1. Connecting to Server –

c:\mongodb\bin\mongod.exe –dbpath c:\mongodb\data\db

E:\mongodb\bin>mongod.exe --dbpath e:\mongodb\data\db
Fri Aug 23 03:51:35.606
Fri Aug 23 03:51:35.606 warning: 32-bit servers don't have journaling enabled by default. Please use --journal if you want durability.
Fri Aug 23 03:51:35.606
Fri Aug 23 03:51:35.606 [initandlisten] MongoDB starting : pid=4260 port=27017 dbpath=e:\mongodb\data\db 32-bit host=409PC-22
Fri Aug 23 03:51:35.606 [initandlisten]
Fri Aug 23 03:51:35.606 [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
Fri Aug 23 03:51:35.606 [initandlisten] ** 32 bit builds are limited to less than 2GB of data (or less with --journal).
Fri Aug 23 03:51:35.606 [initandlisten] ** Note that journaling defaults to off for 32 bit and is currently off.
Fri Aug 23 03:51:35.606 [initandlisten] ** See http://dochub.mongodb.org/core/32bit
Fri Aug 23 03:51:35.606 [initandlisten]
Fri Aug 23 03:51:35.606 [initandlisten] db version v2.4.9
Fri Aug 23 03:51:35.622 [initandlisten] git version: 52fe0d21959e32a5bdbecdc62057db386e4e029c
Fri Aug 23 03:51:35.622 [initandlisten] build info: windows sys.getwindowsversion(major=6, minor=0, build=6002, platform=2, service_pack='Service Pack 2') BOOST_LIB_VERSION=1_49
Fri Aug 23 03:51:35.622 [initandlisten] allocator: system
Fri Aug 23 03:51:35.622 [initandlisten] options: { dbpath: "e:\mongodb\data\db" }
Fri Aug 23 03:51:35.669 [FileAllocator] allocating new datafile e:\mongodb\data\db\local.ns, filling with zeroes...
Fri Aug 23 03:51:35.669 [FileAllocator] creating directory e:\mongodb\data\db_tmp
Fri Aug 23 03:51:35.700 [FileAllocator] done allocating datafile e:\mongodb\data\db\local.ns, size: 16MB, took 0.03 secs
Fri Aug 23 03:51:35.700 [FileAllocator] allocating new datafile e:\mongodb\data\db\local.0, filling with zeroes...
Fri Aug 23 03:51:35.731 [FileAllocator] done allocating datafile e:\mongodb\data\db\local.0, size: 16MB, took 0.03 secs
Fri Aug 23 03:51:35.731 [initandlisten] waiting for connections on port 27017
Fri Aug 23 03:51:35.731 [websvr] admin web console waiting for connections on port 28017
Fri Aug 23 03:52:43.993 [initandlisten] connection accepted from 127.0.0.1:50781 #1 (1 connection now open)

2. Connecting to Client –

c:\mongodb\bin\mongo.exe

3. Create database –

> **use dbshc**

switched to db dbshc

4. Confirm the existence of your database

> **db**
dbshc

5. To get a list of all databases

>**show dbs**;
local 0.03125GB

6. To drop a database

```
>use mydb;
>db.dropDatabase();
{ "dropped" : "dbshc", "ok" : 1 }
```

7. To display the list of collections

```
> Create collection
> db.createCollection("myCollection")
>show collections
myCollection
system.indexes
```

8. To display the current version of MongoDB Server

```
>db.version()
2.4.9
```

9. To display the statistics that reflect the use state of the database

```
>db.stats()
{
        "db" : "dbshc",
        "collections" : 3,
        "objects" : 4,
        "avgObjSize" : 59,
        "dataSize" : 236,
        "storageSize" : 16384,
        "numExtents" : 3,
        "indexes" : 1,
        "indexSize" : 8176,
        "fileSize" : 50331648,
        "nsSizeMB" : 16,
        "dataFileVersion" : {
                "major" : 4,
                "minor" : 5
        },
        "ok" : 1
}
```

10. To display the list of commands

```
>db.help()
DB methods:
        db.addUser(userDocument)
        db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [
just calls db.runCommand(...) ]
        db.auth(username, password)
        db.cloneDatabase(fromhost)
        db.commandHelp(name) returns the help for the command
        db.copyDatabase(fromdb, todb, fromhost)
        db.createCollection(name, { size : ..., capped : ..., max : ... } )
        db.currentOp() displays currently executing operations in the db
        db.dropDatabase()
        db.eval(func, args) run code server-side
        db.fsyncLock() flush data to disk and lock server for backups
```

db.fsyncUnlock() unlocks server following a db.fsyncLock()
db.getCollection(cname) same as db['cname'] or db.cname
db.getCollectionNames()
db.getLastError() - just returns the err msg string
db.getLastErrorObj() - return full status object
db.getMongo() get the server connection object
db.getMongo().setSlaveOk() allow queries on a replication slave server
db.getName()
db.getPrevError()
db.getProfilingLevel() - deprecated
db.getProfilingStatus() - returns if profiling is on and slow threshold
db.getReplicationInfo()
db.getSiblingDB(name) get the db at the same server as this one
db.hostInfo() get details about the server's host
db.isMaster() check replica primary status
db.killOp(opid) kills the current operation in the db
db.listCommands() lists all the db commands
db.loadServerScripts() loads all the scripts in db.system.js
db.logout()
db.printCollectionStats()
db.printReplicationInfo()
db.printShardingStatus()
db.printSlaveReplicationInfo()
db.removeUser(username)
db.repairDatabase()
db.resetError()
db.runCommand(cmdObj) run a database command. if cmdObj is a string, turns it into { cmdObj : 1 }
db.serverStatus()
db.setProfilingLevel(level,<slowms>) 0=off 1=slow 2=all
db.setVerboseShell(flag) display extra information in shell output
db.shutdownServer()
db.stats()
db.version() current version of the server

11. Insert –

```
> db.student.insert ({rollno:'1',name:'ani',contact:'09876543578',city:'mumbai'});
```

12. To check if the document is successfully inserted. Use find() method

```
> db.student.find();
{ "_id" : ObjectId("5d5fc763a92067ac8d573313"), "rollno" : "1", "name" : "ani", "contact" : "09876543578", "city" : "mumbai" }
```

13. To check if the document is successfully inserted and displayed in an organized manner.

```
> db.student.find().pretty();
{
        "_id" : ObjectId("5d5fc763a92067ac8d573313"),
        "rollno" : "1",
        "name" : "ani",
        "contact" : "09876543578",
        "city" : "mumbai"
}
```

14. Multiple Insert –

```
> db.post.insert({“title”:”MongoDB Overview”, “likes”:“20”}, {“title”:“NoSQL Database”, “likes”:”20”})
> db.post.find().pretty()
{
	"_id" : ObjectId("5d5fc7b4a92067ac8d573314"),
	"title" : "MongoDB Overview",
	"likes" : "20"
}
```

15. Add multiple records

```
>db.student.insert({_id:'1',studname:'anu',grade:'VII',school:'kv'});
> db.student.insert({_id:'2',studname:'amu',grade:'VII',school:'ssn'});
> db.student.insert({_id:'3',studname:'chinu',grade:'VII',school:'fatima'});
>db.student.insert({_id:'4',studname:'shruti',grade:'VIII',school:'smhs'});
> db.student.insert({_id:'5',studname:'ABC',grade:'V',school:'svis'});
> db.student.insert({_id:'6',studname:'DEF',grade:'VI',school:'pphs'});
```

16. Update –

```
> db.student.update({_id:'1',studname:'anu'},{$set:{hobbies:'skating'}},{upsert:true});
>db.student.find().pretty()
{
	"_id" : ObjectId("5d5fc763a92067ac8d573313"),
	"rollno" : "1",
	"name" : "ani",
	"contact" : "09876543578",
	"city" : "mumbai"
}
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{
	"_id" : "1",
	"grade" : "VII",
	"hobbies" : "skating",
	"school" : "kv",
	"studname" : "anu"
}
{ "_id" : "3", "grade" : "VII", "school" : "fatima", "studname" : "chinu" }
{ "_id" : "5", "studname" : "ABC", "grade" : "V", "school" : "svis" }
{ "_id" : "6", "studname" : "DEF", "grade" : "VI", "school" : "pphs" }
{
	"_id" : "4",
	"grade" : "VIII",
	"hobbies" : "drawing",
	"school" : "smhs",
	"studname" : "shruti"
}
```

17. Save –

```
> db.student.save()
Fri Aug 23 04:04:30.321 can't save a null at src/mongo/shell/collection.js:250
```

18. Limit – this limits the display of output

```
> db.student.find().limit(2)
{ "_id" : ObjectId("5d5fc763a92067ac8d573313"), "rollno" : "1", "name" : "ani", "contact" : "09876543578", "city" : "mumbai" }
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
```

19. Skip – This skips the first n records.

```
> db.student..find().skip(2)
{ "_id" : "1", "grade" : "VII", "hobbies" : "skating", "school" : "kv", "studname" : "anu" }
{ "_id" : "3", "grade" : "VII", "school" : "fatima", "studname" : "chinu" }
{ "_id" : "5", "studname" : "ABC", "grade" : "V", "school" : "svis" }
{ "_id" : "6", "studname" : "DEF", "grade" : "VI", "school" : "pphs" }
{ "_id" : "4", "grade" : "VIII", "hobbies" : "drawing", "school" : "smhs", "studname" : "shruti" }
```

20. Update to add an attribute

```
> db.student.update({_id:'4'},{$set:{location:'borivali'}});
{
        "_id" : ObjectId("5d5fc763a92067ac8d573313"),
        "rollno" : "1",
        "name" : "ani",
        "contact" : "09876543578",
        "city" : "mumbai"
}
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{
        "_id" : "1",
        "grade" : "VII",
        "hobbies" : "skating",
        "school" : "kv",
        "studname" : "anu"
}
{ "_id" : "3", "grade" : "VII", "school" : "fatima", "studname" : "chinu" }
{ "_id" : "5", "studname" : "ABC", "grade" : "V", "school" : "svis" }
{ "_id" : "6", "studname" : "DEF", "grade" : "VI", "school" : "pphs" }
{
        "_id" : "4",
        "grade" : "VIII",
        "hobbies" : "drawing",
        "location" : "borivali",
        "school" : "smhs",
        "studname" : "shruti"
}
```

21. To remove an attribute use unset

```
> db.student.update({_id:'4'},{$unset:{location:'borivali'}});
                        {
        "_id" : ObjectId("5d5fc763a92067ac8d573313"),
        "rollno" : "1",
        "name" : "ani",
        "contact" : "09876543578",
        "city" : "mumbai"
}
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{
```

```
        "_id" : "1",
        "grade" : "VII",
        "hobbies" : "skating",
        "school" : "kv",
        "studname" : "anu"
}
{ "_id" : "3", "grade" : "VII", "school" : "fatima", "studname" : "chinu" }
{ "_id" : "5", "studname" : "ABC", "grade" : "V", "school" : "svis" }
{ "_id" : "6", "studname" : "DEF", "grade" : "VI", "school" : "pphs" }
{
        "_id" : "4",
        "grade" : "VIII",
        "hobbies" : "drawing",
        "school" : "smhs",
        "studname" : "shruti"
}
```

22. Finding documents based on search criteria-find method

```
> db.student.find({grade:'VII'});
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{ "_id" : "1", "grade" : "VII", "hobbies" : "skating", "school" : "kv", "studname" :
"anu" }
{ "_id" : "3", "grade" : "VII", "school" : "fatima", "studname" : "chinu" }
```

22. Finding projection based on selection operators

```
> db.student.find({_id:'4'},{studname:1});
```

23. { "_id" : "4", "studname" : "shruti" }

24. Finding records with same matching criteria. (Equivalent to "=" clause)

```
> db.student.find({grade:{$eq:'VII'}}).pretty();
```

(record your output here; Check out for other relational operators like
$eq→equal to
$ne→not equal to
$gte→greater than or equal to
$lte→less than or equal to
$gt→ greater than
$lt→lesser than)

```
> db.student.find({grade:{$eq:'VII'}}).pretty();
error: { "$err" : "invalid operator: $eq", "code" : 10068 }
> db.student.find({grade:{$ne:'VII'}}).pretty();
{
        "_id" : ObjectId("5d5fc763a92067ac8d573313"),
        "rollno" : "1",
        "name" : "ani",
        "contact" : "09876543578",
        "city" : "mumbai"
}
> db.student.find({grade:{$gte:'VII'}}).pretty();
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{
        "_id" : "1",
        "grade" : "VII",
        "hobbies" : "skating",
```

```
        "school" : "kv",
        "studname" : "anu"
}
{ "_id" : "3", "grade" : "VII", "school" : "fatima", "studname" : "chinu" }
> db.student.find({grade:{$lte:'VII'}}).pretty();
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{
        "_id" : "1",
        "grade" : "VII",
        "hobbies" : "skating",
        "school" : "kv",
        "studname" : "anu"
}
{ "_id" : "3", "grade" : "VII", "school" : "fatima", "studname" : "chinu" }
> db.student.find({grade:{$gt:'VII'}}).pretty();
> db.student.find({grade:{$lt:'VII'}}).pretty();
```

25. Finding records based on the ‘IN’ operator. Similar to SQL

```
> db.student.find({school:{$in:['kv','ssn']}}).pretty();
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{
        "_id" : "1",
        "grade" : "VII",
        "hobbies" : "skating",
        "school" : "kv",
        "studname" : "anu"
}
```

26. Finding records based on the AND Clause

```
> db.student.find({$and:[{school:'kv'},{ grade:'VII'}]}).pretty()
{
        "_id" : "1",
        "grade" : "VII",
        "hobbies" : "skating",
        "school" : "kv",
        "studname" : "anu"
}
```

27. Finding records based on the OR clause

```
> db.student.find({$and:[{school:'smhs'},{ grade:'VIII'}]}).pretty()
{
        "_id" : "4",
        "grade" : "VIII",
        "hobbies" : "drawing",
        "school" : "smhsa",
        "studname" : "shruti"
}
```

28. Finding records based on matching patterns

```
> db.student.find({studname:/^a/}).pretty();(All students whose name starts with ‘a’)
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{
        "_id" : "1",
```

```
        "grade" : "VII",
        "hobbies" : "skating",
        "school" : "kv",
        "studname" : "anu"
}
> db.student.find({studname:/u$/}).pretty();(All students whose name ends with'u')
{ "_id" : "2", "studname" : "amu", "grade" : "VII", "school" : "ssn" }
{
        "_id" : "1",
        "grade" : "VII",
        "hobbies" : "skating",
        "school" : "kv",
        "studname" : "anu"
}
{ "_id" : "3", "grade" : "VII", "school" : "fatima", "studname" : "chinu" }
```

29. To create arrays in a collection –

> db.food.insert({_id:1,fruits:['banana','apple','cherry']});
> db.food.insert({_id:2,fruits:['orange','butterfruit','mango']});
> db.food.insert({_id:3,fruits:['pineapple','strawberry','grapes']});
> db.food.insert({_id:4,fruits:['banana','strawberry','grapes']});

Items are inserted into the food table

```
> db.food.find().pretty();
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{ "_id" : 2, "fruits" : [ "orange", "butterfruit", "mango" ] }
{ "_id" : 3, "fruits" : [ "pineapple", "strawberry", "grapes" ] }
{ "_id" : 4, "fruits" : [ "banana", "strawberry", "grapes" ] }
```

30. To find certain elements in the array –

> db.food.find({fruits:['banana','apple','cherry']}).pretty()

```
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
```

31. To find elements with certain array positions

> **db.food.find({'fruits.2':'grapes'});** Find the elements having grapes in their array in position 2)

```
{ "_id" : 3, "fruits" : [  "pineapple",  "strawberry",  "grapes" ] }
{ "_id" : 4, "fruits" : [  "banana",  "strawberry",  "grapes" ] }
```

32. To display documents having only two array elements –

> db.food.find({'fruits':{$size:2}});

33. To display selected array elements from a specific document

> db.food.find({_id:1},{"fruits":{$slice:2}});
{ "_id" : 1, "fruits" : ["banana", "apple"] }

34. To display all documents which have the same array elements

```
> db.food.find({fruits:{$all:["banana","apple"]}});
```

```
{ "_id" : 1, "fruits" : [  "banana",  "apple",  "cherry" ] }
```

35. To update a specific document and include a new attribute for it

```
> db.food.update({_id:2},{$push:{price:{orange:60,butterfruit:200,mango:120}}});

> db.food.find().pretty();
{ "_id" : 4, "fruits" : [ "banana", "strawberry", "grapes" ] }
{
        "_id" : 2,
        "fruits" : [
                "orange",
                "butterfruit",
                "mango"
        ],
        "price" : [
                {
                        "orange" : 60,
                        "butterfruit" : 200,
                        "mango" : 120
                }
        ]
}

> db.food.update({_id:4},{$push:{quantity:{banana:50,strawberry:120,grapes:90}}});
> db.food.find({_id:4}).pretty();
{
        "_id" : 4,
        "fruits" : [
                "banana",
                "strawberry",
                "grapes"
        ],
        "quantity" : [
                {
                        "banana" : 50,
                        "strawberry" : 120,
                        "grapes" : 90
                }
        ]
}
```

36. To update a specific document by adding elements to the array

```
> db.food.update({_id:4},{$addToSet:{fruits:"orange"}});
> db.food.update({_id:4},{$addToSet:{fruits:"orange"}});
> db.food.update({_id:3},{$addToSet:{fruits:"banana"}});
> db.food.update({_id:3},{$addToSet:{fruits:"mango"}});
> db.food.find({_id:4}).pretty();
{
        "_id" : 4,
        "fruits" : [
                "banana",
                "strawberry",
                "grapes",
                "orange"
        ],
        "quantity" : [
```

```
            {
                "banana" : 50,
                "strawberry" : 120,
                "grapes" : 90
            }
        ]
}
> db.food.find({_id:3}).pretty();
{
        "_id" : 3,
        "fruits" : [
                "pineapple",
                "strawberry",
                "grapes",
                "banana",
                "mango"
        ]
}
```

37. To update a specific document by removing elements to the array

```
> db.food.update({_id:4},{$pop:{fruits:1}});
> db.food.find({_id:4}).pretty();
{
        "_id" : 4,
        "fruits" : [
                "banana",
                "strawberry",
                "grapes"
        ],
        "quantity" : [
                {
                        "banana" : 50,
                        "strawberry" : 120,
                        "grapes" : 90
                }
        ]
}

> db.food.update({_id:4},{$pop:{fruits:-1}});
> db.food.find({_id:4}).pretty();
{
        "_id" : 4,
        "fruits" : [
                "strawberry",
                "grapes"
        ],
        "quantity" : [
                {
                        "banana" : 50,
                        "strawberry" : 120,
                        "grapes" : 90
                }
        ]
}
```

38. To remove an element from the entire set of documents

```
> db.food.update({fruits:'mango'},{$pull:{fruits:'mango'}});
> db.food.find().pretty();
{ "_id" : 1, "fruits" : [ "banana", "apple", "cherry" ] }
{
        "_id" : 2,
        "fruits" : [
                "orange",
                "butterfruit"
        ],
        "price" : [
                {
                        "orange" : 60,
                        "butterfruit" : 200,
                        "mango" : 120
                }
        ]
}
{
        "_id" : 4,
        "fruits" : [
                "strawberry",
                "grapes"
        ],
        "quantity" : [
                {
                        "banana" : 50,
                        "strawberry" : 120,
                        "grapes" : 90
                }
        ]
}
{
        "_id" : 3,
        "fruits" : [
                "pineapple",
                "strawberry",
                "grapes",
                "banana",
                "mango"
        ]
}
```

39. Aggregate functions. Using map reduce

Create a customer Collection with the following values

```
{
        "_id" : 1,
        "cust_id" : "abc1",
        "ord_date" : ISODate("2012-11-02T17:04:11.102Z"),
        "status" : "A",
        "amount" : 50
}
{
        "_id" : 2,
        "cust_id" : "xyz1",
```

```
        "ord_date" : ISODate("2013-10-01T17:04:11.102Z"),
        "status" : "A",
        "amount" : 100
}
{
        "_id" : 3,
        "cust_id" : "xyz1",
        "ord_date" : ISODate("2013-10-12T17:04:11.102Z"),
        "status" : "D",
        "amount" : 25
}
{
        "_id" : 4,
        "cust_id" : "xyz1",
        "ord_date" : ISODate("2013-10-11T17:04:11.102Z"),
        "status" : "D",
        "amount" : 125
}
{
        "_id" : 5,
        "cust_id" : "abc1",
        "ord_date" : ISODate("2013-11-12T17:04:11.102Z"),
        "status" : "A",
        "amount" : 25
}
```

Create a customer collections and insert 5 documents as given above.
Confirm that the 5 documents exist in the collection customer

```
> db.customer.find().pretty()
{
        "_id" : 1,
        "cust_id" : "abc1",
        "ord_date" : ISODate("2012-11-02T17:04:11.102Z"),
        "status" : "A",
        "amount" : 50
}
{
        "_id" : 2,
        "cust_id" : "xyz1",
        "ord_date" : ISODate("2013-10-01T17:04:11.102Z"),
        "status" : "A",
        "amount" : 100
}
{
        "_id" : 3,
        "cust_id" : "xyz1",
        "ord_date" : ISODate("2013-10-12T17:04:11.102Z"),
        "status" : "D",
        "amount" : 25
}
{
        "_id" : 4,
        "cust_id" : "xyz1",
        "ord_date" : ISODate("2013-10-11T17:04:11.102Z"),
        "status" : "D",
```

```
        "amount" : 125
    }
    {
        "_id" : 5,
        "cust_id" : "abc1",
        "ord_date" : ISODate("2013-11-12T17:04:11.102Z"),
        "status" : "A",
        "amount" : 25
    }
```

40. Creation of map function

> **var map=function(){ emit(this.cust_id,this.amount);}**

41. Creation of reduce function

var reduce=function(key,values){return Array.sum(values);}

42. To execute the query.

> db.customer.mapReduce(map,reduce,{out:"Customer_totals",query:{status:"A"}});

```
{
    "result" : "Customer_totals",
    "timeMillis" : 21,
    "counts" : {
        "input" : 3,
        "emit" : 3,
        "reduce" : 1,
        "output" : 2
    },
    "ok" : 1,
}
> db.customer.mapReduce(map,reduce,{out:"Customer_tot",query:{cust_id:"abc1"}});
{
    "result" : "Customer_tot",
    "timeMillis" : 7,
    "counts" : {
        "input" : 2,
        "emit" : 2,
        "reduce" : 1,
        "output" : 1
    },
    "ok" : 1,
}
```

43. To display the output

db.Customer_totals.find().pretty();

{ "_id" : "abc1", "value" : 75 }
{ "_id" : "xyz1", "value" : 100 }

> db.Customer_tot.find().pretty();
{ "_id" : "abc1", "value" : 75 }

44. Java Script Programming

To create a function the user is required to create a function and insert into the “system.js” Collection.

To check on what is contained in “system.js” collection use

>**db.system.js.find();**

--The output was not obtained as initially no function was present in the system.

45. Creation of a function factorial

```
> db.system.js.insert({_id:"factorial",  value:function(n)
... {
... if(n==1)
... return 1;
... else
... return n*factorial(n-1);
... }
... }
... );

> db.system.js.insert({_id:"factorial",  value:function(n) { if(n==1) return 1; else return
n*factorial(n-1); } } );
> db.system.js.find();
{ "_id" : "factorial", "value" : function (n) { if(n==1) return 1; else return n*factorial(n-1); } }
```

46. Executing the function

```
> db.eval("factorial(3)");
 6
> db.system.js.insert({_id:"sumofn",  value:function(n)
{
... if(n==1)
... return 1;
... else
... return n+sumofn(n-1);
... }
... }
... );
> db.system.js.find();
{ "_id" : "factorial", "value" : function (n) { if(n==1) return 1; else return n*factorial(n-1); } }
{ "_id" : "sumofn", "value" : function (n) { if(n==1) return 1; else return n+sumofn(n-1); } }
> db.eval("sumofn(5)");
15
```

47. Mongo Import

Create a file in CSV format and save it in Bin folder. For importing use the following command

C:\mongodb\bin>mongoimport --db test --collection SampleJSON --type csv --headerline --file ex7.csv

```
connected to: 127.0.0.1
Thu Sep 19 23:24:15.888 check 0 0
Thu Sep 19 23:24:15.891 imported -1 objects
```

48. To check whether the file is imported. Go to MongoDB. Use db test. And check the content in SampleJSON using the command.

db.SampleJSON.find().pretty();

```
> db.SampleJSON.find().pretty();
> db.SampleJSON.find().pretty();
{
    "_id" : ObjectId("5d84a049911abd9ded960197"),
```

```
        "ID" : 1,
        "Name" : "shruti",
        "Rollno" : 8
}
{
        "_id" : ObjectId("5d84a049911abd9ded960198"),
        "ID" : 2,
        "Name" : "wxyz",
        "Rollno" : 2
}
{
        "_id" : ObjectId("5d84a049911abd9ded960199"),
        "ID" : 3,
        "Name" : "abcd",
        "Rollno" : 9
}
```