VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS (20CS6PEBDA)

Submitted by

NOOR FATHIMA ARFA (1BM19CS108)

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 **NOOR FATHIMA ARFA (1BM19CS108),** 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 **ANTARA ROY CHOUDHURY** Designation Assistant Professor Department of CSE Department of CSE BMSCE, Bengaluru BMSCE, Bengaluru
`

\

Index Sheet

Sl. No.	Experiment Title	PageNo.
1.	Mongo CRUD Demonstration	
2.	Cassandra Employee Keyspace	
3.	Casssandra Library Keyspace	

Course Outcome

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

LAB -2

Perform the following DB operations using Cassandra.

1. Create a keyspace by name Employee

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

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

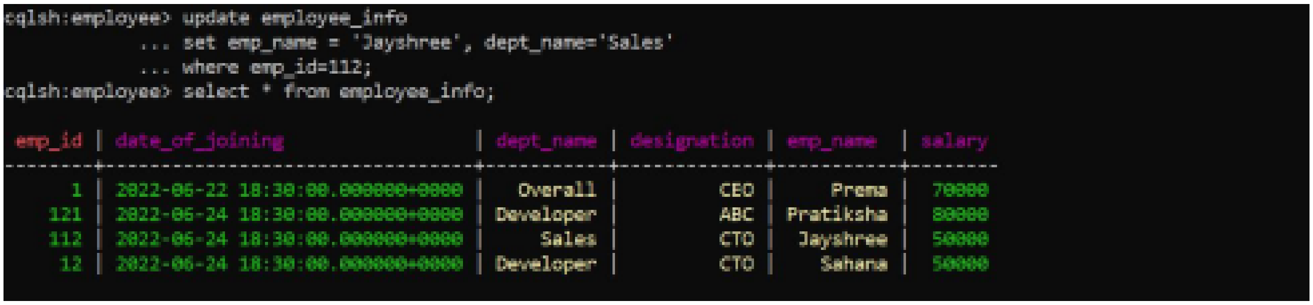
```
create table employee_info( emp_id int, emp_name text, designation text, date_of_joining
timestamp, salary double, dept_name text, PRIMARY KEY(emp_id));
```

3. Insert the values into the table in batch

```
begin batch insert into
employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)values
(1,'Prema','CEO','2022-06-23',70000,'Overall') insert into
employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)values
(12,'Sahana','CTO','2022-06-25',50000,'Developer') insert into
employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values
(121,'Pratiksha','ABC','2022-06-25',80000,'Developer') insert into
employee_info(emp_id,emp_name,designa
tion,date_of_joining,salary,dept_name)values (112,'Pooja','CTO','2022-06-
25',50000,'Developer') apply batch ;
cqlsh:employee> select * from employee_info;
```

```
 emp_id | date_of_joining | dept_name | designation |
emp_name|salary--------+--------------------------------+-----------+-------------+-----------+-------- 1 |
2022-06-22 18:30:00.000000+| Overall | CEO | Prema | 70000121 | 2022-06-24
18:30:00.000000+ | Developer | ABC | Pratiksha | 80000112 | 2022-06-24 18:30:00.000000+|
Developer | CTO | Pooja | 5000012 | 2022-06-24 18:30:00.000000+ | Developer | CTO |
Sahana | 50000
```

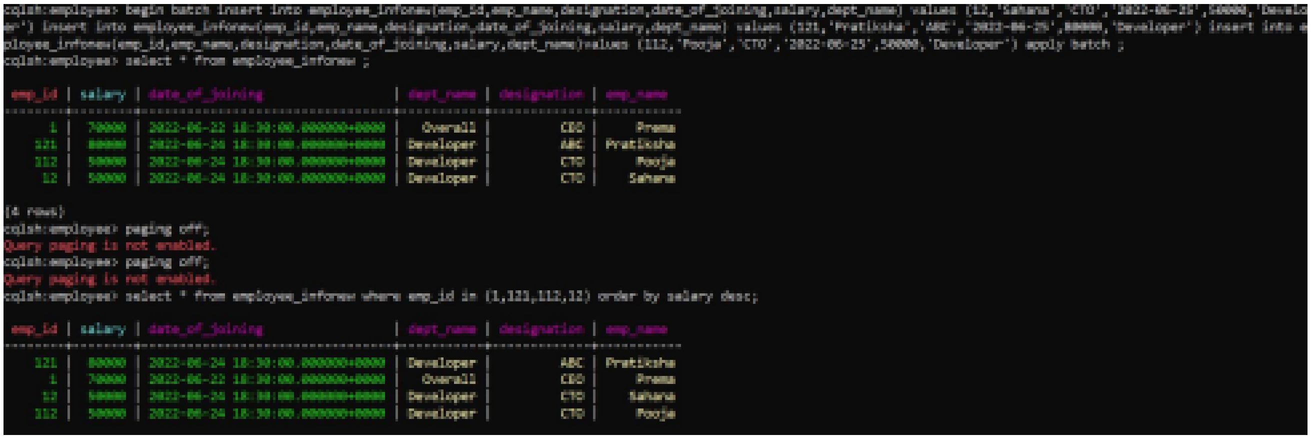
4. Update Employee name and Department of Emp-Id 121



```
cqlsh:employee> update employee_info
            ... set emp_name = 'Jayshree', dept_name='Sales'
            ... where emp_id=112;
cqlsh:employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name | designation | emp_name  | salary
--------+---------------------------------+-----------+-------------+-----------+--------
      1 | 2022-06-22 18:30:00.000000+0000 |   Overall |         CEO |     Prema |  70000
    121 | 2022-06-24 18:30:00.000000+0000 | Developer |         ABC | Pratiksha |  80000
    112 | 2022-06-24 18:30:00.000000+0000 |     Sales |         CTO |  Jayshree |  50000
     12 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |    Sahana |  50000
```

