Adichunchanagiri Institute of Technology

Jyothinagara, Chikkamagaluru – 577102

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

(ACADEMIC YEAR 2025-26)

BIG DATA ANALYTICS LABORATORY MANUAL

SUB CODE: BIS701

SEMESTER: VII

As per Choice Based Credit System (CBCS)
& Outcome Based Education (OBE)

Prepared By

Dr. Anjali B V
Lab Instructor

Approved by

Dr. Sampath S
HOD, IS&E, AIT

INSTITUTIONAL VISION and MISSION

Vision

- To develop Adichunchanagiri Institute of Technology as a center of excellence and to strive for continuous improvement of technical education and human resource advancement.

Mission

- To achieve Excellence in Education, Entrepreneurship, and Innovation by producing Engineers with high Ethical Standards, Integrity, and Credibility.

DEPARTMENT VISION and MISSION

Vision

- To be recognized as a center of excellence in information technology and allied area with quality learning and research environment.

Mission

- M1: Provide intellectual & professional leadership in ethical and social areas pertaining to information in contemporary society.
- M2: Advancing the state of knowledge of information studies through research and development.
- M3: Providing platform to discuss cutting edge technologies.

Department of Information Science & Engineering

Program Outcomes (POs)

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)

Program Specific Outcomes (PSOs)

PSO1: Ability to understand and analyze Information Technology problems

PSO2: Design, Implement and Maintain the Information Systems that fulfill the current needs of the industry and society

Course outcomes (Cos)

CO1: Identify and list various Big Data concepts, tools and applications.

CO2: Develop programs using HADOOP framework..

CO3: Illustrate the concepts of NoSQL using MongoDB for Big Data

CO4 Use Hadoop Cluster to deploy Map Reduce jobs, PIG and HIVE programs.

C05: Concepts of Spark Programs and Analyze the given data set and identify deep insights from the data set

CO-PO Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11
CO1	3	2	1	1	2						2
CO2	3	3	2	2	3						1
CO3	3	2	2	2	3						2
CO4	3	3	3	2	3						2
CO5	3	3	3	3	3						2

CO-PSO Mapping

	PSO1	PSO2
CO1	3	2
CO2	3	3
CO3	3	3
CO4	3	3
CO5	3	3

Conduction of Practical Examination:

- 15 marks for the conduction of the experiment and preparation of laboratory record, and 10 marks for the test to be conducted after the completion of all the laboratory sessions.

- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.

- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks

- The laboratory test (duration 02/03 hours) after completion of all the experiments shall be conducted for 50 marks and scaled down to 10 marks.

- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for 25 marks. The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

PRACTICAL COMPONENT OF IPCC

Sl.NO	Experiments (Java/Python/R)
1	Install Hadoop and Implement the following file management tasks in Hadoop: Adding files and directories Retrieving files Deleting files and directories. Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.
2	Develop a MapReduce program to implement Matrix Multiplication
3	Develop a Map Reduce program that mines weather data and displays appropriate messages indicating the weather conditions of the day
4	Develop a MapReduce program to find the tags associated with each movie by analyzing movie lens data.
5	Implement Functions: Count – Sort – Limit – Skip – Aggregate using MongoDB
6	Write Pig Latin scripts to sort, group, join, project, and filter the data.
7	Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.
8	Implement a word count program in Hadoop and Spark.
9	Use CDH (Cloudera Distribution for Hadoop) and HUE (Hadoop User Interface) to analyze data and generate reports for sample datasets

Experiment 1

AIM: Install Hadoop and implement file management tasks in Hadoop Adding Files and Directories , Retrieving Files , Delete files and directories

Step 1:

Set up JDK

- ⇨ Download jdk : https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html
- ⇨ Install JDK in folder C:\Java\jdk and C:\Java\jre
- ⇨ Set Envirnoment Variable
  - **User Var** JAVA_HOME:C:\Java\jdk
  - **System Var** C:\Java\jdk
    C:\Java\jdk\bin
- ⇨ Check in cmd prompt : java
  (open as Admin) Java –version

Step 2:

Set up Hadoop

- ⇨ Download Hadoop : https://archive.apache.org/dist/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz
- ⇨ **Extract** to Path C:\Hadoop
- ⇨ **Set jdk path** in file under path in C:\Hadoop\etc\hadoop ➔ **hadoop-env.cmd**
- ⇨ **Create** new folders C:\Hadoop**data** with datanode and namenode
- ⇨ **Set up** the Configuration in 4 xml under the folder C:\Hadoop\etc\hadoop

core-site.xml

```
<configuration>
 <property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000
</value>
 </property>
</configuration
```

hdfs-site.xml

```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
 </property>
 <property>
  <name>dfs.namenode.name.dir</name>

<value>C:\Hadoop\data\namenode</value>
 </property>
 <property>
  <name>dfs.datanode.name.dir</name>
  <value>C:\Hadoop\data\datanode</value>
 </property>
</configuration>
```

<table>
<tr><td></td><td>
<table>
<tr><th>mapreduce-site.xml</th><th>yarn-site.xml</th></tr>
<tr><td><property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property></td><td><property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property></td></tr>
</table>

⇨ Replace bin folder under hadoop with bin folder from the below drive
https://drive.google.com/drive/folders/1iURNbow2IglhAhSy3sfY5xxVfAg33NBW

⇨ Double click bin\winutils , if error is displayed then download msvcr120.dll (64 bit, last option) and paste in C:\Windows\System32
https://www.dll-files.com/msvcr120.dll.html

⇨ Set Variable
User Var HADOOP_HOME:C:\Hadoop\bin
System Var C:\Hadoop\bin
C:\Hadoop\sbin

⇨ Check in cmd prompt : hadoop
(open as Admin) hadoop version</td></tr>
<tr><td>Step 3:

Commands to start Hadoop</td><td>⇨ Check in cmd prompt (open as Admin)

⇨ cd C:\Hadoop\sbin

⇨ hdfs namenode -format (1st time)

⇨ start-dfs</td></tr>
</table>

	⇨ jps ⇨ start-yarn ⇨ jps Browse for Application Status ⇨ http://localhost:9870/ Browse for data logs ⇨ http://localhost:8088/cluster
Basic Commands	**Create** a directory in HDFS at given path(s) ⇨ C:\hadoop>hadoop fs -mkdir /Directory1 **List** the contents of the directory ⇨ C:\hadoop>hadoop fs –ls /Directory1 **Upload** local file to Hadoop ⇨ C:\hadoop>hadoop fs –put C:\WCData.txt /Directory1 **Download** Hadoop files to local ⇨ C:\hadoop>hadoop fs –get /Directory1 C:\WCInfo.txt **Display** the content of file ⇨ C:\hadoop>hadoop fds –cat / Directory1/WCData.txt **Remove** a file from a folder recursively ⇨ hadoop dfs -rm -r /input/WCData.txt **Remove** a out file from cluster ⇨ hadoop fs -rm -r hdfs://localhost:9000/out

Experiment 2

Develop a Map Reduce program for Matrix Multiplication

```
1
2  import java.io.IOException;
3  import java.util.*;
4  import java.util.AbstractMap.SimpleEntry;
5  import java.util.Map.Entry;
6
7  import org.apache.hadoop.fs.Path;
8  import org.apache.hadoop.conf.*;
9  import org.apache.hadoop.io.*;
10 import org.apache.hadoop.mapreduce.*;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
15
16 public class MatrixMulti {
17
18     public static class Map extends Mapper<LongWritable, Text, Text, Text> {
19         public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
20             Configuration conf = context.getConfiguration();
21             /*
22              * Row column count
23              */
24             int m = Integer.parseInt(conf.get("m"));
25             int p = Integer.parseInt(conf.get("p"));
26             int s = Integer.parseInt(conf.get("s"));
27             int t = Integer.parseInt(conf.get("t"));
28             int v = Integer.parseInt(conf.get("v"));
29
30             int mPerS = m/s; // Number of blocks in each column of A.
31             int pPerV = p/v; // Number of blocks in each row of B.
32
33             String line = value.toString();
34             String[] indicesAndValue = line.split(",");
35             Text outputKey = new Text();
36             Text outputValue = new Text();
37
38             if (indicesAndValue[0].equals("A")) {
39                 int i = Integer.parseInt(indicesAndValue[1]);
40                 int j = Integer.parseInt(indicesAndValue[2]);
41                 for (int kPerV = 0; kPerV < pPerV; kPerV++) {
42                     outputKey.set(Integer.toString(i/s) + "," + Integer.toString(j/t) + "," + Integer.toString(kPerV));
43                     outputValue.set("A," + Integer.toString(i%s) + "," + Integer.toString(j%t) + "," + indicesAndValue[3]);
44                     context.write(outputKey, outputValue);
45                 }
46             } else {
47                 int j = Integer.parseInt(indicesAndValue[1]);
48                 int k = Integer.parseInt(indicesAndValue[2]);
49                 for (int iPerS = 0; iPerS < mPerS; iPerS++) {
50                     outputKey.set(Integer.toString(iPerS) + "," + Integer.toString(j/t) + "," + Integer.toString(k/v));
51                     outputValue.set("B," + Integer.toString(j%t) + "," + Integer.toString(k%v) + "," + indicesAndValue[3]);
52                     context.write(outputKey, outputValue);
53                 }
54             }
55         }
56     }
57
58     public static class Reduce extends Reducer<Text, Text, Text, Text> {
59         public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
60             String[] value;
61             ArrayList<Entry<String, Float>> listA = new ArrayList<Entry<String, Float>>();
62             ArrayList<Entry<String, Float>> listB = new ArrayList<Entry<String, Float>>();
63             for (Text val : values) {
64                 value = val.toString().split(",");
65                 if (value[0].equals("A")) {
66                     listA.add(new SimpleEntry<String, Float>(value[1] + "," + value[2], Float.parseFloat(value[3])));
67                 } else {
68                     listB.add(new SimpleEntry<String, Float>(value[1] + "," + value[2], Float.parseFloat(value[3])));
69                 }
70             }
71             String[] iModSAndJModT;
72             String[] jModTAndKModV;
```

```
            float a_ij;
            float b_jk;
            String hashKey;
            HashMap<String, Float> hash = new HashMap<String, Float>();
            for (Entry<String, Float> a : listA) {
                iModSAndJModT = a.getKey().split(",");
                a_ij = a.getValue();
                for (Entry<String, Float> b : listB) {
                    jModTAndKModV = b.getKey().split(",");
                    b_jk = b.getValue();
                    if (iModSAndJModT[1].equals(jModTAndKModV[0])) {
                        hashKey = iModSAndJModT[0] + "," + jModTAndKModV[1];
                        if (hash.containsKey(hashKey)) {
                            hash.put(hashKey, hash.get(hashKey) + a_ij*b_jk);
                        } else {
                            hash.put(hashKey, a_ij*b_jk);
                        }
                    }
                }
            }
            String[] blockIndices = key.toString().split(",");
            String[] indices;
            String i;
            String k;
            Configuration conf = context.getConfiguration();
            int s = Integer.parseInt(conf.get("s"));
            int v = Integer.parseInt(conf.get("v"));
            Text outputValue = new Text();
            for (Entry<String, Float> entry : hash.entrySet()) {
                indices = entry.getKey().split(",");
                i = Integer.toString(Integer.parseInt(blockIndices[0])*s + Integer.parseInt(indices[0]));
                k = Integer.toString(Integer.parseInt(blockIndices[2])*v + Integer.parseInt(indices[1]));
                outputValue.set(i + "," + k + "," + Float.toString(entry.getValue()));
                context.write(null, outputValue);
            }
        }
    }
```

```

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        // A is an m-by-n matrix; B is an n-by-p matrix.
        conf.set("m", "2");
        conf.set("n", "2");
        conf.set("p", "2");
        conf.set("s", "2"); // Number of rows in a block in A.
        conf.set("t", "2"); // Number of columns in a block in A = number of rows in a block in B.
        conf.set("v", "2"); // Number of columns in a block in B.


        @SuppressWarnings("deprecation")
        Job job = new Job(conf, "Multiplication");
        job.setJarByClass(MatrixMulti.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

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

        job.waitForCompletion(true);
    }
}
```

<table>
<tr><th><u>Input Data-2*2 matrix</u></th><th><u>Output Data-2*2 matrix</u></th></tr>
<tr><td>A,0,0,1
A,0,1,2
A,1,0,3
A,1,1,4
B,0,0,5
B,0,1,6
B,1,0,7
B,1,1,8</td><td>0,0,19.0
1,0,43.0
0,1,22.0
1,1,50.0</td></tr>
<tr><th colspan="2">Hadoop</th></tr>
<tr><td>Open command prompt as admin</td><td>➢ cd C:\Hadoop\sbin
➢ start-dfs
➢ jps
➢ start-yarn
➢ jps
Browse for Application Status http://localhost:9870/
Browse for data logs http://localhost:8088/cluster
➢ hadoop fs -mkdir /input
➢ hadoop fs -put C:/Matrix_Data.txt /input
➢ hadoop jar C:\Hadoop\share\hadoop\mapreduce\MatrixMultiplication.jar /input /out
➢ hadoop fs -cat /out/</td></tr>
</table>

Experiment 3 :

Develop a Map Reduce program for mines the weather data and display appropriate messages indicating the weather condition of the day.

```java
// Required imports for Hadoop MapReduce
import java.io.IOException;
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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.conf.Configuration;

// Main class
public class WeatherCondition {

    // Mapper class: Extracts max and min temperature from each line
    public static class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text, Text> {

        // Sentinel value used in dataset to represent missing temperature
        public static final int MISSING = 9999;

        // Map method called for each line in the input file
        @Override
        public void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {

            String line = value.toString(); // Convert line to string

            if (line.length() != 0) {  // Skip empty lines
                // Extract date from line (characters 6 to 14)
                String date = line.substring(6, 14);

                // Extract and trim max and min temperatures
                float temp_Max = Float.parseFloat(line.substring(39, 45).trim());
                float temp_Min = Float.parseFloat(line.substring(47, 53).trim());

                // If max temperature is valid and > 30°C, consider it a hot day
                if (temp_Max != MISSING && temp_Max > 30.0) {
                    context.write(new Text("Hot Day: " + date), new Text(String.valueOf(temp_Max)));
                }

                // If min temperature is valid and < 15°C, consider it a cold day
                if (temp_Min != MISSING && temp_Min < 15.0) {
                    context.write(new Text("Cold Day: " + date), new Text(String.valueOf(temp_Min)));
                }
            }
        }
    }

    // Reducer class: Simply passes through the (key, value) pairs from mapper
    public static class MaxTemperatureReducer extends Reducer<Text, Text, Text, Text> {
        @Override
        public void reduce(Text key, Iterable<Text> values, Context context)
                throws IOException, InterruptedException {
            // Write each value to the output (usually only one value per key in this case)
            for (Text val : values) {
                context.write(key, val);
            }
        }
    }

```

```java
    // Driver method: Configures and starts the MapReduce job
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();  // Create Hadoop job configuration
        Job job = Job.getInstance(conf, "Weather Analysis");  // Initialize job with name

        job.setJarByClass(WeatherCondition.class);  // Set main class
        job.setMapperClass(MaxTemperatureMapper.class);  // Set mapper class
        job.setReducerClass(MaxTemperatureReducer.class);  // Set reducer class

        // Set output types for mapper
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        // Set input/output formats
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        // Set input and output file paths from command-line arguments
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        // Submit job and exit based on completion status
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Weather DataSet link

https://www.ncei.noaa.gov/pub/data/uscrn/products/daily01/

OR

https://www.weather.gov/

Austin State data of the year :**2015 (365 rows)**

Field#	Name	Units
1	WBANNO	XXXXX
2	LST_DATE	YYYYMMDD
3	CRX_VN	XXXXXX
4	LONGITUDE	Decimal_degrees
5	LATITUDE	Decimal_degrees
6	T_DAILY_MAX	Celsius
7	T_DAILY_MIN	Celsius
8	T_DAILY_MEAN	Celsius
9	T_DAILY_AVG	Celsius
10	P_DAILY_CALC	mm
11	SOLARAD_DAILY	MJ/m^2
12	SUR_TEMP_DAILY_TYPE	X
13	SUR_TEMP_DAILY_MAX	Celsius
14	SUR_TEMP_DAILY_MIN	Celsius
15	SUR_TEMP_DAILY_AVG	Celsius
16	RH_DAILY_MAX	%
17	RH_DAILY_MIN	%
18	RH_DAILY_AVG	%
19	SOIL_MOISTURE_5_DAILY	m^3/m^3
20	SOIL_MOISTURE_10_DAILY	m^3/m^3
21	SOIL_MOISTURE_20_DAILY	m^3/m^3
22	SOIL_MOISTURE_50_DAILY	m^3/m^3
23	SOIL_MOISTURE_100_DAILY	m^3/m^3
24	SOIL_TEMP_5_DAILY	Celsius
25	SOIL_TEMP_10_DAILY	Celsius
26	SOIL_TEMP_20_DAILY	Celsius
27	SOIL_TEMP_50_DAILY	Celsius
28	SOIL_TEMP_100_DAILY	Celsius

Austin.txt

ncei.noaa.gov/pub/data/uscrn/products/daily01/2015/CRND0103-2015-TX_Austin_33_NW.txt

```
23907 20150101 2.423 -98.08 30.62  2.2 -0.6  0.8  0.9  7.0  1.47 C  3.7  1.1  2.5  99.9  85.4  97.2 0.369 0.308 -99.000 -99.000 -99.000  7.0  8.1 -9999.0 -9999.0 -9999.0
23907 20150102 2.423 -98.08 30.62  3.5  1.3  2.4  2.2 10.2  1.43 C  4.9  2.3  3.1 100.0  98.8  99.8 0.391 0.327 -99.000 -99.000 -99.000  7.1  7.9 -9999.0 -9999.0 -9999.0
23907 20150103 2.423 -98.08 30.62 15.9  2.3  9.1  7.5  3.1 11.00 C 16.4  2.9  7.3 100.0  34.8  73.7 0.450 0.397 -99.000 -99.000 -99.000  7.6  7.9 -9999.0 -9999.0 -9999.0
23907 20150104 2.423 -98.08 30.62  9.2 -1.3  3.9  4.2  0.0 13.24 C 12.4 -0.5  4.9  82.0  40.6  61.7 0.413 0.352 -99.000 -99.000 -99.000  7.3  7.9 -9999.0 -9999.0 -9999.0
23907 20150105 2.423 -98.08 30.62 10.9 -3.7  3.6  2.6  0.0 13.37 C 14.7 -3.0  3.8  77.9  33.3  57.4 0.399 0.340 -99.000 -99.000 -99.000  6.3  7.0 -9999.0 -9999.0 -9999.0
23907 20150106 2.423 -98.08 30.62 20.2  2.9 11.6 10.9  0.0 12.90 C 22.0  1.6  9.9  67.7  30.2  49.3 0.395 0.335 -99.000 -99.000 -99.000  8.0  8.0 -9999.0 -9999.0 -9999.0
23907 20150107 2.423 -98.08 30.62 10.9 -3.4  3.8  4.5  0.0 12.68 C 12.4 -2.1  5.5  82.7  36.5  55.7 0.387 0.328 -99.000 -99.000 -99.000  7.6  8.3 -9999.0 -9999.0 -9999.0
23907 20150108 2.423 -98.08 30.62  0.6 -7.9 -3.6 -3.3  0.0  4.98 C  3.9 -4.8 -0.5  57.7  37.6  48.1 0.372 0.316 -99.000 -99.000 -99.000  4.7  6.1 -9999.0 -9999.0 -9999.0
23907 20150109 2.423 -98.08 30.62  2.0  0.1  1.0  0.8  0.0  2.52 C  4.1  1.2  2.5  87.8  48.9  64.4 0.368 0.312 -99.000 -99.000 -99.000  5.4  6.2 -9999.0 -9999.0 -9999.0
23907 20150110 2.423 -98.08 30.62  0.5 -2.0 -0.8 -0.6  3.9  2.11 C  2.5 -0.1  1.4  99.9  47.7  85.8 0.373 0.314 -99.000 -99.000 -99.000  5.1  6.0 -9999.0 -9999.0 -9999.0
23907 20150111 2.423 -98.08 30.62 10.9  0.0  5.4  4.4  2.6  6.38 C 12.7  1.3  5.8 100.0  77.8  97.1 0.420 0.362 -99.000 -99.000 -99.000  6.5  6.7 -9999.0 -9999.0 -9999.0
23907 20150112 2.423 -98.08 30.62  6.5  1.4  4.0  4.3  0.0  1.55 C  6.9  2.7  5.1 100.0  89.4  97.8 0.412 0.350 -99.000 -99.000 -99.000  7.3  7.5 -9999.0 -9999.0 -9999.0
23907 20150113 2.423 -98.08 30.62  3.0 -0.7  1.1  1.2  0.0  3.26 C  5.6  0.7  2.9  99.7  80.7  90.7 0.401 0.337 -99.000 -99.000 -99.000  6.1  6.8 -9999.0 -9999.0 -9999.0
23907 20150114 2.423 -98.08 30.62  2.9  0.9  1.9  1.8  0.7  1.88 C  4.7  2.0  3.1  99.6  90.8  97.9 0.395 0.331 -99.000 -99.000 -99.000  6.1  6.7 -9999.0 -9999.0 -9999.0
23907 20150115 2.423 -98.08 30.62 13.2  1.2  7.2  6.4  0.0 13.37 C 16.4  1.4  6.7  98.9  46.7  73.4 0.395 0.333 -99.000 -99.000 -99.000  6.7  7.0 -9999.0 -9999.0 -9999.0
23907 20150116 2.423 -98.08 30.62 16.7  3.5 10.1  9.9  0.0 13.68 C 19.2  1.3  8.7  80.2  38.1  58.2 0.391 0.330 -99.000 -99.000 -99.000  7.3  7.4 -9999.0 -9999.0 -9999.0
23907 20150117 2.423 -98.08 30.62 19.5  5.0 12.2 12.3  0.0 10.96 C 20.9  3.3 10.6  87.7  30.4  55.7 0.388 0.327 -99.000 -99.000 -99.000  8.7  8.4 -9999.0 -9999.0 -9999.0
23907 20150118 2.423 -98.08 30.62 20.9  7.6 14.3 13.7  0.0 15.03 C 23.4  3.5 11.9  45.9  14.6  31.4 0.383 0.325 -99.000 -99.000 -99.000  9.5  9.2 -9999.0 -9999.0 -9999.0
23907 20150119 2.423 -98.08 30.62 23.9  6.7 15.3 14.3  0.0 14.10 C 25.6  3.8 12.6  65.3  26.8  45.6 0.376 0.321 -99.000 -99.000 -99.000  9.9  9.5 -9999.0 -9999.0 -9999.0
23907 20150120 2.423 -98.08 30.62 26.0  9.5 17.8 15.9  0.0 14.57 C 27.9  6.5 14.5  88.4  16.1  50.2 0.373 0.320 -99.000 -99.000 -99.000 10.9 10.4 -9999.0 -9999.0 -9999.0
23907 20150121 2.423 -98.08 30.62 11.0  6.9  8.9  8.9  1.7  2.71 C 13.1  6.8  9.7  99.2  68.0  88.1 0.369 0.317 -99.000 -99.000 -99.000 10.7 10.6 -9999.0 -9999.0 -9999.0
23907 20150122 2.423 -98.08 30.62  8.6  3.5  6.1  5.6 40.0  1.28 C  9.1  4.1  6.3  99.6  95.2  98.0 0.546 0.418 -99.000 -99.000 -99.000  9.0  9.3 -9999.0 -9999.0 -9999.0
23907 20150123 2.423 -98.08 30.62  9.4  2.2  5.8  4.2  7.5  6.58 C 11.1  2.0  4.8  98.4  58.8  86.5 0.554 0.409 -99.000 -99.000 -99.000  7.6  8.1 -9999.0 -9999.0 -9999.0
23907 20150124 2.423 -98.08 30.62 16.0  1.4  8.7  8.0  0.0 14.26 C 18.8  0.4  7.7  92.0  33.0  63.0 0.494 0.381 -99.000 -99.000 -99.000  7.7  7.9 -9999.0 -9999.0 -9999.0
23907 20150125 2.423 -98.08 30.62 20.2  6.4 13.3 12.7  0.0 14.99 C 22.0  4.4 11.0  69.2  18.9  43.8 0.456 0.357 -99.000 -99.000 -99.000  9.1  8.9 -9999.0 -9999.0 -9999.0
23907 20150126 2.423 -98.08 30.62 21.5  7.2 14.4 14.1  0.0 12.01 C 22.9  5.5 12.2  56.8  23.7  40.6 0.433 0.349 -99.000 -99.000 -99.000 10.0  9.7 -9999.0 -9999.0 -9999.0
23907 20150127 2.423 -98.08 30.62 26.5 10.7 18.6 17.5  0.0 15.18 C 28.9  8.1 15.5  52.2  21.4  38.8 0.420 0.344 -99.000 -99.000 -99.000 11.4 10.8 -9999.0 -9999.0 -9999.0
23907 20150128 2.423 -98.08 30.62 26.3 13.3 19.8 19.1  0.0 15.11 C 28.1  7.9 16.3  54.9  19.4  35.5 0.410 0.339 -99.000 -99.000 -99.000 12.1 11.5 -9999.0 -9999.0 -9999.0
23907 20150129 2.423 -98.08 30.62 23.1  9.8 16.5 16.4  0.0 13.74 C 27.4  9.7 16.4  87.0  34.2  55.6 0.402 0.334 -99.000 -99.000 -99.000 13.1 12.4 -9999.0 -9999.0 -9999.0
23907 20150130 2.423 -98.08 30.62 13.0  6.9 10.0  9.0  0.2  7.19 C 19.2  8.3 11.0  67.6  48.4  58.8 0.389 0.328 -99.000 -99.000 -99.000 11.6 11.8 -9999.0 -9999.0 -9999.0
23907 20150131 2.423 -98.08 30.62 15.1  7.4 11.3 10.2  8.5  1.18 C 14.5  8.4 10.7 100.0  63.1  90.3 0.401 0.337 -99.000 -99.000 -99.000 11.2 11.3 -9999.0 -9999.0 -9999.0
23907 20150201 2.423 -98.08 30.62 18.3  3.9 11.1 13.3  0.0  8.69 C 22.1  4.1 13.8  98.8  53.6  79.1 0.450 0.386 -99.000 -99.000 -99.000 12.9 12.6 -9999.0 -9999.0 -9999.0
23907 20150202 2.423 -98.08 30.62  8.0 -1.9  3.1  3.3  0.0 12.48 C 15.2 -0.6  5.8  69.4  34.8  54.2 0.420 0.354 -99.000 -99.000 -99.000  9.3 10.1 -9999.0 -9999.0 -9999.0
23907 20150203 2.423 -98.08 30.62  5.3  2.3  3.8  3.8  0.8  2.69 C  8.3  3.9  5.7 100.0  65.1  82.6 0.409 0.343 -99.000 -99.000 -99.000  8.7  9.3 -9999.0 -9999.0 -9999.0
```

Output File

```
Cold Day:  20151217        3.1
Cold Day:  20151218        0.0
Cold Day:  20151219        4.1
Cold Day:  20151220        10.1
Cold Day:  20151221        14.6
Cold Day:  20151222        11.2
Cold Day:  20151224        14.4
Cold Day:  20151225        9.3
Cold Day:  20151227        0.4
Cold Day:  20151228        -0.1
Cold Day:  20151229        -0.1
Cold Day:  20151230        4.0
Cold Day:  20151231        2.5
Hot Day:  20150425         30.8
Hot Day:  20150518         31.1
Hot Day:  20150520         30.6
Hot Day:  20150526         30.2
Hot Day:  20150529         30.9
Hot Day:  20150601         30.1
Hot Day:  20150602         30.3
Hot Day:  20150603         30.4
Hot Day:  20150604         30.9
Hot Day:  20150605         30.8
Hot Day:  20150606         31.7
Hot Day:  20150607         31.7
Hot Day:  20150608         33.4
```

Hadoop

Open command prompt as admin	➢ cd C:\Hadoop\sbin ➢ start-dfs ➢ jps ➢ start-yarn ➢ jps Browse for Application Status http://localhost:9870/ Browse for data logs http://localhost:8088/cluster ➢ hadoop fs -mkdir /input ➢ hadoop fs -put C:/Austin.txt /input ➢ hadoop jar C:\Hadoop\share\hadoop\mapreduce\WeatherReport.jar /input /out ➢ hadoop fs -cat /out/

Experiment 4:

Develop a Map Reduce program to find the tags associated with each movie by analyzing movie lens data

```
import csv
from functools import reduce

# Function to create key-value pairs for the Map step
def mapper(tags):
    key_value_pairs = []
    for row in tags:
        movie_id = row['movieId']
        tag = row['tag']
        key_value_pairs.append((movie_id, tag))
    return key_value_pairs

# Function to collect tags for the Reduce step
def reducer(key, values):
    return list(set(values))

# Function to perform MapReduce
def map_reduce(tags):
    # Apply the mapper function to create key-value pairs
    key_value_pairs = mapper(tags)
    # Group values by key (movieId)
    grouped_values = {}
    for key, value in key_value_pairs:
        if key not in grouped_values:
            grouped_values[key] = []
        grouped_values[key].append(value)

    # Apply the reducer function to get final results
    result = {}
    for key in grouped_values:
        result[key] = reducer(key, grouped_values[key])
    return result

# Function to load tags data from CSV file
def load_tags(file_path):
    tags = []
    with open(file_path, mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            tags.append(row)
    return tags
```

```
# Function to load movie titles from CSV file
def load_movies(file_path):
    movies = {}
    with open(file_path, mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            movies[row['movieId']] = row['title']
    return movies

# Load MovieLens data
tags = load_tags('tags.csv')
movies = load_movies('movies.csv')

# Perform MapReduce to find tags associated with each movie
result = map_reduce(tags)

# Display results
for movie_id, tags in result.items():
    movie_title = movies[movie_id]
    print(f'Movie: {movie_title}, Tags: {", ".join(tags)}')
```

OUTPUT:

```
Movie: dada, Tags: sad
Movie: abhi, Tags: angry
Movie: ragha, Tags: fun
```

Experiment 5:

Implement functions: Count-Sort-Limit-Skip- Aggregate using mongo DB.

Install Mongo Compass and use command to login	mongodb+srv://root:<db_password>@cluster0.iwsjsnm.mongodb.net/
Install Mongo Shell and use command to login	mongosh "mongodb+srv://cluster0.iwsjsnm.mongodb.net/" --apiVersion 1 --username root
Database Query	show **dbs** use DB1 db.**dropDatabase**() show collections
Collection Query	db.**createCollection**("Student") db.Student.**drop**() --Entire Table
Delete Query	db.Student.**deleteOne**({id:01}) db.Student.**remove**({id:01})
Insert Query	db.Student.**insertOne**({_id:11,USN:001,Name:"Ramya",Age:21,Dept:"IS",Grade:"A",Address:{Street:"Street1",Loc:"CKM",State:"KAR"},Hobby:["Singing"]}); db.Student.**insertMany**([{_id:8,USN:008,Name:"Arun",Age:21,Dept:"CS",Grade:"C",Address:{Street:"Street3",Loc:"COCH",State:"KER"}, Hobby:["Travelling"]}, {_id:9,USN:009,Name:"Anu",Age:21,Dept:"IS",Grade:"A",Address:{Street:"Street2",Loc:"RAN",State:"JARK"}, Hobby:["Cooking"]}

	]);
Search Query	db.Student.**find**() --Entire Table -->db.StudentSample2.**find**({},{USN:1,Name:1,_id:0}); Display all rows with USN and Name and _id not displayed -->db.StudentSample2.**find**({_id:8},{Name:1,Age:1}); Display only row with _id ,Name and Age where id=8 -->db.StudentSample2.**find**({Age:{$eq:22}})
Update Query **SAVE**	db.Student.**update**({ _id:1}, {$set:{Hobby:"Singing"}}); UPDATE ELSE INSERT (**SAVE**) (insert if not existing) db.Student.**update**({_id: 12, Name: "Ramya2", Age: 21, Dept: "IS"},{$set:{Hobby:"Sing"}},{**upsert:true**}) (do not insert if not existing) db.Student.**update**({_id: 12, USN: 001, Name: "Ramya", Age: 21, Dept: "IS"},{$set:{Hobby:"Sing"}},{**upsert:false**})
Count Query	db.StudentSample2.find({Age:21}).**count**();
Sort Query	db.StudentSample2.find({Age:21}).**sort**({Name:1}) db.StudentSample2.**find**({Age:21}).**sort**({Name:-1})
Limit Query	db.StudentSample2.find({Age:21}.**limit**(2);
Skip Query	db.StudentSample2.find().**skip**(3).
Aggregation Query	db.Bank.insertMany([{Name:"Shilpa",Custid:"C123",AcctBal:500,AcctType:"S"}, { Name:"Shilpa",Custid:"C123",AcctBal:900,AcctType:"S"}, { Name:"Hema",Custid:"C111",AcctBal:1200,AcctType:"S"},

	{ Name:”Shilpa”,Custid:"C123",AcctBal:1500,AcctType:"C"}, { Name:”Ajay”,Custid:"C134",AcctBal:400,AcctType:"C"}]) db.Bank.**aggregate**([{$out:"Bank1"}]); - make a copy db.Bank.**aggregate**({$**group**:{_id:"$Custid",TotalAcctBal:{$**sum**:"$AcctBal"}}}); db.Bank.**aggregate**({$**group**:{_id:"$Custid",TotalAcctBal:{$**avg**:"$AcctBal"}}}); db.Bank.**aggregate**({$**group**:{_id:"$Custid",TotalAcctBal:{$**min**:"$AcctBal"}}}); db.Bank.**aggregate**({$**match**:{AcctType:"S"}},{$group:{_id:"$Custid",TotalAcctBal:{$sum:"$AcctBal"}}});

Experiment 6:

Write a Pig Latin Scripts to sort, group, join, Project and filter the data Source Code

employees.txt (Employee ID, Name, Age, Department ID, Salary) 101,John,30,1,50000 102,Sam,28,2,60000 103,Anna,32,1,75000 104,David,29,3,62000 105,Lily,27,2,58000	**departments.txt** 1,HR 2,Finance 3,IT

Pig Latin Script

Save the script as employee_analysis.pig and execute it in Cloudera.

```
-- Load the employees dataset
employees = LOAD 'hdfs://localhost:9000/user/cloudera/employees.txt' USING PigStorage(',') AS (emp_id:int, name:chararray, age:int, dept_id:int, salary:int);

 -- Load the departments dataset
departments = LOAD 'hdfs://localhost:9000/user/cloudera/departments.txt' USING PigStorage(',') AS (dept_id:int, dept_name:chararray);
```

```
-- 1. FILTER:
Select employees with age greater than 28 filtered_employees = FILTER employees BY age > 28;
```

```
-- 2. PROJECT:
Select only emp_id, name, and salary columns projected_employees = FOREACH filtered_employees GENERATE emp_id, name, salary;
```

```
-- 3. SORT:
Order employees by salary in descending order sorted_employees = ORDER projected_employees BY salary DESC;
```

```
-- 4. GROUP:
Group employees by department ID grouped_by_department = GROUP employees BY dept_id;
```

-- 5. JOIN: Join employees with department names using dept_id joined_data = JOIN employees BY dept_id, departments BY dept_id;

-- STORE results in HDFS

STORE sorted_employees INTO 'hdfs://localhost:9000/user/cloudera/output/sorted_employees' USING PigStorage(',');
STORE grouped_by_department INTO 'hdfs://localhost:9000/user/cloudera/output/grouped_by_department' USING PigStorage(',');
STORE joined_data INTO 'hdfs://localhost:9000/user/cloudera/output/joined_data' USING PigStorage(',');
-- DISPLAY the results on the screen DUMP sorted_employees; DUMP grouped_by_department; DUMP joined_data;

Upload the data in HDFS

hdfs dfs -mkdir -p /user/cloudera hdfs dfs -put employees.txt /user/cloudera/ hdfs dfs -put departments.txt /user/cloudera/

Run the Script pig -x mapreduce employee_analysis.pig

ouput commands hdfs dfs -cat /user/cloudera/output/sorted_employees/part-r-00000 hdfs dfs -cat /user/cloudera/output/grouped_by_department/part-r-00000 hdfs dfs -cat /user/cloudera/output/joined_data/part-r-00000

Sorted Employee by Salary

```
103,Anna,75000
104,David,62000
102,Sam,60000
101,John,50000
```

Grouped Employees by Departments

```
(1,{(101,John,30,1,50000),(103,Anna,32,1,75000)})
(2,{(102,Sam,28,2,60000),(105,Lily,27,2,58000)})
(3,{(104,David,29,3,62000)})
```

Joined Employees with Departments

```
(101,John,30,1,50000,1,HR)
(102,Sam,28,2,60000,2,Finance)
(103,Anna,32,1,75000,1,HR)
(104,David,29,3,62000,3,IT)
(105,Lily,27,2,58000,2,Finance)
```

Experiment 7:

Use HIVE to create, alter, and drop databases, tables, views, functions and indexes

Create a Database:

```
CREATE DATABASE employee_db;
```

Use the database:

```
USE employee_db; Output:
```

Alter a Database:

```
ALTER DATABASE employee_db SET DBPROPERTIES ('Owner'='Admin');
```

Drop a Data Base

```
DROP DATABASE employee_db CASCADE;
```

Create a Table

```
CREATE TABLE employees ( emp_id INT, name STRING, age INT, dept_id INT, salary FLOAT )
        ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
```

```
LOAD DATA INPATH '/user/cloudera/employees.txt' INTO TABLE employees;
```

```
ALTER TABLE employees ADD COLUMNS (email STRING); Output:
```

```
DROP TABLE employees_new;
```

Create a View

```
CREATE VIEW high_salary_employees AS SELECT emp_id, name, salary FROM employees WHERE salary > 50000;
```

```
ALTER VIEW high_salary_employees AS SELECT emp_id, name, age, salary FROM employees WHERE salary > 60000;
```

```
DROP VIEW high_salary_employees;
```

Create a Function

```
CREATE FUNCTION to_upper AS 'com.example.hiveudf.ToUpperUDF';
```

Use the Function

```
SELECT to_upper(name) FROM employees;
```

Drop Function `DROP FUNCTION to_upper;`

Create an Index `CREATE INDEX emp_dept_idx ON TABLE employees (dept_id) AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;`

Rebuild the Index:

```
INDEX emp_dept_idx ON employees REBUILD
```

Drop Index `DROP INDEX emp_dept_idx ON employees;`

Check all tables in the current database:

```
SHOW TABLES;
```

DESCRIBE employees;

```
emp_id          int
name            string
age             int
dept_id         int
salary          float
email           string
OK
Time taken: 0.234 seconds
```

Display the table data

```
SELECT * FROM employees LIMIT 5;
101   John   30   1   50000.0   NULL
102   Sam    28   2   60000.0   NULL
103   Anna   32   1   75000.0   NULL
104   David  29   3   62000.0   NULL
105   Lily   27   2   58000.0   NULL
OK
Time taken: 0.459 seconds
```

Experiment 8:

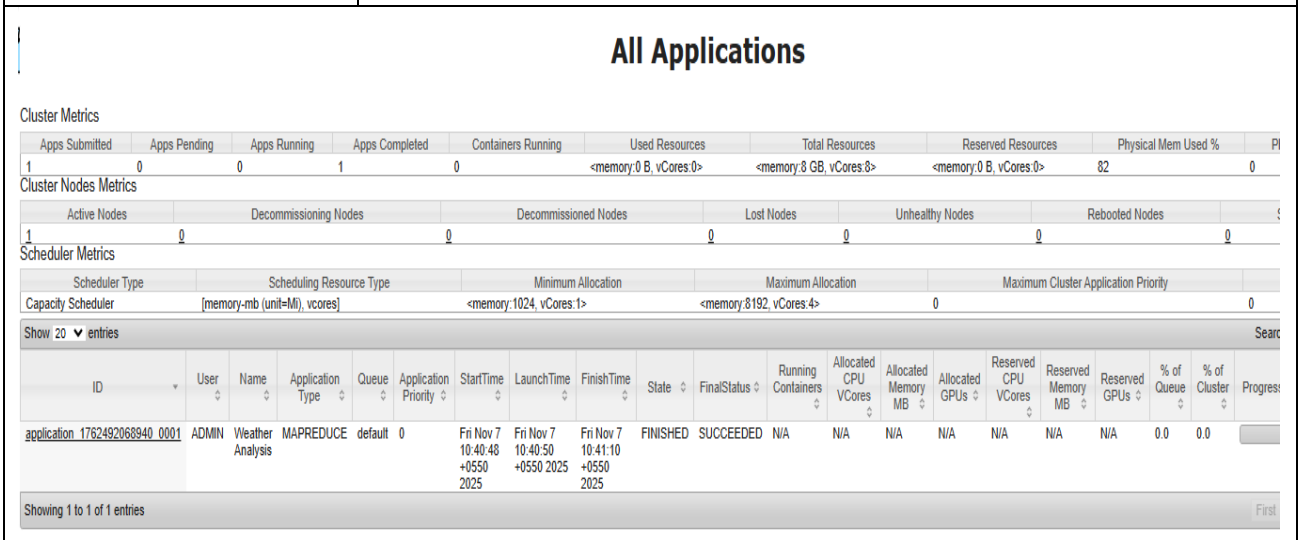
Implement word count program in Hadoop and spark

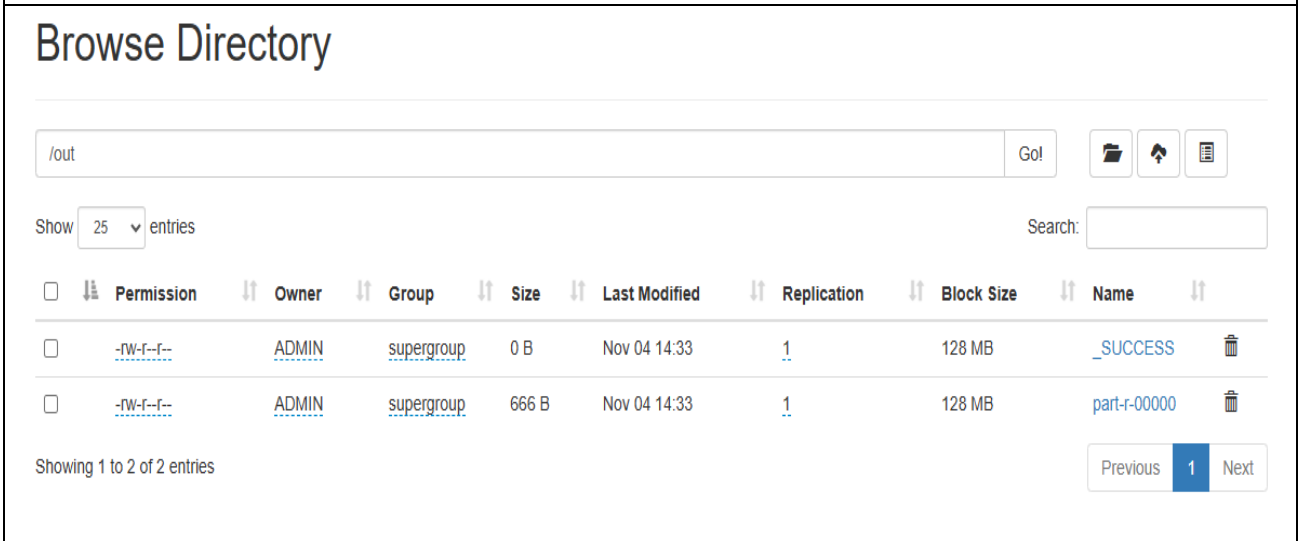
```
import java.io.IOException;
import java.util.StringTokenizer;

