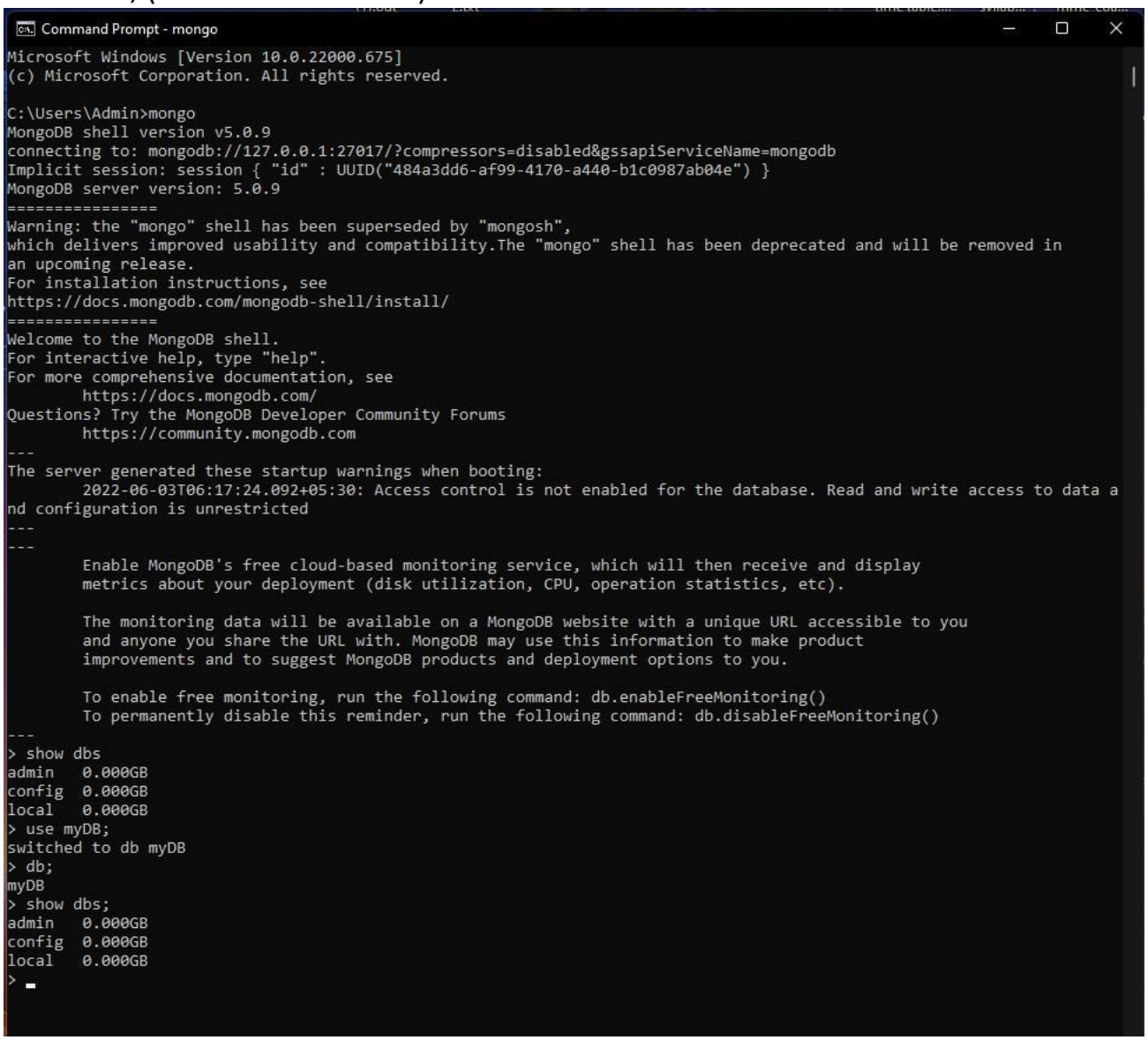
LAB-1
Mongo db CRUD demonstration:

I. CREATE DATABASE IN MONGODB.
use myDB; db; (Confirm the existence of your database)
show dbs; (To list all databases)



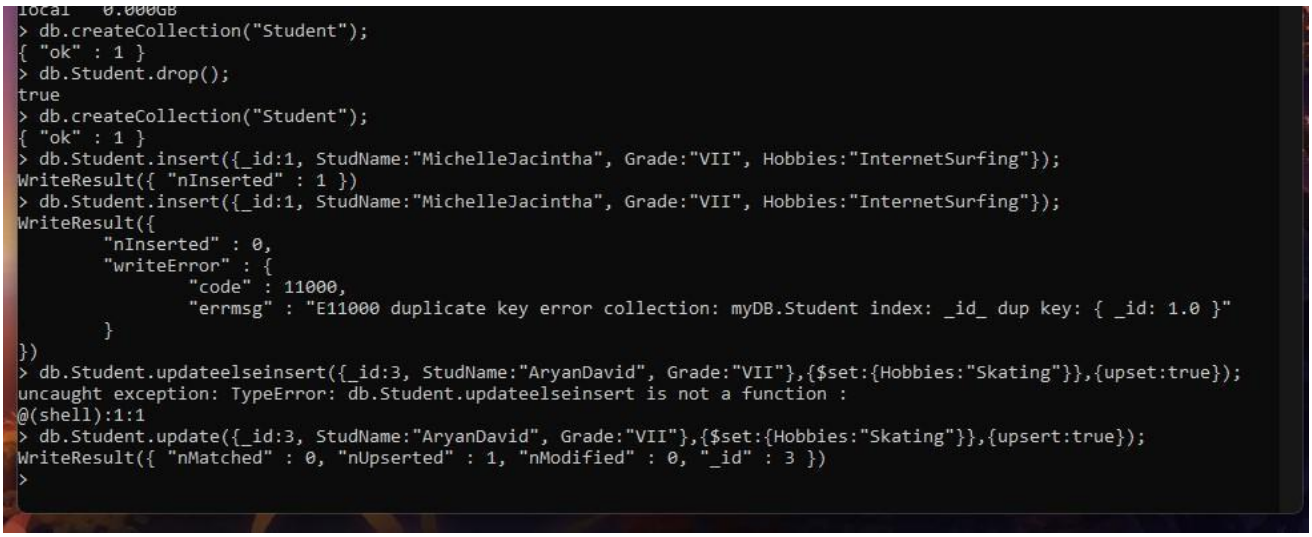
II.CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS

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”); => 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:"MichelleJacintha",Grade:"VII",Hobbies:"Int ernetS urfing"});

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:"AryanDavid",Grade:"VII"},{$set:{Hobbies:&quo t;Skatin g"}},{upsert:true});



5. FIND METHOD

A.

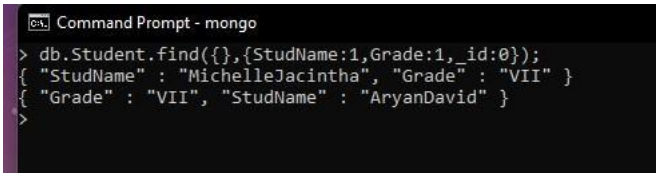
To search for documents from the “Students” collection based on certain search criteria.
db.Student.find({StudName:"Aryan David"});
({cond..},{columns.. column:1, columnname:0})

```
> db.Student.find({StudName:"AryanDavid"});
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
>
```

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.

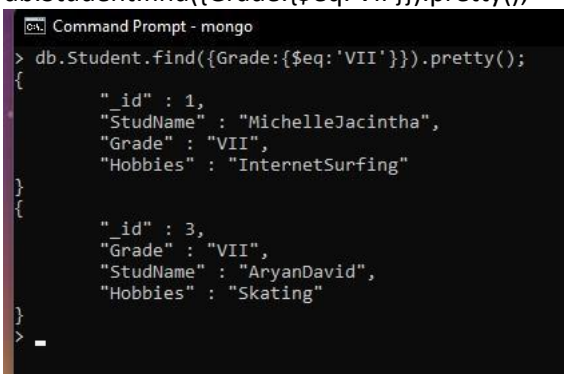
db.Student.find({},{StudName:1,Grade:1,_id:0});



C.

