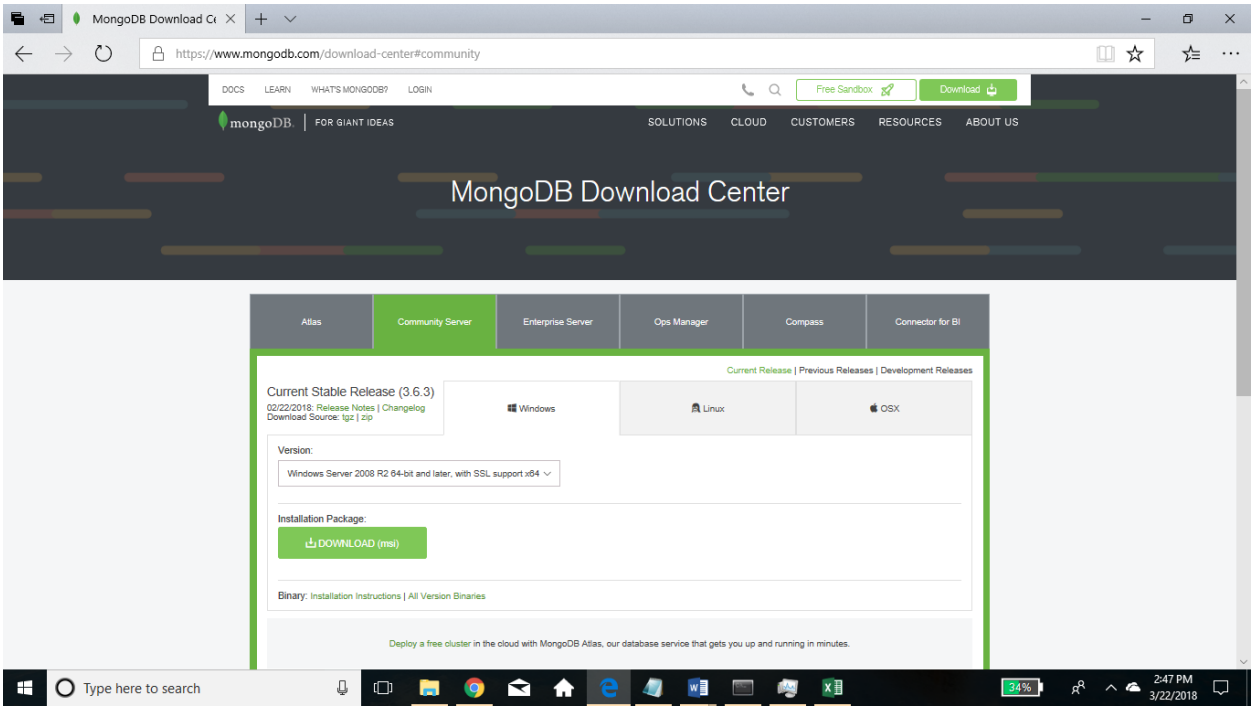
Introduction to MongoDB

In this word document we dive into the MongoDB NoSQL database and look at the fundamentals and the syntax to create, read, update and delete documents/data

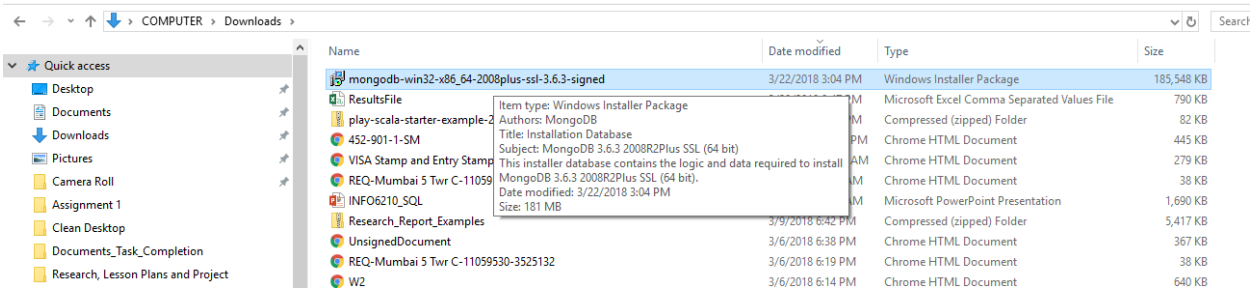
Installation:

Go to https://www.mongodb.com/download-center#community and click on community server.

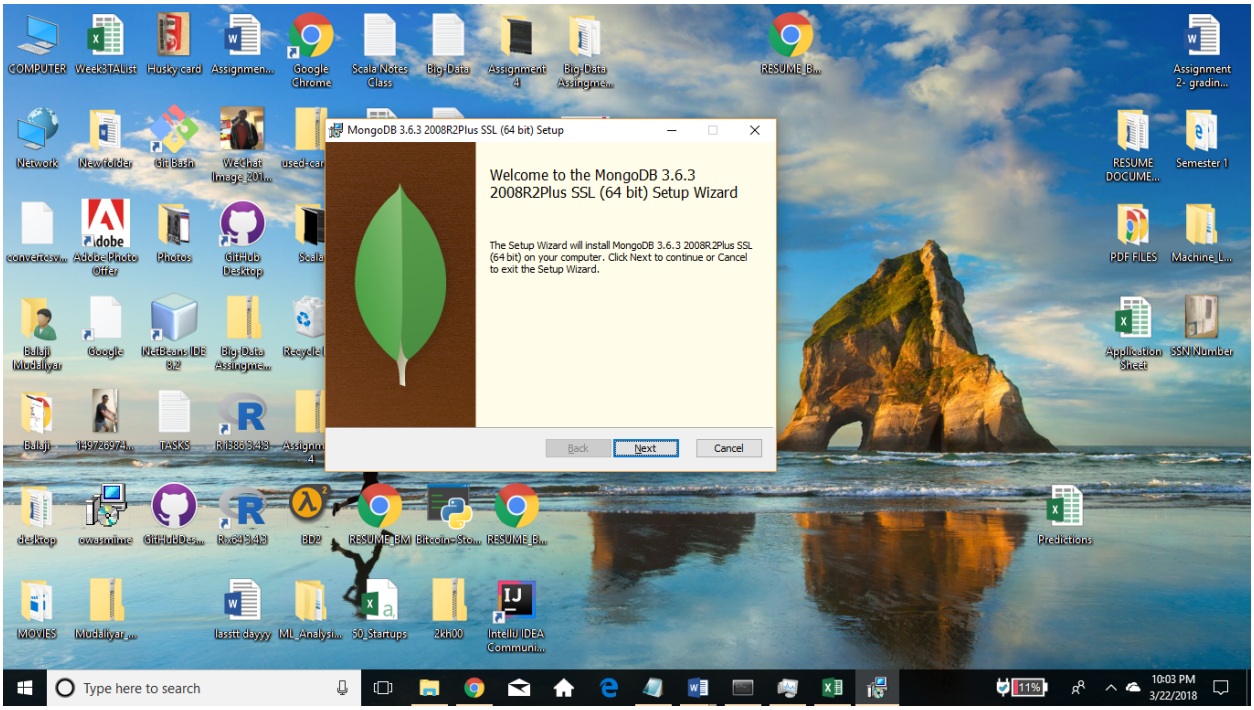
If your using windows, then download the package under windows. (Refer Below)



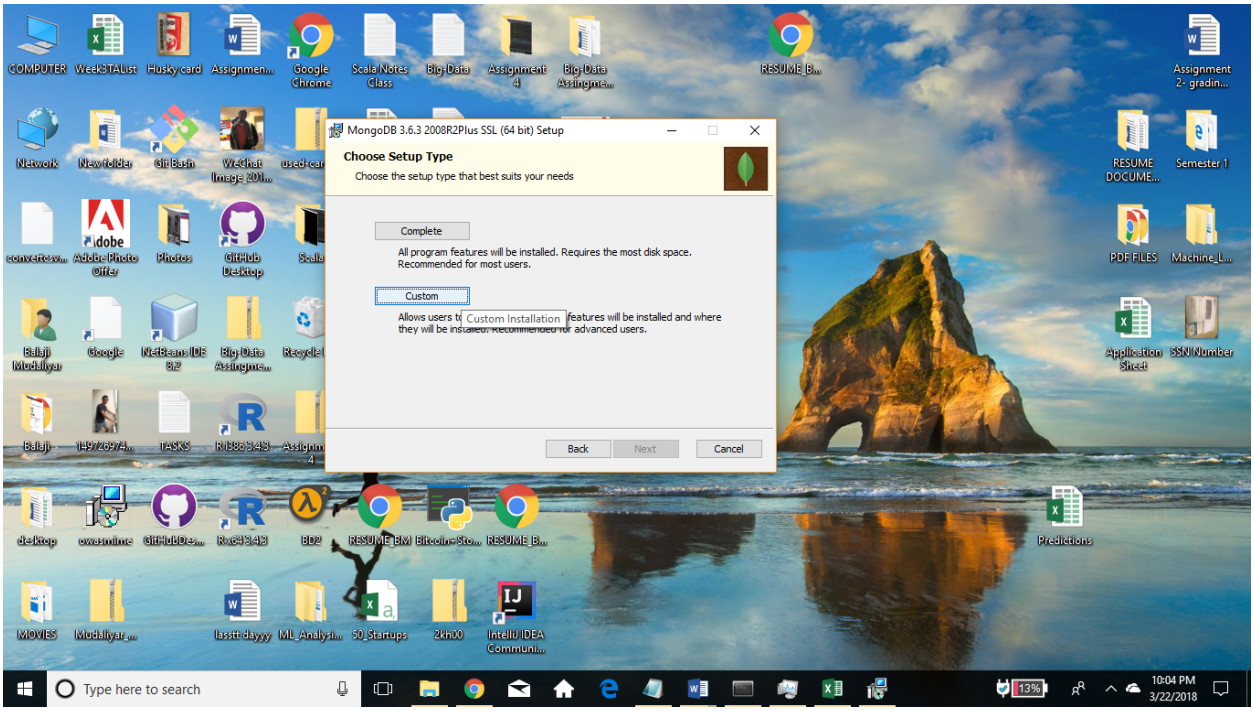
After downloading install it.



