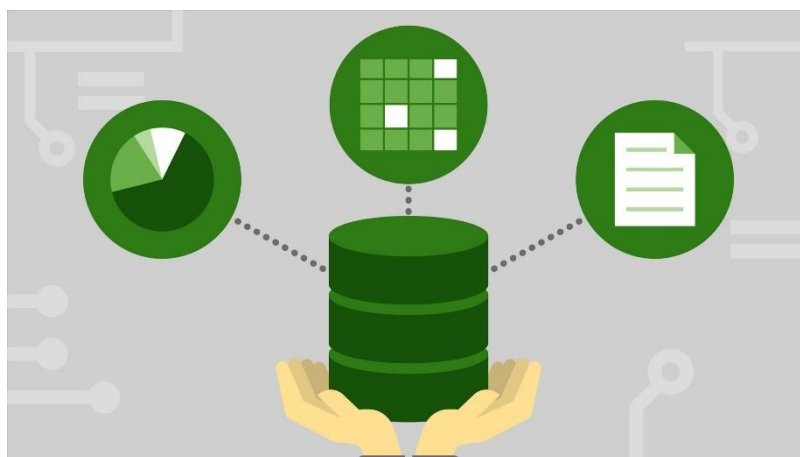


به نام خدا



دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



آزمایشگاه پایگاه داده

دستور کار شماره 6

نگار مرادی

810198543

مهر 1401

گزارش دستورکار انجام شده

پیش نیاز و شروع به کار با مانگو دی بی:

در این قسمت به مشاهده فیلم های آموزشی سایت پرداختیم. در زیر 6 نمونه کوئری را برای مثال آوردیم. در این قسمت سمپل دیتاست را ها اضافه میکنیم.

The screenshot shows the MongoDB Atlas 'Database Deployments' page for 'Project 0'. The main section displays 'Cluster0' with a status of 'All Good'. Key metrics include: Read (R) 0, Write (W) 0, Connections 0, In 0.0 B/s, Out 0.0 B/s, and Data Size 340.0 MB. A message at the top states 'Sample dataset successfully loaded. Access it in Data Explorer by clicking the Collections button, or with the MongoDB Shell.' The page also includes a table with details about the deployment, such as version (5.0.14), region (AWS / N. Virginia), cluster tier (M0 Sandbox), type (Replica Set - 3 nodes), backups (Inactive), linked app services (None Linked), and atlas search (Create Index).

The screenshot shows the MongoDB Atlas 'Collections' page for the 'sample_airbnb' database. It lists 22 collections, with 'listingsAndReviews' being the primary focus. The collection details show a storage size of 51.95MB, logical data size of 85.99MB, total documents of 5555, and index total size of 624KB. A filter query is displayed: `{ student_id: 654321, products: [{ type: 'exam', score: 98 }, { type: 'homework', score: 59 }] }`. The page also includes options to 'Visualize Your Data' and 'Refresh'.

به بررسی چند نمونه دستور find با استفاده از عملگرهای مقایسه میپردازیم:
از عملگر gt\$ برای تطبیق اسناد با فیلدی بزرگتر از مقدار داده شده استفاده میکنیم. در مثال زیر sale هایی که قیمتشان بیشتر از 50 باشد را پیدا میکنیم.

The screenshot shows the MongoDB Atlas web interface. The left sidebar contains navigation options like DEPLOYMENT, Database, SERVICES, and SECURITY. The main panel displays the 'sample_supplies.sales' collection with a filter applied: `{ "items.price": { "$gt": 50 } }`. The query results show two documents:

```

{
  "_id": ObjectId("5bd761dcae323e45a93ccfe9"),
  "saleDate": 2015-03-23T21:06:49.506+00:00,
  "items": Array,
  "storeLocation": "Denver",
  "customer": Object,
  "couponUsed": true,
  "purchaseMethod": "Online"
}

{
  "_id": ObjectId("5bd761dcae323e45a93ccfe9"),
  "saleDate": 2015-08-25T18:01:02.918+00:00,
  "items": Array
}

```

The interface indicates '1-20 of many results'.

