Experiment No.5

Title: NOSQL and MongoDB

Batch: A2 **Roll No.:** 1914072 **Experiment No.:**5

Aim: To implement a NOSQL database using MongoDB and execute queries.

Resources needed: MongoDB, Windows

Theory

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++.

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field

Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)

Sample Document

Following example shows the document structure of a blog site, which is simply a comma separated key value pair.

```
{
  _id: ObjectId(7df78ad8902c)
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'MongoDb info',
  url: 'http://www.MongoDb.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100,
  comments: [
    {
      user:'user1',
      message: 'My first comment',
      dateCreated: new Date(2011,1,20,2,15),
      like: 0
    },
    {
      user:'user2',
      message: 'My second comments',
      dateCreated: new Date(2011,1,25,7,45),
      like: 5
    }
  ]
}
```

_id is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide _id while inserting the document. If you don't provide then MongoDB provides a unique id for every document. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

Data in MongoDB has a flexible schema.documents in the same collection. They do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Some considerations while designing Schema in MongoDB

- Design your schema according to user requirements.
- Combine objects into one document if you will use them together. Otherwise separate them (but make sure there should not be need of joins).
- Duplicate the data (but limited) because disk space is cheap as compare to compute time.

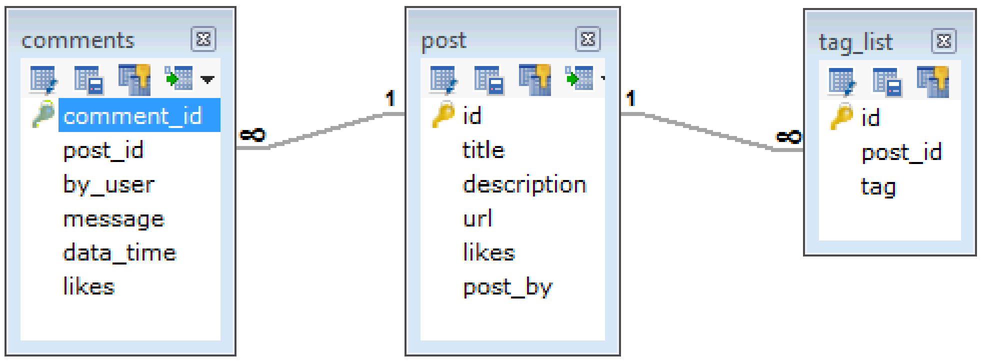
- Do joins while write, not on read.
- Optimize your schema for most frequent use cases.
- Do complex aggregation in the schema.

Example

Suppose a client needs a database design for his blog/website and see the differences between RDBMS and MongoDB schema design. Website has the following requirements.

- Every post has the unique title, description and url.
- Every post can have one or more tags.
- Every post has the name of its publisher and total number of likes.
- Every post has comments given by users along with their name, message, data-time and likes.
- On each post, there can be zero or more comments.

In RDBMS schema, design for above requirements will have minimum three tables.



While in MongoDB schema, design will have one collection post and the following structure –

```
{
  _id: POST_ID
  title: TITLE_OF_POST,
  description: POST_DESCRIPTION,
  by: POST_BY,
  url: URL_OF_POST,
  tags: [TAG1, TAG2, TAG3],
  likes: TOTAL_LIKES,
  comments: [
    {
      user:'COMMENT_BY',
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    },
    {
      user:'COMMENT_BY',
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    }
```

```
   ]
}
```

Why Use MongoDB?

- **Document Oriented Storage** − Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability
- Auto-sharding
- Rich queries
- Fast in-place updates
- Professional support by MongoDB

Where to Use MongoDB?

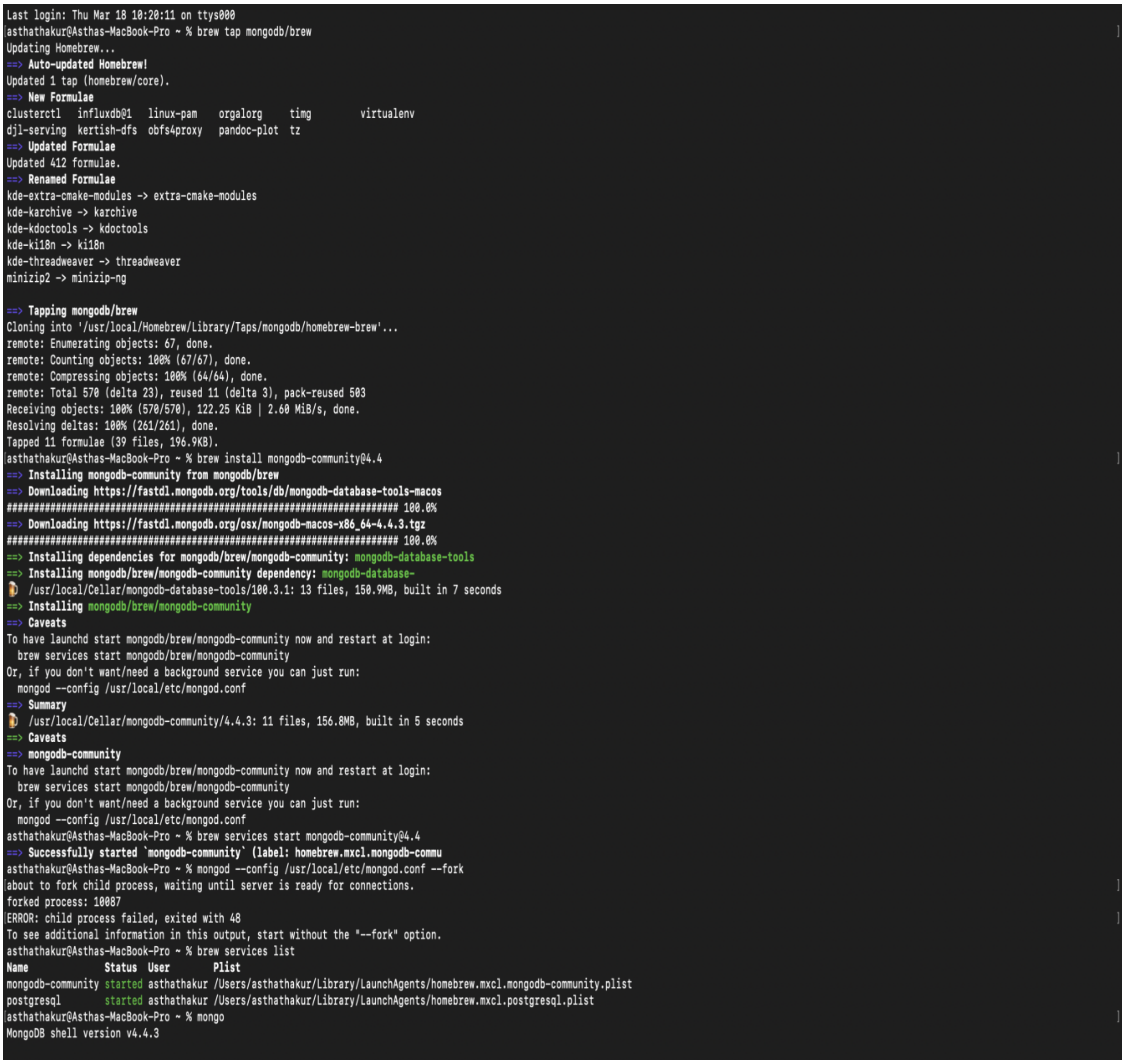
- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

Procedure:

1. Installation of mongodb on windows
2. Create nosql database using mongodb
3. Insert data into collection
4. Update, Retrieve and delete entries from collection
5. Explore any 5 methods(not mentioned in the document)and use it on your database.

