Project Report : OINGO

Ankita Nagpal | an2846@nyu.edu
Yashasvi Dadhe | yd1405@nyu.edu

Introduction

Oingo is a system of textual notes that are visible at a certain location for a certain time range. Every user can create these notes for a certain audience (class) like friends, everyone, or self. These notes can be advertisements, reminders for future or notification notes for friends nearby etc. Every note is associated with certain tags that convey the essence of the note like #shopping, #me or #mustvisit.

The audience gets to decide which notes will they see by uploading filters on location, time, state, class and tags. A single user can upload several filters like the following :

1. Show me notes with tag #fitness from anyone when I am at New York Gym, everyday from 6am to 7am.
2. Show me notes with tag #food from friends when my state is 'lunch break'
3. Show me a reminder note of visiting hospital for a check-up on 28th December, 2018.

We have created a relational schema that can manage such a system.

Relational Schema

note(noteid,text,commentenabled,time,uid,lid,sid,note_radius,class)

noteTag(noteid,tag)

userFilter(fid,uid,lid,rad,sid,class,tag,state)

friendship(uid,friendid)

friendrequest (uid, requestuser,action) **(Added)**

comment(cid,ctext,uid,noteid,time)

user(uid,uname,pwd,email)

location(lid,lname,lat,lon)

history(hid,uid,lid,state,time)

schedule(sid,starttime,endtime,startdate,enddate,repetition) **(Modified)**

Constraints

1. note table
   Primary key :noteid
   Foreign key: lid references location(lid)
   sid references schedule(sid)
   uid references user(uid)
2. noteTag table
   Primary key :noteid,tag
   Foreign key: noteid references note(noteid)

3. userFilter table
   Primary key :fid
   Foreign key: uid references user(uid)
   lid references location(lid)
   sid references schedule(sid)
4. friendship table
   Primary key :uid,friendid
   Foreign key: uid references user(uid)
   Friendid references user(uid)

5. comment table

Primary key :cid
Foreign key: uid references user(uid)
noteid references note(noteid)

6. user table
   Primary key :uid

7. location table
   Primary key :lid

8. history table
   Primary key :lid
   Foreign key: uid references user(uid)
   lid references note(noteid)
9. schedule table
   Primary key :sid

10. friendrequest table
    Foreign key: uid references user(uid)
    requestuser references user(uid)

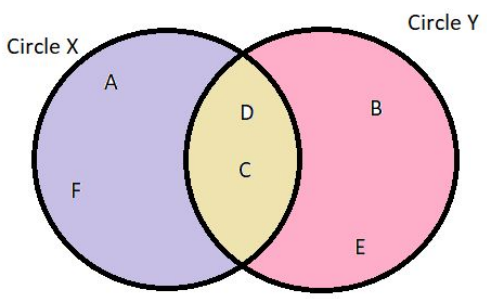
Sample Data

Sample data is such that few users are within the radius of notes. Few users have location outside of the radius of any note.

For instance :
If a note is created for location with name SOHO , then it is ensured that one of the users has location SOHO_near with distance from SOHO less than the note_radius.

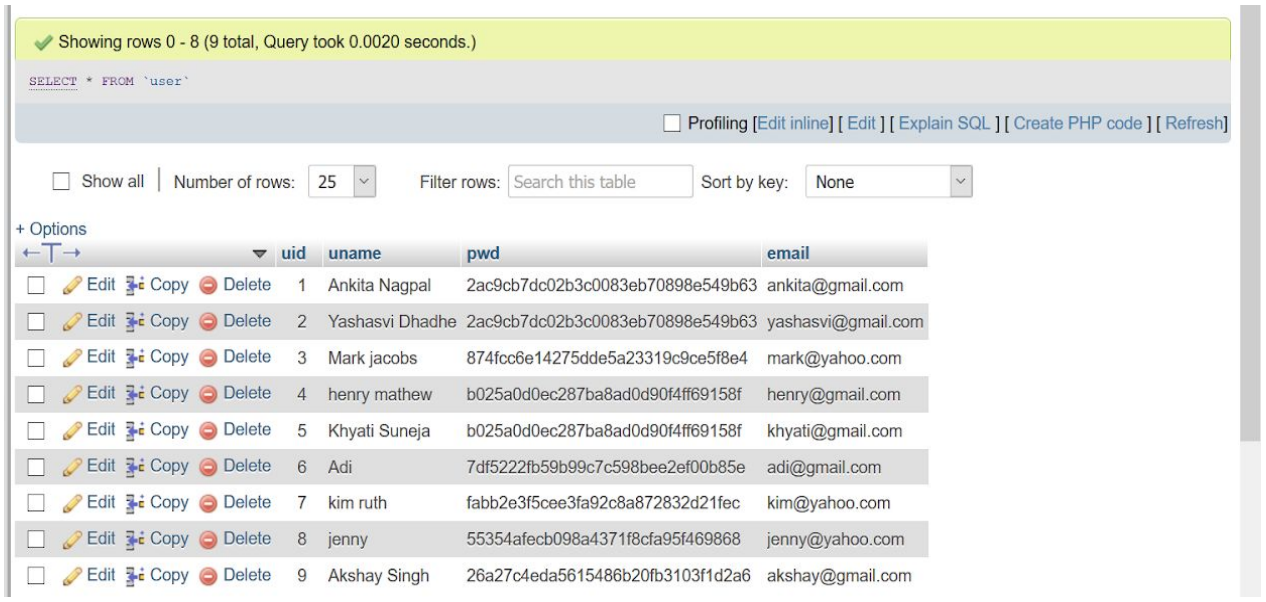
Similarly the locations are created like
Exchange Place and Exchange place harbour
NYU Tandon and 2 metrotech-NYU Tandon

Data can be depicted in the form of following venn diagram.



Circle X represents location of notes. Circle Y represents location of filters. A, B, C, D, E, F are users. C and D are users that satisfy location condition of note as well as their filter. A and F are near notes but they don't satisfy their filter condition on location. B and E are near near location conditions of their filters but they don't satisfy note's condition on location.

1. user table



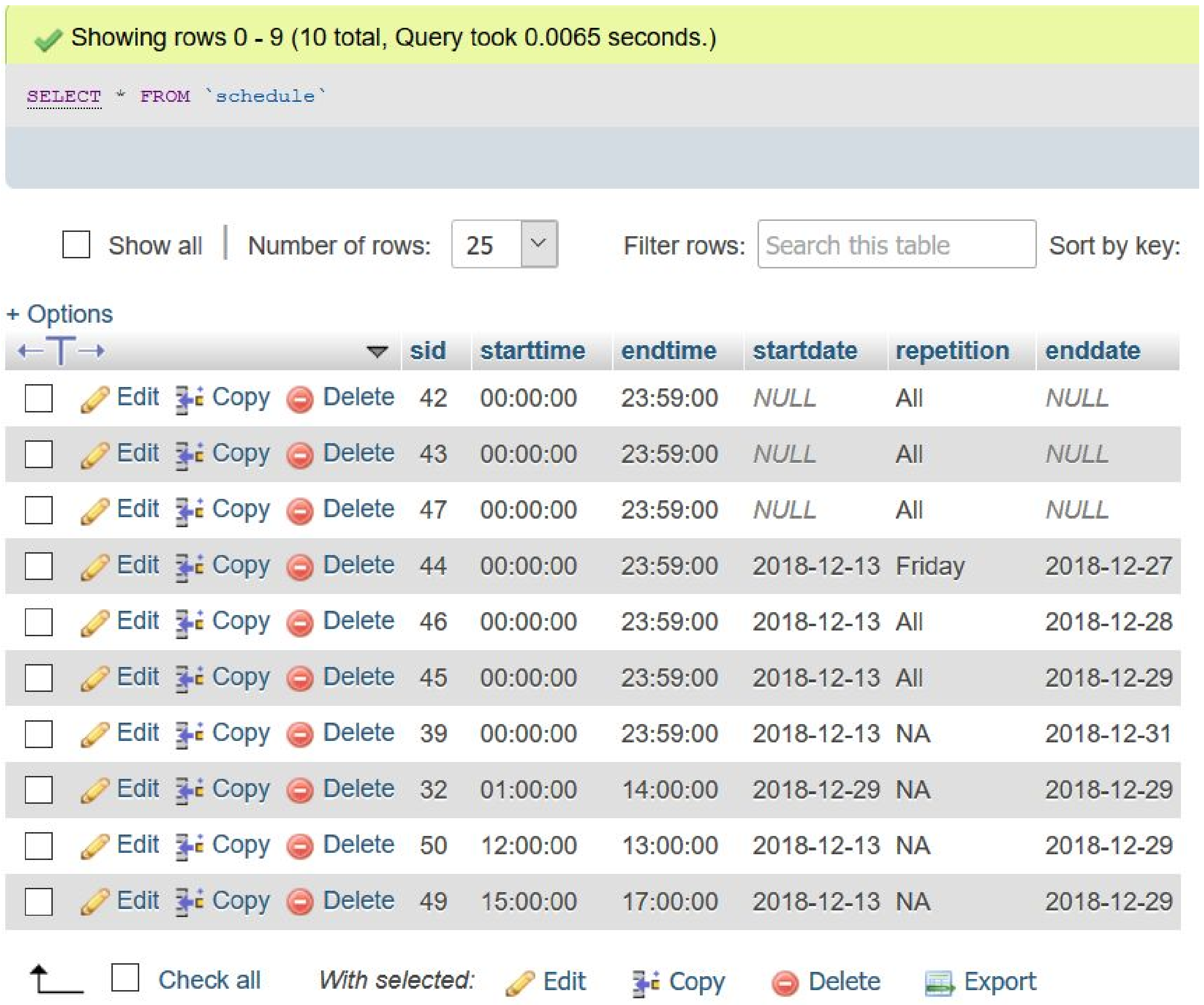
✓ Showing rows 0 - 8 (9 total, Query took 0.0020 seconds.)

```
SELECT * FROM `user`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options

	uid	uname	pwd	email
Edit Copy Delete	1	Ankita Nagpal	2ac9cb7dc02b3c0083eb70898e549b63	ankita@gmail.com
Edit Copy Delete	2	Yashasvi Dhadhe	2ac9cb7dc02b3c0083eb70898e549b63	yashasvi@gmail.com
Edit Copy Delete	3	Mark jacobs	874fcc6e14275dde5a23319c9ce5f8e4	mark@yahoo.com
Edit Copy Delete	4	henry mathew	b025a0d0ec287ba8ad0d90f4ff69158f	henry@gmail.com
Edit Copy Delete	5	Khyati Suneja	b025a0d0ec287ba8ad0d90f4ff69158f	khyati@gmail.com
Edit Copy Delete	6	Adi	7df5222fb59b99c7c598bee2ef00b85e	adi@gmail.com
Edit Copy Delete	7	kim ruth	fabb2e3f5cee3fa92c8a872832d21fec	kim@yahoo.com
Edit Copy Delete	8	jenny	55354afecb098a4371f8cfa95f469868	jenny@yahoo.com
Edit Copy Delete	9	Akshay Singh	26a27c4eda5615486b20fb3103f1d2a6	akshay@gmail.com

2.schedule table **(modified)**



✓ Showing rows 0 - 9 (10 total, Query took 0.0065 seconds.)

