## National University of Computer & Emerging Sciences, Karachi



## FAST School of Computing –Computer Science Department

**Fall 2024, Lab Manual – 11**

|  |  |
| --- | --- |
| **Course Code: CL-2005** | **Course : Database Systems Lab** |
| **Instructor(s) :** | **Mr. Mubashir** |

**Contents:**

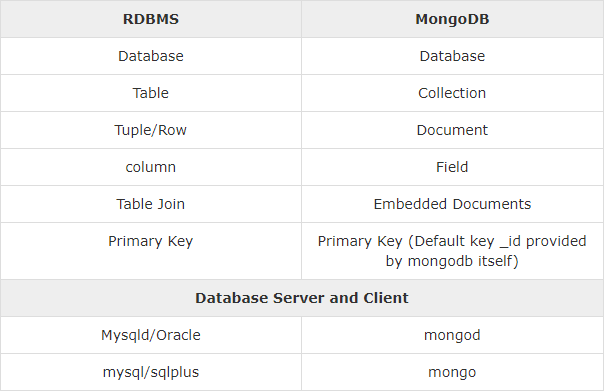
1. Overview of MongoDB
2. Difference in terminology of MongoDB
3. Installation of MongoDB
4. Designing Schema in MongoDB
5. Some important methods in MongoDB
6. Creating Database
7. Creating collections
8. Inserting single/multiple Documents
9. Querying, Deleting & updating of Documents
10. Logical Operations
11. Implementation of where clause

**Overview of MongoDB**

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This manual will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

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.

**Difference in Terminology of MongoDB**



**Figure 1(Difference between RDBMS & MongoDB)**

## 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.

**Install MongoDB on Windows**

To perform the installation of MongoDB do following the below steps:

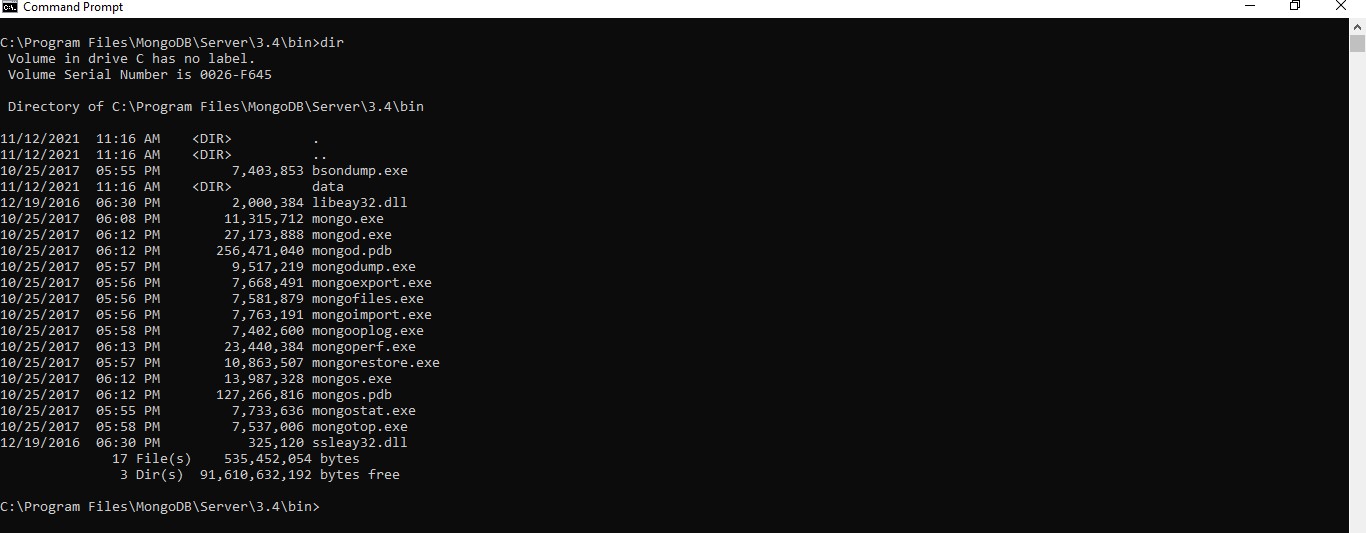
* 1. First of all, open your windows command prompt.



