Twitter Users and Tweets Information Management System

A Final Project Report Submitted for Database 694

by
Team 5
Jiawen Xu
Qiren Wang
Yanping Wang
Haoyuan Zhong

New Brunswick, New Jersey
MAY, 2022

1. Introduction

Throughout the semester, there are plenty of data management concepts and skills that learned from class, such as relational databases, SQL language, file structure, indexing, JSON file, TinyDB, MapReduce, and Hadoop. In this project, to practice the concepts and skills, by firstly inputting the target tweet dataset, *coronal-out-3,* the summary reports of users, tweets, and other Twitter information can be easily pulled out from each tweet in JSON data format. Moreover, by utilizing datastore concepts, it is not complicated to store the user information in a relational store and the tweets data in a non-relational store. Lastly, after running and using a well-designed search application for each store, a test will be performed in the set of representative queries and each operation's running times will be recorded.

2. Data Description and Processing

The target tweet dataset "coronal-out-3" was provided by the professor for this project. The dataset contains a total of 101916 tweets and 61112 retweets. The initial dataset given was in text format. Below Figure 2a is a snippet of the text data:


```
{"created_at":"Sun Apr 12 18:27:25 +0000 2020","id":1249403768023678982,"id_str":"1249403768023678982","text":"RT @lale_karanfil: In Turkey, there are 300 thousand prisoners and 150 thousand
prison employees in prisons. \nPrisons are the most risky pl\u2026","source":"\u003ca href=\"https:\/\/mobile.twitter.com\" rel=\"nofollow\"\u003eTwitter Web App\u003c\/
a\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":
{"id":1225145123920588805,"id_str":"1225145123920588805","name":"efe09","screen_name":"efe0927183508","location":null,"url":null,"description":"Allah'\u013in en de\u011ferli eseri
insand\u0131r.\nCan\u0131 yanan sabretsin.\nCan yakan, can\u0131n\u0131n yanaca\u011f\u0131 g\u00fcn\u00fc beklesin..\n677 khk\nRT d\u00fc\u015f\u00fcncem oldu\u011fu anlam\u0131na gelmez
bilgi
ama\u00e7l\u0131d\u0131r","translator_type":"none","protected":false,"verified":false,"followers_count":653,"friends_count":983,"listed_count":0,"favourites_count":1255,"statuses_count":4177,"
created_at":"Wed Feb 05 19:52:38 +0000
2020","utc_offset":null,"time_zone":null,"geo_enabled":false,"lang":null,"contributors_enabled":false,"is_translator":false,"profile_background_color":"F5F8FA","profile_background_image_url":"
","profile_background_image_url_https":"","profile_background_tile":false,"profile_link_color":"1DA1F2","profile_sidebar_border_color":"C0DEED","profile_sidebar_fill_color":"DDEEF6","profile_t
ext_color":"333333","profile_use_background_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/1243137514665361409\/95Mv0oRS_normal.jpg","profile_image_url_https":"https:
\/\/pbs.twimg.com\/profile_images\/1243137514665361409\/95Mv0oRS_normal.jpg","profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/1225145123920588805\/
1585222614","default_profile":true,"default_profile_image":false,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":nul
l,"retweeted_status":{"created_at":"Sun Apr 12 18:02:41 +0000 2020","id":1249397541596286979,"id_str":"1249397541596286979","text":"In Turkey, there are 300 thousand prisoners and 150 thousand
prison employees in prisons. \nPrisons are the most ris\u2026 https:\/\/t.co\/GogsgfUxkE","source":"\u003ca href=\"http:\/\/twitter.com\/download\/android\" rel=\"nofollow\"\u003eTwitter for
Android\u003c\/
a\u003e","truncated":true,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":
{"id":1087735689091928064,"id_str":"1087735689091928064","name":"Karanfil
Lale","screen_name":"lale_karanfil","location":null,"url":null,"description":null,"translator_type":"none","protected":false,"verified":false,"followers_count":897,"friends_count":1120,"listed
_count":1,"favourites_count":2776,"statuses_count":7986,"created_at":"Tue Jan 22 15:36:12 +0000
2019","utc_offset":null,"time_zone":null,"geo_enabled":false,"lang":null,"contributors_enabled":false,"is_translator":false,"profile_background_color":"F5F8FA","profile_background_image_url":"
","profile_background_image_url_https":"","profile_background_tile":false,"profile_link_color":"1DA1F2","profile_sidebar_border_color":"C0DEED","profile_sidebar_fill_color":"DDEEF6","profile_t
ext_color":"333333","profile_use_background_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/1248313542387757057\/see_F0uX_normal.jpg","profile_image_url_https":"https:
\/\/pbs.twimg.com\/profile_images\/1248313542387757057\/see_F0uX_normal.jpg","profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/1087735689091928064\/
1571123765","default_profile":true,"default_profile_image":false,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":nul
l,"is_quote_status":false,"extended_tweet":{"full_text":"In Turkey, there are 300 thousand prisoners and 150 thousand prison employees in prisons. \nPrisons are the most risky places in case
of corona virus.\nThe lives of thousands of people are at risk.\n\n#COVID19InTurkeysPrisons\n\n@SusannaCeccardi","display_text_range":[0,239],"entities":{"hashtags":
[{"text":"COVID19InTurkeysPrisons","indices":[197,221]}],"urls":[],"user_mentions":[{"screen_name":"SusannaCeccardi","name":"Susanna Ceccardi","id":406110645,"id_str":"406110645","indices":
[223,239]}],"symbols":[]}},"quote_count":0,"reply_count":1,"retweet_count":14,"favorite_count":6,"entities":{"hashtags":[],"urls":[{"url":"https:\/\/t.co\/GogsgfUxkE","expanded_url":"https:\/
\/twitter.com\/i\/web\/status\/1249397541596286979","display_url":"twitter.com\/i\/web\/status\/1\u2026","indices":[117,140]}],"user_mentions":[],"symbols":
[]},"favorited":false,"retweeted":false,"filter_level":"low","lang":"en"},"is_quote_status":false,"quote_count":0,"reply_count":0,"retweet_count":0,"favorite_count":0,"entities":{"hashtags":
[],"urls":[],"user_mentions":[{"screen_name":"lale_karanfil","name":"Karanfil Lale","id":1087735689091928064,"id_str":"1087735689091928064","indices":[3,17]}],"symbols":
[]},"favorited":false,"retweeted":false,"filter_level":"low","lang":"en","timestamp_ms":"1586716045753"}
```


