# COURSE PROJECT

61050295 SUPHACHOKE WATTHAPIMOL
61050323 URAIWAN JANSONG

# STEP 1

import data from JSON file into MongoDB

---- import 1 json file into Collection ---

- mongoimport --db <db name> --collection <collection name> --type=json --file <file>

---- import many JSON files at a time into Collections using for loop. ----
---- Only available on CMD terminal ----

- for %i in (<file part>) do mongoimport --file %i --type json --db <db name> --collection <collection name>

# STEP 1

import data from JSON file into MongoDB

The result on Local MongoDB

---- command ---

- show dbs
- show collections

```
> show dbs
Rumour   0.000GB
```

```
> show collections
retweets
sourcetweets
```

# STEP 1

import data from JSON file into MongoDB

---- export each collections into JSON file then import MongoDB Atlas --------

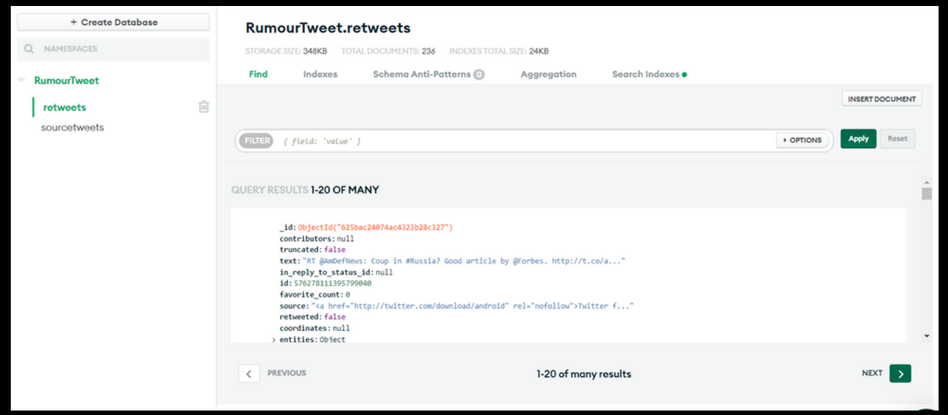
- mongoexport --db <db name> --collection <collection name> --out=<file>

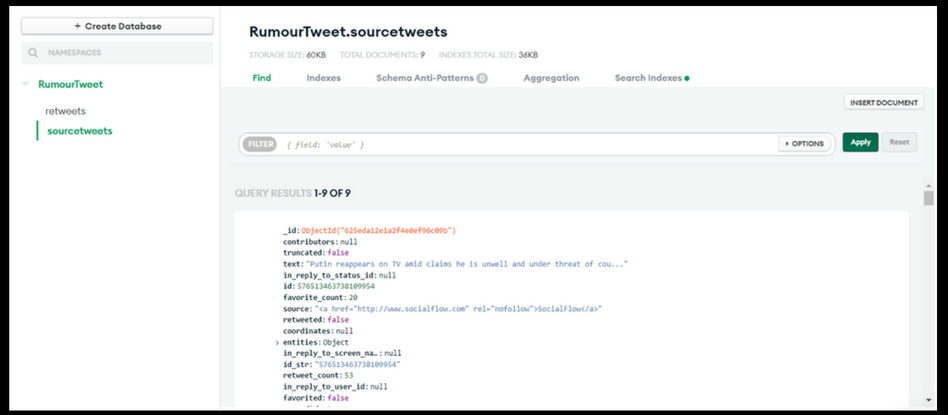
- mongoimport --uri mongodb+srv://<Server_name:password>@cluster0.bed0y.mongodb.net/<db name> --collection <collection name> --type json --file <file>

