به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس آزمایشگاه پایگاه‌داده

پیش‌گزارش پنجم

نام و نام خانوادگی	پاشا براهیمی
شماره دانشجویی	۸۱۰۱۹۹۳۸۵
تاریخ ارسال گزارش	۱۴۰۲/۰۸/۲۷

فهرست

پاسخ ۱. آموزش‌های وبسایت اصلی MongoDB

۱-۱. قسمت insert

۱-۱-۱. 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.

```
{
  acknowledged: true,
  insertedId: ObjectId("65539d5b4b5772f36843f6bc")
}
Atlas atlas-bev207-shard-0 [primary] test>
```

۱-۱-۲. 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.

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("65539f204b5772f36843f6bd"),
    '1': ObjectId("65539f204b5772f36843f6be"),
    '2': ObjectId("65539f204b5772f36843f6bf")
  }
}
```

۱-۲. قسمت find

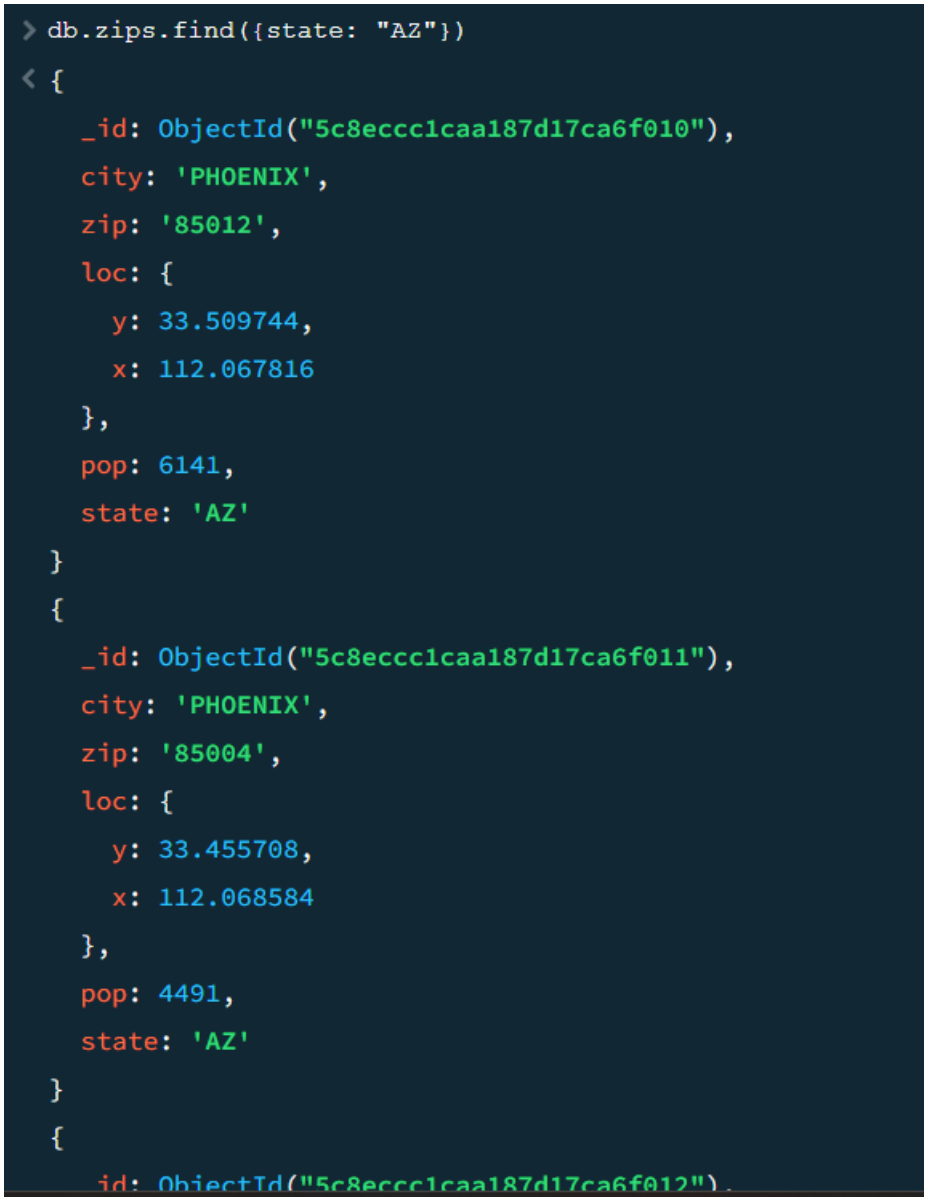
دستور find و اپراتور $eq

