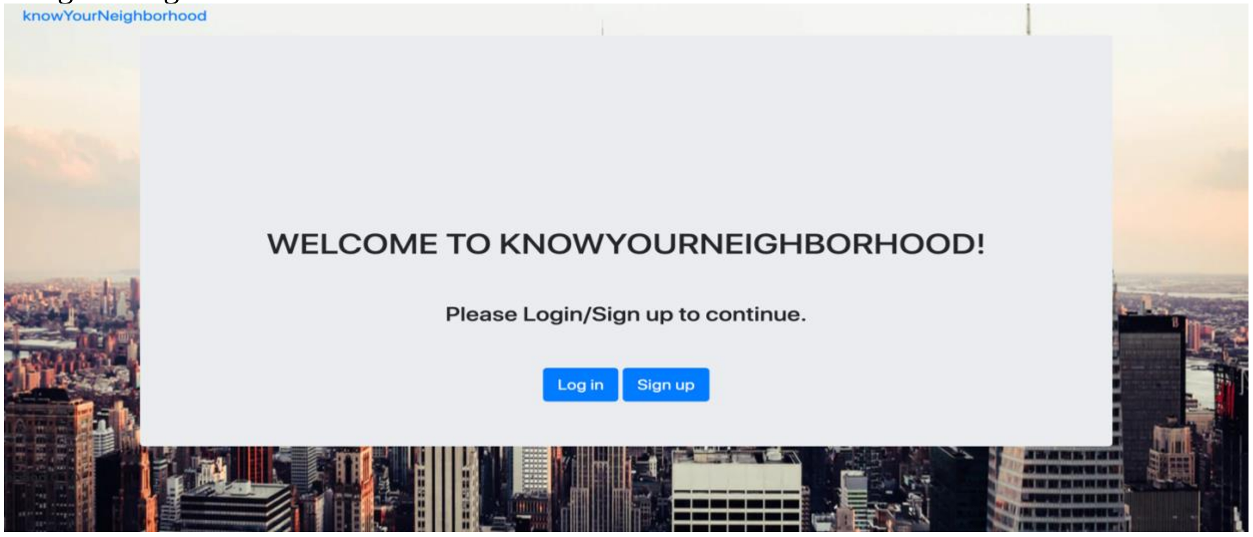
Description:

We have designed a private social network for knowing your neighborhood. ‘KnowYourNeighborhood’ is the best way to stay informed about what’s going on in your neighborhood—whether it’s finding a last-minute babysitter, planning a local event, or sharing safety tips. There are so many ways our neighbors can help us, we just need an easier way to connect with them.

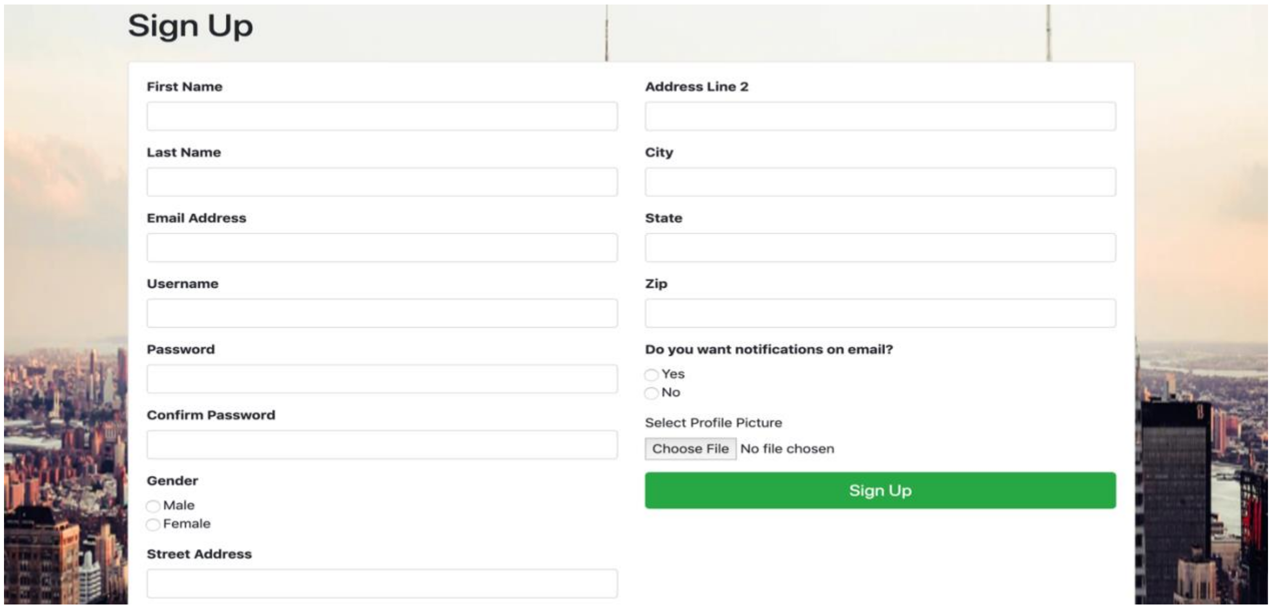
We have designed a relational backend using Flask for this website that allows people to communicate with others in their neighborhood. That is, people should be able to sign up for the service and specify where they live; they can then send and receive messages to other users living close by and participate in discussions with those users.

Workflow:

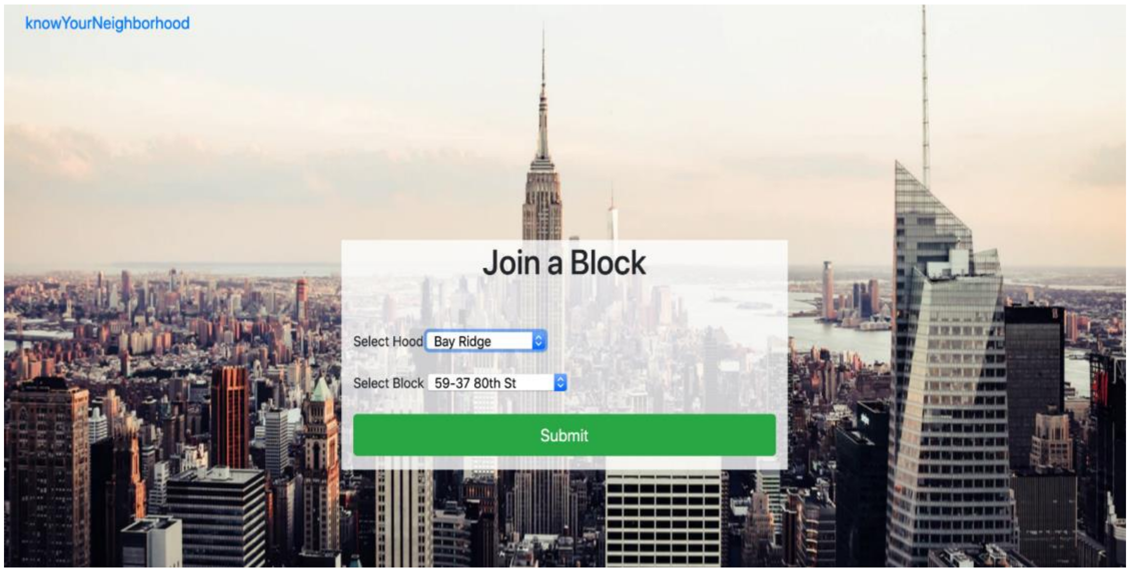
1. Homepage: The user sees this page when he/she opens the website. This will allow a new user to signup on the website using SignUp button and a returning user to sign into his user account using the LogIn button.


