**Information Management II SQL Project – Killian Ronan - 18328687**

The database which I designed was for a Chat Room application. Chat rooms are groups which multiple users can be a member of and send messages to each other from. Different chat rooms can have different participants. This many-to-one relationship is accounted for with the participants table which links users by their user ID to the group they are a part of by ‘roomID’. Messages sent to a group are stored in the Messages table. For every message sent to a group, a notification is added to the Notifications table for each user who is apart of the group the message was sent in. Querying the notifications table would allow a user to see how many unready messages they have.

Users of the chat would have an account created in the ‘User’ table. There is a semantic constraint on the username field. This field must contain a number and be of length greater than 3 characters. The following checks enforce these rules.

CONSTRAINT check\_UsernameHasNum CHECK (username regexp '[0-9]'),

CONSTRAINT check\_UsernameLen CHECK (length(userName) > 3).

The primary key for this table is ‘id’ which is referenced in most of the other tables, linking users to chat rooms, messages etc. I included the auto increment field to this primary key for efficiency, this means after every insert the ‘id’ field will increment with the default starting value being 1. We do not have to specify the ‘id’ field when inserting into this table as it will automatically be assigned.

id int not null AUTO\_INCREMENT

Each column bar ‘profilePicture’ also has the ‘not null’ constraint as some users may not want to upload a profile picture. Profile pictures are stored as a BLOB in the database.

Once the user confirms their email address the ‘email\_confirmed’ bit field would turn to 1 for true. I included a view further on to show users with confirmed emails.

The ‘created\_at’ field is a timestamp which represents the creation time of the user account.

The ‘Room’ table represents each chat group. This also has an id column with the AUTO\_INCREMENT field. The ‘created\_by’ column is a foreign key which references the ‘User’ table linking the creator of the chat room to their user ID.

The ‘Message’ table has the same id field with the AUTO\_INCREMENT set. There are three foreign key constraints in this table which link each record(message sent) to its respective room through ‘roomID’, to which user has sent the message through ‘userID’ and whether the message is a reply to a previous message or not through the ‘reponseTo’ column. Every column bar the ‘responseTo’ column has the ‘not null’ constraint as only some of the messages sent will be replies to other messages.

The ‘Participants’ table links users to which chat rooms they are a member of. This is done through the foreign key constraints: ‘roomID’ and ‘userID’. There are columns for the time a user joined a group and the time a user left a group. All columns have the ‘not null’ constraint bar the ‘time\_left’ column as some users may never leave a chat room. I tested updating this field with the following command.

UPDATE participants set time\_left = NOW() where userID = 2 and roomID = 1;

The ‘Notifications’ table shows users what messages they are yet to read. Whenever a user sends a message to a group a record is added to the ‘Notifications’ table for all users in the chat room bar the sender. The message sent is linked with the ‘messageID’ foreign key. The ‘isRead’ column is set to 0 initially showing the users have not read the message. When the user reads a message, this column should be set to 1. Users can query this table by ‘userID’ and ‘roomID’ to see how many unread messages they have for a chat room.

Finally, the ‘Attachment’ table is for messages which include an attachment. These attachments are linked back to the message through the ‘messageID’ foreign key. The ‘type’ column represents the type of file being stored. The ‘name’ column is the name of the file and the ‘file’ column is the file stored as a BLOB. Storing attachments in a database can however become quite inefficient. To cover for this, I included the option to nullify the ‘file’ field. If this field is null, the path to the attachment is included in the ‘name’ column. These files could be stored on a server somewhere for users to access more efficiently.

**Views**

I created two views for my database. The first was made using a simple date comparison. The view ‘ParticipantsFrom2019’ displays all participants of any chat room who has been a member since after 2019.

CREATE VIEW ParticipantsFrom2019 AS

SELECT userID, time\_joined

FROM participants

WHERE time\_joined >= TIMESTAMP("2019-01-01", "00:00:00");

The second view I created was to see users who have confirmed their email. Users with their emails confirmed would have the column ‘email\_confirmed’ set to 1.

CREATE VIEW EmailConfirmedUsers AS

SELECT username, full\_name, email, created\_at

FROM user

WHERE email\_confirmed = 1;

The third view is extremely similar and pulls users whos’ emails are not confirmed. Users with their emails confirmed would have the column ‘email\_confirmed’ set to 0.

CREATE VIEW NonEmailConfirmedUsers AS

SELECT username, full\_name, email, created\_at

FROM user

WHERE email\_confirmed = 0;

**Triggers**

The first trigger I included in my database is used when a user creates a new chat room. When the user creates a new chat room they should be automatically added as a participant to that room and should not have to add themselves. This is covered for with the ‘addRoomCreator’ trigger.

CREATE TRIGGER addRoomCreator

AFTER INSERT ON room

FOR EACH ROW

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(NEW.id, NEW.created\_by, NOW(), null);

I tested this trigger using the following command.

INSERT INTO Room(name, created\_by, created\_at, active)

VALUES('room4', 1, NOW(), 0);

The next two triggers are used when a message is deleted from the database. ‘removeMessageAttachment’ is a trigger which deletes the associated attachment to a message if there is one.

CREATE TRIGGER removeMessageAttachment

BEFORE DELETE ON message

FOR EACH ROW

DELETE FROM attachment WHERE attachment.messageID=OLD.id;

Similarly the ‘removeMessageNotifications’ trigger deletes all instances of the deleted message from the notifications table.

CREATE TRIGGER removeMessageNotifications

BEFORE DELETE ON message

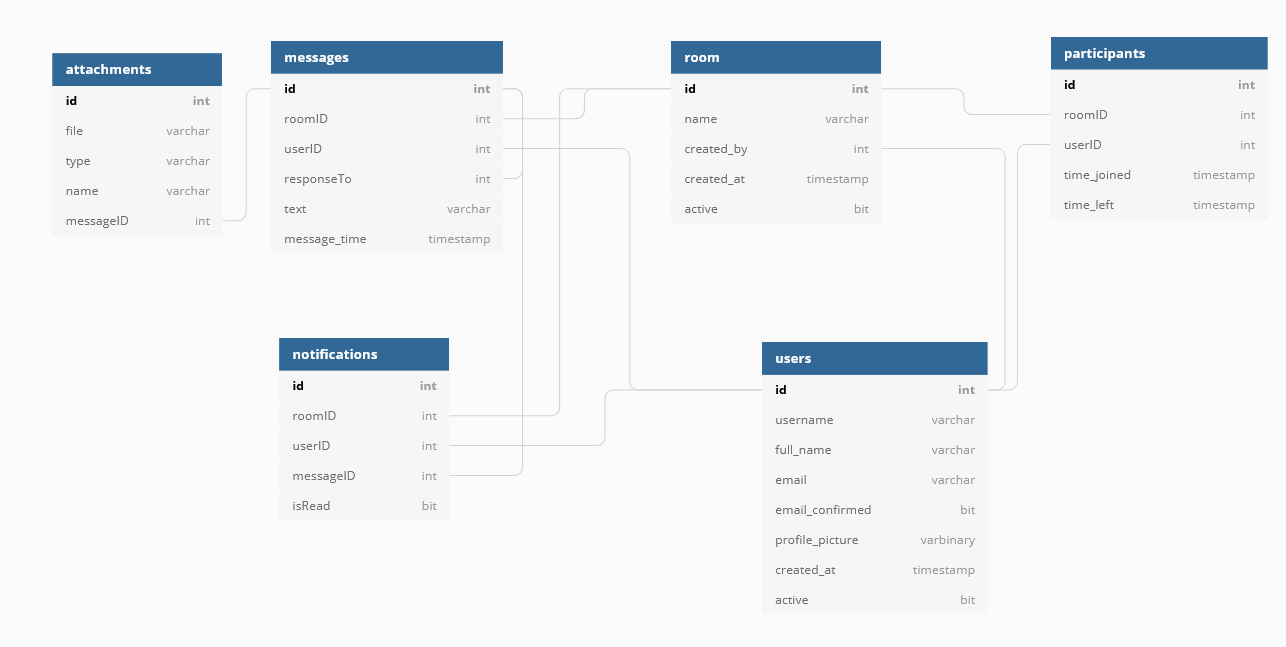
FOR EACH ROW

DELETE FROM notification WHERE notification.messageID=OLD.id;

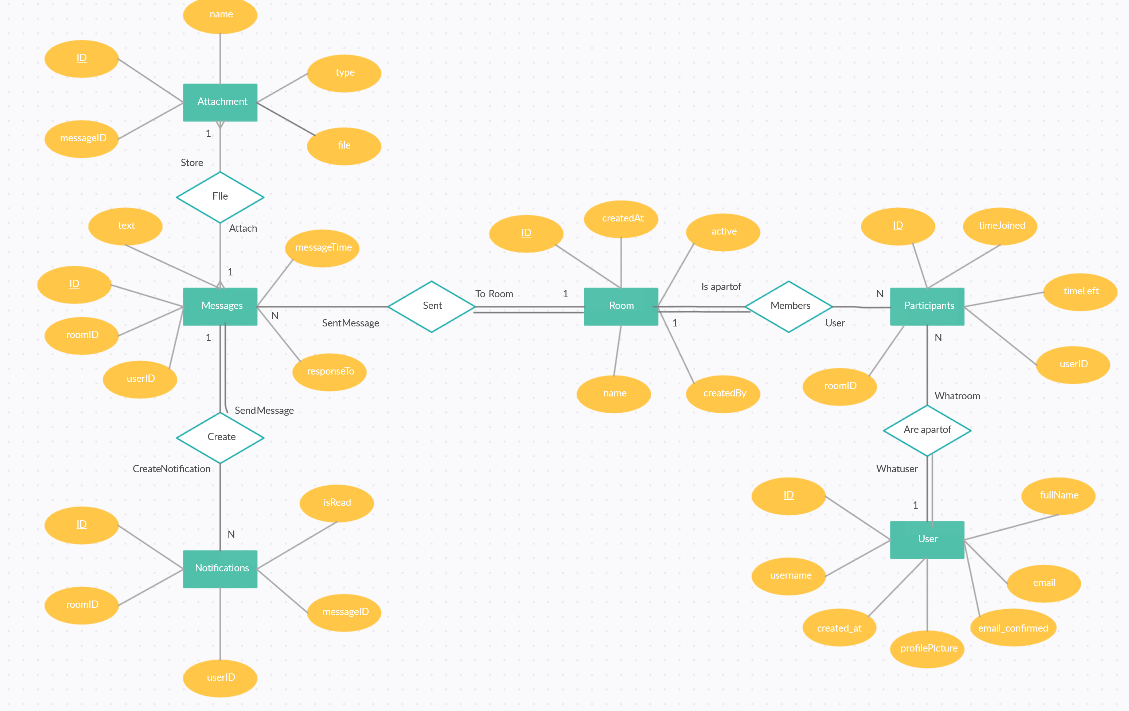
I tested this using the following query which is included at the end of my script.

DELETE FROM message WHERE id=3;

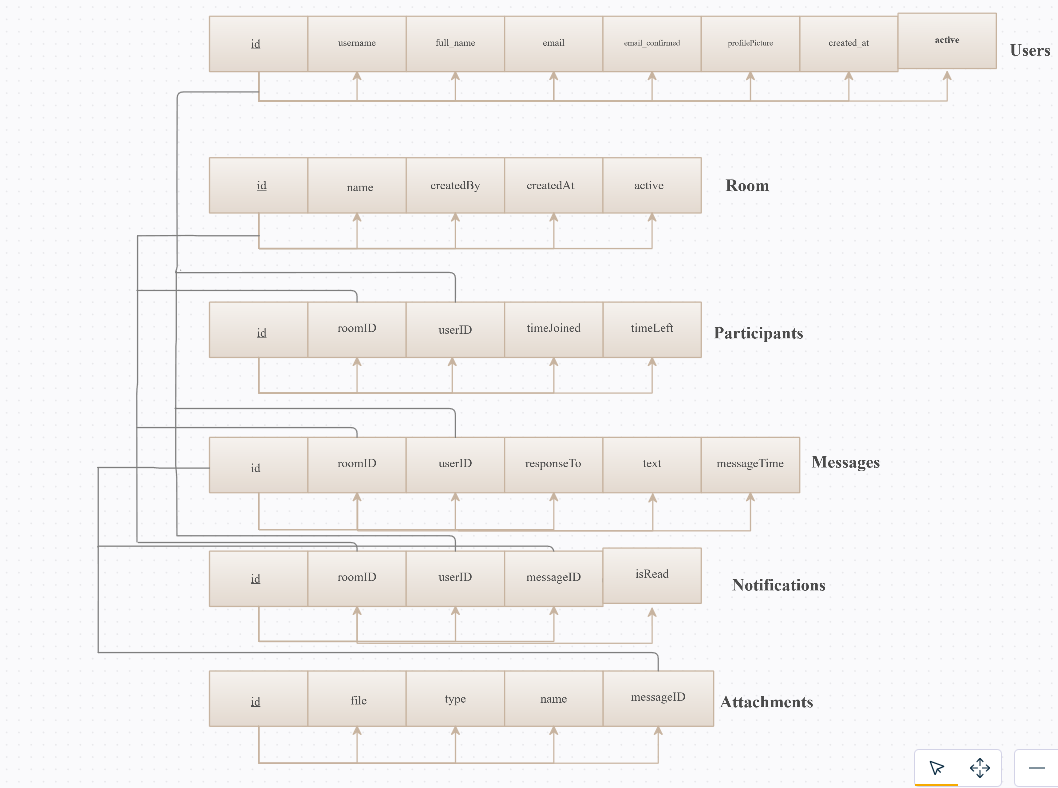
**Relational Schema**



**Entity Relationship Diagram**

****

**Functional Dependency Diagram**

****

**Script**

CREATE TABLE User(

id int not null AUTO\_INCREMENT,

username varchar(20) not null,

full\_name varchar(100) not null,

email varchar(100) not null,

email\_confirmed bit not null,

profile\_picture BLOB null,

created\_at timestamp not null,

active bit not null,

PRIMARY KEY(id),

CONSTRAINT check\_UsernameHasNum CHECK (username regexp '[0-9]'),

CONSTRAINT check\_UsernameLen CHECK (length(userName) > 3)

);

CREATE TABLE Room(

id int not null AUTO\_INCREMENT,

name varchar(100) not null,

created\_by int not null,

created\_at timestamp not null,

active bit not null,

FOREIGN KEY(created\_by) REFERENCES User(id),

PRIMARY KEY(id)

);

CREATE TABLE Message(

id int not null AUTO\_INCREMENT,

roomID int not null,

userID int not null,

responseTo int null,

text varchar(1000) not null,

message\_time timestamp not null,

FOREIGN KEY(roomID) REFERENCES Room(id),

FOREIGN KEY(userID) REFERENCES User(id),

FOREIGN KEY(responseTo) REFERENCES Message(id),

PRIMARY KEY(id)

);

CREATE TABLE Participants(

id int not null AUTO\_INCREMENT,

roomID int not null,

userID int not null,

time\_joined timestamp not null,

time\_left timestamp null,

FOREIGN KEY(roomID) REFERENCES Room(id),

FOREIGN KEY(userID) REFERENCES User(id),

PRIMARY KEY(id)

);

CREATE TABLE Notification(

id int not null AUTO\_INCREMENT,

roomID int not null,

userID int not null,

messageID int not null,

isRead bit not null,

FOREIGN KEY(roomID) REFERENCES Room(id),

FOREIGN KEY(userID) REFERENCES User(id),

FOREIGN KEY(messageID) REFERENCES Message(id),

PRIMARY KEY(id)

);

CREATE TABLE Attachment(

id int not null AUTO\_INCREMENT,

messageID int not null,

type varchar(20) not null,

name varchar(100) not null,

file BLOB null,

FOREIGN KEY(messageID) REFERENCES Message(id),

PRIMARY KEY(id)

);

-- 10 users

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user1','Killian Ronan','kronan@tcd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user2','John Doe','jdoe@tcd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user3','Mary Magdalene','mmagdelene@tcd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user4','Robin Banks','rbanks@tcd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user5','Ben Dover','bdover@tcd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user6','Saad Maan','smaan@tcd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user7','Patrick Ennis','pennis@tcd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user8','Chris Peter Bacon','chrispbacon@tcd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user9','Faruk You','faryou@tecd.ie', 0, null, NOW(), 0);

INSERT INTO user(userName,full\_name,email, email\_confirmed, profile\_picture, created\_at, active)

VALUES('user10','Moe Lester','mlester@tecd.ie', 0, null, NOW(), 0);

-- 3 rooms

INSERT INTO Room(name, created\_by, created\_at, active)

VALUES('room1', 1,NOW(), 0);

INSERT INTO Room(name, created\_by, created\_at, active)

VALUES('room2', 4,NOW(), 0);

INSERT INTO Room(name, created\_by, created\_at, active)

VALUES('room3', 7,NOW(), 0);

-- add room participants

-- room 1

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(1, 1,NOW(), null);

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(1, 2,NOW(), null);

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(1, 3,NOW(), null);

-- room 2

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(2, 4,NOW(), null);

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(2, 5,NOW(), null);

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(2, 6,NOW(), null);

-- room 3

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(3, 7,NOW(), null);

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(3, 8,NOW(), null);

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(3, 9,NOW(), null);

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(3, 10,NOW(), null);

-- messages

-- room1

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(1, 1, null, 'Hello room 1', NOW());

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(1, 2, null, 'Hello room 1', NOW());

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(1, 3, null, 'Hello room 1', NOW());

-- room2

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(2, 4, null, 'Hello room 2', NOW());

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(2, 5, null, 'Hello room 2', NOW());

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(2, 6, null, 'Hello room 2', NOW());

-- room3

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(3, 7, null, 'Hello room 3', NOW());

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(3, 8, null, 'Hello room 3', NOW());

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(3, 9, null, 'Hello room 3', NOW());

INSERT INTO message(roomID, userID, responseTo, text, message\_time)

VALUES(3, 10, null, 'Hello room 3', NOW());

-- notifications for each message

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(1, 2, 1, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(1, 3, 1, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(1, 1, 2, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(1, 3, 2, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(1, 1, 3, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(1, 2, 3, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(2, 5, 4, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(2, 6, 4, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(2, 4, 5, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(2, 6, 5, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(2, 4, 6, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(2, 5, 6, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 8, 7, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 9, 7, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 10, 7, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 7, 8, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 9, 8, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 10, 8, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 7, 9, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 8, 9, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 10, 9, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 7, 10, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 8, 10, 0);

INSERT INTO notification(roomID, userID, messageID, isRead)

VALUES(3, 9, 10, 0);

-- Attachments

INSERT INTO attachment(messageID, type, name, file)

VALUES(3, "jpeg", "photo1", 0xFF);

INSERT INTO attachment(messageID, type, name, file)

VALUES(4, "png", "C:\images\photo1", null);

-- Views

-- Participants from 2019

CREATE VIEW ParticipantsFrom2019 AS

SELECT userID, time\_joined

FROM participants

WHERE time\_joined >= TIMESTAMP("2019-01-01", "00:00:00");

-- Email confirmed users

CREATE VIEW EmailConfirmedUsers AS

SELECT username, full\_name, email, created\_at

FROM user

WHERE email\_confirmed = 1;

-- Non Email confirmed users

CREATE VIEW NonEmailConfirmedUsers AS

SELECT username, full\_name, email, created\_at

FROM user

WHERE email\_confirmed = 0;

-- Triggers

-- Adds user who creates room to participants table

CREATE TRIGGER addRoomCreator

AFTER INSERT ON room

FOR EACH ROW

INSERT INTO participants(roomID, userID, time\_joined, time\_left)

VALUES(NEW.id, NEW.created\_by, NOW(), null);

CREATE TRIGGER removeMessageAttachment

BEFORE DELETE ON message

FOR EACH ROW

DELETE FROM attachment WHERE attachment.messageID=OLD.id;

CREATE TRIGGER removeMessageNotifications

BEFORE DELETE ON message

FOR EACH ROW

DELETE FROM notification WHERE notification.messageID=OLD.id;

INSERT INTO Room(name, created\_by, created\_at, active)

VALUES('room4', 1, NOW(), 0);

DELETE FROM message WHERE id=3;

UPDATE participants set time\_left = NOW() where userID = 2 and roomID = 1;

\*\*chatDB was the name of my database which is why ‘use chatDB;’ is included in the .sql script file\*\*