**(Affiliated to VTU, Belgaum and Approved by AICTE) DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**



# **LABORATORY MANUAL**

# **BIG DATA ANALYTICS LAB**

**[As per Choice Based Credit System (CBCS) scheme] (Effective from the academic year 2025 -2026)**

**BIS701**

# **PREPARED BY**

## **Esther Priscilla E**

**Assistant Professor**

**Department of Information Science &Engineering HKBK College of Engineering**

**Bangalore**

**Email: [priscillae.is@hkbk.edu.in](mailto:priscillae.is@hkbk.edu.in)**

## **Meena.S**

**Assistant Professor**

**Department of Information Science &Engineering HKBK College of Engineering**

**Bangalore**

**Email:meenas[.is@hkbk.edu.in](mailto:.is@hkbk.edu.in)**

# **HKBK COLLEGE OF ENGINEERING**

**Nagawara, Bengaluru -560 045 [www.hkbkeducation.org](http://www.hkbkeducation.org/)**



**Vision and Mission of the Institution**

**Vision**

**To empower students through wholesome education and enable the students to develop into highly qualified and trained professionals with ethics and emerge as responsible citizens with broad outlook to build a vibrant nation.**

## **Mission**

* **To achieve academic excellence through in-depth knowledge in science, engineering and technology through dedication to duty, innovation in teaching and faith in human values.**
* **To enable our students to develop into outstanding professionals with high ethical standards to face the challenges of the 21st century.**
* **To provide educational opportunities to the deprived and weaker section of the society, to uplift their socio-economic status.**

# **DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

**VISION**

**To train skilled and ethical professionals with the ability to plan, design, develop, organize and manage modern and traditional information systems with the knowledge of information technologies, services and organizations globally.**

**MISSION**

**To impart high quality engineering education in the field of Information Science and Technology with strong theoretical and extensive practical training methodologies through innovation and research to make world-class Engineers.**

**Programme Educational Objectives**



|  |  |
| --- | --- |
| **PEO-1** | **To provide students with a strong foundation in engineering fundamentals and in the computer science and engineering to work in the global**  **scenario.** |
| **PEO-2** | **To provide sound knowledge of programming and computing techniques and good communication and interpersonal skills so that they will be**  **capable of analyzing, designing and building innovative software systems.** |
| **PEO-3** | **To equip students in the chosen field of engineering and related fields to enable him to work in multidisciplinary teams.** |
| **PEO-4** | **To inculcate in students professional, personal and ethical attitude to relate engineering issues to broader social context and become responsible**  **citizen.** |
| **PEO-5** | **To provide students with an environment for life-long learning which allow them to successfully adapt to the evolving technologies throughout**  **their professional carrier and face the global challenges.** |

|  |  |
| --- | --- |
| **Programme Outcomes** | |
| **a.** | **Engineering Knowledge: Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.** |
| **b.** | **Problem Analysis: Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences** |
| **c.** | **Design/ Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet specified needs**  **with appropriate consideration for public health and safety, cultural, societal and environmental considerations.** |
| **d.** | **Conduct investigations of complex problems using research-based knowledge and research methods including design of experiments, analysis and**  **interpretation of data and synthesis of information to provide valid conclusions.** |
| **e.** | **Modern Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an under- standing of the limitations.** |
| **f.** | **The Engineer and Society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the**  **consequent responsibilities relevant to professional engineering practice.** |
| **g.** | **Environment and Sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate**  **knowledge of and need for sustainable development.** |
| **h.** | **Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.** |



|  |  |
| --- | --- |
| **i.** | **Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams and in multi disciplinary settings.** |
| **j.** | **Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to**  **comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.** |
| **k.** | **Life-long Learning: Recognize the need for and have the preparation and ability to engage in independent and life- long learning in the broadest context**  **of technological change.** |
| **l.** | **Project Management and Finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to**  **one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.** |
| **Programme Specific Outcomes** | |
| **m.** | **Problem-Solving Skills: An ability to investigate and solve a problem by analysis, interpretation of data, design and implementation through appropriate**  **techniques,tools and skills.** |
| **n.** | **Professional Skills: An ability to apply algorithmic principles, computing**  **skills and computer science theory in the modelling and design of computer- based systems.** |
| **o.** | **Entrepreneurial Ability: An ability to apply design, development principles and management skills in the construction of software product of varying complexity to become an entrepreneur** |

**BIS701**

## **List of Experiments**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiments** | **Page No.** |
| **1** | **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** | **7** |
| **2** | **Develop a MapReduce program to implement Matrix Multiplication** | **11** |
| **3** | **Develop a Map Reduce program that mines weather data and displays appropriate messages indicating the weather conditions of the day.** | **16** |
| **4** | **Develop a MapReduce program to find the tags associated with each movie by analyzing movie lens data.** | **20** |
| **5** | **Implement Functions: Count – Sort – Limit – Skip – Aggregate using MongoDB** | **24** |
| **6** | **Write Pig Latin scripts to sort, group, join, project, and filter the data** | **26** |
| **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** | **38** |

**Course objectives:**

**1. To implement MapReduce programs for processing big data.**

**2. To realize storage and processing of big data using MongoDB, Pig, Hive and Spark.**

**3. To analyze big data using machine learning techniques**

**Course outcomes (Course Skill Set):**

**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.**

**PROGRAM No: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.**

**HADOOP ENVIRONMENT SETUP**

**Hadoop Configuration**

Now, you need to configure some Hadoop files. If you have downloaded the same Hadoop version as me, then you need to go to etc\hadoop folder within the previously extracted Hadoop directory (in my case, the complete path is C:\hadoop-3.2.2\etc\hadoop). Once there, open the following five files with your preferred text editor:

* core-site.xml
* hadoop-env.cmd
* hdfs-site.xml
* mapred-site.xml
* yarn-site.xml

In the core-site.xml you need to set the default Hadoop File System location. Paste this chunk of code inside <configuration> tag:

<property>  
 <name>fs.defaultFS</name>  
 <value>hdfs://localhost:9000</value>  
</property>

n the hadoop-env.cmd file you need to provide the path to Java. This path was previously used when you set the JAVA\_HOME environment variable. In my case, I set the JAVA\_HOME value to : C:\Java\jdk1.8.0\_321\bin; but now, \bin folder must be removed from the path, i.e. I had to use C:\Java\jdk1.8.0\_321 path. You need to assign this path as JAVA\_HOME value around line 25:

Before you edit hdfs-site.xml file, you need to create some new folders. Go to Hadoop main directory in the root of your storage drive and create data folder inside of it:

Now, create the datanode and namenode folders inside the new data directory:

As you can see, datanode folder has the path C:\hadoop-3.2.2\data\datanode and namenode directory path is C:\hadoop-3.2.2\data\namenode.

Once this is done, you need to provide this folders paths as properties in the hdfs-site.xml file. You can copy the following chunk directly into <configuration> tag, just be careful to adjust the datanode and namenode paths according to your machine locations:

<property>  
 <name>dfs.replication</name>  
 <value>1</value>  
</property><property>  
 <name>dfs.namenode.name.dir</name>  
 <value>C:\hadoop-3.2.2\data\namenode</value>  
</property><property>  
 <name>dfs.datanode.data.dir</name>  
 <value>C:\hadoop-3.2.2\data\datanode</value>  
</property>

In the mapred-site.xml file you need to set yarn as the MapReduce framework. Copy the following code inside <configuration> tag:

<property>  
 <name>mapreduce.framework.name</name>  
 <value>yarn</value>  
</property>

Hadoop Environment Variable Configuration

Once all five files are properly edited, now you need to create an environment variable for Hadoop. Open the environment variables window as you did when you created JAVA\_HOME variable (recall you can open the Environment Variables typing *"Edit the system environment variables"* in the Windows search bar) and create the HADOOP\_HOME variable, assign the Hadoop bin folder path as its value (in my case: C:\hadoop-3.2.2\bin):

Now, you need to edit the Path system variable to add paths to bin and sbin folders of Hadoop. Both folders are in the root directoryof Hadoop. So, bin path is the same you've just assigned to HADOOP\_HOME variable (C:\hadoop-3.2.2\bin); sbin path, in my case will be C:\hadoop-3.2.2\sbin :

Once done, be careful and click on OK in all windows related to the environment variables to save the changes; otherwise, you will need to repeat this process again.

