

```

For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
---
The server generated these startup warnings when booting:
  2022-11-06T00:20:07.057+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> use TYITDB
switched to db TYITDB
> dn.dropDatabase()
uncaught exception: ReferenceError: dn is not defined :
@(shell):1:1
> db.dropDatabase()
{ "ok" : 1 }
>

```

```

> use TYITDB
switched to db TYITDB
> db.createCollection("Pradeepkumar")
{ "ok" : 1 }
> db.Pradeepkumar.drop()
true
>

```

```

> db.Pradeepkumar.insert({course:"TYBSCIT",College:"MCC",rollno:229702})
WriteResult({ "nInserted" : 1 })
> db.Pradeepkumar.find().pretty()
{
  "_id" : ObjectId("6367d176328c5b23b41a5b92"),
  "course" : "TYBSCIT",
  "College" : "MCC",
  "rollno" : 229702
}
> db.Pradeepkumar.update({},{$set:{College:"Mulund College of Commerce"}},{})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Pradeepkumar.find().pretty()
{
  "_id" : ObjectId("6367d176328c5b23b41a5b92"),
  "course" : "TYBSCIT",
  "College" : "Mulund College of Commerce",
  "rollno" : 229702
}
> db.Pradeepkumar.remove({})
WriteResult({ "nRemoved" : 1 })
> db.Pradeepkumar.find().pretty()
>

```

```
> db.employee.find().pretty()
{
  "_id" : ObjectId("6367d370328c5b23b41a5b93"),
  "EMPID" : 1,
  "ENAME" : "Pradeepkumar Pal",
  "SAL" : 100000,
  "CITY" : "Mulund",
  "DNAME" : "Testing"
}
{
  "_id" : ObjectId("6367d3ba328c5b23b41a5b94"),
  "EMPID" : 1,
  "ENAME" : "Virat Kohli",
  "SAL" : 10000000,
  "CITY" : "Delhi",
  "DNAME" : "Software Developer"
}
{
  "_id" : ObjectId("6367d42c328c5b23b41a5b95"),
  "EMPID" : 3,
  "ENAME" : "AB de Vellier's",
  "SAL" : 1000000,
  "CITY" : "cape town",
  "DNAME" : "solution analyst"
}
{
  "_id" : ObjectId("6367d471328c5b23b41a5b96"),
  "EMPID" : 4,
  "ENAME" : "Steve smith",
  "SAL" : 250000,
  "CITY" : "australia",
  "DNAME" : "Backend developer"
}
{
  "_id" : ObjectId("6367d4d4328c5b23b41a5b97"),
  "EMPID" : 5,
  "ENAME" : "Chris gayle",
  "SAL" : 350000,
  "CITY" : "west indies",
  "DNAME" : "Frontend developer"
}
>
```

```
> db.employee.find({CITY:"australia",SAL:{$gt:50000}}).pretty()
{
  "_id" : ObjectId("6367d471328c5b23b41a5b96"),
  "EMPID" : 4,
  "ENAME" : "Steve smith",
  "SAL" : 250000,
  "CITY" : "australia",
  "DNAME" : "Backend developer"
}
>
```

```
> db.employee.find({SAL:{$gte:20000},SAL:{$lt:600000}}).pretty()
{
  "_id" : ObjectId("6367d370328c5b23b41a5b93"),
  "EMPID" : 1,
  "ENAME" : "Pradeepkumar Pal",
  "SAL" : 100000,
  "CITY" : "Mulund",
  "DNAME" : "Testing"
}
{
  "_id" : ObjectId("6367d471328c5b23b41a5b96"),
  "EMPID" : 4,
  "ENAME" : "Steve smith",
  "SAL" : 250000,
  "CITY" : "australia",
  "DNAME" : "Backend developer"
}
{
  "_id" : ObjectId("6367d4d4328c5b23b41a5b97"),
  "EMPID" : 5,
  "ENAME" : "Chris gayle",
  "SAL" : 350000,
  "CITY" : "west indies",
  "DNAME" : "Frontend developer"
}
>
```

```
> db.employee.find({CITY:"Mulund"}).pretty()
{
  "_id" : ObjectId("6367d370328c5b23b41a5b93"),
  "EMPID" : 1,
  "ENAME" : "Pradeepkumar Pal",
  "SAL" : 100000,
  "CITY" : "Mulund",
  "DNAME" : "Testing"
}
```

```
> db.employee.find({SAL:{$ne:50000},$or:[{DNAME:"Testing"},{DNAME:"Software Developer"}]}).pretty()
{
  "_id" : ObjectId("6367d370328c5b23b41a5b93"),
  "EMPID" : 1,
  "ENAME" : "Pradeepkumar Pal",
  "SAL" : 100000,
  "CITY" : "Mulund",
  "DNAME" : "Testing"
},
{
  "_id" : ObjectId("6367d3ba328c5b23b41a5b94"),
  "EMPID" : 1,
  "ENAME" : "Virat Kohli",
  "SAL" : 10000000,
  "CITY" : "Delhi",
  "DNAME" : "Software Developer"
}
```

```
> db.rest.find().pretty()
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    },
    {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    }
  ]
}
```

```

> db.rest.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1,"_id":0}).pretty()
{
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "borough" : "Queens",
  "cuisine" : "Jewish/Kosher",
  "name" : "Tov Kosher Kitchen",
  "restaurant_id" : "40356068"
}
{
  "borough" : "Queens",
  "cuisine" : "American ",
  "name" : "Brunos On The Boulevard",
  "restaurant_id" : "40356151"
}

```

```

> db.rest.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1,"_id":0}).pretty()
{
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "borough" : "Queens",
  "cuisine" : "Jewish/Kosher",
  "name" : "Tov Kosher Kitchen",
  "restaurant_id" : "40356068"
}
{
  "borough" : "Queens",
  "cuisine" : "American ",
  "name" : "Brunos On The Boulevard",
  "restaurant_id" : "40356151"
}

```

Command Prompt - mongo

```
:1:31
> db.rest.find({},{"restaurant_id":1,"name":1,"borough":1,"address.zipcode":1}).pretty()
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "address" : {
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e2"),
  "address" : {
    "zipcode" : "11225"
  },
  "borough" : "Brooklyn",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e3"),
  "address" : {
    "zipcode" : "10019"
  },
  "borough" : "Manhattan",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
```