```
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"})
```

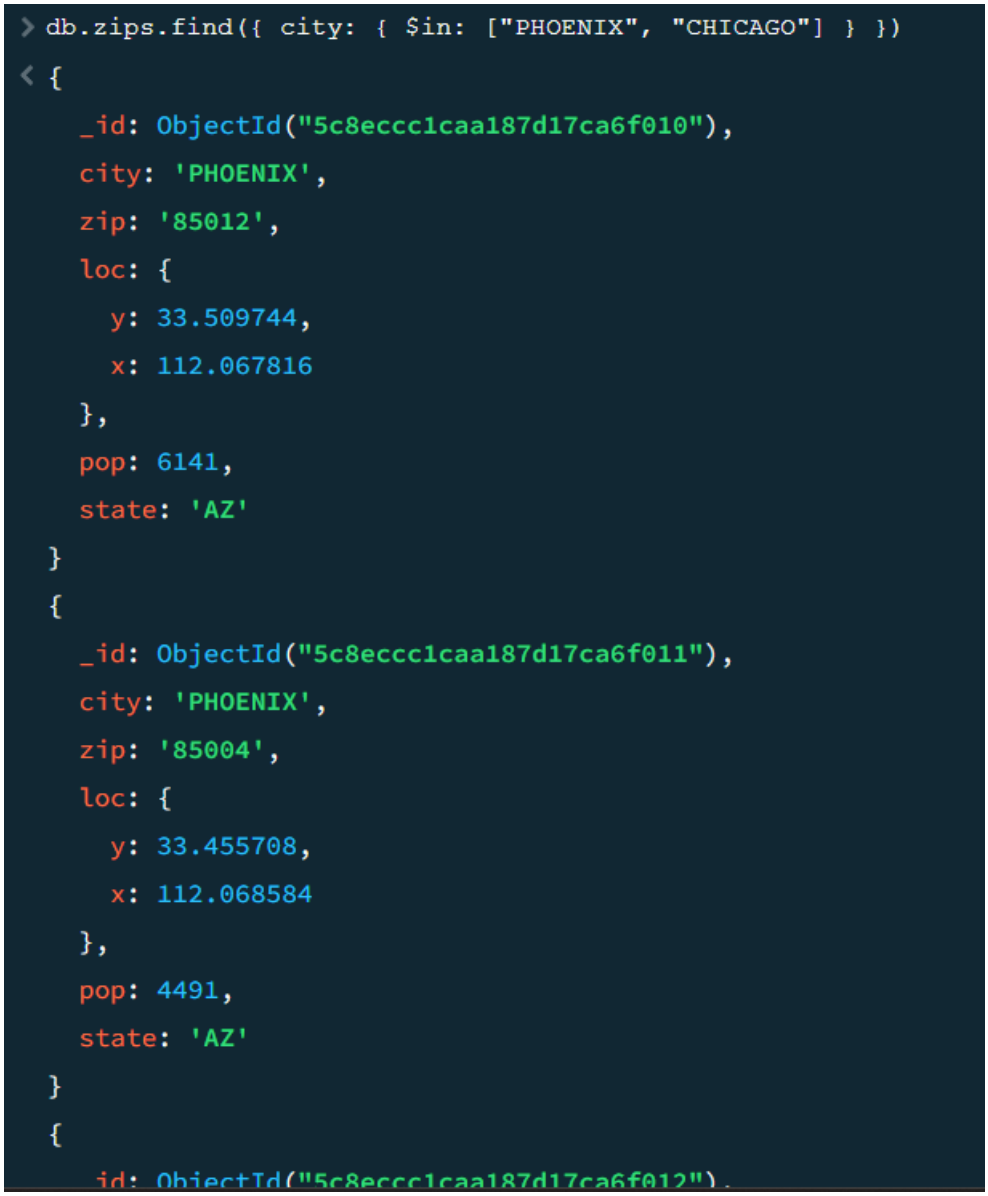
در این حالت، تمامی داکیومنت‌های zip که state آن‌ها آریزونا است برگردانده می‌شود. می‌توانیم از اپراتور $eq هم استفاده کنیم.



۱-۲-۲. 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.



۱-۲-۳. 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.

```
> db.zips.findOne()
< {
    _id: ObjectId("5c8eccc1caa187d17ca6ed16"),
    city: 'ALPINE',
    zip: '35014',
    loc: {
      y: 33.331165,
      x: 86.208934
    },
    pop: 3062,
    state: 'AL'
  }
```

۱-۲-۴. 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.

۱-۲-۵. 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.accounts.find({ products: { $elemMatch: { $eq: "InvestmentStock" } } })
< {
    _id: ObjectId("5ca4bbc7a2dd94ee5816238c"),
    account_id: 371138,
    limit: 9000,
    products: [
      'Derivatives',
      'InvestmentStock'
    ]
  }
  {
    _id: ObjectId("5ca4bbc7a2dd94ee5816238d"),
    account_id: 557378,
    limit: 10000,
    products: [
      'InvestmentStock',
      'Commodity',
      'Brokerage',
      'CurrencyService'
    ]
  }
  {
    _id: ObjectId("5ca4bbc7a2dd94ee5816238e"),
    account_id: 198100,
    limit: 10000,
```

```
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.

```
> db.sales.find({
      items: {
          $elemMatch: { name: "laptop", price: { $gt: 800 }, quantity: { $gte: 1 } }
      }
  })
< {
    _id: ObjectId("5bd761dcae323e45a93ccfe9"),
    saleDate: 2015-08-25T10:01:02.918Z,
    items: [
      {
        name: 'envelopes',
        tags: [
          'stationary',
          'office',
          'general'
        ],
        price: Decimal128("8.05"),
        quantity: 10
      },
      {
        name: 'binder',
        tags: [
          'school',
          'general',
          'organization'
```

۱-۲-۶. 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.

```
> db.routes.find({
      $and: [{ "airline.name": "Southwest Airlines" }, { "stops": { $gte: 1 } }]
  })
< {
    _id: ObjectId("56e9b39c732b6122f878f280"),
    airline: {
      id: 4547,
      name: 'Southwest Airlines',
      alias: 'WN',
      iata: 'SWA'
    },
    src_airport: 'BOS',
    dst_airport: 'MCO',
    codeshare: '',
    stops: 1,
    airplane: '73W'
  }
  {
    _id: ObjectId("56e9b39c732b6122f878f45b"),
    airline: {
      id: 4547,
      name: 'Southwest Airlines',
      alias: 'WN',
      iata: 'SWA'
    },
```

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

```
db.routes.find({
    "airline.name": "Southwest Airlines", "stops": { $gte: 1 }
})
```

```
> db.routes.find({
      "airline.name": "Southwest Airlines", "stops": { $gte: 1 }
  })
< {
    _id: ObjectId("56e9b39c732b6122f878f280"),
    airline: {
      id: 4547,
      name: 'Southwest Airlines',
      alias: 'WN',
      iata: 'SWA'
    },
    src_airport: 'BOS',
    dst_airport: 'MCO',
    codeshare: '',
    stops: 1,
    airplane: '73W'
  }
  {
    _id: ObjectId("56e9b39c732b6122f878f45b"),
    airline: {
      id: 4547,
      name: 'Southwest Airlines',
      alias: 'WN',
      iata: 'SWA'
    },
```

۱-۲-۷. 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.

