



به نام خدا  
دانشگاه تهران  
دانشکده مهندسی برق و کامپیوتر



## درس آزمایشگاه پایگاه داده دستورکار اول

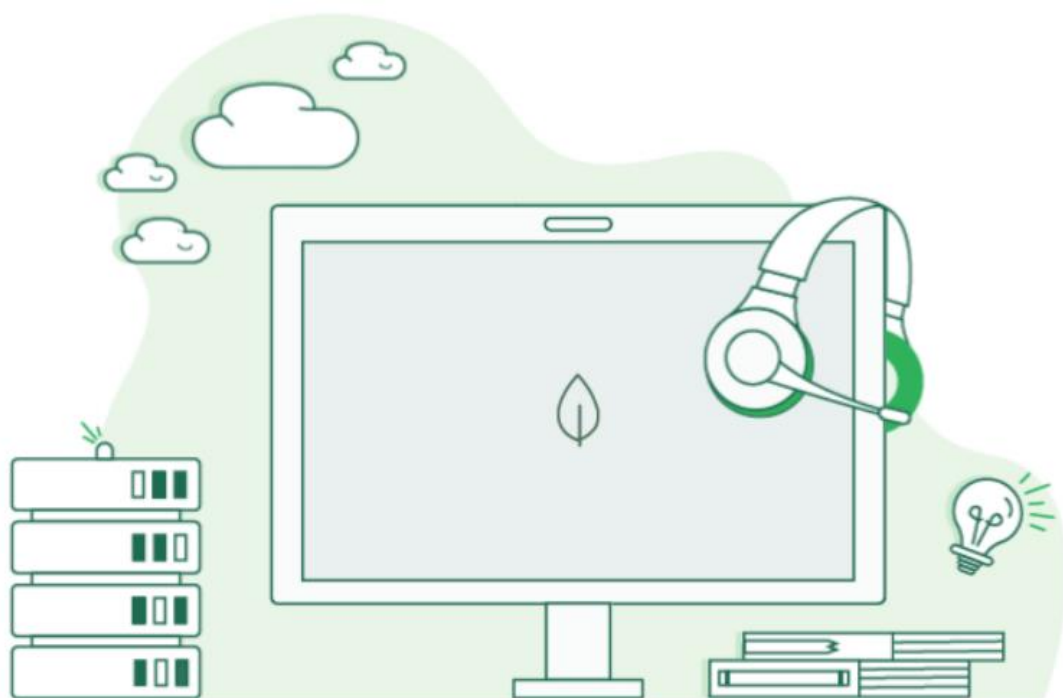
نام و نام خانوادگی	نام نام خانوادگی
شماره دانشجویی	۸۱۰۱۹۹۴۶۱
تاریخ ارسال گزارش	۱۴۰۲.۰۹.۰۴

## فهرست

پاسخ ۱. Introduction to MongoDB	۱
۱-۱. insert	۳
۱-۲. find	۳
۱-۳. replace	۳
۱-۴. delete	۳
۱-۵. aggregation	۳
پاسخ ۲ - retrogames	۱۵
۱-۲. انجام مراحل	۱۵

## پاسخ ۱. Introduction to MongoDB

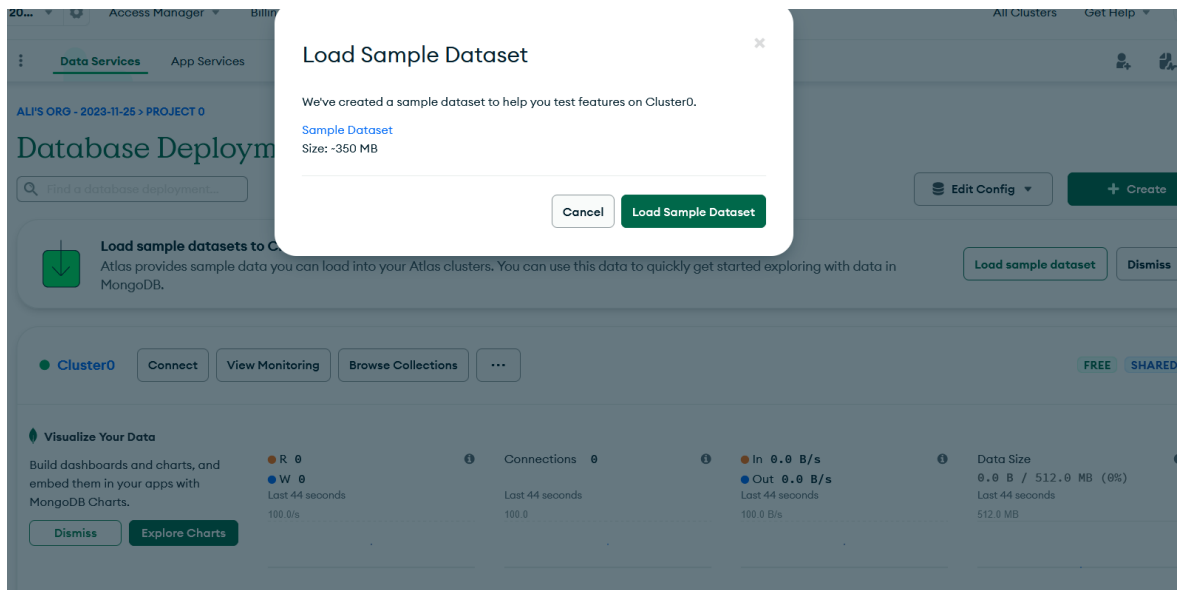
ابتدا در این وبسایت اکانتی ساخته شد.



