MongoDB

- Here, the data is written in the format of the <key> <value> pairs.
- It stores the data in the format of the JSON Objects
- It stores the data in the format of BSON *{Binary + JSON (JavaScript Object Notation)}* format.
- Nornal JSON: Means the Data is stored in the function but it is not used usually.

- In the SQL document means the “ROW and COLUMN”
- Unique ID is of the 12 Bytes (4 Bytes “Timestamp” + 5 Bytes “MAC Address” + 3 Counter values – Random Value)

Installation:

- First install the MongoDB Compass tool:
    - Click Here to go to the website
- Second install the MongoDB CLI tool:
    - Click Here to go to the website

Now, install in the PC.

After installing the Applications just start:

- sudo systemctl start mongod

Now,

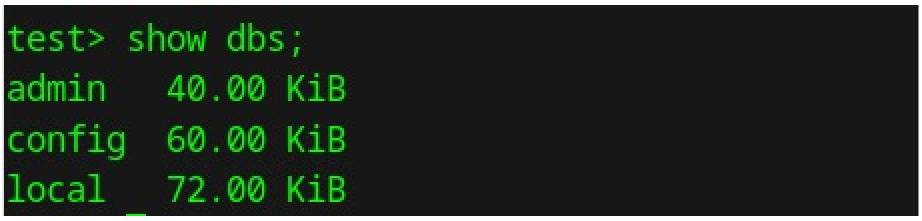
just type the:

- mongodb-compass

It will open the Mongo’s GUI and then just click on the connect

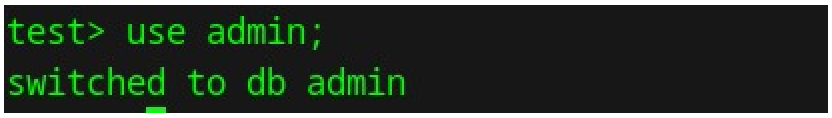
Commands:

- show dbs;
    - It will display the Databases available inside the Server.



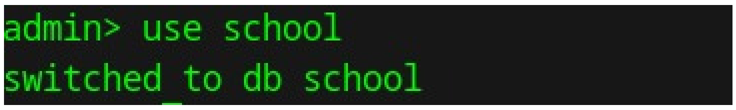
```
test> show dbs;
admin    40.00 KiB
config   60.00 KiB
local    72.00 KiB
```

- use admin;
    - Now it switched to the admin database.



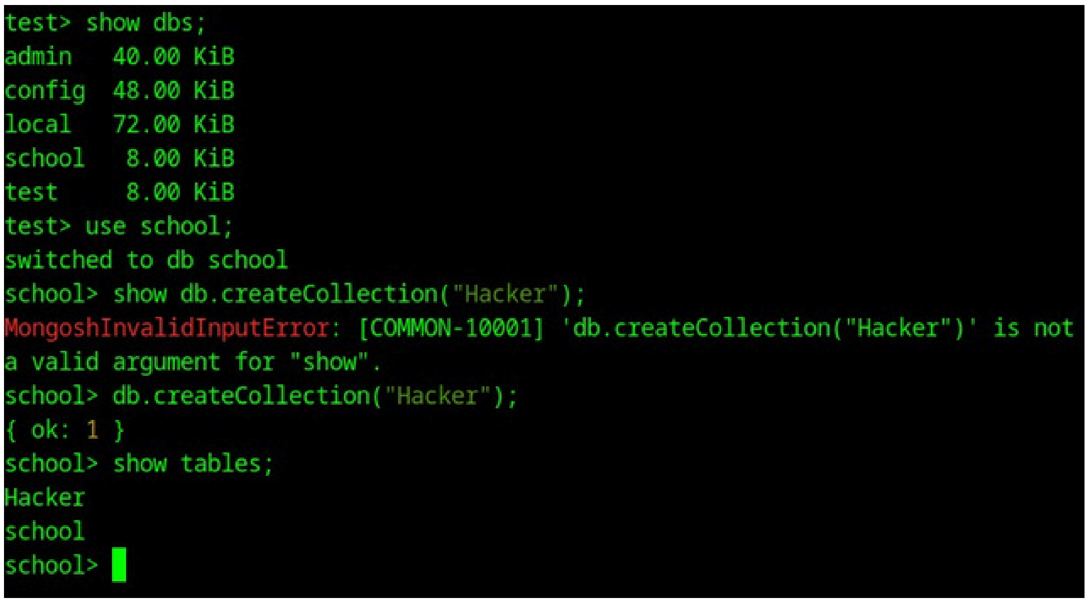
```
test> use admin;
switched to db admin
```

- use school; *#(Here, "school" named database is not available then it will create)*



```
admin> use school
switched to db school
```

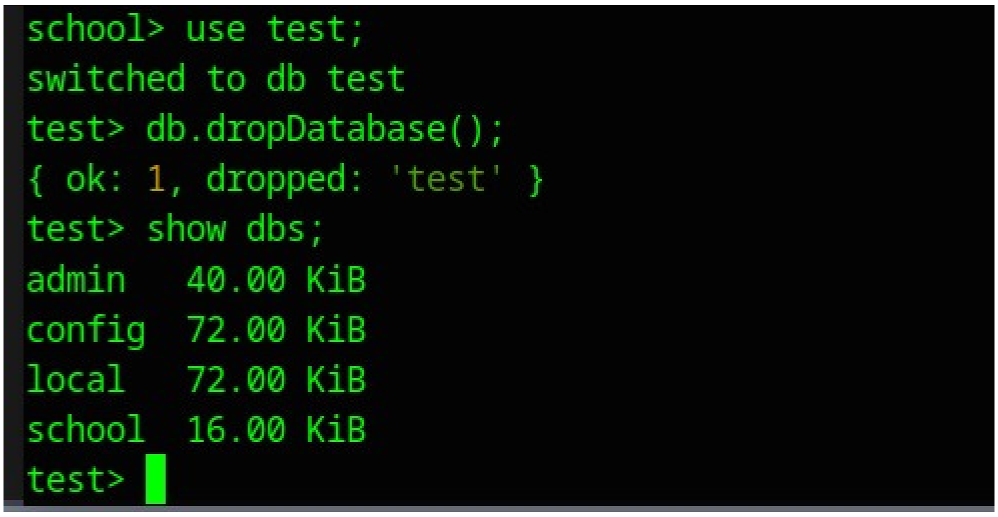
  - It will create a separate database with name “school”
- db.createCollection("Database_name");
    - It will create a new **table** with the name:
        - E.g.:
            - db.createCollection("Hacker");



```
test> show dbs;
admin    40.00 KiB
config   48.00 KiB
local    72.00 KiB
school    8.00 KiB
test      8.00 KiB
test> use school;
switched to db school
school> show db.createCollection("Hacker");
MongoshInvalidInputError: [COMMON-10001] 'db.createCollection("Hacker")' is not
a valid argument for "show".
school> db.createCollection("Hacker");
{ ok: 1 }
school> show tables;
Hacker
school
school>
```

Delete the database by going inside in that database:

- db.dropDatabase();



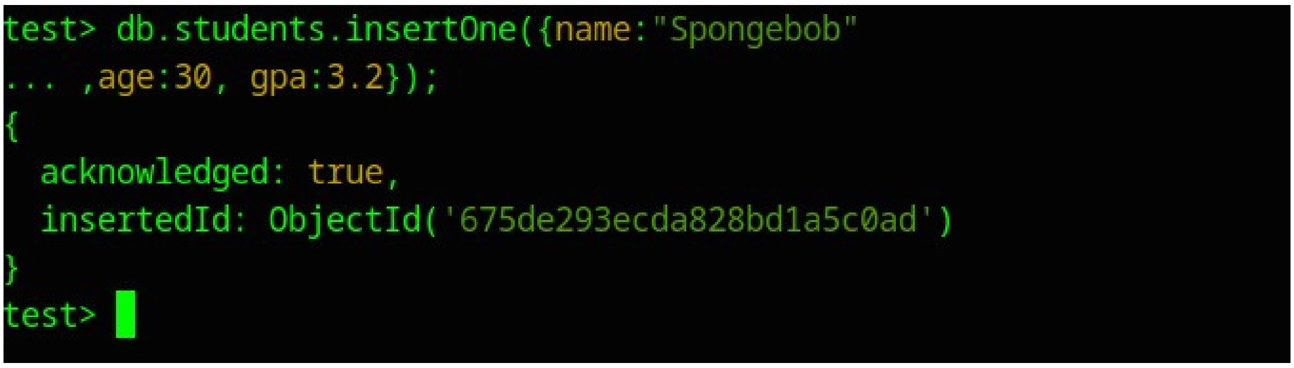
```
school> use test;
switched to db test
test> db.dropDatabase();
{ ok: 1, dropped: 'test' }
test> show dbs;
admin    40.00 KiB
config   72.00 KiB
local    72.00 KiB
school   16.00 KiB
test>
```

- Here, you can see that first we switched the table and then we drop down the table.

Insert the Document in MongoDB:

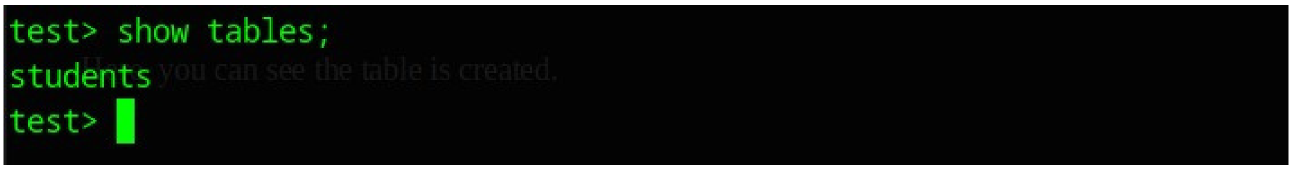
Now, we will make a table automatically and inserting the data:

- db.Database_name.insertOne({parameter:"String", parameter:int, ...});
    - E.g.:
        - db.students.insertOne({name: "Spongebob", age: 30, gpa: 3.2});
    - Here, we have created a table named as Students with different item.



```
test> db.students.insertOne({name:"Spongebob"
... ,age:30, gpa:3.2});
{
  acknowledged: true,
  insertedId: ObjectId('675de293ecda828bd1a5c0ad')
}
test>
```

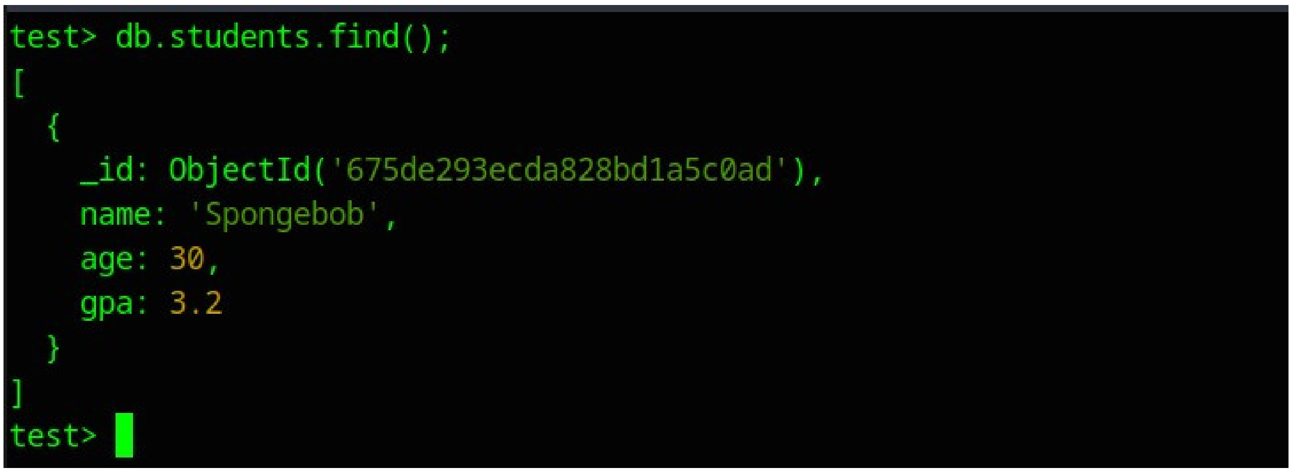
- Here, you can see the table is created.



```
test> show tables;
students
test>
```