Fix of Hadoop ‘bin’ Folder

Now, you need to fix some configuration files. To do it, you need to replace the Hadoop bin folder with another bin folder which already contains all the files properly configured. First, download this compressed file [(hadoop3\_xFixedbin.rar)](https://drive.google.com/file/d/1nCN_jK7EJF2DmPUUxgOggnvJ6k6tksYz/view?usp=sharing" \t "_blank). Then, you need to delete bin folder:

Now, you can check the new and fixed bin folder is in Hadoop root:

And that’s it, you now have Hadoop File System configured on your computer.

Hadoop Installation Verification

Finally, to check if Hadoop is working properly you need to run it. To do so, open a command prompt as administrator. Recall you can do this typing *“Command Prompt”* in the Windows search bar:

Now, you need to go to the sbin directory inside hadoop folder; in my case, sbin directory is in C:\hadoop-3.2.2\sbin. Once you have typed this path press Enter:

Then, write the command start-all.cmd and press Enter:

You will see that several command prompts will open. If Hadoop is properly configured, then this four command prompts will remain open and running:

* hadoop datanode
* hadoop namenode
* yarn resourcemanager
* yarn nodemanager

**HDFS File Management Tasks: Read, Write, Delete**

# **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 No:2**

**Problem Statement : Matrix Multiplication program using MapReduce**

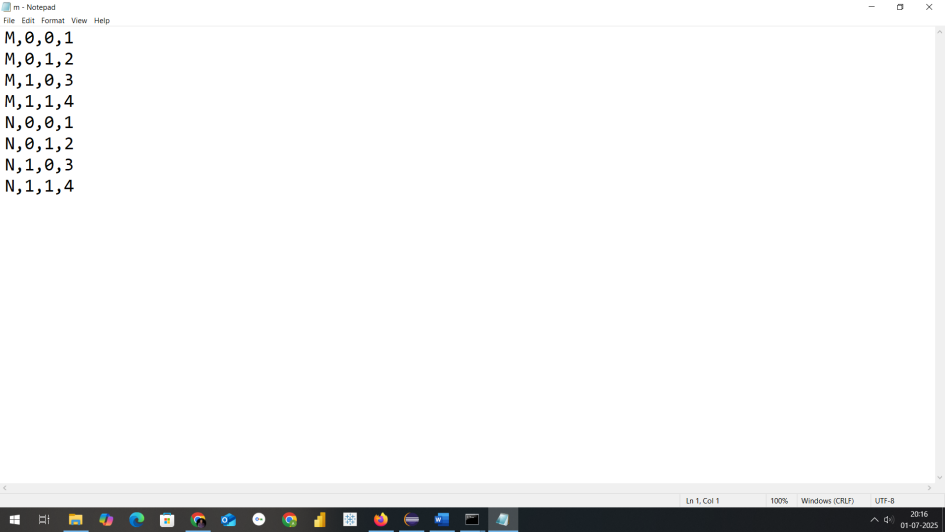
Step1:goto Search button run cmd prompt as administrator

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 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 mapreduce program fr wordcount

So open Eclipse

Create a java project with name MapReduceMatrixMultiplication

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.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.getOrDefault(k, 0.0f);

float b = hashB.getOrDefault(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

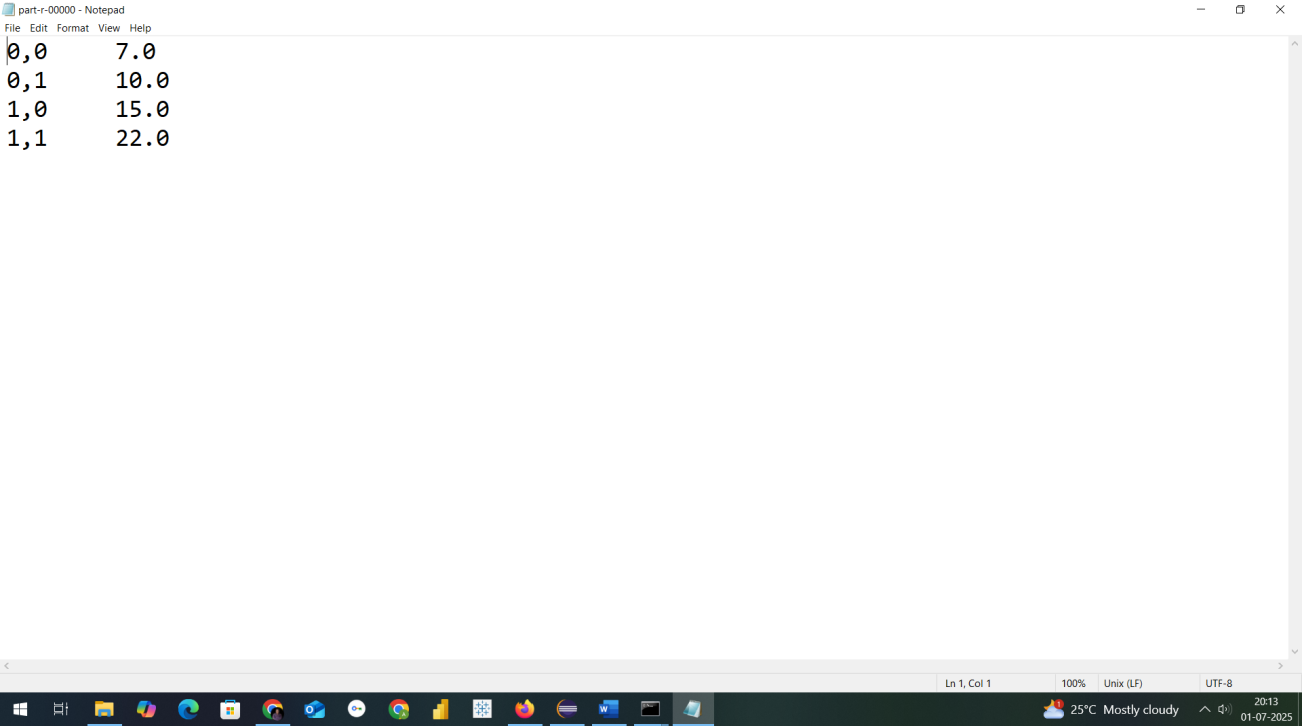
hadoop jar D:\hadoopprograms\MatrixMultiplication.jar com.mapreduce.mm.MatrixMultiplication D:/hadoopprograms/matrixm/m.txt D:/hadoopprograms/matrixm/matrixoutput1

