

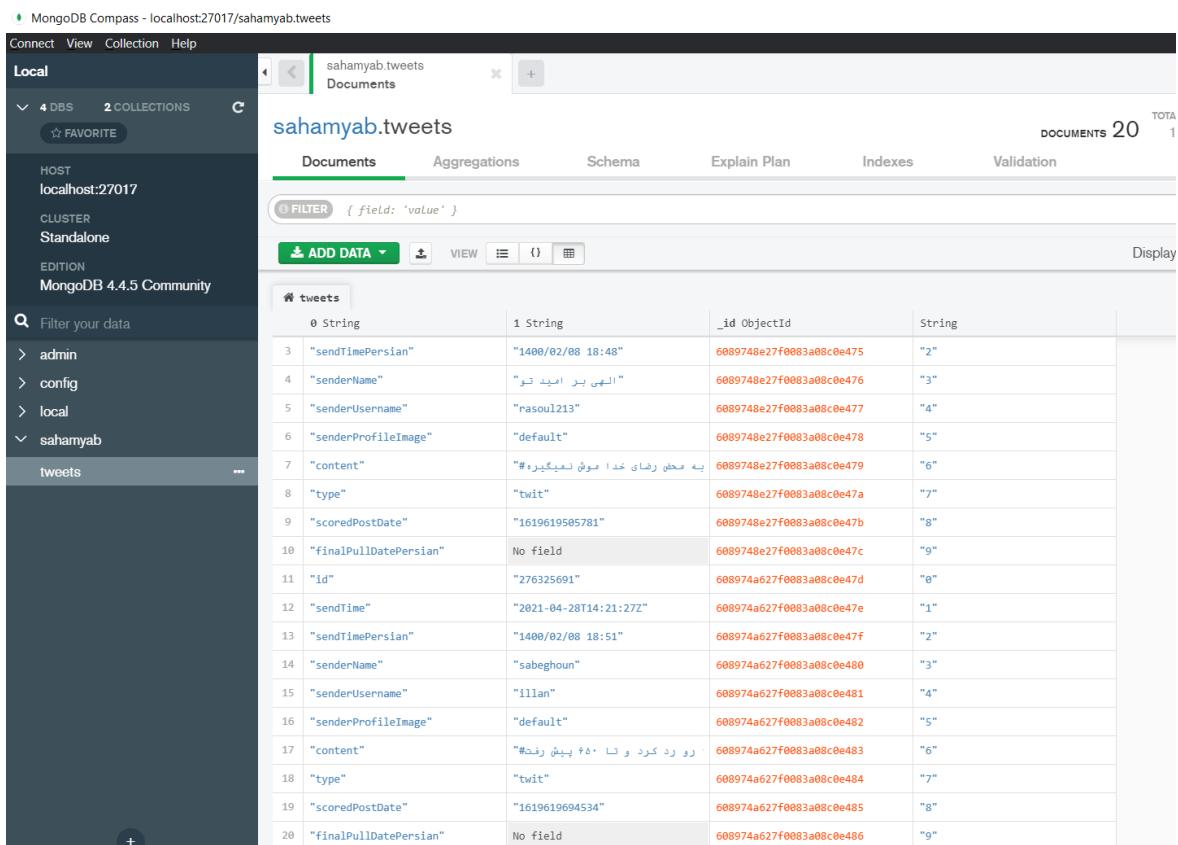


## بخش اول

### گام اول

#### نحوه ورود دستی داده‌ها در مانگو و فایل‌های اضافه شده:

ابتدا به کمک کد پایتون صورت سوال، تعداد ۱۰ توثیت دانلود می‌شود و سپس آن‌ها را با فرمت csv. در هارد ذخیره می‌کنیم. سپس به کمک mongo compass داده‌ها را تک‌تک با فرمت csv. وارد می‌کنیم.



The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays the connection details: Local, HOST localhost:27017, CLUSTER Standalone, and EDITION MongoDB 4.4.5 Community. Under the 'sahamyab' database, the 'tweets' collection is selected. The main area shows the 'Documents' tab with 20 documents listed. Each document has fields like \_id, ObjectId, content, type, and various Persian text fields. The table structure includes columns for \_id, ObjectId, content, type, and several Persian fields.

_id	ObjectId	content	type
3	6089748e27f0083a08c0e475	"۱۴۰۰/۰۲/۰۸ ۱۸:۴۸"	"twit"
4	6089748e27f0083a08c0e476	"الله يرحمه"	"twit"
5	6089748e27f0083a08c0e477	"rasoul213"	"twit"
6	6089748e27f0083a08c0e478	"default"	"twit"
7	6089748e27f0083a08c0e479	"به محظوظ شد! موش نمیگیره"	"twit"
8	6089748e27f0083a08c0e47a	"#"	"twit"
9	6089748e27f0083a08c0e47b	"1619619585781"	"twit"
10	6089748e27f0083a08c0e47c	"finalPullDatePersian"	"twit"
11	6089748e27f0083a08c0e47d	"276325691"	"twit"
12	6089748e27f0083a08c0e47e	"2021-04-28T14:21:27Z"	"twit"
13	6089748e27f0083a08c0e47f	"1400/۰۲/۰۸ ۱۸:۵۱"	"twit"
14	6089748e27f0083a08c0e480	"sabeghoun"	"twit"
15	6089748e27f0083a08c0e481	"illan"	"twit"
16	6089748e27f0083a08c0e482	"default"	"twit"
17	6089748e27f0083a08c0e483	"روز کرد و تا ۴۵ دقیقه رفت"	"twit"
18	6089748e27f0083a08c0e484	"#"	"twit"
19	6089748e27f0083a08c0e485	"1619619694534"	"twit"
20	6089748e27f0083a08c0e486	"finalPullDatePersian"	"twit"

همان‌طور که می‌بینیم، فیلد ObjectId به داده‌ها توسط mongodb اضافه شده است.

#### درج هزار توثیت:

در این قسمت ابتدا از طریق pymongo به کالکشن tweets در mongodb وصل می‌شویم و سپس با استفاده از request داده‌ها را دانلود کرده و در mongodb اضافه می‌کنیم. جهت اطمینان از درج هزار توثیت از دستور series\_collection.count() استفاده

می‌کنیم که به ما تعداد توئیت‌های موجود در کالکشن tweets را می‌دهد. زمان صرف شده: 17478.95367026329 ثانیه

```
1 import pandas as pd
2 import requests
3 import json
4 from pymongo import MongoClient
5 import time
6
7 client = MongoClient('localhost', 27017)
8 db = client['sahamyab']
9 series_collection = db['tweets']
10
11 start_time = time.time()
12 while series_collection.count() < 1000:
13     # Getting Last 10 twiits
14     response = requests.get('https://www.sahamyab.com/guest/twiter/list?v=0.1',
15     data = json.loads(response.text)
16     twiit_10 = data['items']
17     # insert data into mongodb
18     ### cheching if twitt is not in collection
19     for element in twiit_10 :
20         if series_collection.find_one(element) == None:
21             result = series_collection.insert_one(element)
22
23     time.sleep(60 - time.time() % 60)
24
25 end_time = time.time()
26 delta_time = end_time - start_time
27
28#print(series_collection.count())
29
30
```

## گام دوم

در ابتداء سعی کردم به کمک pymongo این قسمت را پیاده‌سازی کنم. اما در قسمت regexFindAll\$ مشکل داشتم و هشتگ‌هایی که در ابتدای جمله قرار داشتند را نمی‌توانستم به عنوان هشتگ استخراج کنم. به همین دلیل با توجه گفته‌ی جناب بنایی، در این قسمت از امکانات پایتون استفاده کردم.

عکس زیر کد پیاده‌سازی به کمک pymongo است.

```
20 ##### Implementing in pymongo #####
21 db.tweets.update_many(
22     {'content': {'$regex': '.*#.*'}},
23     [ {'$set': {'hashtags': {'$regexFindAll': { 'input': '$content' , 'regex': '#.*'}}}}
24   ])
25
26
27
```

کد نوشته شده در پایتون برای این بخش:  
زمان اجرا: 0.7917428016662598 ثانیه

```

1 from pymongo import MongoClient
2 import time
3 import re
4
5 client = MongoClient('localhost', 27017)
6 db = client['sahamyab']
7 series_collection = db['tweets']
8
9 start_time = time.time()
10 for document in series_collection.find({'content': {'$regex': '.*#.*'}}):
11     hashtags_sharp = re.findall('#\w+', document['content'])
12     hashtags = [o.split('#')[1] for o in hashtags_sharp] #removing shashtag si
13     new_dict = {"hashtags": hashtags}
14     db.tweets.update(document, {"$set": new_dict})
15
16 end_time = time.time()
17 delta_time = end_time - start_time
18
19

```

پس از اجرای کد فوق، در mongo compass فیلد هشتگ اضافه میشود. عکس زیر، فیلد اضافه شده به توانیت‌های شامل هشتگ را نشان می‌دهد.

The screenshot shows the MongoDB Compass interface connected to the 'sahamyab' database and its 'tweets' collection. The interface includes a sidebar for navigating databases and collections, and a main area for viewing documents. Two documents are listed:

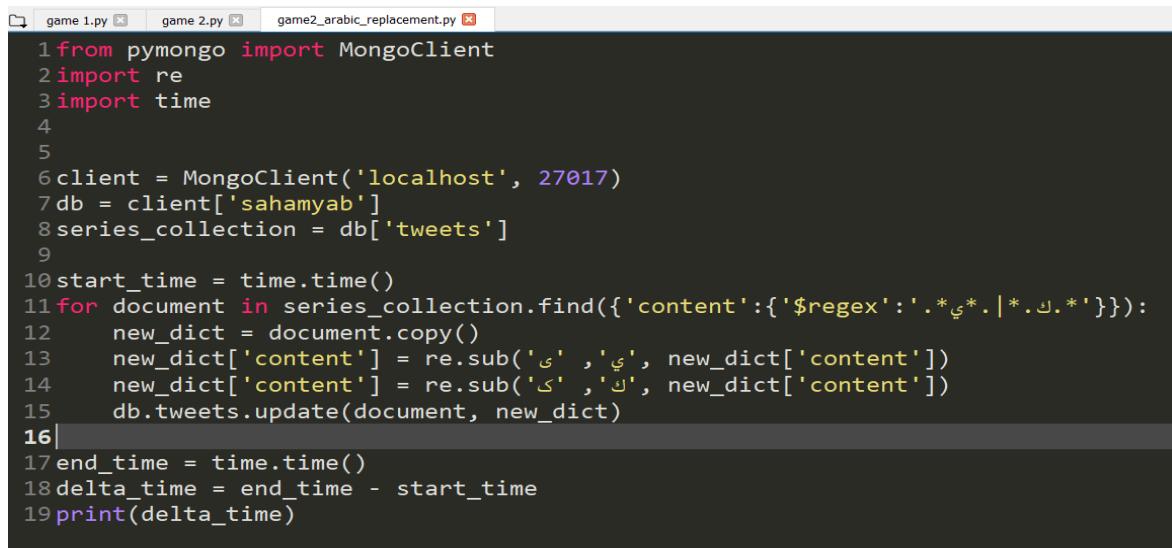
```

{
  "_id": "6097a0f961e33463ca3b8a36",
  "id": "282199290",
  "sendTime": "2021-05-09T08:43:53Z",
  "sendTimePersian": "1400/05/09 13:13",
  "senderName": "Uncle",
  "senderUsername": "hamid1378620",
  "senderProfileImage": "default",
  "content": "این وضاحت قبل قلن مهم ها در مهای خروق از محل برخاسته بیرون و ...",
  "type": "text",
  "scoredPostDate": "1620549833734",
  "finalPullDatePersian": "...",
  "hashtags": []
}

{
  "_id": "6097a0f961e33463ca3b8a37",
  "id": "282199292",
  "sendTime": "2021-05-09T08:43:49Z",
  "sendTimePersian": "1400/05/09 13:13",
  "senderName": "امیر وابسته",
  "senderUsername": "irajrjdmnsh",
  "senderProfileImage": "0c941921-01c3-4916-88b3-a1113851cea",
  "content": "... من یه دل نوخته دائم برای کسانی که مرتبت میکن یه دل خوب پایاند ...",
  "type": "text",
  "scoredPostDate": "1620549878470",
  "finalPullDatePersian": "...",
  "hashtags": []
}

```

البته قبل از اجرای کد فوق، حروف عربی با حروف فارسی به کمک کد زیر جایگزین شد.



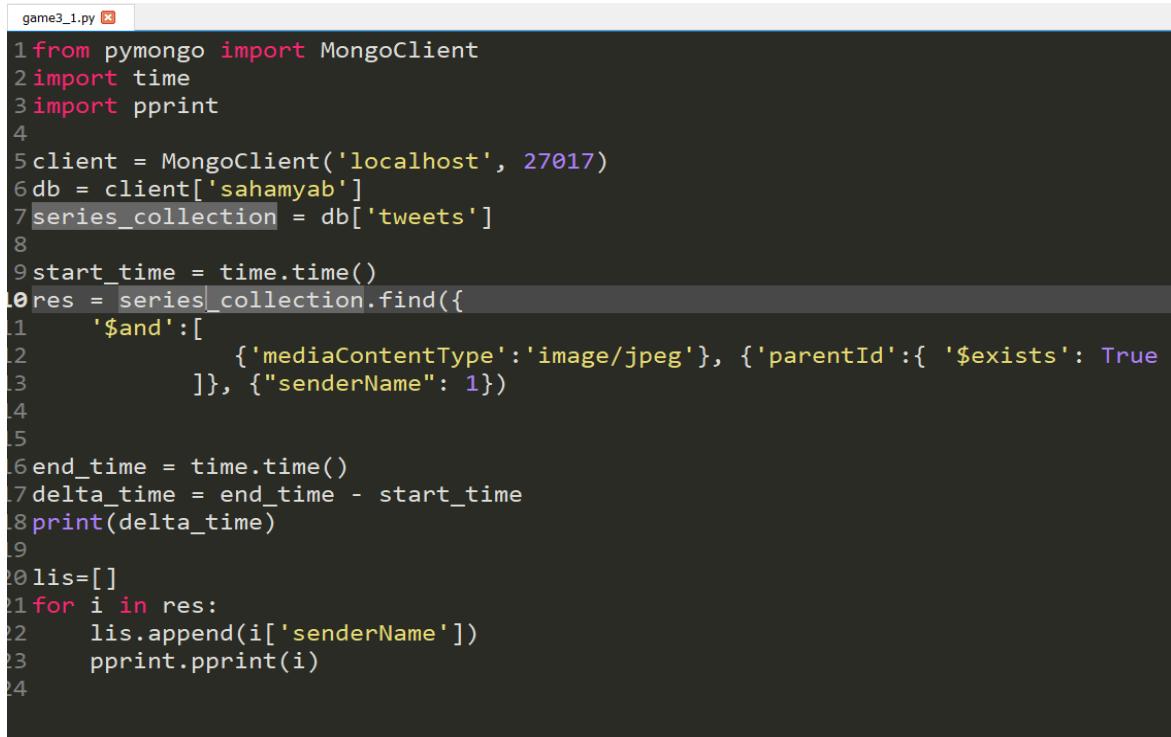
```
game 1.py game 2.py game2_arabic_replacement.py
1 from pymongo import MongoClient
2 import re
3 import time
4
5
6 client = MongoClient('localhost', 27017)
7 db = client['sahamyab']
8 series_collection = db['tweets']
9
10 start_time = time.time()
11 for document in series_collection.find({'content': {'$regex': '.*ي.*.|.*ك.*'}}):
12     new_dict = document.copy()
13     new_dict['content'] = re.sub('ي', 'ي', new_dict['content'])
14     new_dict['content'] = re.sub('ك', 'ك', new_dict['content'])
15     db.tweets.update(document, new_dict)
16
17 end_time = time.time()
18 delta_time = end_time - start_time
19 print(delta_time)
```

## گام سوم

### بخش ۱

دستور نوشته شده:

زمان اجرا: خیلی سریع بود و تقریبا صفر ثانیه



```
game3_1.py
1 from pymongo import MongoClient
2 import time
3 import pprint
4
5 client = MongoClient('localhost', 27017)
6 db = client['sahamyab']
7 series_collection = db['tweets']
8
9 start_time = time.time()
10 res = series_collection.find({
11     '$and': [
12         {'mediaContentType': 'image/jpeg'}, {'parentId': {'$exists': True}},
13         {"senderName": 1}
14     ]
15 }
16 end_time = time.time()
17 delta_time = end_time - start_time
18 print(delta_time)
19
20 lis=[]
21 for i in res:
22     lis.append(i['senderName'])
23     pprint.pprint(i)
24
```

خر و جی :

```
{'_id': ObjectId('6097a0f961e33463ca3b8a39'), 'senderName': 'فرشاد'}  
'_id': ObjectId('6097a0f961e33463ca3b8a3c'), 'senderName': 'محب'}  
'_id': ObjectId('6097a3dc61e33463ca3b8a7e'), 'senderName': 'مهدی'}  
'_id': ObjectId('6097a49061e33463ca3b8a8d'), 'senderName': 'مهدی'}  
'_id': ObjectId('6097abd661e33463ca3b8b2c'), 'senderName': 'tazekar'}  
'_id': ObjectId('6097b6a961e33463ca3b8bc6'), 'senderName': 'Wave Rider'}  
'_id': ObjectId('6097b80761e33463ca3b8bd8'), 'senderName': 'حروفه ای'}  
'_id': ObjectId('6097b87d61e33463ca3b8be0'), 'senderName': 'وحید'}  
'_id': ObjectId('6097b9e561e33463ca3b8bf5'), 'senderName': 'وحید'}  
'_id': ObjectId('6097b9e661e33463ca3b8bf7'), 'senderName': 'وحید'}  
'_id': ObjectId('6097bf0d61e33463ca3b8c3c'), 'senderName': 'مید پوینت'}  
'_id': ObjectId('6097c07561e33463ca3b8c54'), 'senderName': 'fati'}  
'_id': ObjectId('6097cc2d61e33463ca3b8cf3'), 'senderName': 'Amir ahj'}  
'_id': ObjectId('6097d0dd61e33463ca3b8d39'), 'senderName': 'Masoud.RE'}  
'_id': ObjectId('6097dc5a61e33463ca3b8dc6'), 'senderName': 'حمید'}  
'_id': ObjectId('6097e18261e33463ca3b8df1'), 'senderName': 'Vahid2727'}  
'_id': ObjectId('6097e50661e33463ca3b8e1c'), 'senderName': 'علیرضا محمدی'}
```

## بخش ۲

برای این کار، از update\_many استفاده می‌کنیم. همینطور چک می‌کنیم که آیا یکی از المان‌های شبندر، شستا، فولاد در ارائه هشتگ وجود دارد یا خیر. زمان اجرا : ۰.۰۳۳۱۷۵۷۰۶۸۶۳ ثانیه

کد نوشته شده :

```
game3_2.py game3_3.py  
1 from pymongo import MongoClient  
2 import time  
3  
4 client = MongoClient('localhost', 27017)  
5 db = client['sahamyab']  
6 series_collection = db['tweets']  
7  
8 start_time = time.time()  
9 series_collection.update_many(  
10     {'hashtags': {'$in': ['فولاد', 'شستا', 'شبندر']}},  
11     {'$set': {'gov': True}})  
12  
13 end_time = time.time()  
14 delta_time = end_time - start_time  
15 print(delta_time)
```

همان‌طور که می‌بینیم فیلد gov با مقدار true به هشتگ‌های حاوی شبدر، شستا، فولاد اضافه شده است

The screenshot shows the MongoDB Compass interface with the 'sahamyab.tweets' collection selected. The table header includes 'DOCUMENTS 0', 'TOTAL SIZE 0B', 'AVG. SIZE 0B', and 'INDEXES'. Below the table, there are buttons for 'ADD DATA', 'VIEW', 'FILTER', 'OPTIONS', and 'FIND'. A message at the bottom right says 'Displaying documents 161 - 180 of 1001'. One document is expanded to show its full structure:

```

{
  "_id": ObjectId("6097a88c61e33463ca3b8ae5"),
  "id": "282203124",
  "sendTime": "2021-05-09T09:16:27Z",
  "sendTimePersian": "۱۴۰۰/۰۲/۱۹ ۱۳:۴۶",
  "parentSendTime": "2021-05-09T09:06:23Z",
  "parentSendTimePersian": "۱۴۰۰/۰۲/۱۹ ۱۳:۳۶",
  "parentId": "28220246",
  "parentSenderName": "deniz",
  "parentSenderUsername": "denizs81",
  "parentSenderProfileImage": "default",
  "parentContent": "#شبدر ... هر بلاسی سرمون بیاد حنون برا اینکه هر بلاسی سرمون میخواهد ...",
  "senderName": "hbdimeshk",
  "senderUsername": "هندیمشک",
  "senderProfileImage": "81faba73-85a3-4714-90fc-82c39dd99c61",
  "content": "...شیند و قتن خیاب می خرین و نمید رشد لار ... دنطر تیکیکردن بیشه ...",
  "type": "quote",
  "finalPullDatePersian": "",
  "hashtags": [
    "شبدر"
  ],
  "gov": true
}

```

### بخش ۳

افرادی که در بازه‌ی ۱۳ الی ۱۴ توقیت زده بودند را پیدا کردم. تعداد افراد ۲۳۱ نفر بود. زمان اجرا تقریباً صفر ثانیه. کد نوشته شده و قسمتی از خروجی، به شکل زیر است.

