Experiment No.3
To install and configure MongoDB to execute NoSQL commands
Date of Performance:31/7/23
Date of Submission : 7/8/23

AIM:

To install and configure MongoDB/ Cassandra/ HBase/ Hypertable and to executeNoSQL commands.

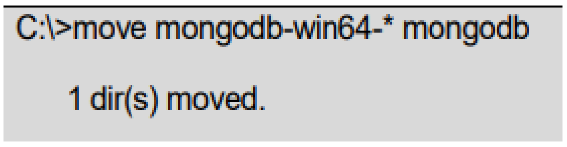
THEORY:

MongoDB can be downloaded from

https://www.mongodb.com/try/download/community2

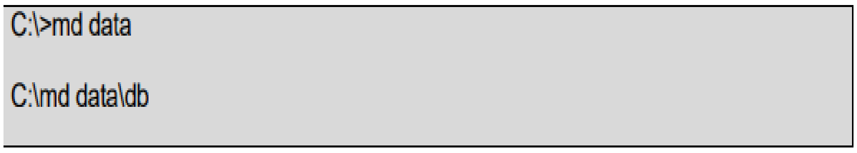
Now open command

prompt and run the following command MongoDB requires a data folder to



```
C:\>move mongodb-win64-* mongodb
     1 dir(s) moved.
```

store files. The default location for the MongoDB datadirectory is c:\data\db. So create the folder using the Command Prompt. Execute the following command sequence.



```
C:\>md data
C:\md data\db
```

In command prompt navigate to the bin directory present into the mongodb

```
C:\Users\XYZ>d:

D:\>cd "set up"

D:\set up>cd mongodb

D:\set up\mongodb>cd bin

D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"
```

installationfolder. Suppose the installation folder is D:\set up\mongodb

Now to run the mongodb, open another command prompt and issue the following command:

```
D:\set up\mongodb\bin>mongo.exe

MongoDB shell version: 2.4.6

connecting to: test

>db.test.save( { a: 1 } )

>db.test.find()

{ "_id" : ObjectId(5879b0f65a56a454), "a" : 1 }

>
```

The use Command

MongoDB use DATABASE_NAME is used to create database. The command will create anew database, if it doesn't exist otherwise it will return the existing database

Syntax:

use DATABASE_NAME

The dropDatabase () Method

MongoDB db.dropDatabase () command is used to drop an existing database.

Syntax:

db.dropDatabase()

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

Syntax:

db.createCollection(name, options)

Insert Document

To insert data into MongoDB collection, you need to use MongoDB's insert() or save()method

Syntax

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

Example:

```
>db.post.insert([
{
 title: 'MongoDB Overview',
 description: 'MongoDB is no sql
 database', tags: ['mongodb',
 'database', 'NoSQL'], likes: 100
},
{
 title: 'NoSQL Database',
 description: 'NoSQL database doesn't have
 tables', tags: ['mongodb', 'database',
 'NoSQL'],
 likes: 20,
 comments: [
 {
 user:'user1',
 message: 'My first comment',
 dateCreated: new
 Date(2022,11,10,2,35),like: 0
 }
 ]])
```

Creating sample document:

Example

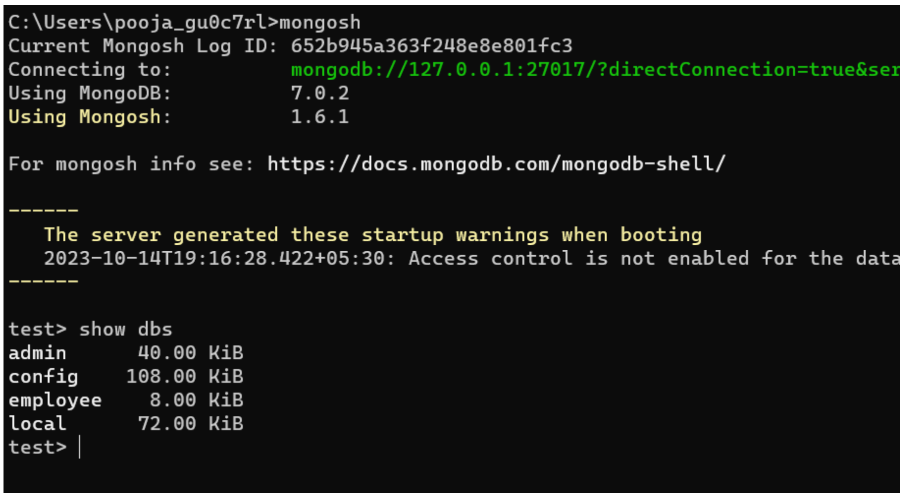
Suppose a client needs a database design for his blog website. Website has the following requirements

