**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



**LAB REPORT**

**on**

BIG DATA ANALYTICS (20CS6PEBDA)

***Submitted by***

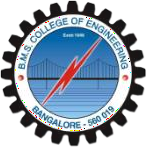
**BALAJI S K (1BM19CS134)**

***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 **BALAJI S K(1BM19CS134),** 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.

**Dr. RAJESHWARI B S Dr. JYOTHI S. NAYAK**

Assistant Professor Professor & HOD

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

INDEX

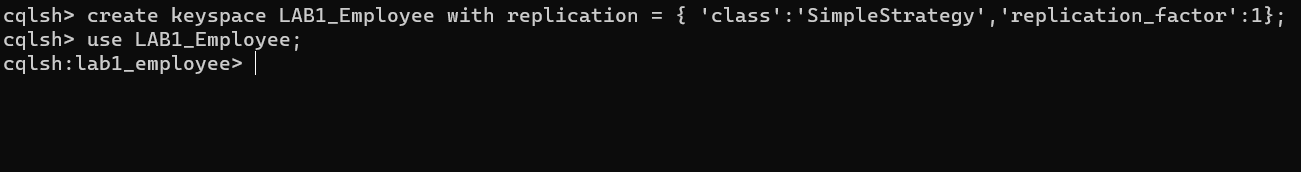
|  |  |  |
| --- | --- | --- |
| **Sl.**  **No.** | **Experiment Title** | **Page No.** |
| **1.** | **Cassandra Lab Program 1: -** Student Database | **4-6** |
| **2.** | **Cassandra Lab Program 2: -** Library Database | **7-8** |
| **3.** | **MongoDB Lab Program 1 (CRUD Demonstration): -** Students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

# Course Outcome

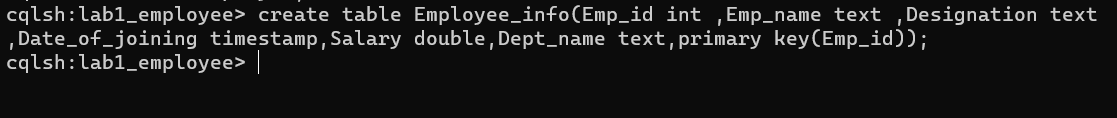
|  |  |
| --- | --- |
| CO1 | Apply the concept of NoSQL, Hadoop or Spark for a given task |
| CO2 | Analyze the Big Data and obtain insight using data analytics mechanisms. |
| CO3 | Design and implement Big data applications by applying NoSQL, Hadoop or Spark |

