

به نام خدا

تمرین دوم
NoSQL Databases

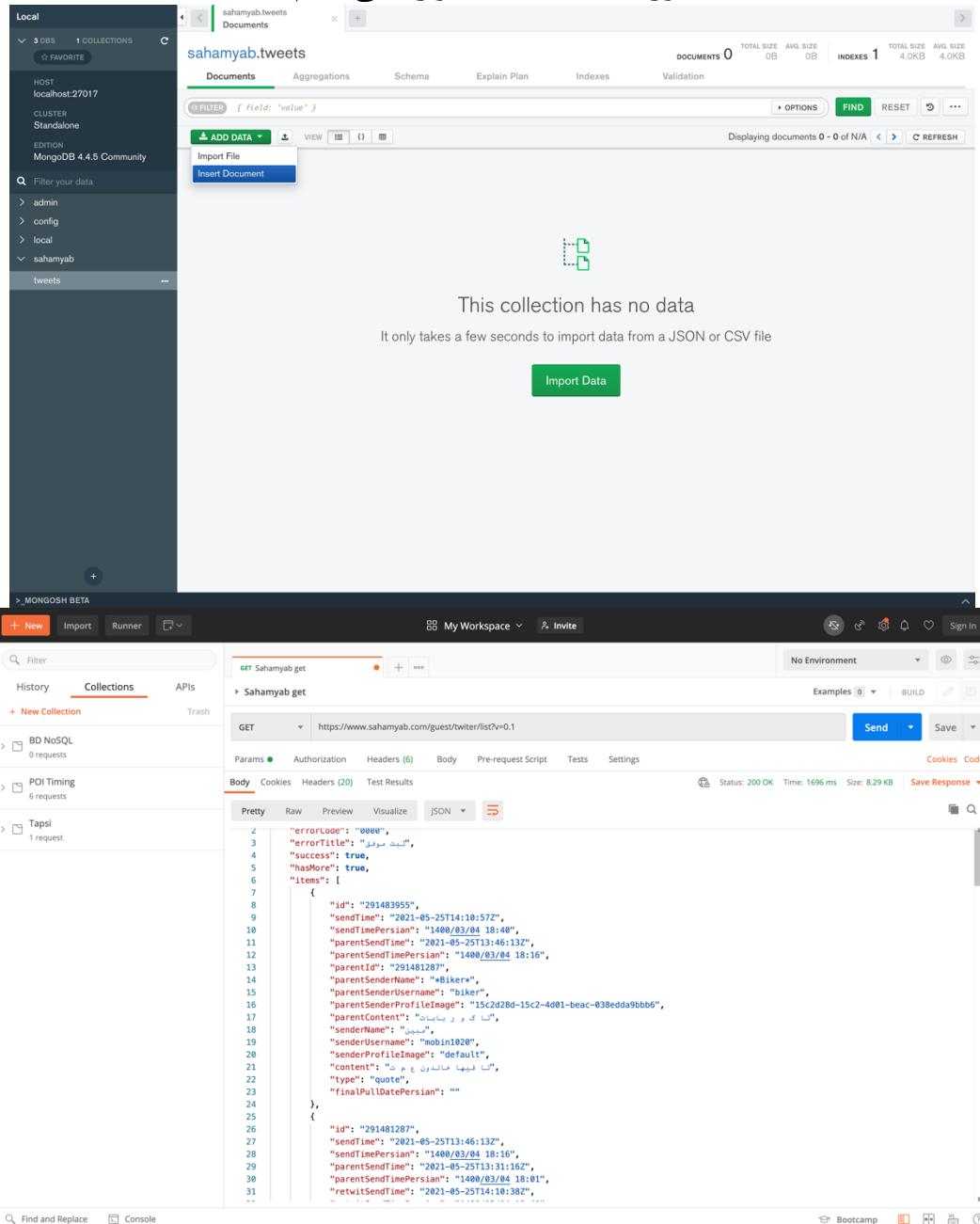
امیرمحمد رنجبر پازکی ۸۱۰۱۹۹۳۴۰

درس بیگ دیتا

دانشکده‌ی مهندسی برق و کامپیوتر
بهار ۱۴۰۰

• بخش اول - کار با MongoDB + گام اول:

برای ورود دستی داده‌ها، پس از ساخت DataBase و collection مورد نظر، با استفاده از postman پاسخ توییت‌ها را دریافت می‌کنم و بعد با استفاده از گزینه Add document به صورت json داده‌ها وارد می‌کنیم.

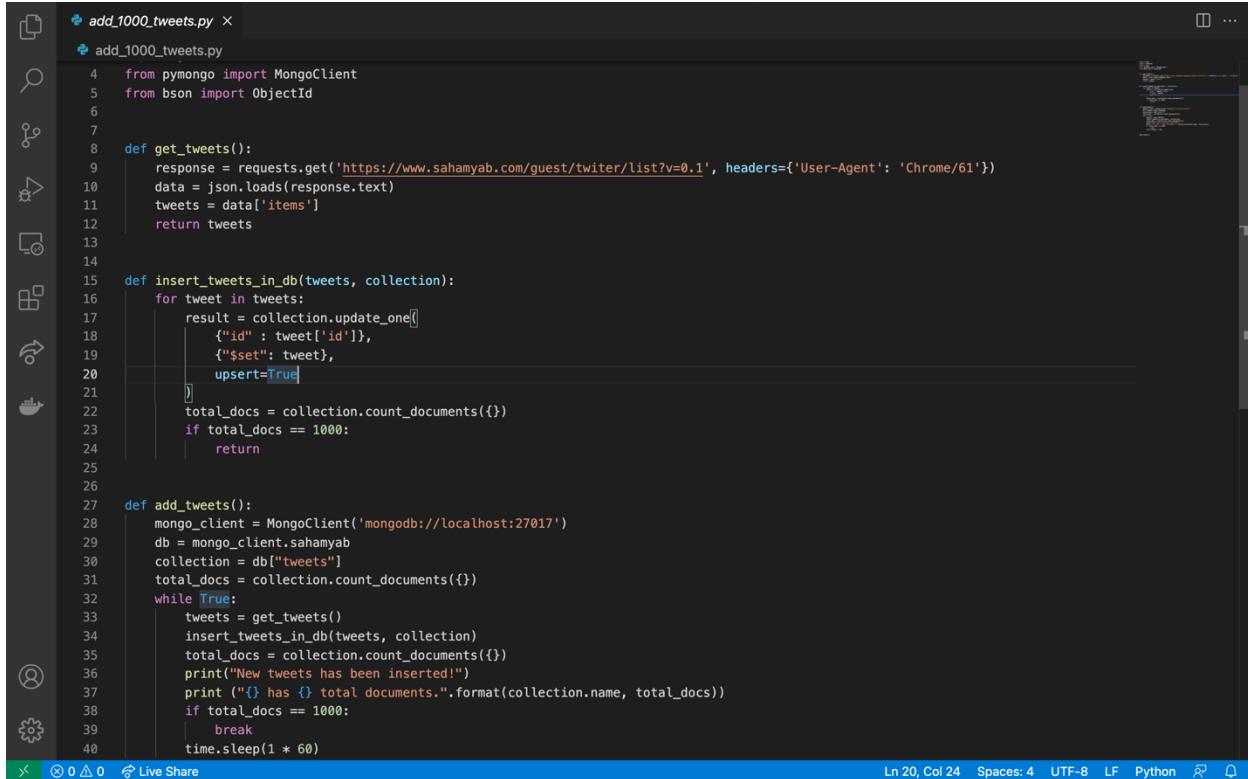


The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Local' selected, showing 'HOST: localhost:27017', 'CLUSTER: Standalone', and 'EDITION: MongoDB 4.4.5 Community'. Below that is a 'Filter your data' section with options like 'admin', 'config', 'local', and 'sahamyab'. Under 'sahamyab', 'tweets' is expanded. In the main area, the 'sahamyab.tweets' collection is selected, and an 'Insert to Collection sahamyab.tweets' dialog is open. The dialog shows a JSON document with various fields, including Persian text such as 'پرست اصلاح معمول برگشت تا چه حد و حدودی' and 'رسانید اصلاح معمول برگشت تا چه حد و حدودی'. The document also contains fields like '_id', 'sendTime', 'parentSendTime', 'parentContent', 'senderName', 'senderProfileImage', 'content', 'type', and 'finalPullDate'. At the bottom of the dialog are 'CANCEL' and 'INSERT' buttons.

The main interface shows the collection 'sahamyab.tweets' with 10 documents. The top bar displays 'DOCUMENTS 10 TOTAL SIZE 7.9KB AVG. SIZE 812B INDEXES 1 TOTAL SIZE 4.0KB AVG. SIZE 4.0KB'. Below the table, there are buttons for 'OPTIONS', 'FIND', 'RESET', and 'REFRESH'.

همانطور که می‌بینید، داده‌ها با موفقیت وارد شده‌اند.
فیلد جدید که **Mongo** اضافه کرده‌است، فیلد **_id** است که با **index** اصلی و به عنوان شناسه داده‌ها استفاده شده‌است.

کدی که برای گرفتن و درج اطلاعات نوشته شده است در فایل add_100_tweets.py بارگذاری شده است. تصویر زیر نیز نمایی از این کد است.

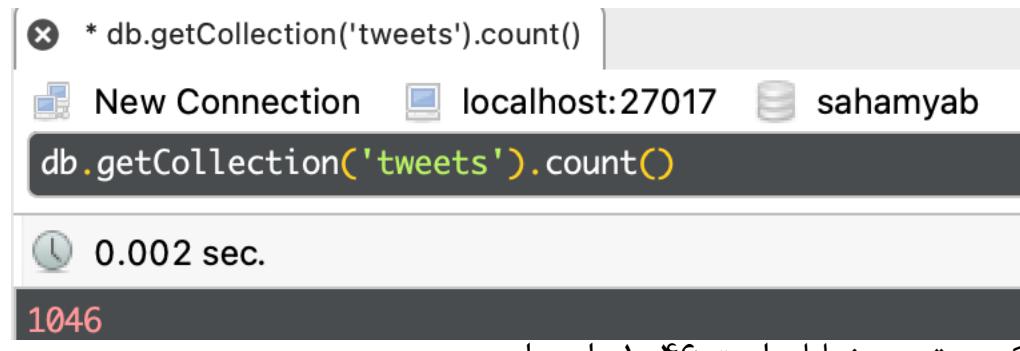


```
add_100_tweets.py
add_1000_tweets.py

4  from pymongo import MongoClient
5  from bson import ObjectId
6
7
8  def get_tweets():
9      response = requests.get('https://www.sahamyab.com/guest/twiter/list?v=0.1', headers={'User-Agent': 'Chrome/61'})
10     data = json.loads(response.text)
11     tweets = data['items']
12     return tweets
13
14
15  def insert_tweets_in_db(tweets, collection):
16      for tweet in tweets:
17          result = collection.update_one([
18              {"id": tweet['id']},
19              {"$set": tweet},
20              {"upsert": True}
21          ])
22      total_docs = collection.count_documents({})
23      if total_docs == 1000:
24          return
25
26
27  def add_tweets():
28      mongo_client = MongoClient('mongodb://localhost:27017')
29      db = mongo_client.sahamyab
30      collection = db['tweets']
31      total_docs = collection.count_documents({})
32      while True:
33          tweets = get_tweets()
34          insert_tweets_in_db(tweets, collection)
35          total_docs = collection.count_documents({})
36          print("New tweets has been inserted!")
37          print("{} has {} total documents.".format(collection.name, total_docs))
38          if total_docs == 1000:
39              break
40          time.sleep(1 * 60)

Ln 20, Col 24  Spaces: 4  UTF-8  LF  Python  🔍  🔍
```

پس از اجرای این کد با استفاده از دستور count تعداد داده‌ها بررسی شد که حداقل ۱۰۰۰ مورد بوده‌اند. دلیل آن نیز این است که شرط روی هر داده گذاشته نشده بود و از حد گذشته است و متوقف نشده است.



```
* db.getCollection('tweets').count()
New Connection localhost:27017 sahamyab
db.getCollection('tweets').count()
0.002 sec.
1046
```

همانطور که در تصویر نمایان است ۱۰۴۶ داده داریم.
+ گام دوم:

```
db.tweets.aggregate([
  {
    $addFields: {
      returnObject: {
        $regexFindAll: { input: "$content", regex:
          /(#[\w-]+)#
        }
      }
    }
  ]
]).forEach(
  function(doc) {
    var result = doc["returnObject"]
    var hashtags = []
    for (found in result) {
      var hashtag = result[found].captures[0]
      hashtags.push(hashtag)
    }
  }
).forEach(
  function(doc) {
    var result = doc["returnObject"]
    var hashtags = []
    for (found in result) {
      var hashtag = result[found].captures[0]
      hashtags.push(hashtag)
    }
    if (hashtags) {
      db.tweets.updateOne({
        id: doc["id"]
      }, {
        $set :{ hashtags: hashtags}
      });
    }
  }
)
```

The screenshot shows the regex101.com interface. In the 'REGULAR EXPRESSION' field, the pattern `/#([\u062f-\u0645]+)` is entered with the 'gm' flag. The 'TEST STRING' is a Persian sentence: "سلام بزر شما دوستان من #کامپیوутری خوبیدم ولی این سهم جز اذیت چیزی نداشتند الان میخوام پس لایست بخشم". The 'EXPLANATION' section details the regex components: `#` matches the character # literally (case sensitive), **1st Capturing Group** `([\u062f-\u0645]+)` matches a single character present in the list below, and `\u062f-\u0645` matches the previous token between `one` and `unLimited times`, as many times as possible, giving back as needed (greedy). The 'MATCH INFORMATION' section shows two matches: Match 1 at 22-27 with value '#کامپیو', Group 1 at 23-27 with value 'کامپیو', Match 2 at 84-91 with value 'پس لایست', and Group 1 at 85-91 with value 'پس لایست'. The 'QUICK REFERENCE' sidebar provides links to various regex concepts.

همینطور با استفاده از کد پایتون زیر کاراکترهای بد جایگذاری شده‌اند.

The screenshot shows a code editor with a Python script named `preprocess.py`. The code defines functions to replace bad characters in a MongoDB collection and to add tweets. It uses the `pymongo` library to interact with a MongoDB database running on localhost port 27017. The code editor interface includes a sidebar with file navigation and a bottom status bar showing file statistics.

```


```

preprocess.py
from pymongo import MongoClient

BAD_CHARS_MAPPING = {
 '\u202c': '\u062f',
 '\u202d': '\u0645'
}

def replace_tweets_bad_chars_in_db(collection):
 for document in collection.find():
 for key, value in BAD_CHARS_MAPPING.items():
 document['content'] = document["content"].replace(key, value)
 result = collection.update_one(
 {"id": document['id']},
 {"$set": document}
)

def add_tweets():
 mongo_client = MongoClient('mongodb://localhost:27017')
 db = mongo_client.sahamyab
 collection = db['tweets']
 replace_tweets_bad_chars_in_db(collection)

add_tweets()

```


```

نتیجه و زمان اجرا به صورت زیر شد.

```

var result = doc["returnObject"]
var hashtags = []
for (found in result) {
    var hashtag = result[found].captures[0]
    hashtags.push(hashtag)
}
if (hashtags) {
    db.tweets.updateOne({
        id: doc["id"]
    }, {
        $set :{ hashtags: hashtags}
    });
}
}

var after = new Date()
execution_mills = after - before
print(execution_mills)

```

⌚ 0 sec.

576

Key	Value	Type
(1) ObjectId("60ad1ae582ea55a485552010")	{ 21 fields }	Object
_id	ObjectId("60ad1ae582ea55a485552010")	ObjectId
id	291628404	String
content	#سپردا فکر کنم سپردا از فردا حرف نشونم را شروع کنه مواظب باشید رکب خنجرید.	String
finalFullDatePersian	1621957903513	String
scoredPostDate	2021-05-25T15:41:48Z	String
sendTime	1400/03/04 20:11	String
sendTimePersian	و ساعت ۲۰:۱۱	String
senderName	سپردا	String
senderProfileImage	429d4c4-8eab-4d6b-b7a2-135d5aed9e2a	String
senderUsername	13540107ahmad	String
type	retwit	String
lastLikeNickName	سیمان سپردا	String
likeCount	1	String
retwitCount	1	String
retwtId	291913922	String
retwitSendTime	2021-05-25T18:08:00Z	String
retwitSendTimePersian	1400/03/04 22:38	String
retwitSenderName	سیمان سپردا	String
retwitSenderProfileImage	default	String
retwitSenderUsername	abolfazl_4532	String
hashtag	[1 element]	Array
[0]	سپردا	String
> (2) ObjectId("60ad1ae582ea55a485552012")	{ 22 fields }	Object
> (3) ObjectId("60ad1ae582ea55a485552014")	{ 18 fields }	Object
> (4) ObjectId("60ad1ae582ea55a485552016")	{ 14 fields }	Object
> (5) ObjectId("60ad1ae582ea55a485552018")	{ 12 fields }	Object
> (6) ObjectId("60ad1ae582ea55a48555201a")	{ 18 fields }	Object
> (7) ObjectId("60ad1ae582ea55a48555201c")	{ 12 fields }	Object
> (8) ObjectId("60ad1ae582ea55a48555201e")	{ 21 fields }	Object
> (9) ObjectId("60ad1ae582ea55a485552020")	{ 22 fields }	Object
> (10) ObjectId("60ad1ae582ea55a485552022")	{ 12 fields }	Object
> (11) ObjectId("60ad1b2382ea55a48555204e")	{ 12 fields }	Object
> (12) ObjectId("60ad1b2382ea55a485552050")	{ 18 fields }	Object
> (13) ObjectId("60ad1b2382ea55a485552052")	{ 12 fields }	Object

+ گام سوم: .1

```

db.getCollection('tweets').find({ "mediaContentType": "image/jpeg", "parentId": { $ne: null } }, { senderName: 1 })
  
```

Key	Value	Type
(1) ObjectId("60ad1b9d82ea55a4855520c1")	{ 2 fields }	Object
__id	ObjectId("60ad1b9d82ea55a4855520c1")	ObjectId
senderName	شمار	String
(2) ObjectId("60ad204a82ea55a4855524f2")	{ 2 fields }	Object
(3) ObjectId("60ad258a82ea55a48555298e")	{ 2 fields }	Object
__id	ObjectId("60ad258a82ea55a48555298e")	ObjectId
senderName	عمو مرتضى	String
(4) ObjectId("60ad30e982ea55a485552d63")	{ 2 fields }	Object
(5) ObjectId("60ad373482ea55a485552ff0")	{ 2 fields }	Object
(6) ObjectId("60ad382582ea55a4855530cc")	{ 2 fields }	Object
__id	ObjectId("60ad382582ea55a4855530cc")	ObjectId
senderName	نسرين	String
(7) ObjectId("60ad38da82ea55a485553181")	{ 2 fields }	Object
(8) ObjectId("60ad415482ea55a48555390")	{ 2 fields }	Object
(9) ObjectId("60ad5cd82ea55a4855541d1")	{ 2 fields }	Object
(10) ObjectId("60add2d082ea55a48555461f")	{ 2 fields }	Object
(11) ObjectId("60addb5882ea55a485554c08")	{ 2 fields }	Object
(12) ObjectId("60addbd182ea55a485554c62")	{ 2 fields }	Object
(13) ObjectId("60adde7582ea55a485554dc5")	{ 2 fields }	Object
(14) ObjectId("60addfa382ea55a485554ea0")	{ 2 fields }	Object

برای این کار باید از امکان update و foreach کردن داکهای موجود استفاده کرد. 2

```

var before = new Date()
db.getCollection('tweets').find({ $or: [
    { hashtags: "خودرو" },
    { hashtags: "شستا" },
    { hashtags: "فولاد" }
]}).forEach(
    function(doc) {
        db.tweets.updateOne({
            id: doc["id"]
        }, {
            $set :{ gov: true}
        });
    }
)
"خودرو"
}, { hashtags:
"شستا"
}, { hashtags:
"فولاد"
}

])}.forEach(
    function(doc) {
        db.tweets.updateOne({
            id: doc["id"]
        }, {
            $set :{ gov: true}
        });
    }
)
var after = new Date()
execution_mills = after - before
print(execution_mills)

```

Q32.js

```

    "جوده"
}, { hashtags:
"شیوه"
}, { hashtags:
"فولاذ"
}

]}).forEach(
function(doc) {
db.tweets.updateOne({
id: doc["id"]
}, {
$set :{ gov: true}
});
}

var after = new Date();
execution_mills = after - before
print(execution_mills)

```

0 sec.

30

New Connection (4)

System

config

sahamyab

Collections (2)

tweets

tweets_copy

Functions (0)

Users

Q32.js

```

db.getCollection('tweets').find({ $or: [
{ hashtags: "جوده" },
{ hashtags: "شیوه" },
{ hashtags: "فولاذ" }
]}).forEach(
function(doc) {
db.tweets.updateOne({
_id: doc._id
}, {
$set :{ gov: true}
});
}

var after = new Date();
execution_mills = after - before
print(execution_mills)

```

0.006 sec.

Key	Value	Type
(1) ObjectId("60ad1ae582ea55a485552022")	{ 13 fields }	Object
(2) ObjectId("60ad1c5982ea55a485552163")	{ 23 fields }	Object
(3) ObjectId("60ad1e1882ea55a485552f9")	{ 13 fields }	Object
(4) ObjectId("60ad1f0d82ea55a4855523d8")	{ 13 fields }	Object
_id	ObjectId("60ad1f0d82ea55a4855523d8")	ObjectID
id	291673701	String
content	#شایخ_پورس #جوده #شیوا اوباس که خون مردم را سالها به شنیشه کرده‌اند ...	String
finalPullDatePersian	1621958365369	String
scoredPostDate	2021-05-25T15:59:18Z	String
sendTime	2021/03/04 20:29	String
sendTimePersian	1400/03/04 20:29	String
senderName	ابن سپا	String
senderProfileImage	7560aae4-0f7b-4d1d-9ed3-ef0ecd4687a8	String
senderUsername	besharat12	String
type	twit	String
hashtags	[3 elements]	Array
gov	true	Boolean
(5) ObjectId("60ad1f4a82ea55a48555240f")	{ 13 fields }	Object
(6) ObjectId("60ad250c82ea55a48555291f")	{ 13 fields }	Object
(7) ObjectId("60ad389d82ea55a485553138")	{ 19 fields }	Object
(8) ObjectId("60ad395282ea55a4855531e8")	{ 13 fields }	Object
(9) ObjectId("60ad3af882ea55a48555335f")	{ 13 fields }	Object
(10) ObjectId("60ad411882ea55a4855538d9")	{ 13 fields }	Object
(11) ObjectId("60ad598182ea55a485553ed5")	{ 19 fields }	Object
(12) ObjectId("60ad5bab82ea55a4855540c1")	{ 20 fields }	Object
(13) ObjectId("60add25882ea55a4855545be")	{ 15 fields }	Object
(14) ObjectId("60add48482ea55a4855546ff")	{ 19 fields }	Object
(15) ObjectId("60add4fc82ea55a48555474f")	{ 13 fields }	Object
(16) ObjectId("60add4fc82ea55a48555475d")	{ 13 fields }	Object

Logs

```

var before = new Date()
db.tweets.aggregate([
{
    $addFields: {
        returnObject: {
            $regexFindAll: { input: "$sendTimePersian", regex: '^\\s(\\d{2})/' }
        }
    }
},
{ $set: { tweetHour: "$returnObject.match" } }
]).forEach(
    function(doc) {
        if(9 <= doc.tweetHour && doc.tweetHour <= 10){
            print(doc.senderName)
            print(doc.senderProfileImage)
            print('-----')
        }
    }
)
var after = new Date()
execution_mills = after - before
print(execution_mills)

```

Q33.js Q32.js

New Connection localhost:27017 sahamyab

```

        ^\\s(\\d{2})/
    }
}
{
    $set: { tweetHour: "$returnObject.match" } }
]).forEach(
    function(doc) {
        if(9 <= doc.tweetHour && doc.tweetHour <= 10){
            print(doc.senderName)
            print(doc.senderProfileImage)
            print('-----')
        }
    }
)
var after = new Date()
execution_mills = after - before
print(execution_mills)

```

db.tweets... execution_...