چون تعداد افراد زیاد است، یکسری از آنها را اینجا می‌آورم.

The screenshot shows a Jupyter Notebook cell with the following code:

```

from pymongo import MongoClient
import time
import pprint

client = MongoClient('localhost', 27017)
db = client['sahamyab']
series_collection = db['tweets']
start_time = time.time()

res = series_collection.find(
    {'sendTimePersian': {'$regex': '.* 13:.*'} },
    {'senderName': 1, 'senderProfileImage': 1, '_id': 0})
end_time = time.time()
delta_time = end_time - start_time
print(delta_time)

twitt_time_intvrl = []
for i in res:
    twitt_time_intvrl.append([i['senderName'],
                             i['senderProfileImage']])
pprint.pprint(i)

```

On the right, the results are displayed in a table and the Python console:

Name	Type	Size	Value
twitt_time_intvrl	list	231	[{'senderName': 'محمدکاهه', 'senderProfileImage': 'd5eeb597-95e1-4b66-bebd-41f43f228358'}, {'senderName': 'mnbv', 'senderProfileImage': 'default'}, {'senderName': 'جال', 'senderProfileImage': 'default'}, {'senderName': 'hv', 'senderProfileImage': 'default'}, {'senderName': 'حیدر رضا', 'senderProfileImage': 'default'}, {'senderName': 'رس فیم', 'senderProfileImage': 'a5e63632-faad-460e-a800-146324e58907'}, {'senderName': 'اصناد دانشگاه', 'senderProfileImage': '4228a3b3-8453-4850-8d85-2249f79447f5'}, {'senderName': 'mnbv', 'senderProfileImage': 'default'}, {"...} (many more rows)

```

{'senderName': 'محمدکاهه', 'senderProfileImage': 'd5eeb597-95e1-4b66-bebd-41f43f228358'},
{'senderName': 'mnbv', 'senderProfileImage': 'default'},
{'senderName': 'جال', 'senderProfileImage': 'default'},
{'senderName': 'hv', 'senderProfileImage': 'default'},
{'senderName': 'حیدر رضا', 'senderProfileImage': 'default'},
{'senderName': 'رس فیم', 'senderProfileImage': 'a5e63632-faad-460e-a800-146324e58907'},
{'senderName': 'اصناد دانشگاه', 'senderProfileImage': '4228a3b3-8453-4850-8d85-2249f79447f5'},
{'senderName': 'mnbv', 'senderProfileImage': 'default'},
{'senderName': 'Seyed68z', 'senderProfileImage': 'default'},
{'senderName': 'مرتضی', 'senderProfileImage': 'f3fb374d-7202-454b-b7eb-c3d19c3c48b8'},
{'senderName': 'ایرج رادمنش', 'senderProfileImage': '0c941921-01c3-4916-88b3-ae1113851cea'},
{'senderName': 'افشین', 'senderProfileImage': 'c6d1f8d5-5cc9-4407-808d-33b12212386d'},
{'senderName': 'شوگان بورس ایران', 'senderProfileImage': 'ee3c031f-e596-4d77-933e-22b259a61b34'},
{'senderName': 'سلطان شکوریا', 'senderProfileImage': '72756ada-135b-4239-bb66-be1fa269c361'}

```

## گام چهارم

### بخش ۱

دستور نوشته شده، خروجی و زمان اجرا در عکس زیر قابل مشاهده است.

The screenshot shows the Spyder IDE interface. On the left, there is a code editor with several tabs open, including game4\_1.py, game4\_2.py, game4\_3.py, game4\_4.py, game4\_5.py, and game4\_6.py. The active tab is game4\_1.py, which contains the following code:

```

10 match_gtr_4 = {'$match': { "twitt_number": { '$eq': 1 } }}
11 project = {'$project' : { "twitt_number":1, '_id': 0}}
12
13 res_one_tweet = series_collection.aggregate(
14     [count_tweet_of_each_ID, match_gtr_4, project])
15 ##### finding users with 2&3 tweets:
16 count_tweet_of_each_ID = {'$group' :{
17     '_id' : "$senderUsername",
18     'twitt_number': { '$sum': 1 }}}
19 match_gtr_4 = {'$match': { "twitt_number": { '$gt': 1, '$lt': 4 } }}
20 project = {'$project' : { "twitt_number":1, '_id': 0}}
21
22 res_2_3_tweets = series_collection.aggregate(
23     [count_tweet_of_each_ID, match_gtr_4, project])
24 ##### finding users with more than 3 tweets:
25 count_tweet_of_each_ID = {'$group' :{'_id' : "$senderUsername",
26     'twitt_number': { '$sum': 1 }}}
27 match_gtr_4 = {'$match': { "twitt_number": { '$gte': 4 } }}
28 project = {'$project' : { "twitt_number":1, '_id': 0}}
29
30 res_more_than_3 = series_collection.aggregate(
31     [count_tweet_of_each_ID, match_gtr_4, project])
32
33 end_time = time.time()
34 delta_time = end_time - start_time
35 print('run time :', delta_time)
36 print(f'Number of users with one tweet: {len(list(res_one_tweet))}')
37 print(f'Number of users with two and three tweets: {len(list(res_2_3_tweets))}')
38 print(f'Number of users with more than three tweets: {len(list(res_more_than_3))}')

```

On the right side, there is a 'Source Console' window with an 'Usage' help panel. Below it is an 'IPython console' window showing the execution results:

```

In [2]: runfile('D:/Big dat HW/02/part 1/1000 twiits/game4_1.py', wdir='D:/Big dat HW/02/part 1/1000 twiits')
run time : 0.07210183143615723
Number of users with one tweet: 365
Number of users with two and three tweets: 140
Number of users with more than three tweets: 48

In [3]:

```

### بخش ۲

دستور نوشته شده، بخشی از خروجی و زمان اجرا در عکس زیر قابل مشاهده است.

The screenshot shows the Spyder IDE interface. On the left, there is a code editor with several tabs open, including game4\_2.py, game4\_3.py, game4\_4.py, game4\_5.py, and game4\_6.py. The active tab is game4\_2.py, which contains the following code:

```

1 from pymongo import MongoClient
2 import time
3 import pprint
4
5 client = MongoClient('localhost', 27017)
6 db = client['sahamyab']
7 series_collection = db['tweets']
8
9 start_time = time.time()
10 unwind_hashtag_arr = { '$unwind': "$hashtags" }
11
12 group_by = {'$group' :{'_id' : "$hashtags",
13     'twitt_number': { '$sum': 1 }}}
14
15 sort = { '$sort' : { 'twitt_number' : -1 } }
16
17 res_one_tweet = series_collection.aggregate(
18     [unwind_hashtag_arr, group_by, sort])
19
20 ##### end time
21 end_time = time.time()
22 delta_time = end_time - start_time
23 print('Run time', delta_time)
24 ## printing elements of res_more_than_3
25 for i in res_one_tweet:
26     pprint.pprint(i)
27
28

```

On the right side, there is a 'Source Console' window with an 'Usage' help panel. Below it is an 'IPython console' window showing the execution results:

```

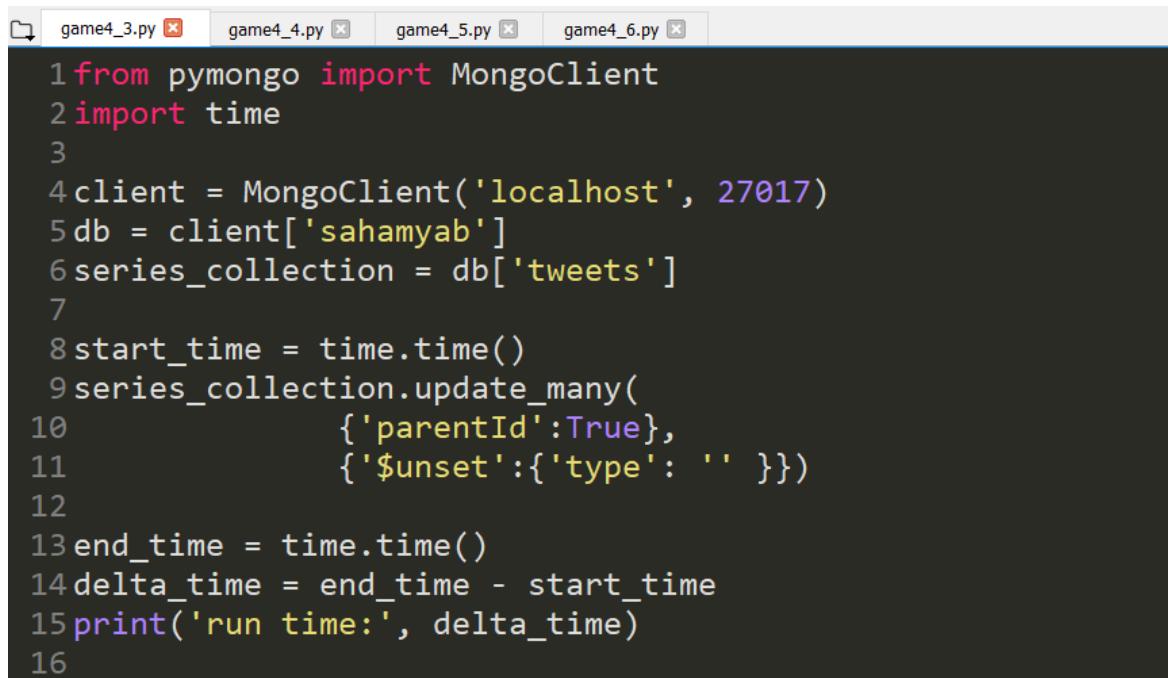
In [24]: runfile('D:/Big dat HW/02/part 1/1000 twiits/game4_2.py', wdir='D:/Big dat HW/02/part 1/1000 twiits')
Run time 0.031000614166259766
[{'_id': 'شاخص‌بیوس', 'twitt_number': 117}, {'_id': 'برکت', 'twitt_number': 58}, {'_id': 'خودرو', 'twitt_number': 36}, {'_id': 'شینا', 'twitt_number': 30}, {'_id': 'پلایش', 'twitt_number': 27}, {'_id': 'شستا', 'twitt_number': 27}, {'_id': 'گشان', 'twitt_number': 26}, {'_id': 'خسپا', 'twitt_number': 16}, {'_id': 'ویملت', 'twitt_number': 15}, {'_id': 'فمنی', 'twitt_number': 14}, {'_id': 'شپلی', 'twitt_number': 13}, {'_id': 'فولاد', 'twitt_number': 12}, {'_id': 'شتراں', 'twitt_number': 11}, {'_id': 'تجارت', 'twitt_number': 11}, {'_id': 'ویمادر', 'twitt_number': 11}, {'_id': 'ذوب', 'twitt_number': 11}, {'_id': 'دی', 'twitt_number': 10}, {'_id': 'پترول', 'twitt_number': 10}, {'_id': 'نگستر', 'twitt_number': 9}]

```

### ٣ بخش

run time: 0.0319676399230957

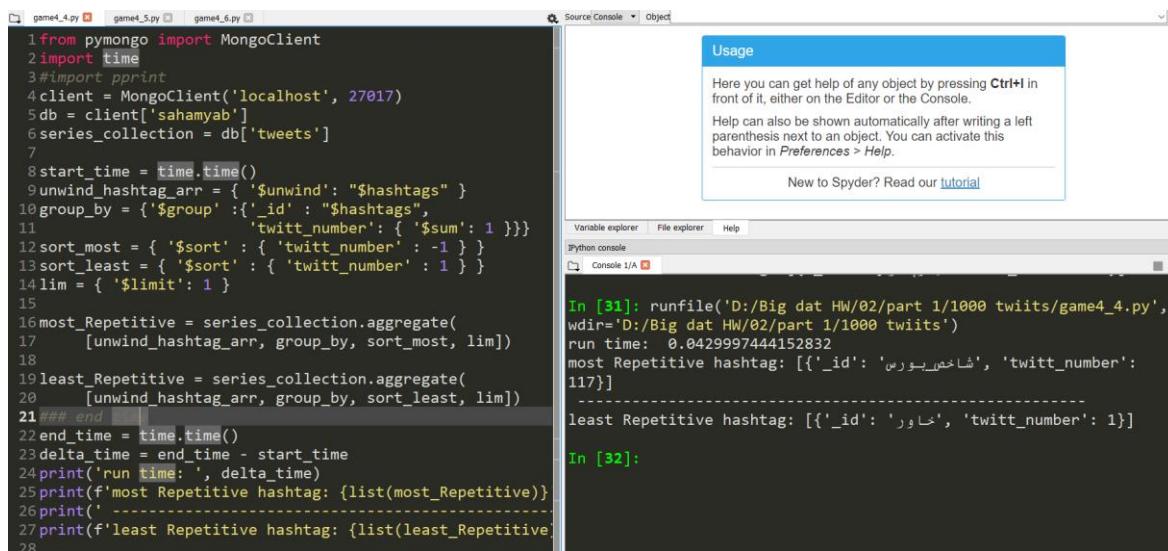
کد نوشته شده:



```
1 from pymongo import MongoClient
2 import time
3
4 client = MongoClient('localhost', 27017)
5 db = client['sahamyab']
6 series_collection = db['tweets']
7
8 start_time = time.time()
9 series_collection.update_many(
10         {'parentId':True},
11         {'$unset':{'type': ''}})
12
13 end_time = time.time()
14 delta_time = end_time - start_time
15 print('run time:', delta_time)
16
```

### ٤ بخش

دستور نوشته شده، خروجی و زمان اجرا در عکس زیر قابل مشاهده است.



```
1 from pymongo import MongoClient
2 import time
3 #import pprint
4 client = MongoClient('localhost', 27017)
5 db = client['sahamyab']
6 series_collection = db['tweets']
7
8 start_time = time.time()
9 unwind_hashtag_arr = { '$unwind': '$hashtags' }
10 group_by = { '$group' : { '_id' : '$hashtags',
11                         'twitt_number': { '$sum': 1 } } }
12 sort_most = { '$sort' : { 'twitt_number': -1 } }
13 sort_least = { '$sort' : { 'twitt_number': 1 } }
14 lim = { '$limit': 1 }
15
16 most_Repetitive = series_collection.aggregate(
17     [unwind_hashtag_arr, group_by, sort_most, lim])
18
19 least_Repetitive = series_collection.aggregate(
20     [unwind_hashtag_arr, group_by, sort_least, lim])
21 ### end []
22 end_time = time.time()
23 delta_time = end_time - start_time
24 print('run time: ', delta_time)
25 print(f'most Repetitive hashtag: {list(most_Repetitive)}')
26 print('-----')
27 print(f'least Repetitive hashtag: {list(least_Repetitive)}')
28
```

Usage  
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.  
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.  
New to Spyder? Read our [tutorial](#)

In [31]: runfile('D:/Big dat HW/02/part 1/1000 twiits/game4\_4.py', wdir='D:/Big dat HW/02/part 1/1000 twiits')  
run time: 0.042997444152832  
most Repetitive hashtag: [{"\_id": "\u062f\u06cc\u0628\u062f\u06cc", "twitt\_number": 117}]  
-----  
least Repetitive hashtag: [{"\_id": "\u062a\u0628\u06cc", "twitt\_number": 1}]  
In [32]:

## بخش ۵

دستور نوشته شده، خروجی و زمان اجرا در عکس زیر قابل مشاهده است.

The screenshot shows the Spyder IDE interface with two tabs open: game4\_5.py and game4\_6.py. The code in game4\_5.py is as follows:

```
1 from pymongo import MongoClient
2 import time
3 import pprint
4
5 client = MongoClient('localhost', 27017)
6 db = client['sahamyab']
7 series_collection = db['tweets']
8
9 start_time = time.time()
10### finding 10-most relevant tweets:
11# filter by day number
12filter_time = { '$match': { 'sendTimePersian': { '$regex': '.*/19 .*' }}}
13### unwind array elements
14unwind_hashtag_arr = { '$unwind': '$hashtags' }
15count_tweets = { '$group' :
16    { '_id' : '$hashtags', 'twitt_number': { '$sum': 1 }}}
17### sorting by tweets number
18sort = { '$sort' : { 'twitt_number' : -1 } }
19### Limiting the most ten hashtags
20lim = { '$limit': 10 }
21
22ten_most_relevant = series_collection.aggregate(
23    [filter_time, unwind_hashtag_arr, count_tweets, sort, lim])
24
25end_time = time.time()
26delta_time = end_time - start_time
27print('run time:', delta_time)
28for i in ten_most_relevant:
29    pprint.pprint(i)
```

The output in the IPython console shows the results:

```
In [34]: runfile('D:/Big dat HW/02/part 1/twiits/game4_5.py', wdir='D:/Big dat HW/02/1/1000 twiits')
run time: 0.04003024101257324
[{'_id': 'فاطمه بورس', 'twitt_number': 116}, {'_id': 'ابرکت', 'twitt_number': 58}, {'_id': 'نگودرو', 'twitt_number': 34}, {'_id': 'شینا', 'twitt_number': 30}, {'_id': 'پالایش', 'twitt_number': 27}, {'_id': 'فشن', 'twitt_number': 27}, {'_id': 'کشان', 'twitt_number': 26}, {'_id': 'خسایا', 'twitt_number': 16}, {'_id': 'شهلی', 'twitt_number': 13}, {'_id': 'نمی', 'twitt_number': 12}]
```

## بخش ۶

دستور نوشته شده، خروجی و زمان اجرا در عکس زیر قابل مشاهده است.