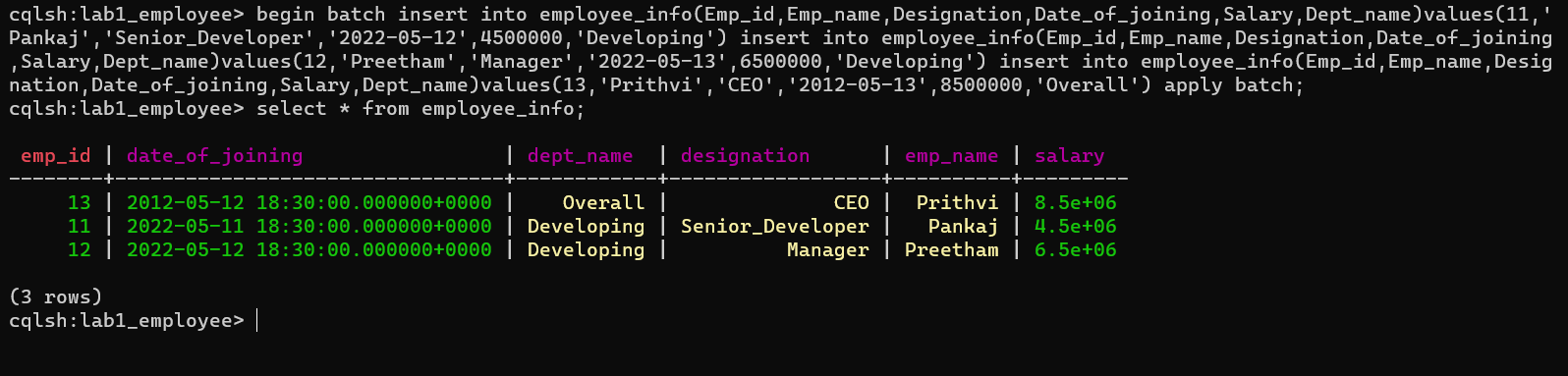
**LAB 1**

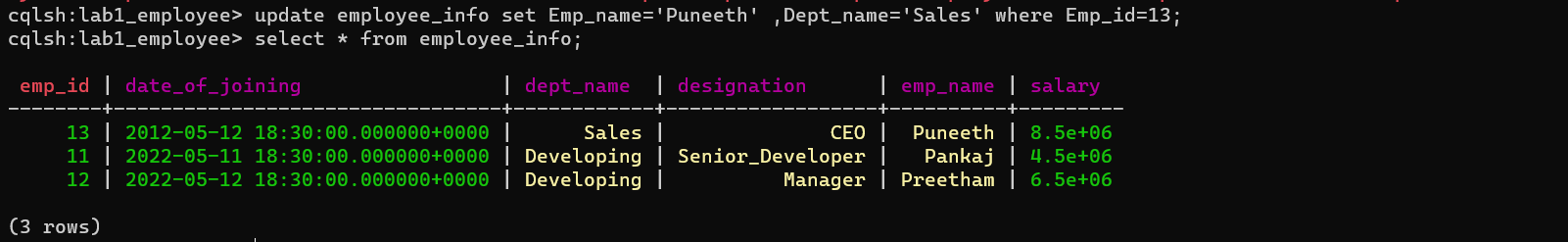
1. Create a key space by name Employee



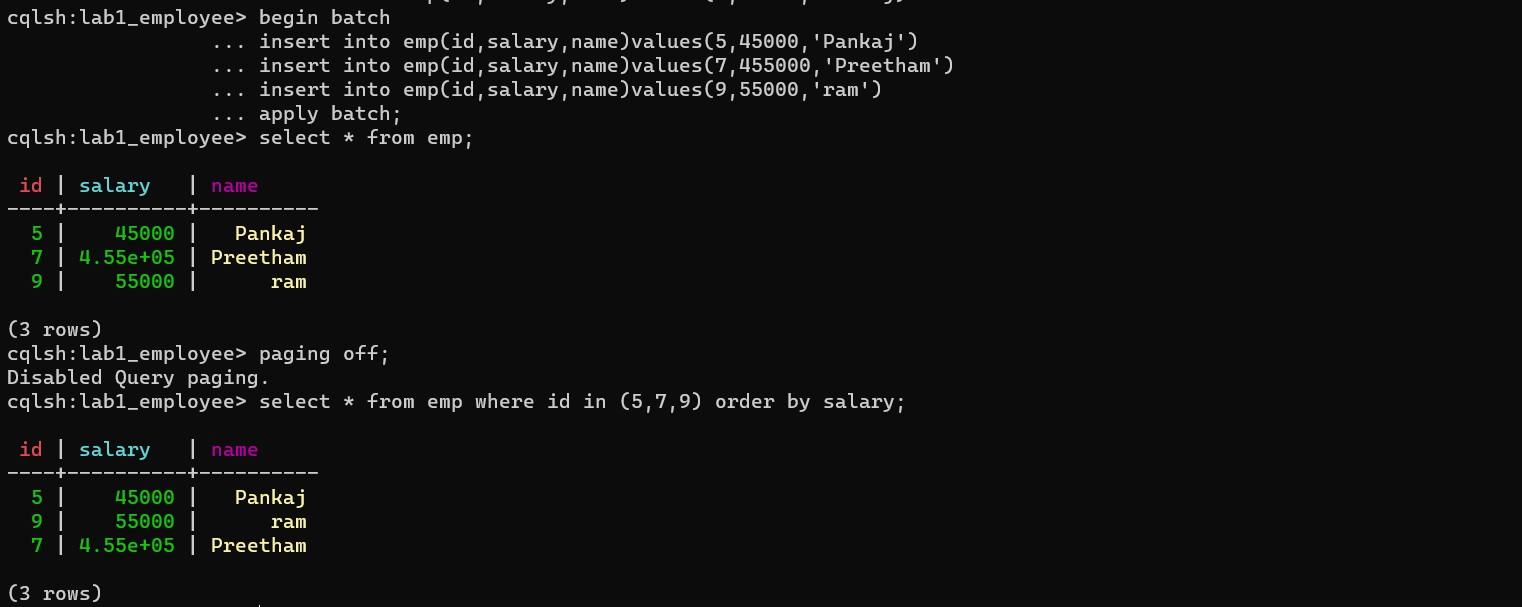
1. Create a column family by name Employee-Info with attributes Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name



