VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

Praveen Kumar S (1BM20CS413)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
May-2022 to July-2022

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**BIG DATA ANALYTICS**" carried out by **Praveen Kumar S (1BM20CS413)** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **BIG DATA ANALYTICS** work prescribed for the said degree.

Mrs. Antara Roy Choudhury
Designation
Assistant Professor
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

Index Sheet

Course Outcome

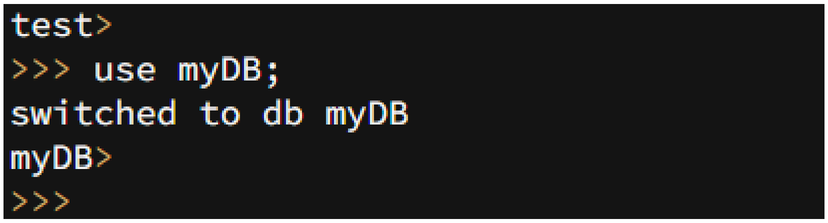
CO1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO2	Analyze the Big Data and obtain insight using data analytics mechanisms.
CO3	Design and implement Big data applications by applying NoSQL, Hadoop or Spark

WORKING WITH MONGODB

I.CREATE DATABASE IN MONGODB.

use myDB;

Confirm the existence of your database

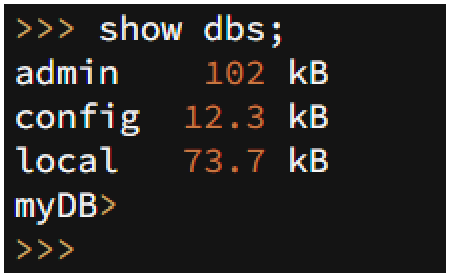


```
test>
>>> use myDB;
switched to db myDB
myDB>
>>>
```

db;

To list all databases

show dbs;

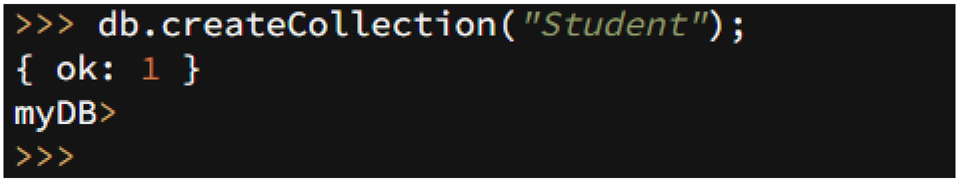


```
>>> show dbs;
admin     102 kB
config   12.3 kB
local    73.7 kB
myDB>
>>>
```

I.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(…);**



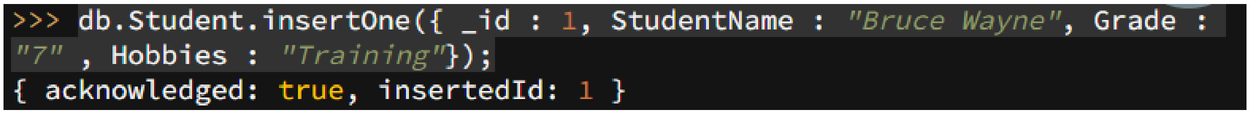
```
>>> db.createCollection("Student");
{ ok: 1 }
myDB>
>>>
```

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

db.Student.drop();

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

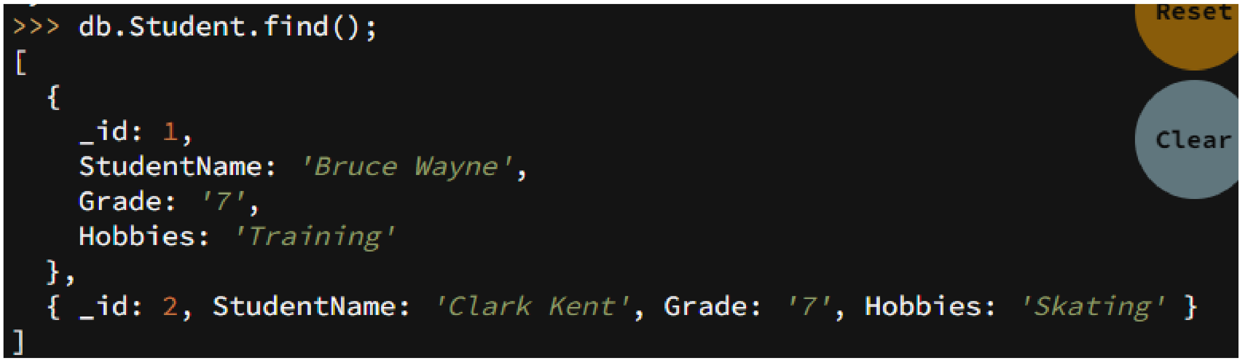
db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetSurfing"});



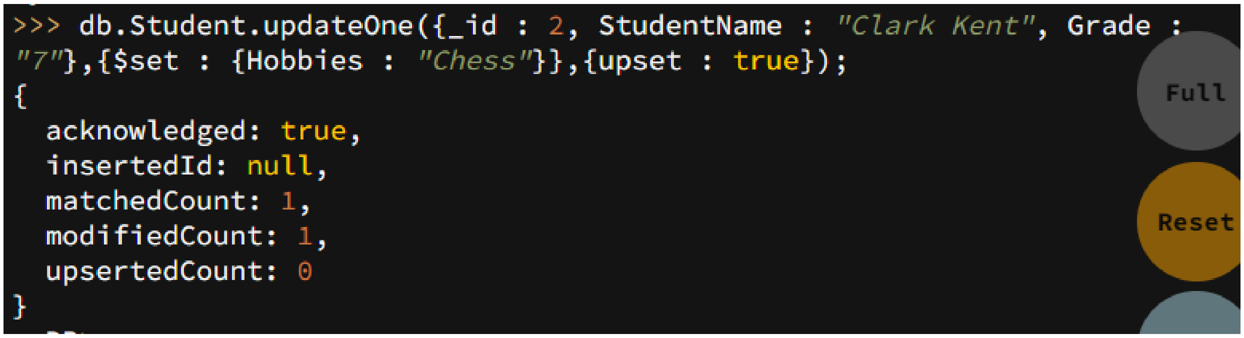
```
>>> db.Student.insertOne({ _id : 1, StudentName : "Bruce Wayne", Grade :
"7" , Hobbies : "Training"});
{ acknowledged: true, insertedId: 1 }
```

1. 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).

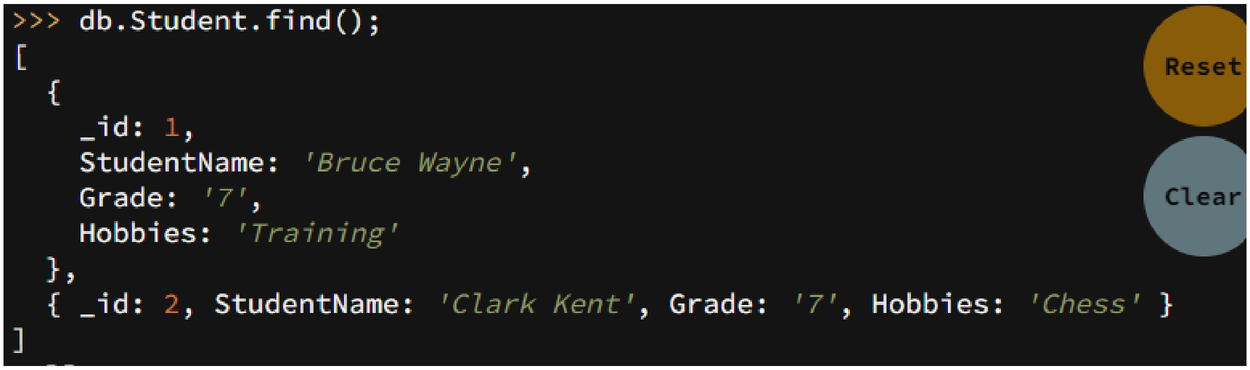
db.Student.update({_id:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true});



```
>>> db.Student.find();
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training'
  },
  { _id: 2, StudentName: 'Clark Kent', Grade: '7', Hobbies: 'Skating' }
]
```



```
>>> db.Student.updateOne({_id : 2, StudentName : "Clark Kent", Grade :
"7"},{$set : {Hobbies : "Chess"}},{upset : true});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

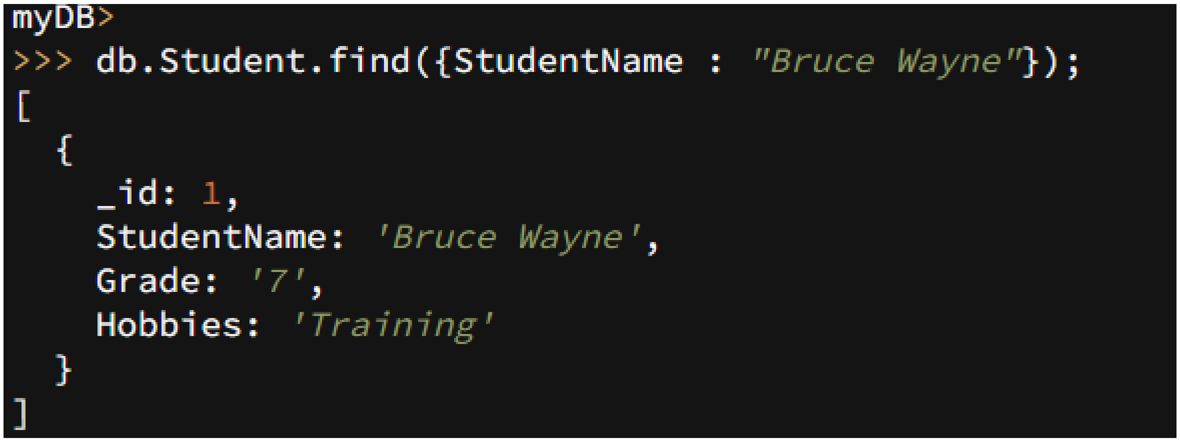