5.

Sort the details of Employee records based on salary

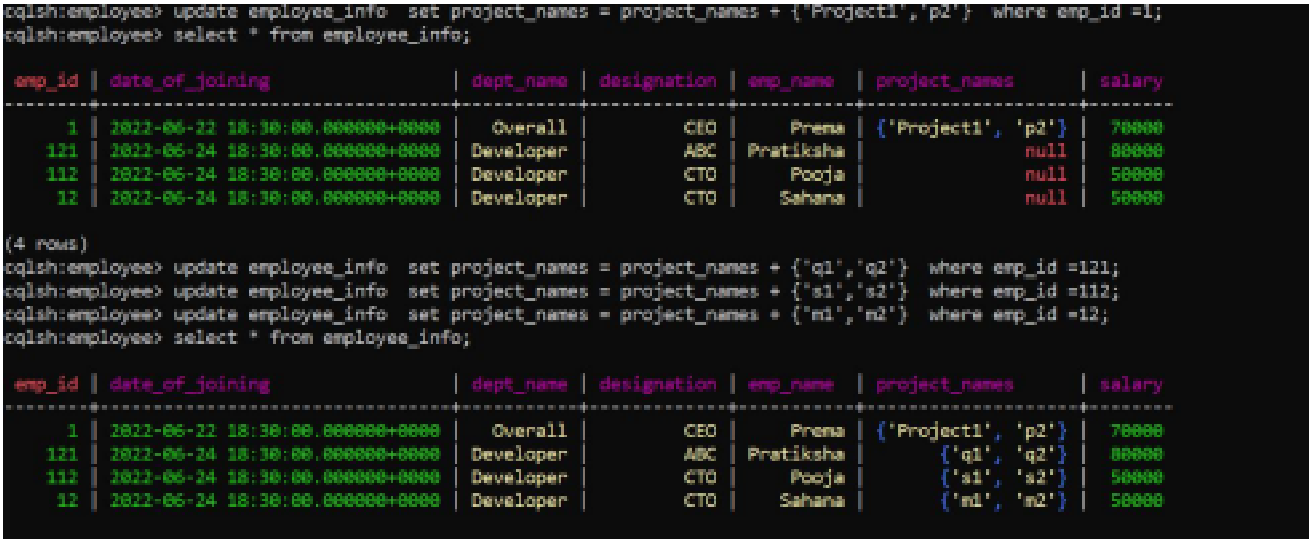


```
cqlsh:employee> begin batch insert into employee_infonew(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (12,'Sahana','CTO','2022-06-25',50000,'Develo
er') insert into employee_infonew(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (121,'Pratiksha','ABC','2022-06-25',80000,'Developer') insert into e
ployee_infonew(emp_id,emp_name,designation,date_of_joining,salary,dept_name)values (112,'Pooja','CTO','2022-06-25',50000,'Developer') apply batch ;
cqlsh:employee> select * from employee_infonew ;

 emp_id | salary | date_of_joining                 | dept_name | designation | emp_name
--------+--------+---------------------------------+-----------+-------------+-----------
      1 |  70000 | 2022-06-22 18:30:00.000000+0000 |   Overall |         CEO |     Prema
    121 |  80000 | 2022-06-24 18:30:00.000000+0000 | Developer |         ABC | Pratiksha
    112 |  50000 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |     Pooja
     12 |  50000 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |    Sahana

(4 rows)
cqlsh:employee> paging off;
Query paging is not enabled.
cqlsh:employee> paging off;
Query paging is not enabled.
cqlsh:employee> select * from employee_infonew where emp_id in (1,121,112,12) order by salary desc;

 emp_id | salary | date_of_joining                 | dept_name | designation | emp_name
--------+--------+---------------------------------+-----------+-------------+-----------
    121 |  80000 | 2022-06-24 18:30:00.000000+0000 | Developer |         ABC | Pratiksha
      1 |  70000 | 2022-06-22 18:30:00.000000+0000 |   Overall |         CEO |     Prema
     12 |  50000 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |    Sahana
    112 |  50000 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |     Pooja
```

6.