The screenshot shows the Spyder IDE interface with two tabs open: game4\_5.py and game4\_6.py. The code in game4\_6.py is as follows:

```
1 from pymongo import MongoClient
2 import time
3 import pprint
4
5 client = MongoClient('localhost', 27017)
6 db = client['sahamyab']
7 series_collection = db['tweets']
8 start_time = time.time()
9 # filter by day number, here I filtered by 14.
10filter_time = { '$match': { 'sendTimePersian': { '$regex': '.*/19 .*' }}}
11# group by senderUsername and counting each one tweets
12count_tweet_of_each_ID = { '$group' :
13    { '_id' : '$senderUsername',
14        'twitt_number': { '$sum': 1 }}}
15### sorting by tweets number
16sort = { '$sort' : { 'twitt_number' : -1 } }
17### Limiting the most ten hashtags
18lim = { '$limit': 1 }
19project = { '$project' : { "twitt_number":1, 'sendTimePersian':1 } }
20
21res_one_tweet = series_collection.aggregate(
22    [filter_time, count_tweet_of_each_ID, sort, lim])
23
24end_time = time.time()
25delta_time = end_time - start_time
26print('run time:', delta_time)
27for i in res_one_tweet:
28    pprint.pprint(i)
```

The output in the IPython console shows the results:

```
In [36]: runfile('D:/Big dat HW/02/part 1/twiits/game4_6.py', wdir='D:/Big dat HW/02/1/1000 twiits')
run time: 0.04200458526611328
[{'_id': 'f1364', 'twitt_number': 30}]

In [37]:
```

## گام پنجم

گام سوم ، سوال ۱:

The screenshot shows a Jupyter Notebook interface. On the left, the code for game5\_1.py is displayed:

```
1 import pymongo
2 from pymongo import MongoClient
3 import time
4 import pprint
5
6 client = MongoClient('localhost', 27017)
7 db = client['sahamyab']
8 series_collection = db['tweets']
9
10### Adding index to field
11series_collection.create_index([('mediaContentType', pymongo.DESCENDING), ("parentId", 1)])
12
13start_time = time.time()
14res = series_collection.find({
15    '$and': [
16        {'mediaContentType': 'image/jpeg'}, {"parentId": {"$exists": True}}
17    ], {"senderName": 1, '_id': 0}
18})
19end_time = time.time()
20delta_time = end_time - start_time
21print('run time:', delta_time)
22
23lis = []
24for i in res:
25    lis.append(i['senderName'])
26    pprint.pprint(i)
27
28
```

On the right, the output of the code execution is shown in the 'Console' tab:

```
twiits/game5_1.py', wdir='D:/Big dat HW/02/part 1/1000 twiits')
run time: 0.0010404586791992188
{'senderName': 'علييرا مهدى'}, {'senderName': 'Vahid2727'}, {'senderName': 'Masoud.RE'}, {'senderName': 'بید بوسینت'}, {'senderName': 'tazekar'}, {'senderName': 'fati'}, {'senderName': 'مهدى'}, {'senderName': 'مهدى'}, {'senderName': 'فرشاد'}, {'senderName': 'Wave Rider'}, {'senderName': 'Amir ahj'}, {'senderName': 'جعید'}, {'senderName': 'حرفة ای'}, {'senderName': 'وحید'}, {'senderName': 'وحید'}, {'senderName': 'وحید'}, {'senderName': 'وحید'}
```

indexing بدون استفاده از run time :

indexing با استفاده از run time : 0.00104

گام سوم ، سوال ۲:

The screenshot shows a Jupyter Notebook interface. On the left, the code for game5\_2.py is displayed:

```
1 from pymongo import MongoClient
2 import time
3 import pymongo
4
5 client = MongoClient('localhost', 27017)
6 db = client['sahamyab']
7 series_collection = db['tweets']
8 ### Adding index to field
9 series_collection.create_index(
10     [('hashtags', pymongo.DESCENDING)])
11
12start_time = time.time()
13series_collection.update_many(
14    {'hashtags': {'$in': ['فولاد', 'شنبه', 'شنبه', 'شنبه']}},
15    {'$set': {'gov': True}})
16
17end_time = time.time()
18delta_time = end_time - start_time
19print('run time:', delta_time)
20
21
```

On the right, the output of the code execution is shown in the 'Console' tab:

```
In [3]: runfile('D:/Big dat HW/02/part 1/1000 twiits/game5_2.py', wdir='D:/Big dat HW/02/part 1/1000 twiits')
run time: 0.004064083099365234
In [4]:
```

indexing بدون استفاده از run time : 0.03317

indexing با استفاده از run time : 0.00406

## گام چهارم ، سوال ۱:

```

Editor - D:/Big dat HW/02/part 1/1000 twiits/game5_3.py
game5_3.py game5_4.py game5_5.py
11     [("senderUsername", pymongo.DESCENDING)])
12 start_time = time.time()
13 ##### finding users with one tweet:
14 count_tweet_of_each_ID = {'$group' :
15     {'_id': '$senderUsername', 'twitt_number': { '$sum': 1 }}}
16 match_gtr_4 = {'$match': { "twitt_number": { '$eq': 1 } }}
17 project = {'$project' : { "twitt_number":1, '_id': 0}}
18
19 res_one_tweet = series_collection.aggregate(
20     [count_tweet_of_each_ID, match_gtr_4, project])
21 ##### finding users with 2&3 tweets:
22 count_tweet_of_each_ID = {'$group' :
23     {'_id': "$senderUsername", 'twitt_number': { '$sum': 1 }}}
24 match_gtr_4 = {'$match': { "twitt_number": { '$gt': 1, '$lt': 4 } }}
25 project = {'$project' : { "twitt_number":1, '_id': 0}}
26
27 res_2_3_tweets = series_collection.aggregate(
28     [count_tweet_of_each_ID, match_gtr_4, project])
29 ##### finding users with more than 3 tweets:
30 count_tweet_of_each_ID = {'$group' :
31     {'_id': "$senderUsername",
32      'twitt_number': { '$sum': 1 }}}
33 match_gtr_4 = {'$match': { "twitt_number": { '$gte': 4 } }}
34 project = {'$project' : { "twitt_number":1, '_id': 0}}
35
36 res_more_than_3 = series_collection.aggregate(
37     [count_tweet_of_each_ID, match_gtr_4, project])
38 end_time = time.time()
39 delta_time = end_time - start_time
40 print('run time:', delta_time)

```

In [6]: runfile('D:/Big dat HW/02/part 1/1000 twiits/game5\_3.py', wdir='D:/Big dat HW/02/part 1/1000 twiits')
run time: 0.012715816497802734
Number of users with one tweet: 365
Number of users with two and three tweets: 140
Number of users with more than three tweets: 48

In [7]:

indexing بدون استفاده از run time : 0.0721

indexing با استفاده از run time : 0.012715

## گام چهارم ، سوال ۵:

```

game5_4.py
1 from pymongo import MongoClient
2 import time
3 import pprint
4 import pymongo
5
6 client = MongoClient('localhost', 27017)
7 db = client['sahamyab']
8 series_collection = db['tweets']
9
10 ##### Adding index to field
11 series_collection.create_index(
12     [("$sendTimePersian", pymongo.DESCENDING),
13      ("hashtags", pymongo.DESCENDING)])
14
15 start_time = time.time()
16 ##### finding 10-most relevant tweets:
17 filter_time = { '$match': { 'sendTimePersian': { '$regex': '.*/14 .*'}}}
18 ### unwind array elements
19 unwind_hashtag_arr = { '$unwind': "hashtags" }
20 count_tweets = {'$group' :{
21     '_id' : "hashtags", 'twitt_number': { '$sum': 1 }}}
22 ### sorting by tweets number
23 sort = { '$sort' : { 'twitt_number' : -1 } }
24 ### Limiting the most ten hashtags
25 lim = { '$limit': 10 }
26
27 ten_most_relevant = series_collection.aggregate(
28     [filter_time, unwind_hashtag_arr, count_tweets, sort, lim])
29
30 end_time = time.time()
31 delta_time = end_time - start_time

```

In [10]: runfile('D:/Big dat HW/02/part 1/1000 twiits/game5\_4.py', wdir='D:/Big dat HW/02/part 1/1000 twiits')
run time: 0.029900434646606445
{'\_id': 'فاخته\_پور', 'twitt\_number': 1}
{'\_id': 'نظرستجو', 'twitt\_number': 1}
{'\_id': 'نگل', 'twitt\_number': 1}

In [11]:

indexing بدون استفاده از run time : 0.040003

indexing با استفاده از run time : 0.02999

## گام چهارم، سوال ۶:

The screenshot shows a Jupyter Notebook environment. On the left, there are two tabs: 'game5\_4.py' and 'game5\_5.py'. The code in 'game5\_5.py' is as follows:

```
1from pymongo import MongoClient
2import time
3import pprint
4import pymongo
5
6client = MongoClient('localhost', 27017)
7db = client['sahamyab']
8series_collection = db['tweets']
9### Adding index to field
10series_collection.create_index(
11    [("sendTimePersian", pymongo.DESCENDING)])
12start_time = time.time()
13# filter by day number, here I filtered by 14.
14filter_time = { '$match':
15    { 'sendTimePersian':{ '$regex': '.*/14 .*'}}}
16# group by senderUsername and counting each one tweets
17count_tweet_of_each_ID = {'$group' :{
18    '_id' : "$senderUsername",'twitt_number': { '$sum': 1 }}}
19### sorting by tweets number
20sort = { '$sort' : { 'twitt_number' : -1 } }
21### limiting the most ten hashtags
22lim = { '$limit': 1 }
23
24res_one_tweet = series_collection.aggregate(
25    [filter_time, count_tweet_of_each_ID, sort, lim])
26
27end_time = time.time()
28delta_time = end_time - start_time
29print('run time:', delta_time)
30for i in res_one_tweet:
31    pprint.pprint(i)
```

The right side of the interface shows the 'Source Console' tab with the output of the code execution. It includes a 'Usage' help panel and the following output:

```
In [7]: runfile('D:/Big dat HW/02/part 1/1000
twiits/game5_5.py', wdir='D:/Big dat HW/02/
part 1/1000 twiits')
run time: 0.0020487308502197266
{'_id': 'saaamm', 'twitt_number': 1}

In [8]:
```

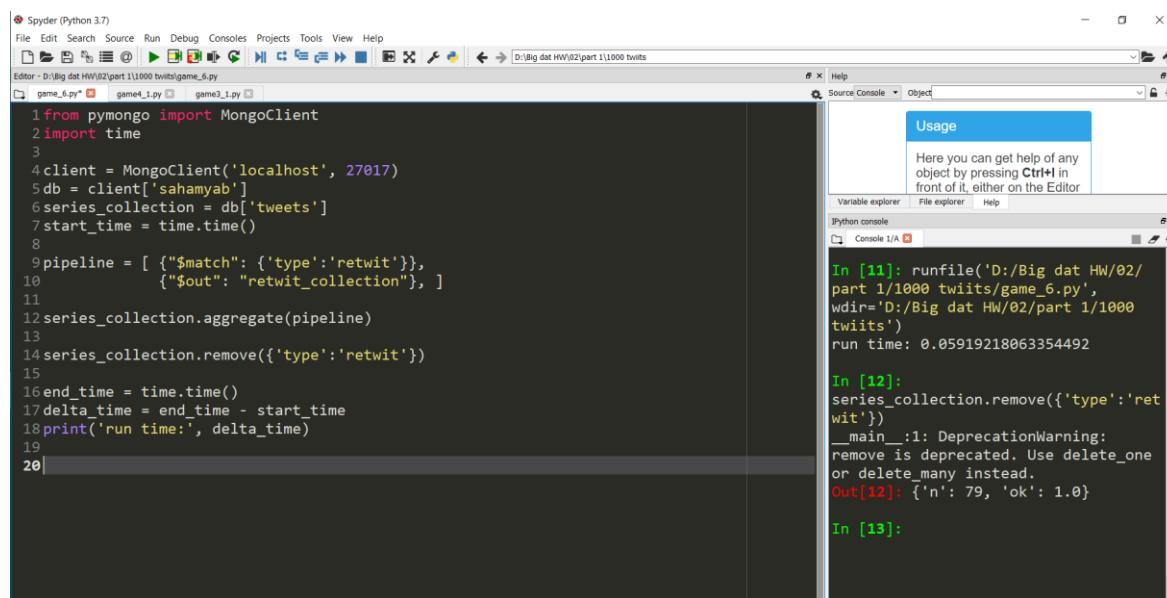
indexing با استفاده از run time : 0.042

indexing با استفاده از run time : 0.00204

همان طور که مشاهده میکنیم، در همه موارد زمان اجرا حدود 0.1 برابر شده است و این مزیت indexing است.

## تفکیک تؤییت‌ها و ریتوئیت‌ها

به کمک کد زیر، retwit‌ها را به کالکشن جدید منتقل می‌کنیم و از کالکشن فعلی حذف می‌کنیم.



The screenshot shows the Spyder Python 3.7 IDE interface. The code editor contains a script named game\_6.py with the following content:

```
1 from pymongo import MongoClient
2 import time
3
4 client = MongoClient('localhost', 27017)
5 db = client['sahamyab']
6 series_collection = db['tweets']
7 start_time = time.time()
8
9 pipeline = [ {"$match": { "type":'retwit'}},
10             {"$out": "retwit_collection"} ]
11
12 series_collection.aggregate(pipeline)
13
14 series_collection.remove({ 'type': 'retwit'})
15
16 end_time = time.time()
17 delta_time = end_time - start_time
18 print('run time:', delta_time)
19
20|
```

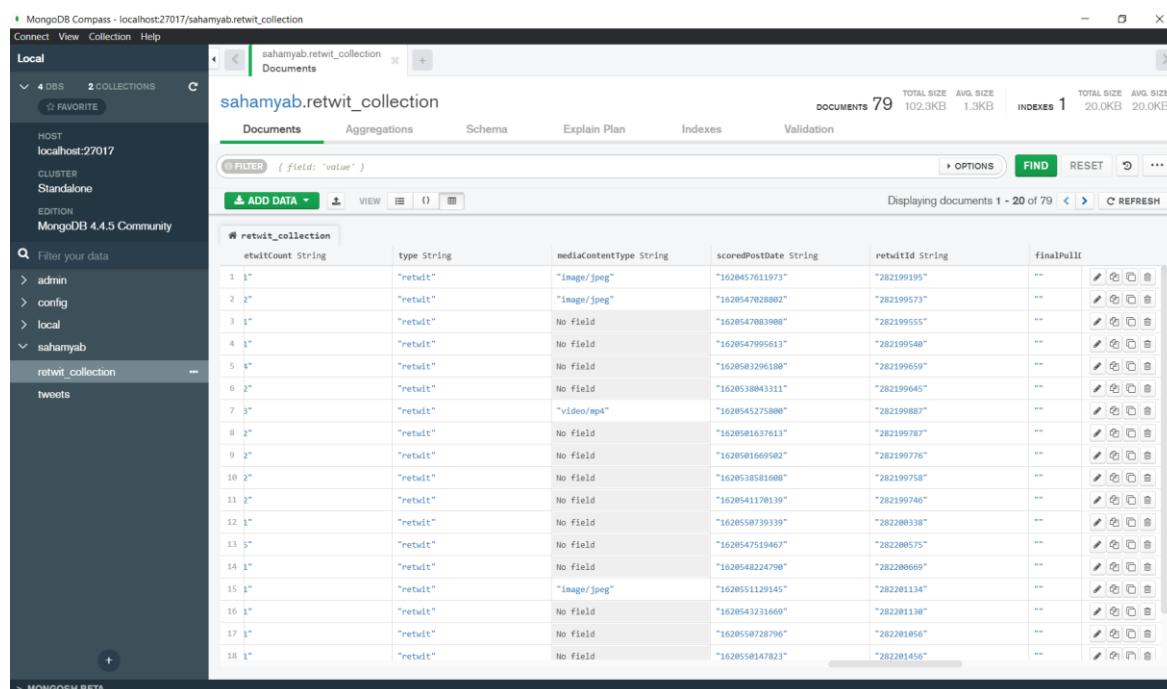
The IPython console shows the execution of the script:

```
In [11]: runfile('D:/Big dat HW/02/part 1/1000 twiits/game_6.py',
wdir='D:/Big dat HW/02/part 1/1000
twiits')
run time: 0.05919218063354492

In [12]:
series_collection.remove({ 'type': 'retwit'})
__main__::: DeprecationWarning:
remove is deprecated. Use delete_one
or delete_many instead.
Out[12]: {'n': 79, 'ok': 1.0}

In [13]:
```

عکس زیر کالکشن جدید شامل retwit‌ها را نشان می‌دهد.



The screenshot shows the MongoDB Compass interface connected to localhost:27017. The left sidebar shows the database structure with a collection named 'retwti\_collection' under the 'sahamyab' database. The main pane displays the contents of this collection:

Document ID	type	mediaContent	scoredPostDate	retwtId	finalPull
1	"retwt"	"image/jpeg"	"1620457611973"	"282190195"	...
2	"retwt"	"image/jpeg"	"1620547028802"	"282199573"	...
3	"retwt"	No field	"1620547083908"	"282199555"	...
4	"retwt"	No field	"1620545995613"	"282190548"	...
5	"retwt"	No field	"162053296180"	"282199659"	...
6	"retwt"	No field	"1620538043311"	"282190645"	...
7	"retwt"	"video/mp4"	"1620545275800"	"282199887"	...
8	"retwt"	No field	"1620501637813"	"282199787"	...
9	"retwt"	No field	"1620501669502"	"282199776"	...
10	"retwt"	No field	"1620538581600"	"282199758"	...
11	"retwt"	No field	"1620541170130"	"282199746"	...
12	"retwt"	No field	"1620550739339"	"282200338"	...
13	"retwt"	No field	"1620547519467"	"282200575"	...
14	"retwt"	No field	"1620548224790"	"282200669"	...
15	"retwt"	"image/jpeg"	"1620551129145"	"282201134"	...
16	"retwt"	No field	"1620543231660"	"282201130"	...
17	"retwt"	No field	"1620550728796"	"282201056"	...
18	"retwt"	No field	"1620550147823"	"282201456"	...