Results: (Program printout with output)

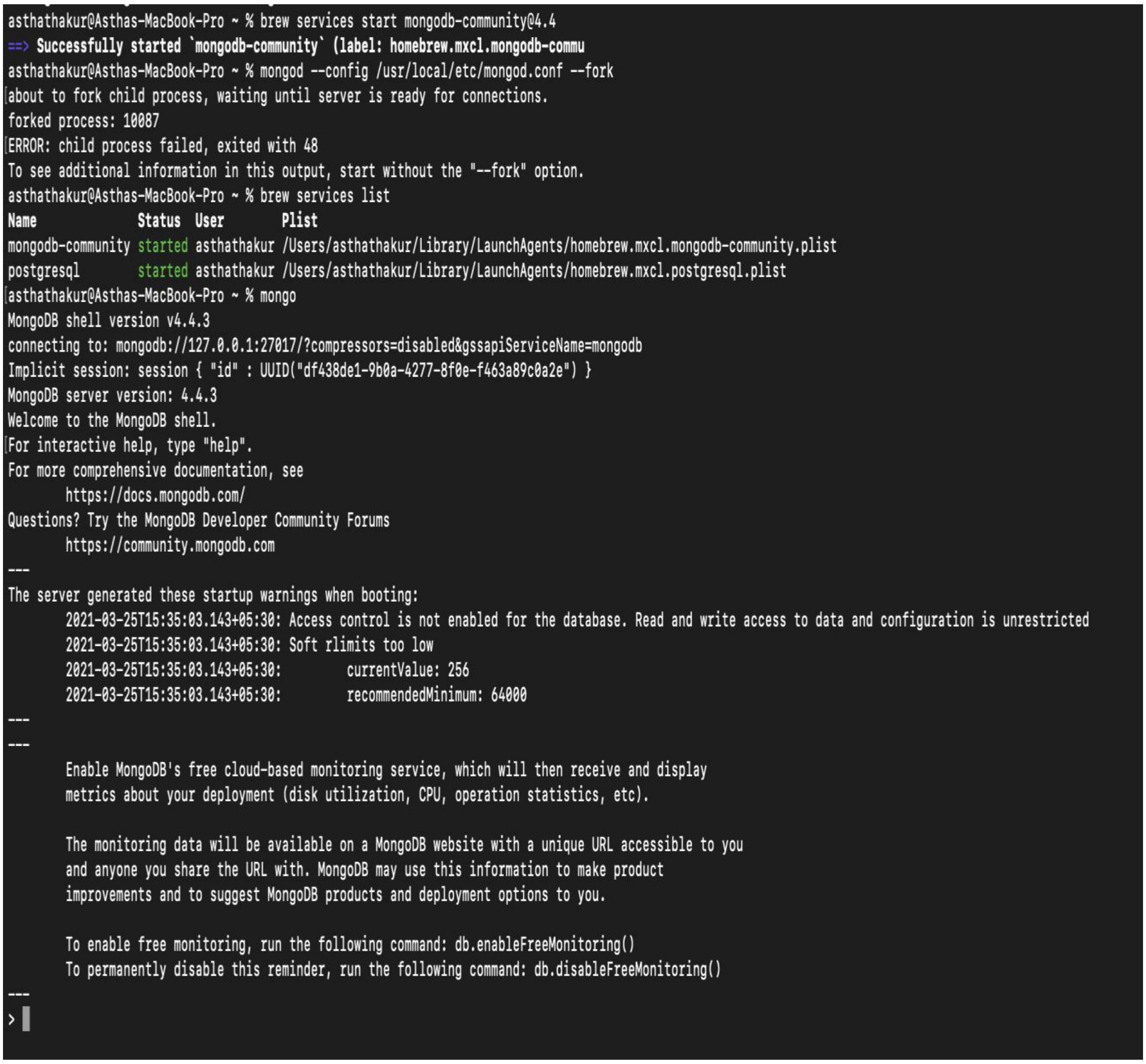
1. Downloading mongodb from brew



```
Last login: Thu Mar 18 10:20:11 on ttys000
asthathakur@Asthas-MacBook-Pro ~ % brew tap mongodb/brew
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 1 tap (homebrew/core).
==> New Formulae
clusterctl   influxdb@1   linux-pam    orgalorg    timg        virtualenv
djl-serving  kertish-dfs  obfs4proxy   pandoc-plot  tz
==> Updated Formulae
Updated 412 formulae.
==> Renamed Formulae
kde-extra-cmake-modules -> extra-cmake-modules
kde-karchive -> karchive
kde-kdoctools -> kdoctools
kde-ki18n -> ki18n
kde-threadweaver -> threadweaver
minizip2 -> minizip-ng

==> Tapping mongodb/brew
Cloning into '/usr/local/Homebrew/Library/Taps/mongodb/homebrew-brew'...
remote: Enumerating objects: 67, done.
remote: Counting objects: 100% (67/67), done.
remote: Compressing objects: 100% (64/64), done.
remote: Total 570 (delta 23), reused 11 (delta 3), pack-reused 503
Receiving objects: 100% (570/570), 122.25 KiB | 2.60 MiB/s, done.
Resolving deltas: 100% (261/261), done.
Tapped 11 formulae (39 files, 196.9KB).
asthathakur@Asthas-MacBook-Pro ~ % brew install mongodb-community@4.4
==> Installing mongodb-community from mongodb/brew
==> Downloading https://fastdl.mongodb.org/tools/db/mongodb-database-tools-macos
######################################################################## 100.0%
==> Downloading https://fastdl.mongodb.org/osx/mongodb-macos-x86_64-4.4.3.tgz
######################################################################## 100.0%
==> Installing dependencies for mongodb/brew/mongodb-community: mongodb-database-tools
==> Installing mongodb/brew/mongodb-community dependency: mongodb-database-
🍺  /usr/local/Cellar/mongodb-database-tools/100.3.1: 13 files, 150.9MB, built in 7 seconds
==> Installing mongodb/brew/mongodb-community
==> Caveats
To have launchd start mongodb/brew/mongodb-community now and restart at login:
  brew services start mongodb/brew/mongodb-community
Or, if you don't want/need a background service you can just run:
  mongod --config /usr/local/etc/mongod.conf
==> Summary
🍺  /usr/local/Cellar/mongodb-community/4.4.3: 11 files, 156.8MB, built in 5 seconds
==> Caveats
==> mongodb-community
To have launchd start mongodb/brew/mongodb-community now and restart at login:
  brew services start mongodb/brew/mongodb-community
Or, if you don't want/need a background service you can just run:
  mongod --config /usr/local/etc/mongod.conf
asthathakur@Asthas-MacBook-Pro ~ % brew services start mongodb-community@4.4
==> Successfully started `mongodb-community` (label: homebrew.mxcl.mongodb-commu
asthathakur@Asthas-MacBook-Pro ~ % mongod --config /usr/local/etc/mongod.conf --fork
about to fork child process, waiting until server is ready for connections.
forked process: 10087
ERROR: child process failed, exited with 48
To see additional information in this output, start without the "--fork" option.
asthathakur@Asthas-MacBook-Pro ~ % brew services list
Name              Status  User        Plist
mongodb-community started asthathakur /Users/asthathakur/Library/LaunchAgents/homebrew.mxcl.mongodb-community.plist
postgresql        started asthathakur /Users/asthathakur/Library/LaunchAgents/homebrew.mxcl.postgresql.plist
asthathakur@Asthas-MacBook-Pro ~ % mongo
MongoDB shell version v4.4.3
```