در مثال زیر sale هایی که قیمتشان کوچک تر از 50 باشد را پیدا میکنیم/

The screenshot shows the MongoDB Atlas web interface. The left sidebar contains navigation options like DEPLOYMENT, Database, SERVICES, and SECURITY. The main panel displays the 'sample_supplies.sales' collection with a filter applied: `{ "items.price": { "$lt": 50 } }`. The query results show two documents:

```

{
  "_id": ObjectId("5bd761dcae323e45a93ccfe9"),
  "saleDate": 2015-03-23T21:06:49.506+00:00,
  "items": Array,
  "storeLocation": "Denver",
  "customer": Object,
  "couponUsed": true,
  "purchaseMethod": "Online"
}

{
  "_id": ObjectId("5bd761dcae323e45a93ccfe9"),
  "saleDate": 2015-08-25T18:01:02.918+00:00,
  "items": Array
}

```

The interface indicates '1-20 of many results'.

در مثال زیر مشتری هایی که سنشان کم تر از 65 است را پیدا میکنیم.

The screenshot shows the MongoDB Atlas web interface. The left sidebar contains navigation options: DEPLOYMENT, Database (selected), SERVICES, and SECURITY. The main panel displays the 'sample_supplies.sales' collection. A filter is applied: `{ "customer.age": { $lte: 65 } }`. The query results show two documents:

```

{
  "_id": ObjectId("5bd761dcae323e45a93ccfe9"),
  "saleDate": 2015-03-23T21:06:49.506+00:00,
  "items": Array,
  "storeLocation": "Denver",
  "customer": Object,
  "couponUsed": true,
  "purchaseMethod": "Online"
}

{
  "_id": ObjectId("5bd761dcae323e45a93ccfe9"),
  "saleDate": 2015-08-25T18:01:02.918+00:00,
  "items": Array
}

```

The interface also shows storage and index statistics: STORAGE SIZE: 996KB, LOGICAL DATA SIZE: 413MB, TOTAL DOCUMENTS: 5000, INDEXES TOTAL SIZE: 106KB.

در مثال زیر مشتری هایی که سنشان بیشتر از 65 است را پیدا میکنیم.

The screenshot shows the MongoDB Atlas web interface with the same 'sample_supplies.sales' collection. The filter is changed to: `{ "customer.age": { $gte: 65 } }`. The query results show two documents:

```

{
  "_id": ObjectId("5bd761dcae323e45a93ccfe9"),
  "saleDate": 2014-03-31T16:02:06.624+00:00,
  "items": Array,
  "storeLocation": "Austin",
  "customer": Object,
  "couponUsed": false,
  "purchaseMethod": "Online"
}

{
  "_id": ObjectId("5bd761dcae323e45a93cd004"),
  "saleDate": 2017-07-13T07:13:43.675+00:00,
  "items": Array
}

```

The interface also shows storage and index statistics: STORAGE SIZE: 996KB, LOGICAL DATA SIZE: 413MB, TOTAL DOCUMENTS: 5000, INDEXES TOTAL SIZE: 106KB.

یافتن داکيومنت ها با استفاده از عملگرهای منطقی.

برای انتخاب اسنادی که با چند عبارت مطابقت دارند از \$and ضمنی استفاده میکنیم.

در مثال زیر airline که اسمش برابر Southwest Airline است را پیدا میکنیم.

The screenshot shows the MongoDB Atlas web interface. On the left, the 'Database' sidebar is expanded, showing the 'sample_training' collection. The main panel displays the 'sample_training.routes' collection with a filter applied: `{ "airline.name": "Southwest Airlines", "stops": { "$gte": 1 } }`. The query results show two documents:

```

{ "_id": ObjectId("56e9b39c732b6122f878f280"),
  "airline": { "src_airport": "BOS", "dst_airport": "MCO", "codeshare": "", "stops": 1, "airplane": "73W" },
  "stops": 1,
  "airplane": "73W" }

{ "_id": ObjectId("56e9b39c732b6122f878f45b"),
  "airline": { "src_airport": "MCO", "dst_airport": "BOS", "codeshare": "", "stops": 1, "airplane": "73W" },
  "stops": 1,
  "airplane": "73W" }

```

از عملگر \$or برای انتخاب اسنادی استفاده میکنیم که حداقل با یکی از عبارات موجود مطابقت دارند.

