Bock University

COSC 3P32

Final Project

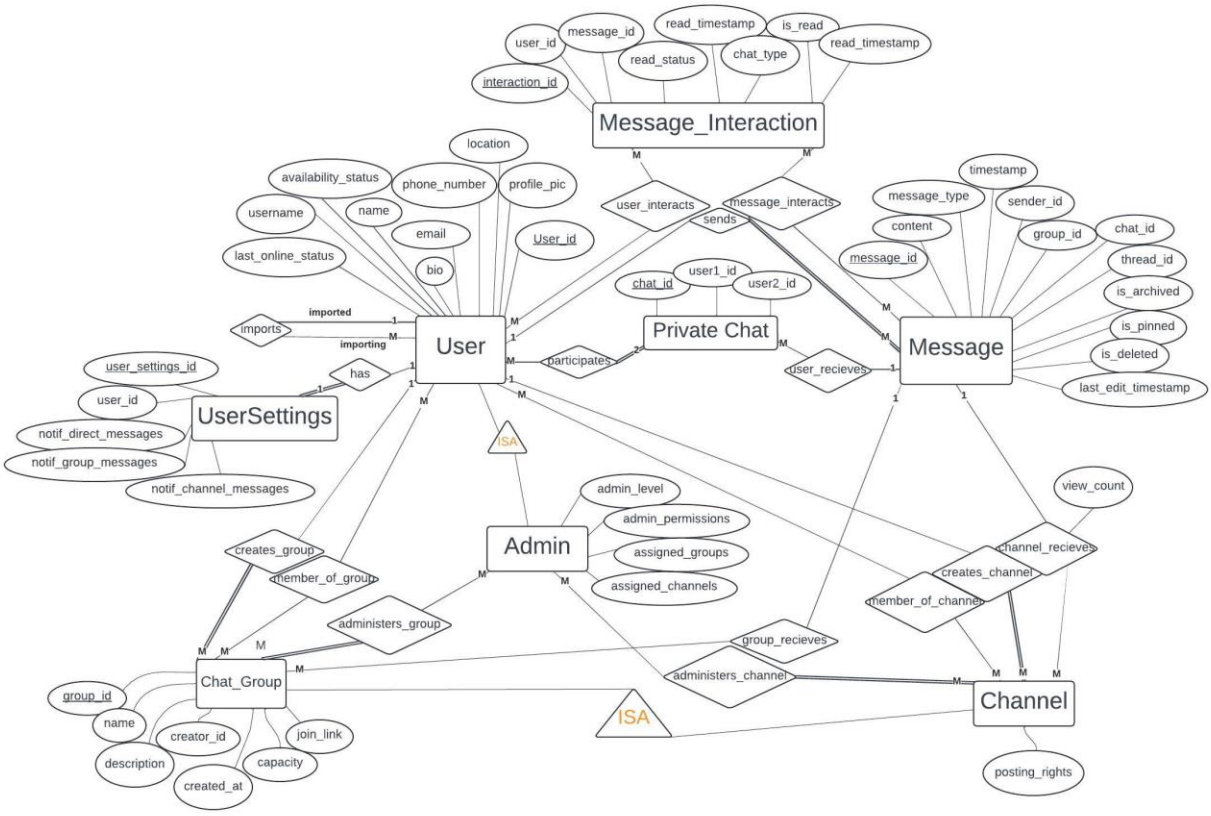
Message Hub

Ameya Chindarkar(solo)

7023609

April 30, 2024

1) Develop an ER model detailing entity sets, relationship sets, keys, and constraints. Explain any necessary constraints that your ER diagram cannot model. (15 pts)



Constraints that can't be modeled:

1. Joining/Creating Groups/Channels

- **Constraint**: Each user is allowed to join or create up to 100 groups. This limit is enforced to prevent excessive strain on the system and to help users manage their group engagements more effectively. The limit includes both the groups created by the user and those they are added to by others.
- **Limitation in ER**: Instead of showing a 1 to 100 relationship in the ERD, a 1 to M relationship is added to retain flexibility.

2. Account Deletion and Retention of Initiated Groups/Channels

- **Constraint**: Users can delete their accounts, removing all personal messages and contacts. However, any groups or channels they started remaining active and accessible to designated administrators.

- **Limitation in ER**: ER diagrams illustrate entity relationships and attributes but cannot model conditional behaviors such as selectively retaining groups/channels while deleting user data. This requires application logic or database procedures, which are beyond the scope of ER diagrams.

3. Dynamic Administrative Role Assignment

- **Constraint**: Group creators can assign and modify administrative roles dynamically based on user activity and group needs.
- **Limitation in ER**: While the ERD can show relationships that allow for the assignment of roles (e.g., an **Administers** relationship between **Admin** and **Chat_Group**), it cannot model the operational logic that allows for dynamic role assignment and modification in real-time.

4. Message Deletion Propagation

- **Constraint**: Deleting a message by one user should result in its removal for all users, effectively making the message disappear from the system entirely.
- **Limitation in ER**: The ER diagram can include an attribute like **is_deleted** in the **Message** table to indicate deletion status, but it cannot model the propagation of this deletion across all users' views, which is handled by application logic.

5. Complex Notification Settings

- **Constraint**: Users can customize notification settings for different types of interactions within private chats, groups, and channels.
- **Limitation in ER**: Although **UserSettings** can store basic notification preferences, the ERD does not capture the complex rules and conditions that govern how notifications are processed and delivered based on user preferences and interaction contexts.

6. Conditional Content Access in Channels

- **Constraint**: Only certain users (creators and admins) have posting privileges in channels, which can change over time based on group policies or user roles.
- **Limitation in ER**: The ERD can show that channels have posting rights linked to user roles, but the conditional logic that enforces these rights based on the evolving roles and policies needs to be handled by application code, not static data modeling.

7. Join Links for Group Access

- **Constraint**: Groups and channels should provide unique join links that facilitate easy access for new users, with links possibly expiring or changing based on administrative settings.
- **Limitation in ER**: While **Chat_Group** can include a **join_link** attribute, the dynamic nature of link validity, expiration, and regeneration based on administrative actions or group settings is a process-oriented feature that ER diagrams do not capture.

