INFO 6210

Data Management and Database Design

NoSQL MongoDB

Assignment 3

Report

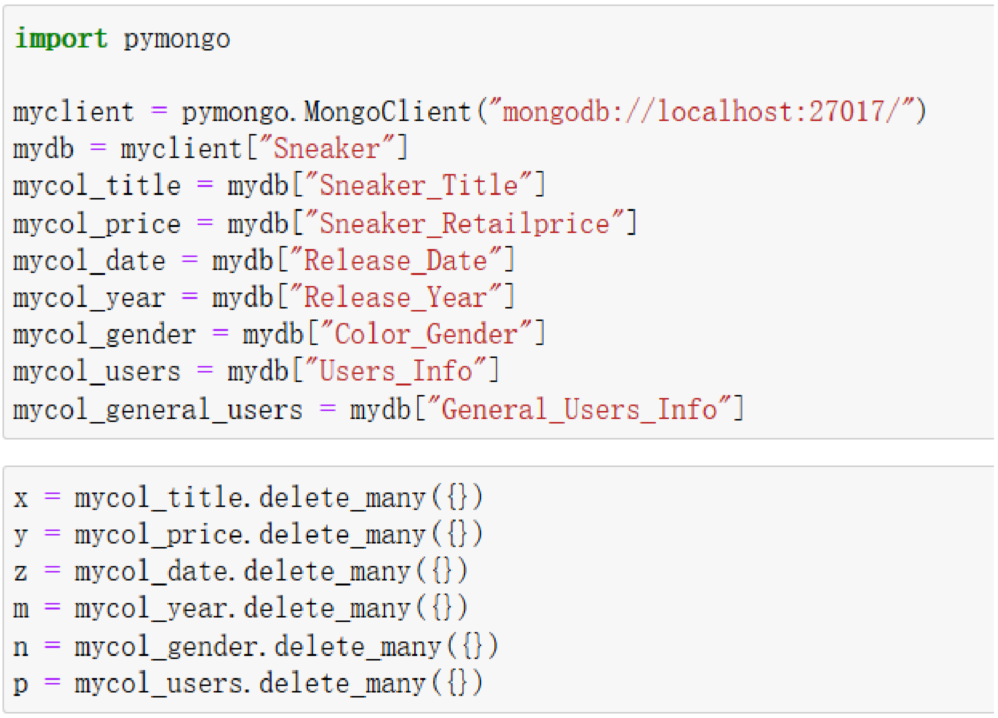
Abstract

This database mainly converts the Sneaker Relational database to the NoSQL database. For the original 5 tables in the MySQL database, I directly converted into the NoSQL database with 5 new collections. For the user's data I collected in twitter, I created two new collections to store them. Finally, queries on the database are added and questions are answered.

1. Connect to the local MongoDB database server

Import library and build the collections in the database

Here follows a screenshot of the code we write:

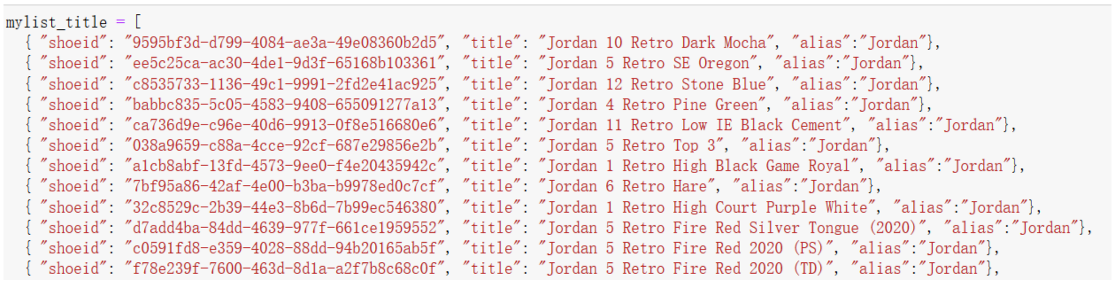


```python
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["Sneaker"]
mycol_title = mydb["Sneaker_Title"]
mycol_price = mydb["Sneaker_Retailprice"]
mycol_date = mydb["Release_Date"]
mycol_year = mydb["Release_Year"]
mycol_gender = mydb["Color_Gender"]
mycol_users = mydb["Users_Info"]
mycol_general_users = mydb["General_Users_Info"]
```

```python
x = mycol_title.delete_many({})
y = mycol_price.delete_many({})
z = mycol_date.delete_many({})
m = mycol_year.delete_many({})
n = mycol_gender.delete_many({})
p = mycol_users.delete_many({})
```

Figure 1-1-1 Screenshot of code for importing and building connection

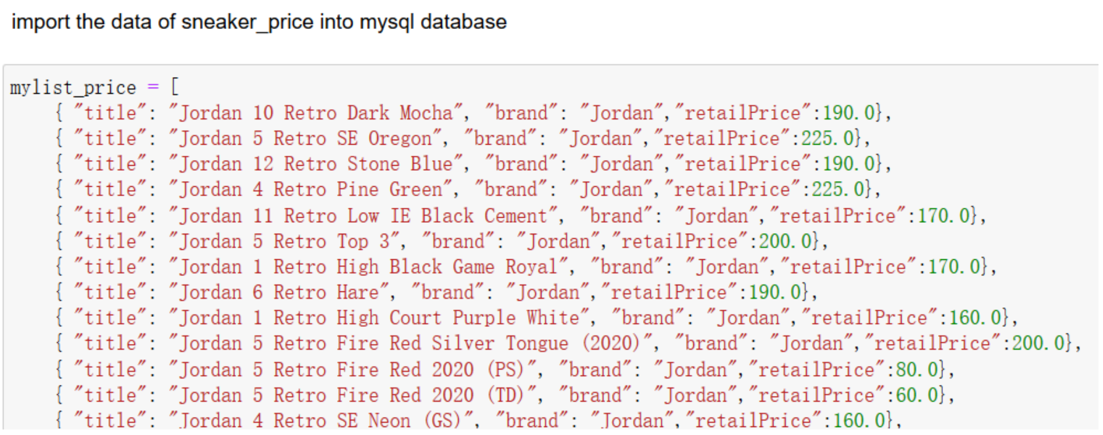
Then we import the data of sneaker_title into MySQL database:



```
mylist_title = [
  { "shoeid": "9595bf3d-d799-4084-ae3a-49e08360b2d5", "title": "Jordan 10 Retro Dark Mocha", "alias":"Jordan"},
  { "shoeid": "ee5c25ca-ac30-4de1-9d3f-65168b103361", "title": "Jordan 5 Retro SE Oregon", "alias":"Jordan"},
  { "shoeid": "c8535733-1136-49c1-9991-2fd2e41ac925", "title": "Jordan 12 Retro Stone Blue", "alias":"Jordan"},
  { "shoeid": "babbc835-5c05-4583-9408-655091277a13", "title": "Jordan 4 Retro Pine Green", "alias":"Jordan"},
  { "shoeid": "ca736d9e-c96e-40d6-9913-0f8e516680e6", "title": "Jordan 11 Retro Low IE Black Cement", "alias":"Jordan"},
  { "shoeid": "038a9659-c88a-4cce-92cf-687e29856e2b", "title": "Jordan 5 Retro Top 3", "alias":"Jordan"},
  { "shoeid": "a1cb8abf-13fd-4573-9ee0-f4e20435942c", "title": "Jordan 1 Retro High Black Game Royal", "alias":"Jordan"},
  { "shoeid": "7bf95a86-42af-4e00-b3ba-b9978ed0c7cf", "title": "Jordan 6 Retro Hare", "alias":"Jordan"},
  { "shoeid": "32c8529c-2b39-44e3-8b6d-7b99ec546380", "title": "Jordan 1 Retro High Court Purple White", "alias":"Jordan"},
  { "shoeid": "d7add4ba-84dd-4639-977f-661ce1959552", "title": "Jordan 5 Retro Fire Red Silver Tongue (2020)", "alias":"Jordan"},
  { "shoeid": "c0591fd8-e359-4028-88dd-94b20165ab5f", "title": "Jordan 5 Retro Fire Red 2020 (PS)", "alias":"Jordan"},
  { "shoeid": "f78e239f-7600-463d-8d1a-a2f7b8c68c0f", "title": "Jordan 5 Retro Fire Red 2020 (TD)", "alias":"Jordan"},
```

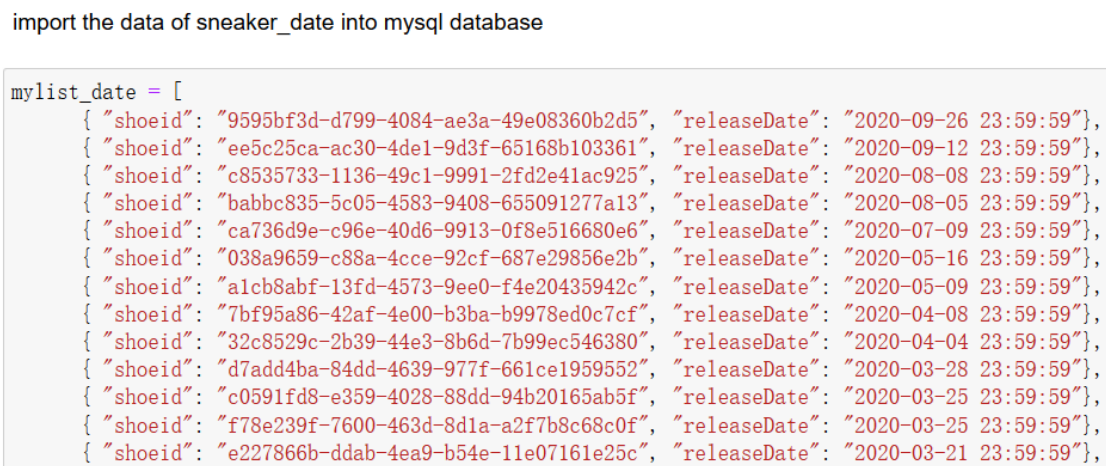
Figure 1-1-2 Screenshot of Sneaker_Title

So does the database for sneaker_price, sneaker_date, Release_year, Sneaker_gender, here follows the screenshots of those process.



