**HARSHA V N**

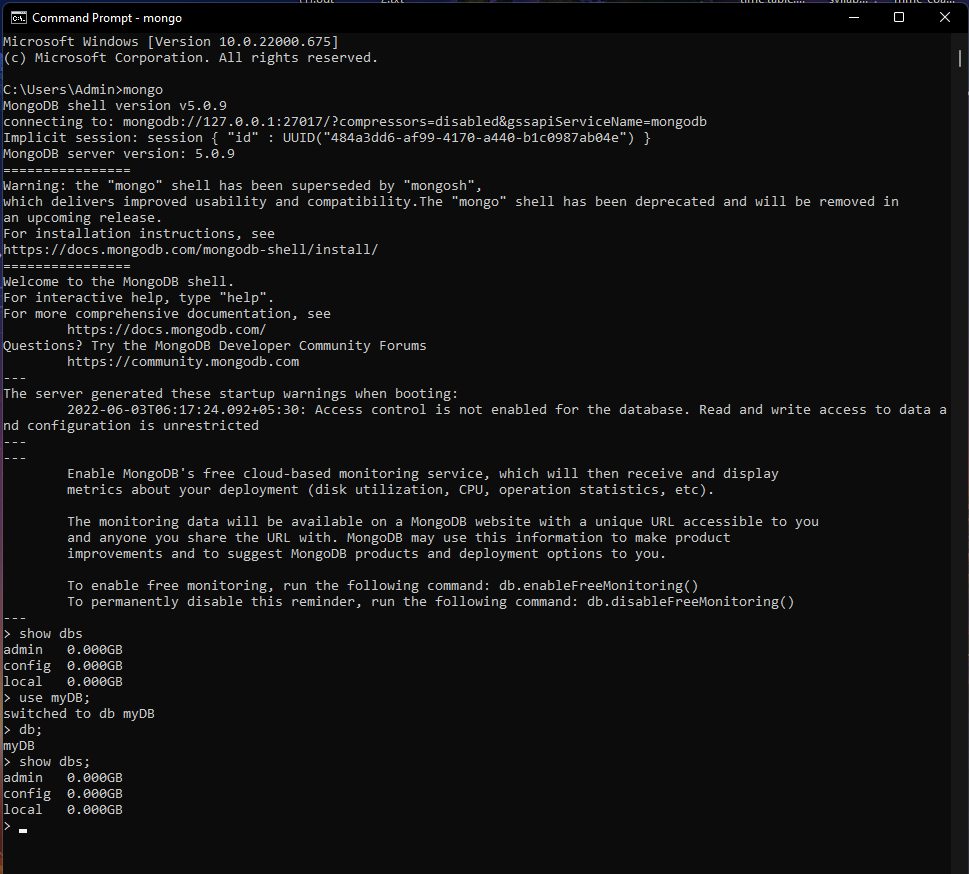
**1BM20CS406**

**Lab Program 1**

I. CREATE DATABASE IN MONGODB.

use myDB;

db

show dbs; 

II.CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS

1. To create a collection by the name “Student”. Let us take a look at the collection list

prior to the creation of the new collection “Student”.

db.createCollection(“Student”); =&gt; sql equivalent CREATE TABLE STUDENT(…);

2. To drop a collection by the name “Student”.

db.Student.drop();

3. Create a collection by the name “Students” and store the following data in it.

db.Student.insert({\_id:1,StudName:&quot;MichelleJacintha&quot;,Grade:&quot;VII&quot;,Hobbies:&quot;InternetS

urfing&quot;});

4. Insert the document for “AryanDavid” in to the Students collection only if it does not

already exist in the collection. However, if it is already present in the collection, then