2) Based on the ER model, design a relational schema for the database. (10 pts)

Relational Schema:

1. **User** (user_id: *integer*, username: *string*, name: *string*, email: *string*, phone_number: *string*, profile_pic: *string*, location: *string*, bio: *string*, availability_status: *boolean*, last_online_status: timestamp, message_id: *integer*, group_id: *integer*, interaction_id: *integer*)
2. **UserSettings** (user_settings_id: *integer*, user_id: *integer*, notif_direct_messages: *boolean*, notif_group_messages: *boolean*, notif_channel_messages: *boolean*, is_archived: *boolean*)
3. **Message** (message_id: *integer*, content: *string*, timestamp: *timestamp*, message_type: *string*, sender_id: *integer*, group_id: *integer*, -- Nullable is_pinned: *boolean*, is_deleted: *boolean*, last_edit_timestamp: *timestamp*, view_count: *integer*, user_id: *integer*, interaction_id: *integer*,)
4. **Chat_Group** (group_id: *integer*, name: *string*, description: *string*, created_at: *timestamp*, creator_id: *integer*, join_link: *string*, user_id: *integer*, message_id: *integer*)
5. **Channel** (group_id: *integer*, posting rights: *string*, user_id: *integer*, message_id: *integer*)
6. **Admin** (user_id: *integer*, admin_level: *string*, admin_permissions: *string*, assigned_groups: *string*, assigned_channels: *string*)
7. **Message_Interaction** (interaction_id: *integer*, user_id: *integer*, message_id: *integer*, read_status: *boolean*, read_timestamp: *timestamp*, chat_type: *string*, read_status: *boolean*, read_timestamp: *timestamp*, user_id: *integer*, message_id: *integer*)
8. **PrivateChat** (chat_id: *integer*, user1_id: *integer*, user2_id: *integer*)

3) For each relation, identify all functional dependencies that hold on the fields of that table. For each table, specify if that table is in BCNF, 3NF, or neither. If a table is not in BCNF, then attempt to find a BCNF decomposition that is both lossless-join and dependency-preserving. If this is not possible, then a lossless-join, dependency-preserving 3NF decomposition is acceptable. Clearly specify the resulting relational schema. (15 pts):

1. **User Table**
    - **Attributes**: user_id, username, name, email, phone_number, profile_pic, location, bio, availability_status, last_online_status
    - **Functional Dependencies**:
        - **user_id → username, name, email, phone_number, profile_pic, location, bio, availability_status, last_online_status**
    - **Normalization**:
        - **BCNF**: This table is in BCNF, as every determinant (**user_id**) is a superkey.
2. **UserSettings Table**
    - **Attributes**: user_settings_id, user_id, notif_direct_messages, notif_group_messages, notif_channel_messages, is_archived

- **Functional Dependencies**:
    - **user_settings_id → user_id, notif_direct_messages, notif_group_messages, notif_channel_messages, is_archived**
- **Normalization**:
    - **BCNF**: Each attribute is functionally dependent only on the primary key.

3. **Message Table**
    - **Attributes**: message_id, content, timestamp, message_type, sender_id, group_id, is_pinned, is_deleted, last_edit_timestamp, view_count
    - **Functional Dependencies**:
        - **message_id → content, timestamp, message_type, sender_id, group_id, is_pinned, is_deleted, last_edit_timestamp, view_count**
    - **Normalization**:
        - **BCNF**: The determinant (**message_id**) is a superkey, and there are no dependencies between non-prime attributes.
4. **Chat_Group Table**
    - **Attributes**: group_id, name, description, created_at, creator_id, join_link
    - **Functional Dependencies**:
        - **group_id → name, description, created_at, creator_id, join_link**
    - **Normalization**:
        - **BCNF**: All attributes are functionally dependent on **group_id**.
5. **Channel Table**
    - **Attributes**: group_id, posting_rights, capacity
    - **Functional Dependencies**:
        - **group_id → posting_rights, capacity**
    - **Normalization**:
        - **BCNF**: Inherits **group_id** as a primary key from **Group** and includes additional attributes unique to **Channel**.
6. **Admin Table**
    - **Attributes**: user_id, admin_level, admin_permissions, assigned_groups, assigned_channels
    - **Functional Dependencies**:

        - **user_id → admin_level, admin_permissions, assigned_groups, assigned_channels**
    - **Normalization**:
        - **BCNF**: **user_id** is the primary key, and all non-key attributes are functionally dependent on it.

7. **Message_Interaction Table**
    - **Attributes**: interaction_id, user_id, message_id, read_status, read_timestamp, chat_type
    - **Functional Dependencies**:
        - **interaction_id → user_id, message_id, read_status, read_timestamp, chat_type**
    - **Normalization**:
        - **BCNF**: **interaction_id** is the primary key, with no partial dependencies or transitive dependencies.
8. **PrivateChat Table**
    - **Attributes**: chat_id, user1_id, user2_id
    - **Functional Dependencies**:
        - **chat_id → user1_id, user2_id**
        - The **chat_id** uniquely determines both **user1_id** and **user2_id**, encapsulating all the information specific to a single private chat instance.
    - **Normalization**:
        - **BCNF**: The **chat_id** is the primary key, with no partial dependencies or transitive dependencies.

4. Write the SQL statements necessary to create the tables for the above database, capturing as many constraints as possible. (15 pts)

a. You must ensure that all domain constraints, primary key constraints, and foreign key constraints are enforced.

b. Your project must be able to support insertions, deletions, and updates for all data stored in the database while also ensuring that constraints are satisfied.

c. Input some data into your tables. You should input enough data that it is possible to verify any query you might have and also any constraints that must hold.

i. The Users table (and possibly Groups, etc.) should contain your own information (your name, email, etc.) as well as some other random info.