```
>>> db.Student.find();
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training'
  },
  { _id: 2, StudentName: 'Clark Kent', Grade: '7', Hobbies: 'Chess' }
]
```

1. FIND METHOD

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

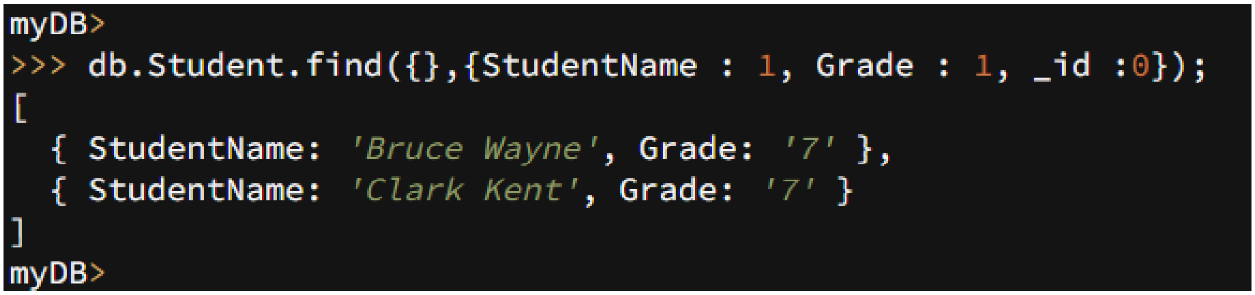
db.Student.find({StudName:"Aryan David"});
({cond..},{columns.. column:1, columnname:0}   )



```
myDB>
>>> db.Student.find({StudentName : "Bruce Wayne"});
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training'
  }
]
```

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.

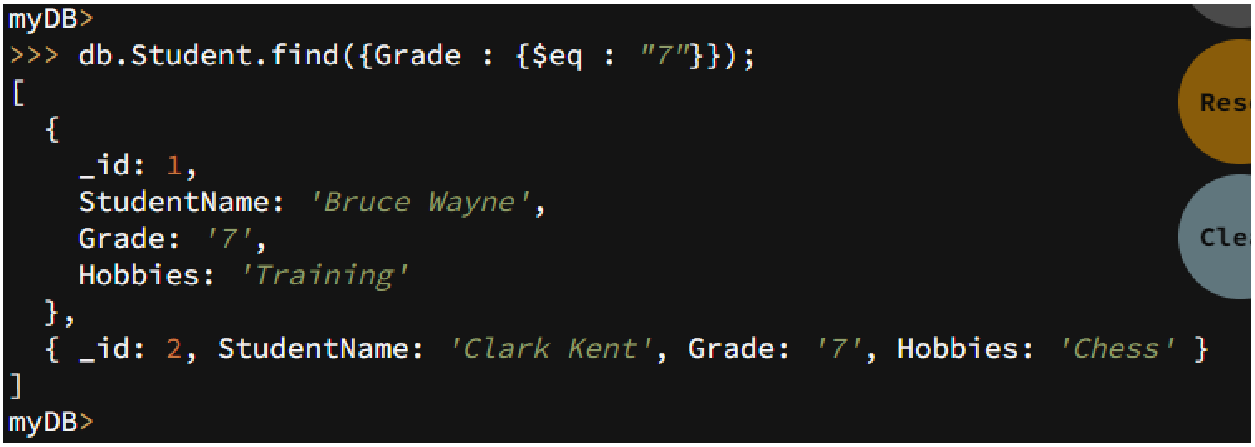
db.Student.find({},{StudName:1,Grade:1,_id:0});



```
myDB>
>>> db.Student.find({},{StudentName : 1, Grade : 1, _id :0});
[
  { StudentName: 'Bruce Wayne', Grade: '7' },
  { StudentName: 'Clark Kent', Grade: '7' }
]
myDB>
```

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

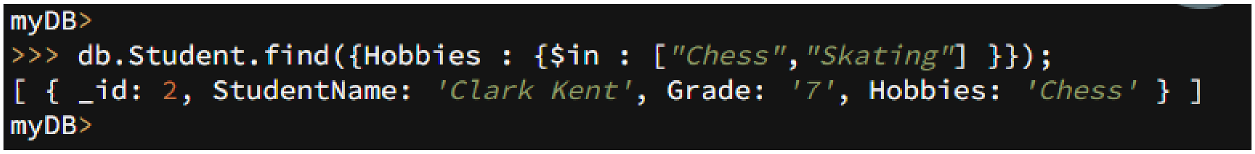
db.Student.find({Grade:{$eq:'VII'}}).pretty();



```
myDB>
>>> db.Student.find({Grade : {$eq : "7"}});
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training'
  },
  { _id: 2, StudentName: 'Clark Kent', Grade: '7', Hobbies: 'Chess' }
]
myDB>
```

D. To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’.

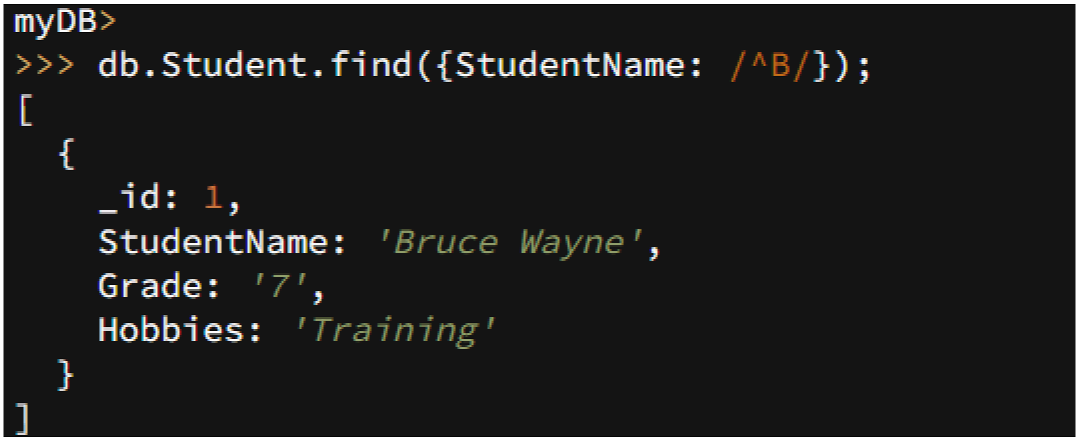
db.Student.find({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();



```
myDB>
>>> db.Student.find({Hobbies : {$in : ["Chess","Skating"] }});
[ { _id: 2, StudentName: 'Clark Kent', Grade: '7', Hobbies: 'Chess' } ]
myDB>
```

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

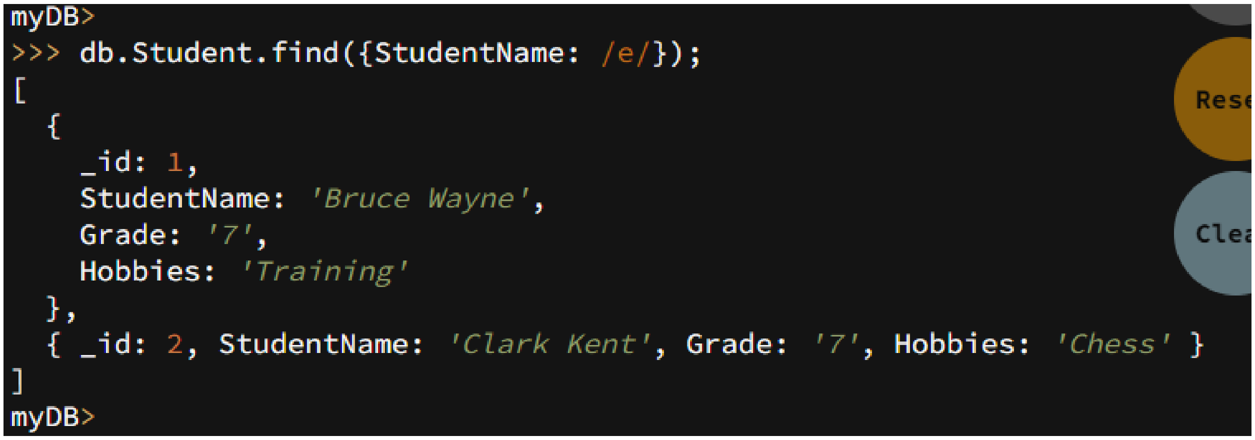
db.Student.find({StudName:/^M/}).pretty();



```
myDB>
>>> db.Student.find({StudentName: /^B/});
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training'
  }
]
```

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

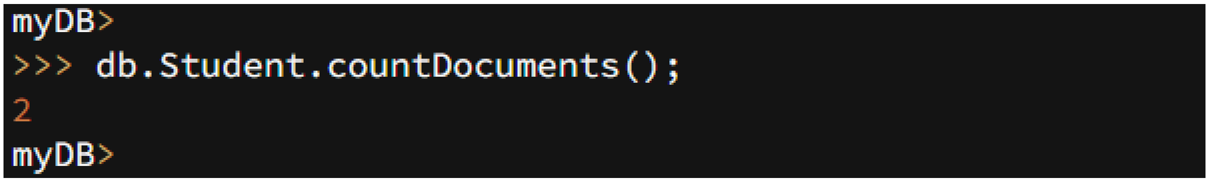
db.Student.find({StudName:/e/}).pretty();



