Group - 30

HP

**IT607 Introduction to Database Management**

**Messaging App Database**

**Student Id – 202318026, 202318059**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. **Introduction** | | | 3 |
| 1.1 | Description | | 4 |
| **2. Documenting the Requirements Collection/ Fact Finding Phase** | | | 10 |
| 2.1 | Documenting the input and the output for the requirements collection techniques used by us: | | 10 |
| 2.1.1 | Background Reading/s | | 10 |
|  | 1 | Description of each reading done | 10 |
|  | 2 | References | 11 |
|  | 3 | Summary of each document that you have read during Background Reading/ s phase. | 12 |
| 2.1.2 | Interview/s | | 13 |
|  | 1 | Interview Plan, Interview Summary | 13 |
|  | 2 | Combined Requirements gathered from all Interview/s. | 20 |
| 2.1.3 | Questionnaire/s | | 22 |
|  | 1 | Questionnaire/s | 22 |
|  | 2 | Summary | 25 |
|  | 3 | Combined Requirements gathered from Response/s. | 30 |
| 2.1.4 | Observation/s | | 32 |
|  | 1 | Summary of observations related to our database. | 32 |
|  | 2 | Combined Requirements gathered from Observation/s. | 33 |
| 2.2 | Fact Finding Chart | | 34 |
| **3.List of Requirements** | | | 36 |
| **4.User categories and privileges** | | | 42 |
| 4.1 | User Names with basic description of user role in the system/database | | 42 |
| 4.2 | List of privileges for each use categories | | 43 |
| **5.List of Assumptions while designing this database** | | | 45 |
| **6.Business Constraints** | | | 48 |

**1.Introduction**

* **Purpose**

A case study on a messaging app can serve various purposes, depending on the context and goals of the study. The purpose of this case study is to address the challenges and requirements associated with managing the database for a messaging application. It will explore the need for a robust and efficient database management system to support the application's data storage and retrieval needs, ensuring scalability, User Engagement and Retention, Global Impact, Understanding Market Dynamics, Analyzing User Behavior, Competitive Analysis, Future Trends, Lessons for Other Tech Companies, Regulatory and Legal Issues, security, and data integrity.

* **Intended Audience and Reading Suggestions**

This case study is intended for common users, developers and business organizations involved in the development and maintenance of a messaging application. It is recommended that readers have a basic understanding of database management principles and software development processes. The intended audience for a case study or research paper on a messaging app can vary depending on the focus and depth of the study. These reading suggestions provide a starting point for different audience groups interested in various aspects of messaging apps. Depending on the specific research focus or area of interest, readers may seek out additional academic papers, reports, and books to gain a deeper understanding of messaging apps and their implications.

* **Product Scope**

This case study covers several important aspects related to managing a database system for messaging apps. First and foremost, it emphasizes the significance of having a well-structured database system in place. It also highlights the common challenges and issues that often arise when dealing with database management in mobile app development.

The study delves into the essential components and functionalities that an ideal database management system for messaging apps should possess. It further discusses best practices for maintaining data security, implementing backup and recovery procedures, all within the specific context of messaging app databases.

Optimizing database performance to ensure a smooth and enjoyable user experience is another crucial topic addressed in this case study. Additionally, it addresses concerns related to

scalability, especially in dealing with increasing volumes of data in the context of messaging apps.

Furthermore, the study explores the integration of cloud-based databases and synchronization mechanisms, which are increasingly vital for distributed apps like messaging apps. It provides real-world examples and case studies to illustrate successful approaches to messaging app database management. Overall, this case study offers valuable insights into the complex world of managing databases for messaging applications.

**1.1 Description**

Our Messaging App will be a free cross-platform messaging service. It will let users of iPhone, Android smartphones, Mac and Windows PC call and exchange text, photo, audio and video messages with others across the globe for free, regardless of the recipient's device. Our Messaging App will use a Wi-Fi connection to communicate cross-platform, unlike Apple iMessage and Messages by Google, which require cellular networks and Short Message Service (SMS). It will serve as a platform for over millions of users globally, exchanging text messages, voice notes, images, videos, and other multimedia content. Managing this immense volume of data while ensuring user privacy and security is a substantial challenge. This case study will search into the database management practices employed by Messaging App to provide reliable service to its users.

The messaging app will use Wi-Fi, making it cost-effective and popular with users who will not have data plans with unlimited calls and text messaging in the future. Its cross-platform feature will also be popular with people who will have family abroad, will travel internationally, or will live outside any country.

In today's digital landscape, Messaging App applications has become an integral part of our daily lives. From social media to communication and productivity tools, Messaging App apps rely heavily on efficient and secure database management systems to store and retrieve data. This case study delves into the critical role that database management plays in the success of Messaging App applications.

**Messaging App's Data Structure**

Within the Messaging App database, a rich tapestry of data types converges to form a comprehensive ecosystem. Messages, whether they take the form of text, voice recordings, or multimedia content, constitute the lifeblood of communication on the platform. User Profiles contribute a personal dimension, housing details such as names, profile pictures, and status updates, providing users with an identity within the digital realm. The database also meticulously logs Calls, capturing crucial data like call durations and metadata, ensuring a thorough record of communication history. Group Data takes form as a repository of information about various chat groups, including member details and chat records. The Multimedia category accommodates a diverse array of content, encompassing images, videos, audio files, and documents shared within conversations. Encryption stands as a pillar of security, with the database housing end-to-end encryption keys and essential security considerations, safeguarding user privacy and data integrity. Together, these data types constitute the backbone of the Messaging App's functionality and user experience.

The development of a comprehensive messaging app would require several intricate modules. First and foremost, a robust User Management Module is crucial. This encompasses user registration and authentication to sign up new users and validate the existing ones. It also manages user profile information, such as usernames, profile pictures, status, and bio. In addition, it would incorporate user settings and preferences, adjusting for read receipts, backup settings, dark mode, and so on. An integral part of this would be syncing and managing contacts imported from the user's device.

The heart of the app lies in its Messaging Module. It should efficiently handle sending, receiving, and displaying text messages. Additionally, it would need to manage multimedia content, including images, videos, audio files, and documents. Users should be able to record, send, and playback voice notes. Moreover, tracking message status, from being sent to delivered to read, is essential. A search functionality would also be beneficial, allowing users to locate specific messages or filter their chats.

Parallelly, the Call Management Module is responsible for voice and video calls. It would take care of recording call data, duration, and other relevant metadata. Notifications of incoming or missed calls are equally vital. Groups form a significant part of modern messaging. Hence, a Group Management Module is a must. It would oversee the creation and management of groups, handle group chats, customize group details, and set permissions.

Data safety cannot be overstated. A Data Backup & Recovery Module would provide local and cloud backup options, ensuring that users can retrieve their data when needed. In our age of cyber threats, a Security & Encryption Module is non-negotiable. This would encrypt data end-to-end, provide two-factor authentication for enhanced security, and allow users to manage their privacy settings.

Given the multimedia nature of modern communication, a Multimedia Management Module is pivotal. This would display shared media, ensure efficient media transmission, and facilitate file sharing. Notifications keep users engaged, making a Notifications & Alerts Module fundamental. This would manage alerts for messages, calls, and even app updates.

Connectivity is the backbone of any online platform. Therefore, a Connectivity Module would manage the switch between Wi-Fi and cellular data and handle message transmission during offline periods. Lastly, user experience is enhanced with a dedicated User Support & Feedback Module. This would guide users on app functionalities and collect valuable feedback, allowing continuous improvement.

In essence, by interweaving these modules, developers can ensure a holistic, cohesive, and feature-rich messaging application.

**Case Studies**

This case study explores the details of how to handle and take care of a Messaging App's database, focusing on the challenges, solutions, and best practices involved in handling the vast amount of data generated by one of the world's most popular messaging platforms. We explore the growth of Messaging App, its data structure, the importance of database management, and real-world examples of companies successfully managing Messaging App databases.

Messaging App developers often face challenges such as ensuring data consistency, protecting sensitive information, and optimizing database performance to deliver a seamless user experience. The study will emphasize the need for a well-designed database schema tailored to the specific requirements of the application, as well as the importance of data indexing, caching, and query optimization.

Security is a paramount concern in mobile app development, and this case study will explore various security measures, including encryption, access control, and authentication, to safeguard user data. It will also discuss strategies for regular data backups and disaster recovery to prevent data loss.

Scalability is another crucial aspect, especially as mobile applications grow in popularity. Readers will learn about techniques for horizontal and vertical scaling, load balancing, and database sharding to handle increasing user loads.

Message Persistence is a critical aspect of the Messaging App, allowing users to maintain their chat history, which may include multimedia content. However, managing this persistence across various platforms such as iOS, Android, and web presents notable challenges that need to be addressed for a seamless user experience.

Data Backup and Recovery are essential functionalities expected by users. They rely on the ability to back up their chat history and easily recover it when switching devices or reinstalling the app. This feature ensures that users can seamlessly transition without losing their valuable conversations.

Real-time Messaging is at the core of the app's functionality. Messages must be delivered instantly, requiring efficient data transmission and synchronization mechanisms to ensure that users experience real-time communication without delays.

Given the enormous volume of data generated daily by the Messaging App, scalable storage solutions are imperative to accommodate this data influx effectively. Scalability is key to maintaining optimal performance.

The diversity of data types within the app necessitates versatile storage and retrieval mechanisms. From text messages to multimedia content, the platform must handle a wide range of data formats efficiently.

Effective data retention policies are essential to manage the wealth of user data and ensure compliance with regulatory requirements. Maintaining data for the appropriate duration and securely disposing of it when necessary is critical.

Lastly, Data Access Control is paramount to safeguarding user privacy and data security. Implementing stringent access controls ensures that only authorized personnel can access sensitive user data, bolstering the overall security posture of the Messaging App.

**Database Management Strategies:**

Messaging App employs a variety of database management strategies to ensure the smooth and secure operation of its services. One of the key techniques it utilizes is sharding, which involves distributing data across multiple database servers horizontally. This approach effectively manages scalability, allowing the app to handle increasing volumes of data without compromising performance.

In terms of user privacy and security, Messaging App is committed to safeguarding messages through end-to-end encryption, utilizing the Signal Protocol. This encryption ensures that only the intended recipient can decrypt and read the messages, enhancing the overall privacy of its users.

Given the substantial amount of multimedia content shared daily, the Messaging App relies on content delivery networks (CDNs) and media optimization algorithms. These technologies help compress and deliver media files efficiently, ensuring a seamless user experience.

When it comes to managing user chat history, the Messaging App provides options for both local device storage and cloud backups. This involves careful coordination to ensure data synchronization across platforms, allowing users to access their chat history seamlessly, even when switching devices.

Furthermore, the app offers data backup features, enabling users to securely store their chat history in cloud services such as Google Drive or iCloud. This functionality simplifies data recovery and ensures that users can easily retrieve their data when transitioning to new devices.

To support real-time messaging, Messaging App relies on an efficient infrastructure, incorporating Web Sockets and push notification systems. These technologies ensure that messages are delivered instantly, contributing to a smooth and responsive user experience.

In terms of data management and security, the app places a strong emphasis on defining and adhering to data retention policies. It also implements role-based access control (RBAC) to regulate data access and maintain security standards. Additionally, Messaging App is committed to compliance with data protection regulations, such as GDPR, to safeguard user data and privacy effectively.

Some other components we can add in our Database

A user-centric design approach is of paramount importance in creating a platform that resonates with users, whether they are designers, developers, or end-users. An intuitive User Interface (UI) coupled with a thoughtful User Experience (UX) can significantly enhance user engagement and satisfaction. When users find the platform easy to navigate and use, it fosters a positive experience that keeps them engaged and returning.

To foster continuous improvement and user involvement, implementing a robust feedback and improvement system is essential. This mechanism allows users to provide valuable insights about the database, its features, and their overall experience. This ongoing feedback loop not only facilitates regular enhancements but also makes users feel more connected and valued.

For new users, effective onboarding through sessions, tutorials, or webinars is crucial. Providing continuous educational resources helps users become familiar with the platform, boosting their confidence and overall engagement.

Additionally, considering the integration of external tools, such as animation software or other social media platforms, can expand the platform's versatility and utility, catering to a broader range of user needs.

To stay relevant and informed, incorporating a feature that aggregates messaging app trends, market demands, or new technologies can benefit both developers and users, keeping them up-to-date with the evolving landscape.

As sustainability gains prominence, prioritizing sustainable performance, features, or design principles can be a distinguishing factor that aligns with the growing emphasis on environmental responsibility.

If the platform aims for global use, features accommodating different languages, currencies, and unique design preferences should be considered. This localization and globalization effort can enhance user accessibility and inclusivity.

Building a sense of community and networking among users, especially developers, by creating a forum or space for idea sharing, discussions, and peer support fosters a collaborative ecosystem.

Regular system checks, maintenance, and updates are imperative to ensure smooth operation and the incorporation of the latest features, keeping the platform competitive and efficient.

Providing emergency support and robust customer service channels ensures that users receive assistance promptly, addressing technical issues or concerns effectively.

Finally, allowing users to customize their dashboard or workspace based on their preferences and needs adds a layer of personalization, making the platform more tailored and appealing to individual users. Incorporating these suggestions creates a comprehensive and user-centric database platform that can attract and retain a diverse user base effectively.

**Conclusion**

Managing a Messaging App database is a complex undertaking due to the sheer volume and diversity of data generated by the platform. Companies and organizations must invest in scalable infrastructure, robust security measures, and compliance with data protection regulations to effectively manage Messaging App databases. Real-world case studies provide valuable insights into successful database management practices, ensuring that user data remains secure and accessible when needed.

Furthermore, the case study will introduce cloud-based database solutions and synchronization mechanisms that enable seamless data access across multiple devices and platforms. Real-world examples and case studies will be provided to illustrate successful Messaging App database management strategies.

By the end of this case study, the audience will have a comprehensive understanding of the challenges, best practices, and technologies involved in managing databases for mobile applications, enabling them to make informed decisions and implement effective solutions in their own projects

**2. Documentation of the Requirements Collection/ Fact Finding Phase**

**2.1Documenting the input and the output for the requirements collection techniquesused by us:**

**2.1.1 Background Reading for Messaging App Database Management**

Book 1: "Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke

Reference: Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems. McGraw-Hill.

Summary:

* This comprehensive book provides a foundational understanding of database management systems, including concepts, design, and implementation.
* It covers topics like data modeling, query languages, transaction management, and database administration, which are crucial for managing Messaging App's database.

Website: Messaging App Business API Documentation

**Reference**: Messaging App Business API Documentation

Link 1 - (<https://www.twilio.com/docs/sms>)

Summary:

* The official Messaging App Business API documentation offers insights into Messaging App's API for businesses, including database-related functionalities.
* It provides details on message templates, message sending, and webhook configurations, which are essential for database integration.

Article: "Messaging App Security and End-to-End Encryption" by Messaging App

**Reference**: Messaging App. (n.d.). Messaging App Security and End-to-End Encryption.

Link -2 (<https://www.WhatsApp.com/security/>)

Summary:

* This article explains Messaging App's commitment to security and end-to-end encryption, which has implications for database management, especially regarding user data protection.
* Understanding encryption is vital when handling sensitive data in Messaging App's database.

Video 1. : "Database Management for Beginners" by Corey Schafer

**Reference**: Corey Schafer. (2019). Database Management for Beginners.

Video 1 - (<https://www.youtube.com/watch?v=HXV3zeQKqGY>)

Summary:

* This YouTube video provides an introductory overview of database management concepts.
* It covers key terms and principles that are applicable to managing databases for Messaging App, such as data modeling and normalization.

Article: "Scaling to Billions" by Messaging App Engineering Team

Reference:Five security principles for billions of messages across Meta’s apps Link -

Link 3 - (<https://engineering.fb.com/2022/07/28/security/five-security-principles-for-billions-of-messages-across-metas-apps/)>

Summary:

* Messaging App Engineering shares insights into scaling Messaging App to handle billions of users and messages.
* This article discusses infrastructure and database management challenges faced by Messaging App and their solutions.

Book 2: "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" by Martin Fowler and Pramod J. Sadalage

**Reference:** Fowler, M., &Sadalage, P. J. (2012). NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Professional.

Summary:

* This book introduces NoSQL databases, which may be relevant for handling different types of data in Messaging App's database system.
* It covers various NoSQL database models and their use cases.

Article: "Messaging App's Use of Erlang" by Francesco Cesarini

**Reference**: Addevice.io: How to Create a Messaging Application from Scratch in 2023.

Link 5 - (<https://www.addevice.io/blog/how-to-create-a-messaging-application-from-zero>)

Summary:

* This article explores how Messaging Appis developed from scratch
* Understanding Messaging App's technical stack is essential for efficient database management.

News: "Messaging App Privacy Policy Changes" by BBC News

**Reference**: BBC News. (2021). Messaging App Privacy Policy Changes.

Link 4 - (<https://www.bbc.com/news/technology-59348921>)

Summary:

* This news article discusses recent changes to Messaging App's privacy policy, highlighting potential impacts on user data management.
* Staying updated on policy changes is crucial when managing Messaging App's database.

**Reference**: Signal president Meredith Whittaker on Big Tech & protecting consumer data

Video 2 - <https://www.youtube.com/watch?v=eouVRp6x2eU>

Summary: A vocal critic of Big Tech, Meredith Whittaker became the first president of the encrypted messaging app Signal last fall. On Tuesday, Jan. 10 at 1:00 p.m. ET, Whittaker joins Cat Zakrzewski, The Post’s technology policy reporter, for a conversation about Signal’s efforts to protect consumer data and privacy, the future of the tech industry and her work on the social implications of artificial intelligence.

**Reference**:WhatsApp founder Jan Koum interviewed by David Rowan at DLD14

Video 4 - <https://www.youtube.com/watch?v=U4iY1CJvF8k>

Summary: This interview tells us about Jan Koum’s business model and the difference between apple and android Ecosystems

**2.1.2 Group\_30 Interview Plan for common users**

**System** – Messaging App

**Project Reference**: MA/DB/2023/09

**Participants**: Anurag Choudhury,Satyam Maravaniya, Aditya Tripathi

**Date**: 25/09/2023

**Time**: 10:00 am

**Duration**: 10 minutes

**Place**: Daiict Hostel

**Purpose of Interview**: To gather information about the current state of the project and discuss potential improvements or challenges.

**Agenda**

To know about Data Security, Privacy, Sharing and Performance, Data Retention and Deletion.

To know about Backup and Disaster Recovery, User Experience and User interface

Documents to Be Brought Into the Interview:

Project Plan: A copy of the current project plan or roadmap to reference during the discussion.

Progress Reports: Any progress reports or documents that provide insights into the project's current state.

**Questions**

**Interviewer:** What basic information would you like your profile to contain? (e.g., username, profile picture, status message)

**Interviewee:** - I'd like my profile to contain a username, profile picture, and a status message. Additionally, an optional bio section would be nice, where I can write a bit about myself.

**Interviewer:**Do you prefer to have a backup of your messages? If yes, how often?

**Interviewee:**- Yes, I'd like to have a backup of my messages for security reasons. Automatic weekly backups would be good, with the option to manually backup at any time.

**Interviewer:**  How do you feel about message read receipts?

**Interviewee:**- I like having read receipts as it lets me know if the other person has seen my message. However, I'd also appreciate the option to turn them off for privacy reasons.

**Interviewer:** How would you feel about two-factor authentication for logging in?

**Interviewee:**- I believe two-factor authentication is essential for added security. While it might be a slight inconvenience, it provides peace of mind knowing that my account is secure.

**Interviewer:** How long would you expect media files (like images and videos) to be available in the chat?

**Interviewee:**- I would expect media files to remain in the chat indefinitely unless I choose to delete them. If there's a space concern, maybe compress older files, but let me access the original if needed.

**Interviewer:** How do you feel about auto-deletion of old messages?

**Interviewee:**- I'm not particularly in favor of auto-deletion unless it's an optional feature. If implemented, I'd like to be able to set the time frame for when messages are deleted.

**Interviewer:** How important is it for you to have a "dark mode" or theme customization?

**Interviewee:**- Very important! Dark mode helps reduce eye strain during nighttime use, and theme customization allows me to personalize my experience.

**Interviewer:** Would you use stickers, GIFs, or emojis regularly in chats?

**Interviewee:**- Yes, I would. They add fun and expressiveness to conversations.

**Interviewer:** How would you feel about a "forwarded" label on forwarded messages to differentiate them from original messages?

**Interviewee:**- I think it's a good idea, as it adds transparency and clarity to chats. It can also help combat misinformation or false claims.

**Interviewer:** Are there other apps' features you particularly like and would want to see?

**Interviewee:**- I really like voice notes from other apps, they're convenient when I don't want to type. Also, the "reply to specific message" feature where you can quote and respond to a particular message in a chat is very helpful.

**Interviewer:** How do you feel about chatbots or automated replies?

**Interviewee:**- They can be helpful in certain scenarios, like customer service or frequently asked questions. But for personal chats, I prefer human interaction. If implemented, it should be clear when I'm interacting with a bot.

**Group 30: Interview Summary for users:**

**System** – Messaging App

**Project Reference** – MA/DB/2023/09

**Participants**– Anurag Choudhury, Satyam Maravaniya, Aditya Tripathi,

**Date** – 17/08/2023 **Time** – 08:15 PM

**Duration** - 10 MINUTES **Place** – DAIICT HOSTEL

**Purpose of Interview**: To gather information about the current state of the project and discuss potential improvements or challenges.

The interview aimed to gather insights about the project's current status, challenges, and potential improvements. The key points discussed during the interview include:

Project objectives and scope were clarified.

The current progress of the project was reviewed, including completed milestones.

Challenges and concerns related to the project were discussed, with a focus on potential areas for improvement.

Next steps for the project were outlined, and responsibilities were defined.

The interviewee had an opportunity to ask questions and provide additional information.

Overall, the interview provided valuable information that will contribute to project planning and decision-making.

**Group\_30 Interview Plan for developers**

**System** – Messaging App

**Project Reference**: MA/DB/2023/09

**Participants**: Anurag Choudhury, Satyam Maravaniya, Divyanshu Singh

**Date**: 25/09/2023

**Time**: 10:00 pm

**Duration:** 10 minutes

**Place:**Daiict Hostel

**Purpose of Interview**: To assess the candidate's qualifications, skills, and suitability for a specific role in the development of messaging applications.

**Agenda:**

The agenda behind conducting interviews for messaging app developers is to thoroughly assess the qualifications, skills, and suitability of candidates for roles related to developing messaging applications. These interviews serve several key objectives, including evaluating candidates' technical proficiency in areas such as programming languages, messaging protocols, and relevant frameworks and tools.

Documents to Be Brought Into the Interview:

Project Plan: A copy of the current project plan or roadmap to reference during the discussion.

Progress Reports: Any progress reports or documents that provide insights into the project's current state.

**Questions**

**Interviewer:** Can you briefly explain the architecture of the messaging app you developed?

**Interviewee:**- The app follows a client-server architecture. The clients are the mobile or desktop applications that users interact with. They communicate with a centralized server where the application logic runs and where messages are temporarily stored and relayed. The server interacts with a database to store user profiles, chat histories, and other relevant data.

**Interviewer:** How did you decide between client-server, peer-to-peer, or hybrid models for your messaging app?

**Interviewee:**- We went with a client-server model because it offers better control, scalability, and ease of deploying updates. While peer-to-peer offers more privacy, it can be challenging in terms of NAT traversal, offline message delivery, and ensuring a seamless experience across devices.

**Interviewer:** What kind of database system did you choose for the app (e.g., relational, NoSQL) and why?

**Interviewee:**- We chose a hybrid approach. For structured data like user profiles, relationships, and chat metadata, we used a relational database. For chat histories and unstructured data, we used a NoSQL database because of its ability to scale and handle large volumes of rapidly changing data.

**Interviewer:**How do you ensure data integrity and consistency in the database?

**Interviewee:**-We use ACID (Atomicity, Consistency, Isolation, Durability) compliant database systems. We also have mechanisms like transactions and rollback features. Regular audits and checks are performed to ensure data integrity.

**Interviewer:** How do you ensure end-to-end encryption for messages?

**Interviewee:**- We implemented the Signal Protocol for end-to-end encryption. When two users initiate a chat, their clients exchange encryption keys, ensuring that only the sender and receiver can read the messages. The server only sees encrypted data, ensuring privacy.

**Interviewer:** Have you ever faced data loss or corruption? How did you recover from it?

**Interviewee:**- We had a few instances in our early stages. We mitigated this by maintaining regular backups and having a robust disaster recovery plan. For corrupted data, we restored from the most recent clean backup and then replayed transaction logs to recover the most up-to-date state.

**Interviewer:** What challenges did you face in ensuring real-time message delivery and sync?

**Interviewee:**- Network latency and connectivity issues were primary challenges. We used WebSocket for a persistent connection between client and server for real-time communication. For syncing issues, we timestamped every message and employed algorithms to check and reconcile any out-of-order or missing messages.

**Interviewer:** What were some of the biggest challenges you faced in developing the messaging app, especially concerning the database?

**Interviewee:**- Scaling was a significant challenge. As user numbers grew, ensuring that the database could handle the increased load, both in terms of storage and query processing, was tricky. Another challenge was ensuring data privacy and security, especially in the face of various cyber threats.

**Interviewer:** What advice would you give to someone looking to design a database for a messaging app?

**Interviewee:**- Start with a clear understanding of your data model and expected loads. Decide on the database type based on your app's needs, not trends. Ensure that your database can scale horizontally to handle future growth. Prioritize security from day one – consider encryption both at rest and in transit. Lastly, always have a backup and disaster recovery plan in place.

**Group 30: Interview Summary for developers:**

System – Messaging App

Project Reference – MA/DB/2023/09

Participants – Anurag Choudhury, Satyam Maravaniya, Divyanshu Singh

Date – 17/08/2023 Time – 08:15 PM

Duration - 10 MINUTES Place – DAIICT HOSTEL

**Purpose of Interview**: To assess the candidate's qualifications, skills, and suitability for a specific role in the development of messaging applications.

In an interview with messaging app developers, the It began with their roles and backgrounds, followed by an overview of the app's purpose and unique features. The technology stack used for development was discussed, along with the challenges faced, emphasizing both technical and user experience aspects.

Security and privacy measures, crucial in messaging apps, were highlighted. Scalability strategies, vital for accommodating rapid user growth, were explained. Future developments, including upcoming features and updates, were previewed. The developers also emphasized the importance of engaging with the user community and shared monetization strategies.

Lastly, compliance with legal and regulatory requirements was addressed, and the interview concluded with the developers summarizing their experiences and offering valuable insights for those interested in the messaging app industry.

Combined Requirements gathered from all Interview/s.

From the User Perspective Interview:

1. User Profile Preferences: Users want basic profile information like username, profile picture, status message, and optionally, a bio section.

2. Data Safety and Retention: Users value the backup of their messages and prefer options for manual and automatic backups.

3. Read Receipts: They provide clarity in communication, but users want the option to turn them off for privacy.

4. Security: Two-factor authentication is appreciated for enhanced security, even if it's a slight inconvenience.

5. Media Handling: Users expect media files to stay in chats indefinitely, but there's openness to compression methods for older files.

6. Message Deletion: Auto-deletion should be optional, allowing users to set their desired timeframe.

7. Customization: Dark mode and theme customization are essential for user experience.

8. Communication Enhancements: Users love expressive tools like stickers, GIFs, and emojis.

9. Transparency: Features like a "forwarded" label on messages provide transparency in conversations.

10. Chatbots: Useful in service scenarios but should be distinguishable from human interaction.

**From the Developer Perspective Interview:**

1. Architecture: The importance of choosing the right architecture (client-server vs. peer-to-peer vs. hybrid) based on control, scalability, and user experience.

2. Database Type Decision: A hybrid database approach might be optimal, using relational databases for structured data and NoSQL for chat histories and unstructured data.

3. Data Integrity: The significance of ACID compliance, transactions, and rollbacks to ensure data integrity.

4. Encryption: The necessity of end-to-end encryption protocols like the Signal Protocol to ensure user data privacy.

5. Disaster Recovery: Regular backups, robust recovery plans, and the replay of transaction logs can save from data loss or corruption.

6. Real-Time Delivery: Tools like WebSocket ensure real-time communication, and timestamping helps in message synchronization.

7. Scaling Challenges: The necessity to design the database for scalability, anticipating growing user numbers and data loads.

8. Security: Prioritizing encryption both at rest and in transit, right from the design phase.

**2.1.3 Questionaires**

**For Users**

* Name

1How do you prefer to log in to a Messaging App?

* Phone number
* Email
* Social media account

1. How important is security and privacy for your  Messaging App account?

* Very important
* Important
* Somewhat important
* Not important

1. What types of messages do you send on a Messaging App? (Select all that apply)

* Text
* Images
* Videos
* Voice messages
* Documents
* Stickers
* GIFs

1. How often do you send messages to individuals vs. groups on a Messaging App?

* Mostly individuals
* Mostly groups
* About the same

1. Do you use a Messaging App's chat backup or archive features?

* Yes
* No

1. Do you have any concerns about the security of your a Messaging App messages?

* Yes
* No

1. Do you experience any performance issues with a Messaging App, such as lag or slow message delivery?

* Yes
* No

1. Please share any additional comments or suggestions you have regarding a Messaging App improvements

* [Text Area]

**For developers**

* Name

1. What is your target audience?

* Young people
* Middle aged people
* Senior citizens

2 Which programming framework have you used for front end development?

* React
* AngularJS
* Bootstrap
* Svelte
* Other:

1. Which programming framework have you used for back end development?

* Django
* Node JS
* Ruby on Rails
* Asp .NET
* Spring Boot ( JAVA )
* Other:

1. How does your messaging app ensure user security?

* End-to-end encryption
* Two-factor authentication
* Spam and malware detection
* User data protection

1. How much time it took for you to develop the app?

* Less than 6 months
* 6-12 months
* 12-18 months
* 18-24 months

1. How does your app handle message backups and data recovery for users?

* Automatic backups
* Manual backup options
* Cloud-based backups
* No backup feature
* **Summary**

**For Users**

**Forms response chart. Question title: How do you prefer to log in to a Messaging App?
. Number of responses: 32 responses.**

Forms response chart. Question title: How important is security and privacy for your  Messaging App account?
. Number of responses: 32 responses.

Forms response chart. Question title: What types of messages do you send on a Messaging App? (Select all that apply)
. Number of responses: 32 responses.

