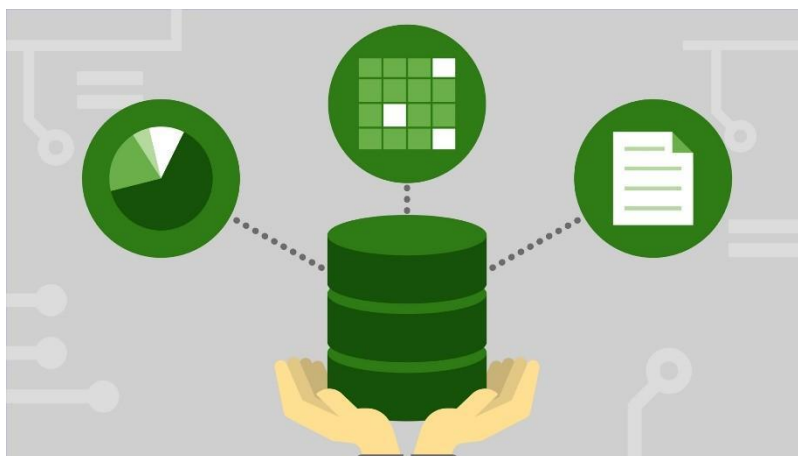


به نام خدا



دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



آزمایشگاه پایگاه داده

دستور کار شماره 6

حمیدرضا خدادادی

810197499

دی ماه ۱۴۰۰

در این گزارش کار می خواهیم با پایگاه داده مانگو آشنا بشویم.

برای این دستور کار، دیتابیس‌ی به نام sahamyab ایجاد می‌کنیم. در ادامه اغلب دستورات پیاده‌سازی

شده، در Robo 3T دوباره اجرا کردم و زمان اجرای آن ها را اندازه گرفتم.

● گام اول:

در ابتدای کار ده توپیت آخر سایت سهام یاب را با استفاده از کد زیر بدست می آوریم.

```
File Edit Selection View Go Run Terminal Help
6.py - DB LAB 5-8 - Visual Studio Code

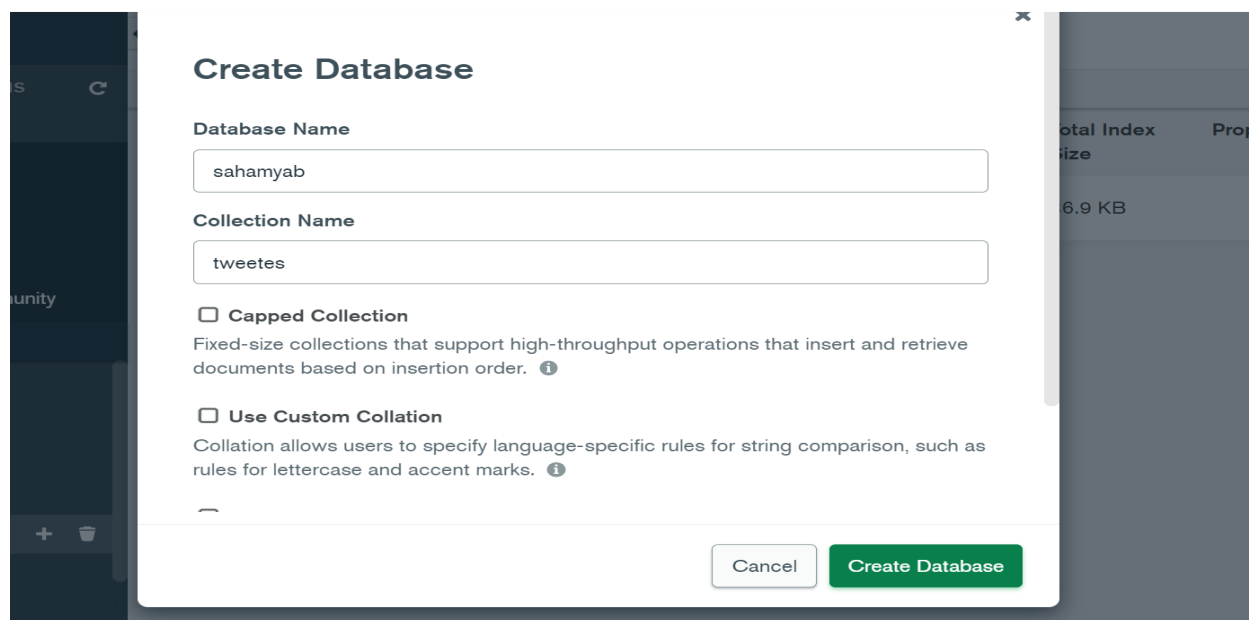
6.py X
6.py > ...
1 import requests
2 import time
3
4 url = "https://www.sahamyab.com/guest/twiter/list?v=0.1"
5 delay = 60
6 while True:
7     response = requests.request('GET', url, headers={'User-Agent': 'Chrome/61'})
8     if response.status_code == requests.codes.ok:
9         items = response.json()['items']
10        for item in items:
11            print(item)
12        time.sleep(60)
```

و خروجی کد بالا به صورت زیر خواهد بود.

[illegible]

حال یک دیتابیس به نام sahamyab و یک کالکشن در آن به نام tweets ایجاد می کنیم.