در مثال زیر پرواز هایی را انتخاب میکنیم که یا مقصدشان SEA است و یا مبداشان SEA است.

The screenshot shows the MongoDB Atlas web interface. On the left, the 'Database' sidebar is expanded, showing the 'sample_training' collection. The main panel displays the 'sample_training.routes' collection with a filter applied: `{ "or": [{ "dst_airport": "SEA" }, { "src_airport": "SEA" }] }`. The query results show two documents:

```

{ "_id": ObjectId("56e9b39b732b6122f8788cfc"),
  "airline": { "src_airport": "BOS", "dst_airport": "SEA", "codeshare": "", "stops": 0, "airplane": "737" },
  "stops": 0,
  "airplane": "737" }

{ "_id": ObjectId("56e9b39b732b6122f8788d0f"),
  "airline": { "src_airport": "SEA", "dst_airport": "SEA", "codeshare": "", "stops": 0, "airplane": "737" },
  "stops": 0,
  "airplane": "737" }

```

در مثال زیر پرواز هایی را پیدا میکنیم که مقصدشان و یا مبداشان SEA است و یا اسمشان American Airline است و یا هواپیمایشان 320 است.

The screenshot shows the MongoDB Atlas interface. On the left, the 'sample_training.routes' collection is selected under the 'sample_training' database. The main panel displays a query filter: `{ $and: [{ $or: [{ dst_airport: "SEA" }, { src_airport: "SEA" }] }, { $or: [{ "airline.name": "American Airlines" }, { airplane: 320 }] }] }`. The query results show 1-20 of many results, with the first document displayed: `{ _id: ObjectId('56e9b39b732b6122f8788cfc'), airline: { src_airport: "BOS", dst_airport: "SEA", codeshares: "N", stops: 0 }, airplane: 737 }`.

در مثال زیر item هایی را پیدا میکنیم که اسم شهرشان جز شهرهای مشخصی باشد. در مثال زیر item هایی را پیدا میکنیم که شهرشان PHOENIX و یا CHICAGO باشد.

The screenshot displays the MongoDB Atlas web interface. The browser address bar shows the URL: `cloud.mongodb.com/v2/63ce90eeba9efa06aba80952#/metrics/replicaSet/63ce924981c522159ce8f4f4/explorer/sample_train...`. The Atlas logo and navigation links like 'Negar's Org', 'Access Manager', and 'Billing' are visible at the top. The left sidebar contains a 'DEPLOYMENT' section with 'Database' and 'Data Lake' options, and a 'SERVICES' section with various database services. The main content area shows the 'sample_training.zip' collection. It includes a search bar with the query `{ city: { $in: ["PHOENIX", "CHICAGO"] } }` and a filter button. Below the search bar, the 'QUERY RESULTS: 1-20 OF MANY' are displayed. The results show two documents: one with `city: "PHOENIX"` and `zip: "85012"`, and another with `city: "PHOENIX"` and `zip: "85004"`. The bottom of the interface shows the system status as 'All Good' and the copyright notice '©2023 MongoDB, Inc.'.

سپس به صورت دستی این 10 تویت را در tweets collection ذخیره میکنیم.