# STEP 1

import data from JSON file into MongoDB

## The result on MongoDB  Atlas



# STEP 2

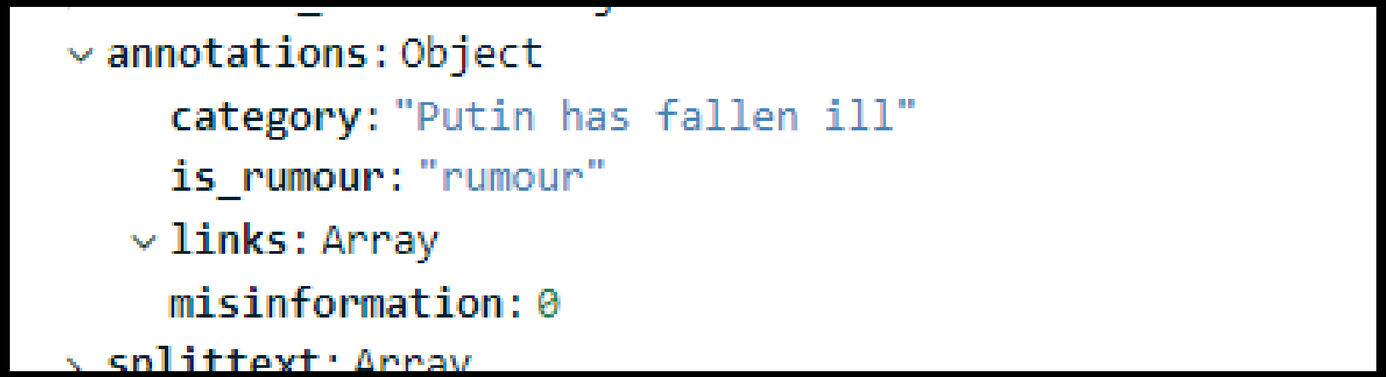
Update each of the source-tweets with labelled annotations specified in the datasets using CRUD commands.

```
db.sourcetweets.update(
      {"id" :  <sourcetweet_id>},
      { $set: {
        "annotations" : {
            "is_rumour":"rumour",
            "category":"Putin is facing a \"palace coup\"",
            "misinformation":0,
            "links":[]
        }
      }
    })
```

# STEP 2

Update each of the source-tweets with labelled annotations specified in the datasets using CRUD commands.

The result on MongoDB Atlas



# STEP 3

1st analysis on the key words and their frequencies

```
---- Creating Word Cloud with word frequency by split text on MongoDb ----
 db.sourcetweets.aggregate(
  [ { $match:
            {"id": <sourcetweet_id>},
            { $project: {
                    splittext:{
                        $split:["$text"," "] }
             }
      },
      { $merge: "sourcetweets" }
 ])
```

# STEP 3: RESULT ON WORD CLOUD

# STEP 4

2nd analysis on how many times each tweet (both rumour and non-rumour)  gets retweeted on social media

Get retweet count for each source tweet using map reduce

```
---- map function to emit ( <source_tweet_id, 1> ) to count retweet ----
var mapTweet = function() {
   emit(this.retweeted_status.id, 1);
};
```

# STEP 4

2nd analysis on how many times each tweet (both rumour and non-rumour)  gets retweeted on social media

Get retweet count for each source tweet using map reduce

```
---- reduce function to count retweet ----
var reduceTweet = function(keyTweet, values) {
   count = 0;
     for(i = 0; i < values.length; i++) {
       count+= values[i];
     }
     return count;
};
```

# STEP 4

2nd analysis on how many times each tweet (both rumour and non-rumour)  gets retweeted on social media

Get retweet count for each source tweet using map reduce

```
---- apply MapReduce function on Collection retweet
and create new Collection retweet_count to store results ----

db.retweets.mapReduce(
   mapTweet ,
   reduceTweet,
   { out: "retweet_count" }
)
```