0 sec.

24

Q33.js

```

db.tweets.aggregate([
  { $set: { tweetHour: "$returnObject.match" } },
  { $group: {
    _id: '$senderName',
    count: { $sum: 1 }
  }},
  { $sort: { count: -1 } },
  { $limit: 10 }
])

```

Logs

```

انسان داره
4302ef5a-1877-4746-9292-e65e8bb3ab0d
انسان داره
4302ef5a-1877-4746-9292-e65e8bb3ab0d
انسان داره
4302ef5a-1877-4746-9292-e65e8bb3ab0d
انسان داره
4302ef5a-1877-4746-9292-e65e8bb3ab0d
کارکنان
8adc74ff-4a7f-4253-8645-5829f90660d
BUYER
de884f68-1650-4be8-846c-9f6c26a7b663
BUYER
de884f68-1650-4be8-846c-9f6c26a7b663
BUYER
de884f68-1650-4be8-846c-9f6c26a7b663

```

+ گام چهارم:

. 1

Q41.js

```

db.tweets.aggregate([
  { $group: { _id: '$senderUsername', count: { $sum: 1 } } },
  { $facet: {
    "categorizedByActivity": [
      { $bucket: {
        groupBy: "$count",
        boundaries: [1, 2, 4],
        default: "Other",
        output: {
          "count": { $sum: 1 },
          "titles": { $push: "$_id" }
        }
      }}
    ]
  }}
])

```

tweets 0.006 sec.

Key	Value	Type
(1)	{1 field}	Object
categorizedByActivity	[3 elements]	Array
[0]	{3 fields}	Object
_id	1.0	Double
count	486.0	Double
titles	[486 elements]	Array
[1]	{3 fields}	Object
_id	2.0	Double
count	169.0	Double
titles	[169 elements]	Array
[2]	{3 fields}	Object
_id	Other	String
count	34.0	Double
titles	[34 elements]	Array
[0]	aref6118	String
[1]	saammm	String
[2]	mgh51	String
[3]	profmoradi	String
[4]	alireza_trader	String
[5]	tahatrend	String
[6]	ali_gh	String
[7]	vantage	String

.2

```
db.tweets.aggregate([
  {$group : { _id : '$hashtags', count : {$sum : 1}}},
  { $sort : { count : -1 } }
])
```

Key	Value
> [1] [0 elements]	{ 2 fields }
▽ [2] [1 element]	{ 2 fields }
▽ [1] _id	[1 element]
[0]	شاحص
## count	52.0
▽ [3] [1 element]	{ 2 fields }
▽ [1] _id	[1 element]
[0]	برکت
## count	47.0

3

```
db.tweets_copy.aggregate(  
  { $match: { parentId: {$ne:null} } },  
  { $unset: "type" }  
)
```

Key	Value	Type
✓ (1) ObjectId("60ad1ae582ea55a485552014")	{ 17 fields }	Object
_id	ObjectId("60ad1ae582ea55a485552014")	ObjectId
id	291626622	String
content	دش انقدم به خودت سخت نگیر تو زندگیت اذیت میش اگه انقدر	String
finalPullDatePersian دوست عزیز هر کس انتخاب بشه حکم به عروسک رو داره والسلام	String
parentContent	291510526	String
parentId	2021-05-25T14:53:34Z	String
parentSendTime	1400/03/04 19:23	String
parentSendTimePersian	asghar musavi	String
parentSenderId	c828bae9-e92b-453c-9053-0399ebfbddd5	String
parentSenderUsername	soosmaas3	String
sendTime	2021-05-25T15:41:02Z	String
sendTimePersian	1400/03/04 20:11	String
senderName	Alireza0099	String
senderProfileImage	default	String
senderUsername	alirezabe123	String
> hashtags	[0 elements]	Array

.4

The screenshot shows the MongoDB Compass interface with two tabs open: 'tweets' and 'Logs'. The 'tweets' tab displays two results from an aggregation query. The first result is a document with an '_id' field containing the Persian word 'پس' (Pas) and a 'count' field of 61.0. The second result is a document with an '_id' field containing the Persian word 'پس' and a 'count' field of 1.0. The 'Logs' tab at the bottom is empty.

```
db.tweets.aggregate([
  {
    $group : { _id : '$hashtags', count : {$sum : 1} },
    { $match: { _id : {$ne:[]} } },
    { $sort : { count : -1 } },
    { $limit : 1 }
  ]
)
db.tweets.aggregate([
  {
    $group : { _id : '$hashtags', count : {$sum : 1} },
    { $match: { _id : {$ne:[]} } },
    { $sort : { count : 1 } },
    { $limit : 1 }
  ]
)
```

Key	Value	Type
\$_id [1 element]	{2 fields}	Object
\$_id [0]	[1 element]	Array
\$_id [0]	پس	String
count	61.0	Double

Key	Value	Type
\$_id [1 element]	{2 fields}	Object
\$_id [0]	[1 element]	Array
\$_id [0]	پس	String
count	1.0	Double

.5

```
db.tweets.aggregate([
  {
    $addFields: {
      returnObject: {
        $regexFindAll: { input: "$sendTimePersian", regex:
          /(.{10})\s/
        }
      }
    }
  },
  { $group : { "_id": {
      "hashtags": "$hashtags",
      "date": "$returnObject.match"
    }, count : {$sum : 1}}
  },
  { $match: { "_id.hashtags" : {$ne:[]} } },
  { "$sort": { "count": -1 } },
  { "$group": {
```

```

        }, count : {$sum : 1}}
    },
    { $match: { "_id.hashtags" : {$ne:[]} } },
    { "$sort": { "count": -1 } },
    { "$group": {
        "_id": "$_id.date",
        "results": {
            "$push": {
                "name": "$_id.hashtags",
                "count": "$count",
            }
        }
    }},
    { "$sort": { "_id.0": 1}},
    { "$project": {
        "results": { "$slice": [ "$results", 10 ] }
    }}
]
)

```

Screenshot of the MongoDB Compass interface showing the results of the aggregation query.

The left sidebar shows the database connection and collections:

- New Connection (4)
- System
- config
- sahamayab
- Collections (2)
 - > tweets
 - > tweets_copy
- Functions (0)
- Users

The main pane displays the aggregation pipeline and its results:

```

{
  "$group": {
    "_id": "$_id.date",
    "results": {
      "$push": {
        "name": "$_id.hashtags",
        "count": "$count",
      }
    }
  },
  "$sort": { "_id.0": 1},
  "$project": {
    "results": { "$slice": [ "$results", 10 ] }
  }
}

```

The results table shows the top 10 hashtags by date:

Key	Value	Type
(14) [1 element]	{2 fields}	Object
(15) [1 element]	{2 fields}	Object
(16) [1 element]	{2 fields}	Object
(17) [1 element]	{2 fields}	Object
(18) [1 element]	{2 fields}	Object
(19) [1 element]	{2 fields}	Object
(20) [1 element]	{2 fields}	Object
_id	[1 element]	Array
[0]	1400/03/05	String
results	[10 elements]	Array
[0]	{2 fields}	Object
name	[1 element]	Array
[0]	شام	String
count	39.0	Double
[1]	{2 fields}	Object
name	[1 element]	Array
[0]	لبن	String
count	30.0	Double
[2]	{2 fields}	Object
[3]	{2 fields}	Object
[4]	{2 fields}	Object
[5]	{2 fields}	Object

The screenshot shows the Robo 3T interface. On the left, the sidebar displays a connection named 'sahamyab' with several collections: myresults, retweets, retweets_copy, tweets, tweets_copy, Functions, and Users. The 'tweets' collection is selected. In the main pane, a query named 'Q45.js' is running against the 'tweets' collection on 'localhost:27017'. The query is:

```

    {
      $group : { "_id": {
        "hashtags": "$hashtags",
        "date": "$returnObject.match"
      }, count : {$sum : 1}}
    },
    { $match: { "_id.date" : {
      $gte:"1400/02/01",
      $lt: "1400/04/01"
    } } },
    { $match: { "_id.hashtags" : {$ne:[]} } },
    { $sort: { "count": -1 } },
    { "$group": {
      "_id": "$_id.date",
      "results": {
        "$push": {
          "name": "$_id.name",
          "count": "$count"
        }
      }
    } }
  ]
}
  
```

The results table shows the output of the aggregation. One document is returned, containing an array of objects under the 'results' key. Each object has a 'name' field (an array of strings) and a 'count' field (a double value). The first few results are:

Key	Value	Type
\$_id [1 element]	{ 2 fields }	Object
\$_id [0]	[1 element]	Array
\$_id [0]	1400/03/04	String
results [10 elements]	[10 elements]	Array
results [0]	{ 2 fields }	Object
results [0] [0]	[1 element]	Array
results [0] [0] [0]	مرک	String
results [0] [0] [1]	23.0	Double
results [0] [1]	{ 2 fields }	Object
results [0] [2]	{ 2 fields }	Object
results [0] [3]	{ 2 fields }	Object
results [0] [4]	{ 2 fields }	Object
results [0] [5]	{ 2 fields }	Object
results [0] [6]	{ 2 fields }	Object

خروجی به همراه فیلتر بازه

.6

```

db.tweets.aggregate([
  {
    $addFields: {
      returnObject: {
        $regexFindAll: { input: "$sendTimePersian", regex:
          /(.{10})\s/
        }
      }
    }
  },
  { $group : { "_id": {
      "user": "$senderUsername",
      "date": "$returnObject.match"
    }, count : {$sum : 1}}
  },
  { "$sort": { "count": -1 } },
  { "$group": {
    "_id": "$_id.date",
    "results": {
      "$push": {
        "name": "$_id.name",
        "count": "$count"
      }
    }
  } }
]
)
  
```

```

        "date": "$returnObject.match"
    }, count : {$sum : 1}}
},
{ "$sort": { "count": -1 } },
{ "$group": {
    "_id": "$_id.date",
    "results": {
        "$push": {
            "user": "$_id.user",
            "count": "$count",
        }
    }
}},
{ "$sort": { "_id.0": 1}},
{ "$project": {
    "results": { "$slice": [ "$results", 1 ] }
}
}
]

```

Screenshot of the MongoDB Compass interface showing the results of the aggregation query.

The left sidebar shows the database structure:

- New Connection (4)
 - System
 - config
 - sahamyab
 - Collections (2)
 - tweets
 - tweets_copy
 - Functions (0)
 - Users

The main area displays the aggregation results for the "tweets" collection. The results are as follows:

Key	Value	Type
\$_id (17) [1 element]	{2 fields}	Object
_id [0]	[1 element]	Array
results	[1 element]	Array
[0]	{2 fields}	Object
user	sm65	String
count	1.0	Double
\$_id (18) [1 element]	{2 fields}	Object
_id [0]	[1 element]	Array
results	[1 element]	Array
[0]	{2 fields}	Object
user	cavoshi65	String
count	8.0	Double
\$_id (19) [1 element]	{2 fields}	Object
_id [0]	[1 element]	Array
results	[1 element]	Array
[0]	{2 fields}	Object
user	mashty2019	String
count	8.0	Double
\$_id (20) [1 element]	{2 fields}	Object
_id [0]	[1 element]	Array
results	[1 element]	Array
[0]	{2 fields}	Object
user		String
count		Double

Logs: [empty]

+ گام پنجم:

۳.۱. با توجه به فیلدهایی که جستجو روی آنها صورت میگیرد باید صعودی باشد و همچنین Index only نیز به صورت صعودی اضافه شود تا بتوان `senderName` جستجو کرد.

The screenshot shows the MongoDB Compass interface with two panes. The left pane displays the database structure for 'sahamyab' with 'tweets' collection and its 'Indexes (1)' sub-section containing an index for '_id'. The right pane shows the mongo shell running two commands: creating an index on 'mediaContentType', 'parentId', and 'senderName' with a name 'query for media content type', and then executing a find operation filtering by 'mediaContentType' (image/jpeg), 'parentId' (\$ne:null), and 'senderName' (1). Below the shell, the results of the query are displayed in a table with columns 'Key', 'Value', and 'Type', showing 14 document objects.

Key	Value	Type
{ "1": 1 }	{ "_id": "60adde7582ea55a485554dc5" }	Object
{ "2": 1 }	{ "_id": "60addbd182ea55a485554c62" }	Object
{ "3": 1 }	{ "_id": "60addb5882ea55a485554c08" }	Object
{ "4": 1 }	{ "_id": "60ad38da82ea55a485553181" }	Object
{ "5": 1 }	{ "_id": "60ad258a82ea55a48555298e" }	Object
{ "6": 1 }	{ "_id": "60ad19d82ea55a4855520c1" }	Object
{ "7": 1 }	{ "_id": "60add2d082ea55a48555461f" }	Object
{ "8": 1 }	{ "_id": "60ad373482ea55a485552f0f" }	Object
{ "9": 1 }	{ "_id": "60ad30e982ea55a485552d63" }	Object
{ "10": 1 }	{ "_id": "60ad204a82ea55a4855524f2" }	Object
{ "11": 1 }	{ "_id": "60ad415482ea55a48555390f" }	Object
{ "12": 1 }	{ "_id": "60ad382582ea55a4855530cc" }	Object
{ "13": 1 }	{ "_id": "60ad5cdf82ea55a4855541d1" }	Object
{ "14": 1 }	{ "_id": "60addfa382ea55a485554ea0" }	Object

همانطور که از مقایسه برمی‌آید، یک میلی ثانیه زمان کاهش یافته است.

۳.۲. با توجه به فیلدهایی که عملیات روی آنها صورت میگیرد `hashtags` باید قرار بگیرد. همانطور که از مقایسه برمی‌آید، هفت میلی ثانیه زمان افزایش یافته است. البته این عدد حائز اهمیت نیست. دلیل عدم بهبود با شاخص می‌تواند مربوط به دیدن هر `doc` باشد. (حلقه‌ی `foreach`)

```

db.tweets.createIndex(
  { hashtags: 1 },
  { name: "query for search on hashtags" }
)

0.14 sec.

Key | Value | Type
---|---|---
(1) | { 4 fields } | Object
  | createdCollectionAutomatically | Boolean
  | numIndexesBefore | Int32
  | numIndexesAfter | Int32
  | ok | Double

-----



db.getCollection('tweets').find({}).forEach(function(doc) {
  db.tweets.updateOne({
    id: doc["id"]
  }, {
    $set :{ gov: true}
  });
})

var after = new Date();
var execution_mills = after - before;
print(execution_mills)

```

۴.۱. برای این سوال نیز تنها موردی که به ذهن می‌رسید شاخص بر روی `senderName` است. همانطور که از مقایسه برمی‌آید، زمان تغییری نکرده استدلال عدم بهبود با شاخص می‌تواند مربوط به دیدن هر `doc` باشد. (حلقه‌ی `foreach`)

```

db.tweets.createIndex(
  { senderUserName: 1, },
  { name: "query for search on username" }
)

```

0.155 sec.

Key	Value	Type
(1)	{ 4 fields }	Object


```

db.tweets.aggregate([
  { $group : { _id : '$senderUsername', count : { $sum : 1 } } },
  { $facet: {
    "categorizedByActivity": [
      {
        $bucket: {
          groupBy: "$count",
          boundaries: [1, 2, 4],
          default: "Other",
          output: {
            "count": { $sum: 1 },
            "titles": { $push: "$_id" }
          }
        }
      }
    ]
  }}
])

```

0.006 sec.

Key	Value	Type
(1)	{ 1 field }	Object
[0]	[3 elements]	Array
[0]	{ 3 fields }	Object
[0]._id	1.0	Double
[0].count	486.0	Double
[0].titles	[486 elements]	Array
[0].titles[0]	hadi2222	String
[0].titles[1]	golnardeh2	String
[0].titles[2]	gharibzahedi	String
[0].titles[3]	mlsdg	String
[0].titles[4]	mahdi1545	String
[0].titles[5]	malbakhteh666	String
[0].titles[6]	talon	String
[0].titles[7]	amirkings32	String
[0].titles[8]	alial4880	String
[0].titles[9]	prime1356	String
[0].titles[10]	amiraliagh20	String
[0].titles[11]	ali68131368	String
[0].titles[12]	mfar1	String
[0].titles[13]	alirezaba1985	String
[0].titles[14]	rajo2200	String
[0].titles[15]	sahamyab	String

۴.۵. با توجه به فیلدهایی که عملیات روی آنها صورت میگیرد hashtags باید قرار بگیرد. همانطور که از مقایسه برمیآید، شش میلی ثانیه زمان کاهش یافته است.(تقریباً نصف شده است).

```

    Q45.js [x] Q54.js [x]
    New Connection localhost:27017 sahamyab
    {
      "$group": {
        "_id": {
          "hashtags": "$hashtags",
          "date": "$returnObject.match"
        },
        "count": {$sum: 1}
      },
      "$match": { "_id.hashtags": { $ne: [] } },
      {"$sort": { "count": -1 } },
      {"$group": {
        "_id": "$_id.date",
        "results": {
          "$push": {
            "name": "$_id.hashtags",
            "count": "$count"
          }
        }
      }},
      ...
    }
  
```

Key	Value	Type
> (1) [1 element]	{ 2 fields }	Object
> (2) [1 element]	{ 2 fields }	Object
> (3) [1 element]	{ 2 fields }	Object
> (4) [1 element]	{ 2 fields }	Object
> (5) [1 element]	{ 2 fields }	Object
> (6) [1 element]	{ 2 fields }	Object
> (7) [1 element]	{ 2 fields }	Object
> (8) [1 element]	{ 2 fields }	Object
> (9) [1 element]	{ 2 fields }	Object
> (10) [1 element]	{ 2 fields }	Object
> (11) [1 element]	{ 2 fields }	Object
> (12) [1 element]	{ 2 fields }	Object
> (13) [1 element]	{ 2 fields }	Object
> (14) [1 element]	{ 2 fields }	Object
> (15) [1 element]	{ 2 fields }	Object
> (16) [1 element]	{ 2 fields }	Object
> (17) [1 element]	{ 2 fields }	Object
> (18) [1 element]	{ 2 fields }	Object
> (19) [1 element]	{ 2 fields }	Object
> (20) [1 element]	{ 2 fields }	Object

۴.۶. برای این سوال نیز تنها موردی که به ذهن می‌رسید شاخص بر روی `senderName` است.
همانطور که از مقایسه برمی‌آید، شش میلی ۳ زمان کاهش یافته است. (۲۵ درصد کاهش
یافته است).

```

    Q46.js [x] Q55.js [x]
    New Connection localhost:27017 sahamyab
    db.tweets.createIndex(
      { senderUsername: 1 },
      { name: "query for search on username" }
    )
  
```

Key	Value	Type
> (1)	{ 4 fields }	Object
UF	false	Boolean
numIndexesBefore	4	Int32
numIndexesAfter	5	Int32
ok	1.0	Double

Q46.js

```
db.tweets.aggregate([
  {
    $addFields: [
      returnObject: {
        $regexFindAll: { input: "$sendTimePersian", regex: /.{10}"/ }
      }
    }
  },
  { $group: { "_id": { "user": "$senderUsername", "date": "$returnObject.match" }, count: { $sum: 1 } } },
  { "$sort": { "count": -1 } },
  { "$group": { "_id": "$_id.date", "count": { $sum: 1 } } }
])

```

tweets 0.009 sec.

Key	Value	Type
> (1) [1 element]	{ 2 fields }	Object
> (2) [1 element]	{ 2 fields }	Object
> (3) [1 element]	{ 2 fields }	Object
> (4) [1 element]	{ 2 fields }	Object
> (5) [1 element]	{ 2 fields }	Object
> (6) [1 element]	{ 2 fields }	Object
> (7) [1 element]	{ 2 fields }	Object
> (8) [1 element]	{ 2 fields }	Object
> (9) [1 element]	{ 2 fields }	Object
> (10) [1 element]	{ 2 fields }	Object
> (11) [1 element]	{ 2 fields }	Object
> (12) [1 element]	{ 2 fields }	Object
> (13) [1 element]	{ 2 fields }	Object
> (14) [1 element]	{ 2 fields }	Object
> (15) [1 element]	{ 2 fields }	Object
> (16) [1 element]	{ 2 fields }	Object
> (17) [1 element]	{ 2 fields }	Object
> (18) [1 element]	{ 2 fields }	Object
> (19) [1 element]	{ 2 fields }	Object
> (20) [1 element]	{ 2 fields }	Object

گام ششم:

Q6.js

```
db.tweets.find({type: "retwit"}).count()
db.tweets.find({type: "retwit"}).forEach(
  function(doc) {
    db.retweets.insert(doc)
  }
)
db.retweets.find().count()
db.tweets.remove({type: "retwit"})
db.tweets.find({type: "retwit"}).count()

```

0.002 sec.

69

Q6.js

```
db.tweets.find({type: "retwt"}).count()
db.tweets.find({type: "retwt"}).forEach(
  function(doc) {
    db.retweets.insert(doc)
  }
)
db.retweets.find().count()
db.tweets.remove({type: "retwt"})
db.tweets.find({type: "retwt"}).count()

```

0.037 sec.

Inserted 1 record(s) in 1ms
Inserted 1 record(s) in 1ms

0.037 sec.

Q6.js

```
db.tweets.find({type: "retwt"}).count()
db.tweets.find({type: "retwt"}).forEach(
  function(doc) {
    db.retweets.insert(doc)
  }
)
db.retweets.find().count()
db.tweets.remove({type: "retwt"})
db.tweets.find({type: "retwt"}).count()

```

0 sec.

0 sec.

```

db.tweets.find({type: "retwit"}).count()
db.tweets.find({type: "retwit"}).forEach(
  function(doc) {
    db.retweets.insert(doc)
  }
)
db.retweets.find().count()
db.tweets.remove({type: "retwit"})
db.tweets.find({type: "retwit"}).count()

db.tweets.find({type: "retwit"}).count()
db.tweets.find({type: "retwit"}).forEach(
  function(doc) {
    db.retweets.insert(doc)
  }
)
db.retweets.find().count()
db.tweets.remove({type: "retwit"})
db.tweets.find({type: "retwit"}).count()

```

Removed 69 record(s) in 5ms

0.005 sec.

0.001 sec.

