|  |  |
| --- | --- |
| نام و نام خانوادگی | **پاشا براهیمی** |
| شماره دانشجویی | **٨١٠١٩٩٣٨٥** |
| تاریخ ارسال گزارش | **٢٧/٠٨/١٤٠٢** |

|  |  |  |
| --- | --- | --- |
|  | **به نام خدا**  **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر** |  |
| **درس آزمایشگاه پایگاه‌داده**  **پیش‌گزارش پنجم** | | |

**فهرست**

[**پاسخ ١**. **آموزش‌های وبسایت اصلی MongoDB** 3](#_Toc151720432)

[۱-۱. قسمت insert 3](#_Toc151720433)

[١-١-١. دستور insertOne 3](#_Toc151720434)

[١-١-٢. دستور insertMany 3](#_Toc151720435)

[١-٢. قسمت find 4](#_Toc151720436)

[دستور find و اپراتور $eq 4](#_Toc151720437)

[١-٢-٢. اپراتور $in 6](#_Toc151720438)

[١-٢-٣. دستور findOne 7](#_Toc151720439)

[١-٢-٤. اپراتور $gt 8](#_Toc151720440)

[١-٢-٥. کار با آرایه 9](#_Toc151720441)

[١-٢-٦. اپراتور منطقی $and 12](#_Toc151720442)

[١-٢-٧. اپراتور منطقی $or 14](#_Toc151720443)

[١-٢-٨. ترکیب اپراتورهای منطقی 15](#_Toc151720444)

[١-٣. قسمت replace 16](#_Toc151720445)

[١-٣-١. تابع replaceOne 16](#_Toc151720446)

[١-٤. بخش update 18](#_Toc151720447)

[١-٤-١. تابع updateOne 18](#_Toc151720448)

[١-٤-٢. آپشن upsert 19](#_Toc151720449)

[١-٤-٣. اپراتور $push 20](#_Toc151720450)

[١-٤-٤. تابع findAndModify 21](#_Toc151720451)

[١-٤-٥. تابع updateMany 22](#_Toc151720452)

[١-٥. بخش delete 23](#_Toc151720453)

[١-٥-١. تابع deleteOne 23](#_Toc151720454)

[١-٥-٢. تابع deleteMany 24](#_Toc151720455)

[١-٦. تغییر نمایش نتایج 24](#_Toc151720456)

[١-٦-١. تابع sort 24](#_Toc151720457)

[١-٦-٢. تابع limit 25](#_Toc151720458)

[١-٦-٣. projection 27](#_Toc151720459)

[١-۶-٤. تابع count 29](#_Toc151720460)

[١-٧. بخش aggregation 30](#_Toc151720461)

[١-٧-١. استیج $match 30](#_Toc151720462)

[١-٧-٢. استیج $group 30](#_Toc151720463)

[١-٧-٣. استیج $sort 31](#_Toc151720464)

[١-٧-٤. استیج $limit 32](#_Toc151720465)

[١-٧-٥. استیج $project 33](#_Toc151720466)

[١-٧-٦. استیج $set 34](#_Toc151720467)

[١-٧-٧. استیج $count 35](#_Toc151720468)

[١-٧-٨. استیج $out 36](#_Toc151720469)

[پاسخ ٢. مقاله سایت مهندسی داده 37](#_Toc151720470)

# **پاسخ ١**. **آموزش‌های وبسایت اصلی MongoDB**

۱-۱. قسمت insert

### ١-١-١. دستور insertOne

db.grades.insertOne({

    student\_id: 546799,

    scores: [

        {

            type: "quiz",

            score: 50

        },

        {

            type: "homework",

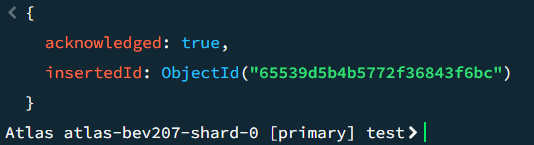
            score: 70

        }

    ]

})

این دستور یک داکیومنت در کالکشن grades اضافه می‌کند. اگر همچین کالکشنی وجود نداشته باشد، این کالکشن به صورت خودکار اضافه می‌شود. لازم به ذکر است که فیلد \_id همواره در هر داکیومنتی وجود دارد و یکتا است. در صورتی که این فیلد را خودمان وارد نکنیم، دیتابیس به صورت خودکار یک id برای داکیومنت ایجاد می‌کند.



### ١-١-٢. دستور insertMany

db.grades.insertMany([

    {

        student\_id: 546789,

        scores: [

            {

                type: "quiz",

                score: 50

            },

            {

                type: "homework",

                score: 70

            }

        ],

        class\_id: 551

    },

    {

        student\_id: 777777,

        scores: [

            {

                type: "quiz",

                score: 72

            },

        ],

        class\_id: 550

    },

    {

        student\_id: 223344,

        scores: [

            {

                type: "exam",

                score: 45

            },

        ],

        class\_id: 551

    }

])

این دستور چند داکیومنت را به صورت همزمان به کالکشن اضافه می‌کند.



## ١-٢. قسمت find

### دستور find و اپراتور $eq

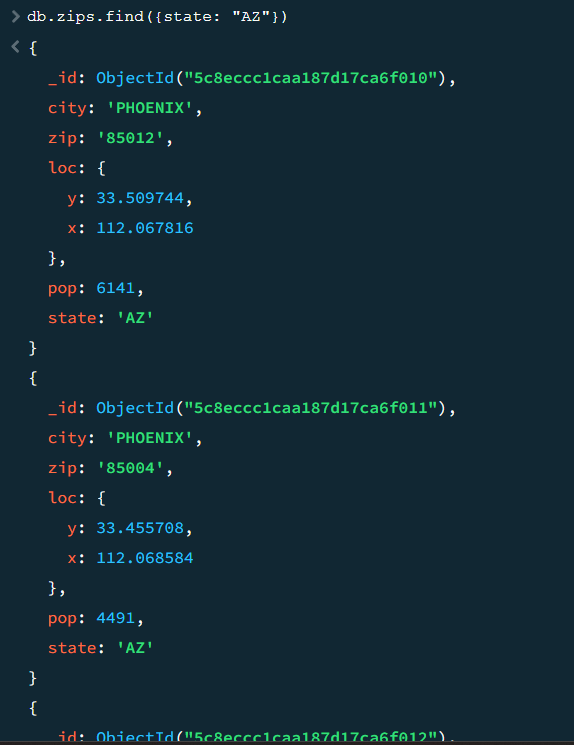
db.zips.find()

این دستور تمام داکیومنت‌های کالکشن zip را بازمی‌گرداند. جهت فیلتر کردن، می‌توانیم پارامترها را در داخل پرانتز مشخص کنیم.



db.zips.find({state: "AZ"})

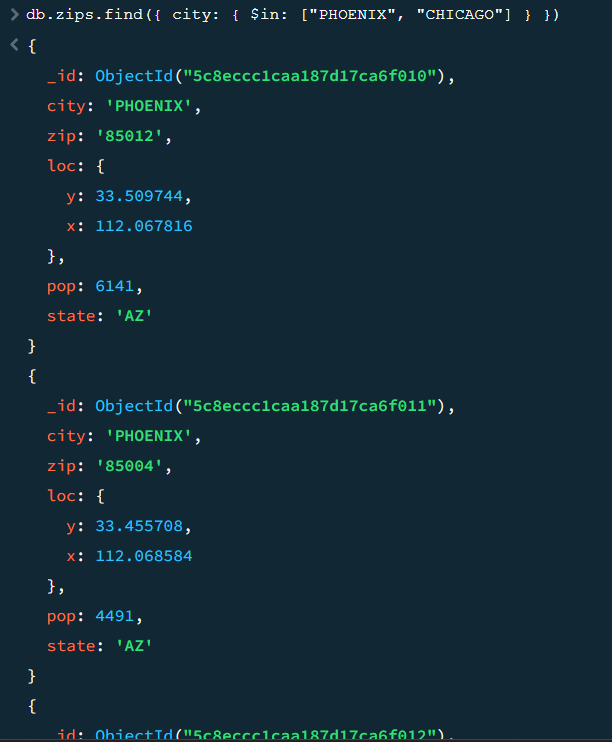
در این حالت، تمامی داکیومنت‌های zip که state آن‌ها آریزونا است برگردانده می‌شود. می‌توانیم از اپراتور $eq هم استفاده کنیم.



### ١-٢-٢. اپراتور $in

db.zips.find({ city: { $in: ["PHOENIX", "CHICAGO"] } })

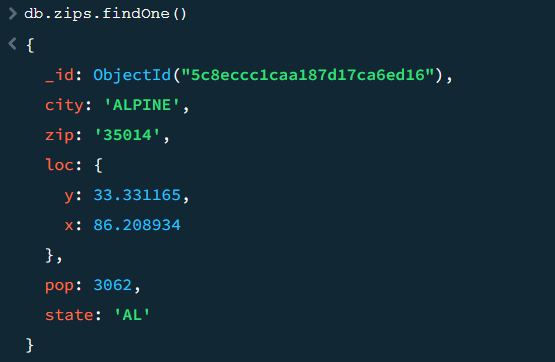
این دستور زمانی استفاده می‌شود که می‌خواهیم مقدار یک فیلد در یک لیست باشد. به نوعی، or کردن مقادیر مختلف را خواهیم داشت.



### ١-٢-٣. دستور findOne

db.zips.findOne()

این دستور یک داکیومنت را از کالکشن zips پیدا کرده و بازمی‌گرداند. برای فیلتر کردن، می‌توانیم پارامترها را در داخل پرانتز مشخص کنیم.



### ١-٢-٤. اپراتور $gt

db.sales.find({ "items.price": { $gt: 50 } })

این دستور تمام sales-هایی را پیدا می‌کند که آیتمی دارد که قیمت آن بیشتر از 50 باشد. همانطور که در تصویر مشخص است، برای فیلدهای nested، می‌توانیم از dot استفاده کنیم.



این کار را با اپراتورهای $lt، $gte و $lte نیز می‌توانیم انجام دهیم.