7017/sahamyab



Create Database

Database Name
sahamyab

Collection Name
tweets

☐ Capped Collection
Fixed-size collections that support high-throughput operations that insert and retrieve documents based on insertion order. ⓘ


☐ Use Custom Collation
Collation allows users to specify language-specific rules for string comparison, such as rules for lettercase and accent marks. ⓘ

Cancel Create Database

سپس به صورت دستی، این ده توییت بدست آمده را در دیتابیس و کالکشن tweets ذخیره می کنیم.

```
> db.tweets.insertOne({
  "id": "405639729",
  "sendTime": "2021-12-14T21:19:56Z",
  "sendTimePersian": "1400/09/24 00:49",
  "senderName": "ali mousavi",
  "senderUsername": "alimousavil357",
  "senderProfileImage": "default",
  "content": "نه . نه اینکه سودش کم بشه . بدانید که سهم کسرا بتانسیل ده برابر شدن داره . اما منافع آقایون تو این هست که سود به صورت\нкسرا#",
  "type": "twit",
  "scoredPostDate": "1639516845147",
  "finalPullDatePersian": ""
})
< { acknowledged: true,
  insertedId: ObjectId("61c9dbef4a65d20748860888") }
sahamyab>
```

در تصویر زیر مشاهده می شود که 10 توییت در کالکشن توییت ها در دیتابیس ذخیره شده است.

| CREATE COLLECTION | | | | | | |
|-------------------|-----------|--------------------|---------------------|--------------|------------------|---|
| Collection Name | Documents | Avg. Document Size | Total Document Size | Num. Indexes | Total Index Size | Properties |
| tweets | 10 | 1.3 KB | 13.3 KB | 1 | 20.5 KB |  |

در تصویر زیر نیز مشاهده می کنید که یک فیلد ObjectId به ازای هر توییت ایجاد شده است.

| tweets | | | | | | | | |
|--------|--------------------------|-------------|------------------------|------------------------|-------------------------|--|--|--|
| | _id ObjectId | id String | sendTime String | sendTimePersian String | senderName String | | | |
| 1 | 61c9dbef4a65d20748860888 | "405639729" | "2021-12-14T21:19:56Z" | "1400/09/24 00:49" | "ali mousavi" | | | |
| 2 | 61c9dc8ba9cec6e1e3140a88 | "405639718" | "2021-12-14T21:19:47Z" | "1400/09/24 00:49" | "aliya110" | | | |
| 3 | 61c9dc8ba9cec6e1e3140a89 | "405639713" | "2021-12-14T21:19:43Z" | "1400/09/24 00:49" | "elmoro" | | | |
| 4 | 61c9dc8ba9cec6e1e3140a8a | "405639704" | "2021-12-14T21:19:37Z" | "1400/09/24 00:49" | "fantom" | | | |
| 5 | 61c9dc8ba9cec6e1e3140a8b | "405639616" | "2021-12-14T21:17:22Z" | "1400/09/24 00:47" | "شیر بهشتی" | | | |
| 6 | 61c9dc8ba9cec6e1e3140a8c | "405639597" | "2021-12-14T21:16:51Z" | "1400/09/24 00:46" | "ممطلی" | | | |
| 7 | 61c9dc8ba9cec6e1e3140a8d | "405639534" | "2021-12-14T21:15:09Z" | "1400/09/24 00:45" | "مالباخته مظلوم" | | | |
| 8 | 61c9dc8ba9cec6e1e3140a8e | "405639527" | "2021-12-14T21:15:00Z" | "1400/09/24 00:45" | "Bitcoin Miner" | | | |
| 9 | 61c9dc8ba9cec6e1e3140a8f | "405639506" | "2021-12-14T21:14:33Z" | "1400/09/24 00:44" | "Decade 60" | | | |
| 10 | 61c9dc8ba9cec6e1e3140a90 | "405639489" | "2021-12-14T21:14:04Z" | "1400/09/24 00:44" | "https://t.me/BOURSE_I" | | | |

کد زیر تا زمانی که 500 تا توییت دریافت و در کالکشن قرار داده شود، هر یک دقیقه کد درون حلقه اجرا می شود.

```

1 import requests
2 import time
3 from pymongo import MongoClient
4
5 client = MongoClient()
6 db = client.sahamyab
7
8 url = "https://www.sahamyab.com/guest/twiter/list?v=0.1"
9 delay = 60
10
11 while db.tweets.count_documents({}) < 500:
12     response = requests.request('GET', url, headers={'User-Agent': 'Chrome/61'})
13     if response.status_code == requests.codes.ok:
14         items = response.json()['items']
15         for item in items:
16             db.tweets.insert_one(item)
17
18     print(db.tweets.count_documents({}))
19     time.sleep(delay)

