WORKING WITH MONGODB

I. CREATE DATABASE IN MONGODB.

use myDB;

Confirm the existence of your database

db;

To list all databases

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”); => *sql equivalent* CREATE TABLE STUDENT(…);

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

   db.Student.drop();

```
Atlas atlas-2evra5-shard-0 [primary] test> db.createCollection("Student");
{ ok: 1 }
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.drop();
true
Atlas atlas-2evra5-shard-0 [primary] test>
```

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:"InternetSurfing"});

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetSurfing"
});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{ acknowledged: true, insertedIds: { '0': 1 } }
Atlas atlas-2evra5-shard-0 [primary] test>
```

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:"Skating"}},{upsert:true});

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.update({_id:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:"Skating"}},{up
upsert:true});
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: 3,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
Atlas atlas-2evra5-shard-0 [primary] test>
```

5. FIND METHOD

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

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

```
Atlas atlas-2evra5-shard-0 [primary] test> 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});

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({},{StudName:1,Grade:1,_id:0});
[
  { StudName: 'MichelleJacintha', Grade: 'VII' },
  { Grade: 'VII', StudName: 'AryanDavid' }
]
```

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

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({Grade:{$eq:"VII"}}).pretty()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  { _id: 3, Grade: 'VII', StudName: 'AryanDavid', Hobbies: 'Skating' }
]
```

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 ();

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({Hobbies:{$in:["Chess","Skating"]}});
[
  { _id: 3, Grade: 'VII', StudName: 'AryanDavid', Hobbies: 'Skating' }
]
```

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

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({StudName:/^M/});
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  }
]
```

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

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({StudName:/e/});
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  }
]
```

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

db.Student.count();

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
2
```

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

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find().sort({StudName:-1}).pretty()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  { _id: 3, Grade: 'VII', StudName: 'AryanDavid', Hobbies: 'Skating' }
]
```

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"}})

```
TypeError: db.Student.save is not a function
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.update({_id:4},{$set:{Location:"Network"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

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});

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.update({_id:4},{$set:{Location:"Network"}},{upsert:true});
{
  acknowledged: true,
  insertedId: 4,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({_id:4});
[ { _id: 4, Location: 'Network' } ]
Atlas atlas-2evra5-shard-0 [primary] test>

Atlas atlas-2evra5-shard-0 [primary] test> db.Student.update({_id:4},{$unset:{Location:"Network"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({_id:4});
[ { _id: 4 } ]
Atlas atlas-2evra5-shard-0 [primary] test>
```

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

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({Grade:{$ne:"VII"}}).pretty()
[ { _id: 4 } ]
```

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

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find({StudName:"/s$/"}).pretty()

Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  { _id: 3, Grade: 'VII', StudName: 'AryanDavid', Hobbies: 'Skating' },
  { _id: 4 }
]
```

IX. to set a particular field value to NULL

db.Students.update({_id:3},{$set:{Location:null}})

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.update({_id:3},{$set:{Location:null}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  {
    _id: 3,
    Grade: 'VII',
    StudName: 'AryanDavid',
    Hobbies: 'Skating',
    Location: null
  },
  { _id: 4 }
]
```

X. Count the number of documents in Student Collections

db.Students.count()

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.count()
3
```

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

db.Students.count({Grade:"VII"})

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.count({Grade:"VII"});
2
```

retrieve first 3 documents

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find().limit(3).pretty()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  {
    _id: 3,
    Grade: 'VII',
    StudName: 'AryanDavid',
    Hobbies: 'Skating',
    Location: null
  },
  { _id: 4 }
]
```

Sort the document in Ascending order

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find().sort({StudName:1});
[
  { _id: 4 },
  {
    _id: 3,
    Grade: 'VII',
    StudName: 'AryanDavid',
    Hobbies: 'Skating',
    Location: null
  },
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  }
]
```

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

to Skip the 1st two documents from the Students Collections

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Student.find().skip(2).pretty()
[ { _id: 4 } ]
```

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'] })

```
Atlas atlas-2evra5-shard-0 [primary] test> db.food.insert( { _id:1, fruits:['grapes','mango','apple'] } )
{ acknowledged: true, insertedIds: { '0': 1 } }
Atlas atlas-2evra5-shard-0 [primary] test> db.food.insert( { _id:2, fruits:['grapes','mango','cherry'] } )
{ acknowledged: true, insertedIds: { '0': 2 } }
Atlas atlas-2evra5-shard-0 [primary] test> db.food.insert( { _id:3, fruits:['banana','mango'] } )
{ acknowledged: true, insertedIds: { '0': 3 } }
```

To find those documents from the “food” collection which has the “fruits array” constitute of “grapes”, “mango” and “apple”.

db.food.find ({fruits: ['grapes','mango','apple'] }). pretty().

```
Atlas atlas-2evra5-shard-0 [primary] test> db.food.find({fruits: ['grapes','mango','apple']}).pretty()
[ { _id: 1, fruits: [ 'grapes', 'mango', 'apple' ] } ]
```

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

db.food.find ({'fruits.1':'mango'})

```
Atlas atlas-2evra5-shard-0 [primary] test> db.food.find({"fruits.1":"mango"});
[
  { _id: 1, fruits: [ 'grapes', 'mango', 'apple' ] },
  { _id: 2, fruits: [ 'grapes', 'mango', 'cherry' ] },
  { _id: 3, fruits: [ 'banana', 'mango' ] }
]
```

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

db.food.find ({“fruits”: {$size:2}})

```
Atlas atlas-2evra5-shard-0 [primary] test> db.food.find({fruits:{$size:2}});
[ { _id: 3, fruits: [ 'banana', 'mango' ] } ]
Atlas atlas-2evra5-shard-0 [primary] test>
```

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}})

```
Atlas atlas-2evra5-shard-0 [primary] test> 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”]}})

```
Atlas atlas-2evra5-shard-0 [primary] test> 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 1st index position of the fruits array with apple

db.food.update({_id:3},{$set:{'fruits.1':'apple'}})

```
Atlas atlas-2evra5-shard-0 [primary] test> db.food.update({_id:3},{$set:{"fruits.1":"apple"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

insert new key value pairs in the fruits array

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

```
Atlas atlas-2evra5-shard-0 [primary] test> db.food.update({_id:2},{$push:{price:{grapes:80,mango:200,cherry:100}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-2evra5-shard-0 [primary] test> db.food.find()
[
  { _id: 1, fruits: [ 'grapes', 'mango', 'apple' ] },
  {
    _id: 2,
    fruits: [ 'grapes', 'mango', 'cherry' ],
    price: [ { grapes: 80, mango: 200, cherry: 100 } ]
  },
  { _id: 3, fruits: [ 'banana', 'apple' ] }
]
```

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”} } });

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Customers.aggregate({$group:{_id:"$cust_id",TotalAccBal:{$sum:"$Balance"}}});
[
  { _id: 2, TotalAccBal: 1650 },
  { _id: 3, TotalAccBal: 500 },
  { _id: 1, TotalAccBal: 1200 }
]
```

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”} } });

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Customers.aggregate({$match:{Type:"S"}},{$group:{_id:"$cust_id",TotalAccBal:{$sum:"$Bal
ance"}}});
[ { _id: 2, TotalAccBal: 50 }, { _id: 1, TotalAccBal: 200 } ]
```

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}}});

```
Atlas atlas-2evra5-shard-0 [primary] test> db.Customers.aggregate({$match:{Type:"S"}},{$group:{_id:"$cust_id",TotalAccBal:{$sum:"$Bal
ance"}}},{$match:{TotAccBal:{$gt:10}}});
```