### ١-٢-٥. کار با آرایه

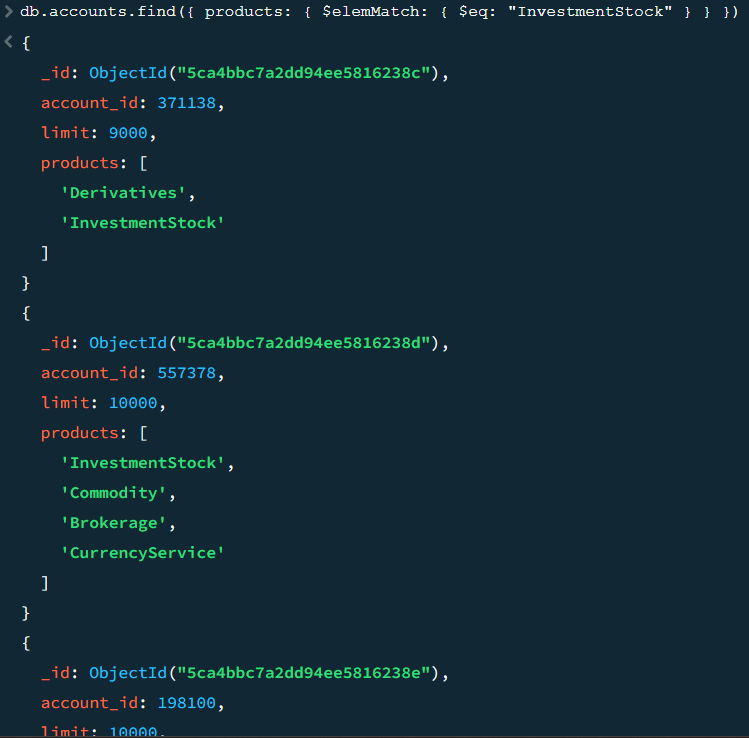
db.accounts.find({ "products": "InvestmentStock" })

این دستور تمام داکیومنت‌هایی از accounts را برمیگرداند که یا مقدار products آن InvestmentStock است و یا اینکه products آرایه‌ای است که مقدار InvestmentStock در آن وجود دارد.



اگر بخواهیم حالتی که مقدار داده شده دقیقا با مقدار فیلد match نشود و و فقط در آرایه باشد، باید از $elemMatch استفاده کنیم:

db.accounts.find({ products: { $elemMatch: { $eq: "InvestmentStock" } } })



db.sales.find({

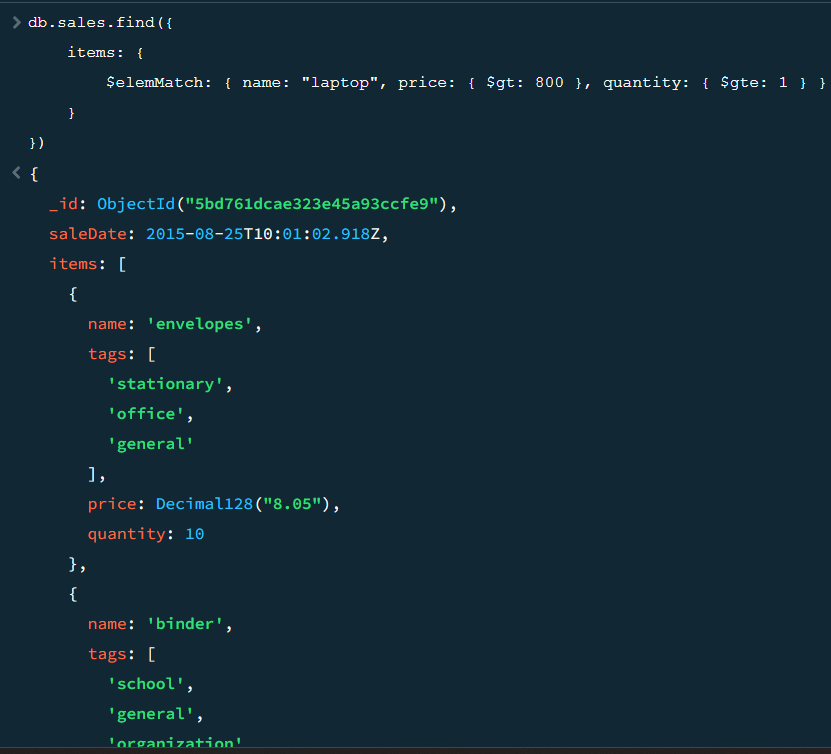
    items: {

        $elemMatch: { name: "laptop", price: { $gt: 800 }, quantity: { $gte: 1 } }

    }

})

این دستور از کالکشن sales داکیومنت‌هایی را پیدا می‌کند که حداقل یک آیتم با نام laptop و قیمت بیشتر از 800 و تعداد بیشتر یا مساوی یک واحد دارند.



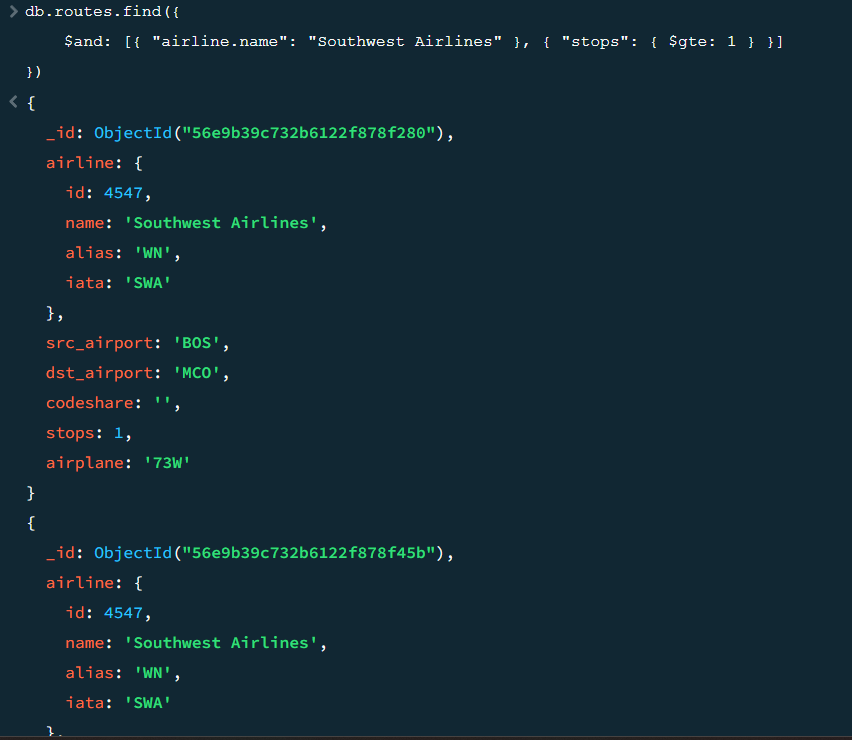
### ١-٢-٦. اپراتور منطقی $and

db.routes.find({

    $and: [{ "airline.name": "Southwest Airlines" }, { "stops": { $gte: 1 } }]

})

این دستور تمام route-هایی که نام ایرلاین آن‌ها Southwest Airlines است و تعداد توقف‌های آن حداقل 1 است را بازمی‌گرداند.



همین کوئری با اپراتور $and به صورت implicit (بدون ذکر اپراتور)، قابل نوشتن است:

db.routes.find({

    "airline.name": "Southwest Airlines", "stops": { $gte: 1 }

})



### ١-٢-٧. اپراتور منطقی $or

db.routes.find({

    $or: [{ "dst\_airport": "SEA" }, { "src\_airport": "SEA" }]

})

این دستور تمام route-هایی را بازمی‌گرداند که مبدا و یا مقصد آن، فرودگاه SEA باشد.



### ١-٢-٨. ترکیب اپراتورهای منطقی

db.routes.find({

    $and: [

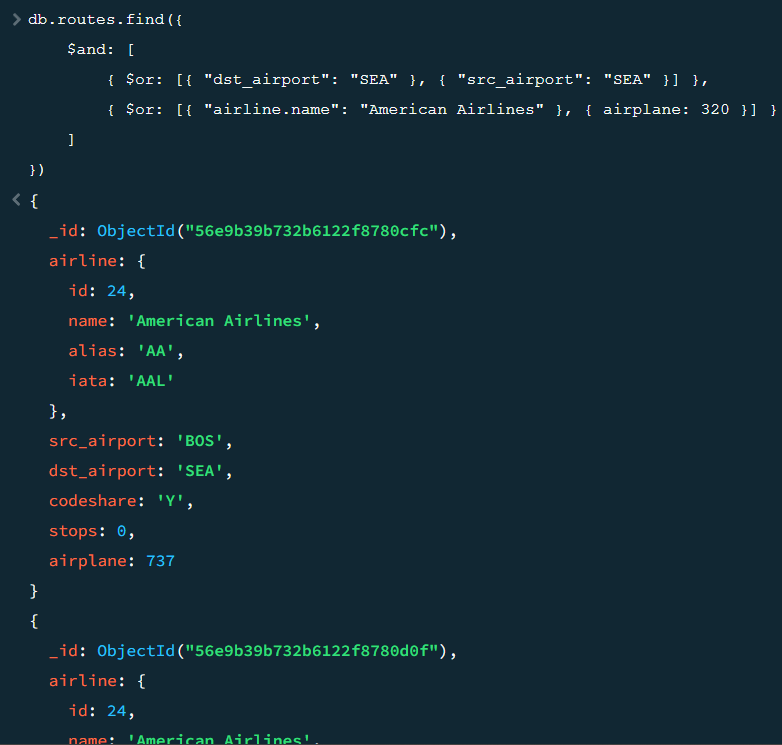
        { $or: [{ "dst\_airport": "SEA" }, { "src\_airport": "SEA" }] },

        { $or: [{ "airline.name": "American Airlines" }, { airplane: 320 }] }

    ]

})

این کوئری تمام route-هایی که هم مبدا یا مقصد آن SEA است و هم اسم ایرلاین American Airlines است یا هواپیمای آن 320 است را بازمی‌گرداند.



در این حالت نمی‌توانیم از حالت implicit برای اپراتور $and استفاده کنیم چون کوئری $or اول توسط $or دوم اورراید می‌شود. دلیل این مورد این است که نمی‌توانیم در یک JSON، دو فیلد با یک نام داشته باشیم. در حالتی که بیش از یک اپراتور استفاده می‌کنیم، باید از حالت explicit اپراتورها استفاده کنیم.