- Now, the display command for displaying the data inside the Table:

- db.Database_name,findOne(); *#findOne() is only for showing one object.*
  - E.g.:
    - db.students.findOne();

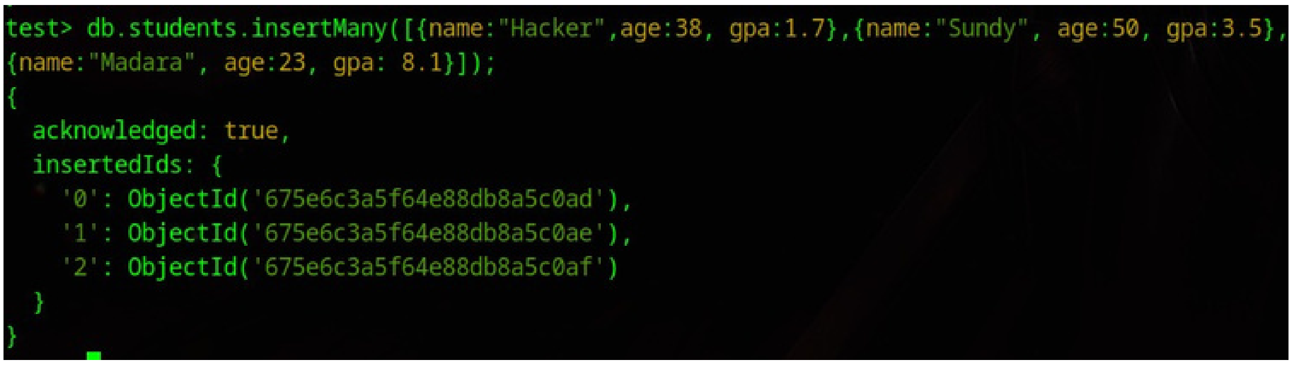


```
test> db.students.find();
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2
  }
]
test> 
```

- Here, we are displaying the content inside the table.

Now, we are inserting multiple queries at a time.

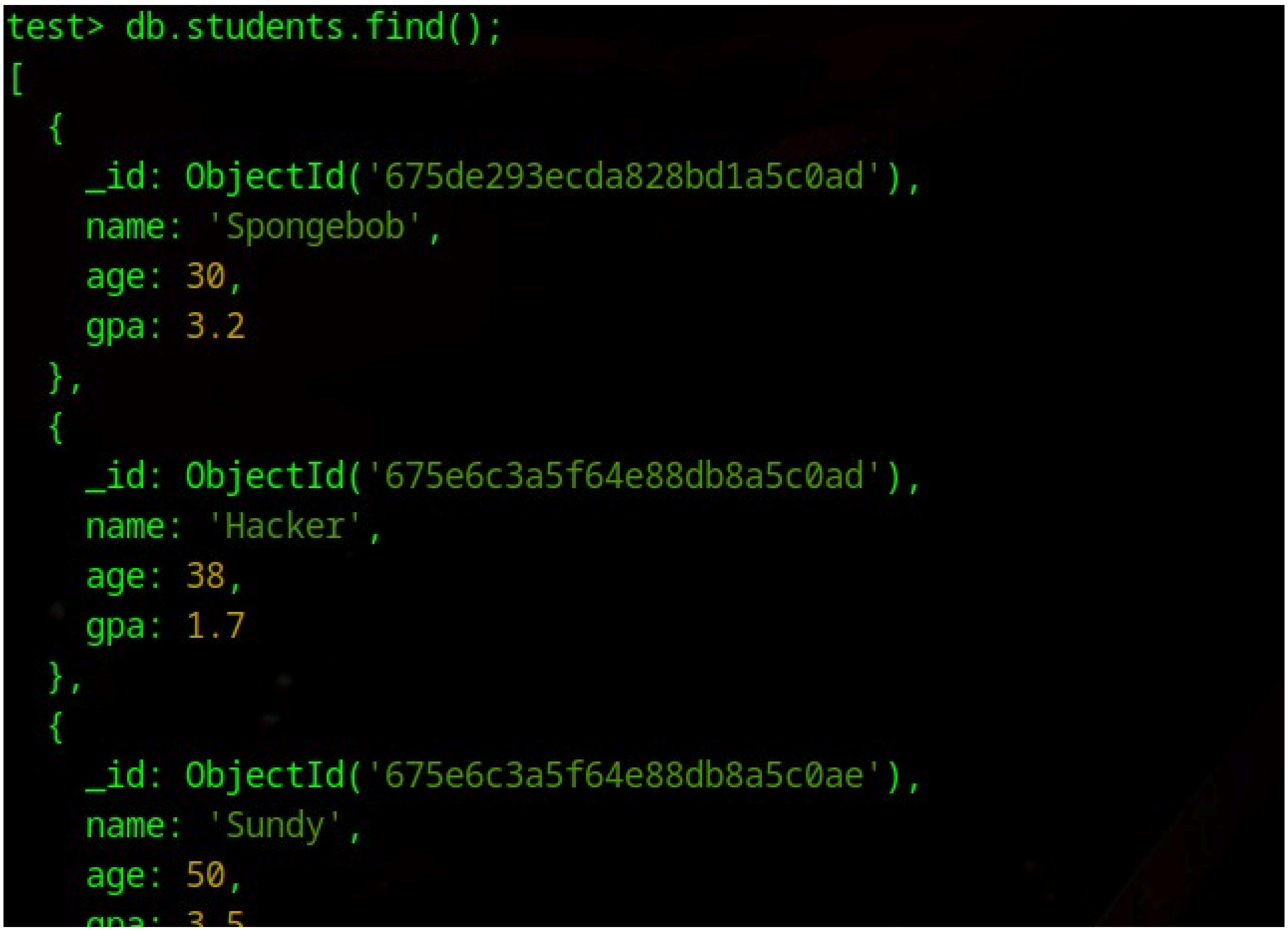
- db.Table_name.insertMany([{},{},{}]);
  - E.g.:
    - db.students.insertMany([{name:"Hacker",age:38, gpa:1.7},{name:"Sundy", age:50, gpa:3.5},{name:"Madara", age:23, gpa: 8.1}]);



```
test> db.students.insertMany([{name:"Hacker",age:38, gpa:1.7},{name:"Sundy", age:50, gpa:3.5},
{name:"Madara", age:23, gpa: 8.1}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('675e6c3a5f64e88db8a5c0ad'),
    '1': ObjectId('675e6c3a5f64e88db8a5c0ae'),
    '2': ObjectId('675e6c3a5f64e88db8a5c0af')
  }
}
```

Now, display the data from the table.

- db.table_name.find(); *#It will show all the data from the database*



```
test> db.students.find();
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ad'),
    name: 'Hacker',
    age: 38,
    gpa: 1.7
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5
```

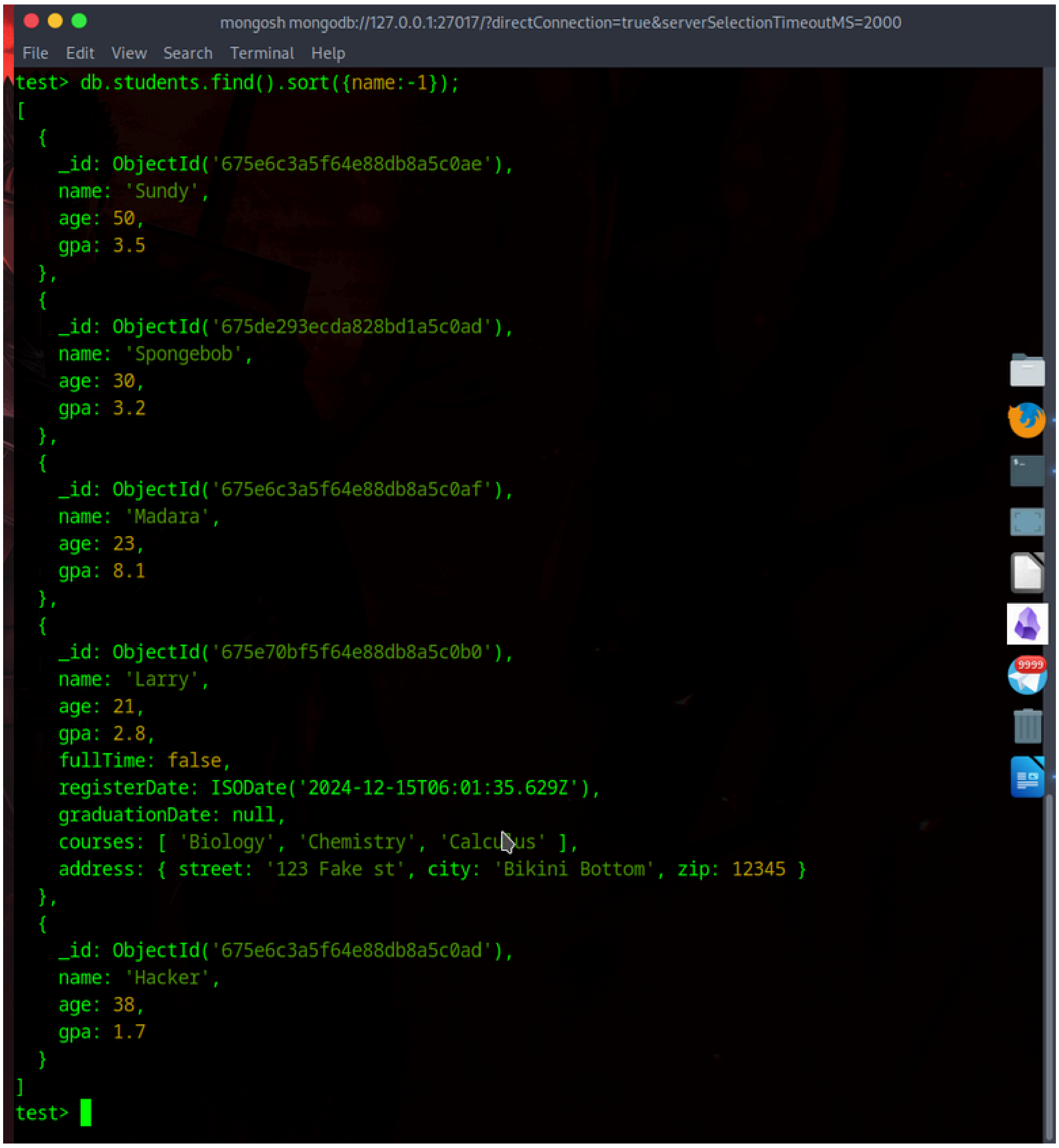
Data types:

```
db.students.insertOne({name: "Larry123", #It is a String, and you can put anything
                                        age: 32, #It is an Integer number, here only int value
                                         gpa: 2.6, #It is a float value or double value
                                                   fullTime: false, #It is a Boolean type value
                                                  registerDate: new Date(), #Here, we are taking the live date without
passing the arguments
                                                   graduationDate: null, #It means no value
                                                   courses: ["Biology", "Chemistry", "Calculus"], #It is an Array type
data
#Now, we are using the nested query.
                                                   address: {street: "123 Fake st",
                                                                     city: "Bikini Bottom",
                                                                      zip: 12345}});
```

- Sorting and Limiting:

Now, sorting:

- db.table_name.find().sort({tuple_name:1 || -1});
    - Here, 1 is for ascending order and -1 is for descending order.
        - E.g.:
            - db.students.find().sort({name:-1}); *#It will print the details in the reverse order.*