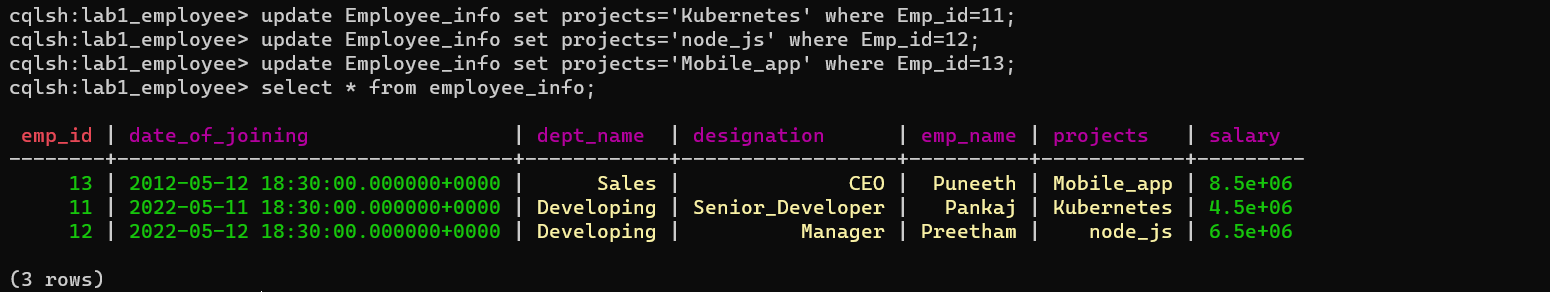
1. Insert the values into the table in batch
2. Update Employee name and Department of Emp-Id 121



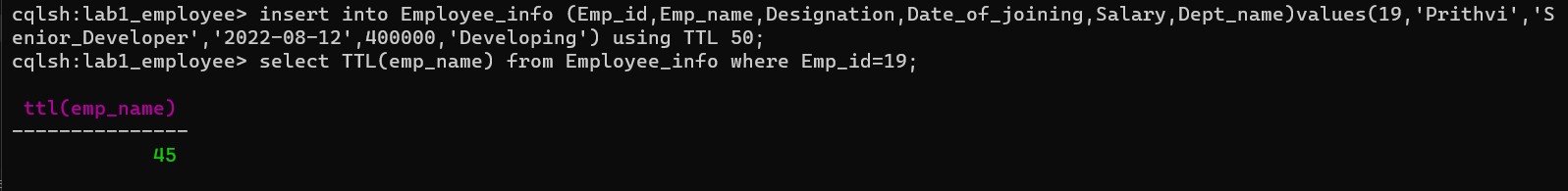
1. Sort the details of Employee records based on salary



1. Alter the schema of the table Employee\_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.
2. Update the altered table to add project names.

