VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

PRATIBHA JAMKHANDI (1BM19CS119)

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 **PRATIBHA JAMKHANDI(1BM19CS119),** 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.

Nameof the Lab-Incharge
Designation
Department of CSE
BMSCE, Bengaluru

ANTARA ROY CHOUDHURY
Assistant Professor
Department of CSE
BMSCE, Bengaluru

`

Index Sheet

Course Outcome

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

LAB 1

CREATE DATABASE IN MONGODB.

use myDB;

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

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”);

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

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

db.Student.drop();

```
> db.Student.drop();
true
```

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:"InternetSurfing"});

```
> db.Student.insert({_id:1,StudName:"pratibha",Grade:"vii",Hobbies:"Chess"});
WriteResult({ "nInserted" : 1 })
```

4. Insert the document for “Rahul” into the Students collection only if it does not already exist in the collection. However, if it is already present in the collection, then update the document with new values. (Update his Hobbies from “Skating” to “Chess”.) Use “Update else insert” (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it).

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

```
> db.Student.update({_id:3,StudName:"rahul",Grade:"vii"},{$set:{Hobbies:"Skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
```

5. FIND METHOD

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

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

```
> db.Student.find({StudName:"pratibha"});
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
```

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" : "pratibha", "Grade" : "vii" }
{ "StudName" : "prathiksha", "Grade" : "viii" }
{ "Grade" : "vii", "StudName" : "rahul" }
```

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" : "pratibha",
        "Grade" : "vii",
        "Hobbies" : "Chess"
}
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "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 ();

```
> db.Student.find({Hobbies:{$in:['Chess','Skating']}}).pretty();
{
        "_id" : 1,
        "StudName" : "pratibha",
        "Grade" : "vii",
        "Hobbies" : "Chess"
}
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating" }
```

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

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

```
> db.Student.find({StudName:/^r/}).pretty();
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating" }
```

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

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

```
> db.Student.find({StudName:/u/}).pretty();
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating" }
```

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

db.Student.count();

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

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({StudName:-1});
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating" }
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
{ "_id" : 2, "StudName" : "prathiksha", "Grade" : "viii", "Hobbies" : "cycling" }
```

6. 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 existing document.

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

```
> db.Student.save({StudName:"Prasansa",Grade:"viii"});
WriteResult({ "nInserted" : 1 })
> db.Student.find();
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
{ "_id" : 2, "StudName" : "prathiksha", "Grade" : "viii", "Hobbies" : "cycling" }
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating" }
{ "_id" : ObjectId("629e2c835e84878fe9a0aea0"), "StudName" : "Prasansa", "Grade" : "viii" }
```

7. Add a new field to existing Document:

db.Students.update({_id:3},{$set:{Location:”Network”}});

```
> db.Student.update({_id:3},{$set:{Location:"Network"}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find();
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
{ "_id" : 2, "StudName" : "prathiksha", "Grade" : "viii", "Hobbies" : "cycling" }
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating", "Location" : "Network" }
{ "_id" : ObjectId("629e2c835e84878fe9a0aea0"), "StudName" : "Prasansa", "Grade" : "viii" }
```

8. Remove the field in an existing Document

db.Students.update({_id:3},{$unset:{Location:”Network”}});

```
> db.Student.update({_id:3},{$unset:{Location:"Network"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find();
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
{ "_id" : 2, "StudName" : "prathiksha", "Grade" : "viii", "Hobbies" : "cycling" }
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating" }
{ "_id" : ObjectId("629e2c835e84878fe9a0aea0"), "StudName" : "Prasansa", "Grade" : "viii" }
```

9.Finding Document based on search criteria suppressing few fields

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

```
> db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});
{ "StudName" : "pratibha", "Grade" : "vii" }
```

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

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

```
> db.Student.find({Grade:{$ne:'vii'}});
{ "_id" : 2, "StudName" : "prathiksha", "Grade" : "viii", "Hobbies" : "cycling" }
{ "_id" : ObjectId("629e2c835e84878fe9a0aea0"), "StudName" : "Prasansa", "Grade" : "viii" }
```

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

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

```
> db.Student.find({StudName:/a$/});
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
{ "_id" : 2, "StudName" : "prathiksha", "Grade" : "viii", "Hobbies" : "cycling" }
{ "_id" : ObjectId("629e2c835e84878fe9a0aea0"), "StudName" : "Prasansa", "Grade" : "viii" }
```

12. to set a particular field value to NULL

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

```
> db.Student.update({_id:3},{$set:{Location:null}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find();
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
{ "_id" : 2, "StudName" : "prathiksha", "Grade" : "viii", "Hobbies" : "cycling" }
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating", "Location" : null }
{ "_id" : ObjectId("629e2c835e84878fe9a0aea0"), "StudName" : "Prasansa", "Grade" : "viii" }
```

13.Retrieve first 3 documents

db.Students.find({Grade:"VII"}).limit(3).pretty();

```
> db.Student.find({Grade:'vii'}).limit(3)
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating", "Location" : null }
> db.Student.find({Grade:'vii'}).limit(1)
{ "_id" : 1, "StudName" : "pratibha", "Grade" : "vii", "Hobbies" : "Chess" }
```

14.To Skip the 1 st two documents from the Students Collections

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

```
> db.Student.find().skip(2)
{ "_id" : 3, "Grade" : "vii", "StudName" : "rahul", "Hobbies" : "Skating", "Location" : null }
{ "_id" : ObjectId("629e2c835e84878fe9a0aea0"), "StudName" : "Prasansa", "Grade" : "viii" }
```

LAB 2

1.Create a keyspace by name Employee

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

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 employee_info(
            ... Emp_id int PRIMARY KEY,
            ... Emp_name text,
            ... Designation text,
            ... Date_of_joining timestamp,
            ... Salary double,
            ... Dept_name text
            ... );
```

3. Insert the values into the table in batch