2. Starting the mongo server

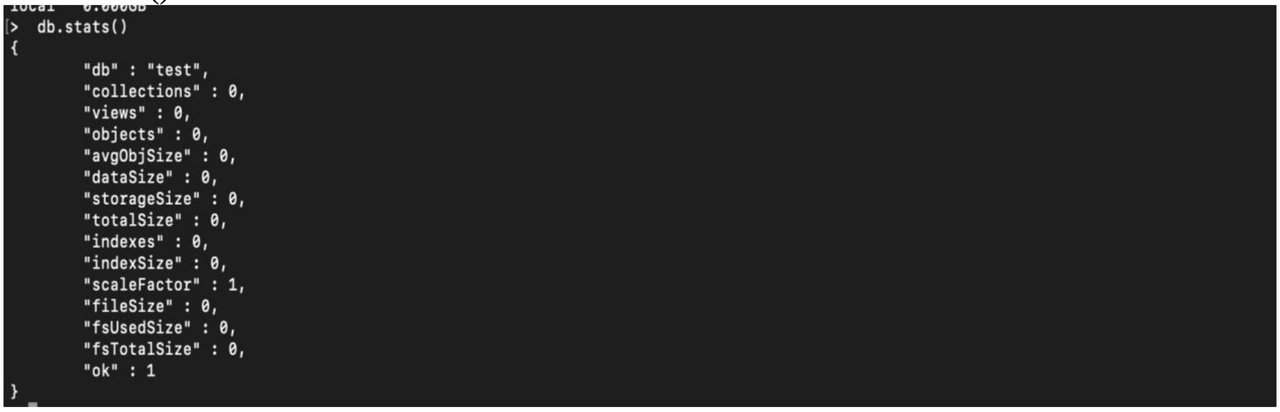


```
asthathakur@Asthas-MacBook-Pro ~ % brew services start mongodb-community@4.4
==> Successfully started `mongodb-community` (label: homebrew.mxcl.mongodb-commu
asthathakur@Asthas-MacBook-Pro ~ % mongod --config /usr/local/etc/mongod.conf --fork
about to fork child process, waiting until server is ready for connections.
forked process: 10087
ERROR: child process failed, exited with 48
To see additional information in this output, start without the "--fork" option.
asthathakur@Asthas-MacBook-Pro ~ % brew services list
Name              Status  User        Plist
mongodb-community started asthathakur /Users/asthathakur/Library/LaunchAgents/homebrew.mxcl.mongodb-community.plist
postgresql        started asthathakur /Users/asthathakur/Library/LaunchAgents/homebrew.mxcl.postgresql.plist
asthathakur@Asthas-MacBook-Pro ~ % mongo
MongoDB shell version v4.4.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("df438de1-9b0a-4277-8f0e-f463a89c0a2e") }
MongoDB server version: 4.4.3
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2021-03-25T15:35:03.143+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
        2021-03-25T15:35:03.143+05:30: Soft rlimits too low
        2021-03-25T15:35:03.143+05:30:         currentValue: 256
        2021-03-25T15:35:03.143+05:30:         recommendedMinimum: 64000
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive 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 accessible to you
        and anyone you share the URL with. 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()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

3. **Stats**
   db.stats()



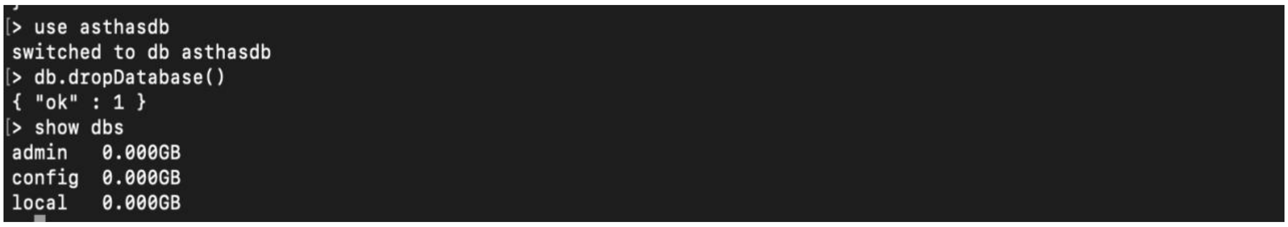
```
>  db.stats()
{
        "db" : "test",
        "collections" : 0,
        "views" : 0,
        "objects" : 0,
        "avgObjSize" : 0,
        "dataSize" : 0,
        "storageSize" : 0,
        "totalSize" : 0,
        "indexes" : 0,
        "indexSize" : 0,
        "scaleFactor" : 1,
        "fileSize" : 0,
        "fsUsedSize" : 0,
        "fsTotalSize" : 0,
        "ok" : 1
}
```

4. **Show and add to database list and drop database**

use asthasdb
switched to db asthasdb

db.dropDatabase()

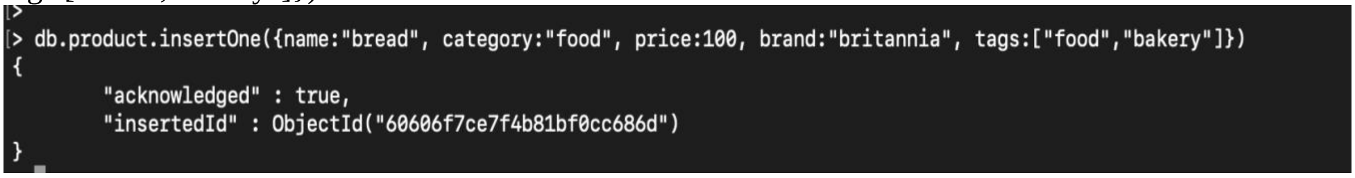
show dbs



```
> use asthasdb
switched to db asthasdb
> db.dropDatabase()
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

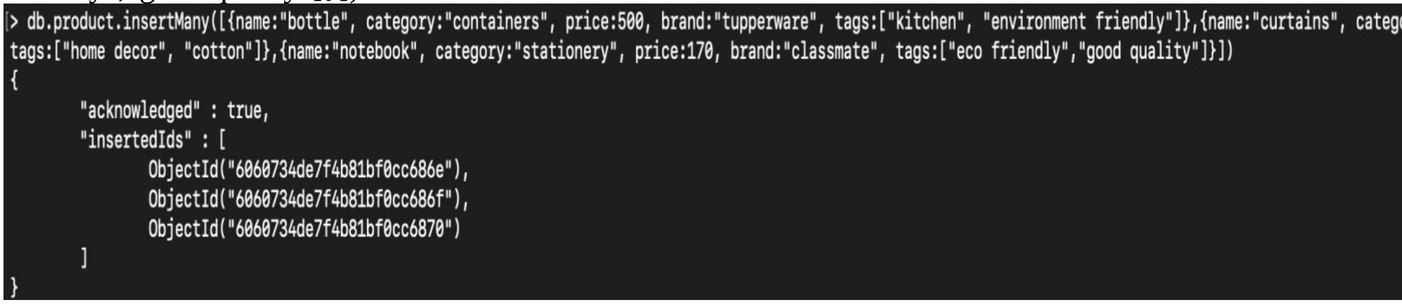
5. Insert

db.product.insertOne({name:"bread", category:"food", price:100, brand:"britannia", tags:["food","bakery"]})