همانطور که مشاهده می‌شود، داده‌ها به collection جدید منتقل شده‌اند و از قبلی حذف شده‌اند.
+ اختیاری:

با استفاده از امکان map_reduce در پایتون mongo این بخش با استفاده از دو تابع map و reduce پیاده‌سازی شد. به این صورت که در مرحله map هر کاربر به همراه یک ۱ آزاد شد و در مرحله reduce پس از جمع شدن این مقادیر میانگین‌شان گرفته شد و در خروجی زیر قابل مشاهده است. کد این قسمت در فایل bonus.py زده شده است.

```

{
  "_id": "iranargoo123", "value": 1.0},
  {"_id": "xczkoldjbdjbe", "value": 1.0},
  {"_id": "emilianozapata", "value": 1.0},
  {"_id": "masuodjhshuwytw", "value": 1.0},
  {"_id": "soltanrah", "value": 1.0},
  {"_id": "akbar2264", "value": 1.0},
  {"_id": "amirariani", "value": 0.0},
  {"_id": "marjanhnj", "value": 1.0},
  {"_id": "vianna", "value": 1.0},
  {"_id": "okramsiid", "value": 1.0},
  {"_id": "mah05", "value": 1.0},
  {"_id": "shayan53", "value": 1.0},
  {"_id": "gholabei378", "value": 1.0},
  {"_id": "ebirahmani", "value": 0.3333333333333333},
  {"_id": "zahrakohansal", "value": 1.0},
  {"_id": "lasthour", "value": 1.0},
  {"_id": "mahnaz1370", "value": 0.0},
  {"_id": "emadgodarzi0313", "value": 1.0},
  {"_id": "baz004", "value": 1.0},
  {"_id": "oligorge", "value": 1.0},
  {"_id": "hkoolak2000", "value": 1.0},
  {"_id": "h400284854", "value": 1.0},
  {"_id": "sottarani", "value": 1.0},
  {"_id": "m_bazad", "value": 0.3333333333333333},
  {"_id": "moeinjenieur", "value": 1.0},
  {"_id": "navid88888888", "value": 0.0},
  {"_id": "sh00459", "value": 1.0},
  {"_id": "reza1919", "value": 1.0},
  {"_id": "rezazal", "value": 1.0},
  {"_id": "mehditb400", "value": 1.0},
  {"_id": "alimahdavy640", "value": 0.5},
  {"_id": "zoroooooooo", "value": 1.0},
  {"_id": "kasmo900", "value": 1.0},
  {"_id": "mfd20864469", "value": 1.0},
  {"_id": "hh1992", "value": 1.0},
  {"_id": "jack1984", "value": 1.0},
  {"_id": "yaserpa1369", "value": 1.0},
  {"_id": "noslem", "value": 1.0},
  {"_id": "fj_fj_123", "value": 0.3333333333333333},
  {"_id": "nada092", "value": 1.0},
  {"_id": "vma1363", "value": 1.0},
  {"_id": "hkazem3000", "value": 0.0},
  {"_id": "admuser072", "value": 1.0},
  {"_id": "sf12161216", "value": 1.0},
  {"_id": "jalalziaei", "value": 1.0},
  {"_id": "konargape", "value": 1.0},
  {"_id": "mortezamsp", "value": 0.0}
}

```

برای خواسته دوم نیز هر کلمه به همراه یک آزاد می‌شود و سپس، در **reducer** جمع زده‌می‌شود. ایست واژه‌های استخراج شد ولی به دلیل عدم آشنایی کافی با **JS**، هر چه تلاش کردم نشد که در کد اثر بگذارند. برای این کار نیز بود قبل از آزادسازی کلمه چک شود که ایست واژه نباشد. همچنین، برای **sort** با استفاده امکان **sort** خود **mongo** این کار صورت گرفته است. اگر بخواهیم از **sort** پشتیبانی شود باید **multi step m/r** داشته باشیم که این کتابخانه این امکان را ندارد. خروجی به صورت زیر درآمد. همچنین، کد این خواسته در فایل **bonus2.py** قرار داده شده است.

```
(base) iaronnbar@Amirs-MBP:Q1(Mongo) % python bonus2.py
[{"_id": "آزاد", "value": 1097.0}, {"_id": "ب\u00e7", "value": 909.0}, {"_id": "ب\u00e7\u00e7", "value": 684.0}, {"_id": "ب\u00e7\u00e7\u00e7", "value": 521.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7", "value": 494.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 465.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 418.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 362.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 336.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 328.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 252.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 195.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 175.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 167.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 166.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 156.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 154.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 149.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 147.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 144.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 141.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 140.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 139.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 135.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 123.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 116.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 112.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 108.0}, {"_id": "ب\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7\u00e7", "value": 104.0}]
```

• بخش دوم – کار با **Neo4j** + گام اول:

نحوه بارگذاری داده‌ها در بخش بعد به طور کامل توضیح داده‌می‌شود. برای آشنایی با زبان سایفر از **movies sandbox** خود سایت استفاده شده است. چهار کوئری زیر به همراه توضیح و خروجی نتیجه این گام بودند.

در اولین و ساده‌ترین درخواست تمام فیلم‌های بعد از سال ۲۰۰۰ بازگردانده شده‌اند که آن‌ها را به صورت تعداد گره می‌بینید. این کار با یک فیلتر ساده انجام پذیرفته است.



خواسته بعدی پنج فیلم اول بر مبنای تعداد نقدهای آن‌هاست. برای این کار تعداد روابط reviewed شمرده شده‌است. بر مبنای آن مرتب سازی صورت گرفته و سپس، تعداد آن‌ها محدود شده‌است.

```

MATCH (m:Movie)←[:REVIEWED]-(u:Person)
WITH m.title AS movie, COUNT(*) AS reviews
RETURN movie, reviews
ORDER BY reviews DESC
LIMIT 5;

```

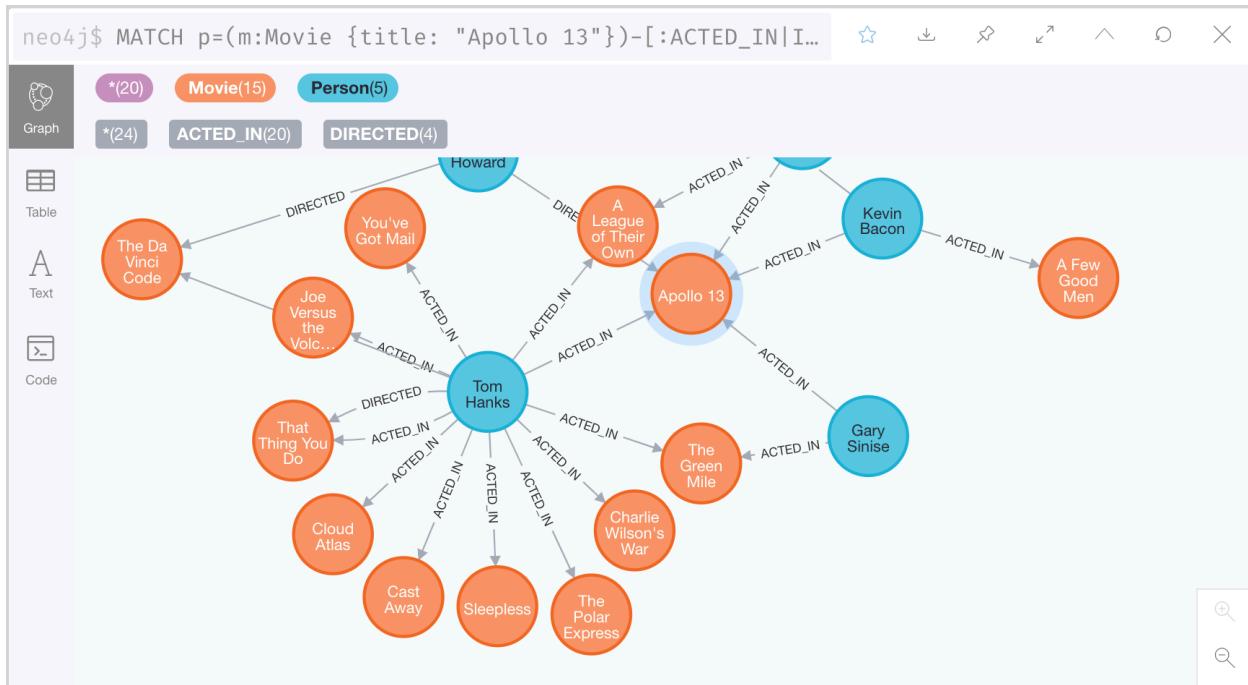
	movie	reviews
1	"The Replacements"	3
2	"The Da Vinci Code"	2
3	"The Birdcage"	1
4	"Cloud Atlas"	1
5	"Unforgiven"	1

خواسته بعد در حقیقت دارد content based recommendation انجام می‌دهد. این خواسته فیلم‌هایی را در می‌آورد که با فیلم Apollo 13 بازیگر، ژانر و یا کارگردان مشترک دارند. برای این کار از وجود دو رابط هم جنس در میانه‌ی راه استفاده می‌کند. به این صورت گویی فیلم‌های مشابه با فیلم موردنظر استخراج می‌شود.

```

MATCH p=(m:Movie {title: "Apollo 13"})-
[:ACTED_IN|IN_GENRE|DIRECTED*2]-()
RETURN p LIMIT 25

```



خواسته بعدی مشابه با این خواسته است با این تفاوت که برای هر نوع رابطه‌ی مشترک امتیاز متفاوتی در نظر گرفته شده است و بر مبنای این موارد امتیاز نهایی محاسبه شده است. سپس، مرتب سازی بر مبنای این امتیاز نهایی صورت گرفته است.

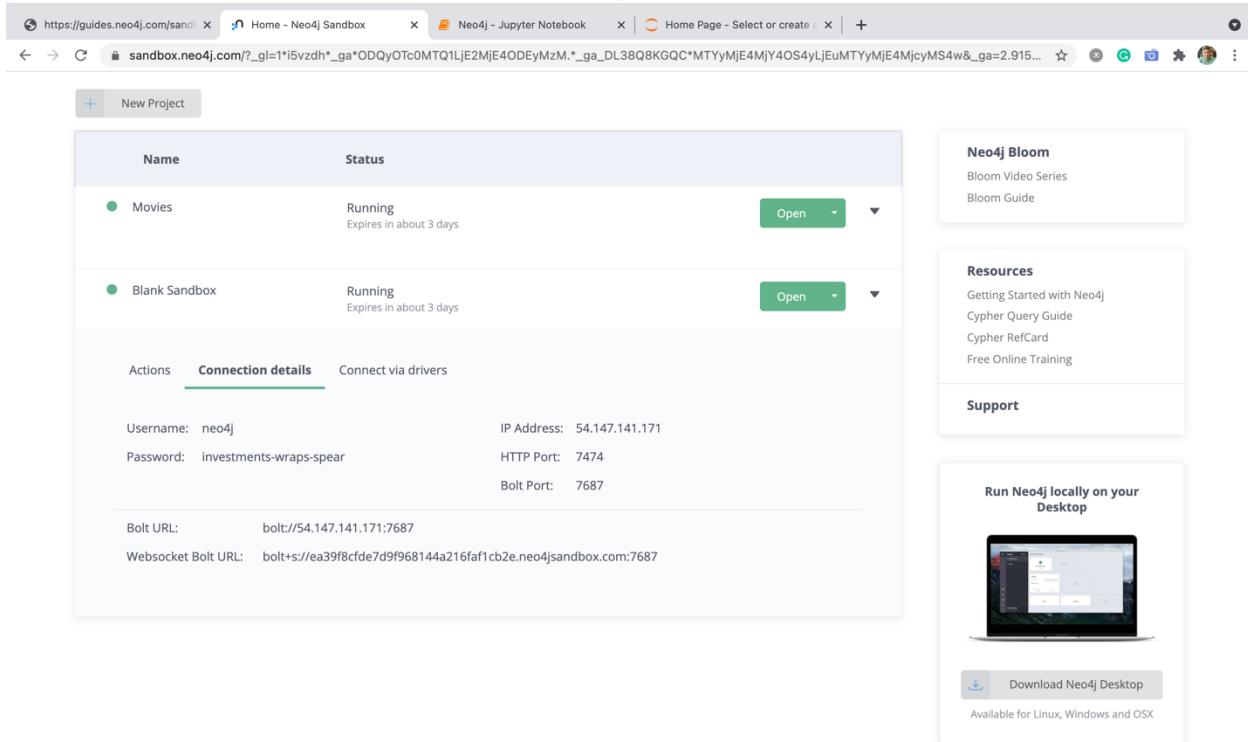
```

1 MATCH (m:Movie) WHERE m.title = "The Replacements"
2 MATCH (m)-[:ACTED_IN]-(a:Person)-[:ACTED_IN]-(rec)
3 WITH m, rec, COUNT(a) AS as
4
5 OPTIONAL MATCH (m)-[:PRODUCED]-(d:Person)-[:PRODUCED]-(rec)
6 WITH m, rec, as, COUNT(d) AS ds
7
8 RETURN rec.title AS recommendation, (3*as)+(4*ds) AS score ORDER BY score DESC LIMIT 100
  
```

recommendation	score
"Something's Gotta Give"	3
"Johnny Mnemonic"	3
"The Devil's Advocate"	3
"The Matrix Revolutions"	3
"The Matrix Reloaded"	3
"The Matrix"	3
"The Replacements"	3
"Cast Away"	3
"Cloud Atlas"	3
"That Thing You Do"	3
"Sleepless"	3
"The Green Mile"	3
"Charlie Wilson's War"	3
"The Polar Express"	3
"Joe Versus the Volcano"	3
"A League of Their Own"	3
"You've Got Mail"	3
"The Da Vinci Code"	3
"Howard"	3
"Kevin Bacon"	3
"A Few Good Men"	3

+ گام دوم:

برای این گام یک sandbox خالی می‌سازیم و اطلاعات credentials آن شامل bolt url و pass در نویت بوک داده شده قرار می‌دهیم.



The screenshot shows the Neo4j Sandbox interface. On the left, a sidebar lists two projects: 'Movies' (Running, Expires in about 3 days) and 'Blank Sandbox' (Running, Expires in about 3 days). Below the sidebar, the 'Connection details' tab is active, showing connection parameters: Username: neo4j, Password: investments-wraps-spear, IP Address: 54.147.141.171, HTTP Port: 7474, Bolt Port: 7687, Bolt URL: bolt://54.147.141.171:7687, and Websocket Bolt URL: bolt+://ea39f8cfde7d9f968144a216faf1cb2e.neo4jsandbox.com:7687. To the right, there are three panels: 'Neo4j Bloom' (Bloom Video Series, Bloom Guide), 'Resources' (Getting Started with Neo4j, Cypher Query Guide, Cypher RefCard, Free Online Training), and 'Support' (Run Neo4j locally on your Desktop, Download Neo4j Desktop, Available for Linux, Windows and OSX).

Put your address and password here.

```
bolt = "bolt://54.147.141.171:7687"
pswrd = "investments-wraps-spear"
graph = Graph(bolt, auth=("neo4j", pswrd))
```

سپس، تمام cell‌ها را اجرا می‌کنیم تا داده‌ها در sandbox موردنظر وارد شوند. این فرآیند کمی طول می‌کشد.

```

In [10]: import_enemy_query = """
MATCH (r:Character)
WHERE exists (r.id)
WITH 'SELECT *'
    WHERE { ?item rdfs:label ?name .
        filter (?item = wd:' + r.id + ')
        filter (lang(?name) = "en" ) .
    OPTIONAL{ ?item wdt:P1830 [rdfs:label ?owner ] .
        filter (lang(?owner) = "en" ) . }
    OPTIONAL{ ?item wdt:P7047 ?enemy })' AS sparql, r
CALL apoc.load.jsonParams( "https://query.wikidata.org/sparql?query=" +
    apoc.text.urlencode(sparql),
    { Accept: "application/sparql-results+json"}, null)
YIELD value
WITH value.r
WHERE value['results'][bindings] <> []
UNWIND value['results'][bindings] as row
FOREACH(ignoreme in case when row['owner'] is not null then [1] else [] end |
    MERGE (c:Item{name:row['owner']}|{value})
    MERGE (r)-[:OWNS_ITEM]->(c))
FOREACH(ignoreme in case when row['enemy'] is not null then [1] else [] end |
    MERGE (c:Character{url:row['enemy']}|{value})
    MERGE (r)-[:ENEMY]->(c))
"""

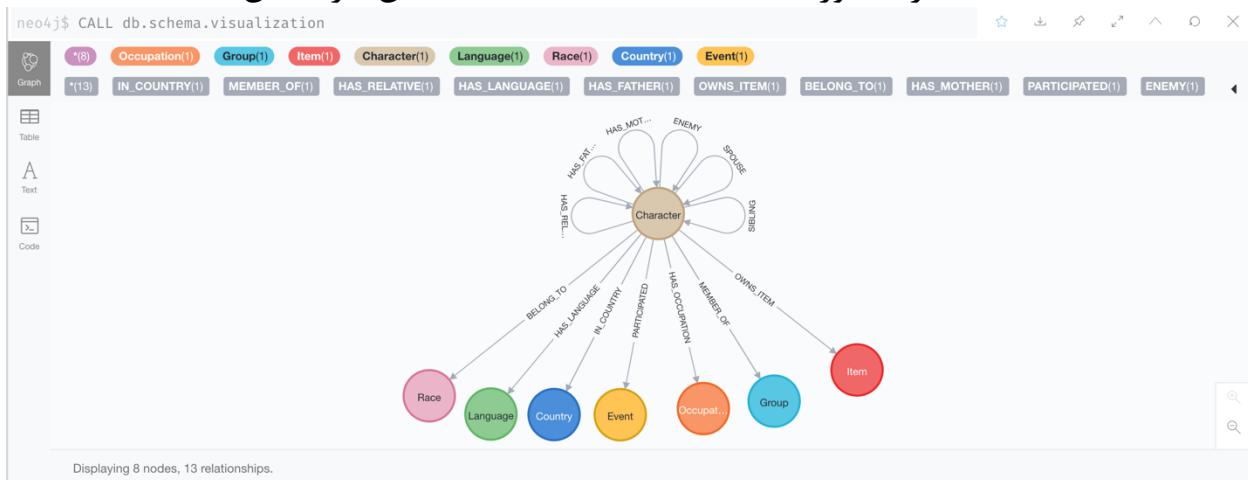
while True:
    try:
        graph.run(import_enemy_query)
        print('Done')
        break
    except:
        print('Please wait...')

Please wait...
Please wait...
Done

```

حال آماده ایم تا درخواست های خود را اجرا کنیم.
+ گام سوم:

1. با استفاده از دستور `schema.visualization` این امر ممکن است.



2. برای این موارد از دستور `count distinct` و `count` استفاده شده است که روی فیلد نام صورت گرفته است. البته این موارد در `with` قرار داده شده اند.

```
neo4j$ MATCH (:Character) WITH Count(*) as char_count RETURN char_count
```

char_count	
1	698

```
neo4j$ MATCH (c:Character) WITH Count(Distinct(c.name)) as char_count RETURN char_count
```

char_count	
1	661

برای این منظور رابطه بودن در سرزمین لحاظ شده است و سپس، این تعداد به عنوان جمعیت شمرده شده است. همچنین مرتب سازی با استفاده از ORDER BY بر روی نام سرزمین صورت گرفته است.

```
neo4j$ MATCH (:Character)-[:IN_COUNTRY]→(co:Country) WITH COUNT(*) AS population, co.name as country RETURN country, population ORDER BY country
```

Server is taking a long time to respond...

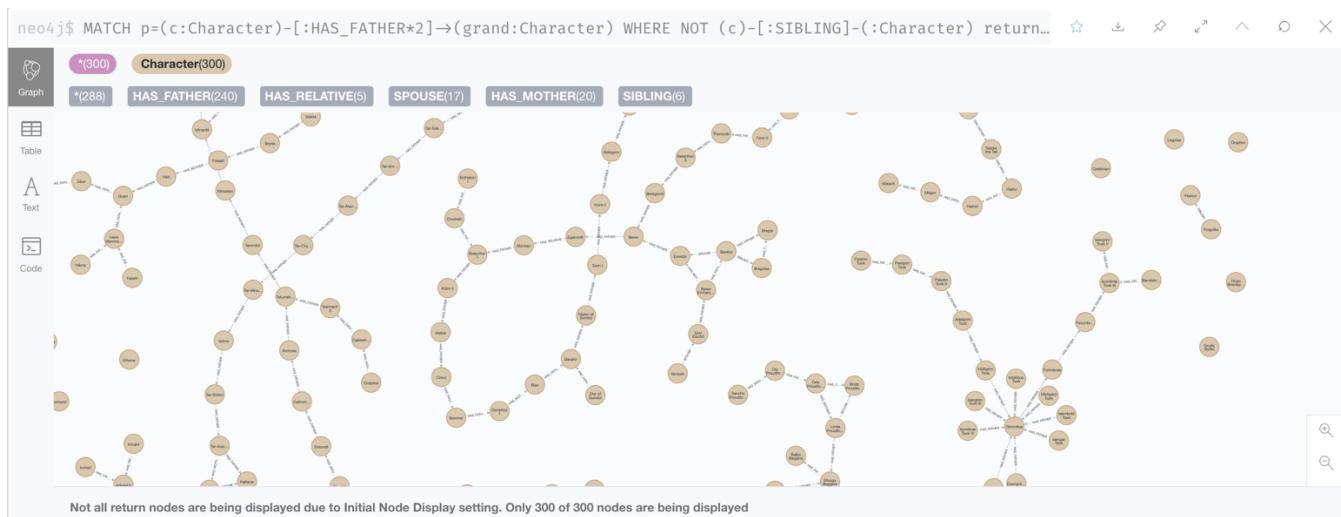
country		population
1	"Arnor"	8
2	"Arthedain"	16
3	"Dale"	2
4	"Doriath"	5
5	"Gondolin"	3
6	"Gondor"	70
7		0

. 4. برای پیدا کردن تک فرزند بودن از WHERE NOT در استفاده می‌کنیم. به این صورت که نداشتن رابط خواهر برادری را بررسی می‌کنیم. برای پیدا کردن نام پدربرگ از دو رابطه has_father استفاده می‌کنیم تا به گره پدربرگ و نام او برسیم.

```
MATCH (c:Character)-[:HAS_FATHER*2]→(grand:Character) WHERE NOT (c)-[:SIBLING]-(:Character) return c.name, grand.name
```

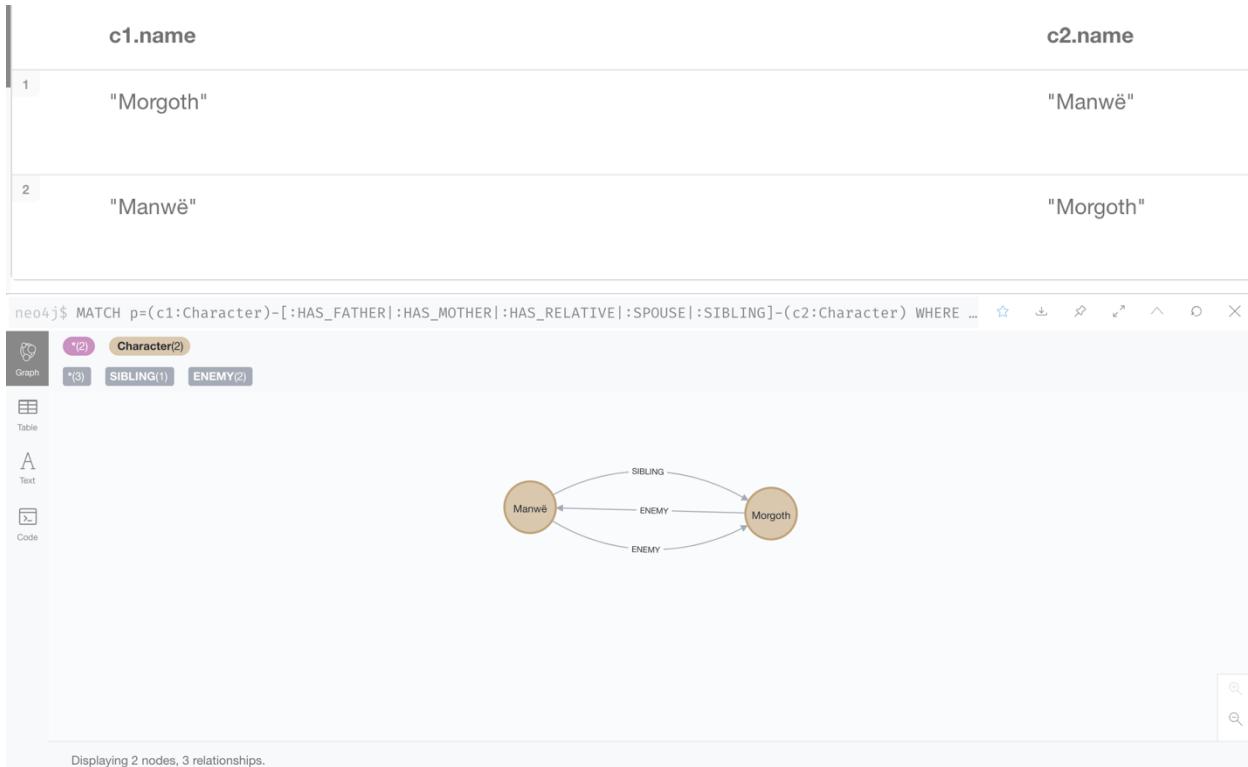
king a long time to respond...