* 1. Change the directory to the MongoDB directory. As I have MongoDB in C: Program Files

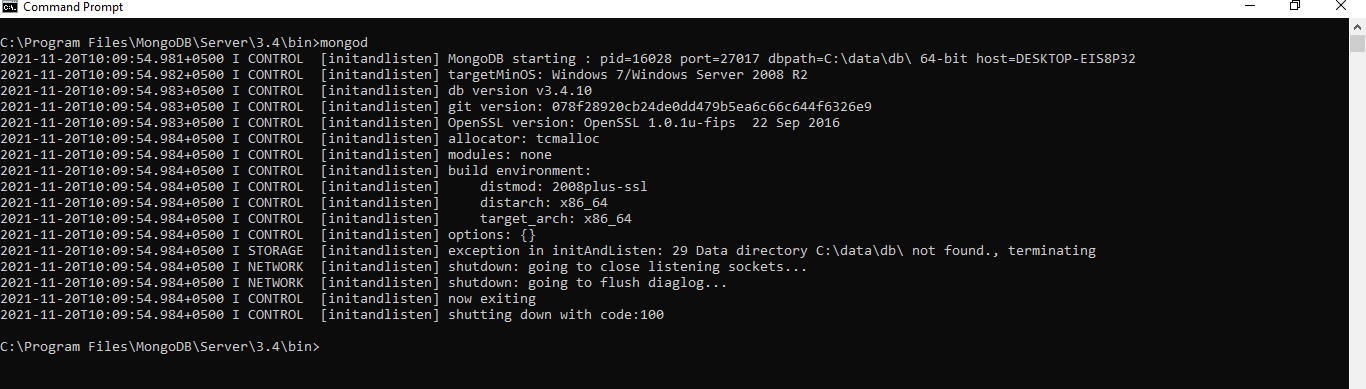


* 1. Now further move to the sub sub directory of the mongodb i.e. bin directory
  2. Now run **dir** command on bin directory i.e. sub sub directory of mongoDB and match the number files as shown in the given screenshot