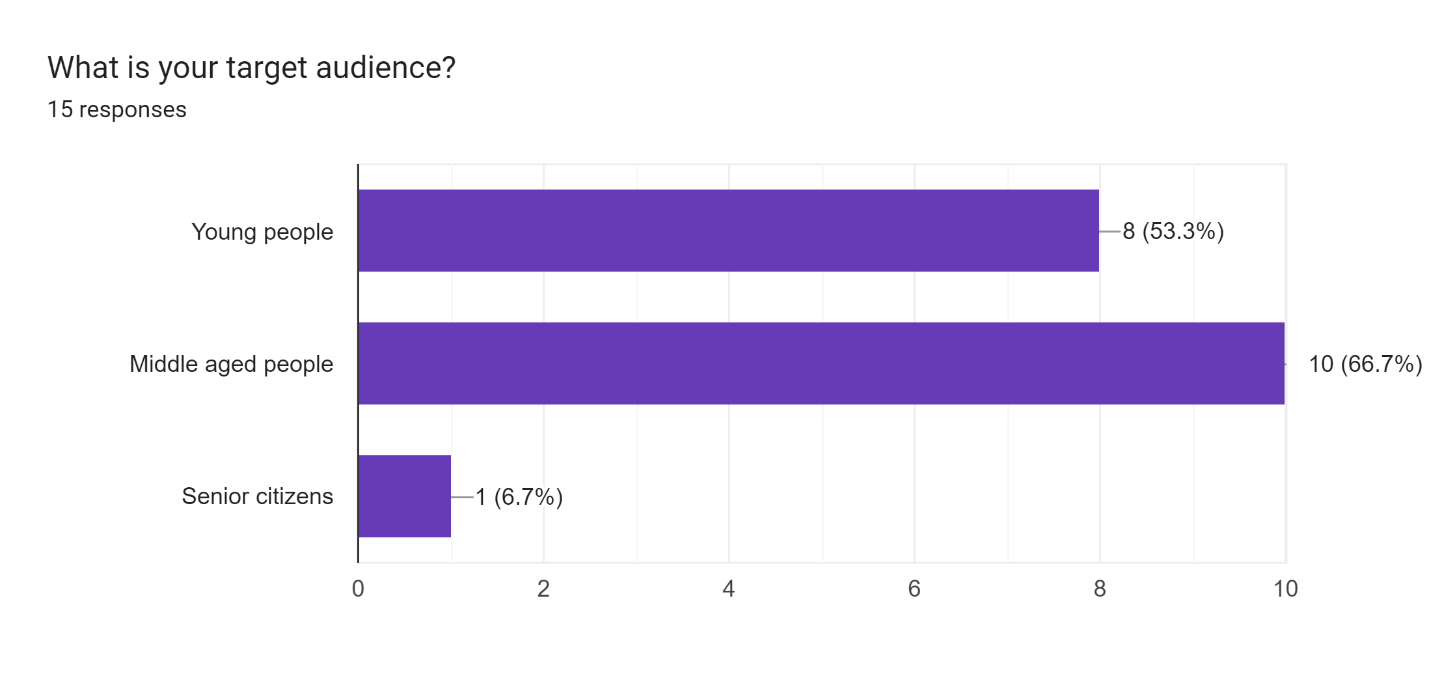
Forms response chart. Question title: How often do you send messages to individuals vs. groups on a Messaging App?
. Number of responses: 32 responses.

Forms response chart. Question title: Do you use a Messaging App&apos;s chat backup or archive features?
. Number of responses: 32 responses.

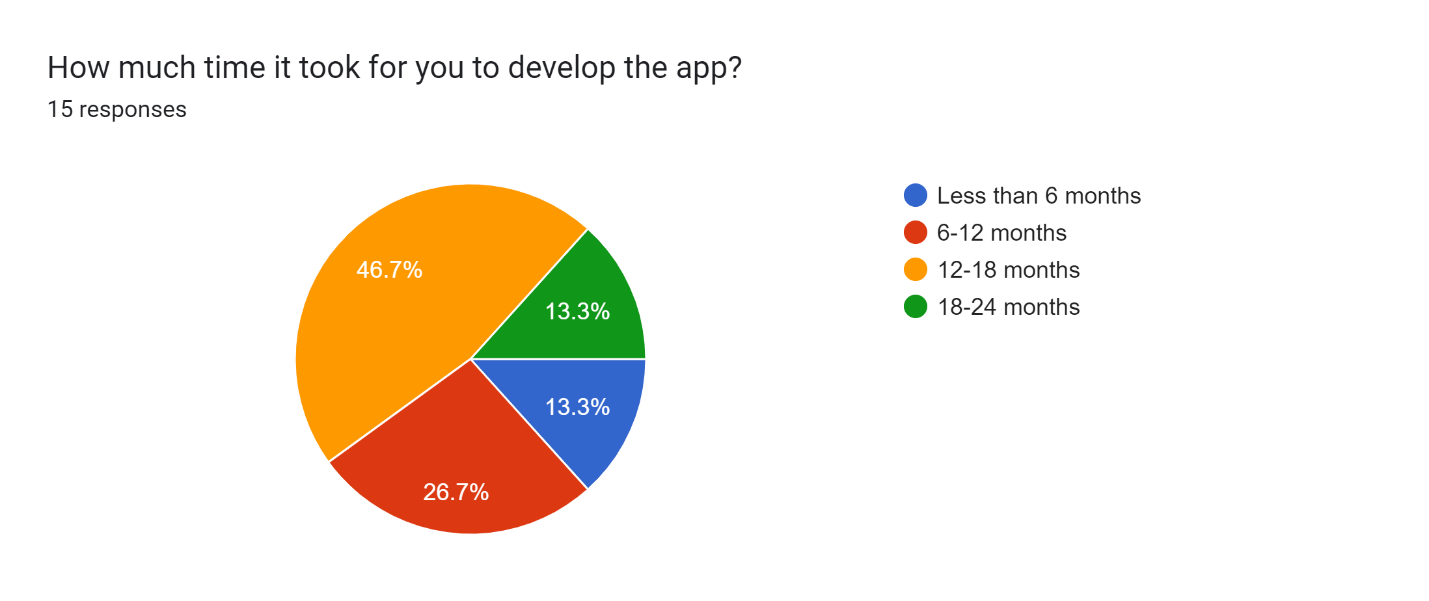
Forms response chart. Question title: Do you have any concerns about the security of your a Messaging App messages?
. Number of responses: 32 responses.

Forms response chart. Question title: Do you experience any performance issues with a Messaging App, such as lag or slow message delivery?
. Number of responses: 32 responses.

**For Developers**

****

Forms response chart. Question title: How does your messaging app ensure user security?
. Number of responses: 15 responses.



Forms response chart. Question title: How does your app handle message backups and data recovery for users?
. Number of responses: 15 responses.

**Combined Requirements gathered from Response**

**For common users**

**1. Login Preferences:**

Most respondents prefer logging in using their Phone number, followed by Email and then Social media account.

**2. Security and Privacy Importance:**

The majority find security and privacy Very important.

A significant number also consider it Important or Somewhat important.

Only a few find it Not important.

**3. Types of Messages Sent:**

Text, Images, and Videos are the most common types of messages sent by respondents.

Documents, Voice messages, Stickers, and GIFs are also popular but slightly less common.

A few respondents send more niche types of messages.

**4. Messaging Pattern:**

Most respondents communicate via Mostly individuals compared to groups.

Some communicate About the same to individuals and groups.

**5. Backup/Archive Feature Usage:**

A majority of the respondents use the backup or archive features of the app.

Some respondents do not use these features.

**6. Security Concerns:**

While many respondents have no concerns about the security of their messages, there's still a significant number who expressed concerns.

**7. Performance Issues:**

Many respondents have not experienced any performance issues such as lag or slow message delivery.

however, a noticeable portion has faced such issues.

In conclusion, the respondents value security and privacy in their messaging app, prefer to communicate with individuals over groups, and have provided specific feedback to enhance app functionality and performance.

**For Developers:**

Primarily, apps are developed for "Young people" and "Middle aged people", with only one developer targeting "Senior citizens".

**Security:**

The most emphasized security features include "Two-factor authentication" and "End-to-end encryption". Developers also highlighted "Spam and malware detection" and "User data protection".

**Development Time:**

Many developers took "12-18 months" to develop their app. Some managed to develop in "Less than 6 months", while others took as long as "18-24 months".

**Backup and Data Recovery:**

Developers offer a mix of "Automatic backups", "Cloud-based backups", and "Manual backup options". Surprisingly, a couple of apps do not feature any backup mechanism.

Several developers, provided more than one entry, possibly indicating different versions or iterations of their app.

**2.1.4 Observations:**

**By Users**

Positive Observations:

* User-Friendly Interface: The app is easy to navigate and intuitive.
* Variety of Features: The app provides features like voice/video calling, stickers, GIFs, etc.
* Privacy Settings: The app offers end-to-end encryption and options to hide 'last seen' or profile pictures from certain users.
* Multi-Platform Accessibility: The messaging app can be accessed from various devices like smartphones, tablets, and desktops.

Negative Observations:

* Battery Drain: The app consumes too much battery on their phone.
* Lack of Customization: Limited options for changing themes or chat backgrounds**.**
* Inconsistent Notifications: Sometimes, the user doesn't receive notifications for new messages.

**By App Developers**

Positive Observations:

* Efficient Code: The app runs smoothly, indicating a well-optimized backend.
* Scalability: The app handles a large number of users without performance degradation.
* 5. Integration Capabilities: The app integrates well with other apps or services (e.g., calendar, photo gallery).

Negative Observations:

* Security Gaps: The developer might notice potential vulnerabilities, like weak encryption or susceptibility to SQL injection.
* Rigidity: It might be difficult to add new features or make changes to the app because of its design.
* Over-reliance on Third-party Services: The app might be using too many third-party services, making it dependent on them for functionality.
* Redundant Processes: The app might be running unnecessary processes in the background, consuming more memory.

**Requirements:**

Based on these observations, the combined requirements for a messaging app database design would include:

**Data Storage and Handling:**

Efficiently store and manage a variety of data types, including text, multimedia, user profiles, and group data.

**Backup and Disaster Recovery:**

Establish reliable backup and disaster recovery mechanisms to prevent data loss.

**User Experience:**

Optimize database performance for fast message delivery and media sharing.

**Data Analysis and Insights:**

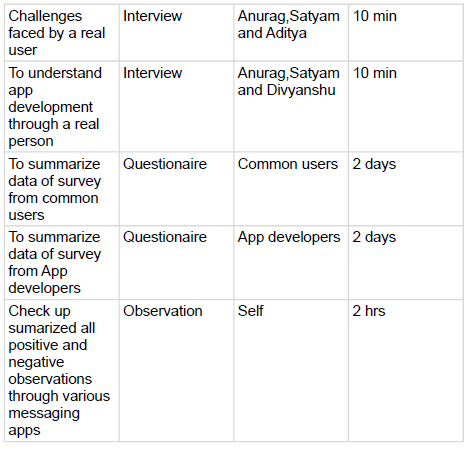
Balance privacy concerns with the ability to analyze user data for insights without compromising user privacy.

**Interoperability:**

Ensure compatibility with various platforms and devices for seamless communication.

**2.2 Fact finding chart**

****

****

**3. List of Requirements**

**Functional Requirements**

Functional requirements for Messaging App App development encompass the specific features and capabilities that the application should have to meet user needs and expectations. Here's a list of functional requirements for Messaging App App development:

**User Registration and Authentication:**

Users should be able to register with a valid phone number.

Authentication should be secure, possibly using SMS verification or other methods.

**User Profile Management:**

Users should be able to create and manage their profiles with profile pictures, status messages, and personal information.

**Contact Management:**

Users should be able to import contacts from their phone's address book.

Users should be able to search for and add contacts manually.

**Messaging:**

Users should be able to send text messages to individual contacts or groups.

Messages should support rich text, emojis, stickers, and multimedia attachments (images, videos, documents, voice notes, etc.).

Messages should be displayed in real-time with read receipts and typing indicators.

Users should have the ability to delete, edit, or forward messages.

Group messaging should support adding/removing members and group admin roles.

**Voice and Video Calls:**

Users should be able to make voice and video calls to their contacts.

Calls should support features like mute, speaker, and video on/off.

**Notifications:**

Users should receive push notifications for new messages and calls.

Notifications should be customizable in terms of sounds, vibrations, and previews.

**Privacy and Security:**

End-to-end encryption should be implemented for all messages and calls.

Users should have control over their privacy settings, such as who can see their last seen status, profile picture, and status message.

**Backup and Restore:**

Users should be able to back up their chat history and multimedia content to cloud storage (e.g., Google Drive or iCloud).

Chat history should be easily restorable when switching devices or reinstalling the app.

**Status Updates:**

Users should be able to post status updates (text, photos, videos) visible to their contacts for a specified duration.

**Location Sharing:**

Users should be able to share their real-time location with contacts.

Users should be able to share a location on the map.

**Search Functionality:**

Users should be able to search for specific messages, contacts, or groups within the app.

**Multilingual Support:**

The app should support multiple languages to cater to a global audience.

**Accessibility Features:**

The app should be designed with accessibility features to accommodate users with disabilities.

**Blocked Contacts:**

Users should be able to block and unblock contacts to prevent unwanted communication.

**Media Gallery:**

Users should have access to a media gallery where they can view and manage all media files sent and received.

**Integration with Camera and Gallery:**

Users should be able to take photos and videos directly from the app and access their device's gallery for sharing media.

**Settings and Preferences:**

Users should have control over app settings, including notification preferences, account settings, and privacy options.

**App Lock and Security Pin:**

Users should have the option to secure their app with a PIN, fingerprint, or other biometric authentication methods.

**Support and Help Center:**

Users should have access to a support and help center for troubleshooting and FAQs.

**In-App Updates:**

The app should support seamless updates to deliver new features, bug fixes, and security enhancements.

These functional requirements provide a comprehensive overview of the features and capabilities expected from a Messaging App-like messaging application. Development teams can use this list as a foundation when building or enhancing such an app.

**Non Functional Requirements**

Non-functional requirements for Messaging Appapp development encompass attributes that describe how the application should perform rather than specific features. They define the quality, performance, and usability aspects of the app. Here's a list of non-functional requirements for Messaging Appapp development:

**Performance:**

Response Time: The app should have low latency in message delivery and responsiveness to user actions.

Scalability: The system should scale gracefully to accommodate a growing number of users and messages.

**Reliability:**

The app should have high availability and be operational 24/7 with minimal downtime.

Messages and data should be highly reliable and not prone to loss or corruption.

**Security:**

All communications must be end-to-end encrypted to ensure user privacy.

User data, including messages and media, should be securely stored and transmitted.

Protection against common security threats like hacking, phishing, and malware should be implemented.

**Compatibility:**

The app should be compatible with a wide range of devices and operating systems (iOS, Android, web, etc.).

It should work well on both smartphones and tablets.

**Usability and Accessibility:**

The app should have an intuitive and user-friendly interface.

Accessibility features should be in place to accommodate users with disabilities.

It should be consistent in its design and navigation across different platforms.

**Scalability:**

The app should be designed to handle a large number of users and messages efficiently.

The database and server infrastructure should be scalable to meet demand.

**Network Resilience:**

The app should handle variable network conditions, including low bandwidth and intermittent connectivity.

Messages should be sent and received even in challenging network environments.

**Resource Efficiency:**

The app should be optimized for resource usage, including CPU, memory, and battery life.

It should not drain device resources excessively.

**Compliance and Regulation:**

The app should comply with local and international regulations regarding data privacy and telecommunications.

It should provide tools for law enforcement in accordance with legal requirements.

**Load Handling:**

The app should handle spikes in user activity, such as during holidays or special events, without performance degradation.

**Internationalization and Localization:**

The app should support multiple languages and cultural preferences.

Dates, times, and currency should be displayed according to the user's locale.

**Data Privacy and Retention:**

The app should allow users to control their data, including options for data deletion and account deactivation.

Data retention policies should comply with privacy regulations.

**Error Handling and Reporting:**

The app should provide meaningful error messages to users.

Critical errors should be reported to administrators for prompt resolution.

**Authentication and Authorization:**

Secure methods for user authentication and authorization should be implemented.

User sessions should be managed securely.

**Cross-Platform Consistency:**

The user experience should be consistent across different platforms (iOS, Android, web).

Features and functionalities should work uniformly.

**Regulatory Compliance:**

The app should adhere to applicable regulations and standards, such as GDPR or HIPAA, depending on the user base and data handled.

**Performance Metrics:**

Performance metrics should be regularly monitored and optimized for factors like server response time, app launch time, and data transfer speed.

**4. User Categories and Privilages**

**4.1 User Categories and Basic Descriptions:**

**Regular User:**

Regular users are the standard Messaging App users who send and receive messages, make calls, and interact with contacts.

They have basic access to all core features of Messaging App.

**Group Admin:**

Group admins are regular users who have additional responsibilities within group chats.

They can add or remove members, change the group's name and profile picture, and manage group settings.

They can delete messages and have some moderation control over group content.

**Business Account:**

Business accounts are used by businesses and organizations to interact with customers.

They can set up a business profile with information such as business name, description, and contact details.

Business accounts have access to business-related features like automated messages and customer support tools.

**Verified Business Account:**

Verified business accounts are distinguished from regular business accounts by having a verified badge.

They gain increased trust from users and may have access to additional features, like API integrations.

**Support Agent:**

Support agents are employed by businesses to provide customer support through Messaging App.

They have access to tools and privileges that allow them to respond to customer inquiries and manage support-related conversations.

**4.2. Privileges for Each User Category:**

**Regular User:**

Messaging: They can send and receive personal messages with contacts and participate in group chats.

Calling: They have the capability to initiate and receive both voice and video calls.

Profile Management: They can set up their profile picture, status, and bio.

Access to Features: Regular users can utilize all basic features of the app, such as sending multimedia (photos, videos, voice notes), using emojis and stickers, and more.

**Group Admin:**

Member Management: They have the authority to add new members to the group or remove existing members. This provides control over group membership.

Group Settings: They can modify the group's name, profile picture, and adjust other group-related settings.

Content Moderation: Group admins have the right to delete any message within the group chat, ensuring that the content remains appropriate and in line with group guidelines.

Permissions Control: They might have the ability to determine who can send messages or change group details.

**Business Account:**

Business Profile: They can set up a specialized profile showcasing their business information, such as opening hours, contact details, and a brief description.

Automated Messaging: These accounts may have the ability to set automated messages, for instances like immediate customer inquiries or out-of-office replies.

Customer Interaction: They can send and receive messages with customers, offering services, promotions, or addressing queries.

Access to Business Tools: Might include insights on message metrics, customer feedback tools, or promotional features.

**Verified Business Account:**

Verification Badge: These accounts have a distinct badge, signaling to users that the account's authenticity has been checked and verified by Messaging App, increasing trust.

Premium Features: Might have access to advanced tools or API integrations that regular business accounts don't, providing enhanced functionality.

Priority Support: Given the verified status, these accounts might receive priority customer support or a dedicated account manager.

**Support Agent:**

Customer Support Tools: They have specialized tools to manage and respond to customer inquiries, maybe in a ticket-based or organized manner.

Conversation Management: Support agents can archive, categorize, or escalate conversations based on the inquiry's nature.

Team Collaboration: They might collaborate with other agents or departments within the business to address specific customer concerns or issues.

Response Templates: Can use pre-defined templates for frequently asked questions or issues to speed up response times.

By clearly defining and understanding these privileges, it becomes easier to structure the roles and functionalities within the messaging app, ensuring a streamlined experience for all user categories.

**5. Assumptions for this database**

Designing a database for Messaging App or any messaging platform requires careful consideration of various assumptions and requirements. Below are some assumptions that you might consider while designing a Messaging App database:

**User Authentication:**

Users are required to create accounts with unique usernames or phone numbers.

Passwords are securely hashed and stored.

User authentication is a critical component for data privacy and security.

**Message Types:**

Messages can be of different types, such as text, images, videos, documents, and voice messages.

**Message Storage:**

Messages are stored with metadata, including sender, receiver, timestamp, and message type.

Messages may need to be encrypted for security and privacy.

**Contacts:**

Users can have a list of contacts with their information.

Contacts can be individuals or groups.

**Group Chats:**

Group chats consist of multiple participants.

Group members can be added or removed by administrators.

**Message Status:**

Messages may have different statuses (e.g., sent, delivered, read) for tracking delivery and read receipts.

**Notification:**

Users may receive push notifications for new messages.

Notification preferences can be customized by users.

**Offline Messaging:**

Messages are stored and delivered when the recipient is offline.

Offline messages are queued and delivered when the recipient is back online.

**Multimedia Storage:**

Multimedia files (e.g., images, videos) are stored and associated with messages.

**Archiving and Backup:**

Users may have the option to archive or back up their chat history.

Backup and archiving frequency and retention policies should be considered.

**Privacy Settings:**

Users can customize their privacy settings, such as who can see their online status, profile picture, and last seen status.

**User Metadata:**

User profiles may contain information such as profile pictures, status messages, and user-generated content.

**End-to-End Encryption:**

End-to-end encryption is implemented to ensure the security and privacy of messages.

**Reporting and Moderation:**

Mechanisms for reporting and moderating inappropriate content or abusive behavior should be in place.

**User Activity Logging:**

User activities and interactions with the platform are logged for analytics and troubleshooting.

**Scalability:**

The database should be designed to handle a large number of users and messages as the platform scales.

**Redundancy and High Availability:**

The database should be redundant and have high availability to minimize downtime.

**Data Backup and Recovery:**

Regular data backups and a recovery plan are essential to prevent data loss in case of failures.

**Compliance and Legal Considerations:**

Compliance with data protection laws and legal requirements is essential, especially for data retention and user data access requests.

**Localization:**

Support for multiple languages and regional preferences should be considered.

These assumptions provide a starting point for designing a Messaging App database, but the specific requirements may vary depending on the platform's features and goals. It's crucial to continuously evaluate and update the database design to meet evolving user needs and ensure data security and integrity.

**6. Business Constraints**

When developing a messaging app's database, there are several business constraints that need to be considered:

**1. Data Storage Costs:**

With billions of messages exchanged daily, the cost of storing this data becomes significant. The design must be efficient to reduce costs without compromising on performance

**2. Scalability:**

The database must be scalable to accommodate growing numbers of users and messages. The system should handle peak loads without performance degradation.

**3. Data Retention Policies:**

Based on regional regulations and internal policies, there might be constraints on how long certain types of data (like chat logs) can be retained. Older data might need to be archived or purged regularly.

**4. Cross-Platform Consistency:**

If the messaging app is available on multiple platforms (iOS, Android, Web), the database should provide consistent data across all of them.

**5. Data Migration and Integration:**

Over time, there might be a need to migrate data to a new system or integrate with other services. This requires a database design that supports easy migration and integration.

**6. Redundancy and Failover:**

To ensure high availability, the database might need to be replicated across multiple locations. This can increase costs but is necessary to prevent downtimes.

**7. Maintenance and Updates:**

The database software, hardware, and associated systems will periodically need updates. These updates should happen with minimal disruption to the service.

**8. Licensing and Third-party Dependencies:**

Depending on the choice of database software, there might be licensing fees. Additionally, reliance on third-party services or software can introduce another layer of constraints and costs.

Incorporating these constraints in the initial planning and design phase will ensure that the messaging app's database is robust, efficient, and compliant with necessary regulations and requirements.

1. **Table 1 :- Nouns and Verbs from the description**

|  |  |
| --- | --- |
| Nouns | Verbs |
| Messaging App | will be, is |
| service | will let, exchange, use, require, serve |
| users | call, exchange, use, serve |
| iPhone | uses |
| Android smartphones | uses |
| Mac | use |
| Windows PC | use |
| connection | use |
| Apple iMessage | require |
| Messages | require |
| networks |  |
| Service (SMS) |  |
| platform | will serve |
| millions | serve |
| data | Managing, ensuring |
| volume | Managing |
| content | exchanging, ensuring |
| challenge | is |
| case study | will search |
| database | will search |
| practices | employed |
| users | have |
| service | to provide |
| Messaging App | will use, has, plays |
| Wi-Fi | will use |
| users | will have, will not have |
| data plans | will not have |
| calls | will have, rely heavily |
| text messaging | will have, rely heavily |
| future | will not have |
| cross-platform feature | will also be |
| people | will have |
| family | will have |
| social media | has, rely heavily |
| communication | has, rely heavily |
| productivity tools | rely heavily |
| database management systems | rely heavily |
| landscape | have |
| applications | has, rely heavily |
| role | plays |
| success | plays |
| case study | delves into |
| messaging app | manages, |
| modules | require |
| User Management Module | is crucial |
| user registration | encompasses, sign up |
| authentication | encompasses, validate |
| users | Participate |
| user profile information | encompasses, manage |
| usernames | manage |
| profile pictures | manages |
| status | manage |
| bio |  |
| user settings | incorporate, adjusting |
| preferences | adjusting |
| read receipts | adjusting |
| backup settings |  |
| dark mode |  |
| syncing | be |
| contacts | syncing, managing |
| device | syncing, managing |
| heart of the app | lies in |
| Messaging Module | should efficiently handle, need, manage |
| text messages | handle, need, manage, send, receive, display |
| multimedia content | need, manage, handle, be |
| images | manage |
| videos | manage |
| audio files | manage |
| documents | manage |
| voice notes | allow, record, send, playback |
| message status | track |
| search functionality | be beneficial, allowing |
| chats | filter |
| Call Management Module | is, is responsible for, would take care of |
| voice and video calls | is responsible for |
| recording call data | would take care of |
| duration | would take care of |
| metadata | would take care of |
| Notifications | are, are equally vital |
| incoming or missed calls | are equally vital |
| Groups | form, is a must |
| Group Management Module | is a must, would oversee, handle, customize, set |
| creation and management | would oversee, handle |
| group chats | handle |
| group details | customize |
| permissions | set |
| Data Backup & Recovery Module | would provide, ensuring |
| local and cloud backup options | would provide, ensuring |
| Data | ensuring |
| age of cyber threats | is |
| Security & Encryption Module | is, would encrypt, provide, allow |
| data end-to-end | would encrypt |
| two-factor authentication | provide |
| privacy settings | allow, manage |
| Multimedia Management Module | is, is pivotal, would display, ensure, facilitate |
| shared media | would display |
| efficient media transmission | ensure |
| file sharing | facilitate |
| Notifications & Alerts Module | is fundamental, would manage |
| alerts | would manage |
| messages | would manage |
| calls | would manage |
| app updates | would manage |
| Connectivity Module | would manage, is |
| switch between Wi-Fi and cellular data | would manage, handle |
| message transmission | would manage, handle |
| offline periods | would manage, handle |
| user experience | is enhanced, allowing |
| User Support & Feedback Module | is, would guide, collect, allowing |
| app functionalities | would guide |
| valuable feedback | collect, allowing |
| continuous improvement | allowing |
| developers | develop |
| messaging application | can ensure |
| case study | explores, is |
| details | explores |
| Messaging App | is |
| database | handle, take care of, is, is tailored |
| challenges | often face |
| data consistency | ensuring |
| sensitive information | protecting |
| database performance | optimizing, deliver |
| user experience | deliver |
| study | will emphasize, is |
| need | will emphasize |
| database schema | is tailored |
| requirements | tailored |
| application | tailored |
| data indexing | emphasize, is |
| caching | emphasize, is |
| query optimization | emphasize, is |
| Security | is, will explore, discuss, safeguard |
| mobile app development | is, will explore, discuss |
| case study | will explore |
| security measures | will explore, discuss, include |
| encryption | have |
| access control | will explore, safeguard |
| authentication |  |
| user data | safeguard, prevent |
| strategies | will discuss, prevent |
| data backups | stored |
| disaster recovery | will discuss, prevent |
| scalability | is, is another, grow |
| mobile applications | is, grow |
| techniques | will learn, |
| horizontal scaling |  |
| vertical scaling |  |
| load balancing | will learn, handle |
| database sharding | will learn, handle |
| Message Persistence | is, is a critical aspect, allowing |
| Messaging App | is, allowing |
| chat history | allowing |
| multimedia content | may include |
| platforms | presents, need to be addressed |
| iOS | presents, need to be addressed |
| Android |  |
| web |  |
| challenges |  |
| seamless user experience | need to be addressed |
| Data Backup and Recovery | are, expected, rely on, ensures |
| functionalities | are, expected |
| users | are, expected, rely on |
| ability | rely on, ensures |
| chat history | back up, recover |
| app | reinstalling, ensures |
| feature | ensures |
| conversations | losing |
| Real-time Messaging | is, is at the core of |
| functionality | is at the core of |
| Messages | stored |
| data transmission | requiring |
| synchronization mechanisms | requiring |
| communication | ensuring, without delays |
| delays | without |
| volume | is, generated daily, accommodate |
| Messaging App | is, generated daily, imperative |
| storage solutions | are, imperative |
| Scalability | is, is key to maintaining |
| optimal performance | is key to maintaining |
| diversity | necessitates |
| data types | necessitates |
| app | necessitates, must handle |
| storage and retrieval mechanisms | necessitates |
| text messages | must handle |
| multimedia content |  |
| platform |  |
| range | must handle |
| data formats | must handle efficiently |
| data retention policies | are, are essential, manage |
| user data | are essential, manage |
| compliance | ensure, with regulatory requirements |
| duration | maintaining, securely disposing of |
| Data Access Control | Belongs to, Implementing |
| user privacy | is paramount |
| data security | is paramount |
| access controls | Implementing ensures |
| authorized personnel | can access |
| sensitive user data | can access, bolstering |
| security posture | bolstering |
| Messaging App | of the Messaging App |
| Messaging App | employs, utilizes, is committed, relies |
| database management strategies | employs |
| operation | ensures, compromising |
| services | ensures, compromising |
| key techniques | utilizes, is |
| sharding | involves, distributing, manages |
| data | distributing |
| database servers | belongs |
| scalability | manages, allowing |
| app | allowing, handle |
| volumes of data | handle, compromising |
| performance | compromising |
| user privacy | is committed, safeguarding |
| messages | is committed, safeguarding |
| end-to-end encryption | ensures, can decrypt, enhancing |
| Signal Protocol | utilizes |
| recipient | can decrypt, read |
| multimedia content | relies, ensuring, compress, deliver |
| content delivery networks | relies |
| CDNs | help, compress, deliver |
| media files | compress, deliver |
| user experience | ensuring |
| Messaging App | provides, involves, allowing, offers, relies |
| user chat history | managing, access, switching, securely store |
| local device storage | provides |
| cloud backups | provides |
| coordination | involves, ensuring |
| data synchronization | involves, allowing |
| platforms | allowing, switching |
| data | securely store, ensuring, easily retrieve |
| cloud services | securely store |
| Google Drive | securely store |
| iCloud | securely store |
| functionality | simplifies, ensures |
| data recovery | simplifies, ensures |
| real-time messaging | relies |
| efficient infrastructure | relies |
| Web Sockets | incorporating |
| push notification systems | incorporating |
| technologies | incorporating |
| messages | delivered, contributing |
| user experience | contributing |
| app | places, implements, is committed |
| data management and security | places, is committed |
| emphasis | places |
| data retention policies | defining, adhering to |
| role-based access control (RBAC) | implements, regulate |
| data access | regulate |
| security standards | maintain |
| Messaging App | is committed |
| data protection regulations | is committed, to safeguard |
| user data and privacy | to safeguard |
| components | can add |
| design approach | is, resonates |
| platform | is, resonates |
| users | are, resonates |
| designers | design |
| developers | are, resonates |
| end-users | are, resonates |
| User Interface (UI) | can enhance |
| User Experience (UX) | can enhance |
| user engagement | can significantly enhance |
| satisfaction | can significantly enhance |
| users | fosters, keeps |
| platform | is, easy to navigate, use, fosters, keeps |
| experience | fosters, keeps |
| continuous improvement | foster, is essential |
| user involvement | foster |
| feedback and improvement system | implementing, allows |
| mechanism | allows |
| users | provide, makes |
| Attachments | include |
| database | provide, about, facilitates |
| features | provide, about |
| experience | provide, about, makes |
| feedback loop | facilitates |
| enhancements | facilitates, makes |
| new users | is, is crucial, helps, become |
| onboarding | is crucial |
| sessions | is crucial |
| tutorials | is crucial |
| System | Host |
| educational resources | helps, become |
| platform | is, boosting, catering |
| confidence | boosting |
| versatility | expanding |
| utility | expanding |
| range of user needs | catering to |
| feature | incorporating, can benefit |
| messaging app trends | aggregates |
| market demands | aggregates |
| Conversations | holds |
| developers | can benefit |
| users | get |
| landscape | evolving |
| sustainability | gains, prioritizing |
| performance | prioritizing |
| design principles | prioritizing |
| factor | can be, aligns |
| emphasis | growing |
| environmental responsibility | on |
| platform | aims, accommodating |
| languages | accommodating |
| currencies | accommodating |
| design preferences | should be considered |
| localization | can enhance |
| globalization |  |
| user accessibility | can enhance |
| inclusivity | can enhance |
| sense of community | building, fosters |
| networking | building |
| users | fosters, receive, addressing |
| developers | receive |
| forum | creating |
| space | creating |
| idea sharing | creating, fosters |
| discussions | creating, fosters |
| peer support | creating, fosters |
| collaborative ecosystem | fosters |
| system checks | are, are imperative |
| maintenance | are imperative |
| updates | are imperative |
| smooth operation | are imperative |
| incorporation | are imperative |
| latest features | are imperative |
| platform | creating, retaining |
| competitive | keeping |
| efficient | keeping |
| emergency support | Providing, ensuring |
| customer service channels | Providing, ensuring |
| assistance | ensuring |
| technical issues | addressing, effectively |
| concerns | addressing, effectively |
| dashboard | allowing, based, making |
| workspace | based, making, tailored |
| preferences | based, making, tailored |
| layer of personalization | adds, making |
| comprehensive | creates |
| user-centric database platform | creates, can attract, retain |
| diverse user base | can attract, retain |
| Managing | is, provide |
| database | is, invest, ensuring, will have |
| complex undertaking | is |
| volume | is |
| diversity | is |
| data | generated, remains |
| platform | is |
| Companies | must invest, will be provided |
| organizations | must invest, will be provided |
| scalable infrastructure | must invest, will be provided |
| security measures | must invest, will be provided |
| compliance | must invest, will be provided |
| data protection regulations | must invest, will be provided |
| Real-world case studies | provide, will be provided |
| insights | provide |
| practices | provide |
| cloud-based database solutions | will introduce |
| synchronization mechanisms | will introduce |
| examples | will be provided |
| technologies | will be provided |
| challenges | will have |
| best practices | will have |
| audience | will have |
| understanding | will have |
| mobile applications | managing |
| decisions | to make, to implement |
| solutions | to make, to implement |
| projects | in their own |