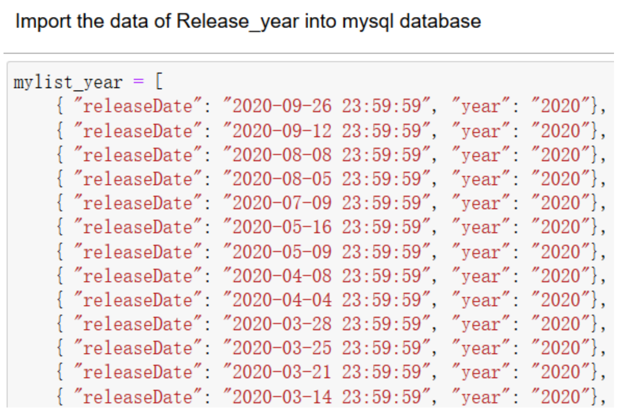
import the data of sneaker_price into mysql database

```
mylist_price = [
    { "title": "Jordan 10 Retro Dark Mocha", "brand": "Jordan","retailPrice":190.0},
    { "title": "Jordan 5 Retro SE Oregon", "brand": "Jordan","retailPrice":225.0},
    { "title": "Jordan 12 Retro Stone Blue", "brand": "Jordan","retailPrice":190.0},
    { "title": "Jordan 4 Retro Pine Green", "brand": "Jordan","retailPrice":225.0},
    { "title": "Jordan 11 Retro Low IE Black Cement", "brand": "Jordan","retailPrice":170.0},
    { "title": "Jordan 5 Retro Top 3", "brand": "Jordan","retailPrice":200.0},
    { "title": "Jordan 1 Retro High Black Game Royal", "brand": "Jordan","retailPrice":170.0},
    { "title": "Jordan 6 Retro Hare", "brand": "Jordan","retailPrice":190.0},
    { "title": "Jordan 1 Retro High Court Purple White", "brand": "Jordan","retailPrice":160.0},
    { "title": "Jordan 5 Retro Fire Red Silver Tongue (2020)", "brand": "Jordan","retailPrice":200.0},
    { "title": "Jordan 5 Retro Fire Red 2020 (PS)", "brand": "Jordan","retailPrice":80.0},
    { "title": "Jordan 5 Retro Fire Red 2020 (TD)", "brand": "Jordan","retailPrice":60.0},
    { "title": "Jordan 4 Retro SE Neon (GS)", "brand": "Jordan","retailPrice":160.0},
```



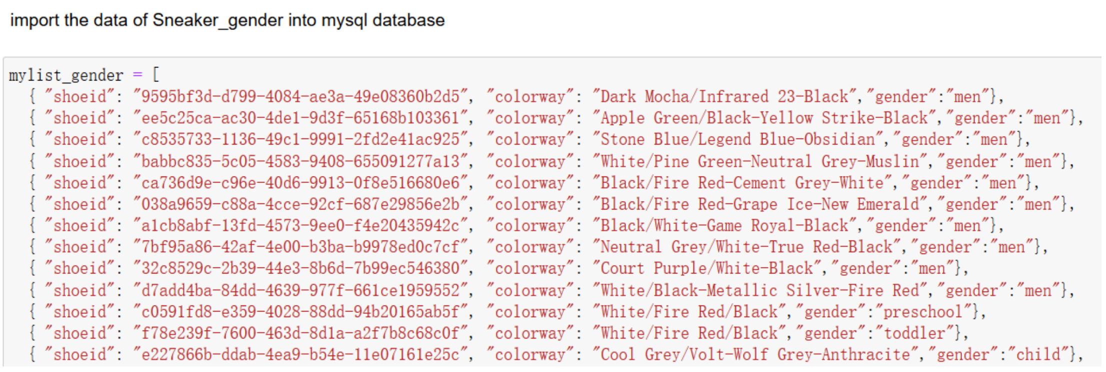
import the data of sneaker_date into mysql database

```
mylist_date = [
      { "shoeid": "9595bf3d-d799-4084-ae3a-49e08360b2d5", "releaseDate": "2020-09-26 23:59:59"},
      { "shoeid": "ee5c25ca-ac30-4de1-9d3f-65168b103361", "releaseDate": "2020-09-12 23:59:59"},
      { "shoeid": "c8535733-1136-49c1-9991-2fd2e41ac925", "releaseDate": "2020-08-08 23:59:59"},
      { "shoeid": "babbc835-5c05-4583-9408-655091277a13", "releaseDate": "2020-08-05 23:59:59"},
      { "shoeid": "ca736d9e-c96e-40d6-9913-0f8e516680e6", "releaseDate": "2020-07-09 23:59:59"},
      { "shoeid": "038a9659-c88a-4cce-92cf-687e29856e2b", "releaseDate": "2020-05-16 23:59:59"},
      { "shoeid": "a1cb8abf-13fd-4573-9ee0-f4e20435942c", "releaseDate": "2020-05-09 23:59:59"},
      { "shoeid": "7bf95a86-42af-4e00-b3ba-b9978ed0c7cf", "releaseDate": "2020-04-08 23:59:59"},
      { "shoeid": "32c8529c-2b39-44e3-8b6d-7b99ec546380", "releaseDate": "2020-04-04 23:59:59"},
      { "shoeid": "d7add4ba-84dd-4639-977f-661ce1959552", "releaseDate": "2020-03-28 23:59:59"},
      { "shoeid": "c0591fd8-e359-4028-88dd-94b20165ab5f", "releaseDate": "2020-03-25 23:59:59"},
      { "shoeid": "f78e239f-7600-463d-8d1a-a2f7b8c68c0f", "releaseDate": "2020-03-25 23:59:59"},
      { "shoeid": "e227866b-ddab-4ea9-b54e-11e07161e25c", "releaseDate": "2020-03-21 23:59:59"},
```



Import the data of Release_year into mysql database

```
mylist_year = [
    { "releaseDate": "2020-09-26 23:59:59", "year": "2020"},
    { "releaseDate": "2020-09-12 23:59:59", "year": "2020"},
    { "releaseDate": "2020-08-08 23:59:59", "year": "2020"},
    { "releaseDate": "2020-08-05 23:59:59", "year": "2020"},
    { "releaseDate": "2020-07-09 23:59:59", "year": "2020"},
    { "releaseDate": "2020-05-16 23:59:59", "year": "2020"},
    { "releaseDate": "2020-05-09 23:59:59", "year": "2020"},
    { "releaseDate": "2020-04-08 23:59:59", "year": "2020"},
    { "releaseDate": "2020-04-04 23:59:59", "year": "2020"},
    { "releaseDate": "2020-03-28 23:59:59", "year": "2020"},
    { "releaseDate": "2020-03-25 23:59:59", "year": "2020"},
    { "releaseDate": "2020-03-21 23:59:59", "year": "2020"},
    { "releaseDate": "2020-03-14 23:59:59", "year": "2020"},
```

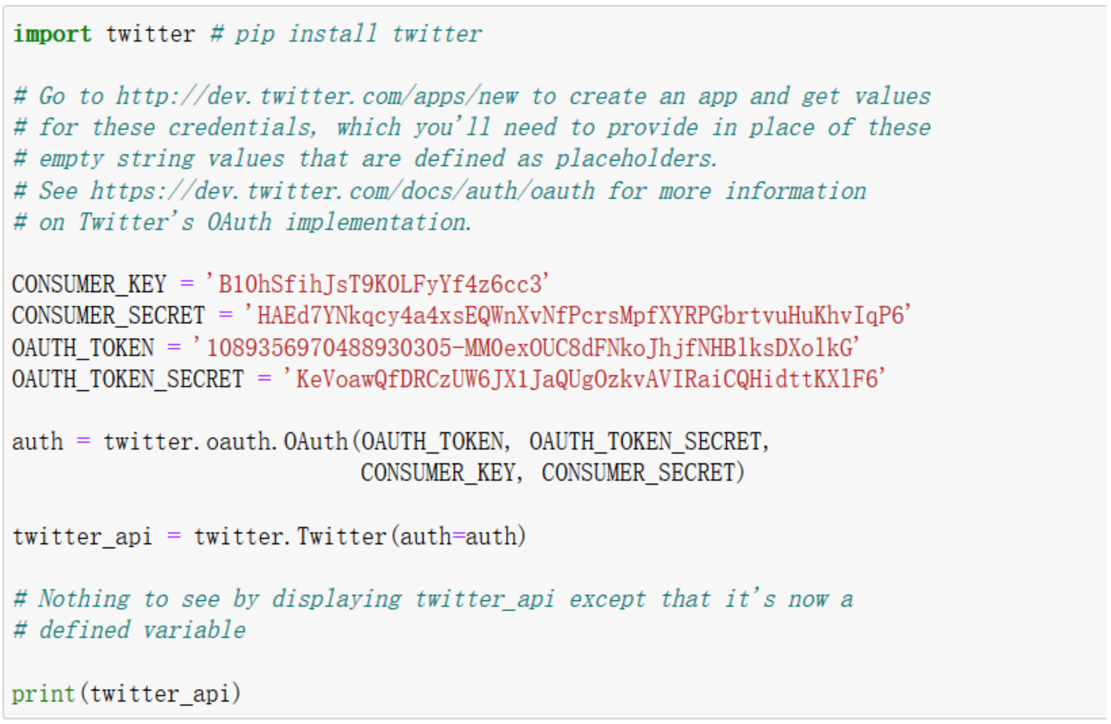

import the data of Sneaker_gender into mysql database

```
mylist_gender = [
  { "shoeid": "9595bf3d-d799-4084-ae3a-49e08360b2d5", "colorway": "Dark Mocha/Infrared 23-Black","gender":"men"},
  { "shoeid": "ee5c25ca-ac30-4de1-9d3f-65168b103361", "colorway": "Apple Green/Black-Yellow Strike-Black","gender":"men"},
  { "shoeid": "c8535733-1136-49c1-9991-2fd2e41ac925", "colorway": "Stone Blue/Legend Blue-Obsidian","gender":"men"},
  { "shoeid": "babbc835-5c05-4583-9408-655091277a13", "colorway": "White/Pine Green-Neutral Grey-Muslin","gender":"men"},
  { "shoeid": "ca736d9e-c96e-40d6-9913-0f8e516680e6", "colorway": "Black/Fire Red-Cement Grey-White","gender":"men"},
  { "shoeid": "038a9659-c88a-4cce-92cf-687e29856e2b", "colorway": "Black/Fire Red-Grape Ice-New Emerald","gender":"men"},
  { "shoeid": "a1cb8abf-13fd-4573-9ee0-f4e20435942c", "colorway": "Black/White-Game Royal-Black","gender":"men"},
  { "shoeid": "7bf95a86-42af-4e00-b3ba-b9978ed0c7cf", "colorway": "Neutral Grey/White-True Red-Black","gender":"men"},
  { "shoeid": "32c8529c-2b39-44e3-8b6d-7b99ec546380", "colorway": "Court Purple/White-Black","gender":"men"},
  { "shoeid": "d7add4ba-84dd-4639-977f-661ce1959552", "colorway": "White/Black-Metallic Silver-Fire Red","gender":"men"},
  { "shoeid": "c0591fd8-e359-4028-88dd-94b20165ab5f", "colorway": "White/Fire Red/Black","gender":"preschool"},
  { "shoeid": "f78e239f-7600-463d-8d1a-a2f7b8c68c0f", "colorway": "White/Fire Red/Black","gender":"toddler"},
  { "shoeid": "e227866b-ddab-4ea9-b54e-11e07161e25c", "colorway": "Cool Grey/Volt-Wolf Grey-Anthracite","gender":"child"},
```