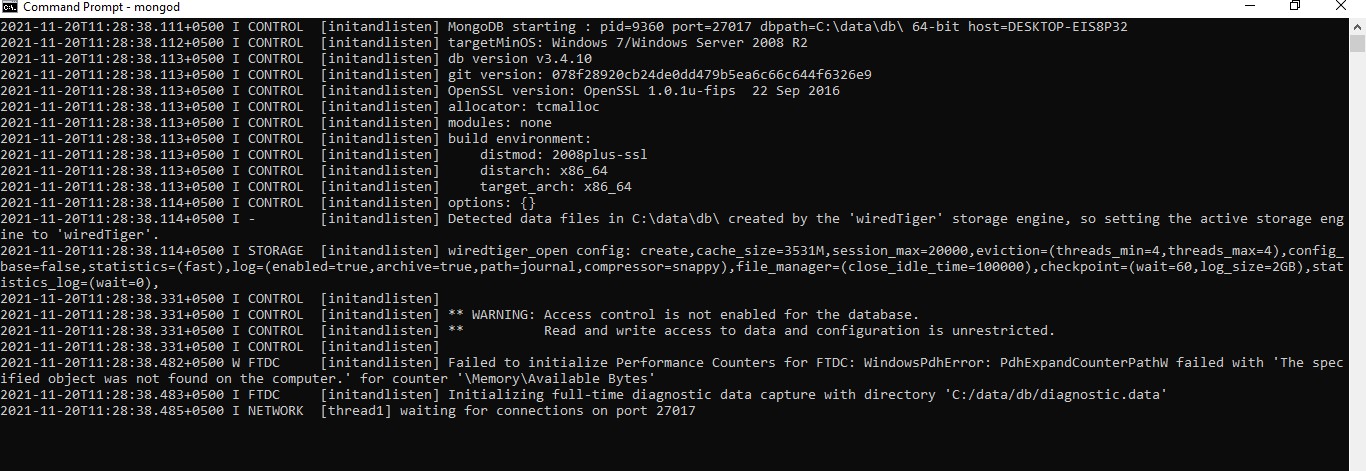


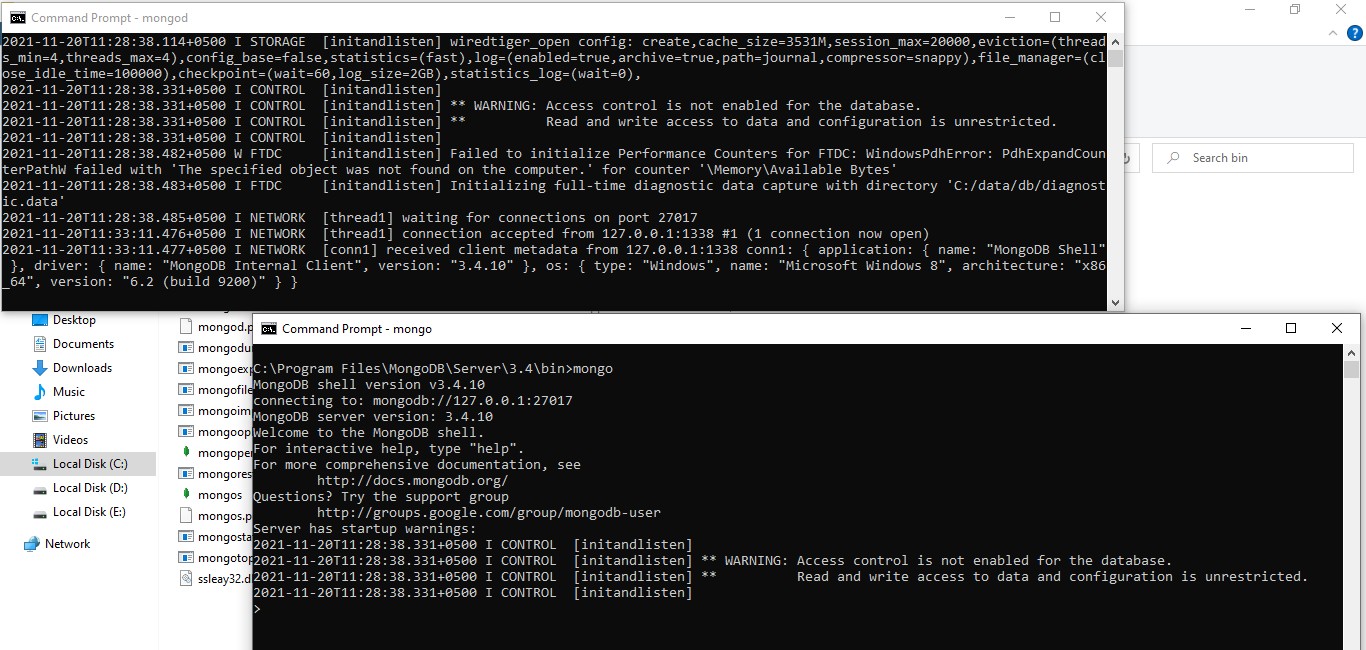
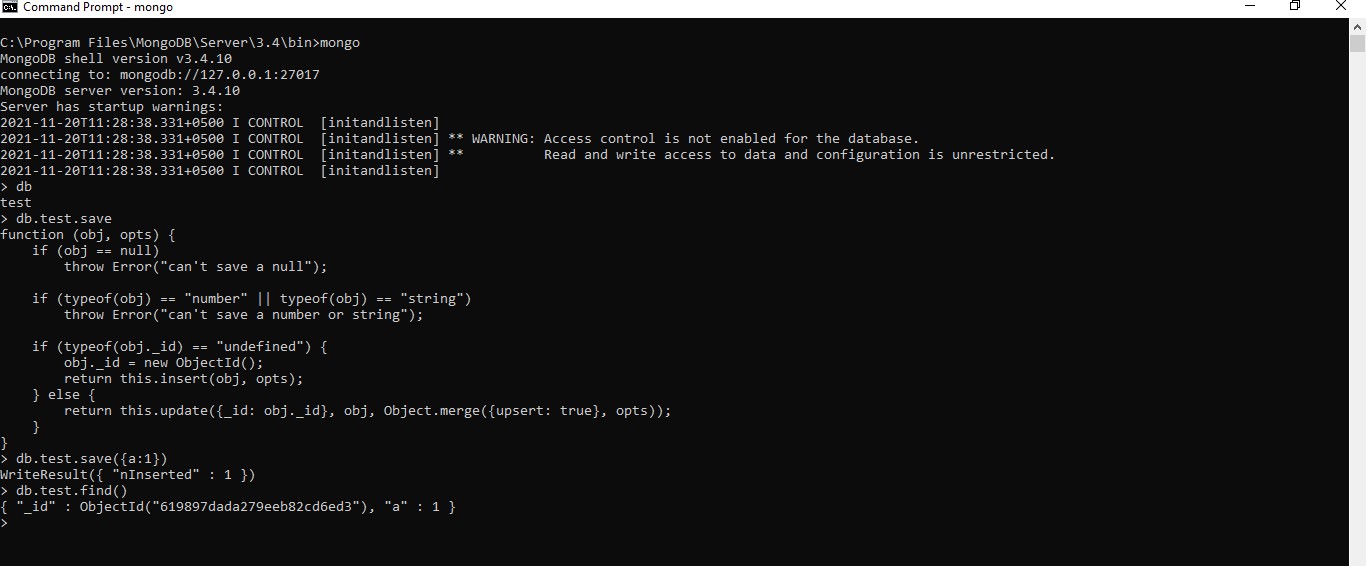
* 1. Now run **mongod** command. There is an error in the given screenshot while running