**Table 2 :- Nouns For Candidate Entity Set**

|  |  |
| --- | --- |
| Users | Users are one of the main entities who are one of the key components in app database management |
|
| Messages | Messages are core entities in a messaging app, storing communication content between users. They encompass text or multimedia exchanges and contain crucial data like message content, sender, recipient, timestamps, etc. |
|
| Calls | Calls represent voice or video communication between users. Including this entity allows the app to log call-related information such as call type, duration, participants (sender/receiver), enabling call history and call-related features. |
|
| Chat | Chats signify ongoing or archived conversations between users. They hold metadata for chats, aiding in organizing messages and managing communication threads between users. |
|
| Groupchat | Group chats enable communication among multiple users simultaneously. This entity stores details about group conversations, such as the group name, member list, creation date, and facilitates interaction between multiple users within a single chat context. |
| Status | Status indicates the current activity or presence status of users, such as uploading photos, videos and texts. It provides real-time information about user availability and is often utilized in messaging apps to display user activity to contacts. |
| Attachments | Attachments represent files, media, or documents exchanged within messages. This entity holds details about attached files, like file name, type, size, associated mess |
| Notification | Notifications are used to alert users about new messages, calls, or other activities within the app. Storing notification-related data aids in notifying users of new content or events, improving user engagement and ensuring they stay informed about app activities. |
| Attachments | In a messaging app, users can often send various types of media, such as images, videos, files, or links. |
| Data Retention Policies | Rules governing data storage duration. |
| Sharding | Data distribution technique. |
| Special Features | Includes GIFS, stickers, emojis etc for spicing up conversation |
| Messaging App | Central subject of the paragraph; represents the entire application system. |
| Profile | Contains user-specific information like names, profile pictures, status updates. |
| User Management Module | Manages user-specific functionalities like registration, authentication, and profile information. |
| Messaging Module | Manages functionalities related to sending, receiving, and displaying messages. |
| Call Management Module | Manages voice and video calls, recording data, and metadata. |
| Group Management Module | Manages group chats, details, and permissions. |
| Multimedia Management | Manages media sharing and display functionalities. |
| Notifications & Alerts | Manages alerts for messages, calls, and updates. |
| Connectivity Module | Handles connection between Wi-Fi and cellular data, and offline message transmission. |
| User Support & Feedback | Collects feedback and guides users on app functionalities. |
| Database | The primary storage component where all user data and messages are stored. |
| CDN (Content Delivery Network) | Helps in the efficient transmission of multimedia content. |
| Web Sockets | Technology ensuring real-time message delivery. |
| Push Notification System | Facilitates instant message |

**Table 3 :- Nouns For Candidate Attribute Set**

|  |  |
| --- | --- |
| **Nouns** | **Entities Likely to be Assigned** |
| Names | User, Profile |
| Profile Pictures | Profile |
| Status Updates | Profile |
| Text | Conversations |
| Voice Recordings | Conversations |
| Multimedia Content | Message, Multimedia |
| Chat Records | Group |
| Member Details | Group |
| Images | Attachments |
| Videos | Attachments, Conversations |
| Documents | Multimedia |
| Registration | User Management Module |
| Authentication | User Management Module, Security & Encryption |
| Sending | Messaging Module |
| Receiving | Messaging Module |
| Displaying | Messaging Module, Multimedia Management |
| Voice and Video Communication | Call Management Module |
| Backup Settings | Data Backup & Recovery |
| Data Safety | Data Backup & Recovery, Database |
| Data Privacy | Security & Encryption |
| UI/UX Design | Designers |
| Offline Message Transmission | Connectivity Module |
| Security | Encryption |
| Instant Message Notifications | Push Notification System |

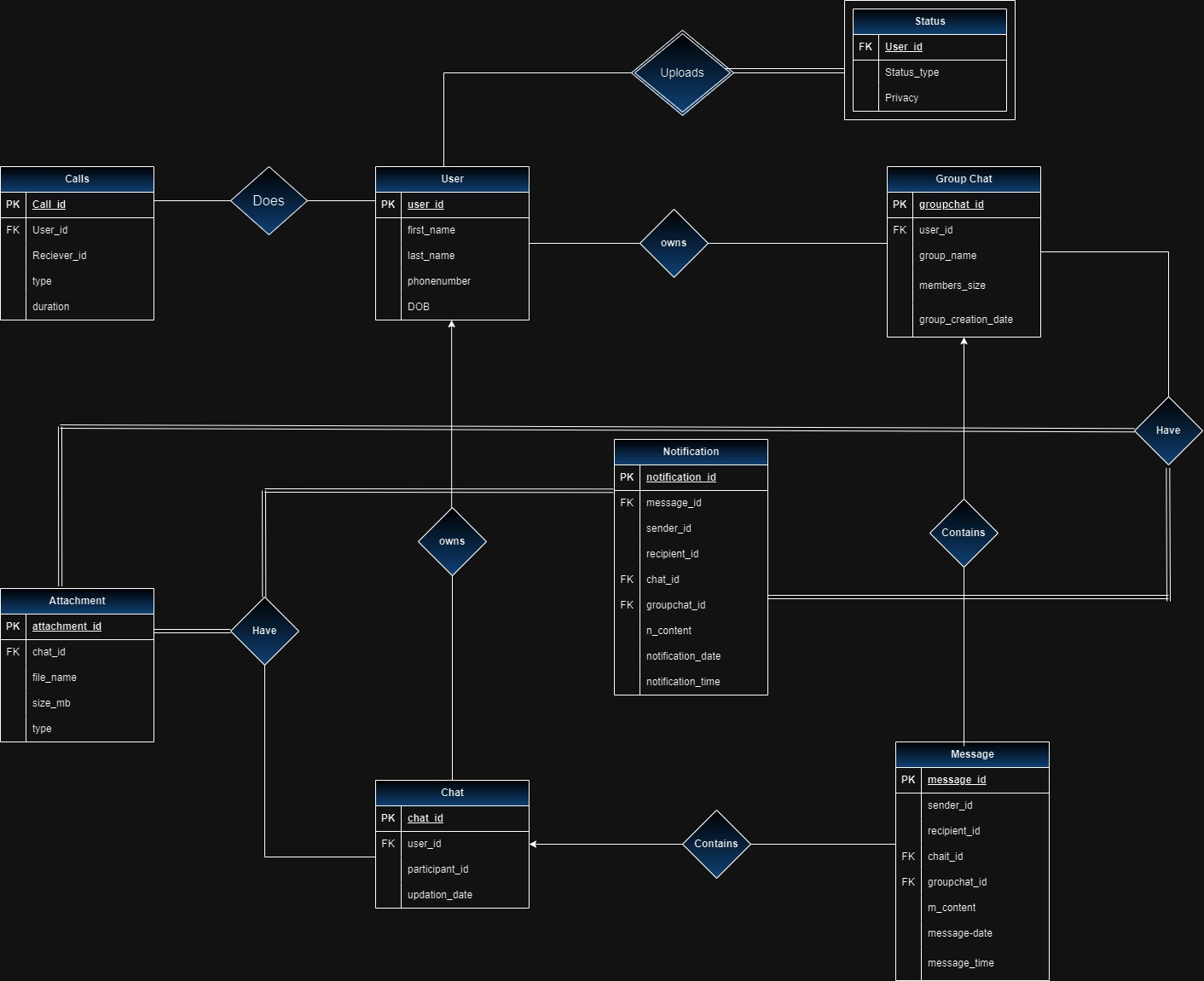
**Table 4 :- Rejected Noun List**

|  |  |
| --- | --- |
| **Nouns** | **Verbs** |
| service | Vague |
| iPhone | Irrelevant |
| Android smartphones | Irrelevant |
| Mac | Irrelevant |
| Windows PC | Irrelevant |
| connection | Irrelevant |
| Apple iMessage | Irrelevant |
| networks | Vague |
| platform | Vague |
| millions | Irrelevant |
| data | Repetitive |
| volume | Managing |
| content | Vague |
| challenge | Irrelevant |
| case study | Irrelevant |
| database | Vague |
| practices | employed |
| Messaging App | Repetitive |
| Wi-Fi | Repetitive |
| data plans | Repetitive |
| future | Repetitive |
| cross-platform feature | Irrelevant |
| people | Attribute |
| family | Repetitive |
| social media | Managing |
| communication | Vague |
| productivity tools | Irrelevant |
| database management systems | Irrelevant |
| landscape | Vague |
| applications | employed |
| role | Repetitive |
| success | Repetitive |
| case study | Irrelevant |
| messaging app | Repetitive |
| authentication | Irrelevant |
| user profile information | Irrelevant |
| status | Irrelevant |
| bio | Irrelevant |
| user settings | Irrelevant |
| preferences | Irrelevant |
| read receipts | Vague |
| dark mode | Irrelevant |
| syncing | Repetitive |
| contacts | Managing |
| heart of the app | Vague |
| Messaging Module | Irrelevant |
| multimedia content | Irrelevant |
| message status | Irrelevant |
| search functionality | Irrelevant |
| Call Management Module | Irrelevant |
| voice and video calls | Irrelevant |
| recording call data | Irrelevant |
| duration | Irrelevant |
| metadata | Vague |
| creation and management | Vague |
| permissions | Irrelevant |
| age of cyber threats | Repetitive |
| efficient media transmission | Managing |
| Notifications & Alerts Module | Vague |
| alerts | Irrelevant |
| app updates | Irrelevant |
| Connectivity Module | Vague |
| switch between Wi-Fi and cellular data | employed |
| message transmission | Repetitive |
| offline periods | Repetitive |
| user experience | Repetitive |
| User Support & Feedback Module | Repetitive |
| app functionalities | Irrelevant |
| valuable feedback | Repetitive |
| continuous improvement | Repetitive |
| messaging application | Irrelevant |
| case study | Irrelevant |
| details | Repetitive |
| database | Repetitive |
| challenges | Irrelevant |
| data consistency | Irrelevant |
| sensitive information | Repetitive |
| database performance | Repetitive |
| user experience | Irrelevant |
| study | Irrelevant |
| need | Repetitive |
| database schema | Repetitive |
| requirements | Irrelevant |
| application | Irrelevant |
| data indexing | Repetitive |
| caching | Repetitive |
| query optimization | Irrelevant |
| case study | Irrelevant |
| security measures | Repetitive |
| access control | Repetitive |
| authentication | Irrelevant |
| strategies | Irrelevant |
| data backups | Repetitive |
| disaster recovery | Repetitive |
| scalability | Irrelevant |
| mobile applications | Irrelevant |
| techniques | Repetitive |
| horizontal scaling | Repetitive |
| vertical scaling | Irrelevant |
| load balancing | Irrelevant |
| database sharding | Repetitive |
| Message Persistence | Repetitive |
| chat history | Irrelevant |
| multimedia content | Irrelevant |
| platforms | Repetitive |
| iOS | Repetitive |
| Android | Irrelevant |
| web | Irrelevant |
| challenges | Repetitive |
| seamless user experience | Repetitive |
| Data Backup and Recovery | Irrelevant |
| functionalities | Irrelevant |
| ability | Repetitive |
| chat history | Repetitive |
| app | Irrelevant |
| data transmission | Irrelevant |
| synchronization mechanisms | Repetitive |
| communication | Repetitive |
| delays | Irrelevant |
| volume | Irrelevant |
| storage solutions | Repetitive |
| diversity | Repetitive |
| data types | Irrelevant |
| storage and retrieval mechanisms | Vague |
| range | must handle |
| data formats | Attribute |
| Data Access Control | Attribute |
| access controls | Attribute |
| authorized personnel | Attribute |
| sensitive user data | Attribute |
| security posture | Vague |
| Messaging App | Repetitive |
| Messaging App | Repetitive |
| database management strategies | Repetitive |
| operation | Repetitive |
| key techniques | Irrelevant |
| data | Irrelevant |
| database servers | Repetitive |
| app | Repetitive |
| performance | Repetitive |
| Signal Protocol | Repetitive |
| recipient | Repetitive |
| multimedia content | Irrelevant |
| content delivery networks | Irrelevant |
| CDNs | Repetitive |
| media files | Repetitive |
| Messaging App | Repetitive |
| local device storage | Repetitive |
| cloud backups | Irrelevant |
| coordination | Irrelevant |
| data synchronization | Repetitive |
| platforms | Repetitive |
| data | Repetitive |
| cloud services | Repetitive |
| Google Drive | Irrelevant |
| iCloud | Irrelevant |
| data recovery | Repetitive |
| efficient infrastructure | Repetitive |
| Web Sockets | Repetitive |
| push notification systems | Repetitive |
| technologies | Irrelevant |
| app | Irrelevant |
| data management and security | Repetitive |
| emphasis | Repetitive |
| data retention policies | Repetitive |
| role-based access control (RBAC) | Repetitive |
| data access | Irrelevant |
| security standards | Irrelevant |
| Messaging App | Repetitive |
| data protection regulations | Attribute |
| user data and privacy | Attribute |
| components | Irrelevant |
| design approach | Irrelevant |
| platform | Repetitive |
| end-users | Repetitive |
| User Interface (UI) | Repetitive |
| User Experience (UX) | Attribute |
| user engagement | Repetitive |
| satisfaction | Repetitive |
| experience | Attribute |
| user involvement | Attribute |
| feedback and improvement system | Irrelevant |
| insights | Irrelevant |
| database | Repetitive |
| features | Repetitive |
| experience | Repetitive |
| feedback loop | Repetitive |
| enhancements | Irrelevant |
| onboarding | Irrelevant |
| sessions | Repetitive |
| tutorials | Repetitive |
| webinars | Irrelevant |
| educational resources | Vague |
| confidence | Repetitive |
| versatility | Irrelevant |
| utility | Irrelevant |
| range of user needs | must handle |
| messaging app trends | Attribute |
| market demands | Attribute |
| new technologies | Attribute |
| landscape | Attribute |
| sustainability | Attribute |
| performance | Repetitive |
| factor | Repetitive |
| emphasis | Repetitive |
| environmental responsibility | Repetitive |
| platform | Repetitive |
| currencies | Irrelevant |
| design preferences | Irrelevant |
| localization | Vague |
| globalization | Irrelevant |
| inclusivity | Irrelevant |
| sense of community | Repetitive |
| networking | Repetitive |
| forum | Irrelevant |
| space | Irrelevant |
| idea sharing | Repetitive |
| discussions | Repetitive |
| peer support | Irrelevant |
| collaborative ecosystem | Vague |
| system checks | Repetitive |
| maintenance | Repetitive |
| updates | Irrelevant |
| smooth operation | Irrelevant |
| incorporation | Repetitive |
| latest features | Repetitive |
| platform | Irrelevant |
| competitive | Irrelevant |
| efficient | Repetitive |
| emergency support | Repetitive |
| customer service channels | Irrelevant |
| assistance | Irrelevant |
| technical issues | Repetitive |
| concerns | Repetitive |
| dashboard | Irrelevant |
| workspace | Irrelevant |
| preferences | Repetitive |
| layer of personalization | Repetitive |
| comprehensive | Irrelevant |
| user-centric database platform | Irrelevant |
| diverse user base | Repetitive |
| Managing | Repetitive |
| database | Irrelevant |
| complex undertaking | Irrelevant |
| volume | Repetitive |
| diversity | Repetitive |
| data | Irrelevant |
| platform | Irrelevant |
| Companies | Repetitive |
| organizations | Repetitive |
| scalable infrastructure | Irrelevant |
| compliance | Irrelevant |
| data protection regulations | Repetitive |
| Real-world case studies | Repetitive |
| insights | Irrelevant |
| practices | Irrelevant |
| cloud-based database solutions | Vague |
| synchronization mechanisms | Vague |
| examples | Irrelevant |
| technologies | Irrelevant |
| challenges | Repetitive |
| best practices | Repetitive |
| audience | Irrelevant |
| understanding | Irrelevant |
| mobile applications | Repetitive |
| decisions | Repetitive |
| solutions | Irrelevant |
| projects | Irrelevant |

**Table 5 :- Final Entity Set**

|  |  |
| --- | --- |
| **NOUNS** | **List of Attributes** |
| **Users** | user\_id  first\_name  last\_name  phonenumber  DOB |
|
| **Chat** | Chat\_ID  User\_id  Participant\_name  Updation\_date |
|
| **Groupchat** | Groupchat\_id  User\_id  Group\_name  Member\_size  Group\_creation\_date |
| **Messages** | Message\_id  Sender\_id  recipient\_id  attachment\_id  m\_content  message\_date  message\_time |
| **Calls** | Call\_id  User\_id  Reciever\_id  Type  Duration |
| **Status** | user\_id  status\_type  privacy |
| **Attachments** | Attachment\_Id  Message\_id  File\_name  Type  Size\_mb |
| **Notification** | notification\_id  message\_id  sender\_id  recipient\_id  notification\_date  notification\_time |

**ER Diagram**



**Redundancies**

1. **Chat Table:**
   * **participant\_id** column is not specified correctly. It seems there should be multiple participants in a chat, so you might want to restructure this to store multiple participant IDs. It could be a separate table that links multiple users to a chat ID, or if it's a one-to-one chat, you might not need this column.
   * The **upadation\_date** column might be misspelled. It could be ended as **update\_date**.
2. **GroupChat Table:**
   * The **user\_id** column in the **GroupChat** table might indicate a user's association with a group. However, if this table represents group-level information, having a single **user\_id** doesn't seem to reflect multiple users in a group chat.
3. **Messages Table:**
   * The **sender\_id** and **recipient\_id** columns might relate to users involved in a conversation. However, if a message is a part of a group chat, you might not need a recipient ID since multiple users might receive the message.
   * The **attachment\_id** might be unnecessary if a message can exist without an attachment. If an attachment is optional, consider making this column nullable.
4. **Attachment Table:**
   * The **message\_id** column in the **Attachment** table links to a message. However, since an attachment can exist independently of a message (if it's sent separately or before a message), this dependency might not be necessary.
   * **file\_name**, **size\_mb**, and **type** columns could be moved to the **Messages** table if an attachment always accompanies a message.
5. **Notification Table:**
   * The **message\_id** column might imply that notifications are directly tied to a specific message. If notifications can be more general or linked to various events, this association might be redundant.
   * The **sender\_id** and **recipient\_id** columns might overlap with the information present in the Messages table, indicating the sender and recipient of a message.
6. **Calls Table:**
   * The **user\_id** and **receiver\_id** columns might duplicate information already present in the Messages or Users table if they represent the caller and receiver of a call.
7. **Status Table:**
   * The **user\_id** column might duplicate information already present in the Users table. Depending on the purpose of the status, it could potentially be stored in the Users table directly.
8. **Naming Consistency:**
   * There are inconsistencies in naming conventions like **upadation\_date** in the **Chat** table. It should likely be **updation\_date**.

**Anomalies**

**Insert Anomalies:**

1. **Users Table:**
   * Inability to add a user without all the optional information (like phone number or date of birth). If these fields are mandatory, an insert operation might fail if any of these fields are left empty.
2. **GroupChat Table:**
   * A situation where a group chat cannot be created without a user associated with it. If a user is mandatory for a group chat, it might not be possible to create a group chat without assigning a user, potentially limiting the ability to create empty or "user-less" group chats.
3. **Messages Table:**
   * Inserting a message without an associated sender or recipient might not be allowed if these fields are mandatory, preventing the creation of messages without specified senders or recipients.

**Update Anomalies:**

1. **Users Table:**
   * Updating a user's information (e.g., first name or last name) might lead to inconsistencies if the same user's information needs to be updated in multiple places where it is referenced (e.g., in GroupChat or Messages).
2. **GroupChat Table:**
   * If a user's information changes (e.g., name), and it is denormalized in this table, updating a user's information might require updating multiple rows in the GroupChat table.

**Delete Anomalies:**

1. **Users Table:**
   * Deleting a user might lead to losing associated information such as their messages, group chats, statuses, notifications, and calls if proper cascading deletes are not implemented or if the database schema doesn’t handle cascading deletes appropriately.
2. **GroupChat Table:**
   * Deleting a user might pose issues in group chats if there is no provision to handle scenarios where a user's deletion needs to be managed without disrupting the entire group chat.

**1NF**

Users ( user\_id , first\_name , last\_name , phonenumber , DOB DATE );

**User**: Already in 1NF.

Chat ( chat\_id , user\_id , participant\_id, upadation\_date DATE );

**Chat**: Already in 1NF.

GroupChat ( groupchat\_id , user\_id group\_name , member\_size , group\_created DATE );

**Groupchat**:Already in 1NF.

Messages ( message\_id , sender\_id , recipient\_id , attachment\_id , m\_content TEXT, message\_date Date, message\_time TIME );

Messages - Already in 1NF with a composite .

Attachment ( attachment\_id , message\_id , file\_name , size\_mb , type );

Attachment - Already in 1NF with a composite .

Notification ( notification\_id , message\_id , sender\_id , recipient\_id , ncontent TEXT, notification\_date date, notification\_time time );

Notification - Already in 1NF with a candidate key .

Calls ( call\_id , user\_id , receiver\_id , call\_type , duration time );

Calls - Already in 1NF with a composite .

Status ( user\_id , status\_type , privacy );

Status - Already in 1NF with a candidate key .

**2NF**

Users ( user\_id , first\_name , last\_name , phonenumber , DOB DATE );

**User**: Already in 2NF as there are no partial dependencies

Chat ( chat\_id , user\_id , participant\_id, upadation\_date DATE );

**Chat**: Already in 2NF as there are no partial dependencies.

GroupChat ( groupchat\_id , user\_id group\_name , member\_size , group\_created DATE );

**Groupchat**: Already in 2NF as there are no partial dependencies

Messages ( message\_id , sender\_id , recipient\_id , attachment\_id , m\_content TEXT, message\_date Date, message\_time TIME );

**Messages** - Already in 2NF as there are no partial dependencies

Attachment ( attachment\_id , message\_id , file\_name , size\_mb , type );

**Attachment** - Already in 2NF as there are no partial dependencies.

Notification ( notification\_id , message\_id , sender\_id , recipient\_id , ncontent TEXT, notification\_date date, notification\_time time );

**Notification** - Already in 2NF as there are no partial dependencies.

Calls ( call\_id , user\_id , receiver\_id , call\_type , duration time );

**Calls** - Already in 2NF as there are no partial dependencies

Status ( Status\_id , user\_id , status\_type , privacy );

**Status** - Already in 2NF as there are no partial dependencies

**3NF**

Users ( user\_id , first\_name , last\_name , phonenumber , DOB DATE );

**User**: Already in 3NF as there are no transitive dependencies

Chat ( chat\_id , user\_id , participant\_id, upadation\_date DATE );

**Chat**: Already in 3NF as there are no transitive dependencies.

GroupChat ( groupchat\_id , user\_id group\_name , member\_size , group\_created DATE );

**Groupchat**: Already in 3NF as there are no transitive dependencies

Messages ( message\_id , sender\_id , recipient\_id , attachment\_id , m\_content TEXT, message\_date Date, message\_time TIME );

**Messages** - Already in 3NF as there are no transitive dependencies

Attachment ( attachment\_id , message\_id , file\_name , size\_mb , type );

**Attachment** - Already in 3NF as there are no transitive dependencies.

Notification ( notification\_id , message\_id , sender\_id , recipient\_id , ncontent TEXT, notification\_date date, notification\_time time );

**Notification** - Already in 3NF as there are no transitive dependencies.

Calls ( call\_id , user\_id , receiver\_id , call\_type , duration time );

**Calls** - Already in 3NF as there are no transitive dependencies

Status ( Status\_id , user\_id , status\_type , privacy );

**Status** - Already in 3NF as there are no transitive dependencies

* **BCNF:**

**User**: user\_id -> user\_id , first\_name , last\_name , phonenumber , DOB

No non-trivial functional dependencies here that don't involve a superkey.

**Groupchat** :groupCHAT\_id -> groupchat\_id , user\_id group\_name , member\_size , group\_created DATE No non-trivial functional dependencies.

**Chat**: ->chat\_id , user\_id , participant\_id, upadation\_date

No non-trivial functional dependencies.

**Messages**-> message\_id , sender\_id , recipient\_id , attachment\_id , m\_content TEXT, message\_date Date, message\_time

No non-trivial functional dependencies that don't involve a superkey.

**Attachment**-> attachment\_id , message\_id , file\_name , size\_mb , type

No non-trivial functional dependencies.

**Notification**-> ( notification\_id , message\_id , sender\_id , recipient\_id , ncontent TEXT, notification\_date date, notification\_time time );

No non-trivial functional dependencies.

**Calls**-> call\_id , user\_id , receiver\_id , call\_type , duration time

No non-trivial functional dependencies.

**Status** -> ( Status\_id , user\_id , status\_type , privacy );

No non-trivial functional dependencies.

**After BCNF we have The Final Relation as**

* Users ( user\_id , first\_name , last\_name , phonenumber , DOB DATE );
* Chat ( chat\_id , user\_id , participant\_id, upadation\_date DATE );
* GroupChat ( groupchat\_id , user\_id group\_name , member\_size , group\_created DATE );
* Messages ( message\_id , sender\_id , recipient\_id , attachment\_id , m\_content TEXT, message\_date Date, message\_time TIME );
* Attachment ( attachment\_id , message\_id , file\_name , size\_mb , type );
* Notification ( notification\_id , message\_id , sender\_id , recipient\_id , ncontent TEXT, notification\_date date, notification\_time time );
* Calls ( call\_id , user\_id , receiver\_id , call\_type , duration time );
* Status ( Status\_id , user\_id , status\_type , privacy );

1. **Database Schema**

* **Relational Schema**

1. Users (

User\_ID (PK) int ,

First\_name varchar(),

Last\_name varchar(),

PhoneNumber int ,

Date\_of\_barth date()

);

1. Messages(

Message\_id (PK) int ,

Sender\_id int ,

recipient\_id int ,

attachment\_id int ,

m\_content ,

message\_date,

message\_time

);

1. Calls(

Call\_id (PK) int ,

User\_id (FK) int ,

Reciever\_id int ,

Type,

Duration

);

1. Chat(

Chat\_ID (PK) int ,

User\_id (FK) int,

Participant\_name,

Updation\_date ,

);

1. Groupchat (

Groupchat\_id (PK) int ,

User\_id int,

Group\_name varchar(),

Member\_size int(),

Group\_creation\_date

);

1. Status(

user\_id (PK) int ,

status\_type int ,

privacy

);

1. Attachments(

Attachment\_Id (PK) int ,

Message\_id (FK) int,

File\_name varchar(),

Type ,

Size\_mb ,

);

1. Notification(

notification\_id (PK) int ,

message\_id,

sender\_id,

recipient\_id

notification\_date

notification\_time

);

1. **Write DDL Scripts.**

i. Create the database by writing all Create Table statements (DDL) to accommodate the new design, which is in 3NF/BCNF (removing your original version of relations)

ii. Define appropriate constraints of all types (domain, PK, FK, Referential )for these tables

iii. Create an instance of this new database by populating it using appropriate INSERT INTO statements scripts. Make sure that every table has at least 80-100 tuples.