Command Prompt - mongo

```
Type "it" for more
> db.rest.find({"borough":"Bronx"}).pretty()
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    }
  ]
}
```

```
Command Prompt - mongo
Type "it" for more
> db.rest.find({"borough":"Bronx"}).limit(5).pretty()
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    },
    {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    }
  ]
}
```

```
Command Prompt - mongo
Type "it" for more
> db.rest.find({"borough":"Bronx"}).limit(5).pretty()
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    },
    {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    }
  ]
}
```

Command Prompt - mongo

```
> db.rest.find( { grades : { $elemMatch:{ "score":{ $gt : 90} } } } ).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f73f"),
  "address" : {
    "building" : "65",
    "coord" : [
      -73.9782725,
      40.7624022
    ],
    "street" : "West 54 Street",
    "zipcode" : "10019"
  },
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "grades" : [
    {
      "date" : ISODate("2014-08-22T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    },
    {
      "date" : ISODate("2014-03-28T00:00:00Z"),
      "grade" : "C",
      "score" : 131
    },
    {
      "date" : ISODate("2013-09-25T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    }
  ]
}
```

Command Prompt - mongo

```
> db.rest.find( { grades : { $elemMatch:{ "score":{ $gt : 80 , $lt :100} } } } ).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f7df"),
  "address" : {
    "building" : "345",
    "coord" : [
      -73.9864626,
      40.7266739
    ],
    "street" : "East 6 Street",
    "zipcode" : "10003"
  },
  "borough" : "Manhattan",
  "cuisine" : "Indian",
  "grades" : [
    {
      "date" : ISODate("2014-09-15T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    },
    {
      "date" : ISODate("2014-01-14T00:00:00Z"),
      "grade" : "A",
      "score" : 8
    },
    {
      "date" : ISODate("2013-05-30T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    }
  ]
}
```



```
Command Prompt - mongo
> db.rest.find( {"address.coord" : { $lt : -95.754168}} ).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17fc27"),
  "address" : {
    "building" : "3707",
    "coord" : [
      -101.8945214,
      33.5197474
    ],
    "street" : "82 Street",
    "zipcode" : "11372"
  },
  "borough" : "Queens",
  "cuisine" : "American ",
  "grades" : [
    {
      "date" : ISODate("2014-06-04T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    },
    {
      "date" : ISODate("2013-11-07T00:00:00Z"),
      "grade" : "B",
      "score" : 19
    },
    {
      "date" : ISODate("2013-05-17T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    }
  ]
}
```

```
> db.rest.find({ $and: [{"cuisine" : { $ne : "American " }}, {"grades.score" : { $gt : 70 }}, {"address.coord" : { $lt : -65.754168}}]}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f7df"),
  "address" : {
    "building" : "345",
    "coord" : [
      -73.9864626,
      40.7266739
    ],
    "street" : "East 6 Street",
    "zipcode" : "10003"
  },
  "borough" : "Manhattan",
  "cuisine" : "Indian",
  "grades" : [
    {
      "date" : ISODate("2014-09-15T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    },
    {
      "date" : ISODate("2014-01-14T00:00:00Z"),
      "grade" : "A",
      "score" : 8
    },
    {
      "date" : ISODate("2013-05-30T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    }
  ]
}
```

```

> db.rest.find( {"cuisine" : {$ne : "American "}, "grades.grade" : "A", "borough": "Brooklyn" }).sort({"cuisine":-1}).pretty();
{
  "_id" : ObjectId("60f9237591f2ead52b17fde9"),
  "address" : {
    "building" : "2268",
    "coord" : [
      -73.9564939,
      40.650368
    ],
    "street" : "Church Avenue",
    "zipcode" : "11226"
  },
  "borough" : "Brooklyn",
  "cuisine" : "Vegetarian",
  "grades" : [
    {
      "date" : ISODate("2014-07-28T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2014-02-25T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    }
  ],
  "restaurant_id" : "40356483"
}

```

```

> db.rest.find({name: /^Wil/},{ "restaurant_id" : 1, "name":1,"borough":1, "cuisine" :1 }).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e8"),
  "borough" : "Brooklyn",
  "cuisine" : "Delicatessen",
  "name" : "Wilken'S Fine Food",
  "restaurant_id" : "40356483"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5eb"),
  "borough" : "Bronx",
  "cuisine" : "American ",
  "name" : "Wild Asia",
  "restaurant_id" : "40357217"
}
{
  "_id" : ObjectId("60f9237591f2ead52b1803f0"),
  "borough" : "Bronx",
  "cuisine" : "Pizza",
  "name" : "Wilbel Pizza",
  "restaurant_id" : "40871979"
}
>

```

```
> db.rest.find({name: /ces$/},{ "restaurant_id" : 1, "name":1, "borough":1, "cuisine" :1 }).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17fa84"),
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Pieces",
  "restaurant_id" : "40399910"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17fb32"),
  "borough" : "Queens",
  "cuisine" : "American ",
  "name" : "S.M.R Restaurant Services",
  "restaurant_id" : "40403857"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17fb37"),
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Good Shepherd Services",
  "restaurant_id" : "40403989"
}
{
  "_id" : ObjectId("60f9237591f2ead52b17ffee"),
  "borough" : "Queens",
  "cuisine" : "Ice Cream, Gelato, Yogurt, Ices",
  "name" : "The Ice Box-Ralph'S Famous Italian Ices",
  "restaurant_id" : "40690899"
}
```

```
> db.rest.find({"borough":"Bronx",$or:[{"cuisine":"American"}, {"cuisine":"Chinese"}]}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f601"),
  "address" : {
    "building" : "1236",
    "coord" : [
      -73.8893654,
      40.81376179999999
    ],
    "street" : "238 Spofford Ave",
    "zipcode" : "10474"
  },
  "borough" : "Bronx",
  "cuisine" : "Chinese",
  "grades" : [
    {
      "date" : ISODate("2013-12-30T00:00:00Z"),
      "grade" : "A",
      "score" : 8
    },
    {
      "date" : ISODate("2013-01-08T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2012-06-12T00:00:00Z"),
      "grade" : "B",
      "score" : 15
    }
  ],
}
```

```

> db.rest.find({"borough":{$in:["StatenIsland","Queens","Bronx","Brooklyn"]}},{"restaurant_id":1
"name":1,"borough":1,"cuisine":1}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e2"),
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e4"),
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e5"),
  "borough" : "Queens",
  "cuisine" : "Jewish/Kosher",
  "name" : "Tov Kosher Kitchen",
  "restaurant_id" : "40356068"
}

