

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**  
“JnanaSangama”, Belgaum -590014, Karnataka.



**LAB REPORT**  
**on**

**Big Data Analytics (23CS6PCBDA)**

*Submitted by:*

**Rani Aishwarya H S (1BM22CS217)**

**Under the Guidance of  
Vikranth B.M.  
Assistant Professor, BMSCE**

*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 - 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 **Rani Aishwarya H S (1BM22CS217)**, 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 2024. The Lab report has been approved as it satisfies the academic requirements in respect of **Big Data Analytics –(23CS6PCBDA)** work prescribed for the said degree.

**Vikranth B.M.**  
Associate Professor  
Department of CSE  
BMSCE, Bengaluru

**Dr. Kavitha sooda**  
Professor and Head  
Department of CSE  
BMSCE, Bengaluru

## Table Of Contents

<b>Sl.no</b>	<b>Program details</b>	<b>Pg no</b>
1	MongoDB- CRUD Operations Demonstration (Practice and Self Study)	5-9
2	Perform the DB operations using Cassandra.	10-17
3	Perform the DB operations using Cassandra	18-20
4	Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)	21-23
5	Implement Wordcount program on Hadoop framework	24-27
6	a)Create a MapReduce program to find average temperature for each year from the NCDC data set. b) find the mean max temperature for every month.	28-30
7	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.	31-32
8	Write a Scala program to print numbers from 1 to 100 using a for loop.	33-34
9	Using RDD and FlatMap count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark.	35-36

## **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.

# Program 1

## MongoDB- CRUD Operations Demonstration (Practice and Self Study)

- Created a database named **myDB** and verified its existence.
- Created and dropped collections like **Student** and **Students**.
- Inserted student data into collections.
- Performed **upsert** to insert or update a student record.
- Used to find queries with various filters: by name, grade, hobbies, regex, etc.
- Retrieved specific fields while suppressing **\_id**.
- Counted total documents and documents with specific criteria.
- Sorted records in ascending and descending order.
- Imported data from a CSV file and exported data to a CSV file.
- Used **save()** to insert or replace documents.
- Added, removed, and set fields to **null** in documents.
- Retrieved limited records and skipped initial entries.
- Created a **food** collection with arrays and queried arrays by value, index, size, etc.
- Updated specific elements in an array.
- Practiced query optimizations using **\$in**, **\$all**, **\$ne**, **\$regex**, **\$slice**, and more.

## Observation:

04/03/25 (Lab - 01)

1. Create database in MongoDB and use my DB.
2. Create student collection.  

```
> db.createCollection("student");
1 document inserted.
```
3. Drop collection.  

```
db.Student.drop()
```
4. Create collection student and insert values  

```
db.Student.insert({id: 1, sname: "Rani", grade: 7, hobbies: ["Sketching"]})
db.Student.insert({id: 2, sname: "Chandana", grade: 9, hobbies: ["Scrolling"]})
db.Student.insert({id: 3, sname: "Pranam", grade: 7, hobbies: ["Writing"]})
```
5. Update Chandana's hobbies to chess  

```
> db.Student.update({id: 2, sname: "Chandana"}, {grade: 7, $set: {hobbies: ["Chess"]}}, {upsert: true});
1 document updated.
```
6. Find method  

```
db.Student.find({sname: "Rani"}).pretty();
[{"_id": 1, "sname": "Rani", "grade": 7, "hobbies": ["Sketching"]}]
```

Date / /  
Page / /

grade: 7  
hobbies: "Sketching"

> db.Student.find({sname: "Rani", grade: 7});

[{"\_id": 1, "sname": "Rani", "grade": 7}, {"\_id": 2, "sname": "Chandana", "grade": 9}, {"\_id": 3, "sname": "Pranam", "grade": 7}]

> db.Student.find({sname: "Chandana", grade: 9}).pretty();
[{"\_id": 2, "sname": "Chandana", "grade": 9, "hobbies": ["Scrolling"]}]

> db.Student.find({sname: "Rani", grade: 7}).pretty();
[{"\_id": 1, "sname": "Rani", "grade": 7, "hobbies": ["Sketching"]}]

→ Import csv file

```
> mongoimport -c student --db student --type csv --file student.csv
```

Ran in: bda lab\Bda load\student.csv  
DB: student

→ Collection: student

→ Headers: \_id, name, age, gender

→ Read from: file:///C:/Users/Admin/Desktop/bda\Sample\data.csv

4 documents imported successfully

→ Export csv file

```
> mongoexport --db student --collection student --type csv --file student.csv
```

Ran in: bda lab\Bda load\student.csv  
DB: student

→ Collection: student

→ Headers: \_id, name, age, gender

→ Read from: file:///C:/Users/Admin/Desktop/bda\Sample\data.csv

Exported 4 documents to "student.csv"

## Code with Output:

```
Atlas atlas-wanmtx-shard-0 [primary] Student> use Students
switched to db Students
Atlas atlas-wanmtx-shard-0 [primary] Students> show collections

Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.insertMany([
...   { "Rollno": 10, "Name": "John", "Age": 20, "ContactNo": "1234567890", "Email-Id": "john@example.com", "grade": "A", "hobby": "Reading" },
...   { "Rollno": 11, "Name": "Alice", "Age": 21, "ContactNo": "9876543210", "Email-Id": "alice@example.com", "grade": "B", "hobby": "Painting" },
...   { "Rollno": 12, "Name": "Bob", "Age": 22, "ContactNo": "2345678901", "Email-Id": "bob@example.com", "grade": "C", "hobby": "Cooking" },
...   { "Rollno": 13, "Name": "Eve", "Age": 23, "ContactNo": "3456789012", "Email-Id": "eve@example.com", "grade": "A" },
},
...   { "Rollno": 14, "Name": "Charlie", "Age": 24, "ContactNo": "4567890123", "Email-Id": "charlie@example.com", "hobby": "Gardening" }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("661ce9dc76a00ff8cc51dae1"),
    '1': ObjectId("661ce9dc76a00ff8cc51dae2"),
    '2': ObjectId("661ce9dc76a00ff8cc51dae3"),
    '3': ObjectId("661ce9dc76a00ff8cc51dae4"),
    '4': ObjectId("661ce9dc76a00ff8cc51dae5")
  }
}

Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.updateOne(
...   { "Rollno": 10 },
...   { $set: { "Email-Id": "john.doe@example.com" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.updateOne
...      { "Rollno": 11 },
...      { $set: { "Name": "Alicee" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.find({ "hobby": { $in: ["Chess", "Skating"] } })
[
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae1"),
    Rollno: 10,
    Name: 'John',
    Age: 20,
    ContactNo: '1234567890',
    'Email-Id': 'john.doe@example.com',
    grade: 'A',
    hobby: 'Reading'
  },
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae2"),
    Rollno: 11,
    Name: 'Alicee',
    Age: 21,
    ContactNo: '9876543210',
    'Email-Id': 'alice@example.com',
    grade: 'B',
    hobby: 'Painting'
  },
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae3"),
    Rollno: 12,
    Name: 'Bob',
    Age: 22,
    ContactNo: '2345678901',
    'Email-Id': 'bob@example.com',
    grade: 'C',
    hobby: 'Cooking'
  }
]
Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.find({}, { "Name": 1, "grade": 1, "$ifNull": ["$grade", "Not available"] }, { "_id": 0 })
[
  { Name: 'John', grade: 'A' },
  { Name: 'Alicee', grade: 'B' },
  { Name: 'Bob', grade: 'C' },
  { Name: 'Eve', grade: 'A' },
  { Name: 'Charlie', grade: 'Not available' }
]
```

## Program 2

**Perform the following DB operations using Cassandra.**

- a) Create a keyspace by name Employee

- b) Create a column family by name

Employee-Info with attributes Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name

- c) Insert the values into the table in batch

- d) Update Employee name and Department of Emp-Id 121

- e) Sort the details of Employee records based on salary

- f) Alter the schema of the table Employee\_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.