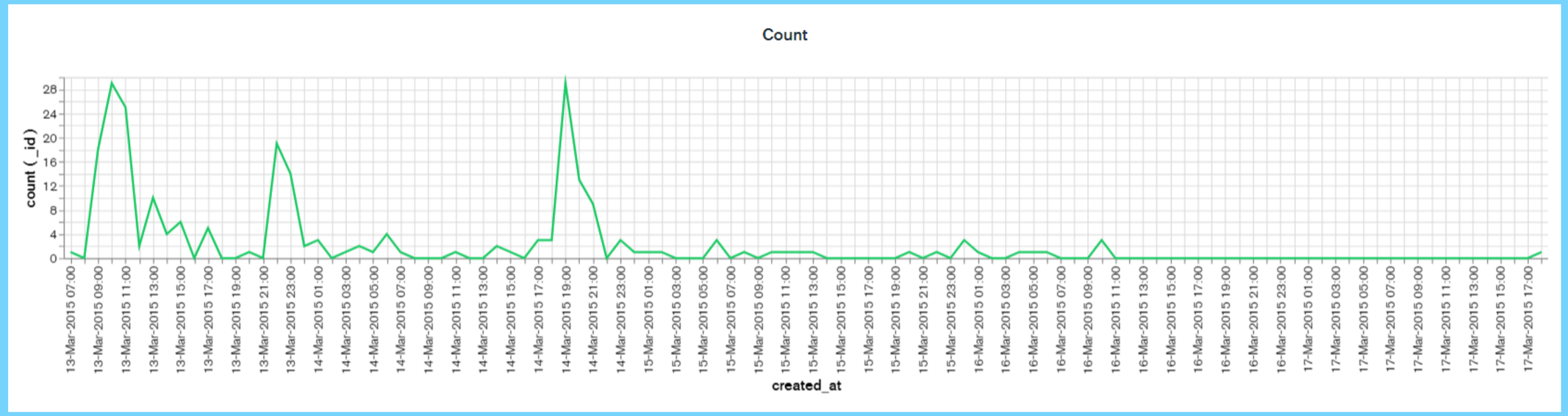
# STEP 4

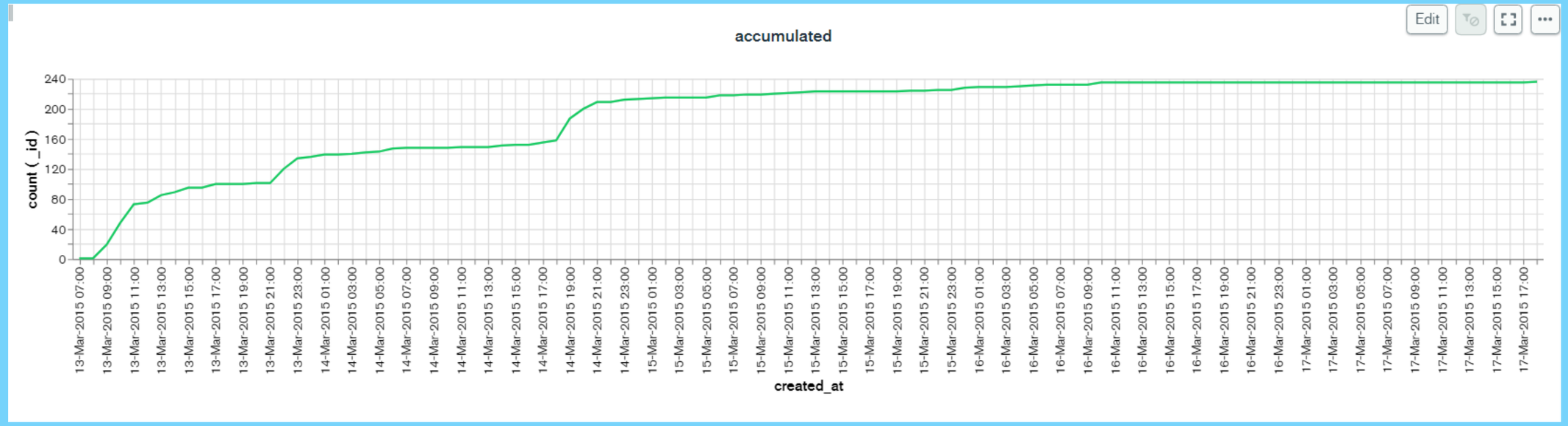
2nd analysis on how many times each tweet (both rumour and non-rumour)  gets retweeted on social media

