دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

به نام خدا

آزمایشگاه پایگاه داده

جلسه نهم

دستورات اولیه‌ی

MongoDB

ایجاد پایگاه داده

- برای ایجاد یک پایگاه داده‌ی جدید از دستور زیر استفاده کنیم. توجه داشته باشید که در صورتی که نام استفاده شده از قبل وجود داشته باشد با پیغام خطا مواجه می‌شویم.

```
> use DATABASE_NAME
```

- به عنوان مثال برای ایجاد یک پایگاه داده به نام aut_ce دستور زیر را وارد می‌کنیم:

```
> use aut_ce
switched to db aut_ce
```

بررسی پایگاه داده‌های موجود

- در به منظور بررسی پایگاه‌داده‌ای که در حال حاضر درون آن هستیم از دستور زیر استفاده می‌کنیم:

```
> db
aut_ce
```

- برای بررسی پایگاه داده‌های موجود از دستور زیر استفاده می‌کنیم:

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

- سوال: پایگاه داده‌ی aut_ce در لیست گزارش شده موجود نیست. چرا؟

حذف پایگاه داده

- به منظور حذف پایگاه داده‌ی فعلی دستور زیر را وارد می‌کنیم:

```
> db.dropDatabase()
{ "dropped" : "aut_ce", "ok" : 1 }
```

- توجه: در دستور dropDatabase() رعایت حروف کوچک و بزرگ الزامی است! (همانند دیگر دستورات MongoDB)

ایجاد مجموعه

- برای ایجاد یک مجموعه‌ی جدید از دستوری به شکل زیر استفاده می‌کنیم:

```
> db.createCollection(name, options)
```

فیلد	نوع	توضیحات
capped	بولین	(اختیاری) اگر true باشد، یک مجموعه capped فعال می‌شود. مجموعه capped مجموعه‌ای با اندازه‌ی ثابت است که زمانی که به حداکثر اندازه‌ی خود می‌رسد، به طور خودکار قدیمی‌ترین ورودی خود را پاک کرده و در محل آن ورودی جدید را می‌نویسد. اگر مقدار آن true باشد، باید پارامتر size را نیز مشخص کنید.
autoIndexId	بولین	(اختیاری) اگر true باشد، به طور خودکار در فیلد _id مقدار وارد می‌کند. مقدار پیش‌فرض این فیلد false است. این فیلد در نسخه‌های بعدی حذف خواهد شد.
size	عدد	(اختیاری) حداکثر اندازه در واحد بایت برای یک مجموعه capped است.
max	عدد	(اختیاری) حداکثر تعداد اسناد مجاز در مجموعه‌ی capped را مشخص می‌کند.

ایجاد مجموعه

- به عنوان مثال برای ایجاد یک مجموعه به نام software بدون تعیین فیلدهای options به صورت زیر عمل می‌کنیم:

```
> db.createCollection("software")
{ "ok" : 1 }
```

- برای مشاهده‌ی مجموعه‌های موجود دستور زیر را وارد می‌کنیم:

```
> show collections
software
```

- همچنین برای ایجاد یک مجموعه به نام network با تعیین برخی از فیلدهای options به صورت زیر عمل می‌کنیم:

```
> db.createCollection("network", {capped : true, autoIndexId : true, size : 6142800, max : 10000})
{ "ok" : 1 }
```

حذف مجموعه

- برای حذف مجموعه و متعلقات آن از دستور زیر استفاده می‌کنیم:

```
> db.COLLECTION_NAME.drop()
```

- به عنوان مثال برای حذف مجموعه‌ی network از پایگاه‌داده به صورت زیر عمل می‌کنیم:

```
> db.network.drop()
true
```

درج سند

- برای درج یک سند درون یک مجموعه، از دستوری به شکل زیر استفاده می‌کنیم:

```
>db.COLLECTION_NAME.insert(document)
```

- به عنوان مثال برای درج یک سند حاوی نام و نام خانوادگی درون یک مجموعه به نام users به صورت زیر عمل می‌کنیم. (در صورتی که مجموعه‌ی مورد نظر موجود نباشد یک مجموعه‌ی جدید به این نام ساخته می‌شود.)

```
> db.users.insert( { "firstName" : "Ali", "lastName" : "Ahmadi" } )
WriteResult({ "nInserted" : 1 })
```