در عکس زیر مشاهده می‌کنیم document هایی که type آنها است از کاکشن قدیمی حذف شده‌اند.

```
_id: ObjectId("56097a0bf901e33403ca3b8a35")
1: 282199315
sendTime: "2021-05-09T08:44:16Z"
sendTimePersian: "1400/02/19 13:14"
parentSendTimePersian: "1400/02/19 13:14"
parentSendTime: "2021-05-09T05:53:07Z"
parentSenderId: "1400/02/21 10:23"
parentId: "266241923"
parentSenderName: "رضا آپن"
parentSenderUsername: "reza_apn"
parentSenderProfileImage: "db8f9c5b-ce3d-47d1-8a92-e80724ef5001"
parentContent: "شایعه محمدی تمام لحظه خودشو داشت میگردید که بزار روزه میگذرد ..."

senderName: "Hamid"
senderUsername: "Hamid2337"
senderProfileImage: "adff6410e-8929-4526-ae88-f657027987f2"
content: "این وعده قدری قبل شدن سه ماه در مدهای قریب از قبل برداشت روزی که برو و ..."
type: "quote"
finalPullDatePersian: ""

_id: ObjectId("56097a0bf901e33403ca3b8a36")
1: 282199299
sendTime: "2021-05-09T08:43:53Z"
sendTimePersian: "1400/02/19 13:13"
senderName: "Hamid"
senderUsername: "Hamid13676509"
senderProfileImage: "default"
content: "وقیدر خسته نمی‌ازم و ..."
type: "Twit"
scoredPostDate: "1626540833734"
finalPullDatePersian: ""
hashtags: Array
```

## بخش دوم

### گام اول

جهت آشنایی با زبان سایفر، به movie project tutorial که به صورت پیش فرض داده های محدودی در آن وجود دارد. عکس زیر محیط سندباکس مربوط به movie را نشان می دهد.

The screenshot shows the Neo4j browser interface. On the left, there's a sidebar with various icons and a "Movies Project Tutorial" section. The main area has a title "What is Cypher?" with a description of what Cypher is and how it's used. Below that is a code editor containing a Cypher query:

```
Match (m:Movie) where m.released > 2000 RETURN m limit 5
```

A blue box below the code editor says: "Hint: You can click on the query above to populate it in the editor." To the right of the code editor, the results of the query are displayed in a graph format. It shows five nodes, each representing a movie: "The Polar Express", "Rescue", "Somethi", "The Matrix Revol...", and "The Matrix Relo...". The nodes are orange circles with their names written inside. Below the graph, it says "Displaying 5 nodes, 0 relationships." At the bottom of the browser window, there's a status bar with the text "\$ :server connect" and a message "You are connected as user neo4j".

توضیح چند کوئری:

This screenshot shows another view of the Neo4j browser. In the code editor, there's a MERGE query:

```
MERGE (p:Person {name: 'John Doe'})  
ON MATCH SET p.lastLoggedInAt = timestamp()  
ON CREATE SET p.createdAt = timestamp()  
RETURN p
```

The results show one node named "John Doe" represented by a blue circle. The status bar at the bottom indicates "Displaying 1 nodes, 0 relationships."

این کوئری می‌گوید اگر نود person وجود دارد، ویژگی `current` آن را برابر زمان فعلی قرار دهد. اگر نود person وجود نداشته باشد، آن را می‌سازد و آن را برابر زمان فعلی قرار می‌دهد.

### کوئری دوم :

```
neo4j$ MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]→
  (:Movie)←[:ACTED_IN]-(p:Person) return p.name
```

p.name
1 "Ed Harris"
2 "Gary Sinise"
3 "Kevin Bacon"
4 "Bill Paxton"
5 "Parker Posey"
6 "Greg Kinnear"

ابتدا برای نودی که از نوع person است و آن "Tom Hanks" است ارتباط ACTED\_IN با فیلم‌ها را بررسی می‌کند و سپس افراد دیگری که با همان فیلم‌ها ارتباط ACTED\_IN دارند را هم پیدا می‌کند (در اینجا این افراد را p نام‌گذاری کرده است). در واقع این کوئری، بازیگران همکار Tom Hanks را معرفی می‌کند.

### کوئری سوم :

```
⚠ 1 MATCH (p:Person), (m:Movie)
  2 WHERE p.name = "Tom Hanks" and m.title = "Cloud Atlas"
  3 CREATE (p)-[w:WATCHED]→(m)
  4 RETURN type(w)
```

```
neo4j$ MATCH (p:Person), (m:Movie) WHERE p.name = "..."
```

type(w)
1 "WATCHED"

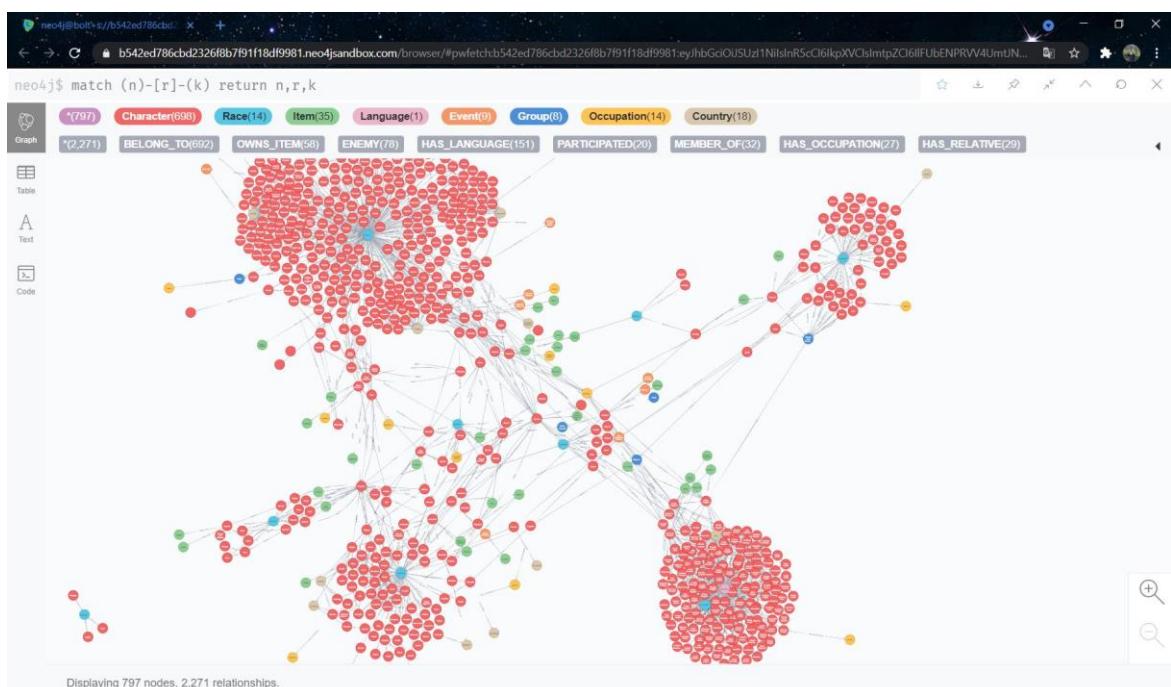
بین نودی شخصی که نام آن Tom Hanks و نود از نوع فیلم که نام آن Cloud Atlas هست، ارتباط جهت دار WATCHED را ایجاد می‌کند.

## گام دوم

password و Bolt username را از سندباکس خالی خودم کپی کردم و در فایل نوت بوکی که در اختیارمان قرار داده شده بود، کپی کردم و سپس همان فایل نوت بوک را در گوگل کولب اجرا کردم. در نتیجه داده‌ها بارگذاری شد.

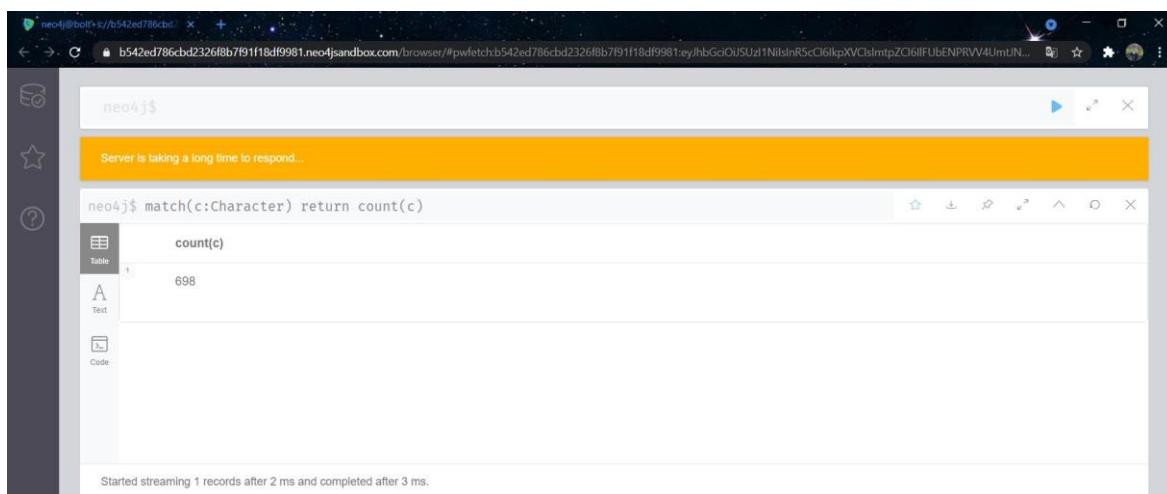
## گام سوم

(۱)



در این کوئری، همهی نودهایی که با نودهای دیگر ارتباطی دارند، آورده شده است.

(الف) در این کوئری نودهای با لیبل "Character" را گرفتیم و تعداد این نودها را به عنوان خروجی دریافت کردیم



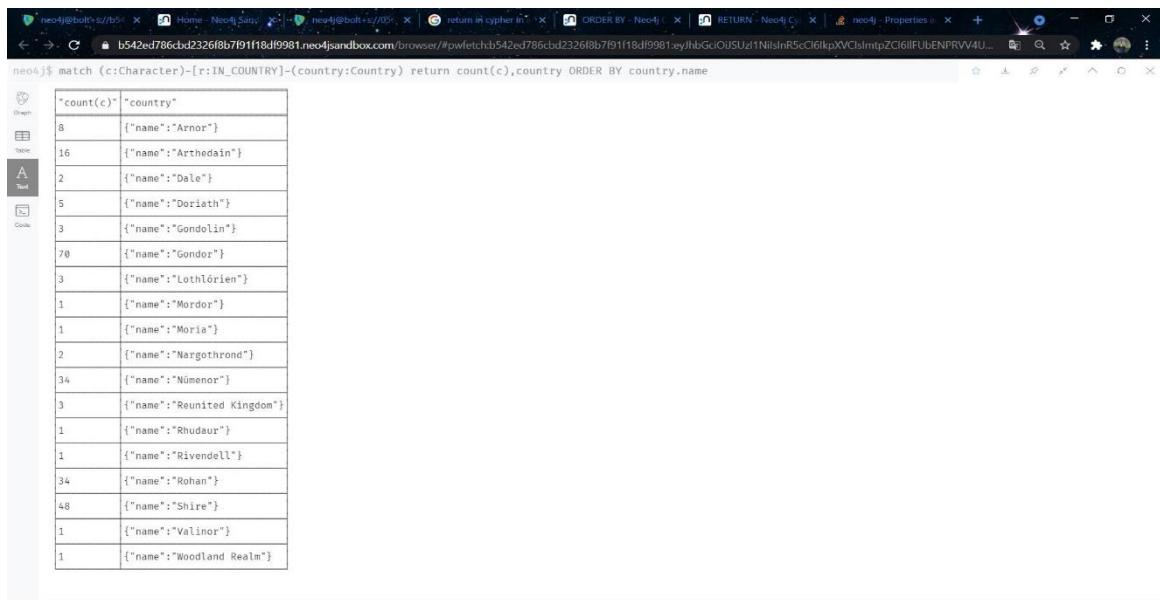
The screenshot shows the Neo4j browser interface. The URL is <http://b542ed786cbd2326f8b7f91f18df9981.neo4jsandbox.com/browser/#pwfetch:b542ed786cbd2326f8b7f91f18df9981:eyJhbGciOiJSUzI1NiIsInR5cIj6IkpXVClsImtpZC16IffUbENPRVv4UmJN..>. The query entered is `neo4j$ match(c:Character) return count(c)`. The results table shows one row with the value 698. A message at the bottom says "Started streaming 1 records after 2 ms and completed after 3 ms."

ب) در این قسمت علاوه بر آنچه در قسمت الف اجرا شد، از `distinct` استفاده می‌کنیم تا کاراکترهایی که نام یکتا دارند در خروجی ظاهر شوند.



The screenshot shows the Neo4j browser interface. The URL is <http://b542ed786cbd2326f8b7f91f18df9981.neo4jsandbox.com/browser/#pwfetch:b542ed786cbd2326f8b7f91f18df9981:eyJhbGciOiJSUzI1NiIsInR5cIj6IkpXVClsImtpZC16IffUbENPRVv4UmJN..>. The query entered is `neo4j$ match(c:Character) return count(distinct(c.name))`. The results table shows one row with the value 661. A message at the bottom says "Started streaming 1 records after 3 ms and completed after 11 ms."

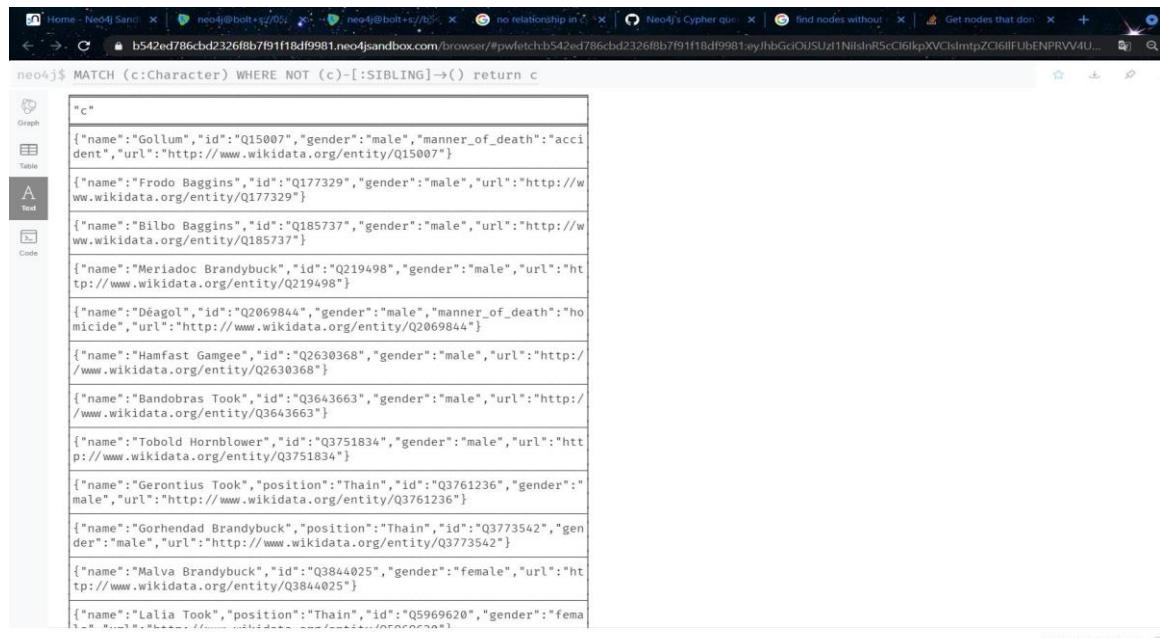
۳) نودهای با لیبل “Character” که با نودهای با لیبل “country” رابطه “IN\_COUNTRY” دارند را پیدا می‌کنیم. سپس تعداد افراد و نام کشور را بر می‌گردانیم. از “order by” برای برگرداندن نتیجه برحسب حروف الفبا استفاده می‌کنیم.



```
neo4j$ match {c:Character}-[r:IN_COUNTRY]-(country:Country) return count(c),country ORDER BY country.name
```

"count(c)"	"country"
8	{"name":"Arnor"}
16	{"name":"Arthedain"}
2	{"name":"Dale"}
5	{"name":"Doriath"}
3	{"name":"Gondolin"}
70	{"name":"Gondor"}
3	{"name":"Lothlórien"}
1	{"name":"Mordor"}
1	{"name":"Moria"}
2	{"name": "Nargothrond"}
34	{"name": "Númenor"}
3	{"name": "Reunited Kingdom"}
1	{"name": "Rhûdaur"}
1	{"name": "Rivendell"}
34	{"name": "Rohan"}
48	{"name": "Shire"}
1	{"name": "Valinor"}
1	{"name": "Woodland Realm"}

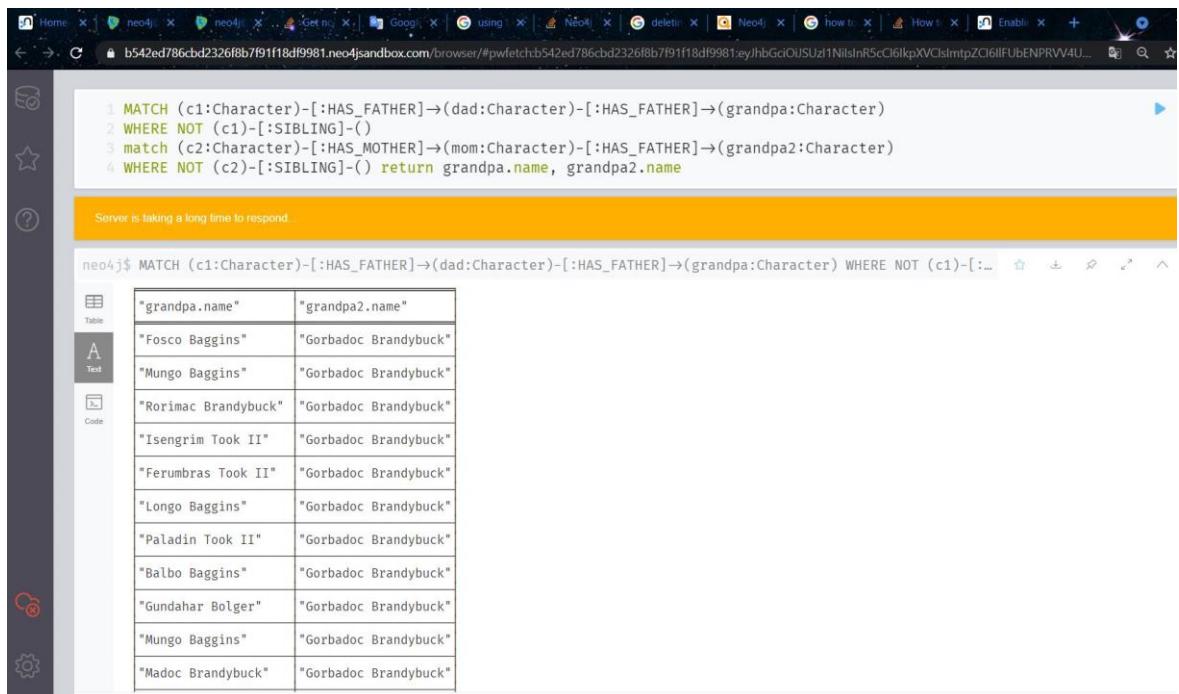
الف) نودهای با لیبل “Character” که با نودهای دیگر رابطه ندارد را می‌یابیم.



```
neo4j$ MATCH (c:Character) WHERE NOT (c)-[:SIBLING]->() return c
```

"c"
{"name": "Gollum", "id": "Q15007", "gender": "male", "manner_of_death": "accident", "url": "http://www.wikidata.org/entity/Q15007"}
{"name": "Frodo Baggins", "id": "Q177329", "gender": "male", "url": "http://www.wikidata.org/entity/Q177329"}
{"name": "Bilbo Baggins", "id": "Q185737", "gender": "male", "url": "http://www.wikidata.org/entity/Q185737"}
{"name": "Meriadoc Brandybuck", "id": "Q219498", "gender": "male", "url": "http://www.wikidata.org/entity/Q219498"}
{"name": "Déagol", "id": "Q2069844", "gender": "male", "manner_of_death": "homicide", "url": "http://www.wikidata.org/entity/Q2069844"}
{"name": "Hamfast Gamgee", "id": "Q2630368", "gender": "male", "url": "http://www.wikidata.org/entity/Q2630368"}
{"name": "Bandobras Took", "id": "Q3643663", "gender": "male", "url": "http://www.wikidata.org/entity/Q3643663"}
{"name": "Tobold Hornblower", "id": "Q3751834", "gender": "male", "url": "http://www.wikidata.org/entity/Q3751834"}
{"name": "Gerontius Took", "position": "Thain", "id": "Q3761236", "gender": "male", "url": "http://www.wikidata.org/entity/Q3761236"}
{"name": "Gorhendad Brandybuck", "position": "Thain", "id": "Q3773542", "gender": "male", "url": "http://www.wikidata.org/entity/Q3773542"}
{"name": "Malva Brandybuck", "id": "Q3844025", "gender": "female", "url": "http://www.wikidata.org/entity/Q3844025"}
{"name": "Lalia Took", "position": "Thain", "id": "Q5969620", "gender": "female", "url": "http://www.wikidata.org/entity/Q5969620"}

ب) نودهای با لیبل "Character" به نام "c1" (نامگذاری توسط خودمان دلبهخواه صورت می‌گیرد) که با نودهای لیبل "Character" به نام "dad" رابطه "HAS\_FATHER" دارند را می‌یابیم. سپس یه همین طریق پدربرزگ نودهای "dad" را می‌یابیم. در نهایت شرط عدم وجود رابطه "SIBLINGS" برای نودهای "c1" را قرار می‌دهیم.



