VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
On

BIG DATA ANALYTICS

Submitted by

KAUSHIK POTLURI (1BM21CS089)

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

March 2024 to June 2024

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 **KAUSHIK POTLURI (1BM21CS089)**, 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 2023-24. The Lab report has been approved as it satisfies the academic requirements in respect of Big Data Analytics Lab - **(22CS6PCBDA)** work prescribed for the said degree.

Vikranth B.M
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

Index

Sl. No.	Experiment Title	Page No.
1	MongoDB- CRUD Demonstration	1
2	Perform the following DB operations using CassandraEmployee keyspace.	10
3	Perform the following DB operations using Cassandra-Librarykeyspace.	12
4	Screenshot of Hadoop installed	14
5	Execution of HDFS Commands for interaction with HadoopEnvironment.	15
6	Implement WordCount Program on Hadoop framework	17
7	From the following link extract the weather data https://github.com/tomwhite/hadoopbook/tree/master/input/ncdc/all Create a Map Reduce program to a) find average temperature for each year from NCDC data set. b) find the mean max temperature for every month	20
8	For a given Text file, Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.	23

Course outcomes:

CO1	Apply the concepts of NoSQL, Hadoop, Spark for a given task
CO2	Analyse data analytic techniques for a given problem
CO3	Conduct experiments using data analytics mechanisms for a given problem.

1. MongoDB- CRUD Demonstration

```
/DBMS_Demo?appName=mongosh+2.1.5
Using MongoDB:          7.0.7 (API Version 1)
Using Mongosh:          2.1.5
mongosh 2.2.1 is available for download: https://www.mongodb.com/try/download/sh
ell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-12eb3b-shard-0 [primary] DBMS_Demo> use MY_DB
switched to db MY_DB
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
```

I. Perform the following DB operations using MongoDB.

1. Create a database "Student" with the following attributes Rollno, Age, ContactNo, Email- Id.

2. Insert appropriate values

3. Write query to update Email-Id of a student with rollno 10.

4. . Replace the student name from "ABC" to "FEM" of rollno 11

Solution:

Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.createCollection("Student")
;
{ ok: 1 }
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Student.insert({_id:1,name:"Alice",rollNo:80,age:20,phNo:"9999988888",email:"alice@gmail.com"})
;
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{ acknowledged: true, insertedIds: { '0': 1 } }
Atlasatlas-12eb3b-shard-0 [primary] MY_DB>
db.Student.insert({_id:2,name:"Bob",rollNo:81,age:20,phno:"8888855555",email:"bob@gmail.com"});
{ acknowledged: true, insertedIds: { '0': 2 } } Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Student.insert({_id:3,name:"Cath",rollNo:82,age:21,phno:"8888877777",email:"cath@gmail.com"});
{acknowledged: true, insertedIds: { '0': 3 } }

Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Student.find();[
  {
    _id: 1,
name:
'Alice',
rollNo: 80,
    age: 20,   phNo:
'9999988888',
    email: 'alice@gmail.com'
  },

```
  {
    _id: 2,
name:
'Bob',
rollNo: 81,
age: 20,
    email:
    'bob@gmail.com',phNo:
    '8888855555'
  },
  {
    _id: 3,
name:
'Cath',
rollNo: 82,
age: 21,
    email:
    'cath@gmail.com',phNo:
    '8888877777'
  }
]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.Student.update({rollNo:80},
{$set:{email:"alice123@gmail.com"}});
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{  acknowledged:
true,  insertedId:
null,
matchedCount: 1,
modifiedCount: 1,
upsertedCount: 0
}
```

II. Perform the following DB operations using MongoDB.

1. Create a collection by name Customers with the following attributes.Cust_id, Acc_Bal, Acc_Type

2. Insert at least 5 values into the table

3. Write a query to display those records whose total account balance is greater than 1200 ofaccount type 'Z' for each customer_id.