```

```

> db.rest.find({"borough":{$nin:["StatenIsland","Queens","Bronx","Brooklyn"]}},{"restaurant_id":
1,"name":1,"borough":1,"cuisine":1}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e3"),
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e7"),
  "borough" : "Staten Island",
  "cuisine" : "Jewish/Kosher",
  "name" : "Kosher Island",
  "restaurant_id" : "40356442"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5ee"),
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "1 East 66Th Street Kitchen",
  "restaurant_id" : "40359480"
}

```

```

> db.rest.find({"grades.score":{"$not":{"$gt:10}}},"restaurant_id":1,"name":1,"borough":1,"cuisine":1}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f5ec"),
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "C & C Catering Service",
  "restaurant_id" : "40357437"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5ee"),
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "1 East 66Th Street Kitchen",
  "restaurant_id" : "40359480"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5f2"),
  "borough" : "Brooklyn",
  "cuisine" : "Delicatessen",
  "name" : "Nordic Delicacies",
  "restaurant_id" : "40361390"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f60b"),
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Sonny'S Heros",
  "restaurant_id" : "40363744"
}

```

```

> db.rest.find({$or:[{name:/^Wil/},{ "$and":[{"cuisine":{"$ne:"American"}},{ "cuisine":{"$ne:"Chinese"}]}]}},{ "restaurant_id":1,"name":1,"borough":1,"cuisine":1}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e2"),
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e3"),
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e4"),
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e5"),

```

```

> db.rest.find({"grades.1.date":ISODate("2014-0811T00:00:00Z"),"grades.1.grade":"A","grades.1.score":9},
{"restaurant_id":1,"name":1,"grades":1}).pretty()
{
  "_id" : ObjectId("60f9237491f2ead52b17fc0b"),
  "grades" : [
    {
      "date" : ISODate("2015-01-12T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2014-08-11T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    },
    {
      "date" : ISODate("2014-01-14T00:00:00Z"),
      "grade" : "A",
      "score" : 13
    },
    {
      "date" : ISODate("2013-02-07T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2012-04-30T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    }
  ],
  "name" : "Club Macanudo (Cigar Bar)",
  "restaurant_id" : "40526406"
}
>

```

```

> db.rest.find({"address.coord.1":{"$gt:42,$lte:52}},{"restaurant_id":1,"name":1,"address":1,"coord":1}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f881"),
  "address" : {
    "building" : "47",
    "coord" : [
      -78.877224,
      42.895461999999999
    ],
    "street" : "Broadway @ Trinity Pl",
    "zipcode" : "10006"
  },
  "name" : "T.G.I. Friday'S",
  "restaurant_id" : "40387990"

  "_id" : ObjectId("60f9237491f2ead52b17f8b4"),
  "address" : {
    "building" : "1",
    "coord" : [
      0.7119979,
      51.6514664
    ],
    "street" : "Pennplaza E, Penn Sta",
    "zipcode" : "10001"
  },
  "name" : "T.G.I. Fridays",
  "restaurant_id" : "40388936"

  "_id" : ObjectId("60f9237491f2ead52b17fb07"),

```



```

> db.rest.find().sort({"name":1}).pretty();
{
  "_id" : ObjectId("60f9237591f2ead52b180272"),
  "address" : {
    "building" : "129",
    "coord" : [
      -73.962943,
      40.685007
    ],
    "street" : "Gates Avenue",
    "zipcode" : "11238"
  },
  "borough" : "Brooklyn",
  "cuisine" : "Italian",
  "grades" : [
    {
      "date" : ISODate("2014-03-06T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    },
    {
      "date" : ISODate("2013-08-29T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-03-08T00:00:00Z"),
      "grade" : "A",
      "score" : 7
    },
    {
      "date" : ISODate("2012-06-27T00:00:00Z"),

```

```

> db.rest.find().sort({"name":-1}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f69d"),
  "address" : {
    "building" : "6946",
    "coord" : [
      -73.8811834,
      40.7017759
    ],
    "street" : "Myrtle Avenue",
    "zipcode" : "11385"
  },
  "borough" : "Queens",
  "cuisine" : "German",
  "grades" : [
    {
      "date" : ISODate("2014-09-24T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    },
    {
      "date" : ISODate("2014-04-17T00:00:00Z"),
      "grade" : "A",
      "score" : 7
    },
    {
      "date" : ISODate("2013-03-12T00:00:00Z"),
      "grade" : "A",
      "score" : 13
    },
    {
      "date" : ISODate("2012-10-02T00:00:00Z"),

```

```
> db.rest.find().sort({"cuisine":1,"borough":-1,}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17fccc"),
  "address" : {
    "building" : "1345",
    "coord" : [
      -73.959249,
      40.768076
    ],
    "street" : "2 Avenue",
    "zipcode" : "10021"
  },
  "borough" : "Manhattan",
  "cuisine" : "Afghan",
  "grades" : [
    {
      "date" : ISODate("2014-10-07T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    },
    {
      "date" : ISODate("2013-10-23T00:00:00Z"),
      "grade" : "A",
      "score" : 8
    },
    {
      "date" : ISODate("2012-10-26T00:00:00Z"),
      "grade" : "A",
      "score" : 13
    },
    {
```



```

> db.rest.find({"address.street":{"$exists:true}}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    },
    {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2011-11-23T00:00:00Z"),

```

```

> db.rest.find({"address.coord":{"$type:1}}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    },
    {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2011-11-23T00:00:00Z"),