```
SELECT * FROM `schedule`
```

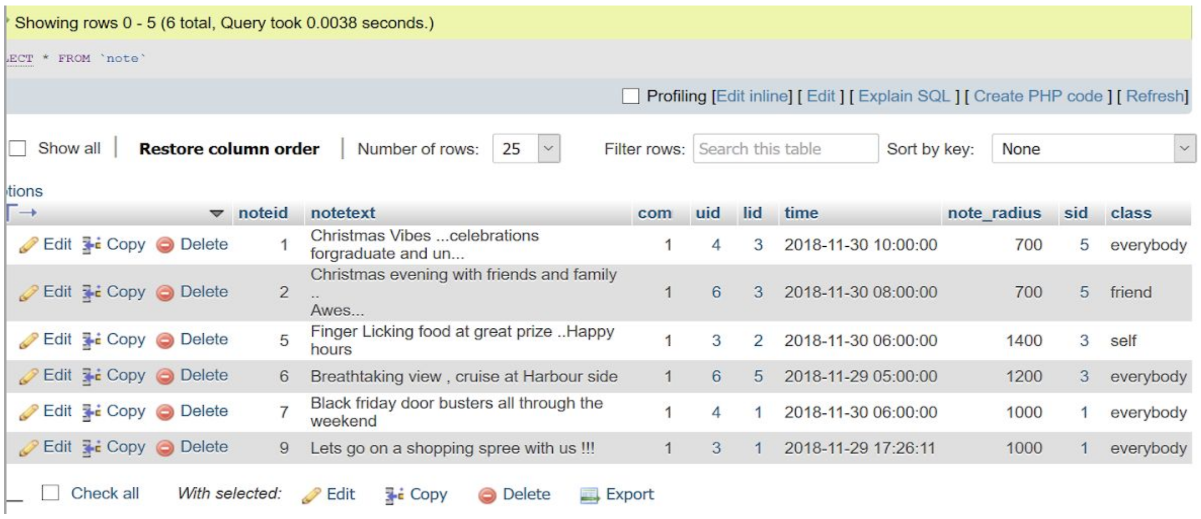
Show all | Number of rows: 25 Filter rows: Search this table Sort by key:

+ Options

	sid	starttime	endtime	startdate	repetition	enddate
Edit Copy Delete	42	00:00:00	23:59:00	*NULL*	All	*NULL*
Edit Copy Delete	43	00:00:00	23:59:00	*NULL*	All	*NULL*
Edit Copy Delete	47	00:00:00	23:59:00	*NULL*	All	*NULL*
Edit Copy Delete	44	00:00:00	23:59:00	2018-12-13	Friday	2018-12-27
Edit Copy Delete	46	00:00:00	23:59:00	2018-12-13	All	2018-12-28
Edit Copy Delete	45	00:00:00	23:59:00	2018-12-13	All	2018-12-29
Edit Copy Delete	39	00:00:00	23:59:00	2018-12-13	NA	2018-12-31
Edit Copy Delete	32	01:00:00	14:00:00	2018-12-29	NA	2018-12-29
Edit Copy Delete	50	12:00:00	13:00:00	2018-12-13	NA	2018-12-29
Edit Copy Delete	49	15:00:00	17:00:00	2018-12-13	NA	2018-12-29

Check all *With selected:* Edit Copy Delete Export

3. note table



Showing rows 0 - 5 (6 total, Query took 0.0038 seconds.)

```
ECT * FROM `note`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

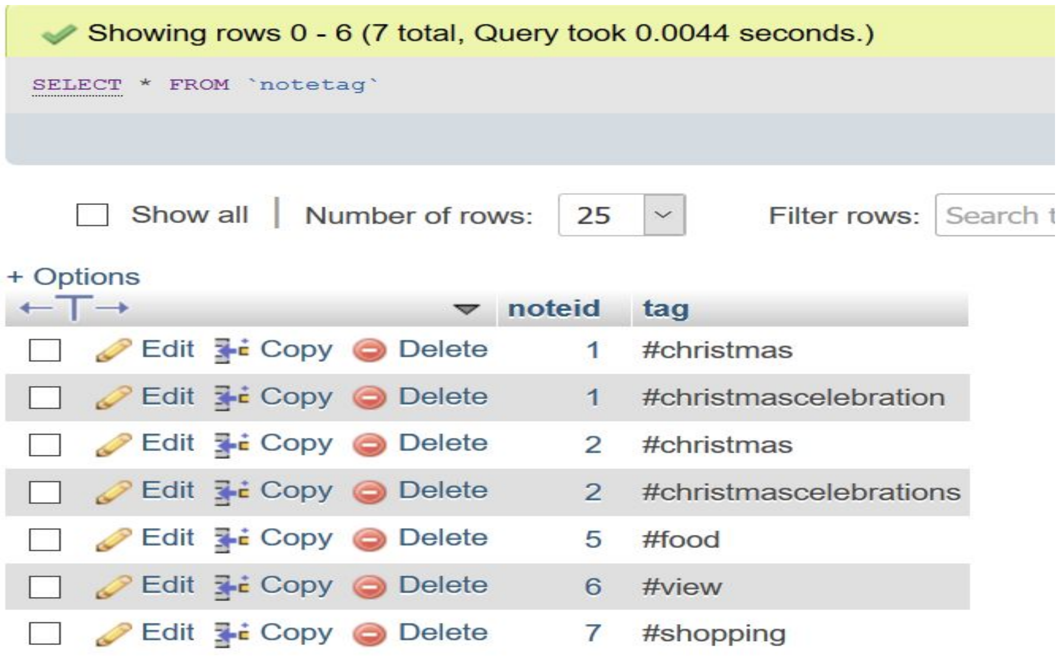
Show all | Restore column order | Number of rows: 25 Filter rows: Search this table Sort by key: None

tions

	noteid	notetext	com	uid	lid	time	note_radius	sid	class
Edit Copy Delete	1	Christmas Vibes ...celebrations forgraduate and un...	1	4	3	2018-11-30 10:00:00	700	5	everybody
Edit Copy Delete	2	Christmas evening with friends and family .. Awes...	1	6	3	2018-11-30 08:00:00	700	5	friend
Edit Copy Delete	5	Finger Licking food at great prize ..Happy hours	1	3	2	2018-11-30 06:00:00	1400	3	self
Edit Copy Delete	6	Breathtaking view , cruise at Harbour side	1	6	5	2018-11-29 05:00:00	1200	3	everybody
Edit Copy Delete	7	Black friday door busters all through the weekend	1	4	1	2018-11-30 06:00:00	1000	1	everybody
Edit Copy Delete	9	Lets go on a shopping spree with us !!!	1	3	1	2018-11-29 17:26:11	1000	1	everybody

Check all With selected: Edit Copy Delete Export

4. notetag table



Showing rows 0 - 6 (7 total, Query took 0.0044 seconds.)

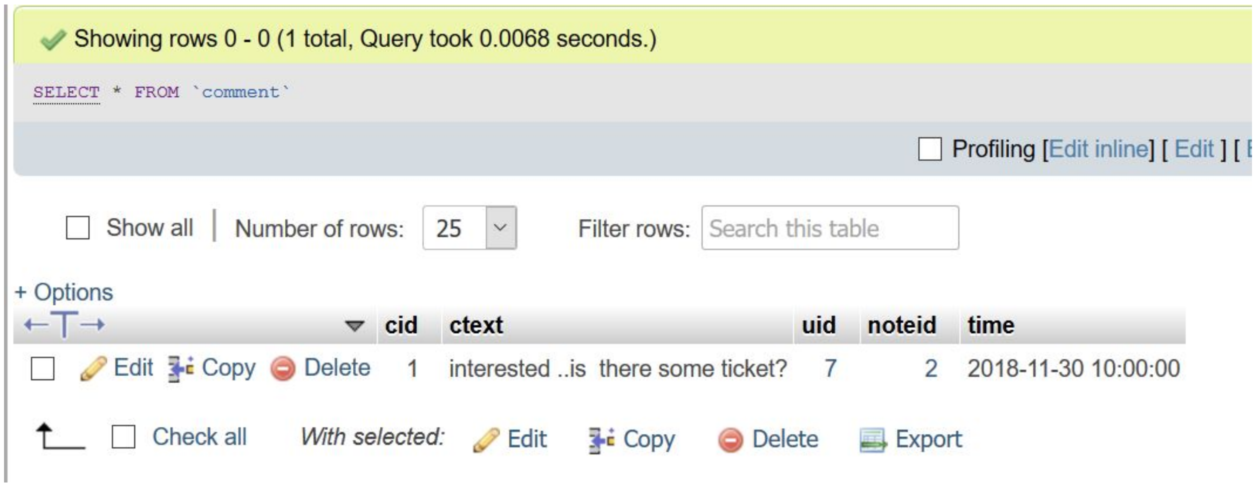
```
SELECT * FROM `notetag`
```

Show all | Number of rows: 25 Filter rows: Search t

+ Options

	noteid	tag
Edit Copy Delete	1	#christmas
Edit Copy Delete	1	#christmascelebration
Edit Copy Delete	2	#christmas
Edit Copy Delete	2	#christmascelebrations
Edit Copy Delete	5	#food
Edit Copy Delete	6	#view
Edit Copy Delete	7	#shopping

5.comment table



✓ Showing rows 0 - 0 (1 total, Query took 0.0068 seconds.)

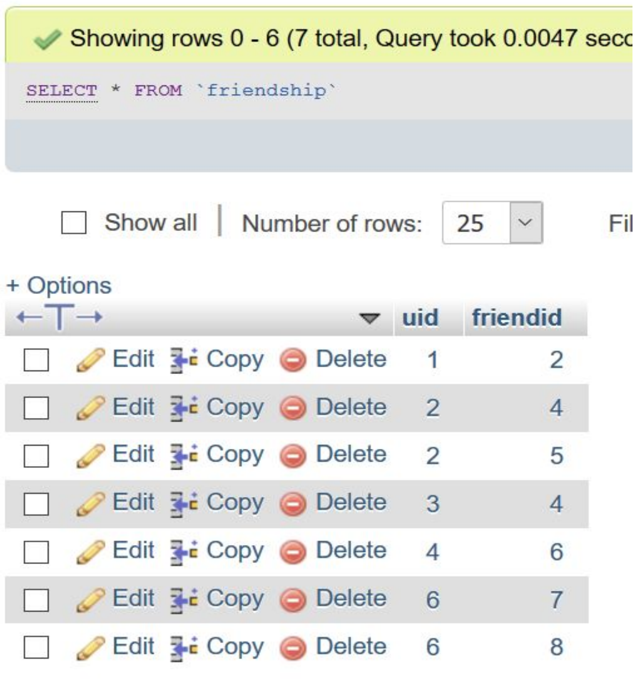
```
SELECT * FROM `comment`
```

Profiling [Edit inline] [Edit] [
Show all | Number of rows: 25
Filter rows: Search this table
+ Options

	cid	ctext	uid	noteid	time
Edit Copy Delete	1	interested ..is there some ticket?	7	2	2018-11-30 10:00:00