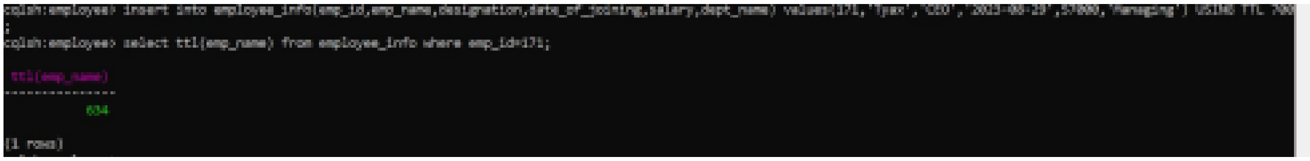
Alter the schema of the table Employee_Info to add a column Projects whichstoresa set of Projects done by the corresponding Employee.

alter table employee_info add project_names set<text>; 7. Update the altered table to add project names.



```
cqlsh:employee> update employee_info  set project_names = project_names + {'Project1','p2'}  where emp_id =1;
cqlsh:employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name | designation | emp_name  | project_names       | salary
--------+---------------------------------+-----------+-------------+-----------+---------------------+--------
      1 | 2022-06-22 18:30:00.000000+0000 |   Overall |         CEO |     Prema | {'Project1', 'p2'}  |  70000
    121 | 2022-06-24 18:30:00.000000+0000 | Developer |         ABC | Pratiksha |                null |  80000
    112 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |     Pooja |                null |  50000
     12 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |    Sahana |                null |  50000

(4 rows)
cqlsh:employee> update employee_info  set project_names = project_names + {'q1','q2'}  where emp_id =121;
cqlsh:employee> update employee_info  set project_names = project_names + {'s1','s2'}  where emp_id =112;
cqlsh:employee> update employee_info  set project_names = project_names + {'m1','m2'}  where emp_id =12;
cqlsh:employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name | designation | emp_name  | project_names       | salary
--------+---------------------------------+-----------+-------------+-----------+---------------------+--------
      1 | 2022-06-22 18:30:00.000000+0000 |   Overall |         CEO |     Prema | {'Project1', 'p2'}  |  70000
    121 | 2022-06-24 18:30:00.000000+0000 | Developer |         ABC | Pratiksha |        {'q1', 'q2'} |  80000
    112 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |     Pooja |        {'s1', 's2'} |  50000
     12 | 2022-06-24 18:30:00.000000+0000 | Developer |         CTO |    Sahana |        {'m1', 'm2'} |  50000
```

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



```
cqlsh:employee> insert into employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values(171,'Tyax','CEO','2023-05-29',57000,'Managing') USING TTL 700
;
cqlsh:employee> select ttl(emp_name) from employee_info where emp_id=171;

 ttl(emp_name)
---------------
           634

(1 rows)
```

LAB -3

3. Perform the following DB operations using Cassandra.

1.Create a keyspace by name Library

```
CREATE KEYSPACE LIBRARY1 WITH REPLICATION = { ... 'class':'SimpleStrategy',
   ... 'replication_factor':1};
```

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

```
create table library_info( stud_id int, counter_value counter, stud_name text, book_name text,
book_id int, date_of_issue timestamp,PRIMARY
KEY(stud_id,stud_name,book_name,book_id,date_of_issue));
```

3.Insert the values into the table in batch

```
update library_info
          ... set counter_value = counter_value +1 where stud_id=121 and stud_name='Prema'
and book_name='cns' and book_id=113 and date_of_issue='2022-06-29';
select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+----------+---------------------------------+--------------- 121 | Prema | cns |
    113 | 2022-06-28 18:30:00.000000+0000 | 1
```

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

```
 update library_info set counter_value = counter_value +1 where
stud_id=121andstud_name='Prema' and book_name='cns' and book_id=113 and
date_of_issue='2022-06-29';
cqlsh:library1> select * from library_info;
 stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+----------+---------------------------------+--------------- 121 | Prema | cns |
    113 | 2022-06-28 18:30:00.000000+0000 | 2
```

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

```
cqlsh:library1> update library_info set counter_value = counter_value +2wherestud_id=111
```

```
and stud_name='Pooja' and book_name='bda' and book_id=112anddate_of_issue='202
2-06-29';
select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+---------+-------------------------------+--------------- 111 | Pooja | bda |
    112 | 2022-06-28 18:30:00.000000+0000 | 2121 | Prema | cns | 113 | 2022-06-28
    18:30:00.000000+0000 | 2
```