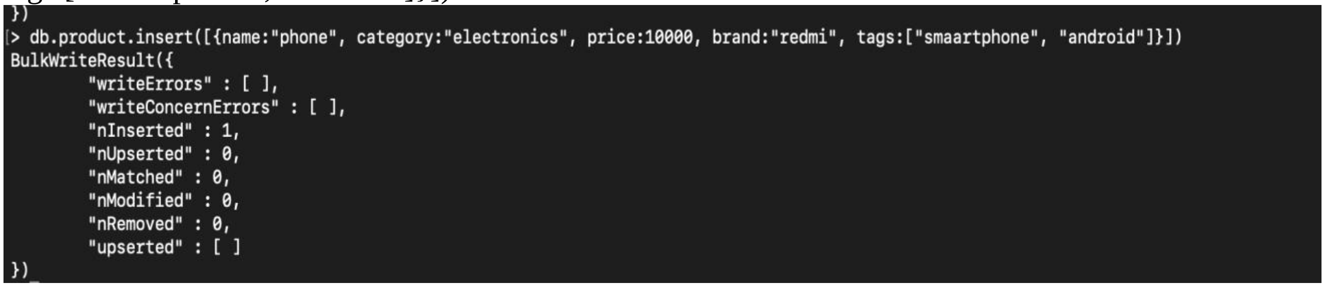
```
>
> db.product.insertOne({name:"bread", category:"food", price:100, brand:"britannia", tags:["food","bakery"]})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("60606f7ce7f4b81bf0cc686d")
}
```

db.product.insertMany([{name:"bottle", category:"containers", price:500, brand:"tupperware", tags:["kitchen", "environment friendly"]},{name:"curtains", category:"home decor", price:1000, brand:"ikea", tags:["home decor", "cotton"]},{name:"notebook", category:"stationery", price:170, brand:"classmate", tags:["eco friendly","good quality"]}])



```
> db.product.insertMany([{name:"bottle", category:"containers", price:500, brand:"tupperware", tags:["kitchen", "environment friendly"]},{name:"curtains", categ
tags:["home decor", "cotton"]},{name:"notebook", category:"stationery", price:170, brand:"classmate", tags:["eco friendly","good quality"]}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6060734de7f4b81bf0cc686e"),
                ObjectId("6060734de7f4b81bf0cc686f"),
                ObjectId("6060734de7f4b81bf0cc6870")
        ]
}
```