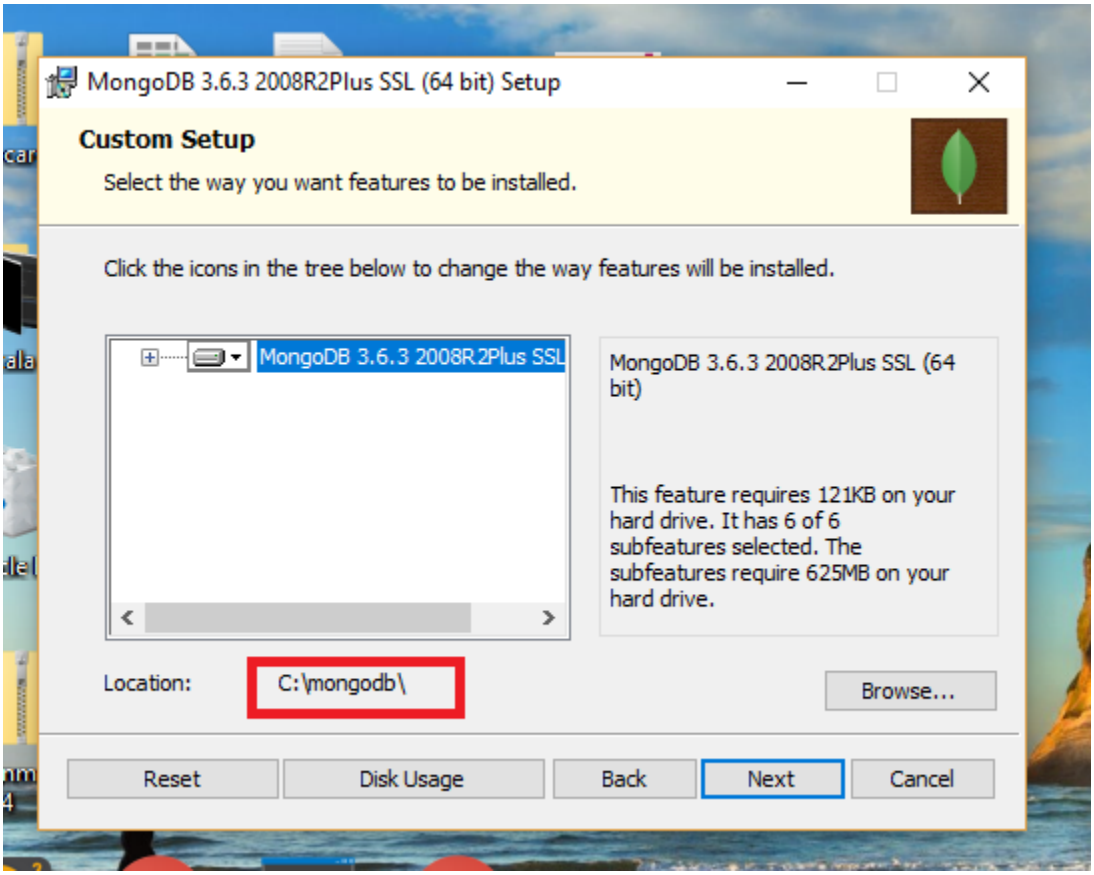
Installation steps:

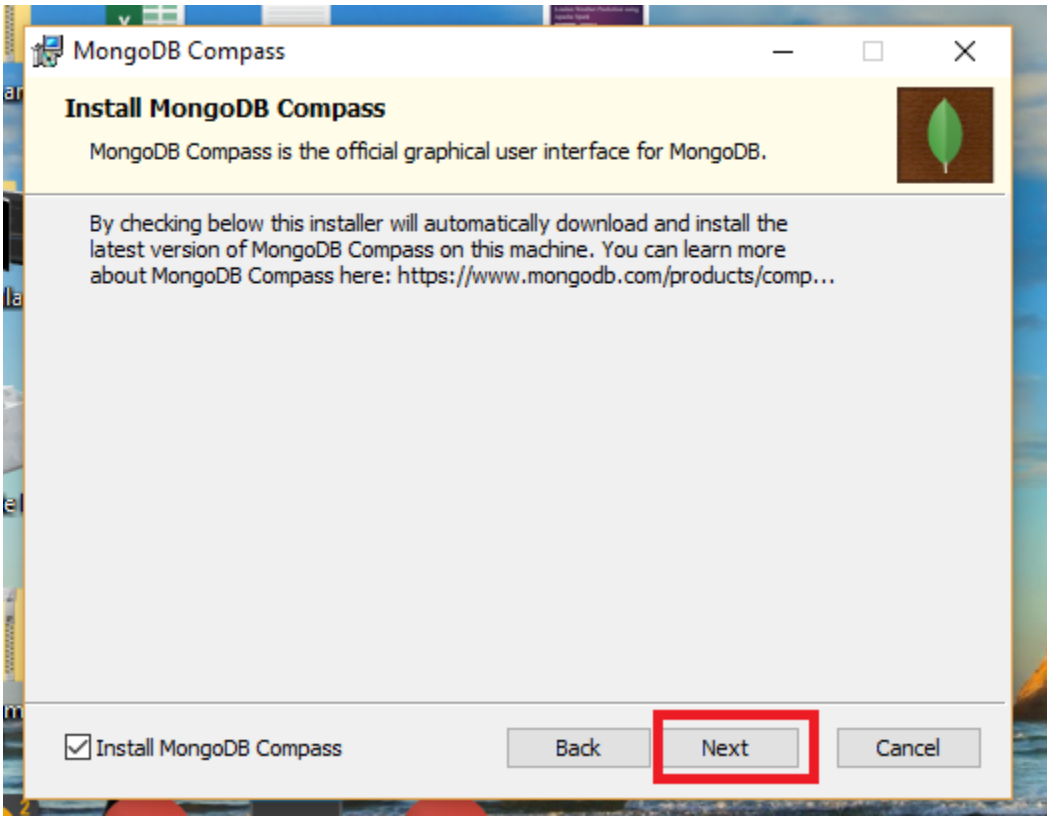


2. Choose Custom in Setup Types so that we can change the location.

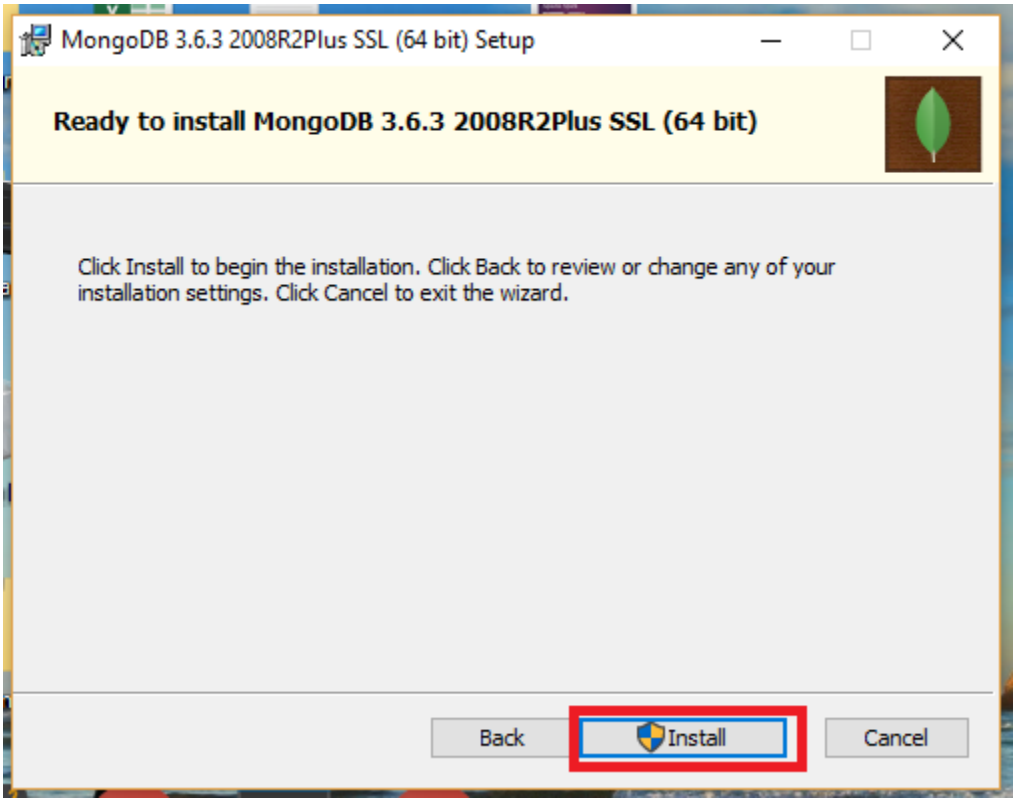


3. Create a file called **mongodb** in your C drive (you can skip this step). Click **Next**.

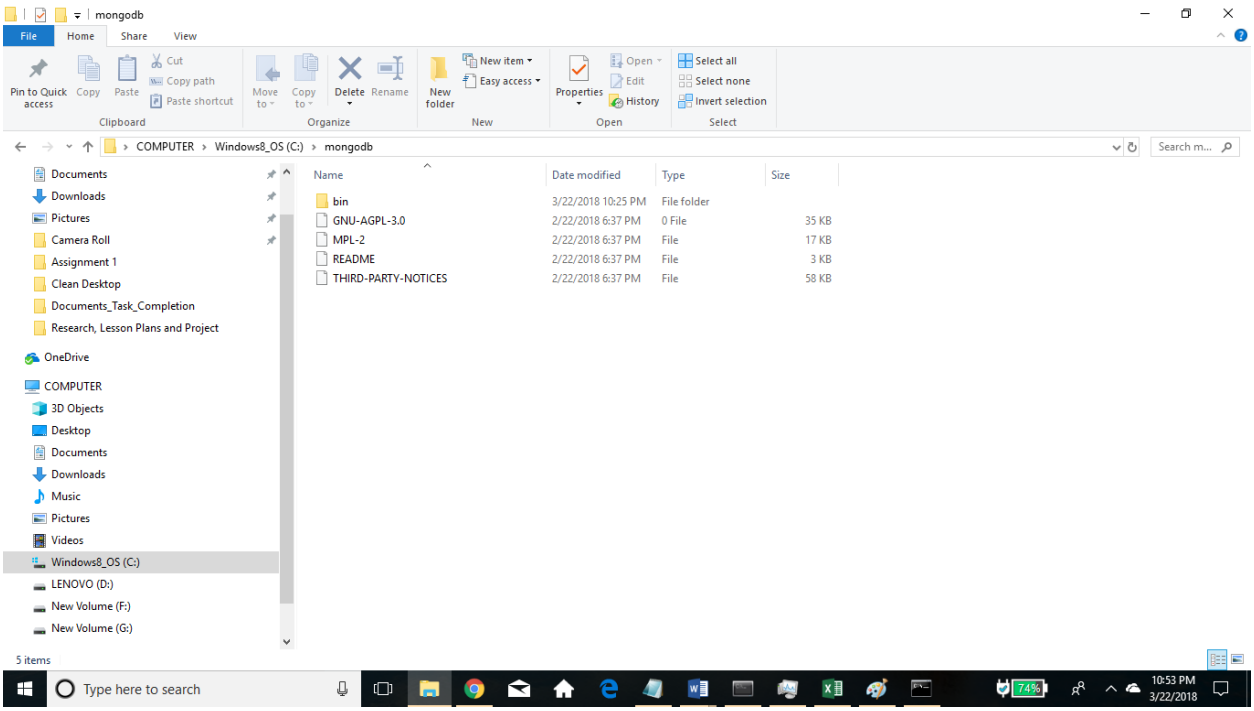




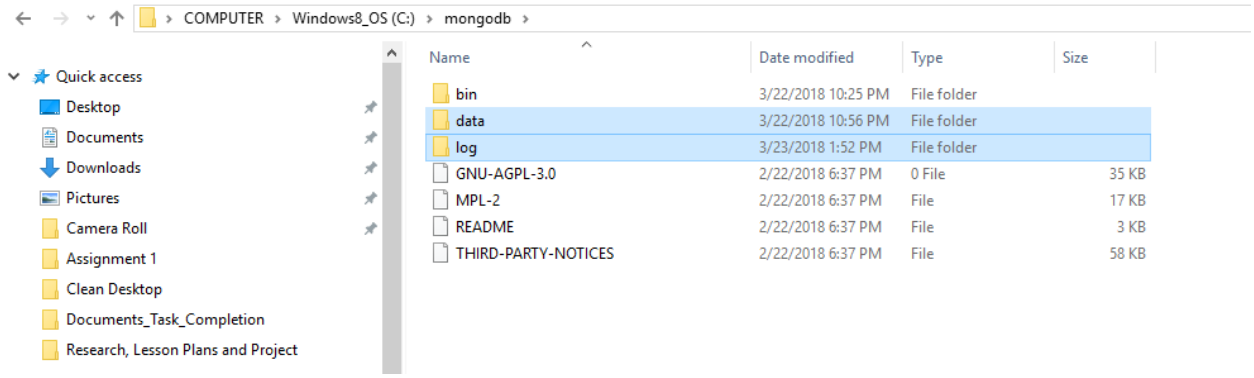
Click on Install.



