S K SOMAIYA COLLEGE OF ARTS, SCIENCE & COMMERCE

Vidyavihar (East), Mumbai 400 077

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the experiments done in the subject of _____Next Generation Technology_____ At S. K. Somaiya College Of Arts, Science And Commerce by_____Rasika Sunil Kambli_____ Seat no. ____13_____ is partial fulfillment of B.Sc. IT degree (Semester- V) Examination for the academic year _____2020-2021_____.

Teacher Incharge: Prof. Rakhee Yadav

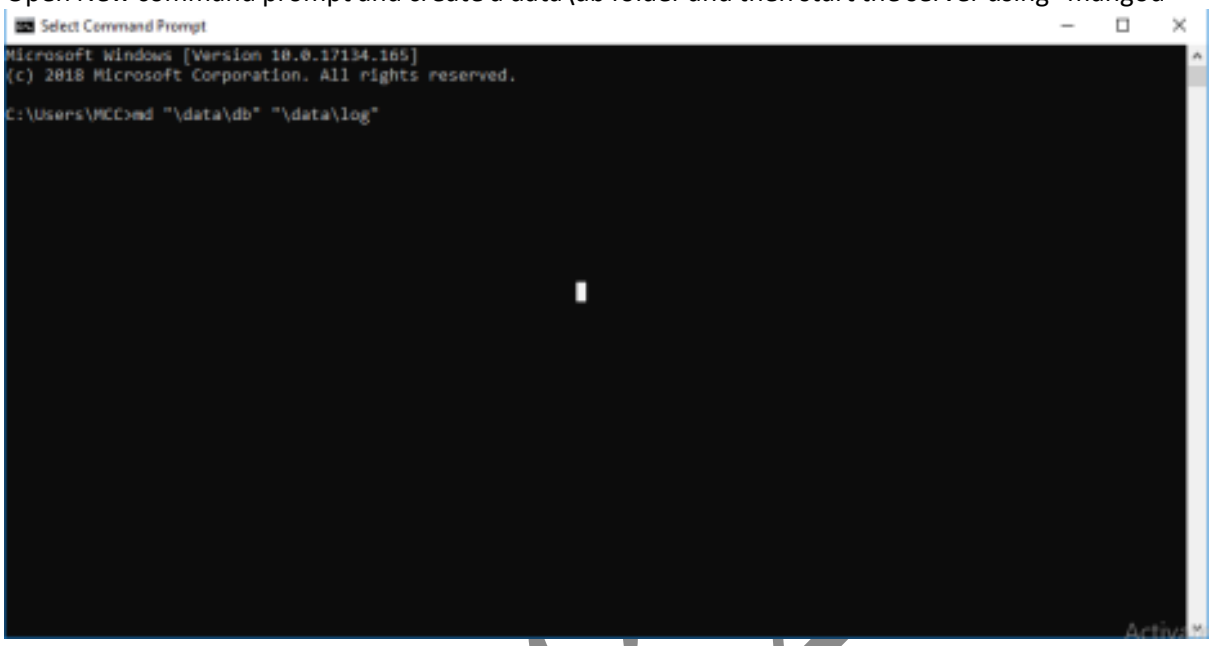
Name of Internal Examiner:

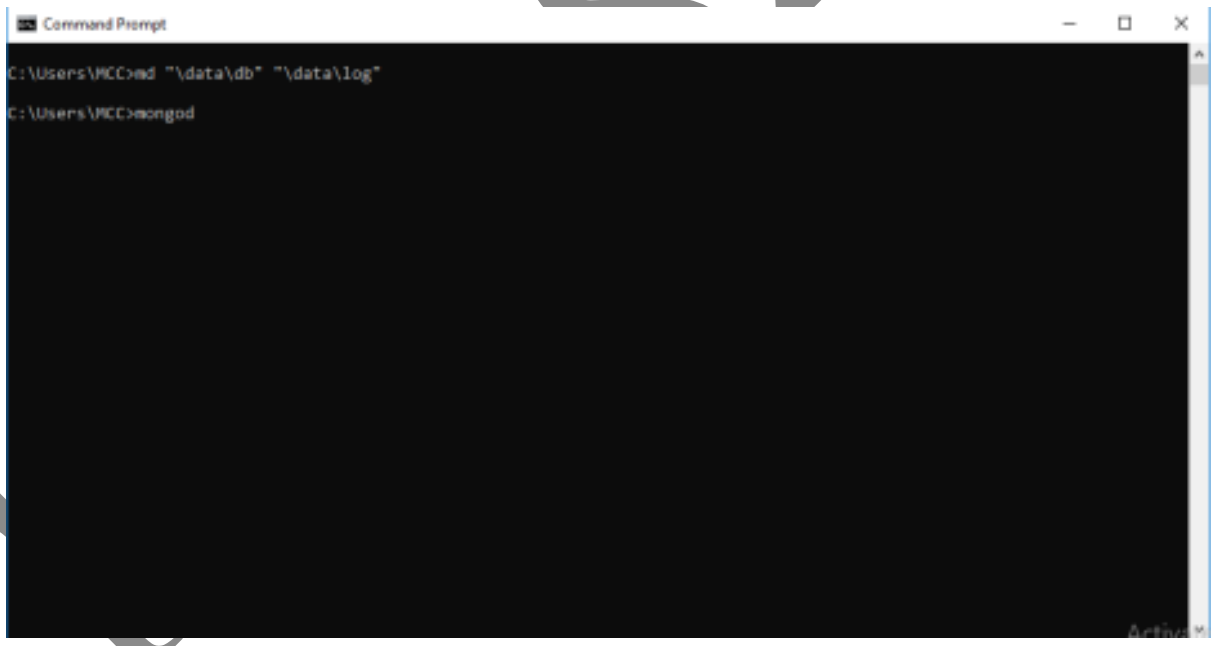
Name of Co-ordinator: Prof. Marielia Assumption

