatwave	3	
63	2017-03-15	Gurgaon	Tornado	4	
64	2018-04-20	Nagpur	Earthquake	5	
65	2019-05-10	Gwalior	Flood	1	
66	2020-06-25	Mangaluru	Cyclone	4	
67	2021-07-15	Bengaluru	Landslide	5	
68	2019-08-30	Jodhpur	Heatwave	2	
69	2020-09-10	Kanpur	Flood	3	
70	2021-10-05	Kolkata	Earthquake	5	
71	2005-06-10	Bhuj	Cyclone	4	
72	2006-07-15	Mysore	Flood	2	
73	2007-08-20	Agartala	Earthquake	5	
74	2008-09-25	Srinagar	Landslide	3	
75	2009-10-15	Delhi	Flood	4	
76	2010-11-20	Bhopal	Tornado	2	
77	2011-12-10	Indore	Heatwave	3	
78	2012-01-20	Lucknow	Flood	5	
79	2013-02-15	Nagpur	Earthquake	4	
80	2014-03-05	Surat	Cyclone	5	
81	2015-04-25	Puducherry	Flood	1	
82	2016-05-15	Nashik	Tornado	3	
83	2017-06-01	Patiala	Flood	4	
84	2018-07-10	Visakhapatnam	Earthquake	5	

Disaster and Agency Management

Disaster Details			Agency Details		
Disaster ID:	90	Agency ID:			
Date (YYYY-MM-DD):	2019-08-15	Agency Name:			
Location:	Ahmedabad	Type:			
Type:	Cyclone	Role:			
Severity (1-5):	2				
Add Disaster		Update Disaster	Delete Disaster	Add Agency	
Update Agency		Delete Agency	Show Disasters	Show Agencies	
Show Agencies for Disaster		Show Disasters for Agency	Add Agency for Disaster	Delete Agency for Disaster	
disaster_id	date	location	type	severity	
61	2015-01-25	K	Flood	4	
62	2016-02-10	G	Heatwave	3	
63	2017-03-15	G	Tornado	4	
64	2018-04-20	N	Earthquake	5	
65	2019-05-10	G	Flood	1	
66	2020-06-25	Mangaluru	Cyclone	4	
67	2021-07-15	Bengaluru	Landslide	5	
68	2019-08-30	Jodhpur	Heatwave	2	
69	2020-09-10	Kanpur	Flood	3	
70	2021-10-05	Kolkata	Earthquake	5	
71	2005-06-10	Bhuj	Cyclone	4	
72	2006-07-15	Mysore	Flood	2	
73	2007-08-20	Agartala	Earthquake	5	
74	2008-09-25	Srinagar	Landslide	3	
75	2009-10-15	Delhi	Flood	4	
76	2010-11-20	Bhopal	Tornado	2	
77	2011-12-10	Indore	Heatwave	3	
78	2012-01-20	Lucknow	Flood	5	
79	2013-02-15	Nagpur	Earthquake	4	
80	2014-03-05	Surat	Cyclone	5	
81	2015-04-25	Puducherry	Flood	1	
82	2016-05-15	Nashik	Tornado	3	
83	2017-06-01	Patiala	Flood	4	
84	2018-07-10	Visakhapatnam	Earthquake	5	

Message

i Disaster ID does not exist!

OK

4. Show all Disasters

Disaster and Agency Management

Disaster Details			Agency Details		
Disaster ID:			Agency ID:		
Date (YYYY-MM-DD):			Agency Name:		
Location:			Type:		
Type:			Role:		
Severity (1-5):					
Add Disaster		Update Disaster	Delete Disaster	Add Agency	
Update Agency		Delete Agency	Show Disasters	Show Agencies	
Show Agencies for Disaster		Show Disasters for Agency	Add Agency for Disaster	Delete Agency for Disaster	
disaster_id	date	location	type	severity	
61	2015-01-25	Kochi	Flood	4	
62	2016-02-10	Guwahati	Heatwave	3	
63	2017-03-15	Gurgaon	Tornado	4	
64	2018-04-20	Nagpur	Earthquake	5	
65	2019-05-10	Gwalior	Flood	1	
66	2020-06-25	Mangalore	Cyclone	4	
67	2021-07-15	Bengaluru	Landslide	5	
68	2019-08-30	Jodhpur	Heatwave	2	
69	2020-09-10	Kanpur	Flood	3	
70	2021-10-05	Kolkata	Earthquake	5	
71	2005-06-10	Bhuj	Cyclone	4	
72	2006-07-15	Mysore	Flood	2	
73	2007-08-20	Agartala	Earthquake	5	
74	2008-09-25	Srinagar	Landslide	3	
75	2009-10-15	Delhi	Flood	4	
76	2010-11-20	Bhopal	Tornado	2	
77	2011-12-10	Indore	Heatwave	3	
78	2012-01-20	Lucknow	Flood	5	
79	2013-02-15	Nagpur	Earthquake	4	
80	2014-03-05	Surat	Cyclone	5	
81	2015-04-25	Puducherry	Flood	1	
82	2016-05-15	Nashik	Tornado	3	
83	2017-06-01	Patiala	Flood	4	
84	2018-07-10	Visakhapatnam	Earthquake	5	

5. Add Agency

Disaster and Agency Management

Disaster Details			Agency Details		
Disaster ID:			Agency ID:		
Date (YYYY-MM-DD):			Agency Name:		
Location:			Type:		
Type:			Role:		
Severity (1-5):					
Add Disaster		Update Disaster	Delete Disaster	Add Agency	
Update Agency		Delete Agency	Show Disasters	Show Agencies	
Show Agencies for Disaster		Show Disasters for Agency	Add Agency for Disaster	Delete Agency for Disaster	
agency_id	name	Message	type	role	
1059	International Organization for Migration	Agency added successfully.	International	Migration support in crises	
1060	Mennonite Central Committee		International	Disaster response and support	
1061	Relief International		International	Emergency response in disasters	
1062	ShelterBox		International	Emergency shelter support	
1063	Catholic Relief Services		International	Emergency relief efforts	
1064	NetHope		International	Technology support in crises	
1065	Friends of the Earth		International	Environmental advocacy and recovery	
1066	Institute for Disaster Management		International	Research on disaster management	
1067	International Council of Voluntary Agencies		International	Coordination of NGOs in disasters	
1068	InterAction		International	Coalition of NGOs for humanitarian aid	
1069	Save the Children		International	Child welfare and humanitarian aid	
1070	Women for Women International		International	Support for women in conflict zones	
1071	Global Network for Disaster Reduction		International	Disaster risk reduction strategies	
1072	Foundation for Environmental Education		International	Environmental education in crises	
1073	ActionAidIndia		Government	Humanitarian aid and social justice	
1074	World Federation of United Nations Associations		International	Advocacy for global cooperation	
1075	International Institute for Disaster Risk Reduction		International	Disaster risk reduction strategies	
1076	World Disaster Preparedness Center		International	Disaster preparedness training centre	
1077	International Society for Disaster Risk Reduction		International	Research on disaster management	
1078	International Council for Science		International	Science for global challenges	
1079	Global Disaster Risk Reduction Network		International	Promoting disaster risk reduction	
1080	European Network for Humanitarian Action		International	Coordination of humanitarian response	
1081	International Disaster Committee		International	Emergency response	
1082	Global Health Security Agenda		International	Strengthening health security	