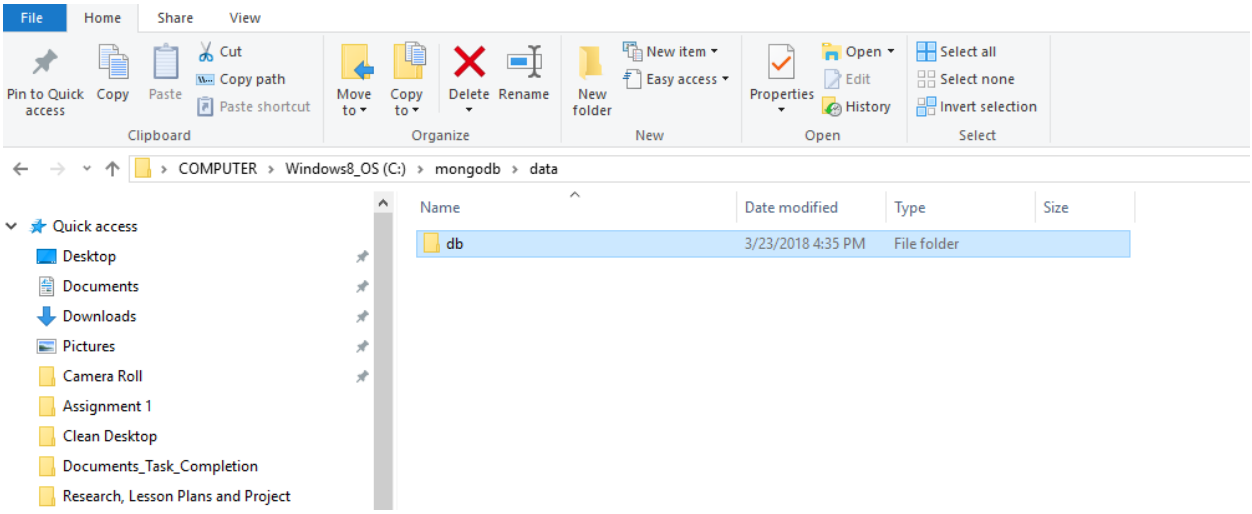
4. Once you are done with installation, go to C drive and open **mongodb** folder (or the location where you have saved your file).



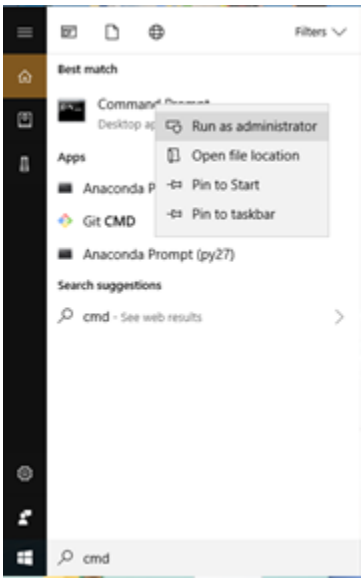
>> <u>Create 2 new folders named **data** and **log**.</u>



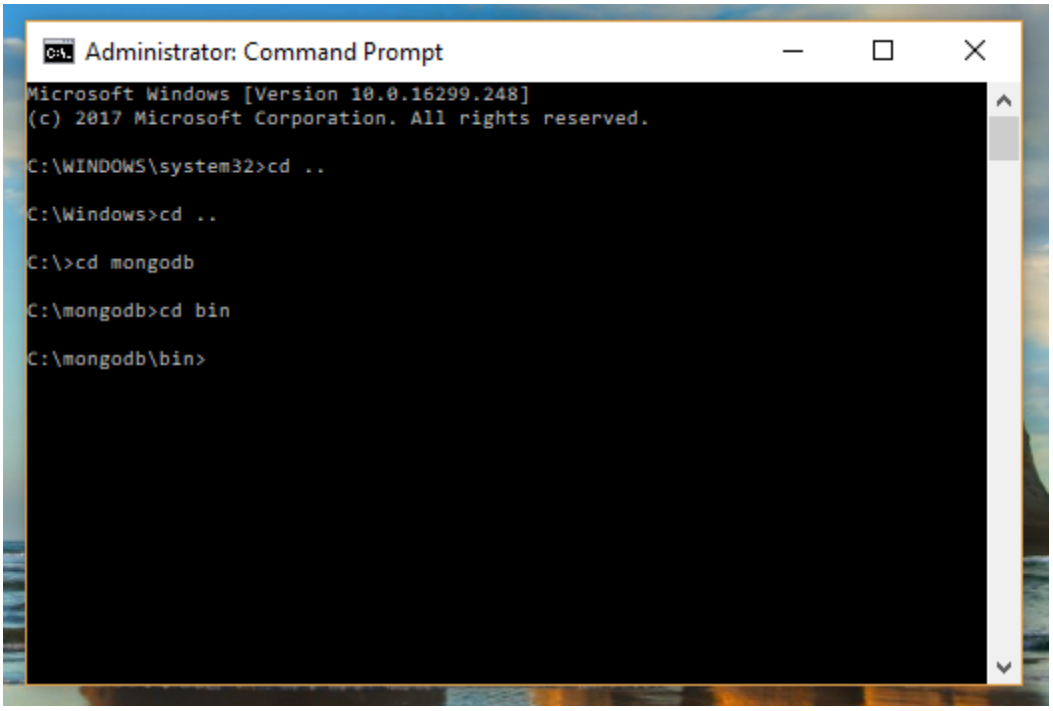
Inside data folder, create one more folder called **db** (that's where all the data will be stored.)



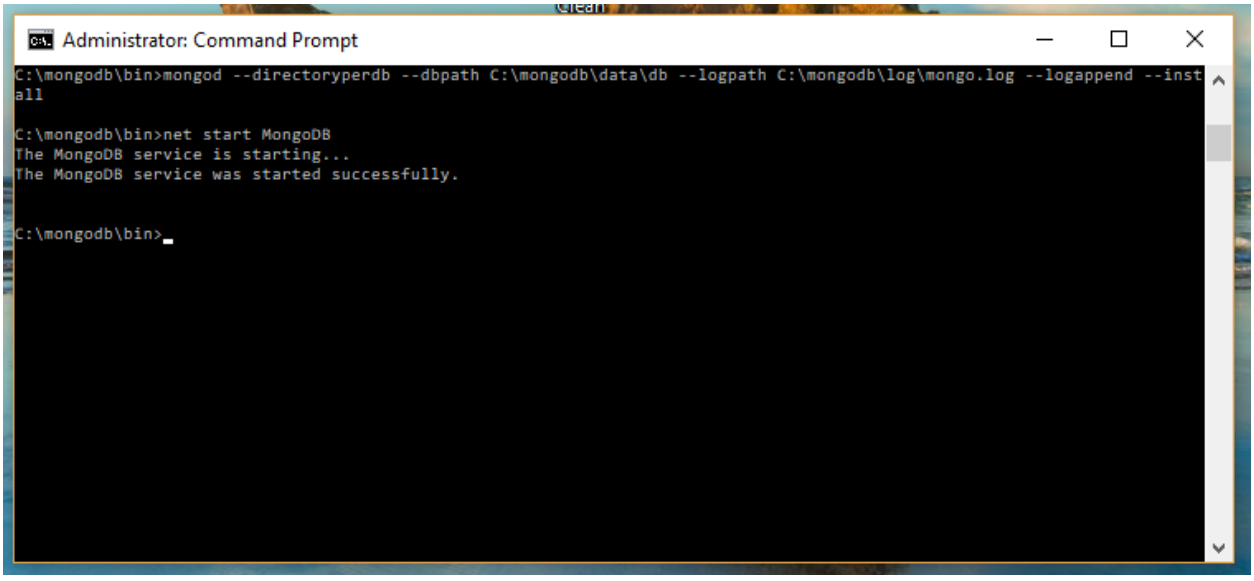
5. open command line. (Run as administrator)



>> <u>Will go to bin folder</u>

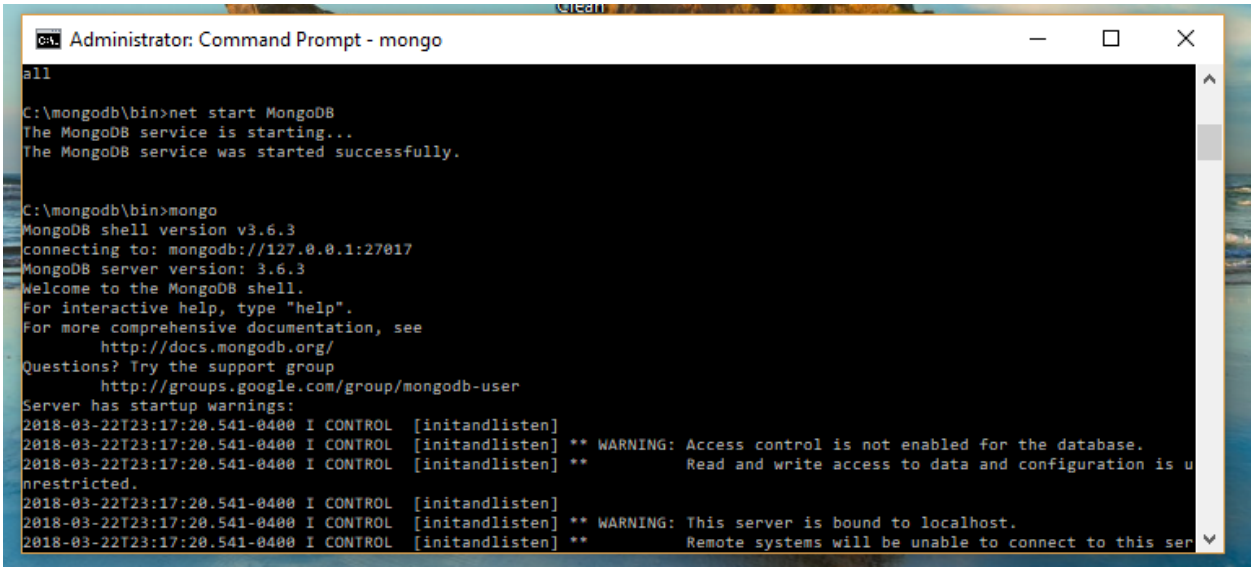


Now we will run **mongod** and set a bunch of flags like **directoryperdb** and then we will specify dbpath & logpath which is going to be the path to the folder **db** we created and where logs will be saved. The command "**mongod --directoryperdb --dbpath C:\mongodb\data\db --logpath C:\mongodb\log\mongo.log --logappend –install**" will allow us to run a service.



