How to import data from csv file to MongoDB?

```
------------------------------------------
vidya.csv:
----------
empno	name	salary
1	Vidya	5000
2	Ravi	2000
3	Durga	6000
4	Sushma	3000
```

Command: mongoimport -d storedb -c emp --type csv --headerline --drop vidya.csv

```
> show collections
books
courses
emp
homeBudget
learners
phonebook
shop
students
> db.emp.find().pretty()
{
	"_id" : ObjectId("5ff7cfe9ba62f324ceb5df09"),
	"empno" : 1,
	"name" : "Vidya",
	"salary" : 5000
}
{
	"_id" : ObjectId("5ff7cfe9ba62f324ceb5df0a"),
	"empno" : 2,
	"name" : "Ravi",
	"salary" : 2000
}
{
        "_id" : ObjectId("5ff7cfe9ba62f324ceb5df0b"),
        "empno" : 4,
        "name" : "Sushma",
        "salary" : 3000
}
{
        "_id" : ObjectId("5ff7cfe9ba62f324ceb5df0c"),
        "empno" : 3,
        "name" : "Durga",
        "salary" : 6000
}
```

eg-2:

-----

learners.csv:

```
_id     name        marks
1       narayan pradhan         10
2       abhilash    20
3       rasika      30
4       pankaj bhandari 40
5       Sheshanand Singh        50
6       dhanaraju  60
7       Satyasundar Panigrahi           70
8       jyothi      80
```

Command: mongoimport -d storedb -c learners --type csv --headerline --drop learners.csv

Q. How to insert documents from csv file

Command 1:

```
mongoimport --db csvdb --collection sample_csv --file
"Sample_csv_file.csv" --type csv --headerline
```

# --headerline is used to ensure that our csv headerline should be read as keys while importing into database

Command 2:

```
mongoimport --db csvdb --collection sample_csv --file
"Sample_csv_file.csv" --type csv --headerline --maintainInsertionOrder
```

# --maintainInsertionOrder ensures that the data will be inserted in the top to bottom order from the csv file (row by row)

Command 3:

```
mongoimport --db csvdb --collection sample_csv --file
"Sample_csv_file.csv" --type csv --headerline --maintainInsertionOrder --
ignoreBlanks
```

# --ignoreBlanks ---> Whereever the value for key is blank inside csv file, that particular key itself will not be imported to database for that particular document

The Complete Story of Cursor concept:

Diagram

In a collection there may be a chance of lakhs of documents. Whenever we are trying to retrieve data from database, if MongoDB server sends total data, there may be a chance of the following problems:

1. Storage problems
2. Network traffic problem
3. Performance problems
etc

To prevent these problems, most of the databases including MongoDB, uses cursor concept.

In MongoDB, if we are using find() method we won't get documents and we will get cursor object.

The return type of find() method is cursor object.
By using cursor object we can get data either batch wise or document wise.
Bydefault cursor object will provide documents in batch wise. The default batch size is 20. But we can customize this value. For this we have to use DBQuery.shellBatchSize property.

```
> db.learners.find().pretty()
{ "_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 }
{ "_id" : 32, "name" : "kusuma", "marks" : 20 }
{ "_id" : 33, "name" : "Vaishnavi Lende", "marks" : 30 }
{ "_id" : 19, "name" : "Deepak", "marks" : 90 }
{ "_id" : 23, "name" : "Zaid", "marks" : 30 }
{ "_id" : 20, "name" : "Bhim Kumar", "marks" : 100 }
{ "_id" : 22, "name" : "Naveen", "marks" : 20 }
{ "_id" : 35, "name" : "Prakash", "marks" : 50 }
{ "_id" : 31, "name" : "rajat kumar maurya", "marks" : 10 }
{ "_id" : 34, "name" : "Vengadesan", "marks" : 40 }
{ "_id" : 29, "name" : "Bharti Kardile", "marks" : 90 }
{ "_id" : 37, "name" : "sanat", "marks" : 70 }
{ "_id" : 38, "name" : "Aneesh Fathima", "marks" : 80 }
{ "_id" : 36, "name" : "raj", "marks" : 60 }
{ "_id" : 39, "name" : "Ratemo", "marks" : 90 }
{ "_id" : 41, "name" : "jignesh", "marks" : 10 }
Type "it" for more

> DBQuery.shellBatchSize = 5;
5
> db.learners.find().pretty()
```

```
{ "_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 }
Type "it" for more
> it
{ "_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 }
Type "it" for more
```

Advantages of cursor:

----

1. We can get only required number of documents.
2. We can get either batch wise or document wise.
3. No chance of storage problems.
4. No chance of network traffic problems.
5. No chance of performance issues.

Important Methods of cursor:

----

1. count()
To get total number of documents

2. hasNext()
To check whether the next document is available or not.If it avaialble then it returns true, otherwise returns false.

3. next()

To get next document. If there is no next document then we will get error.

Q. Why we are getting same document in the following case?

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

Ans: Here two different cursor objects we are using.

```
var mycursor = db.learners.find()
mycursor.next()
mycursor.next()
```

In this case we are using same cursor object and hence different documents we will get.

```
> var mycursor = db.learners.find()
> mycursor.next()
{ "_id" : 2, "name" : "abhilash", "marks" : 20 }
> mycursor.next()
{ "_id" : 3, "name" : "rasika", "marks" : 30 }
> mycursor.next()
{ "_id" : 1, "name" : "narayan pradhan", "marks" : 10 }
```

Javascript based code to get documents one by one:

---

eg-1:

```
> var mycursor = db.emp.find()
> mycursor.hasNext()
true
> mycursor.next()
{
	"_id" : ObjectId("5ff7cfe9ba62f324ceb5df09"),
	"empno" : 1,
	"name" : "Vidya",
	"salary" : 5000
}
> mycursor.next()
{
	"_id" : ObjectId("5ff7cfe9ba62f324ceb5df0a"),
	"empno" : 2,
	"name" : "Ravi",
	"salary" : 2000
}
```

eg-2:

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

on shell:

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

Note: mycursor.next() returns BSON object. We have to convert BSON Object to json by using tojson() method.

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():