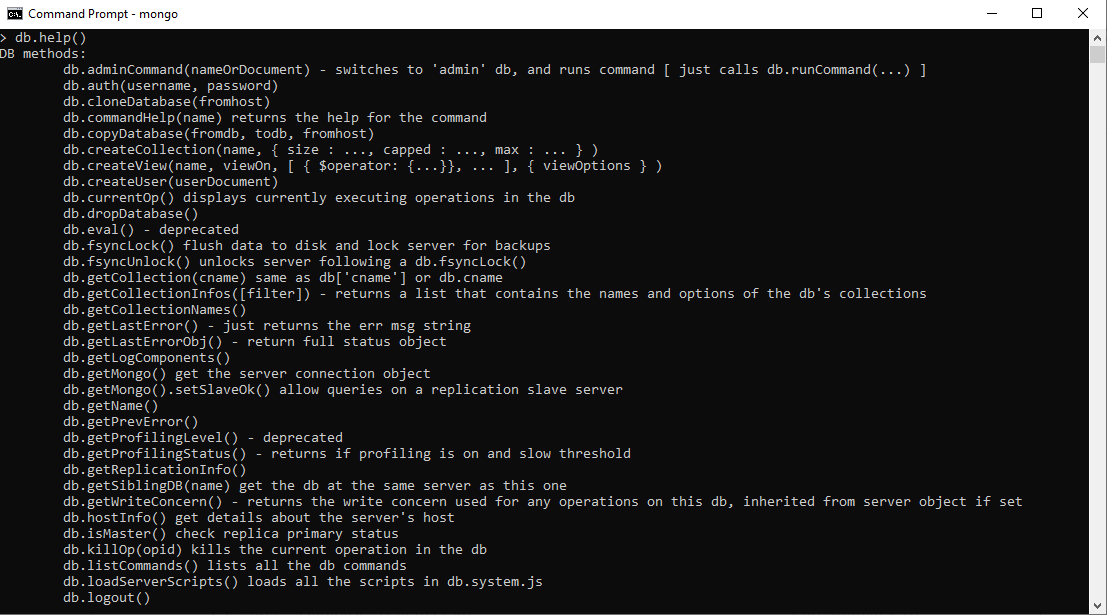
**mongod** command so in order to handle this problem first we need to run **mkdir**

**\data\db** then **mongod** command

* 1. Running **mkdir \data\db** command
  2. Now again run **mongod** command then connection is now waiting to connect as shown in the screenshot



* 1. Now we are establishing the **mongod** and **mongo** connection in the given screenshot
  2. After successful connection we will check the already made Database
  3. Now running **db.help()** to see different helping functions



* 1. Now run **db.stats()** to see the statistics of the database

# Some Consideration While Designing Schema in MongoDB

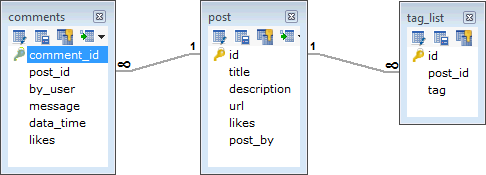
1. Design your schema according to user requirements.
2. Combine objects into one document if you will use them together. Otherwise separate them (but make sure there should not be need of joins).
3. Duplicate the data (but limited) because disk space is cheap as compare to compute time.
4. Do joins while write, not on read.
5. Optimize your schema for most frequent use cases.
6. 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.

1. Every post has the unique title, description and url.
2. Every post can have one or more tags.
3. Every post has the name of its publisher and total number of likes.
4. Every post has comments given by users along with their name, message, data-time and likes.
5. 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:



So while showing the data, in RDBMS you need to join three tables and in MongoDB, data will be shown from one collection only.

# MongoDB – Create Database

MongoDB **use DATABASE\_NAME** is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

### Syntax

Basic syntax of **use DATABASE** statement is as follows − use DATABASE\_NAME

### Command:

>show dbs

If you want to check your databases list, use the command **show dbs**.

Your created database (myfirstdb) is not present in list. To display database, you need to insert at least one document into it.

> db.myfirstdb.insert({"name":"Amin Sadiq"}) WriteResult({ "nInserted" : 1 })



# The dropDatabase() Method

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

### Syntax

Basic syntax of dropDatabase() command is as follows −

> db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example

If you want to delete new database <mydb>, then dropDatabase() command would be as

follows −

>use myfirstdb

// Switched to db mydb

>db.dropDatabase()

{ "dropped" : "myfirstdb", "ok" : 1 }

Now check list of databases.

>show dbs

# The createCollection() Method

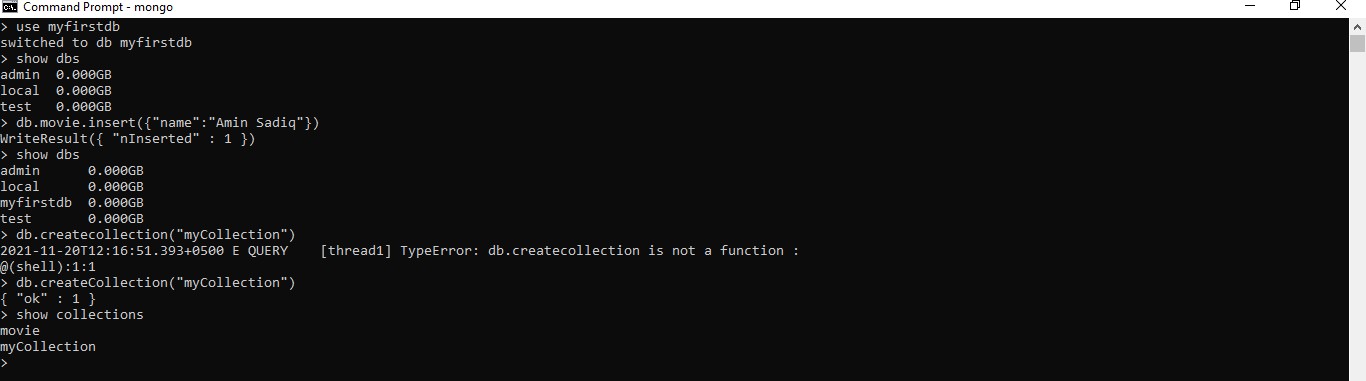
MongoDB db.createCollection(name, options) is used to create collection.

