VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS (20CS6PEBDA)

Submitted by

Nikhil Abraham(1BM19CS101)

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

Nikhil Abraham (1BM19CS101), 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 -(20CS6PEBDA)**work prescribed for the said degree.

Name of the Lab-In charge **PALLAVI G B** Designation Assistant Professor Department of CSE Department of CSE BMSCE, Bengaluru BMSCE, Bengaluru
`

INDEX

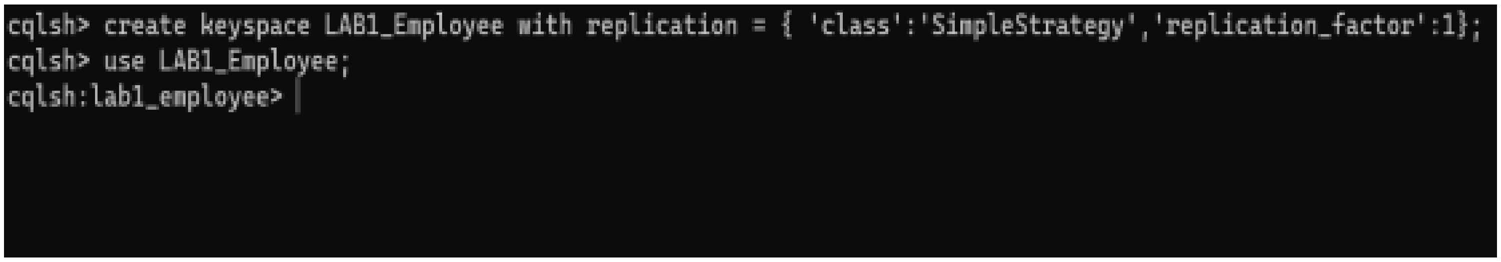
Sl. No.	Experiment Title	Page No.
1.	**Cassandra Lab Program 1: -Employee Database**	**4-6**
2.	**Cassandra Lab Program 2: -** Library Database	**7-8**
3.	**MongoDB Lab Program 1 (CRUD Demonstration): -** Students should be classifying a dataset into one of the standard forms and apply suitable querying rules to obtain suitable results	**9-20**

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

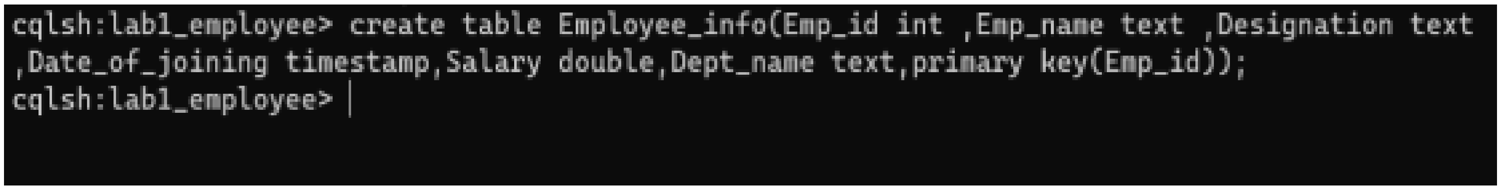
LAB 1

1.Create a key space by name Employee



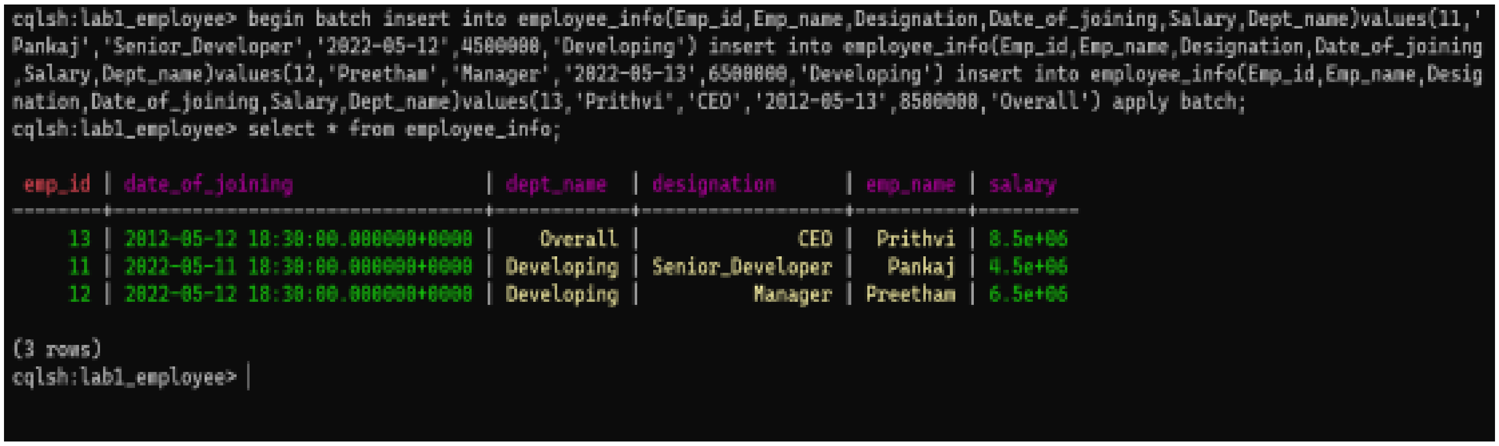
```
cqlsh> create keyspace LAB1_Employee with replication = { 'class':'SimpleStrategy','replication_factor':1};
cqlsh> use LAB1_Employee;
cqlsh:lab1_employee>
```

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



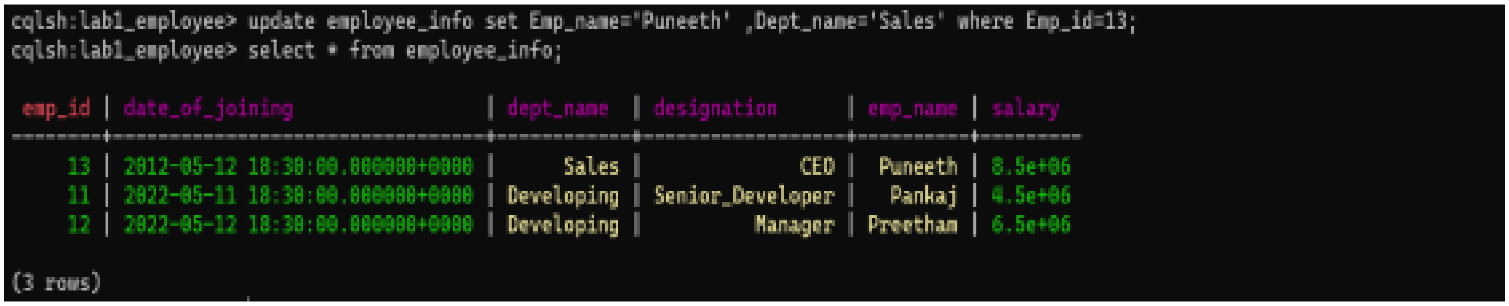
```
cqlsh:lab1_employee> create table Employee_info(Emp_id int ,Emp_name text ,Designation text
,Date_of_joining timestamp,Salary double,Dept_name text,primary key(Emp_id));
cqlsh:lab1_employee>
```