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:		Agency ID:	1083
Date (YYYY-MM-DD):		Agency Name:	agency1
Location:		Type:	government
Type:		Role:	Disaster response and support
Severity (1-5):			
Add Disaster		Update Disaster	
Update Agency		Delete Agency	
Show Agencies for Disaster		Show Disasters for Agency	
		Add Agency for Disaster	
		Delete Agency for Disaster	
agency_id	name	type	role
1080	Mennonite Central Committee	International	Disaster response and support
1081	Relief International	International	Emergency response in disasters
1082	ShelterBox	International	Emergency shelter support
1083	Catholic Relief Services	International	Emergency relief efforts
1084	NetHope	International	Technology support in crises
1085	Friends of the Earth	International	Environmental advocacy and recovery
1086	Institute for Disaster Management	International	Research on disaster management
1087	International Council of Voluntary Agencies	International	Coordination of NGOs in disasters
1088	InterAction	International	Coalition of NGOs for humanitarian aid
1089	Save the Children	International	Child welfare and humanitarian aid
1070	Women for Women International	International	Support for women in conflict zones
1071	Global Network for Disaster Reduction	International	Disaster risk reduction strategies
1072	Foundation for Environmental Education	International	Environmental education in crises
1073	ActionAidIndia	Government	Humanitarian aid and social justice
1074	World Federation of United Nations Associations	International	Advocacy for global cooperation
1075	International Institute for Disaster Risk Reduction	International	Disaster risk reduction strategies
1076	World Disaster Preparedness Center	International	Disaster preparedness training centre
1077	International Society for Disaster Risk Reduction	International	Research on disaster management
1078	International Council for Science	International	Science for global challenges
1079	Global Disaster Risk Reduction Network	International	Promoting disaster risk reduction
1080	European Network for Humanitarian Action	International	Coordination of humanitarian response
1081	International Disaster Committee	International	Emergency response
1082	Global Health Security Agenda	International	Strengthening health security
1083	agency1	government	Disaster response and support

6. Update Agency

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:		Agency ID:	1083
Date (YYYY-MM-DD):		Agency Name:	agency_test
Location:		Type:	government
Type:		Role:	Disaster response and support
Severity (1-5):			
Add Disaster		Update Disaster	
Update Agency		Delete Agency	
Show Agencies for Disaster		Show Disasters for Agency	
		Add Agency for Disaster	
		Delete Agency for Disaster	
agency_id	name	type	role
1080	Mennonite Central Committee	International	Disaster response and support
1081	Relief International	International	Emergency response in disasters
1082	ShelterBox	International	Emergency shelter support
1083	Catholic Relief Services	International	Emergency relief efforts
1084	NetHope	International	Technology support in crises
1085	Friends of the Earth	International	Environmental advocacy and recovery
1086	Institute for Disaster Management	International	Research on disaster management
1087	International Council of Voluntary Agencies	International	Coordination of NGOs in disasters
1088	InterAction	International	Coalition of NGOs for humanitarian aid
1089	Save the Children	International	Child welfare and humanitarian aid
1070	Women for Women International	International	Support for women in conflict zones
1071	Global Network for Disaster Reduction	International	Disaster risk reduction strategies
1072	Foundation for Environmental Education	International	Environmental education in crises
1073	ActionAidIndia	Government	Humanitarian aid and social justice
1074	World Federation of United Nations Associations	International	Advocacy for global cooperation
1075	International Institute for Disaster Risk Reduction	International	Disaster risk reduction strategies
1076	World Disaster Preparedness Center	International	Disaster preparedness training centre
1077	International Society for Disaster Risk Reduction	International	Research on disaster management
1078	International Council for Science	International	Science for global challenges
1079	Global Disaster Risk Reduction Network	International	Promoting disaster risk reduction
1080	European Network for Humanitarian Action	International	Coordination of humanitarian response
1081	International Disaster Committee	International	Emergency response
1082	Global Health Security Agenda	International	Strengthening health security
1083	agency1	government	Disaster response and support

Message

Agency updated successfully!