knowYourNeighborhood
WELCOME TO KNOWYOURNEIGHBORHOOD!
Please Login/Sign up to continue.
Log in
Sign up

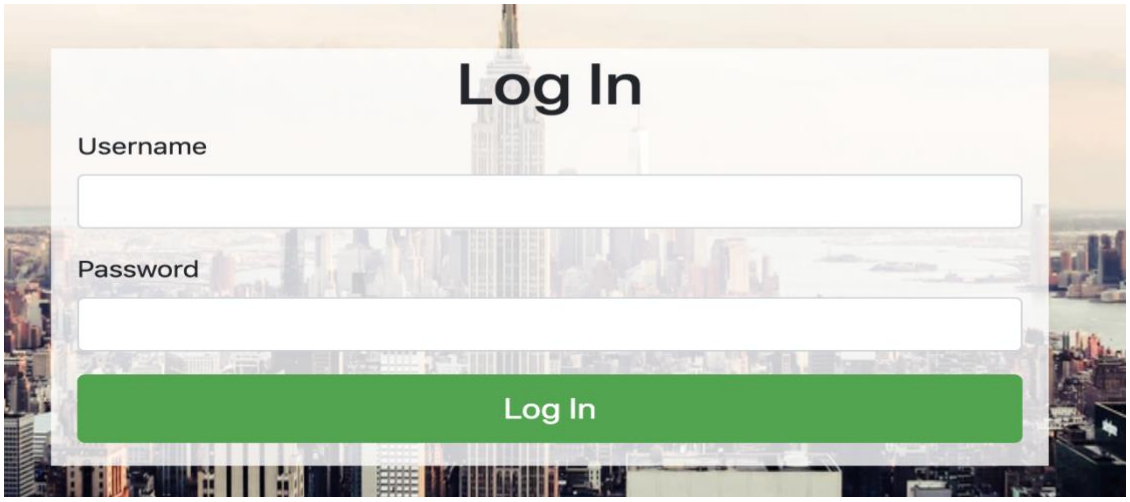
2. **Signup:** The new user registration page requests the user information with a unique username. The passwords are hashed and stored in the database.



3. **Join Block:** Once an user account is created, the user is redirected to choose his hood and block. The user can only join a block if at least three existing members (or all members if less than three) approve the user's request. In the mean time, user is redirected to login page. If the request is approved, the user can login and see the feed of the neighborhood otherwise a message is displayed "You are not a part of any block as of now".



4. <u>Login</u>: The user can login using username and password. The user is redirected to the feed page.



5. <u>User Feed</u>: This page shows all the feeds posted after the user last logged out. It categorises the feed into feeds posted by friends, neighbors, people of same block and hood.

a) <u>Search People:</u> There is a search people tab in which you can search a people and add him as a neighbor or send him a friend request.

b) <u>Search Messages:</u> There is a search message tab in which you can type something and it will list all the feeds that contain that string.

c) <u>Create Thread:</u> This tab allows the user to create a thread and let other people in the same block or their friend to comment on it.

d) <u>Block Approval Request:</u> A new user sends an approval request to all the users within the block that he/she will be joining. Using this tab, a user can allow/deny request of other users to join the block

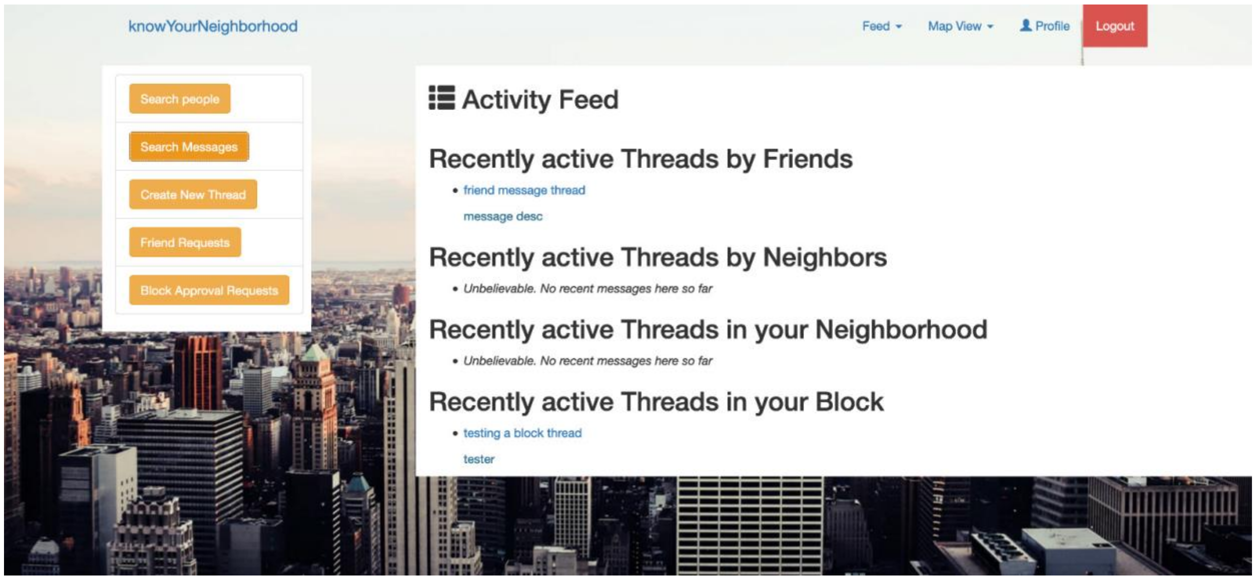
e) <u>Friend Requests:</u> A user can send friend request to any other user and the other user can accept or decline the request.

f) <u>Profile:</u> The user can see/edit his profile and change his password and block.

g) <u>Feed:</u> This button has four options- friend, block, hood and neighbor. On selecting one (say, friend), it will show all the threads and comments on that friend that were created by a friend.

h) <u>Map View:</u> This button has three options- threads, friends and neighbors. On selecting “threads”, it will show the location on the map from where the thread was created. On selecting “friends”, it will show the location of your friends on the map and same for the neighbors.

i) <u>Logout:</u> This button logs the user out and redirect to the login page.