Figure 1-1-3 to Figure 1-1-6 Screenshot of sneaker_data, release_year and Sneak_gender

2. Adding social media User into the database

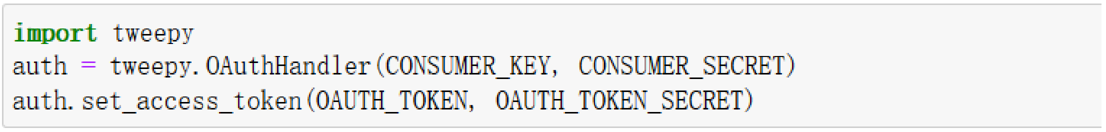
Authenticate the twitter user



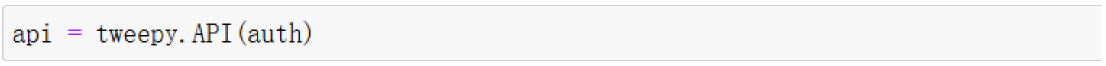
```
import twitter # pip install twitter

# Go to http://dev.twitter.com/apps/new to create an app and get values
# for these credentials, which you'll need to provide in place of these
# empty string values that are defined as placeholders.
# See https://dev.twitter.com/docs/auth/oauth for more information
# on Twitter's OAuth implementation.

CONSUMER_KEY = 'B10hSfihJsT9K0LFyYf4z6cc3'
CONSUMER_SECRET = 'HAEd7YNkqcy4a4xsEQWnXvNfPcrsMpfXYRPGbrtvuHuKhvIqP6'
OAUTH_TOKEN = '1089356970488930305-MM0exOUC8dFNkoJhjfNHBlksDXolkG'
OAUTH_TOKEN_SECRET = 'KeVoawQfDRCzUW6JX1JaQUgOzkvAVIRaiCQHidttKXlF6'

auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
                           CONSUMER_KEY, CONSUMER_SECRET)

twitter_api = twitter.Twitter(auth=auth)

# Nothing to see by displaying twitter_api except that it's now a
# defined variable

print(twitter_api)
```

Figure 2-1-1 screenshot of code of the Authenticate twitter user

Authenticate the Tweepy handler



```
import tweepy
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)
```

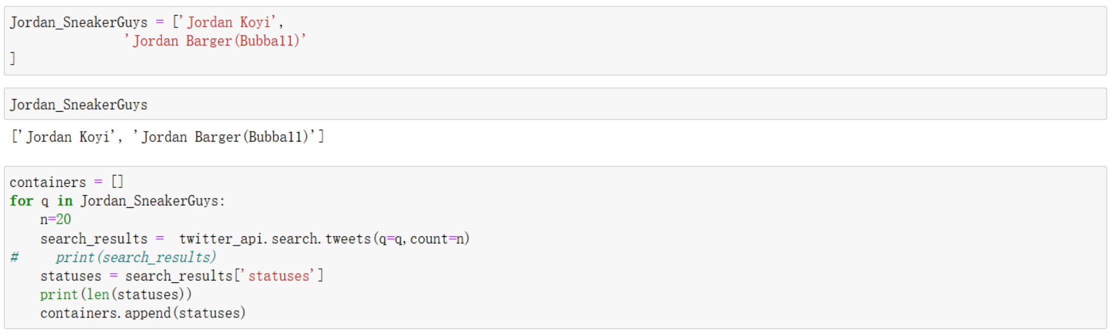


```
api = tweepy.API(auth)
```

Adding the user information to the database

The Intuition thought of mine on my NoSQL database is that the people might be interested in what people say or behave on twitter based on certain brand of sneakers, namely, Jordan, Nike and adidas. The Users collection would contain 3 documents, in each document's value, there would be a list of documents storing the user information of that certain brand. I also added a foreign key called 'alias' to link to the new users table. Here is the example of code we write for the Jordan_ueser.

Jordan_users

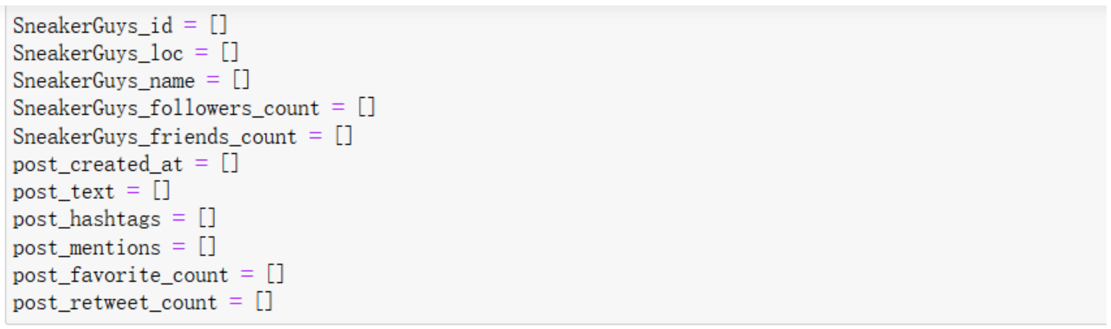


```
Jordan_SneakerGuys = ['Jordan Koyi',
              'Jordan Barger(Bubba11)'
]
```

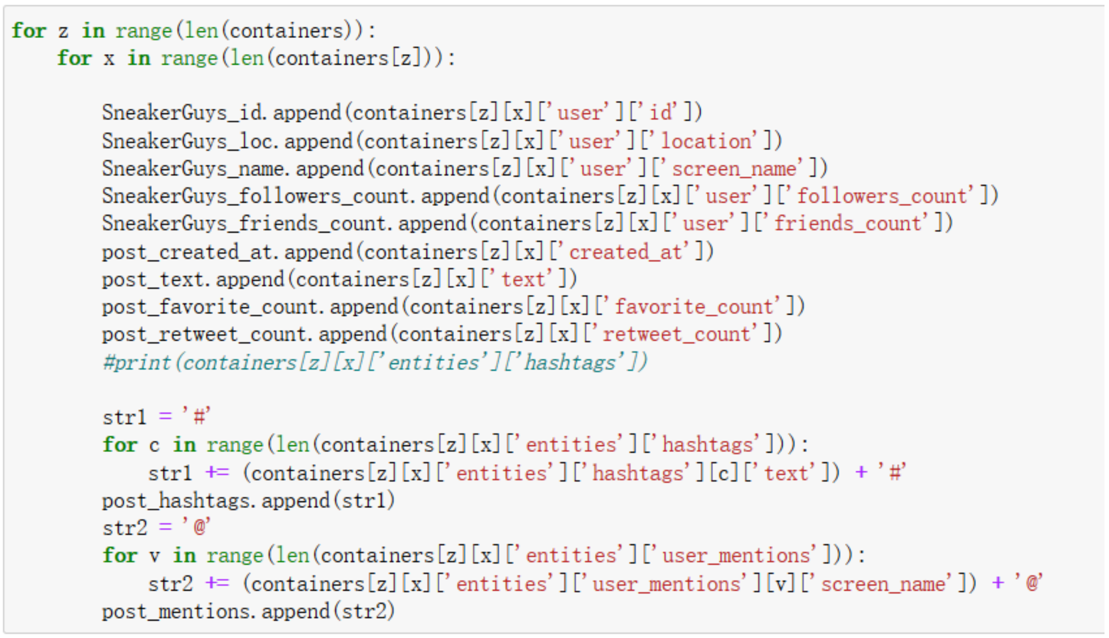
```
Jordan_SneakerGuys
```

```
['Jordan Koyi', 'Jordan Barger(Bubba11)']
```

```
containers = []
for q in Jordan_SneakerGuys:
    n=20
    search_results =  twitter_api.search.tweets(q=q, count=n)
#     print(search_results)
    statuses = search_results['statuses']
    print(len(statuses))
    containers.append(statuses)
```