Check all
With selected: Edit Copy Delete Export

6.friendship table

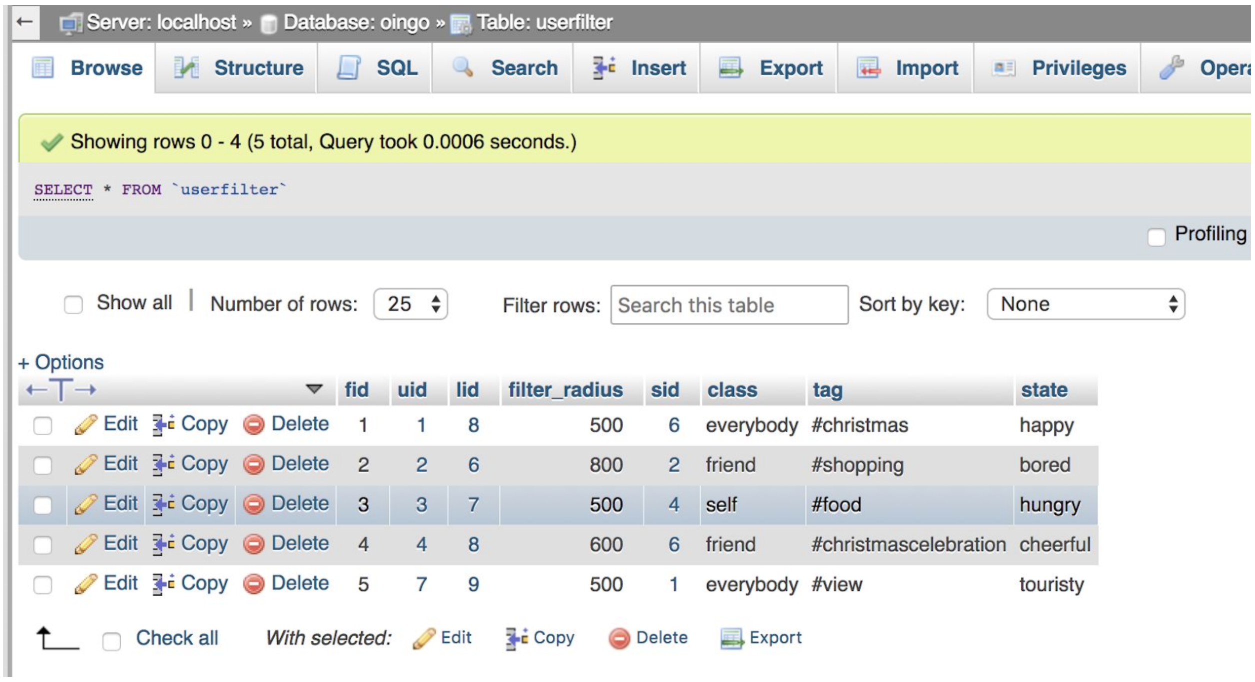


✓ Showing rows 0 - 6 (7 total, Query took 0.0047 seco

```
SELECT * FROM `friendship`
```

	uid	friendid
Edit Copy Delete	1	2
Edit Copy Delete	2	4
Edit Copy Delete	2	5
Edit Copy Delete	3	4
Edit Copy Delete	4	6
Edit Copy Delete	6	7
Edit Copy Delete	6	8

7.userfilter table



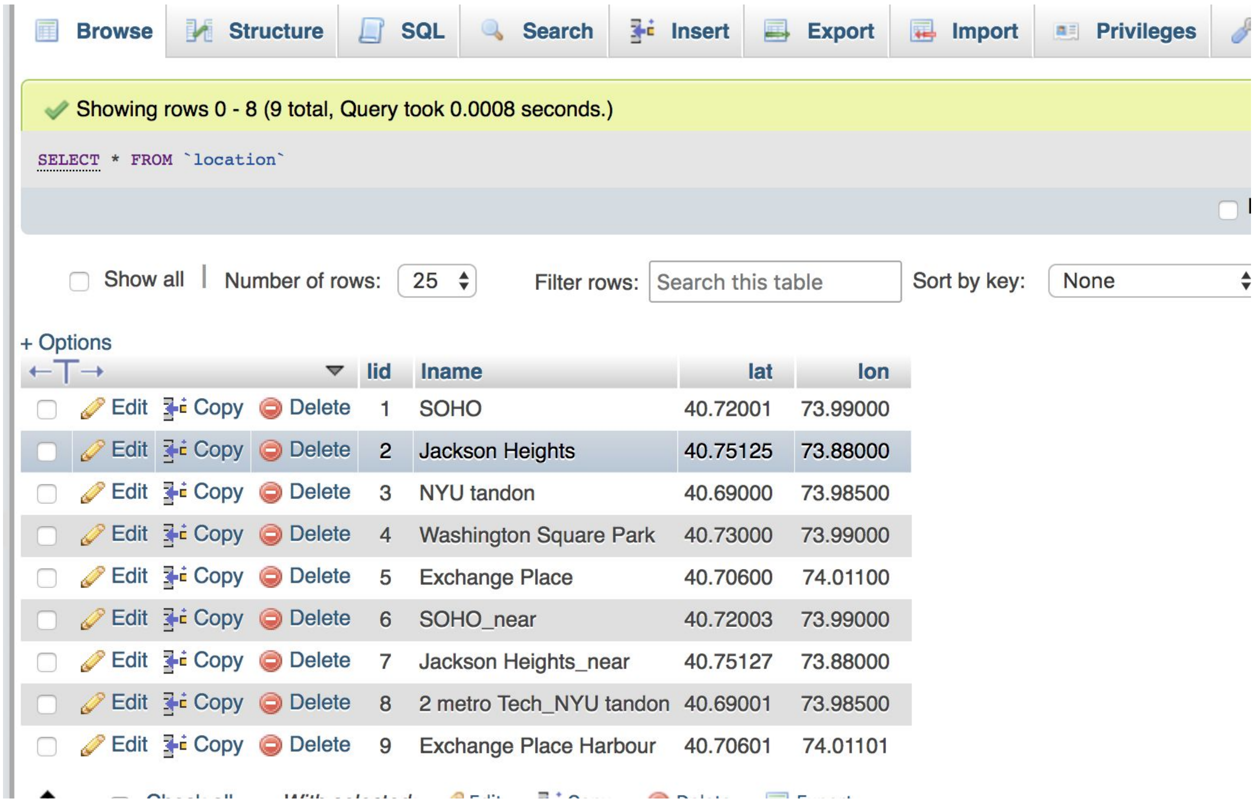
Server: localhost » Database: oingo » Table: userfilter

Showing rows 0 - 4 (5 total, Query took 0.0006 seconds.)

```
SELECT * FROM `userfilter`
```

fid	uid	lid	filter_radius	sid	class	tag	state
1	1	8	500	6	everybody	#christmas	happy
2	2	6	800	2	friend	#shopping	bored
3	3	7	500	4	self	#food	hungry
4	4	8	600	6	friend	#christmascelebration	cheerful
5	7	9	500	1	everybody	#view	touristy

8. location

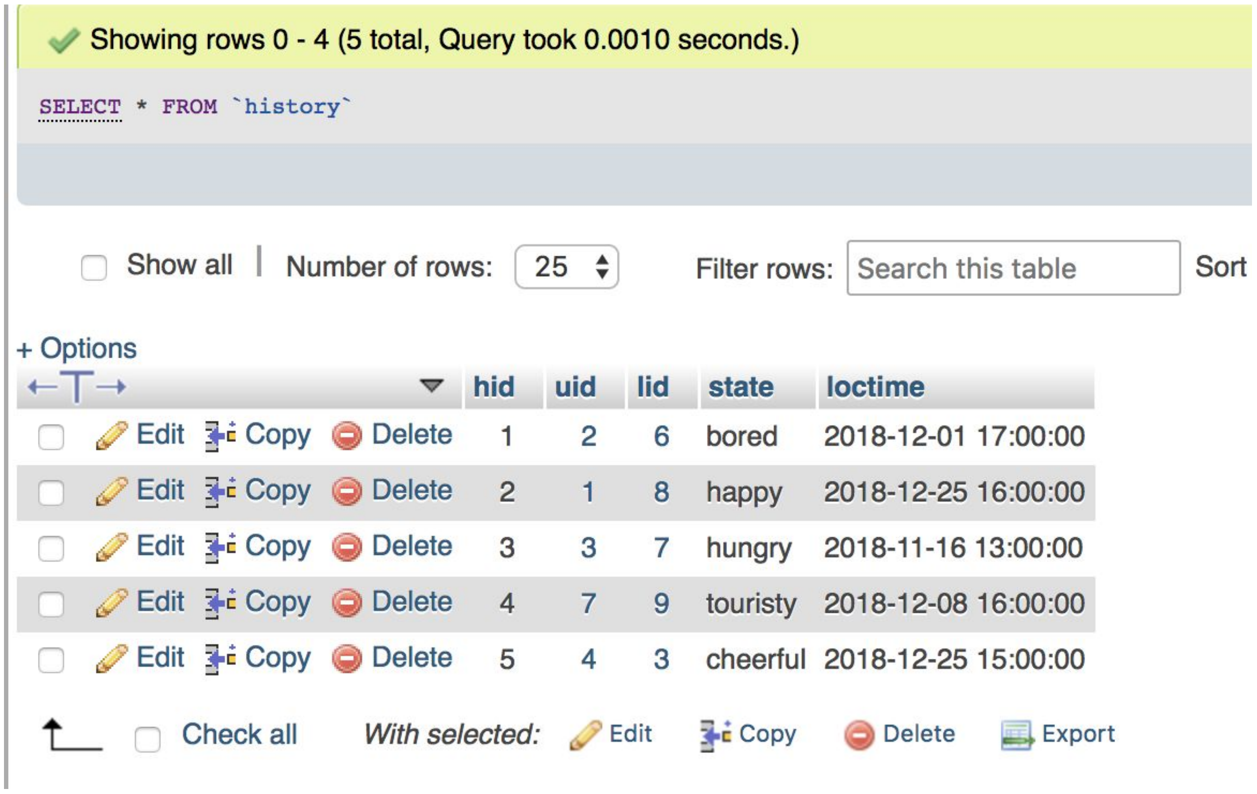


Showing rows 0 - 8 (9 total, Query took 0.0008 seconds.)

```
SELECT * FROM `location`
```

lid	lname	lat	lon
1	SOHO	40.72001	73.99000
2	Jackson Heights	40.75125	73.88000
3	NYU tandon	40.69000	73.98500
4	Washington Square Park	40.73000	73.99000
5	Exchange Place	40.70600	74.01100
6	SOHO_near	40.72003	73.99000
7	Jackson Heights_near	40.75127	73.88000
8	2 metro Tech_NYU tandon	40.69001	73.98500
9	Exchange Place Harbour	40.70601	74.01101

