

 Add Cover  Add Subtitle

MongoDB

How to add multiple records together?

To add multiple records at once in MongoDB we can use:

JavaScript



```
db.collectionName.insertMany([
  {
    key1: value1,
    key2: value2
  },
  {
    key1: value1,
    key2: value2,
    key3: value3,
    .
    .
    .
  },
  {
    key5: value5,
```

```
    key7: value7,  
    ...  
    ...  
}  
]);
```

```
University> db.students.insertMany([ {name: 'Sarthak', standard: 11, marks: 80}, {name: 'JD', marks: 90, age: 18}, {name: 'Tanmay', marks: 75, rollno: 2112, phone: 98998899}]);  
{  
  acknowledged: true,  
  insertedIds: {  
    '0': ObjectId("6471f4c5072f5055404eb40e"),  
    '1': ObjectId("6471f4c5072f5055404eb40f"),  
    '2': ObjectId("6471f4c5072f5055404eb410")  
  }  
}
```

Can we add an array in the JSON in MongoDB?

We can simply use [] to add an array.

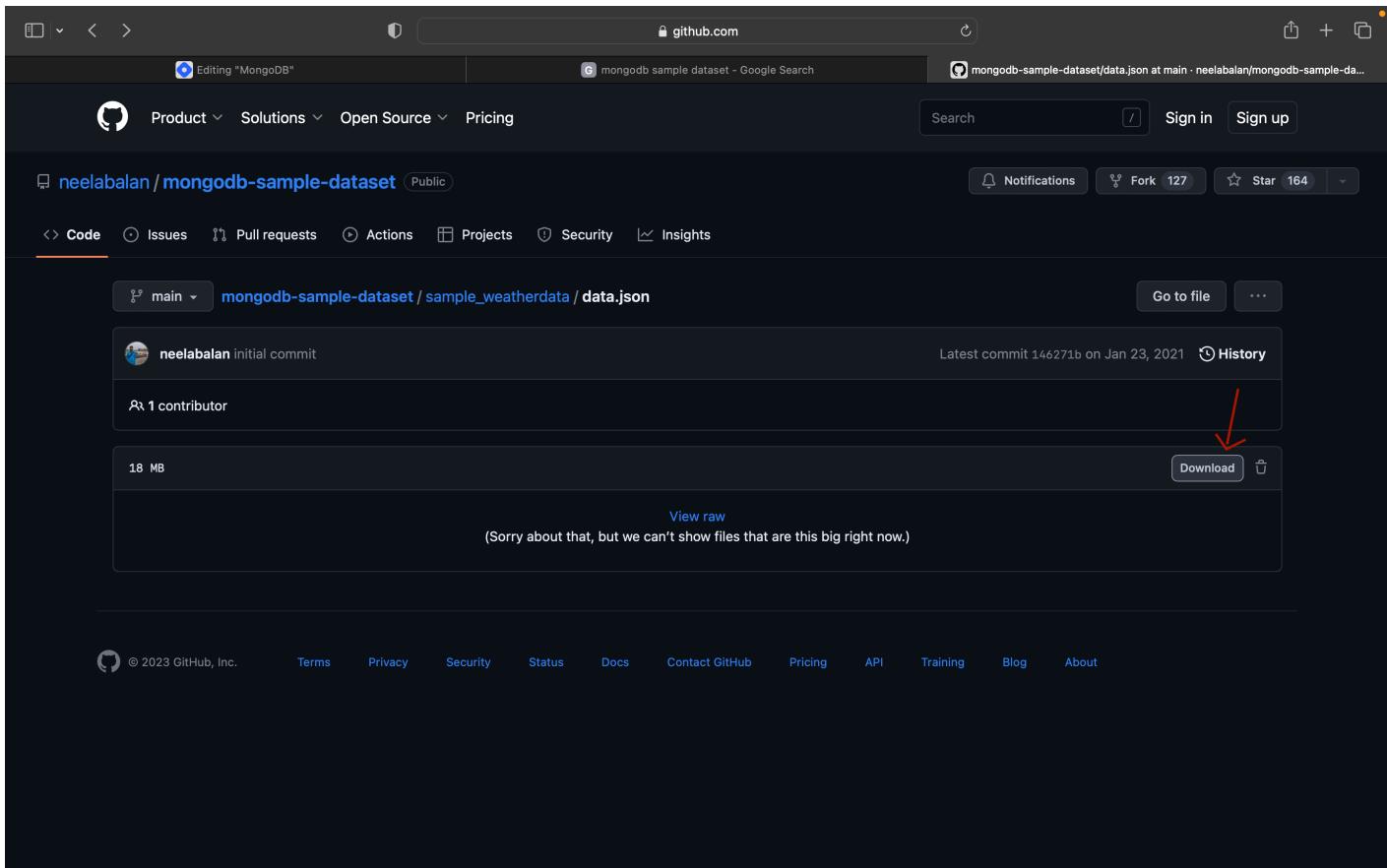
```
University> db.students.insertOne({name: 'Anurag', subjects: ["Web", "Java"]});  
{  
  acknowledged: true,  
  insertedId: ObjectId("6471f5a2072f5055404eb411")  
}
```

