COMP 7115 Database Systems

Instructor: Fatih Șen Project

Final Project

Name: Rahul Marru work by team up with Manivardhan reddy pindi

UID: U00843883

1. Overview:

The aim of this project is to design and implement a simple Twitter-like social networking platform using both MySQL and Neo4j graph databases. Students are expected to gain an understanding of all the steps involved in using both relational and graph databases for a practical application.

2. Requirements:

• Each user can sign up with their name, lastname, and email.

• A user may choose to “follow" other users.

• A user may post a feed. • Each feed might have zero or more comments.

• A feed may be liked by a user.

• A feed may be re-tweeted.

• A comment or a feed should be able to deleted by the owner when needed.

• There is no restriction of the number of characters for a feed or comment.

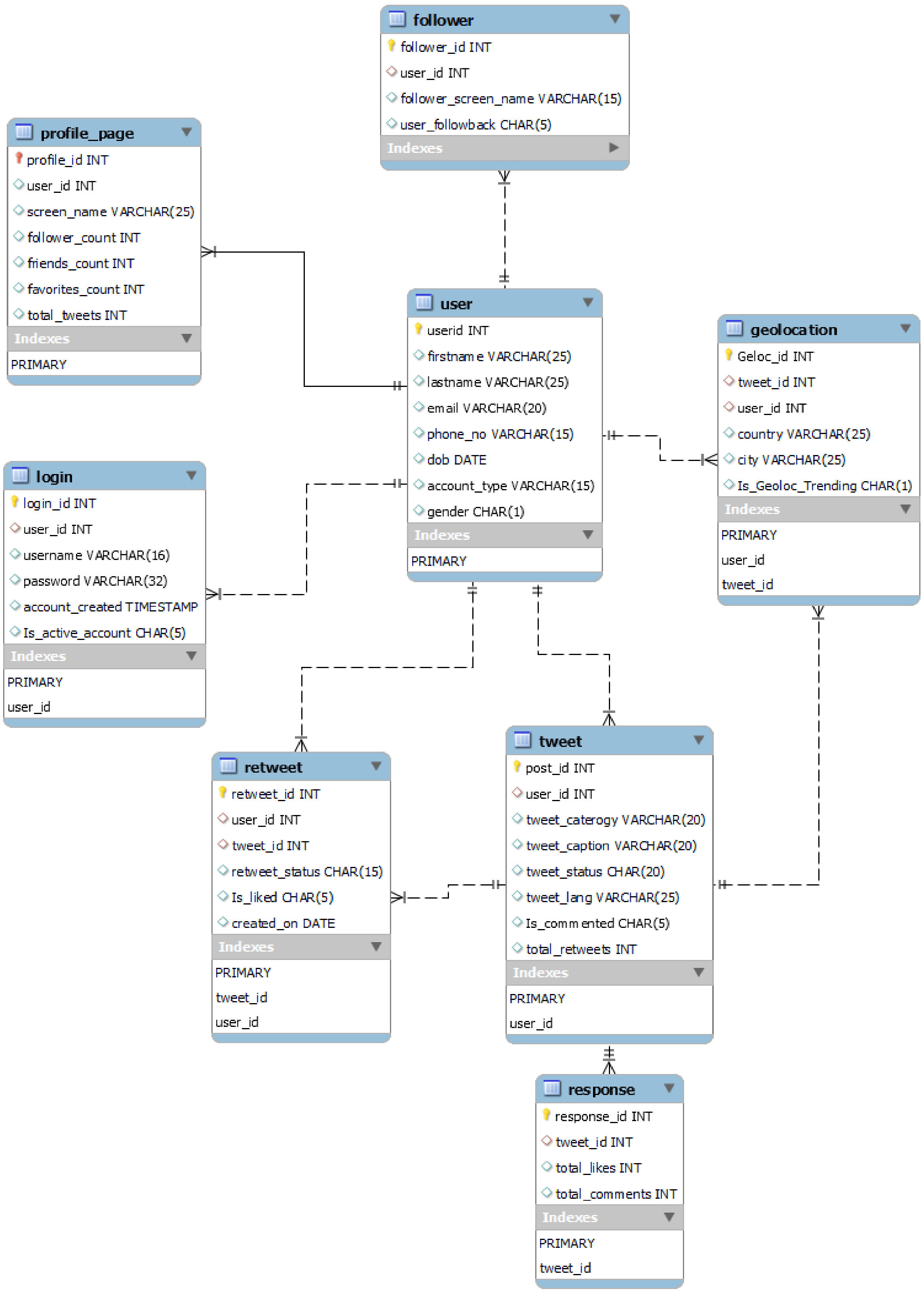
• Login page – a person should be able to sign up.

• Main page – displays news feeds according to the network of a user (his/her feeds and people he/she follows).

• Profile page – list user’s profile information, the people he/she follows and feeds of that user.

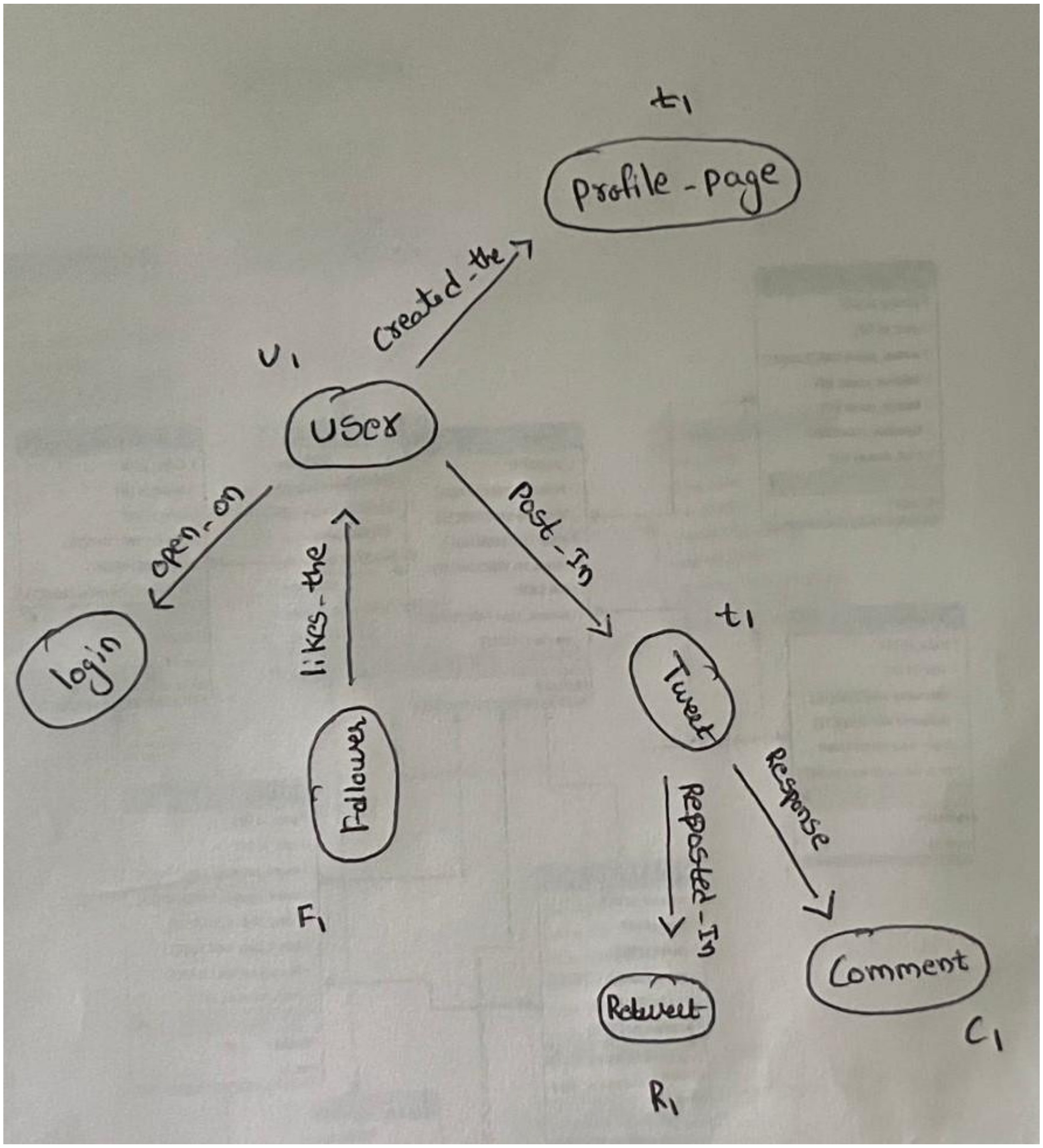
3. MySQL Database ER-Diagram:

The ER diagram used to create the model for this project is as below,

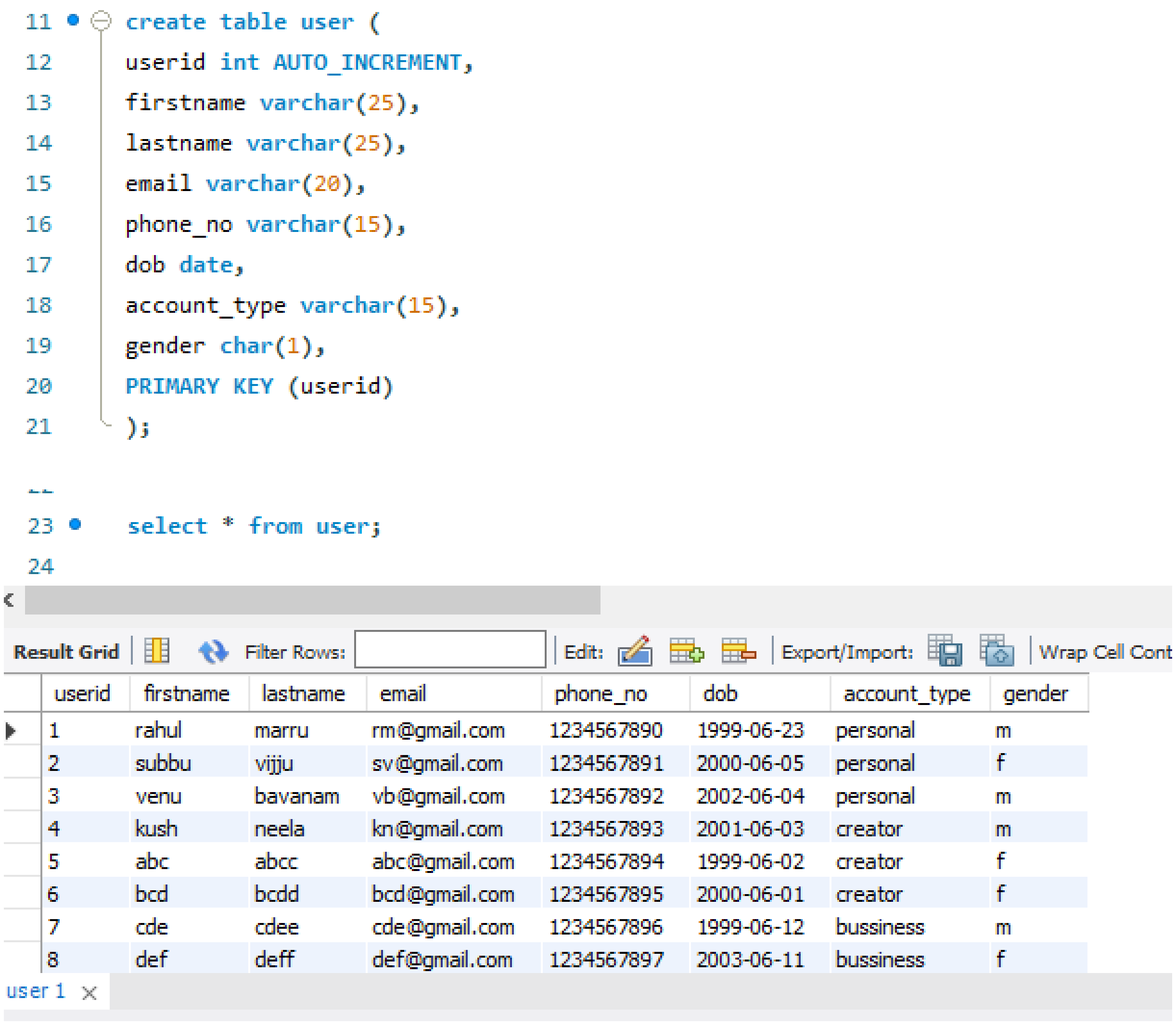

follower
follower_id INT
user_id INT
follower_screen_name VARCHAR(15)
user_followback CHAR(5)
Indexes
profile_page
profile_id INT
user_id INT
screen_name VARCHAR(25)
follower_count INT
friends_count INT
favorites_count INT
total_tweets INT
Indexes
PRIMARY
user
userid INT
firstname VARCHAR(25)
lastname VARCHAR(25)
email VARCHAR(20)
phone_no VARCHAR(15)
dob DATE
account_type VARCHAR(15)
gender CHAR(1)
Indexes
PRIMARY
geolocation
Geloc_id INT
tweet_id INT
user_id INT
country VARCHAR(25)
city VARCHAR(25)
Is_Geoloc_Trending CHAR(1)
Indexes
PRIMARY
user_id
tweet_id
login
login_id INT
user_id INT
username VARCHAR(16)
password VARCHAR(32)
account_created TIMESTAMP
Is_active_account CHAR(5)
Indexes
PRIMARY
user_id
retweet
retweet_id INT
user_id INT
tweet_id INT
retweet_status CHAR(15)
Is_liked CHAR(5)
created_on DATE
Indexes
PRIMARY
tweet_id
user_id
tweet
post_id INT
user_id INT
tweet_caterogy VARCHAR(20)
tweet_caption VARCHAR(20)
tweet_status CHAR(20)
tweet_lang VARCHAR(25)
Is_commented CHAR(5)
total_retweets INT
Indexes
PRIMARY
user_id
response
response_id INT
tweet_id INT
total_likes INT
total_comments INT
Indexes
PRIMARY
tweet_id]

4. Twitter Database Graph Diagram:

The Graph diagram used to create the model for this project is as below,

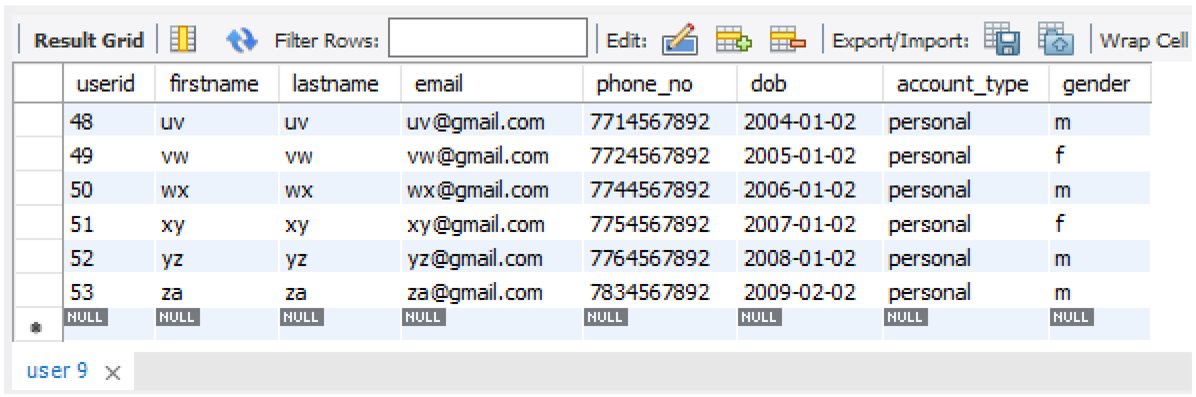


User Table: User table stores the individual user personal details and account type. User table consists of total 53 records.



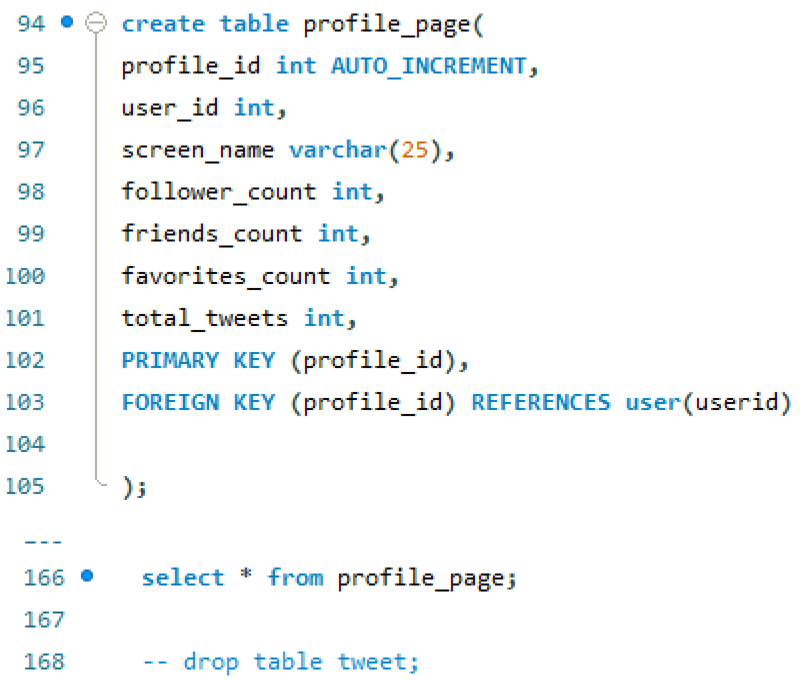
```
create table user (
userid int AUTO_INCREMENT,
firstname varchar(25),
lastname varchar(25),
email varchar(20),
phone_no varchar(15),
dob date,
account_type varchar(15),
gender char(1),
PRIMARY KEY (userid)
);

select * from user;
```

userid	firstname	lastname	email	phone_no	dob	account_type	gender
1	rahul	marru	rm@gmail.com	1234567890	1999-06-23	personal	m
2	subbu	vijju	sv@gmail.com	1234567891	2000-06-05	personal	f
3	venu	bavanam	vb@gmail.com	1234567892	2002-06-04	personal	m
4	kush	neela	kn@gmail.com	1234567893	2001-06-03	creator	m
5	abc	abcc	abc@gmail.com	1234567894	1999-06-02	creator	f
6	bcd	bcdd	bcd@gmail.com	1234567895	2000-06-01	creator	f
7	cde	cdee	cde@gmail.com	1234567896	1999-06-12	bussiness	m
8	def	deff	def@gmail.com	1234567897	2003-06-11	bussiness	f

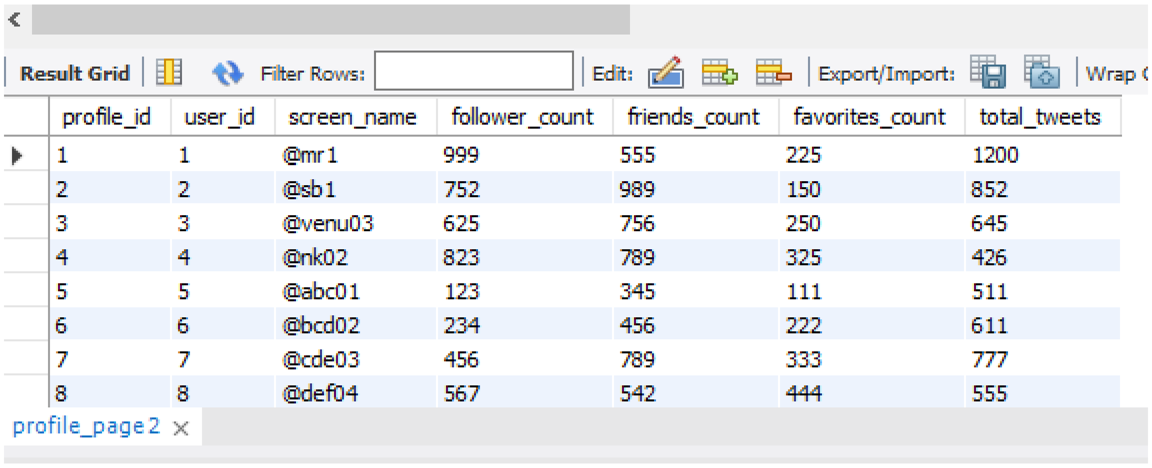


userid	firstname	lastname	email	phone_no	dob	account_type	gender
48	uv	uv	uv@gmail.com	7714567892	2004-01-02	personal	m
49	vw	vw	vw@gmail.com	7724567892	2005-01-02	personal	f
50	wx	wx	wx@gmail.com	7744567892	2006-01-02	personal	m
51	xy	xy	xy@gmail.com	7754567892	2007-01-02	personal	f
52	yz	yz	yz@gmail.com	7764567892	2008-01-02	personal	m
53	za	za	za@gmail.com	7834567892	2009-02-02	personal	m
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