After execution in command the output looks as in below screens

THE OUTFORMAT WILL BE CREATED THIS TIME IN LOCALFOLDER WHERE MENTIONED

----🡪 D:\hadoopprograms\matrixm\matrixoutput1

In part-r-00000 text file

****

**PROGRAM NO: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

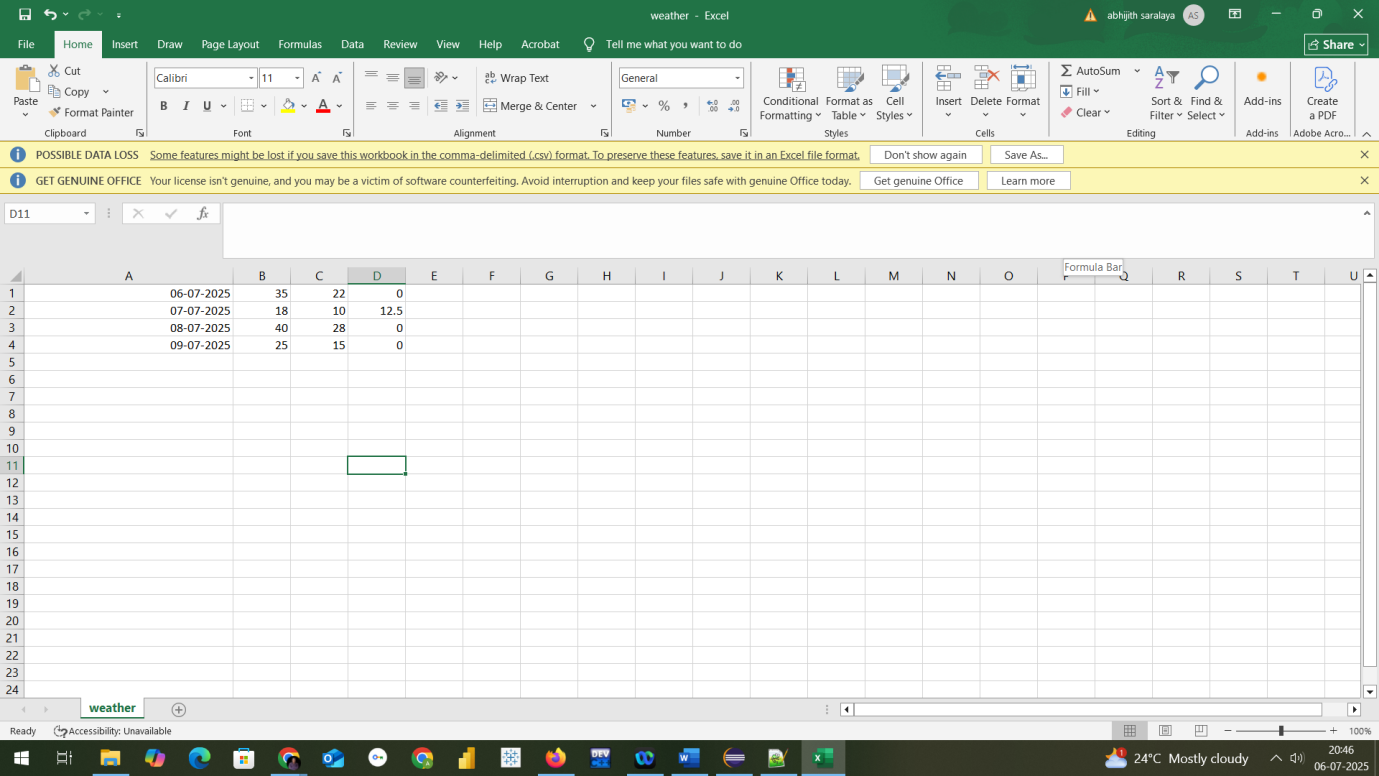
Step2: Initiate all required component of hadoop by command

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

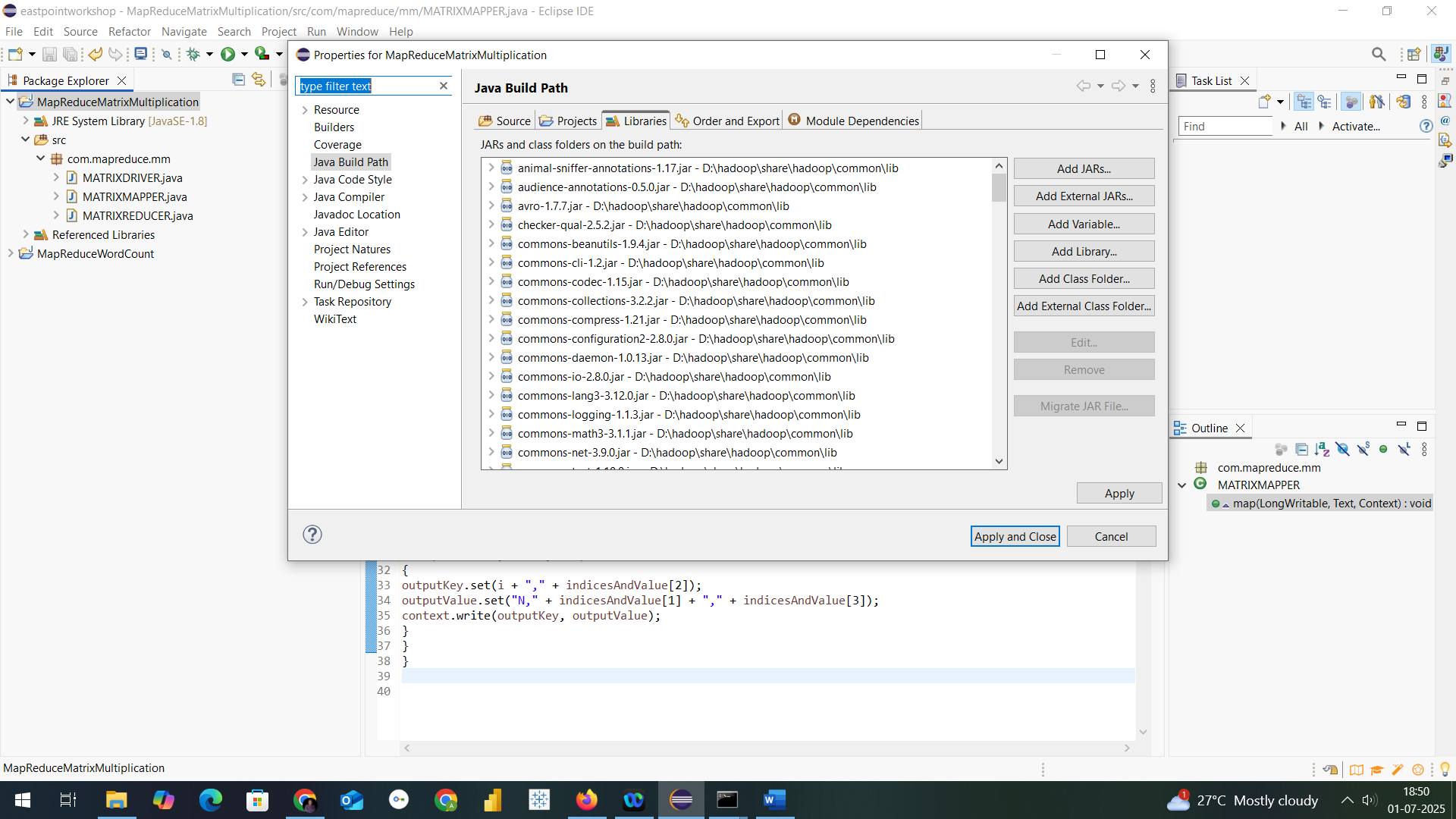
Select execution environment as JavaSE-1.8

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

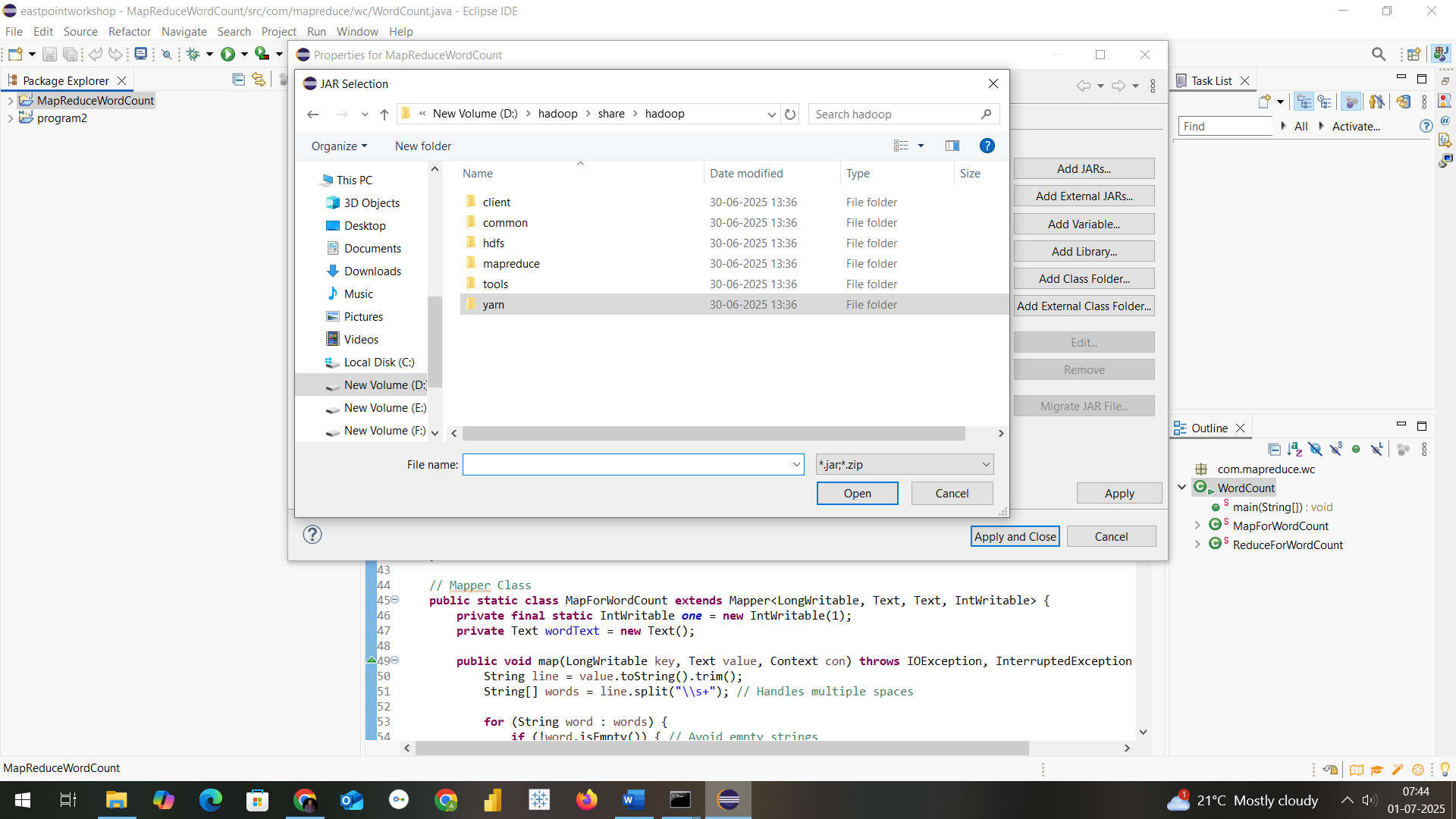
Step5: Add required libraries to support hadoop by navigating as in below