c.name	grand.name
"Sigismund Took"	"Gerontius Took"
"Adalgrim Took"	"Gerontius Took"
"Faramir Took"	"Paladin Took II"
"Frodo Baggins"	"Fosco Baggins"
"Daisy Boffin"	"Fosco Baggins"
"Otto Sackville-Baggins"	"Mungo Baggins"



. 5. برای پیدا کردن رابطه‌ی دشمنی تمام افراد خانواده و آشنایان را در match بررسی کرده‌ایم. سپس، در where وجود رابطه‌ی دشمنی بین این اعضا را بررسی کرده‌ایم.

```
MATCH p=(c1:Character)-[:HAS_FATHER|:HAS_MOTHER|:HAS_RELATIVE|:SPOUSE|:SIBLING]-(c2:Character) WHERE (c1)-[:ENEMY]-(c2) return p
```



البته برای حذف آن با توجه به عدم توانایی تعریف متغیر برای رابطه در WHERE این کوئری در جایگاه where و match و where جابجایی رخ داد و در زیر نتیجه قابل مشاهده است.

```
4j$ MATCH (c1:Character)-[e:ENEMY]-(c2:Character) WHERE (c1)-[:HAS_FATHER|:HAS_MOTHER|:HAS_RELATIVE|:SPOUSE|:SIBLING]-(c2) DELETE e
```

r is taking a long time to respond...

```
$ MATCH (c1:Character)-[e:ENEMY]-(c2:Character) WHERE (c1)-[:HAS_FATHER|:HAS_MOTHER|:HAS_RELATIVE|:SPOUSE|:...
```

Deleted 2 relationships, completed after 66 ms.

```
MATCH p=(c1:Character)-[:HAS_FATHER|:HAS_MOTHER|:HAS_RELATIVE|:SPOUSE|:SIBLING]-(c2:Character) WHERE (c1)...  
(no changes, no records)
```

۶. الف) برای این منظور باید دو رابطه بودن در یک کشور و داشتن شغل شمشیرزنی بررسی شود که به ترتیب در where و match و where بررسی شده‌اند. سپس، تعداد این شمشیرزن‌های هر

سرزمین شمرده شده است. این کار با توجه به بودن سرزمین در match و در خروجی صورت گرفته است. همچنین، یک مرتب سازی بر مبنای این تعداد نیز صورت گرفته است.

```
$ MATCH (c:Character)-[:IN_COUNTRY]→(co:Country) WHERE (c)-[:HAS_OCCUPATION]-(:Occupation {name: 'swordfighter'}) WITH COUNT(*) AS swords, co.name as country RETURN country, swords ORDER BY swords DESC
```

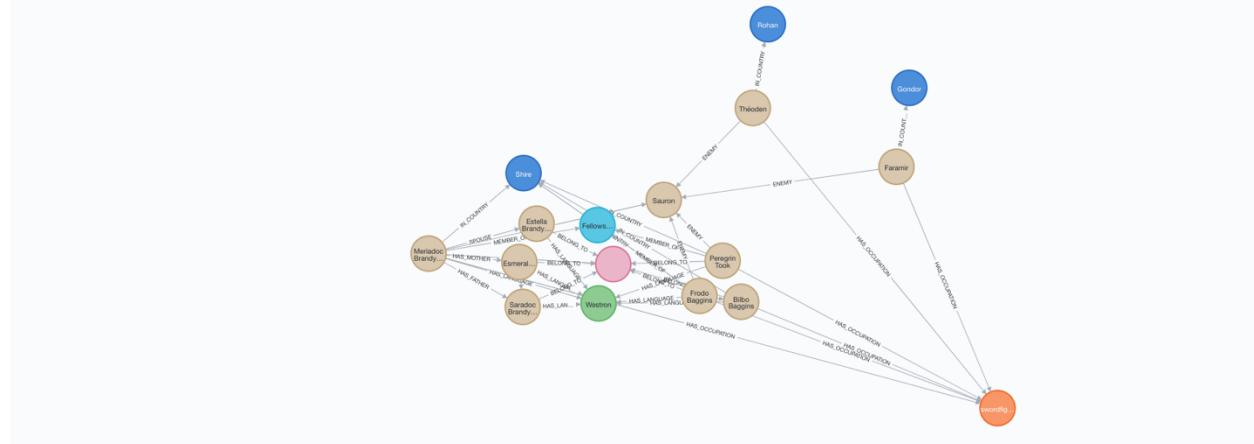
er is taking a long time to respond...

```
$ MATCH (c:Character)-[:IN_COUNTRY]→(co:Country) WHERE (c)-[:HAS_OCCUPATION]-(:Occupation {name: 'swor... ⚡ ↻ ✎
```

country	swords
"Shire"	4
"Gondor"	1
"Rohan"	1

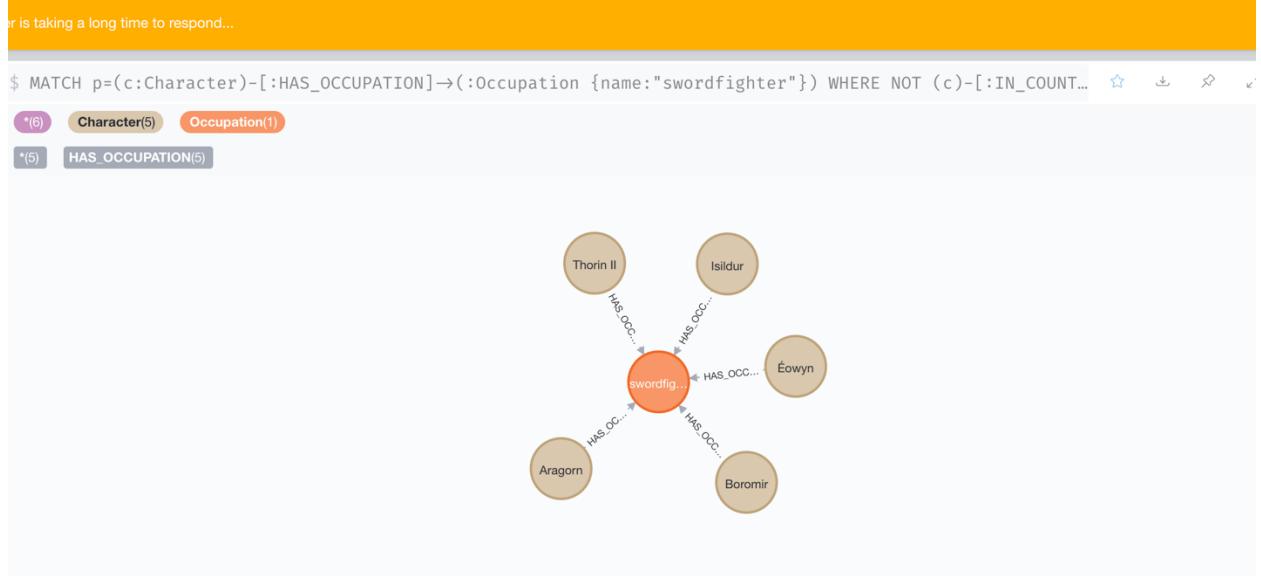
```
MATCH p=(c:Character)-[:IN_COUNTRY]→(co:Country) WHERE (c)-[:HAS_OCCUPATION]-(:Occupation {name: 'sw... H
```

*(17) Character(10) Country(3) Race(1) Language(1) Occupation(1) Group(1)
 (39) IN_COUNTRY(6) HAS_RELATIVE(1) BELONG_TO(7) HAS_MOTHER(1) HAS_FATHER(1) SPOUSE(2) ENEMY(5) HAS_LANGUAGE(7) H



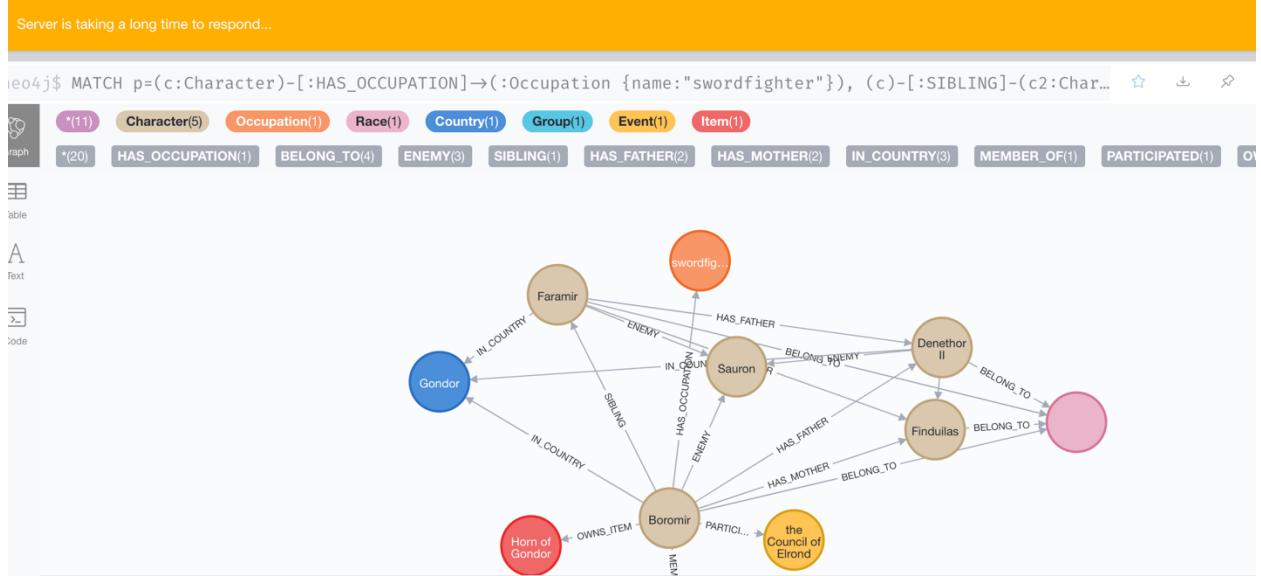
ب) شمشیرزن‌های با بررسی شغل به دست می‌آیند و بدون سرزمین بودن با نداشتم رابطه بودن در سرزمین در where مورد بررسی قرار می‌گیرد.

```
i4j$ MATCH p=(c:Character)-[:HAS_OCCUPATION]→(:Occupation {name:"swordfighter"}) WHERE NOT (c)-[:IN_COUNTRY]→(:Country) return p
```



سپس، با اضافه شدن شرط داشتن رابطه‌ی برادر/خواهری و رابطه سرزمین برادر/خواهر به match می‌توان رابطه سرزمین شخص را اضافه کرد. همانطور که مشاهده می‌کنید، Boromir سرزمین خود را شناخت.

```
1 MATCH p=(c:Character)-[:HAS_OCCUPATION]→(:Occupation {name:"swordfighter"}), (c)-[:SIBLING]-(c2:Character),
(c2)-[:IN_COUNTRY]-(co:Country) WHERE NOT (c)-[:IN_COUNTRY]→(:Country) CREATE (c)-[:IN_COUNTRY]→(co)
2 RETURN p
```



7. برای این درخواست رابطه‌ی بودن در *shire* و سپس رابطه شرکت در رویداد موردنظر بررسی شده است. تنها نکته انتقال تعداد با استفاده از *with* به گام بعدی بوده است. البته بودن در سرزمین در قسمت دوم نیز آمده است تا تعداد درست شمرده شود. انگار در اینجا دو عبارت *match* داشته‌ایم. همانطور که دیدیم تقریباً ۴ درصد اهالی در این رویداد بوده‌اند.

```
i4j$ MATCH (c:Character)-[co:IN_COUNTRY]→(:Country {name: "Shire"}) WITH COUNT(co) as population MATCH
(c2:Character)-[pa:PARTICIPATED]→(:Event {name: "the Council of Elrond"}), (c2)-[:IN_COUNTRY]→(:Country
{name: "Shire"}) WITH population, COUNT(pa) as target RETURN target, population, target*100/population
```

er is taking a long time to respond...

target	population	target*100/population
1	2	48

8. برای این کوئری ابتدا سرزمین‌ها را *match* می‌کنیم تا نام تمام سرزمین‌ها در خروجی بیاید. سپس با استفاده از *optional match* شرط بودن شخصیتی در آن سرزمین که در رویدادی شرکت داشته است را بررسی می‌کنیم ولی به دلیل *optional* بودن در صورت عدم وجود *null* برمی‌گردد. نکته‌ی دیگر استفاده از *distinct* برای یکتا سازی اسم افراد و رویدادها در هر سرزمین است. نکته‌ی دیگر استفاده از *aggregate* است که تمام نتایج را در قالب یک لیست برای هر سرزمین برمی‌گرداند. تعداد نیز با استفاده از *count* شمرده می‌شود.

```

neo4j$ MATCH (country:Country) OPTIONAL MATCH (country)-[:IN_COUNTRY]-(char:Character), (char)-[:PARTICIPATED]→
  (event:Event) WITH country.name AS county, count(distinct char.name) AS count, collect(distinct char.name)
  AS characters, collect(distinct event.name) AS events RETURN county, count, characters, events

```

Server is taking a long time to respond...

county	count	characters	events
"Valinor"	0	[]	[]
"Rivendell"	1	["Erestor"]	["the Council of Elrond"]
"Moria"	0	[]	[]
"Gondor"	4	["Boromir", "Faramir", "Eldacar", "Castamir"]	["the Council of Elrond", "Sauron's attack on Osgiliath", "Faramir's defense of Osgiliath", "Kin-strife"]
county	count	characters	events
"Shire"	2	["Bilbo Baggins", "Frodo Baggins"]	["the Council of Elrond"]
"Gondolin"	0	[]	[]
"Doriath"	0	[]	[]
"Woodland Realm"	0	[]	[]
"Lothlórien"	0	[]	[]
"Nargothrond"	0	[]	[]

• بخش سوم – کار با کاساندرا:

+ گام دوم و سوم:

در کاساندرا متناسب با نیاز باید جدول مورد نیاز طراحی و پر شود. چرا که فقط با primary key آن هم با رعایت توالی می‌توان فیلتر کرد. همچنین عملیات مرتب سازی تنها بر روی cluster key امکان پذیر است.

در این درخواست باید عنوان آلبوم به عنوان primary key قرار بگیرد. همچنین، برای یکتا ماندن داده‌ها به این فیلد track_id نیز افزوده شده است. کد ایجاد این جدول در پوشه مربوط به این بخش در فایل Q1.py آمده است. زمان وارد کردن داده‌ها، اجرای کوئری و نتایج به صورت زیر است.

```
session.execute(
```

```
'CREATE TABLE IF NOT EXISTS Q1 (' +
'title_album text, track_id int, title_track text, date_released 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, track_id))')
```

```
amir@Amir-System:~/Desktop/Q3(cassandra)$ python script.py
100%|██████████| 106574/106574 [00:49<00:00, 2171.47it/s]
Data adding time: 49.08132886886597s
Query time: 0.0014858245849609375s
Take One (Featuring Joey Rippss)
This Is Madness (Just Plain Black) (Featuring Black Liquid)
I'm Not Lyin' (Featuring Octavion Xcellence)
No Title For It (Featuring Obliv and Gordy Michael)
I Listen 2 Classical (Featuring Sleaze)
Request (Featuring Barcodex and SamSun of Photosynthesizers)
Primitve Bandit (Remix) (Featuring Draztig)
Way Back When (Featuring Chuck D)
Some People Never Learn (Featuring NOTE)
These Times (Featuring Sleaze and Joey Rippss)
Out Of Here (Featuring Mic Jordan)
Scream Out (Featuring Braintrust)
```

همانطور که در زیر مشاهده می‌کنید، کوئری موردنظر یک فیلتر ساده روی قسمت اول primary key است.

```
= session.execute('SELECT title_track FROM Q1 WHERE title_album = \'Rumble, Young Man, Rumble\'')
```

در این درخواست باید نام آرتیست به همراه ژانر به عنوان primary key قرار بگیرد. همچنین، برای یکتا ماندن داده‌ها به این فیلدها track_id نیز افزوده شده است. کد ایجاد این جدول در پوشه مربوط به این بخش در فایل Q2.py آمده است. زمان وارد کردن داده‌ها، اجرای کوئری و نتایج به صورت زیر است.

```
session.execute(
    'CREATE TABLE IF NOT EXISTS Q2 (' +
    'title_album text, track_id int, title_track text, date_released text,' +
    'artist text, duration int, genre text, favorites_artist int, listens_track int,' +
    'favorites_track int, listens_album int, favorites_album int,' +
    'PRIMARY KEY(artist, genre, track_id))')
```

```
amir@Amir-System:~/Desktop/Q3(cassandra)$ python Q2.py
100%|██████████| 106574/106574 [00:48<00:00, 2181.58it/s]
Data adding time: 48.85366249084473 s
Query time: 0.0014607906341552734 s
Results:
Welcome!
Egyptian Swaggah
Accidental Skwugg
SwingJeDing
Bleeps Galore
Pumped
Fuck Sidechain Compression on Gameboy
Sweet Self Satisfaction
```

همانطور که در زیر مشاهده می‌کنید، کوئری موردنظر یک فیلتر ساده روی قسمت اول و دوم primary key است.

```
rows = session.execute('SELECT title_track FROM Q2 WHERE artist = \'RoccoW\' AND genre=\'\''')
```

برای پیش پردازش و ورود داده‌ها از کد پایتون استفاده شده است. به همین دلیل فیلدهای خالی به صورت "ذخیره شده‌اند".

در این درخواست ژانر، طول آهنگ و سال به عنوان primary key قرار بگیرد. همچنین، برای یکتا ماندن داده‌ها به این فیلدها track_id نیز افزوده شده است. کد ایجاد این جدول در پوشه مربوط به این بخش در فایل Q3.py آمده است.

```

session.execute(
'CREATE TABLE IF NOT EXISTS Q3 (' +
'title_album text, track_id int, title_track text, year_released int, month_released int, day_released int,' +
'artist text, duration int, genre text, favorites_artist int, listens_track int,' +
'favorites_track int, listens_album int, favorites_album int,' +
'PRIMARY KEY(duration, genre, year_released, track_id))')

```

همانطور که مشاهده می‌کنید، ترتیب به این صورت است چراکه فیلد خالی نمی‌تواند به عنوان قرار بگیرد. (genre) همچنین در Partition key cluttering key با استفاده از range باید عقب‌تر قرار بگیرد.
زمان وارد کردن داده‌ها، اجرای کوئری و نتایج به صورت زیر است.

```

100%| Data adding time: 50.54932188987732 s | 106574/106574 [00:50<00:00, 2108.45it/s
Query time: 0.023351192474365234 s
Results:
Souvenirs, Novelties, Party Tricks
Strideless Pride (Sloth Lust Greed Demo)
Sea And Pea
Spot Rockers (Gotcha Man Mix)
Anthen Eater (TJ Nightmare Remix)
Footprints
Mandingo Ocean
Homemade Raps
Take Your Poison
Liars & Conmen
Gimme That Tape!
Instructions
Welcome To Chi-Raq Outtro
Hey You Guys (Acapella)
The Raps Well (Album Version Acapella)
Underground Press featuring Timezone LaFontaine (00 Mixx)
drunkenCPU
Armchair Revolution
Tiny Town
The 1 Upper (84 Mixx)
Intro
GoodLookN
The BlizBoxx
God's Radio Hell
Eat Some Cake
Funky Motion (Motion Lotion Mixx)
Welcome To Chi-Raq Intro
Dirty Old Clown
I.D.K.?
Pusher For Record
Daddy's Drinkin' Song
Fox 103
Jah Rastafari Loop Dub
Sara Sampao
Spot Rockers
Spot Rockers (Doc's 2-Live Mixx)
Bad Cop (Instrumental)
Funky Motion (Album Version Acapella)
Silverdirt
prometheus
J.J.'s Joint
goldheart
Important Famous Guy
Life Is Good
The 1 Upper (BreakBeat Mixx)
Gruta
Idle Hands
Fireheart

```

She Left Me For A Yeti (Wanda's Gone)
Dead Homie Intro
Hey, Guys
Poetikal Refugee ft. Mikey Krumins aka Dyems
The 1 Upper
Michael Jordan
Underground Press (OG Instrumental)
Thickey Savey
Revenge Of Tiny Town
SpaceDoubt (Previously Unreleased)
These Dreams And Shit
ET The Extra Testicle
Homie Stray
Funky Motion (TSPMIX)
Funky Motion (TSPMental)
The Rebel featuring Cheese N Pot-C
#MakeAChange
Spot Rockers (Blank Mixx)
Mic Ale Jay Scotch
Back To It (12inchstrumental)
Burns (Featuring Cable)
Hey Fuckface!
Kids Room
Garbage
Before The Devil Knows You're Dead
Soulbox
Floridian Fighting Bear
C-N-P Enuff
Fallin' Apart
Musicbox
54%
Stay Fly
Suggestion Rocks
Money Beets: A Tribute to RHP (Featuring Milkdrop)
Max Awe
activate
TheEnsuranceTrap featuring Tab (PMRC Mix)
TheEnsuranceTrap (Tab's Instrumental)
pathetic at best
Afterschool Speshul Bonus Beats
crumbs
Anything 2000
Kick featuring Flatline
Phon's Waltz
The Roses
Homie Jota
May 9th
Back Against Interlude
Whentro
Lyrically Spoken
Bad Cop
PS Love Always
Til It Hurts (Binky Mixx)

همانطور که در زیر مشاهده می‌کنید، کوئری موردنظر یک range کوئری، یک فیلتر و یک Allow Filtering ترتیب موردنظر برای اجرای دستور است.

```
rows = session.execute(['SELECT title_track FROM Q3 WHERE duration < 180 AND genre = \'Hip-Hop\' AND '+
|   ' year_released >= 2015 AND year_released <= 2016 ALLOW FILTERING'])
```

روز، ماه و سال تولید آهنگ از روی فیلد تاریخ جداسده‌اند و بیرون کشیده‌شده‌اند و جداگانه ذخیره شده‌اند چراکه هم در این کوئری و هم در کوئری‌های بعدی به آن‌ها نیاز داریم.

در این درخواست ژانر، ماه و تعداد شنیده‌شدن به عنوان primary key قرار 4.

بگیرد. همچنین، برای یکتا ماندن داده‌ها به این فیلدها track_id نیز افزوده شده است.