6. Export the created column to a csv file

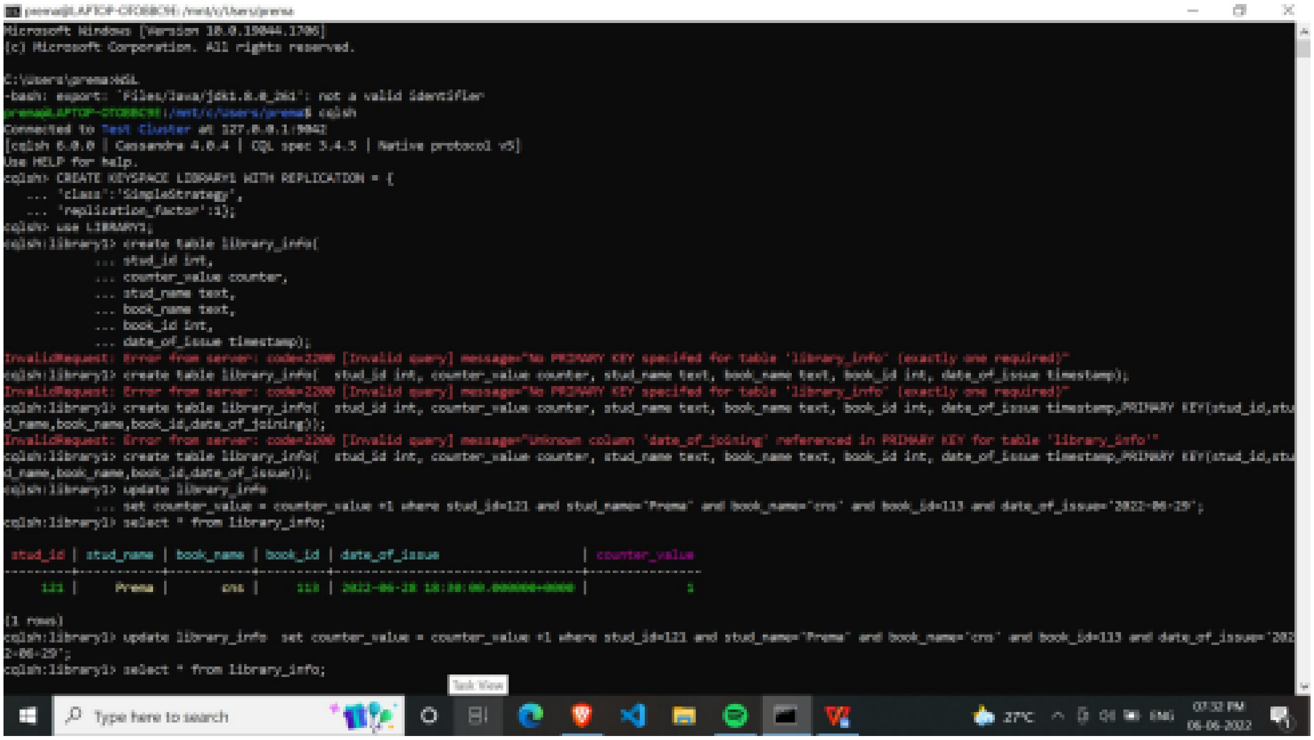
```
COPY
library_info(stud_id,counter_value,stud_name,book_name,book_id,date_of_issue
)TO 'lib1.csv'
         ... ;
Using 7 child processes

Starting copy of library1.library_info with columns [stud_id, counter_value,
stud_name, book_name, book_id, date_of_issue].
Processed: 2 rows; Rate: 17 rows/s; Avg. rate: 17 rows/s 2 rows exported to 1
files in 0.143 seconds.
```