3. Insert the values into the table in batch



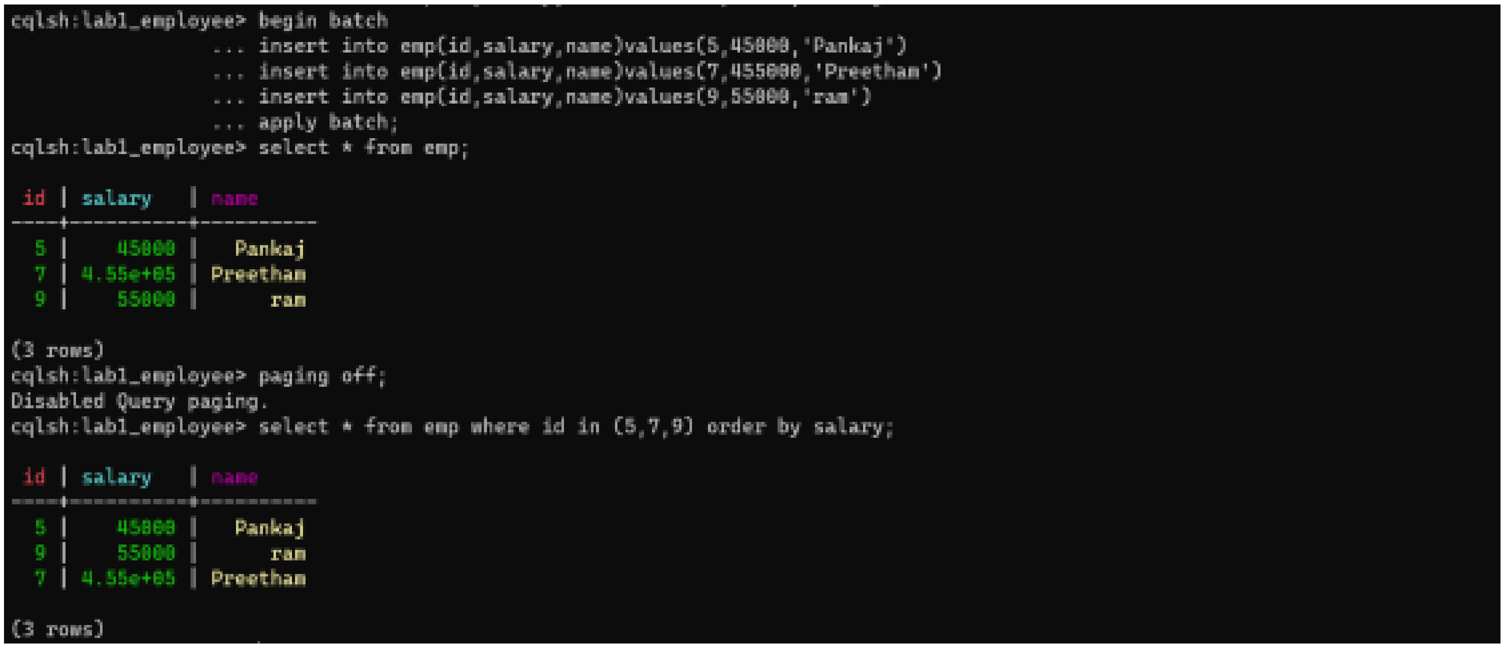
```
cqlsh:lab1_employee> begin batch insert into employee_info(Emp_id,Emp_name,Designation,Date_of_joining,Salary,Dept_name)values(11,'
Pankaj','Senior_Developer','2022-05-12',4500000,'Developing') insert into employee_info(Emp_id,Emp_name,Designation,Date_of_joining
,Salary,Dept_name)values(12,'Preetham','Manager','2022-05-13',6500000,'Developing') insert into employee_info(Emp_id,Emp_name,Desig
nation,Date_of_joining,Salary,Dept_name)values(13,'Prithvi','CEO','2012-05-13',8500000,'Overall') apply batch;
cqlsh:lab1_employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name  | designation      | emp_name | salary
--------+---------------------------------+------------+------------------+----------+---------
     13 | 2012-05-12 18:30:00.000000+0000 |    Overall |              CEO |  Prithvi | 8.5e+06
     11 | 2022-05-11 18:30:00.000000+0000 | Developing | Senior_Developer |   Pankaj | 4.5e+06
     12 | 2022-05-12 18:30:00.000000+0000 | Developing |          Manager | Preetham | 6.5e+06

(3 rows)
cqlsh:lab1_employee>
```

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



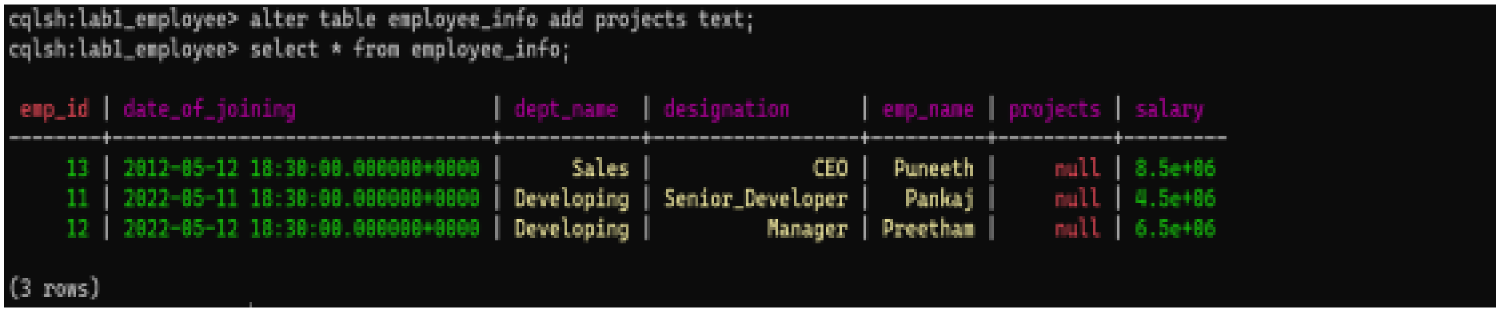
```
cqlsh:lab1_employee> update employee_info set Emp_name='Puneeth' ,Dept_name='Sales' where Emp_id=13;
cqlsh:lab1_employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name  | designation      | emp_name | salary
--------+---------------------------------+------------+------------------+----------+---------
     13 | 2012-05-12 18:30:00.000000+0000 |      Sales |              CEO |  Puneeth | 8.5e+06
     11 | 2022-05-11 18:30:00.000000+0000 | Developing | Senior_Developer |   Pankaj | 4.5e+06
     12 | 2022-05-12 18:30:00.000000+0000 | Developing |          Manager | Preetham | 6.5e+06

(3 rows)
```

5. Sort the details of Employee records based on salary



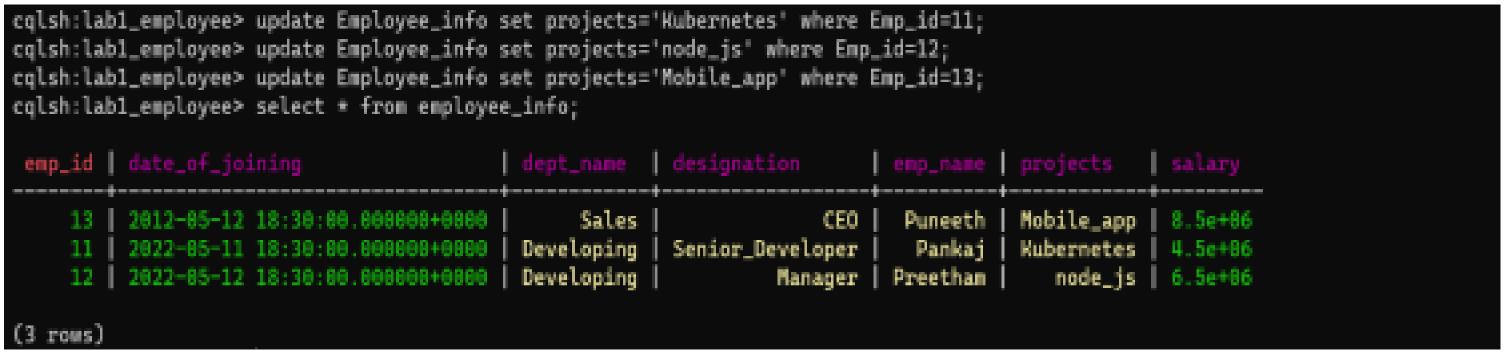
```
cqlsh:lab1_employee> begin batch
                 ... insert into emp(id,salary,name)values(5,45000,'Pankaj')
                 ... insert into emp(id,salary,name)values(7,455000,'Preetham')
                 ... insert into emp(id,salary,name)values(9,55000,'ram')
                 ... apply batch;
cqlsh:lab1_employee> select * from emp;

 id | salary    | name
----+-----------+----------
  5 |     45000 |   Pankaj
  7 | 4.55e+05  | Preetham
  9 |     55000 |      ram

(3 rows)
cqlsh:lab1_employee> paging off;
Disabled Query paging.
cqlsh:lab1_employee> select * from emp where id in (5,7,9) order by salary;

 id | salary    | name
----+-----------+----------
  5 |     45000 |   Pankaj
  9 |     55000 |      ram
  7 | 4.55e+05  | Preetham

(3 rows)
```

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.