کد ایجاد این جدول در پوشه مربوط به این بخش در فایل Q4.py آمده است.

```
session.execute(
    'CREATE TABLE IF NOT EXISTS Q4 (' +
    'title_album text, track_id int, title_track text, year_released int, month_released int, day_released int,' +
    'artist text, duration int, genre text, favorites_artist int, listens_track int,' +
    'favorites_track int, listens_album int, favorites_album int,' +
    'PRIMARY KEY(month_released, genre, listens_track, track_id))')
```

همانطور که مشاهده می‌کنید، ترتیب به این صورت است چراکه فیلد خالی نمی‌تواند به عنوان Partition key قرار بگیرد. همچنین در clustering key (genre) با استفاده از range باید عقب‌تر قرار بگیرد.

زمان وارد کردن داده‌ها، اجرای کوئری و نتایج به صورت زیر است.

```
00% | ata adding time: 51.08044362068176 s
query time: 0.005644083023071289 s
results:
obbie
ough Sex
elopponnes
ff In The Distance
ow on Ideas
mbrella Pete Strut
andy Funk
nce Upon A Time
arathon
odka Bitter Smartie Party (Ergo Phizmiz' version)
o I Went Home
low jam
ick Your Brain
ocolate Sky
escalito
morphheus (2b20 rebuild)
don't know
ilboquet (La Mélodie Jetable)
r. Pig & Mr. Chrome
oodbye
.
el Vuoto
reaks
ising in the Wake of Supernova
ismantling Earth Based Authority Via Satellite
rack 3
olan Thing 3
ausea
Track 2]
Track 1]
abric
ogue
ightly nice
leep Just To Dream
tright Blimpin'
luml
vening Capital of World
odka Alone (Vernon's version)
err
idget Madness
'm Not An Animal
oz
pot
ntitled3
pocalyptic Samurai Drift Break
he Descent Into The Abyss
ime: disabled (Step by Step edit)
bsolute
```

Fin
Yeah
found (vip)
The Psychedelic And
The Lab Is Working
October
Eu sou o fado
found
fvneral blves
Monster
Sweet Shop
Monument
Beats tornado
Runner
Why does my heart...
fvneral blves (kinlaw remix)
The And Of The World
Zombie Disco
Breacking down
Pour Petite Tete
She Does Her Best (feat. Small Colin)
Chicken and Cheese 2 (Foot Village cover)
Idea
Ode To A Baby Snowstorm
Old Family Photo Album
Springa Springa
Signs
Discrete Forests
A Place To Call Home
Moni Mon Amie
Bridges
My Best Friend
India
Marienbad
Brute (One Remix)
Moi Samolety
Our Sorrows
Stinger Sisters
Brute
probably digging my own grave
Nevesta
When The Sea Called Me Home
gneiss
Hey, Predator Drone
Nevesta (Flayve Remix)
Giraffes
Betsy On The Roof
Brute (Intro)
Discount Store
Study For A Pop Band
that face you wear
Malta
Mirrors

همانطور که در زیر مشاهده می‌کنید، کوئری موردنظر یک فیلتر کوئری، یک کوئری IN (یا همان OR) و یک کوئری است.

```
'SELECT title_track FROM Q4 WHERE month_released=4 AND genre in (\'Pop\', \'Electronic\') AND listens_track>300')
```

روز، ماه و سال تولید آهنگ از روی فیلد تاریخ جداسده‌اند و بیرون کشیده شده‌اند و جداگانه ذخیره شده‌اند چراکه هم در این کوئری و هم در کوئری‌های بعدی به آن‌ها نیاز داریم.
6. در این درخواست ماه عرضه، ژانر و تعداد شنیده شدن به عنوان primary key

قرار بگیرد. همچنین، برای یکتا ماندن داده‌ها به این فیلدات track_id نیز افزوده شده‌است. کد ایجاد این جدول در پوشه مربوط به این بخش در فایل Q6.py آمده است.

```
CREATE TABLE IF NOT EXISTS Q6 (' +  
'title_album text, track_id int, title_track text, year_released int, month_released int, day_released int,' +  
'artist text, duration int, genre text, favorites_artist int, listens_track int,' +  
'favorites_track int, listens_album int, favorites_album int,' +  
'PRIMARY KEY(year_released, genre, month_released, track_id))')
```

همانطور که مشاهده می‌کنید، ترتیب به این صورت است چراکه فیلد خالی نمی‌تواند به عنوان Partition key قرار بگیرد. همچنین در clustering key (genre) با استفاده از range باید عقب‌تر قرار بگیرد.

زمان وارد کردن داده‌ها، اجرای کوئری و نتایج به صورت زیر است.

```
100% |  
Data adding time: 47.84368896484375 s | 106574/106574 [00:47<00:00, 2227.64it/s]  
Query time: 0.004810810089111328 s  
Results:  
98
```

همانطور که مشاهده می‌کنید تنها نکته این کوئری استفاده از count aggregate function است.

```
'SELECT COUNT(*) AS song_count FROM Q6 WHERE year_released=2008 AND genre = \'Folk\' AND month_released<7'
```

7. در این درخواست ژانر و طول آهنگ به عنوان primary key قرار بگیرد. همچنین، برای یکتا ماندن داده‌ها به این فیلدات track_id نیز افزوده شده‌است. کد ایجاد این جدول در پوشه مربوط به این بخش در فایل Q7.py آمده است.

```
session.execute(  
'CREATE TABLE IF NOT EXISTS Q7 (' +  
'title_album text, track_id int, title_track text, year_released int, month_released int, day_released int,' +  
'artist text, duration int, genre text, favorites_artist int, listens_track int,' +  
'favorites_track int, listens_album int, favorites_album int,' +  
'PRIMARY KEY(genre, duration, track_id))' +  
'WITH CLUSTERING ORDER BY (duration DESC);')
```

در اینجا ژانرهای خالی ignore شده‌اند چراکه در این جنس درخواست‌ها کارآمد نیستند ولی در این حالت امکان قرارگیری partition key genre به عنوان partition key را به ما می‌دهند.

از نکات مهم دیگری که در این سوال وجود دارد نیاز به مرتب بودن داده‌ها بر اساس زمان آهنگ است. راهکار کاساندرا برای این مورد اجرای Clustering order by clustering key در هنگام تعریف جدول است که در کد بالا دیده‌می‌شود. این مرتب سازی تنها بر روی clustering key امکان‌پذیر است.

زمان واردکردن داده‌ها، اجرای کوئری‌ها و نتایج به صورت زیر است.

```
amir@Amir-System:~/Desktop/03(cassandra)$ python Q7.py
100%|██████████| 106574/106574 [00:26<00:00, 3965.86it/s]
Data adding time: 26.87518286705017 s
Average Query time: 0.1480410099029541 s
Results:
Average songs duration: 268
Query time: 0.0015337467193603516 s
Results:
Track: Live Set, Length: 1557
Track: Funk Shoppe, Length: 710
Track: Little Business, Length: 681
Track: Rайдин' The cCM Century, Length: 607
Track: DK's song 'untitled' (live at WFMU Record Fair), Length: 540
Track: You'll Never Get Away With That, Length: 478
Track: Life After Sundown, Length: 477
Track: Sugar & Whitebread (instrumental), Length: 475
Track: Great Dark Spot, Length: 464
Track: Interview, Length: 459
```

تنها نکته کوئری اول استفاده از تابع AVG یا همان میانگین گیری است.

`rows = session.execute('SELECT AVG(duration) AS average FROM Q7')`

برای کوئری دوم نیز تنها باید یک LIMIT 10 تایی اجرا کرد چرا که داده‌ها در هنگام ورود بر مبنای طول آهنگ که clustering key بوده‌است به صورت نزولی مرتب شده‌اند.

`rows = session.execute('SELECT title_track, duration FROM Q7 LIMIT 10')`

8. در این درخواست سال تولید و ژانر به عنوان partition key و clustering key به عنوان favorites_album ماندن داده‌ها به این فیلدها track_id نیز افزوده شده‌است. کد ایجاد این جدول در پوشه مربوط به این بخش در فایل Q8.py آمده‌است.

```
session.execute(
    'CREATE TABLE IF NOT EXISTS Q8 (' +
    'title_album text, track_id int, title_track text, year_released int, month_released int, day_released int,' +
    'artist text, duration int, genre text, favorites_artist int, listens_track int,' +
    'favorites_track int, listens_album int, favorites_album int,' +
    'PRIMARY KEY((year_released, genre), favorites_album, track_id))' +
    'WITH CLUSTERING ORDER BY (favorites_album DESC);')
```

همانطور که مشاهده می‌کنید، ترتیب به این صورت است چراکه فیلد خالی نمی‌تواند به تنها‌یی به عنوان Partition key قرار بگیرد. (genre)

از نکات مهم دیگری که در این سوال وجود دارد نیاز به مرتب بودن داده‌ها بر اساس تعداد محبوبیت آلبوم است. راهکار کاساندرا برای این مورد اجرای Clustering order by clustering key در هنگام تعریف جدول است که در کد بالا دیده‌می‌شود. این مرتب سازی تنها بر روی clustering key امکان‌پذیر است.

زمان واردکردن داده‌ها، اجرای کوئری‌ها و نتایج به صورت زیر است.

100% | 106574/106574 [00:48<00:00, 2210.18it/s]

Data adding time: 48.221632957458496 s
Query time: 0.004401206970214844 s
Results:

```

Album: The Great Cold, Favorites Counts: 3, Album Artist Favorites: 1
Album: Errors and Omissions, Favorites Counts: 3, Album Artist Favorites: 6
Album: FrostWire Creative Commons Mixtape Vol. 5, Favorites Counts: 3, Album Artist Favorites: 0
Album: Live at WFMU with Marty McSorley, 2/19/2016, Favorites Counts: 2, Album Artist Favorites: 0
Album: netBloc Vol. 51: Embark!, Favorites Counts: 2, Album Artist Favorites: 1
Album: Live at Monty Hall, 5/26/2016, Favorites Counts: 2, Album Artist Favorites: 1
Album: Without Borders: On Behalf of Refugees, Favorites Counts: 2, Album Artist Favorites: 0
Album: Medo de Morrer | Medo de Tentar, Favorites Counts: 2, Album Artist Favorites: 3
Album: Live on WFMU's Three Chord Monte with Joe Belock - November 9, 2015, Favorites Counts: 1, Album Artist Favorites: 1
Album: Live at WFMU on the Evan "Funk" Davies Show, 1/20/2016, Favorites Counts: 1, Album Artist Favorites: 1
Album: Live on WFMU with Liz Berg: Jan 19, 2016, Favorites Counts: 1, Album Artist Favorites: 2
Album: netBloc Vol. 50: superSonic!, Favorites Counts: 1, Album Artist Favorites: 0
Album: Live on WFMU's Surface Noise with Joe McGasko, 1/25/2016, Favorites Counts: 1, Album Artist Favorites: 0
Album: ca suffa comme ç!, Favorites Counts: 1, Album Artist Favorites: 1
Album: Live on WFMU's Three Chord Monte with Joe Belock - April 25, 2016, Favorites Counts: 1, Album Artist Favorites: 1
Album: Live on WFMU's Undancing In The Dirt with Thomas Storck: June 14, 2016, Favorites Counts: 1, Album Artist Favorites: 2
Album: Live on WFMU's Surface Noise with Joe McGasko - July 18, 2016, Favorites Counts: 1, Album Artist Favorites: 7
Album: Live Den, Favorites Counts: 1, Album Artist Favorites: 1
Album: Cryptids, Favorites Counts: 1, Album Artist Favorites: 3
Album: Live on WFMU's The Evan "Funk" Davies Show - June 8, 2016, Favorites Counts: 1, Album Artist Favorites: 1
Album: The Spellbreaker EP, Favorites Counts: 1, Album Artist Favorites: 45
Album: Live at WFMU on What Was Music? with Marcel M, 10/19/16, Favorites Counts: 1, Album Artist Favorites: 1
Album: Live at Kanal 103, Favorites Counts: 1, Album Artist Favorites: 6
Album: Live at WFMU's Monty Hall, 5/23/2016, Favorites Counts: 1, Album Artist Favorites: 2
Album: Legally Blind, Favorites Counts: 0, Album Artist Favorites: 3
Album: Quiet the Mind (DCV), Favorites Counts: 0, Album Artist Favorites: 17
Album: The Good Die Young, Favorites Counts: 0, Album Artist Favorites: 13
Album: River Cult, Favorites Counts: 0, Album Artist Favorites: 3
Album: Live on WFMU's Prove It All Night Show with Pat Byrne 1/30/16, Favorites Counts: 0, Album Artist Favorites: 1
Album: Ceremony X, Favorites Counts: 0, Album Artist Favorites: 1
Album: Бывал и БылХед, БылХед и Брокенхард, Favorites Counts: 0, Album Artist Favorites: 3
Album: The Hidden Album, Favorites Counts: 0, Album Artist Favorites: 8
Album: Spare Succulence, Favorites Counts: 0, Album Artist Favorites: 0
Album: Circling The Sun, Favorites Counts: 0, Album Artist Favorites: 6
Album: The New Geometry, Favorites Counts: 0, Album Artist Favorites: 0
Album: The Long Trick, Favorites Counts: 0, Album Artist Favorites: 1
Album: Mountain of the Moon, Favorites Counts: 0, Album Artist Favorites: 41
Album: PUNK Rock Opera , Vol. 1, Favorites Counts: 0, Album Artist Favorites: 4
Album: Live on WFMU's Three Chord Monte with Joe Belock - April 11, 2016, Favorites Counts: 0, Album Artist Favorites: 1
Album: Live At Monty Hall - Feb 12, 2016, Favorites Counts: 0, Album Artist Favorites: 9
Album: Sireano, Favorites Counts: 0, Album Artist Favorites: 1
Album: Live on WFMU with Todd-o-phonics Todd - March 26, 2016, Favorites Counts: 0, Album Artist Favorites: 1
Album: Live on WFMU's 100% Whatever with Mary Wing - May 1, 2016, Favorites Counts: 0, Album Artist Favorites: 1
Album: MMXVI, Favorites Counts: 0, Album Artist Favorites: 4
Album: AS220 Foo Fest 2016 Sampler, Favorites Counts: 0, Album Artist Favorites: 7

```

برای کوئری موردنظر کافی است که بر مبنای سال و ژانر فیلتر کنیم و خروجی موردنظر را بدهیم چرا که بر مبنای محبوبیت آلبوم به صورت نزولی مرتب شده است. همچنین تعداد محبوبیت آرتیست آلبومها نیز خروجی داده شده است. فقط این خروجی duplicate زیادی دارد برای حذف این موارد نمی‌توان از distinct استفاده کرد چرا که این عملگر فقط بر روی partition قابل انجام است. به همین، این منطق در کد پایتون با استفاده از یک مجموعه انجام شده است تا خروجی مفیدتر باشد.

```
('SELECT title_album, favorites_album, favorites_artist FROM Q8 WHERE year_released=2016 AND genre = \'Rock\'')
```

9. این درخواست را می‌توان پیچیده‌ترین این مجموعه دانست. برای خروجی دادن این اطلاعات آماری از یک جدول کناری استفاده شده است که تعداد آهنگ‌های هر ژانر را نگه می‌دارد. البته این پایان ماجرا نیست. پس یک جدول اصلی داریم که مثلاً ژانر و primary key به عنوان track_id قرار گرفته‌اند. یک جدول میانی داریم چراکه جدول با اطلاعات آماری نیاز به به روزرسانی تعداد دارد. از طرفی مایلیم تا بررسی count مرتباً سازی انجام دهیم پس باید clustering key ما باشد. در کاساندرا نمی‌توان متغیر counter با یک جمع زد و باید حتماً نوع counter تعریف شود. از طرفی clustering key نمی‌تواند به عنوان primary key قرار بگیرد. در نتیجه، در این جدول میانی صرفاً ژانر به عنوان primary key قرار می‌گیرد و اطلاعات این جدول با هر بار ورود داده به روزرسانی می‌شود و counter تعداد را می‌شمارد. در انتهایا با یک کوئری این مقادیر به جدول نهایی که int count است که clustering key واقع شده و مرتب‌سازی نزولی نیز دارد منتقل می‌شوند. البته در کاساندرا دستور ساده‌ای برای این کار وجود ندارد.

دستور COPY نیز تنها در cqlsh است و نمی‌توان از آن در CQL استفاده کرد. بنابراین با یک کوئری زد و تک به تک سطرها را منتقل کرد.

البته داستان به این جا ختم نمی‌شود چرا مرتب‌سازی در هر partition انجام می‌شود و در اینجا partition key ماژانر است و به همین دلیل، نتیجه مرتب‌شده واقعی نخواهد بود. ایده‌ای که اینجا زدم قرار دادن یک ۱ به عنوان partition key بود تا تمام داده‌ها هم partition شوند و مرتب‌سازی نهایی به درستی صورت پذیرد. با توجه به توضیحات بالا جداول به صورت زیر خواهند بود. کد ایجاد این جداول در پوشه مربوط به این بخش در فایل Q9.py آمده است.

```
session.execute(
    'CREATE TABLE IF NOT EXISTS Q91 (' +
    'title_album text, track_id int, title_track text, year_released int, month_released int, day_released int,' +
    'artist text, duration int, genre text, favorites_artist int, listens_track int,' +
    'favorites_track int, listens_album int, favorites_album int,' +
    'PRIMARY KEY(genre, track_id));')

session.execute(
    'CREATE TABLE IF NOT EXISTS Q92 (' +
    'genre text, music_count counter, ' +
    'PRIMARY KEY(genre))')

session.execute(
    'CREATE TABLE IF NOT EXISTS Q93 (' +
    'genre text, music_count int, same_partition int,' +
    'PRIMARY KEY(same_partition, music_count, genre))' +
    'WITH CLUSTERING ORDER BY (music_count DESC);')

insert_query = "INSERT INTO Q91(title_album, track_id, title_track, year_released, month_released, day_released," + \
    "artist, duration, genre, favorites_artist, listens_track," + \
    "favorites_track, listens_album, favorites_album) VALUES" + \
    "(\\"{}\",{},\\\"{}\",{},\\\"{}\",{},\\\"{}\",{},\\\"{}\",{},\\\"{}\",{},\\\"{}\",{})".format(
        title_album, track_id, title_track, year_released, month_released, day_released,
        artist, duration, genre, favorites_artist, listens_track,
        favorites_track, listens_album, favorites_album)
if genre == '':
    continue
session.execute(insert_query)

# insert_statistics = "INSERT INTO Q92(genre, music_count) VALUES" + \
#     "(\\"{}\",{}) IF NOT EXISTS;".format(genre, 1)
# session.execute(insert_statistics)
update_statistic = "UPDATE Q92 " + \
    "SET music_count = music_count + 1 " + \
    "WHERE genre = \\"{}\";".format(genre)
session.execute(update_statistic)

# session.execute('COPY Q92(genre, music_count)' + \
#     'TO \'q92.csv\' WITH HEADER = TRUE ;')
# session.execute('COPY Q93(genre, music_count)' + \
#     'FROM \'q92.csv\' WITH HEADER = TRUE ;')
rows = session.execute('SELECT genre, music_count FROM Q92')
for row in rows:
    genre, count = row.genre, row.music_count
    insert_statistics = "INSERT INTO Q93(genre, music_count,same_partition) VALUES" + \
        "(\\"{}\",{},1)".format(genre, count)
    session.execute(insert_statistics)
```

زمان وارد کردن داده‌ها، اجرای کوئری‌ها و نتایج به صورت زیر است.

```

100% | 106574/106574 [00:45<00:00, 2349.68it/s]
Data adding time: 45.36988878250122 s
Query time: 0.0011479854583740234 s
Results:
Genre: Rock, Music Count: 14182
Genre: Experimental, Music Count: 10608
Genre: Electronic, Music Count: 9372
Genre: Hip-Hop, Music Count: 3552
Genre: Folk, Music Count: 2803
Genre: Pop, Music Count: 2332
Genre: Instrumental, Music Count: 2079
Genre: International, Music Count: 1389
Genre: Classical, Music Count: 1230
Genre: Jazz, Music Count: 571
Genre: Old-Time / Historic, Music Count: 554
Genre: Spoken, Music Count: 423
Genre: Country, Music Count: 194
Genre: Soul-RnB, Music Count: 175
Genre: Blues, Music Count: 110
Genre: Easy Listening, Music Count: 24

```

در اینجا ژانرهای خالی ignore شده‌اند چراکه در این جنس درخواست‌ها کارآمد نیستند ولی در این حالت امکان قرارگیری genre به عنوان partition key را به ما می‌دهند که به آن نیاز داریم.

با توجه به آماده کردن داده‌ی مورد نیاز در جدول کافی است ژانر و تعداد را از هر کدام بیرون بکشیم.

```
( 'SELECT genre, music_count FROM Q93'
```

1. در این درخواست سال تولید و تعداد شنیده‌شدن آلبوم به عنوان Primary key باید قرار بگیرد. همچنین، برای یکتا ماندن داده‌ها به این فیلدها track_id نیز افزوده شده‌است. کد ایجاد این جدول در پوشه مربوط به این بخش در فایل Q10.py آمده‌است.

```
session.execute(
    'CREATE TABLE IF NOT EXISTS Q10 (' +
    'title_album text, track_id int, title_track text, year_released int, month_released int, day_released int,' +
    'artist text, duration int, genre text, favorites_artist int, listens_track int,' +
    'favorites_track int, listens_album int, favorites_album int,' +
    'PRIMARY KEY(year_released, listens_album, track_id))' +
    'WITH CLUSTERING ORDER BY (listens_album DESC);')
```

نکته مهمی که در این سوال وجود دارد نیاز به مرتب بودن داده‌ها بر اساس تعداد شنیده‌شدن آلبوم است. راهکار کاساندرا برای این مورد اجرای Clustering order by در هنگام تعریف جدول است که در کد بالا دیده‌می‌شود. این مرتب سازی تنها بر روی clustering key ها امکان‌پذیر است.

زمان وارد کردن داده‌ها، اجرای کوئری‌ها و نتایج به صورت زیر است.

```
amir@Amir-System:~/Desktop/Q3(cassandra)$ python Q10.py
100% | 106574/106574 [00:48<00:00, 2182.23it/s]
Data adding time: 48.83942937850952 s
Query time: 0.04624438285827637 s
Results:
Artist: Kai Engel, Album Listened: 463630, Year: 2015
Artist: Scott Holmes, Album Listened: 468967, Year: 2016
Artist: Lobo Loco, Album Listened: 88340, Year: 2017
```

برای کوئری موردنظر کافی است سال مورد نظر در یکی از سه سال گفته‌شده باشد و سپس، بر مبنای سال عرضه group by کنیم و سپس، از هر partition یک مورد را انتخاب کنیم. در اینجا به همین دلیل سال عرضه به عنوان partition key در نظر گرفته شده است که از هر سال یک نماینده انتخاب شود. برای خروجی نام artist و تعداد شنیده‌شدن آلبوم و سال این موفقیت را خروجی می‌دهیم. فقط این خروجی duplicate زیادی دارد. برای حذف این موارد نمی‌توان از

استفاده کرد چرا که این عملگر فقط بر روی **partition key** قابل انجام است. به همین، این منطق در کد پایتون با استفاده از یک مجموعه انجام شده است تا خروجی مفیدتر باشد.

```
= session.execute('SELECT artist, listens_album, year_released FROM Q10 WHERE year_released in (2015, 2016, 2017) + ' + 'GROUP BY year_released PER PARTITION LIMIT 1')
```

همچنین، برای اینکه مجموع شنیده شدن آلبوم ها در یک سال را داشته باشیم باید مانند سوال ۹ جدولی موازی این جدول داشته باشیم تا تعداد شنیده شدن آلبوم های یک هنرمند در یک سال را داشته باشد. این کار به این دلیل است که در کاساندرا **subquery** نداریم.