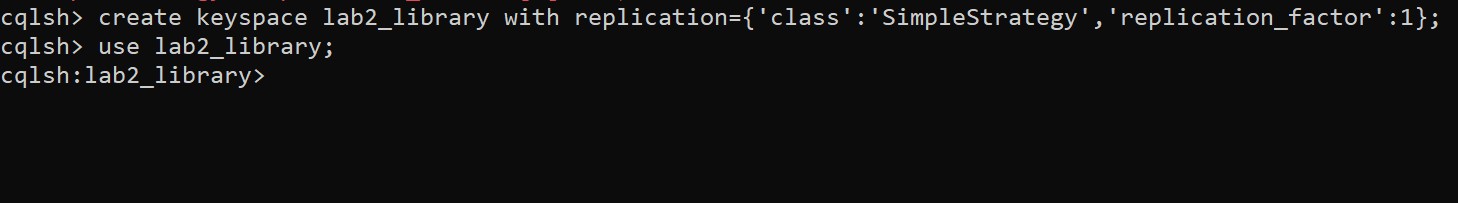


8 Create a TTL of 15 seconds to display the values of Employees.

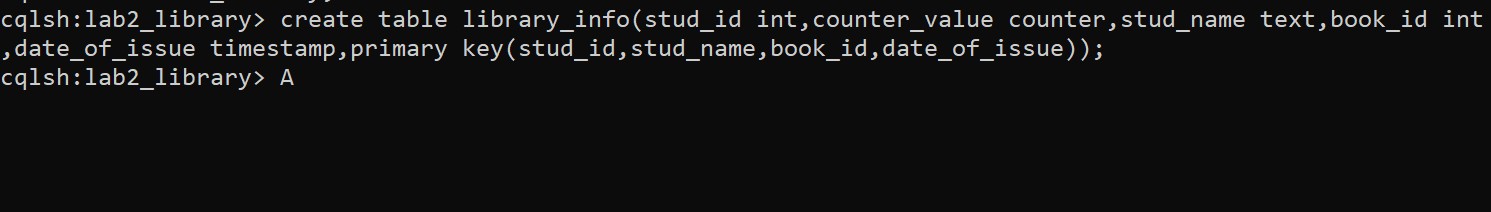


# LAB 2

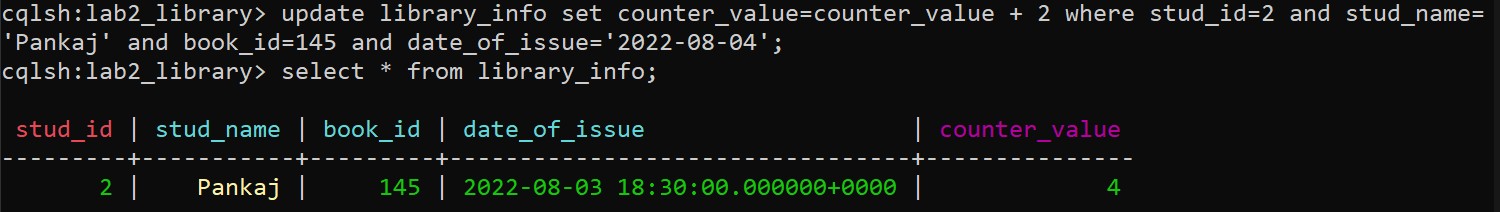
1 Create a key space by name Library



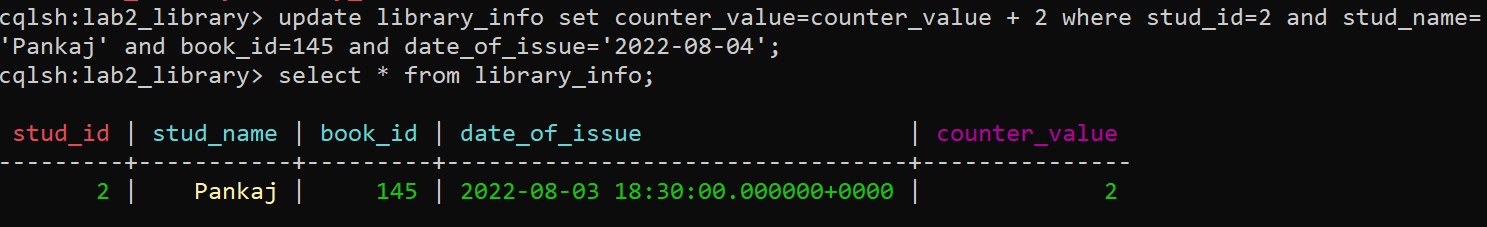
1. Create a column family by name Library-Info with attributes Stud\_Id Primary Key, Counter\_value of type Counter,

Stud\_Name, Book-Name, Book-Id, Date\_of\_issue

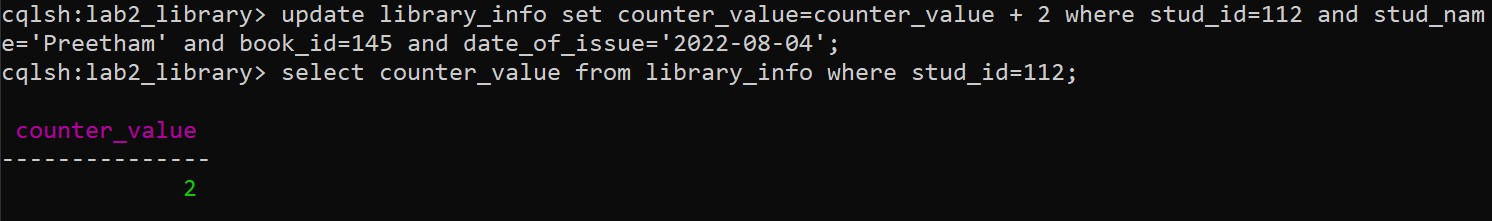
1. Insert the values into the table in batch