CREATE TABLE Users (

user\_id INT PRIMARY KEY NOT NULL,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255),

phonenumber VARCHAR(20) NOT NULL,

DOB DATE NOT NULL

);

CREATE TABLE Chat (

chat\_id INT PRIMARY KEY NOT NULL,

user\_id int NOT NULL,

participant\_name varchar(20),

upadation\_date DATE

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

CREATE TABLE GroupChat (

groupchat\_id INT PRIMARY KEY NOT NULL,

user\_id int NOT NULL

group\_name VARCHAR(255),

member\_size INT,

group\_created DATE

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

CREATE TABLE Messages (

message\_id INT PRIMARY KEY NOT NULL,

sender\_id int NOT NULL,

recipient\_id int NOT NULL,

attachment\_id INT,

m\_content TEXT,

message\_date Date,

message\_time TIMESTAMP

);

CREATE TABLE Attachment (

attachment\_id INT PRIMARY KEY NOT NULL,

message\_id INT NOT NULL,

file\_name VARCHAR(255),

size\_mb INT,

type VARCHAR(50)

FOREIGN KEY (message\_id) REFERENCES Messages(message\_id)

);

CREATE TABLE Notification (

notification\_id INT PRIMARY KEY NOT NULL,

message\_id INT NOT NULL,

sender\_id INT NOT NULL,

recipient\_id INT NOT NULL,

ncontent TEXT,

notification\_date DATE,

notification\_time TIME,

FOREIGN KEY (message\_id) REFERENCES Messages(message\_id)

);

CREATE TABLE Calls (

call\_id INT PRIMARY KEY NOT NULL,

user\_id INT NOT NULL,

dialer\_id INT NOT NULL,

receiver\_id INT NOT NULL,

call\_type VARCHAR(50),

duration time

);

CREATE TABLE Status (

user\_id INT NOT NULL,

status\_type VARCHAR(255),

privacy VARCHAR(50)

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

);

INSERT INTO Users (user\_id, first\_name, last\_name, phonenumber, DOB)

VALUES

(1, 'John', 'Doe', '1234567890', '1990-01-15'),

(2, 'Jane', 'Smith', '9876543210', '1985-06-22'),

(3, 'Michael', 'Johnson', '5551234567', '1993-03-10'),

(4, 'Emily', 'Brown', '7778889999', '1988-12-05'),

(5, 'David', 'Wilson', '3334445555', '1995-09-28'),

(6, 'Sarah', 'Miller', '1112223333', '1992-04-19'),

(7, 'Robert', 'Davis', '4445556666', '1998-08-14'),

(8, 'Jessica', 'Martinez', '6667778888', '1987-11-25'),

(9, 'William', 'Lee', '2223334444', '1991-07-30'),

(10, 'Jennifer', 'Garcia', '8889990000', '1986-02-07'),

(11, 'Daniel', 'Harris', '5556667777', '1996-10-03'),

(12, 'Linda', 'Anderson', '9990001111', '1989-03-18'),

(13, 'Christopher', 'Jackson', '7771112222', '1994-05-12'),

(14, 'Mary', 'Taylor', '4448889999', '1997-06-26'),

(15, 'Matthew', 'Brown', '1115556666', '1990-09-01'),

(16, 'Patricia', 'Moore', '3337778888', '1988-12-30'),

(17, 'Joseph', 'White', '6662223333', '1993-04-09'),

(18, 'Elizabeth', 'Clark', '2224445555', '1995-07-15'),

(19, 'Andrew', 'Thomas', '5554443333', '1991-11-20'),

(20, 'Karen', 'Hall', '8881110000', '1992-01-28'),

(21, 'Suresh', 'Kumar', '7890123456', '1980-05-20'),

(22, 'Priya', 'Sharma', '9876543210', '1975-03-15'),

(23, 'Amit', 'Verma', '6541234567', '1983-09-10'),

(24, 'Smita', 'Patel', '9876549876', '1978-12-25'),

(25, 'Raj', 'Singh', '9871234567', '1987-02-08'),

(26, 'Geeta', 'Mishra', '9993335555', '1981-06-01'),

(27, 'Vikas', 'Gupta', '1112223333', '1990-07-12'),

(28, 'Meera', 'Sharma', '4445556666', '1994-11-05'),

(29, 'Alok', 'Sharma', '6667778888', '1989-03-30'),

(30, 'Anita', 'Verma', '2223334444', '1982-04-28'),

(31, 'Rahul', 'Kumar', '8889990000', '1976-12-15'),

(32, 'Neeta', 'Yadav', '5556667777', '1991-10-20'),

(33, 'Sanjay', 'Rajput', '7770001111', '1985-09-26'),

(34, 'Kavita', 'Gupta', '4445558888', '1988-04-01'),

(35, 'Vishal', 'Sharma', '1115556666', '1993-08-14'),

(36, 'Mala', 'Verma', '3337778888', '1979-07-25'),

(37, 'Rajesh', 'Kumar', '6662223333', '1984-01-03'),

(38, 'Anjali', 'Yadav', '5554445555', '1996-06-18'),

(39, 'Rakesh', 'Mishra', '7771113333', '1987-10-30'),

(40, 'Pooja', 'Sharma', '8881110000', '1982-04-22'),

(41, 'Arun', 'Gupta', '7896543210', '1994-03-05'),

(42, 'Neha', 'Kumar', '6549876543', '1988-01-10'),

(43, 'Rajat', 'Verma', '9879871234', '1982-11-15'),

(44, 'Anita', 'Singh', '9876549876', '1977-12-05'),

(45, 'Rahul', 'Sharma', '9871239876', '1989-09-28'),

(46, 'Neelam', 'Yadav', '9993335555', '1994-04-19'),

(47, 'Amit', 'Gupta', '1112223333', '1983-08-14'),

(48, 'Sunita', 'Sharma', '4445556666', '1978-11-25'),

(49, 'Alok', 'Mishra', '6667778888', '1991-07-30'),

(50, 'Meera', 'Yadav', '2223334444', '1986-02-07'),

(51, 'Amit', 'Kumar', '8889990000', '1991-10-03'),

(52, 'Kavita', 'Rajput', '5556667777', '1989-03-18'),

(53, 'Sanjay', 'Sharma', '7770001111', '1985-05-12'),

(54, 'Mala', 'Mishra', '4445558888', '1994-06-26'),

(55, 'Rajesh', 'Verma', '1115556666', '1990-09-01'),

(56, 'Vikas', 'Gupta', '3337778888', '1977-12-30'),

(57, 'Meera', 'Kumar', '6662223333', '1993-04-09'),

(58, 'Rahul', 'Yadav', '5554445555', '1995-07-15'),

(59, 'Neeta', 'Sharma', '7771113333', '1987-11-20'),

(60, 'Suresh', 'Mishra', '8881110000', '1989-01-28'),

(61, 'Priya', 'Kumar', '7896543210', '1980-05-20'),

(62, 'Amit', 'Verma', '6549876543', '1975-03-15'),

(63, 'Smita', 'Singh', '9876549876', '1978-12-25'),

(64, 'Raj', 'Sharma', '9871239876', '1987-02-08'),

(65, 'Geeta', 'Yadav', '9993335555', '1981-06-01'),

(66, 'Vikas', 'Gupta', '1112223333', '1984-10-01'),

(67, 'Meera', 'Sharma', '4445556666', '1983-12-15'),

(68, 'Alok', 'Kumar', '6667778888', '1977-09-10'),

(69, 'Anita', 'Verma', '2223334444', '1976-02-08'),

(70, 'Rahul', 'Sharma', '8889990000', '1975-07-01'),

(71, 'Neeta', 'Rajput', '5556667777', '1981-03-12'),

(72, 'Sanjay', 'Mishra', '7770001111', '1989-10-25'),

(73, 'Kavita', 'Gupta', '4445558888', '1984-12-01'),

(74, 'Vishal', 'Sharma', '1115556666', '1996-04-22'),

(75, 'Mala', 'Verma', '3337778888', '1994-11-05'),

(76, 'Rajesh', 'Kumar', '6662223333', '1992-09-30'),

(77, 'Anjali', 'Yadav', '5554445555', '1990-07-20'),

(78, 'Rakesh', 'Mishra', '7771113333', '1987-03-18'),

(79, 'Pooja', 'Sharma', '8881110000', '1982-10-03'),

(80, 'Arun', 'Gupta', '7896543210', '1979-01-15'),

(81, 'Neha', 'Kumar', '6549876543', '1992-06-22'),

(82, 'Rajat', 'Verma', '9879871234', '1984-03-10'),

(83, 'Anita', 'Singh', '9876549876', '1977-12-05'),

(84, 'Rahul', 'Sharma', '9871239876', '1989-09-28'),

(85, 'Neelam', 'Yadav', '9993335555', '1994-04-19'),

(86, 'Amit', 'Gupta', '1112223333', '1983-08-14'),

(87, 'Sunita', 'Sharma', '4445556666', '1978-11-25'),

(88, 'Alok', 'Mishra', '6667778888', '1991-07-30'),

(89, 'Meera', 'Yadav', '2223334444', '1986-02-07'),

(90, 'Amit', 'Kumar', '8889990000', '1991-10-03'),

(91, 'Kavita', 'Rajput', '5556667777', '1989-03-18'),

(92, 'Sanjay', 'Sharma', '7770001111', '1985-05-12'),

(93, 'Mala', 'Mishra', '4445558888', '1994-06-26'),

(94, 'Rajesh', 'Verma', '1115556666', '1990-09-01'),

(95, 'Vikas', 'Gupta', '3337778888', '1977-12-30'),

(96, 'Meera', 'Kumar', '6662223333', '1993-04-09'),

(97, 'Rahul', 'Yadav', '5554445555', '1995-07-15'),

(98, 'Neeta', 'Sharma', '7771113333', '1987-11-20'),

(99, 'Suresh', 'Mishra', '8881110000', '1989-01-28'),

(100, 'Priya', 'Kumar', '7896543210', '1980-05-20');

INSERT INTO Status (user\_id, status\_type, privacy)

VALUES

( 1, 'photos', 'public'),

( 2, 'videos', 'contacts only'),

( 3, 'text', 'close friends'),

( 4, 'photos', 'public'),

( 5, 'videos', 'contacts only'),

(6, 'text', 'close friends'),

( 7, 'photos', 'public'),

( 8, 'videos', 'contacts only'),

( 9, 'text', 'close friends'),

(10, 'photos', 'public'),

(11, 'videos', 'contacts only'),

( 12, 'text', 'close friends'),

(3, 'photos', 'public'),

(4, 'videos', 'contacts only'),

(15, 'text', 'close friends'),

(1, 'photos', 'public'),

(17, 'videos', 'contacts only'),

(18, 'text', 'close friends'),

(19, 'photos', 'public'),

(30, 'videos', 'contacts only'),

(21, 'text', 'close friends'),

(22, 'photos', 'public'),

(22, 'videos', 'contacts only'),

(2, 'text', 'close friends'),

(25, 'photos', 'public'),

(26, 'videos', 'contacts only'),

(27, 'text', 'close friends'),

(27, 'photos', 'public'),

(29, 'videos', 'contacts only'),

(30, 'text', 'close friends'),

(31, 'photos', 'public'),

(32, 'videos', 'contacts only'),

(3, 'text', 'close friends'),

(34, 'photos', 'public'),

(5, 'videos', 'contacts only'),

(36, 'text', 'close friends'),

(37, 'photos', 'public'),

(38, 'videos', 'contacts only'),

(9, 'text', 'close friends'),

(40, 'photos', 'public'),

(41, 'videos', 'contacts only'),

(42, 'text', 'close friends'),

(43, 'photos', 'public'),

(44, 'videos', 'contacts only'),

(5, 'text', 'close friends'),

(46, 'photos', 'public'),

(47, 'videos', 'contacts only'),

(48, 'text', 'close friends'),

(49, 'photos', 'public'),

(50, 'videos', 'contacts only'),

(1, 'text', 'close friends'),

(22, 'photos', 'public'),

(33, 'videos', 'contacts only'),

(44, 'text', 'close friends'),

(5, 'photos', 'public'),

(46, 'videos', 'contacts only'),

(17, 'text', 'close friends'),

( 28, 'photos', 'public'),

(39, 'videos', 'contacts only'),

(40, 'text', 'close friends'),

(41, 'photos', 'public'),

(22, 'videos', 'contacts only'),

(33, 'text', 'close friends'),

(44, 'photos', 'public'),

(25, 'videos', 'contacts only'),

(16, 'text', 'close friends'),

(37, 'photos', 'public'),

(32, 'videos', 'contacts only'),

(32, 'text', 'close friends'),

(32, 'photos', 'public'),

(11, 'videos', 'contacts only'),

(22, 'text', 'close friends'),

(33, 'photos', 'public'),

(44, 'videos', 'contacts only'),

(45, 'text', 'close friends'),

(36, 'photos', 'public'),

(27, 'videos', 'contacts only'),

(8, 'text', 'close friends'),

(9, 'photos', 'public'),

(40, 'videos', 'contacts only'),

(31, 'text', 'close friends'),

(12, 'photos', 'public'),

(13, 'videos', 'contacts only'),

(14, 'text', 'close friends'),

(5, 'photos', 'public'),

(26, 'videos', 'contacts only'),

(37, 'text', 'close friends'),

(28, 'photos', 'public'),

(29, 'videos', 'contacts only'),

(20, 'text', 'close friends'),

(21, 'photos', 'public'),

(32, 'videos', 'contacts only'),

(43, 'text', 'close friends'),

(24, 'photos', 'public'),

(35, 'videos', 'contacts only'),

(46, 'text', 'close friends'),

(27, 'photos', 'public'),

(48, 'videos', 'contacts only'),

(29, 'text', 'close friends'),

(20, 'photos', 'public'),

(1, 'videos', 'contacts only');

INSERT INTO Calls (call\_id, user\_id, receiver\_id, call\_type, duration)

VALUES

(101, 1, 30, 'voice', '00:05:00'),

(102, 2, 32, 'video', '00:07:00'),

(103, 3, 33, 'voice', '00:00:00'),

(104, 4, 34, 'voice', '00:03:00'),

(105, 5, 35, 'video', '00:10:00'),

(106, 6, 36, 'voice', '00:04:00'),

(107, 7, 37, 'voice', '00:00:00'),

(108, 8, 38, 'video', '00:06:00'),

(109, 9, 39, 'voice', '00:08:00'),

(110, 10, 30, 'video', '00:09:00'),

(111, 1, 31, 'voice', '00:12:00'),

(112, 12, 32, 'voice', '00:00:00'),

(113, 1, 31, 'video', '00:05:00'),

(114, 14, 34, 'voice', '00:06:00'),

(115, 15, 31, 'video', '00:08:00'),

(116, 16, 31, 'voice', '00:00:00'),

(117, 7, 31, 'voice', '00:04:00'),

(118, 18, 38, 'video', '00:10:00'),

(119, 19, 39, 'voice', '00:03:00'),

(120, 20, 30, 'video', '00:07:00'),

(121, 21, 31, 'voice', '00:05:00'),

(122, 22, 32, 'voice', '00:00:00'),

(123, 23, 33, 'video', '00:04:00'),

(124, 4, 3, 'voice', '00:06:00'),

(125, 25, 5, 'video', '00:08:00'),

(126, 26, 6, 'voice', '00:09:00'),

(127, 7, 3, 'video', '00:12:00'),

(128, 28 , 3, 'voice', '00:00:00'),

(129, 29, 32, 'voice', '00:05:00'),

(130, 30, 33, 'video', '00:07:00'),

(131, 1, 33, 'voice', '00:04:00'),

(132, 32, 3, 'video', '00:10:00'),

(133, 33, 3, 'voice', '00:00:00'),

(134, 34, 3, 'voice', '00:06:00'),

(135, 5, 33, 'video', '00:08:00'),

(136, 36, 36, 'voice', '00:09:00'),

(137, 37, 7, 'video', '00:12:00'),

(138, 38, 8, 'voice', '00:00:00'),

(139, 39, 9, 'voice', '00:05:00'),

(140, 20, 4, 'video', '00:07:00'),

(141, 41, 3, 'voice', '00:05:00'),

(142, 42, 32, 'voice', '00:00:00'),

(143, 43, 33, 'video', '00:04:00'),

(144, 44, 34, 'voice', '00:06:00'),

(145, 45, 35, 'video', '00:08:00'),

(146, 46, 36, 'voice', '00:09:00'),

(147, 47, 37, 'video', '00:12:00'),

(148, 48, 38, 'voice', '00:00:00'),

(149, 49, 39, 'voice', '00:05:00'),

(150, 20, 30, 'video', '00:07:00'),

(151, 5, 34, 'voice', '00:05:00'),

(152, 22, 32, 'voice', '00:00:00'),

(153, 33, 33, 'video', '00:04:00'),

(154, 44, 34, 'voice', '00:06:00'),

(155, 25, 35, 'video', '00:08:00'),

(156, 46, 36, 'voice', '00:09:00'),

(157, 27, 37, 'video', '00:12:00'),

(158, 38, 3, 'voice', '00:00:00'),

(159, 49, 39, 'voice', '00:05:00'),

(160, 20, 30, 'video', '00:07:00'),

(161, 41, 31, 'voice', '00:04:00'),

(162, 12, 32, 'video', '00:10:00'),

(163, 33, 3, 'voice', '00:00:00'),

(164, 14, 34, 'voice', '00:06:00'),

(165, 25, 35, 'video', '00:08:00'),

(166, 36, 3, 'voice', '00:09:00'),

(167, 37, 3, 'video', '00:12:00'),

(168, 28, 38, 'voice', '00:00:00'),

(169, 49, 39, 'voice', '00:05:00'),

(170, 20, 30, 'video', '00:07:00'),

(171, 11, 31, 'voice', '00:05:00'),

(172, 22, 32, 'voice', '00:00:00'),

(173, 3, 33, 'video', '00:04:00'),

(174, 4, 24, 'voice', '00:06:00'),

(175, 45, 25, 'video', '00:08:00'),

(176, 36, 6, 'voice', '00:09:00'),

(177, 27, 37, 'video', '00:12:00'),

(178, 28, 38, 'voice', '00:00:00'),

(179, 49, 39, 'voice', '00:05:00'),

(180, 40, 30, 'video', '00:07:00'),

(181, 41, 31, 'voice', '00:04:00'),

(182, 42, 32, 'video', '00:10:00'),

(183, 43, 33, 'voice', '00:00:00'),

(184, 44, 34, 'voice', '00:06:00'),

(185, 45, 5, 'video', '00:08:00'),

(186, 46, 6, 'voice', '00:09:00'),

(187, 47, 37, 'video', '00:12:00'),

(188, 38, 3, 'voice', '00:00:00'),

(189, 29, 39, 'voice', '00:05:00'),

(190, 20, 30, 'video', '00:07:00'),

(191, 31, 3, 'voice', '00:05:00'),

(192, 32, 42, 'voice', '00:00:00'),

(193, 33, 23, 'video', '00:04:00'),

(194, 34, 44, 'voice', '00:06:00'),

(195, 35, 44, 'video', '00:08:00'),

(196, 46, 22, 'voice', '00:09:00'),

(197, 47, 27, 'video', '00:12:00'),

(198, 18, 38, 'voice', '00:00:00'),

(199, 9, 32, 'voice', '00:05:00'),

(200, 20, 1, 'video', '00:07:00');

INSERT INTO Notification (notification\_id, message\_id, sender\_id, recipient\_id, ncontent, notification\_date, notification\_time)

VALUES

(101, 1, 1, 3, 'text', '2023-11-05', '08:00:00'),

(102, 2, 2, 32, 'voice', '2023-11-05', '08:15:00'),

(103, 3, 7, 33, 'photos', '2023-11-05', '08:30:00'),

(104, 44, 14, 44, 'videos', '2023-11-05', '08:45:00'),

(105, 54, 15, 55, 'emojis', '2023-11-05', '09:00:00'),

(106, 62, 6, 36, 'stickers', '2023-11-05', '09:15:00'),

(107, 17, 7, 35, 'GIF', '2023-11-05', '09:30:00'),

(108, 18, 8, 35, 'text', '2023-11-05', '09:45:00'),

(109, 9, 29, 34, 'voice', '2023-11-05', '10:00:00'),

(110, 1, 20, 30, 'photos', '2023-11-05', '10:15:00'),

(111, 1, 21, 31, 'videos', '2023-11-05', '10:30:00'),

(112, 1, 22, 32, 'emojis', '2023-11-05', '10:45:00'),

(113, 13, 3, 33, 'stickers', '2023-11-05', '11:00:00'),

(114, 14, 4, 34, 'GIF', '2023-11-05', '11:15:00'),

(115, 15, 15, 1, 'text', '2023-11-05', '11:30:00'),

(116, 16, 16, 31, 'voice', '2023-11-05', '11:45:00'),

(117, 17, 17, 31, 'photos', '2023-11-05', '12:00:00'),

(118, 8, 28, 31, 'videos', '2023-11-05', '12:15:00'),

(119, 9, 29, 31, 'emojis', '2023-11-05', '12:30:00'),

(120, 20, 20, 2, 'stickers', '2023-11-05', '12:45:00'),

(121, 21, 21, 30, 'GIF', '2023-11-05', '13:00:00'),

(122, 22, 12, 22, 'text', '2023-11-05', '13:15:00'),

(123, 23, 13, 27, 'voice', '2023-11-05', '13:30:00'),

(124, 24, 84, 14, 'photos', '2023-11-05', '13:45:00'),

(125, 25, 75, 25, 'videos', '2023-11-05', '14:00:00'),

(126, 26, 76, 26, 'emojis', '2023-11-05', '14:15:00'),

(127, 24, 27, 2, 'stickers', '2023-11-05', '14:30:00'),

(128, 22, 88, 38, 'GIF', '2023-11-05', '14:45:00'),

(129, 29, 29, 39, 'text', '2023-11-05', '15:00:00'),

(130, 35, 30, 33, 'voice', '2023-11-05', '15:15:00'),

(131, 31, 31, 34, 'photos', '2023-11-05', '15:30:00'),

(132, 32, 32, 2, 'videos', '2023-11-05', '15:45:00'),

(133, 43, 33, 3, 'emojis', '2023-11-05', '16:00:00'),

(134, 34, 14, 34, 'stickers', '2023-11-05', '16:15:00'),

(135, 35, 95, 35, 'GIF', '2023-11-05', '16:30:00'),

(136, 76, 96, 76, 'text', '2023-11-05', '16:45:00'),

(137, 67, 37, 7, 'voice', '2023-11-05', '17:00:00'),

(138, 38, 38, 8, 'photos', '2023-11-05', '17:15:00'),

(139, 69, 39, 9, 'videos', '2023-11-05', '17:30:00'),

(140, 40, 47, 40, 'emojis', '2023-11-05', '17:45:00'),

(141, 41, 41, 1, 'stickers', '2023-11-05', '18:00:00'),

(142, 42, 42, 2, 'GIF', '2023-11-05', '18:15:00'),

(143, 43, 47, 43, 'text', '2023-11-05', '18:30:00'),

(144, 64, 47, 44, 'voice', '2023-11-05', '18:45:00'),

(145, 85, 47, 45, 'photos', '2023-11-05', '19:00:00'),

(146, 26, 46, 6, 'videos', '2023-11-05', '19:15:00'),

(147, 47, 47, 7, 'emojis', '2023-11-05', '19:30:00'),

(148, 78, 8, 38, 'stickers', '2023-11-05', '19:45:00'),

(149, 89, 9, 39, 'GIF', '2023-11-05', '20:00:00'),

(150, 60, 40, 10, 'text', '2023-11-05', '20:15:00'),

(151, 51, 41, 51, 'voice', '2023-11-05', '20:30:00'),

(152, 52, 42, 12, 'photos', '2023-11-05', '20:45:00'),

(153, 33, 73, 53, 'videos', '2023-11-05', '21:00:00'),

(154, 54, 57, 54, 'emojis', '2023-11-05', '21:15:00'),

(155, 55, 55, 15, 'stickers', '2023-11-05', '21:30:00'),

(156, 56, 57, 56, 'GIF', '2023-11-05', '21:45:00'),

(157, 47, 57, 17, 'text', '2023-11-05', '22:00:00'),

(158, 78, 78, 58, 'voice', '2023-11-05', '22:15:00'),

(159, 89, 59, 9, 'photos', '2023-11-05', '22:30:00'),

(160, 40, 90, 60, 'videos', '2023-11-05', '22:45:00'),

(161, 61, 61, 1, 'emojis', '2023-11-05', '23:00:00'),

(162, 42, 92, 60, 'stickers', '2023-11-05', '23:15:00'),

(163, 43, 93, 43, 'GIF', '2023-11-05', '23:30:00'),

(164, 74, 64, 4, 'text', '2023-11-05', '23:45:00'),

(165, 25, 75, 65, 'voice', '2023-11-06', '00:00:00'),

(166, 66, 66, 46, 'photos', '2023-11-06', '00:15:00'),

(167, 67, 67, 7, 'videos', '2023-11-06', '00:30:00'),

(168, 68, 69, 68, 'emojis', '2023-11-06', '00:45:00'),

(169, 29, 69, 49, 'stickers', '2023-11-06', '01:00:00'),

(170, 72, 10, 70, 'GIF', '2023-11-06', '01:15:00'),

(171, 51, 11, 41, 'text', '2023-11-06', '01:30:00'),

(172, 52, 72, 42, 'voice', '2023-11-06', '01:45:00'),

(173, 53, 13, 73, 'photos', '2023-11-06', '02:00:00'),

(174, 44, 74, 78, 'videos', '2023-11-06', '02:15:00'),

(175, 45, 75, 5, 'emojis', '2023-11-06', '02:30:00'),

(176, 16, 76, 6, 'stickers', '2023-11-06', '02:45:00'),

(177, 17, 77, 76, 'GIF', '2023-11-06', '03:00:00'),

(178, 8, 29, 38, 'text', '2023-11-06', '03:15:00'),

(179, 9, 29, 39, 'voice', '2023-11-06', '03:30:00'),

(180, 8, 20, 30, 'photos', '2023-11-06', '03:45:00'),

(181, 1, 21, 31, 'videos', '2023-11-06', '04:00:00'),

(182, 82, 82, 2, 'emojis', '2023-11-06', '04:15:00'),

(183, 3, 23, 32, 'stickers', '2023-11-06', '04:30:00'),

(184, 4, 24, 34, 'GIF', '2023-11-06', '04:45:00'),

(185, 25, 85, 45, 'text', '2023-11-06', '05:00:00'),

(186, 83, 76, 46, 'voice', '2023-11-06', '05:15:00'),

(187, 83, 27, 7, 'photos', '2023-11-06', '05:30:00'),

(188, 88, 88, 8, 'videos', '2023-11-06', '05:45:00');

INSERT INTO groupchat (groupchat\_id, user\_id, group\_name, member\_size, group\_created) VALUES

(101, 1, 'Family Gathering', 8, '2023-01-01'),

(102, 2, 'Workplace Chats', 15, '2023-01-02'),

(103, 3, 'College Friends', 6, '2023-01-03'),

(104, 4, 'Sports Enthusiasts', 10, '2023-01-04'),

(105, 5, 'Tech Innovators', 12, '2023-01-05'),

(106, 6, 'Travel Enthusiasts', 9, '2023-01-06'),

(107, 7, 'Food Lovers', 7, '2023-01-07'),

(108, 8, 'Gaming Community', 11, '2023-01-08'),

(109, 9, 'Bookworms', 5, '2023-01-09'),

(110, 10, 'Hiking Adventures', 6, '2023-01-10'),

(111, 11, 'Music Maniacs', 8, '2023-01-11'),

(112, 12, 'Art Aficionados', 10, '2023-01-12'),

(113, 13, 'Fitness Fanatics', 7, '2023-01-13'),

(114, 14, 'Cinema Buffs', 9, '2023-01-14'),

(115, 15, 'Photography Enthusiasts', 14, '2023-01-15'),

(116, 16, 'Pet Lovers', 6, '2023-01-16'),

(117, 17, 'Language Exchange', 8, '2023-01-17'),

(118, 18, 'Science Geeks', 12, '2023-01-18'),

(119, 19, 'Fashionistas', 10, '2023-01-19'),

(120, 20, 'DIY Enthusiasts', 7, '2023-01-20'),

(121, 21, 'Finance Professionals', 9, '2023-01-21'),

(122, 22, 'Nature Explorers', 11, '2023-01-22'),

(123, 23, 'Anime & Manga Fans', 6, '2023-01-23'),

(124, 24, 'History Buffs', 8, '2023-01-24'),

(125, 25, 'Cooking Connoisseurs', 13, '2023-01-25'),

(126, 26, 'Entrepreneurs', 9, '2023-01-26'),

(127, 27, 'Yoga and Meditation', 10, '2023-01-27'),

(128, 28, 'Environmentalists', 7, '2023-01-28'),

(129, 29, 'Film Production Crew', 14, '2023-01-29'),

(130, 30, 'Dance Enthusiasts', 8, '2023-01-30'),

(131, 31, 'Science Fiction Fans', 7, '2023-01-31'),

(132, 32, 'Travel Bloggers', 9, '2023-02-01'),

(133, 33, 'Fashion Designers', 11, '2023-02-02'),

(134, 34, 'Tech Gurus', 12, '2023-02-03'),

(135, 35, 'Book Club', 6, '2023-02-04'),

(136, 36, 'Board Gamers', 8, '2023-02-05'),

(137, 37, 'Artistic Creations', 10, '2023-02-06'),

(138, 38, 'Fitness Freaks', 14, '2023-02-07'),

(139, 39, 'Movie Buffs', 9, '2023-02-08'),

(140, 40, 'Music Lovers', 7, '2023-02-09'),

(141, 41, 'Writers Guild', 11, '2023-02-10'),

(142, 42, 'Gardening Enthusiasts', 8, '2023-02-11'),

(143, 43, 'Coding Enthusiasts', 12, '2023-02-12'),

(144, 44, 'Vintage Collectors', 10, '2023-02-13'),

(145, 45, 'Home Decor Enthusiasts', 9, '2023-02-14'),

(146, 46, 'Cycling Club', 6, '2023-02-15'),

(147, 47, 'Astrology Enthusiasts', 7, '2023-02-16'),

(148, 48, 'Beer Connoisseurs', 8, '2023-02-17'),

(149, 49, 'Parenting Tips', 13, '2023-02-18'),

(150, 50, 'Astronomy Club', 14, '2023-02-19'),

(151, 5, 'Fitness Fanatics', 7, '2023-02-20'),

(152, 5, 'Adventure Seekers', 10, '2023-02-21'),

(153, 3, 'Movie Buffs', 9, '2023-02-22'),

(154, 4, 'Tech Enthusiasts', 11, '2023-02-23'),

(155, 5, 'Book Club', 6, '2023-02-24'),

(156, 6, 'Yoga and Meditation', 8, '2023-02-25'),

(157, 7, 'Music Maniacs', 14, '2023-02-26'),

(158, 8, 'Art Enthusiasts', 7, '2023-02-27'),

(159, 9, 'Fashionistas', 12, '2023-02-28'),

(160, 20, 'Cooking Connoisseurs', 9, '2023-03-01'),

(161, 21, 'Hiking Adventures', 10, '2023-03-02'),

(162, 22, 'Science Geeks', 8, '2023-03-03'),

(163, 33, 'Travel Bloggers', 13, '2023-03-04'),

(164, 44, 'Coding Enthusiasts', 11, '2023-03-05'),

(165, 45, 'History Buffs', 6, '2023-03-06'),

(166, 26, 'Language Exchange', 8, '2023-03-07'),

(167, 7, 'Sports Enthusiasts', 10, '2023-03-08'),

(168, 38, 'Fitness Freaks', 7, '2023-03-09'),

(169, 49, 'Travel Enthusiasts', 9, '2023-03-10'),

(170, 40, 'Tech Innovators', 12, '2023-03-11'),

(171, 21, 'Board Gamers', 6, '2023-03-12'),

(172, 32, 'Dance Enthusiasts', 8, '2023-03-13'),

(173, 43, 'Fashion Designers', 12, '2023-03-14'),

(174, 24, 'Nature Explorers', 10, '2023-03-15'),

(175, 35, 'Astrology Enthusiasts', 9, '2023-03-16'),

(176, 26, 'Home Decor Enthusiasts', 7, '2023-03-17'),

(177, 47, 'Cinema Buffs', 14, '2023-03-18'),

(178, 28, 'Artistic Creations', 8, '2023-03-19'),

(179, 39, 'Gaming Community', 11, '2023-03-20'),

(180, 40, 'Science Fiction Fans', 10, '2023-03-21'),

(181, 21, 'Entrepreneurs', 9, '2023-03-22'),

(182, 22, 'Vintage Collectors', 7, '2023-03-23'),

(183, 23, 'Gardening Enthusiasts', 12, '2023-03-24'),

(184, 34, 'Pet Lovers', 6, '2023-03-25'),

(185, 45, 'Astronomy Club', 8, '2023-03-26'),

(186, 26, 'Cooking Enthusiasts', 10, '2023-03-27'),

(187, 17, 'Music Lovers', 7, '2023-03-28'),

(188, 48, 'Writers Guild', 9, '2023-03-29'),

(189, 9, 'Beer Connoisseurs', 11, '2023-03-30'),

(190, 20, 'Environmentalists', 13, '2023-03-31'),

(191, 31, 'Vintage Car Collectors', 8, '2023-04-01'),

(192, 42, 'Parenting Tips', 10, '2023-04-02'),

(193, 23, 'Movie Buffs', 6, '2023-04-03'),

(194, 24, 'Fashionistas', 7, '2023-04-04'),

(195, 35, 'Dance Enthusiasts', 9, '2023-04-05'),

(196, 46, 'Tech Gurus', 11, '2023-04-06'),

(197, 27, 'Astrology Enthusiasts', 12, '2023-04-07'),

(198, 28, 'Cycling Club', 7, '2023-04-08'),

(199, 29, 'Cooking Connoisseurs', 8, '2023-04-09'),

(200, 20, 'Entrepreneurs', 10,'2023-04-10');