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

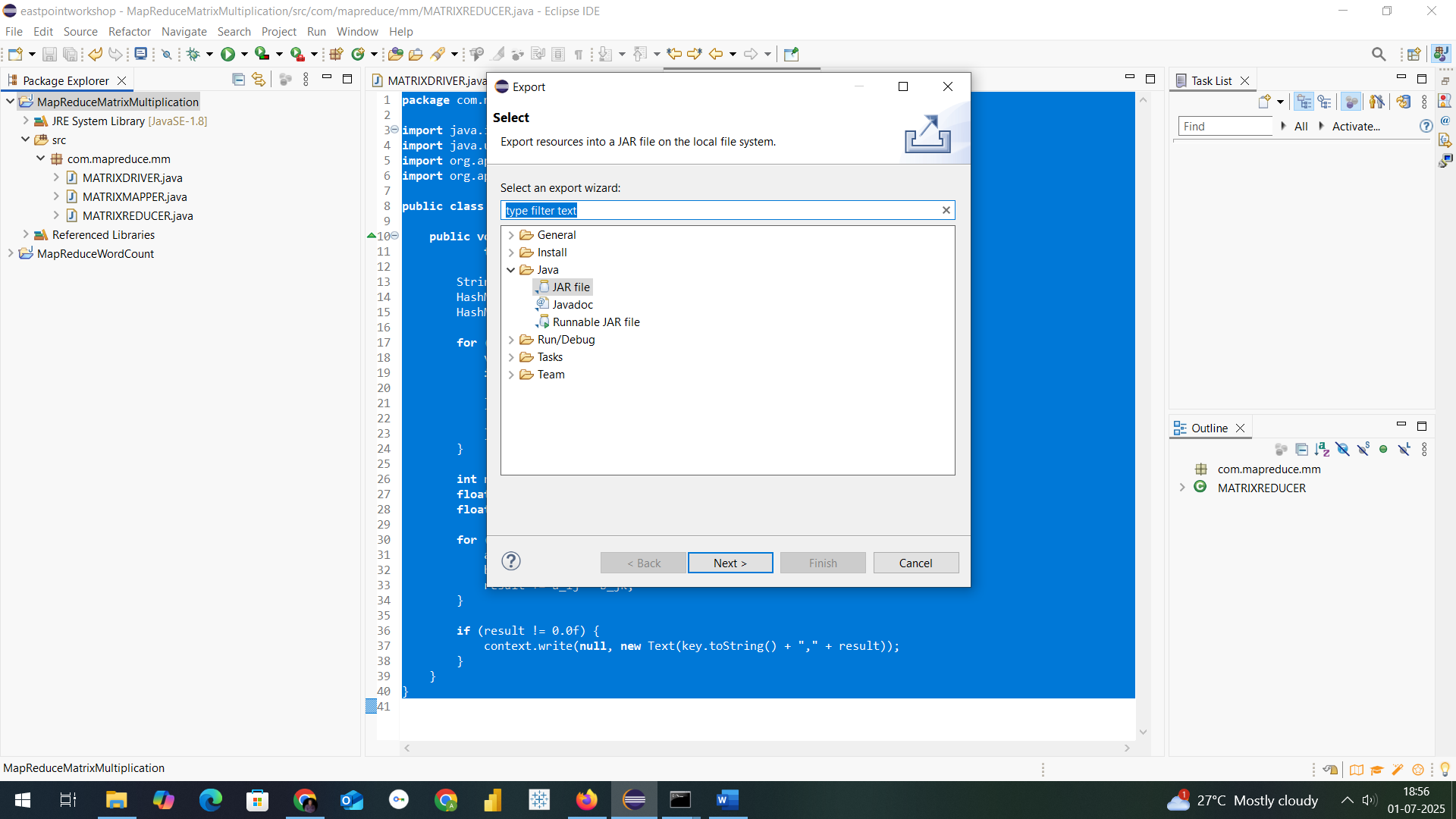
System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

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

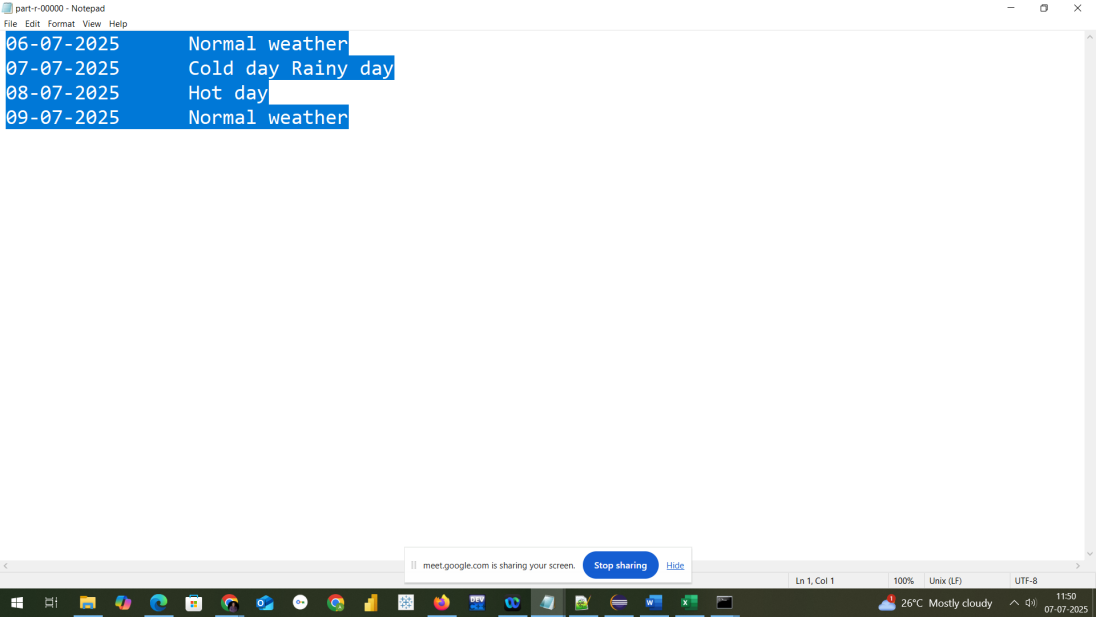
hadoop jar D:\hadoopprograms\Weather.jar com.mapreduce.we.WeatherAnalysis D:\hadoopprograms\weatherinput\weather.csv D:\hadoopprograms\weatheroutput