INDEX

MongoDB Basics

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





```
Select Command Prompt - mongod
ver.
2018-07-13T18:21:33.958+0530 I CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specify
 which IP
2018-07-13T18:21:33.959+0530 I CONTROL  [initandlisten] **          addresses it should serve responses from, or with --
bind_ip_all to
2018-07-13T18:21:33.960+0530 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired,
 start the
2018-07-13T18:21:33.961+0530 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this warn
ing.
2018-07-13T18:21:33.961+0530 I CONTROL  [initandlisten]
2018-07-13T18:21:33.964+0530 I STORAGE  [initandlisten] createCollection: admin.system.version with provided UUID: fa2c3
6f0-6182-49db-b6a1-fba62c205e20
2018-07-13T18:21:34.125+0530 I COMMAND  [initandlisten] setting featureCompatibilityVersion to 4.0
2018-07-13T18:21:34.138+0530 I STORAGE  [initandlisten] createCollection: local.startup_log with generated UUID: d6a2ecb
3-2ac0-462f-b1d8-5f203a0ee547
2018-07-13T18:21:34.425+0530 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'C
:/data/db/diagnostic.data'
2018-07-13T18:21:34.428+0530 I STORAGE  [LogicalSessionCacheRefresh] createCollection: config.system.sessions with gener
ated UUID: 46713e3f-8119-4475-b531-2de7d2c6130e
2018-07-13T18:21:34.428+0530 I NETWORK  [initandlisten] waiting for connections on port 27017
2018-07-13T18:21:34.647+0530 I INDEX    [LogicalSessionCacheRefresh] build index on: config.system.sessions properties:
{ v: 2, key: { lastUse: 1 }, name: "lsidTTLIndex", ns: "config.system.sessions", expireAfterSeconds: 1800 }
2018-07-13T18:21:34.647+0530 I INDEX    [LogicalSessionCacheRefresh]     building index using bulk method; build may tem
porarily use up to 500 megabytes of RAM
2018-07-13T18:21:34.665+0530 I INDEX    [LogicalSessionCacheRefresh] build index done.  scanned 0 total records. 0 secs
2018-07-13T18:21:34.666+0530 I COMMAND  [LogicalSessionCacheRefresh] command config.$cmd command: createIndexes { create
Indexes: "system.sessions", indexes: [ { key: { lastUse: 1 }, name: "lsidTTLIndex", expireAfterSeconds: 1800 } ], $db: "
config" } numYields:0 reslen:114 locks:{ Global: { acquireCount: { r: 1, w: 1 } }, Database: { acquireCount: { W: 1 } },
 Collection: { acquireCount: { w: 1 } } } protocol:op_msg 237ms
```