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	

تغییرات داده شده برای دریافت 500 تویت در کد زیر آورده شده ولی بدلیل گرفتن حجم و زمان زیاد دریافت این تویت ها از تویت های دوستانم استفاده کردم.

```
sahamyab.py > ...
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/twitter/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)
20
21
```

اضافه کردن این 400 تویت به tweets collection

The screenshot displays the MongoDB Atlas web interface. On the left, a sidebar contains navigation links for DEPLOYMENT, Database, SERVICES, and SECURITY. The main area is titled 'Cluster0' and shows the 'Collections' tab for the 'sahamyab' database. It lists the 'tweets' collection with the following statistics: STORAGE SIZE: 320KB, LOGICAL DATA SIZE: 480.8KB, TOTAL DOCUMENTS: 400, and INDEXES TOTAL SIZE: 28KB. Below these statistics, there are tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A 'QUERY RESULTS' section shows a sample document from the collection, including fields like _id, id, sendTime, senderName, senderUsername, senderProfileImage, content, type, scoredPostDate, and finalPullDatePersian.

گام دوم:

در این گام می‌خواهیم برای هر توییت هشتگ‌های آن را پیدا کنیم. در کد نوشته شده از کتابخانه re برای regex استفاده کردیم و با استفاده از کتابخانه pymongo بر روی همه ی توییت‌ها می‌گردیم و فیلد content آن‌ها را بررسی می‌کنیم. برای اینکار تمامی توییت‌ها را پیمایش کرده و اگر در فیلد content آن‌ها کاراکتر # وجود داشت. کلماتی که با # شروع شده باشند را به عنوان یک هشتگ در نظر می‌گیریم. و در فیلد hashtags به رکورد اضافه می‌کنیم.

در شکل زیر کد این قسمت به همراه زمان انجام فرایند آورده شده‌اند.

```

10
11 from pymongo import MongoClient
12 import time
13 import re
14
15
16 client = MongoClient()
17
18 db = client.sahamyab
19 db_collection = db.tweets
20
21 start_time = time.time()
22
23 for record in db_collection.find():
24     hashtags = list(re.findall("#(\w+)", record['content']))
25     filter = {'_id': record['_id']}
26     value = {'$set': {'hashtags': hashtags}}
27     db_collection.update_one(filter, value)
28
29 end_time = time.time()
30
31 print("Time: ", end_time - start_time, " s")
32

```

```

negar@negars-MacBook-Pro 6 % cd /Users/negar/Documents/UT/Term_7/DB\ LAB/6 : /usr/bin/env /opt/homebrew/bin/python3 /Users/negar/.vscode/extensions/ms-python.python-2022.20/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 64288 - /Users/negar/Documents/UT/Term_7/DB\ LAB/6/sahamyab.py
Time: 0.002787111896972656 s
negar@negars-MacBook-Pro 6 %

```

فیلد اضافه شده:

11

گام سوم:

1. کوئری را به صورت زیر مینویسیم. با استفاده از متود find سندهایی که مقدار mediaContentType آن ها برابر image و یا png بود و همچنین مقدار parentId برای آن ها وجود داشت را پیدا میکنیم و در آخر نام آن ها را نشان میدهیم.

```

55
56 from pymongo import MongoClient
57
58 client = MongoClient("mongodb+srv://negar:13792000@cluster0.on3imri.mongodb.net/?retryWrites=true&w=majority")
59 db = client.sahamyab
60
61 start_time = time.time()
62 output = db.new_tweets.find({"mediaContentType": "image/png", "parentId": {"$exists": True}}, {"senderName": 1})
63
64 end_time = time.time()
65
66 for item in output:
67     print(item["_id"], item["senderName"])
68
69 print(end_time - start_time)
70
71
72
73
74

```

PROBLEMS (13) OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY

negar@negars-MacBook-Pro 6 % cd /Users/negar/Documents/UT/Term_7/DB\ LAB/6 ; /usr/bin/env /opt/homebrew/bin/python3 /Users/negar/.vscode/extensions/ms-python.python-2022.20.2/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 53663 -- /Users/negar/Documents/UT/Term_7/DB\ LAB/6/sahamyab.py

negar@negars-MacBook-Pro 6 % cd /Users/negar/Documents/UT/Term_7/DB\ LAB/6 ; /usr/bin/env /opt/homebrew/bin/python3 /Users/negar/.vscode/extensions/ms-python.python-2022.20.2/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 53681 -- /Users/negar/Documents/UT/Term_7/DB\ LAB/6/sahamyab.py

63b68e7d19b1751141359e9e pc م2
63b5fed97cd69033b96c3c94 Edd1
63ba88af7aca47a08c0f9d42
0.0002989768981933594
negar@negars-MacBook-Pro 6 %

در کد بالا نمونه کد به همراه زمان فرایند و خروجی آورده شده است.