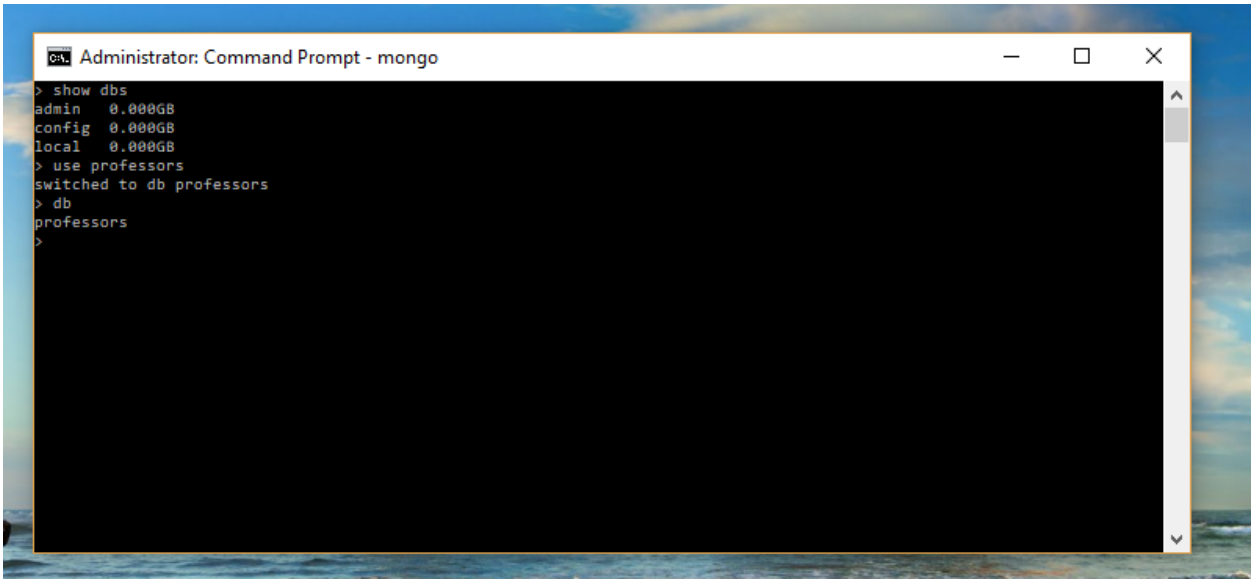
Enter "**net start MongoDB**" to start the service.

Now we will be working in the mongo shell. Just type mongo in the bin directory.



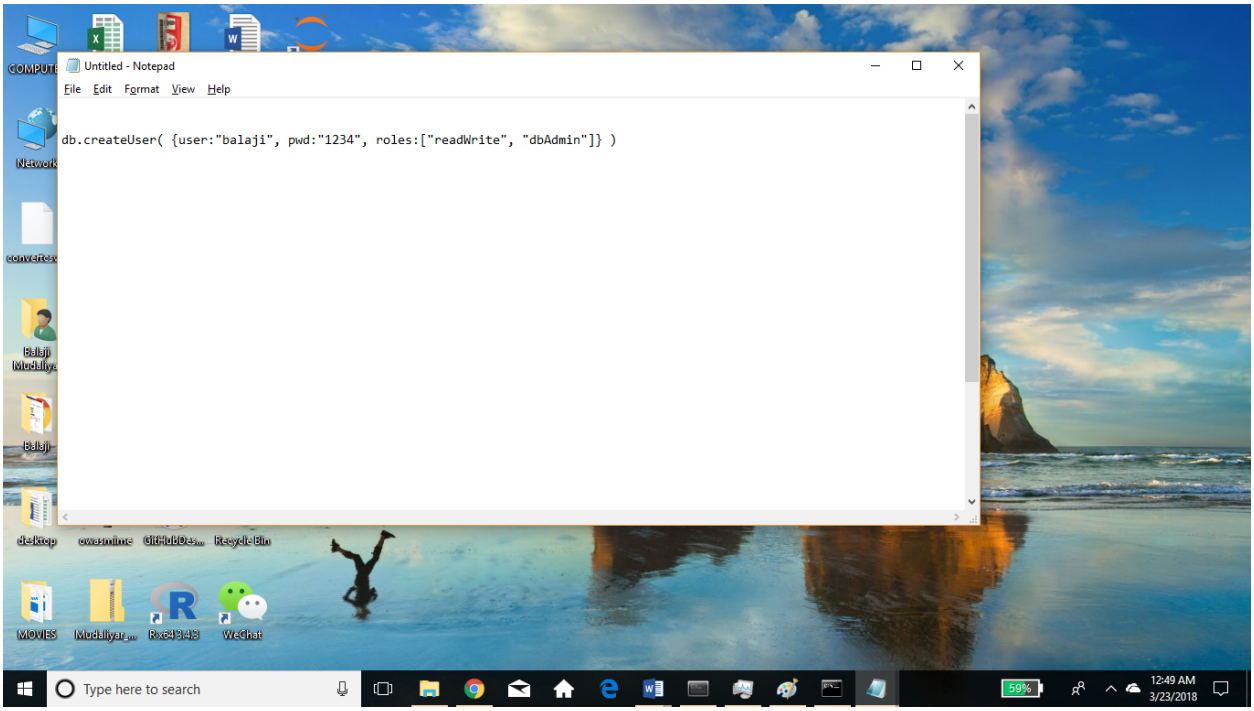
Type **cls** and press enter. That will clear everything out.

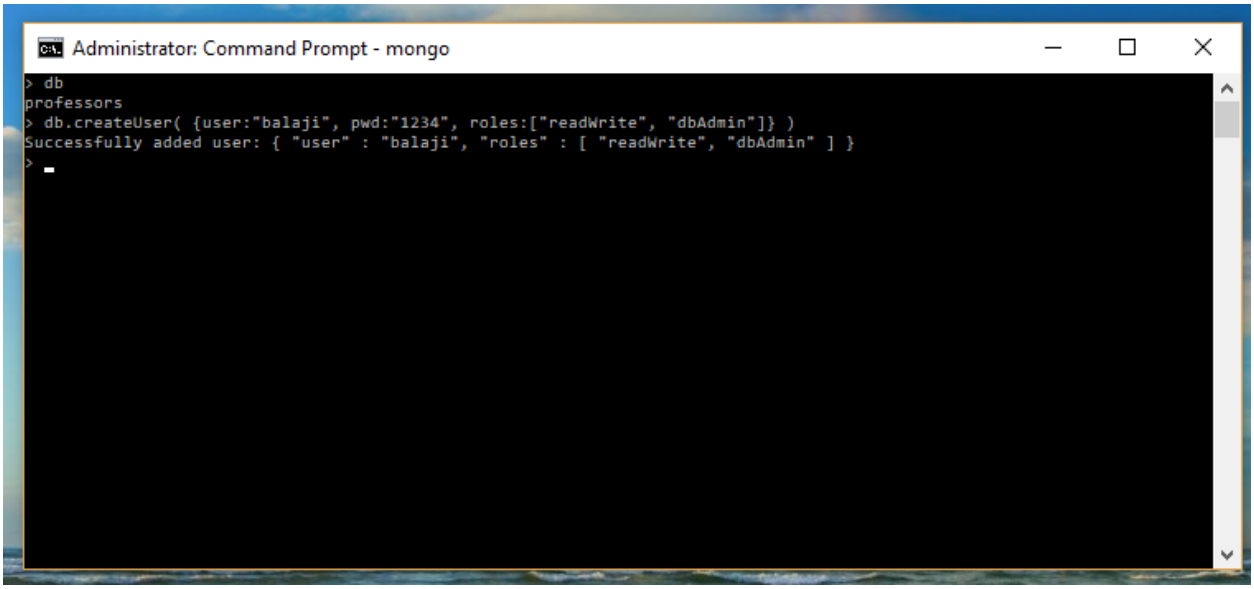
Create your **db** by giving any name. Here I have given '**professors'.**



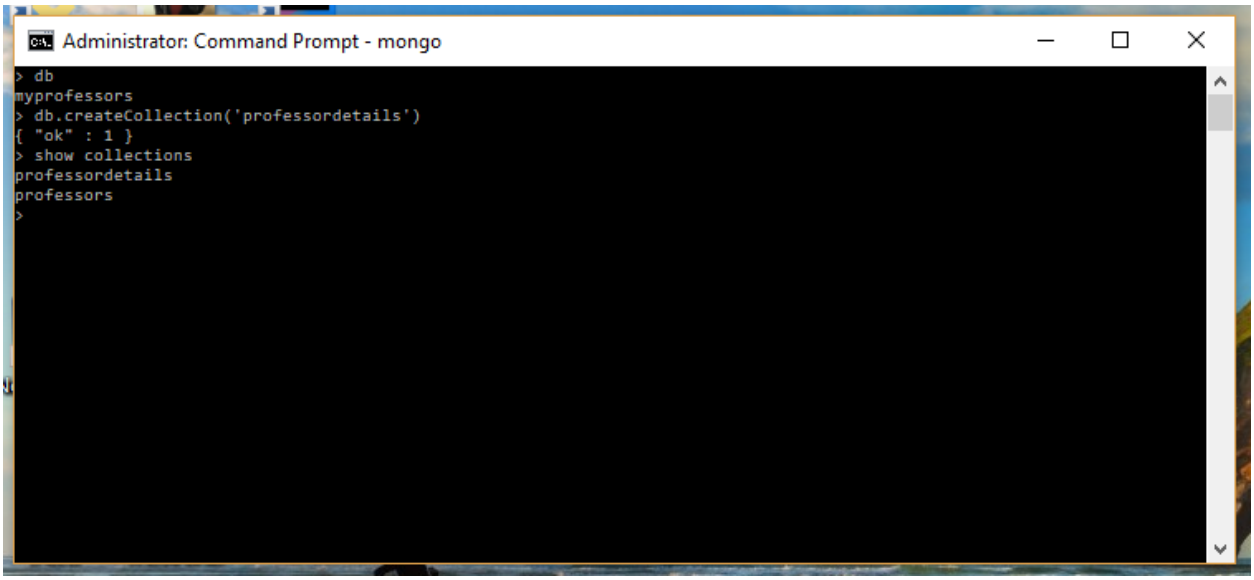
Open a notepad so that it will be easy to write commands on it and copy paste on you Command line.