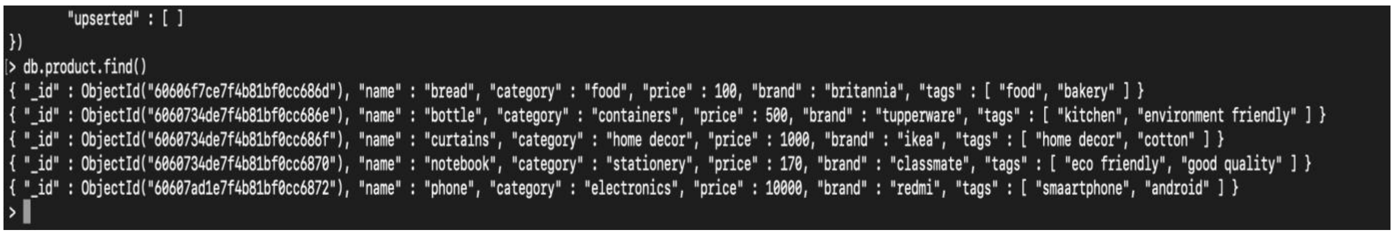
db.product.insert([{name:"phone", category:"electronics", price:10000, brand:"redmi", tags:["smaartphone", "android"]}])



```
})
> db.product.insert([{name:"phone", category:"electronics", price:10000, brand:"redmi", tags:["smaartphone", "android"]}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 1,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

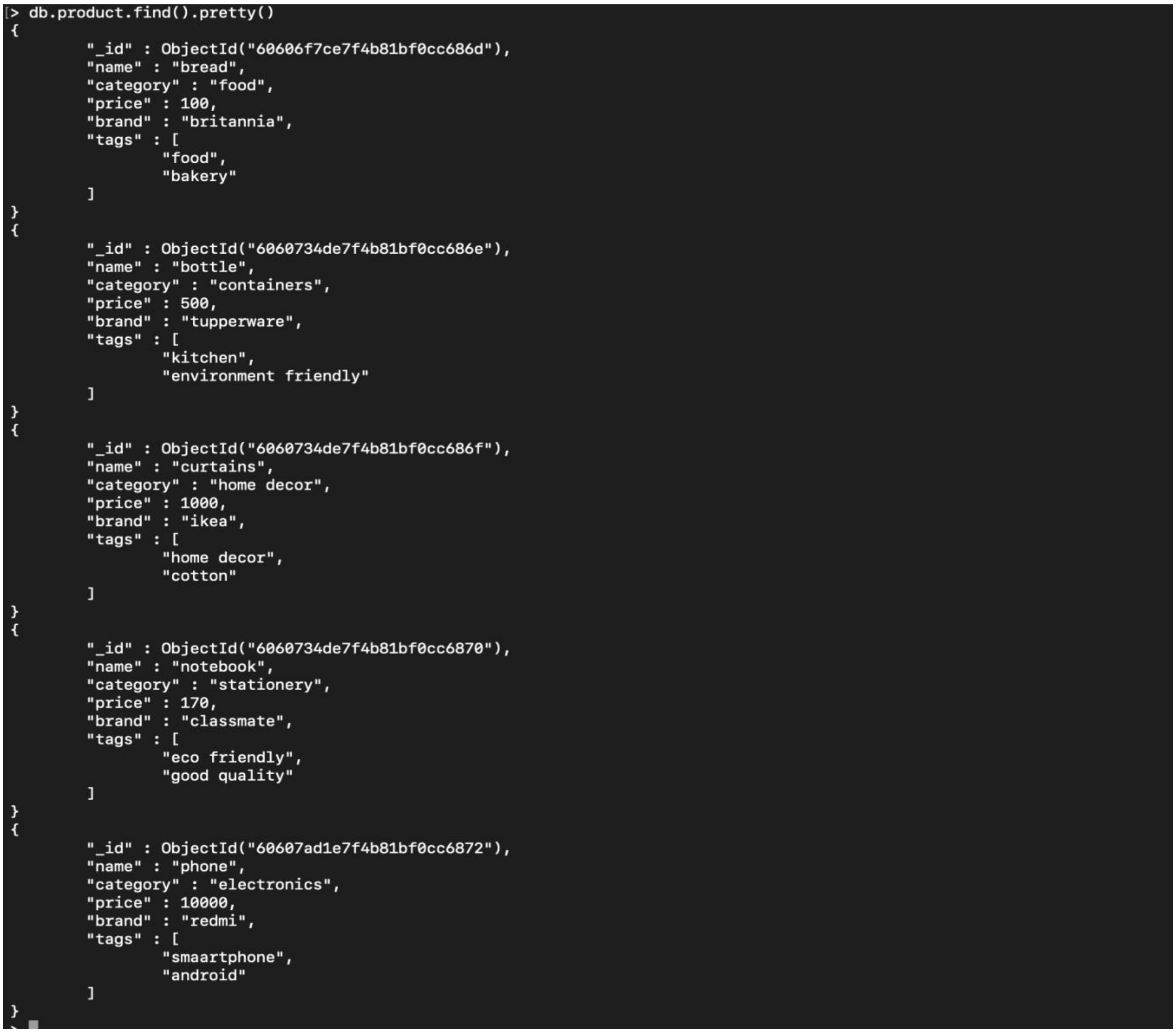
6. Find

db.product.find()



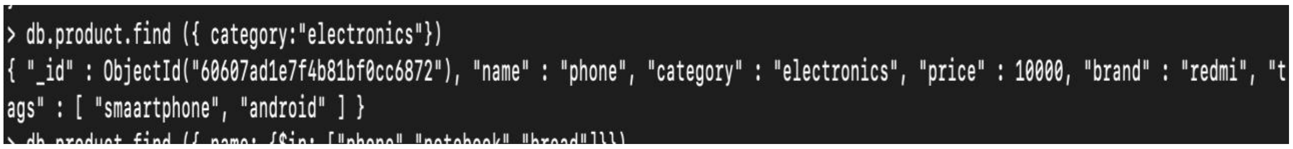
```
        "upserted" : [ ]
})
> db.product.find()
{ "_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name" : "bread", "category" : "food", "price" : 100, "brand" : "britannia", "tags" : [ "food", "bakery" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686e"), "name" : "bottle", "category" : "containers", "price" : 500, "brand" : "tupperware", "tags" : [ "kitchen", "environment friendly" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686f"), "name" : "curtains", "category" : "home decor", "price" : 1000, "brand" : "ikea", "tags" : [ "home decor", "cotton" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc6870"), "name" : "notebook", "category" : "stationery", "price" : 170, "brand" : "classmate", "tags" : [ "eco friendly", "good quality" ] }
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10000, "brand" : "redmi", "tags" : [ "smaartphone", "android" ] }
>
```

To display it in formatted way use pretty
db.product.find().pretty()



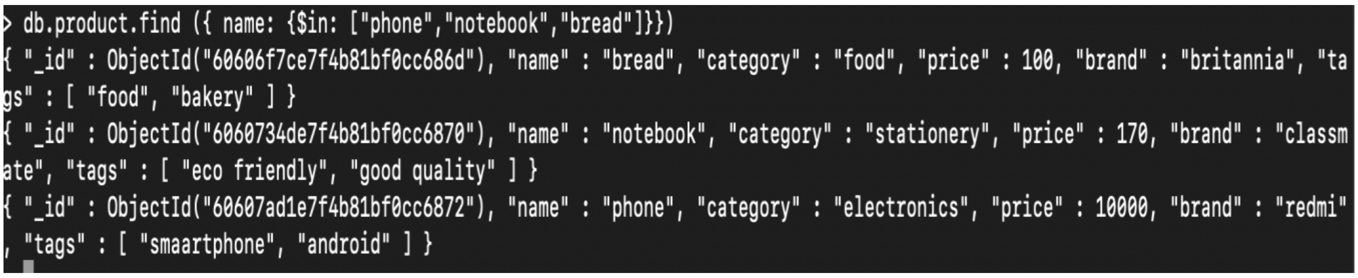
```
> db.product.find().pretty()
{
        "_id" : ObjectId("60606f7ce7f4b81bf0cc686d"),
        "name" : "bread",
        "category" : "food",
        "price" : 100,
        "brand" : "britannia",
        "tags" : [
                "food",
                "bakery"
        ]
}
{
        "_id" : ObjectId("6060734de7f4b81bf0cc686e"),
        "name" : "bottle",
        "category" : "containers",
        "price" : 500,
        "brand" : "tupperware",
        "tags" : [
                "kitchen",
                "environment friendly"
        ]
}
{
        "_id" : ObjectId("6060734de7f4b81bf0cc686f"),
        "name" : "curtains",
        "category" : "home decor",
        "price" : 1000,
        "brand" : "ikea",
        "tags" : [
                "home decor",
                "cotton"
        ]
}
{
        "_id" : ObjectId("6060734de7f4b81bf0cc6870"),
        "name" : "notebook",
        "category" : "stationery",
        "price" : 170,
        "brand" : "classmate",
        "tags" : [
                "eco friendly",
                "good quality"
        ]
}
{
        "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"),
        "name" : "phone",
        "category" : "electronics",
        "price" : 10000,
        "brand" : "redmi",
        "tags" : [
                "smaartphone",
                "android"
        ]
}
>
```

db.product.find ({ category:"electronics"})



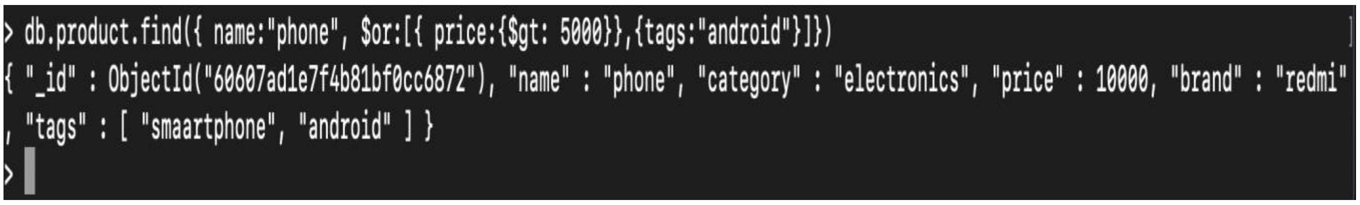
```
> db.product.find ({ category:"electronics"})
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10000, "brand" : "redmi", "t
ags" : [ "smaartphone", "android" ] }
> db.product.find ({ name: {$in: ["phone","notebook","bread"]}})
```

db.product.find ({ name: {$in: ["phone","notebook","bread"]}})



```
> db.product.find ({ name: {$in: ["phone","notebook","bread"]}})
{ "_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name" : "bread", "category" : "food", "price" : 100, "brand" : "britannia", "ta
gs" : [ "food", "bakery" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc6870"), "name" : "notebook", "category" : "stationery", "price" : 170, "brand" : "classm
ate", "tags" : [ "eco friendly", "good quality" ] }
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10000, "brand" : "redmi"
, "tags" : [ "smaartphone", "android" ] }
```

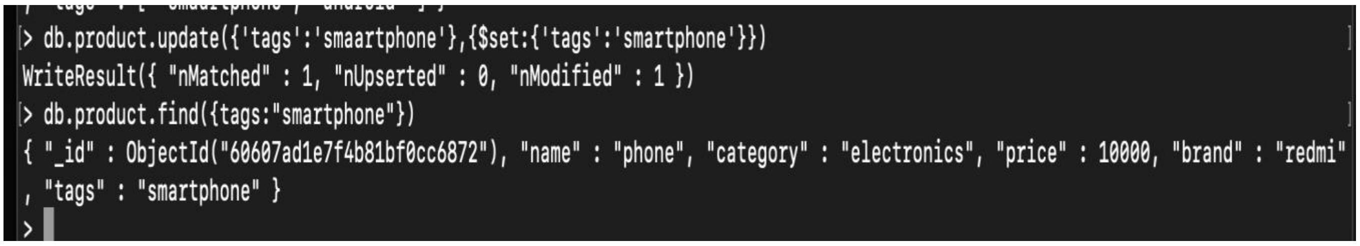
db.product.find({ name:"phone", $or:[{ price:{$gt: 5000}},{tags:"android"}]})