2. برای این کوئری همه ی تویت هایی که در تاریخ 2023/01/04 از ساعت 23:14 تا 23:29 فرستاده شده اند و فیلد content آن ها خالی نیست را با استفاده از find و استفاده از دستور های gte, lte پیدا میکنیم. id, content, sendTime آن ها را نمایش میدهیم. به همراه زمان اجرای فرایند نشان میدهیم.

```

47
48 from pymongo import MongoClient
49
50
51 client = MongoClient(
52     "mongodb+srv://negar:13792000@cluster0.on3imri.mongodb.net/?retryWrites=true&w=majority")
53 db = client.sahamyab
54
55 start_time = time.time()
56 output = db.tweets.find({"sendTime": {"$gte": "2023-01-04T23:14:00Z", "$lte": "2023-01-04T23:29:00Z"}}, {"sendTime": 1, "content": 1})
57
58 end_time = time.time()
59 for item in output:
60     print(item["_id"], item["sendTime"], item["content"])
61
62 print(end_time - start_time)
63
64
65

```

PROBLEMS (12) OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY

negar@negars-MacBook-Pro 6 % cd /Users/negar/Documents/UT/Term_7/DB\ LAB/6 ; /usr/bin/env /opt/homebrew/bin/python3 /Users/negar/.vscode/extensions/ms-python.python-2022.20.2/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 51177 -- /Users/negar/Documents/UT/Term_7/DB\ LAB/6/sahamyab.py

63b68e247cd69033b96c3d14 2023-01-04T23:16:19Z اکرم#
63b68e247cd69033b96c3d14 2023-01-04T23:19:05Z اکرم#
0.00021791458129882812
negar@negars-MacBook-Pro 6 %

3. در این قسمت با استفاده از aggregate یک پایپ لاین ایجاد میکنیم. سپس با استفاده از match توییت های بازه مورد نظر را پیدا میکنیم و بر اساس senderUserName اشان با group گروه بندی میکنیم. تعداد توییت هر گروه را با sum می‌شماریم و در total ذخیره میکنیم در آخر گروه هایی که تعداد توییتشان بیش از 1 باشد نشان میدهیم.

```

92
93
94
95 from pymongo import MongoClient
96
97 client = MongoClient(
98     "mongodb+srv://negar:13792000@cluster0.on3imri.mongodb.net/?retryWrites=true&w=majority"
99 )
100 db = client.sahamyab
101
102 start_time = time.time()
103 output = db.tweets.aggregate([{"$match":
104     {"sendTime": {"$gte": "2023-01-04T23:00:00Z", "$lte": "2023-01-04T23:59:00Z"}},
105     {"$group":
106         {"_id": "$senderUserName", "total": {"$sum": 1}, "senderProfileImage": {"$first": "$senderProfileImage"}},
107     {"$match":
108         {"total": {"$gt": 1}}]])
109
110 end_time = time.time()
111
112 for item in output:
113     print(item)

```

```

negar@negars-MacBook-Pro 6 % cd /Users/negar/Documents/UT/Term_7/DB\ LAB/6 ; /usr/bin/env /opt/homebrew/bin/python3 /Users/negar/.vscode/extensions/ms-python.python-2022.20.2/pythonFiles/Lib/python/debugpy/adapter/.../debugpy/launcher 54820 -- /Users/negar/Documents/UT/Term_7/DB\ LAB/6/sahamyab.py
{'_id': 'hosseinfatapour', 'total': 2, 'senderProfileImage': '6c88ddab-0e8b-4709-80f9-4ffcc6892887'}
{'_id': 'ichikoku1972', 'total': 3, 'senderProfileImage': '41fe1327-801f-4445-a177-e45d829386c3'}
2.504438070019227
negar@negars-MacBook-Pro 6 %