- g) Update the altered table to add project names.

- h) Create a TTL of 15 seconds to display the values ofEmployees.

### **Observation:**

11/03/25 LAB- 02. Date \_\_\_\_\_  
 Page \_\_\_\_\_

```

    1) → db.createCollection ("customers")
    → db.customers.insertMany([
      { Cust-id: 1, Acc-Bal: 1500, Acc-Type: 'Z' },
      { Cust-id: 2, Acc-Bal: 800, Acc-Type: 'A' },
      { Cust-id: 3, Acc-Bal: 1800, Acc-type: 'Z' },
      { Cust-id: 4, Acc-Bal: 1200, Acc-Type: 'B' },
      { Cust-id: 5, Acc-Bal: 1300, Acc-Type: 'Z' }
    ]);

    → db.Customers.find({ $gt: 1200 }, { Acc-Type: 'Z' });

    → db.Customers.aggregate([
      { $group: {
        _id: "$Cust-id",
        minBalance: { $min: "$Acc-Bal" },
        maxBalance: { $max: "$Acc-Bal" }
      } };
  
```

2) → db.createCollection("Products")  
db.createCollection("Cars")  
db.createCollection("Oranges")  
  
db.insert("Products", {  
 id: ObjectId("productID123"),  
 name: "Smartphone",  
 category: "Electronics",  
 price: 1999.99,  
 quantity: 50  
})  
  
db.insert("Products", {  
 id: ObjectId("productID124"),  
 name: "Laptop",  
 category: "Electronics",  
 price: 999.99,  
 quantity: 30  
})  
  
db.insert("Products", {  
 id: ObjectId("productID125"),  
 name: "Tshirt",  
 category: "Clothing",  
 price: 19.99,  
 quantity: 200  
})

→ db.Carts.insertMany([
   
 {
 user\_id: "123456789ghi...",
 items: [
 {
 product\_id: ObjectID("product1d1234"),
 quantity: 2,
 product\_id: ObjectID("product1d1234"),
 quantity: 3
 }
 ],
 user\_id: "123abc...",
 items: [
 {
 product\_id: ObjectID("product1d1234"),
 quantity: 1
 }
 ]
 },
 {
 user\_id: "123abc...",
 items: [
 {
 product\_id: ObjectID("product1d1234"),
 quantity: 1
 }
 ]
 }
 ]);
 → db.Orders.insertMany([
 {
 user\_id: "123abc...",
 products: [
 {
 product\_id: ObjectID("product1d1234"),
 quantity: 1,
 price: 199.99
 },
 {
 product\_id: ObjectID("product1d1234"),
 quantity: 2,
 price: 199.99
 }
 ],
 total\_price: 599.97,
 order\_date: ISODate("2023-03-11T07:00:00Z")
 }
 ]);

Date / /  
 Page / /

→ db.products.find({}).sort({name: 1});
   
 products: [
 {
 \_id: "789ghi...",
 name: "Laptop",
 price: 999.99,
 quantity: 10
 },
 {
 \_id: "123abc...",
 name: "Smartphone",
 price: 499.99,
 quantity: 10
 }
 ];
 total\_price: 999.99;
 order\_date: ISODate("2023-03-10T07:00:00Z");

1) ↴

→ db.products.find({}).sort({name: 1});
 products: [
 {
 \_id: "789ghi...",
 name: "Laptop",
 price: 999.99,
 quantity: 10
 },
 {
 \_id: "123abc...",
 name: "Smartphone",
 price: 499.99,
 quantity: 10
 }
 ];

→ db.products.find({category: "Electronics"});
 products: [
 {
 \_id: "789ghi...",
 name: "Laptop",
 price: 999.99,
 quantity: 10
 },
 {
 \_id: "123abc...",
 name: "Smartphone",
 price: 499.99,
 quantity: 10
 }
 ];

→ db.products.find({quantity: {\$gt: 6}});
 products: [
 {
 \_id: "789ghi...",
 name: "Laptop",
 price: 999.99,
 quantity: 10
 }
 ];

→ db.products.find({}).sort({price: 1});
 products: [
 {
 \_id: "123abc...",
 name: "Smartphone",
 price: 499.99,
 quantity: 10
 }
 ];

→ db.products.find({price: {\$lt: 100}});
 products: [
 {
 \_id: "123abc...",
 name: "Smartphone",
 price: 499.99,
 quantity: 10
 }
 ];

→ db.Carts.aggregate([
 {
 \$match: {user\_id: "123456789ghi..."},
 \$unwind: "\$items",
 \$lookup: {
 from: "products",
 localField: "items.product\_id",
 foreignField: ".\_id",
 as: "product\_info"
 }
 },
 {
 \$project: {
 "product\_info": 1,
 "item\_qty": "\$items.quantity"
 }
 }
 ]);

→ db.Orders.find({  
    $\{\text{user\_id} : "123abc\cdot\cdot\cdot12"\}$ ,  
    $\{\text{numbers}\}$   
 });

→ db.Orders.aggregate([  
    $\{\$match: \{\text{user\_id} : "123abc\cdot\cdot\cdot12"\}$ ,  
    $\{\$group: \{\text{id} : "queries\_iq"\}, \text{total\_price},$   
    $\{\$sum: "\text{total\_price}"\}$   
 ]);  
 ↳  $\{\text{id} : "queries\_iq", \text{total\_price} : 123\}$

↳ Aggregation;  
 ↳ db.products.aggregate([  
    $\{\$group: \{\text{id} : "\text{category}"\}, \text{total\_price},$   
    $\{\$sum: "\text{total\_price}"\}$   
 ]);  
 ↳ db.products.aggregate([  
    $\{\$group: \{\text{id} : "category"\}, \text{total\_price},$   
    $\{\$sum: "\$total\_price"\}, \{\$avg: "\$total\_price"\}$   
 ]);  
 ↳ db.products.find({  
    $\{\text{category} : "Electronics"\}$ ,  
    $\{\text{total\_price} > 100\}$   
 });

→ db.products.aggregate([  
    $\{\$group: \{\text{id} : null\}, \text{average\_price} : \{\$avg: "\text{price}"\}, \{\$sort: "\text{average\_price}"\}\}$   
 ]);  
 ↳  $\{\text{id} : null, \text{average\_price} : 123\}$