Before we insert data in the database, we will create a user for the database.

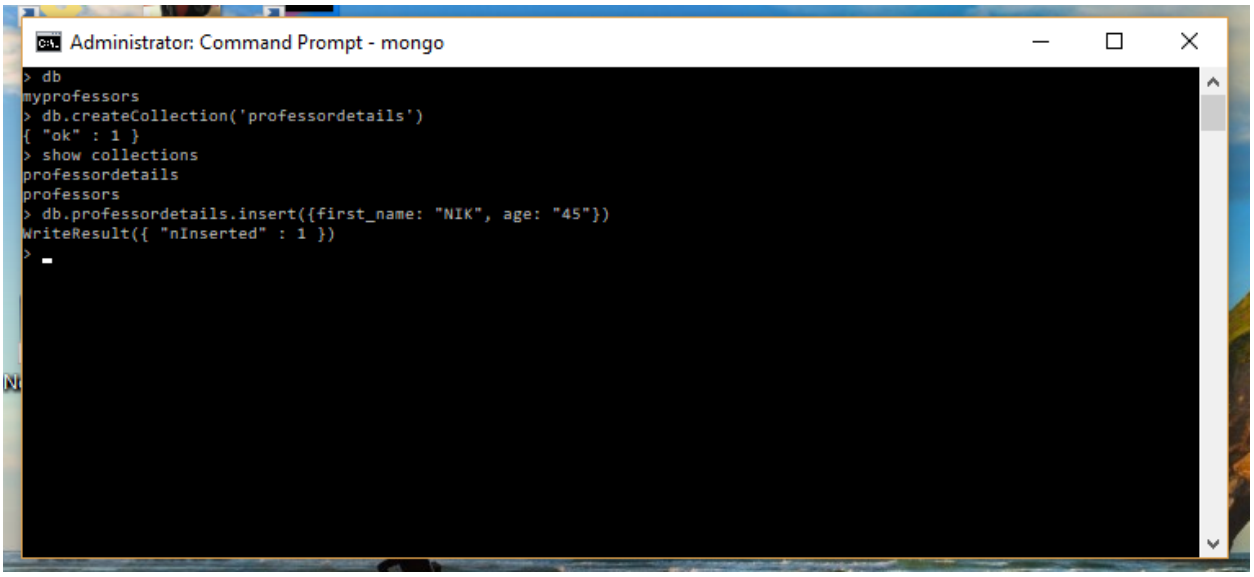




>> Creating collection for the database. I have created 2 collections: professor details and professors. But will be using only one.



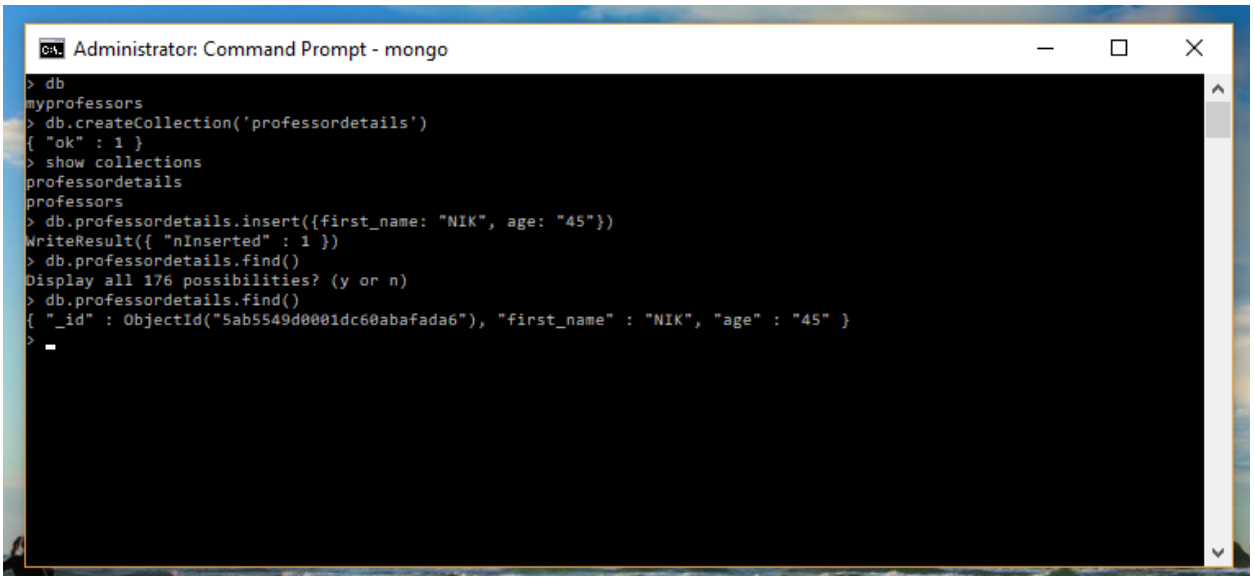
Now we will be inserting documents in to the collections.



>> **<u>Finding documents in the collection</u>**

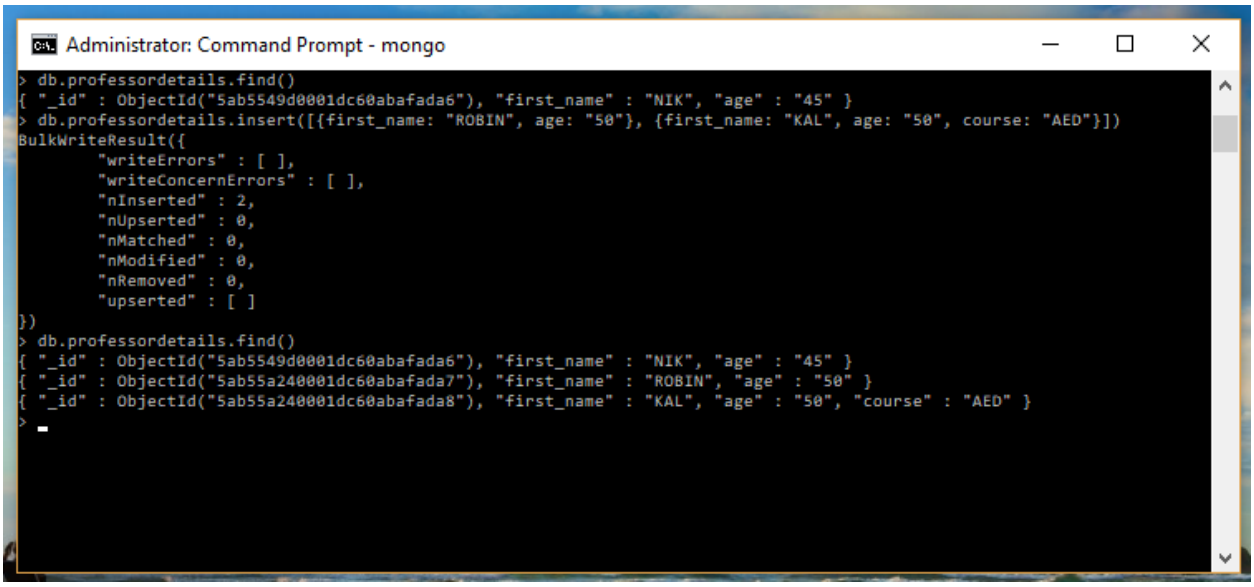
Command - Db.Collection.find()

This command will display the documents in a collection. We have only one document in the collection, so the below image is displaying only one. The document also has an "_id" field which is set to be an object id and is a unique value to find documents (and other stuffs). It is automatically created and can be considered as primary key.

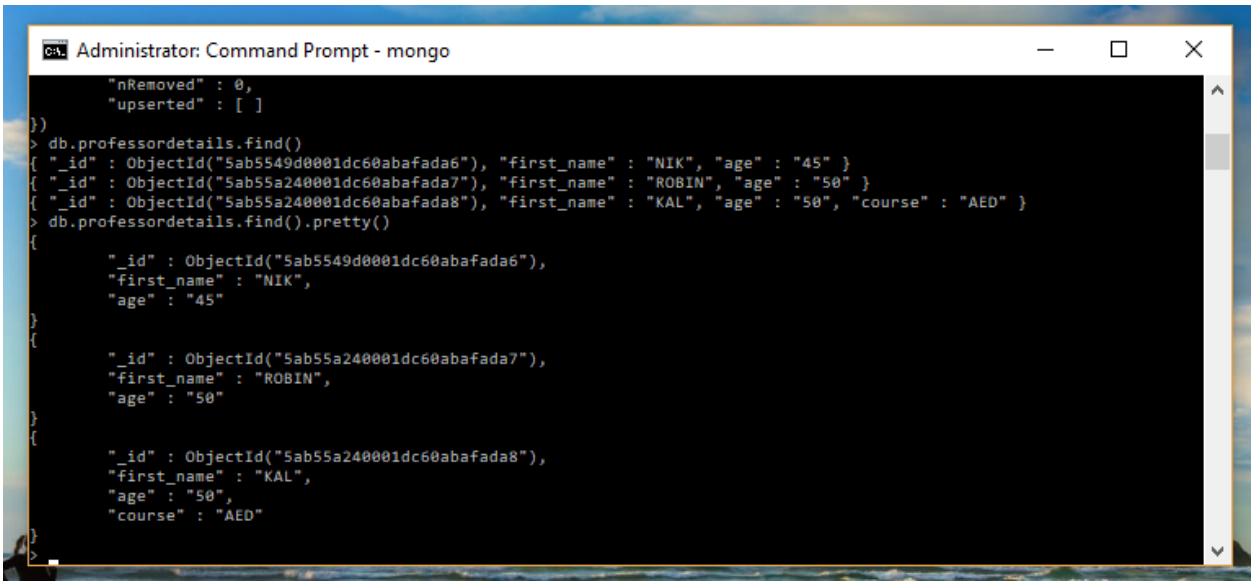


>> **Adding more documents to the collection at once by making an array (using square brackets).**

We can see the documents have been added to the collection and also there is an additional field course under KAL which we didn't specify for NIK and ROBIN. So we can have whatever fields we can have for our documents in a collection.