Activity Feed

Recently active Threads by Friends

- friend message thread
  message desc

Recently active Threads by Neighbors

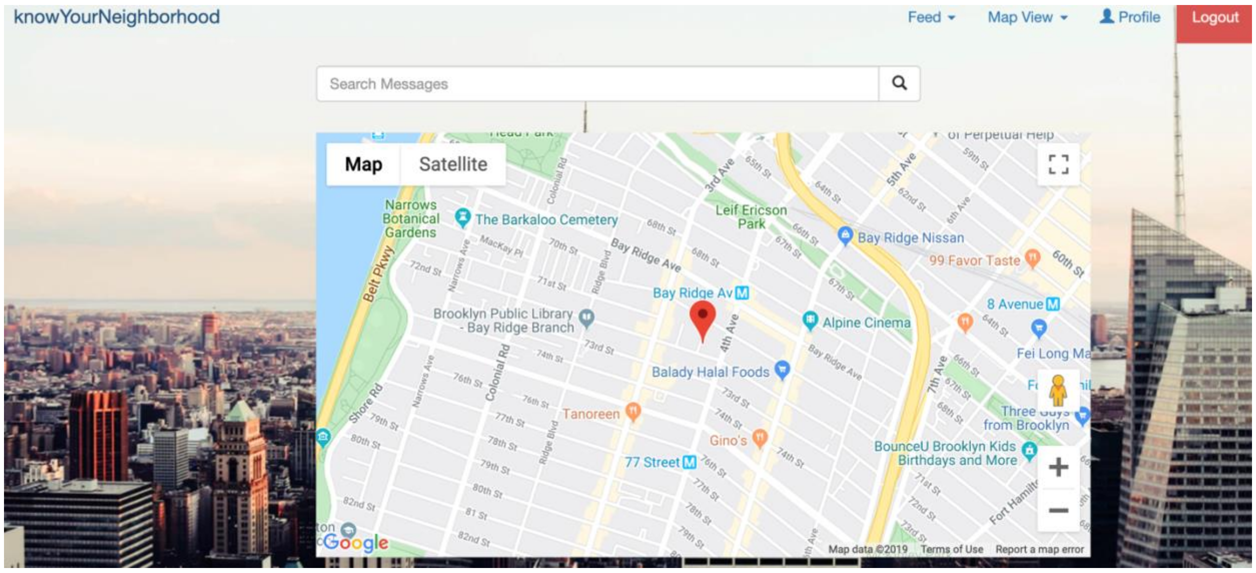
- Unbelievable. No recent messages here so far

Recently active Threads in your Neighborhood

- Unbelievable. No recent messages here so far

Recently active Threads in your Block

- testing a block thread
  tester



6. <u>Profile:</u> User can see his profile and edit it also. He can change his password and change his block when he wants. When the user changes his block, an approval request is sent to all the members of that block.

<u>Additional Features:</u>

1) We are keeping a track of the user session. If the session is expired, user needs to login again.
2) We are listing all the crimes in the block that were discussed about using "Crime Feed" option in Feed tab.
3) We have enabled logging for every activity.
4) We have taken appropriate measures to guard against SQL injection

5) We have enabled CSRF protection globally to guard against cross-site scripting attacks.

Relationship Schema

SIGNUP_DETAILS (uid, username, password, signup_time, last_updated)

Neighborood_Details (nid, nname, maxl, maxlong, minlat, minlong)

BLOCK_DETAILS (bid, zip, nid, maxlat, maxlong, minlat, minlong, bname)

USER_INFO (uid, fname, lname, email, street_address, apt_suite, city, state, zip, block_id, introduction, photo,logout_time, email_preference)

USER_LOCALITY (uid, bid, starttime, endtime)

LOCALITY_ACCESS_REQUEST (uid, bid, status, isActive)

LOCALITY_APPROVAL (requestor_id, uid, bid, approval)

FRIEND_REQUEST (requestor_id, friend_id, status)

FRIENDSHIP (uid, friends_id, starttime, endtime)

NEIGHBORS (uid, neighbour_id, starttime, endtime)

MESSAGETHREADS (tid, created_by_uid, title, description, bid, created_at, access_level)

THREADCOMMENTS (tid, commentId, comment_text, commented_by, commented_at)

Schema Explanation

From the E-R diagram, we derive the following as the relational schema:

a. USER_INFO
This table contains the user information such as the first name, last name, last_login etc.

b. SIGNUP_DETAILS
Stores signup details such as username, password and timestamp of account creation.

c. Neighborood_Details
Stores neighborhood details along with latitude and longitude.

d. BLOCK_DETAILS
Stores block name with coordinates.

e. USER_LOCALITY
Stores the block in which the user resides.

f. LOCALITY_ACCESS_REQUEST
Stores all the requests made by new entrants to a locality and whether that user's is active (git a locality assigned).

g. LOCALITY_APPROVAL
Stores all the recipient information on each request for locality access.

h. FRIEND_REQUEST
Stores all the friend request sent.

i. FRIENDSHIP
Stores information about users who are currently friends.

j. NEIGHBORS
Stores information about users who are currently neighbors.

k. MESSAGETHREADS
Stores information about all the message threads like the title, description and access level assigned by the creator.