Creating Tables:

```sql
CREATE TABLE User (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    name VARCHAR(255),
    email VARCHAR(255) UNIQUE NOT NULL,
    phone_number VARCHAR(15),
    profile_pic VARCHAR(255),
    location VARCHAR(255),
    bio TEXT,
    availability_status BOOLEAN DEFAULT TRUE,
    last_online_status TIMESTAMP
);
```

```sql
CREATE TABLE UserSettings (
    user_settings_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    notif_direct_messages BOOLEAN DEFAULT TRUE,
    notif_group_messages BOOLEAN DEFAULT TRUE,
    notif_channel_messages BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

```sql
CREATE TABLE Message (
    message_id INTEGER PRIMARY KEY,
    content TEXT NOT NULL,
    message_type VARCHAR(50),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    sender_id INTEGER,
    group_id INTEGER,
    private_chat_id INTEGER,
    is_archived BOOLEAN DEFAULT FALSE,
    is_pinned BOOLEAN DEFAULT FALSE,
    is_deleted BOOLEAN DEFAULT FALSE,
    last_edit_timestamp TIMESTAMP,
    view_count INTEGER DEFAULT 0,
    FOREIGN KEY (sender_id) REFERENCES User(user_id),
    FOREIGN KEY (group_id) REFERENCES ChatGroup(group_id),
    FOREIGN KEY (private_chat_id) REFERENCES PrivateChat(private_chat_id)
);
```

```sql
CREATE TABLE Chat_Group (
    group_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    creator_id INT,
    join_link VARCHAR(255),
    FOREIGN KEY (creator_id) REFERENCES User(user_id)
);
```

```sql
CREATE TABLE Channel (
    group_id INT PRIMARY KEY,
    posting_rights VARCHAR(255) NOT NULL,
    FOREIGN KEY (group_id) REFERENCES Chat_Group(group_id)
);
```

```sql
CREATE TABLE Admin (
    user_id INT PRIMARY KEY,
    admin_level VARCHAR(50),
    admin_permissions TEXT,
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

```sql
CREATE TABLE PrivateChat (
    private_chat_id INTEGER PRIMARY KEY,
    user1_id INTEGER,
    user2_id INTEGER,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user1_id) REFERENCES User(user_id),
    FOREIGN KEY (user2_id) REFERENCES User(user_id),
    CHECK (user1_id != user2_id)
);
```

```sql
CREATE TABLE Imports (
    user_id INT,
    importing_user_id INT,
    PRIMARY KEY (user_id, importing_user_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (importing_user_id) REFERENCES User(user_id)
);
```

```sql
CREATE TABLE Message_Interacts (
    message_id INT,
    interaction_id INT,
    PRIMARY KEY (message_id, interaction_id),
    FOREIGN KEY (message_id) REFERENCES Message(message_id),
    FOREIGN KEY (interaction_id) REFERENCES Message_Interaction(interaction_id)
);
```

```sql
CREATE TABLE Has (
    user_id INT,
    user_settings_id INT,
    PRIMARY KEY (user_id, user_settings_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (user_settings_id) REFERENCES UserSettings(user_settings_id)
);
```

```sql
CREATE TABLE Participates (
    user_id INTEGER,
    private_chat_id INTEGER,
    PRIMARY KEY (user_id, private_chat_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (private_chat_id) REFERENCES PrivateChat(private_chat_id)
);
```

```sql
CREATE TABLE Sends (
    user_id INT,
    message_id INT,
    PRIMARY KEY (user_id, message_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (message_id) REFERENCES Message(message_id)
);
```

```sql
CREATE TABLE User_Receives (
    user_id INT,
    message_id INT,
    PRIMARY KEY (user_id, message_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (message_id) REFERENCES Message(message_id)
);
```

```sql
CREATE TABLE Group_Receives (
    group_id INT,
    message_id INT,
    PRIMARY KEY (group_id, message_id),
    FOREIGN KEY (group_id) REFERENCES Group(group_id),
    FOREIGN KEY (message_id) REFERENCES Message(message_id)
);
```

```sql
CREATE TABLE Channel_Receives (
    channel_id INT,
    message_id INT,
    PRIMARY KEY (channel_id, message_id),
    FOREIGN KEY (channel_id) REFERENCES Channel(group_id),
    FOREIGN KEY (message_id) REFERENCES Message(message_id)
);
```

```sql
CREATE TABLE Creates_Group (
    user_id INT,
    group_id INT,
    PRIMARY KEY (user_id, group_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (group_id) REFERENCES Chat_Group(group_id)
);
```

```
CREATE TABLE Creates_Channel (
    user_id INT,
    channel_id INT,
    PRIMARY KEY (user_id, channel_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (channel_id) REFERENCES Channel(group_id)
);
```

```
CREATE TABLE Member_Of_Group (
    user_id INT,
    group_id INT,
    PRIMARY KEY (user_id, group_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (group_id) REFERENCES Group(group_id)
);
```

```
CREATE TABLE Member_Of_Channel (
    user_id INT,
    channel_id INT,
    PRIMARY KEY (user_id, channel_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (channel_id) REFERENCES Channel(group_id)
);
```

```
CREATE TABLE Administers_Group (
    admin_id INT,
    group_id INT,
    PRIMARY KEY (admin_id, group_id),
    FOREIGN KEY (admin_id) REFERENCES Admin(user_id),
    FOREIGN KEY (group_id) REFERENCES Chat_Group(group_id)
);
```

```
CREATE TABLE Administers_Channel (
    admin_id INT,
    channel_id INT,
    PRIMARY KEY (admin_id, channel_id),
    FOREIGN KEY (admin_id) REFERENCES Admin(user_id),
    FOREIGN KEY (channel_id) REFERENCES Channel(group_id)
);
```

```
CREATE TABLE User_Interacts (
    user_id INT,
    interaction_id INT,
    PRIMARY KEY (user_id, interaction_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (interaction_id) REFERENCES Message_Interaction(interaction_id)
);
```

```
CREATE TABLE Message_Interacts (
    message_id INT,
    interaction_id INT,
    PRIMARY KEY (message_id, interaction_id),
    FOREIGN KEY (message_id) REFERENCES Message(message_id),
    FOREIGN KEY (interaction_id) REFERENCES Message_Interaction(interaction_id)
);
```

Resulting Tables:

Available Tables

Admin

user_id	admin_level	admin_permissions
empty		

Administers_Channel

admin_id	channel_id
empty	

Administers_Group

admin_id	group_id
empty	

Channel

group_id	posting_rights
empty	

Channel_Receives

channel_id	message_id
empty	

Chat_Group

group_id	name	description	created_at	creator_id	join_link
empty					

Creates_Channel

user_id	channel_id
empty	

Creates_Group

user_id	group_id
empty	

Group_Receives

group_id	message_id
empty	

Has

user_id	user_settings_id
empty	

Imports

user_id	importing_user_id
empty	

Member_Of_Channel

user_id	channel_id
empty	

Message

message_id	content	message_type	timestamp	sender_id	group_id	private_chat_id	is_archived	is_pinned	is_deleted	last_edit_timestamp	view_count
empty											

Message_Interacts

message_id	interaction_id
empty	

MessageInteraction

interaction_id	message_id	user_id	private_chat_id	read_status	read_timestamp
empty					

Participates

Participates

user_id	private_chat_id
empty	

PrivateChat

private_chat_id	user1_id	user2_id	created_at
empty			

Sends

user_id	message_id
empty	

User

user_id	username	name	email	phone_number	profile_pic	location	bio	availability_status	last_online_status
empty									

User_Interacts

user_id	interaction_id
empty	

User_Receives

user_id	message_id
empty	

UserSettings

user_settings_id	user_id	notif_direct_messages	notif_group_messages	notif_channel_messages
empty				

Inserting Data into Each Table:

User Table

```
INSERT INTO User (user_id, username, name, email, phone_number, profile_pic, location, bio, availability_status, last_online_status)
VALUES
(1, 'ameya_chindarkar', 'Ameya Chindarkar', 'ac20@brocku.ca', '100-100', 'ameyas_pic.jpg', 'St. Catharines', 'I am a Student at Brock University.',
TRUE, '2023-04-01 06:24:10'),
(2, 'alice_blue', 'Alice Blue', 'alice.blue@gmail.com', '555-1001', 'alice_pic.jpg', 'Hamilton', 'Loves hiking and photography.', TRUE, '2023-04-01
08:00:00'),
(3, 'bob_green', 'Bob Green', 'bob.green@hotmail.com', '555-1002', 'bob_pic.jpg', 'Montreal', 'Guitarist and tech enthusiast.', FALSE, '2023-04-01
09:00:00'),
(4, 'carol_red', 'Carol Red', 'carol.red@yahoo.com', '555-1003', 'carol_pic.jpg', 'Vancouver', 'Startup founder and speaker.', TRUE, '2023-04-01
10:00:00'),
(5, 'dave_black', 'Dave Black', 'dave.black@gmail.com', '555-1004', 'dave_pic.jpg', 'Calgary', 'Writer and editor.', TRUE, '2023-04-01 11:00:00'),
(6, 'eve_white', 'Eve White', 'eve.white@hotmail.com', '555-1005', 'eve_pic.jpg', 'Winnipeg', 'Data scientist and ML enthusiast.', FALSE, '2023-04-01
12:00:00'),
(7, 'frank_gray', 'Frank Gray', 'frank.gray@yahoo.com', '555-1006', 'frank_pic.jpg', 'Ottawa', 'Beach lover and real estate mogul.', TRUE, '2023-04-01
13:00:00');
```

User

user_id	username	name	email	phone_number	profile_pic	location	bio	availability_status	last_online_status
1	ameya_chindarkar	Ameya Chindarkar	ac20@brocku.ca	100-100	ameyas_pic.jpg	St. Catharines	I am a Student at Brock University.	1	2023-04-01 06:24:10
2	alice_blue	Alice Blue	alice.blue@gmail.com	555-1001	alice_pic.jpg	Hamilton	Loves hiking and photography.	1	2023-04-01 08:00:00
3	bob_green	Bob Green	bob.green@hotmail.com	555-1002	bob_pic.jpg	Montreal	Guitarist and tech enthusiast.	0	2023-04-01 09:00:00
4	carol_red	Carol Red	carol.red@yahoo.com	555-1003	carol_pic.jpg	Vancouver	Startup founder and speaker.	1	2023-04-01 10:00:00
5	dave_black	Dave Black	dave.black@gmail.com	555-1004	dave_pic.jpg	Calgary	Writer and editor.	1	2023-04-01 11:00:00
6	eve_white	Eve White	eve.white@hotmail.com	555-1005	eve_pic.jpg	Winnipeg	Data scientist and ML enthusiast.	0	2023-04-01 12:00:00
7	frank_gray	Frank Gray	frank.gray@yahoo.com	555-1006	frank_pic.jpg	Ottawa	Beach lover and real estate mogul.	1	2023-04-01 13:00:00

Admin Table

```
INSERT INTO Admin (user_id, admin_level, admin_permissions) VALUES
(1, 'Super', 'All'),
(2, 'Moderator', 'Create, Delete'),
(3, 'Moderator', 'Create, Update'),
(4, 'Viewer', 'Read');
```

Admin

user_id	admin_level	admin_permissions
1	Super	All
2	Moderator	Create, Delete
3	Moderator	Create, Update
4	Viewer	Read

PrivateChat Table

```
INSERT INTO PrivateChat (private_chat_id, user1_id, user2_id, created_at)
VALUES
(16, 1, 2, '2023-04-01 15:00:00'),
(17, 1, 3, '2023-04-01 16:00:00'),
(18, 2, 3, '2023-04-01 17:00:00'),
(19, 2, 4, '2023-04-01 18:00:00'),
(20, 3, 4, '2023-04-01 19:00:00'),
(21, 3, 5, '2023-04-01 20:00:00'),
(22, 4, 5, '2023-04-01 21:00:00'),
(23, 4, 6, '2023-04-01 22:00:00'),
(24, 5, 6, '2023-04-01 23:00:00'),
(25, 5, 7, '2023-04-02 00:00:00'),
(26, 6, 7, '2023-04-02 01:00:00'),
(27, 1, 4, '2023-04-02 02:00:00'),
(28, 2, 5, '2023-04-02 03:00:00'),
(29, 3, 6, '2023-04-02 04:00:00'),
(30, 4, 7, '2023-04-02 05:00:00');
```

