



**RV Educational Institutions®**

**RV Institute of Technology and Management**

**(Affiliated to VTU, Belagavi)**

**JP Nagar 8<sup>th</sup> Phase, Bengaluru-560076**

**Department of Information Science and Engineering**



**Course Name: Big Data Analytics Lab**

**Course Code: BIS701**

**VII Semester**

**2022 Scheme**

**Prepared By:**

**Dr. Shruthi P, Assistant Professor, Dept of ISE, RVITM**

**Mrs.Nisha Wilvicta, Assistant Professor, Dept of ISE, RVITM**

### About the Course:

Course Title : Big Data Analytics Laboratory	Course Code : BIS701
Teaching Hours/Week (L:T:P: S) 3:0:2:0	Credits : 04
SEE Marks :50	CIE Marks :50
Semester :7	Academic Year :2025-26
Lesson Plan Author : Dr.Shruthi P, Mrs.Nisha Wilvicta	Date :07-08-2025

### Course objectives:

This course will enable to,

- To implement Map Reduce programs for processing big data.
- To realize storage and processing of big data using Mongo DB, Pig, Hive and Spark.
- To analyze big data using machine learning techniques \

### Course Outcomes:

At the end of the course, the student will be able to:

- Identify and list various Big Data concepts, tools and applications
- Develop programs using HADOOP framework.
- Use Hadoop Cluster to deploy Map Reduce jobs, PIG,HIVE and Spark programs.
- Analyze the given data set and identify deep insights from the data set.

### Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together..

### Continuous Internal Evaluation (CIE):

IPCC means practical portion integrated with the theory of the course.

- CIE marks for the theory component are **25 marks** and that for the practical component is **25 marks**.

- 25 marks for the theory component are split into **15 marks** for two Internal Assessment Tests (Two Tests, each of **15 Marks** with 01-hour duration, are to be conducted) and **10 marks** for other assessment methods mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.

- Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for **25 marks**).

- The student has to secure 40% of **25 marks** to qualify in the CIE of the theory component of IPCC.

### **CIE for the practical component of the IPCC**

- **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.

### **SEE for IPCC**

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the course (duration 03 hours)

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), should have a mix of topics under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.
4. Marks scored by the student shall be proportionally scaled down to 50 Marks

## INDEX

S.NO	EXPERIMENTS	PAGE NO
1.	Install Hadoop and Implement the following file management tasks in Hadoop: Adding files and directories Retrieving files Deleting files and directories. <b>Hint:</b> 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.	5
2.	Develop a Map Reduce program to implement Matrix Multiplication	6
3.	. Develop a Map Reduce program that mines weather data and displays appropriate messages indicating the weather conditions of the day.	14
4.	Develop a Map Reduce program to find the tags associated with each movie by analyzing movie lens data.	20
5.	Implement Functions: Count – Sort – Limit – Skip – Aggregate using Mongo DB	25
6.	Write Pig Latin scripts to sort, group, join, project, and filter the data.	27
7.	Use Hive to create, alter, and drop databases, tables, views, functions, and indexes	29
8.	Implement a word count program in Hadoop and Spark.	31
9.	Use CDH (Cloudera Distribution for Hadoop) and HUE (Hadoop User Interface) to analyze data and generate reports for sample datasets	42
I	<b>SAMPLE VIVA QUESTIONS</b>	44

**Program 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.

Before starting with we should format namenode and run Hadoop services<datanode and namenode>

***hdfs namenode -format***

***start-dfs.cmd***

### 1. Adding Files and Directories to HDFS

Create a directory in HDFS:

***hdfs dfs -mkdir /mydata***

Copy a local file to HDFS:

***hdfs dfs -put C:/Users/YourUser/Desktop/sample.txt /mydata/***

### 2. Retrieving Files from HDFS

Copy a file from HDFS to the local filesystem:

***hdfs dfs -get /mydata/sample.txt C:/Users/YourUser/Desktop/***

Display the content of a file in HDFS:

***hdfs dfs -cat /mydata/sample.txt***

### 3. Deleting Files and Directories from HDFS

Delete a file in HDFS:

***hdfs dfs -rm /mydata/sample.txt***

Delete a directory in HDFS:

***hdfs dfs -rm -r /mydata***

## Program 2. Develop a Map Reduce program to implement Matrix Multiplication

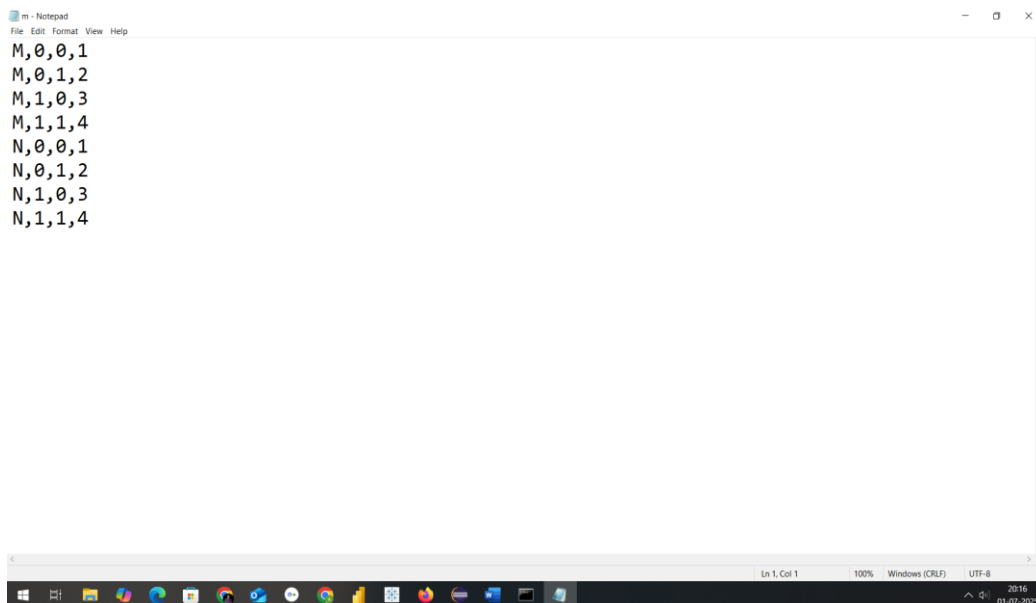
**Step1:** goto Search button run cmd prompt as administrator

**Step2:** Initiate all required component of hadoop by command

start-all.cmd

**Step3:** Create a text file where we have folder running hadoop java program in my case I have created in Desktop folder D:\hadoopprograms\matrixm

U can create anywhere as u wish with m.txt and content looks as in below



**Step3:** Now we need a jar file to execute map reduce program fr wordcount

So open Eclipse

Create a java project with name Map Reduce Matrix Multiplication

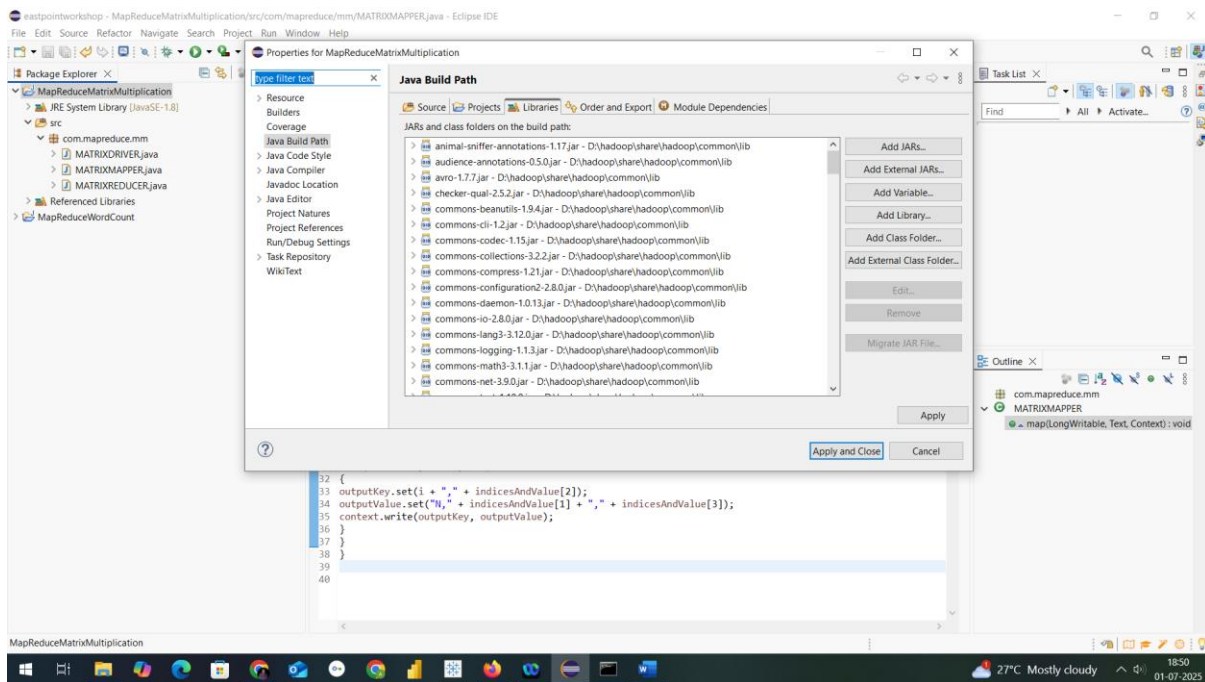
Select execution environment as JavaSE-1.8

And click on next & finish

**Step4:** Right Click on Project name---→ Click on New---→create a package with name com.mapreduce.mm and click on Finish

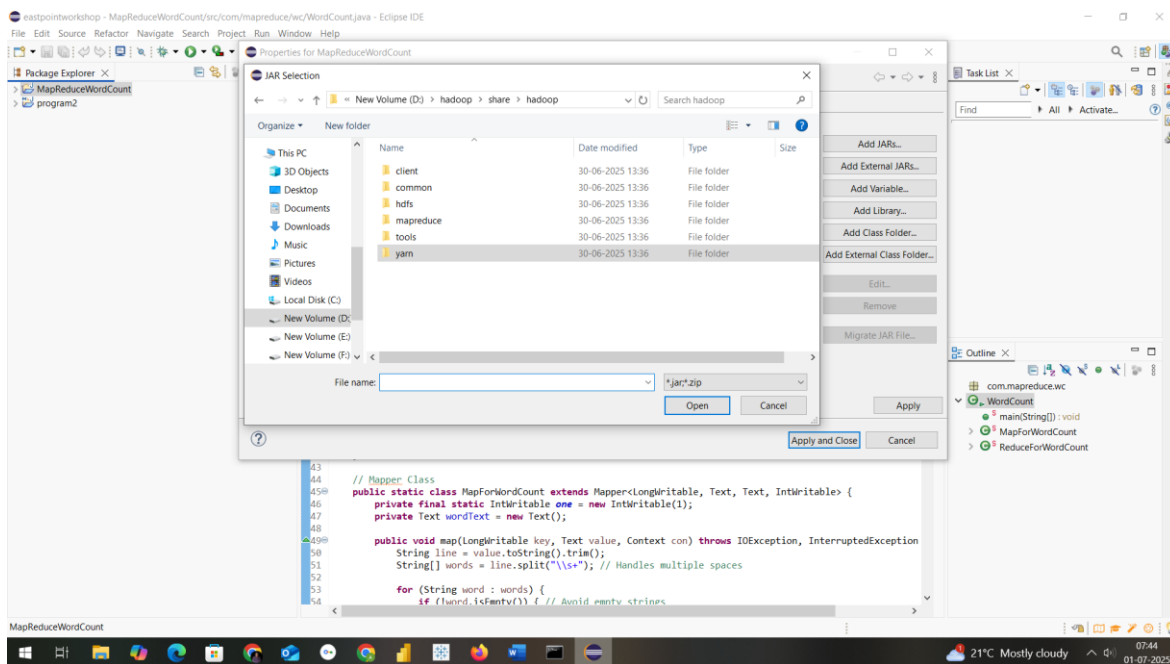
**Step5:** Add required libraries to support hadoop by navigating as in below

Right Click on Project-----→Right Click on Build Path-----→ Configure Build path Then go to Libraries as in screen below



**Step6:** Add necessary External jar file from D:\hadoop\share\hadoop

Like clients,common,hdfs,mapreduce & yarn to support packages for hadoop



**Step7:** Create a class within package com.mapreduce.mm with name MatrixMultiplication and paste this code

```
package com.mapreduce.mm;

import java.io.IOException;
import java.util.HashMap;

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

public class MatrixMultiplication {

    // Mapper Class
    public static class MatrixMapper extends Mapper<Object, Text, Text, Text> {
        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {

            String[] tokens = value.toString().split(",");
            String matrixName = tokens[0]; // M or N

            int i = Integer.parseInt(tokens[1]);
            int j = Integer.parseInt(tokens[2]);
            float val = Float.parseFloat(tokens[3]);

            Configuration conf = context.getConfiguration();
            int n = Integer.parseInt(conf.get("n")); // Matrix dimension

            if (matrixName.equals("M")) {
```

```
// Emit for each column in N
for (int k = 0; k < n; k++) {
    context.write(new Text(i + "," + k), new Text("M," + j + "," + val));
}
} else {
    // Emit for each row in M
    for (int k = 0; k < n; k++) {
        context.write(new Text(k + "," + j), new Text("N," + i + "," + val));
    }
}
}

// Reducer Class
public static class MatrixReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {

        HashMap<Integer, Float> hashA = new HashMap<>();
        HashMap<Integer, Float> hashB = new HashMap<>();

        for (Text val : values) {
            String[] parts = val.toString().split(",");
            String matrix = parts[0];
            int index = Integer.parseInt(parts[1]);
            float value = Float.parseFloat(parts[2]);

            if (matrix.equals("M")) {
                hashA.put(index, value);
            } else {
```

```
        hashB.put(index, value);
    }
}

Configuration conf = context.getConfiguration();
int n = Integer.parseInt(conf.get("n"));
float result = 0.0f;

for (int k = 0; k < n; k++) {
    float a = hashA.getDefault(k, 0.0f);
    float b = hashB.getDefault(k, 0.0f);
    result += a * b;
}

if (result != 0.0f) {
    context.write(key, new Text(String.valueOf(result)));
}
}
}

// Driver (Main Method)
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    // Set matrix dimension n (number of columns in M or rows in N)
    conf.set("mapreduce.framework.name", "local");
    conf.set("fs.defaultFS", "file:///");

    conf.set("n", "3");
```

```

Job job = Job.getInstance(conf, "Matrix Multiplication");

job.setJarByClass(MatrixMultiplication.class);

job.setMapperClass(MatrixMapper.class);

job.setReducerClass(MatrixReducer.class);


job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileInputFormat.setInputPaths(job, new Path(args[0]));

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

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

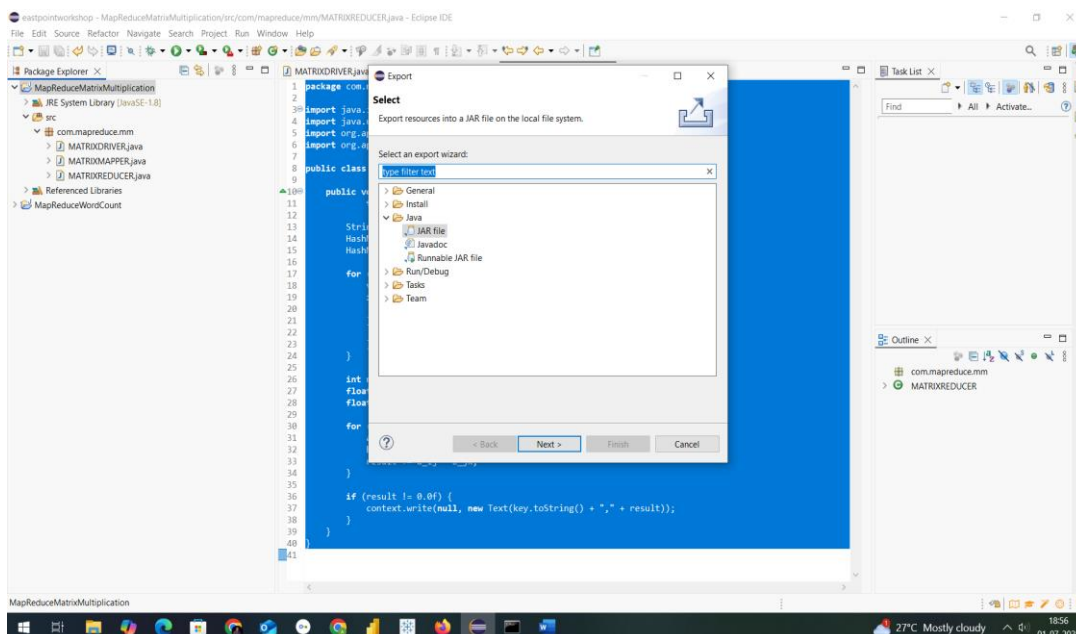
}

}

```

### Step8: Procedure to create jar file from eclipse project

Now Right click on Project and click on export and select jar as in screen

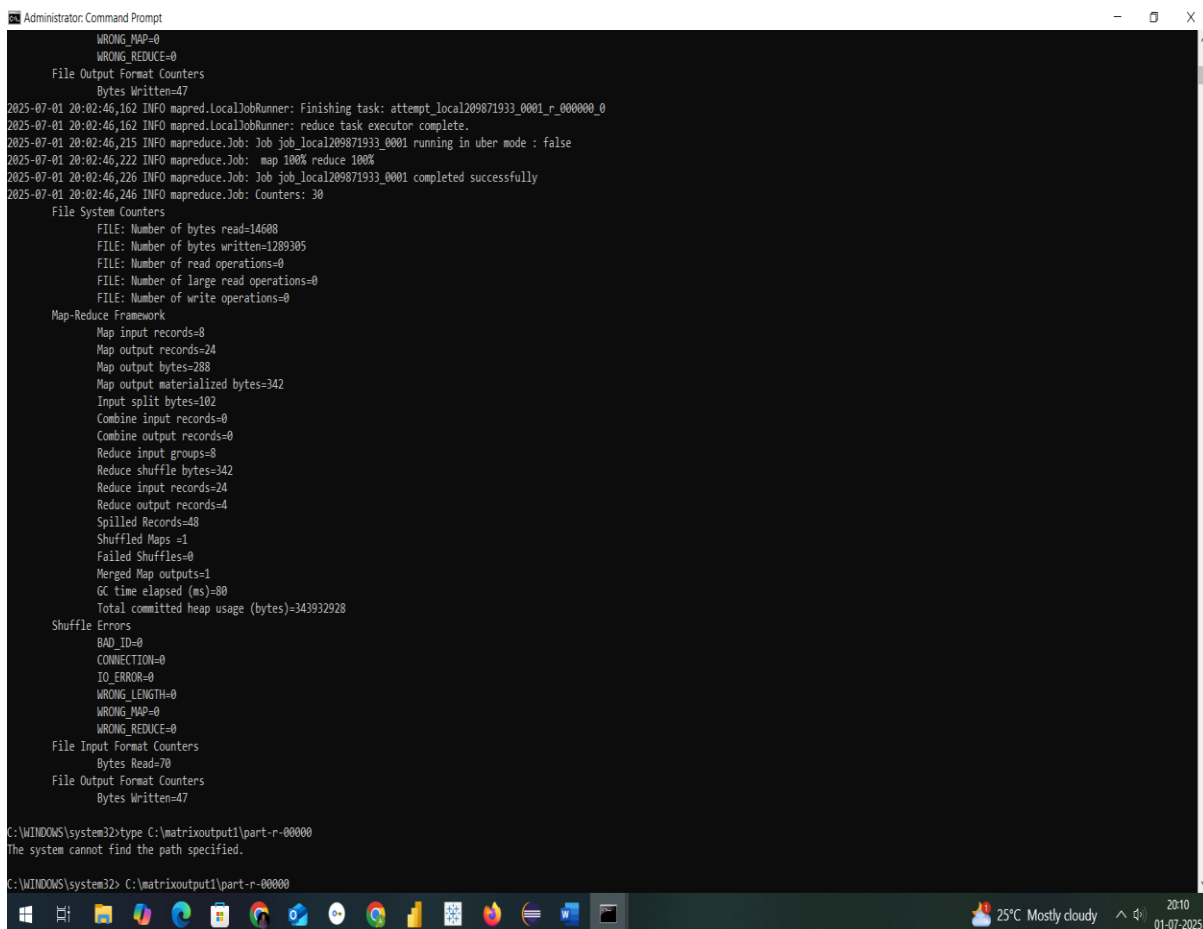


Mention the path where u want to save jar with name of Jar File in my case I have saved in D:\hadoopprograms as MatrixMultiplication

Final step: Run the jar file with command as in below

```
hadoop jar D:\hadoopprograms\Matrix Multiplication.jar  
com.mapreduce.mm.MatrixMultiplication D:/hadoopprograms/matrixm/m.txt D:/hadoop  
programs/matrixm/matrixoutput1
```

After execution in command the output looks as in below screens

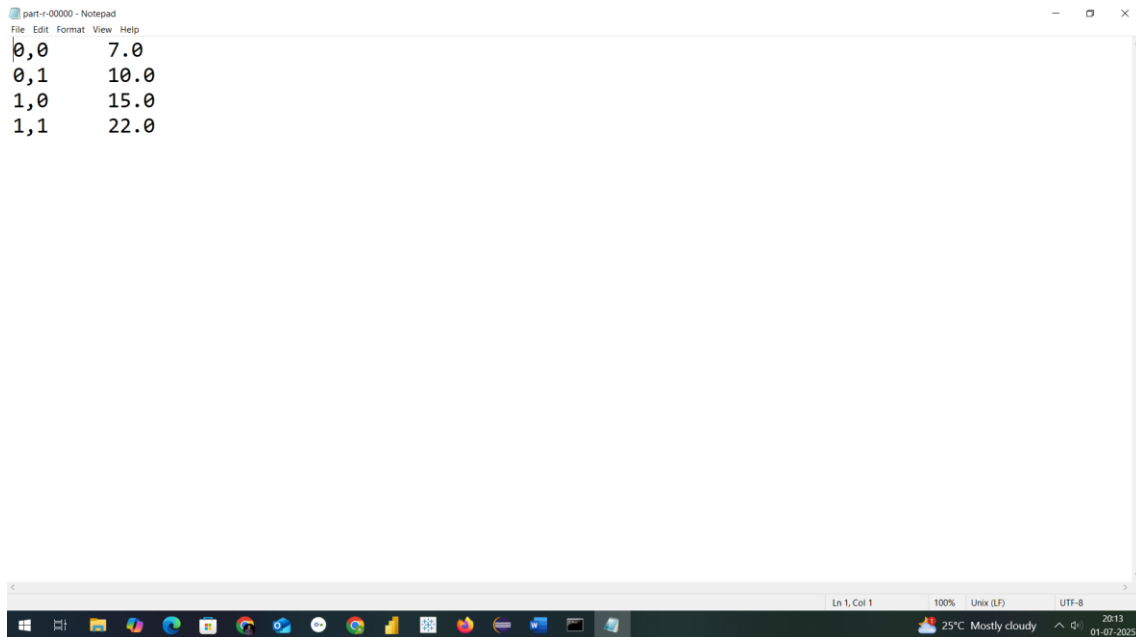


```
Administrator: Command Prompt
WRONG_MAP=0
WRONG_REDUCE=0
File Output Format Counters
  Bytes Written=47
2025-07-01 20:02:46,162 INFO mapred.LocalJobRunner: Finishing task: attempt_local209871933_0001_r_000000_0
2025-07-01 20:02:46,162 INFO mapred.LocalJobRunner: reduce task executor complete.
2025-07-01 20:02:46,215 INFO mapreduce.Job: Job job_local209871933_0001 running in uber mode : false
2025-07-01 20:02:46,222 INFO mapreduce.Job: map 100% reduce 100%
2025-07-01 20:02:46,226 INFO mapreduce.Job: Job job_local209871933_0001 completed successfully
2025-07-01 20:02:46,246 INFO mapreduce.Job: Counters: 30
File System Counters
  FILE: Number of bytes read=14608
  FILE: Number of bytes written=1289305
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
Map-Reduce Framework
  Map input records=8
  Map output records=24
  Map output bytes=288
  Map output materialized bytes=342
  Input split bytes=102
  Combine input records=0
  Combine output records=0
  Reduce input groups=8
  Reduce shuffle bytes=342
  Reduce input records=24
  Reduce output records=4
  Spilled Records=48
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=80
  Total committed heap usage (bytes)=343932928
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=70
File Output Format Counters
  Bytes Written=47
C:\WINDOWS\system32>type C:\matrixoutput1\part-r-00000
The system cannot find the path specified.
C:\WINDOWS\system32> C:\matrixoutput1\part-r-00000
```

THE OUTFORMAT WILL BE CREATED THIS TIME IN LOCAL FOLDER WHERE MENTIONED

----→ D:\hadoopprograms\matrixm\matrixoutput1

In part-r-00000 text file

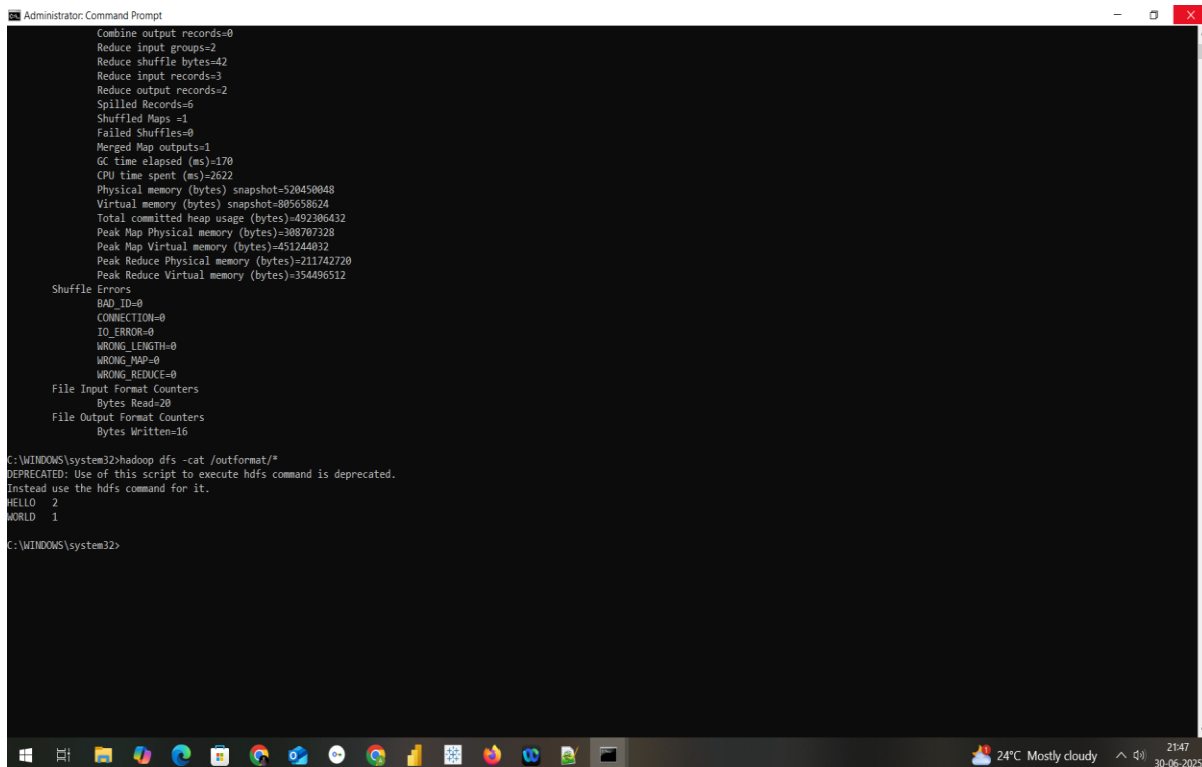


part-r-00000 - Notepad

0,0	7.0
0,1	10.0
1,0	15.0
1,1	22.0

Ln 1, Col 1 | 100% | Unix (LF) | UTF-8 | 2013 01-07-2025

NOW TO CHECK THE RESULT OF A PROGRAM



Administrator: Command Prompt

```
Combine output records=0
Reduce input groups=2
Reduce shuffle bytes=42
Reduce input records=3
Reduce output records=2
Spilled Records=6
Shuffled Maps=1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=170
CPU time spent (ms)=2622
Physical memory (bytes) snapshot=520450048
Virtual memory (bytes) snapshot=805658624
Total committed heap usage (bytes)=492386432
Peak Map Physical memory (bytes)=308707328
Peak Map Virtual memory (bytes)=451244032
Peak Reduce Physical memory (bytes)=211742720
Peak Reduce Virtual memory (bytes)=354496512

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=20
File Output Format Counters
  Bytes Written=16

c:\WINDOWS\system32>hadoop dfs -cat /outformat/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
HELLO  2
WORLD  1

c:\WINDOWS\system32>
```

24°C Mostly cloudy | 21:47 | 30-06-2025

### Program 3. Develop a Map Reduce program that mines weather data and displays appropriate messages indicating the weather conditions of the day.

**Step1:** goto Search button run cmd prompt as administrator

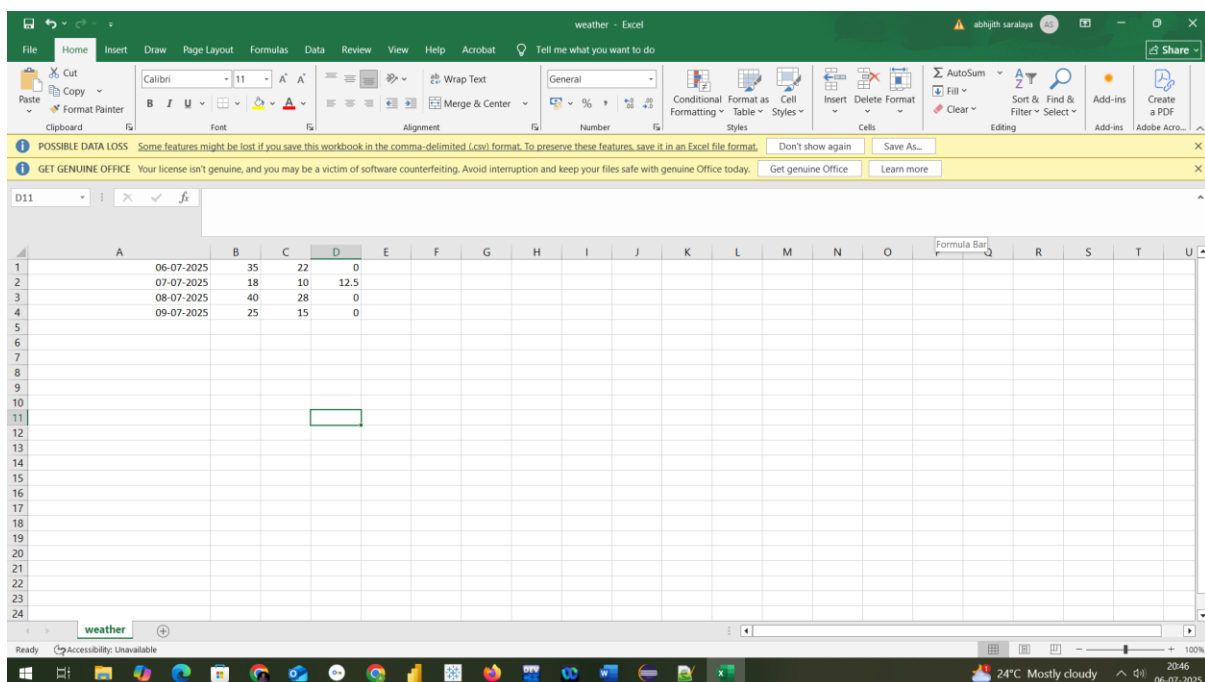
**Step2:** Initiate all required component of hadoop by command

**start-all.cmd**

**Step3:** Create a weather.csv file where we have folder running hadoop java program in my case I have created in Desktop folder D:\hadoopprograms\weatherinput

U can create anywhere as u wish with weather and content looks as in below

06-07-2025	35	22	0
07-07-2025	18	10	12.5
08-07-2025	40	28	0
09-07-2025	25	15	0



**Step3:** Now we need a jar file to execute mapreduce program fr wordcount

So open Eclipse

Create a java project with name MapReduceWeather

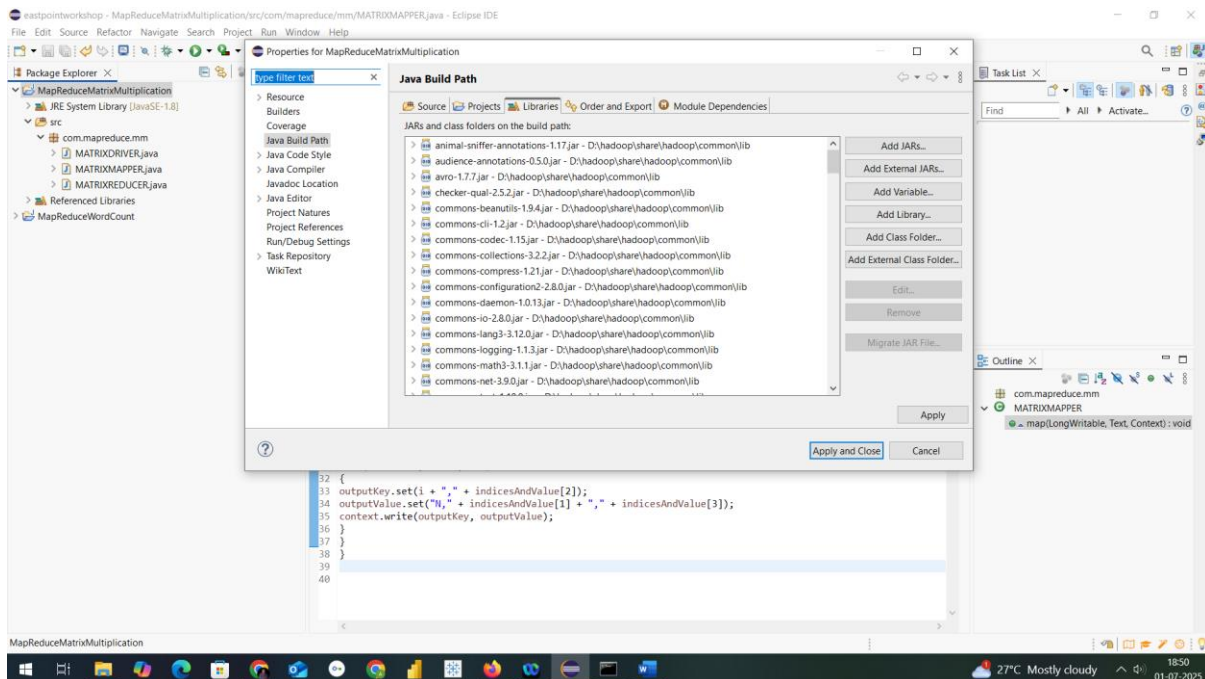
Select execution environment as JavaSE-1.8

And click on next&finish

**Step4:** Right Click on Projectname---→ Click on New---→create a package with name com.mapreduce.mm and click on Finish

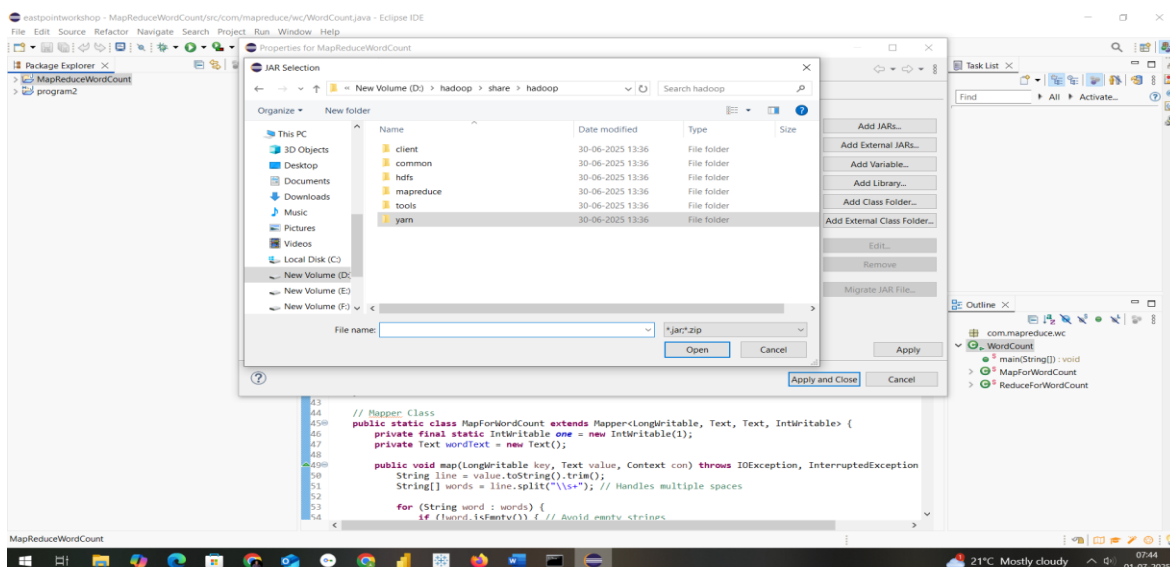
**Step5:** Add required libraries to support hadoop by navigating as in below

Right Click on Project-----→Right Click on BuildPath-----→ Configure Buildpath Then goto Libraries as in screen below



**Step6:** Add necessary External jar file from D:\hadoop\share\hadoop

Like clients,common,hdfs,mapreduce & yarn to support packages for hadoop



**Step7:** Create a class within package com.mapreduce.we with name WeatherAnalysis and paste this code

```
package com.mapreduce.we
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
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 WeatherAnalysis {

    public static class WeatherMapper extends Mapper<LongWritable, Text, Text, Text> {
        public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

            String line = value.toString();
            String[] fields = line.split(",");

            if (fields.length == 4) {
                String date = fields[0];
                try {
                    float tmax = Float.parseFloat(fields[1]);
                    float tmin = Float.parseFloat(fields[2]);
                    float prcp = Float.parseFloat(fields[3]);

                    String message = "";

                    if (tmax > 35) {
                        message += "Hot day ";
                    }
                    if (tmin < 15) {
                        message += "Cold day ";
                    }
                    if (prcp > 10) {
                        message += "Rainy day ";
                    }

                    if (message.isEmpty()) {
                        message = "Normal weather";
                    }
                }
            }
        }
    }
}
```

```
        context.write(new Text(date), new Text(message.trim()));
    } catch (NumberFormatException e) {
        // Ignore malformed rows
    }
}
}
}

public static class WeatherReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {
        for (Text val : values) {
            context.write(key, val); // One value per date
        }
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    conf.set("mapreduce.framework.name", "local");
    conf.set("fs.defaultFS", "file:///");
    conf.set("n", "3");

    Job job = Job.getInstance(conf, "Weather Analysis");

    job.setJarByClass(WeatherAnalysis.class);
    job.setMapperClass(WeatherMapper.class);
    job.setReducerClass(WeatherReducer.class);

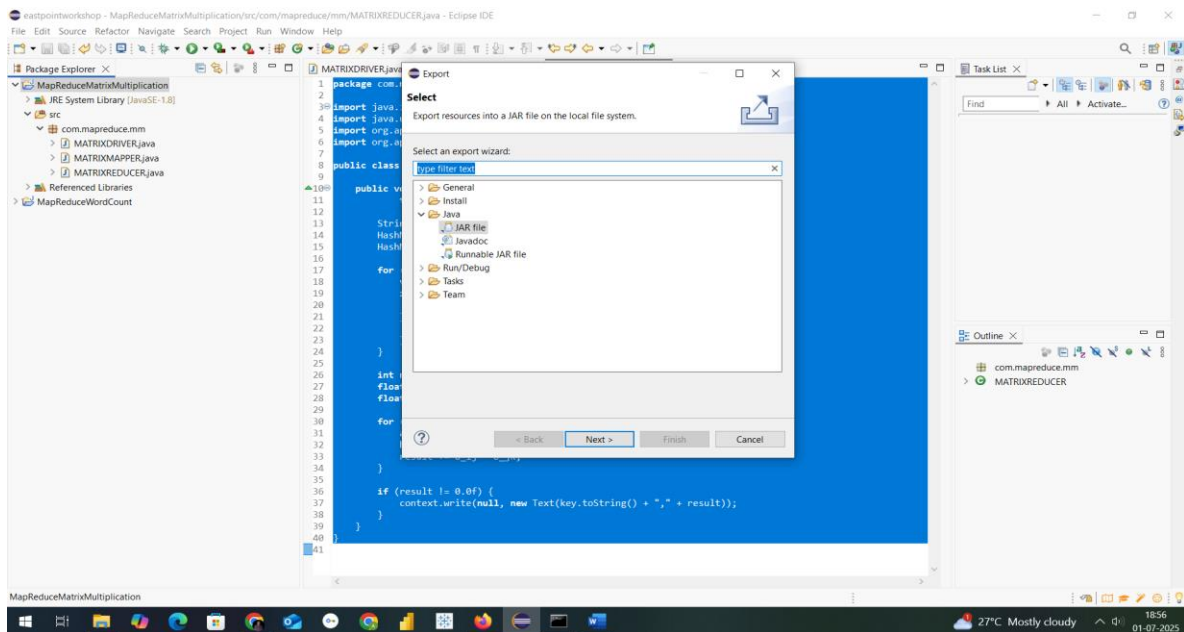
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    FileInputFormat.addInputPath(job, new Path(args[0])); // Input file path
    FileOutputFormat.setOutputPath(job, new Path(args[1])); // Output directory path

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

**Step8:** Procedure to create jar file from eclipse project

Now Right click on Project and click on export and select jar as in screen

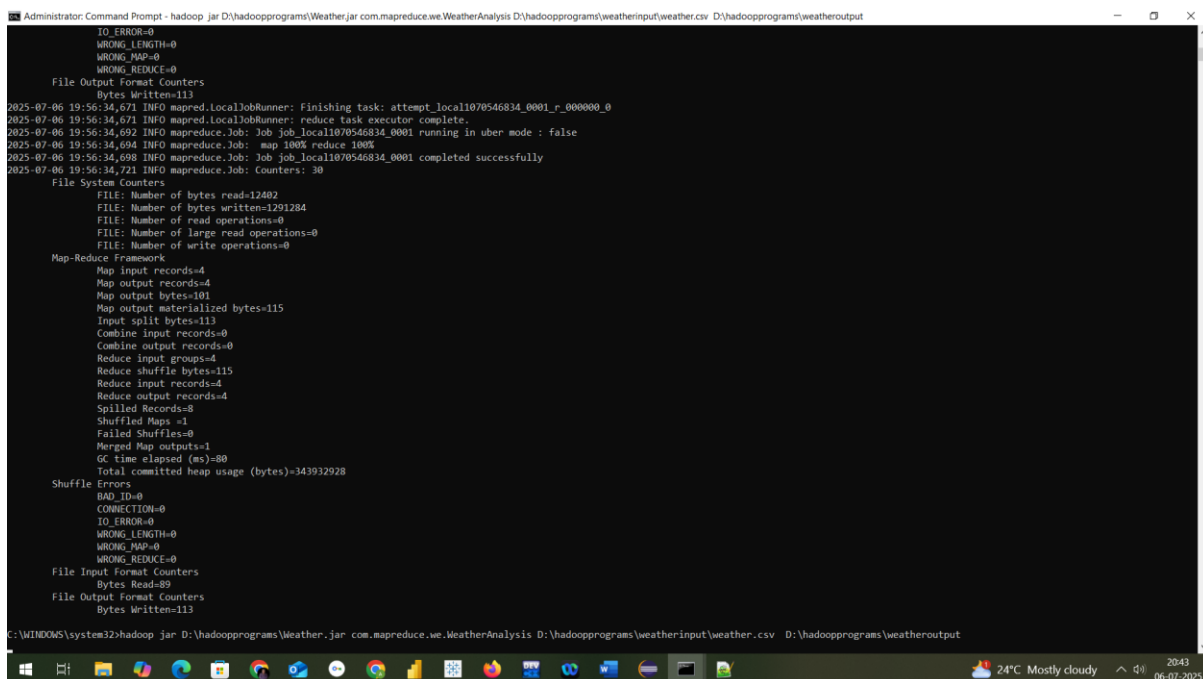


Mention the path where u want to save jar with name of Jar File in my case I have saved in D:\hadoopprograms as Weather.jar

Final step: Run the jar file with command as in below

hadoop jar D:\hadoopprograms\Weather.jar com.mapreduce.we.WeatherAnalysis  
D:\hadoopprograms\weatherinput\weather.csv D:\hadoopprograms\weatheroutput

After execution in command the output looks as in below screens

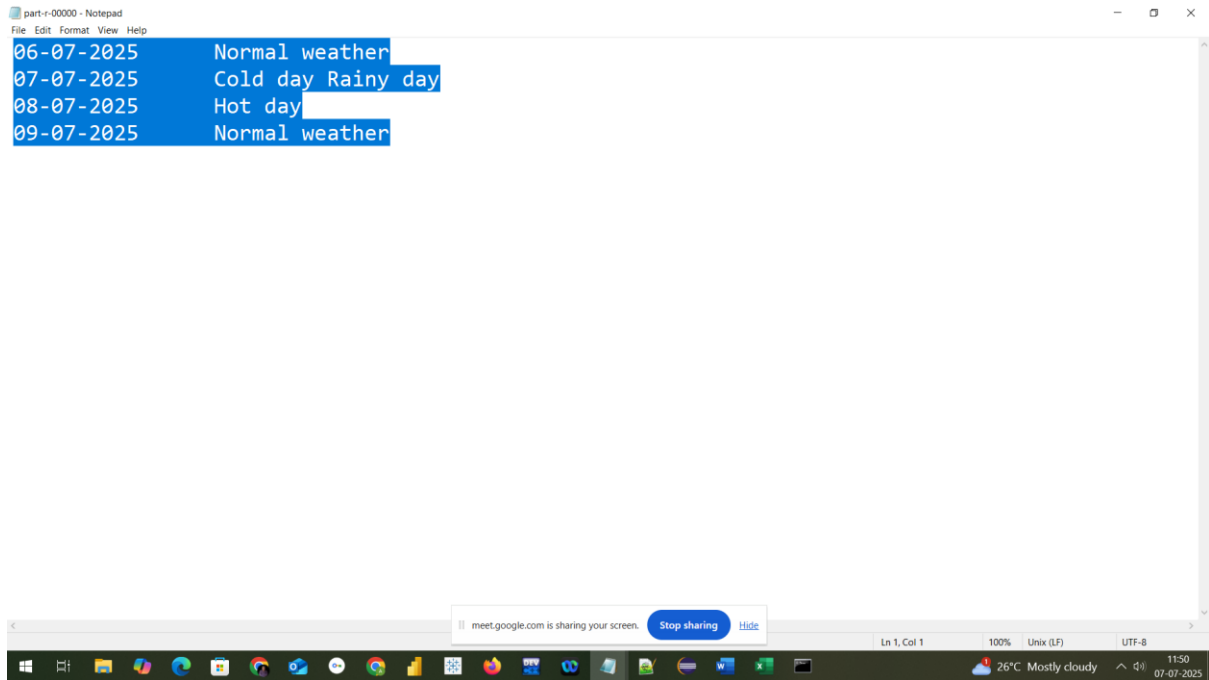


THE OUTPUT WILL BE CREATED THIS TIME IN LOCAL FOLDER WHERE MENTIONED

----> D:\hadoopprograms\weatheroutput

In part-r-00000 text file

NOW TO CHECK THE RESULT OF A PROGRAM



```
part-r-00000 - Notepad
File Edit Format View Help
06-07-2025 Normal weather
07-07-2025 Cold day Rainy day
08-07-2025 Hot day
09-07-2025 Normal weather
```

## Program 4: Develop a Map Reduce program to find the tags associated with each movie by analyzing movie lens data.

**Step1:** goto Search button run cmd prompt as administrator

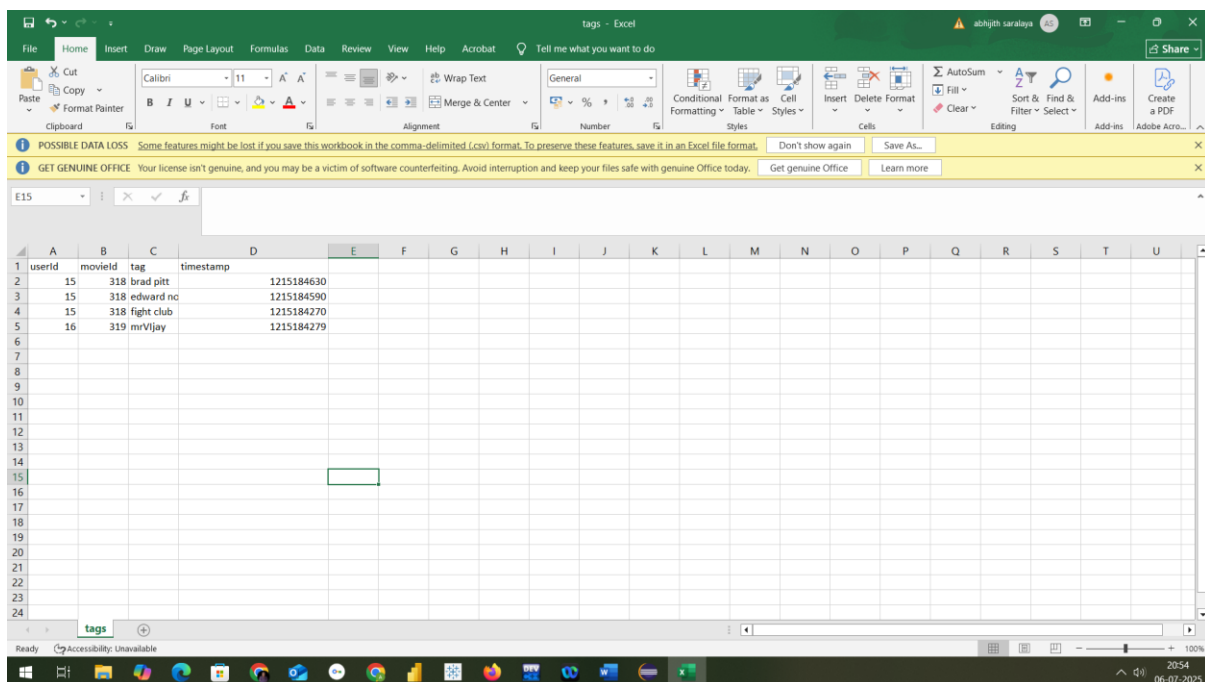
**Step2:** Initiate all required component of hadoop by command

**start-all.cmd**

**Step3:** Create a weather.csv file where we have folder running hadoop java program in my case I have created in Desktop folder D:\hadoopprograms\movietag

U can create anywhere as u wish with weather and content looks as in below

userId	movieId	tag	timestamp
15	318	brad pitt	1215184630
15	318	edward	1215184590
		norton	
15	318	fight club	1215184270
16	319	mrVijay	1215184279



**Step3:** Now we need a jar file to execute mapreduce program for wordcount

So open Eclipse

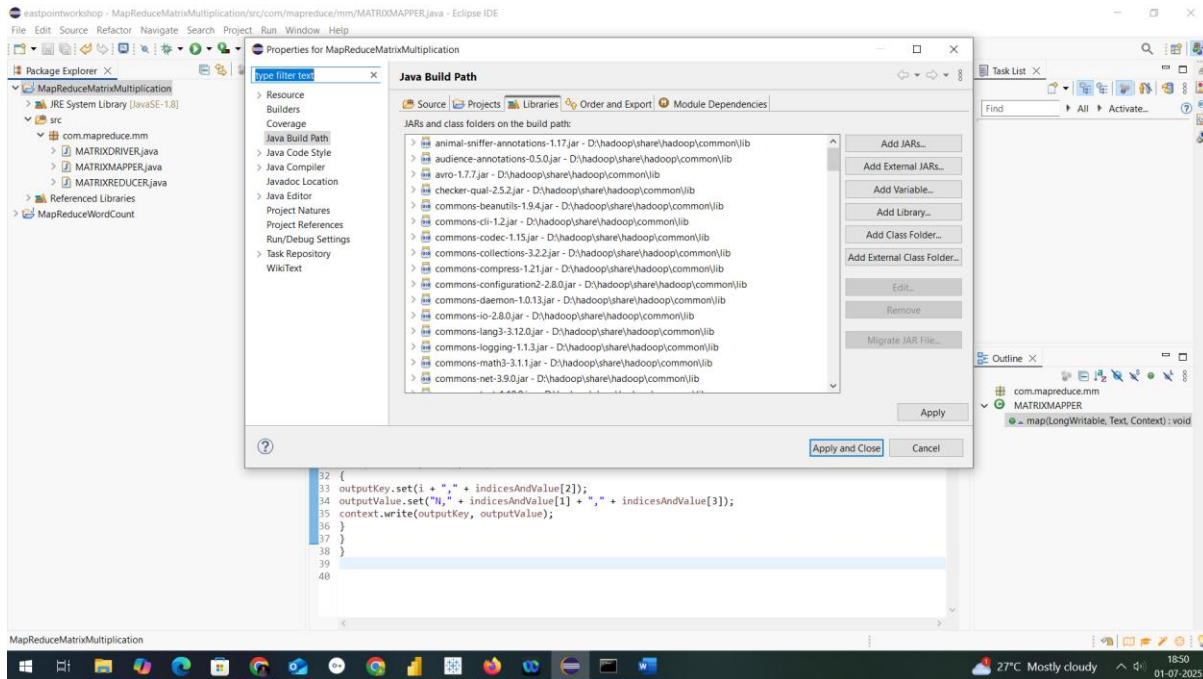
Create a java project with name MoviesTagsWithTitles

Select execution environment as JavaSE-1.8

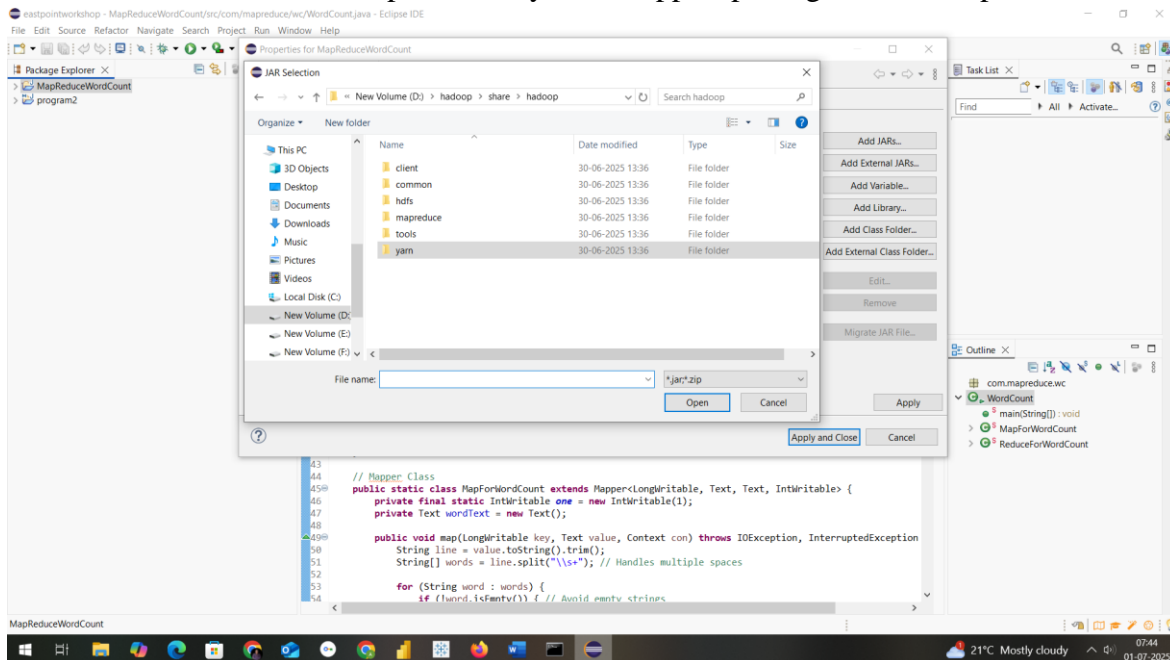
And click on next&finish

**Step4:** Right Click on Projectname---→ Click on New---→create a package with name com.mapreduce.mm and click on Finish

**Step5:** Add required libraries to support hadoop by navigating as in below  
Right Click on Project-----→Right Click on BuildPath-----→ Configure Buildpath Then goto Libraries as in screen below



**Step6:** Add necessary External jar file from D:\hadoop\share\hadoop  
Like clients,common,hdfs,mapreduce & yarn to support packages for hadoop



**Step7:** Create a class within package com.mapreduce.tag with name Movies Tags With Titles and paste this code

```
package com.mapreduce.tg
import java.io.IOException;
import java.util.HashSet;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MovieTagsWithTitles {

    public static class TagsMapper extends Mapper<LongWritable, Text, Text, Text> {
        public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
            String[] fields = value.toString().split(",", 4); // userId, movieId, tag, timestamp
            if (fields.length >= 3 && !fields[0].equals("userId")) { // skip header
                String movieId = fields[1].trim();
                String tag = fields[2].trim();
                context.write(new Text(movieId), new Text(tag));
            }
        }
    }

    public static class TagsReducer extends Reducer<Text, Text, Text, Text> {
        public void reduce(Text movieId, Iterable<Text> tags, Context context)
            throws IOException, InterruptedException {
            HashSet<String> uniqueTags = new HashSet<>();
            for (Text tag : tags) {
                uniqueTags.add(tag.toString());
            }
            context.write(movieId, new Text(String.join(", ", uniqueTags)));
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        conf.set("mapreduce.framework.name", "local");
        conf.set("fs.defaultFS", "file:///");
        conf.set("n", "3");
    }
}
```

```

Job job = Job.getInstance(conf, "Movie Tags By Movie ID");

job.setJarByClass(MovieTagsWithTitles.class);
job.setMapperClass(TagsMapper.class);
job.setReducerClass(TagsReducer.class);

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

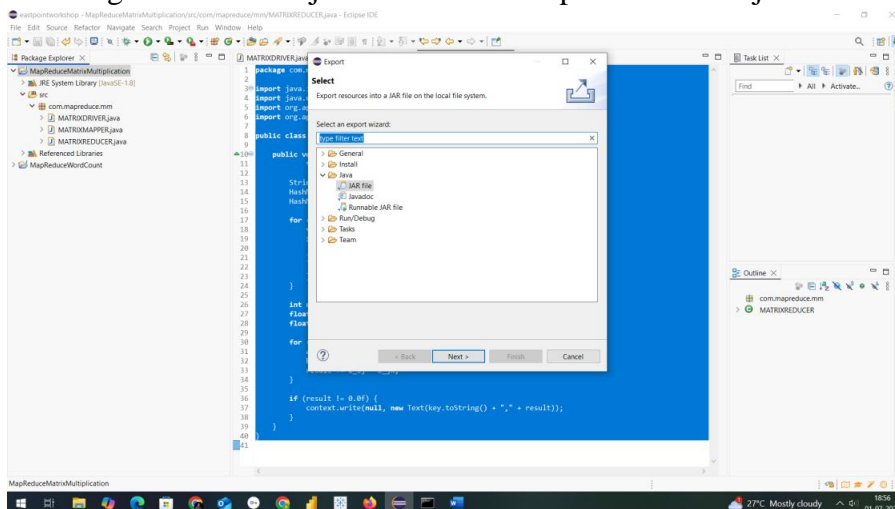
// Input/output from command-line args
FileInputFormat.addInputPath(job, new Path(args[0])); // path to tags.csv
FileOutputFormat.setOutputPath(job, new Path(args[1])); // output folder

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

```

### Step8: Procedure to create jar file from eclipse project

Now Right click on Project and click on export and select jar as in screen



Mention the path where u want to save jar with name of Jar File in my case I have saved in D:\hadoopprograms as Movie.jar

Final step: Run the jar file with command as in below

```

hadoop jar D:\hadoopprograms\Movie.jar com.mapreduce.tag.MovieTagsWithTitles
D:\hadoopprograms\movietag\tags.csv D:\hadoopprograms\movieoutput

```

After execution in command the output looks as in below screens

```
Administrator: Command Prompt - hadoop jar D:\hadoopprograms\Weather.jar com.mapreduce.we.WeatherAnalysis D:\hadoopprograms\weatherinput\weather.csv D:\hadoopprograms\weatheroutput
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Output Format Counters
  Bytes Written=115
2025-07-06 19:56:34,671 INFO mapred.LocalJobRunner: Finishing task: attempt_local1070546834_0001_r_000000_0
2025-07-06 19:56:34,692 INFO mapred.LocalJobRunner: reduce task executor complete.
2025-07-06 19:56:34,694 INFO mapreduce.Job: Job job_local1070546834_0001 running in uber mode : false
2025-07-06 19:56:34,698 INFO mapreduce.Job: map 100% reduce 100%
2025-07-06 19:56:34,721 INFO mapreduce.Job: Job job_local1070546834_0001 completed successfully
2025-07-06 19:56:34,721 INFO mapreduce.Job: Counters: 30
File System Counters
  FILE: Number of bytes read=12402
  FILE: Number of bytes written=1201284
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
Map-Reduce Framework
  Map input records=4
  Map output records=4
  Map output bytes=101
  Map output materialized bytes=115
  Input split bytes=113
  Combine input records=0
  Combine output records=0
  Reduce input groups=4
  Reduce shuffle bytes=115
  Reduce input records=4
  Reduce output records=4
  Spilled Records=4
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=80
  Total committed heap usage (bytes)=343932928
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=89
File Output Format Counters
  Bytes Written=115
C:\WINDOWS\system32>hadoop jar D:\hadoopprograms\Weather.jar com.mapreduce.we.WeatherAnalysis D:\hadoopprograms\weatherinput\weather.csv D:\hadoopprograms\weatheroutput
```

THE OUTFORMAT WILL BE CREATED THIS TIME IN LOCAL FOLDER WHERE MENTIONED

----→ D:\hadoopprograms\movieoutput

In part-r-00000 text file

NOW TO CHECK THE RESULT OF A PROGRAM

```
part-r-00000 - Notepad
File Edit Format View Help
318 edward norton, brad pitt, fight club
319 mrVijay
```

## Program 5: Implement Functions: Count – Sort – Limit – Skip – Aggregate using Mongo DB

### Sample Collection: students

Example document structure:

```
{
  "_id": 1,
  "name": "Alice",
  "age": 21,
  "marks": 85,
  "department": "CSE"
}
```

### 1. Count Documents

Count all documents:

```
db.students.countDocuments()
```

Count with a condition (e.g., marks > 80):

```
db.students.countDocuments({ marks: { $gt: 80 } })
```

### 2. Sort Documents

Sort by marks in descending order:

```
db.students.find().sort({ marks: -1 })
```

Sort by name ascending and age descending:

```
db.students.find().sort({ name: 1, age: -1 })
```

### 3. Limit Results

Return top 5 students:

```
db.students.find().limit(5)
```

### 4. Skip Documents

Skip first 5 documents and get the next 5:

```
db.students.find().skip(5).limit(5)
```

## 5. Aggregate Documents

### Group by Department and Count Students:

```
db.students.aggregate([ { $group: { _id: "$department", totalStudents: { $sum: 1 } } } ])
```

---

### Group by Department and Average Marks:

```
db.students.aggregate([ { $group: { _id: "$department", avgMarks: { $avg: "$marks" } } } ])
```

---

### Filter → Group → Sort → Limit (Full Pipeline):

```
db.students.aggregate([  
  { $match: { marks: { $gt: 60 } } },  
  { $group: { _id: "$department", avgMarks: { $avg: "$marks" } } },  
  { $sort: { avgMarks: -1 } },  
  { $limit: 3 }  
])
```

---

**Program 6 : Write Pig Latin scripts to sort, group, join, project, and filter the data.**

**Step1:** First create a Folder with name pigdata in D Drive then create students.txt with this content as in below

101, John, CS, 80  
102, Alice, EC, 90  
103, Bob, CS, 75  
104, David, EC, 85  
105, Eve, ME, 70

**Then add departments.txt and add below content**

CS, Computer Science  
EC, Electronics  
ME, Mechanical

**Step2 :** Now open cmd prompt as administrator and run command  
pig -x local

**Step3****A. LOADING OF DATA**

```
students = LOAD 'D:/pigdata/students.txt'  
  USING PigStorage(',')  
  AS (id:int, name:chararray, dept:chararray, marks:int);
```

```
departments = LOAD 'D:/pigdata/departments.txt'  
  USING PigStorage(',')  
  AS (code:chararray, dept_name:chararray);
```

**B. Project Specific Columns**

```
projected_data = FOREACH students GENERATE name, marks;  
  
DUMP projected_data;
```

**C. Filter Rows**

```
high_scorers = FILTER students BY marks > 80;
```

```
DUMP high_scorers;
```

**D. Group by Department**

```
grouped_by_dept = GROUP students BY dept;
```

```
DUMP grouped_by_dept;
```

**E. To get average marks per department:**

```
avg_marks = FOREACH grouped_by_dept GENERATE  
group AS dept, AVG(students.marks) AS avg_score;
```

```
DUMP avg_marks;
```

**F. Sort by Marks Descending**

```
sorted_students = ORDER students BY marks DESC;
```

```
DUMP sorted_students;
```

**G. Join Students with Departments**

```
joined_data = JOIN students BY dept, departments BY code;
```

```
DUMP joined_data;
```

**H. To project joined output:**

```
result = FOREACH joined_data GENERATE  
students::id,  
students::name,  
departments::dept_name,  
students::marks;
```

```
DUMP result;
```

**Program7: Use Hive to create, alter, and drop databases, tables, views, functions, and indexes**

First open localhost:8888

With docker running image in backend then follow below procedure

**Hive DDL Operations on Databases, Tables, Views, Functions, and Indexes****1. Databases**

- Create a Database:

```
CREATE DATABASE college_db;
```

- Use the Database:

```
USE college_db;
```

- Alter the Database:

```
ALTER DATABASE college_db SET DBPROPERTIES ('owner'='Abhijith');
```

- Drop the Database:

```
DROP DATABASE college_db;
```

```
DROP DATABASE college_db CASCADE; -- If it contains tables
```

**2. Tables**

- Create Table:

```
CREATE TABLE students (  
  id INT,  
  name STRING,  
  dept STRING,  
  marks INT  
)
```

```
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '  
STORED AS TEXTFILE;
```

- Load Data into Table:

```
LOAD DATA LOCAL INPATH '/home/user/students.txt' INTO TABLE students;
```

- Alter Table (Add column):

```
ALTER TABLE students ADD COLUMNS (email STRING);
```

- Drop Table:

```
DROP TABLE students;
```

### 3. Views

- Create View:

**CREATE VIEW high\_scorers AS SELECT name, marks FROM students WHERE marks > 80;**

- Use/View it:

**SELECT \* FROM high\_scorers;**

- Drop View:

**DROP VIEW high\_scorers;**

### 4. Functions

- Built-in Function Example:

**SELECT UPPER(name) FROM students;**

### 5. Indexes

- Create Index (Hive  $\leq 3.x$ ):

**CREATE INDEX student\_idx  
ON TABLE students (dept)  
AS 'COMPACT'  
WITH DEFERRED REBUILD;**

- Build Index:

**ALTER INDEX student\_idx ON students REBUILD;**

- Drop Index:

**DROP INDEX student\_idx ON students;**

## Program 8 :Implement a word count program in Hadoop and Spark.

### Word Count program using MapReduce

**Step1:** goto Search button run cmd prompt as administrator

**Step2:** Initiate all required component of hadoop by command

start-all.cmd

**Step3:** Create a text file where we have folder running hadoop java program in my case I have created in Desktop

U can create anywhere as u wish

**Step4:** Create a Directory in HDFS console using command

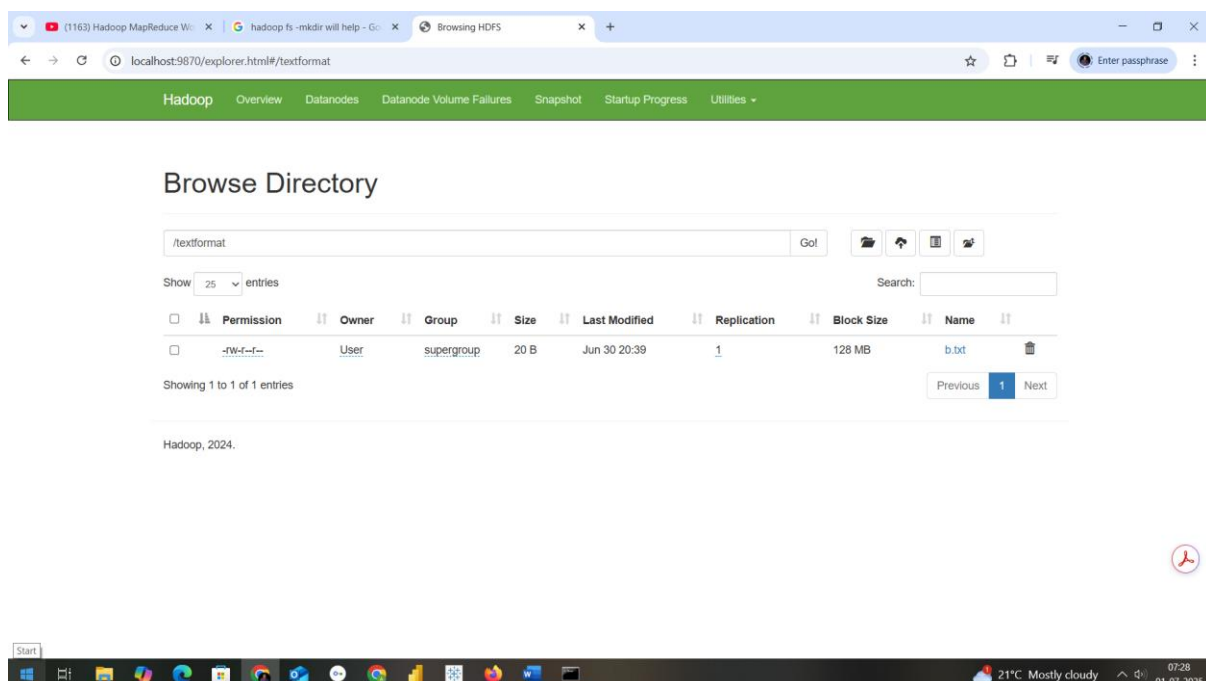
hadoop fs -mkdir /textformat

Where textform is ur input directory

**Step5:** We have to add text into a directory<textformat> in my case b.txt is in desktop folder copy the path and type command

hadoop fs -put C:\Users\User\Desktop\b.txt /textformat

After completion of above command u be able to see b.txt in /textformat directory of hdfs console



In cmd prmt if u want to check type the command as

hadoop fs -ls /textformat

**Step6:** Now we need a jar file to execute mapreduce program for wordcount

So open Eclipse

Create a java project with name MapReduceWordCount

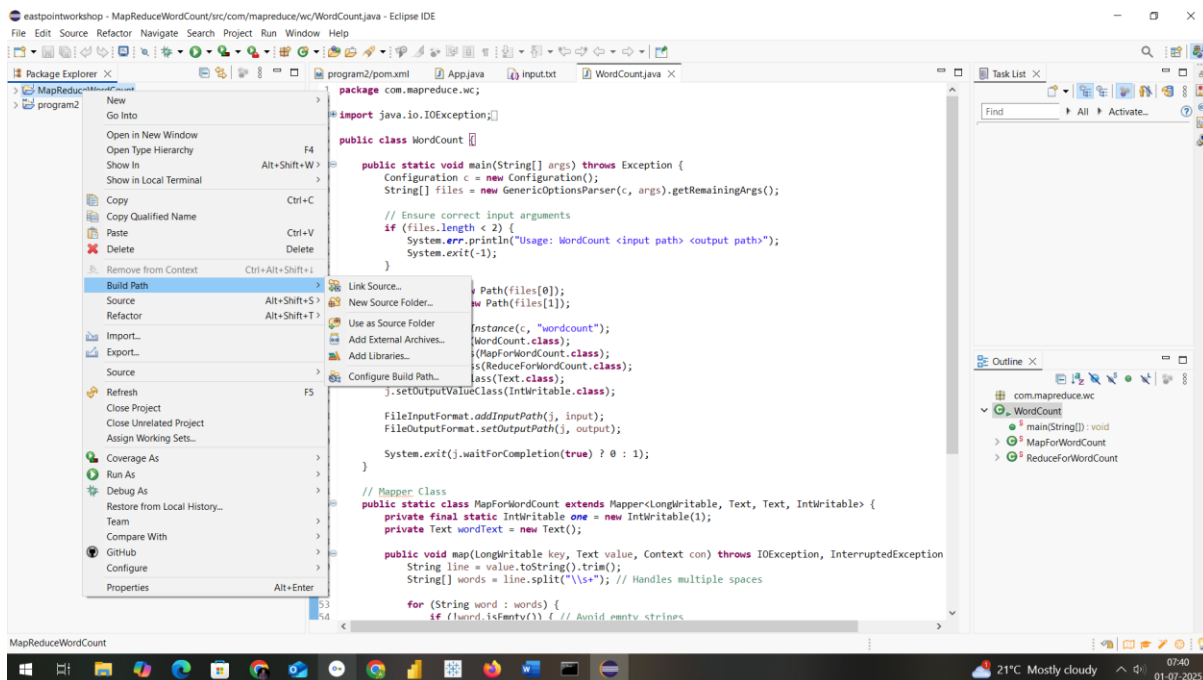
Select execution environment as JavaSE-1.8

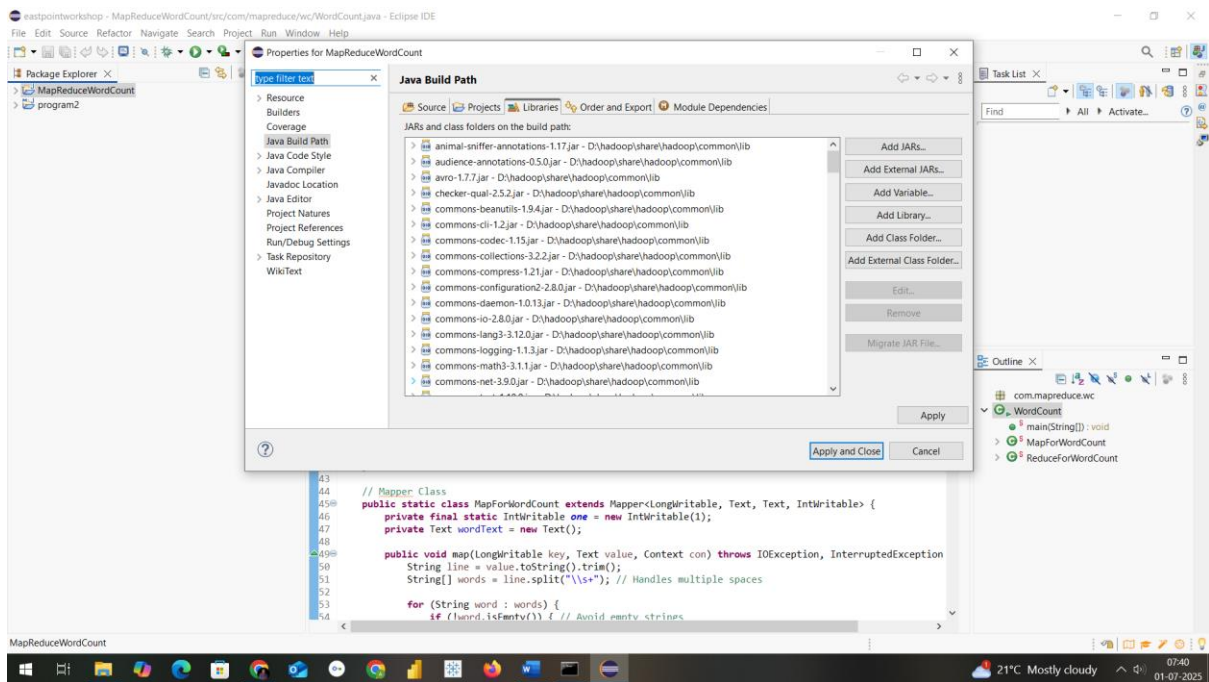
And click on next&finish

**Step7:** Right Click on Projectname---→ Click on New---→create a package with name com.mapreduce.wc and click on Finish

**Step8:** Add required libraries to support hadoop by navigating as in below

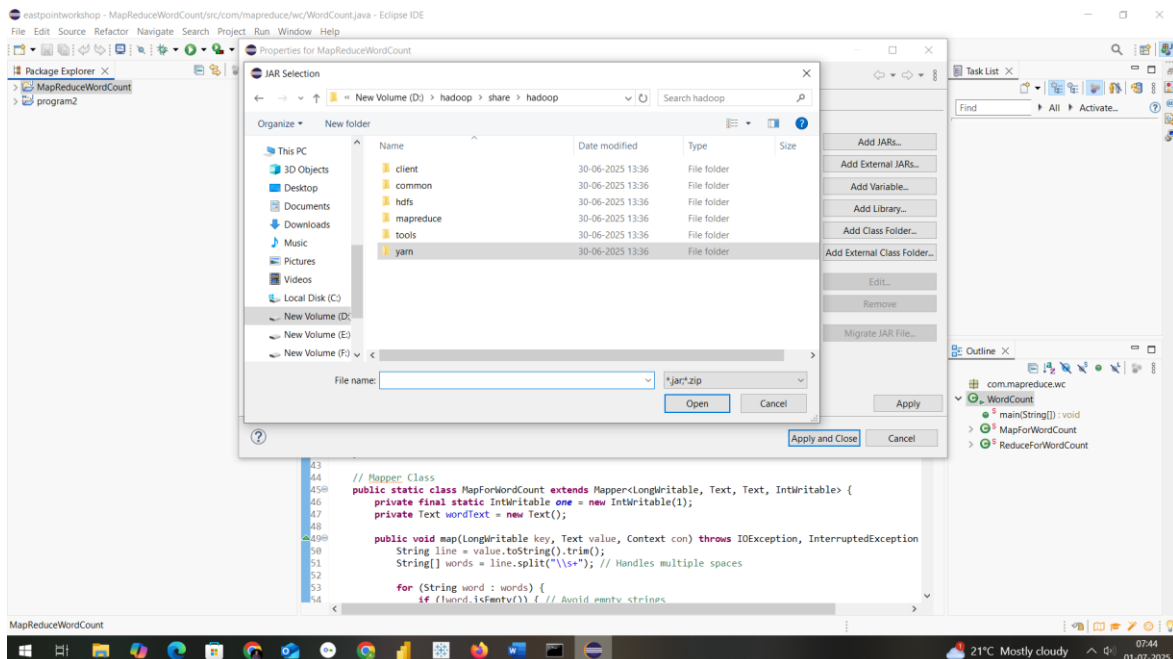
Right Click on Project-----→Right Click on BuildPath-----→ Configure Buildpath Then goto Libraries as in second screen below





**Step9:** Add necessary External jar file from D:\hadoop\share\hadoop

Like clients,common,hdfs,mapreduce & yarn to support packages for hadoop



**Step10:** Create a class within package com.mapreduce.wc with name WordCount and paste this code

```
package com.mapreduce.wc;
```

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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.util.GenericOptionsParser;
```

```
public class WordCount {
```

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

        // Ensure correct input arguments
        if (files.length < 2) {
            System.err.println("Usage: WordCount <input path> <output path>");
            System.exit(-1);
        }

        Path input = new Path(files[0]);
        Path output = new Path(files[1]);

        Job j = Job.getInstance(c, "wordcount");
        j.setJarByClass(WordCount.class);
        j.setMapperClass(MapForWordCount.class);
        j.setReducerClass(ReduceForWordCount.class);
        j.setOutputKeyClass(Text.class);
        j.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(j, input);
        FileOutputFormat.setOutputPath(j, output);

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

```
// Mapper Class
```

```
public static class MapForWordCount extends Mapper<LongWritable, Text, Text,
IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text wordText = new Text();

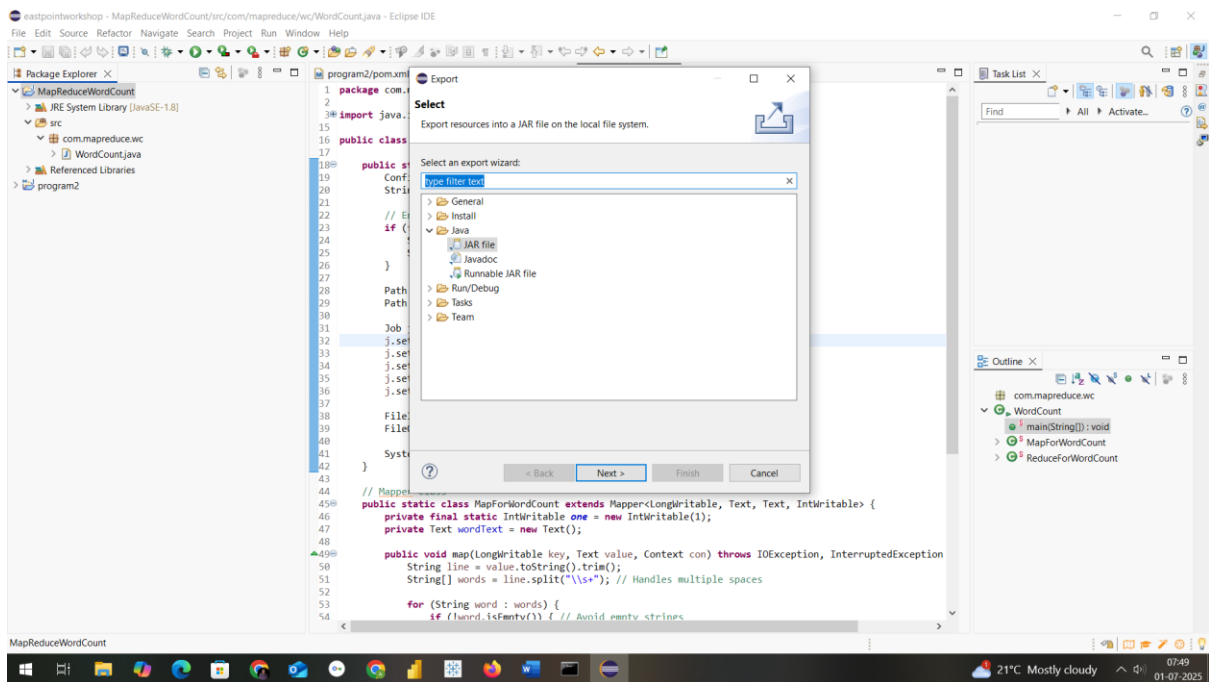
    public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException {
        String line = value.toString().trim();
        String[] words = line.split("\\s+"); // Handles multiple spaces

        for (String word : words) {
            if (!word.isEmpty()) { // Avoid empty strings
                wordText.set(word.trim().toUpperCase());
                con.write(wordText, one);
            }
        }
    }
}

// Reducer Class
public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text,
IntWritable> {
    public void reduce(Text word, Iterable<IntWritable> values, Context con) throws
IOException, InterruptedException {
        int sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }
        con.write(word, new IntWritable(sum));
    }
}
```

Step11: Procedure to create jar file from eclipse project

Now Right click on Project and click on export and select jar as in screen

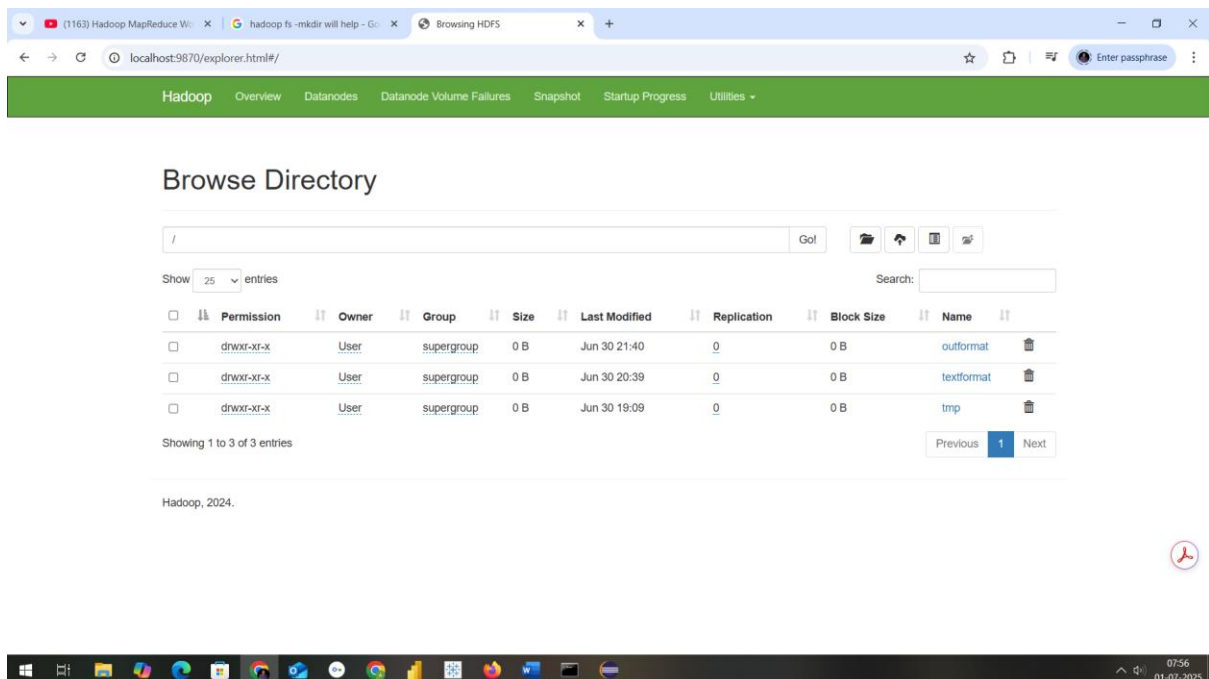


Mention the path where u want to save jar with name of Jar File in my case I have saved in D:\hadoopprograms as WordCountMapReduce

Final step: Run the jar file with command as in below

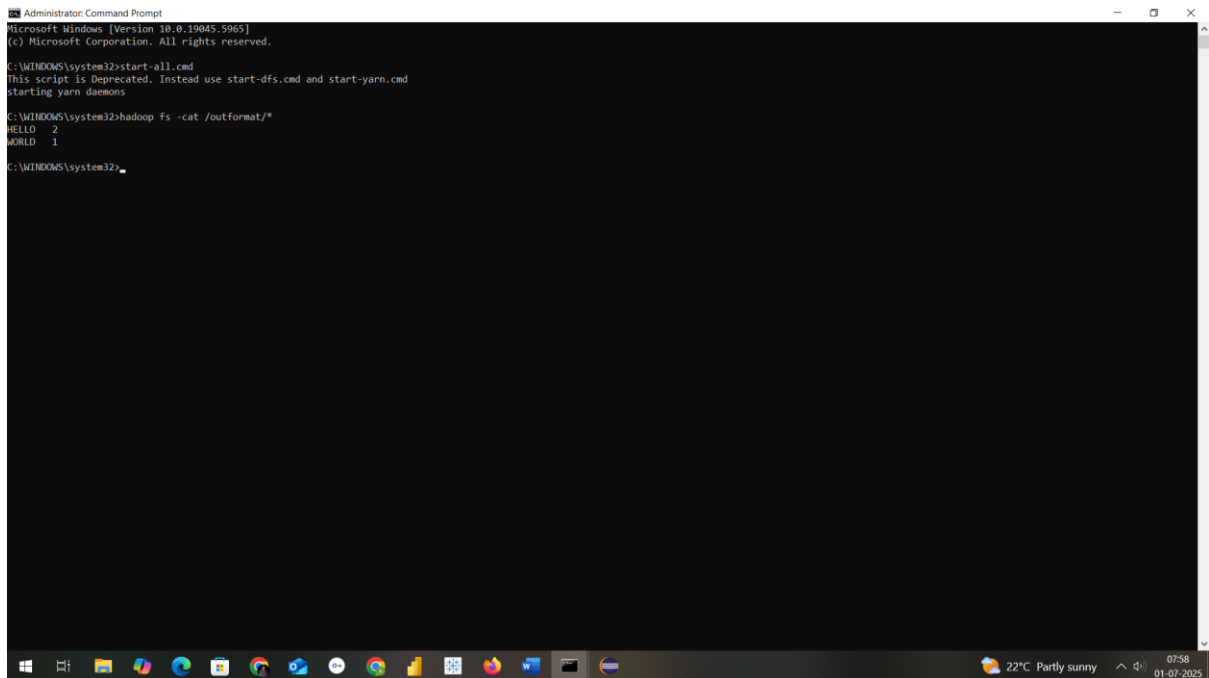
hadoop jar D:\hadoopprograms\WordCountMapReduce.jar com.mapreduce.wc/WordCount /textformat/b.txt /outformat

THE OUTFORMAT WILL BE CREATED IN CONSOLE



NOW TO CHECK THE RESULT OF WEATHER PROGRAM IS CORRECT TYPE

`hadoop fs -cat /outformat/*`



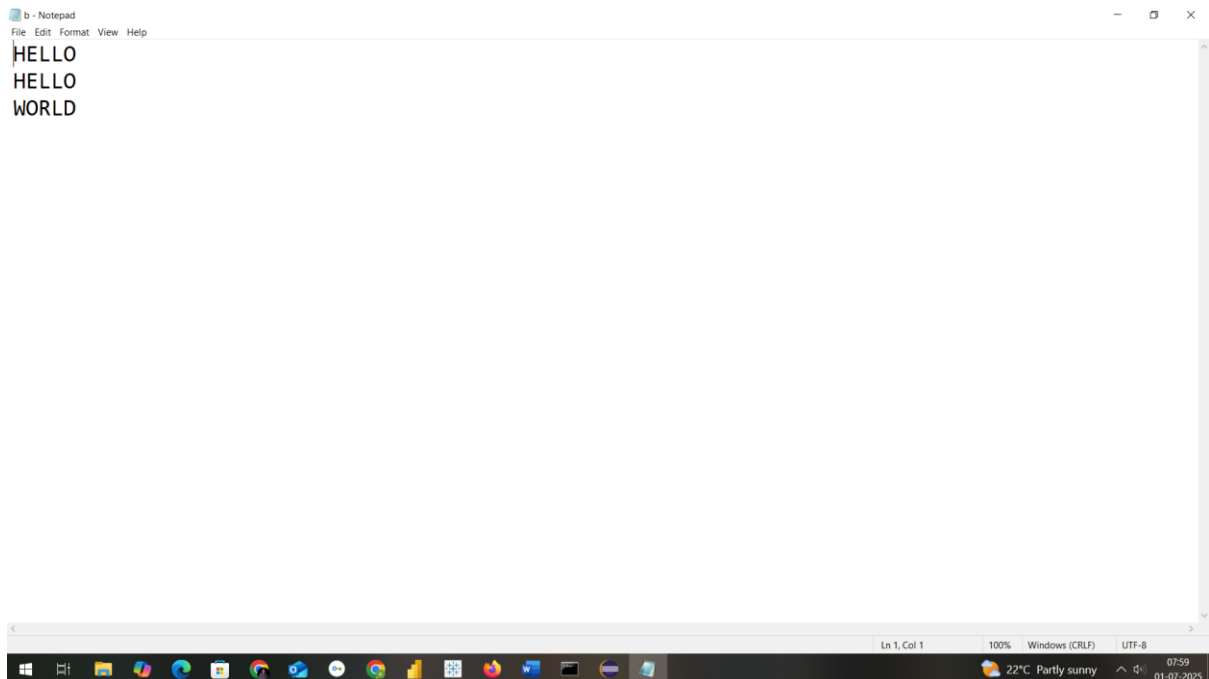
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.5965]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\WINDOWS\system32>hadoop fs -cat /outformat/*
HELLO 2
WORLD 1

C:\WINDOWS\system32>
```

AS YOU CAN SEE HELLO IS 2 in count and WORLD to 1 as my textformat(b.txt) is



```
b - Notepad
File Edit Format View Help
HELLO
HELLO
WORLD
```

## Word Program Count Procedure using Apache Spark

### Step 1: Prerequisites

Ensure you have the following installed:

- Java JDK (8 or 11)
- Eclipse IDE (preferably Eclipse IDE for Java Developers)
- Apache Maven

### Step 2: Create Maven Project in Eclipse

1. Open Eclipse → File → New → Maven Project → Next.
2. Select maven-archetype-quickstart, click Next.
3. Provide the following:
  - Group Id: com.example.spark
  - Artifact Id: spark-wordcount
  - Click Finish.

### Step 3: Add Spark Dependencies in pom.xml

Add this to your pom.xml:

```
<dependencies>
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.12</artifactId>
  <version>3.3.2</version>
</dependency>

<!-- Apache Spark SQL (Optional) -->
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-sql_2.12</artifactId>
  <version>3.3.2</version>
</dependency>

<!-- Logging (to avoid SLF4J warnings) -->
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.7.30</version>
</dependency>
```

```
</dependencies>

<build>
  <plugins>
    <!-- Plugin to build a fat JAR -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.2.4</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals><goal>shade</goal></goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

#### Step 4: Create WordCount Program

Create a class WordCount.java under:  
src/main/java/com/example/spark/WordCount.java

```
package com.example.spark.spark_wordcount;
import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;
import scala.Tuple2;
import java.util.Arrays;
```

```
public class App {
  public static void main(String[] args) {
    // Set Spark Configuration
    SparkConf conf = new SparkConf().setAppName("WordCount").setMaster("local[*]");
    JavaSparkContext sc = new JavaSparkContext(conf);

    // Load input file
    JavaRDD<String> input = sc.textFile("D:/hadoopprograms/input.txt"); // Replace with
your file path

    // FlatMap each line to words
    JavaRDD<String> words = input.flatMap(line -> Arrays.asList(line.split(" ")).iterator());
```

```
// Map each word to a pair (word, 1)
JavaPairRDD<String, Integer> wordPairs = words.mapToPair(word -> new
Tuple2<>(word, 1));

// Reduce by key (word)
JavaPairRDD<String, Integer> wordCounts = wordPairs.reduceByKey(Integer::sum);

// Print output
wordCounts.foreach(result -> System.out.println(result._1() + ": " + result._2()));

sc.close();
}
}
```

### Step 5: Input File

Create a input.txt file in folder where u r saving Hadoop programs in mine its in

D:\hadoopprograms

Example content:

hello world

hello spark

spark is fast

### Step 6: Run the Program

1. Right-click WordCount.java → Run As → Java Application.
2. Console should print:

hello: 2

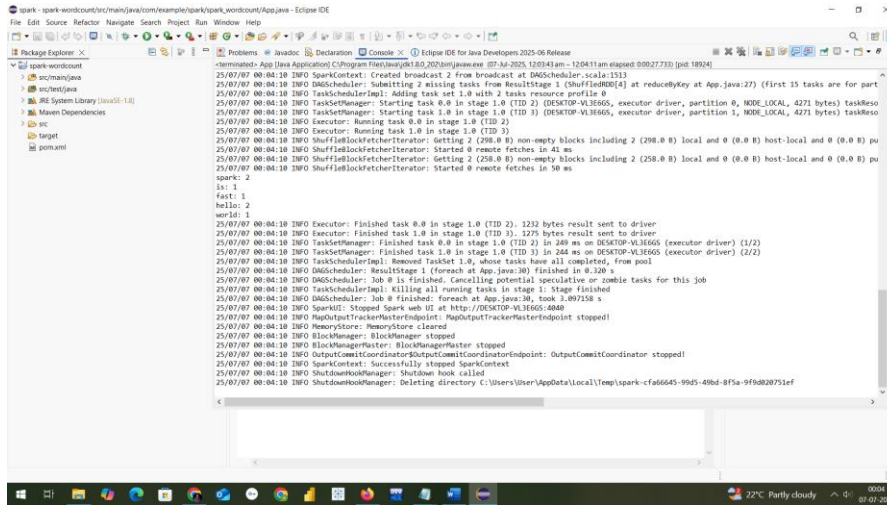
world: 1

spark: 2

is: 1

fast: 1

## RESULT as in below



```

-terminated: App [java Application] C:\Program Files\Java\jdk-11.0.20\bin\java.exe [07-Jul-2025, 12:03:41am - 12:04:11am elapsed: 0:00:27.733] [pid:18924]
25/07/07 00:04:10 INFO SparkContext: Created broadcast 2 from broadcast at DAGScheduler.scala:1511
25/07/07 00:04:10 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 1 (ShuffledRDD[4] at reduceByKey at App.java:27) (First 15 tasks are for part
25/07/07 00:04:10 INFO TaskSchedulerImpl: Adding task set 1.0 with 2 tasks resource profile 0
25/07/07 00:04:10 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 2) (DESKTOP-VL3E6G5, executor driver, partition 0, NODE_LOCAL, 4271 bytes) taskReso
25/07/07 00:04:10 INFO TaskSetManager: Starting task 1.0 in stage 1.0 (TID 3) (DESKTOP-VL3E6G5, executor driver, partition 1, NODE_LOCAL, 4271 bytes) taskReso
25/07/07 00:04:10 INFO Executor: Running task 0.0 in stage 1.0 (TID 2)
25/07/07 00:04:10 INFO Executor: Running task 1.0 in stage 1.0 (TID 3)
25/07/07 00:04:10 INFO ShuffleBlockFetcherIterator: Getting 2 (208.0 B) non-empty blocks including 2 (208.0 B) local and 0 (0.0 B) host-local and 0 (0.0 B) pu
25/07/07 00:04:10 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 41 ms
25/07/07 00:04:10 INFO ShuffleBlockFetcherIterator: Getting 2 (258.0 B) non-empty blocks including 2 (258.0 B) local and 0 (0.0 B) host-local and 0 (0.0 B) pu
25/07/07 00:04:10 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 50 ms
spark: 2
ls: 1
fast: 1
hello: 2
word: 1
25/07/07 00:04:10 INFO Executor: Finished task 0.0 in stage 1.0 (TID 2). 1232 bytes result sent to driver
25/07/07 00:04:10 INFO Executor: Finished task 1.0 in stage 1.0 (TID 3). 1275 bytes result sent to driver
25/07/07 00:04:10 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in 249 ms on DESKTOP-VL3E6G5 (executor driver) (1/2)
25/07/07 00:04:10 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 3) in 248 ms on DESKTOP-VL3E6G5 (executor driver) (2/2)
25/07/07 00:04:10 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool.
25/07/07 00:04:10 INFO DAGScheduler: ResultStage 1 (foreach at App.java:30) finished in 0.320 s
25/07/07 00:04:10 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
25/07/07 00:04:10 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
25/07/07 00:04:10 INFO DAGScheduler: Job 0 finished: foreach at App.java:30, task 1.007538 s
25/07/07 00:04:10 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-VL3E6G5:4040
25/07/07 00:04:10 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
25/07/07 00:04:10 INFO MemoryStore: MemoryStore cleared
25/07/07 00:04:10 INFO BlockManager: BlockManager stopped
25/07/07 00:04:10 INFO BlockManagerMaster: BlockManagerMaster stopped
25/07/07 00:04:10 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
25/07/07 00:04:10 INFO SparkContext: Successfully stopped SparkContext
25/07/07 00:04:10 INFO ShutdownHookManager: Shutdown hook called
25/07/07 00:04:10 INFO ShutdownHookManager: Deleting directory C:\Users\User\AppData\Local\Temp\spark-cfa66645-99d5-48bd-8f5a-9f9d828751ef
  
```

## **Program 9: Use CDH (Cloudera Distribution for Hadoop) and HUE (Hadoop User Interface) to analyze data and generate reports for sample datasets**

### **1. Start CDH and Hue Services**

1. Boot your Cloudera VM (e.g., Cloudera Quickstart VM).
2. Start Cloudera Manager if not already running.
3. Ensure services like HDFS, YARN, Hive, and Hue are running.

### **2. Access Hue**

Open a browser and navigate to: <http://localhost:8888>

Default login credentials:

Username: cloudera

Password: cloudera

### **3. Upload Sample Dataset**

1. Go to the File Browser in Hue.
2. Upload a CSV file (e.g., employee.csv) with content like:

```
id,name,dept,salary
1,John,IT,50000
2,Jane,HR,60000
3,Tom,IT,55000
```

### **4. Create Hive Table from Dataset**

Go to Query Editors → Hive and run:

```
CREATE TABLE IF NOT EXISTS employee (
  id INT,
  name STRING,
  dept STRING,
  salary INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

Load the CSV file into the table:

```
LOAD DATA INPATH '/user/cloudera/employee.csv' INTO TABLE employee;
```

## 5. Analyze Data Using Hive Queries

a) Count employees in each department:

```
SELECT dept, COUNT(*) AS emp_count FROM employee GROUP BY dept;
```

b) Find average salary per department:

```
SELECT dept, AVG(salary) AS avg_salary FROM employee GROUP BY dept;
```

c) Find maximum salary:

```
SELECT MAX(salary) FROM employee;
```

## 6. Generate Reports in Hue

1. Click on Dashboards or Charts in Hue.
2. Select a Hive query output.
3. Choose chart type (Bar, Pie, Table, etc.).
4. Save or export the report as PDF or image.

### Final Result

You can now visualize insights such as department-wise employee count, average salary distribution, and more using Hue's built-in charting and dashboard tools.

## Sample Viva Questions

### Big Data Concepts

1. What is Big Data?
2. What are the 5 V's of Big Data?
3. Difference between traditional RDBMS and Big Data systems?
4. What is the need for Hadoop in Big Data?
5. What are structured, semi-structured, and unstructured data? Give examples.

#### ▪ Hadoop and HDFS Viva Questions

6. What is Hadoop?
7. What are the main components of Hadoop?
8. What is HDFS? How is it different from a traditional file system?
9. What is the block size in HDFS?
10. What is NameNode and DataNode? What are their responsibilities?
11. How does HDFS handle fault tolerance?
12. How do you put data into HDFS from your local machine?
13. How do you view files in HDFS?

#### ▪ MapReduce Viva Questions

14. What is MapReduce?
15. Explain the working of MapReduce with an example.
16. What are the input and output formats in MapReduce?
17. What are Mapper and Reducer classes?
18. Can we have multiple mappers and reducers in a job?

#### ▪ Apache Hive Viva Questions

19. What is Hive?
20. What is the difference between Hive and RDBMS?
21. What is the use of HiveQL?
22. What are the different modes in Hive (local vs. distributed)?

23. What is a managed table vs. external table in Hive?
24. How do you load data into Hive tables?
25. How do partitions work in Hive?
26. Can we perform joins in Hive?

▪ **Apache Pig Viva Questions**

27. What is Apache Pig?
28. What is the difference between Pig and Hive?
29. What is Pig Latin?
30. What are relations in Pig?
31. How do you load and filter data in Pig?

▪ **Apache Spark Viva Questions**

32. What is Apache Spark? How is it different from Hadoop MapReduce?
33. What are RDDs (Resilient Distributed Datasets)?
34. Difference between RDD and DataFrame?
35. What is lazy evaluation in Spark?
36. What is the Spark execution flow?
37. What is a transformation and an action in Spark?
38. What is the role of SparkContext?

▪ **NoSQL (MongoDB / Cassandra) Viva Questions**

39. What is NoSQL? Why is it used in Big Data?
40. Difference between SQL and NoSQL databases?
41. What are the types of NoSQL databases?
42. What is a document store? Give an example.
43. What is a key-value store? Give an example.
44. What is sharding and replication in NoSQL?

▪ **Practical Scenario-Based Questions**

45. **How would you process a large CSV file in Hadoop?**
46. **How do you count word frequency using MapReduce?**
47. **How do you write a Hive query to get the top 5 products sold?**
48. **Explain how a join operation works in Hive.**
49. **How do you optimize performance in Spark jobs?**
50. **How do you handle missing or null data in a dataset?**