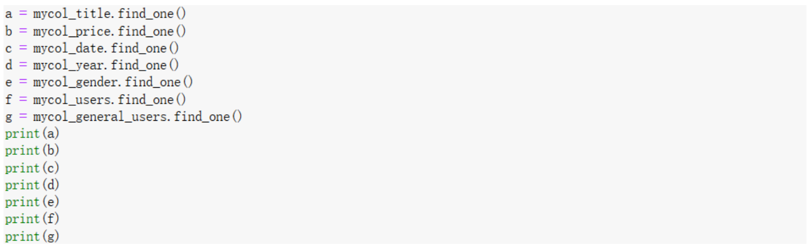
```
SneakerGuys_id = []
SneakerGuys_loc = []
SneakerGuys_name = []
SneakerGuys_followers_count = []
SneakerGuys_friends_count = []
post_created_at = []
post_text = []
post_hashtags = []
post_mentions = []
post_favorite_count = []
post_retweet_count = []
```



```
for z in range(len(containers)):
    for x in range(len(containers[z])):

        SneakerGuys_id.append(containers[z][x]['user']['id'])
        SneakerGuys_loc.append(containers[z][x]['user']['location'])
        SneakerGuys_name.append(containers[z][x]['user']['screen_name'])
        SneakerGuys_followers_count.append(containers[z][x]['user']['followers_count'])
        SneakerGuys_friends_count.append(containers[z][x]['user']['friends_count'])
        post_created_at.append(containers[z][x]['created_at'])
        post_text.append(containers[z][x]['text'])
        post_favorite_count.append(containers[z][x]['favorite_count'])
        post_retweet_count.append(containers[z][x]['retweet_count'])
        #print(containers[z][x]['entities']['hashtags'])

        str1 = '#'
        for c in range(len(containers[z][x]['entities']['hashtags'])):
            str1 += (containers[z][x]['entities']['hashtags'][c]['text']) + '#'
        post_hashtags.append(str1)
        str2 = '@'
        for v in range(len(containers[z][x]['entities']['user_mentions'])):
            str2 += (containers[z][x]['entities']['user_mentions'][v]['screen_name']) + '@'
        post_mentions.append(str2)
```

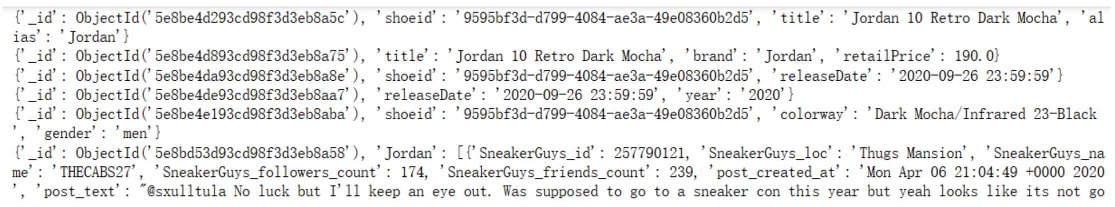
3. Querying the MongoDB database

(1). Find the first field of each element in the database



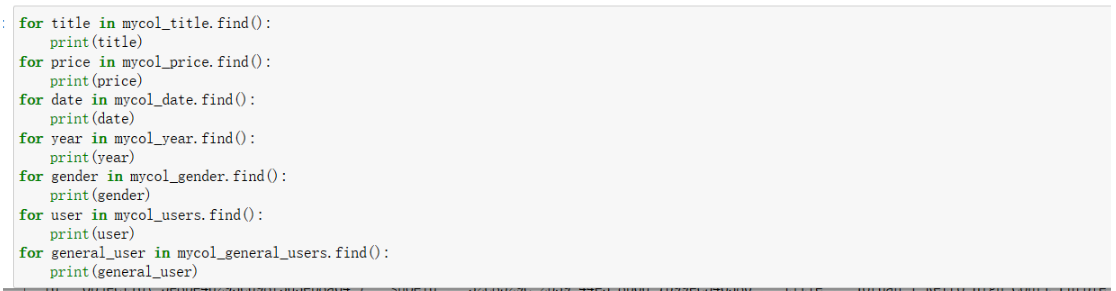
```
a = mycol_title.find_one()
b = mycol_price.find_one()
c = mycol_date.find_one()
d = mycol_year.find_one()
e = mycol_gender.find_one()
f = mycol_users.find_one()
g = mycol_general_users.find_one()
print(a)
print(b)
print(c)
print(d)
print(e)
print(f)
print(g)
```

Here is the screenshot of results:

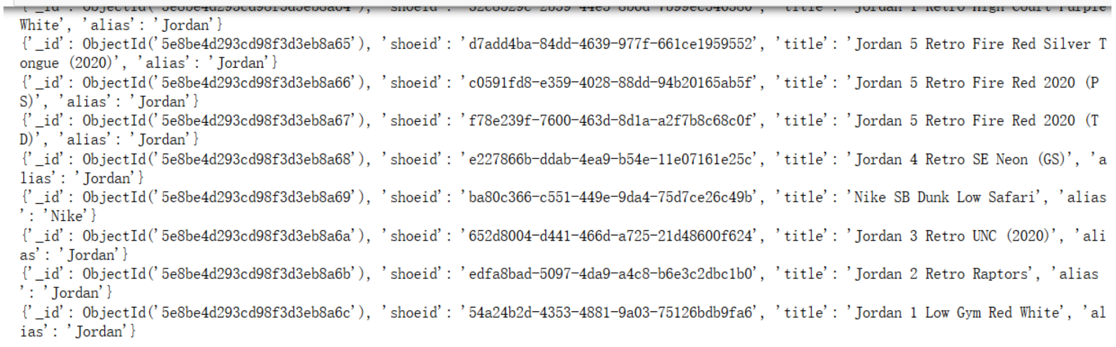


```
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a5c'), 'shoeid': '9595bf3d-d799-4084-ae3a-49e08360b2d5', 'title': 'Jordan 10 Retro Dark Mocha', 'al
ias': 'Jordan'}
{'_id': ObjectId('5e8be4d893cd98f3d3eb8a75'), 'title': 'Jordan 10 Retro Dark Mocha', 'brand': 'Jordan', 'retailPrice': 190.0}
{'_id': ObjectId('5e8be4da93cd98f3d3eb8a8e'), 'shoeid': '9595bf3d-d799-4084-ae3a-49e08360b2d5', 'releaseDate': '2020-09-26 23:59:59'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aa7'), 'releaseDate': '2020-09-26 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4e193cd98f3d3eb8aba'), 'shoeid': '9595bf3d-d799-4084-ae3a-49e08360b2d5', 'colorway': 'Dark Mocha/Infrared 23-Black
', 'gender': 'men'}
{'_id': ObjectId('5e8bd53d93cd98f3d3eb8a58'), 'Jordan': [{'SneakerGuys_id': 257790121, 'SneakerGuys_loc': 'Thugs Mansion', 'SneakerGuys_na
me': 'THECABS27', 'SneakerGuys_followers_count': 174, 'SneakerGuys_friends_count': 239, 'post_created_at': 'Mon Apr 06 21:04:49 +0000 2020
', 'post_text': "@sxulltula No luck but I'll keep an eye out. Was supposed to go to a sneaker con this year but yeah looks like its not go
```

(2)Find all the documents in each collection

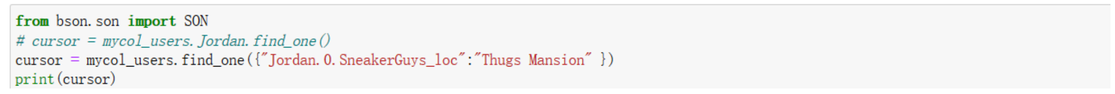


```
for title in mycol_title.find():
    print(title)
for price in mycol_price.find():
    print(price)
for date in mycol_date.find():
    print(date)
for year in mycol_year.find():
    print(year)
for gender in mycol_gender.find():
    print(gender)
for user in mycol_users.find():
    print(user)
for general_user in mycol_general_users.find():
    print(general_user)
```

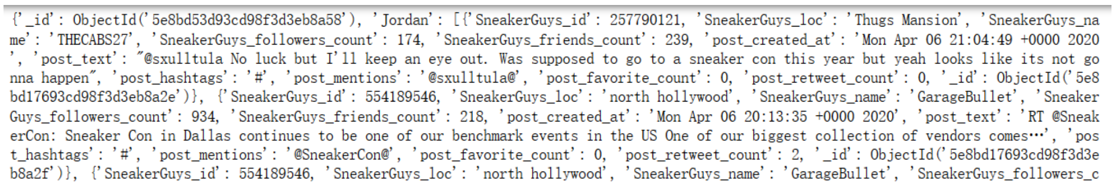


```
White', 'alias': 'Jordan'}
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a65'), 'shoeid': 'd7add4ba-84dd-4639-977f-661ce1959552', 'title': 'Jordan 5 Retro Fire Red Silver T
ongue (2020)', 'alias': 'Jordan'}
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a66'), 'shoeid': 'c0591fd8-e359-4028-88dd-94b20165ab5f', 'title': 'Jordan 5 Retro Fire Red 2020 (P
S)', 'alias': 'Jordan'}
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a67'), 'shoeid': 'f78e239f-7600-463d-8d1a-a2f7b8c68c0f', 'title': 'Jordan 5 Retro Fire Red 2020 (T
D)', 'alias': 'Jordan'}
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a68'), 'shoeid': 'e227866b-ddab-4ea9-b54e-11e07161e25c', 'title': 'Jordan 4 Retro SE Neon (GS)', 'a
lias': 'Jordan'}
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a69'), 'shoeid': 'ba80c366-c551-449e-9da4-75d7ce26c49b', 'title': 'Nike SB Dunk Low Safari', 'alias
': 'Nike'}
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a6a'), 'shoeid': '652d8004-d441-466d-a725-21d48600f624', 'title': 'Jordan 3 Retro UNC (2020)', 'ali
as': 'Jordan'}
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a6b'), 'shoeid': 'edfa8bad-5097-4da9-a4c8-b6e3c2dbc1b0', 'title': 'Jordan 2 Retro Raptors', 'alias
': 'Jordan'}
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a6c'), 'shoeid': '54a24b2d-4353-4881-9a03-75126bdb9fa6', 'title': 'Jordan 1 Low Gym Red White', 'al
ias': 'Jordan'}
```

(3) Find what collection does the field belongs to in the user_collection and retrive all the information in that certain group