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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.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);
  }
}
```

Steps for Execution

Eclipse	
Create New Java **Project**	Project1
Create **new class** under scr	WordCount.java
Build Project Select Project -> Build Path-> Configure build Path-> Library->Add External jar	Include all jar files under C:\Hadoop\share\hadoop\mapreduce and C:\Hadoop\share\hadoop\common
Create JAR file Select java file ->export->Java-Jar → Give the path and name of jar file → next → next→ select the corresponding Java file	C:\Hadoop\share\hadoop\mapreduce**WordCount**.jar
Hadoop	
Open command prompt as admin	➢ cd C:\Hadoop\sbin ➢ start-dfs ➢ jps ➢ start-yarn ➢ jps Browse for Application Status http://localhost:9870/ Browse for data logs http://localhost:8088/cluster ➢ hadoop fs -mkdir /input ➢ hadoop fs -put C:/WCData.txt /input ➢ hdfs dfs -ls / ➢ hdfs dfs -ls /input

- **hadoop jar C:\Hadoop\share\hadoop\mapreduce\WordCount.jar /input /out**
- hdfs dfs -ls /out
- hdfs dfs -ls /out/part-r-00000
- hadoop fs -cat /out/*





File information - part-r-00000

Download Head the file (first 32K) Tail the file (last 32K)

Block information — Block 0

Block ID: 1073741927

Block Pool ID: BP-235667679-172.15.0.121-1757590350259

Generation Stamp: 1103

Size: 666

Availability:

- DESKTOP-8N4CTKA

File contents

```
Beyond  1
Overall,  1
They  1
a  2
```

```
C:\Hadoop\sbin>hadoop jar C:\Hadoop\share\hadoop\mapreduce\WordCount.jar /input /out
2025-11-04 14:32:42,567 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2025-11-04 14:32:43,194 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
2025-11-04 14:32:43,211 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ADMIN/.staging/job_1762240319901_0003
2025-11-04 14:32:43,408 INFO input.FileInputFormat: Total input files to process : 1
2025-11-04 14:32:43,872 INFO mapreduce.JobSubmitter: number of splits:1
2025-11-04 14:32:43,997 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1762240319901_0003
2025-11-04 14:32:43,999 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-11-04 14:32:44,182 INFO conf.Configuration: resource-types.xml not found
2025-11-04 14:32:44,183 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-11-04 14:32:44,266 INFO impl.YarnClientImpl: Submitted application application_1762240319901_0003
2025-11-04 14:32:44,300 INFO mapreduce.Job: The url to track the job: http://DESKTOP-8N4CTKA:8088/proxy/application_1762240319901_0003/
2025-11-04 14:32:44,300 INFO mapreduce.Job: Running job: job_1762240319901_0003
2025-11-04 14:32:51,431 INFO mapreduce.Job: Job job_1762240319901_0003 running in uber mode : false
2025-11-04 14:32:51,432 INFO mapreduce.Job:  map 0% reduce 0%
2025-11-04 14:32:56,534 INFO mapreduce.Job:  map 100% reduce 0%
2025-11-04 14:33:02,600 INFO mapreduce.Job:  map 100% reduce 100%
2025-11-04 14:33:02,610 INFO mapreduce.Job: Job job_1762240319901_0003 completed successfully
2025-11-04 14:33:02,692 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=936
                FILE: Number of bytes written=479855
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=695
                HDFS: Number of bytes written=666
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
```

```
C:\Hadoop\sbin>hadoop fs -cat /out/*
Beyond  1
Overall,        1
They    1
a       2
advantages,     1
also    1
and     6
are     1
as      1
balanced,       1
benefits,       1
body    1
both    1
build   1
by      2
career  1
communities,    1
comprehensive   1
connections,    1
crucial 1
development,    1
discipline,     1
disease,        1
enhancing       1
even    1
for     1
```

Viva Questions:

1. What is Hadoop and why is it used?
2. What are the main components of Hadoop?
3. What is HDFS? Explain its architecture.
4. What is the function of the NameNode in HDFS?
5. What is the function of the DataNode in HDFS?
6. What is the difference between NameNode and Secondary NameNode?
7. What is a block in HDFS, and how large is the default block size?
8. What is the function of YARN in Hadoop?
9. Explain the architecture of YARN.
10. What is MapReduce and how does it work?
11. What is a JobTracker in Hadoop 1.x?
12. What is a TaskTracker in Hadoop 1.x?
13. What is the difference between Hadoop 1.x and Hadoop 2.x?
14. What are the advantages of Hadoop over traditional RDBMS?
15. What is the role of ResourceManager and NodeManager in YARN?
16. What is data replication in HDFS? How is data redundancy handled?
17. What is the function of the JobHistory Server in Hadoop?
18. What are the main differences between HDFS and traditional file systems?
19. What is a combiner in MapReduce?
20. What is the difference between a combiner and a reducerWhat is MongoDB?
21. What are the advantages of using MongoDB over relational databases?
22. What are the data types supported by MongoDB?
23. What is a collection in MongoDB?
24. What is a document in MongoDB?
25. How does MongoDB handle data replication?
26. What is Sharding in MongoDB?
27. What is the use of an index in MongoDB?
28. What is the Aggregation framework in MongoDB?
29. How does MongoDB ensure fault tolerance?
30. What is the difference between MongoDB and RDBMS?
31. What are the different types of indexes in MongoDB?
32. What is the purpose of the find() method in MongoDB?
33. What is Hive and how is it different from traditional RDBMS?
34. What is the architecture of Hive? 35. What is HiveQL?
36. What is the function of the Hive Metastore?
37. What are the different types of tables in Hive?
38. What are the advantages of using Hive over SQL databases?
39. What are partitions in Hive?
40. What are buckets in Hive?
41. What is the difference between internal and external tables in Hive?
42. What file formats does Hive support?
43. What is Apache Spark?
44. What are the core components of Spark?
45. What are the advantages of using Spark over Hadoop MapReduce?