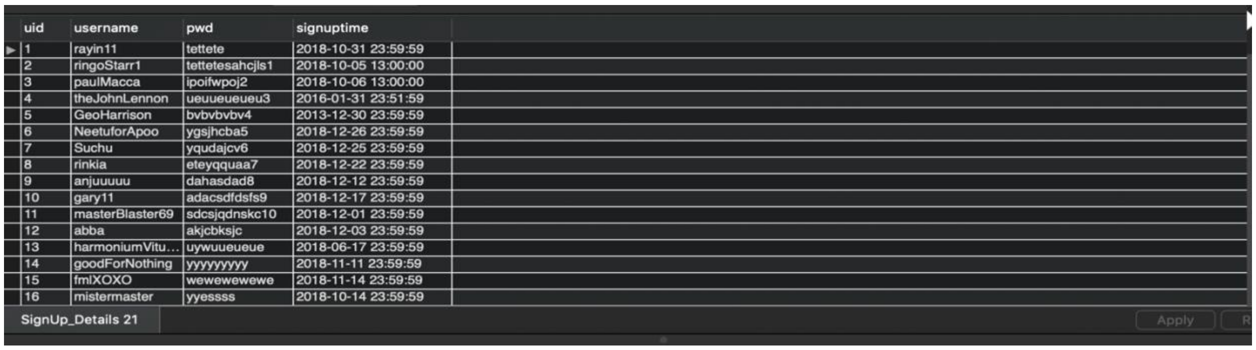
l. THREADCOMMENTS
Stores message content and all other relevant message metadata.

Primary & Foreign Keys

- In the USER_INFO table, uid is the primary key, bid is foreign key referenced from BLOCK DETAILS.
- In the SIGNUP_DETAILS table, uid is the primary key, and refers to USER.uid.
- In the BLOCK_DETAILS table, bid is the primary key and nid refers to nid in the NEIGHBORHOODHOOD table.
- In the Neighborhood_Details table, nid is the primary key.
- In the USER_LOCALITY table, uid refers to uid in the USER_INFO table and bid refers to bid in the BLOCK_DETAILS table.
- In the LOCALITY_ACCESS_REQUEST table, uid refers to uid in the USER_INFO table and bid refers to bid in the BLOCK table.
- In the LOCALITY_APPROVAL table, requestor_id and uid refer to uid in the USER_INFO table and bid refers to bid in the BLOCK table.
- In the FRIEND_REQUEST table, requestor_id and friend_id refer to uid in the USER_INFO table.
- In the FRIENDSHIP table, uid and friend_id refer to uid in the USER_INFO table.
- In the NEIGHBOR table, uid and neighbour_id refer to uid in the USER_INFO table.
- In the MESSAGETHREAD table, created_by_uid refers to uid in the USER_INFO table, bid refers to bid in the BLOCK table.
- In the THREADCOMMENTS table, tid refers to tid in the THREAD table, commented_by refers to uid in the USER_INFO table.

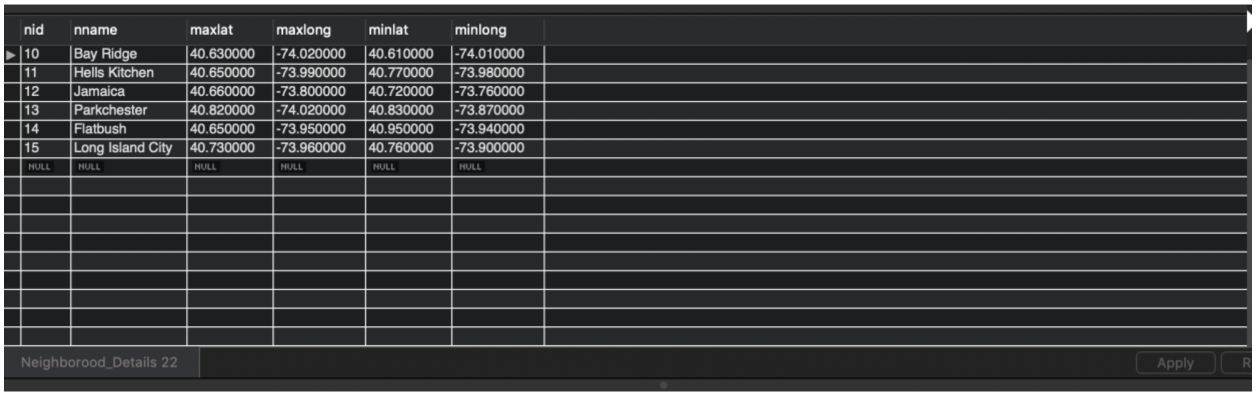
Database Schema

```
CREATE TABLE SignUp_Details
(
    uid int NOT NULL AUTO_INCREMENT,
    username varchar(20) NOT NULL UNIQUE,
    pwd varchar(50) NOT NULL,
    signuptime datetime NOT NULL,
    PRIMARY KEY (uid)
);
```



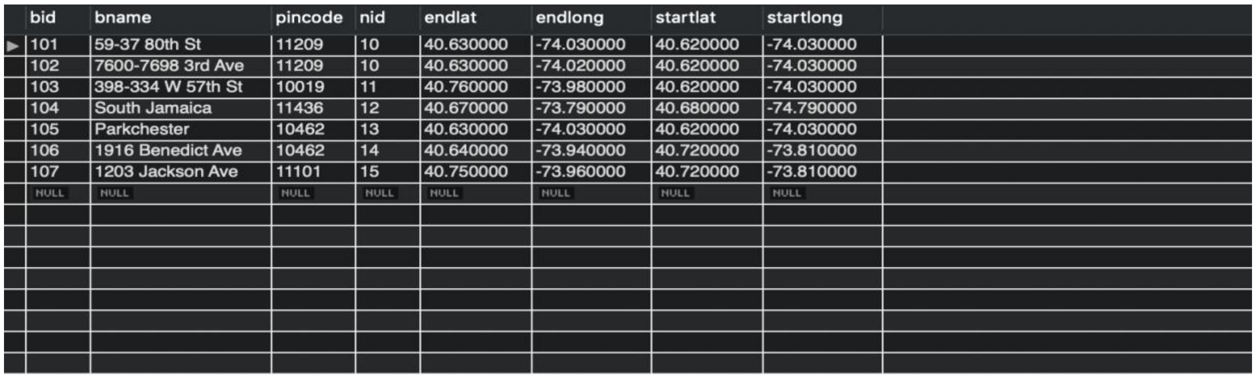
uid	username	pwd	signuptime
1	rayin11	tettete	2018-10-31 23:59:59
2	ringoStarr1	tettetesahcjls1	2018-10-05 13:00:00
3	paulMacca	ipoifwpoj2	2018-10-06 13:00:00
4	theJohnLennon	ueuueueueu3	2016-01-31 23:51:59
5	GeoHarrison	bvbvbvbv4	2013-12-30 23:59:59
6	NeetuforApoo	ygsjhcba5	2018-12-26 23:59:59
7	Suchu	yqudajcv6	2018-12-25 23:59:59
8	rinkia	eteyqquaa7	2018-12-22 23:59:59
9	anjuuuuu	dahasdad8	2018-12-12 23:59:59
10	gary11	adacsdfdsfs9	2018-12-17 23:59:59
11	masterBlaster69	sdcsjqdnskc10	2018-12-01 23:59:59
12	abba	akjcbksjc	2018-12-03 23:59:59
13	harmoniumVitu…	uywuueueue	2018-06-17 23:59:59
14	goodForNothing	yyyyyyyyy	2018-11-11 23:59:59
15	fmlXOXO	wewewewewe	2018-11-14 23:59:59
16	mistermaster	yyessss	2018-10-14 23:59:59

```
CREATE TABLE Neighborood_Details
(
      nid int NOT NULL AUTO_INCREMENT,
    nname varchar(100) NOT NULL,
    maxlat decimal(9,6) NOT NULL,
    maxlong decimal(9,6) NOT NULL,
    minlat decimal(9,6) NOT NULL,
    minlong decimal(9,6) NOT NULL,
    PRIMARY KEY(nid)
);
```