Get retweet count for each source tweet using map reduce

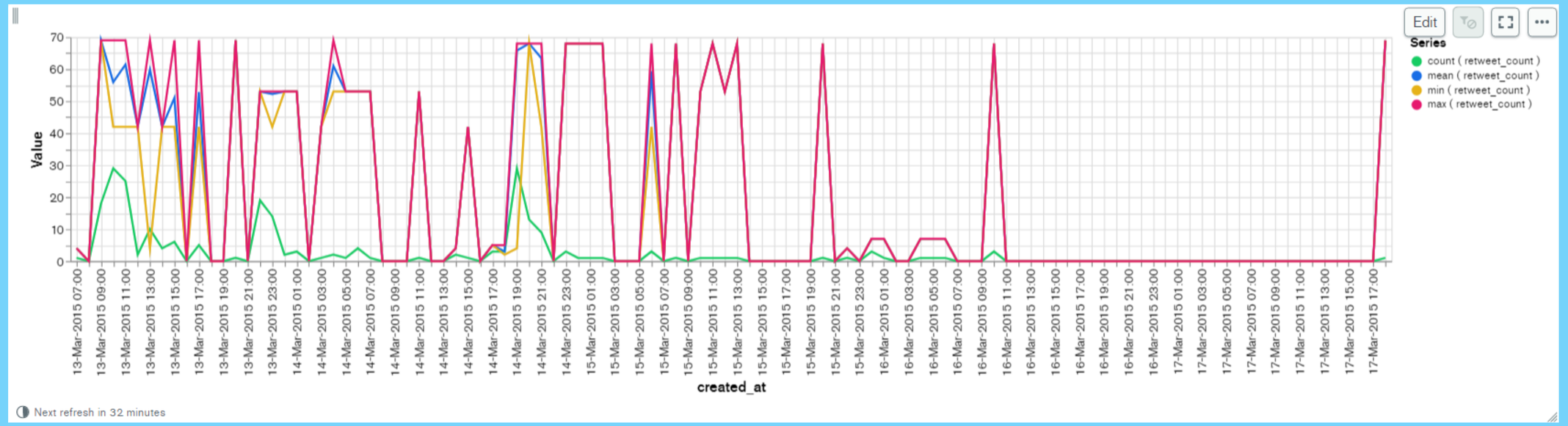
```
> db.retweet_count.find();
{ "_id" : NumberLong("576319832800555008"), "value" : 66 }
{ "_id" : NumberLong("576829262927413248"), "value" : 63 }
{ "_id" : NumberLong("576796432730071040"), "value" : 4 }
{ "_id" : NumberLong("577258317942149120"), "value" : 7 }
{ "_id" : NumberLong("576513463738109954"), "value" : 48 }
{ "_id" : NumberLong("576276947648405505"), "value" : 3 }
{ "_id" : NumberLong("576323086888361984"), "value" : 40 }
{ "_id" : NumberLong("576755174531862529"), "value" : 3 }
{ "_id" : NumberLong("576812998418939904"), "value" : 2 }
>
```