## ١-٣. قسمت replace

### ١-٣-١. تابع replaceOne

db.comments.replaceOne(

    { \_id: ObjectId("5a9427648b0beebeb69579e7") },

    {

        name: 'Mercedes Tyler',

        email: 'mercedes\_tyler@fakegmail.com',

        movie\_id: ObjectId("573a1390f29313caabcd4323"),

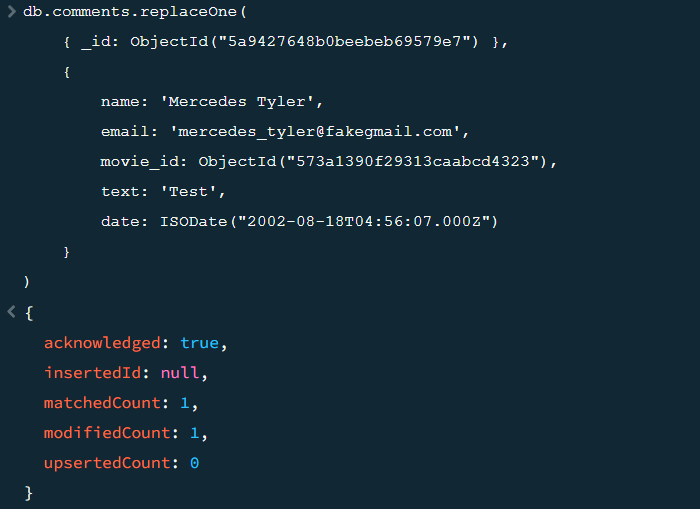
        text: 'Test',

        date: ISODate("2002-08-18T04:56:07.000Z")

    }

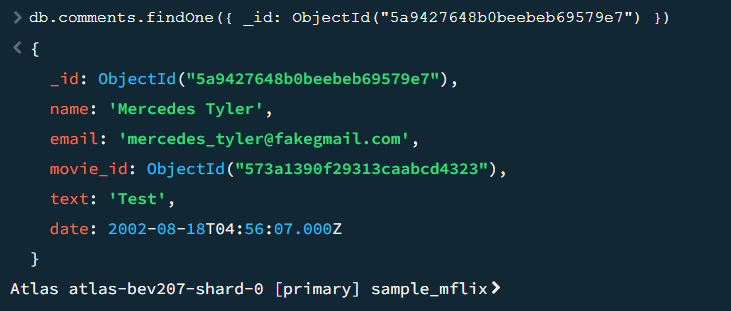
)

این تابع یک داکیومنت با \_id ذکر شده را می‌گیرد، و دقیقا یک داکیومنت که با آیدی مچ می‌شود را با داکیومنت جدید replace می‌کند. در این تغییر، \_id داکیومنت تغییر نمی‌کند.



db.comments.findOne({ \_id: ObjectId("5a9427648b0beebeb69579e7") })

پس از اجرای این دستور، باید مشاهده کنیم قسمت text ادیت شده است:



## ١-٤. بخش update

### ١-٤-١. تابع updateOne

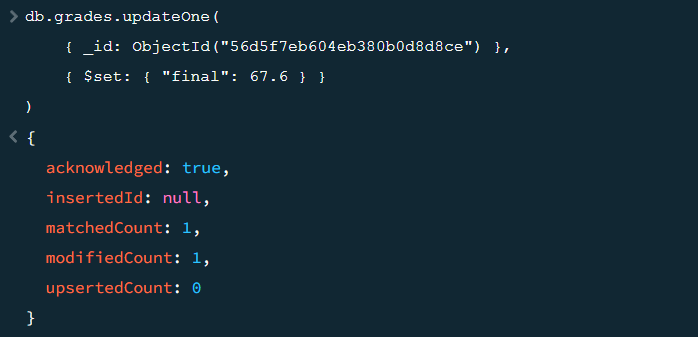
db.grades.updateOne(

    { \_id: ObjectId("56d5f7eb604eb380b0d8d8ce") },

    { $set: { "final": 67.6 } }

)

اپراتور $set که در این بخش استفاده شده، در صورتی که آن فیلد در داکیومنت با \_id ذکر شده وجود داشته باشد، آن را با مقدار جدید replace می‌کند و اگر آن فیلد وجود نداشته باشد، فیلد را اضافه کرده و مقدار ذکر شده را به عنوان مقدار آن قرار می‌دهد.



db.grades.findOne({ \_id: ObjectId("56d5f7eb604eb380b0d8d8ce") })



### ١-٤-٢. آپشن upsert

این تابع به معنای update or insert است که اگر داکیومنت با آن مقادیر وجود نداشته باشد، آن را insert می‌کند. تابع updateOne به تنهایی، در صورتی که همچین داکیومنتی وجود نداشته باشد، آن را insert نمی‌کند.

db.grades.updateOne(

    { student\_id: 20000 },

    { $set: { "final": 88 } },

    { upsert: true }

)



همانطور که مشاهده می‌شود، در حالت اول هیچ تغییری رخ نداده است. اما در حالت دوم، مقدار upsertedCount برابر با 1 شده است.

### ١-٤-٣. اپراتور $push

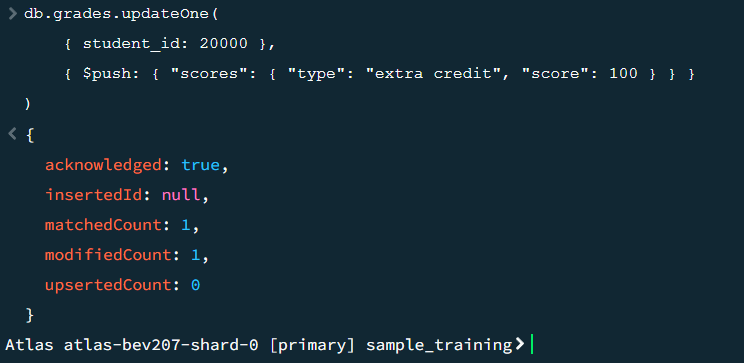
این اپراتور، اگر فیلد مورد نظر به عنوان آرایه وجود داشته باشد، مقدار داده شده را به انتهای آن اضافه می‌کند. اما اگر وجود نداشته باشد، یک آرایه با آن اسم و تنها فیلد مقدار داده شده، قرار می‌دهد.

db.grades.updateOne(

    { student\_id: 20000 },

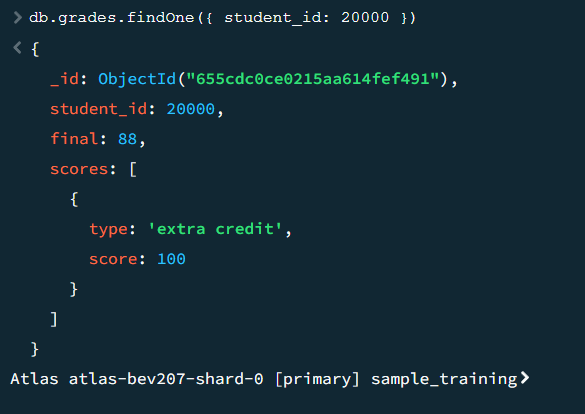
    { $push: { "scores": { "type": "extra credit", "score": 100 } } }

)



همانطور که مشاهده می‌شود، modifiedCount برابر با 1 شده است.

db.grades.findOne({ student\_id: 20000 })



### ١-٤-٤. تابع findAndModify

تفاوت این تابع با تابع updateOne این است که پس از آپدیت، خود داکیومنت را هم برمی‌گرداند. خوبی این روش نسبت به ترکیب updateOne و سپس findOne این است که اولا دو سری ریکوئست و ریسپانس نداریم، و دوما کل عملیات یک transaction حساب شده و به صورت thread-safe اجرا می‌شود.

db.grades.findAndModify({

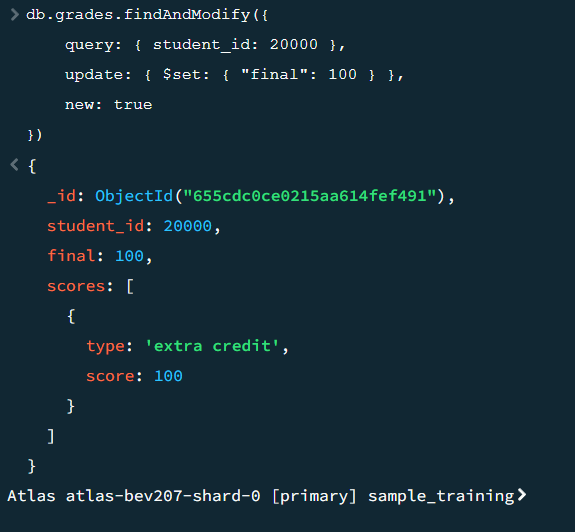
    query: { student\_id: 20000 },

    update: { $set: { "final": 100 } },

    new: true

})

این تابع ابتدا دانشجو با آیدی 20000 را پیدا می‌کند، مقدار final او را به 100 ست کرده و با true کردن آپشن new، داکیومنت آپدیت شده برگردانده می‌شود.



### ١-٤-٥. تابع updateMany

این تابع یک فیلتر و یک آپدیت را می‌گیرد، تمام داکیومنت‌هایی که با آن فیلتر مچ می‌شوند را پیدا کرده و تمامی آن‌ها را با آپدیت داده شده، آپدیت می‌کند.

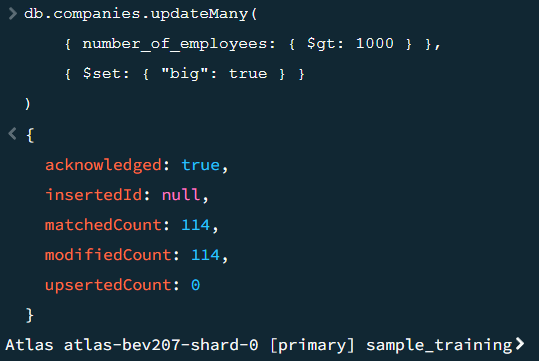
db.companies.updateMany(

    { number\_of\_employees: { $gt: 1000 } },

    { $set: { "big": true } }

)

این دستور کمپانی‌هایی که تعداد کارمندانش بیشتر از 1000 است را پیدا کرده و پارامتر big را برای آن true می‌کند.



## ١-٥. بخش delete

### ١-٥-١. تابع deleteOne

db.zips.find({ zip: "35014" })

db.zips.deleteOne({ \_id: ObjectId("5c8eccc1caa187d17ca6ed16") })

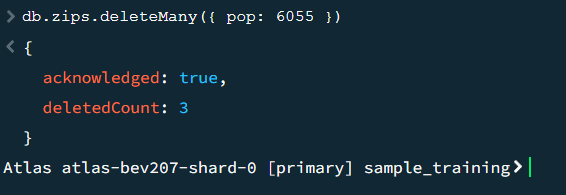
این کوئری اول داکیومنت با زیپ 35014 را پیدا کرده و با استفاده از \_id آن، آن را پاک می‌کند.



### ١-٥-٢. تابع deleteMany

db.zips.deleteMany({ pop: 6055 })

این کوئری تمام zip-هایی که مقدار pop برابر با 6055 دارند را پاک می‌کند.

