Kristu Jayanti College

AUTONOMOUS Bengaluru

Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

NOSQL PRACTICAL RECORD

Bachelor of Computer Application (BCA)

Department of Computer Science (UG)

Submitted by:

NAME	**: VINITH M**
REG.NO	**: 20CS2K5300**
CLASS	**: V BCA ‘E’**
YEAR	**: 2022- 2023**

Kristu Jayanti College
AUTONOMOUS Bengaluru
Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

LABORATORY CERTIFICATE

This is to certify that Mr. / Ms. VINITH M

has satisfactorily completed the practical assignments of the

Course: NOSQL PROGRAMMING PRACTICAL LAB

prescribed by Kristu Jayanti College (Autonomous) Bangalore (Affiliated to Bangalore North University) during the academic year 2022 - 23

Date :

Signature of the Course Teacher

Head of the Department

Examiners (Name & Signature)

1.

2.

Name of the Student : VINITH M

Reg. No. 20CS2K5300

Programme & Semester : V BCA "E"

Date of Practical Examination :

Kristu Jayanti College
AUTONOMOUS Bengaluru
Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

CourseCode:BCADL2A51 **CourseTitle:NOSQL DATABASE PRACTICALS**

PARTA				
S.no	**Date**	**Exercise**	**Page No**	**Sign**
1.	02-09-21	Write a program to connect database, you need to specify the database name, if the database doesn't exist then MongoDB creates it automatically.	01	
2.	02-09-21	Write a program to create a collection, createCollection() method.	03	
3.	02-09-21	Write a program to insert a document into MongoDB,	05	
4.	02-09-21	Write a program to list all the collections in a database.	07	
5.	09-09-21	Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant.	09	
6.	09-09-21	Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.	12	
7.	09-09-21	Write a program to create collections for student data and to display the student_Id in ascending order.	15	
8.	09-09-21	Write a program to create collections for student data and to display the student_Name in descending order.	19	
9.	16-09-21	Write a MongoDB query to find the student data that achieved a score, more than 80 but less than 100.	24	
10.	30-09-21	Write a program to set up a replica set and to add members for the same.	27	
11.	16-09-21	Write a program to establish a connection between PHP and MongoDB.	31	
12.	16-09-21	Write a program to create a document in Collection Cars with fields name and price.	33	
13.	23-09-21	Write a program for updating the data in MongoDB.	35	
14.	23-09-21	Write a program for retrieving the data in MongoDB.	37	
15.	23-09-21	Write a program for deleting the data in MongoDB.	39	

1. **Write a program to connect database, you need to specify the database name, if the database doesn't exist then MongoDB creates it automatically.**

   Step 1: - show dbs

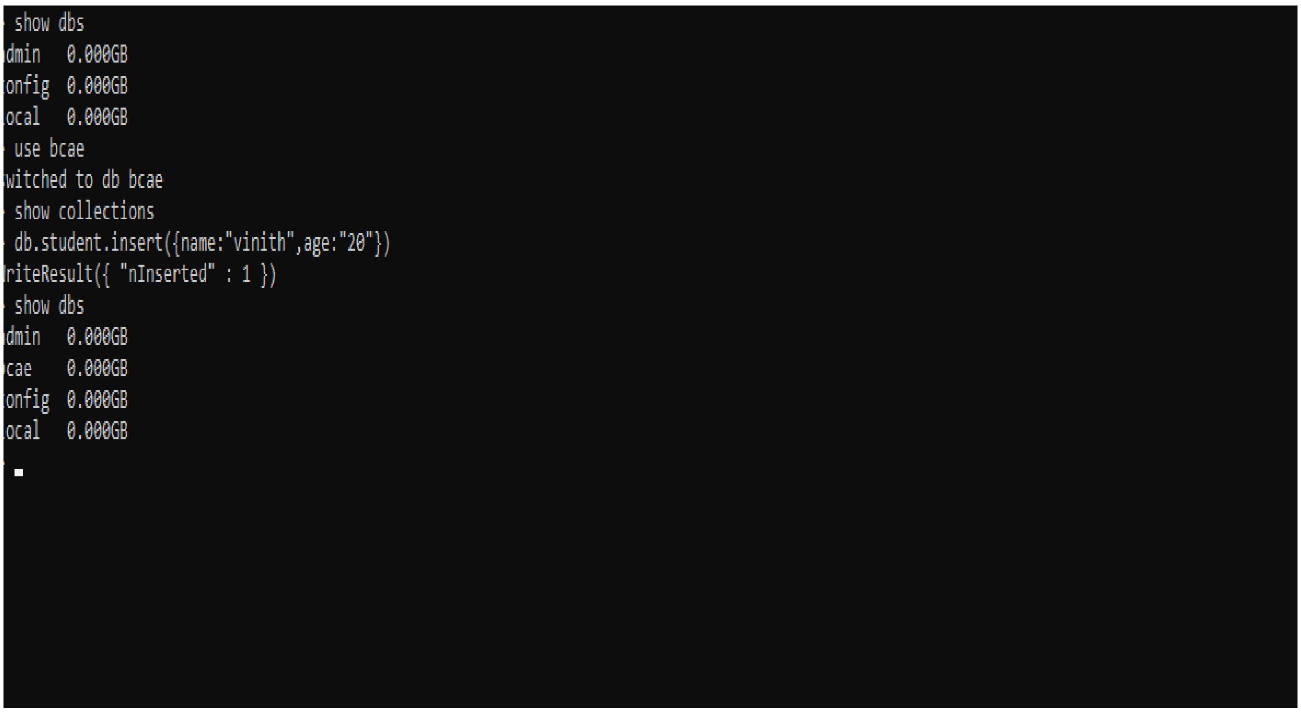
   Step 2:- use bcae

   Step 3:- show collections

   Step 4: -db.student.insert({Name:"Vinith",Age:20})

   Step 5:- show dbs

OUTPUT


```
 show dbs
dmin   0.000GB
onfig  0.000GB
ocal   0.000GB
 use bcae
witched to db bcae
 show collections
 db.student.insert({name:"vinith",age:"20"})
riteResult({ "nInserted" : 1 })
 show dbs
dmin   0.000GB
cae    0.000GB
onfig  0.000GB
ocal   0.000GB

 ■
```

2. Write a program to create a collection, createCollection() method.

<u>Step 1:</u> - use student

<u>Step 2:</u>- db.createCollection(“studentbca”)

<u>Step 3:</u>- db.studentbca.insert({"Name":"Vinith","Age":"22"})

db.studentbca.insert({"Name":"Harry Styles","Age":"21"})

db.studentbca.insert({"Name":"Billie Eilish","Age":"39"})

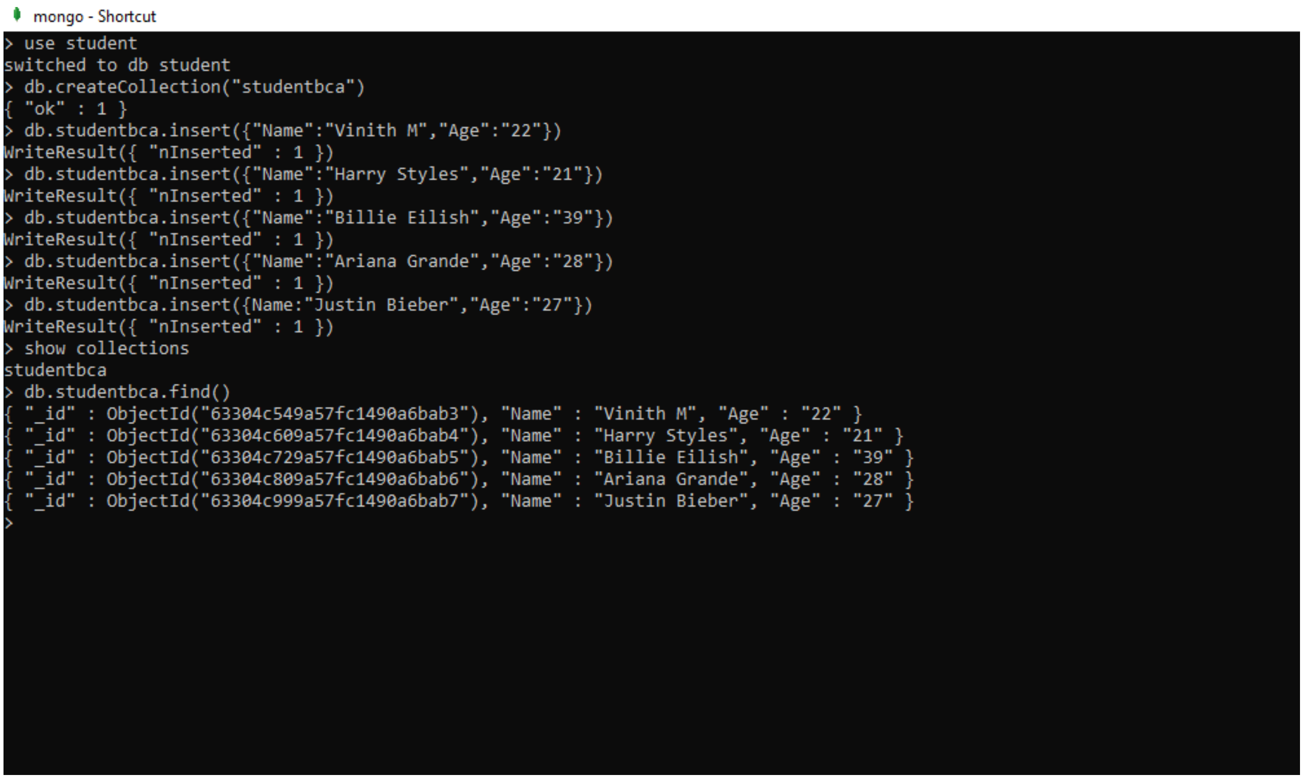
db.studentbca.insert({"Name":"Ariana Grande","Age":"28"})

db.studentbca.insert({Name:"Justin Bieber","Age":"27"})

<u>Step 4:</u> -show collections

<u>Step 5:</u>- db.studentbca.find()

OUTPUT



mongo - Shortcut

```
> use student
switched to db student
> db.createCollection("studentbca")
{ "ok" : 1 }
> db.studentbca.insert({"Name":"Vinith M","Age":"22"})
WriteResult({ "nInserted" : 1 })
> db.studentbca.insert({"Name":"Harry Styles","Age":"21"})
WriteResult({ "nInserted" : 1 })
> db.studentbca.insert({"Name":"Billie Eilish","Age":"39"})
WriteResult({ "nInserted" : 1 })
> db.studentbca.insert({"Name":"Ariana Grande","Age":"28"})
WriteResult({ "nInserted" : 1 })
> db.studentbca.insert({Name:"Justin Bieber","Age":"27"})
WriteResult({ "nInserted" : 1 })
> show collections
studentbca
> db.studentbca.find()
{ "_id" : ObjectId("63304c549a57fc1490a6bab3"), "Name" : "Vinith M", "Age" : "22" }
{ "_id" : ObjectId("63304c609a57fc1490a6bab4"), "Name" : "Harry Styles", "Age" : "21" }
{ "_id" : ObjectId("63304c729a57fc1490a6bab5"), "Name" : "Billie Eilish", "Age" : "39" }
{ "_id" : ObjectId("63304c809a57fc1490a6bab6"), "Name" : "Ariana Grande", "Age" : "28" }
{ "_id" : ObjectId("63304c999a57fc1490a6bab7"), "Name" : "Justin Bieber", "Age" : "27" }
>
```