```

```

> db.rest.find({"grades.score":{$mod:[7,0]}},{ "restaurant_id":1,"name":1,"grades":1}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    },
    {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2011-11-23T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    },
    {
      "date" : ISODate("2011-03-10T00:00:00Z"),
      "grade" : "B",
      "score" : 14
    }
  ],
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}

```

```

> db.rest.find({name:{$regex:/^Mad/i}}, {"name":1,"borough":1,"address.coord":1,"cuisine":1}).pretty();
{
  "_id" : ObjectId("60f9237491f2ead52b17fb1c"),
  "address" : {
    "coord" : [
      -73.9860597,
      40.7431194
    ]
  },
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Madison Square"
},
{
  "_id" : ObjectId("60f9237491f2ead52b17fbee"),
  "address" : {
    "coord" : [
      -73.98302199999999,
      40.742313
    ]
  },
  "borough" : "Manhattan",
  "cuisine" : "Indian",
  "name" : "Madras Mahal"
},
{
  "_id" : ObjectId("60f9237591f2ead52b17fea2"),
  "address" : {
    "coord" : [
      -74.000002,
      40.72735
    ]
  },
  "borough" : "Manhattan",
  "cuisine" : "Indian",
  "name" : "Madras Mahal"
}
}

```

```

> use TYIT
switched to db TYIT
> db.users.insert({"FName":"Aryamaan","LName":"Phadnis","Age":19,"Gender":"M","Country":"India");
... }
2021-07-23T10:29:35.340+0530 E QUERY [thread1] SyntaxError: missing } after property list @(shell):1:9
> db.users.insert({"FName":"Aryamaan","LName":"Phadnis","Age":19,"Gender":"M","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"John","LName":"Gary","Age":41,"Gender":"M","Country":"USA"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Emily","LName":"Blunt","Age":38,"Gender":"F","Country":"UK"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Goldy","LName":"Jaiswal","Age":38,"Gender":"F","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Ritik","LName":"Jaiswal","Age":38,"Gender":"M","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Ronald","LName":"Donald","Age":40,"Gender":"M","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Donald","LName":"Trump","Age":52,"Gender":"M","Country":"USA"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Rakesh","LName":"Gill","Age":35,"Gender":"M","Country":"UK"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Lilly","LName":"Singh","Age":30,"Gender":"F","Country":"USA"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Sweety","LName":"Singh","Age":27,"Gender":"F","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Omkar","LName":"Parkar","Age":25,"Gender":"M","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Swaroop","LName":"Dhamankar","Age":29,"Gender":"M","Country":"USA"});
WriteResult({ "nInserted" : 1 })

```

```

> db.users.update({'Gender':'F'},{$set:{"Country":"UK"}},{multi:true})
WriteResult({ "nMatched" : 4, "nUpserted" : 0, "nModified" : 3 })
>

```

```

> db.your_collection.update({},{$set: {"Company": 1}},false,true)
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.users.update({},{$set: {"Company": 1}},false,true)
WriteResult({ "nMatched" : 4, "nUpserted" : 0, "nModified" : 4 })
>

```

```

>
> db.users.remove({'Gender':'M'})
WriteResult({ "nRemoved" : 8 })
>
>

```

```

> db.users.count({"gender":"F", $or:[{"Country":"India"}, {"Country":"USA"}]})
0
>
>
>
>

```

```
> db.users.find({"Gender":"F"},{"FName":1,"Age":1,"_id":0})
{ "FName" : "Emily", "Age" : 38 }
{ "FName" : "Goldy", "Age" : 38 }
{ "FName" : "Lilly", "Age" : 30 }
{ "FName" : "Sweety", "Age" : 27 }
```

```
> db.mycol.find().pretty()
{
  "_id" : ObjectId("60ee64f48d30381dbaf4ff7e"),
  "title" : "MongoDB Overview",
  "description" : "MongoDB is no sql database",
  "by_user" : "tutorials point",
  "url" : "http://www.tutorialspoint.com",
  "tags" : [
    "mongodb",
    "database",
    "NoSQL"
  ],
  "likes" : 100
}
{
  "_id" : ObjectId("60ee651c8d30381dbaf4ff7f"),
  "title" : "NoSQL Overview",
  "description" : "No sql database is very fast",
  "by_user" : "tutorials point",
  "url" : "http://www.tutorialspoint.com",
  "tags" : [
    "mongodb",
    "database",
    "NoSQL"
  ],
  "likes" : 10
}
{
  "_id" : ObjectId("60ee65768d30381dbaf4ff80"),
  "title" : "Neo4j Overview",
```

```
> db.mycol.aggregate([{$group: {_id: "$by_user", num_tutorial: {$sum: 1}}}])
{ "_id" : "Neo4j", "num_tutorial" : 1 }
{ "_id" : "tutorials point", "num_tutorial" : 2 }
```

```
> db.mycol.aggregate([{$group: {_id: "$by_user", num_tutorial: {$sum: "$likes"}}}])
{ "_id" : "Neo4j", "num_tutorial" : 750 }
{ "_id" : "tutorials point", "num_tutorial" : 110 }
```

```
> db.mycol.aggregate([{$group:{_id:"$by_user",num_tutorial:{$avg:"$likes"}}}])
{ "_id" : "Neo4j", "num_tutorial" : 750 }
{ "_id" : "tutorials point", "num_tutorial" : 55 }
>
```

```
> db.mycol.aggregate([{$group:{_id:"$by_user",num_tutorial:{$min:"$likes"}}}])
{ "_id" : "Neo4j", "num_tutorial" : 750 }
{ "_id" : "tutorials point", "num_tutorial" : 10 }
>
```

```
> db.mycol.aggregate([{$group:{_id:"$by_user",num_tutorial:{$max:"$likes"}}}])
{ "_id" : "Neo4j", "num_tutorial" : 750 }
{ "_id" : "tutorials point", "num_tutorial" : 100 }
>
```

```
> db.mycol.aggregate([{$group:{_id:"$by_user",url:{$push:"$url"}}}])
{ "_id" : "Neo4j", "url" : [ "http://www.neo4j.com" ] }
{ "_id" : "tutorials point", "url" : [ "http://www.tutorialspoint.com", "http://www.tutorialspoint.com" ] }
>
```

```
> db.mycol.aggregate([{$group:{_id:"$by_user",url:{$addToSet:"$url"}}}])
{ "_id" : "Neo4j", "url" : [ "http://www.neo4j.com" ] }
{ "_id" : "tutorials point", "url" : [ "http://www.tutorialspoint.com" ] }
>
```

```
> db.mycol.aggregate([{$group:{_id:"$by_user",first_url:{$first:"$url"}}}])
{ "_id" : "Neo4j", "first_url" : "http://www.neo4j.com" }
{ "_id" : "tutorials point", "first_url" : "http://www.tutorialspoint.com" }
>
```

```
> db.mycol.aggregate([{$group:{_id:"$by_user",last_url:{$last:"$url"}}}])
{ "_id" : "Neo4j", "last_url" : "http://www.neo4j.com" }
{ "_id" : "tutorials point", "last_url" : "http://www.tutorialspoint.com" }
>
■
```



```
Microsoft Windows [Version 10.0.19042.1110]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>start mongod --replSet ty219713 --logpath \data\rs1\2.log -dbpath \data\rs2 --port 27018 --smallfiles --oplogSize 64

C:\Users\Admin>
```

```
Microsoft Windows [Version 10.0.19042.1110]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>start mongod --replSet ty219713 --logpath \data\rs1\2.log -dbpath \data\rs2 --port 27018 --smallfiles --oplogSize 64

C:\Users\Admin>
```

```
C:\Users\Admin>start mongod --replSet ty219713 --logpath \data\rs1\3.log --dbpath \data\rs3 --port 27019 --smallfiles --oplogSize 64

C:\Users\Admin>
```

```
C:\Program Files\MongoDB\Server\3.6\bin>mongod.exe
```

```
Command Prompt - mongo --port 27017
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>mongo --port 27017
MongoDB shell version v3.6.0
connecting to: mongodb://127.0.0.1:27017/
MongoDB server version: 3.6.0
Server has startup warnings:
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten]
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten] **      Read and write access to data and configuration is unrestricted.
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten]
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten] **      Remote systems will be unable to connect to this server.
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten] **      Start the server with --bind_ip <address> to specify which IP
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten] **      addresses it should serve responses from, or with --bind_ip_all to
2021-08-12T14:20:13.550+0530 I CONTROL [initandlisten] **      bind to all interfaces. If this behavior is desired, start the
2021-08-12T14:20:13.551+0530 I CONTROL [initandlisten] **      server with --bind_ip 127.0.0.1 to disable this warning.
2021-08-12T14:20:13.551+0530 I CONTROL [initandlisten]
2021-08-12T14:20:13.551+0530 I CONTROL [initandlisten]
2021-08-12T14:20:13.551+0530 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2021-08-12T14:20:13.551+0530 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2021-08-12T14:20:13.551+0530 I CONTROL [initandlisten]
```