→ db.products.find({  
    $\{\text{category} : "Electronics"\}$ ,  
    $\{\text{average\_price} > 100\}$   
 });

→ db.products.find({  
    $\{\text{category} : "Electronics"\}$ ,  
    $\{\text{sort: "\text{price} : -1"\}\}$   
 });

→ db.orders.aggregate([  
    $\{\$group: \{\text{id} : "user_id"\}, \text{total\_price},$   
    $\{\$sum: "\text{total\_price}"\}$   
 ]);  
 ↳  $\{\text{id} : "user_id", \text{total\_price} : 123\}$

→ db.Orders.aggregate([  
    $\{\$group: \{\text{id} : "user_id"\}, \text{total\_price},$   
    $\{\$sum: "\text{total\_price}"\}$   
    $\{\$sort: "\text{total\_price} : -1\}\}$   
 ]);  
 ↳  $\{\text{id} : "user_id", \text{total\_price} : 123\}$

↳ db.Orders.aggregate([  
    $\{\$group: \{\text{id} : null, \text{average\_order\_price} : \{\$avg: "\$total\_price"\}\}$   
    $\{\$sort: "\text{average\_order\_price} : -1\}\}$   
 ]);  
 ↳  $\{\text{id} : null, \text{average\_order\_price} : 123\}$

↳ db.products.find({  
    $\{\text{category} : "Electronics"\}$ ,  
    $\{\text{average\_order\_price} > 100\}$   
 });

↳ db.products.find({  
    $\{\text{category} : "Electronics"\}$ ,  
    $\{\text{sort: "\text{average\_order\_price} : -1"\}\}$   
 });

↳ db.products.find({  
    $\{\text{category} : "Electronics"\}$ ,  
    $\{\text{sort: "\text{average\_order\_price} : -1"\}\}$   
 });

## Code with Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> mongosh
Current Mongosh Log ID: 67cff492cea8162357fa4213
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.4.0
Using MongoDB:     7.0.5
Using Mongosh:    2.4.0

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

The server generated these startup warnings when booting
2025-03-05T10:38:37.702+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
test> use lab2
switched to db lab2
lab2> db.createCollection("Customers")
{
  "ok": 1
}
lab2> db.Customers.insertMany([
...   { "Cust_Id": "C001", "Acc_Bal": 1500, "Acc_Type": "Z" },
...   { "Cust_Id": "C002", "Acc_Bal": 800, "Acc_Type": "Z" },
...   { "Cust_Id": "C003", "Acc_Bal": 2500, "Acc_Type": "A" },
...   { "Cust_Id": "C004", "Acc_Bal": 1800, "Acc_Type": "Z" },
...   { "Cust_Id": "C005", "Acc_Bal": 950, "Acc_Type": "B" }
... ]);
...
lab2> db.Customers.insertMany([
...   { "Cust_Id": "001", "Acc_Bal": 1500, "Acc_Type": "S" },
...   { "Cust_Id": "002", "Acc_Bal": 800, "Acc_Type": "S" },
...   { "Cust_Id": "003", "Acc_Bal": 2500, "Acc_Type": "C" },
...   { "Cust_Id": "004", "Acc_Bal": 1800, "Acc_Type": "C" },
...   { "Cust_Id": "005", "Acc_Bal": 950, "Acc_Type": "S" }
... ]);
{
  "acknowledged": true,
  "insertedIds": [
    "0": ObjectId('67cff561cea8162357fa4214'),
    "1": ObjectId('67cff561cea8162357fa4215'),
    "2": ObjectId('67cff561cea8162357fa4216'),
    "3": ObjectId('67cff561cea8162357fa4217'),
    "4": ObjectId('67cff561cea8162357fa4218')
  ]
}
lab2> db.find({Acc_Bal: {$gt: 1200}, Acc_Type: "S"})
TypeError: db.find is not a function
lab2> db.Customers.find({Acc_Bal: {$gt: 1200}, Acc_Type: "S"})
[
```

```
  price: 999,
  quantity: 30
]
lab2_2> db.Products.find({price: {$lte : 100}})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
    name: 'Jeans',
    category: 'Clothing',
    price: 40,
    quantity: 150
  }
]
lab2_2> db.Carts.find({user_id: "123abc"})
lab2_2> db.Carts.insertMany([
...   {
    user_id: "123abc",
    cart_items: [
      { product_id: "P001", quantity: 1 },
      { product_id: "P004", quantity: 2 }
    ]
  },
  {
    user_id: "789ghi",
    cart_items: [
      { product_id: "P003", quantity: 1 },
      { product_id: "P002", quantity: 1 }
    ]
  },
  {
    user_id: "456def",
    cart_items: [
      { product_id: "P005", quantity: 2 }
    ]
  },
  {
    user_id: "123abc",
    cart_items: [
      { product_id: "P004", quantity: 1 },
      { product_id: "P003", quantity: 1 }
    ]
  }
])
13
```

```

lab2_2> db.Carts.insertMany([
...   {
...     user_id: "123abc",
...     cart_items: [
...       { product_id: "P001", quantity: 1 },
...       { product_id: "P004", quantity: 2 }
...     ]
...   },
...   {
...     user_id: "789ghi",
...     cart_items: [
...       { product_id: "P003", quantity: 1 },
...       { product_id: "P002", quantity: 1 }
...     ]
...   },
...   {
...     user_id: "456def",
...     cart_items: [
...       { product_id: "P005", quantity: 2 }
...     ]
...   },
...   {
...     user_id: "123abc",
...     cart_items: [
...       { product_id: "P004", quantity: 1 },
...       { product_id: "P003", quantity: 1 }
...     ]
...   },
...   {
...     user_id: "789ghi",
...     cart_items: [
...       { product_id: "P002", quantity: 1 },
...       { product_id: "P005", quantity: 3 }
...     ]
...   }
... ],
... acknowledged: true,
insertedIds: {
  '0': ObjectId('67cffcab2f58b006f2fa4228'),
  '1': ObjectId('67cffcab2f58b006f2fa4229'),
  '2': ObjectId('67cffcab2f58b006f2fa422a'),
  '3': ObjectId('67cffcab2f58b006f2fa422b'),
  '4': ObjectId('67cffcab2f58b006f2fa422c')
}
]
lab2_2> db.Carts.find({user_id: "123abc"})
[
```