OK

Disaster Details		Agency Details																																																																																																			
Disaster ID:		Agency ID:	1083																																																																																																		
Date (YYYY-MM-DD):		Agency Name:	agency_test																																																																																																		
Location:		Type:	government																																																																																																		
Type:		Role:	Disaster response and support																																																																																																		
Severity (1-5):																																																																																																					
Add Disaster	Update Disaster	Delete Disaster	Add Agency																																																																																																		
Update Agency	Delete Agency	Show Disasters	Show Agencies																																																																																																		
Show Agencies for Disaster	Show Disasters for Agency	Add Agency for Disaster	Delete Agency for Disaster																																																																																																		
<table border="1"> <thead> <tr> <th>agency_id</th> <th>name</th> <th>type</th> <th>role</th> </tr> </thead> <tbody> <tr><td>TU6U</td><td>Mennonite Central Committee</td><td>International</td><td>Disaster response and support</td></tr> <tr><td>1061</td><td>Relief International</td><td>International</td><td>Emergency response in disasters</td></tr> <tr><td>1062</td><td>ShelterBox</td><td>International</td><td>Emergency shelter support</td></tr> <tr><td>1063</td><td>Catholic Relief Services</td><td>International</td><td>Emergency relief efforts</td></tr> <tr><td>1064</td><td>NetHope</td><td>International</td><td>Technology support in crises</td></tr> <tr><td>1065</td><td>Friends of the Earth</td><td>International</td><td>Environmental advocacy and recovery</td></tr> <tr><td>1066</td><td>Institute for Disaster Management</td><td>International</td><td>Research on disaster management</td></tr> <tr><td>1067</td><td>International Council of Voluntary Agencies</td><td>International</td><td>Coordination of NGOs in disasters</td></tr> <tr><td>1068</td><td>InterAction</td><td>International</td><td>Coalition of NGOs for humanitarian aid</td></tr> <tr><td>1069</td><td>Save the Children</td><td>International</td><td>Child welfare and humanitarian aid</td></tr> <tr><td>1070</td><td>Women for Women International</td><td>International</td><td>Support for women in conflict zones</td></tr> <tr><td>1071</td><td>Global Network for Disaster Reduction</td><td>International</td><td>Disaster risk reduction strategies</td></tr> <tr><td>1072</td><td>Foundation for Environmental Education</td><td>International</td><td>Environmental education in crises</td></tr> <tr><td>1073</td><td>ActionAidIndia</td><td>Government</td><td>Humanitarian aid and social justice</td></tr> <tr><td>1074</td><td>World Federation of United Nations Associations</td><td>International</td><td>Advocacy for global cooperation</td></tr> <tr><td>1075</td><td>International Institute for Disaster Risk Reduction</td><td>International</td><td>Disaster risk reduction strategies</td></tr> <tr><td>1076</td><td>World Disaster Preparedness Center</td><td>International</td><td>Disaster preparedness training centre</td></tr> <tr><td>1077</td><td>International Society for Disaster Risk Reduction</td><td>International</td><td>Research on disaster management</td></tr> <tr><td>1078</td><td>International Council for Science</td><td>International</td><td>Science for global challenges</td></tr> <tr><td>1079</td><td>Global Disaster Risk Reduction Network</td><td>International</td><td>Promoting disaster risk reduction</td></tr> <tr><td>1080</td><td>European Network for Humanitarian Action</td><td>International</td><td>Coordination of humanitarian response</td></tr> <tr><td>1081</td><td>International Disaster Committee</td><td>International</td><td>Emergency response</td></tr> <tr><td>1082</td><td>Global Health Security Agenda</td><td>International</td><td>Strengthening health security</td></tr> <tr><td>1083</td><td>agency_test</td><td>government</td><td>Disaster response and support</td></tr> </tbody> </table>		agency_id	name	type	role	TU6U	Mennonite Central Committee	International	Disaster response and support	1061	Relief International	International	Emergency response in disasters	1062	ShelterBox	International	Emergency shelter support	1063	Catholic Relief Services	International	Emergency relief efforts	1064	NetHope	International	Technology support in crises	1065	Friends of the Earth	International	Environmental advocacy and recovery	1066	Institute for Disaster Management	International	Research on disaster management	1067	International Council of Voluntary Agencies	International	Coordination of NGOs in disasters	1068	InterAction	International	Coalition of NGOs for humanitarian aid	1069	Save the Children	International	Child welfare and humanitarian aid	1070	Women for Women International	International	Support for women in conflict zones	1071	Global Network for Disaster Reduction	International	Disaster risk reduction strategies	1072	Foundation for Environmental Education	International	Environmental education in crises	1073	ActionAidIndia	Government	Humanitarian aid and social justice	1074	World Federation of United Nations Associations	International	Advocacy for global cooperation	1075	International Institute for Disaster Risk Reduction	International	Disaster risk reduction strategies	1076	World Disaster Preparedness Center	International	Disaster preparedness training centre	1077	International Society for Disaster Risk Reduction	International	Research on disaster management	1078	International Council for Science	International	Science for global challenges	1079	Global Disaster Risk Reduction Network	International	Promoting disaster risk reduction	1080	European Network for Humanitarian Action	International	Coordination of humanitarian response	1081	International Disaster Committee	International	Emergency response	1082	Global Health Security Agenda	International	Strengthening health security	1083	agency_test	government	Disaster response and support
agency_id	name	type	role																																																																																																		
TU6U	Mennonite Central Committee	International	Disaster response and support																																																																																																		
1061	Relief International	International	Emergency response in disasters																																																																																																		
1062	ShelterBox	International	Emergency shelter support																																																																																																		
1063	Catholic Relief Services	International	Emergency relief efforts																																																																																																		
1064	NetHope	International	Technology support in crises																																																																																																		
1065	Friends of the Earth	International	Environmental advocacy and recovery																																																																																																		
1066	Institute for Disaster Management	International	Research on disaster management																																																																																																		
1067	International Council of Voluntary Agencies	International	Coordination of NGOs in disasters																																																																																																		
1068	InterAction	International	Coalition of NGOs for humanitarian aid																																																																																																		
1069	Save the Children	International	Child welfare and humanitarian aid																																																																																																		
1070	Women for Women International	International	Support for women in conflict zones																																																																																																		
1071	Global Network for Disaster Reduction	International	Disaster risk reduction strategies																																																																																																		
1072	Foundation for Environmental Education	International	Environmental education in crises																																																																																																		
1073	ActionAidIndia	Government	Humanitarian aid and social justice																																																																																																		
1074	World Federation of United Nations Associations	International	Advocacy for global cooperation																																																																																																		
1075	International Institute for Disaster Risk Reduction	International	Disaster risk reduction strategies																																																																																																		
1076	World Disaster Preparedness Center	International	Disaster preparedness training centre																																																																																																		
1077	International Society for Disaster Risk Reduction	International	Research on disaster management																																																																																																		
1078	International Council for Science	International	Science for global challenges																																																																																																		
1079	Global Disaster Risk Reduction Network	International	Promoting disaster risk reduction																																																																																																		
1080	European Network for Humanitarian Action	International	Coordination of humanitarian response																																																																																																		
1081	International Disaster Committee	International	Emergency response																																																																																																		
1082	Global Health Security Agenda	International	Strengthening health security																																																																																																		
1083	agency_test	government	Disaster response and support																																																																																																		

7. Delete agency

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:		Agency ID:	1083
Date (YYYY-MM-DD):		Agency Name:	agency_test
Location:		Type:	government
Type:		Role:	Disaster response and support
Severity (1-5):			

Add Disaster	Update Disaster	Delete Disaster	Add Agency
Update Agency	Delete Agency	Show Disasters	Show Agencies
Show Agencies for Disaster	Show Disasters for Agency	Add Agency for Disaster	Delete Agency for Disaster
agency_id	name	type	role
1050	Mennonite Central Committee	Disaster response and support	Emergency response in disasters
1051	Relief International	Emergency shelter support	Emergency relief efforts
1052	ShelterBox	Technology support in crises	Environmental advocacy and recovery
1053	Catholic Relief Services	Research on disaster management	Coordination of NGOs in disasters
1054	NetHope	Coalition of NGOs for humanitarian aid	Child welfare and humanitarian aid
1055	Friends of the Earth	Support for women in conflict zones	Disaster risk reduction strategies
1056	Institute for Disaster Management	Environmental education in crises	Humanitarian aid and social justice
1057	International Council of Voluntary Agencies	Advocacy for global cooperation	Disaster risk reduction strategies
1058	InterAction	Disaster preparedness training centre	Research on disaster management
1059	Save the Children	Science for global challenges	Promoting disaster risk reduction
1070	Women for Women International	International	Coordination of humanitarian response
1071	Global Network for Disaster Reduction	International	Emergency response
1072	Foundation for Environmental Education	International	Strengthening health security
1073	ActionAidIndia	Government	Disaster response and support
1074	World Federation of United Nations Associations	International	
1075	International Institute for Disaster Risk Reduction	International	
1076	World Disaster Preparedness Center	International	
1077	International Society for Disaster Risk Reduction	International	
1078	International Council for Science	International	
1079	Global Disaster Risk Reduction Network	International	
1080	European Network for Humanitarian Action	International	
1081	International Disaster Committee	International	
1082	Global Health Security Agenda	International	
1083	agency test	government	