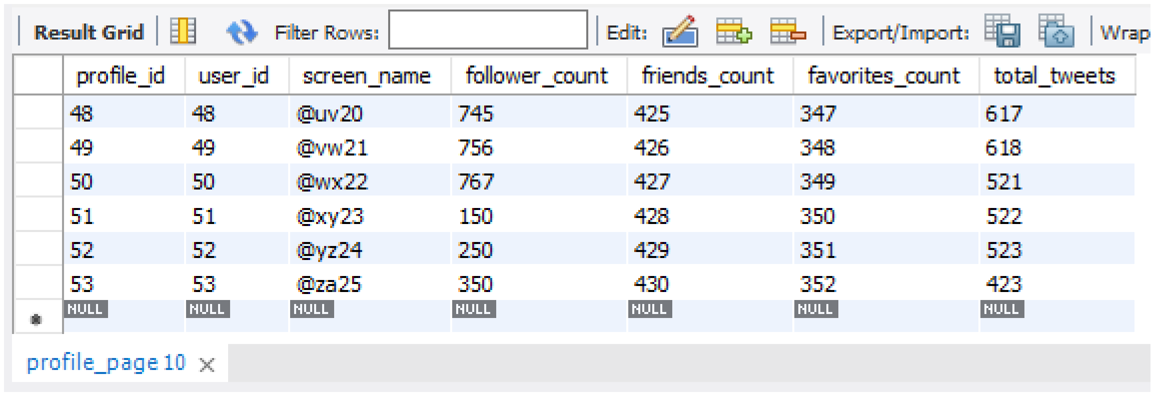
Profile_Page Table: It stores the profile details of user which will see by the follower(other user) having user screen name, follower_count, frinds_count. Table consists of total 53 records.



```
create table profile_page(
profile_id int AUTO_INCREMENT,
user_id int,
screen_name varchar(25),
follower_count int,
friends_count int,
favorites_count int,
total_tweets int,
PRIMARY KEY (profile_id),
FOREIGN KEY (profile_id) REFERENCES user(userid)

);
---
select * from profile_page;

-- drop table tweet;
```

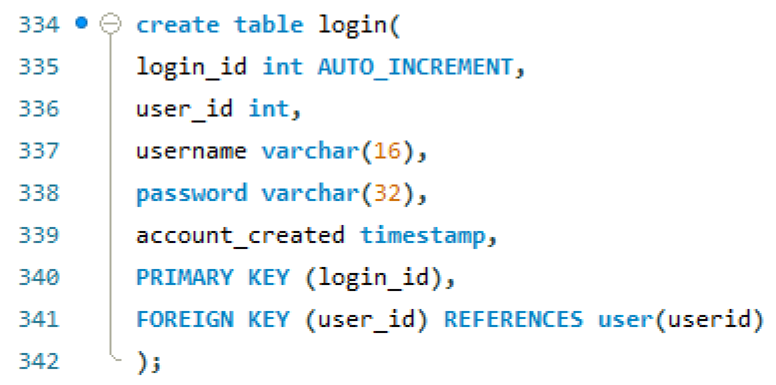


profile_id	user_id	screen_name	follower_count	friends_count	favorites_count	total_tweets
1	1	@mr1	999	555	225	1200
2	2	@sb1	752	989	150	852
3	3	@venu03	625	756	250	645
4	4	@nk02	823	789	325	426
5	5	@abc01	123	345	111	511
6	6	@bcd02	234	456	222	611
7	7	@cde03	456	789	333	777
8	8	@def04	567	542	444	555

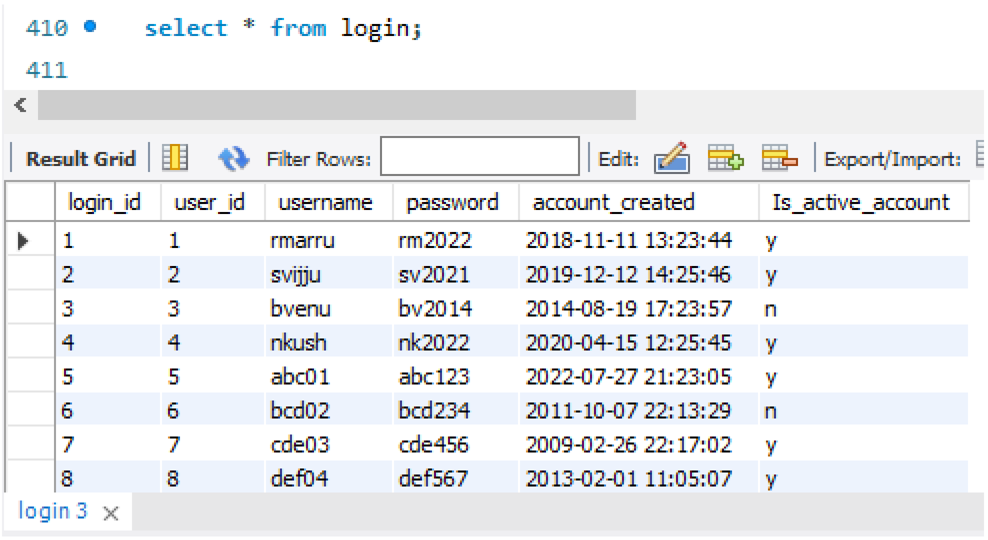


profile_id	user_id	screen_name	follower_count	friends_count	favorites_count	total_tweets
48	48	@uv20	745	425	347	617
49	49	@vw21	756	426	348	618
50	50	@wx22	767	427	349	521
51	51	@xy23	150	428	350	522
52	52	@yz24	250	429	351	523
53	53	@za25	350	430	352	423
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Login Table: This table stores the login details of user while login on twitter like Username and password. It consists of 53 records.



```
create table login(
login_id int AUTO_INCREMENT,
user_id int,
username varchar(16),
password varchar(32),
account_created timestamp,
PRIMARY KEY (login_id),
FOREIGN KEY (user_id) REFERENCES user(userid)
);
```

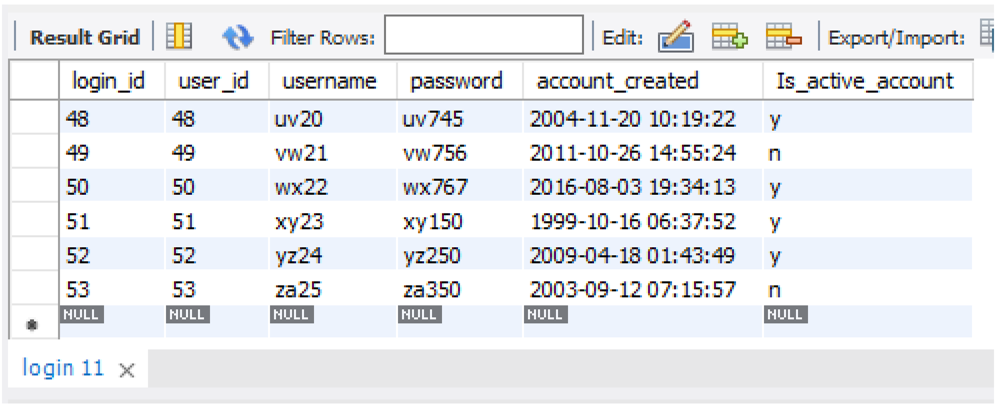


```
select * from login;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

login_id	user_id	username	password	account_created	Is_active_account
1	1	rmarru	rm2022	2018-11-11 13:23:44	y
2	2	svijju	sv2021	2019-12-12 14:25:46	y
3	3	bvenu	bv2014	2014-08-19 17:23:57	n
4	4	nkush	nk2022	2020-04-15 12:25:45	y
5	5	abc01	abc123	2022-07-27 21:23:05	y
6	6	bcd02	bcd234	2011-10-07 22:13:29	n
7	7	cde03	cde456	2009-02-26 22:17:02	y
8	8	def04	def567	2013-02-01 11:05:07	y

login 3 ×

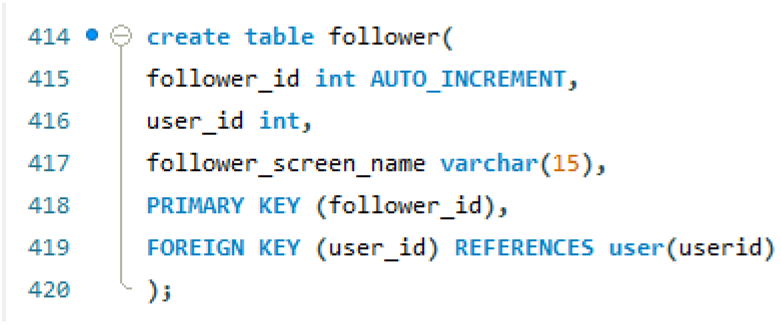


Result Grid | Filter Rows: | Edit: | Export/Import:

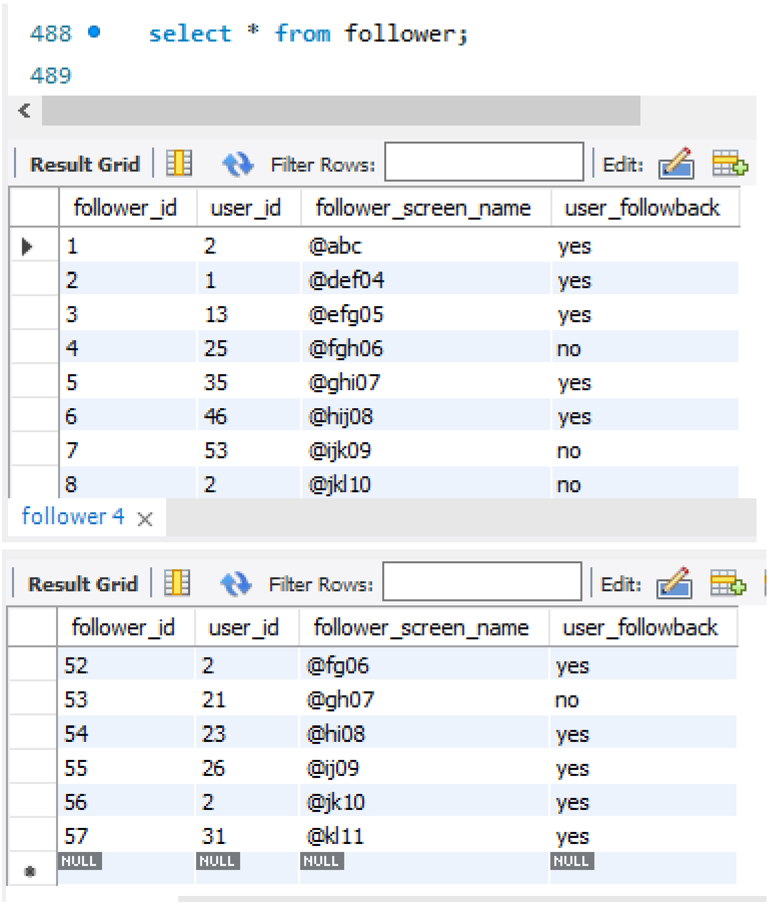
login_id	user_id	username	password	account_created	Is_active_account
48	48	uv20	uv745	2004-11-20 10:19:22	y
49	49	vw21	vw756	2011-10-26 14:55:24	n
50	50	wx22	wx767	2016-08-03 19:34:13	y
51	51	xy23	xy150	1999-10-16 06:37:52	y
52	52	yz24	yz250	2009-04-18 01:43:49	y
53	53	za25	za350	2003-09-12 07:15:57	n
NULL	NULL	NULL	NULL	NULL	NULL

login 11 ×

Follower Table: The Follower table stores the follower details like follower screen and user followback. This table consists of 57 records.



```
create table follower(
follower_id int AUTO_INCREMENT,
user_id int,
follower_screen_name varchar(15),
PRIMARY KEY (follower_id),
FOREIGN KEY (user_id) REFERENCES user(userid)
);
```



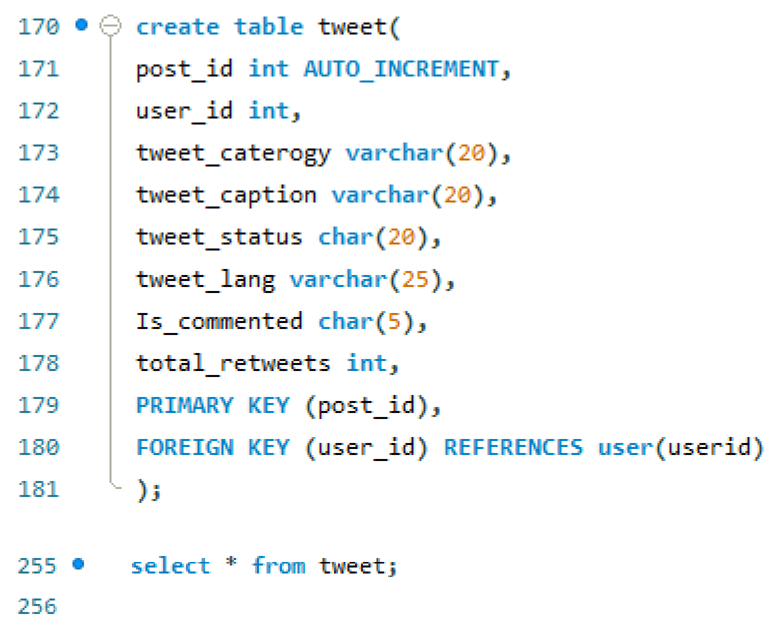
```
select * from follower;
```

follower_id	user_id	follower_screen_name	user_followback
1	2	@abc	yes
2	1	@def04	yes
3	13	@efg05	yes
4	25	@fgh06	no
5	35	@ghi07	yes
6	46	@hij08	yes
7	53	@ijk09	no
8	2	@jkl10	no

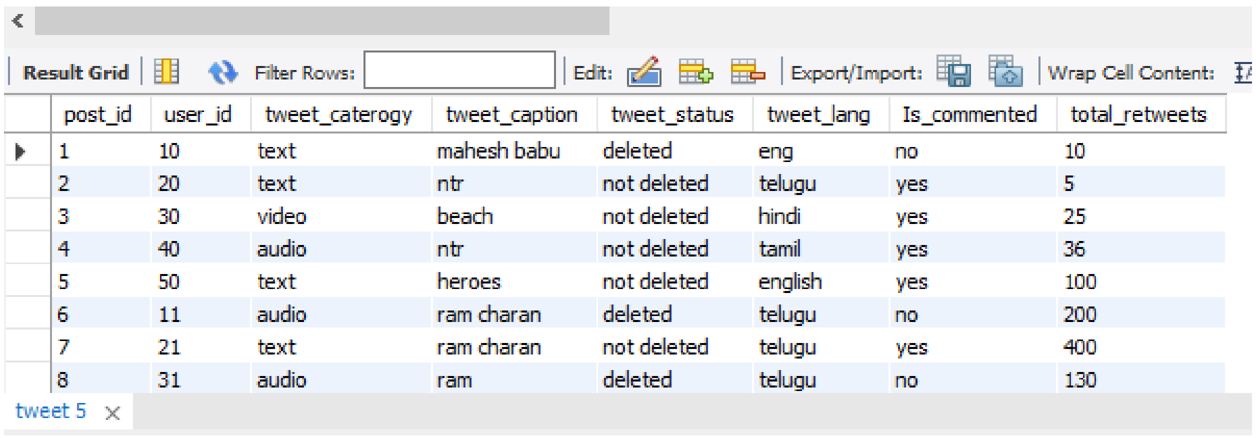
follower 4

follower_id	user_id	follower_screen_name	user_followback
52	2	@fg06	yes
53	21	@gh07	no
54	23	@hi08	yes
55	26	@ij09	yes
56	2	@jk10	yes
57	31	@kl11	yes
NULL	NULL	NULL	NULL

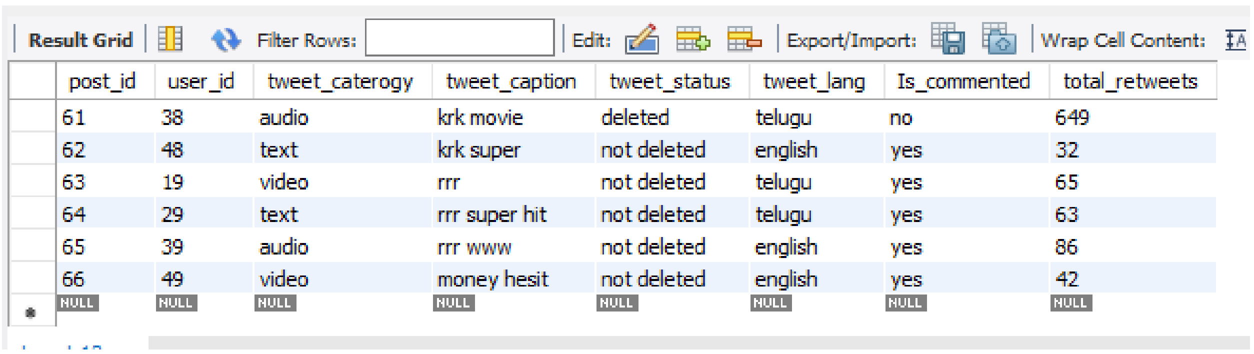
Tweet Table: This table store the details of tweet in twitter like tweet caterogy is audio, video and text, tweet_caption is tweet and tweet status whether the tweet is deleted or present. It consists of 66 records.