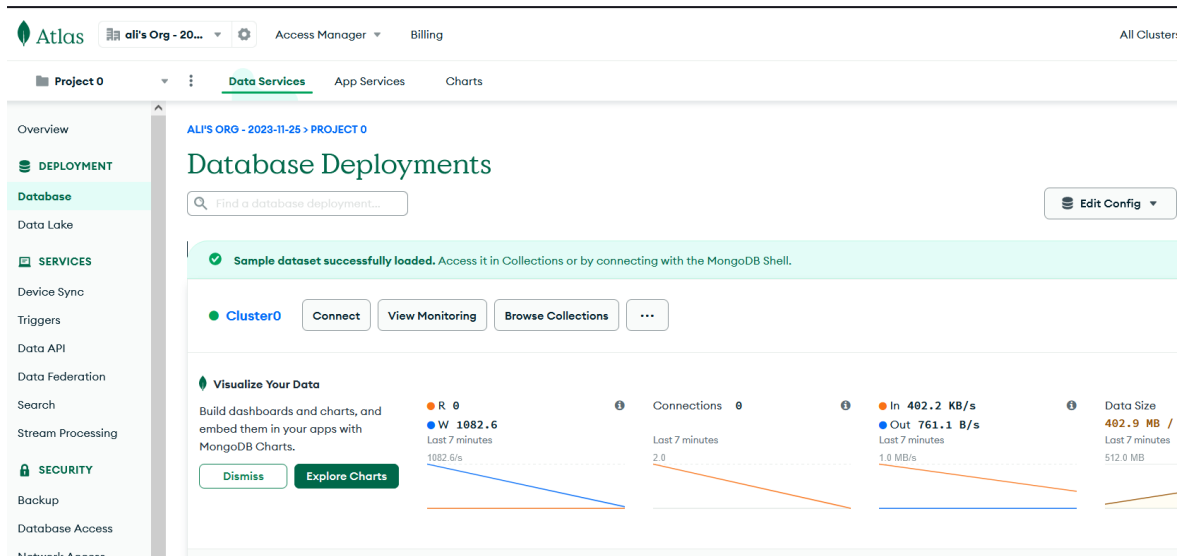
## Welcome!

Use your account to deploy a **cloud database**  
with **MongoDB Atlas** and contact **Support**.

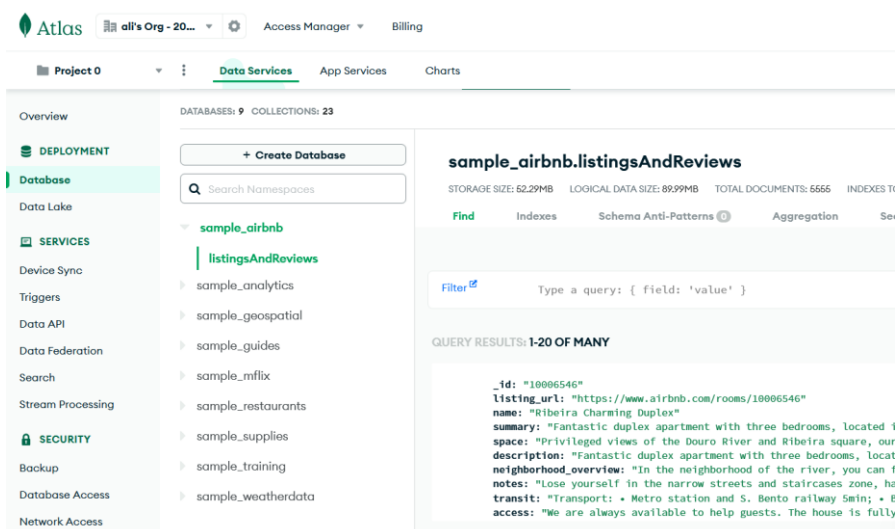
سپس مراحل را تا قسمتی می‌رویم که بتوانیم sample ها را لود بکنیم.



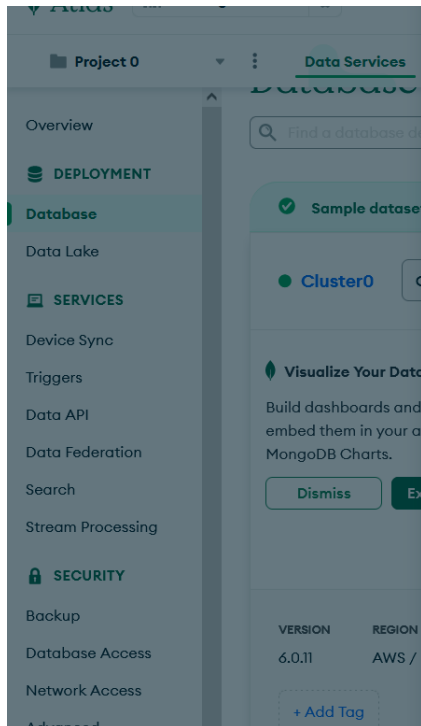
که دیتا بدین شکل لود شد.



در اینجا می توانیم data را اضافه کم یا مودیفای کنیم. که دستورات گفته شده انجام شد.



در ادامه با محیط این قسمت آشنا شدیم و سپس به دیتابیس mongodb متصل شدیم.



