**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**BIG DATA ANALYTICS**

**(20CS6PEBDA)**

***Submitted by***

**KRISHNA MOHAN DULLOLLI (1BM19CS075)**

***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 **KRISHNA MOHAN DULLOLLI (1BM19CS075),** 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-Incharge               **Prof. Pallavi G B**

Designation Assistant Professor

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| **1** | **MongoDB CRUD Operations** | **4** |
| **2** | **MongoDB Operations** | **7** |
| **3** | **Cassandra Lab 1** | **10** |
| **4** | **Cassandra Lab 2** | **11** |

**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 |

**1 MongoDB CRUD Operations**

**I. CREATE DATABASE IN MONGODB**

**>use krishnaDB**

switched to db krishnaDB

**II. CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS**

**>db.createCollection("Student");**

{ "ok" : 1 }

**>db.Student.insert({\_id:1,name:"Krishna",grade:9});**

WriteResult({ "nInserted" : 1 })

**>db.Student.update({\_id:6,name:"qwert"},{$set:{grade:4}},{upsert:true});**

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "\_id" : 6 })

**>db.Student.find();**

{ "\_id" : 1, "name" : "Krishna", "grade" : 9 }

{ "\_id" : 2, "name" : "Abc", "grade" : 10 }

{ "\_id" : 3, "name" : "Mno", "grade" : 5 }

{ "\_id" : 4, "name" : "Pqr", "grade" : 8 }

**> show collections;**

Student

**III. Save Method**

**> db.Student.save({name:"zzz",\_id:10,grade:8});**

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "\_id" : 10 })

**IV. COUNT**

**> db.Student.count();**

6

**> db.Student.count({grade:9});**

1

**V FIND**

**> db.Student.find({grade:{$lt:5}},{name:1,grade:1,\_id:0});**

{ "grade" : 2, "name" : "qwert" }

**> db.Student.find({name:{$in:["Krishna","Abc","Mno"]}},{name:1,grade:1,\_id :0});**

{ "name" : "Krishna", "grade" : 9 }

{ "name" : "Abc", "grade" : 10 }

{ "name" : "Mno", "grade" : 5 }

**> db.Student.find({name:/^S/},{name:1,grade:1,\_id:0});**

{ "name" : "Krishna", "grade" : 9 }

**> db.Student.find({name:/.b/},{name:1,grade:1,\_id:0});**

{ "name" : "Abc", "grade" : 10 }

**> db.Student.find().sort({name:1});**

{ "\_id" : 2, "name" : "Abc", "grade" : 10 }

{ "\_id" : 3, "name" : "Mno", "grade" : 5 }

{ "\_id" : 4, "name" : "Pqr", "grade" : 8 }

{ "\_id" : 1, "name" : "Krishna", "grade" : 9 }

{ "\_id" : 7, "name" : "kkk", "grade" : 6 }

{ "\_id" : 6, "grade" : 2, "name" : "qwert" }

**> db.Student.find().sort({name:1,grade:-1});**

{ "\_id" : 2, "name" : "Abc", "grade" : 10 }

{ "\_id" : 3, "name" : "Mno", "grade" : 5 }

{ "\_id" : 4, "name" : "Pqr", "grade" : 8 }

{ "\_id" : 1, "name" : "Krishna", "grade" : 9 }

{ "\_id" : 7, "name" : "kkk", "grade" : 6 }

{ "\_id" : 6, "grade" : 2, "name" : "qwert" }

**> db.Student.find({grade:8}).limit(3);**

{ "\_id" : 4, "name" : "Pqr", "grade" : 8 }

{ "\_id" : 10, "name" : "zzz", "grade" : 8 }

**> db.Student.find().skip(2);**

{ "\_id" : 3, "name" : "Mno", "grade" : 5 }

{ "\_id" : 4, "name" : "Pqr", "grade" : 8 }

{ "\_id" : 6, "grade" : 2, "name" : "qwert" }

{ "\_id" : 7, "name" : "kkk", "grade" : 6 }

{ "\_id" : 10, "name" : "zzz", "grade" : 8 }

**VI.AGGREGATE FUNCTIONS**

**> db.faculty.aggregate ( {$match:{department:"mech"}}, {$group : {\_id : "$designation", AverageSal :{$avg:"$salary"} } }, {$match:{AverageSal:{$gt:50000}}});**