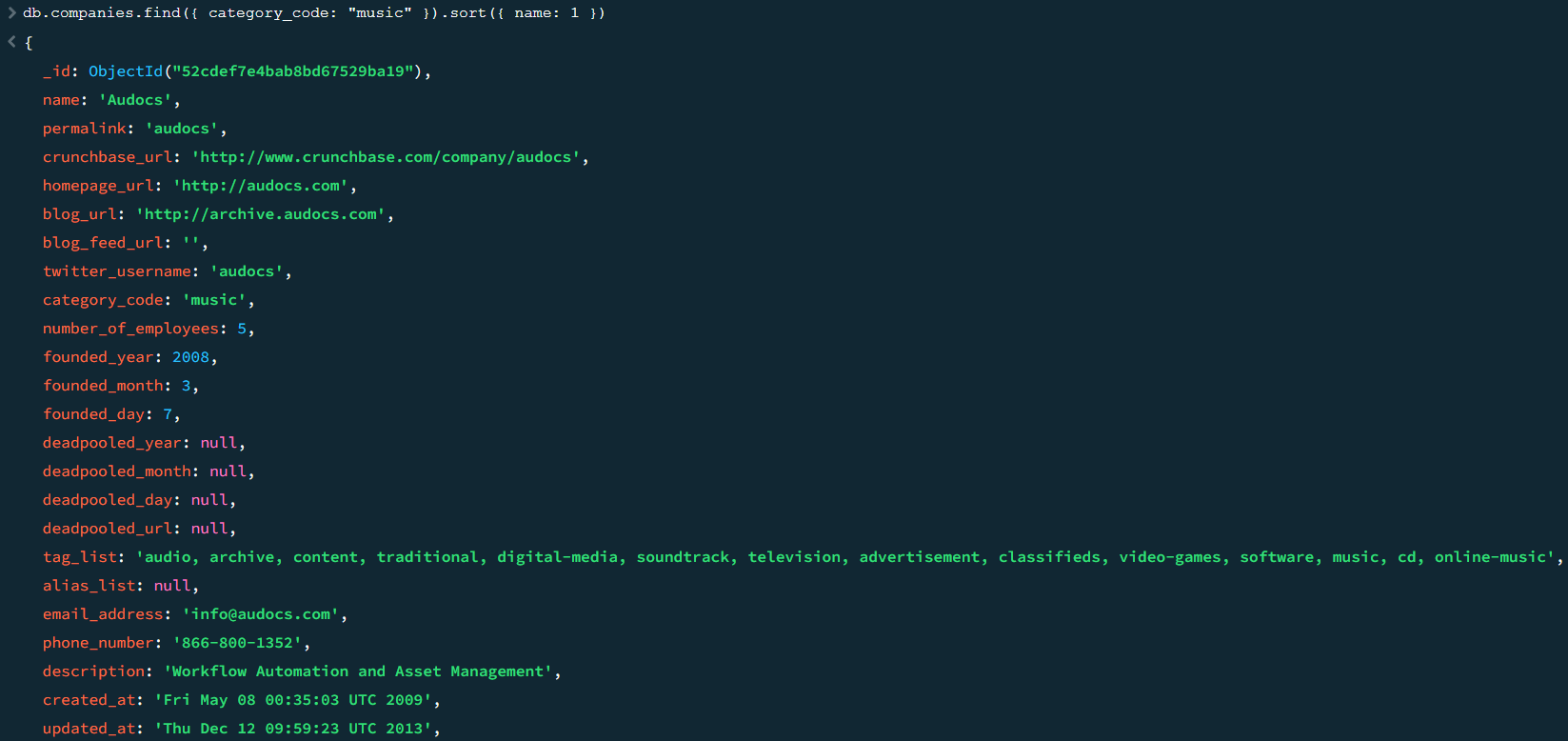


## ١-٦. تغییر نمایش نتایج

### ١-٦-١. تابع sort

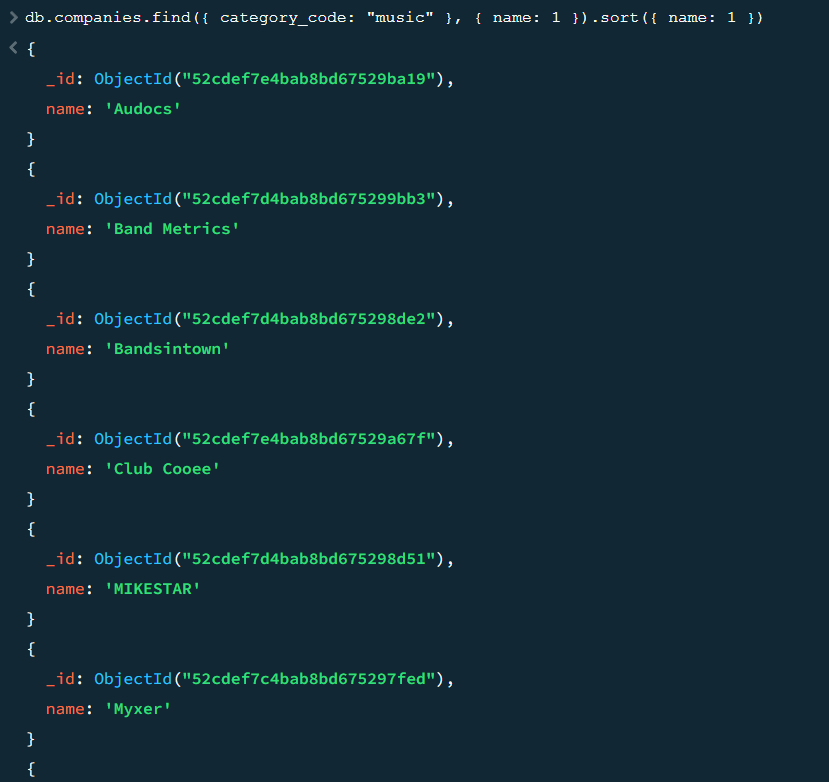
db.companies.find({ category\_code: "music" }).sort({ name: 1 })

این دستور تمام کمپانی‌هایی که در حوزه آهنگ کار می‌کنند را پیدا کرده و بر حسب نام و به صورت صعودی، آن‌ها را مرتب می‌کند. برای مرتب‌سازی به ترتیب نزولی، کافی‌ست مقدار 1 را به مقدار 1- تغییر دهیم.



همچنین می‌توانیم از projection استفاده کنیم:

db.companies.find({ category\_code: "music" }, { name: 1 }).sort({ name: 1 })

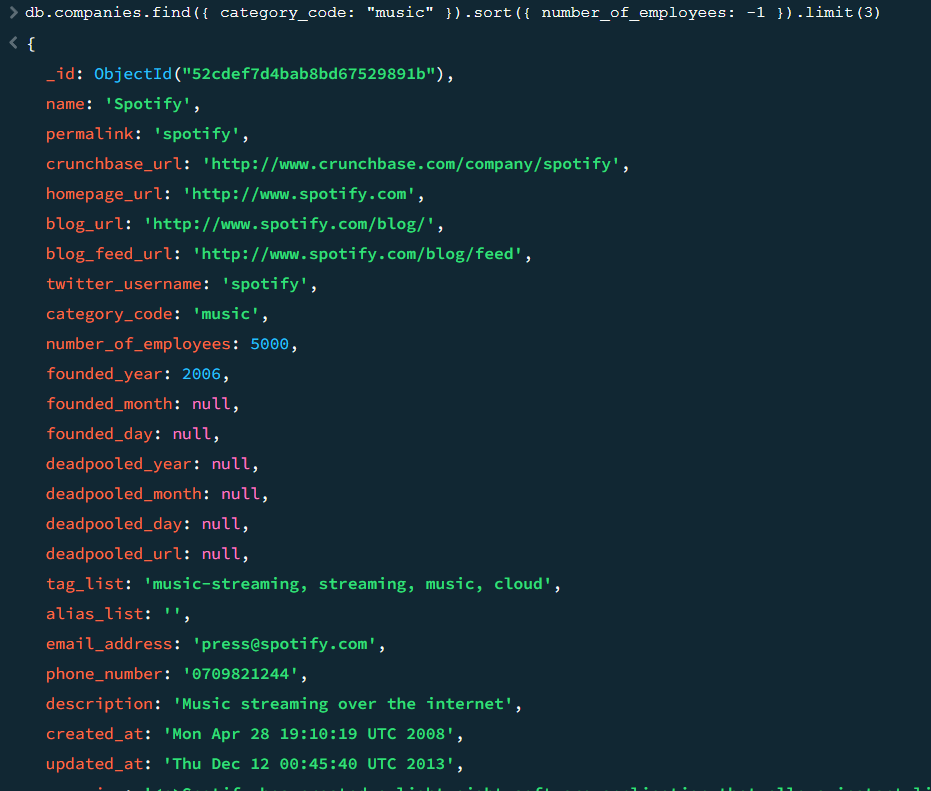


### ١-٦-٢. تابع limit

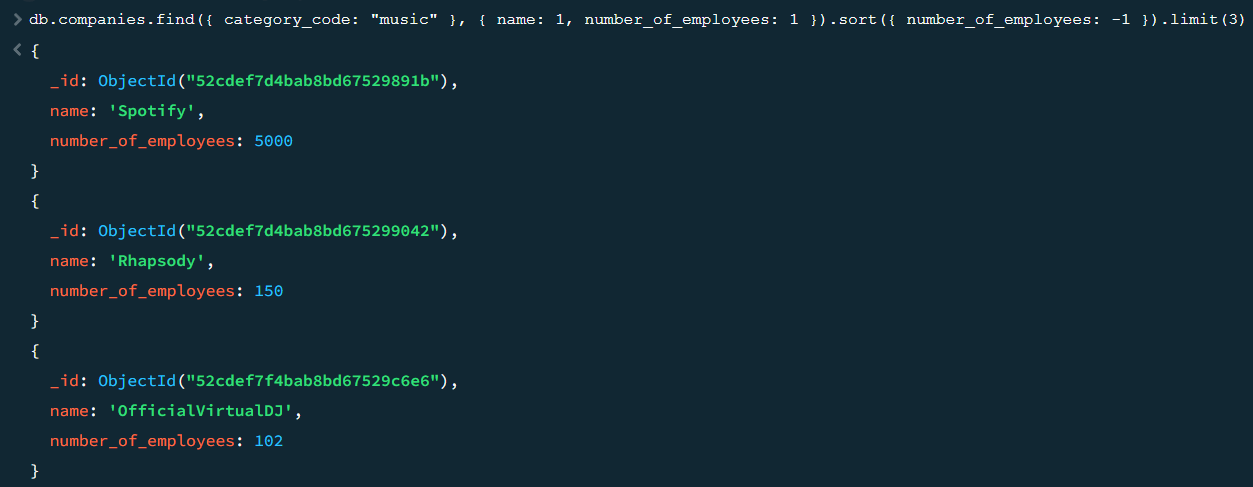
این تابع تعداد نتایج را محدود می‌کند.

db.companies.find({ category\_code: "music" }).sort({ number\_of\_employees: -1 }).limit(3)

در این قسمت 3 کمپانی با بیشترین تعداد کارمند را نشان می‌دهیم:



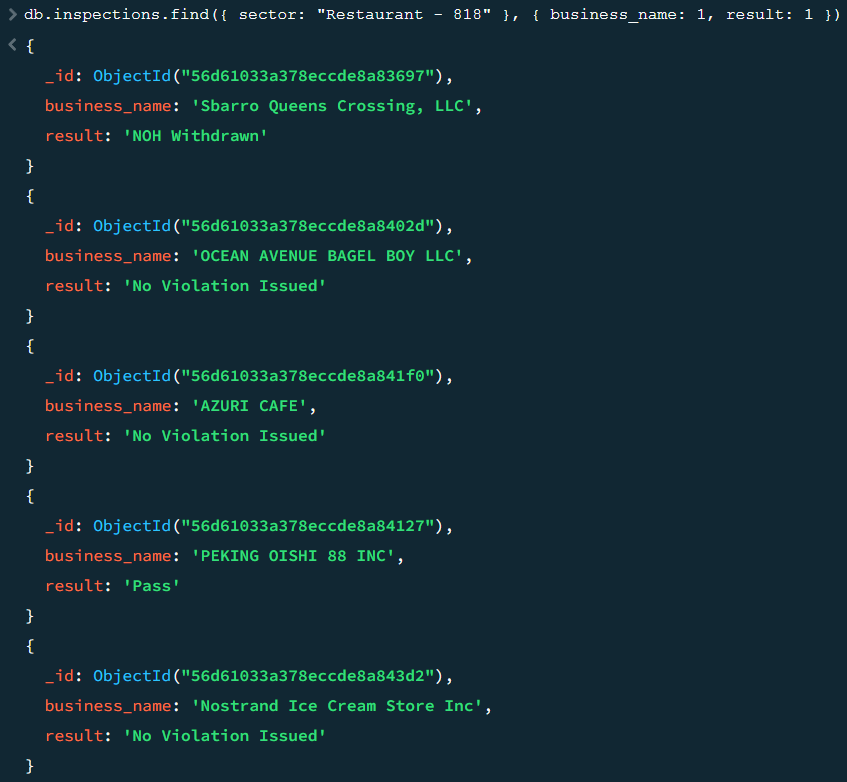
db.companies.find({ category\_code: "music" }, { name: 1, number\_of\_employees: 1 }).sort({ number\_of\_employees: -1 }).limit(3)



### ١-٦-٣. projection

db.inspections.find({ sector: "Restaurant - 818" }, { business\_name: 1, result: 1 })

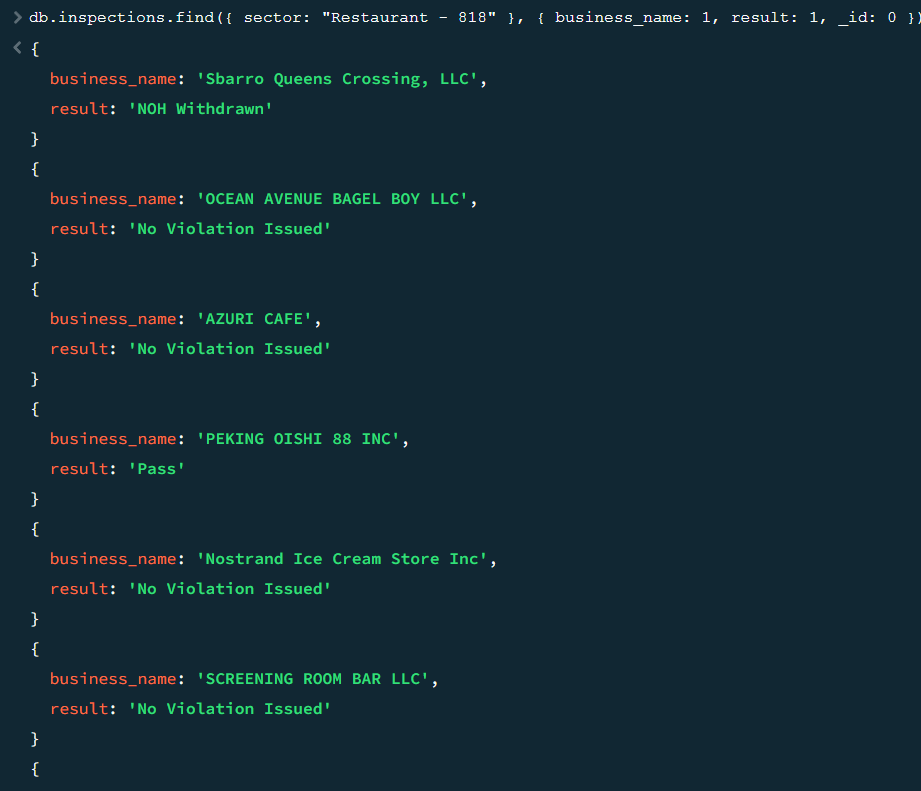
این کوئری تمام inspection-ها که سکتور آن برابر با Restaurant – 818 است را فقط با فیلدهای business\_name و result باز می‌گرداند.



برای حذف \_id می‌توانیم مقدار آن را برابر با 0 قرار دهیم:

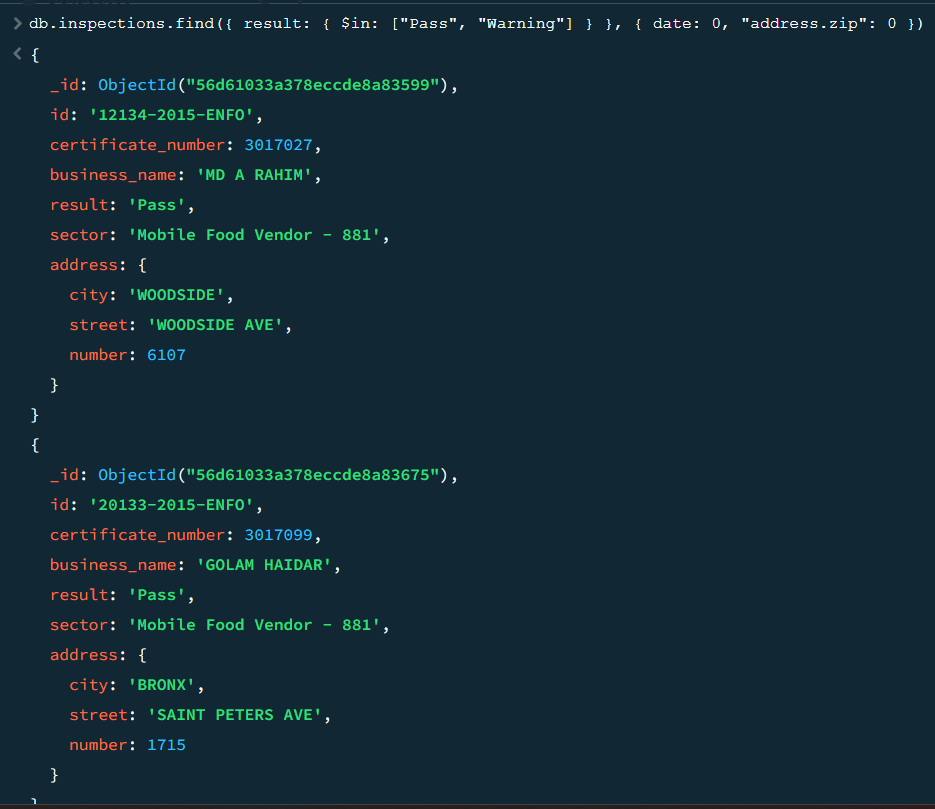
db.inspections.find({ sector: "Restaurant - 818" }, { business\_name: 1, result: 1, \_id: 0 })

نتیجه:



db.inspections.find({ result: { $in: ["Pass", "Warning"] } }, { date: 0, "address.zip": 0 })

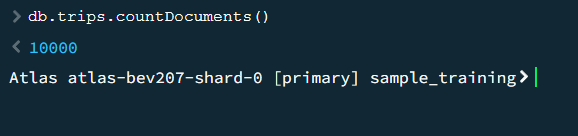
این دستور تمام inspection-هایی که result-اش در بین Pass یا Warning است را به همراه تمام فیلدهایش به جز date و address.zip بازمی‌گرداند.



### ١-۶-٤. تابع count

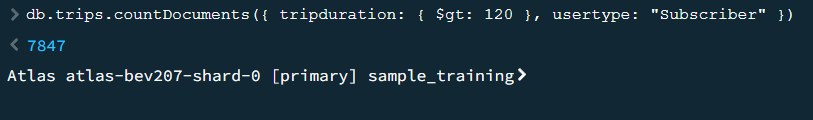
db.trips.countDocuments()

این کوئری تعداد تمام داکیومنت‌های کالکشن trip را برمی‌گرداند.



db.trips.countDocuments({ tripduration: { $gt: 120 }, usertype: "Subscriber" })

این کوئری تعداد تمام trip-هایی که زمان‌شان بیش از 120 دقیقه است و مشتری آن نیز subscriber است را بازمی‌گرداند.



