

Big Data

Rumour Spreading Analysis on Twittwer

Task 1

Prince
Toronto

Task 1: Tweets

```
mongoimport --uri mongodb+srv://benz1224:raikmitl01@toronto-prince-rumour.v0cqw.mongodb.net/big-data-project --collection tweets --type json --file /Users/benzpreuengineer/Desktop/pheme-rumour-scheme-dataset/threads/en/prince-toronto/529540733020405760/source-tweets/529540733020405760.json
```

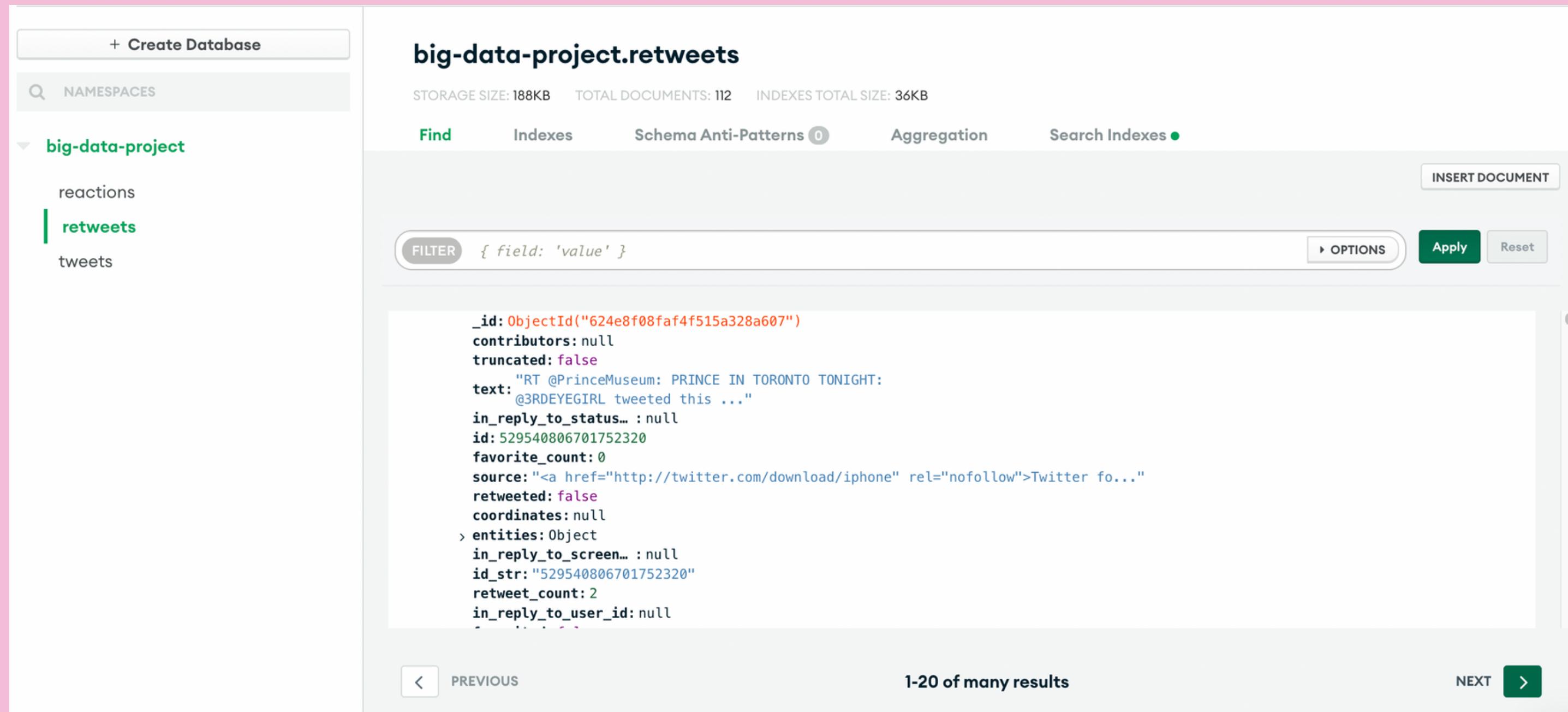


A screenshot of the MongoDB Compass interface showing the 'big-data-project.tweets' collection. The left sidebar shows namespaces: 'big-data-project' (selected), 'reactions', 'retweets', and 'tweets'. The main panel displays document details for a tweet with _id: 624e8b9c91bb827da0b68b77. The document content is:

```
_id: ObjectId("624e8b9c91bb827da0b68b77")
contributors: null
truncated: false
text: "PRINCE IN TORONTO TONIGHT:
@3RDEYEGIRL tweeted this morning to say OTN..."
in_reply_to_status_: null
id: 529540733020405760
favorite_count: 7
source: "<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter fo...""
retweeted: false
coordinates: null
entities: Object
in_reply_to_screen_: null
id_str: "529540733020405760"
retweet_count: 3
in_reply_to_user_id: null
favorited: false
user: Object
geo: null
in_reply_to_user_i... : null
lang: "en"
created_at: "Tue Nov 04 07:48:43 +0000 2014"
```

Task 1: Retweets

```
mongoimport --uri mongodb+srv://benz1224:raikmitl01@toronto-prince-rumour.v0cqw.mongodb.net/big-data-project --collection retweets --type json --file /Users/benzpreuengineer/Desktop/pheme-rumour-scheme-dataset/threads/en/prince-toronto/529540733020405760/retweets.json
```



The screenshot shows the MongoDB Compass interface. On the left sidebar, under the database 'big-data-project', the 'retweets' collection is selected. The main panel displays the 'big-data-project.retweets' collection with the following details:

- STORAGE SIZE: 188KB
- TOTAL DOCUMENTS: 112
- INDEXES TOTAL SIZE: 36KB

The 'Find' tab is active. A search bar at the top contains the query: { field: 'value' }. Below it, there are buttons for 'OPTIONS', 'Apply', and 'Reset'. A single document is expanded to show its full JSON structure:

```
_id: ObjectId("624e8f08faf4f515a328a607")
contributors: null
truncated: false
text: "RT @PrinceMuseum: PRINCE IN TORONTO TONIGHT:
@3RDEYEGIRL tweeted this ..."
in_reply_to_status_: null
id: 529540806701752320
favorite_count: 0
source: "<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter fo...
retweeted: false
coordinates: null
entities: Object
in_reply_to_screen_: null
id_str: "529540806701752320"
retweet_count: 2
in_reply_to_user_id: null"
```

At the bottom, navigation buttons include 'PREVIOUS' and 'NEXT' with arrows, and a center button labeled '1-20 of many results'.