```
> db.routes.find({
      $or: [{ "dst_airport": "SEA" }, { "src_airport": "SEA" }]
  })
< {
    _id: ObjectId("56e9b39b732b6122f8780cfc"),
    airline: {
      id: 24,
      name: 'American Airlines',
      alias: 'AA',
      iata: 'AAL'
    },
    src_airport: 'BOS',
    dst_airport: 'SEA',
    codeshare: 'Y',
    stops: 0,
    airplane: 737
  }
  {
    _id: ObjectId("56e9b39b732b6122f8780d0f"),
    airline: {
      id: 24,
      name: 'American Airlines',
      alias: 'AA',
      iata: 'AAL'
    },
```

۱-۲-۸. 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.

```
> db.routes.find({
      $and: [
          { $or: [{ "dst_airport": "SEA" }, { "src_airport": "SEA" }] },
          { $or: [{ "airline.name": "American Airlines" }, { airplane: 320 }] }
      ]
  })
< {
    _id: ObjectId("56e9b39b732b6122f8780cfc"),
    airline: {
      id: 24,
      name: 'American Airlines',
      alias: 'AA',
      iata: 'AAL'
    },
    src_airport: 'BOS',
    dst_airport: 'SEA',
    codeshare: 'Y',
    stops: 0,
    airplane: 737
  }
  {
    _id: ObjectId("56e9b39b732b6122f8780d0f"),
    airline: {
      id: 24,
      name: 'American Airlines',
```

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.

۱-۳. قسمت replace

۱-۳-۱. 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.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")
      }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

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

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

```
> db.comments.findOne({ _id: ObjectId("5a9427648b0beebeb69579e7") })
< {
    _id: ObjectId("5a9427648b0beebeb69579e7"),
    name: 'Mercedes Tyler',
    email: 'mercedes_tyler@fakegmail.com',
    movie_id: ObjectId("573a1390f29313caabcd4323"),
    text: 'Test',
    date: 2002-08-18T04:56:07.000Z
  }
Atlas atlas-bev207-shard-0 [primary] sample_mflix>
```

۱-۴. بخش update

۱-۴-۱. 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.updateOne(
      { _id: ObjectId("56d5f7eb604eb380b0d8d8ce") },
      { $set: { "final": 67.6 } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

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

```
> db.grades.findOne({ _id: ObjectId("56d5f7eb604eb380b0d8d8ce") })
< {
    _id: ObjectId("56d5f7eb604eb380b0d8d8ce"),
    student_id: 0,
    scores: [
      {
        type: 'exam',
        score: 78.40446309504266
      },
      {
        type: 'quiz',
        score: 73.36224783231339
      },
      {
        type: 'homework',
        score: 46.980982486720535
      },
      {
        type: 'homework',
        score: 76.67556138656222
      }
    ],
    class_id: 339,
    final: 67.6
  }
```

۱-۴-۲. 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 }
)
```

```
> db.grades.updateOne(
      { student_id: 20000 },
      { $set: { "final": 88 } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
> db.grades.updateOne(
      { student_id: 20000 },
      { $set: { "final": 88 } },
      { upsert: true }
  )
< {
    acknowledged: true,
    insertedId: ObjectId("655cdc0ce0215aa614fef491"),
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 1
  }
```

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

۱-۴-۳. 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 } }
}
)
```

```
> db.grades.updateOne(
      { student_id: 20000 },
      { $push: { "scores": { "type": "extra credit", "score": 100 } } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

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

```
db.grades.findOne({ student_id: 20000 })
```

```
> db.grades.findOne({ student_id: 20000 })
< {
    _id: ObjectId("655cdc0ce0215aa614fef491"),
    student_id: 20000,
    final: 88,
    scores: [
      {
        type: 'extra credit',
        score: 100
      }
    ]
  }
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

۱-۴-۴. 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.

```
> db.grades.findAndModify({
      query: { student_id: 20000 },
      update: { $set: { "final": 100 } },
      new: true
  })
< {
    _id: ObjectId("655cdc0ce0215aa614fef491"),
    student_id: 20000,
    final: 100,
    scores: [
      {
        type: 'extra credit',
        score: 100
      }
    ]
  }
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

۱-۴-۵. 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.

```
> db.companies.updateMany(
      { number_of_employees: { $gt: 1000 } },
      { $set: { "big": true } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 114,
    modifiedCount: 114,
    upsertedCount: 0
  }
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

۱-۵. بخش delete

۱-۵-۱. 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.

```
> db.zips.find({ zip: "35014" })
< {
    _id: ObjectId("5c8eccc1caa187d17ca6ed16"),
    city: 'ALPINE',
    zip: '35014',
    loc: {
      y: 33.331165,
      x: 86.208934
    },
    pop: 3062,
    state: 'AL'
  }
> db.zips.deleteOne({ _id: ObjectId("5c8eccc1caa187d17ca6ed16") })
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.zips.find({ zip: "35014" })
<
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