```

mongosh mongodb://127.0.0.1:27017
+ | - | X
lab2_2> db.Carts.find({user_id: "123abc"})
[
  {
    _id: ObjectId('67cffcab2f58b006f2fa4228'),
    user_id: '123abc',
    cart_items: [
      { product_id: 'P001', quantity: 1 },
      { product_id: 'P004', quantity: 2 }
    ]
  },
  {
    _id: ObjectId('67cffcab2f58b006f2fa422b'),
    user_id: '123abc',
    cart_items: [
      { product_id: 'P004', quantity: 1 },
      { product_id: 'P003', quantity: 1 }
    ]
  }
]
lab2_2> db.Orders.find({user_id: "123abc"})
[
  {
    _id: ObjectId('67cfffb6c2f58b006f2fa421e'),
    user_id: '123abc',
    order_items: [
      { product_id: 'P001', quantity: 1, price: 699 },
      { product_id: 'P004', quantity: 2, price: 25 }
    ],
    total_price: 749,
    order_date: ISODate('2025-03-01T00:00:00.000Z')
  },
  {
    _id: ObjectId('67cfffb6c2f58b006f2fa4221'),
    user_id: '123abc',
    order_items: [
      { product_id: 'P002', quantity: 1, price: 999 },
      { product_id: 'P004', quantity: 2, price: 25 }
    ],
    total_price: 1049,
    order_date: ISODate('2025-03-04T00:00:00.000Z')
  }
]
lab2_2> db.Orders.aggregate([
...   { $match: { user_id: "123abc..." } },
...   { $group: { _id: "$user_id", total_spent: { $sum: "$total_price" } } }
... ]);
...
lab2_2> db.Orders.aggregate([
...   { $match: { user_id: "123abc" } },
```

```
[1] lab2_2> db.Orders.aggregate([
...   { $match: { user_id: "123abc..." } },
...   { $group: { _id: "$user_id", total_spent: { $sum: "$total_price" } } }
... ]);

lab2_2> db.Orders.aggregate([
...   { $match: { user_id: "123abc" } },
...   { $group: { _id: "$user_id", total_spent: { $sum: "$total_price" } } }
... ],
[ { _id: '123abc', total_spent: 1798 } ]
lab2_2> db.Products.aggregate([
...   { $group: { _id: "$category", total_products: { $sum: 1 } } }
... ]);
[ { _id: 'Electronics', total_products: 3 },
{ _id: 'Clothing', total_products: 2 } ]
lab2_2> db.Products.aggregate([
...   { $group: { category: "$category", total_products: { $sum: 1 } } }
... ]);
MongoServerError[Location#0234]: The field 'category' must be an accumulator object
lab2_2> db.Products.aggregate([
...   { $group: { _id: "$category", total_price: { $sum: "price" } } }
... ],
[ { _id: 'Electronics', total_price: 0 },
{ _id: 'Clothing', total_price: 0 } ]
lab2_2> db.Products.aggregate([
...   { $group: { _id: "$category", total_price: { $sum: "price" } } }
... ]);
lab2_2> db.Products.aggregate([
...   { $group: { _id: "$category", total_price: { $sum: "$price" } } }
... ],
[ { _id: 'Electronics', total_price: 1848 },
{ _id: 'Clothing', total_price: 65 } ]
lab2_2> db.Products.aggregate([
...   { $group: { _id: NULL, avg_price: { $sum: "$price" } / { $sum: 1 } } }
... ]);
ReferenceError: NULL is not defined
lab2_2> db.Products.aggregate([
...   { $group: { _id: null, avg_price: { $sum: "$price" } / { $sum: 1 } } }
... ]);
MongoServerError[Location#0234]: The field 'avg_price' must be an accumulator object
lab2_2> db.Products.aggregate([
```

```
'3': ObjectId('67cfffb682f58b006f2fa421c'),
'4': ObjectId('67cfffb682f58b006f2fa421d')
}
lab2_2> db.Orders.insertMany([
... {
...     user_id: "123abc",
...     order_items: [
...         { product_id: "P001", quantity: 1, price: 699 },
...         { product_id: "P004", quantity: 2, price: 25 }
...     ],
...     total_price: 749,
...     order_date: new Date("2025-03-01")
... },
... {
...     user_id: "789ghi",
...     order_items: [
...         { product_id: "P003", quantity: 1, price: 150 },
...         { product_id: "P002", quantity: 1, price: 999 }
...     ],
...     total_price: 1149,
...     order_date: new Date("2025-03-02")
... },
... {
...     user_id: "456def",
...     order_items: [
...         { product_id: "P005", quantity: 1, price: 40 }
...     ],
...     total_price: 40,
...     order_date: new Date("2025-03-03")
... },
... {
...     user_id: "123abc",
...     order_items: [
...         { product_id: "P002", quantity: 1, price: 999 },
...         { product_id: "P004", quantity: 2, price: 25 }
...     ],
...     total_price: 1049,
...     order_date: new Date("2025-03-04")
... },
... {
...     user_id: "789ghi",
...     order_items: [
...         { product_id: "P001", quantity: 1, price: 699 }
...     ],
...     total_price: 699,
...     order_date: new Date("2025-03-05")
... }
... ])
...
```

```

... 1);
...
{
  acknowledged: true,
  insertedIds: [
    '0': ObjectId('67cfffb6c2f58b006f2fa421e'),
    '1': ObjectId('67cfffb6c2f58b006f2fa421f'),
    '2': ObjectId('67cfffb6c2f58b006f2fa4220'),
    '3': ObjectId('67cfffb6c2f58b006f2fa4221'),
    '4': ObjectId('67cfffb6c2f58b006f2fa4222')
  ]
}
lab2_2> db.UserCarts.insertMany([
...   {
...     user_id: "123abc",
...     cart_items: [
...       { product_id: "P001", quantity: 1 },
...       { product_id: "P004", quantity: 2 }
...     ]
...   },
...   {
...     user_id: "789ghi",
...     cart_items: [
...       { product_id: "P003", quantity: 1 },
...       { product_id: "P002", quantity: 1 }
...     ]
...   },
...   {
...     user_id: "456def",
...     cart_items: [
...       { product_id: "P005", quantity: 2 }
...     ]
...   },
...   {
...     user_id: "123abc",
...     cart_items: [
...       { product_id: "P004", quantity: 1 },
...       { product_id: "P003", quantity: 1 }
...     ]
...   },
...   {
...     user_id: "789ghi",
...     cart_items: [
...       { product_id: "P002", quantity: 1 },
...       { product_id: "P005", quantity: 3 }
...     ]
...   }
... ]);
...
{
  acknowledged: true,

```

```

...   ]
... );
...
{
  acknowledged: true,
  insertedIds: [
    '0': ObjectId('67cfffb712f58b006f2fa4223'),
    '1': ObjectId('67cfffb712f58b006f2fa4224'),
    '2': ObjectId('67cfffb712f58b006f2fa4225'),
    '3': ObjectId('67cfffb712f58b006f2fa4226'),
    '4': ObjectId('67cfffb712f58b006f2fa4227')
  ]
}
lab2_2> db.Products.find()
[
  {
    _id: ObjectId('67cffb682f58b006f2fa4219'),
    product_id: 'P001',
    name: 'Smartphone',
    category: 'Electronics',
    price: 699,
    quantity: 50
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421a'),
    product_id: 'P002',
    name: 'Laptop',
    category: 'Electronics',
    price: 999,
    quantity: 30
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421b'),
    product_id: 'P003',
    name: 'Headphones',
    category: 'Electronics',
    price: 150,
    quantity: 100
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',

```