Figure 2a. *rawtweet_info*

One of the most important tasks was to pre-process the dataset. Data preprocessing was done in two parts: One part was to load the tweets into the relational database MySql and the Non-relational database MongoDB using Python code. Another way to design store actions on a tweet.

For the steps to store Tweet data, as mentioned in the introduction, There will be separated two ways to store tweet data: For user information, it is saved in a relational datastore MySQL. PyMySQL was used as an interface to connect a MySQL database server. For tweets information, it is saved in a non-relational datastore MongoDB. Pymongo was used as an interface to connect a MongoDB database server.

After loading the data to each database. It can be easily found that the structure of the data contains different types of information for every single tweet. For example, if try to pull out the summary report of tweet information, all the information, such as tweets created time, tweets, reply status, and user's id will be shown on the summary report:

```
# The Tweet Info
tweet
```

```
{'created_at': 'Sun Apr 12 18:47:53 +0000 2020',
 'id': 1249408918196871168,
 'id_str': '1249408918196871168',
 'text': 'RT @liputan6dotcom: Lagi, 2 Dokter Senior Meninggal karena Corona Covid-19 https://t.co/eDbMsxFA84',
 'source': '<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>',
 'truncated': False,
 'in_reply_to_status_id': None,
 'in_reply_to_status_id_str': None,
 'in_reply_to_user_id': None,
 'in_reply_to_user_id_str': None,
 'in_reply_to_screen_name': None,
 'user': {'id': 2283278160,
  'id_str': '2283278160',
```

Figure 2b. *tweet_info*

Pulling out the summary report of user information, the report contains user_id, screen_name, location, and all other profile information. The summary report of retweets will include information, such as retweet time, the user who retweets it, and if any user is mentioned for this retweet.