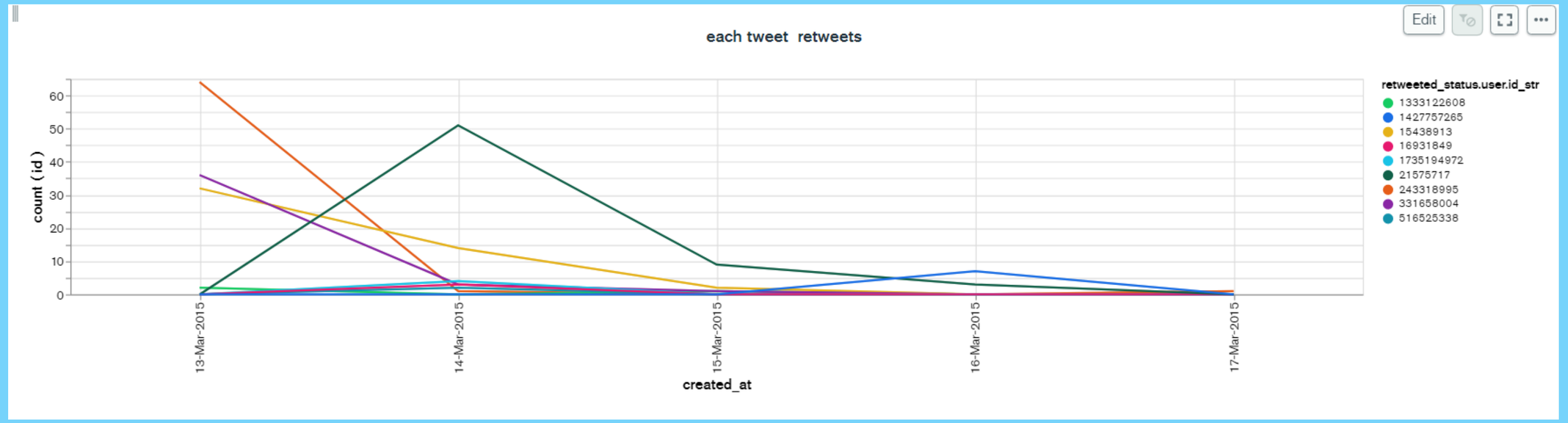
# STEP 5: RESULT CHART



# STEP 5: RESULT CHART



# STEP 6

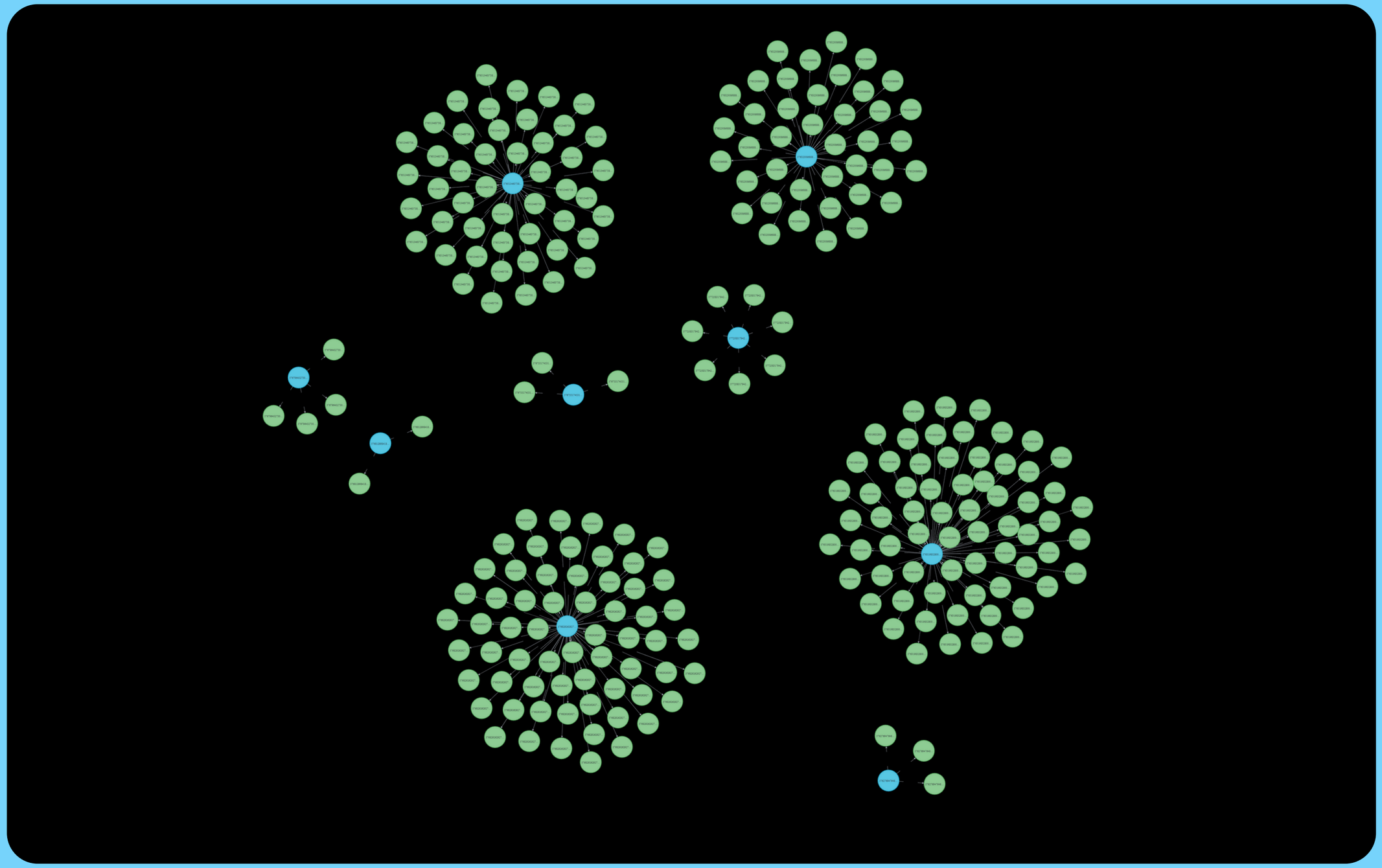
4th analysis on who tweeted the tweets, each of them was retweeted by whom Neo4J

