MONGO DB

ChandraSekhar(CS) Baratm

1 Environment Setup

2 Database

3 Collection

4 CRUD

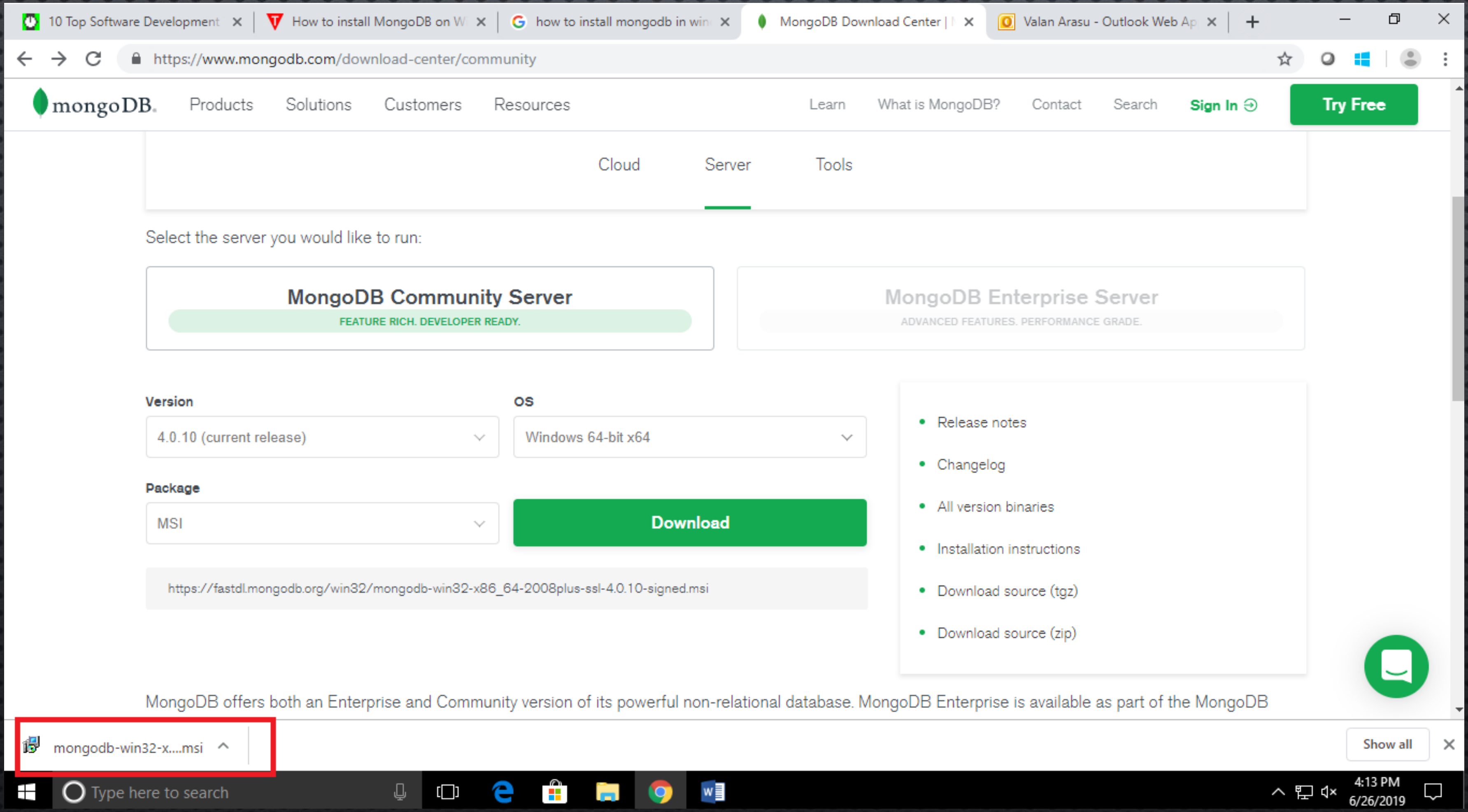
5 Limiting Records

6 Sorting Records

ChandraSekhar(CS) Baratm

ENVIRONMENT SETUP

ChandraSekhar(CS) Baratm


10 Top Software Development
How to install MongoDB on W
how to install mongodb in win
MongoDB Download Center |
Valan Arasu - Outlook Web Ap
https://www.mongodb.com/download-center/community
mongoDB.
Products
Solutions
Customers
Resources
Learn
What is MongoDB?
Contact
Search
Sign In
Try Free
Cloud
Server
Tools

Select the server you would like to run:

MongoDB Community Server
FEATURE RICH. DEVELOPER READY.

MongoDB Enterprise Server
ADVANCED FEATURES. PERFORMANCE GRADE.

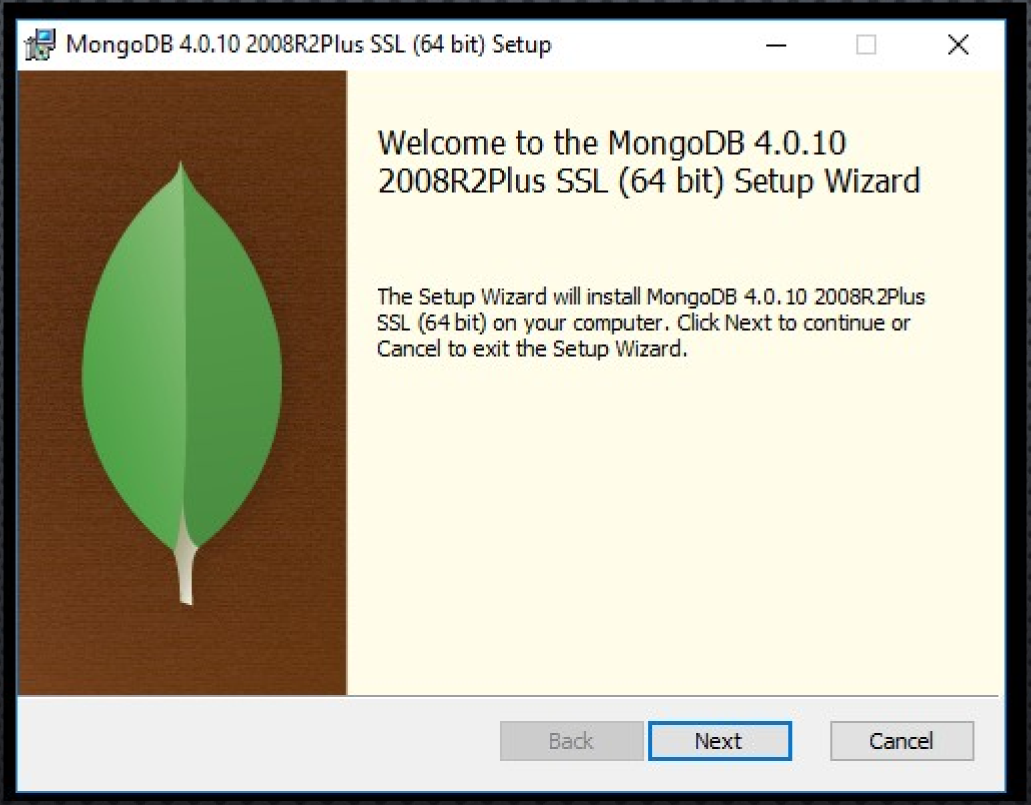
Version: 4.0.10 (current release)
OS: Windows 64-bit x64
Package: MSI

https://fastdl.mongodb.org/win32/mongodb-win32-x86_64-2008plus-ssl-4.0.10-signed.msi

Release notes
Changelog
All version binaries
Installation instructions
Download source (tgz)
Download source (zip)

MongoDB offers both an Enterprise and Community version of its powerful non-relational database. MongoDB Enterprise is available as part of the MongoDB

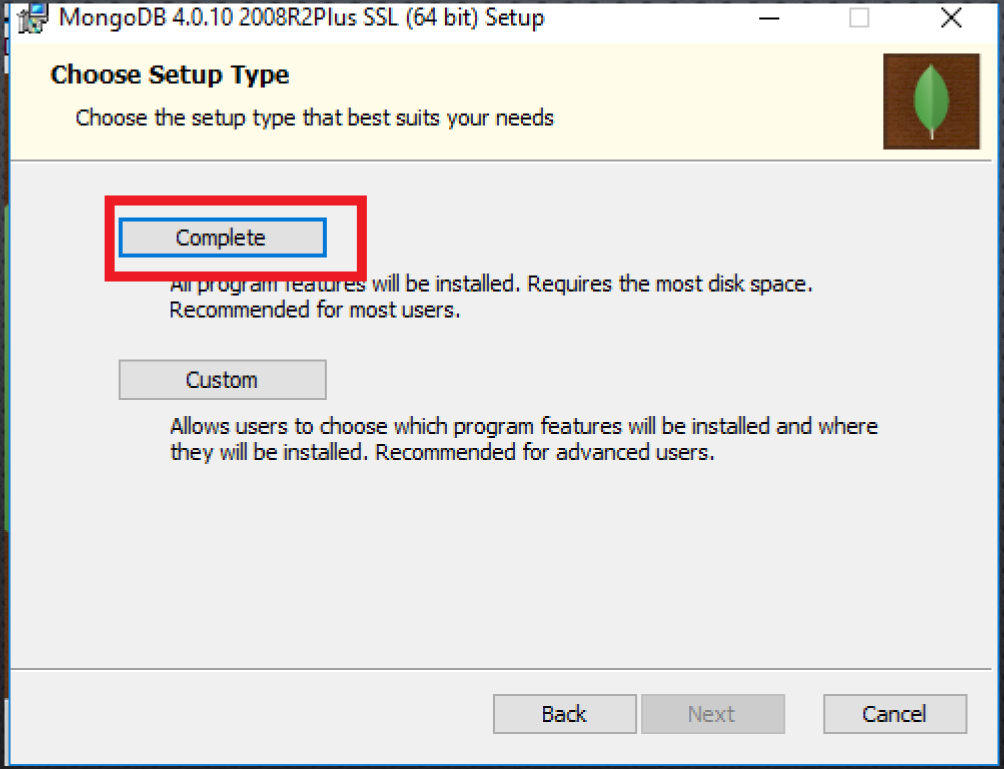
ChandraSekhar(CS) Baratm


MongoDB 4.0.10 2008R2Plus SSL (64 bit) Setup

Welcome to the MongoDB 4.0.10 2008R2Plus SSL (64 bit) Setup Wizard

The Setup Wizard will install MongoDB 4.0.10 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

ChandraSekhar(CS) Baratm


MongoDB 4.0.10 2008R2Plus SSL (64 bit) Setup

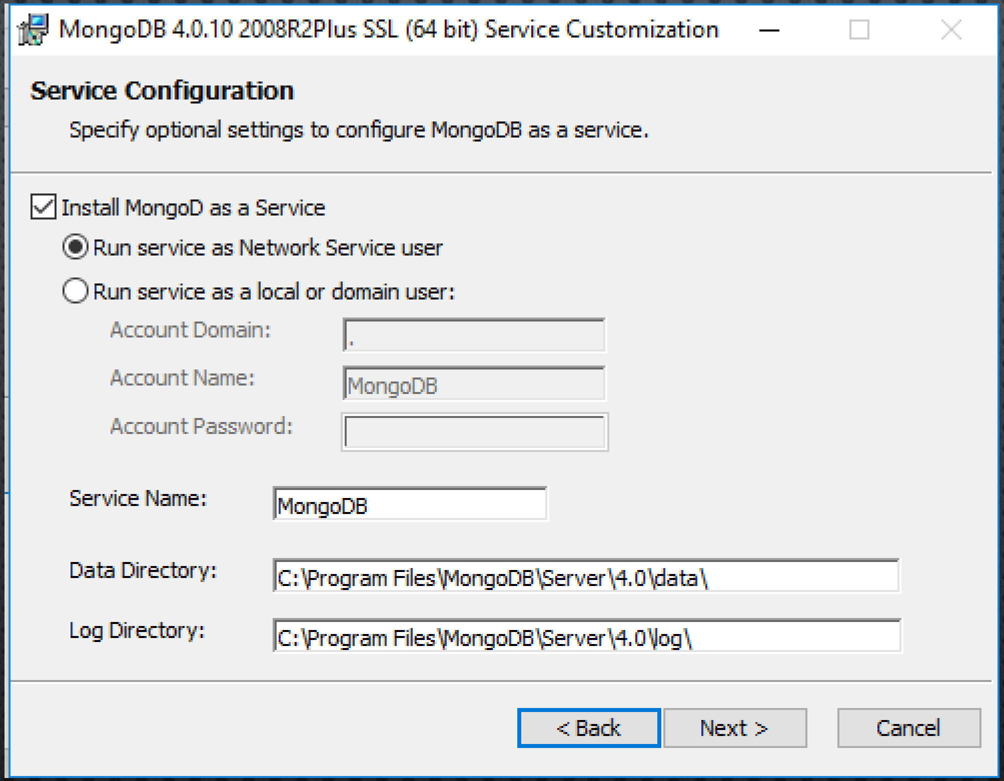
Choose Setup Type

Choose the setup type that best suits your needs

All program features will be installed. Requires the most disk space. Recommended for most users.

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

ChandraSekhar(CS) Baratm


MongoDB 4.0.10 2008R2Plus SSL (64 bit) Service Customization

Service Configuration

Specify optional settings to configure MongoDB as a service.