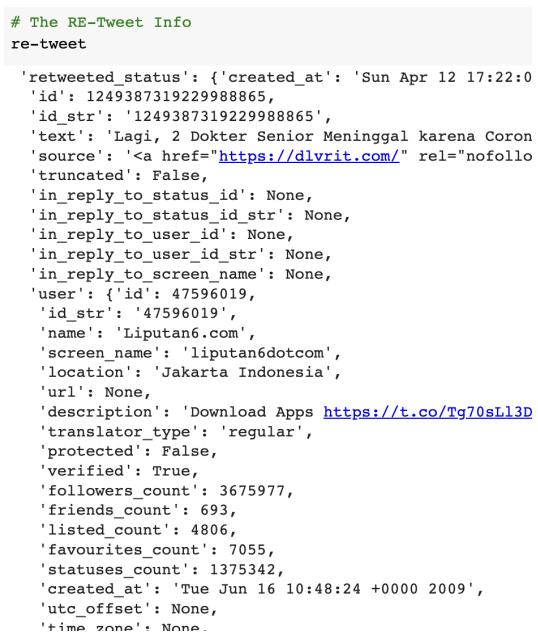

Figure 2c. *Re-tweet_info*

Figure 2d. *tweet-user_info*

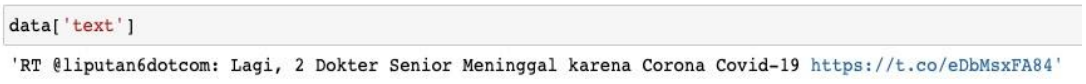

Figure 2e. *tweet-text_info*

3. Data Storage

a. MySQL

In designing the MySQL storage, there were several important considerations. The version of MySQL Server was chosen 8.0.2. All requirements are listed below: Jupyter Notebook, Pymysql, MySQL Workbench. MySQL Server on a local macOS or PC connected with the port number 3306. Database login requires a username and password. During the test, it did not require a password to log in. Users just need to use *root* as a username to access the database. Since the data was stored in JSON Object in text file format, all JSON objects were traversed. In the MySQL storage part, there was an initial parameter (*data: type list*) to store records that have been already iterated. The data was stored in the database called *my_twitter_db*. The table called user_info was created in my_twitter_db to store Twitter user information. user_info stored multiple fields: *user_id, name, screen_name, location, url, protected, followers_count, listed_count, favourites_count, created_at, statuses_count, following, follow_request_sent*. Here is Figure 2a.1 which shows the type of each field. User_id was chosen for the primary key.

Field	**Type**
user_id	VARCHAR(20), PRIMARY KEY
name	VARCHAR(50)
screen_name	TEXT
location	TEXT
url	TEXT
protected	VARCHAR(10), DEFAULT NULL
followers_count	INT(11) DEFAULT NULL
friends_count	INT(11) DEFAULT NULL
listed_count	INT(11) DEFAULT NULL
favourites_count	INT(11) DEFAULT NULL
created_at	TEXT
statuses_count	INT(11) DEFAULT NULL
following	TEXT
following_request_sent	TEXT

Figure 2a.1 Table *user_info*

Since the text file was transferred to the data list, each record could be manipulated as a data frame. For example, the value of user_id was assigned at the level of the id of the user in a piece of a JSON Object record. The other fields in which values need to be stored were assigned by using the same method. Figure 2a.2 shows as below:

```
user_id = item['user']['id']
name = item['user']['name']
screen_name = item['user']['screen_name']
location = item['user']['location']
url = item['user']['url']
protected = item['user']['protected']
followers_count = item['user']['followers_count']
friends_count = item['user']['friends_count']
listed_count = item['user']['listed_count']
created_at = item['user']['created_at']
favourites_count = item['user']['favourites_count']
statuses_count = item['user']['statuses_count']
following = item['user']['following']
follow_request_sent = item['user']['follow_request_sent']
```

Figure 2a.2 Method to get value in data and assign to new parameter

During inserting each record to table *user_info*, unique user_id was checked by function. A method designed to count the user_id if it exists or not. If the user_id existed before, it would be counted as a number "1", if not the count number would indicate "0". Therefore, the insert function processes all data and inserts each tweet's user's information one by one while checking the duplication of user information. Figure 2a.3 shows how it works as below:

```
indicator = "select * from user_info where user_id = %s;" % (user_id)
count = cursor.execute(indicator)
val = (user_id, name, screen_name, location, url, protected, followers_count, friends_count, listed_count,
  created_at, favourites_count, statuses_count, following, follow_request_sent)
if count == 0 :
    #print('####:',name)

    sql2 = """INSERT INTO user_info(user_id, name, screen_name, location, url,
          protected, followers_count,friends_count,listed_count,created_at,
          favourites_count,statuses_count,following,follow_request_sent)
          VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
          """

    cursor.execute(sql2, val)
    conn.commit()
```

Figure 2a.3 Insert data to table *user_info* and check duplication

After all operations are complete, the cursor should be closed. The final step is to disconnect the database.
In this project, MySQL official GUI Workbench was used to visualize the data. Also, it allows users to query data directly through this GUI. Figure 2a.4 shows below data storage in MySQL.

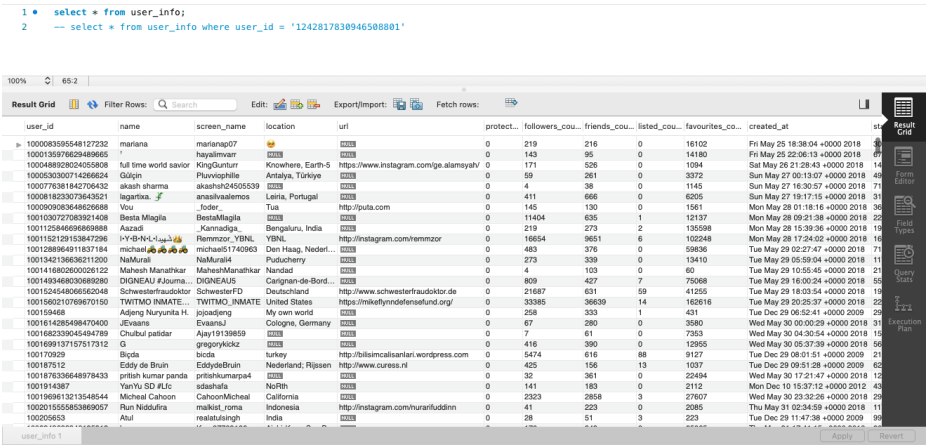

Figure 2a.4 Data Visualization through MySQL Workbench

b. MongoDB

For the non-relational database, MongoDB was used for saving tweet information. MongoDB is a document-based database that uses documents instead of tuples in tables to store data. To connect the MongoDB server to Python, MongoDB's official Python driver called PyMongo was used. PyMongo provides a rich set of tools that can be used to communicate with a MongoDB server. It provides functionality to query, retrieve results, write and delete data, and run