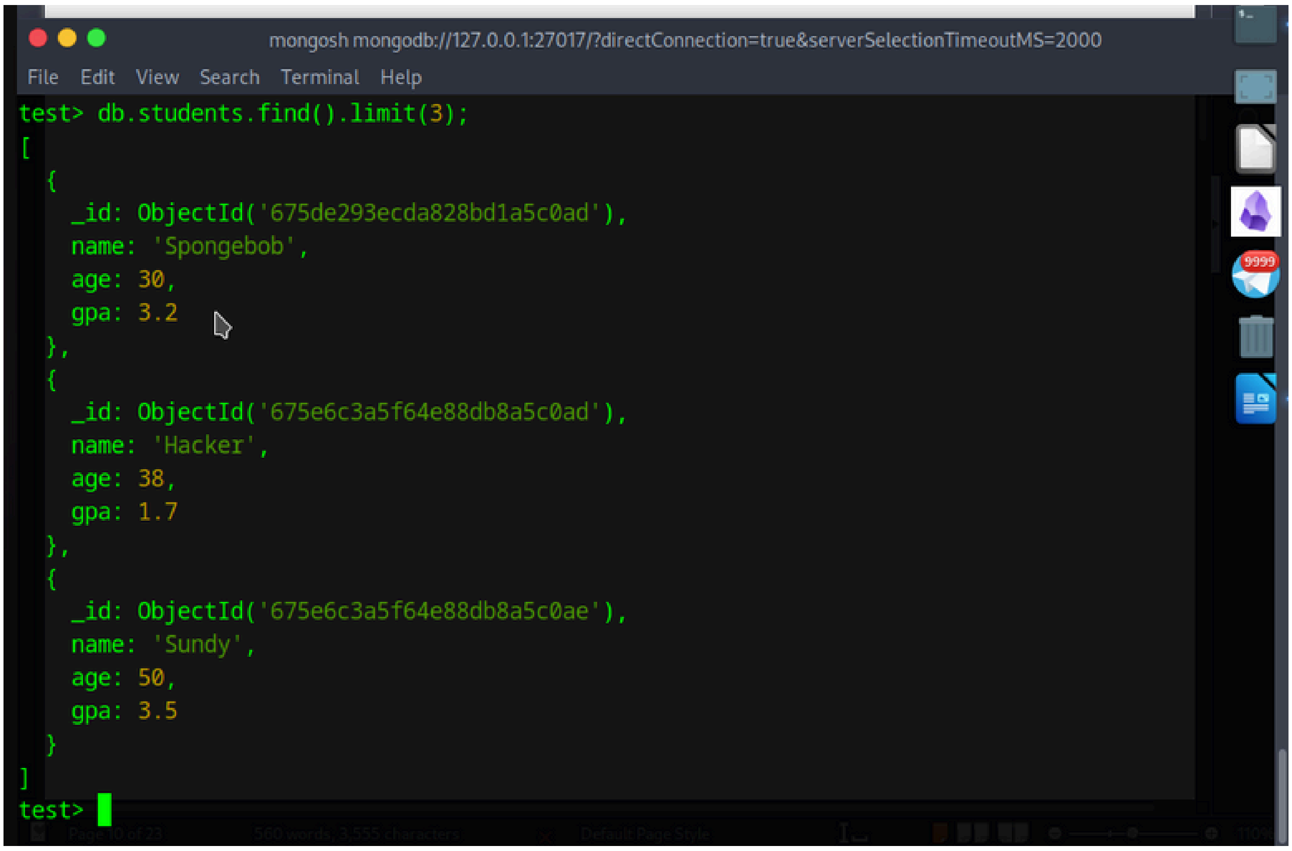

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
File Edit View Search Terminal Help

```
test> db.students.find().sort({name:-1});
[
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5
  },
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0af'),
    name: 'Madara',
    age: 23,
    gpa: 8.1
  },
  {
    _id: ObjectId('675e70bf5f64e88db8a5c0b0'),
    name: 'Larry',
    age: 21,
    gpa: 2.8,
    fullTime: false,
    registerDate: ISODate('2024-12-15T06:01:35.629Z'),
    graduationDate: null,
    courses: [ 'Biology', 'Chemistry', 'Calculus' ],
    address: { street: '123 Fake st', city: 'Bikini Bottom', zip: 12345 }
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ad'),
    name: 'Hacker',
    age: 38,
    gpa: 1.7
  }
]
test> 
```

- To get in the ascending order just change -1 to 1.

- Limit() method:

- db.table_name.find().limit(1,2,3,.....n);
    - It only provide the no. of the output where the limit's value is equal to the limit's
    - It provides with reference to the Object ID.
        - E.g.:
            - db.students.find().limit(3);

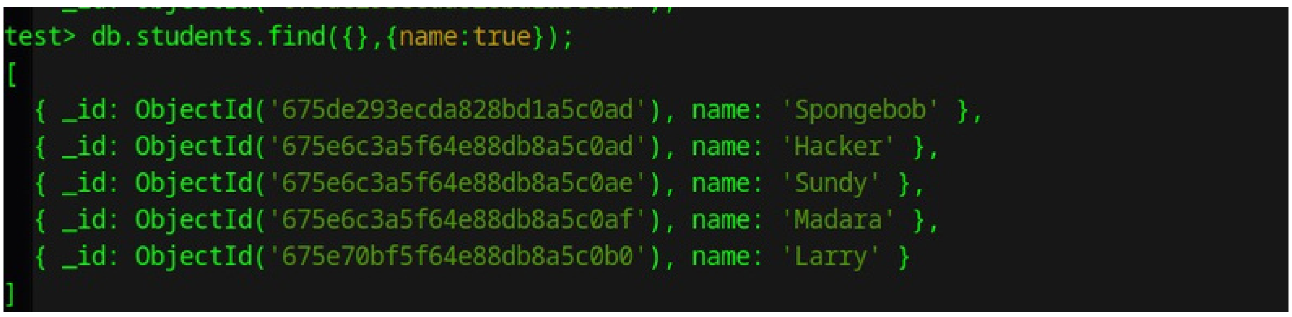

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
File Edit View Search Terminal Help

```
test> db.students.find().limit(3);
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ad'),
    name: 'Hacker',
    age: 38,
    gpa: 1.7
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5
  }
]
test> 
```

- Here, you can see that the list is sorted in order with respective to the ObjectID.\

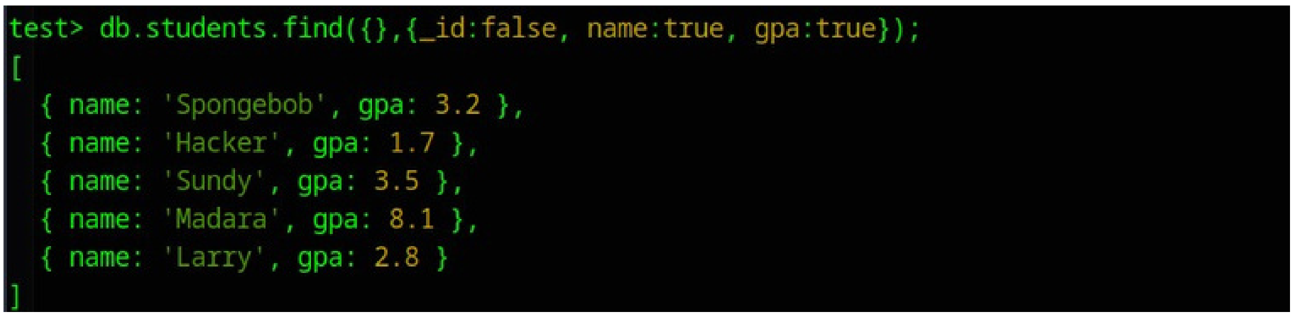
Now, there is one more is Query Parameter:

- db.collectionName.fine({query},{projection});
    - With this projectiion parameter we will get the exact data
    - E.g.:
        - db.students.find({},{name:**true**});
    - Here, you'll get the list of the only **name** and **_id's**



```
test> db.students.find({},{name:true});
[
  { _id: ObjectId('675de293ecda828bd1a5c0ad'), name: 'Spongebob' },
  { _id: ObjectId('675e6c3a5f64e88db8a5c0ad'), name: 'Hacker' },
  { _id: ObjectId('675e6c3a5f64e88db8a5c0ae'), name: 'Sundy' },
  { _id: ObjectId('675e6c3a5f64e88db8a5c0af'), name: 'Madara' },
  { _id: ObjectId('675e70bf5f64e88db8a5c0b0'), name: 'Larry' }
]
```

- We can type whatever the data that we want to fetch just type the tuple name:



```
test> db.students.find({},{_id:false, name:true, gpa:true});
[
  { name: 'Spongebob', gpa: 3.2 },
  { name: 'Hacker', gpa: 1.7 },
  { name: 'Sundy', gpa: 3.5 },
  { name: 'Madara', gpa: 8.1 },
  { name: 'Larry', gpa: 2.8 }
]
```

- Here, is how you can fetch the values from the data.

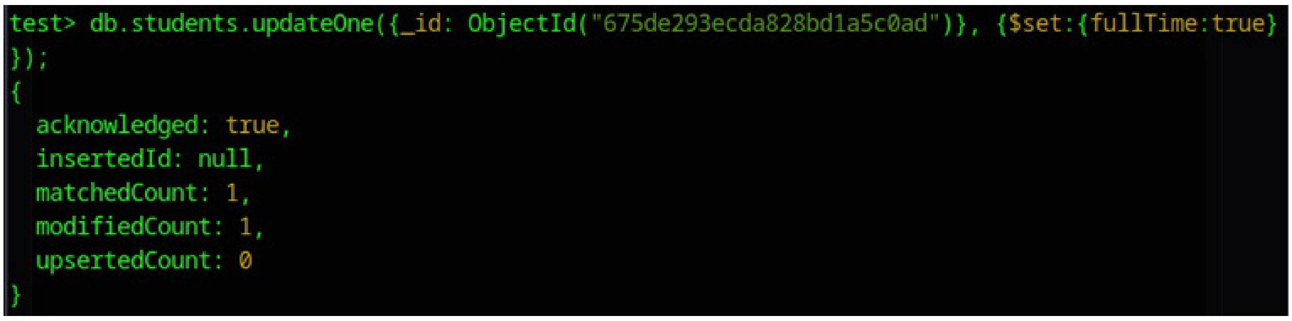
Updating the Data:

- db.collectionName.updateOne({***variableId that you want to update***}, {**$set**:{***Set the new values according to your need***}});

- Here, the value is getting added or updated:

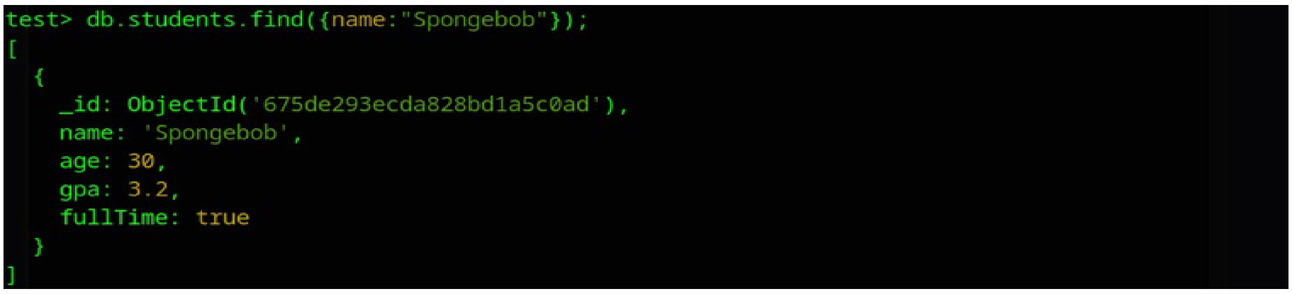
  - E.g.:

    - db.students.updateOne({_id: ObjectId("675de293ecda828bd1a5c0ad")}, {$set:{fullTime:true}});



```
test> db.students.updateOne({_id: ObjectId("675de293ecda828bd1a5c0ad")}, {$set:{fullTime:true}
});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

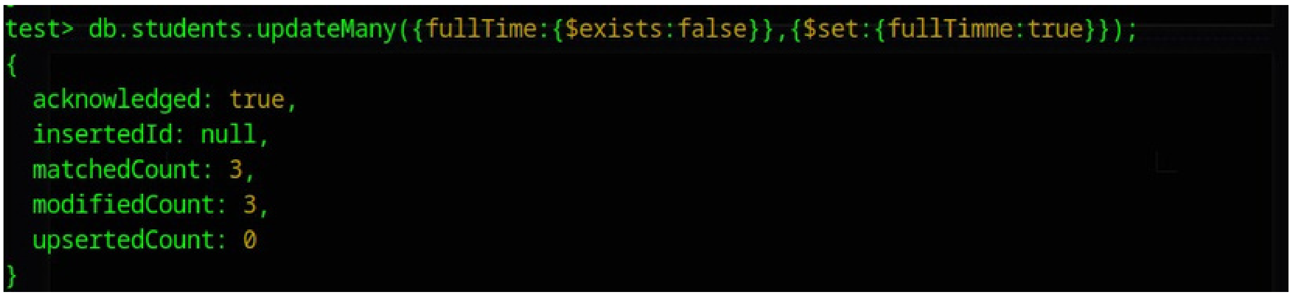
- Now, just see the updated collection:



```
test> db.students.find({name:"Spongebob"});
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2,
    fullTime: true
  }
]
```

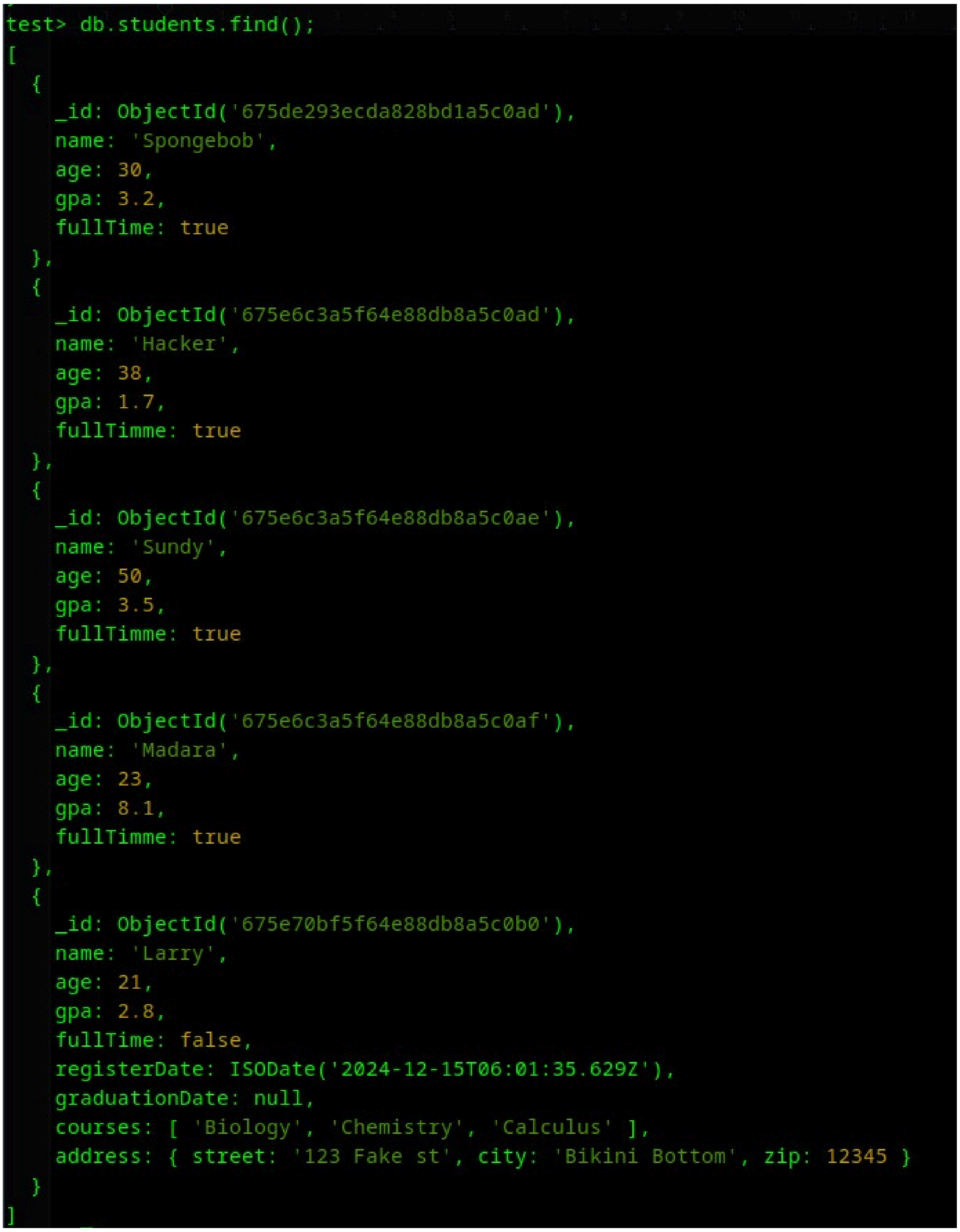
Now, we are using the updateMany method to update the things at a time:

- db.collectionName.updateMany(**{variable_name:{to_get_to_be_updated}}**,{$set:{set_the_value_as_per_your_choice}}); *#Here, it takes first argument to check or is to be updated or not, and in the second argument it is* ***getting updated.***



```
test> db.students.updateMany({fullTime:{$exists:false}},{$set:{fullTimme:true}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

- Here, is the input:
    - db.students.updateMany({fullTime:{$exists:false}},{$set:{fullTimme:true}});



```
test> db.students.find();
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2,
    fullTime: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ad'),
    name: 'Hacker',
    age: 38,
    gpa: 1.7,
    fullTimme: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5,
    fullTimme: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0af'),
    name: 'Madara',
    age: 23,
    gpa: 8.1,
    fullTimme: true
  },
  {
    _id: ObjectId('675e70bf5f64e88db8a5c0b0'),
    name: 'Larry',
    age: 21,
    gpa: 2.8,
    fullTime: false,
    registerDate: ISODate('2024-12-15T06:01:35.629Z'),
    graduationDate: null,
    courses: [ 'Biology', 'Chemistry', 'Calculus' ],
    address: { street: '123 Fake st', city: 'Bikini Bottom', zip: 12345 }
  }
]
```

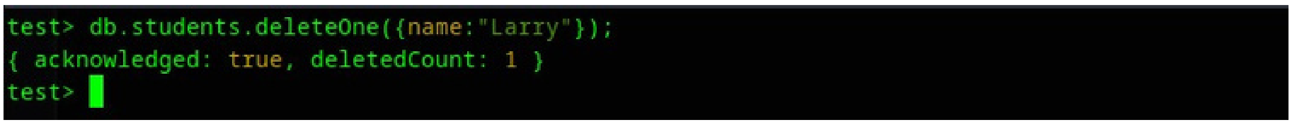
Delete documents in the MongoDB:

- Now, we will learn that how to delete a document in the MongoDB shell.

- It is easy in the MongoDB – Compass.

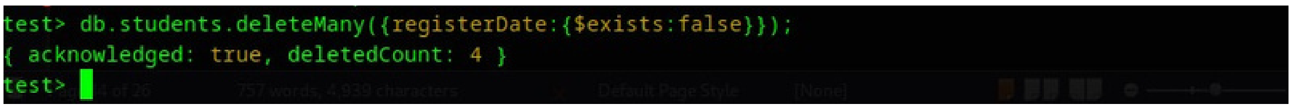
- db.collectionName.deleteOne/deleteMany({specific arguments});

    - db.students.deleteOne({name: "Larry"});



```
test> db.students.deleteOne({name:"Larry"});
{ acknowledged: true, deletedCount: 1 }
test> 
```

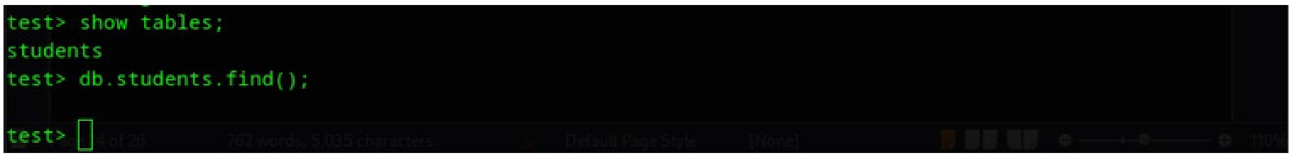
    - db.students.deleteMany({registerDate:{$exists:false}});



```
test> db.students.deleteMany({registerDate:{$exists:false}});
{ acknowledged: true, deletedCount: 4 }
test> 
```

- Now, check is there is any data is remembered or stored by the student:

- show tables;
- db.students.find();



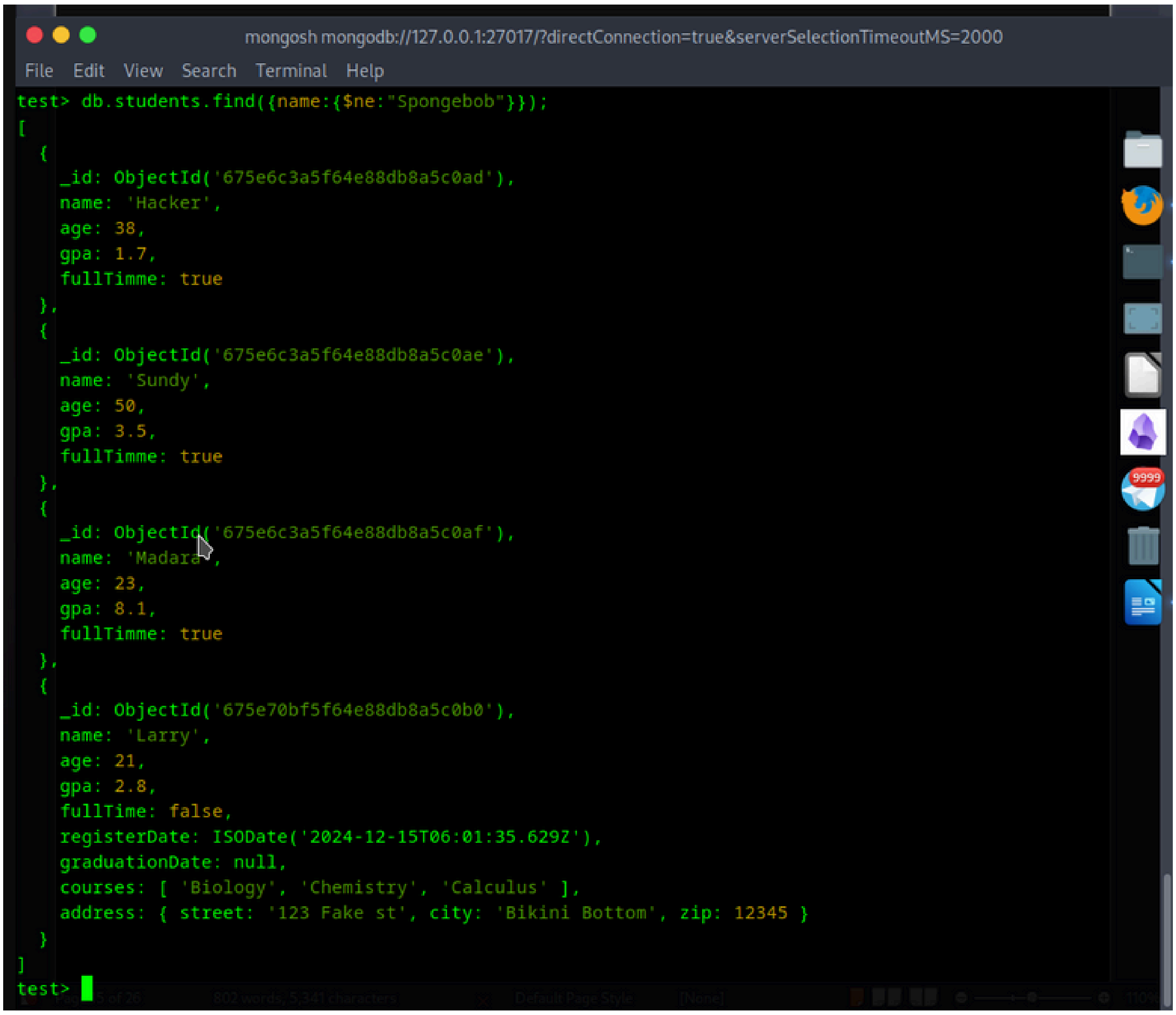
```
test> show tables;
students
test> db.students.find();

test> 
```

Comparison Query Operators:

Comparison operators return data based on value comparisons.