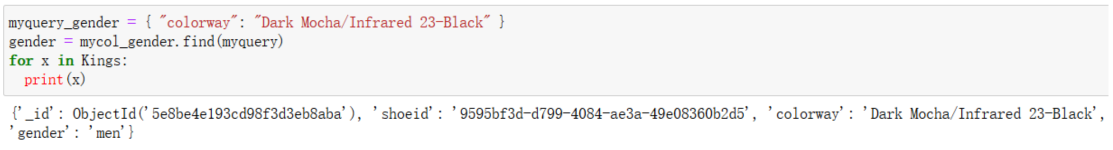
```
from bson.son import SON
# cursor = mycol_users.Jordan.find_one()
cursor = mycol_users.find_one({"Jordan.0.SneakerGuys_loc":"Thugs Mansion" })
print(cursor)
```



```
{'_id': ObjectId('5e8bd53d93cd98f3d3eb8a58'), 'Jordan': [{'SneakerGuys_id': 257790121, 'SneakerGuys_loc': 'Thugs Mansion', 'SneakerGuys_na
me': 'THECABS27', 'SneakerGuys_followers_count': 174, 'SneakerGuys_friends_count': 239, 'post_created_at': 'Mon Apr 06 21:04:49 +0000 2020
', 'post_text': "@sxulltula No luck but I'll keep an eye out. Was supposed to go to a sneaker con this year but yeah looks like its not go
nna happen", 'post_hashtags': '#', 'post_mentions': '@sxulltula@', 'post_favorite_count': 0, 'post_retweet_count': 0, '_id': ObjectId('5e8
bd17693cd98f3d3eb8a2e')}, {'SneakerGuys_id': 554189546, 'SneakerGuys_loc': 'north hollywood', 'SneakerGuys_name': 'GarageBullet', 'Sneaker
Guys_followers_count': 934, 'SneakerGuys_friends_count': 218, 'post_created_at': 'Mon Apr 06 20:13:35 +0000 2020', 'post_text': 'RT @Sneak
erCon: Sneaker Con in Dallas continues to be one of our benchmark events in the US One of our biggest collection of vendors comes…', 'pos
t_hashtags': '#', 'post_mentions': '@SneakerCon@', 'post_favorite_count': 0, 'post_retweet_count': 2, '_id': ObjectId('5e8bd17693cd98f3d3e
b8a2f')}, {'SneakerGuys_id': 554189546, 'SneakerGuys_loc': 'north hollywood', 'SneakerGuys_name': 'GarageBullet', 'SneakerGuys_followers_c
```

(4) Find a specific field with certain value

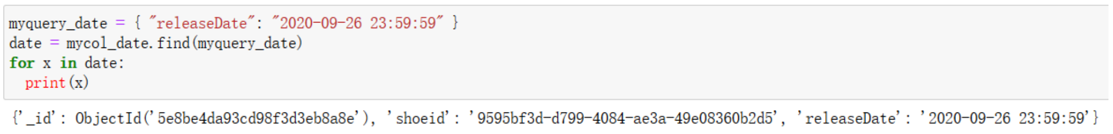
(a) In the color_gender collection, find the document whose colorway value is Dark Mocha/Infrared 23-Black



```
myquery_gender = { "colorway": "Dark Mocha/Infrared 23-Black" }
gender = mycol_gender.find(myquery)
for x in Kings:
  print(x)
```

```
{'_id': ObjectId('5e8be4e193cd98f3d3eb8aba'), 'shoeid': '9595bf3d-d799-4084-ae3a-49e08360b2d5', 'colorway': 'Dark Mocha/Infrared 23-Black',
'gender': 'men'}
```

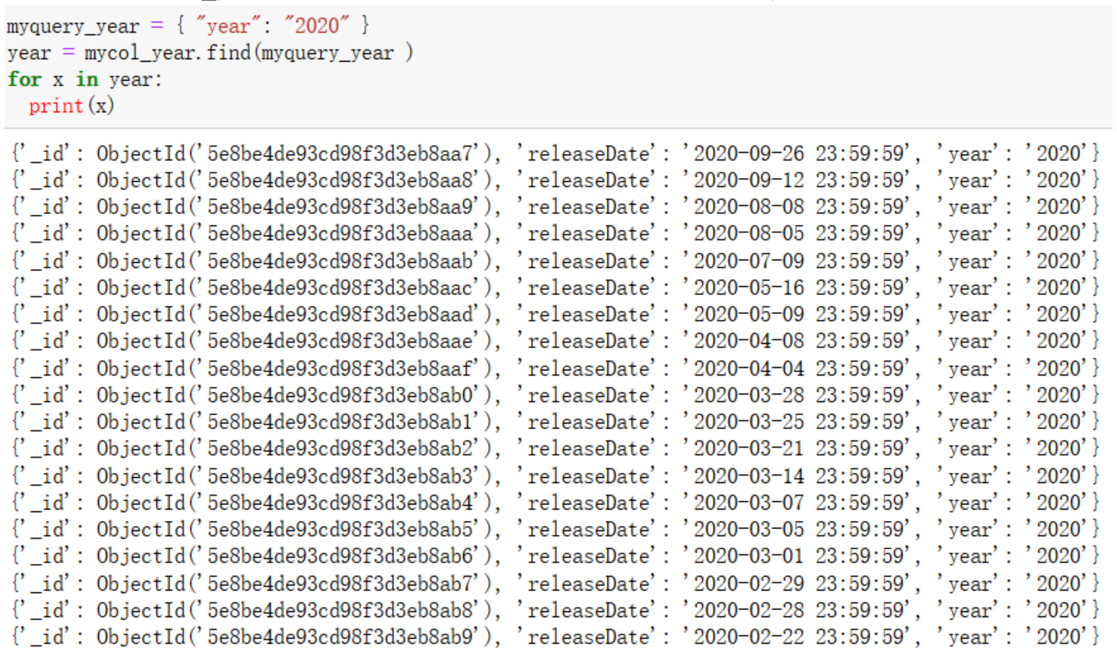
(b)In the Release_date collection, find the document whosereleaseDate is 2020-09-26 23:59:59.



```
myquery_date = { "releaseDate": "2020-09-26 23:59:59" }
date = mycol_date.find(myquery_date)
for x in date:
  print(x)
```

```
{'_id': ObjectId('5e8be4da93cd98f3d3eb8a8e'), 'shoeid': '9595bf3d-d799-4084-ae3a-49e08360b2d5', 'releaseDate': '2020-09-26 23:59:59'}
```

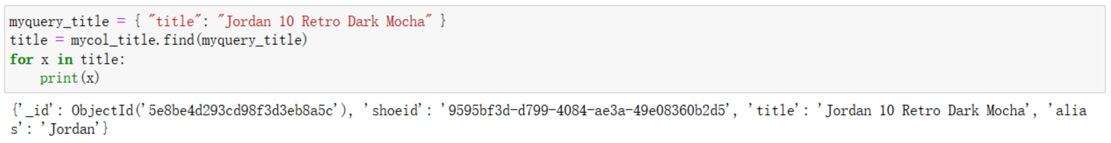
(c) In the Release_year collection, find the document whose release year is 2020.



```
myquery_year = { "year": "2020" }
year = mycol_year.find(myquery_year )
for x in year:
  print(x)
```

```
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aa7'), 'releaseDate': '2020-09-26 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aa8'), 'releaseDate': '2020-09-12 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aa9'), 'releaseDate': '2020-08-08 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aaa'), 'releaseDate': '2020-08-05 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aab'), 'releaseDate': '2020-07-09 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aac'), 'releaseDate': '2020-05-16 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aad'), 'releaseDate': '2020-05-09 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aae'), 'releaseDate': '2020-04-08 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8aaf'), 'releaseDate': '2020-04-04 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab0'), 'releaseDate': '2020-03-28 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab1'), 'releaseDate': '2020-03-25 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab2'), 'releaseDate': '2020-03-21 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab3'), 'releaseDate': '2020-03-14 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab4'), 'releaseDate': '2020-03-07 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab5'), 'releaseDate': '2020-03-05 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab6'), 'releaseDate': '2020-03-01 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab7'), 'releaseDate': '2020-02-29 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab8'), 'releaseDate': '2020-02-28 23:59:59', 'year': '2020'}
{'_id': ObjectId('5e8be4de93cd98f3d3eb8ab9'), 'releaseDate': '2020-02-22 23:59:59', 'year': '2020'}
```

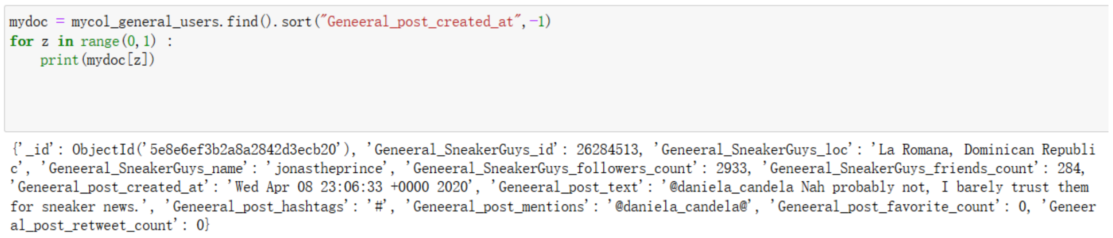
(d) In the Sneaker_title collection, find the document whose Sneaker_title is Jordan 10 Retro Dark Mocha



```
myquery_title = { "title": "Jordan 10 Retro Dark Mocha" }
title = mycol_title.find(myquery_title)
for x in title:
    print(x)
```

```
{'_id': ObjectId('5e8be4d293cd98f3d3eb8a5c'), 'shoeid': '9595bf3d-d799-4084-ae3a-49e08360b2d5', 'title': 'Jordan 10 Retro Dark Mocha', 'alia
s': 'Jordan'}
```

(e) Find the latest post of general_users collection



```
mydoc = mycol_general_users.find().sort("Geneeral_post_created_at",-1)
for z in range(0,1) :
    print(mydoc[z])
```

```
{'_id': ObjectId('5e8e6ef3b2a8a2842d3ecb20'), 'Geneeral_SneakerGuys_id': 26284513, 'Geneeral_SneakerGuys_loc': 'La Romana, Dominican Republi
c', 'Geneeral_SneakerGuys_name': 'jonastheprince', 'Geneeral_SneakerGuys_followers_count': 2933, 'Geneeral_SneakerGuys_friends_count': 284,
'Geneeral_post_created_at': 'Wed Apr 08 23:06:33 +0000 2020', 'Geneeral_post_text': '@daniela_candela Nah probably not, I barely trust them
for sneaker news.', 'Geneeral_post_hashtags': '#', 'Geneeral_post_mentions': '@daniela_candela@', 'Geneeral_post_favorite_count': 0, 'Geneer
al_post_retweet_count': 0}
```

(f) Find the sneaker whose retail price is 190.0

```
myquery_price = { "retailPrice": 190.0 }
prices = mycol_price.find(myquery_price)
for x in prices:
    print(x)
```