database commands. In designing the Mongodb storage, there were several important considerations. The version of MongoDB Server was chosen 1.31.2. All requirements are listed below: Jupyter Notebook, pymongo. Connected MongoDB Server on local MacOS or PC with the port number 27017.
In the MongoDB storage part, a database named twitter1 was created and all the total of 101916 tweets were loaded into the Database using Python code. The collection named tweets were created on twitter1 to store tweet information. Figure3b.1shows the process.

```
import sys
import json
from pymongo import MongoClient
from pymongo.errors import ConnectionFailure

def main():
    "==== Connect to MongoDB ===="
    try:
        client = MongoClient(host="localhost", port=27017)
        twitter1 = client.twitter1
        print("Connected to MongoDB")
    except(ConnectionFailure, e):
        sys.stderr.write("Could not connect to MongoDB %s" % e)
        sys.exit(1)

    with open('corona-out-3', 'r') as f:
        print("Inserting tweets in MongoDB")
        for line in f:
            if(line=='\n'):
                continue
            tweet = json.loads(line) # load it as Python dict/document for db
            twitter1.tweets.insert_one(tweet)
        print("Completed tweets insert in MongoDB")
        client.close # close db connection
    f.close() # close file handle

if __name__ == "__main__":
    main()
```

Figure 3b.1 Load twitter.json in MongoDB and fetch data

To efficiently store the data for fast access and search specific information. The indexes were created for all 4 collections. Based on the collection " tweets ", MongoDB's MapReduce function was used to create 3 other collections. The collection " hashtags " was created as Figure 3b.2 to store hashtag information which includes *hashtag label* and *value*. The collection " user_mentions " was created as Figure 3b.3 to store user_mentions information which includes *usermention label* and *value*. The collection " retweets " was created as Figure 3b.4 to store retweet information which includes *user_id*, *retweet_count*, and *text*.

```
%%bash
mongo twitter1

db.tweets.mapReduce(
function() {
    if(this.entities) {
        this.entities.hashtags.forEach( function(ele) {
            emit(ele.text, 1)
        })
    }
},
function(key, values) { return Array.sum(values) },
{ query: { },  out: "hashtags"}
)
```

Figure 3b.2 create hashtags collection

```
%%bash
mongo twitter1
db.tweets.mapReduce(
function() {
    if(this.entities) {
        this.entities.user_mentions.forEach( function(ele) {
            emit(ele.screen_name, 1)
        })
    }
},
function(key, values) { return Array.sum(values) },
{ query: { },  out: "usermentions" }
)
```

Figure3b.3 create user_mention collection

```
%%bash
mongo twitter1

db.tweets.mapReduce(
function() {
    if(this.retweeted) {
        emit(this.user.id, 1)
    }
},
function(key, values) { return Array.sum(values) },
{ query: { },  out: "retweets" }
)
```

Figure 3b.4 create retweets collection

After creating all of the collections, the MongoDB Compass can easily visualize the data. Also, it allows users to query data directly through this GUI. Figure 3b.5-8 shows below all tweet information storage in MongoDB.