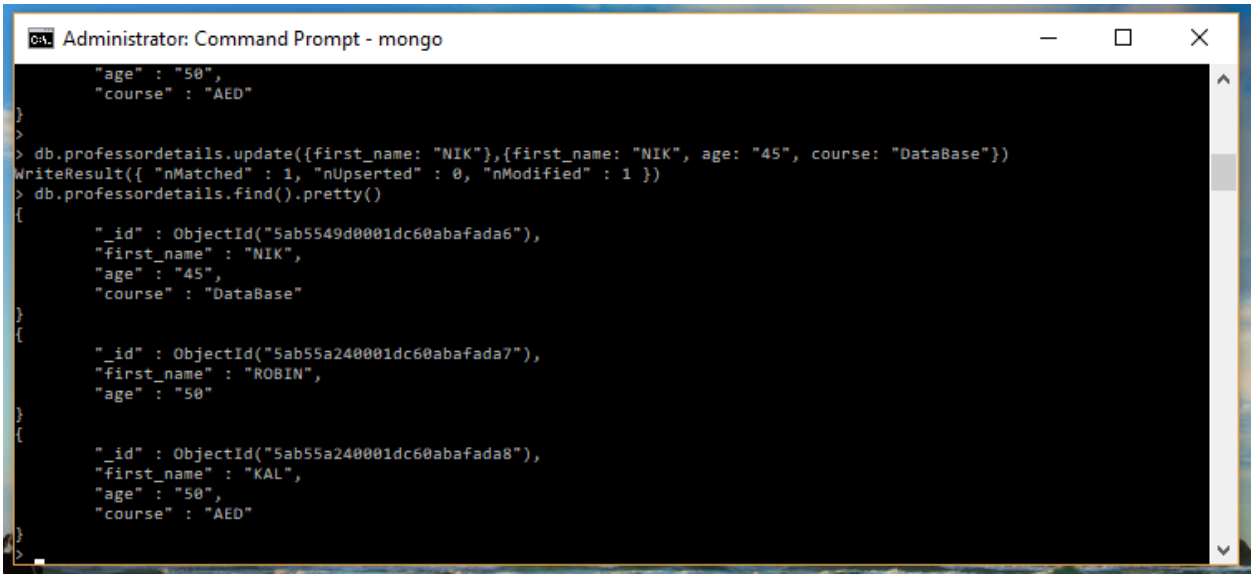
The above documents looks clean since there are only few but if there are ton of fields it can get messy, so we can use **.pretty()** which is a helper function to make it look neat.



>> <u>Updating/adding fields for the documents in a collection</u>.

Command: **db.professordetails.update({first_name: "NIK"},{first_name: "NIK", age: "45", course: "DataBase"})**

The first parameter is the matching parameter, so here it will match the first name with the documents in the collection. Generally, the matching parameter should be object ID or something which is unique for each document. If there are multiple documents with first name NIK, it will update all of them. For the sake of simplicity, we will be using first name here. The 2nd parameter is the one to be changed/updated.

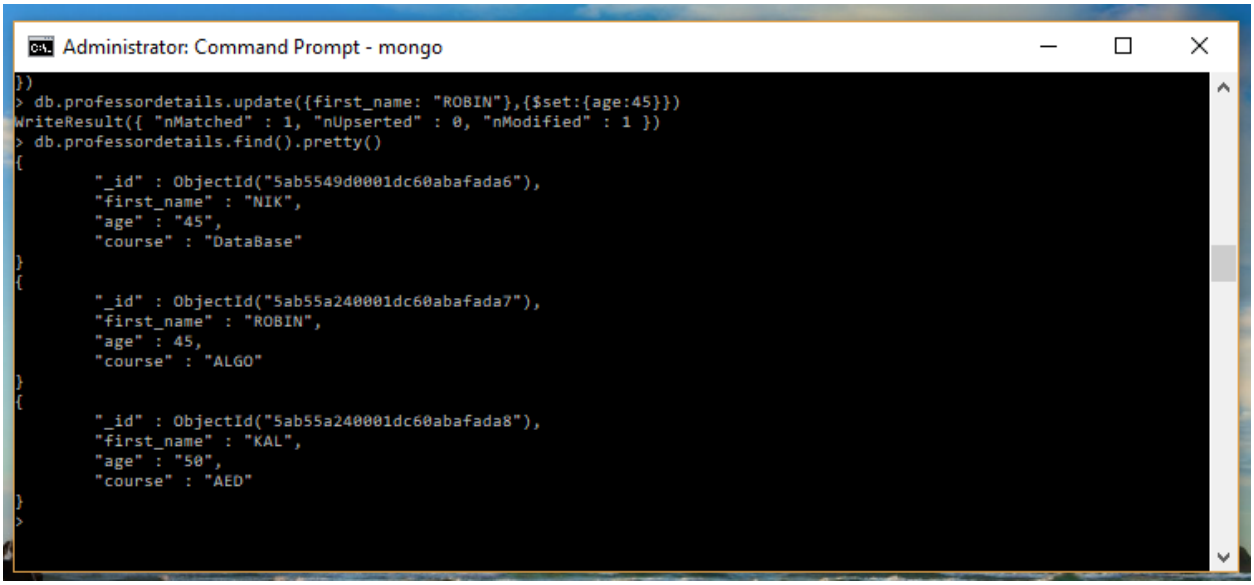


We can clearly see that the course '**DataBase'** has been added for **NIK**.

>> SET Operator

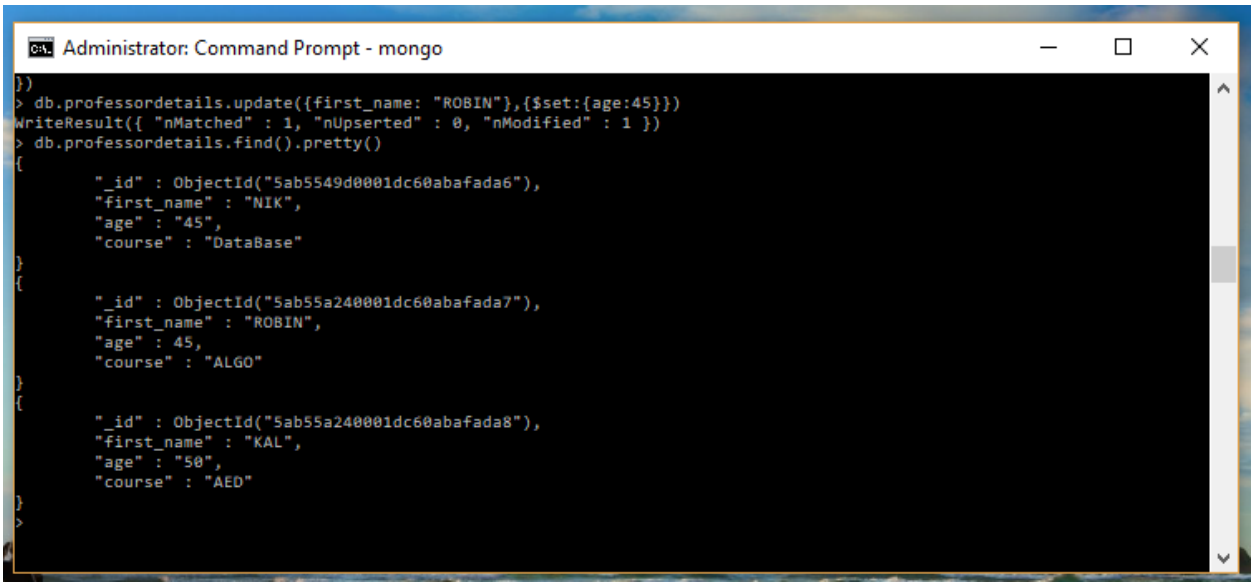
Updating the document with set operator.

Command**: db.professordetails.update({first_name: "ROBIN"},{$set:{course: "ALGO"}})**



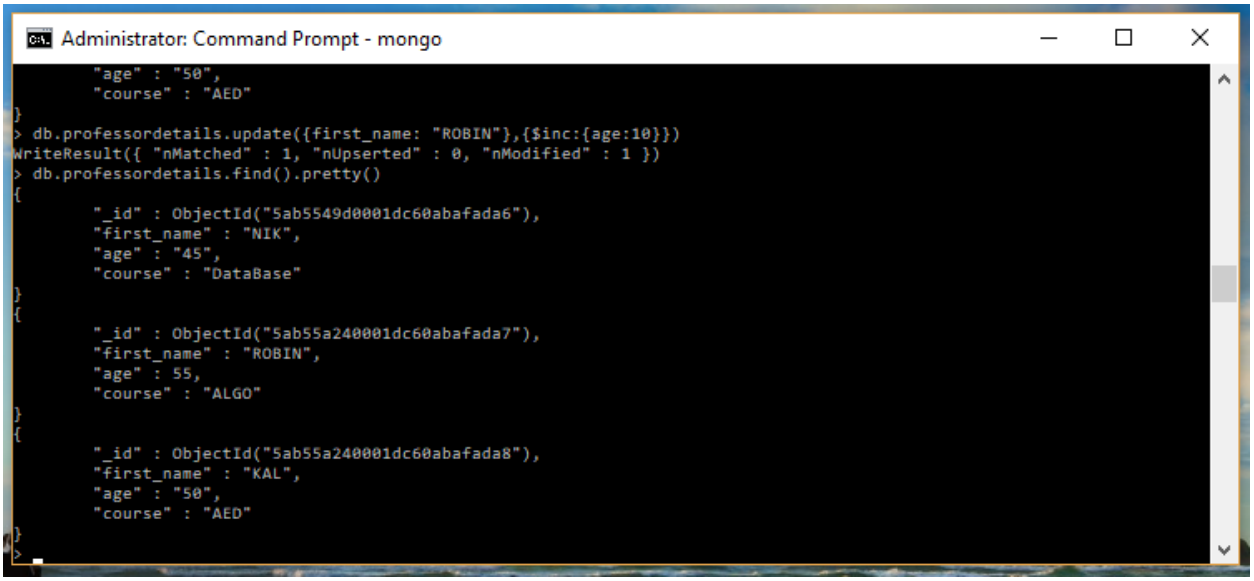
Using set operator, we can change/add a particular field without disturbing the other fields in the document. Course ALGO has been added under ROBIN without any changes in the other fields.

Changing the age of professor **ROBIN** to numeric type:



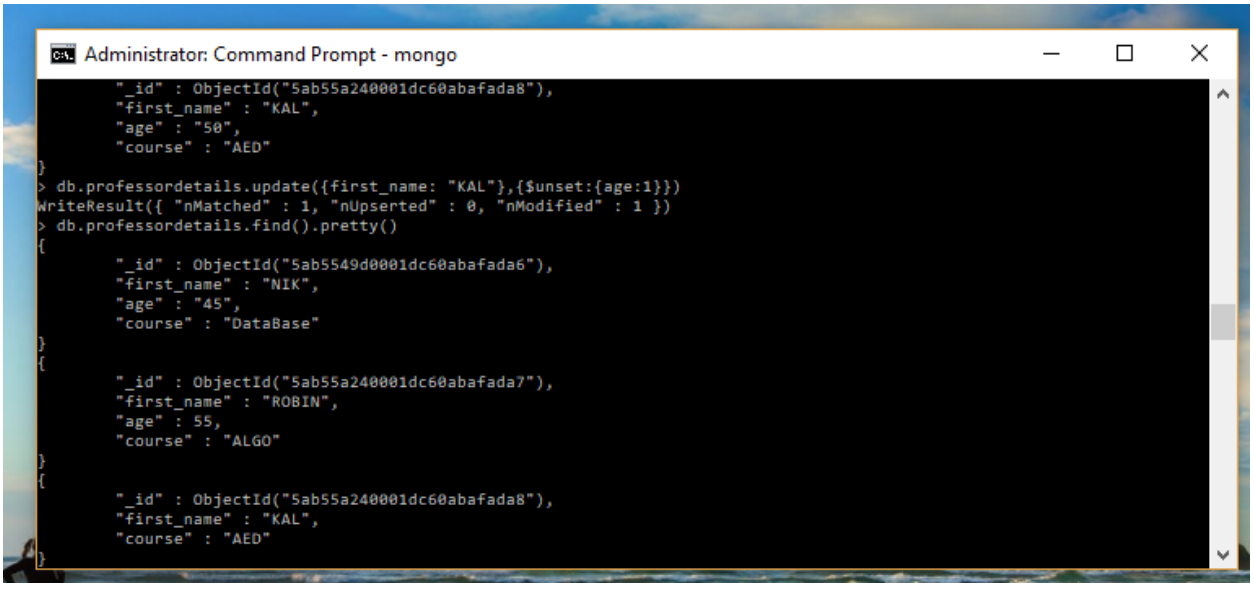
>> INC Operator

We will use to **inc** operator to increase the age of professor by 10 years.



