به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس کلان داده

تمرین شماره دو

نام و نام خانوادگی : ?

شماره دانشجویی : ?

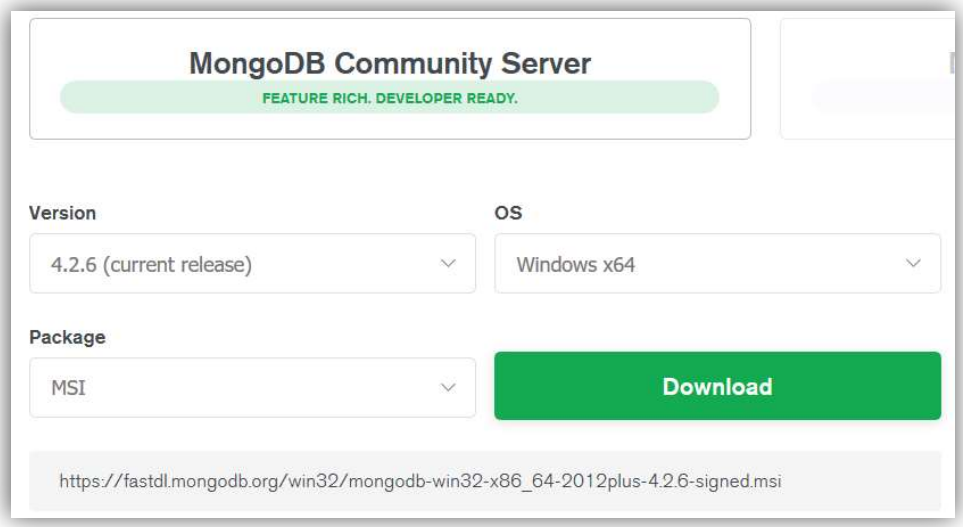
خرداد 99

فهرست گزارش سوالات

بخش اول: ذخیره اطلاعات بدون ساختار / کار با MongoDB

مراحل دانلود، نصب و آماده‌سازی کار با MongoDB به‌صورت زیر می‌باشد:

1- ابتدا از آدرس https://www.mongodb.com/download-center/community، مطابق با تصویر زیر، نسخه‌ی 4.2.6 مانگودی‌بی را به‌صورت بسته‌بندی MSI برای سیستم‌عامل ویندوز دانلود می‌کنیم:



2- سپس در آدرس پیش‌فرض خود که همان **C:\Program Files\MongoDB\Server\4.2** است، آنرا نصب می‌کنیم.

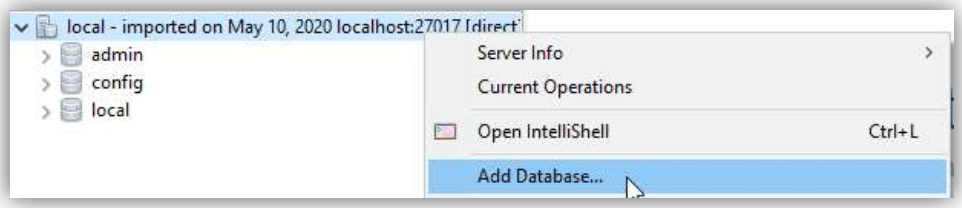
3- به‌طور پیش‌فرض داده‌های مانگو در آدرس **C:\data\db** ذخیره خواهند شد، به‌همین دلیل، در درایو C یک فولدر به نام data را ایجاد می‌کنیم و سپس در داخل آن یک فولدر به‌نام db می‌سازیم تا داده‌های مانگو بدون مشکل در این فولدر ذخیره شوند.

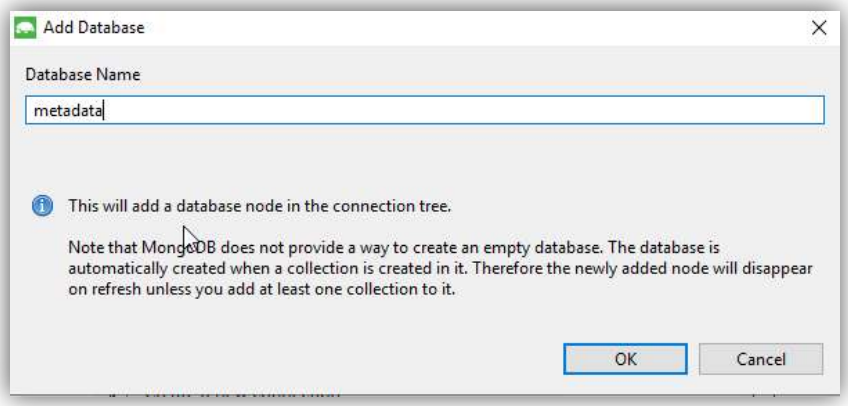
4- سپس آدرس **C:\Program Files\MongoDB\Server\4.2\bin** را به Pathهای سیستم اضافه می‌کنیم.

5- در نهایت با اجرای دستور mongod.exe در cmd، مانگودی‌بی را اجرا می‌کنیم. در ضمن اگر بخواهیم که از خط فرمان مانگودی‌بی (mongo shell) استفاده کنیم، دستور mongo.exe را در cmd اجرا می‌کنیم.

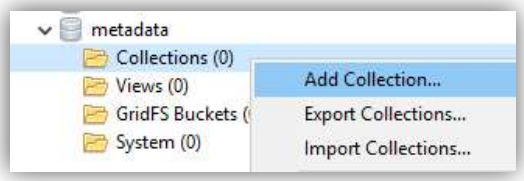
6- در اینجا به‌جای خط فرمان مانگودی‌بی از ابزار گرافیکی بسیار قدرتمند Studio 3T استفاده می‌کنیم که نسخه‌ی قدرتمندتر ابزار Robo 3T می‌باشد.

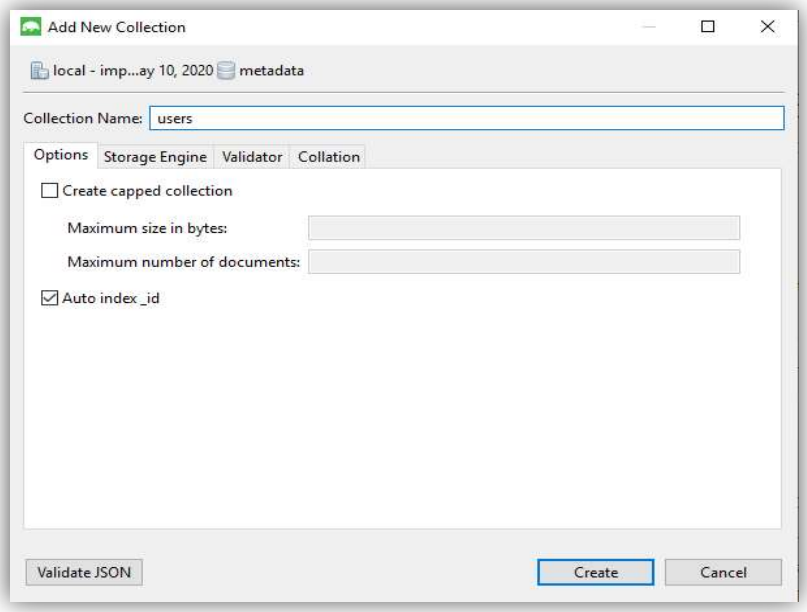
7- در Studio 3T، ابتدا برای اتصال به DB Server به آدرس **localhost:27017** متصل می‌شویم. سپس دیتابیس metadata را به‌صورت زیر ایجاد می‌کنیم:





پس از ایجاد دیتابیس metadata، کالکشن users را به دیتابیس metadata اضافه می‌نماییم:





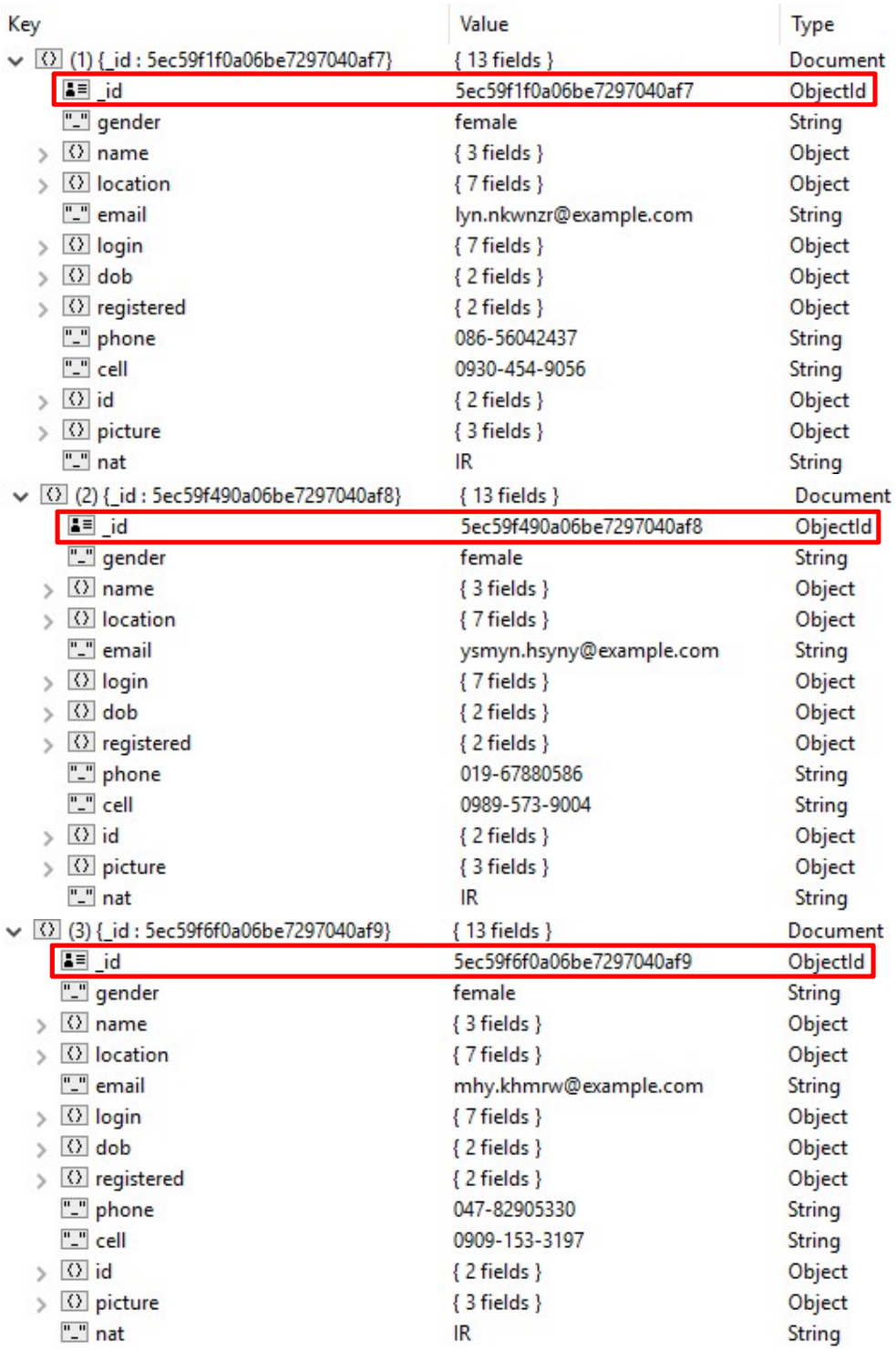
گام اول تمرین – درج داده‌ها

ابتدا گزینه‌ی 

را انتخاب کرده تا به خط فرمان مانگودی‌بی دسترسی داشته باشیم. سپس آدرس https://randomuser.me/api/?nat=ir را فراخوانی کرده، فایل JSON آنرا کپی می‌کنیم و سپس اطلاعات موجود در فیلد results را با دستور زیر در کالکشن users وارد می‌کنیم:

```
db.getCollection("users").insertOne({"gender":"female","name":{"title":"Ms","first":
"الینا","last":"نکو نظر"},"location":{"street":{"number":288,"name":" شهید محمد
منتظری"},"city":"اهواز","state":"مرکزی","country":"Iran","postcode":32571,"coordinates"
:{"latitude":"45.7234","longitude":"100.7914"},"timezone":{"offset":"-
11:00","description":"Midway Island,
Samoa"}},"email":"lyn.nkwnzr@example.com","login":{"uuid":"3bd0072a-4a54-47da-b025-
a26f3eb9a074","username":"goldenmouse258","password":"boat","salt":"OvrEYO9c","md5":
"0eb8b489793ee4c7dd78dd0d0befb833","sha1":"2313446429c10d1a494ddc8e7b363843fb8de37e"
,"sha256":"f0119ba3f1166cdbf1e53305cc9131d7d06bd9f9a893ef1ae9e41e3ce4d39861"},"dob":
{"date":"1967-03-23T19:18:51.750Z","age":53},"registered":{"date":"2019-07-
16T03:50:04.010Z","age":1},"phone":"086-56042437","cell":"0930-454-
9056","id":{"name":"","value":null},"picture":{"large":"https://randomuser.me/api/po
rtraits/women/18.jpg","medium":"https://randomuser.me/api/portraits/med/women/18.jpg
","thumbnail":"https://randomuser.me/api/portraits/thumb/women/18.jpg"},"nat":"IR"})
```

همین عمل را سه بار تکرار کرده و در نهایت از کالکشن users خروجی می‌گیریم (در صفحه‌ی بعد). همانطور که مشاهده می‌شود، به ازای هر insertای که داشته‌ایم، یک ObjectId با نام _id به‌عنوان کلید به مشخصات کاربر اضافه می‌شود.