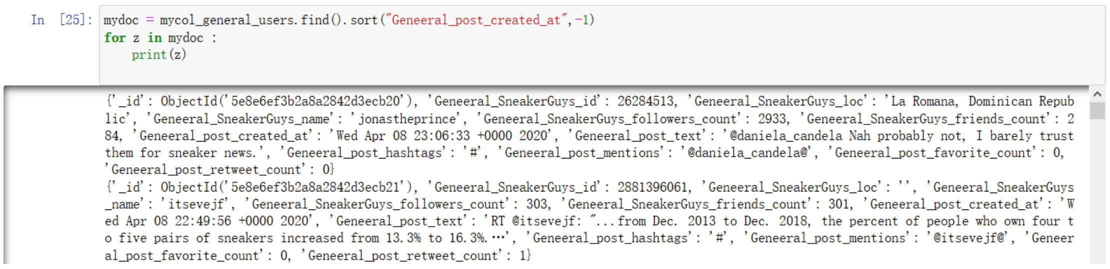
```
{'_id': ObjectId('5e8be4d893cd98f3d3eb8a75'), 'title': 'Jordan 10 Retro Dark Mocha', 'brand': 'Jordan', 'retailPrice': 190.0}
{'_id': ObjectId('5e8be4d893cd98f3d3eb8a77'), 'title': 'Jordan 12 Retro Stone Blue', 'brand': 'Jordan', 'retailPrice': 190.0}
{'_id': ObjectId('5e8be4d893cd98f3d3eb8a7c'), 'title': 'Jordan 6 Retro Hare', 'brand': 'Jordan', 'retailPrice': 190.0}
{'_id': ObjectId('5e8be4d893cd98f3d3eb8a83'), 'title': 'Jordan 3 Retro UNC (2020)', 'brand': 'Jordan', 'retailPrice': 190.0}
{'_id': ObjectId('5e8be4d893cd98f3d3eb8a84'), 'title': 'Jordan 2 Retro Raptors', 'brand': 'Jordan', 'retailPrice': 190.0}
```

(g) Find the most popular guy in the comments by the number of followers.

```
mydoc = mycol_general_users.find().sort("Geneeral_SneakerGuys_name", -1)
for z in range(0,1):
    print(mydoc[z])
```

```
{'_id': ObjectId('5e8e5186343e20771cb402a8'), 'Geneeral_SneakerGuys_id': 750002134440116225, 'Geneeral_SneakerGuys_loc': '', 'Geneeral_Sneak
erGuys_name': 'whitekniife', 'Geneeral_SneakerGuys_followers_count': 1408, 'Geneeral_SneakerGuys_friends_count': 1944, 'Geneeral_post_create
d_at': 'Wed Apr 08 22:20:34 +0000 2020', 'Geneeral_post_text': 'Swish and flick https://t.co/nHkyTmHoCz', 'Geneeral_post_hashtags': '#', 'Ge
neeral_post_mentions': '@', 'Geneeral_post_favorite_count': 0, 'Geneeral_post_retweet_count': 0}
```

(h) Find the latest post time of general_users collection



```
In  [25]: mydoc = mycol_general_users.find().sort("Geneeral_post_created_at",-1)
          for z in mydoc :
              print(z)
```

```
{'_id': ObjectId('5e8e6ef3b2a8a2842d3ecb20'), 'Geneeral_SneakerGuys_id': 26284513, 'Geneeral_SneakerGuys_loc': 'La Romana, Dominican Repub
lic', 'Geneeral_SneakerGuys_name': 'jonastheprince', 'Geneeral_SneakerGuys_followers_count': 2933, 'Geneeral_SneakerGuys_friends_count': 2
84, 'Geneeral_post_created_at': 'Wed Apr 08 23:06:33 +0000 2020', 'Geneeral_post_text': '@daniela_candela Nah probably not, I barely trust
them for sneaker news.', 'Geneeral_post_hashtags': '#', 'Geneeral_post_mentions': '@daniela_candela@', 'Geneeral_post_favorite_count': 0,
'Geneeral_post_retweet_count': 0}
{'_id': ObjectId('5e8e6ef3b2a8a2842d3ecb21'), 'Geneeral_SneakerGuys_id': 2881396061, 'Geneeral_SneakerGuys_loc': '', 'Geneeral_SneakerGuys
_name': 'itsevejf', 'Geneeral_SneakerGuys_followers_count': 303, 'Geneeral_SneakerGuys_friends_count': 301, 'Geneeral_post_created_at': 'W
ed Apr 08 22:49:56 +0000 2020', 'Geneeral_post_text': 'RT @itsevejf: "...from Dec. 2013 to Dec. 2018, the percent of people who own four t
o five pairs of sneakers increased from 13.3% to 16.3%.…', 'Geneeral_post_hashtags': '#', 'Geneeral_post_mentions': '@itsevejf@', 'Geneer
al_post_favorite_count': 0, 'Geneeral_post_retweet_count': 1}
```

(5) What tags are associated with a person, place or thing?

For the Users_info collection, I basically tagged all the users to 'Jordan', 'Nike', and 'Addidas' to show their relationship to the specific categories they belong to.

(6) What social media users are like other social media users in your domain?

For this question, I picked two most active users in my general_collection and calculate their similarities.

```
from pymongo import MongoClient
from difflib import SequenceMatcher


coll = mydb["General_Users_Info"]

def uid_chats():
    list_chat = list(coll.find({"Geneeral_SneakerGuys_name": "sneaker_camel"}, {"_id": 0, "Geneeral_post_text": 1}))
    chats = [d['Geneeral_post_text'] for d in list_chat]
    print(chats)
    return chats


def similar_ratio(strA, strB):
    return SequenceMatcher(lambda x:x==" ", strA, strB).ratio()

#find the similarities among the list
def compute_similar():
    chats = uid_chats()
    for index in range(len(chats) - 1):
        ratios = similar_ratio(chats[index], chats[index+1])
        print(ratios)


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

```
['広島県 新たに４人感染確認 計24人に 新型コロナウイルス | NHKニュース   https://t.co/1p7pyfXCFz\n広島もついに増えてきた。\nしかも福山市の男性の1人は感染経路不明? \nそしてどの人も県外に出てから感… https://t.co/y7rz4IxDjC', '富山 新たに30代男性の感染確認 県内14人に 新型コロナ | NHKニュース   https://t.co/JZri6NoKkN\n高岡だったら金沢経由の感染も疑った方がいい。\n福井に行く途中金沢に立ち寄ってない? \n福井は今のとこ… https://t.co/p3049kIJT5', '静岡 浜松で60代男性の感染確認 新型コロナウイルス | NHKニュース   https://t.co/HgRgbIEBu5\nNHKが珍しく静岡の記事を載せてる…それは置いといて。\nえっ? 先月21日に京都で感染して今更陽性反応? ! … https://t.co/0uIKIfMkLr', '石川県は、11人が新型コロナウイルスに感染していることが新たに確認されたと発表しました。石川県での感染確認は、これで66… https://t.co/TY4Uyg82qX\nあーあ、終わったな。\nはよ、会見しろや! \n石川の約2倍の人… https://t.co/VjjpvY7rpL', '東京都 新たに144人の感染確認 １日の確認数で最多 | NHKニュース   https://t.co/01uHdwb9F0\nふーんあっそうですか。\n日曜日に検査サボったからそれが回ってきたって事ね。\n昨日、一昨日は休み明けで少な… https://t.co/JzoiLmerGo', '名古屋 新たに９人の感染確認 愛知県内で延べ260人 新型コロナ | NHKニュース   https://t.co/aWcihk0Juv\nえっ? さっき7人出たってニュース見て、またプラス9人? \nさらに豊橋で2人出てるから計18人。… https://t.co/eU00tB9PQu', '河村市長“名古屋も緊急事態宣言対象に” 大村知事は慎重姿勢 | NHKニュース   https://t.co/9mXmrtTJWc\n河村市長の意見に賛成です。\n大村は相変わらず危機感ゼロだけど、名古屋市だけが対象になってもその周辺… https://t.co/X1WKlmNQ37', '沖縄県 新たに12人感染確認 新型コロナウイルス | NHKニュース   https://t.co/krySAoSiYX\n感染経路不明が7人って沖縄ヤバくない? \n宮古島に疎開した人から移されたんじゃないだろうね? ']
0.5
0.4642857142857143
0.3142857142857143
0.32857142857142857
0.45714285714285713
0.37142857142857144
0.2682926829268293
```

```python
from pymongo import MongoClient
from difflib import SequenceMatcher


coll = mydb["General_Users_Info"]

def uid_chats():
    list_chat = list(coll.find({"Geneeral_SneakerGuys_name": "swish_bs"}, {"_id": 0, "Geneeral_post_text": 1}))
    chats = [d['Geneeral_post_text'] for d in list_chat]
    print(chats)
    return chats


def similar_ratio(strA, strB):
    return SequenceMatcher(lambda x:x==" ", strA, strB).ratio()

#find the similarities among the list
def compute_similar():
    chats = uid_chats()
    for index in range(len(chats) - 1):
        ratios = similar_ratio(chats[index], chats[index+1])
        print(ratios)


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

```
['@Nubbz3YT @BrawlStars sandy😴😴😴', '@ApocYT3 @Nubbz3YT @BrawlStars @Kodii_20  Y @Cesar30057276', 'RT @ApocYT3: ⭐⭐HORUS BO GIVEAWAY ⭐⭐\n\n⚒Rules⚒\n\n 🔥Follow @ApocYT3 and @Nubbz3YT \n🔥Retweet &amp; Like\n🔥Tag 2 friends \n\n@BrawlStars #BrawlStars…', '@x6tence_ES @BsReivaj @Moncor23 @AleSSJ666 @x6tence @Kodii_20  Y @Cesar30057276', 'RT @x6tence_ES: 🎁 ¡SORTEO DE #BRAWLSTARS! 🎁\n\n⭐ 3 Premios de 15 € en Google Play o PayPal ⭐\n\n➡ Follow @Moncor23\n@BsReivaj y @AleSSJ666. \n\n🔄…']
0.5454545454545454
0.3283582089552239
0.13513513513513514
0.29357798165137616
```

From the above results we could see that the two most popular users' text similarities are generally decreasing with time goes on.