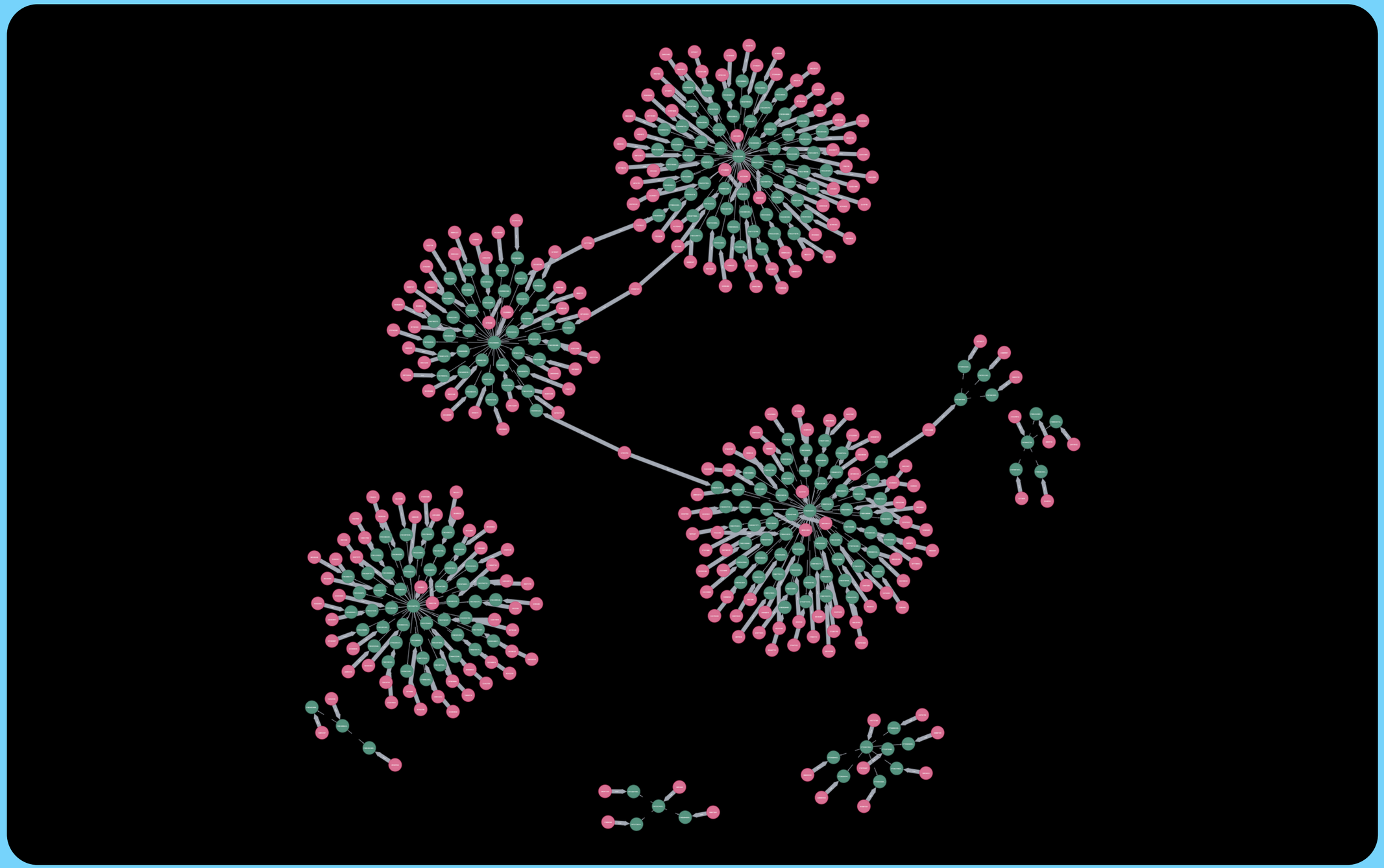
```
---- import  json file from both source tweet and retweet and set value for each node ----

CALL  apoc.load.json("sourcetweets.json")
YIELD value AS sourcetweets
MERGE (s:tweeted {id:sourcetweets.id})

CALL apoc.load.json("retweets.json")
YIELD value  AS retweets MERGE (r:retweeted {id:retweets.id})
SET r.owner=retweets.retweeted_status.id
```