```
myDB>
>>> db.Student.find({StudentName: /e/});
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training'
  },
  { _id: 2, StudentName: 'Clark Kent', Grade: '7', Hobbies: 'Chess' }
]
myDB>
```

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

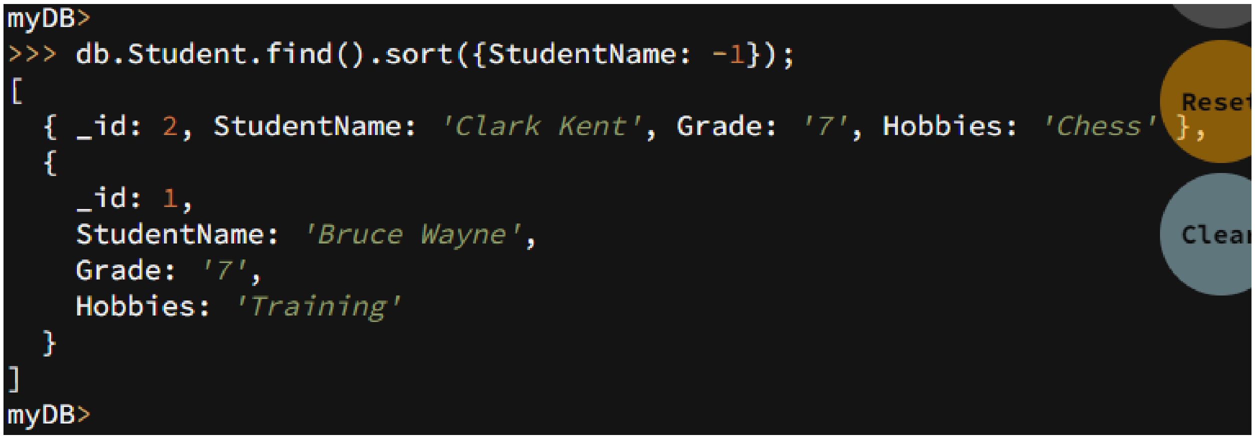
db.Student.count();



```
myDB>
>>> db.Student.countDocuments();
2
myDB>
```

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

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



```
myDB>
>>> db.Student.find().sort({StudentName: -1});
[
  { _id: 2, StudentName: 'Clark Kent', Grade: '7', Hobbies: 'Chess' },
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training'
  }
]
myDB>
```

I.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

I.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”

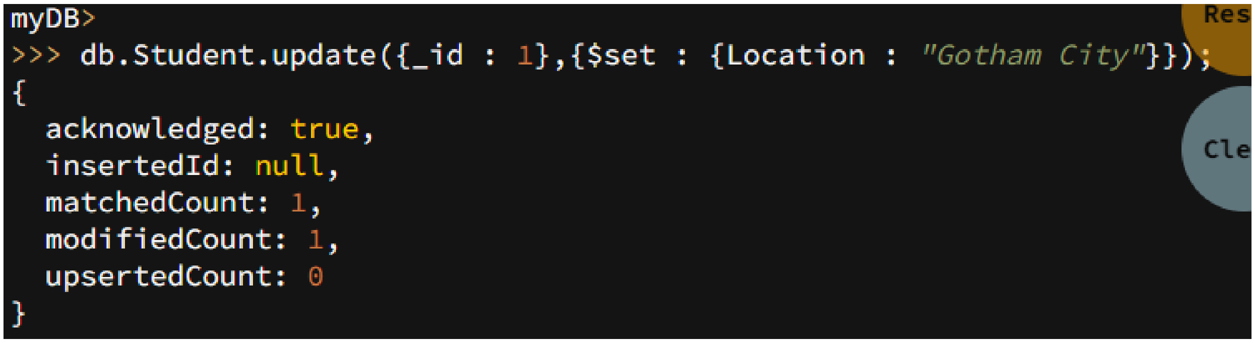
I.Save Method :

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.

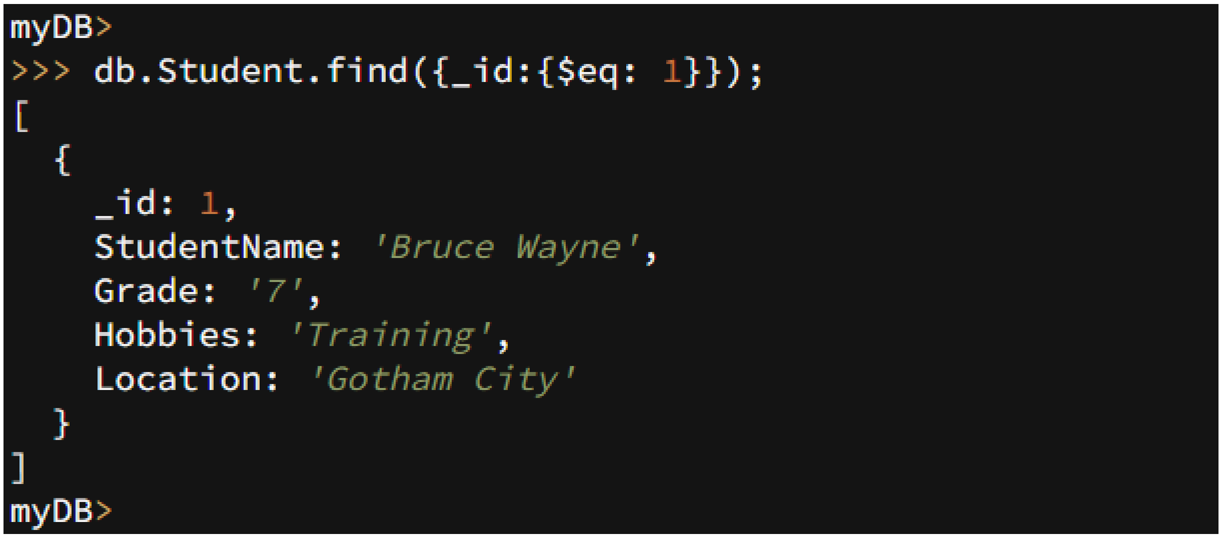
```
db.Students.save({StudName:”Vamsi”, Grade:”VI”})
```

I. Add a new field to existing Document:

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



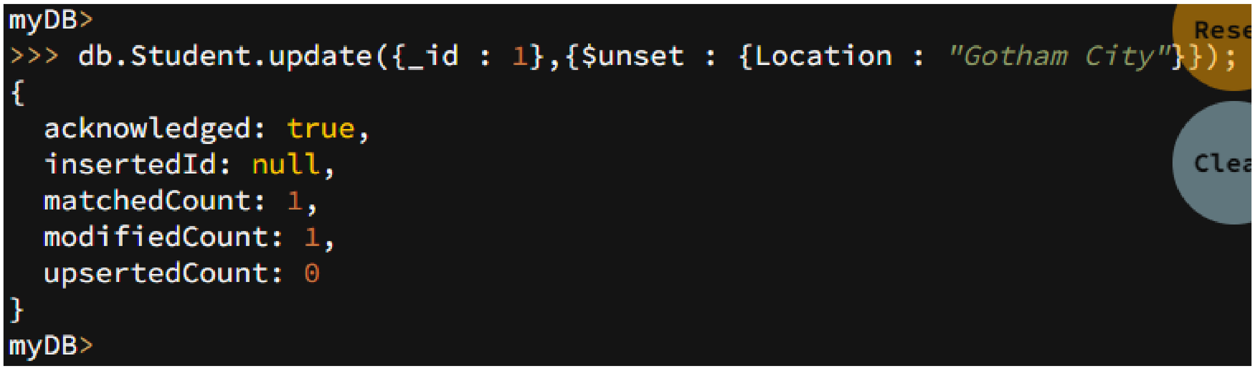
```
myDB>
>>> db.Student.update({_id : 1},{$set : {Location : "Gotham City"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```



```
myDB>
>>> db.Student.find({_id:{$eq: 1}});
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training',
    Location: 'Gotham City'
  }
]
myDB>
```

I.Remove the field in an existing Document

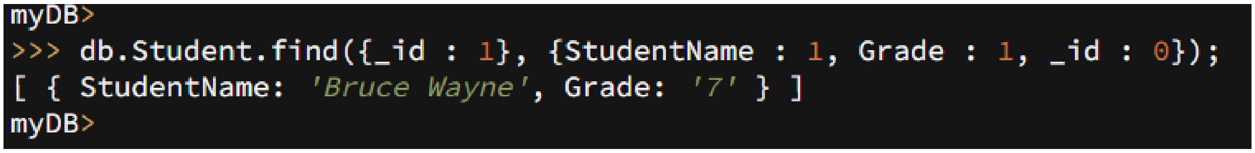
db.Students.update({_id:4},{$unset:{Location:”Network”}})



```
myDB>
>>> db.Student.update({_id : 1},{$unset : {Location : "Gotham City"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
myDB>
```

I. Finding Document based on search criteria suppressing few fields

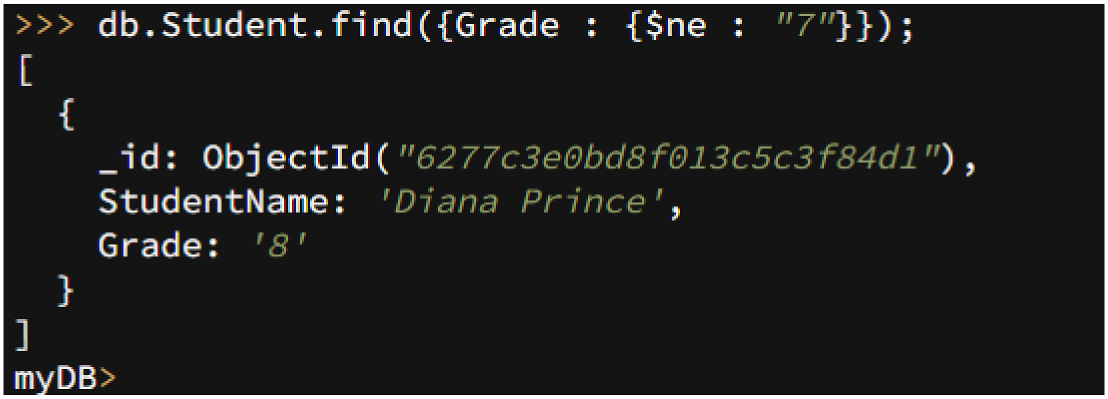
db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});



```
myDB>
>>> db.Student.find({_id : 1}, {StudentName : 1, Grade : 1, _id : 0});
[ { StudentName: 'Bruce Wayne', Grade: '7' } ]
myDB>
```