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

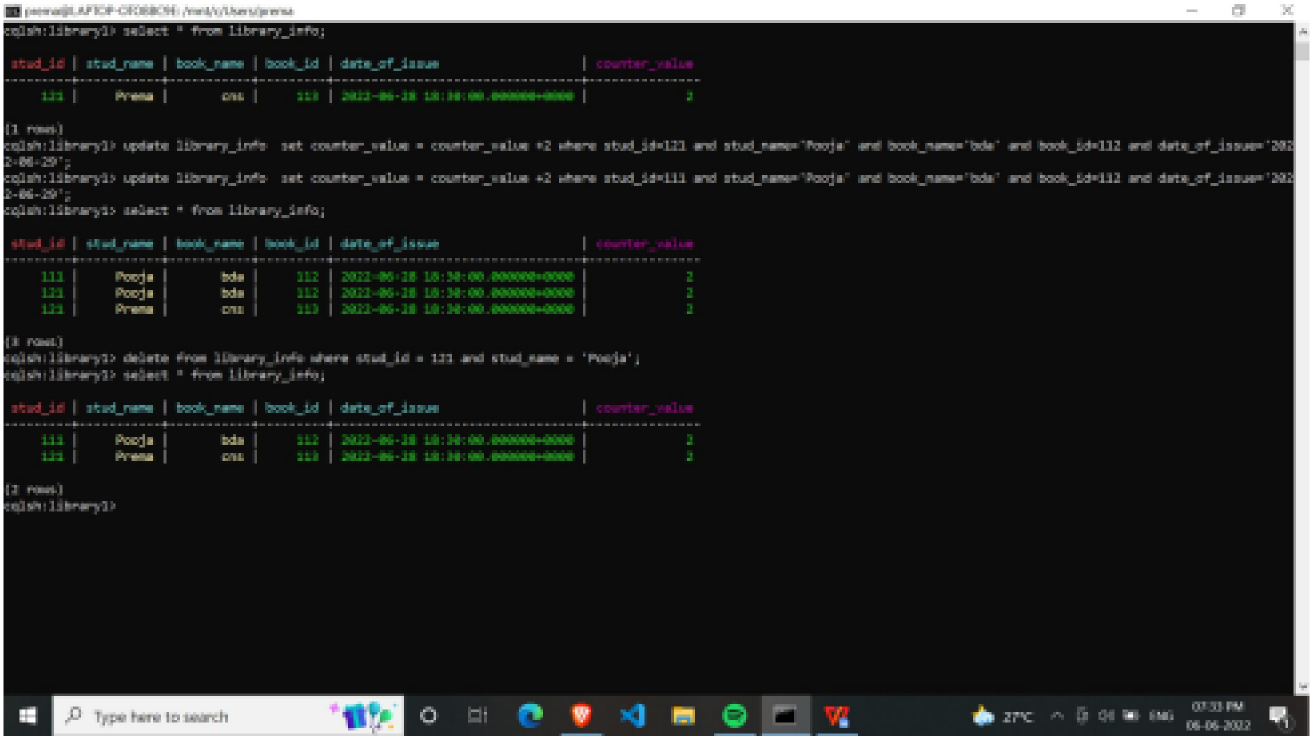
```
TRUNCATE library_info;
cqlsh:library1> select * from library_info;

    stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+---------+---------------+---------------

(0 rows)
cqlsh:library1> COPY
library_info(stud_id,counter_value,stud_name,book_name,book_id,date_of_issue
)FROM 'lib1.csv' ;
Using 7 child processes

Starting copy of library1.library_info with columns [stud_id, counter_value,
stud_name, book_name, book_id, date_of_issue].
Processed: 2 rows; Rate: 4 rows/s; Avg. rate: 6 rows/s 2 rows imported from 1
files in 0.364 seconds (0 skipped).
cqlsh:library1> select * from library_info;
```

```
stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+---------+---------------------------------+--------------- 111 | Pooja | bda |
    112 | 2022-06-28 18:30:00.000000+0000 | 2121 | Prema | cns | 113 | 2022-06-28
    18:30:00.000000+0000 | 2
```

Output screenshots:



```
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\prema>WSL
-bash: export: `Files/Java/jdk1.8.0_261': not a valid identifier
prema@LAPTOP-CFOBBC9I:/mnt/c/Users/prema$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.4 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE LIBRARY1 WITH REPLICATION = {
   ... 'class':'SimpleStrategy',
   ... 'replication_factor':1};
cqlsh> use LIBRARY1;
cqlsh:library1> create table library_info(
            ... stud_id int,
            ... counter_value counter,
            ... stud_name text,
            ... book_name text,
            ... book_id int,
            ... date_of_issue timestamp);
InvalidRequest: Error from server: code=2200 [Invalid query] message="No PRIMARY KEY specifed for table 'library_info' (exactly one required)"
cqlsh:library1> create table library_info(  stud_id int, counter_value counter, stud_name text, book_name text, book_id int, date_of_issue timestamp);
InvalidRequest: Error from server: code=2200 [Invalid query] message="No PRIMARY KEY specifed for table 'library_info' (exactly one required)"
cqlsh:library1> create table library_info(  stud_id int, counter_value counter, stud_name text, book_name text, book_id int, date_of_issue timestamp,PRIMARY KEY(stud_id,stu
d_name,book_name,book_id,date_of_joining));
InvalidRequest: Error from server: code=2200 [Invalid query] message="Unknown column 'date_of_joining' referenced in PRIMARY KEY for table 'library_info'"
cqlsh:library1> create table library_info(  stud_id int, counter_value counter, stud_name text, book_name text, book_id int, date_of_issue timestamp,PRIMARY KEY(stud_id,stu
d_name,book_name,book_id,date_of_issue));
cqlsh:library1> update library_info
            ... set counter_value = counter_value +1 where stud_id=121 and stud_name='Prema' and book_name='cns' and book_id=113 and date_of_issue='2022-06-29';
cqlsh:library1> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     121 |     Prema |       cns |     113 | 2022-06-28 18:30:00.000000+0000 |             1

(1 rows)
cqlsh:library1> update library_info  set counter_value = counter_value +1 where stud_id=121 and stud_name='Prema' and book_name='cns' and book_id=113 and date_of_issue='202
2-06-29';
cqlsh:library1> select * from library_info;
```