Command Prompt - mongo --port 27018

(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>mongo --port 27018

MongoDB shell version v3.6.0

connecting to: mongodb://127.0.0.1:27018/

MongoDB server version: 3.6.0

Server has startup warnings:

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten]

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten]

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.

2021-08-12T14:21:01.747+0530 I CONTROL [initandlisten]

2021-08-12T14:21:01.748+0530 I CONTROL [initandlisten]

2021-08-12T14:21:01.748+0530 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.

2021-08-12T14:21:01.748+0530 I CONTROL [initandlisten] See <http://dochub.mongodb.org/core/wt-windows-system-file-cache>

2021-08-12T14:21:01.748+0530 I CONTROL [initandlisten]

(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>mongo --port 27019

MongoDB shell version v3.6.0

connecting to: mongodb://127.0.0.1:27019/

MongoDB server version: 3.6.0

Server has startup warnings:

2021-08-12T14:21:27.893+0530 I CONTROL [initandlisten]

2021-08-12T14:21:27.893+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.

2021-08-12T14:21:27.893+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten]

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten]

2021-08-12T14:21:27.894+0530 I CONTROL [initandlisten]

2021-08-12T14:21:27.895+0530 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.

2021-08-12T14:21:27.895+0530 I CONTROL [initandlisten] See <http://dochub.mongodb.org/core/wt-windows-system-file-cache>

2021-08-12T14:21:27.895+0530 I CONTROL [initandlisten]

— □ ×

Command Prompt - mongo --port 27018

Command Prompt - mongo --port 27019

[illegible]

Command Prompt - mongo --port 27017

```
ty219713:PRIMARY> db.student.insert({"name":"Aryamaan","class":"TYIT"})
WriteResult({ "nInserted" : 1 })
ty219713:PRIMARY>
```

Command Prompt - mongo --port 27018

```
ty219713:SECONDARY> db.student.find().pretty()
{
  "_id" : ObjectId("6115551cd42059e4ea47adba"),
  "name" : "Aryamaan",
  "class" : "TYIT"
}
ty219713:SECONDARY>
```

Command Prompt - mongo --port 27019

```
ty219713:SECONDARY> db.student.find().pretty()
{
  "_id" : ObjectId("6115551cd42059e4ea47adba"),
  "name" : "Aryamaan",
  "class" : "TYIT"
}
ty219713:SECONDARY>
```

```
D:\NGT_Practical\javaMOD>java insert
Sep 15, 2022 9:56:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Connected to the database successfully
Collection sampleCollection selected successfully
Sep 15, 2022 9:56:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: No server chosen by PrimaryServerSelector from cluster description ClusterDescription{type=UNKNOWN, connectionMode=SINGLE, all=[ServerDescription{address=localhost:27017, type=UNKNOWN, state=CONNECTING}]]. Waiting for 30000 ms before timing out
Sep 15, 2022 9:56:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:204}] to localhost:27017
Sep 15, 2022 9:56:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[5, 0, 9]}, minWireVersion=8, maxWireVersion=13, electionId=null, maxDocumentSize=16777216, roundTripTimeNanos=1579800}
Sep 15, 2022 9:56:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:205}] to localhost:27017
Document inserted successfully
```

```

> db.TYITCOL.find().pretty()
{
  "_id" : ObjectId("632352289e51d473d0f430c3"),
  "id" : 229726,
  "name" : "Kartik Sharma",
  "age" : 19,
  "class" : "TYBScIT",
  "college" : "Mulund cillege of Commerce"
}
{
  "_id" : ObjectId("6325793418280e41a0b1331d"),
  "id" : 229727,
  "name" : "Tom",
  "age" : 34,
  "class" : "TYBScCS",
  "college" : "Mulund cillege of Commerce"
}
{
  "_id" : ObjectId("6325799518280e0560b5868f"),
  "id" : 229727,
  "name" : "Jerry",
  "age" : 23,
  "class" : "TYBSC",
  "college" : "Mulund cillege of Commerce"
}