```

> 1 | db.Products.find({quantity: {$gt: 0}}) ^
2 |
lab2_2> db.Products.find({quantity: {$gt: 0}})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa4219'),
    product_id: 'P001',
    name: 'Smartphone',
    category: 'Electronics',
    price: 699,
    quantity: 50
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421a'),
    product_id: 'P002',
    name: 'Laptop',
    category: 'Electronics',
    price: 999,
    quantity: 30
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421b'),
    product_id: 'P003',
    name: 'Headphones',
    category: 'Electronics',
    price: 150,
    quantity: 100
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
    name: 'Jeans',
    category: 'Clothing',
    price: 40,
    quantity: 150
  }
]
lab2_2> db.Products.find().sort({price: 1})
[
  {
    quantity: 150
  }
]

```

```

    quantity: 150
  }
]
lab2_2> db.Products.find().sort({price: 1})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
    name: 'Jeans',
    category: 'Clothing',
    price: 40,
    quantity: 150
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421b'),
    product_id: 'P003',
    name: 'Headphones',
    category: 'Electronics',
    price: 150,
    quantity: 100
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa4219'),
    product_id: 'P001',
    name: 'Smartphone',
    category: 'Electronics',
    price: 699,
    quantity: 50
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421a'),
    product_id: 'P002',
    name: 'Laptop',
    category: 'Electronics',
    price: 999,
    quantity: 30
  }
]
lab2_2> db.Products.find({price: {$lte: 100}})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  }
]
```

## Program 3

Perform the following DB operations using Cassandra.

a) Create a keyspace by name Library

b) 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

c) Insert the values into the table in batch

d) Display the details of the table created and increase the value of the counter e)

Write a query to show that a student with id 112 has taken a book "BDA" 2 times. f)

Export the created column to a csv file

g) Import a given csv dataset from local file system into Cassandra column family

**Observation:**

option 25 cqlsh  
cqlsh > create keyspace Students with Replication  
'class': 'SimpleStrategy', 'replication\_factor': 1;  
cqlsh > describe keyspace;  
→ students System auth, System schema  
System System distributed  
→ cqlsh > select \* from System.schema, Keyspace;  
→ cqlsh > USE Students;  
→ cqlsh:student> create table Students ( Roll\_no int Primary Key, studName text,  
DateOfJoining timestamp, lastExam\_percent double);  
→ cqlsh:student> describe tables;  
- students\_info  
→ cqlsh:student> describe Table students\_info;  
→ cqlsh:student> BEGIN BATCH  
| INSERT INTO Students (Roll\_no, studName,  
DateOfJoining, lastExam\_percent)  
VALUES (1, 'Ashu', '2012-03-12', 79.9)  
| INSERT INTO Students\_info (Roll\_no, StudName,  
DateOfJoining, lastExam\_percent)  
VALUES (2, 'Firan', '2012-03-12', 78.9)  
| INSERT INTO Students\_info (Roll\_no, StudName,  
DateOfJoining, lastExam\_percent)  
VALUES (3, 'Sarun', '2012-03-12', 78.9)

cqlsh:student> CREATE INDEX on Students\_info(Roll\_no);  
→ select \* from Students\_info where StudName='Ashu';  
roll\_no | dateofjoining | lastexam\_percent | studname  
1 | 2012-03-12 | 79.9 | Ashu  
→ select Roll\_no,studName from students\_info  
LIMIT 2;  
roll\_no | studName  
1 | Ashu  
2 | Smitha  
3 | Firan  
→ Select Roll\_no as "RN" from Students\_info;  
RN  
1  
2  
3  
→ update students\_info set StudName='David Sheen'  
where Roll\_no=2;  
→ update students\_info set Roll\_no=6 where Roll\_no=3;  
→ delete lastExamPercent from Students\_info where  
Roll\_no=2;  
→ select \* from students\_info  
David Sheen :- is null in Row 2

Date / /  
 Page / /

→ > Alter table students\_info ADD hobbies set (hobbies)

→ > Alter table students\_info add language list flag;

→ > Update students\_info  
 set hobbies = hobbies + 'cheer\_table\_fem' ;  
 where foll\_no = 1 ;

→ select \* from students\_info where foll\_no = 1 ;  
 | 2011-09 | doj | hobbies | language |  
 | 1 | 2012-03-11 | cheer\_table\_fem | null |

→ > Update students\_info  
 set (language = language + ('Hindi',  
 'English'))  
 where foll\_no = 1 ;

→ > Create table library\_book (rounds value =  
 (round\_value || 'book name') || 'By - data -'  
 'Analytics' And stud\_name = 'jed');

→ > Create table user\_login (user\_id, password)  
 key, password third);

→ > Insert into user\_login (user\_id, password)  
 values ('1', '1234567890123456');

→ > Select m (password) from user\_login where  
 user\_id = 1 ;

## Code with Output:

```
cqlsh> CREATE KEYSPACE Employee WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };
cqlsh> CREATE TABLE Employee.Employee_Info (
...     Emp_Id int,
...     Salary DECIMAL,
...     Emp_Name TEXT,
...     Designation TEXT,
...     Date_of_Joining DATE,
...     Dept_Name TEXT,
...     PRIMARY KEY (Emp_Id, Salary)
... ) WITH CLUSTERING ORDER BY (Salary ASC);
cqlsh> BEGIN BATCH
...     INSERT INTO Employee.Employee_Info (Emp_Id, Salary, Emp_Name, Designation, Date_of_Joining, Dept_Name) VALUES (121, 60000, 'John Doe', 'Developer', '2023-01-15', 'IT');
...     INSERT INTO Employee.Employee_Info (Emp_Id, Salary, Emp_Name, Designation, Date_of_Joining, Dept_Name) VALUES (122, 80000, 'Jane Smith', 'Manager', '2022-05-20', 'HR');
...     INSERT INTO Employee.Employee_Info (Emp_Id, Salary, Emp_Name, Designation, Date_of_Joining, Dept_Name) VALUES (123, 55000, 'Alice Johnson', 'Analyst', '2021-11-10', 'Finance');
...     APPLY BATCH;
cqlsh> UPDATE Employee.Employee_Info SET Emp_Name = 'Johnathan Doe', Dept_Name = 'Engineering' WHERE Emp_Id = 121 AND Salary = 60000;
cqlsh> SELECT * FROM Employee.Employee_Info WHERE Emp_Id = 121 ORDER BY Salary;
emp_id | salary | date_of_joining | dept_name | designation | emp_name
-----+-----+-----+-----+-----+
  121 |   60000 | 2023-01-15 | Engineering | Developer | Johnathan Doe
(1 rows)

cqlsh> ALTER TABLE Employee.Employee_Info ADD Projects SET<TEXT>;
cqlsh> UPDATE Employee.Employee_Info SET Projects = {'Project A', 'Project B'} WHERE Emp_Id = 121 AND Salary = 60000;
cqlsh> INSERT INTO Employee.Employee_Info (Emp_Id, Salary, Emp_Name, Designation, Date_of_Joining, Dept_Name) VALUES (124, 30000, 'Temp Employee', 'Intern', '2023-10-01', 'Temp Dept') USING TTL 15;
cqlsh> SELECT * FROM Employee.Employee_Info;
emp_id | salary | date_of_joining | dept_name | designation | emp_name | projects
-----+-----+-----+-----+-----+-----+-----+
  123 | 55000 | 2021-11-10 | Finance | Analyst | Alice Johnson | null
  122 | 80000 | 2022-05-20 | HR | Manager | Jane Smith | null
  121 | 60000 | 2023-01-15 | Engineering | Developer | Johnathan Doe | {'Project A', 'Project B'}
(3 rows)
cqlsh> █
```

## Program 4

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

#### Observation:

LAB - 04.

8/4/25