## ١-٧. بخش aggregation

### ١-٧-١. استیج $match

db.zips.aggregate([

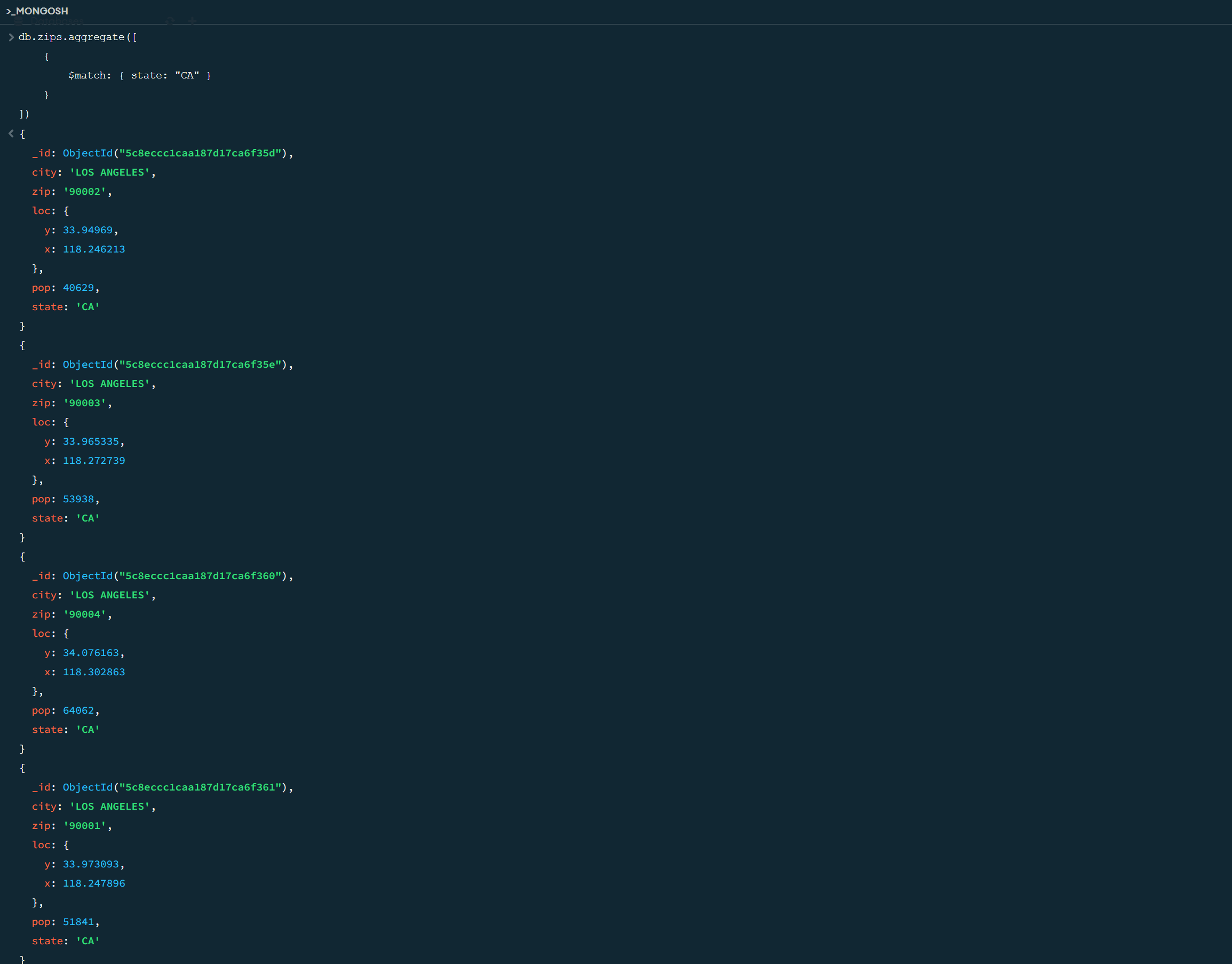
    {

        $match: { state: "CA" }

    }

])

این کوئری تمام zip-هایی که state-شان CA است را بازمی‌گرداند. این کوئری در واقع یک پایپ‌لاین agrregation تک استیجی است.



### ١-٧-٢. استیج $group

هدف این اپراتور، گروه کردن داکیومنت‌هایی است که مقدار یک کلید خاص آن‌ها که در کوئری مشخص شده است، با هم برابر باشد.

db.zips.aggregate([

    {

        $match: { state: "CA"}

    },

    {

        $group: {

            \_id: "$city",

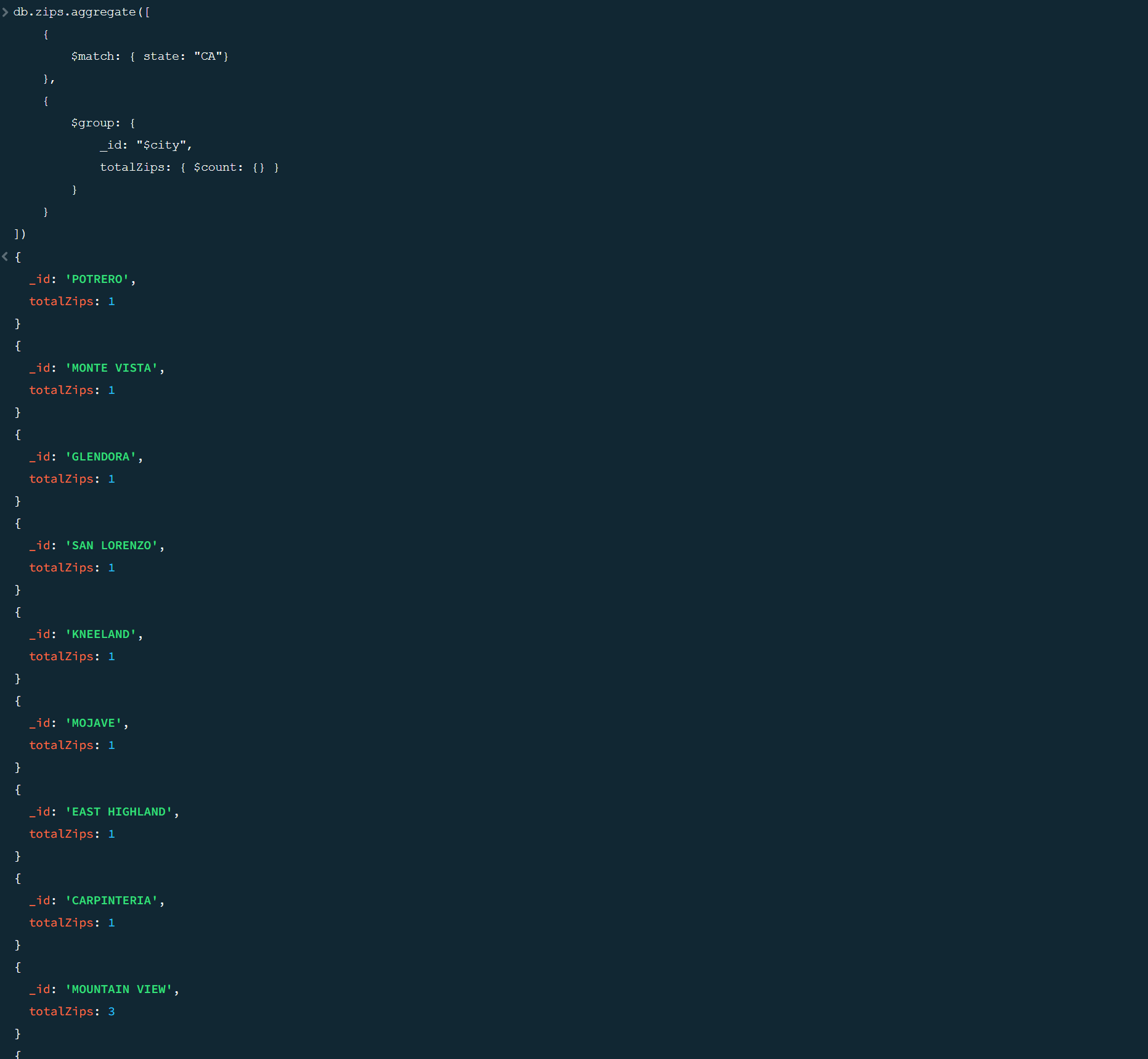
            totalZips: { $count: {} }

        }

    }

])

*این کوئری ابتدا در استیج اول تمام* zip-*هایی که* state-شان CA است را انتخاب کرده، و سپس، به ازای هر city، تعداد داکیومنت‌های آن city را بازمی‌گرداند. نام هر شهر در فیلد \_id نوشته شده است:



### ١-٧-٣. استیج $sort

db.zips.aggregate([

    {

        $sort: {

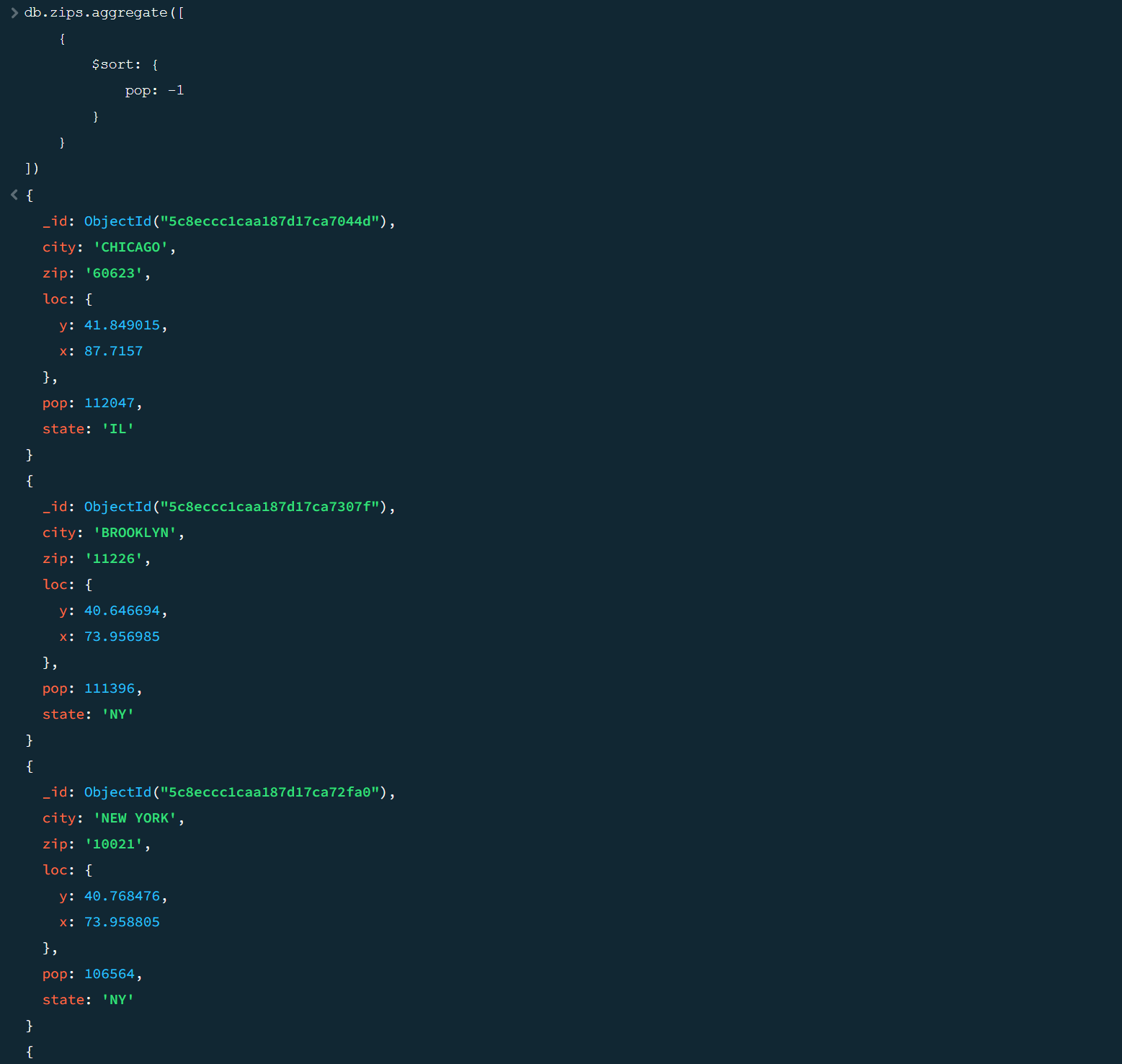
            pop: -1

        }

    }

])

این کوئری در واقع zip-ها را بر اساس جمعیتشان و به صورت نزولی مرتب می‌کند.