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:		Agency ID:	1083
Date (YYYY-MM-DD):		Agency Name:	agency_test
Location:		Type:	government
Type:		Role:	Disaster response and support
Severity (1-5):			
Add Disaster		Update Disaster	Delete Disaster
Update Agency		Delete Agency	Show Disasters
Show Agencies for Disaster		Show Disasters for Agency	Add Agency for Disaster
			Delete Agency for Disaster
agency_id	name	type	role
1059	International Organization for Migration	International	Migration support in crises
1060	Mennonite Central Committee	International	Disaster response and support
1061	Relief International	International	Emergency response in disasters
1062	ShelterBox	International	Emergency shelter support
1063	Catholic Relief Services	International	Emergency relief efforts
1064	NetHope	International	Technology support in crises
1065	Friends of the Earth	International	Environmental advocacy and recovery
1066	Institute for Disaster Management	International	Research on disaster management
1067	International Council of Voluntary Agencies	International	Coordination of NGOs in disasters
1068	InterAction	International	Coalition of NGOs for humanitarian aid
1069	Save the Children	International	Child welfare and humanitarian aid
1070	Women for Women International	International	Support for women in conflict zones
1071	Global Network for Disaster Reduction	International	Disaster risk reduction strategies
1072	Foundation for Environmental Education	International	Environmental education in crises
1073	ActionAidIndia	Government	Humanitarian aid and social justice
1074	World Federation of United Nations Associations	International	Advocacy for global cooperation
1075	International Institute for Disaster Risk Reduction	International	Disaster risk reduction strategies
1076	World Disaster Preparedness Center	International	Disaster preparedness training centre
1077	International Society for Disaster Risk Reduction	International	Research on disaster management
1078	International Council for Science	International	Science for global challenges
1079	Global Disaster Risk Reduction Network	International	Promoting disaster risk reduction
1080	European Network for Humanitarian Action	International	Coordination of humanitarian response
1081	International Disaster Committee	International	Emergency response
1082	Global Health Security Agenda	International	Strengthening health security

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:		Agency ID:	1090
Date (YYYY-MM-DD):		Agency Name:	agency_test
Location:		Type:	government
Type:		Role:	Disaster response and support
Severity (1-5):			
Add Disaster		Update Disaster	Delete Disaster
Update Agency		Delete Agency	Show Disasters
Show Agencies for Disaster		Show Disasters for Agency	Add Agency for Disaster
			Delete Agency for Disaster
agency_id	name	type	role
1059	International Organization for Migration	International	Migration support in crises
1060	Mennonite Central Committee	International	Disaster response and support
1061	Relief International	International	Emergency response in disasters
1062	ShelterBox	International	Emergency shelter support
1063	Catholic Relief Services	International	Emergency relief efforts
1064	NetHope	International	Technology support in crises
1065	Friends of the Earth	International	Environmental advocacy and recovery
1066	Institute for Disaster Management	International	Research on disaster management
1067	International Council of Voluntary Agencies	International	Coordination of NGOs in disasters
1068	InterAction	International	Coalition of NGOs for humanitarian aid
1069	Save the Children	International	Child welfare and humanitarian aid
1070	Women for Women International	International	Support for women in conflict zones
1071	Global Network for Disaster Reduction	International	Disaster risk reduction strategies
1072	Foundation for Environmental Education	International	Environmental education in crises
1073	ActionAidIndia	Government	Humanitarian aid and social justice
1074	World Federation of United Nations Associations	International	Advocacy for global cooperation
1075	International Institute for Disaster Risk Reduction	International	Disaster risk reduction strategies
1076	World Disaster Preparedness Center	International	Disaster preparedness training centre
1077	International Society for Disaster Risk Reduction	International	Research on disaster management
1078	International Council for Science	International	Science for global challenges
1079	Global Disaster Risk Reduction Network	International	Promoting disaster risk reduction
1080	European Network for Humanitarian Action	International	Coordination of humanitarian response
1081	International Disaster Committee	International	Emergency response
1082	Global Health Security Agenda	International	Strengthening health security

Message

Agency ID does not exist!

8. Show all agencies

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:		Agency ID:	1090
Date (YYYY-MM-DD):		Agency Name:	agency_test
Location:		Type:	government
Type:		Role:	Disaster response and support
Severity (1-5):			

Add Disaster	Update Disaster	Delete Disaster	Add Agency
Update Agency	Delete Agency	Show Disasters	Show Agencies
Show Agencies for Disaster	Show Disasters for Agency	Add Agency for Disaster	Delete Agency for Disaster

agency_id	name	type	role
1059	International Organization for Migration	International	Migration support in crises
1060	Mennonite Central Committee	International	Disaster response and support
1061	Relief International	International	Emergency response in disasters
1062	ShelterBox	International	Emergency shelter support
1063	Catholic Relief Services	International	Emergency relief efforts
1064	NetHope	International	Technology support in crises
1065	Friends of the Earth	International	Environmental advocacy and recovery
1066	Institute for Disaster Management	International	Research on disaster management
1067	International Council of Voluntary Agencies	International	Coordination of NGOs in disasters
1068	InterAction	International	Coalition of NGOs for humanitarian aid
1069	Save the Children	International	Child welfare and humanitarian aid
1070	Women for Women International	International	Support for women in conflict zones
1071	Global Network for Disaster Reduction	International	Disaster risk reduction strategies
1072	Foundation for Environmental Education	International	Environmental education in crises
1073	ActionAidIndia	Government	Humanitarian aid and social justice
1074	World Federation of United Nations Associations	International	Advocacy for global cooperation
1075	International Institute for Disaster Risk Reduction	International	Disaster risk reduction strategies
1076	World Disaster Preparedness Center	International	Disaster preparedness training centre
1077	International Society for Disaster Risk Reduction	International	Research on disaster management
1078	International Council for Science	International	Science for global challenges
1079	Global Disaster Risk Reduction Network	International	Promoting disaster risk reduction
1080	European Network for Humanitarian Action	International	Coordination of humanitarian response
1081	International Disaster Committee	International	Emergency response
1082	Global Health Security Agenda	International	Strengthening health security

Type here to search ENG 15:09 09-11-2024

9. Add agency for Disaster

Disaster and Agency Management

Disaster Details		Agency Details	
Disaster ID:	84	Agency ID:	1083
Date (YYYY-MM-DD):		Agency Name:	agency_test
Location:		Type:	government
Type:		Role:	Disaster response and support
Severity (1-5):			