```
cqlsh:employee> begin batch
           ... insert into employee_info (emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (1,'prathiksha','HR','2020-03-01',50000,'HR dept')
           ... apply batch;
cqlsh:employee> select * from Employee_info;

 emp_id | date_of_joining                 | dept_name | designation | emp_name   | salary
--------+---------------------------------+-----------+-------------+------------+--------
      1 | 2020-02-29 18:30:00.000000+0000 |   HR dept |          HR | prathiksha |  50000

(1 rows)
cqlsh:employee> begin batch
           ... insert into employee_info (emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (2,'pranav','Editor','2020-04-01',40000,'Marketing dept')
           ... insert into employee_info (emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (3,'rahul','Software Engineer','2020-05-01',60000,'technical')
           ... insert into employee_info (emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (4,'anuradha','Security Manager','2020-05-01',60000,'security')
           ... insert into employee_info (emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (5,'sonal','HR employee','2020-05-01',60000,'HR dept')
           ... apply batch;
```

```
 emp_id | date_of_joining                 | dept_name      | designation       | emp_name   | salary
--------+---------------------------------+----------------+-------------------+------------+--------
      5 | 2020-04-30 18:30:00.000000+0000 |        HR dept |       HR employee |      sonal |  60000
      1 | 2020-02-29 18:30:00.000000+0000 |        HR dept |                HR | prathiksha |  50000
      2 | 2020-03-31 18:30:00.000000+0000 | Marketing dept |            Editor |     pranav |  40000
      4 | 2020-04-30 18:30:00.000000+0000 |       security |  Security Manager |   anuradha |  60000
      3 | 2020-04-30 18:30:00.000000+0000 |      technical | Software Engineer |      rahul |  60000
```

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

```
cqlsh:employee> update employee_info set emp_name='prashansa',dept_name='Marketing' where emp_id=1;
cqlsh:employee> select * from Employee_info;
```

```
 emp_id | date_of_joining                 | dept_name      | designation       | emp_name   | salary
--------+---------------------------------+----------------+-------------------+------------+--------
      5 | 2020-04-30 18:30:00.000000+0000 |        HR dept |       HR employee |      sonal |  60000
      1 | 2020-02-29 18:30:00.000000+0000 |      Marketing |                HR | prashansa |  50000
      2 | 2020-03-31 18:30:00.000000+0000 | Marketing dept |            Editor |     pranav |  40000
      4 | 2020-04-30 18:30:00.000000+0000 |       security |  Security Manager |   anuradha |  60000
      3 | 2020-04-30 18:30:00.000000+0000 |      technical | Software Engineer |      rahul |  60000
```

5. Sort the details of Employee records based on salary

```
create table emp( id int, salary int, name text,primary key(id,salary) );
cqlsh:employee> begin batch
            ... insert into emp(id,salary,name)values (1,10000,'prathiksha')
            ... insert into emp(id,salary,name)values (2,10000,'pooja')
            ... insert into emp(id,salary,name)values (3,10000,'prema')
            ... insert into emp(id,salary,name)values (3,20000,'rahul')
            ... insert into emp(id,salary,name)values (4,30000,'raghu')
            ... apply batch
            ... ;
```

```
cqlsh:employee> paging off;
Disabled Query paging.
cqlsh:employee> select *from emp where id in (1,2,3,4) order by salary;

 id | salary | name
----+--------+------------
  1 |  10000 | prathiksha
  2 |  10000 |      pooja
  3 |  10000 |      prema
  3 |  20000 |      rahul
  4 |  30000 |      raghu
```

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

```
cqlsh:employee> alter table employee_info add projects text;
cqlsh:employee> describe table Employee_info;

CREATE TABLE employee.employee_info (
    emp_id int PRIMARY KEY,
    date_of_joining timestamp,
    dept_name text,
    designation text,
    emp_name text,
    projects text,
    salary double
```

7. Update the altered table to add project names.

```
cqlsh:employee> begin batch
            ... update employee_info set projects='abc' where emp_id=1
            ... update employee_info set projects='def' where emp_id=2
            ... update employee_info set projects='ghi' where emp_id=3
            ... update employee_info set projects='jkl' where emp_id=4
            ... update employee_info set projects='mno' where emp_id=5
            ... apply batch;
```

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

 emp_id | date_of_joining                 | dept_name      | designation       | emp_name  | projects | salary
--------+---------------------------------+----------------+-------------------+-----------+----------+--------
      5 | 2020-04-30 18:30:00.000000+0000 |        HR dept |       HR employee |     sonal |      mno |  60000
      1 | 2020-02-29 18:30:00.000000+0000 |      Marketing |                HR | prashansa |      abc |  50000
      2 | 2020-03-31 18:30:00.000000+0000 | Marketing dept |            Editor |    pranav |      def |  40000
      4 | 2020-04-30 18:30:00.000000+0000 |       security |  Security Manager |  anuradha |      jkl |  60000
      3 | 2020-04-30 18:30:00.000000+0000 |      technical | Software Engineer |     rahul |      ghi |  60000

(5 rows)
```

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 (19,'prithvi',senior_developer','2022-08-09',40000,'Developing') using TTL 50;
```

LAB 3

1 Create a key space by name Library

```
cqlsh> create keyspace lab2_library with replication={'class':'SimpleStrategy','replication_factor'
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
,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_n
'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_
'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 stu
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';
Using 7 child processes

Starting copy of lab2_library.library_info with columns [stud_id, stud_name, book_id, date_of_issue, count
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 LAB 3 I. CREATE

```
qlsh:lab2_library> create table library_info2(stud_id int,counter_value counter,stud_name text,book_id int
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, count
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>
```