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

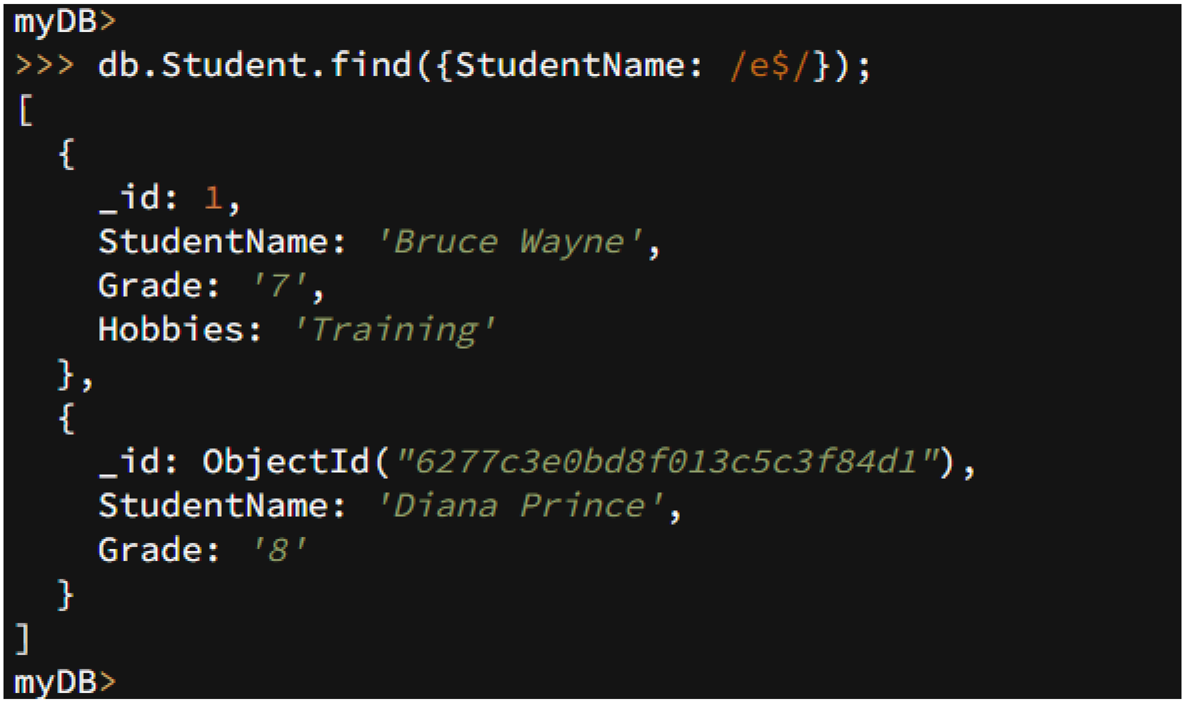
db.Student.find({Grade:{$ne:'VII'}}).pretty();



```
>>> db.Student.find({Grade : {$ne : "7"}});
[
  {
    _id: ObjectId("6277c3e0bd8f013c5c3f84d1"),
    StudentName: 'Diana Prince',
    Grade: '8'
  }
]
myDB>
```

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

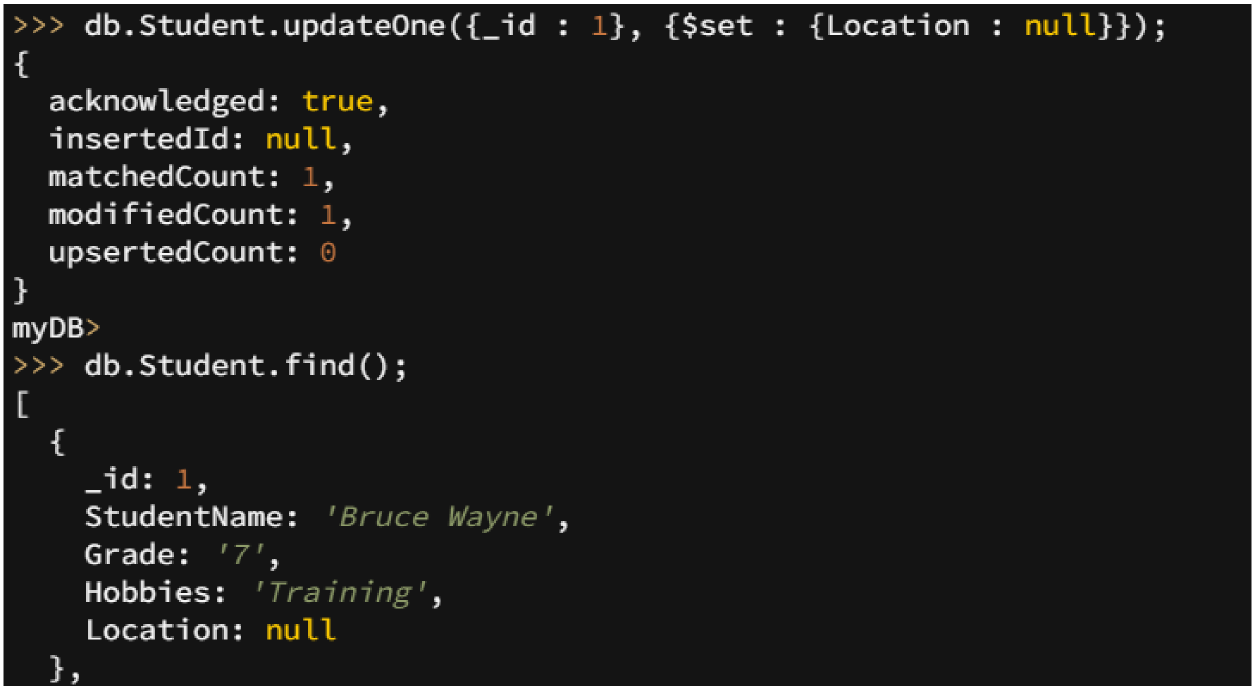
db.Student.find({StudName:/s$/}).pretty();



```
myDB>
>>> db.Student.find({StudentName: /e$/});
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training'
  },
  {
    _id: ObjectId("6277c3e0bd8f013c5c3f84d1"),
    StudentName: 'Diana Prince',
    Grade: '8'
  }
]
myDB>
```

I.to set a particular field value to NULL

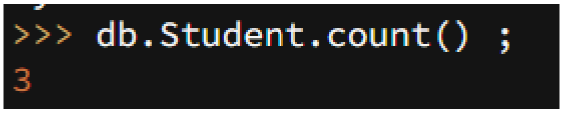
db.Students.update({_id:3},{$set:{Location:null}})



```
>>> db.Student.updateOne({_id : 1}, {$set : {Location : null}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
myDB>
>>> db.Student.find();
[
  {
    _id: 1,
    StudentName: 'Bruce Wayne',
    Grade: '7',
    Hobbies: 'Training',
    Location: null
  },
```

I.Count the number of documents in Student Collections

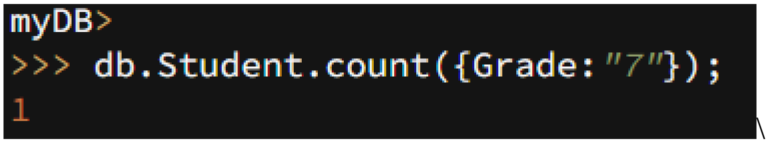
db.Students.count()



```
>>> db.Student.count() ;
3
```

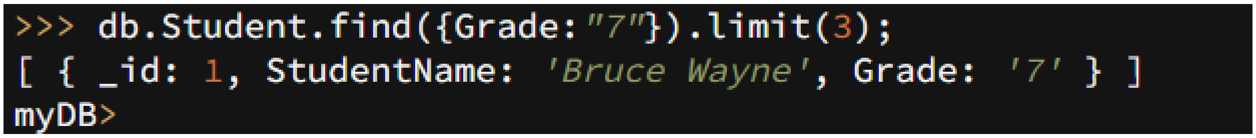
I.Count the number of documents in Student Collections with grade :VII

db.Students.count({Grade:”VII”})



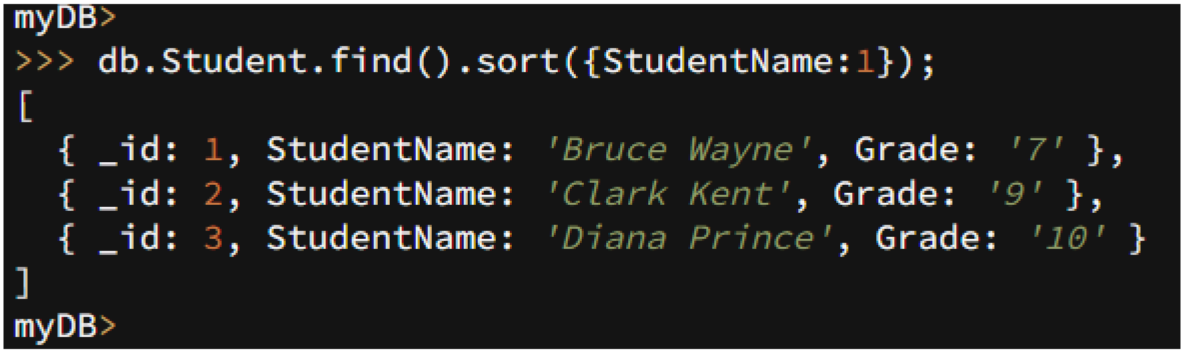
```
myDB>
>>> db.Student.count({Grade:"7"});
1
```