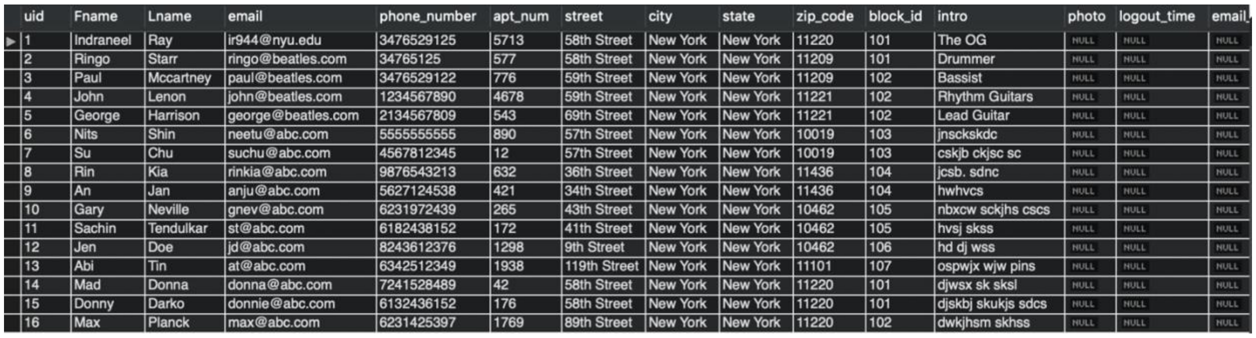
nid	nname	maxlat	maxlong	minlat	minlong
10	Bay Ridge	40.630000	-74.020000	40.610000	-74.010000
11	Hells Kitchen	40.650000	-73.990000	40.770000	-73.980000
12	Jamaica	40.660000	-73.800000	40.720000	-73.760000
13	Parkchester	40.820000	-74.020000	40.830000	-73.870000
14	Flatbush	40.650000	-73.950000	40.950000	-73.940000
15	Long Island City	40.730000	-73.960000	40.760000	-73.900000
NULL	NULL	NULL	NULL	NULL	NULL

```
CREATE TABLE Block_Details
(
bid int NOT NULL AUTO_INCREMENT,
    bname varchar(100) NOT NULL,
    pincode int NOT NULL,
    nid int NOT NULL,
    endlat decimal(9,6) NOT NULL,
    endlong decimal(9,6) NOT NULL,
    startlat decimal(9,6) NOT NULL,
    startlong decimal(9,6) NOT NULL,
    PRIMARY KEY (bid),
    FOREIGN KEY (nid) REFERENCES Neighborood_Details (nid) ON DELETE CASCADE
);
```



bid	bname	pincode	nid	endlat	endlong	startlat	startlong
101	59-37 80th St	11209	10	40.630000	-74.030000	40.620000	-74.030000
102	7600-7698 3rd Ave	11209	10	40.630000	-74.020000	40.620000	-74.030000
103	398-334 W 57th St	10019	11	40.760000	-73.980000	40.620000	-74.030000
104	South Jamaica	11436	12	40.670000	-73.790000	40.680000	-74.790000
105	Parkchester	10462	13	40.630000	-74.030000	40.620000	-74.030000
106	1916 Benedict Ave	10462	14	40.640000	-73.940000	40.720000	-73.810000
107	1203 Jackson Ave	11101	15	40.750000	-73.960000	40.720000	-73.810000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

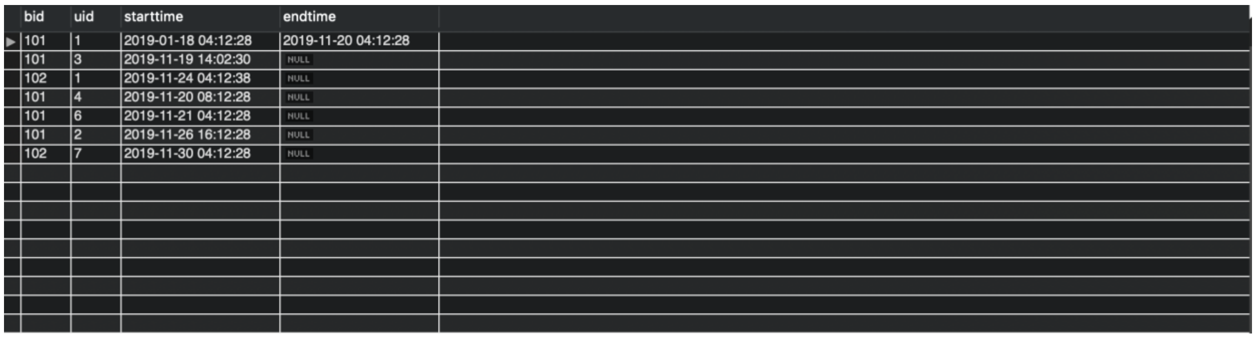
```
Create table User_Info
(
      uid int NOT NULL,
    Fname varchar(20) NOT NULL,
    Lname varchar(20) NOT NULL,
    email varchar(50) NOT NULL,
    phone_number varchar(10) NOT NULL,
    apt_num varchar(20) NOT NULL,
    street varchar(50) NOT NULL,
    city varchar(20) NOT NULL,
    state varchar(20) NOT NULL,
    zip_code int NOT NULL,
    block_id int DEFAULT NULL,
    intro varchar(50) NOT NULL,
    photo blob,
    logout_time datetime DEFAULT NULL,
    email_preference boolean DEFAULT 0,
    FOREIGN KEY (uid)
    REFERENCES SignUp_Details (uid) ON DELETE CASCADE,
    FOREIGN KEY (block_id) REFERENCES Block_Details (bid) ON DELETE CASCADE
);
```