9. history

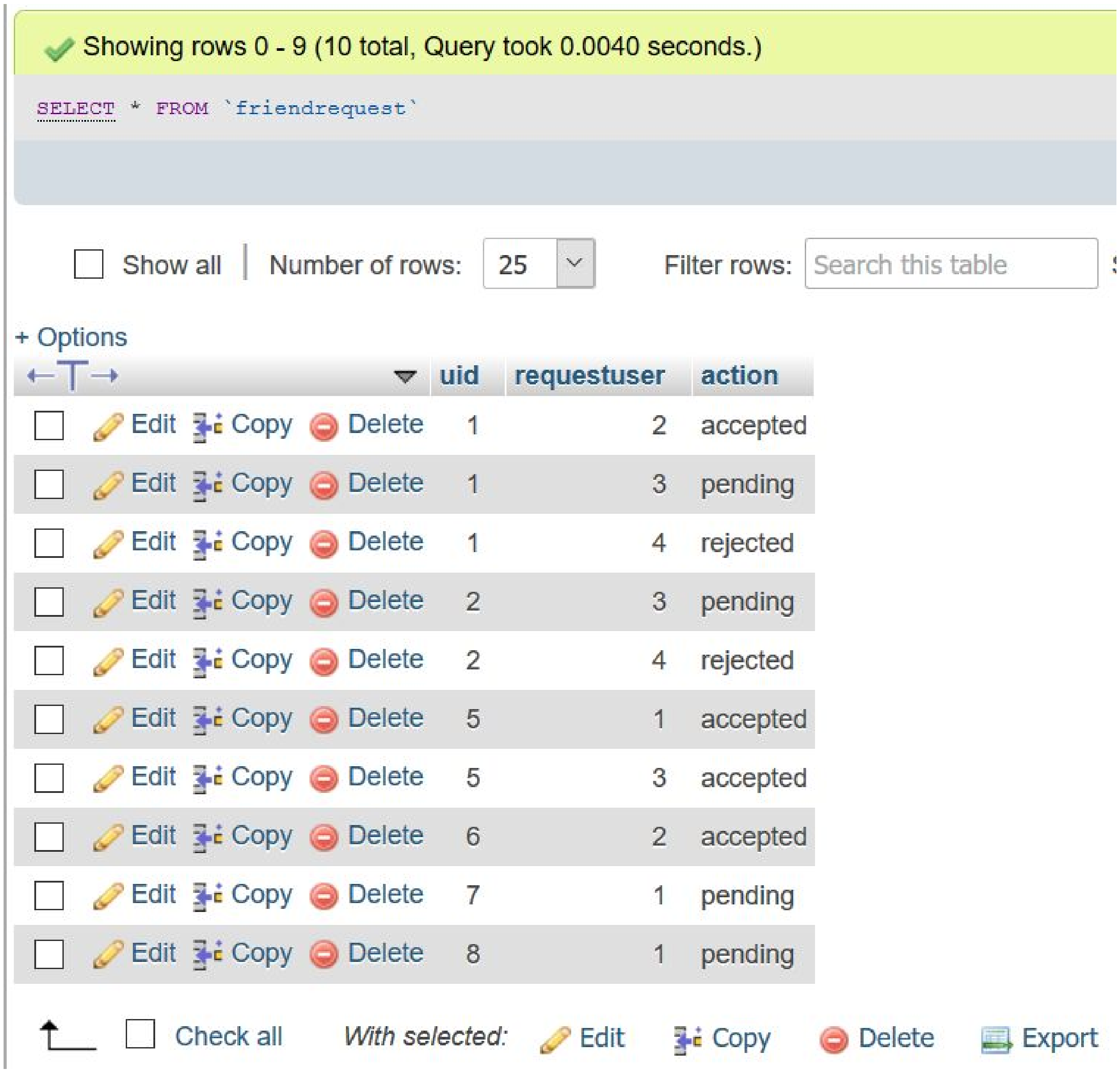


Showing rows 0 - 4 (5 total, Query took 0.0010 seconds.)

```
SELECT * FROM `history`
```

	hid	uid	lid	state	loctime
Edit Copy Delete	1	2	6	bored	2018-12-01 17:00:00
Edit Copy Delete	2	1	8	happy	2018-12-25 16:00:00
Edit Copy Delete	3	3	7	hungry	2018-11-16 13:00:00
Edit Copy Delete	4	7	9	touristy	2018-12-08 16:00:00
Edit Copy Delete	5	4	3	cheerful	2018-12-25 15:00:00

10. friendrequest **(added)**


Showing rows 0 - 9 (10 total, Query took 0.0040 seconds.)

```
SELECT * FROM `friendrequest`
```

uid	requestuser	action
1	2	accepted
1	3	pending
1	4	rejected
2	3	pending
2	4	rejected
5	1	accepted
5	3	accepted
6	2	accepted
7	1	pending
8	1	pending

Default Values

Column	Default value
class	everybody
commentenabled	true
text	Hello, world
tag	happy

Sample of supported schedules

1. Next half hour
2. On 21st dec, 2019
3. Every friday between 5pm and 7pm
4. Always
5. Everyday between 11pm and 12am
6. 2pm to 3pm on 31st Dec, 2018
7. From 23rd Dec to 25th Dec from 5pm to 9pm **(Now Supported)**

Description of design (includes assumptions)

Users and notes

1. A user can author multiple notes.
2. A note can be authored by a single user.
3. There are few users in the system who haven't yet written any notes.

Notes and comments

4. A note can have several comments on it.
5. A specific comment is only linked to a single specific note.
6. Comment cannot exist without a note. Hence comment table has total participation in relation (is for) with note table.
7. A comment cannot exist without a user who posts it. Hence comment table has total participation in relation (post) with user table.
8. A user can write multiple comments.
9. A comment can only be written by a single user.

Friendship

10. (Friend) is a recursive relationship set. A user can have several other users that are related to him as friends.

Notes and Schedule

11. A note is associated to a single schedule.
12. A single schedule can be there for several notes.
13. A note cannot exist without a schedule.

Notes and Location

14. A note is associated to a single location.
15. A specific location can be there for several notes.
16. A note cannot exist without a location hence note has total participation in (is for).

History

17. A history tuple is associated to a single location.
18. A specific location can be there for several history tuples.
19. A history tuple cannot exist without a location hence history has total participation in (is linked to).
20. A history tuple is associated to a single user.
21. A specific user can be there in several history tuples.

22. A history tuple cannot exist without a user hence history has total participation in (is of).

Filter

23. A filter can be associated with a single location and/or a single schedule.
24. A filter cannot exist without a user (who uploads it) hence filter has total participation in (uploaded by).
25. A filter is uploaded by a single user.
26. A user can upload multiple filters.

Notes/Filters and tags

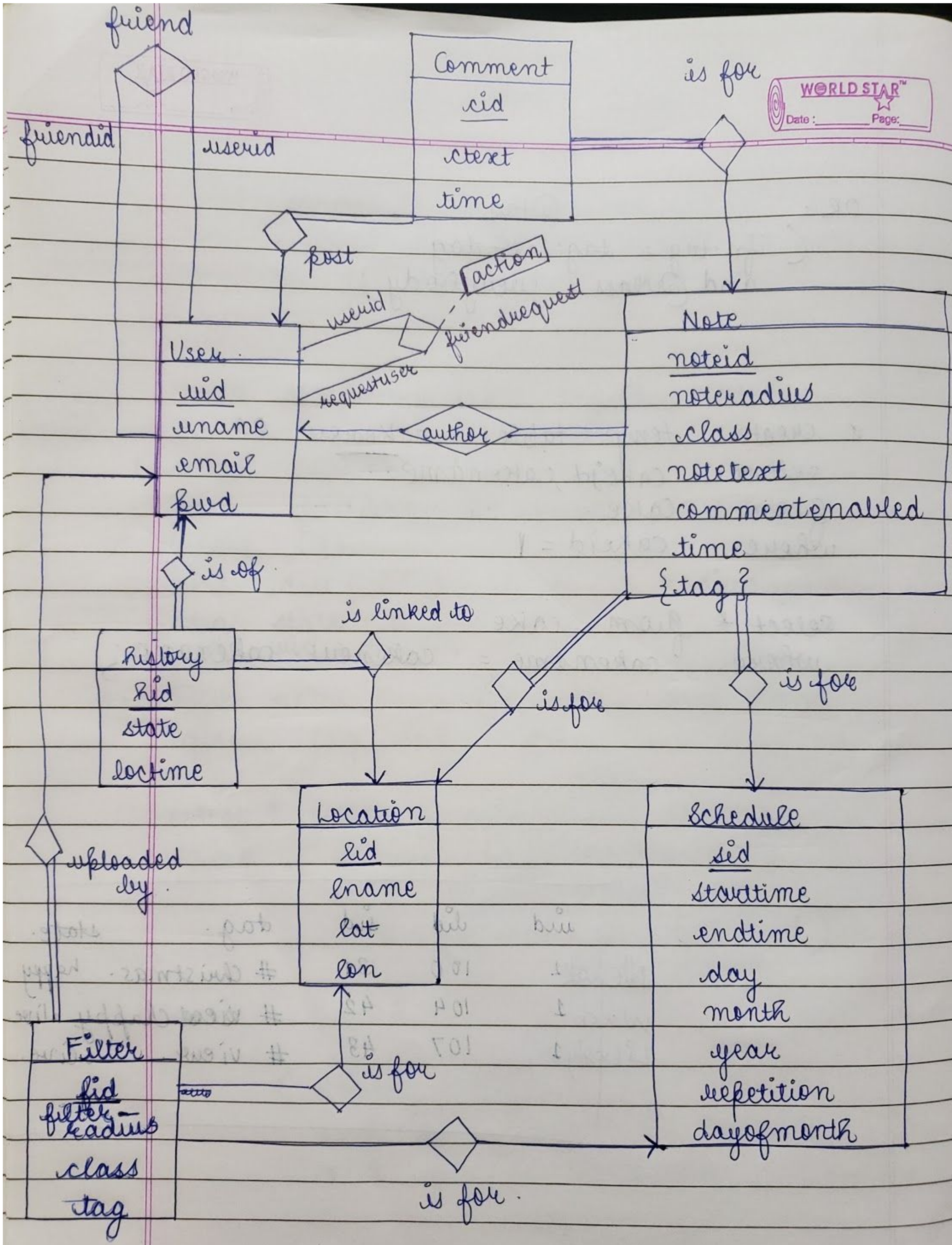
27. A note can have several tags associated with it.
28. A filter can have only one tag associated with it for simplicity.

Functions

We are using function checkdistance() to calculate distance between two locations whose latitude and longitude are given. This function returns distance in yards.
Reference : stackoverflow

```
DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `checkdistance` (`lat1` FLOAT(9,5), `lon1` FLOAT(9,5),
`lat2` FLOAT(9,5), `lon2` FLOAT(9,5)) RETURNS FLOAT NO SQL
        DETERMINISTIC
        COMMENT 'Returns the distance in degrees on the Earth between two known points of latitude and
longitude. To get miles, multiply by 3961, and km by 6373'
BEGIN

        RETURN (1093*6373 *DEGREES(ACOS(
        COS(RADIANS(lat1)) *
        COS(RADIANS(lat2)) *
        COS(RADIANS(lon2) - RADIANS(lon1)) +
        SIN(RADIANS(lat1)) * SIN(RADIANS(lat2))
        )));

END$$

DELIMITER ;
```

Entity Relationship Diagram (modified)


friend
friendid
userid
Comment
cid
ctext
time
is for
post
action
userid
friendrequest
requestuser
User
uid
uname
email
pwd
author
Note
noteid
noteradius
class
notetext
commentenabled
time
{tag}
is of
is linked to
history
hid
state
loctime
is for
is for
Location
lid
lname
lat
lon
Schedule
sid
starttime
endtime
day
month
year
repetition
dayofmonth
uploaded by
Filter
fid
filter-radius
class
tag
is for
is for.