Open Another New command prompt

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

C:\Users\MCC>mongo_
```

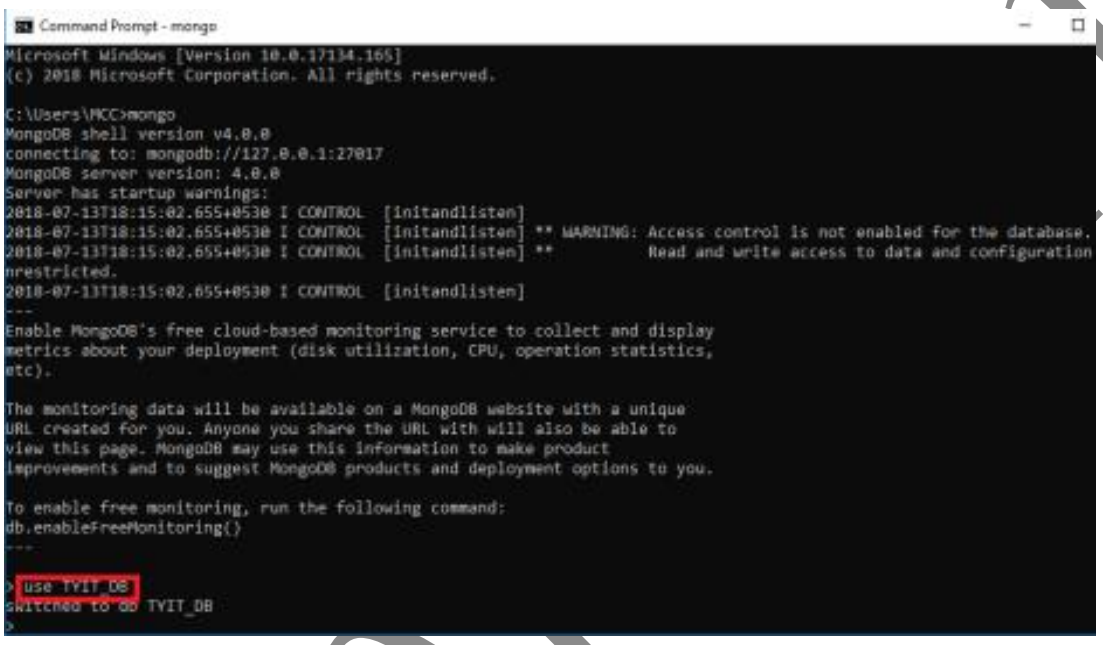
Activate Windows
Go to Settings to activate Windows.

Practical 01

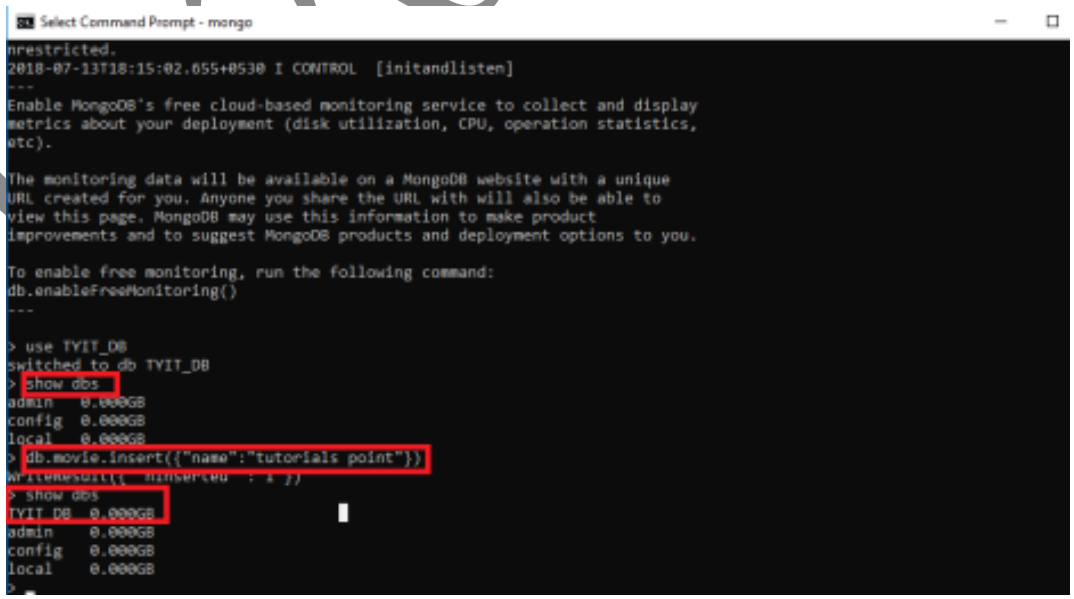
(a)Write a MongoDB command to create and drop database

(i)Creating database

```
use DATABASE_NAME
```



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



ii)DROP database

```
db.dropDatabase()
```

It drop the current database

```
Command Prompt - mongo
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

(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
```

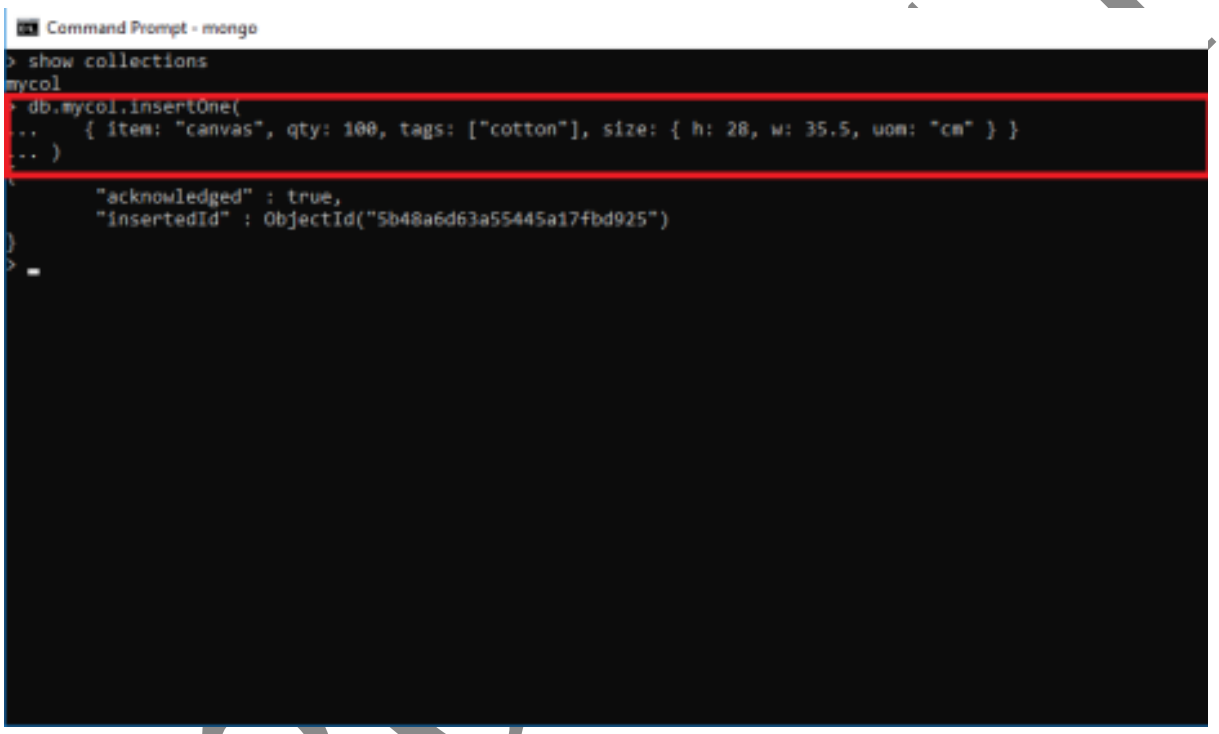
(ii)Inserting into Collections

Inserting into collection:-

db.mycol.insertOne(

{ item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom:

"cm" } })



(iii) Displaying the Collections

db.mycol.find().pretty()