How to import sample DB in MongoDB?

To get some free sample datasets you can check out this Github link:

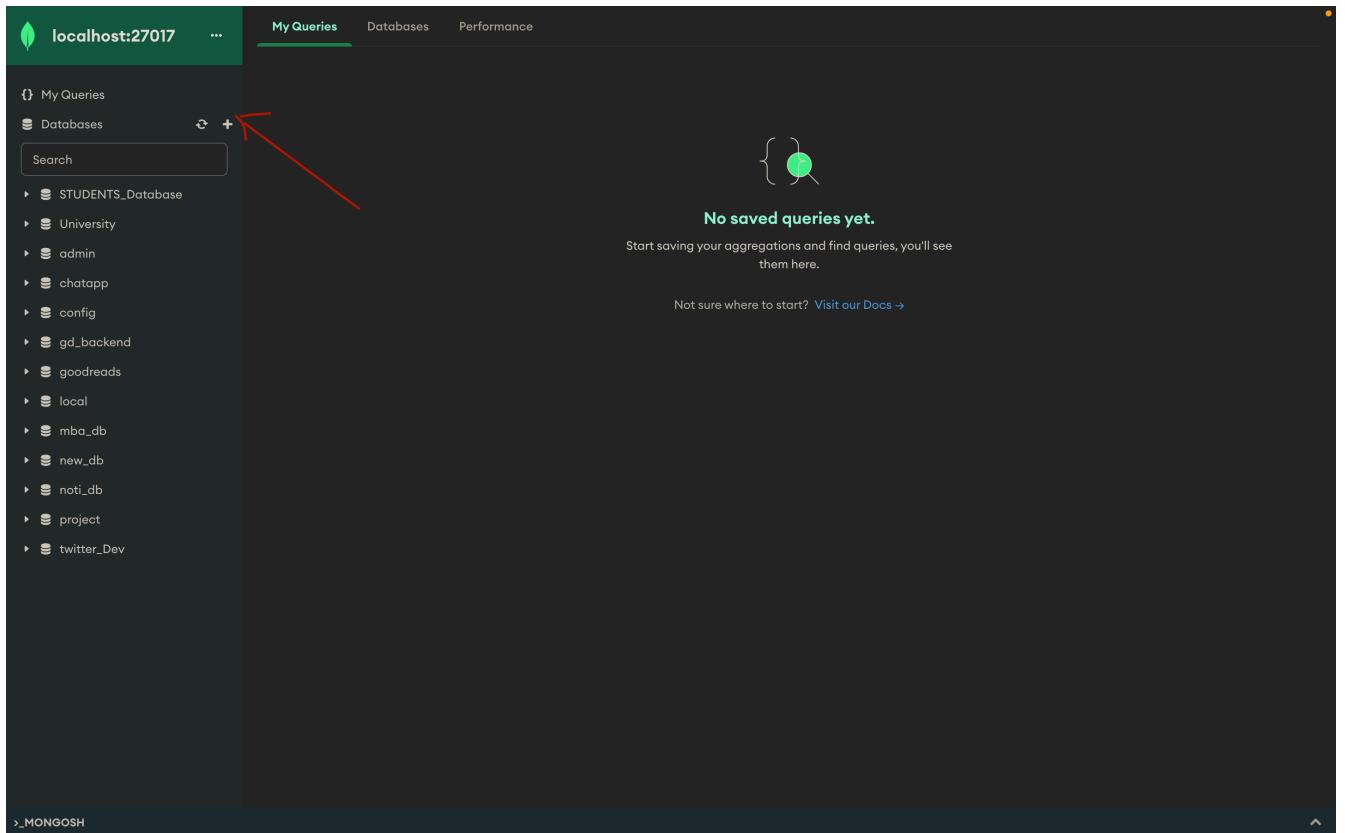
<https://github.com/neelabalan/mongodb-sample-dataset>.