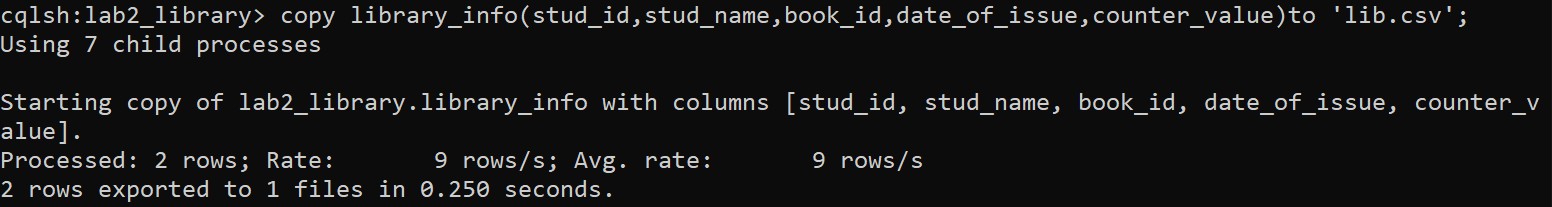
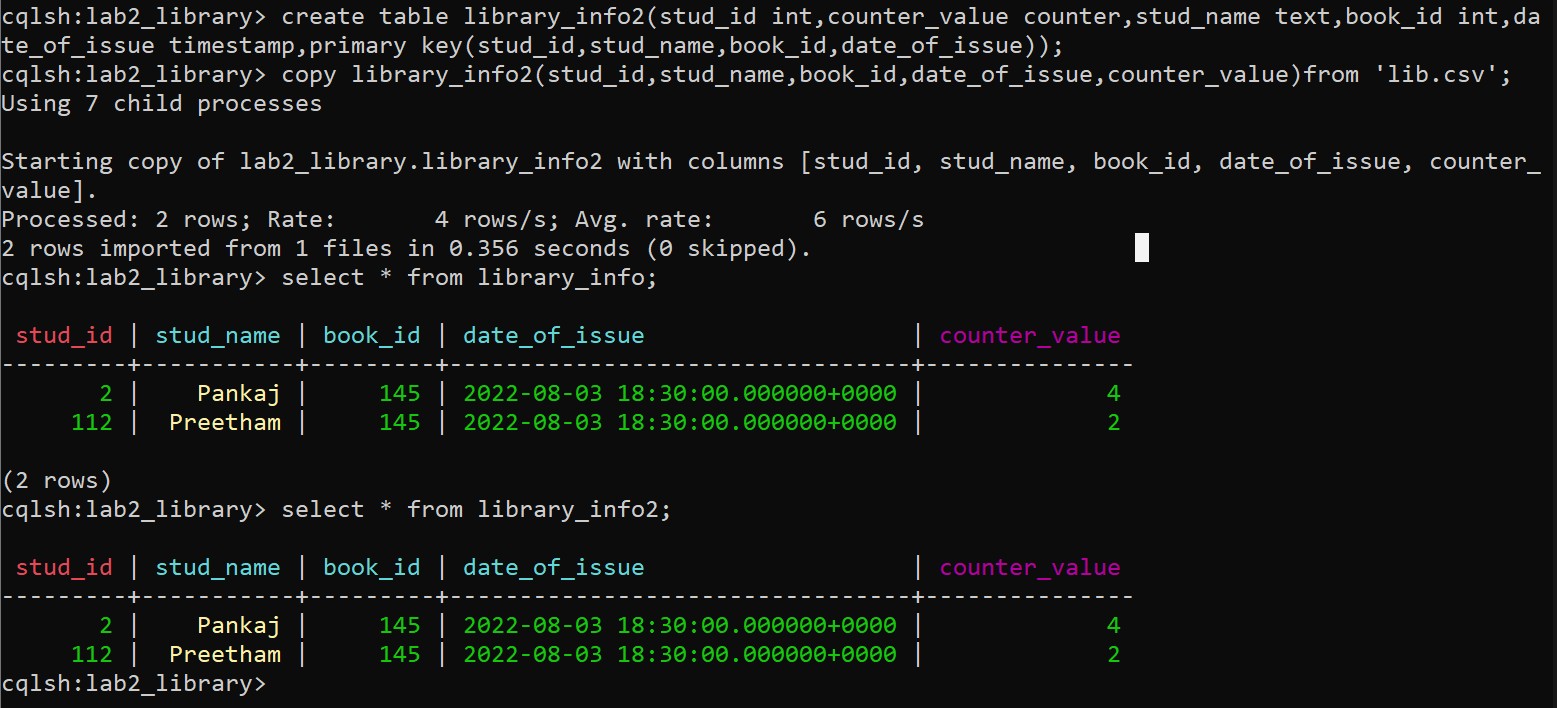