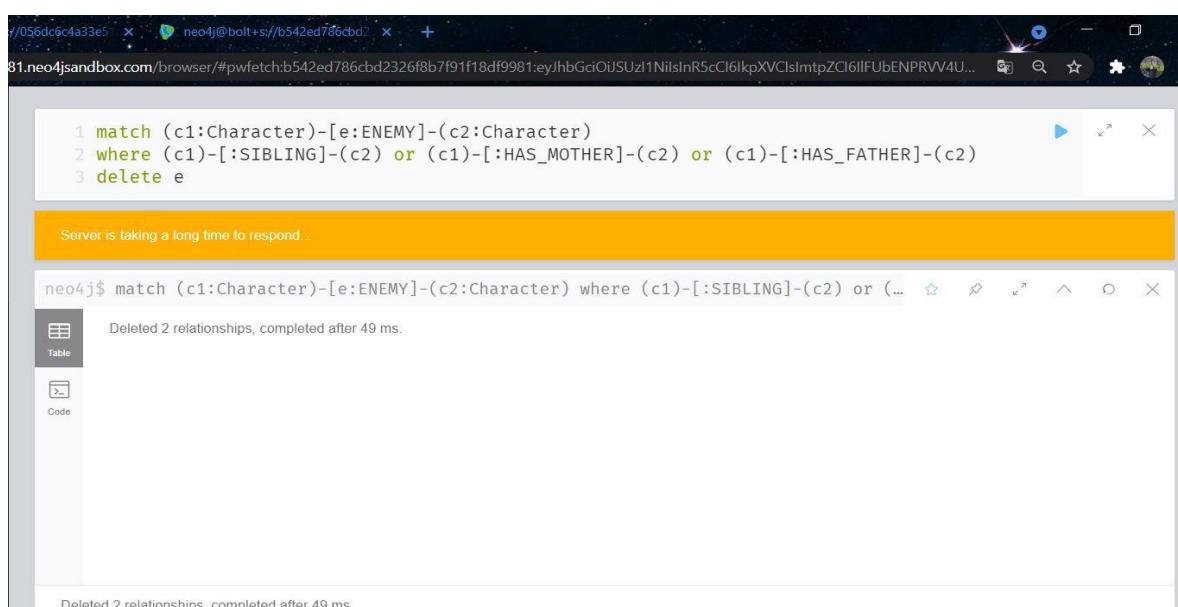
```

1 MATCH (c1:Character)-[:HAS_FATHER]-(dad:Character)-[:HAS_FATHER]-(grandpa:Character)
2 WHERE NOT (c1)-[:SIBLING]-()
3 match (c2:Character)-[:HAS_MOTHER]-(mom:Character)-[:HAS_FATHER]-(grandpa2:Character)
4 WHERE NOT (c2)-[:SIBLING]-() return grandpa.name, grandpa2.name
    
```

Server is taking a long time to respond

grandpa.name	grandpa2.name
"Fosco Baggins"	"Gorbadoc Brandybuck"
"Mungo Baggins"	"Gorbadoc Brandybuck"
"Rorimac Brandybuck"	"Gorbadoc Brandybuck"
"Isengrim Took II"	"Gorbadoc Brandybuck"
"Ferumbras Took II"	"Gorbadoc Brandybuck"
"Longo Baggins"	"Gorbadoc Brandybuck"
"Paladin Took II"	"Gorbadoc Brandybuck"
"Balbo Baggins"	"Gorbadoc Brandybuck"
"Gundahar Bolger"	"Gorbadoc Brandybuck"
"Mungo Baggins"	"Gorbadoc Brandybuck"
"Madoc Brandybuck"	"Gorbadoc Brandybuck"

۵) نودهایی با لیبل "Character" که با هم رابطهی دشمنی دارند را می‌یابیم. سپس اگر رابطه خواهر/برادری یا مادری یا پدری داشته باشند، رابطهی دشمنی را حذف می‌کنیم.



```

1 match (c1:Character)-[e:ENEMY]-(c2:Character)
2 where (c1)-[:SIBLING]-(c2) or (c1)-[:HAS_MOTHER]-(c2) or (c1)-[:HAS_FATHER]-(c2)
3 delete e
    
```

Server is taking a long time to respond...

neo4j\$ match (c1:Character)-[e:ENEMY]-(c2:Character) where (c1)-[:SIBLING]-(c2) or ...

Deleted 2 relationships, completed after 49 ms.

Deleted 2 relationships, completed after 49 ms.

الف) نودهایی با لیبل "Character" که هم با نودهایی با لیبل "IN\_COUNTRY" رابطه‌ی "country" دارند و هم با نودهایی با لیبل "HAS\_OCCUPATION" رابطه‌ی "Occupation" و نام "swordfighter" دارند را می‌یابیم

```
neo4j$ match (country:Country)-[:IN_COUNTRY]-(person:Character)-[r:HAS_OCCUPATION]->(o:Occupation{name:'swordfighter'}) return country.name, count(person)
```

Server is taking a long time to respond.

"country.name"	"count(person)"
"Shire"	4
"Gondor"	1
"Rohan"	1

MAX COLUMN WIDTH:

| ب )

در این قسمت علاوه بر قسمت الف، خواهر برادر افراد را می‌یابیم. همین‌طور رابطه‌ی "IN\_COUNTRY" برای خواهر/برادر را می‌یابیم. سرزمین شمشیرزن‌هایی که رابطه‌ی "IN\_COUNTRY" ندارند را از روی سرزمین خواهر/برادر آن‌ها مشخص می‌کنیم.

```
neo4j$ match (country_sis_bro:Country)-[:IN_COUNTRY]-(sis_bro:Character)-[:SIBLING]-(person:Character)-[r:HAS_OCCUPATION]->(o:Occupation{name:'swordfighter'}) where not (person)-[:IN_COUNTRY]-(c:Country) return country_sis_bro, person
```

Server is taking a long time to respond.

"country_sis_bro"	"person"
{"name": "Gondor"}	{"name": "Boromir", "id": "Q219504", "gender": "male", "manner_of_death": "homicide", "url": "http://www.wikidata.org/entity/Q219504"}

MAX COLUMN WIDTH:

(۷)

در خط اول، تعداد شرونداں shire را پیدا کردم و سپس آن را در متغیر total قرار می‌دهم (چون از دو تا match استفاده می‌کنیم، باید این کار را انجام داد). سپس در خط دوم، تعداد انها‌یی که در رخداد خاص شرکت کردند را می‌یابیم.

```

1 match (person_total:Character)-[:IN_COUNTRY]-(c:Country {name:"Shire"}) with count(person_total) as Total
2 match (:Event{name:"the Council of Elrond"})-[:PARTICIPATED]-(person:Character)-[:IN_COUNTRY]-(c:Country {name:"Shire"})
3 return Total, count(person), tofloat(count(person)) / tofloat(Total) * 100 as percent
    
```

Server is taking a long time to respond.

neo4j\$ match (person\_total:Character)-[:IN\_COUNTRY]-(c:Country {name:"Shire"}) with count(person\_total) as Total match (:Event{name:"the Council of Elrond"})-[:PARTICIPATED]-(person:Character)-[:IN\_COUNTRY]-(c:Country {name:"Shire"}) return Total, count(person), tofloat(count(person)) / tofloat(Total) \* 100 as percent

Total	count(person)	percent
48	2	4.166666666666666

Started streaming 1 records after 3 ms and completed after 5 ms.

(۸)

برای سرزمین‌هایی که افراد آن در هیچ رخدادی شرکت نکردند، باید لیست خالی برگردانیم. بدین جهت از «optional match» استفاده کردم. در عکس صفحه‌ی بعد، نتایج به صورت کامل وجود دارد.

```

match (c:Country)
optional match (c)-[:IN_COUNTRY]-(person:Character)-[:PARTICIPATED]-(e:Event)
return c.name as country ,collect(distinct(e.name)) as event, collect(distinct(person.name)) as person ,count(person) order by c.name
    
```

Server is taking a long time to respond.

neo4j\$ match (c:Country) optional match (c)-[:IN\_COUNTRY]-(person:Character)-[:PARTICIPATED]-(e:Event) return c.name as country ,\_,event,person,count(person)

country	event	person	count(person)
"Arnor"	[]	[]	0
"Arthedain"	[]	[]	0
"Dale"	[]	[]	0
"Doriath"	[]	[]	0
"Gondolin"	[]	[]	0
"Gondor"	["Sauron's attack on Osgiliath","Faramir's defense of Osgiliath","Kin-strife"]	["Faramir","Castamir","Eldacar"]	4
"Lothlórien"	[]	[]	0
"Mordor"	[]	[]	0
"Moria"	[]	[]	0
"Nargothrond"	[]	[]	0

MAX COLUMN WIDTH: 1000

The screenshot shows a browser window with several tabs open, including "Install Cassandra on Win", "python version 3.6.5", "OPTIONAL MATCH - Ne...", "Home - Neo4j Sandbox", "neo4j@bolt+ssl/b542ed", and "OPTIONAL MATCH - Ne...". The main content area displays a query in the Neo4j Cypher language:

```

1 match (c:Country)
2 optional match (c)-[:IN_COUNTRY]-(person:Character)-[:PARTICIPATED]-(e:Event)
3 return c.name as country ,collect(distinct(e.name)) as event, collect(distinct(person.name)) as person ,count(person) order by
4 c.name

```

A yellow status bar at the bottom indicates: "Server is taking a long time to respond". Below the query, the results are presented in a table:

"country"	"event"	"person"	"count(person)"
"Arnor"	[]	[]	0
"Arthedain"	[]	[]	0
"Dale"	[]	[]	0
"Doriath"	[]	[]	0
"Gondolin"	[]	[]	0
"Gondor"	["Sauron's attack on Osgiliath", "Faramir's defense of Osgiliath", "Kin-strife"]	["Faramir", "Castamir", "Eldacar"]	4
"Lothlórien"	[]	[]	0
"Mordor"	[]	[]	0
"Moria"	[]	[]	0
"Nargothrond"	[]	[]	0

On the left side of the interface, there are buttons for "Table" (selected), "Text", and "Code". At the bottom right, there is a "MAX COLUMN WIDTH:" slider.

## بخش سوم

### گام دوم

(۱)

برای ساخت table از کوئری زیر استفاده کردم:

```
CREATE TABLE part_one(track_id int, title_track text, date_released text, title_album  
text, artist text, duration int, genre text, favorites_artist int, listens_track int, favorites_track  
int, listens_album int, favorites_album int, PRIMARY KEY(title_album , title_track));
```

سپس با دستور کپی داده‌ها را از فایل csv وارد جدول می‌کنیم:

```
COPY part_one(track_id, title_track, date_released, title_album, artist, duration, genre,  
favorites_artist, listens_track, favorites_track, listens_album, favorites_album) FROM  
'fma_dataset.csv' WITH DELIMITER=',' AND HEADER=TRUE;
```

زمان وارد کردن داده‌ها در عکس زیر آمده است:

```
Processed: 106574 rows; Rate: 15505 rows/s; Avg. rate: 15490 rows/s  
105159 rows imported from 1 files in 6.882 seconds (0 skipped).  
cqlsh:fma>
```

سپس مطلوب سوال و زمان اجرای کوئری به کمک کوئری زیر بدست می‌آید:

```
cqlsh:fma> TRACING ON  
Now Tracing is enabled  
cqlsh:fma> SELECT title_TRACK FROM part_one where title_album='Rumble, Young Man, Rumble';  
title_track  
-----  
I Listen 2 Classical (Featuring Slease)  
I'm Not Lyin' (Featuring Octavion Xcellence)  
No Title For It (Featuring Ohbliv and Gordy Michael)  
Out Of Here (Featuring Mic Jordan)  
Primetime Bandit (Remix) (Featuring Draizeq)  
Request (Featuring Barcodez and SamSun of Photosynthesizers)  
Scream Out (Featuring Braintrust)  
Some People Never Learn (Featuring NOTE)  
Take One (Featuring Joey Rippes)  
These Times (Featuring Slease and Joey Rippes)  
This Is Madness (Just Plain Black) (Featuring Black Liquid)  
Way Back When (Featuring Chuck D)  
(12 rows)  
Tracing session: f33a94f0-bbcb-11eb-8fdd-d9c8370df47b  
activity | timestamp | source | sou  
-----  
Execute CQL3 query | 2021-05-23 18:06:59.842000 | 127.0.0.1 |  
Parsing SELECT title_TRACK FROM part_one where title_album='Rumble, Young Man, Rumble'; [Native-Transport-Requests-1] | 2021-05-23 18:06:59.850000 | 127.0.0.1 |  
Preparing statement [Native-Transport-Requests-1] | 2021-05-23 18:06:59.850000 | 127.0.0.1 |  
Executing single-partition query on part_one [ReadStage-3] | 2021-05-23 18:06:59.851000 | 127.0.0.1 |  
Acquiring sstable references [ReadStage-3] | 2021-05-23 18:06:59.851000 | 127.0.0.1 |  
Skipped 0/0 non-slice-intersecting sstables, included 0 due to tombstones [ReadStage-3] | 2021-05-23 18:06:59.851000 | 127.0.0.1 |  
Merged data from memtables and 0 sstables [ReadStage-3] | 2021-05-23 18:06:59.851000 | 127.0.0.1 |  
Read 12 live rows and 0 tombstone cells [ReadStage-3] | 2021-05-23 18:06:59.852000 | 127.0.0.1 |  
Request complete | 2021-05-23 18:06:59.853226 | 127.0.0.1 |
```

جهت دریافت زمان اجرای کوئری از TRACING ON استفاده شده است (مدت زمان 0.011226 ثانیه است).

(۲)

در این قسمت کلیدهای اساسی چنین تعریف شده اند:

### PRIMARY KEY(artist , title\_track)

و کوئری زده شده چنین است:

**TRACING ON; SELECT artist, title\_track, genre FROM part\_two where artist='RoccoW';**

توجه داریم که در کاساندرا مقدار null وجود ندارد و برای کلیدهایی که مقداری ندارند، به کاربران برای فهم بهتر null نشان می‌دهند. از این باب، کلیدهایی که مقدار null دارند، نمی‌توانند به عنوان primary key تعریف شوند.

خروجی مورد نظر و زمان اجرا:

Now Tracing is enabled		genre
RoccoW	Accidental Skwugg	null
RoccoW	Ace of Spades	null
RoccoW	Bang!	null
RoccoW	Blarugh	Electronic
RoccoW	Bleeps Galore	null
RoccoW	Braadslee	Electronic
RoccoW	Break-A-Leg	Electronic
RoccoW	Chiphlo Instrumental	Electronic
RoccoW	Chips Got Kicks	Electronic
RoccoW	Create A Song in A Day (Extended)	Electronic
RoccoW	DM1 + LSD1 Jam Session	Electronic
RoccoW	Egyptian Swaggah	null
RoccoW	Electric Donkey Muscles	null
RoccoW	Famifunk Over Breakfast	null
RoccoW	Fuck Sidechain Compression on Gameboy	null
RoccoW	Fuckabotz	Electronic
RoccoW	Hello (Chiptune Style)	Electronic
RoccoW	Ideetje (Extended)	Electronic
RoccoW	Lady Bad Luck (VRCG)	Electronic
RoccoW	Let's Start Out Slow	Electronic
RoccoW	Messiah	Electronic
RoccoW	Out of Sight, Into the Mind	Electronic
RoccoW	Pumped	null
RoccoW	Sea Battles	Electronic
RoccoW	Something, Something, Knight	Electronic
RoccoW	Sweet Self Satisfaction	null
RoccoW	SwingJedding	null
RoccoW	Try A Music	null
RoccoW	Vapperrweef	Electronic
RoccoW	Vengaboy - Boom Boom Boom Boom! (LSD1 Remix)	Electronic
RoccoW	Weather	Electronic
RoccoW	Weeklybeats 2014 #12 - xyce xycel baby	Electronic
RoccoW	Weeklybeats 2014 #1 - KORG Woid Jam	Electronic
RoccoW	Weeklybeats 2014 #10 - Trap'd in A DS	Electronic
RoccoW	Weeklybeats 2014 #2 - Daniel's Kruis	Electronic
RoccoW	Weeklybeats 2014 #3 - Play Nicely	Electronic
RoccoW	Weeklybeats 2014 #4 - All Will Be Well	Electronic
RoccoW	Weeklybeats 2014 #5 - I've Got Nothing	Electronic
RoccoW	Weeklybeats 2014 #7 - Freaking Likid	Electronic
RoccoW	Weeklybeats 2014 #9 - This Little Piggy Discreet	Electronic
RoccoW	Welcome!	null
RoccoW	lalalalala lala lala	Electronic
RoccoW	wbdffbgkg	null
RoccoW	xyce - Quelle Surprise (VRG6 Remix)	Electronic

activity	timestamp	source	source_elapsed	client
Execute CQL3 query	2021-05-25 13:06:31.428000	127.0.0.1	0	127.0.0.1
Parsing SELECT artist, title_track, genre FROM part_two where artist='RoccoW'	[Native-Transport-Requests-1]	2021-05-25 13:06:31.430000	127.0.0.1	852
Preparing statement [Native-Transport-Requests-1]	2021-05-25 13:06:31.430000	127.0.0.1	8859	127.0.0.1
Executing single-partition query on part_two [ReadStage-2]	2021-05-25 13:06:31.437000	127.0.0.1	9331	127.0.0.1
Acquiring sstable references [ReadStage-2]	2021-05-25 13:06:31.437000	127.0.0.1	9496	127.0.0.1
Skipped 0/0 non-slice-intersecting stables, included 0 due to tombstones [ReadStage-2]	2021-05-25 13:06:31.437000	127.0.0.1	9681	127.0.0.1

(۳) در این قسمت ابتدا به کمک پایتون، ستون year را به فایل csv اضافه کردم و سپس به کمک کوئری زیر خروجی بدست می‌آید.