# STEP 6

4th analysis on who tweeted the tweets, each of them was retweeted by whom Neo4J

```
---- Create relation for tweet and retweet ----

MATCH (s:tweeted)
WITH s
MATCH (r:retweeted)
WHERE s.id = r.owner
CREATE (s) - [ :retweeted_by ] -> (r);
```





# STEP 7

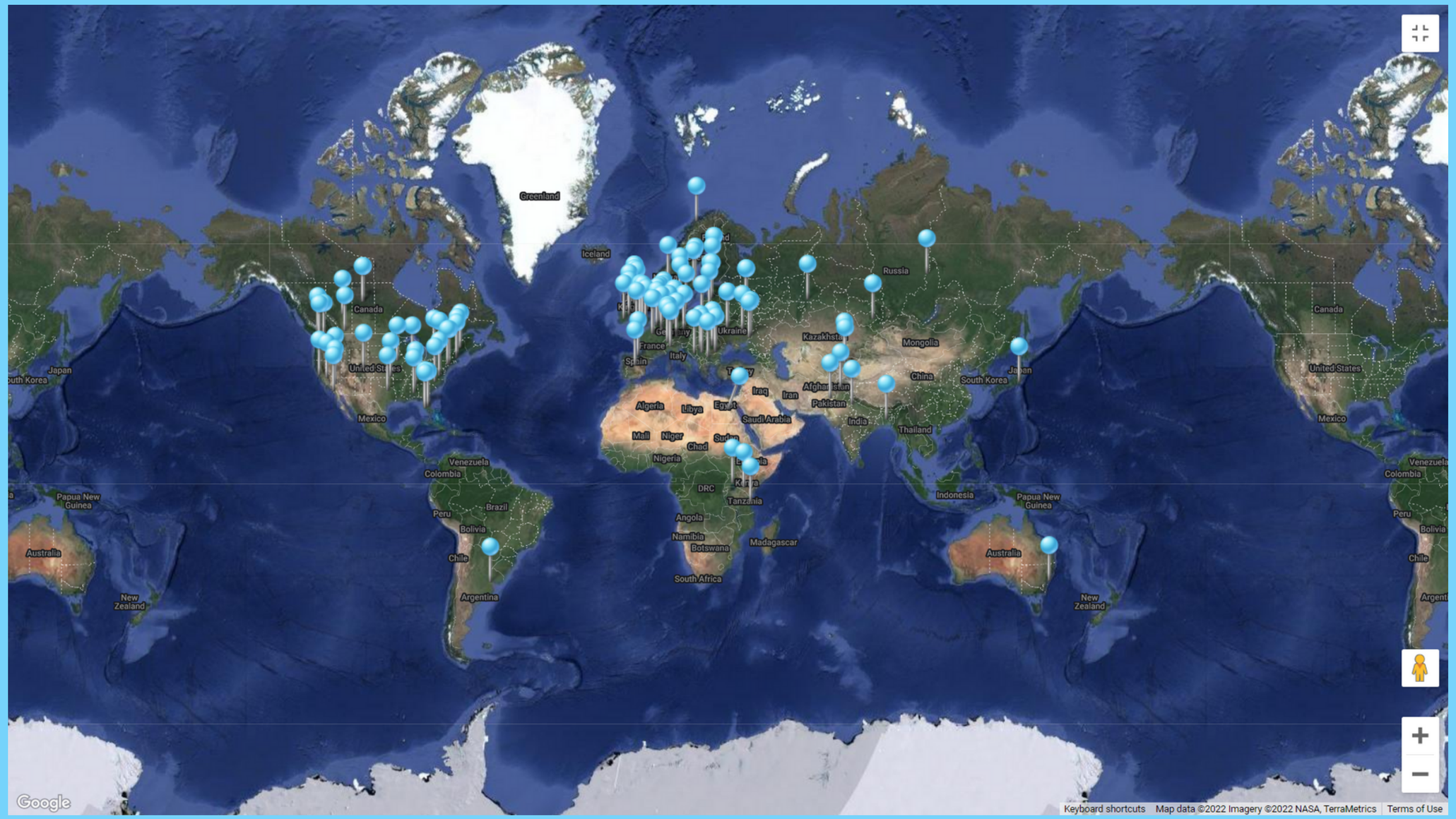
5th analysis on where each tweet is tweeted
by visualizing users' locations on the map

```
<html>
 <head>
 <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
 <script>

  google.charts.load('current', { 'packages': ['map'],
  'mapsApiKey': 'AIzaSyDLhAdGQ1y_Tm8dKGEFham8Ug1ZeA_NcFo'
  });
  google.charts.setOnLoadCallback(drawMap);
```