1. Display the details of the table created and increase the value of the counter



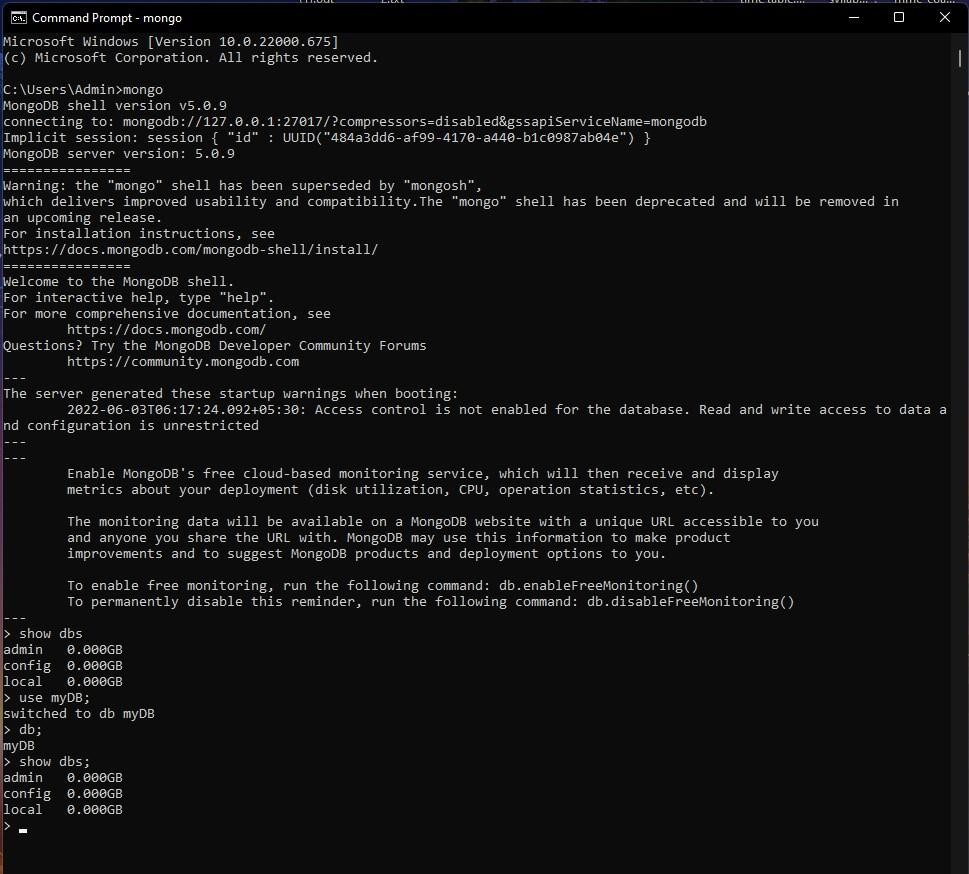
1. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.



1. Export the created column to a csv file
2. Import a given csv dataset from local file system into Cassandra column family

# LAB 3

1. CREATE DATABASE IN MONGODB. use myDB; db; (Confirm the

existence of your database) show dbs; (To list all databases)

1. 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(…);

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

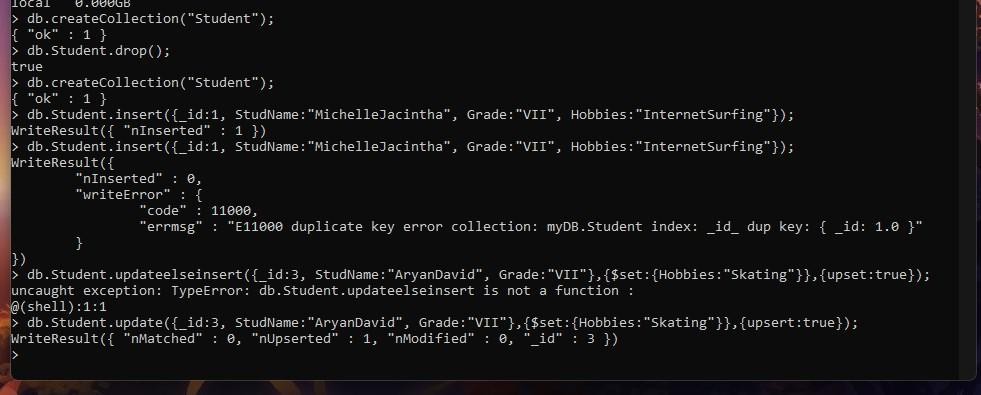
db.Student.drop();