Add Disaster	Update Disaster	Delete Disaster	Add Agency
Update Agency	Delete Agency	Show Disasters	Show Agencies
Show Agencies for Disaster	Show Disasters for Agency	Add Agency for Disaster	Delete Agency for Disaster

agency_id	name	type	role
1059	International Organization for Migration	International	Migration support in crises
1060	Mennonite Central Committee	International	Disaster response and support
1061	Relief International	International	Emergency response in disasters
1062	ShelterBox	International	Emergency shelter support
1063	Catholic Relief Services	International	Emergency relief efforts
1064	NetHope	International	Technology support in crises
1065	Friends of the Earth	International	Environmental advocacy and recovery
1066	Institute for Disaster Management	International	Research on disaster management
1067	International Council of Voluntary Agencies	International	Coordination of NGOs in disasters
1068	InterAction	International	Coalition of NGOs for humanitarian aid
1069	Save the Children	International	Child welfare and humanitarian aid
1070	Women for Women International	International	Support for women in conflict zones
1071	Global Network for Disaster Reduction	International	Disaster risk reduction strategies
1072	Foundation for Environmental Education	International	Environmental education in crises
1073	ActionAidIndia	Government	Humanitarian aid and social justice
1074	World Federation of United Nations Associations	International	Advocacy for global cooperation
1075	International Institute for Disaster Risk Reduction	International	Disaster risk reduction strategies
1076	World Disaster Preparedness Center	International	Disaster preparedness training centre
1077	International Society for Disaster Risk Reduction	International	Research on disaster management
1078	International Council for Science	International	Science for global challenges
1079	Global Disaster Risk Reduction Network	International	Promoting disaster risk reduction
1080	European Network for Humanitarian Action	International	Coordination of humanitarian response
1081	International Disaster Committee	International	Emergency response
1082	Global Health Security Agenda	International	Strengthening health security

Message Agency added for disaster.

Disaster and Agency Management

Disaster Details				Agency Details				
Disaster ID:	84	Agency ID:	1083					
Date (YYYY-MM-DD):		Agency Name:	agency_test					
Location:		Type:	government					
Type:		Role:	Disaster response and support					
Severity (1-5):								
Add Disaster		Update Disaster		Delete Disaster		Add Agency		
Update Agency		Delete Agency		Show Disasters		Show Agencies		
Show Agencies for Disaster		Show Disasters for Agency		Add Agency for Disaster		Delete Agency for Disaster		
Disaster ID	Date	Location	Type	Severity	Agency ID	Agency Name	Agency Type	Agency Role
84	2018-07-10	Visakhapatnam	Earthquake	5	1003	Indian Meteorological De...	Earthquake	Weather forecasting and Humanitarian assistance...
84	2018-07-10	Visakhapatnam	Earthquake	5	1008	Indian Navy	Earthquake	Humanitarian assistance...
84	2018-07-10	Visakhapatnam	Earthquake	5	1013	United Nations Office for t...	Earthquake	Coordination of disaster r...
84	2018-07-10	Visakhapatnam	Earthquake	5	1083	agency_test	Earthquake	Disaster response and s...

10. Show Agencies which worked in given disaster

Disaster and Agency Management

Disaster Details				Agency Details				
Disaster ID:	10	Agency ID:						
Date (YYYY-MM-DD):		Agency Name:						
Location:		Type:						
Type:		Role:						
Severity (1-5):								
Add Disaster		Update Disaster		Delete Disaster		Add Agency		
Update Agency		Delete Agency		Show Disasters		Show Agencies		
Show Agencies for Disaster		Show Disasters for Agency		Add Agency for Disaster		Delete Agency for Disaster		
Disaster ID	Date	Location	Type	Severity	Agency ID	Agency Name	Agency Type	Agency Role
10	2012-10-10	Patna	Flood	3	1002	State Disaster Response...	Flood	State-level disaster resp...
10	2012-10-10	Patna	Flood	3	1009	Indian Air Force	Flood	Airlift and rescue operatio...
10	2012-10-10	Patna	Flood	3	1014	Food and Agriculture Org...	Flood	Food security in emergen...

11. show disaster for which an agency worked

Disaster and Agency Management

Disaster Details				Agency Details					
Disaster ID:				Agency ID:	1030				
Date (YYYY-MM-DD):				Agency Name:					
Location:				Type:					
Type:				Role:					
Severity (1-5):									
Add Disaster		Update Disaster			Delete Disaster		Add Agency		
Update Agency		Delete Agency			Show Disasters		Show Agencies		
Show Agencies for Disaster		Show Disasters for Agency			Add Agency for Disaster		Delete Agency for Disaster		
Agency ID	Agency Name	Agency Type	Agency Role	Disaster ID	Date	Location	Type	Severity	
1030	Mercy Corps	International	Economic recovery support	52	2004-04-15	Guwahati	Flood	3	
1030	Mercy Corps	International	Economic recovery support	67	2021-07-15	Bengaluru	Landslide	5	

12. Delete agency for Disaster

Disaster and Agency Management

Disaster Details				Agency Details					
Disaster ID:	67			Agency ID:	1030				
Date (YYYY-MM-DD):				Agency Name:					
Location:				Type:					
Type:				Role:					
Severity (1-5):									
Add Disaster		Update Disaster			Delete Disaster		Add Agency		
Update Agency		Delete Agency			Show Disasters		Show Agencies		
Show Agencies for Disaster		Show Disasters for Agency			Add Agency for Disaster		Delete Agency for Disaster		
Agency ID	Agency Name	Agency Type	Agency Role	Disaster ID	Date	Location	Type	Severity	
1030	Mercy Corps	International	Economic recover	67	2004-04-15	Guwahati	Flood	3	
1030	Mercy Corps	International	Economic recover	67	2021-07-15	Bengaluru	Landslide	5	

Message

OK

i Agency deleted from disaster.

Disaster and Agency Management

Disaster Details				Agency Details				
Disaster ID:	67	Agency ID:	1030					
Date (YYYY-MM-DD):		Agency Name:						
Location:		Type:						
Type:		Role:						
Severity (1-5):								
Add Disaster		Update Disaster		Delete Disaster		Add Agency		
Update Agency		Delete Agency		Show Disasters		Show Agencies		
Show Agencies for Disaster		Show Disasters for Agency		Add Agency for Disaster		Delete Agency for Disaster		
Agency ID	Agency Name	Agency Type	Agency Role	Disaster ID	Date	Location	Type	Severity
1030	Mercy Corps	International	Economic recovery support	52	2004-04-15	Guwahati	Flood	3

Disaster and Agency Management

Disaster Details				Agency Details				
Disaster ID:	67	Agency ID:	1030					
Date (YYYY-MM-DD):		Agency Name:						
Location:		Type:						
Type:		Role:						
Severity (1-5):								
Add Disaster		Update Disaster		Delete Disaster		Add Agency		
Update Agency		Delete Agency		Show Disasters		Show Agencies		
Show Agencies for Disaster		Show Disasters for Agency		Add Agency for Disaster		Delete Agency for Disaster		
Disaster ID	Date	Location	Type	Severity	Agency ID	Agency Name	Agency Type	Agency Role
67	2021-07-15	Bengaluru	Landslide	5	1004	National Remote Sensin...	Landslide	Satellite data for disaster ...
67	2021-07-15	Bengaluru	Landslide	5	1011	World Health Organization	Landslide	Health support during dis...
67	2021-07-15	Bengaluru	Landslide	5	1068	InterAction	Landslide	Coalition of NGOs for hu...

3. Code:

```
import javax.swing.*;  
  
import javax.swing.table.DefaultTableModel;  
import java.awt.*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.sql.*;  
import java.util.Vector;  
  
public class DisasterAgencyApp extends JFrame {  
    private JTextField disasterIdField, dateField, locationField,  
disasterTypeField, severityField;  
    private JTextField agencyIdField, agencyNameField, agencyTypeField,  
roleField;  
    private JTable dataTable;  
    private Connection connection;  
  
    public DisasterAgencyApp() {  
        setTitle("Disaster and Agency Management");  
        setSize(1000, 500);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        setLocationRelativeTo(null);  
  
        // Establish database connection  
        connectToDatabase();  
  
        // Layout configuration  
        setLayout(new BorderLayout());  
        JPanel formPanel = new JPanel(new GridLayout(1, 2));  
        add(formPanel, BorderLayout.NORTH);  
  
        // Disaster Panel  
        JPanel disasterPanel = new JPanel(new GridLayout(6, 2));  
        disasterPanel.setBorder(BorderFactory.createTitledBorder("Disaster  
Details"));  
        disasterIdField = new JTextField();  
        dateField = new JTextField();  
        locationField = new JTextField();  
        disasterTypeField = new JTextField();  
        severityField = new JTextField();  
  
        disasterPanel.add(new JLabel("Disaster ID:"));
```

```

disasterPanel.add(disasterIdField);
disasterPanel.add(new JLabel("Date (YYYY-MM-DD):"));
disasterPanel.add(dateField);
disasterPanel.add(new JLabel("Location:"));
disasterPanel.add(locationField);
disasterPanel.add(new JLabel("Type:"));
disasterPanel.add(disasterTypeField);
disasterPanel.add(new JLabel("Severity (1-5):"));
disasterPanel.add(severityField);

// Agency Panel
JPanel agencyPanel = new JPanel(new GridLayout(5, 2));
agencyPanel.setBorder(BorderFactory.createTitledBorder("Agency Details"));
agencyIdField = new JTextField();
agencyNameField = new JTextField();
agencyTypeField = new JTextField();
roleField = new JTextField();

agencyPanel.add(new JLabel("Agency ID:"));
agencyPanel.add(agencyIdField);
agencyPanel.add(new JLabel("Agency Name:"));
agencyPanel.add(agencyNameField);
agencyPanel.add(new JLabel("Type:"));
agencyPanel.add(agencyTypeField);
agencyPanel.add(new JLabel("Role:"));
agencyPanel.add(roleField);

formPanel.add(disasterPanel);
formPanel.add(agencyPanel);

// Button Panel
JPanel buttonPanel = new JPanel(new GridLayout(3, 4));
JButton addDisasterBtn = new JButton("Add Disaster");
JButton updateDisasterBtn = new JButton("Update Disaster");
JButton deleteDisasterBtn = new JButton("Delete Disaster");
JButton showDisastersBtn = new JButton("Show Disasters");

JButton addAgencyBtn = new JButton("Add Agency");
JButton updateAgencyBtn = new JButton("Update Agency");
JButton deleteAgencyBtn = new JButton("Delete Agency");
JButton showAgenciesBtn = new JButton("Show Agencies");

JButton addAgencyForDisasterBtn = new JButton("Add Agency for Disaster");

```

```

        JButton deleteAgencyForDisasterBtn = new JButton("Delete Agency for
Disaster");
        JButton showAgenciesForDisasterBtn = new JButton("Show Agencies for
Disaster");
        JButton showDisastersForAgencyBtn = new JButton("Show Disasters for
Agency");

        addDisasterBtn.addActionListener(e -> addDisaster());
        updateDisasterBtn.addActionListener(e -> updateDisaster());
        deleteDisasterBtn.addActionListener(e -> deleteDisaster());
        addAgencyBtn.addActionListener(e -> addAgency());
        updateAgencyBtn.addActionListener(e -> updateAgency());
        deleteAgencyBtn.addActionListener(e -> deleteAgency());
        showDisastersBtn.addActionListener(e -> showDisasters());
        showAgenciesBtn.addActionListener(e -> showAgencies());
        showAgenciesForDisasterBtn.addActionListener(e ->
showAgenciesForDisaster());
        showDisastersForAgencyBtn.addActionListener(e ->
showDisastersForAgency());
        addAgencyForDisasterBtn.addActionListener(e -> addAgencyForDisaster());
        deleteAgencyForDisasterBtn.addActionListener(e ->
deleteAgencyForDisaster());

        buttonPanel.add(addDisasterBtn);
        buttonPanel.add(updateDisasterBtn);
        buttonPanel.add(deleteDisasterBtn);
        buttonPanel.add(addAgencyBtn);
        buttonPanel.add(updateAgencyBtn);
        buttonPanel.add(deleteAgencyBtn);
        buttonPanel.add(showDisastersBtn);
        buttonPanel.add(showAgenciesBtn);
        buttonPanel.add(showAgenciesForDisasterBtn);
        buttonPanel.add(showDisastersForAgencyBtn);
        buttonPanel.add(addAgencyForDisasterBtn);
        buttonPanel.add(deleteAgencyForDisasterBtn);
        add(buttonPanel, BorderLayout.CENTER);

        // Data Table
        dataTable = new JTable();
        add(new JScrollPane(dataTable), BorderLayout.SOUTH);

        // to make everything is visible and packed
        pack();
        setVisible(true);
    }
}

```

```

private void connectToDatabase() {
    try {
        connection =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/disaster_agency",
"postgres", "password");
        System.out.println("Connected to database.");
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error connecting to database: "
+ e.getMessage());
    }
}

// helper function to clear out all input fields

private void clearInputFields() {
    // Clear all disaster-related fields
    disasterIdField.setText("");
    dateField.setText("");
    locationField.setText("");
    disasterTypeField.setText("");
    severityField.setText("");

    // Clear all agency-related fields
    agencyIdField.setText("");
    agencyNameField.setText("");
    agencyTypeField.setText("");
    roleField.setText("");
}

// CRUD methods for Disaster and Agency
private void addDisaster() {
    try {
        String sql = "INSERT INTO Disaster (disaster_id, date, location,
type, severity) VALUES (?, ?, ?, ?, ?)";
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setInt(1, Integer.parseInt(disasterIdField.getText()));
        statement.setDate(2, java.sql.Date.valueOf(dateField.getText()));
        statement.setString(3, locationField.getText());
        statement.setString(4, disasterTypeField.getText());
        statement.setInt(5, Integer.parseInt(severityField.getText()));
        statement.executeUpdate();
        JOptionPane.showMessageDialog(this, "Disaster added successfully.");
    }

    clearInputFields();
}