```

خروجی اجرای کد بالا را در تصویر زیر می بینید.

```

5 client = MongoClient()
6 db = client.sahamyab
7
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS

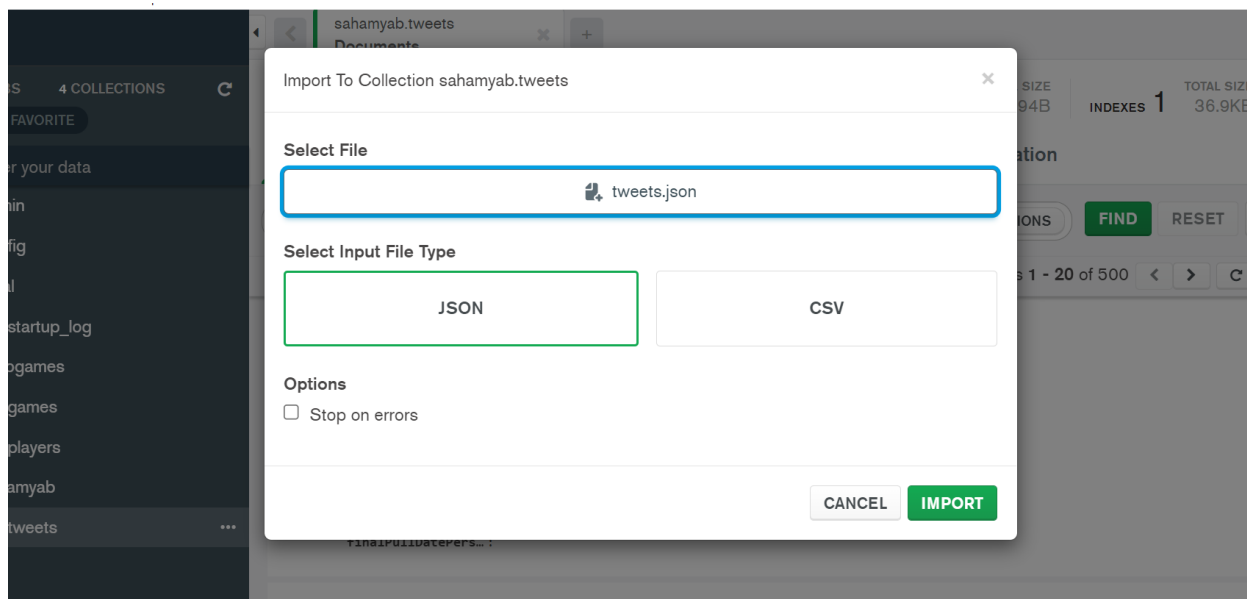
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

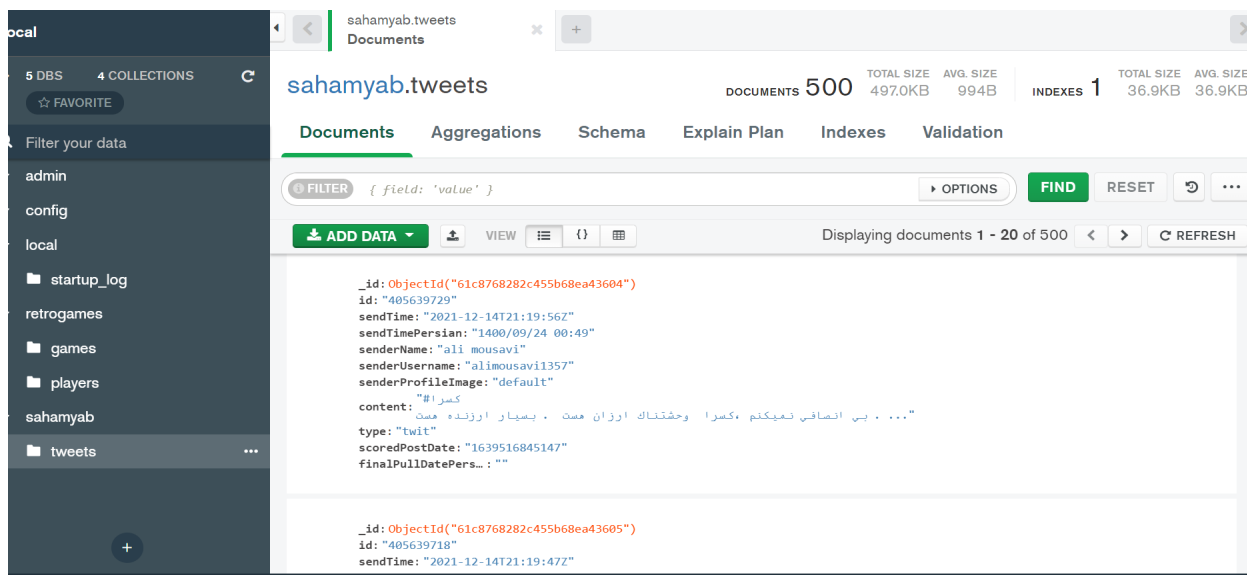
PS C:\Users\ASUS\Desktop\DB LAB 5-8> python -u "c:\Users\ASUS\Desktop\DB LAB 5-8\6.py"
307
318
329
340
351
362
373
384
395

```


به دلیل اینکه پیدا کردن 500 توییت یکتا کاری زمان بر بود، از فایل‌هایی که در گروه درس قرار داده شد استفاده کردم. در تصویر زیر نحوه اضافه کردن 500 توییت از فایل جیسون را مشاهده می‌کنید.



در تصویر زیر، بخشی از داکيومنت‌های این کالکشن را مشاهده می‌کنیم. که فیلد ObjectId را خود مونگو برای این داکيومنت‌ها ایجاد کرده است.