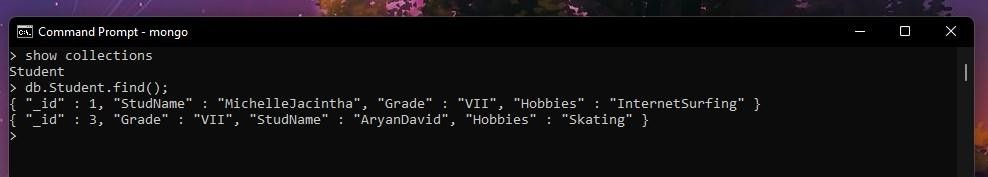
* 1. Create a collection by the name “Students” and store the following data in it.

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

* 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:&quot;AryanDavid&quot;,Grade:&quot;VII&quot;},{$set:{Hobbie s:&quo t;Skatin g&quot;}},{upsert:true});





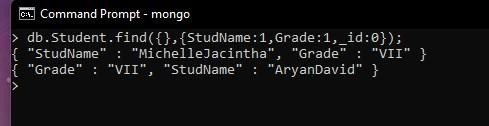
* 1. FIND METHOD

1. 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} )

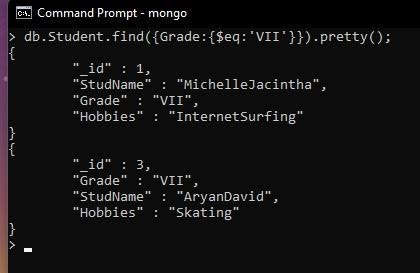


1. To display only the StudName and Grade from all the documents of the Students collection. The identifier\_id should be suppressed and NOT displayed. db.Student.find({},{StudName:1,Grade:1,\_id:0});

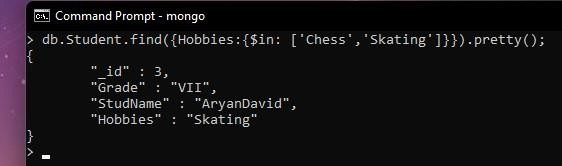


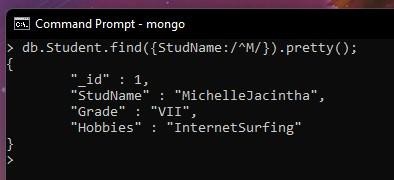
1. To find those documents where the Grade is set to ‘VII’

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



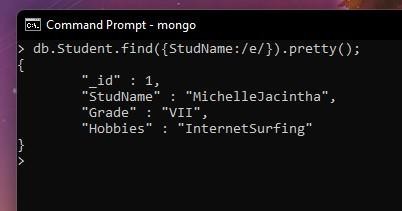
1. To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’. db.Student.find({Hobbies :{ $in: [&#39;Chess&#39;,&#39;Skating&#39;]}}).pretty ();



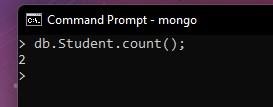
1. To find documents from the Students collection where the StudName begins with “M”. db.Student.find({StudName:/^M/}).pretty();
2. To

find documents from the

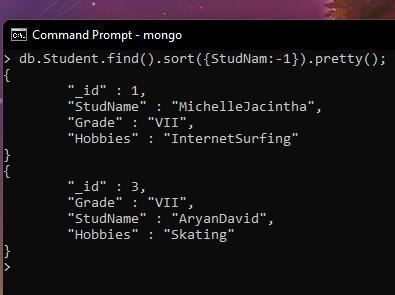
Students collection where the StudNamehas an “e” in any position. db.Student.find({StudName:/e/}).pretty();