- 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 have comments given by users along with their name, message, data-time andlikes.
- On each post there can be zero or more comments.

```
Document:{
_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:
LIKE
```

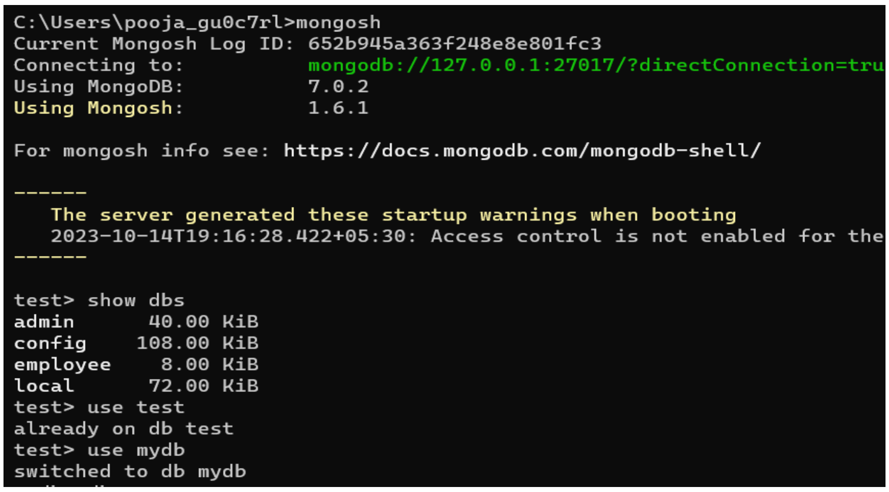
}]}

Show All Databases



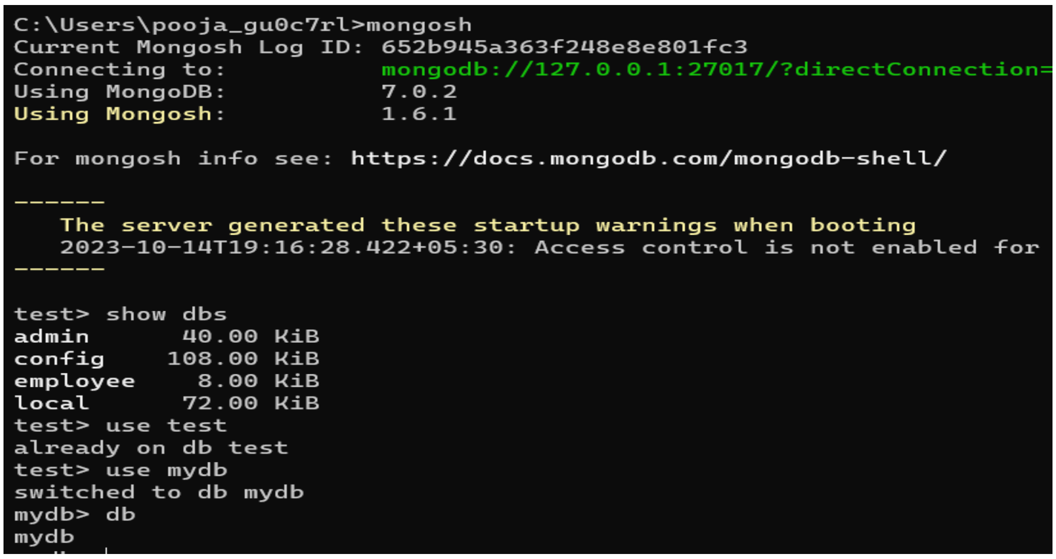
```
C:\Users\pooja_gu0c7rl>mongosh
Current Mongosh Log ID: 652b945a363f248e8e801fc3
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&ser
Using MongoDB:          7.0.2
Using Mongosh:          1.6.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-10-14T19:16:28.422+05:30: Access control is not enabled for the data
------

test> show dbs
admin      40.00 KiB
config    108.00 KiB
employee    8.00 KiB
local      72.00 KiB
test>
```

Create new database:



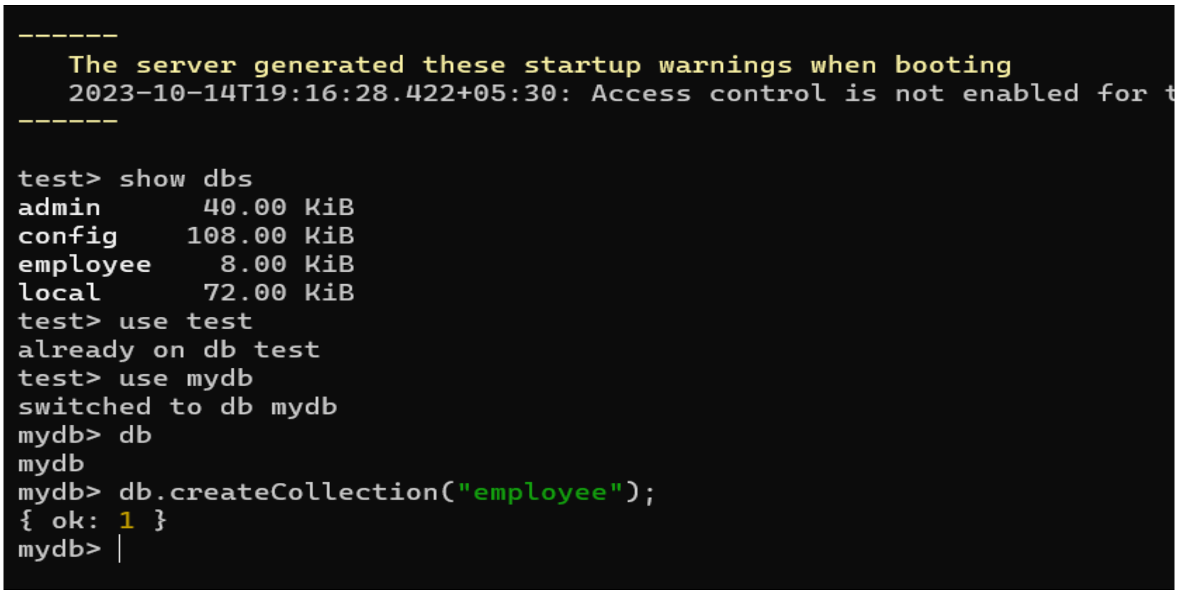
```
C:\Users\pooja_gu0c7rl>mongosh
Current Mongosh Log ID: 652b945a363f248e8e801fc3
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=tru
Using MongoDB:          7.0.2
Using Mongosh:          1.6.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-10-14T19:16:28.422+05:30: Access control is not enabled for the
------

test> show dbs
admin      40.00 KiB
config    108.00 KiB
employee    8.00 KiB
local      72.00 KiB
test> use test
already on db test
test> use mydb
switched to db mydb
```

Know your current selected database:



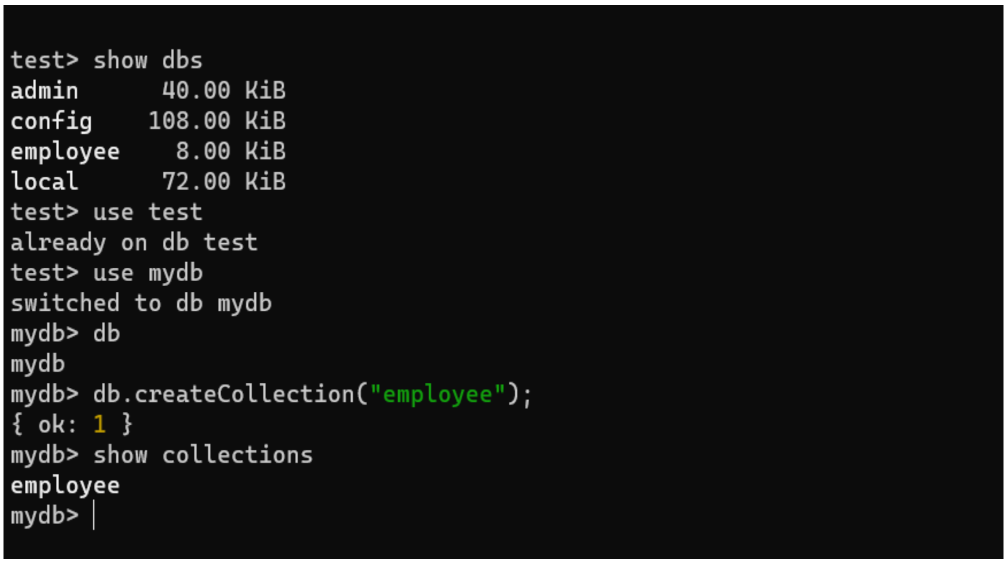
```
C:\Users\pooja_gu0c7rl>mongosh
Current Mongosh Log ID: 652b945a363f248e8e801fc3
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=
Using MongoDB:          7.0.2
Using Mongosh:          1.6.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-10-14T19:16:28.422+05:30: Access control is not enabled for
------

test> show dbs
admin       40.00 KiB
config     108.00 KiB
employee     8.00 KiB
local       72.00 KiB
test> use test
already on db test
test> use mydb
switched to db mydb
mydb> db
mydb
```