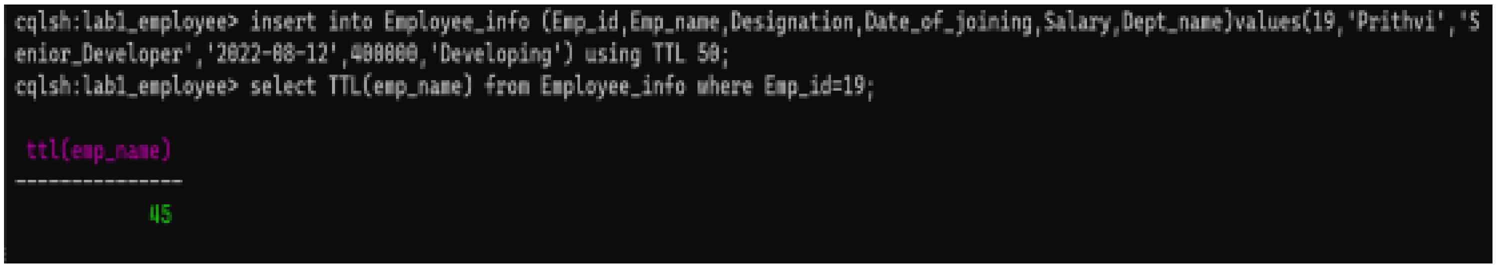
```
cqlsh:lab1_employee> alter table employee_info add projects text;
cqlsh:lab1_employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name  | designation      | emp_name | projects | salary
--------+---------------------------------+------------+------------------+----------+----------+---------
     13 | 2012-05-12 18:30:00.000000+0000 |      Sales |              CEO |  Puneeth |     null | 8.5e+06
     11 | 2022-05-11 18:30:00.000000+0000 | Developing | Senior_Developer |   Pankaj |     null | 4.5e+06
     12 | 2022-05-12 18:30:00.000000+0000 | Developing |          Manager | Preetham |     null | 6.5e+06

(3 rows)
```

7. Update the altered table to add project names.



```
cqlsh:lab1_employee> update Employee_info set projects='Kubernetes' where Emp_id=11;
cqlsh:lab1_employee> update Employee_info set projects='node_js' where Emp_id=12;
cqlsh:lab1_employee> update Employee_info set projects='Mobile_app' where Emp_id=13;
cqlsh:lab1_employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name  | designation      | emp_name | projects   | salary
--------+---------------------------------+------------+------------------+----------+------------+---------
     13 | 2012-05-12 18:30:00.000000+0000 |      Sales |              CEO |  Puneeth | Mobile_app | 8.5e+06
     11 | 2022-05-11 18:30:00.000000+0000 | Developing | Senior_Developer |   Pankaj | Kubernetes | 4.5e+06
     12 | 2022-05-12 18:30:00.000000+0000 | Developing |          Manager | Preetham |    node_js | 6.5e+06

(3 rows)
```

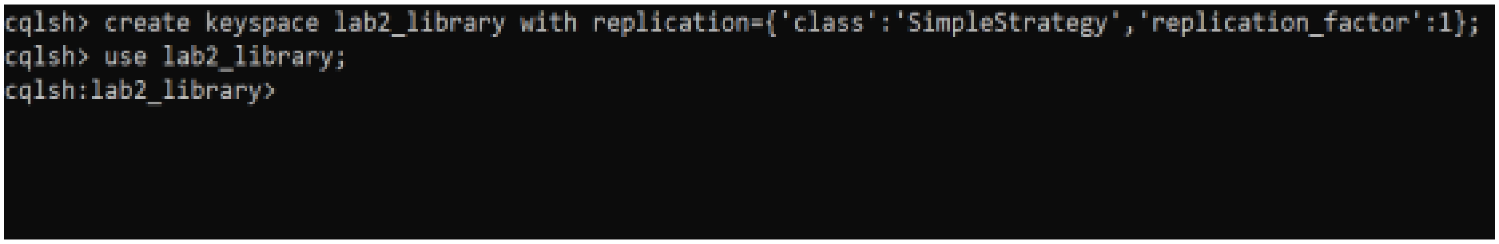
8 Create a TTL of 15 seconds to display the values of Employees.



```
cqlsh:lab1_employee> insert into Employee_info (Emp_id,Emp_name,Designation,Date_of_joining,Salary,Dept_name)values(19,'Prithvi','S
enior_Developer','2022-08-12',400000,'Developing') using TTL 50;
cqlsh:lab1_employee> select TTL(emp_name) from Employee_info where Emp_id=19;

 ttl(emp_name)
---------------
            45
```

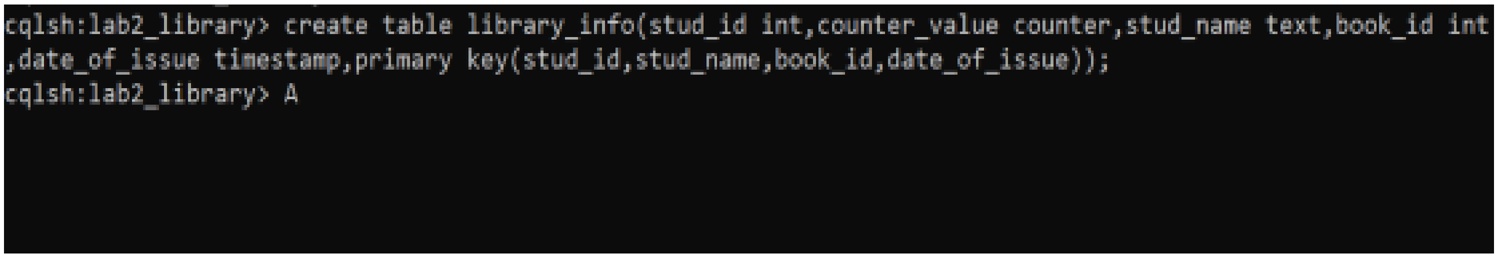
LAB 2

1 Create a key space by name Library



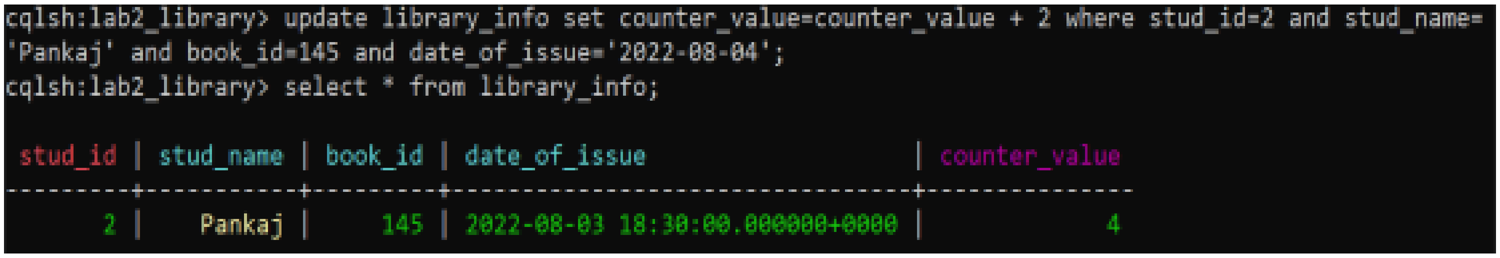
```
cqlsh> create keyspace lab2_library with replication={'class':'SimpleStrategy','replication_factor':1};
cqlsh> use lab2_library;
cqlsh:lab2_library>
```