PrivateChat

private_chat_id	user1_id	user2_id	created_at
16	1	2	2023-04-01 15:00:00
17	1	3	2023-04-01 16:00:00
18	2	3	2023-04-01 17:00:00
19	2	4	2023-04-01 18:00:00
20	3	4	2023-04-01 19:00:00
21	3	5	2023-04-01 20:00:00
22	4	5	2023-04-01 21:00:00
23	4	6	2023-04-01 22:00:00
24	5	6	2023-04-01 23:00:00
25	5	7	2023-04-02 00:00:00
26	6	7	2023-04-02 01:00:00
27	1	4	2023-04-02 02:00:00
28	2	5	2023-04-02 03:00:00
29	3	6	2023-04-02 04:00:00
30	4	7	2023-04-02 05:00:00

Group Table (Chat_Group)

```
INSERT INTO Chat_Group (group_id, name, description, created_at, creator_id, join_link) VALUES
(1, 'Project Team', 'Discussion group for project team members.', '2023-04-01', 1, 'http://joinlink.com/team'),
(2, 'Hiking Club', 'Group for discussing hiking plans and sharing photos.', '2023-04-02', 2, 'http://joinlink.com/hiking'),
(3, 'Tech Talks', 'Group for sharing and discussing latest technology trends.', '2023-04-03', 3, 'http://joinlink.com/tech'),
(4, 'Book Lovers', 'Book reading club for sharing and discussing literature.', '2023-04-04', 4, 'http://joinlink.com/books'),
(5, 'Music Fans', 'Group for discussing all things music and sharing tracks.', '2023-04-05', 5, 'http://joinlink.com/music');
```

Chat_Group

group_id	name	description	created_at	creator_id	join_link
1	Project Team	Discussion group for project team members.	2023-04-01	1	http://joinlink.com/team
2	Hiking Club	Group for discussing hiking plans and sharing photos.	2023-04-02	2	http://joinlink.com/hiking
3	Tech Talks	Group for sharing and discussing latest technology trends.	2023-04-03	3	http://joinlink.com/tech
4	Book Lovers	Book reading club for sharing and discussing literature.	2023-04-04	4	http://joinlink.com/books
5	Music Fans	Group for discussing all things music and sharing tracks.	2023-04-05	5	http://joinlink.com/music

Channel Table

```
INSERT INTO Channel (group_id, posting_rights)
VALUES
(1, 'AdminsOnly'),
(2, 'AdminsOnly'),
(5, 'AdminsOnly');
```

Channel

group_id	posting_rights
1	AdminsOnly
2	AdminsOnly
5	AdminsOnly

Message Table

```
INSERT INTO Message (message_id, content, message_type, timestamp, sender_id, group_id, private_chat_id,
is_archived, is_pinned, is_deleted, last_edit_timestamp, view_count)
VALUES
(1, 'Hey everyone, welcome to the project team!', 'text', '2023-04-01 10:00:00', 1, 1, NULL, FALSE, FALSE,
FALSE, NULL, 0),
(2, 'Looking forward to our hike next week!', 'text', '2023-04-01 11:00:00', 2, 2, NULL, FALSE, FALSE, FALSE,
NULL, 0),
(3, 'New tech release on Friday at noon!', 'text', '2023-04-01 12:00:00', 3, 3, NULL, FALSE, FALSE, FALSE, NULL,
0),
(4, 'Has anyone read the latest by Stephen King?', 'text', '2023-04-01 13:00:00', 4, 4, NULL, FALSE, FALSE,
FALSE, NULL, 0),
(5, 'This new album is a hit!', 'text', '2023-04-01 14:00:00', 5, 5, NULL, FALSE, FALSE, FALSE, NULL, 0),
(6, 'Can we discuss the project timeline?', 'text', '2023-04-02 15:00:00', 1, 1, NULL, FALSE, TRUE, FALSE, NULL,
0),
(7, 'Reminder about the safety gear for hiking.', 'text', '2023-04-02 16:00:00', 2, 2, NULL, FALSE, FALSE,
FALSE, NULL, 0),
(8, 'Who is attending the tech webinar?', 'text', '2023-04-02 17:00:00', 3, 3, NULL, FALSE, FALSE, FALSE, NULL,
0),
(9, 'Book club meeting rescheduled to next Friday.', 'text', '2023-04-02 18:00:00', 4, 4, NULL, FALSE, FALSE,
FALSE, NULL, 0),
(10, 'Anyone going to the live concert this weekend?', 'text', '2023-04-02 19:00:00', 5, 5, NULL, FALSE, FALSE,
FALSE, NULL, 0);
```

Message

message_id	content	message_type	timestamp	sender_id	group_id	private_chat_id	is_archived	is_pinned	is_deleted	last_edit_timestamp	view_count
1	Hey everyone, welcome to the project team!	text	2023-04-01 10:00:00	1	1		0	0	0		0
2	Looking forward to our hike next week!	text	2023-04-01 11:00:00	2	2		0	0	0		0
3	New tech release on Friday at noon!	text	2023-04-01 12:00:00	3	3		0	0	0		0
4	Has anyone read the latest by Stephen King?	text	2023-04-01 13:00:00	4	4		0	0	0		0
5	This new album is a hit!	text	2023-04-01 14:00:00	5	5		0	0	0		0
6	Can we discuss the project timeline?	text	2023-04-02 15:00:00	1	1		0	1	0		0
7	Reminder about the safety gear for hiking.	text	2023-04-02 16:00:00	2	2		0	0	0		0
8	Who is attending the tech webinar?	text	2023-04-02 17:00:00	3	3		0	0	0		0
9	Book club meeting rescheduled to next Friday.	text	2023-04-02 18:00:00	4	4		0	0	0		0
10	Anyone going to the live concert this weekend?	text	2023-04-02 19:00:00	5	5		0	0	0		0

Message_Interacts Table

```
INSERT INTO Message_Interacts (message_id, interaction_id) VALUES
(2,1),
(3,2),
(5,3),
(7,4),
(10,5);
```

Message_Interacts

message_id	interaction_id
2	1
3	2
5	3
7	4
10	5

UserSettings Table

```
INSERT INTO UserSettings (user_settings_id, user_id, notif_direct_messages, notif_group_messages,
notif_channel_messages)
VALUES
(1, 1, TRUE, TRUE, TRUE),
(2, 2, TRUE, FALSE, TRUE),
(3, 3, FALSE, TRUE, FALSE),
(4, 4, TRUE, TRUE, TRUE),
(5, 5, FALSE, TRUE, TRUE),
(6, 6, TRUE, FALSE, FALSE),
(7, 7, TRUE, TRUE, TRUE);
```