4. Determine Minimum and Maximum account balance for each customer_i Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.createCollection(" Customers");
{ ok: 1 }
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Student.insertOne({custId:1,accBal:10000,accType:"saving"});
{ acknowledged:
true,
insertedId: ObjectId('660295b055dc2f3d86c4479f')
}
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Customers.insertOne({custId:1,accBal:10000,accType:"saving"});
{ acknowledged:
true,
insertedId: ObjectId('660295c155dc2f3d86c447a0')
}
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Customers.insertOne({custId:2,accBal:50000,accType:"current"});
{ acknowledged:
true,
insertedId: ObjectId('6602960055dc2f3d86c447a1')
}
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Customers.insertOne({custId:3,accBal:60000,accType:"current"});
{ acknowledged:
true,
insertedId: ObjectId('6602960e55dc2f3d86c447a2')
}
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Customers.insertOne({custId:4,accBal:20000,accType:"savings"});
{ acknowledged:
true,
insertedId: ObjectId('6602961c55dc2f3d86c447a3')
}
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
db.Customers.insertOne({custId:5,accBal:200000,accType:"current"});
{ acknowledged:
true,
insertedId: ObjectId('6602962955dc2f3d86c447a4')
}
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.Customers.find();
[

```
{
    _id: ObjectId('
660295c155dc2f3d86c447a0'),
    custId: 1,
accBal: 10000,
accType: 'savings'
  },
  {
    _id:
ObjectId('6602960055dc2f3d86c447a1'),custId:
2,                accBal: 50000,
    accType: 'current'
  },
  {
    _id:
ObjectId('6602960e55dc2f3d86c447a2'),custId:
3,                accBal: 60000,
    accType: 'current'
  },
  {
    _id:
ObjectId('6602961c55dc2f3d86c447a3'),custId:
4,                accBal: 20000,     accType:
'savings'
  },
  {
    _id:
ObjectId('6602962955dc2f3d86c447a4'),custId:
5,                accBal: 200000,
    accType: 'current'
  },
  {
    _id:
ObjectId('6602980955dc2f3d86c447a5'),custId:
1,                accBal: 30000,
    accType: 'savings'
  },
  {
    _id:
ObjectId('6602982f55dc2f3d86c447a6'),custId:
2,                accBal: 20000,     accType:
'current'
  }
]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.Customers.aggregate({$ match:{accType:'savings'}},{$group:
{_id:"$custId",TotalAccBal:{$sum:"$accBal"}}},{$match:{TotalAccBal:{$gt:20000}}});
[ { _id: 1, TotalAccBal: 40000 } ]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.Customers.aggregate({$group:{_id:"$custId",minAccBal:
{$min:"$accBal"},maxAccBal:{$max:"accB al"}}});
```

```
[
  { _id: 1, minAccBal: 10000, maxAccBal: 'accBal' },
  { _id: 4, minAccBal: 20000, maxAccBal: 'accBal' },
  { _id: 5, minAccBal: 200000, maxAccBal: 'accBal' },
  { _id: 2, minAccBal: 20000, maxAccBal: 'accBal' },
  { _id: 3, minAccBal: 60000, maxAccBal: 'accBal' }
]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.Customers.aggregate({$group:{_id:"$custId",minAccBal:
{$min:"$accBal"},maxAccBal:{$max:"$acc Bal"}}});
[
  { _id: 3, minAccBal: 60000, maxAccBal: 60000 },
  { _id: 4, minAccBal: 20000, maxAccBal: 20000 },
  { _id: 5, minAccBal: 200000, maxAccBal: 200000 },
  { _id: 2, minAccBal: 20000, maxAccBal: 50000 },
  { _id: 1, minAccBal: 10000, maxAccBal: 30000 }
]
```

```
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.Customers.find().sort({custId:1}).pretty();
[
  {
    _id: ObjectId('660295c155dc2f3d86c447a0'),
    custId: 1,
    accBal: 10000,
    accType: 'savings'
  },
  {
    _id: ObjectId('6602980955dc2f3d86c447a5'),
    custId: 1,
    accBal: 30000,
    accType: 'savings'
  },
  {
    _id: ObjectId('6602960055dc2f3d86c447a1'),
    custId: 2,
    accBal: 50000,
    accType: 'current'
  },
  {
    _id: ObjectId('6602982f55dc2f3d86c447a6'),
    custId: 2,
    accBal: 20000,
    accType: 'current'
  },
  {
    _id: ObjectId('6602960e55dc2f3d86c447a2'),
    custId: 3,
    accBal: 60000,
    accType: 'current'
  },
  {
    _id: ObjectId('6602961c55dc2f3d86c447a3'),
    custId: 4,
    accBal: 20000,
    accType: 'savings'
  },
  {
    _id: ObjectId('6602962955dc2f3d86c447a4'),
    custId: 5,
    accBal: 200000,
    accType: 'current'
  }
]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
```

```
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.Customers.find().sort({accBal:-1}).pretty();
[
  {
    _id: ObjectId('6602962955dc2f3d86c447a4'),
    custId: 5,
    accBal: 200000,
    accType: 'current'
  },
  {
    _id: ObjectId('6602960e55dc2f3d86c447a2'),
    custId: 3,
    accBal: 60000,
    accType: 'current'
  },
  {
    _id: ObjectId('6602960055dc2f3d86c447a1'),
    custId: 2,
    accBal: 50000,
    accType: 'current'
  },
  {
    _id: ObjectId('6602980955dc2f3d86c447a5'),
    custId: 1,
    accBal: 30000,
    accType: 'savings'
  },
  {
    _id: ObjectId('6602961c55dc2f3d86c447a3'),
    custId: 4,
    accBal: 20000,
    accType: 'savings'
  },
  {
    _id: ObjectId('6602982f55dc2f3d86c447a6'),
    custId: 2,
    accBal: 20000,
    accType: 'current'
  },
  {
    _id: ObjectId('660295c155dc2f3d86c447a0'),
    custId: 1,
    accBal: 10000,
    accType: 'savings'
  }
]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
```

III.Create a collection by the name blogPosts and it has 3 fields id, title and comments. In the collection the comments field is an array which consists of user details. Each collection consistsof two user details inside the comments array- user name and text

Demonstrate the following

1. **Adding an element into array**
2. **Display second element**
3. **Display size of the array**
4. **Display first two elements of the array**
5. **Update the document with id 4 and replace the element present in 1st index position of the array withanother array**

```
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.blogPosts.insertOne({_id:1, title: "Introduction to MongoDB", comments: [ { userName: "Alice", text: "Great article!" }, { userName: "Bob", text: "Looking fo
rward to more content." }] })
{ acknowledged: true, insertedId: 1 }
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.blogPosts.insertOne({_id:2, title: "Advanced MongoDB Techniques", comments: [ { userName: "Charlie", text: "Very informative." }, { userName: "David", text:
"Helped me a lot!" }] })
{ acknowledged: true, insertedId: 2 }
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.blogPosts.insertOne({_id:3, title: "MongoDB Performance Optimization", comments: [ { userName: "Eve", text: "I have a question." }, { userName: "Frank", text
: "This is exactly what I needed!" }] })
{ acknowledged: true, insertedId: 3 }
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.blogPosts.update( { _id:1 }, { $push: { comments: { userName: "John", text: "This is a new comment." } } } )
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
```

```
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.blogPosts.aggregate([
...   { $unwind: "$comments" }, // Unwind the comments array
...   { $project: { _id: 0, commentText: "$comments.text" } } // Project only the text field
... ])
[
  { commentText: 'Great article!' },
  { commentText: 'Looking forward to more content.' },
  { commentText: 'This is a new comment.' },
  { commentText: 'Very informative.' },
  { commentText: 'Helped me a lot!' },
  { commentText: 'I have a question.' },
  { commentText: 'This is exactly what I needed!' }
]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
```

```
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.blogPosts.aggregate([
...   { $project: { commentCount: { $size: "$comments" } } }
... ])
[
  { _id: 1, commentCount: 3 },
  { _id: 2, commentCount: 2 },
  { _id: 3, commentCount: 2 }
]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
```

```
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.blogPosts.aggregate([
...   { $project: { firstTwoComments: { $slice: ["$comments", 2] } } }
... ])
[
  {
    _id: 1,
    firstTwoComments: [
      { userName: 'Alice', text: 'Great article!' },
      { userName: 'Bob', text: 'Looking forward to more content.' }
    ]
  },
  {
    _id: 2,
    firstTwoComments: [
      { userName: 'Charlie', text: 'Very informative.' },
      { userName: 'David', text: 'Helped me a lot!' }
    ]
  },
  {
    _id: 3,
    firstTwoComments: [
      { userName: 'Eve', text: 'I have a question.' },
      { userName: 'Frank', text: 'This is exactly what I needed!' }
    ]
  }
]
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
```

```
Atlas atlas-12eb3b-shard-0 [primary] MY_DB> db.blogPosts.update( { _id: 3 }, { $set: { "comments.1": { userName: "Alice", text: "Replaced comment." } } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-12eb3b-shard-0 [primary] MY_DB>
```

```
cqlsh> create keyspace Employee with replication ={
   ... 'class':'SimpleStrategy',
   ... 'replication_factor':1
   ... };
cqlsh> use Employee
   ... ;
cqlsh:employee> create table Employee_info(
            ... Name text,
            ... Emp_Id int PRIMARY KEY,
            ... Designation text,
            ... DateofJoining timestamp,
            ... Department text
            ... ,Salary int
            ... );
```

```
cqlsh:employee> begin batch insert into Employee_info(Name,Emp_Id,Designation,Da
teofJoining,Department,Salary) values('Raj',121,'Tester','2012-03-29','Testing',
40000) insert into Employee_info(Name,Emp_Id,Designation,DateofJoining,Departmen
t,Salary) values('Anand',122,'Developer','2013-02-27','SE',60000) insert into Em
ployee_info(Name,Emp_Id,Designation,DateofJoining,Department,Salary) values('Sha
nthi',123,'Developer','2014-04-12','SE',80000) insert into Employee_info(Name,Em
p_Id,Designation,DateofJoining,Department,Salary) values('Priya',124,'Analyst','
2012-05-29','Data',50000) apply batch;
```

2. Perform the following DB operations using Cassandra.

1. Create a keyspace by name Employee
2. Create a column family by
nameEmployee-Info with
attributes Emp_Id Primary Key,
Emp_Name,
Designation, Date_of_Joining, Salary, Dept_Name
3. Insert the values into the table in batch
4. Update Employee name and Department of Emp-Id 121
5. Sort the details of Employee records based on salary
6. Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projectsdone by the corresponding Employee.
7. Update the altered table to add project names.
8.Create a TTL of 15 seconds to display the values of Employees.

```
cqlsh:employee> update Employee_info set Name='Rajesh' where Emp_Id=121;
cqlsh:employee> select * from Employee_info;

 emp_id | dateofjoining                   | department | designation | name    |
 salary
--------+---------------------------------+------------+-------------+---------+
--------
    123 | 2014-04-11 18:30:00.000000+0000 |         SE |   Developer | Shanthi |
  80000
    122 | 2013-02-26 18:30:00.000000+0000 |         SE |   Developer |   Anand |
  60000
    121 | 2012-03-28 18:30:00.000000+0000 |    Testing |      Tester |  Rajesh |
  40000
    124 | 2012-05-28 18:30:00.000000+0000 |       Data |     Analyst |   Priya |
  50000

(4 rows)
```

```
cqlsh:employee> update Employee_info set Department='Data' where Emp_Id=121;
cqlsh:employee> select * from Employee_info;

 emp_id | dateofjoining                   | department | designation | name    |
 salary
--------+---------------------------------+------------+-------------+---------+
--------
    123 | 2014-04-11 18:30:00.000000+0000 |         SE |   Developer | Shanthi |
  80000
    122 | 2013-02-26 18:30:00.000000+0000 |         SE |   Developer |   Anand |
  60000
    121 | 2012-03-28 18:30:00.000000+0000 |       Data |      Tester |  Rajesh |
  40000
    124 | 2012-05-28 18:30:00.000000+0000 |       Data |     Analyst |   Priya |
  50000

(4 rows)
```

1. Alter the schema of the table Employee Info to add a column Projects which stores a set of Projectsdone by the corresponding Employee.

```
cqlsh:employee> ALTER TABLE Employee_info ADD Projects text;
cqlsh:employee> select * from Employee_info;

 emp_id | dateofjoining                   | department | designation | name    | projects | salary
--------+---------------------------------+------------+-------------+---------+----------+--------
    123 | 2014-04-11 18:30:00.000000+0000 |         SE |   Developer | Shanthi |     null |  80000
    122 | 2013-02-26 18:30:00.000000+0000 |         SE |   Developer |   Anand |     null |  60000
    121 | 2012-03-28 18:30:00.000000+0000 |       Data |      Tester |  Rajesh |     null |  40000
    124 | 2012-05-28 18:30:00.000000+0000 |       Data |     Analyst |   Priya |     null |  50000
```

2. Update the altered table to add project names.

```
cqlsh:employee> begin batch insert into Employee_info(Emp_id,Projects) values(121,'App development') insert into Employee_info(Emp_id,Projects) values(122,'Web development')insert into Employee_info(Emp_i
d,Projects) values(123,'App development') insert into Employee_info(Emp_id,Projects) values(124,'Stock data') apply batch;
cqlsh:employee> select * from Employee_info;

 emp_id | dateofjoining                   | department | designation | name    | projects        | salary
--------+---------------------------------+------------+-------------+---------+-----------------+--------
    123 | 2014-04-11 18:30:00.000000+0000 |         SE |   Developer | Shanthi | App development |  80000
    122 | 2013-02-26 18:30:00.000000+0000 |         SE |   Developer |   Anand | Web development |  60000
    121 | 2012-03-28 18:30:00.000000+0000 |       Data |      Tester |  Rajesh | App development |  40000
    124 | 2012-05-28 18:30:00.000000+0000 |       Data |     Analyst |   Priya |      Stock data |  50000
```

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

```
cqlsh:employee> INSERT INTO Employee_Info (Emp_Id, name, designation, DateofJoining, Salary, Department,Projects)  VALUES (125, 'Emily Johnson', 'Analyst', '2023-03-01', 45000, 'Finance','Stock Data')  US
ING TTL 15;
cqlsh:employee> select * from Employee_info;

 emp_id | dateofjoining                   | department | designation | name          | projects        | salary
--------+---------------------------------+------------+-------------+---------------+-----------------+--------
    123 | 2014-04-11 18:30:00.000000+0000 |         SE |   Developer |       Shanthi | App development |  80000
    125 | 2023-02-28 18:30:00.000000+0000 |    Finance |     Analyst | Emily Johnson |      Stock Data |  45000
    122 | 2013-02-26 18:30:00.000000+0000 |         SE |   Developer |         Anand | Web development |  60000
    121 | 2012-03-28 18:30:00.000000+0000 |       Data |      Tester |        Rajesh | App development |  40000
    124 | 2012-05-28 18:30:00.000000+0000 |       Data |     Analyst |         Priya |      Stock data |  50000

(5 rows)
cqlsh:employee>
```

3. Perform the following DB operations using Cassandra.

1. Create a keyspace by name Library
2. Create a column family by name Library-Info with attributesStud_Id Primary Key, Counter_value of type Counter, Stud_Name, Book-Name, Book-Id, Date_of_issue
3. Insert the values into the table in batch
4. Display the details of the table created and increase the value of the counter
5. Write a query to show that a student with id 112 has taken a book "BDA" 2 times.
6. Export the created column to a csv file
7. Import a given csv dataset from local file system into Cassandra column family

1. Create a keyspace by name Library

```
cqlsh> CREATE KEYSPACE IF NOT EXISTS Library
   ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
```

2. Create a column family by name Library-Info with attributes Stud_Id Primary Key,Counter_value oftype Counter,Stud_Name, Book-Name, Book-Id, Date_of_issue.

```
cqlsh:library> CREATE TABLE libraryinfo (BookValue COUNTER,Stud_Id INT,Stud_Name TEXT,Book_Name TEXT,Book_Id TEXT,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

```
cqlsh:library> UPDATE libraryinfo SET bookvalue = bookvalue + 1 WHERE Stud_Id = 101 AND Stud_Name = 'Alice' AND Book_Name = 'History of India' AND Book_Id = '201' AND Date_of_issue = '2024-05-09';
cqlsh:library> UPDATE libraryinfo SET bookvalue = bookvalue + 1 WHERE Stud_Id = 102 AND Stud_Name = 'John' AND Book_Name = 'Python' AND Book_Id = '203' AND Date_of_issue = '2024-02-09';
cqlsh:library> UPDATE libraryinfo SET bookvalue = bookvalue + 1 WHERE Stud_Id = 103 AND Stud_Name = 'Priya' AND Book_Name = 'C Fundamentals' AND Book_Id = '206' AND Date_of_issue = '2024-02-18';
cqlsh:library> UPDATE libraryinfo SET bookvalue = bookvalue + 1 WHERE Stud_Id = 104 AND Stud_Name = 'Shreya' AND Book_Name = 'Mechanical Engineering' AND Book_Id = '205' AND Date_of_issue = '2024-01-18';
```

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

```
cqlsh:library> select * from libraryinfo;

 stud_id | stud_name | book_name             | book_id | date_of_issue                 | bookvalue
---------+-----------+-----------------------+---------+-------------------------------+----------
     104 |    Shreya | Mechanical Engineering |     205 | 2024-01-17 18:30:00.000000+0000 |         1
     102 |      John |                Python |     203 | 2024-02-08 18:30:00.000000+0000 |         1
     101 |     Alice |       History of India |     201 | 2024-05-08 18:30:00.000000+0000 |         1
     103 |     Priya |        C Fundamentals |     206 | 2024-02-17 18:30:00.000000+0000 |         1

(4 rows)
cqlsh:library> UPDATE libraryinfo SET bookvalue = bookvalue + 1 WHERE Stud_Id = 112 AND Stud_Name = 'Ashok' AND Book_Name = 'BDA' AND Book_Id = '210' AND Date_of_issue = '2023-08-18';
```

5. Write a query to show that a student with id 112 has taken a book "BDA" 2 time

```
(5 rows)
cqlsh:library> select * from libraryinfo where Stud_Id=112;

 stud_id | stud_name | book_name | book_id | date_of_issue                   | bookvalue
---------+-----------+-----------+---------+---------------------------------+-----------
     112 |     Ashok |       BDA |     210 | 2023-08-17 18:30:00.000000+0000 |         2

(1 rows)
```

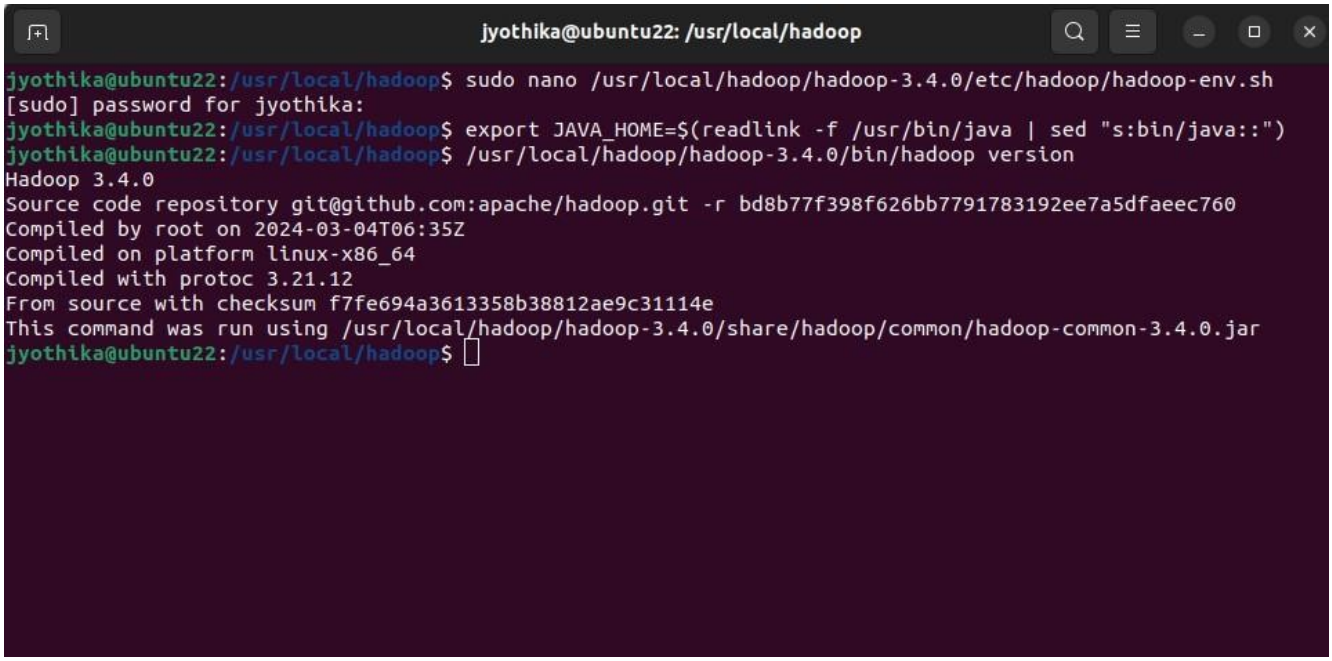
6. Export the created column to a csv file

```
(5 rows)
cqlsh:library> copy libraryinfo (bookvalue,stud_id,stud_name,book_name,book_id,date_of_issue) TO 'Documents:\library.csv';
Using 16 child processes

Starting copy of library.libraryinfo with columns [bookvalue, stud_id, stud_name, book_name, book_id, date_of_issue].
Processed: 5 rows; Rate:      76 rows/s; Avg. rate:      76 rows/s
5 rows exported to 1 files in 0.100 seconds.
cqlsh:library>
```

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

```
cqlsh:library> copy libraryinfo (bookvalue,stud_id,stud_name,book_name,book_id,date_of_issue) FROM 'Documents:\library.csv';
Using 16 child processes

Starting copy of library.libraryinfo with columns [bookvalue, stud_id, stud_name, book_name, book_id, date_of_issue].
```

4. Screenshot of Hadoop installed



5. Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)

to start hadoop services

start-all.sh jps

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ start-all.sh
```

make a new directory and display the dierctory contents

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -mkdir /bda_hadoop
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2024-05-14 14:10 /abc
drwxr-xr-x   - hadoop supergroup          0 2024-05-14 14:30 /bda_hadoop
```

use put to copy files from local to bda_hadoop folder

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Desktop/welcome.txt /bda_hadoop/file.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /bda_hadoop/file.txt
hadoop is an open source platform
```

use copyFromLocal to copy files from local to bda_hadoop folder

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -copyFromLocal /home/hadoop/Desktop/welcome.txt /bda_hadoop/file_cp_local.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /bda_hadoop/file_cp_local.txt
hadoop is an open source platform
```

use get to copy files from hadoop folder to local

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /bda_hadoop/file.txt /home/hadoop/Desktop/file_get.txt
```

make a merged file from hadoop and store it in local desktop

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -getmerge /bda_hadoop/file.txt /bda_hadoop/file_cp_local.txt /home/hadoop/Desktop/merged_file.txt
```

use getfacl to show the access rights

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -getfacl /bda_hadoop/
# file: /bda_hadoop
# owner: hadoop
# group: supergroup
user::rwx
group::r-x
other::r-x
```

move the contents of a directory to another directory in hadoop

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -mv /bda_hadoop /abc
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /abc
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2024-05-14 14:38 /abc/bda_hadoop
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -cp /abc/ /hadoop_lab
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /hadoop_lab
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2024-05-14 14:46 /hadoop_lab/bda_hadoop
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$
```

6. Implement Wordcount Program on Hadoop framework

Mapper Code: You have to copy paste this program into the WCMapper Java Class file.

```
// Importing libraries import java.io.IOException;
import org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.LongWritable; import
org.apache.hadoop.io.Text; import
org.apache.hadoop.mapred.MapReduceBase;
importorg.apache.hadoop.mapred.Mapper; import
org.apache.hadoop.mapred.OutputCollector; import
org.apache.hadoop.mapred.Reporter;
public class WCMapper extends MapReduceBase implements Mapper&lt;LongWritable, Text,
Text,

IntWritable&gt;
{ //Map function
public void map(LongWritable key, Text value, OutputCollector&lt;Text,
IntWritable&gt; output, Reporter rep) throws IOException

{
String line =
value.toString(); //Splitting the
line on spaces
for (String word : line.split(&quot; &quot;))
{
if (word.length() &gt; 0)
{
output.collect(new Text(word), new IntWritable(1));
} } } }
```

Reducer Code: You have to copy paste this program into the WCReducer Java Class file

```
// Importing libraries
importjava.io.IOException;
importjava.util.Iterator;
import org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text; import
org.apache.hadoop.mapred.MapReduceBase;
importorg.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer; import
org.apache.hadoop.mapred.Reporter;
public class WCReducer extends MapReduceBase implements Reducer&lt;Text,

IntWritable, Text, IntWritable&gt; {

// Reduce function
public void reduce(Text key, Iterator&lt;IntWritable&gt; value,
```

```
OutputCollector&lt;Text, IntWritable&gt;

output,Reporter rep) throws IOException

{
int count = 0;
// Counting the frequency of each words
while(value.hasNext())
{
IntWritable i = value.next(); count
+= i.get();
}
output.collect(key, new IntWritable(count));
} }
```

Driver Code: You have to copy paste this program into the WCDriver Java Class file.

```
// Importing libraries import java.io.IOException;
import org.apache.hadoop.conf.Configured; import
org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text; import
org.apache.hadoop.mapred.FileInputFormat; import
org.apache.hadoop.mapred.FileOutputFormat; import
org.apache.hadoop.mapred.JobClient; import
org.apache.hadoop.mapred.JobConf; import
org.apache.hadoop.util.Tool; import
org.apache.hadoop.util.ToolRunner;
public class WCDriver extends Configured implements Tool
{public int run(String args[]) throws IOException
{
if (args.length &lt; 2)
{
System.out.println(&quot;Please give valid inputs&quot;); return
-1;
}
JobConf conf = new JobConf(WCDriver.class);
FileInputFormat.setInputPaths(conf, new Path(args[0])); FileOutputFormat.setOutputPath(conf,
new Path(args[1]));
conf.setMapperClass(WCMapper.class);
conf.setReducerClass(WCReducer.class);
conf.setMapOutputKeyClass(Text.class);
conf.setMapOutputValueClass(IntWritable.class);
conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);
```

```
JobClient.runJob(conf);
return 0;
}
// Main Method
public static void main(String args[]) throws Exception
{
int exitCode = ToolRunner.run(new WCDriver(), args);
System.out.println(exitCode);
}
}
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bmscecse-HP-Elite-Tower-800-G9-Desktop-PC]
Starting resourcemanager
Starting nodemanagers
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ jps
5504 Jps
4130 NameNode
4903 ResourceManager
4296 DataNode
4540 SecondaryNameNode
5084 NodeManager
```

Output:

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -cat /output/part-00000
are     1
brother 1
family  1
hi      1
how     5
is      4
job     1
sister  1
you     1
your    4
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$
```

7. <u>From the following link extract the weather datahttps://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all Create a Map Reduce program to</u>

a) find average temperature for each year from NCDC data set.

b) find the mean max temperature for every month

```
import org.apache.hadoop.conf.Configuration; import
org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text; import
org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class AvgTemp {              public static void
main(Stringargs[])throws Exception {
                    Configuration conf = new
Configuration();Job job = Job.getInstance(conf, "Avg
Temp"); job.setJarByClass(AvgTemp.class);
job.setMapperClass(AvgTempMapper.class);
job.setCombinerClass(AvgTempReducer.class);
job.setReducerClass(AvgTempReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job,new Path(args[0]));
                    FileOutputFormat.setOutputPath(job, new Path(args[1]));
                    System.exit(job.waitForCompletion(true) ? 0:1);
          }
}


import java.io.*;
i         m         p         o         r         t
org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text;  import
org.apache.commons.lang.StringUtils;
import org.apache.hadoop.mapreduce.Mapper;

public class AvgTempMapper
                    extends Mapper<Object, Text, Text, IntWritable> {
```

```
        public void map(Object key, Text value, Context
                        context)throws IOException, InterruptedException{

                String[] line = value.toString().split(",");
                String datePart = line[1];
        String temp = line[10];
if(StringUtils.isNumeric(temp))
                        context.write(new Text(datePart), new IntWritable(Integer.parseInt(temp)));
        }

}

import java.io.*; import
org.apache.hadoop.io.IntWritable;
importorg.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class AvgTempReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

 public void reduce(Text key, Iterable<IntWritable> values, Context context)throws IOException,
InterruptedException {

  int sumTemps = 0;  int numItems =
0;
                for(IntWritable val : values)
                        { sumTemps +=
                        val.get();numItems +=
                        1;
                }
                context.write(key, new IntWritable(sumTemps/numItems));
        }

}
```

Output:

```
01      4
02      0
03      7
04      44
05      100
06      168
07      219
08      198
09      141
10      100
11      19
12      3
```

8. For a given Text file, Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.

```
import java.io.*;
importjava.util.*;
import org.apache.hadoop.io.LongWritable;
importorg.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class top_10_Movies_Mapper
        extends Mapper<Object, Text, Text, LongWritable>

        {private TreeMap<Long, String> tmap;

        @Override
        public void setup(Context context)
                throws IOException, InterruptedException
        {
                tmap = new TreeMap<Long, String>();
        }

        @Override
        public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException
        {

                // input data format => movie_name
                // no_of_views (tab separated)
                // we split the input data
                String[] tokens = value.toString().split("\t");

                String movie_name = tokens[0];
                long no_of_views = Long.parseLong(tokens[1]);

                // insert data into treeMap,
                // we want top 10 viewed movies
        // so we pass no_of_views as
keytmap.put(no_of_views,
movie_name);

                // we remove the first key-value
        // if it's size increases 10
if(tmap.size() > 10)
{ tmap.remove(tmap.firstKey());          }
```

```
        }

        @Override
        public void cleanup(Context context)
                throws IOException, InterruptedException
        {
                for (Map.Entry<Long, String>
                        entry :tmap.entrySet()) {

                        long count =
        entry.getKey();String name =
        entry.getValue();

                        context.write(new Text(name),
                                                new LongWritable(count));
                }
        }
} import java.io.IOException; import
java.util.Map; import java.util.TreeMap;
import
org.apache.hadoop.io.LongWritable;import
org.apache.hadoop.io.Text; import
org.apache.hadoop.mapreduce.Reducer;

public class top_10_Movies_Reducer
extendsReducer<Text, LongWritable,
LongWritable,
                                        Text>

        {private TreeMap<Long, String>

        tmap2;

        @Override
        public void setup(Context context)
                throws IOException, InterruptedException
        {
                tmap2 = new TreeMap<Long, String>();
        }

        @Override
        public void reduce(Text key,
                                        Iterable<LongWritable> values,
                                Context context)
                throws IOException, InterruptedException
        {

                // input data from mapper
                // key                          values
```

```
                // movie_name
                [ count ]String name =
                key.toString(); long count = 0;

                for (LongWritable val : values)
                        {count = val.get();
                }

                // insert data into treeMap,
                // we want top 10 viewed movies
        // so we pass count as key
                tmap2.put(count, name);

                // we remove the first key-value
        // if it's size increases
10if (tmap2.size() > 10) {
                        tmap2.remove(tmap2.firstKey());
                }
        }

        @Override
        public void cleanup(Context
context)throws IOException,
InterruptedException
        {

                for (Map.Entry<Long, String>
                        entry :tmap2.entrySet()) {

                        long count =
        entry.getKey();String name =
        entry.getValue();
context.write(new LongWritable(count),
                                                new Text(name));
                }
        }
}
import org.apache.hadoop.conf.Configuration; import
org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.LongWritable; import
org.apache.hadoop.io.Text; import
org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
```

```
public class Driver {

	public static void main(String[] args) throws Exception
	{
		Configuration conf = new
		Configuration();String[] otherArgs
			= new GenericOptionsParser(conf, args)
				.getRemainingArgs();


		// if less than two paths
// provided will show error
if(otherArgs.length & lt; 2) {
			System.err.println(" Error

								: please provide two paths
		System.exit(2);                                 & quot;);

		}

		Job job
			= Job.getInstance(conf, " top 10 & quot;);
job.setJarByClass(Driver.class);

		job.setMapperClass(top_10_Movies_Mapper.class);
		job.setReducerClass(top_10_Movies_Reducer.class);

		job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(LongWritable.class);

		job.setOutputKeyClass(LongWritable.class);job.setOutputValueClass(Text.class);

		FileInputFormat.addInputPath
	(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(
			job, new Path(otherArgs[1]));

		System.exit(job.waitForCompletion(true) ? 0 : 1);
	}
}
```

Input:

```
she is a nice person
hadoop is a distributed master slave framework
java is required for hadoop
```

Output:

```
a       2
distributed     1
for     1
framework       1
hadoop  2
is      3
java    1
master  1
nice    1
person  1
```