{ "\_id" : " associate prof", "AverageSal" : 85000 }

{ "\_id" : "assistant prof", "AverageSal" : 70000 }

**VII. ARRAYS**

**> db.food.insert({\_id:1,fruits:['apple','mango']});**

WriteResult({ "nInserted" : 1 })

**> db.food.find({fruits:['pineapple','mango','orange']});**

{ "\_id" : 3, "fruits" : [ "pineapple", "mango", "orange" ] }

**> db.food.find({fruits:{$all:['pineapple']}});**

{ "\_id" : 2, "fruits" : [ "pineapple", "mango", "grapes" ] }

{ "\_id" : 3, "fruits" : [ "pineapple", "mango", "orange" ] }

**> db.food.update({\_id:2},{$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 })

**2. MongoDB Operations**

**1)Faculty DB**

**i) Create a database for Faculty and Create a Faculty Collection(Faculty\_id, Name, Designation ,Department, Age, Salary, Specialization(Set)).**

**>use Faculty**

**> db.createCollection("faculty")  
ii) Insert required documents to the collection.**

**> db.faculty.insert({\_id:1,name:"abc",designation:"assistant prof",department:"mech",age:31,salary:90000,specialization:['python','mysql','autocad']});  
iii) First Filter on “Dept\_Name:MECH” and then group it on “Designation” and  
compute the Average Salary for that Designation and filter those  
documents where the “Avg\_Sal” is greater than 650000.**

**> db.faculty.aggregate ( {$match:{department:"mech"}}, {$group : {\_id : "$designation", AverageSal :{$avg:"$salary"} } }, {$match:{AverageSal:{$gt:50000}}});**

{ "\_id" : " associate prof", "AverageSal" : 85000 }

{ "\_id" : "assistant prof", "AverageSal" : 70000 }

**2) Consider a table “Product” with the following columns:  
Product \_id  
ProductName  
ManufacturingDate  
Price  
Quantity  
Write MongoDB queries for the following:**

**> use Products switched to db Products**

**> db.createCollection("product");**

{ "ok" : 1 }

**> db.product.insert({pid:1,pname:"keyboard",mdate:2001,price:1800,quantity:2});**

WriteResult({ "nInserted" : 1 }) **i)To display only the product name from all the documents of the product collection.**

**> db.product.find({},{pname:1,\_id:0});**

{ "pname" : "keyboard" }

{ "pname" : "mouse" }

{ "pname" : "motherboard" } **ii)To display only the Product ID, ExpiryDate as well as the quantity from the document of the product collection where the \_id column is 1.**

**> db.product.find({pid:1},**

{pid:1,\_id:0,mdate:1,quantity:1});

{ "pid" : 1, "mdate" : 2001, "quantity" : 2 }

**iii)To find those documents where the price is not set to 45000.**

**> db.product.find({price:{$ne:45000}},{pname:1,\_id:0});**

{ "pname" : "keyboard" }

{ "pname" : "mouse" }

{ "pname" : "motherboard" } **iv)To find those documents from the Product collection where the quantity is set to 30    and the product name is set to ‘LEDTV’.**

**> db.product.find({$and:[{quantity:{$eq:30}},{pname:{$eq:"LED TV"}}]},{pname:1,\_id:0})**

{ "pname" : "LED TV" } **v)To find documents from the Product collection where the Product name  
ends in ‘r’.**

**> db.product.find({pname:/d$/},{pname:1,quantity:1,\_id:0})**

{ "pname" : "keyboard", "quantity" : 2 }

{ "pname" : "motherboard", "quantity" : 150 } **3)Create a mongodb collection Hospital. Demonstrate the following by choosing fields of      your choice.**