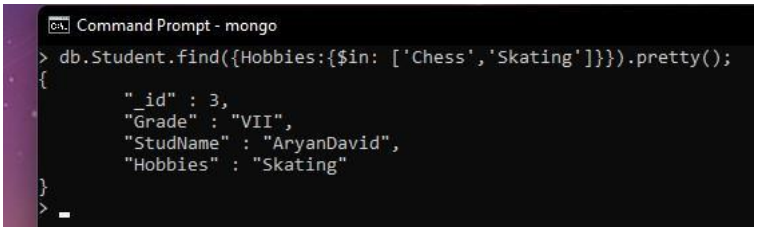
To find those documents where the Grade is set to 'VII'

db.Student.find({Grade:{$eq:'VII'}}).pretty();



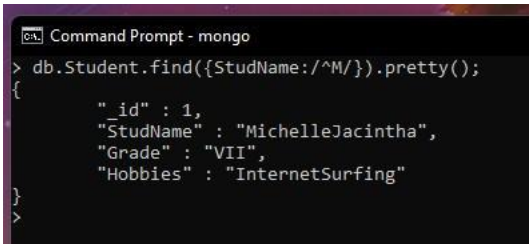
D.

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: ['Chess','Skating']}}).pretty ();

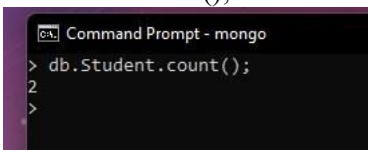


E.

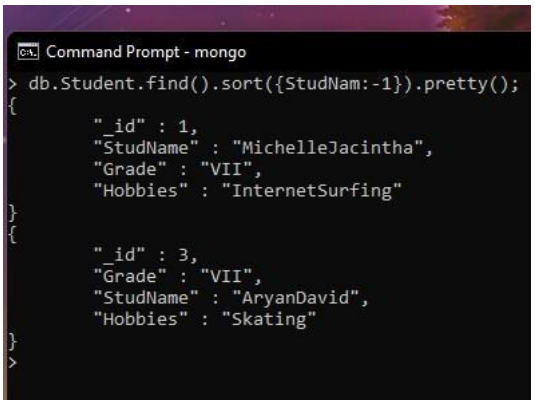
To find documents from the Students collection where the StudName begins with "M". db.Student.find({StudName:/^M/}).pretty();



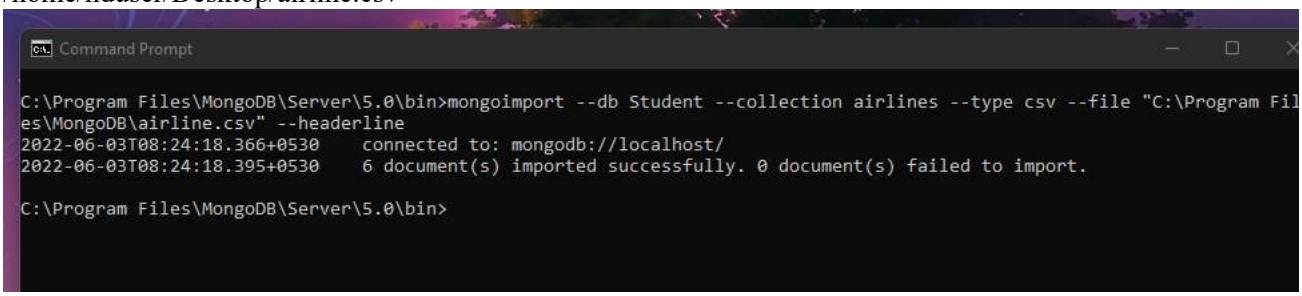
G.
To find the number of documents in the Students collection.
db.Student.count();



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”

```
C:\Program Files\MongoDB\Server\5.0\bin>mongoexport --host localhost --db Student --collection airlines
 --csv --out "C:\home\hduser\Desktop\output.txt" --fields "Year","Quarter"
2022-06-03T08:28:58.325+0530    csv flag is deprecated; please use --type=csv instead
2022-06-03T08:28:58.946+0530    connected to: mongodb://localhost/
2022-06-03T08:28:58.972+0530    exported 6 records

C:\Program Files\MongoDB\Server\5.0\bin>
```