```
> db.product.find({ name:"phone", $or:[{ price:{$gt: 5000}},{tags:"android"}]})
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10000, "brand" : "redmi"
, "tags" : [ "smaartphone", "android" ] }
>
```

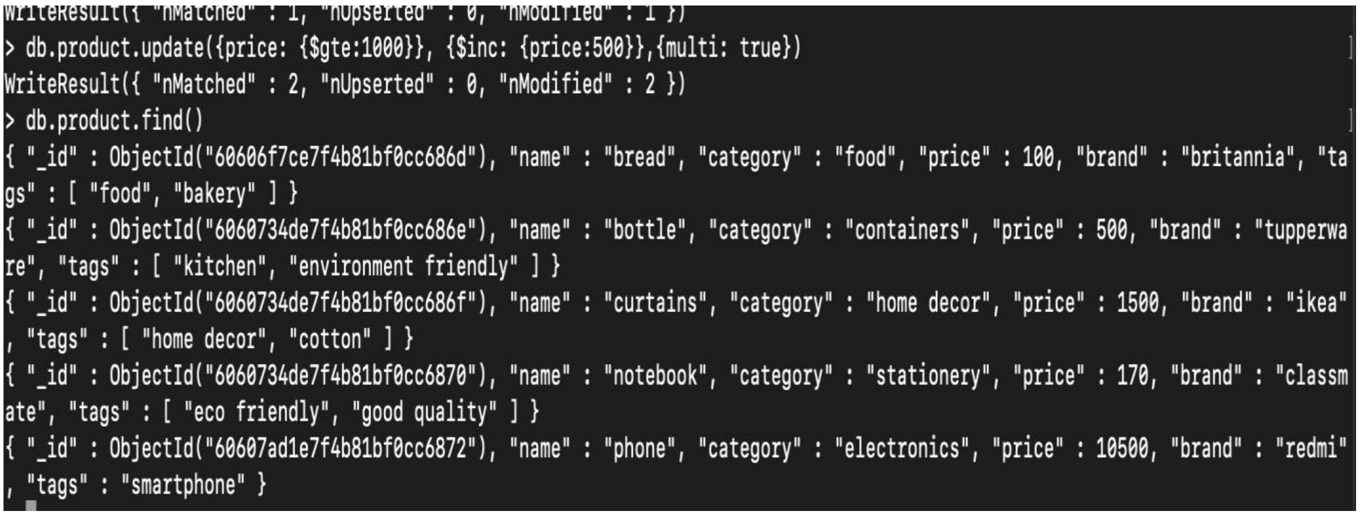
7. Update

db.product.update({'tags':'smaartphone'},{$set:{'tags':'smartphone'}})



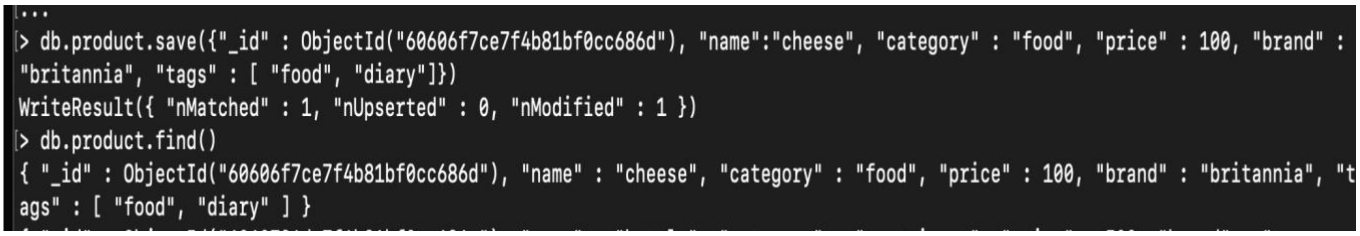
```
, "tags" : [ "smaartphone", "android" ] }
> db.product.update({'tags':'smaartphone'},{$set:{'tags':'smartphone'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.product.find({tags:"smartphone"})
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10000, "brand" : "redmi"
, "tags" : "smartphone" }
>
```

db.product.update({price: {$gte:1000}}, {$inc: {price:500}},{multi: true})



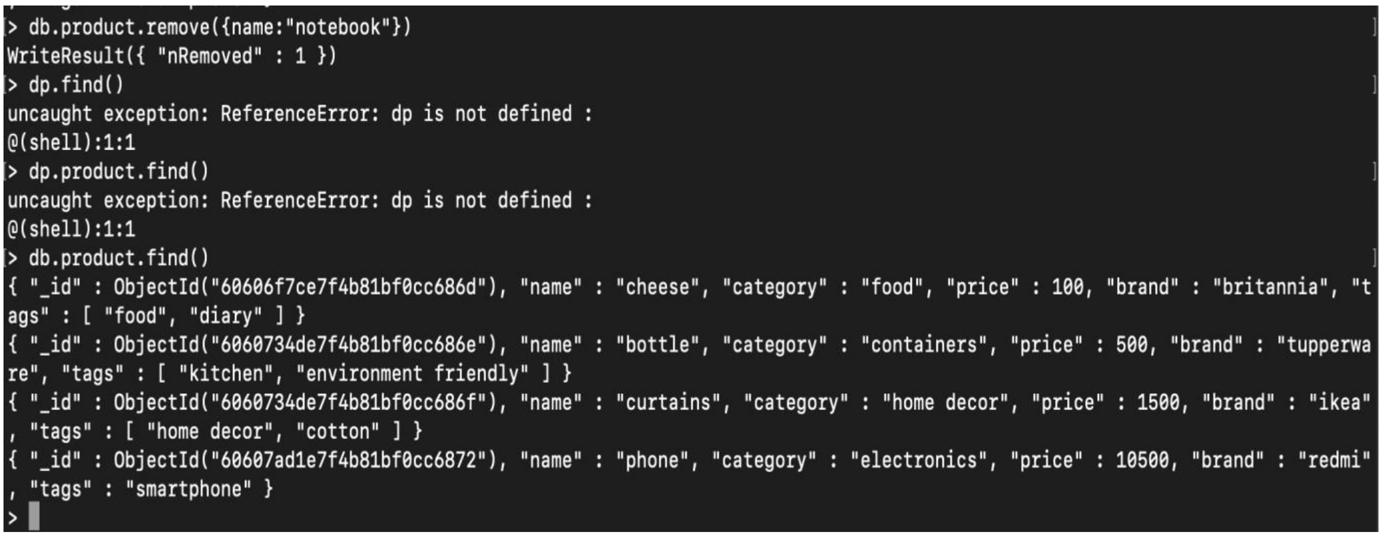
```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.product.update({price: {$gte:1000}}, {$inc: {price:500}},{multi: true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
> db.product.find()
{ "_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name" : "bread", "category" : "food", "price" : 100, "brand" : "britannia", "ta
gs" : [ "food", "bakery" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686e"), "name" : "bottle", "category" : "containers", "price" : 500, "brand" : "tupperwa
re", "tags" : [ "kitchen", "environment friendly" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686f"), "name" : "curtains", "category" : "home decor", "price" : 1500, "brand" : "ikea"
, "tags" : [ "home decor", "cotton" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc6870"), "name" : "notebook", "category" : "stationery", "price" : 170, "brand" : "classm
ate", "tags" : [ "eco friendly", "good quality" ] }
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10500, "brand" : "redmi"
, "tags" : "smartphone" }
```

8. Save
db.product.save({"_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name":"cheese", "category" : "food", "price" : 100, "brand" : "britannia", "tags" : ["food", "diary"]})



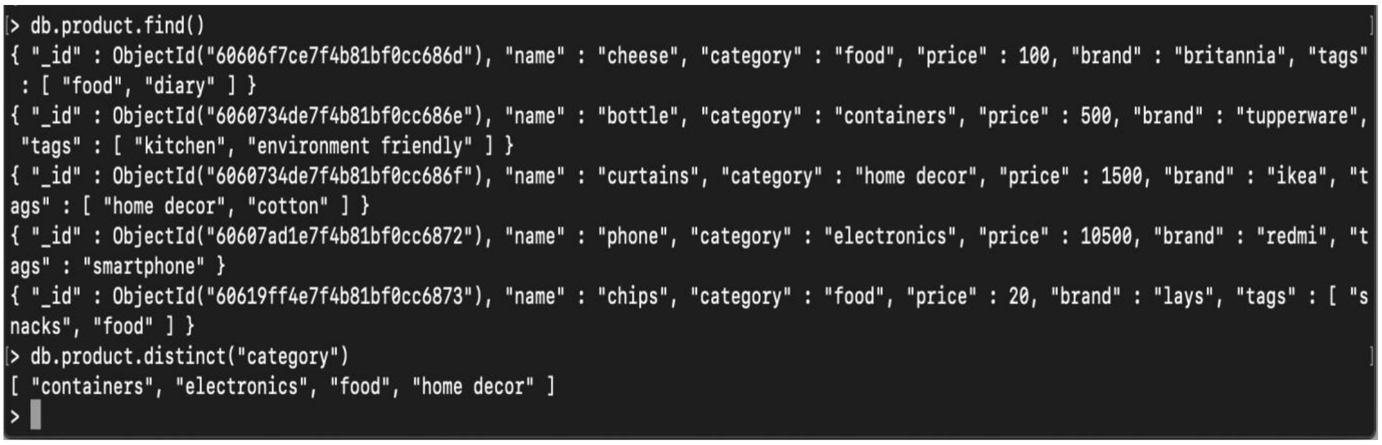
```
...
> db.product.save({"_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name":"cheese", "category" : "food", "price" : 100, "brand" :
"britannia", "tags" : [ "food", "diary"]})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.product.find()
{ "_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name" : "cheese", "category" : "food", "price" : 100, "brand" : "britannia", "t
ags" : [ "food", "diary" ] }
```

9. Remove
db.product.remove({name:"notebook"})



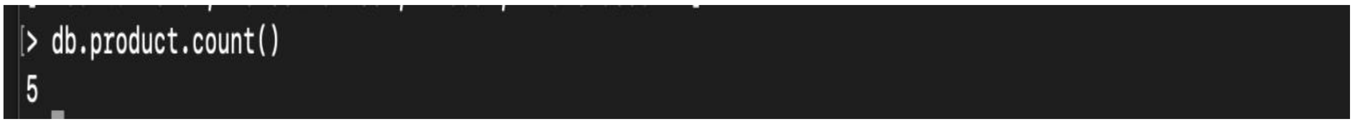
```
> db.product.remove({name:"notebook"})
WriteResult({ "nRemoved" : 1 })
> dp.find()
uncaught exception: ReferenceError: dp is not defined :
@(shell):1:1
> dp.product.find()
uncaught exception: ReferenceError: dp is not defined :
@(shell):1:1
> db.product.find()
{ "_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name" : "cheese", "category" : "food", "price" : 100, "brand" : "britannia", "t
ags" : [ "food", "diary" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686e"), "name" : "bottle", "category" : "containers", "price" : 500, "brand" : "tupperwa
re", "tags" : [ "kitchen", "environment friendly" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686f"), "name" : "curtains", "category" : "home decor", "price" : 1500, "brand" : "ikea"
, "tags" : [ "home decor", "cotton" ] }
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10500, "brand" : "redmi"
, "tags" : "smartphone" }
>
```

Additional commands
10. Distinct
db.product.distinct("category")



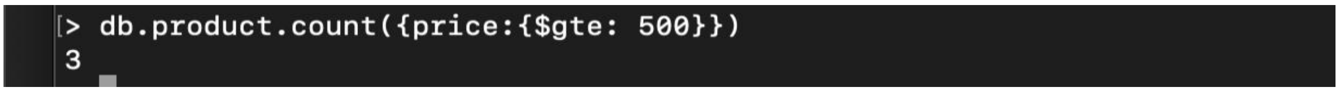
```
> db.product.find()
{ "_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name" : "cheese", "category" : "food", "price" : 100, "brand" : "britannia", "tags"
 : [ "food", "diary" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686e"), "name" : "bottle", "category" : "containers", "price" : 500, "brand" : "tupperware",
 "tags" : [ "kitchen", "environment friendly" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686f"), "name" : "curtains", "category" : "home decor", "price" : 1500, "brand" : "ikea", "t
ags" : [ "home decor", "cotton" ] }
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10500, "brand" : "redmi", "t
ags" : "smartphone" }
{ "_id" : ObjectId("60619ff4e7f4b81bf0cc6873"), "name" : "chips", "category" : "food", "price" : 20, "brand" : "lays", "tags" : [ "s
nacks", "food" ] }
> db.product.distinct("category")
[ "containers", "electronics", "food", "home decor" ]
>
```

11. Count
db.product.count()



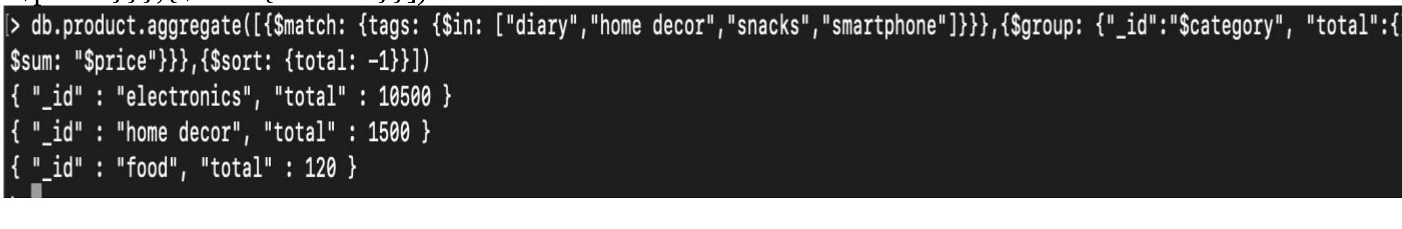
```
> db.product.count()
5
```

db.product.count({price:{$gte: 500}})