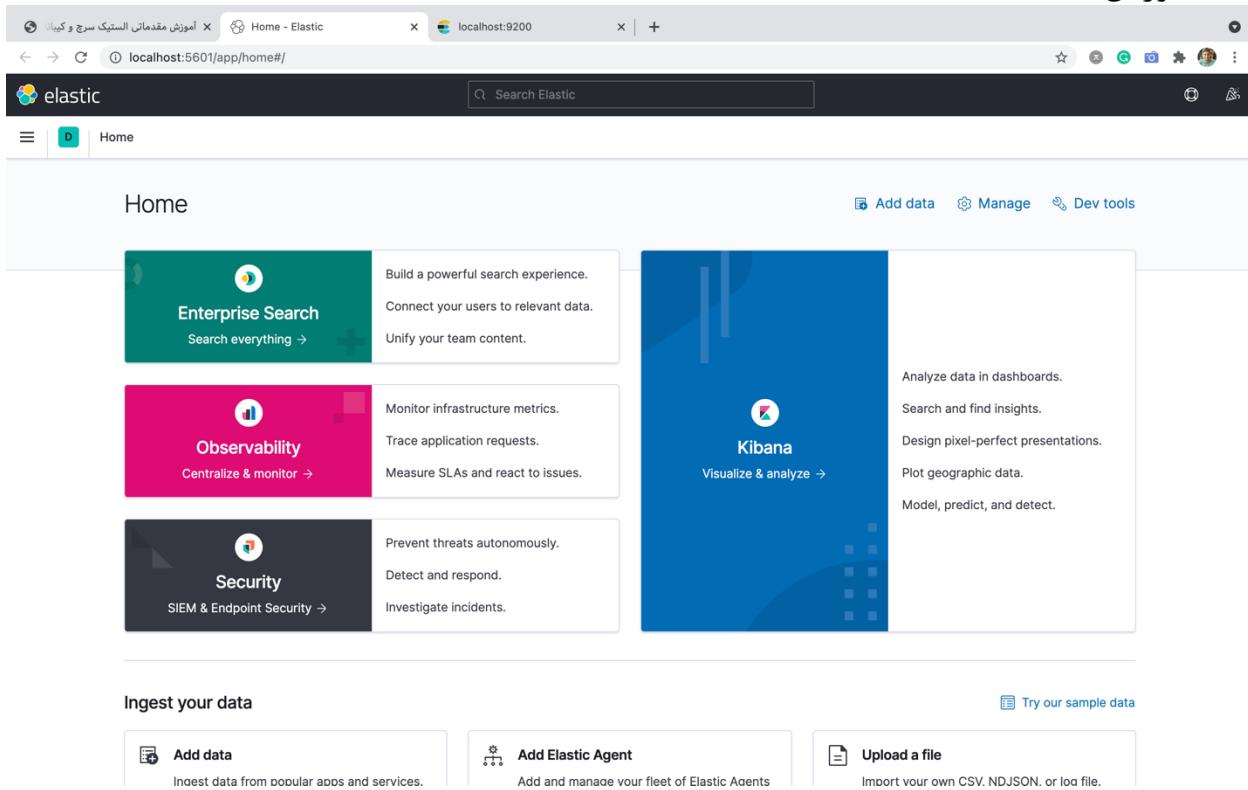
• بخش چهارم - کار با الستیک سرچ:

ابتدا آموزش سایت به صورت کامل انجام شد که در زیر تک به تک به مراحل آن اشاره می شود.
+ بالا آوردن elastic search



```
{
  "name": "Amirs-MacBook-Pro.local",
  "cluster_name": "elasticsearch",
  "cluster_uuid": "rkvkOoIisSkyUAl_T8JgjEQ",
  "version": {
    "number": "7.13.0",
    "build_flavor": "default",
    "build_type": "tar",
    "build_hash": "5ca8591cfcfdb1260ce95b08a8e023559635c6f3",
    "build_date": "2021-05-19T22:22:26.081971330Z",
    "build_snapshot": false,
    "lucene_version": "8.8.2",
    "minimum_wire_compatibility_version": "6.8.0",
    "minimum_index_compatibility_version": "6.0.0-beta1"
  },
  "tagline": "You Know, for Search"
}
```

+ بالا آوردن kibana

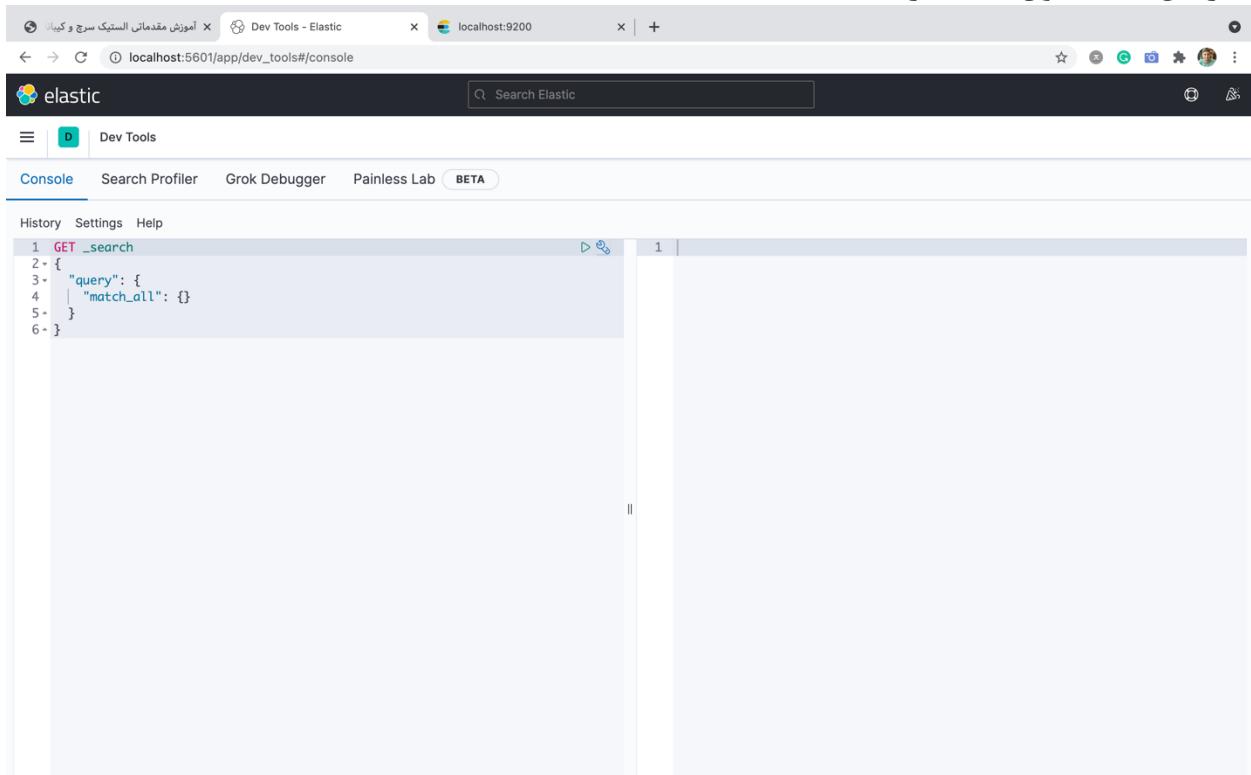


The screenshot shows the Elastic homepage with the following sections:

- Enterprise Search**: Search everything →
- Observability**: Centralize & monitor →
- Security**: SIEM & Endpoint Security →
- Kibana**: Visualize & analyze →

Below these sections, there are buttons for "Ingest your data" (Add data, Add Elastic Agent, Upload a file) and a "Try our sample data" button.

+ رفتن به کنسول کیبانا از .dev tools

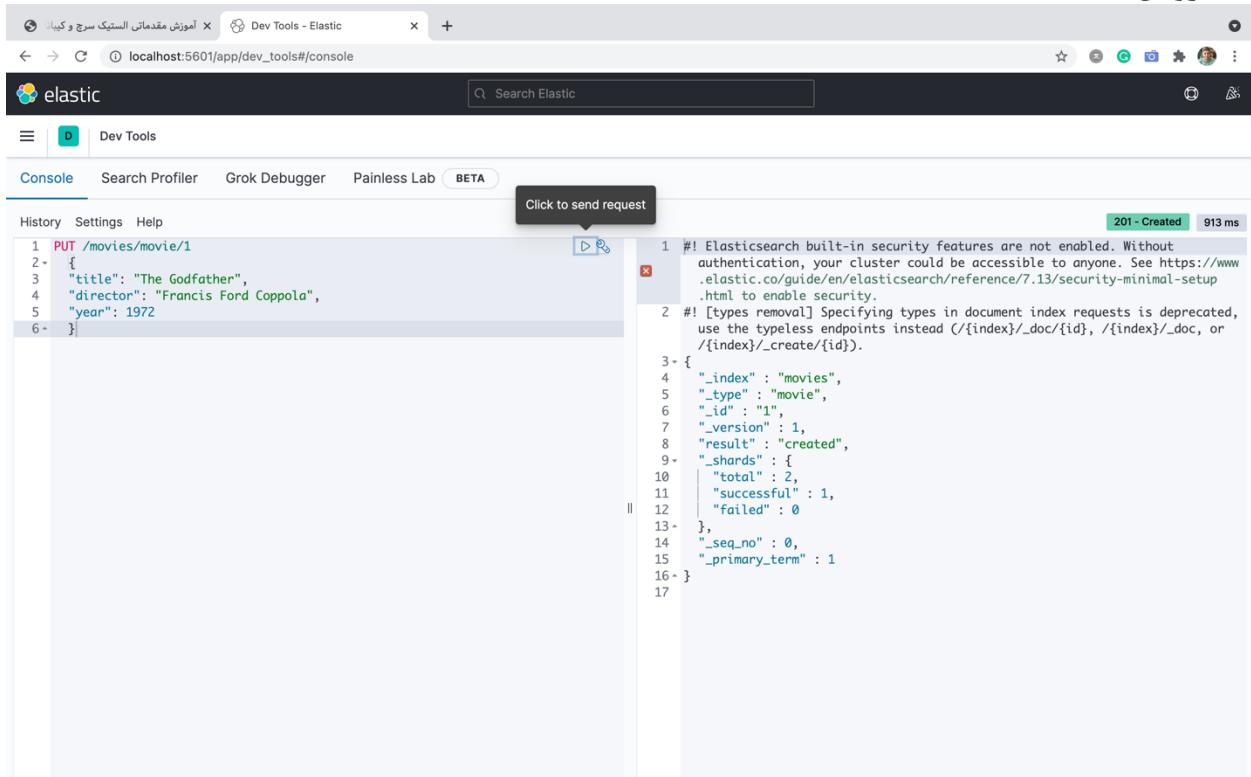


A screenshot of the Elasticsearch Dev Tools Console. The URL is `localhost:5601/app/dev_tools#/console`. The interface shows a code editor with a query:

```
1 GET _search
2 {
3   "query": {
4     | "match_all": {}
5   }
6 }
```

 and a results panel on the right.

+ افزودن داده:



A screenshot of the Elasticsearch Dev Tools Console. The URL is `localhost:5601/app/dev_tools#/console`. The interface shows a code editor with a PUT request:

```
1 PUT /movies/movie/1
2 {
3   "title": "The Godfather",
4   "director": "Francis Ford Coppola",
5   "year": 1972
6 }
```

 and a results panel on the right. A tooltip says "Click to send request". The results show a successful response:

```
201 - Created 913 ms
```

```
1 #! Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html to enable security.
2 #! [types removal] Specifying types in document index requests is deprecated, use the typeless endpoints instead /{index}/_doc/{id}, /{index}/_doc, or /{index}/_create/{id}.
3 +
4   "_index" : "movies",
5   "_type" : "movie",
6   "_id" : "1",
7   "_version" : 1,
8   "result" : "created",
9   "_shards" : {
10     "total" : 2,
11     "successful" : 1,
12     "failed" : 0
13   },
14   "_seq_no" : 0,
15   "_primary_term" : 1
16 }
```

+ افزودن داده با شناسه خودکار:

```

POST /movies/_doc
{
  "title": "The Godfather",
  "director": "Francis Ford Coppola",
  "year": 1972
}

```

Click to send request

```

#! Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html to enable security.

{
  "_index": "movies",
  "_type": "_doc",
  "_id": "0X-upHkBmHTifiL8wd76",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 2,
  "_primary_term": 1
}

```

201 - Created 55 ms

+ افرودن داده‌ی بیشتر و گرفتن یک داده:

```

PUT movies/_doc/3
{
  "director": "David Lean",
  "year": 1962,
  "genres": ["Adventure", "Biography", "Drama"]
}

PUT movies/_doc/4
{
  "title": "To Kill a Mockingbird",
  "director": "Robert Mulligan",
  "year": 1962,
  "genres": ["Crime", "Drama", "Mystery"]
}

PUT movies/_doc/5
{
  "title": "Apocalypse Now",
  "director": "Francis Ford Coppola",
  "year": 1979,
  "genres": ["Drama", "War"]
}

PUT movies/_doc/6
{
  "title": "Kill Bill: Vol. 1",
  "director": "Quentin Tarantino",
  "year": 2003,
  "genres": ["Action", "Crime", "Thriller"]
}

PUT movies/_doc/7
{
  "title": "The Assassination of Jesse James by the Coward Robert Ford",
  "director": "Andrew Dominik",
  "year": 2007,
  "genres": ["Biography", "Crime", "Drama"]
}

GET /movies/_doc/1

```

```

#! Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html to enable security.

{
  "_index": "movies",
  "_type": "_doc",
  "_id": "1",
  "_version": 2,
  "_seq_no": 1,
  "_primary_term": 1,
  "found": true,
  "_source": {
    "title": "The Godfather",
    "director": "Francis Ford Coppola",
    "year": 1972
  }
}

```

200 - OK 20 ms

داده‌ها به نوعی اضافه شده‌اند که هر جستجو حداقل دو خروجی داشته باشد.

+ جستجو بر اساس query string

+ جستجو در یک فیلد خاص:

```

45 PUT /movies/_doc/8
46 {
47   "title": "Kill John",
48   "director": "BB Ford",
49   "year": 2007,
50   "genres": ["Biography", "Crime", "Drama"]
51 }
52 POST /movies/_search
53 {
54   "query": {
55     "query_string": {
56       "query": "kill"
57     }
58   }
59 }
60 GET /movies/_search
61 {
62   "query": {
63     "query_string": {
64       "query": "ford",
65       "fields": ["title"]
66     }
67   }
68 }
69
70
71
72
73
74
75
76

```

200 - OK 694 ms

```

1 #! Elasticsearch built-in security features are not enabled. Without
2 authentication, your cluster could be accessible to anyone. See https://www.
3 .elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup
4 .html to enable security.
5
6 {
7   "took" : 680,
8   "timed_out" : false,
9   "_shards" : {
10    "total" : 1,
11    "successful" : 1,
12    "skipped" : 0,
13    "failed" : 0
14  },
15  "hits" : {
16    "total" : {
17      "value" : 4,
18      "relation" : "eq"
19    },
20    "max_score" : 0.9725769,
21    "hits" : [
22      {
23        "_index" : "movies",
24        "_type" : "movie",
25        "_id" : "8",
26        "_score" : 0.9725769,
27        "_source" : {
28          "title" : "Kill John",
29          "director" : "BB Ford",
30          "year" : 2007,
31          "genres" : [
32            "Biography",
33            "Crime",
34            "Drama"
35          ]
36        }
37      },
38      {
39        "_index" : "movies",
40        "_type" : "movie",
41        "_id" : "6",
42        "_score" : 0.79604506,
43        "_source" : {
44          "title" : "The Assassination of Jesse James by the Coward Robert
45          Ford",
46          "director" : "Andrew Dominik",
47          "year" : 2007,
48          "genres" : [
49            ...
50          ]
51        }
52      }
53    ]
54  }
55 }

```

+ جستجو با چسباندن شرط (and or يا)

```

48   "director": "BB Ford",
49   "year": 2007,
50   "genres": ["Biography", "Crime", "Drama"]
51 }
52 POST /movies/_search
53 {
54   "query": {
55     "query_string": {
56       "query": "kill"
57     }
58   }
59 }
60 GET /movies/_search
61 {
62   "query": {
63     "query_string": {
64       "query": "ford",
65       "fields": ["title"]
66     }
67   }
68 }
69 GET /movies/_search
70 {
71   "query": {
72     "query_string": {"query" : "title:ford"}
73   }
74 }
75
76
77
78
79
80

```

200 - OK 27 ms

```

13   "value" : 2,
14   "relation" : "eq"
15 },
16   "max_score" : 1.4809579,
17   "hits" : [
18     {
19       "_index" : "movies",
20       "_type" : "movie",
21       "_id" : "7",
22       "_score" : 1.4809579,
23       "_source" : {
24         "title" : "BB kill Ford",
25         "director" : "AA",
26         "year" : 2007,
27         "genres" : [
28           "Biography",
29           "Crime",
30           "Drama"
31         ]
32       }
33     },
34     {
35       "_index" : "movies",
36       "_type" : "movie",
37       "_id" : "6",
38       "_score" : 0.79604506,
39       "_source" : {
40         "title" : "The Assassination of Jesse James by the Coward Robert
41         Ford",
42         "director" : "Andrew Dominik",
43         "year" : 2007,
44         "genres" : [
45           ...
46         ]
47       }
48     }
49   ]
50 }

```

Dev Tools - Elastic +

localhost:5601/app/dev_tools#/console

elastic Search Elastic

Dev Tools

Console Search Profiler Grok Debugger Painless Lab (BETA)

History Settings Help

```

57-   }
58- }
59- }
60 GET /movies/_search
61 {
62 "query": {
63   "query_string": {
64     "query": "ford",
65     "fields": ["title"]
66   }
67 }
68 }
69 GET /movies/_search
70 {
71 "query": {
72   "query_string" : {"query" : "title:ford"}
73 }
74 }
75 GET /movies/_search
76 {
77 "query": {
78   "query_string" : {"query" : "title:ford OR kill"}
79 }
80 }
81
82
83
84
85
86
87
88
89

```

200 - OK 28 ms

```

19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

```

```

"_index": "movies",
"_type": "movie",
"_id": "7",
"_score": 2.333992,
"_source": {
  "title": "BB kill Ford",
  "director": "AA",
  "year": 2007,
  "genres": [
    "Biography",
    "Crime",
    "Drama"
  ]
},
{
  "_index": "movies",
  "_type": "movie",
  "_id": "8",
  "_score": 0.9725769,
  "_source": {
    "title": "Kill John",
    "director": "BB Ford",
    "year": 2007,
    "genres": [
      "Biography",
      "Crime",
      "Drama"
    ]
  }
},
{
  "_index": "movies"
}

```

Dev Tools - Elastic +

localhost:5601/app/dev_tools#/console

elastic Search Elastic

Dev Tools

Console Search Profiler Grok Debugger Painless Lab (BETA)

History Settings Help

```

1/ "query": {
2   "query_string" : {"query" : "title:ford"}
3 }
4 }
5 GET /movies/_search
6 {
7 "query": {
8   "query_string" : {"query" : "title:ford OR kill"}
9 }
10 }
11 PUT movies/_doc/9
12 {
13   "title": "Kill Mamad: Vol. 1",
14   "director": "Asghar Tarantino",
15   "year": 2003,
16   "genres": ["Action", "Crime", "Thriller"]
17 }
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

```

200 - OK 943 ms

```

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

```

```

  "max_score": 2.2018442,
  "hits": [
    {
      "_index": "movies",
      "_type": "movie",
      "_id": "5",
      "_score": 2.2018442,
      "_source": {
        "title": "Kill Bill: Vol. 1",
        "director": "Quentin Tarantino",
        "year": 2003,
        "genres": [
          "Action",
          "Crime",
          "Thriller"
        ]
      }
    },
    {
      "_index": "movies",
      "_type": "movie",
      "_id": "9",
      "_score": 2.2018442,
      "_source": {
        "title": "Kill Mamad: Vol. 1",
        "director": "Asghar Tarantino",
        "year": 2003,
        "genres": [
          "Action",
          "Crime",
          "Thriller"
        ]
      }
    }
  ]
}

```

+ جستجوی فیلم‌های سال ۱۹۶۲ با ژانر درام:

Dev Tools - Elastic + localhost:5601/app/dev_tools#/console

elastic Search Elastic

☰ D Dev Tools

Console Search Profiler Grok Debugger Painless Lab (BETA)

History Settings Help

```

78 "query_string": {"query": "title:ford UK KILL"}
79 }
80 }
81 PUT movies/_doc/9
82 {
83   "title": "Kill Mamad: Vol. 1",
84   "director": "Asghar Tarantino",
85   "year": 2003,
86   "genres": ["Action", "Crime", "Thriller"]
87 }
88 GET /movies/_search
89 {
90   "query": {
91     "query_string": {"query": "(title:kill) AND (director:Tarantino)"}
92   }
93 }
94 GET /movies/_search
95 {
96   "query": {
97     "query_string": {"query": "(genres:Drama) AND (year:1962)"}
98   }
99 }
100
101
102
103
104
105
106
107
108
109
110

```

200 - OK | 15 ms

```

16   "max_score" : 1.3197354,
17   "hits" : [
18     {
19       "_index": "movies",
20       "_type": "movie",
21       "_id": "2",
22       "_score": 1.3197354,
23       "_source": {
24         "title": "Lawrence of Arabia",
25         "director": "David Lean",
26         "year": 1962,
27         "genres": [
28           "Adventure",
29           "Biography",
30           "Drama"
31         ]
32       },
33     },
34     {
35       "_index": "movies",
36       "_type": "movie",
37       "_id": "3",
38       "_score": 1.3197354,
39       "_source": {
40         "title": "To Kill a Mockingbird",
41         "director": "Robert Mulligan",
42         "year": 1962,
43         "genres": [
44           "Crime",
45           "Drama",
46           "Mystery"
47         ]
48       }
49     }
50   ]
51 }
```

+ جستجوی range روی سال:

Dev Tools - Elastic + localhost:5601/app/dev_tools#/console

elastic Search Elastic

☰ D Dev Tools

Console Search Profiler Grok Debugger Painless Lab (BETA)

History Settings Help

```

78 "query_string": {"query": "title:ford UK KILL"}
79 }
80 }
81 PUT movies/_doc/9
82 {
83   "title": "Kill Mamad: Vol. 1",
84   "director": "Asghar Tarantino",
85   "year": 2003,
86   "genres": ["Action", "Crime", "Thriller"]
87 }
88 GET /movies/_search
89 {
90   "query": {
91     "query_string": {"query": "(title:kill) AND (director:Tarantino)"}
92   }
93 }
94 GET /movies/_search
95 {
96   "query": {
97     "query_string": {"query": "(genres:Drama) AND (year:[* TO 2005])"}}
98   }
99 }
100
101
102
103
104
105
106
107
108
109
110

```

200 - OK | 31 ms

```

12   "total": 1
13   "value": 3,
14   "relation": "eq"
15 },
16   "max_score": 1.3717015,
17   "hits": [
18     {
19       "_index": "movies",
20       "_type": "movie",
21       "_id": "4",
22       "_score": 1.3717015,
23       "_source": {
24         "title": "Apocalypse Now",
25         "director": "Francis Ford Coppola",
26         "year": 1979,
27         "genres": [
28           "Drama",
29           "War"
30         ]
31       }
32     },
33     {
34       "_index": "movies",
35       "_type": "movie",
36       "_id": "2",
37       "_score": 1.3197354,
38       "_source": {
39         "title": "Lawrence of Arabia",
40         "director": "David Lean",
41         "year": 1962,
42         "genres": [
43           "Adventure",
44           "Biography",
45           "Drama"
46         ]
47       }
48     }
49   ]
50 }
```