- برای دیدن تمامی اسناد موجود در یک مجموعه، به صورت زیر عمل می‌کنیم:

```
> db.users.find()
{ "_id" : ObjectId("5f412b67a0e165e5a9bc4e06"), "firstName" : "Ali", "lastName" : "Ahmadi" }
```

تعریف عبارت JSON

- سندی که به عنوان پارامتر ورودی در insert نوشته می‌شود، یک عبارت JSON است.
- یک عبارت JSON می‌تواند به یکی از دو صورت ذیل تعریف شود:
  - یا عبارت JSON (از همان ابتدا) مشخص‌کننده‌ی یک شیء می‌باشد، که در این صورت کل عبارت بین {}نوشته می‌شود.
  - یا عبارت JSON (از همان ابتدا) مشخص‌کننده‌ی یک مجموعه (آرایه) می‌باشد، که در این صورت کل عبارت بین [] نوشته می‌شود.

تعریف عبارت JSON

- نکات زیر نیز باید مورد توجه قرار گرفته شوند:

- یک شی JSON، چیزی بیش از یک سری Key, Value نمی‌باشد. باید دقت داشته باشیم که بر خلاف اشیاء JavaScript (که البته بسیار به عبارات JSON شبیه می‌باشند)، باید کلمات Key، بین "" نوشته شود.
- بعد از Key، و قبل از Value، همیشه از : استفاده می‌شود.
- بین Key, Value ها، باید , نوشته شود.
- در صورتی که Value، یک متن باشد، باید بین "" نوشته شود.
- در صورتی که Value، یک Boolean باشد، می‌تواند فقط یکی از دو مقدار true و یا false را به خودش اختصاص دهد. دقت داشته باشیم که دستورات و مقادیر از پیش تعریف شده JavaScript از قانون Case Sensitive تبعیت می‌کنند، لذا نسبت به بزرگ و کوچک نوشتن آن‌ها حساس می‌باشد.
- در صورتی که Value، یک عدد باشد، بدون هیچ پیشوند و پسوندی (بدون "") نوشته می‌شود.
- در صورتی که Value، تهی (null) باشد، به همان شکل null نوشته می‌شود.
- Value می‌تواند خودش یک شی یا یک آرایه از هر چیزی باشد و به صورت تو در تو (Nested) نوشته شود.

به روز رسانی سند

- برای به روز رسانی مقدار یا مقادیر یک سند از دستوری به شکل زیر استفاده می‌کنیم:

```
>db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
```

- به عنوان مثال در مجموعه‌ی aut_ce برای به روز رسانی مقدار title مربوط به سندهایی که مقدار title فعلی آن‌ها ceit است و تغییر آن به ce به شکل زیر عمل می‌کنیم:

```
>db.aut_ce.update({'title':'ceit'},{$set:{'title':'ce'}}, {multi:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

- توجه: مقدار multi را true قرار داده‌ایم تا همه‌ی سندهایی که شرط مورد نظر را دارند به روز رسانی شود. به صورت پیش‌فرض مقدار multi را false در نظر گرفته‌اند و فرآیند به روز رسانی تنها برای یک سند اجرا می‌شود.

- سوال: توابع save()، findOneAndUpdate()، updateOne() و updateMany() چه عملیاتی را انجام می‌دهند؟

حذف سند

- برای حذف سند در یک مجموعه دستوری به شکل زیر را اجرا می‌کنیم:

```
>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
```

- به عنوان مثال دستور زیر تمامی سندهایی که title آن‌ها مقدار IT دارد را حذف می‌کند:

```
>db.aut_ce.remove({'title':'IT'})
WriteResult({"nRemoved" : 1})
```

- توجه: برای حذف تمامی سندهای یک مجموعه کافی است شرط درون متد remove را خالی بگذارید.
- توجه: اگر بخواهید تنها یک سند را حذف کنید، مقدار ۱ را بعد از شرط درون متد remove قرار دهید.

تابع find

- همانگونه که پیش از این دیدیم برای دیدن فهرست کلیه‌ی سندهای یک مجموعه از دستور زیر استفاده می‌کنیم:

```
> db.users.find()
```

- به عنوان مثال با استفاده از دستور زیر می‌توان تمامی افرادی که دارای اسم Ali هستند را از مجموعه‌ی users پیدا کرد.