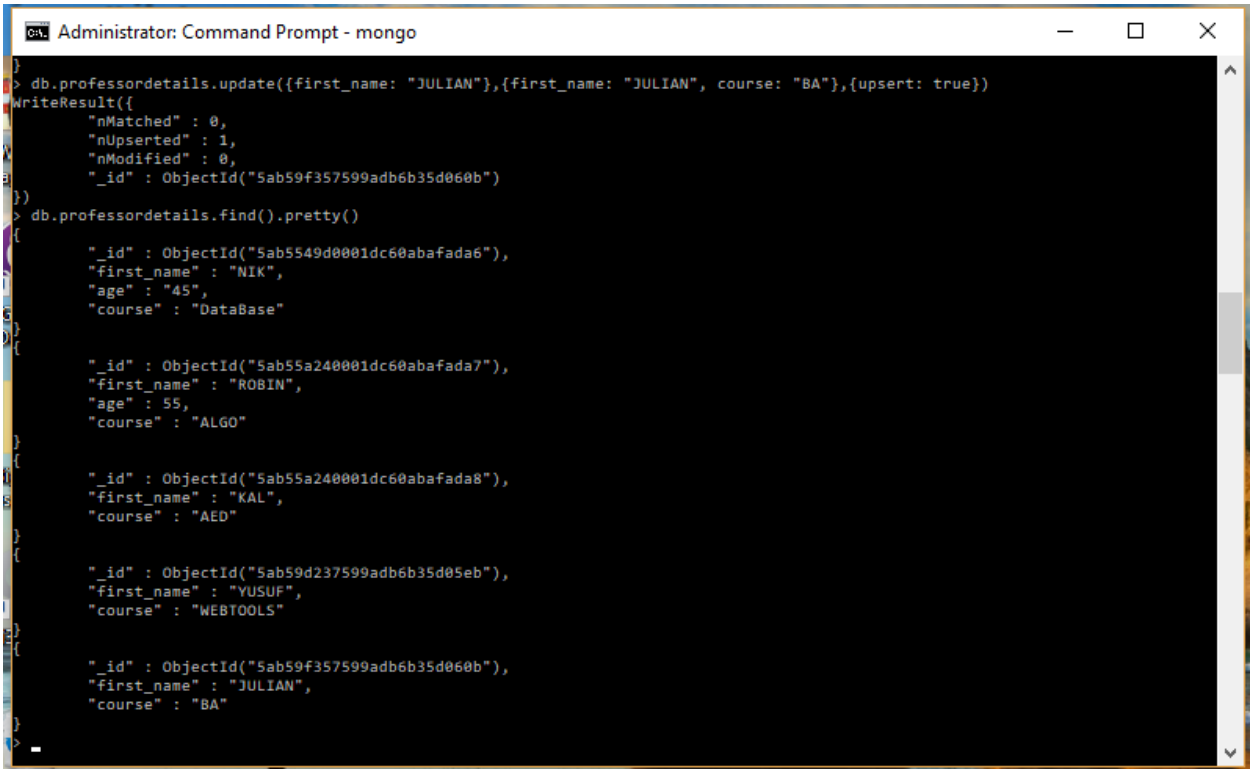
>> Unset Operator

We will use unset operator to remove a field of the document in a collection.



Age field for ROBIN has been removed.

>> Updating a document which is not present

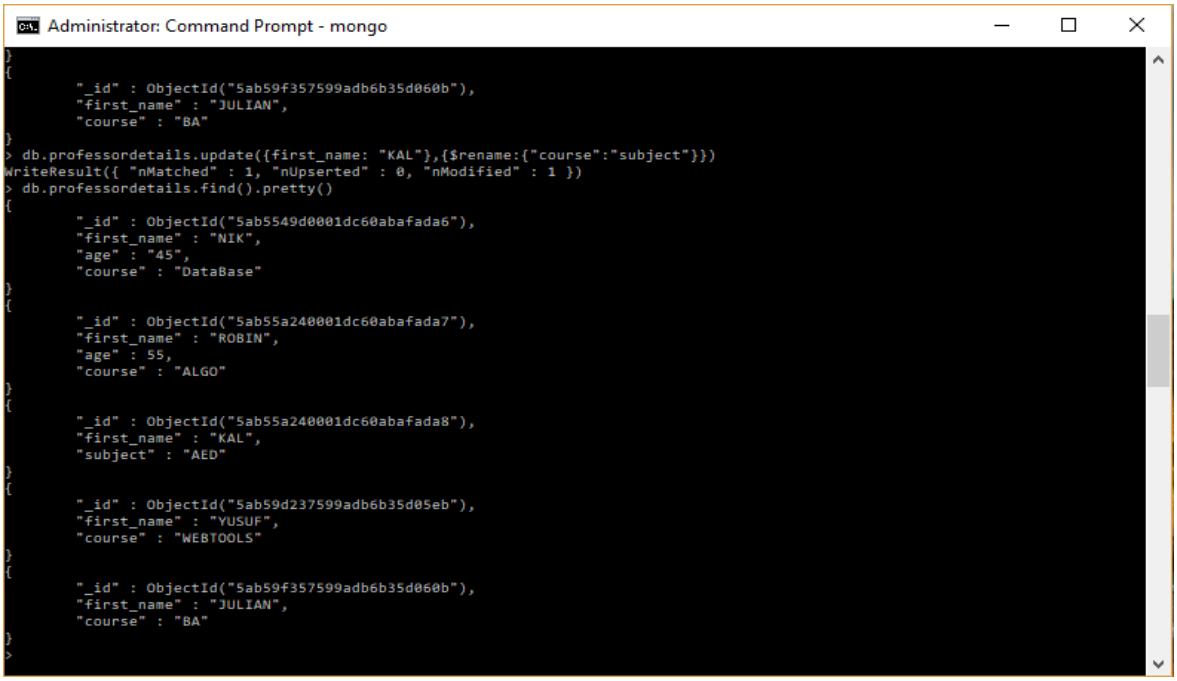


JULIAN and **YUSUF** were updated in the collection though they were not present. Command for adding **JULIAN** is shown above. Here we have added third element as **upsert** and set it to **true**.

Also, an object ID has been created for YUSUF and JULIAN.

>> **rename field in document:**

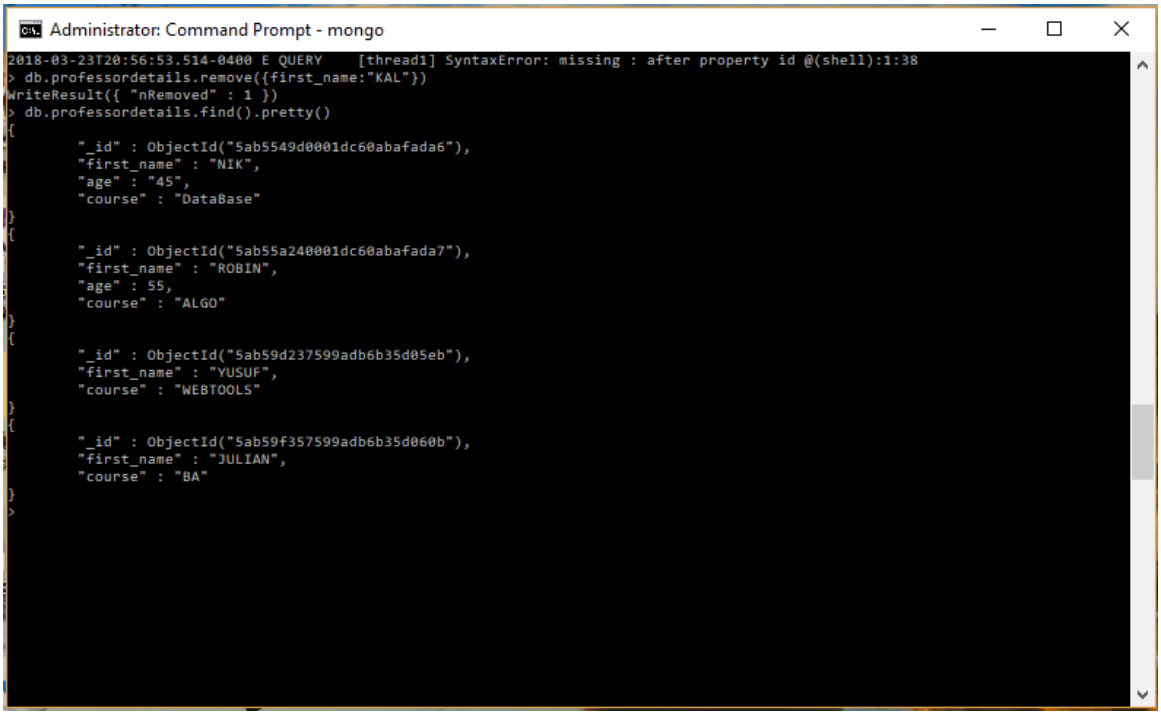
The field course has been renamed to subject for KAL.



>> Removing document from collection

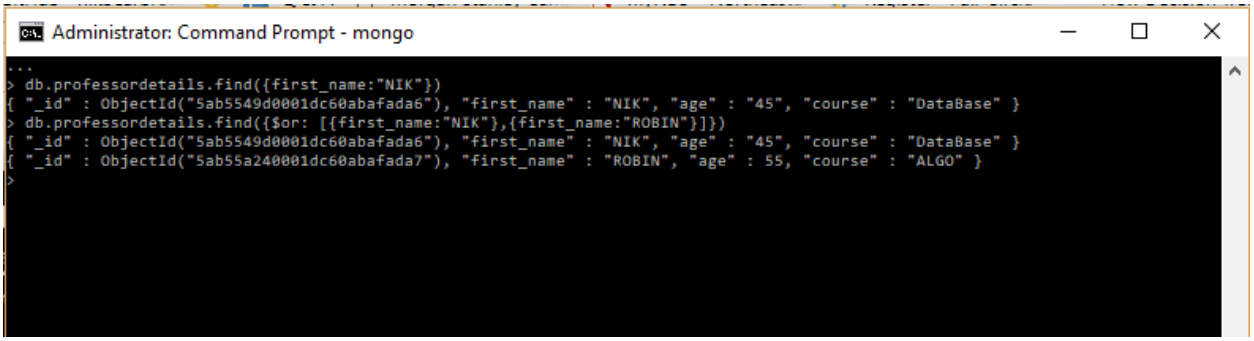
Command: **db.professordetails.remove({first_name:"KAL"})**

KAL has been removed.