```

گام چهارم:

1.

```

96
97 output = db.tweets.aggregate([{"$group":
98   {"_id": "$senderUsername", "total": {"$sum": 1}},
99   {"$group":
100    {"_id":
101     {"$switch":
102      {"branches": [
103       {"case": {"$eq": {"$total", 1 }},
104        "then": "OneTweet" },
105       {"case": {"$and": [ {"$gt" : {"$total", 1 }}, {"$lte" : {"$total", 3 } } ] },
106        "then": "TwoOrThreeTweet" },
107       {"case": {"$gt": {"$total", 3 } },
108        "then": "MoreThanThreeTweet" }
109      ] , "default": "ZeroTweet"
110     }, "numUsers": {"$sum": 1}}]])
111
112
113
114
115
116 end_time = time.time()
117
118 for item in output:
119     print(item)
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

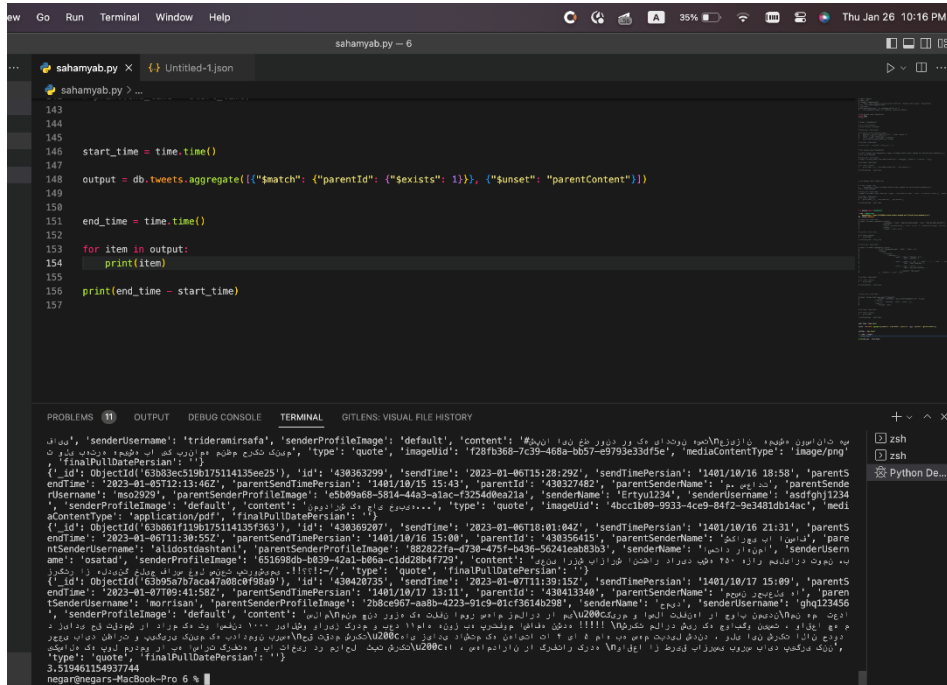
```

در این قسمت با استفاده از aggregate یک پایپ لاین ایجاد میکنیم. سپس با استفاده از match توییت های بازه مورد نظر را پیدا میکنیم و بر اساس senderUserName اشان با group گروه بندی میکنیم. تعداد توییت هر گروه را با sum میشماریم و در total ذخیره میکنیم. حال بر اساس تعداد توییت با استفاده از switch case گروه بندی میکنیم. تعداد هرگروه را میشماریم و در numusers ذخیره میکنیم و در آخر آن را نشان میدهیم. در کد بالا پیاده سازی و زمان اجرا آمده است.

[illegible]

15

3. در این قسمت با استفاده از aggregate یک پایپ لاین ایجاد میکنیم. سپس با استفاده از match تویت هایی که parentId دارند را پیدا میکنیم و با استفاده از unset فیلد parentContent آن ها را حذف میکنیم.



```

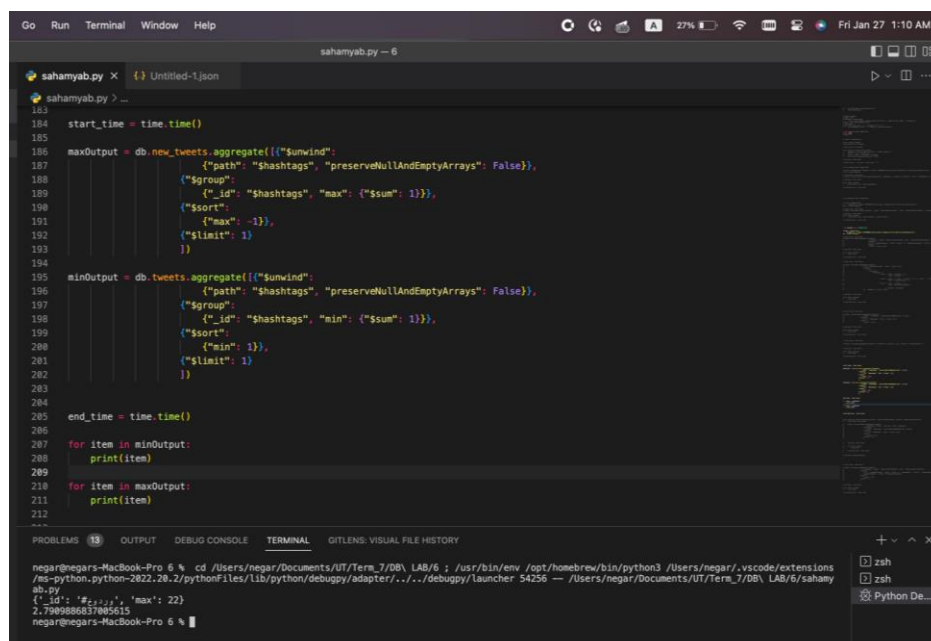
143
144
145
146 start_time = time.time()
147
148 output = db.tweets.aggregate([{"$match": {"parentId": {"$exists": 1}}, {"$unset": "parentContent"}}])
149
150
151 end_time = time.time()
152
153 for item in output:
154     print(item)
155
156 print(end_time - start_time)
157

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY

negar@negars-MacBook-Pro 6 %

4. در این قسمت با استفاده از aggregate یک پایپ لاین ایجاد میکنیم. سپس با استفاده از unwind هشتک ها را جدا میکنیم. برای هر دسته تویت جمع آن ها را در total ذخیره میکنیم و نزولی مرتب میکنیم. با لیمیت 1 بیشترین هشتک استفاده شده را به دست می آوریم برای کم ترین هم همین کار را میکنیم فقط به صورت صعودی مرتب میکنیم.



```

184 start_time = time.time()
185
186 maxOutput = db.new_tweets.aggregate([{"$unwind":
187     {"path": "$hashtags", "preserveNullAndEmptyArrays": False},
188     {"$group":
189         {"_id": "$hashtags", "max": {"$sum": 1}},
190         {"$sort":
191             {"max": -1}},
192         {"$limit": 1}
193     ]})
194
195 minOutput = db.tweets.aggregate([{"$unwind":
196     {"path": "$hashtags", "preserveNullAndEmptyArrays": False},
197     {"$group":
198         {"_id": "$hashtags", "min": {"$sum": 1}},
199         {"$sort":
200             {"min": 1}},
201         {"$limit": 1}
202     ]})
203
204
205 end_time = time.time()
206
207 for item in minOutput:
208     print(item)
209
210 for item in maxOutput:
211     print(item)
212