You can pick any folder and download the dataset which is present in JSON format.



After the file is downloaded you can import this data using MongoDB Compass.

- Create a new Database (we can directly create it using MongoDB Compass)



localhost:27017 ... My Queries Databases Performance

My Queries Databases +

Search

STUDENTS_Database University admin chatapp config gd_backend goodreads local mba_db new_db noti_db project twitter_Dev

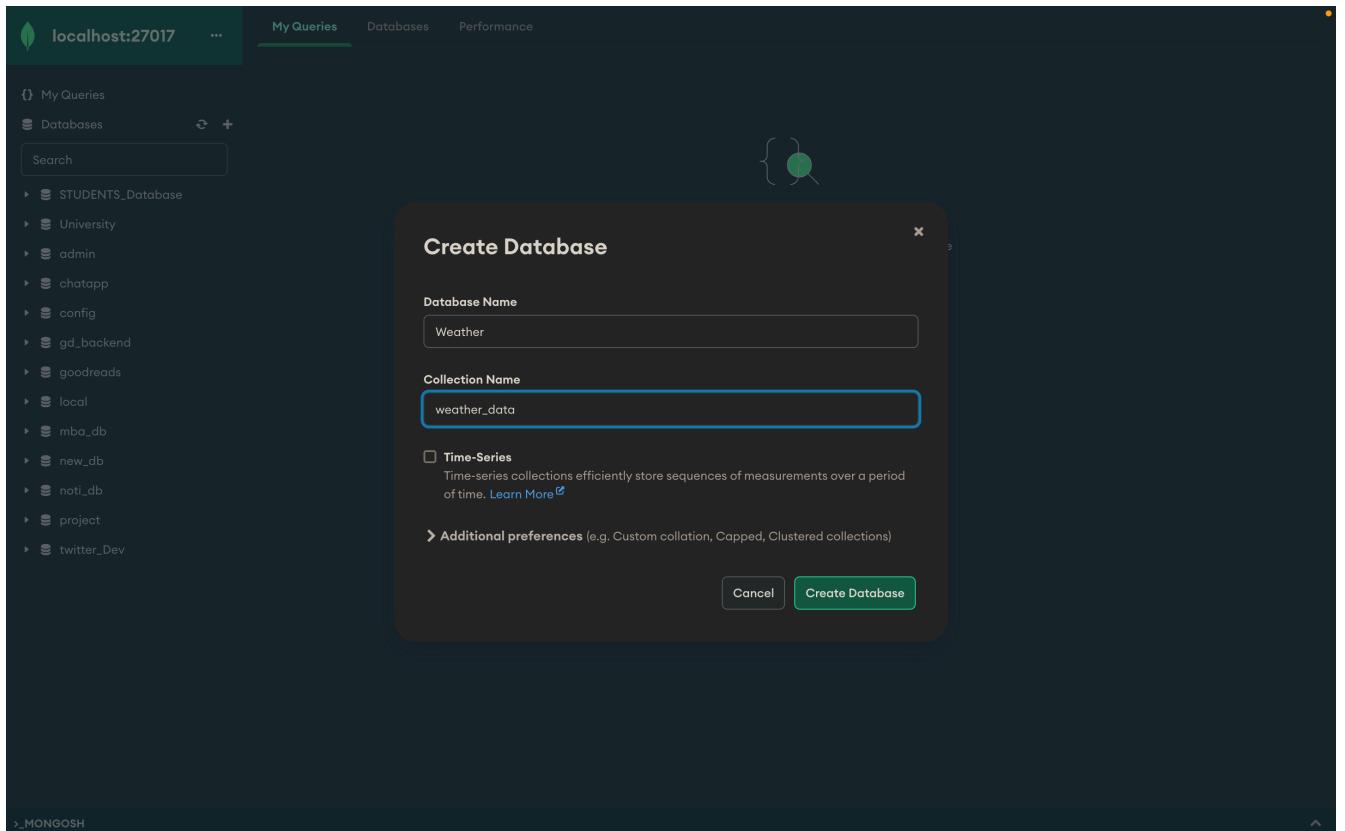
No saved queries yet.