در دو تصویر زیر نیز از اینکه 500 توییت در دیتابیس قرار گرفته باشد، به دو صورت اطمینان حاصل می‌کنیم.

| CREATE COLLECTION | | | | | | |
|-------------------|-----------|--------------------|---------------------|--------------|------------------|---|
| Collection Name ^ | Documents | Avg. Document Size | Total Document Size | Num. Indexes | Total Index Size | Properties |
| tweets | 500 | 994.0 B | 497.0 KB | 1 | 36.9 KB |  |

```

> _MONGOSH
>
>
>
> use sahamyab
< 'switched to db sahamyab'
> db.tweets.estimatedDocumentCount()
< 500
sahamyab >

```

● کام دوم: پیش پردازش

```

6_PreProcess.py > ...
1  from pymongo import MongoClient
2  import time
3  import re
4
5
6  client = MongoClient()
7
8  db = client.sahamyab
9  db_collection = db.tweets
10
11  start_time = time.time()
12
13  for record in db_collection.find():
14      hashtags = list(re.findall(r"#(\w+)", record['content']))
15      filter = { 'id': record['id'] }
16      value = { "$set": { 'hashtags': hashtags } }
17      db_collection.update_one(filter, value)
18
19  end_time = time.time()
20
21  print("Time: ", end_time - start_time, " s")
22

```

در کد نوشته شده بالا که با استفاده از کتابخانه های pymongo و re برای ریجیکس، زده شده است، روی همه ی تویت های درون کالکشن دیتابیس می گردیم و فیلد content آن ها را بررسی کرده و اگر دارای کارکتر "#" بود، کلمه پس از آن را درون لیست هشتگ ها اضافه می کنیم. سپس کالکشن را آپدیت کرده و یک فیلد جدید hashtags با مقداری برابر لیست هشتگ ها ست می کنیم. برای این کار از متد update_one که فیلد filter آن را برابر شناسه یکتا تویت قرار می دهیم و فیلد value آن را طوری می نویسیم که لیست هشتگ ها به عنوان یک مقدار جدید به سند های کالکشن اضافه شود. برای چاپ کردن زمان نیز از کتابخانه time استفاده کردم.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ASUS\Desktop\DB LAB 5-8> python -u "c:\Users\ASUS\Desktop\DB LAB 5-8\6_PreProcess.py"
Time: 0.312086820602417 s
PS C:\Users\ASUS\Desktop\DB LAB 5-8>
```

در تصویر زیر مشاهده می کنید که فیلد hashtags برای سند ها اضافه شده است.

| | scoredPostDate String | finalPullDatePersian String | hashtags Array | imageUid String | mediaConten |
|---|-----------------------|-----------------------------|----------------|-----------------|--------------|
| 1 | "1639516845147" | "" | [] 1 elements | No field | No field |
| 2 | No field | "" | [] 0 elements | No field | No field |
| 3 | No field | "" | [] 0 elements | No field | No field |
| 4 | "1639516784307" | "" | [] 1 elements | No field | No field |
| 5 | "1639516677141" | "" | [] 1 elements | No field | No field |
| 6 | No field | "" | [] 0 elements | No field | "image/jpeg" |

در تصویر زیر نیز که از تویت شماره 452 گرفته شده است، لیست هشتگ های آن قابل مشاهده است.

```
>
_id: ObjectId("61c9e916a9cec6e1e3140c58")
id: "405780789"
sendTime: "2021-12-16T17:16:20Z"
sendTimePersian: "1400/09/25 20:46"
senderName: "ویلیام اونیل"
senderUsername: "ph.d.exchange"
senderProfileImage: "fd02819d-0d85-4500-8901-f19248a263f9"
content: "خودرو #خسایا #خگستر"
type: "twit"
scoredPostDate: "1639674924574"
finalPullDatePersian: ""
~ hashtags: Array
  0: "خودرو"
  1: "خسایا"
  2: "خگستر"
```

● گام سوم: دستورات اصلی

1. در این قسمت بدین صورت عمل کردیم که با استفاده از متد find، سند هایی که مقدار mediaContentType آن ها برابر image یا jpeg بود و مقدار parentId آن ها null نبود را پیدا کردیم. متد {\$exists: true} به ما در فیلتر کردن مقدار parentId های سند ها کمک کرد. سپس مشخص کردیم که برای سند هایی که پیدا کردیم، فیلد senderName آن ها نشان داده بشود و فیلد شناسه آن ها در نظر گرفته نشود. در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```
db.tweets.find({"mediaContentType": "image/jpeg", "parentId": {$exists: true}}, {"senderName": 1, "_id": 0})
```

```
> db.tweets.find({"mediaContentType": "image/jpeg", "parentId": {$exists: true}}, {"senderName": 1, "_id": 0})
< { senderName: 'مسطقی' }
  { senderName: 'ورودی 97 بورس' }
  { senderName: 'پیمان جانقریان' }
  { senderName: 'Mohsen' }
  { senderName: 'مهراب' }
  { senderName: 'm' }
  { senderName: 'محمد' }
  { senderName: 'داریوش' }
  { senderName: 'trader' }
  { senderName: 'AAAA' }
  { senderName: 'سینا مالکی' }
  { senderName: 'shahramdehqanii' }
  { senderName: 'حسابگر' }
```

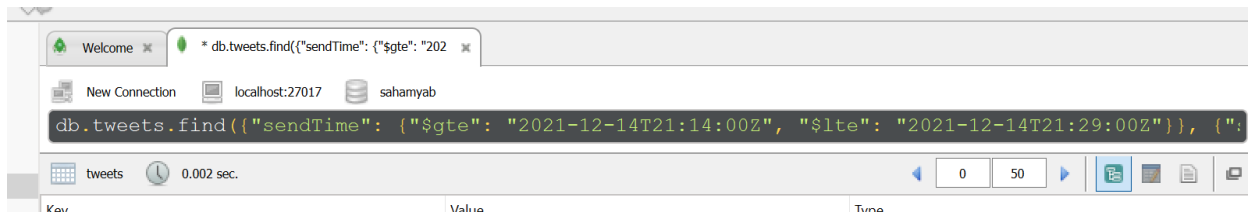


2. در این قسمت بدین صورت عمل کردیم که با استفاده از متد find، سند هایی که مقدار sendTime آن ها در بازه زمانی 15 دقیقه ای داده شد بود را پیدا کردیم. برای مقایسه استرینگ بازه اول و آخر زمان، از gte و lte استفاده کردم. سپس مشخص کردیم که برای

سند هایی که پیدا کردیم، فیلد senderUsername آن ها و فیلد type آن ها نشان داده بشود و فیلد شناسه آن ها در نظر گرفته نشود.
در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```
db.tweets.find({"sendTime": {"$gte": "2021-12-14T21:14:00Z", "$lte": "2021-12-14T21:29:00Z"}}, {"senderUsername": 1, "type": 1, "_id": 0})
```

```
db.tweets.find({"sendTime": {"$gte": "2021-12-14T21:14:00Z", "$lte": "2021-12-14T21:29:00Z"}}, {"senderUsername": 1, "type": 1, "_id": 0})
{ senderUsername: 'alimousavi1357', type: 'twit' }
{ senderUsername: 'aliya110' }
{ senderUsername: 'elmoro63' }
{ senderUsername: 'fantom65', type: 'twit' }
{ senderUsername: '22deniro', type: 'twit' }
{ senderUsername: 'mrmmostafaa' }
{ senderUsername: 'mohammumfghgfgg', type: 'twit' }
{ senderUsername: 'bitcoin_miner' }
{ senderUsername: 'leo.king' }
{ senderUsername: 'bourse_iran_', type: 'twit' }
sahamyab>
```



3. در این قسمت بدین صورت عمل کردیم که با استفاده از متد aggregate، یک پایپ لاین برای اجرای دستورات ایجاد کردیم. در این پایپ لاین، در ابتدا توییت هایی که sendTime آن ها در بازه زمانی داده شده قرار دارد را با match فیلتر می کنیم. سپس آن ها را بر اساس senderUsername شان با استفاده از group گروه بندی می کنیم. سپس تعداد توییت های هر گروه را با استفاده از sum می شماریم و در متغیر total ذخیره می کنیم. سپس با استفاده از match گروه هایی که تعداد توییت آن ها بیش از یک عدد است را پیدا کرده و نشان می دهیم.
در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```
db.tweets.aggregate([{"$match":
  {"sendTime": {"$gte": "2021-12-15T11:30:00Z", "$lte": "2021-12-15T12:30:00Z"}},
  {"$group":
    {"_id": "$senderUsername", "total": {"$sum": 1}, "senderProfileImage": {"$first": "$senderProfileImage"}},
  {"$match":
    {"total": {"$gt": 1}}]])
```

```
> db.tweets.aggregate([{"$match":
  {"sendTime": {"$gte": "2021-12-15T11:30:00Z", "$lte": "2021-12-15T12:30:00Z"}},
  {"$group":
    {"_id": "$senderUsername", "total": {"$sum": 1}, "senderProfileImage": {"$first": "$senderProfileImage"}},
  {"$match":
    {"total": {"$gt": 1}}]])
< { _id: 'ali.alizadeh', total: 3, senderProfileImage: 'default' }
sahamyab>
```



● گام چهارم: دستورات تجمعی و آماری

1. در این قسمت بدین صورت عمل کردیم که با استفاده از متد aggregate، یک پایپ لاین برای اجرای دستورات ایجاد کردیم. در این پایپ لاین، در ابتدا توییت ها را بر اساس senderUsername شان با استفاده از group گروه بندی می کنیم. سپس تعداد توییت های هر گروه را با استفاده از sum می شماریم و در متغیر total ذخیره می کنیم. سپس بار دیگر این افراد توییت زننده را با استفاده از group گروه بندی می کنیم.

برای گروه بندی از متد switch case استفاده کردم. بدین صورت که case اول بررسی می کند که تعداد توییت های فرد، برابر یک است یا خیر. و اگر برابر با یک بود، این فرد را جزو گروه OneTweet قرار می دهیم. و case دوم بررسی می کند که تعداد توییت های فرد، بزرگ تر از دو توییت و کوچک تر یا مساوی با سه توییت است یا خیر. و اگر برابر با این بازه بود، این فرد را جزو گروه TwoOrThreeTweet قرار می دهیم. و case سوم بررسی می کند که تعداد توییت های فرد، بزرگ تر از سه توییت است یا خیر. و اگر برابر با این بازه بود، این فرد را جزو گروه MoreThanThreeTweet قرار می دهیم. و حالت دیفالت این switch case نیز برابر با قرار گرفتن در گروه ZeroTweet است. در آخر نیز، تعداد افراد هر گروه را با استفاده از sum می شماریم و در متغیر numUsers ذخیره می کنیم.

در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```
db.tweets.aggregate([{"$group":
  {"_id": "$senderUsername", "total": {"$sum": 1}}},
  {"$group":
    {"_id":
      {"$switch":
        {"branches": [
          {"case": { "$eq": [ "$total", 1 ] },
            "then": "OneTweet" },
          {"case": { "$and": [ { "$gt" : [ "$total", 1 ] }, { "$lte" : [ "$total", 3 ] ] },
            "then": "TwoOrThreeTweet" },
          {"case": { "$gt": [ "$total", 3 ] },
            "then": "MoreThanThreeTweet" }
        ]
      }, "default": "ZeroTweet"
    }
  }}, {"$sum": {"numUsers": {"$sum": 1}}}]])
```

```
> db.tweets.aggregate([{"$group":
  {"_id": "$senderUsername", "total": {"$sum": 1}},
  {"$group":
    {"_id":
      {"$switch":
        {"branches": [
          { "case": { "$eq": [ "$total", 1 ] },
            "then": "OneTweet" },

          { "case": { "$and": [ { "$gt" : [ "$total", 1 ] }, { "$lte" : [ "$total", 3 ] } ] },
            "then": "TwoOrThreeTweet" },

          { "case": { "$gt": [ "$total", 3 ] },
            "then": "MoreThanThreeTweet" }

        ]}, "default": "ZeroTweet"

      }, "numUsers": {"$sum": 1}}]])
< { _id: 'MoreThanThreeTweet', numUsers: 10 }
  { _id: 'TwoOrThreeTweet', numUsers: 66 }
  { _id: 'OneTweet', numUsers: 307 }
sahamyab>
```



2. در این قسمت بدین صورت عمل کردیم که با استفاده از متد aggregate، یک پایپ لاین

برای اجرای دستورات ایجاد کردیم. در این پایپ لاین، در ابتدا لیست هشتگ ها را با

unwind جدا جدا کردیم تا هر هشتگ را بررسی کنیم. سپس هشتگ ها را با استفاده از

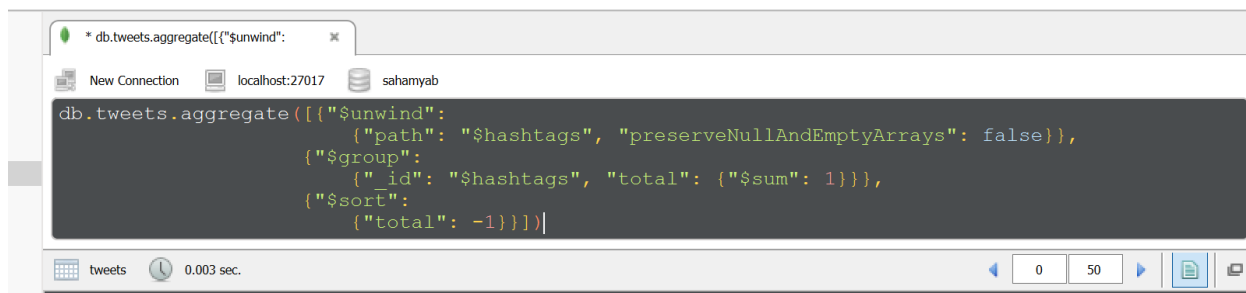
group گروه بندی می کنیم. سپس تعداد توییت های هر گروه را با استفاده از sum می

شماریم و در متغیر total ذخیره می کنیم. سپس با استفاده از متد sort(-1) این گروه ها را بر اساس تعداد توییت های هر گروه، نزولی سورت می کنیم.

در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```
4
5
6 db.tweets.aggregate([{"$unwind":
7   [{"path": "$hashtags", "preserveNullAndEmptyArrays": false}],
8   {"$group":
9     {"_id": "$hashtags", "total": {"$sum": 1}}},
10  {"$sort":
11    {"total": -1}}])
```

```
> db.tweets.aggregate([{"$unwind":
  {"path": "$hashtags", "preserveNullAndEmptyArrays": false}},
  {"$group":
    {"_id": "$hashtags", "total": {"$sum": 1}}},
  {"$sort":
    {"total": -1}}])
< { _id: 'شاحص بورس', total: 44 }
{ _id: 'خودرو', total: 24 }
{ _id: 'برکت', total: 19 }
{ _id: 'شپلی', total: 17 }
{ _id: 'شستا', total: 15 }
{ _id: 'کسرا', total: 13 }
{ _id: 'پالایش', total: 10 }
{ _id: 'تپکو', total: 8 }
{ _id: 'کدما', total: 7 }
{ _id: 'نژاهد', total: 6 }
{ _id: 'وتجارت', total: 6 }
{ _id: 'نوری', total: 6 }
```

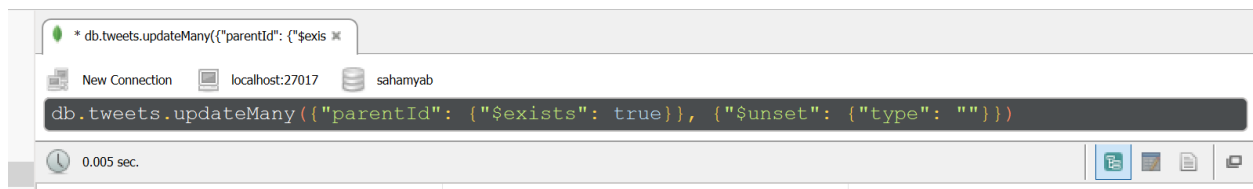


3. در اینجا با استفاده از متد `updateMany` همه توییت هایی که فیلتر شده اند، آپدیت می کنیم. باید برای توییت هایی که مقدار `parentId` آن ها `null` نبود، فیلد `type` آن ها را حذف کنیم. متد `{ $exists: true }` به ما در فیلتر کردن مقدار `parentId` های سند ها کمک کرد. و با متد `upset` فیلد `type` آن ها را خالی کردم.

در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```
db.tweets.updateMany({"parentId": {"$exists": true}}, {"$unset": {"type": ""}})
```

```
> db.tweets.updateMany({"parentId": {"$exists": true}}, {"$unset": {"type": ""}})
< { acknowledged: true,
  insertedId: null,
  matchedCount: 181,
  modifiedCount: 0,
  upsertedCount: 0 }
sahamyab >
```



مشاهده می کنید که این کوئری روی توییت ها به درستی اعمال شده است.

| | content String | type String | parentId String | parentSendTime String | scoredPostDate Stri |
|----|--------------------------------|-------------|-----------------|------------------------|---------------------|
| 1 | رزان هست . بسیار ارزنده هست# | "twit" | No field | No field | "1639516845147" |
| 2 | مند سرما مند می گذاشت می گفت | No field | "405619448" | "2021-12-14T19:12:05Z" | No field |
| 3 | ی تو پیج نوری چرت و پرت میگی | No field | "405567668" | "2021-12-14T08:04:42Z" | No field |
| 4 | . منفی میره به سمت صف خرید# | "twit" | No field | No field | "1639516784307" |
| 5 | ه به روند چندین چند سال پور# | "twit" | No field | No field | "1639516677141" |
| 6 | منبع: https://r.website/s/khar | No field | "405537410" | "2021-12-14T04:19:45Z" | No field |
| 7 | چندتا نظر درباره وضعیت واو# | "twit" | No field | No field | "1639516551969" |
| 8 | ندیم که آسمان را هم گرم کر . | No field | "405639171" | "2021-12-14T21:05:28Z" | No field |
| 9 | بودجه سال بعد تسعیر این رق# | No field | "405639202" | "2021-12-14T21:06:17Z" | No field |
| 10 | ر از تحلیل این سهم در کانال# | "twit" | No field | No field | "1639516514589" |
| 11 | نه شنبه معاملتش جذاب تر باشه | No field | "405588948" | "2021-12-14T11:11:22Z" | No field |

4. در این قسمت بدین صورت عمل کردیم که با استفاده از متد aggregate، یک پایپ لاین

برای اجرای دستورات ایجاد کردیم. در این پایپ لاین، در ابتدا لیست هشتگ ها را با unwind جدا جدا کردیم تا هر هشتگ را بررسی کنیم. سپس هشتگ ها را با استفاده از group گروه بندی می کنیم. سپس تعداد توییت های هر گروه را با استفاده از sum می شماریم و در متغیر total ذخیره می کنیم. سپس با استفاده از متد sort(-1) این گروه ها را بر اساس تعداد هشتگ های هر گروه، نزولی سورت می کنیم و با استفاده از limit(1) پرتکرار ترین هشتگ را نشان می دهیم. و یک بار دیگر نیز، با استفاده از متد sort(1) این گروه ها را بر اساس تعداد توییت های هر گروه، صعودی سورت می کنیم و با استفاده از limit(1) کم تکرار ترین هشتگ را نشان می دهیم.

در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```
db.tweets.aggregate([{"$unwind":
    {"path": "$hashtags", "preserveNullAndEmptyArrays": false}},
{"$group":
    {"_id": "$hashtags", "max": {"$sum": 1}}},
{"$sort":
    {"max": -1}},
{"$limit": 1}
])

db.tweets.aggregate([{"$unwind":
    {"path": "$hashtags", "preserveNullAndEmptyArrays": false}},
{"$group":
    {"_id": "$hashtags", "min": {"$sum": 1}}},
{"$sort":
    {"min": 1}},
{"$limit": 1}
])
```

```

< { _id: 'مدیریت', min: 1 }
> db.tweets.aggregate([{"$unwind":
    {"path": "$hashtags", "preserveNullAndEmptyArrays": false}},
    {"$group":
    {"_id": "$hashtags", "min": {"$sum": 1}}},
    {"$sort":
    {"min": 1}},
    {"$limit": 1}
  ])

```



```

> db.tweets.aggregate([{"$unwind":
    {"path": "$hashtags", "preserveNullAndEmptyArrays": false}},
    {"$group":
    {"_id": "$hashtags", "max": {"$sum": 1}}},
    {"$sort":
    {"max": -1}},
    {"$limit": 1}
  ])
< { _id: 'شخص_ورس', max: 44 }

```



5. در این قسمت بدین صورت عمل کردیم که با استفاده از متد aggregate، یک پایپ لاین برای اجرای دستورات ایجاد کردیم. در این پایپ لاین، در ابتدا تویت هایی که sendTime

آن ها در بازه زمانی داده شده قرار دارد را با match فیلتر می کنیم. لیست هشتگ ها را با unwind جدا جدا کردیم تا هر هشتگ را بررسی کنیم. سپس هشتگ ها را با استفاده از group گروه بندی می کنیم. سپس تعداد توییت های هر گروه را با استفاده از sum می شماریم و در متغیر count ذخیره می کنیم. سپس با استفاده از متد sort(-1) این گروه ها را بر اساس تعداد توییت های هر گروه، نزولی سورت می کنیم. با استفاده از limit(10) ده تا از هشتگ های پر تکرار را نشان می دهیم.

در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```

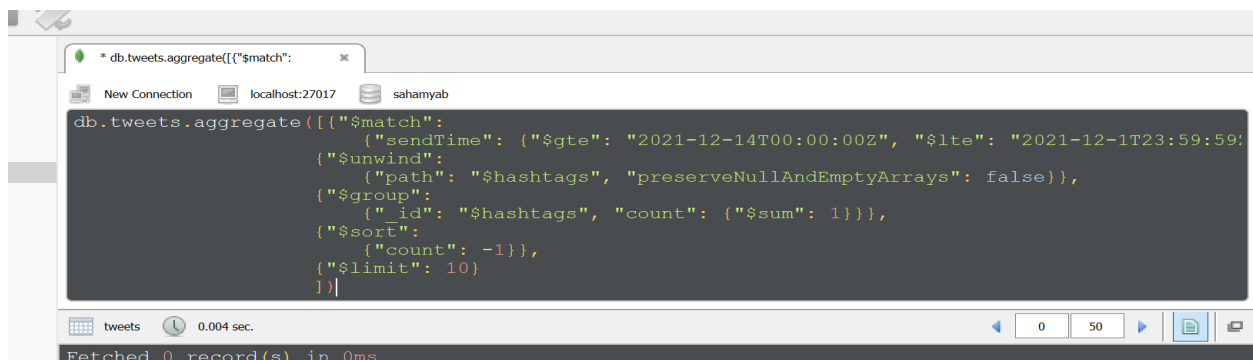
6
7 db.tweets.aggregate([{"$match":
8   {"sendTime": {"$gte": "2021-12-14T00:00:00Z", "$lte": "2021-12-14T23:59:59Z"}},
9   {"$unwind":
10    {"path": "$hashtags", "preserveNullAndEmptyArrays": false}},
11   {"$group":
12    {"_id": "$hashtags", "count": {"$sum": 1}}},
13   {"$sort":
14    {"count": -1}},
15   {"$limit": 10}
16 ]])

```

```

> db.tweets.aggregate([{"$match":
  {"sendTime": {"$gte": "2021-12-14T00:00:00Z", "$lte": "2021-12-14T23:59:59Z"}},
  {"$unwind":
    {"path": "$hashtags", "preserveNullAndEmptyArrays": false}},
  {"$group":
    {"_id": "$hashtags", "count": {"$sum": 1}}},
  {"$sort":
    {"count": -1}},
  {"$limit": 10}
])
< { _id: 'شستا', count: 2 }
  { _id: 'بوس', count: 1 }
  { _id: 'شاخص بوس', count: 1 }
  { _id: 'کسرا', count: 1 }
  { _id: 'خودرو', count: 1 }
  { _id: 'ولگرد', count: 1 }
sahamyab>

```



6. در این قسمت بدین صورت عمل کردیم که با استفاده از متد aggregate، یک پایپ لاین برای اجرای دستورات ایجاد کردیم. در این پایپ لاین، در ابتدا توییت هایی که sendTime آن ها در بازه زمانی داده شده قرار دارد را با match فیلتر می کنیم. سپس بر اساس senderUsername شان با استفاده از group گروه بندی می کنیم. سپس تعداد توییت های هر گروه را با استفاده از sum می شماریم و در متغیر count ذخیره می کنیم. و با استفاده از first و last دو متغیر دیگری که می خواهیم نشان دهیم را انتخاب می کنیم. با استفاده از متد sort(-1) این گروه ها را بر اساس تعداد توییت های هر گروه، نزولی سورت می کنیم و با استفاده از limit(1) کم تکرار ترین هشتگ را نشان می دهیم. در تصاویر زیر، خروجی ها و زمان اجرای این کوئری را مشاهده می کنید.

```
db.tweets.aggregate([{"$match":
  {"sendTime": {"$gte": "2021-12-16T00:00:00Z", "$lte": "2021-12-16T23:59:59Z"}},
  {"$group":
    {"_id": "$senderUsername", "count": {"$sum": 1}, "senderName": { "$first": "$senderName"},
    "senderuserName": { "$last": "$senderUsername"}},
  {"$sort":
    {"count": -1}},
  {"$limit": 2}
])
```

```
> db.tweets.aggregate([{"$match":
  {"sendTime": {"$gte": "2021-12-16T00:00:00Z", "$lte": "2021-12-16T23:59:59Z"}},
  {"$group":
    {"_id": "$senderUsername", "count": {"$sum": 1}, "senderName": { "$first": "$senderName"},
    "senderuserName": { "$last": "$senderUsername"}},
  {"$sort":
    {"count": -1}},
  {"$limit": 2}
])
< { _id: 'alidsh88',
  count: 8,
  senderName: 'علی',
  senderuserName: 'alidsh88' }
{ _id: 'maahr',
  count: 6,
  senderName: 'داریوش',
  senderuserName: 'maahr' }
sahamyab>
```

db.tweets.aggregate({ "\$match":

```
db.tweets.aggregate([{"$match":
  {"sendTime": {"$gte": "2021-12-16T00:00:00Z", "$lte": "2021-12-16T23:59:59Z"},
  {"$group":
    {"_id": "$senderUsername", "count": {"$sum": 1}, "senderName": { "$first":
      "senderusername": { "$last": "$senderUsername"}}},
  {"$sort":
    {"count": -1}},
  {"$limit": 1}
])
```

tweets 0.006 sec. 0 50