Advanced Chat Application with Group Chat and Additional Features
You are tasked with designing and querying a highly functional MySQL database schema for a chat application that supports individual and group chats. The application has advanced features such as message reactions, multimedia support, read receipts, chat roles, and permissions for group chats.

Schema Requirements
Part 1: Table Creation

1. **Users Table**
   Stores information about the users of the chat application, including their unique identifier, username, email, status, and profile picture.
2. **Chats Table**
   Represents all chat sessions, whether individual or group chats. Each chat has a unique identifier, a type (individual or group), and a timestamp of its creation.
3. **Chat_Members Table**
   Captures the relationship between users and chats, enabling many-to-many associations. It also stores the role of the user in the chat and whether notifications are muted.
4. **Messages Table**
   Stores messages exchanged within chats. Each message is associated with a chat and a sender and can optionally include media attachments or indicate deletion.
5. **Message_Reactions Table**
   Tracks reactions given to messages, including the type of reaction and the user who gave it.
6. **Groups Table**
   Contains details specific to group chats, such as the group name and creation timestamp. Each group is linked to a corresponding chat.
7. **Permissions Table**
   Defines the permissions associated with user roles (e.g., admin, moderator, member), specifying capabilities such as adding or removing members, sending media, and editing messages.
8. **Read_Receipts Table**
   Tracks when users have read specific messages, providing a timestamp for the read action.

Relationships

1. A user can belong to multiple chats, and a chat can have multiple users (many-to-many).
2. A message belongs to a single chat and is sent by a single user.
3. Group-specific settings and roles are tied to the Groups table and Chat_Members table.
4. Reactions are tied to individual messages and are given by users.
5. Permissions are tied to roles and determine the capabilities of users in group chats.
6. Read receipts track when users have read specific messages.

Data Retrieval: Advanced Queries

1. List all chats (individual and group) a user is part of, showing the user's role, chat type, and whether notifications are muted.
2. Retrieve the last 50 messages in a chat, including the sender's name, timestamp, and reactions. Sort them by sent_at.
3. Fetch all unread messages for a user in group chats where the user is a member.
4. Get a list of group members in a specific group, showing their role and online/offline status.
5. List the most reacted messages in the application, grouped by reaction type.
6. Fetch the total number of media messages sent in a specific group chat over the last 7 days.
7. Find all users who have muted a specific group chat.
8. Retrieve a user's last read message in each of their chats.
9. Fetch all messages in a chat containing a specific keyword, sorted by the number of reactions.
10. List all group chats where the user is an admin and has added at least 5 new members in the last 30 days.

Bonus Questions

1. Implement message threading: Allow users to reply to specific messages. Add a parent_message_id field to the Messages table and write a query to fetch a thread of messages for a given message_id.
2. Optimize the schema for large-scale data with billions of messages. Propose indexing strategies for frequently queried columns such as chat_id, sender_id, and sent_at.
3. Add functionality to pin messages in group chats. Create a Pinned_Messages table and write a query to fetch pinned messages for a specific group.
4. Introduce typing indicators: Track when a user is typing in a chat. Design a Typing_Indicators table and write a query to fetch all users currently typing in a group chat.