Limitations of schedules

Schema does not support schedules like the following:

1. Every mondays and thursdays from 5pm to 7pm.
2. Easter Sunday
3. Second thursday of each month

Using the oingo App

We have login page and sign-up page for existing users and new users respectively.

When a person is registering following are checked:

1. Username is unique
2. Email is unique
3. Password and Confirm Password Match

A logged in user has the option to

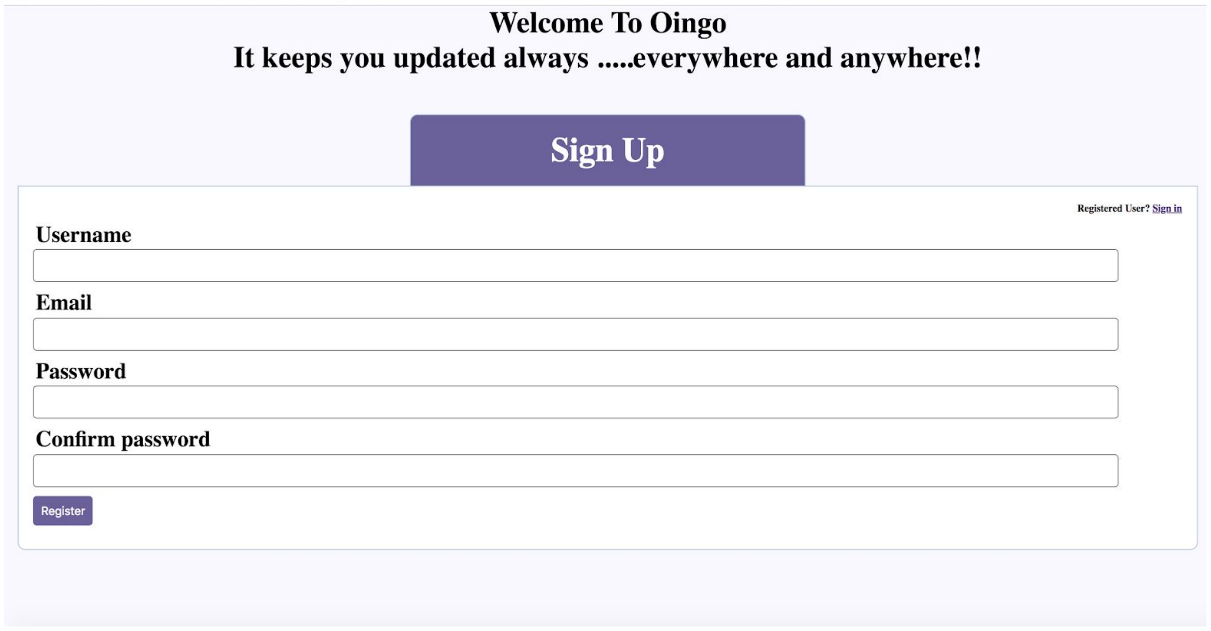
1. View notes and post comment on them is comments are enabled.
2. Add notes for a location and schedule,
3. add filter,
4. view users using the oingo app,
5. See friend requests(if any) and accept/reject them,
6. update his/her profile.

User sees notes on the homepage based on the filters he/she has added and based on the notes visible at that time and at that location.

User can select any location on the map, and the app will automatically pick up the coordinates for that location.

Application Screens

1. **Sign Up page**



Welcome To Oingo
It keeps you updated always .....everywhere and anywhere!!

Sign Up

Registered User? Sign in

Username

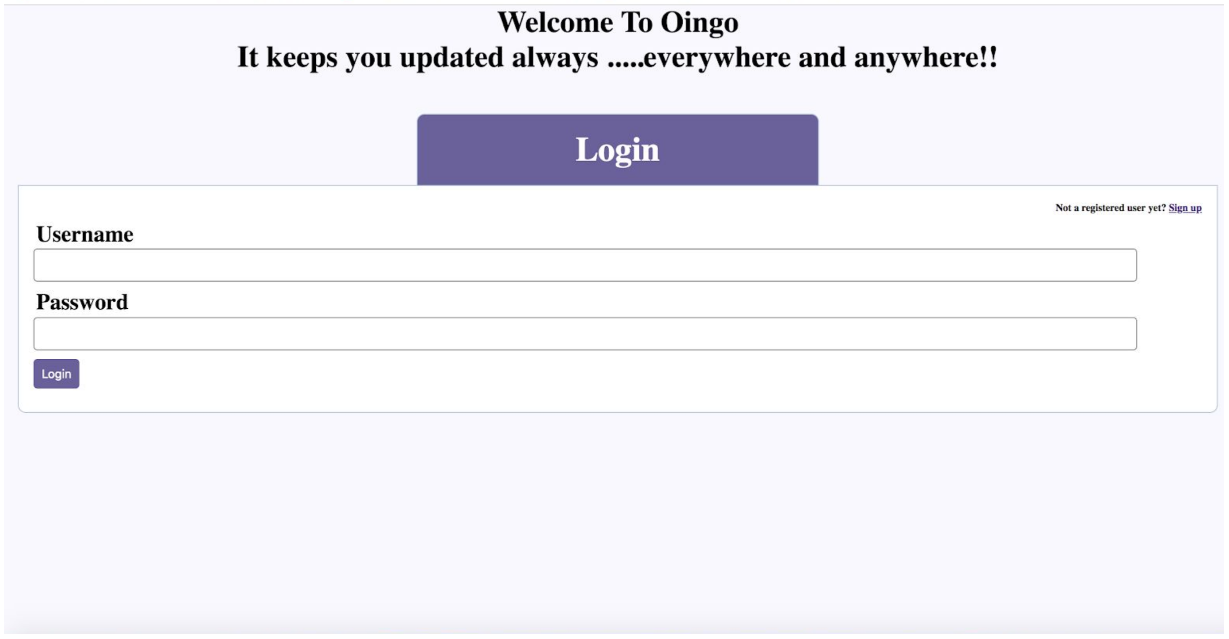
Email

Password

Confirm password

Register

2. **Login page**



Welcome To Oingo
It keeps you updated always .....everywhere and anywhere!!