**THE OUTPUT FORMAT WILL BE CREATED THIS TIME IN LOCALFOLDER WHERE MENTIONED**

**----🡪 D:\hadoopprograms\weatheroutput**

**In part-r-00000 text file**

**NOW TO CHECK THE RESULT OF A PROGRAM**

****

**PROGRAM NO:4**

**Problem Statement : Develop a MapReduce 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 content looks as in below

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

Step3: So open Eclipse

Create a java project with name MoviesTagsWithTitles

Select execution environment as JavaSE-1.8

And click on next&amp;finish

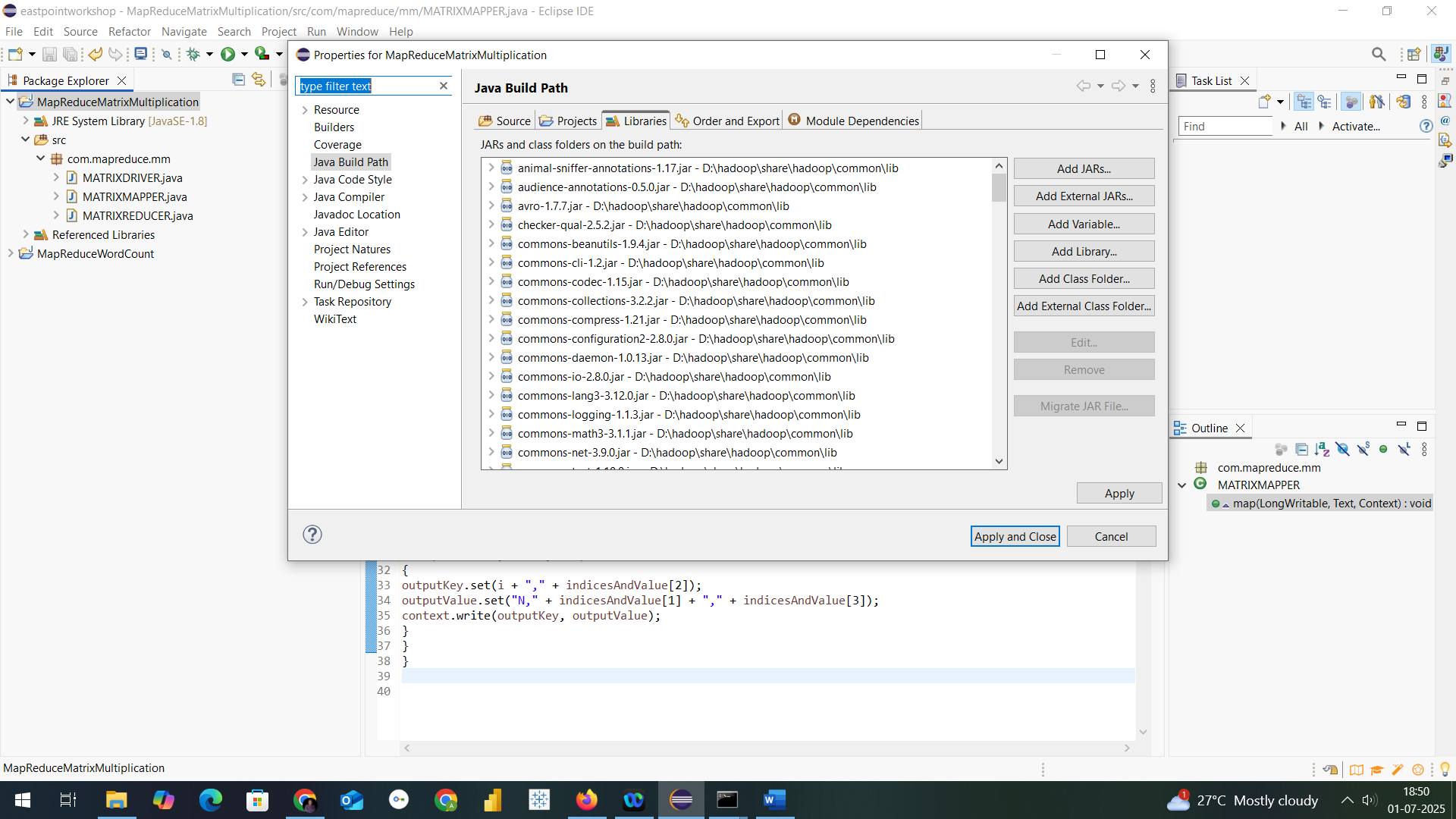
Step4: Right Click on Projectname--- Click on New---create a package

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 MoviesTagsWithTitles 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

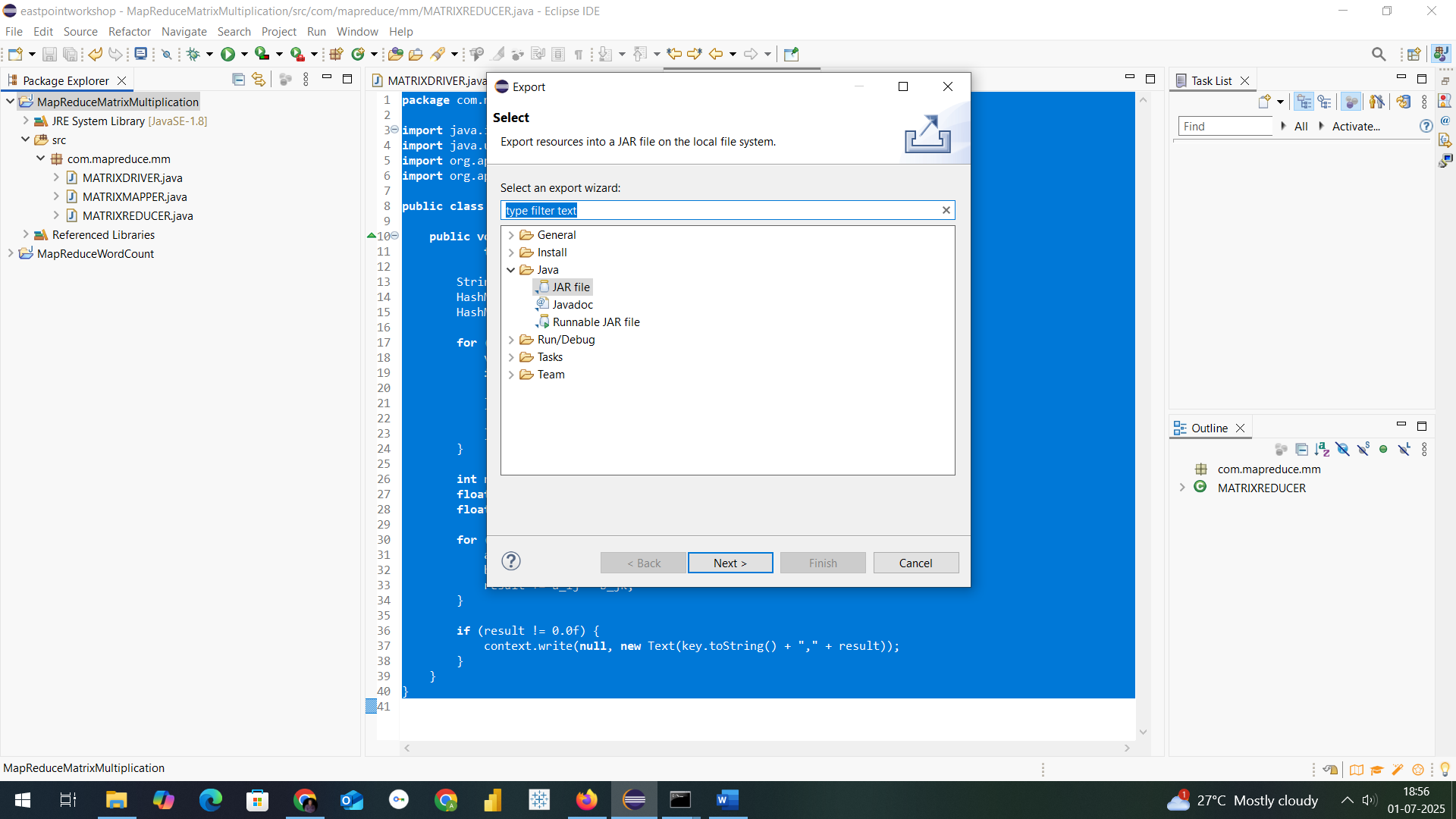
System.*exit*(job.waitForCompletion(true) ? 0 : 1);