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



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

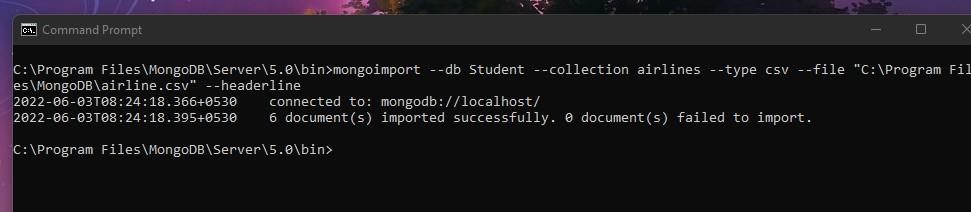


1. Import data from a CSV file

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



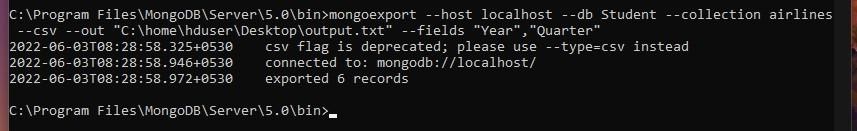
1. Export data to a CSV file

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.

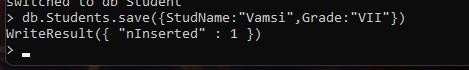
mongoexport --host localhost --db Student --collection airlines --csv --out

/home/hduser/Desktop/output.txt –fields “Year”,”Quarter”

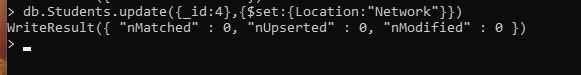


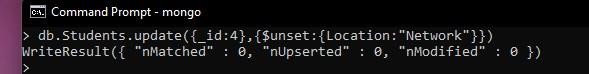
1. Save Method :

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”})



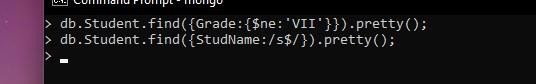
1. Add a new field to existing Document: db.Students.update({\_id:4},{$set:{Location:”Network”}})



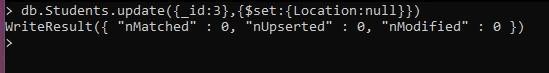
1. Remove the field in an existing Document db.Students.update({\_id:4},{$unset:{Location:”Network”}})
2. Finding Document based on search criteria suppressing few fields db.Student.find({\_id:1},{StudName:1,Grade:1,\_id:0});

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();



1. to set a particular field value to NULL

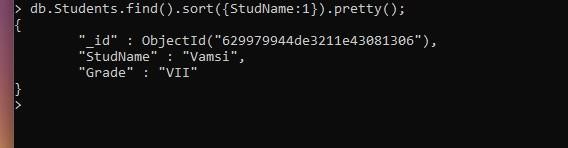


X Count the number of documents in Student Collections

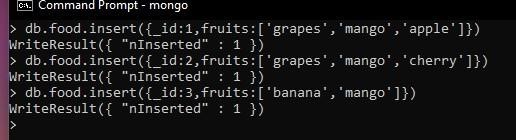


1. Count the number of documents in Student Collections with grade :VII db.Students.count({Grade:”VII”}) retrieve first 3 documents

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()



1. 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;] } )



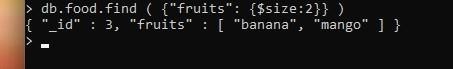
To find those documents from the “food” collection which has the “fruits array” constitute of “grapes”, “mango” and “apple”. db.food.find ( {fruits: [&#39;grapes&#39;,&#39;mango&#39;,&#39;apple&#39;] } ). pretty().



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



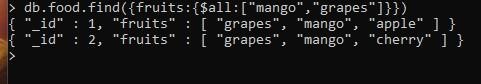
To find those documents from the “food” collection where the size of the array is two. db.food.find ( {“fruits”: {$size:2}} )



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}})

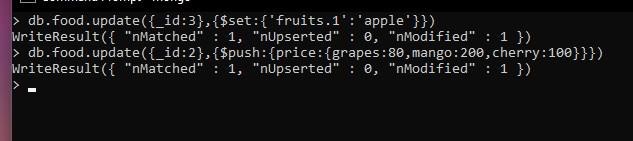


To find all the documets from the food collection which have elements mango and grapes in the array “fruits” db.food.find({fruits:{$all:[“mango”,”grapes”]}})



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},{$set:{&#39;fruits.1&#39;:&#39;apple&#39;}}) insert new key value pairs in the fruits array db.food.update({\_id:2},{$push:{price:{grapes:80,mango:200,cherry:100

}}})



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}}});