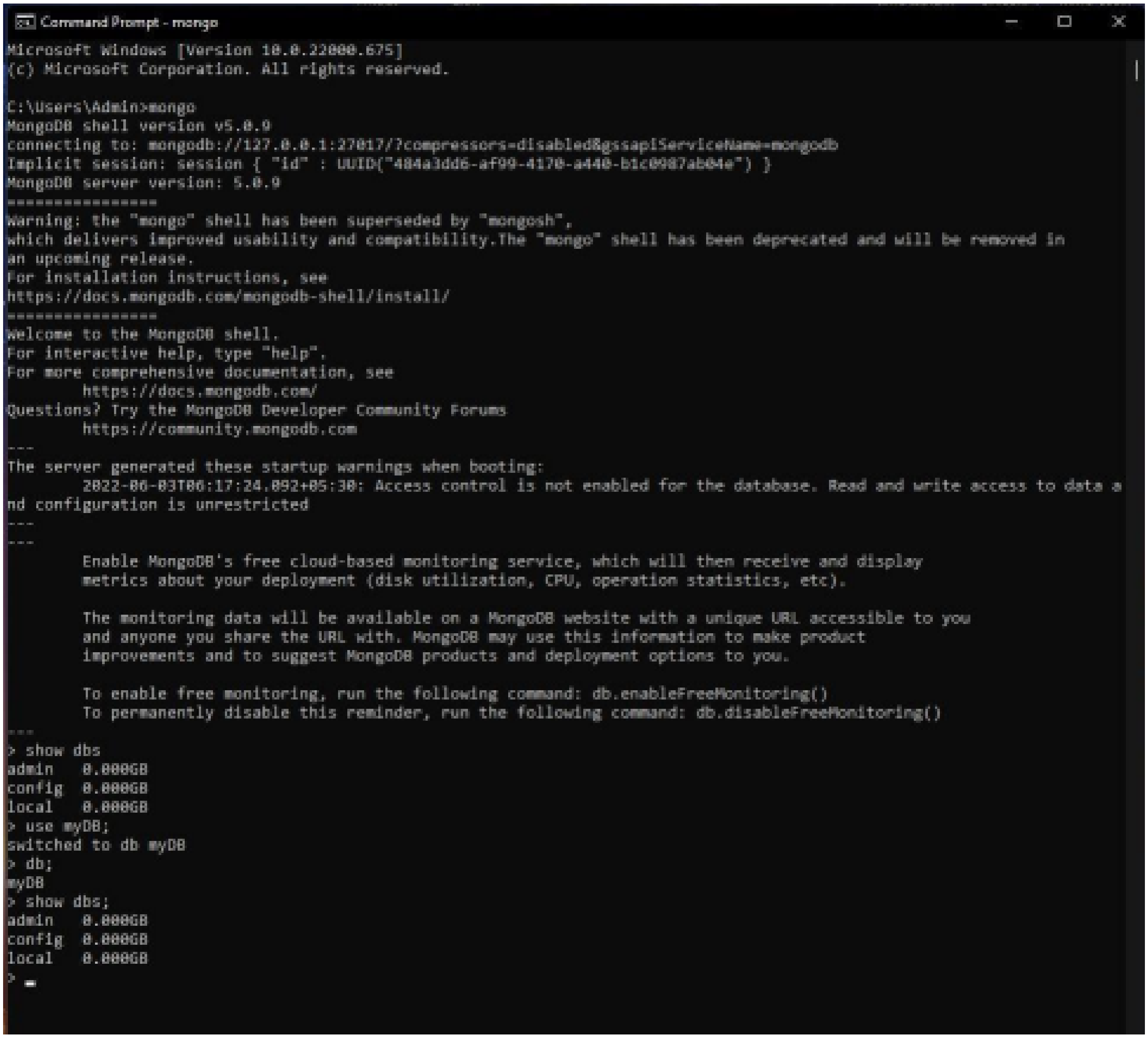


```
cqlsh:library1> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     121 |     Prema |       cns |     113 | 2022-06-28 18:30:00.000000+0000 |             2

(1 rows)
cqlsh:library1> update library_info  set counter_value = counter_value +2 where stud_id=121 and stud_name='Pooja' and book_name='bda' and book_id=112 and date_of_issue='202
2-06-29';
cqlsh:library1> update library_info  set counter_value = counter_value +2 where stud_id=111 and stud_name='Pooja' and book_name='bda' and book_id=112 and date_of_issue='202
2-06-29';
cqlsh:library1> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     111 |     Pooja |       bda |     112 | 2022-06-28 18:30:00.000000+0000 |             2
     121 |     Pooja |       bda |     112 | 2022-06-28 18:30:00.000000+0000 |             2
     121 |     Prema |       cns |     113 | 2022-06-28 18:30:00.000000+0000 |             2

(3 rows)
cqlsh:library1> delete from library_info where stud_id = 121 and stud_name = 'Pooja';
cqlsh:library1> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     111 |     Pooja |       bda |     112 | 2022-06-28 18:30:00.000000+0000 |             2
     121 |     Prema |       cns |     113 | 2022-06-28 18:30:00.000000+0000 |             2

(2 rows)
cqlsh:library1>
```

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)



```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>mongo
MongoDB shell version v5.0.9
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("484a3dd6-af99-4170-a440-b1c0987ab04e") }
MongoDB server version: 5.0.9
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
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”); => sql equivalent CREATE TABLE STUDENT(…);

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

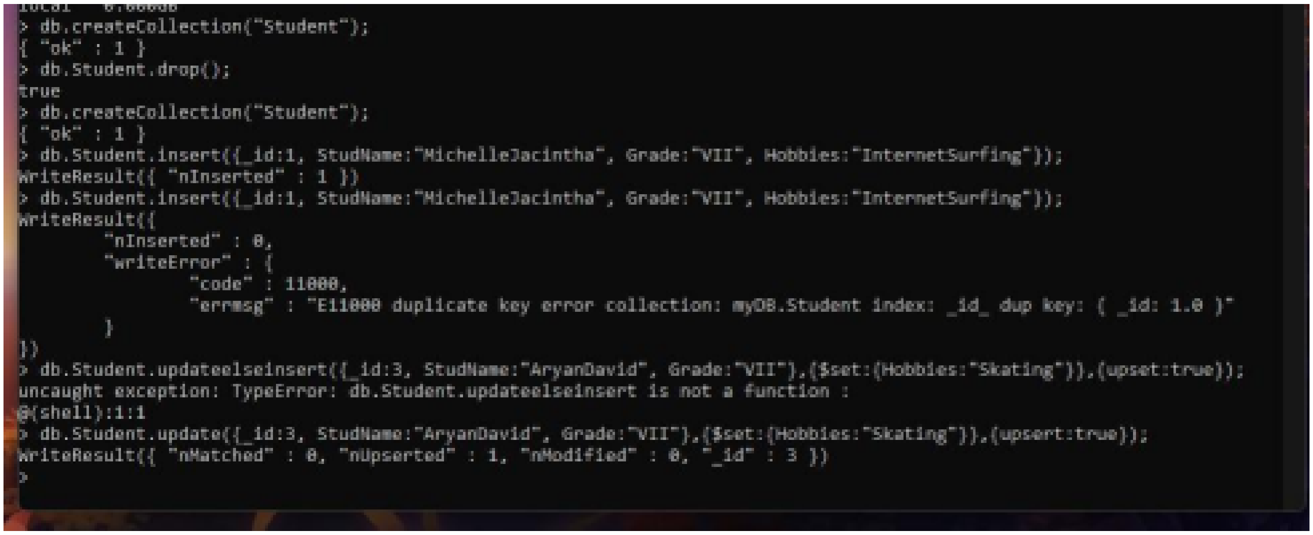
db.Student.drop();

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

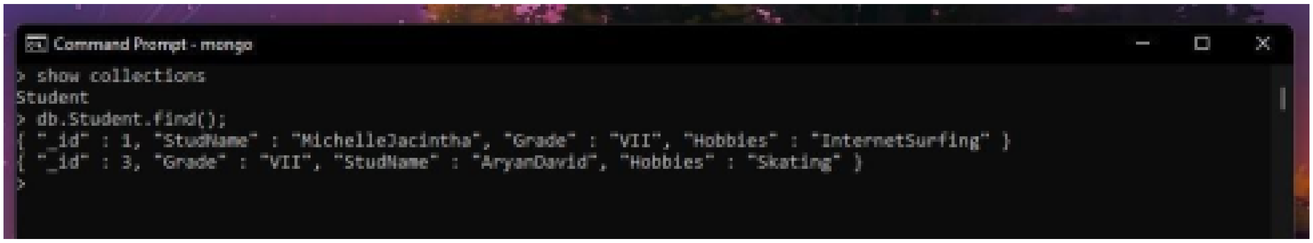
db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:& quot;Int ernetS urfing"});

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:{Hobbie s:&quo t;Skatin g"}},{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" }
>
```

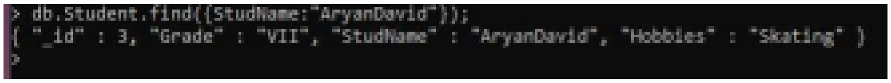
5. FIND METHOD

A.

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

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

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

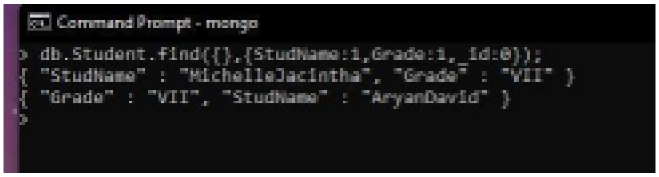


```
> db.Student.find({StudName:"AryanDavid"});
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
>
```

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});

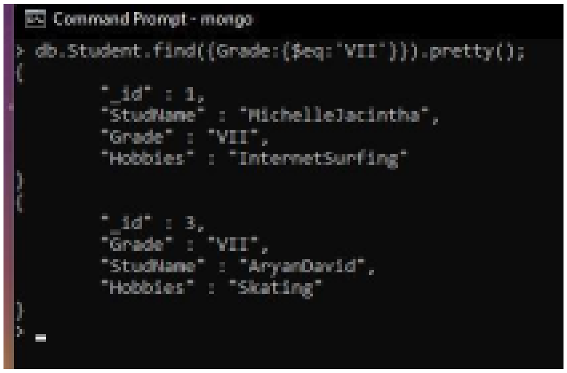


```
> db.Student.find({},{StudName:1,Grade:1,_id:0});
{ "StudName" : "MichelleJacintha", "Grade" : "VII" }
{ "Grade" : "VII", "StudName" : "AryanDavid" }
>
```

C.

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

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



```
> 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 ();

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

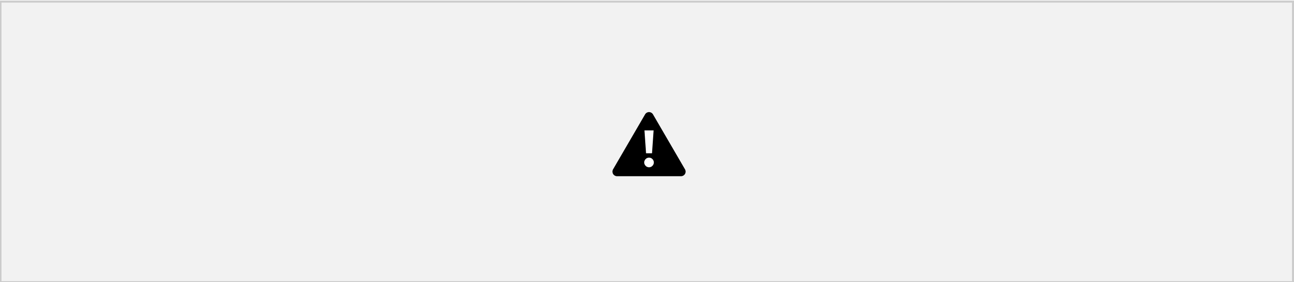
G.
To find the number of documents in the Students collection.
db.Student.count();

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

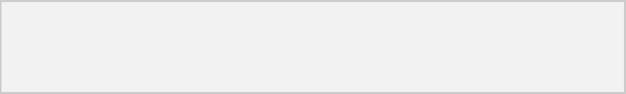
This command used at the command prompt exports MongoDB JSON documents from “Customers” collection in the “test” database into a CSV file “Output.txt” in the D:drive.

```
mongoexport --host localhost --db Student --collection airlines --csv --out
/home/hduser/Desktop/output.txt –fields “Year”,”Quarter”
```

V. Save Method :

Save() method will insert a new document, if the document with the _id does not exist. If it exists it will replace the exisiting document.

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



VI. Add a new field to existing Document:

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

VII. Remove the field in an existing Document

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

VIII. Finding Document based on search criteria suppressing few fields
db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});
To find those documents where the Grade is not set to ‘VII’
db.Student.find({Grade:{$ne:'VII'}}).pretty();
To find documents from the Students collection where the StudName ends with s.
db.Student.find({StudName:/s$/}).pretty();

IX. to set a particular field value to NULL

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'] })

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

To find the document with a particular id and display the first two elements from the array “fruits”
db.food.find({_id:1},{“fruits”:{$slice:2}})

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

update on Array: using particular id replace the element present in the 1 st index position of the fruits array with apple
db.food.update({_id:3},{$set:{'fruits.1':'apple'}})
insert new key value pairs in the fruits array
db.food.update({_id:2},{$push:{price:{grapes:80,mango:200,cherry:100 }}})

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}}});