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



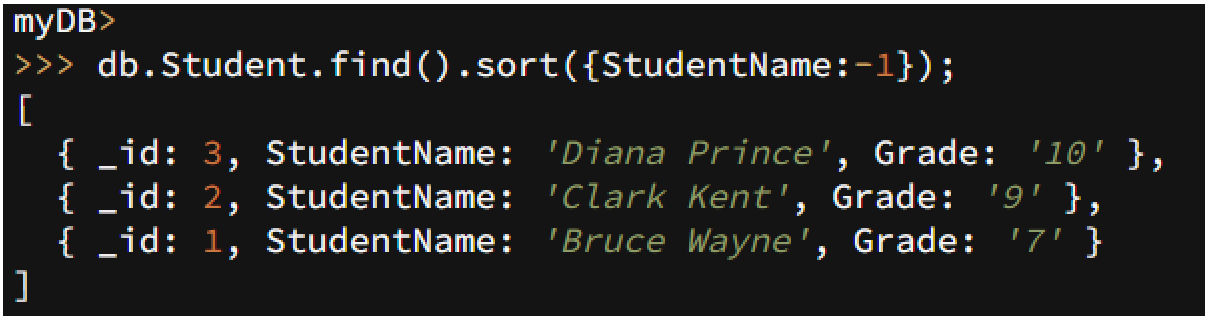
```
>>> db.Student.find({Grade:"7"}).limit(3);
[ { _id: 1, StudentName: 'Bruce Wayne', Grade: '7' } ]
myDB>
```

Sort the document in Ascending order
db.Students.find().sort({StudName:1}).pretty();



```
myDB>
>>> db.Student.find().sort({StudentName:1});
[
  { _id: 1, StudentName: 'Bruce Wayne', Grade: '7' },
  { _id: 2, StudentName: 'Clark Kent', Grade: '9' },
  { _id: 3, StudentName: 'Diana Prince', Grade: '10' }
]
myDB>
```

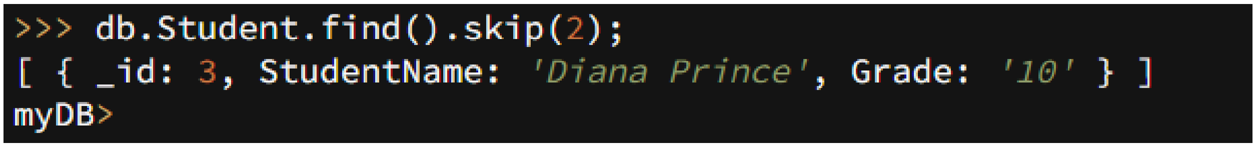
Note:
for desending order : db.Students.find().sort({StudName:-1}).pretty();



```
myDB>
>>> db.Student.find().sort({StudentName:-1});
[
  { _id: 3, StudentName: 'Diana Prince', Grade: '10' },
  { _id: 2, StudentName: 'Clark Kent', Grade: '9' },
  { _id: 1, StudentName: 'Bruce Wayne', Grade: '7' }
]
```

to Skip the 1st two documents from the Students Collections

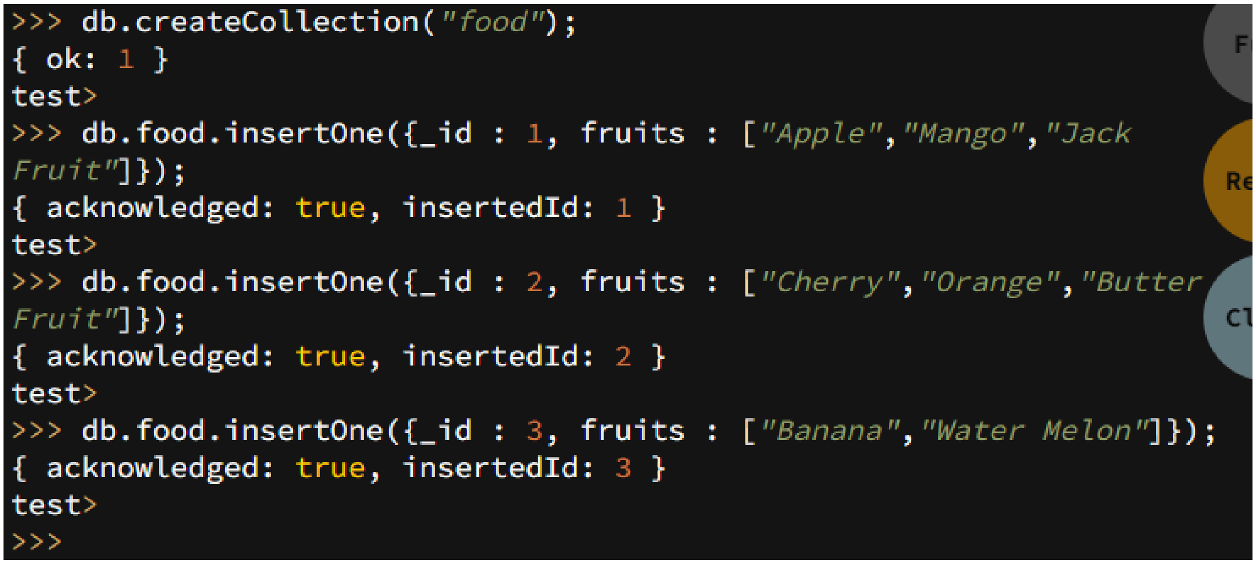
db.Students.find().skip(2).pretty()



```
>>> db.Student.find().skip(2);
[ { _id: 3, StudentName: 'Diana Prince', Grade: '10' } ]
myDB>
```

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

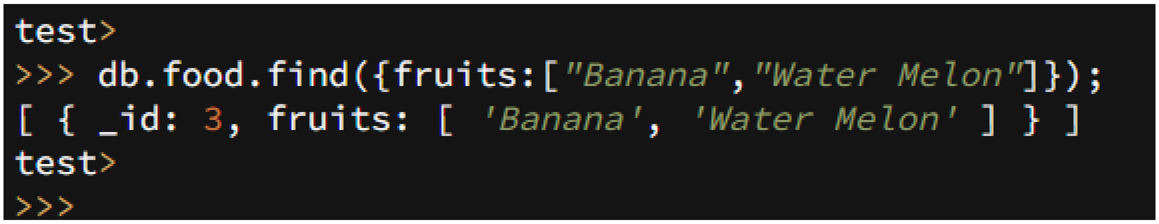
db.food.insert({ _id:1, fruits:['grapes','mango','apple'] })
db.food.insert({ _id:2, fruits:['grapes','mango','cherry'] })
db.food.insert({ _id:3, fruits:['banana','mango'] })



```
>>> db.createCollection("food");
{ ok: 1 }
test>
>>> db.food.insertOne({_id : 1, fruits : ["Apple","Mango","Jack
Fruit"]});
{ acknowledged: true, insertedId: 1 }
test>
>>> db.food.insertOne({_id : 2, fruits : ["Cherry","Orange","Butter
Fruit"]});
{ acknowledged: true, insertedId: 2 }
test>
>>> db.food.insertOne({_id : 3, fruits : ["Banana","Water Melon"]});
{ acknowledged: true, insertedId: 3 }
test>
>>>
```

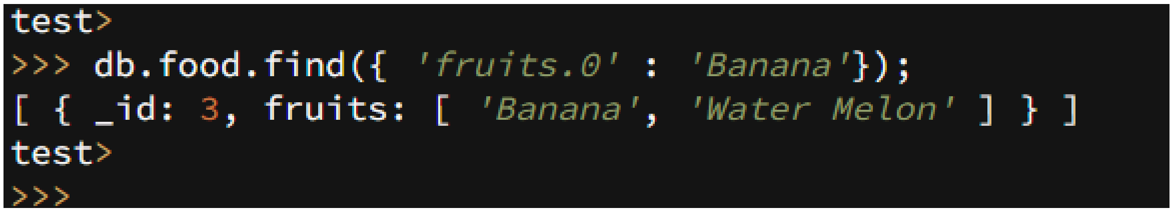
To find those documents from the “food” collection which has the “fruits array” constitute of “grapes”, “mango” and “apple”.

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



```
test>
>>> db.food.find({fruits:["Banana","Water Melon"]});
[ { _id: 3, fruits: [ 'Banana', 'Water Melon' ] } ]
test>
>>>
```

To find in “fruits” array having “mango” in the first index position.
db.food.find ({'fruits.1':'grapes'})



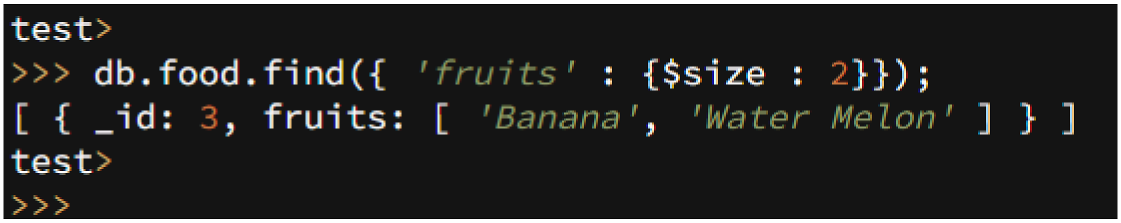
```
test>
>>> db.food.find({ 'fruits.0' : 'Banana'});
[ { _id: 3, fruits: [ 'Banana', 'Water Melon' ] } ]
test>
>>>
```

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”