**> use Hospital switched to db Hospital**

**> db.createCollection("hospital");**

{ "ok" : 1 }

**> db.hospital.insert({\_id:1,name:"xyz",diseases:["diabetes","high bp","fever"]});**

WriteResult({ "nInserted" : 1 }) **1.      Insert three documents**

**> db.hospital.updateMany({},{$pull:{diseases:"fever"}});**

{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 2 } **2.      Use Arrays(Use Pull and Pop operation)**

**> db.hospital.updateOne({\_id:1},{$pop:{diseases:-1}});**

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 } **3.      Use Index**

**> db.hospital.find({"diseases.2":"nausea"});**

{ "\_id" : 3, "name" : "mno", "diseases" : [ "covid", "sarscov", "nausea" ] } **4.      Use Cursors  
> db.hospital.find({}).count();**

3

**> db.hospital.find({}).limit(2);**

{ "\_id" : 1, "name" : "xyz", "diseases" : [ "high bp" ] } { "\_id" : 2, "name" : "abc", "diseases" : [ "typhoid", "cholera" ] }

**> db.hospital.find({}).size();**

3 **5.      Updation**

**> db.hospital.update({\_id:3},{$set:{'diseases.1':'sarscov'}});**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**3. Cassandra Lab 1**

**1. Create a key space by name Employee**

**cqlsh:saf> create keyspace Employee with replication={'class':'SimpleStrategy','replication\_factor':1}; cqlsh:saf> use 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:employee> create table empInfo( emp\_id int PRIMARY KEY, emp\_name text,desig text,dpj timestamp,salary int,dept\_name text );  
3. Insert the values into the table in batch**

**cqlsh:employee> insert into empInfo(emp\_id,emp\_name,desig,dpj,salary,dept\_name) values( 1, 'krishna', 'sde', '2022-05-05', 200000, 'cse' );  
4. Update Employee name and Department of Emp-Id 121**

**cqlsh:employee> update empInfo set emp\_name='zzz',dept\_name='ie'where emp\_id=2;  
5. Sort the details of Employee records based on salary**

**.cqlsh:employee> select \* from emp\_Info where emp\_id in (1,2,3) order by salary;  
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 empInfo add project set**

**7. Update the altered table to add project names.**

**cqlsh:employee> update empInfo set project={'reactJs','Ml'} where emp\_id=1;  
8 Create a TTL of 15 seconds to display the values of Employees.**

**cqlsh:employee> insert into empInfo(emp\_id,emp\_name,desig,dpj,salary,dept\_name) values( 5, 'wxy', 'sde', '2022-02-05', 250000, 'cse' ) using ttl 30; cqlsh:employee> select ttl(emp\_name) from empInfo;**

**4. Cassandra Lab 2**

**1 Create a key space 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 Primary Key,Counter\_value of type Counter,Stud\_Name, Book-Name, Book-Id, Date\_of\_issue**

**CREATE TABLE lib.libinfo1 ( s\_id int, sname text, book text, bid int, doi timestamp, counter\_val counter, PRIMARY KEY (s\_id, sname, book, bid, doi) );  
3. Insert the values into the table in batch**

**update libinfo set counter\_val=counter\_val+1 where s\_id=1 and sname='saf' and book='harry potter1' and bid=1 and doi='2022-05-05';  
4. Display the details of the table created and increase the value of the counter**

**cqlsh:lib> update libinfo set counter\_val=counter\_val+1 where s\_id=1 and sname='saf' and book='harry potter1'; cqlsh:lib> select \* from libinfo;  
5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.**

**cqlsh:lib> select counter\_val from libinfo where s\_id=1 and sname='saf' and book='harry potter1';**

counter\_val

-------------

2 **6. Export the created column to a csv file**

**COPY libinfo(s\_id,sname,book,bid,doi,counter\_val) TO 'data1.csv' WITH HEADER = TRUE;  
7. Import a given csv dataset from local file system into Cassandra column family**

**COPY libinfo(s\_id,sname,book,bid,doi) FROM 'libdata.csv' WITH HEADER = TRUE;**