Syntax

Basic syntax of createCollection() command is as follows –

db.createCollection(name, options)

In the command, name is name of collection to be created. Options (Optional Parameter)is a document and is used to specify configuration of collection.



# MongoDB – Drop Collection

### The drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database

### Syntax

Basic syntax of **drop()** command is as follows −

db.COLLECTION\_NAME.drop()

# MongoDB – Insert Document

### The insert() Method

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

### Syntax

The basic syntax of insert() command is as follows −

>db.COLLECTION\_NAME.insert(document)

# Example

> db.mycollection.insert({ title: 'MongoDB Overview',

description: 'MongoDB is no sql database', by: 'Amin Sadiq',

url: 'http://www.gmail.com',

tags: ['mongodb', 'database', 'NoSQL'], likes: 100

})



Notepad: Here **mycollection** is our collection name, as created in the previous slide. If the collection doesn't exist in the database, then MongoDB will create this collection and then insert a document into it.

\_id parameter

In the inserted document, if we don't specify the \_id parameter, then MongoDB assigns a unique ObjectId for this document.

# Insert Document multiple document

To insert multiple documents in a single query, you can pass an array of documents in insert() command.

db.mycollection.insert([

{

title: 'MongoDB Overview',

description: 'MongoDB is no sql database', by: 'Amin Sadiq',

url: 'http://www.gmail.com',

tags: ['mongodb', 'database', 'NoSQL'], likes: 100

},

{

title: 'NoSQL Database',

description: "NoSQL database doesn't have tables", by: 'Ali Shah Fatmi',

url: 'http://www.gmail.com',

tags: ['mongodb', 'database', 'NoSQL'], likes: 20,

comments: [

{

user:'user1',

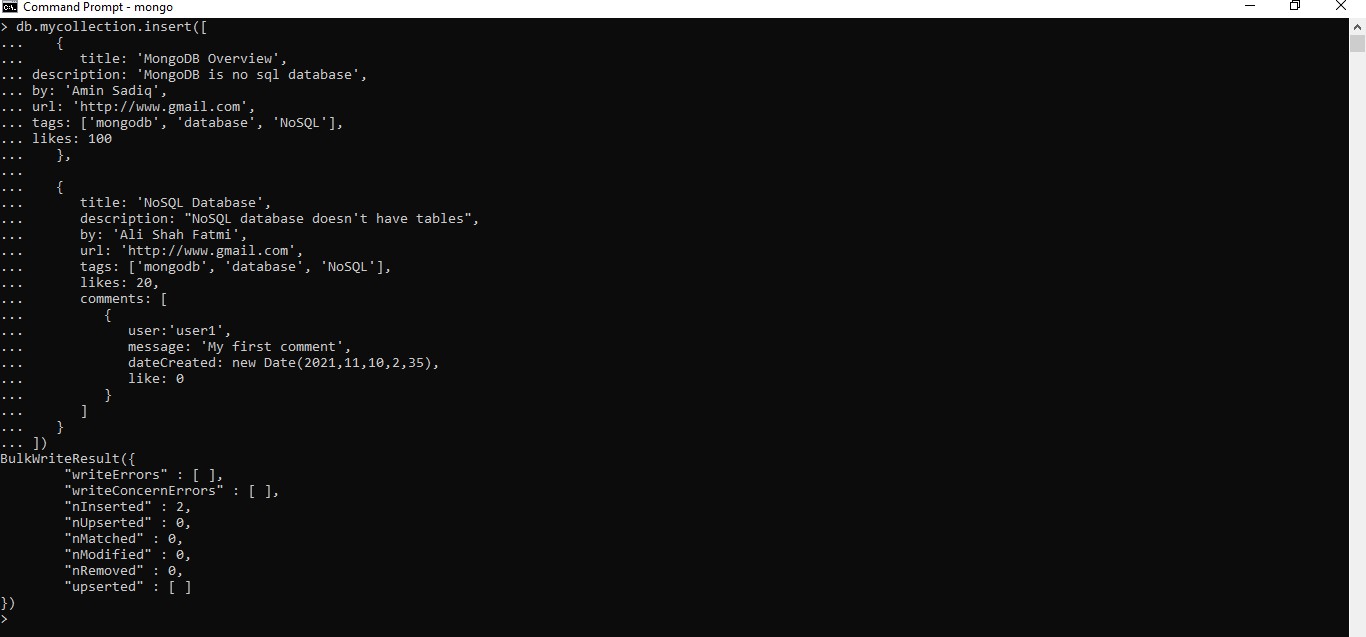
message: 'My first comment', dateCreated: new Date(2021,11,10,2,35), like: 0

}

]

}

])



# MongoDB – Query Document

### The find() Method

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()

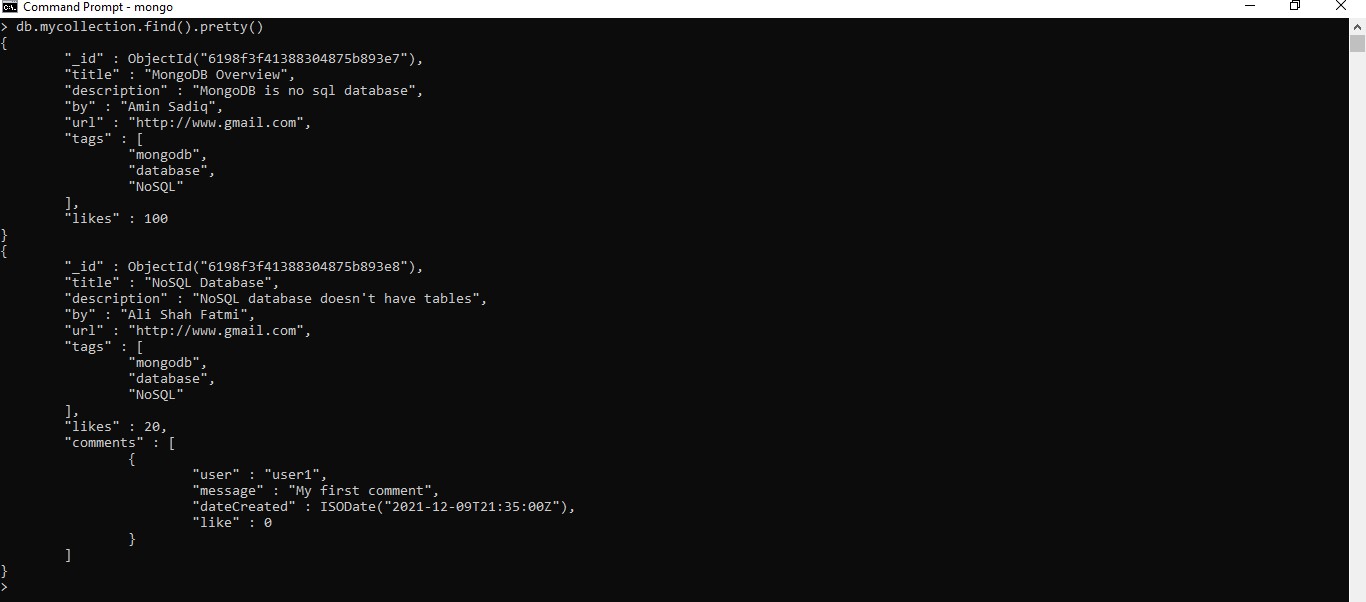
Note: **find()** method will display all the documents in a non-structured way.

### The pretty() Method

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

### Syntax

* >db.mycol.find().pretty()



# RDBMS Where Clause Equivalents in MongoDB

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Syntax** | **Example** | **RDBMS**  **Equivalent** |
| **Equality** | {<key>:<value>} | db.mycollection.find({"by":"Amin Sadiq"}).pretty() | where by = ‘Amin Sadiq' |
| **Less Than** | {<key>:{$lt:<value>}} | db.mycollection.find({"likes":{$lt:50}}).prett  y() | where likes < 50 |
| **Less Than Equals** | {<key>:{$lte:<value>}} | db.mycollection.find({"likes":{$lte:50}}).pre tty() | where likes <= 50 |
| **Greater Than** | {<key>:{$gt:<value>}} | db.mycollection.find({"likes":{$gt:50}}).pret ty() | where likes > 50 |
| **Greater Than Equals** | {<key>:{$gte:<value>}} | db.mycollection.find({"likes":{$gte:50}}).pr etty() | where likes >= 50 |
| **Not Equals** | {<key>:{$ne:<value>}} | db.mycollection.find({"likes":{$ne:50}}).pre tty() | where likes != 50 |

**AND in MongoDB**

### Syntax

In the find() method, if you pass multiple keys by separating them by ',' then MongoDB treats it as AND condition. Following is the basic syntax of AND –

db.mycol.find(

{

$and: [

{key1: value1}, {key2:value2}

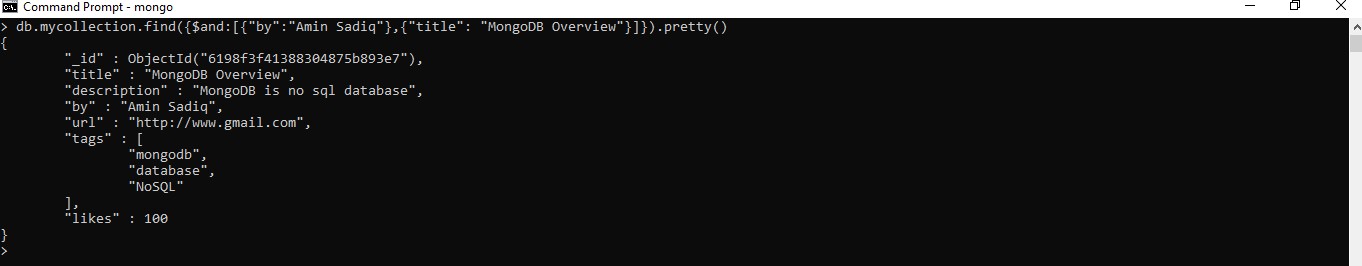
]

}

).pretty()

Example

Following example will show all the tutorials written by ‘Amin Sadiq' and whose title is 'MongoDB Overview'.

db.mycollection.find({$and:[{"by":"Amin Sadiq"},{"title": "MongoDB Overview"}]}).pretty()

# OR in MongoDB

### Syntax

* To query documents based on the OR condition, you need to use $or keyword. Following is the basic syntax of OR –

db.mycol.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}

).pretty()

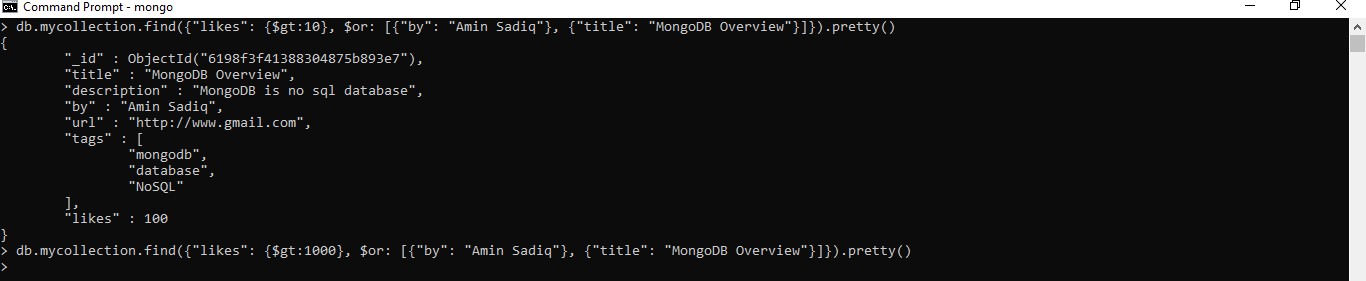
Example

Following example will show all the tutorials written by ‘Amin Sadiq' or whose title is 'MongoDB Overview'.

db.mycollection.find({$or:[{"by":"Ali Fatimi"},{"title": "MongoDB Overview"}]}).pretty()

# Using AND & OR together in MongoDB

The following example will show the documents that have likes greater than 10 and whose title is either 'MongoDB Overview' or by is ‘Amin Sadiq'. Equivalent SQL where clause is 'where likes>10 AND (by = ‘Amin Sadiq' OR title = 'MongoDB Overview')'

db.mycollection.find({"likes": {$gt:10}, $or: [{"by": "Amin Sadiq"}, {"title": "MongoDB Overview"}]}).pretty()

# MongoDB – Update Document

### MongoDB Update() Method

The update() method updates the values in the existing document

### Syntax

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

db.COLLECTION\_NAME.update(SELECTION\_CRITERIA, UPDATED\_DATA)

Example

Consider the mycollection collection has the following data.