INSERT INTO Chat (chat\_id, user\_id, participant\_name, updation\_date) VALUES

(1, 1, 'Alice', '2023-01-01'),

(2, 21, 'Bob', '2023-01-02'),

(3, 31, 'Charlie', '2023-01-03'),

(4, 4, 'David', '2023-01-04'),

(5, 5, 'Eva', '2023-01-05'),

(6, 6, 'Frank', '2023-01-06'),

(7, 7, 'Grace', '2023-01-07'),

(8, 8, 'Hannah', '2023-01-08'),

(9, 91, 'Ian', '2023-01-09'),

(10, 10, 'Jack', '2023-01-10'),

(11, 11, 'Katie', '2023-01-11'),

(12, 12, 'Liam', '2023-01-12'),

(13, 3, 'Mia', '2023-01-13'),

(14, 1, 'Noah', '2023-01-14'),

(15, 5, 'Olivia', '2023-01-15'),

(16, 1, 'Peter', '2023-01-16'),

(17, 17, 'Quinn', '2023-01-17'),

(18, 18, 'Riley', '2023-01-18'),

(19, 9, 'Sam', '2023-01-19'),

(20, 20, 'Tara', '2023-01-20'),

(21, 1, 'Uma', '2023-01-21'),

(22, 12, 'Victor', '2023-01-22'),

(23, 23, 'Wendy', '2023-01-23'),

(24, 24, 'Xander', '2023-01-24'),

(25, 2, 'Yara', '2023-01-25'),

(26, 6, 'Zane', '2023-01-26'),

(27, 7, 'Abigail', '2023-01-27'),

(28, 28, 'Benjamin', '2023-01-28'),

(29, 29, 'Charlotte', '2023-01-29'),

(30, 50, 'Daniel', '2023-01-30'),

(31, 31, 'Emily', '2023-01-31'),

(32, 32, 'Finn', '2023-02-01'),

(33, 33, 'Grace', '2023-02-02'),

(34, 34, 'Hannah', '2023-02-03'),

(35, 5, 'Isaac', '2023-02-04'),

(36, 36, 'Jane', '2023-02-05'),

(37, 7, 'Kai', '2023-02-06'),

(38, 38, 'Lily', '2023-02-07'),

(39, 39, 'Mason', '2023-02-08'),

(40, 30, 'Nora', '2023-02-09'),

(41, 41, 'Oliver', '2023-02-10'),

(42, 42, 'Penelope', '2023-02-11'),

(43, 33, 'Quinn', '2023-02-12'),

(44, 64, 'Riley', '2023-02-13'),

(45, 45, 'Sophia', '2023-02-14'),

(46, 66, 'Theo', '2023-02-15'),

(47, 47, 'Uma', '2023-02-16'),

(48, 48, 'Vincent', '2023-02-17'),

(49, 9, 'Willow', '2023-02-18'),

(50, 50, 'Xander', '2023-02-19'),

(51, 61, 'Yara', '2023-02-20'),

(52, 62, 'Zane', '2023-02-21'),

(53, 53, 'Ava', '2023-02-22'),

(54, 54, 'Benjamin', '2023-02-23'),

(55, 75, 'Caleb', '2023-02-24'),

(56, 56, 'Daisy', '2023-02-25'),

(57, 57, 'Ethan', '2023-02-26'),

(58, 58, 'Fiona', '2023-02-27'),

(59, 9, 'George', '2023-02-28'),

(60, 6, 'Hazel', '2023-03-01'),

(61, 61, 'Isaiah', '2023-03-02'),

(62, 62, 'Julia', '2023-03-03'),

(63, 73, 'Kaden', '2023-03-04'),

(64, 64, 'Lila', '2023-03-05'),

(65, 65, 'Maddox', '2023-03-06'),

(66, 66, 'Natalie', '2023-03-07'),

(67, 77, 'Oscar', '2023-03-08'),

(68, 68, 'Peyton', '2023-03-09'),

(69, 89, 'Quincy', '2023-03-10'),

(70, 70, 'Riley', '2023-03-11'),

(71, 81, 'Sofia', '2023-03-12'),

(72, 82, 'Theodore', '2023-03-13'),

(73, 73, 'Ursula', '2023-03-14'),

(74, 74, 'Victor', '2023-03-15'),

(75, 75, 'Willow', '2023-03-16'),

(76, 76, 'Xavier', '2023-03-17'),

(77, 77, 'Yasmine', '2023-03-18'),

(78, 78, 'Zachary', '2023-03-19'),

(79, 79, 'Abby', '2023-03-20'),

(80, 80, 'Brian', '2023-03-21'),

(81, 81, 'Caroline', '2023-03-22'),

(82, 82, 'David', '2023-03-23'),

(83, 83, 'Ella', '2023-03-24'),

(84, 84, 'Felix', '2023-03-25'),

(85, 85, 'Giselle', '2023-03-26'),

(86, 86, 'Henry', '2023-03-27'),

(87, 87, 'Ivy', '2023-03-28'),

(88, 88, 'Jacob', '2023-03-29'),

(89, 89, 'Katherine', '2023-03-30'),

(90, 90, 'Landon', '2023-03-31'),

(91, 91, 'Molly', '2023-04-01'),

(92, 92, 'Nathan', '2023-04-02'),

(93, 93, 'Oliver', '2023-04-03'),

(94, 94, 'Piper', '2023-04-04'),

(95, 75, 'Quincy', '2023-04-05'),

(96, 96, 'Riley', '2023-04-06'),

(97, 77, 'Sophia', '2023-04-07'),

(98, 48, 'Theo', '2023-04-08'),

(99, 49, 'Uma', '2023-04-09'),

(100, 100, 'Victor','2023-04-10');

INSERT INTO Messages (message\_id, sender\_id, recipient\_id, attachment\_id, m\_content, message\_date, message\_time) VALUES

(1, 23, 2, 201, 'Text', '2023-01-01', '08:30'),

(2, 24, 3, 202, 'Voice', '2023-01-02', '09:15'),

(3, 25, 34, 203, 'Emojis', '2023-01-03', '10:00'),

(4, 26, 35, 204, 'Sticker', '2023-01-04', '11:45'),

(5, 27, 36, 205, 'Gif', '2023-01-05', '12:30'),

(6, 28, 37, 206, 'Text', '2023-01-06', '13:15'),

(7, 29, 38, 207, 'Voice', '2023-01-07', '14:00'),

(8, 30, 39, 248, 'Emojis', '2023-01-08', '14:45'),

(9, 31, 3, 249, 'Sticker', '2023-01-09', '15:30'),

(10, 32, 31, 240, 'Gif', '2023-01-10', '16:15'),

(11, 33, 32, 211, 'Text', '2023-01-11', '17:00'),

(12, 34, 343, 212, 'Voice', '2023-01-12', '17:45'),

(13, 35, 3, 213, 'Emojis', '2023-01-13', '18:30'),

(14, 36, 35, 214, 'Sticker', '2023-01-14', '19:15'),

(15, 47, 36, 214, 'Gif', '2023-01-15', '20:00'),

(16, 48, 37, 214, 'Text', '2023-01-16', '20:45'),

(17, 39, 38, 217, 'Voice', '2023-01-17', '21:30'),

(18, 40, 349, 218, 'Emojis', '2023-01-18', '22:15'),

(19, 41, 350, 219, 'Sticker', '2023-01-19', '23:00'),

(20, 42, 31, 220, 'Gif', '2023-01-20', '23:45'),

(21, 40, 52, 222, 'Text', '2023-01-21', '00:30'),

(22, 44, 53, 222, 'Voice', '2023-01-22', '01:15'),

(23, 45, 34, 223, 'Emojis', '2023-01-23', '02:00'),

(24, 146, 35, 224, 'Sticker', '2023-01-24', '02:45'),

(25, 47, 35, 225, 'Gif', '2023-01-25', '03:30'),

(26, 48, 3, 226, 'Text', '2023-01-26', '04:15'),

(27, 40, 5, 222, 'Voice', '2023-01-27', '05:00'),

(28, 50, 59, 222, 'Emojis', '2023-01-28', '05:45'),

(29, 51, 60, 229, 'Sticker', '2023-01-29', '06:30'),

(30, 42, 61, 260, 'Gif', '2023-01-30', '07:15'),

(31, 53, 62, 241, 'Text', '2023-01-31', '08:00'),

(32, 54, 63, 239, 'Voice', '2023-02-01', '08:45'),

(33, 35, 64, 234, 'Emojis', '2023-02-02', '09:30'),

(34, 56, 65, 234, 'Sticker', '2023-02-03', '10:15'),

(35, 57, 45, 234, 'Gif', '2023-02-04', '11:00'),

(36, 58, 45, 246, 'Text', '2023-02-05', '11:45'),

(37, 59, 48, 247, 'Voice', '2023-02-06', '12:30'),

(38, 60, 59, 248, 'Emojis', '2023-02-07', '13:15'),

(39, 61, 50, 239, 'Sticker', '2023-02-08', '14:00'),

(40, 62, 71, 240, 'Gif', '2023-02-09', '14:45'),

(41, 63, 72, 261, 'Text', '2023-02-10', '15:30'),

(42, 44, 73, 242, 'Voice', '2023-02-11', '16:15'),

(43, 65, 74, 263, 'Emojis', '2023-02-12', '17:00'),

(44, 66, 75, 244, 'Sticker', '2023-02-13', '17:45'),

(45, 37, 76, 245, 'Gif', '2023-02-14', '18:30'),

(46, 23, 37, 246, 'Text', '2023-02-15', '19:15'),

(47, 24, 78, 247, 'Voice', '2023-02-16', '20:00'),

(48, 25, 79, 248, 'Emojis', '2023-02-17', '20:45'),

(49, 26, 80, 249, 'Sticker', '2023-02-18', '21:30'),

(50, 27, 81, 260, 'Gif', '2023-02-19', '22:15'),

(51, 28, 82, 251, 'Text', '2023-02-20', '23:00'),

(52, 29, 83, 252, 'Voice', '2023-02-21', '23:45'),

(53, 30, 84, 253, 'Emojis', '2023-02-22', '00:30'),

(54, 31, 85, 254, 'Sticker', '2023-02-23', '01:15'),

(55, 32, 86, 275, 'Gif', '2023-02-24', '02:00'),

(56, 33, 7, 256, 'Text', '2023-02-25', '02:45'),

(57, 34, 8, 277, 'Voice', '2023-02-26', '03:30'),

(58, 35, 89, 278, 'Emojis', '2023-02-27', '04:15'),

(59, 36, 90, 259, 'Sticker', '2023-02-28', '05:00'),

(60, 37, 31, 250, 'Gif', '2023-03-01', '05:45'),

(61, 38, 32, 261, 'Text', '2023-03-02', '06:30'),

(62, 39, 33, 262, 'Voice', '2023-03-03', '07:15'),

(63, 40, 39, 223, 'Emojis', '2023-03-04', '08:00'),

(64, 14, 395, 264, 'Sticker', '2023-03-05', '08:45'),

(65, 14, 6, 225, 'Gif', '2023-03-06', '09:30'),

(66, 13, 37, 266, 'Text', '2023-03-07', '10:15'),

(67, 14, 38, 247, 'Voice', '2023-03-08', '11:00'),

(68, 15, 39, 248, 'Emojis', '2023-03-09', '11:45'),

(69, 14, 30, 239, 'Sticker', '2023-03-10', '12:30'),

(70, 14, 31, 270, 'Gif', '2023-03-11', '13:15'),

(71, 14, 32, 271, 'Text', '2023-03-12', '14:00'),

(72, 19, 33, 272, 'Voice', '2023-03-13', '14:45'),

(73, 15, 34, 273, 'Emojis', '2023-03-14', '15:30'),

(74, 15, 33, 274, 'Sticker', '2023-03-15', '16:15'),

(75, 15, 36, 275, 'Gif', '2023-03-16', '17:00'),

(76, 15, 5, 276, 'Text', '2023-03-17', '17:45'),

(77, 4, 7, 277, 'Voice', '2023-03-18', '18:30'),

(78, 5, 9, 278, 'Emojis', '2023-03-19', '19:15');

INSERT INTO Attachment (attachment\_id, message\_id, file\_name, size\_mb, type)

VALUES

(200, 1, 'document1.txt', 1.2, 'documents'),

(201, 2, 'report.pdf', 0.5, 'documents'),

(202, 3, 'vacation\_photo.jpg', 3.8, 'photos'),

(203, 34, 'proposal.docx', 2.7, 'documents'),

(204, 25, 'family\_pic.png', 1.1, 'photos'),

(205, 16, 'notes.txt', 0.9, 'documents'),

(206, 75, 'budget.xlsx', 4.3, 'documents'),

(207, 86, 'landscape.jpg', 2.5, 'photos'),

(208, 96, 'presentation.zip', 5.2, 'documents'),

(209, 10, 'invoice.html', 1.8, 'link'),

(210, 1, 'memo.txt', 0.6, 'documents'),

(211, 1, 'project.docx', 3.2, 'documents'),

(212, 3, 'sunset.jpg', 2.1, 'photos'),

(213, 4, 'agenda.html', 0.7, 'link'),

(214, 15, 'meeting\_notes.txt', 1.9, 'documents'),

(215, 6, 'quarterly\_report.xlsx', 4.7, 'documents'),

(216, 7, 'beach\_day.mp4', 3.4, 'videos'),

(217, 8, 'instructions.html', 2.3, 'link'),

(218, 19, 'archive.zip', 5.6, 'documents'),

(219, 20, 'manual.pdf', 1.5, 'documents'),

(220, 21, 'user\_guide.txt', 0.8, 'documents'),

(221, 82, 'project\_plan.docx', 2.1, 'documents'),

(222, 83, 'sightseeing.mp4', 3.1, 'video'),

(223, 84, 'announcement.pdf', 1.3, 'documents'),

(224, 85, 'important\_notes.txt', 0.7, 'documents'),

(225, 26, 'sales\_data.xlsx', 4.0, 'documents'),

(226, 77, 'product\_shot.html', 2.9, 'link'),

(227, 78, 'research.html', 2.0, 'link'),

(228, 89, 'archive.zip', 6.1, 'documents'),

(229, 38, 'agreement.html', 2.8, 'link'),

(230, 38, 'document1.txt', 1.0, 'documents'),

(231, 36, 'report.pdf', 1.2, 'documents'),

(232, 53, 'photo.jpg', 3.5, 'photos'),

(233, 54, 'spreadsheet.xlsx', 4.8, 'documents'),

(234, 35, 'nature.jpg', 3.0, 'photos'),

(235, 36, 'notes.txt', 1.1, 'documents'),

(236, 67, 'report.docx', 2.3, 'documents'),

(237, 68, 'manual.pdf', 1.0, 'documents'),

(238, 39, 'guide.txt', 0.5, 'documents'),

(239, 40, 'archive.zip', 6.3, 'documents'),

(240, 41, 'landscape.jpg', 3.3, 'photos'),

(241, 42, 'instructions.txt', 1.4, 'documents'),

(242, 42, 'project.xlsx', 4.5, 'documents'),

(243, 44, 'proposal.docx', 2.4, 'documents'),

(244, 45, 'photo.png', 1.1, 'photos'),

(245, 44, 'notes.txt', 0.6, 'documents'),

(246, 44, 'report.docx', 2.3, 'documents'),

(247, 48, 'archive.zip', 6.7, 'documents'),

(248, 49, 'instructions.txt', 1.7, 'documents'),

(249, 50, 'invoice.pdf', 1.4, 'documents'),

(250, 41, 'sunset.mp4', 3.9, 'videos'),

(251, 52, 'project.docx', 2.2, 'documents'),

(252, 53, 'memo.txt', 1.3, 'documents'),

(253, 54, 'quarterly\_report.xlsx', 4.9, 'documents'),

(254, 55, 'beach\_day.jpg', 3.7, 'photos'),

(255, 56, 'notes.txt', 1.6, 'documents'),

(256, 57, 'report.docx', 2.6, 'documents'),

(257, 58, 'manual.pdf', 1.5, 'documents'),

(258, 59, 'archive.zip', 7.0, 'documents'),

(259, 60, 'landscape.mp4', 4.0, 'video'),

(260, 61, 'guide.txt', 1.9, 'documents'),

(261, 62, 'project\_plan.docx', 2.8, 'documents'),

(262, 63, 'photo.jpg', 1.8, 'photos'),

(263, 64, 'notes.txt', 1.0, 'documents'),

(264, 65, 'sunset.mp4', 4.2, 'video'),

(265, 66, 'archive.zip', 7.3, 'documents'),

(266, 67, 'report.docx', 3.0, 'documents'),

(267, 68, 'landscape.jpg', 4.5, 'photos'),

(268, 69, 'memo.txt', 2.1, 'documents'),

(269, 70, 'proposal.pdf', 2.0, 'documents'),

(270, 71, 'presentation.xlsx', 5.0, 'documents'),

(271, 72, 'meeting\_notes.txt', 1.1, 'documents'),

(272, 73, 'beach\_day.jpg', 4.8, 'photos'),

(273, 74, 'research.html', 3.2, 'link'),

(274, 75, 'user\_guide.pdf', 2.2, 'documents'),

(275, 76, 'notes.txt', 1.2, 'documents'),

(276, 77, 'archive.zip', 7.7, 'documents'),

(277, 58, 'landscape.jpg', 4.4, 'photos'),

(278, 59, 'project\_plan.docx', 2.3, 'documents'),

(279, 50, 'proposal.pdf', 2.4, 'documents'),

(280, 91, 'memo.txt', 3.5, 'documents'),

(281, 92, 'photo.jpg', 5.0, 'photos'),

(282, 86, 'instructions.txt', 2.5, 'documents'),

(283, 94, 'spreadsheet.xlsx', 5.5, 'documents'),

(284, 85, 'beach\_day.jpg', 4.7, 'photos'),

(285, 86, 'user\_guide.pdf', 2.6, 'documents'),

(286, 97, 'proposal.docx', 3.8, 'documents'),

(287, 88, 'report.pdf', 2.6, 'documents'),

(288, 99, 'archive.zip', 8.0, 'documents'),

(289, 90, 'photo.jpg', 5.2, 'photos'),

(290, 91, 'instructions.txt', 2.8, 'documents'),

(291, 92, 'project\_plan.docx', 3.7, 'documents'),

(292, 93, 'user\_guide.pdf', 2.8, 'documents'),

(293, 54, 'notes.txt', 2.0, 'documents'),

(294, 5, 'beach\_day.jpg', 5.4, 'photos'),

(295, 6, 'archive.zip', 8.3, 'documents'),

(296, 7, 'proposal.docx', 4.0, 'documents'),

(297, 58, 'photo.jpg', 5.5, 'photos'),

(298, 99, 'memo.txt', 3.0, 'documents'),

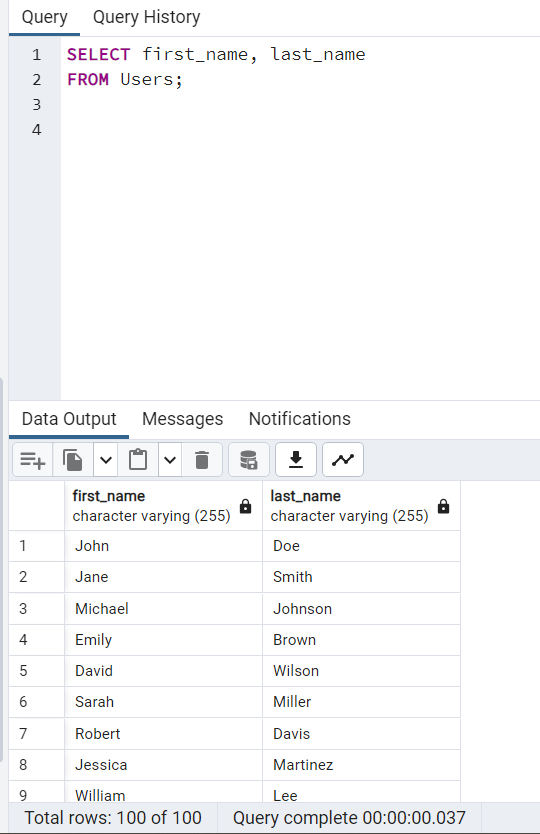
(299, 100, 'report.pdf', 3.0,'documents');

**English Query 1) Retrieve all users' first names and last names.**

**SQL Query** - SELECT first\_name, last\_name

FROM User;

**Count of Tuples**- 100

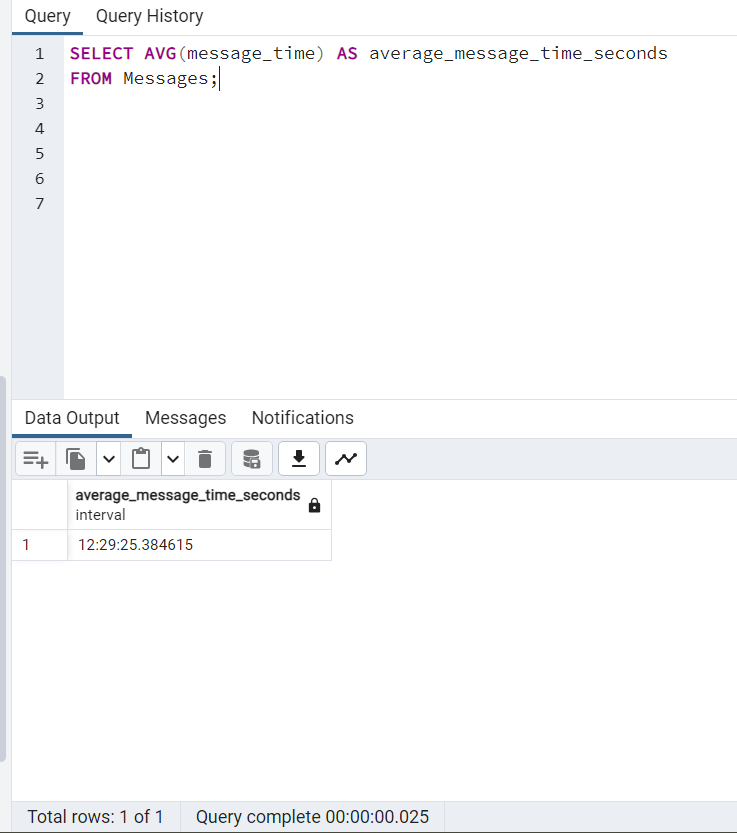


**English Query 2) Calculate the average time of messages sent.**

**SQL Query** – SELECT AVG(message\_time) AS average\_message\_time\_seconds

FROM Messages;

**Count of Tuples**-1



**English Query 3) Find the messages sent by users that contain emojis.**

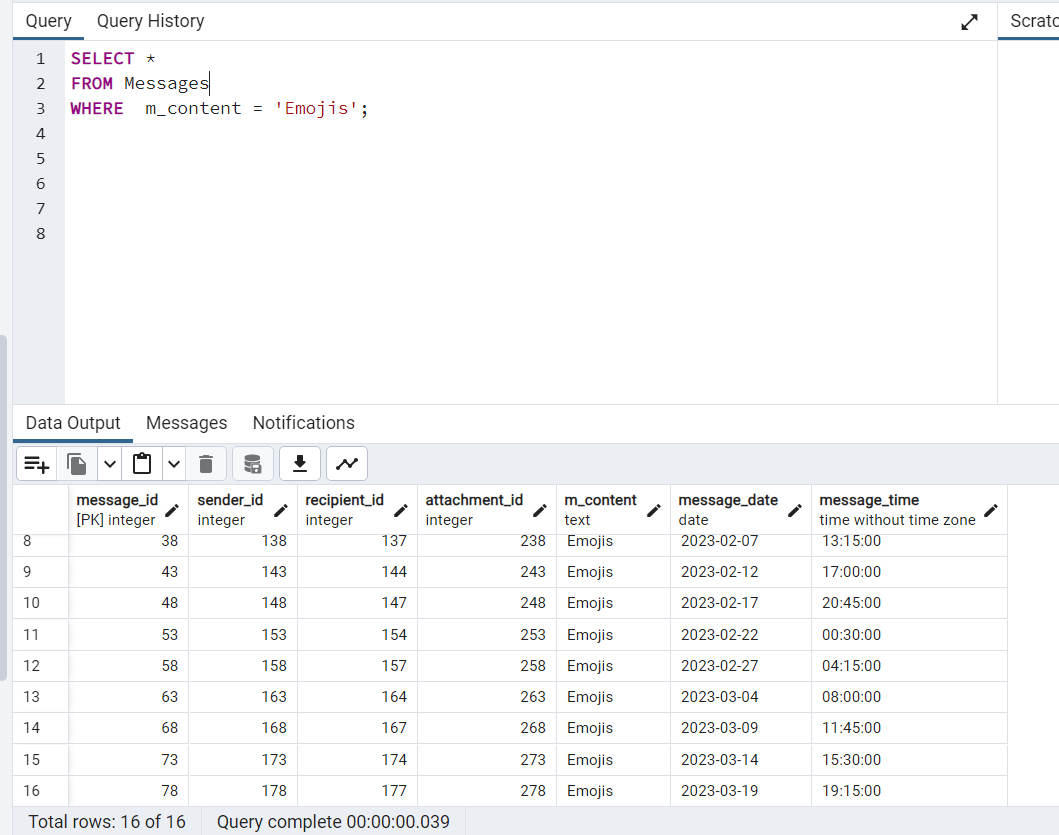
**SQL Query** -

SELECT \*

FROM Message

WHERE m\_ content = 'Emojis';

**Count of Tuples**-16



**English Query 4) Count the number of different types of messages (Text, Voice, Emojis, Sticker, Gif) in the table.**

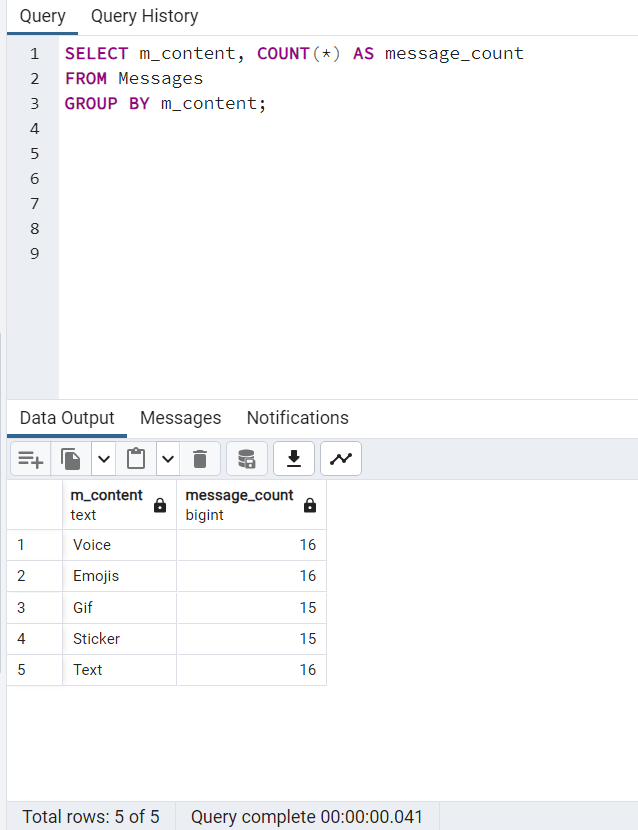
**SQL Query** –

SELECT m\_content, COUNT(\*) AS message\_count

FROM Messages

GROUP BY m\_content;

**Count of Tuples**- 5



**English Query 5) Find messages sent on or after '2023-02-15'.**

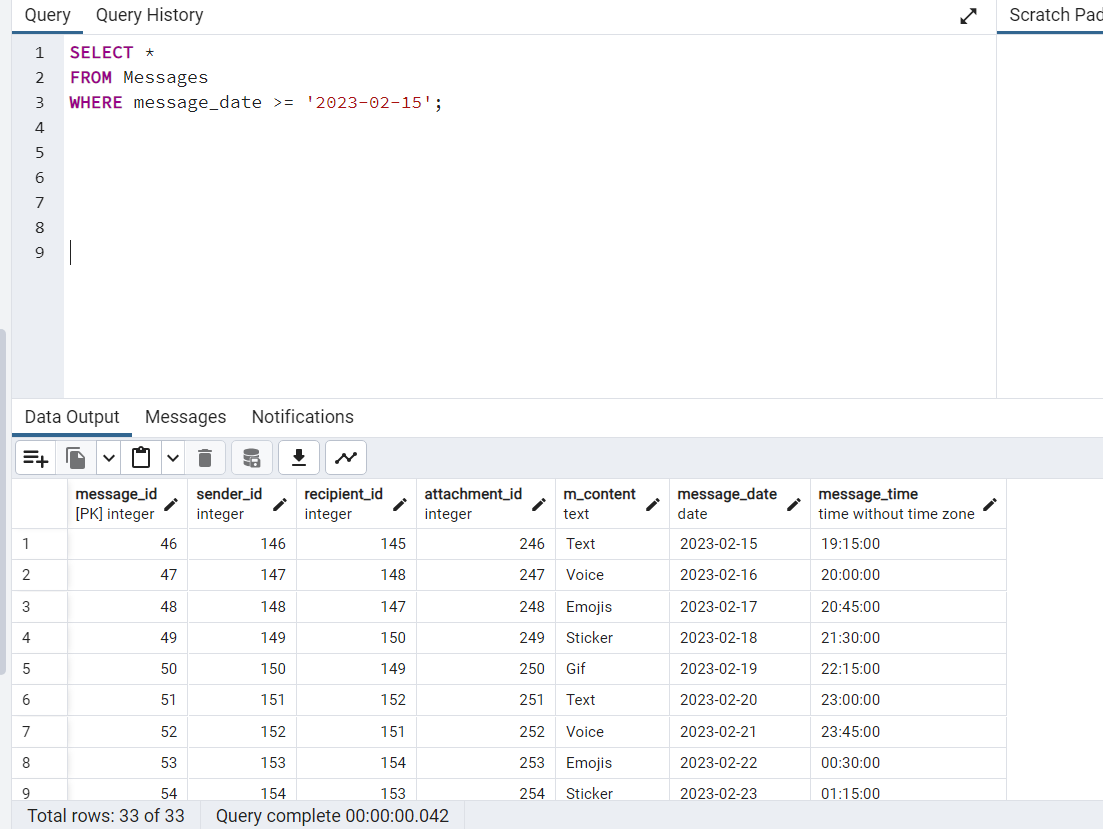
**SQL Query** –

SELECT \*

FROM Messages

WHERE message\_date >= '2023-02-15';