UserSettings

user_settings_id	user_id	notif_direct_messages	notif_group_messages	notif_channel_messages
1	1	1	1	1
2	2	1	0	1
3	3	0	1	0
4	4	1	1	1
5	5	0	1	1
6	6	1	0	0
7	7	1	1	1

Administers_Group Table

```
INSERT INTO Administers_Group (admin_id, group_id)
VALUES
(1, 1),
(2, 2),
(3, 3);
```

Administers_Group

admin_id	group_id
1	1
2	2
3	3

Administers_Channel Table

```
INSERT INTO Administers_Channel (admin_id, channel_id)
VALUES

(2, 2);
```

Administers_Channel

admin_id	channel_id
2	2

Channel_Recieves Table

```
INSERT INTO Channel_Receives (channel_id, message_id) VALUES
(1, 1),
(2, 2),
(5, 5);
```

Channel_Receives

channel_id	message_id
1	1
2	2
5	5

Creates_Channel Table

```
INSERT INTO Creates_Channel (user_id, channel_id) VALUES
(3, 1),
(5, 2),
(7, 5);
```

Creates_Channel

user_id	channel_id
3	1
5	2
7	5

Creates_Group Table

```
INSERT INTO Creates_Group (user_id, group_id) VALUES
(1, 2),
(2, 2),
(6, 4);
```

Creates_Group

user_id	group_id
1	2
2	2
6	4

Group_Recieves Table

```
INSERT INTO Group_Receives (group_id, message_id) VALUES
(1, 6),
(2, 7),
(1, 8),
(2, 9),
(3, 10);
```

Group_Receives

group_id	message_id
1	6
2	7
1	8
2	9
3	10

Has Table

```
INSERT INTO has (user_id, user_settings_id) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7);
```

Has

user_id	user_settings_id
1	1
2	2
3	3
4	4
5	5
6	6
7	7

Imports Table

```
INSERT INTO imports (user_id, importing_user_id) VALUES
(1, 2),
(1, 3),
(1, 4),
(3, 5),
(4, 6),
(1, 7),
(2, 1),
(2, 4),
(5, 7),
(6, 1),
(7, 5),
(6, 2);
```

Imports

user_id	importing_user_id
1	2
1	3
1	4
3	5
4	6
1	7
2	1
2	4
5	7
6	1
7	5
6	2

Member_Of_Group Table

```
INSERT INTO Member_Of_Group (user_id, group_id)
VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 1),
(7, 1);
```

Member_Of_Group

user_id	group_id	joined_at
1	1	2024-05-01 00:07:19
2	2	2024-05-01 00:07:19
3	3	2024-05-01 00:07:19
4	4	2024-05-01 00:07:19
5	5	2024-05-01 00:07:19
6	1	2024-05-01 00:07:19
7	1	2024-05-01 00:07:19

Member_Of_Channel Table

```
INSERT INTO Member_Of_Channel (user_id, channel_id)
VALUES
(2, 1),
(3, 2),
(6, 5);
```

Member_Of_Channel

user_id	channel_id
2	1
3	2
6	5

MessageInteraction Table

```
INSERT INTO MessageInteraction (interaction_id, user_id, message_id, read_status, read_timestamp) VALUES
(1, 1, 1, TRUE, '2023-04-01 09:00:00'),
(2, 2, 2, TRUE, '2023-04-02 10:00:00'),
(3, 3, 3, FALSE, '2023-04-03 11:00:00'),
(4, 4, 4, TRUE, '2023-04-04 12:00:00'),
(5, 5, 5, FALSE, '2023-04-05 13:00:00');
```

MessageInteraction

interaction_id	message_id	user_id	private_chat_id	read_status	read_timestamp
1	1	1		1	2023-04-01 09:00:00
2	2	2		1	2023-04-02 10:00:00
3	3	3		0	2023-04-03 11:00:00
4	4	4		1	2023-04-04 12:00:00
5	5	5		0	2023-04-05 13:00:00

Participates Table

```
INSERT INTO participates (user_id, private_chat_id) VALUES
(1,16),
(2,16),
(1,17),
(3,17),
(3,20),
(4,20),
(4,22),
(5,22),
(4,30),
(7,30);
```

Participates

user_id	private_chat_id
1	16
2	16
1	17
3	17
3	20
4	20
4	22
5	22
4	30
7	30

Sends Table

```
INSERT INTO Sends (user_id, message_id) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);
```

Sends

user_id	message_id
1	1
2	2
3	3
4	4
5	5

User_Interacts Table

```
INSERT INTO User_Interacts (user_id, interaction_id) VALUES
(1,1),
(2,2),
(4,3),
(6,4),
(7,5);
```

User_Interacts

user_id	interaction_id
1	1
2	2
4	3
6	4
7	5

User_Recieves Table

```
INSERT INTO User_Receives (user_id, message_id) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);
```

User_Receives

user_id	message_id
1	1
2	2
3	3
4	4
5	5

5. The project must support the following queries: (45 pts)

a. Retrieve the list of all users.

```
Select * from User;
```

Output

user_id	username	name	email	phone_number	profile_pic	location	bio	availability_status	last_online_status
1	ameya_chindarkar	Ameya Chindarkar	ac20@brocku.ca	100-100	ameyas_pic.jpg	St. Catharines	I am a Student at Brock University.	1	2023-04-01 06:24:10
2	alice_blue	Alice Blue	alice.blue@gmail.com	555-1001	alice_pic.jpg	Hamilton	Loves hiking and photography.	1	2023-04-01 08:00:00
3	bob_green	Bob Green	bob.green@hotmail.com	555-1002	bob_pic.jpg	Montreal	Guitarist and tech enthusiast.	0	2023-04-01 09:00:00
4	carol_red	Carol Red	carol.red@yahoo.com	555-1003	carol_pic.jpg	Vancouver	Startup founder and speaker.	1	2023-04-01 10:00:00
5	dave_black	Dave Black	dave.black@gmail.com	555-1004	dave_pic.jpg	Calgary	Writer and editor.	1	2023-04-01 11:00:00
6	eve_white	Eve White	eve.white@hotmail.com	555-1005	eve_pic.jpg	Winnipeg	Data scientist and ML enthusiast.	0	2023-04-01 12:00:00
7	frank_gray	Frank Gray	frank.gray@yahoo.com	555-1006	frank_pic.jpg	Ottawa	Beach lover and real estate mogul.	1	2023-04-01 13:00:00

b. Retrieve the list of all online users.

```
Select * from User
where availability_status = 1;
```