(7)What people, places or things are popular in your domain?

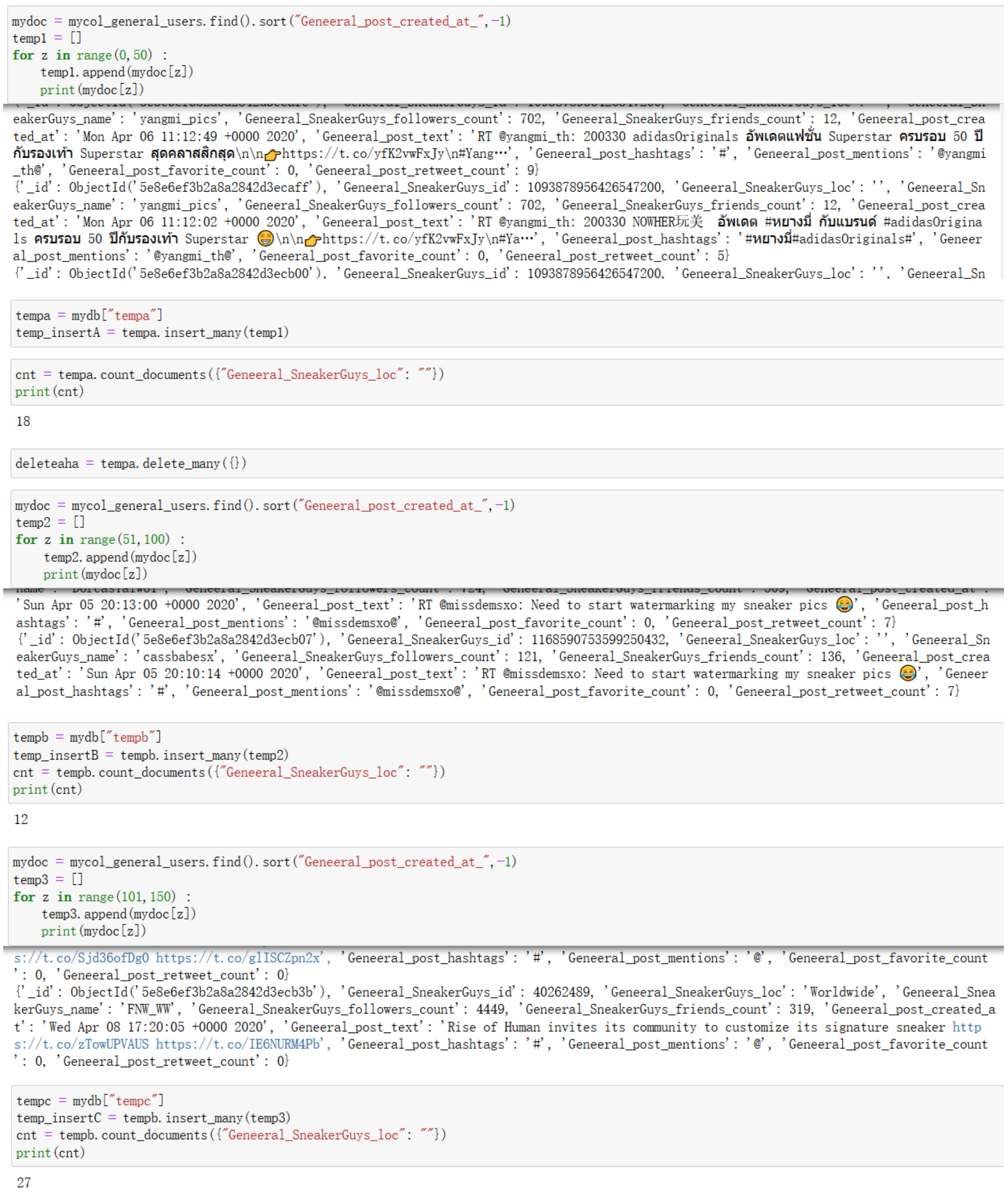
From the below sorted data based on Geneeral_SneakerGuys_followers_count, we could see that the guy named ezr4reeves is the most popular guy in this collection and has the most followers.

```python
mydoc = mycol_general_users.find().sort("Geneeral_SneakerGuys_followers_count_",-1)
for z in mydoc :
    print(z)
```

```
{'_id': ObjectId('5e8e5186343e20771cb40297'), 'Geneeral_SneakerGuys_id': 1168290958120837122, 'Geneeral_SneakerGuys_loc': '', 'Geneeral_SneakerGuys_name': 'ezr4reeves', 'Geneeral_SneakerGuys_followers_count': 627, 'Geneeral_SneakerGuys_friends_count': 607, 'Geneeral_post_created_at': 'Wed Apr 08 22:34:28 +0000 2020', 'Geneeral_post_text': "RT @artelothbrok: So keep calm, honey, I'mma stick around for more than a minute, get used to it. Funny my name keeps comin' out your mouth…", 'Geneeral_post_hashtags': '#', 'Geneeral_post_mentions': '@artelothbrok@', 'Geneeral_post_favorite_count': 0, 'Geneeral_post_retweet_count': 32}
```

(8) What people, places or things are trending in your domain?

For this part, I would like to check whether users on twitter are more likely to indicate their locations over time or the opposite. I took the following process. Firstly, sort the user collection based on the tag 'Geneeral_post_created_at_', so that the collection is represented based on timeline. Then I slice the entire database into 3 parts, each part represents a certain group of users who twite in this certain period. Finally, I count the number of users who leave their location as blank. Here follows screenshots of the process.



```
mydoc = mycol_general_users.find().sort("Geneeral_post_created_at_",-1)
temp1 = []
for z in range(0,50) :
    temp1.append(mydoc[z])
    print(mydoc[z])
```

```
eakerGuys_name': 'yangmi_pics', 'Geneeral_SneakerGuys_followers_count': 702, 'Geneeral_SneakerGuys_friends_count': 12, 'Geneeral_post_crea
ted_at': 'Mon Apr 06 11:12:49 +0000 2020', 'Geneeral_post_text': 'RT @yangmi_th: 200330 adidasOriginals อัพเดตแฟชั่น Superstar ครบรอบ 50 ปี
กับรองเท้า Superstar สุดคลาสสิกสุด\n\n👉https://t.co/yfK2vwFxJy\n#Yang…', 'Geneeral_post_hashtags': '#', 'Geneeral_post_mentions': '@yangmi
_th@', 'Geneeral_post_favorite_count': 0, 'Geneeral_post_retweet_count': 9}
{'_id': ObjectId('5e8e6ef3b2a8a2842d3ecaff'), 'Geneeral_SneakerGuys_id': 1093878956426547200, 'Geneeral_SneakerGuys_loc': '', 'Geneeral_Sn
eakerGuys_name': 'yangmi_pics', 'Geneeral_SneakerGuys_followers_count': 702, 'Geneeral_SneakerGuys_friends_count': 12, 'Geneeral_post_crea
ted_at': 'Mon Apr 06 11:12:02 +0000 2020', 'Geneeral_post_text': 'RT @yangmi_th: 200330 NOWHER玩美  อัพเดต #หยางมี่ กับแบรนด์ #adidasOrigina
ls ครบรอบ 50 ปีกับรองเท้า Superstar 😁\n\n👉https://t.co/yfK2vwFxJy\n#Ya…', 'Geneeral_post_hashtags': '#หยางมี่#adidasOriginals#', 'Geneer
al_post_mentions': '@yangmi_th@', 'Geneeral_post_favorite_count': 0, 'Geneeral_post_retweet_count': 5}
{'_id': ObjectId('5e8e6ef3b2a8a2842d3ecb00'), 'Geneeral_SneakerGuys_id': 1093878956426547200, 'Geneeral_SneakerGuys_loc': '', 'Geneeral_Sn
```

```
tempa = mydb["tempa"]
temp_insertA = tempa.insert_many(temp1)
```

```
cnt = tempa.count_documents({"Geneeral_SneakerGuys_loc": ""})
print(cnt)
```

```
18
```

```
deleteaha = tempa.delete_many({})
```

```
mydoc = mycol_general_users.find().sort("Geneeral_post_created_at_",-1)
temp2 = []
for z in range(51,100) :
    temp2.append(mydoc[z])
    print(mydoc[z])
```

```
'Sun Apr 05 20:13:00 +0000 2020', 'Geneeral_post_text': 'RT @missdemsxo: Need to start watermarking my sneaker pics 😂', 'Geneeral_post_h
ashtags': '#', 'Geneeral_post_mentions': '@missdemsxo@', 'Geneeral_post_favorite_count': 0, 'Geneeral_post_retweet_count': 7}
{'_id': ObjectId('5e8e6ef3b2a8a2842d3ecb07'), 'Geneeral_SneakerGuys_id': 1168590753599250432, 'Geneeral_SneakerGuys_loc': '', 'Geneeral_Sn
eakerGuys_name': 'cassbabesx', 'Geneeral_SneakerGuys_followers_count': 121, 'Geneeral_SneakerGuys_friends_count': 136, 'Geneeral_post_crea
ted_at': 'Sun Apr 05 20:10:14 +0000 2020', 'Geneeral_post_text': 'RT @missdemsxo: Need to start watermarking my sneaker pics 😂', 'Geneer
al_post_hashtags': '#', 'Geneeral_post_mentions': '@missdemsxo@', 'Geneeral_post_favorite_count': 0, 'Geneeral_post_retweet_count': 7}
```

```
tempb = mydb["tempb"]
temp_insertB = tempb.insert_many(temp2)
cnt = tempb.count_documents({"Geneeral_SneakerGuys_loc": ""})
print(cnt)
```

```
12
```

```
mydoc = mycol_general_users.find().sort("Geneeral_post_created_at_",-1)
temp3 = []
for z in range(101,150) :
    temp3.append(mydoc[z])
    print(mydoc[z])
```

```
s://t.co/Sjd36ofDg0 https://t.co/glISCZpn2x', 'Geneeral_post_hashtags': '#', 'Geneeral_post_mentions': '@', 'Geneeral_post_favorite_count
': 0, 'Geneeral_post_retweet_count': 0}
{'_id': ObjectId('5e8e6ef3b2a8a2842d3ecb3b'), 'Geneeral_SneakerGuys_id': 40262489, 'Geneeral_SneakerGuys_loc': 'Worldwide', 'Geneeral_Snea
kerGuys_name': 'FNW_WW', 'Geneeral_SneakerGuys_followers_count': 4449, 'Geneeral_SneakerGuys_friends_count': 319, 'Geneeral_post_created_a
t': 'Wed Apr 08 17:20:05 +0000 2020', 'Geneeral_post_text': 'Rise of Human invites its community to customize its signature sneaker http
s://t.co/zTowUPVAUS https://t.co/IE6NURM4Pb', 'Geneeral_post_hashtags': '#', 'Geneeral_post_mentions': '@', 'Geneeral_post_favorite_count
': 0, 'Geneeral_post_retweet_count': 0}
```

```
tempc = mydb["tempc"]
temp_insertC = tempb.insert_many(temp3)
cnt = tempb.count_documents({"Geneeral_SneakerGuys_loc": ""})
print(cnt)
```

```
27
```

From the result we could see that, the latest users who leave their location blank is 18, the second latest is 12 and the third latest is 27. So based on the data, we could see that at the beginning, people are generally not willing to indicate their locations, and then they are gradually willing to indicate their locations. But most recently, which might be caused by Covid-19, they are tending not to show their locations

4. Answer questions from the Assignment 3

(1) What are tags are associated with a person, place or thing?