```
create table tweet(
post_id int AUTO_INCREMENT,
user_id int,
tweet_caterogy varchar(20),
tweet_caption varchar(20),
tweet_status char(20),
tweet_lang varchar(25),
Is_commented char(5),
total_retweets int,
PRIMARY KEY (post_id),
FOREIGN KEY (user_id) REFERENCES user(userid)
);

select * from tweet;
```

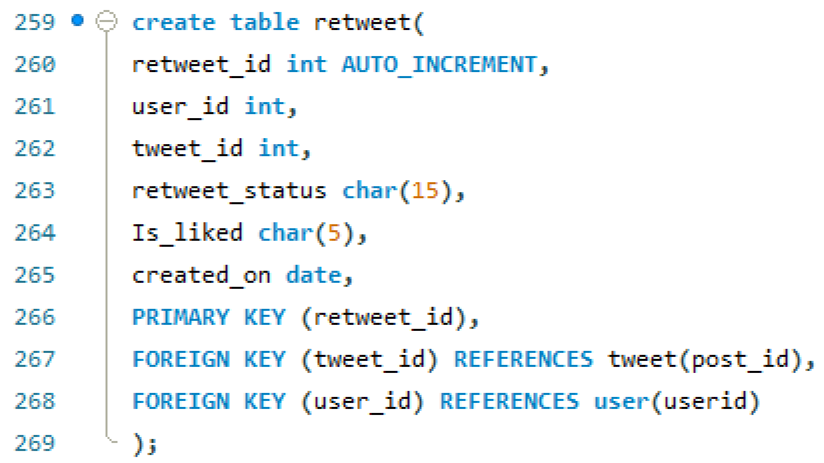


post_id	user_id	tweet_caterogy	tweet_caption	tweet_status	tweet_lang	Is_commented	total_retweets
1	10	text	mahesh babu	deleted	eng	no	10
2	20	text	ntr	not deleted	telugu	yes	5
3	30	video	beach	not deleted	hindi	yes	25
4	40	audio	ntr	not deleted	tamil	yes	36
5	50	text	heroes	not deleted	english	yes	100
6	11	audio	ram charan	deleted	telugu	no	200
7	21	text	ram charan	not deleted	telugu	yes	400
8	31	audio	ram	deleted	telugu	no	130

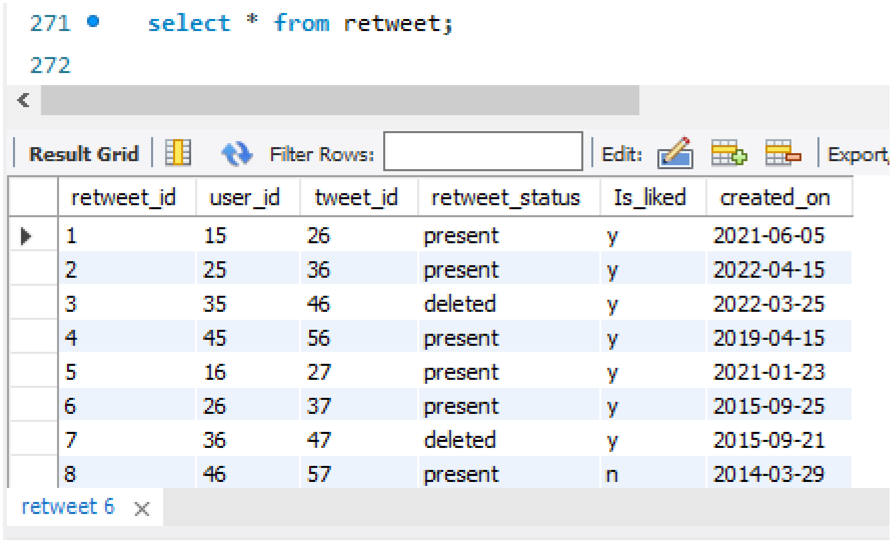


post_id	user_id	tweet_caterogy	tweet_caption	tweet_status	tweet_lang	Is_commented	total_retweets
61	38	audio	krk movie	deleted	telugu	no	649
62	48	text	krk super	not deleted	english	yes	32
63	19	video	rrr	not deleted	telugu	yes	65
64	29	text	rrr super hit	not deleted	telugu	yes	63
65	39	audio	rrr www	not deleted	english	yes	86
66	49	video	money hesit	not deleted	english	yes	42
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Retweet Table: This table stores the retweets created on the tweet in twitter. It consists of 55 rows.



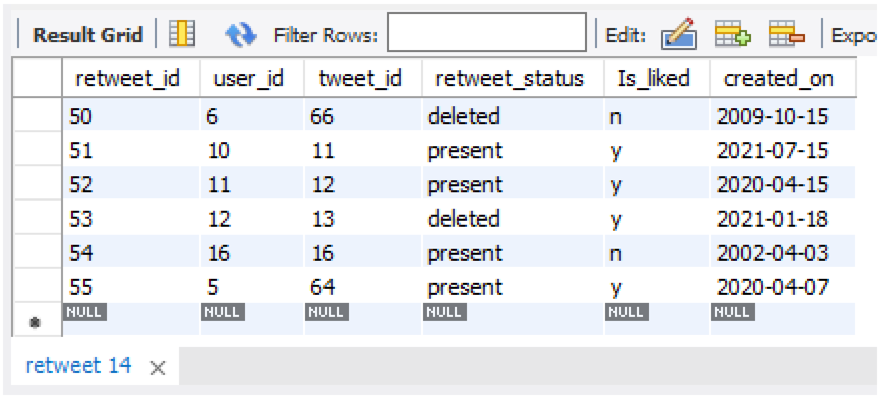
```
create table retweet(
retweet_id int AUTO_INCREMENT,
user_id int,
tweet_id int,
retweet_status char(15),
Is_liked char(5),
created_on date,
PRIMARY KEY (retweet_id),
FOREIGN KEY (tweet_id) REFERENCES tweet(post_id),
FOREIGN KEY (user_id) REFERENCES user(userid)
);
```



```
select * from retweet;
```

retweet_id	user_id	tweet_id	retweet_status	Is_liked	created_on
1	15	26	present	y	2021-06-05
2	25	36	present	y	2022-04-15
3	35	46	deleted	y	2022-03-25
4	45	56	present	y	2019-04-15
5	16	27	present	y	2021-01-23
6	26	37	present	y	2015-09-25
7	36	47	deleted	y	2015-09-21
8	46	57	present	n	2014-03-29

retweet 6

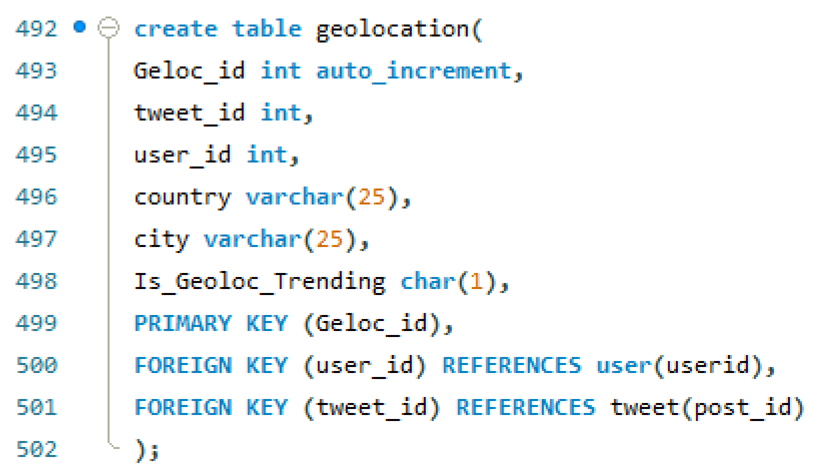


retweet_id	user_id	tweet_id	retweet_status	Is_liked	created_on
50	6	66	deleted	n	2009-10-15
51	10	11	present	y	2021-07-15
52	11	12	present	y	2020-04-15
53	12	13	deleted	y	2021-01-18
54	16	16	present	n	2002-04-03
55	5	64	present	y	2020-04-07
NULL	NULL	NULL	NULL	NULL	NULL

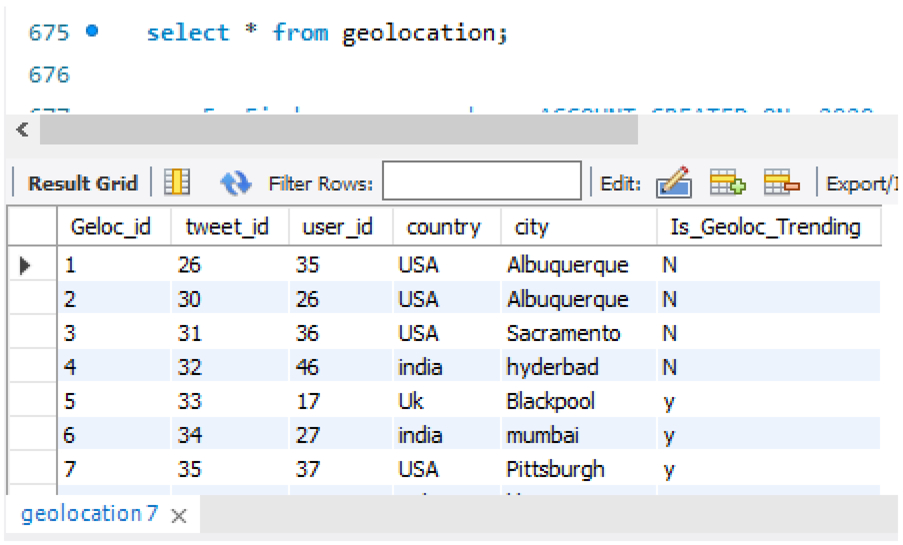
retweet 14

Geolocation Table: This table stores location where the tweet is posted by the user.

It consists of 56 records.

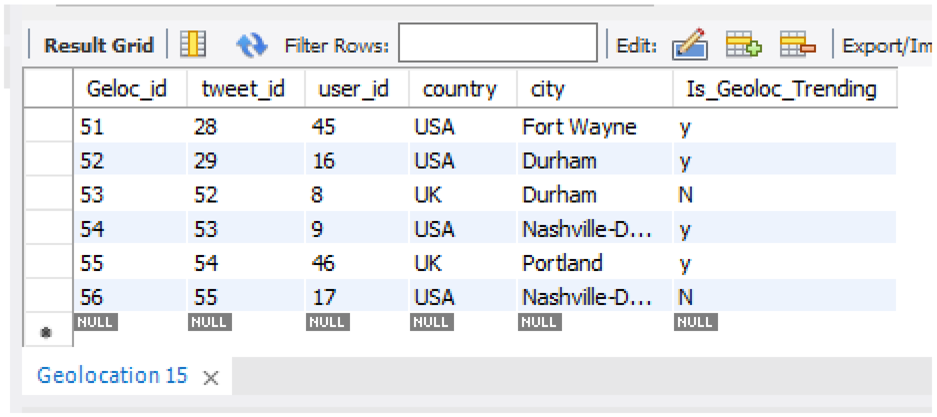


```sql
create table geolocation(
Geloc_id int auto_increment,
tweet_id int,
user_id int,
country varchar(25),
city varchar(25),
Is_Geoloc_Trending char(1),
PRIMARY KEY (Geloc_id),
FOREIGN KEY (user_id) REFERENCES user(userid),
FOREIGN KEY (tweet_id) REFERENCES tweet(post_id)
);
```



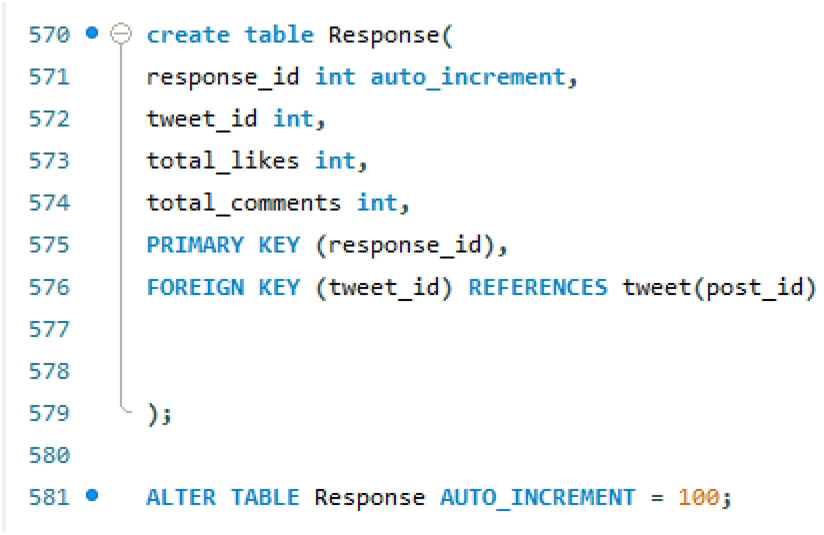
```sql
select * from geolocation;
```

Geloc_id	tweet_id	user_id	country	city	Is_Geoloc_Trending
1	26	35	USA	Albuquerque	N
2	30	26	USA	Albuquerque	N
3	31	36	USA	Sacramento	N
4	32	46	india	hyderbad	N
5	33	17	Uk	Blackpool	y
6	34	27	india	mumbai	y
7	35	37	USA	Pittsburgh	y

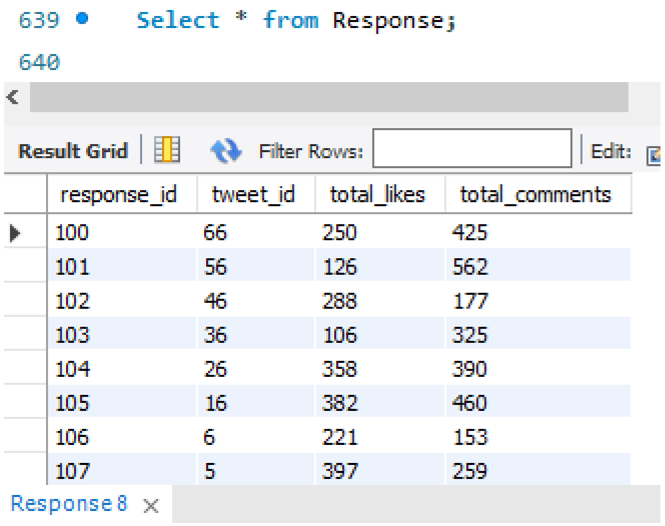


Geloc_id	tweet_id	user_id	country	city	Is_Geoloc_Trending
51	28	45	USA	Fort Wayne	y
52	29	16	USA	Durham	y
53	52	8	UK	Durham	N
54	53	9	USA	Nashville-D...	y
55	54	46	UK	Portland	y
56	55	17	USA	Nashville-D...	N
NULL	NULL	NULL	NULL	NULL	NULL

Response Table: This table stores the response made by the follower to the tweet by likes and comments. It is consists of 52 records.



```sql
create table Response(
response_id int auto_increment,
tweet_id int,
total_likes int,
total_comments int,
PRIMARY KEY (response_id),
FOREIGN KEY (tweet_id) REFERENCES tweet(post_id)


);

ALTER TABLE Response AUTO_INCREMENT = 100;
```

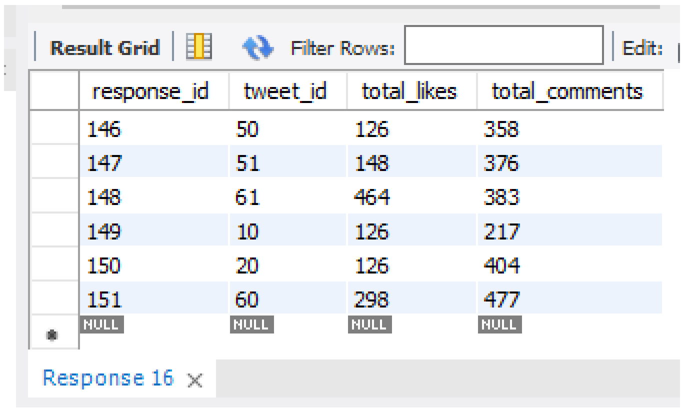


```sql
Select * from Response;
```

Result Grid | Filter Rows: | Edit:

response_id	tweet_id	total_likes	total_comments
100	66	250	425
101	56	126	562
102	46	288	177
103	36	106	325
104	26	358	390
105	16	382	460
106	6	221	153
107	5	397	259

Response 8



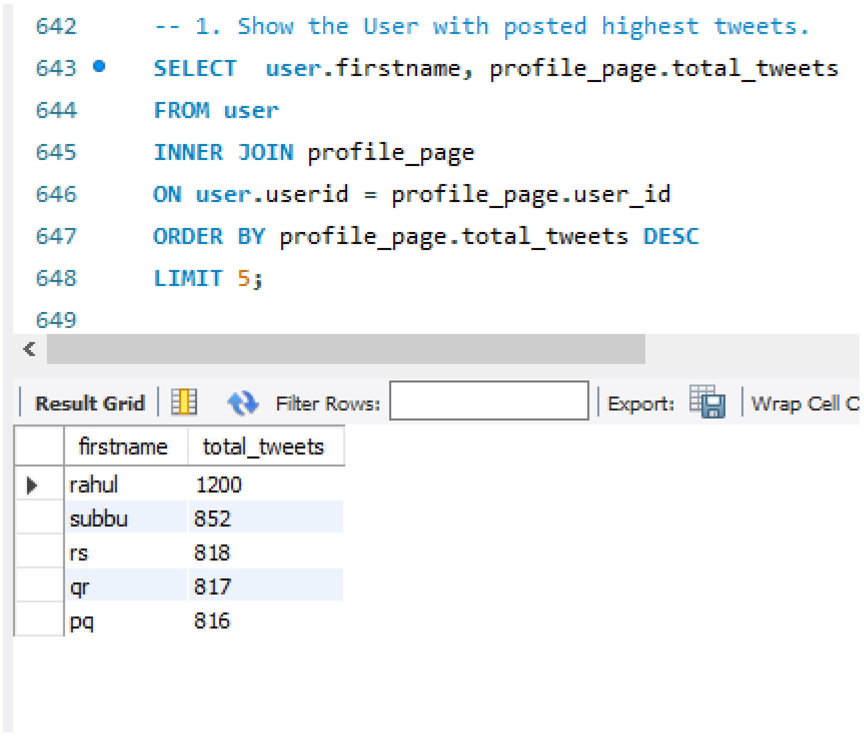
Result Grid | Filter Rows: | Edit:

response_id	tweet_id	total_likes	total_comments
146	50	126	358
147	51	148	376
148	61	464	383
149	10	126	217
150	20	126	404
151	60	298	477
NULL	NULL	NULL	NULL

Response 16

5. 15 Use-Cases in English and Implement the Queries for MySQL:

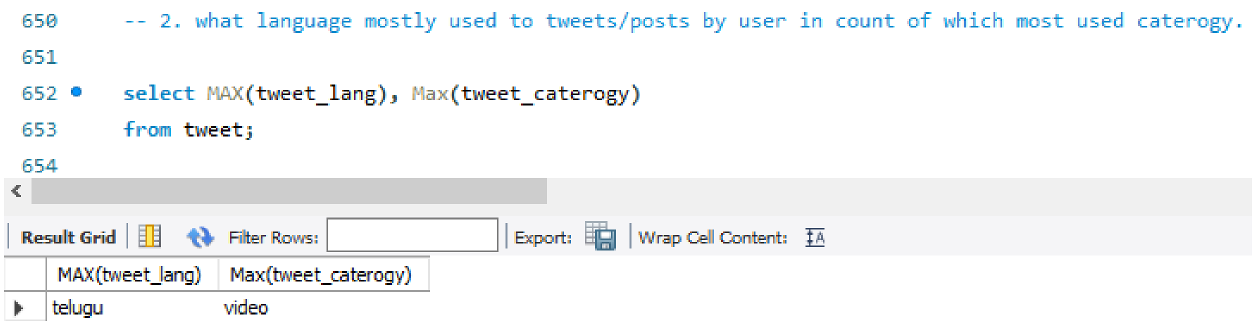
1. Show the User with posted highest tweets.



```sql
-- 1. Show the User with posted highest tweets.
SELECT  user.firstname, profile_page.total_tweets
FROM user
INNER JOIN profile_page
ON user.userid = profile_page.user_id
ORDER BY profile_page.total_tweets DESC
LIMIT 5;
```

firstname	total_tweets
rahul	1200
subbu	852
rs	818
qr	817
pq	816

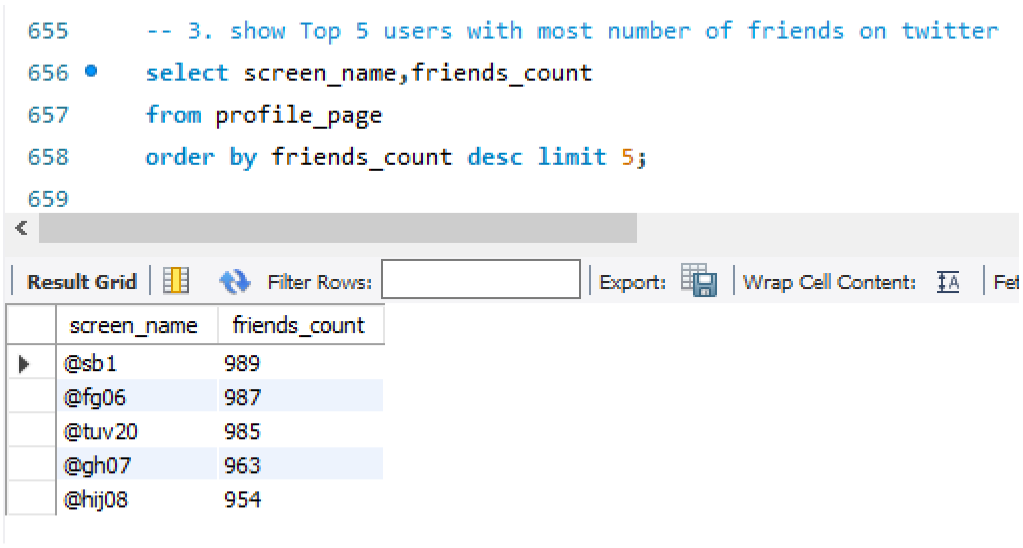
2. Retrieve what language mostly used to tweets/posts by user in count of which most used caterogy.



```sql
-- 2. what language mostly used to tweets/posts by user in count of which most used caterogy.

select MAX(tweet_lang), Max(tweet_caterogy)
from tweet;
```

MAX(tweet_lang)	Max(tweet_caterogy)
telugu	video

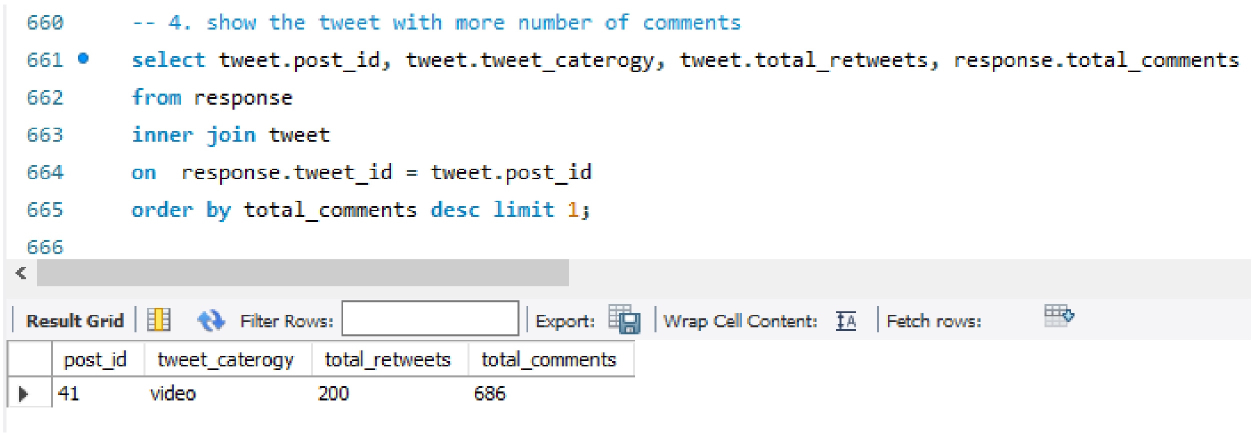
3. show Top 5 users with most number of friends on twitter.