Not Equal to operator: ($ne) -

- db.collectionName.find({variableNameThatYouWantToFind:{$ne: "Variables_Value"}});

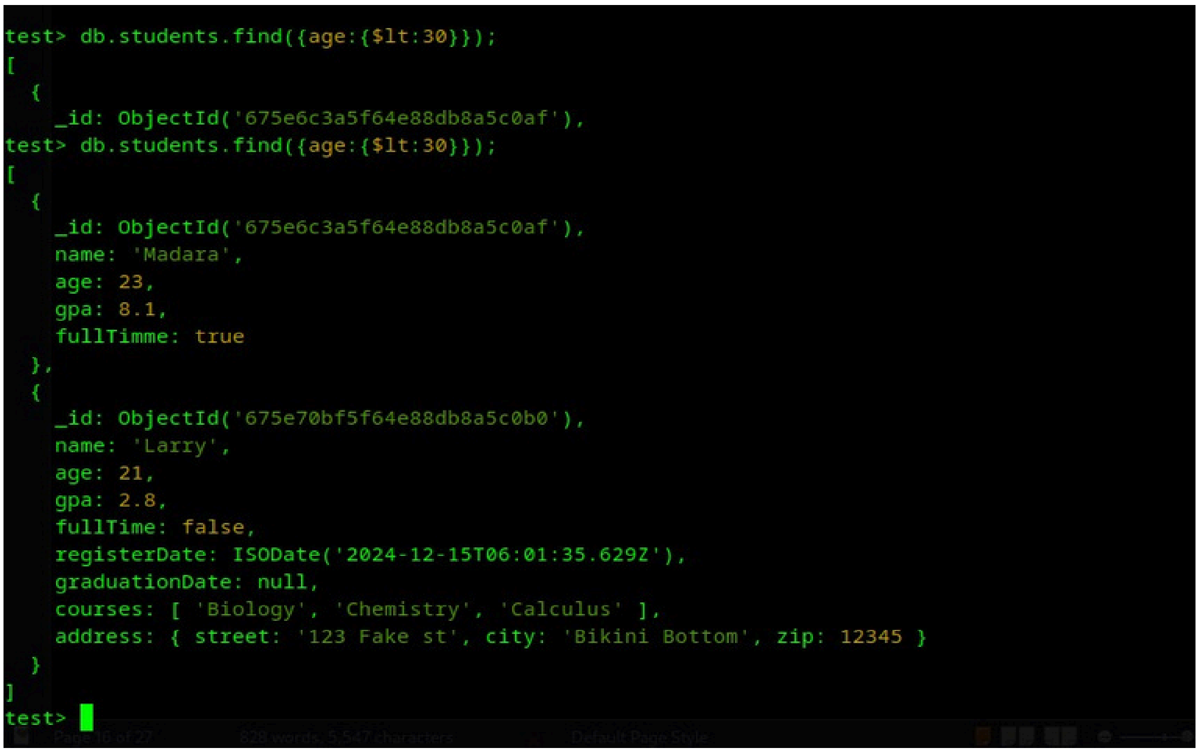

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
File Edit View Search Terminal Help

```
test> db.students.find({name:{$ne:"Spongebob"}});
[
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ad'),
    name: 'Hacker',
    age: 38,
    gpa: 1.7,
    fullTimme: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5,
    fullTimme: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0af'),
    name: 'Madara',
    age: 23,
    gpa: 8.1,
    fullTimme: true
  },
  {
    _id: ObjectId('675e70bf5f64e88db8a5c0b0'),
    name: 'Larry',
    age: 21,
    gpa: 2.8,
    fullTime: false,
    registerDate: ISODate('2024-12-15T06:01:35.629Z'),
    graduationDate: null,
    courses: [ 'Biology', 'Chemistry', 'Calculus' ],
    address: { street: '123 Fake st', city: 'Bikini Bottom', zip: 12345 }
  }
]
test> 
```

- Here, this will provide the name is not equal to the "**Spongebob**"

Less than *($lt)* or Greater than *($gt)* or combination operator:

- db.**collectionName**.find({variableName:{$lt:conditionsValue}});
    - db.students.find({age:{$lt:30}});

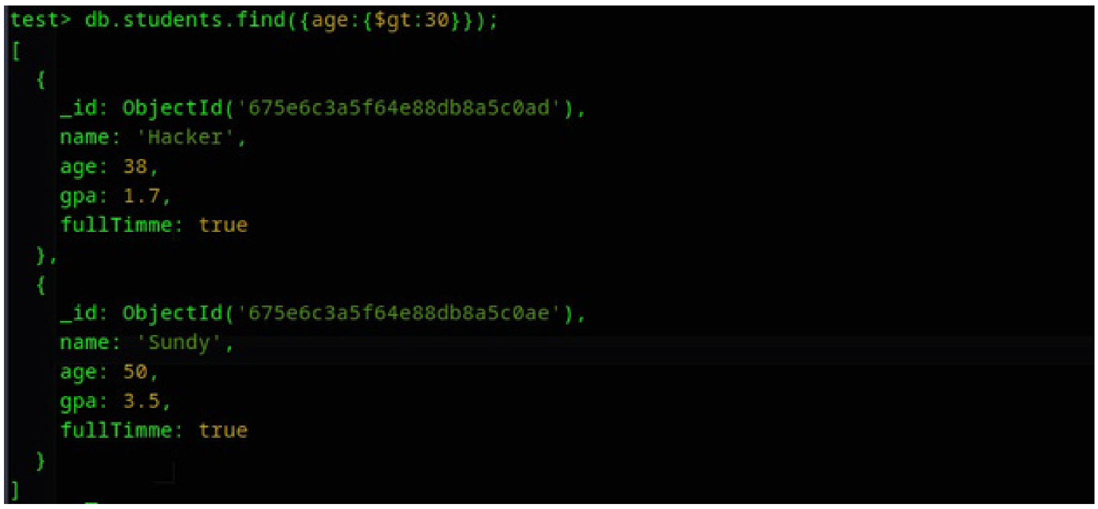


```
test> db.students.find({age:{$lt:30}});
[
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0af'),
test> db.students.find({age:{$lt:30}});
[
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0af'),
    name: 'Madara',
    age: 23,
    gpa: 8.1,
    fullTimme: true
  },
  {
    _id: ObjectId('675e70bf5f64e88db8a5c0b0'),
    name: 'Larry',
    age: 21,
    gpa: 2.8,
    fullTime: false,
    registerDate: ISODate('2024-12-15T06:01:35.629Z'),
    graduationDate: null,
    courses: [ 'Biology', 'Chemistry', 'Calculus' ],
    address: { street: '123 Fake st', city: 'Bikini Bottom', zip: 12345 }
  }
]
test> 
```

- Here, you can see the values less then the 30 ages.

Now, for the greater than the values:

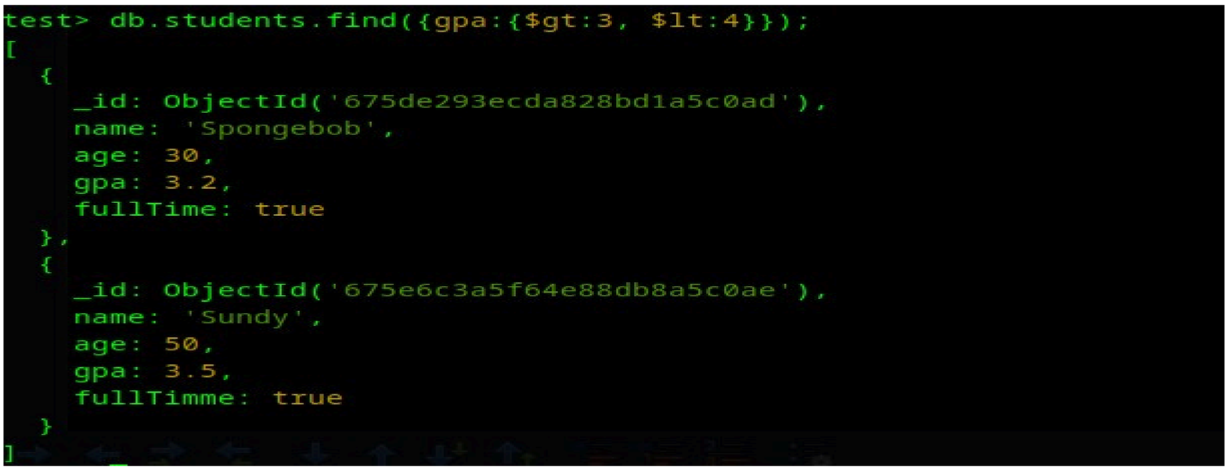
- db.**collectionName**.find({variableName:{$gt:conditionsValue}});
    - db.students.find({age:{$gt:30}});



```
test> db.students.find({age:{$gt:30}});
[
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ad'),
    name: 'Hacker',
    age: 38,
    gpa: 1.7,
    fullTimme: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5,
    fullTimme: true
  }
]
```

In-between operator:

- db.collectionName.find({variableName:{$gt:conditionsValue, $lt:another_condition}});
    - db.students..find({gpa:{$gt:3, $lt:4}});
- AND **for** greater than or less than equals to we will use the: **$gte** AND **$lte**



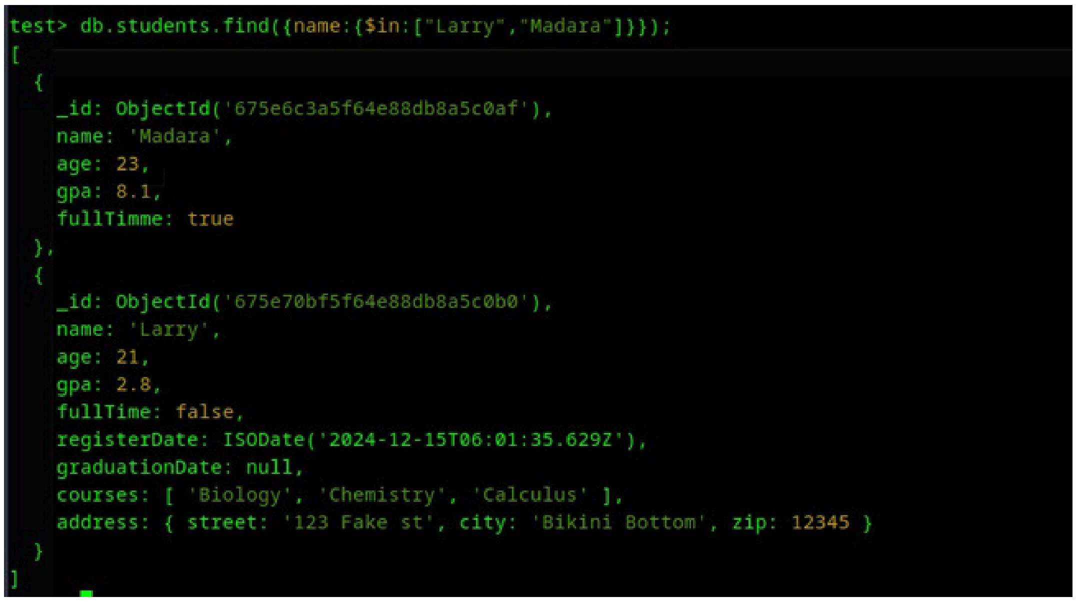
```
test> db.students.find({gpa:{$gt:3, $lt:4}});
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2,
    fullTime: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5,
    fullTimme: true
  }
]
```

in Operator:

- db.collectionName.find({variableName:{$in:["conditionsValue"]}});

- Here, we will get the only the values that is getting to be matched or not:

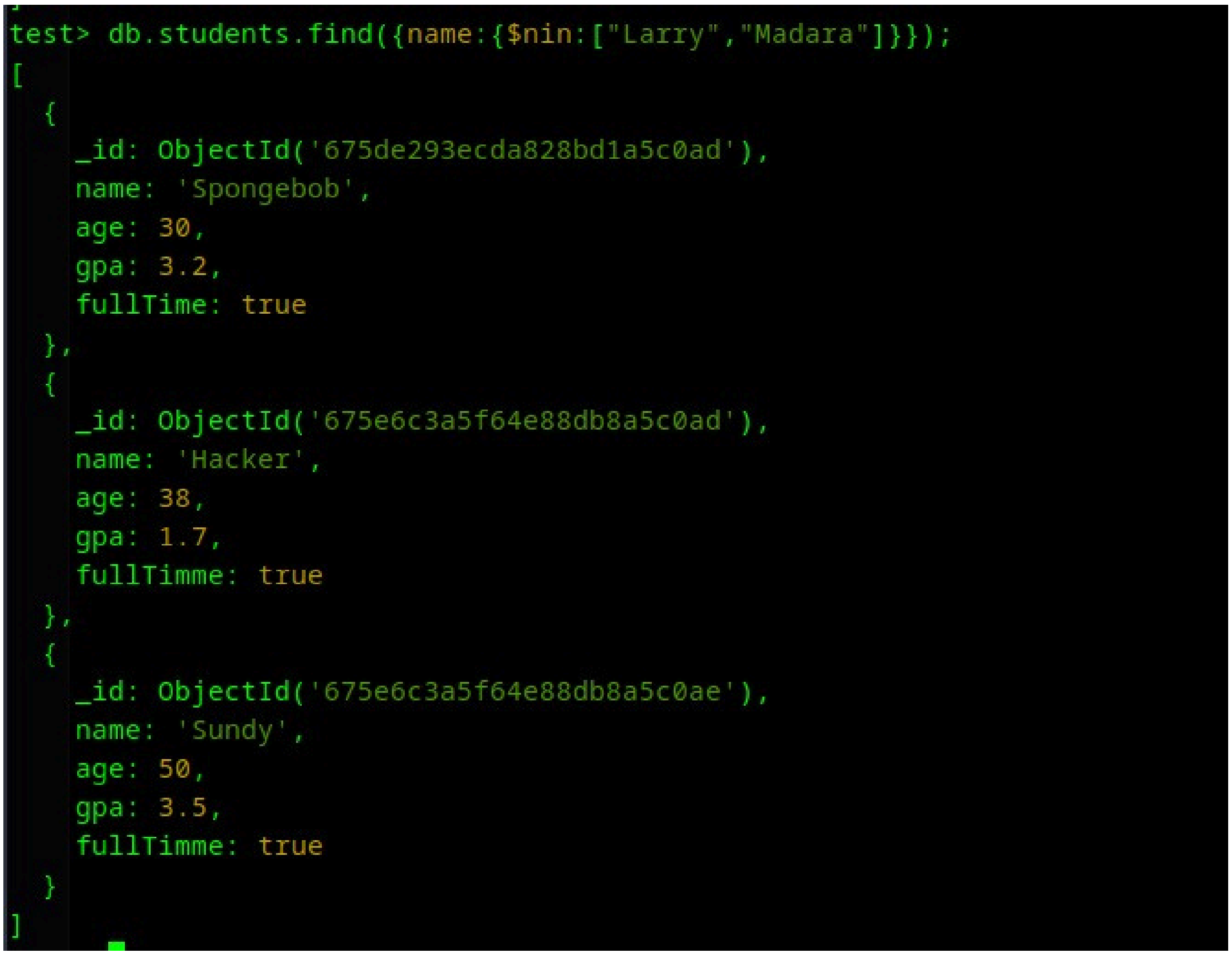
    - db.students.find({name:{$in:["Larry","Madara"]}});



```
test> db.students.find({name:{$in:["Larry","Madara"]}});
[
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0af'),
    name: 'Madara',
    age: 23,
    gpa: 8.1,
    fullTimme: true
  },
  {
    _id: ObjectId('675e70bf5f64e88db8a5c0b0'),
    name: 'Larry',
    age: 21,
    gpa: 2.8,
    fullTime: false,
    registerDate: ISODate('2024-12-15T06:01:35.629Z'),
    graduationDate: null,
    courses: [ 'Biology', 'Chemistry', 'Calculus' ],
    address: { street: '123 Fake st', city: 'Bikini Bottom', zip: 12345 }
  }
]
```

Now, not in:

- db.collectionName.find({variableName:{$nin:["conditionsValue"]}});



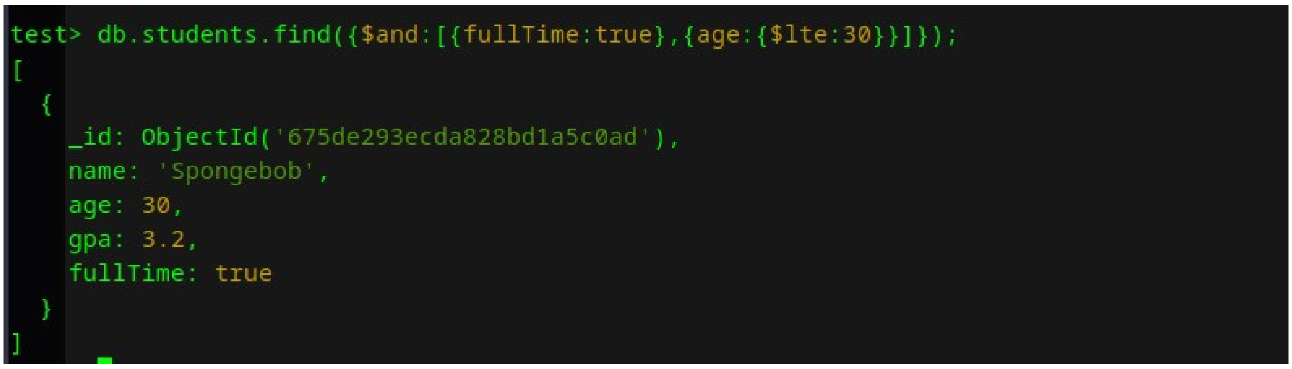
```
test> db.students.find({name:{$nin:["Larry","Madara"]}});
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2,
    fullTime: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ad'),
    name: 'Hacker',
    age: 38,
    gpa: 1.7,
    fullTimme: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5,
    fullTimme: true
  }
]
```

Logical Operators:

Logical operators returns data based on expressions that evaluate to true or false.

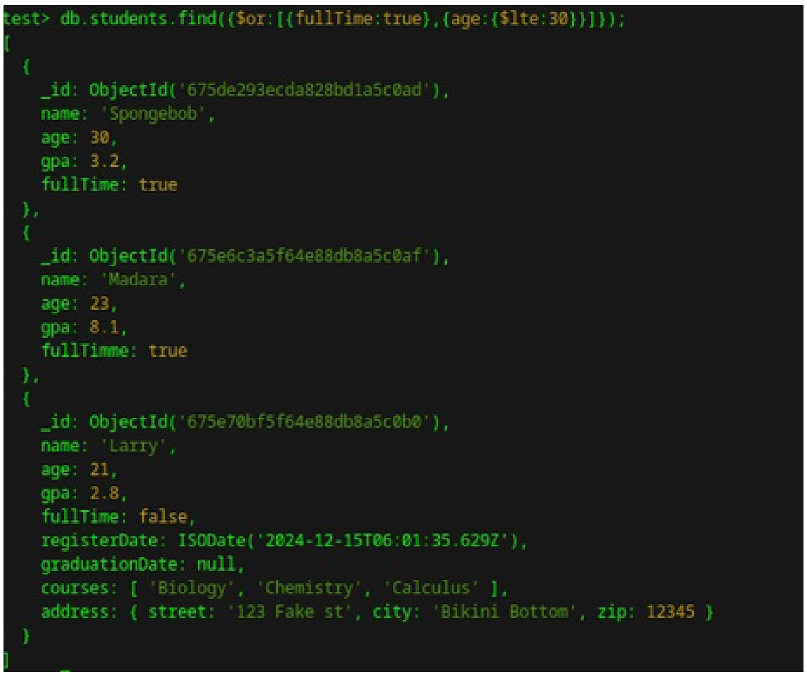
- **$and:** Joins query clauses with a logical AND returns all documents that match the conditions of both clauses
- **$nor:** Inverts the effects of a query expressions and returns documents that don't match the query expressions
- **$not**: Joins query with a logical NOR returns all documents that fail to match both clauses
- **$or:** Joins query clauses with a logical OR returns all documents that match the conditions of either clauses.

    - db.collectionName.find({$and: [{firstCondition}, {secondCondition}]});
        - db.students.find({$and:[{fullTime:true},{age:{$lte:30}}]});



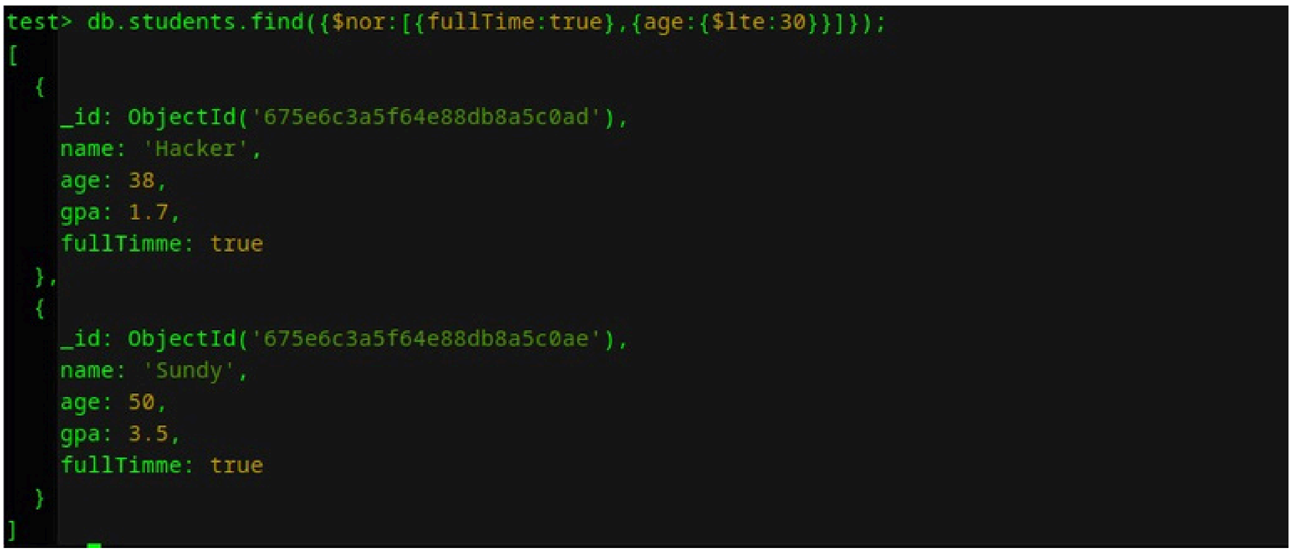
```
test> db.students.find({$and:[{fullTime:true},{age:{$lte:30}}]});
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2,
    fullTime: true
  }
]
```

    - db.collectionName.find({$or: [{firstCondition}, {secondCondition}]});
        - db.students.find({$or:[{fullTime:true},{age:{$lte:30}}]});