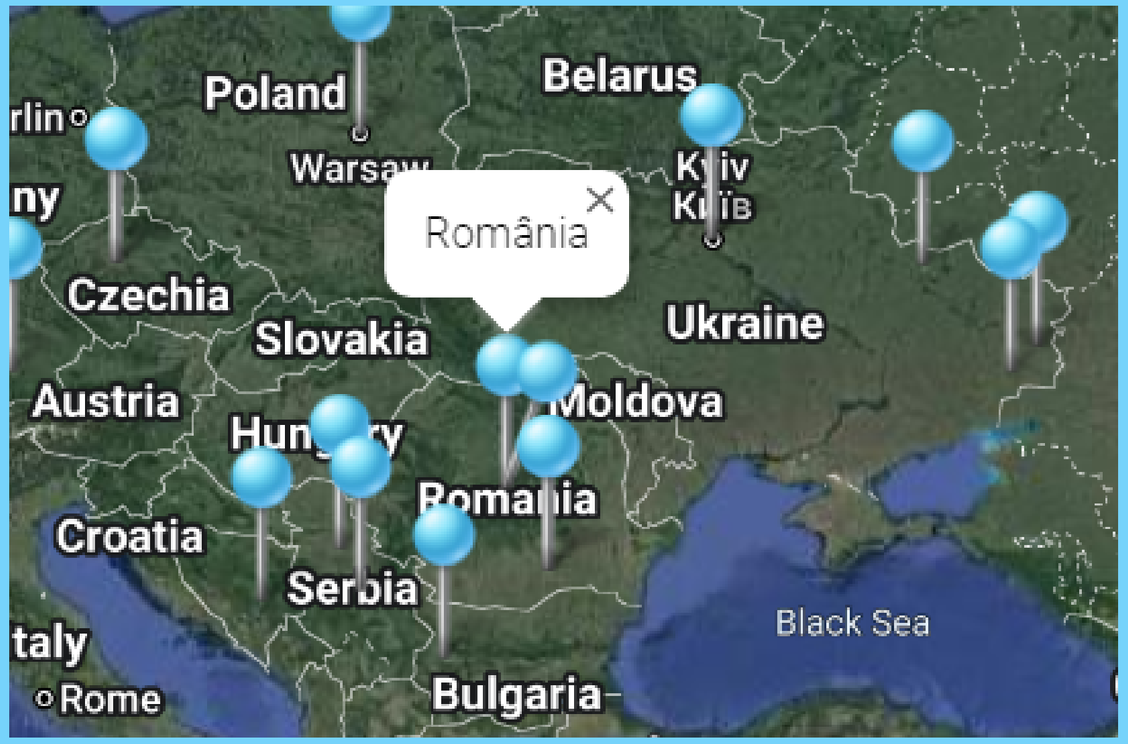
```
function drawMap() {
  var data = google.visualization.arrayToDataTable([
   ['Country', 'name'],
   ['Tokyo', 'Tokyo'],
   ['San Francisco', 'San Francisco'],
   ['West Bromwich', 'West Bromwich'],
   . . .
   . . .
   . . .
   ['Omnipresent', 'Omnipresent'],
   ['Penfield', 'Penfield']
  ]);
```

```
var options = {
    icons: {
      default: {
       normal: 'https://icons.iconarchive.com/icons/icons-land/vista-map-markers/48/Map-
Marker-Ball-Azure-icon.png'
      }
     },
    showTooltip: true,
    showInfoWindow: true
  };

var map = new google.visualization.Map(document.getElementById('chart_div'));

  map.draw(data, options);
 };
 </script>
 </head>
 <body>
  <div id="chart_div"></div>
 </body>
</html>
```

# STEP 7: RESULT MAP



# STEP 7: RESULT MAP TOOL TIP



# THANK YOU!

Have a great day ahead.