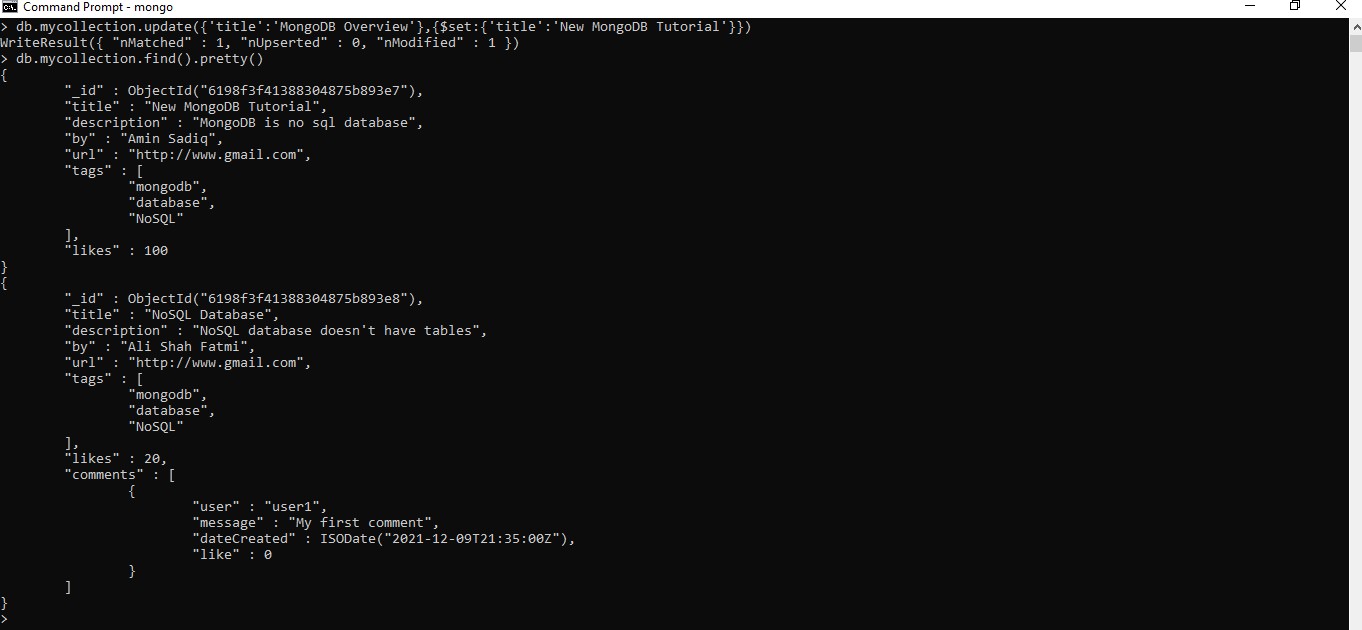
{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is 'MongoDB Overview'.

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



# MongoDB – Delete Document

### MongoDB remove() Method

* MongoDB's remove() method is used to remove a document from the collection. remove() method accepts two parameters. One is deletion criteria and second is justOne flag.
* deletion criteria − (Optional) deletion criteria according to documents will be removed.
* justOne − (Optional) if set to true or 1, then remove only one document.

### Syntax

Basic syntax of remove() method is as follows –

db.COLLECTION\_NAME.remove(DELETION\_CRITERIA)

Example

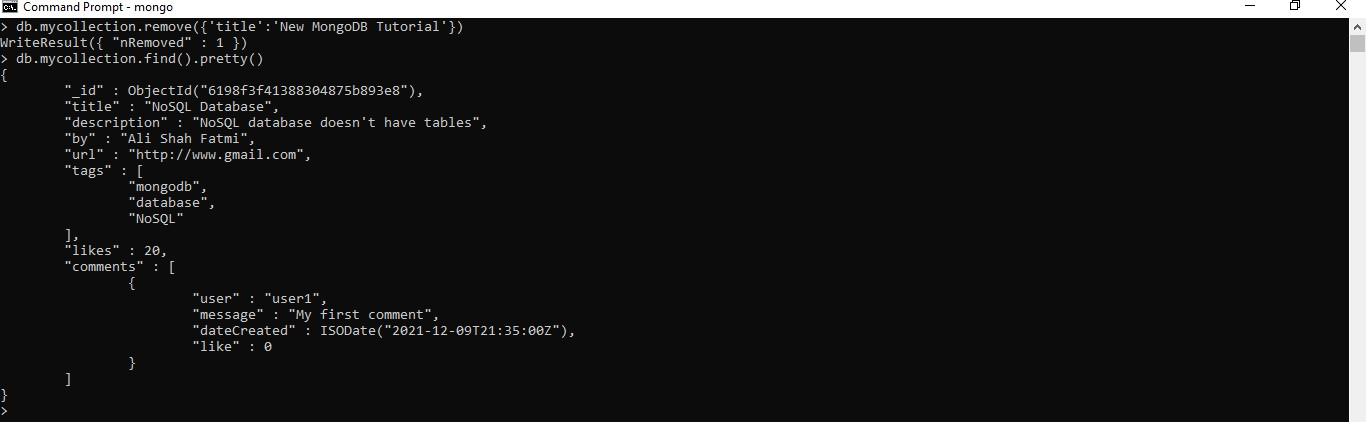
Let’s consider the mycollection collection has the following data.

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will remove all the documents whose title is 'MongoDB Overview'. db.mycollection.remove({'title':'New MongoDB Tutorial'})



**Lab Tasks:**

Consider the Structure of 'restaurants' collection given below:

{

"address": {

"building": "1007",

"coord": [ -73.856077, 40.848447 ],

"street": "Morris Park Ave",

"zipcode": "10462"

},

"borough": "Bronx",

"cuisine": "Bakery",

"grades": [

{ "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },

{ "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },

{ "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },

{ "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },

{ "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }

],

"name": "Morris Park Bake Shop",

"restaurant\_id": "30075445"

}

1. Write a MongoDB query to display all the documents in the collection restaurants.
2. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant.
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.
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.
5. Write a MongoDB query to display all the restaurant which is in the borough Bronx.
6. Write a MongoDB query to display the first 5 restaurants which is in the borough Bronx.
7. Write a MongoDB query to find the restaurants who achieved a score more than 90.
8. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.
9. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.
10. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.