Output

user_id	username	name	email	phone_number	profile_pic	location	bio	availability_status	last_online_status
1	ameya_chindarkar	Ameya Chindarkar	ac20@brocku.ca	100-100	ameyas_pic.jpg	St. Catharines	I am a Student at Brock University.	1	2023-04-01 06:24:10
2	alice_blue	Alice Blue	alice.blue@gmail.com	555-1001	alice_pic.jpg	Hamilton	Loves hiking and photography.	1	2023-04-01 08:00:00
4	carol_red	Carol Red	carol.red@yahoo.com	555-1003	carol_pic.jpg	Vancouver	Startup founder and speaker.	1	2023-04-01 10:00:00
5	dave_black	Dave Black	dave.black@gmail.com	555-1004	dave_pic.jpg	Calgary	Writer and editor.	1	2023-04-01 11:00:00
7	frank_gray	Frank Gray	frank.gray@yahoo.com	555-1006	frank_pic.jpg	Ottawa	Beach lover and real estate mogul.	1	2023-04-01 13:00:00

c. Given a user (by phone number or unique ID or username), retrieve all information of the user.

Input

Run SQL

```
Select * from User
where phone_number = '555-1001'
;
```

Output

user_id	username	name	email	phone_number	profile_pic	location	bio	availability_status	last_online_status
2	alice_blue	Alice Blue	alice.blue@gmail.com	555-1001	alice_pic.jpg	Hamilton	Loves hiking and photography.	1	2023-04-01 08:00:00

d. Given a user (by phone number, unique ID or username) retrieve all his/her chats (private chats, normal groups and channels)

```
SELECT 'Private Chat' AS chat_type, pc.private_chat_id AS chat_id
FROM PrivateChat AS pc
JOIN User AS u1 ON pc.user1_id = u1.user_id
JOIN User AS u2 ON pc.user2_id = u2.user_id
WHERE u1.username = 'alice_blue'

UNION

SELECT 'Group Chat' AS chat_type, cg.group_id AS chat_id
FROM Chat_Group AS cg
JOIN Member_Of_Group AS mog ON cg.group_id = mog.group_id
JOIN User AS u ON mog.user_id = u.user_id
WHERE u.username = 'alice_blue'

UNION

SELECT 'Channel' AS chat_type, ch.group_id AS chat_id
FROM Channel AS ch
JOIN Member_Of_Channel AS moc ON ch.group_id = moc.channel_id
JOIN User AS u ON moc.user_id = u.user_id
WHERE u.username = 'alice_blue';
```

Output

chat_type	chat_id
Channel	1
Group Chat	2
Private Chat	18
Private Chat	19
Private Chat	28

e. For a given chat, retrieve its metadata (chat title, bio, join link (if applicable), etc.)

```
SELECT 'Private Chat' AS chat_type, pc.private_chat_id AS chat_id,
        u1.username || ' and ' || u2.username AS chat_title,
       'Private conversation between ' || u1.username || ' and ' || u2.username AS bio,
       NULL AS join_link,
       pc.created_at
FROM PrivateChat AS pc
JOIN User AS u1 ON pc.user1_id = u1.user_id
JOIN User AS u2 ON pc.user2_id = u2.user_id
WHERE pc.private_chat_id = 1

UNION

SELECT 'Group Chat' AS chat_type, cg.group_id AS chat_id,
       cg.name AS chat_title,
       cg.description AS bio,
       cg.join_link,
       cg.created_at
FROM Chat_Group AS cg
WHERE cg.group_id = 1
```

Output

chat_type	chat_id	chat_title	bio	join_link	created_at
Group Chat	1	Project Team	Discussion group for project team members.	http://joinlink.com/team	2023-04-01

f. For a given chat, retrieve all its users.

```
SELECT 'Private Chat' AS chat_type, pc.private_chat_id AS chat_id, u.user_id, u.username
FROM PrivateChat AS pc
JOIN User AS u ON u.user_id = pc.user1_id OR u.user_id = pc.user2_id
WHERE pc.private_chat_id = 16

UNION

-- Retrieving users from a specific Group Chat
SELECT 'Group Chat' AS chat_type, cg.group_id AS chat_id, u.user_id, u.username
FROM Chat_Group AS cg
JOIN Member_Of_Group AS mog ON mog.group_id = cg.group_id
JOIN User AS u ON u.user_id = mog.user_id
WHERE cg.group_id = 2

UNION

-- Retrieving users from a specific Channel
SELECT 'Channel' AS chat_type, ch.group_id AS chat_id, u.user_id, u.username
FROM Channel AS ch
JOIN Member_Of_Channel AS moc ON moc.channel_id = ch.group_id
JOIN User AS u ON u.user_id = moc.user_id
WHERE ch.group_id = 5;
```

Output

chat_type	chat_id	user_id	username
Channel	5	6	eve_white
Group Chat	2	2	alice_blue
Private Chat	16	1	ameya_chindarkar
Private Chat	16	2	alice_blue

g. For a given chat, retrieve all its online users.

```
SELECT 'Private Chat' AS chat_type, pc.private_chat_id AS chat_id, u.user_id, u.username
FROM PrivateChat AS pc
JOIN User AS u ON (u.user_id = pc.user1_id OR u.user_id = pc.user2_id)
WHERE pc.private_chat_id = 16 AND u.availability_status = TRUE

UNION

SELECT 'Group Chat' AS chat_type, cg.group_id AS chat_id, u.user_id, u.username
FROM Chat_Group AS cg
JOIN Member_Of_Group AS mog ON mog.group_id = cg.group_id
JOIN User AS u ON u.user_id = mog.user_id
WHERE cg.group_id = 2 AND u.availability_status = TRUE

UNION

SELECT 'Channel' AS chat_type, ch.group_id AS chat_id, u.user_id, u.username
FROM Channel AS ch
JOIN Member_Of_Channel AS moc ON moc.channel_id = ch.group_id
JOIN User AS u ON u.user_id = moc.user_id
WHERE ch.group_id = 5 AND u.availability_status = TRUE;
```

Output

chat_type	chat_id	user_id	username
Group Chat	2	2	alice_blue
Private Chat	16	1	ameya_chindarkar
Private Chat	16	2	alice_blue

h. For a given chat, retrieve its creator.

```
SELECT 'Group Chat' AS chat_type, cg.group_id AS chat_id, u.user_id, u.username, u.email
FROM Chat_Group AS cg
JOIN User AS u ON cg.creator_id = u.user_id
WHERE cg.group_id = 3;
```

Output

chat_type	chat_id	user_id	username	email
Group Chat	3	3	bob_green	bob.green@hotmail.com

i. For a given chat, retrieve all its admins (including the creator).

```
SELECT 'Group Chat' AS chat_type, cg.group_id AS chat_id, u.user_id, u.username, u.email, 'Creator' AS role
FROM Chat_Group AS cg
JOIN User AS u ON cg.creator_id = u.user_id
WHERE cg.group_id = 2

UNION

SELECT 'Group Chat' AS chat_type, cg.group_id AS chat_id, u.user_id, u.username, u.email, 'Admin' AS role
FROM Administers_Group AS ag
JOIN Chat_Group AS cg ON ag.group_id = cg.group_id
JOIN User AS u ON ag.admin_id = u.user_id
WHERE cg.group_id = 2;
```

Output

chat_type	chat_id	user_id	username	email	role
Group Chat	2	2	alice_blue	alice.blue@gmail.com	Admin
Group Chat	2	2	alice_blue	alice.blue@gmail.com	Creator

j. For a given chat admin, retrieve his/her permissions.

```
SELECT cg.group_id AS chat_id, u.user_id, u.username, ad.admin_permissions AS permissions
FROM Administers_Group AS ag
JOIN Admin AS ad ON ad.user_id = ag.admin_id
JOIN Chat_Group AS cg ON ag.group_id = cg.group_id
JOIN User AS u ON ag.admin_id = u.user_id
WHERE ag.admin_id = 2;
```

Output

chat_id	user_id	username	permissions
2	2	alice_blue	Create, Delete

k. For a given chat, retrieve all its message history.

```
SELECT M.*
FROM Message M
WHERE M.private_chat_id = null
   OR M.group_id = 3
   OR M.group_id = 5;
```

Output

message_id	content	message_type	timestamp	sender_id	group_id	private_chat_id	is_archived	is_pinned	is_deleted	last_edit_timestamp	view_count
3	New tech release on Friday at noon!	text	2023-04-01 12:00:00	3	3		0	0	0		0
5	This new album is a hit!	text	2023-04-01 14:00:00	5	5		0	0	0		0
8	Who is attending the tech webinar?	text	2023-04-02 17:00:00	3	3		0	0	0		0
10	Anyone going to the live concert this weekend?	text	2023-04-02 19:00:00	5	5		0	0	0		0

l. For a given chat, retrieve its message during a specific date-time range.

```
SELECT M.*
FROM Message M
WHERE (private_chat_id = null OR group_id = 3 OR group_id = 5)
   AND timestamp between '2023-04-01 12:00:00' AND '2023-04-02 17:00:00'
   ;
```

Output

message_id	content	message_type	timestamp	sender_id	group_id	private_chat_id	is_archived	is_pinned	is_deleted	last_edit_timestamp	view_count
3	New tech release on Friday at noon!	text	2023-04-01 12:00:00	3	3		0	0	0		0
5	This new album is a hit!	text	2023-04-01 14:00:00	5	5		0	0	0		0
8	Who is attending the tech webinar?	text	2023-04-02 17:00:00	3	3		0	0	0		0

m. For a given chat, retrieve all messages posted by a user during a specific date-time range.

```
SELECT *
FROM Message
WHERE sender_id = 3
  AND (private_chat_id = null OR group_id = 3 )
  AND timestamp BETWEEN '2023-04-01 11:00:00' AND '2023-04-02 18:00:00  ';
```

Output

message_id	content	message_type	timestamp	sender_id	group_id	private_chat_id	is_archived
3	New tech release on Friday at noon!	text	2023-04-01 12:00:00	3	3		0
8	Who is attending the tech webinar?	text	2023-04-02 17:00:00	3	3		0

n. For a given chat, retrieve its unread messages.

```
SELECT m.*
FROM Message m
JOIN MessageInteraction mi ON m.message_id = mi.message_id
WHERE (m.group_id = 5)
  AND mi.read_status = FALSE;
```

Output

message_id	content	message_type	timestamp	sender_id	group_id	private_chat_id	is_archived	is_pinned	is_deleted	last_edit_timestamp	view_count
5	This new album is a hit!	text	2023-04-01 14:00:00	5	5		0	0	0		0

o. For a given chat, retrieve the last n (say 100) message.

```
SELECT *
FROM Message
WHERE  group_id = 2
ORDER BY timestamp DESC
LIMIT 100;
```

Output

message_id	content	message_type	timestamp	sender_id	group_id	private_chat_id	is_archived	is_pinned	is_deleted	last_edit_timestamp	view_count
7	Reminder about the safety gear for hiking.	text	2023-04-02 16:00:00	2	2		0	0	0		0
2	Looking forward to our hike next week!	text	2023-04-01 11:00:00	2	2		0	0	0		0

p. For a given message ID, retrieve all its information.

```
SELECT *
FROM Message
WHERE message_id = 4;
```

Output

message_id	content	message_type	timestamp	sender_id	group_id	private_chat_id	is_archived	is_pinned	is_deleted	last_edit_timestamp	view_count
4	Has anyone read the latest by Stephen King?	text	2023-04-01 13:00:00	4	4		0	0	0		0