TRACING ON; SELECT title\_track, duration, genre, year FROM part\_three where genre='Hip-Hop' AND year in ('2015', '2016') AND duration < 180;

خروجی: ۱۰۵ آهنگ به عنوان خروجی نمایش داده شد.

title_track	duration	genre	year
Gimme That Tape!	5	Hip-Hop	2015
Quando O Universo Explode	27	Hip-Hop	2015
pathetic at best	31	Hip-Hop	2015
Hey, Guys	37	Hip-Hop	2015
Vamos Perguntar Já Plateia	38	Hip-Hop	2015
Homie Jota	41	Hip-Hop	2015
Aquele Polvo	57	Hip-Hop	2015
The 1 Upper (84 Mixx)	60	Hip-Hop	2015
Sara Sampaio	61	Hip-Hop	2015
The 1 Upper (BreakBeat Mixx)	73	Hip-Hop	2015
She's a man?	75	Hip-Hop	2015
These Dreams And Shit	93	Hip-Hop	2015
Whentro	97	Hip-Hop	2015
Afterschool Speshul Bonus Beats	101	Hip-Hop	2015
54%	109	Hip-Hop	2015
Simon Pegg	116	Hip-Hop	2015
prometheus	122	Hip-Hop	2015
Hey You Guys (Sloth Cause Mixx)	131	Hip-Hop	2015
Bird Up!	135	Hip-Hop	2015
Footprints	140	Hip-Hop	2015
Pop Tart	141	Hip-Hop	2015
Blue Steel (Instrumental)	143	Hip-Hop	2015
Spot Rockers (Gotcha Man Mix)	144	Hip-Hop	2015
Dragão Azul, Dragão Vermelho	148	Hip-Hop	2015
The 1 Upper (Do Me Remix)	150	Hip-Hop	2015
Penny17	152	Hip-Hop	2015
Hey You Guys (Acapella)	153	Hip-Hop	2015
Tonight You Belong To Me (Featuring Kat Wahamaa)	155	Hip-Hop	2015
Thrashin	156	Hip-Hop	2015
Spot Rockers (Doc's 2-Live Mixx)	157	Hip-Hop	2015
Spot Rockers (Blank Mixx)	158	Hip-Hop	2015
PS Love Always	159	Hip-Hop	2015
crumbs	163	Hip-Hop	2015
Poetikal Refugee ft. Mikey Krumins aka Dyems	168	Hip-Hop	2015
Eat Some Cake	169	Hip-Hop	2015
Homie Stray	170	Hip-Hop	2015
C-N-P Enuff	175	Hip-Hop	2015
goldheart	176	Hip-Hop	2015
Gruta	177	Hip-Hop	2015
Welcome To Chi-Raq Intro	30	Hip-Hop	2016
Welcome To Chi-Raq Outtro	42	Hip-Hop	2016
Drugs By Roman Candlelight	57	Hip-Hop	2016
Phon's Waltz	62	Hip-Hop	2016
Dead Homie Intro	64	Hip-Hop	2016
Intro	67	Hip-Hop	2016
Intro	70	Hip-Hop	2016
drunkenCPU	82	Hip-Hop	2016
Fireheart	84	Hip-Hop	2016
Needy Ass	85	Hip-Hop	2016
Cawper Crow	87	Hip-Hop	2016
Mello Hello	100	Hip-Hop	2016
Money Beets: A Tribute to RHP (Featuring Milkdrop)	104	Hip-Hop	2016
activate	106	Hip-Hop	2016
Suggestion Rocks	109	Hip-Hop	2016
Pusher For Record	111	Hip-Hop	2016
Daddy's Drinkin' Song	113	Hip-Hop	2016
Souvenirs, Novelties, Party Tricks	114	Hip-Hop	2016
Creepin' On Your Shh	116	Hip-Hop	2016

ادامه خروجی و زمان اجرا در عکس زیر قابل مشاهده است:

...NONE...	duration	genre	year		timestamp	source	source_elapsed
title_track							
Fallin' Apart	175	Hip-Hop	2016				
Love	172	Hip-Hop	2016				
Idle Hands	177	Hip-Hop	2016				
The Raps Well (Album Version Acapella)	178	Hip-Hop	2016				
Vintage featuring The Honorable Slezee	179	Hip-Hop	2016				
{105 rows}							
Tracing session: 5dfcd80-bd5b-11eb-ae72-7fa203b78e52							
activity							
client							
-----							
127.0.0.1					Execute CQL query   2021-05-25 17:46:08.152000   127.0.0.1   0		
Parsing SELECT title_track, duration, genre, year FROM part_three where genre='Hip-Hop' AND year in ('2015', '2016') AND duration < 180; [Native-Transport-Requests-1]					2021-05-25 17:46:08.152000   127.0.0.1   122		
127.0.0.1					Preparing statement [Native-Transport-Requests-1]   2021-05-25 17:46:08.152000   127.0.0.1   259		
127.0.0.1					Executing single-partition query on part_three [ReadStage-3]   2021-05-25 17:46:08.154000   127.0.0.1   1700		
127.0.0.1					Acquiring sstable references [ReadStage-3]   2021-05-25 17:46:08.154000   127.0.0.1   1754		
127.0.0.1					Skipped 0/0 non-slice-intersecting sstables, included 0 due to tombstones [ReadStage-3]   2021-05-25 17:46:08.154000   127.0.0.1   1820		
127.0.0.1					Merged data from memtables and 0 sstables [ReadStage-3]   2021-05-25 17:46:08.154000   127.0.0.1   2056		
127.0.0.1					Read 39 live rows and 0 tombstone cells [ReadStage-3]   2021-05-25 17:46:08.154000   127.0.0.1   2101		
127.0.0.1					Executing single-partition query on part_three [ReadStage-3]   2021-05-25 17:46:08.156000   127.0.0.1   3671		
127.0.0.1					Acquiring sstable references [ReadStage-3]   2021-05-25 17:46:08.156000   127.0.0.1   3738		
127.0.0.1					Skipped 0/0 non-slice-intersecting sstables, included 0 due to tombstones [ReadStage-3]   2021-05-25 17:46:08.156000   127.0.0.1   3774		
127.0.0.1					Merged data from memtables and 0 sstables [ReadStage-3]   2021-05-25 17:46:08.156000   127.0.0.1   4064		
127.0.0.1					Read 61 live rows and 0 tombstone cells [ReadStage-3]   2021-05-25 17:46:08.156000   127.0.0.1   4106		
127.0.0.1					Request complete   2021-05-25 17:46:08.156378   127.0.0.1   4378		

(۴) مشابهه قسمت قبل، ستون month را به فایل csv اضافه می‌کنیم و سپس به کمک کوئری زیر خروجی بدست می‌آید:

TRACING ON; SELECT title\_track, genre, month, listens\_track FROM part\_four where genre in ('Electronic', 'Pop') AND month='Apr' AND listens\_track > 300;

خروجی: ۲۷۳ آهنگ به عنوان خروجی چاپ شد. به دلیل زیاد بودن تعداد خروجی ها، از نمایش همهی آنها در اینجا صرف نظر می‌کنم و تعدادی از آنها را در دو عکس نمایش می‌دهم:

```
cqlsh:fma> TRACING ON; SELECT title_track, genre, month, listens_track FROM part_four WHERE genre IN ('Electronic', 'Pop') AND month='Apr' AND listens_track > 300;
Tracing is already enabled. Use TRACING OFF to disable.
```

title_track	genre	month	listens_track
wobbles	Electronic	Apr	381
Rough Sex	Electronic	Apr	302
Peloponnes	Electronic	Apr	309
Off In The Distance	Electronic	Apr	315
Low on Ideas	Electronic	Apr	317
Umbrella Pete Strut	Electronic	Apr	322
Candy Funk	Electronic	Apr	326
Once Upon A Time	Electronic	Apr	329
Marathon	Electronic	Apr	332
Vodka Bitter Smartie Party (Ergo Phizmiz' version)	Electronic	Apr	340
So I Went Home	Electronic	Apr	343
Slow jam	Electronic	Apr	346
Pick Your Brain	Electronic	Apr	353
Chocolate Sky	Electronic	Apr	356
Amorpheus (zbz reverb)	Electronic	Apr	358
I don't know	Electronic	Apr	363
Bilboquet (La Mélodie Jetable)	Electronic	Apr	370
Mr. Pig & Mr. Chrome	Electronic	Apr	380
Goodbye	Electronic	Apr	382
5.7	Electronic	Apr	388
Nel Vuoto	Electronic	Apr	391
Freaks	Electronic	Apr	415
Rising in the Wake of Supernova	Electronic	Apr	422
Dismantling Earth Based Authority Via Satellite	Electronic	Apr	423
Track 3	Electronic	Apr	428
Nolan Thing 3	Electronic	Apr	438
Nausea	Electronic	Apr	443
[Track 2]	Electronic	Apr	445
Fabric	Electronic	Apr	450
I	Electronic	Apr	451
mighty nice	Electronic	Apr	453
Sleep Just To Dream	Electronic	Apr	460
Straight Blimpin'	Electronic	Apr	467
Sluni	Electronic	Apr	478
Vodka Alone (Vernon's version)	Electronic	Apr	479
Zerr	Electronic	Apr	481
Midget Madness	Electronic	Apr	483
Coz	Electronic	Apr	485
Spot	Electronic	Apr	493
Untitled3	Electronic	Apr	499
Time: disabled (Step by Step edit)	Electronic	Apr	502
Absolute	Electronic	Apr	513
N N R M L Z M	Electronic	Apr	518
Save, Develop, Protect	Electronic	Apr	519
Track 1	Electronic	Apr	521
Janet's Voice	Electronic	Apr	523
Sogno D'Autunno	Electronic	Apr	527
Santa Claus	Electronic	Apr	530
Greeneheys	Electronic	Apr	531
you often ebb	Electronic	Apr	534
I Put A Spell On You	Electronic	Apr	536
green butter	Electronic	Apr	539
Light	Electronic	Apr	540
chicken factory	Electronic	Apr	551
Empty Field	Electronic	Apr	559
Snail! Snail! Rocknroll	Electronic	Apr	567
Putting Spells On Birds	Electronic	Apr	570
Landing	Electronic	Apr	573
Face Distortion	Electronic	Apr	575

```
c:\Windows\System32\cmd.exe - cqlsh
    Moi Samletry | Pop | Apr | 635
    Our Sorrows | Pop | Apr | 663
    Stinger Sisters | Pop | Apr | 703
    Brute | Pop | Apr | 704
probably digging my own grave | Pop | Apr | 710
    Nevesta | Pop | Apr | 714
When The Sea Called Me | Pop | Apr | 722
    Snelliss | Pop | Apr | 726
    Hey, Predator Drone | Pop | Apr | 824
    Nevesta (Flayve Remix) | Pop | Apr | 855
    Giraffes | Pop | Apr | 872
    Betsy On The Roof | Pop | Apr | 880
    Brute (Intro) | Pop | Apr | 972
    Discount Store | Pop | Apr | 999
Study For A Pop Band | Pop | Apr | 1037
that face you wear | Pop | Apr | 1132
    Malta | Pop | Apr | 1166
    Mirrors | Pop | Apr | 1207
    Palm Springs | Pop | Apr | 1432
    Alright | Pop | Apr | 2010
    Trouble | Pop | Apr | 2079
    An Island | Pop | Apr | 2093
    I Feel | Pop | Apr | 2582

(273 rows)

Tracing session: 2ed197d00-bd60-11eb-ae72-7fa203b78e52
activity
ce_elapsed | client
-----+-----+
0 | 127.0.0.1
108 | 127.0.0.1
287 | 127.0.0.1
1888 | 127.0.0.1
1925 | 127.0.0.1
2036 | 127.0.0.1
2522 | 127.0.0.1
2641 | 127.0.0.1
3124 | 127.0.0.1
timestamp
-----+-----+
Execute CQL3 query | 2021-05-25 18:20:36
Preparing statement [Native-Transport-Requests-1] | 2021-05-25 18:20:36
Executing single-partition query on part_four [ReadStage-3] | 2021-05-25 18:20:36
Acquiring sstable references [ReadStage-3] | 2021-05-25 18:20:36
Skipped 0/0 non-slice-intersecting sstables, included 0 due to tombstones [ReadStage-3] | 2021-05-25 18:20:36
Merged data from memtables and 0 sstables [ReadStage-3] | 2021-05-25 18:20:36
Read 100 live rows and 0 tombstone cells [ReadStage-3] | 2021-05-25 18:20:36
Request complete | 2021-05-25 18:20:36
```

(۷) متوسط طول آهنگ‌ها به کمک کوئری زیر محاسبه شده است:

`SELECT AVG (duration) FROM part_seven;`

و خروجی بدین شکل است:

```
cqlsh:fma> SELECT AVG (duration) FROM part_seven;

system.avg(duration)
-----
279

(1 rows)

Warnings:
```

کوئری و خروجی برای ۱۰ آهنگی که بیشترین طول را دارند، از این قرارند:

```
cqlsh:fma> SELECT title_track, duration FROM part_seven_2 where temp=1 ORDER BY duration DESC LIMIT 10;

title_track | duration
-----
Harmonia Live at ATP 08 part 1 | 18350
In the Garden | 11030
Adam Are You Free? | 11016
RAPE OF NANKING | 10999
Orrrrgaaanooone | 7372
OceanTapping | 7320
rad[Incidental Music]ius 0025 | 3718
Elliott Sharp - ISSUE Podcast | 3717
Passio Domini Nostri Jesu Christe | 3716
To Our Yes | 3714

(10 rows)
cqlsh:fma>
```

(۸) کوئری، خروجی و زمان اجرا در عکس قابل مشاهده است.

```
cqlsh:fma> TRACING ON; SELECT title_track, year, genre, favorites_album, favorites_artist FROM part_eight where year='2016' AND genre='Rock' ORDER BY favorites_album DESC;
Tracing session: 1c53a3d0-be1b-11eb-8de2-2f5f7f1331e6

title_track | year | genre | favorites_album | favorites_artist
-----
Transmit | 2016 | Rock | 3 | 23
Medo de Tentar | 2016 | Rock | 2 | 3
Warm Waveform | 2016 | Rock | 1 | 2
You Look Unwell | 2016 | Rock | 0 | 6

(4 rows)

Tracing session: 1c53a3d0-be1b-11eb-8de2-2f5f7f1331e6

activity | timestamp | source | source_elapsed | client
-----+-----+-----+-----+-----+
3 query | 2021-05-26 16:38:41.363000 | 127.0.0.1 | 0 | 127.0.0.1
| Parsing SELECT title_track, year, genre, favorites_album, favorites_artist FROM part_eight where year='2016' AND genre='Rock' ORDER BY favorites_album DESC; [Native-Transport-Requests-1] | 2021-05-26 16:38:41.364000 | 127.0.0.1 | 2380 | 127.0.0.1
| Preparing statement [Native-Transport-Requests-1] | 2021-05-26 16:38:41.365000 | 127.0.0.1 | 2903 | 127.0.0.1
| Executing single-partition query on part_eight [Reads-3] | 2021-05-26 16:38:41.365000 | 127.0.0.1 | 3313 | 127.0.0.1
| Acquiring sstable references [Reads-3] | 2021-05-26 16:38:41.365000 | 127.0.0.1 | 3505 | 127.0.0.1
| Skipped 0/0 non-slice-intersecting stables, included 0 due to tombstones [Reads-3] | 2021-05-26 16:38:41.366000 | 127.0.0.1 | 3715 | 127.0.0.1
| Merged data from memtables and 0 stables [Reads-3] | 2021-05-26 16:38:41.366000 | 127.0.0.1 | 3811 | 127.0.0.1
| Read 4 live rows and 0 tombstone cells [Reads-3] | 2021-05-26 16:38:41.366000 | 127.0.0.1 | 3910 | 127.0.0.1
| Request complete | 2021-05-26 16:38:41.367125 | 127.0.0.1 | 4125 | 127.0.0.1
```

(۹) برای حل این قسمت، دو جدول تعریف کردم. از آنجایی که در کاساندرا امکان سورت کردن روی خروجی counter وجود ندارد، خروجی کوئری table اول را در یک فایل csv موقت ذخیره کردم و سپس یک table دوم تعریف کردم و داده‌ها را از فایل csv موقت وارد table دوم کردم. تمامی کوئری هایی که مورد استفاده قرار گرفتن در عکس زیر قابل مشاهده است:

```

CREATE TABLE part_nine(track_id int, title_track text, date_released text, title_album text, artist text, duration int, genre text
COPY part_nine(track_id, title_track, date_released, title_album, artist, duration, genre, favorites_artist, listens_track, favori
#using capture to write output as csv file in temp_9:
|
CAPTURE; CAPTURE 'temp_9.csv'; SELECT genre, COUNT(listens_track) as counter, temp FROM part_nine GROUP BY genre;
CAPTURE off;

# second table:
CREATE TABLE part_nine_2(genre text, counter int, temp int, PRIMARY KEY(temp, counter ));

COPY part_nine_2(genre, counter, temp ) FROM 'temp_9.csv' WITH DELIMITER='|' AND HEADER=TRUE;
select genre, counter from part_nine_2 WHERE temp=1 ORDER BY counter DESC;

```

: خروجی

```
cqlsh:fma> select genre, counter from part_nine_2 WHERE temp=1 ORDER BY counter DESC;
```

genre	counter
Electronic	3702
Rock	2876
Hip-Hop	1897
Experimental	1873
Instrumental	1789
Folk	1438
Pop	1399
International	1165
Classical	1072
Jazz	538
Old-Time / Historic	521
Spoken	368
Country	193
Soul-RnB	171
Blues	99
Easy Listening	24

(16 rows)

## بخش چهارم

### نصب الستیک سرچ:

The terminal window shows Elasticsearch logs from May 12, 2021, at 05:12:11. It details the creation of an index named 'ilm-history' with shards [1/1] and documents [0]. It also shows the 'branch-check-unfollow-prerequisites' phase transitioning to 'rollover' and 'check-rollover-ready'. The browser window shows the Elasticsearch Dev Tools interface at localhost:9200, specifically the 'Monitoring' tab.

```
...[{"name": "DESKTOP-V1L9J0V", "cluster_name": "elasticsearch", "cluster_uuid": "CRW9jceWRNS8rtzR2cy5Lg", "version": {"number": "7.12.1", "build_flavor": "default", "build_type": "zip", "build_hash": "31868371399bc6b6d23c3200870651f10d3343b7", "build_date": "2021-04-20T20:56:39.048Z", "build_snapshot": false, "lucene_version": "8.8.0", "minimum_wire_compatibility_version": "6.8.0", "minimum_index_compatibility_version": "6.0.0-beta"}, "tagline": "You Know, for Search"}]
```

### نصب کیبانا:

The terminal window shows Kibana logs from May 12, 2021, at 05:29:41. It details the migration of saved objects from 'production' cluster to 'kibana' cluster, including the creation of new targets and the completion of the migration process. The browser window shows the Kibana Dev Tools interface at localhost:5601, specifically the 'Monitoring' tab.