uid	Fname	Lname	email	phone_number	apt_num	street	city	state	zip_code	block_id	intro	photo	logout_time	email_
1	Indraneel	Ray	ir944@nyu.edu	3476529125	5713	58th Street	New York	New York	11220	101	The OG	NULL	NULL	NULL
2	Ringo	Starr	ringo@beatles.com	34765125	577	58th Street	New York	New York	11209	101	Drummer	NULL	NULL	NULL
3	Paul	Mccartney	paul@beatles.com	3476529122	776	59th Street	New York	New York	11209	102	Bassist	NULL	NULL	NULL
4	John	Lenon	john@beatles.com	1234567890	4678	59th Street	New York	New York	11221	102	Rhythm Guitars	NULL	NULL	NULL
5	George	Harrison	george@beatles.com	2134567809	543	69th Street	New York	New York	11221	102	Lead Guitar	NULL	NULL	NULL
6	Nits	Shin	neetu@abc.com	5555555555	890	57th Street	New York	New York	10019	103	jnsckskdc	NULL	NULL	NULL
7	Su	Chu	suchu@abc.com	4567812345	12	57th Street	New York	New York	10019	103	cskjb ckjsc sc	NULL	NULL	NULL
8	Rin	Kia	rinkia@abc.com	9876543213	632	36th Street	New York	New York	11436	104	jcsb. sdnc	NULL	NULL	NULL
9	An	Jan	anju@abc.com	5627124538	421	34th Street	New York	New York	11436	104	hwhvcs	NULL	NULL	NULL
10	Gary	Neville	gnev@abc.com	6231972439	265	43th Street	New York	New York	10462	105	nbxcw sckjhs cscs	NULL	NULL	NULL
11	Sachin	Tendulkar	st@abc.com	6182438152	172	41th Street	New York	New York	10462	105	hvsj skss	NULL	NULL	NULL
12	Jen	Doe	jd@abc.com	8243612376	1298	9th Street	New York	New York	10462	106	hd dj wss	NULL	NULL	NULL
13	Abi	Tin	at@abc.com	6342512349	1938	119th Street	New York	New York	11101	107	ospwjx wjw pins	NULL	NULL	NULL
14	Mad	Donna	donna@abc.com	7241528489	42	58th Street	New York	New York	11220	101	djwsx sk sksl	NULL	NULL	NULL
15	Donny	Darko	donnie@abc.com	6132436152	176	58th Street	New York	New York	11220	101	djskbj skukjs sdcs	NULL	NULL	NULL
16	Max	Planck	max@abc.com	6231425397	1769	89th Street	New York	New York	11220	102	dwkjhsm skhss	NULL	NULL	NULL

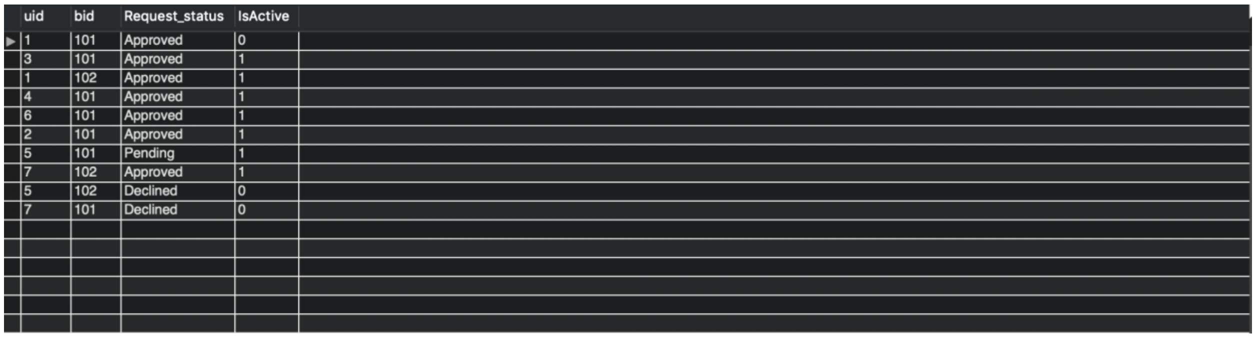
```
Create table User_Locality
(
      bid int NOT NULL,
    uid int NOT NULL,
    starttime datetime NOT NULL,
    endtime datetime DEFAULT NULL,
FOREIGN KEY (bid)
REFERENCES Block_Details (bid)
        ON DELETE CASCADE,
FOREIGN KEY (uid)
REFERENCES SignUp_Details (uid)
        ON DELETE CASCADE

);
```



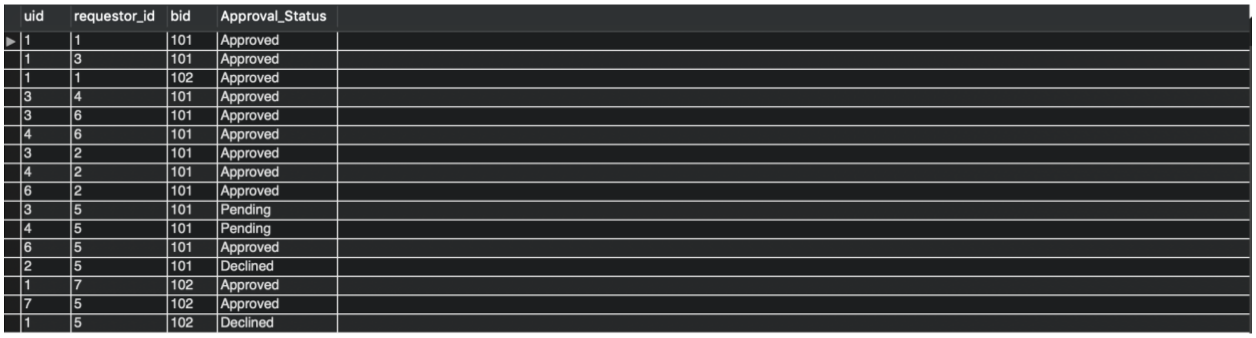
bid	uid	starttime	endtime
101	1	2019-01-18 04:12:28	2019-11-20 04:12:28
101	3	2019-11-19 14:02:30	NULL
102	1	2019-11-24 04:12:38	NULL
101	4	2019-11-20 08:12:28	NULL
101	6	2019-11-21 04:12:28	NULL
101	2	2019-11-26 16:12:28	NULL
102	7	2019-11-30 04:12:28	NULL

```
Create table Locality_Access_Request
(
    uid int NOT NULL,
    bid int NOT NULL,
    Request_status Enum('Approved','Declined','Pending') DEFAULT 'Pending',
    IsActive boolean DEFAULT 1,
    FOREIGN KEY (uid)
REFERENCES SignUp_Details (uid)
        ON DELETE CASCADE,
FOREIGN KEY (bid)
REFERENCES Block_Details (bid)
        ON DELETE CASCADE
);
```