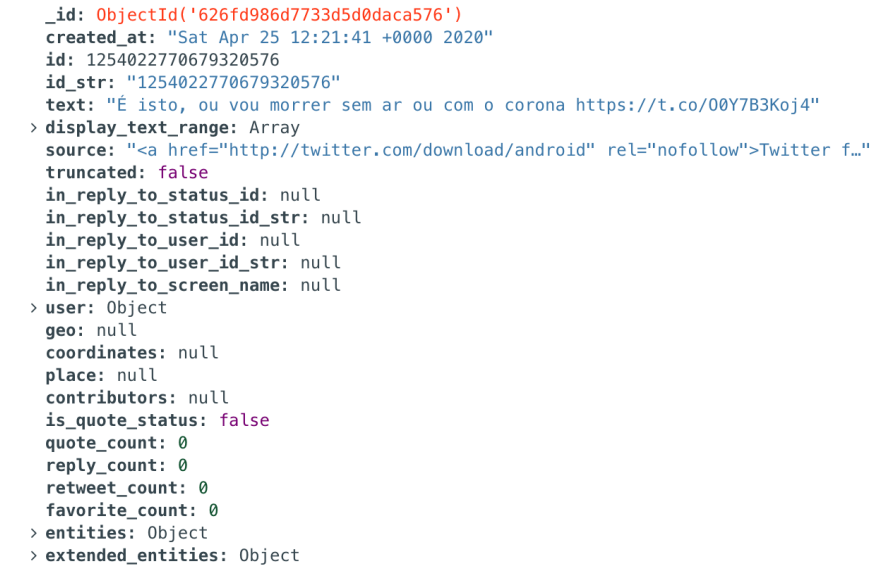

Figure 3b.5 Tweet collection data visualization through MongoDB compass

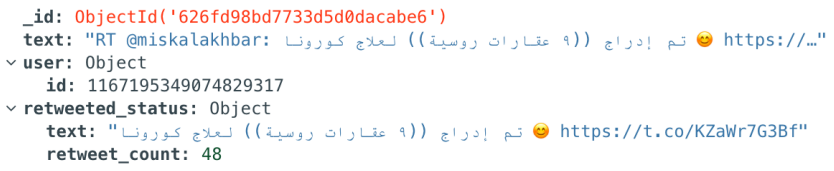

Figure 3b.6 Retweet collection data visualization through MongoDB compass

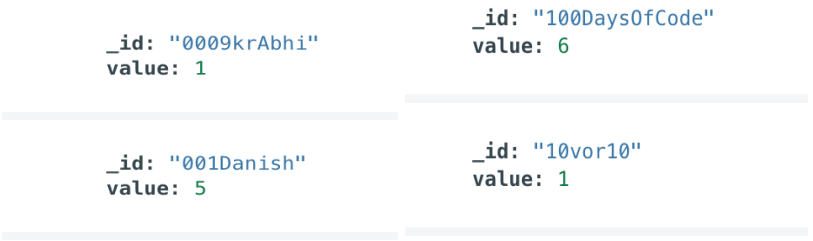

Figure 3b.7 Hashtags collection data visualization through MongoDB compass
Figure 3b.8 Usermention collection data visualization through MongoDB compass

4. Search Application

a. Design

Search Application is implemented in Python and used Jupternotebook as the User Interface, then ask the users to input their option of search, next will return the result by the combination of MySQL database and MongoDB database. In addition, Redis is used for caching popular data which is corresponding to 'rank' from the search options. And, the user can enter

b. Search Options:

i. MySQL

1. user id & user screen name

MySQL database will be used for search input is 'user id' or 'user name'.
Step 1 is to connect the MySQL server and choose the specific database. Step 2 is to generate a SQL query to search the content by user id or user screen name from the table, Figure 4b.1, which stores users' information. Step 3 is to return the detail of this user's information.

user_id	name	screen_name	location	url
100008359554812…	mariana	marianap07	🥺	NULL
100013597662948…	⁷	hayalimvarr	NULL	NULL
100048892802405…	full time world savior	KingGunturr	Knowhere, Earth-5	https://v
100053030071426…	Gülçin	Pluvviophille	Antalya, Türkiye	NULL
100077638184270…	akash sharma	akashsh24505539	NULL	NULL
100081823307364…	lagartixa. 🦎	anasilvaalemos	Leiria, Portugal	NULL
100090908364862…	Vou	_foder_	Tua	http://pu
100103072708392…	Besta Mlagila	BestaMlagila	NULL	NULL

Figure 4b.1 The partial information of the 'user_info' table from the MySQL database

In addition, it will also return one tweet of this user. Step 1 is to connect the MongoDB server and then choose the collection; Step 2 is to query the 'text' field from the documents by user id or user name which we get from the input. For example, Figure 4b.2, if the input of the search option is 'user screen name', the application will ask the user to enter the user screen name, then return the information and also one tweet content of this user.

```
Options:
- user id
- user screen name
- tweets
- hashtag
- Top Rank for hashtags
- Quit
Which option you want to choose? user screen name

Search by user screen name...

Please enter the user screen name:marianap07
```

```
RESULT:


[User Information]
user_id :  1000083595548127232
name :  mariana
screen_name :  marianap07
location :  🥺
url :  None
protected :  0
followers_count :  219
friends_count :  216
listed_count :  0
favourites_count :  16102
created_at :  Fri May 25 18:38:04 +0000 2018
statuses_count :  30962
following :  None
follow_request_sent :  None
[Tweet]
tweet: RT @xx_duuda: Para: corona
   • " logo agora que tudo tava me a correr tão bem, apareceste para me fazer parar no time "
```

Figure 4b.2 The example of searching by user screen name

ii. MongoDB

For the search application on tweets, three types of information can be searched from MongoDB. They are tweet content, hashtag, and rank on two top-level metrics. No matter which information are need to be searched for, connecting to the MongoDB server should be always the first step. Then, the option of MongoDB can be chosen either tweet content, hashtag, or rank in the search application.