Start saving your aggregations and find queries, you'll see them here.

Not sure where to start? [Visit our Docs →](#)

>_MONGOSH

- We can give a name to the database and give the first collection name also directly using MongoDB Compass.



localhost:27017 ... My Queries Databases Performance

My Queries Databases +

Search

STUDENTS_Database University admin chatapp config gd_backend goodreads local mba_db new_db noti_db project twitter_Dev

Create Database

Database Name: Weather

Collection Name: weather_data

Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More ↗](#)

Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Cancel Create Database

>_MONGOSH

- Once you click on create a database, a brand new empty db will be created.

The screenshot shows the MongoDB Compass application interface. At the top, there's a header bar with the URL 'localhost:27017' and a search bar. Below the header, the left sidebar lists various databases and collections, with 'weather_data' selected. The main panel displays the 'Weather.weather_data' collection. It shows '0 DOCUMENTS' and '1 INDEXES'. A message at the top of the main area says 'This collection has no data'. Below this message is a button labeled 'Import Data'. The overall interface is dark-themed.

- Now click on Add Data, which will give you two options:
 - Either you can import from JSON or CSV
 - Or you can manually insert the document.
- We will go for Option 1.

This screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases and collections, with 'weather_data' selected. The main area displays the 'Weather.weather_data' collection, which currently has 0 documents and 1 index. A red arrow points from the 'Import JSON or CSV file' button in the 'ADD DATA' dropdown menu to the 'Import Data' button at the bottom of the collection view.

- It will open the file explorer where you can select the JSON we just downloaded.

This screenshot shows the 'Import' dialog box overlaid on the MongoDB Compass interface. The dialog is titled 'Import' and specifies 'To Collection Weather.weather_data'. The 'Import file:' field contains 'data.json'. Under 'Options', there is a checkbox labeled 'Stop on errors' which is unchecked. At the bottom of the dialog are 'Cancel' and 'Import' buttons.

- Click on import and it will automatically import the data from the JSON

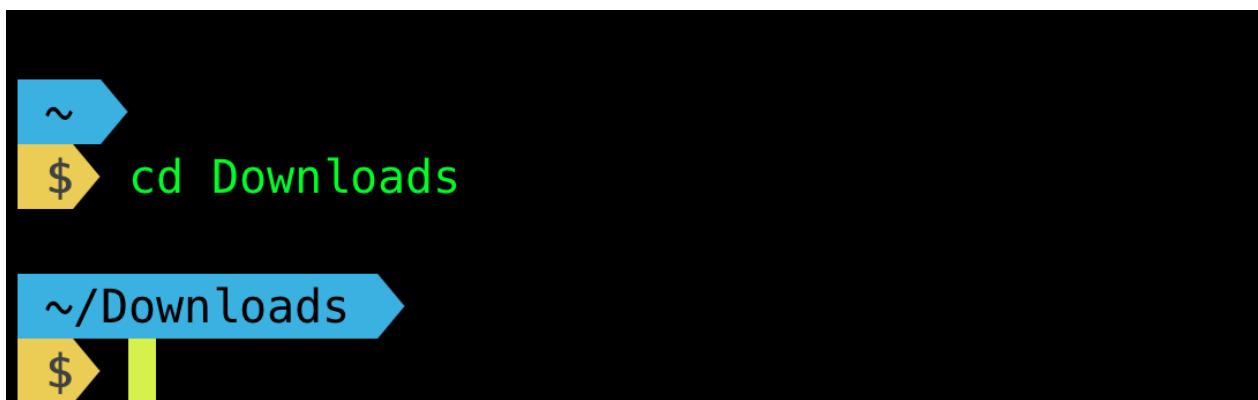
The screenshot shows the MongoDB Compass application interface. The left sidebar lists databases and collections, with 'weather_data' selected under the 'Weather' database. The main area displays the 'Weather.weather_data' collection, which currently contains 0 documents and 1 index. A message states 'This collection has no data'. Below this, a note says 'It only takes a few seconds to import data from a JSON or CSV file.' A prominent 'Import Data' button is visible. At the bottom of the screen, a progress bar shows 'Importing data.json...' with '6000 documents written.' and a 'STOP' button.

- After it finishes importing, it should look like this:

The screenshot shows the MongoDB Compass interface. On the left sidebar, there's a tree view of databases and collections, with 'weather_data' selected under the 'Weather' database. The main area is titled 'Weather.weather_data' and shows a list of documents. One document is expanded to show its fields: _id, st, ts, position, elevation, callletters, qualityControlProcess, dataSource, type, airTemperature, dewPoint, pressure, wind, visibility, skyCondition, sections, and precipitationEstimatedObservation. At the bottom of the interface, a green success message reads: 'Import completed. 10000 documents written.'

How to import without MongoDB Compass?

- We can use a terminal or CMD to import data without a compass.
- Open your terminal or CMD.
- Change the directory to the one where you have downloaded the JSON. If you are using Windows refer to this [link](#) to understand how to change the directory from cmd.



- Now we can use a MongoDB tool called as `mongoimport` (generally by default installed with MongoDB). We can use the following command:

JavaScript



```
mongoimport --db db_name --collection collection_name --
file data.json;
```

```
~/Downloads ➔ $ mongoimport --db new_weather_db --collection weather_data --file data.json;
2023-05-27T18:15:43.724+0530      connected to: mongodb://localhost/
2023-05-27T18:15:44.740+0530      10000 document(s) imported successfully. 0 document(s) failed to import.

~/Downloads ➔ $
```

- You can check the import directly from `mongosh` or compass

The screenshot shows the MongoDB Compass interface connected to the `localhost:27017` database. The `new_weather_db` is selected, and the `weather_data` collection is currently active. The interface displays the following information:

- Documents:** `new_weather_db.weather_data`
- Count:** 10.0k DOCUMENTS, 1 INDEXES
- Preview:** Shows two documents from the collection.
- Tools:** Includes buttons for `ADD DATA`, `EXPORT COLLECTION`, and various document manipulation icons.
- Left Sidebar:** Lists other databases and collections available in the system.

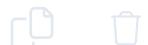
```
The monitoring data will be available on a MongoDB website with a unique URL accessible to you  
and anyone you share the URL with. MongoDB may use this information to make product  
improvements and to suggest MongoDB products and deployment options to you.  
  
To enable free monitoring, run the following command: db.enableFreeMonitoring()  
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()  
----  
  
Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.  
You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.  
test> show dbs;  
STUDENTS_Database    88.00 KiB  
University           72.00 KiB  
Weather              2.49 MiB  
admin                40.00 KiB  
chatapp              72.00 KiB  
config               108.00 KiB  
gd_backend           416.00 KiB  
goodreads            416.00 KiB  
local                104.00 KiB  
mba_db               504.00 KiB  
new_db               85.89 MiB  
new_weather_db        2.57 MiB  
noti_db              72.00 KiB  
project              76.00 KiB  
testing               2.55 MiB  
twitter_Dev          360.00 KiB  
test> use new_weather_db  
switched to db new_weather_db  
new_weather_db> show collections  
weather_data  
new_weather_db>
```

Let's explore the DB we imported

How to see the count of documents in a collection?

There is a `count` function that we can use

JavaScript

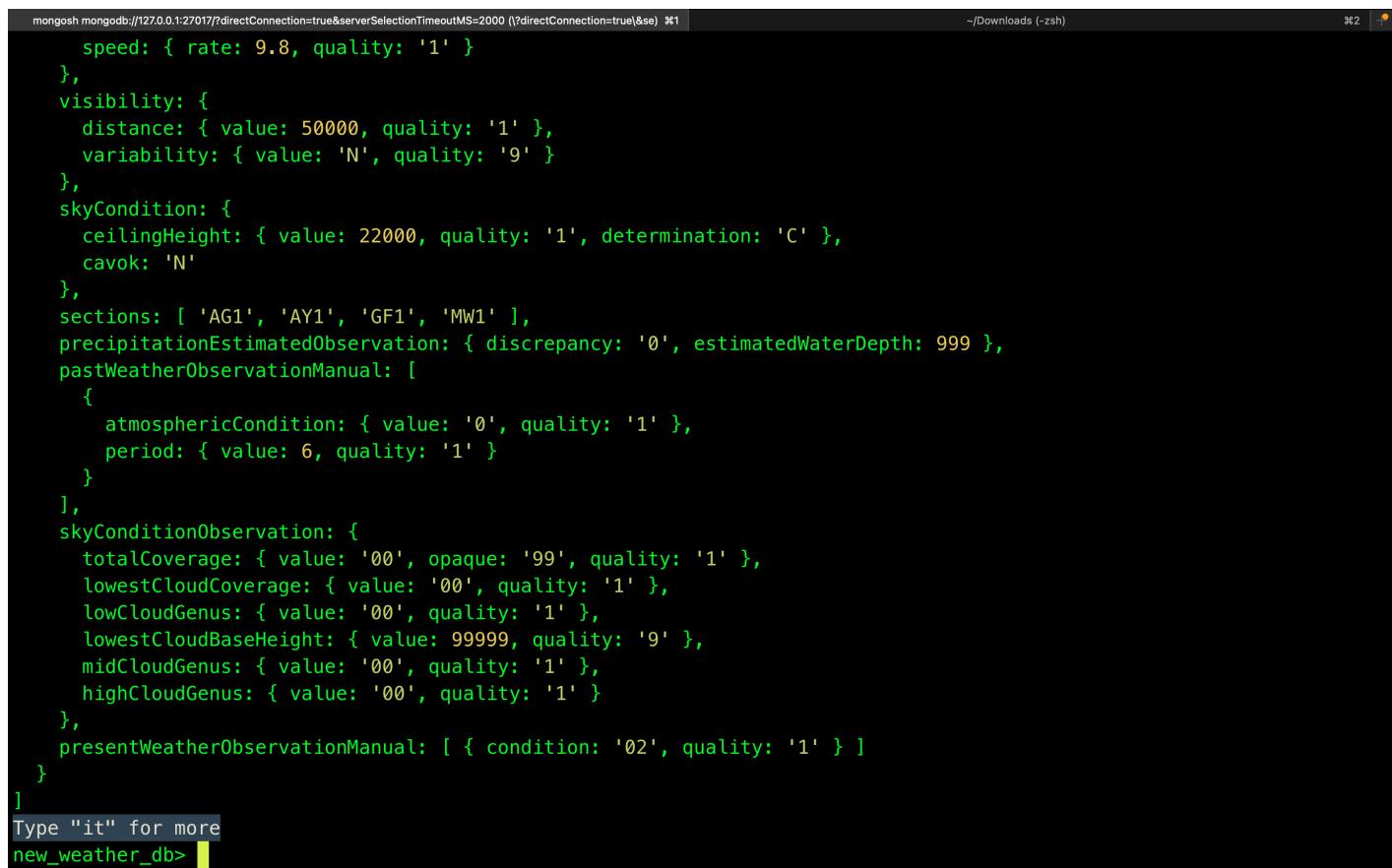


```
db.collectionName.find().count();
```

```
new_weather_db> db.weather_data.find().count()  
(node:88641) [MONGODB DRIVER] Warning: cursor.count is deprecated and will be removed in the next major version, please  
use `collection.estimatedDocumentCount` or `collection.countDocuments` instead  
(Use `node --trace-warnings ...` to show where the warning was created)  
10000  
new_weather_db> █
```

How to handle the printing of huge amounts of data?

If you have a huge amount of data, just like we have here approx 10,000 documents, if you try to execute `db.collectionName.find` in your `mongosh` then will not print all the 10,000 records, because it cannot handle that in the shell. If you try it then in the console, it will print some data and then gives you a prompt of `Try "it" for more`



A screenshot of a terminal window titled "mongosh" showing a large JSON document. The document describes weather conditions, including speed, visibility, sky conditions, sections, precipitation, and past weather observations. It uses nested objects and arrays to represent various parameters like distance, variability, and atmospheric conditions over time periods. The terminal shows the end of the document with "[Type "it" for more]" and a prompt "new_weather_db>".

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 (!?directConnection=true&se) ✘1
{
  speed: { rate: 9.8, quality: '1' },
  visibility: {
    distance: { value: 50000, quality: '1' },
    variability: { value: 'N', quality: '9' }
  },
  skyCondition: {
    ceilingHeight: { value: 22000, quality: '1', determination: 'C' },
    cavok: 'N'
  },
  sections: [ 'AG1', 'AY1', 'GF1', 'MW1' ],
  precipitationEstimatedObservation: { discrepancy: '0', estimatedWaterDepth: 999 },
  pastWeatherObservationManual: [
    {
      atmosphericCondition: { value: '0', quality: '1' },
      period: { value: 6, quality: '1' }
    }
  ],
  skyConditionObservation: {
    totalCoverage: { value: '00', opaque: '99', quality: '1' },
    lowestCloudCoverage: { value: '00', quality: '1' },
    lowCloudGenus: { value: '00', quality: '1' },
    lowestCloudBaseHeight: { value: 99999, quality: '9' },
    midCloudGenus: { value: '00', quality: '1' },
    highCloudGenus: { value: '00', quality: '1' }
  },
  presentWeatherObservationManual: [ { condition: '02', quality: '1' } ]
}
]
Type "it" for more
new_weather_db>
```

If you write `it` and press enter then you will get next group of data.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 (!?directConnection=true&se) #1 ~ /Downloads (-zsh) 362


```

precipitationEstimatedObservation: { discrepancy: '2', estimatedWaterDepth: 0 },
pastWeatherObservationManual: [
 {
 atmosphericCondition: { value: '0', quality: '1' },
 period: { value: 6, quality: '1' }
 }
],
skyConditionObservation: {
 totalCoverage: { value: '08', opaque: '99', quality: '1' },
 lowestCloudCoverage: { value: '08', quality: '1' },
 lowCloudGenus: { value: '06', quality: '1' },
 lowestCloudBaseHeight: { value: 450, quality: '1' },
 midCloudGenus: { value: '99', quality: '9' },
 highCloudGenus: { value: '99', quality: '9' }
},
atmosphericPressureChange: {
 tendency: { code: '7', quality: '1' },
 quantity3Hours: { value: 0.6, quality: '1' },
 quantity24Hours: { value: 99.9, quality: '9' }
},
presentWeatherObservationManual: [{ condition: '02', quality: '1' }],
seaSurfaceTemperature: { value: 15.5, quality: '9' },
waveMeasurement: {
 method: 'M',
 waves: { period: 3, height: 1, quality: '9' },
 seaState: { code: '99', quality: '9' }
}
}
]
Type "it" for more
new_weather_db> it
```


```

How to get a certain number of documents?

We can use a function called as limit

JavaScript



```
db.collectionName.find().limit(no_of_records_to_fetch);
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 (!?directConnection=true&se) ✘1
new_weather_db> db.weather_data.find().limit(3)
[
  {
    _id: ObjectId("5553a998e4b02cf7151190b9"),
    st: 'x+45200-066500',
    ts: ISODate("1984-03-05T14:00:00.000Z"),
    position: { type: 'Point', coordinates: [ -66.5, 45.2 ] },
    elevation: 9999,
    callLetters: 'VC81',
    qualityControlProcess: 'V020',
    dataSource: '4',
    type: 'FM-13',
    airTemperature: { value: -4.7, quality: '1' },
    dewPoint: { value: 999.9, quality: '9' },
    pressure: { value: 1025.9, quality: '1' },
    wind: {
      direction: { angle: 999, quality: '9' },
      type: '9',
      speed: { rate: 999.9, quality: '9' }
    },
    visibility: {
      distance: { value: 999999, quality: '9' },
      variability: { value: 'N', quality: '9' }
    },
    skyCondition: {
      ceilingHeight: { value: 99999, quality: '9', determination: '9' },
      cavok: 'N'
    },
    sections: [ 'AG1' ],
    precipitationEstimatedObservation: { discrepancy: '2', estimatedWaterDepth: 999 }
  },
]
```

How to set an offset while querying data?

We can set an offset using the `skip` function. Using the `skip` function we can skip some records and then start fetching records post the `skip`.

JavaScript



```
db.collectionName.find().skip(5).limit(3)
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 (!?directConnection=true&se) #1
new_weather_db> db.weather_data.find().skip(10).limit(2);
[ {
  _id: ObjectId("5553a998e4b02cf7151190b8"),
  st: 'x+47600-047900',
  ts: ISODate("1984-03-05T13:00:00.000Z"),
  position: { type: 'Point', coordinates: [ -47.9, 47.6 ] },
  elevation: 9999,
  callLetters: 'VCSZ',
  qualityControlProcess: 'V020',
  dataSource: '4',
  type: 'FM-13',
  airTemperature: { value: -3.1, quality: '1' },
  dewPoint: { value: 999.9, quality: '9' },
  pressure: { value: 1015.3, quality: '1' },
  wind: {
    direction: { angle: 999, quality: '9' },
    type: '9',
    speed: { rate: 999.9, quality: '9' }
  },
  visibility: {
    distance: { value: 999999, quality: '9' },
    variability: { value: 'N', quality: '9' }
  },
  skyCondition: {
    ceilingHeight: { value: 99999, quality: '9', determination: '9' },
    cavok: 'N'
  },
  sections: [ 'AG1' ],
  precipitationEstimatedObservation: { discrepancy: '2', estimatedWaterDepth: 999 }
},
```



Saved



Upgrade

Preview

Publish

To filter the records, we can pass an object in the arguments of the `find` function, where we can add our conditions.

JavaScript



```
db.collectionName.find({key1: value1, key2: value2})
```

```
new_weather_db> db.weather_data.find({type: 'FM-13'}).count()
9994
new_weather_db>
```

Projections

If we want to not get all the properties of the JSON, and instead get some specific key-value pairs, this process is called Projection. In the world of SQL, if you do `SELECT * FROM TABLE;` then you get all the columns but if you do `SELECT name, address FROM TABLE;` you only get name and address. This same thing is achieved in Projections.

How to do projections?

So the first argument of the `find` function is an object which takes filtration criteria. It can accept another argument as an object, where we can write whatever properties we have to include and assign them a value true.

JavaScript



```
db.collectionName.find({filter1: value1...}, {property1: true,  
property2: true....});
```

```
new_weather_db> db.weather_data.find({type: 'FM-13'}, {position: true, visibility: true});
```

You can also pass the first argument as an empty JSON object.

```
new_weather_db> db.weather_data.find({}, {position: true, visibility: true});
```

If we want to manually exclude specific properties, you can write their names with false value allocated:

```
new_weather_db> db.weather_data.find({}, {skyConditionObservation: false, pastWeatherObservationManual: false});
```

Now this will bring everything apart from `pastWeatherObservationManual` and `skyConditionObservation`

How to delete a document?

If we want to delete some documents we can use functions like `deleteOne`, `deleteMany` and `findByIdAndDelete`.

JavaScript



```
db.collectionName.deleteOne({filter1: value1, filter2:  
value2..});
```

```
new_weather_db> db.weather_data.deleteOne({st: 'x-19300+060300'})  
{ acknowledged: true, deletedCount: 1 }  
new_weather_db> db.weather_data.find().count()  
9999  
new_weather_db> █
```

JavaScript



```
db.collectionName.deleteMany({filter1: value1, filter2:  
value2..});
```

```
new_weather_db> db.weather_data.find({callLetters: 'FNPG'}).count()
7
new_weather_db> db.weather_data.deleteMany({callLetters: 'FNPG'})
{ acknowledged: true, deletedCount: 7 }
new_weather_db> db.weather_data.find().count()
9992
new_weather_db>
```

```
new_weather_db> db.weather_data.deleteOne({_id: ObjectId("5553a998e4b02cf7151190b9")})
{ acknowledged: true, deletedCount: 1 }
new_weather_db> db.weather_data.find({_id: ObjectId("5553a998e4b02cf7151190b9")})

new_weather_db>
```

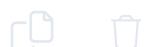
We can similarly use `findOneAndDelete` to filter the data and then delete one record it. [Docs](#)

How to update a record?

To Update records we can use `updateOne` , `updateMany` and a few similar functions. These functions take two arguments,

1. Filtration criteria viz. how to filter what data to update
2. With what value we should update

JavaScript



```
db.collectionName.updateOne({filter1: value1}, {$operator:
{key1: value1, key2: value2...}})
```

Now MongoDB provides us some operators for these updates for example:

- **\$set** -> This will allocate the value to the key directly passed in the object
- **\$inc** -> This will increment the value in the key

```
new_weather_db> db.weather_data.updateOne({_id: ObjectId("5553a998e4b02cf7151190c0")}, {$set: {elevation: 9998}});  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

```
new_weather_db> db.weather_data.updateOne({_id: ObjectId("5553a998e4b02cf7151190c0")}, {$inc: {elevation: 1}});  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

To decrease we can do `$inc` with a negative value.

```
new_weather_db> db.weather_data.updateOne({_id: ObjectId("5553a998e4b02cf7151190c0")}, {$inc: {elevation: -1}});  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

If you will use `updateMany` then all the records which are complying to the filtration criteria will be update.