Create collection:



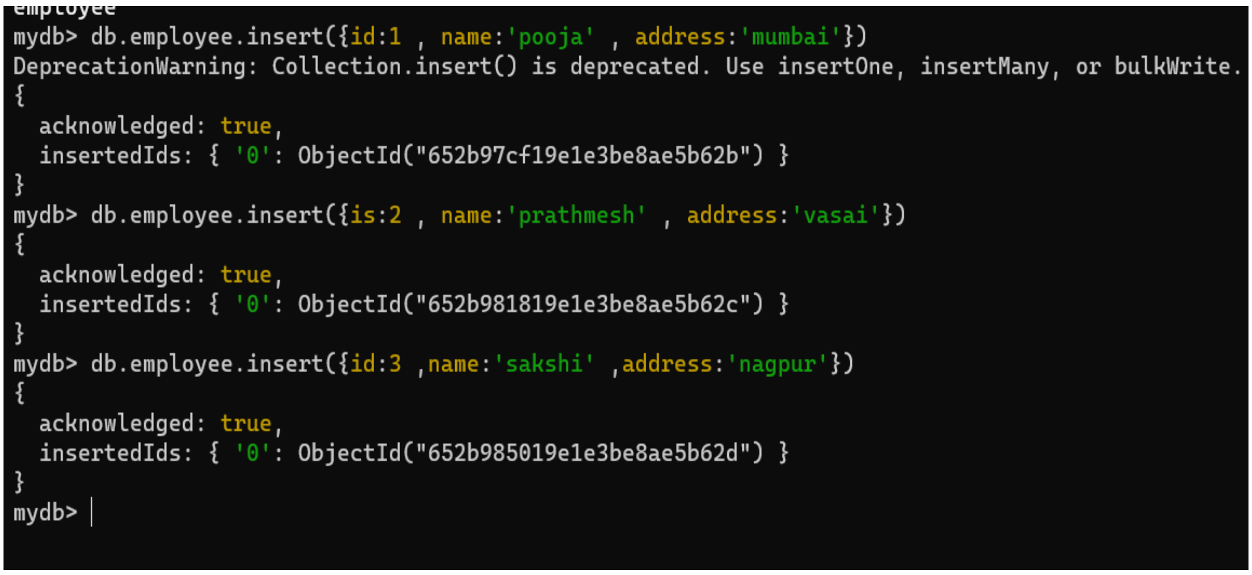
```
------
   The server generated these startup warnings when booting
   2023-10-14T19:16:28.422+05:30: Access control is not enabled for t
------

test> show dbs
admin       40.00 KiB
config     108.00 KiB
employee     8.00 KiB
local       72.00 KiB
test> use test
already on db test
test> use mydb
switched to db mydb
mydb> db
mydb
mydb> db.createCollection("employee");
{ ok: 1 }
mydb> |
```

To check collections list



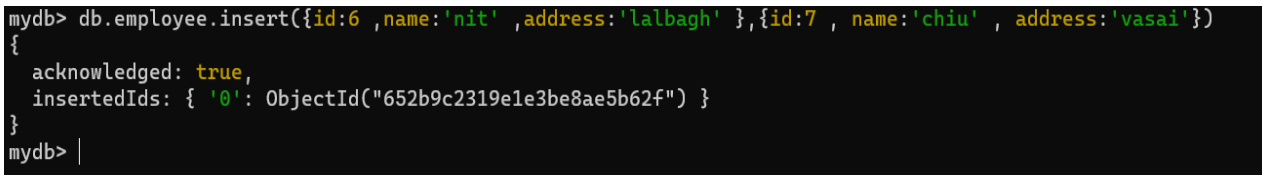
```
test> show dbs
admin       40.00 KiB
config     108.00 KiB
employee     8.00 KiB
local       72.00 KiB
test> use test
already on db test
test> use mydb
switched to db mydb
mydb> db
mydb
mydb> db.createCollection("employee");
{ ok: 1 }
mydb> show collections
employee
mydb> |
```