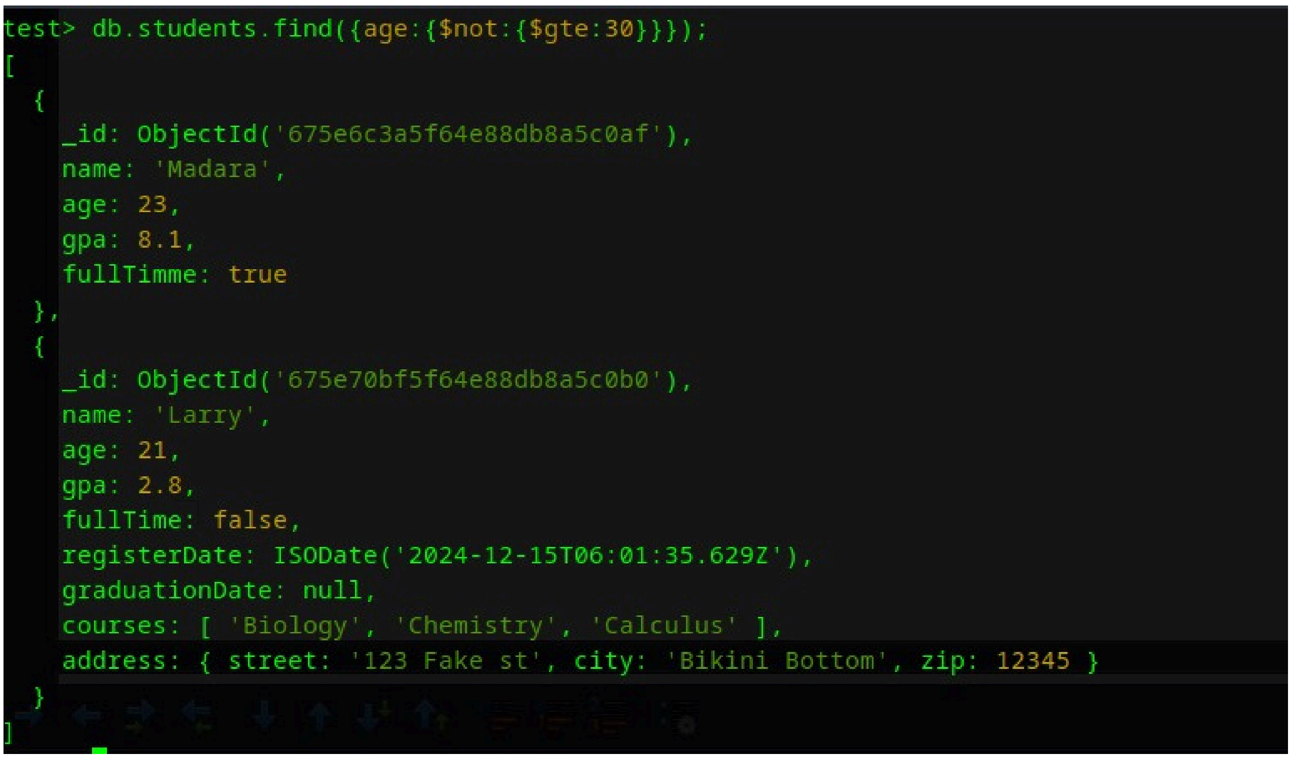
```
test> db.students.find({$or:[{fullTime:true},{age:{$lte:30}}]});
[
  {
    _id: ObjectId('675de293ecda828bd1a5c0ad'),
    name: 'Spongebob',
    age: 30,
    gpa: 3.2,
    fullTime: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0af'),
    name: 'Madara',
    age: 23,
    gpa: 8.1,
    fullTimme: true
  },
  {
    _id: ObjectId('675e70bf5f64e88db8a5c0b0'),
    name: 'Larry',
    age: 21,
    gpa: 2.8,
    fullTime: false,
    registerDate: ISODate('2024-12-15T06:01:35.629Z'),
    graduationDate: null,
    courses: [ 'Biology', 'Chemistry', 'Calculus' ],
    address: { street: '123 Fake st', city: 'Bikini Bottom', zip: 12345 }
  }
]
```

- db.collectionName.find({$nor: [{firstCondition}, {secondCondition}]});
  - db.students.find({$nor:[{fullTime:true},{age:{$lte:30}}]});



```
test> db.students.find({$nor:[{fullTime:true},{age:{$lte:30}}]});
[
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ad'),
    name: 'Hacker',
    age: 38,
    gpa: 1.7,
    fullTimme: true
  },
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0ae'),
    name: 'Sundy',
    age: 50,
    gpa: 3.5,
    fullTimme: true
  }
]
```

- db.collectionName.find({firstVariable: {$not{Condition}}});
  - db.students.find({age:{$not:{$gte:30}}});

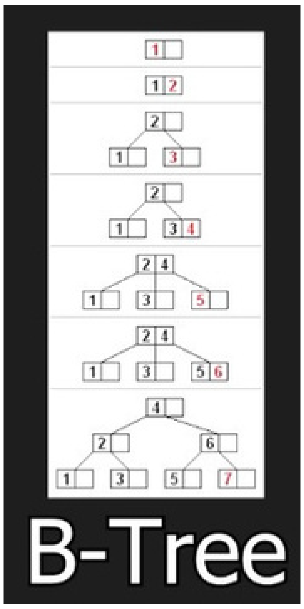


```
test> db.students.find({age:{$not:{$gte:30}}});
[
  {
    _id: ObjectId('675e6c3a5f64e88db8a5c0af'),
    name: 'Madara',
    age: 23,
    gpa: 8.1,
    fullTimme: true
  },
  {
    _id: ObjectId('675e70bf5f64e88db8a5c0b0'),
    name: 'Larry',
    age: 21,
    gpa: 2.8,
    fullTime: false,
    registerDate: ISODate('2024-12-15T06:01:35.629Z'),
    graduationDate: null,
    courses: [ 'Biology', 'Chemistry', 'Calculus' ],
    address: { street: '123 Fake st', city: 'Bikini Bottom', zip: 12345 }
  }
]
```

Indexes:

Indexes supports the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a collection scan.

- i.e.: Scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of document it must inspect..
- We are storing the data in the format of the Binary Tree OR B – Tree.



- Here, we are using the explain() method:

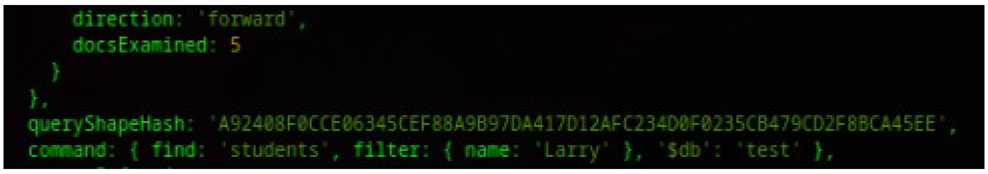
  - db.collectionName.find({variableName: "data"}).explain("explainStats");
    #.explain("explainStats") – is method to know that how we fetched the data from the DB.

    - E.g.: db.students.find({name: "Larry"}).explain("explainStats");

The output is on the next page and you can see that it fetches all the query and then execute our output that we provided in the input.

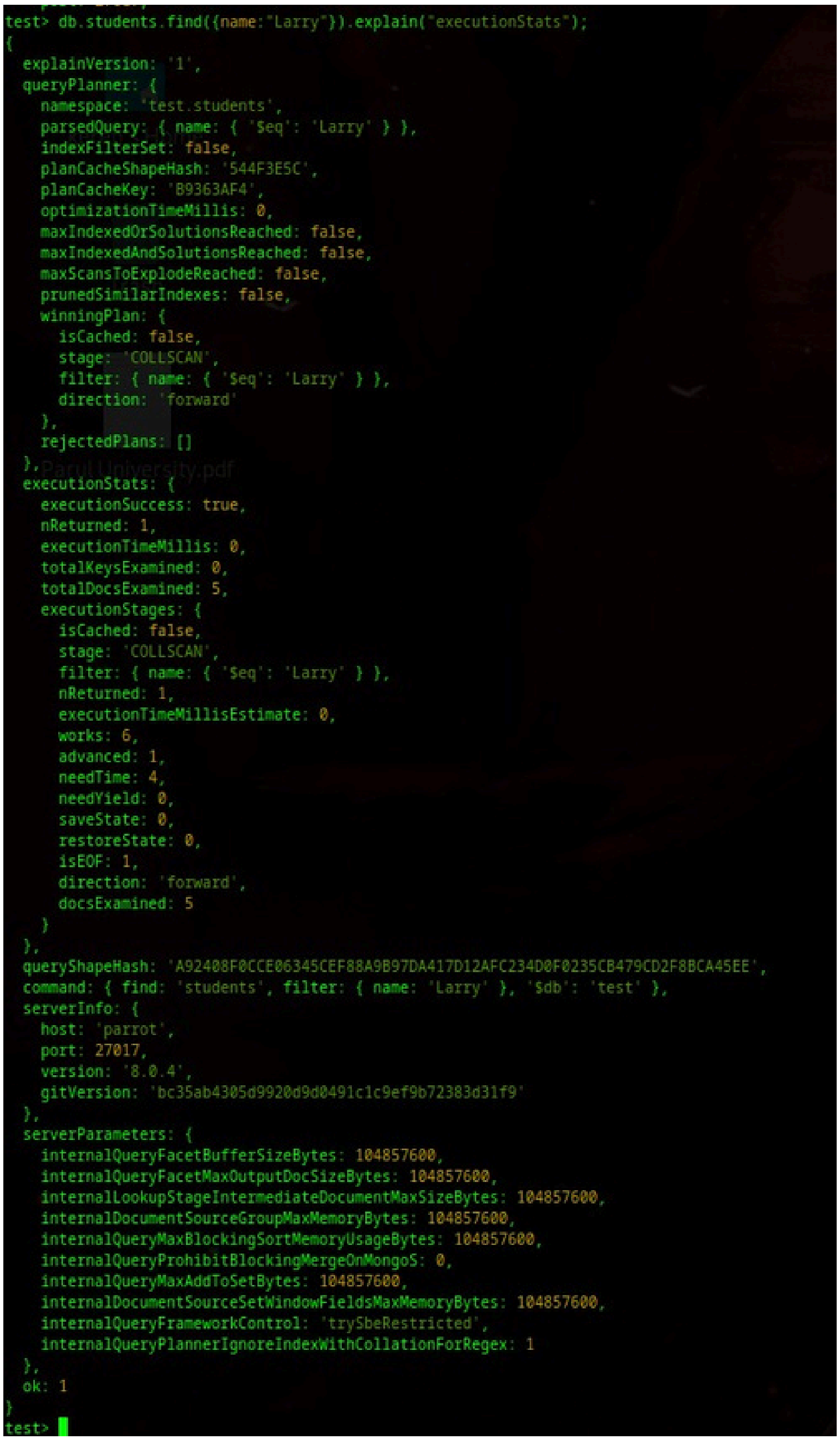
- It uses the different types of searching algorithm:
  - Indexing Strategies
  - Search Features
  - Key Search Characteristics
  - Search Techniques:
    - Text operator for searching
    - Compound search capabilities
    - Autocomplete functionality
    - Language-specific search options.

- Here, you can see that docsExamined = 5, and it is very time consuming.



```
    direction: 'forward',
    docsExamined: 5
  }
},
queryShapeHash: 'A92408F0CCE06345CEF88A9B97DA417D12AFC234D0F0235CB479CD2F8BCA45EE',
command: { find: 'students', filter: { name: 'Larry' }, '$db': 'test' },
```

- It is just a subpart of the below picture

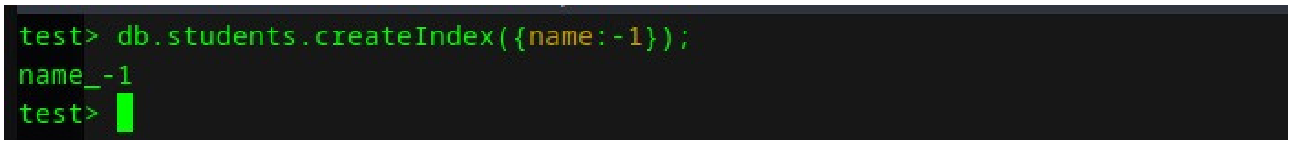


```
test> db.students.find({name:"Larry"}).explain("executionStats");
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'test.students',
    parsedQuery: { name: { '$eq': 'Larry' } },
    indexFilterSet: false,
    planCacheShapeHash: '544F3E5C',
    planCacheKey: 'B9363AF4',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { name: { '$eq': 'Larry' } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 5,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { name: { '$eq': 'Larry' } },
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      works: 6,
      advanced: 1,
      needTime: 4,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 5
    }
  },
  queryShapeHash: 'A92408F0CCE06345CEF88A9B97DA417D12AFC234D0F0235CB479CD2F8BCA45EE',
  command: { find: 'students', filter: { name: 'Larry' }, '$db': 'test' },
  serverInfo: {
    host: 'parrot',
    port: 27017,
    version: '8.0.4',
    gitVersion: 'bc35ab4305d9920d9d0491c1c9ef9b72383d31f9'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}
test>
```

- To reduce the speed we are using the Indexing method:

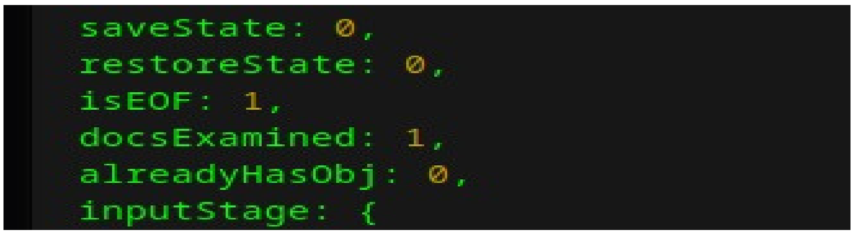
- db.collectionName.createIndex({variableName:1/-1}); *#1 is for Ascending Order and -1 is for Descending Order*

    - db.students.createIndex({name:-1});



```
test> db.students.createIndex({name:-1});
name_-1
test> 
```

- Here, it will create a index named as ***name_-1.*** Now, we will access it -



```
    saveState: 0,
    restoreState: 0,
    isEOF: 1,
    docsExamined: 1,
    alreadyHasObj: 0,
    inputStage: {
```

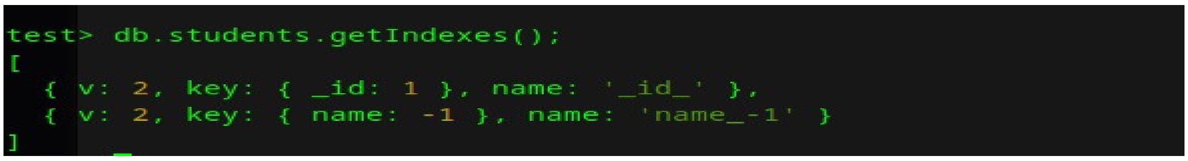
- Here, you can see that due to the indexing, the docsExamined stats is just only – 1.

- So, this is how you can save lots of time.

To fetch the indexes we will use the given below command:

- db.**collectionName**.getIndexex():

    - db.students.getIndexex();

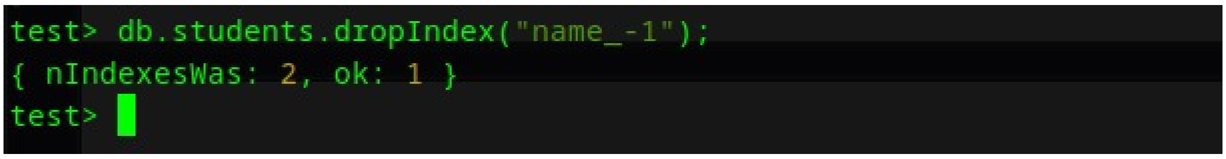


```
test> db.students.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: -1 }, name: 'name_-1' }
]
```

Now, drop the index:

- db.collectionName.dropIndex("index_name");

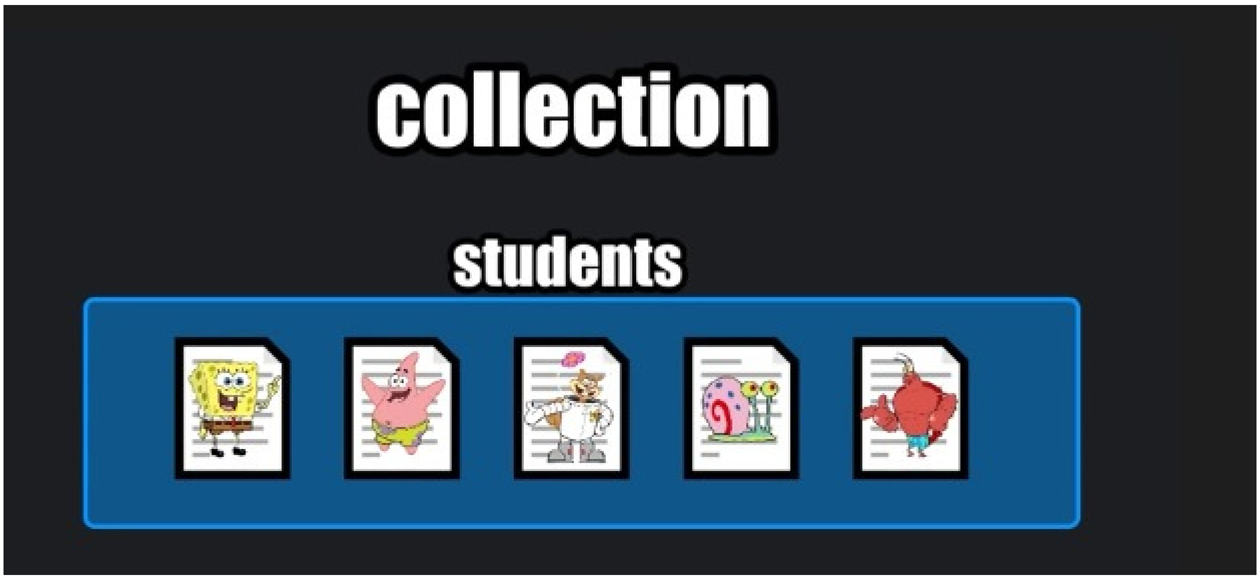
    - db.students.dropIndex("name_-1");



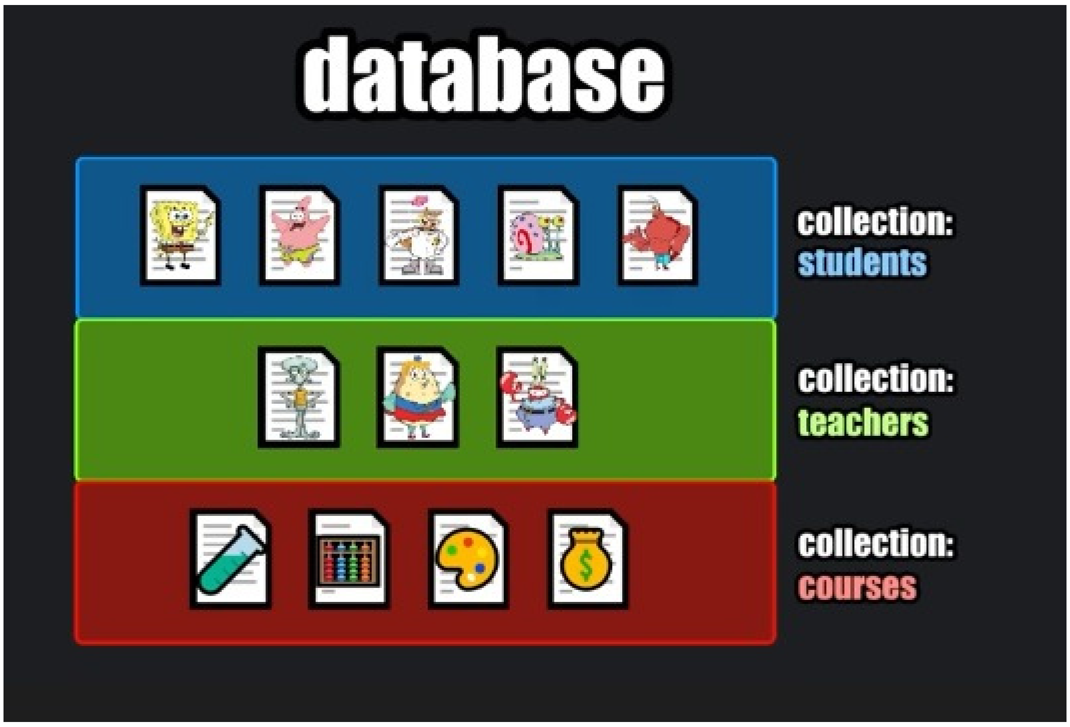
```
test> db.students.dropIndex("name_-1");
{ nIndexesWas: 2, ok: 1 }
test> 
```

Collections:

Collection is a group of documents



- And Database is a group of collection.



Fetch all the collections:

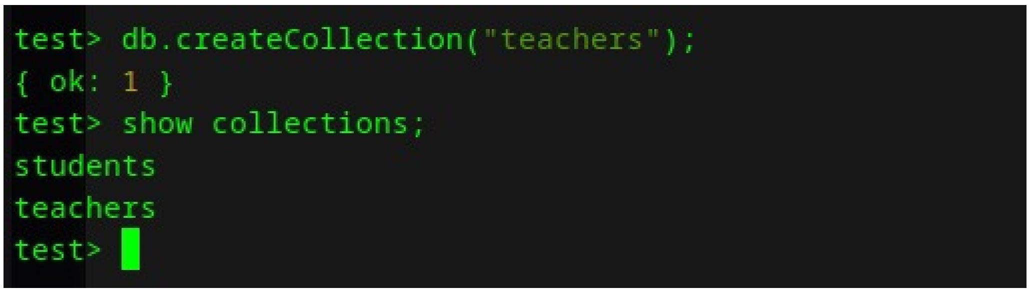
- show collections/tables;

- It will show the no. of collections inside the database.

Now,

Create a collection

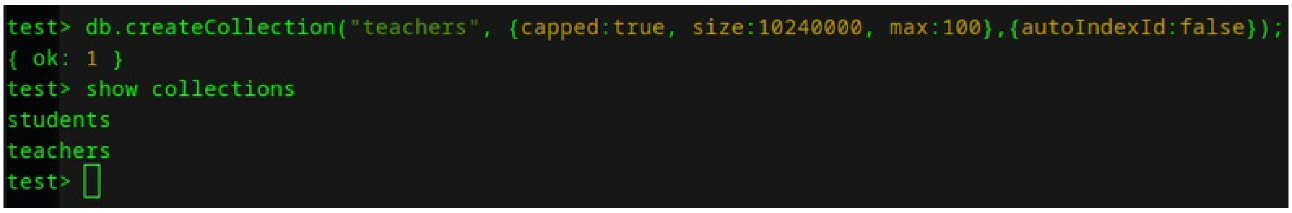
- db.createCollection("Collection_Name");
  - db.createCollection("teachers");



```
test> db.createCollection("teachers");
{ ok: 1 }
test> show collections;
students
teachers
test> 
```

Now, there are some specific ways to make the collection as better with some argumments:

- db.createCollection("Collection_name",{capped:true, size:1024000, max:100}, {autoIndexId:false}); *#You can change the valuse of the arguments like capped:false, size will N bytes, max will N no. of people, autoIndexId will be true and you can add more than that arguments*
  - db.createCollection("teachers",{capped:true, size:1024000, max:100}, {autoIndexId:false});



```
test> db.createCollection("teachers", {capped:true, size:10240000, max:100},{autoIndexId:false});
{ ok: 1 }
test> show collections
students
teachers
test> 
```