1. tweet content

For tweet content, by inputting a tweets' keyword that the user wants to look for regarding the tweet, the search application will return five or fewer relating tweets that contain the text, the retweet number, and the user id for further user search. The result is ordered by the retweet count of the tweets in descending order. If the key error happens, it will return no result. Here is an example of the search application on tweets. While searching for tweets regarding the keywords, Covid, the search application will return five related tweets about Germany with its corresponding retweet count. The output shows in Figure 4b.3:

```
Options:
- user id
- user screen name
- tweet
- hashtag
- rank
Which option you want to choose? tweet
Searching by tweets...
please enter content:Covid

Tweet:  Gua resah dengan kondisi saat ini, gua pengen speak up sebagai pasien suspect Covid-19. Gua akan cerita tenta
ng pen… https://t.co/qvvxEvYpZ6
retweet count:  69562
user id:  1197168671212240896

Tweet:  Dengan dampak dari virus corona (Covid-19) dan dipenuhi dengan ketakutan untuk masa depan yang tidak pasti di
sertai… https://t.co/E3lKy8GkSz
retweet count:  26796
user id:  123827118

Tweet:  Instead of staying idle due to Corona or Covid #Lockdown as an Architect I decided to make ''5 STAR SUSTAINAB
LE HOT… https://t.co/lx2TaHCBMZ
retweet count:  7790
user id:  1020315571018174465

Tweet:  SITUS RESMI soal Covid-19:

1. Nasional https://t.co/oikDzQObtd
2. Jawa barat https://t.co/m8SRMoRPYP
3. DKI Jakart… https://t.co/TyVnafvpwX
retweet count:  7412
user id:  1183206164

Tweet:  CORONA JATO: PF faz operação que mira desvio de verbas da Covid-19 https://t.co/GgLqLTfJ9b
retweet count:  2265
user id:  325109785
Time:  0.1805858612060547
```

Figure 4b.3

Moreover, the search application will ask whether user would like to search for user information of one tweet above. If user type yes, one user id needs to be input and the user information will be pulled out via MySQL relational database. As we can see here in Figure 4b.4, MySQL datastore return user information of first Germany tweet:

```
Would you like to search for the user information of one tweet above?
- yes
- no
yes


Please paste the user id of the tweet above: 1197168671212240896

                                        [User Information]
                                        user_id :  1197168671212240896
                                        name :  Abu Hamzah Ar Rizal
                                        screen_name :  ArtiAbuAbuHamza
                                        location :  Bumi Alloh
                                        url :  None
                                        protected :  0
                                        followers_count :  290
                                        friends_count :  601
                                        listed_count :  0
                                        favourites_count :  606
                                        created_at :  Wed Nov 20 15:04:32 +0000 2019
                                        statuses_count :  2367
                                        following :  None
                                        follow_request_sent :  None
```

Figure 4b.4

2. hashtag

Similarly for hashtag, by inputting a hashtag that the user wants to look for regarding the tweet, the search application will return five or fewer relating tweets with the searched hashtag in the database, with the user id for further user search. The result is ordered by the created time of the tweets in descending order. If the searched hashtag doesn't exist in the database, it will return no result.

Here is the example of the search application on hashtag. When a user searches for tweets regarding the hashtag, Corona. The search application will return five relating tweets that contain the hashtag, corona. The output shows in Figure 4b.4:

```
Options:
- user id
- user screen name
- tweet
- hashtag
- rank
Which option you want to choose? hashtag
Searching by hashtag...
please enter hashtag:Corona

Tweets:  We have 1/4 of all deaths from #Corona - thanks to our 3rd world response by @realDonaldTrump  #ashamed
@Arpx… https://t.co/3sP4AIJNJc
user id:  3130596629

Tweets:  RT @LubanaManoj: #StudentsFightCorona
#LadengeAurJeetenge
NSUI members all over the country following the way shown by @Neerajkundan ji and…
user id:  1204829507808419840

Tweets:  Unmut über Isolation in Pflegeheimen wächst.
https://t.co/ypbAj0h7sh
#Coronavirus #Pflegeheime
via @tagesschau
user id:  1249698272203223045

Tweets:  RT @JoergRodenwaldt: Wie #Großspender in der #Corona-Diskussion mitmischen.
Auch bekannt unter: #RheinischerKlüngel
 https://t.co/xMR5ncWYY…
user id:  492151077

Tweets:  RT @Ms28200: Süleyman Özışık durumu çok iyi açıklamış.
Nedir sizin bu bitmek bilmeyen düşmanlığınız?

#Iftar
#cumartesi #Corona
#pidealmay…
user id:  1233694652924125185
Time:  0.4394400119781494
```

Figure 4b.5

Furthermore, the search application will ask whether user would like to search for user information of one tweet above. If user type yes, one user id needs to be input and the user information will be pulled out via MySQL relational database. As we can see here in Figure 4b.6, MySQL datastore return user information of first tweet with #Corona:

```
Would you like to search for user information of one tweet above?
- yes
- no
yes

Please paste the user id of the tweet above: 3130596629

[User Information]
user_id :  3130596629
name :  Arkansas Boy
screen_name :  ArkansasWatch
location :  #Ark #LittleRock
url :  None
protected :  0
followers_count :  276
friends_count :  867
listed_count :  16
favourites_count :  1128
created_at :  Tue Mar 31 08:40:52 +0000 2015
statuses_count :  4410
following :  None
follow_request_sent :  None
```