Insert document in collection



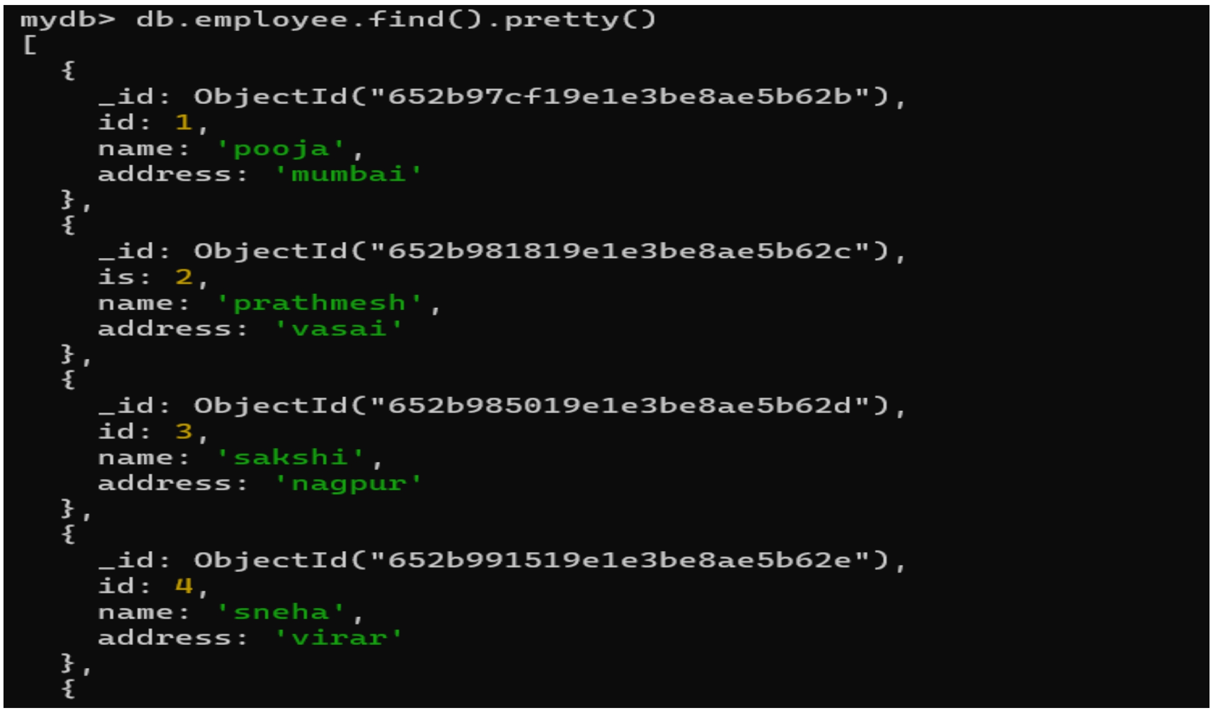
```
employee
mydb> db.employee.insert({id:1 , name:'pooja' , address:'mumbai'})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("652b97cf19e1e3be8ae5b62b") }
}
mydb> db.employee.insert({is:2 , name:'prathmesh' , address:'vasai'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("652b981819e1e3be8ae5b62c") }
}
mydb> db.employee.insert({id:3 ,name:'sakshi' ,address:'nagpur'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("652b985019e1e3be8ae5b62d") }
}
mydb> |
```