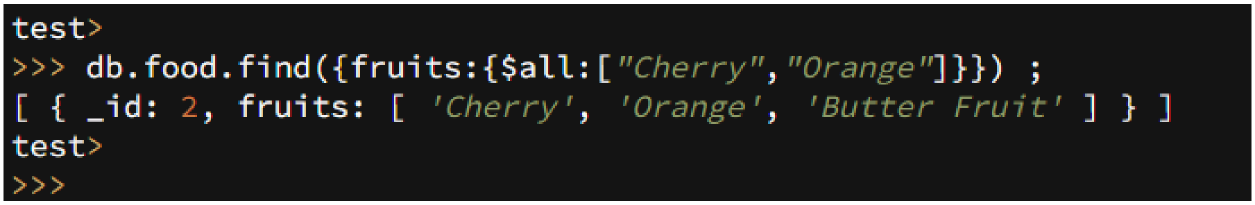
db.food.find({_id:1},{“fruits”:{$slice:2}})



```
test>
>>> db.food.find({ 'fruits' : {$size : 2}});
[ { _id: 3, fruits: [ 'Banana', 'Water Melon' ] } ]
test>
>>>
```

To find all the documets from the food collection which have elements mango and grapes in the array “fruits”

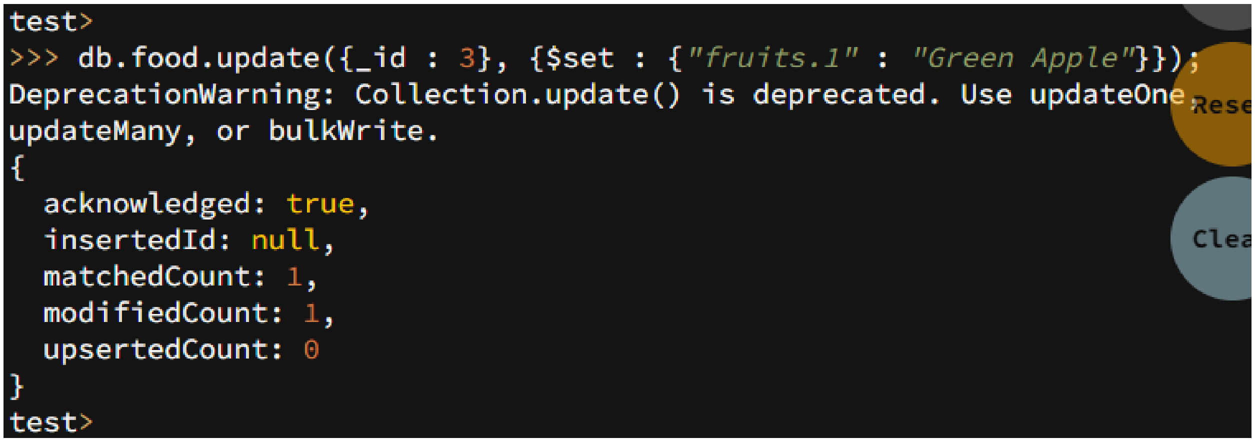
db.food.find({fruits:{$all:[“mango”,”grapes”]}})



```
test>
>>> db.food.find({fruits:{$all:["Cherry","Orange"]}}) ;
[ { _id: 2, fruits: [ 'Cherry', 'Orange', 'Butter Fruit' ] } ]
test>
>>>
```

update on Array:
using particular id replace the element present in the 1st index position of the fruits array with apple

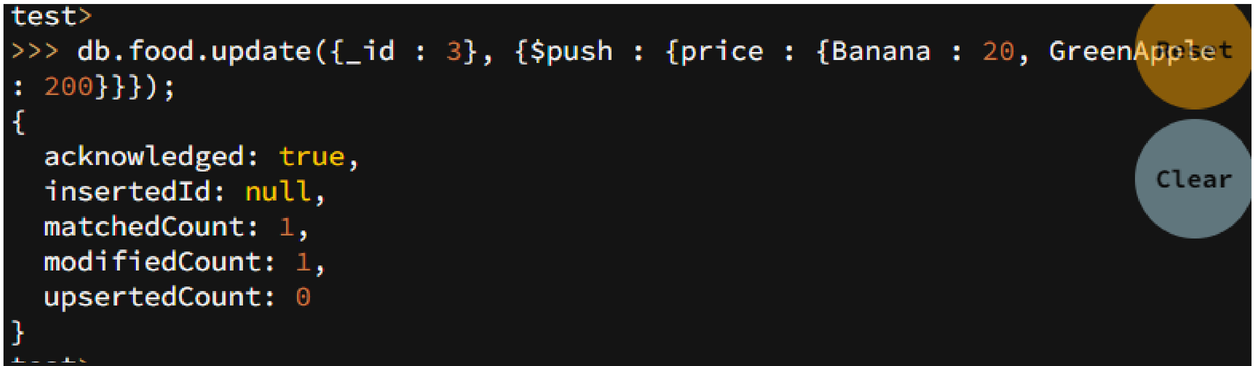
db.food.update({_id:3},{$set:{'fruits.1':'apple'}})



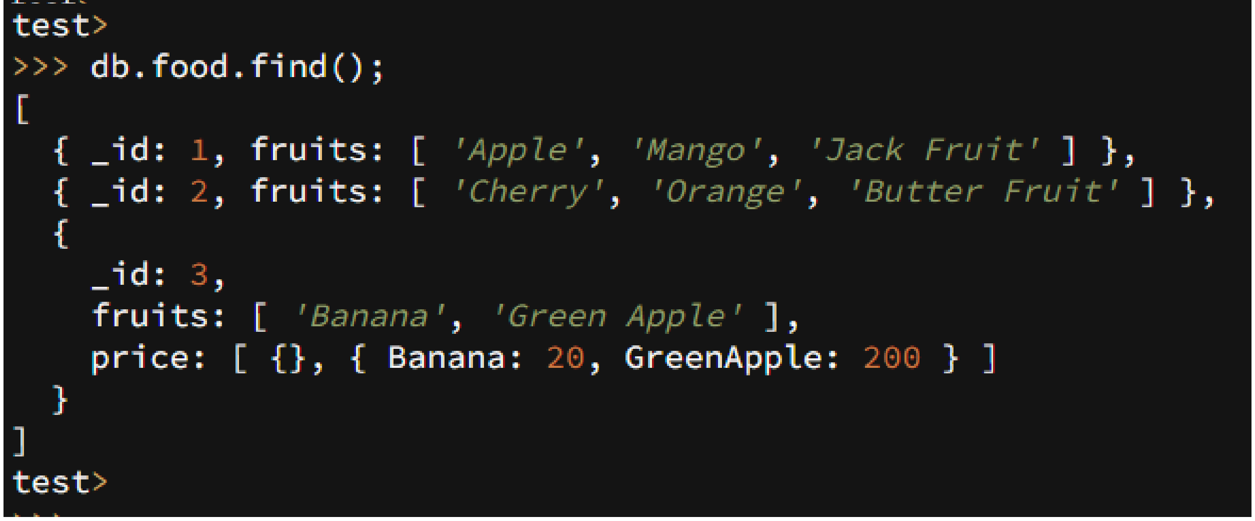
```
test>
>>> db.food.update({_id : 3}, {$set : {"fruits.1" : "Green Apple"}});
DeprecationWarning: Collection.update() is deprecated. Use updateOne,
updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test>
```

insert new key value pairs in the fruits array

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



```
test>
>>> db.food.update({_id : 3}, {$push : {price : {Banana : 20, GreenApple
: 200}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

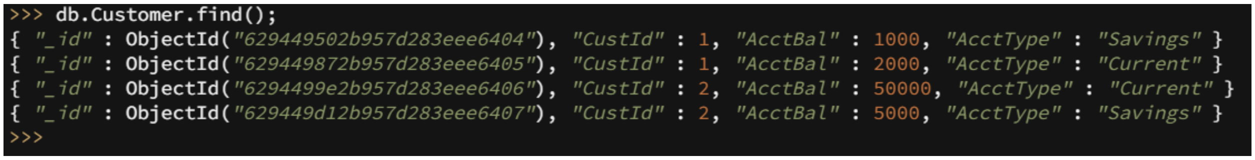


```
test>
>>> db.food.find();
[
  { _id: 1, fruits: [ 'Apple', 'Mango', 'Jack Fruit' ] },
  { _id: 2, fruits: [ 'Cherry', 'Orange', 'Butter Fruit' ] },
  {
    _id: 3,
    fruits: [ 'Banana', 'Green Apple' ],
    price: [ {}, { Banana: 20, GreenApple: 200 } ]
  }
]
test>
```

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

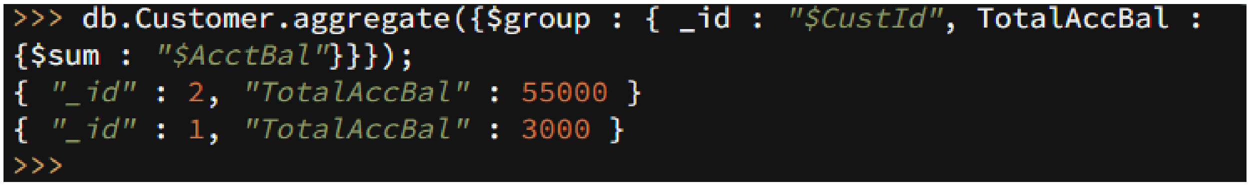
XII. Aggregate Function :

Create a collection Customers with fields custID, AcctBal, AcctType.
Now group on “custID” and compute the sum of “AccBal”.



```
>>> db.Customer.find();
{ "_id" : ObjectId("629449502b957d283eee6404"), "CustId" : 1, "AcctBal" : 1000, "AcctType" : "Savings" }
{ "_id" : ObjectId("629449872b957d283eee6405"), "CustId" : 1, "AcctBal" : 2000, "AcctType" : "Current" }
{ "_id" : ObjectId("6294499e2b957d283eee6406"), "CustId" : 2, "AcctBal" : 50000, "AcctType" : "Current" }
{ "_id" : ObjectId("629449d12b957d283eee6407"), "CustId" : 2, "AcctBal" : 5000, "AcctType" : "Savings" }
>>>
```

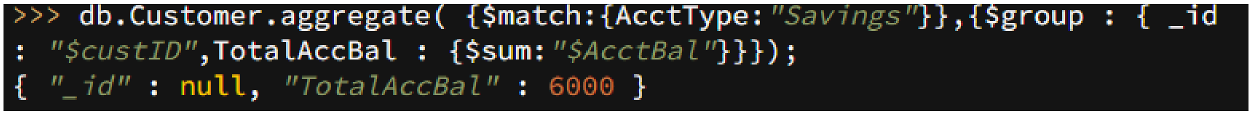
db.Customers.aggregate ({$group : { _id : “$custID”,TotAccBal : {$sum:”$AccBal”} } });



```
>>> db.Customer.aggregate({$group : { _id : "$CustId", TotalAccBal :
{$sum : "$AcctBal"}}});
{ "_id" : 2, "TotalAccBal" : 55000 }
{ "_id" : 1, "TotalAccBal" : 3000 }
>>>
```

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

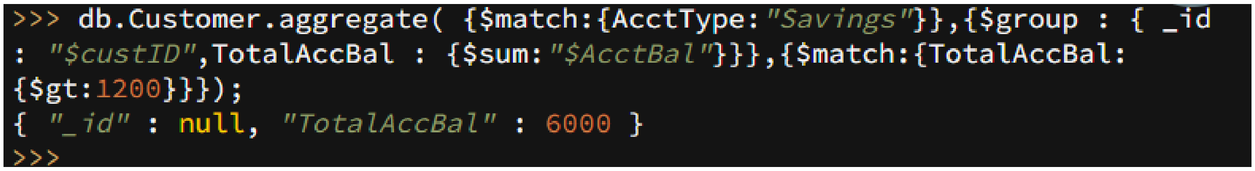
db.Customers.aggregate ({$match:{AcctType:”S”}},{$group : { _id : “$custID”,TotAccBal : {$sum:”$AccBal”} } });



```
>>> db.Customer.aggregate( {$match:{AcctType:"Savings"}},{$group : { _id
: "$custID",TotalAccBal : {$sum:"$AcctBal"}}});
{ "_id" : null, "TotalAccBal" : 6000 }
```

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}}});



```
>>> db.Customer.aggregate( {$match:{AcctType:"Savings"}},{$group : { _id
: "$custID",TotalAccBal : {$sum:"$AcctBal"}}},{$match:{TotalAccBal:
{$gt:1200}}});
{ "_id" : null, "TotalAccBal" : 6000 }
>>>
```

Cassandra Program - 1

1. Create a key space by name Employee

```
cqlsh> CREATE KEYSPACE Empyolees WITH REPLICATION = { 'class' : 'SimpleStrategy',
'replication_factor' : 1 };

cqlsh> DESCRIBE KEYSPACES;

system_schema  crud    project  system_distributed  system_traces
system_auth    system  student  empyolees

cqlsh> USE Employees;
```

2. Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name

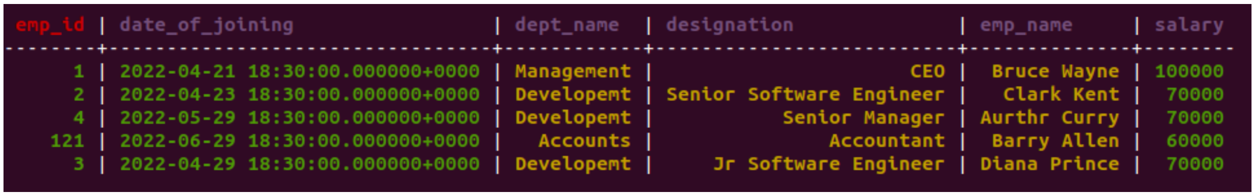
```
cqlsh:employees> CREATE TABLE Employee_Info (
            ... Emp_Id int PRIMARY KEY,
            ... Emp_Name text,
            ... Designation text,
            ... Date_Of_Joining timestamp,
            ... Salary int,
            ... Dept_Name text
            ... );

cqlsh:employees> DESCRIBE TABLES;

employee_info

cqlsh:employees> DESCRIBE TABLE Employee_Info;

CREATE TABLE employees.employee_info (
    emp_id int PRIMARY KEY,
    date_of_joining timestamp,
    dept_name text,
    designation text,
    emp_name text,
    salary int
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32',
'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
```

AND crc_check_chance = 1.0
AND dclocal_read_repair_chance = 0.1
AND default_time_to_live = 0
AND gc_grace_seconds = 864000
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 0
AND min_index_interval = 128
AND read_repair_chance = 0.0
AND speculative_retry = '99PERCENTILE';

3. Insert the values into the table in batch

```
cqlsh:employees> BEGIN BATCH
            ... INSERT INTO Employee_Info (Emp_Id,Emp_Name,Designation,Date_of_Joining,Salary,Dept_Name) VALUES (1,'Bruce Wayne','CEO','2022-04-22',100000,'Management')
            ... INSERT INTO Employee_Info (Emp_Id,Emp_Name,Designation,Date_of_Joining,Salary,Dept_Name) VALUES (2,'Clark Kent','Senior Software Engineer','2022-04-24',70000,'Developemt')
            ... INSERT INTO Employee_Info (Emp_Id,Emp_Name,Designation,Date_of_Joining,Salary,Dept_Name) VALUES (3,'Diana Prince','Jr Software Engineer','2022-04-30',70000,'Developemt')
            ... INSERT INTO Employee_Info (Emp_Id,Emp_Name,Designation,Date_of_Joining,Salary,Dept_Name) VALUES (4,'Aurthr Curry','Senior Manager','2022-05-30',70000,'Developemt')
            ... APPLY BATCH;
```

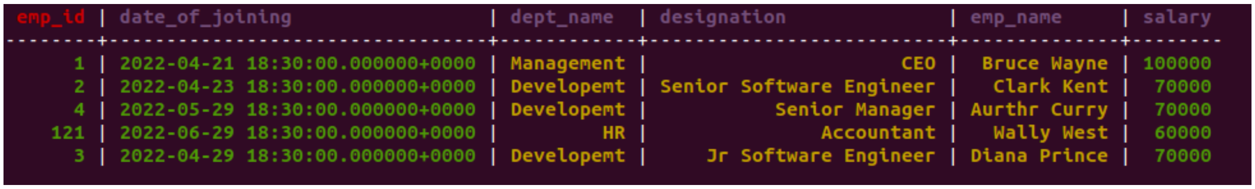
```
cqlsh:employees> SELECT * FROM Employee_Info;
```



```
 emp_id | date_of_joining                 | dept_name  | designation              | emp_name     | salary
--------+---------------------------------+------------+--------------------------+--------------+--------
      1 | 2022-04-21 18:30:00.000000+0000 | Management |                      CEO |  Bruce Wayne | 100000
      2 | 2022-04-23 18:30:00.000000+0000 | Developemt | Senior Software Engineer |   Clark Kent |  70000
      4 | 2022-05-29 18:30:00.000000+0000 | Developemt |           Senior Manager | Aurthr Curry |  70000
    121 | 2022-06-29 18:30:00.000000+0000 |   Accounts |               Accountant |  Barry Allen |  60000
      3 | 2022-04-29 18:30:00.000000+0000 | Developemt |     Jr Software Engineer | Diana Prince |  70000
```

4. Update Employee name and Department of Emp-Id 121

```
cqlsh:employees> UPDATE Employee_Info SET Emp_Name = 'Wally West', dept_name = 'HR' WHERE Emp_id = 121;
```



```
 emp_id | date_of_joining                 | dept_name  | designation              | emp_name     | salary
--------+---------------------------------+------------+--------------------------+--------------+--------
      1 | 2022-04-21 18:30:00.000000+0000 | Management |                      CEO |  Bruce Wayne | 100000
      2 | 2022-04-23 18:30:00.000000+0000 | Developemt | Senior Software Engineer |   Clark Kent |  70000
      4 | 2022-05-29 18:30:00.000000+0000 | Developemt |           Senior Manager | Aurthr Curry |  70000
    121 | 2022-06-29 18:30:00.000000+0000 |         HR |               Accountant |   Wally West |  60000
      3 | 2022-04-29 18:30:00.000000+0000 | Developemt |     Jr Software Engineer | Diana Prince |  70000
```

5. Sort the details of Employee records based on salary

```
cqlsh:employees> CREATE TABLE Employee_Info (
            ... Emp_Id int,
            ... Emp_Name text,
            ... Designation text,
            ... Date_Of_Joining timestamp,
            ... Salary int,
            ... Dept_Name text,
           … PRIMARY KEY (Emp_Id , Salary)
            ... ) WITH CLUSTERING ORDER BY (Salary desc);
```

```
cqlsh:employee> select * from Employee_Info;
```

```
 emp_id | date_of_joining                     | dept_name          | designation              | emp_name          |salary
---------+---------------------------------------+--------------------+-------------------------------+---------------+--------
  121 | 2022-06-29 18:30:00.000000+0000 |       HR               |         Accountant              | Wally West    |60000
    3 | 2022-04-29 18:30:00.000000+0000 |       Development|    Jr Software Manager    | Diana Prince |70000
    2 | 2022-04-23 18:30:00.000000+0000 |       Management |Senior Software Manager|Clark Kent     |70000
    4 | 2022-05-29 18:30:00.000000+0000 |       Development|         Senior Manager       | Aurthur Curry |70000
 1 | 2022-04-21 18:30:00.000000+0000 |       Management |                   CEO              |Bruce Wayne  |100000
```

6. Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.

```
cqlsh:employee> ALTER TABLE Employee_Info  ADD Projects text;
```

```
cqlsh:employee> select * from Employee_Info;
```

```
 emp_id | date_of_joining                     | dept_name          | designation              | emp_name           | projects     |salary
---------+---------------------------------------+--------------------+-------------------------------+---------------+------------+--------
    1 | 2022-04-21 18:30:00.000000+0000 |       Management |                   CEO           |Bruce Wayne  |     null  |100000
    2 | 2022-04-23 18:30:00.000000+0000 |       Management |Senior Software Manager|Clark Kent     |     null  |70000
    4 | 2022-05-29 18:30:00.000000+0000 |       Development|         Senior Manager       | Aurthur Curry |     null  |70000
  121 | 2022-06-29 18:30:00.000000+0000 |       HR               |         Accountant              | Wally West    |     null  |60000
    3 | 2022-04-29 18:30:00.000000+0000 |       Development|    Jr Software Manager    | Diana Prince  |     null  |70000
```

7. Update the altered table to add project names.

```
cqlsh:employee> UPDATE Employee_Info SET Projects='Research' WHERE Emp_id=1 and
salary=100000.0;
cqlsh:employee> select * from Employee_Info;

cqlsh:employee> select * from Employee_Info;
```

```
 emp_id | date_of_joining                        | dept_name          | designation              | emp_name            | projects     |salary
---------+--------------------------------------+--------------------+-----------------------------+---------------+------------+--------
    1 | 2022-04-21 18:30:00.000000+0000 |       Management |                    CEO           |Bruce Wayne  | Research  |100000
    2 | 2022-04-23 18:30:00.000000+0000 |       Management |Senior Software Manager|Clark Kent     |     null  |70000
    4 | 2022-05-29 18:30:00.000000+0000 |       Development|         Senior Manager       | Aurthur Curry |     null  |70000
  121 | 2022-06-29 18:30:00.000000+0000 |       HR              |         Accountant              | Wally West    |     null  |60000
    3 | 2022-04-29 18:30:00.000000+0000 |       Development|    Jr Software Manager    | Diana Prince  |     null  |70000
```

```
cqlsh:employee> UPDATE Employee_Info SET Projects='Data Migration' WHERE Emp_id=2
and salary=70000.0;
cqlsh:employee> UPDATE Employee_Info SET Projects='Data analysis' WHERE Emp_id=3
and salary=70000.0;
cqlsh:employee> UPDATE Employee_Info SET Projects='Reporting' WHERE Emp_id=121 and
salary=60000.0;
cqlsh:employee> UPDATE Employee_Info SET Projects='Research' WHERE Emp_id=4 and
salary=70000.0;

cqlsh:employee> select * from Employee_Info;
```

```
 emp_id | date_of_joining                        | dept_name          | designation              | emp_name            | projects     |salary
---------+--------------------------------------+--------------------+-----------------------------+---------------+------------+--------
    1 | 2022-04-21 18:30:00.000000+0000 |       Management |                    CEO           |Bruce Wayne  | Research  |100000
    2 | 2022-04-23 18:30:00.000000+0000 |     Management |Senior Software Manager|Clark Kent     | Data Migration  |70000
    4 | 2022-05-29 18:30:00.000000+0000 |      Development|         Senior Manager       | Aurthur Curry |Data analysis  |70000
  121 | 2022-06-29 18:30:00.000000+0000 |       HR              |         Accountant              | Wally West    |Reporting  |60000
    3 | 2022-04-29 18:30:00.000000+0000 |       Development|    Jr Software Manager    | Diana Prince  |Research  |70000
```

8 Create a TTL of 15 seconds to display the values of Employees

```
cqlsh:employee> INSERT INTO Employee_Info(Emp_id, Emp_Name, Designation, Date_Of_Joining,
salary, Dept_name) VALUES (5,'John Jones','CTO','2022-04-01',80000.0,'Space Station') using ttl 15;

cqlsh:employee> select ttl(Emp_Name) from Employee_Info Where Emp_id=5;

 ttl(emp_name)
---------------
          6
```

Cassandra Program - 2

1 Create a key space by name Library

```
bmsce@bmsce-Precision-T1700:~$ Cassandra/apache-cassandra-3.11.0/bin
bash: Cassandra/apache-cassandra-3.11.0/bin: Is a directory
bmsce@bmsce-Precision-T1700:~$ Cassandra/apache-cassandra-3.11.0/bin/
bash: Cassandra/apache-cassandra-3.11.0/bin/: Is a directory
bmsce@bmsce-Precision-T1700:~$ cd Cassandra/apache-cassandra-3.11.0/bin/
bmsce@bmsce-Precision-T1700:~/Cassandra/apache-cassandra-3.11.0/bin$ ./cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> create keyspace library with replication = {
   ... 'class':'SimpleStrategy', 'replication_factor':1
   ... };
cqlsh> describe keyspaces

system_schema  system   student            system_traces
system_auth    library  system_distributed
```

2. Create a column family by name Library-Info with attributes Stud_Id Primary Key, Counter_value of type Counter, Stud_Name, Book-Name, Book-Id, Date_of_issue

```
cqlsh:library> create table library_info(stud_id int, counter_value counter, stud_name text, book_name
text, book_id int, date_of_issue date, primary key(stud_id, stud_name, book_name, book_id,
date_of_issue));

cqlsh:library> describe library_info

CREATE TABLE library.library_info (
    stud_id int,
    stud_name text,
    book_name text,
    book_id int,
    date_of_issue date,
    counter_value counter,
    PRIMARY KEY (stud_id, stud_name, book_name, book_id, date_of_issue)
) WITH CLUSTERING ORDER BY (stud_name ASC, book_name ASC, book_id ASC, date_of_issue
ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy',
'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

3. Insert the values into the table in batch
4. Display the details of the table created and increase the value of the counter

```
cqlsh:library> update library_info  set counter_value = counter_value + 1 where stud_id = 1 and stud_name = 'Bruce' and book_name = 'Game of Thrones' and book_id = 1 and date_of_issue = '2022-04-20';
cqlsh:library> select * from library_info;

 stud_id | stud_name | book_name    | book_id | date_of_issue | counter_value
-------------+------------------+----------------------+-------------+----------------------+---------------
      1     |    Bruce       | Game of Thrones |       1 |    2022-04-20 |            1

(1 rows)

cqlsh:library> update library_info  set counter_value = counter_value + 1 where stud_id = 2 and stud_name = 'Clark' and book_name = 'Song of Ice and Fire' and book_id = 2 and date_of_issue = '2022-04-21';
cqlsh:library> select * from library_info;

 stud_id | stud_name | book_name     | book_id | date_of_issue | counter_value
-------------+------------------+----------------------+-------------+----------------------+---------------
      1     |    Bruce       | Game of Thrones |       1 |    2022-04-20 |            1
       2    |    Clark        |Song of Ice and Fire |       2 |    2022-04-21 |            1

(2 rows)
cqlsh:library> update library_info  set counter_value = counter_value + 1 where stud_id = 112 and stud_name = 'Diana' and book_name = 'BDA' and book_id = 3 and date_of_issue = '2022-05-04';
cqlsh:library> select * from library_info;

 stud_id | stud_name | book_name     | book_id | date_of_issue | counter_value
-------------+------------------+----------------------+-------------+----------------------+---------------
      1     |    Bruce       | Game of Thrones |       1 |    2022-04-20 |            1
       2    |    Clark        |Song of Ice and Fire |       2 |    2022-04-21 |            1
     112  |    Diana                  |           BDA         |       3 |    2022-05-04 |            1

(3 rows)
```

5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.

```
cqlsh:library> update library_info  set counter_value = counter_value + 1 where stud_id = 112 and
stud_name = 'Diana' and book_name = 'BDA' and book_id = 3 and date_of_issue = '2022-05-04';
cqlsh:library> select * from library_info;

 stud_id | stud_name | book_name     | book_id | date_of_issue | counter_value
-------------+-----------------+---------------------+-----------+---------------------+--------------
      1    |    Bruce       | Game of Thrones |       1 |    2022-04-20 |            1
       2   |    Clark        |Song of Ice and Fire |       2 |    2022-04-21 |            1
     112  |    Diana                 |          BDA         |       3 |    2022-05-04 |            2

(3 rows)

cqlsh:library> select * from library_info where stud_id = 112;

 stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
-------------+-----------------+---------------------+-----------+---------------------+--------------
     112  |    Diana                 |          BDA         |       3 |    2022-05-04 |            2

(1 rows)
```

6. Export the created column to a csv file

```
cqlsh:library> copy library_info (stud_id, stud_name, book_name, book_id, date_of_issue,
counter_value) to '/home/bmsce/Desktop/data.csv';
Using 11 child processes

Starting copy of library.library_info with columns [stud_id, stud_name, book_name, book_id,
date_of_issue, counter_value].
Processed: 4 rows; Rate:      21 rows/s; Avg. rate:      21 rows/s
4 rows exported to 1 files in 0.200 seconds.
```

7. Import a given csv dataset from local file system into Cassandra column family

```
cqlsh:library> copy library_info (stud_id, stud_name, book_name, book_id, date_of_issue,
counter_value) from '/home/bmsce/Desktop/data1.csv';
Using 11 child processes

Starting copy of library.library_info with columns [stud_id, stud_name, book_name, book_id,
date_of_issue, counter_value].
Processed: 4 rows; Rate:       7 rows/s; Avg. rate:      11 rows/s
4 rows imported from 1 files in 0.381 seconds (0 skipped).

cqlsh:library> select * from library_info;
```

```
 stud_id | stud_name | book_name      | book_id | date_of_issue | counter_value
-------------+------------------+----------------------+-------------+----------------------+---------------
      1     |    Bruce       | Game of Thrones |       1 |    2022-04-20 |            1
       2    |    Clark        |Song of Ice and Fire |       2 |    2022-04-21 |            1
    112  |    Diana                   |           BDA          |       3 |    2022-05-04 |            2
      1     |    Bruce       | Game of Thrones |       1 |    2022-04-20 |            1
       2    |    Clark        |Song of Ice and Fire |       2 |    2022-04-21 |            1
    112  |    Diana                   |           BDA          |       3 |    2022-05-04 |            2
```