State Institute of Engineering & Technology, Nilokheri
Department of Computer Science & Engineering (AI &ML)

Practical File Record

Of

Big Data Analytics for IOT Lab

(PE-CS- AI ML-421LA)

Submitted to:

Mrs. Aarti Chahal

Submitted by:

Name: ……

Session: 2026-27 (Odd Semester)

INDEX

S. No.	Name of Practical	Date of Practical	Assessment					Faculty Sign.
			Attendance (10)	File Record (10)	Lab Performance (10)	Viva (10)	Total Marks (40)	
1.	To Study Big Data Analytics and Hadoop Architecture.							
2.	Installation of Single Node Hadoop Cluster on Ubuntu.							
3.	Hadoop Programming: Word Count Map Reduce Program Using Eclipse							
4.	Implementing Matrix Multiplication Using One Map-Reduce Step.							
5	Implementing Relational Algorithm on Pig.							
6	Implementing database operations on Hive.							
7	Implementing Bloom Filter using Map-Reduce							
8	Implementing Frequent Item set algorithm using Map-Reduce.							
9	Implementing Clustering algorithm using Map-Reduce							
10	Implementing Page Rank algorithm using Map-Reduce							
							Total Marks	
							Aggregated Marks	

Practical No. 1

Aim: To Study Big Data Analytics and Hadoop Architecture.

Theory: -

1. Big Data Analytics: -

Foundational Concepts:

- Big Data Characteristics: Familiarize yourself with the characteristics of big data, including volume, velocity, variety, veracity, and value.
- Data Types: Understand structured, semi-structured, and unstructured data formats.
- Data Sources: Explore various sources of big data, such as social media, sensors, transactional systems, and IoT devices.

Data Storage and Management:

- Data Warehousing: Learn about traditional data warehousing concepts and technologies for storing and managing structured data.
- NoSQL Databases: Explore NoSQL databases like MongoDB, Cassandra, and HBase, which are designed to handle semi-structured and unstructured data at scale.
- Data Lakes: Understand the concept of data lakes for storing raw, unstructured data in its native format.

Data Processing and Analysis:

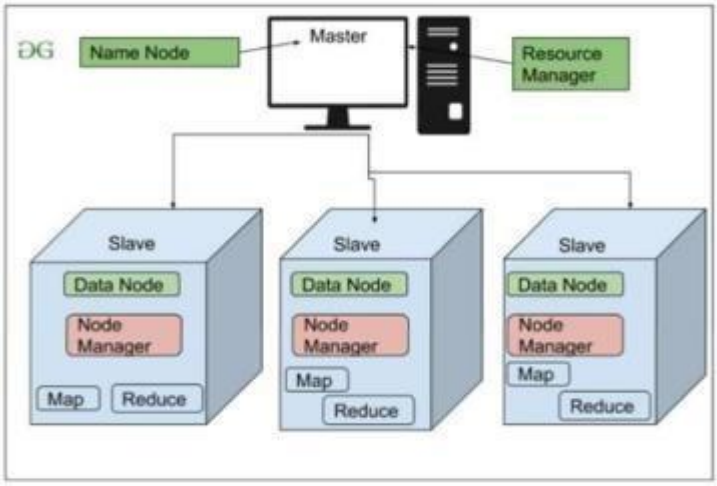
- Data Preprocessing: Study techniques for data cleaning, transformation, and normalization to prepare data for analysis.
- Statistical Analysis: Learn statistical methods and techniques for descriptive and inferential analysis of big data.
- Machine Learning: Explore machine learning algorithms and techniques for predictive analytics, classification, clustering, and anomaly detection.
- Text Mining and Natural Language Processing (NLP): Understand how to extract insights from text data using NLP techniques like sentiment analysis, topic modeling, and named entity recognition.

Tools and Technologies:

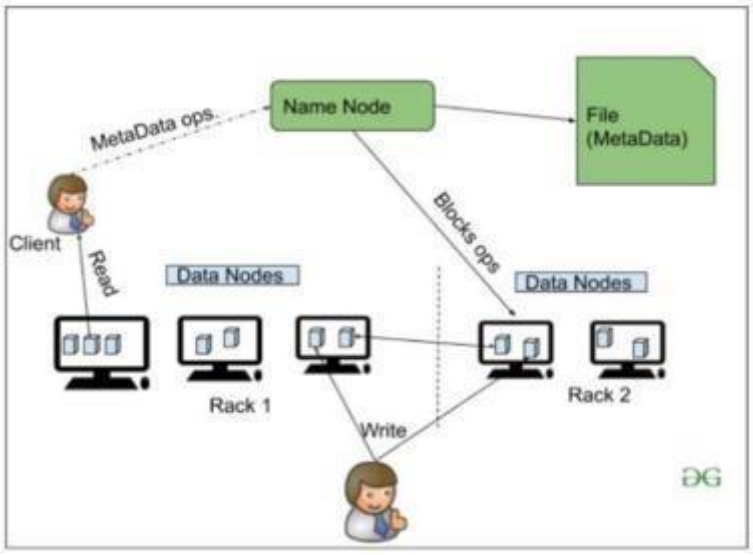
- Apache Hadoop: Dive into Hadoop, an open-source framework for distributed storage and processing of big data. ○ Apache Spark: Explore Spark, a fast and general-purpose cluster computing system compatible with Hadoop data.
- Data Analytics Platforms: Familiarize yourself with analytics platforms like Apache Flink, Databricks, and Google BigQuery.
- Data Visualization: Learn data visualization tools like Tableau, Power BI, and Matplotlib for creating visual representations of data insights. Real-World Applications:
- Industry Use Cases: Study real-world use cases of big data analytics across various industries, including finance, healthcare, retail, marketing, and cybersecurity.
- Case Studies: Analyze case studies to understand how organizations leverage big data analytics to drive business outcomes and gain a competitive edge.