}

}

Step8: Procedure to create jar file from eclipse project

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

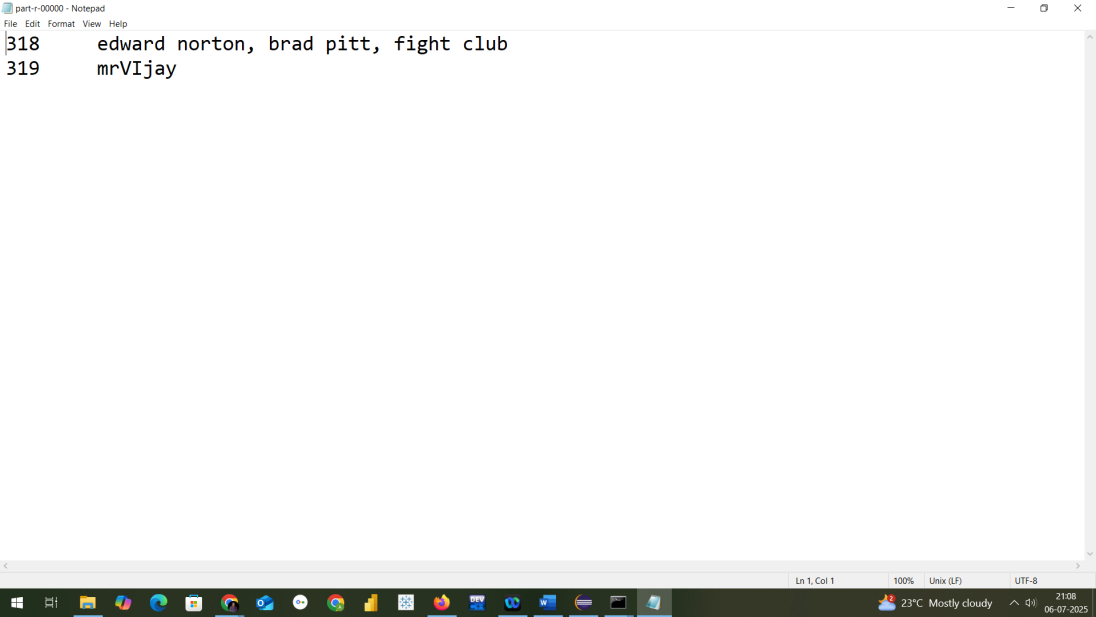
****

**THE OUTPUT WILL BE CREATED THIS TIME IN LOCALFOLDER WHERE MENTIONED**

**----🡪 D:\hadoopprograms\movieoutput**

**In part-r-00000 text file**

**NOW TO CHECK THE RESULT OF A PROGRAM**

****

**PROGRAM NO:5**

**Implement Functions: Count – Sort – Limit – Skip – Aggregate using MongoDB MongoDB Operations: Count, Sort, Limit, Skip, Aggregate**

# **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 NO:6**

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

**PIG INSTALLATION**

Extract to: C:\pig

Set Environment Variables:

PIG\_HOME = C:\pig

HADOOP\_HOME = (path to your Hadoop, e.g., C:\hadoop)

JAVA\_HOME = (path to JDK, e.g., C:\Program Files\Java\jdk1.8.0\_311)

Add to Path:

%PIG\_HOME%\bin

Run Pig

Open Command Prompt:

For Local Mode:

pig -x local

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

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);

Project Specific Columns

projected\_data = FOREACH students GENERATE name, marks;

DUMP projected\_data;

Filter Rows

high\_scorers = FILTER students BY marks > 80;

DUMP high\_scorers;

Group by Department

grouped\_by\_dept = GROUP students BY dept;

DUMP grouped\_by\_dept;

1. 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;

1. Sort by Marks Descending

sorted\_students = ORDER students BY marks DESC;

DUMP sorted\_students;

1. Join Students with Departments

joined\_data = JOIN students BY dept, departments BY code;

DUMP joined\_data;

1. To project joined output:

result = FOREACH joined\_data GENERATE

students::id,

students::name,

departments::dept\_name,

students::marks;

DUMP result;

# **PROGRAM NO7: 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 ≤ 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 8A : Word Count program using MapReduce WITH hadoop**