### ورود داده ها:

The screenshot shows a POST request to '/movies/\_doc/1' with the following JSON payload:

```
1 POST /movies/_doc/1
2 {
3   "title": "The Godfather",
4   "director": "Francis Ford
Coppola",
5   "year": 1972
6 }
7 PUT movies/_doc/2
8 {
9   "title": "Lawrence of
Arabia",
10  "director": "David Lean",
11  "year": 1962,
12  "genres": ["Adventure",
13    "Biography", "Drama"]
```

The response shows the updated document with '\_id': '1', '\_version': 11, and '\_index': 'movies'. The response status is 200 - OK and the time taken is 70 ms.

جستجو فیلم هایی که کلمه kill در آنها وجود دارد:

```

History Settings Help
1 POST /movies/_doc/1
2 (DOC)
3 PUT movies/_doc/2
4 (DOC)
5 PUT movies/_doc/3
6 (DOC)
7 PUT movies/_doc/4
8 (DOC)
9 PUT movies/_doc/5
10 (DOC)
11 PUT movies/_doc/6
12 (DOC)
13 PUT movies/_doc/7
14 (DOC)
15 PUT movies/_doc/8
16 (DOC)
17 PUT movies/_doc/9
18 (DOC)
19 PUT movies/_doc/10
20 (DOC)
21 PUT movies/_doc/11
22 (DOC)
23 PUT movies/_doc/12
24 (DOC)
25 PUT movies/_doc/13
26 (DOC)
27 PUT movies/_doc/14
28 (DOC)
29 PUT movies/_doc/15
30 (DOC)
31 PUT movies/_doc/16
32 (DOC)
33 PUT movies/_doc/17
34 (DOC)
35 PUT movies/_doc/18
36 (DOC)
37 PUT movies/_doc/19
38 (DOC)
39 PUT movies/_doc/20
40 (DOC)
41 PUT movies/_doc/21
42 (DOC)
43 PUT movies/_doc/22
44 (DOC)
45 PUT movies/_doc/23
46 (DOC)
47 PUT movies/_doc/24
48 (DOC)
49 PUT movies/_doc/25
50 (DOC)
51 PUT movies/_doc/26
52 POST /_search
53 {
54   "query": {
55     "query_string": {
56       "query": "kill"
57     }
58   }
59 }
60
61
62
63
64
65
66
67
68
69
70
71
72

```

200 - OK | 27 ms

```

1 #! this request accesses system indices: [.apm-agent-configuration, .apm-custom-link, .kibana.7.12.1.001, .kibana-task-manager.7.12.1.001], but in a future major version, direct access to system indices will be prevented by default
2 {
3   "took": 3,
4   "timed_out": false,
5   "_shards": {
6     "total": 6,
7     "successful": 6,
8     "skipped": 0,
9     "failed": 0
10   },
11   "hits": [
12     {
13       "total": {
14         "value": 3,
15         "relation": "eq"
16       },
17       "max_score": 0.7404973,
18       "hits": [
19         {
20           "_index": "movies",
21           "_type": "movie",
22           "_id": "3",
23           "_score": 0.7404973,
24           "_source": {
25             "title": "To Kill a Mockingbird",
26             "director": "Robert Mulligan",
27             "year": 1962,
28             "genres": [
29               "Crime",
30               "Drama",
31               "Mystery"
32             ]
33           }
34         }
35       ],
36       "_index": "movies",
37     }
38   ]
39 }

```

جستجو فیلم هایی که کلمه kill یا ford در آنها وجود دارد:

```

History Settings Help
1 POST /movies/_doc/1
2 (DOC)
3 PUT movies/_doc/2
4 (DOC)
5 PUT movies/_doc/3
6 (DOC)
7 PUT movies/_doc/4
8 (DOC)
9 PUT movies/_doc/5
10 (DOC)
11 PUT movies/_doc/6
12 (DOC)
13 PUT movies/_doc/7
14 (DOC)
15 PUT movies/_doc/8
16 (DOC)
17 PUT movies/_doc/9
18 (DOC)
19 PUT movies/_doc/10
20 (DOC)
21 PUT movies/_doc/11
22 (DOC)
23 PUT movies/_doc/12
24 (DOC)
25 PUT movies/_doc/13
26 (DOC)
27 PUT movies/_doc/14
28 (DOC)
29 PUT movies/_doc/15
30 (DOC)
31 PUT movies/_doc/16
32 (DOC)
33 PUT movies/_doc/17
34 (DOC)
35 PUT movies/_doc/18
36 (DOC)
37 PUT movies/_doc/19
38 (DOC)
39 PUT movies/_doc/20
40 (DOC)
41 PUT movies/_doc/21
42 (DOC)
43 PUT movies/_doc/22
44 (DOC)
45 PUT movies/_doc/23
46 (DOC)
47 PUT movies/_doc/24
48 (DOC)
49 PUT movies/_doc/25
50 (DOC)
51 PUT movies/_doc/26
52 POST /_search
53 {
54   "query": {
55     "query_string": {
56       "query": "kill OR ford",
57       "fields": ["title"]
58     }
59   }
60 }
61
62
63
64
65
66
67
68
69
70
71
72

```

200 - OK | 31 ms

```

1 #! this request accesses system indices: [.apm-agent-configuration, .apm-custom-link, .kibana.7.12.1.001, .kibana-task-manager.7.12.1.001], but in a future major version, direct access to system indices will be prevented by default
2 {
3   "took": 8,
4   "timed_out": false,
5   "_shards": {
6     "total": 6,
7     "successful": 6,
8     "skipped": 0,
9     "failed": 0
10   },
11   "hits": [
12     {
13       "total": {
14         "value": 4,
15         "relation": "eq"
16       },
17       "max_score": 1.211086,
18       "hits": [
19         {
20           "_index": "movies",
21           "_type": "movie",
22           "_id": "4",
23           "_score": 1.211086,
24           "_source": {
25             "title": "The Assassination of Jesse James by the Coward Robert Ford",
26             "director": "Andrew Dominik",
27             "year": 2007,
28             "genres": [
29               "Biography",
30               "Crime",
31               "Drama"
32             ]
33           }
34         }
35       ],
36       "_index": "movies",
37     }
38   ]
39 }

```

ساخت درخت جستجو:

فیلم های با ژانر drama را برمیگرداند

```

History Settings Help
1 # inserting data
2 POST /movies/_doc/1
3 (DOC)
4 PUT movies/_doc/2
5 (DOC)
6 PUT movies/_doc/3
7 (DOC)
8 PUT movies/_doc/4
9 (DOC)
10 PUT movies/_doc/5
11 (DOC)
12 PUT movies/_doc/6
13 (DOC)
14 PUT movies/_doc/7
15 (DOC)
16 PUT movies/_doc/8
17 (DOC)
18 PUT movies/_doc/9
19 (DOC)
20 PUT movies/_doc/10
21 (DOC)
22 PUT movies/_doc/11
23 (DOC)
24 PUT movies/_doc/12
25 (DOC)
26 PUT movies/_doc/13
27 (DOC)
28 PUT movies/_doc/14
29 (DOC)
30 PUT movies/_doc/15
31 (DOC)
32 PUT movies/_doc/16
33 (DOC)
34 PUT movies/_doc/17
35 (DOC)
36 PUT movies/_doc/18
37 (DOC)
38 PUT movies/_doc/19
39 (DOC)
40 PUT movies/_doc/20
41 (DOC)
42 PUT movies/_doc/21
43 (DOC)
44 PUT movies/_doc/22
45 (DOC)
46 PUT movies/_doc/23
47 (DOC)
48 PUT movies/_doc/24
49 (DOC)
50 PUT movies/_doc/25
51 (DOC)
52 # "kill" in title
53 POST /_search
54 (DOC)
55
56
57
58
59
60
61
62
63 # "kill" or "ford" in title
64 POST /_search
65 (DOC)
66
67
68
69
70
71
72
73
74 # getting all drama genres
75 POST /movies/_search
76 {
77   "query": {
78     "match": {
79       "genres": "drama"
80     }
81   }
82 }

```

200 - OK | 44 ms

```

3   "took": 0,
4   "timed_out": false,
5   "_shards": {
6     "total": 1,
7     "successful": 1,
8     "skipped": 0,
9     "failed": 0
10   },
11   "hits": [
12     {
13       "total": {
14         "value": 4,
15         "relation": "eq"
16       },
17       "max_score": 0.65585554,
18       "hits": [
19         {
20           "_index": "movies",
21           "_type": "movie",
22           "_id": "4",
23           "_score": 0.65585554,
24           "_source": {
25             "title": "Apocalypse Now",
26             "director": "Francis Ford Coppola",
27             "year": 1979,
28             "genres": [
29               "Drama",
30               "War"
31             ]
32           }
33         }
34       ],
35       "_index": "movies",
36       "_type": "movie",
37       "_id": "2",
38       "_score": 0.56383127,
39       "_source": {
40         "title": "Lawrence of Arabia",
41       }
42     }
43   ]
44 }

```

فیلم هایی که در title آنها "kill" یا "bill" وجود دارد

```

History Settings Help
9+ { }
10 PUT movies/_doc/3
11 | { }
12 PUT movies/_doc/4
13 | { }
14 PUT movies/_doc/5
15 | { }
16 PUT movies/_doc/6
17 | { }
18 PUT movies/_doc/7
19 | { }
20 PUT movies/_doc/8
21 | { }
22 PUT movies/_doc/9
23 | { }
24 PUT movies/_doc/10
25 | { }
26 PUT movies/_doc/11
27 | { }
28 PUT movies/_doc/12
29 | { }
30 PUT movies/_doc/13
31 | { }
32 PUT movies/_doc/14
33 | { }
34 PUT movies/_doc/15
35 | { }
36 PUT movies/_doc/16
37 | { }
38 PUT movies/_doc/17
39 | { }
40 PUT movies/_doc/18
41 | { }
42 PUT movies/_doc/19
43 | { }
44 PUT movies/_doc/20
45 | { }
46 PUT movies/_doc/21
47 | { }
48 PUT movies/_doc/22
49 | { }
50 PUT movies/_doc/23
51 # "kill" in title
52 POST /search
53 | { }
54 PUT movies/_doc/24
55 | { }
56 PUT movies/_doc/25
57 | { }
58 PUT movies/_doc/26
59 | { }
60 PUT movies/_doc/27
61 | { }
62 PUT movies/_doc/28
63 # "kill" or "ford" in title
64 POST /search
65 | { }
66 PUT movies/_doc/29
67 | { }
68 PUT movies/_doc/30
69 | { }
70 PUT movies/_doc/31
71 | { }
72 PUT movies/_doc/32
73 # getting all darama genres
74 POST /movies/movie/_search
75 | { }
76 PUT movies/_doc/33
77 | { }
78 PUT movies/_doc/34
79 | { }
80 PUT movies/_doc/35
81 | { }
82 PUT movies/_doc/36
83 # existing "kill" or "bill" in title
84 POST movies/_search
85 { }
86 "query": {
87     "multi_match": {
88         "query": "Kill Bill",
89         "operator": "or"
90     }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }

200 - OK | 40 ms
1+ [
2   "took" : 1,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 3,
13      "relation" : "eq"
14    },
15    "max_score" : 1.933216,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "7",
21        "score" : 1.933216,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 1.933216,
38      }
39    ]
40  }
41 }

200 - OK | 21 ms
1+ [
2   "took" : 1,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 3,
13      "relation" : "eq"
14    },
15    "max_score" : 3.63558,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "6",
21        "score" : 3.63558,
22        "source" : {
23          "title" : "The Assassination of Jesse James by the Coward Robert Ford",
24          "director" : "Andrew Dominik",
25          "year" : 2007,
26          "genres" : [
27            "Biography",
28            "Crime",
29            "Drama"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35      }
36    ]
37  }
38 }

200 - OK | 20 ms
1+ [
2   "took" : 0,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 3.0586002,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "5",
21        "score" : 3.0586002,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.0586002,
38      }
39    ]
40  }
41 }

200 - OK | 20 ms
1+ [
2   "took" : 0,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 3.0586002,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "5",
21        "score" : 3.0586002,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.0586002,
38      }
39    ]
40  }
41 }

```

سرچ کردن همزمان روی دو یا چند فیلد:

```

History Settings Help
29 PUT movies/_doc/5
30 | { }
31 PUT movies/_doc/6
32 | { }
33 PUT movies/_doc/7
34 | { }
35 PUT movies/_doc/8
36 | { }
37 PUT movies/_doc/9
38 | { }
39 PUT movies/_doc/10
40 | { }
41 PUT movies/_doc/11
42 | { }
43 PUT movies/_doc/12
44 | { }
45 PUT movies/_doc/13
46 | { }
47 PUT movies/_doc/14
48 | { }
49 PUT movies/_doc/15
50 | { }
51 PUT movies/_doc/16
52 | { }
53 PUT movies/_doc/17
54 | { }
55 PUT movies/_doc/18
56 | { }
57 PUT movies/_doc/19
58 | { }
59 PUT movies/_doc/20
60 | { }
61 PUT movies/_doc/21
62 | { }
63 PUT movies/_doc/22
64 | { }
65 PUT movies/_doc/23
66 | { }
67 PUT movies/_doc/24
68 | { }
69 PUT movies/_doc/25
70 | { }
71 PUT movies/_doc/26
72 | { }
73 PUT movies/_doc/27
74 | { }
75 PUT movies/_doc/28
76 | { }
77 PUT movies/_doc/29
78 | { }
79 PUT movies/_doc/30
80 | { }
81 PUT movies/_doc/31
82 | { }
83 PUT movies/_doc/32
84 | { }
85 PUT movies/_doc/33
86 | { }
87 PUT movies/_doc/34
88 | { }
89 PUT movies/_doc/35
90 | { }
91 PUT movies/_doc/36
92 | { }
93 PUT movies/_doc/37
94 | { }
95 PUT movies/_doc/38
96 | { }
97 PUT movies/_doc/39
98 | { }
99 PUT movies/_doc/40
100 | { }
101 PUT movies/_doc/41
102 | { }
103 PUT movies/_doc/42
104 | { }
105 PUT movies/_doc/43
106 | { }
107 PUT movies/_doc/44
108 | { }
109 }

200 - OK | 21 ms
1+ [
2   "took" : 1,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 3,
13      "relation" : "eq"
14    },
15    "max_score" : 3.63558,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "6",
21        "score" : 3.63558,
22        "source" : {
23          "title" : "The Assassination of Jesse James by the Coward Robert Ford",
24          "director" : "Andrew Dominik",
25          "year" : 2007,
26          "genres" : [
27            "Biography",
28            "Crime",
29            "Drama"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.63558,
38      }
39    ]
40  }
41 }

200 - OK | 20 ms
1+ [
2   "took" : 1,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 3.0586002,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "5",
21        "score" : 3.0586002,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.0586002,
38      }
39    ]
40  }
41 }

200 - OK | 20 ms
1+ [
2   "took" : 0,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 3.0586002,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "5",
21        "score" : 3.0586002,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.0586002,
38      }
39    ]
40  }
41 }

200 - OK | 20 ms
1+ [
2   "took" : 0,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 3.0586002,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "5",
21        "score" : 3.0586002,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.0586002,
38      }
39    ]
40  }
41 }

```

فیلم های اکشن با عنوانی شامل bill برگردانده می شود :

```

History Settings Help
72 # getting all darama genres
73 POST /movies/movie/_search
74 | { }
75 PUT movies/_doc/1
76 | { }
77 PUT movies/_doc/2
78 | { }
79 PUT movies/_doc/3
80 | { }
81 PUT movies/_doc/4
82 | { }
83 PUT movies/_doc/5
84 | { }
85 PUT movies/_doc/6
86 | { }
87 PUT movies/_doc/7
88 | { }
89 PUT movies/_doc/8
90 | { }
91 PUT movies/_doc/9
92 | { }
93 PUT movies/_doc/10
94 | { }
95 PUT movies/_doc/11
96 | { }
97 PUT movies/_doc/12
98 | { }
99 PUT movies/_doc/13
100 | { }
101 PUT movies/_doc/14
102 | { }
103 PUT movies/_doc/15
104 | { }
105 PUT movies/_doc/16
106 | { }
107 PUT movies/_doc/17
108 | { }
109 PUT movies/_doc/18
110 | { }
111 PUT movies/_doc/19
112 | { }
113 "query": {
114   "bool": {
115     "must": [
116       {
117         "match": {
118           "genres": "action"
119         }
120       },
121       {
122         "match": {
123           "title": {
124             "query": "kill",
125             "boost": 3
126           }
127         }
128       }
129     ]
130   }
131 }
132 }

200 - OK | 20 ms
1+ [
2   "took" : 0,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 3.0586002,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "5",
21        "score" : 3.0586002,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.0586002,
38      }
39    ]
40  }
41 }

200 - OK | 20 ms
1+ [
2   "took" : 0,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 3.0586002,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "5",
21        "score" : 3.0586002,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.0586002,
38      }
39    ]
40  }
41 }

200 - OK | 20 ms
1+ [
2   "took" : 0,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 3.0586002,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "id" : "5",
21        "score" : 3.0586002,
22        "source" : {
23          "title" : "Kill Bill: Vol. 1",
24          "director" : "Quentin Tarantino",
25          "year" : 2003,
26          "genres" : [
27            "Action",
28            "Crime",
29            "Thriller"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "id" : "7",
37        "score" : 3.0586002,
38      }
39    ]
40  }
41 }

```

## ترکیب must, must\_not, should

```

History Settings Help
96 # search on several fields at same time
97 POST /movies/_search
98+ ([])
109
110 # combining of: match, AND, OR
111 | POST /movies/_search
112+ ([])
133
134 #using "must", "must_not", "should"
135 POST /movies/_search
136+ {
137+   "query": {
138+     "bool": {
139+       "must": [
140+         {
141+           "should": [
142+             {
143+               "match": {
144+                 "title": "ford"
145+               }
146+             },
147+             {
148+               "match": {
149+                 "title": "kill"
150+               }
151+             }
152+           ],
153+         },
154+         {
155+           "must_not": [
156+             "match": {
157+               "genres": "Mystery"
158+             }
159+           ]
160+         }
161+       ]
162+     }
163+   }
164+ }