```
Command Prompt - mongo
> 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

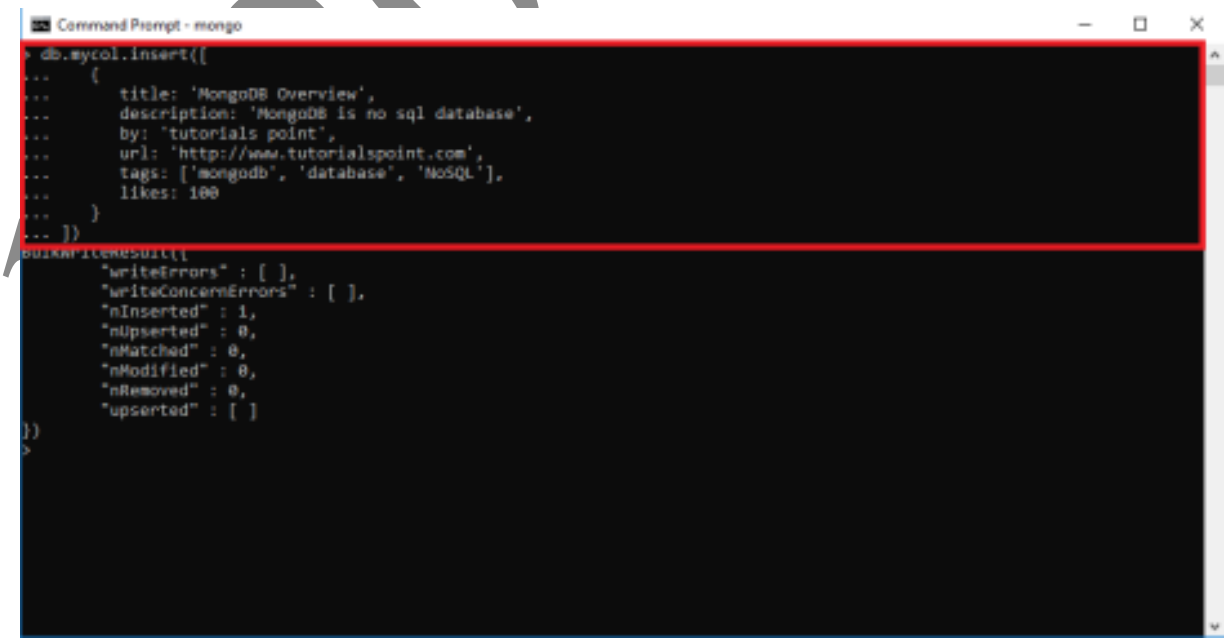
db.mycol.drop()

```
Command Prompt - mongo
> 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
 }
])
```



