ACS COLLEGE OF ENGINEERING
DEPARTMENT OF CSE-Data Science

BIG DATA ANALYTICS
BAD601
MANUAL

Semester –VI

Prepared By:
Mrs. AARTHI K
ASSISTANT PROFESSOR
Department of CSE- Data Science
ACSCE, Bengaluru

LIST OF EXPERIMENTS

PRACTICAL COMPONENT OF IPCC

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

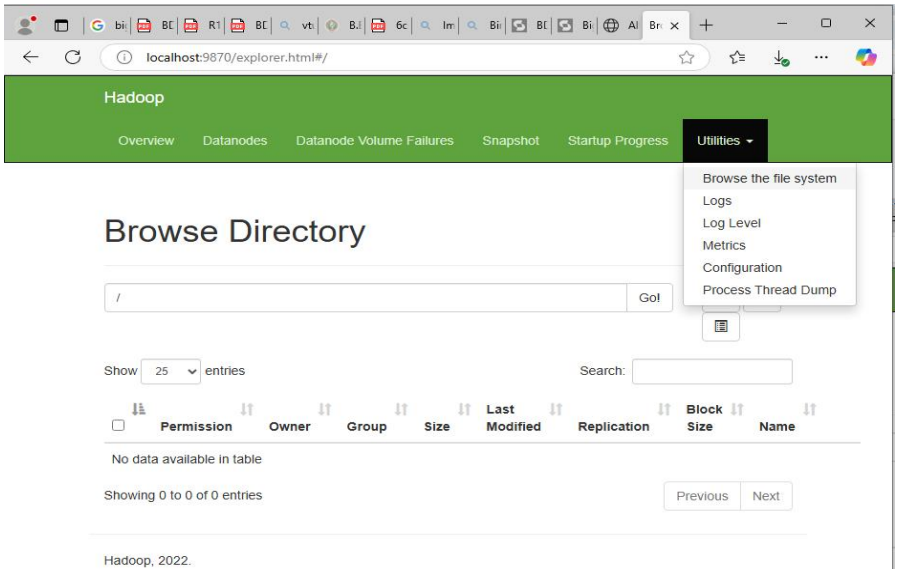
1.Install Hadoop and Implement the following file management tasks in Hadoop: Adding files and directories Retrieving files Deleting files and directories. Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

Starting the Hadoop Server:
(i)To start the Hadoop Servers, go to the Command prompt and then run this command. Syntax: Start-all.cmd (or) Start-dfs.cmd & Start-yarn.cmd

After that all the demons will start running.
- Namenode
- Datanode
- Resourcemanager
- Nodemanager

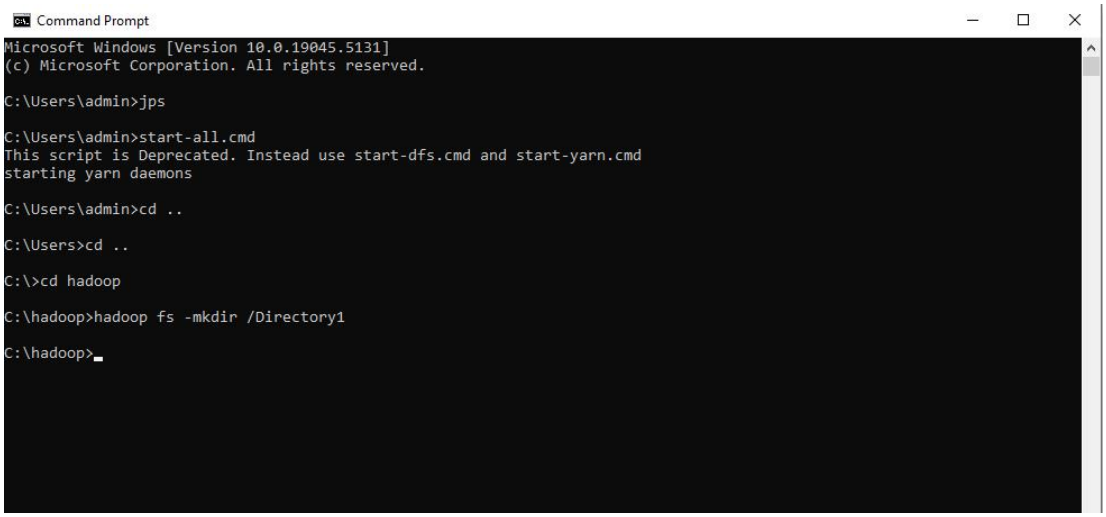
After Successfully running all these demons we can access these in localhost ports. This completes the installation part of Hadoop.
http://localhost:8088

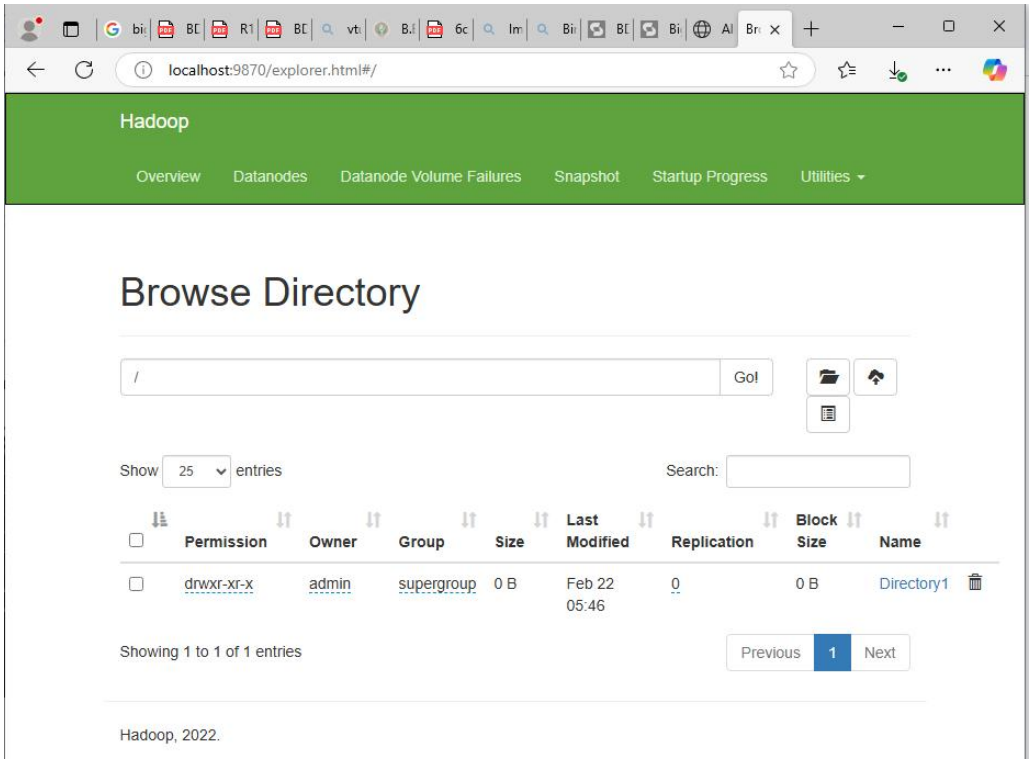
(ii)Use web-based tools to monitor your Hadoop setup.
Hadoop accessing through Web UI
1.http://localhost:8042/node
2.http://localhost:8088/cluster
3.http://localhost:9870/dfshealth.html#tab-overview
4.http://localhost:9864/datanode.html



Create a directory in HDFS at given path(s)

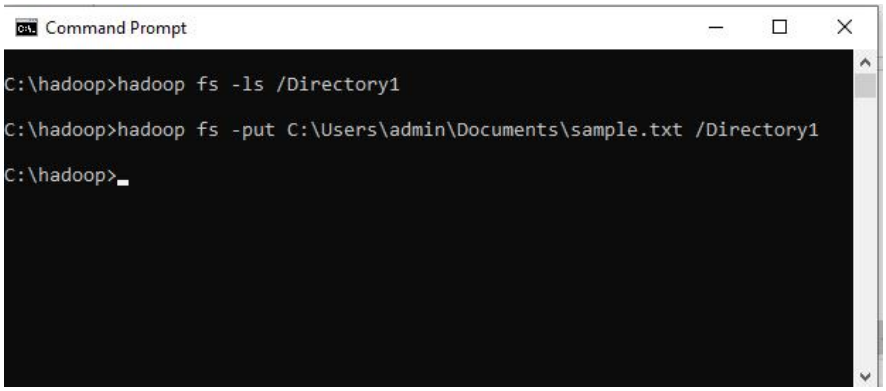
Syntax: C:\hadoop>hadoop fs -mkdir /Directory1

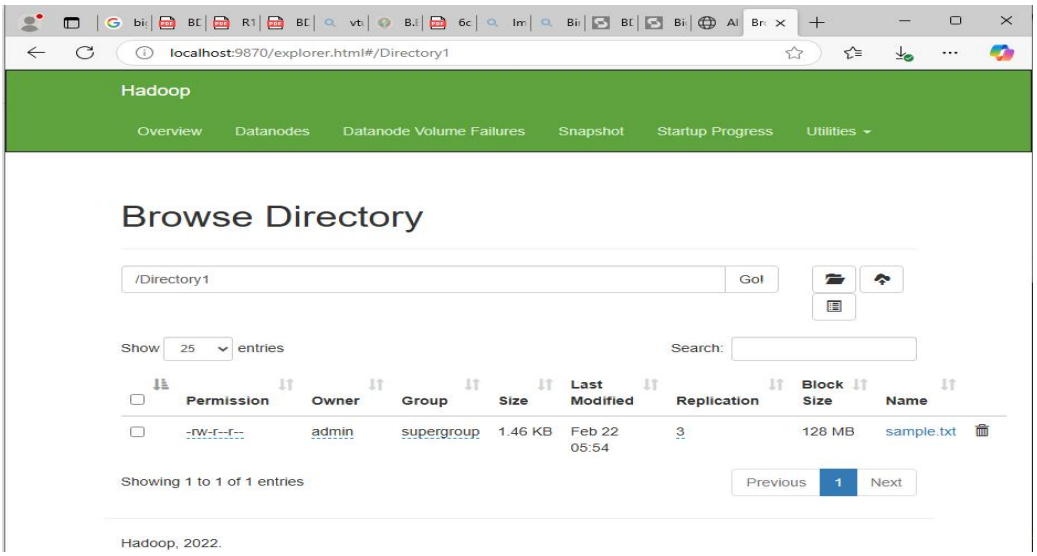




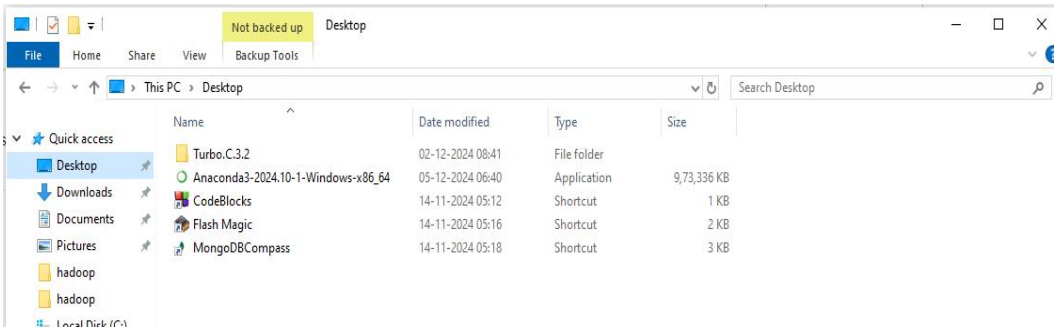
Upload and download a file in HDFS. Copy single src file, or multiple src files from local file system to the Hadoop data file system

Syntax:C:\hadoop>hadoop fs -put C:\Users\admin\Documents\sample.txt /Directory1



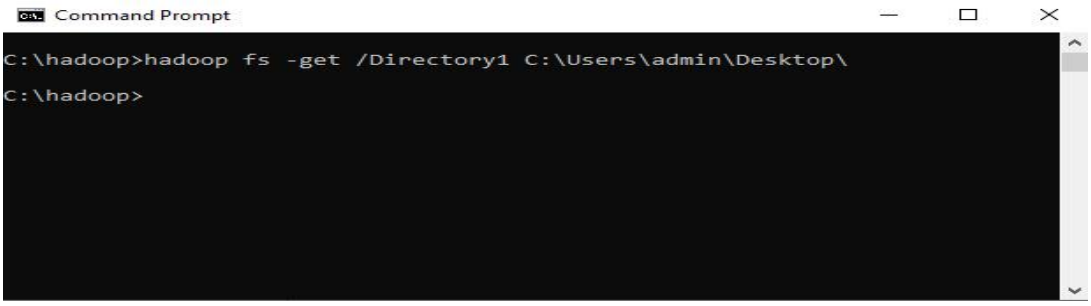


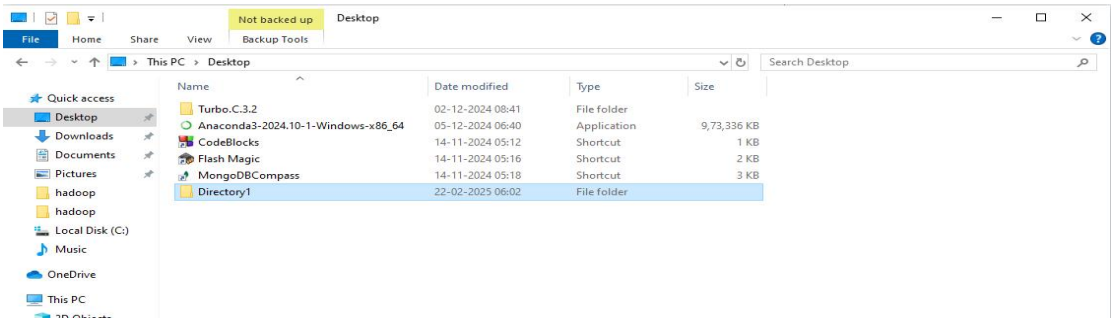
To download file from HDFS to desktop



Copies/Downloads files to the local file system

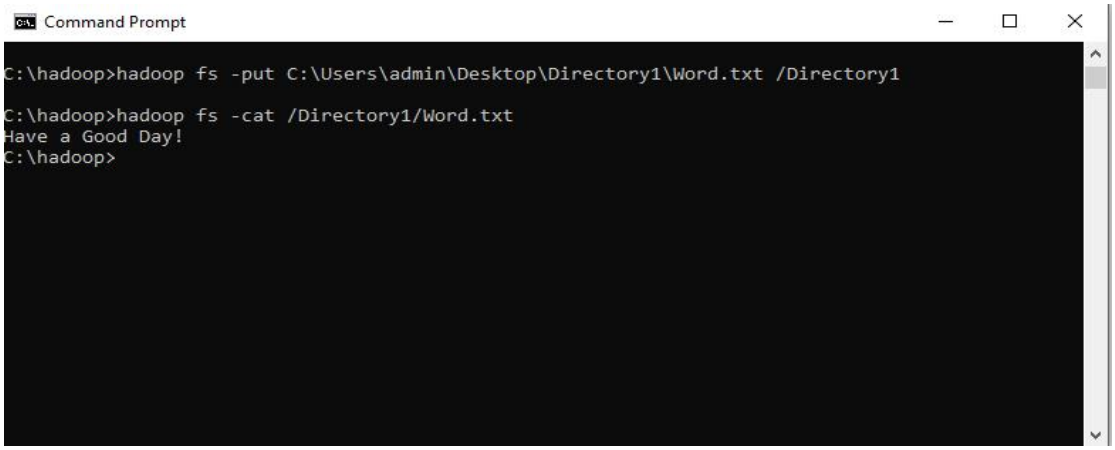
Syntax: C:\hadoop>hadoop fs -get /Directory1 C:\Users\admin\Desktop



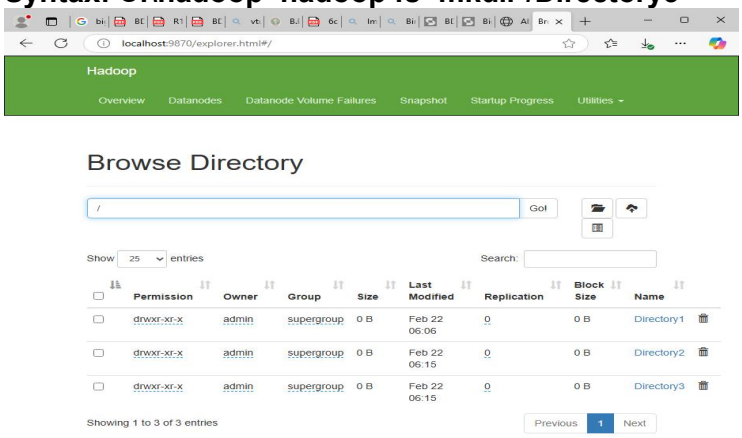


See contents of a file

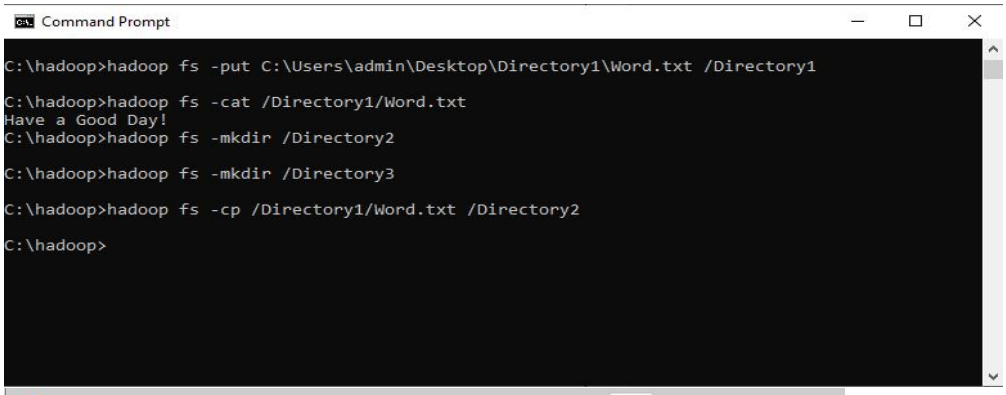
Create a File (Word.txt)
Upload the (Word.txt) file to HDFS
Syntax: C:\hadoop>hadoop fs -put C:\Users\admin\Desktop\Directory1\Word.txt /Directory1
To see the content of file
Syntax: C:\hadoop>hadoop fs -cat /Directory1/Word.txt

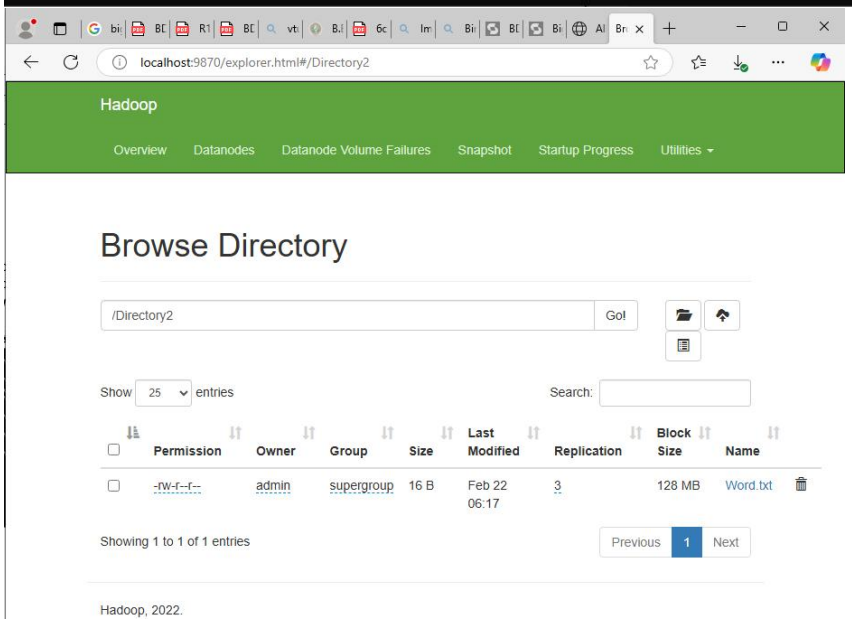


Copy a file from source to destination
This command allows multiple sources as well in which case the destination must be a directory.
Syntax: C:\hadoop>hadoop fs -mkdir /Directory2
Syntax: C:\hadoop>hadoop fs -mkdir /Directory3

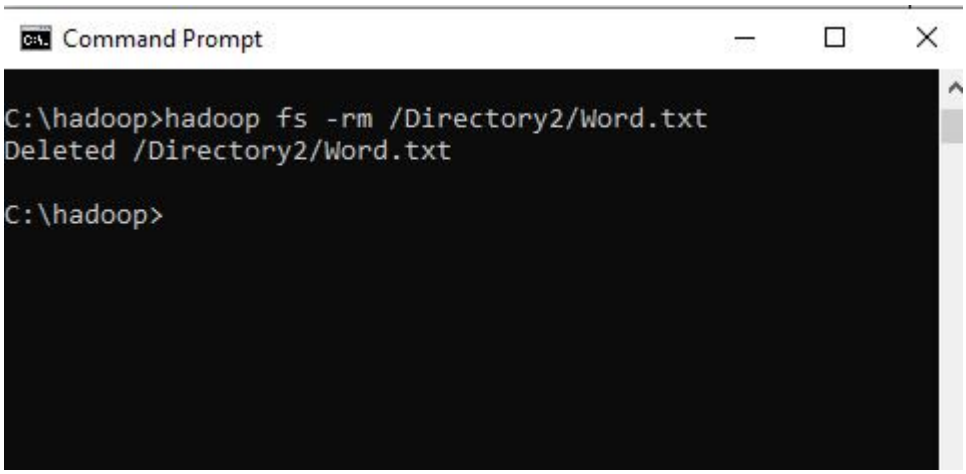


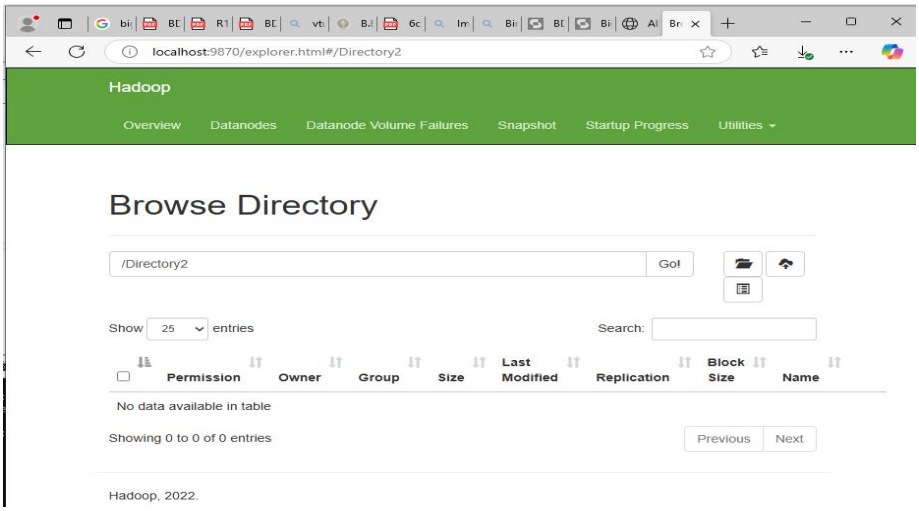
Copy a file from/To Local file system to HDFS:
Copy To Local Similar to put command, except that the source is restricted to a local file reference.
Syntax: C:\hadoop>hadoop fs -cp /Directory1/Word.txt /Directory2





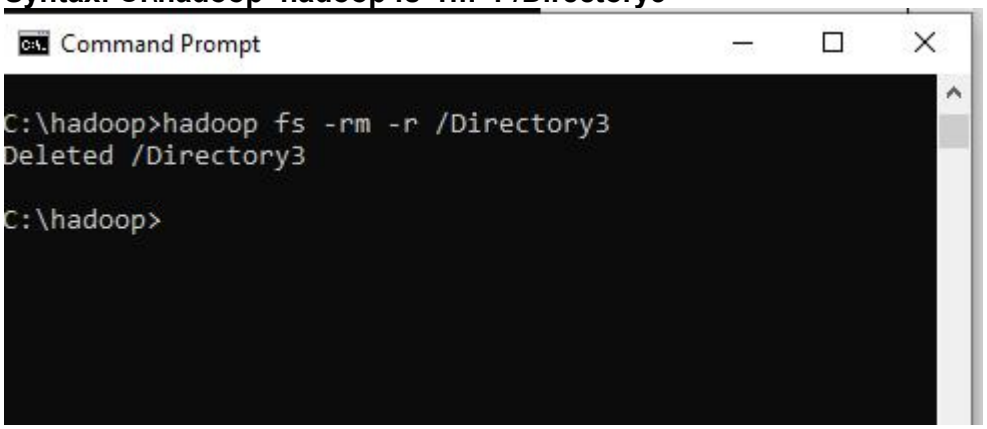
Remove a file or directory in HDFS.
Remove files specified as arguments. Deletes directory only when it is empty
Syntax: C:\hadoop>hadoop fs -rm /Directory2/Word.txt





Removing a Directory.

Syntax: C:\hadoop>hadoop fs -rm -r /Directory3



2.Develop a MapReduce program to implement Matrix Multiplication

Java :

Open a notepad or editor and save the below program as MatrixMultiply.java

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

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 MatrixMultiply {

   public static class MatrixMapper extends Mapper<LongWritable, Text, Text, Text>
{
      private int m = 3; // rows in A
      private int p = 3; // cols in B

      public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
         String[] tokens = value.toString().split(",");
         String matrix = tokens[0];
         int i = Integer.parseInt(tokens[1]);
         int j = Integer.parseInt(tokens[2]);
         String val = tokens[3];

         if (matrix.equals("A")) {
            for (int k = 0; k < p; k++) {
               context.write(new Text(i + "," + k), new Text("A," + j + "," + val));
            }
         } else if (matrix.equals("B")) {
            for (int k = 0; k < m; k++) {
               context.write(new Text(k + "," + j), new Text("B," + i + "," + val));
            }
         }
      }
   }

   public static class MatrixReducer extends Reducer<Text, Text, Text,
DoubleWritable> {
      public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
         Map<Integer, Double> A = new HashMap<>();
         Map<Integer, Double> B = new HashMap<>();

         for (Text val : values) {
            String[] tokens = val.toString().split(",");
```

```
            String matrixName = tokens[0];
            int index = Integer.parseInt(tokens[1]);
            double value = Double.parseDouble(tokens[2]);

            if (matrixName.equals("A")) {
                A.put(index, value);
            } else if (matrixName.equals("B")) {
                B.put(index, value);
            }
        }

        double sum = 0;
        for (int k : A.keySet()) {
            if (B.containsKey(k)) {
                sum += A.get(k) * B.get(k);
            }
        }

        context.write(key, new DoubleWritable(sum));
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Matrix Multiplication");

    job.setJarByClass(MatrixMultiply.class);
    job.setMapperClass(MatrixMapper.class);
    job.setReducerClass(MatrixReducer.class);

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

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

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

Create a matrix.txt file in a folder 'input'
Sample data:

```
A,0,0,1
A,0,1,2
A,1,0,3
A,1,1,4
B,0,0,5
B,1,0,6
```

In Command Prompt:(**Always run CMD as administrator only**)

c:\Users\KUMARESH\Documents>javac -classpath "C:\hadoop\share\hadoop\common*;C:\hadoop\share\hadoop\common\lib*;C:\hadoop\share\hadoop\mapreduce*;C:\hadoop\share\hadoop\mapreduce\lib*;C:\hadoop\share\hadoop\hdfs*;C:\hadoop\share\hadoop\hdfs\lib*" -d classes MatrixMultiply.java

c:\Users\KUMARESH\Documents>jar -cvf MatrixMultiply.jar -C classes/ .

c:\Users\KUMARESH\Documents>start-all.cmd

```
c:\Users\KUMARESH\Documents>javac -classpath "C:\hadoop\share\hadoop\common\*;C:\hadoop\share\hadoop\co
mon\lib\*;C:\hadoop\share\hadoop\mapreduce\*;C:\hadoop\share\hadoop\mapreduce\lib\*;C:\hadoop\share\had
op\hdfs\*;C:\hadoop\share\hadoop\hdfs\lib\*" -d classes MatrixMultiply.java

c:\Users\KUMARESH\Documents>jar -cvf MatrixMultiply.jar -C classes/ .
added manifest
adding: MatrixMultiply$MatrixMapper.class(in = 2238) (out= 968)(deflated 56%)
adding: MatrixMultiply$MatrixReducer.class(in = 2734) (out= 1223)(deflated 55%)
adding: MatrixMultiply.class(in = 1527) (out= 808)(deflated 47%)
adding: WordCount$IntSumReducer.class(in = 1739) (out= 742)(deflated 57%)
adding: WordCount$TokenizerMapper.class(in = 1926) (out= 857)(deflated 55%)
adding: WordCount.class(in = 1656) (out= 918)(deflated 44%)

c:\Users\KUMARESH\Documents>start-all.cmd
```

c:\hadoop>hdfs dfs -mkdir /inputmatrix

c:\hadoop>hdfs dfs -put input/matrix.txt /inputmatrix

```
c:\hadoop>hdfs dfs -put input/matrix.txt /inputmatrix
```

c:\Users\KUMARESH\Documents>hadoop jar MatrixMultiply.jar MatrixMultiply /inputmatrix /output3

```
c:\Users\KUMARESH\Documents>hadoop jar MatrixMultiply.jar MatrixMultiply /inputmatrix /output3
2025-04-08 19:42:11,431 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2025-04-08 19:42:12,752 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not perfo
rmed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-04-08 19:42:12,791 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hado
op-yarn/staging/KUMARESH/.staging/job_1744121327545_0001
2025-04-08 19:42:13,378 INFO input.FileInputFormat: Total input files to process : 1
2025-04-08 19:42:13,509 INFO mapreduce.JobSubmitter: number of splits:1
2025-04-08 19:42:13,780 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1744121327545_0001
2025-04-08 19:42:13,784 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-04-08 19:42:14,066 INFO conf.Configuration: resource-types.xml not found
2025-04-08 19:42:14,067 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-04-08 19:42:14,599 INFO impl.YarnClientImpl: Submitted application application_1744121327545_0001
2025-04-08 19:42:14,700 INFO mapreduce.Job: The url to track the job: http://DESKTOP-1HKM3JN:8088/proxy/
application_1744121327545_0001/
2025-04-08 19:42:14,704 INFO mapreduce.Job: Running job: job_1744121327545_0001
2025-04-08 19:42:34,089 INFO mapreduce.Job: Job job_1744121327545_0001 running in uber mode : false
2025-04-08 19:42:34,090 INFO mapreduce.Job:  map 0% reduce 0%
2025-04-08 19:42:42,234 INFO mapreduce.Job:  map 100% reduce 0%
2025-04-08 19:42:50,338 INFO mapreduce.Job:  map 100% reduce 100%
2025-04-08 19:42:56,436 INFO mapreduce.Job: Job job_1744121327545_0001 completed successfully
2025-04-08 19:42:56,578 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=222
```

Output:

```
c:\Users\KUMARESH\Documents>hdfs dfs -cat /output3/part-r-00000
0,0     17.0
0,1     0.0
0,2     0.0
1,0     39.0
1,1     0.0
1,2     0.0
2,0     0.0
```

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

Java :

Open a notepad or editor and save the below program as WeatherAnalysis.java

```
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.*;
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();

         if (!line.startsWith("Date")) {  // skip header
            String[] fields = line.split(",");
            String date = fields[0];
            String location = fields[1];
            double temperature = Double.parseDouble(fields[2]);

            String condition;
            if (temperature >= 35) {
               condition = "Hot";
            } else if (temperature <= 15) {
               condition = "Cold";
            } else {
               condition = "Moderate";
            }

            context.write(new Text(date + " - " + location), new Text("Weather is " +
condition));
         }
      }
   }

   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);
         }
      }
```

```
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Weather Condition Analyzer");

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

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

Save input file as Weather.csv in(C:/users/documents/input/):

```
2025-04-01,Delhi,38
2025-04-01,Shimla,10
2025-04-01,Mumbai,30
2025-04-01,Bengaluru,28
2025-04-01,Chennai,36
2025-04-02,Delhi,37
2025-04-02,Shimla,12
2025-04-02,Mumbai,32
2025-04-02,Bengaluru,29
2025-04-02,Chennai,35
2025-04-03,Delhi,34
2025-04-03,Shimla,8
2025-04-03,Mumbai,31
2025-04-03,Bengaluru,27
2025-04-03,Chennai,37
2025-04-04,Delhi,40
2025-04-04,Shimla,14
2025-04-04,Mumbai,33
2025-04-04,Bengaluru,26
2025-04-04,Chennai,38
```

In Command Prompt:(**Always run CMD as administrator only**)

```
c:\Users\KUMARESH\Documents>javac -classpath
"C:\hadoop\share\hadoop\common\*;C:\hadoop\share\hadoop\common\lib\*;C:\hadoop\share\
hadoop\mapreduce\*;C:\hadoop\share\hadoop\mapreduce\lib\*;C:\hadoop\share\hadoop\hdfs\
*;C:\hadoop\share\hadoop\hdfs\lib\*" -d classes WeatherAnalysis.java

c:\Users\KUMARESH\Documents>jar -cvf WeatherAnalysis.jar -C classes/ .

c:\Users\KUMARESH\Documents>start-all.cmd
```

c:\Users\KUMARESH\Documents>hdfs dfs -mkdir /inputWeather

c:\Users\KUMARESH\Documents>hdfs dfs -put input/Weather.csv /inputWeather

```
c:\Users\KUMARESH\Documents>hdfs dfs -mkdir /inputWeather

c:\Users\KUMARESH\Documents>hdfs dfs -put input/Weather.csv /inputWeather
```

c:\Users\KUMARESH\Documents>hadoop jar WeatherAnalysis.jar WeatherAnalysis /inputWeather /output4

```
c:\Users\KUMARESH\Documents>hadoop jar WeatherAnalysis.jar WeatherAnalysis /inputWeather /output4
2025-04-17 21:42:47,265 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2025-04-17 21:42:48,024 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not perfo
rmed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-04-17 21:42:48,055 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadc
op-yarn/staging/KUMARESH/.staging/job_1744906270757_0001
2025-04-17 21:42:48,276 INFO input.FileInputFormat: Total input files to process : 1
2025-04-17 21:42:48,834 INFO mapreduce.JobSubmitter: number of splits:1
2025-04-17 21:42:49,367 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1744906270757_0001
2025-04-17 21:42:49,369 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-04-17 21:42:49,504 INFO conf.Configuration: resource-types.xml not found
2025-04-17 21:42:49,505 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-04-17 21:42:49,735 INFO impl.YarnClientImpl: Submitted application application_1744906270757_0001
2025-04-17 21:42:49,774 INFO mapreduce.Job: The url to track the job: http://DESKTOP-1HKM3JN:8088/proxy/
application_1744906270757_0001/
2025-04-17 21:42:49,774 INFO mapreduce.Job: Running job: job_1744906270757_0001
2025-04-17 21:43:04,045 INFO mapreduce.Job: Job job_1744906270757_0001 running in uber mode : false
2025-04-17 21:43:04,046 INFO mapreduce.Job:  map 0% reduce 0%
2025-04-17 21:43:10,158 INFO mapreduce.Job:  map 100% reduce 0%
2025-04-17 21:43:16,249 INFO mapreduce.Job:  map 100% reduce 100%
```

Output:

c:\Users\KUMARESH\Documents>hdfs dfs -cat /output4/part-r-00000

```
c:\Users\KUMARESH\Documents>hdfs dfs -cat /output4/part-r-00000
2025-04-01 - Bengaluru  Weather is Moderate
2025-04-01 - Chennai    Weather is Hot
2025-04-01 - Delhi      Weather is Hot
2025-04-01 - Mumbai     Weather is Moderate
2025-04-01 - Shimla     Weather is Cold
2025-04-02 - Bengaluru  Weather is Moderate
2025-04-02 - Chennai    Weather is Hot
2025-04-02 - Delhi      Weather is Hot
2025-04-02 - Mumbai     Weather is Moderate
2025-04-02 - Shimla     Weather is Cold
2025-04-03 - Bengaluru  Weather is Moderate
2025-04-03 - Chennai    Weather is Hot
2025-04-03 - Delhi      Weather is Moderate
2025-04-03 - Mumbai     Weather is Moderate
2025-04-03 - Shimla     Weather is Cold
2025-04-04 - Bengaluru  Weather is Moderate
2025-04-04 - Chennai    Weather is Hot
2025-04-04 - Delhi      Weather is Hot
2025-04-04 - Mumbai     Weather is Moderate
2025-04-04 - Shimla     Weather is Cold
```

4.Develop a MapReduce program to find the tags associated with each movie by analyzing movie lensdata.

Java :

Open a notepad or editor and save the below program as

MovieTagJoinDriver.java

```
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.lib.input.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MovieTagJoinDriver {
   public static void main(String[] args) throws Exception {
      if (args.length != 3) {
         System.err.println("Usage: MovieTagJoinDriver
<movie.csv><tag.csv><output>");
         System.exit(-1);
      }

      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "Movie Tags Join");

      job.setJarByClass(MovieTagJoinDriver.class);
      job.setReducerClass(JoinTagReducer.class);

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

      MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,
TagMovieMapper.class);
      MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,
TagMovieMapper.class);
      FileOutputFormat.setOutputPath(job, new Path(args[2]));

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

TagMovieMapper.java

```
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

public class TagMovieMapper extends Mapper<LongWritable, Text, Text, Text> {
   private boolean isTagFile = false;
```

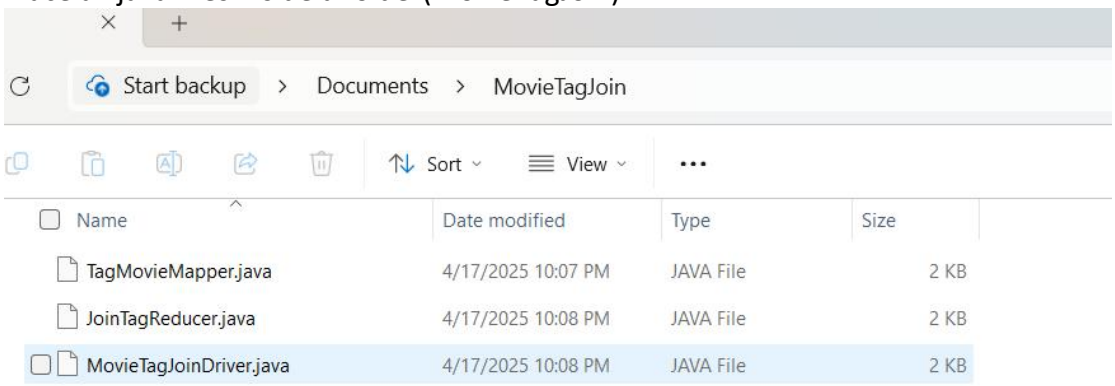
```
  @Override
  protected void setup(Context context) {
    String filePath = context.getInputSplit().toString();
    isTagFile = filePath.contains("tag");
  }

  public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
    String[] fields = value.toString().split(",", -1);
    if (isTagFile) {
      if (!fields[0].equals("userId") && fields.length >= 3) {
        String movieId = fields[1].trim();
        String tag = fields[2].trim();
        context.write(new Text(movieId), new Text("T|" + tag));
      }
    } else {
      if (!fields[0].equals("movieId") && fields.length >= 2) {
        String movieId = fields[0].trim();
        String title = fields[1].trim();
        context.write(new Text(movieId), new Text("M|" + title));
      }
    }
  }
}
```

JoinTagReducer.Java

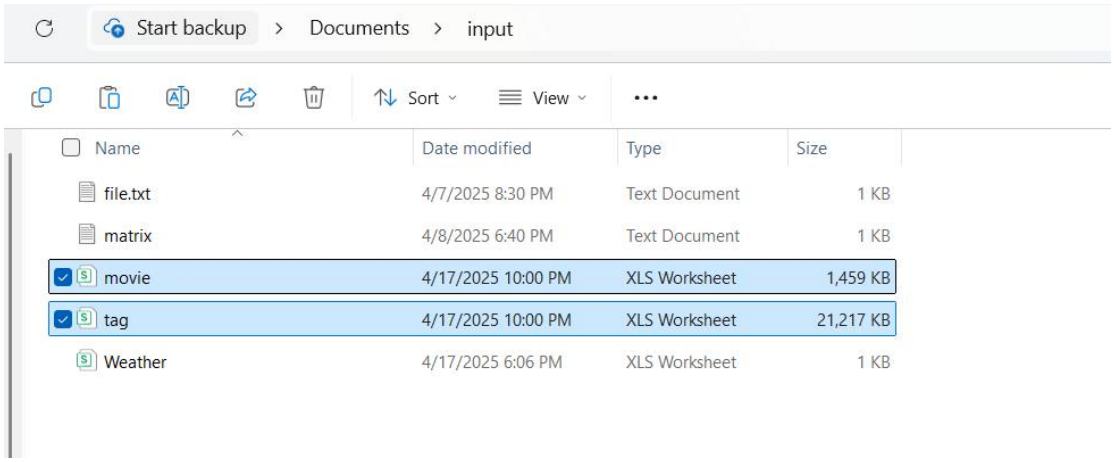
```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class JoinTagReducer extends Reducer<Text, Text, Text, Text> {
  public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
    String movieTitle = null;
    Map<String, Integer> tagCount = new HashMap<>();

    for (Text val : values) {
      String[] parts = val.toString().split("\\|", 2);
      if (parts[0].equals("M")) {
        movieTitle = parts[1];
      } else if (parts[0].equals("T")) {
        String tag = parts[1];
        tagCount.put(tag, tagCount.getOrDefault(tag, 0) + 1);
      }
    }
```

```
        if (movieTitle != null && !tagCount.isEmpty()) {
            List<String> tagList = new ArrayList<>();
            for (Map.Entry<String, Integer> entry : tagCount.entrySet()) {
                tagList.add(entry.getKey() + ": " + entry.getValue());
            }
            context.write(new Text(movieTitle), new Text(String.join(", ", tagList)));
        }
    }
}
```

Place all java files inside a folder(MovieTagJoin)



Download movie lens dataset from below Kaggle link
(https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset/code)and place the movie and tag csv files in C:/Documents/input folder as shown below.



In Command Prompt:(**Always run CMD as administrator only**)

c:\Users\KUMARESH\Documents\MovieTagJoin>javac -classpath "C:\hadoop\share\hadoop\common*;C:\hadoop\share\hadoop\common\lib*;C:\hadoop\share\hadoop\mapreduce*;C:\hadoop\share\hadoop\mapreduce\lib*;C:\hadoop\share\hadoop\hdfs*;C:\hadoop\share\hadoop\hdfs\lib*" -d . MovieTagJoinDriver.java TagMovieMapper.java JoinTagReducer.java

c:\Users\KUMARESH\Documents\MovieTagJoin>jar cf movietagjoin.jar *.class

c:\Users\KUMARESH\Documents\MovieTagJoin>cd ..

c:\Users\KUMARESH\Documents>hdfs dfs -mkdir /inputMovies

c:\Users\KUMARESH\Documents\input>hdfs dfs -put movie.csv /inputMovies

c:\Users\KUMARESH\Documents\input>hdfs dfs -put tag.csv /inputMovies

```
c:\Users\KUMARESH\Documents\input>hdfs dfs -put movie.csv /inputMovies

c:\Users\KUMARESH\Documents\input>hdfs dfs -put tag.csv /inputMovies
```

c:\Users\KUMARESH\Documents\MovieTagJoin>hadoop jar movietagjoin.jar MovieTagJoinDriver /inputMovies/movie.csv /inputMovies/tag.csv /output5/movie-tags

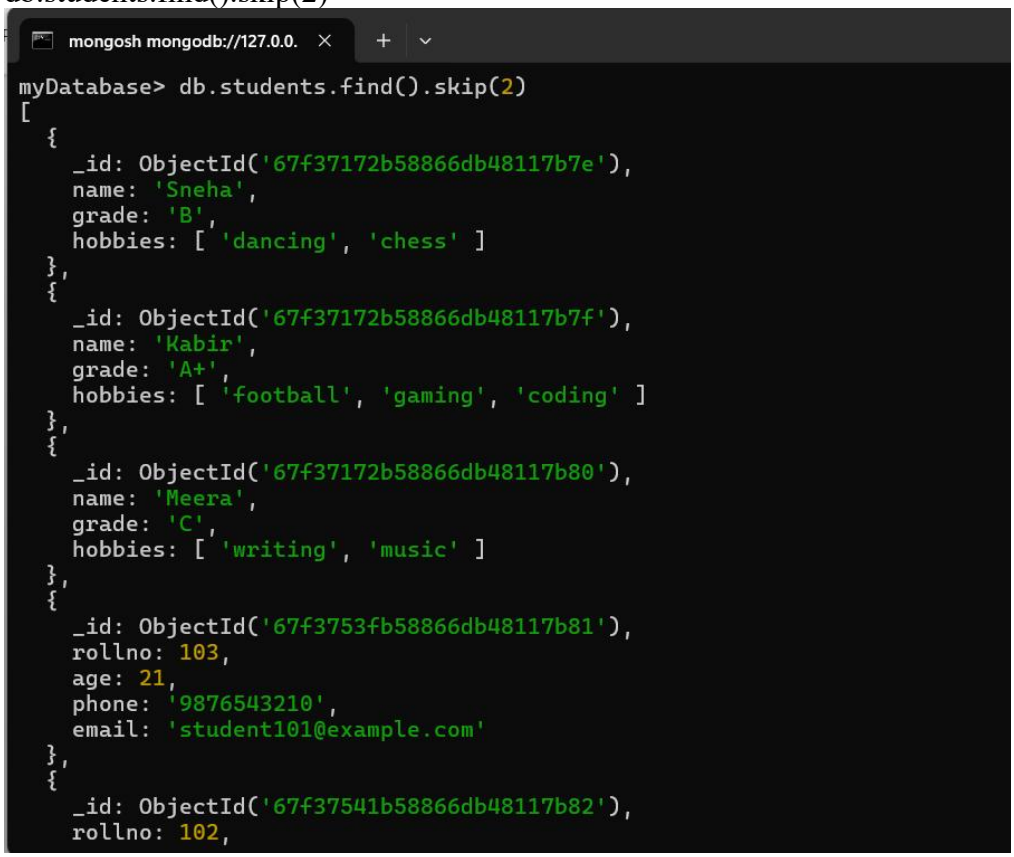
OUTPUT:
c:\Users\KUMARESH\Documents\MovieTagJoin>hdfs dfs -cat /output5/movie-tags/part-r-00000


```
"title" "tag": 1
"Toy Story (1995)"      "CGI": 1, "Tim Allen": 5, "children": 19, "pixar": 12, "family": 12, "story": 1, "lots of heart": 1, "Disney": 26, "Animation": 1, "bright": 1, "fantasy": 8,
"rated-G": 1, "toys": 13, "time travel": 11, "imdb top 250": 5, "animated": 10, "55 movies every kid should see--Entertainment Weekly": 1, "want to see again": 1, "cute": 1, "humorous":
11, "Best of Rotten Tomatoes: All Time": 1, "Watched": 1, "National Film Registry": 1, "clever": 5, "innovative": 1, "John Lasseter": 3, "witty": 14, "Woody": 1, "villian hurts toys": 1,
"action figure": 1, "kids and family": 1, "CLV": 1, "fun": 2, "CG animation": 1, "classic": 7, "fanciful": 1, "toy": 2, "2009 reissue in Stereoscopic 3-D": 1, "soothing": 1, "voice
acting": 1, "light": 1, "dolls": 1, "HEROIC MISSION": 1, "funny": 18, "ya boy": 1, "Pixar": 67, "Tumey's To See Again": 1, "Tom Hanks": 23, "USA": 1, "computer animation": 20, "buddy
movie": 4, "comedy": 13, "the boys": 1, "Friendship": 2, "first cgi film": 1, "cgi": 3, "BD-Video": 1, "Cartoon": 2, "animation": 48, "Pixar animation": 3, "DVD-Video": 1, "UNLIKELY
FRIENDSHIPS": 1, "rousing": 1, "disney": 2, "almost favorite": 1, "action figures": 1, "ê¯ºä¸€é¸£": 1, "family film": 1, "warm": 1, "erlend's DVDs": 1, "Disney animated feature": 1,
"DARING RESCUES": 1, "kids movie": 1, "adventure": 10, "Want": 1, "TOYS COME TO LIFE": 1, "TÃ©a Leoni does not star in this movie": 1, "very good": 1, "Tumey's VHS": 1, "buy": 1, "Buzz
Lightyear": 1, "Engaging": 1, "avi": 1, "3D": 3
"GoldenEye (1995)"      "bond": 1, "Puerto Rico": 1, "owned": 1, "killer as protagonist": 1, "espionage": 6, "spies": 2, "assassin": 1, "violence": 2, "murder": 1, "BD-Video": 1,
"gadgets": 2, "007": 12, "Tumey's DVDs": 1, "boys with toys": 1, "Best Of the Brosnon Bonds": 1, "Latin America": 1, "sexuality": 1, "I wanted the bad guy to win.  Sean Bean out-Bonds
Bond.": 1, "James Bond": 12, "sequel": 1, "action": 3, "Bob*ola": 1, "Pierce Brosnan": 7, "Memorable Characters": 1, "007 movies are bad.": 1, "it's a good movie but the end of 007 is
always the same": 1, "Bond": 12, "secret service": 2, "CLV": 1, "adventure": 1, "Caribbean": 1, "memorable lines": 1, "Sean Bean dies": 1, "good dialogue": 2, "MTSKAF": 1, "btaege": 1,
"Judi Dench": 1, "franchise": 1, "seen more than once": 2, "one-liners": 1, "tank chase scene": 1, "dark": 1, "james bond": 2, "funny": 1, "007 (series)": 2
"City Hall (1996)"      "Al Pacino": 3, "corruption": 2, "CLV": 1, "politics": 2, "Harold Becker": 1, "own": 1
"Curdled (1996)"        "netflix": 1
```

5.Implement Functions: Count – Sort – Limit – Skip – Aggregate using MongoDB

```
db.createCollection("students");

//Insert documents
db.students.insertMany([
  {
   name: "Sneha",
   grade: "B",
   hobbies: ["dancing", "chess"]
  },
  {
   name: "Kabir",
   grade: "A",
   hobbies: ["football", "gaming", "coding"]
  },
  {
   name: "Meera",
   grade: "C",
   hobbies: ["writing", "music"]
  }
]);

//  Find all students and pretty print
db.students.find().pretty();
[
   {
   _id: ObjectId('67f37172b58866db48117b7e'),
   name: 'Sneha',
   grade: 'B',
   hobbies: [ 'dancing', 'chess' ]
  },
  {
   _id: ObjectId('67f37172b58866db48117b7f'),
   name: 'Kabir',
   grade: 'A+',
   hobbies: [ 'football', 'gaming', 'coding' ]
  },
  {
   _id: ObjectId('67f37172b58866db48117b80'),
   name: 'Meera',
   grade: 'C',
   hobbies: [ 'writing', 'music' ]
  },
  {
   _id: ObjectId('67f3753fb58866db48117b81'),
   rollno: 103,
   age: 21,
   phone: '9876543210',
   email: 'student101@example.com'
  },
```

```
  {
    _id: ObjectId('67f37541b58866db48117b82'),
    rollno: 102,
    age: 19,
    phone: '9876543210',
    email: 'student101@example.com'
  }
]

// Returns the total number of documents in 'students' collection
db.students.countDocuments();
5
// Count with a condition
// Returns number of students with grade A
db.students.countDocuments({ grade: "A" });
1
//Skip Documents
db.students.find().skip(2)
```




```
//limit the number of results
db.students.find().limit(3)
```

```
myDatabase> db.students.find().limit(3)
[
  {
    _id: ObjectId('67f3710fb58866db48117b7c'),
    rollno: 101,
    age: 20,
    phone: '9876543210',
    email: 'student101@example.com'
  },
  {
    _id: ObjectId('67f3715bb58866db48117b7d'),
    name: 'Aarav',
    grade: 'A',
    hobbies: [ 'reading', 'cycling', 'painting' ],
    location: null
  },
  {
    _id: ObjectId('67f37172b58866db48117b7e'),
    name: 'Sneha',
    grade: 'B',
    hobbies: [ 'dancing', 'chess' ]
  }
]
```

// Ascending sort (1 = ascending)
db.students.find().sort({ age: 1 })
// Descending sort (-1 = descending)
db.students.find().sort({ age: -1 })

```
myDatabase> db.students.find().sort({ age: -1 })
[
  {
    _id: ObjectId('67f3753fb58866db48117b81'),
    rollno: 103,
    age: 21,
    phone: '9876543210',
    email: 'student101@example.com'
  },
  {
    _id: ObjectId('67f3710fb58866db48117b7c'),
    rollno: 101,
    age: 20,
    phone: '9876543210',
    email: 'student101@example.com'
  },
  {
    _id: ObjectId('67f37541b58866db48117b82'),
    rollno: 102,
    age: 19,
    phone: '9876543210',
    email: 'student101@example.com'
```

//Aggregate Functions

```
// Insert multiple customer documents with fields: CustID, accbal, acctype
db.customer.insertMany([
  { CustID: "C123", accbal: 500,  acctype: "S" },
  { CustID: "C123", accbal: 900,  acctype: "S" },
  { CustID: "C111", accbal: 1200, acctype: "S" },
  { CustID: "C123", accbal: 1500, acctype: "C" }
]);
// Aggregate total balance per customer
db.customer.aggregate([
  {
   $group: {
     _id: "$CustID",          // Group by CustID
     total_balance: { $sum: "$accbal" } // Sum of accbal
   }
  }
]);
```

```
myDatabase> db.customer.aggregate([
...   {
...     $group: {
...       _id: "$CustID",           // Group by CustID
...       total_balance: { $sum: "$accbal" } // Sum of accbal
...     }
...   }
... ]);
[
  { _id: 'C123', total_balance: 2900 },
  { _id: 'C111', total_balance: 1200 }
]
```

```
// Find max and min account balances for each customer
db.customer.aggregate([
  {
   $group: {
     _id: "$CustID",
     max_balance: { $max: "$accbal" },
     min_balance: { $min: "$accbal" }
   }
  }
]);
```

```
myDatabase> db.customer.aggregate([
...   {
...     $group: {
...       _id: "$CustID",
...       max_balance: { $max: "$accbal" },
...       min_balance: { $min: "$accbal" }
...     }
...   }
... ]);
[
  { _id: 'C123', max_balance: 1500, min_balance: 500 },
  { _id: 'C111', max_balance: 1200, min_balance: 1200 }
]
```

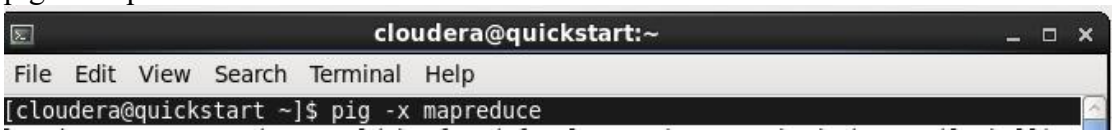
```
// Average balance per account type
db.customer.aggregate([
```

```
  {
   $group: {
     _id: "$acctype",
     avg_balance: { $avg: "$accbal" }
   }
  }
]);
```

```
myDatabase> db.customer.aggregate([
...   {
...     $group: {
...       _id: "$acctype",
...       avg_balance: { $avg: "$accbal" }
...     }
...   }
... ]);
[
  { _id: 'S', avg_balance: 866.6666666666666 },
  { _id: 'C', avg_balance: 1500 }
]
```

6.Develop Pig Latin scripts to sort, group, join, project, and filter the data.

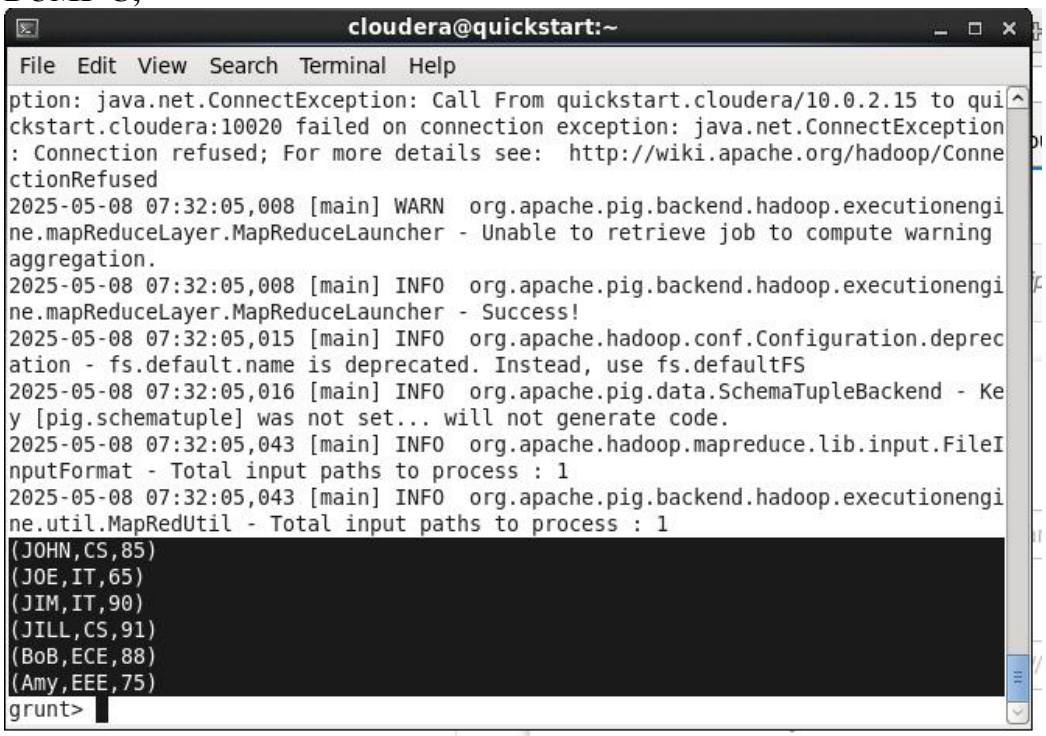
pig -x mapreduce



Order:

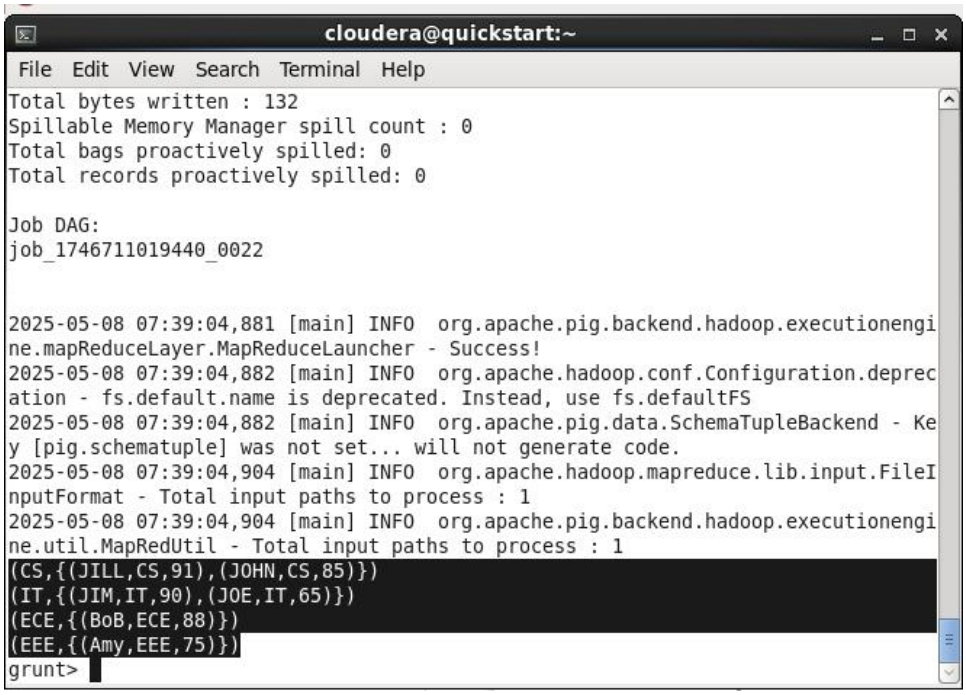
Order the tuples by name

A = load '/user/cloudera/pigdemo/student.txt' as (name:chararray, dept:chararray, gpa:int);
C = ORDER A BY name DESC;
DUMP C;



Group:

A = load '/user/cloudera/pigdemo/student.txt' as (name:chararray, dept:chararray, gpa:int);
group_dept = GROUP A BY dept;
DUMP group_dept;



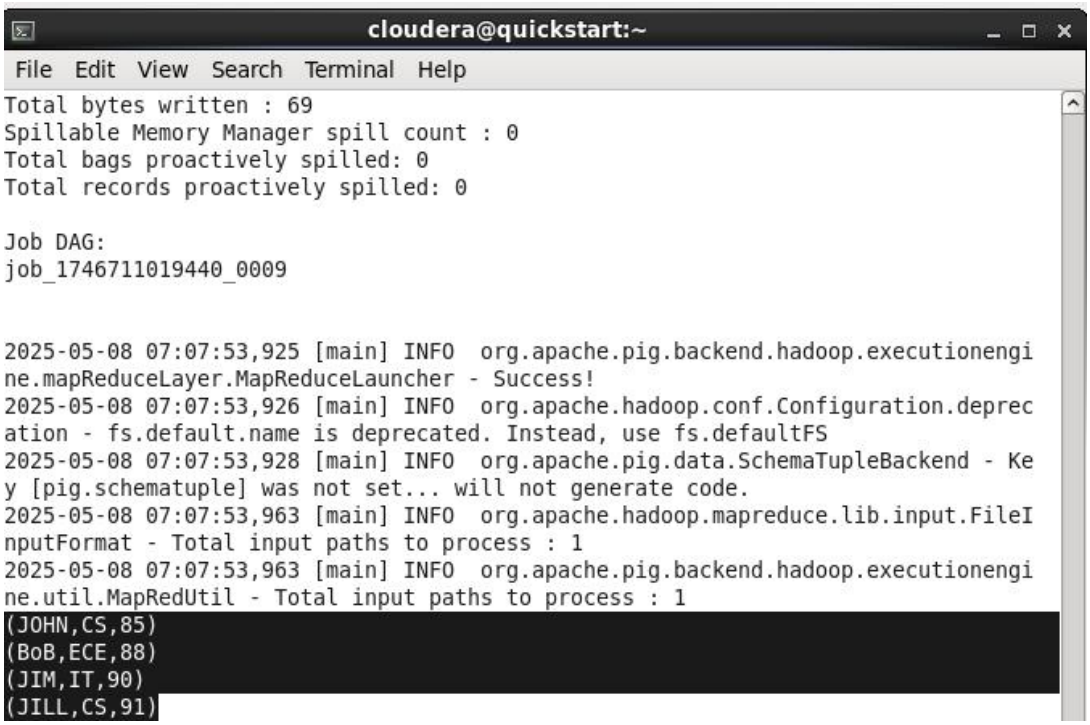
Filter

Find the tuples of those student where the GPA is greater than 80.

A = load '/user/cloudera/pigdemo/student.txt' as (name:chararray, dept:chararray, gpa:int);

B = filter A by gpa > 80;

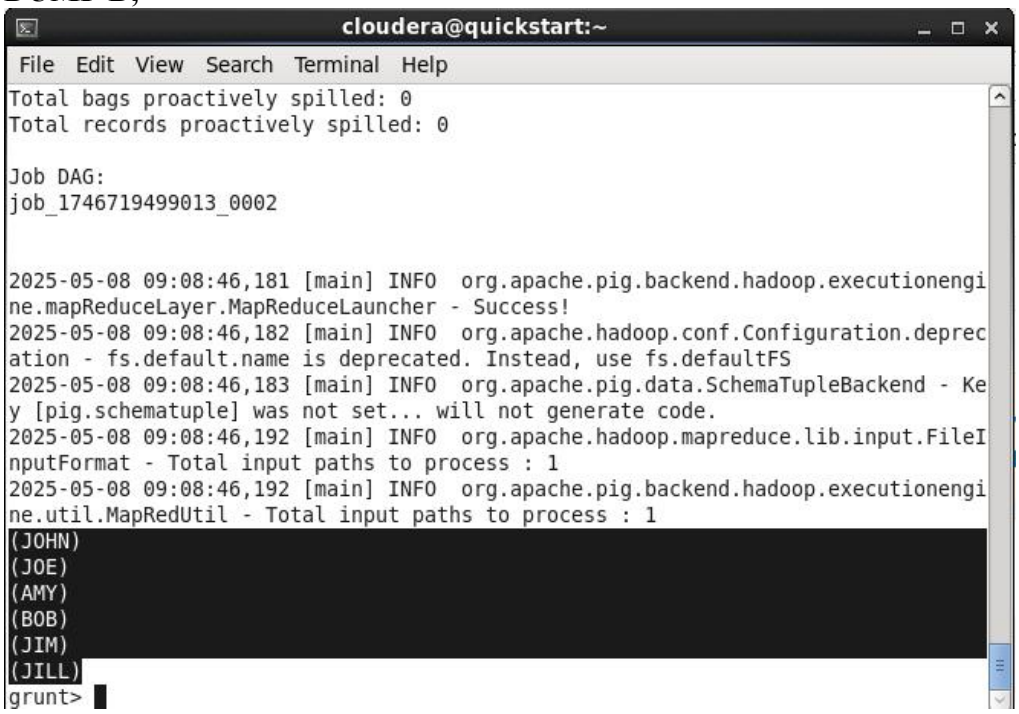
DUMP B;



Project:

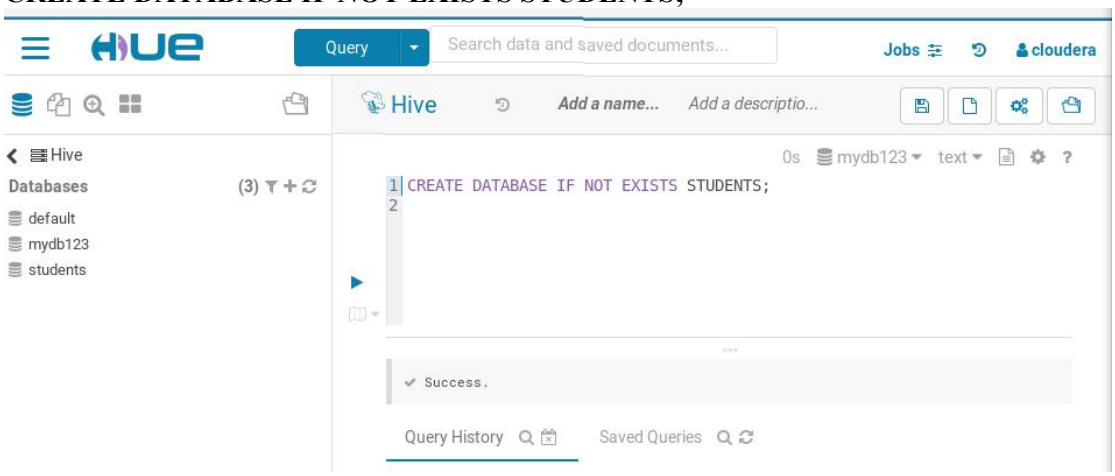
Project the students name

A = load '/user/cloudera/pigdemo/student.txt' as (name:chararray, dept:chararray, gpa:int);
B = foreach A generate UPPER (name);
DUMP B;

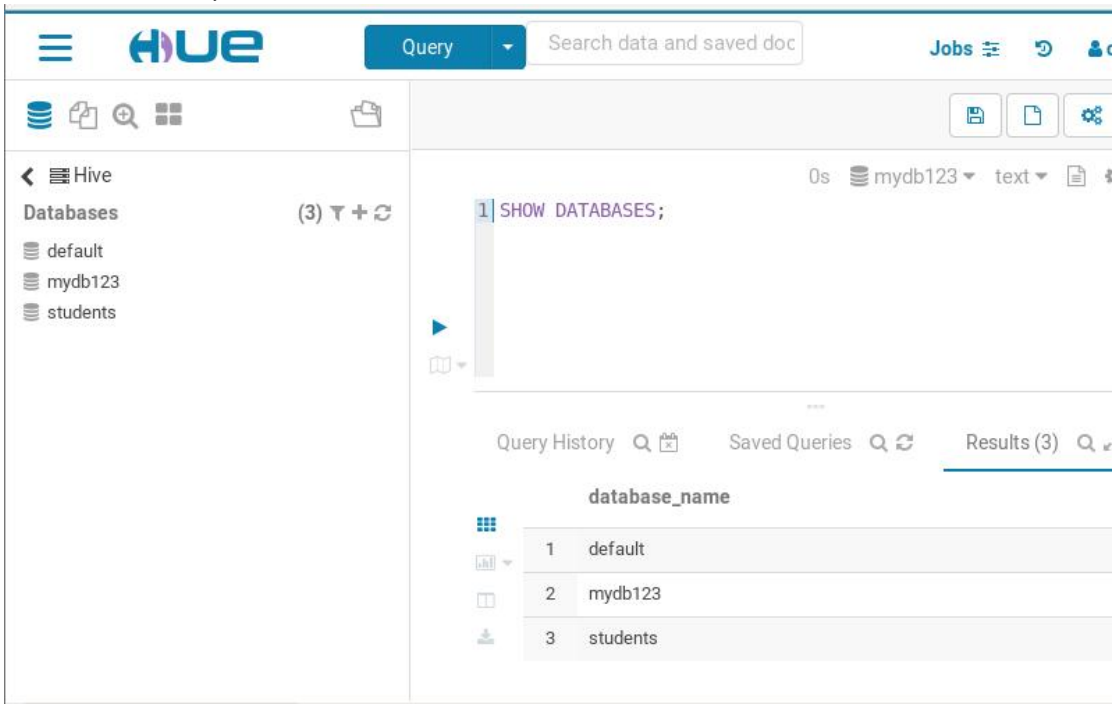


7. Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.

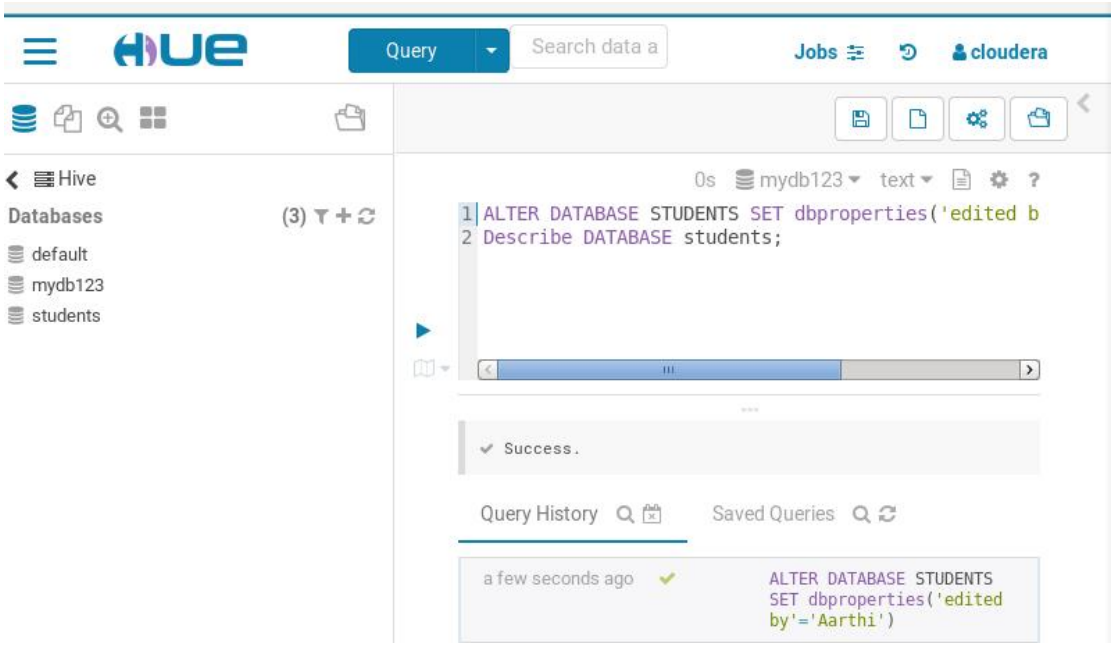
CREATE DATABASE IF NOT EXISTS STUDENTS;



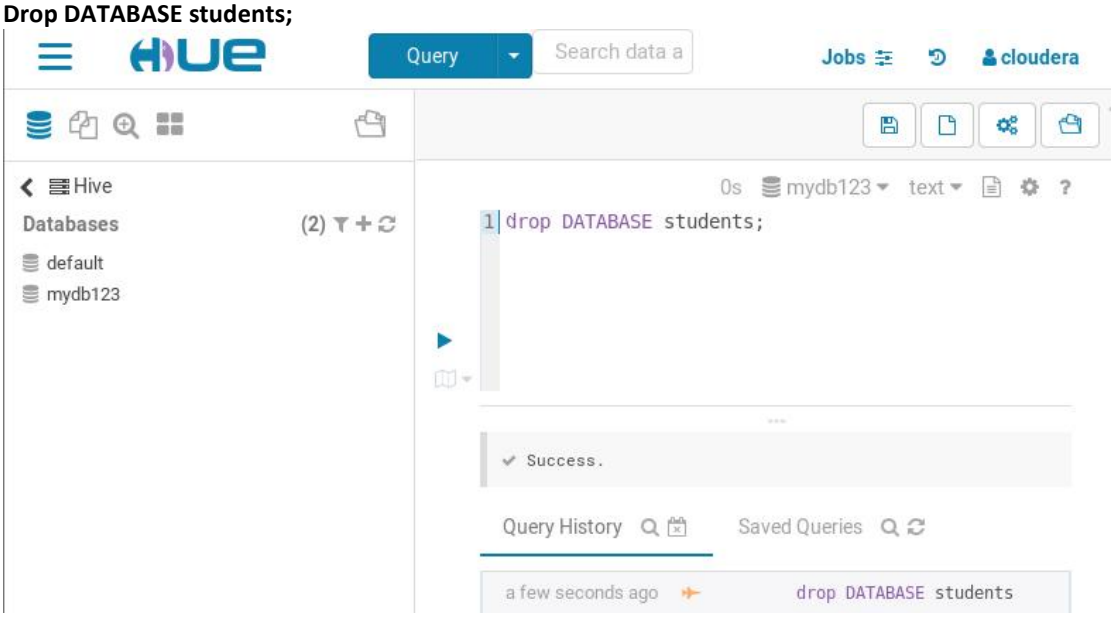
SHOW DATABASES;



ALTER DATABASE STUDENTS SET dbproperties('edited by' = 'Aarthi');



Drop DATABASE students;

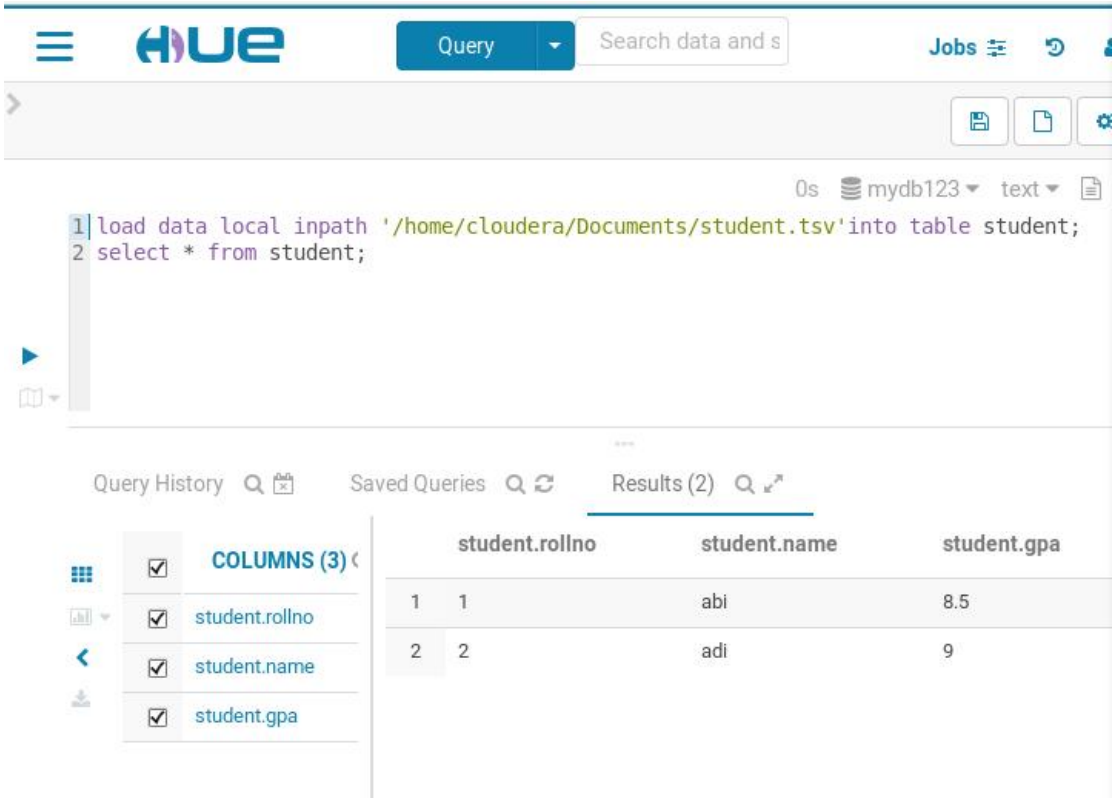


Create database students;

Create table if not exists
Student(rollno INT, NAME STRING,GPA FLOAT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

Describe student;

load data local inpath '/home/cloudera/Documents/students.tsv' into table student;
Select * from student;

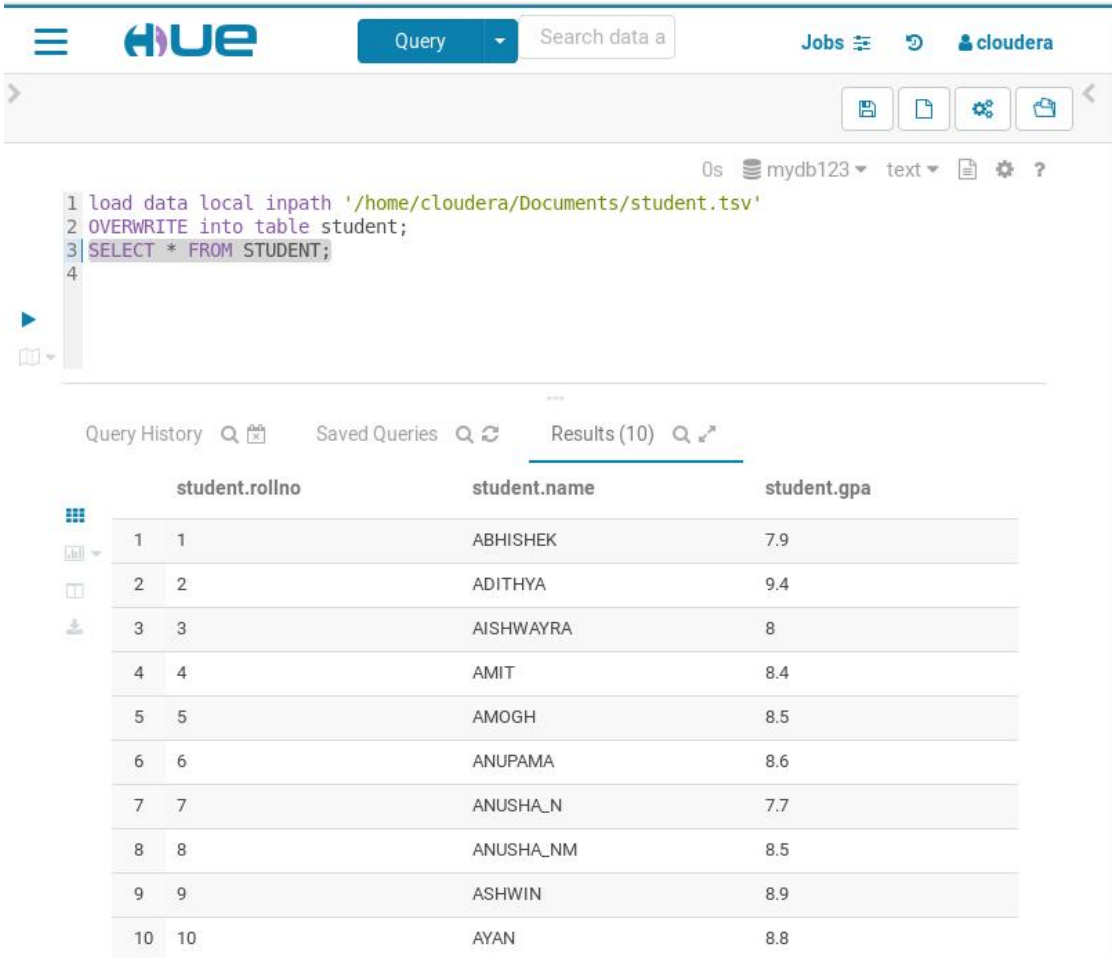


Sample data student.tsv

student.tsv

```
001	ABHISHEK	7.9
002	ADITHYA	9.4
003	AISHWAYRA	8.0
004	AMIT	8.4
005	AMOGH	8.5
006	ANUPAMA	8.6
007	ANUSHA_N	7.7
008	ANUSHA_NM	8.5
009	ASHWIN	8.9
010	AYAN	8.8
```

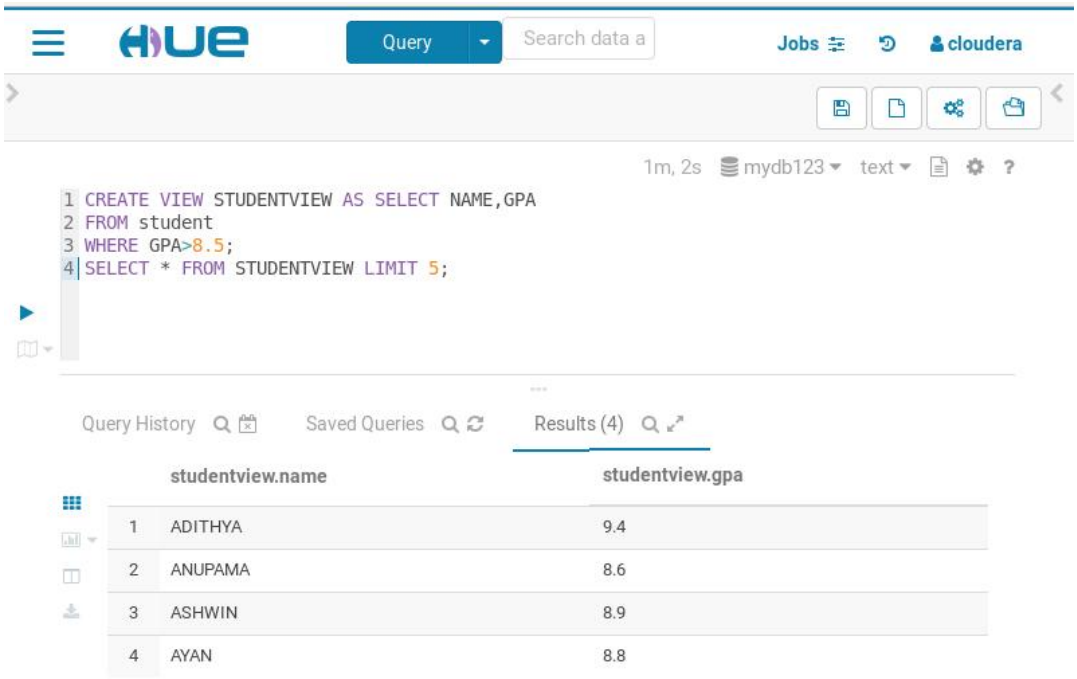
Select * from student;



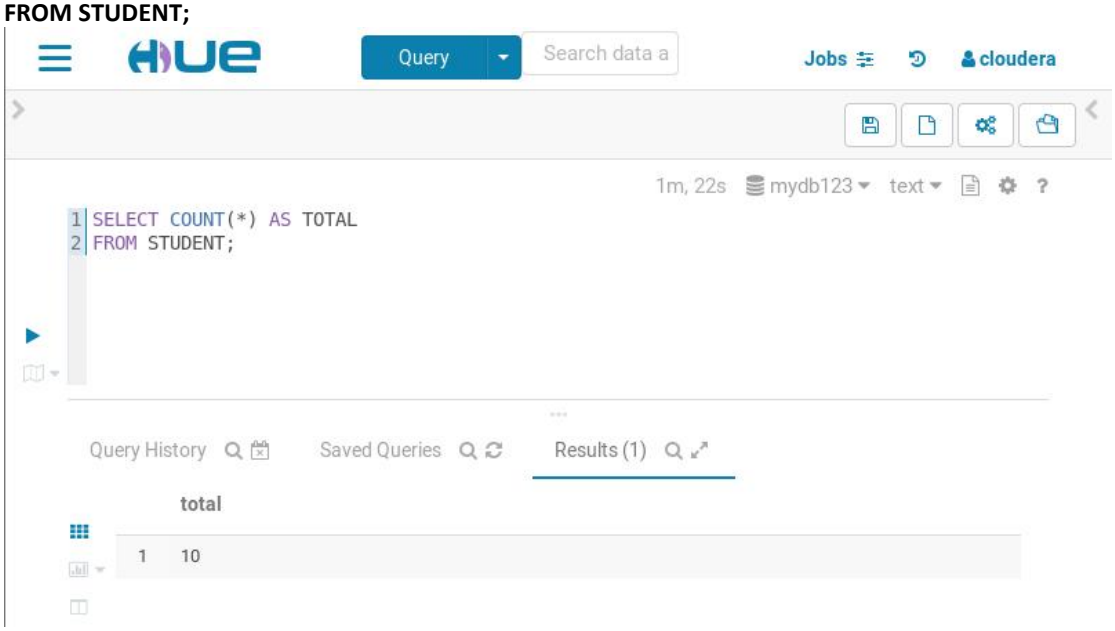
	student.rollno	student.name	student.gpa
1	1	ABHISHEK	7.9
2	2	ADITHYA	9.4
3	3	AISHWAYRA	8
4	4	AMIT	8.4
5	5	AMOGH	8.5
6	6	ANUPAMA	8.6
7	7	ANUSHA_N	7.7
8	8	ANUSHA_NM	8.5
9	9	ASHWIN	8.9
10	10	AYAN	8.8

CREATE VIEW STUDENTVIEW AS SELECT NAME,GPA
FROM STUDENT
WHERE GPA=8.5;

SELECT * FROM STUDENTVIEW LIMIT 5;

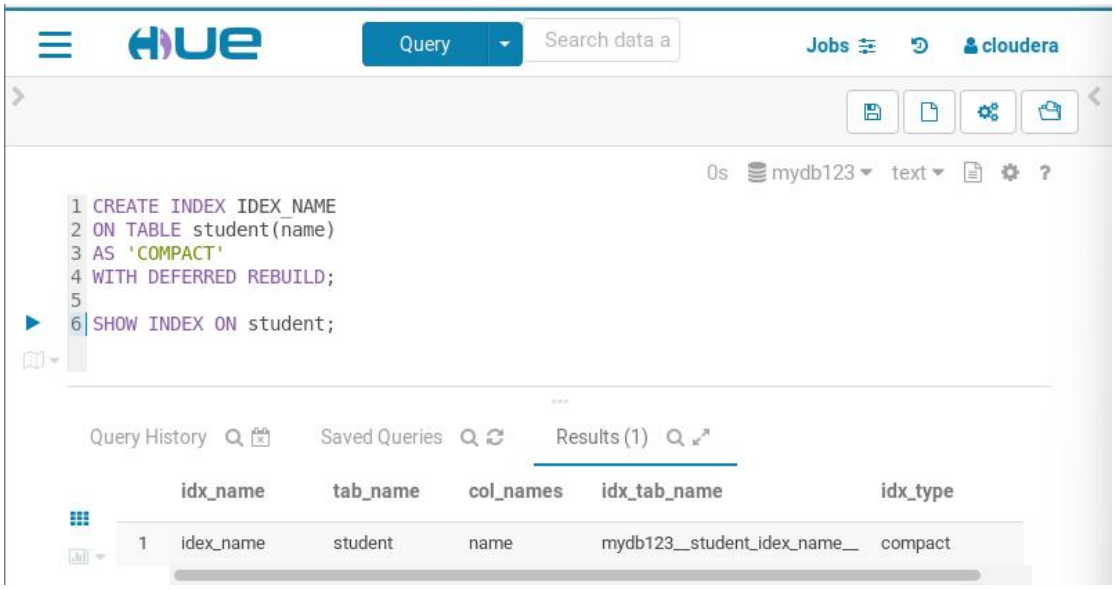


SELECT COUNT(*) AS TOTAL
FROM STUDENT;



CREATE INDEX INDEX_NAME
ON TABLE student(name)
AS 'COMPACT'
WITH DEFERED REBUILD;

SHOW INDEX ON student;


HUE
Query
Search data a
Jobs
cloudera
0s
mydb123
text
1 CREATE INDEX IDEX_NAME
2 ON TABLE student(name)
3 AS 'COMPACT'
4 WITH DEFERRED REBUILD;
5
6 SHOW INDEX ON student;
Query History
Saved Queries
Results (1)
idx_name
tab_name
col_names
idx_tab_name
idx_type
1
idex_name
student
name
mydb123__student_idex_name__
compact

8.Implement a word count program in Hadoop

Java :

Open a notepad or editor and save the below program as WordCount.java

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

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

public class WordCount {

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

      public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
         StringTokenizer itr = new StringTokenizer(value.toString());
         while (itr.hasMoreTokens()) {
            word.set(itr.nextToken().replaceAll("\\W+", "").toLowerCase());
            if (!word.toString().isEmpty()) {
               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 {
      if (args.length != 2) {
         System.err.println("Usage: WordCount <input path> <output path>");
         System.exit(-1);
      }

      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "word count");

      job.setJarByClass(WordCount.class);
      job.setMapperClass(TokenizerMapper.class);
      job.setCombinerClass(IntSumReducer.class);
      job.setReducerClass(IntSumReducer.class);

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

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

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

Create a sample file.txt file in a folder 'input'(in C:/hadoop/input)
Sample data:
apple banana apple
orange banana apple
grape orange banana
apple banana apple
grape orange banana

In Command Prompt:(**Always run CMD as adimistrator only**)

C:\Users\KUMARESH\Documents>mkdir -p classes

C:\Users\KUMARESH\Documents>javac -classpath "C:\hadoop\share\hadoop\common*;C:\hadoop\share\hadoop\common\lib*;C:\hadoop\share\hadoop\mapreduce*;C:\hadoop\share\hadoop\mapreduce\lib*;C:\hadoop\share\hadoop\hdfs*;C:\hadoop\share\hadoop\hdfs\lib*" -d classes WordCount.java

C:\Users\KUMARESH\Documents>jar -cvf wordcount.jar -C classes/ .
added manifest
adding: WordCount$IntSumReducer.class(in = 1739) (out= 742)(deflated 57%)
adding: WordCount$TokenizerMapper.class(in = 1926) (out= 857)(deflated 55%)
adding: WordCount.class(in = 1656) (out= 918)(deflated 44%)

c:\hadoop>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

c:\hadoop>hdfs dfs -mkdir -p /input

c:\hadoop>hdfs dfs -put input/file.txt /input

```
c:\hadoop>hdfs dfs -put c:\hadoop\input\file.txt /input
```

c:\hadoop>hadoop jar wordcount.jar WordCount /input /output2

```
c:\Users\KUMARESH\Documents>hadoop jar wordcount.jar WordCount /input /output2
2025-04-07 22:37:38,017 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2025-04-07 22:37:38,689 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not perfo
rmed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-04-07 22:37:38,716 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hado
op-yarn/staging/KUMARESH/.staging/job_1744045620350_0001
2025-04-07 22:37:39,410 INFO input.FileInputFormat: Total input files to process : 1
2025-04-07 22:37:39,472 INFO mapreduce.JobSubmitter: number of splits:1
2025-04-07 22:37:39,584 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1744045620350_0001
2025-04-07 22:37:39,586 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-04-07 22:37:39,716 INFO conf.Configuration: resource-types.xml not found
2025-04-07 22:37:39,716 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-04-07 22:37:39,937 INFO impl.YarnClientImpl: Submitted application application_1744045620350_0001
2025-04-07 22:37:39,970 INFO mapreduce.Job: The url to track the job: http://DESKTOP-1HKM3JN:8088/proxy/
application_1744045620350_0001/
2025-04-07 22:37:39,971 INFO mapreduce.Job: Running job: job_1744045620350_0001
2025-04-07 22:37:53,215 INFO mapreduce.Job: Job job_1744045620350_0001 running in uber mode : false
2025-04-07 22:37:53,216 INFO mapreduce.Job:  map 0% reduce 0%
2025-04-07 22:37:58,327 INFO mapreduce.Job:  map 100% reduce 0%
2025-04-07 22:38:05,422 INFO mapreduce.Job:  map 100% reduce 100%
2025-04-07 22:38:10,498 INFO mapreduce.Job: Job job_1744045620350_0001 completed successfully
2025-04-07 22:38:10,596 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=56
                FILE: Number of bytes written=478157
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=347
                HDFS: Number of bytes written=36
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
```

c:\Users\KUMARESH\Documents>hdfs dfs -cat /output2/part-r-00000

apple 12
banana 12
grape 4
orange 8

```
c:\Users\KUMARESH\Documents>hdfs dfs -cat /output2/part-r-00000
apple   12
banana  12
grape   4
orange  8
```

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

Use Cloudera QuickStart VM (Local Setup)

Download the Cloudera QuickStart VM

Import it into VirtualBox or VMware.

Start the VM and log in (default user/pass: cloudera/cloudera).

Hue is pre-installed and runs at http://localhost:8888.

Load Sample Data

Use built-in sample datasets or upload your own CSV.

Log in to **Hue** at http://localhost:8888.

Go to **File Browser** > Upload a file (e.g., sales_data.csv).

Go to **Data Browsers** > **Metastore Tables** > Create a new table from the uploaded file.

Choose:

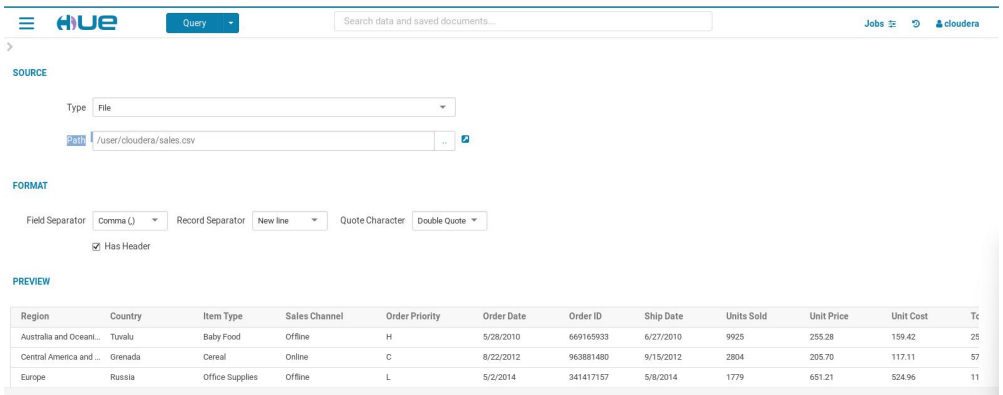
Database: default

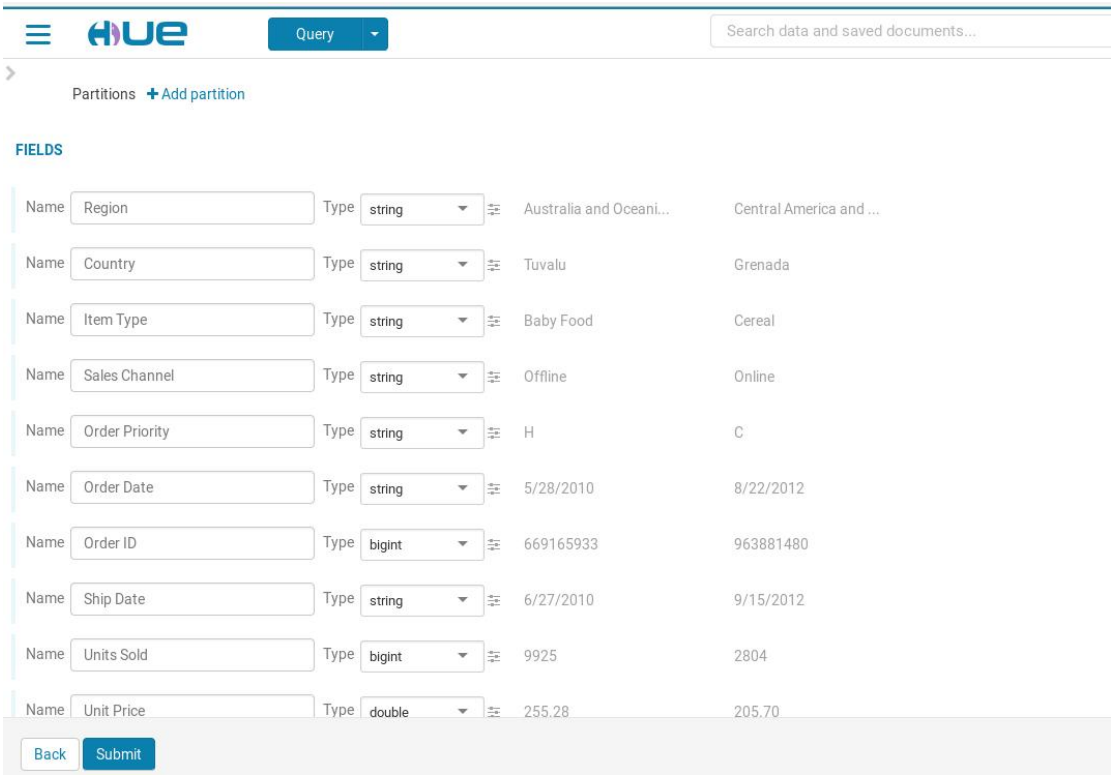
Table name: sales_data

Format: CSV

Delimiter: ,

Column names: Infer from header





Analyze Data Using Hive via Hue

Open **Query Editors** > **Hive** or **Impala**.

Run SQL queries such as:

```
Select region, sum(total_revenue) as total_revenue
From sales
Group by region;
```

Generate Reports or Visualizations in Hue

After running a query, click the **Chart icon** (top right of result panel).

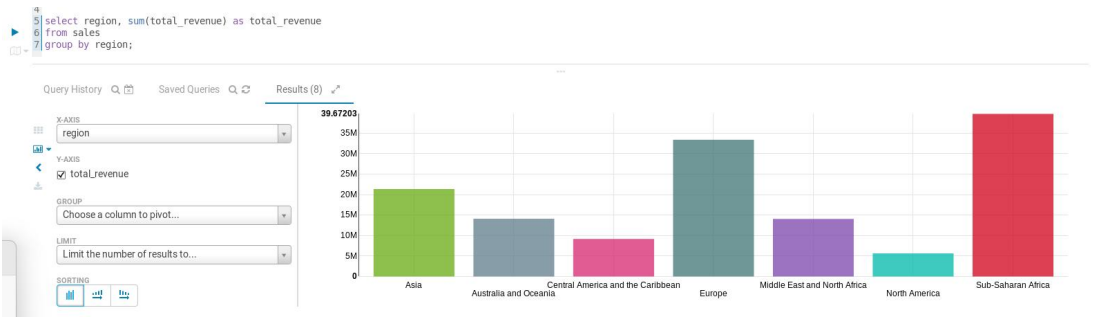
Choose chart type (bar, pie, line, etc.).

Customize:

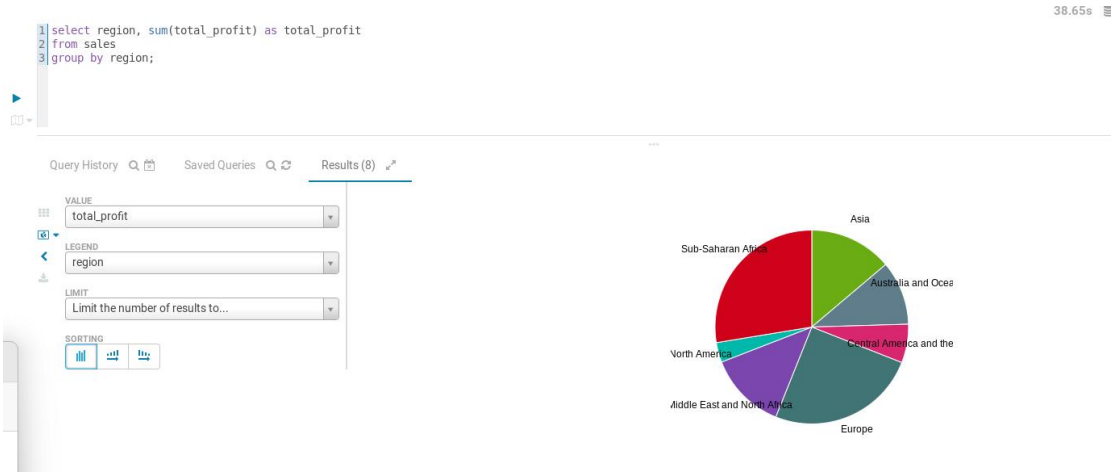
X-axis: category

Y-axis: total_sales

Save or export the chart.



Select region, sum(total_profit) as total_profit
From sales
Group by region;



Select item_type, sum(unit_price) as total_sales
From sales
Group by item_type
Sort by total_sales;