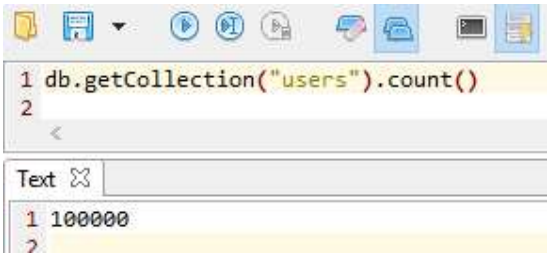


Key	Value	Type
(1) {_id : 5ec59f1f0a06be7297040af7}	{ 13 fields }	Document
_id	5ec59f1f0a06be7297040af7	ObjectId
gender	female	String
name	{ 3 fields }	Object
location	{ 7 fields }	Object
email	lyn.nkwnzr@example.com	String
login	{ 7 fields }	Object
dob	{ 2 fields }	Object
registered	{ 2 fields }	Object
phone	086-56042437	String
cell	0930-454-9056	String
id	{ 2 fields }	Object
picture	{ 3 fields }	Object
nat	IR	String
(2) {_id : 5ec59f490a06be7297040af8}	{ 13 fields }	Document
_id	5ec59f490a06be7297040af8	ObjectId
gender	female	String
name	{ 3 fields }	Object
location	{ 7 fields }	Object
email	ysmyn.hsyny@example.com	String
login	{ 7 fields }	Object
dob	{ 2 fields }	Object
registered	{ 2 fields }	Object
phone	019-67880586	String
cell	0989-573-9004	String
id	{ 2 fields }	Object
picture	{ 3 fields }	Object
nat	IR	String
(3) {_id : 5ec59f6f0a06be7297040af9}	{ 13 fields }	Document
_id	5ec59f6f0a06be7297040af9	ObjectId
gender	female	String
name	{ 3 fields }	Object
location	{ 7 fields }	Object
email	mhy.khmrw@example.com	String
login	{ 7 fields }	Object
dob	{ 2 fields }	Object
registered	{ 2 fields }	Object
phone	047-82905330	String
cell	0909-153-3197	String
id	{ 2 fields }	Object
picture	{ 3 fields }	Object
nat	IR	String

سپس اطلاعات قبل را پاک کرده و با استفاده از کتابخانه‌ی pymongo و به‌صورت زیر، اطلاعات 100000 کاربر تصادفی را در مانگو ذخیره می‌کنیم:

```
from pymongo import MongoClient
import requests, json, time
url = "https://randomuser.me/api/"

client = MongoClient("mongodb://localhost:27017/")
database = client["metadata"]
collection = database["users"]

def get_random_users(n):
    for i in range(0,n//5000):
        querystring = {"nat": "ir", "results": "5000"}
        response = None
        while True:
            response = requests.request("GET", url, params=querystring)
            if response.status_code==503:
                print("120 secs wait...")
                time.sleep(120)
            else:
                break
        data = json.loads(response.text)
        collection.insert_many(data['results'])
        print(i+1)
    print("finished.")
get_random_users(n=100000)
```

در نهایت، با دستور count مطمئن می‌شویم که 100000 کاربر تصادفی در کالکشن users ذخیره شده باشد:



گام دوم تمرین – دستورات اصلی

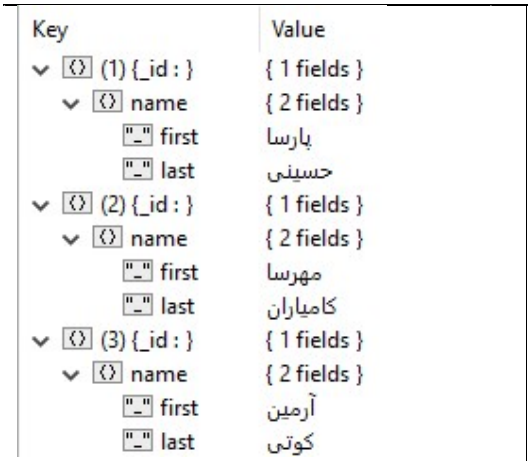
1. نام و نام‌خانوادگی کاربرانی که بالای 50 سال سن داشته و ساکن نیشابور باشند:

کوئری:

```
db.getCollection("users").find(
    {
        "dob.age":{
            "$gte": 50
        },
        "location.city": "نیشابور"
    },
    {
        "name.first": 1,
        "name.last" : 1,
        "_id" : 0
    }
)
```

خروجی:

Tree



JSON

```
{
    "name" : {
        "first" : "پارسا",
        "last" : "حسینی"
    }
}
{
    "name" : {
        "first" : "مهرسا",
        "last" : "کامیاران"
    }
}
{
    "name" : {
        "first" : "آرمین",
        "last" : "کوتی"
    }
}
```

* لیست کامل خروجی که شامل 995 مورد است، در آدرس **Code\Part 1- Mongodb\Step 2 - main commands** قرار گرفته است.

زمان اجرا: 0.191s

2. نام خانوادگی، آدرس و شماره موبایل کاربرانی که بیش از 20 سال است که در سایت ما ثبت‌نام کرده‌اند:

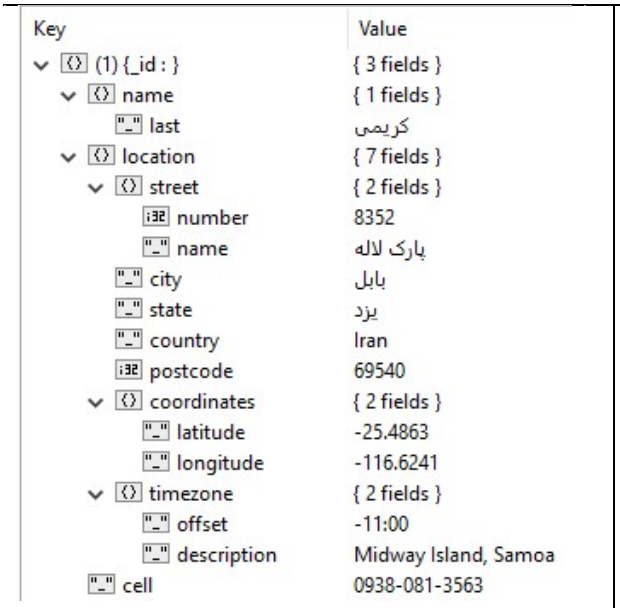
نکته: با توجه به اینکه کاربران این سایت، حداکثر 18 سال پیش در این سایت ثبت‌نام‌کرده‌اند، در نتیجه برای اینکه سوال ذکر شده، خروجی داشته باشد، به‌جای 20، از 18 استفاده خواهیم کرد.

کوئری:

```
db.getCollection("users").find(
    {
        "registered.age":{
            "$gte": 18
        }
    },
    {
        "name.last" : 1,
        "location": 1,
        "cell": 1,
        "_id": 0
    }
)
```

خروجی:

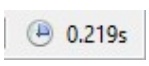
Tree



JSON

* لیست کامل خروجی که شامل 4399 مورد است، در آدرس **Code\Part 1- Mongodb\Step 2 - main commands** قرار گرفته است.

زمان اجرا:



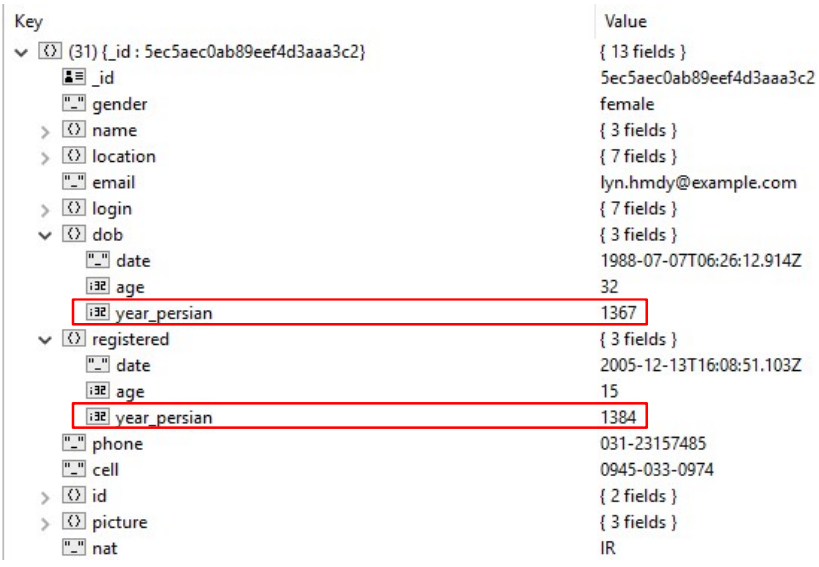
3. افزودن فیلد year_persian به فیلد dob و registered:

کوئری:

```
db.getCollection("users").update(
    {

    },
    [
        {
            "$set":{
                "dob.year_persian":{
                    "$year":{
                        "$subtract":[{"$dateFromString":{"dateString":"$dob.date"}},226899*24*60*60000]
                    }
                },
                "registered.year_persian":{
                    "$year":{
                        "$subtract":[{"$dateFromString":{"dateString":"$registered.date"}},226899*24*60*60000]
                    }
                }
            }
        }
    ],
    {
        "multi":true
    }
)
```

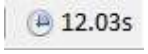
توضیح: ابتدا نوع فیلد dob.date که به‌صورت رشته ذخیره شده است را به Date تغییر می‌دهیم. سپس با توجه به اینکه اختلاف تاریخ شمسی و میلادی 226899 روز می‌باشد، 226899 روز از این تاریخ کم می‌کنیم تا به تاریخ شمسی دست یابیم. در نهایت با استفاده از $year، سال را از تاریخ شمسی حاصل استخراج می‌کنیم و آنرا به فیلد جدیدی به نام dob.year_persian اختصاص می‌دهیم. ما این عمل را برای registered.date نیز انجام می‌دهیم.

خروجی کالکشن users به‌صورت Tree:



* به‌همین صورت تمام ردیف‌های کالکشن users ویرایش‌شده، حاوی زیرفیلد year_persian در فیلد dob و registered هستند. با توجه به اینکه خروجی این سوال، حدود 180مگابایت حجم دارد، از ارائه‌ی آن صرف نظر می‌کنیم.

زمان اجرا:



4. دستوری بنویسید که با اجرا شدن آن در هر روز، نام و نام خانوادگی و ایمیل افرادی که در آن روز تولدشان است، به ما برگردانده شود:

کوئری:

```
db.getCollection("users").find(
    {
        "$expr":{
            "$and":[
                    {
                        "$eq":[
                                {"$month":{date:{"$dateFromString":{"dateString":"$dob.date"}}}},
                                {"$month":{date:new Date()}}
                        ]
                    },
                    {
                        "$eq":[
                                {"$dayOfMonth":{date:{"$dateFromString":{"dateString":"$dob.date"}}}},
                                {"$dayOfMonth":{date:new Date()}}
                        ]
                    }
            ]
        }
    },
    {
        "name.first": 1,
        "name.last": 1,
        "email":1,
        "_id":0
    }
)
```

توضیح: در کوئری بالا، نام و نام‌خانوادگی افرادی که روز و ماه تولدشان با روز و ماه جاری برابر است، برگردانده می‌شود. لازم به ذکر است که میتوانستیم بجای $month و $dayOfMonth از $dayOfYear استفاده کنیم، ولی چونکه با این روش، سال کبیسه در نظر گرفته نمی‌شود، به‌همین دلیل از روش اول استفاده نمودیم.

خروجی:

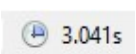
Tree

JSON

```
{
    "name" : {
        "first" : "مهدي",
        "last" : "نجاتی"
    },
    "email" : "mhdy.njty@example.com"
}
{
    "name" : {
        "first" : "محمدپارسا",
        "last" : "سلطانی نژاد"
    },
    "email" : "mhmdprs.sltnynjd@example.com"
}
```

* لیست کامل خروجی که شامل 251 مورد است، در آدرس Code\Part 1- Mongodb\Step 2 - main commands\ قرار گرفته است.

زمان اجرا:



5. ذخیره پسورد به شکل خام در اطلاعات کاربر، کاری غیرحرفه‌ای است. میخواهیم این مشکل را برطرف کنیم. چه راه حلی برای حل مساله پیشنهاد میکنید؟

برای حل این مسئله، می‌توان بجای پسورد از hex_md5 آن استفاده کرد.

راه حل را ابتدا روی یک سند خاص، امتحان کنید و مطمئن شوید بعد از اعمال تغییرات، آن کاربر خاص را با داشتن یوزنیم و پسورد، میتوانید پیدا کنید. سپس تمام کاربران را به روز رسانی کنید.

باتوجه به اینکه hex_md5 یک تابع جاوا اسکریپتی است، در نتیجه می‌توان از این تابع در خط فرمان مانگودی‌بی استفاده کرد. ولی بایستی ورودی این تابع، پسورد کاربر مورد نظر باشد. ولی مشکلی که وجود دارد این است که در یک تابع جاوا اسکریپتی، نمی‌توان با "$login.password"، مقدار پسورد کاربر مورد نظر را دریافت و از آن به‌عنوان ورودی تابع hex_md5 استفاده کرد. به‌همین دلیل ما مجبور شدیم که از foreach بر روی find استفاده کنیم تا بتوانیم با دستورات جاوا اسکریپتی، پسورد کاربر(های) مورد نیاز را استخراج و از آنها به‌عنوان ورودی تابع hex_md5 استفاده کنیم. لازم به ذکر است که با توجه به اینکه استفاده از دستورات و توابع جاوااسکریپتی در مانگودی‌بی مجاز است، در نتیجه تمام دستورات زیر در خط فرمان مانگودی‌بی قابل اجرا است.

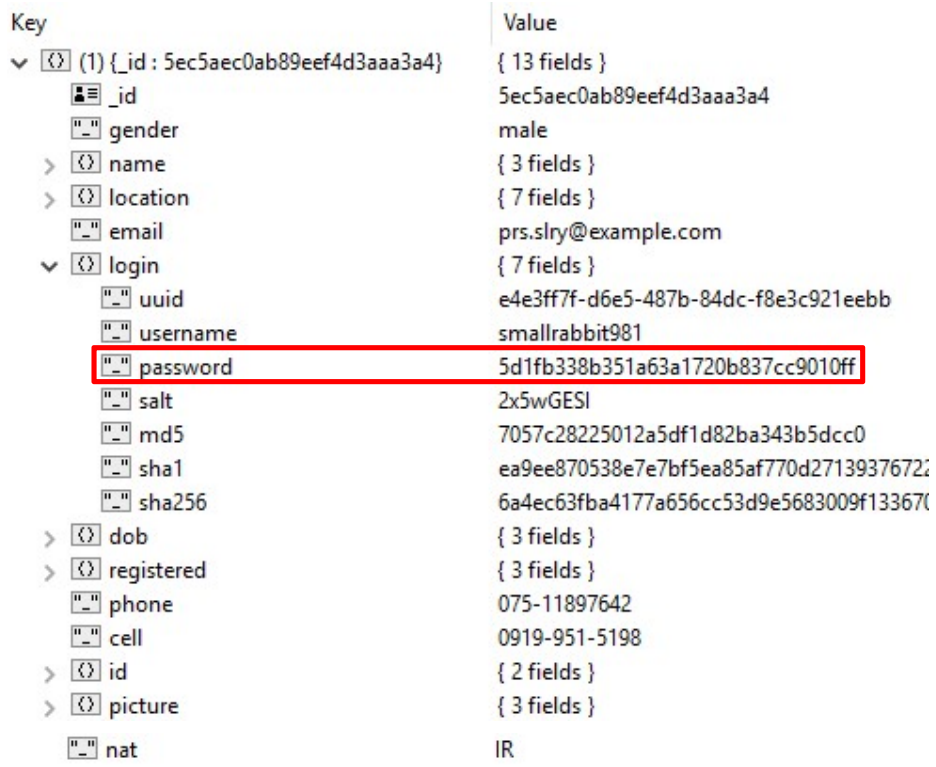
ابتدا با استفاده از کوئری زیر، به‌جای پسورد کاربری با نام کاربری `smallrabbit981`، مقدار hex_md5 آنرا قرار می‌دهیم:

```
db.getCollection("users").find({"login.username":"smallrabbit981"}).forEach(function(data){
    data.login.password = hex_md5(data.login.password)
    db.getCollection("users").save(data)
});
```

پسورد خام کاربر `smallrabbit981` ، daytona بود. حال اگر بخواهیم که این کاربر را با داشتن نام‌کاربری و پسورد خام پیدا کنیم، کافیست از کوئری زیر استفاده کنیم:

```
1 db.getCollection("users").find(
2     {
3         "login.username":"smallrabbit981",
4         "login.password":hex_md5("daytona")
5     }
6 )
```

که خروجی آن به‌صورت زیر می‌باشد:



با توجه به اینکه مطمئن شدیم که عملکرد روش ما درست است، در نتیجه با کوئری زیر، پسورد تمام کاربران را به‌روز رسانی می‌کنیم (البته قبل از آن، پسورد کاربر بالا را به‌صورت قبل برمی‌گردانیم):

```
db.getCollection("users").find().forEach(function(data){
    data.login.password = hex_md5(data.login.password)
    db.getCollection("users").save(data)
});
```

همانطور که از خروجی username و password چند کاربر اول کالکشن users به‌صورت زیر مشخص است، پسورد کاربران تغییر کرده است:

```
{
    "login" : {
        "username" : "smallrabbit981",
        "password" : "5d1fb338b351a63a1720b837cc9010ff"
    }
}
{
    "login" : {
        "username" : "beautifulmeercat896",
        "password" : "507fcbb50bf9e88168dcd0d083600c0b"
    }
}
{
    "login" : {
        "username" : "whiterabbit388",
        "password" : "7bf4a177e8677227277c63b6d5352741"
    }
}
```

زمان اجرای کوئری به‌روز رسانی پسورد تمام کاربران نیز 2دقیقه و 2 ثانیه و 50 صدم ثانیه به‌طول انجامید.

گام سوم تمرین – دستورات تجمعی و آماری

1. ابتدا بایستی کاربران را به سه گروه سنی کمتر از 16 (نوجوان) / بین 16 تا 30 (جوان) / بالاتر از 30 (میانسال به بالا) تقسیم کنید، سپس دستوری بنویسید که تعداد هر گروه را به ما برگرداند:

کوئری:

```
db.getCollection("users").aggregate(
    [
        {
            "$project" : {
                "AgeGroup" : {
                    "$cond" : {
                        "if" : { "$lt" : ["$dob.age", 16.0] },
                        "then" : "teenager",
                        "else" : {
                            "$cond" : {
                                "if" : { "$lte" : [ "$dob.age", 30.0 ] },
                                "then" : "young",
                                "else" : "middle-aged"
                            }
                        }
                    }
                }
            }
        },
        {
            "$group" : {
                "_id" : "$AgeGroup",
                "count" : { "$sum" : 1.0 }
            }
        },
        {
            "$project" : {
                "_id" : 0.0,
                "age-group" : "$_id",
                "count" : { "$toInt" : "$count" }
            }
        }
    ]
)
```

خروجی:

Tree

Key	Value
(1) {_id : }	{ 2 fields }
age-group	young
count	16006
(2) {_id : }	{ 2 fields }
age-group	middle-aged
count	83994

JSON

```
{
    "age-group" : "young",
    "count" : NumberInt(16006)
}
{
    "age-group" : "middle-aged",
    "count" : NumberInt(83994)
}
```

زمان اجرا: 0.389s

2. تعداد کاربران هر استان را به تفکیک تولید کنید:

کوئری:

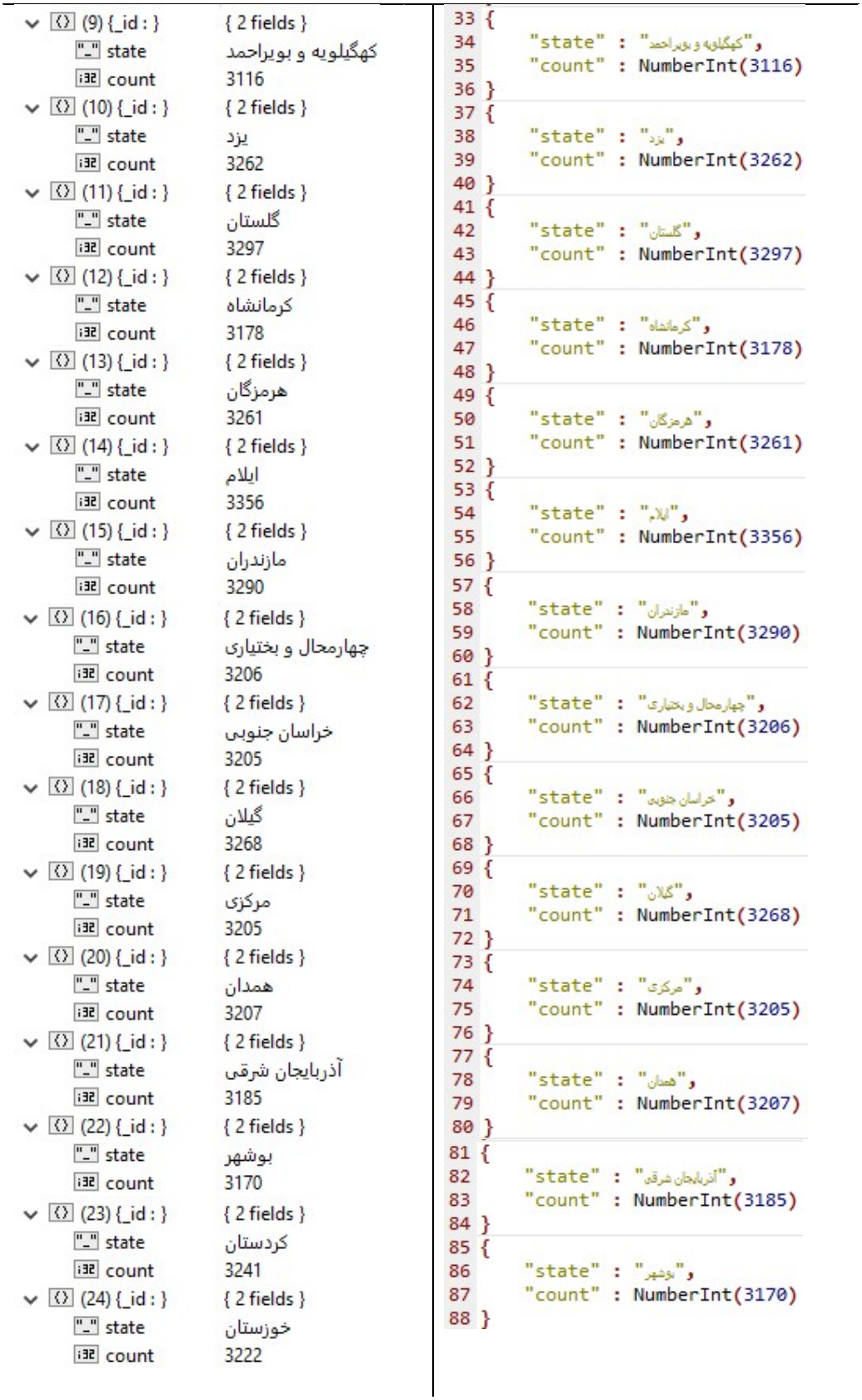
```
db.getCollection("users").aggregate(
    [
        {
            "$group" : {
                "_id" : "$location.state",
                "count" : { "$sum" : 1.0 }
            }
        },
        {
            "$project" : {
                "_id" : 0.0,
                "state" : "$_id",
                "count" : { "$toInt" : "$count" }
            }
        }
    ]
)
```

خروجی:

Tree

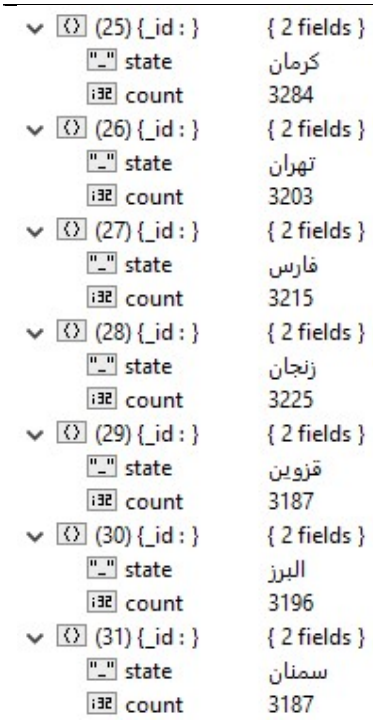
JSON

```
{
    "state" : "قم",
    "count" : NumberInt(3253)
}
{
    "state" : "آذربایجان غربی",
    "count" : NumberInt(3186)
}
{
    "state" : "سیستان و بلوچستان",
    "count" : NumberInt(3239)
}
{
    "state" : "خراسان رضوی",
    "count" : NumberInt(3324)
}
{
    "state" : "اردبیل",
    "count" : NumberInt(3187)
}
{
    "state" : "خراسان شمالی",
    "count" : NumberInt(3196)
}
{
    "state" : "لرستان",
    "count" : NumberInt(3205)
}
{
    "state" : "اصفهان",
    "count" : NumberInt(3244)
}
```



(9) {_id : }	{ 2 fields }
state	کهگیلویه و بویراحمد
count	3116
(10) {_id : }	{ 2 fields }
state	یزد
count	3262
(11) {_id : }	{ 2 fields }
state	گلستان
count	3297
(12) {_id : }	{ 2 fields }
state	کرمانشاه
count	3178
(13) {_id : }	{ 2 fields }
state	هرمزگان
count	3261
(14) {_id : }	{ 2 fields }
state	ایلام
count	3356
(15) {_id : }	{ 2 fields }
state	مازندران
count	3290
(16) {_id : }	{ 2 fields }
state	چهارمحال و بختیاری
count	3206
(17) {_id : }	{ 2 fields }
state	خراسان جنوبی
count	3205
(18) {_id : }	{ 2 fields }
state	گیلان
count	3268
(19) {_id : }	{ 2 fields }
state	مرکزی
count	3205
(20) {_id : }	{ 2 fields }
state	همدان
count	3207
(21) {_id : }	{ 2 fields }
state	آذربایجان شرقی
count	3185
(22) {_id : }	{ 2 fields }
state	بوشهر
count	3170
(23) {_id : }	{ 2 fields }
state	کردستان
count	3241
(24) {_id : }	{ 2 fields }
state	خوزستان
count	3222

```
{
    "state" : "کهگیلویه و بویراحمد",
    "count" : NumberInt(3116)
}
{
    "state" : "یزد",
    "count" : NumberInt(3262)
}
{
    "state" : "گلستان",
    "count" : NumberInt(3297)
}
{
    "state" : "کرمانشاه",
    "count" : NumberInt(3178)
}
{
    "state" : "هرمزگان",
    "count" : NumberInt(3261)
}
{
    "state" : "ایلام",
    "count" : NumberInt(3356)
}
{
    "state" : "مازندران",
    "count" : NumberInt(3290)
}
{
    "state" : "چهارمحال و بختیاری",
    "count" : NumberInt(3206)
}
{
    "state" : "خراسان جنوبی",
    "count" : NumberInt(3205)
}
{
    "state" : "گیلان",
    "count" : NumberInt(3268)
}
{
    "state" : "مرکزی",
    "count" : NumberInt(3205)
}
{
    "state" : "همدان",
    "count" : NumberInt(3207)
}
{
    "state" : "آذربایجان شرقی",
    "count" : NumberInt(3185)
}
{
    "state" : "بوشهر",
    "count" : NumberInt(3170)
}
```



(25) {_id : }	{ 2 fields }
state	کرمان
count	3284
(26) {_id : }	{ 2 fields }
state	تهران
count	3203
(27) {_id : }	{ 2 fields }
state	فارس
count	3215
(28) {_id : }	{ 2 fields }
state	زنجان
count	3225
(29) {_id : }	{ 2 fields }
state	قزوین
count	3187
(30) {_id : }	{ 2 fields }
state	البرز
count	3196
(31) {_id : }	{ 2 fields }
state	سمنان
count	3187

```
{
    "state" : "کردستان",
    "count" : NumberInt(3241)
}
{
    "state" : "خوزستان",
    "count" : NumberInt(3222)
}
{
    "state" : "کرمان",
    "count" : NumberInt(3284)
}
{
    "state" : "تهران",
    "count" : NumberInt(3203)
}
{
    "state" : "فارس",
    "count" : NumberInt(3215)
}
{
    "state" : "زنجان",
    "count" : NumberInt(3225)
}
{
    "state" : "قزوین",
    "count" : NumberInt(3187)
}
{
    "state" : "البرز",
    "count" : NumberInt(3196)
}
{
    "state" : "سمنان",
    "count" : NumberInt(3187)
}
```

زمان اجرا: 0.247s

3. فیلد شماره موبایل افرادی که آفست timezone آنها برابر 5:00+ را حذف نمایید، سپس تعداد افرادی را بیابید که شماره موبایل ندارند.

کوئری حذف شماره موبایل:

```
db.getCollection("users").update(
    {
        "location.timezone.offset": "+5:00"
    },
    [{
        $unset:"cell"
    }],
    {
        "multi":true
    }
)
```

کوئری جست‌وجوی تعداد افراد:

```
db.getCollection("users").aggregate(
    [
        {
            "$match" : {
                "cell" : { "$exists" : false }
            }
        },
        {
            "$group" : {
                "_id" : null,
                "count" : { "$sum" : 1 }
            }
        },
        {
            "$project" : {
                "_id" : 0,
                "count" : { "$toInt" : "$count" }
            }
        }
    ]
)
```

خروجی:

Tree

Key	Value
(1) {_id : }	{ 1 fields }
count	3398

JSON

```
{
    "count" : NumberInt(3398)
}
```

زمان اجرا:

0.096s

4. میانگین سن کاربران استان تهران را با میانگین سن کاربران سایر استان‌ها مقایسه کنید:

کوئری:

درک اینجانب از سوال، این است که میانگین سن کاربران استان تهران را با میانگین سن دیگر کاربران مقایسه کنیم. به‌همین دلیل، کوئری ما به‌صورت زیر می‌باشد:

```
db.getCollection("users").aggregate(
    [
        {
            "$project" : {
                "state" : {
                    "$cond" : {
                        "if" : { "$eq" : [ "$location.state",  "تهران" ] },
                        "then" : "تهران",
                        "else" : "سایر"
                    }
                },
                "age" : "$dob.age"
            }
        },
        {
            "$group" : {
                "_id" : "$state",
                "avgAge" : { "$avg" : "$age"}
            }
        },
        {
            "$project" : {
                "_id" : 0.0,
                "state" : "$_id",
                "avgAge" : "$avgAge"
            }
        }
    ]
)
```

خروجی:

Tree

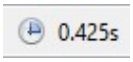
Key	Value
(1) {_id : }	{ 2 fields }
state	سایر
avgAge	48.87177288552331
(2) {_id : }	{ 2 fields }
state	تهران
avgAge	48.39993755853887

JSON

```
{
    "state" : "سایر",
    "avgAge" : 48.87177288552331
}
{
    "state" : "تهران",
    "avgAge" : 48.39993755853887
}
```

همانطورکه مشاهده می‌شود، میانگین سن کاربران استان تهران از میانگین سن دیگر کاربران کمتر است.

زمان اجرا:



کوئری: ولی اگر هدف سوال از مقایسه این است که مشخص کنیم که چه استان‌هایی، میانگین سن کمتر، چه استان‌هایی، میانگین سن برابر و چه استان‌هایی، میانگین سن بیشتر از استان تهران را دارند، کوئری آن به‌صورت زیر می‌باشد:

```
db.getCollection("users").aggregate(
    [
        {
            "$group" : {
                "_id" : "$location.state",
                "avgAge" : { "$avg" : "$dob.age" }
            }
        },
        {
            "$facet" : {
                "tehran" : [
                    {
                        "$match" : { "$expr" : { "$eq" : [ "$_id", "تهران"] }}
                    },
                    {
                        "$project" : { "avg" : "$avgAge" }
                    }
                ],
                "states" : [
                    {
                        "$match" : { "$expr" : { "$ne" : [ "$_id", "تهران" ] }}
                    },
                    {
                        "$project" : { "_id" : 1.0, "avgAge" : 1.0 }
                    }
                ]
            }
        },
         "$facet" : {
             "less" : [
                 {
                     "$match" : { "$expr" : { "$lt" : [ "$states.avgAge", "$tehran.avg" ] }}
                 },
                 {
                     "$group" : { "_id" : "$states", "states" : { "$addToSet" : "$states._id" }}
                 }
             ],
             "eql" : [
                 {
                     "$match" : { "$expr" : { "$eq" : [ "$states.avgAge", "$tehran.avg" ]}}
                 },
                 {
                     "$group" : { "_id" : "$states", "states" : { "$addToSet" : "$states._id" }}
                 }
             ],
             "grt" : [
                 {
                     "$match" : { "$expr" : { "$gt" : [ "$states.avgAge", "$tehran.avg" ] }}
                 },
                 {
                     "$group" : { "_id" : "$states",  "states" : {  "$addToSet" : "$states._id"}}
                 }
             ]
         }
        },
        {
            "$project" : {
                "less_cities" : { "$arrayElemAt" :[{ "$arrayElemAt":[ "$less.states",0.0 ]}, 0.0]},
                "equal_cities" : { "$arrayElemAt" :[{ "$arrayElemAt":[ "$eql.states",0.0 ]}, 0.0]},
                "greater_cities" : { "$arrayElemAt" :[{ "$arrayElemAt":["$grt.states", 0.0]},0.0]}
            }
        }
    ]
)
```

خروجی:

Tree

Key	Value
(1) {_id : }	{ 3 fields }
less_cities	null
equal_cities	null
greater_cities	[30 elements]
0	سمنان
1	البرز
2	قزوین
3	فارس
4	زنجان
5	کرمان
6	خوزستان
7	بوشهر
8	آذربایجان شرقی
9	همدان
10	کردستان
11	گیلان
12	مرکزی
13	خراسان جنوبی
14	چهارمحال و بختیاری
15	مازندران
16	هرمزگان
17	ایلام
18	کرمانشاه
19	گلستان
20	یزد
21	کهگیلویه و بویراحمد
22	اصفهان
23	لرستان
24	خراسان شمالی
25	خراسان رضوی
26	اردبیل
27	سیستان و بلوچستان
28	آذربایجان غربی
29	قم

JSON

```
{
    "less_cities" : null,
    "equal_cities" : null,
    "greater_cities" : [
        "سمنان",
        "البرز",
        "قزوین",
        "فارس",
        "زنجان",
        "کرمان",
        "خوزستان",
        "بوشهر",
        "آذربایجان شرقی",
        "همدان",
        "کردستان",
        "گیلان",
        "مرکزی",
        "خراسان جنوبی",
        "چهارمحال و بختیاری",
        "مازندران",
        "هرمزگان",
        "ایلام",
        "کرمانشاه",
        "گلستان",
        "یزد",
        "کهگیلویه و بویراحمد",
        "اصفهان",
        "لرستان",
        "خراسان شمالی",
        "خراسان رضوی",
        "اردبیل",
        "سیستان و بلوچستان",
        "آذربایجان غربی",
        "قم"
    ]
}
```

همانطورکه مشاهده می‌شود، میانگین سن کاربران استان تهران از میانگین سن کاربران تمام استان‌های دیگر کمتر است.

زمان اجرا: 0.317s

5. کدام شهر، بیشترین کاربر و کدام شهر، کمترین کاربر را دارد؟

کوئری:

```
db.getCollection("users").aggregate(
    [
        {
            "$group" : {
                "_id" : "$location.city",
                "count" : { "$sum" : 1.0 }
            }
        },
        {
            "$facet" : {
                "max_users" : [
                    {
                        "$sort" : { "count" : -1.0 }
                    },
                    {
                        "$limit" : 1.0
                    }
                ],
                "min_users" : [
                    {
                        "$sort" : { "count" : 1.0 }
                    },
                    {
                        "$limit" : 1.0
                    }
                ]
            }
        },
        {
            "$project" : {
                "city_with_max_users" : {
                    "$arrayElemAt" : [ "$max_users._id",  0.0 ]
                },
                "city_with_min_users" : {
                    "$arrayElemAt" : [ "$min_users._id",  0.0 ]
                }
            }
        }
    ]
)
```

در اینجا، ابتدا تعداد کاربران هر استان را محاسبه کرده، سپس یک بار، استان‌ها را بر اساس تعداد کاربرانشان به‌صورت نزولی مرتب کرده و اولین استان را برمیداریم، و یک بار، استان‌ها را بر اساس تعداد کاربرانشان به‌صورت صعودی مرتب کرده و اولین استان را برمیداریم. سپس این دو استان را به‌صورت مناسب در خروجی نمایش می‌دهیم.

خروجی:

Tree

Key	Value
(1) {_id : }	{ 2 fields }
city_with_max_users	آمل
city_with_min_users	نجف‌آباد

JSON

```
{
    "city_with_max_users" : "آمل",
    "city_with_min_users" : "نجف‌آباد"
}
```

زمان اجرا: 0.250s

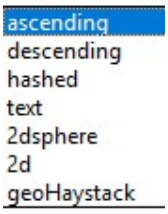
گام چهارم تمرین – بررسی کارآیی شاخص‌ها

به منظور ایجاد شاخص، مراحل زیر در ابزار Studio 3T طی می‌شود:

ابتدا بر روی کالکشن users کلیک راست نموده، سپس گزینه‌ی Add Index را انتخاب می‌کنیم.

حال از طریق گزینه‌ی Add Field(s)، فیلد(های) مورد نظر را انتخاب می‌نماییم.

پس از انتخاب فیلد(های) مورد نظر، Index Type آنها را از بین یکی از موارد زیر انتخاب می‌کنیم:



استفاده از شاخص در گام دوم – سوال 1 :

اجرای کوئری با شاخص	اجرای کوئری بدون شاخص	
0.014s	0.191s	Indexed Fields: Field: dob.age — Index Type: descending Field: location.city — Index Type: descending

استفاده از شاخص در گام دوم – سوال 4:

اجرای کوئری با شاخص	اجرای کوئری بدون شاخص	
2.853s	3.041s	Indexed Fields: Field: dob.date — Index Type: ascending

استفاده از شاخص در گام دوم – سوال 5:

اجرای کوئری با شاخص	اجرای کوئری بدون شاخص	
1 دقیقه و 59 ثانیه و 8صدم ثانیه	2 دقیقه و 2 ثانیه و 50صدم ثانیه	Indexed Fields: Field: login.password — Index Type: ascending

استفاده از شاخص در گام سوم – سوال 2 :

اجرای کوئری با شاخص	اجرای کوئری بدون شاخص	Indexed Fields:
0.242s	0.247s	Field: location.state — Index Type: ascending

استفاده از شاخص در گام سوم – سوال 5 :

اجرای کوئری با شاخص	اجرای کوئری بدون شاخص	Indexed Fields:
0.272s	0.250s	Field: location.city — Index Type: ascending

همانطور که از جداول بالا مشخص شد، استفاده از شاخص در مسئله‌ی گام دوم – سوال 1 در کاهش زمان اجـرا بسیار تاثیرگذار بوده است (که دلیل آن این است که فقط از find استفاده شده است). ولی اسـتفاده از شـاخص در سایر موارد، تاثیر چندانی در کاهش زمان اجرا نداشته است، که احتمالا دلیل آن به نوع کوئری نوشـته شـده برمی‌گیرد.

گام پنجم تمرین – Map Reduce

اسکریپت زیر، قابلیت بازیابی میانگین سن کاربران به تفکیک هر شهر را دارد. این اسکریپت با توجه به اینکه به زبان جاوا اسکریپت نوشته شده است، در نتیجه می‌تواند در خط فرمان مانگودی‌بی نیز اجرا شود:

```
// *** 3T Software Labs, Studio 3T: MapReduce Job ****

// Variable for db
var __3tsoftwarelabs_db = "metadata";

// Variable for map
var __3tsoftwarelabs_map = function () {
    emit(this.location.city,this.dob.age);
}
;

// Variable for reduce
var __3tsoftwarelabs_reduce = function (key, values) {
    return (key, Array.avg(values));
}
;

db.runCommand({
    mapReduce: "users",
    map: __3tsoftwarelabs_map,
    reduce: __3tsoftwarelabs_reduce,
    out: {"inline": 1},
    query: {},
    sort: {},
    inputDB: "metadata",
 });
```

توضیح: خروجی تابع map، لیستی از زوج مرتب‌های (مقدار, کلید) به‌صورت (سن, شهر) است. سپس تابع reduce، به ازای هر شهر، میانگین سن کاربران آنها را محاسبه و برمی‌گرداند.

با اجرای این اسکریپت در خط فرمان مانگودی‌بی، خروجی آن به‌صورت زیر خواهد بود:

Tree

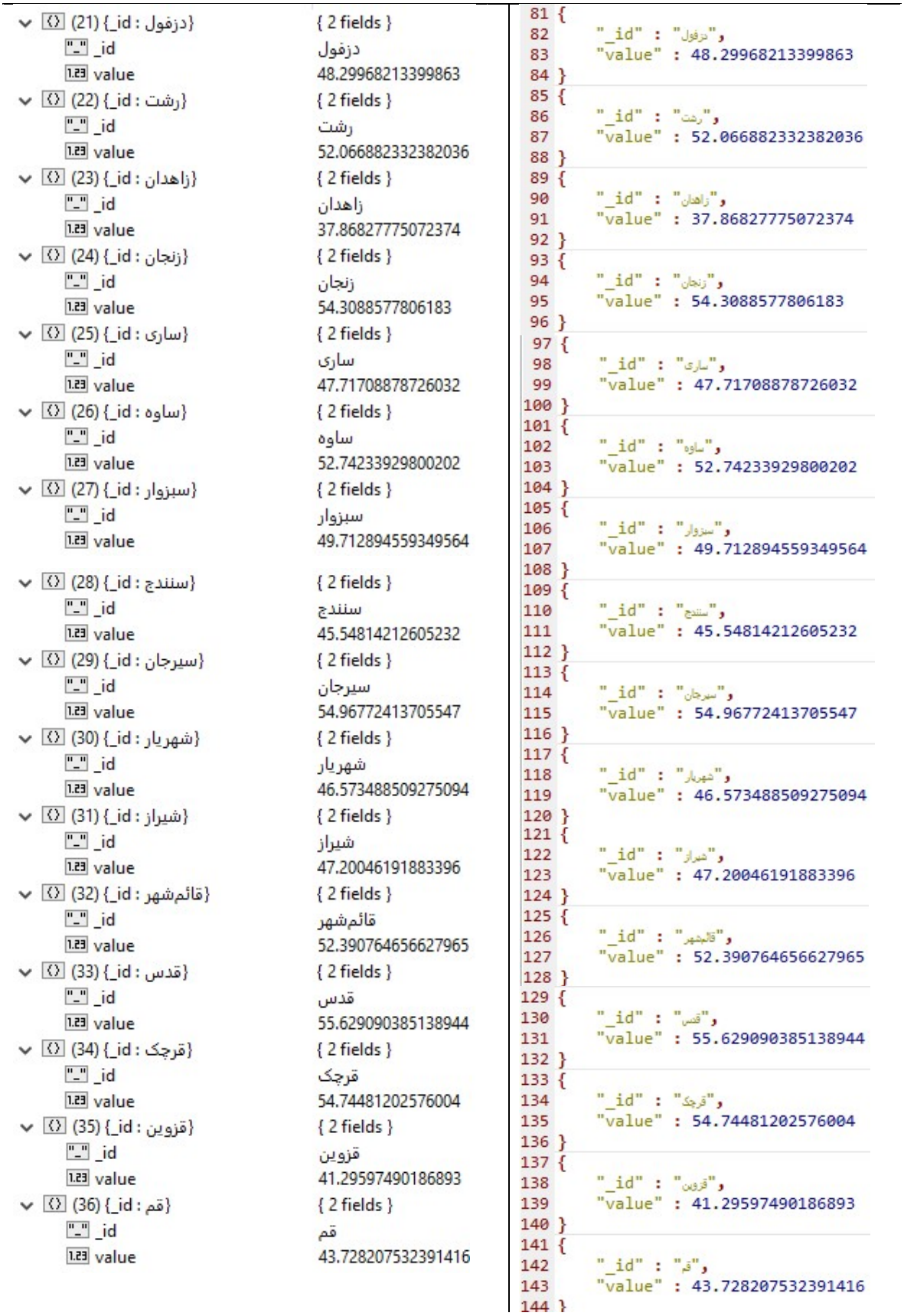
Key	Value
(1) {_id : آبادان}	{ 2 fields }
_id	آبادان
value	46.30684734774464
(2) {_id : آمل}	{ 2 fields }
_id	آمل
value	62.7272287416071
(3) {_id : اراک}	{ 2 fields }
_id	اراک
value	53.29119558235929
(4) {_id : اردبیل}	{ 2 fields }
_id	اردبیل
value	43.21822560199644
(5) {_id : ارومیه}	{ 2 fields }
_id	ارومیه
value	47.02806895439129

JSON

```
{
    "_id" : "آبادان",
    "value" : 46.30684734774464
}
{
    "_id" : "آمل",
    "value" : 62.7272287416071
}
{
    "_id" : "اراک",
    "value" : 53.29119558235929
}
{
    "_id" : "اردبیل",
    "value" : 43.21822560199644
}
```

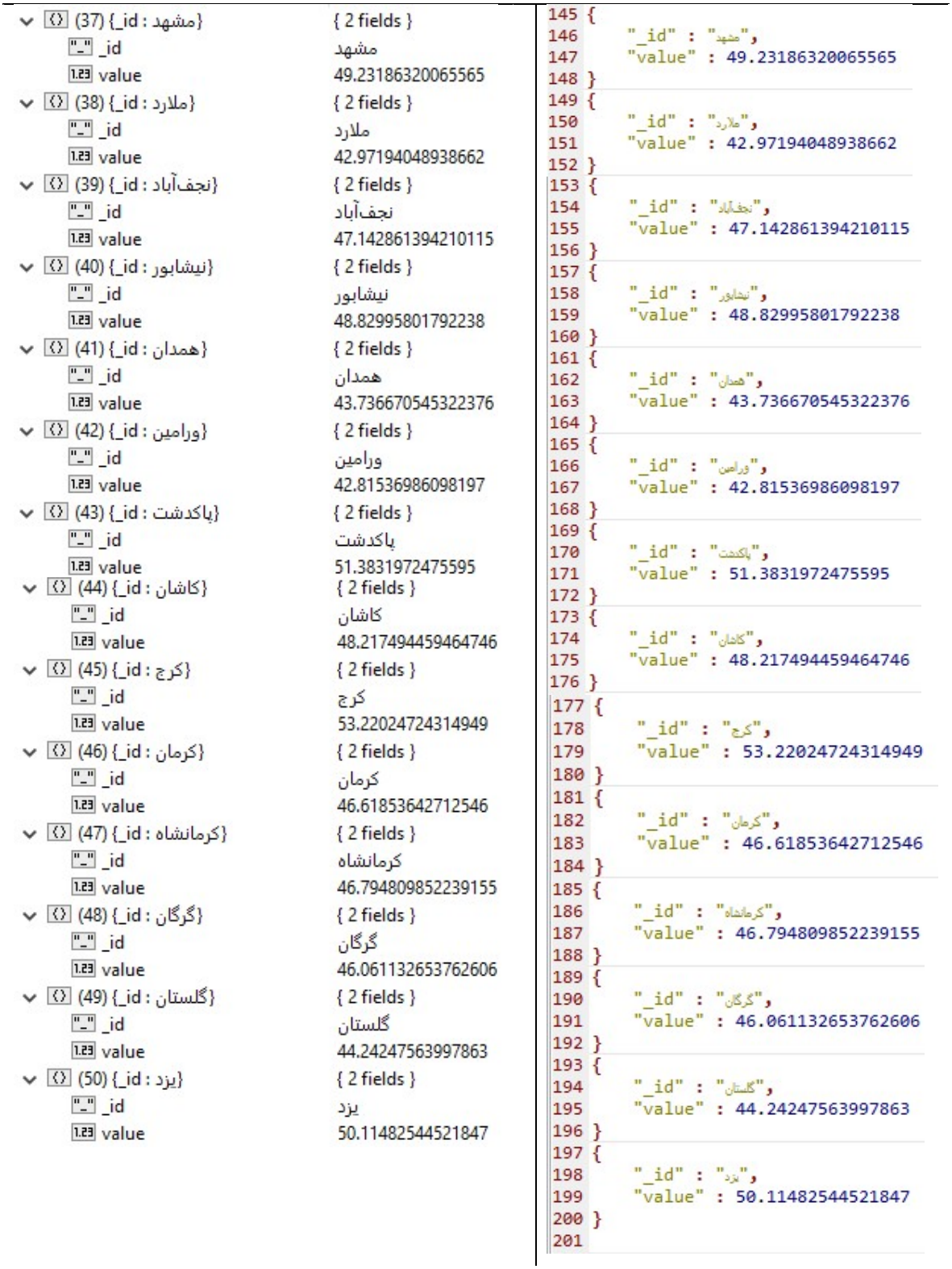
(6) {_id : اسلام‌شهر}	{ 2 fields }
_id	اسلام‌شهر
value	46.04880682299492
(7) {_id : اصفهان}	{ 2 fields }
_id	اصفهان
value	40.62441790895408
(8) {_id : اهواز}	{ 2 fields }
_id	اهواز
value	44.31602943268522
(9) {_id : ایلام}	{ 2 fields }
_id	ایلام
value	43.454963331113426
(10) {_id : بابل}	{ 2 fields }
_id	بابل
value	48.43959303896065
(11) {_id : بجنورد}	{ 2 fields }
_id	بجنورد
value	39.1487795341395
(12) {_id : بروجرد}	{ 2 fields }
_id	بروجرد
value	38.23225981929865
(13) {_id : بندرعباس}	{ 2 fields }
_id	بندرعباس
value	48.83462002017418
(14) {_id : بوشهر}	{ 2 fields }
_id	بوشهر
value	45.187953695153055
(15) {_id : بیرجند}	{ 2 fields }
_id	بیرجند
value	51.973124742873466
(16) {_id : تبریز}	{ 2 fields }
_id	تبریز
value	61.177931171109336
(17) {_id : تهران}	{ 2 fields }
_id	تهران
value	49.52043100284166
(18) {_id : خرم‌آباد}	{ 2 fields }
_id	خرم‌آباد
value	48.11363125019587
(19) {_id : خمینی‌شهر}	{ 2 fields }
_id	خمینی‌شهر
value	48.07379103960122
(20) {_id : خوی}	{ 2 fields }
_id	خوی
value	38.81108519607666

```
{
    "_id" : "ارومیه",
    "value" : 47.02806895439129
}
{
    "_id" : "اسلامشهر",
    "value" : 46.04880682299492
}
{
    "_id" : "اصفهان",
    "value" : 40.62441790895408
}
{
    "_id" : "اهواز",
    "value" : 44.31602943268522
}
{
    "_id" : "ایلام",
    "value" : 43.454963331113426
}
{
    "_id" : "بابل",
    "value" : 48.43959303896065
}
{
    "_id" : "بجنورد",
    "value" : 39.1487795341395
}
{
    "_id" : "بروجرد",
    "value" : 38.23225981929865
}
{
    "_id" : "بندرعباس",
    "value" : 48.83462002017418
}
{
    "_id" : "بوشهر",
    "value" : 45.187953695153055
}
{
    "_id" : "بیرجند",
    "value" : 51.973124742873466
}
{
    "_id" : "تبریز",
    "value" : 61.177931171109336
}
{
    "_id" : "تهران",
    "value" : 49.52043100284166
}
{
    "_id" : "خرم‌آباد",
    "value" : 48.11363125019587
}
{
    "_id" : "خمینی‌شهر",
    "value" : 48.07379103960122
}
{
    "_id" : "خوی",
    "value" : 38.81108519607666
}
```



(21) {_id : دزفول}	{ 2 fields }
_id	دزفول
value	48.29968213399863
(22) {_id : رشت}	{ 2 fields }
_id	رشت
value	52.066882332382036
(23) {_id : زاهدان}	{ 2 fields }
_id	زاهدان
value	37.86827775072374
(24) {_id : زنجان}	{ 2 fields }
_id	زنجان
value	54.3088577806183
(25) {_id : ساری}	{ 2 fields }
_id	ساری
value	47.71708878726032
(26) {_id : ساوه}	{ 2 fields }
_id	ساوه
value	52.74233929800202
(27) {_id : سبزوار}	{ 2 fields }
_id	سبزوار
value	49.712894559349564
(28) {_id : سنندج}	{ 2 fields }
_id	سنندج
value	45.54814212605232
(29) {_id : سیرجان}	{ 2 fields }
_id	سیرجان
value	54.96772413705547
(30) {_id : شهریار}	{ 2 fields }
_id	شهریار
value	46.573488509275094
(31) {_id : شیراز}	{ 2 fields }
_id	شیراز
value	47.20046191883396
(32) {_id : قائم‌شهر}	{ 2 fields }
_id	قائم‌شهر
value	52.390764656627965
(33) {_id : قدس}	{ 2 fields }
_id	قدس
value	55.629090385138944
(34) {_id : قرچک}	{ 2 fields }
_id	قرچک
value	54.74481202576004
(35) {_id : قزوین}	{ 2 fields }
_id	قزوین
value	41.29597490186893
(36) {_id : قم}	{ 2 fields }
_id	قم
value	43.728207532391416


```
{
    "_id" : "دزفول",
    "value" : 48.29968213399863
}
{
    "_id" : "رشت",
    "value" : 52.066882332382036
}
{
    "_id" : "زاهدان",
    "value" : 37.86827775072374
}
{
    "_id" : "زنجان",
    "value" : 54.3088577806183
}
{
    "_id" : "ساری",
    "value" : 47.71708878726032
}
{
    "_id" : "ساوه",
    "value" : 52.74233929800202
}
{
    "_id" : "سبزوار",
    "value" : 49.712894559349564
}
{
    "_id" : "سنندج",
    "value" : 45.54814212605232
}
{
    "_id" : "سیرجان",
    "value" : 54.96772413705547
}
{
    "_id" : "شهریار",
    "value" : 46.573488509275094
}
{
    "_id" : "شیراز",
    "value" : 47.20046191883396
}
{
    "_id" : "قائم‌شهر",
    "value" : 52.390764656627965
}
{
    "_id" : "قدس",
    "value" : 55.629090385138944
}
{
    "_id" : "قرچک",
    "value" : 54.74481202576004
}
{
    "_id" : "قزوین",
    "value" : 41.29597490186893
}
{
    "_id" : "قم",
    "value" : 43.728207532391416
}
```



(37) {_id : مشهد}	{ 2 fields }
_id	مشهد
value	49.23186320065565
(38) {_id : ملارد}	{ 2 fields }
_id	ملارد
value	42.97194048938662
(39) {_id : نجف‌آباد}	{ 2 fields }
_id	نجف‌آباد
value	47.142861394210115
(40) {_id : نیشابور}	{ 2 fields }
_id	نیشابور
value	48.82995801792238
(41) {_id : همدان}	{ 2 fields }
_id	همدان
value	43.736670545322376
(42) {_id : ورامین}	{ 2 fields }
_id	ورامین
value	42.81536986098197
(43) {_id : پاکدشت}	{ 2 fields }
_id	پاکدشت
value	51.3831972475595
(44) {_id : کاشان}	{ 2 fields }
_id	کاشان
value	48.217494459464746
(45) {_id : کرج}	{ 2 fields }
_id	کرج
value	53.22024724314949
(46) {_id : کرمان}	{ 2 fields }
_id	کرمان
value	46.61853642712546
(47) {_id : کرمانشاه}	{ 2 fields }
_id	کرمانشاه
value	46.794809852239155
(48) {_id : گرگان}	{ 2 fields }
_id	گرگان
value	46.061132653762606
(49) {_id : گلستان}	{ 2 fields }
_id	گلستان
value	44.24247563997863
(50) {_id : یزد}	{ 2 fields }
_id	یزد
value	50.11482544521847

```
{
    "_id" : "مشهد",
    "value" : 49.23186320065565
}
{
    "_id" : "ملارد",
    "value" : 42.97194048938662
}
{
    "_id" : "نجف‌آباد",
    "value" : 47.142861394210115
}
{
    "_id" : "نیشابور",
    "value" : 48.82995801792238
}
{
    "_id" : "همدان",
    "value" : 43.736670545322376
}
{
    "_id" : "ورامین",
    "value" : 42.81536986098197
}
{
    "_id" : "پاکدشت",
    "value" : 51.3831972475595
}
{
    "_id" : "کاشان",
    "value" : 48.217494459464746
}
{
    "_id" : "کرج",
    "value" : 53.22024724314949
}
{
    "_id" : "کرمان",
    "value" : 46.61853642712546
}
{
    "_id" : "کرمانشاه",
    "value" : 46.794809852239155
}
{
    "_id" : "گرگان",
    "value" : 46.061132653762606
}
{
    "_id" : "گلستان",
    "value" : 44.24247563997863
}
{
    "_id" : "یزد",
    "value" : 50.11482544521847
}
```

زمان اجرا: 3.278s

بخش دوم: مدل‌سازی داده‌ها با گراف / کار با دیتابیس Neo4j

در اینجا ابتدا خلاصه‌ای از مراحل انجام شده برای پاسخ به سوالات داده شده ارائه می‌شود:

1. ابتدا نسخه‌ی 1M دیتاست MovieLens را از آدرس زیر دانلود می‌کنیم:

http://files.grouplens.org/datasets/movielens/ml-1m.zip

این دیتاست شامل 3 فایل movies.dat ،users.dat ،ratings.dat می‌باشد:

- فایل movies.dat شامل اطلاعات مربوط به 3883 فیلم با فرمت:

MovieID::Title (Year)::Genres

و به‌صورت زیر می‌باشد:

```
1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action|Adventure|Thriller
11::American President, The (1995)::Comedy|Drama|Romance
```

- فایل users.dat شامل اطلاعات مربوط به 6040 کاربر با فرمت:

UserID::Gender::Age::Occupation::Zip-code

و به‌صورت زیر می‌باشد:

```
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
9::M::25::17::61614
10::F::35::1::95370
11::F::25::1::04093
```

- فایل ratings.dat شامل اطلاعات مربوط به 1000209 مورد Rating با فرمت:

UserID::MovieID::Rating::Timestamp

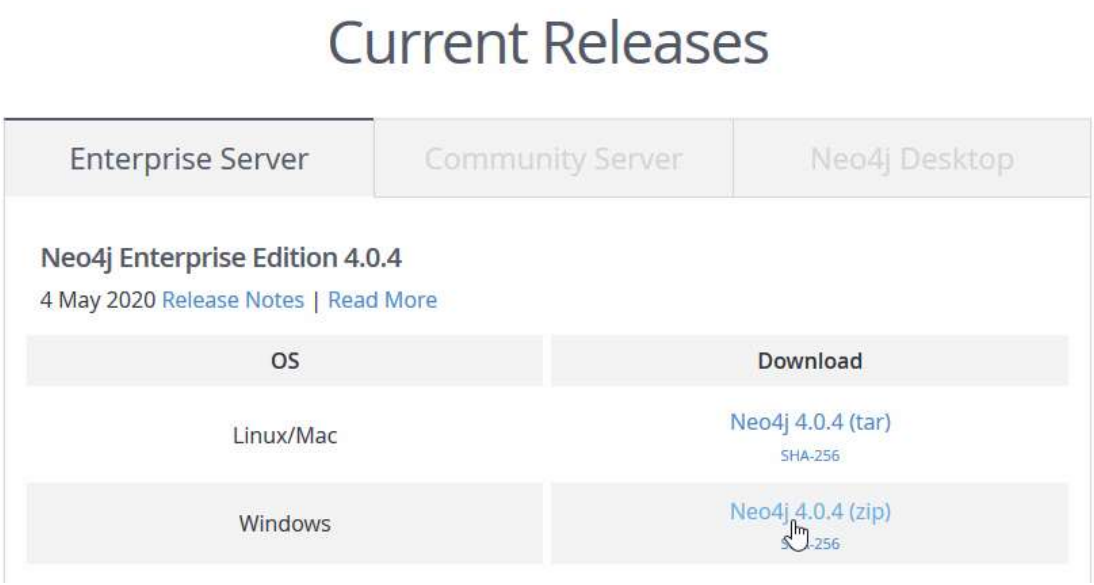
و به‌صورت زیر می‌باشد:

```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
1::595::5::978824268
```

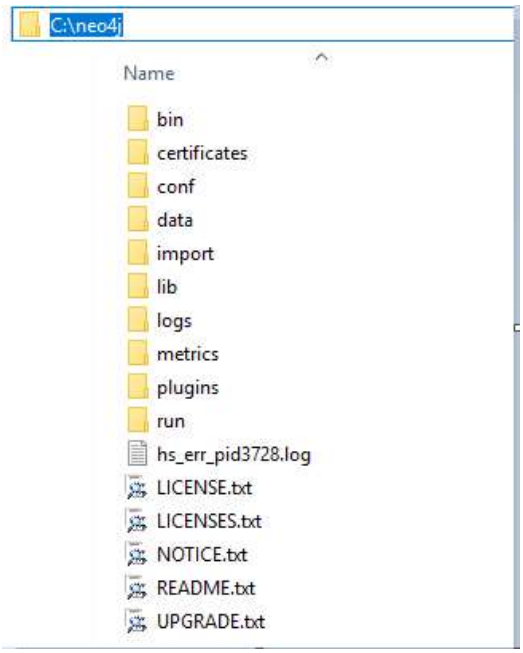
---

2. سپس نسخه‌ی Enterprise server مربوط به سیستم‌عامل ویندوز دیتاست Neo4j را از آدرس زیر دانلود می‌کنیم:

https://neo4j.com/download-center/



و سپس تمام موارد داخل این فایل فشرده را در آدرس **C:\neo4j** اکسترکت می‌نماییم. محتوای این آدرس به‌صورت زیر می‌باشد:



سپس آدرس **C:\neo4j\bin** را به Pathهای سیستم اضافه می‌کنیم.

و در نهایت، دستور neo4j install-service را در cmd اجرا می‌کنیم تا از این به‌بعد بتوانیم Neo4j را به‌عنوان یک windows service داشته باشیم. سپس با دستور neo4j start، این سرویس را اجرا می‌کنیم:

```
C:\Users\Peyman>neo4j start
Neo4j service started
```

---

3. سپس دیتاست دانلود شده را از طریق کتابخانه‌ی py2neo نسخه‌ی 4.0.0b1 و به‌صورت زیر در Neo4j درج می‌کنیم:

لازم به ذکر است که با توجه به اینکه دیتاست 1M با دیتاست 100K متفاوت است، به‌همین دلیل مجبور به تغییر بخش‌هایی از کد ارائه شده در آدرس زیر شدیم:

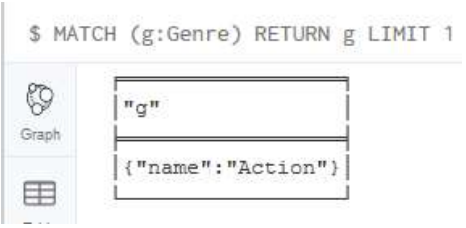
https://github.com/tkcsdvd/neo4j-movielens/blob/master/ingestion/ingestion.py

```
import pandas as pd
from py2neo import Graph, Node

USERNAME = "neo4j"
PASS = "[redacted]"

graph = Graph("bolt://localhost:7687", auth = (USERNAME, PASS))
graph.delete_all()

def main():
    print("Create Genre Nodes")
    createGenreNodes()

    print("Step 1 out of 3: loading movie nodes")
    loadMovies()

    print("Step 2 out of 3: loading user nodes")
    loadUsers()

    print("Step 3 out of 3: loading rating relationships")
    loadRatings()
#-----------------------------------------------
def createGenreNodes():
    allGenres = ["Action", "Adventure", "Animation", "Children's", "Comedy", "Crime",
                 "Documentary", "Drama", "Fantasy", "Film-Noir", "Horror", "Musical",
                 "Mystery", "Romance", "Sci-Fi", "Thriller", "War", "Western"]

    for genre in allGenres:
        gen = Node("Genre", name=genre)
        graph.create(gen)

#-----------------------------------------------
def loadMovies():
    readCSV = pd.read_csv('data/movies.dat', sep="::", engine="python", header=None)
    for row in readCSV.itertuples():
        i = row.Index
        createMovieNodes(row)
        createGenreMovieRelationships(row)
        if i%100==0:
            print(f"{i} Movie nodes created")

#-----------------------------------------------
def createMovieNodes(row):
    movieId = row._1
    title = row._2[:-7]
    year = row._2[-5:-1]
    mov = Node("Movie", id=movieId, title=title, year=year)
    graph.create(mov)
```

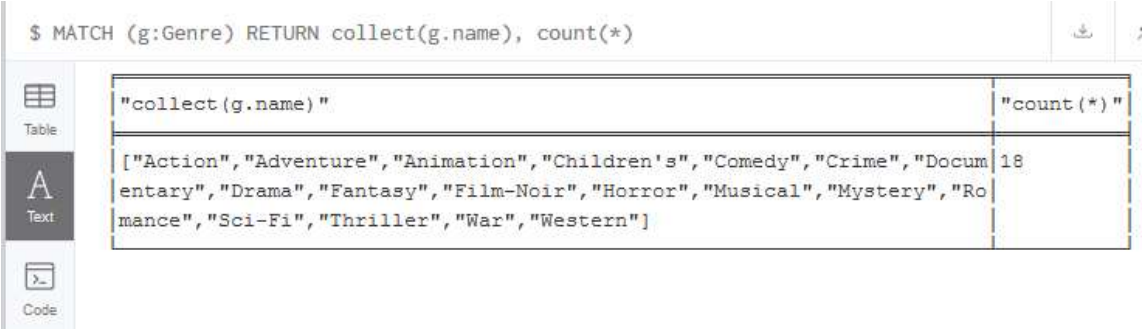
```
#-----------------------------------------------
def createGenreMovieRelationships(row):
    movieId = row._1
    movieGenres = row._3.split("|")

    for movieGenre in movieGenres:
        graph.run('MATCH (g:Genre {name: {genre}}), (m:Movie {id: {movieId}}) CREATE (g)-[:IS_GENRE_OF]->(m)',
                  genre=movieGenre, movieId=movieId)

#-----------------------------------------------
def loadUsers():
    readCSV = pd.read_csv('data/users.dat', sep="::", engine="python", header=None)
    for row in readCSV.itertuples():
        i = row.Index
        createUserNodes(row)
        if i % 100 == 0:
            print(f"{i} user nodes created")

# -----------------------------------------------
def createUserNodes(row):
    userId = "User "+str(row._1)
    gender = row._2
    age = row._3
    occupation = row._4
    zipcode = row._5
    usr = Node("User", id=userId, gender=gender, age=age, occupation=occupation, zipcode=zipcode)
    graph.create(usr)

# -----------------------------------------------
def loadRatings():
    readCSV = pd.read_csv('data/ratings.dat', sep="::", engine="python", header=None)
    for row in readCSV.itertuples():
        i = row.Index
        createRatingRelationship(row)
        if i%100==0:
            print(f"{i} Rating relationships created")

# -----------------------------------------------
def createRatingRelationship(row):
    ratingData = parseRowRatingRelationships(row)
    graph.run(
        'MATCH (u:User {id: {userId}}), (m:Movie {id: {movieId}}) CREATE (u)-[:RATED { rating: {rating}, timestamp: {timestamp} }]->(m)
        userId=ratingData[0], movieId=ratingData[1], rating=ratingData[2], timestamp=ratingData[3])

# -----------------------------------------------
def parseRowRatingRelationships(row):
    userId = "User "+str(row._1)
    movieId = row._2
    rating = float(row._3)
    timestamp = row._4
    return (userId, movieId, rating, timestamp)

# -----------------------------------------------

if __name__ == '__main__':
    main()
```

4. آدرس http://localhost:7474/browser/ را از طریق مرورگر خود بازکرده و کوئری‌های زیر را به‌منظور آشنایی با زبان Cypher و همچنین بررسی صحت داده‌های درج شده، اجرا می‌کنیم:

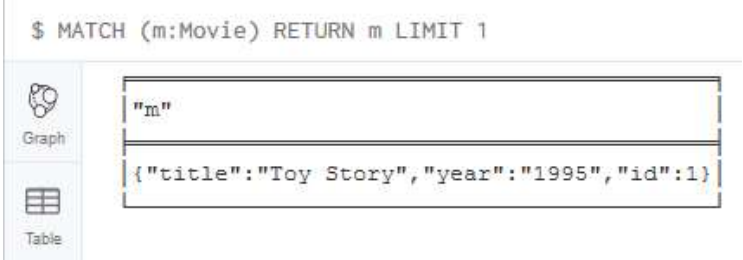
- اطلاعات مربوط به یک نود ژانر:



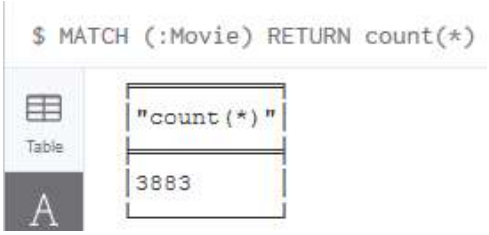
- لیست ژانرها و تعداد آنها:



- اطلاعات مربوط به یک نود فیلم:



- تعداد فیلم‌ها:



- اطلاعات مربوط به یک نود کاربر:

```
$ MATCH (u:User) RETURN u LIMIT 1
```

Graph
Table

"u"
{"zipcode":"70072","occupation":16,"id":"User 2","gender":"M","age":56 }

- تعداد کاربران:

```
$ MATCH (u:User) RETURN count(*)
```

Table
A

"count(*)"
6040

- اطلاعات مربوط به یک یال RATED:

```
$ MATCH (:User)-[r:RATED]-(:Movie) RETURN r LIMIT 1
```

Table
A
Text

"r"
{"rating":3.0,"timestamp":965328072}

- تعداد Ratingها:

```
$ MATCH (:User)-[r:RATED]-(:Movie) RETURN count(*)
```

Table
A

"count(*)"
1000209

سوالات و پرس‌وجوهای مورد نیاز:

1. چه ژانرهایی در این دیتاست وجود دارد:

Query

```
MATCH (g:Genre)
WITH collect(g.name) as genres
RETURN genres
```

Output

Table

genres
["Action", "Adventure", "Animation", "Children's", "Comedy", "Crime", "Documentary", "Drama", "Fantasy", "Film-Noir", "Horror", "Musical" "Mystery", "Romance", "Sci-Fi", "Thriller", "War", "Western"]

Run time

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

2. تعداد کل فیلم‌ها چقدر است:

Query

```
MATCH (:Movie)
WITH count(*) as count
RETURN count
```

Output

Table

count
3883

Run time

Started streaming 1 records in less than 1 ms and completed in less than 1 ms.

3. تعداد امتیازات داده شده به فیلم Silence of the Lambs در هر رده امتیازی چقدر است:

Query

```
MATCH (:User)-[r:RATED]->(m:Movie)
WHERE m.title="Silence of the Lambs, The"
WITH r.rating as rating, count(*) as count
RETURN rating, count
ORDER BY rating
```

Output

Table | Text | Code

rating	count
1.0	37
2.0	43
3.0	246
4.0	902
5.0	1350

Run time

Started streaming 5 records after 20 ms and completed after 20 ms.

4. تعداد امتیازات هر فیلم به‌صورت نزولی:

Query

```
MATCH (:User)-[r:RATED]->(m:Movie)
WITH m.id as id, m.title as title, count(*) as count
RETURN title, count
ORDER BY count DESC
```

Output

Table | Text | Code

title	count
"American Beauty"	3428
"Star Wars: Episode IV - A New Hope"	2991
"Star Wars: Episode V - The Empire Strikes Back"	2990
"Star Wars: Episode VI - Return of the Jedi"	2883
"Jurassic Park"	2672
"Saving Private Ryan"	2653
"Terminator 2: Judgment Day"	2649
"Matrix, The"	2590
"Back to the Future"	2583
"Silence of the Lambs, The"	2578
"Men in Black"	2538

* خروجی کامل این سوال در فولدر Part 2- Neo4j از فولدر Code قرار گرفته است.

Run time

Started streaming 3706 records after 3356 ms and completed after 3368 ms, displaying first 1000 rows.

5. کدام ژانر بیشترین فیلم تولید شده را به خود اختصاص داده است:

Query

```
MATCH (g:Genre)-[:IS_GENRE_OF]->(m:Movie)
WITH g.name as genre, count(*) as count
ORDER BY count DESC
LIMIT 1
RETURN genre
```

Output

Run time

Started streaming 1 records after 11 ms and completed after 12 ms.

کدام ژانر بیشترین امتیاز کاربران را به‌دست آورده است؟ (بر اساس میانگین امتیاز برای هر ژانر)

Query

```
MATCH (g:Genre)-[:IS_GENRE_OF]->(:Movie)<-[r:RATED]-(:User)
WITH g.name as genre, avg(r.rating) as avgrating
ORDER BY avgrating DESC
LIMIT 1
RETURN genre
```

Output

Run time

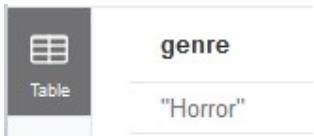
Started streaming 1 records after 6161 ms and completed after 6161 ms.

کدام ژانر کمترین امتیاز کاربران را به‌دست آورده است؟ (بر اساس میانگین امتیاز برای هر ژانر)

Query

```
MATCH (g:Genre)-[:IS_GENRE_OF]->(:Movie)<-[r:RATED]-(:User)
WITH g.name as genre, avg(r.rating) as avgrating
ORDER BY avgrating
LIMIT 1
RETURN genre
```

Output



Run time

Started streaming 1 records after 6084 ms and completed after 6084 ms.

لازم به ذکر است که در قالب یک کوئری نیز می‌توان به سه سوال ذکر شده پاسخ داد:

Query

```
MATCH (g:Genre)-[:IS_GENRE_OF]->(:Movie)
WITH g.name as genre, count(*) as count, "max #movies" as QType
ORDER BY count DESC
LIMIT 1
RETURN QType,genre
UNION ALL MATCH (g:Genre)-[:IS_GENRE_OF]->(:Movie)<-[r:RATED]-(:User)
WITH g.name as genre, avg(r.rating) as avgrating, "max avg rating" as QType
ORDER BY avgrating DESC
LIMIT 1
RETURN QType, genre
UNION ALL MATCH (g:Genre)-[:IS_GENRE_OF]->(:Movie)<-[r:RATED]-(:User)
WITH g.name as genre, avg(r.rating) as avgrating, "min avg rating" as QType
ORDER BY avgrating
LIMIT 1
RETURN QType, genre
```

Output

Table | Text

QType	genre
"max #movies"	"Drama"
"max avg rating"	"Film-Noir"
"min avg rating"	"Horror"

Run time

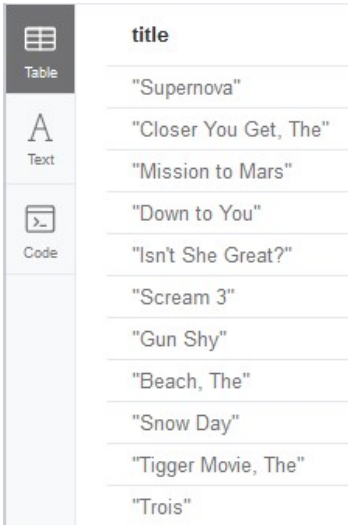
Started streaming 3 records after 18910 ms and completed after 18910 ms.

6. عنوان فیلم‌های تولید شده در سال 2000 را بیابید:

Query

```
MATCH (m:Movie)
WHERE m.year="2000"
WITH m.title as title
RETURN title
```

Output



title
"Supernova"
"Closer You Get, The"
"Mission to Mars"
"Down to You"
"Isn't She Great?"
"Scream 3"
"Gun Shy"
"Beach, The"
"Snow Day"
"Tigger Movie, The"
"Trois"

* خروجی کامل این سوال در فولدر Part 2- Neo4j از فولدر Code قرار گرفته است.

Run time

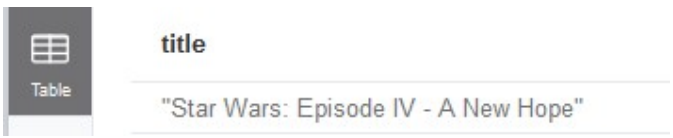
Started streaming 156 records in less than 1 ms and completed after 5 ms.

7. با توجه به فیلد occupation در فایل users.dat، برنامه‌نویسان به کدام فیلم بیشترین امتیاز را داده‌اند؟ (کدام فیلم تعداد امتیازی 5 بیشتری دریافت کرده است)

Query

```
MATCH (u:User)-[r:RATED]->(m:Movie)
WHERE u.occupation=12 and r.rating=5
WITH m.id as id, m.title as title, count(*) as count
ORDER BY count DESC
LIMIT 1
RETURN title
```

Output



title
"Star Wars: Episode IV - A New Hope"

Run time

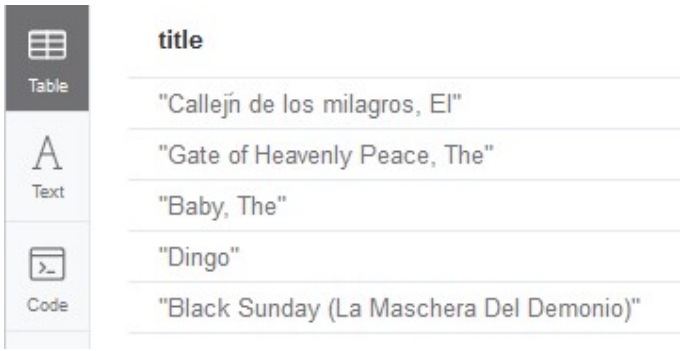
Started streaming 1 records after 348 ms and completed after 348 ms.

8. پنج فیلم محبوب رده‌ی سنی 18 تا 34سال را به‌دست آورید. (بر اساس میانگین امتیاز)

Query

```
MATCH (u:User)-[r:RATED]->(m:Movie)
WHERE u.age=18 or u.age=25
WITH m.id as id, m.title as title, avg(r.rating) as avgrating
ORDER BY avgrating DESC
LIMIT 5
RETURN title
```

Output



* با توجه به اینکه در اینجا تعداد زیادی فیلم، میانگین امتیاز برابر 5 دارند، به‌همین دلیل اگر بخواهیم فقط 5 مورد از آنهایی که بیشترین میانگین امتیاز (در اینجا، 5) داشته‌اند را انتخاب کنیم، آنگاه این 5 مورد می‌تواند متفاوت باشد.

Run time

Started streaming 5 records after 3125 ms and completed after 3125 ms.

9. بیست فیلم محبوب که حداقل 100 کاربر به آنها رای داده باشند را بدست آورید. (بر اساس میانگین امتیاز)

Query

```
MATCH (u:User)-[r:RATED]->(m:Movie)
WITH m.id as id,m.title as title, count(*) as count, avg(r.rating) as avgrati
WHERE count>=100
RETURN title
ORDER BY avgrating DESC
LIMIT 20
```

Output

Table | Text | Code

title
"Seven Samurai (The Magnificent Seven) (Shichinin no samurai)"
"Shawshank Redemption, The"
"Godfather, The"
"Close Shave, A"
"Usual Suspects, The"
"Schindler's List"
"Wrong Trousers, The"
"Sunset Blvd. (a.k.a. Sunset Boulevard)"
"Raiders of the Lost Ark"
"Rear Window"
"Paths of Glory"
"Star Wars: Episode IV - A New Hope"
"Third Man, The"
"Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb"
"Wallace & Gromit: The Best of Aardman Animation"
"To Kill a Mockingbird"
"Double Indemnity"
"Casablanca"
"Sixth Sense, The"
"Yojimbo"

Run time

Started streaming 20 records after 3780 ms and completed after 3780 ms.

بخش سوم: دیتابیس‌های سطرگسترده / کار با HBase

گام اول:

به‌منظور پاسخ به کوئری‌های داده شده، تنها از دو جدول به‌صورت زیر استفاده خواهیم کرد:

1. جدول ژانر:

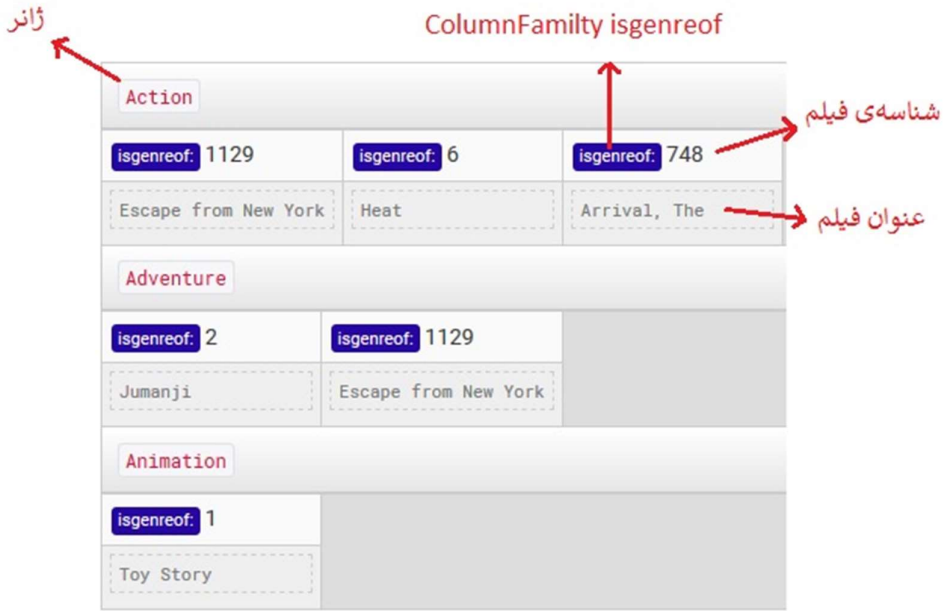
این جدول، شامل اطلاعات فیلم‌های مربوط به هر ژانر است.

در اینجا کلید سطر، نام ژانر است و یک ColumnFamily به نام isgenreof ایجاد شده است که هر ستون آن، اطلاعات یک فیلم مربوط به آن ژانر را نشان می‌دهد. این جدول به‌صورت زیر می‌باشد:

Row_id Column family

genre name	isgenreof		
	isgenreof:movieId1	isgenreof:movieId2	isgenreof:movieId3
Action	movie title 1	movie title 2	movie title 3
Adventure	movie title 4	movie title 5	-

همچنین در شکل زیر که سه سطر نمونه از این جدول را نشان می‌دهد، توضیحات تکمیلی ارائه شده است:



ما از این جدول برای پاسخ به سوال 1 استفاده می‌نماییم.

2- جدول فیلم:

این جدول، شامل اطلاعات مربوط به هرکدام از فیلم‌ها و ratingهای داده شده به آنها می‌باشد.

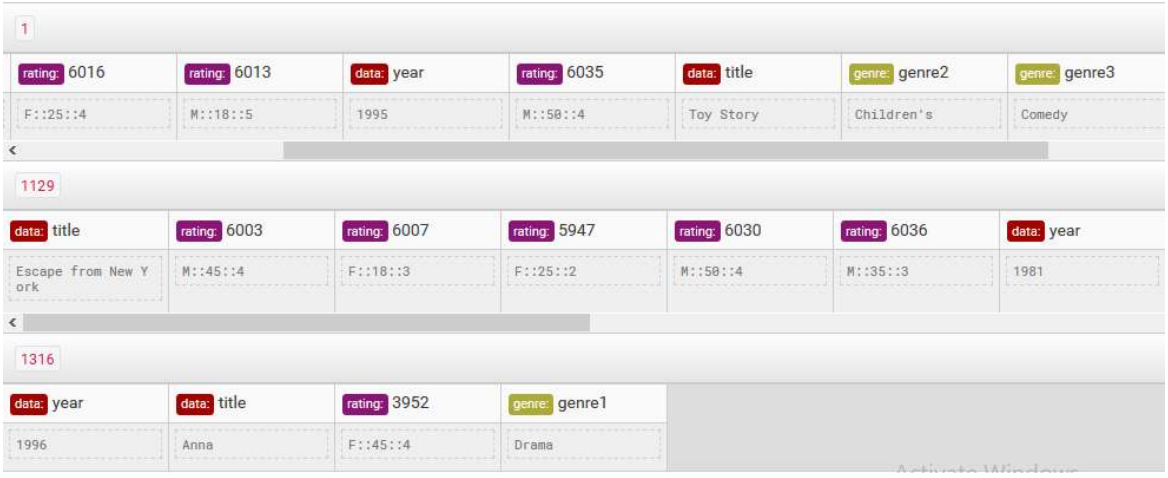
در اینجا، کلید سطر، شناسه‌ی فیلم است و دو ColumnFamily به‌شرح زیر ایجاد شده است:

1. data که اطلاعات اصلی یک فیلم را ذخیره می‌کند، یعنی شامل دو ستون data:title و data:year می‌باشد.
2. rating که اطلاعات مربوط به ratingهای داده شده به فیلم مربوطه را ذخیره می‌کند، یعنی شامل ستون‌هایی به‌صورت rating:userId می‌باشد. هر ستون rating:userId شامل مقادیر gender::age::rating_score از آن کاربر خواهد بود.

این جدول به‌صورت زیر می‌باشد:

Row_id	Column family		Column family		
movieId	data		rating		
	data::title	data:year	rating:userId	rating:userId2	…
movieId1	title name	year value	gender::age::rating_value	…	…

همچنین در شکل زیر سه سطر نمونه از این جدول را نشان می‌دهد:



ما از این جدول برای پاسخ به سوالات 2 تا 5 استفاده می‌نماییم.

گام دوم:

نمونه‌هایی از داده‌های دستی وارد شده به دو جدول ایجاد شده، در صفحات قبل ارائه شد. در اینجا بر اساس داده‌های وارد شده، به‌سوالات زیر پاسخ می‌دهیم:

1. فهرست فیلم‌های یک ژانر:

با توجه به اینکه گفته شده فهرست فیلم‌های **یک ژانر** تولید شود، به‌همین دلیل ما در اینجا به‌عنوان نمونه، فیلم‌های ژانر Thriller را تولید خواهیم کرد.

دستورات نوشته شده:

```
import happybase

connection = happybase.Connection(host='quickstart.cloudera',port=9090)
connection.open()


table = connection.table('Genre')
for key,value in table.scan(row_start='Thriller',row_stop='Thriller',columns=['isgenreof']):
    print(value.values())
```

خروجی:

```
sh-4.1# python3 code/hbase.py
dict_values([b'Heat', b'Arrival, The', b'Escape from New York'])
```

2. تعداد ژانرهای هر فیلم:

دستورات نوشته شده:

```
import happybase

connection = happybase.Connection(host='quickstart.cloudera',port=9090)
connection.open()


table = connection.table('Movie')
for key,value in table.scan(columns=['data:title','genre']):
    print(value[b'data:title'],len(value)-1)
```

خروجی:

```
sh-4.1# python3 code/hbase.py
b'Toy Story' 3
b'Escape from New York' 4
b'Anna' 1
b'Jumanji' 3
b'Heat' 3
b'Arrival, The' 3
```

---

3. کدام فیلم بیشترین تعداد امتیازدهنده را داشته است:

دستورات نوشته شده:

```
import happybase
import numpy as np

connection = happybase.Connection(host='quickstart.cloudera',port=9090)
connection.open()


table = connection.table('Movie')

title_list, ratingcount_list = list(), list()
for key,value in table.scan(columns=['data:title','rating']):
    title_list.append(value[b'data:title'])
    ratingcount_list.append(len(value)-1)

title = title_list[np.argmax(ratingcount_list)]
print(title)
```

خروجی:

```
sh-4.1# python3 code/hbase.py
b'Arrival, The'
```

---

4. سه فیلم محبوب‌تر بازه سنی 45 سال به بالا:

دستورات نوشته شده:

```
import happybase
import numpy as np

connection = happybase.Connection(host='quickstart.cloudera',port=9090)
connection.open()


table = connection.table('Movie')

def avg_per_movie(value):
    avg =  np.mean([int(user_rating[7:]) for user_rating in value.values()])
    return avg

dic = dict()
for key,value in table.scan(filter=" ValueFilter (=,'regexstring:^[MF]::45::[1-5]$') OR"
                                  " ValueFilter (=,'regexstring:^[MF]::50::[1-5]$') OR"
                                  " ValueFilter (=,'regexstring:^[MF]::56::[1-5]$')"):
    dic[key]=avg_per_movie(value)

best_movies = [int(k) for k,v in sorted(dic.items(), key=lambda item: -item[1])]
print(best_movies[:3])
```

خروجی:

```
sh-4.1# python3 code/hbase.py
[1316, 1, 1129]
```

یعنی فیلم‌هایی با سه شناسه‌فیلم **1316**، **1** و **1129**، به ترتیب سه فیلم محبوب بازه سنی 45 سال به بالا هستند.

5. میانگین امتیاز خانم‌ها و آقایان در فیلم‌های سال 1995:

دستورات نوشته شده برای یافتن میانگین امتیاز آقایان در فیلم‌های سال 1995:

```
import happybase
import numpy as np

connection = happybase.Connection(host='quickstart.cloudera',port=9090)
connection.open()


table = connection.table('Movie')

avglist = list()
for key,value in table.scan(filter="SingleColumnValueFilter ('data','year',=,'binary:1995') AND "
                                   "ValueFilter (=,'regexstring:^M::[0-9]{2}::[1-5]$')"):
    avglist += list(value.values())

avg = np.mean([int(value[7:]) for value in avglist])
print(avg)
```

خروجی:

```
sh-4.1# python3 code/hbase.py
3.375
```

دستورات نوشته شده برای یافتن میانگین امتیاز خانم‌ها در فیلم‌های سال 1995:

```
import happybase
import numpy as np

connection = happybase.Connection(host='quickstart.cloudera',port=9090)
connection.open()


table = connection.table('Movie')

avglist = list()
for key,value in table.scan(filter="SingleColumnValueFilter ('data','year',=,'binary:1995') AND "
                                   "ValueFilter (=,'regexstring:^F::[0-9]{2}::[1-5]$')"):
    avglist += list(value.values())

avg = np.mean([int(value[7:]) for value in avglist])
print(avg)
```

خروجی:

```
sh-4.1# python3 code/hbase.py
4.0
```