### ١-٧-٤. استیج $limit

db.zips.aggregate([

    {

        $sort: {

            pop: -1

        }

    },

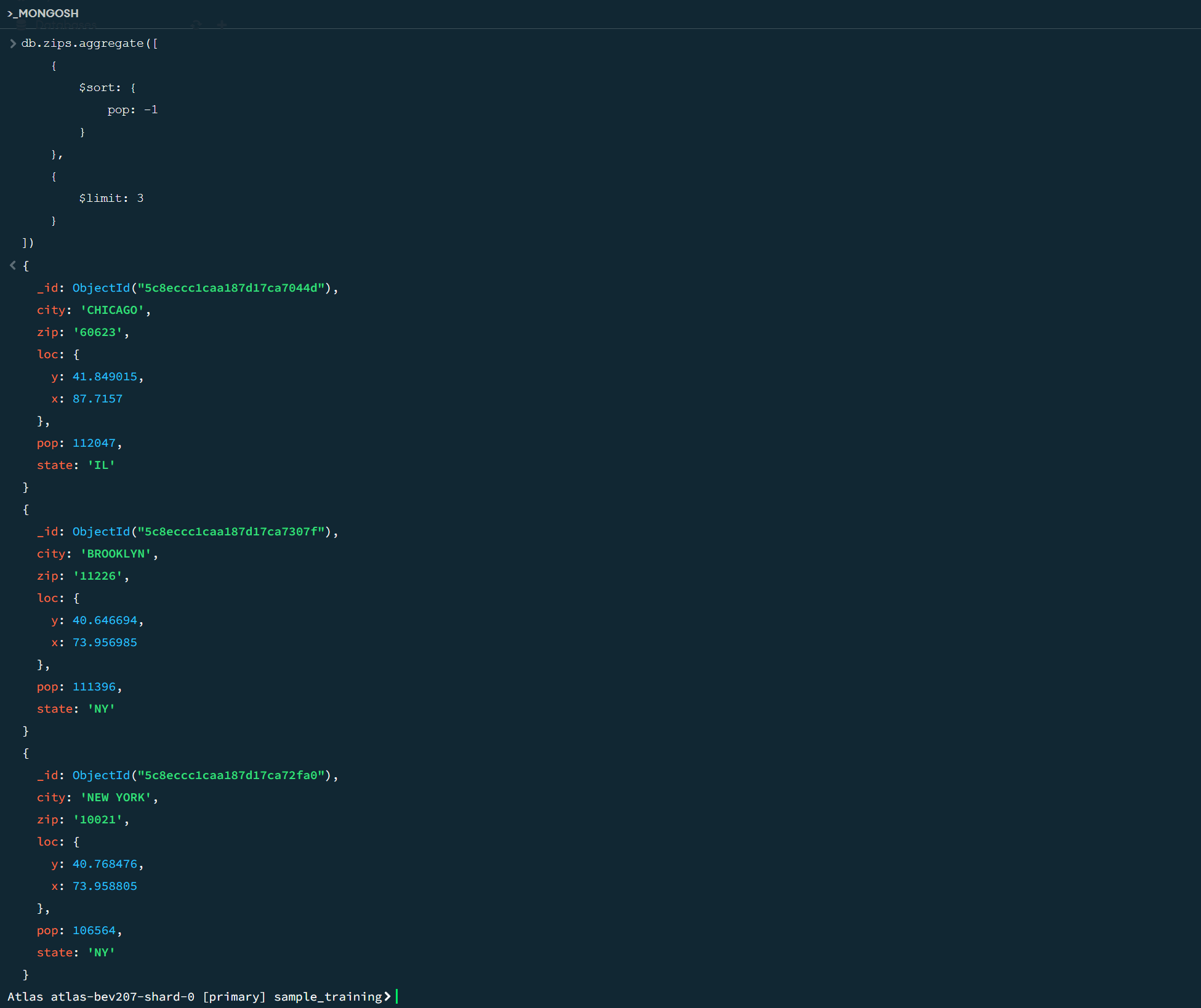
    {

        $limit: 3

    }

])

این مورد مشابه کوئری قبل است با این تفاوت که تنها 3 نتیجه اول را نگه می‌دارد.



### ١-٧-٥. استیج $project

db.zips.aggregate([

    {

        $project: {

            state: 1,

            zip: 1,

            population: "$pop",

            \_id: 0

        }

    }

])

این کوئری در واقع داکیومنت‌های کالکشن zips را گرفته، تنها مقادیر state و zip را نگه داشته، کلید population را به آن اضافه کرده و مقدار آن را برابر با pop قرار داده (در واقع rename کرده) و \_id را از آن حذف کرده است.



### ١-٧-٦. استیج $set

db.zips.aggregate([

    {

        $set: {

            pop\_2022: {

                $round: { $multiply: [1.0031, "$pop"] }

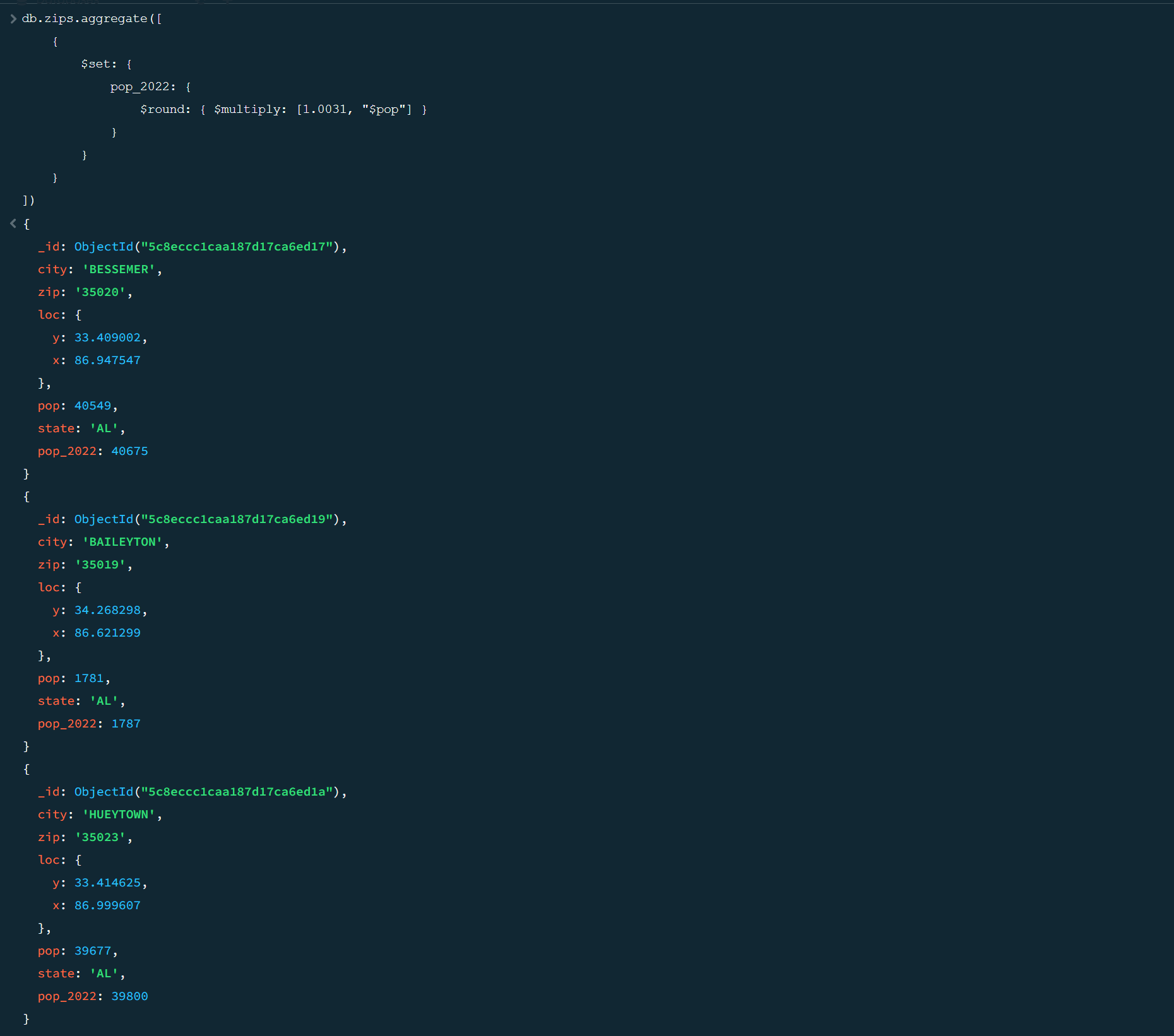
            }

        }

    }

])

توسط این استیج می‌توانیم مقدار یک فیلد جدید را تعریف کنیم. برای مثال، در این بخش، با توجه به مقدار pop، مقدار pop\_2022 ست شده است.



### ١-٧-٧. استیج $count

db.zips.aggregate([

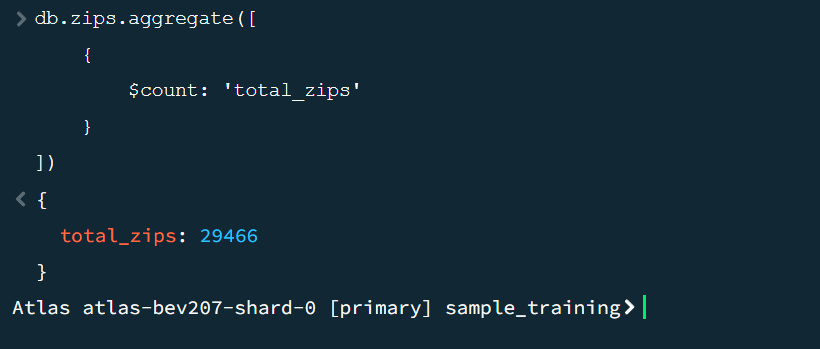
    {

        $count: 'total\_zips'

    }

])

این کوئری نیز تعداد کل zip-ها را برمی‌گرداند.



### ١-٧-٨. استیج $out

db.zips.aggregate([

    {

        $group: {

            \_id: "$state",

            total\_pop: { $sum: "$pop" }

        }

    },

    {

        $match: {

            total\_pop: { $lt: 1000000 }

        }

    },

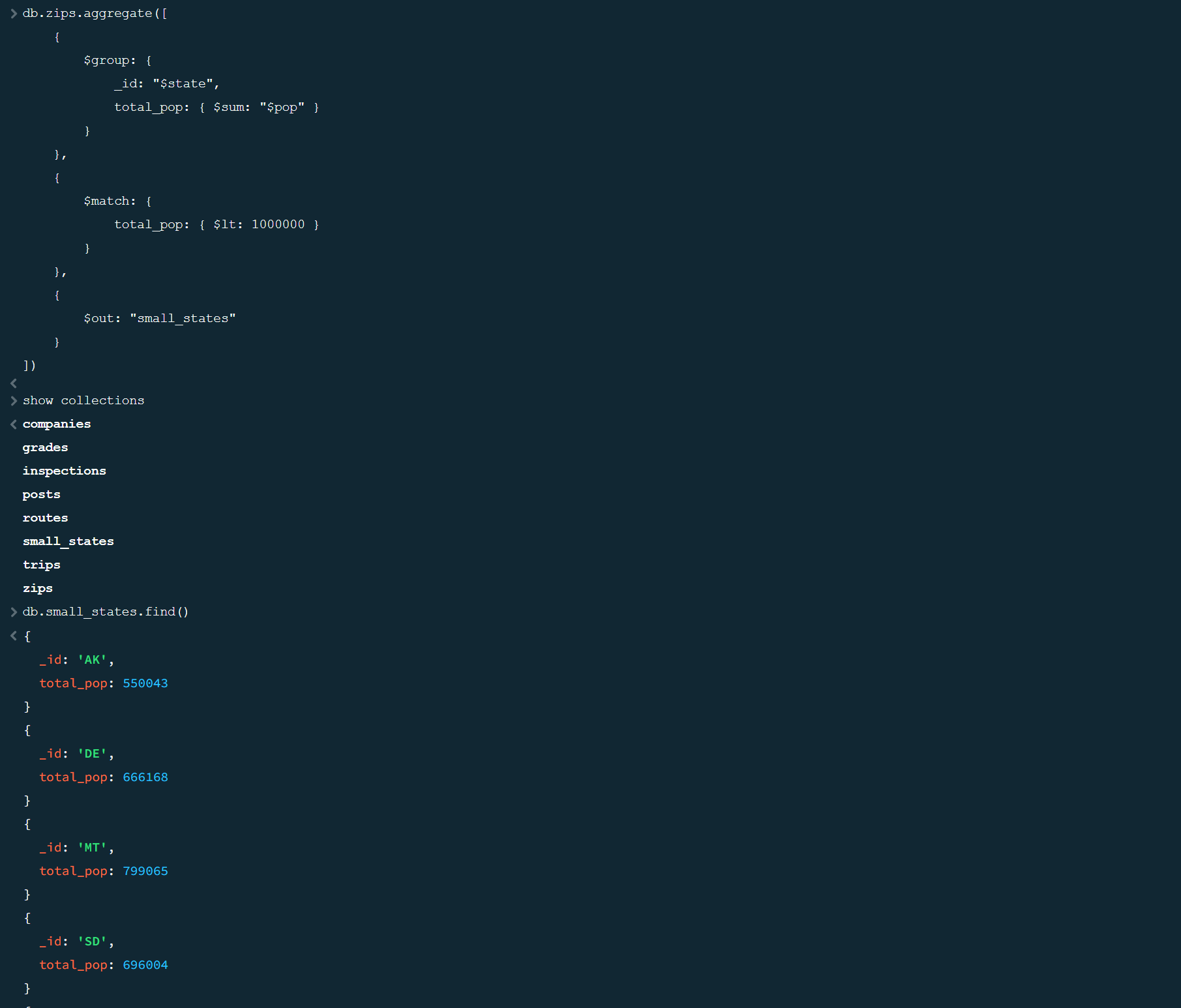
    {

        $out: "small\_states"

    }

])

این کوئری zip-ها را بر اساس state گروه کرده و جمع جمعیت هر state را محاسبه می‌کند. سپس، state-هایی که جمعیتشان کمتر از 1 میلیون است را نگه داشته و آن را در کالکشن small\_states می‌نویسد.



# پاسخ ٢. مقاله سایت مهندسی داده

ابتدا پایگاه داده را ساخته و یک game به آن اضافه می‌کنیم:

// use retrogames

game1 =

{

    name: "Invaders 2013",

    release\_date: new Date(2013, 03, 02),

    categories: ["space", "shooter", "remake"],

    played: false

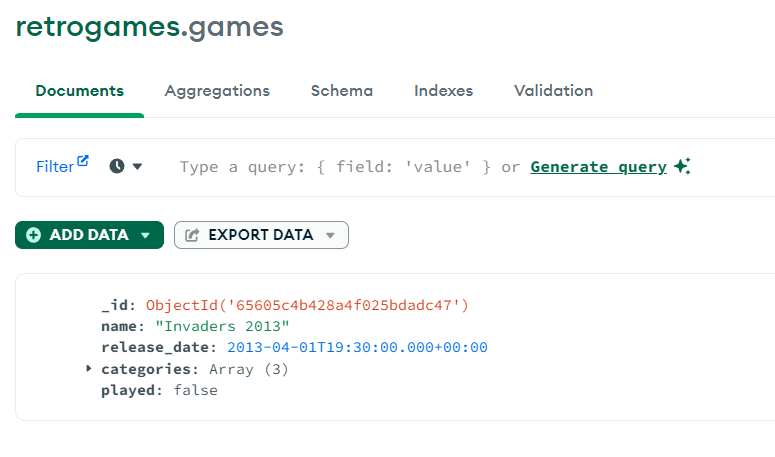
}

db.games.insertOne(game1);

دستور save وجود نداشته و به جای آن از insertOne استفاده شده است.



نتیجه در Compass:



db.games.find()

این دستور تمام game-ها را نشان می‌دهد:



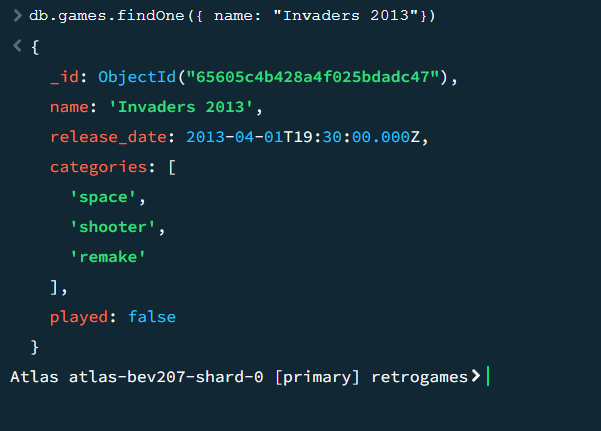
db.games.find().limit(100)

این دستور هم 100 game اول را نشان می‌دهد.



db.games.findOne({ name: "Invaders 2013"})

این دستور به صورت یک فیلتر عمل کرده و تنها game-هایی را نمایش می‌دهد که نامشان به صورت Invaders 2013 باشد.



player1 =

{

    name: "PUZZLEGAMESMASTER",

    gender: "male",

    scores: [

        {

            game\_id: new ObjectId("65605c4b428a4f025bdadc47"),

            game\_name: "Invaders 2013",

            score: 10500,

            score\_date: new Date(2013, 03, 02)

        }

    ]

}

db.players.insertOne(player1)

با این دستور کالکشن players ایجاد شده و یک داکیومنت به آن افزوده می‌شود.



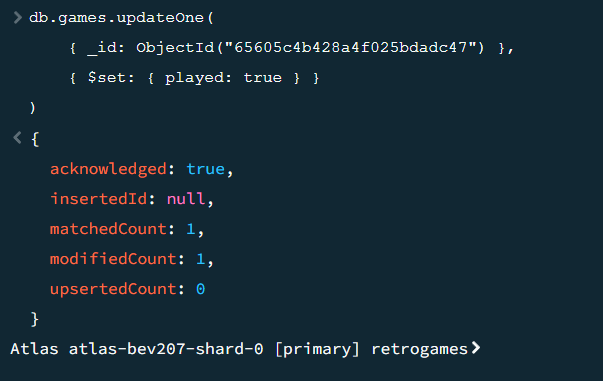
db.games.updateOne(

    { \_id: ObjectId("65605c4b428a4f025bdadc47") },

    { $set: { played: true } }

)

این دستور هم مقدار فیلد played در game ایجاد شده را true می‌کند.



db.players.update(

    { \_id: new ObjectId("65605fe8428a4f025bdadc49") },

    {

        $push: {

            scores: {

                game\_id: new ObjectId("51e10c50085977bc3cd92a65"),

                game\_name: "Invaders 2013",

                score: 30250,

                score\_date: new Date(2013, 03, 03)

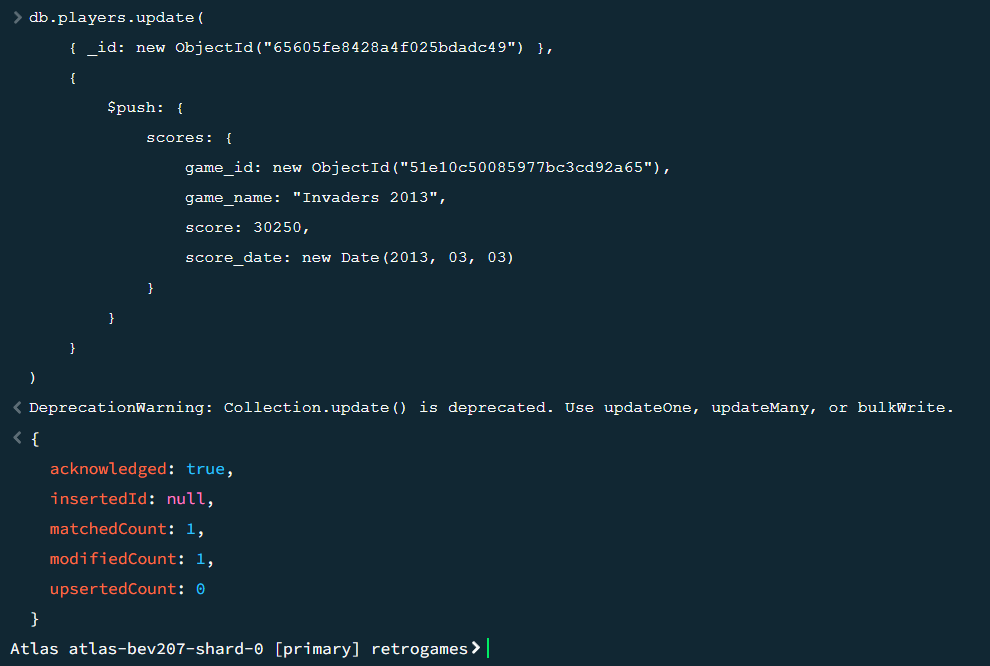
            }

        }

    }

)

این دستور player قبلی را آپدیت کرده با استفاده از اپراتور $push، یک score به scores آن اضافه می‌کند.



db.players.findOne()

نتیجه:



همچنین می‌توانستیم از اپراتور $addToSet نیز استفاده کنیم که مشابه $push است با این تفاوت که تکراری نبودن المان‌ها را نیز بررسی می‌کند.