For my Users_info collection, I basically tagged all the users to 'Jordan', 'Nike', and 'Addidas' to show their relationship to the specific categories they belong to.

(2) What social media users are like other social media users in your domain?

For this question, I picked two most active users and check the likeness of their comments' similarities with each of them. The most active user has 7 twittees, the highest similarity rate is 0.5 and the lowest similarity rate is about 0.269. The second most active user have 4 twittees , the highest similarity rate is 0.54 and the lowest is 0.29. We could see that the similarity rate of these two most active users are almost the same.

(3) What people, places or things are popular in your domain?

We sorted the user database based on Geneeral_SneakerGuys_followers_count and check the user who has most followers. The guy named 'ezr4reeves' has the most followers and he is the most popular one.

(4) What people, places or things are trending in your domain? (A trend is popularity over time.)

For this question, I would like to show the trend of whether users on twitter are more likely to indicate their locations over time or the opposite. I took the following process. First sort the user collection based on the tag 'Geneeral_post_created_at_' so that the collection is represented based on timeline. Then I slice the entire database into 3 parts, each part represents a certain group of users who twite in this certain period. Finally, I count the number of users who leave their location as blank.

From the data I collected, we could see that the latest users who leave their location blank is 18, the second latest is 12 and the third latest is 27. So based on the data, we could see that at the beginning, people are generally not willing to indicate their locations, and then they are gradually willing to indicate their locations. But most recently, which might be caused by Covid-19, they are tending not to show their locations.

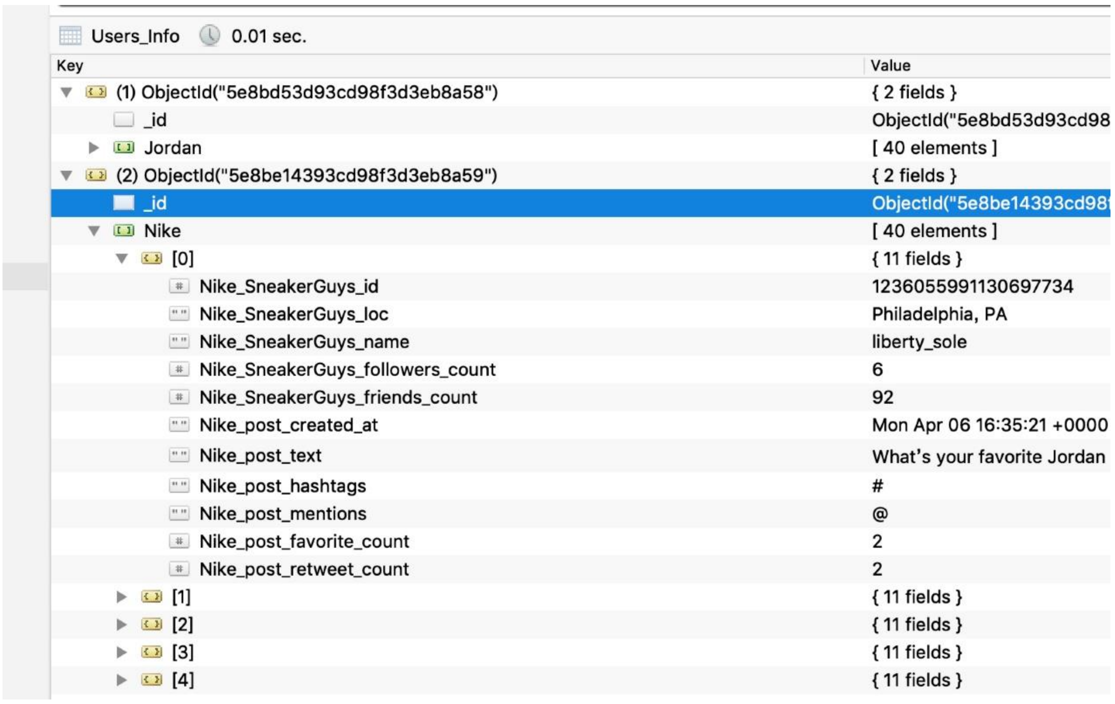
(5) General Process

(a) Convert the original MySQL database to NoSQL database.
(b) Scrape data from twitter API and added the new data two the database.
(c) Make queries on the database.
(d) Answer the questions

5.DESIGN CHOICES AND FINAL REPORT

When converting the SQL database to NoSQL database, there are two options for me. The first one is directly converting the 5 tables in the MySQL Sneaker database to 5 plain collections without nested documents inside. This strategy would flatten the data and makes the data look clearer, but it would also slow down the querying speed. For example, if the name of a certain field is provided and the querying process involves more than 1 collection. We would write more lines of code to get what we want, but if the information we are querying is inside another nested document, one simple “.”operation is good enough to query what we want. On the other hand, the nested documents would make the NoSQL database way more complicated. The database is quite small, so the speed won’t be that important. Thus, we choose the first strategy and directly convert the SQL database to NoSQL database without using nested documents.

When building the user NoSQL database, there are four main groups of users information I collected, namely 'Jordan', 'Nike', 'Adidas' and 'Sneakers'. for each group of users, I picked one or two users on twitter and collect their retwittee user information. Then I aggregated the first three group of users in new collection called 'User_Info'. This collection contains only three field, each of the key in this new collection is 'Jordan', 'Nike', 'Adidas'. The value is the group of users under this tag. The structure of this collection is shown in the image below. This collection shows the users who are specifically interested in a brand. In order to show the relationships of the collection 'User_Info' and the original tables, I added a foreign key to the 'Sneaker_Title' collection to link them together. Then I store the sneaker collection in a separate collection, which contains users who are interested in general sneakers. The below image shows the structure of embedded 'User_Info' collection.



```
Users_Info   0.01 sec.

Key                                                Value
▼ (1) ObjectId("5e8bd53d93cd98f3d3eb8a58")         { 2 fields }
      _id                                          ObjectId("5e8bd53d93cd98
    ▶ Jordan                                       [ 40 elements ]
▼ (2) ObjectId("5e8be14393cd98f3d3eb8a59")         { 2 fields }
      _id                                          ObjectId("5e8be14393cd98
    ▼ Nike                                         [ 40 elements ]
        ▼ [0]                                      { 11 fields }
              Nike_SneakerGuys_id                  1236055991130697734
              Nike_SneakerGuys_loc                 Philadelphia, PA
              Nike_SneakerGuys_name                liberty_sole
              Nike_SneakerGuys_followers_count     6
              Nike_SneakerGuys_friends_count       92
              Nike_post_created_at                 Mon Apr 06 16:35:21 +0000
              Nike_post_text                       What’s your favorite Jordan
              Nike_post_hashtags                   #
              Nike_post_mentions                   @
              Nike_post_favorite_count             2
              Nike_post_retweet_count              2
        ▶ [1]                                      { 11 fields }
        ▶ [2]                                      { 11 fields }
        ▶ [3]                                      { 11 fields }
        ▶ [4]                                      { 11 fields }
```

Conclusion

For this project, I first transferred the Sneaker MySQL database to NoSQL database, then I added two main collections of twitter user collection. One is based on their tags of different brand sneakers. The other one is a general sneaker information collection. The user information is from twitter api. After I built the mongodb database. I implemented bunches of different queries on the database, including finding all the documents of each database, finding documents with certain values in a collection, etc. When answering the last question of 'What people, places or things are trending in your domain? (A trend is popularity over time.) ', I added three more collections, each of which is a sub collection of the General user collection. The purpose of these three collections are counting the certain field of the user and show their trends. Finally, I answered the questions. For this assignment, I deeply dive into the mongodb database and found it much more convenient than the relational database for querying and retrieving data. It's flexibility, however, is not always a good point, since sometimes it is challenging to normalize the data, clean them and showing the relationship among all the collections. Thus, whether we should choose NoSQL database or Relational database should always be based on what we need.

Contribution

Contributed on my Own: 60% By referring to external source: 40%.

REFERENCES

Below is a part of websites where I gained knowledge of the project.

https://www.geeksforgeeks.org/python-ways-to-create-a-dictionary-of-lists/

https://api.mongodb.com/python/current/tutorial.html

https://blog.csdn.net/weixin_30532369/article/details/99893280

https://github.com/python/cpython/blob/3.8/Lib/difflib.py

https://dzone.com/articles/text-similarity-python-sklearn-on-mongodb-collecti

https://docs.mongodb.com/manual/tutorial/query-array-of-documents/

https://www.w3schools.com/python/python_mongodb_find.asp

https://stackoverflow.com/questions/19795012/how-to-convert-a-list-to-jsonarray-in-python

https://api.mongodb.com/python/current/tutorial.html#inserting-a-document

https://www.udemy.com/join/login-popup/?next=/course/mongodb-the-complete-developers-guide/learn/lecture/11739234#overview

https://www.educative.io/edpresso/how-to-convert-a-list-to-an-array-in-python

6.License