```
-- 3. show Top 5 users with most number of friends on twitter
select screen_name,friends_count
from profile_page
order by friends_count desc limit 5;
```

screen_name	friends_count
@sb1	989
@fg06	987
@tuv20	985
@gh07	963
@hij08	954

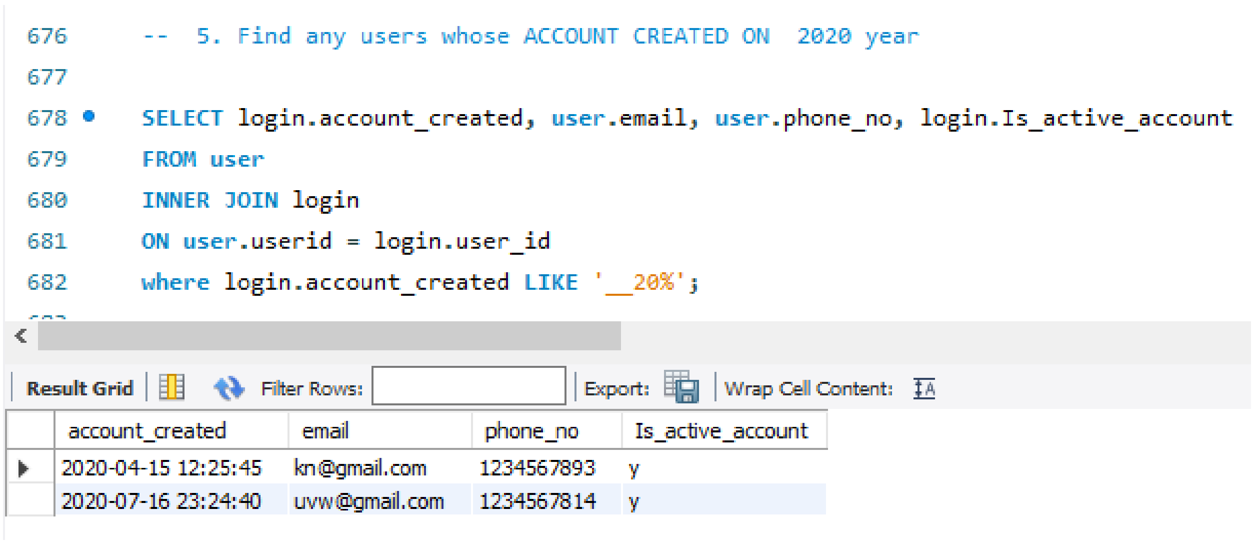
4. show the tweet with more number of comments.



```
-- 4. show the tweet with more number of comments
select tweet.post_id, tweet.tweet_caterogy, tweet.total_retweets, response.total_comments
from response
inner join tweet
on  response.tweet_id = tweet.post_id
order by total_comments desc limit 1;
```

post_id	tweet_caterogy	total_retweets	total_comments
41	video	200	686

5. Find any users whose ACCOUNT CREATED ON 2020 year.

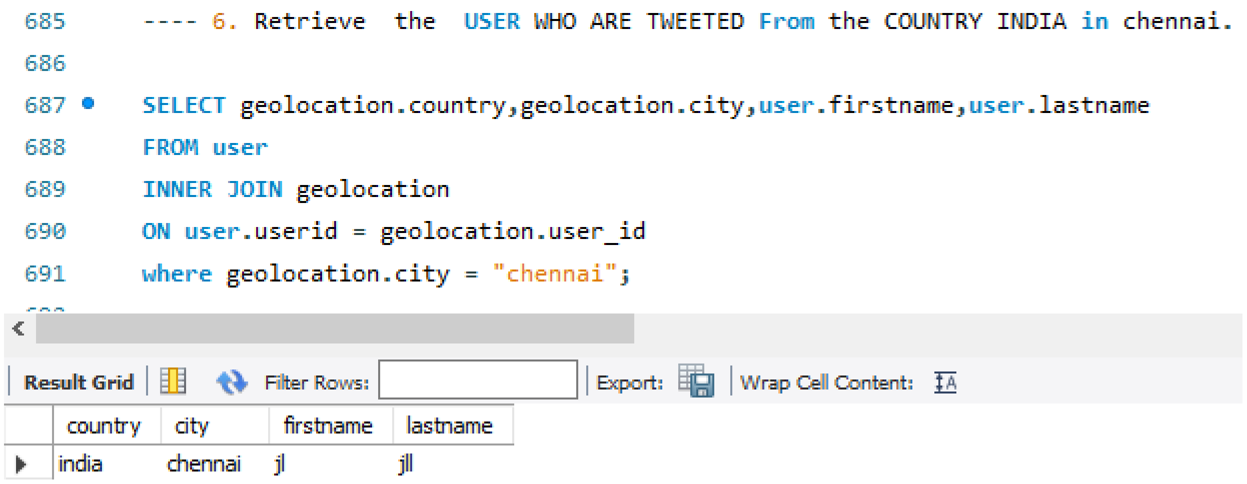


```
-- 5. Find any users whose ACCOUNT CREATED ON  2020 year

SELECT login.account_created, user.email, user.phone_no, login.Is_active_account
FROM user
INNER JOIN login
ON user.userid = login.user_id
where login.account_created LIKE '__20%';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
account_created      email          phone_no     Is_active_account
2020-04-15 12:25:45  kn@gmail.com   1234567893   y
2020-07-16 23:24:40  uvw@gmail.com  1234567814   y
```

6. Retrieve the USER WHO ARE TWEETED From the COUNTRY INDIA in chennai.

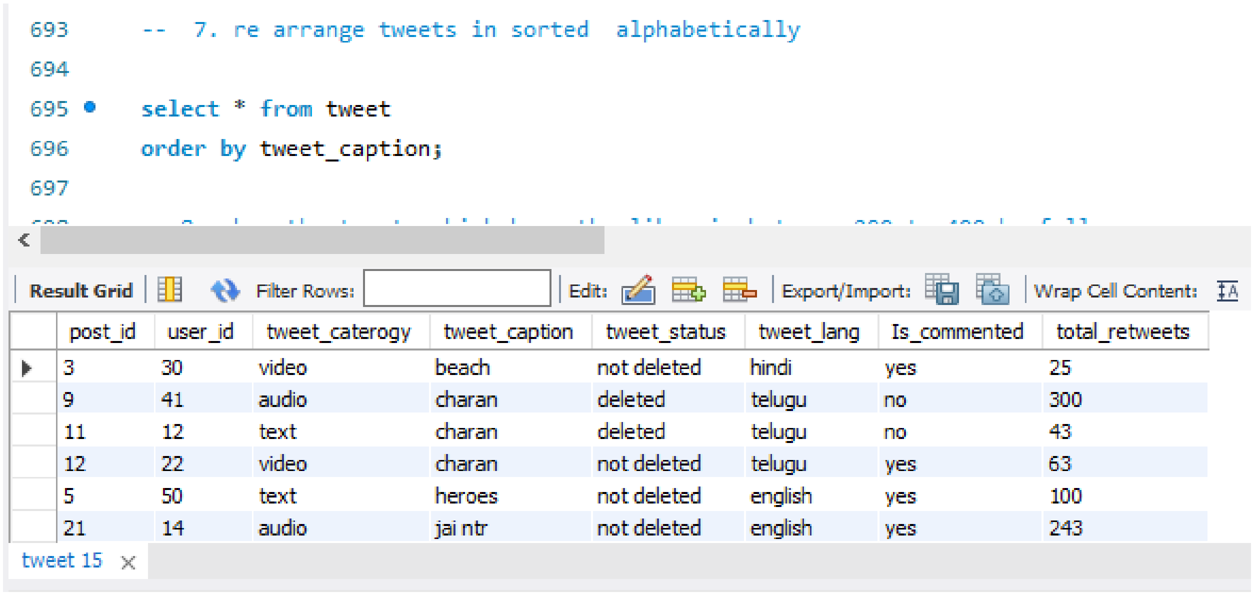


```
---- 6. Retrieve  the  USER WHO ARE TWEETED From the COUNTRY INDIA in chennai.

SELECT geolocation.country,geolocation.city,user.firstname,user.lastname
FROM user
INNER JOIN geolocation
ON user.userid = geolocation.user_id
where geolocation.city = "chennai";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
country  city     firstname  lastname
india    chennai  jl         jll
```

7. re arrange tweets in sorted alphabetically.



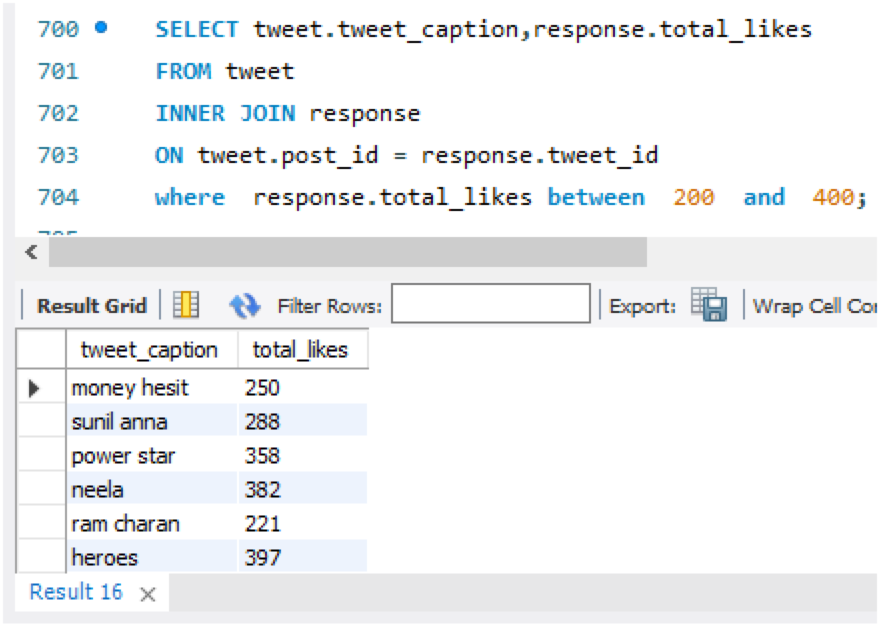
```
-- 7. re arrange tweets in sorted  alphabetically

select * from tweet
order by tweet_caption;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

post_id	user_id	tweet_caterogy	tweet_caption	tweet_status	tweet_lang	Is_commented	total_retweets
3	30	video	beach	not deleted	hindi	yes	25
9	41	audio	charan	deleted	telugu	no	300
11	12	text	charan	deleted	telugu	no	43
12	22	video	charan	not deleted	telugu	yes	63
5	50	text	heroes	not deleted	english	yes	100
21	14	audio	jai ntr	not deleted	english	yes	243

tweet 15

8. show the tweets which have the likes in between 200 to 400 by followers.



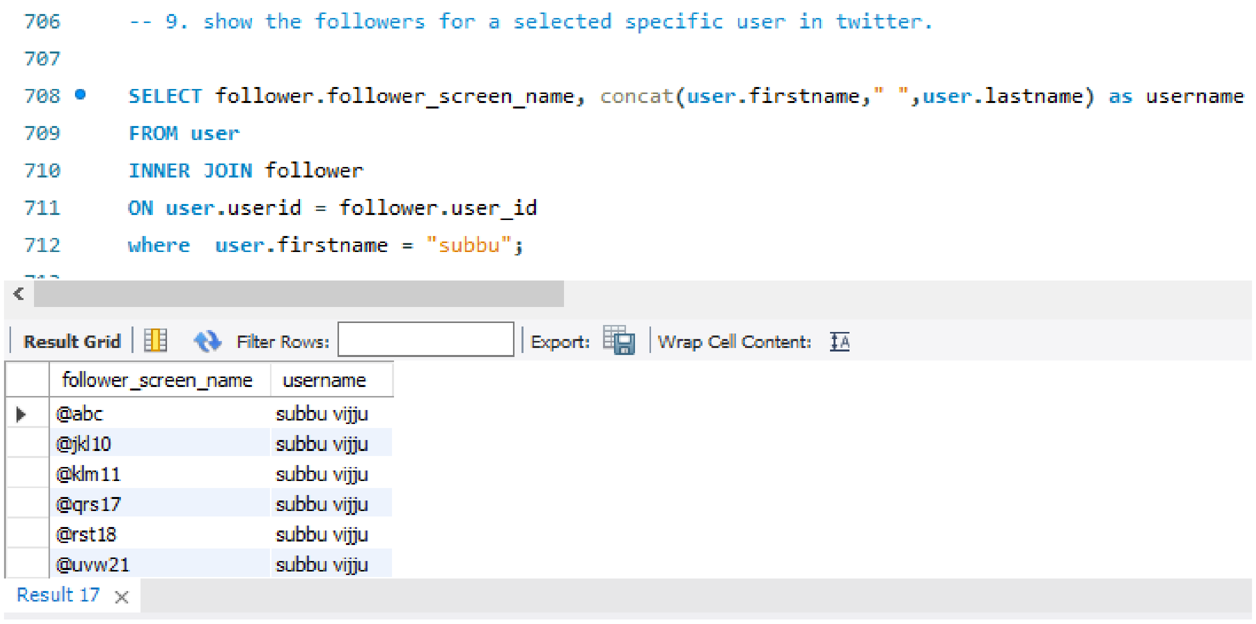
```
SELECT tweet.tweet_caption,response.total_likes
FROM tweet
INNER JOIN response
ON tweet.post_id = response.tweet_id
where  response.total_likes between  200  and  400;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

tweet_caption	total_likes
money hesit	250
sunil anna	288
power star	358
neela	382
ram charan	221
heroes	397

Result 16

9. show the followers for a selected specific user in twitter.

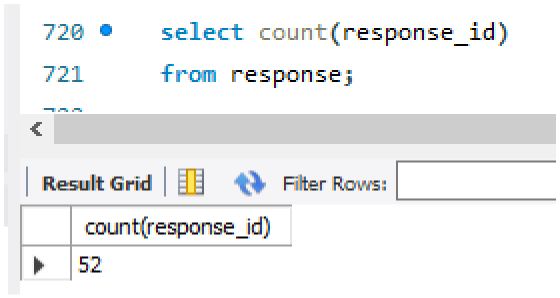


```
-- 9. show the followers for a selected specific user in twitter.

SELECT follower.follower_screen_name, concat(user.firstname," ",user.lastname) as username
FROM user
INNER JOIN follower
ON user.userid = follower.user_id
where  user.firstname = "subbu";
```

```
follower_screen_name   username
@abc                   subbu vijju
@jkl10                 subbu vijju
@klm11                 subbu vijju
@qrs17                 subbu vijju
@rst18                 subbu vijju
@uvw21                 subbu vijju
```

Result 17

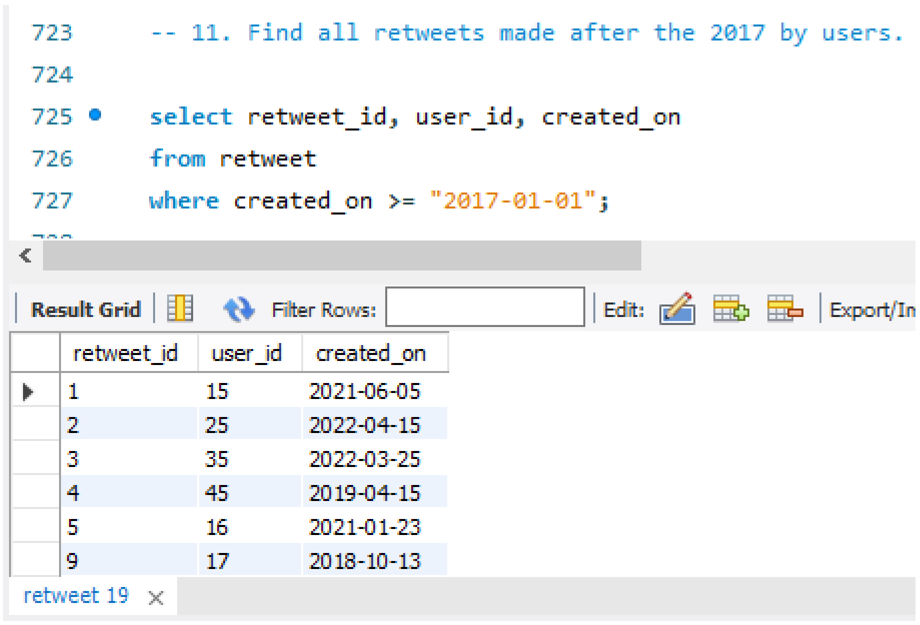
10. total number of responses given to tweets.



```
select count(response_id)
from response;
```

```
count(response_id)
52
```

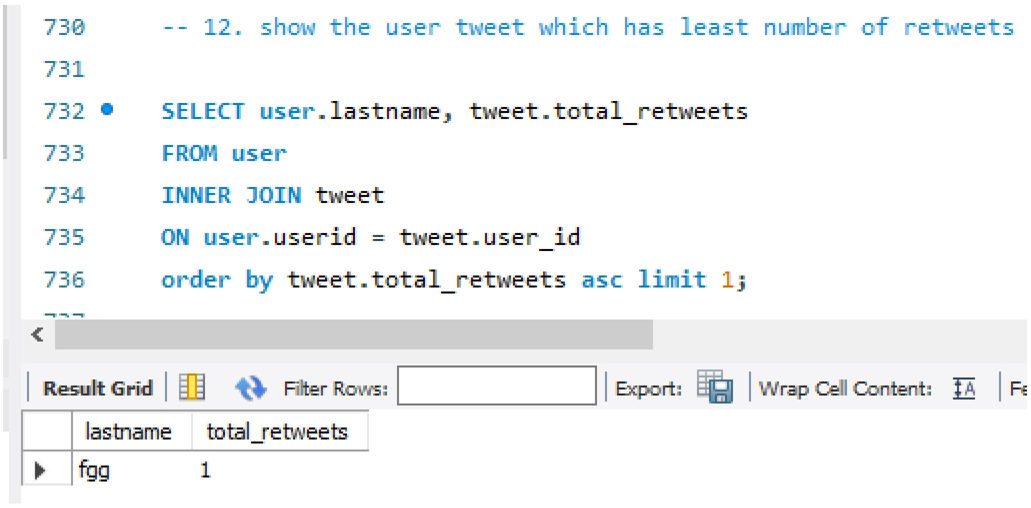
11. Find all retweets made after the 2017 by users.



```
-- 11. Find all retweets made after the 2017 by users.

select retweet_id, user_id, created_on
from retweet
where created_on >= "2017-01-01";
```

retweet_id	user_id	created_on
1	15	2021-06-05
2	25	2022-04-15
3	35	2022-03-25
4	45	2019-04-15
5	16	2021-01-23
9	17	2018-10-13

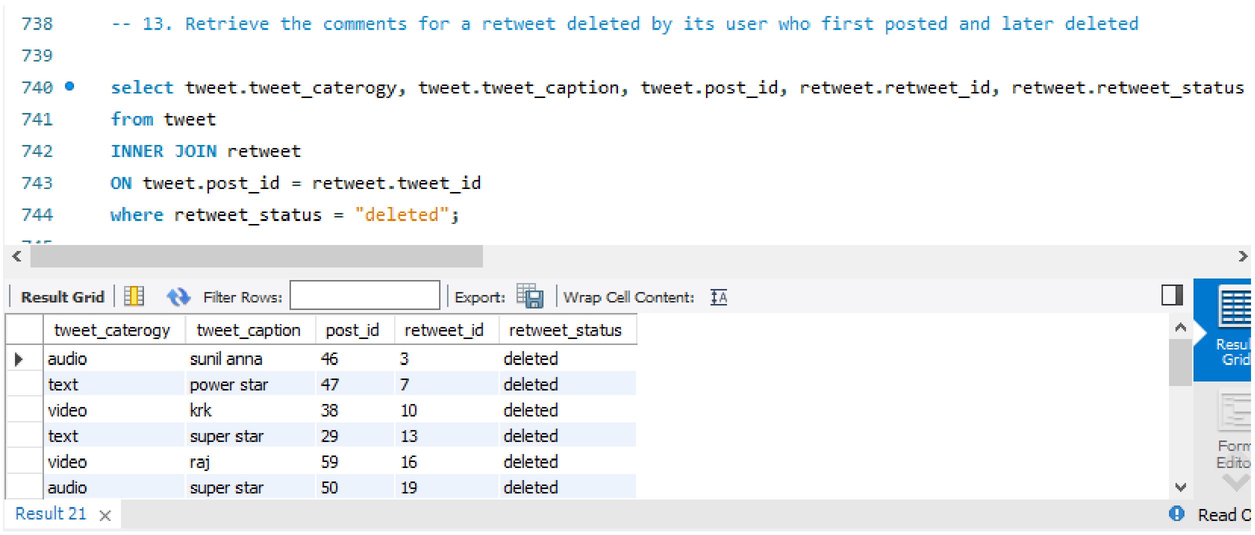
12. show the user tweet which has least number of retweets.