(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" : [ ]
})
>
```

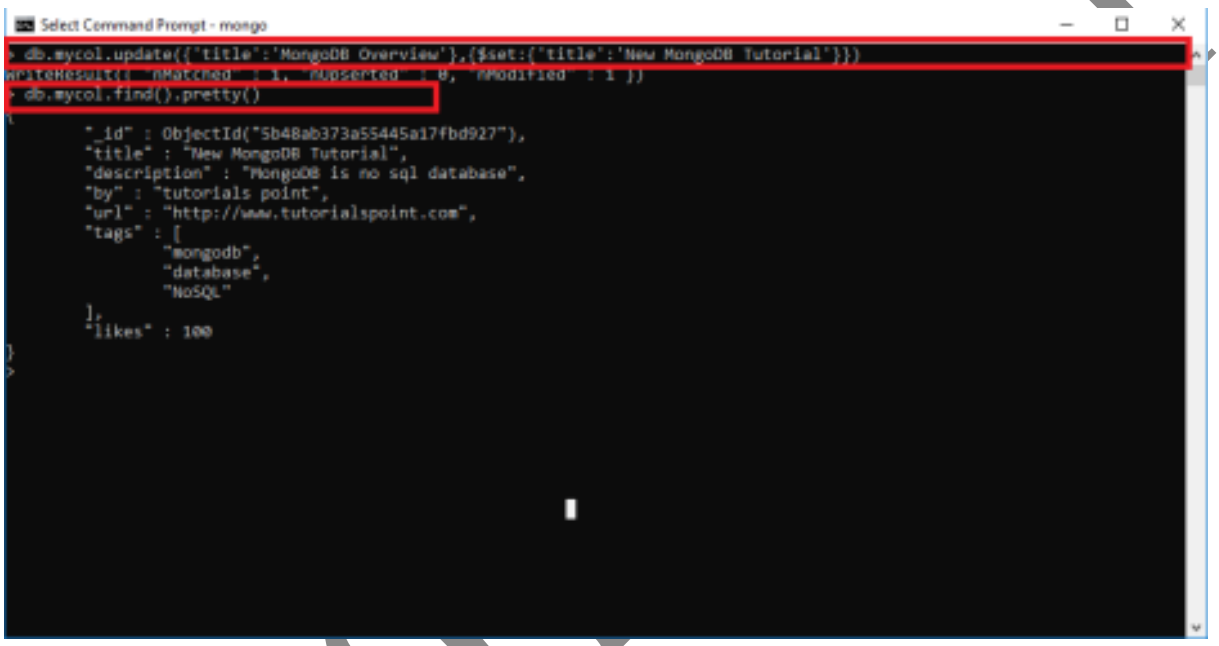
(iii) Updating Documents

To update the Documents

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

To display the Documents

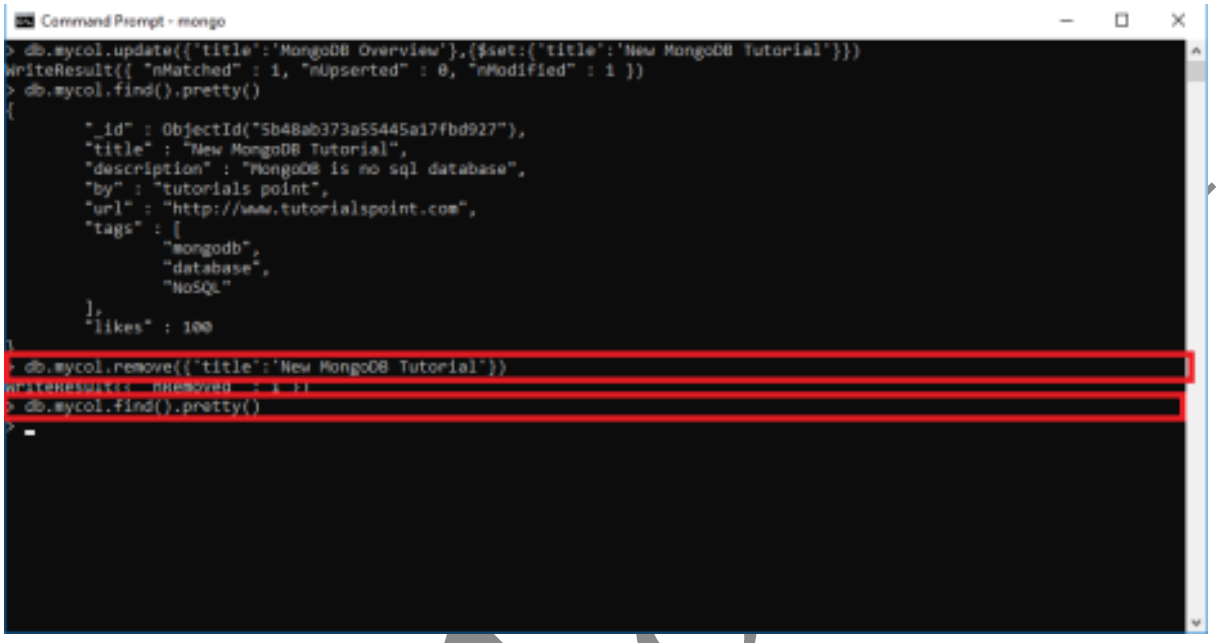
db.mycol.find().pretty()



(iv) Removing Documents

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

db.mycol.find().pretty()



Practical No. 02

1.Write a MongoDB query to display all the documents in the collection restaurants.

```
db.resaurants.find();
```

2.Write a MongoDB query to display the fields , restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

```
db.resaurants.find({},{“name”:1,”restaurant_id”:1,”borough”:1,”cuisine”:1});
```

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

```
db.resaurants.find({},”restaurant_id”:1,”name”:1,”borough”:1,”cuisine”:1,”_id”:0});
```

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

```
db.resaurants.find({},”restaurant_id”:1,”name”:1,”borough”:1,”address.zipcode”:1,”_id
”:0});
```

5.Write a MongoDB query to display all the restaurant which is in the borough Bronx

```
db.resaurants.find({“borough”:”Bronx”});
```

6.Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx

```
db.resaurants.find({“borough”:”Bronx”}).limit(5);
```

7.Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.

```
db.resaurants.find({“borough”:”Bronx”}).skip(5).limit(5);
```

8.Write a MongoDB query to find the restaurants who achieved a score more than 90

```
db.resaurants.find({grades:{$elemMatch:{“score”:{$gt:90}}}});
```

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

```
db.resaurants.find({“grades.score”:{$gt:80,$lt:100}});
```

10.Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168

```
db.resaurants.find({“address.coord” :{ $lt : -95.754168}});
```

11.Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168

```
db.resaurants.find({“cuisine”:{$ne:”American”},”grades.score”:{$gt:70},”address.coord”:{$lt: -65.754168}});
```

12.Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.Note : Do this query without using $and operator.

```
db.resaurants.find({“cuisine”:{$ne:”American”},”grades.score”:{$gt:70),”address.coord”:{$lt:-65.754168});
```

13.Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order

```
db.resaurants.find({“cuisine”:{$ne:”American”},”grades.grade”:”A”,”borough”:{$ne:””Brooklyn”}.sort({“cuisine”:1});
```

14Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

```
db.resaurants.find({“name”:/^will/},{“resaurants_id”:1.”name”:1.”borogh”1,”cuisine”1});
```

15.Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

```
db.resaurants.find({“name”:/res$/},{“resaurants_id”:1.”name”:1.”borogh”1,”cuisine”1});
```

16.Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name

```
db.resaurants.find({“name”:/Reg/},{“resaurants_id”:1.”name”:1.”borogh”1,”cuisine”1});
```

17.Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.