Task 2: CRUD Commands to UPDATE

```
mongo mongodb+srv://toronto-  
prince-  
rumour.v0cqw.mongodb.net/big-  
data-project --username benz1224
```

```
db.tweets.update({id:  
529695367680761856}, {$set:  
{"is_rumour": "rumour",  
"category": "Prince will play a show in  
Toronto on November 4",  
"misinformation": "1", "links": [],  
"is_turnaround": 0}})
```

```
        "is_translator": false  
    },  
    "geo": null,  
    "in_reply_to_user_id_str": null,  
    "possibly_sensitive": false,  
    "lang": "en",  
    "created_at": "Tue Nov 04 18:03:11 +0000 2014",  
    "filter_level": "low",  
    "in_reply_to_status_id_str": null,  
    "place": null,  
    "category": "Prince will play a show in Toronto on November 4",  
    "is_rumour": "rumour",  
    "is_turnaround": 0.0,  
    "links": [],  
    "misinformation": "1"  
}
```

Task 3: Word Cloud

We use array reduction and unwind the array of the 'hashtags' of each tweet '_id' in the data sources and sorted by their values.

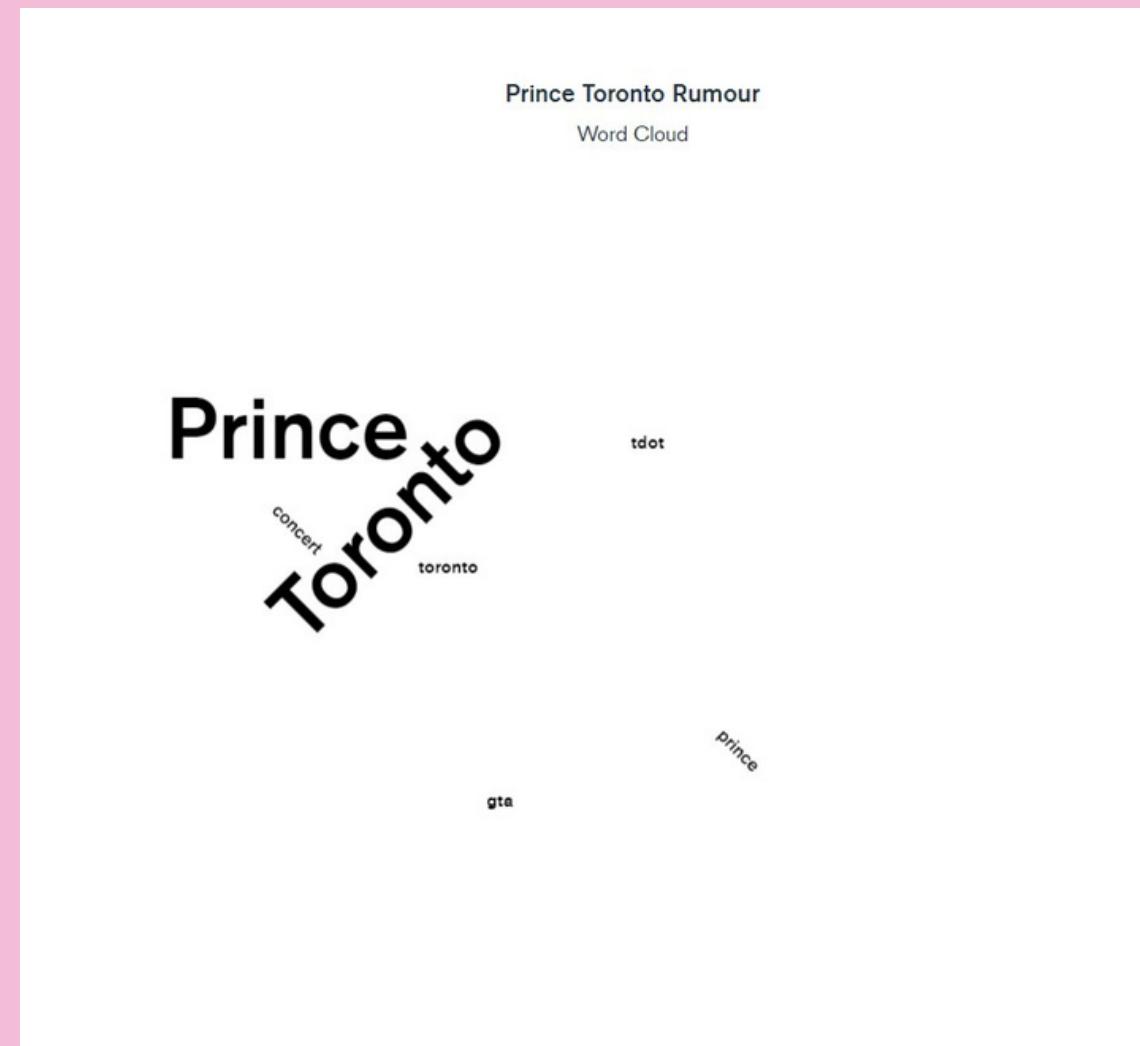
The rumour and non rumour data are filter by query: `{is_rumour: "rumour"}` and `{is_rumour: "non-rumour"}`