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



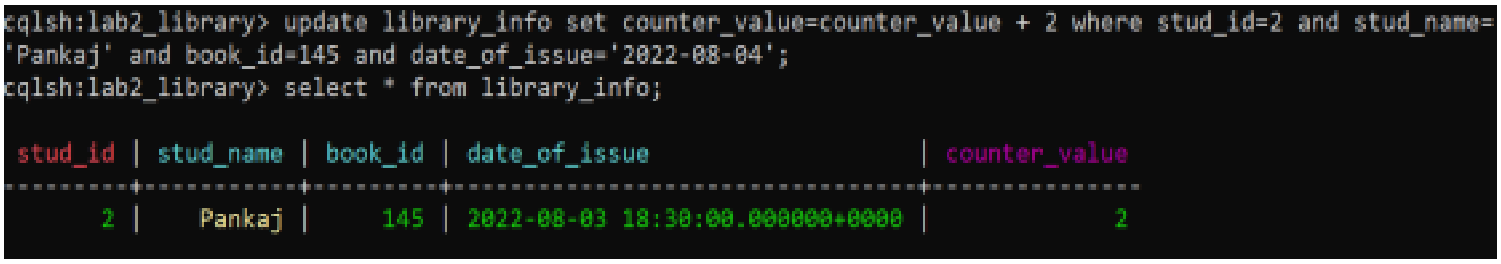
```
cqlsh:lab2_library> create table library_info(stud_id int,counter_value counter,stud_name text,book_id int
,date_of_issue timestamp,primary key(stud_id,stud_name,book_id,date_of_issue));
cqlsh:lab2_library> A
```

3. Insert the values into the table in batch



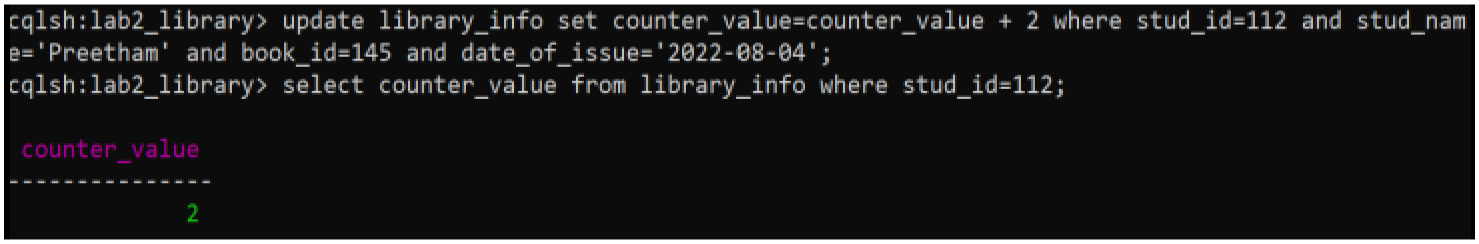
```
cqlsh:lab2_library> update library_info set counter_value=counter_value + 2 where stud_id=2 and stud_name=
'Pankaj' and book_id=145 and date_of_issue='2022-08-04';
cqlsh:lab2_library> select * from library_info;

 stud_id | stud_name | book_id | date_of_issue                   | counter_value
---------+-----------+---------+---------------------------------+---------------
       2 |    Pankaj |     145 | 2022-08-03 18:30:00.000000+0000 |             4
```

4. Display the details of the table created and increase the value of the counter



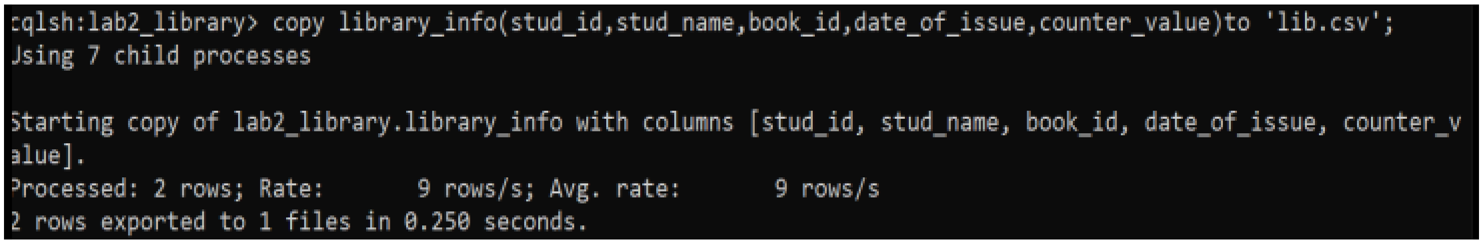
```
cqlsh:lab2_library> update library_info set counter_value=counter_value + 2 where stud_id=2 and stud_name=
'Pankaj' and book_id=145 and date_of_issue='2022-08-04';
cqlsh:lab2_library> select * from library_info;

 stud_id | stud_name | book_id | date_of_issue                   | counter_value
---------+-----------+---------+---------------------------------+---------------
       2 |    Pankaj |     145 | 2022-08-03 18:30:00.000000+0000 |             2
```

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

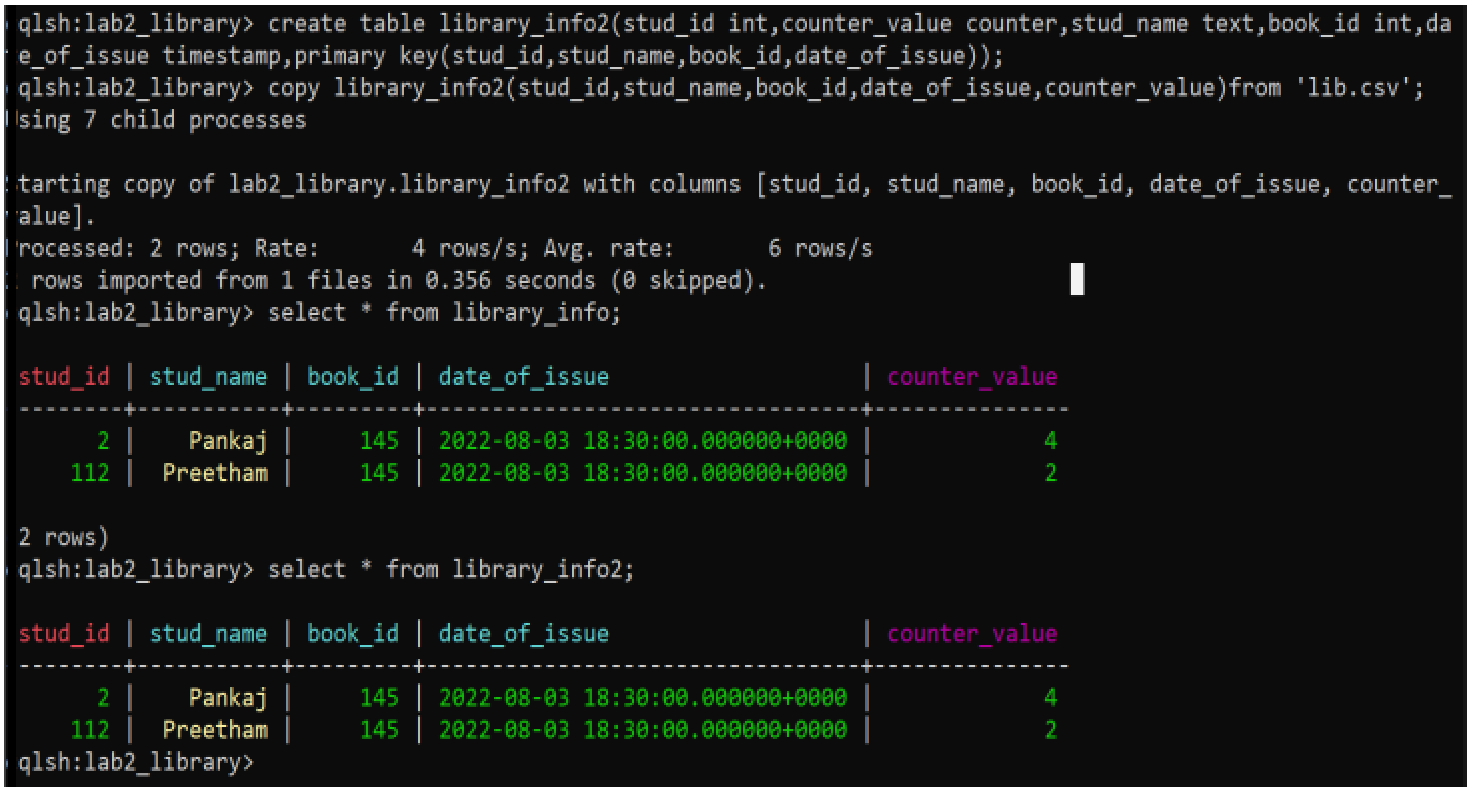


```
cqlsh:lab2_library> update library_info set counter_value=counter_value + 2 where stud_id=112 and stud_nam
e='Preetham' and book_id=145 and date_of_issue='2022-08-04';
cqlsh:lab2_library> select counter_value from library_info where stud_id=112;

 counter_value
---------------
             2
```

6. Export the created column to a csv file


```
cqlsh:lab2_library> copy library_info(stud_id,stud_name,book_id,date_of_issue,counter_value)to 'lib.csv';
Jsing 7 child processes

Starting copy of lab2_library.library_info with columns [stud_id, stud_name, book_id, date_of_issue, counter_v
alue].
Processed: 2 rows; Rate:       9 rows/s; Avg. rate:       9 rows/s
2 rows exported to 1 files in 0.250 seconds.
```

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


```
qlsh:lab2_library> create table library_info2(stud_id int,counter_value counter,stud_name text,book_id int,da
e_of_issue timestamp,primary key(stud_id,stud_name,book_id,date_of_issue));
qlsh:lab2_library> copy library_info2(stud_id,stud_name,book_id,date_of_issue,counter_value)from 'lib.csv';
sing 7 child processes

tarting copy of lab2_library.library_info2 with columns [stud_id, stud_name, book_id, date_of_issue, counter_
alue].
rocessed: 2 rows; Rate:       4 rows/s; Avg. rate:       6 rows/s
 rows imported from 1 files in 0.356 seconds (0 skipped).
qlsh:lab2_library> select * from library_info;

 stud_id | stud_name | book_id | date_of_issue                   | counter_value
---------+-----------+---------+---------------------------------+---------------
       2 |    Pankaj |     145 | 2022-08-03 18:30:00.000000+0000 |             4
     112 |  Preetham |     145 | 2022-08-03 18:30:00.000000+0000 |             2

2 rows)
qlsh:lab2_library> select * from library_info2;

 stud_id | stud_name | book_id | date_of_issue                   | counter_value
---------+-----------+---------+---------------------------------+---------------
       2 |    Pankaj |     145 | 2022-08-03 18:30:00.000000+0000 |             4
     112 |  Preetham |     145 | 2022-08-03 18:30:00.000000+0000 |             2
qlsh:lab2_library>
```

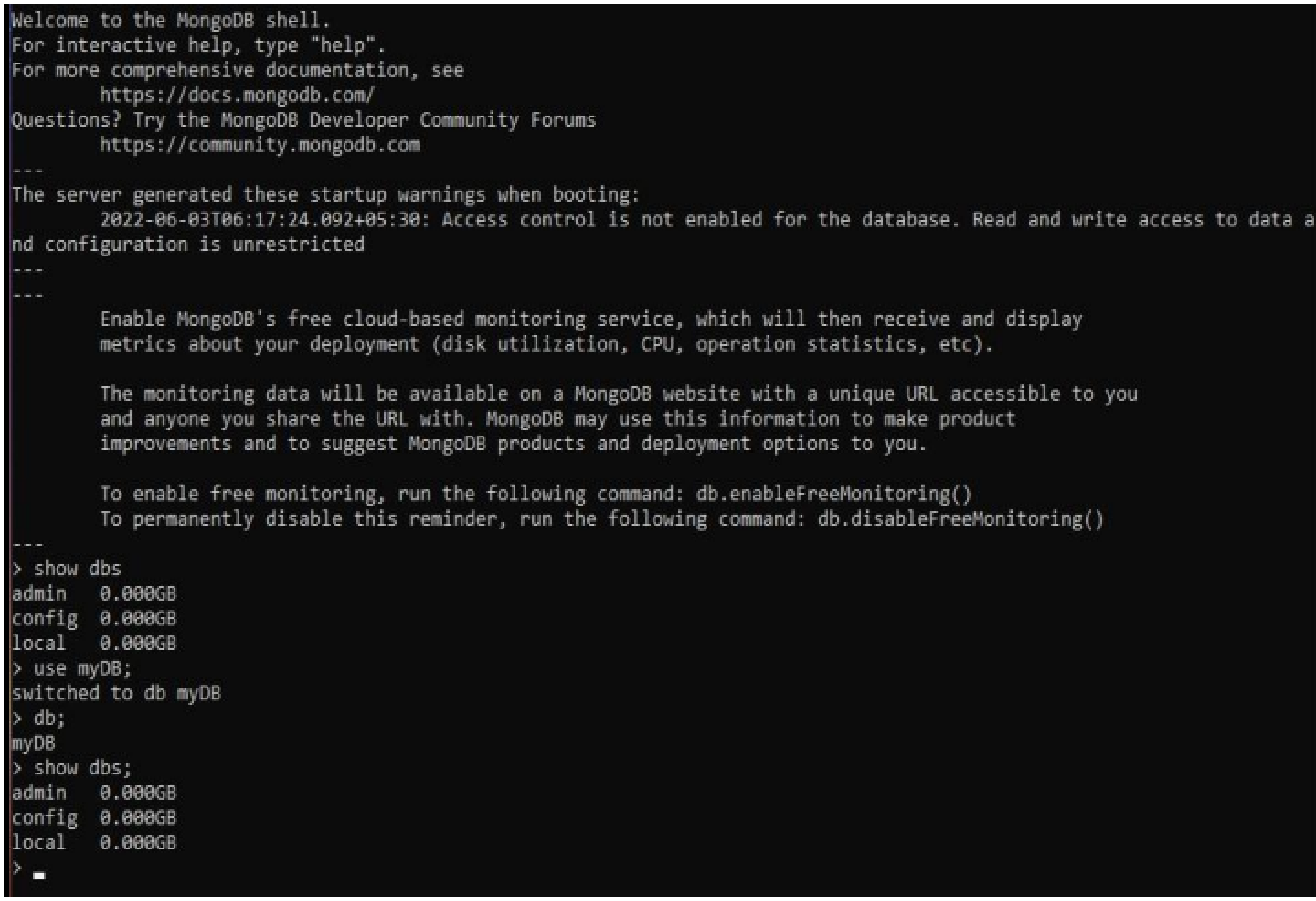
LAB 3

I. CREATE DATABASE IN MONGODB.

use myDB; db; (Confirm the

existence of your database)

show dbs; (To list all databases)



```
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2022-06-03T06:17:24.092+05:30: Access control is not enabled for the database. Read and write access to data a
nd configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use myDB;
switched to db myDB
> db;
myDB
> show dbs;
admin   0.000GB
config  0.000GB
local   0.000GB
> _
```

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();

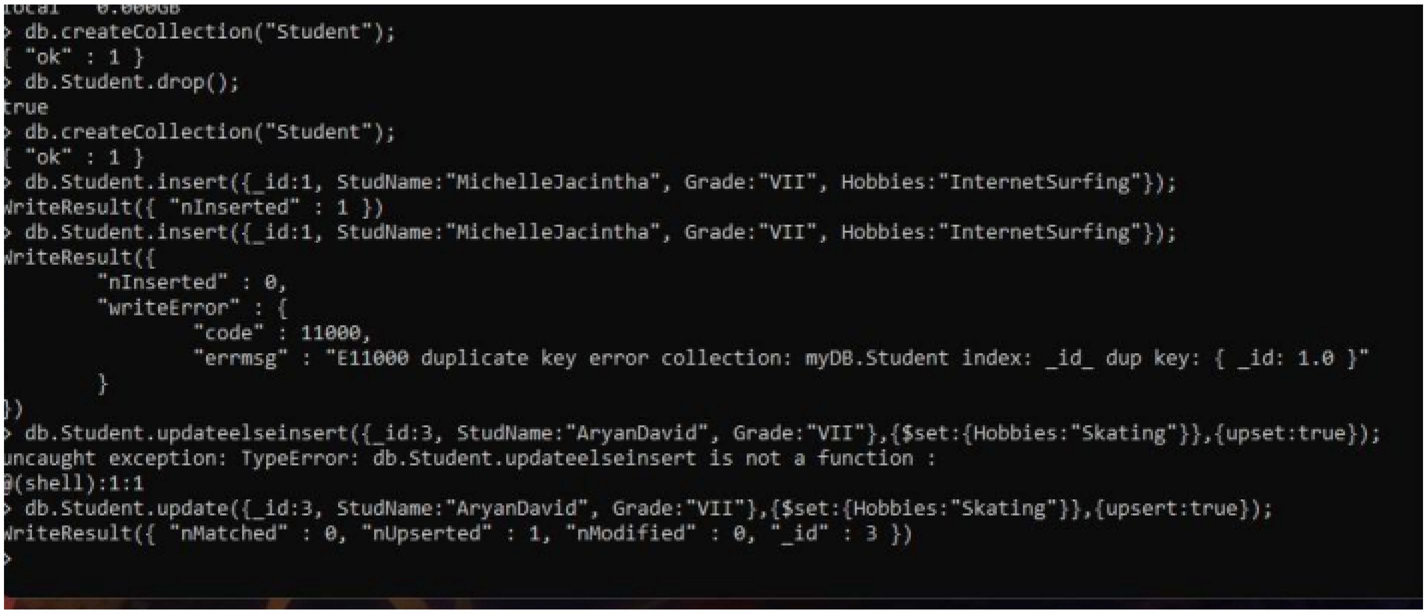
3. Create a collection by the name “Students” and store the following data in it.

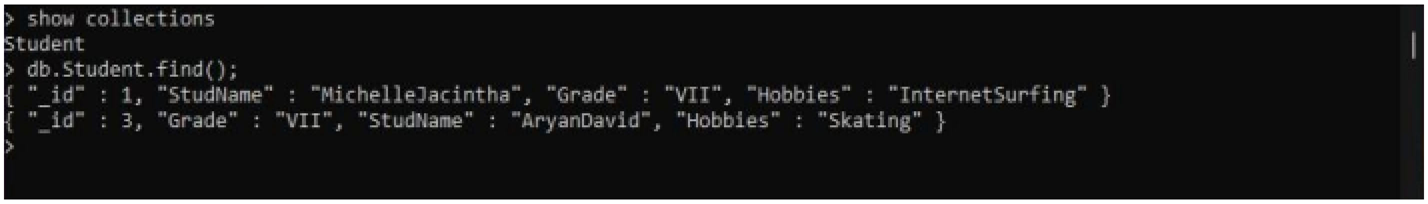
db.Student.insert({_id:1,StudName:&quot;MichelleJacintha&quot;,Grade:&quot;VII&quot;,Hobbies:& quot;Int ernetS urfing&quot;});

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:{Hobbie s:&quo t;Skatin g&quot;}},{upsert:true});


```
local   0.000GB
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.drop();
true
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.insert({_id:1, StudName:"MichelleJacintha", Grade:"VII", Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:1, StudName:"MichelleJacintha", Grade:"VII", Hobbies:"InternetSurfing"});
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 11000,
                "errmsg" : "E11000 duplicate key error collection: myDB.Student index: _id_ dup key: { _id: 1.0 }"
        }
})
> db.Student.updateelseinsert({_id:3, StudName:"AryanDavid", Grade:"VII"},{$set:{Hobbies:"Skating"}},{upset:true});
uncaught exception: TypeError: db.Student.updateelseinsert is not a function :
@(shell):1:1
> db.Student.update({_id:3, StudName:"AryanDavid", Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
>
```

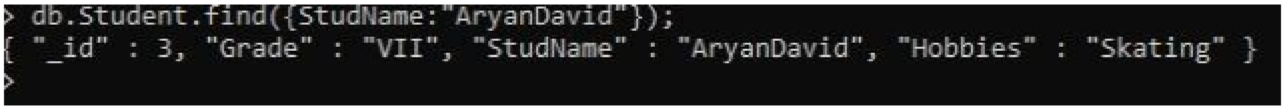

```
> show collections
Student
> db.Student.find();
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
>
```

5. FIND METHOD

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

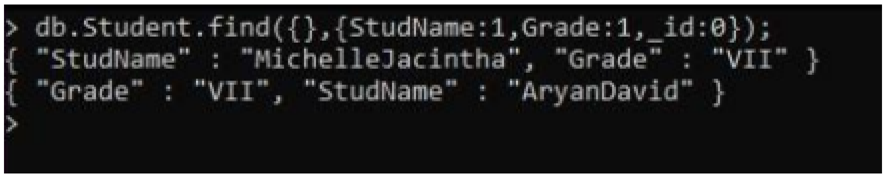
db.Student.find({StudName:&quot;Aryan David&quot;});

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


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

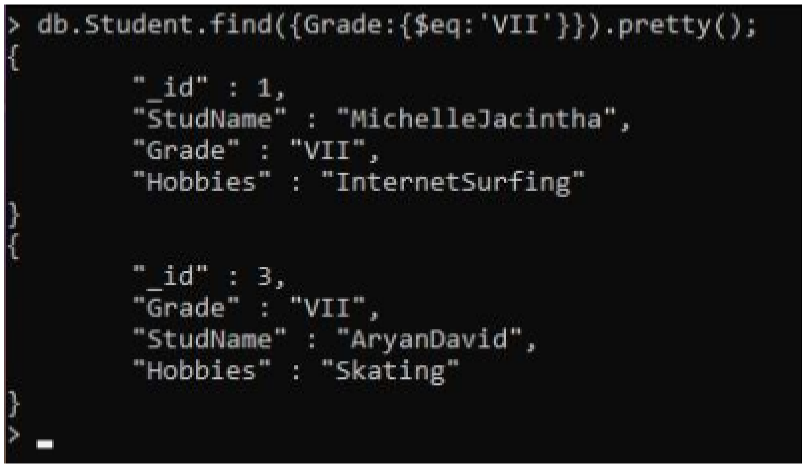
B. To display only the StudName and Grade from all the documents of the Students collection. The identifier_id should be suppressed and NOT displayed.

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


```
> db.Student.find({},{StudName:1,Grade:1,_id:0});
{ "StudName" : "MichelleJacintha", "Grade" : "VII" }
{ "Grade" : "VII", "StudName" : "AryanDavid" }
>
```

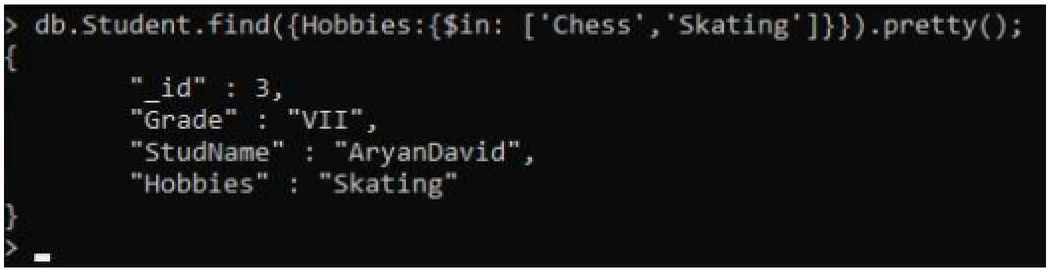
C. To find those documents where the Grade is set to ‘VII’

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



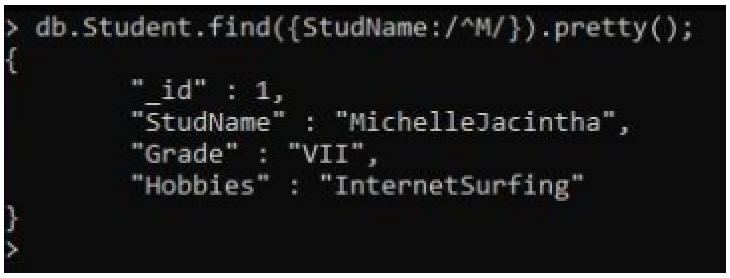
```
> db.Student.find({Grade:{$eq:'VII'}}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
> _
```

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: [&#39;Chess&#39;,&#39;Skating&#39;]}}).pretty ();



```
> db.Student.find({Hobbies:{$in: ['Chess','Skating']}}).pretty();
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
> _
```

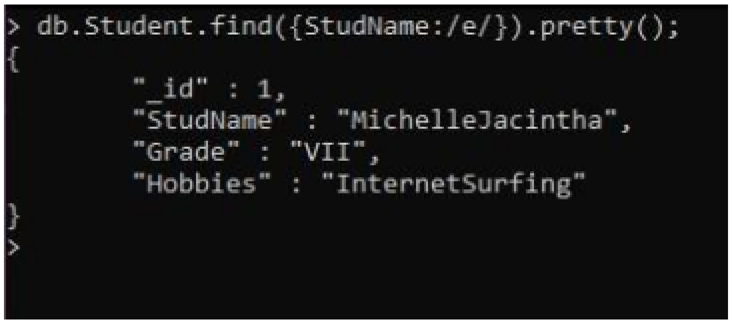
E. To find documents from the Students collection where the StudName begins with “M”. db.Student.find({StudName:/^M/}).pretty();



```
> db.Student.find({StudName:/^M/}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
>
```

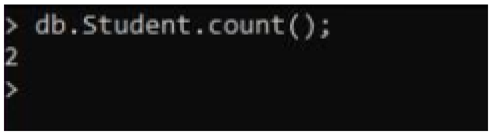
F. To find documents from the Students collection where the StudNamehas an “e” in any position.

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

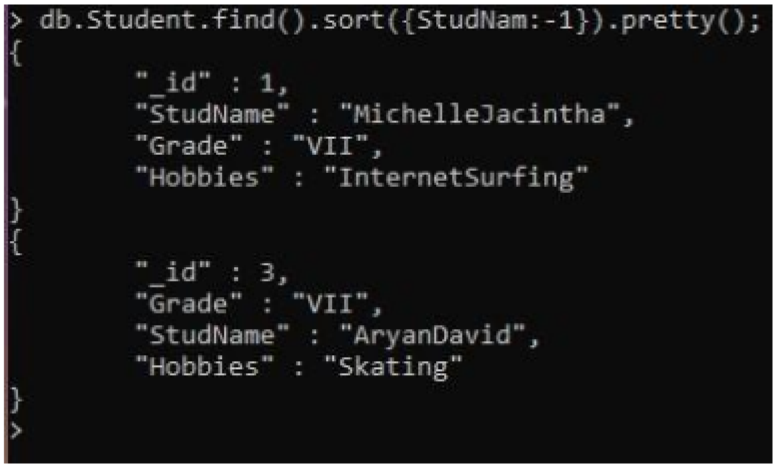

```
> db.Student.find({StudName:/e/}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
>
```

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

db.Student.count();


```
> db.Student.count();
2
>
```

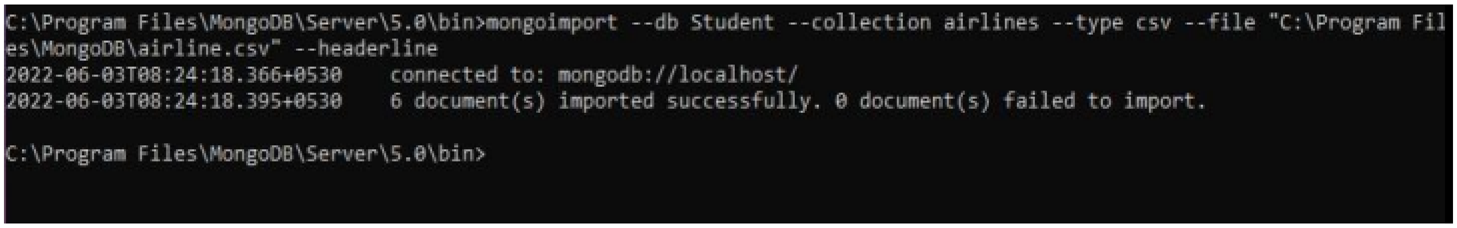
H. To sort the documents from the Students collection in the descending order of StudName. db.Student.find().sort({StudName:-1}).pretty();


```
> db.Student.find().sort({StudNam:-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


```
C:\Program Files\MongoDB\Server\5.0\bin>mongoimport --db Student --collection airlines --type csv --file "C:\Program Fil
es\MongoDB\airline.csv" --headerline
2022-06-03T08:24:18.366+0530    connected to: mongodb://localhost/
2022-06-03T08:24:18.395+0530    6 document(s) imported successfully. 0 document(s) failed to import.

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

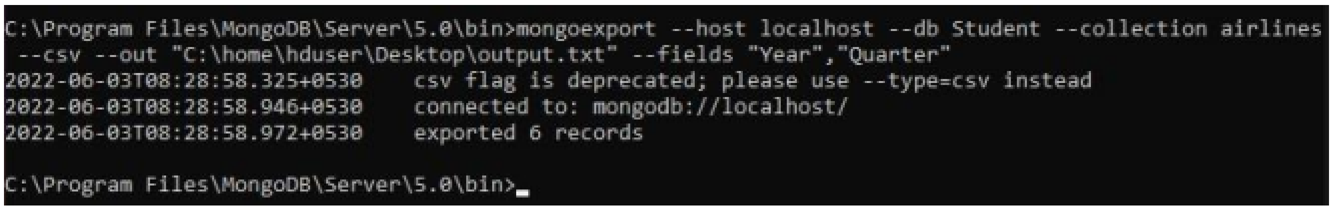
IV. Export data to a CSV file

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”

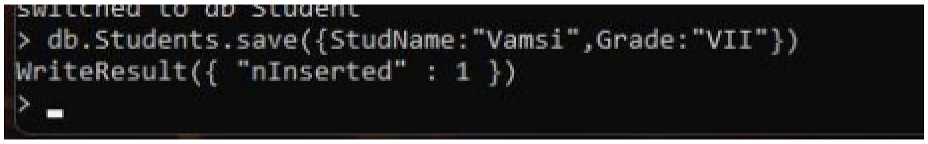

```
C:\Program Files\MongoDB\Server\5.0\bin>mongoexport --host localhost --db Student --collection airlines
 --csv --out "C:\home\hduser\Desktop\output.txt" --fields "Year","Quarter"
2022-06-03T08:28:58.325+0530    csv flag is deprecated; please use --type=csv instead
2022-06-03T08:28:58.946+0530   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

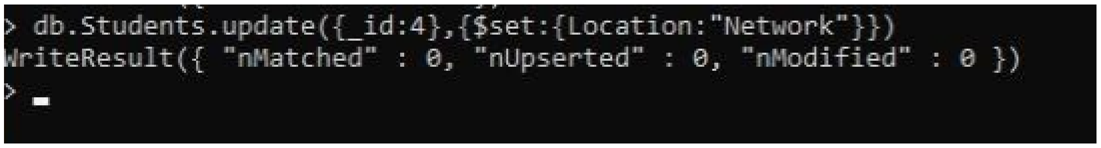
not exist. If it exists it will replace the exisiting document.

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

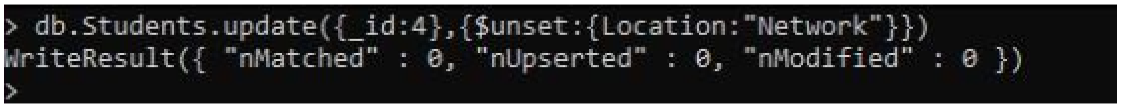

```
switched to db Student
> db.Students.save({StudName:"Vamsi",Grade:"VII"})
WriteResult({ "nInserted" : 1 })
> 
```

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"}})



```
> 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

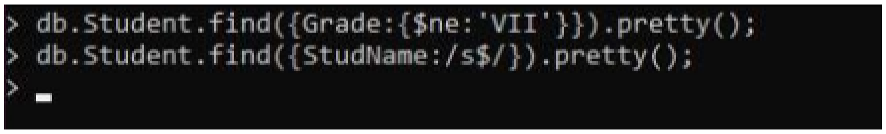
db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});

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

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

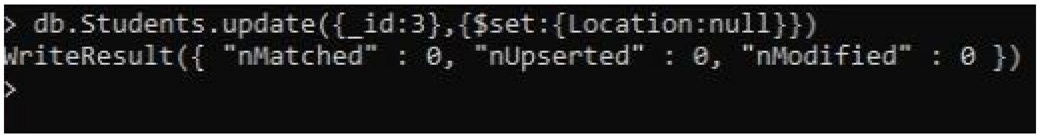
To find documents from the Students collection where the StudName ends with s.