```
db.restaurants.find($and:[{“borough”:”Bronx”},{$or:[{“cuisine”:”American”},{“cuisine”:” Chinese”}]}]);
```

18.Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronx or Brooklyn

db.reaurants.find({“borough”:{$in:[”Staten Island”,”Queens”,”Bronx”,”Brooklyn”]}}, {“restaurant_id”:1,”name”:1,”borough”:1,”cuisine”:1});

19.Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Brooklyn.
db.reaurants.find({“borough”:{$nin:[”Staten Island”,”Queens”,”Bronx”,”Brooklyn”]}}, {“restaurant_id”:1,”name”:1,”borough”:1,”cuisine”:1});

20.Write a MongoDB queryto find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.

db.resaurants.find({grades.elementsmatch:{“score”;{$lt:10}}}{“restaurant_id”:1,”name”:1,”borough”1,”cuisine”:1});

21.Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'

db.resaurants.find({$or:[{“name”:/^wil/},{$and:[{“cuisine”:{$ne:”American”}}, {“cuisine”:{$ne:”Chinees”}}]}]},{“restaurant_id”:1,”name”:1,”borough”:1,”cuisine:1});

22.Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates

db.resaurants.find($and:[{“grades.date”: ISODate("2014-08-11 T00:00:00Z")},{“grades.grade”:”A”},{“grades.score”:11}]},{“restaurant_id”:1,”name”1,” grades”:1});

23.Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z"

db.resaurants.find($and:[{“grades.1.date”: ISODate("2014-08-11 T00:00:00Z")},{“grades.1.grade”:”A”},{“grades.1.score”:9}]},{“restaurant_id”:1,”name” 1,”grades”:1});

24Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52...

db.resaurants.find({“address.coord:1”:{$gt:42,$lt:52}},{“restaurant_id”:1,”name”:1,”address”:1,”coord:1”});

25.Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

db.restaurants.find().sort({“name”:1});

26.Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns

```
db.resaurants.find().sort({“name”:-1});
```

27.Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order

```
db.resaurants.find().sort({“cuisine”:1,”borough”:-1});
```

28.Write a MongoDB query to know whether all the addresses contains the street or not.

```
db.resaurants.find({“address.street”:{$exists:true}});
```

29.Write a MongoDB query which will select all documents in the restaurants collection where the coord field value isDouble

```
db.resaurants.find({“address.coord”:{$type:1}});
```

30Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

```
db.resaurants.find({“grades.score”{$mod:[7,0]}},{“restaurant_id”:1,”name”:1,”grades”:
1});
```

31.Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name

```
db.resaurants.find({name:{$regex:”mon”,$options:”i”}},{“name”:1,”borogh”:1,”address.
coord”:1,”cuisine”:1});
```

32.Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

```
db.resaurants.find({name:{$regex:/^Mad/i}},”}},{“name”:1,”borogh”:1,”address.coord”:
1,”cuisine”:1});
```

Practical 03:

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" : [ ]
})
>
Activate Windows
Go to Settings to activate Wi
```

(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 }
>
Activate Windows
Go to Settings to activate Wi
```

(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 }
>
Activate Windows
Go to Settings to activate Wi
```

(iii)Min

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

```
Command Prompt - mongo
...
...     {
...        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 }
>
Activate Windows
Go to Settings to activate Wi
```

(iv)MAX

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

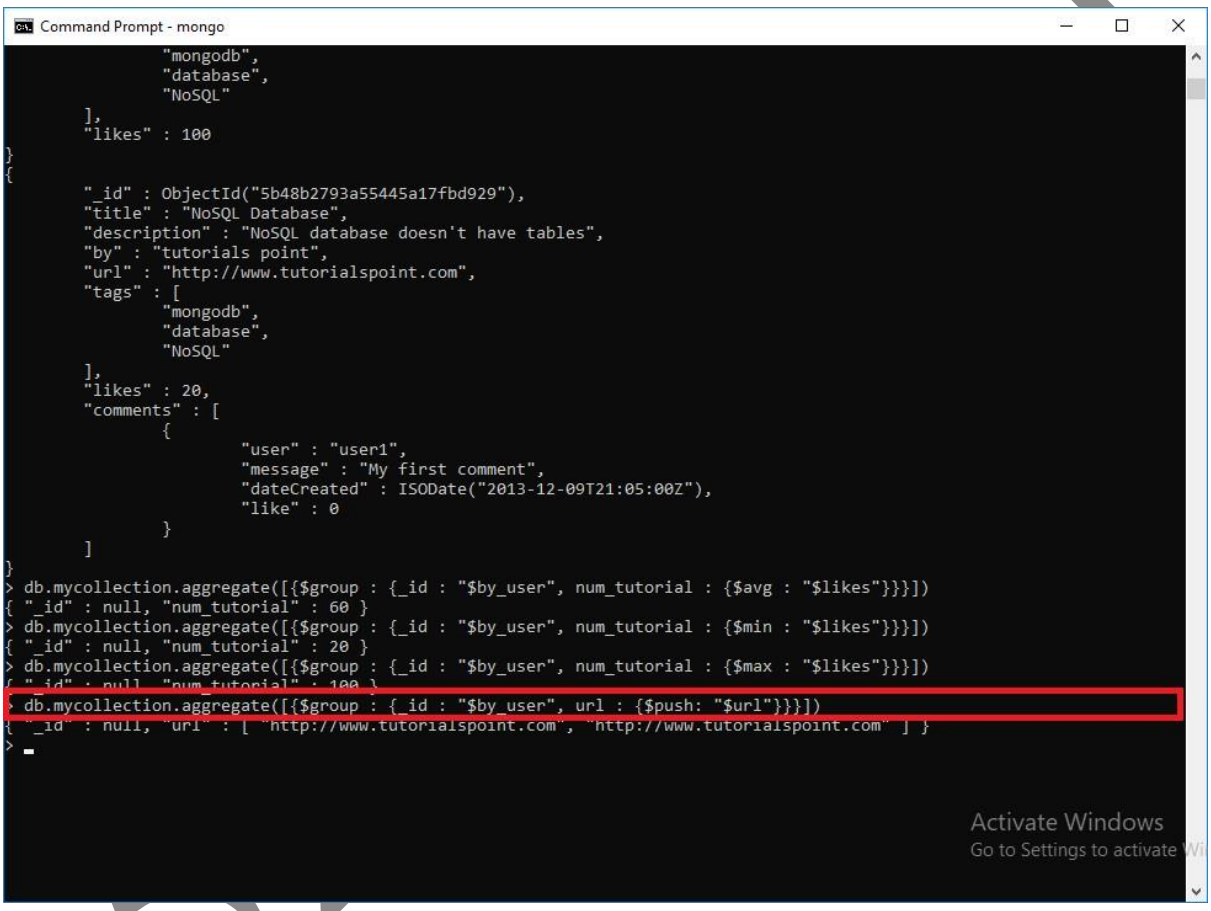
```
Command Prompt - mongo
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("5b48b2793a55445a17fbd929"),
        "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"}}}])



(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.)