**Count of Tuples**- 33



**English Query 6) Retrieve all group chats with a member size greater than 10.**

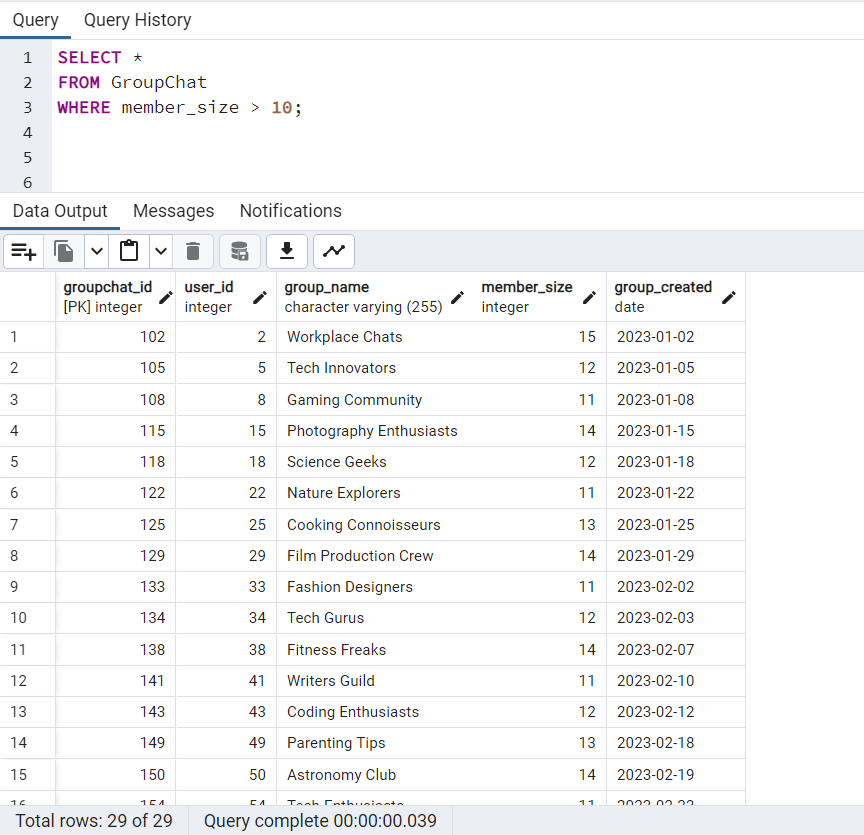
**SQL Query** -

SELECT \*

FROM GroupChat

WHERE member\_size > 10;

**Count of Tuples**- 29



**English Query 7) Find the group names of group chats created on or after '2023-02-15'.**

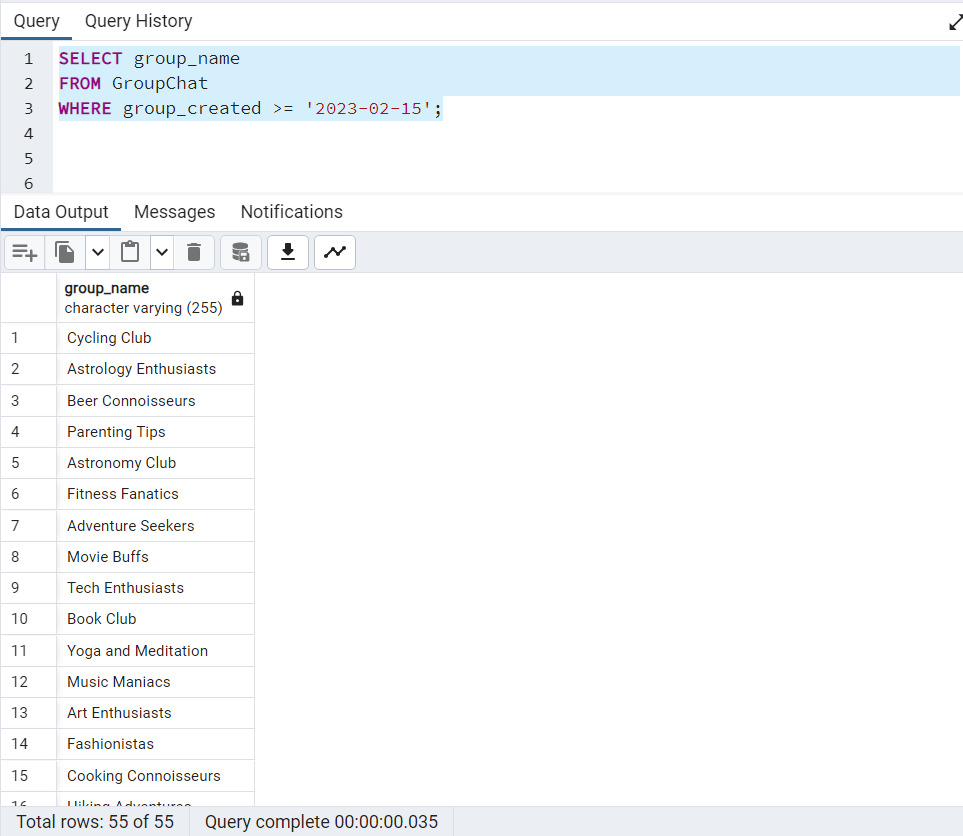
**SQL Query** -

SELECT group\_name

FROM GroupChat

WHERE group\_created >= '2023-02-15';

**Count of Tuples**- 55



**English Query 8) Retrieve all attachments with a type of 'photos' and a file size (size\_mb) greater than 3.0 MB.**

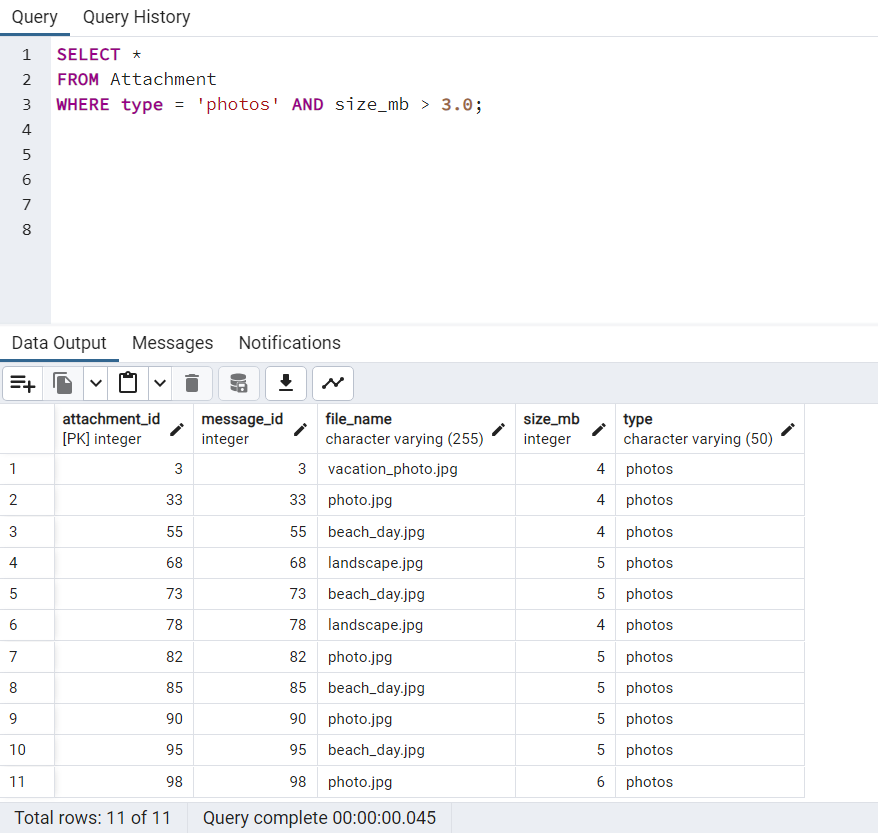
**SQL Query** -

SELECT \*

FROM Attachment

WHERE type = 'photos' AND size\_mb > 3.0;

**Count of Tuples**- 11



**English Query 9) Retrieve the total file size of all attachments with a type of 'documents'.**

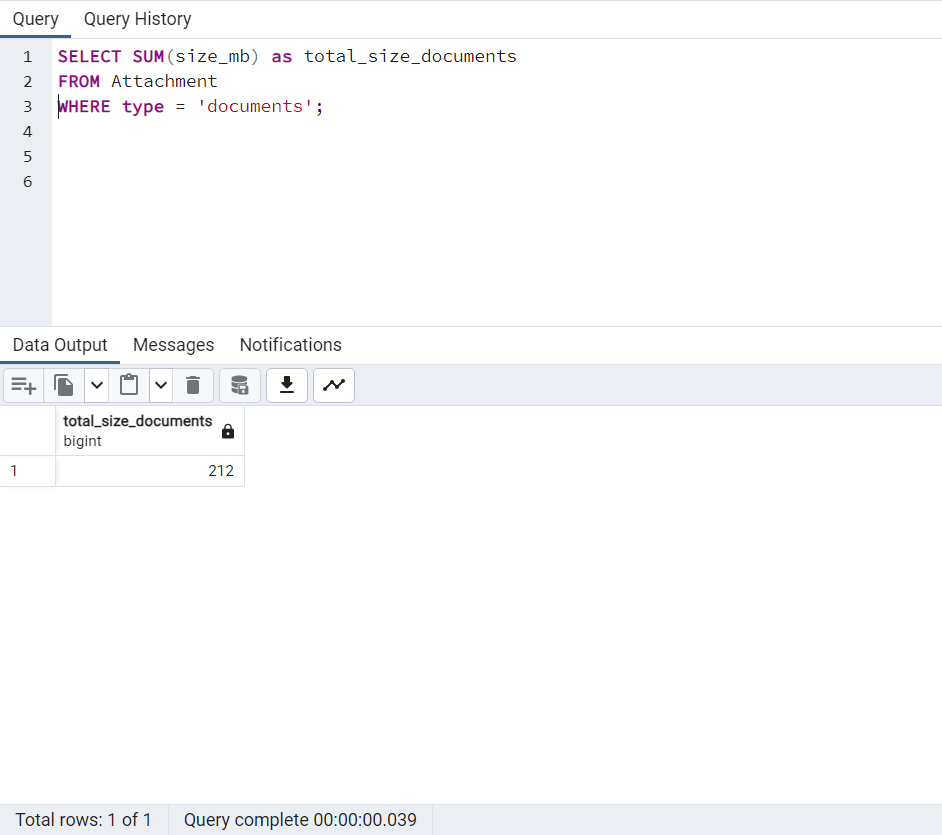
**SQL Query** –

SELECT SUM(size\_mb) as total\_size\_documents

FROM Attachment

WHERE type = 'documents';

**Count of Tuples**- 1



**English Query 10) Retrieve the names of attachments that have 'pdf' in their file\_name.**

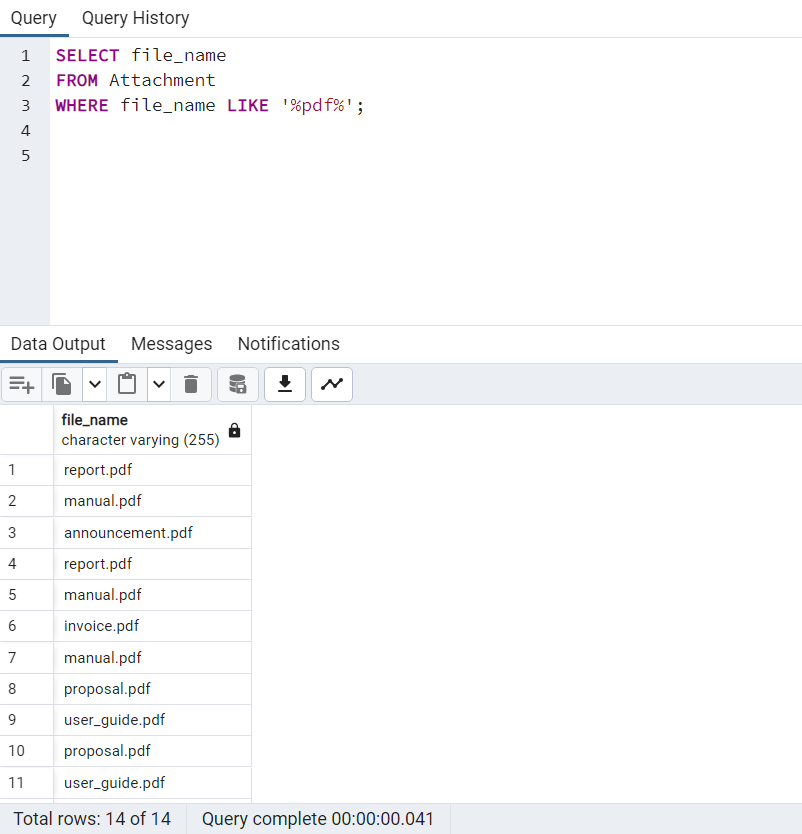
**SQL Query** -

SELECT file\_name

FROM Attachment

WHERE file\_name LIKE '%pdf%';

**Count of Tuples**- 14



**English Query 11) Retrieve notifications sent on '2023-11-05' after '10:00:00'.**

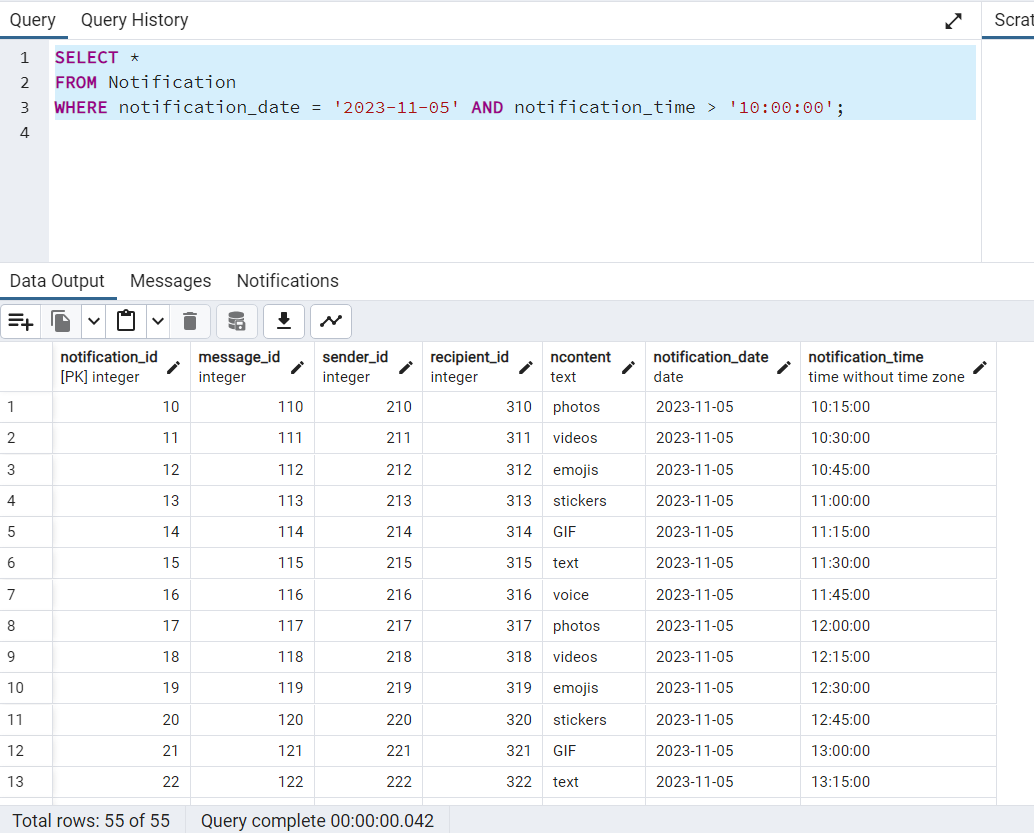
**SQL Query** -

SELECT \*

FROM Notification

WHERE notification\_date = '2023-11-05' AND notification\_time > '10:00:00';

**Count of Tuples**- 55



**English Query 12) Retrieve the latest notification sent.**

**SQL Query** -

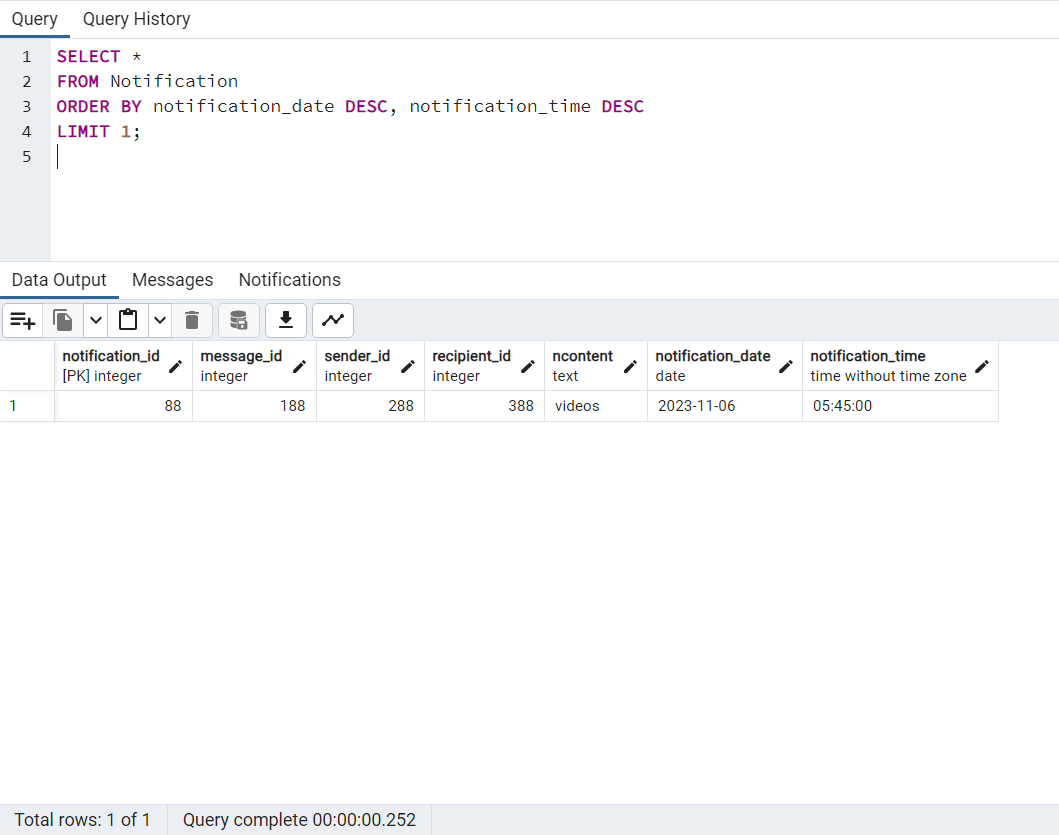
SELECT \*

FROM Notification

ORDER BY notification\_date DESC, notification\_time DESC

LIMIT 1;

**Count of Tuples**- 1



**English Query 13) Retrieve notifications sent on '2023-11-06' with 'GIF' content.**

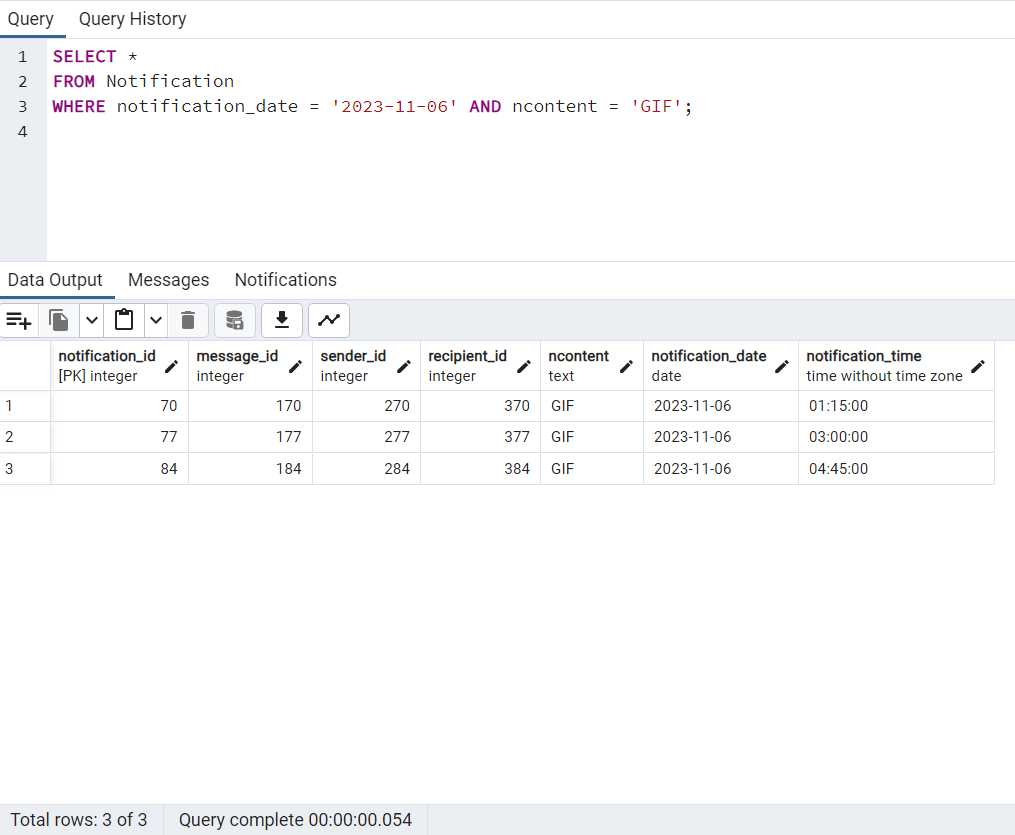
**SQL Query** -

SELECT \*

FROM Notification

WHERE notification\_date = '2023-11-06' AND ncontent = 'GIF';

**Count of Tuples**- 3



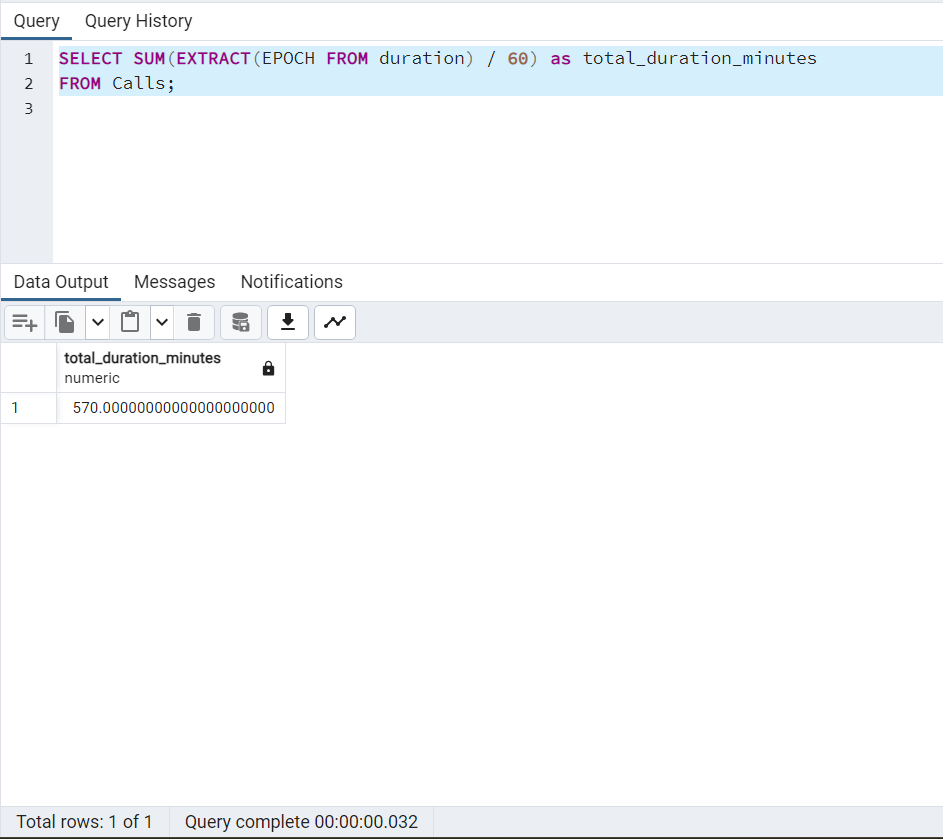
**English Query 14) Retrieve the total duration of all calls in minutes.**

**SQL Query** -

SELECT SUM(EXTRACT(EPOCH FROM duration) / 60) as total\_duration\_minutes

FROM Calls;

**Count of Tuples**- 1



**English Query 15) Retrieve the user who made the longest call (voice or video) and the duration in minutes.**

**SQL Query** –

SELECT user\_id, MAX(EXTRACT(EPOCH FROM duration) / 60) as max\_call\_duration\_minutes

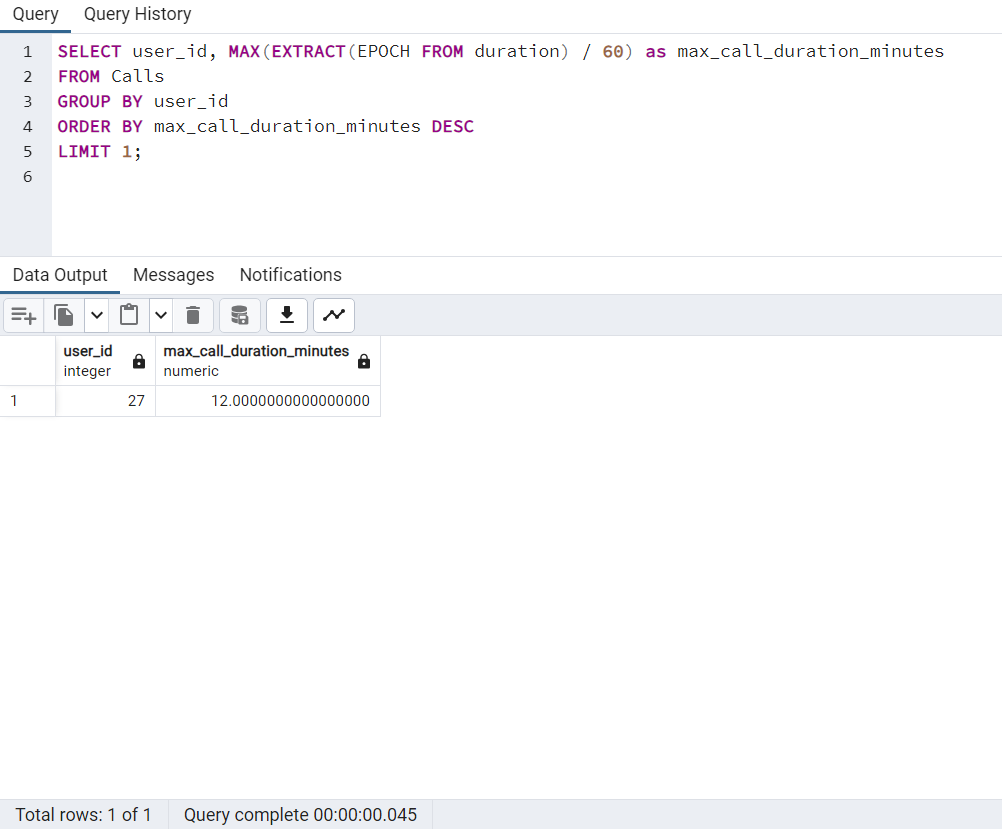
FROM Calls

GROUP BY user\_id

ORDER BY max\_call\_duration\_minutes DESC

LIMIT 1;

**Count of Tuples**- 1



**English Query 16) Retrieve the user who posted the most public status updates.**

**SQL Query** -

SELECT user\_id, COUNT(\*) as public\_status\_count

FROM Status

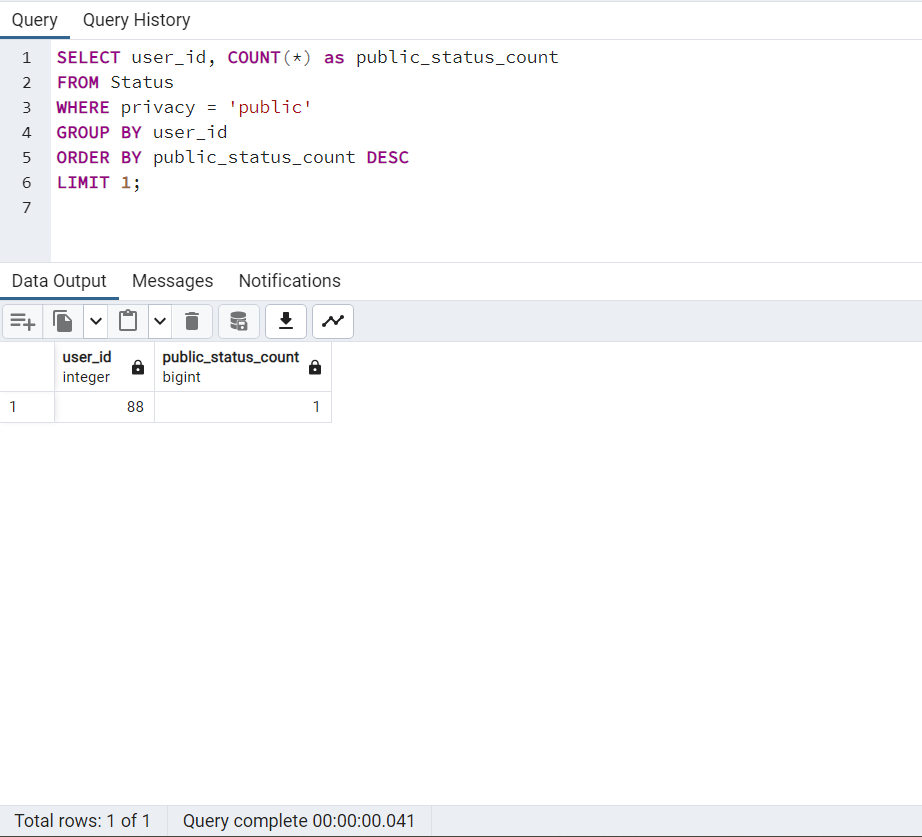
WHERE privacy = 'public'

GROUP BY user\_id

ORDER BY public\_status\_count DESC

LIMIT 1;