>> Finding a document:

Command: **db.professordetails.find({$or: [{first_name:"NIK"},{first_name:"ROBIN"}]})**



Using OR we can find multiple documents in a collection.

Command: **db.professordetails.find({age:{$lt:60}})**

Finds age of professor less than 60. The document for NIK won't be displayed since age for NIK is non-numeric i.e. "45". If it was 45 and not "45", the document for NIK would have been displayed.

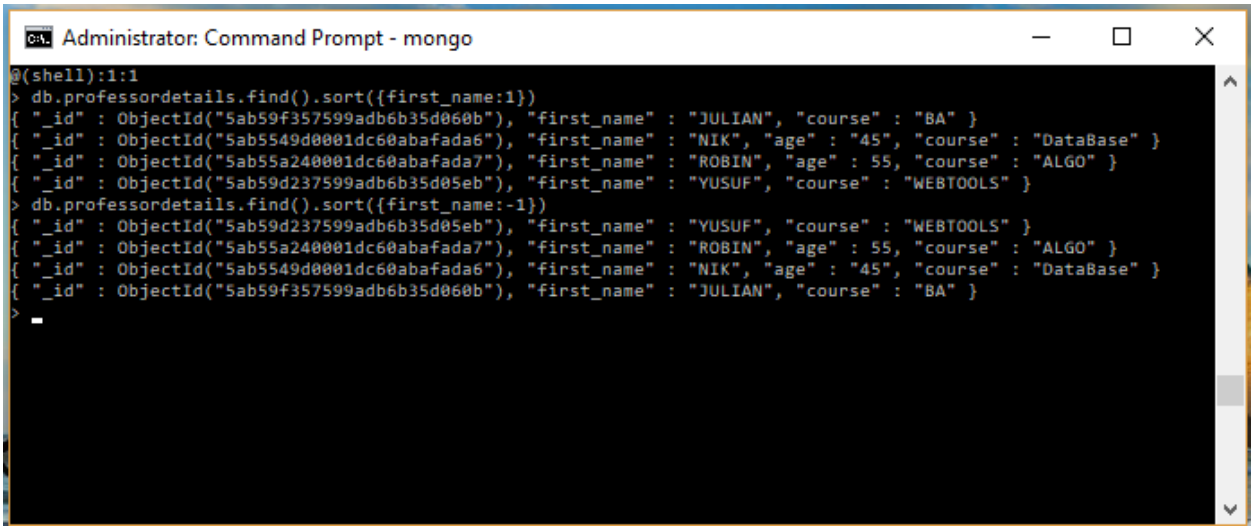
>> Sorting documents in an order:

Command**: db.professordetails.find().sort({first_name:1})**

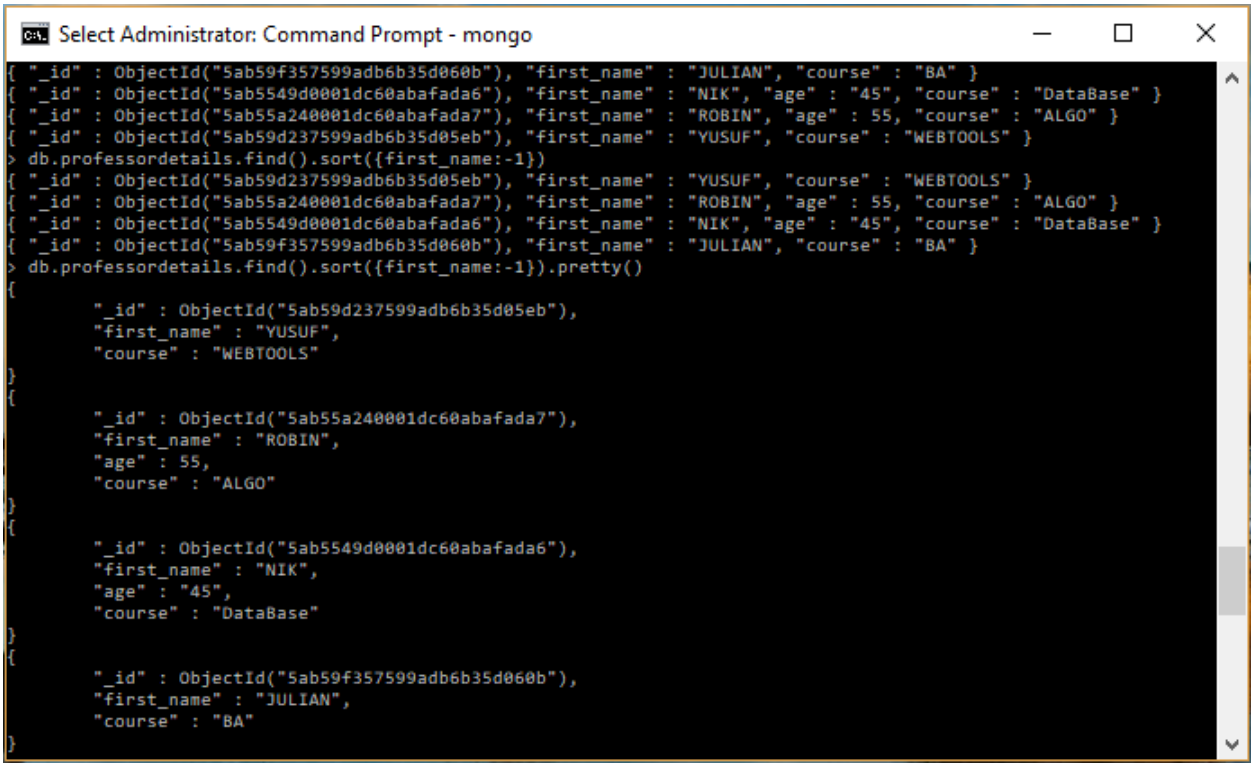
The above command will sort the documents in ascending order(since the value is 1) by their first name.

For descending order we can set it to -1.

Command**: db.professordetails.find().sort({first_name:-1})**



To make it look neat, we can use **pretty**()

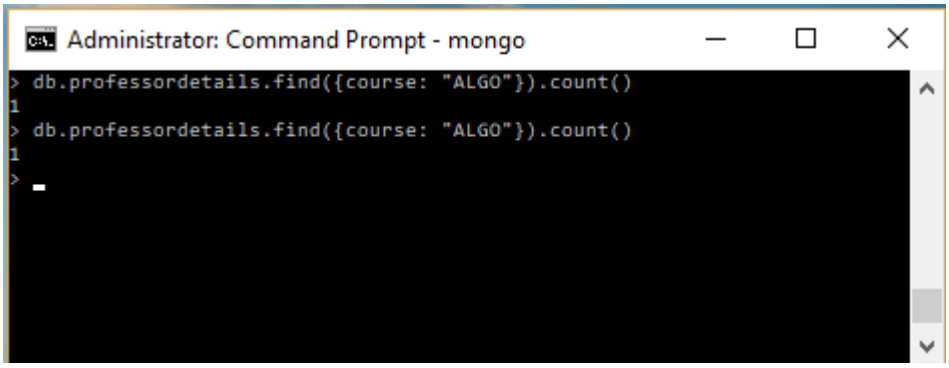


>>Counting document in a collection:

Command: **db.professordetails.find().count()**

The above command will count the number of documents in a collection.

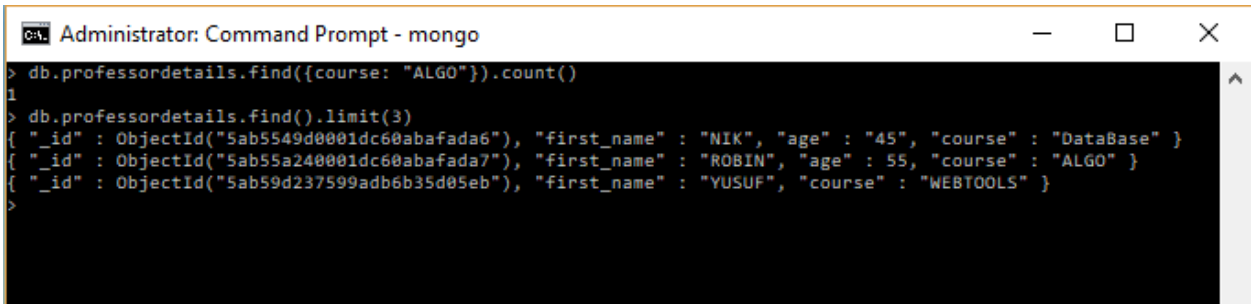
Command: **db.professordetails.find({course: “ALGO”}).count()**



>> finding all document but with a limit:

Command: **db.professordetails.find().limit(3)**

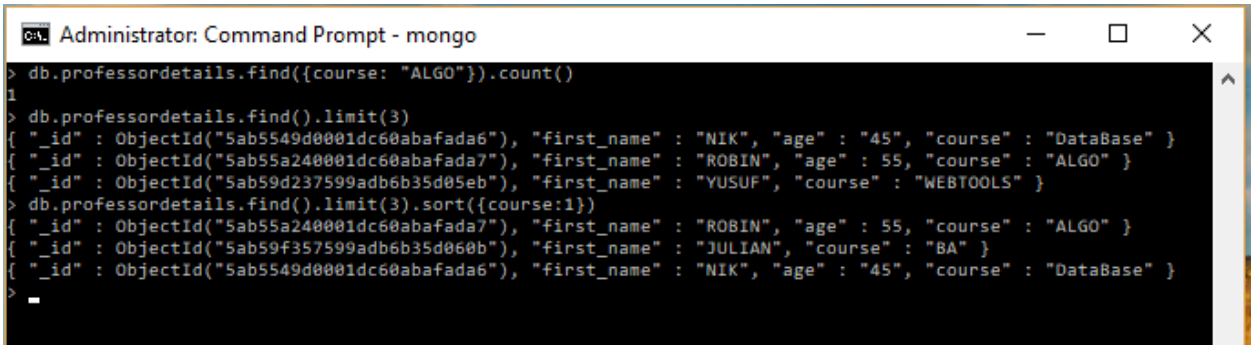
Will display only 3 documents out of total 4, since the limit has been set to 3.



>> Limit & Sorting:

Command: **db.professordetails.find().limit(3).sort({course:1})**

Will sort the documents by course and display the first 3.



>> **Iterating through Foreach:**

Command:

db.professordetails.find().forEach(function(doc){print("ProfessorName:" +doc.first_name)})