```

→ create a keyspace by name library
cqlsh > CREATE KEYSPACE library
replication = '{'class': 'SimpleStrategy', 'replication_factor': '1'};

→ curl cqlsh > library
library > create table library_info (stud_id
int, stud-name text, book_name text,
book_id text, date-of-issue date, primary
key (stud_id, book_id));
library > create table book_counter (stud_id
int, book-name text, counter-value
counter, primary key (stud_id, book-
name));
library > begin batch
insert into library_info(stud_id
stud-name, book-name, book_id, date-of-issue)
values ('112', 'alice', 'book1', '1.10.01', '2025-04-07');
insert into library_info(stud_id, stud-name
book-name, book_id, date-of-issue)
values ('113', 'bob', 'book2', '10.12.02', '2025-04-07');
apply batch;
library > select * from library_info;

```

stud_id	book_id	book-name	date	stud-name
113	b102	book1	2025-04-07	bob
112	10101	book2	2025-04-07	alice

HO - 04

update book\_counter set counter\_value = count
value + 1 where stud\_id = 112 and book\_name =
'bob';
update book\_counter set counter\_value =
counter
value + 1 where stud\_id = 112 and book\_name =
'book1';
update book\_counter set counter\_value =
counter
value + 1 where stud\_id = 113 and book\_
name = 'book2';
curl cqlsh -e "copy library.library\_info to
'library-info.csv' with HEADER = TRUE;"

**import**

cqlsh -e "copy library.library\_info from
'myfile.csv' with header = true;"

## Code with Output:

```
bmssecse@bmssecse-HP-Elite-Tower-800-G9-Desktop-PC: $ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.4 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE Students WITH REPLICATION={
    ... 'class':'SimpleStrategy','replication_factor':1};
cqlsh> DESCRIBE KEYSPACES
students  system_auth      system_schema  system_views
system    system_distributed  system_traces  system_virtual_schema

cqlsh> SELECT * FROM system.schema_keyspaces;
InvalidRequest: Error from server: code=2200 [Invalid query] message="table schema_keyspaces does not exist"
cqlsh> use Students;
cqlsh:Students> create table Students_info(Roll_No int Primary key,StudName text,DateOfJoining timestamp,last_exam_Percent double);
cqlsh:Students> describe tables;
students_info

cqlsh:Students> describe table students;
Table 'students' not found in keyspace 'students'
cqlsh:Students> describe table students_info;

CREATE TABLE students.students_info (
    roll_no int PRIMARY KEY,
    dateofjoining timestamp,
    last_exam_percent double,
    studname text
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

cqlsh:Students> Begin batch insert into Students_info(Roll_no, StudName, DateOfJoining, last_exam_Percent) values(1,'Sadhana', '2023-10-09', 98) insert into Students_info(Roll_no, StudName, DateOfJoining, last_exam_Percent) values(2,'Rutu', '2023-10-10', 97) insert into Students_info(Roll_no, StudName, DateOfJoining, last_exam_Percent) values(3,'Rachana', '2023-10-10', 97.5) apply batch;
cqlsh:Students> select * from students_info;
roll_no | dateofjoining | last_exam_percent | studname
-----+-----+-----+-----
  1 | 2023-10-09 18:30:00.000000+0000 |      98 | Sadhana
  2 | 2023-10-09 18:30:00.000000+0000 |      97 | Rutu
  3 | 2023-10-09 18:30:00.000000+0000 |  97.5 | Rachana

(4 rows)
cqlsh:Students> select * from students_info where roll_no in (1,2,3);
roll_no | dateofjoining | last_exam_percent | studname
-----+-----+-----+-----
  1 | 2023-10-09 18:30:00.000000+0000 |      98 | Sadhana
  2 | 2023-10-09 18:30:00.000000+0000 |      97 | Rutu
  3 | 2023-10-09 18:30:00.000000+0000 |  97.5 | Rachana

(3 rows)
cqlsh:Students> select * from students_info where Studname='Charu';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
cqlsh:Students> create index on Students_info(StudName);
cqlsh:Students> select * from students_info where Studname='Charu';
roll_no | dateofjoining | last_exam_percent | studname
-----+-----+-----+-----
  4 | 2023-10-05 18:30:00.000000+0000 |      96.5 | Charu

(1 rows)
cqlsh:Students> select Roll_no,StudName from students_info LIMIT 2;
```

```
(4 rows)
cqlsh:students> select * from students_info where roll_no in (1,2,3);
roll_no | dateofjoining           | last_exam_percent | studname
-----+-----+-----+-----+
  1 | 2023-10-08 18:30:00.000000+0000 |      98 | Sadhana
  2 | 2023-10-09 18:30:00.000000+0000 |      97 | Rutu
  3 | 2023-10-09 18:30:00.000000+0000 |    97.5 | Rachana

(3 rows)
cqlsh:students> select * from students_info where Studname='Charu';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query, espte the performance unpredictability, use ALLOW FILTERING"
cqlsh:students> create index on Students_info(StudName);
cqlsh:students> select * from students_info where Studname='Charu';
roll_no | dateofjoining           | last_exam_percent | studname
-----+-----+-----+-----+
  4 | 2023-10-05 18:30:00.000000+0000 |     96.5 | Charu

(1 rows)
cqlsh:students> select Roll_no,StudName from students_info LIMIT 2;
roll_no | studname
-----+-----
  1 | Sadhana
  2 | Rutu

(2 rows)
cqlsh:students> SELECT Roll_no as "USN" from Students_info;
USN
-----
  1
  2
  4
  3
```

## Program 5

### Implement Wordcount program on Hadoop framework

#### Observation:

15/04/2023 CAB - 05.

```
$ start-all.sh  
$ hdfs dfs - mkdir /bda-hadoop  
$ hadoop fs -ls /  
found 1 items  
$ hdfs dfs -put /home/hadoop/Desktop/  
newfile.txt /bda-hadoop/file.txt  
$ hdfs dfs -cat /bda-hadoop/file.txt  
hello  
$ hdfs dfs -copyFromLocal /home/hadoop/  
Desktop/newfile.txt /bda-hadoop/file-copy.local  
$ hdfs dfs -cat /bda-hadoop/file-copy.local  
hello  
$ hdfs dfs -get /bda-hadoop/file.txt/  
home/hadoop/Downloads/downloaded-file.txt  
$ hdfs dfs -get merge /bda-hadoop/file.txt/  
bda-hadoop/file-copy.local.txt /home/  
hadoop/Downloads/downloaded-file.txt  
hadoop fs - get face /bda-hadoop  
# file : /bda-hadoop  
# owner : hadoop  
# group : supergroup  
# user : root
```

15/04/2023 CAB - 05.