3. Write a program to insert a document into MongoDB.

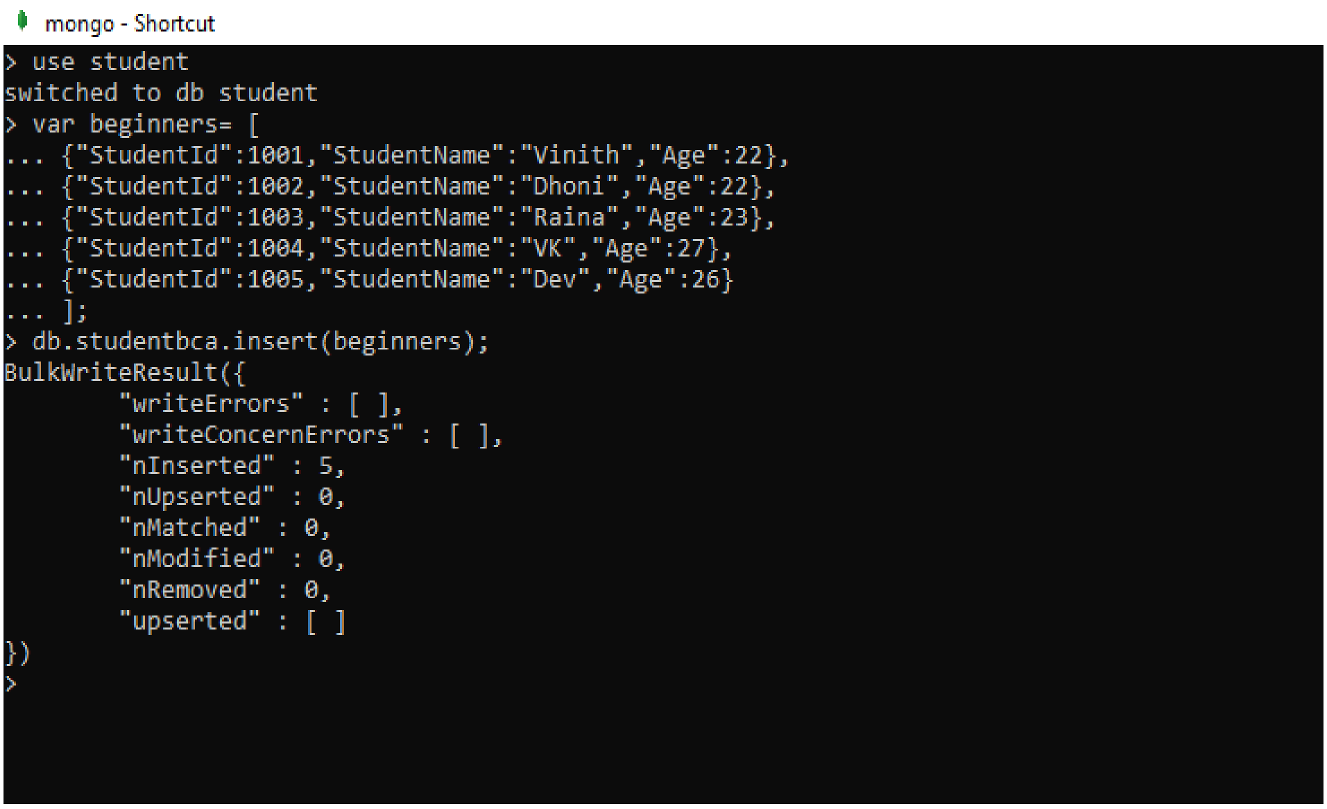
Step 1: - use student

Step 2:- var beginners= [

```
{"StudentId":1001,"StudentName":"Vinith","Age":22},
{"StudentId":1002,"StudentName":"Dhoni","Age":22},
{"StudentId":1003,"StudentName":"Raina","Age":23},
{"StudentId":1004,"StudentName":"VK","Age":27},
{"StudentId":1005,"StudentName":"Dev","Age":26}
];
```

Step 3:- db.studentbca.insert(beginners)

OUTPUT


mongo - Shortcut

```
> use student
switched to db student
> var beginners= [
... {"StudentId":1001,"StudentName":"Vinith","Age":22},
... {"StudentId":1002,"StudentName":"Dhoni","Age":22},
... {"StudentId":1003,"StudentName":"Raina","Age":23},
... {"StudentId":1004,"StudentName":"VK","Age":27},
... {"StudentId":1005,"StudentName":"Dev","Age":26}
... ];
> db.studentbca.insert(beginners);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

4. Write a program to list all the collections in a database.

Step 1: - use bcatest

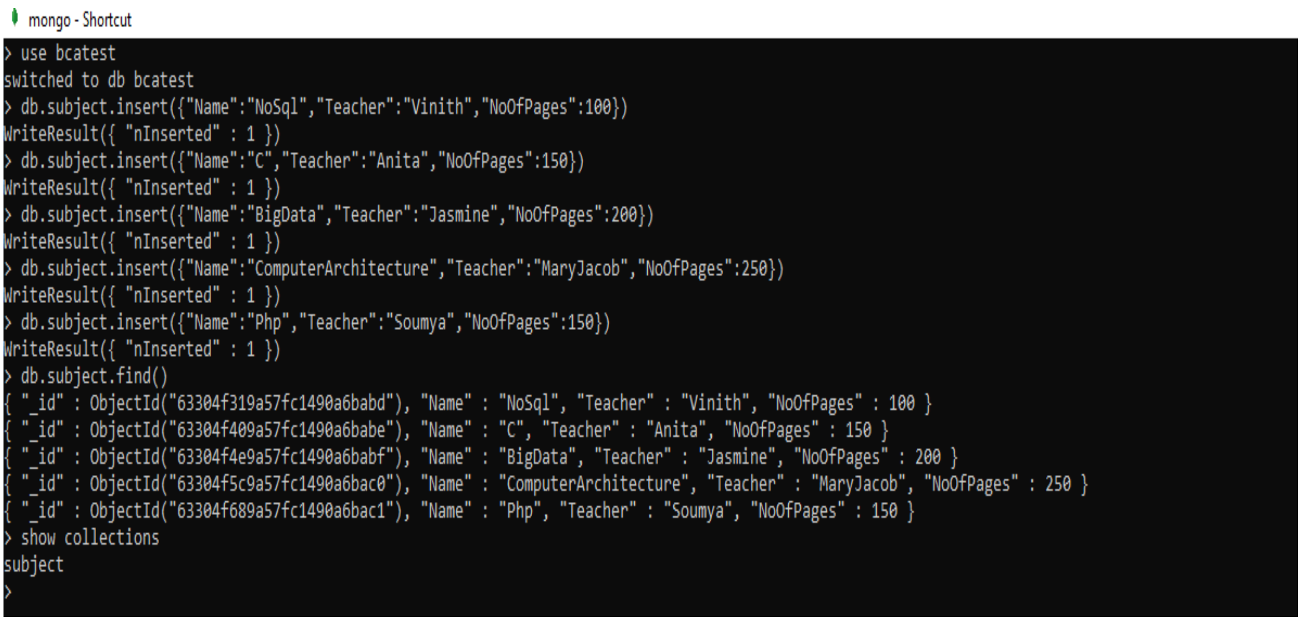
Step 2:-

```
db.subject.insert({"Name":"NoSql","Teacher":"Vinith","NoOfPages":100})
db.subject.insert({"Name":"C","Teacher":"Anita","NoOfPages":150})
db.subject.insert({"Name":"BigData","Teacher":"Jasmine","NoOfPages":20}
)
db.subject.insert({"Name":"ComputerArchitecture","Teacher":"Mary
Jacob","NoOfPages":250})
db.subject.insert({"Name":"Php","Teacher":"Soumya","NoOfPages":150})
```

Step 3:- db.subject.find()

Step 4: - show collections

OUTPUT



```
> use bcatest
switched to db bcatest
> db.subject.insert({"Name":"NoSql","Teacher":"Vinith","NoOfPages":100})
WriteResult({ "nInserted" : 1 })
> db.subject.insert({"Name":"C","Teacher":"Anita","NoOfPages":150})
WriteResult({ "nInserted" : 1 })
> db.subject.insert({"Name":"BigData","Teacher":"Jasmine","NoOfPages":200})
WriteResult({ "nInserted" : 1 })
> db.subject.insert({"Name":"ComputerArchitecture","Teacher":"MaryJacob","NoOfPages":250})
WriteResult({ "nInserted" : 1 })
> db.subject.insert({"Name":"Php","Teacher":"Soumya","NoOfPages":150})
WriteResult({ "nInserted" : 1 })
> db.subject.find()
{ "_id" : ObjectId("63304f319a57fc1490a6babd"), "Name" : "NoSql", "Teacher" : "Vinith", "NoOfPages" : 100 }
{ "_id" : ObjectId("63304f409a57fc1490a6babe"), "Name" : "C", "Teacher" : "Anita", "NoOfPages" : 150 }
{ "_id" : ObjectId("63304f4e9a57fc1490a6babf"), "Name" : "BigData", "Teacher" : "Jasmine", "NoOfPages" : 200 }
{ "_id" : ObjectId("63304f5c9a57fc1490a6bac0"), "Name" : "ComputerArchitecture", "Teacher" : "MaryJacob", "NoOfPages" : 250 }
{ "_id" : ObjectId("63304f689a57fc1490a6bac1"), "Name" : "Php", "Teacher" : "Soumya", "NoOfPages" : 150 }
> show collections
subject
>
```

5. **Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant.**

<u>Step 1:</u> - show dbs

<u>Step 2:</u>- use testDb

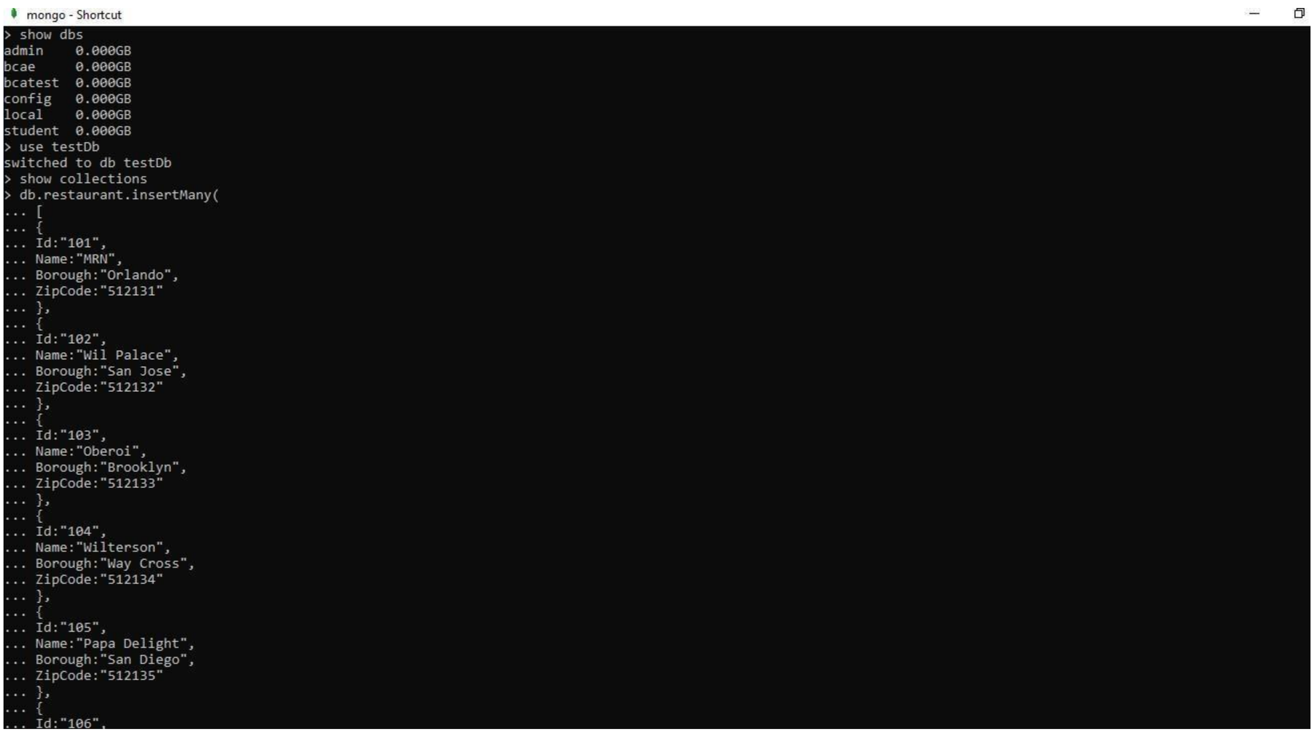
<u>Step 3:</u>- show collections()

<u>Step 4:</u>- db.restaurant.insertMany(