+ اهمیت دادن به یک فیلد در جستجو:

Dev Tools - Elastic x +

localhost:5601/app/dev_tools#/console

elastic

Search Elastic

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

200 - OK | 23 ms

```
112 GET /movies/_search
113 {
114 "query": {
115   "query_string": {"query": "(genres:Drama) OR (title:kill^2)"}
116 }
117 }
```

```
12 "total" : {
13   "value" : 8,
14   "relation" : "eq"
15 },
16 "max_score" : 2.0140953,
17 "hits" : [
18 {
19   "_index" : "movies",
20   "_type" : "movie",
21   "_id" : "8",
22   "_score" : 2.0140953,
23   "_source" : {
24     "title" : "Kill John",
25     "director" : "BB Ford",
26     "year" : 2007,
27     "genres" : [
28       "Biography",
29       "Crime",
30       "Drama"
31     ]
32   }
33 },
34 {
35   "_index" : "movies",
36   "_type" : "movie",
37   "_id" : "7",
38   "_score" : 1.8074659,
39   "_source" : {
40     "title" : "BB kill Ford",
41     "director" : "AA",
42     "year" : 2007,
43     "genres" : [
44       "Biaranyh".
45     ]
46   }
47 }
```

همانطور که می‌بینید امتیازها متفاوت شده‌اند.
+ جستجو بر اساس درخت جستجو:

Dev Tools - Elastic x +

localhost:5601/app/dev_tools#/console

elastic

Search Elastic

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

200 - OK 25 ms

```
125 }
126 POST movies/_search
127 {
128   "query": {
129     "match": {
130       "title": {
131         "query": "Kill Bill",
132         "operator": "or"
133       }
134     }
135   }
136 }
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157 }
```

15 },
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47 },

},
"max_score" : 2.568813,
"hits" : [
{
 "_index" : "movies",
 "_type" : "movie",
 "_id" : "8",
 "_score" : 2.568813,
 "_source" : {
 "title" : "Kill Bill: Vol. 1",
 "director" : "Quentin Tarantino",
 "year" : 2003,
 "genres" : [
 "Action",
 "Crime",
 "Thriller"
]
 }
},
{
 "_index" : "movies",
 "_type" : "movie",
 "_id" : "8",
 "_score" : 0.8471799,
 "_source" : {
 "title" : "Kill John",
 "director" : "BB Ford",
 "year" : 2007,
 "genres" : [
 "Biography",
 "Crime",
 "Drama"
]
 }
}]

+ وزن دهی در درخت جستجو:

The screenshot shows the Elasticsearch Dev Tools interface with the 'Console' tab selected. A search query is entered in the text area:

```
136 - }  
137 POST /movies/_search  
138 {  
139   "query": {  
140     "multi_match": {  
141       "query": "ford",  
142       "fields": [  
143         "title^3",  
144         "director"  
145       ]  
146     }  
147   }  
148 }
```

The response on the right side shows two hits from the 'movies' index:

```
17 - "hits": [  
18 -   {  
19 -     "_index": "movies",  
20 -     "_type": "movie",  
21 -     "_id": "7",  
22 -     "_score": 4.7700443,  
23 -     "_source": {  
24 -       "title": "BB kill Ford",  
25 -       "director": "AA",  
26 -       "year": 2007,  
27 -       "genres": [  
28 -         "Biography",  
29 -         "Crime",  
30 -         "Drama"  
31 -       ]  
32 -     },  
33 -   {  
34 -     "_index": "movies",  
35 -     "_type": "movie",  
36 -     "_id": "6",  
37 -     "_score": 2.5733132,  
38 -     "_source": {  
39 -       "title": "The Assassination of Jesse James by the Coward Robert  
40 -         Ford",  
41 -         "director": "Andrew Dominik",  
42 -         "year": 2007,  
43 -         "genres": [  
44 -           "Biography",  
45 -           "Crime",  
46 -           "Drama"  
47 -         ]  
48 -     },  
49 -   }  
50 - ]
```

+ جستجوی bool و انواع شرطی .must و .should و .must_not

The screenshot shows the Elasticsearch Dev Tools interface with the 'Console' tab selected. A search query is entered in the text area:

```
172 - {  
173   "query": {  
174     "bool": {  
175       "must": {  
176         "bool": {  
177           "should": [  
178             {  
179               "match": {  
180                 "title": "ford"  
181               }  
182             },  
183             {  
184               "match": {  
185                 "title": "kill"  
186               }  
187             }  
188           ]  
189         }  
190       },  
191       "must_not": {  
192         "match": {  
193           "genres": "Mystery"  
194         }  
195       }  
196     }  
197   }  
198 }
```

The response on the right side shows the hits and their details:

```
11 - "hits": {  
12 -   "total": {  
13 -     "value": 5,  
14 -     "relation": "eq"  
15 -   },  
16 -   "max_score": 2.33388,  
17 -   "hits": [  
18 -     {  
19 -       "_index": "movies",  
20 -       "_type": "movie",  
21 -       "_id": "7",  
22 -       "_score": 2.33388,  
23 -       "_source": {  
24 -         "title": "BB kill Ford",  
25 -         "director": "AA",  
26 -         "year": 2007,  
27 -         "genres": [  
28 -           "Biography",  
29 -           "Crime",  
30 -           "Drama"  
31 -         ]  
32 -       },  
33 -     },  
34 -     {  
35 -       "_index": "movies",  
36 -       "_type": "movie",  
37 -       "_id": "6",  
38 -       "_score": 0.8577709,  
39 -       "_source": {  
40 -         "title": "The Assassination of Jesse James by the Coward Robert  
41 -           Ford",  
42 -         "director": "Andrew Dominik",  
43 -         "year": 2007,  
44 -       },  
45 -     },  
46 -     {  
47 -       "_index": "movies",  
48 -       "_type": "movie",  
49 -       "_id": "5",  
50 -       "_score": 0.8577709,  
51 -       "_source": {  
52 -         "title": "The Assassination of Jesse James by the Coward Robert  
53 -           Ford",  
54 -         "director": "Andrew Dominik",  
55 -         "year": 2007,  
56 -       },  
57 -     },  
58 -     {  
59 -       "_index": "movies",  
60 -       "_type": "movie",  
61 -       "_id": "4",  
62 -       "_score": 0.8577709,  
63 -       "_source": {  
64 -         "title": "The Assassination of Jesse James by the Coward Robert  
65 -           Ford",  
66 -         "director": "Andrew Dominik",  
67 -         "year": 2007,  
68 -       },  
69 -     },  
70 -   }  
71 - }
```

+ جستجوی با عبارت Term

```

195 -     }
196 -   }
197 - }
198 - }
199 - GET /movies/_search
200 - {
201 -   "query": {
202 -     "term": {
203 -       "year": {
204 -         "value": "2003",
205 -         "boost": 1.0
206 -       }
207 -     }
208 -   }
209 - }

210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227

```

Response:

```

17 - "hits" : [
18 -   {
19 -     "_index" : "movies",
20 -     "_type" : "movie",
21 -     "_id" : "5",
22 -     "_score" : 1.0,
23 -     "_source" : {
24 -       "title" : "Kill Bill: Vol. 1",
25 -       "director" : "Quentin Tarantino",
26 -       "year" : 2003,
27 -       "genres" : [
28 -         "Action",
29 -         "Crime",
30 -         "Thriller"
31 -       ]
32 -     }
33 -   },
34 -   {
35 -     "_index" : "movies",
36 -     "_type" : "movie",
37 -     "_id" : "9",
38 -     "_score" : 1.0,
39 -     "_source" : {
40 -       "title" : "Kill Mamad: Vol. 1",
41 -       "director" : "Asghar Tarantino",
42 -       "year" : 2003,
43 -       "genres" : [
44 -         "Action",
45 -         "Crime",
46 -         "Thriller"
47 -       ]
48 -     }
49 -   }
50 - ]

```

+ فیلتر کردن کوئری:

```

211 - {
212 -   "query": {
213 -     "bool": {
214 -       "must": [
215 -         {
216 -           "match": {
217 -             "title": "kill"
218 -           }
219 -         },
220 -         {
221 -           "filter": {
222 -             "bool": {
223 -               "must": [
224 -                 {
225 -                   "range": {
226 -                     "year": {
227 -                       "gte": 1960
228 -                     }
229 -                   },
230 -                   {
231 -                     "term": {
232 -                       "genres": {
233 -                         "value": "drama"
234 -                       }
235 -                     }
236 -                   }
237 -                 ]
238 -               }
239 -             }
240 -           }
241 -         }
242 -       ]
243 -     }
244 -   }
245 - }

246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
399

```

Response:

```

11 - "hits" : {
12 -   "total" : {
13 -     "value" : 3,
14 -     "relation" : "eq"
15 -   },
16 -   "max_score" : 0.8471799,
17 -   "hits" : [
18 -     {
19 -       "_index" : "movies",
20 -       "_type" : "movie",
21 -       "_id" : "8",
22 -       "_score" : 0.8471799,
23 -       "_source" : {
24 -         "title" : "Kill John",
25 -         "director" : "BB Ford",
26 -         "year" : 2007,
27 -         "genres" : [
28 -           "Biography",
29 -           "Crime",
30 -           "Drama"
31 -         ]
32 -       }
33 -     },
34 -     {
35 -       "_index" : "movies",
36 -       "_type" : "movie",
37 -       "_id" : "7",
38 -       "_score" : 0.74386525,
39 -       "_source" : {
40 -         "title" : "BB kill Ford",
41 -         "director" : "AA",
42 -         "year" : 2007,
43 -         "genres" : [
44 -           "Action",
45 -           "Crime",
46 -           "Thriller"
47 -         ]
48 -       }
49 -     }
50 -   ]
51 - }
52 - }
53 - }
54 - }
55 - }
56 - }
57 - }
58 - }
59 - }
60 - }
61 - }
62 - }
63 - }
64 - }
65 - }
66 - }
67 - }
68 - }
69 - }
70 - }
71 - }
72 - }
73 - }
74 - }
75 - }
76 - }
77 - }
78 - }
79 - }
80 - }
81 - }
82 - }
83 - }
84 - }
85 - }
86 - }
87 - }
88 - }
89 - }
90 - }
91 - }
92 - }
93 - }
94 - }
95 - }
96 - }
97 - }
98 - }
99 - }
100 - }
101 - }
102 - }
103 - }
104 - }
105 - }
106 - }
107 - }
108 - }
109 - }
110 - }
111 - }
112 - }
113 - }
114 - }
115 - }
116 - }
117 - }
118 - }
119 - }
120 - }
121 - }
122 - }
123 - }
124 - }
125 - }
126 - }
127 - }
128 - }
129 - }
130 - }
131 - }
132 - }
133 - }
134 - }
135 - }
136 - }
137 - }
138 - }
139 - }
140 - }
141 - }
142 - }
143 - }
144 - }
145 - }
146 - }
147 - }
148 - }
149 - }
150 - }
151 - }
152 - }
153 - }
154 - }
155 - }
156 - }
157 - }
158 - }
159 - }
160 - }
161 - }
162 - }
163 - }
164 - }
165 - }
166 - }
167 - }
168 - }
169 - }
170 - }
171 - }
172 - }
173 - }
174 - }
175 - }
176 - }
177 - }
178 - }
179 - }
180 - }
181 - }
182 - }
183 - }
184 - }
185 - }
186 - }
187 - }
188 - }
189 - }
190 - }
191 - }
192 - }
193 - }
194 - }
195 - }
196 - }
197 - }
198 - }
199 - }
200 - }
201 - }
202 - }
203 - }
204 - }
205 - }
206 - }
207 - }
208 - }
209 - }
210 - }
211 - }
212 - }
213 - }
214 - }
215 - }
216 - }
217 - }
218 - }
219 - }
220 - }
221 - }
222 - }
223 - }
224 - }
225 - }
226 - }
227 - }
228 - }
229 - }
230 - }
231 - }
232 - }
233 - }
234 - }
235 - }
236 - }
237 - }
238 - }
239 - }
240 - }
241 - }
242 - }
243 - }
244 - }
245 - }
246 - }
247 - }
248 - }
249 - }
250 - }
251 - }
252 - }
253 - }
254 - }
255 - }
256 - }
257 - }
258 - }
259 - }
260 - }
261 - }
262 - }
263 - }
264 - }
265 - }
266 - }
267 - }
268 - }
269 - }
270 - }
271 - }
272 - }
273 - }
274 - }
275 - }
276 - }
277 - }
278 - }
279 - }
280 - }
281 - }
282 - }
283 - }
284 - }
285 - }
286 - }
287 - }
288 - }
289 - }
290 - }
291 - }
292 - }
293 - }
294 - }
295 - }
296 - }
297 - }
298 - }
299 - }
299 - }

```

Dev Tools - Elastic X + localhost:5601/app/dev_tools#/console

elastic Search Elastic

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

```
200 - OK | 21 ms
```

```
242 ~     }
243 ~   }
244 GET /movies/_search
245 {
246   "query": {
247     "bool": {
248       "must": [
249         {
250           "match": {
251             "title": "kill"
252           }
253         },
254         {
255           "range": {
256             "year": {
257               "gte": 1960
258             }
259           }
260         },
261         {
262           "term": {
263             "genres": {
264               "value": "drama"
265             }
266           }
267         }
268       ]
269     }
270   }
271 }
```

```
13
14   "value" : 3,
15   "relation" : "eq"
16 },
17   "max_score" : 2.1669154,
18   "hits" : [
19     {
20       "_index" : "movies",
21       "_type" : "movie",
22       "_id" : "8",
23       "_score" : 2.1669154,
24       "_source" : {
25         "title" : "Kill John",
26         "director" : "BB Ford",
27         "year" : 2007,
28         "genres" : [
29           "Biography",
30           "Crime",
31           "Drama"
32         ]
33       }
34     },
35     {
36       "_index" : "movies",
37       "_type" : "movie",
38       "_id" : "7",
39       "_score" : 2.0636008,
40       "_source" : {
41         "title" : "BB kill Ford",
42         "director" : "AA",
43         "year" : 2007,
44         "genres" : [
45           "Biography",
46           "Crime",
47           "Drama"
48         ]
49       }
50     }
51   ]
52 }
```

+ جستجو تقریبی(Fuzzy) به دلیل غلط املایی و ...:

The screenshot shows the Elasticsearch Dev Tools interface. The top navigation bar includes tabs for 'Console' (selected), 'Search Profiler', 'Grok Debugger', and 'Painless Lab'. A search bar at the top right contains the placeholder 'Search Elastic'. The main area displays a code editor with a history of requests and a results panel.

History

```
315-    }
316-    ]
317-  ]
318- }
319- }
320- }
321 PUT movies/_doc/10
322 {
323   "title": "Kill godfatter: Vol. 1",
324   "director": "Asghar Tarantino",
325   "year": 2003,
326   "genres": ["Action", "Crime", "Thriller"]
327 }
328 GET /movies/_search
329 {
330   "query": {
331     "match": {}
332     "title": {
333       "query": "godfater",
334       "fuzziness": "AUTO"
335     }
336   }
337 }
338 }
```

Results

```
18-
19  {
20    "_index": "movies",
21    "_type": "movie",
22    "_id": "1",
23    "_score": 1.6822205,
24    "_source": {
25      "title": "The Godfather",
26      "director": "Francis Ford Coppola",
27      "year": 1972
28    }
29  },
30  {
31    "_index": "movies",
32    "_type": "movie",
33    "_id": "QX-upHk8mHtifiL8wd76",
34    "_score": 1.6822205,
35    "_source": {
36      "title": "The Godfather",
37      "director": "Francis Ford Coppola",
38      "year": 1972
39    }
40  },
41  {
42    "_index": "movies",
43    "_type": "movie",
44    "_id": "10",
45    "_score": 1.3185964,
46    "_source": {
47      "title": "Kill godfatter: Vol. 1",
48      "director": "Asghar Tarantino",
49      "year": 2003,
50      "genres": [
51        "Action"
52      ]
53    }
54  }
```

Details: 200 OK | 764 ms

+ مج کردن کامل عبارت:

Dev Tools - Elastic x +

localhost:5601/app/dev_tools#/console

elastic

Search Elastic

Dev Tools

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

200 - OK | 22 ms

```
334 |     "fuzziness": "AUTO"
335 |   }
336 | }
337 |
338 }
339 PUT movies/_doc/11
340 {
341   "title": "Kill Bill Mill",
342   "director": "Asghar Tarantino",
343   "year": 2003,
344   "genres": ["Action", "Crime", "Thriller"]
345 }
346 GET /movies/_search
347 {
348   "query": {
349     "match_phrase": {
350       "title": {
351         "query": "Kill Bill",
352         "slop": 2
353       }
354     }
355   }
356 }
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2597
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2678
2679
2680

```

+ عدم یافتن نتیجه به دلیل tokenize کردن:

Dev Tools - Elastic X +

localhost:5601/app/dev_tools#/console

elastic Dev Tools

Search Elastic

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help 200 - OK 27 ms

```
357 GET /movies/_search
358 {
359   "query": {
360     "term": {
361       "director": {
362         "value": "Francis Ford Coppola"
363       }
364     }
365   }
366 }
```

1 #! Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html to enable security.

```
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    "total" : {
13      "value" : 0,
14      "relation" : "eq"
15    },
16    "max_score" : null,
17    "hits" : [ ]
18  }
19 }
```

+ راه حل اول (ایجاد فیلدهای چند مقداری):

البته نیاز به تعریف از ابتدا جدول دارد.

Dev Tools - Elastic + localhost:5601/app/dev_tools#/console

elastic Search Elastic

☰ D Dev Tools

Console Search Profiler Grok Debugger Painless Lab (BETA)

History Settings Help

```
367 PUT /movies/_mapping
368 {
369   "movie": {
370     "properties": {
371       "director": {
372         "type": "text",
373         "fields": {
374           "original": {
375             "type": "keyword"
376           }
377         }
378       }
379     }
380   }
381 }
```

1 `#! Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html to enable security.`

2+ {
3 "acknowledged" : true
4+ }

200 - OK 170 ms

Dev Tools - Elastic + localhost:5601/app/dev_tools#/console

elastic Search Elastic

☰ D Dev Tools

Console Search Profiler Grok Debugger Painless Lab (BETA)

History Settings Help

```
356 }
357 GET /movies/_search
358 {
359   "query": {
360     "term": {
361       "director": {
362         "value": "Francis Ford Coppola"
363       }
364     }
365   }
366 }
367 PUT /movies/_mapping
368 {
369   "movie": {
370     "properties": {
371       "director": {
372         "type": "text",
373         "fields": {
374           "original": {
375             "type": "keyword"
376           }
377         }
378       }
379     }
380   }
381 }
```

16 "max_score" : 1.3121864,
17+ "hits" : [
18+ {
19 "_index" : "movies",
20 "_type" : "movie",
21 "_id" : "1",
22 "_score" : 1.3121864,
23 "_source" : {
24 "title" : "The Godfather",
25 "director" : "Francis Ford Coppola",
26 "year" : 1972
27 }
28 },
29 {
30 "_index" : "movies",
31 "_type" : "movie",
32 "_id" : "Ox-upHkBmHTifil8wd76",
33 "_score" : 1.3121864,
34 "_source" : {
35 "title" : "The Godfather",
36 "director" : "Francis Ford Coppola",
37 "year" : 1972
38 }
39 },
40 {
41 "_index" : "movies",
42 "_type" : "movie",
43 "_id" : "4",
44 "_score" : 1.3121864,
45 "_source" : {
46 "title" : "Apocalypse Now",
47 "director" : "Francis Ford Coppola",
48 "year" : 1979,

200 - OK 23 ms

+ راه حل دوم(گزینه بهتر - استفاده از `:keyword`)

The screenshot shows the Elasticsearch Dev Tools interface at localhost:5601/app/dev_tools/#/console. The search bar contains "Search Elastic". The results pane displays a search response for the query "director.keyword: Francis Ford Coppola". The results are as follows:

```

max_score": 1.3121864,
"hits": [
  {
    "_index": "movies",
    "_type": "movie",
    "_id": "1",
    "_score": 1.3121864,
    "_source": {
      "title": "The Godfather",
      "director": "Francis Ford Coppola",
      "year": 1972
    }
  },
  {
    "_index": "movies",
    "_type": "movie",
    "_id": "QX-upHkBmHTifilL8wd76",
    "_score": 1.3121864,
    "_source": {
      "title": "The Godfather",
      "director": "Francis Ford Coppola",
      "year": 1972
    }
  },
  {
    "_index": "movies",
    "_type": "movie",
    "_id": "4",
    "_score": 1.3121864,
    "_source": {
      "title": "Apocalypse Now",
      "director": "Francis Ford Coppola",
      "year": 1979
    }
  }
]

```

اسکریپت‌های اجراشده در این قسمت در فایل `bigdata_ir_script.py` قرار داده شده است. سپس، اسکریپتی نوشته شد که توییت‌های بورسی را از سایت [sahamyab](#) می‌خواند و هشتگ‌ها را جدا می‌کرد و آن‌ها روی الستیک می‌ریخت. این اسکریپت ۱۰۰۰ توییت را از سایت سهام یاب در الستیک سرج ریخت. این اسکریپت در فایل `add_tweets.py` قرار گرفته است. نتایج اجرای این اسکریپت به صورت زیر است.