```

```

D:\NGT_Practical\javaMOD>javac update.java
Note: update.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\NGT_Practical\javaMOD>java update
Sep 17, 2022 1:19:50 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Connected to the database successfully
Sep 17, 2022 1:19:50 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: No server chosen by PrimaryServerSelector from cluster description ClusterDescription{type=UNKNOWN, connectionMode=SINGLE, all=[ServerDescription{address=localhost:27017, type=UNKNOWN, state=CONNECTING}]}: Waiting for 30000 ms before timing out
Sep 17, 2022 1:19:50 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:25}] to localhost:27017
Sep 17, 2022 1:19:50 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[5, 0, 9]}, minWireVersion=0, maxWireVersion=13, electionId=null, maxDocumentSize=16777216, roundTripTimeNanos=1508700}
Sep 17, 2022 1:19:50 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:26}] to localhost:27017
Sep 17, 2022 1:19:50 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Closed connection [connectionId{localValue:2, serverValue:26}] to localhost:27017 because the pool has been closed

> db.TYITCOL.find().pretty()
{
  "_id" : ObjectId("632352289e51d473d0f430c3"),
  "id" : 229726,
  "name" : "Shivay",
  "age" : 19,
  "class" : "TYBScIT",
  "college" : "Mulund cillege of Commerce"
}
{
  "_id" : ObjectId("6325793418280e41a0b1331d"),
  "id" : 229727,
  "name" : "Tom",
  "age" : 34,
  "class" : "TYBScIT",
  "college" : "Mulund cillege of Commerce"
}
{
  "_id" : ObjectId("6325799518280e0560b5868f"),
  "id" : 229727,
  "name" : "Jerry",
  "age" : 23,
  "class" : "TYBSC",

```

```

D:\NGT_Practical\javaMOD>javac delete.java

D:\NGT_Practical\javaMOD>java delete
Sep 28, 2022 8:58:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Connected to the database successfully
Collection sampleCollection selected successfully
Sep 28, 2022 8:58:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: No server chosen by PrimaryServerSelector from cluster description ClusterDescription{type=UNKNOWN, connectionMode=SINGLE, all=[ServerDescription{address=localhost:27017, type=UNKNOWN, state=CONNECTING}]}]. Waiting for 30000 ms before timing out
Sep 28, 2022 8:58:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:1}] to localhost:27017
Sep 28, 2022 8:58:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=SINGLE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[5, 0, 9]}, minWireVersion=0, maxWireVersion=13, electionId=null, maxDocumentSize=16777216, roundTripTimeNanos=1060800}
Sep 28, 2022 8:58:49 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:2}] to localhost:27017
Document deleted successfully

D:\NGT_Practical\javaMOD>
> db.TYITCOL.find().pretty()
{
  "_id" : ObjectId("632352289e51d473d0f430c3"),
  "id" : 229726,
  "name" : "Shivay",
  "age" : 19,
  "class" : "TYBScIT",
  "college" : "Mulund cillege of Commerce"
}

```

```

D:\NGT_Practical\javaMOD>javac retrieve.java

D:\NGT_Practical\javaMOD>java retrieve
Sep 17, 2022 1:29:21 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Connected to the database successfully
Sep 17, 2022 1:29:21 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: No server chosen by ReadPreferenceServerSelector{readPreference=primary} from cluster description ClusterDescription{type=UNKNOWN, connectionMode=SINGLE, all=[ServerDescription{address=localhost:27017, type=UNKNOWN, state=CONNECTING}]}]. Waiting for 30000 ms before timing out
Sep 17, 2022 1:29:21 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:31}] to localhost:27017
Sep 17, 2022 1:29:21 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=SINGLE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[5, 0, 9]}, minWireVersion=0, maxWireVersion=13, electionId=null, maxDocumentSize=16777216, roundTripTimeNanos=3480701}
Sep 17, 2022 1:29:21 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:32}] to localhost:27017
Document({_id=632352289e51d473d0f430c3, id=229726, name=Shivay, age=19, class=TYBScIT, college=Mulund cillege of Commerce})

```

```

MongoDB Enterprise > db.PRADEEPCOL.find().pretty()
{
  "_id" : ObjectId("634d43c7d7271a8813000029"),
  "name" : "Shivay",
  "age" : 20,
  "dept" : "TYIT",
  "pin" : 229602
}

```

```

MongoDB Enterprise > db.PRADEEPCOL.find().pretty()
{
  "_id" : ObjectId("634d43c7d7271a8813000029"),
  "name" : "Shivay",
  "age" : 20,
  "dept" : "TYIT",
  "pin" : 229602
}
{
  "_id" : ObjectId("634d43c7d7271a8813000030"),
  "name" : "Pradeep",
  "age" : 20,
  "dept" : "TYBSCIT",
  "pin" : 229802
}

```

```

MongoDB Enterprise > db.PRADEEPCOL.find().pretty()
{
  "_id" : ObjectId("634d43c7d7271a8813000029"),
  "name" : "Shivay",
  "age" : 20,
  "dept" : "TYIT",
  "pin" : 229602
}

```

```

> db.TYITCOL.find().pretty()
{
  "_id" : ObjectId("63208a5ba295d4c2d37e5626"),
  "name" : "Kartik",
  "age" : "20"
}
{
  "_id" : ObjectId("63208a66860793d45955d36f"),
  "name" : "Tom",
  "age" : "21"
}
{
  "_id" : ObjectId("63208a70694b9f16718b5323"),
  "name" : "Jerry",
  "age" : "22"
}

```

```

> db.TYITCOL.find().pretty()
{
  "_id" : ObjectId("63208a5ba295d4c2d37e5626"),
  "name" : "Kartik",
  "age" : 45
}
{ "_id" : ObjectId("63208a66860793d45955d36f"), "name" : "Tom", "age" : 45 }
{
  "_id" : ObjectId("63208a70694b9f16718b5323"),
  "name" : "Jerry",
  "age" : "22"
}

```


Well Come to beautiful world

Click Me

Hello!

Student

Well Come

```
> db.TYITCOL.find().pretty()  
{  
  "_id" : ObjectId("63208a5ba295d4c2d37e5626"),  
  "name" : "Kartik",  
  "age" : 45  
}
```

← → ↻ ⓘ File | D:/NGT_Practical/229726_event.html

hover

Well Come to Mulund College of Commerce

TYIT Student

off

Hello All

on

Hello world Text Marked!Text Marked!Text Marked!

trigger

← → ↻ ⓘ File | D:/NGT_Practical/229726_hide_show.html

Mulund college of Commerce

Hide

Show

← → ↻ ⓘ File | D:/NGT_Practical/229726_HS_toggle.html

Well Come to beautiful world

Toggle Between Hide And Show

Well Come to beautiful world

Fade Out Fade In

Well Come to beautiful world

Fade Toggle

Well Come to beautiful world

Slide Up Slide Down

Well Come to beautiful world

Slide Toggle



Animation



Well Come to beautiful world

Click

This is a simple Paragraph.

Click

← → ↻ ⓘ File | D:/NGT_Practical/229726_callback.html

Click

This page says
The paragraph is now hide

Get All Paragraph's text

This is a paragraph.

This is an another paragraph.

Set all paragraph text

This is a test paragraph.

This is an another test paragraph.

This page says
This is a paragraph.This is an another paragraph.This is a test paragraph.This is an another test paragraph.

OK

Get paragraph HTML content

The quick **brown for** jumps over the lazy dog.

Write Message

This page says
The quick brown for jumps over the lazy dog.

OK

```
{"Bname":"Pradeep","publiser":"Rolls-RoyceMotor Cars","author":"Motor Cars","price":6.22cr"}
{"Bname":"Pradeep","author":"Motor Cars","price":6.22cr"}
{"Bname":"Pradeep","author":"Motor Cars","price":6.22cr"}
```

```
Pradeepkumar
name:Pradeepkumar
sem:5
type:Programming
name:Karthik
type:programming|
```

```
Bname : Pradeepkumar
publiser :Rolls-royce motor cars|
```

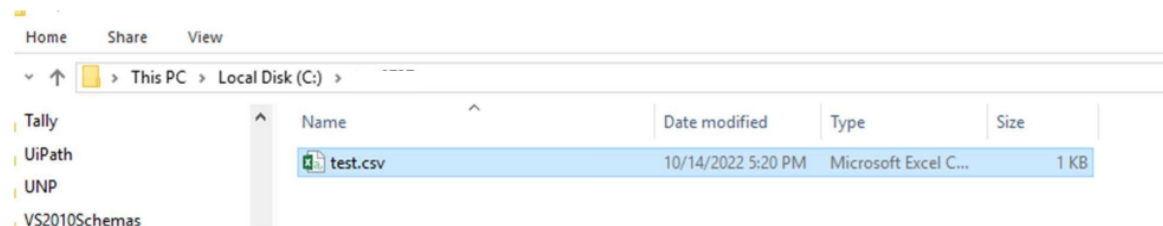
```
> use TYIT229702
switched to db TYIT229737
> db.createCollection("TYITCOLA")
{"ok":1}|
```

```
db.TYITCOLA.insertMany([ {name:'abc',class:'TYIT',rollno:11}, {name:'efg',class:'SYIT',rollno:12}, {name:'ijk',class:'FYIT',rollno:13}, {name:'mno',class:'FYCS',rollno:14}, {name:'qrs',class:'SYCS',rollno:15} ])
{"acknowledged":true,"insertedIds":{"_id":"634949feada02399cad9d64b"},
{"_id":"634949feada02399cad9d64c"},
{"_id":"634949feada02399cad9d64d"},
{"_id":"634949feada02399cad9d64e"},
{"_id":"634949feada02399cad9d64f"}]}

> use TYIT229737
switched to db TYIT229737
db.TYITCOLA.find().pretty()
{"_id":"634949feada02399cad9d64b",
"name":"abc",
"class":"TYIT",
"rollno":11},
{"_id":"634949feada02399cad9d64c",
"name":"efg",
"class":"SYIT",
"rollno":12},
{"_id":"634949feada02399cad9d64d",
"name":"ijk",
"class":"FYIT",
"rollno":13},
{"_id":"634949feada02399cad9d64e",
"name":"mno",
"class":"FYCS",
"rollno":14},
{"_id":"634949feada02399cad9d64f",
"name":"qrs",
"class":"SYCS",
"rollno":15}

C:\WINDOWS\system32\mongoexport --host localhost --db TYIT229737 --collection TYITCOLA --type=csv --out C:\test9737\test.csv --fields name,class,rollno
2022-10-14T17:20:47.012+0530 connected to: mongodb://localhost/
2022-10-14T17:20:47.056+0530 exported 5 records




C:\WINDOWS\system32>
```



