**eg-3:**

---

```
var mycursor = db.learners.find();
while( mycursor.hasNext() )
{
   printjson(mycursor.next());
}
```

**on shell:**

```
var mycursor = db.learners.find();
while( mycursor.hasNext() ) {  printjson(mycursor.next()); }
```

**eg-4:**

---

```
var mycursor = db.learners.find();
mycursor.forEach( doc => { printjson(doc) } )
```

**eg-5:**

---

```
var mycursor = db.learners.find();
mycursor.forEach(printjson)
```

## Cursor Helper Methods:

---

**We can use the following helper methods to shape our results:**

1. limit()
2. skip()
3. sort()

**1. limit():**

---

**We can use this limit() method to limit the number of documents in the result.**

```
> db.learners.find().count()
6571

> db.learners.find().limit(1)
{ "_id" : 2, "name" : "abhilash", "marks" : 20 }

> db.learners.find().limit(2)
{ "_id" : 2, "name" : "abhilash", "marks" : 20 }
{ "_id" : 3, "name" : "rasika", "marks" : 30 }

> db.learners.find().limit(5)
{ "_id" : 2, "name" : "abhilash", "marks" : 20 }
{ "_id" : 3, "name" : "rasika", "marks" : 30 }
{ "_id" : 1, "name" : "narayan pradhan", "marks" : 10 }
{ "_id" : 6, "name" : "dhanaraju", "marks" : 60 }
{ "_id" : 7, "name" : "Satyasundar Panigrahi", "marks" : 70 }


> db.learners.find().limit(25)
{ "_id" : 2, "name" : "abhilash", "marks" : 20 }
{ "_id" : 3, "name" : "rasika", "marks" : 30 }
{ "_id" : 1, "name" : "narayan pradhan", "marks" : 10 }
{ "_id" : 6, "name" : "dhanaraju", "marks" : 60 }
{ "_id" : 7, "name" : "Satyasundar Panigrahi", "marks" : 70 }
{ "_id" : 5, "name" : "Sheshanand Singh", "marks" : 50 }
{ "_id" : 8, "name" : "jyothi", "marks" : 80 }
{ "_id" : 10, "name" : "bindhiya", "marks" : 100 }
{ "_id" : 9, "name" : "Hari", "marks" : 90 }
{ "_id" : 4, "name" : "pankaj bhandari", "marks" : 40 }
```

{ "_id" : 11, "name" : "vikas kale", "marks" : 10 }
{ "_id" : 13, "name" : "shashank sanap", "marks" : 30 }
{ "_id" : 12, "name" : "Sunita Kumati Choudhuri", "marks" : 20 }
{ "_id" : 15, "name" : "TharunK", "marks" : 50 }
{ "_id" : 14, "name" : "Atul", "marks" : 40 }
{ "_id" : 18, "name" : "Dusmant Kumar Mohapatra", "marks" : 80 }
{ "_id" : 16, "name" : "aron", "marks" : 60 }
{ "_id" : 24, "name" : "G.shukeshreddy", "marks" : 40 }
{ "_id" : 25, "name" : "Dakshesh", "marks" : 50 }
{ "_id" : 26, "name" : "Paramesh", "marks" : 60 }
Type "it" for more
> it
{ "_id" : 27, "name" : "Mahmodul Hasan", "marks" : 70 }
{ "_id" : 28, "name" : "ASHA", "marks" : 80 }
{ "_id" : 17, "name" : "pooja", "marks" : 70 }
{ "_id" : 30, "name" : "Maheshbabu", "marks" : 100 }
{ "_id" : 21, "name" : "Suraj Prasim Patel", "marks" : 10 }

## 2. skip():

---

We can use skip() method to skip the number of documents in the result.

> db.learners.find().skip(10)
To skip the first 10 documents.

Q. To skip first 10 documents and to display next 10 documents?
> db.learners.find().skip(10).limit(10)

## Use Case:

---

In general we can use skip() and limit() methods in pagination concept while displaying our data.

**Assume per page 10 documents:**

**To display 1st page: db.learners.find().limit(10)**

**To display 2nd page: db.learners.find().skip(10).limit(10)**

**To display 3rd page: db.learners.find().skip(20).limit(10)**

**etc**

## 3. sort():

---

**We can use sort() method to sort documents based on value of a particular field.**

**Syntax:**

**sort({ field: 1})**

**1 ===> means Ascending order/Alphabetical order**

**-1 ===> means Descending order/ Reverse of Alphabetical order**

**Q1. To display all learners based on ascending order of marks?**

**> db.learners.find().sort({ marks: 1}).pretty()**

**Q2. To display all learners based on descending order of marks?**

**> db.learners.find().sort({ marks: -1}).pretty()**

**Q3. To display all learners based on alphabetical order of names?**

**> db.learners.find().sort({ name: 1}).pretty()**

**Sorting based on multiple fields:**

---

**We can sort based on multiple fields also.**

**Syntax: sort({field1: 1, field2: 1,...})**

**Sorting is based on field1, if field1 values are same then sorting based on field2 for those documents.**

**Q. Sort based on ascending order or marks. If two learners have same marks then sort based on reverse of alphabetical order of names?**

**> db.learners.find().sort({ marks: 1, name: -1}).pretty()**

**Note:**
**-----**
**Chaining of these helper methods is possible.**

**> db.learners.find().sort({ name: -1}).skip(100).limit(15)**

**All these methods will be executed from left to right and hence order is important.**

**Pagination based on alphabetical order of names:**
**----------**
**We have to sort based on alphabetical order of names. If two students having same name then consider ascending order of marks.**
**Per page only 15 documents.**

**1st page: db.learners.find().sort({name: 1, marks: 1}).limit(15).pretty()**
**2nd page: db.learners.find().sort({name: 1, marks: 1}).skip(15).limit(15).pretty()**
**3rd page: db.learners.find().sort({name: 1, marks: 1}).skip(30).limit(15).pretty()**
**etc**

**How to get documents with only required fields:**
**----------**

# PROJECTION:

**We can get documents with only required fields instead of all fields. This is called projection.**

**Relational databases/sql dabases:**

**---**

**without projection: select * from employees;**

**with projection: select ename,esal from employees;**

**Projection in MongoDB?**

**---**

**db.collection.find({filter}) ===>without projection**

**db.collection.find({filter},{projection fields}) ===>with projection**

**Note: If we are providing projection list, compulsory we should provide filter object also, atleast empty java script object. i.e without providing first argument, we cannot talk about second argument.**

**eg: db.collection.find({},{projection fields})**

**Case Study:**

**---**

**books collection: sample document**

```
{
    "title": "Linux in simple way",
    "isbn":  6677,
    "downloadable": false,
    "no_of_reviews": 1,
    "tags": ["os","freeware","shell programming"],
```