```
Select 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" ] }
>
Activate Windows
Go to Settings to activate Wi
```

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"}}}])
{ "_id" : null, "first_url" : "http://www.tutorialspoint.com" }
>
Activate Windows
Go to Settings to activate Wi
```

(ii) Last

```
Command Prompt - mongo
{ "_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" }
>
Activate Windows
Go to Settings to activate Wi
```

Practical 04:

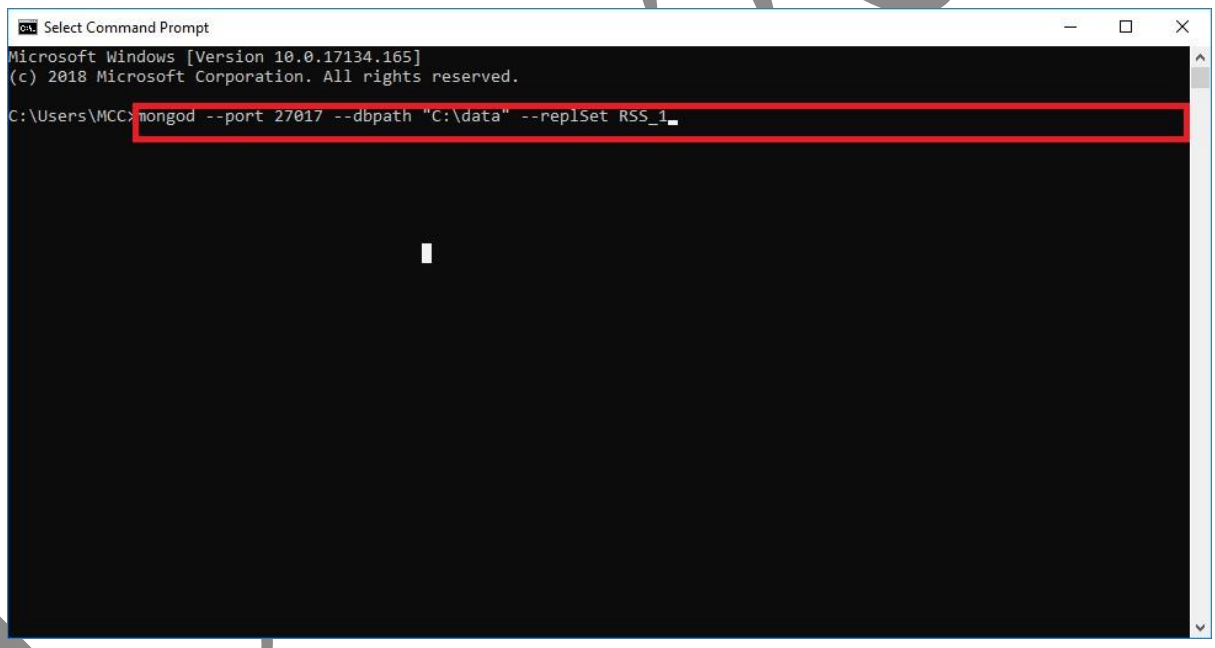
a)Write a MongoDB query to create a Replica of an existing database

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



Open new cmd and type mongo to connect this mongod instance

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

C:\Users\MCC>mongo
```

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

rs.initiate()

```
Command Prompt - mongo
---
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()**

Command Prompt - mongo

```
        "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)
                },
```

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

First start mongod on 1 cmd

```
Command Prompt - mongo
        "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)
                },
```

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 No.05

a) jQuery Basic,jQuery Events

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>
<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!

