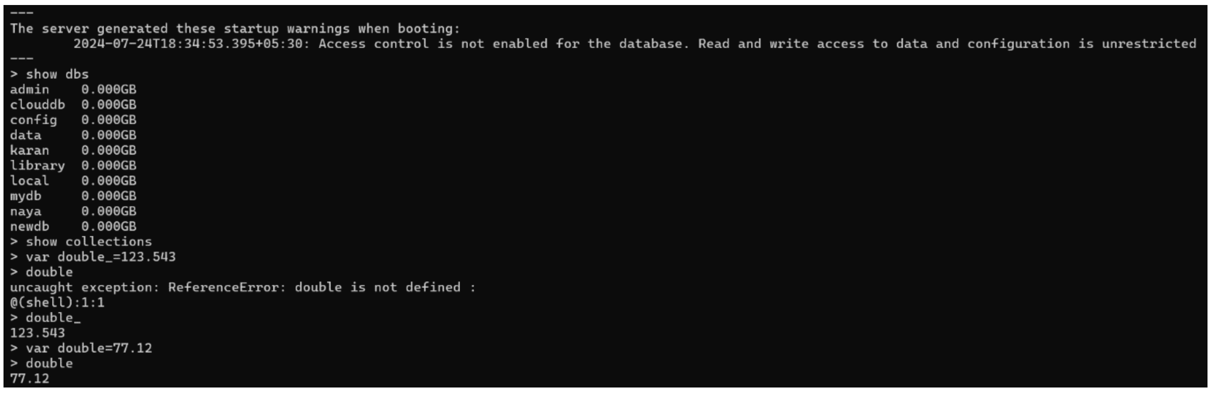
NO SQL DATA TYPES

Name : Dheeraj verma

Roll no : 22BCS012

1. DOUBLE

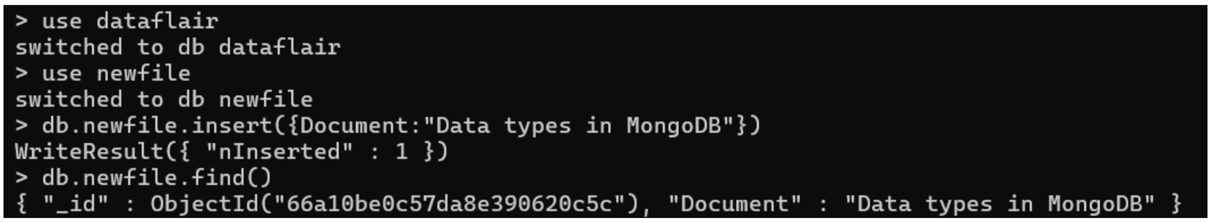
The double data type is used to store floating point values.



```
---
The server generated these startup warnings when booting:
        2024-07-24T18:34:53.395+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
> show dbs
admin    0.000GB
clouddb  0.000GB
config   0.000GB
data     0.000GB
karan    0.000GB
library  0.000GB
local    0.000GB
mydb     0.000GB
naya     0.000GB
newdb    0.000GB
> show collections
> var double_=123.543
> double
uncaught exception: ReferenceError: double is not defined :
@(shell):1:1
> double_
123.543
> var double=77.12
> double
77.12
```

2. String

This is the most commonly used MongoDB data types, BSON strings are UTF-8.

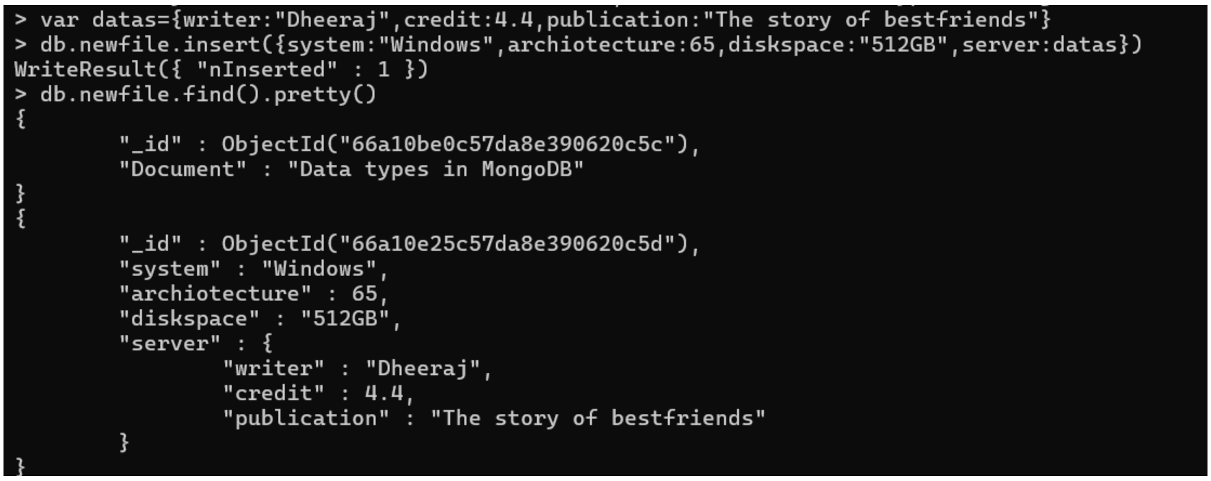


```
> use dataflair
switched to db dataflair
> use newfile
switched to db newfile
> db.newfile.insert({Document:"Data types in MongoDB"})
WriteResult({ "nInserted" : 1 })
> db.newfile.find()
{ "_id" : ObjectId("66a10be0c57da8e390620c5c"), "Document" : "Data types in MongoDB" }
```

3. OBJECT

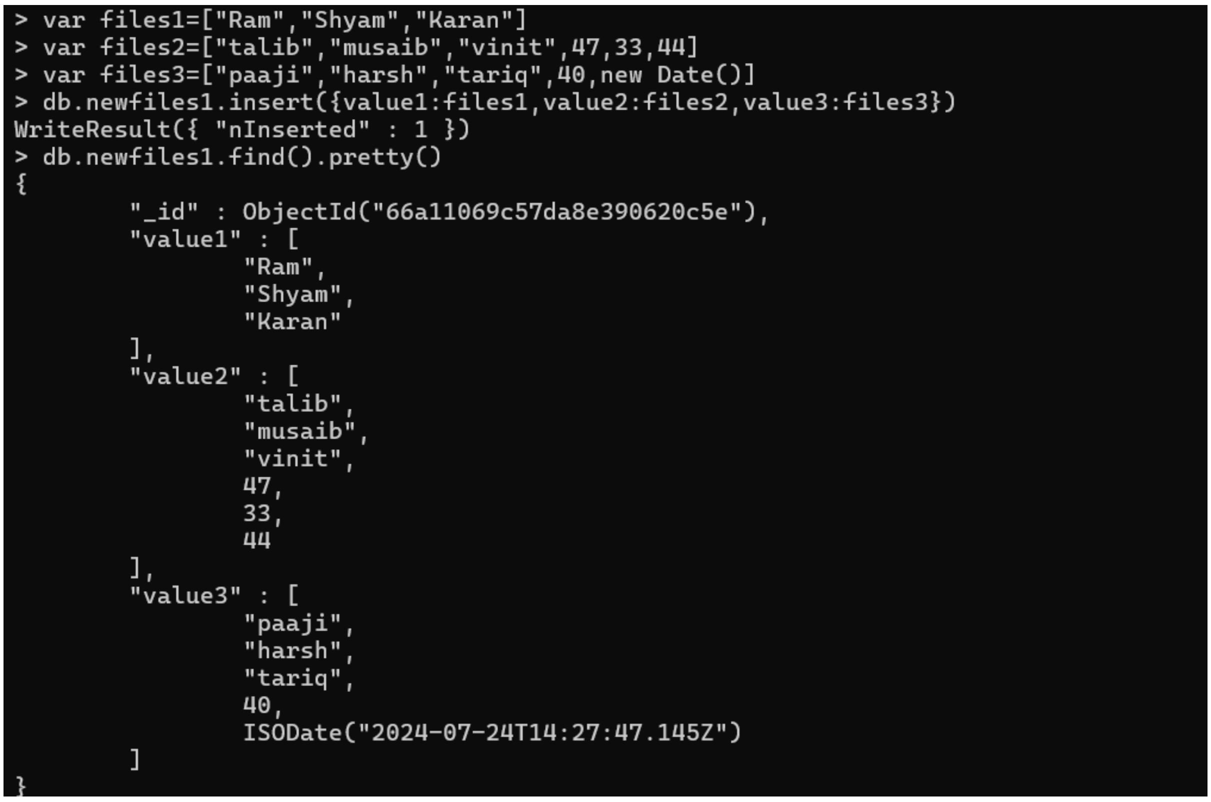
Object data type stores embedded documents. If a document contains another document in the

form of the key-value pair then such type of document is known as an embedded document.



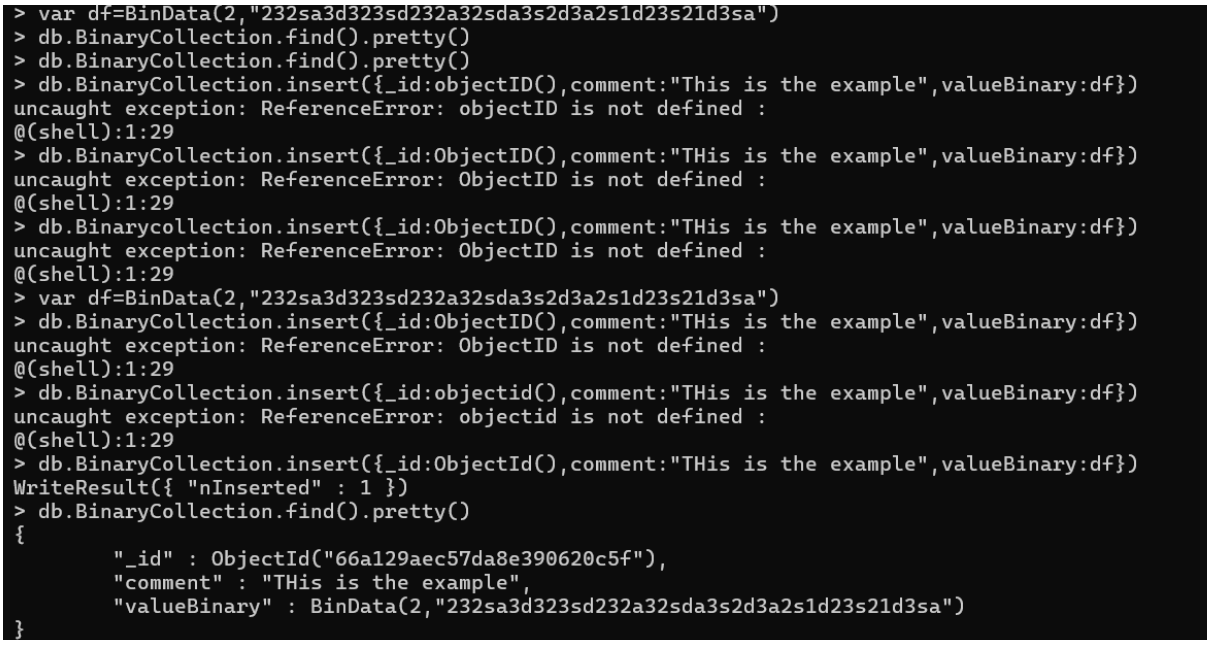
```
> var datas={writer:"Dheeraj",credit:4.4,publication:"The story of bestfriends"}
> db.newfile.insert({system:"Windows",archiotecture:65,diskspace:"512GB",server:datas})
WriteResult({ "nInserted" : 1 })
> db.newfile.find().pretty()
{
        "_id" : ObjectId("66a10be0c57da8e390620c5c"),
        "Document" : "Data types in MongoDB"
}
{
        "_id" : ObjectId("66a10e25c57da8e390620c5d"),
        "system" : "Windows",
        "archiotecture" : 65,
        "diskspace" : "512GB",
        "server" : {
                "writer" : "Dheeraj",
                "credit" : 4.4,
                "publication" : "The story of bestfriends"
        }
}
```

4. ARRAY



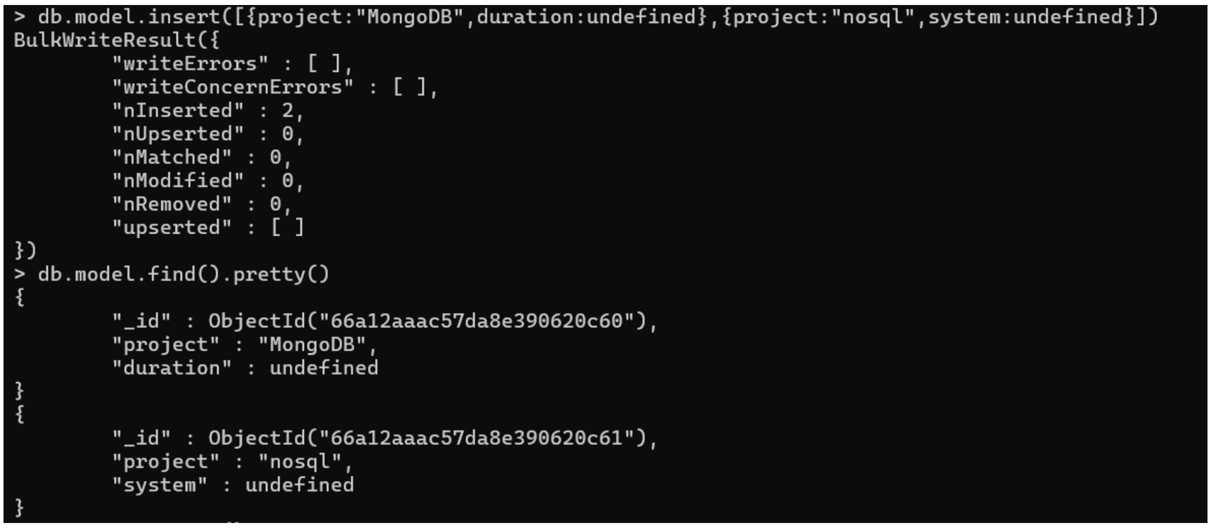
```
> var files1=["Ram","Shyam","Karan"]
> var files2=["talib","musaib","vinit",47,33,44]
> var files3=["paaji","harsh","tariq",40,new Date()]
> db.newfiles1.insert({value1:files1,value2:files2,value3:files3})
WriteResult({ "nInserted" : 1 })
> db.newfiles1.find().pretty()
{
        "_id" : ObjectId("66a11069c57da8e390620c5e"),
        "value1" : [
                "Ram",
                "Shyam",
                "Karan"
        ],
        "value2" : [
                "talib",
                "musaib",
                "vinit",
                47,
                33,
                44
        ],
        "value3" : [
                "paaji",
                "harsh",
                "tariq",
                40,
                ISODate("2024-07-24T14:27:47.145Z")
        ]
}
```

5. BINARY DATA



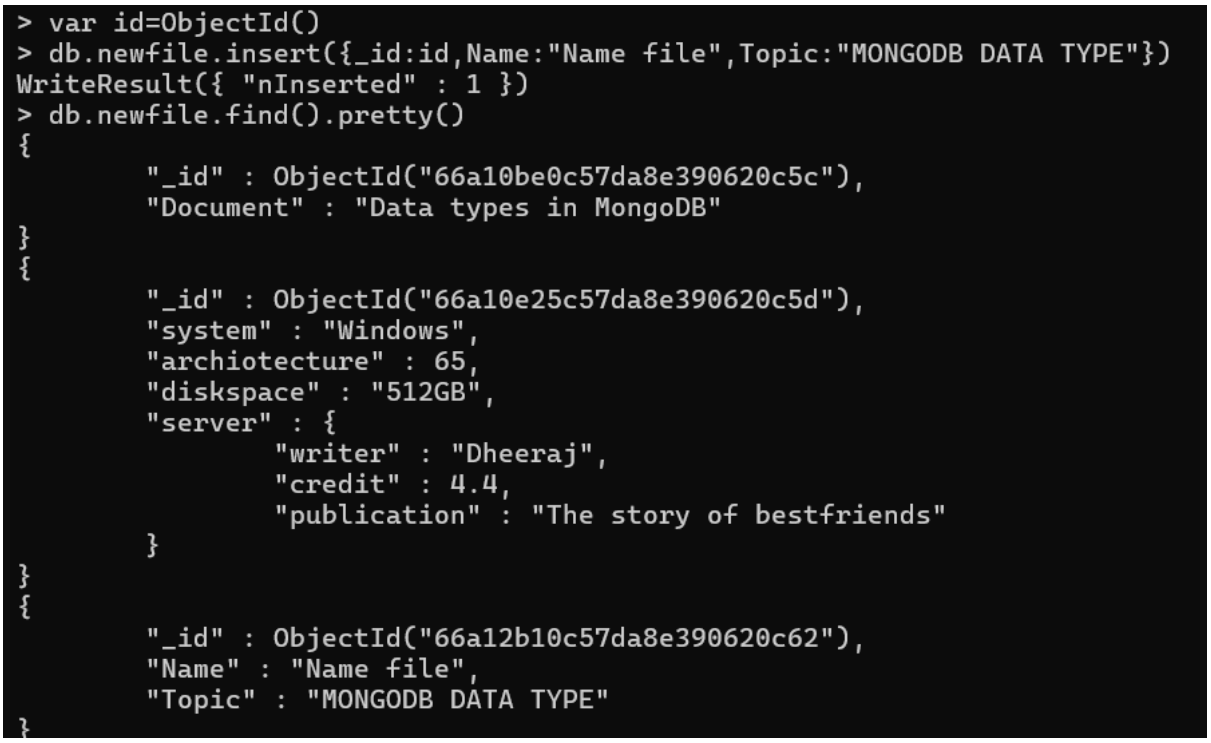
```
> var df=BinData(2,"232sa3d323sd232a32sda3s2d3a2s1d23s21d3sa")
> db.BinaryCollection.find().pretty()
> db.BinaryCollection.find().pretty()
> db.BinaryCollection.insert({_id:objectID(),comment:"This is the example",valueBinary:df})
uncaught exception: ReferenceError: objectID is not defined :
@(shell):1:29
> db.BinaryCollection.insert({_id:ObjectID(),comment:"THis is the example",valueBinary:df})
uncaught exception: ReferenceError: ObjectID is not defined :
@(shell):1:29
> db.Binarycollection.insert({_id:ObjectID(),comment:"THis is the example",valueBinary:df})
uncaught exception: ReferenceError: ObjectID is not defined :
@(shell):1:29
> var df=BinData(2,"232sa3d323sd232a32sda3s2d3a2s1d23s21d3sa")
> db.BinaryCollection.insert({_id:ObjectID(),comment:"THis is the example",valueBinary:df})
uncaught exception: ReferenceError: ObjectID is not defined :
@(shell):1:29
> db.BinaryCollection.insert({_id:objectid(),comment:"THis is the example",valueBinary:df})
uncaught exception: ReferenceError: objectid is not defined :
@(shell):1:29
> db.BinaryCollection.insert({_id:ObjectId(),comment:"THis is the example",valueBinary:df})
WriteResult({ "nInserted" : 1 })
> db.BinaryCollection.find().pretty()
{
        "_id" : ObjectId("66a129aec57da8e390620c5f"),
        "comment" : "THis is the example",
        "valueBinary" : BinData(2,"232sa3d323sd232a32sda3s2d3a2s1d23s21d3sa")
}
```

6. UNDEFINED



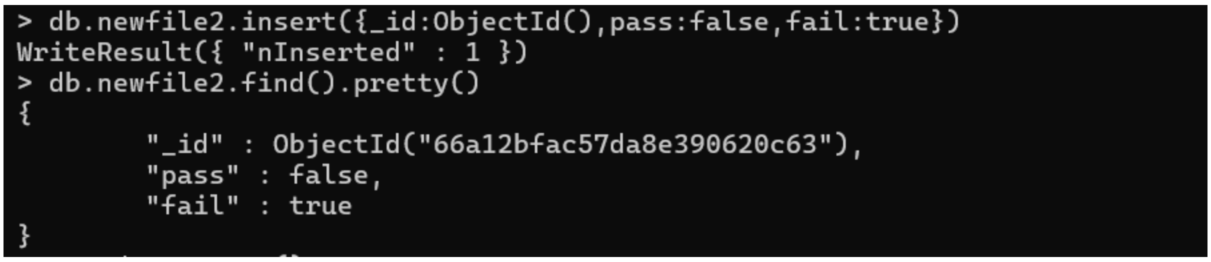
```
> db.model.insert([{project:"MongoDB",duration:undefined},{project:"nosql",system:undefined}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.model.find().pretty()
{
        "_id" : ObjectId("66a12aaac57da8e390620c60"),
        "project" : "MongoDB",
        "duration" : undefined
}
{
        "_id" : ObjectId("66a12aaac57da8e390620c61"),
        "project" : "nosql",
        "system" : undefined
}
```

7. OBJECT ID



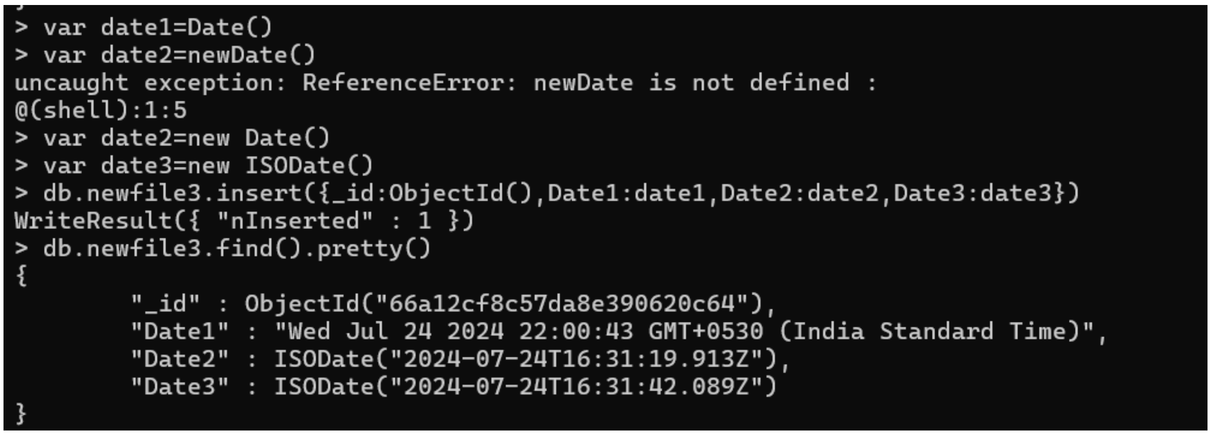
```
> var id=ObjectId()
> db.newfile.insert({_id:id,Name:"Name file",Topic:"MONGODB DATA TYPE"})
WriteResult({ "nInserted" : 1 })
> db.newfile.find().pretty()
{
        "_id" : ObjectId("66a10be0c57da8e390620c5c"),
        "Document" : "Data types in MongoDB"
}
{
        "_id" : ObjectId("66a10e25c57da8e390620c5d"),
        "system" : "Windows",
        "archiotecture" : 65,
        "diskspace" : "512GB",
        "server" : {
                "writer" : "Dheeraj",
                "credit" : 4.4,
                "publication" : "The story of bestfriends"
        }
}
{
        "_id" : ObjectId("66a12b10c57da8e390620c62"),
        "Name" : "Name file",
        "Topic" : "MONGODB DATA TYPE"
}
```

8. BOOLEAN



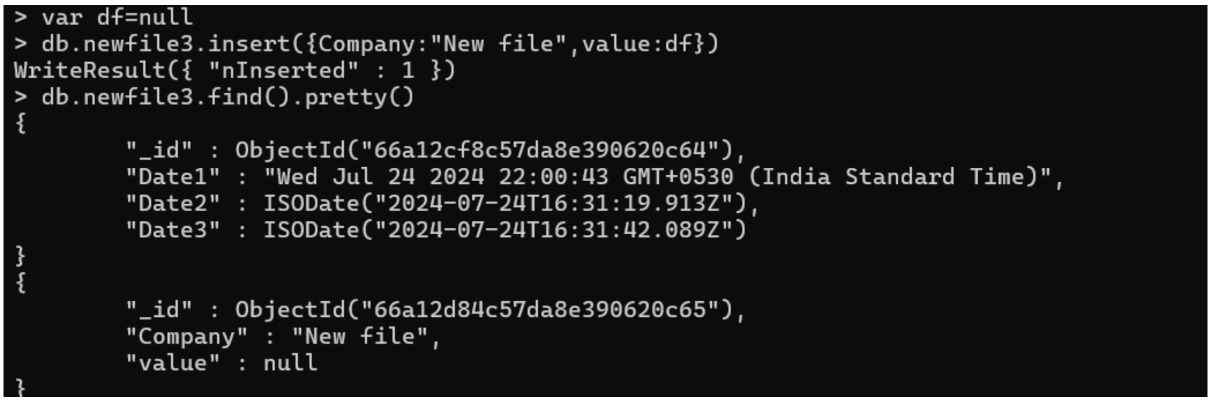
```
> db.newfile2.insert({_id:ObjectId(),pass:false,fail:true})
WriteResult({ "nInserted" : 1 })
> db.newfile2.find().pretty()
{
        "_id" : ObjectId("66a12bfac57da8e390620c63"),
        "pass" : false,
        "fail" : true
}
```

9. DATE



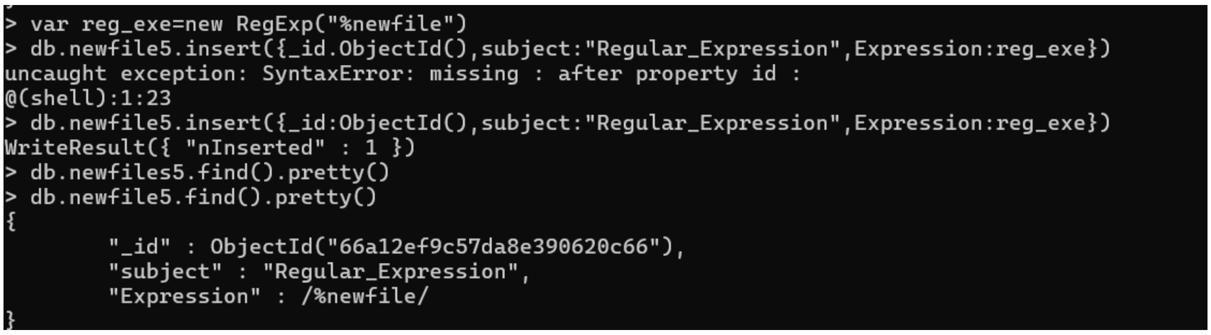
```
> var date1=Date()
> var date2=newDate()
uncaught exception: ReferenceError: newDate is not defined :
@(shell):1:5
> var date2=new Date()
> var date3=new ISODate()
> db.newfile3.insert({_id:ObjectId(),Date1:date1,Date2:date2,Date3:date3})
WriteResult({ "nInserted" : 1 })
> db.newfile3.find().pretty()
{
        "_id" : ObjectId("66a12cf8c57da8e390620c64"),
        "Date1" : "Wed Jul 24 2024 22:00:43 GMT+0530 (India Standard Time)",
        "Date2" : ISODate("2024-07-24T16:31:19.913Z"),
        "Date3" : ISODate("2024-07-24T16:31:42.089Z")
}
```

10. NULL



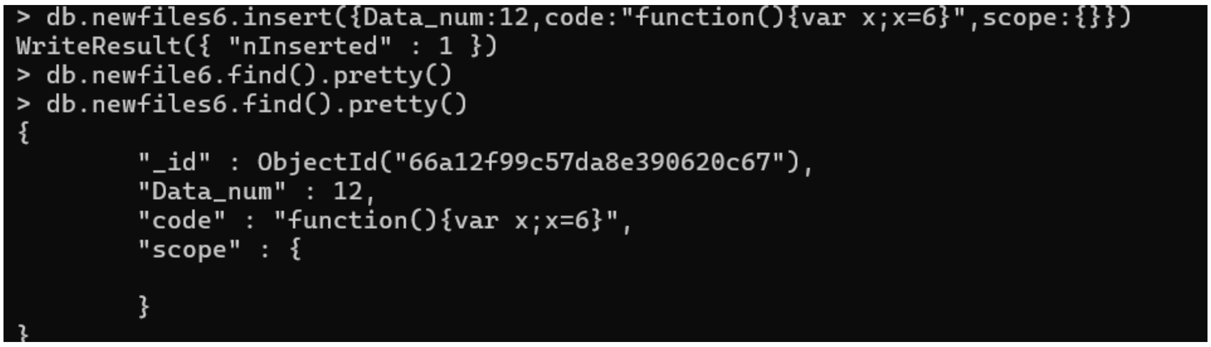
```
> var df=null
> db.newfile3.insert({Company:"New file",value:df})
WriteResult({ "nInserted" : 1 })
> db.newfile3.find().pretty()
{
        "_id" : ObjectId("66a12cf8c57da8e390620c64"),
        "Date1" : "Wed Jul 24 2024 22:00:43 GMT+0530 (India Standard Time)",
        "Date2" : ISODate("2024-07-24T16:31:19.913Z"),
        "Date3" : ISODate("2024-07-24T16:31:42.089Z")
}
{
        "_id" : ObjectId("66a12d84c57da8e390620c65"),
        "Company" : "New file",
        "value" : null
}
```

11. REGULAR EXPRESSION



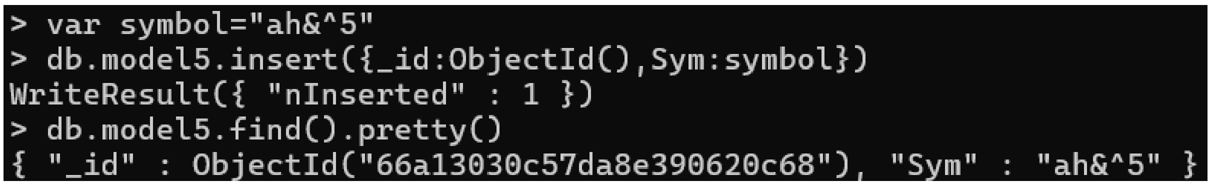
```
> var reg_exe=new RegExp("%newfile")
> db.newfile5.insert({_id.ObjectId(),subject:"Regular_Expression",Expression:reg_exe})
uncaught exception: SyntaxError: missing : after property id :
@(shell):1:23
> db.newfile5.insert({_id:ObjectId(),subject:"Regular_Expression",Expression:reg_exe})
WriteResult({ "nInserted" : 1 })
> db.newfiles5.find().pretty()
> db.newfile5.find().pretty()
{
        "_id" : ObjectId("66a12ef9c57da8e390620c66"),
        "subject" : "Regular_Expression",
        "Expression" : /%newfile/
}
```

12. JAVA SCRIPT



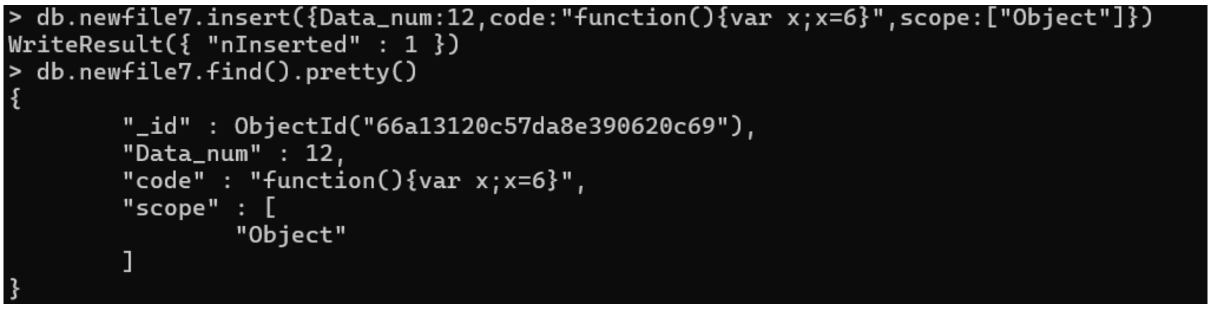
```
> db.newfiles6.insert({Data_num:12,code:"function(){var x;x=6}",scope:{}})
WriteResult({ "nInserted" : 1 })
> db.newfile6.find().pretty()
> db.newfiles6.find().pretty()
{
        "_id" : ObjectId("66a12f99c57da8e390620c67"),
        "Data_num" : 12,
        "code" : "function(){var x;x=6}",
        "scope" : {

        }
}
```

13. SYMBOL



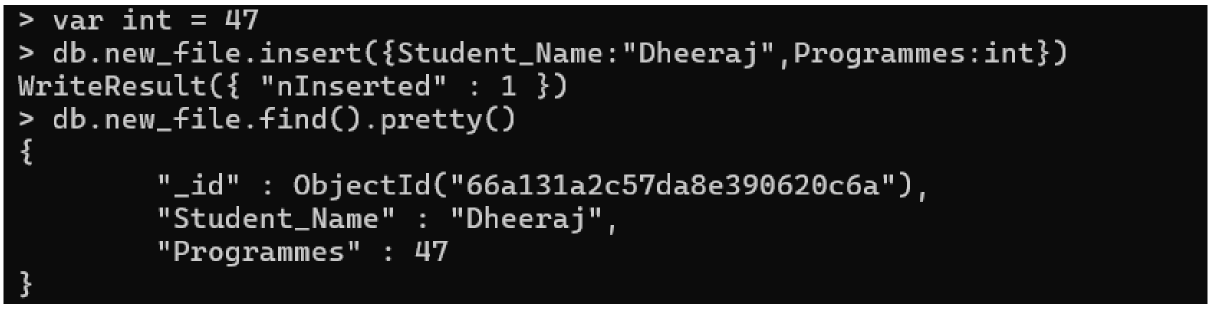
```
> var symbol="ah&^5"
> db.model5.insert({_id:ObjectId(),Sym:symbol})
WriteResult({ "nInserted" : 1 })
> db.model5.find().pretty()
{ "_id" : ObjectId("66a13030c57da8e390620c68"), "Sym" : "ah&^5" }
```

14. JavaScript with Scope



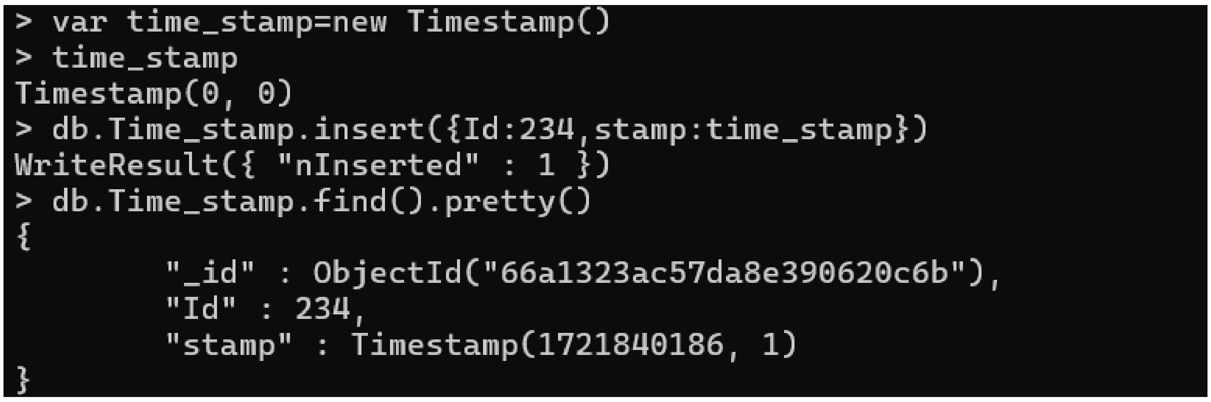
```
> db.newfile7.insert({Data_num:12,code:"function(){var x;x=6}",scope:["Object"]})
WriteResult({ "nInserted" : 1 })
> db.newfile7.find().pretty()
{
        "_id" : ObjectId("66a13120c57da8e390620c69"),
        "Data_num" : 12,
        "code" : "function(){var x;x=6}",
        "scope" : [
                "Object"
        ]
}
```

15. INTEGET



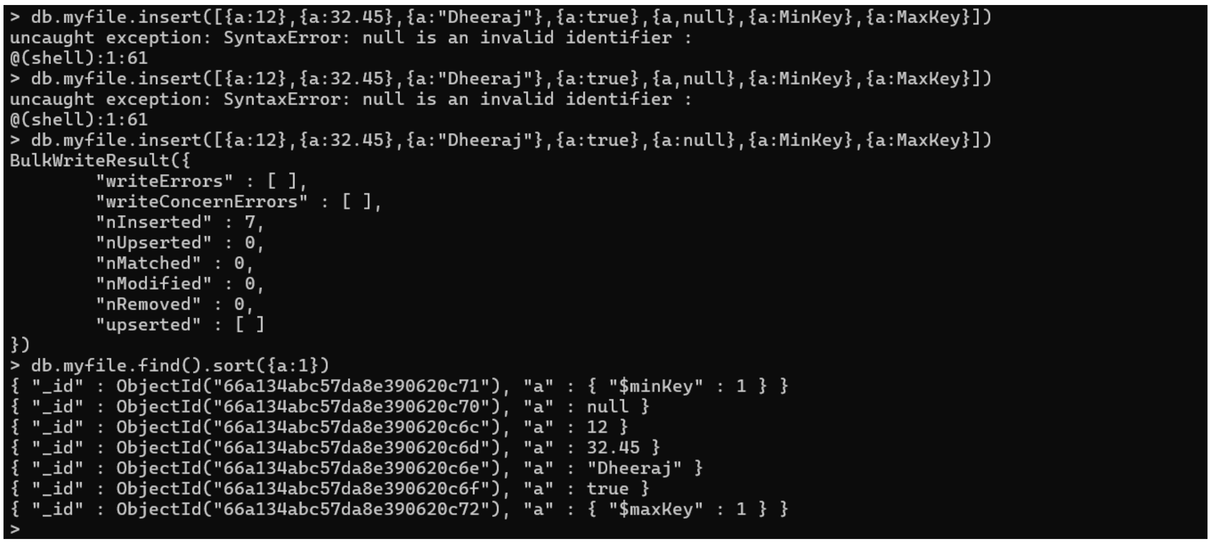
```
> var int = 47
> db.new_file.insert({Student_Name:"Dheeraj",Programmes:int})
WriteResult({ "nInserted" : 1 })
> db.new_file.find().pretty()
{
        "_id" : ObjectId("66a131a2c57da8e390620c6a"),
        "Student_Name" : "Dheeraj",
        "Programmes" : 47
}
```

16. TIMESTAMP



```
> var time_stamp=new Timestamp()
> time_stamp
Timestamp(0, 0)
> db.Time_stamp.insert({Id:234,stamp:time_stamp})
WriteResult({ "nInserted" : 1 })
> db.Time_stamp.find().pretty()
{
        "_id" : ObjectId("66a1323ac57da8e390620c6b"),
        "Id" : 234,
        "stamp" : Timestamp(1721840186, 1)
}
```

17. MIN & MAX KEY



```
> db.myfile.insert([{a:12},{a:32.45},{a:"Dheeraj"},{a:true},{a,null},{a:MinKey},{a:MaxKey}])
uncaught exception: SyntaxError: null is an invalid identifier :
@(shell):1:61
> db.myfile.insert([{a:12},{a:32.45},{a:"Dheeraj"},{a:true},{a,null},{a:MinKey},{a:MaxKey}])
uncaught exception: SyntaxError: null is an invalid identifier :
@(shell):1:61
> db.myfile.insert([{a:12},{a:32.45},{a:"Dheeraj"},{a:true},{a:null},{a:MinKey},{a:MaxKey}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 7,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.myfile.find().sort({a:1})
{ "_id" : ObjectId("66a134abc57da8e390620c71"), "a" : { "$minKey" : 1 } }
{ "_id" : ObjectId("66a134abc57da8e390620c70"), "a" : null }
{ "_id" : ObjectId("66a134abc57da8e390620c6c"), "a" : 12 }
{ "_id" : ObjectId("66a134abc57da8e390620c6d"), "a" : 32.45 }
{ "_id" : ObjectId("66a134abc57da8e390620c6e"), "a" : "Dheeraj" }
{ "_id" : ObjectId("66a134abc57da8e390620c6f"), "a" : true }
{ "_id" : ObjectId("66a134abc57da8e390620c72"), "a" : { "$maxKey" : 1 } }
>
```