۱-۵-۲. تابع deleteMany

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

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

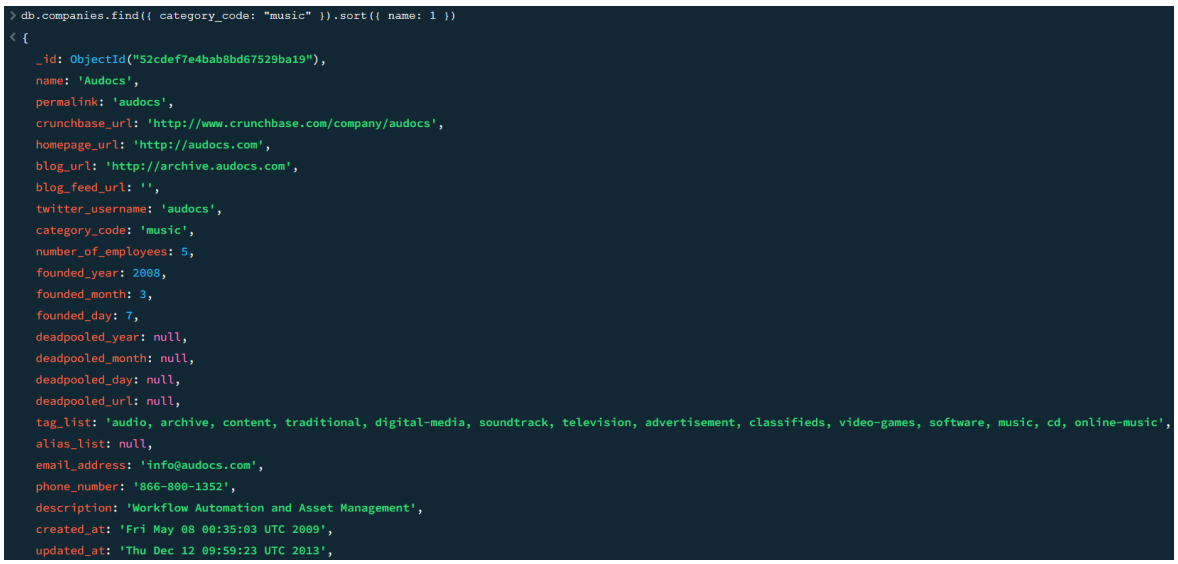
```
> db.zips.deleteMany({ pop: 6055 })
< {
    acknowledged: true,
    deletedCount: 3
  }
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

۱-۶. تغییر نمایش نتایج

۱-۶-۱. 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 })
```

```
> db.companies.find({ category_code: "music" }, { name: 1 }).sort({ name: 1 })
< {
    _id: ObjectId("52cdef7e4bab8bd67529ba19"),
    name: 'Audocs'
  }
  {
    _id: ObjectId("52cdef7d4bab8bd675299bb3"),
    name: 'Band Metrics'
  }
  {
    _id: ObjectId("52cdef7d4bab8bd675298de2"),
    name: 'Bandsintown'
  }
  {
    _id: ObjectId("52cdef7e4bab8bd67529a67f"),
    name: 'Club Cooee'
  }
  {
    _id: ObjectId("52cdef7d4bab8bd675298d51"),
    name: 'MIKESTAR'
  }
  {
    _id: ObjectId("52cdef7c4bab8bd675297fed"),
    name: 'Myxer'
  }
  {
```

۱-۶-۲. 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" }).sort({ number_of_employees: -1 }).limit(3)
< {
    _id: ObjectId("52cdef7d4bab8bd67529891b"),
    name: 'Spotify',
    permalink: 'spotify',
    crunchbase_url: 'http://www.crunchbase.com/company/spotify',
    homepage_url: 'http://www.spotify.com',
    blog_url: 'http://www.spotify.com/blog/',
    blog_feed_url: 'http://www.spotify.com/blog/feed',
    twitter_username: 'spotify',
    category_code: 'music',
    number_of_employees: 5000,
    founded_year: 2006,
    founded_month: null,
    founded_day: null,
    deadpooled_year: null,
    deadpooled_month: null,
    deadpooled_day: null,
    deadpooled_url: null,
    tag_list: 'music-streaming, streaming, music, cloud',
    alias_list: '',
    email_address: 'press@spotify.com',
    phone_number: '0709821244',
    description: 'Music streaming over the internet',
    created_at: 'Mon Apr 28 19:10:19 UTC 2008',
    updated_at: 'Thu Dec 12 00:45:40 UTC 2013',
```

```
db.companies.find({ category_code: "music" }, { name: 1,
number_of_employees: 1 }).sort({ number_of_employees: -1
}).limit(3)
```

```
> db.companies.find({ category_code: "music" }, { name: 1, number_of_employees: 1 }).sort({ number_of_employees: -1 }).limit(3)
< {
    _id: ObjectId("52cdef7d4bab8bd67529891b"),
    name: 'Spotify',
    number_of_employees: 5000
  }
  {
    _id: ObjectId("52cdef7d4bab8bd675299042"),
    name: 'Rhapsody',
    number_of_employees: 150
  }
  {
    _id: ObjectId("52cdef7f4bab8bd67529c6e6"),
    name: 'OfficialVirtualDJ',
    number_of_employees: 102
  }
```

۱-۶-۳. 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.

```
> db.inspections.find({ sector: "Restaurant - 818" }, { business_name: 1, result: 1 })
< {
    _id: ObjectId("56d61033a378eccde8a83697"),
    business_name: 'Sbarro Queens Crossing, LLC',
    result: 'NOH Withdrawn'
  }
  {
    _id: ObjectId("56d61033a378eccde8a8402d"),
    business_name: 'OCEAN AVENUE BAGEL BOY LLC',
    result: 'No Violation Issued'
  }
  {
    _id: ObjectId("56d61033a378eccde8a841f0"),
    business_name: 'AZURI CAFE',
    result: 'No Violation Issued'
  }
  {
    _id: ObjectId("56d61033a378eccde8a84127"),
    business_name: 'PEKING OISHI 88 INC',
    result: 'Pass'
  }
  {
    _id: ObjectId("56d61033a378eccde8a843d2"),
    business_name: 'Nostrand Ice Cream Store Inc',
    result: 'No Violation Issued'
  }
```

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({ sector: "Restaurant - 818" }, { business_name: 1, result: 1, _id: 0 })
< {
    business_name: 'Sbarro Queens Crossing, LLC',
    result: 'NOH Withdrawn'
  }
  {
    business_name: 'OCEAN AVENUE BAGEL BOY LLC',
    result: 'No Violation Issued'
  }
  {
    business_name: 'AZURI CAFE',
    result: 'No Violation Issued'
  }
  {
    business_name: 'PEKING OISHI 88 INC',
    result: 'Pass'
  }
  {
    business_name: 'Nostrand Ice Cream Store Inc',
    result: 'No Violation Issued'
  }
  {
    business_name: 'SCREENING ROOM BAR LLC',
    result: 'No Violation Issued'
  }
  {
```

```
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.