```
-- 12. show the user tweet which has least number of retweets

SELECT user.lastname, tweet.total_retweets
FROM user
INNER JOIN tweet
ON user.userid = tweet.user_id
order by tweet.total_retweets asc limit 1;
```

lastname	total_retweets
fgg	1

13. Retrieve the comments for a retweet deleted by its user who first posted and later deleted.

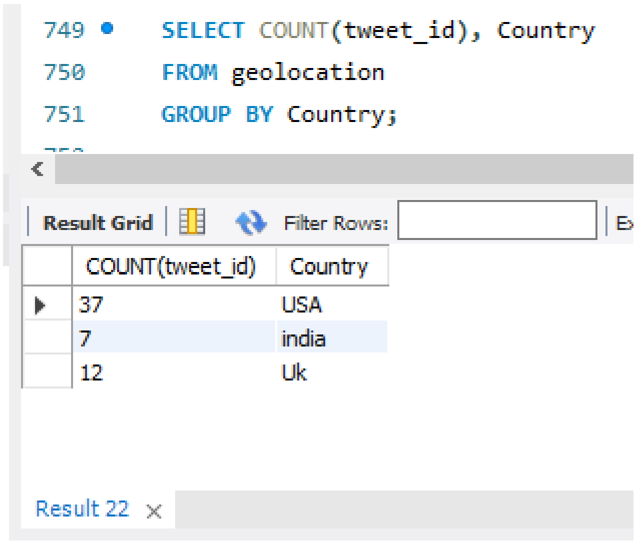


```
-- 13. Retrieve the comments for a retweet deleted by its user who first posted and later deleted

select tweet.tweet_caterogy, tweet.tweet_caption, tweet.post_id, retweet.retweet_id, retweet.retweet_status
from tweet
INNER JOIN retweet
ON tweet.post_id = retweet.tweet_id
where retweet_status = "deleted";
```

tweet_caterogy	tweet_caption	post_id	retweet_id	retweet_status
audio	sunil anna	46	3	deleted
text	power star	47	7	deleted
video	krk	38	10	deleted
text	super star	29	13	deleted
video	raj	59	16	deleted
audio	super star	50	19	deleted

Result 21

14.Retrieve the users along with its Geo locations from where the tweets are posted and when it was posted.

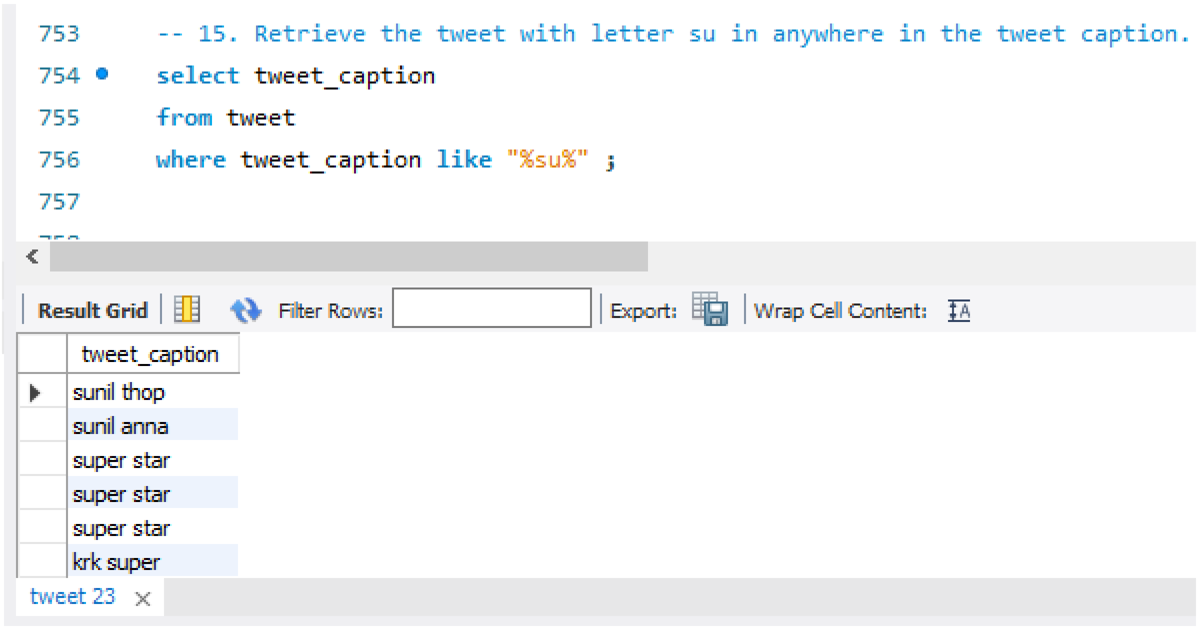


```
SELECT COUNT(tweet_id), Country
FROM geolocation
GROUP BY Country;
```

COUNT(tweet_id)	Country
37	USA
7	india
12	Uk

Result 22

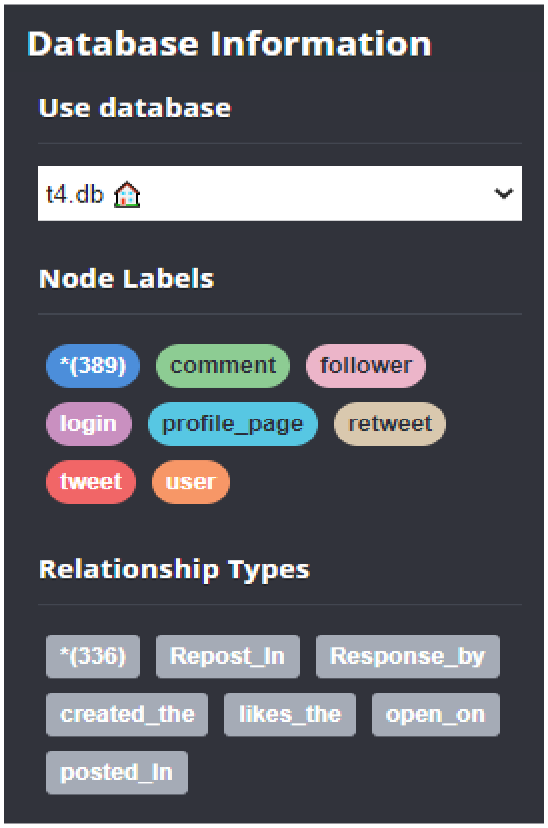
15. Retrieve the tweets with letters “su” in anywhere in the tweet caption.



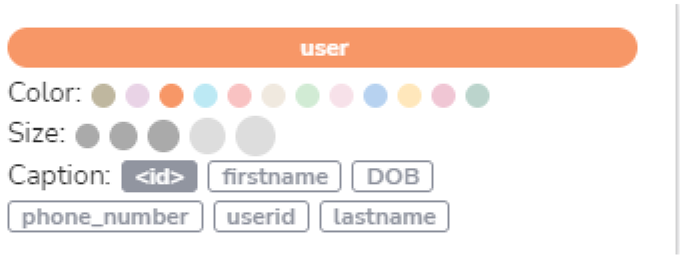
```
-- 15. Retrieve the tweet with letter su in anywhere in the tweet caption.
select tweet_caption
from tweet
where tweet_caption like "%su%" ;
```

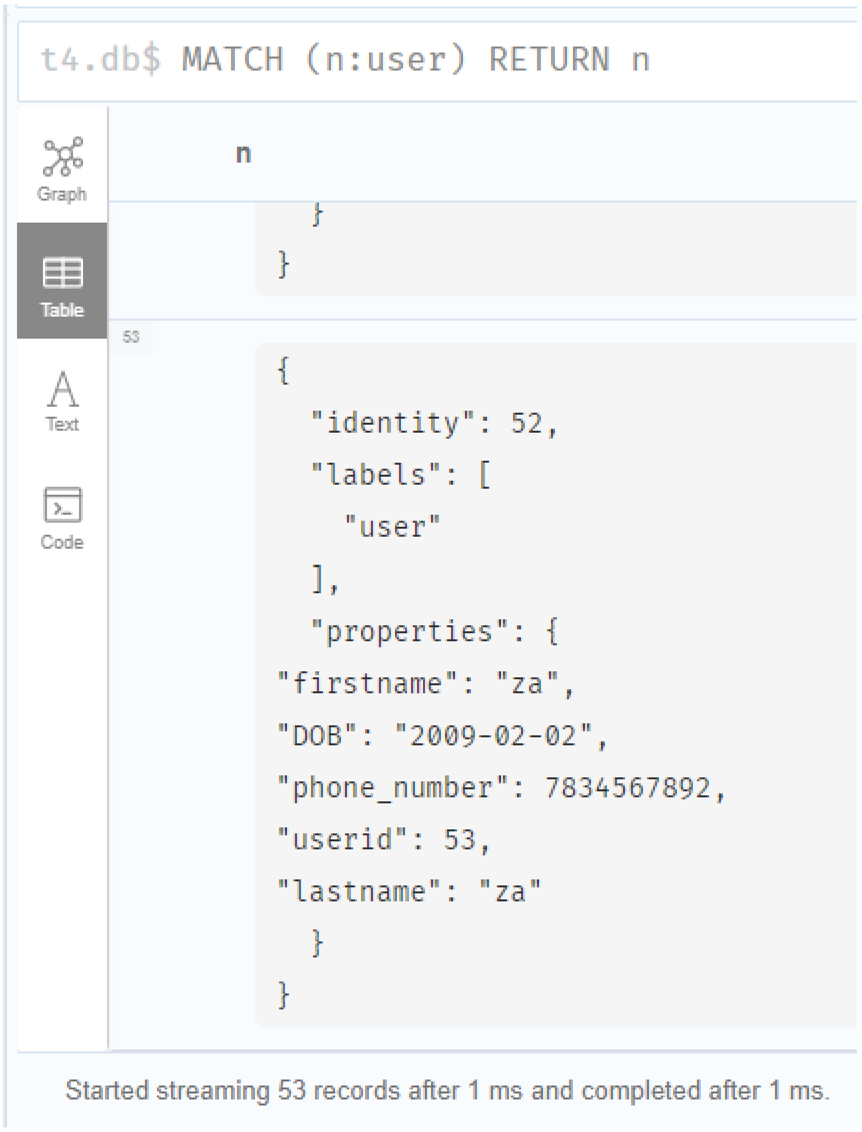
tweet_caption
sunil thop
sunil anna
super star
super star
super star
krk super

6. Sample Twitter Graph DataBase:



User (Node) -> Property Keys:





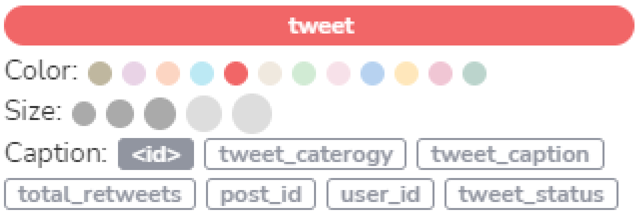
```
t4.db$ MATCH (n:user) RETURN n
```

```
    }
  }
```

```
  {
    "identity": 52,
    "labels": [
      "user"
    ],
    "properties": {
  "firstname": "za",
  "DOB": "2009-02-02",
  "phone_number": 7834567892,
  "userid": 53,
  "lastname": "za"
    }
  }
```

Started streaming 53 records after 1 ms and completed after 1 ms.

Tweet (Node) -> Property Keys:

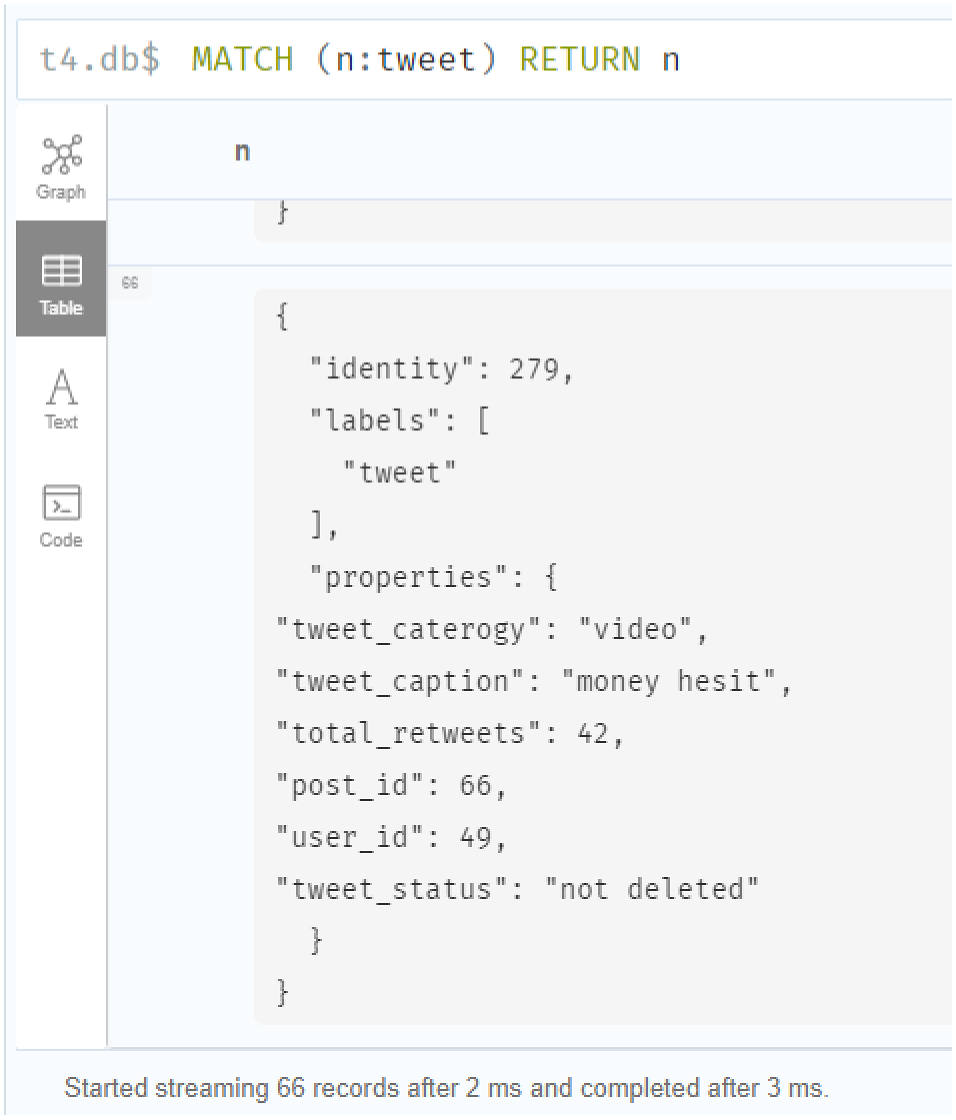


tweet

Color: ● ● ● ● ● ● ● ● ● ● ● ●

Size: ● ● ● ● ●

Caption: <id> tweet_caterogy tweet_caption total_retweets post_id user_id tweet_status

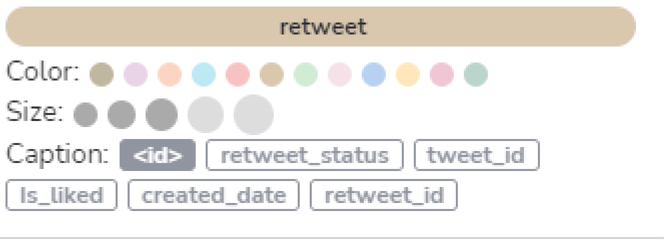


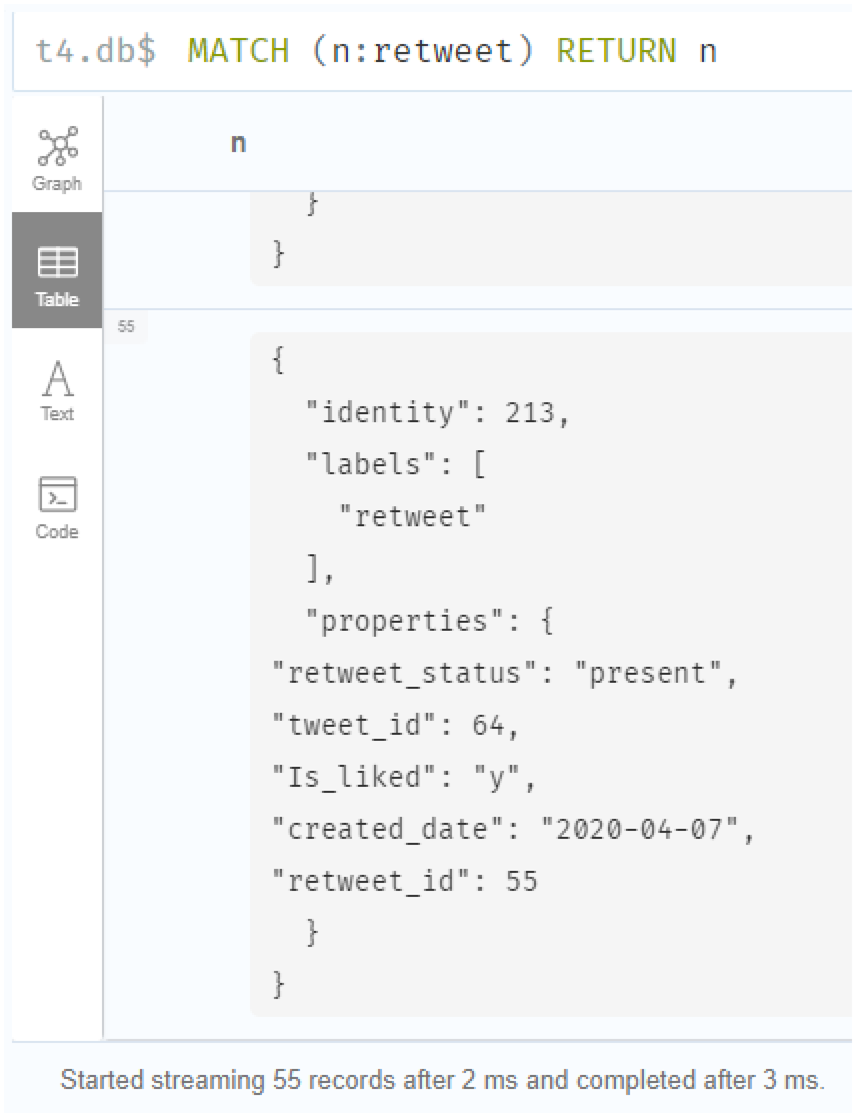
```
t4.db$ MATCH (n:tweet) RETURN n
```

```
n
}
66
{
  "identity": 279,
  "labels": [
    "tweet"
  ],
  "properties": {
"tweet_caterogy": "video",
"tweet_caption": "money hesit",
"total_retweets": 42,
"post_id": 66,
"user_id": 49,
"tweet_status": "not deleted"
  }
}
```

Started streaming 66 records after 2 ms and completed after 3 ms.

Retweet (Node) -> Property Keys:


retweet
Color:
Size:
Caption: <id> retweet_status tweet_id Is_liked created_date retweet_id



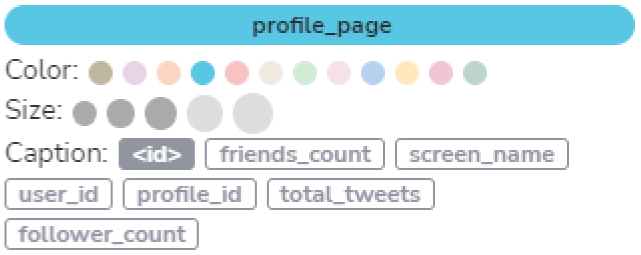
```
t4.db$ MATCH (n:retweet) RETURN n
```

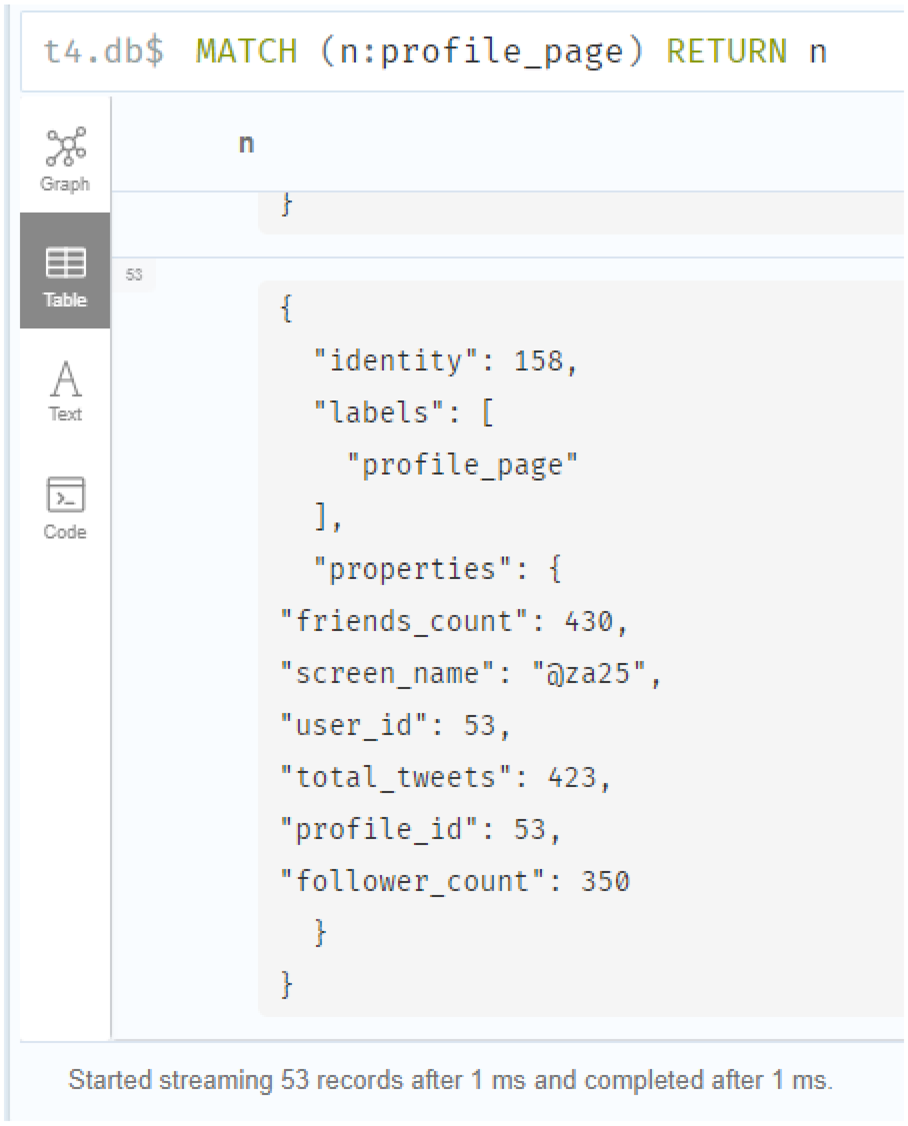
```
    }
}
```

```
{
  "identity": 213,
  "labels": [
    "retweet"
  ],
  "properties": {
"retweet_status": "present",
"tweet_id": 64,
"Is_liked": "y",
"created_date": "2020-04-07",
"retweet_id": 55
  }
}
```

Started streaming 55 records after 2 ms and completed after 3 ms.

Profile_Page (Node) -> Property Keys:





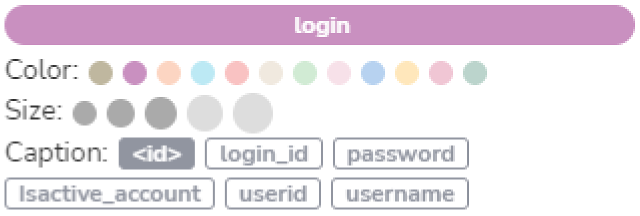
```
t4.db$ MATCH (n:profile_page) RETURN n
```

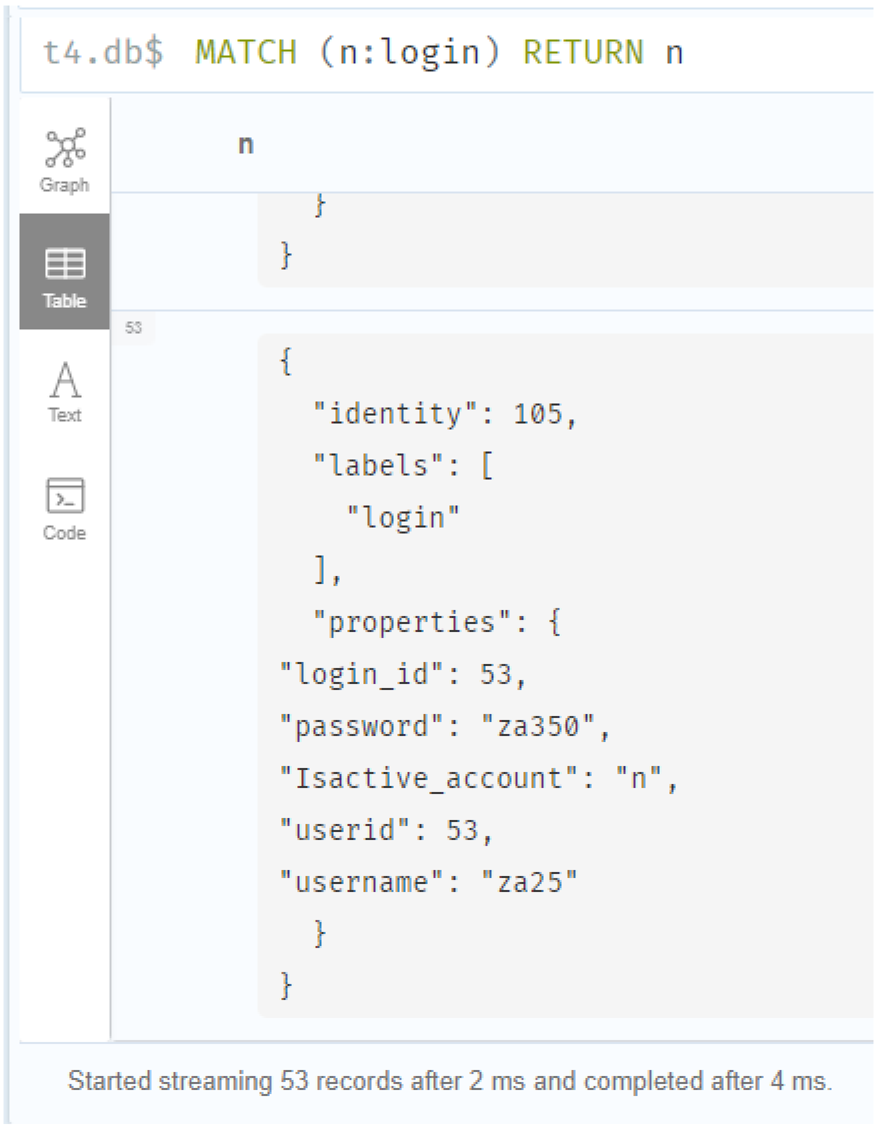
```
}
```

```
{
  "identity": 158,
  "labels": [
    "profile_page"
  ],
  "properties": {
"friends_count": 430,
"screen_name": "@za25",
"user_id": 53,
"total_tweets": 423,
"profile_id": 53,
"follower_count": 350
  }
}
```

Started streaming 53 records after 1 ms and completed after 1 ms.

Login (Node) -> Property Keys:


login
Color:
Size:
Caption: <id> login_id password
Isactive_account userid username

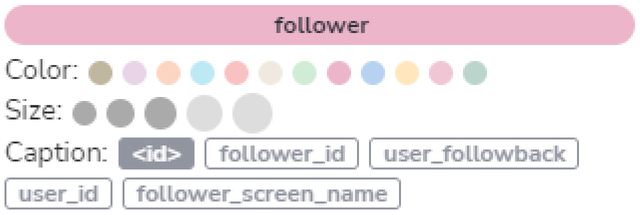


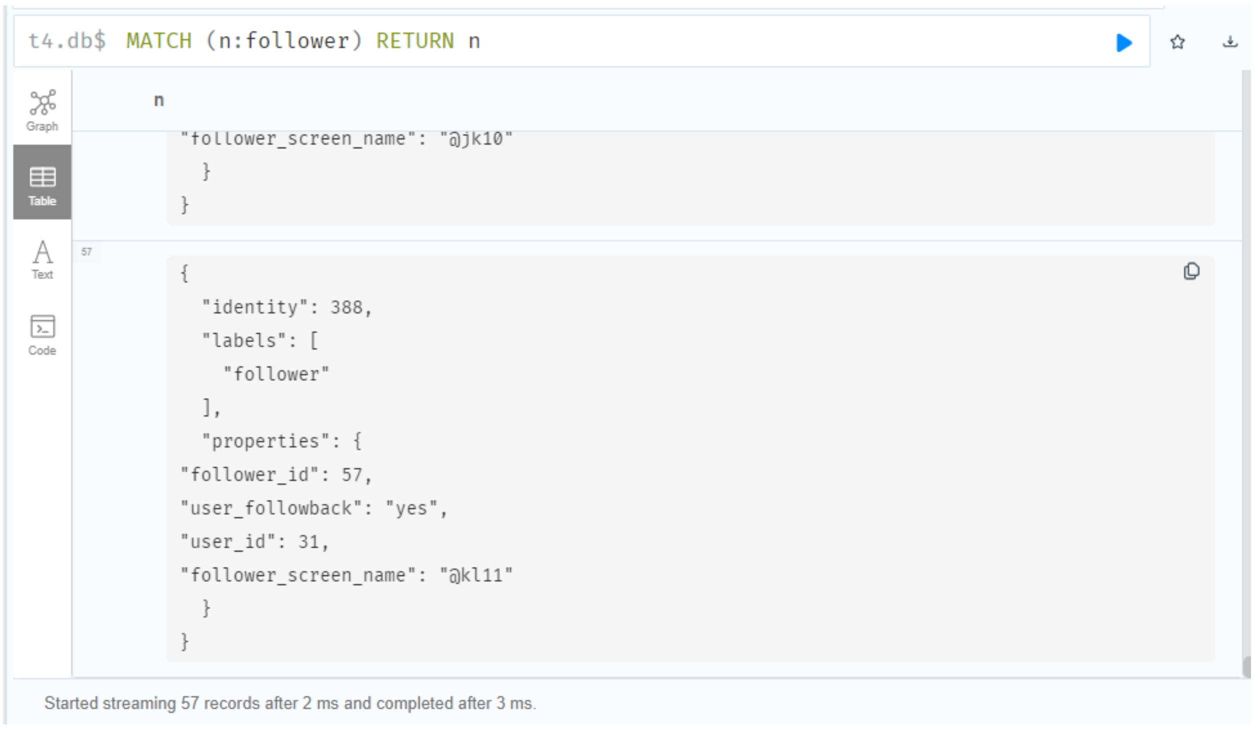
```
t4.db$ MATCH (n:login) RETURN n
```

```
n
    }
}
53
{
  "identity": 105,
  "labels": [
    "login"
  ],
  "properties": {
"login_id": 53,
"password": "za350",
"Isactive_account": "n",
"userid": 53,
"username": "za25"
  }
}
```

Started streaming 53 records after 2 ms and completed after 4 ms.

Follower (Node) -> Property Keys:


follower
Color:
Size:
Caption: <id> follower_id user_followback
user_id follower_screen_name

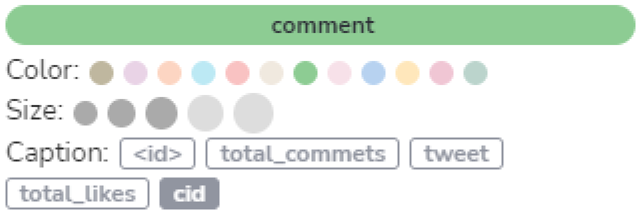


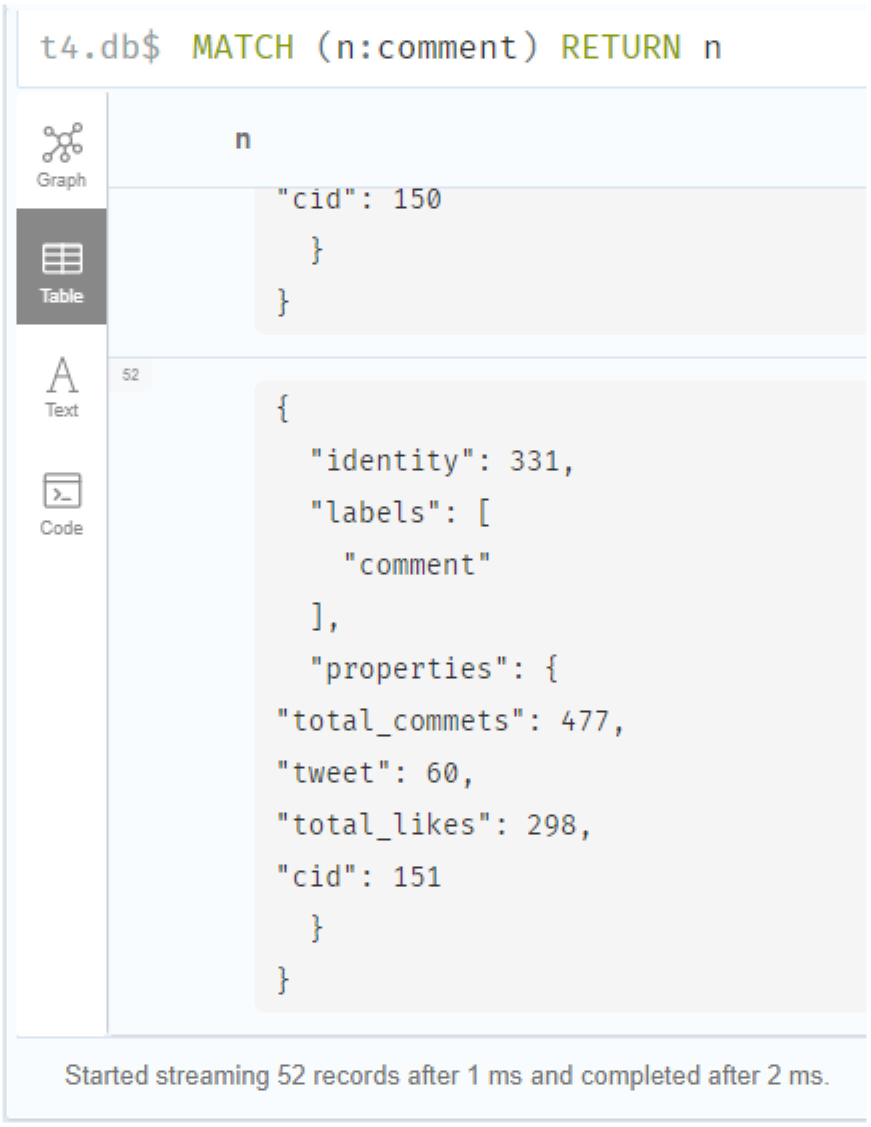
```
t4.db$ MATCH (n:follower) RETURN n
```

```
n
"follower_screen_name": "@jk10"
  }
}

{
  "identity": 388,
  "labels": [
    "follower"
  ],
  "properties": {
"follower_id": 57,
"user_followback": "yes",
"user_id": 31,
"follower_screen_name": "@kl11"
  }
}
```

Started streaming 57 records after 2 ms and completed after 3 ms.

Comment (Node) -> Property Keys:





```
t4.db$ MATCH (n:comment) RETURN n
```

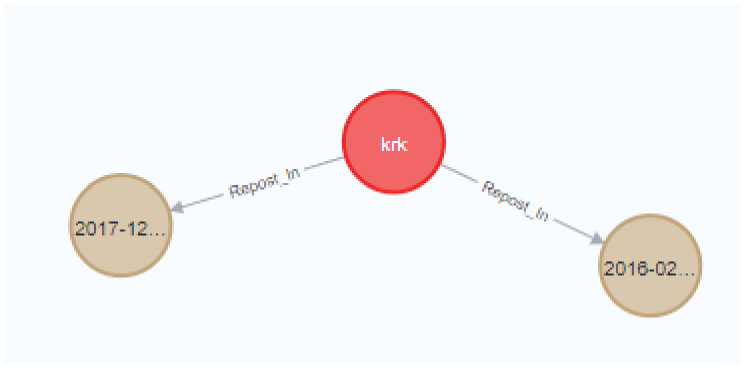
```
"cid": 150
  }
}
```

```
{
  "identity": 331,
  "labels": [
    "comment"
  ],
  "properties": {
"total_commets": 477,
"tweet": 60,
"total_likes": 298,
"cid": 151
  }
}
```

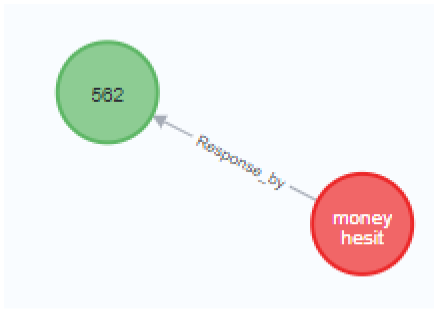
Started streaming 52 records after 1 ms and completed after 2 ms.

Therefore, relationship types According to cypher query:

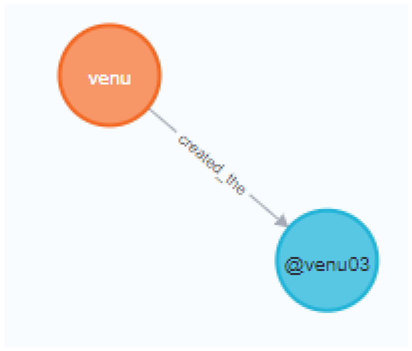
1. Tweet -> Repost_In -> Retweet



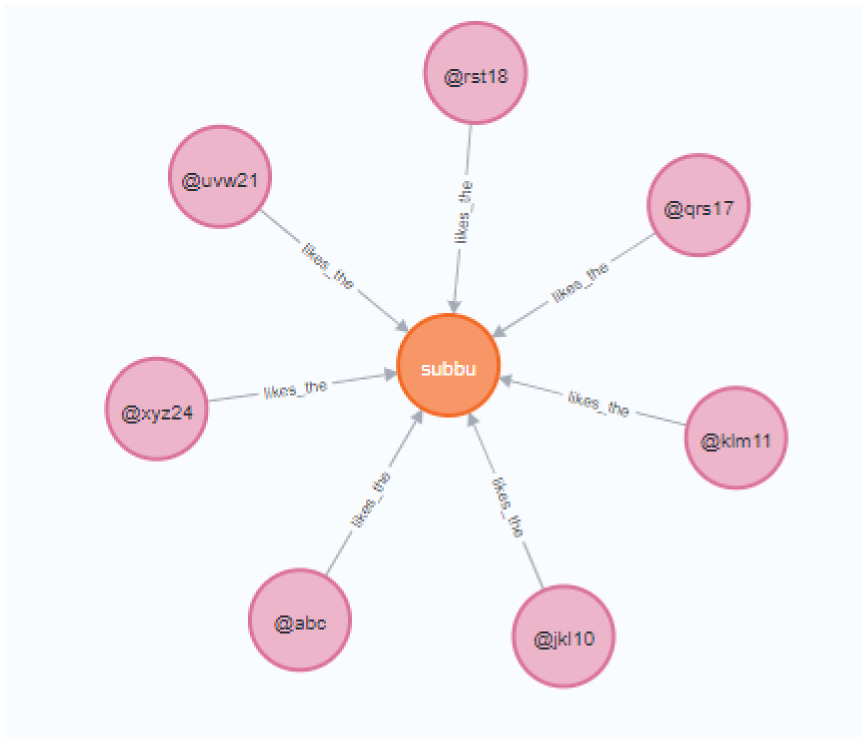
2. Tweet -> Reponse_by -> Comment



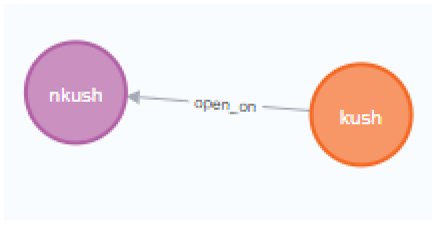
3. User -> Created_the -> Profile_Page



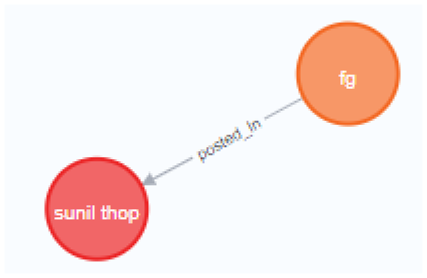
4. Follower -> Likes_the -> User



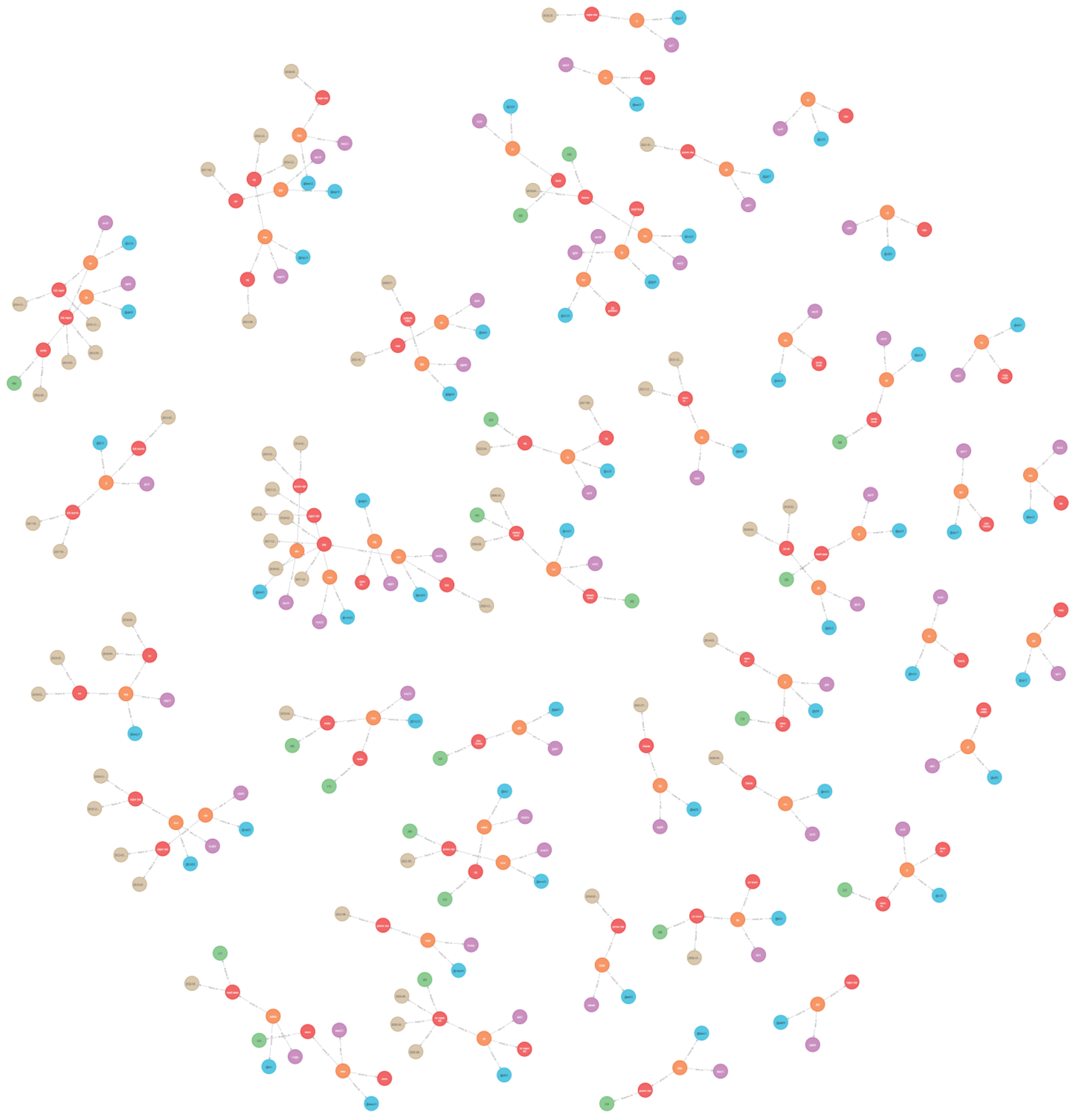
5. User -> Open_on -> Login



6. User-> Posted_In -> Tweet



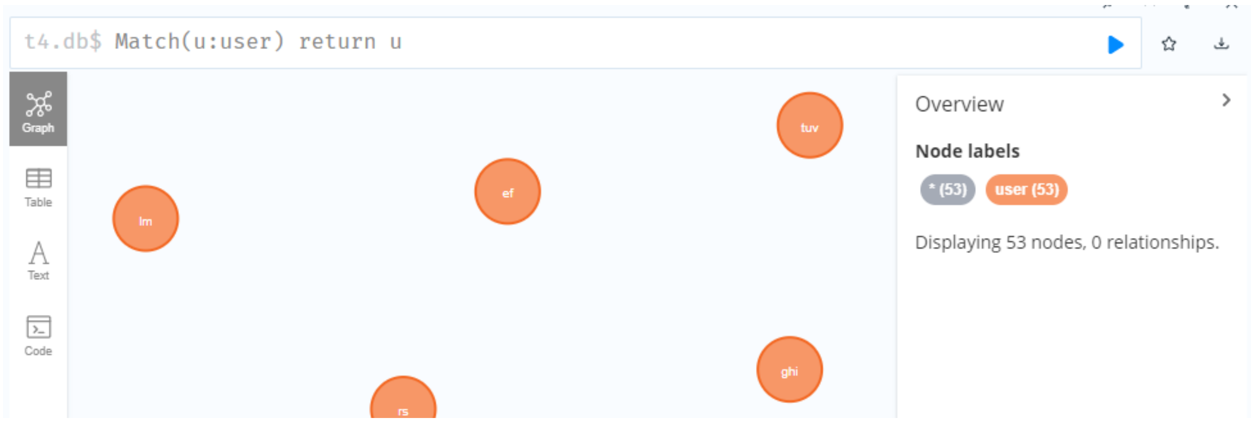
Excuting the Cypher queries to create sample twitter graph database with Node Labels and Relationship Types.



15 Use-Cases in English and Implement the Cypher Queries for Graph Database:

1. Retrieve all the users (nodes).

Match(u:user) return u



```
t4.db$ Match(u:user) return u
```

Overview

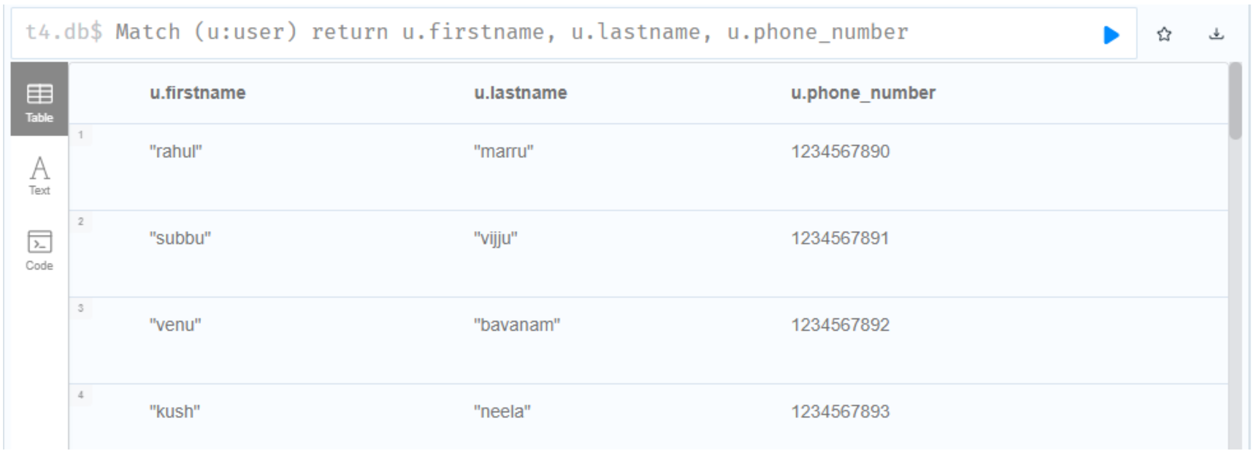
Node labels

* (53) user (53)

Displaying 53 nodes, 0 relationships.

2. List the first name, last name and phone_number of all users.

Match (u:user) return u.firstname, u.lastname, u.phone_number

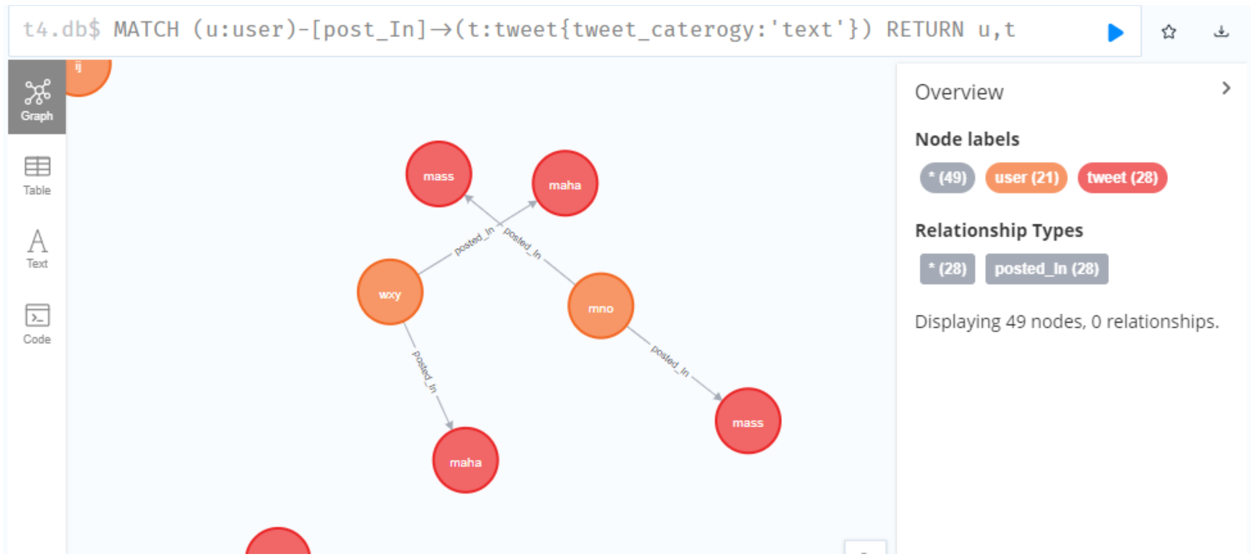


```
t4.db$ Match (u:user) return u.firstname, u.lastname, u.phone_number
```

	u.firstname	u.lastname	u.phone_number
1	"rahul"	"marru"	1234567890
2	"subbu"	"vijju"	1234567891
3	"venu"	"bavanam"	1234567892
4	"kush"	"neela"	1234567893

3. Retrieve all users (nodes) who tweet is with the 'text'.

MATCH (u:user)-[post_In]->(t:tweet{tweet_caterogy:'text'}) RETURN u,t



```
t4.db$ MATCH (u:user)-[post_In]→(t:tweet{tweet_caterogy:'text'}) RETURN u,t
```

Overview

Node labels

* (49) user (21) tweet (28)

Relationship Types

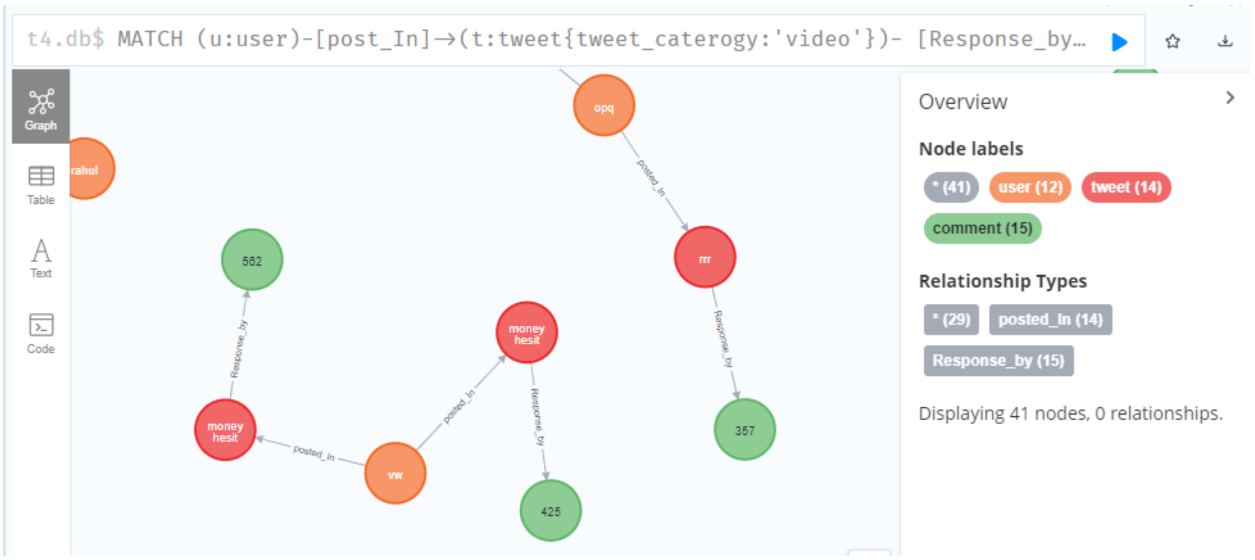
* (28) posted_In (28)

Displaying 49 nodes, 0 relationships.

4. Retrieve all users (nodes) who tweet is with 'video' with corresponding comments.

MATCH (u:user)-[post_In]->(t:tweet{tweet_caterogy:'video'})-

[Response_by]->(c:comment) RETURN u,t,c



```
t4.db$ MATCH (u:user)-[post_In]→(t:tweet{tweet_caterogy:'video'})- [Response_by…
```

Overview

Node labels

* (41) user (12) tweet (14)

comment (15)

Relationship Types

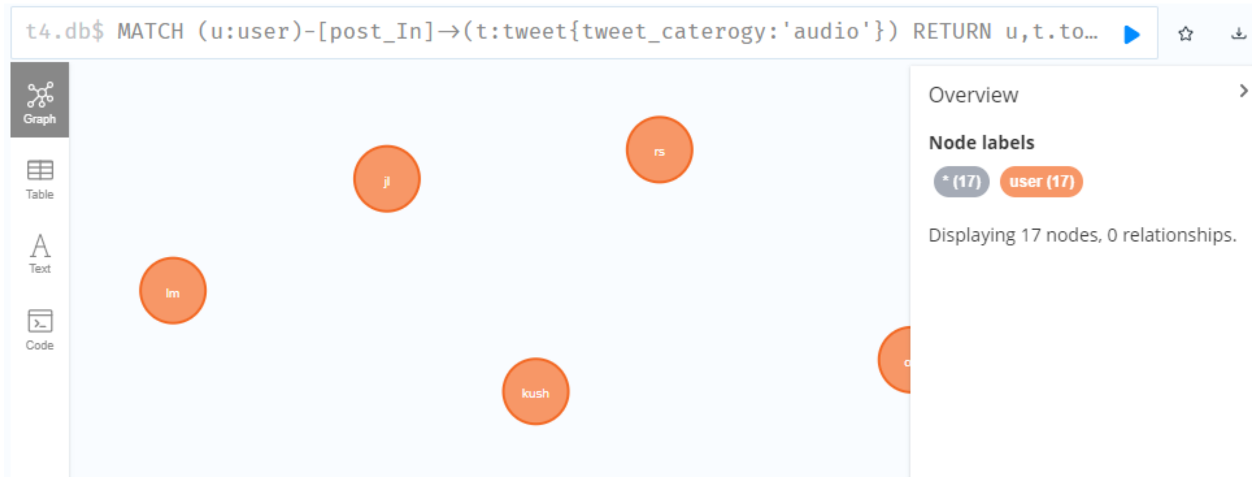
* (29) posted_In (14)

Response_by (15)

Displaying 41 nodes, 0 relationships.

5. Retrieve all users (nodes) who tweet is "audio" and having retweets more than 50.

MATCH (u:user)-[post_In]->(t:tweet{tweet_caterogy:'audio'}) RETURN u,t.total_retweets >= 50

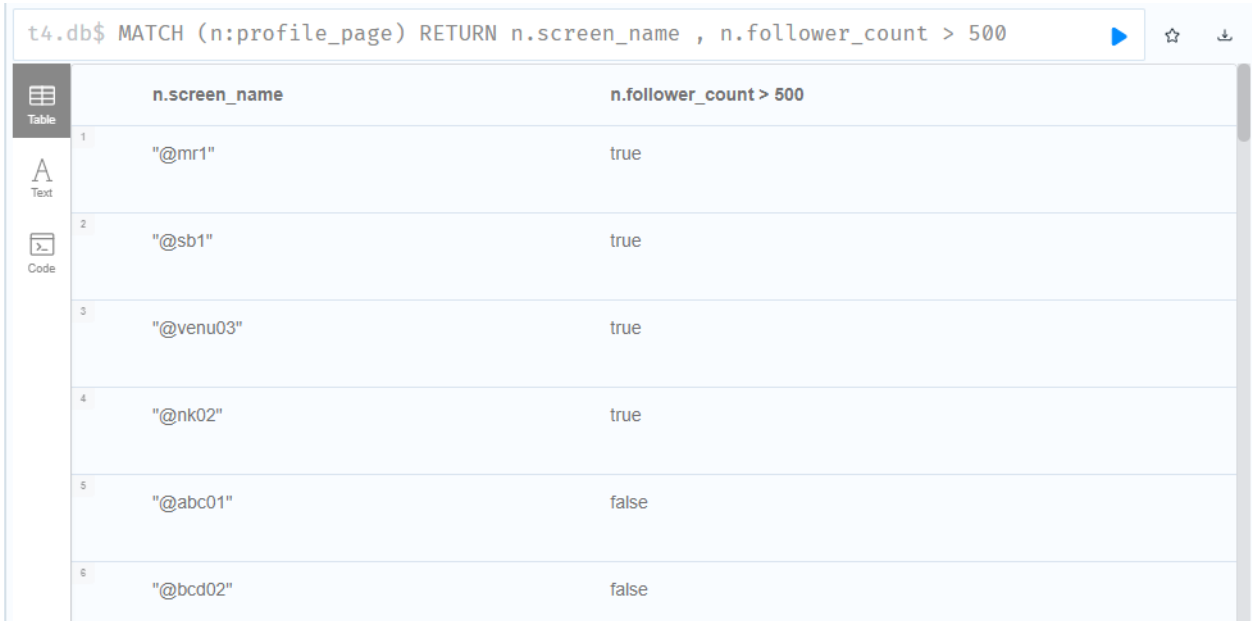


```
t4.db$ MATCH (u:user)-[post_In]→(t:tweet{tweet_caterogy:'audio'}) RETURN u,t.to…
```

Displaying 17 nodes, 0 relationships.

6. Retrive the screen name of the users who's follower count is more than the 500.

MATCH (n:profile_page) RETURN n.screen_name , n.follower_count > 500

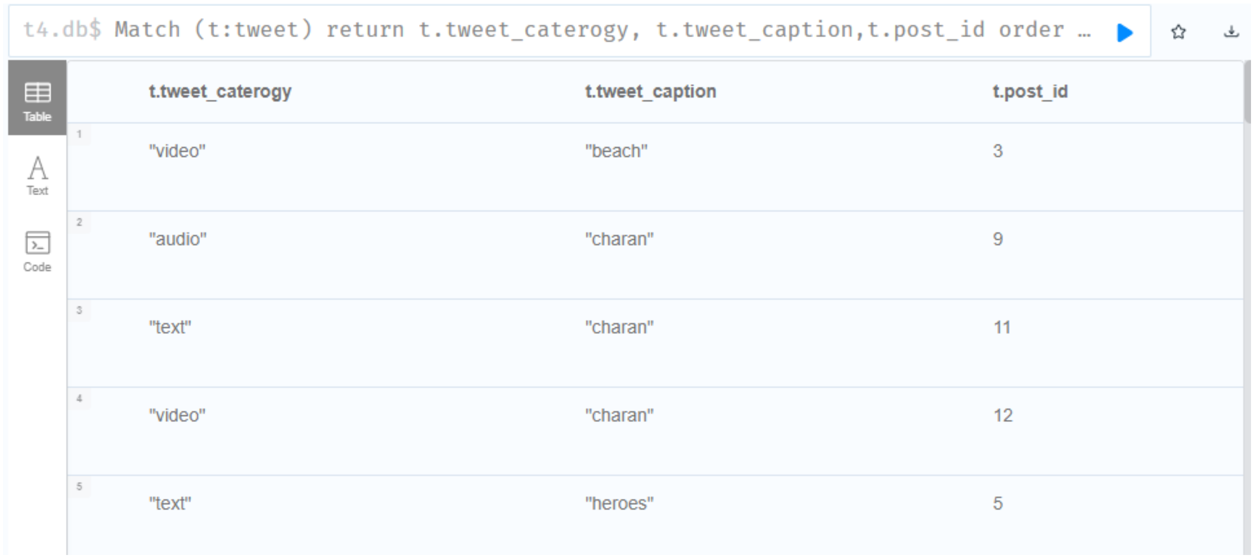


```
t4.db$ MATCH (n:profile_page) RETURN n.screen_name , n.follower_count > 500
```

	n.screen_name	n.follower_count > 500
1	"@mr1"	true
2	"@sb1"	true
3	"@venu03"	true
4	"@nk02"	true
5	"@abc01"	false
6	"@bcd02"	false

7. **Retrive the tweet in ascending order.**

Match (t:tweet) return t.tweet_caterogy, t.tweet_caption,t.post_id order BY t.tweet_caption



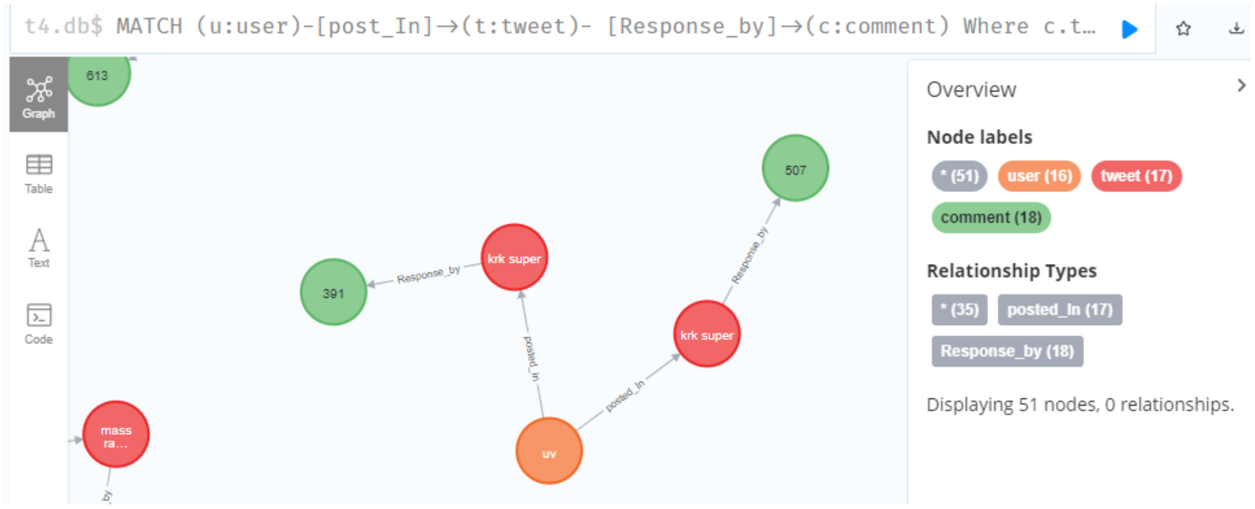
```
t4.db$ Match (t:tweet) return t.tweet_caterogy, t.tweet_caption,t.post_id order …
```

	t.tweet_caterogy	t.tweet_caption	t.post_id
1	"video"	"beach"	3
2	"audio"	"charan"	9
3	"text"	"charan"	11
4	"video"	"charan"	12
5	"text"	"heroes"	5

8. **retrive the users who's having the total like more than 400 for the tweet.**

MATCH (u:user)-[post_In]->(t:tweet)-

[Response_by]->(c:comment) Where c.total_likes >= 400 RETURN u,t,c

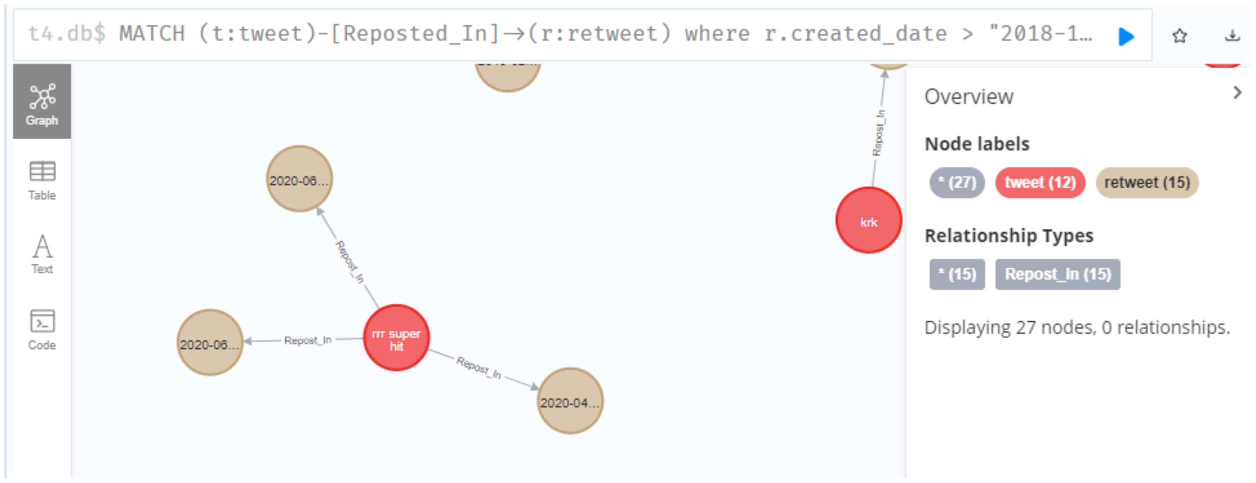


```
t4.db$ MATCH (u:user)-[post_In]→(t:tweet)- [Response_by]→(c:comment) Where c.t…
```

Displaying 51 nodes, 0 relationships.

9. retrive the retweet that created after the 2018 january 1st.

MATCH (t:tweet)-[Reposted_In]->(r:retweet) where r.created_date > "2018-1-1" RETURN t,r



```
t4.db$ MATCH (t:tweet)-[Reposted_In]→(r:retweet) where r.created_date > "2018-1…
```

Overview

Node labels

* (27) tweet (12) retweet (15)

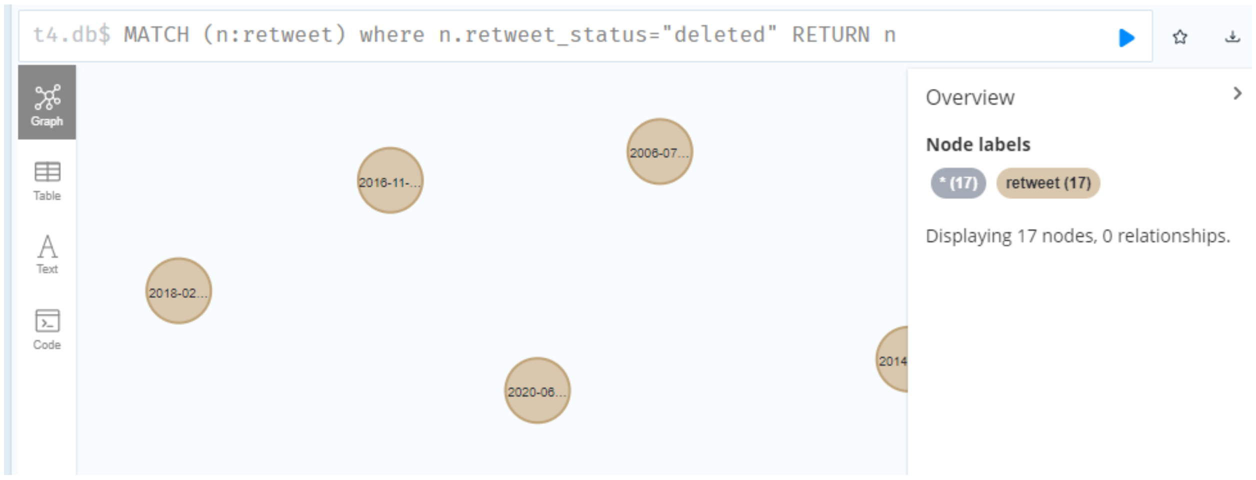
Relationship Types

* (15) Repost_In (15)

Displaying 27 nodes, 0 relationships.

10. retrive the retweets which are first posted and than deleted.

MATCH (n:retweet) where n.retweet_status="deleted" RETURN n



```
t4.db$ MATCH (n:retweet) where n.retweet_status="deleted" RETURN n
```

Overview

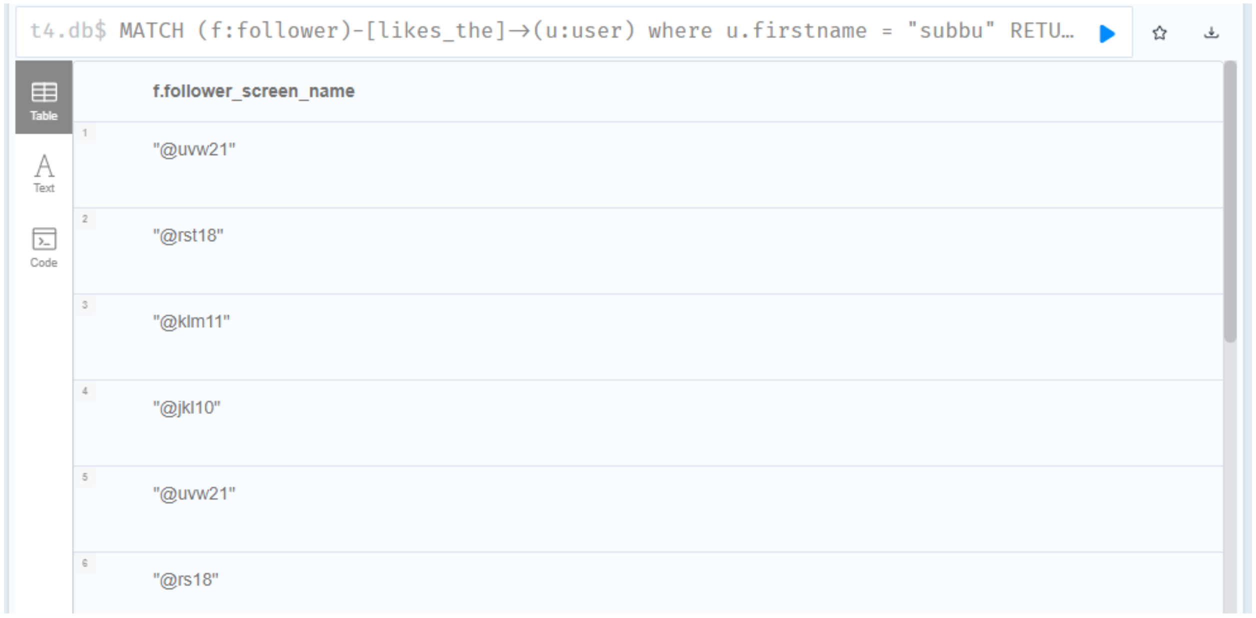
Node labels

* (17) retweet (17)

Displaying 17 nodes, 0 relationships.

11. show the followers for a selected specific user in twitter.

MATCH (f:follower)-[likes_the]->(u:user) where u.firstname = "subbu" RETURN f.follower_screen_name

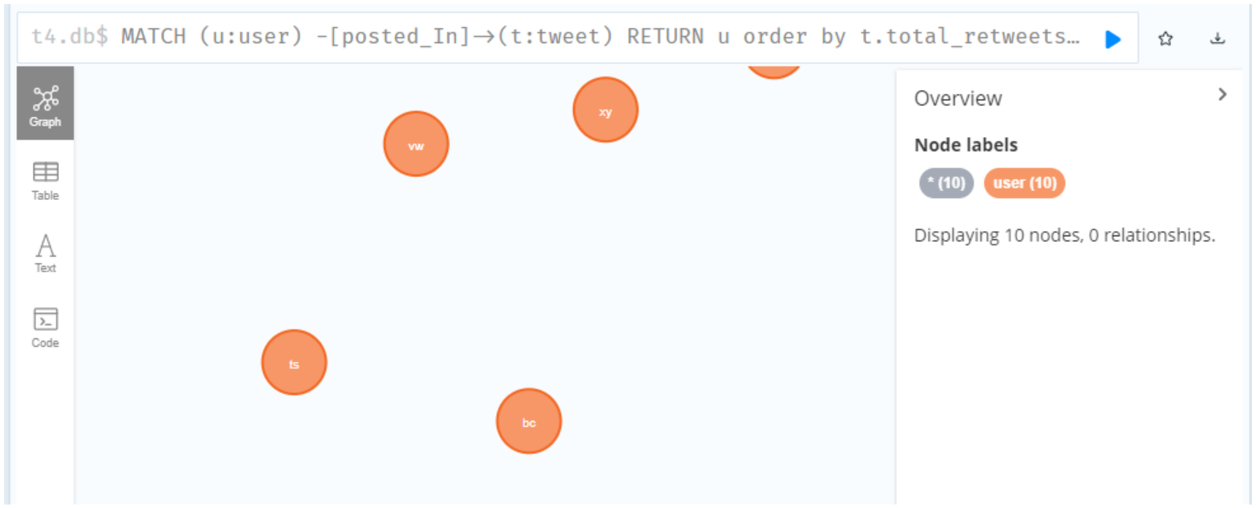


```
t4.db$ MATCH (f:follower)-[likes_the]→(u:user) where u.firstname = "subbu" RETU…
```

	f.follower_screen_name
1	"@uvw21"
2	"@rst18"
3	"@klm11"
4	"@jkl10"
5	"@uvw21"
6	"@rs18"

12. show the top 10 user tweet which has least number of retweets.

MATCH (u:user) -[posted_In]->(t:tweet) RETURN u order by t.total_retweets asc limit 10

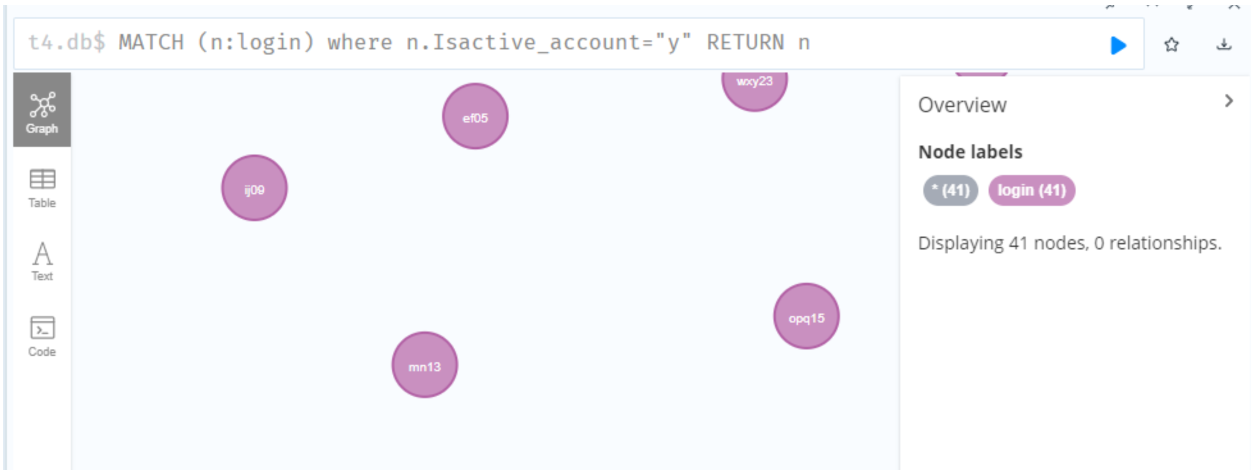


```
t4.db$ MATCH (u:user) -[posted_In]→(t:tweet) RETURN u order by t.total_retweets…
```

Displaying 10 nodes, 0 relationships.

13. Retrive the account that are still active on twitter.

MATCH (n:login) where n.Isactive_account="y" RETURN n

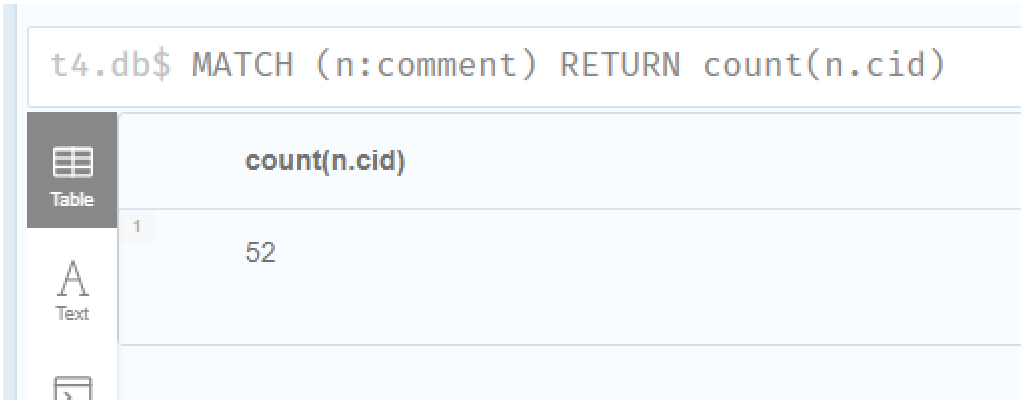


```
t4.db$ MATCH (n:login) where n.Isactive_account="y" RETURN n
```

Displaying 41 nodes, 0 relationships.

14. Retrive the total number of responses given to tweets.

MATCH (n:comment) RETURN count(n.cid)

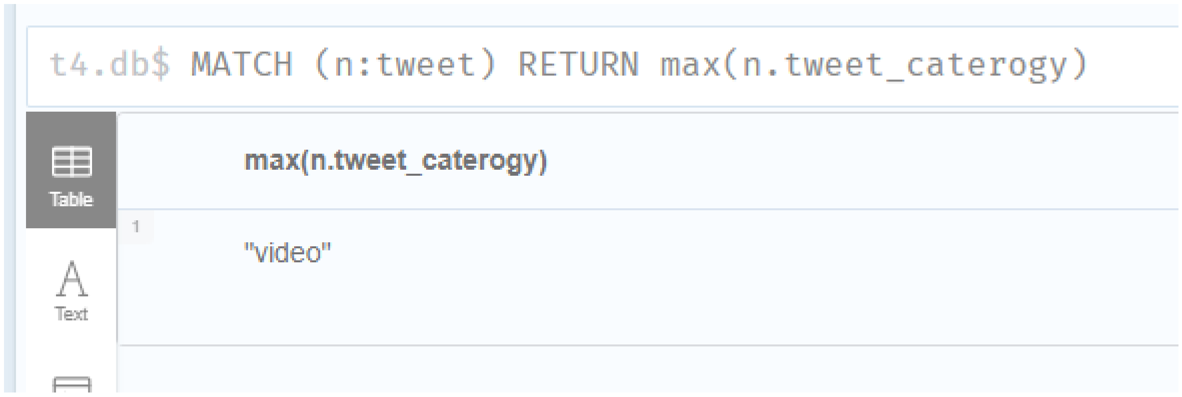


```
t4.db$ MATCH (n:comment) RETURN count(n.cid)
```

	count(n.cid)
1	52

15. Retrive the what caterogy mostly used to tweets/posts.

MATCH (n:tweet) RETURN max(n.tweet_caterogy)



```
t4.db$ MATCH (n:tweet) RETURN max(n.tweet_caterogy)
```

```
max(n.tweet_caterogy)
1  "video"
```

7. Future works:

Implementing the mini web application(dashboard) on this sample twitter database with some plsql procedures, triggers and functions created in the backend of application that help user to design clone twitter application.