**Count of Tuples**- 1



**English Query 17) Retrieve the user who posted the most public status updates.**

**SQL Query** –

SELECT user\_id, COUNT(\*) as public\_status\_count

FROM Status

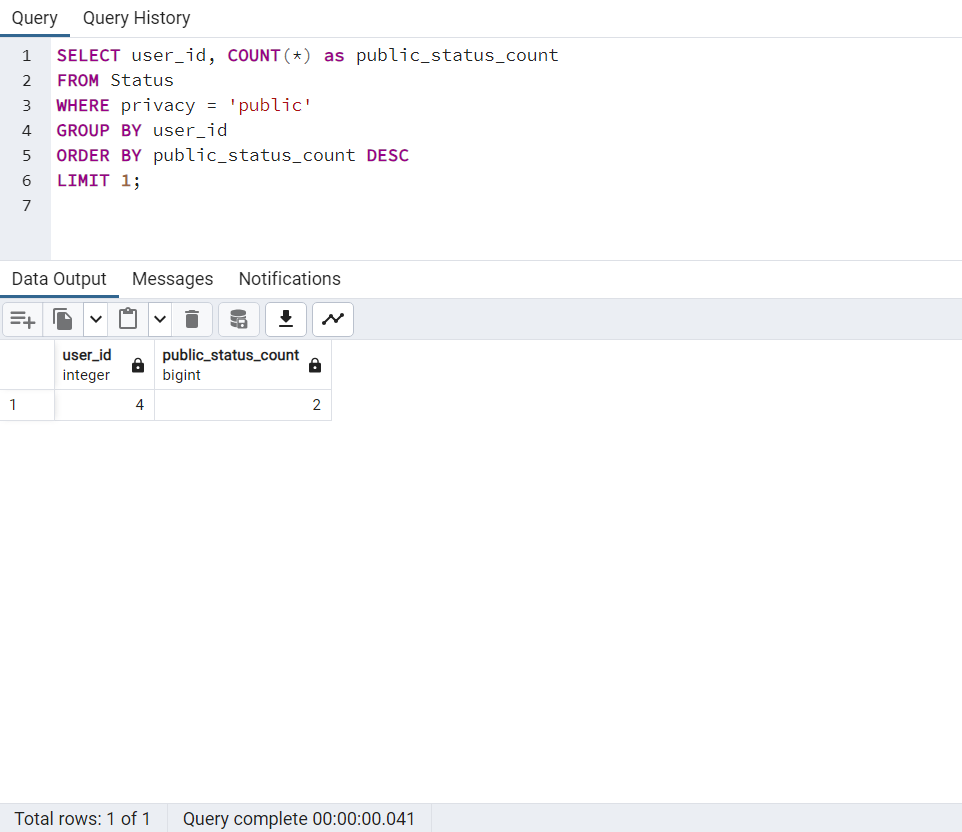
WHERE privacy = 'public'

GROUP BY user\_id

ORDER BY public\_status\_count DESC

LIMIT 1;

**Count of Tuples**- 1



**English Query 18) Retrieve the total number of status updates posted by user 1.**

**SQL Query** -

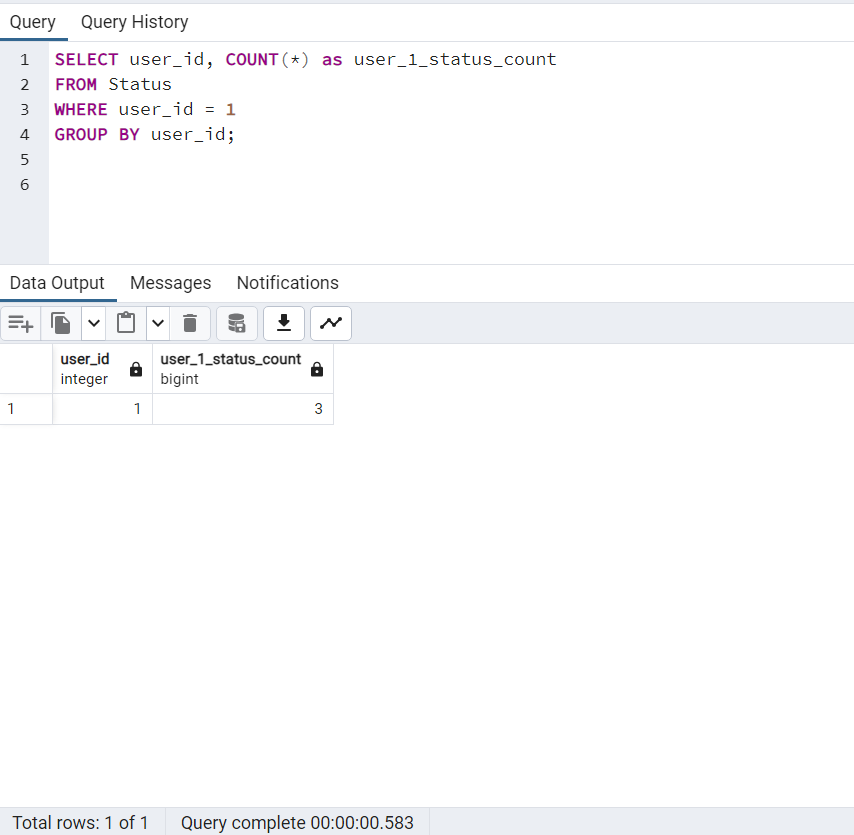
SELECT user\_id, COUNT(\*) as user\_1\_status\_count

FROM Status

WHERE user\_id = 1

GROUP BY user\_id;

**Count of Tuples**- 1



**English Query 19) Retrieve the attachment with the largest size\_mb.**

**SQL Query** -

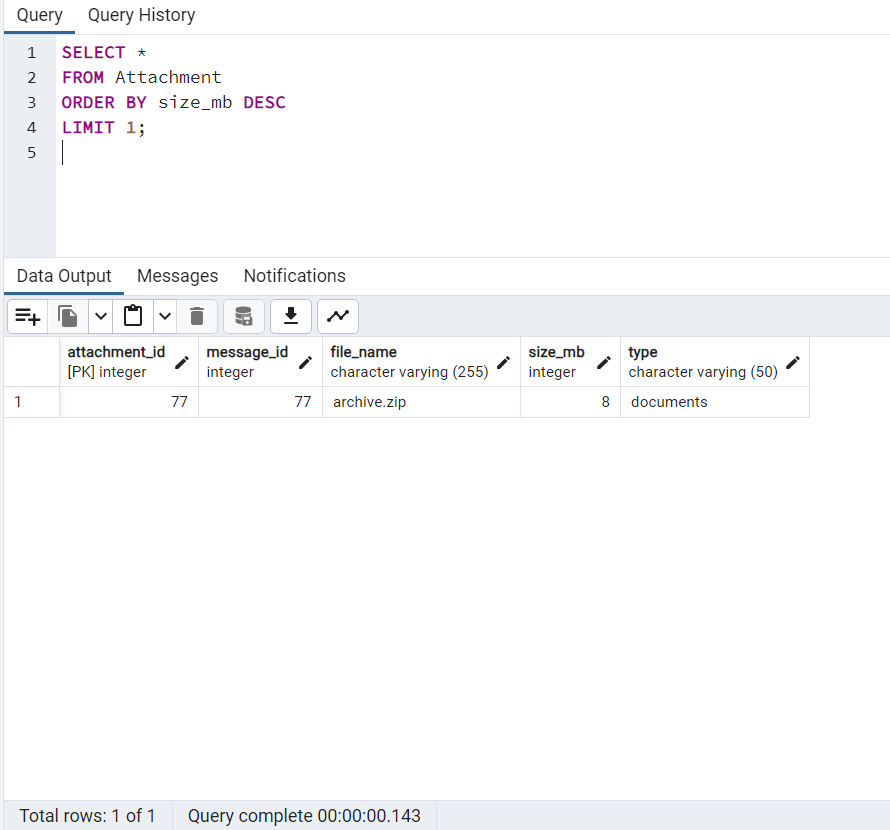
SELECT \*

FROM Attachment

ORDER BY size\_mb DESC

LIMIT 1;

**Count of Tuples**- 1



**English Query 20) Retrieve the count of notifications with 'voice' content.**

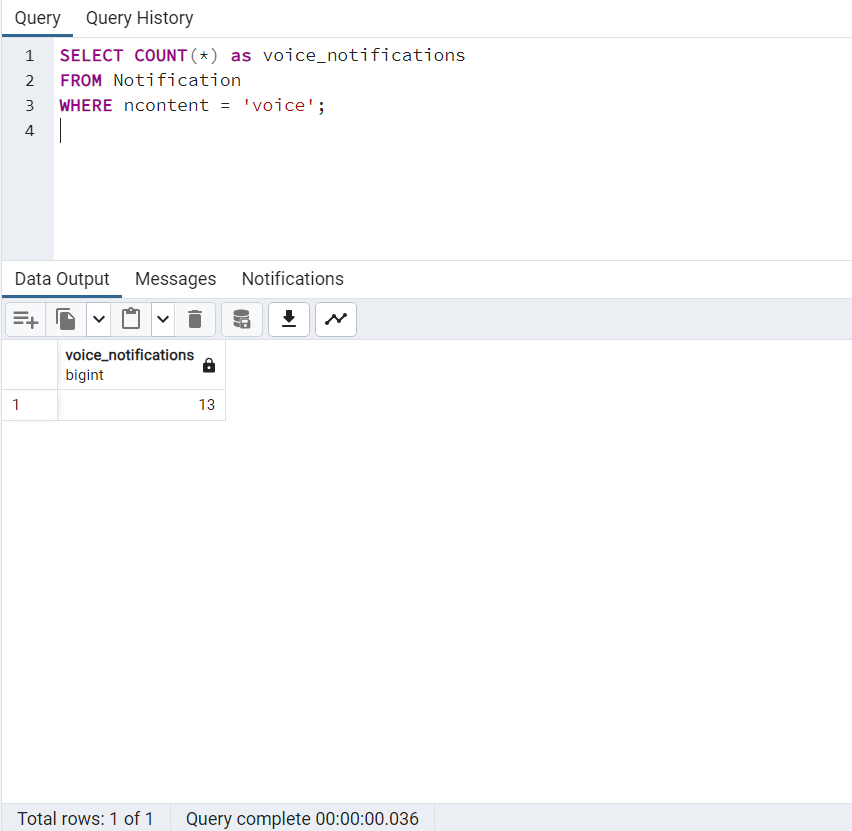
**SQL Query** -

SELECT COUNT(\*) as voice\_notifications

FROM Notification

WHERE ncontent = 'voice';

**Count of Tuples**- 1



**English Query 21)** Count the number of chats each user has participated in and display the results in descending order.

**SQL Query** -

SELECT u.user\_id, u.first\_name, u.last\_name, COUNT(c.chat\_id) AS chat\_count

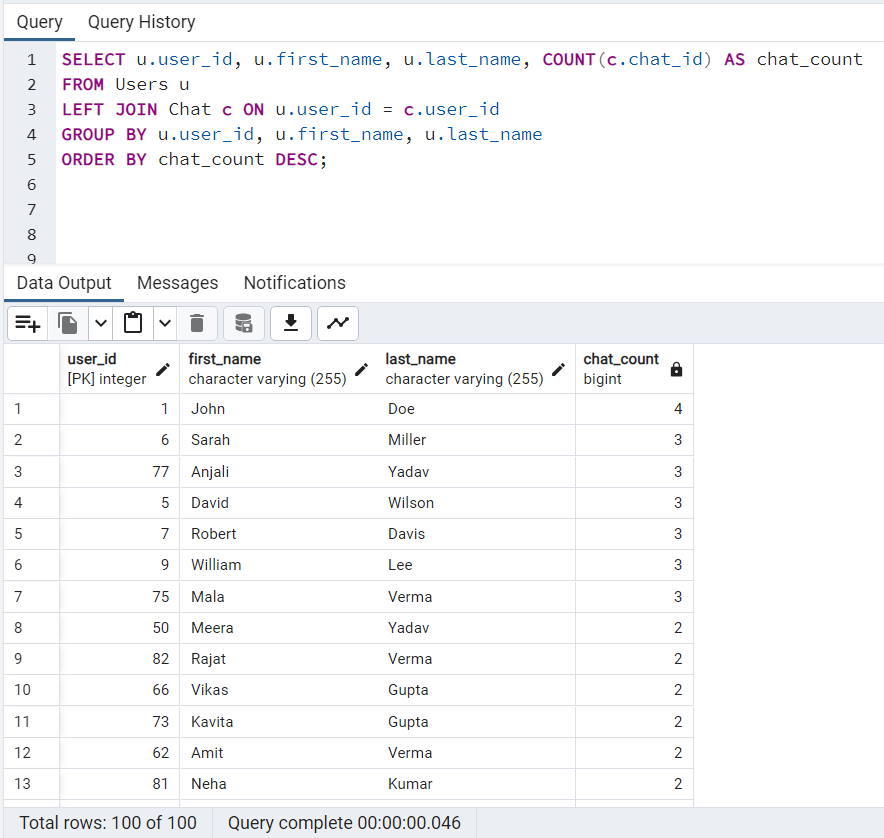
FROM Users u

LEFT JOIN Chat c ON u.user\_id = c.user\_id

GROUP BY u.user\_id, u.first\_name, u.last\_name

ORDER BY chat\_count DESC;

**Count of Tuples**- 100



**English Query 22)** Find the user with the most chats.

**SQL Query** -

SELECT u.user\_id, u.first\_name, u.last\_name, COUNT(c.chat\_id) AS chat\_count

FROM Users u

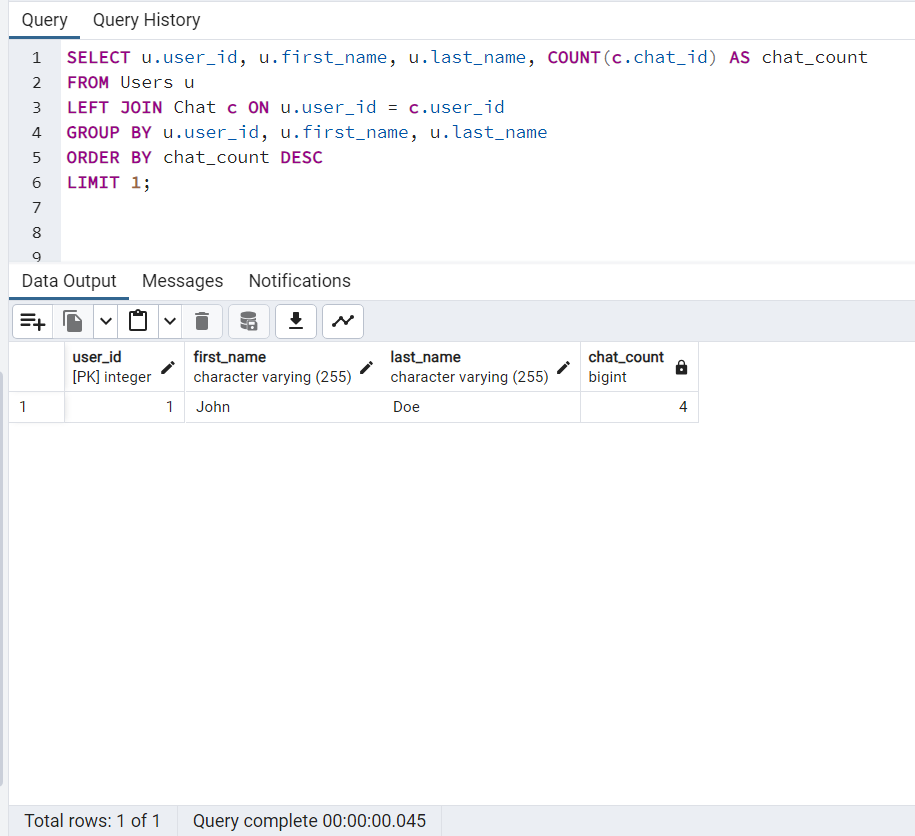
LEFT JOIN Chat c ON u.user\_id = c.user\_id

GROUP BY u.user\_id, u.first\_name, u.last\_name

ORDER BY chat\_count DESC

LIMIT 1;

**Count of Tuples**- 1



**English Query 23)** Calculate the average number of chats per user.

**SQL Query** -

SELECT AVG(chat\_count) AS average\_chats\_per\_user

FROM (

SELECT u.user\_id, COUNT(c.chat\_id) AS chat\_count

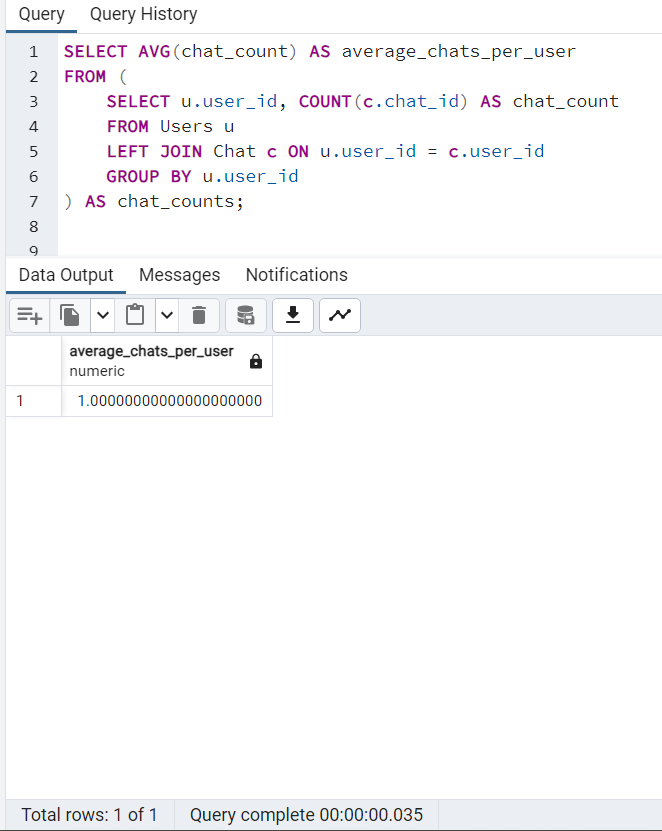
FROM Users u

LEFT JOIN Chat c ON u.user\_id = c.user\_id

GROUP BY u.user\_id

) AS chat\_counts;

**Count of Tuples**- 1



**English Query 24)** List the users who have not participated in any chat.

**SQL Query** -

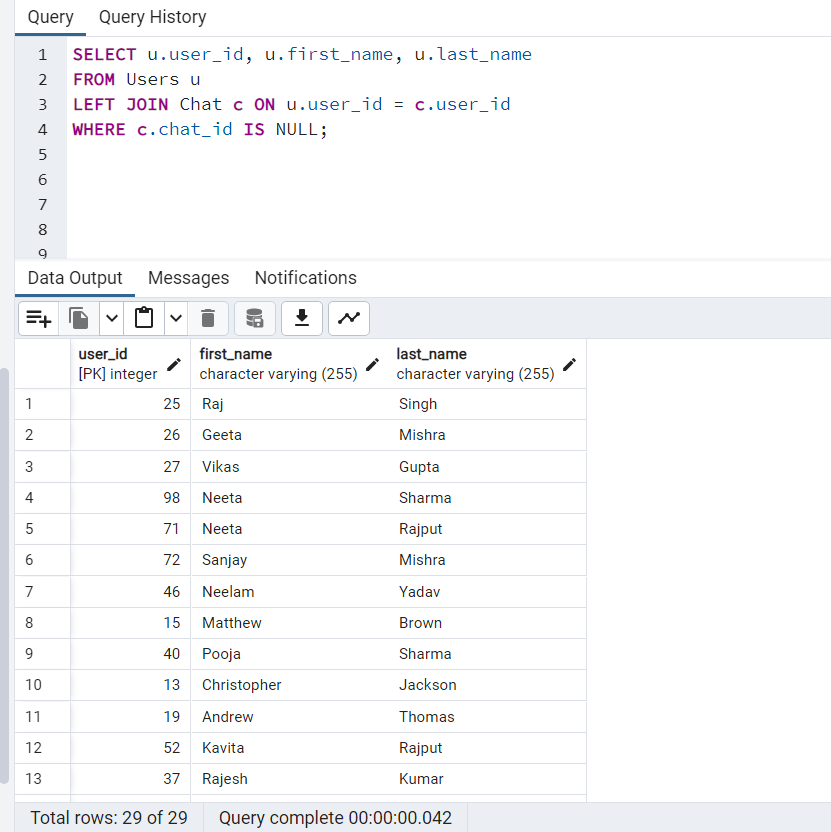
SELECT u.user\_id, u.first\_name, u.last\_name

FROM Users u

LEFT JOIN Chat c ON u.user\_id = c.user\_id

WHERE c.chat\_id IS NULL;

**Count of Tuples**- 29



**English Query 25)** Retrieve the first names and last names of users who have posted a public photo status:

**SQL Query** -

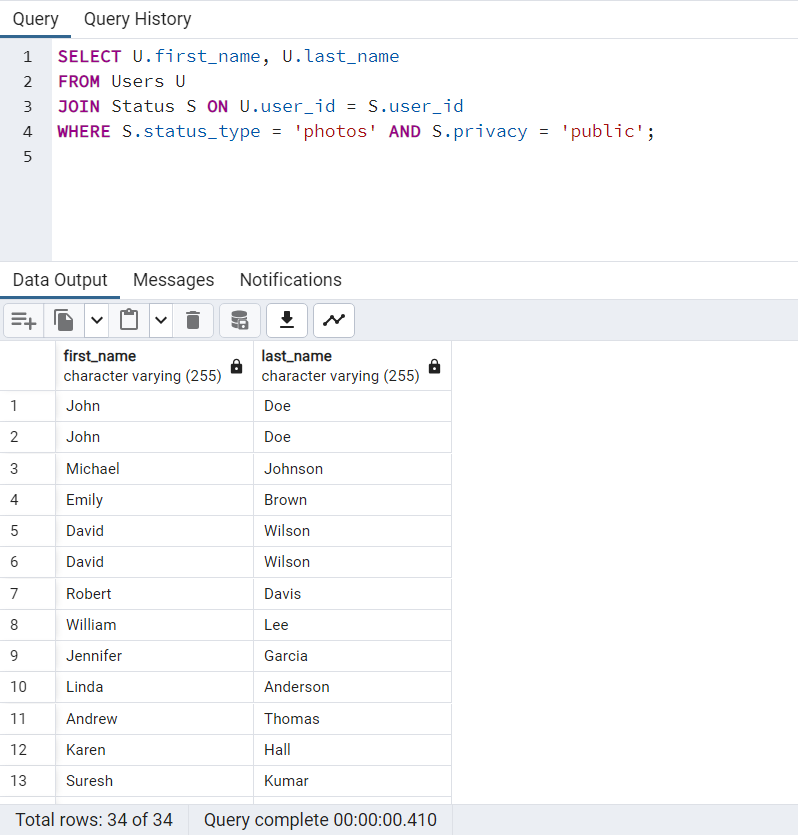
SELECT U.first\_name, U.last\_name

FROM Users U

JOIN Status S ON U.user\_id = S.user\_id

WHERE S.status\_type = 'photos' AND S.privacy = 'public';

**Count of Tuples**- 34



**English Query 26)** Find the user with the most number of close friends text statuses:

**SQL Query** -

SELECT U.first\_name, U.last\_name, COUNT(S.status\_id) AS close\_friends\_text\_count

FROM Users U

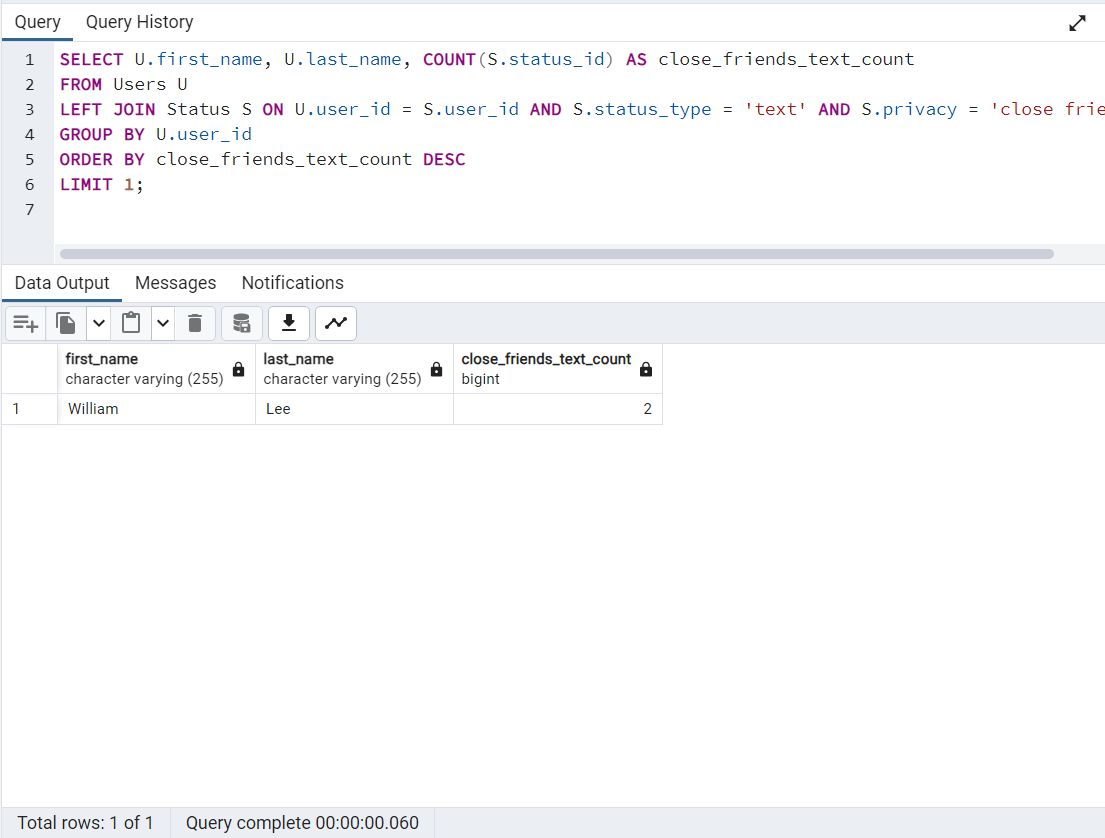
LEFT JOIN Status S ON U.user\_id = S.user\_id AND S.status\_type = 'text' AND S.privacy = 'close friends'

GROUP BY U.user\_id

ORDER BY close\_friends\_text\_count DESC

LIMIT 1;

**Count of Tuples**- 1



**English Query 27)**  Retrieve the first names and last names of users who have posted at least one status of each type (text, photos, videos):

**SQL Query** -

SELECT U.first\_name, U.last\_name

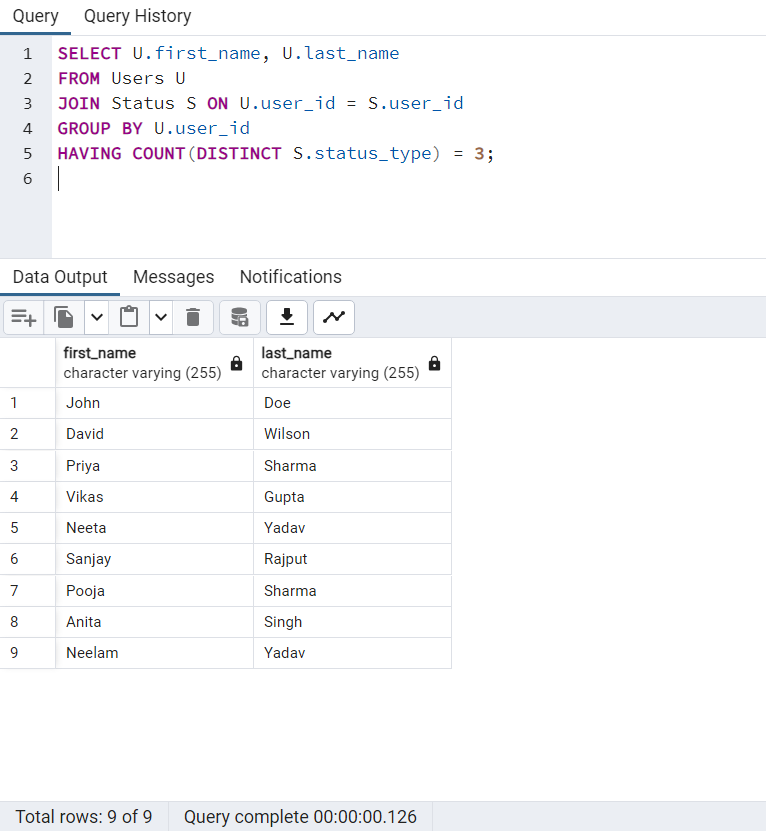
FROM Users U

JOIN Status S ON U.user\_id = S.user\_id

GROUP BY U.user\_id

HAVING COUNT(DISTINCT S.status\_type) = 3;

**Count of Tuples**- 9



**English Query 28) Find the average call duration for video calls made by users who were born in or after 1990:**

**SQL Query** -

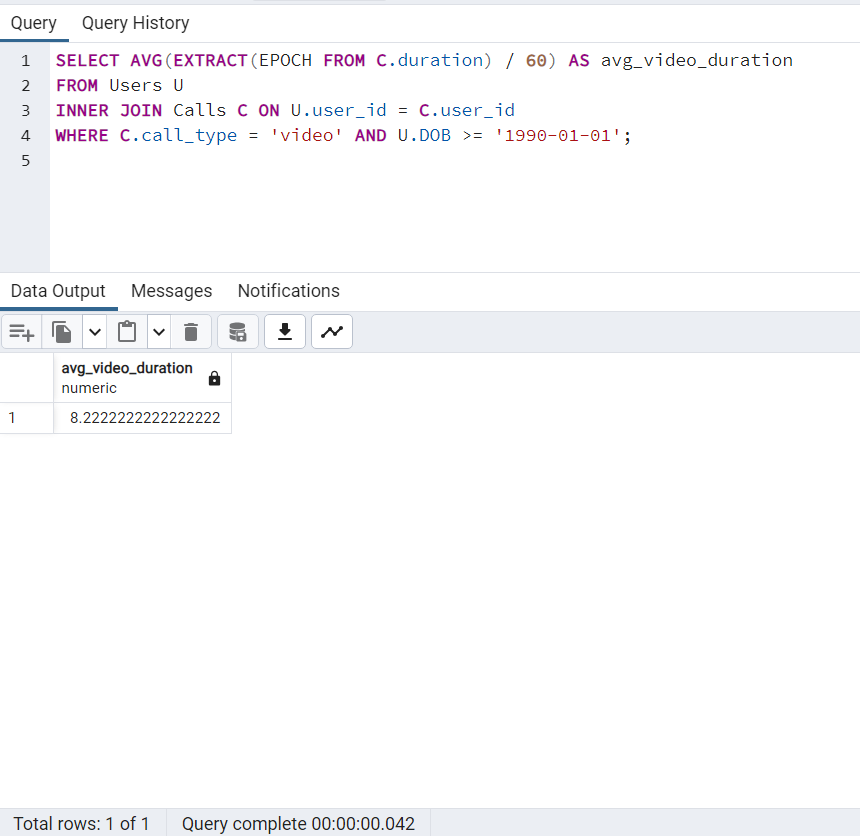
SELECT AVG(EXTRACT(EPOCH FROM C.duration) / 60) AS avg\_video\_duration

FROM Users U

INNER JOIN Calls C ON U.user\_id = C.user\_id

WHERE C.call\_type = 'video' AND U.DOB >= '1990-01-01';

**Count of Tuples**- 1



**English Query 29) Retrieve the total size of attachments (in MB) sent in messages with 'Gif' content by joining Messages and Attachment tables.**