```
group: : r-s  
other: : r-w  
hdfs dfs - copyToLocal /bda-hadoop/file.txt  
/home/hadoop/Desktop  
$ hadoop fs -mv /bda-hadoop/lab  
$ hadoop fs -ls /lab  
found 2 items  
$ hadoop fs -cp /lab /hadoop-lab  
✓
```

Ubuntu → Eclipse

→ create JAR file & input text file

→ create three java classes into the project  
Named WCDriver, WCMapper, WCReducer.

→ import java.io.IOException;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapred.Reducer;

public class WCMapper extends MapReduceBase  
implements Mapper<LongWritable, Text, IntWritable> {  
 public void map(LongWritable key, Text value,  
 OutputCollector<Text, IntWritable> output, Reporter  
 report) throws IOException {  
 String line = value.toString();  
 for (String word : line.split(" ")){  
 if (word.length() > 0){  
 output.collect(new Text(word),  
 new IntWritable(1));  
 }  
 }  
 }  
}

→ import java.io.IOException;  
import java.util.List;

public class WCReducer extends MapReduceBase  
implements Reducer<Text, IntWritable, Text,  
IntWritable> {  
 public void reduce(Text key, IntWritable  
 value, OutputCollector<Text, IntWritable> output,  
 Reporter report) throws IOException {  
 int count = 0;  
 while (value != null) {  
 count += value.get();  
 value = value.next();  
 }  
 output.collect(key, new IntWritable(count));  
 }  
}

→ public class WCDriver extends Configured implements  
Tool {  
 public void run(String args[]) throws IOException {  
 if (args.length < 2) {  
 System.out.println("please give valid  
 inputs");  
 return -1;  
 }  
 JobConf conf = new JobConf(WCReducer.  
 class);  
 FileInputFormat.setInputPath(conf,  
 new Path(args[0]));  
 FileOutputFormat.setOutputPath(conf,  
 new Path(args[1]));  
 conf.setMapperClass(WCMapper.class);  
 conf.setMapOutputKeyClass(Text.class);  
 conf.setMapOutputValueClass(IntWritable.class);  
 }  
}

conf.setOutputValueClass(Iterit.class);  
Job client runJob(conf);

return D;

public static void main(String args[]) throws  
Exception {

inf.outfile = ToolRunner.num(new  
WordDriver(), args);

System.out.print("n" + inf.outfile);

if (inf.outfile != null) open();

try {  
String s = inf.outfile.readLine();  
if (s != null) {  
String[] ss = s.split("\\s+");  
if (ss.length > 1) {  
String word = ss[0];  
int count = 1;  
for (int i = 1; i < ss.length; i++) {  
if (ss[i].equals(word)) {  
count++;  
} else {  
break; // word found  
}}  
if (word.equals("\*")) {  
System.out.println(count);  
} else {  
System.out.println(word + " " + count);  
}  
}  
}  
}

} else {  
System.out.println("No file specified");  
}

if (inf.outfile != null) inf.outfile.close();  
}

if (inf.outfile != null) close();  
}

if (inf.outfile != null) inf.outfile.close();  
}

if (inf.outfile != null) close();  
}

## Code with Output:

```
FILE: Number of little operations=0
HDFS: Number of bytes read=162
HDFS: Number of bytes written=67
HDFS: Number of read operations=15
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
    Map input records=5
    Map output records=20
    Map output bytes=161
    Map output materialized bytes=207
    Input split bytes=86
    Combine input records=0
    Combine output records=0
    Reduce input groups=10
    Reduce shuffle bytes=207
    Reduce input records=20
    Reduce output records=10
    Spilled Records=40
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=0
    Total committed heap usage (bytes)=1052770304
Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
File Input Format Counters
    Bytes Read=81
File Output Format Counters
    Bytes Written=67
0
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hadoop fs -ls /rgs/output/
Found 2 items
-rw-r--r--  1 hadoop supergroup          0 2025-05-06 15:50 /rgs/output/_SUCCESS
-rw-r--r--  1 hadoop supergroup       67 2025-05-06 15:50 /rgs/output/part-00000
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hadoop fs -cat /rgs/output/part-00000
are    1
brother 1
family  1
hi      1
how     5
is      4
job     1
sister  1
ur      4
you     1
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ ~
bash: /home/hadoop: Is a directory
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$
```

## Program 6

**From the following link extract the weather data**

<https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all>

a) Create a MapReduce program to find average temperature for each year from the NCDC data set.

b) find the mean max temperature for every month

## **Observation:**

Date \_\_\_\_\_  
Page \_\_\_\_\_

LAB - 06.

Create a Map Reduce program.

Q) Find average temp for each year from CDC dataset package temp;

package temp;

```
public class AverageDriver {
    public static void main (String [] args) throws
        exception {
        if (args.length != 2) {
            System.out.println ("please Enter the input
                and output");
            System.exit (-1);
        }
        Job job = new Job ();
        job.setJobByClass (AverageDriver.class);
        job.setJobName ("Max Temp");
        job.setMapperClass (AverageMapper.class);
        job.setReducerClass (AverageReducer.class);
        job.setOutputKeyClass (Text.class);
        System.exit (1);
    }
}
```

public class AverageMapper extends Mapper < Long  
Writable > {

int temp;

String line = value.toString();

String year = line.substring (15, 19);

if (line.charAt (87) == '+')

temp = Integer.parseInt (line.substring (18, 21));

} else {

temp = Integer.parseInt (line.substring (87, 21));

Date / /  
 Page \_\_\_\_\_

```

  (defn (month=datetime-stop-time (date-suo,
  "4 %o / - %m - %d").month
  print ("{} & monthly & {}")
  except Exception: raise
  continue
  )
  reduce:
  import sys
  current = None
  total = 0.0
  count = 0
  for line in sys.stdin:
    line = line.strip()
    if not line:
      continue
    month, temp = line.split(" ")
    temp = float(temp)
    if current == month:
      total += temp
      count += 1
    else:
      if current is not None:
        mean = total / count
        print ("{} & current month {} & mean: {}")
      current = month
      total = temp
      count = 1
    if current is not None:
      mean = total / count
      print ("{} & current month {} & mean: {}")
  )
  
```

## Code with Output:

### a) Average temperature

```
Map-Reduce Framework
    Map input records=6
    Map output records=6
    Map output bytes=60
    Map output materialized bytes=78
    Input split bytes=84
    Combine input records=0
    Combine output records=0
    Reduce input groups=3
    Reduce shuffle bytes=78
    Reduce input records=6
    Reduce output records=1
    Spilled Records=12
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=18
    Total committed heap usage (bytes)=403701760
Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
File Input Format Counters
    Bytes Read=60
File Output Format Counters
    Bytes Written=25
2025-05-24 17:20:45,936 INFO streaming.StreamJob: Output directory: /bda/out1
prajwal@PrajwalDevice:~$ hdfs dfs -cat /bda/out1/part-00000
Mean Temperature: 31.18
prajwal@PrajwalDevice:~$
```

### b) Maximum temperature