Login

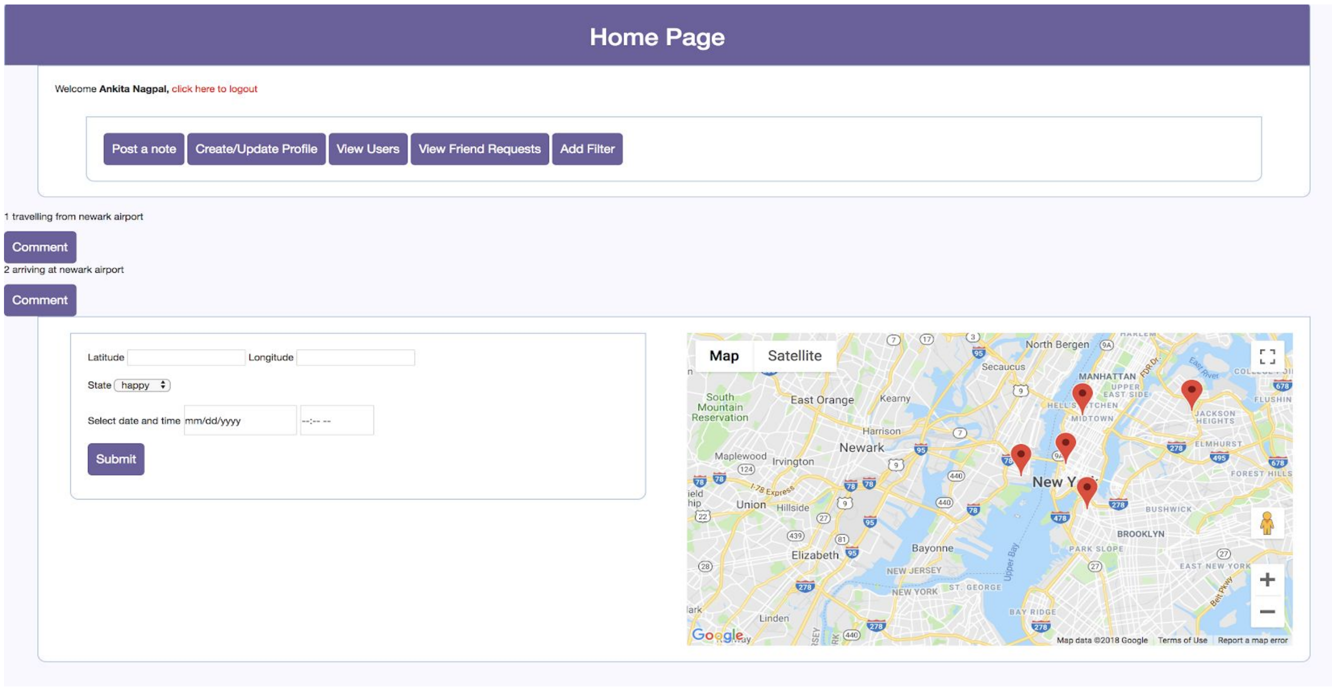
Not a registered user yet? Sign up

Username

Password

Login

3. Homepage



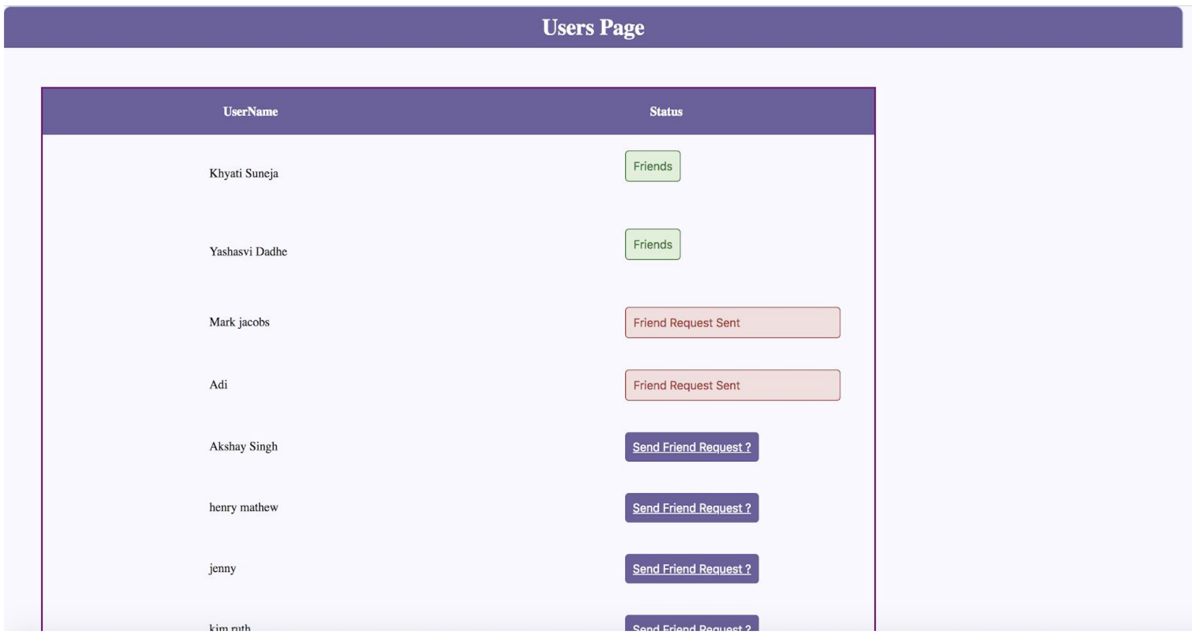
Home Page

Welcome **Ankita Nagpal,** click here to logout

1 travelling from newark airport

2 arriving at newark airport

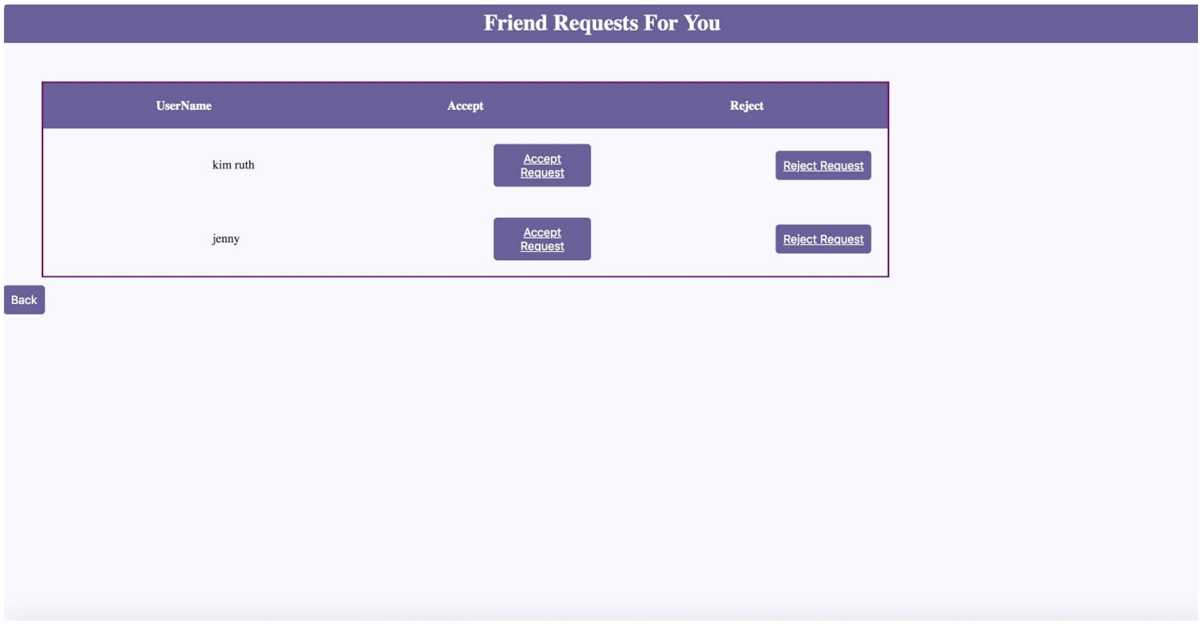
4. View Users



Users Page

UserName	Status
Khyati Suneja	Friends
Yashasvi Dadhe	Friends
Mark jacobs	Friend Request Sent
Adi	Friend Request Sent
Akshay Singh	Send Friend Request ?
henry mathew	Send Friend Request ?
jenny	Send Friend Request ?
kim ruth	Send Friend Request ?

5. View Friend Requests

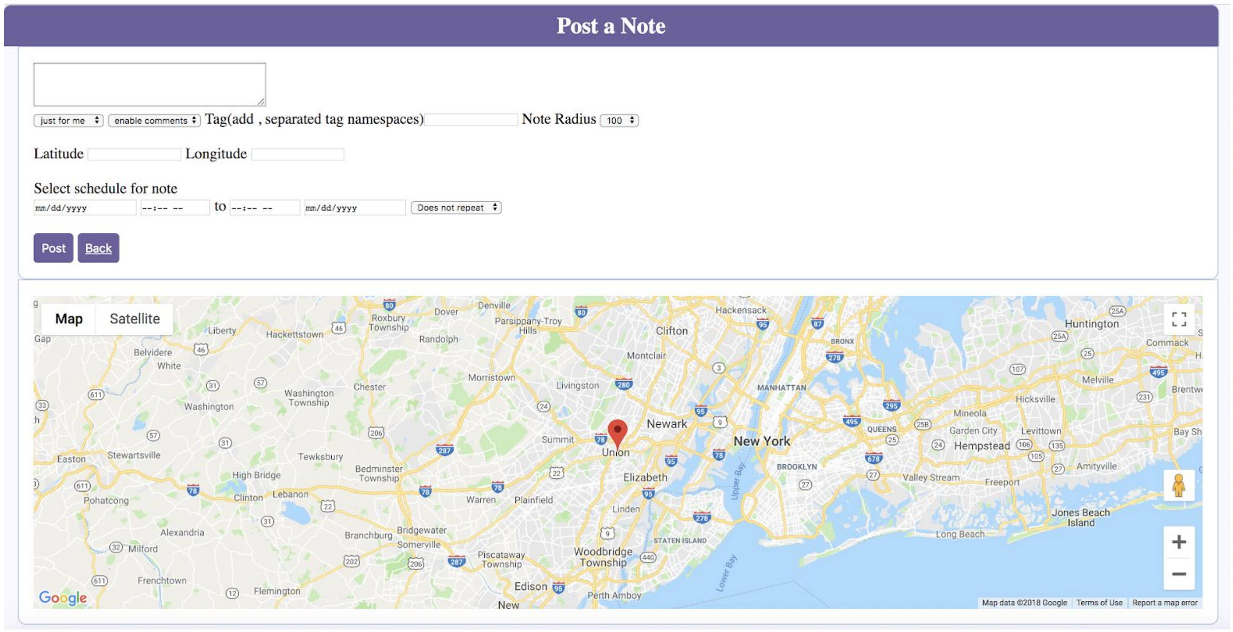


Friend Requests For You

UserName	Accept	Reject
kim ruth	Accept Request	Reject Request
jenny	Accept Request	Reject Request

Back

6. Post a note



Post a Note

just for me | enable comments | Tag(add , separated tag namespaces) | Note Radius 100

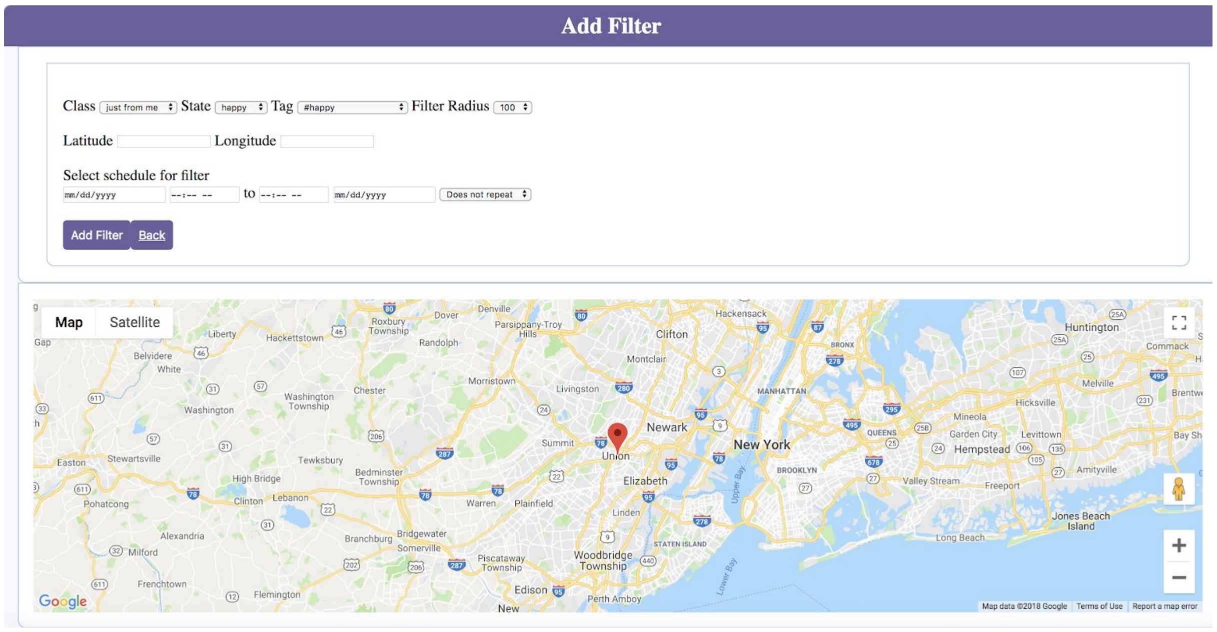
Latitude | Longitude

Select schedule for note

mm/dd/yyyy --:-- -- to --:-- -- mm/dd/yyyy | Does not repeat

Post | Back

7. Add a filter



Add Filter

Class just from me State happy Tag #happy Filter Radius 100

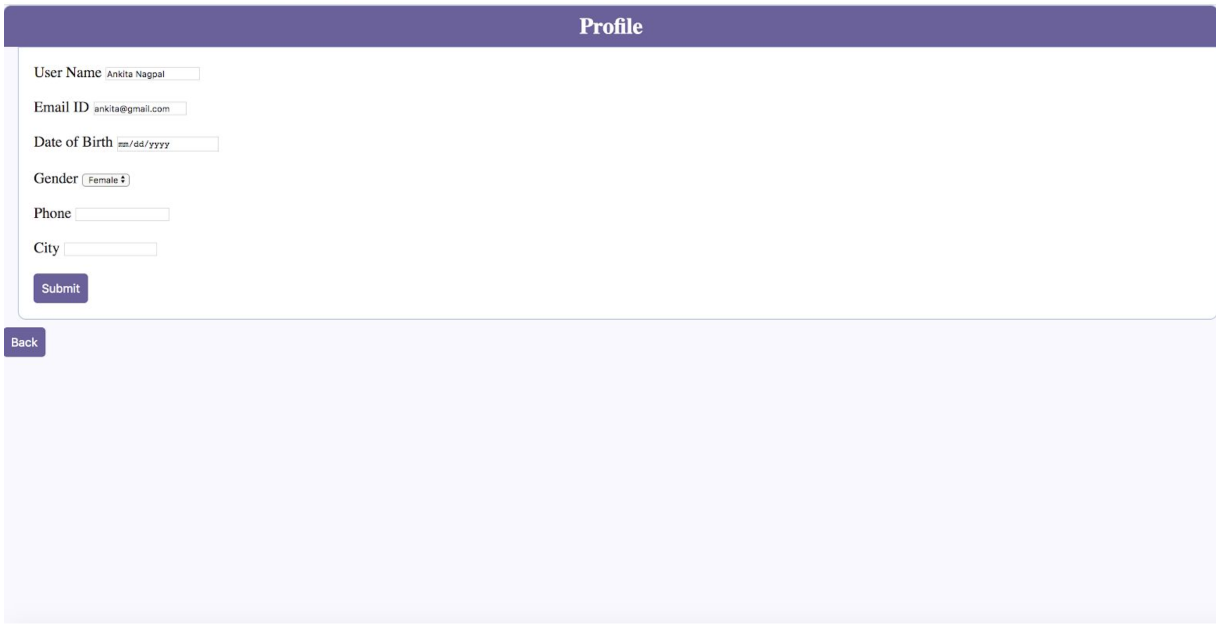
Latitude Longitude

Select schedule for filter

mm/dd/yyyy --:-- -- to --:-- -- mm/dd/yyyy Does not repeat

Add Filter Back

8. Create/Modify Profile



Profile

User Name Ankita Nagpal

Email ID ankita@gmail.com

Date of Birth mm/dd/yyyy

Gender Female

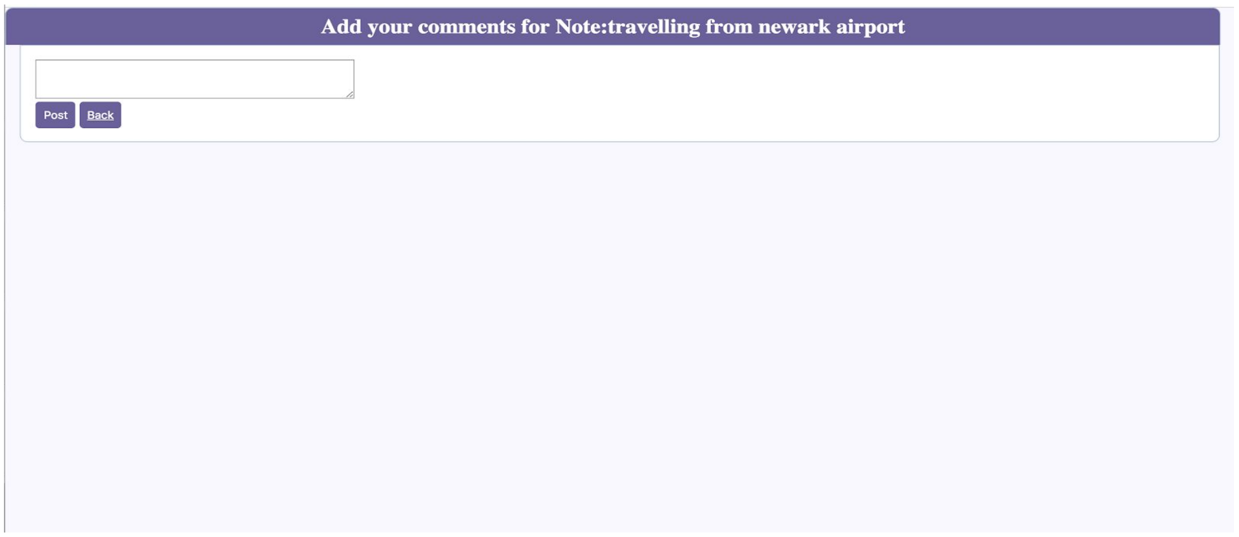
Phone

City

Submit

Back

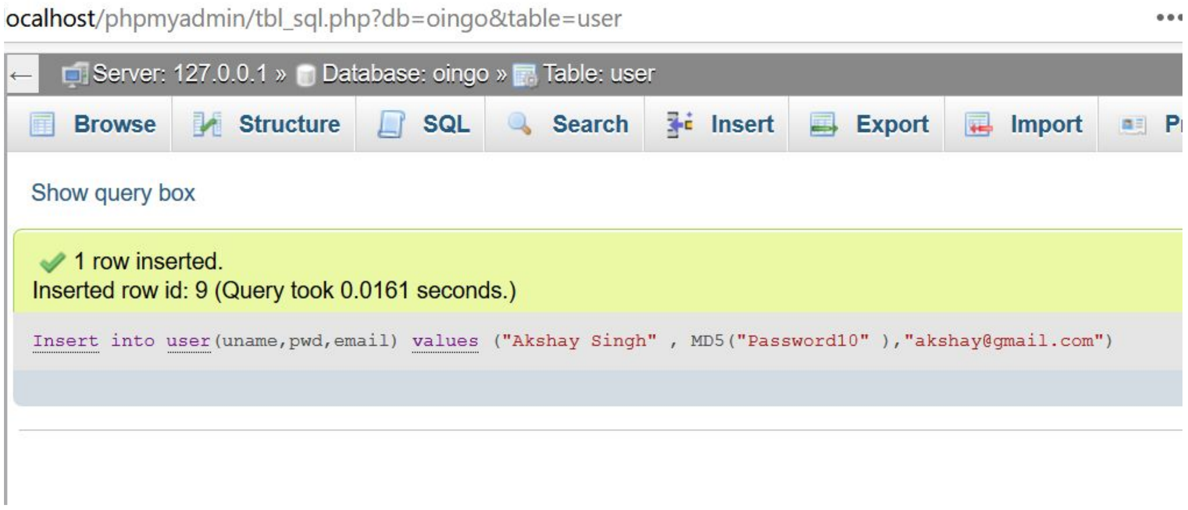
9. Add comment



Queries

1. Create a new user account, with name, login, and password

   Insert into user(uname,pwd,email) values ("Akshay Singh" , MD5("Password10"),"akshay@gmail.com")

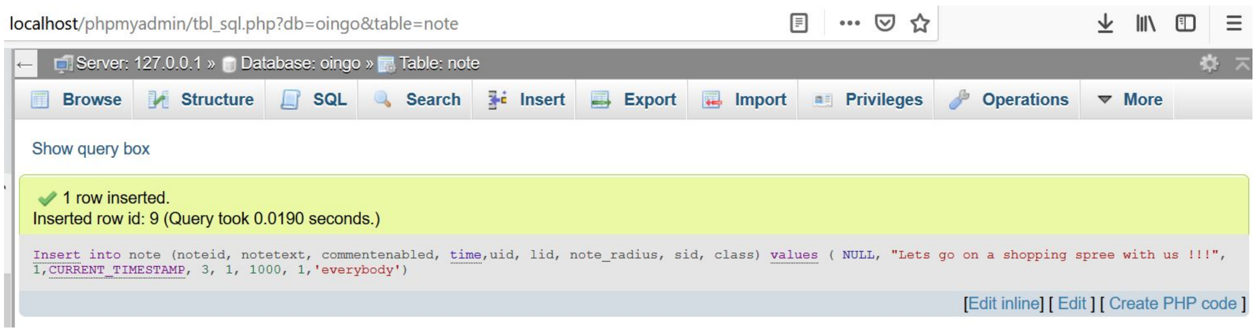

ocalhost/phpmyadmin/tbl_sql.php?db=oingo&table=user
Server: 127.0.0.1 » Database: oingo » Table: user
Browse
Structure
SQL
Search
Insert
Export
Import
Show query box

✓ 1 row inserted.
Inserted row id: 9 (Query took 0.0161 seconds.)

```
Insert into user(uname,pwd,email) values ("Akshay Singh" , MD5("Password10" ),"akshay@gmail.com")
```

2. Add a new note to the system, together with tags, and spatial and temporal constraints.

   Insert into note (noteid, notetext, commentenabled, time,uid, lid, note_radius, sid, class) values (NULL, "Lets go on a shopping spree with us !!!", 1,CURRENT_TIMESTAMP, 3, 1, 1000, 1,'everybody');



localhost/phpmyadmin/tbl_sql.php?db=oingo&table=note

Server: 127.0.0.1 » Database: oingo » Table: note

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | More

Show query box

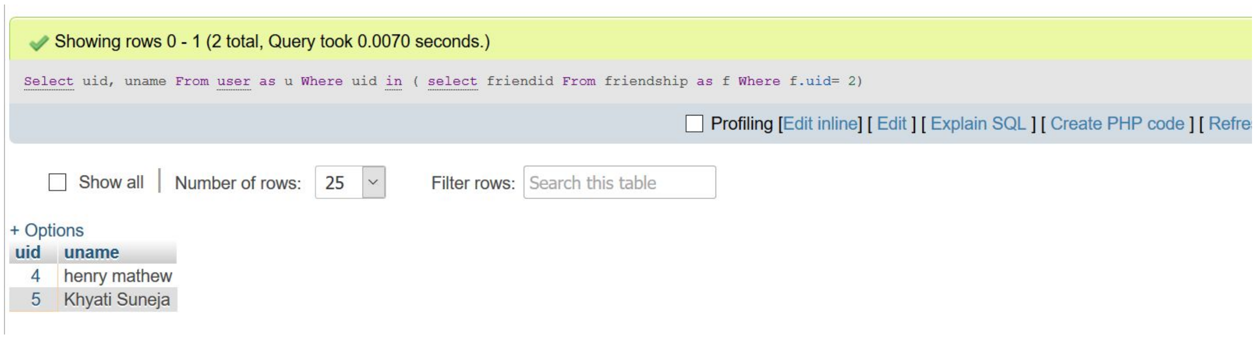
```
1 row inserted.
Inserted row id: 9 (Query took 0.0190 seconds.)
```

```
Insert into note (noteid, notetext, commentenabled, time,uid, lid, note_radius, sid, class) values ( NULL, "Lets go on a shopping spree with us !!!",
1,CURRENT_TIMESTAMP, 3, 1, 1000, 1,'everybody')
```

[Edit inline] [Edit] [Create PHP code]

3. For a given user, list all her friends.
   If we are given the user's id , Let the id of given user be 2:

   Select uid, uname
   From user as u
   Where uid in (select friendid
                  From friendship as f
                  Where f.uid= 2)



```
Showing rows 0 - 1 (2 total, Query took 0.0070 seconds.)
```

```
Select uid, uname From user as u Where uid in ( select friendid From friendship as f Where f.uid= 2)
```

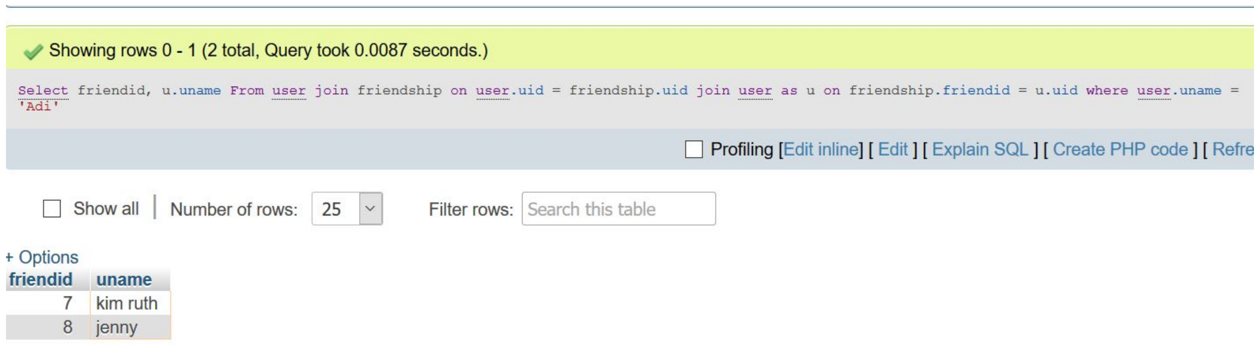
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refre