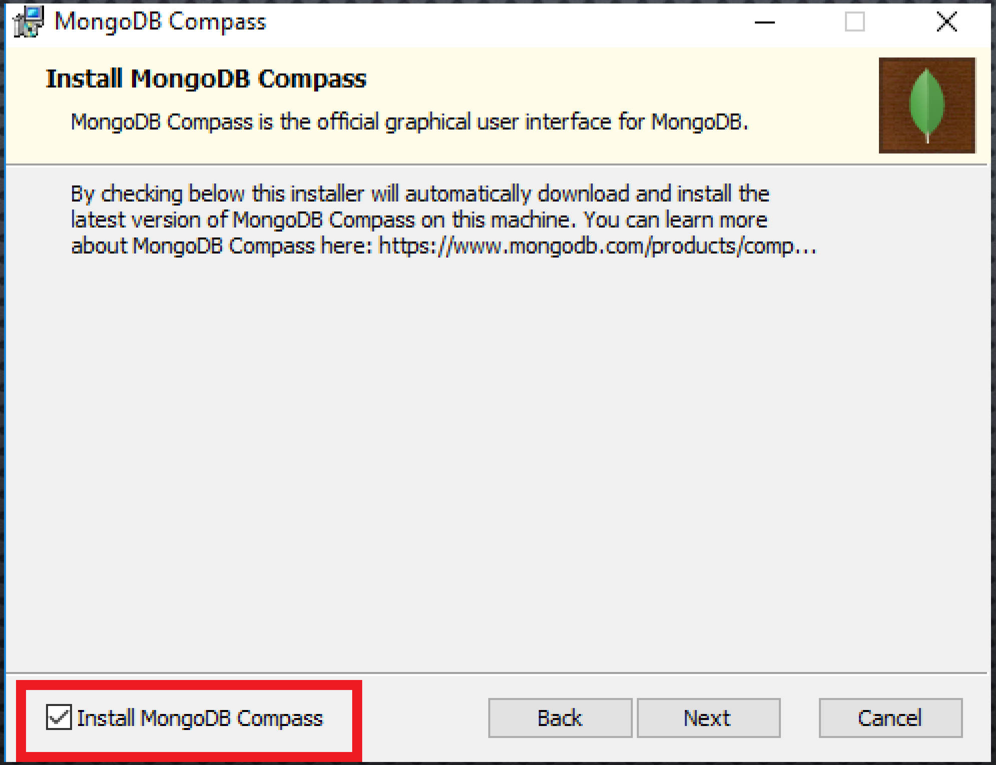
[✓] Install MongoD as a Service

● Run service as Network Service user

○ Run service as a local or domain user:

Account Domain:	.
Account Name:	MongoDB
Account Password:	
Service Name:	MongoDB
Data Directory:	C:\Program Files\MongoDB\Server\4.0\data\
Log Directory:	C:\Program Files\MongoDB\Server\4.0\log\

ChandraSekhar(CS) Baratm


MongoDB Compass

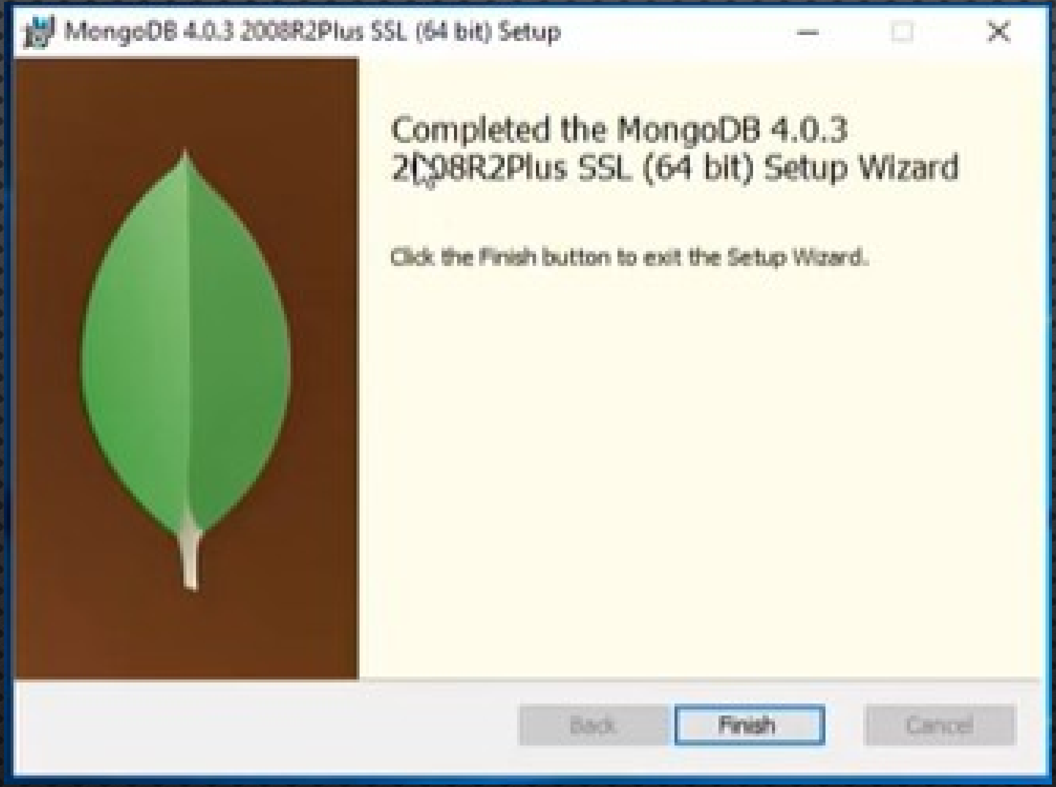
Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/comp...

[✓] Install MongoDB Compass

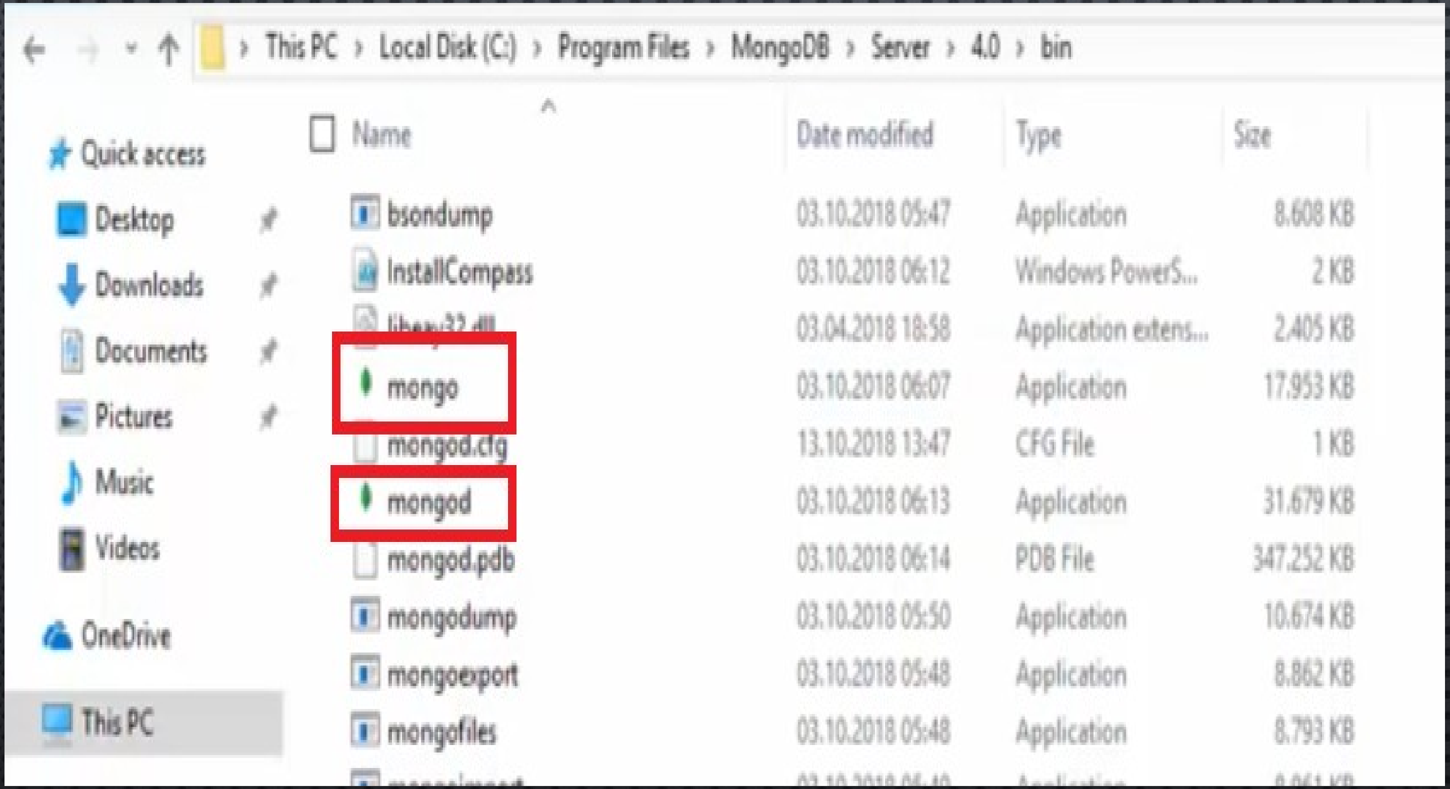
ChandraSekhar(CS) Baratm


MongoDB 4.0.3 2008R2Plus SSL (64 bit) Setup

Completed the MongoDB 4.0.3 2008R2Plus SSL (64 bit) Setup Wizard

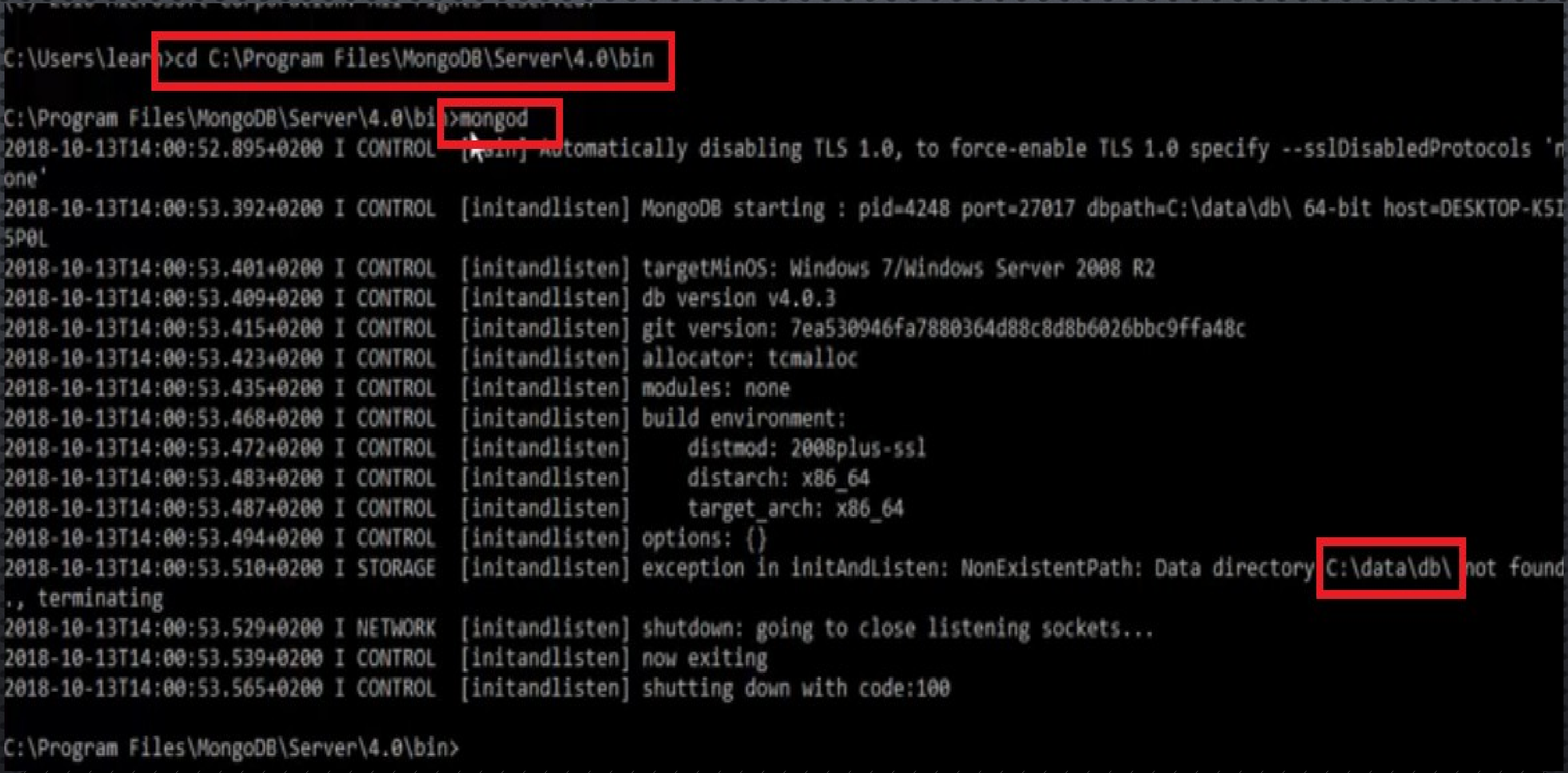
Click the Finish button to exit the Setup Wizard.

ChandraSekhar(CS) Baratm


This PC › Local Disk (C:) › Program Files › MongoDB › Server › 4.0 › bin
Quick access
Desktop
Downloads
Documents
Pictures
Music
Videos
OneDrive
This PC

Name	Date modified	Type	Size
bsondump	03.10.2018 05:47	Application	8.608 KB
InstallCompass	03.10.2018 06:12	Windows PowerS...	2 KB
libeay32.dll	03.04.2018 18:58	Application extens...	2.405 KB
mongo	03.10.2018 06:07	Application	17.953 KB
mongod.cfg	13.10.2018 13:47	CFG File	1 KB
mongod	03.10.2018 06:13	Application	31.679 KB
mongod.pdb	03.10.2018 06:14	PDB File	347.252 KB
mongodump	03.10.2018 05:50	Application	10.674 KB
mongoexport	03.10.2018 05:48	Application	8.862 KB
mongofiles	03.10.2018 05:48	Application	8.793 KB
[illegible]	[illegible]	[illegible]	[illegible]

ChandraSekhar(CS) Baratm



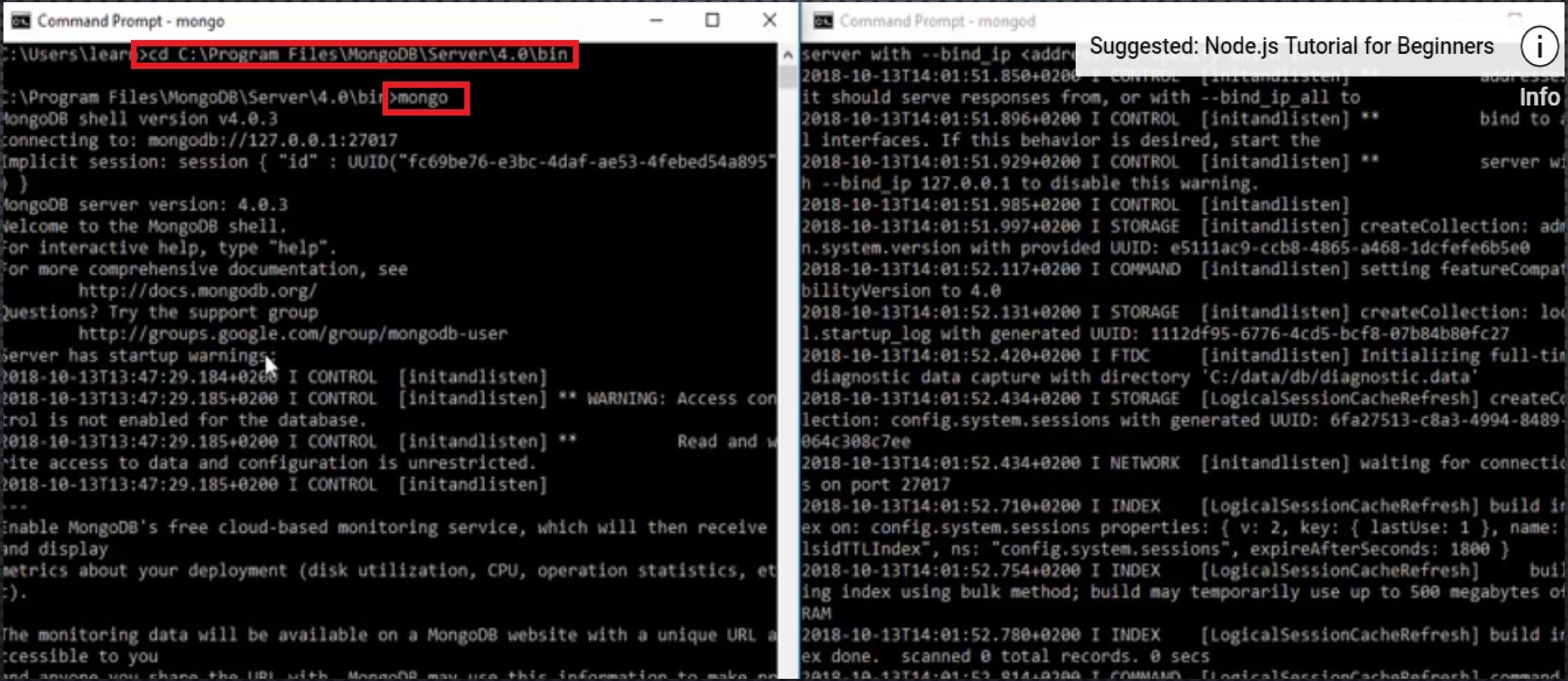
```
C:\Users\learn>cd C:\Program Files\MongoDB\Server\4.0\bin

C:\Program Files\MongoDB\Server\4.0\bin>mongod
2018-10-13T14:00:52.895+0200 I CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'n
one'
2018-10-13T14:00:53.392+0200 I CONTROL  [initandlisten] MongoDB starting : pid=4248 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-K5I
5P0L
2018-10-13T14:00:53.401+0200 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-10-13T14:00:53.409+0200 I CONTROL  [initandlisten] db version v4.0.3
2018-10-13T14:00:53.415+0200 I CONTROL  [initandlisten] git version: 7ea530946fa7880364d88c8d8b6026bbc9ffa48c
2018-10-13T14:00:53.423+0200 I CONTROL  [initandlisten] allocator: tcmalloc
2018-10-13T14:00:53.435+0200 I CONTROL  [initandlisten] modules: none
2018-10-13T14:00:53.468+0200 I CONTROL  [initandlisten] build environment:
2018-10-13T14:00:53.472+0200 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-10-13T14:00:53.483+0200 I CONTROL  [initandlisten]     distarch: x86_64
2018-10-13T14:00:53.487+0200 I CONTROL  [initandlisten]     target_arch: x86_64
2018-10-13T14:00:53.494+0200 I CONTROL  [initandlisten] options: {}
2018-10-13T14:00:53.510+0200 I STORAGE  [initandlisten] exception in initAndListen: NonExistentPath: Data directory C:\data\db\ not found
., terminating
2018-10-13T14:00:53.529+0200 I NETWORK  [initandlisten] shutdown: going to close listening sockets...
2018-10-13T14:00:53.539+0200 I CONTROL  [initandlisten] now exiting
2018-10-13T14:00:53.565+0200 I CONTROL  [initandlisten] shutting down with code:100

C:\Program Files\MongoDB\Server\4.0\bin>
```

Create this folder in C drive. And then run the "mongod" command again.

ChandraS



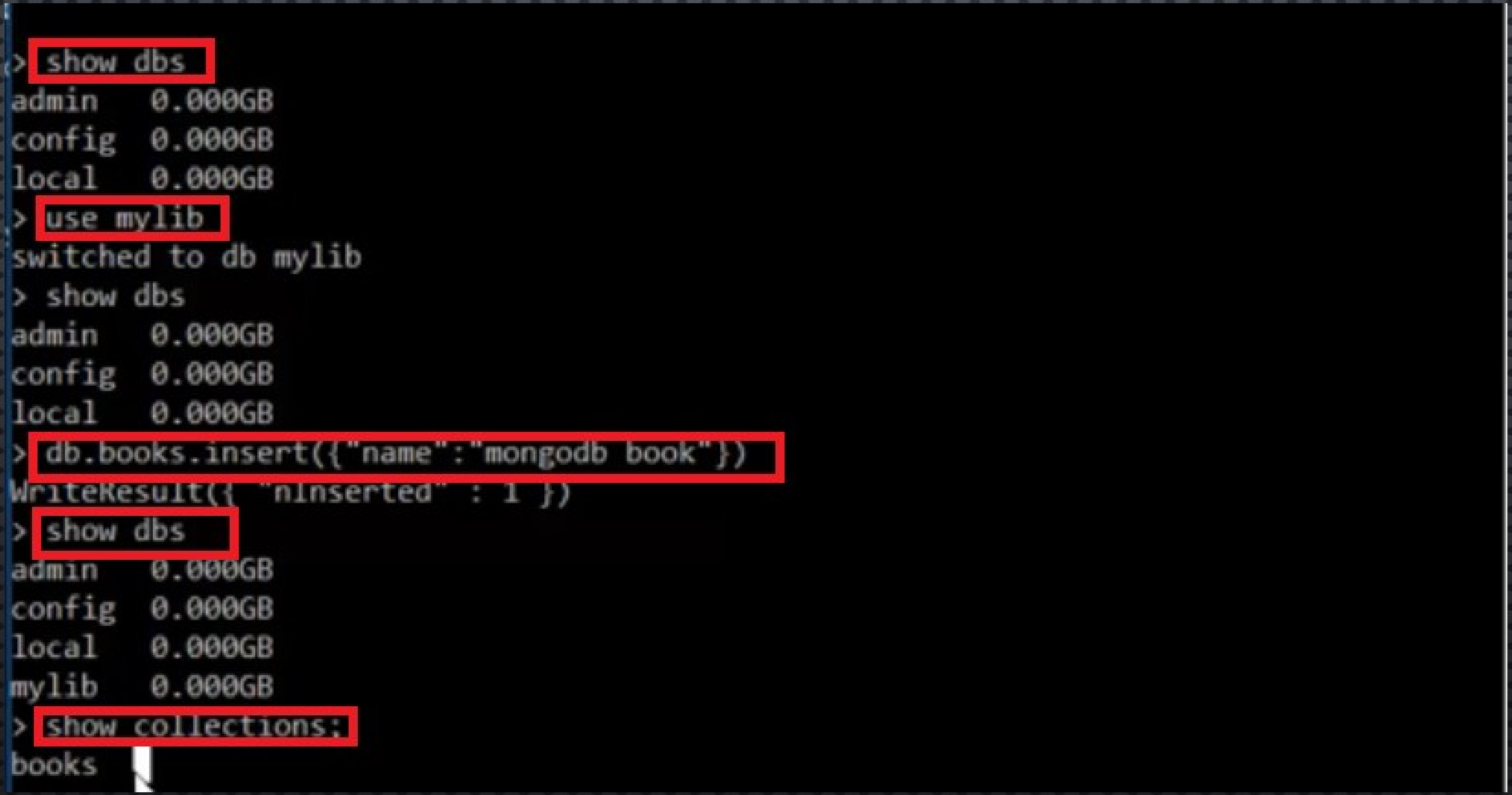
```
:\Users\lear>cd C:\Program Files\MongoDB\Server\4.0\bin

:\Program Files\MongoDB\Server\4.0\bin>mongo
longoDB shell version v4.0.3
onnecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("fc69be76-e3bc-4daf-ae53-4febed54a895"
) }
longoDB server version: 4.0.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2018-10-13T13:47:29.184+0200 I CONTROL  [initandlisten]
2018-10-13T13:47:29.185+0200 I CONTROL  [initandlisten] ** WARNING: Access con
trol is not enabled for the database.
2018-10-13T13:47:29.185+0200 I CONTROL  [initandlisten] **          Read and w
rite access to data and configuration is unrestricted.
2018-10-13T13:47:29.185+0200 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive
and display
metrics about your deployment (disk utilization, CPU, operation statistics, et
c).

The monitoring data will be available on a MongoDB website with a unique URL a
ccessible to you
and anyone you share the URL with. MongoDB may use this information to make pr
```

```
server with --bind_ip <addre
2018-10-13T14:01:51.850+0200 I CONTROL  [initandlisten] **          addresse
it should serve responses from, or with --bind_ip_all to
2018-10-13T14:01:51.896+0200 I CONTROL  [initandlisten] **          bind to a
l interfaces. If this behavior is desired, start the
2018-10-13T14:01:51.929+0200 I CONTROL  [initandlisten] **          server wi
h --bind_ip 127.0.0.1 to disable this warning.
2018-10-13T14:01:51.985+0200 I CONTROL  [initandlisten]
2018-10-13T14:01:51.997+0200 I STORAGE  [initandlisten] createCollection: adm
n.system.version with provided UUID: e5111ac9-ccb8-4865-a468-1dcfefe6b5e0
2018-10-13T14:01:52.117+0200 I COMMAND  [initandlisten] setting featureCompat
bilityVersion to 4.0
2018-10-13T14:01:52.131+0200 I STORAGE  [initandlisten] createCollection: loc
l.startup_log with generated UUID: 1112df95-6776-4cd5-bcf8-07b84b80fc27
2018-10-13T14:01:52.420+0200 I FTDC     [initandlisten] Initializing full-tim
 diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2018-10-13T14:01:52.434+0200 I STORAGE  [LogicalSessionCacheRefresh] createCo
lection: config.system.sessions with generated UUID: 6fa27513-c8a3-4994-8489-
064c308c7ee
2018-10-13T14:01:52.434+0200 I NETWORK  [initandlisten] waiting for connectio
s on port 27017
2018-10-13T14:01:52.710+0200 I INDEX    [LogicalSessionCacheRefresh] build in
ex on: config.system.sessions properties: { v: 2, key: { lastUse: 1 }, name:
lsidTTLIndex", ns: "config.system.sessions", expireAfterSeconds: 1800 }
2018-10-13T14:01:52.754+0200 I INDEX    [LogicalSessionCacheRefresh]      buil
ing index using bulk method; build may temporarily use up to 500 megabytes of
RAM
2018-10-13T14:01:52.780+0200 I INDEX    [LogicalSessionCacheRefresh] build in
ex done.  scanned 0 total records. 0 secs
2018-10-13T14:01:52.814+0200 I COMMAND  [LogicalSessionCacheRefresh] command
```

In the new command prompt run the "mongo" command to get the mongo db terminal.



```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use mylib
switched to db mylib
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.books.insert({"name":"mongodb book"})
WriteResult({ "nInserted" : 1 })
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
mylib   0.000GB
> show collections;
books
```

DATABASE

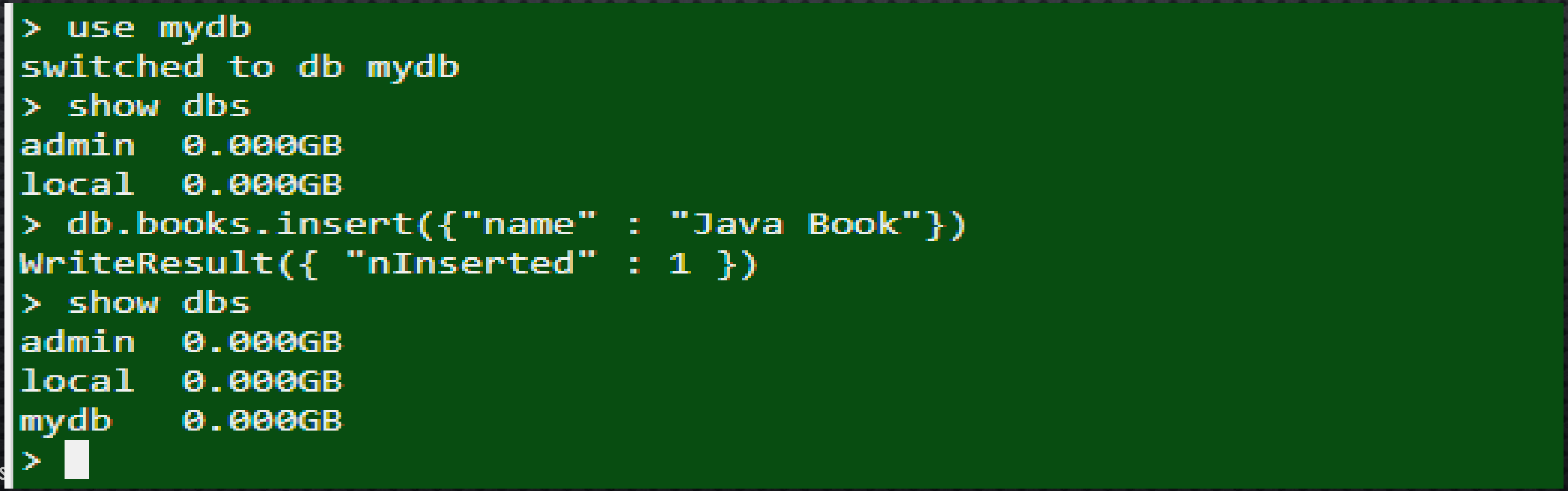
ChandraSekhar(CS) Baratm

1 CREATE DATABASE

MongoDB do not provide any command to create database. If there is no existing database, the following command is used to create a new database.

Syntax:

use DATABASE_NAME



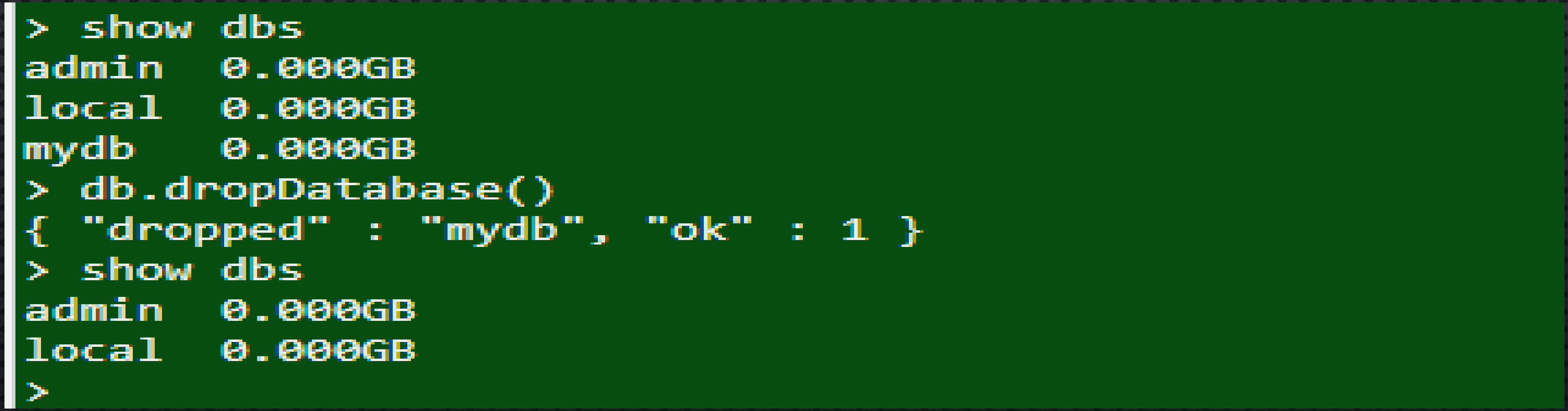
```
> use mydb
switched to db mydb
> show dbs
admin   0.000GB
local   0.000GB
> db.books.insert({"name" : "Java Book"})
WriteResult({ "nInserted" : 1 })
> show dbs
admin   0.000GB
local   0.000GB
mydb    0.000GB
>
```

2 DROP DATABASE

The dropDatabase command is used to drop a database. It also deletes the associated data files. It operates on the current database.

Syntax:

    db.dropDatabase()



```
> show dbs
admin   0.000GB
local   0.000GB
mydb    0.000GB
> db.dropDatabase()
{ "dropped" : "mydb", "ok" : 1 }
> show dbs
admin   0.000GB
local   0.000GB
>
```

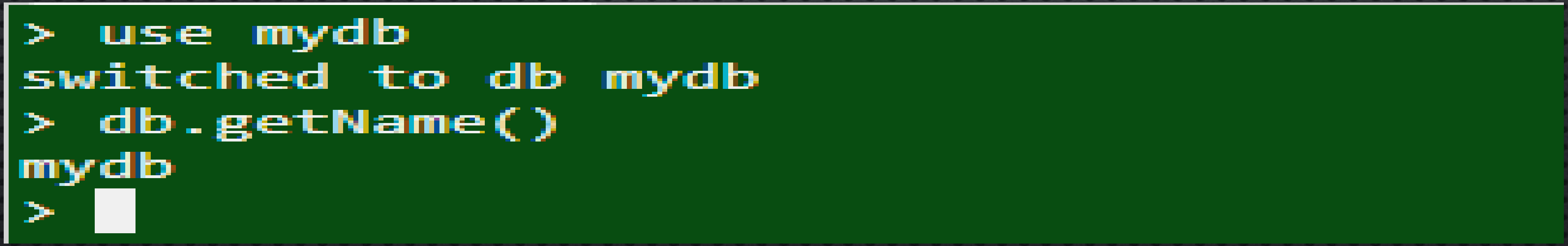
ChandraSekhar(CS) Baratm

3 FIND CURRENT DATABASE

The getName command is used to display the database name. It operates on the current database.

Syntax:

```
db.getName()
```



```
> use mydb
switched to db mydb
> db.getName()
mydb
>
```

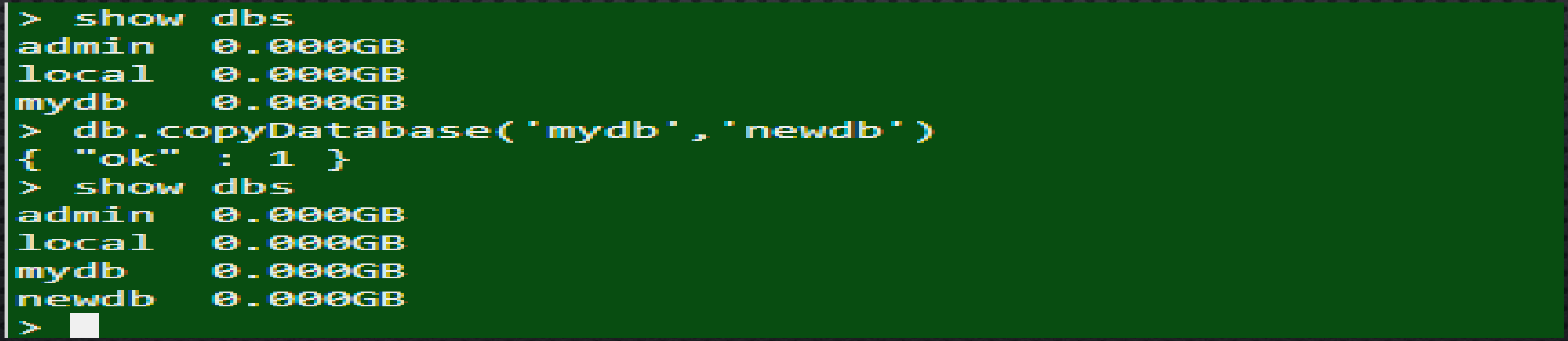
ChandraSekhar(CS) Baratm

4 COPY DATABASE

The copyDatabase command is used to copy or rename the database. It also copy the associated data files.

Syntax:

db.copyDatabase('old', 'new')



```
> show dbs
admin   0.000GB
local   0.000GB
mydb    0.000GB
> db.copyDatabase('mydb','newdb')
{ "ok" : 1 }
> show dbs
admin   0.000GB
local   0.000GB
mydb    0.000GB
newdb   0.000GB
> 
```

ChandraSekhar(CS) Baratm

COLLECTION

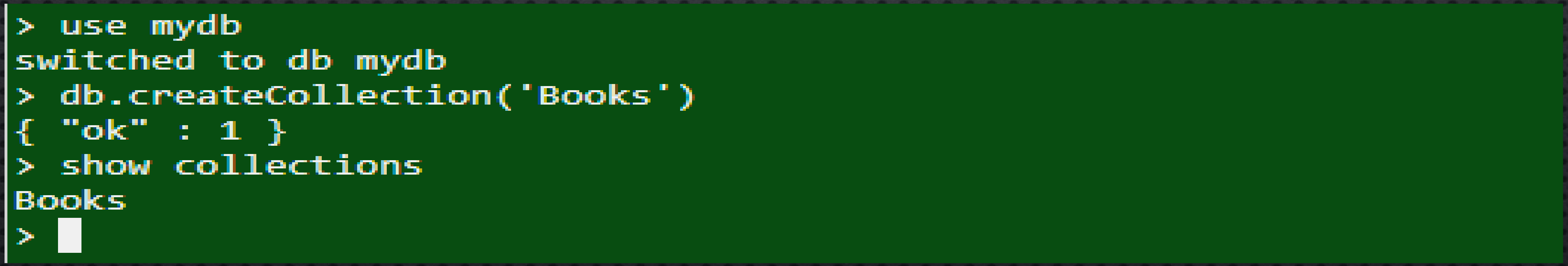
ChandraSekhar(CS) Baratm

1 CREATE COLLECTION

In MongoDB, createCollection command is used to create a collection. It operates on the current database.

Syntax:

db.createCollection('CollectionName')



```
> use mydb
switched to db mydb
> db.createCollection('Books')
{ "ok" : 1 }
> show collections
Books
>
```

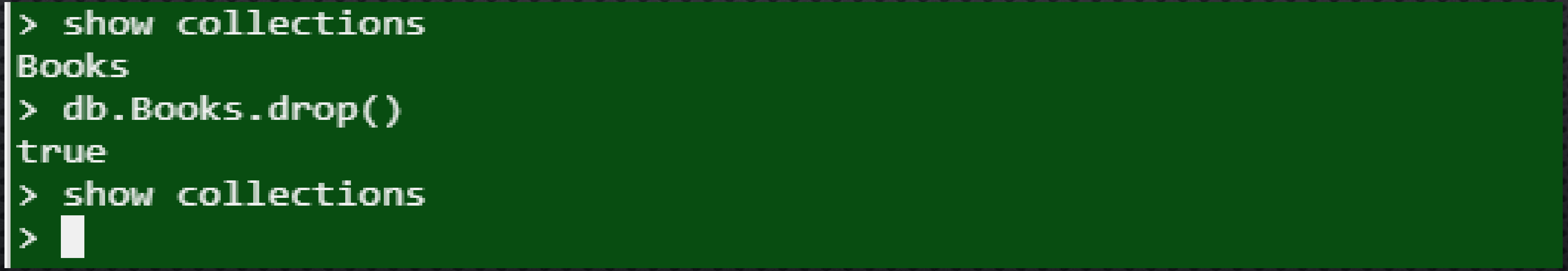
ChandraSekhar(CS) Baratm

2 DROP COLLECTION

In MongoDB, db.collection.drop() method is used to drop a collection from a database. It completely removes a collection from the database and does not leave any indexes associated with the dropped collections.

Syntax:

```
db.collection_name.drop()
```



```
> show collections
Books
> db.Books.drop()
true
> show collections
>
```

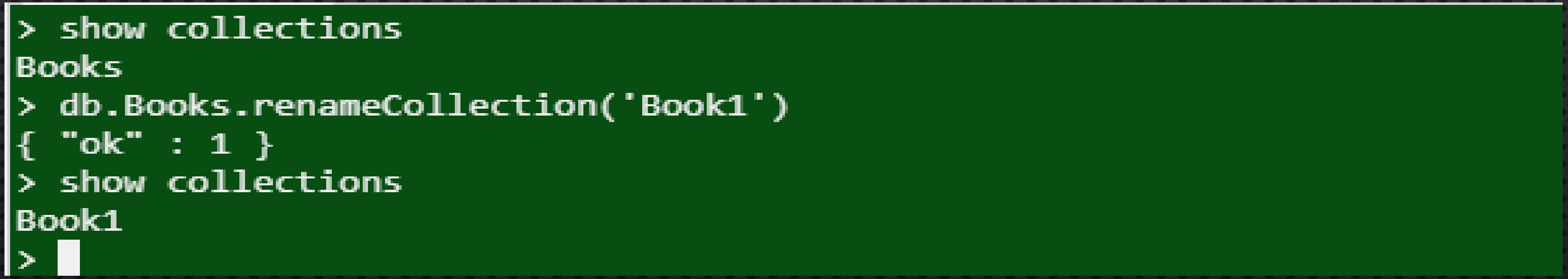
ChandraSekhar(CS) Baratm

3 RENAME COLLECTION

In MongoDB, db.collection.renameCollection() method is used to rename a collection.

Syntax:

db.collection_name.renameCollection('new_name')



```
> show collections
Books
> db.Books.renameCollection('Book1')
{ "ok" : 1 }
> show collections
Book1
> 
```

ChandraSekhar(CS) Baratm

CRUD

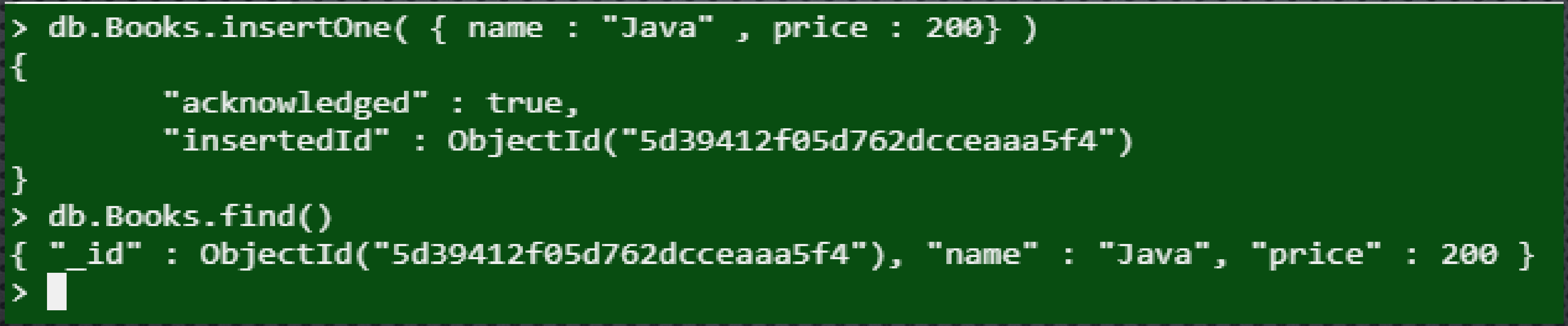
ChandraSekhar(CS) Baratm

INSERT

- ❖ db.collection.insertOne()
  Inserts a single document into a collection.
- ❖ db.collection.insertMany()
  inserts multiple documents into a collection.
- ❖ db.collection.insert()
  inserts a single document or multiple documents into a collection.

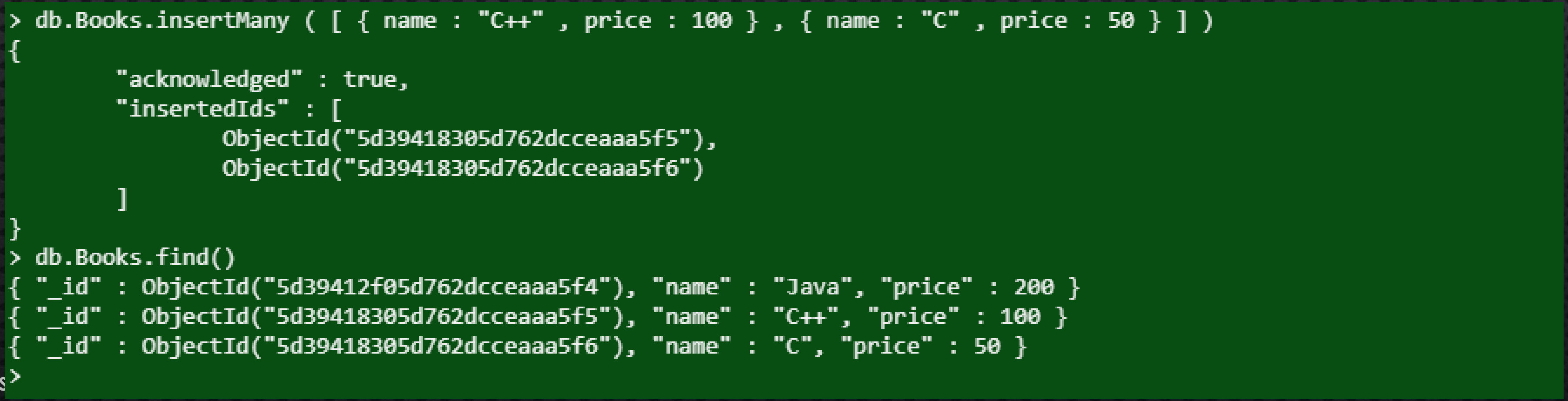
INSERT...

db.collection.insertOne()



```
> db.Books.insertOne( { name : "Java" , price : 200} )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5d39412f05d762dcceaaa5f4")
}
> db.Books.find()
{ "_id" : ObjectId("5d39412f05d762dcceaaa5f4"), "name" : "Java", "price" : 200 }
>
```

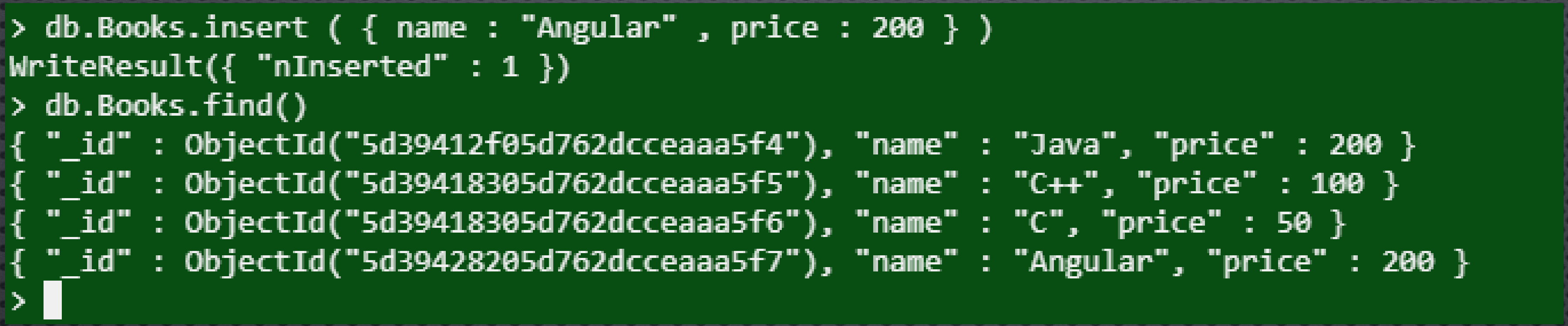
db.collection.insertMany()



```
> db.Books.insertMany ( [ { name : "C++" , price : 100 } , { name : "C" , price : 50 } ] )
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5d39418305d762dcceaaa5f5"),
                ObjectId("5d39418305d762dcceaaa5f6")
        ]
}
> db.Books.find()
{ "_id" : ObjectId("5d39412f05d762dcceaaa5f4"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f5"), "name" : "C++", "price" : 100 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f6"), "name" : "C", "price" : 50 }
>
```

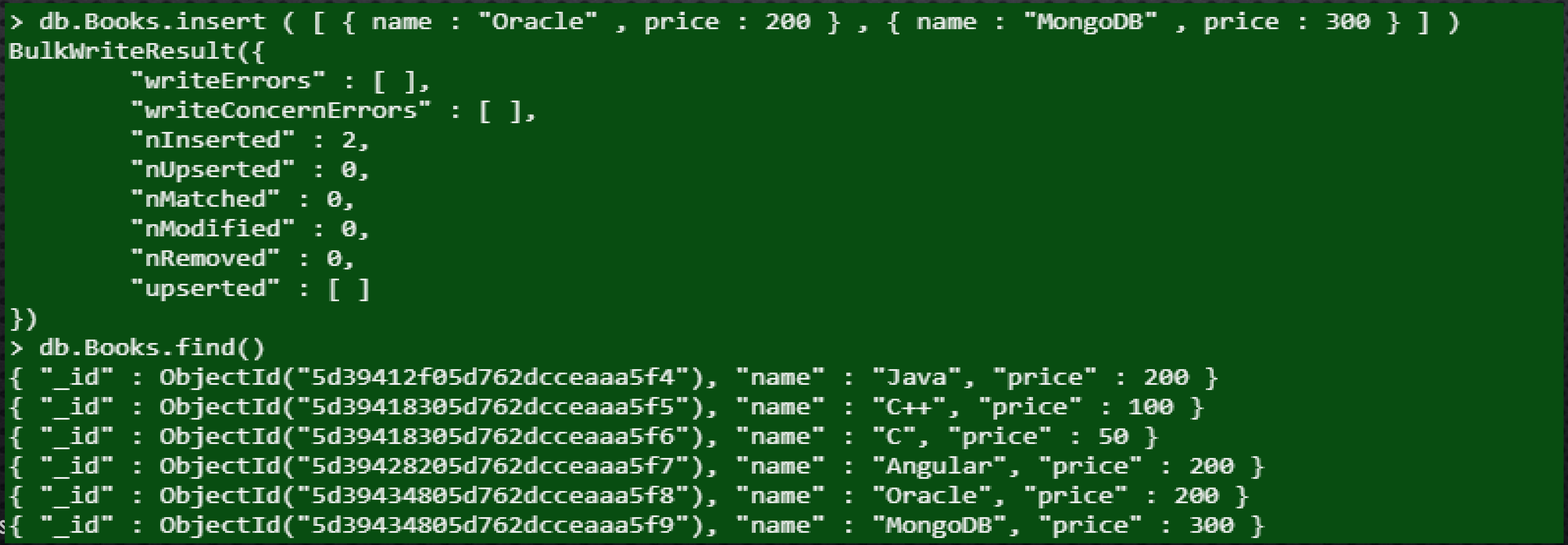
INSERT...

db.collection.insert()



```
> db.Books.insert ( { name : "Angular" , price : 200 } )
WriteResult({ "nInserted" : 1 })
> db.Books.find()
{ "_id" : ObjectId("5d39412f05d762dcceaaa5f4"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f5"), "name" : "C++", "price" : 100 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f6"), "name" : "C", "price" : 50 }
{ "_id" : ObjectId("5d39428205d762dcceaaa5f7"), "name" : "Angular", "price" : 200 }
>
```

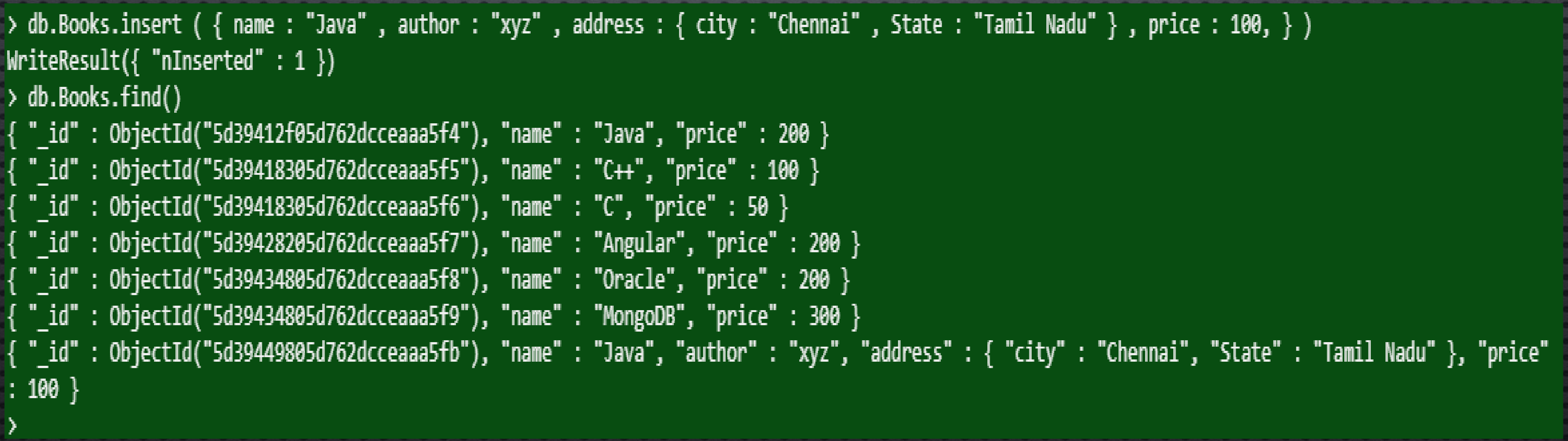
db.collection.insert()



```
> db.Books.insert ( [ { name : "Oracle" , price : 200 } , { name : "MongoDB" , price : 300 } ] )
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.Books.find()
{ "_id" : ObjectId("5d39412f05d762dcceaaa5f4"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f5"), "name" : "C++", "price" : 100 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f6"), "name" : "C", "price" : 50 }
{ "_id" : ObjectId("5d39428205d762dcceaaa5f7"), "name" : "Angular", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f8"), "name" : "Oracle", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f9"), "name" : "MongoDB", "price" : 300 }
```

INSERT...

db.collection.insert()



```
> db.Books.insert ( { name : "Java" , author : "xyz" , address : { city : "Chennai" , State : "Tamil Nadu" } , price : 100, } )
WriteResult({ "nInserted" : 1 })
> db.Books.find()
{ "_id" : ObjectId("5d39412f05d762dcceaaa5f4"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f5"), "name" : "C++", "price" : 100 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f6"), "name" : "C", "price" : 50 }
{ "_id" : ObjectId("5d39428205d762dcceaaa5f7"), "name" : "Angular", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f8"), "name" : "Oracle", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f9"), "name" : "MongoDB", "price" : 300 }
{ "_id" : ObjectId("5d39449805d762dcceaaa5fb"), "name" : "Java", "author" : "xyz", "address" : { "city" : "Chennai", "State" : "Tamil Nadu" }, "price"
: 100 }
>
```

F2ID

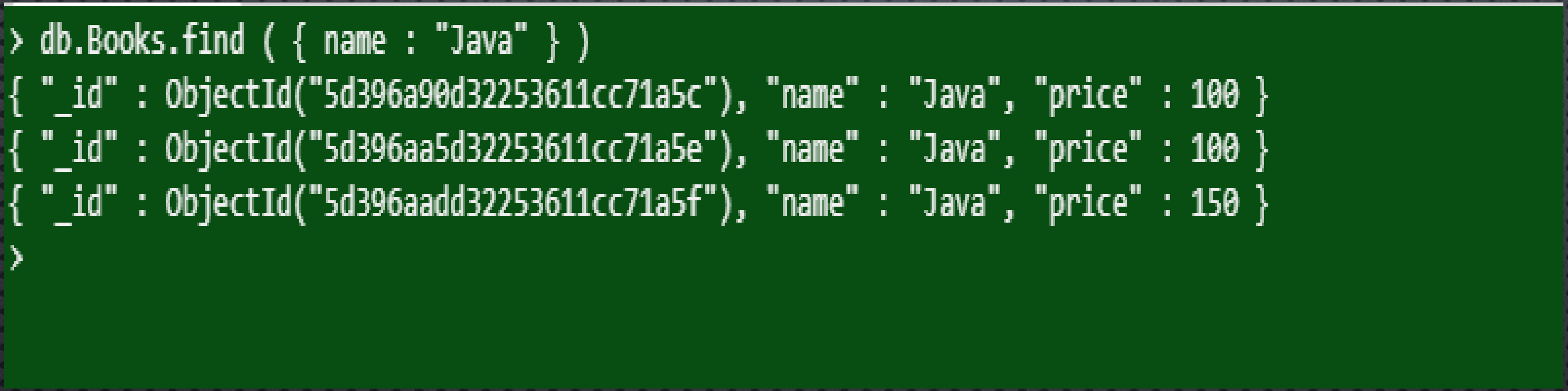
- db.collection.find()
- db.collection.find({ })

To retrieve all documents from a collection.

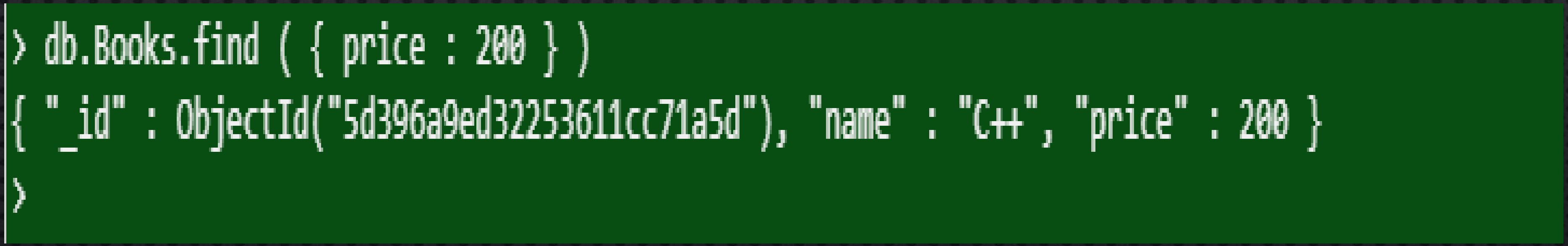
ChandraSekhar(CS) Baratm

FIND...

Find all documents that match a condition.



```
> db.Books.find ( { name : "Java" } )
{ "_id" : ObjectId("5d396a90d32253611cc71a5c"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d396aa5d32253611cc71a5e"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d396aadd32253611cc71a5f"), "name" : "Java", "price" : 150 }
>
```



```
> db.Books.find ( { price : 200 } )
{ "_id" : ObjectId("5d396a9ed32253611cc71a5d"), "name" : "C++", "price" : 200 }
>
```

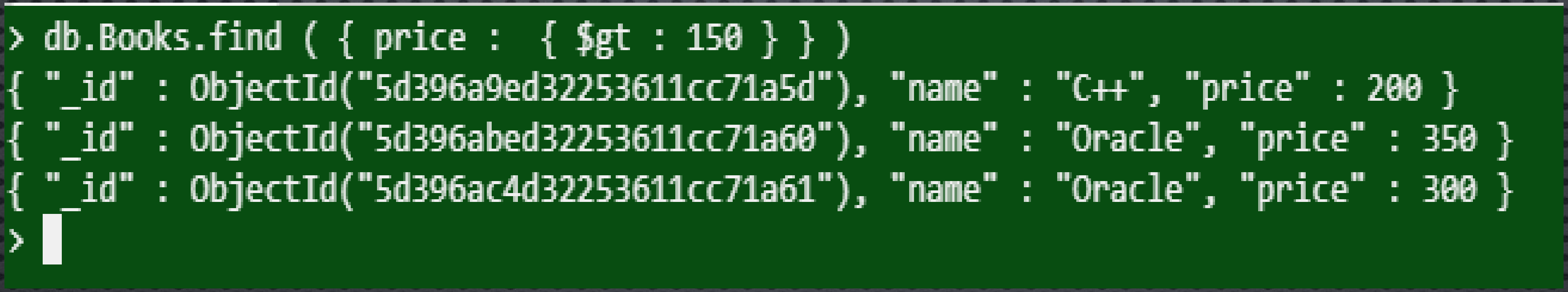
FI2D...

Comparison

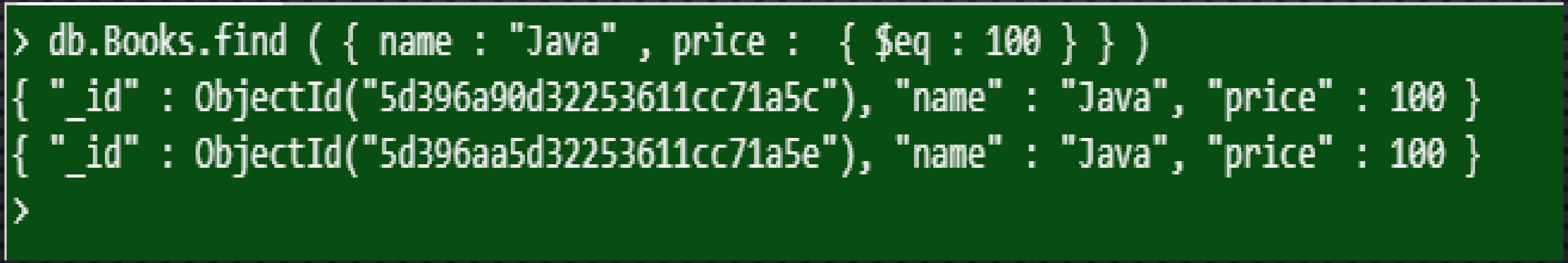
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

FI 2 D...

Comparison...



```
> db.Books.find ( { price :  { $gt : 150 } } )
{ "_id" : ObjectId("5d396a9ed32253611cc71a5d"), "name" : "C++", "price" : 200 }
{ "_id" : ObjectId("5d396abed32253611cc71a60"), "name" : "Oracle", "price" : 350 }
{ "_id" : ObjectId("5d396ac4d32253611cc71a61"), "name" : "Oracle", "price" : 300 }
>
```

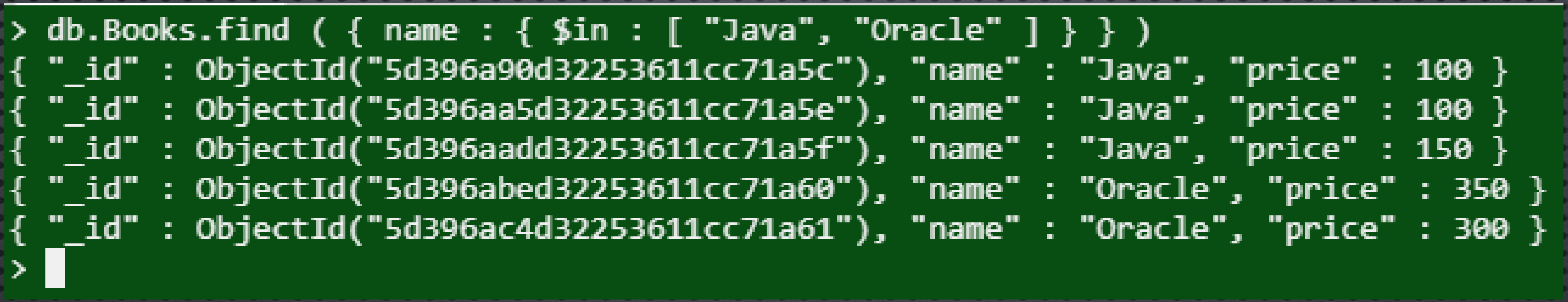


```
> db.Books.find ( { name : "Java" , price :  { $eq : 100 } } )
{ "_id" : ObjectId("5d396a90d32253611cc71a5c"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d396aa5d32253611cc71a5e"), "name" : "Java", "price" : 100 }
>
```

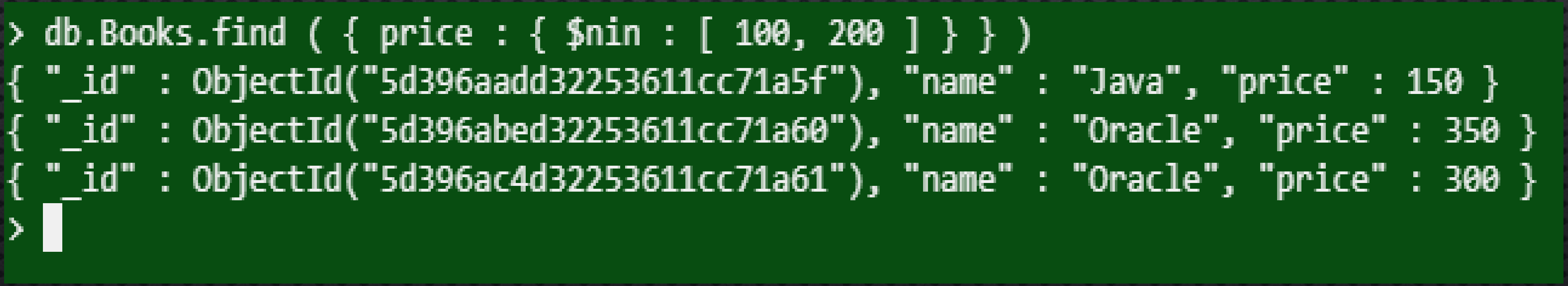
FIND...

2

Comparison...



```
> db.Books.find ( { name : { $in : [ "Java", "Oracle" ] } } )
{ "_id" : ObjectId("5d396a90d32253611cc71a5c"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d396aa5d32253611cc71a5e"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d396aadd32253611cc71a5f"), "name" : "Java", "price" : 150 }
{ "_id" : ObjectId("5d396abed32253611cc71a60"), "name" : "Oracle", "price" : 350 }
{ "_id" : ObjectId("5d396ac4d32253611cc71a61"), "name" : "Oracle", "price" : 300 }
>
```



```
> db.Books.find ( { price : { $nin : [ 100, 200 ] } } )
{ "_id" : ObjectId("5d396aadd32253611cc71a5f"), "name" : "Java", "price" : 150 }
{ "_id" : ObjectId("5d396abed32253611cc71a60"), "name" : "Oracle", "price" : 350 }
{ "_id" : ObjectId("5d396ac4d32253611cc71a61"), "name" : "Oracle", "price" : 300 }
>
```

FI2D...

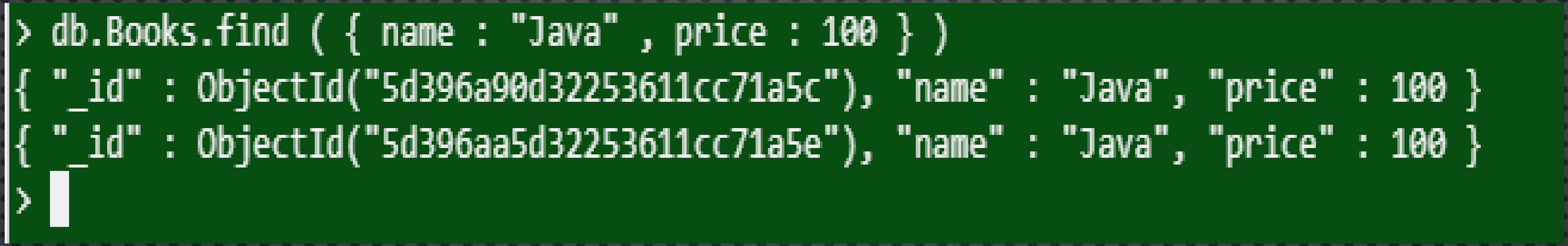
Logical

- $and
  - Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
- $not
  - Inverts the effect of a query expression and returns documents that do not match the query expression.
- $nor
  - Joins query clauses 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 clause.

FI2D...

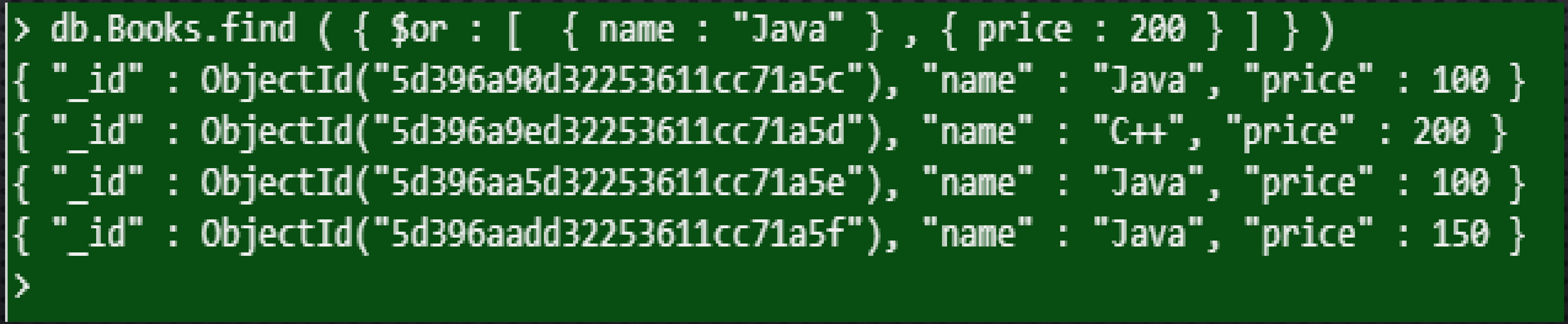
Logical

Implicit AND



```
> db.Books.find ( { name : "Java" , price : 100 } )
{ "_id" : ObjectId("5d396a90d32253611cc71a5c"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d396aa5d32253611cc71a5e"), "name" : "Java", "price" : 100 }
>
```

OR



```
> db.Books.find ( { $or : [  { name : "Java" } , { price : 200 } ] } )
{ "_id" : ObjectId("5d396a90d32253611cc71a5c"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d396a9ed32253611cc71a5d"), "name" : "C++", "price" : 200 }
{ "_id" : ObjectId("5d396aa5d32253611cc71a5e"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d396aadd32253611cc71a5f"), "name" : "Java", "price" : 150 }
>
```

DE3ETE

- db.collection.deleteOne()

  Delete at most a single document that match a specified filter even though multiple documents may match the specified filter..

- db.collection.deleteMany()

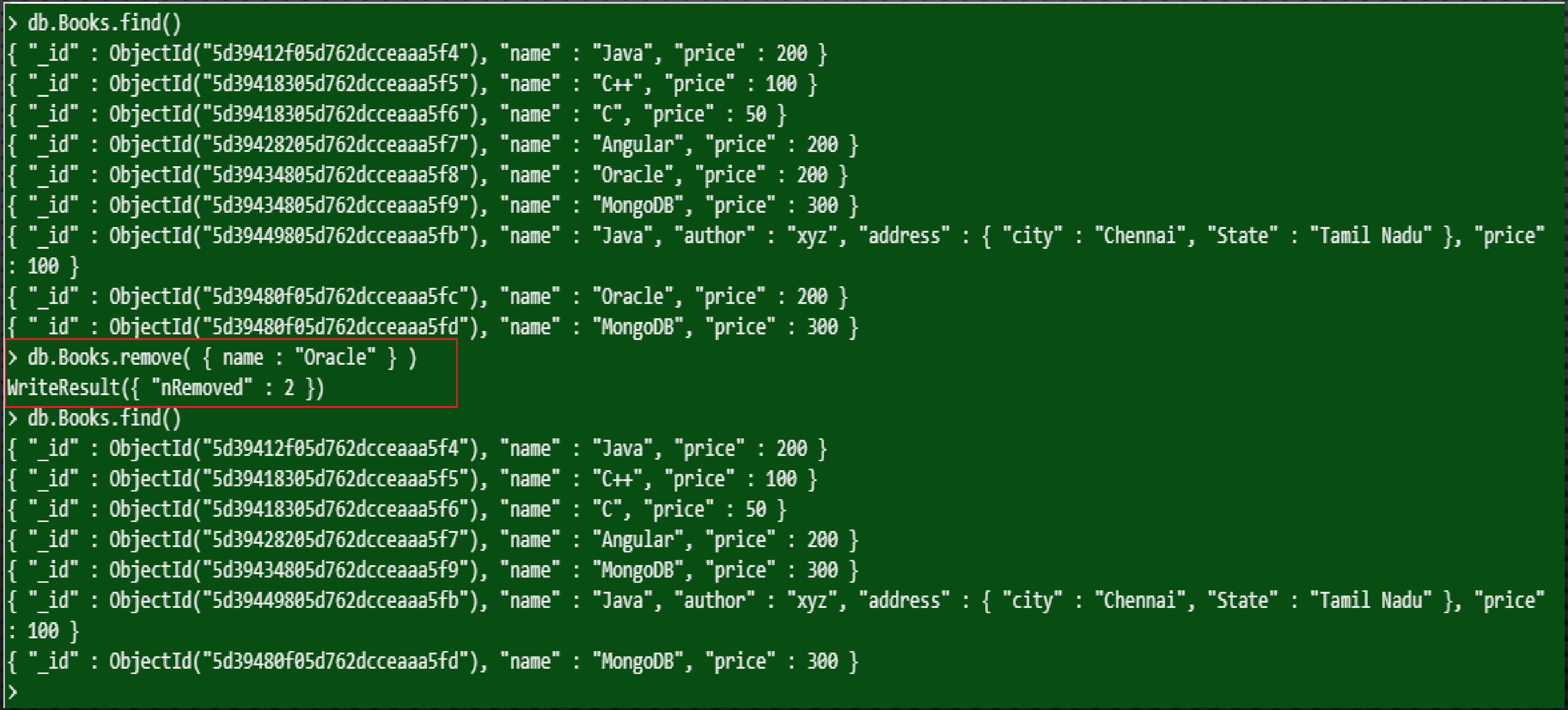
  Delete all documents that match a specified filter..

- db.collection.remove()

  Delete a single document or all documents that match a specified filter.

DE3ETE…

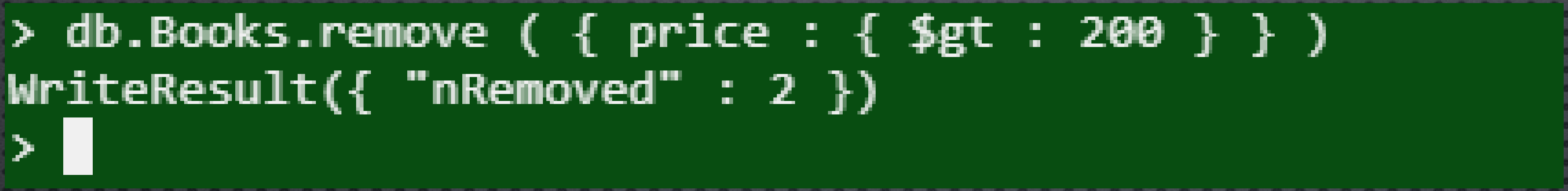
Remove all documents that match a condition.



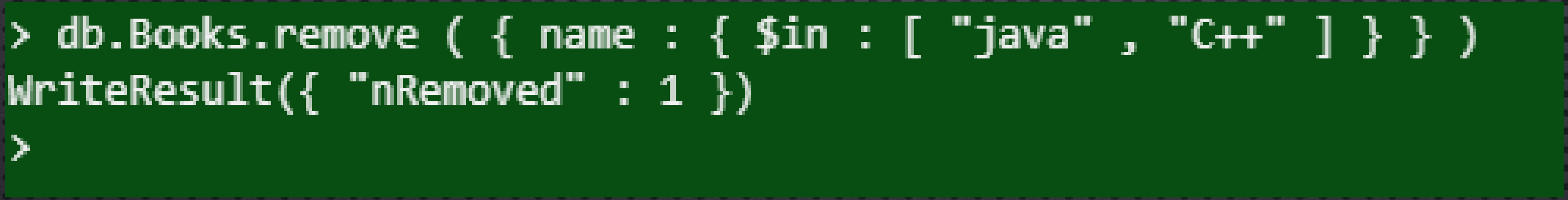
```
> db.Books.find()
{ "_id" : ObjectId("5d39412f05d762dcceaaa5f4"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f5"), "name" : "C++", "price" : 100 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f6"), "name" : "C", "price" : 50 }
{ "_id" : ObjectId("5d39428205d762dcceaaa5f7"), "name" : "Angular", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f8"), "name" : "Oracle", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f9"), "name" : "MongoDB", "price" : 300 }
{ "_id" : ObjectId("5d39449805d762dcceaaa5fb"), "name" : "Java", "author" : "xyz", "address" : { "city" : "Chennai", "State" : "Tamil Nadu" }, "price"
: 100 }
{ "_id" : ObjectId("5d39480f05d762dcceaaa5fc"), "name" : "Oracle", "price" : 200 }
{ "_id" : ObjectId("5d39480f05d762dcceaaa5fd"), "name" : "MongoDB", "price" : 300 }
> db.Books.remove( { name : "Oracle" } )
WriteResult({ "nRemoved" : 2 })
> db.Books.find()
{ "_id" : ObjectId("5d39412f05d762dcceaaa5f4"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f5"), "name" : "C++", "price" : 100 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f6"), "name" : "C", "price" : 50 }
{ "_id" : ObjectId("5d39428205d762dcceaaa5f7"), "name" : "Angular", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f9"), "name" : "MongoDB", "price" : 300 }
{ "_id" : ObjectId("5d39449805d762dcceaaa5fb"), "name" : "Java", "author" : "xyz", "address" : { "city" : "Chennai", "State" : "Tamil Nadu" }, "price"
: 100 }
{ "_id" : ObjectId("5d39480f05d762dcceaaa5fd"), "name" : "MongoDB", "price" : 300 }
>
```

DELETE…

Remove all documents that match a condition.

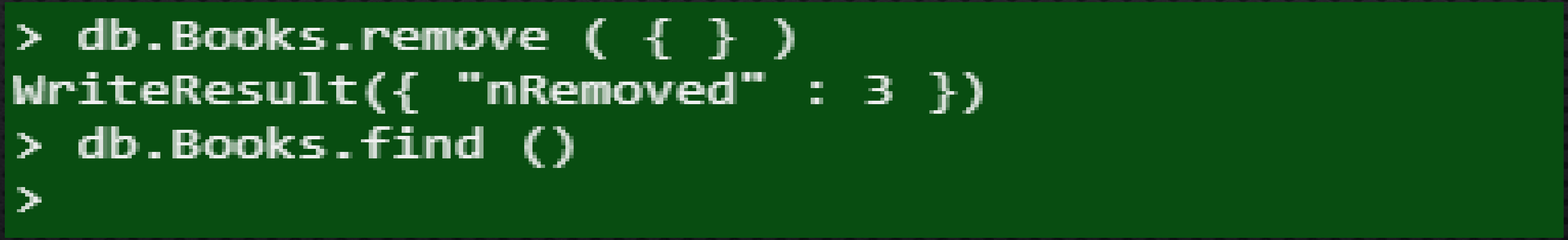


```
> db.Books.remove ( { price : { $gt : 200 } } )
WriteResult({ "nRemoved" : 2 })
>
```



```
> db.Books.remove ( { name : { $in : [ "java" , "C++" ] } } )
WriteResult({ "nRemoved" : 1 })
>
```

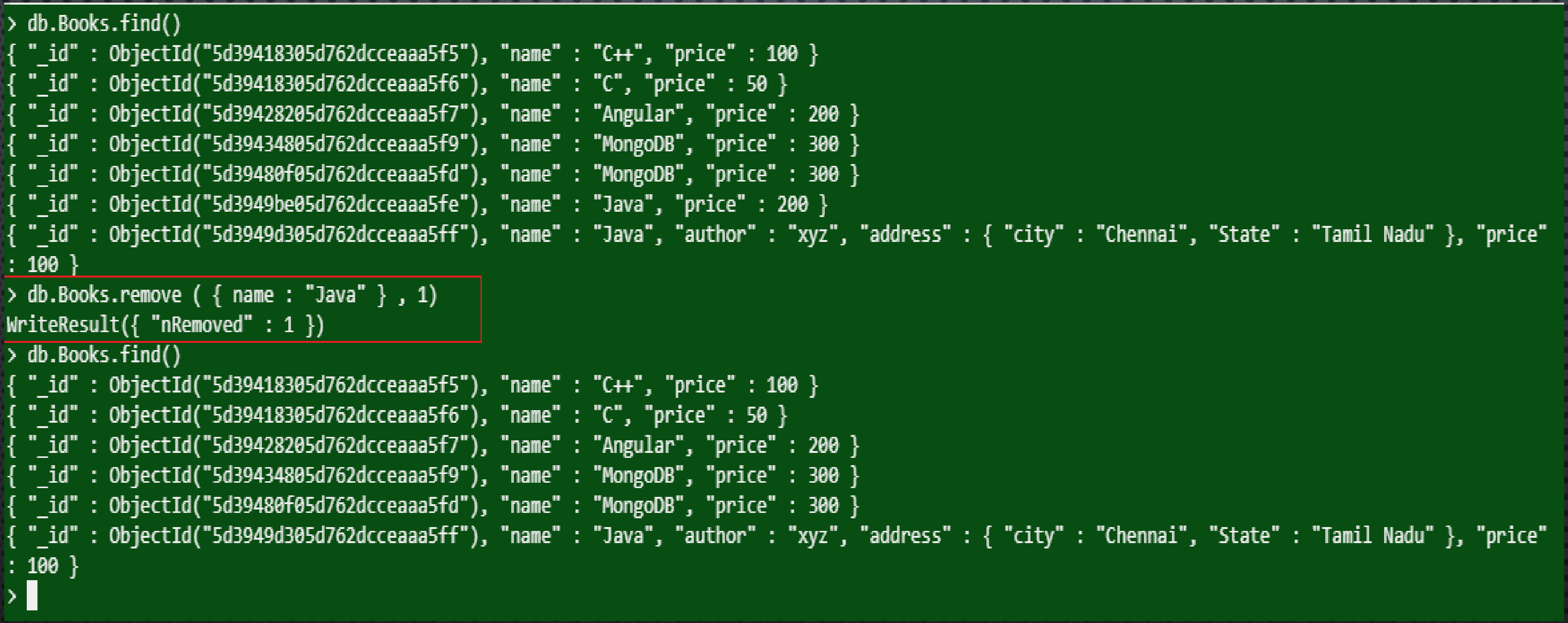
Remove all documents.



```
> db.Books.remove ( { } )
WriteResult({ "nRemoved" : 3 })
> db.Books.find ()
>
```

DE3ETE...

Remove a single document that match a condition.



```
> db.Books.find()
{ "_id" : ObjectId("5d39418305d762dcceaaa5f5"), "name" : "C++", "price" : 100 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f6"), "name" : "C", "price" : 50 }
{ "_id" : ObjectId("5d39428205d762dcceaaa5f7"), "name" : "Angular", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f9"), "name" : "MongoDB", "price" : 300 }
{ "_id" : ObjectId("5d39480f05d762dcceaaa5fd"), "name" : "MongoDB", "price" : 300 }
{ "_id" : ObjectId("5d3949be05d762dcceaaa5fe"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d3949d305d762dcceaaa5ff"), "name" : "Java", "author" : "xyz", "address" : { "city" : "Chennai", "State" : "Tamil Nadu" }, "price"
: 100 }
> db.Books.remove ( { name : "Java" } , 1)
WriteResult({ "nRemoved" : 1 })
> db.Books.find()
{ "_id" : ObjectId("5d39418305d762dcceaaa5f5"), "name" : "C++", "price" : 100 }
{ "_id" : ObjectId("5d39418305d762dcceaaa5f6"), "name" : "C", "price" : 50 }
{ "_id" : ObjectId("5d39428205d762dcceaaa5f7"), "name" : "Angular", "price" : 200 }
{ "_id" : ObjectId("5d39434805d762dcceaaa5f9"), "name" : "MongoDB", "price" : 300 }
{ "_id" : ObjectId("5d39480f05d762dcceaaa5fd"), "name" : "MongoDB", "price" : 300 }
{ "_id" : ObjectId("5d3949d305d762dcceaaa5ff"), "name" : "Java", "author" : "xyz", "address" : { "city" : "Chennai", "State" : "Tamil Nadu" }, "price"
: 100 }
> 
```

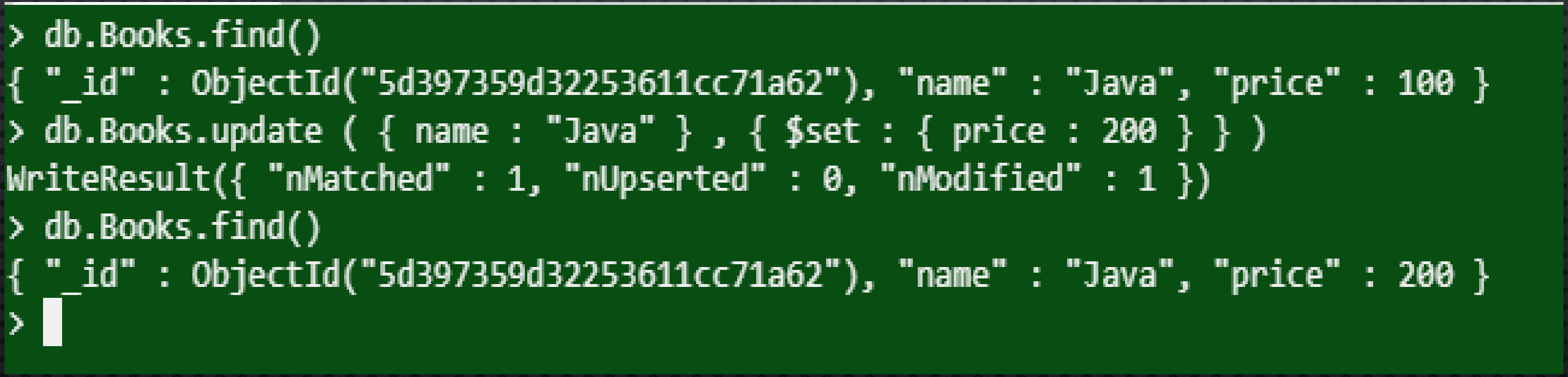
Note : 1 is true, To enable single remove. And 0 is false to disable single remove. By default is 0. Also, Instead of 1 / 0 we can use true / false.

UPDATE

In MongoDB, update() method is used to update or modify the existing documents of a collection.

Syntax:

db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)



```
> db.Books.find()
{ "_id" : ObjectId("5d397359d32253611cc71a62"), "name" : "Java", "price" : 100 }
> db.Books.update ( { name : "Java" } , { $set : { price : 200 } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Books.find()
{ "_id" : ObjectId("5d397359d32253611cc71a62"), "name" : "Java", "price" : 200 }
>
```

LIMITING RECORDS

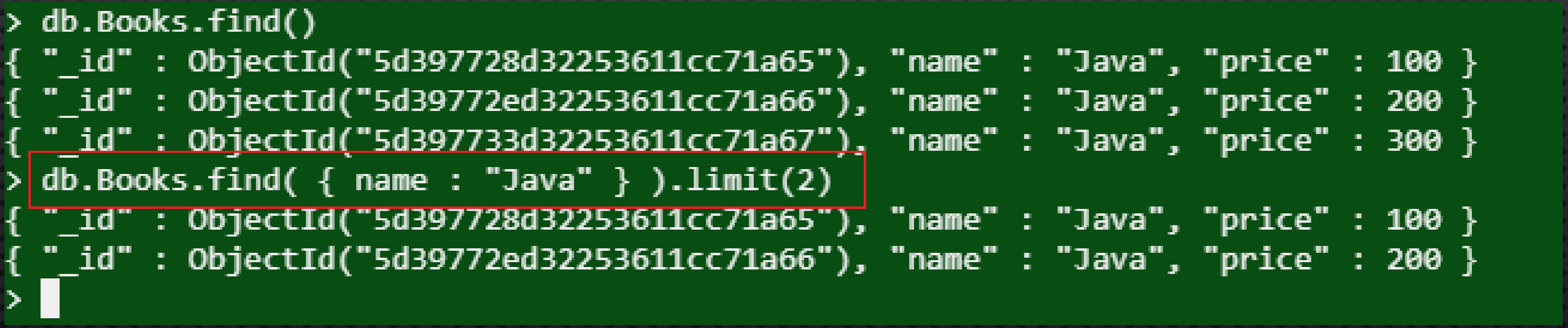
ChandraSekhar(CS) Baratm

LIMIT

To limit the records in MongoDB, you need to use limit() method. The method accepts one number type argument, which is the number of documents that you want to be displayed..

Syntax:

db.COLLECTION_NAME.find().limit(NUMBER)



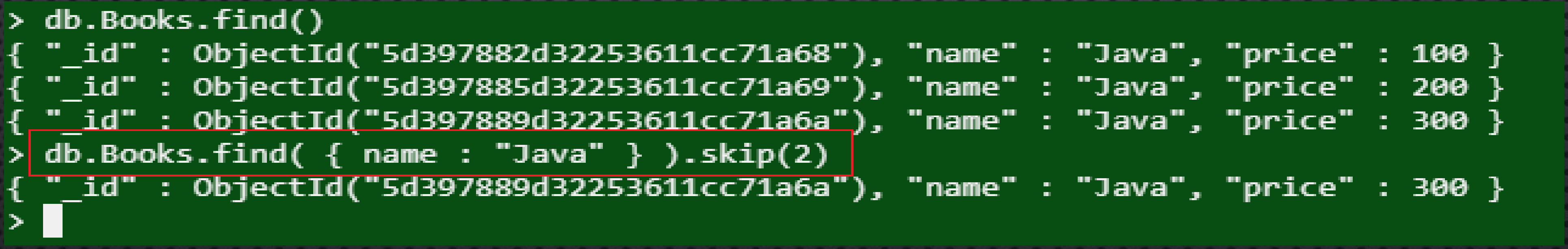
```
> db.Books.find()
{ "_id" : ObjectId("5d397728d32253611cc71a65"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d39772ed32253611cc71a66"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d397733d32253611cc71a67"), "name" : "Java", "price" : 300 }
> db.Books.find( { name : "Java" } ).limit(2)
{ "_id" : ObjectId("5d397728d32253611cc71a65"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d39772ed32253611cc71a66"), "name" : "Java", "price" : 200 }
>
```

SKIP

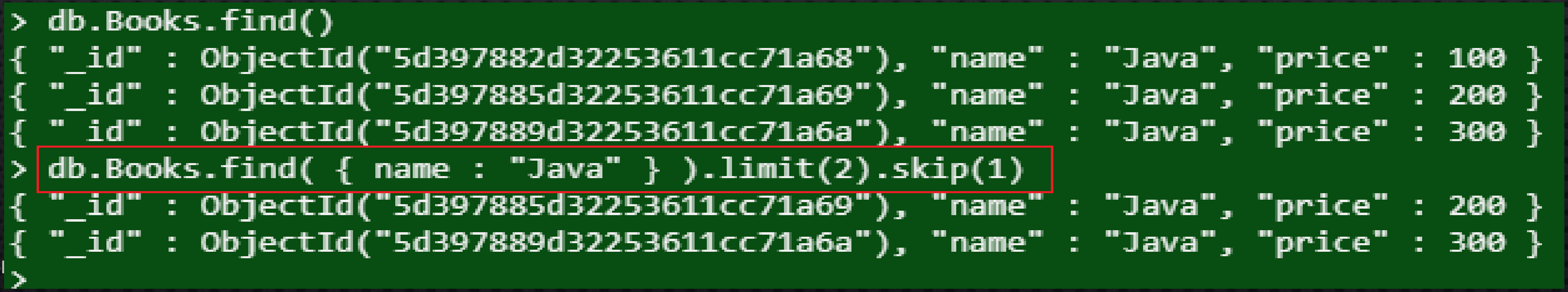
Apart from limit() method, there is one more method skip() which also accepts number type argument and is used to skip the number of documents.

Syntax:

db.COLLECTION_NAME.find().skip(NUMBER)



```
> db.Books.find()
{ "_id" : ObjectId("5d397882d32253611cc71a68"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d397885d32253611cc71a69"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d397889d32253611cc71a6a"), "name" : "Java", "price" : 300 }
> db.Books.find( { name : "Java" } ).skip(2)
{ "_id" : ObjectId("5d397889d32253611cc71a6a"), "name" : "Java", "price" : 300 }
>
```



```
> db.Books.find()
{ "_id" : ObjectId("5d397882d32253611cc71a68"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d397885d32253611cc71a69"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d397889d32253611cc71a6a"), "name" : "Java", "price" : 300 }
> db.Books.find( { name : "Java" } ).limit(2).skip(1)
{ "_id" : ObjectId("5d397885d32253611cc71a69"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d397889d32253611cc71a6a"), "name" : "Java", "price" : 300 }
>
```

SORTING RECORDS

ChandraSekhar(CS) Baratm

SORT

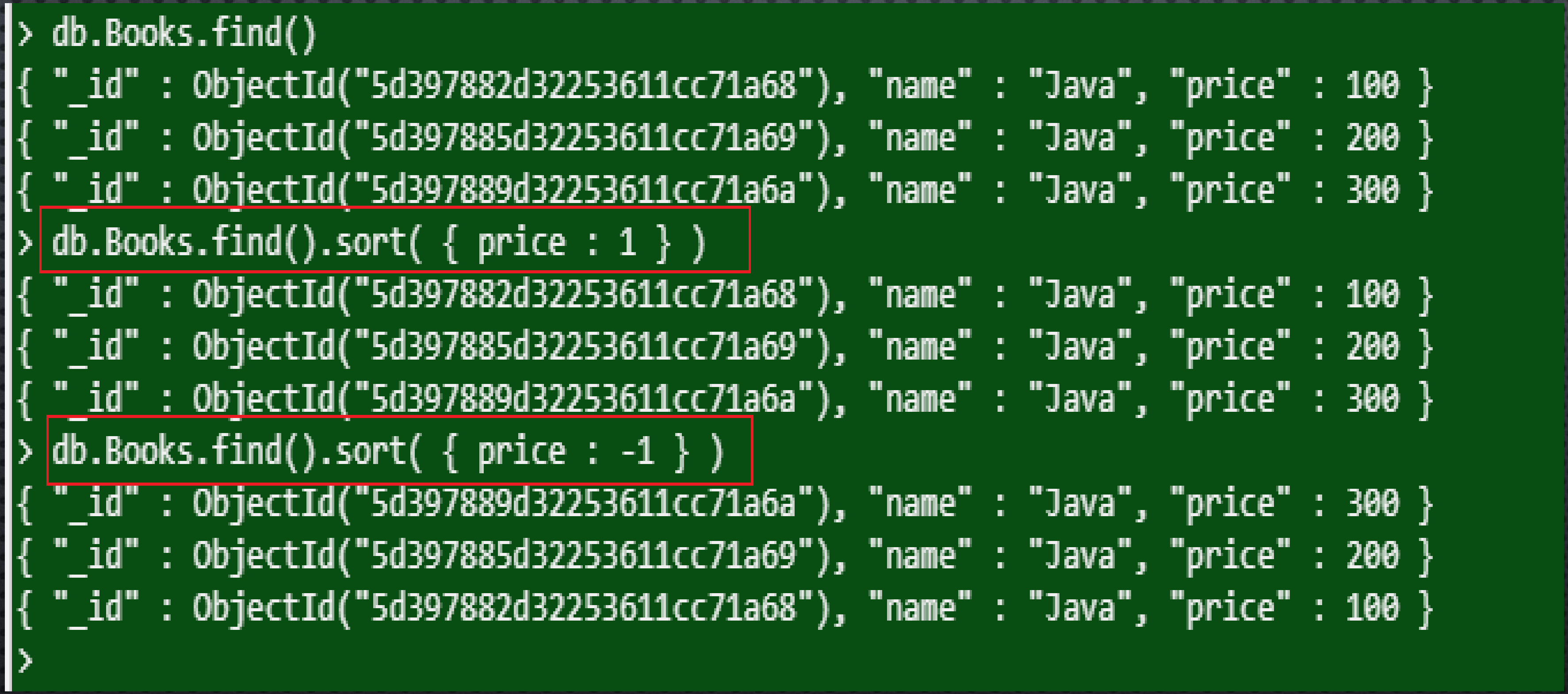
In MongoDB, sort() method is used to sort the documents in the collection. This method accepts a document containing list of fields along with their sorting order.

- The sorting order is specified as 1 or -1.
- 1 is used for ascending order sorting.
- -1 is used for descending order sorting.

Syntax:

```
db.COLLECTION_NAME.find().sort({KEY:1})
```

SORT…



```
> db.Books.find()
{ "_id" : ObjectId("5d397882d32253611cc71a68"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d397885d32253611cc71a69"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d397889d32253611cc71a6a"), "name" : "Java", "price" : 300 }
> db.Books.find().sort( { price : 1 } )
{ "_id" : ObjectId("5d397882d32253611cc71a68"), "name" : "Java", "price" : 100 }
{ "_id" : ObjectId("5d397885d32253611cc71a69"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d397889d32253611cc71a6a"), "name" : "Java", "price" : 300 }
> db.Books.find().sort( { price : -1 } )
{ "_id" : ObjectId("5d397889d32253611cc71a6a"), "name" : "Java", "price" : 300 }
{ "_id" : ObjectId("5d397885d32253611cc71a69"), "name" : "Java", "price" : 200 }
{ "_id" : ObjectId("5d397882d32253611cc71a68"), "name" : "Java", "price" : 100 }
>
```