```
[
{
Id:"101",
Name:"MRN",
Borough:"Orlando",
ZipCode:"512131"
},
{
Id:"102",
Name:"Wil  Palace",
Borough:"San Jose",
ZipCode:"512132"
},
{
Id:"103",
Name:"Oberoi"
Borough:"Brooklyn",
ZipCode:"512133"
},
{
Id:"104",
Name:"Wilterson",
Borough:"Way Cross",
ZipCode:"512134
},
{
Id:"105",
Name:"Papa`s Delight",
Borough:"San Diego",
ZipCode:"512135"
},
{
Id:"106",
Name:"Willie's Cafe",
Borough:"Bay City",
ZipCode:"512136"
}
]
)
```

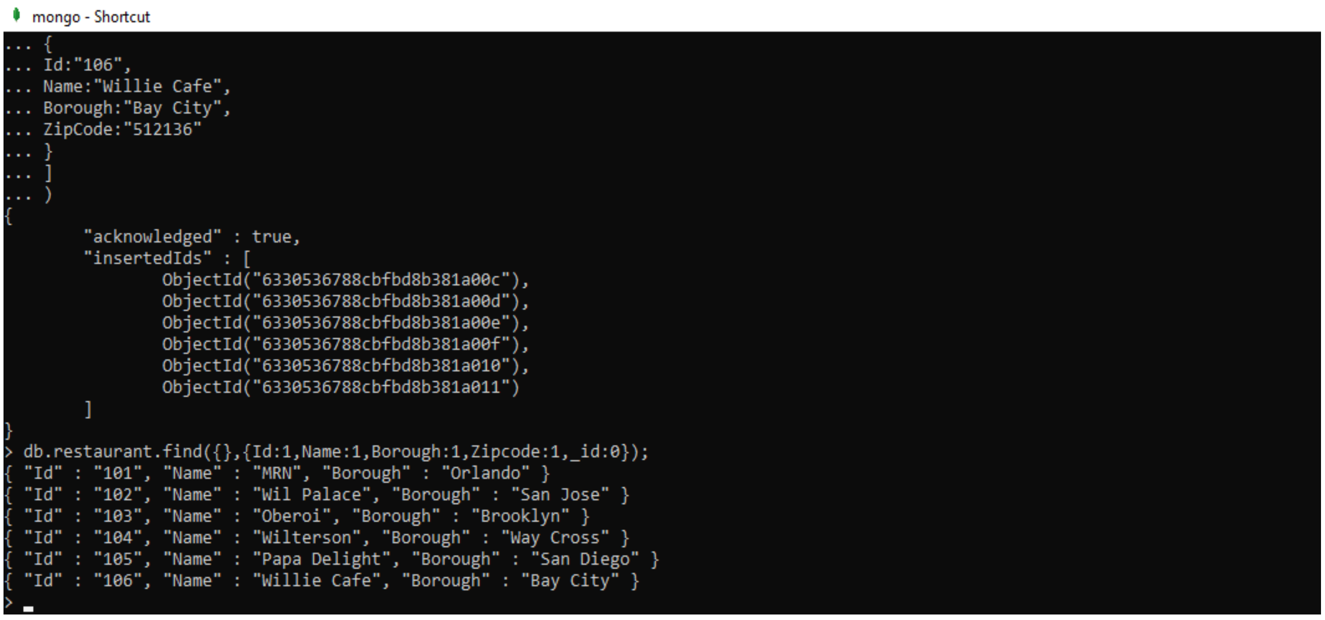
Step 5: - `db.restaurant.find({},{Id:1,Name:1,Borough:1,Zipcode:1,_id:0});`

OUTPUT



mongo - Shortcut

```
> show dbs
admin    0.000GB
bcae     0.000GB
bcatest  0.000GB
config   0.000GB
local    0.000GB
student  0.000GB
> use testDb
switched to db testDb
> show collections
> db.restaurant.insertMany(
... [
... {
... Id:"101",
... Name:"MRN",
... Borough:"Orlando",
... ZipCode:"512131"
... },
... {
... Id:"102",
... Name:"Wil Palace",
... Borough:"San Jose",
... ZipCode:"512132"
... },
... {
... Id:"103",
... Name:"Oberoi",
... Borough:"Brooklyn",
... ZipCode:"512133"
... },
... {
... Id:"104",
... Name:"Wilterson",
... Borough:"Way Cross",
... ZipCode:"512134"
... },
... {
... Id:"105",
... Name:"Papa Delight",
... Borough:"San Diego",
... ZipCode:"512135"
... },
... {
... Id:"106",
```



mongo - Shortcut

```
... {
... Id:"106",
... Name:"Willie Cafe",
... Borough:"Bay City",
... ZipCode:"512136"
... }
... ]
... )
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6330536788cbfbd8b381a00c"),
                ObjectId("6330536788cbfbd8b381a00d"),
                ObjectId("6330536788cbfbd8b381a00e"),
                ObjectId("6330536788cbfbd8b381a00f"),
                ObjectId("6330536788cbfbd8b381a010"),
                ObjectId("6330536788cbfbd8b381a011")
        ]
}
> db.restaurant.find({},{Id:1,Name:1,Borough:1,Zipcode:1,_id:0});
{ "Id" : "101", "Name" : "MRN", "Borough" : "Orlando" }
{ "Id" : "102", "Name" : "Wil Palace", "Borough" : "San Jose" }
{ "Id" : "103", "Name" : "Oberoi", "Borough" : "Brooklyn" }
{ "Id" : "104", "Name" : "Wilterson", "Borough" : "Way Cross" }
{ "Id" : "105", "Name" : "Papa Delight", "Borough" : "San Diego" }
{ "Id" : "106", "Name" : "Willie Cafe", "Borough" : "Bay City" }
> _
```

6. **Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.**

<u>Step 1:</u> - show dbs

<u>Step 2:</u>- use easedb

<u>Step 3:</u>- show collections

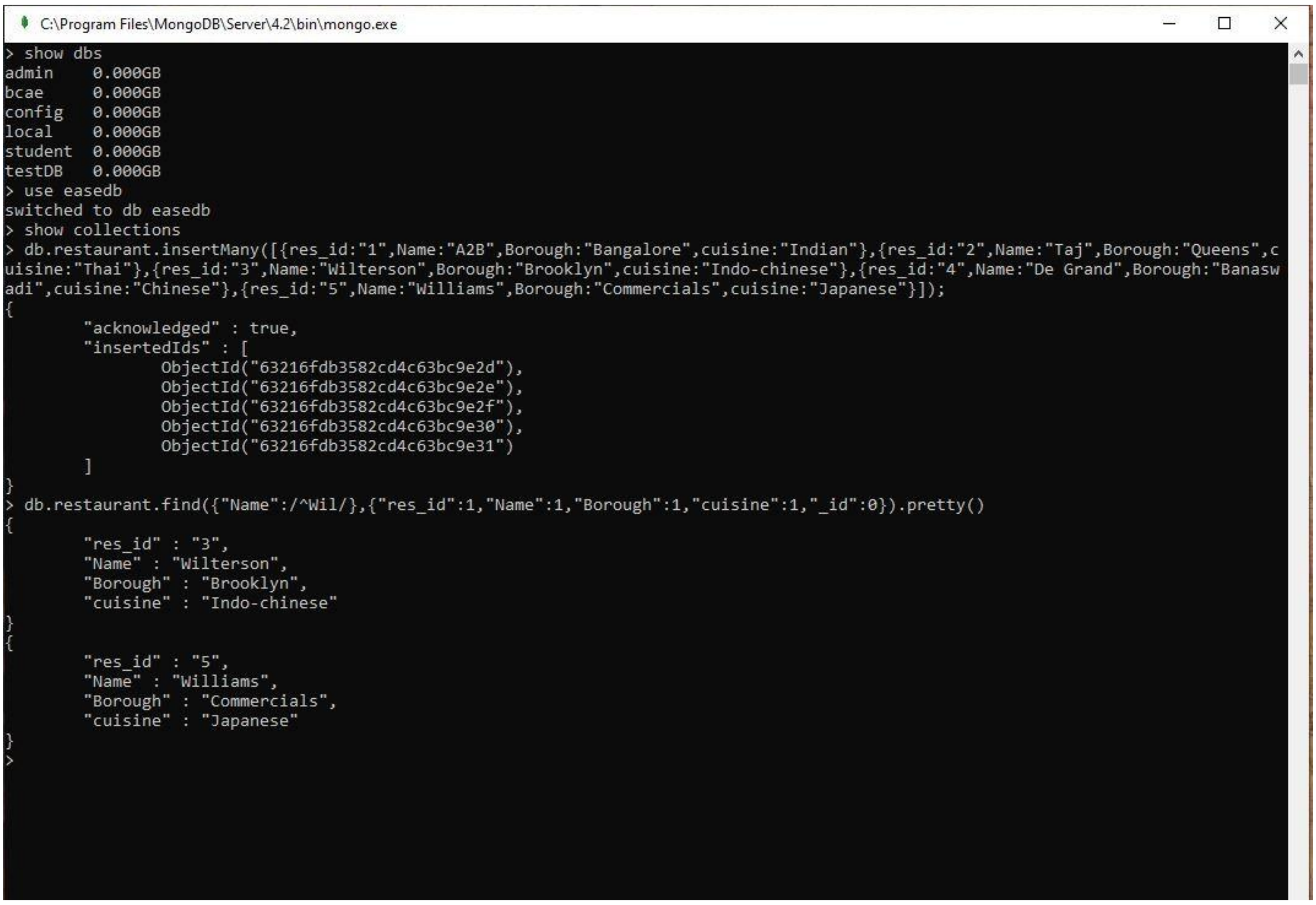
<u>Step 4</u>: - db.restaurant.insertMany(

```
[
{
res_id:"1",
Name:"bhai shop",
Borough:"Bangalore",
cuisine:"Indian"
},
{
res_id:"2",
Name:"Hotel Taj",
Borough:"Queens",
cuisine:"American"
},
{
res_id:"3",
Name:"Wilterson",
Borough:"Brooklyn",
cuisine:"Indo-Chinese"
},
{
res_id:"4",
Name:"De Grand",
Borough:"Banaswadi",
cuisine:"Chinese"
},
{
res_id:"5",
Name:"Williams",
Borough:"Commercial",
cuisine:"Japenese"
},
]
);
```

Step 5:-

```
db.restaurant.find({"name":/^Wil/},{"res_id":1,"name":1,"borough":1,"cuisine
":1,"_id":0}).pretty();
```

OUTPUT



C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe

```
> show dbs
admin    0.000GB
bcae     0.000GB
config   0.000GB
local    0.000GB
student  0.000GB
testDB   0.000GB
> use easedb
switched to db easedb
> show collections
> db.restaurant.insertMany([{res_id:"1",Name:"A2B",Borough:"Bangalore",cuisine:"Indian"},{res_id:"2",Name:"Taj",Borough:"Queens",c
uisine:"Thai"},{res_id:"3",Name:"Wilterson",Borough:"Brooklyn",cuisine:"Indo-chinese"},{res_id:"4",Name:"De Grand",Borough:"Banasw
adi",cuisine:"Chinese"},{res_id:"5",Name:"Williams",Borough:"Commercials",cuisine:"Japanese"}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("63216fdb3582cd4c63bc9e2d"),
                ObjectId("63216fdb3582cd4c63bc9e2e"),
                ObjectId("63216fdb3582cd4c63bc9e2f"),
                ObjectId("63216fdb3582cd4c63bc9e30"),
                ObjectId("63216fdb3582cd4c63bc9e31")
        ]
}
> db.restaurant.find({"Name":/^Wil/},{"res_id":1,"Name":1,"Borough":1,"cuisine":1,"_id":0}).pretty()
{
        "res_id" : "3",
        "Name" : "Wilterson",
        "Borough" : "Brooklyn",
        "cuisine" : "Indo-chinese"
}
{
        "res_id" : "5",
        "Name" : "Williams",
        "Borough" : "Commercials",
        "cuisine" : "Japanese"
}
>
```

7. **Write a program to create collections for student data and to display the student_Id in ascending order.**

Step 1: - show dbs

Step 2: - use easedb

Step 3:- show collections

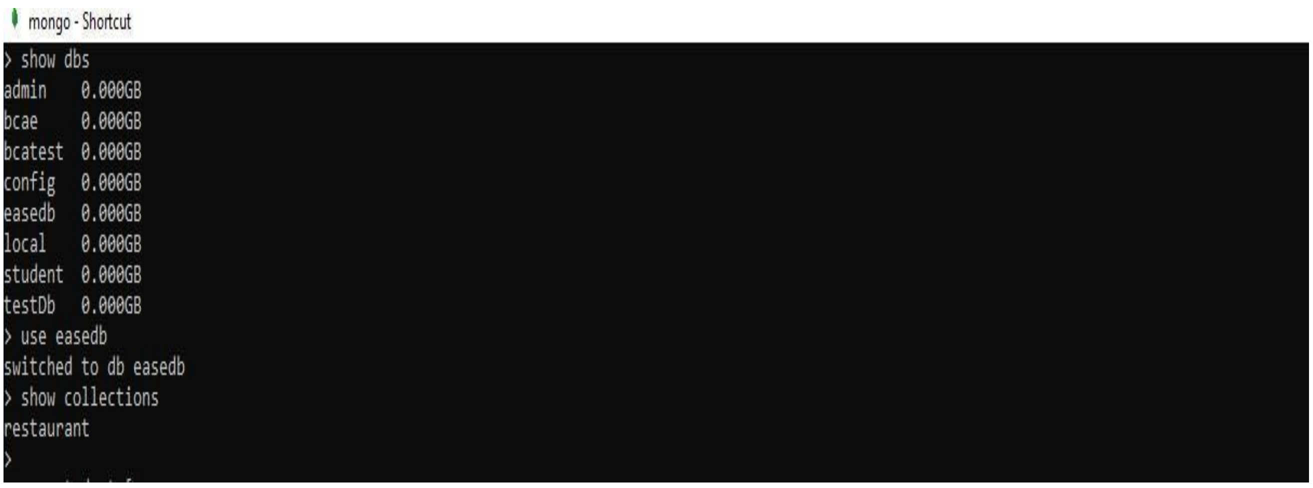
Step 4: - var student=[