### Connect to your application



**Drivers**  
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

### Access your data through tools



**Compass**  
Explore, modify, and visualize your data with MongoDB's GUI



**Shell**  
Quickly add & update data using MongoDB's Javascript command-line interface

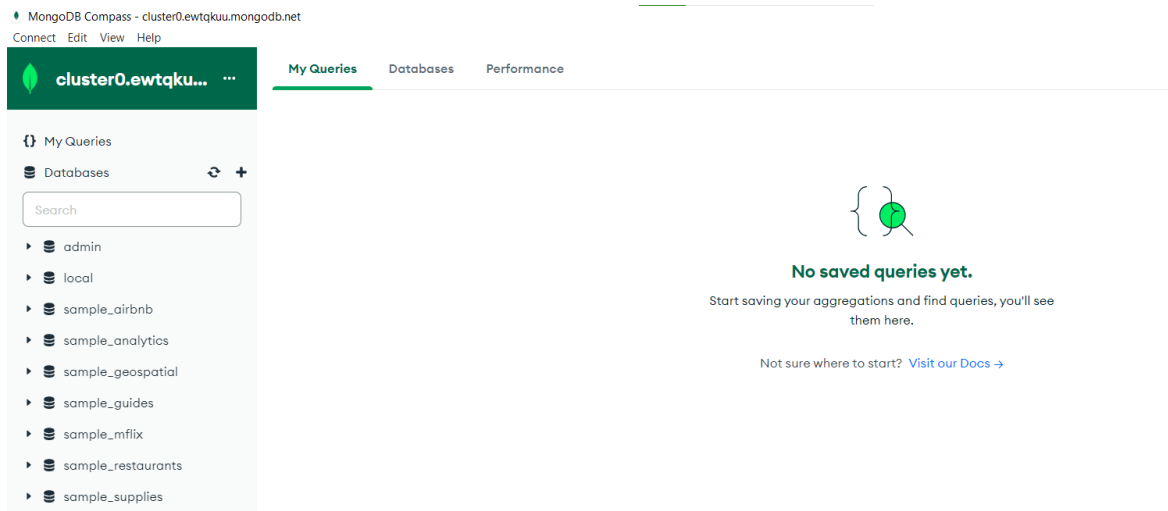


**MongoDB for VS Code**  
Work with your data in MongoDB directly from your VS Code environment



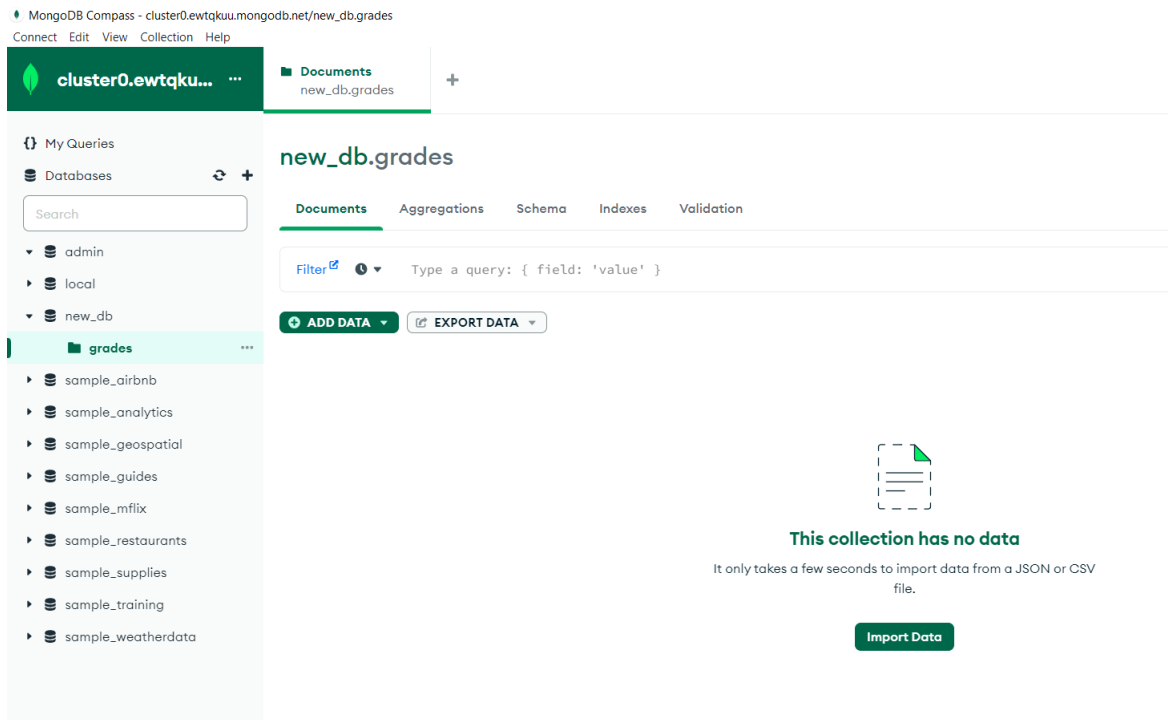
**Atlas SQL**  
Easily connect SQL tools to Atlas for data analysis and visualization

حال وارد محیط MongoDB Compass می شویم تا دستورات را با استفاده از MongoDB Shell وارد کنیم (قبل از آن به دیتابیس کانکت کرده ایم).



۱-۱. insert

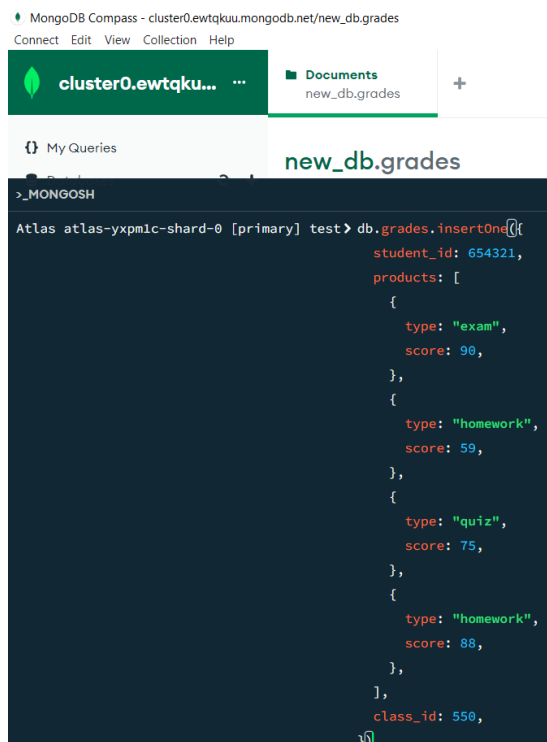
برای انجام این بخش ابتدا new\_db با داکيومنت grades ایجاد شد و حال دستورات روی آن اجرا می شود.

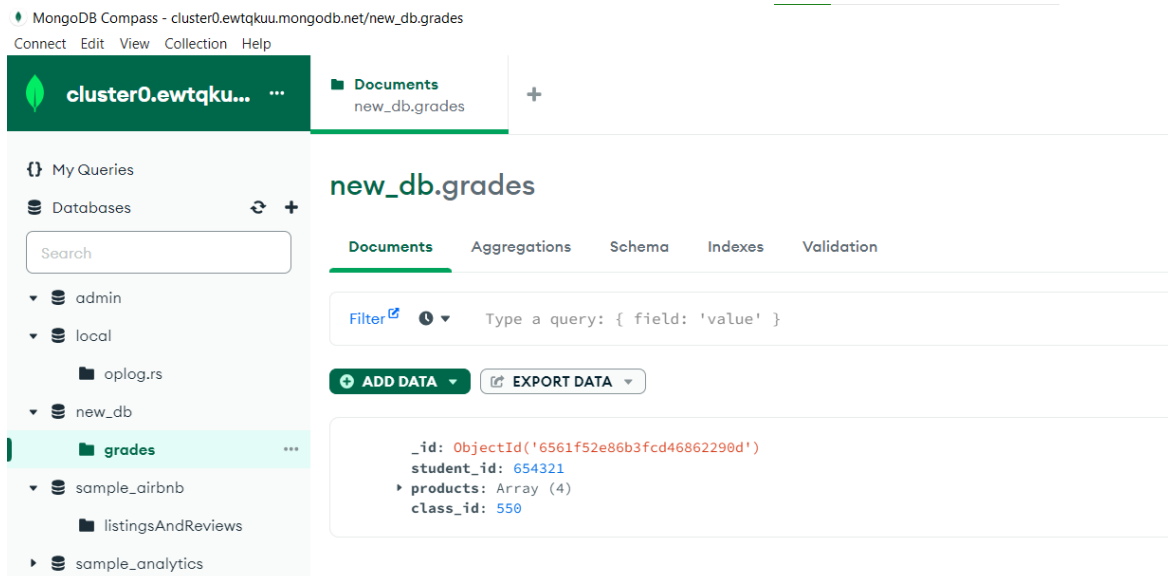


برای insert دو حالت داریم: ۱- insertOne - ۲- insertMany

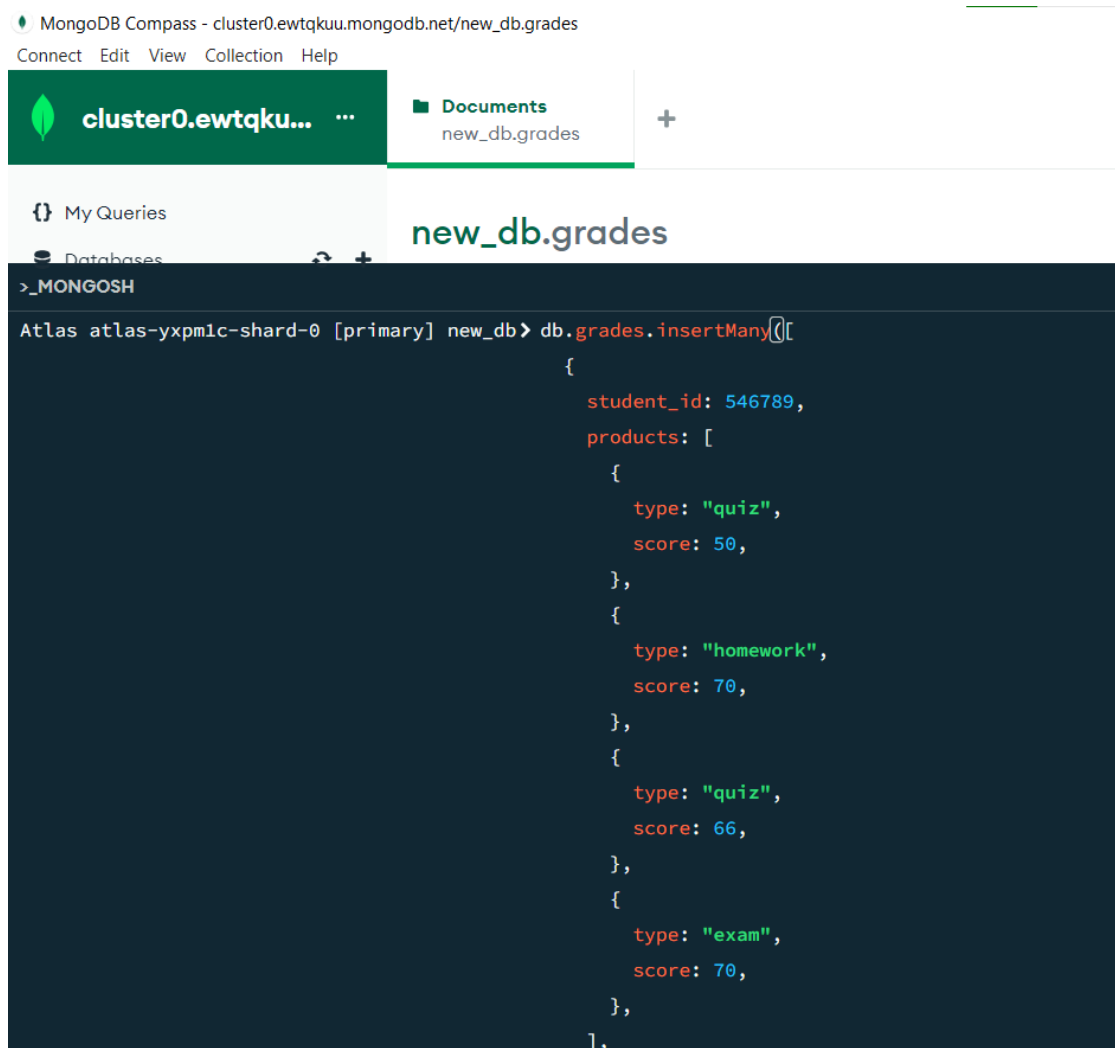
حال برای اضافه کردن به کالکشن اسم آنرا میگوییم و اضافه می کنیم.

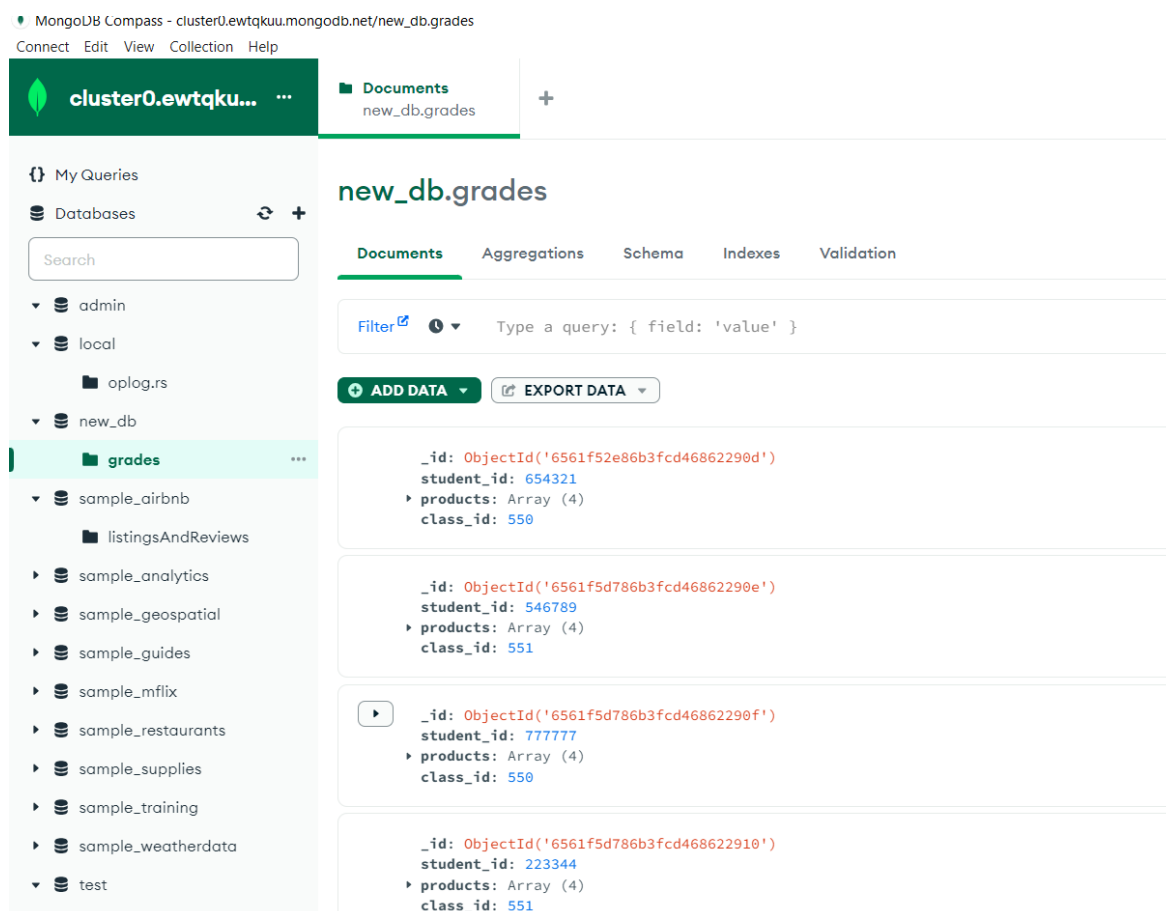
دستور insertOne در اینجا بدین صورت که یک document که فرمت آن Json بوده و به صورت Bson ذخیره می شود به grades اضافه می شود و خروجی آن در زیر نمایش داده ایم.





حالا از insertMany استفاده می شود و خروجی در زیر آن نمایش می دهیم. که توضیحات آن مانند قسمت قبل است.





۲-۱. find

ابتدا طبق ویدیو از sample\_training استفاده می کنیم. برای یافتن میتوانیم از find، findOne، countDocuments استفاده کنیم. اولی تمام داده هایی که متج می شوند ریترن می شوند. دومی نتیجه کوئری را می گوید و سومی تعداد این نتیجه هایی که بدست می آیند.

find:

```

s_MONGOOSH
> db.zips.find()
< {
  _id: ObjectId("5c8eccc1ca187d17ca6ed16"),
  city: 'ALPINE',
  zip: '35014',
  loc: {
    y: 33.331165,
    x: 86.208934
  },
  pop: 3062,
  state: 'AL'
}
{
  _id: ObjectId("5c8eccc1ca187d17ca6ed17"),

```

با زدن find تمام داکيومنت های مربوط به grades بدست آورده می شود.



findOne:

```
>_MONGOSH
> db.zips.findOne()
< {
  _id: ObjectId("5c8eccc1caa187d17ca6ed16"),
  city: 'ALPINE',
  zip: '35014',
  loc: {
    y: 33.331165,
    x: 86.208934
  },
  pop: 3062,
  state: 'AL'
}
Atlas atlas-yxpm1c-shard-0 [primary] sample_training>|
```

با زدن find تمام داکيومنت های مربوط به grades بدست آورده می شود.

در اینجا می توان از الگوهایی برای کوئری ها استفاده کرد تا خروجی جستجو را تعیین کرد. شرط هایی که میتوان استفاده کرد عبارت اند از:

شرط	توضیح	نمونه
\$eq	\$eq را می توانیم حذف کنیم.	db
\$in	وجود مقدار مورد نظر در میان عناصر مورد نظر	
\$lt \$lte \$ne \$gt \$gte	برای انجام مقایسه	
\$elemMatch	شرط روی یک المان از آرایه ای	
\$or و \$and	دقت شود که , همان معادل \$and است در اکثر مواقع	

تمام دستورات بالا امتحان شده اند و در زیر تصاویر آن آمده است.

```
>_MONGOSH
> db.zips.find({city: {$eq:"PHOENIX"}})
< {
  _id: ObjectId("5c8eccc1caa187d17ca6f010"),
  city: 'PHOENIX',
  zip: '85012',
  loc: {
    y: 33.509744,
    x: 112.067816
  },
  pop: 6141,
  state: 'AZ'
}
{
  _id: ObjectId("5c8eccc1caa187d17ca6f011"),
  city: 'PHOENIX',
  zip: '85012',
  loc: {
    y: 33.509744,
    x: 112.067816
  },
  pop: 6141,
  state: 'AZ'
}
```

sample\_supplies

sample\_training

companies

grades

inspections

Filter

Type a query: { field: 'value' }

ADD DATA

EXPORT DATA

\_id: ObjectId('56d61033a378eccde8a8354f')

>\_MONGOSH

> db.zips.find({state: {\$in:["AL", "AZ"]}})

< {

\_id: ObjectId("5c8eccc1caa187d17ca6ed16"),

city: 'ALPINE',

zip: '35014',

loc: {

y: 33.331165,

x: 86.208934

}

pop: 3062,

state: 'AL'

}

{

\_id: ObjectId("5c8eccc1caa187d17ca6ed17"),

city: 'BESSEMER',

grades

inspections

\_id: ObjectId('56d61033a378eccde8a8354f')

>\_MONGOSH

> db.zips.findOne({pop : {\$gt : 3000}})

< {

\_id: ObjectId("5c8eccc1caa187d17ca6ed16"),

city: 'ALPINE',

zip: '35014',

loc: {

y: 33.331165,

x: 86.208934

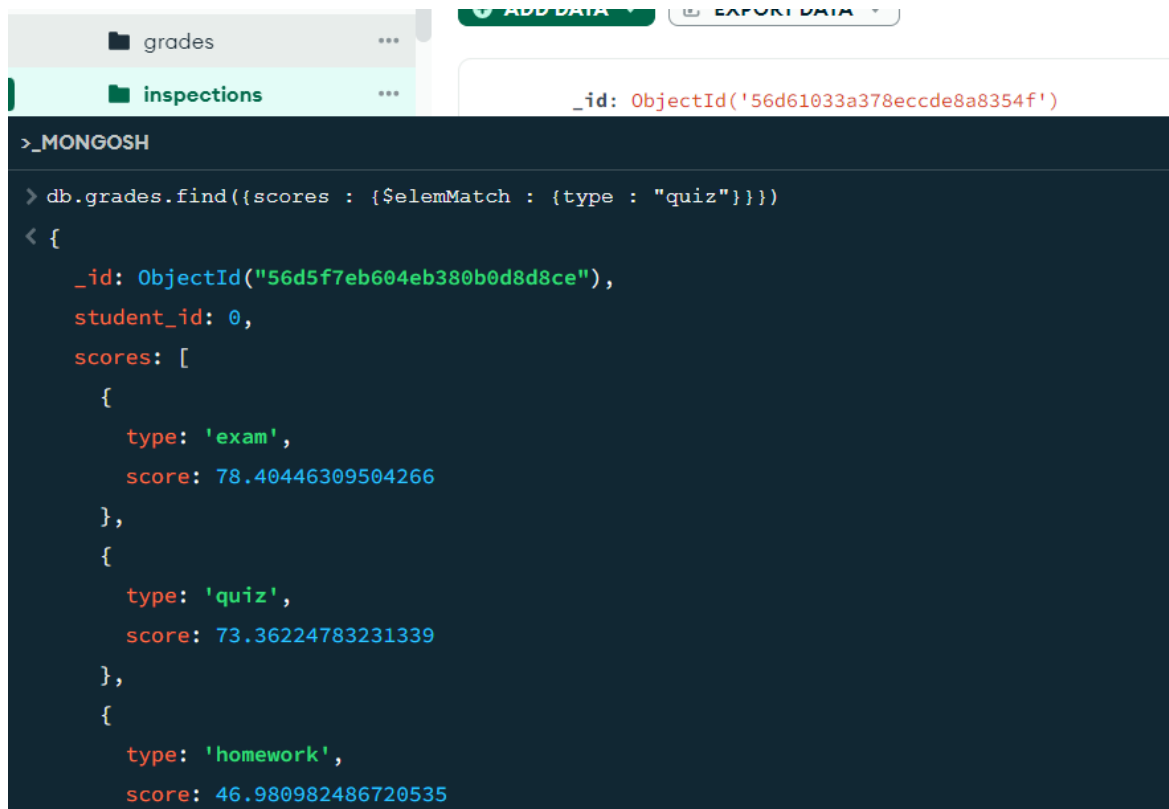
}

pop: 3062,

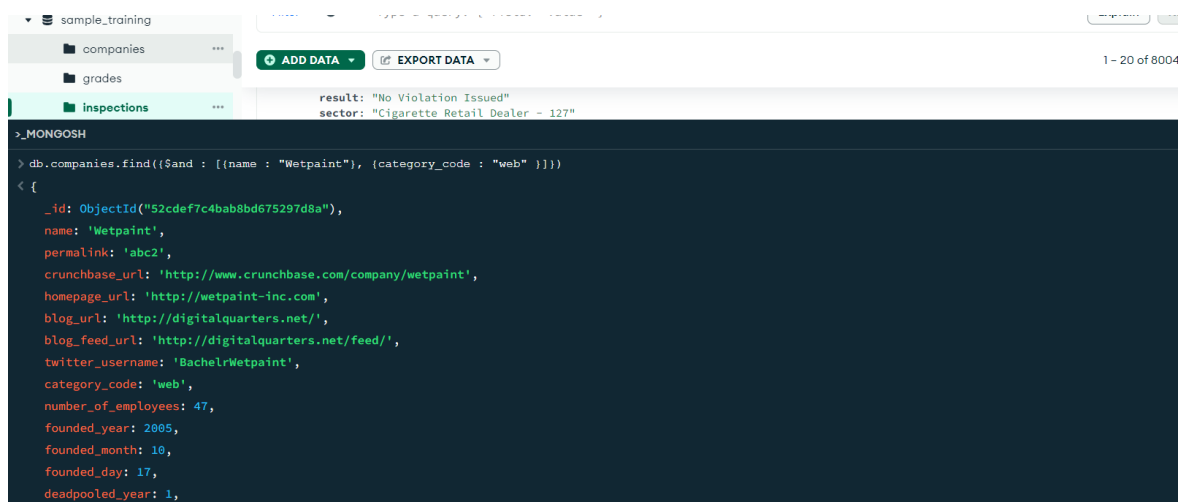
state: 'AL'

}

Atlas atlas-yxpm1c-shard-0 [primary] sample\_training>



```
>_MONGOSH
> db.grades.find({scores : {$elemMatch : {type : "quiz"}}})
< [
  {
    _id: ObjectId("56d5f7eb604eb380b0d8d8ce"),
    student_id: 0,
    scores: [
      {
        type: 'exam',
        score: 78.40446309504266
      },
      {
        type: 'quiz',
        score: 73.36224783231339
      },
      {
        type: 'homework',
        score: 46.980982486720535
      }
    ]
  }
]
```



```
>_MONGOSH
> db.companies.find({$and : [{name : "Wetpaint"}, {category_code : "web" } ]})
< [
  {
    _id: ObjectId("52cdef7c4bab8bd675297d8a"),
    name: 'Wetpaint',
    permalink: 'abc2',
    crunchbase_url: 'http://www.crunchbase.com/company/wetpaint',
    homepage_url: 'http://wetpaint-inc.com',
    blog_url: 'http://digitalquarters.net/',
    blog_feed_url: 'http://digitalquarters.net/feed/',
    twitter_username: 'BachelrWetpaint',
    category_code: 'web',
    number_of_employees: 47,
    founded_year: 2005,
    founded_month: 10,
    founded_day: 17,
    deadpooled_year: 1,
  }
]
```

### ۳-۱. replace

از دستوراتی که می توانیم در این بخش استفاده کنیم می توانیم به replaceOne اشاره کنیم و در آن داکيومنتی را با داکيومنتی جدید عوض کنیم. برای در زیر ابتدا یک داکيومنت را با \_id آن پیدا می کنیم و سپس با مقادیر جدیدی آن را جایگزین می کنیم:

```
sample_weatherda... +  ADD DATA EXPORT DATA
```

```
test result: "No Violation Issued"
sector: "Cigarette Retail Dealer - 127"
```

```
>_MONGOSH

> db.routes.replaceOne({},)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

> db.routes.replaceOne(
  {_id: ObjectId("56e9b39b732b6122f877fa31")},
  {
    airline: {
      id: 410,
      name: 'Aerocondor',
      alias: '2B',
      iata: 'ARD'
    }
  }
)

> db.routes.findOne({_id: ObjectId("56e9b39b732b6122f877fa31")})
< {
  _id: ObjectId("56e9b39b732b6122f877fa31"),
  airline: {
    id: 410,
    name: 'Aerocondor',
    alias: '2B',
    iata: 'ARD'
  },
  src_airport: 'CEK',
  dst_airport: 'KZN',
  codeshare: '123',
  stops: 123,
  airplane: 'CR2'
}

Atlas atlas-yxpm1c-shard-0 [primary] sample_training>|
```

۴-۱. delete

برای حذف کردن از توابع deleteOne و deleteMany می توانیم استفاده کنیم. فرض کنیم که می خواهیم یک داکيومنت از routes حذف کنیم. داریم:

```

> db.routes.deleteOne({_id: ObjectId("56e9b39b732b6122f877fa31")})
< {
  acknowledged: true,
  deletedCount: 1
}
Atlas atlas-yxpm1c-shard-0 [primary] sample_training>

```

## ۵-۱. aggregation

از این دستور گزارش دادن و نمایش اطلاعات و تعداد و مجموع و سایر موارد مرتبط با داده ها استفاده می شود.

از جمله دستور هایی که در این قسمت استفاده می شود aggregate است که شامل stage های مختلف میشود. به طور کلی هر aggregation شامل چندین stage است که در هر کدام مقادیری محاسبه می شود و آنها را تغییر نخواهد داد. مثلا در sample\_training استیج \$match که معادل \$WHERE در SQL است. مثلا می توانیم aggregation زیر را بنویسیم که آنهایی که state مربوط به LA را دارند، بر حسب شهر گروه می کنیم و تعداد آنها را بر میگردانیم. سپس براساس جمعیت سورت میکنیم و دوتای اول را نمایش میدهیم. (از چهار استیج استفاده کرده ایم).

The screenshot shows the MongoDB Atlas web interface. On the left, a sidebar lists collections: sample\_guides, sample\_mflix, sample\_restaurants, sample\_supplies, and sample\_training. The main area has a 'Filter' section with a query input field containing '{ field: 'value' }'. Below the filter are buttons for 'ADD DATA' and 'EXPORT DATA'. A result box shows 'result: "No Violation Issued"' and 'sector: "Cigarette Retail Dealer - 127"'. At the bottom, a terminal window shows the following MongoDB command:

```

>_MONGOSH
Atlas atlas-yxpm1c-shard-0 [primary] sample_training> db.zips.aggregate([
  {$match: {state: "LA"}},
  {$group: {
    _id: "$city",
    total: {$count: {}}
  }},
  {$sort: {pop:-1}},
  {$limit: 2}
])

```

```

>_MONGOSH
    {$group: {
      _id: "$city",
      total: {$count: {}}
    }
  },
  {$sort: {pop:-1}},
  {$limit: 2}
])
< {
  _id: 'HARAHAN',
  total: 1
}
{
  _id: 'FRENCH SETTLEMEN',
  total: 1
}

```

می توانیم استیج های دیگری تعریف کنیم. مثلا \$project و \$set که اولی معادل پروجکشن جدول خروجی می باشد و ستون های خاصی را نگه داریم. مثلا اینجا فقط دو ستون و id را پروجکت می کنیم :

```

sample_mflix
sector: "Cigarette Retail D

>_MONGOSH
> db.routes.aggregate([
  {$project: {src_airport:1, dst_airport:1}},
  {$limit: 10}
])
< {
  _id: ObjectId("56e9b39b732b6122f877fa32"),
  src_airport: 'ASF',
  dst_airport: 'KZN'
}

```

\$set هم مانند آپدیت کردن فیلد می باشد (برای ویو جدید). مثلاً اینجا جمعیت را دو برابر میکنیم :

```
sample_mflix      sector: "Tax Preparers - 891"
                  address: Object

>_MONGOSH
Atlas atlas-yxpm1c-shard-0 [primary] sample_training> db.zips.aggregate([
    {$set: {new_zip_pop : {$round: {$multiply: [2.0, "$pop"]}}}},
    {$project: {pop : 1, new_zip_pop : 1}}
])

sample_guides     sector: "Tax Preparers - 891"
sample_mflix      address: Object

>_MONGOSH
> db.zips.aggregate([
    {$set: {new_zip_pop : {$round: {$multiply: [2.0, "$pop"]}}}},
    {$project: {pop : 1, new_zip_pop : 1}}
])
< {
  _id: ObjectId("5c8eccc1caa187d17ca6ed16"),
  pop: 3062,
  new_zip_pop: 6124
}
{
  _id: ObjectId("5c8eccc1caa187d17ca6ed17"),
  pop: 40549,
  new_zip_pop: 81098
}
{
  _id: ObjectId("5c8eccc1caa187d17ca6ed18"),
  pop: 40549,
  new_zip_pop: 81098
}
```

استیج دیگری هم وجود دارد به نام \$out که باید استیج آخر باشد و خروجی را در کالکشن جدیدی میریزد (شبیه ویو است).

```
> db.zips.aggregate([
  {$match: {state: "LA"}},
  {$group: {
    _id: "$city",
    total: {$count: {}}
  }
},
{$sort: {pop:-1}},
{$limit: 2},
{$out: "output"}
])
<
> show collections
< companies
  grades
  inspections
  output
  posts
  routes
  trips
  zips
```