```
> db.inspections.find({ result: { $in: ["Pass", "Warning"] } }, { date: 0, "address.zip": 0 })
< {
    _id: ObjectId("56d61033a378eccde8a83599"),
    id: '12134-2015-ENFO',
    certificate_number: 3017027,
    business_name: 'MD A RAHIM',
    result: 'Pass',
    sector: 'Mobile Food Vendor - 881',
    address: {
      city: 'WOODSIDE',
      street: 'WOODSIDE AVE',
      number: 6107
    }
  }
  {
    _id: ObjectId("56d61033a378eccde8a83675"),
    id: '20133-2015-ENFO',
    certificate_number: 3017099,
    business_name: 'GOLAM HAIDAR',
    result: 'Pass',
    sector: 'Mobile Food Vendor - 881',
    address: {
      city: 'BRONX',
      street: 'SAINT PETERS AVE',
      number: 1715
    }
```

۱-۶-۴. تابع count

```
db.trips.countDocuments()
```

این کوئری تعداد تمام داکیومنت‌های کالکشن trip را برمی‌گرداند.

```
> db.trips.countDocuments()
< 10000
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

```
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.

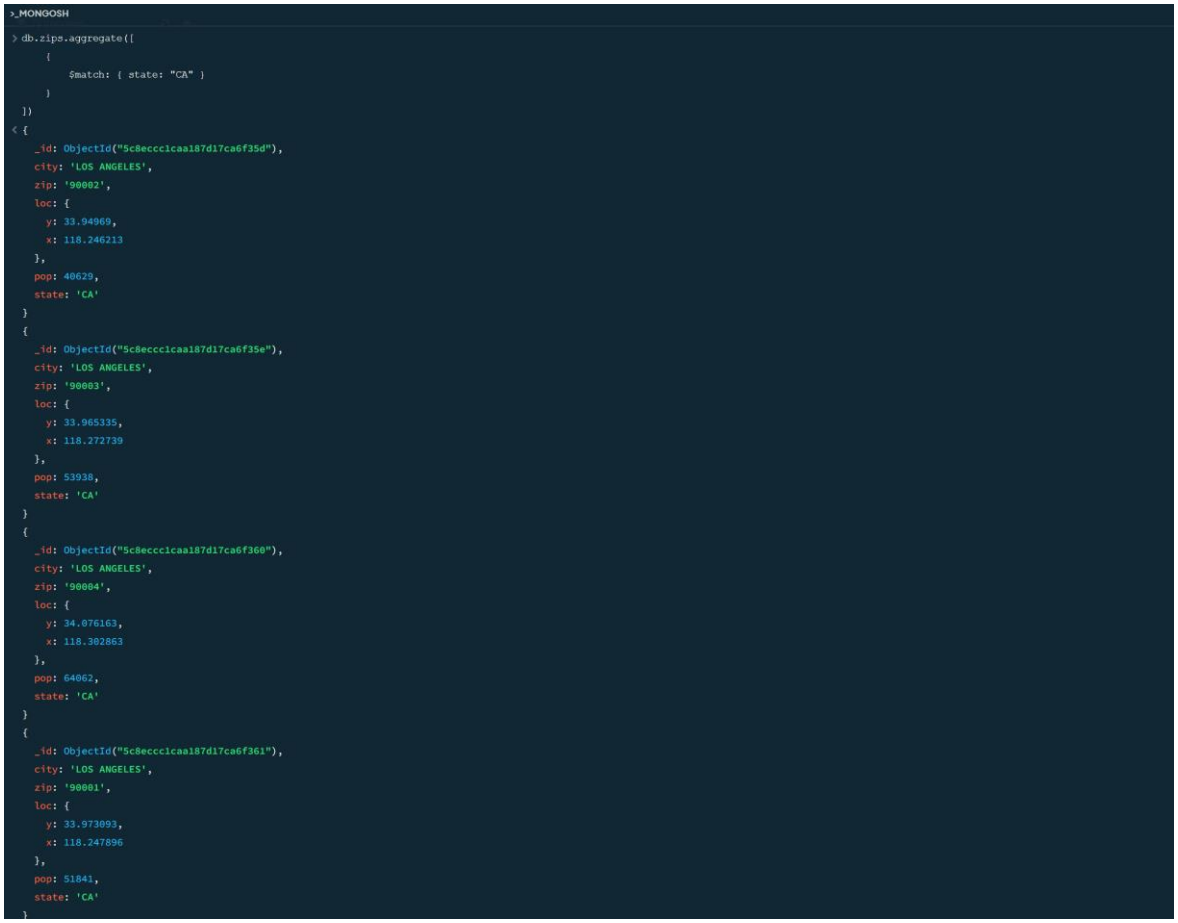
```
> db.trips.countDocuments({ tripduration: { $gt: 120 }, usertype: "Subscriber" })
< 7847
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

۱-۷. بخش aggregation

۱-۷-۱. 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.