```

```

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error adding disaster: " +
e.getMessage());
    }
}

private void updateDisaster() {
    try {
        int disasterId = Integer.parseInt(disasterIdField.getText());

        // Check if the disaster exists
        String checkDisasterSQL = "SELECT * FROM Disaster WHERE disaster_id = ?";
        PreparedStatement checkStatement =
connection.prepareStatement(checkDisasterSQL);
        checkStatement.setInt(1, disasterId);
        ResultSet rs = checkStatement.executeQuery();

        if (!rs.next()) {
            JOptionPane.showMessageDialog(this, "Disaster ID does not
exist!");
            return;
        }

        // Update the Disaster details
        String updateDisasterSQL = "UPDATE Disaster SET date = ?, location = ?
, type = ?, severity = ? WHERE disaster_id = ?";
        PreparedStatement updateStatement =
connection.prepareStatement(updateDisasterSQL);
        updateStatement.setDate(1, Date.valueOf(dateField.getText()));
        updateStatement.setString(2, locationField.getText());
        updateStatement.setString(3, disasterTypeField.getText()); // Corrected field name
        updateStatement.setInt(4, Integer.parseInt(severityField.getText()));
        updateStatement.setInt(5, disasterId);
        updateStatement.executeUpdate();

        JOptionPane.showMessageDialog(this, "Disaster updated
successfully!");

        clearInputFields();
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error updating disaster: " +
e.getMessage());
    }
}

```

```

        }

    }

    private void deleteDisaster() {
        try {
            int disasterId = Integer.parseInt(disasterIdField.getText());

            // Check if the disaster exists
            String checkDisasterSQL = "SELECT * FROM Disaster WHERE disaster_id = ?";
            PreparedStatement checkStatement =
connection.prepareStatement(checkDisasterSQL);
            checkStatement.setInt(1, disasterId);
            ResultSet rs = checkStatement.executeQuery();

            if (!rs.next()) {
                JOptionPane.showMessageDialog(this, "Disaster ID does not
exist!");
                return;
            }

            // Delete from junction table first to maintain referential integrity
            String deleteJunctionSQL = "DELETE FROM disaster_agencies WHERE
disaster_id = ?";
            PreparedStatement deleteJunctionStatement =
connection.prepareStatement(deleteJunctionSQL);
            deleteJunctionStatement.setInt(1, disasterId);
            deleteJunctionStatement.executeUpdate();

            // Now delete from Disaster table
            String deleteDisasterSQL = "DELETE FROM Disaster WHERE disaster_id = ?";
            PreparedStatement deleteDisasterStatement =
connection.prepareStatement(deleteDisasterSQL);
            deleteDisasterStatement.setInt(1, disasterId);
            deleteDisasterStatement.executeUpdate();

            JOptionPane.showMessageDialog(this, "Disaster deleted
successfully!");

            clearInputFields();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error deleting disaster: " +
e.getMessage());
        }
    }
}

```

```

        }

    }

    private void addAgency() {
        try {
            String sql = "INSERT INTO Agency (agency_id, name, type, role) VALUES (?, ?, ?, ?)";
            PreparedStatement statement = connection.prepareStatement(sql);
            statement.setInt(1, Integer.parseInt(agencyIdField.getText()));
            statement.setString(2, agencyNameField.getText());
            statement.setString(3, agencyTypeField.getText());
            statement.setString(4, roleField.getText());
            statement.executeUpdate();
            JOptionPane.showMessageDialog(this, "Agency added successfully.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error adding agency: " + e.getMessage());
        }
    }

    private void updateAgency() {
        try {
            int agencyId = Integer.parseInt(agencyIdField.getText());

            // Check if the agency exists
            String checkAgencySQL = "SELECT * FROM Agency WHERE agency_id = ?";
            PreparedStatement checkStatement =
            connection.prepareStatement(checkAgencySQL);
            checkStatement.setInt(1, agencyId);
            ResultSet rs = checkStatement.executeQuery();

            if (!rs.next()) {
                JOptionPane.showMessageDialog(this, "Agency ID does not exist!");
                return;
            }

            // Update the Agency details
            String updateAgencySQL = "UPDATE Agency SET name = ?, type = ?, role = ? WHERE agency_id = ?";
            PreparedStatement updateStatement =
            connection.prepareStatement(updateAgencySQL);
            updateStatement.setString(1, agencyNameField.getText());
        }
    }
}

```

```

        updateStatement.setString(2, agencyTypeField.getText());
        updateStatement.setString(3, roleField.getText());
        updateStatement.setInt(4, agencyId);
        updateStatement.executeUpdate();

        JOptionPane.showMessageDialog(this, "Agency updated successfully!");

        clearInputFields();

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error updating agency: " +
e.getMessage());
    }
}

private void deleteAgency() {
    try {
        int agencyId = Integer.parseInt(agencyIdField.getText());

        // Check if the agency exists
        String checkAgencySQL = "SELECT * FROM Agency WHERE agency_id = ?";
        PreparedStatement checkStatement =
connection.prepareStatement(checkAgencySQL);
        checkStatement.setInt(1, agencyId);
        ResultSet rs = checkStatement.executeQuery();

        if (!rs.next()) {
            JOptionPane.showMessageDialog(this, "Agency ID does not exist!");
            return;
        }

        // Delete from junction table first
        String deleteJunctionSQL = "DELETE FROM disaster_agencies WHERE
agency_id = ?";
        PreparedStatement deleteJunctionStatement =
connection.prepareStatement(deleteJunctionSQL);
        deleteJunctionStatement.setInt(1, agencyId);
        deleteJunctionStatement.executeUpdate();

        // Now delete from Agency table
        String deleteAgencySQL = "DELETE FROM Agency WHERE agency_id = ?";
        PreparedStatement deleteAgencyStatement =
connection.prepareStatement(deleteAgencySQL);
        deleteAgencyStatement.setInt(1, agencyId);
        deleteAgencyStatement.executeUpdate();
    }
}

```

```

        JOptionPane.showMessageDialog(this, "Agency deleted successfully!");

        clearInputFields();

    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error deleting agency: " +
e.getMessage());
    }
}

private void showDisasters() {
    try {
        String sql = "SELECT * FROM Disaster";
        PreparedStatement statement = connection.prepareStatement(sql);
        ResultSet rs = statement.executeQuery();
        ResultSetMetaData rsmd = rs.getMetaData();

        Vector<String> columnNames = new Vector<>();
        for (int i = 1; i <= rsmd.getColumnCount(); i++) {
            columnNames.add(rsmd.getColumnName(i));
        }

        Vector<Vector<Object>> data = new Vector<>();
        while (rs.next()) {
            Vector<Object> row = new Vector<>();
            for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                row.add(rs.getObject(i));
            }
            data.add(row);
        }

        dataTable.setModel(new DefaultTableModel(data, columnNames));
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error showing disasters: " +
e.getMessage());
    }
}

private void showAgencies() {
    try {
        String sql = "SELECT * FROM Agency";
        PreparedStatement statement = connection.prepareStatement(sql);
        ResultSet rs = statement.executeQuery();
        ResultSetMetaData rsmd = rs.getMetaData();