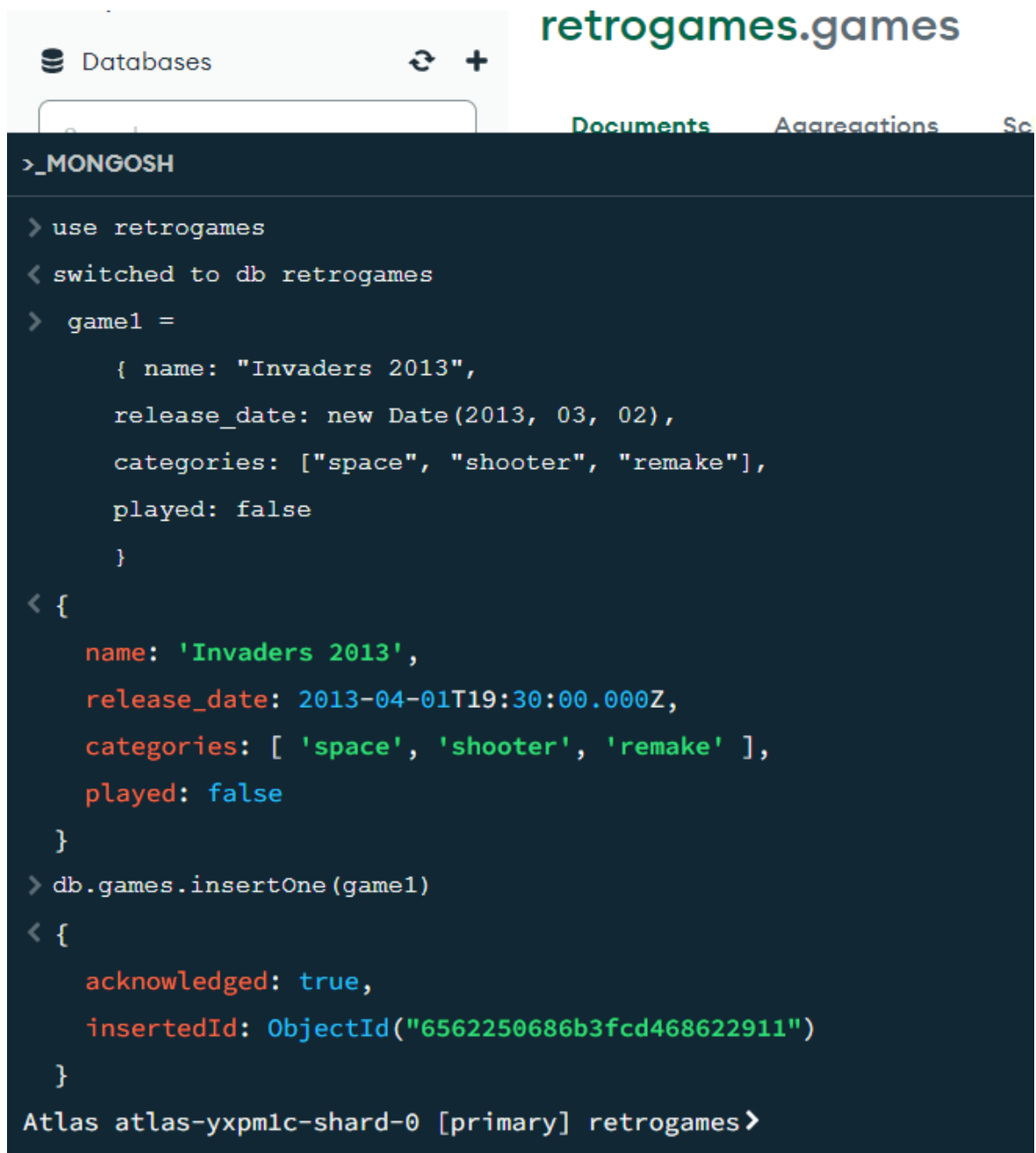
همان aggregate اول را زدیم و خروجی را در output قرار دادیم.



## پاسخ ۲ – عنوان پرسش دوم به فارسی

### ۱-۲. انجام مراحل

دیتابیس گفته شده یعنی retrogames را ساخته و در آن دو کالکشن گفته شده یعنی games و players را قرار می دهیم. حال دستورات را میزنیم (اول یک بازی به بازی ها اضافه میکنیم).



```
>_MONGOSH
> use retrogames
< switched to db retrogames
> game1 =
  { name: "Invaders 2013",
    release_date: new Date(2013, 03, 02),
    categories: ["space", "shooter", "remake"],
    played: false
  }
< {
  name: 'Invaders 2013',
  release_date: 2013-04-01T19:30:00.000Z,
  categories: [ 'space', 'shooter', 'remake' ],
  played: false
}
> db.games.insertOne(game1)
< {
  acknowledged: true,
  insertedId: ObjectId("6562250686b3fcd468622911")
}
Atlas atlas-yxpm1c-shard-0 [primary] retrogames>
```

یک بازی بیشتر در games نداریم که با زدن دستورات مختلف گفته شده میتوانیم این را تایید کنیم:

```
> db.games.find()
< {
  _id: ObjectId("6562250686b3fcd468622911"),
  name: 'Invaders 2013',
  release_date: 2013-04-01T19:30:00.000Z,
  categories: [
    'space',
    'shooter',
    'remake'
  ],
  played: false
}
> db.games.find().limit(100)
< {
  _id: ObjectId("6562250686b3fcd468622911"),
  name: 'Invaders 2013',
  release_date: 2013-04-01T19:30:00.000Z,
  categories: [
    'space',
    'shooter',
    'remake'
  ],
  played: false
}
```

My Queries

Databases

Search

▼ admin

▶ local

▼ new\_db

grades

▼ retrogames

**games** ...

▶ sample\_airbnb

### retrogames.games

Documents   Aggregations   Schema   Indexes   Validation

Filter ⓘ ⓘ ▼ Type a query: { field: 'value' }

➕ ADD DATA ▼   📄 EXPORT DATA ▼

```
_id: ObjectId('6562250686b3fcd468622911')
name: "Invaders 2013"
release_date: 2013-04-01T19:30:00.000+00:00
▶ categories: Array (3)
played: false
```

با توجه به اینکه فقط یک بازی داریم با findOne آنرا پیدا کرده و یک پلیر به players اضافه میکنیم.

```
> player1 =
  { name: "PUZZLEGAMESMASTER",
    gender: "male",
    scores: [
      { game_id: new ObjectId("51e10c50085977bc3cd92a65"),
        game_name: "Invaders 2013",
        score: 10500,
        score_date: new Date(2013, 03, 02)
      }
    ]
  }
< {
  name: 'PUZZLEGAMESMASTER',
  gender: 'male',
  scores: [
    {
      game_id: ObjectId("51e10c50085977bc3cd92a65"),
      game_name: 'Invaders 2013',
      score: 10500,
      score_date: 2013-04-01T19:30:00.000Z
    }
  ]
}
> db.players.insertOne(player1)
< {
  acknowledged: true,
  insertedId: ObjectId("6562267086b3fcd468622912")
}
```

در اینجا با توجه به اینکه هم id و هم نام بازی را ذخیره کرده ایم، یک redundancy ایجاد شده است. در این مرحله باید played بازی true شود. چون که یک نفر در آن بازی کرده است.

```
17. MONGODB
> _MONGOSH
> db.games.update(
  { _id: new ObjectId("51e10c50085977bc3cd92a65") },
  { $set: { played: true } }
)
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

حال باید یک score برای بازیکن اضافه کنیم که push استفاده می کنیم تا به آرایه اضافه کنیم.

```
> db.players.update(  
  { _id: new ObjectId("6562267086b3fcd468622912") },  
  { $push: { scores: { game_id: new ObjectId("6562250686b3fcd468622911"),  
    game_name: "Invaders 2013",  
    score: 30250,  
    score_date: new Date(2013, 03, 03)  
  }  
  }  
})  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}  
Atlas atlas-yxpm1c-shard-0 [primary] retrogames>|
```