```
{"regno":1,

"name":"Libin",

"course":{

"coursename":"Bed",

"duration":"2 years"

},

"address":{

"city":"Thodupuzha",

"state":"KL"

}

},

{"regno":4,

"name":"Sreejish",

"course":{

"coursename":"BCom",

"duration":"3 years"
```

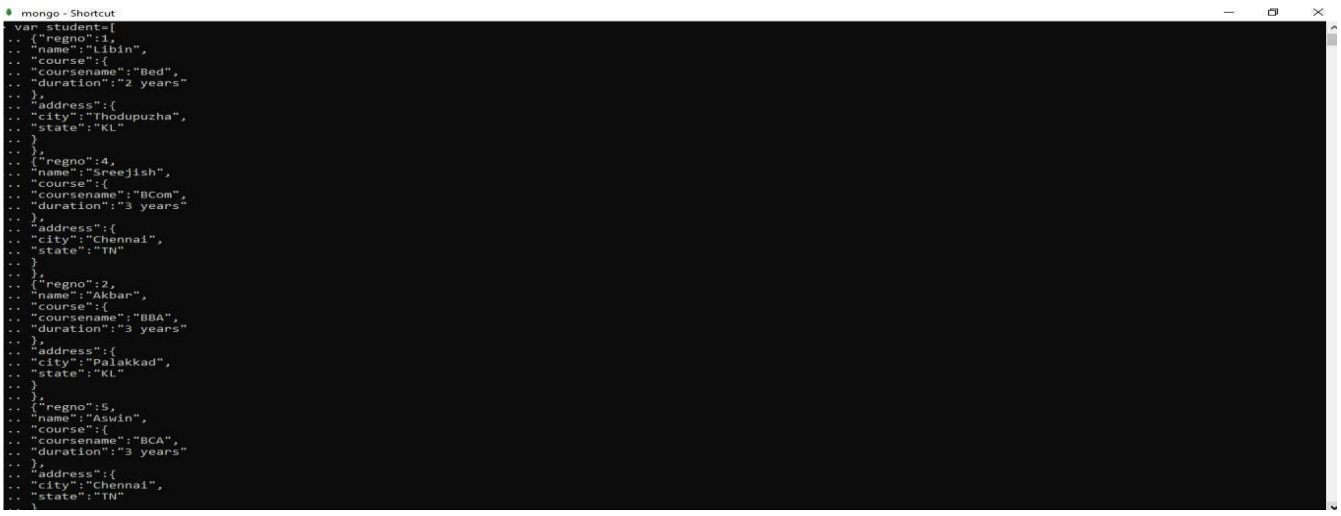
```
},
"address":{
"city":"Chennai",
"state":"TN"
}
},
{"regno":2,
"name":"Akbar",
"course":{
"coursename":"BBA",
"duration":"3 years"
},
"address":{
"city":"Palakkad",
"state":"KL"
}
},
{"regno":5,
"name":"Aswin",
"course":{
 "duration":"3 years"

},

"address":{

"city":"Chennai",

"state":"TN"

}

},

{"regno":3,

"name":"Bharath",

"course":{

"coursename":"BA",

"duration":"3 years"

},

"address":{

"city":"Bangalore",

"state":"KA"

}

}

]
```

Step 5: - db.studentdata.insert(studentdata)

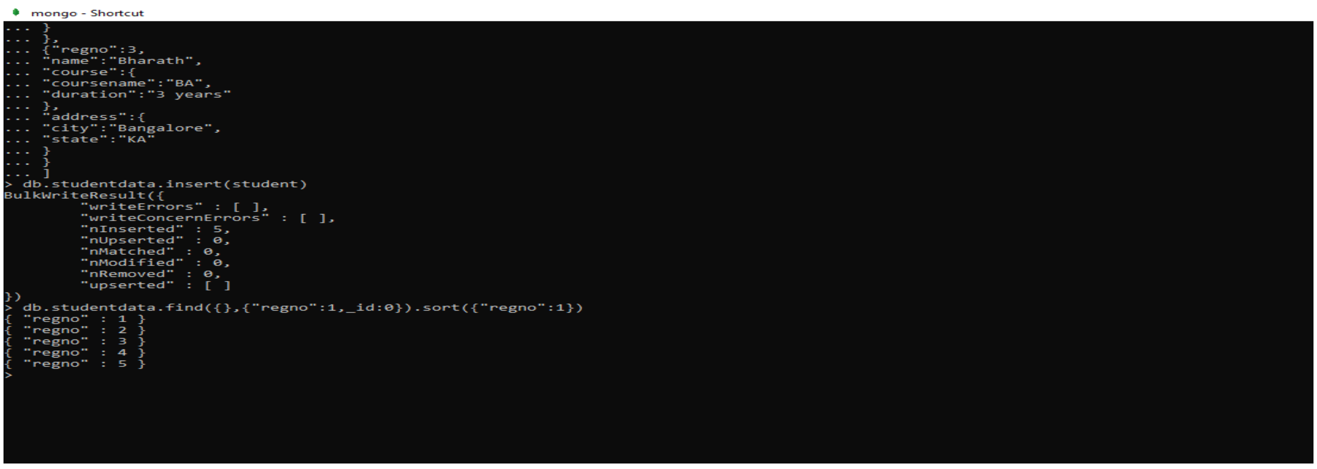
OUTPUT



```
> show dbs
admin    0.000GB
bcae     0.000GB
bcatest  0.000GB
config   0.000GB
easedb   0.000GB
local    0.000GB
student  0.000GB
testDb   0.000GB
> use easedb
switched to db easedb
> show collections
restaurant
>
```



```
 var student=[
.. {"regno":1,
.. "name":"Libin",
.. "course":{
.. "coursename":"Bed",
.. "duration":"2 years"
.. },
.. "address":{
.. "city":"Thodupuzha",
.. "state":"KL"
.. }
.. },
.. {"regno":4,
.. "name":"Sreejish",
.. "course":{
.. "coursename":"BCom",
.. "duration":"3 years"
.. },
.. "address":{
.. "city":"Chennai",
.. "state":"TN"
.. }
.. },
.. {"regno":2,
.. "name":"Akbar",
.. "course":{
.. "coursename":"BBA",
.. "duration":"3 years"
.. },
.. "address":{
.. "city":"Palakkad",
.. "state":"KL"
.. }
.. },
.. {"regno":5,
.. "name":"Aswin",
.. "course":{
.. "coursename":"BCA",
.. "duration":"3 years"
.. },
.. "address":{
.. "city":"Chennai",
.. "state":"TN"
```



```
... }
... },
... {"regno":3,
... "name":"Bharath",
... "course":{
... "coursename":"BA",
... "duration":"3 years"
... },
... "address":{
... "city":"Bangalore",
... "state":"KA"
... }
... }
... ]
> db.studentdata.insert(student)
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.studentdata.find({},{"regno":1,_id:0}).sort({"regno":1})
{ "regno" : 1 }
{ "regno" : 2 }
{ "regno" : 3 }
{ "regno" : 4 }
{ "regno" : 5 }
>
```

8. **Write a program to create collections for student data and to display the student_Name in descending order.**

Step 1: - show dbs

Step 2: - use easedb

Step 3:- show collections

Step 4: - var student=[

```
{"regno":1,

"name":"Libin",

"course":{

"coursename":"Bed",

"duration":"2 years"

},

"address":{

"city":"Thodupuzha",

"state":"KL"

}

},

{"regno":4,

"name":"Sreejish",

"course":{

"coursename":"BCom",

"duration":"3 years"
```

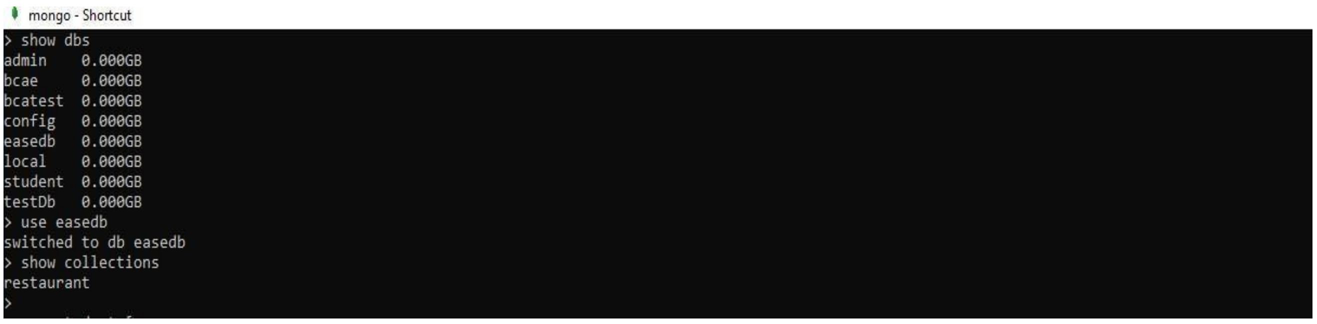
```
},
"address":{
"city":"Chennai",
"state":"TN"
}
},
{"regno":2,
"name":"Akbar",
"course":{
"coursename":"BBA",
"duration":"3 years"
},
"address":{
"city":"Palakkad",
"state":"KL"
}
},
{"regno":5,
"name":"Aswin",
"course":{
```

```
"coursename":"BCA",

"duration":"3 years"

},

"address":{

"city":"Chennai",

"state":"TN"

}

},

{"regno":3,

"name":"Bharath",

"course":{

"coursename":"BA",

"duration":"3 years"

},

"address":{

"city":"Bangalore",

"state":"KA"

}

}

]
```

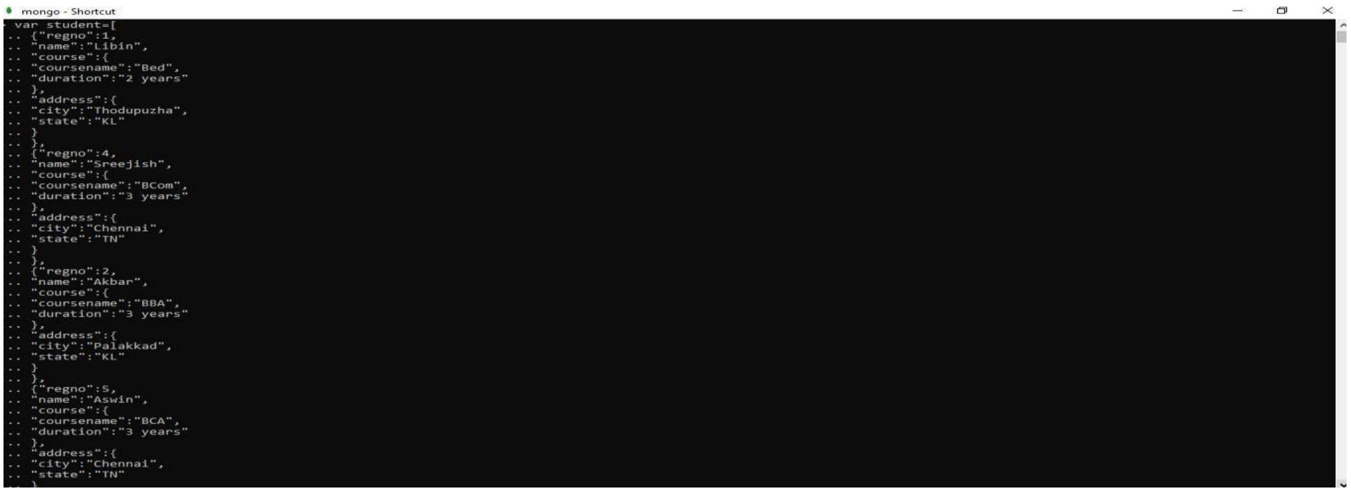
Step 5: - db.studentdata1.insert(studentdata1)

Step 6:- db.studentdata.find({},{"name":1,_id:0}).sort({"name":-1})

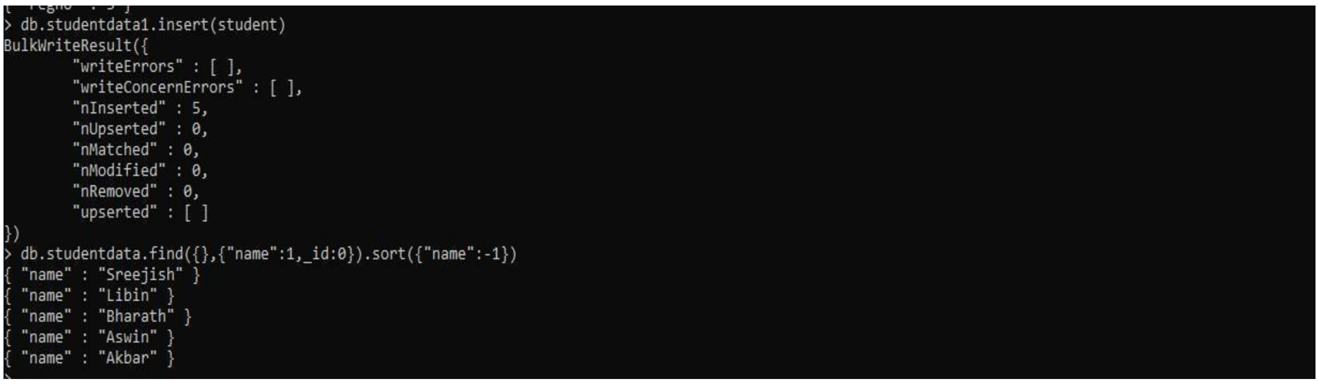
OUTPUT



```
> show dbs
admin    0.000GB
bcae     0.000GB
bcatest  0.000GB
config   0.000GB
easedb   0.000GB
local    0.000GB
student  0.000GB
testDb   0.000GB
> use easedb
switched to db easedb
> show collections
restaurant
>
```



```
> var student=[
.. {"regno":1,
.. "name":"Libin",
.. "course":{
.. "coursename":"Bed",
.. "duration":"2 years"
.. },
.. "address":{
.. "city":"Thodupuzha",
.. "state":"KL"
.. }
.. },
.. {"regno":4,
.. "name":"Sreejish",
.. "course":{
.. "coursename":"BCom",
.. "duration":"3 years"
.. },
.. "address":{
.. "city":"Chennai",
.. "state":"TN"
.. }
.. },
.. {"regno":2,
.. "name":"Akbar",
.. "course":{
.. "coursename":"BBA",
.. "duration":"3 years"
.. },
.. "address":{
.. "city":"Palakkad",
.. "state":"KL"
.. }
.. },
.. {"regno":5,
.. "name":"Aswin",
.. "course":{
.. "coursename":"BCA",
.. "duration":"3 years"
.. },
.. "address":{
.. "city":"Chennai",
.. "state":"TN"
```



```
> db.studentdata1.insert(student)
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.studentdata.find({},{"name":1,_id:0}).sort({"name":-1})
{ "name" : "Sreejish" }
{ "name" : "Libin" }
{ "name" : "Bharath" }
{ "name" : "Aswin" }
{ "name" : "Akbar" }
```

9. **Write a MongoDB query to find the student data that achieved a score, more than 80 but less than 100.**

Step 1: - use testDb

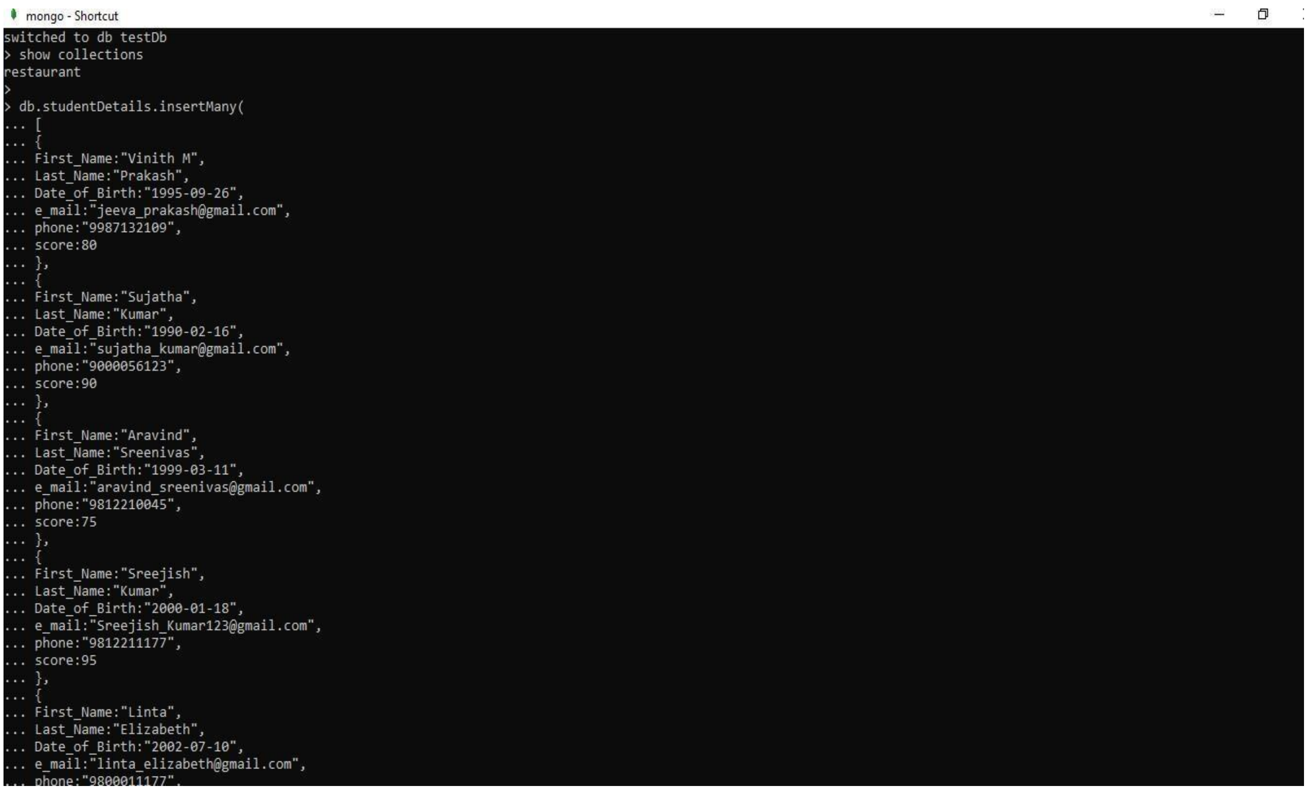
Step 2: - show collections

Step 3:- db.studentDetails.insertMany(

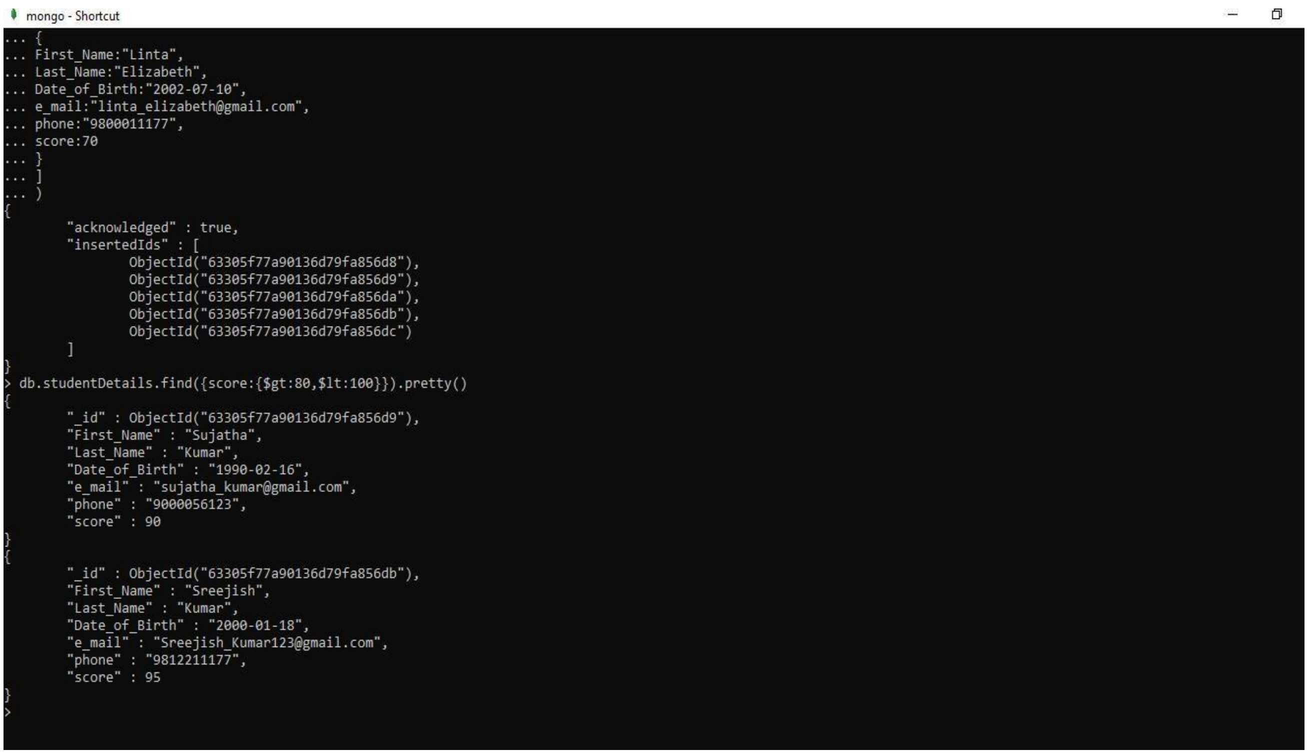
```
[
{
First_Name:"Jeeva",
Last_Name:"Prakash",
Date_of_Birth:"1995-09-26",
e_mail:"jeeva_prakash@gmail.com",
phone:"9987132109",
score:80
}
{
First_Name:"Sujatha",
Last_Name:"Kumar",
Date_of_Birth:"1990-02-16",
e_mail:"sujatha_kumar@gmail.com"
phone:"9000056123",
 score:90
 }
{
First_Name:"Aravind",
Last_Name:"Sreenivas",
Date_of_Birth:"1999-03-11",
e_mail:"aravind_sreenivas@gmail.com",
phone:"9812210045",
score:75
},
{
First_Name:"Sreejish",
Last_Name:"Kumar",
Date_of_Birth:"2000-01-18",
e_mail:"Sreejish_Kumar123@gmail.com",
phone:"9812211177",
score:95
},
{
First_Name:"Linta",
Last_Name:"Elizabeth",
Date_of_Birth:"2002-07-10",
e_mail:"linta_elizabeth@gmail.com",
phone:"9800011177",
score:70
}
]
)
```

Step 4: - db.studentDetails.find({score:{$gt:80,$lt:100}}).pretty()

OUTPUT



```
switched to db testDb
> show collections
restaurant
>
> db.studentDetails.insertMany(
... [
... {
... First_Name:"Vinith M",
... Last_Name:"Prakash",
... Date_of_Birth:"1995-09-26",
... e_mail:"jeeva_prakash@gmail.com",
... phone:"9987132109",
... score:80
... },
... {
... First_Name:"Sujatha",
... Last_Name:"Kumar",
... Date_of_Birth:"1990-02-16",
... e_mail:"sujatha_kumar@gmail.com",
... phone:"9000056123",
... score:90
... },
... {
... First_Name:"Aravind",
... Last_Name:"Sreenivas",
... Date_of_Birth:"1999-03-11",
... e_mail:"aravind_sreenivas@gmail.com",
... phone:"9812210045",
... score:75
... },
... {
... First_Name:"Sreejish",
... Last_Name:"Kumar",
... Date_of_Birth:"2000-01-18",
... e_mail:"Sreejish_Kumar123@gmail.com",
... phone:"9812211177",
... score:95
... },
... {
... First_Name:"Linta",
... Last_Name:"Elizabeth",
... Date_of_Birth:"2002-07-10",
... e_mail:"linta_elizabeth@gmail.com",
... phone:"9800011177",
```



```
... {
... First_Name:"Linta",
... Last_Name:"Elizabeth",
... Date_of_Birth:"2002-07-10",
... e_mail:"linta_elizabeth@gmail.com",
... phone:"9800011177",
... score:70
... }
... ]
... )
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("63305f77a90136d79fa856d8"),
                ObjectId("63305f77a90136d79fa856d9"),
                ObjectId("63305f77a90136d79fa856da"),
                ObjectId("63305f77a90136d79fa856db"),
                ObjectId("63305f77a90136d79fa856dc")
        ]
}
> db.studentDetails.find({score:{$gt:80,$lt:100}}).pretty()
{
        "_id" : ObjectId("63305f77a90136d79fa856d9"),
        "First_Name" : "Sujatha",
        "Last_Name" : "Kumar",
        "Date_of_Birth" : "1990-02-16",
        "e_mail" : "sujatha_kumar@gmail.com",
        "phone" : "9000056123",
        "score" : 90
}
{
        "_id" : ObjectId("63305f77a90136d79fa856db"),
        "First_Name" : "Sreejish",
        "Last_Name" : "Kumar",
        "Date_of_Birth" : "2000-01-18",
        "e_mail" : "Sreejish_Kumar123@gmail.com",
        "phone" : "9812211177",
        "score" : 95
}
>
```

10. Write a program to set up a replica set and to add members for the same.

Step 1: Let us consider 3 ports for replications task as 27017,27020,27021 in which 27017 as our primary servers.
Step 2: Create two folders named data1,data2 in C drive and 3 folders in each folder by name config,log and db.Then open config file and create a file by the name mongo.cfg and update the following info in the file.

```
dbpath: C:\data1\db\path
logpath: C:\data1\log\mongod.log\
port=27020
```

Step 3: Then copy all the 3 folders from data1 and paste to data2 and update mongo.cfg file in data2 as follows

```
dbpath: C:\data2\db\path
logpath: C:\data2\log\mongod.log\
port=27021
```

Step 4: Start stand alone serever as shown below mongod –dbpath"C:\programfiles\mongoDB\server\4.4\log\mongod.log" -port 27017 - -storageEngine=WiredTiger --journal --replset bcatest
Step 5: Connect the server with port no.27017

```
mongo - -port 27017
```

Step 6: Create variable rsconf

```
rsconf={_id:"bcatest",members:[{_id:0,host:"localhost:27017"}]}
rs.initiate(rsconf)
```

Step 7: Start secondary server on the port 27021

```
mongod --dbpath "C:\data1\db" --logpath "C:\data1\log\mongod.log" --
port 27020 --storageEngine=wiredTiger --journal --replSet bcatest
```

Step 8: Logon to secondary server

```
mongo --port 27020
```

Step 9: Start secondary server on the port 27021 mongod --dbpath "C:\data2\db" --logpath "C:\data2\log\mongod.log" --port 27021 --storageEngine=wiredTiger --journal --replSet bcatest
Step 10: Logon to secondary server

```
mongo --port 27071
```

Step 11: Run the following commands on primary server

```
rs.add("localhost:27020")
rs.add("localhost:27021")
```

Step 12: Now go to secondary servers and run below command on both the secondary servers.

```
rs.slaveOk()
```

Replication Setup verification:

Create a collection primary server and verify this reflects on secondary server or not.

Step 1: Connect to primary server

```
use bcae
```

Step 2: Create a collection in primary server

```
db.test.insert({name:”meer”})
```

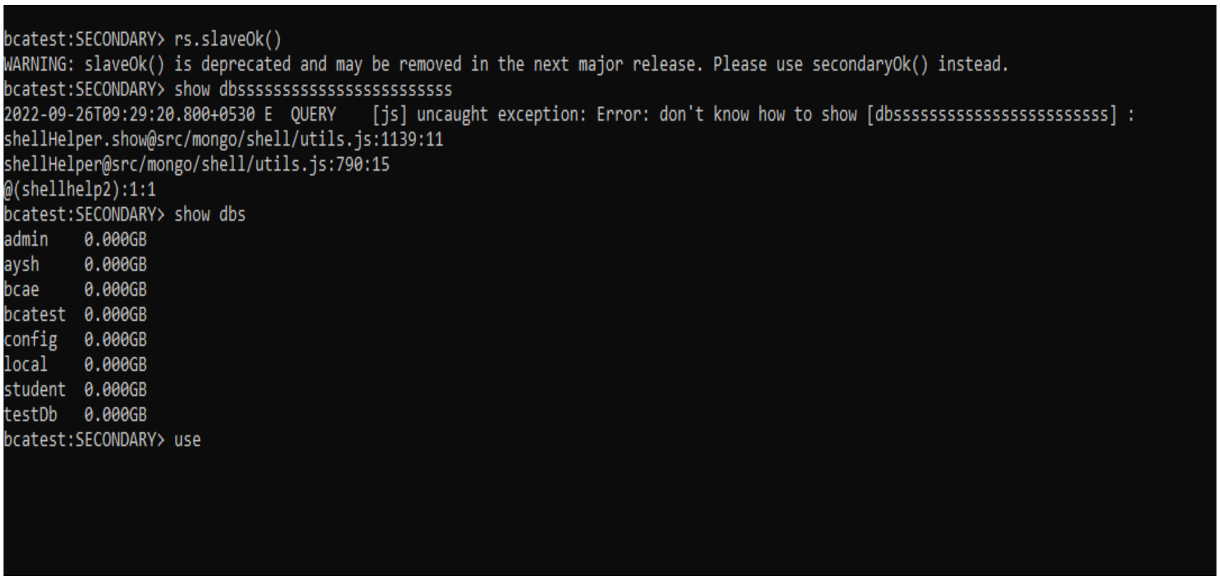
Step 3: Now connect to secondary server and check the list of database by running the following commands.

```
show dbs
use bcae
```

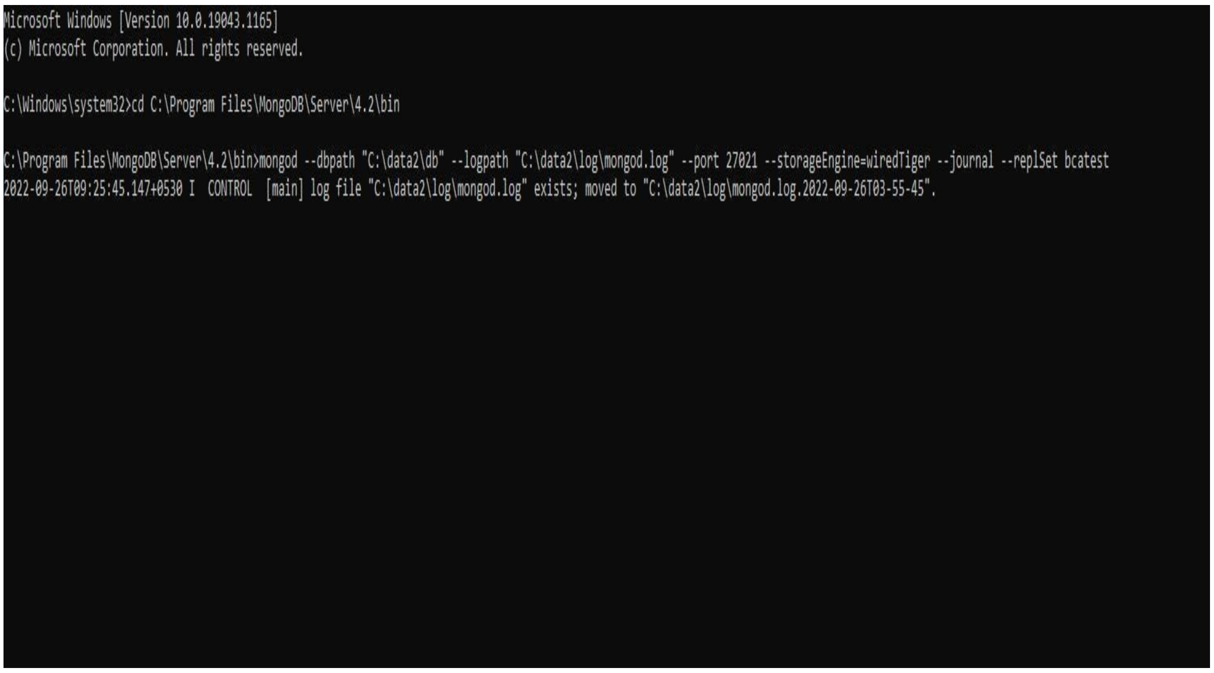
Run the command again in bcae

```
db.test.find().pretty()
```

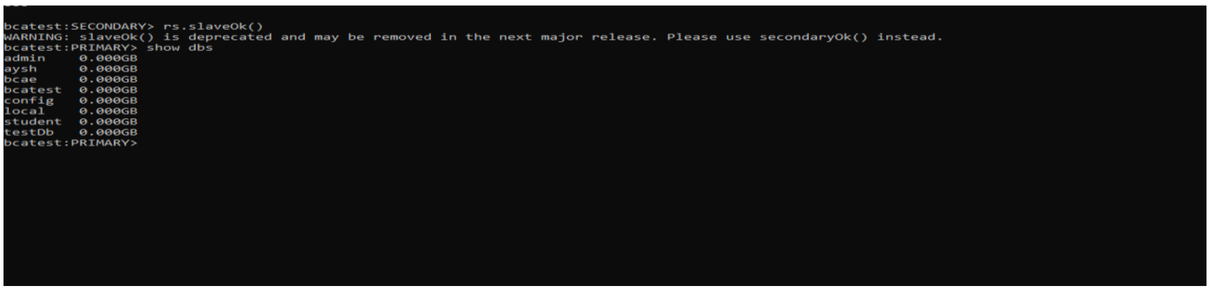
OUTPUT



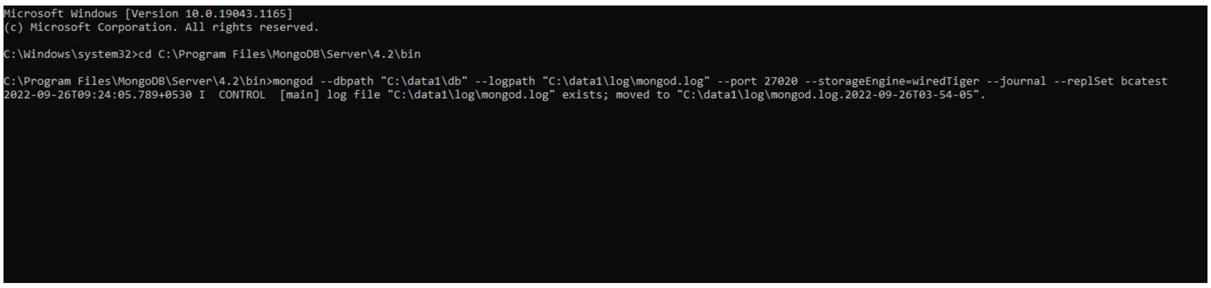
```
bcatest:SECONDARY> rs.slaveOk()
WARNING: slaveOk() is deprecated and may be removed in the next major release. Please use secondaryOk() instead.
bcatest:SECONDARY> show dbssssssssssssssssssssssss
2022-09-26T09:29:20.800+0530 E  QUERY    [js] uncaught exception: Error: don't know how to show [dbssssssssssssssssssssssss] :
shellHelper.show@src/mongo/shell/utils.js:1139:11
shellHelper@src/mongo/shell/utils.js:790:15
@(shellhelp2):1:1
bcatest:SECONDARY> show dbs
admin    0.000GB
aysh     0.000GB
bcae     0.000GB
bcatest  0.000GB
config   0.000GB
local    0.000GB
student  0.000GB
testDb   0.000GB
bcatest:SECONDARY> use
```



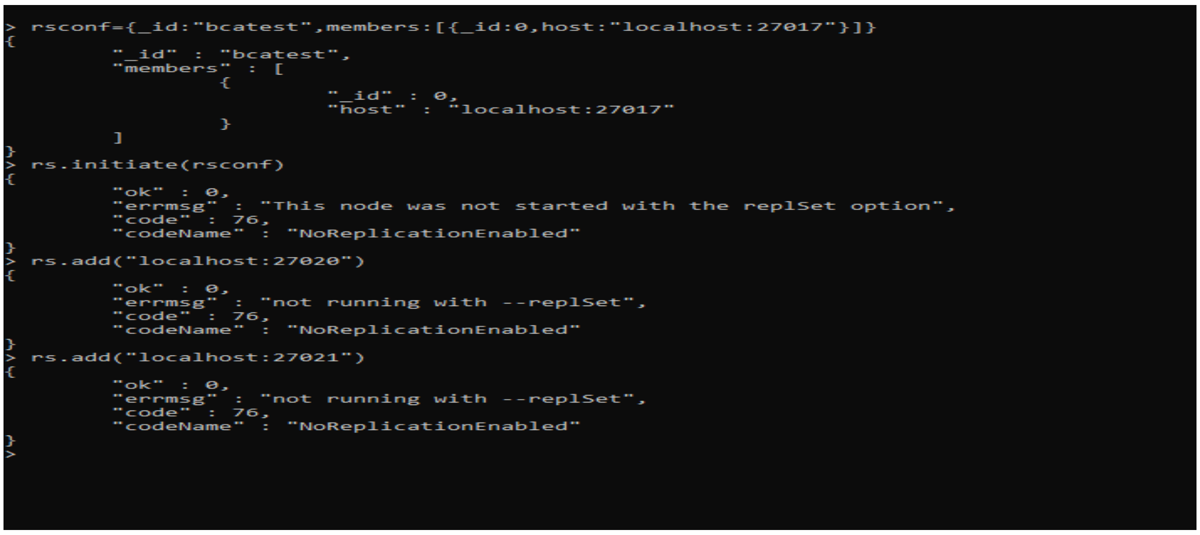
```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Program Files\MongoDB\Server\4.2\bin

C:\Program Files\MongoDB\Server\4.2\bin>mongod --dbpath "C:\data2\db" --logpath "C:\data2\log\mongod.log" --port 27021 --storageEngine=wiredTiger --journal --replSet bcatest
2022-09-26T09:25:45.147+0530 I  CONTROL  [main] log file "C:\data2\log\mongod.log" exists; moved to "C:\data2\log\mongod.log.2022-09-26T03-55-45".
```



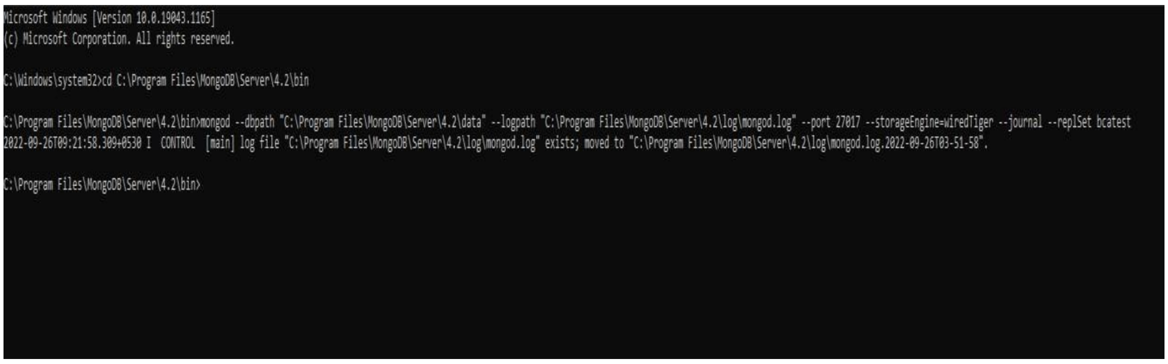
```
bcatest:SECONDARY> rs.slaveOk()
WARNING: slaveOk() is deprecated and may be removed in the next major release. Please use secondaryOk() instead.
bcatest:PRIMARY> show dbs
admin     0.000GB
aysh      0.000GB
bcae      0.000GB
bcatest   0.000GB
config    0.000GB
local     0.000GB
student   0.000GB
testDb    0.000GB
bcatest:PRIMARY>
```



```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Program Files\MongoDB\Server\4.2\bin

C:\Program Files\MongoDB\Server\4.2\bin>mongod --dbpath "C:\data1\db" --logpath "C:\data1\log\mongod.log" --port 27020 --storageEngine=wiredTiger --journal --replSet bcatest
2022-09-26T09:24:05.789+0530 I  CONTROL  [main] log file "C:\data1\log\mongod.log" exists; moved to "C:\data1\log\mongod.log.2022-09-26T03-54-05".
```



```
> rsconf={_id:"bcatest",members:[{_id:0,host:"localhost:27017"}]}
{
        "_id" : "bcatest",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:27017"
                }
        ]
}
> rs.initiate(rsconf)
{
        "ok" : 0,
        "errmsg" : "This node was not started with the replSet option",
        "code" : 76,
        "codeName" : "NoReplicationEnabled"
}
> rs.add("localhost:27020")
{
        "ok" : 0,
        "errmsg" : "not running with --replSet",
        "code" : 76,
        "codeName" : "NoReplicationEnabled"
}
> rs.add("localhost:27021")
{
        "ok" : 0,
        "errmsg" : "not running with --replSet",
        "code" : 76,
        "codeName" : "NoReplicationEnabled"
}
>
```



```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Program Files\MongoDB\Server\4.2\bin

C:\Program Files\MongoDB\Server\4.2\bin>mongod --dbpath "C:\Program Files\MongoDB\Server\4.2\data" --logpath "C:\Program Files\MongoDB\Server\4.2\log\mongod.log" --port 27017 --storageEngine=wiredTiger --journal --replSet bcatest
2022-09-26T09:21:58.309+0530 I  CONTROL  [main] log file "C:\Program Files\MongoDB\Server\4.2\log\mongod.log" exists; moved to "C:\Program Files\MongoDB\Server\4.2\log\mongod.log.2022-09-26T03-51-58".

C:\Program Files\MongoDB\Server\4.2\bin>
```

11. Write a program to establish a connection between PHP and MongoDB.

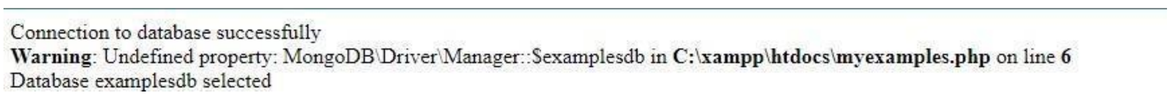
Step 1: Open Xampp server and start services.
Step 2: Install MongoDB php drivers.
Step 3: Change ini(config.file) in php folder with extension = php_mongodb.dll
Step 4: Write php code using mongoclient() to connect to database

```
<?php
 // connect to mongodb
 $m =new MongoDB\Driver\Manager("MongoDB:\\Localhost:27017");
 echo "Connection to database successfully";
 // select a database
 $db = $m->examplesdb;
 echo "Database examplesdb selected";
?>
```

When the program is executed, it will produce the following result −

```
Connection to database successfully
Database mydb selected
```

OUTPUT



```
Connection to database successfully
Warning: Undefined property: MongoDB\Driver\Manager::$examplesdb in C:\xampp\htdocs\myexamples.php on line 6
Database examplesdb selected
```

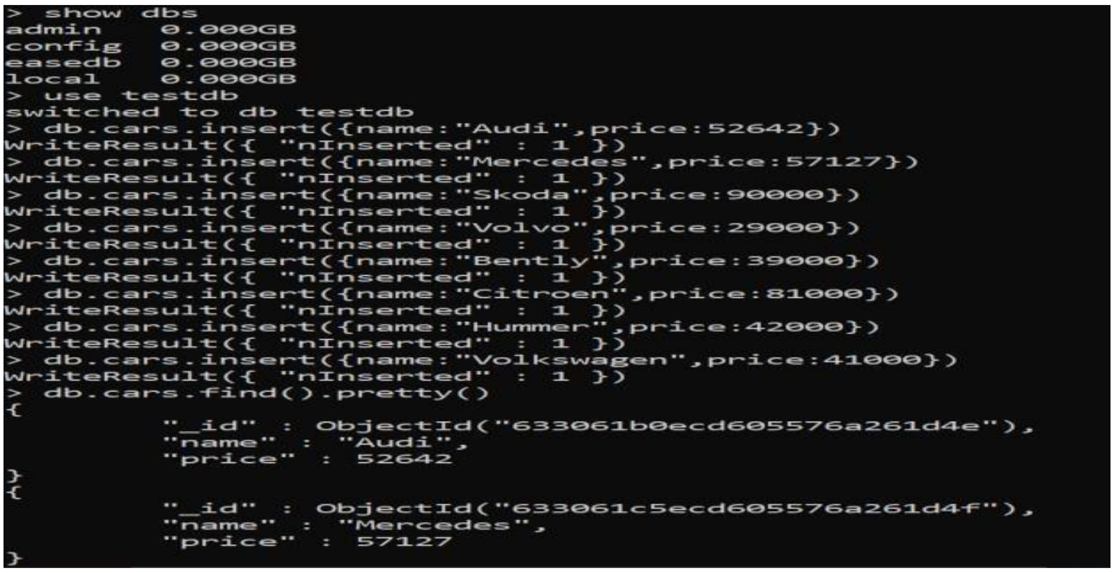
12. Write a program to create

Step 1: show dbs
Step 2: use testdb
Step 3:

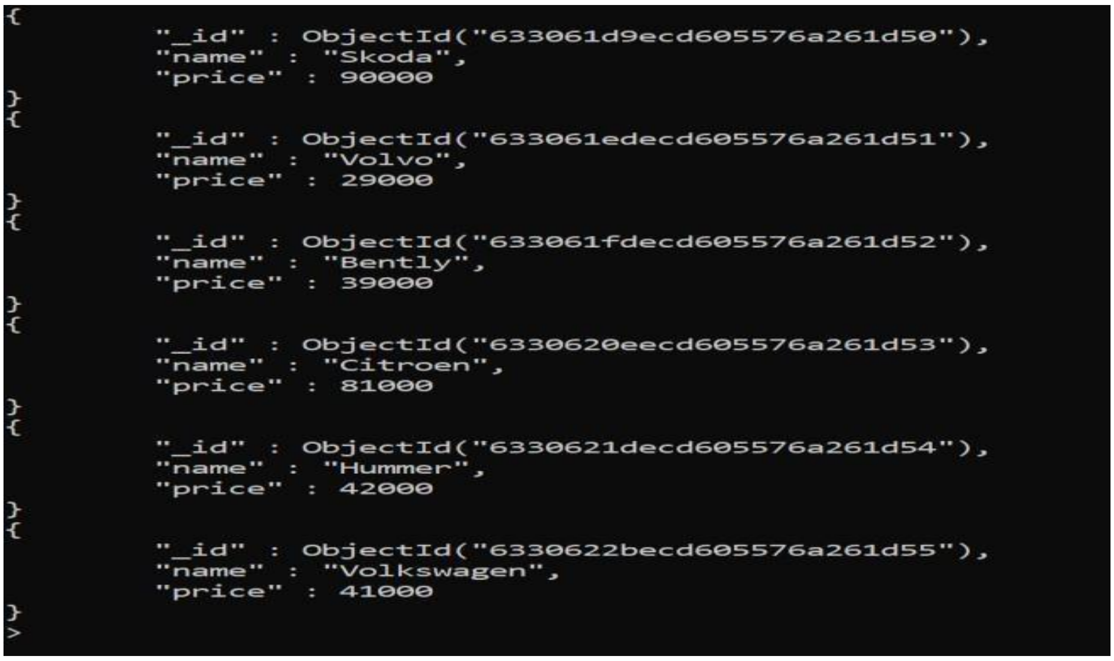
```
db.cars.insert({name:"Audi",price:52642})
db.cars.insert({name:"Mercedes",price:57127})
db.cars.insert({name:"Skoda",price:90000})
db.cars.insert({name:"Volvo",price:29000})
db.cars.insert({name:"Bently",price:39000})
db.cars.insert({name:"Citroen",price:81000})
db.cars.insert({name:"Hummer",price:42000})
db.cars.insert({name:"Volkswagen",price:41000})
```

Step 4: db.cars.find().pretty()

OUTPUT



```
> show dbs
admin    0.000GB
config   0.000GB
easedb   0.000GB
local    0.000GB
> use testdb
switched to db testdb
> db.cars.insert({name:"Audi",price:52642})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name:"Mercedes",price:57127})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name:"Skoda",price:90000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name:"Volvo",price:29000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name:"Bently",price:39000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name:"Citroen",price:81000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name:"Hummer",price:42000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name:"Volkswagen",price:41000})
WriteResult({ "nInserted" : 1 })
> db.cars.find().pretty()
{
        "_id" : ObjectId("633061b0ecd605576a261d4e"),
        "name" : "Audi",
        "price" : 52642
}
{
        "_id" : ObjectId("633061c5ecd605576a261d4f"),
        "name" : "Mercedes",
        "price" : 57127
}
```

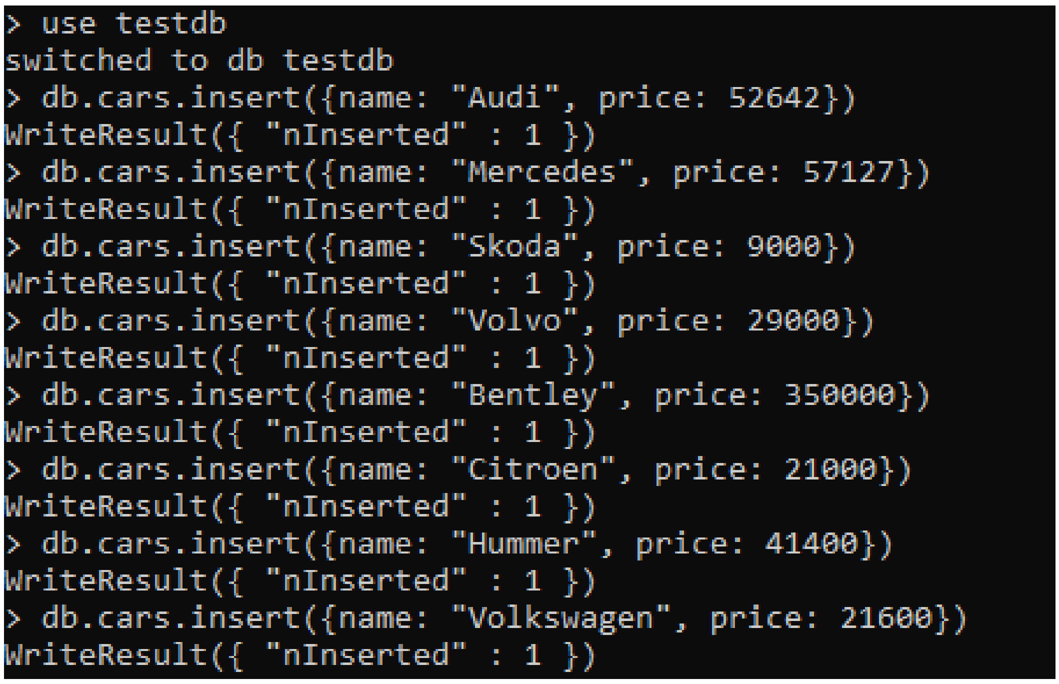


```
{
        "_id" : ObjectId("633061d9ecd605576a261d50"),
        "name" : "Skoda",
        "price" : 90000
}
{
        "_id" : ObjectId("633061edecd605576a261d51"),
        "name" : "Volvo",
        "price" : 29000
}
{
        "_id" : ObjectId("633061fdecd605576a261d52"),
        "name" : "Bently",
        "price" : 39000
}
{
        "_id" : ObjectId("6330620eecd605576a261d53"),
        "name" : "Citroen",
        "price" : 81000
}
{
        "_id" : ObjectId("6330621decd605576a261d54"),
        "name" : "Hummer",
        "price" : 42000
}
{
        "_id" : ObjectId("6330622becd605576a261d55"),
        "name" : "Volkswagen",
        "price" : 41000
}
>
```

13. Write a program for retrieving the data in MongoDB.

```
<?php
try {
$mng  =new   MongoDB\Driver\Manager("mongodb://localhost:27017");
$query =  new  MongoDB\Driver\Query([]);
$rows = $mng->executeQuery("testdb.cars", $query);
foreach ($rows as $row) {
        echo  "$row->name  : $row->price\n";
             }
} catch (MongoDB\Driver\Exception\Exception $e) {
    $filename = basename(__FILE__);
    echo "The $filename script has experienced an error.\n";
    echo "It failed with the following exception:\n";
    echo "Exception:", $e->getMessage(), "\n";
echo "In file:", $e->getFile(), "\n";
echo "On line:", $e->getLine(), "\n";
}
?>
```

OUTPUT

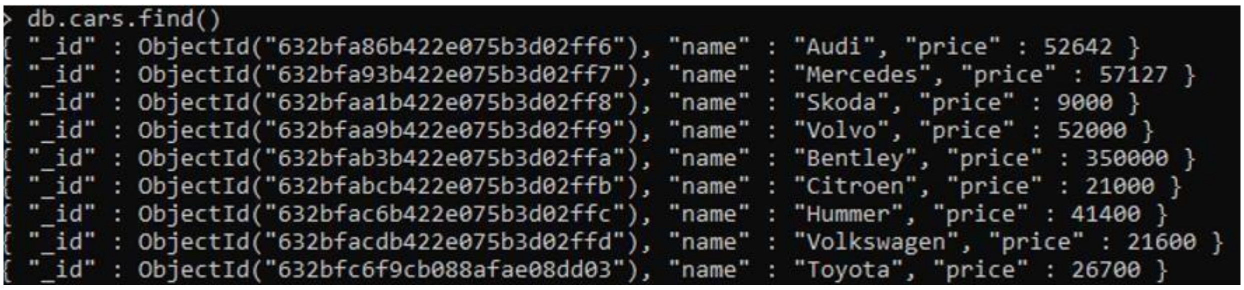


```
> use testdb
switched to db testdb
> db.cars.insert({name: "Audi", price: 52642})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name: "Mercedes", price: 57127})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name: "Skoda", price: 9000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name: "Volvo", price: 29000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name: "Bentley", price: 350000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name: "Citroen", price: 21000})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name: "Hummer", price: 41400})
WriteResult({ "nInserted" : 1 })
> db.cars.insert({name: "Volkswagen", price: 21600})
WriteResult({ "nInserted" : 1 })
```

14. Write a program for updating the data in MongoDB.

```
<?php
try {
   $mng = new MongoDB\Driver\Manager("mongodb://localhost:27017");
   $bulk = new MongoDB\Driver\BulkWrite;
   $doc = ['_id' => new MongoDB\BSON\ObjectID, 'name' => 'Toyota', 'price'
=> 26700];
   $bulk->insert($doc);
   $bulk->update(['name' => 'Volvo'], ['$set' => ['price' => 52000]]);
   $bulk->delete(['name' => 'Audi']);
   $mng->executeBulkWrite('testdb.cars', $bulk);
} catch (MongoDB\Driver\Exception\Exception $e) {
   $filename = basename(__FILE__);
   echo "The $filename script has experienced an error.\n";
   echo "It failed with the following exception:\n";
   echo "Exception:", $e->getMessage(), "\n";
   echo "In file:", $e->getFile(), "\n";
   echo "On line:", $e->getLine(), "\n";
}
?>
```

OUTPUT

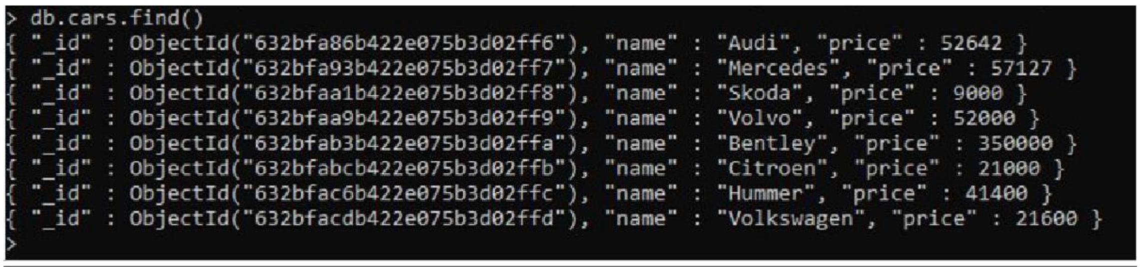


```
> db.cars.find()
{ "_id" : ObjectId("632bfa86b422e075b3d02ff6"), "name" : "Audi", "price" : 52642 }
{ "_id" : ObjectId("632bfa93b422e075b3d02ff7"), "name" : "Mercedes", "price" : 57127 }
{ "_id" : ObjectId("632bfaa1b422e075b3d02ff8"), "name" : "Skoda", "price" : 9000 }
{ "_id" : ObjectId("632bfaa9b422e075b3d02ff9"), "name" : "Volvo", "price" : 52000 }
{ "_id" : ObjectId("632bfab3b422e075b3d02ffa"), "name" : "Bentley", "price" : 350000 }
{ "_id" : ObjectId("632bfabcb422e075b3d02ffb"), "name" : "Citroen", "price" : 21000 }
{ "_id" : ObjectId("632bfac6b422e075b3d02ffc"), "name" : "Hummer", "price" : 41400 }
{ "_id" : ObjectId("632bfacdb422e075b3d02ffd"), "name" : "Volkswagen", "price" : 21600 }
{ "_id" : ObjectId("632bfc6f9cb088afae08dd03"), "name" : "Toyota", "price" : 26700 }
```

15. Write a program for deleting the data in MongoDB.

```
<?php
try {
    $mng = new MongoDB\Driver\Manager("mongodb://localhost:27017");
    $bulk = new MongoDB\Driver\BulkWrite;
    $bulk->delete(['name' => 'Toyota']);
    $mng->executeBulkWrite('testdb.cars', $bulk);
} catch (MongoDB\Driver\Exception\Exception $e) {
    $filename = basename(__FILE__);
    echo "The $filename script has experienced an error.\n";
    echo "It failed with the following exception:\n";
    echo "Exception:", $e->getMessage(), "\n";
    echo "In file:", $e->getFile(), "\n";
    echo "On line:", $e->getLine(), "\n";
}
?>
```

OUTPUT



```
> db.cars.find()
{ "_id" : ObjectId("632bfa86b422e075b3d02ff6"), "name" : "Audi", "price" : 52642 }
{ "_id" : ObjectId("632bfa93b422e075b3d02ff7"), "name" : "Mercedes", "price" : 57127 }
{ "_id" : ObjectId("632bfaa1b422e075b3d02ff8"), "name" : "Skoda", "price" : 9000 }
{ "_id" : ObjectId("632bfaa9b422e075b3d02ff9"), "name" : "Volvo", "price" : 52000 }
{ "_id" : ObjectId("632bfab3b422e075b3d02ffa"), "name" : "Bentley", "price" : 350000 }
{ "_id" : ObjectId("632bfabcb422e075b3d02ffb"), "name" : "Citroen", "price" : 21000 }
{ "_id" : ObjectId("632bfac6b422e075b3d02ffc"), "name" : "Hummer", "price" : 41400 }
{ "_id" : ObjectId("632bfacdb422e075b3d02ffd"), "name" : "Volkswagen", "price" : 21600 }
>
```