```

```

1+ [
2+   {
3+     "took" : 0,
4+     "timed_out" : false,
5+     "_shards" : {
6+       "total" : 1,
7+       "successful" : 1,
8+       "skipped" : 0,
9+       "failed" : 0
10+     },
11+     "hits" : [
12+       {
13+         "total" : {
14+           "value" : 2,
15+           "relation" : "eq"
16+         },
17+         "max_score" : 3.0586002,
18+         "hits" : [
19+           {
20+             "_index" : "movies",
21+             "_type" : "movie",
22+             "_id" : "3",
23+             "score" : 3.0586002,
24+             "_source" : {
25+               "title": "Kill Bill: Vol. 1",
26+               "director": "Quentin Tarantino",
27+               "year": 2003,
28+               "genres": [
29+                 "Action",
30+                 "Crime",
31+                 "Thriller"
32+               ]
33+             }
34+           },
35+           {
36+             "_index" : "movies",
37+             "_type" : "movie",
38+             "_id" : "2",
39+             "score" : 3.0586002,
40+           }
41+         ]
42+       }
43+     ],
44+     "max_score" : 3.0586002,
45+     "hits" : [
46+       {
47+         "total" : {
48+           "value" : 2,
49+           "relation" : "eq"
50+         },
51+         "max_score" : 1.0,
52+         "hits" : [
53+           {
54+             "_index" : "movies",
55+             "_type" : "movie",
56+             "_id" : "1",
57+             "score" : 1.0,
58+             "_source" : {
59+               "title": "Kill Bill: Vol. 1",
60+               "director": "Quentin Tarantino",
61+               "year": 2003,
62+               "genres": [
63+                 "Action",
64+                 "Crime",
65+                 "Thriller"
66+               ]
67+             }
68+           }
69+         ]
70+       }
71+     ]
72+   }
73+ ]

```

## جستجوی فیلدهای غیر متنی:

```

History Settings Help
90
91 # existing "kill" or "bill" in title
92 | POST movies/_search
93+ ([])
103
104 # search on several fields at same time
105 POST /movies/_search
106+ ([])
117
118 # combining of: match, AND, OR
119 | POST /movies/_search
120+ ([])
141
142 #using "must", "must_not", "should"
143 POST /movies/_search
144+ ([])
171
172 #searching non-string field
173 | _search
174+ {
175+   "query": {
176+     "term": {
177+       "year": {
178+         "value": "2003",
179+         "boost": 1.0
180+       }
181+     }
182+   }
184
185 # filtering in search
186 | POST /movies/movie/_search
187+ ([])
220
221 #above query summary
222 POST /movies/_search
223+ ([])

```

```

1+ #! this request accesses system indices: [.app-agent-configuration, .app-custom-link, .kibana.7.12.1.001, .kibana.task_manager.7.12.1.001], but in a future major version
2+ {
3+   "took" : 2,
4+   "timed_out" : false,
5+   "shards" : {
6+     "total" : 0,
7+     "successful" : 0,
8+     "skipped" : 0,
9+     "failed" : 0
10+   },
11+   "hits" : [
12+     {
13+       "value" : 2,
14+       "relation" : "eq"
15+     },
16+     {
17+       "max_score" : 1.0,
18+       "hits" : [
19+         {
20+           "_index" : "movies",
21+           "_type" : "movie",
22+           "_id" : "3",
23+           "score" : 1.0,
24+           "_source" : {
25+             "title": "Kill Bill: Vol. 1",
26+             "director": "Quentin Tarantino",
27+             "year": 2003,
28+             "genres": [
29+               "Action",
30+               "Crime",
31+               "Thriller"
32+             ]
33+           }
34+         }
35+       ]
36+     }
37+   ]
38+ }
39+ {
40+   "_index" : "movies",

```

## اعمال فیلتر در جستجو:

```

184
185 # filtering in search
186 | POST /movies/movie/_search
187+ {
188+   "query": {
189+     "bool": {
190+       "must": [
191+         {
192+           "match": {
193+             "title": "kill"
194+           }
195+         },
196+       ],
197+       "filter": {
198+         "bool": {
199+           "must": [
200+             {
201+               "range": {
202+                 "year": {
203+                   "gte": 1960
204+                 }
205+               },
206+             },
207+             {
208+               "term": {
209+                 "genres": {
210+                   "value": "drama"
211+                 }
212+               }
213+             }
214+           ]
215+         }
216+       }
217+     }
218+   }
219+ }

```

```

1+ #! [types removal] Specifying types in search requests is deprecated.
2+ {
3+   "took" : 4,
4+   "timed_out" : false,
5+   "shards" : {
6+     "total" : 1,
7+     "successful" : 1,
8+     "skipped" : 0,
9+     "failed" : 0
10+   },
11+   "hits" : [
12+     {
13+       "total" : {
14+         "value" : 1,
15+         "relation" : "eq"
16+       },
17+       "max_score" : 0.8373641,
18+       "hits" : [
19+         {
20+           "_index" : "movies",
21+           "_type" : "movie",
22+           "_id" : "3",
23+           "score" : 0.8373641,
24+           "_source" : {
25+             "title": "To Kill a Mockingbird",
26+             "director": "Robert Mulligan",
27+             "year": 1962,
28+             "genres": [
29+               "Drama",
30+               "Mystery"
31+             ]
32+           }
33+         }
34+       ]
35+     }
36+   ]
37+ }

```

## کوئری بالا به شکل خلاصه تر:

The screenshot shows the Elasticsearch Dev Tools interface with two panes. The left pane displays the search query code, and the right pane shows the search results.

```

History Settings Help
173 GET /search
174 | ( )
184 # filtering in search
185 | POST /movies/movie/_search
187 | ( )
220
221 #above query summary
222 POST /movies/_search
223 {
224     "query": {
225         "bool": {
226             "must": [
227                 {
228                     "match": {
229                         "title": "kill"
230                     }
231                 },
232                 {
233                     "range": {
234                         "year": {
235                             "gte": 1960
236                         }
237                     }
238                 },
239                 {
240                     "term": {
241                         "genres": {
242                             "value": "drama"
243                         }
244                     }
245                 }
246             ]
247         }
248     }
249 }

```

```

1: [
2   "took" : 1,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 1,
13      "relation" : "eq"
14    },
15    "max_score" : 2.3212342,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "_id" : "3",
21        "_score" : 2.3212342,
22        "_source" : {
23          "title" : "To Kill a Mockingbird",
24          "director" : "Robert Mulligan",
25          "year" : 1962,
26          "genres" : [
27            "crime",
28            "drama",
29            "mystery"
30          ]
31        }
32      }
33    ]
34  }
35 }
36

```

## فیلتر کردن روی ژانر و سال بدون جستجوی متن:

The screenshot shows the Elasticsearch Dev Tools interface with two panes. The left pane displays the search query code, and the right pane shows the search results.

```

History Settings Help
174 | ( )
184 # filtering in search
185 | POST /movies/movie/_search
187 | ( )
220
221 #above query summary
222 POST /movies/_search
223 | ( )
250
251 #filter on "genre" and "year" and not on text
252 POST movies/_search
253 {
254     "query": {
255         "bool": {
256             "must": [
257                 {
258                     "range": {
259                         "year": {
260                             "gte": 2000
261                         }
262                     }
263                 },
264                 {
265                     "term": {
266                         "genres": {
267                             "value": "drama"
268                         }
269                     }
270                 }
271             ]
272         }
273     }
274 }
275
276 #searching in miss dictation
277 POST /movies/_search
278

```

```

1: [
2   "took" : 0,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 2,
13      "relation" : "eq"
14    },
15    "max_score" : 1.48387,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "_id" : "6",
21        "_score" : 1.48387,
22        "_source" : {
23          "title" : "The Assassination of Jesse James by the Coward Robert Ford",
24          "director" : "Andrew Dominik",
25          "year" : 2007,
26          "genres" : [
27            "Biography",
28            "Crime",
29            "Drama"
30          ]
31        }
32      },
33      {
34        "index" : "movies",
35        "type" : "movie",
36        "_id" : "8",
37        "_score" : 1.48387,

```

## سرچ کردن با در نظر گرفتن غلط دیکته ای:

The screenshot shows the Elasticsearch Dev Tools interface with two panes. The left pane displays the search query code, and the right pane shows the search results.

```

Console Search Profiler Grok Debugger Painless Lab (BETA)
History Settings Help
186 | POST /movies/movie/_search
187 | ( )
220
221 #above query summary
222 POST /movies/_search
223 | ( )
250
251 #filter on "genre" and "year" and not on text
252 POST movies/_search
253 | ( )
275
276 #searching in miss dictation
277 POST /movies/_search
278 {
279     "query": {
280         "match": {
281             "title": {
282                 "query": "godfather",
283                 "fuzziness": "AUTO"
284             }
285         }
286     }
287 }
288 #searching free snare

```

```

1: [
2   "took" : 31,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 1,
13      "relation" : "eq"
14    },
15    "max_score" : 1.1472913,
16    "hits" : [
17      {
18        "index" : "movies",
19        "type" : "movie",
20        "_id" : "1",
21        "_score" : 1.1472913,
22        "_source" : {
23          "title" : "The Godfather",
24          "director" : "Francis Ford Coppola",
25          "year" : 1972

```

## قبول کردن فطاوی خالی در کوئری:

```

History Settings Help
185 #filtering in search
186 | POST /movies/movie/_search
187 | { }
220
221 #above query summary
222 POST /movies/_search
223 | { }
296
295 #filter on "genre" and "year" and not on text
292 POST movies/_search
293 | { }
276 #searching in miss dictation
277 POST /movies/_search
278 | { }
288
289 #accepting free space
290 POST /movies/_search
291 | { }
292 "query": {
293     "match_phrase": {
294         "title": {
295             "query": "kill Bill",
296             "slop": 2
297         }
298     }
299 }
300 }
301
302 ### common error
303 POST movies/movie/_search
314 |
315 #
316 | PUT /movies/movie/_mapping
317 | { }
330
    
```

```

1 [
2   {
3     "took": 13,
4     "timed_out": false,
5     "_shards": 1,
6     "_total": 1,
7     "successful": 1,
8     "skipped": 0,
9     "failed": 0
10   },
11   "hits": [
12     {
13       "total": {
14         "value": 2,
15         "relation": "eq"
16       },
17       "max_score": 2.114714,
18       "hits": [
19         {
20           "_index": "movies",
21           "_type": "movie",
22           "_id": "5",
23           "_score": 2.114714,
24           "_source": {
25             "title": "kill Bill: vol. 1",
26             "director": "Quentin Tarantino",
27             "year": 2003,
28             "genres": [
29               "Action",
30               "Crime",
31               "Thriller"
32             ]
33           }
34         },
35         {
36           "_index": "movies",
37           "_type": "movie",
37           "_id": "7",
37           "_score": 2.114714,
38         }
39       ]
40     }
41   }
42 ]
    
```

اشتباه رایج:

```

History Settings Help
186 | POST /movies/movie/_search
187 | { }
220
221 #above query summary
222 POST /movies/_search
223 | { }
296
295 #filter on "genre" and "year" and not on text
292 POST movies/_search
293 | { }
276 #searching in miss dictation
277 POST /movies/_search
278 | { }
288
289 #accepting free space
290 POST /movies/_search
291 | { }
301
302 ### common error
303 POST movies/movie/_search
304 | { }
305 "query": [
306   {
307     "term": {
308       "director": {
309         "value": "Francis Ford Coppola"
310       }
311     }
312   }
313 ]
    
```

```

1 #! [types removal] Specifying types in search requests is deprecated.
2 {
3   "took": 1,
4   "timed_out": false,
5   "_shards": 1,
6   "_total": 1,
7   "successful": 1,
8   "skipped": 0,
9   "failed": 0
10 },
11 "hits": [
12   {
13     "total": {
14       "value": 0,
15       "relation": "eq"
16     },
17     "max_score": null,
18     "hits": []
19   }
20 ]
    
```

## سه کوئری دلخواه روی داده های بورس:

۱) در این کوئری داده هایی که در قسمت content سامل کلمه‌ی «خساپا» هستند را برمی‌گردانیم.

```

341 #####GET /twitter/_search
342 # سه کوئری شنود
343 GET /twitter/_search
344 {
345   "query": {
346     "query_string": {
347       "query": "خساپا",
348       "fields": ["content"]
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

```

200 - OK | 19 ms

```

9 },
10 "hits": {
11   "total": {
12     "value": 10,
13     "relation": "eq"
14   },
15   "max_score": 8.466699,
16   "hits": [
17     {
18       "index": "twitter",
19       "type": "twitter",
20       "id": "hlvznkBBFMUgQ1dlQks",
21       "score": 8.466699,
22       "source": {
23         "id": "28476605",
24         "sendTime": "2021-05-13T15:51:38Z",
25         "sendTimePersian": "1400/02/24 20:21",
26         "senderName": "مد",
27         "senderUsername": "brahimkohestan",
28         "senderProfileImage": "default",
29         "content": "خساپا یک ماه دیگه واقعیت قیمت هشیبا 70 درصد رشد کرده است  
نیمه هیله از بازار طبق اطلاعات الاق فروزه ای که این این این این  
بله خصوصاً که صادراتی میتوان 4 تریلیون تومان شد از این این این  
کوئری درست  
30       "type": "twit",
31       "scoredPostDate": "1620921119240",
32       "finalFullDatePersian": "",
33       "hashtags": [
34         "خساپا"
35       ]
36     },
37     {
38       "index": "twitter",
39       "type": "twitter",
40       "id": "CijaaKbxMxSKL3XNyui",
41       "score": 7.27575,
42     }

```

۲) داده هایی که type pull است را برمی‌گردانیم

```

355 GET /twitter/_search
356 {
357   "query": {
358     "bool": {
359       "filter": {
360         "bool": {
361           "must": [
362             {
363               "term": {
364                 "type": {
365                   "value": "pull"
366                 }
367               }
368             }
369           ]
370         }
371       }
372     }
373   }
374 }
375
376
377
378
379
380
381
382
383
384
385

```

200 - OK

```

1 {
2   "took": 5,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10   "hits": {
11     "total": {
12       "value": 4,
13       "relation": "eq"
14     },
15     "max_score": 0.0,
16     "hits": [
17       {
18         "index": "twitter",
19         "type": "twitter",
20         "id": "dc1XankBvxMsKL3XtyI4",
21         "score": 0.0,
22         "source": {
23           "id": "285053039",
24           "sendTime": "2021-05-14T08:50:58Z",
25           "sendTimePersian": "1400/02/24 13:20",
26           "senderName": "Abbas",
27           "senderUsername": "abbasabbas",
28           "senderProfileImage": "default",
29           "content": "#طرسنگی_تساحص_بورس# همین حرام سرعی هیبت و حضرت را بروی  
30           ""جهة این مسنه"

```

۳) توييت هاى که در content شامل کلمه «بخرید» می باشد را بازمي گردانيم

```

354
355 GET /twitter/_search
356 {
375
376 GET /twitter/_search
377 {
378 "query": {
379 "query_string": {
380 "query": "بخرید",
381 "fields": ["content"]
382 }
383 }
384 }
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402

```

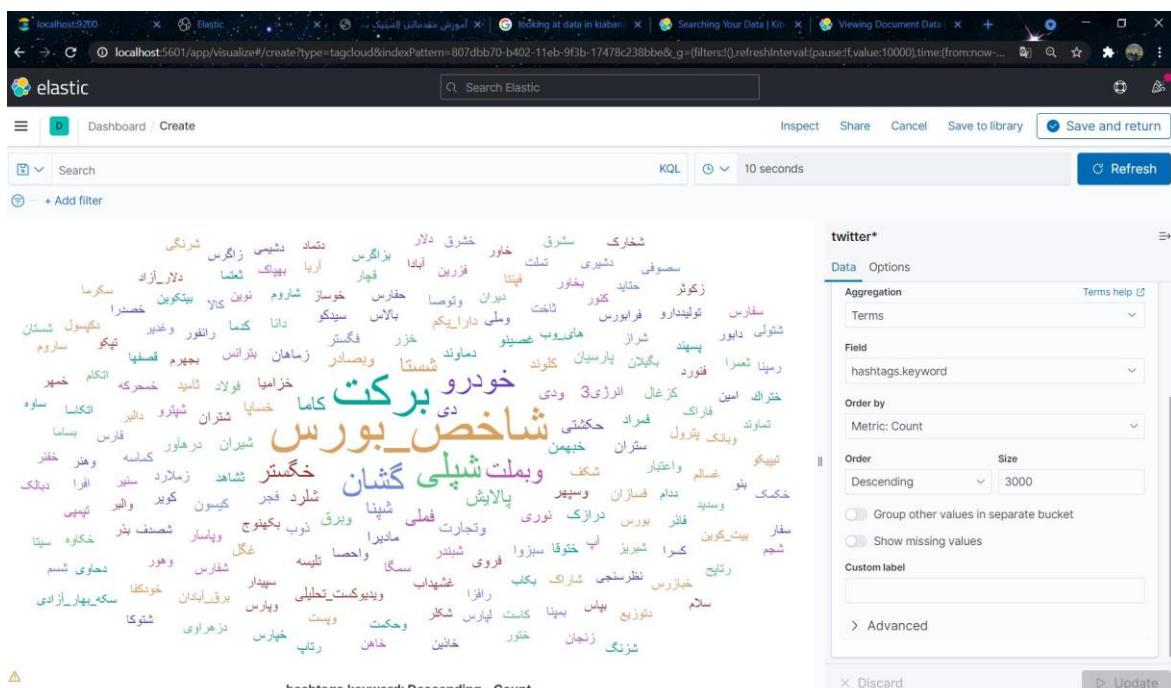
```

1 * [
2   "took" : 29,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  ],
11  "hits" : {
12    "total" : {
13      "value" : 10,
14      "relation" : "eq"
15    },
16    "max_score" : 7.096911,
17    "hits" : [
18      {
19        "_index" : "twitter",
20        "_type" : "twitter",
21        "_id" : "PSLBankBvxMsKL3XOCvr",
22        "_score" : 7.096911,
23        "_source" : {
24          "id" : "285067052",
25          "sendTime" : "2021-05-14T12:01:41Z",
26          "sendTimePersian" : "1400/02/24 16:31",
27          "parentSendTime" : "2021-05-11T18:14:48Z",
28          "parentSendTimePersian" : "1400/02/21 22:44",
29          "parentId" : "283902946",
30          "parentSenderName" : "مسعود",
31          "parentSenderUsername" : "mgh51",
          "parentSenderProfileImage" : "a0d4dc...-a6ad"
        }
      }
    ]
  }
}

```

## :Visualization

۱) ابر کلمات



## ۲) تعداد نمادهای پر تکرار

## ۳) تعداد کاربران با بیشترین توزیع