```

```

        Vector<String> columnNames = new Vector<>();
        for (int i = 1; i <= rsmd.getColumnCount(); i++) {
            columnNames.add(rsmd.getColumnName(i));
        }

        Vector<Vector<Object>> data = new Vector<>();
        while (rs.next()) {
            Vector<Object> row = new Vector<>();
            for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                row.add(rs.getObject(i));
            }
            data.add(row);
        }

        dataTable.setModel(new DefaultTableModel(data, columnNames));
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error showing agencies: " +
e.getMessage());
    }
}

// join queries methods

private void showAgenciesForDisaster() {
    try {
        String sql = "SELECT d.disaster_id, d.date, d.location, d.type AS
disaster_type, d.severity, " +
                    "a.agency_id, a.name, a.type AS agency_type, a.role " +
                    "FROM Disaster d " +
                    "JOIN disaster_agencies da ON d.disaster_id = da.disaster_id
" +
                    "JOIN Agency a ON a.agency_id = da.agency_id " +
                    "WHERE da.disaster_id=?";
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setInt(1, Integer.parseInt(disasterIdField.getText()));
        ResultSet resultSet = statement.executeQuery();

        DefaultTableModel model = new DefaultTableModel();
        model.addColumn("Disaster ID");
        model.addColumn("Date");
        model.addColumn("Location");
        model.addColumn("Disaster Type");
        model.addColumn("Severity");
        model.addColumn("Agency ID");
    }
}

```

```

model.addColumn("Agency Name");
model.addColumn("Agency Type");
model.addColumn("Agency Role");

while (resultSet.next()) {
    Vector<Object> row = new Vector<>();
    row.add(resultSet.getInt("disaster_id"));
    row.add(resultSet.getDate("date"));
    row.add(resultSet.getString("location"));
    row.add(resultSet.getString("disaster_type"));
    row.add(resultSet.getInt("severity"));
    row.add(resultSet.getInt("agency_id"));
    row.add(resultSet.getString("name"));
    row.add(resultSet.getString("agency_type"));
    row.add(resultSet.getString("role"));
    model.addRow(row);
}
dataTable.setModel(model);
clearInputFields();

} catch (SQLException e) {
    JOptionPane.showMessageDialog(this, "Error fetching agencies for
disaster: " + e.getMessage());
}
}

private void showDisastersForAgency() {
try {
    String sql = "SELECT a.agency_id, a.name, a.type AS agency_type,
a.role, " +
                "d.disaster_id, d.date, d.location, d.type AS disaster_type,
d.severity " +
                "FROM Agency a " +
                "JOIN disaster_agencies da ON a.agency_id = da.agency_id " +
                "JOIN Disaster d ON d.disaster_id = da.disaster_id " +
                "WHERE da.agency_id=?";
    PreparedStatement statement = connection.prepareStatement(sql);
    statement.setInt(1, Integer.parseInt(agencyIdField.getText()));
    ResultSet resultSet = statement.executeQuery();

    DefaultTableModel model = new DefaultTableModel();
    model.addColumn("Agency ID");
    model.addColumn("Agency Name");
    model.addColumn("Agency Type");
    model.addColumn("Agency Role");

```

```

        model.addColumn("Disaster ID");
        model.addColumn("Date");
        model.addColumn("Location");
        model.addColumn("Disaster Type");
        model.addColumn("Severity");

        while (resultSet.next()) {
            Vector<Object> row = new Vector<>();
            row.add(resultSet.getInt("agency_id"));
            row.add(resultSet.getString("name"));
            row.add(resultSet.getString("agency_type")); // Renamed column
            row.add(resultSet.getString("role"));
            row.add(resultSet.getInt("disaster_id"));
            row.add(resultSet.getDate("date"));
            row.add(resultSet.getString("location"));
            row.add(resultSet.getString("disaster_type")); // Renamed column
            row.add(resultSet.getInt("severity"));
            model.addRow(row);
        }
        dataTable.setModel(model);
        clearInputFields();
    }
    catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error fetching disasters for
agency: " + e.getMessage());
    }
}

private void addAgencyForDisaster() {
    try {
        String sql = "INSERT INTO disaster_agencies (disaster_id, agency_id)
VALUES (?, ?)";
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setInt(1, Integer.parseInt(disasterIdField.getText()));
        statement.setInt(2, Integer.parseInt(agencyIdField.getText()));
        statement.executeUpdate();
        JOptionPane.showMessageDialog(this, "Agency added for disaster.");
        clearInputFields();
    }
    catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error adding agency for
disaster: " + e.getMessage());
    }
}

```

```

private void deleteAgencyForDisaster() {
    try {
        // Check if the combination of disaster_id and agency_id exists in
        // the junction table first
        String checkSql = "SELECT * FROM disaster_agencies WHERE disaster_id
= ? AND agency_id = ?";
        PreparedStatement checkStatement =
connection.prepareStatement(checkSql);
        checkStatement.setInt(1,
Integer.parseInt(disasterIdField.getText()));
        checkStatement.setInt(2, Integer.parseInt(agencyIdField.getText()));
        ResultSet rs = checkStatement.executeQuery();

        if (!rs.next()) {
            // If no matching record is found, show an error message
            JOptionPane.showMessageDialog(this,
                "The specified combination of Disaster ID and Agency ID
does not exist.");
            return;
        }

        // If the combination exists, proceed with the delete operation
        String deleteSql = "DELETE FROM disaster_agencies WHERE disaster_id =
? AND agency_id = ?";
        PreparedStatement deleteStatement =
connection.prepareStatement(deleteSql);
        deleteStatement.setInt(1,
Integer.parseInt(disasterIdField.getText()));
        deleteStatement.setInt(2, Integer.parseInt(agencyIdField.getText()));
        int rowsAffected = deleteStatement.executeUpdate();

        if (rowsAffected > 0) {
            JOptionPane.showMessageDialog(this, "Agency successfully deleted
from disaster.");
        } else {
            JOptionPane.showMessageDialog(this, "Error: The agency could not
be deleted.");
        }
        clearInputFields();
    }
    catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Error deleting agency from
disaster: " + e.getMessage());
    }
}

```

```
public static void main(String[] args) {
    SwingUtilities.invokeLater(DisasterAgencyApp::new);
}
```

TECHNICAL ISSUES AND SOLUTION

Technical Challenges and solution

1. Challenge: Ensuring Referential Integrity for Deletions in Junction Table

Solution: Added checks for related records in the disaster_agencies junction table before deleting entries from the Agency and Disaster tables. Deleted related junction records first, ensuring smooth and consistent data management.

2. Challenge: Dynamic Table Display with Join Queries

Solution: Utilized DefaultTableModel to dynamically populate tables based on the results of SQL join queries. This allowed for flexible and accurate data displays, facilitating the real-time visualization of disaster-agency relationships.

3. Challenge: Avoiding Data Redundancy in Junction Tables

Solution: The disaster_agencies table, as a junction table, faced redundancy risks if the same disaster_id and agency_id were linked multiple times. Before any addition, the code checked the existence of the disaster_id and agency_id pair to prevent duplicate entries.