```
> db.users.find( { "firstName" : "Ali" } )
```

- برای شمارش تعداد سندهایی که توسط find پیدا می‌شود می‌توان در انتهای دستور آن از count(). استفاده کرد. برای محدود کردن تعداد اسنادی که توسط find پیدا می‌شود از limit(NUM). استفاده می‌شود، که به جای NUM باید تعداد اسنادی که می‌خواهید نمایش دهید باشد.

تابع find

عملیات	Syntax	مثال
مساوی	{<key>:{$eg;<value>}}	db.aut_ce.find({"course":"DBLAB"})
کوچکتر	{<key>:{$lt:<value>}}	db.aut_ce.find({"count":{$lt:50}})
کوچکتر مساوی	{<key>:{$lte:<value>} }	db.aut_ce.find({"count":{$lte:50}})
بزرگتر	{<key>:{$gt:<value>}}	db.aut_ce.find({"count":{$gt:50}})
بزرگتر مساوی	{<key>:{$gte:<value>} }	db.aut_ce.find({"count":{$gte:50}})
نامساوی	{<key>:{$ne:<value>} }	db.aut_ce.find({"count":{$ne:50}})
وجود مقادیر در آرایه	{<key>:{$in:[<value1> , <value2>,……<valueN >]}}	db.aut_ce.find({"name":{$in:["Ali", "Sahel", "Farhang"]}})
عدم وجود مقادیر در آرایه	{<key>:{$nin:<value>} }	db.aut_ce.find({"name":{$nin:["Ali", "Sahel"]}})

عملگر منطقی AND و OR

- با استفاده از دو نوع دستور زیر به ترتیب AND و OR منطقی را می‌توان اجرا کرد:

>db.COLLECTION_NAME.find({ $and: [{<key1>:<value1>}, { <key2>:<value2>}] })

>db. COLLECTION_NAME.find({ $or: [{<key1>:<value1>}, {<key2>:<value2>}] })

- به عنوان مثال دستور زیر فهرست کاربرانی که سن آن‌ها بزرگتر از ۲۰ سال و کوچکتر از ۳۰ سال است را پیدا می‌کند(AND منطقی):

> db.users.find({ "age" : { $gt : 20 } , "age" : { $lt : 30 } })

تابع sort

- با استفاده از تابع sort و ارسال یک عبارت JSON به آن، مشخص می‌کنیم که نتیجه اطلاعات، بر اساس چه فیلد و یا فیلدهایی مرتب شده و نمایش داده شوند. برای این منظور از دستوری به شکل زیر استفاده می‌کنیم:

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

- فرض کنید که می‌خواهیم فهرست کاربران را مرتب شده بر حسب سن آن‌ها مشاهده نماییم. برای این منظور از دستور ذیل استفاده می‌کنیم:

```
> db.users.find().sort( { "age" : 1 } )
```

- در دستور بالا عدد ۱ نشان دهنده‌ی صعودی بودن مرتب‌سازی است. در صورتی که به جای آن از ۱- استفاده کنیم، مرتب‌سازی نزولی انجام می‌شود.

تمرین

پایگاه داده‌ی نمونه‌ی restaurants.json که به همراه دستورکار قرارداده شده است را با استفاده از دستور زیر import کنید:

mongoimport --db **DatabaseName** --collection **CollectionName** --drop --file **FilePath**

ساختار این پایگاه داده به صورت زیر است:


```
{
  "address": {
     "building": "1007",
     "coord": [ -73.856077, 40.848447 ],
     "street": "Morris Park Ave",
     "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
     { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
     { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
     { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10
},
     { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
     { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

تمرین

پرس و جوهای زیر را در پایگاه داده‌ی بالا اجرا کرده و پرس جوی خواسته شده را به همراه بخش ابتدای خروجی را در گزارش خود بیاورید:

1. تمامی اسناد موجود در پایگاه داده را نمایش دهید.
2. رستوران‌هایی که فیلد borough آن‌ها Bronx است را نمایش دهید.
3. تعداد رستوران‌هایی که score کمتر از ۳۰ دارند را نمایش دهید.
4. رستوران‌هایی که فیلد cuisine آن‌ها American نیست و امتیاز نمره‌ی آن‌ها A است را بر اساس cuisine به صورت نزولی مرتب کرده و نمایش دهید.