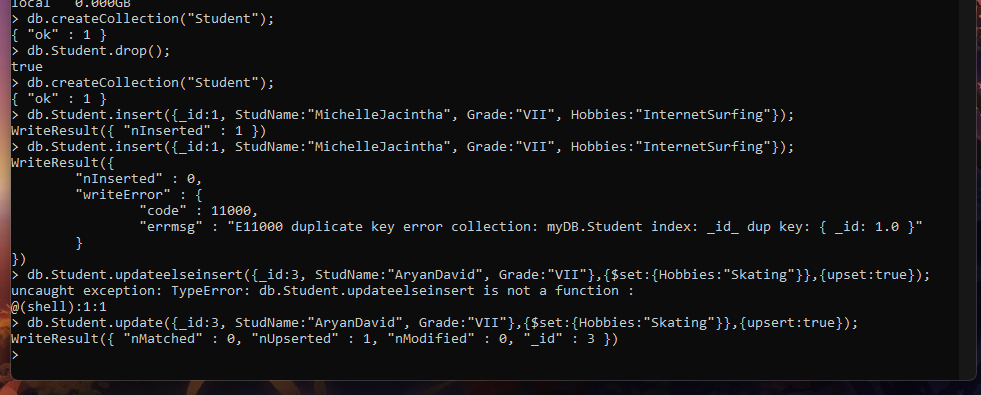
update the document with new values. (Update his Hobbies from “Skating” to “Chess”.

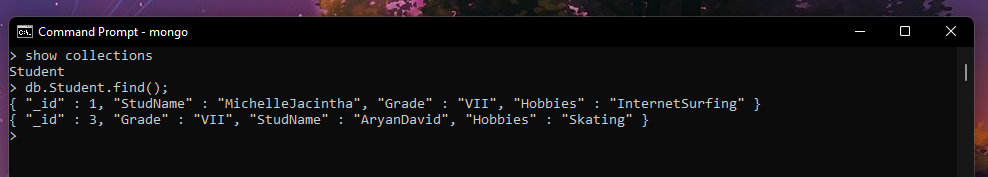
) Use “Update else insert” (if there is an existing document, it will attempt to update it,

if there is no existing document then it will insert it).

db.Student.update({\_id:3,StudName:&quot;AryanDavid&quot;,Grade:&quot;VII&quot;},{$set:{Hobbies:&quot;Skatin

g&quot;}},{upsert:true});





5. FIND METHOD

A. To search for documents from the “Students” collection based on certain search

criteria.

db.Student.find({StudName:&quot;Aryan David&quot;});

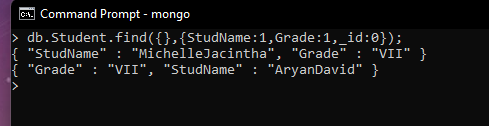
({cond..},{columns.. column:1, columnname:0} )



B. To display only the StudName and Grade from all the documents of the Students

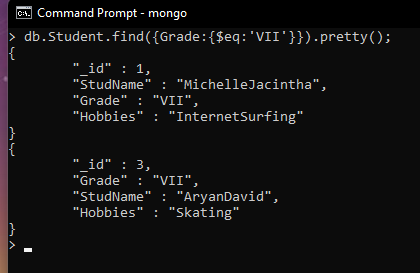
collection. The identifier\_id should be suppressed and NOT displayed.

db.Student.find({},{StudName:1,Grade:1,\_id:0});



C. To find those documents where the Grade is set to ‘VII’

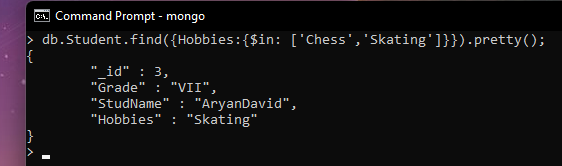
db.Student.find({Grade:{$eq:&#39;VII&#39;}}).pretty();



D. To find those documents from the Students collection where the Hobbies is set to

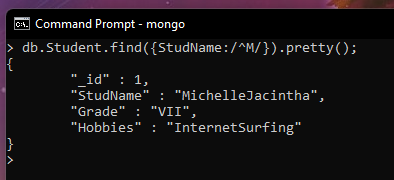
either ‘Chess’ or is set to ‘Skating’.

db.Student.find({Hobbies :{ $in: [&#39;Chess&#39;,&#39;Skating&#39;]}}).pretty ();



E. To find documents from the Students collection where the StudName begins with “M”.

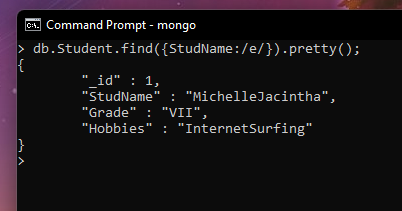
db.Student.find({StudName:/^M/}).pretty();



F. To find documents from the Students collection where the StudNamehas an “e” in any

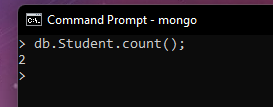
position.

db.Student.find({StudName:/e/}).pretty();



G. To find the number of documents in the Students collection.

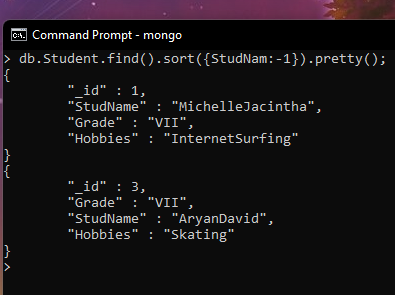
db.Student.count();



H. To sort the documents from the Students collection in the descending order of

StudName.

db.Student.find().sort({StudName:-1}).pretty();



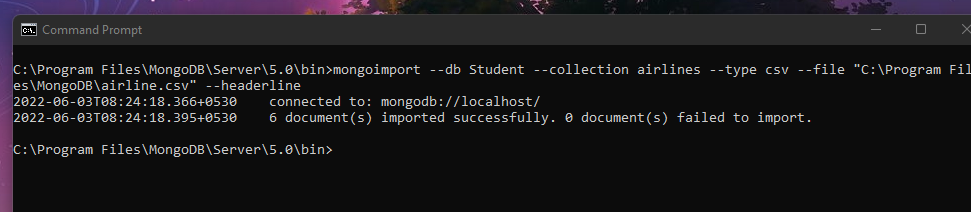
III. Import data from a CSV file

Given a CSV file “sample.txt” in the D:drive, import the file into the MongoDB

collection, “SampleJSON”. The collection is in the database “test”.

mongoimport --db Student --collection airlines --type csv –headerline --file

/home/hduser/Desktop/airline.csv



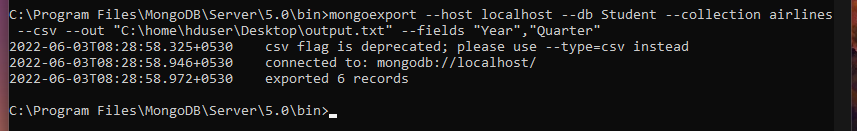
IV. Export data to a CSV file

This command used at the command prompt exports MongoDB JSON documents from

“Customers” collection in the “test” database into a CSV file “Output.txt” in the D:drive.

mongoexport --host localhost --db Student --collection airlines --csv --out

/home/hduser/Desktop/output.txt –fields “Year”,”Quarter”

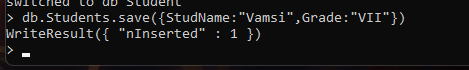


V. Save Method :

Save() method will insert a new document, if the document with the \_id does not

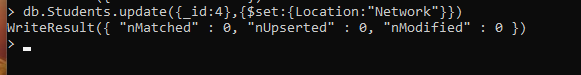
exist. If it exists it will replace the exisiting document.

db.Students.save({StudName:”Vamsi”, Grade:”VI”})



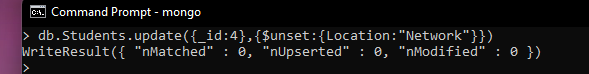
VI. Add a new field to existing Document:

db.Students.update({\_id:4},{$set:{Location:”Network”}})



VII. Remove the field in an existing Document

db.Students.update({\_id:4},{$unset:{Location:”Network”}})



VIII. Finding Document based on search criteria suppressing few fields

db.Student.find({\_id:1},{StudName:1,Grade:1,\_id:0});

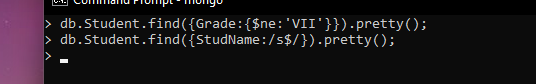
To find those documents where the Grade is not set to ‘VII’

db.Student.find({Grade:{$ne:&#39;VII&#39;}}).pretty();

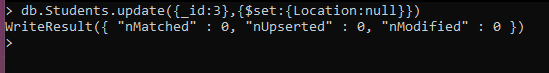
To find documents from the Students collection where the StudName ends with s.

db.Student.find({StudName:/s$/}).pretty();





IX. to set a particular field value to NULL



XCount the number of documents in Student Collections



XI. Count the number of documents in Student Collections with grade :VII

db.Students.count({Grade:”VII”})

retrieve first 3 documents

db.Students.find({Grade:”VII”}).limit(3).pretty();

Sort the document in Ascending order

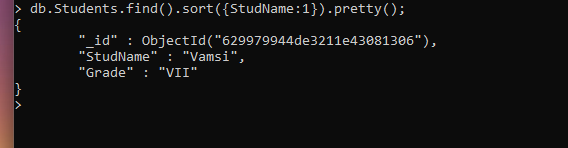
db.Students.find().sort({StudName:1}).pretty();

Note:

for desending order : db.Students.find().sort({StudName:-1}).pretty();

to Skip the 1 st two documents from the Students Collections

db.Students.find().skip(2).pretty()

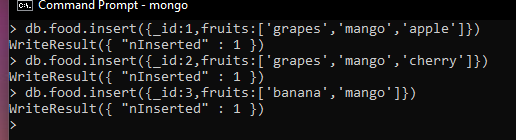


XII. Create a collection by name “food” and add to each document add a “fruits” array

db.food.insert( { \_id:1, fruits:[&#39;grapes&#39;,&#39;mango&#39;,&#39;apple&#39;] } )

db.food.insert( { \_id:2, fruits:[&#39;grapes&#39;,&#39;mango&#39;,&#39;cherry&#39;] } )

db.food.insert( { \_id:3, fruits:[&#39;banana&#39;,&#39;mango&#39;] } )



To find those documents from the “food” collection which has the “fruits array”

constitute of “grapes”, “mango” and “apple”.

db.food.find ( {fruits: [&#39;grapes&#39;,&#39;mango&#39;,&#39;apple&#39;] } ). pretty().



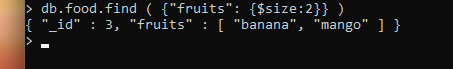
To find in “fruits” array having “mango” in the first index position.

db.food.find ( {&#39;fruits.1&#39;:&#39;grapes&#39;} )



To find those documents from the “food” collection where the size of the array is two.

db.food.find ( {“fruits”: {$size:2}} )



To find the document with a particular id and display the first two elements from the

array “fruits”

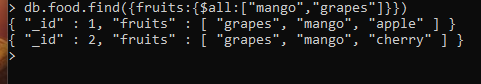
db.food.find({\_id:1},{“fruits”:{$slice:2}})



To find all the documets from the food collection which have elements mango and

grapes in the array “fruits”

db.food.find({fruits:{$all:[“mango”,”grapes”]}})



update on Array:

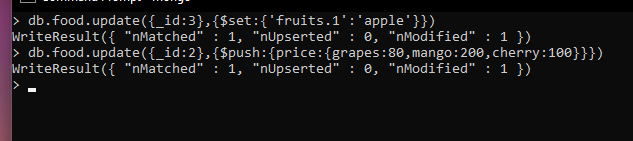
using particular id replace the element present in the 1 st index position of the fruits

array with apple

db.food.update({\_id:3},{$set:{&#39;fruits.1&#39;:&#39;apple&#39;}})

insert new key value pairs in the fruits array

db.food.update({\_id:2},{$push:{price:{grapes:80,mango:200,cherry:100}}})



Note: perform query operations using - pop, addToSet, pullAll and pull

XII. Aggregate Function :

Create a collection Customers with fields custID, AcctBal, AcctType.

Now group on “custID” and compute the sum of “AccBal”.

db.Customers.aggregate ( {$group : { \_id : “$custID”,TotAccBal : {$sum:”$AccBal”} } } );

match on AcctType:”S” then group on “CustID” and compute the sum of “AccBal”.

db.Customers.aggregate ( {$match:{AcctType:”S”}},{$group : { \_id : “$custID”,TotAccBal :

{$sum:”$AccBal”} } } );

match on AcctType:”S” then group on “CustID” and compute the sum of “AccBal” and

total balance greater than 1200.

db.Customers.aggregate ( {$match:{AcctType:”S”}},{$group : { \_id : “$custID”,TotAccBal :

{$sum:”$AccBal”} } }, {$match:{TotAccBal:{$gt:1200}}});