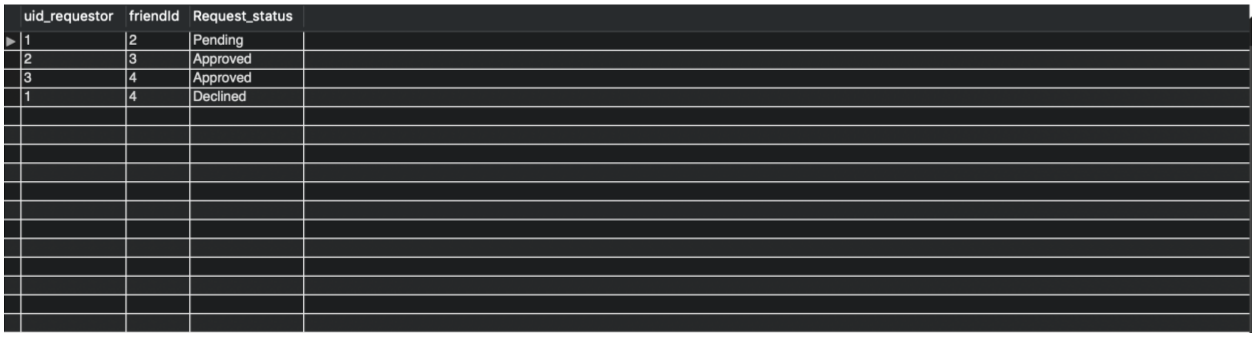
uid	bid	Request_status	IsActive
1	101	Approved	0
3	101	Approved	1
1	102	Approved	1
4	101	Approved	1
6	101	Approved	1
2	101	Approved	1
5	101	Pending	1
7	102	Approved	1
5	102	Declined	0
7	101	Declined	0

```
Create table Locality_Approval
(
      uid int NOT NULL,
    requestor_id int NOT NULL,
    bid int NOT NULL,
    Approval_Status Enum('Approved','Declined','Pending') DEFAULT 'Pending',
    FOREIGN KEY (uid) REFERENCES User_Locality (uid) ON DELETE CASCADE,
    FOREIGN KEY (requestor_id) REFERENCES SignUp_Details (uid) ON DELETE
CASCADE,
    FOREIGN KEY (bid) REFERENCES User_Locality (bid) ON DELETE CASCADE
);
```



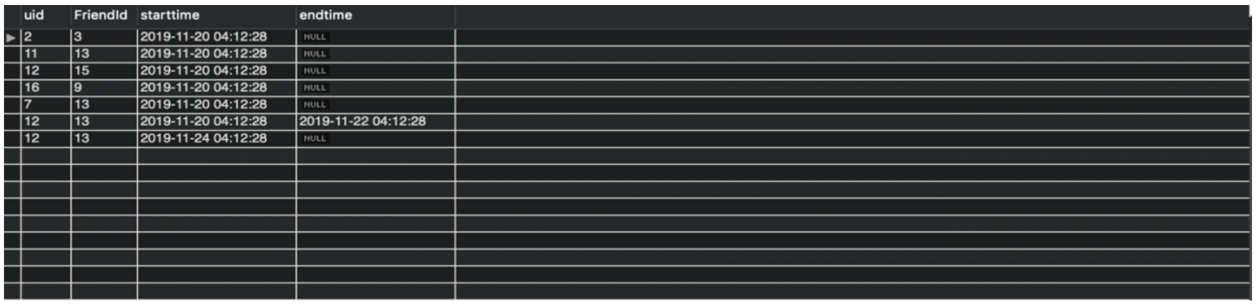
uid	requestor_id	bid	Approval_Status
1	1	101	Approved
1	3	101	Approved
1	1	102	Approved
3	4	101	Approved
3	6	101	Approved
4	6	101	Approved
3	2	101	Approved
4	2	101	Approved
6	2	101	Approved
3	5	101	Pending
4	5	101	Pending
6	5	101	Approved
2	5	101	Declined
1	7	102	Approved
7	5	102	Approved
1	5	102	Declined

```
Create table Friend_Request
(
uid_requestor int NOT NULL,
    friendId int NOT NULL,
    Request_status Enum('Approved','Declined','Pending') DEFAULT 'Pending',
    FOREIGN KEY (uid_requestor) REFERENCES SignUp_Details (uid)
        ON DELETE CASCADE,
FOREIGN KEY (friendId) REFERENCES SignUp_Details (uid) ON DELETE CASCADE
);
```



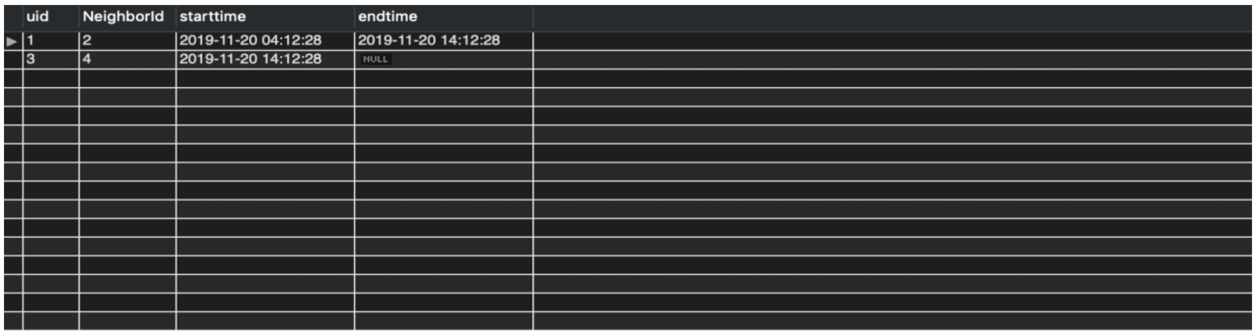
uid_requestor	friendId	Request_status
1	2	Pending
2	3	Approved
3	4	Approved
1	4	Declined

```
Create table Friendship
(
      uid int NOT NULL,
    FriendId int NOT NULL,
      starttime datetime NOT NULL,
    endtime datetime DEFAULT NULL,
    FOREIGN KEY (uid) REFERENCES SignUp_Details (uid) ON DELETE CASCADE,
FOREIGN KEY (FriendId) REFERENCES SignUp_Details (uid) ON DELETE CASCADE
);
```