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

uid	uname
4	henry mathew
5	Khyati Suneja

If we are given user name , Let the name of given user be Adi:
Select friendid, u.uname
From user join friendship on user.uid = friendship.uid
join user as u on friendship.friendid = u.uid
where user.uname = 'Adi'



```
Showing rows 0 - 1 (2 total, Query took 0.0087 seconds.)
```

```
Select friendid, u.uname From user join friendship on user.uid = friendship.uid join user as u on friendship.friendid = u.uid where user.uname =
'Adi'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refre

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

friendid	uname
7	kim ruth
8	jenny

4. Given a user and her current location, current time, and current state, output all notes that she should currently be able to see given the filters she has set up.

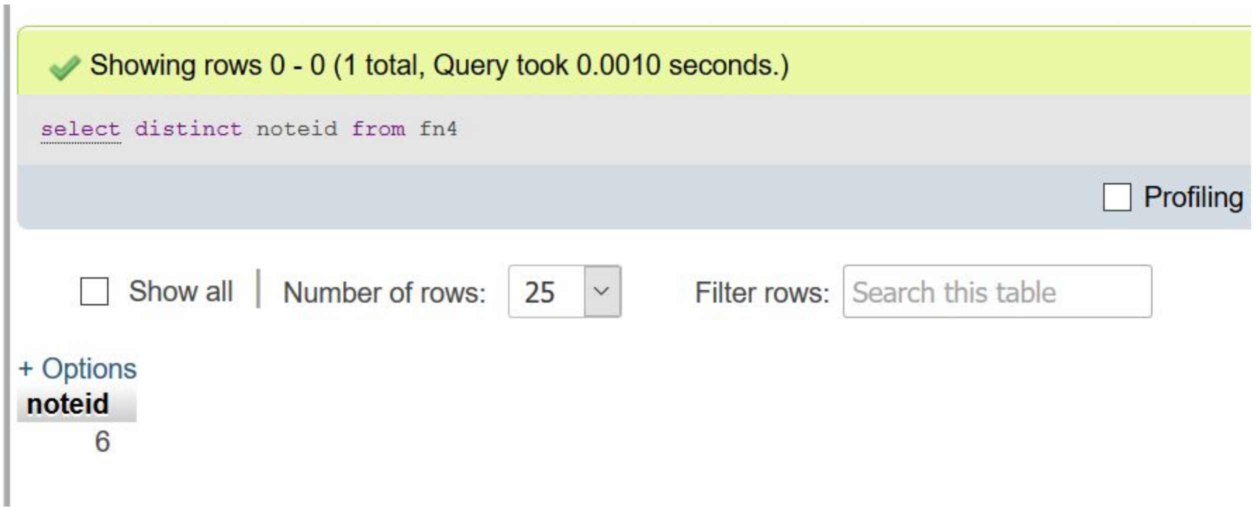
Input:
Userid : 7
Current location: Exchange Place Harbour
Current time: 2018-11-24 15:30:00
Current state: alive

Query:
```
set @l1 := (select lat from location where lname = 'Exchange Place Harbour');
set @l2 := (select lon from location where lname = 'Exchange Place Harbour');
set @ctime = '2018-12-29 14:30:00';
set @cstate = 'alive';
set @uid = 7;

create temporary table vn
select * from note natural join location natural join schedule
where checkdistance(@l1,@l2,lat,lon) <= note_radius
and
(dayname(@ctime) = day or day = 'All' ) and (time(@ctime) <=endtime and time(@ctime) >=starttime) and (
(month(@ctime) = month and year(@ctime) = year and
  dayofmonth(@ctime)=dayofmonth) or repetition = 'Yes'  );

create TEMPORARY table fn1
select * from vn natural join friendship
where vn.class= 'self'
and vn.uid = @uid;

create TEMPORARY table fn2
select * from vn natural join friendship
where vn.class= 'friend'
and friendship.friendid = @uid;
create TEMPORARY table fn3
select * from vn natural join friendship
where vn.class= 'everybody';

create temporary table fn4
select * from fn1 union (select * from fn2) union (select * from fn3);

Set @tag = (select tag from userfilter NATURAL JOIN location NATURAL JOIN schedule
where checkdistance(@l1,@l2,lat,lon) <= filter_radius
and(dayname(@ctime) = day or day = 'All' ) and (time(@ctime) <=endtime and time(@ctime) >=starttime) and
( (month(@ctime) = month and year(@ctime) = year and
  dayofmonth(@ctime)=dayofmonth) or repetition = 'Yes'  ) and uid = @uid and state = @cstate);
```

Set @class = (select class from userfilter NATURAL JOIN location NATURAL JOIN schedule
where checkdistance(@l1,@l2,lat,lon) <= filter_radius
and(dayname(@ctime) = day or day = 'All') and (time(@ctime) <=endtime and time(@ctime) >=starttime) and
((month(@ctime) = month and year(@ctime) = year and
  dayofmonth(@ctime)=dayofmonth) or
   repetition = 'Yes') and uid = @uid and state = @cstate);

 select * from fn4 NATURAL JOIN notetag
   where tag = @tag and @class ='self' and uid = @uid
   or tag = @tag and @class ='friend' and friendid = @uid
   or tag = @tag and @class ='everybody';

select distinct noteid from fn4;



✓ Showing rows 0 - 0 (1 total, Query took 0.0010 seconds.)

```
select distinct noteid from fn4
```

```
noteid
     6
```

5. Given a note (that maybe was just added to the system) and the current time, output all users that should currently be able to see this note based on their filter and their last recorded location.

Input:
noteid =7 ,node tag = #shopping , node author = 4 and current time =2018-11-24 16:00:00

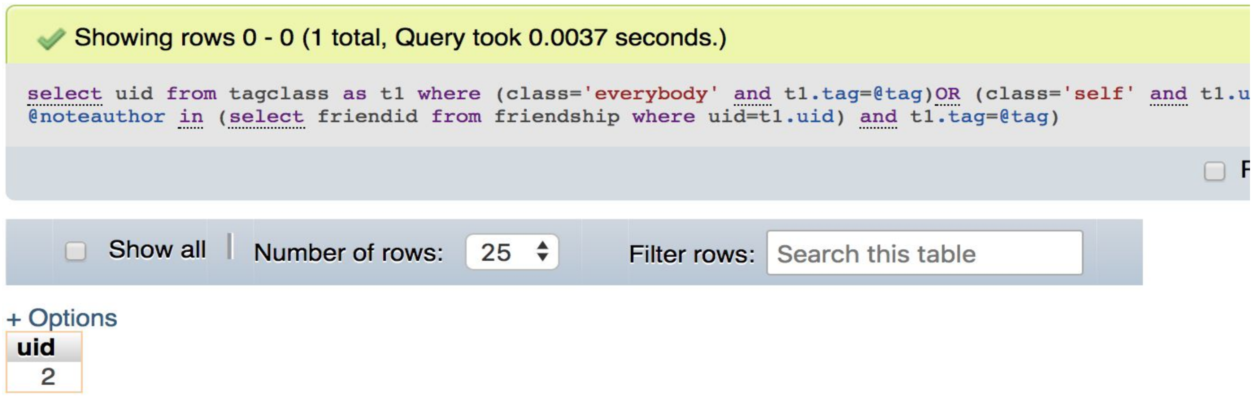
Query:
set @ctime = '2018-11-24 16:00:00 ';
set @givennote = 7;
set @tag = '#shopping';
set @noteauthor= 4;

Create Temporary Table latest_location_users
SELECT uid, lid,lat,lon,state, loctime
FROM history natural join location

```
WHERE (uid, loctime )
IN ( SELECT uid, MAX( loctime )
   FROM history GROUP BY uid );
```

```
Create Temporary Table tagclass Select * From latest_location_users as t1 natural join userfilter as t2
NATURAL join location as t3 natural join schedule where checkdistance(t1.lat,t1.lon,t3.lat,t3.lon) <=
filter_radius and (dayname(loctime) = day or day = 'All' ) and (time(loctime) <=endtime and time(loctime)
>=starttime) and ( (month(loctime) = month and year(loctime) = year and dayofmonth(loctime)=dayofmonth) or
repetition = 'Yes' ) and t1.state = t2.state;
```

```
select *
from tagclass as t1
where (class='everybody' and t1.tag=@tag)OR
(class='self' and t1.uid=@noteauthor and t1.tag=@tag) OR
(class='friend' and @noteauthor in (select friendid from friendship where uid=t1.uid) and t1.tag=@tag);
```



Showing rows 0 - 0 (1 total, Query took 0.0037 seconds.)

```
select uid from tagclass as t1 where (class='everybody' and t1.tag=@tag)OR (class='self' and t1.u
@noteauthor in (select friendid from friendship where uid=t1.uid) and t1.tag=@tag)
```

uid
2

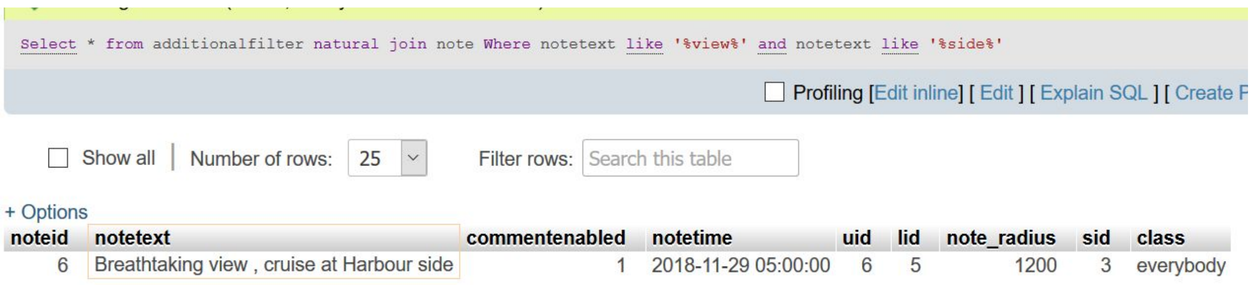
6. In some scenarios, in very dense areas or when the user has defined very general filters, there may be a lot of notes that match the current filters for a user. Write a query showing how the user can further filter these notes by inputting one or more keywords that are matched against the text in the notes using the contains operator.

Input:
Keyword1 = ‘view’
Keyword2 = ‘side’

Query:
Execute queries in 5.

```
Create temporary table additionalfilter
    Select distinct noteid from fn4;
```

```
 Select * from additionalfilter natural join note Where notetext like '%view%' and notetext like '%side%'
```



```
Select * from additionalfilter natural join note Where notetext like '%view%' and notetext like '%side%'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create P

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

noteid	notetext	commentenabled	notetime	uid	lid	note_radius	sid	class
6	Breathtaking view , cruise at Harbour side	1	2018-11-29 05:00:00	6	5	1200	3	everybody