```
> db.product.count({price:{$gte: 500}})
3
```

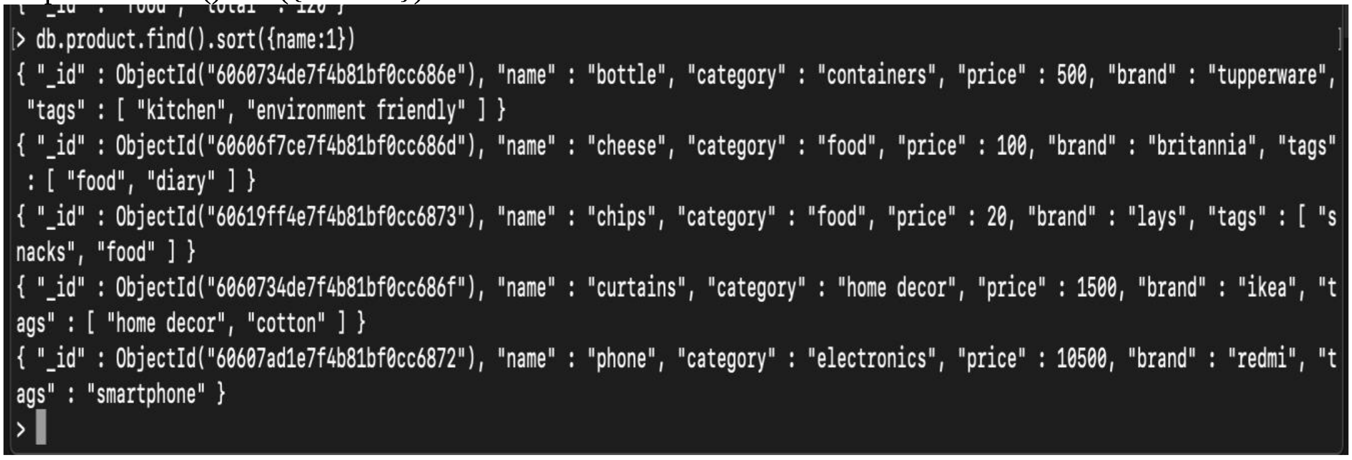
12. Aggregate
db.product.aggregate([{$match: {tags: {$in: ["diary","home decor","snacks","smartphone"]}}},{$group: {"_id":"$category", "total":{$sum: "$price"}}},{$sort: {total: -1}}])



```
> db.product.aggregate([{$match: {tags: {$in: ["diary","home decor","snacks","smartphone"]}}},{$group: {"_id":"$category", "total":{
$sum: "$price"}}},{$sort: {total: -1}}])
{ "_id" : "electronics", "total" : 10500 }
{ "_id" : "home decor", "total" : 1500 }
{ "_id" : "food", "total" : 120 }
```

13. Sort

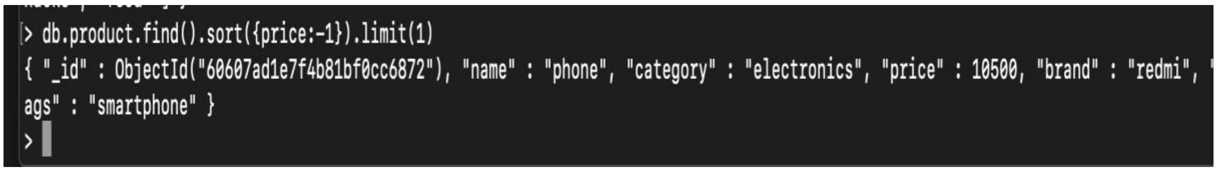
db.product.find().sort({name:1})



```
{ "_id" : "food", "total" : 120 }
> db.product.find().sort({name:1})
{ "_id" : ObjectId("6060734de7f4b81bf0cc686e"), "name" : "bottle", "category" : "containers", "price" : 500, "brand" : "tupperware",
 "tags" : [ "kitchen", "environment friendly" ] }
{ "_id" : ObjectId("60606f7ce7f4b81bf0cc686d"), "name" : "cheese", "category" : "food", "price" : 100, "brand" : "britannia", "tags"
 : [ "food", "diary" ] }
{ "_id" : ObjectId("60619ff4e7f4b81bf0cc6873"), "name" : "chips", "category" : "food", "price" : 20, "brand" : "lays", "tags" : [ "s
nacks", "food" ] }
{ "_id" : ObjectId("6060734de7f4b81bf0cc686f"), "name" : "curtains", "category" : "home decor", "price" : 1500, "brand" : "ikea", "t
ags" : [ "home decor", "cotton" ] }
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10500, "brand" : "redmi", "t
ags" : "smartphone" }
>
```

14. Finding Max value of a particular key in a collection using Limit and Sort:

db.product.find().sort({price:-1}).limit(1)



```
> db.product.find().sort({price:-1}).limit(1)
{ "_id" : ObjectId("60607ad1e7f4b81bf0cc6872"), "name" : "phone", "category" : "electronics", "price" : 10500, "brand" : "redmi", "
ags" : "smartphone" }
>
```

Questions:

1. **Explain the 5 methods used for these queries in detail.**

1. **Count():**

   Count() **method** is used to return the **count** of documents that would match a find() query. The db. collection. **count**() **method** does not perform the find() operation but instead **counts** and returns the number of results that match a query.

2. **Stats():**

   Stats() **method** is used to return a document that reports on the state of the current database. The scale at which to deliver results. Unless specified, this command returns all data in bytes.

3. **Limit():**

   To limit the records in MongoDB, you need to use **limit()** method. The method accepts one number type argument, which is the number of documents that you want to be displayed.

   Syntax

   The basic syntax of **limit()** method is as follows −

   ```
   >db.COLLECTION_NAME.find().limit(NUMBER)
   ```

4. **Sort():**

   To sort documents in MongoDB, you need to use **sort()** method. The method accepts a document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.

   Syntax

   The basic syntax of **sort()** method is as follows −

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

5. **Find():**

To query data from MongoDB collection, you need to use MongoDB's **find()** method.

Syntax

The basic syntax of **find()** method is as follows −

```
>db.COLLECTION_NAME.find()
```

To display the results in a formatted way, you can use **pretty()** method.

Syntax

```
>db.COLLECTION_NAME.find().pretty()
```

Outcomes: (CO2) Design advanced database systems using Object Relational, Spatial and NOSQL Databases and its implementation

---

Conclusion: (Conclusion to be based on outcomes achieved)
Through this experiment we learnt how to implement a NOSQL database using MongoDB and execute queries. After installation of Mongodb we created our own collection in the database and implemented various methods and queries on the collection.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

1. Elmasri and Navathe, “Fundamentals of Database Systems”, Pearson Education

2. Raghu Ramakrishnan and Johannes Gehrke, “Database Management Systems” 3rd Edition, McGraw Hill,2002
3. Korth, Silberchatz, Sudarshan, “Database System Concepts” McGraw Hill

4. http://www.bostongis.com/PrinterFriendly.aspx?content_name=postgis_tut01