Figure 4b.6

3. rank

Lastly, If the option, rank, is chosen, it will return these two top-level metrics, which are the top 10 most frequent hashtags in the dataset and the top 10 most frequently mentioned users in the dataset. From the result, the #Corona is the most frequent hashtag in the dataset, with 4582 times used. And @brithume with the 1496 times mentioned becomes the most frequently mentioned username in the dataset.

```
Options:
- user id
- user screen name
- tweet
- hashtag
- rank
Which option you want to choose? rank
Searching by rank...
What type rank would you like to see:
- mention
- hashtag
Which option you want to choose? mention
{'_id': 'brithume', 'value': 1496.0}
{'_id': 'Quirinale', 'value': 1466.0}
{'_id': 'benwikler', 'value': 987.0}
{'_id': 'oxfara', 'value': 846.0}
{'_id': 'yalim_funda', 'value': 734.0}
{'_id': 'realDonaldTrump', 'value': 705.0}
{'_id': 'narendramodi', 'value': 636.0}
{'_id': 'aajtak', 'value': 625.0}
{'_id': 'CrazyinRussia', 'value': 587.0}
{'_id': 'IngrahamAngle', 'value': 559.0}
Time:  0.02261495590209961
```

```
Options:
- user id
- user screen name
- tweet
- hashtag
- rank
Which option you want to choose? rank
Searching by rank...
What type rank would you like to see:
- mention
- hashtag
Which option you want to choose? hashtag
{'_id': 'Corona', 'value': 4582.0}
{'_id': 'Mattarella', 'value': 1506.0}
{'_id': '25Aprile', 'value': 1472.0}
{'_id': 'corona', 'value': 1449.0}
{'_id': 'Covid_19', 'value': 973.0}
{'_id': 'AltaredellaPatria', 'value': 805.0}
{'_id': 'PideAlmayaDiyeÇıkıp', 'value': 776.0}
{'_id': 'COVID19', 'value': 764.0}
{'_id': 'Liberazione', 'value': 696.0}
{'_id': 'coronavirus', 'value': 629.0}
Time:  0.010176897048950195
```

Figure 4b.4

c. Cache

Redis is used for caching popular data, such as the result of the 'rank' option. According to Figure 4c.1, after the user input the search option, the application will first check if the result is in the cache, then it will directly return the result, which is faster. If data is not found in the cache, it will go to the database to look for the match results, then save the data to the cache, and lastly return the result to the output.

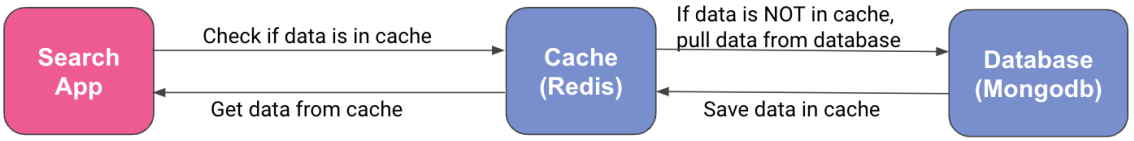

Figure 4c.1 the workflow of cache in search application

For search the popular hashtags, the function *hashtag_rank_search_helper* has been created. It can be called by using the redis_key which is format as *{1}: {"hashtag"}*, *1* can be understood as a warehouse that stores the popular hashtags at the time, and the *hashtag* is the name of the shelf in this warehouse and the actual hashtag words are the item on this shelf.

Use redis_client.exists() and redis_client.ttl() to check if the data is in the cache or not, if the data is in the cache, redis_client.exists(redis_key) will be 1, if not, it will be 0. The function, redis_client.ttl(redis_key), will return the remaining time to live of a key that has a timeout or return -1 if the key does not exist or expired within the time range. So, check whether the data is in the cache by determining whether these two outputs from these functions are greater than zero. If yes, by using redis_client.get(redis_key) to query the result.