```
Map-Reduce Framework
    Map input records=6
    Map output records=6
    Map output bytes=60
    Map output materialized bytes=78
    Input split bytes=84
    Combine input records=0
    Combine output records=0
    Reduce input groups=3
    Reduce shuffle bytes=78
    Reduce input records=6
    Reduce output records=1
    Spilled Records=12
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=15
    Total committed heap usage (bytes)=403701760
Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
File Input Format Counters
    Bytes Read=60
File Output Format Counters
    Bytes Written=24
2025-05-24 17:23:40,195 INFO streaming.StreamJob: Output directory: /bda/out2
prajwal@PrajwalDevice:~$ hdfs dfs -cat /bda/out2/part-00000
Max Temperature: 33.50
```

## Program 7

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.

Observation:

Lah-09  
Given a text file, create a Map Reduce program to sort the content in an alphabetic order listing only top 10 max occurrences of words.

```
import sys
def mapper():
    for line in sys.stdin:
        line = line.strip()
        if not line:
            continue
        words = line.split()
        for word in words:
            print(word + "\t1")
def reducer():
    current_word = None
    current_count = 0
    for line in line.split():
        if not line:
            continue
        try:
            count = int(line)
        except ValueError:
            continue
        if current_word == word:
            current_count += count
        else:
            if current_word:
                print(current_word + "\t" + str(current_count))
            current_word = word
            current_count = count
    if current_word:
        print(current_word + "\t" + str(current_count))
```

if current == word
print(current\_word)
if --name == main
reducer().  
else:
print(word + "\t" + str(count))

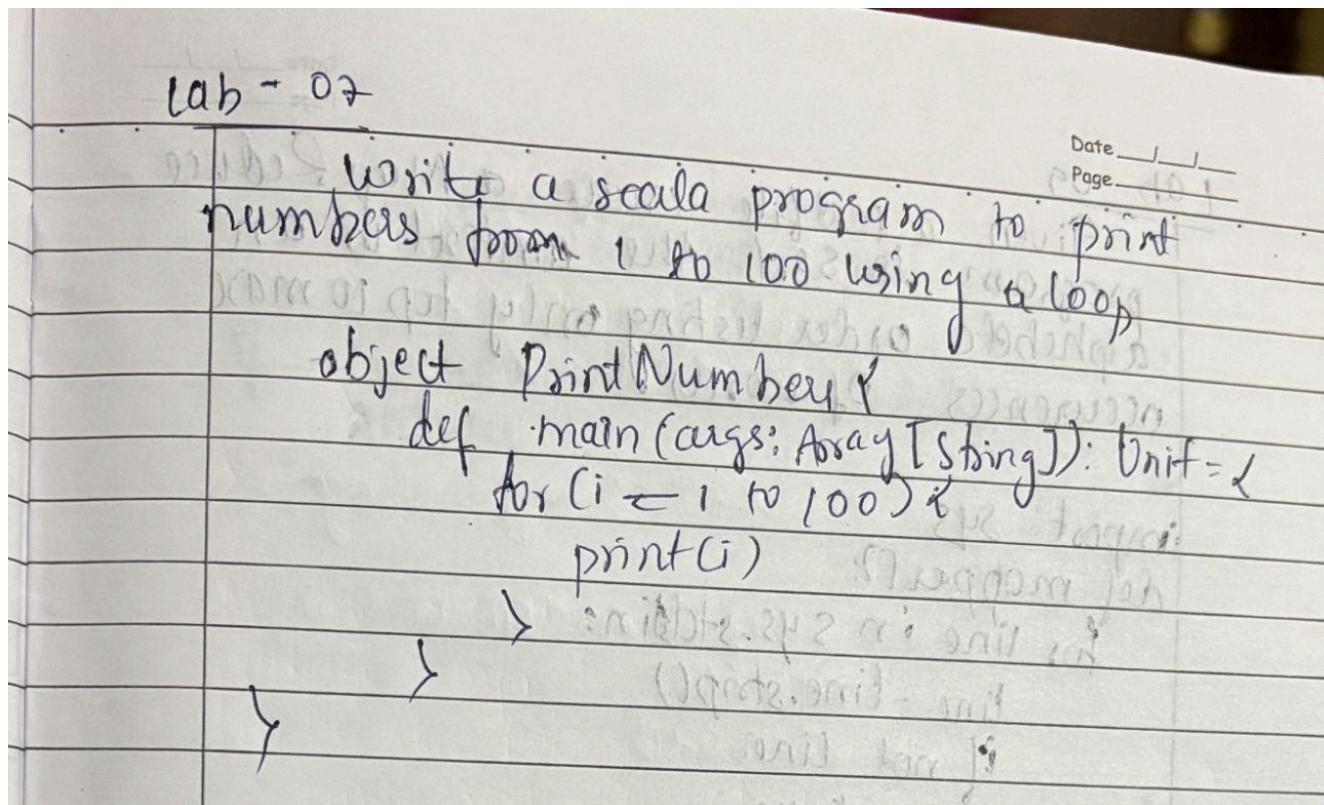
## Code with Output:

```
COMBINE output records=0
Reduce input groups=18
Reduce shuffle bytes=239
Reduce input records=25
Reduce output records=10
Spilled Records=50
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=15
Total committed heap usage (bytes)=421527552
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=137
File Output Format Counters
Bytes Written=72
2025-05-24 17:25:13,559 INFO streaming.StreamJob: Output directory: /bda/out3
prajwal@PrajwalDevice:~$ hdfs dfs -cat /bda/out3/part-00000
the      3
foxes    2
hares    2
jumps    2
quick    2
than     2
are      1
blue     1
brown    1
dog      1
```

## Program 8

Write a Scala program to print numbers from 1 to 100 using for loop.

**Observation:**



## Code with Output:

```
scala> for(i <- 0 to 100){
    |   println(i)
    |
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
```

## Program 9

Using RDD and FlatMap count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark

Observation:

Lab - 07

Date \_\_\_\_\_  
Page \_\_\_\_\_ 101

Write a scala program to print numbers from 1 to 100 using a loop

```
object PrintNumber {
    def main(args: Array[String]): Unit = {
        for (i ← 1 to 100) {
            print(i)
        }
    }
}
```

LAB - 08 : Using RDD & flatMap count how many times each word appears in a file and write out list of words whose count is strictly greater than 4 using spark

```
from pyspark import sparkContext
sc = sparkContext("local", "WordCountApp")
textRDD = sc.textFile("input.txt")
wordsRDD = textRDD.flatMap(lambda line: line.split())
pairingRDD = wordsRDD.map(lambda word: (word, 1))
wordCount = pairingRDD.filter(lambda x: x[1] > 4)
for word, count in wordCount.collect():
    print(f'{word}: {count}')
filteredRDD.saveAsTextFile("output-directory")
```

## Code with Output: