ST. GONSALO GARCIA COLLEGE
PAPDI, VASAI

ST.GONSALO GARCIA COLLEGE OF ARTS AND COMMERCE

CERTIFICATE

This is to certify that Master ______ARNOLD AWLYN VAZ______,
Roll No.:___55___ of T.Y.B.SC. (IT) Sem-V has successfully completed practical's of the subject **'Next Generation Technology'** the year 2020-2021 under the guidance of Prof. Brensa Cerejo.

Internal Examiner's Sign   Head of Department

Date: Principal

Index

	b. jQuery Callback, jQuery Get and Set Contents		
	c. jQuery Insert Content, jQuery Remove Elements and Attribute		
9	**JSON**		
	a. Creating JSON		
	b. Parsing JSON		
	c. Persisting JSON		
10	**Create a JSON file and import it to MongoDB**		
	a. Export MongoDB to JSON		
	b. Write a MongoDB query to delete JSON object from MongoDB		

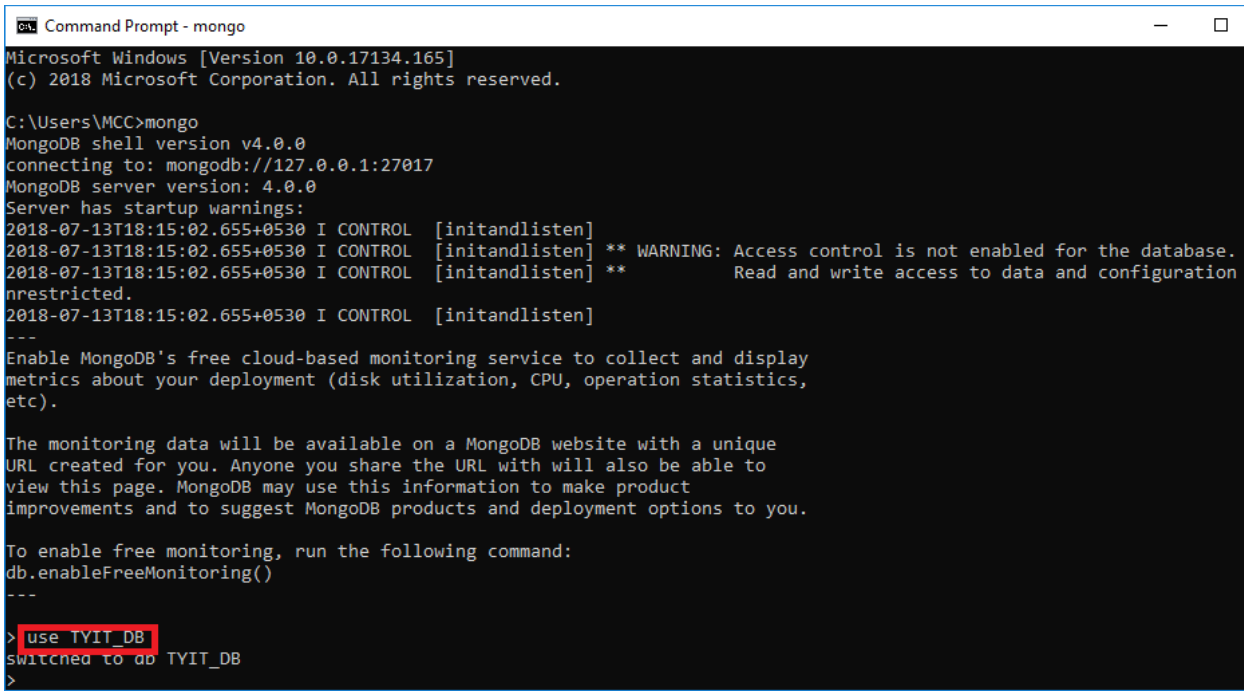
Practical 1

Practical 1:-

(a) Write a MongoDB command to create and drop database

(i)Creating database

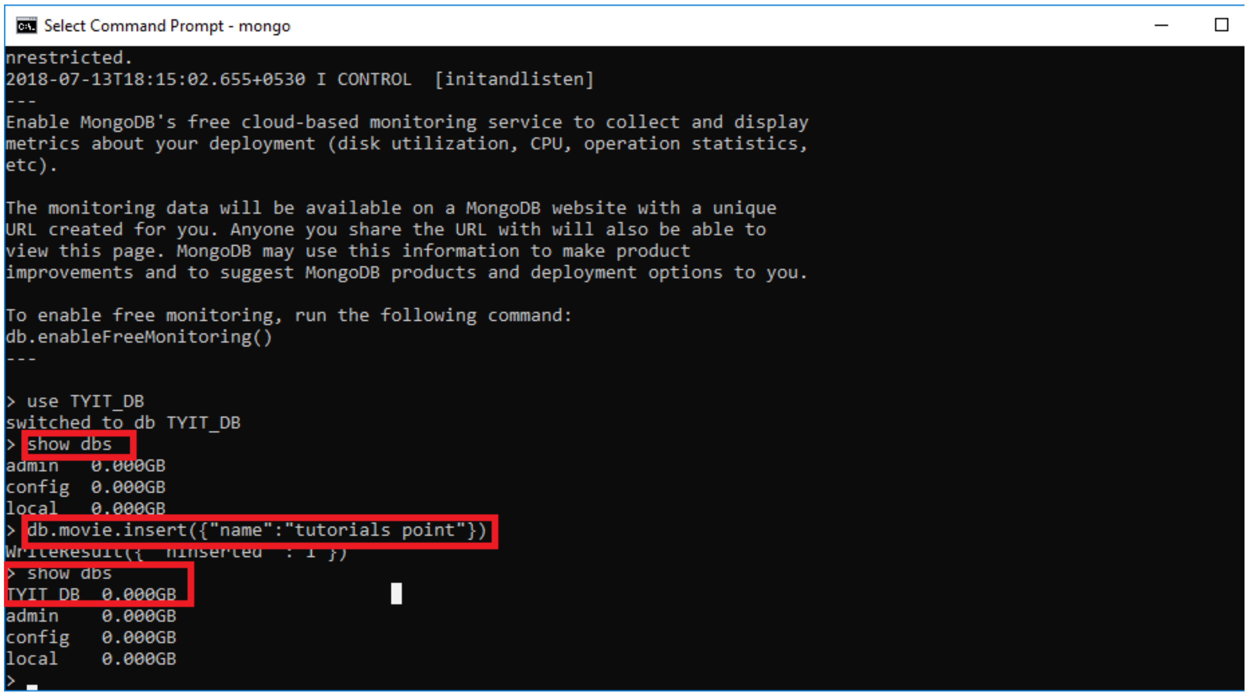
```
use DATABASE_NAME
```



Command Prompt - mongo

```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MCC>mongo
MongoDB shell version v4.0.0
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 4.0.0
Server has startup warnings:
2018-07-13T18:15:02.655+0530 I CONTROL  [initandlisten]
2018-07-13T18:15:02.655+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-07-13T18:15:02.655+0530 I CONTROL  [initandlisten] **          Read and write access to data and configuration
nrestricted.
2018-07-13T18:15:02.655+0530 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> use TYIT_DB
switched to db TYIT_DB
>
```

Use “show dbs “ to display databases available . Note: You have to insert document to make your database visible " db.movie.insert({"name":"tutorials point"})”



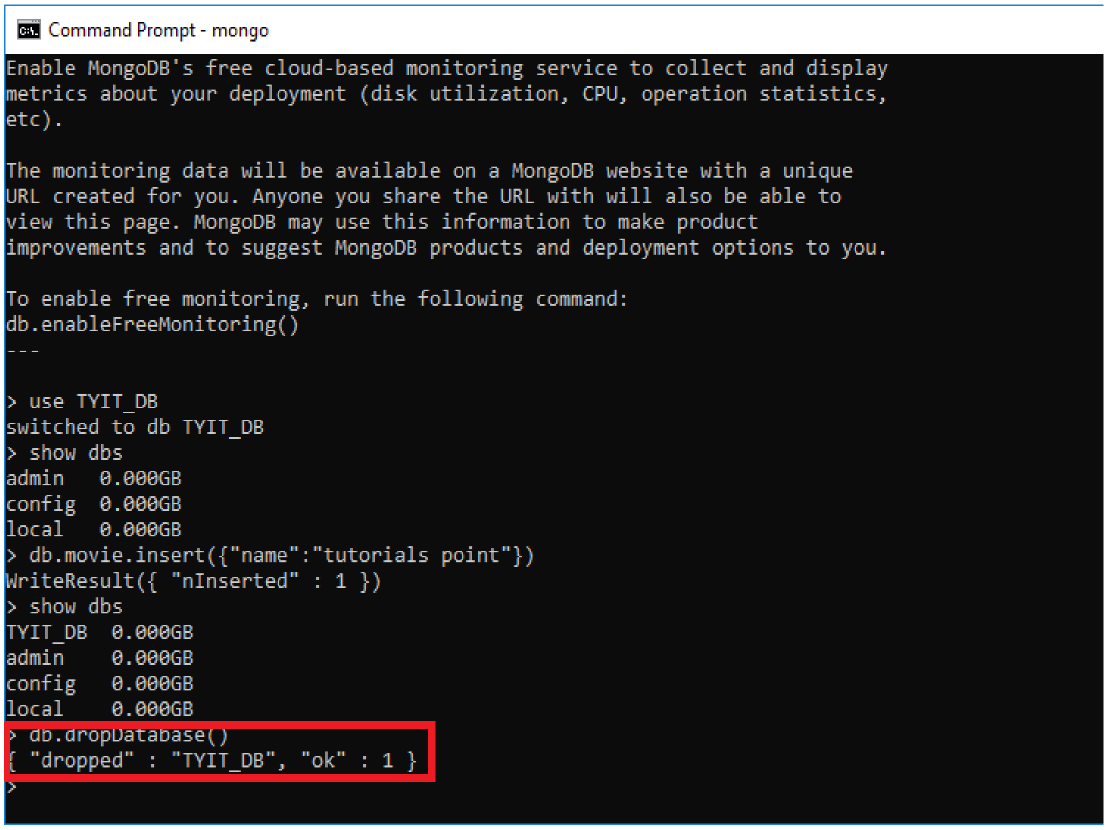
Select Command Prompt - mongo

```
nrestricted.
2018-07-13T18:15:02.655+0530 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> use TYIT_DB
switched to db TYIT_DB
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.movie.insert({"name":"tutorials point"})
WriteResult({ "nInserted" : 1 })
> show dbs
TYIT_DB  0.000GB
admin    0.000GB
config   0.000GB
local    0.000GB
>
```

(ii)DROP database

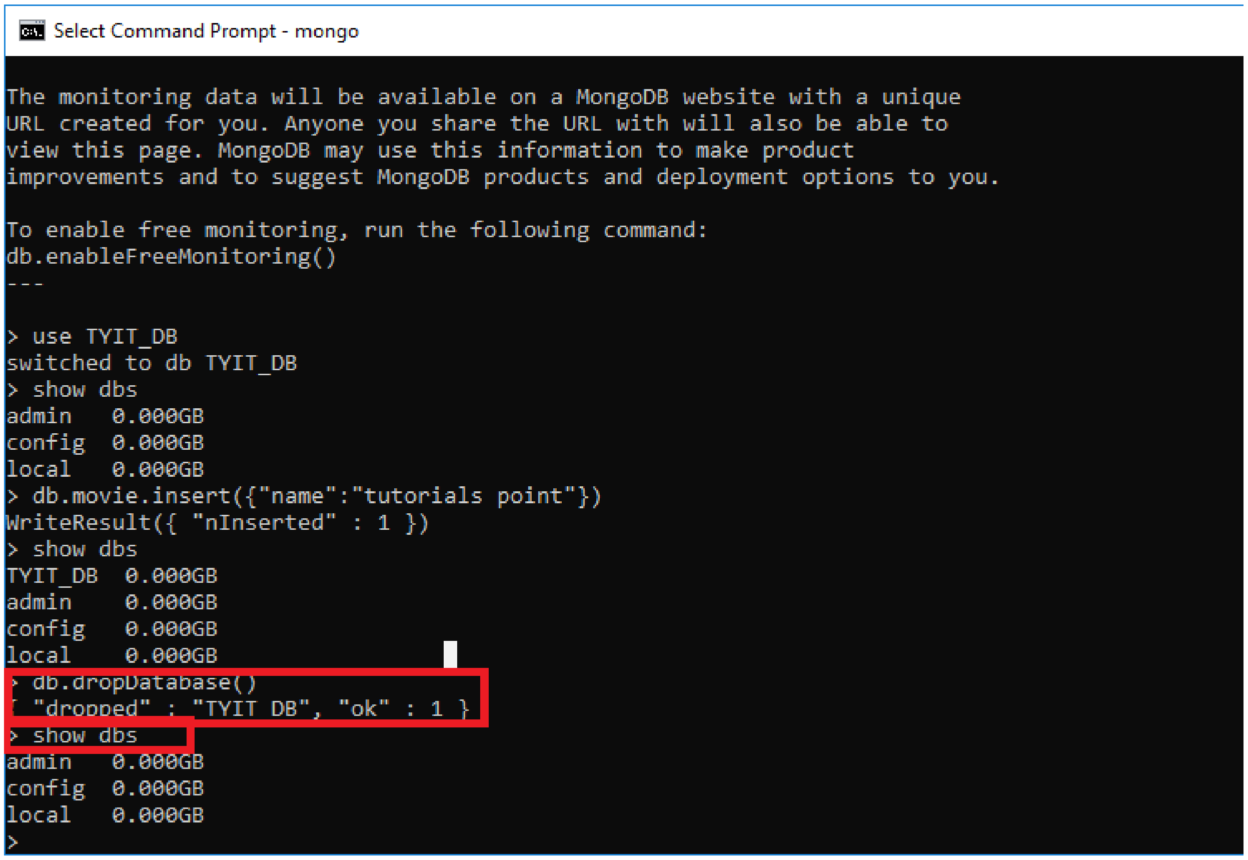
```
db.dropDatabase()
```

It drop the current database

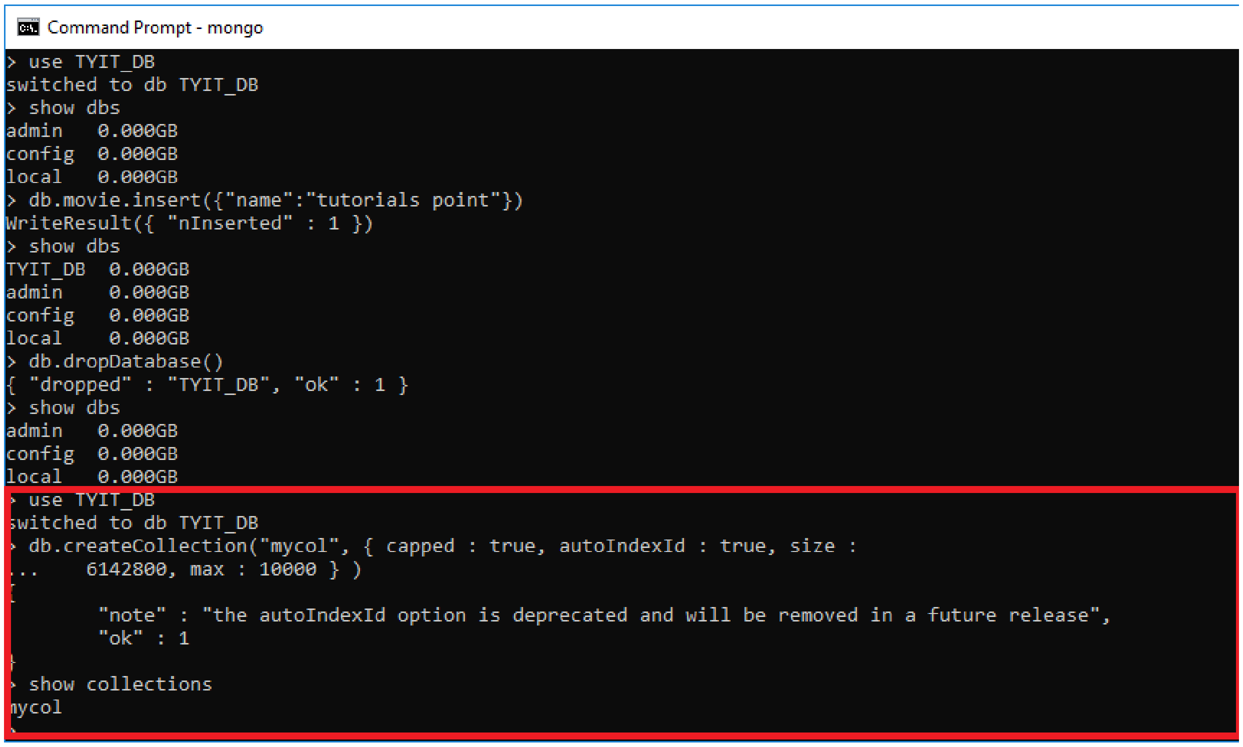


```
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> use TYIT_DB
switched to db TYIT_DB
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.movie.insert({"name":"tutorials point"})
WriteResult({ "nInserted" : 1 })
> show dbs
TYIT_DB  0.000GB
admin    0.000GB
config   0.000GB
local    0.000GB
> db.dropDatabase()
{ "dropped" : "TYIT_DB", "ok" : 1 }
>
```

Show dbs to see the changes


Select Command Prompt - mongo

```
The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> use TYIT_DB
switched to db TYIT_DB
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.movie.insert({"name":"tutorials point"})
WriteResult({ "nInserted" : 1 })
> show dbs
TYIT_DB  0.000GB
admin    0.000GB
config   0.000GB
local    0.000GB
> db.dropDatabase()
{ "dropped" : "TYIT_DB", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```



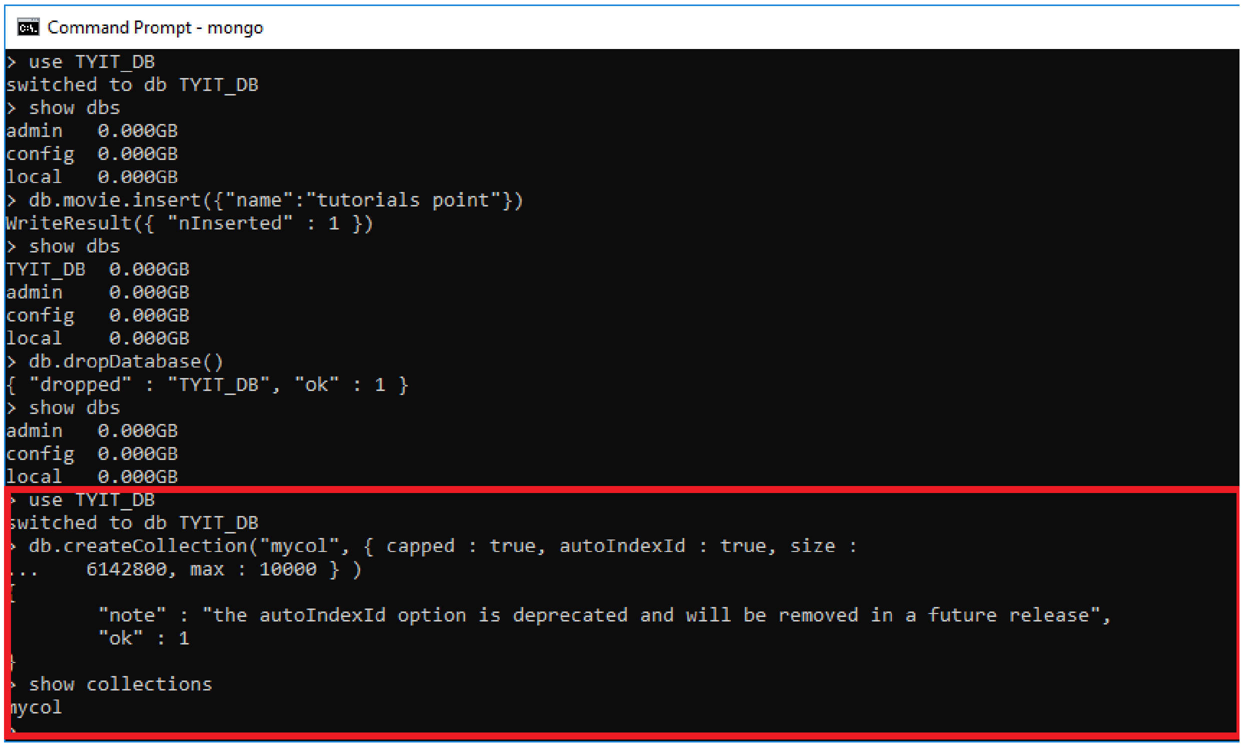
```
> use TYIT_DB
switched to db TYIT_DB
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.movie.insert({"name":"tutorials point"})
WriteResult({ "nInserted" : 1 })
> show dbs
TYIT_DB  0.000GB
admin    0.000GB
config   0.000GB
local    0.000GB
> db.dropDatabase()
{ "dropped" : "TYIT_DB", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use TYIT_DB
switched to db TYIT_DB
> db.createCollection("mycol", { capped : true, autoIndexId : true, size :
...    6142800, max : 10000 } )
{
        "note" : "the autoIndexId option is deprecated and will be removed in a future release",
        "ok" : 1
}
> show collections
mycol
>
```

(b) write a mongodb query to create ,display and drop collection

(i)Creating Collection

db.createCollection("mycol", { capped : true, autoIndexId : true, size :

  6142800, max : 10000 })

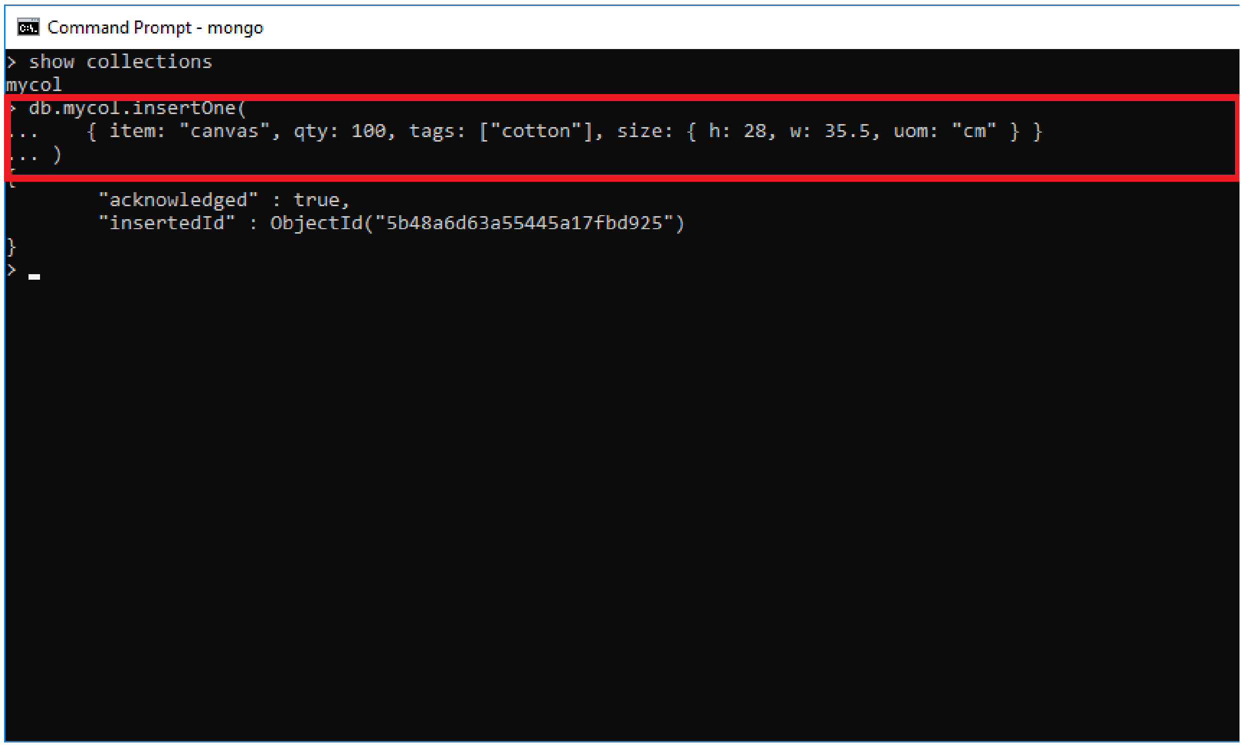

Command Prompt - mongo

```
> use TYIT_DB
switched to db TYIT_DB
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.movie.insert({"name":"tutorials point"})
WriteResult({ "nInserted" : 1 })
> show dbs
TYIT_DB  0.000GB
admin    0.000GB
config   0.000GB
local    0.000GB
> db.dropDatabase()
{ "dropped" : "TYIT_DB", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use TYIT_DB
switched to db TYIT_DB
> db.createCollection("mycol", { capped : true, autoIndexId : true, size :
...     6142800, max : 10000 } )
{
        "note" : "the autoIndexId option is deprecated and will be removed in a future release",
        "ok" : 1
}
> show collections
mycol
>
```

(ii) Inserting into Collections

Inserting into collection:-

```
db.mycol.insertOne(
   { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
)
```

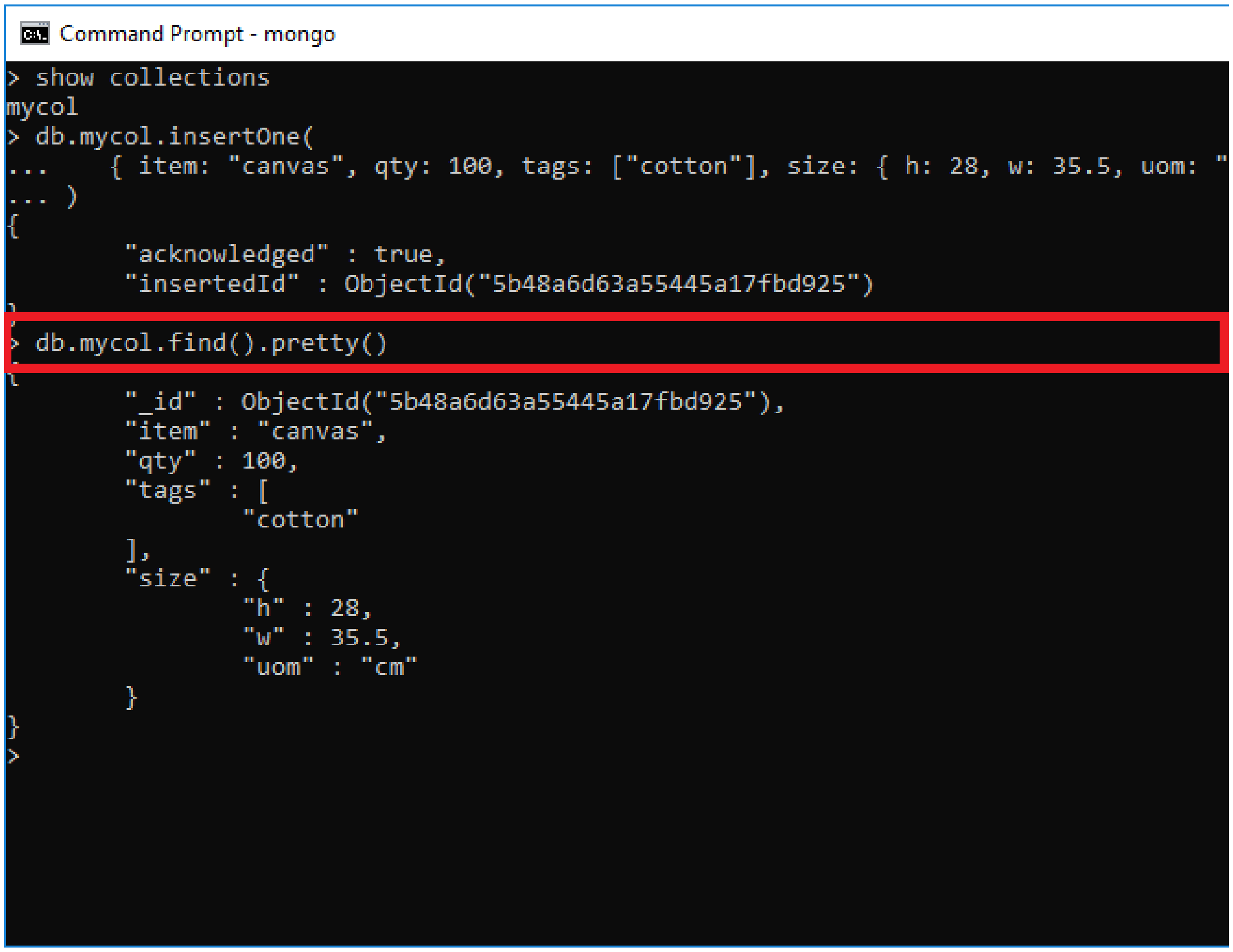


Command Prompt - mongo

```
> show collections
mycol
> db.mycol.insertOne(
...    { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5b48a6d63a55445a17fbd925")
}
> _
```

(iii) Displaying the Collections

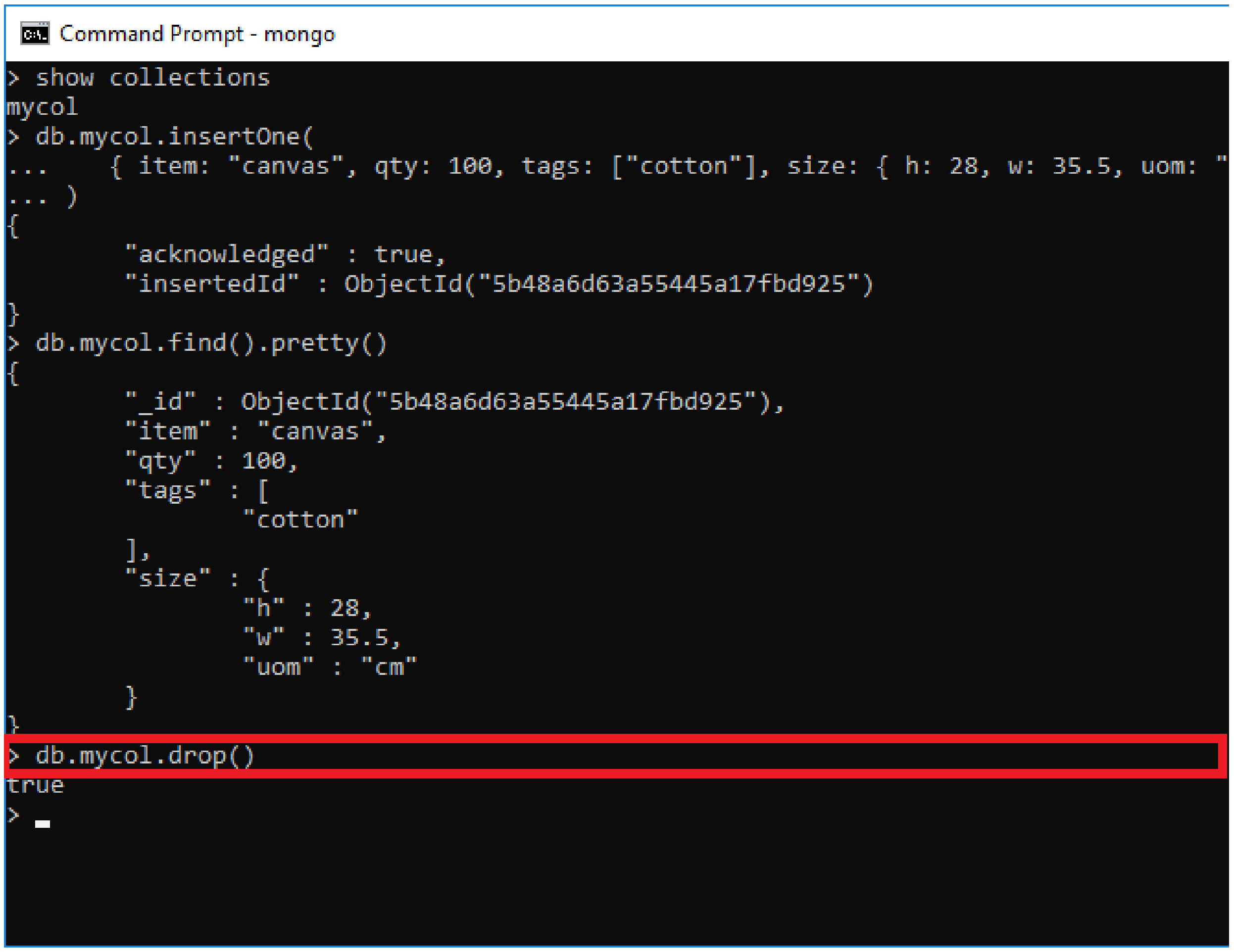
db.mycol.find().pretty()



```
> show collections
mycol
> db.mycol.insertOne(
...    { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5b48a6d63a55445a17fbd925")
}
> db.mycol.find().pretty()
{
        "_id" : ObjectId("5b48a6d63a55445a17fbd925"),
        "item" : "canvas",
        "qty" : 100,
        "tags" : [
                "cotton"
        ],
        "size" : {
                "h" : 28,
                "w" : 35.5,
                "uom" : "cm"
        }
}
>
```

(iv) Dropping Collections

db.mycol.drop()

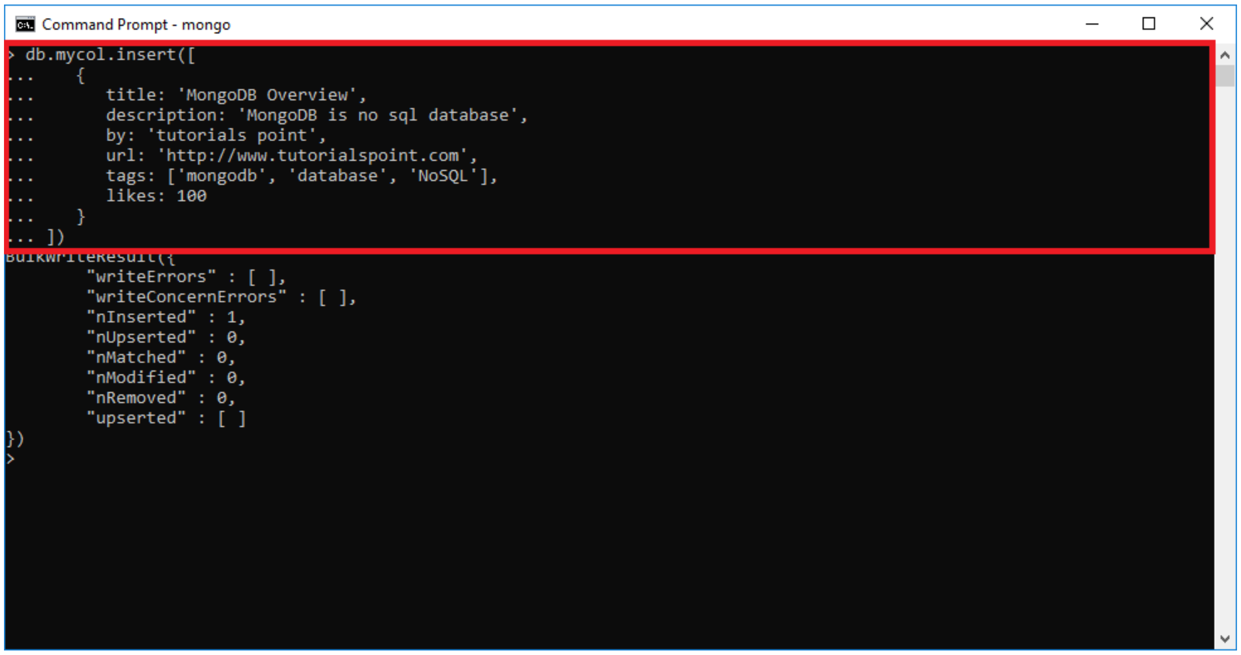


```
> show collections
mycol
> db.mycol.insertOne(
...    { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5b48a6d63a55445a17fbd925")
}
> db.mycol.find().pretty()
{
        "_id" : ObjectId("5b48a6d63a55445a17fbd925"),
        "item" : "canvas",
        "qty" : 100,
        "tags" : [
                "cotton"
        ],
        "size" : {
                "h" : 28,
                "w" : 35.5,
                "uom" : "cm"
        }
}
> db.mycol.drop()
true
>
```

c) Write a MongoDB to insert, query ,update and delete the documents

(i) Inserting into the documents

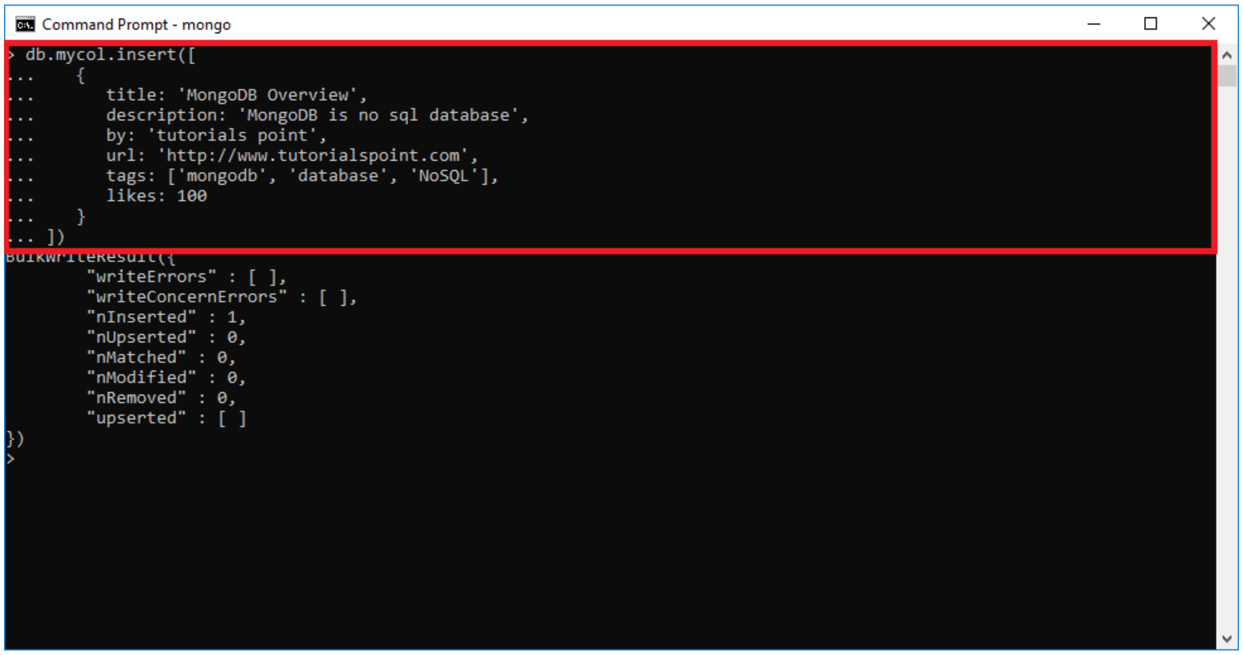
```
db.mycol.insert([
  {
    title: 'MongoDB Overview',
    description: 'MongoDB is no sql database',
    by: 'tutorials point',
    url: 'http://www.tutorialspoint.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
  }
])
```



```
> db.mycol.insert([
...    {
...       title: 'MongoDB Overview',
...       description: 'MongoDB is no sql database',
...       by: 'tutorials point',
...       url: 'http://www.tutorialspoint.com',
...       tags: ['mongodb', 'database', 'NoSQL'],
...       likes: 100
...    }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 1,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

(ii) Quering documents

db.mycol.find().pretty()



Command Prompt - mongo

```
> db.mycol.insert([
...    {
...       title: 'MongoDB Overview',
...       description: 'MongoDB is no sql database',
...       by: 'tutorials point',
...       url: 'http://www.tutorialspoint.com',
...       tags: ['mongodb', 'database', 'NoSQL'],
...       likes: 100
...    }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 1,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

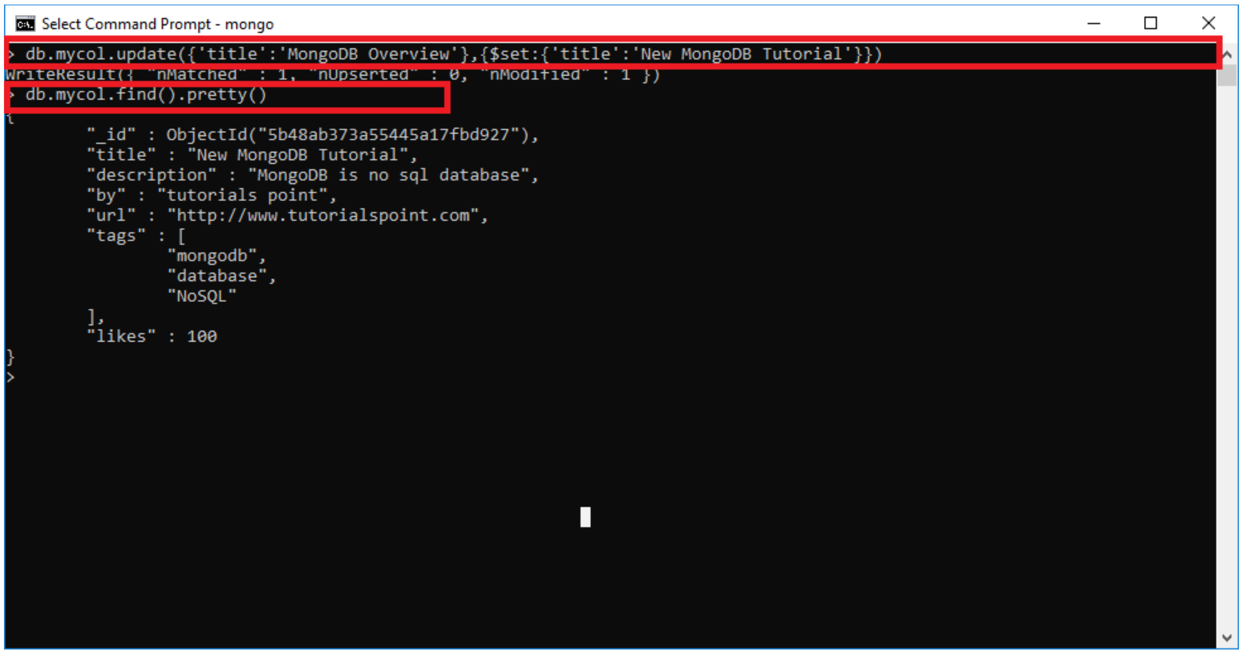
(iii) Updating Documents

To update the Documents

db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})

To display the Documents

db.mycol.find().pretty()



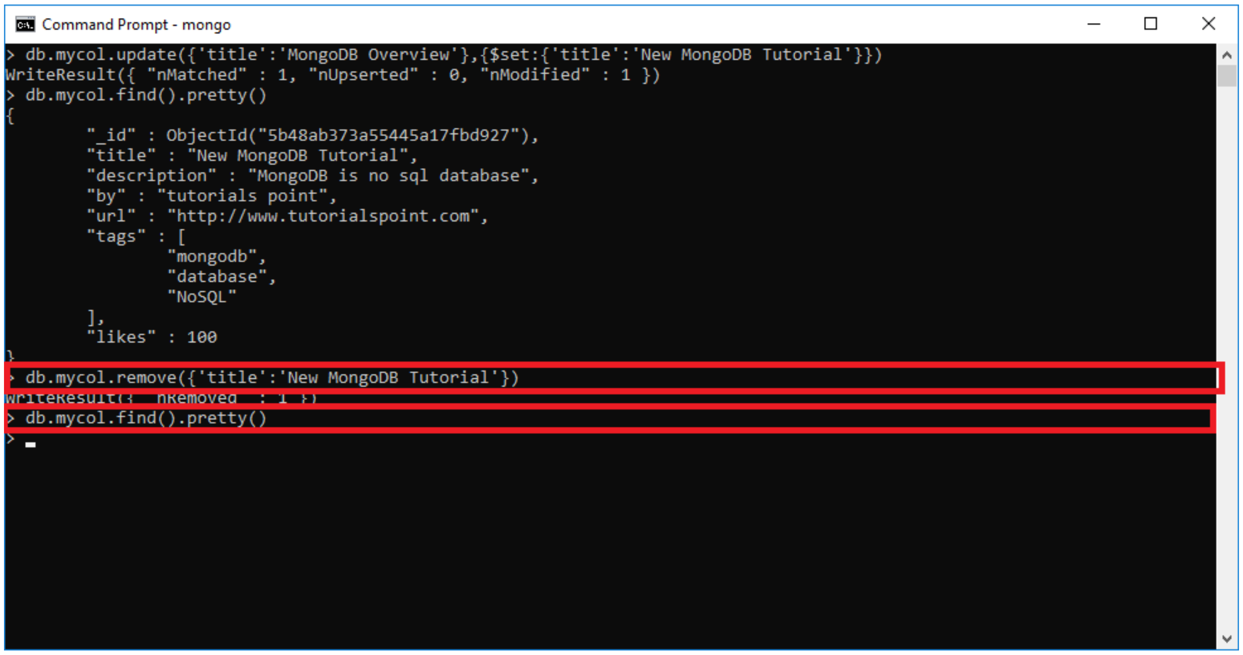
Select Command Prompt - mongo

```
> db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.mycol.find().pretty()
{
        "_id" : ObjectId("5b48ab373a55445a17fbd927"),
        "title" : "New MongoDB Tutorial",
        "description" : "MongoDB is no sql database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
>
```

(iv) Removing Documents

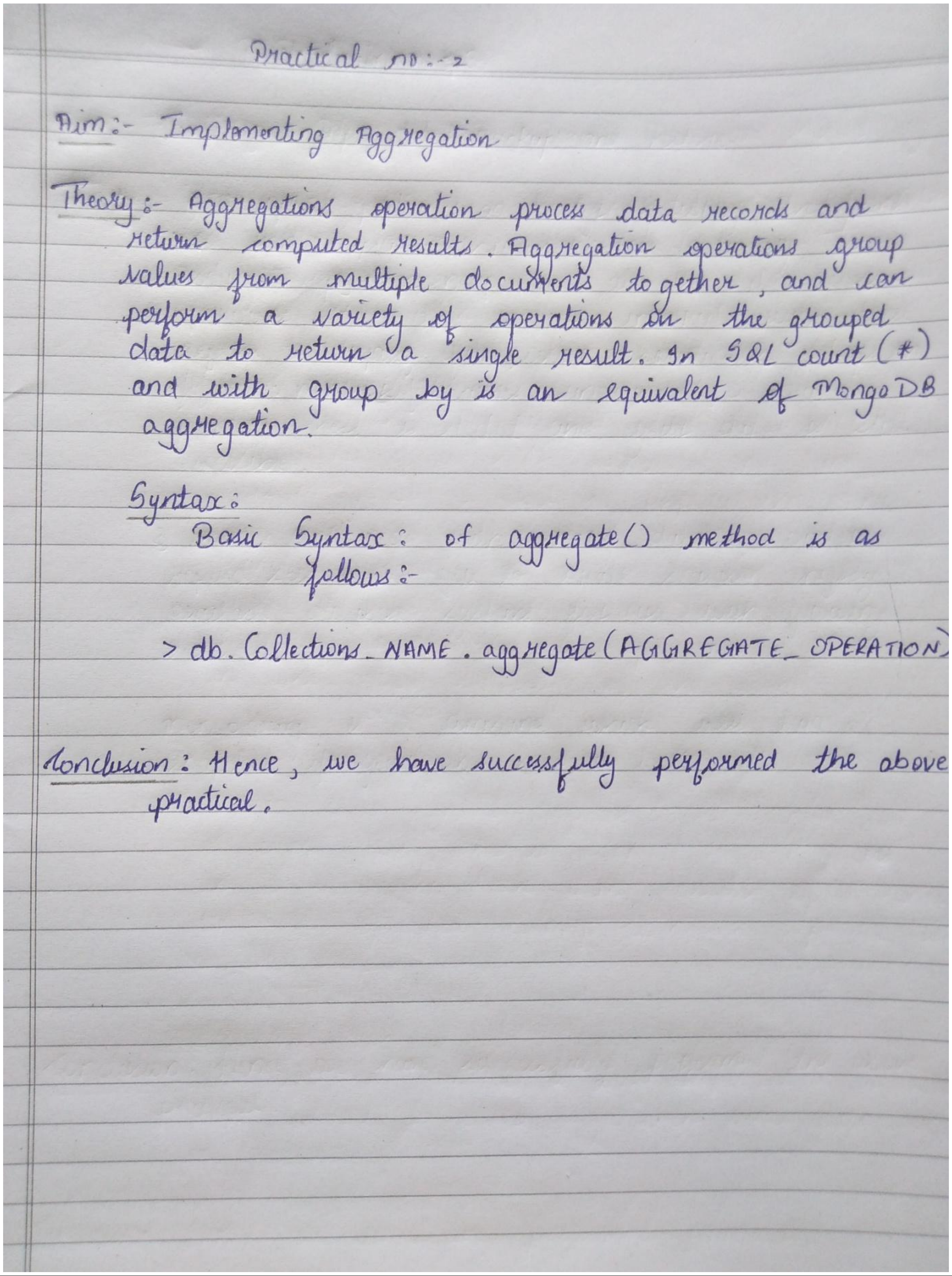
db.mycol.remove({'title':'New MongoDB Tutorial'})

db.mycol.find().pretty()



```
> db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.mycol.find().pretty()
{
        "_id" : ObjectId("5b48ab373a55445a17fbd927"),
        "title" : "New MongoDB Tutorial",
        "description" : "MongoDB is no sql database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
> db.mycol.remove({'title':'New MongoDB Tutorial'})
WriteResult({ "nRemoved" : 1 })
> db.mycol.find().pretty()
> _
```

Practical 2



Practical no :- 2

Aim :- Implementing Aggregation

Theory :- Aggregations operation process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In SQL count (*) and with group by is an equivalent of MongoDB aggregation.

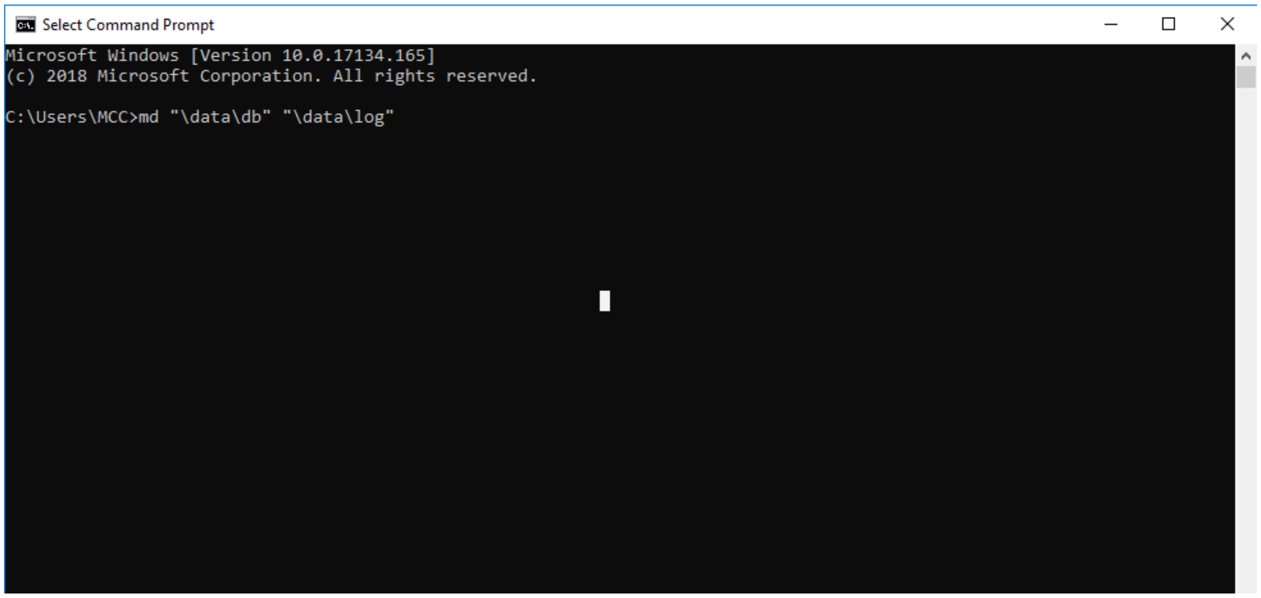
Syntax :

Basic Syntax : of aggregate() method is as follows :-

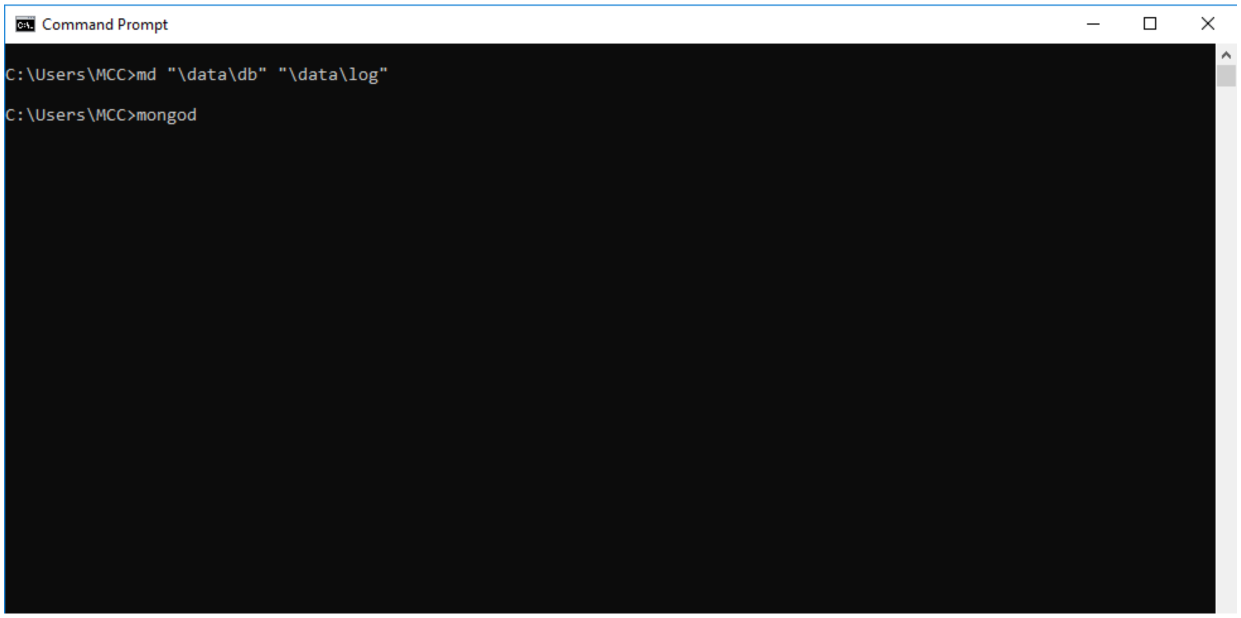
```
> db.Collections_NAME . aggregate(AGGREGATE_OPERATION)
```

Conclusion : Hence, we have successfully performed the above practical.

Open New command prompt and create a data\db folder and then start the server using “mangod”

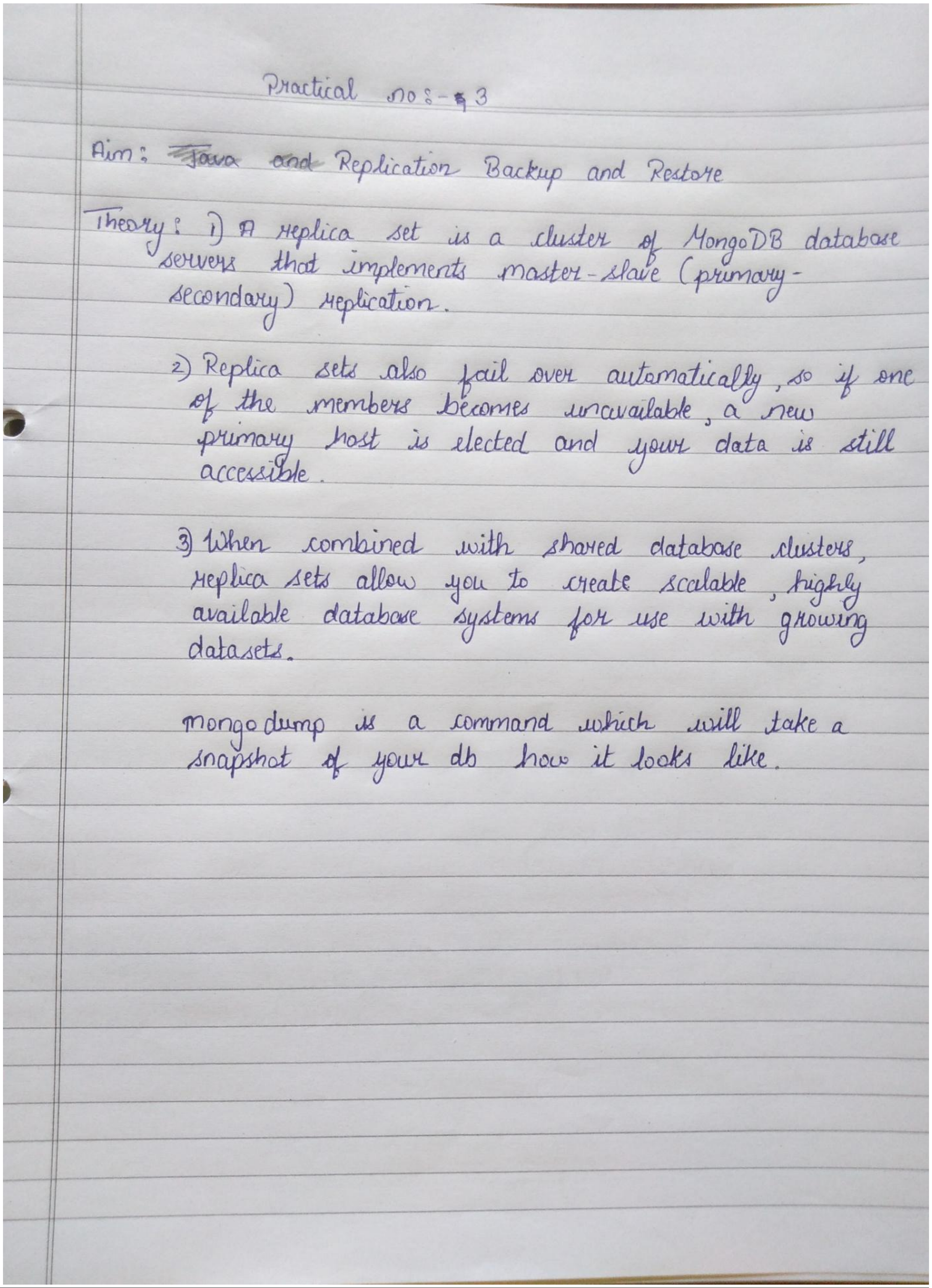


```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MCC>md "\data\db" "\data\log"
```



```
C:\Users\MCC>md "\data\db" "\data\log"

C:\Users\MCC>mongod
```

Practical 3



Practical no:- 3

Aim: ~~Java and~~ Replication Backup and Restore

Theory: 1) A replica set is a cluster of MongoDB database servers that implements master-slave (primary-secondary) replication.

2) Replica sets also fail over automatically, so if one of the members becomes unavailable, a new primary host is elected and your data is still accessible.

3) When combined with shared database clusters, replica sets allow you to create scalable, highly available database systems for use with growing datasets.

mongodump is a command which will take a snapshot of your db how it looks like.

Prac 3(Implementing aggregation)

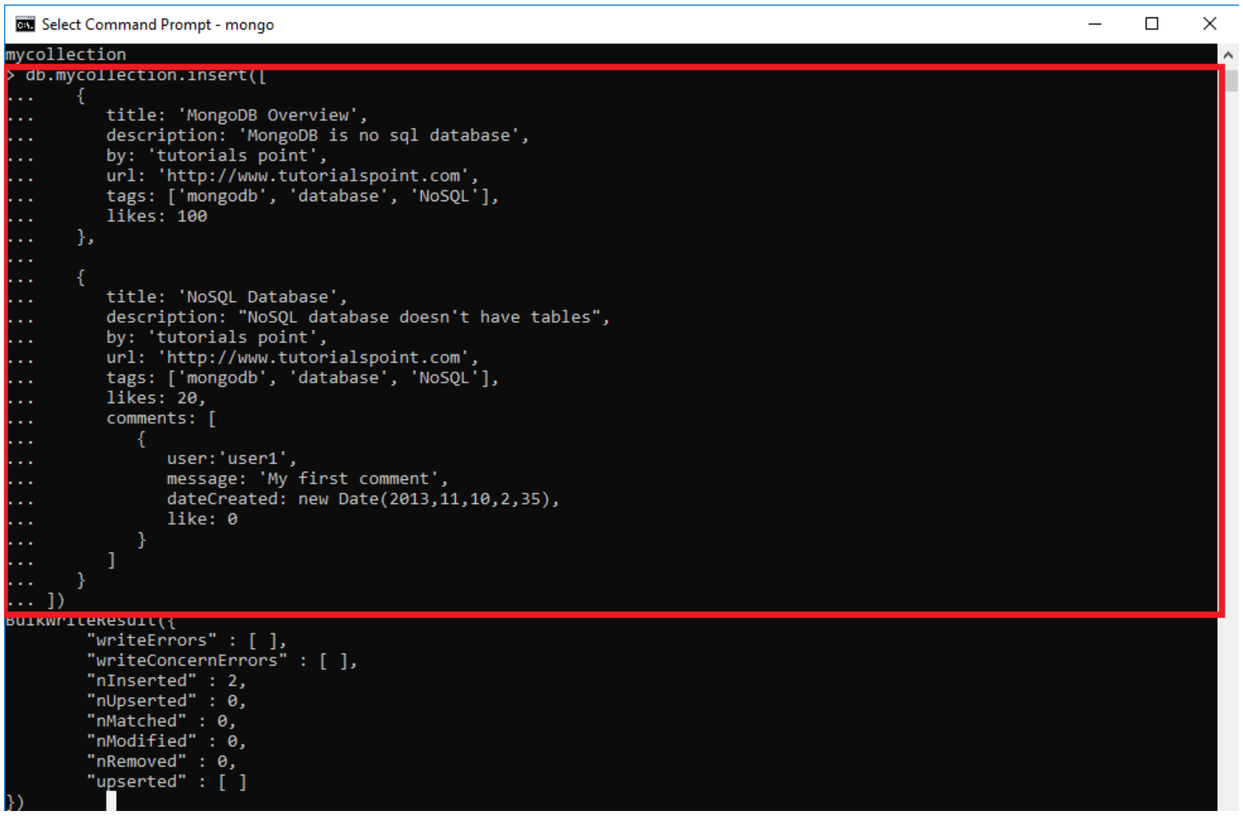
a) Write a MongoDB query to use sum,avg,min,max expression

Firstly create a New collection

```
db.createCollection("mycollection")
```

Then Insert 2 documents into it

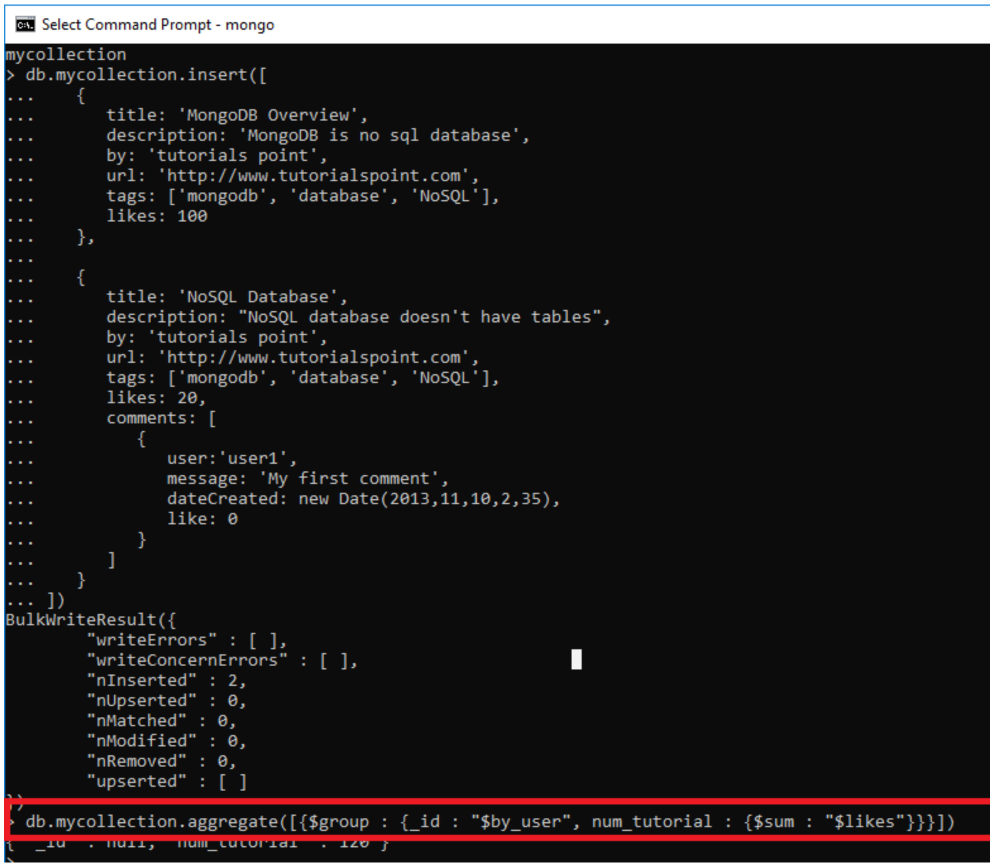
```
db.mycollection.insert([
  {
    title: 'MongoDB Overview',
    description: 'MongoDB is no sql database',
    by: 'tutorials point',
    url: 'http://www.tutorialspoint.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
  },
  {
    title: 'NoSQL Database',
    description: "NoSQL database doesn't have tables",
    by: 'tutorials point',
    url: 'http://www.tutorialspoint.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 20,
    comments: [
      {
        user:'user1',
        message: 'My first comment',
        dateCreated: new Date(2013,11,10,2,35),
        like: 0
      }
    ]
  }
])
```


Select Command Prompt - mongo

```
mycollection
> db.mycollection.insert([
...    {
...       title: 'MongoDB Overview',
...       description: 'MongoDB is no sql database',
...       by: 'tutorials point',
...       url: 'http://www.tutorialspoint.com',
...       tags: ['mongodb', 'database', 'NoSQL'],
...       likes: 100
...    },
...
...    {
...       title: 'NoSQL Database',
...       description: "NoSQL database doesn't have tables",
...       by: 'tutorials point',
...       url: 'http://www.tutorialspoint.com',
...       tags: ['mongodb', 'database', 'NoSQL'],
...       likes: 20,
...       comments: [
...          {
...             user:'user1',
...             message: 'My first comment',
...             dateCreated: new Date(2013,11,10,2,35),
...             like: 0
...          }
...       ]
...    }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

(i)SUM

db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])

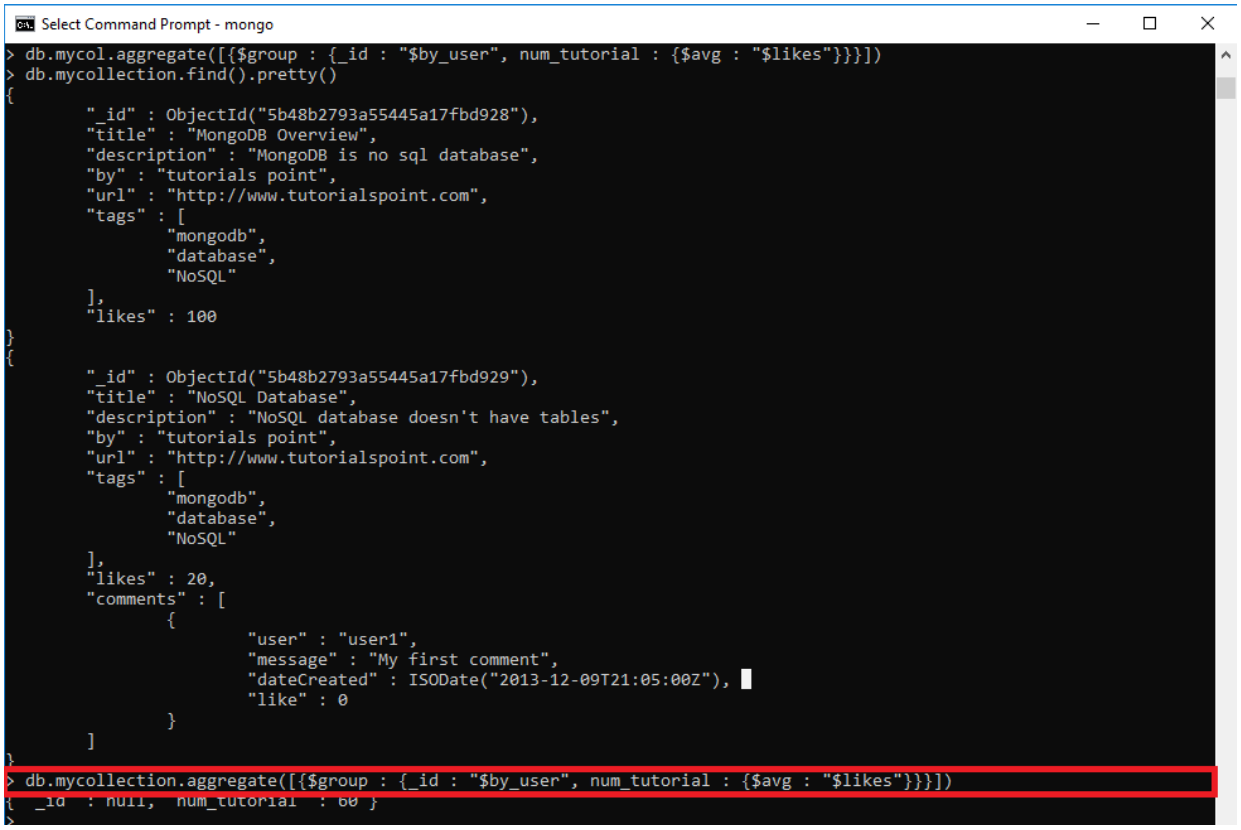


Select Command Prompt - mongo

```
mycollection
> db.mycollection.insert([
...     {
...        title: 'MongoDB Overview',
...        description: 'MongoDB is no sql database',
...        by: 'tutorials point',
...        url: 'http://www.tutorialspoint.com',
...        tags: ['mongodb', 'database', 'NoSQL'],
...        likes: 100
...     },
...
...     {
...        title: 'NoSQL Database',
...        description: "NoSQL database doesn't have tables",
...        by: 'tutorials point',
...        url: 'http://www.tutorialspoint.com',
...        tags: ['mongodb', 'database', 'NoSQL'],
...        likes: 20,
...        comments: [
...           {
...              user:'user1',
...              message: 'My first comment',
...              dateCreated: new Date(2013,11,10,2,35),
...              like: 0
...           }
...        ]
...     }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 120 }
>
```

(ii) Avg

db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])

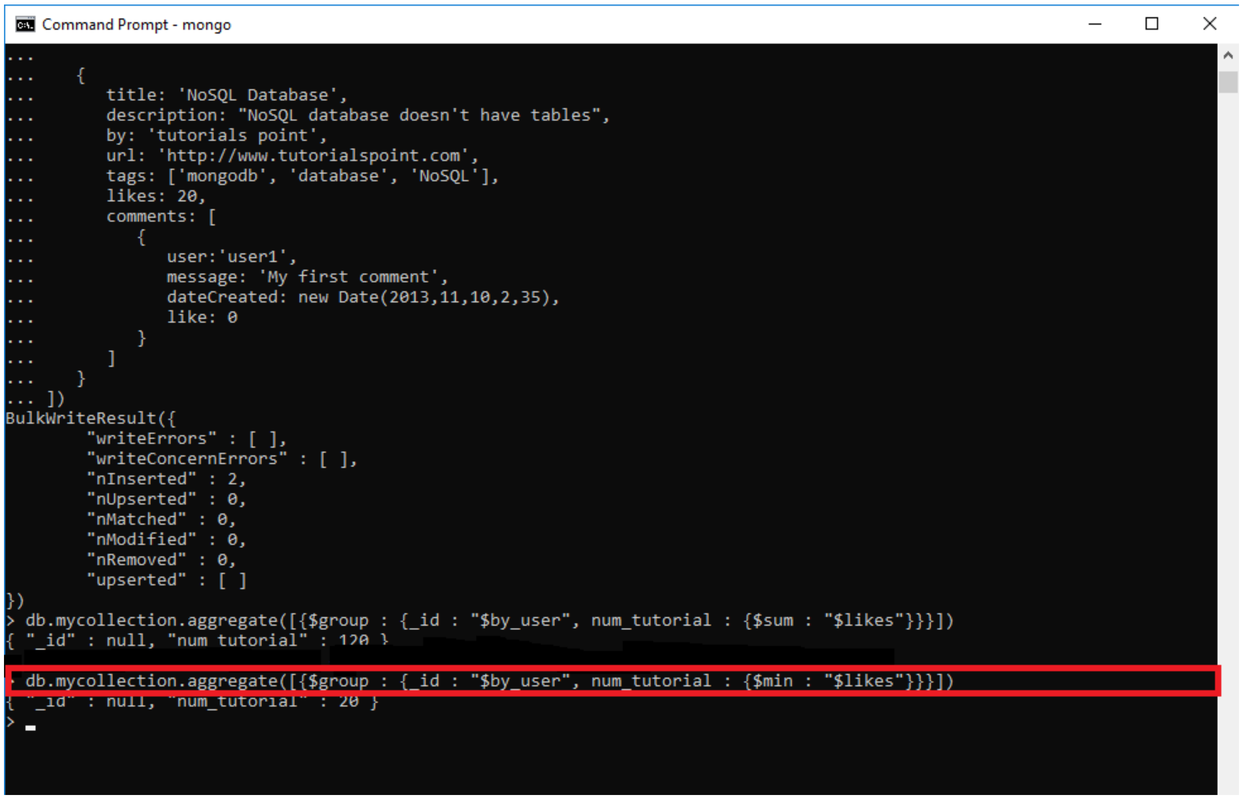


Select Command Prompt - mongo

```
> db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
> db.mycollection.find().pretty()
{
        "_id" : ObjectId("5b48b2793a55445a17fbd928"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no sql database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("5b48b2793a55445a17fbd929"),
        "title" : "NoSQL Database",
        "description" : "NoSQL database doesn't have tables",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 20,
        "comments" : [
                {
                        "user" : "user1",
                        "message" : "My first comment",
                        "dateCreated" : ISODate("2013-12-09T21:05:00Z"),
                        "like" : 0
                }
        ]
}
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 60 }
>
```

(iii) Min

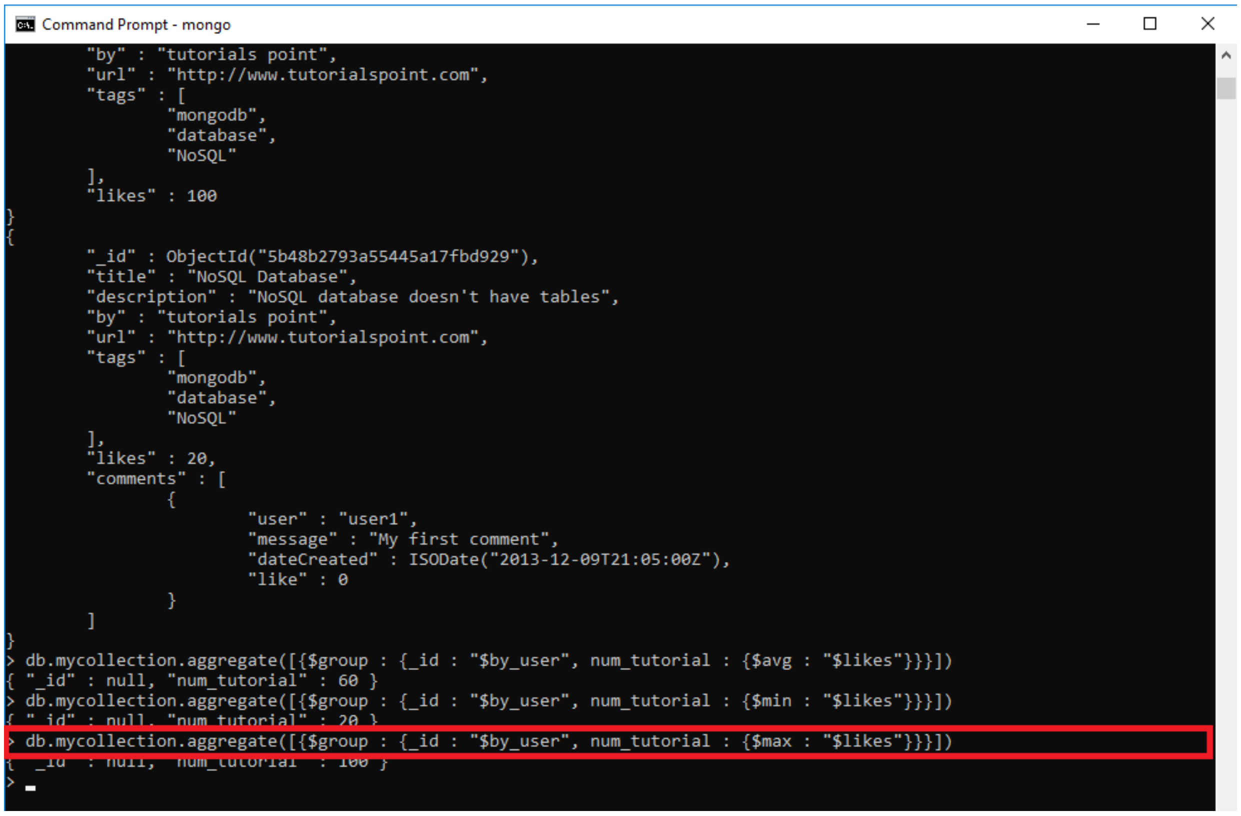
db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])



```
...
...    {
...       title: 'NoSQL Database',
...       description: "NoSQL database doesn't have tables",
...       by: 'tutorials point',
...       url: 'http://www.tutorialspoint.com',
...       tags: ['mongodb', 'database', 'NoSQL'],
...       likes: 20,
...       comments: [
...          {
...             user:'user1',
...             message: 'My first comment',
...             dateCreated: new Date(2013,11,10,2,35),
...             like: 0
...          }
...       ]
...    }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 120 }

> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 20 }
>
```

(iv) MAX

db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])



Command Prompt - mongo

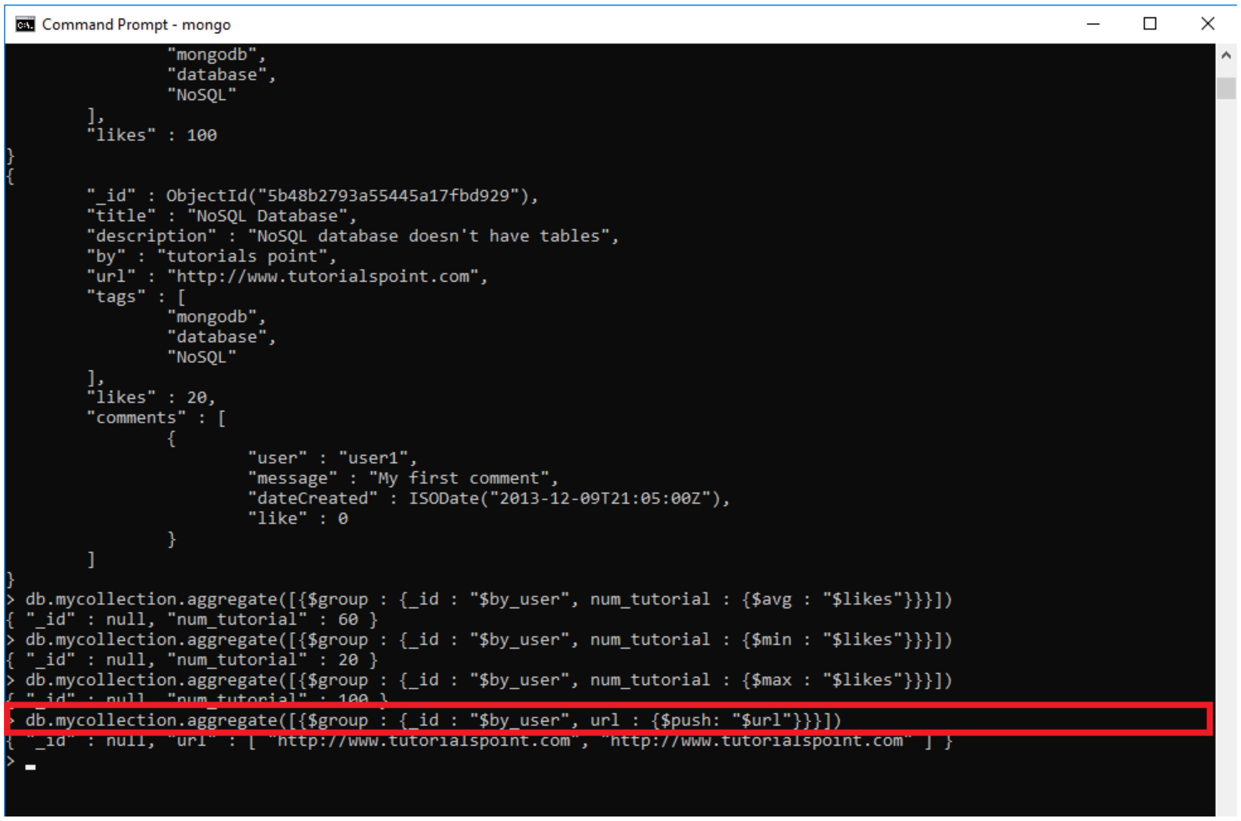
```
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("5b48b2793a55445a17fbd929"),
        "title" : "NoSQL Database",
        "description" : "NoSQL database doesn't have tables",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 20,
        "comments" : [
                {
                        "user" : "user1",
                        "message" : "My first comment",
                        "dateCreated" : ISODate("2013-12-09T21:05:00Z"),
                        "like" : 0
                }
        ]
}
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 60 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 20 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 100 }
> _
```

b) write a MangoDB query to push and addToSet expression

(i)Push

(Inserts the value to an array in the resulting document.)

db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])



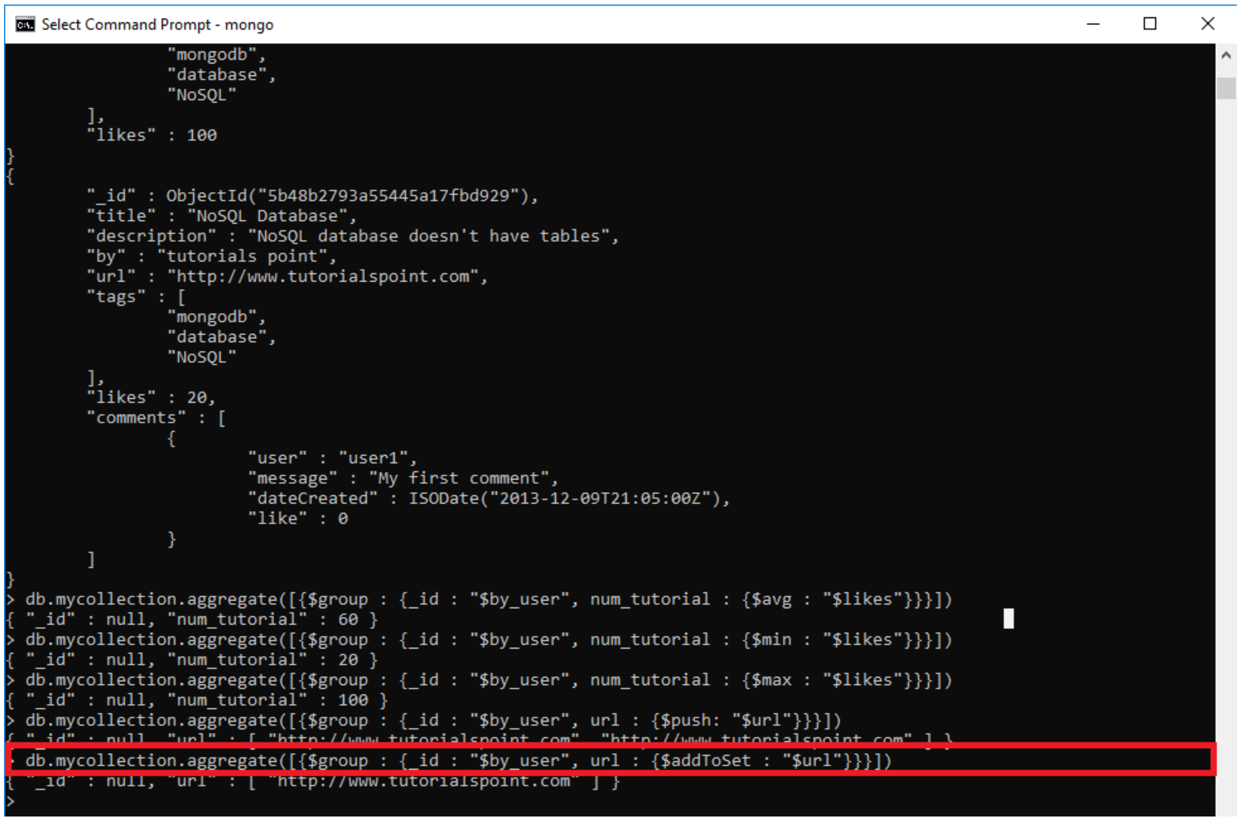
Command Prompt - mongo

```
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("5b48b2793a55445a17fbd929"),
        "title" : "NoSQL Database",
        "description" : "NoSQL database doesn't have tables",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 20,
        "comments" : [
                {
                        "user" : "user1",
                        "message" : "My first comment",
                        "dateCreated" : ISODate("2013-12-09T21:05:00Z"),
                        "like" : 0
                }
        ]
}
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 60 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 20 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 100 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
{ "_id" : null, "url" : [ "http://www.tutorialspoint.com", "http://www.tutorialspoint.com" ] }
> _
```

(ii) addToSet

db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])

(Inserts the value to an array in the resulting document but does not create duplicates.)

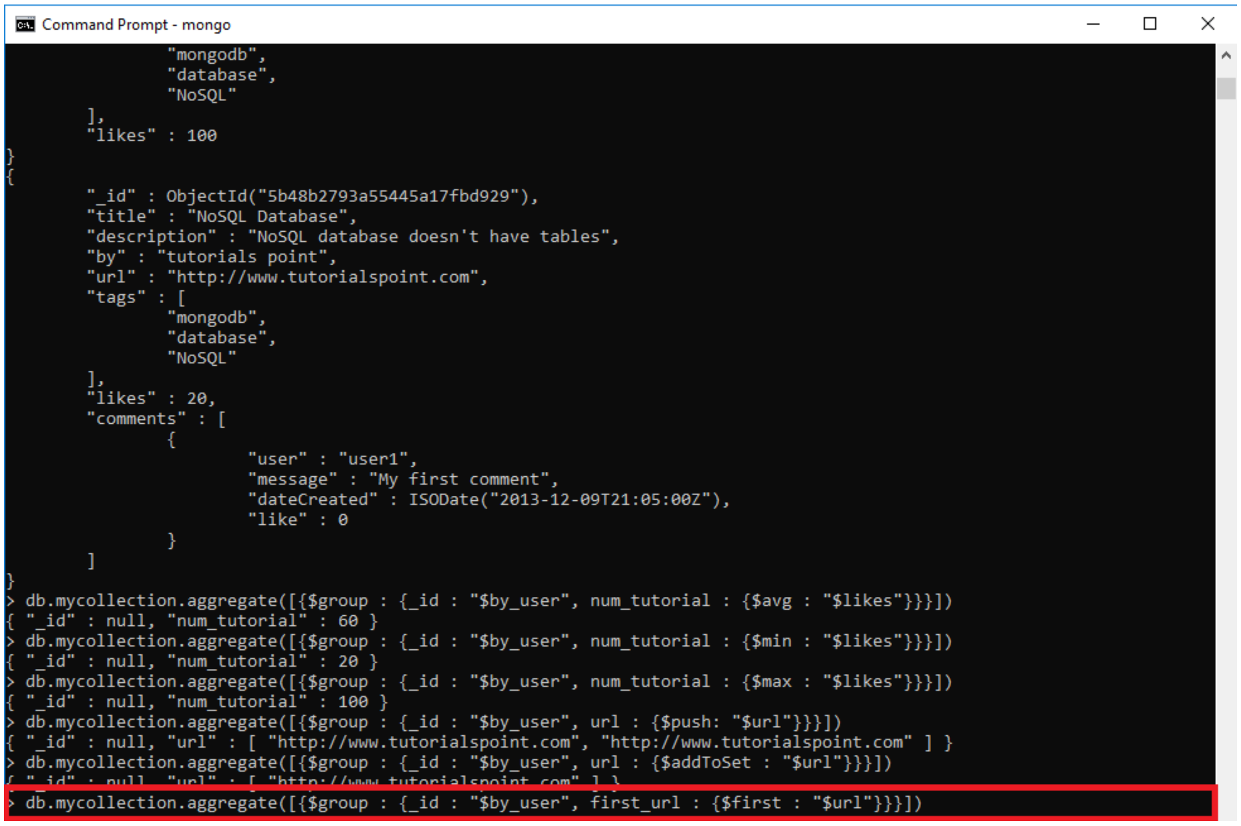


```
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("5b48b2793a55445a17fbd929"),
        "title" : "NoSQL Database",
        "description" : "NoSQL database doesn't have tables",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 20,
        "comments" : [
                {
                        "user" : "user1",
                        "message" : "My first comment",
                        "dateCreated" : ISODate("2013-12-09T21:05:00Z"),
                        "like" : 0
                }
        ]
}
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 60 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 20 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 100 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
{ "_id" : null, "url" : [ "http://www.tutorialspoint.com", "http://www.tutorialspoint.com" ] }
> db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])
{ "_id" : null, "url" : [ "http://www.tutorialspoint.com" ] }
>
```

c) Write a MongoDB query to use first and last expression.

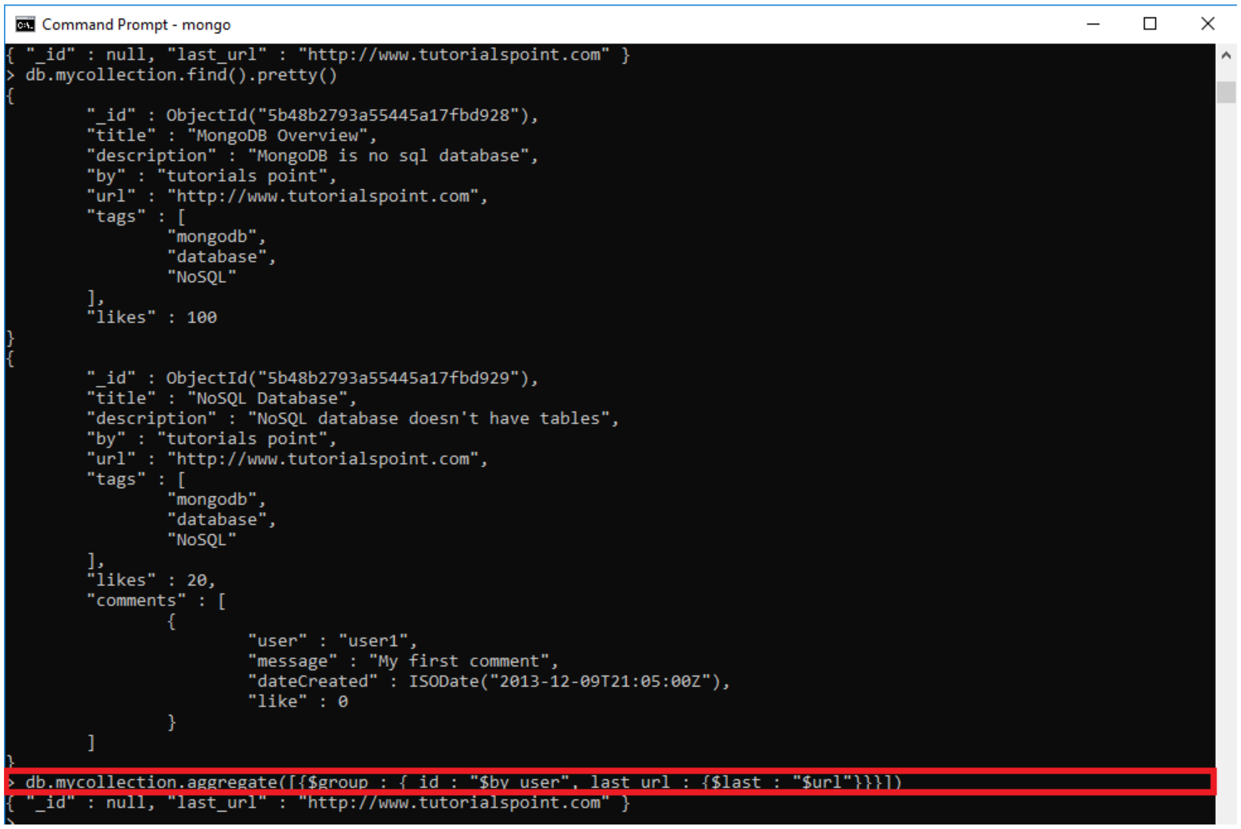
(i) First

db.mycollection.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])


Command Prompt - mongo

```
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("5b48b2793a55445a17fbd929"),
        "title" : "NoSQL Database",
        "description" : "NoSQL database doesn't have tables",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 20,
        "comments" : [
                {
                        "user" : "user1",
                        "message" : "My first comment",
                        "dateCreated" : ISODate("2013-12-09T21:05:00Z"),
                        "like" : 0
                }
        ]
}
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 60 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 20 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
{ "_id" : null, "num_tutorial" : 100 }
> db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
{ "_id" : null, "url" : [ "http://www.tutorialspoint.com", "http://www.tutorialspoint.com" ] }
> db.mycollection.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])
{ "_id" : null, "url" : [ "http://www.tutorialspoint.com" ] }
> db.mycollection.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
```

(ii) Last



```
{ "_id" : null, "last_url" : "http://www.tutorialspoint.com" }
> db.mycollection.find().pretty()
{
        "_id" : ObjectId("5b48b2793a55445a17fbd928"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no sql database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("5b48b2793a55445a17fbd929"),
        "title" : "NoSQL Database",
        "description" : "NoSQL database doesn't have tables",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 20,
        "comments" : [
                {
                        "user" : "user1",
                        "message" : "My first comment",
                        "dateCreated" : ISODate("2013-12-09T21:05:00Z"),
                        "like" : 0
                }
        ]
}
> db.mycollection.aggregate([{$group : { id : "$by user", last url : {$last : "$url"}}}])
{ "_id" : null, "last_url" : "http://www.tutorialspoint.com" }
```

Practical 4

Prac 4:

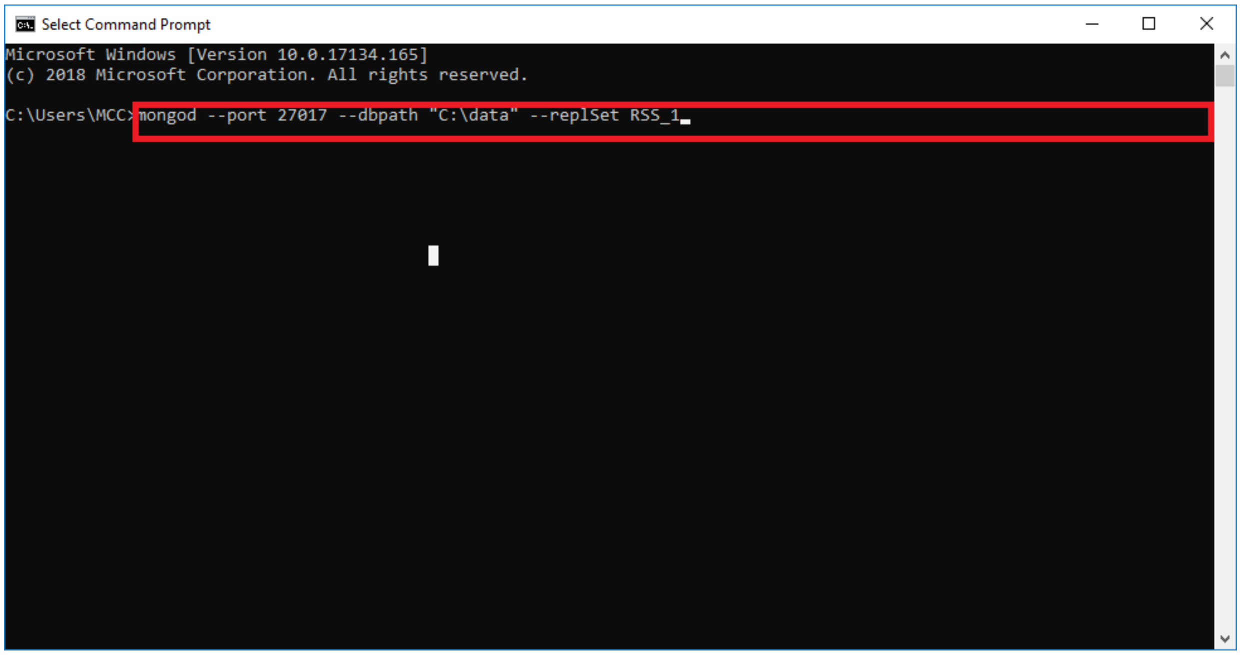
a)Write a MongoDB query to create a Replica of an existing database

MongoDB achieves replication by the use of replica set. A replica set is a group of mongod instances that host the same data set. In a replica, one node is primary node that receives all write operations. All other instances, such as secondaries, apply operations from the primary so that they have the same data set. Replica set can have only one primary node.

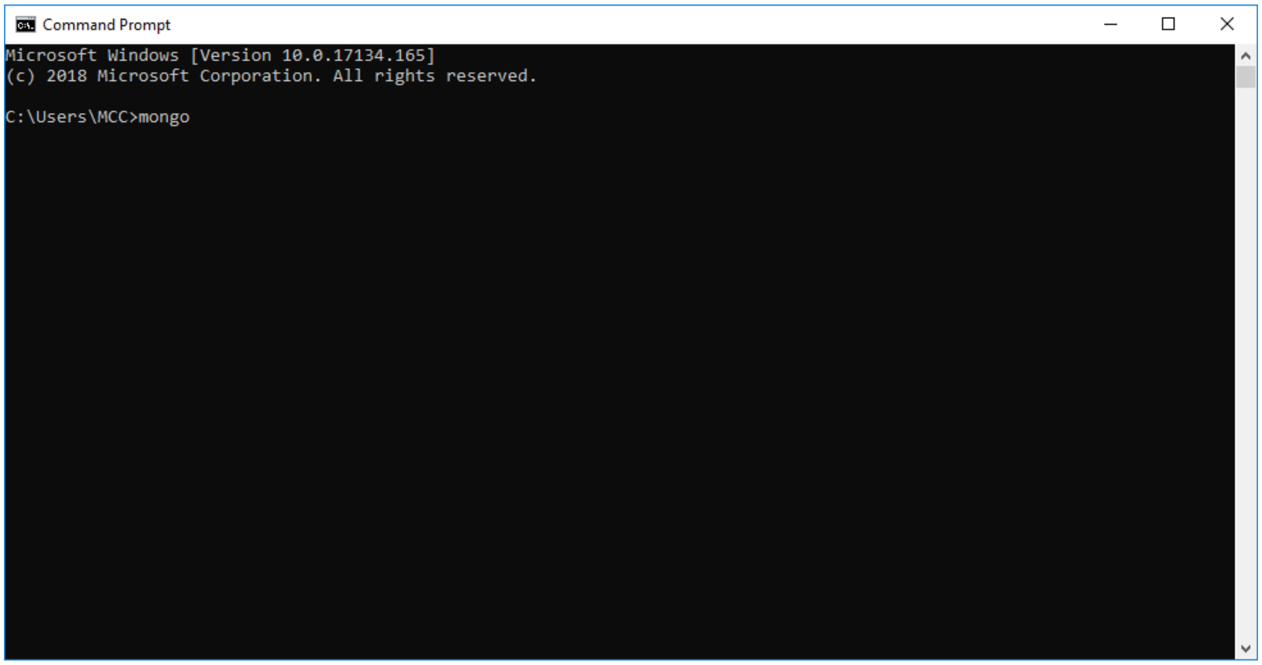
A replica set in MongoDB is a group of `mongod` processes that maintain the same data set.

```
mongod --port "PORT" --dbpath "YOUR_DB_DATA_PATH" --replSet "REPLICA_SET_INSTANCE_NAME"
```

mongod --port 27017 --dbpath "C:\data" --replSet RSS_1



```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MCC>mongod --port 27017 --dbpath "C:\data" --replSet RSS_1
```

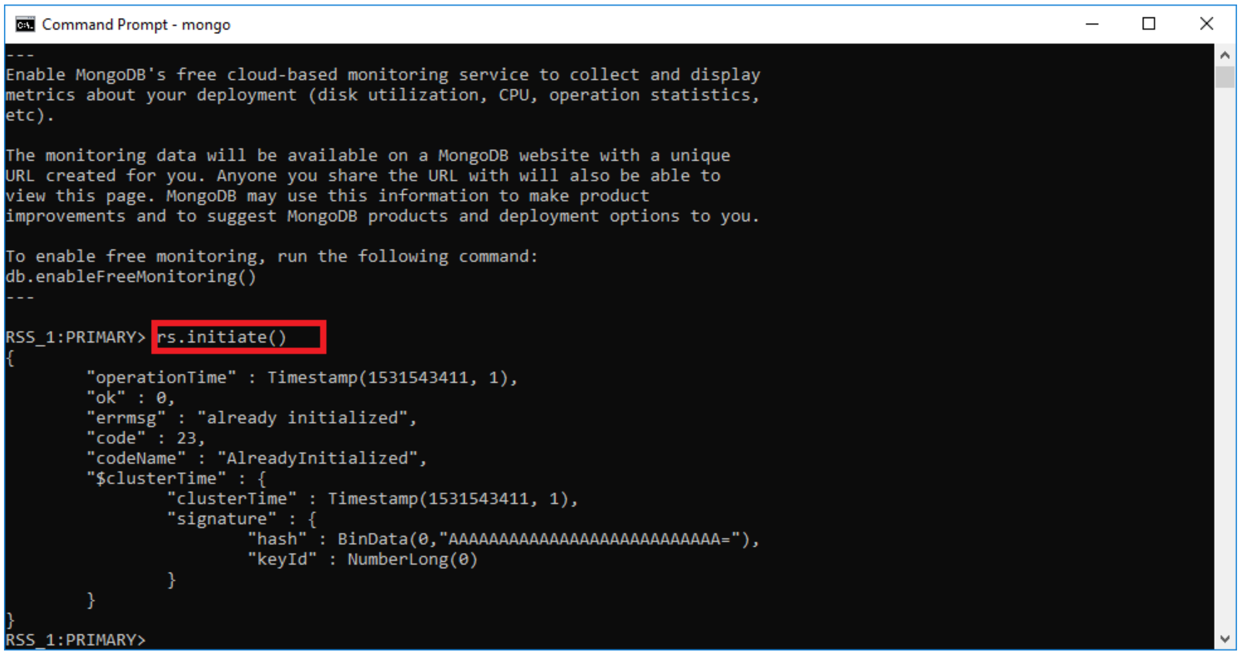


```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MCC>mongo
```

Open new cmd and type mongo to connect this mongod instance

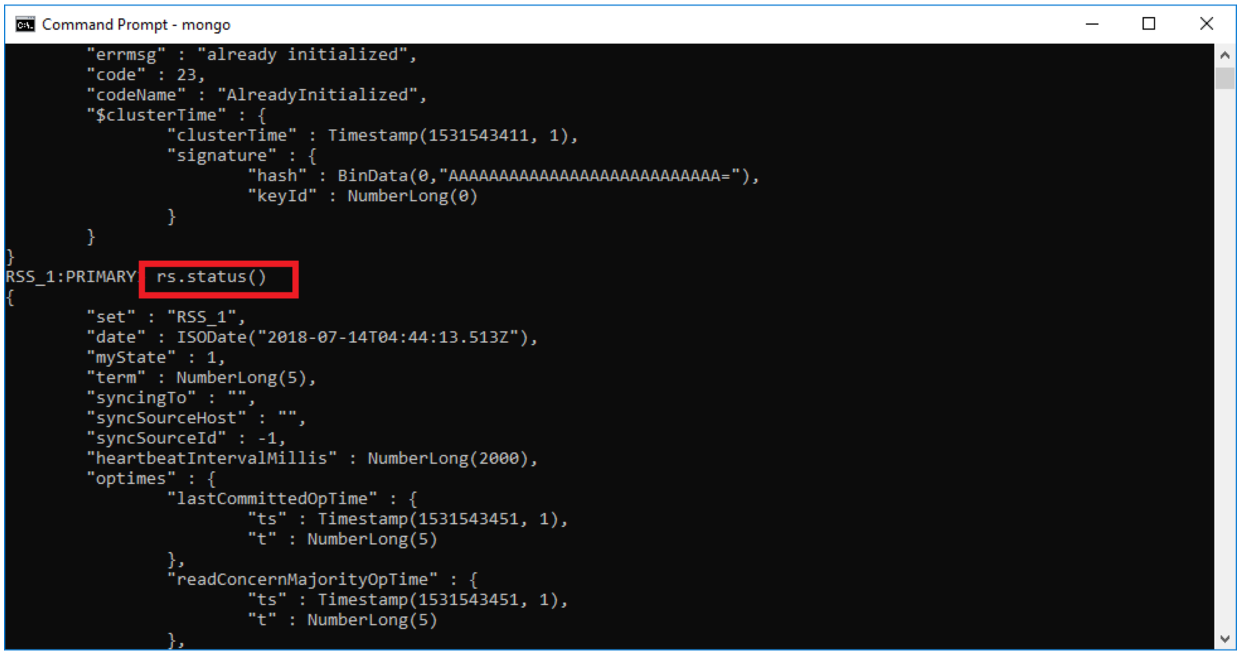
In Mongo client, issue the command rs.initiate() to initiate a new replica set.

rs.initiate()



```
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

RSS_1:PRIMARY> rs.initiate()
{
        "operationTime" : Timestamp(1531543411, 1),
        "ok" : 0,
        "errmsg" : "already initialized",
        "code" : 23,
        "codeName" : "AlreadyInitialized",
        "$clusterTime" : {
                "clusterTime" : Timestamp(1531543411, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
RSS_1:PRIMARY>
```

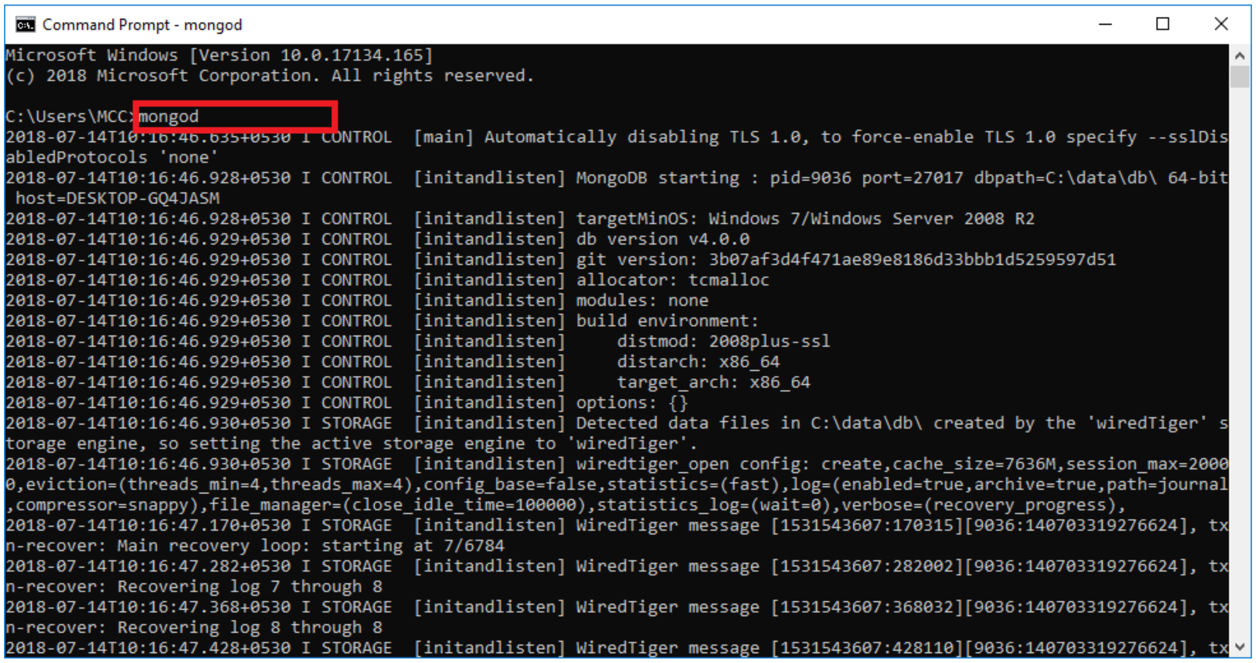
To check the status of replica set issue the command **rs.status()**
Rs.status()



```
        "errmsg" : "already initialized",
        "code" : 23,
        "codeName" : "AlreadyInitialized",
        "$clusterTime" : {
                "clusterTime" : Timestamp(1531543411, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
RSS_1:PRIMARY> rs.status()
{
        "set" : "RSS_1",
        "date" : ISODate("2018-07-14T04:44:13.513Z"),
        "myState" : 1,
        "term" : NumberLong(5),
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "heartbeatIntervalMillis" : NumberLong(2000),
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1531543451, 1),
                        "t" : NumberLong(5)
                },
                "readConcernMajorityOpTime" : {
                        "ts" : Timestamp(1531543451, 1),
                        "t" : NumberLong(5)
                },
```

(ii) Write a MongoDB query to create a backup of existing database

First start mongod on 1 cmd

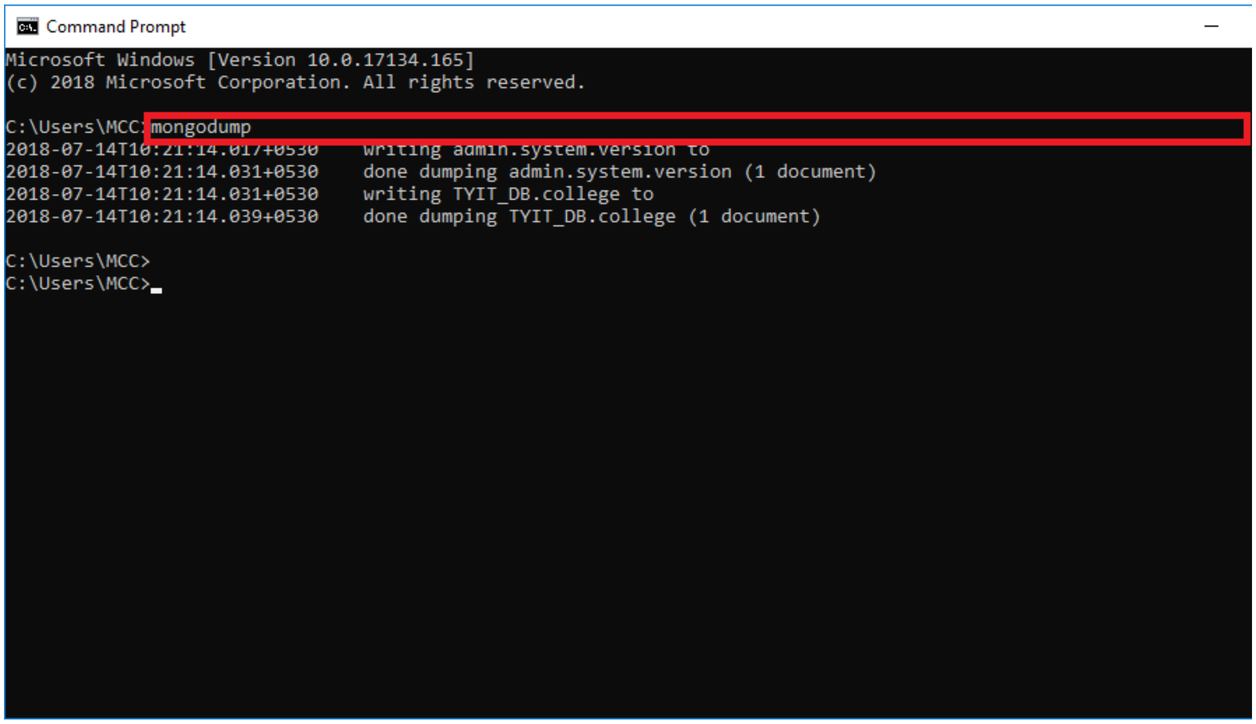


Command Prompt - mongod

```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MCC>mongod
2018-07-14T10:16:46.635+0530 I CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDis
abledProtocols 'none'
2018-07-14T10:16:46.928+0530 I CONTROL  [initandlisten] MongoDB starting : pid=9036 port=27017 dbpath=C:\data\db\ 64-bit
 host=DESKTOP-GQ4JASM
2018-07-14T10:16:46.928+0530 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten] db version v4.0.0
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten] git version: 3b07af3d4f471ae89e8186d33bbb1d5259597d51
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten] allocator: tcmalloc
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten] modules: none
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten] build environment:
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten]     distarch: x86_64
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten]     target_arch: x86_64
2018-07-14T10:16:46.929+0530 I CONTROL  [initandlisten] options: {}
2018-07-14T10:16:46.930+0530 I STORAGE  [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' s
torage engine, so setting the active storage engine to 'wiredTiger'.
2018-07-14T10:16:46.930+0530 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7636M,session_max=2000
0,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal
,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2018-07-14T10:16:47.170+0530 I STORAGE  [initandlisten] WiredTiger message [1531543607:170315][9036:140703319276624], tx
n-recover: Main recovery loop: starting at 7/6784
2018-07-14T10:16:47.282+0530 I STORAGE  [initandlisten] WiredTiger message [1531543607:282002][9036:140703319276624], tx
n-recover: Recovering log 7 through 8
2018-07-14T10:16:47.368+0530 I STORAGE  [initandlisten] WiredTiger message [1531543607:368032][9036:140703319276624], tx
n-recover: Recovering log 8 through 8
2018-07-14T10:16:47.428+0530 I STORAGE  [initandlisten] WiredTiger message [1531543607:428110][9036:140703319276624], tx
```

And then in second cmd type "mongodump"

mongodump

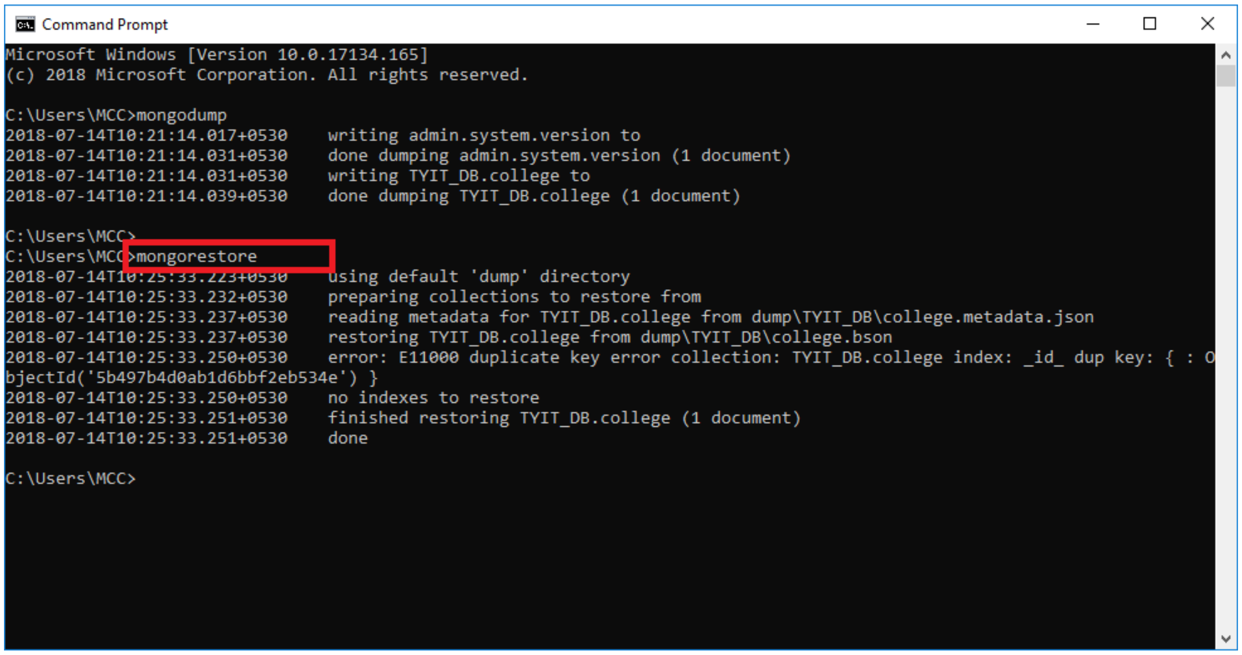


Command Prompt

```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MCC>mongodump
2018-07-14T10:21:14.017+0530    writing admin.system.version to
2018-07-14T10:21:14.031+0530    done dumping admin.system.version (1 document)
2018-07-14T10:21:14.031+0530    writing TYIT_DB.college to
2018-07-14T10:21:14.039+0530    done dumping TYIT_DB.college (1 document)

C:\Users\MCC>
C:\Users\MCC>_
```

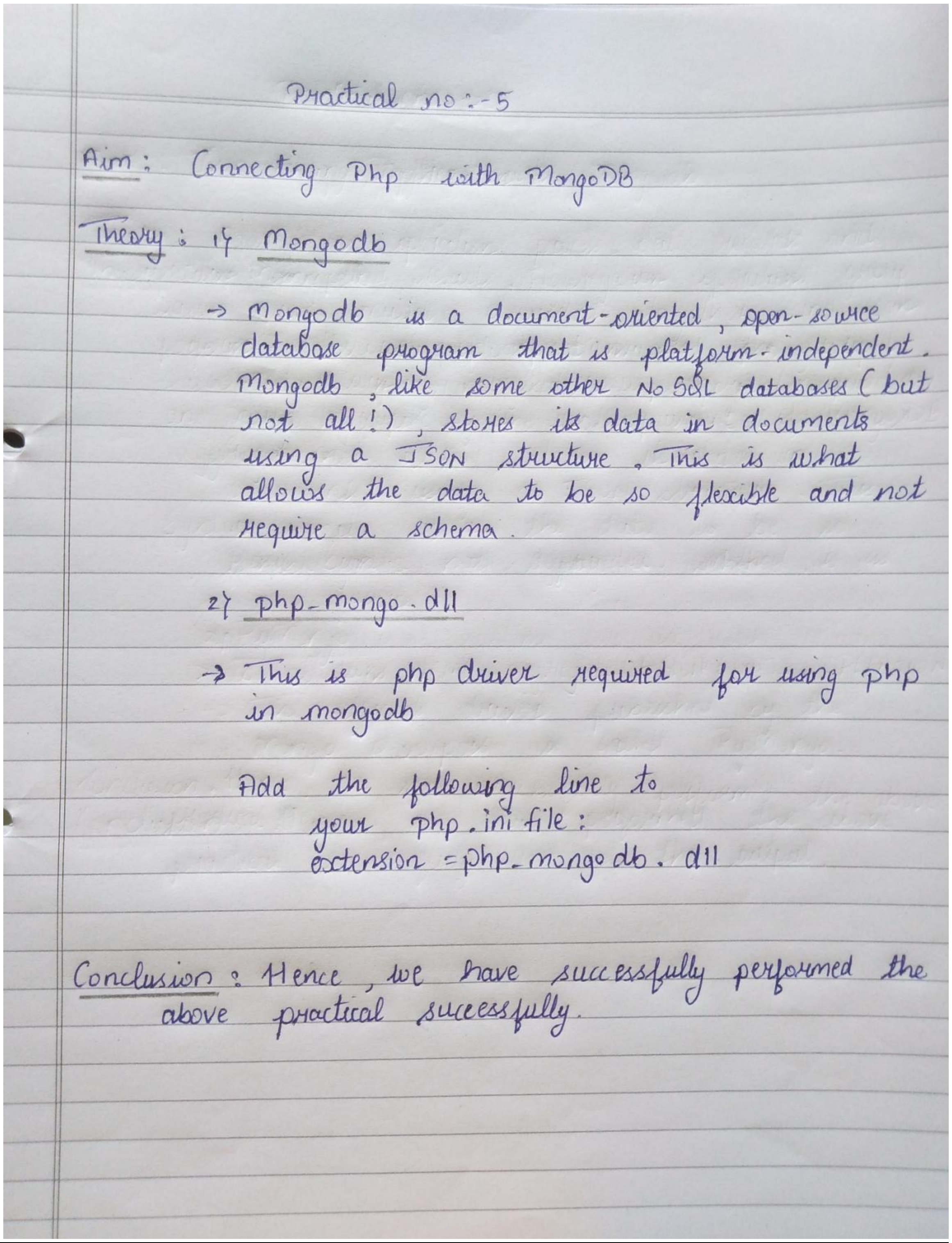
(iii) Write a MongoDB query to restore of existing

database mongorestore


Command Prompt

```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\MCC>mongodump
2018-07-14T10:21:14.017+0530    writing admin.system.version to
2018-07-14T10:21:14.031+0530    done dumping admin.system.version (1 document)
2018-07-14T10:21:14.031+0530    writing TYIT_DB.college to
2018-07-14T10:21:14.039+0530    done dumping TYIT_DB.college (1 document)

C:\Users\MCC>
C:\Users\MCC>mongorestore
2018-07-14T10:25:33.223+0530    using default 'dump' directory
2018-07-14T10:25:33.232+0530    preparing collections to restore from
2018-07-14T10:25:33.237+0530    reading metadata for TYIT_DB.college from dump\TYIT_DB\college.metadata.json
2018-07-14T10:25:33.237+0530    restoring TYIT_DB.college from dump\TYIT_DB\college.bson
2018-07-14T10:25:33.250+0530    error: E11000 duplicate key error collection: TYIT_DB.college index: _id_ dup key: { : O
bjectId('5b497b4d0ab1d6bbf2eb534e') }
2018-07-14T10:25:33.250+0530    no indexes to restore
2018-07-14T10:25:33.251+0530    finished restoring TYIT_DB.college (1 document)
2018-07-14T10:25:33.251+0530    done

C:\Users\MCC>
```

Practical 5



Practical no :- 5

Aim : Connecting Php with MongoDB

Theory : 1) Mongodb

→ Mongodb is a document-oriented, open-source database program that is platform-independent. Mongodb, like some other NoSQL databases (but not all!), stores its data in documents using a JSON structure. This is what allows the data to be so flexible and not require a schema.

2) php-mongo.dll

→ This is php driver required for using php in mongodb

Add the following line to your php.ini file :

```
extension =php_mongodb.dll
```

Conclusion : Hence, we have successfully performed the above practical successfully.

Connecting PHP and MongoDB and inserting, retrieving, updating and deleting MongoDB-PHP

Connect.php

```
<?php
  // connect to mongodb
  $m = new MongoClient();
  echo "Connection to database successfully";

  // select a database
  $db = $m->mydb;
  echo "Database mydb selected";
  $collection = $db->createCollection("myusers");
  echo "Collection created succsessfully";
?>
```

Insert.php

```
<?php
  // connect to mongodb
  $m = new MongoClient();
  echo "Connection to database successfully";
  echo "<br>";

  // select a database
  $db = $m->mydb;
  echo "Database mydb selected";
  echo "<br>";
  $collection = $db->myusers;
  echo "Collection selected succsessfully";
  echo "<br>";

  $user1 = array(
    "name" => "ABC",
    "age" => 30

  );

  $user2 = array(
    "name" => "XYZ",
    "age" => 35

  );

  $user3 = array(
    "name" => "PQR",
    "age" => 32
  );

  $collection->insert($user1);
  $collection->insert($user2);
  $collection->insert($user3);
  echo "Document inserted successfully";
?>
```

Update.php

```
<?php
  // connect to mongodb
  $m = new MongoClient();
  echo "Connection to database successfully";
  echo "<br>";

  // select a database
  $db = $m->mydb;
  echo "Database mydb selected";
  echo "<br>";
  $collection = $db->myusers;
  echo "Collection selected succsessfully";
  echo "<br>";

  // now update the document
  $collection->update(array("name"=>"PQR"),
    array('$set'=>array("name"=>"LMN")));
  echo "Document updated successfully";

?>
```

Delete.php

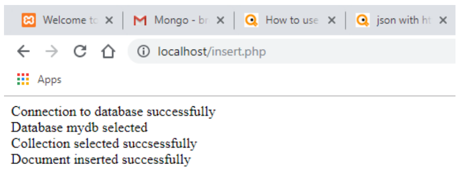
```
<?php
  // connect to mongodb
  $m = new MongoClient();
  echo "Connection to database successfully";
  echo "<br>";

  // select a database
  $db = $m->mydb;
  echo "Database mydb selected";
  echo "<br>";
  $collection = $db->myusers;
  echo "Collection selected succsessfully";
  echo "<br>";

  // now remove the document
  $collection->remove(array("name"=>"LMN"));
  echo "Documents deleted successfully";
?>
```

Retrieve.php

```
<?php
  // connect to mongodb
  $m = new MongoClient();
  echo "Connection to database successfully";

  // select a database
  $db = $m->mydb;
  echo "Database mydb selected";

  //select collection
  $collection = $db->myusers;
  echo "Collection selected succsessfully";

  $cursor = $collection->find();
  // iterate cursor to display title of documents

  foreach ($cursor as $user) {
    echo "<br>";

    echo $user["name"], ": ", $user["age"]."<br>";
   // echo $document["title"] . "\n";
  }
?>
```

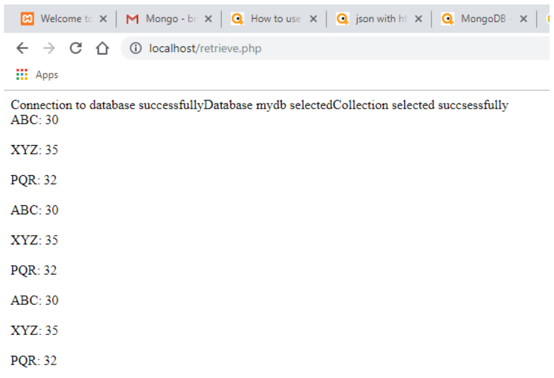
OUTPUT:

After insert.php



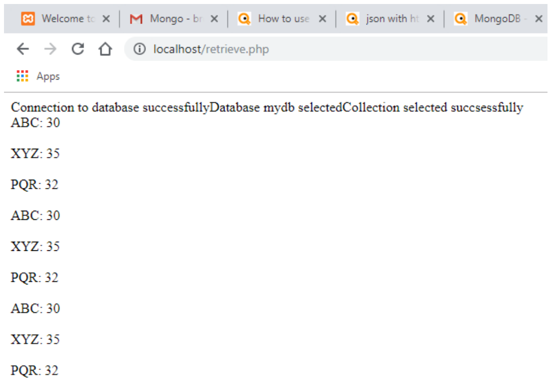
```
Connection to database successfully
Database mydb selected
Collection selected succsessfully
Document inserted successfully
```

After insert, retrieve data



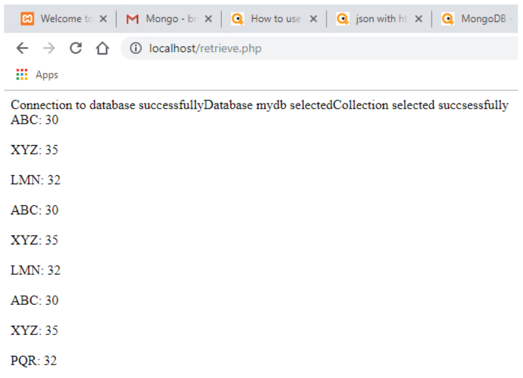
```
Connection to database successfullyDatabase mydb selectedCollection selected succsessfully
ABC: 30

XYZ: 35

PQR: 32

ABC: 30

XYZ: 35

PQR: 32

ABC: 30

XYZ: 35

PQR: 32
```

Then Update the document



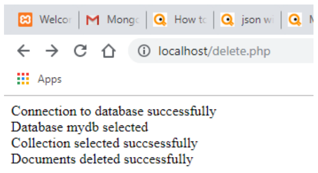
```
Connection to database successfullyDatabase mydb selectedCollection selected succsessfully
ABC: 30

XYZ: 35

PQR: 32

ABC: 30

XYZ: 35

PQR: 32

ABC: 30

XYZ: 35

PQR: 32
```

After Update, retrieve data



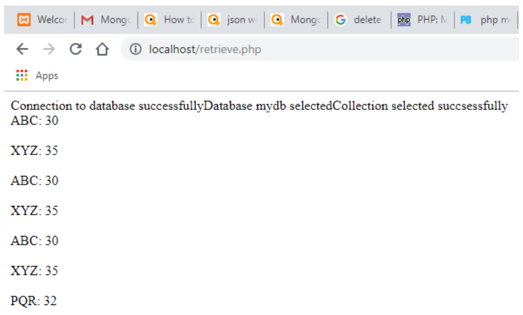
```
Connection to database successfullyDatabase mydb selectedCollection selected succsessfully
ABC: 30

XYZ: 35

LMN: 32

ABC: 30

XYZ: 35

LMN: 32

ABC: 30

XYZ: 35

PQR: 32
```

Then delete the document



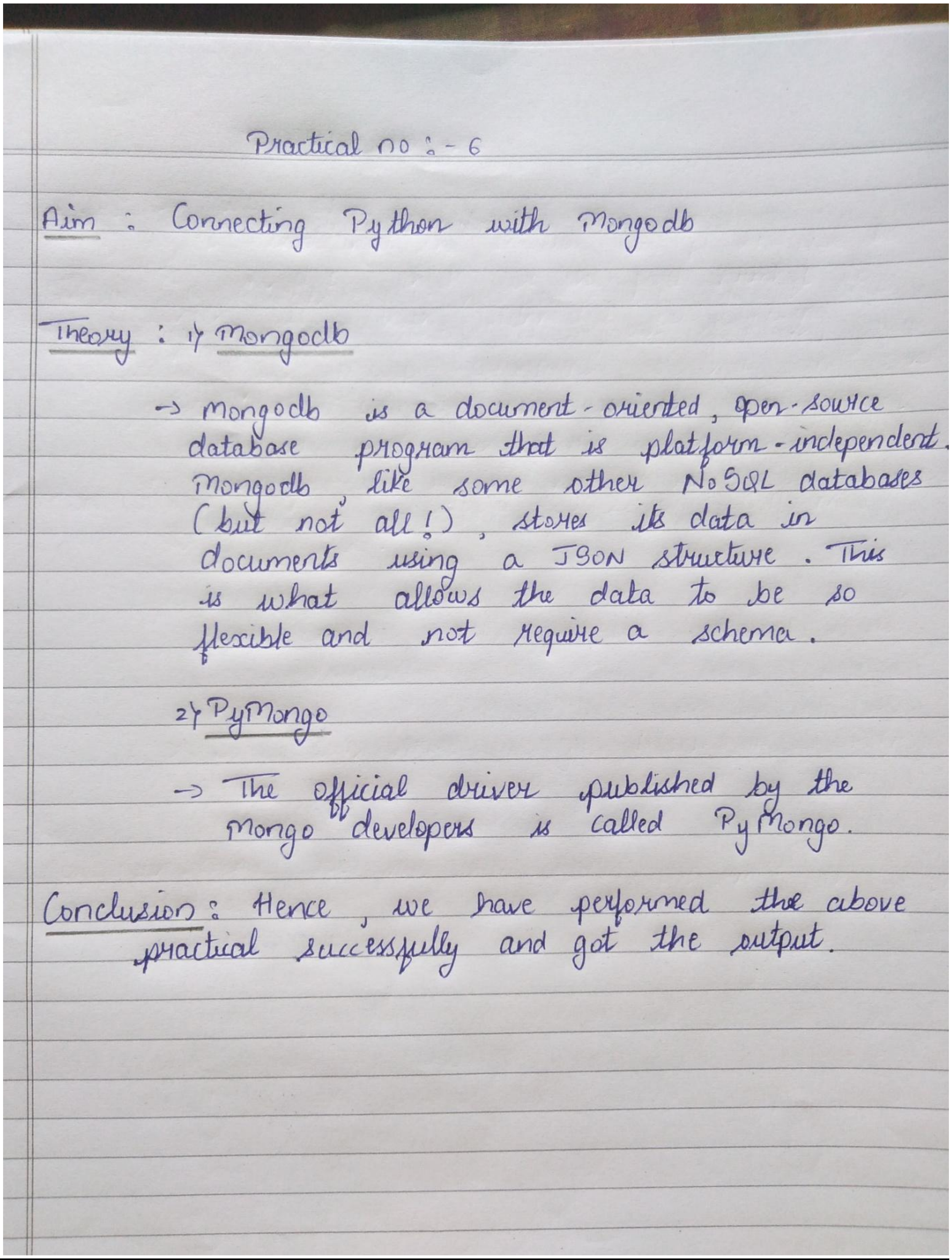
```
Connection to database successfully
Database mydb selected
Collection selected succsessfully
Documents deleted successfully
```

After Delete, retrieve data



```
Connection to database successfullyDatabase mydb selectedCollection selected succsessfully
ABC: 30

XYZ: 35

ABC: 30

XYZ: 35

ABC: 30

XYZ: 35

PQR: 32
```

Practical 6



Practical no :- 6

Aim : Connecting Python with Mongodb

Theory : 1) Mongodb

-> mongodb is a document - oriented, open - source database program that is platform - independent. Mongodb, like some other NoSQL databases (but not all!), stores its data in documents using a JSON structure. This is what allows the data to be so flexible and not require a schema.

2) PyMongo

-> The official driver published by the Mongo developers is called PyMongo.

Conclusion : Hence, we have performed the above practical successfully and got the output.

Connecting Python with MongoDB and inserting, retrieving, updating and deleting

```
>>> import pymongo
```

Making a Connection with MongoClient

```
>>> from pymongo import MongoClient
>>> client = MongoClient()
```

Getting a Database

```
>>> db = client.studentdb
```

Insert a document

```
>>> student1 = {"name": "Arun","rollno": 1}
>>> students = db.students
>>> students_id = students.insert(student1)
>>> students_id
ObjectId('548c02cd838d1f11b0d17d52')
```

Add two more records :

```
>>> student2 = {"name": "David","rollno": 2}
>>> student3 = {"name": "Shekhar","rollno": 3}

>>> students = db.students
>>> students_id = students.insert(student2)
>>> students_id = students.insert(student3)
```

Find a document

```
>>> students = db.students
>>> students.find_one()
{ '_id': ObjectId('548c02cd838d1f11b0d17d52'), 'name': 'Arun', 'rollno': 1}
```

Find a specific document :

```
>>> students = db.students
>>> students.find_one({"name": "Shekhar"})
{'_id': ObjectId('548c058a838d1f11b0d17d54'), 'name': 'Shekhar', 'rollno': 3}
```

Multiple documents Query

```
>>> students = db.students
>>> for student in students.find():
                                        student
{'_id': ObjectId('548c02cd838d1f11b0d17d52'), 'name': 'Arun', 'rollno': 1}
{'_id': ObjectId('548c0584838d1f11b0d17d53'), 'name': 'David', 'rollno': 2}
 {'_id': ObjectId('548c058a838d1f11b0d17d54'), 'name': 'Shekhar', 'rollno': 3}
```

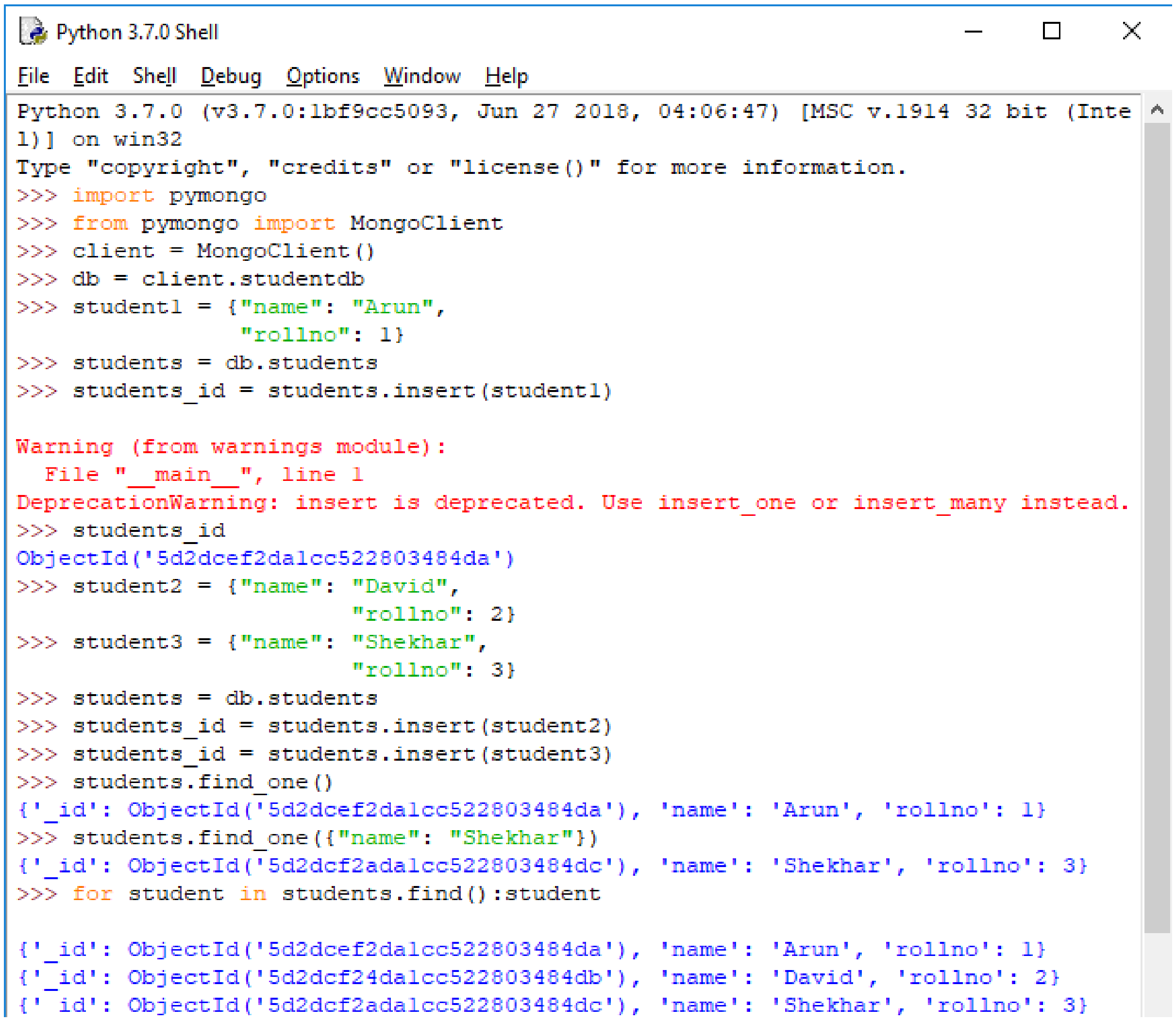
Update a specific document

```
>>> students.find_one({"name": "Shekhar"})
{'_id': ObjectId('548c058a838d1f11b0d17d54'), 'name': 'Shekhar', 'rollno': 3}

>>> students.update({"name": "Shekhar"}, {'$set':{'rollno': 12}})
{'updatedExisting': True, 'nModified': 1, 'ok': 1, 'n': 1}

>>> students.find_one({"name": "Shekhar"})
{'_id': ObjectId('548c058a838d1f11b0d17d54'), 'name': 'Shekhar', 'rollno': 12}
```

Remove a specific document

```
>>> students.remove({"rollno": 12})
{'ok': 1, 'n': 1}

>>> for student in students.find():
           student
             {'_id': ObjectId('548c02cd838d1f11b0d17d52'), 'name': 'Arun', 'rollno': 1}
            {'_id': ObjectId('548c0584838d1f11b0d17d53'), 'name': 'David', 'rollno': 2}
```

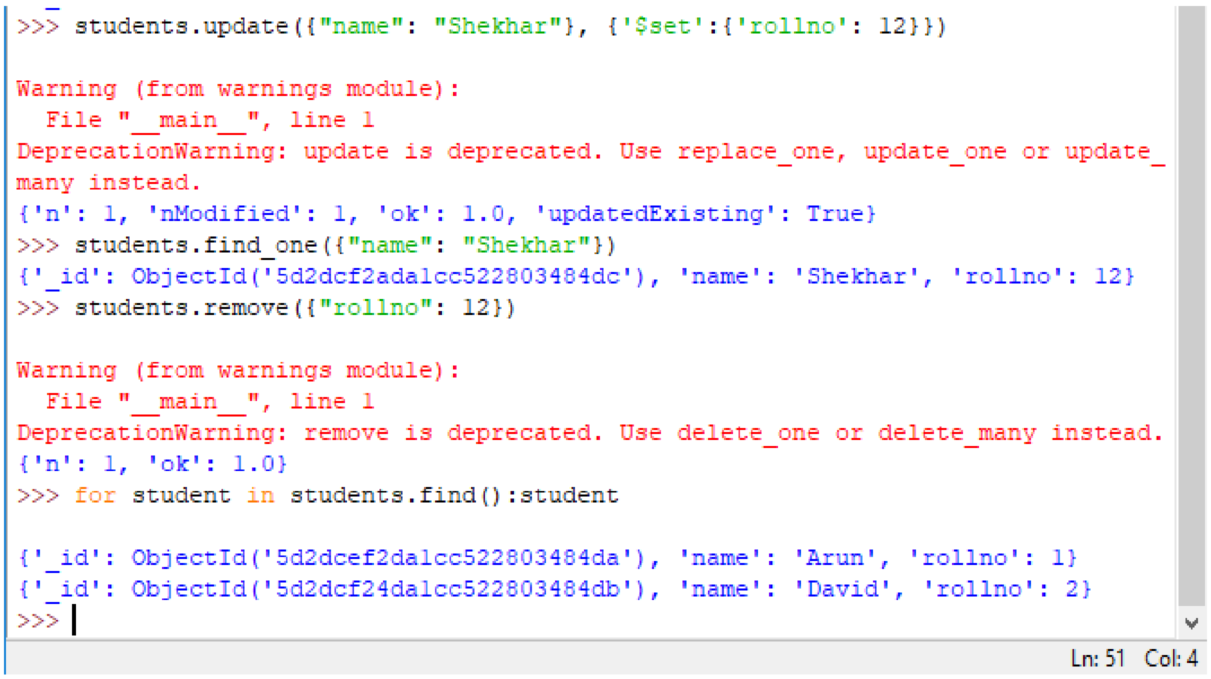
OUTPUT:



Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import pymongo
>>> from pymongo import MongoClient
>>> client = MongoClient()
>>> db = client.studentdb
>>> student1 = {"name": "Arun",
                "rollno": 1}
>>> students = db.students
>>> students_id = students.insert(student1)

Warning (from warnings module):
  File "__main__", line 1
DeprecationWarning: insert is deprecated. Use insert_one or insert_many instead.
>>> students_id
ObjectId('5d2dcef2da1cc522803484da')
>>> student2 = {"name": "David",
                        "rollno": 2}
>>> student3 = {"name": "Shekhar",
                        "rollno": 3}
>>> students = db.students
>>> students_id = students.insert(student2)
>>> students_id = students.insert(student3)
>>> students.find_one()
{'_id': ObjectId('5d2dcef2da1cc522803484da'), 'name': 'Arun', 'rollno': 1}
>>> students.find_one({"name": "Shekhar"})
{'_id': ObjectId('5d2dcf2ada1cc522803484dc'), 'name': 'Shekhar', 'rollno': 3}
>>> for student in students.find():student

{'_id': ObjectId('5d2dcef2da1cc522803484da'), 'name': 'Arun', 'rollno': 1}
{'_id': ObjectId('5d2dcf24da1cc522803484db'), 'name': 'David', 'rollno': 2}
{'_id': ObjectId('5d2dcf2ada1cc522803484dc'), 'name': 'Shekhar', 'rollno': 3}
```



```
>>> students.update({"name": "Shekhar"}, {'$set':{'rollno': 12}})

Warning (from warnings module):
  File "__main__", line 1
DeprecationWarning: update is deprecated. Use replace_one, update_one or update_
many instead.
{'n': 1, 'nModified': 1, 'ok': 1.0, 'updatedExisting': True}
>>> students.find_one({"name": "Shekhar"})
{'_id': ObjectId('5d2dcf2ada1cc522803484dc'), 'name': 'Shekhar', 'rollno': 12}
>>> students.remove({"rollno": 12})

Warning (from warnings module):
  File "__main__", line 1
DeprecationWarning: remove is deprecated. Use delete_one or delete_many instead.
{'n': 1, 'ok': 1.0}
>>> for student in students.find():student

{'_id': ObjectId('5d2dcef2da1cc522803484da'), 'name': 'Arun', 'rollno': 1}
{'_id': ObjectId('5d2dcf24da1cc522803484db'), 'name': 'David', 'rollno': 2}
>>> 
```

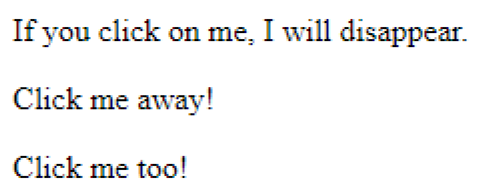
Practical 7

Practical 7A

1 JQuery Basic

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
</script>
</head>
<body>

<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>

</body>
</html>
```

OUTPUT:



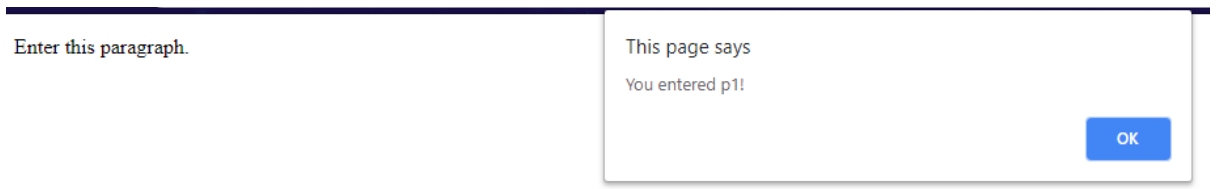
```
If you click on me, I will disappear.

Click me away!

Click me too!
```

2. Query events:

2.1 Mousecenter:

```
<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

  $("#p1").mouseenter(function(){

    alert("You entered p1!");

  });

});

</script>

</head>

<body>


<p id="p1">Enter this paragraph.</p>


</body>

</html>
```

OUTPUT:



Enter this paragraph.

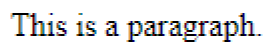
This page says

You entered p1!

2.2 Mouseup:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#p1").mouseup(function(){
    alert("Mouse up over p1!");
  });
});
</script>
</head>
<body>

<p id="p1">This is a paragraph.</p>

</body>
</html>
```

OUTPUT:

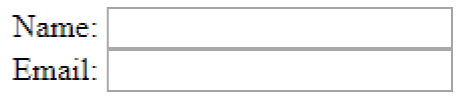


This is a paragraph.

2.3 Blur:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("input").focus(function(){
    $(this).css("background-color", "#cccccc");
  });
  $("input").blur(function(){
    $(this).css("background-color", "#ffffff");
  });
});
</script>
</head>
<body>

Name: <input type="text" name="fullname"><br>
Email: <input type="text" name="email">

</body>
</html>
```

OUTPUT:



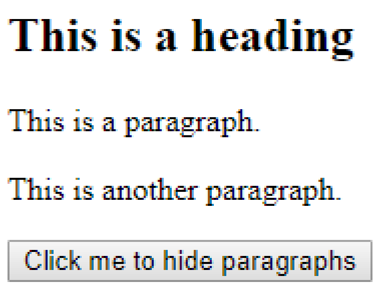
Practical 7B

1 JQuery Selector:

1.1 Control:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>
<body>

<h2>This is a heading</h2>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

<button>Click me to hide paragraphs</button>

</body>
</html>
```

OUTPUT:

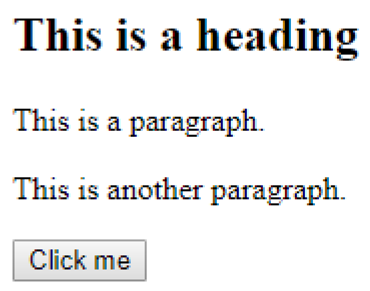


```
This is a heading

This is a paragraph.

This is another paragraph.
```

1.2 Class selector:

```
<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

  $("button").click(function(){

    $(".test").hide();

  });

});

</script>

</head>

<body>

<h2 class="test">This is a heading</h2>

<p class="test">This is a paragraph.</p>

<p>This is another paragraph.</p>

<button>Click me</button>

</body>

</html>
```

OUTPUT:

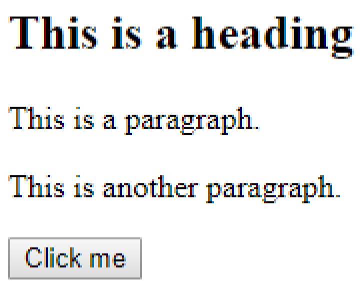


```
This is a heading

This is a paragraph.

This is another paragraph.
```

1.3 ID Selector:

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
   $("button").click(function(){
      $("#test").hide();
   });
});
</script>
</head>
<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p id="test">This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```

OUTPUT:



This is a heading

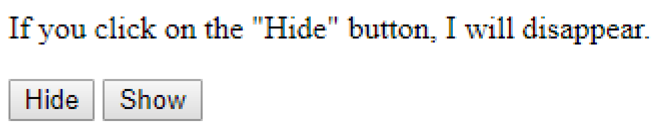
This is a paragraph.

This is another paragraph.

2. JQuery Hide/Show:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#hide").click(function(){
    $("p").hide();
  });
  $("#show").click(function(){
    $("p").show();
  });
});
</script>
</head>
<body>
<p>If you click on the "Hide" button, I will disappear.</p>
<button id="hide">Hide</button>
<button id="show">Show</button>
</body>
</html>
```

OUTPUT:


If you click on the "Hide" button, I will disappear.

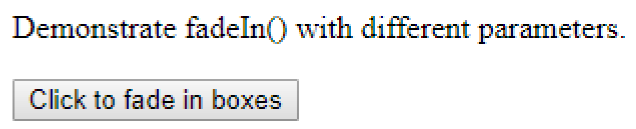
Practical 7C

1 JQuery Fading effect:

1.1 FadeIn:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
  });
});
</script>
</head>
<body>
<p>Demonstrate fadeIn() with different parameters.</p>
<button>Click to fade in boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>
</body>
</html>
```

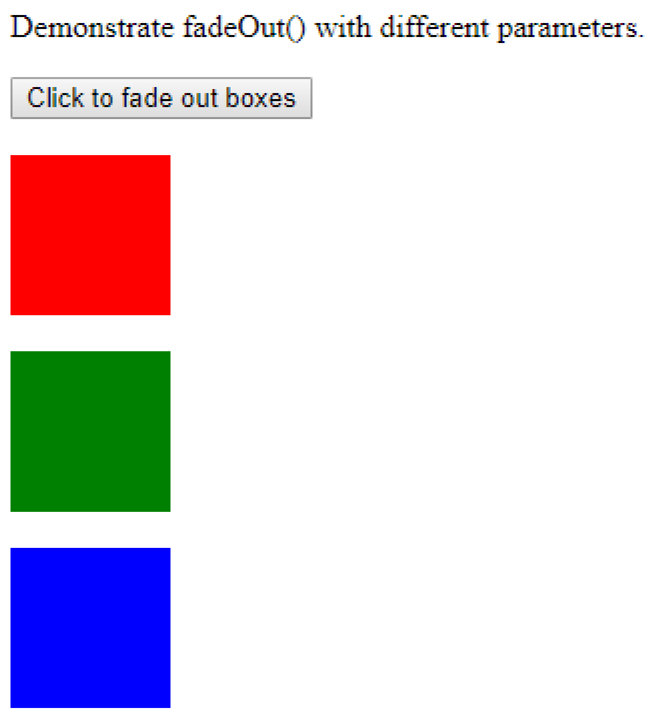
OUTPUT:


Demonstrate fadeIn() with different parameters.

1.2 FadeOut:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
  });
});
</script>
</head>
<body>
<p>Demonstrate fadeOut() with different parameters.</p>
<button>Click to fade out boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
</body>
</html>
```

OUTPUT:

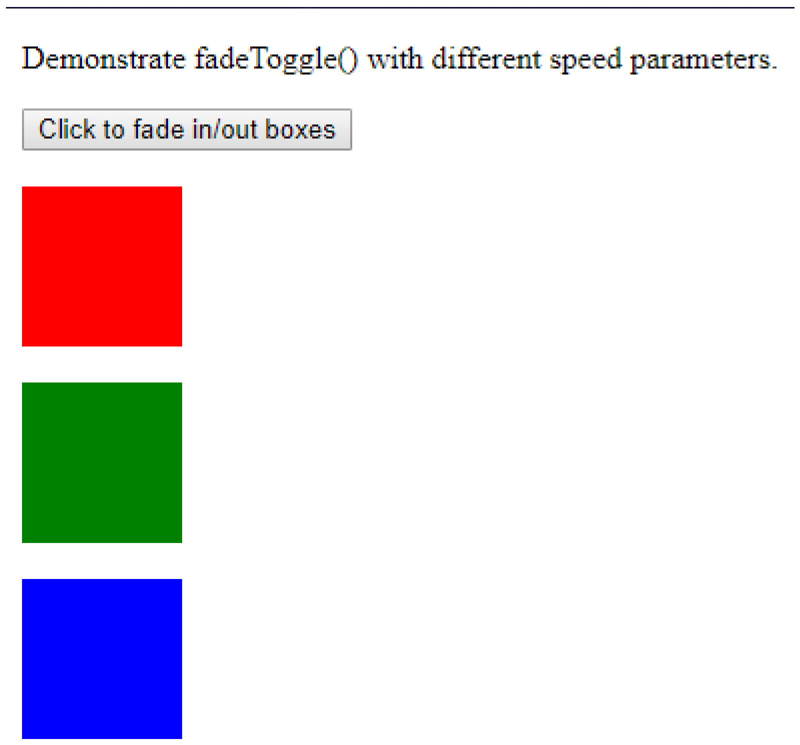


Demonstrate fadeOut() with different parameters.

1.3 FadeToggle:

```
<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

  $("button").click(function(){

    $("#div1").fadeToggle();

    $("#div2").fadeToggle("slow");

    $("#div3").fadeToggle(3000);

  });

});

</script>

</head>

<body>

<p>Demonstrate fadeToggle() with different speed parameters.</p>

<button>Click to fade in/out boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div>

<br>

<div id="div2" style="width:80px;height:80px;background-color:green;"></div>

<br>

<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

</body>

</html>
```

OUTPUT:



Demonstrate fadeToggle() with different speed parameters.

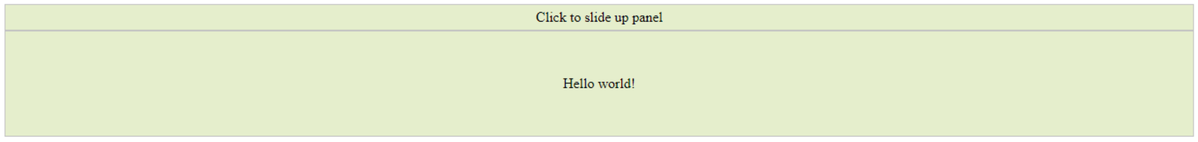
2. Sliding Effects:

2.1 SlideUp:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideUp("slow");
  });
});
</script>
<style>
#panel, #flip {
  padding: 5px;
  text-align: center;
  background-color: #e5eecc;
  border: solid 1px #c3c3c3;
}
#panel {
  padding: 50px;
}
</style>
</head>
<body>
<div id="flip">Click to slide up panel</div>
<div id="panel">Hello world!</div>
</body>
```

```
</html>
```

OUTPUT:



Click to slide up panel

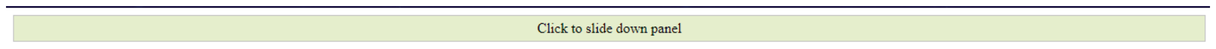
Hello world!

2.2 SlideDown:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideDown("slow");
  });
});
</script>
<style>
#panel, #flip {
  padding: 5px;
  text-align: center;
  background-color: #e5eecc;
  border: solid 1px #c3c3c3;
}
#panel {
  padding: 50px;
  display: none;
}
</style>
</head>
<body>
<div id="flip">Click to slide down panel</div>
<div id="panel">Hello world!</div>
</body>
```

```
</html>
```

OUTPUT:

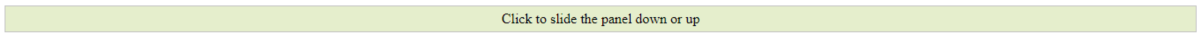


2.3 SlideToggle:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideToggle("slow");
  });
});
</script>

<style>
#panel, #flip {
  padding: 5px;
  text-align: center;
  background-color: #e5eecc;
  border: solid 1px #c3c3c3;
}
#panel {
  padding: 50px;
  display: none;
}
</style>
</head>
<body>
<div id="flip">Click to slide the panel down or up</div>
<div id="panel">Hello world!</div>
```

```
</body>

</html>
```

OUTPUT:


Click to slide the panel down or up

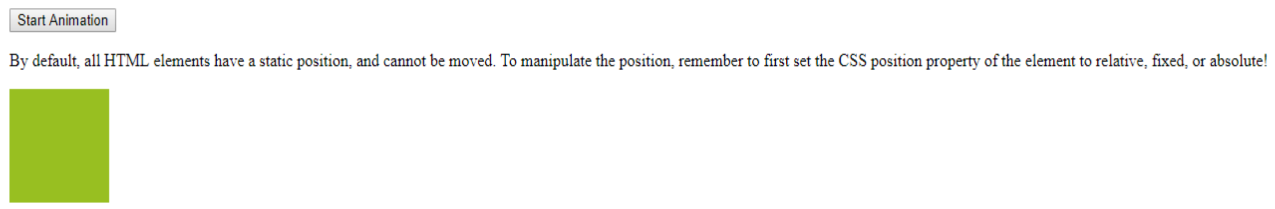
Practical 8

Practical 8A

Animation:

```
<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

  $("button").click(function(){

    $("div").animate({left: '250px'});

  });

});

</script>

</head>

<body>

<button>Start Animation</button>

<p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>

</html>
```

OUTPUT:

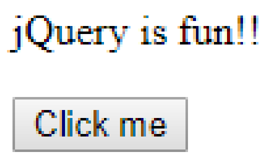


By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!

Chaining:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#p1").css("color", "red").slideUp(2000).slideDown(2000);
  });
});
</script>
</head>
<body>
<p id="p1">jQuery is fun!!</p>
<button>Click me</button>
</body>
</html>
```

OUTPUT:


jQuery is fun!!

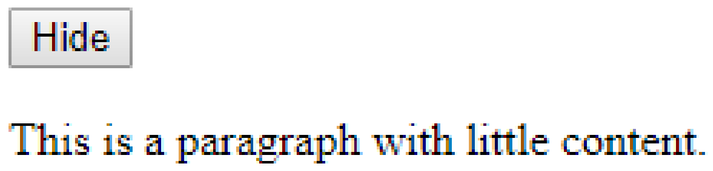
Practical 8B

CallBack:

1. **With:**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide("slow", function(){
      alert("The paragraph is now hidden");
    });
  });
});
</script>
</head>
<body>
<button>Hide</button>
<p>This is a paragraph with little content.</p>
</body>
</html>
```

OUTPUT:

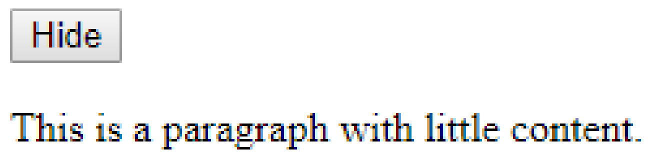


This is a paragraph with little content.

2. **Without:**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide(1000);
    alert("The paragraph is now hidden");
  });
});
</script>
</head>
<body>
<button>Hide</button>
<p>This is a paragraph with little content.</p>
</body>
</html>
```

OUTPUT:

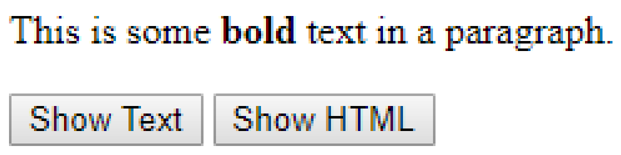


This is a paragraph with little content.

GET:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
   $("#btn1").click(function(){
      alert("Text: " + $("#test").text());
   });
   $("#btn2").click(function(){
      alert("HTML: " + $("#test").html());
   });
});
</script>
</head>
<body>
<p id="test">This is some <b>bold</b> text in a paragraph.</p>
<button id="btn1">Show Text</button>
<button id="btn2">Show HTML</button>
</body>
</html>
```

OUTPUT:

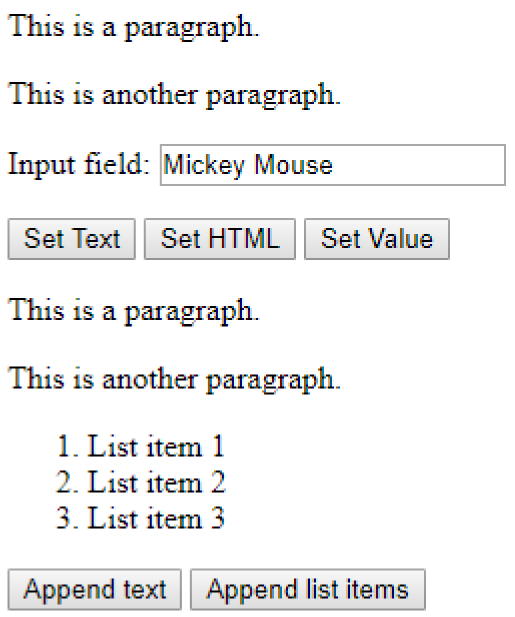

This is some **bold** text in a paragraph.

SET:

```
<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

  $("#btn1").click(function(){

    $("#test1").text("Hello world!");

  });

  $("#btn2").click(function(){

    $("#test2").html("<b>Hello world!</b>");

  });

  $("#btn3").click(function(){

    $("#test3").val("Dolly Duck");

  });

});

</script>

</head>

<body>

<p id="test1">This is a paragraph.</p>

<p id="test2">This is another paragraph.</p

<p>Input field: <input type="text" id="test3" value="Mickey Mouse"></p

<button id="btn1">Set Text</button>

<button id="btn2">Set HTML</button>

<button id="btn3">Set Value</button>

</body>

</html><!DOCTYPE html>

<html>
```

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        $("p").append(" <b>Appended text</b>.");
    });
    $("#btn2").click(function(){
        $("ol").append("<li>Appended item</li>");
    });
});
</script>
</head>
<body>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<ol>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>
<button id="btn1">Append text</button>
<button id="btn2">Append list items</button>
</body>
</html>
```

OUTPUT:


This is a paragraph.

This is another paragraph.

Input field: Mickey Mouse

This is a paragraph.

This is another paragraph.

1. List item 1
2. List item 2
3. List item 3

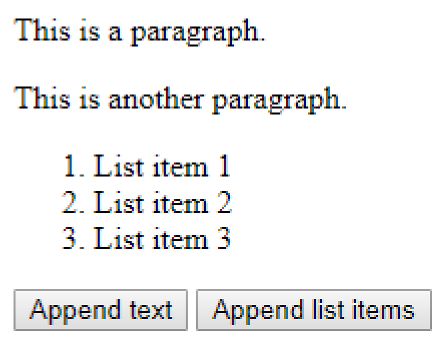
Practical 8C

ADD:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        $("p").append(" <b>Appended text</b>.");
    });
    $("#btn2").click(function(){
        $("ol").append("<li>Appended item</li>");
    });
});
</script>
</head>
<body>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<ol>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>
<button id="btn1">Append text</button>
<button id="btn2">Append list items</button>
</body>
```

```
</html>
```

OUTPUT:



This is a paragraph.

This is another paragraph.

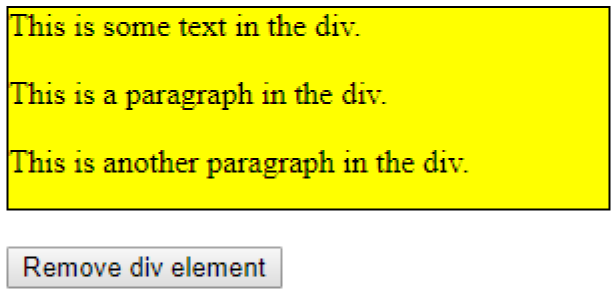
1. List item 1
2. List item 2
3. List item 3

REMOVE:

```
<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

  $("button").click(function(){

    $("#div1").remove();

  });

});

</script>

</head>

<body>

<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">

This is some text in the div.

<p>This is a paragraph in the div.</p>

<p>This is another paragraph in the div.</p>
```

```
</div>

<br>

<button>Remove div element</button>

</body>

</html>
```

OUTPUT:



```
This is some text in the div.

This is a paragraph in the div.

This is another paragraph in the div.
```

Practical 9

9.1 JSON

Create JSON file(artists.txt) on Localhost

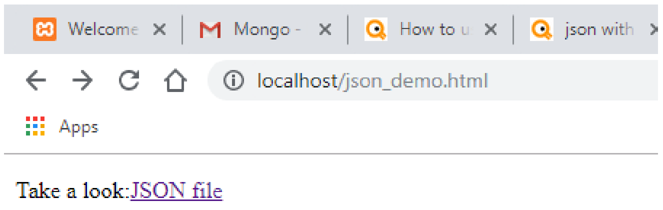
```
{
  "artists" : [
    {
      "artistname" : "Leonard Cohen",
      "born" : "1934"
      },
    {
      "artistname" : "Joe Satriani",
      "born" : "1956"
      },
    {
      "artistname" : "Snoop Dogg",
      "born" : "1971"
      }
  ]
}
```

Parsing JSON(above json file)

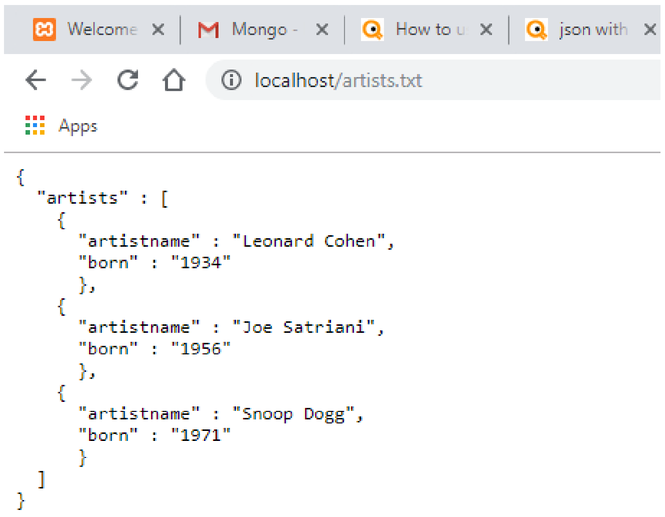
```
<!doctype html>
<title>Example</title>

<script>
// Store XMLHttpRequest and the JSON file location in variables
var xhr = new XMLHttpRequest();
var url = "http://localhost/artists.txt";

// Called whenever the readyState attribute changes
xhr.onreadystatechange = function() {

  // Check if fetch request is done
  if (xhr.readyState == 4 && xhr.status == 200) {

    // Parse the JSON string
    var jsonData = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML=jsonData.name;
    // Call the showArtists(), passing in the parsed JSON string
    //showArtists(jsonData);
```

```
 }
};

// Do the HTTP call using the url variable we specified above
xhr.open("GET", url, true);
xhr.send();


</script>
<p>Take a look:<a href="http://localhost/artists.txt">JSON file</a></p>
```

Run the above file on localhost



Take a look:JSON file



```
{
  "artists" : [
    {
      "artistname" : "Leonard Cohen",
      "born" : "1934"
      },
    {
      "artistname" : "Joe Satriani",
      "born" : "1956"
      },
    {
      "artistname" : "Snoop Dogg",
      "born" : "1971"
      }
  ]
}
```

9.2 Persisting JSON file in MongoDB

Switch to a MongoDB database

Here, our database is "myinfo".

```
> use myinfo

switch to db myinfo
```

Define a document for MongoDB database

```
> user1=({"user_id" : "ABCDBWN","password" :"ABCDBWN" ,"date_of_join" : "15/10/2010"
,"education" :"B.Sc" , "profession" : "DEVELOPER","interest" : "MUSIC”});
```

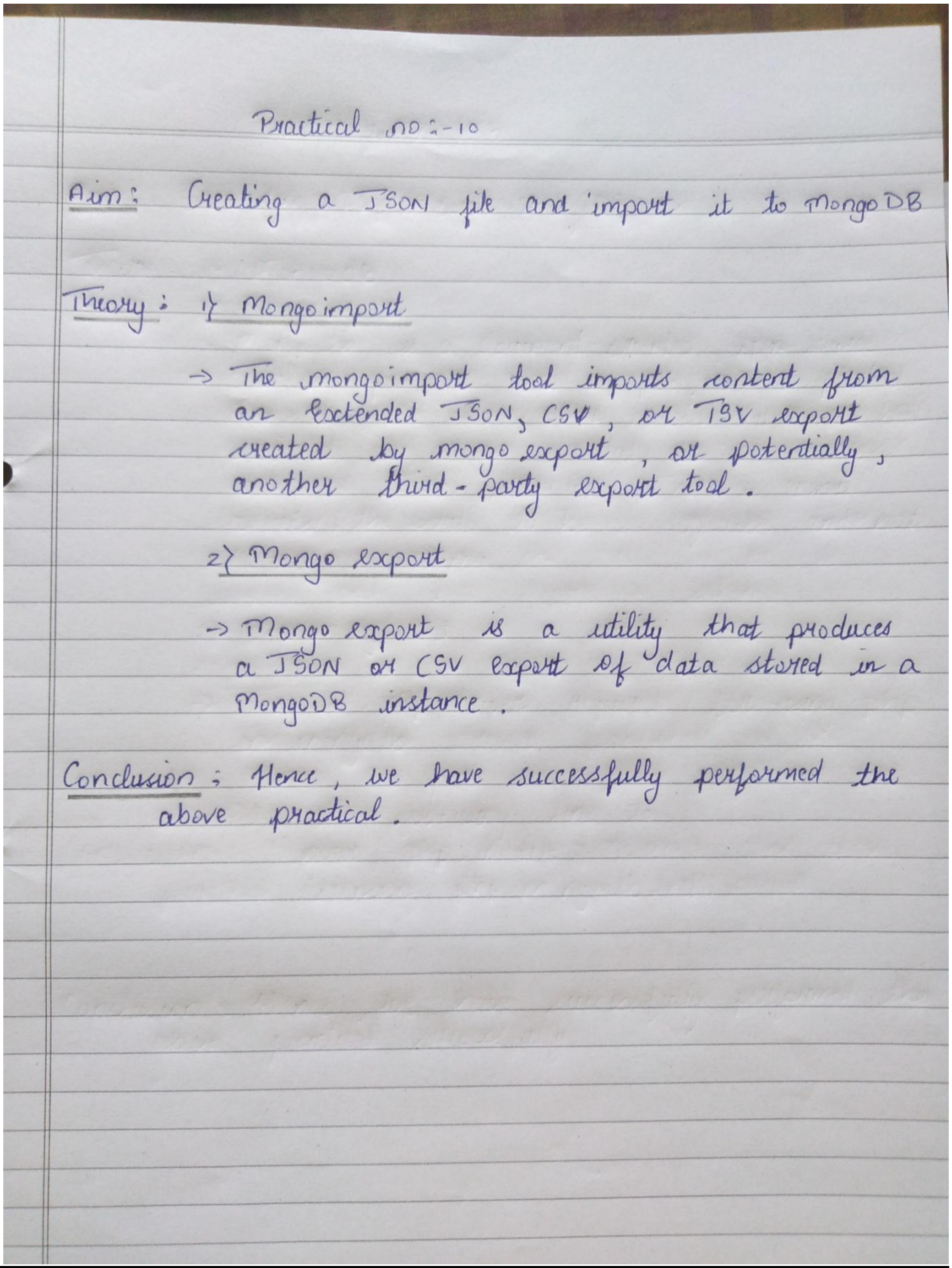
```
{
    "user_id" : "ABCDBWN",
    "password" : "ABCDBWN",
    "date_of_join" : "15/10/2010",
    "education" : "B.C.A.",
    "profession" : "DEVELOPER",
    "interest" : "MUSIC"

}
```

Insert a document into a collection

To save the above document into the collection "userdetails" under "myinfo" database the following command can be used –

```
> db.userdetails.insert(document)
```

Practical 10



Practical no:-10

Aim: Creating a JSON file and import it to MongoDB

Theory: 1) Mongoimport

→ The mongoimport tool imports content from an Extended JSON, CSV, or TSV export created by mongo export, or potentially, another third-party export tool.

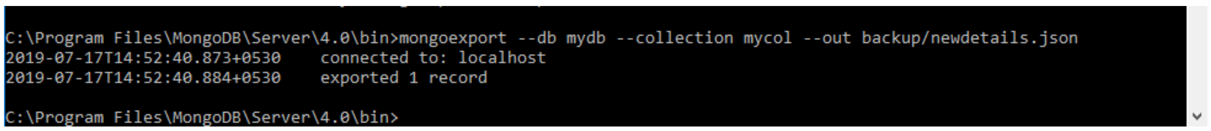
2) Mongo export

→ Mongo export is a utility that produces a JSON or CSV export of data stored in a MongoDB instance.

Conclusion: Hence, we have successfully performed the above practical.

Export MongoDB to JSON

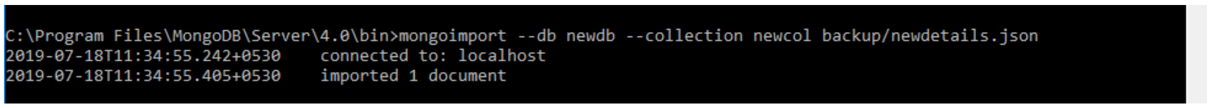
Mongoexport



```
C:\Program Files\MongoDB\Server\4.0\bin>mongoexport --db mydb --collection mycol --out backup/newdetails.json
2019-07-17T14:52:40.873+0530    connected to: localhost
2019-07-17T14:52:40.884+0530    exported 1 record

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

Import JSON file to MongoDB

Mongoimport



```
C:\Program Files\MongoDB\Server\4.0\bin>mongoimport --db newdb --collection newcol backup/newdetails.json
2019-07-18T11:34:55.242+0530    connected to: localhost
2019-07-18T11:34:55.405+0530    imported 1 document
```