If the data is not found in the cache, the application will go the 'else' condition which is pulling the hashtags data from the MongoDB database by generating a query, that can sort the number of times each hashtag is used, from the documents in the 'hashtags' collection from the 'twitter' database. Then, update the summary of these hashtags and their number of uses to the cache by using redis_client.setex(redis_key, time=timedelta(minutes=15), value=summary) with 15 minutes as the time range. Lastly, return the running time and popular hashtags to the user as Figure 4c.2 shows. The generation time for the first time is longer than the second time since the data is not in the cache, then data needs to be pulled from the database and saved to the cache, then be returned to the user. When it is searched for the second time within the time range of 15 minutes, the data is found in the cache and be returned directly which is much faster.

```
redis_key = """{}:{}""".format(1, "hashtags")
hashtag_rank_search_helper(redis_key)
```

```
{'_id': 'Corona', 'value': 4582.0}
{'_id': 'Mattarella', 'value': 1506.0}
{'_id': '25Aprile', 'value': 1472.0}
{'_id': 'corona', 'value': 1449.0}
{'_id': 'Covid_19', 'value': 973.0}
{'_id': 'AltaredellaPatria', 'value': 805.0}
{'_id': 'PideAlmayaDiyeÇıkıp', 'value': 776.0}
{'_id': 'COVID19', 'value': 764.0}
{'_id': 'Liberazione', 'value': 696.0}
{'_id': 'coronavirus', 'value': 629.0}

'Not found in redis cache... Pulling data from DB and updating cache  ## Popular Hashtag : Corona , Count: 4582.0 #
#   ## Popular Hashtag : Mattarella , Count: 1506.0 ##   ## Popular Hashtag : 25Aprile , Count: 1472.0 ##   ## Popu
lar Hashtag : corona , Count: 1449.0 ##   ## Popular Hashtag : Covid_19 , Count: 973.0 ##   ## Popular Hashtag : Al
taredellaPatria , Count: 805.0 ##   ## Popular Hashtag : PideAlmayaDiyeÇıkıp , Count: 776.0 ##   ## Popular Hashtag
: COVID19 , Count: 764.0 ##   ## Popular Hashtag : Liberazione , Count: 696.0 ##   ## Popular Hashtag : coronavirus
, Count: 629.0 ##    *** Summary generation time:10ms'
```

```
redis_key = """{}:{}""".format(1, "hashtags")
hashtag_rank_search_helper(redis_key)
```

```
'Found in redis cache...  ## Popular Hashtag : Corona , Count: 4582.0 ##   ## Popular Hashtag : Mattarella , Count:
1506.0 ##   ## Popular Hashtag : 25Aprile , Count: 1472.0 ##   ## Popular Hashtag : corona , Count: 1449.0 ##   ##
Popular Hashtag : Covid_19 , Count: 973.0 ##   ## Popular Hashtag : AltaredellaPatria , Count: 805.0 ##   ## Popula
r Hashtag : PideAlmayaDiyeÇıkıp , Count: 776.0 ##   ## Popular Hashtag : COVID19 , Count: 764.0 ##   ## Popular Has
htag : Liberazione , Count: 696.0 ##   ## Popular Hashtag : coronavirus , Count: 629.0 ##    *** Summary generation
time:1ms'
```

Figure 4c.2 output of the function *hashtag_rank_search_helper*

5. Results

Every search operation was timed after performing each of the options in the search application. The results are here:

Search Options	**Time (sec) - without Cache**	**Time (sec) - with Cache**
user id	2.7382290363311768	
user name	1.4618136882781982	
tweet content	0.1805858612060547	
hashtag	0.4394400119781494	
rank - user mentions	0.020933151245117188	
rank - hashtag	0.011295080184936523	0.001

We can see User Id option requires more time than all other options. Also, the rank for user mentions needs more time to run than the rank for hashtags. Moreover, operation with Cache is fast then that without Cache.

6. Conclusion

For the project, data are stored separately with user information stored in MySQL as an SQL database and with tweets information stored in MongoDB as a Non-SQL database. Also, the search application is implemented in Jupyternote Book as the user interface and the result will be queried from two databases depending on the search options. Moreover, Redis is used for caching the popular data for the 'rank' search options, which provides a more speedy result.

This is a great opportunity to learn about databases, not only from the conceptual side but also from the practical side. During the project, firstly, going through the tweets example helps get a feeling of how to use PyMongo to create database applications with MongoDB and Python. PyMongo provides a rich set of tools that can be used to communicate with a MongoDB server. It provides functionality to query, retrieve results, write and delete data, and run database commands in the Non-SQL database. Secondly, MySQL is an awesome tool for beginners to learn how to store the data as an SQL database. Within the database, support can be found for stored procedures, functions, views, cursors, and more. Thirdly, Redis is a great tool that can be quickly applied to the application once the structure has been understood. It obviously speeds up the generation time by caching the data.

Last, there is definitely a lot of room for improvement. The conditional equation could be improved. There was a MySQL query embedded into the insertion function. While a record needs to be inserted, a whole user id should be selected from the table for checking duplication. Although this insertion function could process 10,000 pieces of data very fast, 100,000 or more pieces of data must cost a large amount of time. A new duplication checking method should became up in the future. In MongoDB part of the project, three different collections were created to store the whole information. However, this should be stored in just one collection to fulfill the goal. Not only one collection would fasten the running time, but also it will make future maintenance easy. In the project, only one key category was used. Cache will be promoted with more Redis keys to save query time.

References

MongoDB. “MongoDB: The Application Data Platform | MongoDB.” MongoDB Official Website, https://www.mongodb.com/. 5/7/2022.

MySQL. “MySQL.” MySQL Official Website, https://www.mysql.com/. 5/7/2022.

Redis. “Redis.” Redis Official Website, https://redis.io/. 5/7/2022.