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”})

```
> db.Students.save({StudName:"Vamsi",Grade:"VII"})
WriteResult({ "nInserted" : 1 })
>
```

VI. Add a new field to existing Document:
db.Students.update({_id:4},{$set:{Location:”Network”}})

```
> db.Students.update({_id:4},{$set:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

VII. Remove the field in an existing Document
db.Students.update({_id:4},{$unset:{Location:”Network”}})

```
Command Prompt - mongo
> db.Students.update({_id:4},{$unset:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

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:{$ne:'VII'}}).pretty();
To find documents from the Students collection where the StudName ends with s.
db.Student.find({StudName:/s$/}).pretty();

```
> db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});
>
> db.Student.find({Grade:{$ne:'VII'}}).pretty();
> db.Student.find({StudName:/s$/}).pretty();
>
```

IX. to set a particular field value to NULL

```
> db.Students.update({_id:3},{$set:{Location:null}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

XII. Create a collection by name “food” and add to each document add a “fruits” array db.food.insert({ _id:1,
fruits:['grapes','mango','apple'] })
db.food.insert({ _id:2,

fruits:['grapes','mango','cherry'] })
db.food.insert({ _id:3, fruits:['banana','mango'] })

```
Command Prompt - mongo
> db.food.insert({_id:1,fruits:['grapes','mango','apple']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:2,fruits:['grapes','mango','cherry']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:3,fruits:['banana','mango']})
WriteResult({ "nInserted" : 1 })
>
```

To find those documents from the “food” collection where the size of the array is two. db.food.find ({“fruits”: {$size:2}})

```
> db.food.find ( {"fruits": {$size:2}} )
{ "_id" : 3, "fruits" : [ "banana", "mango" ] }
>
```

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}})

```
> db.food.find({_id:1},{"fruits":{$slice:2}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
>
```

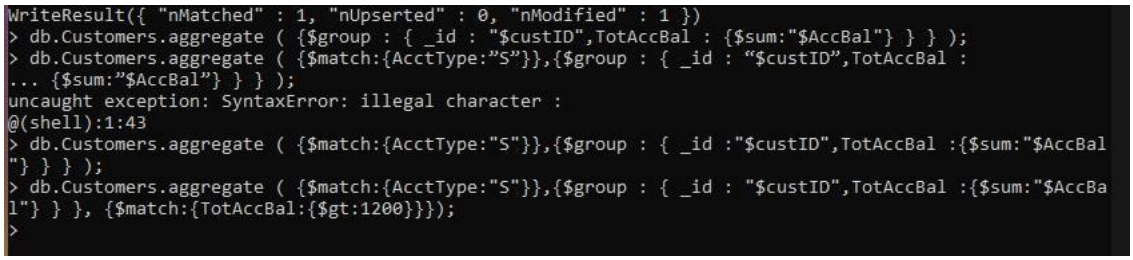
To find all the documets from the food collection which have elements mango and grapes in the array “fruits”
db.food.find({fruits:{$all:[“mango”,”grapes”]}})

```
> db.food.find({fruits:{$all:["mango","grapes"]}})
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
>
```

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},{$set:{'fruits.1':'apple'}})
insert new key value pairs in the fruits array
db.food.update({_id:2},{$push:{price:{grapes:80,mango:200,cherry:100}}})

```
> db.food.update({_id:3},{$set:{'fruits.1':'apple'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.food.update({_id:2},{$push:{price:{grapes:80,mango:200,cherry:100}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

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}}});



```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Customers.aggregate ( {$group : { _id : "$custID",TotAccBal : {$sum:"$AccBal"} } } );
> db.Customers.aggregate ( {$match:{AcctType:”S”}},{$group : { _id : “$custID”,TotAccBal :
... {$sum:”$AccBal”} } } );
uncaught exception: SyntaxError: illegal character :
@(shell):1:43
> db.Customers.aggregate ( {$match:{AcctType:"S"}},{$group : { _id :"$custID",TotAccBal :{$sum:"$AccBal
"} } } );
> db.Customers.aggregate ( {$match:{AcctType:"S"}},{$group : { _id : "$custID",TotAccBal :{$sum:"$AccBa
l"} } }, {$match:{TotAccBal:{$gt:1200}}});
>
```