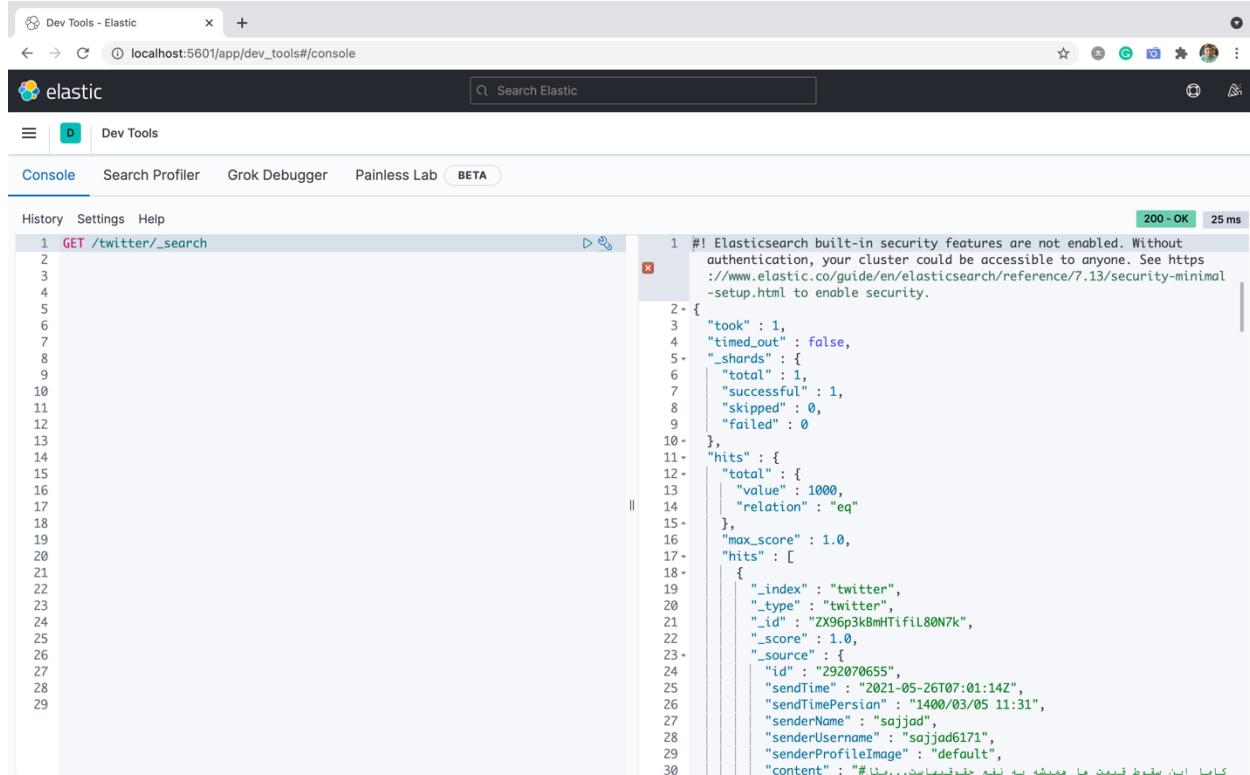
```
(base) iamiranjabar@Amirs-MacBook-Pro Q4(Elastic) % python add_tweets.py
<Elasticsearch([{'host': 'localhost', 'port': 9200}])>
/Users/iamiranjabar/anaconda3/lib/python3.8/site-packages/elasticsearch/connection/base.py:208: ElasticsearchWarning: Elasticsearch built-in security features are not enabled. Without authentication, your cluster could be accessible to anyone. See https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html to enable security.
  warnings.warn(message, category=ElasticsearchWarning)
/Users/iamiranjabar/anaconda3/lib/python3.8/site-packages/elasticsearch/connection/base.py:208: ElasticsearchWarning: [types removal] Specifying types in document index requests is deprecated, use the typeless endpoints instead /{index}/_doc/{id}, /{index}/_doc, or /{index}/_create/{id}.
  warnings.warn(message, category=ElasticsearchWarning)
Tweet 292066797 has been fetched. Fetched: 1 tweets
Tweet 292066793 has been fetched. Fetched: 2 tweets
Tweet 292066536 has been fetched. Fetched: 3 tweets
Tweet 292066521 has been fetched. Fetched: 4 tweets
Tweet 292066500 has been fetched. Fetched: 5 tweets
Tweet 292066495 has been fetched. Fetched: 6 tweets
Tweet 292066318 has been fetched. Fetched: 7 tweets
Tweet 292066305 has been fetched. Fetched: 8 tweets
Tweet 292066136 has been fetched. Fetched: 9 tweets
Tweet 292066066 has been fetched. Fetched: 10 tweets
Tweet 292067178 has been fetched. Fetched: 11 tweets
Tweet 292067170 has been fetched. Fetched: 12 tweets
Tweet 292067053 has been fetched. Fetched: 13 tweets
Tweet 292067555 has been fetched. Fetched: 14 tweets
Tweet 292067544 has been fetched. Fetched: 15 tweets
Tweet 292067474 has been fetched. Fetched: 16 tweets
Tweet 292067812 has been fetched. Fetched: 17 tweets
Tweet 292067775 has been fetched. Fetched: 18 tweets
Tweet 292067738 has been fetched. Fetched: 19 tweets
Tweet 292067666 has been fetched. Fetched: 20 tweets
Tweet 292067661 has been fetched. Fetched: 21 tweets
Tweet 292067592 has been fetched. Fetched: 22 tweets
Tweet 292068263 has been fetched. Fetched: 23 tweets
Tweet 292068247 has been fetched. Fetched: 24 tweets
Tweet 292068231 has been fetched. Fetched: 25 tweets
Tweet 292068096 has been fetched. Fetched: 26 tweets
Tweet 292068659 has been fetched. Fetched: 27 tweets
Tweet 292068621 has been fetched. Fetched: 28 tweets
Tweet 292068424 has been fetched. Fetched: 29 tweets
Tweet 292069056 has been fetched. Fetched: 30 tweets
Tweet 292068767 has been fetched. Fetched: 31 tweets
Tweet 292068739 has been fetched. Fetched: 32 tweets
Tweet 292068725 has been fetched. Fetched: 33 tweets
Tweet 292069332 has been fetched. Fetched: 34 tweets
Tweet 292069196 has been fetched. Fetched: 35 tweets
```

```

Tweet 292132033 has been fetched. Fetched: 957 tweets
Tweet 292132014 has been fetched. Fetched: 958 tweets
Tweet 292132176 has been fetched. Fetched: 959 tweets
Tweet 292132172 has been fetched. Fetched: 960 tweets
Tweet 292132155 has been fetched. Fetched: 961 tweets
Tweet 292132234 has been fetched. Fetched: 962 tweets
Tweet 292132211 has been fetched. Fetched: 963 tweets
Tweet 292132283 has been fetched. Fetched: 964 tweets
Tweet 292132265 has been fetched. Fetched: 965 tweets
Tweet 292132364 has been fetched. Fetched: 966 tweets
Tweet 292132336 has been fetched. Fetched: 967 tweets
Tweet 292132464 has been fetched. Fetched: 968 tweets
Error code: 502
Tweet 292132609 has been fetched. Fetched: 969 tweets
Tweet 292132578 has been fetched. Fetched: 970 tweets
Tweet 292132556 has been fetched. Fetched: 971 tweets
Tweet 292132544 has been fetched. Fetched: 972 tweets
Tweet 292132771 has been fetched. Fetched: 973 tweets
Tweet 292132915 has been fetched. Fetched: 974 tweets
Tweet 292133017 has been fetched. Fetched: 975 tweets
Tweet 292133013 has been fetched. Fetched: 976 tweets
Tweet 292133066 has been fetched. Fetched: 977 tweets
Tweet 292133208 has been fetched. Fetched: 978 tweets
Tweet 292133202 has been fetched. Fetched: 979 tweets
Tweet 292133194 has been fetched. Fetched: 980 tweets
Tweet 292133278 has been fetched. Fetched: 981 tweets
Tweet 292133263 has been fetched. Fetched: 982 tweets
Tweet 292133392 has been fetched. Fetched: 983 tweets
Tweet 292133365 has been fetched. Fetched: 984 tweets
Tweet 292054059 has been fetched. Fetched: 985 tweets
Tweet 292133497 has been fetched. Fetched: 986 tweets
Tweet 292133489 has been fetched. Fetched: 987 tweets
Tweet 292133479 has been fetched. Fetched: 988 tweets
Tweet 292133473 has been fetched. Fetched: 989 tweets
Tweet 292133467 has been fetched. Fetched: 990 tweets
Tweet 267474251 has been fetched. Fetched: 991 tweets
Tweet 292133536 has been fetched. Fetched: 992 tweets
Tweet 292133606 has been fetched. Fetched: 993 tweets
Tweet 292133603 has been fetched. Fetched: 994 tweets
Tweet 292133568 has been fetched. Fetched: 995 tweets
Tweet 292133678 has been fetched. Fetched: 996 tweets
Tweet 292133787 has been fetched. Fetched: 997 tweets
Tweet 292133772 has been fetched. Fetched: 998 tweets
Tweet 292133747 has been fetched. Fetched: 999 tweets
Tweet 292133812 has been fetched. Fetched: 1000 tweets

```

همانطور که مشاهده می‌کنید، ۱۰۰۰ توبیت در الستیک قابل مشاهده است.



The screenshot shows the Elasticsearch Dev Tools interface at localhost:5601/app/dev_tools/#/console. A search bar at the top contains the term "elastic". Below it, a navigation bar includes tabs for "Console", "Search Profiler", "Grok Debugger", and "Painless Lab (BETA)". The "Console" tab is selected. The main area displays a code editor with a query and its results. The query is:

```
1 GET /twitter/_search
```

The results are as follows:

```

1 #! Elasticsearch built-in security features are not enabled. Without
2 authentication, your cluster could be accessible to anyone. See https
3 ://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal
4 -setup.html to enable security.
5 {
6   "took" : 1,
7   "timed_out" : false,
8   "_shards" : {
9     "total" : 1,
10    "successful" : 1,
11    "skipped" : 0,
12    "failed" : 0
13  },
14  "hits" : {
15    "total" : {
16      "value" : 1000,
17      "relation" : "eq"
18    },
19    "max_score" : 1.0,
20    "hits" : [
21      {
22        "_index" : "twitter",
23        "_type" : "twitter",
24        "_id" : "ZX96p3kBmHTifil80N7k",
25        "_score" : 1.0,
26        "_source" : {
27          "id" : "292070655",
28          "sendTime" : "2021-05-26T07:01:14Z",
29          "sendTimePersian" : "1400/03/05 11:31",
30          "senderName" : "sajjad",
31          "senderUsername" : "sajjad6171",
32          "senderProfileImage" : "default",
33          "content" : "#"
34        }
35      }
36    ]
37  }
38}

```

A message at the bottom right of the results pane reads: "کاتا آن سلط قبیله ها منشی به نظر نمیگردید... اینا".

سه کوئری دلخواه روی این مجموع اجرا شده است که نتایج آنها را در ادامه می‌بینید.

۲۵ توبیت به این صورت وجود داشت.
ابتدا نماد کاما در میان هشتگ‌ها جستجو شده است تا توبیت‌هایی با این هشتگ نشان داده شوند.

The screenshot shows the Elasticsearch Dev Tools interface with the following details:

- URL:** localhost:5601/app/dev_tools#/console
- Toolbar:** Dev Tools, Search Elastic, BETA.
- Console Tab:** Selected.
- Query:** A GET request to /twitter/_search with the following JSON body:

```
1 GET /twitter/_search
2 GET /twitter/_search
3 {
4   "query": {
5     "query_string": {
6       "query": "\u20ac",
7       "fields": ["hashtags"]
8     }
9   }
10 }
```
- Response:** Status 200 OK, took 3ms, total hits 1, max score 3.4644156, with one hit object:

```
3   "took" : 2,
4   "timed_out" : false,
5   "_shards" : {
6     "total" : 1,
7     "successful" : 1,
8     "skipped" : 0,
9     "failed" : 0
10 },
11   "hits" : {
12     "total" : {
13       "value" : 25,
14       "relation" : "eq"
15     },
16     "max_score" : 3.4644156,
17     "hits" : [
18       {
19         "_index" : "twitter",
20         "_type" : "twitter",
21         "_id" : "2X9p63k8mHifil80N7k",
22         "_score" : 3.4644156,
23         "_source" : {
24           "id" : "292070655",
25           "sendTime" : "2021-05-26T07:01:14Z",
26           "sendTimePersian" : "1400/03/05 11:31",
27           "senderName" : "sajjad",
28           "senderUsername" : "sajjad6171",
29           "senderProfileImage" : "default",
30           "content" : "کاماب این سقوط قیمت ها میشه به نفع حقوقیهاست... مثل
```
- Text Overlay:** اینه که همان مرغ خلایق را با پیش شرک میتبدید شما بول لازمید وسی حقوقی میخواهد که مرغ روچاخد بشے پیش که همهاداران خرد به کمد هم اوئنلدر سهم رو یابینی بیاپن که حقوقی بندون میله بفرده دلیل سقوط قیمت ها دست باز حقوقیهاست در ایندای سقوط بازار و تبوده...

سپس، توییت‌هایی که در متن آن‌ها عبارت شاخص آمده است مورد جستجو قرار گرفتند. ۲۲ توییت به این صورت وجود داشت.

```

1 GET /twitter/_search
2 GET /twitter/_search
3 {
4   "query": {
5     "query_string": {
6       "query": "کاتخ",
7       "fields": ["hashtags"]
8     }
9   }
10 }
11 GET /twitter/_search
12 {
13   "query": {
14     "query_string": {
15       "query": "کاتخ",
16       "fields": ["content"]
17     }
18   }
19 }
20
21
22
23
24
25
26
27
28
29
30
31
32
33

```

سلام بر فعالین بازار سرمایه و مسئولین محترم سازمان بورس اقتصادی و فناوری بهم بگویید میدان داین انتقاله بازار را به بهبود شده مطمئن باشید اگر مثل قبیل فنار میگردید آن شاخ قرمز بود اگر میخواهد زخم بازار سرمایه با هزینه کم انتقام باید ۷۰٪ حمایت شاخ سازما را نظف گویان گیریان شاخ سازما یک روز پیشتر دام نواده اورد

```

1 GET /twitter/_search
2 GET /twitter/_search
3 {
4   "query": {
5     "query_string": {
6       "query": "کاتخ",
7       "fields": ["hashtags"]
8     }
9   }
10 }
11 GET /twitter/_search
12 {
13   "query": {
14     "query_string": {
15       "query": "کاتخ",
16       "fields": ["content"]
17     }
18   }
19 }
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

```

سلام بر فعالین بازار سرمایه و مسئولین محترم سازمان بورس اقتصادی و فناوری بهم بگویید میدان داین انتقاله بازار را به بهبود شده مطمئن باشید اگر مثل قبیل فنار میگردید آن شاخ قرمز بود اگر میخواهد زخم بازار سرمایه با هزینه کم انتقام باید ۷۰٪ حمایت شاخ سازما را نظف گویان گیریان شاخ سازما یک روز پیشتر دام نواده اورد

آن بر سهام داران حقیقتی روشن شده شاخ سازما نهاد تابیت کامل بازار سمت شاخ سازما خواهد رفت

بسیار ساده شاخ سازما خواهد بود

"content": "#فناوریس#"

فناوریس

ویضاد کسی که تا حال تو بورس مونده بیشی دیگر نمیخواهد

بنروشه، حال یعنی بیان از در ترس بازاند، مر روز منتفی کنن شاخ را

هیچ فرقی ندارد، این افراد دیگه تا آخر تو بورس موندگانند، بخوبی

آخرین کوئری نیز به بررسی تويیت‌هایی که هم از شپنا و هم از پالایش در متن صحبت به میان آورده بودند بررسی شدند تا تحلیل‌های مشترک این دو به دست آید. حاصل تنها یک تويیت بود البته تويیت مفصلی!

```

14+     "query": {
15+       "query_string": {
16+         "query": "شپنا"
17+       }
18+     }
19+   }
20 GET /twitter/_search
21 {
22   "query": {
23     "bool": {
24       "must": [
25         {
26           "match": {
27             "content": "پالایش"
28           }
29         },
30         {
31           "match": {
32             "content": "شپنا"
33           }
34         }
35       ]
36     }
37   }
38 }
39
40
41
42
43
44
45

```

max_score : 5.102227,
 hits : [
 {
 _index : "twitter",
 _type : "twitter",
 _id : "bn_p3kBmHTifil8ouci",
 _score : 5.102227,
 _source : {
 id : "29211318",
 sendTime : "2021-05-26T08:16:51Z",
 sendTimePersian : "1400/03/05 12:46",
 senderName : "Sina222",
 senderUsername : "poorsina222",
 senderUserImage : "default",
 content : "#فولاد_شپنا_شیندر این عرضه اولینه ها را توزیع بازار می کنند بعد بازار را شکم می زنند و هما میگذارند خاطر چند روز سود خودشان بازار را خراب نمی کنند کجا می آیند تو بازاری که درد، نقدینگی دارد عرضه اولیه انجام می دهند، همیزی قتل و خیانت آشکار به مردم، اینها فقط به فکر ناماین بودجه خودشون مستند و کلایم‌داری از مردم، عرضه اولینه های قیلی را تکاه کنند اون چه وضعیتی دارند؟ پس این دادار یکم و رججه کسانی به این کلایم‌داری داشتند من و قابایی که بازم غریب ار مستندم. کدام آدم اتفاقی فعل ، فولاد ، شپنا ،شیندر و سهام با ازیز دیگر را من فروخته عرضه اولینه می خرید .

کوئری مربوط به این قسمت در فایل `twitter_script.py` آمده است.
 در انتهای سه ویژوالیزیشن در یک داشبورد قرار داده شده‌اند که هر کدام را در ادامه می‌بینیم.
 + ابر هشتگ‌ها:

- نحوه ایجاد: گزینه tag cloud

New visualization

- Emphasize the data between an axis and a line.
- Display data in rows and columns.
- Show the status of a metric.
- Goal** Track how a metric progresses to a goal.
- Heat map** Shade data in cells in a matrix.
- Horizontal bar** Present data in horizontal bars on an axis.
- Line** Display data as a series of points.
- Metric** Show a calculation as a single number.
- Pie** Compare data in proportion to a whole.
- Tag cloud** Display word frequency with font size.
- Timelion** Show time series data on a graph.
- Vertical bar** Present data in vertical bars on an axis.

خروجی:

+ تعداد نمادهای پر تکرار:

Lens - Elastic

localhost:5601/app/lens#/edit_by_value?_g=(filters:{},refreshInterval:(pause:!t,value:0),time:(from:now-15m,to:now))

elastic

Dashboard / Edit visualization

Search Elastic

Download as CSV Cancel Save to library Save and return

Search Add filter

KQL Last 15 minutes Show dates Refresh

twitter*

Search field names

Field filters 0

Records

Available fields 32

content.keyword

finalPullDatePersian.keyword

hasChart.keyword

hashtags.keyword

id.keyword

imageUid.keyword

lastLikeNickName.keyword

likeCount.keyword

mediaContentType.keyword

Bar vertical stacked

Count of records

Top values of hashtags.keyword

Horizontal axis

Top values of hashtags.keyword

Vertical axis

Count of records

Drop a field or click to add

Break down by

Drop a field or click to add

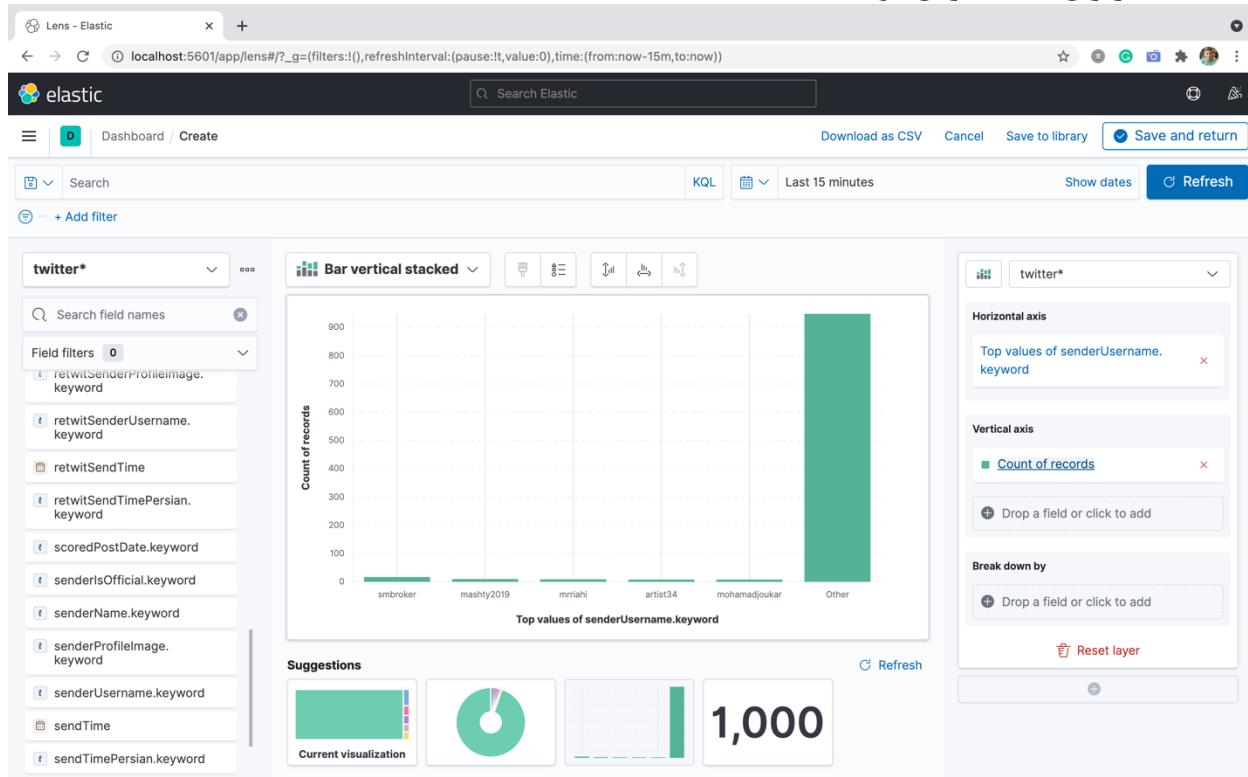
Reset layer

Suggestions

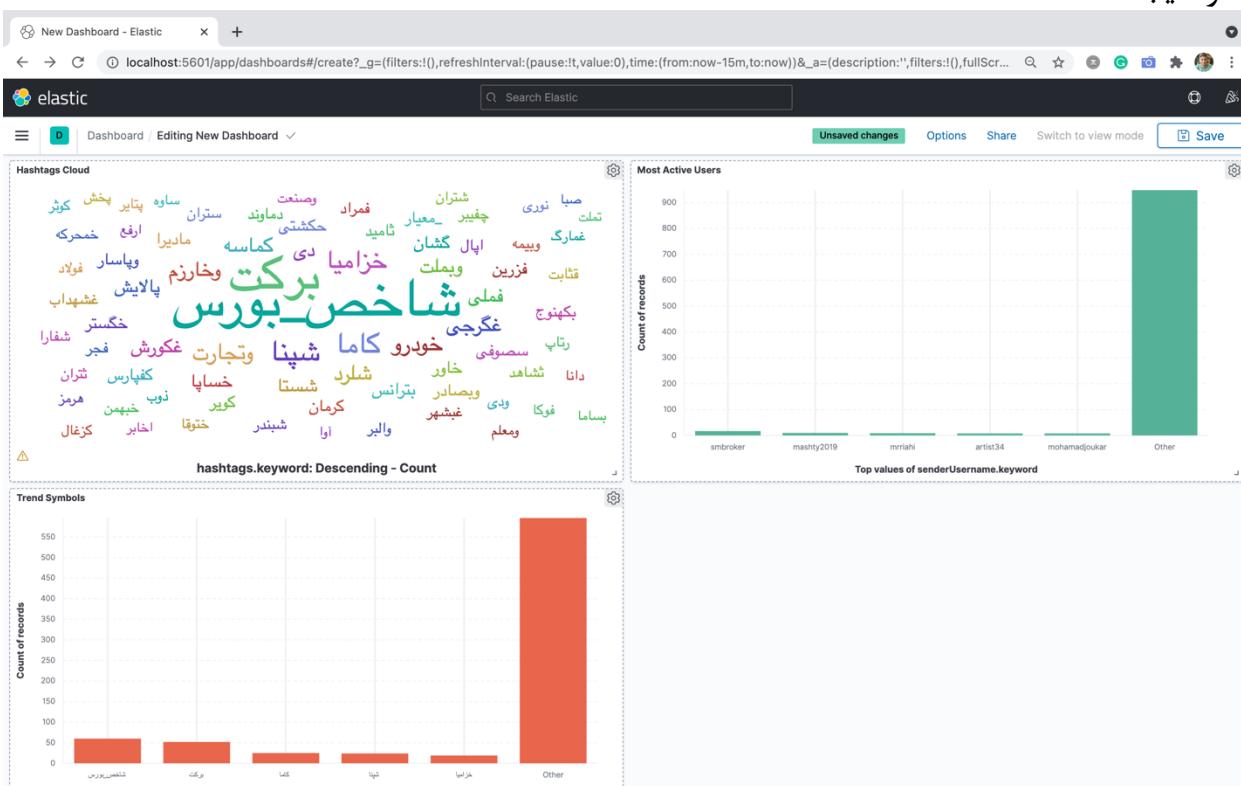
Current visualization

1,00

+ تعداد کاربران با بیشترین توییت:



+ داشبورد نهایی:
- نحوه ایجاد:



- خروجی نهایی:

