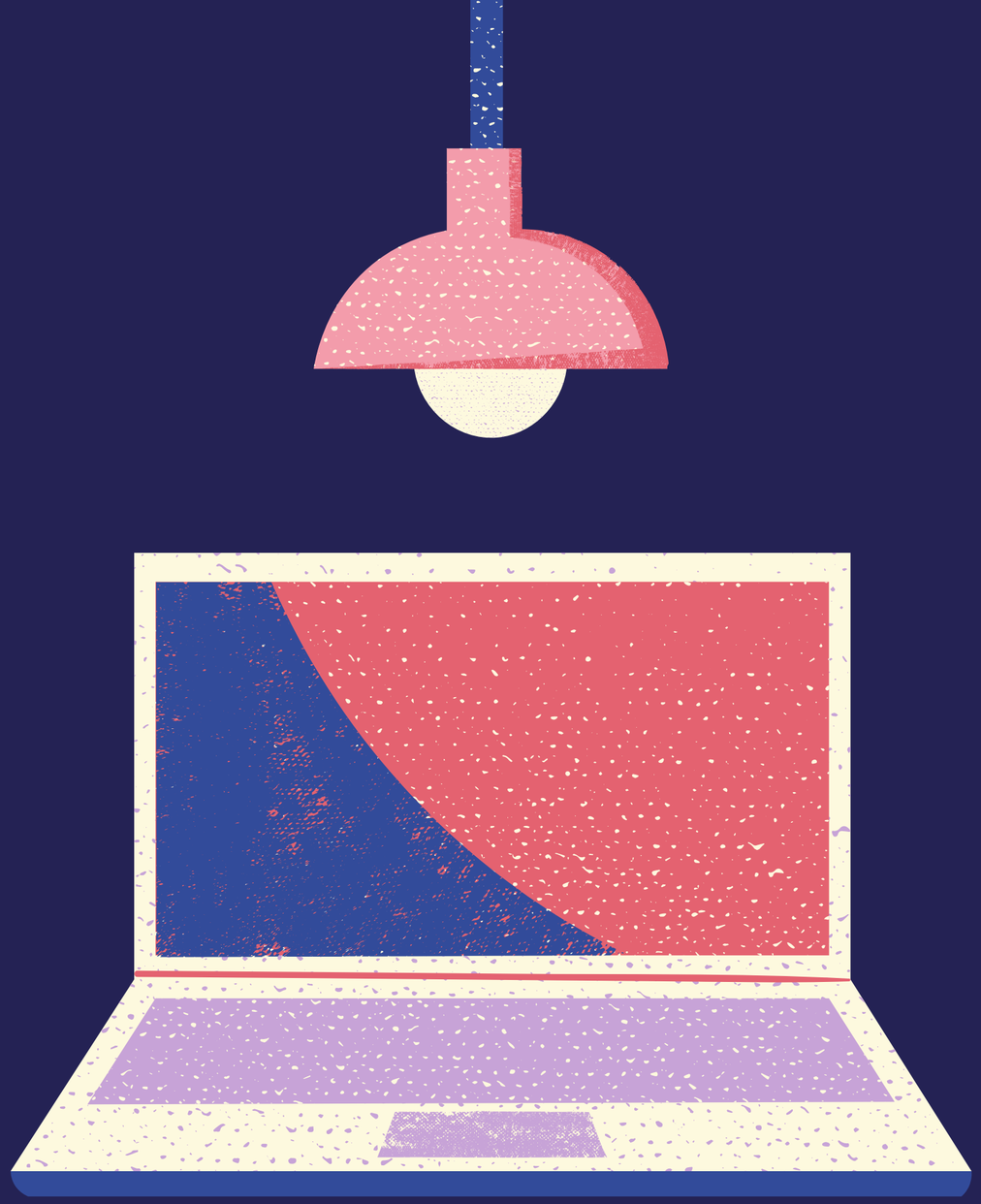


Big Data

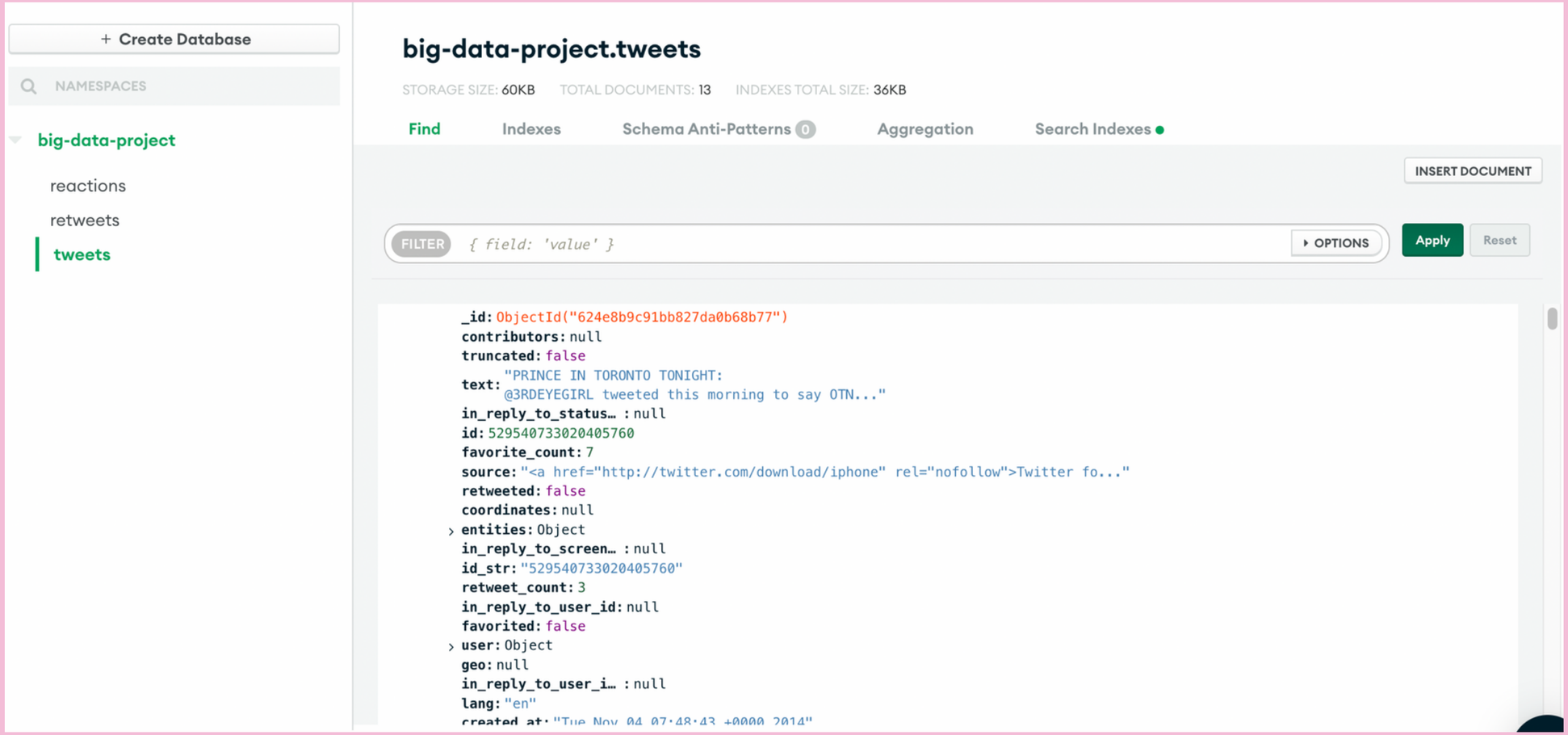
Rumour Spreading Analysis on Twittwer

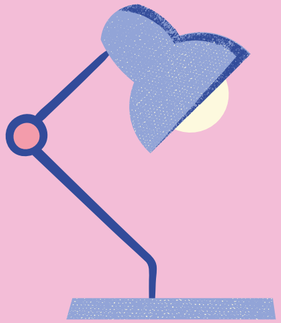
Task 1

Prince Toronto

Task 1: Tweets

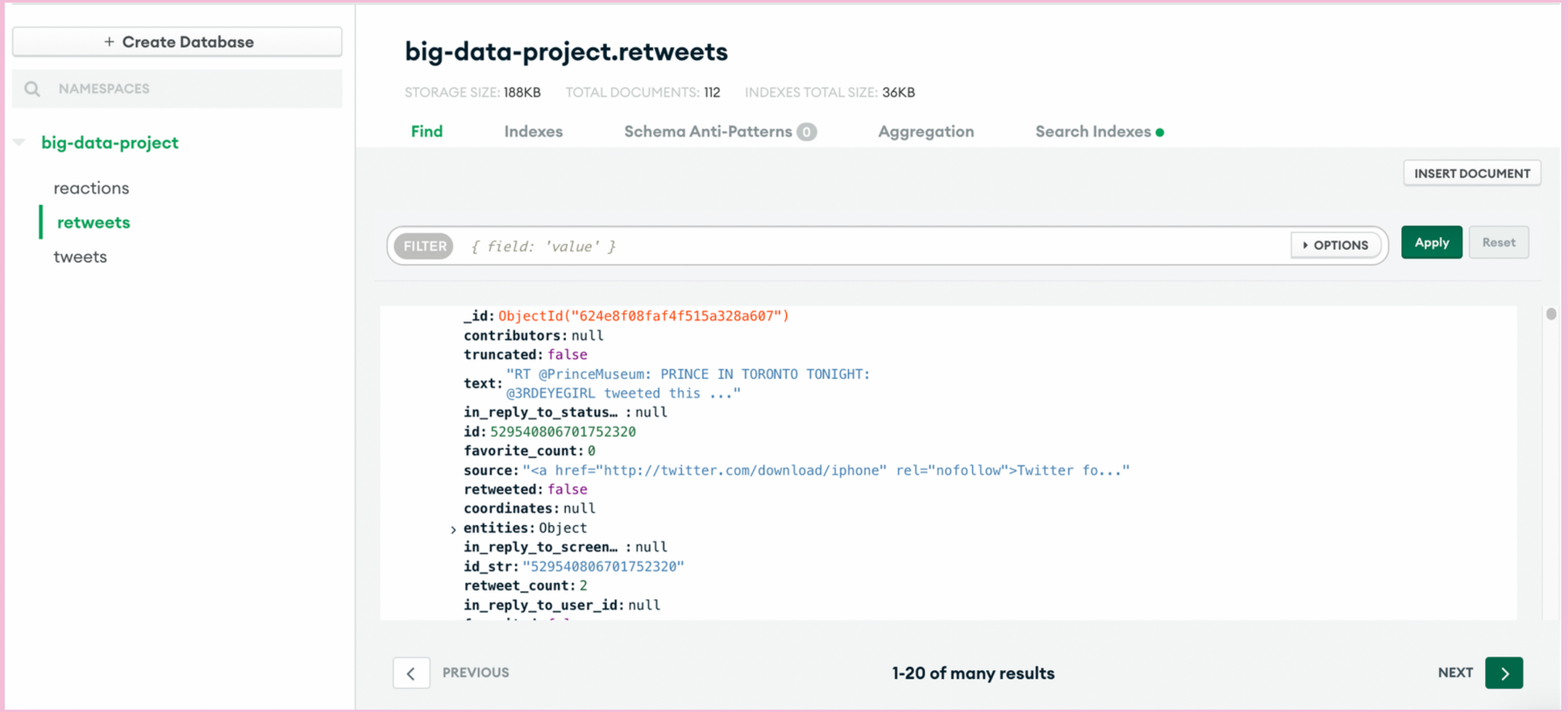
mongoimport --uri mongodb+srv://benz1224:raikmitl01@toronto-prince-rumour.v0cqw.mongodb.net/big-data-project --collection tweets --type json --file /Users/benzpreuengineer/Desktop/pheme-rumour-scheme-dataset/threads/en/prince-toronto/529540733020405760/source-tweets/529540733020405760.json





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



Task 2: CRUD Commands to UPDATE

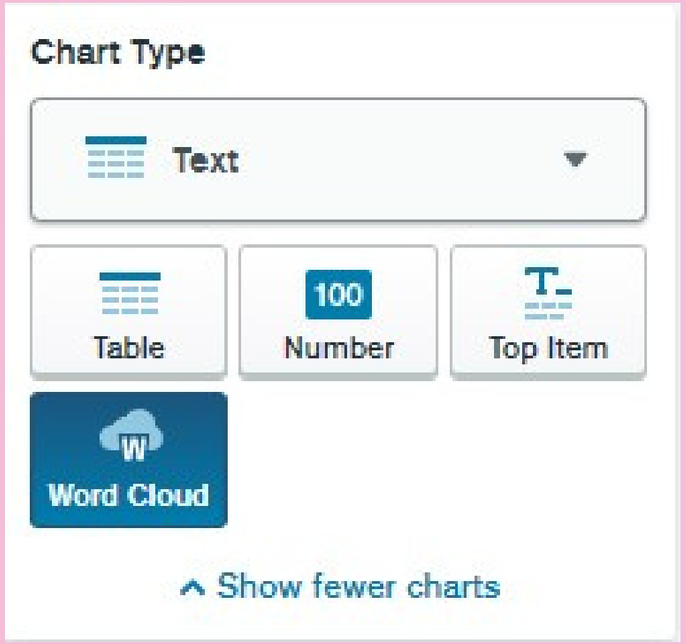
```
mongo mongodb+srv://toronto-prince-rumour.v0cqw.mongodb.net/big-data-project --username benz1224


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}})
```

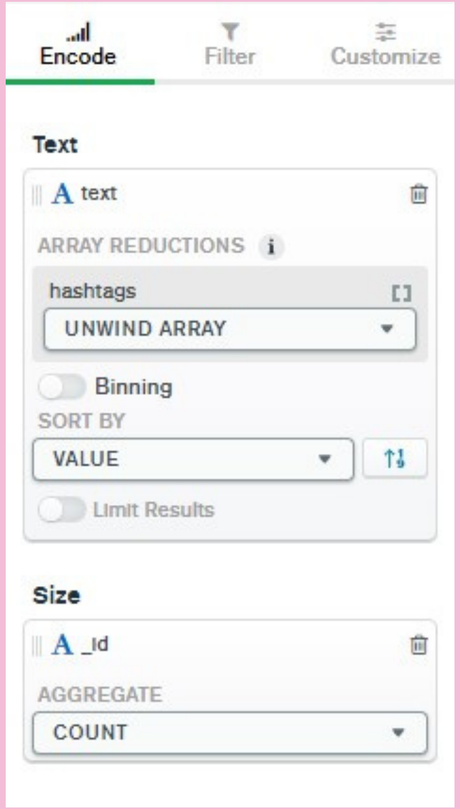

```
        "is_translator": false
    },
    "geo": null,
    "in_reply_to_user_id_str": null,
    "possibly_sensitive": false,
    "lang": "en",
    "created_at": "Tue Nov 04 18:03:11 +0000 2014",
    "filter_level": "low",
    "in_reply_to_status_id_str": null,
    "place": null,
    "category": "Prince 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.

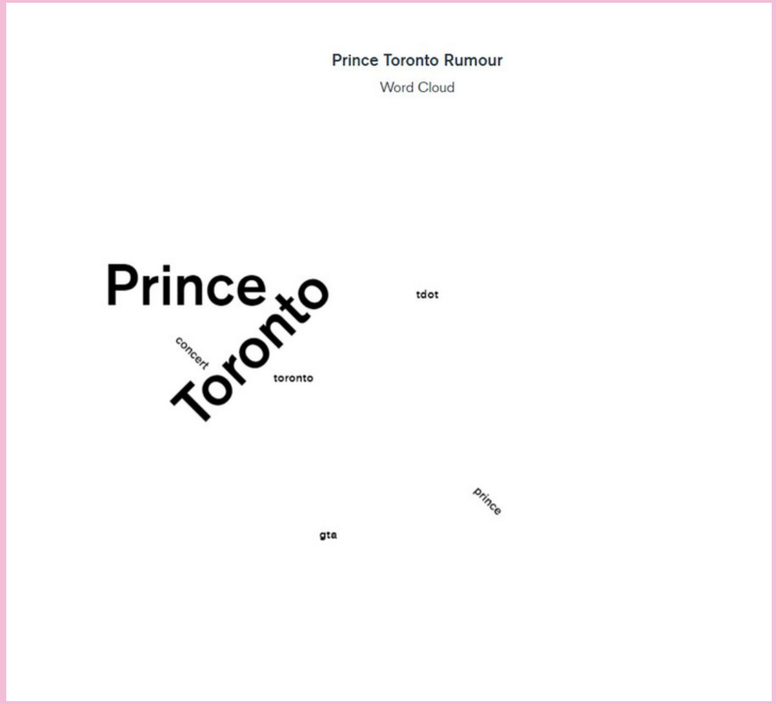
The rumour and non rumour data are filter by query: {is_rumour: "rumour"} and {is_rumour: "non-rumour"}



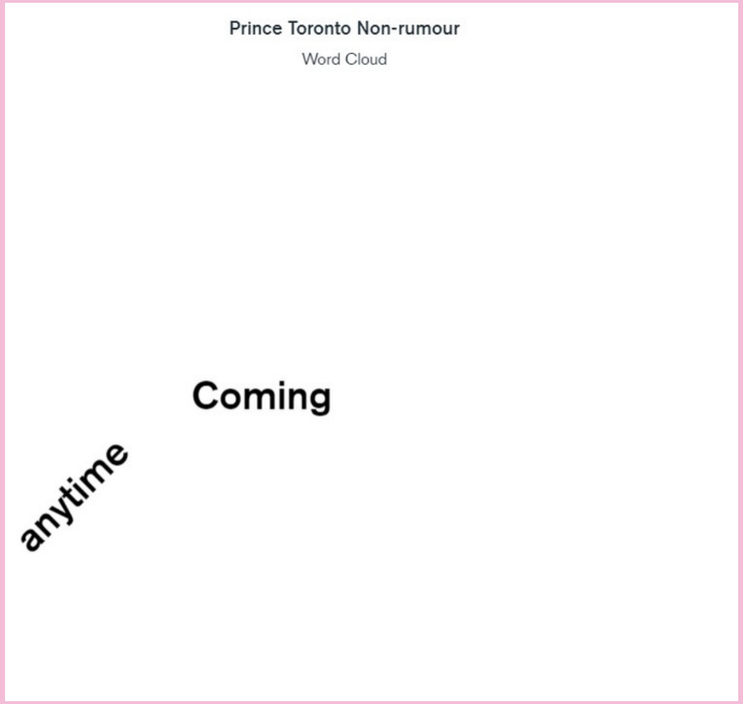


Task 3: Word Cloud

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
    }
  }
]
```

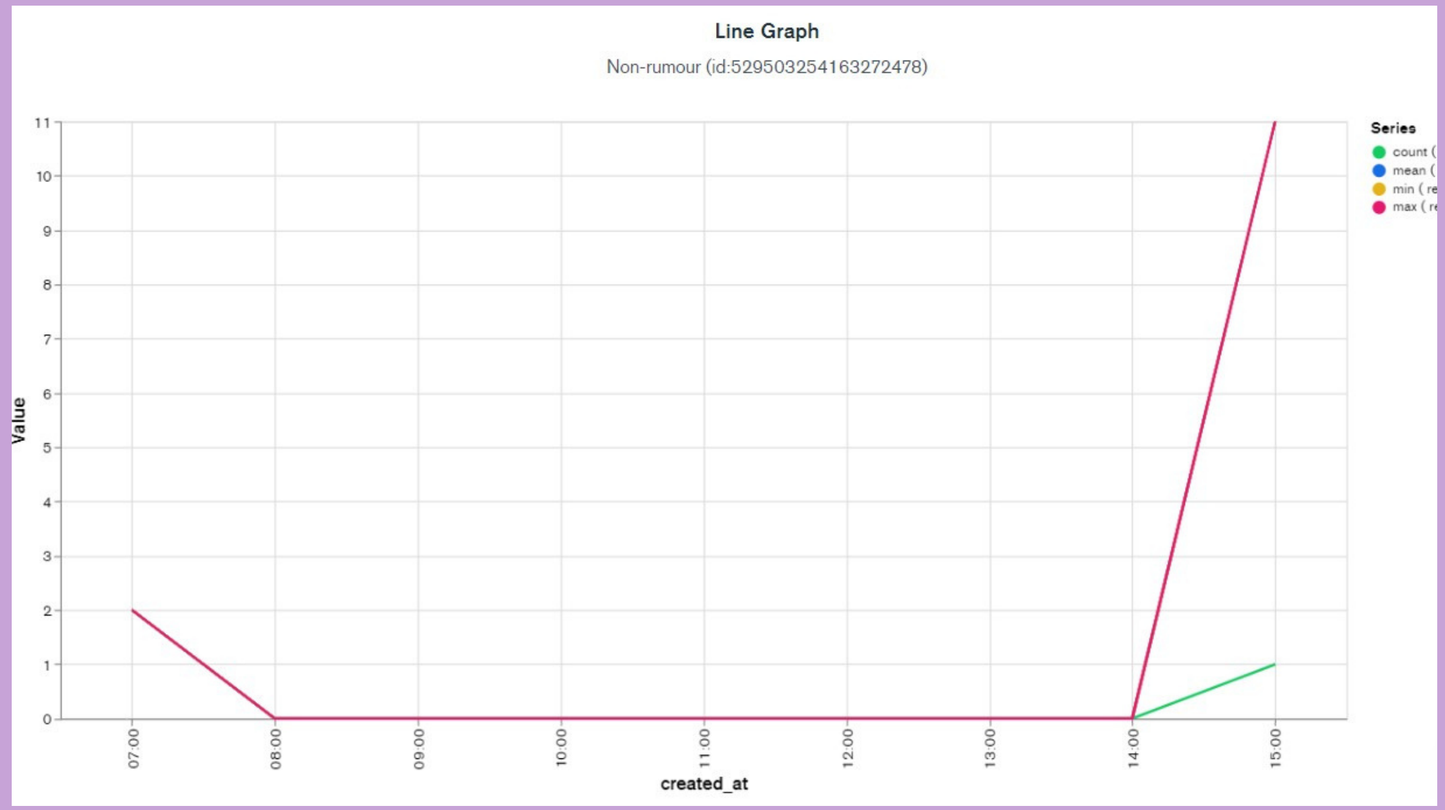
Query

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

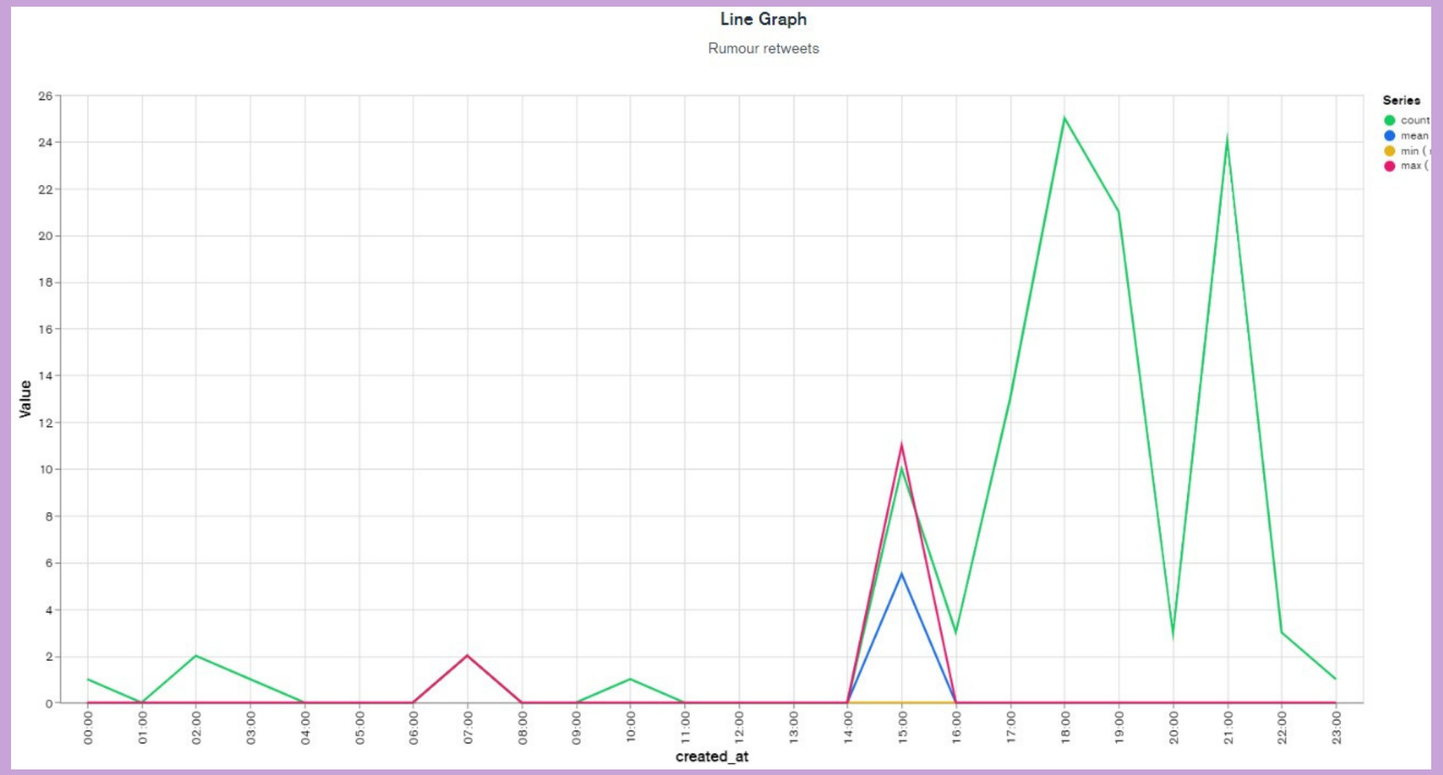
- X Coordinate: Count of retweet id
- Y Coordinate: Created Time (Hours)



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)



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

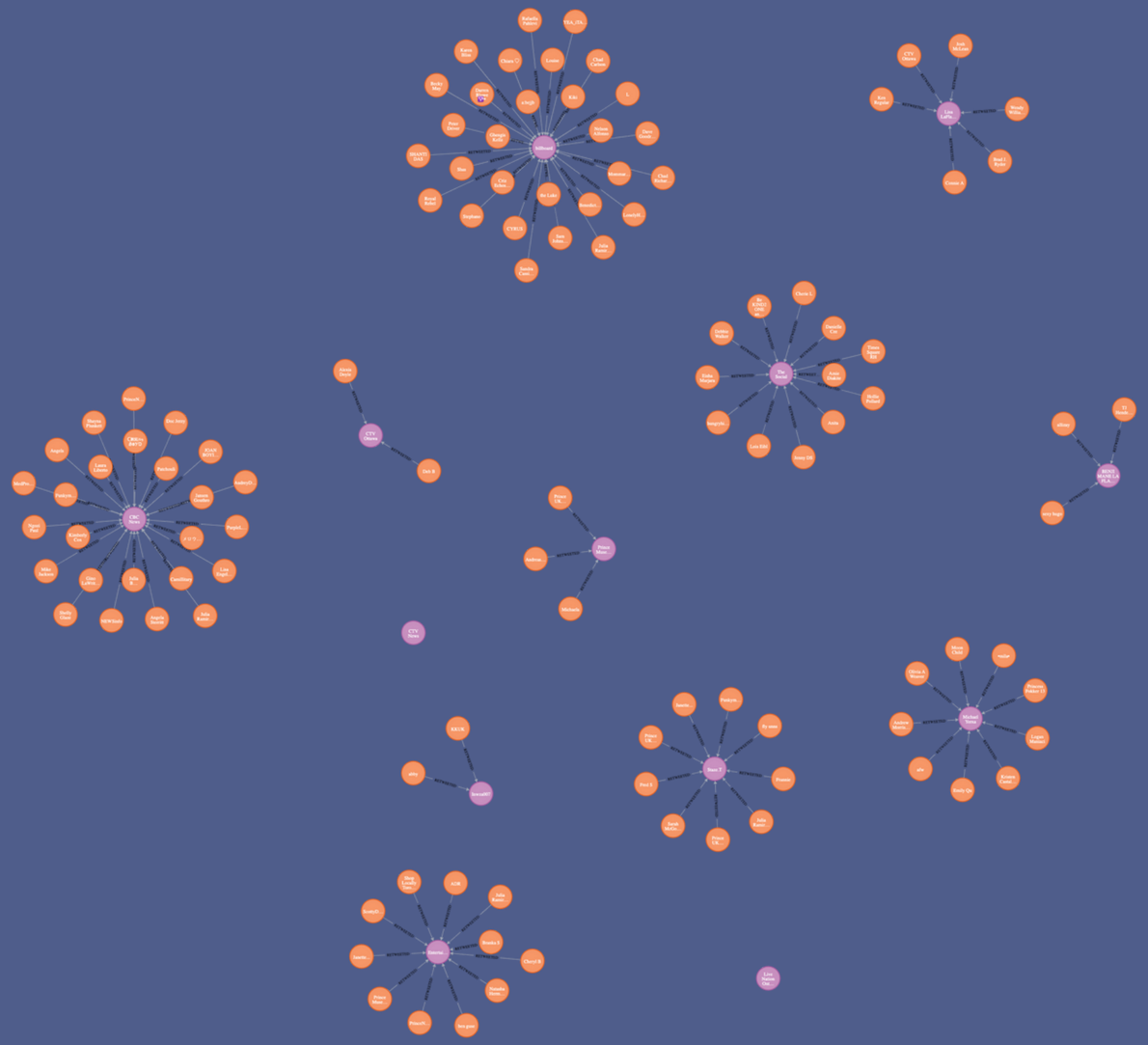
```
CALL apoc.json.load($tweets)
YIELD value AS v
UNWIND v.user AS u
CREATE (n:User:Tweet {id:u.id_str, name:u.name, post:v.text, post_id:v.id_str})
```

```
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

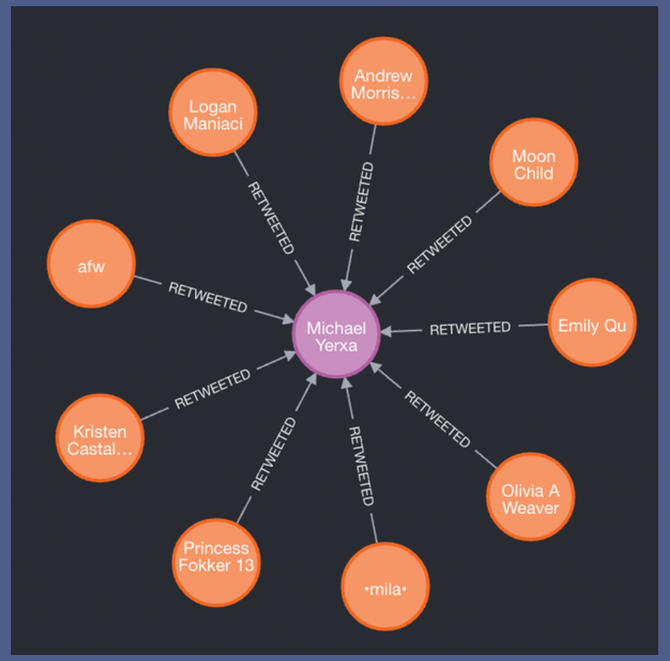
```
MATCH (a:User), (b:Retweeted)
WHERE a.id = b.retweeted_id
CREATE (b)-[r:RETWEETED]->(a)
```

Task 6: Tweets & Retweets



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

MATCH (n) RETURN n

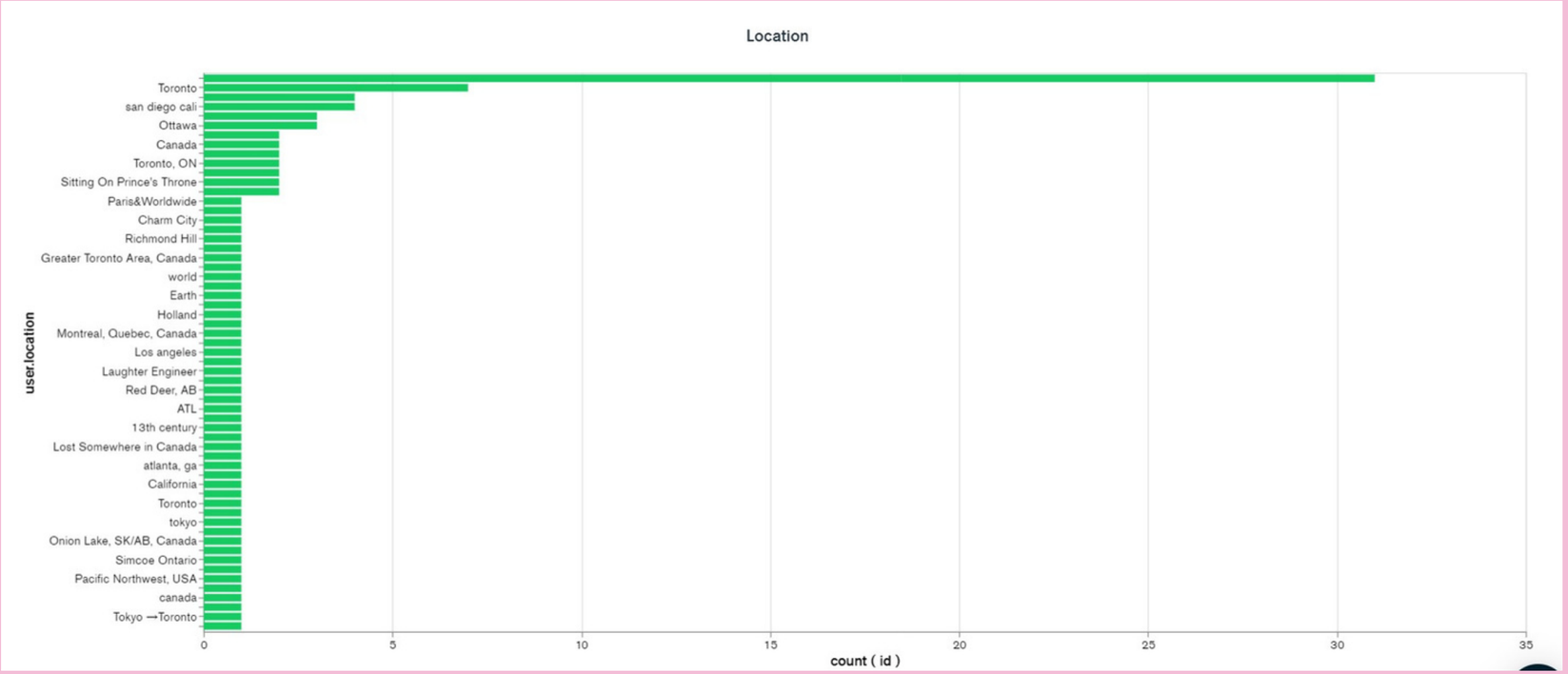


Task 7: Users' locations

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

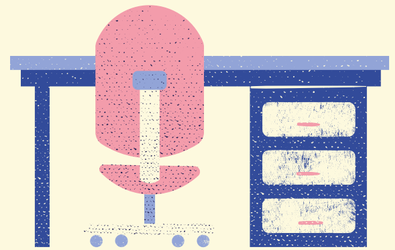


Visualization of Locations of each tweet

Task 8: Converting Script (1)

```python
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
```

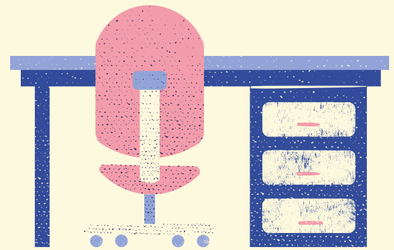
Converting script to convert a **structure** file into the corresponding tweet and its hierarchical replies



Task 8: Converting Script (2)

```python
# 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)
```

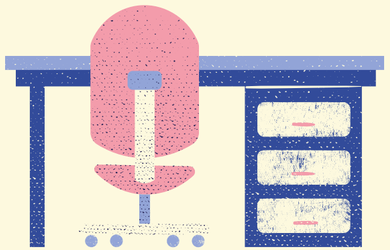
Converting script to convert a **structure** file into the corresponding tweet and its hierarchical replies



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
```

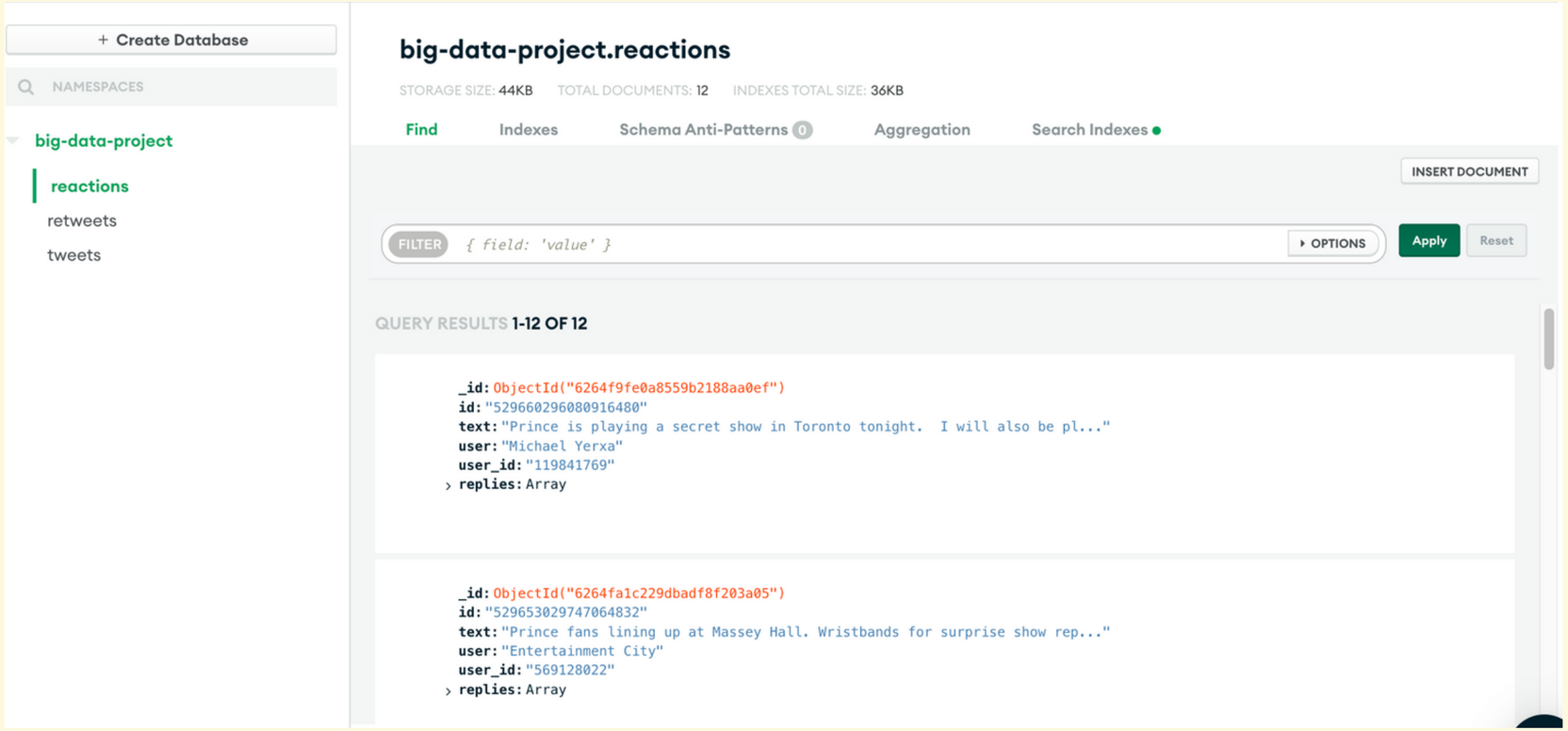
Converting script to convert a **structure** file into the corresponding tweet and its hierarchical replies

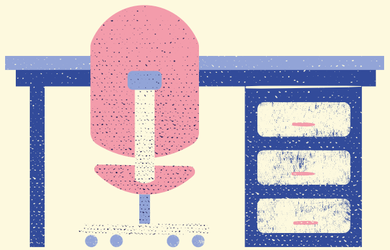


Task 8: Converting Script (4)

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





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

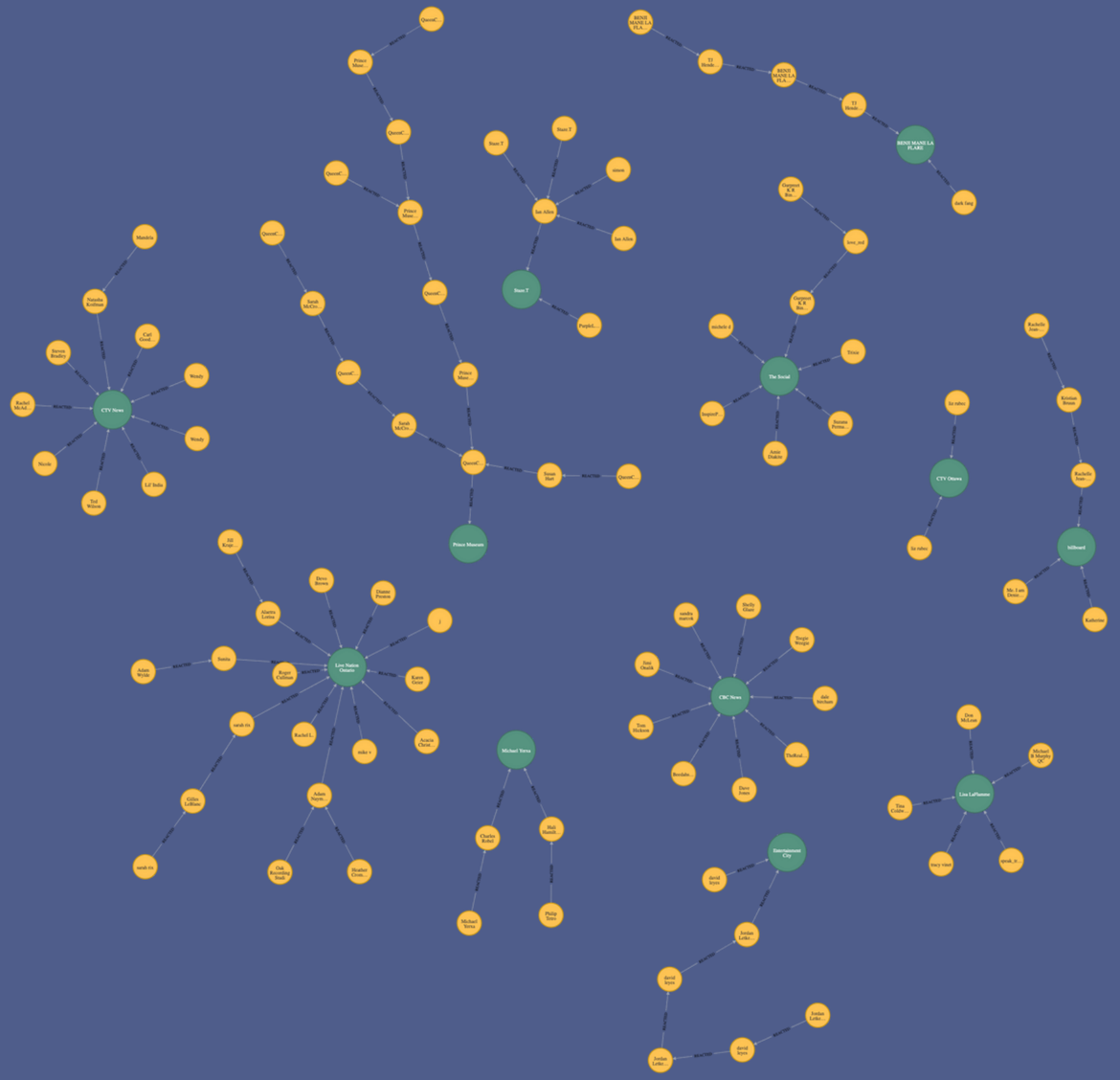
Creating nodes

```
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/1iblpzgK_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