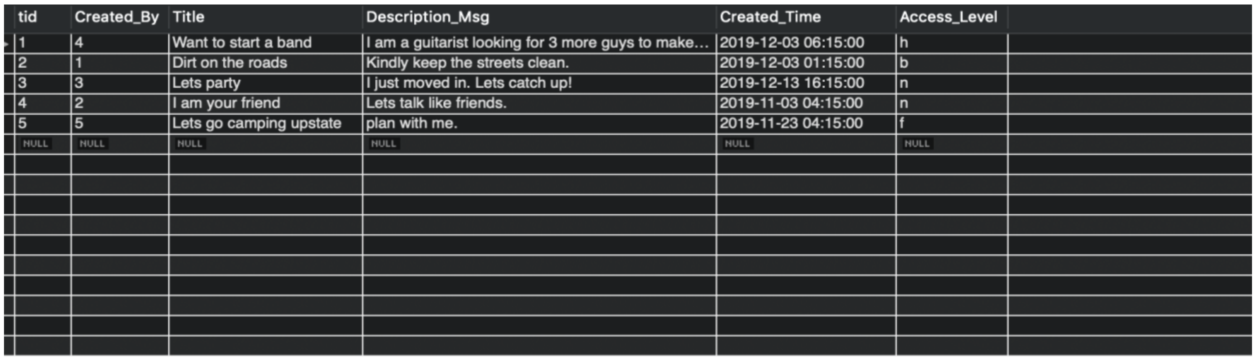
uid	FriendId	starttime	endtime
2	3	2019-11-20 04:12:28	NULL
11	13	2019-11-20 04:12:28	NULL
12	15	2019-11-20 04:12:28	NULL
16	9	2019-11-20 04:12:28	NULL
7	13	2019-11-20 04:12:28	NULL
12	13	2019-11-20 04:12:28	2019-11-22 04:12:28
12	13	2019-11-24 04:12:28	NULL

```
Create table Neighbors
(
uid int NOT NULL,
    NeighborId int NOT NULL,
    starttime datetime NOT NULL,
    endtime datetime DEFAULT NULL,
    FOREIGN KEY (uid) REFERENCES SignUp_Details (uid) ON DELETE CASCADE,
FOREIGN KEY (NeighborId) REFERENCES SignUp_Details (uid) ON DELETE CASCADE
);
```



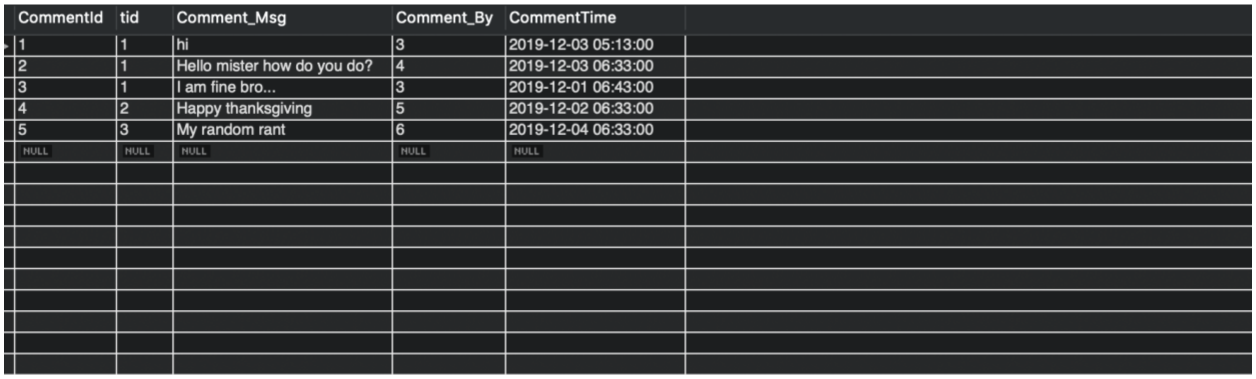
uid	NeighborId	starttime	endtime
1	2	2019-11-20 04:12:28	2019-11-20 14:12:28
3	4	2019-11-20 14:12:28	NULL

```
Create table MessageThreads
(
tid int AUTO_INCREMENT,
    Created_By int NOT NULL,
    Title varchar(50) NOT NULL,
    Description_Msg varchar(200),
    Created_Time datetime NOT NULL,
    Access_Level Enum('f','n','b','h'),
    PRIMARY KEY (tid),
    FOREIGN KEY (Created_By) REFERENCES SignUp_Details (uid) ON DELETE
CASCADE
);
```



tid	Created_By	Title	Description_Msg	Created_Time	Access_Level
1	4	Want to start a band	I am a guitarist looking for 3 more guys to make…	2019-12-03 06:15:00	h
2	1	Dirt on the roads	Kindly keep the streets clean.	2019-12-03 01:15:00	b
3	3	Lets party	I just moved in. Lets catch up!	2019-12-13 16:15:00	n
4	2	I am your friend	Lets talk like friends.	2019-11-03 04:15:00	n
5	5	Lets go camping upstate	plan with me.	2019-11-23 04:15:00	f
NULL	NULL	NULL	NULL	NULL	NULL

```
Create table ThreadComments
(
      CommentId int AUTO_INCREMENT,
      tid int NOT NULL,
    Comment_Msg varchar(200) NOT NULL,
    Comment_By int NOT NULL,
    CommentTime datetime NOT NULL,
    PRIMARY KEY (CommentId),
    FOREIGN KEY (tid)
REFERENCES MessageThreads (tid)
        ON DELETE CASCADE,
FOREIGN KEY (Comment_By)
REFERENCES SignUp_Details (uid)
        ON DELETE CASCADE
);
```



CommentId	tid	Comment_Msg	Comment_By	CommentTime
1	1	hi	3	2019-12-03 05:13:00
2	1	Hello mister how do you do?	4	2019-12-03 06:33:00
3	1	I am fine bro...	3	2019-12-01 06:43:00
4	2	Happy thanksgiving	5	2019-12-02 06:33:00
5	3	My random rant	6	2019-12-04 06:33:00
NULL	NULL	NULL	NULL	NULL