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



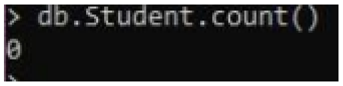
```
> db.Student.find({Grade:{$ne:'VII'}}).pretty();
> db.Student.find({StudName:/s$/}).pretty();
> _
```

IX. to set a particular field value to NULL



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

X Count the number of documents in Student Collections



```
> db.Student.count()
0
>
```

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

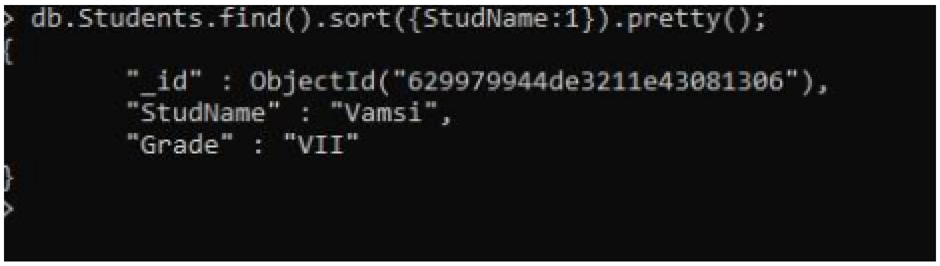
document in Ascending order

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()

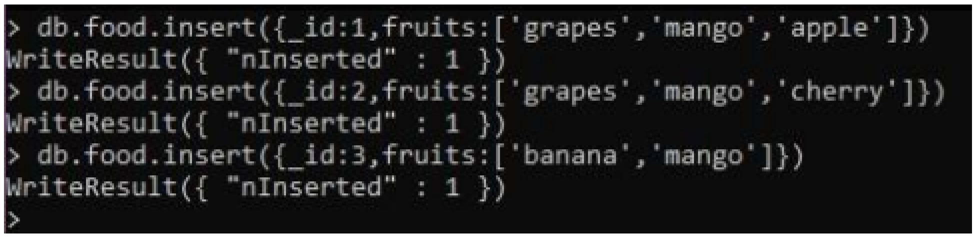


```
> db.Students.find().sort({StudName:1}).pretty();
{
        "_id" : ObjectId("629979944de3211e43081306"),
        "StudName" : "Vamsi",
        "Grade" : "VII"
}
>
```

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;] })

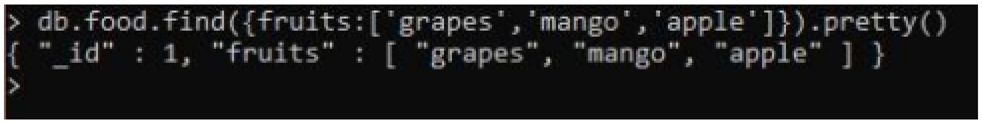
db.food.insert({ _id:2, fruits:[&#39;grapes&#39;,&#39;mango&#39;,&#39;cherry&#39;] })

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



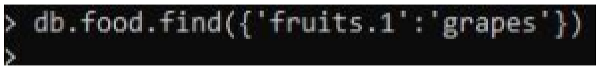
```
> db.food.insert({_id:1,fruits:['grapes','mango','apple']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:2,fruits:['grapes','mango','cherry']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:3,fruits:['banana','mango']})
WriteResult({ "nInserted" : 1 })
>
```

To find those documents from the “food” collection 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().



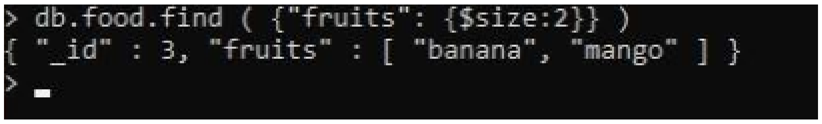
```
> db.food.find({fruits:['grapes','mango','apple']}).pretty()
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
>
```

To find in “fruits” array having “mango” in the first index position. db.food.find ({&#39;fruits.1&#39;:&#39;grapes&#39;})



```
> db.food.find({'fruits.1':'grapes'})
>
```

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

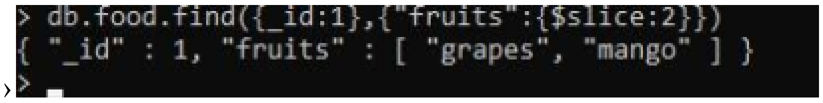


```
> db.food.find ( {"fruits": {$size:2}} )
{ "_id" : 3, "fruits" : [ "banana", "mango" ] }
> ▬
```

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}})

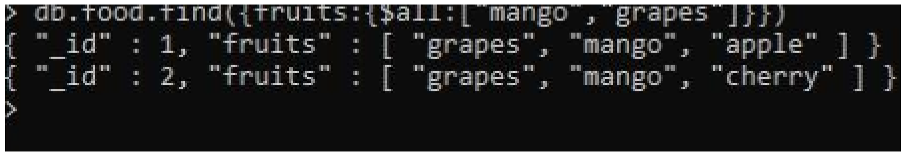


```
> db.food.find({_id:1},{"fruits":{$slice:2}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
> _
```

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

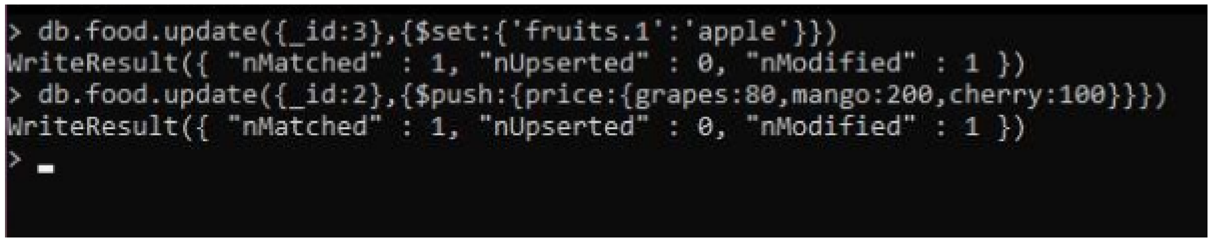
index position of the fruits array with apple

db.food.update({_id:3},{$set:{&#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

}}})



```
> 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 })
> _
```

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 :

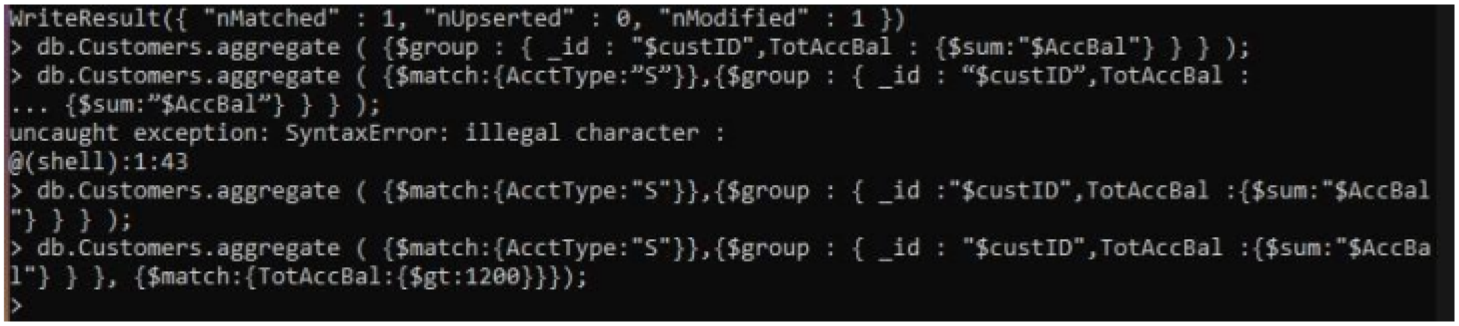
{$sum:”$AccBal”} } });

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}}});



```
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}}});
>
```