Clipboard

Font

A1

na

	A	B	C	D
1	name	class	rollno	
2	abc	TYIT	11	
3	efg	SYIT	12	
4	ijk	FYIT	13	
5	mno	FYCS	14	
6	qrs	SYCS	15	
7				
8				
9				
10				

```
@(shell):1:1
> db.dropDatabase()
{ "ok":1 }
> show dbs
Employee 0.000GB
TYITDB 0.000GB
TYITDB1 0.000GB
TYITDB123 0.000GB
admin 0.000GB
config 0.000GB
jeet 0.000GB
local 0.000GB
student 0.000GB
```

```
2022-10-14T17:37:39.378+0530 0 document(s) restored successfully. 0 document(s) failed to restore.
C:\WINDOWS\system32>mongorestore c:\test229737\TYIT229737\TYITCOLA.bson
2022-10-14T17:38:45.140+0530 checking for collection data in c:\test229737\TYIT229737\TYITCOLA.bson
2022-10-14T17:38:45.143+0530 reading metadata for TYIT229737.TYITCOLA from c:\test229737\TYIT229737\TYITCOLA.metadata.json
2022-10-14T17:38:45.256+0530 restoring TYIT229737.TYITCOLA from c:\test229737\TYIT229737\TYITCOLA.bson
2022-10-14T17:38:45.303+0530 finished restoring TYIT229737.TYITCOLA (5 documents, 0 failures)
2022-10-14T17:38:45.303+0530 no indexes to restore for collection TYIT229737.TYITCOLA
2022-10-14T17:38:45.305+0530 5 document(s) restored successfully. 0 document(s) failed to restore.
C:\WINDOWS\system32>
2022-10-14T17:35:33.971+0530 writing Employee.employee to c:\test229737\
2022-10-14T17:35:33.972+0530 writing TYIT229737.TYITCOLA to c:\test229737\
```

```

local      0.000GB
student    0.000GB
> show dbs
Employee   0.000GB
TYIT229737 0.000GB
TYITDB     0.000GB
TYITDB1    0.000GB
TYITDB123  0.000GB
admin      0.000GB
config     0.000GB
jeet       0.000GB
local      0.000GB
student    0.000GB
> _

```

```

C:\WINDOWS\system32>mongoimport --db testl23 --collection TYITCOLA --file D:\restaurants.json
2022-10-14T18:07:43.205+0530 connected to: mongodb://localhost/
2022-10-14T18:07:43.599+0530 3772 document(s) imported successfully. 0 document(s) failed to import.

```

```

2022-10-14T17:35:34.037+0530 done dumping TYITDB123
--- 2022-10-14T17:35:34.043+0530 done dumping student.e
> show dbs 2022-10-14T17:35:34.053+0530 done dumping student.s
Employee 2022-10-14T17:35:34.053+0530 done dumping student.s
TYIT229737 2022-10-14T17:35:34.053+0530 writing TYITDB1.TYITCOLA
TYITDB 2022-10-14T17:35:34.053+0530 writing student.employ
TYITDB1 2022-10-14T17:35:34.055+0530 done dumping TYITDB.TYITCOLA
TYITDB123 2022-10-14T17:35:34.058+0530 done dumping TYITDB1.TYITCOLA
admin 2022-10-14T17:35:34.059+0530 done dumping student.e
config 0.000GB
jeet 0.000GB
local 0.000GB
student 0.000GB
testl23 0.001GB
> use testl23
switched to db testl23
> db.TYITCOLA.find().pretty()
{
  "2022" : {
    "id" : ObjectId("6349581748f3a6088be84a14"),
    "address" : {
      "building" : "351",

```

```

C:\Windows\system32>mongodump --host=localhost --port=27017 --out c:\test
2022-10-14T17:40:49.740+0530 writing admin.system.version to c:\test\admin\system.version.bson
2022-10-14T17:40:49.746+0530 done dumping admin.system.version (1 document)
2022-10-14T17:40:49.747+0530 writing TYIT229721.TYITCOL to c:\test\TYIT229721\TYITCOL.bson
2022-10-14T17:40:49.755+0530 done dumping TYIT229721.TYITCOL (5 documents)

```

```
MongoDB Enterprise > db.dropDatabase()
{ "dropped" : "TYIT229721", "ok" : 1 }
MongoDB Enterprise >
```

```
C:\Windows\system32>mongorestore c:\test\TYIT229721\TYITCOL.bson
2022-10-14T17:50:13.843+0530 checking for collection data in c:\test\TYIT229721\TYITCOL.bson
2022-10-14T17:50:13.844+0530 reading metadata for TYIT229721.TYITCOL from c:\test\TYIT229721\TYITCOL.metadata.json
2022-10-14T17:50:13.861+0530 restoring TYIT229721.TYITCOL from c:\test\TYIT229721\TYITCOL.bson
2022-10-14T17:50:13.903+0530 finished restoring TYIT229721.TYITCOL (5 documents, 0 failures)
2022-10-14T17:50:13.903+0530 no indexes to restore for collection TYIT229721.TYITCOL
2022-10-14T17:50:13.903+0530 5 document(s) restored successfully. 0 document(s) failed to restore.
```

```
MongoDB Enterprise > show dbs
TYIT229721 0.000GB
admin      0.000GB
config     0.000GB
local      0.000GB
```

```
MongoDB Enterprise > show dbs
TYIT229721 0.000GB
admin      0.000GB
config     0.000GB
local      0.000GB
test229721 0.001GB
MongoDB Enterprise > use test229721
switched to db test229721
MongoDB Enterprise > show collections
TYITCOL
MongoDB Enterprise > db.TYITCOL.find().pretty()
{
  "_id" : ObjectId("634957fc26f7592d655fa767"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    }
  ]
}
```

Download MongoDB Command x +

mongodb.com/try/download/database-tools

Sign in. Don't have Ahrefs? Subscribe now or check our free SEO tools. +

Web vitals

MongoDB. Products Solutions Resources Company Pricing

Sign In Try Free

Atlas
MongoDB as a service

On-premises
MongoDB locally

Tools
Boost productivity

Mobile & Edge
Realm Datastore

MongoDB Database Tools

The MongoDB Database Tools are a collection of command-line utilities for working with a MongoDB deployment. These tools release independently from the MongoDB Server schedule enabling you to receive more frequent updates and leverage new features as soon as they are available. See the [MongoDB Database Tools](#) documentation for more information.

Available Downloads

Version: 100.6.0

Platform: Windows x86_64

Package: zip

[Download](#) [Copy Link](#)

[Documentation](#)
[Archived releases](#)

```
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>mongoexport --host localhost --db TVIT229721 --collection TVITCOL --type=csv --out c:\test\test.csv
--fields name,class,rollno
2022-10-14T17:36:27.475+0530    connected to: mongodb://localhost/
2022-10-14T17:36:27.575+0530    exported 5 records

C:\Windows\system32>
```