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

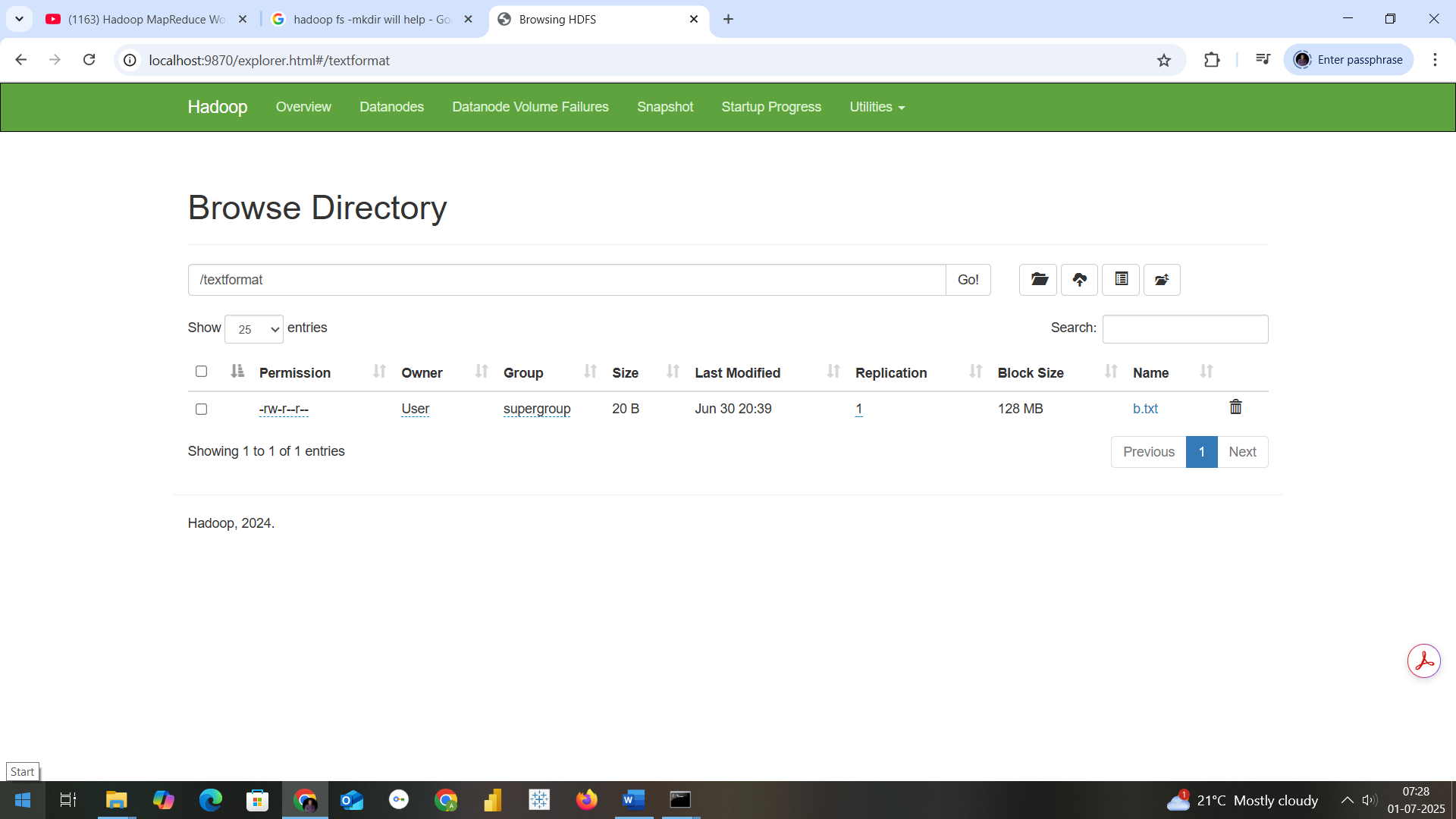
hadoop fs -mkdir /textformat

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 consol**e**

****

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

hadoop fs -ls /textformat

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

So open Eclipse

Create a java project with name MapReduceWordCount

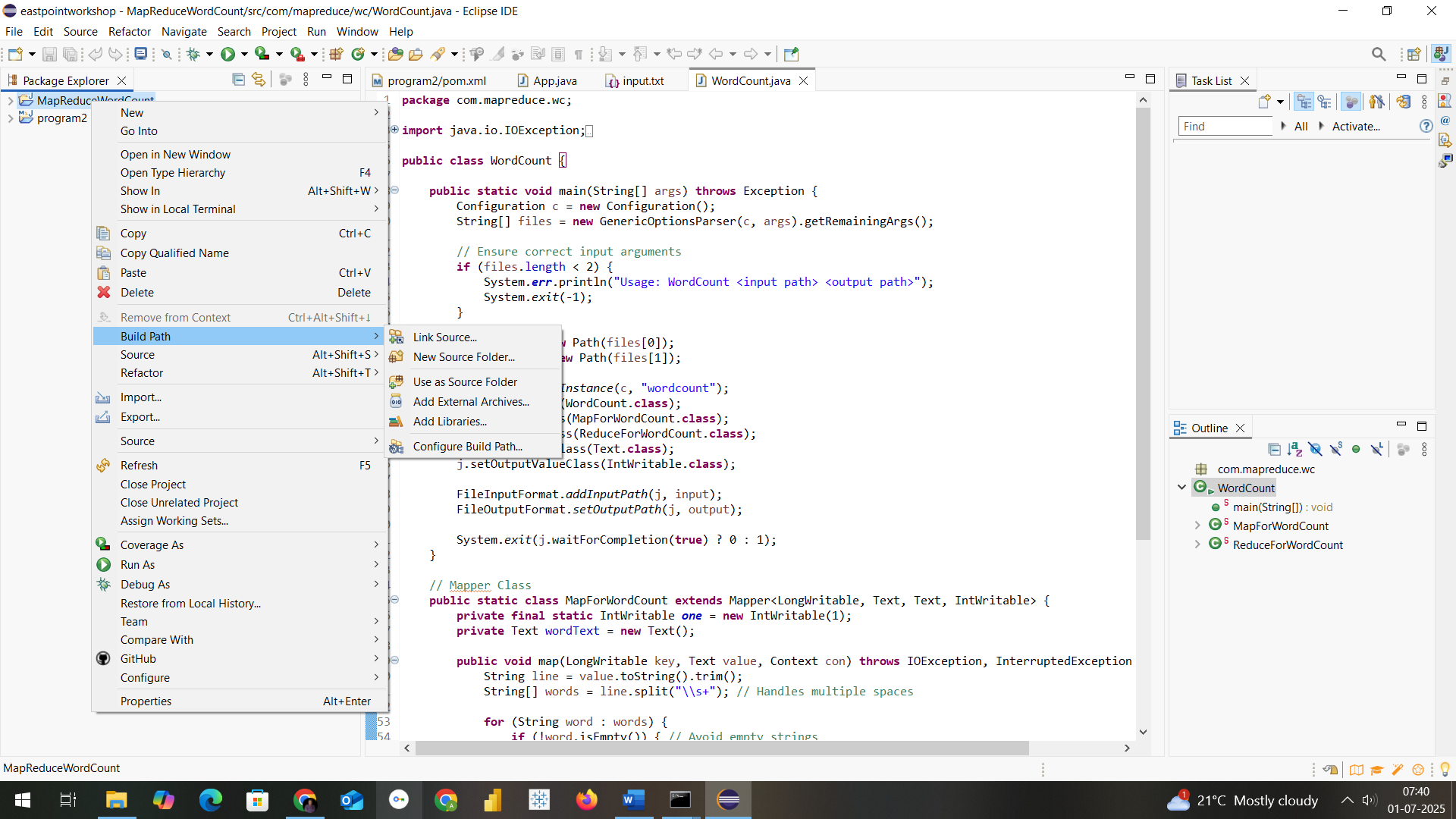
Select execution environment as JavaSE-1.8

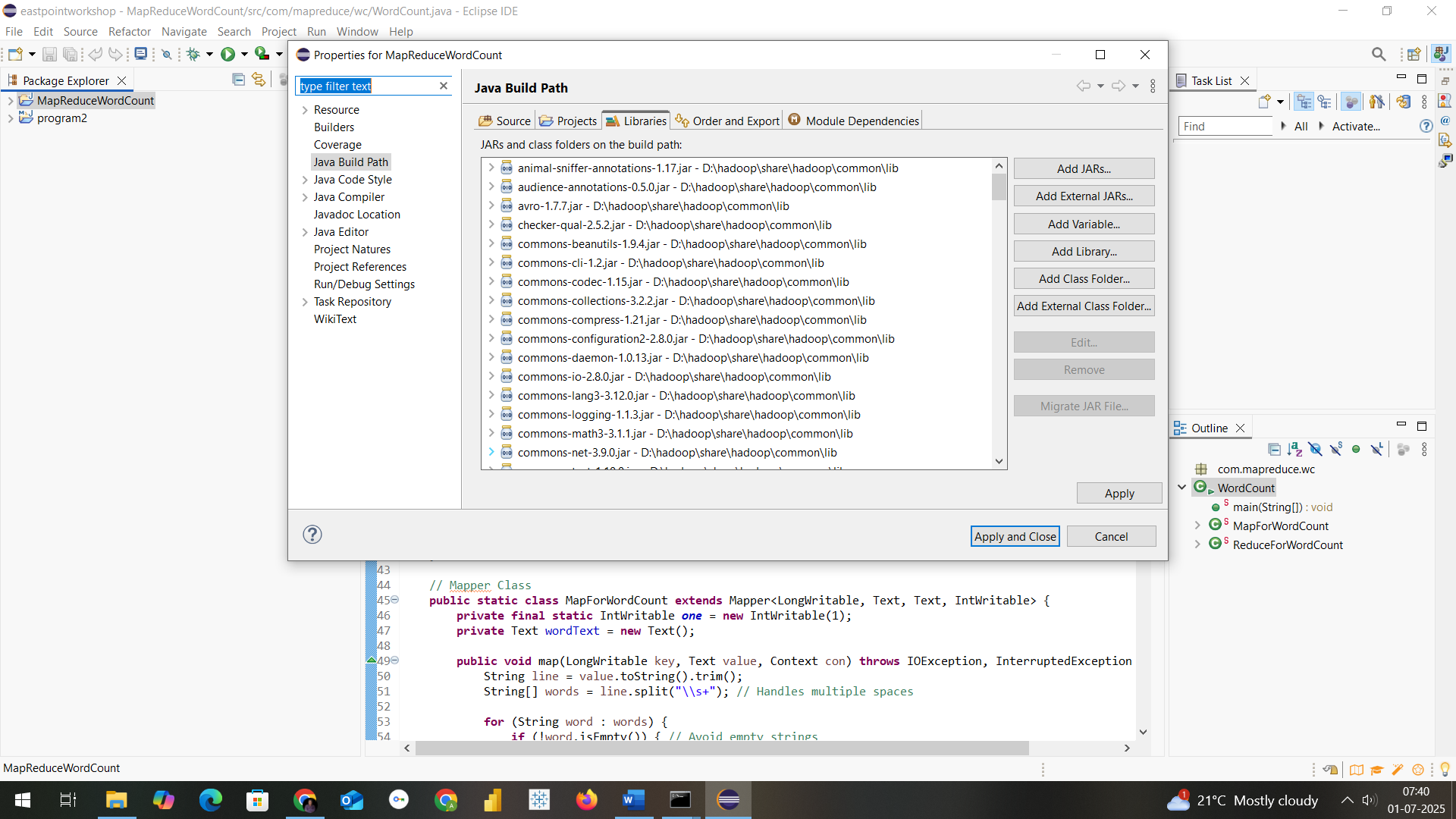
And click on next&finish

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

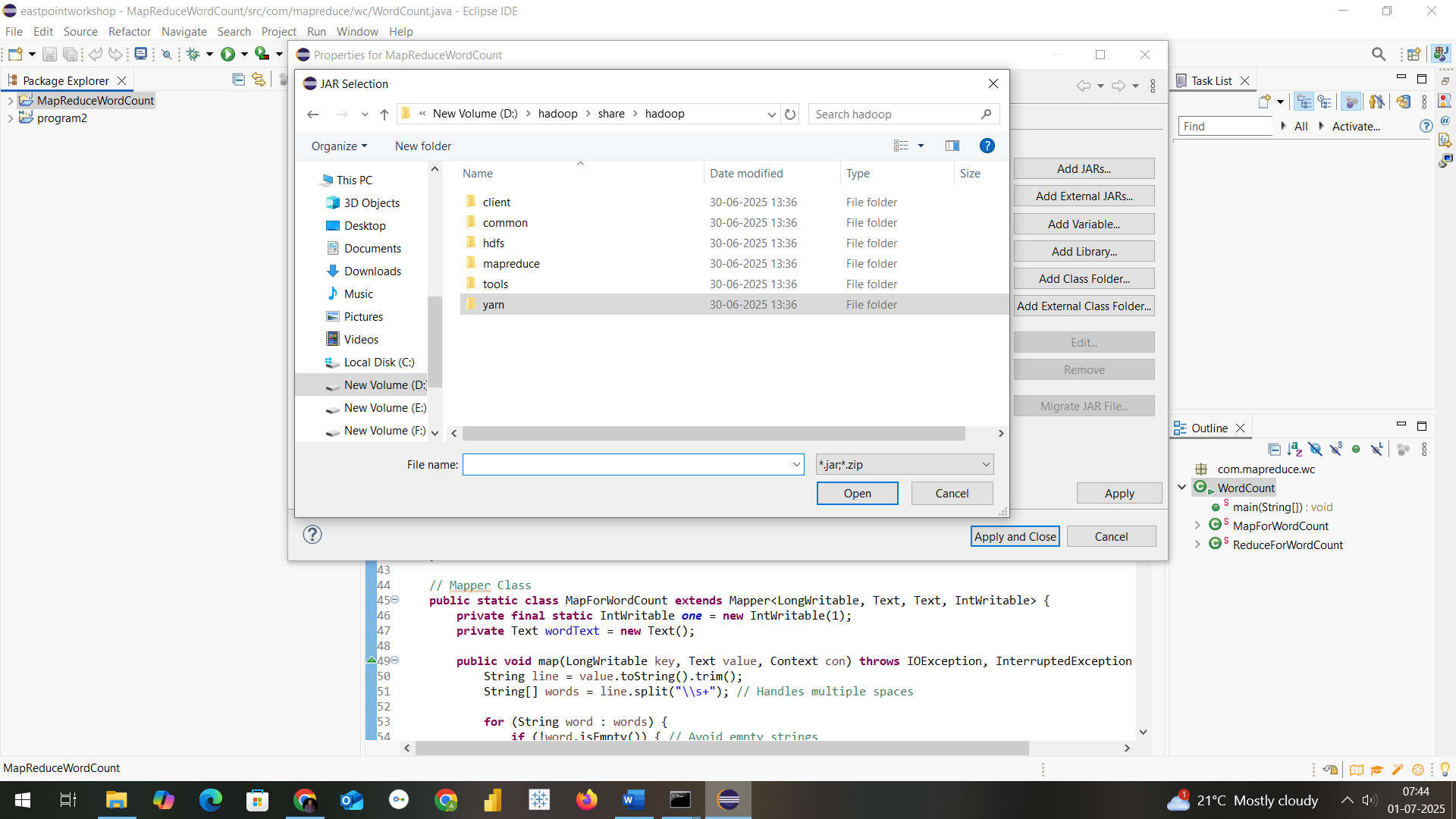
Right Click on Project-------🡪Right Click on BuildPath-------🡪 Configure Buildpath Then goto Libraries as in second screen below

****

****

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

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

****

Step10: Create aclass 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));

}

}

}

**PROGRAM 8B: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

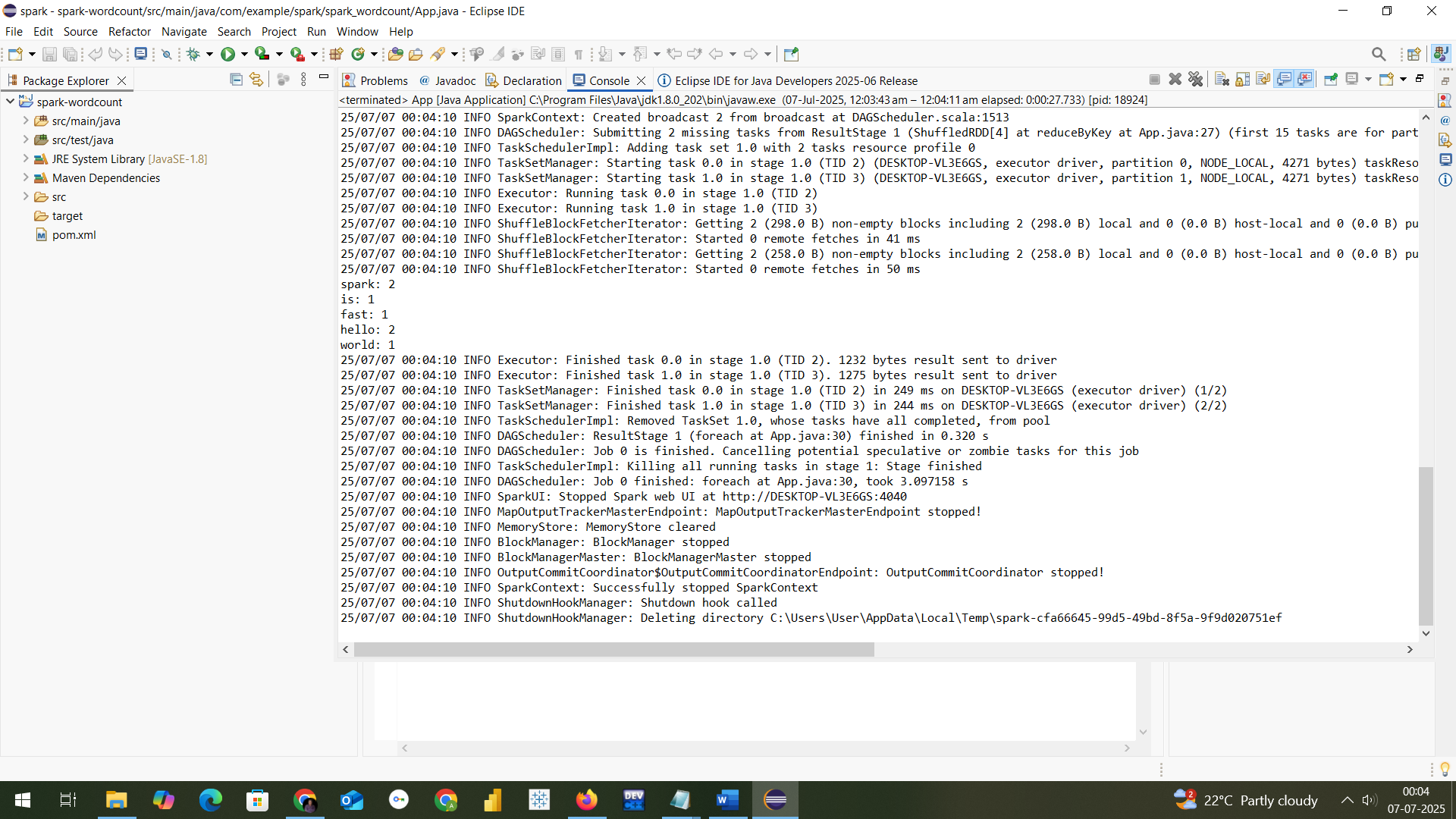
world: 1

spark: 2

is: 1

fast: 1

**RESULT**

****

**PROGRAM9-Using CDH and Hue to Analyze Data and Generate Reports**

# **Overview**

CDH (Cloudera Distribution for Hadoop) is a comprehensive Hadoop distribution. Hue (Hadoop User Experience) is a web interface that simplifies using Hadoop components such as Hive, Impala, and HDFS.

# 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.