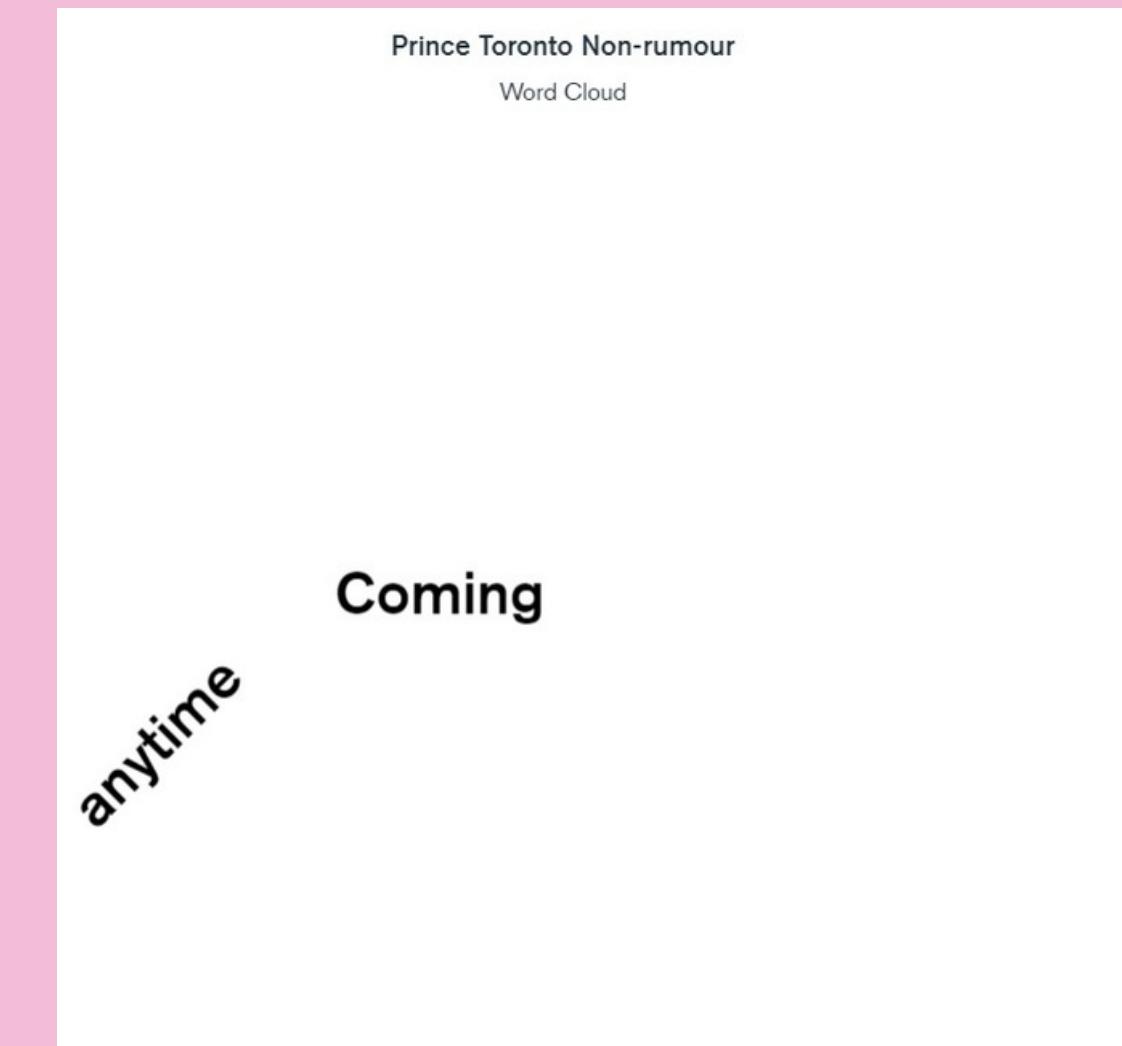
A screenshot of the Tableau Data Source configuration for a Word Cloud. The top navigation bar includes 'Encode', 'Filter', and 'Customize', with 'Encode' being the active tab. The main area is titled 'Text' and contains a section for 'ARRAY REDUCTIONS'. It shows 'hashtags' selected with the 'UNWIND ARRAY' option. There are also sections for 'Binning', 'SORT BY' (set to 'VALUE'), and 'Limit Results'. Below this is a 'Size' section for '_id' with an 'AGGREGATE' dropdown set to 'COUNT'.

Task 3: Word Cloud

OUTPUT RESULTS



Rumour



Non- Rumour

Task 4: Aggregation

```
[  
  {  
    '$project': {  
      '_id': 0,  
      'id': 1,  
      'is_rumour': 1,  
      'retweet_count': 1  
    }  
  }, {  
    '$sort': {  
      'is_rumour': -1,  
      'retweet_count': -1  
    }  
  }]
```

`id: 529720273285566464
retweet_count: 234
is_rumour: "rumour"`

`id: 529713467184676864
retweet_count: 31
is_rumour: "rumour"`

`id: 529739968470867968
retweet_count: 26
is_rumour: "rumour"`

`id: 529653029747064832
retweet_count: 12
is_rumour: "rumour"`

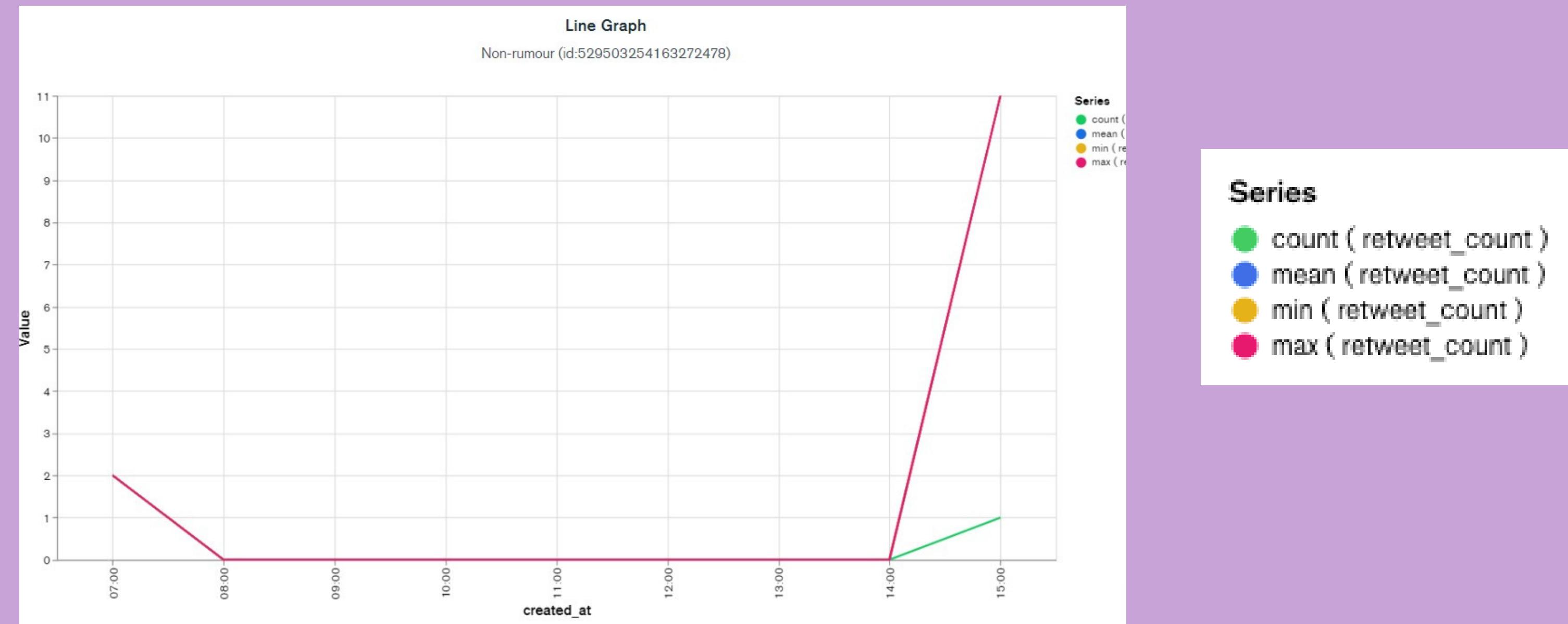
Query

Top 4 results after aggregation

Task 5: Non-rumour

For non rumour retweets, we use **Filter Function** to filtered out all of the rumour ones and left only non-rumours in the line chart

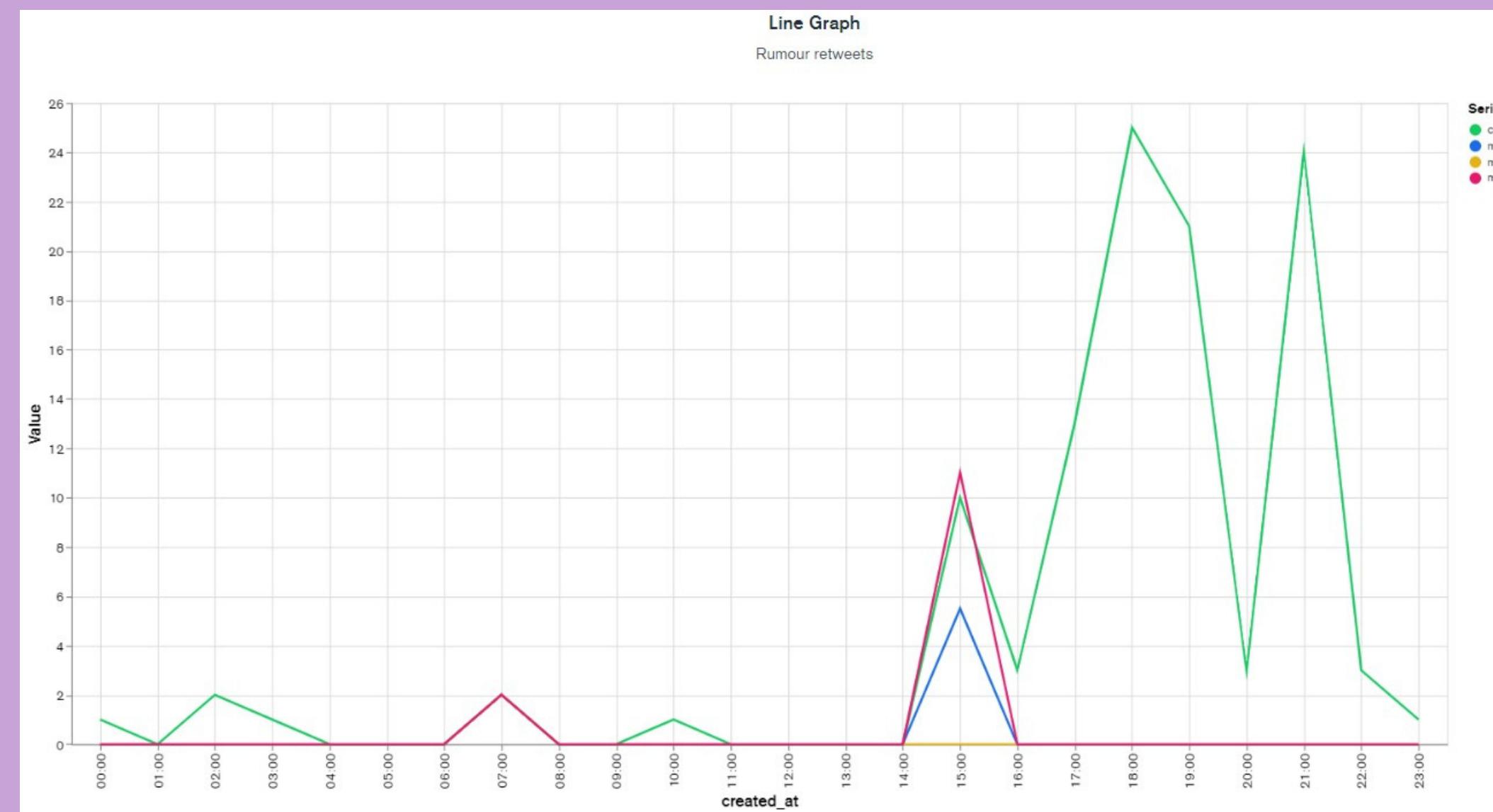
- X Coordinate: Count of retweet id
- Y Coordinate: Created Time (Hours)



Task 5: Rumour

For rumour retweets, we Filter Function to filtered out all of the rumour ones and left only rumours in the line chart

- X Coordinate: Count/Mean/Max/Min of tweet id
- Y Coordinate: Created Time (Hours)



Series

- count (retweet_count)
- mean (retweet_count)
- min (retweet_count)
- max (retweet_count)

Task 6

Exporting Data

```
mongoexport --uri  
mongodb+srv://benz1224:raikmitl01  
@toronto-prince-  
rumour.v0cqw.mongodb.net/big-  
data-project --collection tweets --  
type json --out  
/Users/benzpreuengineer/Desktop/  
pheme-rumour-scheme-  
dataset/Prince/tweets.json
```

```
mongoexport --uri  
mongodb+srv://benz1224:raikmitl01  
@toronto-prince-  
rumour.v0cqw.mongodb.net/big-  
data-project --collection retweets -  
-type json --out  
/Users/benzpreuengineer/Desktop/  
pheme-rumour-scheme-  
dataset/Prince/retweets.json
```



Task 6

Creating nodes

```
CALL apoc.json.load($tweets)
YIELD value AS v
UNWIND v.user AS u
CREATE (n:User:Tweet {id:u.id_str, name:u.name, post:v.text, post_id:v.id_str})
```

```
CALL apoc.json.load($retweets)
yield value AS v
UNWIND v.user AS u
UNWIND v.retweeted_status AS r
CREATE (n:User:Retweet {id:u.id_str, name:u.name, post:v.text, post_id:v.id_str, retweeted:r.user.name,
retweeted_id:r.id_str})
```

Creating relationships

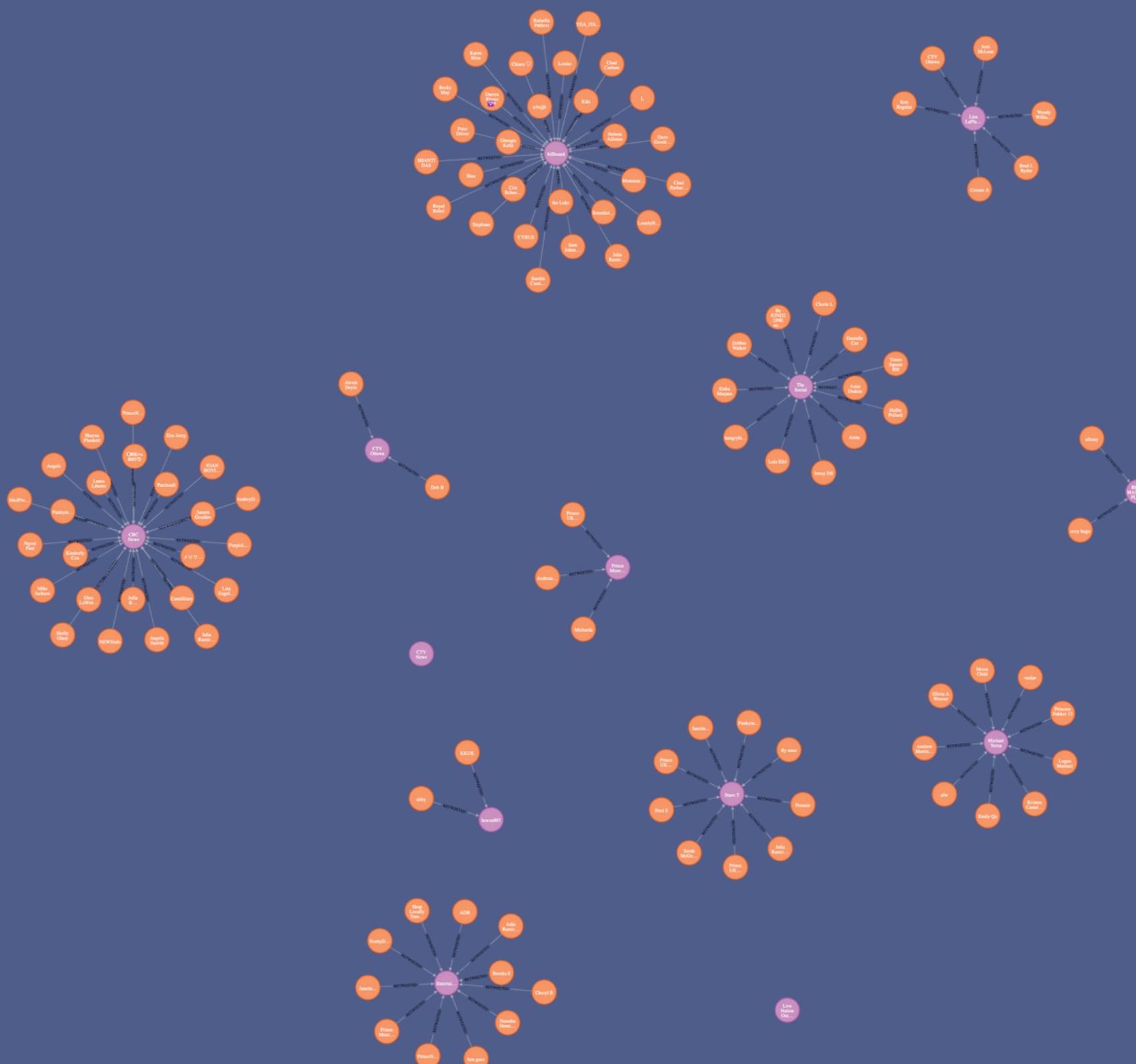
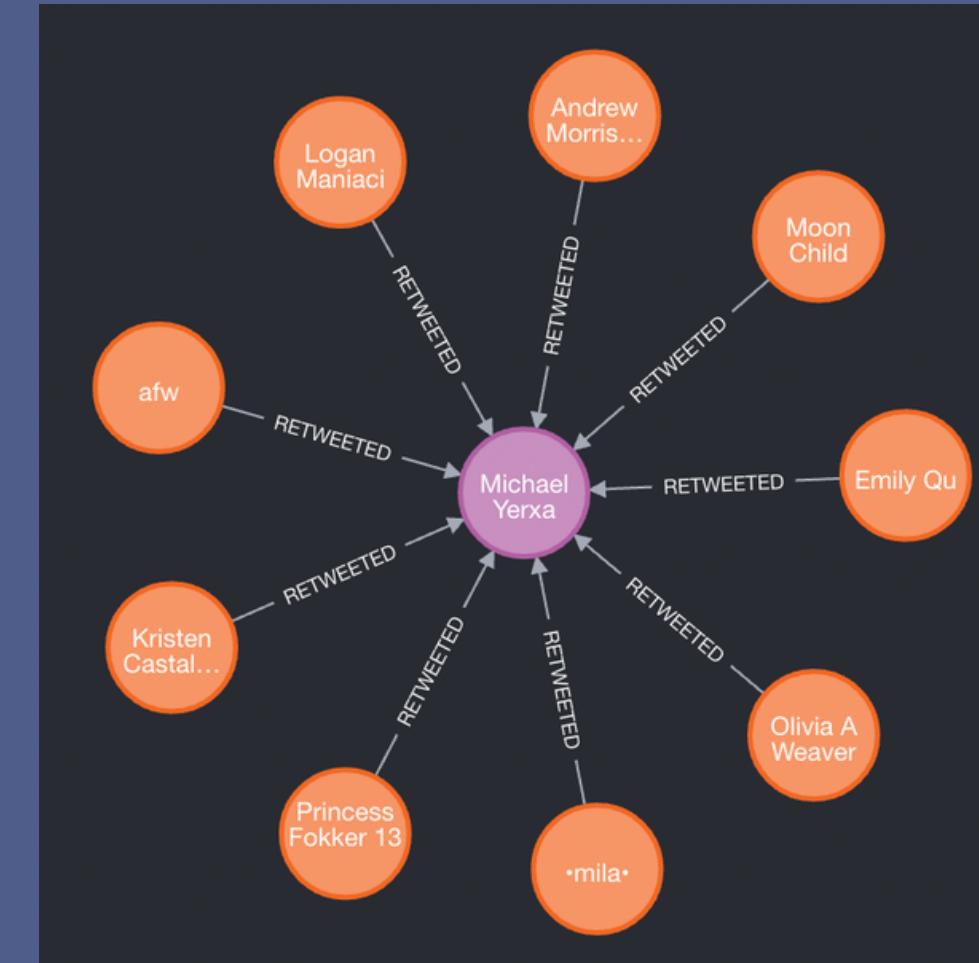
```
MATCH (a:User), (b:Retweeted)
WHERE a.id = b.retweeted_id
CREATE (b)-[r:RETWEETED]->(a)
```



Task 6: Tweets & Retweets

Graph database of who tweeted each tweets and each of them retweeted by whom

MATCH (n) RETURN n

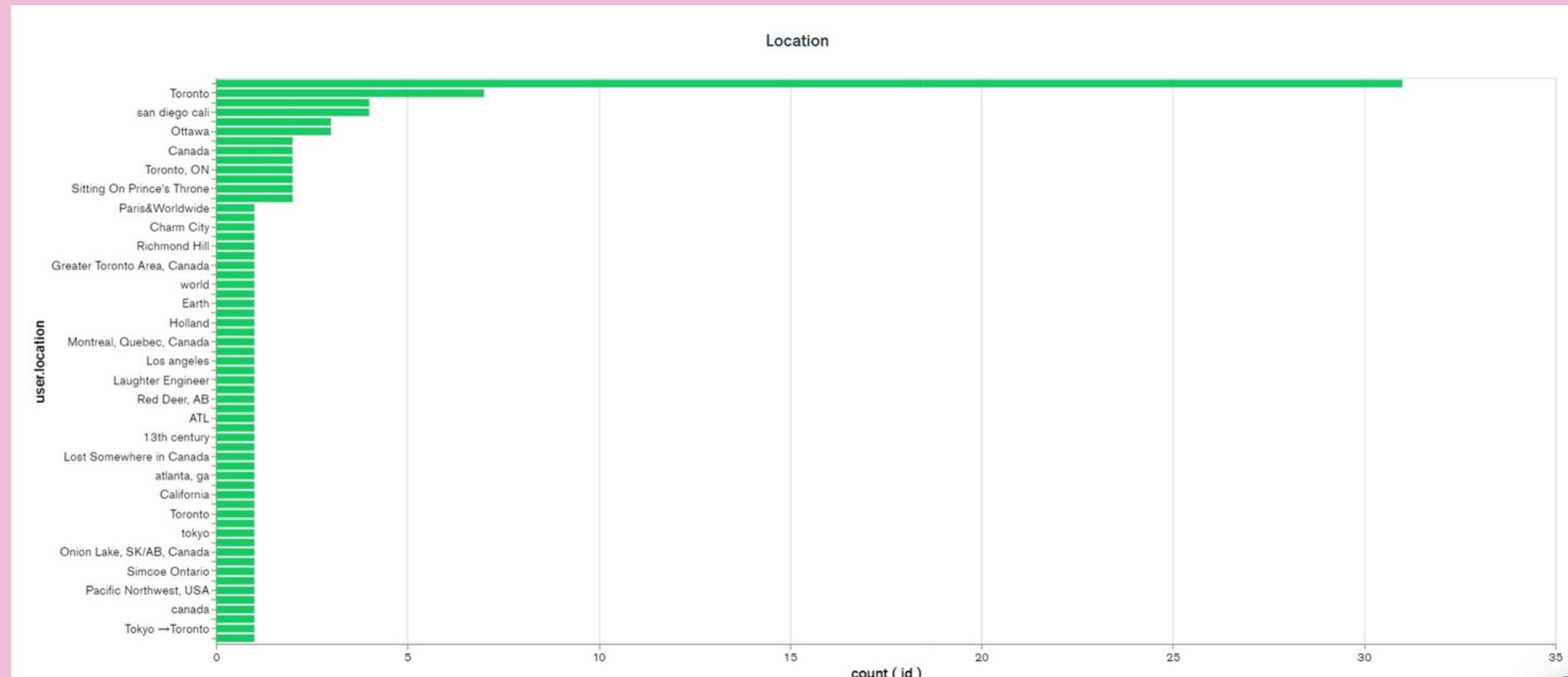


Task 7: Users' locations

We use Bar chart to visualize the location of each tweet instead of Geospatial. This is due to missing information in the given data source. For example; no country coordinates or country codes.

Line Graph

- X Coordinate: Count/Mean/Max/Min of tweet id
- Y Coordinate: Location sorted by value



Visualization of Locations of each tweet

Task 8: Converting Script (1)

```
import os
import json

# file path to the dataset
dataset = 'prince-toronto'
path_file = '/Users/benzpreuengineer/Desktop/pheme-rumour-scheme-dataset/threads/en/' + dataset

# Access the tweet (original tweet structure, folder)
def Tweet(data, file):

    os.chdir(path_file + '/' + file + '/source-tweets')
    user = open(file + '.json')
    user_data = json.load(user)

    tweet = {
        "id": file,
        "text": user_data["text"],
        "user": user_data["user"]["name"],
        "user_id": user_data["user"]["id_str"],
        "replies": []
    }
    # if there is nested object
    if data[file] != []:
        Hierarchical(data[file], file, file, tweet['replies'])

    return tweet
```

Converting script
to convert a
structure file into
the
corresponding
tweet and its
hierarchical
replies





Task 8: Converting Script (2)

```
# Access nested object (object of previous tweet, reply to whom, folder that contains data, array to store replies)
def Hierarchical(origin, pretweet, file, reply_list):

    replies = list(origin.keys())
    for reply in replies:

        os.chdir(path_file + '/' + file + '/reactions')
        user = open(reply + '.json')
        user_data = json.load(user)
        post = {
            "id": reply,
            "text": user_data["text"],
            "user": user_data["user"]["name"],
            "user_id": user_data["user"]["id_str"],
            "replied": pretweet
        }
        reply_list.append(post)

        if origin[reply] != []:
            Hierarchical(origin[reply], reply, file, reply_list)

# convert the object to json (object, iteration)
def convert_json(new_data, index):
    filename = 'reactions' + str(index) + '.json'

    os.chdir(path_file + '/reactions')
    with open(filename, 'w') as r:
        json.dump(new_data, r, indent=4)

# change directory to prince toronto
os.chdir(path_file)
```

Converting script
to convert a
structure file into
the
corresponding
tweet and its
hierarchical
replies



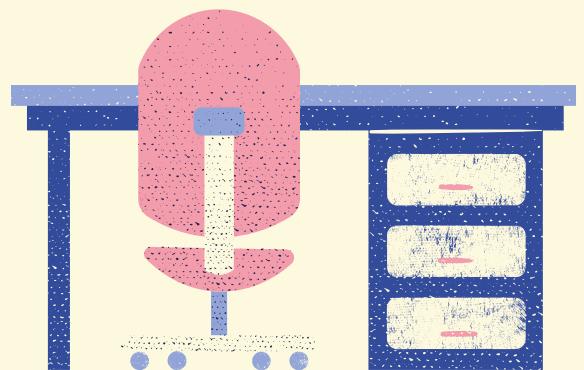
Task 8: Converting Script (3)

```
num = 0
# access to each tweet
for folder in os.listdir():
    if folder == '.DS_Store' or folder == 'reactions':
        continue

    os.chdir(path_file + '/' + folder)
    structure = open('structure.json')
    s = json.load(structure)

    convert_json(Tweet(s, folder), num)
    print("reaction file No.{} has been successfully created!".format(num))
    num += 1
```

Converting script
to convert a
structure file into
the
corresponding
tweet and its
hierarchical
replies





Task 8: Converting Script (4)

Import Data to MongoDB

```
mongoimport --uri mongodb+srv://benz1224:raikmitl01@toronto-prince-rumour.v0cqw.mongodb.net/big-data-project --collection reactions --type json --file /Users/benzpreuengineer/Desktop/pheme-rumour-scheme-dataset/threads/en/prince-toronto/reactions/reactions0.json
```

The screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases and collections: '+ Create Database', 'big-data-project' (selected), 'reactions' (selected), 'retweets', and 'tweets'. The main area is titled 'big-data-project.reactions' and shows storage details: STORAGE SIZE: 44KB, TOTAL DOCUMENTS: 12, INDEXES TOTAL SIZE: 36KB. It includes tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A 'FILTER' bar contains the query '{ field: 'value' }'. Buttons for 'OPTIONS', 'Apply', and 'Reset' are present. An 'INSERT DOCUMENT' button is at the top right. Below the filter, the text 'QUERY RESULTS 1-12 OF 12' is displayed. Two documents are shown:

```
_id: ObjectId("6264f9fe0a8559b2188aa0ef")
id: "529660296080916480"
text: "Prince is playing a secret show in Toronto tonight. I will also be pl..."
user: "Michael Yerxa"
user_id: "119841769"
> replies:Array
```



```
_id: ObjectId("6264fa1c229dbadf8f203a05")
id: "529653029747064832"
text: "Prince fans lining up at Massey Hall. Wristbands for surprise show rep..."
user: "Entertainment City"
user_id: "569128022"
> replies:Array
```



Task 9 - Tweets & Replies

Export Data

```
mongoexport --uri mongodb+srv://benz1224:raikmitl01@toronto-prince-rumour.v0cqw.mongodb.net/big-data-project --collection reactions --type json --out /Users/benzpreuengineer/Desktop/pheme-rumour-scheme-dataset/Prince/reactions.json
```

Creating nodes

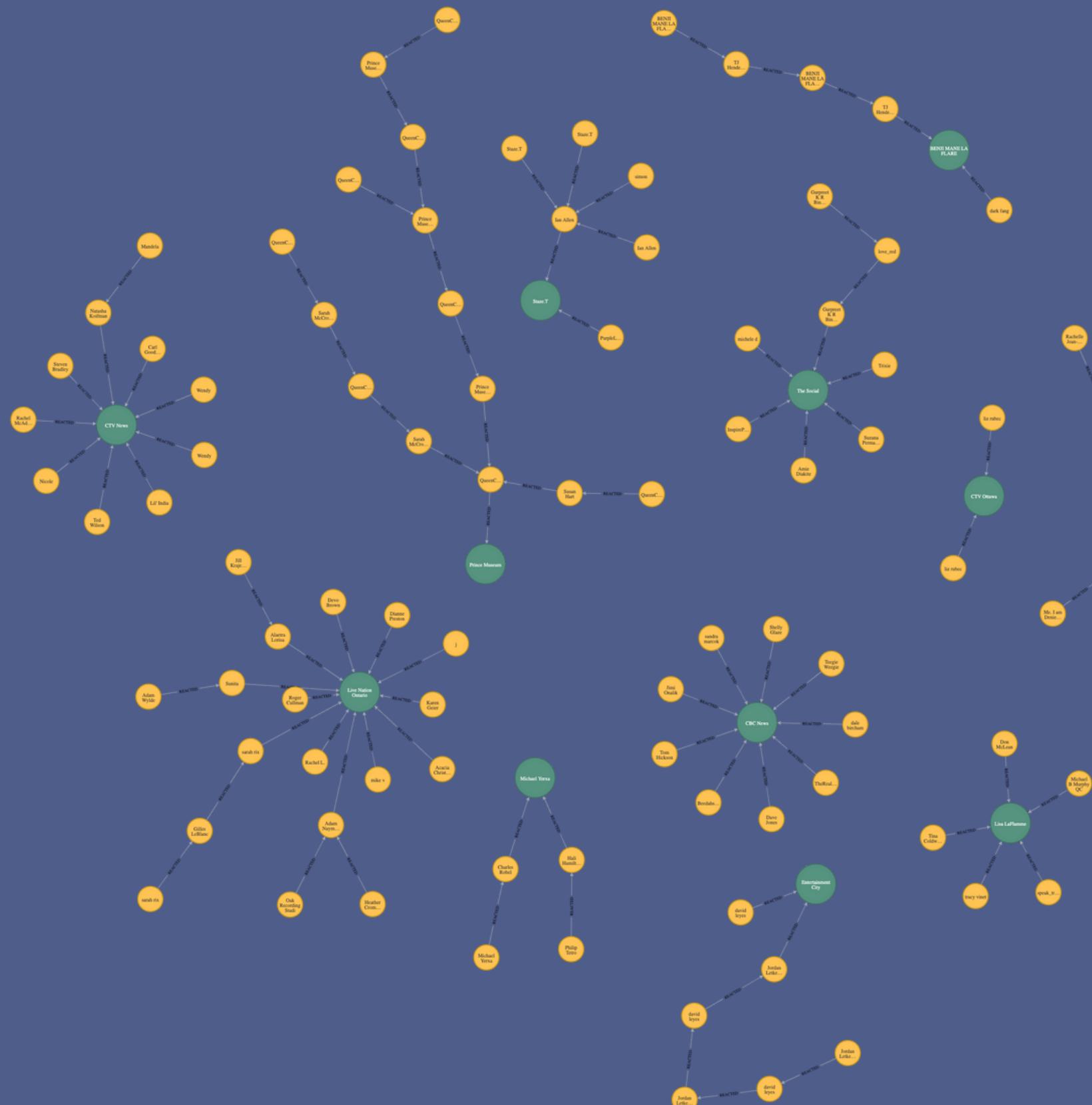
```
CALL apoc.json.load($reactions)
YIELD value AS v
UNWIND v.replies AS r
CREATE (n:User:Tweet {id:v.id, name:v.user, user_id:v.user_id, text:v.text})
MERGE (m:User:Reaction {id:r.id, name:r.user, user_id:v.user_id, text:r.text, replied:r.replied})
```

Creating relationships

```
MATCH (a:User), (b:Reaction)
WHERE a.id = b.replied
CREATE (b)-[r:REACTED]->(a)
```

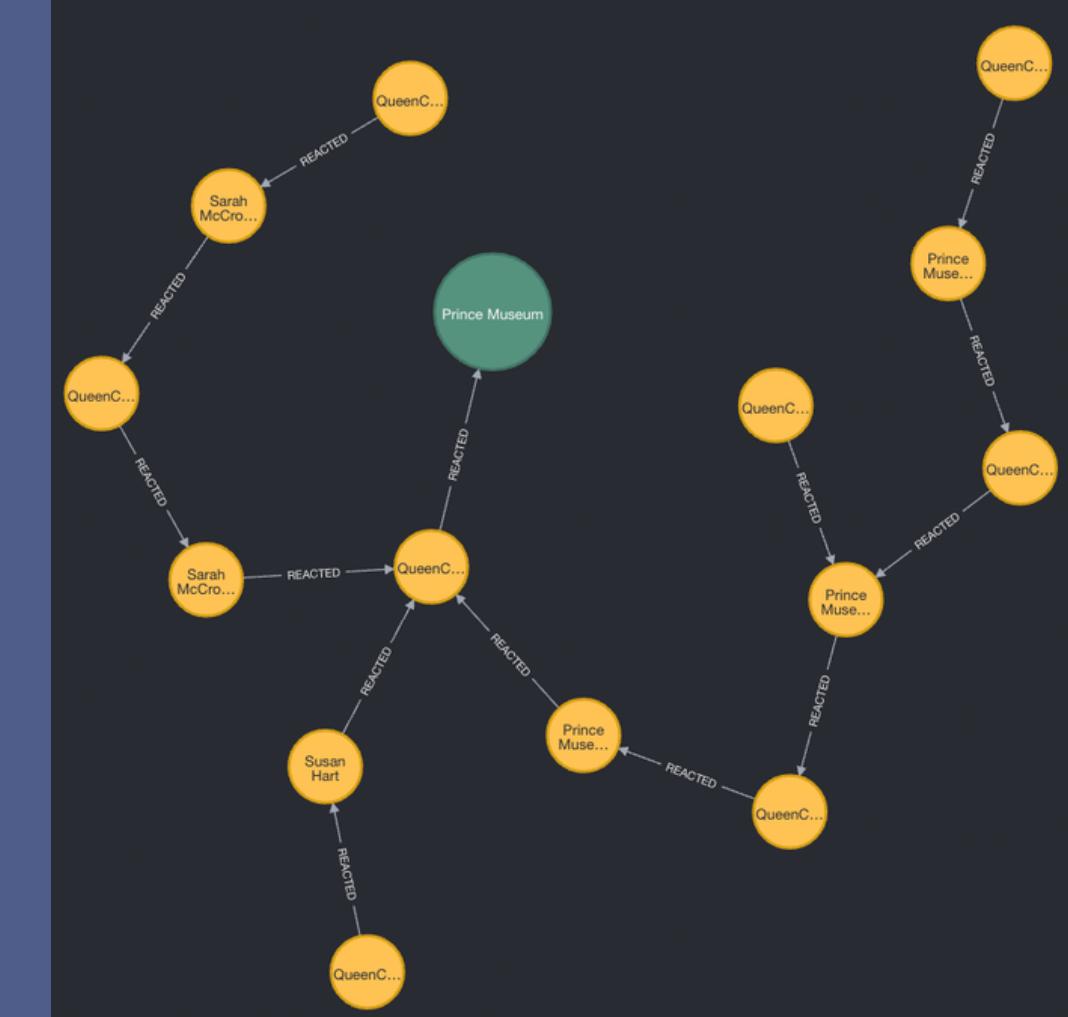


Task 9 - Tweets & Replies



Hierarchical
graph database
on each tweets
and their replies

MATCH (n) RETURN n



Database

MongoDB

Write the command in terminal

```
mongo mongodb+srv://toronto-prince-rumour.v0cqw.mongodb.net/big-data-project --username guest-sp
```

```
password: raikmitl99
```

Neo4j

Google Drive

https://drive.google.com/drive/folders/1ibIpzgK_kUmXcySI-sWZsf_t_4vnX56L?usp=sharing

Import to 'Neo4j Desktop' or 'Neo4j Browser'





Members

Mohamed Arfan Mohamed Nayas
62011151

Prutikorn Chirattitikarn
62011224

Puhnyanuj Smizdhanond
62011225

Thaninrath Thiraphotiwat
62011276