۱-۷-۲. 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:

```
> db.zips.aggregate([
    {
        $match: { state: "CA"}
    },
    {
        $group: {
            _id: "$city",
            totalZips: { $count: {} }
        }
    }
])
< {
    _id: 'POTRERO',
    totalZips: 1
  }
  {
    _id: 'MONTE VISTA',
    totalZips: 1
  }
  {
    _id: 'GLENDORA',
    totalZips: 1
  }
  {
    _id: 'SAN LORENZO',
    totalZips: 1
  }
  {
    _id: 'KNEELAND',
    totalZips: 1
  }
  {
    _id: 'MOJAVE',
    totalZips: 1
  }
  {
    _id: 'EAST HIGHLAND',
    totalZips: 1
  }
  {
    _id: 'CARPINTERIA',
    totalZips: 1
  }
  {
    _id: 'MOUNTAIN VIEW',
    totalZips: 3
  }
  {
```

۱-۷-۳. استیج $sort

```
db.zips.aggregate([
    {
        $sort: {
```

```
            pop: -1
        }
    }
])
```

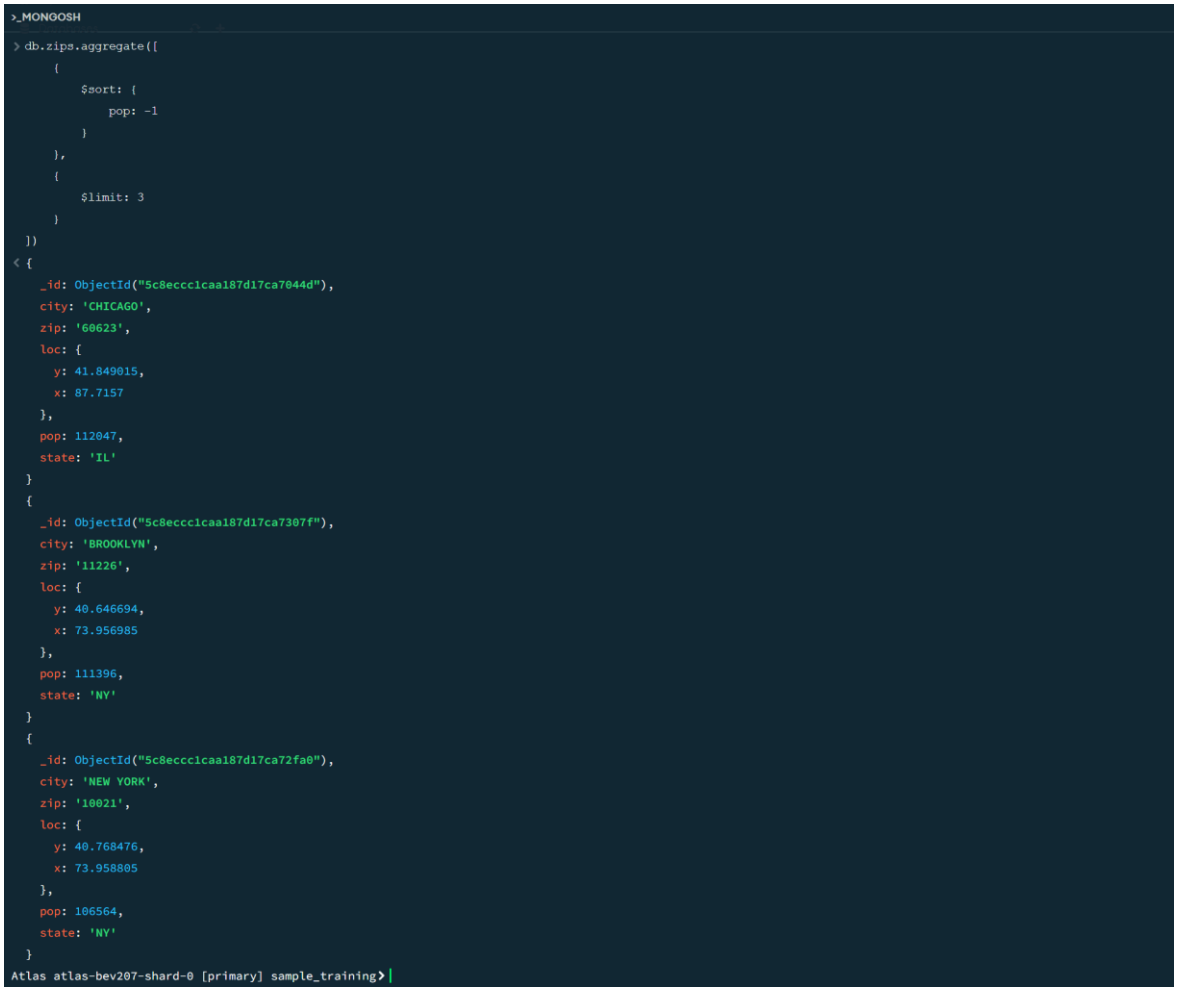
این کوئری در واقع zip-ها را بر اساس جمعیتشان و به صورت نزولی مرتب می‌کند.

```
> db.zips.aggregate([
    {
        $sort: {
            pop: -1
        }
    }
])
< {
    _id: ObjectId("5c8eccc1caa187d17ca7044d"),
    city: 'CHICAGO',
    zip: '60623',
    loc: {
      y: 41.849015,
      x: 87.7157
    },
    pop: 112047,
    state: 'IL'
  }
  {
    _id: ObjectId("5c8eccc1caa187d17ca7307f"),
    city: 'BROOKLYN',
    zip: '11226',
    loc: {
      y: 40.646694,
      x: 73.956985
    },
    pop: 111396,
    state: 'NY'
  }
  {
    _id: ObjectId("5c8eccc1caa187d17ca72fa0"),
    city: 'NEW YORK',
    zip: '10021',
    loc: {
      y: 40.768476,
      x: 73.958805
    },
    pop: 106564,
    state: 'NY'
  }
  {
```

۱-۷-۴. 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.



۱-۷-۵. 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.

```
> db.zips.aggregate([
    {
        $project: {
            state: 1,
            zip: 1,
            population: "$pop",
            _id: 0
        }
    }
])
< {
    zip: '35020',
    state: 'AL',
    population: 40549
  }
  {
    zip: '35019',
    state: 'AL',
    population: 1781
  }
  {
    zip: '35023',
    state: 'AL',
    population: 39677
  }
  {
    zip: '35031',
    state: 'AL',
    population: 9058
  }
  {
    zip: '35035',
    state: 'AL',
    population: 1282
  }
  {
    zip: '35033',
    state: 'AL',
    population: 3448
  }
```

۱-۷-۶. 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.

```
> db.zips.aggregate([
    {
        $set: {
            pop_2022: {
                $round: { $multiply: [1.0031, "$pop"] }
            }
        }
    }
])
< {
    _id: ObjectId("5c8eccc1caa187d17ca6ed17"),
    city: 'BESSEMER',
    zip: '35020',
    loc: {
      y: 33.409002,
      x: 86.947547
    },
    pop: 40549,
    state: 'AL',
    pop_2022: 40675
  }
  {
    _id: ObjectId("5c8eccc1caa187d17ca6ed19"),
    city: 'BAILEYTON',
    zip: '35019',
    loc: {
      y: 34.268298,
      x: 86.621299
    },
    pop: 1781,
    state: 'AL',
    pop_2022: 1787
  }
  {
    _id: ObjectId("5c8eccc1caa187d17ca6ed1a"),
    city: 'HUEYTOWN',
    zip: '35023',
    loc: {
      y: 33.414625,
      x: 86.999607
    },
    pop: 39677,
    state: 'AL',
    pop_2022: 39800
  }
```

۱-۷-۷. استیج $count

```
db.zips.aggregate([
    {
        $count: 'total_zips'
    }
])
```

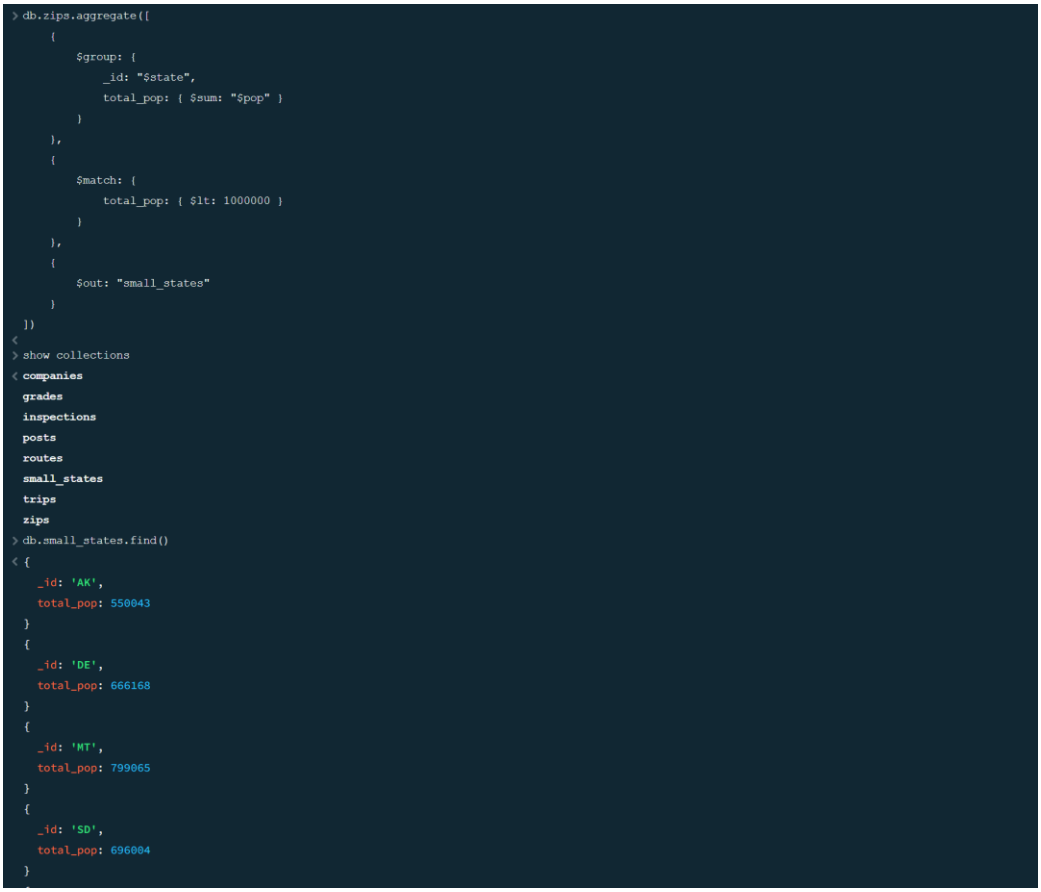
این کوئری نیز تعداد کل zip-ها را برمی‌گرداند.

```
> db.zips.aggregate([
    {
        $count: 'total_zips'
    }
])
< {
    total_zips: 29466
  }
Atlas atlas-bev207-shard-0 [primary] sample_training>
```

۱-۷-۸. 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.

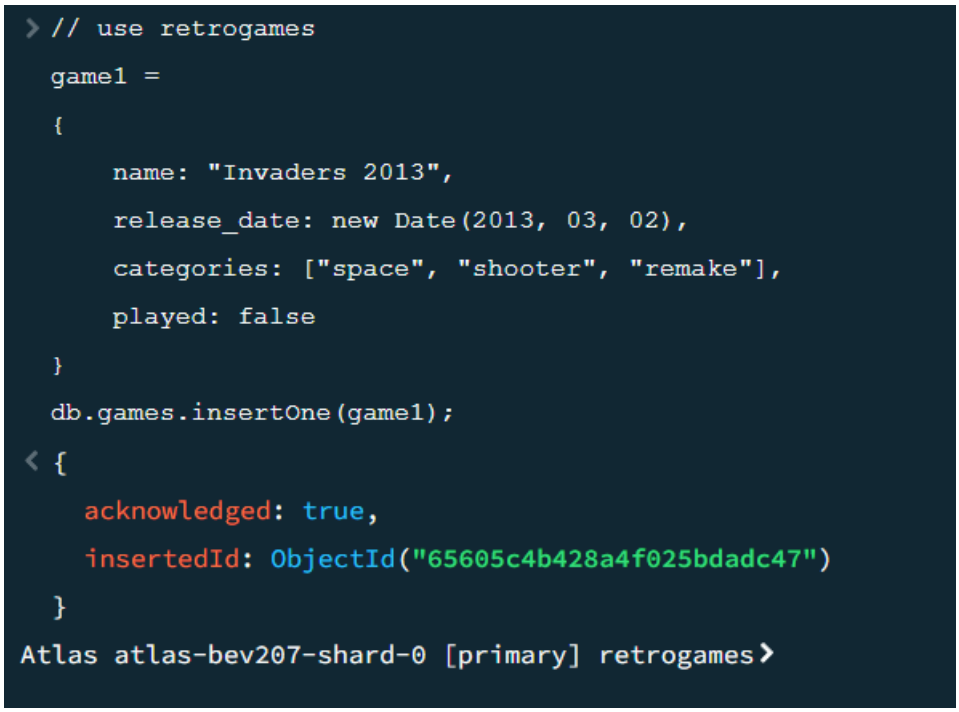


پاسخ ۲. مقاله سایت مهندسی داده

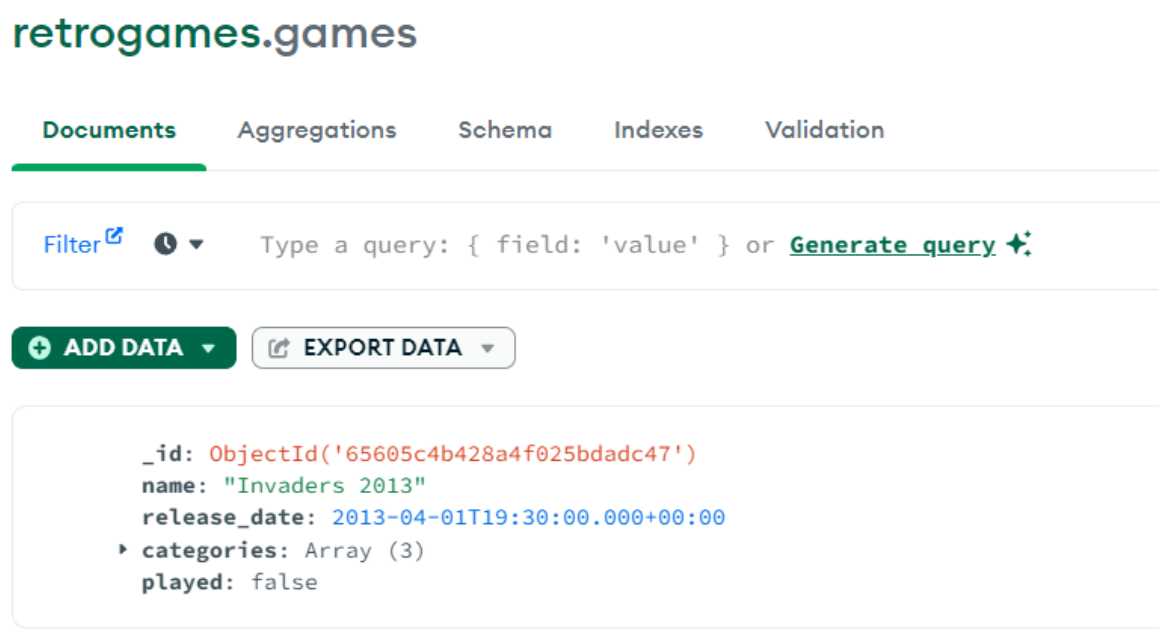
ابتدا پایگاه داده را ساخته و یک game به آن اضافه می‌کنیم:

```
// 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.find().limit(100)
< {
    _id: ObjectId("65605c4b428a4f025bdadc47"),
    name: 'Invaders 2013',
    release_date: 2013-04-01T19:30:00.000Z,
    categories: [
      'space',
      'shooter',
      'remake'
    ],
    played: false
  }
Atlas atlas-bev207-shard-0 [primary] retrogames>
```

```
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.

```
> db.games.findOne({ name: "Invaders 2013"})
< {
    _id: ObjectId("65605c4b428a4f025bdadc47"),
    name: 'Invaders 2013',
    release_date: 2013-04-01T19:30:00.000Z,
    categories: [
      'space',
      'shooter',
      'remake'
    ],
    played: false
  }
Atlas atlas-bev207-shard-0 [primary] retrogames>
```

```
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.

```
> 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)
< {
    acknowledged: true,
    insertedId: ObjectId("65605fe8428a4f025bdadc49")
  }
Atlas atlas-bev207-shard-0 [primary] retrogames>
```

```
db.games.updateOne(
    { _id: ObjectId("65605c4b428a4f025bdadc47") },
    { $set: { played: true } }
)
```

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

```
> db.games.updateOne(
      { _id: ObjectId("65605c4b428a4f025bdadc47") },
      { $set: { played: true } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
Atlas atlas-bev207-shard-0 [primary] retrogames>
```

```
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.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)
              }
          }
      }
  )
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
Atlas atlas-bev207-shard-0 [primary] retrogames>
```

```
db.players.findOne()
```

نتیجه:

```
> db.players.findOne()
< {
    _id: ObjectId("65605fe8428a4f025bdadc49"),
    name: 'PUZZLEGAMESMASTER',
    gender: 'male',
    scores: [
      {
        game_id: ObjectId("65605c4b428a4f025bdadc47"),
        game_name: 'Invaders 2013',
        score: 10500,
        score_date: 2013-04-01T19:30:00.000Z
      },
      {
        game_id: ObjectId("51e10c50085977bc3cd92a65"),
        game_name: 'Invaders 2013',
        score: 30250,
        score_date: 2013-04-02T19:30:00.000Z
      }
    ]
  }
Atlas atlas-bev207-shard-0 [primary] retrogames>
```

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