b) jQuery Selector,jQuery Hide and Show effects

- **jQuery Selector**

```
$("#test")
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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.

Click me

- **jQuery hide() and show() method**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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.

Hide Show

c) jQuery Fading effects ,jQuery Sliding effects

- **jQuery Fading effects**

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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.

Click to fade out boxes

- **jQuery Sliding effects**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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:

PRACTICAL NO.06

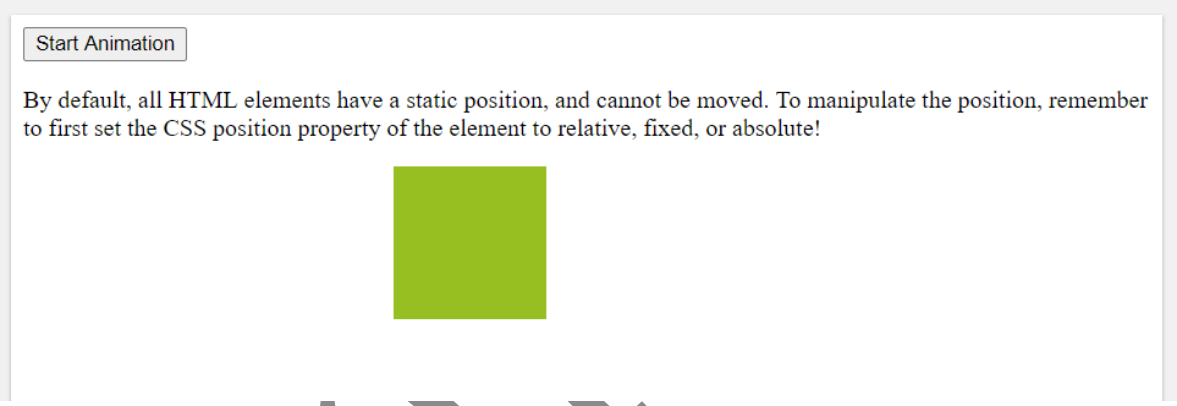
- **jQuery Advanced**

a) jQuery Animation effects,,jQuery Chaining

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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:



- **jQuery Chaining**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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!!

Click me

b) jQuery Callback,jQuery Get and Set Contents

- **jQuery Callback**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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:

Hide

This is a paragraph with little content.

- **jQuery Get**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

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

    alert("Value: " + $("#test").val());

  });

});

</script>

</head>

<body>
```

```
<p>Name: <input type="text" id="test" value="Mickey Mouse"></p>

<button>Show Value</button>

</body>
</html>
```

Output:

Name: Mickey Mous

Show Value

- **jQuery Set**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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>
```

Output:

This is a paragraph.

This is another paragraph.

Input field: Mickey Mous

Set Text Set HTML Set Value

c) jQuery Insert Content,jQuery Remove Elements

- **jQuery Insert**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script>

function appendText() {

  var txt1 = "<p>Text.</p>";        // Create text with HTML

  var txt2 = $("<p></p>").text("Text.");  // Create text with jQuery

  var txt3 = document.createElement("p");

  txt3.innerHTML = "Text.";         // Create text with DOM

  $("body").append(txt1, txt2, txt3);   // Append new elements

}

</script>

</head>

<body>


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

<button onclick="appendText()">Append text</button>


</body>
```

```
</html>
```

Output:

This is a paragraph.

Append text

Text.

Text.

Text.

- **jQuery Remove**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/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.

Remove div element

PRACTICAL NO.07

a) Creating JSON

```
<!DOCTYPE html>

<html>

<body>


<h2>Convert a JavaScript object into a JSON string, and send it to the
server.</h2>


<script>

var myObj = { name: "John", age: 31, city: "New York" };

var myJSON = JSON.stringify(myObj);

window.location = "demo_json.php?x=" + myJSON;

</script>


</body>

</html>
```

Output:

demo_json.php:

John from New York is 31

b) Parsing JSON

```
<!DOCTYPE html>
<html>
<body>

<h2>Create Object from JSON String</h2>

<p id="demo"></p>

<script>
var txt = '{"name":"John", "age":30, "city":"New York"}'
var obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>

</body>
</html>
```

Output:

Create Object from JSON String

John, 30

PRACTICAL No.08

Create a JSON file and import it to MongoDB

mongoimport --db warehouse --collection umongo --file <file_path>\warehouse_umongo_production_bkp_feb28.json --jsonArray

```
> show dbs
admin     0.000GB
config    0.000GB
local     0.000GB
warehouse 0.000GB
> use warehouse
switched to db warehouse
> show collections
umongo
> db.umongo.find()
{ "_id" : ObjectId("5a96e8791bfea118986a910c"), "name" : "Justin Heather", "loves" : [ "carrot", "grape" ], "weight" : "45 Kg", "gender" : "m", "votes
" : 43 }
{ "_id" : ObjectId("5a96e8791bfea118986a910f"), "name" : "April Sunners", "loves" : [ "strawberry", "lemon" ], "weight" : "73 Kg", "gender" : "f", "vo
tes" : 40 }
{ "_id" : ObjectId("5a96e8791bfea118986a910b"), "name" : "Harry Potter", "date_of_bith" : ISODate("1992-03-13T02:17:00Z"), "loves" : [ "carrot", "papa
ya" ], "weight" : "60 Kg", "gender" : "m", "votes" : 63 }
{ "_id" : ObjectId("5a96e8791bfea118986a910e"), "name" : "Charlotte Neil", "loves" : [ "apple" ], "weight" : "57 Kg", "gender" : "f", "votes" : 99 }
{ "_id" : ObjectId("5a96e8791bfea118986a910d"), "name" : "Daniel Atlas", "date_of_bith" : ISODate("1985-07-03T20:31:00Z"), "loves" : [ "apple", "carro
t" ], "weight" : "57 Kg", "gender" : "m", "votes" : 99 }
```