**SQL Query** -

SELECT M.m\_content, SUM(A.size\_mb) AS total\_size\_mb

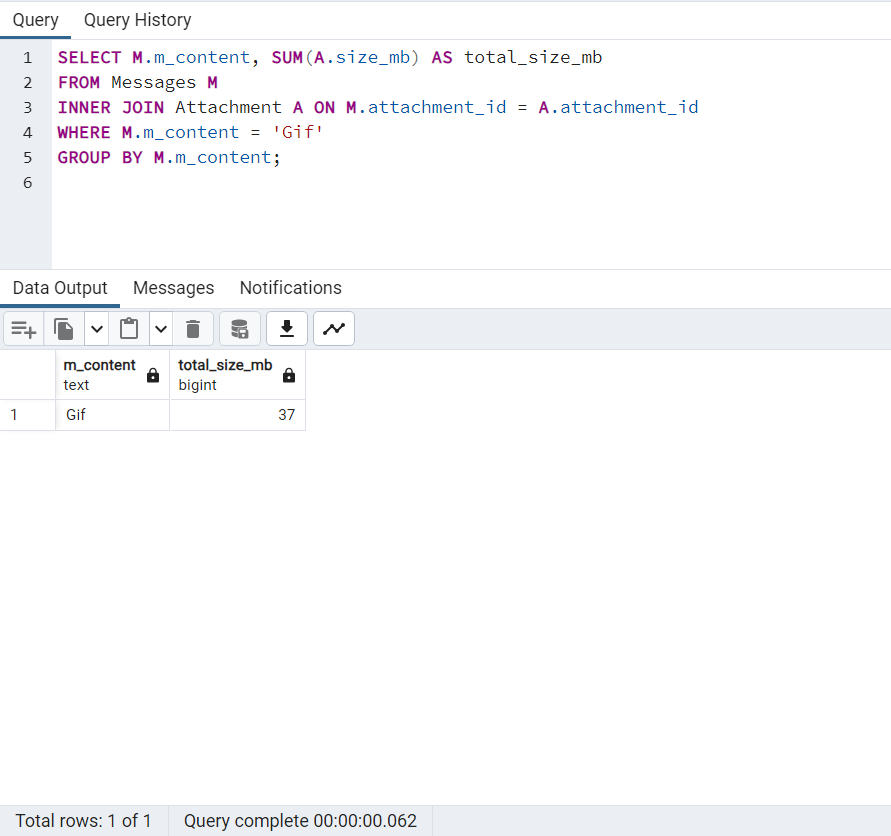
FROM Messages M

INNER JOIN Attachment A ON M.attachment\_id = A.attachment\_id

WHERE M.m\_content = 'Gif'

GROUP BY M.m\_content;

**Count of Tuples**- 1



**English Query 30) List messages with their attachments for the sender with 'sender\_id' 140.**

**SQL Query** -

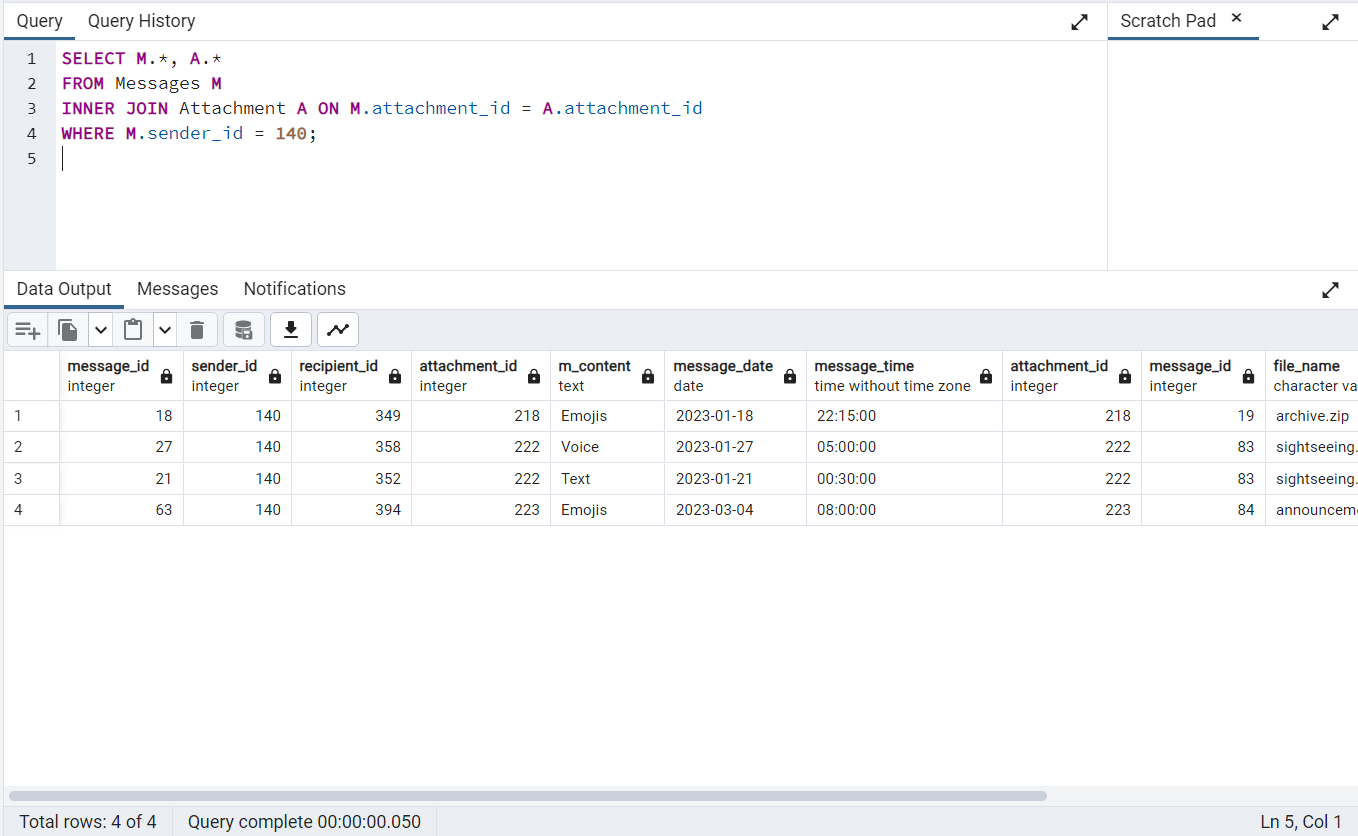
SELECT M.\*, A.\*

FROM Messages M

INNER JOIN Attachment A ON M.attachment\_id = A.attachment\_id

WHERE M.sender\_id = 140;

**Count of Tuples**- 4



**English Query 31) Find the top 5 senders who sent the most messages with attachments and their total attachment size (in MB).**

**SQL Query** -

SELECT M.sender\_id, COUNT(M.message\_id) AS message\_count, SUM(A.size\_mb) AS total\_size\_mb

FROM Messages M

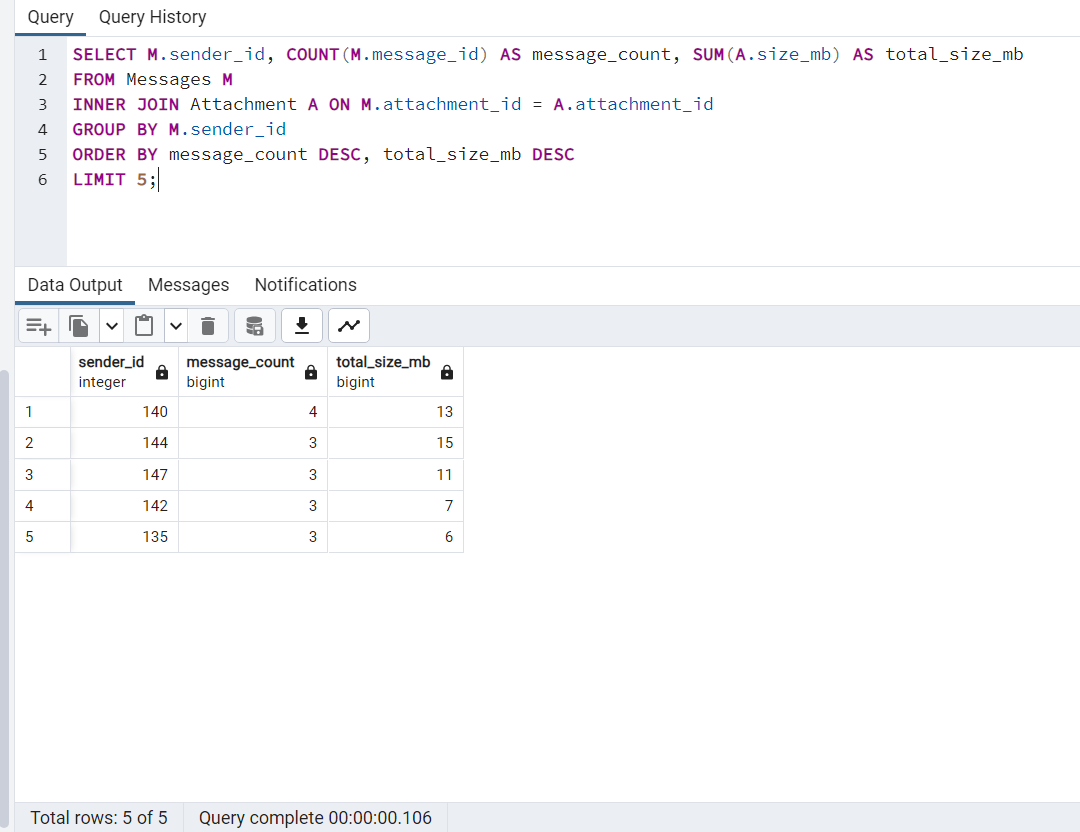
INNER JOIN Attachment A ON M.attachment\_id = A.attachment\_id

GROUP BY M.sender\_id

ORDER BY message\_count DESC, total\_size\_mb DESC

LIMIT 5;

**Count of Tuples**- 5



**English Query 32) Get the total number of notifications for each message content type.**

**SQL Query** -

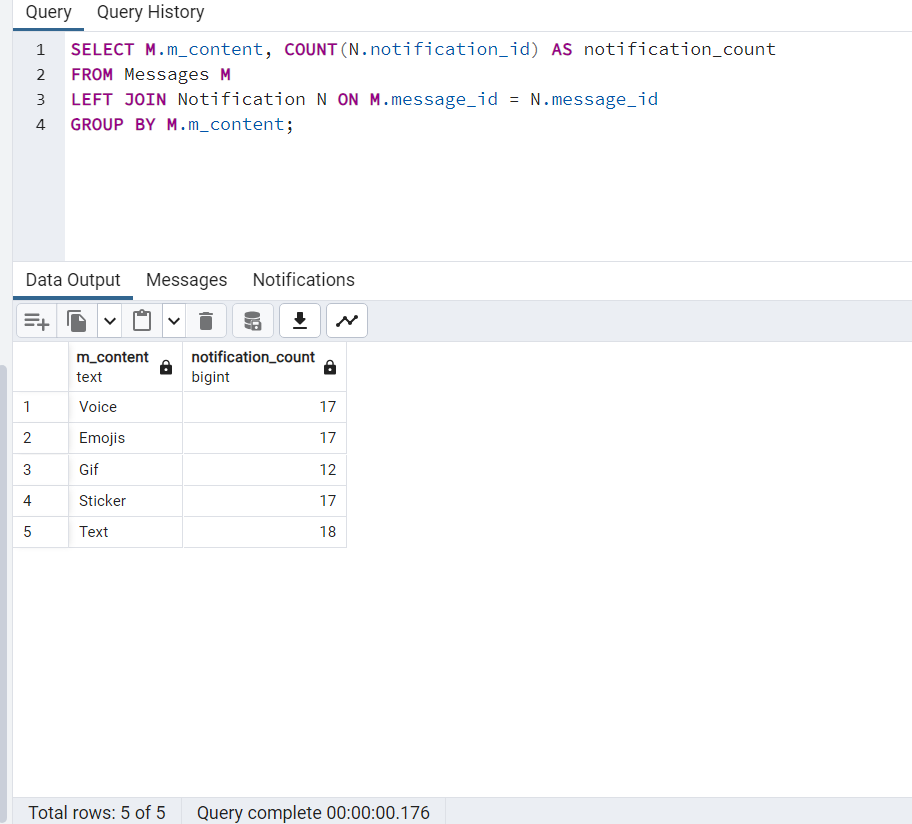
SELECT M.m\_content, COUNT(N.notification\_id) AS notification\_count

FROM Messages M

LEFT JOIN Notification N ON M.message\_id = N.message\_id

GROUP BY M.m\_content;

**Count of Tuples**- 5



**English Query 33) Find the recipient who received the most messages with 'Voice' content.**

**SQL Query** -

SELECT M.recipient\_id, COUNT(M.message\_id) AS message\_count

FROM Messages M

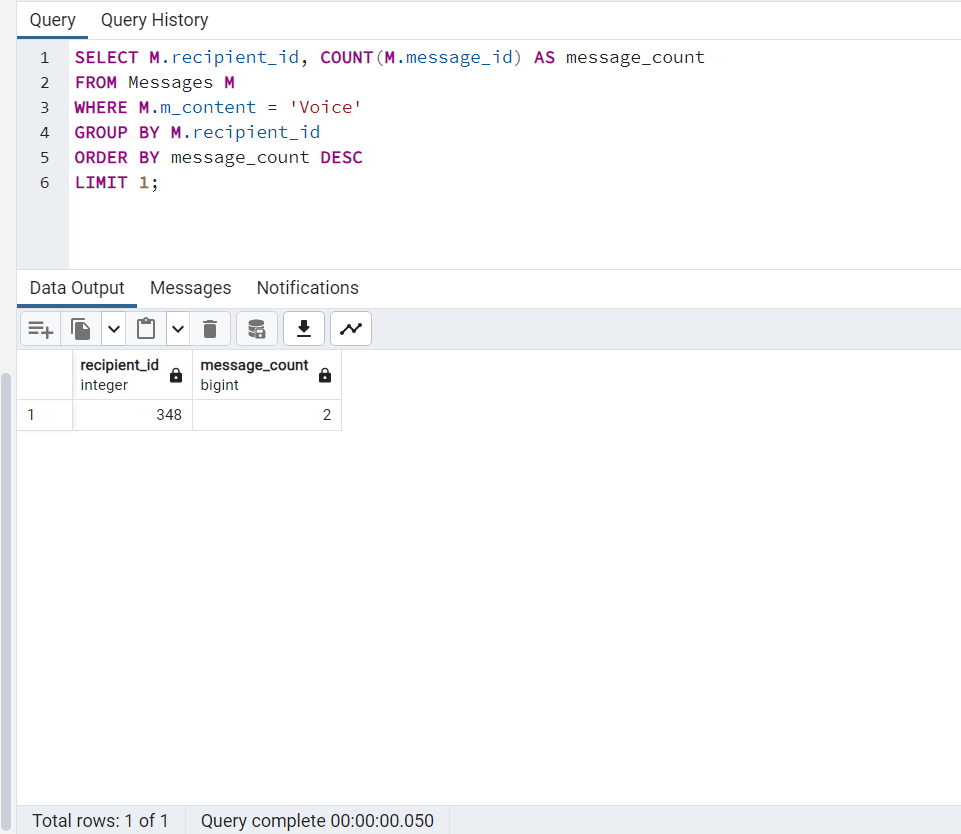
WHERE M.m\_content = 'Voice'

GROUP BY M.recipient\_id

ORDER BY message\_count DESC

LIMIT 1;

**Count of Tuples**-1



**English Query 34) Get the number of notifications sent by each sender on '2023-11-05'.**

**SQL Query** -

SELECT M.sender\_id, COUNT(N.notification\_id) AS notification\_count

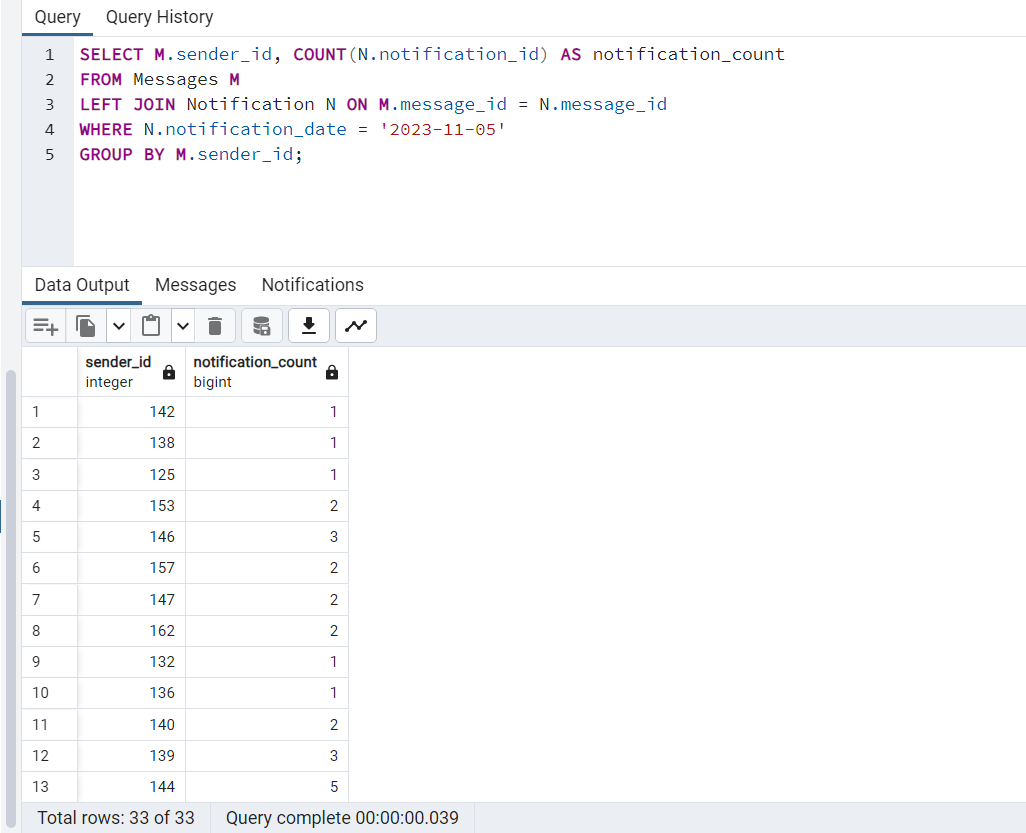
FROM Messages M

LEFT JOIN Notification N ON M.message\_id = N.message\_id

WHERE N.notification\_date = '2023-11-05'

GROUP BY M.sender\_id;

**Count of Tuples**- 33



**English Query 35) List all messages and their attachments for recipients who have received notifications on '2023-11-05'.**

**SQL Query** -

SELECT M.\*, A.\*

FROM Messages M

INNER JOIN Attachment A ON M.attachment\_id = A.attachment\_id

WHERE M.recipient\_id IN (

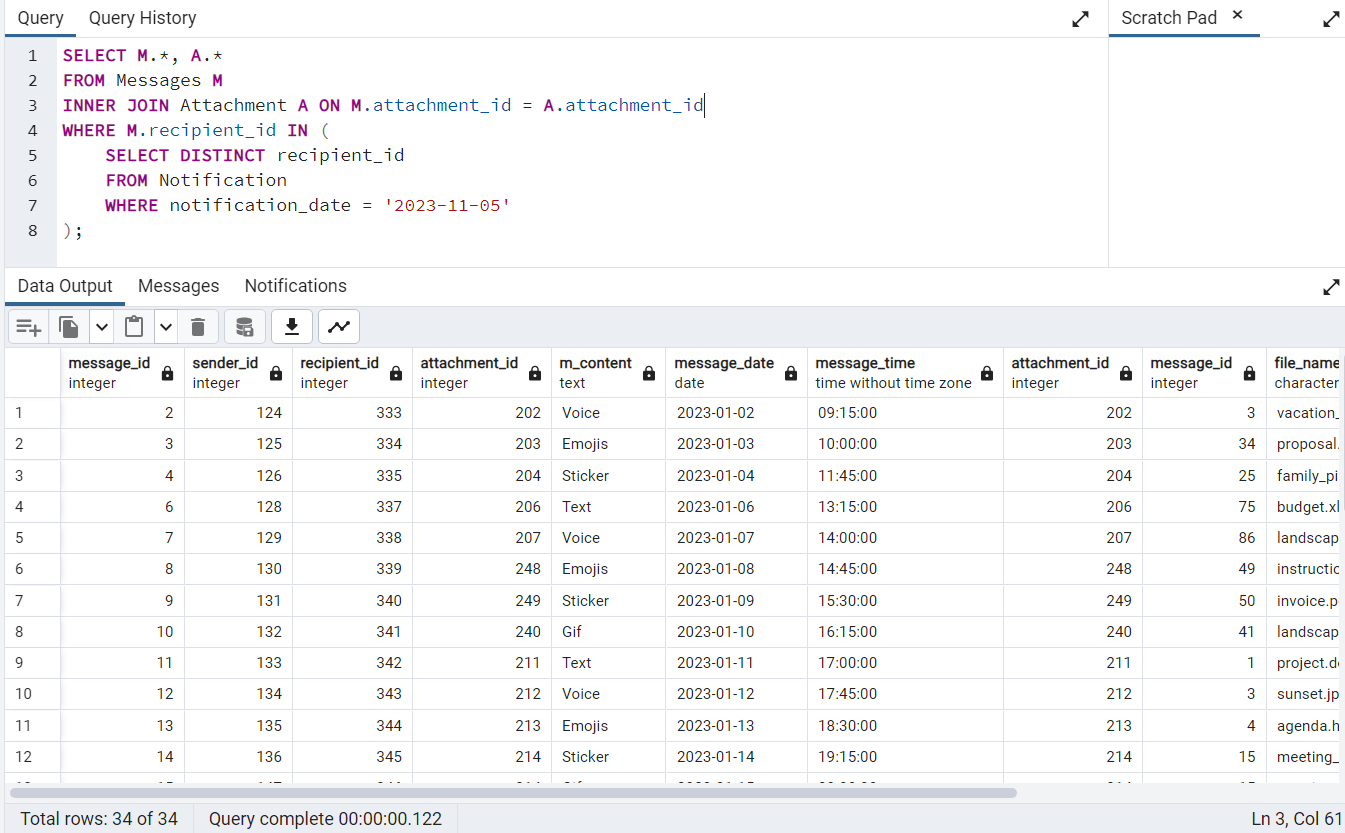
SELECT DISTINCT recipient\_id

FROM Notification

WHERE notification\_date = '2023-11-05'

);

**Count of Tuples**- 34



**English Query 36) Retrieve the largest attachments, in terms of size, for each attachment type within the 'Messages' table.**

**SQL Query** -

WITH RankedAttachments AS

(

SELECT M.message\_id, A.type, A.file\_name, A.size\_mb, ROW\_NUMBER ()

OVER

(PARTITION BY A.type

ORDER BY A.size\_mb DESC) AS rn

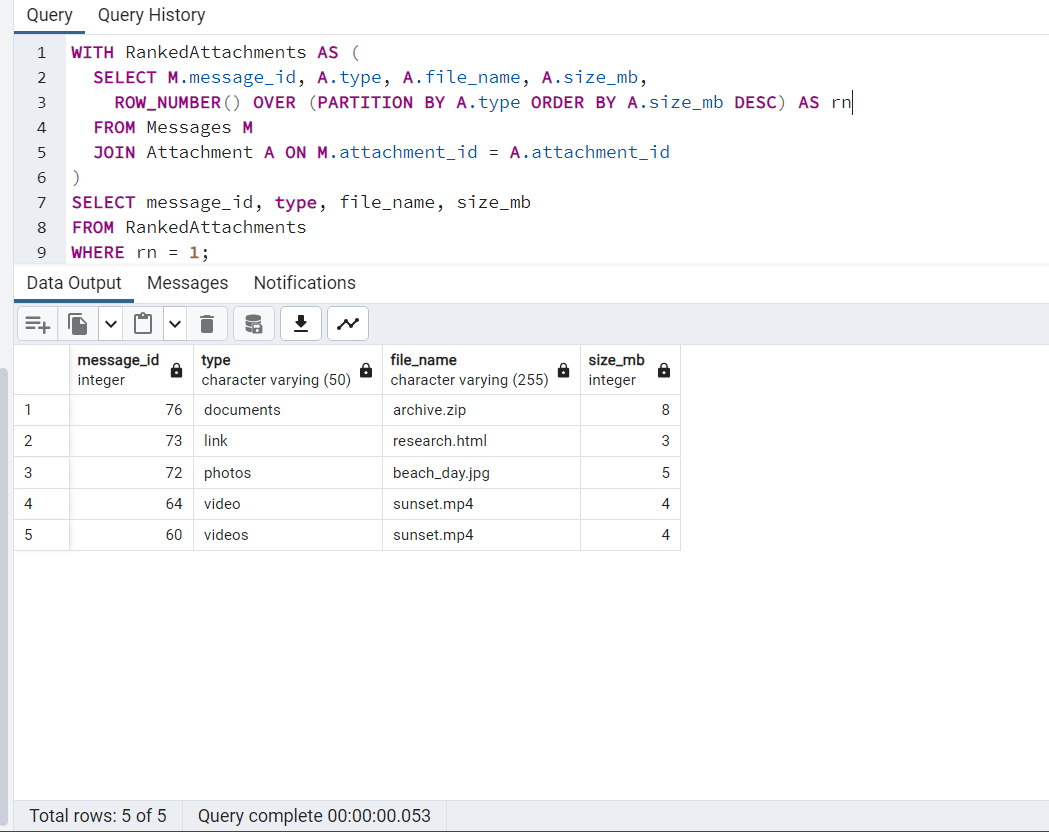
FROM Messages M

JOIN Attachment A ON M.attachment\_id = A.attachment\_id)

SELECT message\_id, type, file\_name, size\_mb FROM

RankedAttachments WHERE rn= 1;

**Count of Tuples**- 5



**English Query 37) Find the groups with the maximum number of members and list their names and member counts.**

**SQL Query** -

SELECT g.group\_name, COUNT(u.user\_id) AS total\_members

FROM Users u

JOIN groupchat g ON u.user\_id = g.user\_id

GROUP BY g.group\_name

HAVING COUNT(u.user\_id) = (

SELECT MAX(group\_size)

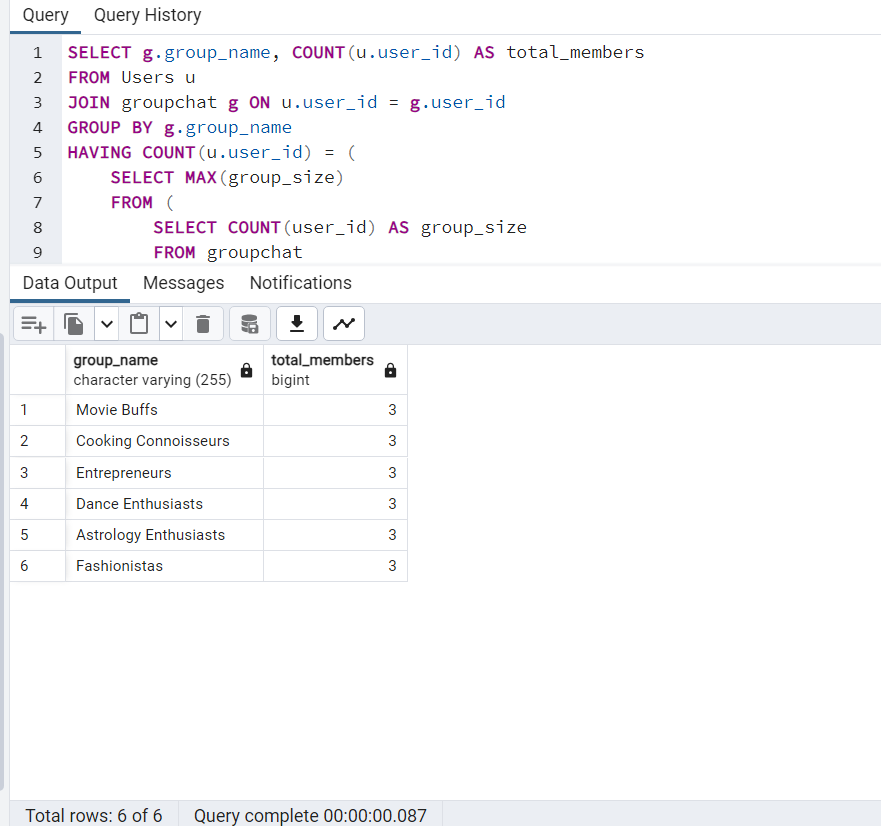
FROM ( SELECT COUNT(user\_id) AS group\_size

FROM groupchat

GROUP BY group\_name

) max\_group\_size);

**Count of Tuples**- 6



**English Query 38) List the users who are members of more than one groupchat.**

**SQL Query** -

SELECT u.first\_name, u.last\_name

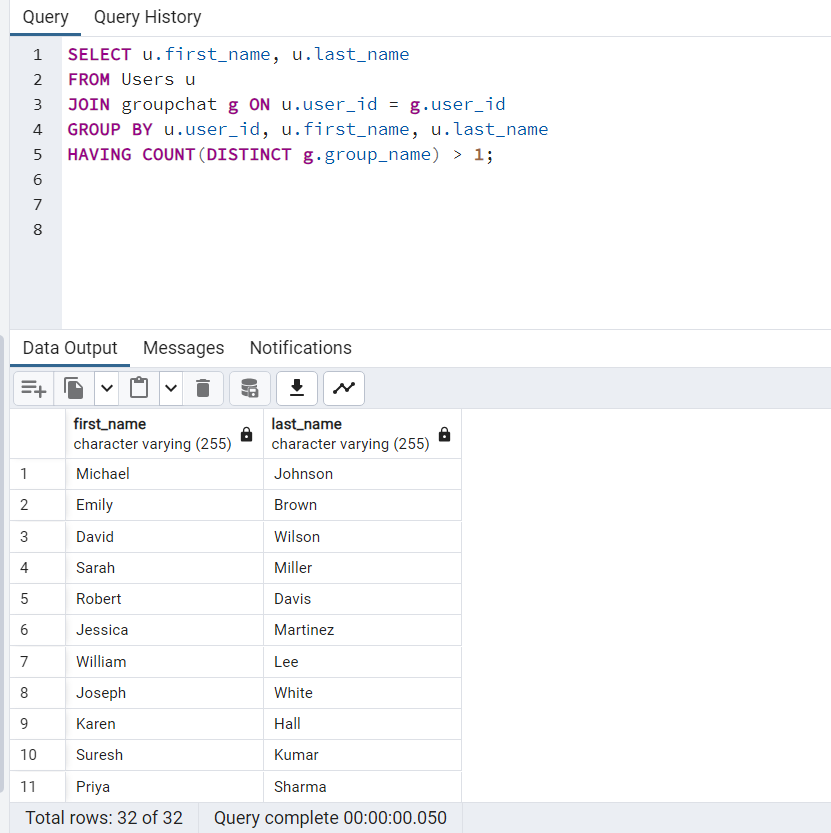
FROM Users u

JOIN groupchat g ON u.user\_id = g.user\_id

GROUP BY u.user\_id, u.first\_name, u.last\_name

HAVING COUNT(DISTINCT g.group\_name) > 1;

**Count of Tuples**- 32



**English Query 39) Write a stored procedure to retrieve the names of all users.**

**SQL Query** -

CREATE OR REPLACE PROCEDURE get\_user\_names()

LANGUAGE 'plpgsql'

AS $$

DECLARE

u\_list record;

BEGIN

FOR u\_list IN (SELECT first\_name, last\_name FROM Users)

LOOP

RAISE INFO 'User Name: % %', u\_list.first\_name, u\_list.last\_name;

END LOOP;

END;

$$;

CALL get\_user\_names();

**Count of Tuples**- 32



**English Query 40) create a stored procedure to update the participant's name based on the chat\_id.**

**SQL Query** -

CREATE OR REPLACE PROCEDURE UpdateParticipantName(chatId INT, newParticipantName VARCHAR(255))

LANGUAGE 'plpgsql' AS $$

BEGIN

UPDATE Chat

SET participant\_name = newParticipantName

WHERE chat\_id = chatId;

END;

$$;

call UpdateParticipantName(7, 'harry' )

**Count of Tuples**- 5