To insert multiple documents in selected collection



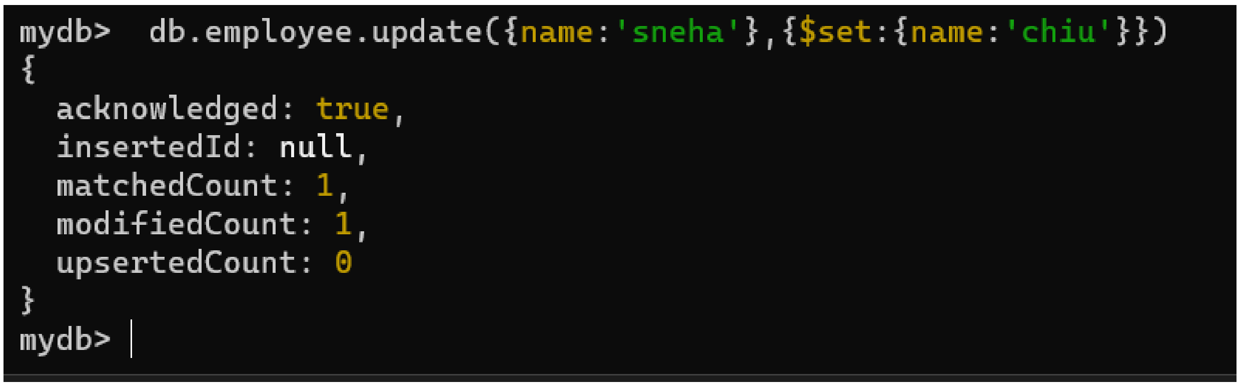
```
mydb> db.employee.insert({id:6 ,name:'nit' ,address:'lalbagh' },{id:7 , name:'chiu' , address:'vasai'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("652b9c2319e1e3be8ae5b62f") }
}
mydb> |
```

Get collection document

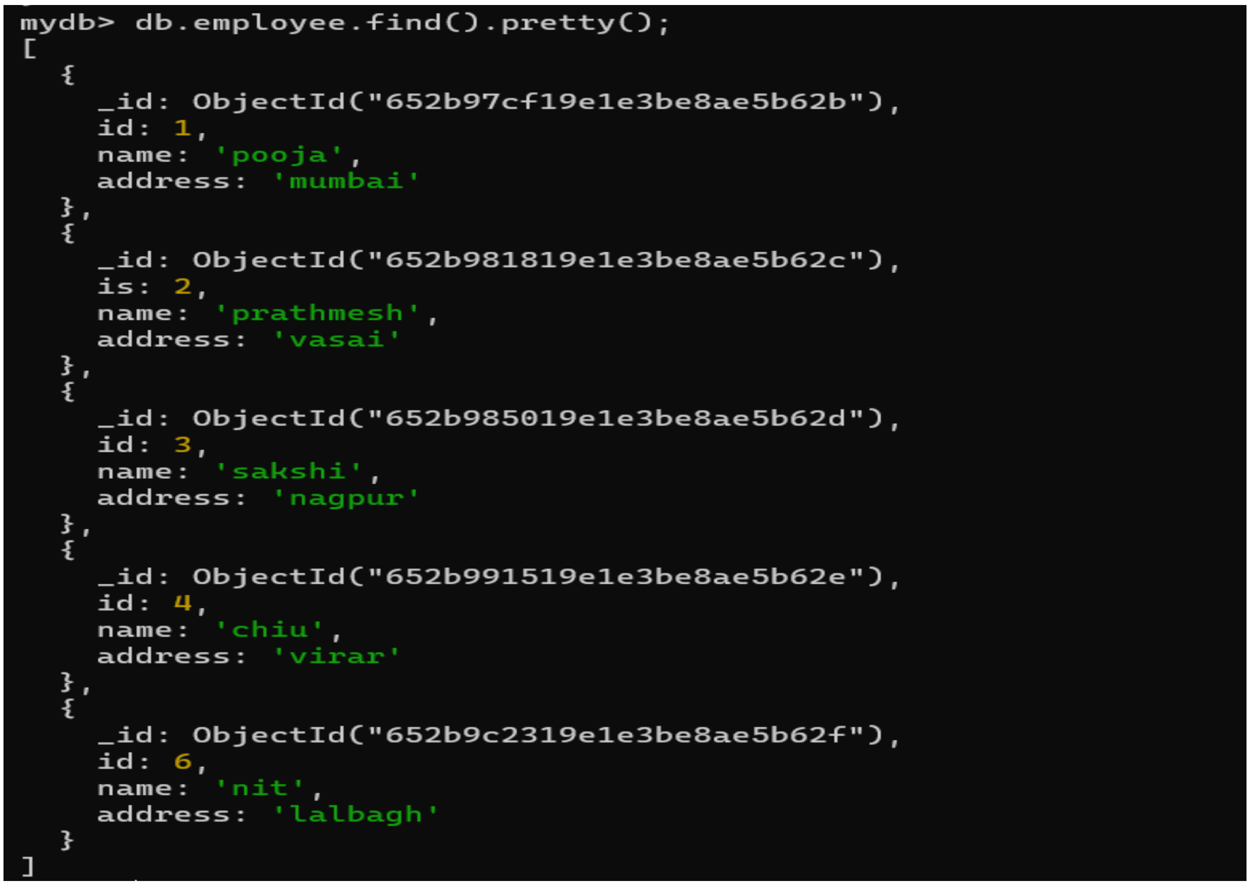


```
mydb> db.employee.find().pretty()
[
  {
    _id: ObjectId("652b97cf19e1e3be8ae5b62b"),
    id: 1,
    name: 'pooja',
    address: 'mumbai'
  },
  {
    _id: ObjectId("652b981819e1e3be8ae5b62c"),
    is: 2,
    name: 'prathmesh',
    address: 'vasai'
  },
  {
    _id: ObjectId("652b985019e1e3be8ae5b62d"),
    id: 3,
    name: 'sakshi',
    address: 'nagpur'
  },
  {
    _id: ObjectId("652b991519e1e3be8ae5b62e"),
    id: 4,
    name: 'sneha',
    address: 'virar'
  },
  {
```