- Ethical and Legal Considerations:
- Privacy: Understand the importance of data privacy and the ethical considerations involved in handling sensitive data.
- Regulatory Compliance: Familiarize yourself with regulations like GDPR, HIPAA, and CCPA governing the collection, storage, and processing of personal data.

1. Hadoop Architecture: -



2. Hadoop Distributed File System (HDFS): ○ NameNode: Centralized master node that manages the metadata and namespace of files stored in HDFS. ○ DataNode: Worker nodes responsible for storing and managing the actual data blocks on the local filesystem.



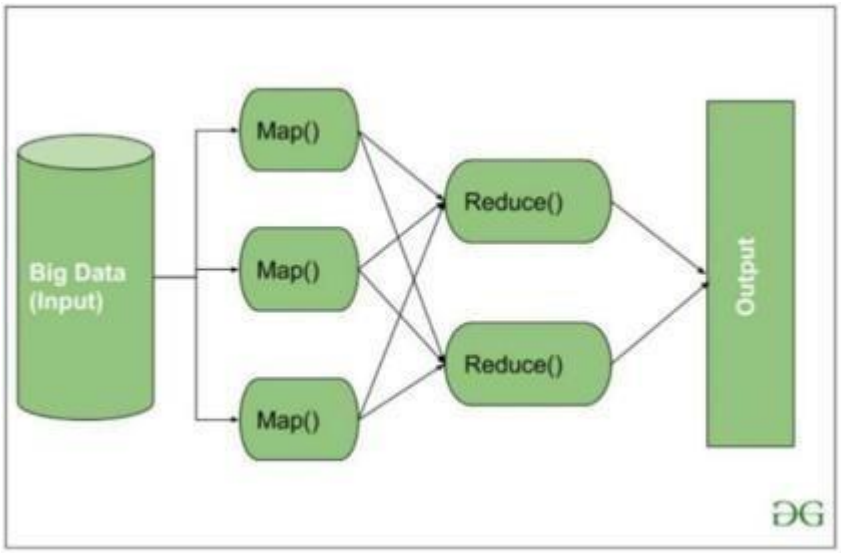
Yet Another Resource Negotiator (YARN):

- ○ ResourceManager: Manages cluster resources and schedules applications' resource requirements across the cluster. ○ NodeManager: Manages resources and executes tasks on individual cluster

nodes.

MapReduce: ○ MapReduce Engine: Executes MapReduce jobs across the cluster, handling parallel processing of data.

- Map Task: Processes input data and generates intermediate key-value pairs. Reduce Task: Aggregates intermediate results from map tasks based on keys.



3. Hadoop Common:
    - Utilities and Libraries: Provides common utilities, libraries, and dependencies required by Hadoop components.

Practical No. 2

Aim: Installation of Single Node Hadoop Cluster on Windows.

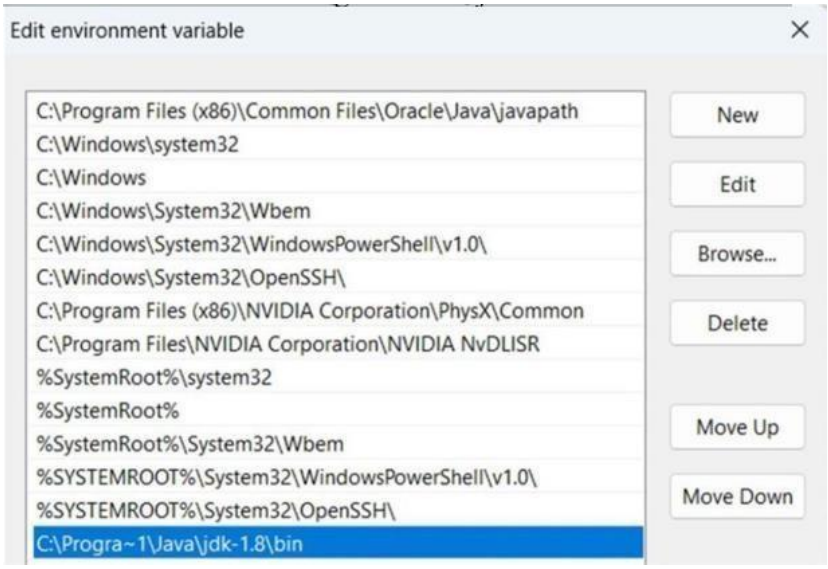
Procedure: -

Step 1: - , Download Java 8.
https://www.oracle.com/java/technologies/downloads/#java8-windows

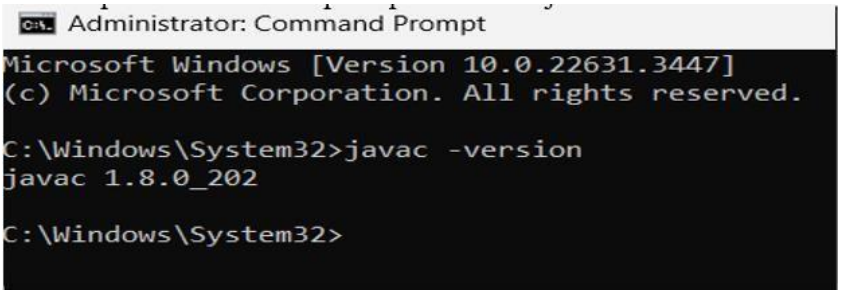
Step 2: - **click on the Windows icon -> Search for Edit system environment variable and Click on Environment variables.**

Step 3: - The Inside the User variables. Click on New and Create
Variable name: JAVA_HOME
Variable Value: C:\Progra~1\Java\jdk-1.8\bin

Step 4: - Go to System variables double click on Path -> Then inside the Path. Click on New and Add the Variable value like this C:\Progra~1\Java\jdk-1.8\bin Click on Ok.



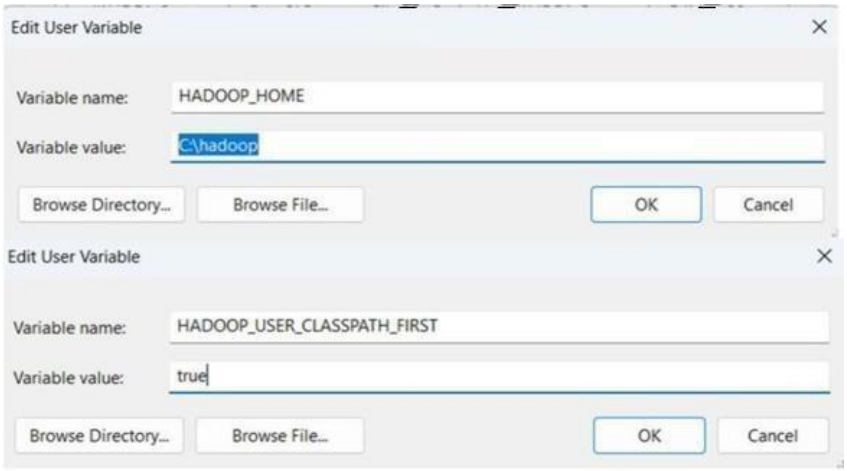
Step 5: - Now open the command prompt and write java -version



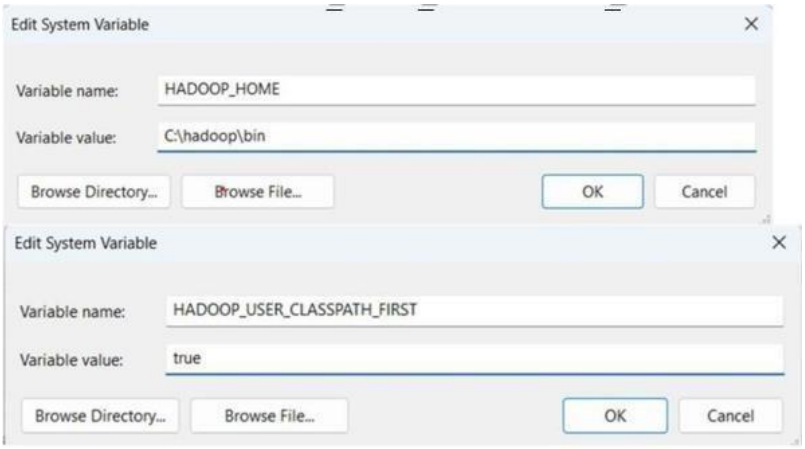
Step 6: - Next download the Hadoop files from the google drive, extract it, and place them in C drive.

Step 7: - After that click on the Windows icon -> Search for Edit system environment variable and Click on Environment variables.

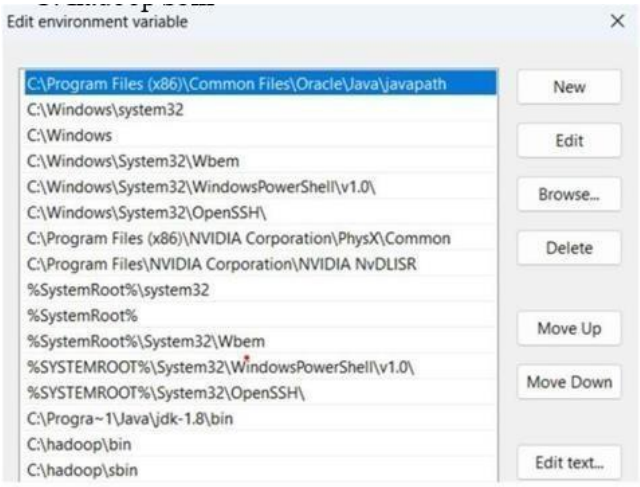
Step 8: - The Inside the User variables. Click on New and Create
Variable Name: HADOOP_HOME & Variable Value: C:\hadoop
Variable Name: HADOOP_USER_CLASSPATH_FIRST & Variable Value: true



Step 9: - The Inside the System variables. Click on New and Create
Variable Name: HADOOP_HOME & Variable Value: C:\hadoop\bin
Variable Name: HADOOP_USER_CLASSPATH_FIRST & Variable Value: true


Edit System Variable
Variable name:
HADOOP_HOME
Variable value:
C:\hadoop\bin
Browse Directory...
Browse File...
OK
Cancel
Edit System Variable
Variable name:
HADOOP_USER_CLASSPATH_FIRST
Variable value:
true
Browse Directory...
Browse File...
OK
Cancel

Step 10: - Now Go to Path and click on New and add.
C:\hadoop\bin
C:\hadoop\sbin



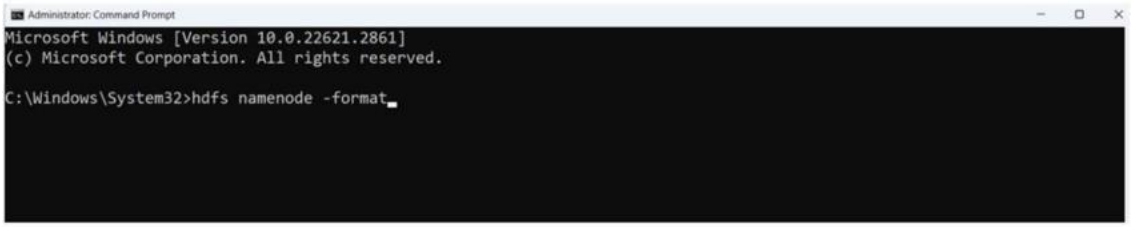
Step 11: - Now open the cmd and type Hadoop version

Step 12: - After that download the msvcr120.dll and install it. https://www.dll-files.com/download/b70474fe249402e251a94753b742788c/msvcr120.dll.html?c=dTZFRXVYZX doaTBkQjA0b1Z5UFRNUT09
Extract and place it in the system 32 folder present here (C:\Windows\System32)

Step 13: - Now download msvc-170 from the below link
https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170
Download the 64-bit and install it.

Step 14: - Now open the command prompt and run as administrator.

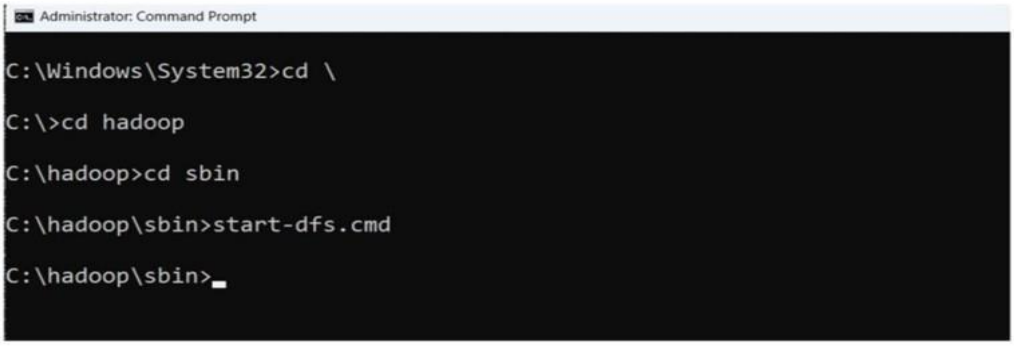
Step 15: - Now write-> hdfs namenode -format



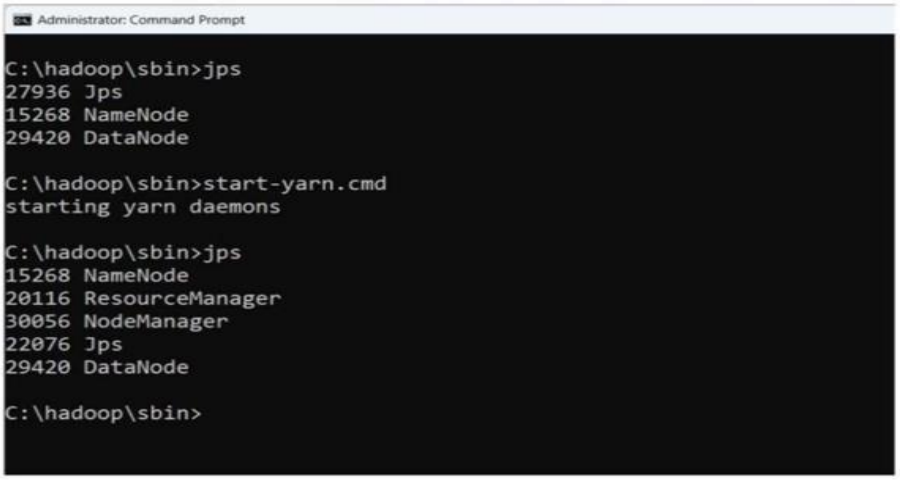
Step 16: - Now open the CMD as administrator again

Now follow these commands one by one

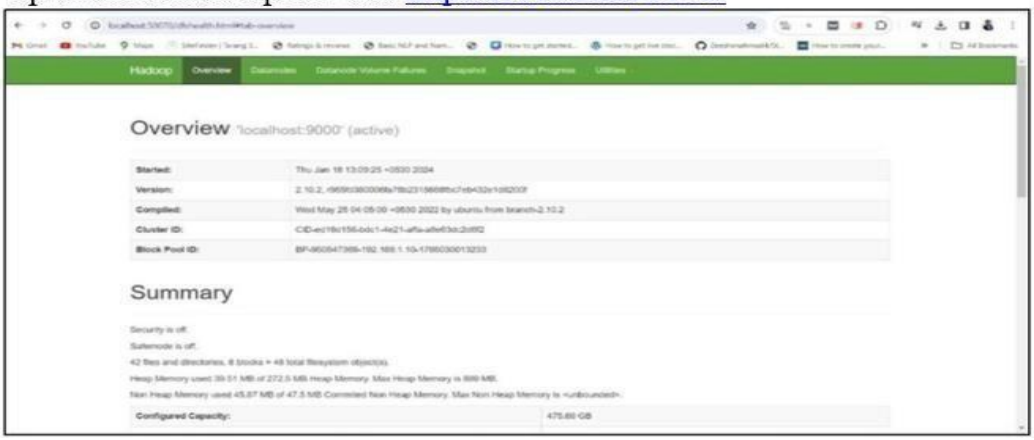
- cd \
- cd Hadoop
- cd sbin
- start-dfs.cmd



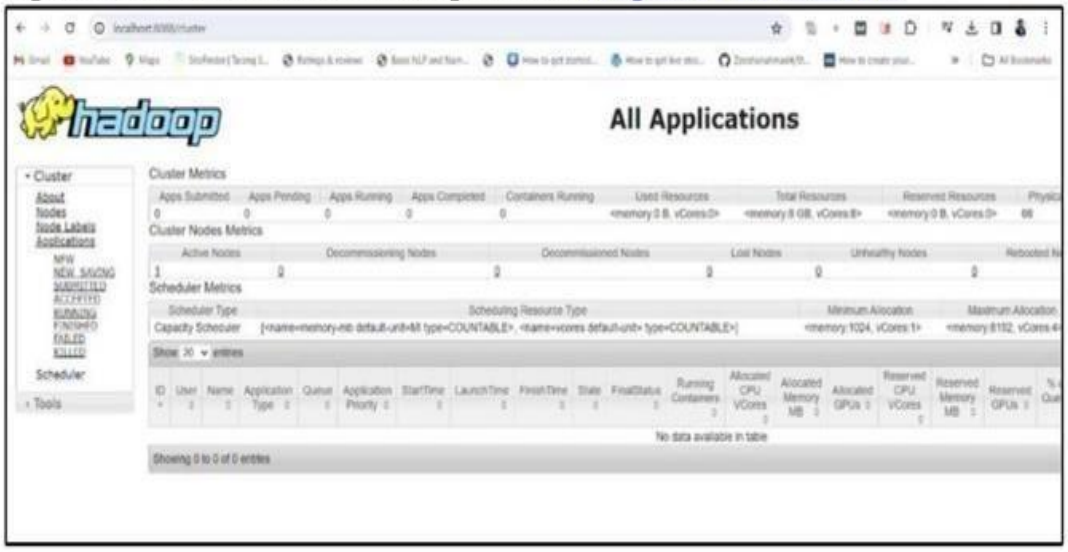
- jps
- start-yarn.cmd
- jps



- Open chrome and paste this http://localhost:50070/



- Open a new tab in chrome and paste this http://localhost:8088/



Practical No. 3 Aim:

Hadoop Programming: Word Count MapReduce Program..

Code: -

1. **pom.xml file-**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>Clustering</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

<dependencies>
  <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>3.3.3</version>
    </dependency>

    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-mapreduce-client-core</artifactId>
      <version>3.3.3</version>
  </dependency>
</dependencies>
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
```

</project>

2. **Word Count code (java file): -**

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text; import
org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

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

public class WCount { public static class TokenizerMapper extends
    Mapper<Object, Text, Text,
IntWritable> { private static final IntWritable one = new
    IntWritable(1); private Text word = new Text();

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

 public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {

      int sum = 0;
      for (IntWritable val : values) { sum
          += val.get();
       }
       result.set(sum);
       context.write(key, result);
    }
  } public static void main(String[] args) throws
 Exception {

      Configuration conf = new Configuration(); Job
      job = Job.getInstance(conf, "word count");
      job.setJarByClass(WCount.class);
      job.setMapperClass(TokenizerMapper.class);
      job.setCombinerClass(IntSumReducer.class);
      job.setReducerClass(IntSumReducer.class);
```

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


        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Run: -

```
C:\hadoop\sbin>hadoop jar "C:\hadoop\share\hadoop\mapreduce\hadoop-mapre
duce-examples-3.3.6.jar" wordcount /WC/ai_tools.txt /output_wordcount/pa
rt_1
2024-05-11 13:17:04,076 INFO client.DefaultNoHARMFailoverProxyProvider:
```

```
C:\hadoop\sbin>hdfs dfs -cat /output_WC1/*
...text 1
1.      1
2.      1
3.      1
4,      1
5.      1
5.d-id..        1
7.      1
adobe   1
copilot 1
firefly..       1
from    1
generation      1
image   1
pixverse.ai     1
selector        1
summary 1
text    2
text(select     1
ti      1
to      1
video   2
video)  1
youtube 1

C:\hadoop\sbin>_
```

Practical No. 4

Aim:- Implementing Matrix Multiplication Using One Map-Reduce Step.

Code: -

1. **pom.xml file-**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>Clustering</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

<dependencies>
  <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-common</artifactId>
      <version>3.3.3</version>
    </dependency>

    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-mapreduce-client-core</artifactId>
      <version>3.3.3</version>
  </dependency>
</dependencies>
    </project>
```

2. **Matrix Multiplication (Java Code): -**

```
package org.example;
import java.io.IOException;
import java.util.StringTokenizer;
```

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

public class MatrixMultiplication { public static class MatrixMapper extends

Mapper<Object, Text, Text, IntWritable> {

public void map(Object key, Text value, Context context) throws IOException,
        InterruptedException {
            // Split the input line into matrix, row, column,
            and value String[] input =
            value.toString().split(",");
            String matrix = input[0];
            int row =
            Integer.parseInt(input[1]);
            int col =
            Integer.parseInt(input[2]);
            int val =
            Integer.parseInt(input[3]);

            // Emit intermediate key-
            value pairs if
            (matrix.equals("A")) { for (int k = 0; k <
              context.getConfiguration().getInt("numColsB", 0); k++) {
              context.write(new Text(row + "," + k), new IntWritable(col * val));
              }
            } else { // Matrix B for (int k = 0; k <
              context.getConfiguration().getInt("numRowsA", 0); k++) {
              context.write(new Text(k + "," + col), new IntWritable(row * val));
              }
            }
        }
    }
    public static class MatrixReducer extends Reducer<Text, IntWritable,
```

```
Text, IntWritable> { public void reduce(Text key,
    Iterable<IntWritable> values, Context context) throws
    IOException, InterruptedException {
      int sum = 0;
for (IntWritable val : values) {
sum += val.get();
      } context.write(key, new
      IntWritable(sum));
    }
  } public static void main(String[] args) throws Exception
  { Configuration conf = new Configuration();
  conf.setInt("numRowsA", 2); // Number of rows in
  Matrix A conf.setInt("numColsB", 2); // Number of
  columns in Matrix B
    Job job = Job.getInstance(conf, "matrix multiplication");
    job.setJarByClass(MatrixMultiplication.class);
    job.setMapperClass(MatrixMapper.class);
    job.setReducerClass(MatrixReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Run:-

```
C:\hadoop\sbin>hdfs dfs -mkdir /MatMul

C:\hadoop\sbin>hdfs dfs -put "C:\Users\DELL\OneDrive\Desktop\aml\A.txt"
/MatMul

C:\hadoop\sbin>hdfs dfs -put "C:\Users\DELL\OneDrive\Desktop\aml\B.txt"
/MatMul
```

```
PS C:\Users\DELL\IdeaProjects\Matrix_mult> hadoop jar .\target\Matrix_mult-1.0-SNAP
SHOT.jar org.example.MatrixMultiplication /MatMul/ /output_mat
```

```
C:\hadoop\sbin>hdfs dfs -cat /output_mat/*
0,0     39
0,1     42
1,0     48
1,1     51
```

Practical No. 5

Aim:- Implementing Relational Algorithm on Pig.

Code:-

```
students = LOAD 'students.csv' USING PigStorage(',') AS (id:int, name:chararray, age:int);
marks = LOAD 'marks.csv' USING PigStorage(',') AS (id:int, subject:chararray, score:int);
selected_students = FILTER students BY age > 20; projected_names = FOREACH students
GENERATE name;
joined_data = JOIN students BY id, marks BY id;
batch1 = LOAD 'batch1.csv' USING PigStorage(',') AS (id:int, name:chararray, age:int);
batch2 = LOAD 'batch2.csv' USING PigStorage(',') AS (id:int, name:chararray, age:int);
union_data = UNION batch1, batch2;
intersect_temp = JOIN batch1 BY (id, name, age), batch2 BY (id, name, age);
intersection = FOREACH intersect_temp GENERATE batch1::id, batch1::name, batch1::age;
batch2_ids = FOREACH batch2 GENERATE id; difference = FILTER batch1 BY NOT (id IN
batch2_ids);
DUMP selected_students;
DUMP projected_names;
DUMP joined_data;
DUMP union_data;
DUMP intersection; DUMP
difference;
```

Output:-

```
grunt> students = LOAD 'students.csv' USING PigStorage(',') AS (id:int, name:chararray, age:int);
grunt> marks = LOAD 'marks.csv' USING PigStorage(',') AS (id:int, subject:chararray, score:int);
grunt> selected_students = FILTER students BY age > 20;
grunt> DUMP selected_students;
(1,Alice,21)
(3,Charlie,22)
grunt> projected_names = FOREACH students GENERATE name;
grunt> DUMP projected_names;
(Alice)
(Bob)
(Charlie)
(David)
grunt> joined_data = JOIN students BY id, marks BY id;
grunt> DUMP joined_data;
(1,Alice,21,1,Math,85)
(1,Alice,21,1,Science,90)
(2,Bob,19,2,English,78)
(3,Charlie,22,3,Math,88)
```

Practical No. 6

Aim:- Implementing Database Operations on Hive.

Code:-

```
CREATE DATABASE college;
USE college;
CREATE TABLE students(id INT, name STRING, age INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
LOAD DATA LOCAL INPATH 'students.csv' INTO TABLE students;
SELECT * FROM students;
SELECT name FROM students WHERE age > 20;
```

Output:-

```
hive> CREATE DATABASE college;
hive> USE college;
hive> CREATE TABLE students(id INT, name STRING, age INT)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
hive> LOAD DATA LOCAL INPATH 'students.csv' INTO TABLE students;
hive> SELECT * FROM students;
1Alice21
2Bob19
3Charlie22
4David20
hive> SELECT name FROM students WHERE age > 20;
Alice
Charlie
```

Practical No. 7

Aim:- Implementing Bloom Filter using Map-Reduce

Code: -

1. **pom.xml file-**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
         instance"
         xsi:schemaLocation="http://maven.apache.org/PO
M/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>Clustering</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
```

```
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      </properties>

  <dependencies>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>3.3.3</version>
      </dependency>

      <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
        <version>3.3.3</version>
    </dependency>
  </dependencies>
      </project>
```

2. Bloom Filter (Java Code): -

```
package org.example; import
java.io.IOException; import
java.util.BitSet; import
java.util.HashMap; import
java.util.Map;
import org.apache.hadoop.conf.Configuration; import
org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.BooleanWritable; 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; public
class BloomFilter {

    // Size of the Bloom Filter private
    static final int SIZE = 10000;
    private static final int NUM_HASH_FUNCTIONS
    = 3; private static final BitSet bitSet = new BitSet(SIZE);
```

```
public static class BloomFilterMapper extends
Mapper<LongWritable, Text, IntWritable, BooleanWritable> {

    private final IntWritable outputKey = new IntWritable();
    private final BooleanWritable outputValue = new BooleanWritable(true);

    @Override
    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException { String
      word = value.toString();
      for (int i = 0; i < NUM_HASH_FUNCTIONS; i++)
        { int hash = hashFunction(i, word);
        outputKey.set(hash); context.write(outputKey,
        outputValue);
      }
    }

 private int hashFunction(int index, String word) {

   return Math.abs((word.hashCode() + index * word.length()) % SIZE);
    }

   }

   public static class BloomFilterReducer extends
Reducer<IntWritable, BooleanWritable, Text, BooleanWritable> { private

    final Map<Integer, Boolean> bloomFilter = new

    HashMap<>(); @Override
    protected void reduce(IntWritable key, Iterable<BooleanWritable>
values, Context context) throws IOException, InterruptedException {
      int index = key.get(); if (!bloomFilter.containsKey(index)) {
      bloomFilter.put(index, true);
      }
    }
    @Override
    protected void cleanup(Context context) throws IOException,
    InterruptedException
{ for (Map.Entry<Integer, Boolean> entry :
      bloomFilter.entrySet()) { context.write(new
        Text(entry.getKey().toString()), new
```

```
BooleanWritable(entry.getValue()));
        }
      }
    } public static void main(String[] args)
    throws
      Exception { Configuration conf = new
      Configuration();
      Job job = Job.getInstance(conf, "Bloom Filter");
      job.setJarByClass(BloomFilter.class);
      job.setMapperClass(BloomFilterMapper.class);
      job.setReducerClass(BloomFilterReducer.class);
      job.setOutputKeyClass(IntWritable.class);
      job.setOutputValueClass(BooleanWritable.class);
      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Run: -

```
C:\hadoop\sbin>hdfs dfs -mkdir /Bloom

C:\hadoop\sbin>hdfs dfs -put "C:\Users\DELL\OneDrive\Desktop\aml\data2.txt" /Bloom
```

```
PS C:\Users\DELL\IdeaProjects\bloom_filter> hadoop jar .\target\bloom_filter-1.0-SN
APSHOT.jar org.example.BloomFilter /Bloom/ /output_Bloom
```

```
C:\hadoop\sbin>hdfs dfs -cat /output_Bloom/*
8802    true
6852    true
6855    true
8807    true
6858    true
8812    true
9839    true
3504    true
4465    true
2322    true
9842    true
3507    true
9845    true
3510    true
4470    true
2327    true
4475    true
2332    true
```

Practical No. 8

Aim:- Implementing Frequent Item set algorithm using Map-Reduce.

Code: -

1. **pom.xml file-**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
         instance"
         xsi:schemaLocation="http://maven.apache.org/PO
M/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>Clustering</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
```

```
            <maven.compiler.source>8</maven.compiler.source>
            <maven.compiler.target>8</maven.compiler.target>
            <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        </properties>

<dependencies>
  <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>3.3.3</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-core</artifactId>
            <version>3.3.3</version>
  </dependency>
</dependencies>
    </project>
```

1. Frequency Item Set (Java Code): -

```
package org.example;
import
java.io.IOException;
import  java.util.ArrayList;
import    java.util.HashSet;
import          java.util.List;
import java.util.Set;
import org.apache.hadoop.conf.Configuration;
import    org.apache.hadoop.fs.Path;    import
org.apache.hadoop.io.IntWritable;       import
org.apache.hadoop.io.Text;              import
org.apache.hadoop.mapreduce.Job;        import
org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

   public class Apriori { public static class Mapper1 extends
     Mapper<Object, Text, Text, IntWritable> { private final static
     IntWritable one = new IntWritable(1);
        private final Text itemset = new Text();
        public void map(Object key, Text value, Context context) throws
        IOException,
```

```
InterruptedException { String[]
        items =
        value.toString().split("\\s+"); for
        (String item : items) {
           itemset.set(item);
           context.write(items
           et, one);
        }
     }
  } public static class Reducer1 extends Reducer<Text, IntWritable,
  Text, IntWritable> { private final IntWritable result = new
  IntWritable(); public void reduce(Text key, Iterable<IntWritable>
  values, Context context) throws IOException, InterruptedException {
        int sum = 0; for
        (IntWritable val :
           values) {
      sum += val.get();
        }
        result.set(sum);
        context.write(key,
        result);
     }
  } public static class Mapper2 extends Mapper<Object, Text,
  Text,
     IntWritable> { private final Text itemset = new Text();
     public void map(Object key, Text value, Context context) throws
     IOException,
InterruptedException {
        String[] items = value.toString().split("\\s+");
        Set<String> set = new HashSet<>();
        for (String item : items) {
           set.add(item);
        }
        List<String> subset = new
        ArrayList<>(set); int size
        = subset.size();
        for (int i = 0; i < size; i++) { for (int j = i + 1;
           j < size; j++) { itemset.set(subset.get(i) +
           "," + subset.get(j)); context.write(itemset,
           new IntWritable(1));
           }
        }
     }
```

```
} public static class Reducer2 extends Reducer<Text, IntWritable,
Text, IntWritable> { private final IntWritable result = new
IntWritable(); public void reduce(Text key, Iterable<IntWritable>
values, Context context) throws IOException, InterruptedException {
    int sum = 0; for
    (IntWritable val :
      values) {
   sum += val.get();
    }
    result.set(sum);
    context.write(key,
    result);
  }
} public static void main(String[] args)
throws
  Exception { Configuration conf = new
  Configuration();
Job job1 = Job.getInstance(conf, "Apriori1"); job1.setJarByClass(Apriori.class);
  job1.setMapperClass(Mapper1.class);
  job1.setCombinerClass(Reducer1.class);
  job1.setReducerClass(Reducer1.class);
  job1.setOutputKeyClass(Text.class);
  job1.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job1, new Path(args[0]));
  FileOutputFormat.setOutputPath(job1, new Path(args[1]));
  job1.waitForCompletion(true);

    Job job2 = Job.getInstance(conf, "Apriori2");
    job2.setJarByClass(Apriori.class);
    job2.setMapperClass(Mapper2.class);
    job2.setCombinerClass(Reducer2.class);
    job2.setReducerClass(Reducer2.class);
    job2.setOutputKeyClass(Text.class);
    job2.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job2, new
    Path(args[1]));
    FileOutputFormat.setOutputPath(job2, new
    Path(args[2]));
    System.exit(job2.waitForCompletion(true) ? 0 :
    1);
  }
}
```

Run: -

```
C:\hadoop\sbin>hdfs dfs -mkdir /freq

C:\hadoop\sbin>hdfs dfs -put "C:\Users\DELL\OneDrive\Desktop\aml\data.txt"
 /freq
```

```
PS C:\Users\SHIVAM\IdeaProjects\FreqItemSet> hadoop jar .\target\FreqItemSet-1.0-SNAPSHOT.jar org.example.Apriori /freq/ /output_freq
```

```
C:\hadoop\sbin>hdfs dfs -cat /output_freq/*
1       3
2       3
3       4
4       1
5       4
```

Practical No. 9

Aim:- Implementing Clustering algorithm using Map-Reduce

Code: -

1. **pom.xml file-**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>Clustering</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

<dependencies>
  <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
```

```
        <version>3.3.3</version>
      </dependency>
      <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
        <version>3.3.3</version>
    </dependency>
</dependencies>
    </project>
```

2. Clustering (Java File): -

```
package org.example;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.DoubleWritable;
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 KMeansMapReduce {

  // Mapper class
  public static class KMeansMapper extends Mapper<Object, Text,
Text, DoubleWritable> { private final List<double[]> centroids
    = new ArrayList<>();
    // Read initial centroids from the file
    protected void setup(Context context) throws
      IOException, InterruptedException { Configuration
      conf = context.getConfiguration();
      FileSystem fs = FileSystem.get(conf);
      BufferedReader reader = new
BufferedReader(new InputStreamReader(fs.open(new
Path(conf.get("centroidPath"))))));
      String line;
      while ((line = reader.readLine())
        != null) { String[] parts =
```

```
                    line.split(","); double[] centroid
                    = new
                    double[parts.length]; for (int i = 0; i <
                    parts.length; i++) { centroid[i] =
                    Double.parseDouble(parts[i]);
                    }
                    centroids.add(centroid);
                } reader.close();
            }
// Map each point to its nearest centroid public void map(Object key, Text value, Context context)
                throws IOException,
        InterruptedException { String[]
                parts =
                value.toString().split(",");
                double[] point = new
                double[parts.length]; for (int i = 0;
                i < parts.length; i++)
                { point[i] =
                Double.parseDouble(parts[i]); }
```

```
        double minDistance = Double.MAX_VALUE;
        double[] nearestCentroid = null; for (double[]
        centroid : centroids) { double distance =
        calculateEuclideanDistance(point, centroid); if
        (distance < minDistance) {
            minDistance =
            distance;
            nearestCentroid
            = centroid;
        }
        } context.write(new Text(arrayToString(nearestCentroid)),
        new
DoubleWritable(minDistance));
    }
    // Calculate Euclidean distance between two points private
    double calculateEuclideanDistance(double[] p1, double[]
      p2) { double sum = 0.0;
      for (int i = 0; i < p1.length; i++)
        { sum +=
        Math.pow(p1[i] - p2[i],
        2);
      }                    return
      Math.sqrt(sum);
    }
    // Convert array to string private
    String arrayToString(double[]
      array) { StringBuilder sb = new StringBuilder();
      for (int i = 0; i < array.length; i++)
        { sb.append(array[i]); if (i <
        array.length - 1) {
        sb.append(",");
        }
      }                return
      sb.toString();
    }
  }
  // Reducer class
  public static class KMeansReducer extends Reducer<Text, DoubleWritable,
Text, DoubleWritable> {
    // Reduce step just outputs the key-value pairs
```

```
        public void reduce(Text key, Iterable<DoubleWritable>values, Context
context) throws IOException, InterruptedException {
            for (DoubleWritable value : values) {
               context.write(key, value);
            }
    // Driver method
    public static void main(String[] args) throws
       Exception { Configuration conf = new
       Configuration();
       String[] otherArgs = new GenericOptionsParser(conf,
       args).getRemainingArgs(); if (otherArgs.length != 3) {
          System.err.println("Usage:KMeansMapReduce<input_path> <centroid_path>
<output_path>");
          System.exit(2);
       }
       conf.set("centroidPath",otherArgs[1]);
       Job job = Job.getInstance(conf, "KMeans MapReduce");
       job.setJarByClass(KMeansMapReduce.class);
       job.setMapperClass(KMeansMapper.class);
       job.setReducerClass(KMeansReducer.class);
       job.setOutputKeyClass(Text.class);
       job.setOutputValueClass(DoubleWritable.class);
       FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
       FileOutputFormat.setOutputPath(job,new Path(otherArgs[2]));
       System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Result:-

```
C:\hadoop\sbin>hadoop jar target/clust-1.0-SNAPSHOT.jar org.example.KMea
nsMapReduce /cluster/ /cluster/centroids.txt /output_cluster
```

```
PS C:\Users\DELL\IdeaProjects\clust> hadoop jar target/clust-1.0-SNAPSHOT.jar org.example.KMeansMapReduce /cluster/ /cluster/centroids.txt /output_cluster
2024-05-11 12:32:35,427 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
```

```
C:\hadoop\sbin>hdfs dfs -cat /output_cluster/*
1.0,1.0,1.0     0.0
1.0,1.0,1.0     2.23606797749979
4.0,4.0,4.0     8.774964387392123
4.0,4.0,4.0     4.55521678957215
4.0,4.0,4.0     1.6583123951777
4.0,4.0,4.0     0.0
```

Practical No. 10

Aim:- Implementing Page Rank algorithm using Map-Reduce

Code: -

1. **pom.xml file-**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
        instance"
        xsi:schemaLocation="http://maven.apache.org/PO
M/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>Clustering</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

<dependencies>
  <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>3.3.3</version>
      </dependency>

      <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
        <version>3.3.3</version>
  </dependency>
</dependencies>
    </project>
```

2. Page Rank (Java File): -

```
package org.example;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; public class
PageRank { public static class PageRankMapper extends Mapper<Object, Text,
Text,

Text> { public void map(Object key, Text value, Context context) throws

IOException,
InterruptedException {
String[] parts =
value.toString().split("\\s+"); String
page = parts[0]; double
pageRank =
Double.parseDouble(parts[1]); int
linkCount = parts.length - 2; double
contribution = pageRank /
linkCount; context.write(new
Text(page), new Text("!")); for (int i
= 2; i < parts.length; i++) { String
linkedPage = parts[i];
context.write(new Text(linkedPage), new Text(String.valueOf(contribution)));
}
}
}

public static class PageRankReducer extends Reducer<Text, Text, Text, Text>

{ public void reduce(Text key, Iterable<Text> values, Context

context) throws
IOException,
InterruptedException
{ double newPageRank =
0.0; String links = ""; for
(Text value : values) {
```

```
            if
              (value.toString().equals("
              !")) { links = "!" + links;
            }          else          {          newPageRank          +=
              Double.parseDouble(value.toString());
            }
          } context.write(key, new Text(String.valueOf(newPageRank) +
          links));
        }
      }

      public static void main(String[] args) throws
        Exception { Configuration conf = new
        Configuration();
        Job job = Job.getInstance(conf, "PageRank");
        job.setJarByClass(PageRank.class);
        job.setMapperClass(PageRankMapper.class);
        job.setReducerClass(PageRankReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new
        Path(args[0]));
        FileOutputFormat.setOutputPath(job, new
        Path(args[1])); job.waitForCompletion(true);
      }
    }
```

Result:-

```
C:\hadoop\sbin>hadoop jar target/Pagerank-1.0-SNAPSHOT.jar org.example.P
ageRank /Pagerank/ /out_page
```

```
PS C:\Users\DELL\IdeaProjects\Pagerank> hadoop jar target/Pagerank-1.0-SNAPSHOT.jar org.example.PageRank /Pagerank/ /output_pagerank
2024-05-11 12:48:26,171 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
```

```
C:\hadoop\sbin>hdfs dfs -cat /out_page/*
PageA   1.0!
PageB   0.5!
PageC   1.5!
```