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY

negar@negars-MacBook-Pro 6 %

5. در این قسمت مانند قسمت های قبل از aggregate استفاده میکنیم سپس بر اساس بازه مورد نظر فیلتر کرده گروه بندی کرده و با sum تعداد تویت های هرگروه را حساب میکنیم و نزولی مرتب کرده و سپس 10 تای اول را با استفاده از لیمیت 10 نشان میدهم.

```

sahamyab.py
def find_most_usable_hashtags(startTime = "2022-12-01T00:00:00Z", endTime = "2023-01-01T23:59:59Z"):
    start_time = time.time()

    output = db.new_tweets.aggregate([{"$match":
        {"sendTime": {"$gte": start_time, "$lte": endTime}},
        {"$wind":
            {"path": "$hashtags", "preserveNullAndEmptyArrays": False}},
        {"$group":
            {"_id": "$hashtags", "count": {"$sum": 1}},
            {"$sort":
                {"count": -1}},
            {"$limit": 10}
        ])

    end_time = time.time()
    for item in output:
        print(item)
    print(end_time - start_time)

```

```

neqar@neqars-MacBook-Pro 6 % cd /Users/neqar/Documents/UT/Term_7/DB\ LAB/6 : /usr/bin/env /opt/homebrew/bin/python3 /Users/neqar/.vscode/extensions
/ns-python.python-2022.20.2/pythonFiles/Lib/python/debugpy/adapter/.../debugpy/launcher 51755 -- /Users/neqar/Documents/UT/Term_7/DB\ LAB/6/sahamy
ab.py
{"_id": "#درد", "count": 22}
{"_id": "#دردمن", "count": 14}
{"_id": "#دردمن", "count": 11}
{"_id": "#درد", "count": 9}
{"_id": "#دردمن", "count": 7}
{"_id": "#دردمن", "count": 7}
{"_id": "#دردمن", "count": 7}
{"_id": "#دردمن", "count": 7}
{"_id": "#دردمن", "count": 7}
{"_id": "#دردمن", "count": 7}
{"_id": "#دردمن", "count": 6}
2.5335322474121
neqar@neqars-MacBook-Pro 6 %

```

1. در این قسمت مانند قسمت های قبل از aggregate استفاده میکنیم سپس بر اساس بازه مورد نظر فیلتر کرده بر اساس senderUsername گروه بندی کرده و با sum تعداد تویت های هرگروه را حساب میکنیم و نزولی مرتب کرده و سپس اولی را که کاربری است که بیشترین تعداد تویت را زده است نشان میدهم.

```

sahamyab.py
start_time = time.time()

output = db.tweets.aggregate([{"$match":
    {"sendTime": {"$gte": "2023-01-04T00:00:00Z", "$lte": "2023-01-04T23:59:59Z"}},
    {"$group":
        {"_id": "senderUsername", "count": {"$sum": 1}, "senderName": {"$first": "senderName"},
        "senderUserName": {"$last": "senderUserName"}},
    {"$sort":
        {"count": -1}},
    {"$limit": 1}
    ])

end_time = time.time()
for item in output:
    print(item)
print(end_time - start_time)

```

```

neqar@neqars-MacBook-Pro 6 % cd /Users/neqar/Documents/UT/Term_7/DB\ LAB/6 : /usr/bin/env /opt/homebrew/bin/python3 /Users/neqar/.vscode/extensions
/ns-python.python-2022.20.2/pythonFiles/Lib/python/debugpy/adapter/.../debugpy/launcher 51924 -- /Users/neqar/Documents/UT/Term_7/DB\ LAB/6/sahamy
ab.py
{"_id": "behmac", "count": 3, "senderName": "امانوب", "senderUserName": "behmac"}
2.655074119567071
neqar@neqars-MacBook-Pro 6 %

```

آنچه آموختم / پیشنهادات

در این دستور کار کار با mongoDb را آموختیم که بنظر من بسیار جالب بود و در آینده حتما به کارمان می آید. فقط برای راه اندازی mongoDb و کار کردن با شل و compass و کانکت شدن به port درست به مشکل برخورددم که بعد از 2 روز تونستم حلش کنم.