Update document

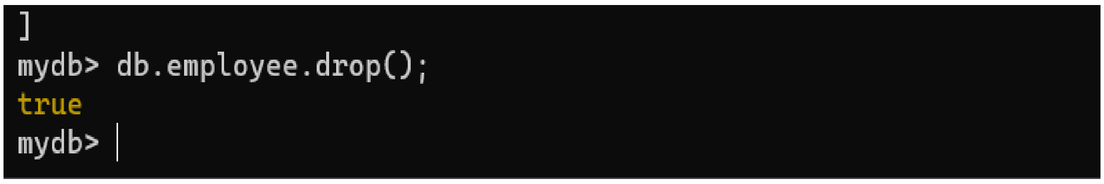


```
mydb>  db.employee.update({name:'sneha'},{$set:{name:'chiu'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mydb>
```



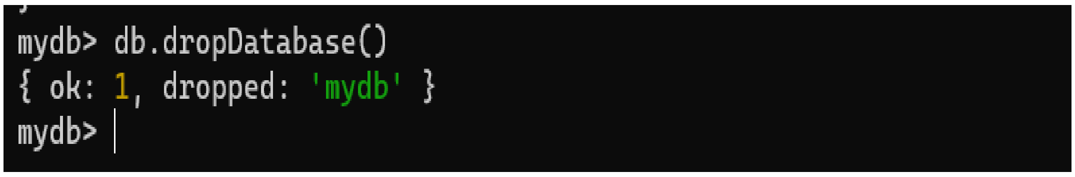
```
mydb> db.employee.find().pretty();
[
  {
    _id: ObjectId("652b97cf19e1e3be8ae5b62b"),
    id: 1,
    name: 'pooja',
    address: 'mumbai'
  },
  {
    _id: ObjectId("652b981819e1e3be8ae5b62c"),
    is: 2,
    name: 'prathmesh',
    address: 'vasai'
  },
  {
    _id: ObjectId("652b985019e1e3be8ae5b62d"),
    id: 3,
    name: 'sakshi',
    address: 'nagpur'
  },
  {
    _id: ObjectId("652b991519e1e3be8ae5b62e"),
    id: 4,
    name: 'chiu',
    address: 'virar'
  },
  {
    _id: ObjectId("652b9c2319e1e3be8ae5b62f"),
    id: 6,
    name: 'nit',
    address: 'lalbagh'
  }
]
```

Drop collection



```
]
mydb> db.employee.drop();
true
mydb> 
```

Drop database



```
mydb> db.dropDatabase()
{ ok: 1, dropped: 'mydb' }
mydb> 
```

CONCLUSION:

To run NoSQL commands, the experiment's goal was to install and set up MongoDB. Installing and configuring MongoDB to satisfy particular needs—such as security precautions and system specifications—was accomplished effectively. Data insertion, querying, indexing, and other database operations were among the many NoSQL commands we learnt to use. MongoDB is a good fit for NoSQL applications because of its scalability and performance with unstructured data. An effective experiment with useful abilities for effective NoSQL data management with MongoDB was made possible by the provision of copious documentation and community support.