ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ, ಬೆಳಗಾವಿ

VISVESVARAYA TECHNOLOGICAL UNIVERSITY - BELAGAVI

BANGALORE INSTITUTE OF TECHNOLOGY

K.R. ROAD, V.V PURAM, BANGALORE – 560 004

DEPARTMENT OF
INFORMATION SCIENCE AND ENGINEERING

SUBJECT CODE: BIS701

BIG DATA ANALYTICS LAB MANUAL

As per Choice Based Credit System Scheme (CBCS)

FOR VII SEMESTER ISE AS PRESCRIBED BY VTU

Effective from the Academic year 2025-2026

Prepared By:

Dr. H Roopa
Professor
Dept. of ISE, BIT

BANGALORE INSTITUTE OF TECHNOLOGY

VISION

- To establish and develop the Institute as the center of higher learning, ever abreast with expanding horizon of knowledge in the field of Engineering and Technology with entrepreneurial thinking, leadership excellence for life-long success and solve societal problems.

MISSION

- Provide high quality education in the Engineering disciplines from the undergraduate through doctoral levels with creative academic and professional programs.
- Develop the Institute as a leader in Science, Engineering, Technology, Management and Research and apply knowledge for the benefit of society.
- Establish mutual beneficial partnerships with Industry, Alumni, Local, State and Central Governments by Public Service Assistance and Collaborative Research.
- Inculcate personality development through sports, cultural and extracurricular activities and engage in the social, economic and professional challenges

Bangalore Institute of Technology

K R Road, VV Pura, Bangalore- 560004

Department of Information Science and Engineering

VISION:

Empower every student to be innovative, creative and productive in the field of Information Technology by imparting quality technical education, **developing Professional Skills** and inculcating human values..

MISSION:

- To evolve continually as a centre of excellence in offering quality Information Technology **Education**.
- To nurture the students to meet the global competency in industry for **Employment**.
- To promote collaboration with industry and academia for constructive interaction to empower **Entrepreneurship**.

  To provide reliable, contemporary and integrated technology to support and facilitate **Life Long Learning**

PROGRAM EDUCATIONAL OBJECTIVES

- Uplift the students through Information Technology **Education.**
- Provide exposure to emerging technologies and train them to **Employable** in Multi-disciplinary industries.
- Motivate them to become good professional Engineers and **Entrepreneur.**
- Inspire them to prepare for **Higher Learning and Research.**

PROGRAMME SPECIFIC OUTCOMES (PSOs)

- To provide our graduates **with Core Competence in Information Technology and Management.**
- To prepare our graduates with **relevant skills for higher Education, Entrepreneurship, Professional career & Social values.**

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **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 understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **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.
11. **Project management and finance**: Demonstrate knowledge and understanding of the 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.
12. **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.

Prerequisites:

Course Objectives:

This course will enable students to:

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:

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.

RESOURSES REQUIRED:

- Hardware resources
  - Desktop PC
  - Windows operating system
- Software resources
  - Virtual Box
  - cloudera-quickstart-vm-5.12.0-0-virtualbox

Mapping of COs-POs and COs-PSOs

Big Data Analytics Laboratory (BIS701)

Year of Study: 2025 -2026 (ODD)

CO to PO & PSO MAPPING

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1														
CO2														
CO3														
CO4														
CO5														
AVG														

BIG DATA ANALYTICS		Semester	VII
Course Code:	BIS701	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 hours Theory + 8-10 Lab slots	Total Marks	100
Credits	04	Exam Hours	03
Examination nature (SEE)	Theory/practical		

List of Programs	
1.	Install Hadoop and Implement the following file management tasks in Hadoop: Adding files and directories Retrieving files Deleting files and directories. **Hint:** A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.
2.	Develop a MapReduce program to implement Matrix Multiplication
3.	Develop a Map Reduce program that mines weather data and displays appropriate messages indicating the weather conditions of the day.
4.	Develop a MapReduce program to find the tags associated with each movie by analyzing movie lens data.
5.	Implement Functions: Count – Sort – Limit – Skip – Aggregate using MongoDB
6.	Write Pig Latin scripts to sort, group, join, project, and filter the data.
7.	Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.
8.	Implement a word count program in Hadoop and Spark.
9.	Use CDH (Cloudera Distribution for Hadoop) and HUE (Hadoop User Interface) to analyze data and generate reports for sample datasets.
	Practical Sessions need to be assessed by appropriate rubrics and viva-voce method. This will contribute to 25 marks • Daily conduction with record – 15 • Test conduction with viva-voce - 10

RUBRIC SHEET

DAILY CONDUCTION (Max: 15 Marks)

Sl. No	Experiment Name	Write-Up & Implementation 3 Marks	Analysis & Execution 4 Marks	Results & Tabulation 3 Marks	Record 5 Marks	Total 15 Marks
1	Install Hadoop and Implement the following file management tasks in Hadoop: Adding files and directories Retrieving files Deleting files and directories. **Hint:** A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.					
2	Develop a MapReduce program to implement Matrix Multiplication					
3	Develop a Map Reduce program that mines weather data and displays appropriate messages indicating the weather conditions of the day.					
4	Develop a MapReduce program to find the tags associated with each movie by analyzing movie lens data.					
5	Implement Functions: Count – Sort – Limit – Skip – Aggregate using MongoDB					
6	Write Pig Latin scripts to sort, group, join, project, and filter the data.					
7	Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.					
8	Implement a word count program in Hadoop and Spark.					
9	Use CDH (Cloudera Distribution for Hadoop) and HUE (Hadoop User Interface) to analyze data and generate reports for sample datasets					

Test (T1+T2) Rubrics (Max: 10 Marks)

	Write-up & Implementation (10Marks)	**Analysis & Execution** (20Marks)	**Results & Tabulation** (10Marks)	**Viva** (10Marks)	**Total 50 Marks**
Test-1					
Test-2					
	T1+T2 (100 Marks scaled down 10 Marks)			10	
	Daily conduction + Test Marks (15 + 10)			25	

Program 1

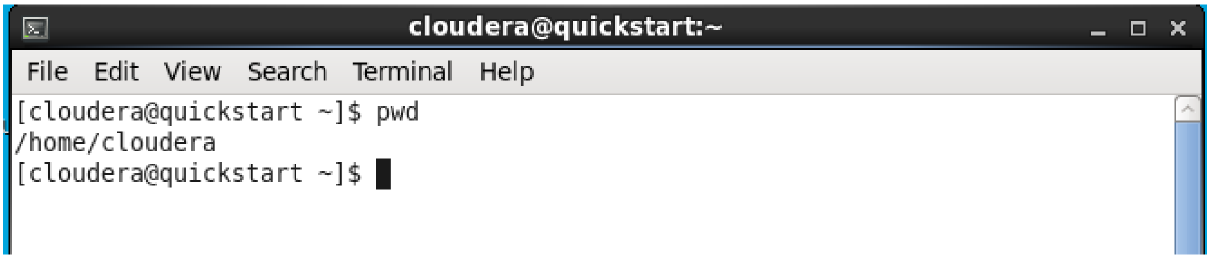
Install Hadoop and Implement the following file management tasks in Hadoop:

- **Adding files and directories**
- **Retrieving files.**
- **Deleting files and directories.**
- **Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.**

Note: Every command for HDFS starts with **hdfs dfs**

Execute the following commands-

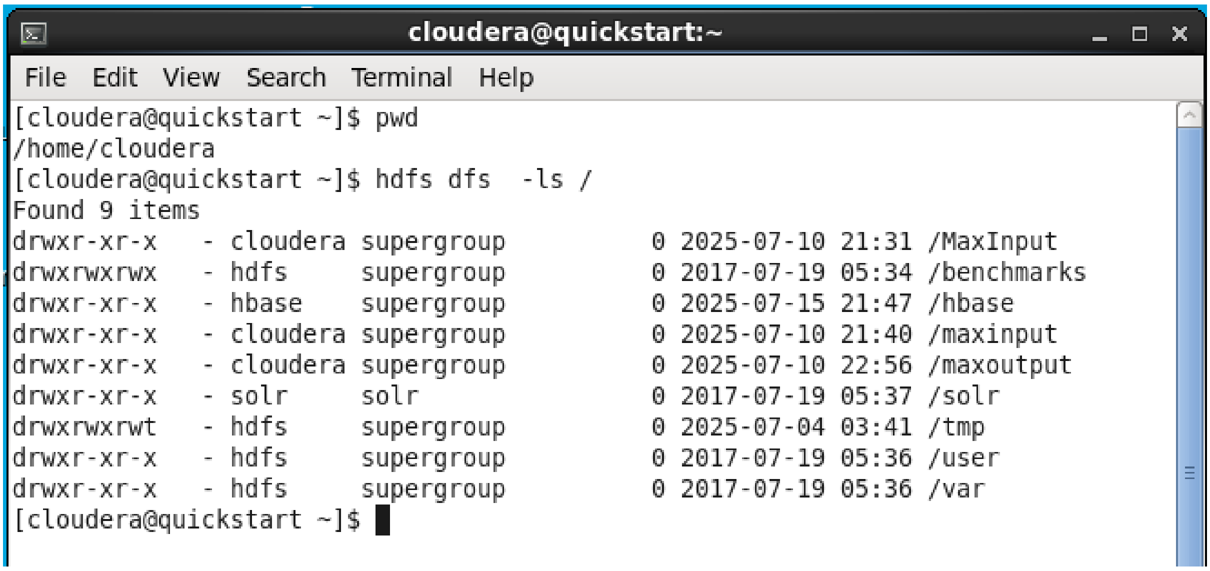
1. **pwd** - to know the present working directory of the local file system.



```
[cloudera@quickstart ~]$ pwd
/home/cloudera
[cloudera@quickstart ~]$ 
```

To explore the HDFS use following command

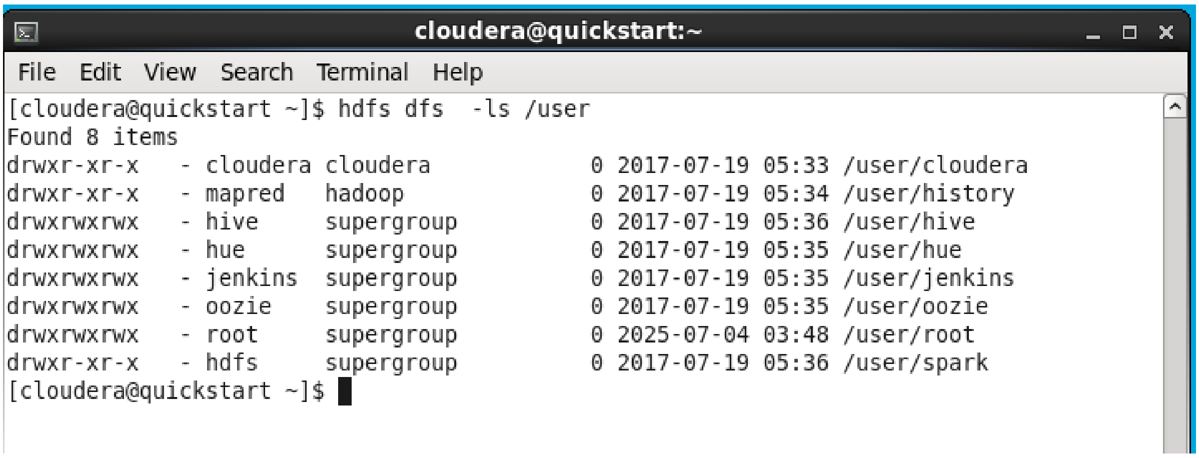
2. **hdfs dfs -ls /**



```
[cloudera@quickstart ~]$ pwd
/home/cloudera
[cloudera@quickstart ~]$ hdfs dfs  -ls /
Found 9 items
drwxr-xr-x   - cloudera supergroup          0 2025-07-10 21:31 /MaxInput
drwxrwxrwx   - hdfs     supergroup          0 2017-07-19 05:34 /benchmarks
drwxr-xr-x   - hbase    supergroup          0 2025-07-15 21:47 /hbase
drwxr-xr-x   - cloudera supergroup          0 2025-07-10 21:40 /maxinput
drwxr-xr-x   - cloudera supergroup          0 2025-07-10 22:56 /maxoutput
drwxr-xr-x   - solr     solr                0 2017-07-19 05:37 /solr
drwxrwxrwt   - hdfs     supergroup          0 2025-07-04 03:41 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2017-07-19 05:36 /user
drwxr-xr-x   - hdfs     supergroup          0 2017-07-19 05:36 /var
[cloudera@quickstart ~]$ 
```

To explore the specific directory of HDFS for example user directory of HDFS

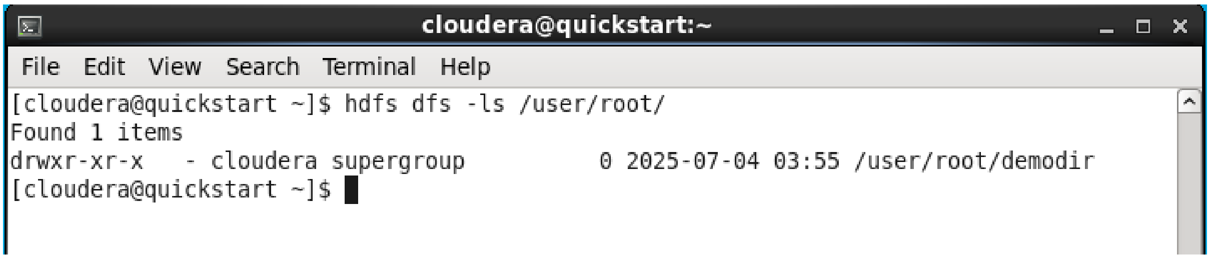
3. **hdfs dfs -ls /user**



```
[cloudera@quickstart ~]$ hdfs dfs  -ls /user
Found 8 items
drwxr-xr-x   - cloudera cloudera            0 2017-07-19 05:33 /user/cloudera
drwxr-xr-x   - mapred   hadoop              0 2017-07-19 05:34 /user/history
drwxrwxrwx   - hive     supergroup          0 2017-07-19 05:36 /user/hive
drwxrwxrwx   - hue      supergroup          0 2017-07-19 05:35 /user/hue
drwxrwxrwx   - jenkins  supergroup          0 2017-07-19 05:35 /user/jenkins
drwxrwxrwx   - oozie    supergroup          0 2017-07-19 05:35 /user/oozie
drwxrwxrwx   - root     supergroup          0 2025-07-04 03:48 /user/root
drwxr-xr-x   - hdfs     supergroup          0 2017-07-19 05:36 /user/spark
[cloudera@quickstart ~]$ 
```

To create the directory in the /user/root and list the /user/root use following command

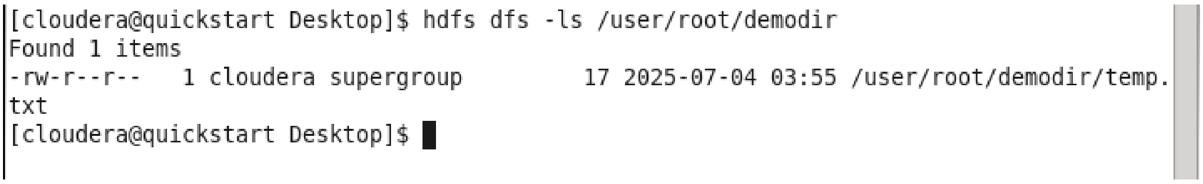
4. **hdfs dfs -mkdir /user/root/demodir**
5. **hdfs dfs -ls /user/root/**



```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/root/
Found 1 items
drwxr-xr-x   - cloudera supergroup          0 2025-07-04 03:55 /user/root/demodir
[cloudera@quickstart ~]$ 
```

To copy the file temp.txt from Local file system location (/user/cloudera/Desktop/temp.txt) use the following command:

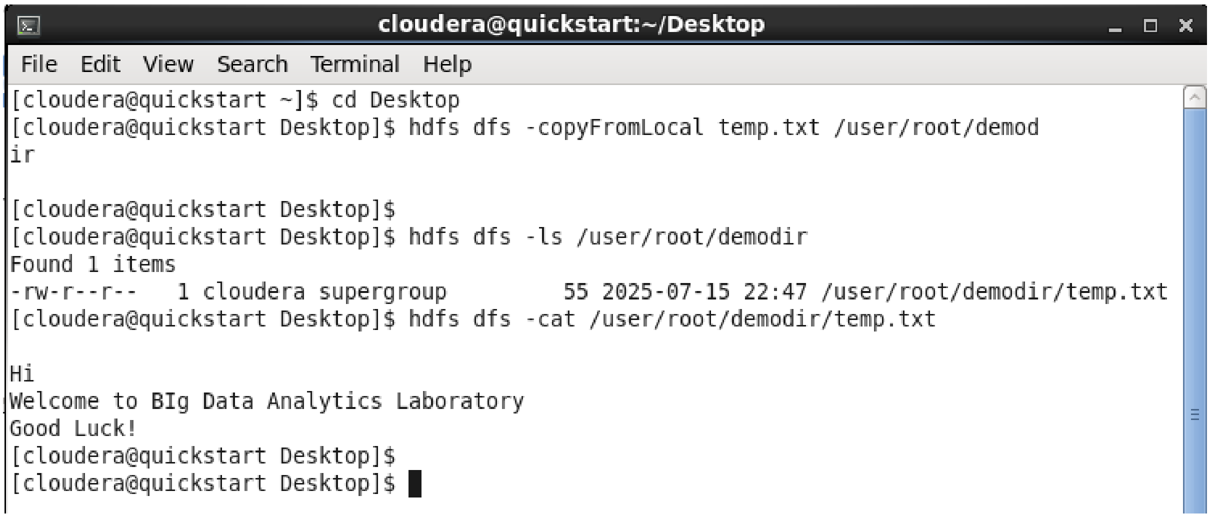
6. **cd Desktop** // since file is on Desktop
7. **hdfs dfs -copyFromLocal temp.txt /user/root/demodir**
8. **hdfs dfs -ls /user/root/demodir**



```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /user/root/demodir
Found 1 items
-rw-r--r--   1 cloudera supergroup         17 2025-07-04 03:55 /user/root/demodir/temp.
txt
[cloudera@quickstart Desktop]$ 
```

To list the content of temp.txt use following command

9. **hdfs dfs -cat /user/root/demodir/temp.txt**



cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ cd Desktop
[cloudera@quickstart Desktop]$ hdfs dfs -copyFromLocal temp.txt /user/root/demod
ir

[cloudera@quickstart Desktop]$
[cloudera@quickstart Desktop]$ hdfs dfs -ls /user/root/demodir
Found 1 items
-rw-r--r--   1 cloudera supergroup         55 2025-07-15 22:47 /user/root/demodir/temp.txt
[cloudera@quickstart Desktop]$ hdfs dfs -cat /user/root/demodir/temp.txt

Hi
Welcome to BIg Data Analytics Laboratory
Good Luck!
[cloudera@quickstart Desktop]$
[cloudera@quickstart Desktop]$ 
```

Please Note: Before working with all these commands, HDFS, NAmeNode, DataNode services must be ON. Open Cloudera Manager and start the services. Command line utility can also be used.

Program 2

Develop a MapReduce program to implement Matrix Multiplication.

MapReduce is a technique in which a huge program is subdivided into small tasks and run parallelly to make computation faster, save time, and mostly used in distributed systems. It has 2 important parts:

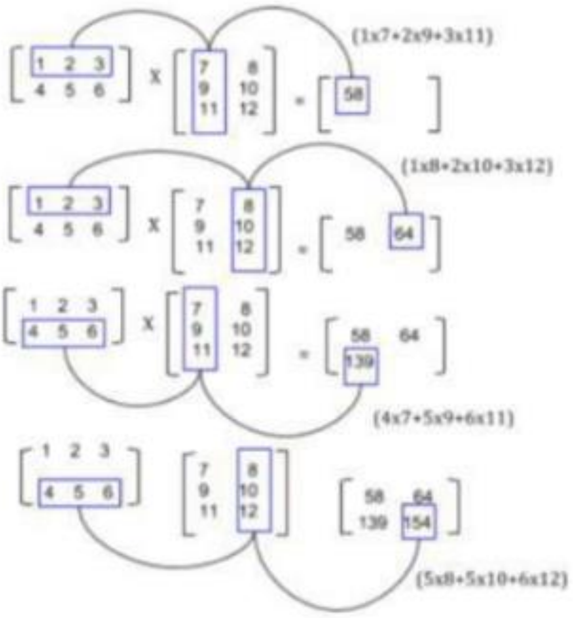
Mapper: It takes raw data input and organizes into key, value pairs. For example, In a dictionary, you search for the word "Data" and its associated meaning is "facts and statistics collected together for reference or analysis". Here the Key is Data and the Value associated with is facts and statistics collected together for reference or analysis.

Reducer: It is responsible for processing data in parallel and produce final output.

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

In Hadoop, Map Reduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

In mathematics, matrix multiplication or the matrix product is a binary operation that produces a matrix from two matrices. In more detail, if A is an n × m matrix and B is an m × p matrix, their matrix product AB is an n × p matrix, in which the m entries across a row of A are multiplied with the m entries down a column of B and summed to produce an entry of AB. When two linear transformations are represented by matrices, then the matrix product represents the composition of the two transformations



ALGORITHM:

Algorithm for Map Function:

for each element mij of M do

produce (key,value) pairs as ((i,k), (M,j,mij), for k=1,2,3,.. upto the number of columns of N

return Set of (key,value) pairs that each key (i,k), has list with values (M,j,mij) and (N, j,njk) for all possible values of j.

Algorithm for Reduce Function:

for each key (i,k) do

sort values begin with M by j in listM

sort values begin with N by j in listN

Program:

```java
import java.io.IOException;
import java.util.HashMap;
import org.apache.hadoop.conf.*;
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class MatMul {
    public static class MatrixMapper extends 
Mapper<LongWritable,Text, Text, Text>
    {
        public void map(LongWritable key, Text value, Context 
context)
                throws IOException, InterruptedException {
            Configuration conf = context.getConfiguration();
            int m = Integer.parseInt(conf.get("m"));
            int p = Integer.parseInt(conf.get("p"));
            String line = value.toString();
            String[] indicesAndValue = line.split(",");

            Text outputKey = new Text(); //object for output key
            Text outputValue = new Text(); // object for output 
key

            if (indicesAndValue[0].equals("M"))
            {
                for (int k = 0; k < p; k++)
                {
                    outputKey.set(indicesAndValue[1] + "," +
k);
                    outputValue.set("M," + indicesAndValue[2]
+ "," + indicesAndValue[3]);
                    context.write(outputKey, outputValue);
                }
            }
            else
            {
                for (int i = 0; i < m; i++)
                {
                    outputKey.set(i + "," +
indicesAndValue[2]);
                    outputValue.set("N," + indicesAndValue[1]
+ "," + indicesAndValue[3]);
                    context.write(outputKey, outputValue);
                }
            }

        }

    }
    public static class MatrixReducer extends Reducer<Text, Text,
Text, Text>
    {
        public void reduce(Text key, Iterable<Text> values,
Context context)
                throws IOException, InterruptedException
                {
            String[] value;

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

            HashMap<Integer, Float> hashB = new HashMap<Integer,
Float>();
            for (Text val : values)
            {
                value = val.toString().split(",");
                if (value[0].equals("M"))

    hashA.put(Integer.parseInt(value[1]),Float.parseFloat(value[2]))
;
                else
                    hashB.put(Integer.parseInt(value[1]),
Float.parseFloat(value[2]));
            }
            int n =
Integer.parseInt(context.getConfiguration().get("n"));
```

```
            float result = 0.0f;
            for (int j = 0; j < n; j++)
            {
                float a_ij = hashA.containsKey(j) ?
hashA.get(j) : 0.0f;
                float b_jk = hashB.containsKey(j) ?
hashB.get(j) : 0.0f;
                result += a_ij * b_jk;
            }
            if (result != 0.0f)
                context.write(null, new Text(key.toString() +
"," + Float.toString(result)));

                }
    }


    public static void main(String[] args) throws Exception
    {
        Configuration conf = new Configuration();   // M is an m-
by-n matrix; N is an n-by-p matrix
        conf.set("m", "2");
        conf.set("n", "2");
        conf.set("p", "2");
        Job job = Job.getInstance(conf, "MatrixMultiplication");
        job.setJarByClass(MatMul.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        job.setMapperClass(MatrixMapper.class);
        job.setReducerClass(MatrixReducer.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true);
    }
}
```

OUTPUT:

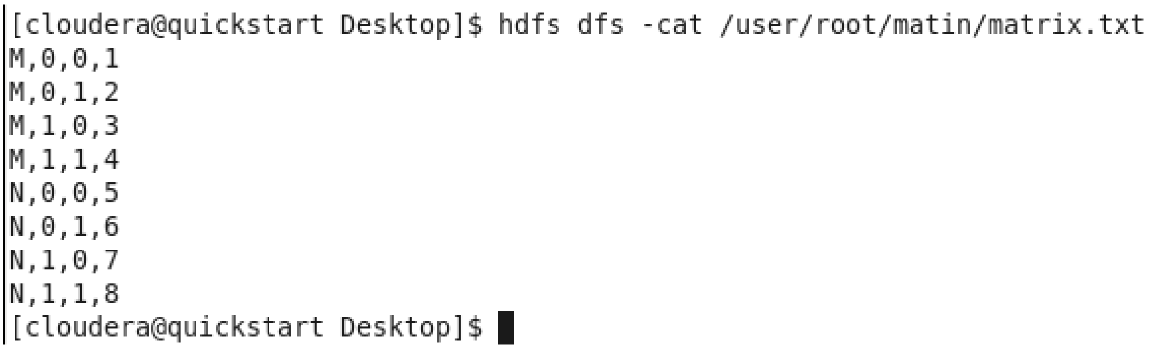
[cloudera@quickstart Desktop]$ hadoop dfs -mkdir /matin

[cloudera@quickstart Desktop]$ hadoop dfs -copyFromLocal matrix.txt /matin

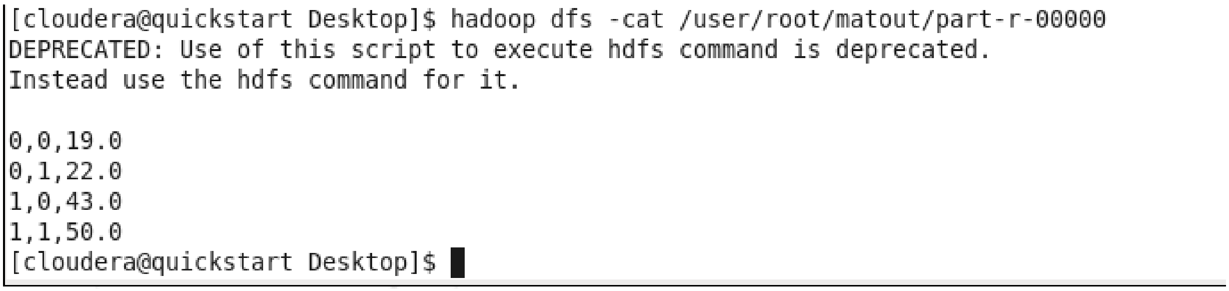
[cloudera@quickstart Desktop]$ hadoop dfs -cat /user/root/matin/matrix.txt

[cloudera@quickstart Desktop]$ hadoop jar MatMul.jar MatMul /user/root/matin /user/root/matout

[cloudera@quickstart Desktop]$ hadoop dfs -cat /user/root/matout/part-r-00000



```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /user/root/matin/matrix.txt
M,0,0,1
M,0,1,2
M,1,0,3
M,1,1,4
N,0,0,5
N,0,1,6
N,1,0,7
N,1,1,8
[cloudera@quickstart Desktop]$ 
```



```
[cloudera@quickstart Desktop]$ hadoop dfs -cat /user/root/matout/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

0,0,19.0
0,1,22.0
1,0,43.0
1,1,50.0
[cloudera@quickstart Desktop]$ 
```

Program 3

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

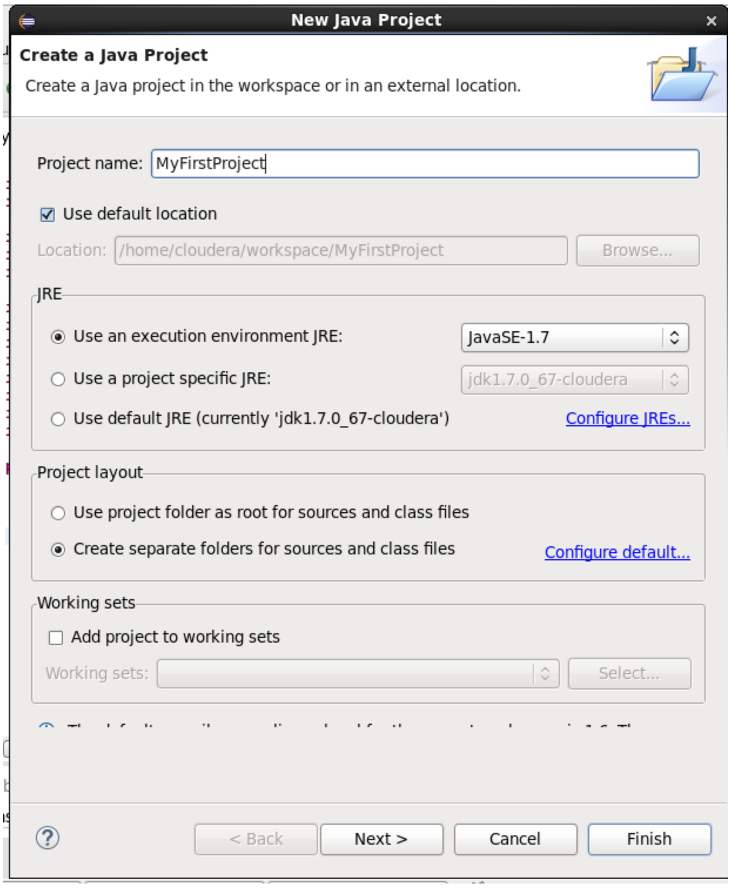
```
import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;

public class MyMaxMin {

     //Mapper

     public static class MaxTemperatureMapper extends
                Mapper<LongWritable, Text, Text, Text> {

           @Override
           public void map(LongWritable arg0, Text Value, Context
context)
                      throws IOException, InterruptedException {

           //Converting the record (single line) to String and
storing it in a String variable line

           String line = Value.toString();

           //Checking if the line is not empty

                if (!(line.length() == 0))
                {

               String date = line.substring(6, 14);          //date

                float temp_Max = Float.parseFloat(line.substring(39,
45).trim());  //maximum temperature

                float temp_Min = Float.parseFloat(line.substring(47,
53).trim());  //minimum temperature
```

```
    //if maximum temperature is greater than 35 , its a hot day

        if (temp_Max > 35.0) {

            context.write(new Text("Hot Day " + date),new
Text(String.valueOf(temp_Max)));        // Hot day

                        }

        //if minimum temperature is less than 10 , its a cold day

        if (temp_Min < 10) {

    context.write(new Text("Cold Day " + date),new
Text(String.valueOf(temp_Min)));        // Cold day
                        }
            }
        }
    }

//Reducer

    public static class MaxTemperatureReducer extends
            Reducer<Text, Text, Text, Text> {

        public void reduce(Text Key, Iterator<Text> Values,
Context context)
                throws IOException, InterruptedException {

            //putting all the values in temperature variable of
type String

            String temperature = Values.next().toString();
            context.write(Key, new Text(temperature));
        }
    }

    public static void main(String[] args) throws Exception {

//reads the default configuration of cluster from the configuration
xml files
        Configuration conf = new Configuration();

//Initializing the job with the default configuration of the cluster

        Job job = new Job(conf, "weather example");

        job.setJarByClass(MyMaxMin.class);  //Assigning the driver
class name
```

```
           job.setMapOutputKeyClass(Text.class); //Key type coming
out of mapper

           job.setMapOutputValueClass(Text.class); //value type
coming out of mapper

           job.setMapperClass(MaxTemperatureMapper.class);
      //Defining the mapper class name

           job.setReducerClass(MaxTemperatureReducer.class);
      //Defining the reducer class name

           //Defining input Format class which is responsible to
parse the dataset into a key value pair
           job.setInputFormatClass(TextInputFormat.class);

           //Defining output Format class which is responsible to
parse the dataset into a key value pair
           job.setOutputFormatClass(TextOutputFormat.class);

      //setting the second argument as a path in a path variable
           Path OutputPath = new Path(args[1]);

           //Configuring the input path from the filesystem into the
job
           FileInputFormat.addInputPath(job, new Path(args[0]));

           //Configuring the output path from the filesystem into the
job
           FileOutputFormat.setOutputPath(job, new Path(args[1]));

           //deleting the context path automatically from hdfs so
that we don't have delete it explicitly
           OutputPath.getFileSystem(conf).delete(OutputPath);

           //exiting the job only if the flag value becomes false
           System.exit(job.waitForCompletion(true) ? 0 : 1);

      }
}
```

OUTPUT:

Procedure for the Execution:



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: MyFirstProject

☑ Use default location

Location: /home/cloudera/workspace/MyFirstProject

JRE

● Use an execution environment JRE: JavaSE-1.7

○ Use a project specific JRE: jdk1.7.0_67-cloudera

○ Use default JRE (currently 'jdk1.7.0_67-cloudera')

Project layout

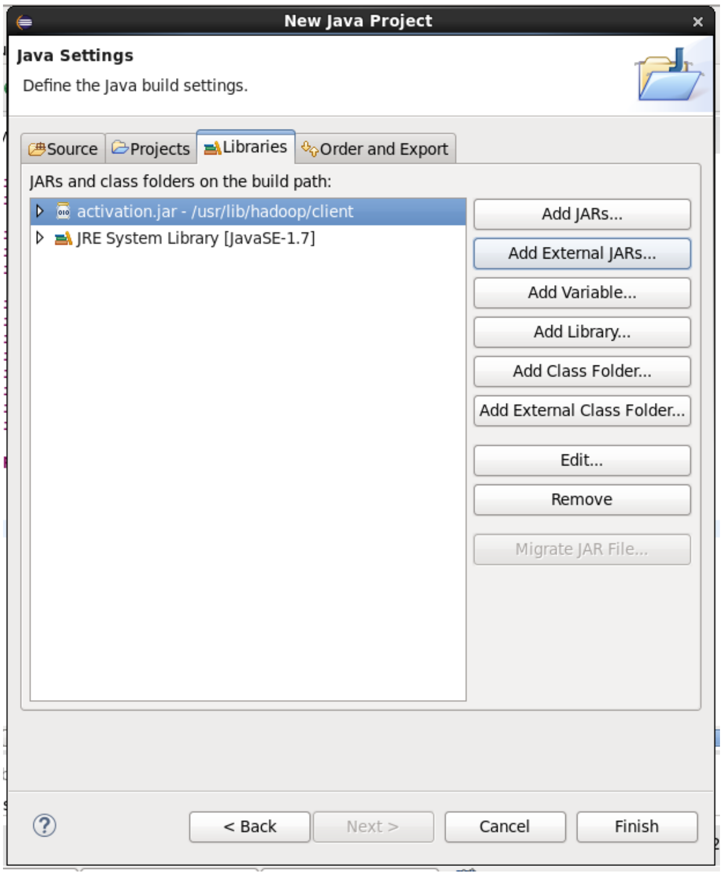
○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

Working sets

□ Add project to working sets

Working sets:



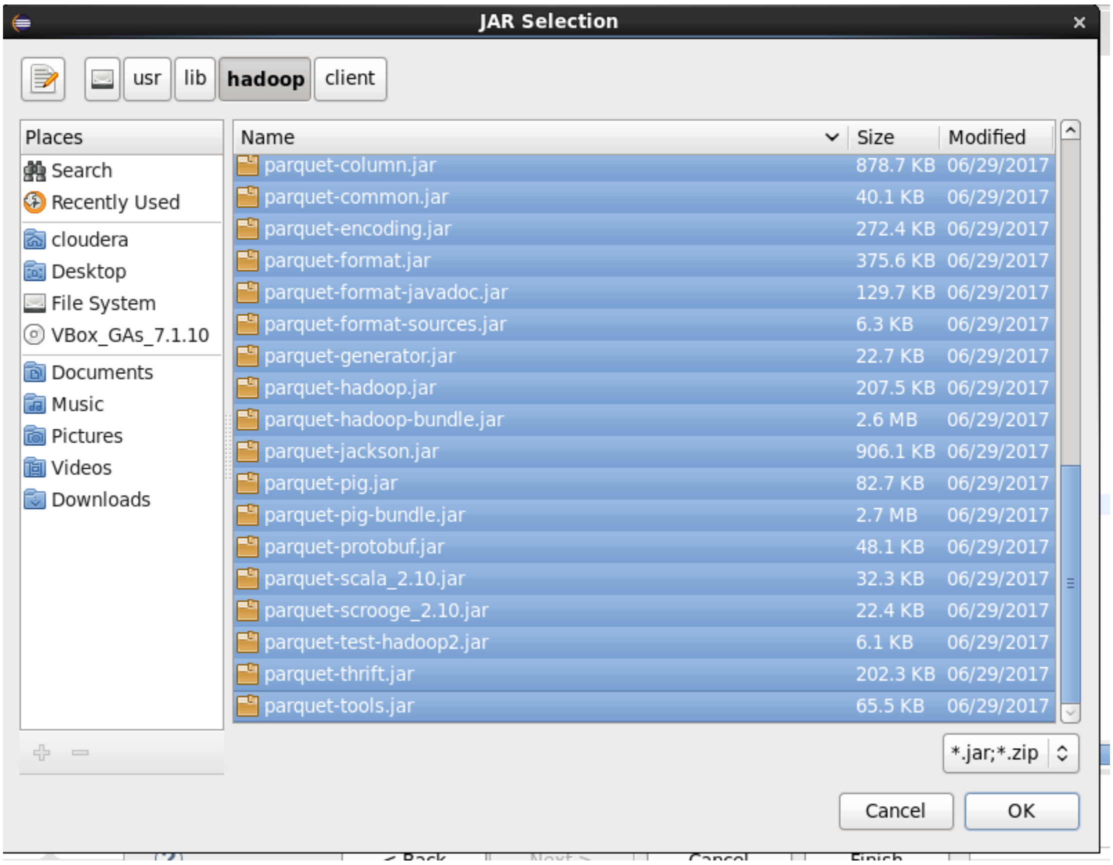
New Java Project

Java Settings

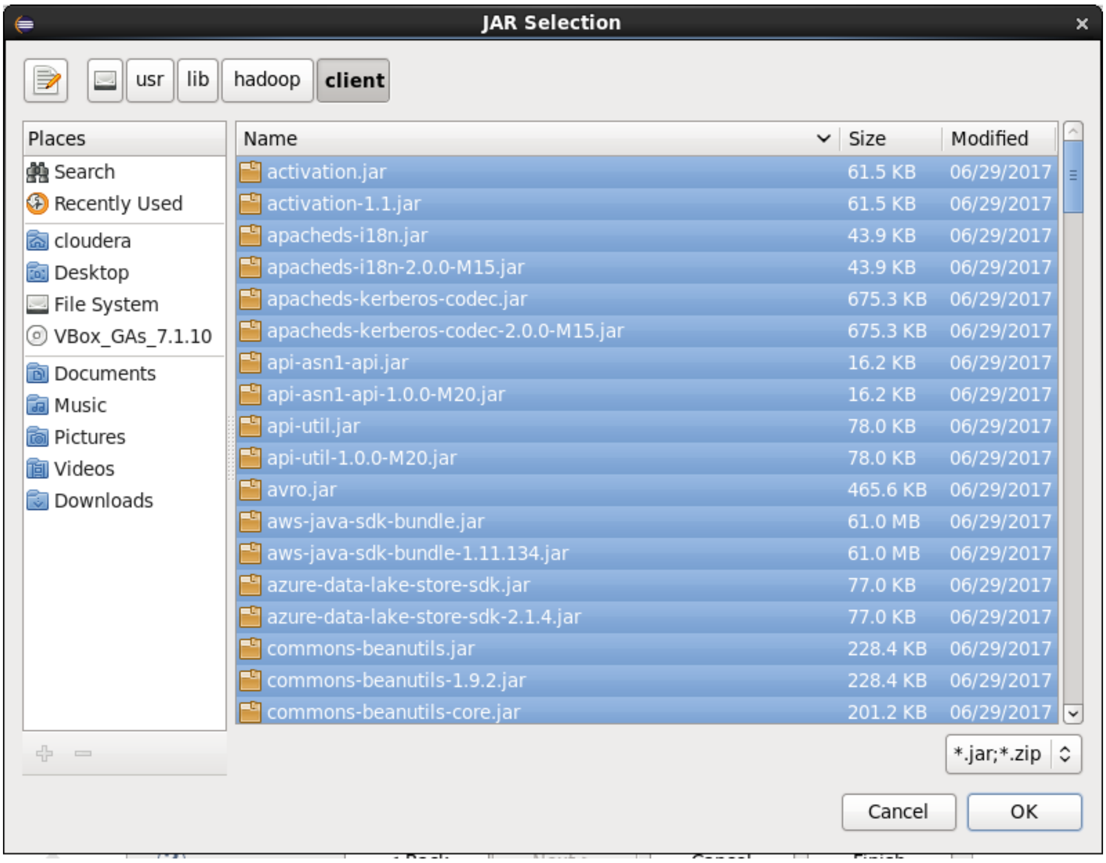
Define the Java build settings.

JARs and class folders on the build path:

- activation.jar - /usr/lib/hadoop/client
- JRE System Library [JavaSE-1.7]



Name	Size	Modified
parquet-column.jar	878.7 KB	06/29/2017
parquet-common.jar	40.1 KB	06/29/2017
parquet-encoding.jar	272.4 KB	06/29/2017
parquet-format.jar	375.6 KB	06/29/2017
parquet-format-javadoc.jar	129.7 KB	06/29/2017
parquet-format-sources.jar	6.3 KB	06/29/2017
parquet-generator.jar	22.7 KB	06/29/2017
parquet-hadoop.jar	207.5 KB	06/29/2017
parquet-hadoop-bundle.jar	2.6 MB	06/29/2017
parquet-jackson.jar	906.1 KB	06/29/2017
parquet-pig.jar	82.7 KB	06/29/2017
parquet-pig-bundle.jar	2.7 MB	06/29/2017
parquet-protobuf.jar	48.1 KB	06/29/2017
parquet-scala_2.10.jar	32.3 KB	06/29/2017
parquet-scrooge_2.10.jar	22.4 KB	06/29/2017
parquet-test-hadoop2.jar	6.1 KB	06/29/2017
parquet-thrift.jar	202.3 KB	06/29/2017
parquet-tools.jar	65.5 KB	06/29/2017



Name	Size	Modified
activation.jar	61.5 KB	06/29/2017
activation-1.1.jar	61.5 KB	06/29/2017
apacheds-i18n.jar	43.9 KB	06/29/2017
apacheds-i18n-2.0.0-M15.jar	43.9 KB	06/29/2017
apacheds-kerberos-codec.jar	675.3 KB	06/29/2017
apacheds-kerberos-codec-2.0.0-M15.jar	675.3 KB	06/29/2017
api-asn1-api.jar	16.2 KB	06/29/2017
api-asn1-api-1.0.0-M20.jar	16.2 KB	06/29/2017
api-util.jar	78.0 KB	06/29/2017
api-util-1.0.0-M20.jar	78.0 KB	06/29/2017
avro.jar	465.6 KB	06/29/2017
aws-java-sdk-bundle.jar	61.0 MB	06/29/2017
aws-java-sdk-bundle-1.11.134.jar	61.0 MB	06/29/2017
azure-data-lake-store-sdk.jar	77.0 KB	06/29/2017
azure-data-lake-store-sdk-2.1.4.jar	77.0 KB	06/29/2017
commons-beanutils.jar	228.4 KB	06/29/2017
commons-beanutils-1.9.2.jar	228.4 KB	06/29/2017
commons-beanutils-core.jar	201.2 KB	06/29/2017

Right click on your project -> Export -> Java - > JAR file -> Select your file -> select export destination ->Desktop -> Finish

```
[cloudera@quickstart ~]$ hdfs dfs –mkdir /maxinput
[cloudera@quickstart ~]$ cd Desktop
[cloudera@quickstart Desktop]$ hdfs dfs -copyFromLocal datafileofweather.txt  /maxinput
[cloudera@quickstart Desktop]$ hdfs jar MaxTemp.jar MyMaxMin /maxinput /maxoutput
[cloudera@quickstart Desktop]$ hdfs dfs -cat /maxoutput/part-r-00000
```

```
DEPRECATED: Use of this script to execute hdfs command is deprecated.

Instead use the hdfs command for it.

Cold Day 20150101     -21.8
Cold Day 20150102     -24.9
Cold Day 20150103     -28.2
Cold Day 20150104     -28.9
Cold Day 20150105     -29.3
Cold Day 20150106     -26.3
Cold Day 20150107     -28.7
Cold Day 20150108     -24.1
Cold Day 20150109     -20.3
Cold Day 20150110     -25.8
Cold Day 20150111     -28.2
Cold Day 20150112     -29.1
Cold Day 20150113     -29.9
Cold Day 20150114     -29.0
Cold Day 20150115     -24.2
Cold Day 20150116     -24.6
Cold Day 20150117     -23.2
Cold Day 20150118     -23.0
Cold Day 20150119     -30.4
Cold Day 20150120     -24.7
Cold Day 20150121     -24.1
Cold Day 20150122     -27.5
Cold Day 20150123     -29.3
Cold Day 20150124     -30.3
Cold Day 20150125     -30.0
Cold Day 20150126     -30.7
Cold Day 20150127     -26.9
Cold Day 20150128     -36.2
Cold Day 20150129     -35.0
Cold Day 20150130     -24.2
Cold Day 20150131     -26.5
Cold Day 20150201     -22.1
Cold Day 20150202     -19.0
Cold Day 20150203     -31.1
Cold Day 20150204     -38.2
Cold Day 20150205     -37.0
Cold Day 20150206     -29.7
Cold Day 20150207     -27.9
Cold Day 20150208     -35.4
Cold Day 20150209     -36.8
Cold Day 20150210     -34.5
Cold Day 20150211     -34.1
Cold Day 20150212     -32.5
Cold Day 20150213     -34.0
Cold Day 20150214     -34.6
Cold Day 20150215     -32.8
Cold Day 20150216     -25.7
Cold Day 20150217     -22.1
Cold Day 20150218     -18.5
Cold Day 20150219     -19.2
Cold Day 20150220     -23.2
Cold Day 20150221     -19.3
Cold Day 20150222     -19.0
Cold Day 20150223     -20.0
Cold Day 20150224     -18.1
Cold Day 20150225     -15.6
Cold Day 20150226     -6.5
Cold Day 20150227     -16.3
Cold Day 20150228     -26.5
Cold Day 20150301     -28.5
Cold Day 20150302     -23.6
Cold Day 20150303     -16.4
Cold Day 20150304     -25.4
Cold Day 20150305     -26.9
Cold Day 20150306     -27.6
Cold Day 20150307     -25.0
Cold Day 20150308     -29.6
Cold Day 20150309     -31.5
Cold Day 20150310     -36.5
Cold Day 20150311     -38.6
Cold Day 20150312     -39.3
Cold Day 20150313     -39.3
Cold Day 20150314     -38.0
Cold Day 20150315     -32.7
Cold Day 20150316     -33.1
Cold Day 20150317     -23.7
Cold Day 20150318     -23.9
Cold Day 20150319     -24.2
Cold Day 20150320     -24.3
Cold Day 20150321     -20.2
Cold Day 20150322     -23.0
Cold Day 20150323     -27.0
Cold Day 20150324     -33.0
Cold Day 20150325     -27.4
Cold Day 20150326     -26.5
Cold Day 20150327     -25.0
Cold Day 20150328     -26.3
Cold Day 20150329     -25.0
Cold Day 20150330     -23.3
Cold Day 20150331     -19.3
Cold Day 20150401     -23.7
Cold Day 20150402     -26.1
Cold Day 20150403     -21.6
Cold Day 20150404     -23.7
Cold Day 20150405     -24.2
Cold Day 20150406     -20.1
Cold Day 20150407     -18.4
Cold Day 20150408     -16.4
Cold Day 20150409     -17.6
Cold Day 20150410     -20.0
Cold Day 20150411     -22.1
Cold Day 20150412     -23.1
Cold Day 20150413     -27.7
Cold Day 20150414     -26.1
Cold Day 20150415     -27.4
Cold Day 20150416     -24.2
Cold Day 20150417     -28.1
Cold Day 20150418     -25.0
Cold Day 20150419     -15.6
Cold Day 20150420     -11.8
Cold Day 20150421     -13.8
Cold Day 20150422     -8.4
Cold Day 20150423     -8.8
Cold Day 20150424     -12.4
Cold Day 20150425     -9.0
Cold Day 20150426     -8.6
Cold Day 20150427     -9.4
Cold Day 20150428     -9.5
Cold Day 20150429     -9.4
Cold Day 20150430     -13.7
Cold Day 20150501     -16.1
Cold Day 20150502     -8.6
Cold Day 20150503     -9.6
Cold Day 20150504     -8.6
Cold Day 20150505     -13.0
Cold Day 20150506     -12.3
Cold Day 20150507     -10.8
Cold Day 20150508     -9.6
Cold Day 20150509     -4.7
Cold Day 20150510     -6.1
Cold Day 20150511     -2.1
Cold Day 20150512     -3.8
Cold Day 20150513     -4.6
Cold Day 20150514     -6.7
Cold Day 20150515     -5.1
Cold Day 20150516     -2.9
Cold Day 20150517     -0.3
Cold Day 20150518     0.7
Cold Day 20150519     -1.4
Cold Day 20150520     -2.6
Cold Day 20150521     0.0
Cold Day 20150522     0.2
Cold Day 20150523     -0.4
Cold Day 20150524     -0.5
Cold Day 20150525     -0.3
Cold Day 20150526     -1.8
Cold Day 20150527     -2.4
Cold Day 20150528     -2.1
Cold Day 20150529     0.2
Cold Day 20150530     -2.3
Cold Day 20150531     -3.8
Cold Day 20150601     -4.0
Cold Day 20150602     -2.4
Cold Day 20150603     -3.2
Cold Day 20150604     -3.7
Cold Day 20150605     -3.3
Cold Day 20150606     -1.0
Cold Day 20150607     -1.7
Cold Day 20150608     1.0
Cold Day 20150609     0.6
Cold Day 20150610     -1.5
Cold Day 20150611     -2.9
Cold Day 20150612     -2.4
Cold Day 20150613     3.3
Cold Day 20150614     3.3
Cold Day 20150615     0.8
Cold Day 20150616     1.4
Cold Day 20150617     1.0
Cold Day 20150618     0.6
Cold Day 20150619     0.6
Cold Day 20150620     7.7
Cold Day 20150621     7.1
Cold Day 20150622     0.1
Cold Day 20150623     1.3
Cold Day 20150624     1.3
Cold Day 20150625     1.6
Cold Day 20150626     2.6
Cold Day 20150627     1.4
Cold Day 20150628     1.9
Cold Day 20150629     6.5
Cold Day 20150630     6.7
Cold Day 20150701     4.7
Cold Day 20150702     3.7
Cold Day 20150703     1.3
Cold Day 20150704     3.9
Cold Day 20150705     8.5
Cold Day 20150706     8.1
Cold Day 20150707     0.8
Cold Day 20150708     0.3
Cold Day 20150709     0.2
Cold Day 20150710     0.4
Cold Day 20150711     0.4
Cold Day 20150712     0.6
Cold Day 20150713     0.9
Cold Day 20150714     2.6
Cold Day 20150715     1.1
Cold Day 20150716     2.2
Cold Day 20150717     1.4
Cold Day 20150718     1.4
Cold Day 20150719     2.0
Cold Day 20150721     3.8
Cold Day 20150722     0.5
Cold Day 20150723     0.0
Cold Day 20150724     -0.1
Cold Day 20150725     1.0
Cold Day 20150726     0.9
Cold Day 20150727     0.9
Cold Day 20150728     -0.2
Cold Day 20150729     0.3
Cold Day 20150730     -0.7
Cold Day 20150731     3.8
Cold Day 20150801     0.8
Cold Day 20150802     0.3
Cold Day 20150803     3.5
Cold Day 20150804     2.3
Cold Day 20150805     1.9
Cold Day 20150806     0.7
Cold Day 20150807     4.2
Cold Day 20150808     3.6
Cold Day 20150809     4.2
Cold Day 20150810     1.9
Cold Day 20150811     0.2
Cold Day 20150812     0.4
Cold Day 20150813     3.1
Cold Day 20150814     4.3
Cold Day 20150815     1.2
Cold Day 20150816     0.6
Cold Day 20150817     0.5
Cold Day 20150818     0.4
Cold Day 20150819     -2.3
Cold Day 20150820     -3.2
Cold Day 20150821     0.8
Cold Day 20150822     1.4
Cold Day 20150823     0.6
Cold Day 20150824     0.6
Cold Day 20150825     1.3
Cold Day 20150826     2.4
Cold Day 20150827     1.9
Cold Day 20150828     0.5
Cold Day 20150829     0.0
Cold Day 20150830     -0.2
Cold Day 20150831     -0.1
Cold Day 20150901     1.4
Cold Day 20150902     2.6
Cold Day 20150903     1.3
Cold Day 20150904     0.0
Cold Day 20150905     -0.3
Cold Day 20150906     2.7
Cold Day 20150907     0.5
Cold Day 20150908     -0.2
Cold Day 20150909     -1.4
Cold Day 20150910     -1.5
Cold Day 20150911     -1.2
Cold Day 20150912     -2.0
Cold Day 20150913     -2.4
Cold Day 20150914     0.1
Cold Day 20150915     -1.1
Cold Day 20150916     -2.0
Cold Day 20150917     -3.0
Cold Day 20150918     -3.2
Cold Day 20150919     -3.4
Cold Day 20150920     -3.6
Cold Day 20150921     -4.9
Cold Day 20150922     -5.6
Cold Day 20150923     -6.1
Cold Day 20150924     -5.1
Cold Day 20150925     -1.5
Cold Day 20150926     -2.1
Cold Day 20150927     -3.9
Cold Day 20150928     -5.6
Cold Day 20150929     -3.0
Cold Day 20150930     -4.4
Cold Day 20151001     -6.9
Cold Day 20151002     -8.1
Cold Day 20151003     -3.8
Cold Day 20151004     -4.6
Cold Day 20151005     -2.9
Cold Day 20151006     -4.1
Cold Day 20151007     -4.6
Cold Day 20151008     -4.7
Cold Day 20151009     -4.6
Cold Day 20151010     -4.6
Cold Day 20151011     -3.7
Cold Day 20151012     -5.0
Cold Day 20151013     -9.4
Cold Day 20151014     -6.7
Cold Day 20151015     -9.6
Cold Day 20151016     -14.2
Cold Day 20151017     -4.9
Cold Day 20151018     -5.5
Cold Day 20151019     -6.9
Cold Day 20151020     -9.1
Cold Day 20151021     -9.5
Cold Day 20151022     -8.3
Cold Day 20151023     -6.4
Cold Day 20151024     -4.4
Cold Day 20151025     -14.5
Cold Day 20151026     -9.2
Cold Day 20151027     -9.7
Cold Day 20151028     -17.0
Cold Day 20151029     -18.3
Cold Day 20151030     -15.5
Cold Day 20151031     -15.3
Cold Day 20151101     -16.5
Cold Day 20151102     -10.0
Cold Day 20151103     -8.5
Cold Day 20151104     -7.4
Cold Day 20151105     -8.7
Cold Day 20151106     -10.3
Cold Day 20151107     -12.1
Cold Day 20151108     -14.1
Cold Day 20151109     -7.6
Cold Day 20151110     -9.6
Cold Day 20151111     -13.9
Cold Day 20151112     -22.3
Cold Day 20151113     -24.7
Cold Day 20151114     -25.1
Cold Day 20151115     -26.7
Cold Day 20151116     -28.4
Cold Day 20151117     -23.6
Cold Day 20151118     -29.8
Cold Day 20151119     -30.8
Cold Day 20151120     -21.7
Cold Day 20151121     -23.2
Cold Day 20151122     -28.3
Cold Day 20151123     -26.8
Cold Day 20151124     -16.9
Cold Day 20151125     -14.9
Cold Day 20151126     -19.8
Cold Day 20151127     -24.1
Cold Day 20151128     -17.9
Cold Day 20151129     -18.8
Cold Day 20151130     -25.7
Cold Day 20151201     -25.5
Cold Day 20151202     -25.6
Cold Day 20151203     -27.4
Cold Day 20151204     -27.4
Cold Day 20151205     -28.2
Cold Day 20151206     -28.9
Cold Day 20151207     -28.4
Cold Day 20151208     -28.3
Cold Day 20151209     -28.3
Cold Day 20151210     -27.2
Cold Day 20151211     -29.2
Cold Day 20151212     -26.0
Cold Day 20151213     -22.3
Cold Day 20151214     -23.2
Cold Day 20151215     -20.8
Cold Day 20151216     -19.4
Cold Day 20151217     -22.9
Cold Day 20151218     -25.7
Cold Day 20151219     -22.3
Cold Day 20151220     -22.5
Cold Day 20151221     -27.5
Cold Day 20151222     -30.0
Cold Day 20151223     -30.3
Cold Day 20151224     -30.8
Cold Day 20151225     -37.5
Cold Day 20151226     -29.6
Cold Day 20151227     -26.0
Cold Day 20151228     -27.1
Cold Day 20151229     -33.0
Cold Day 20151230     -25.7
Cold Day 20151231     -22.7
Hot Day 20150720     9999.0
```

Program 4

Develop a MapReduce program to find the tags associated with each movie by analyzing movie lens data.

```
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.*;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

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

            String line = value.toString();
            if (key.get() == 0 && line.contains("movieId"))
return;    // skip header
            String[] fields = line.split(",", 3); //
movieId,title,genres
            if (fields.length >= 2) {
                String movieId = fields[0].trim();
                String title = fields[1].trim();
                context.write(new Text(movieId), new
Text("MOVIE::" + title));
            }
        }
    }

    public static class TagMapper extends Mapper<LongWritable, Text,
Text, Text> {
        public void map(LongWritable key, Text value, Context
context)
                throws IOException, InterruptedException    {
            String line = value.toString(); // Read the line from
the tags.txt
            if (key.get() == 0 && line.contains("userId"))
return; // skip header
            String[] fields = line.split(",", 4); //
userId,movieId,tag,timestamp
            if (fields.length >= 3) {
                String movieId = fields[1].trim();
                String tag = fields[2].trim();
                context.write(new Text(movieId), new
Text("TAG::" + tag));
            }
        }
    }
    public static class JoinReducer extends Reducer<Text, Text,
Text, Text> {
        public void reduce(Text key, Iterable<Text> values,
Context context)
                throws IOException, InterruptedException {
            String movieTitle = null;
            List<String> tags = new ArrayList<>();
            for (Text val : values)     {
                String value = val.toString();
                if (value.startsWith("MOVIE::"))
                    movieTitle = value.substring(7);
                else if (value.startsWith("TAG::"))
                    tags.add(value.substring(5));
            }
            if (movieTitle != null && !tags.isEmpty()) {
                context.write(new Text(movieTitle), new
Text("," + tags));
            }
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
```

```
           Job job = Job.getInstance(conf, "Movie Tags Join");
           job.setJarByClass(MovieTagsJoin.class);
           // Set Mappers
           MultipleInputs.addInputPath(job, new Path(args[0]),
TextInputFormat.class, MovieMapper.class);
           MultipleInputs.addInputPath(job, new Path(args[1]),
TextInputFormat.class, TagMapper.class);
           job.setReducerClass(JoinReducer.class);
           // Output Types
           job.setOutputKeyClass(Text.class);
           job.setOutputValueClass(Text.class);
           FileOutputFormat.setOutputPath(job, new Path(args[2]));
           job.waitForCompletion(true);
      }
}
```

OUTPUT:

[cloudera@quickstart ~]$ hdfs dfs -mkdir /movie

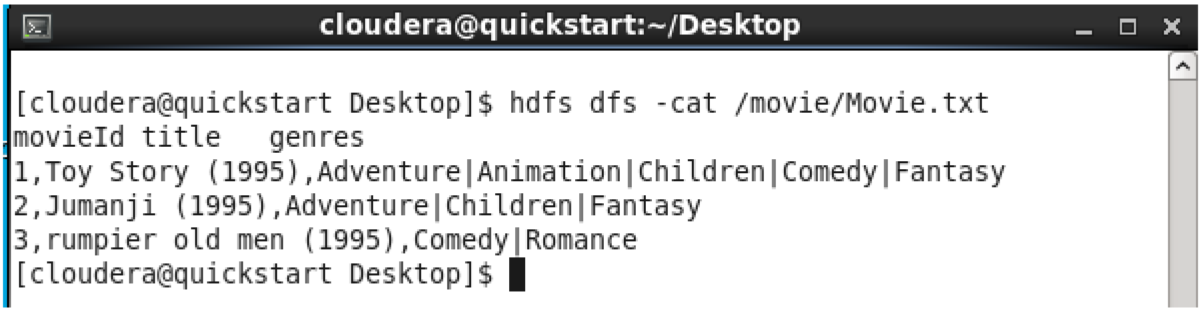
[cloudera@quickstart ~]$ hdfs dfs -mkdir /tag

[cloudera@quickstart ~]$ cd Desktop

[cloudera@quickstart Desktop]$ hdfs dfs -copyFromLocal Movie.txt /movie

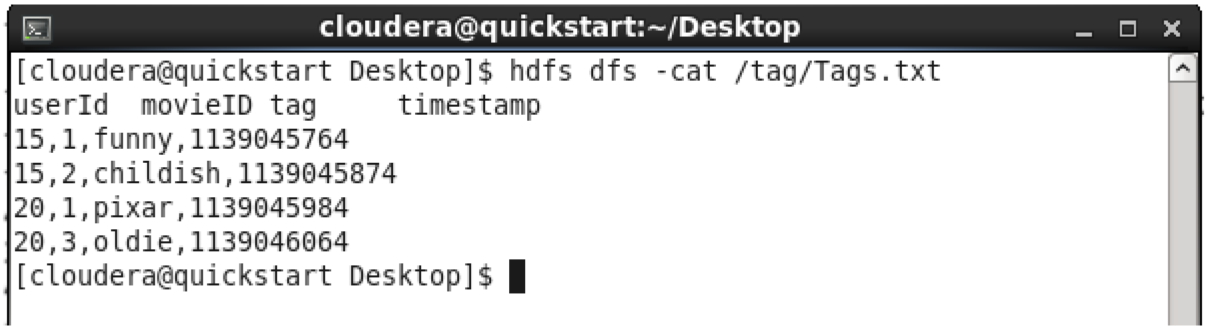
[cloudera@quickstart Desktop]$ hdfs dfs -copyFromLocal Tags.txt /tag

[cloudera@quickstart Desktop]$ hdfs dfs -cat /movie/Movie.txt



```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /movie/Movie.txt
movieId title   genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,rumpier old men (1995),Comedy|Romance
[cloudera@quickstart Desktop]$
```

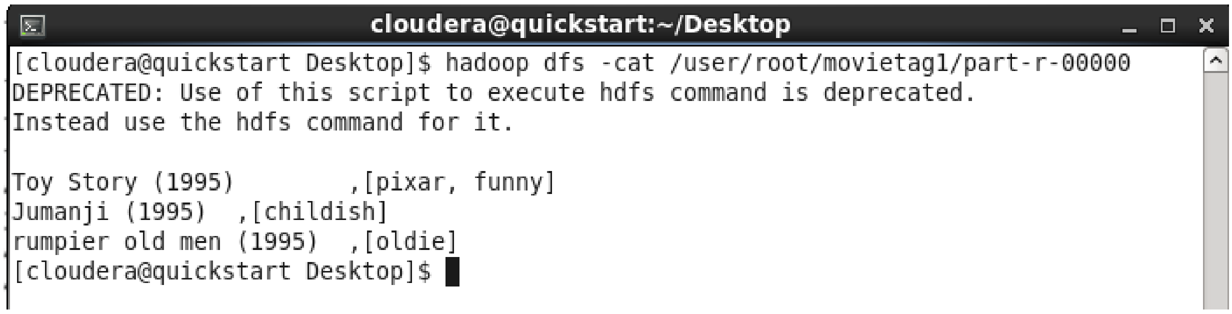
[cloudera@quickstart Desktop]$ hdfs dfs -cat /tag/Tags.txt



```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /tag/Tags.txt
userId  movieID tag     timestamp
15,1,funny,1139045764
15,2,childish,1139045874
20,1,pixar,1139045984
20,3,oldie,1139046064
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart Desktop]$ hadoop jar MovieTagsJoin.jar MovieTagsJoin /movie /tag /user/root/movietag1

[cloudera@quickstart Desktop]$ hadoop dfs -cat /user/root/movietag1/part-r-00000



```
[cloudera@quickstart Desktop]$ hadoop dfs -cat /user/root/movietag1/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Toy Story (1995)        ,[pixar, funny]
Jumanji (1995)  ,[childish]
rumpier old men (1995)  ,[oldie]
[cloudera@quickstart Desktop]$
```

Program 5

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

MongoDB: A Powerful NoSQL Database

MongoDB is a popular open-source, non-relational database management system (DBMS) that stores data in flexible, JSON-like documents. Unlike traditional relational databases (RDBMS) that use tables and rows, MongoDB utilizes collections and documents for data storage and retrieval.

MongoDB's key features:

1. **Document-oriented model**
    - Data is stored in documents similar to JSON objects (actually BSON, a binary JSON format), making it intuitive for developers and aligning well with object-oriented programming.
    - Documents are grouped into collections, similar to tables in RDBMS, according to Board Infinity.
    - Offers a flexible schema, meaning documents within a collection can have different fields, allowing for dynamic changes to the data model without downtime

2. **High scalability**
    - Supports horizontal scaling through sharding, distributing data across multiple servers to handle large datasets and high traffic efficiently.
    - Replica sets provide high availability and redundancy by maintaining multiple copies of data across different servers, enabling automatic failover in case of primary server failure

3. **Performance**
    - Optimized for high read and write throughput, suitable for applications demanding rapid data processing.
    - Leverages indexing, replication, and sharding to enhance performance and manage intensive workloads.
    - Its in-memory storage engine contributes to fast performance, especially for read-heavy operations.

4. **Rich query language and aggregation**
    - Provides a powerful query language (MQL) that's flexible and allows for complex queries, including field, range, and regular expression searches.
    - The aggregation framework enables sophisticated data transformations and aggregations within the database, according to Ksolves.

5. **Other important features**
   - Ad-hoc queries: Supports flexible and real-time queries without predefined schemas.
   - Geospatial data support: Offers built-in capabilities for applications requiring location-based services.
   - Transactions: Supports multi-document ACID transactions, though they might not be as mature or efficient as in traditional RDBMS.
   - Built-in security: Includes authentication mechanisms like SCRAM and role-based access controls

Advantages of MongoDB

- **Scalability**: Handles large datasets and traffic spikes effectively via horizontal scaling.
- **Flexibility**: Adapts easily to evolving data requirements due to its schema-less nature.
- **Performance**: Delivers high read and write performance, particularly for large volumes of data.
- **Developer-friendly**: Intuitive document model and query language simplify development and reduce the need for complex object-relational mapping (ORM).
- **Cloud-native**: Offers MongoDB Atlas, a fully managed cloud database service on major cloud providers like AWS, Azure, and Google Cloud, says MongoDB

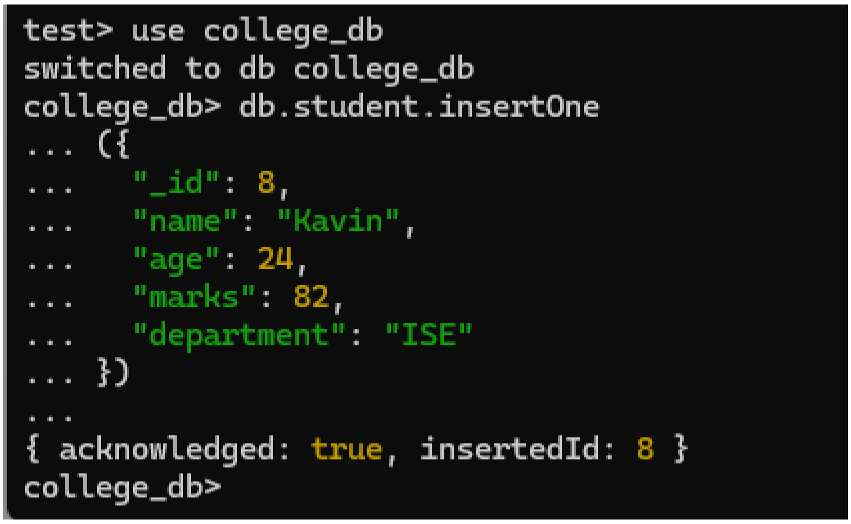
Disadvantages of MongoDB

- **Memory Usage**: Can be memory-intensive, especially for large datasets, potentially leading to higher resource costs.
- **Transactions**: Although MongoDB has transaction capabilities, complex transactions across multiple operations might be less robust compared to RDBMS.
- **Consistency**: MongoDB prioritizes scalability and availability, potentially leading to eventual consistency rather than immediate consistency in certain scenarios.
- **Indexing limitations**: While robust, incorrect or excessive indexing can impact write performance.
- **Data Duplication**: Denormalized data modeling can lead to redundancy and increased storage.

MongoDB Operations: Count, Sort, Limit, Skip, Aggregate

shop_db> use college_db

switched to db college_db



```
test> use college_db
switched to db college_db
college_db> db.student.insertOne
... ({
...   "_id": 8,
...   "name": "Kavin",
...   "age": 24,
...   "marks": 82,
...   "department": "ISE"
... })
...
{ acknowledged: true, insertedId: 8 }
college_db>
```

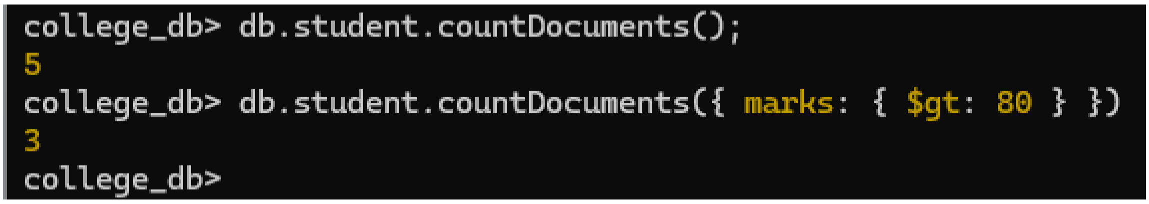
1. Count Documents

Count all documents:

db.student.countDocuments()

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

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

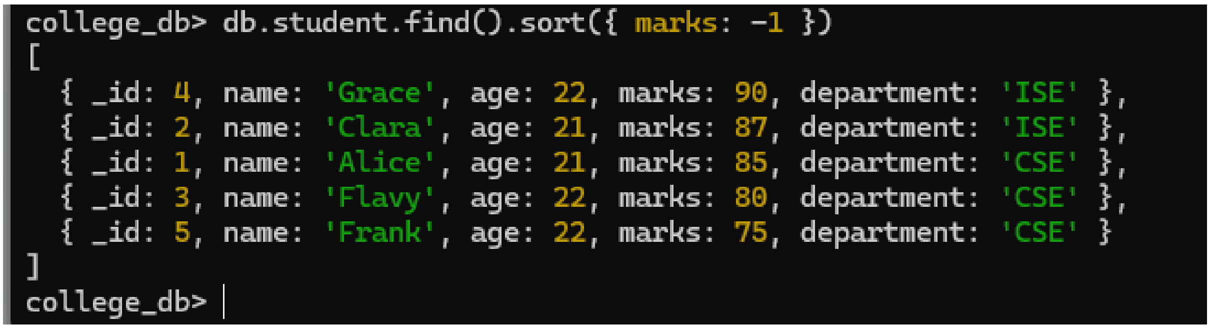


```
college_db> db.student.countDocuments();
5
college_db> db.student.countDocuments({ marks: { $gt: 80 } })
3
college_db>
```

2. Sort Documents

Sort by marks in descending order:

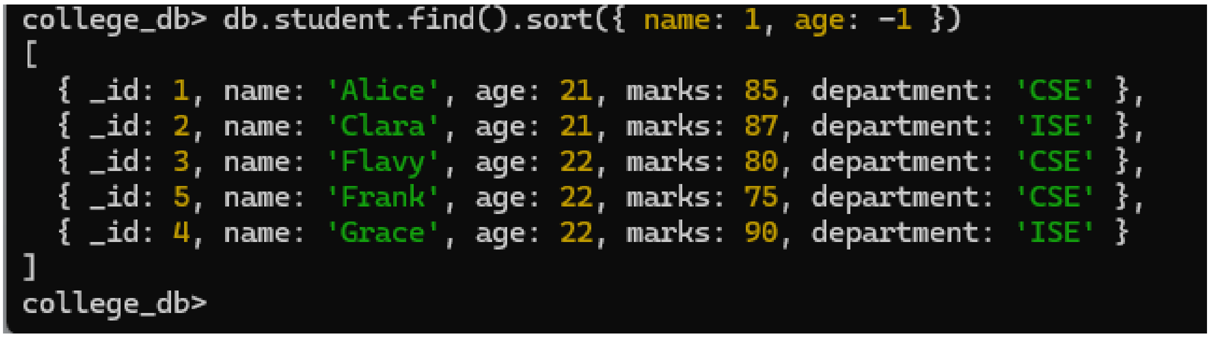
db.student.find().sort({ marks: -1 })



```
college_db> db.student.find().sort({ marks: -1 })
[
  { _id: 4, name: 'Grace', age: 22, marks: 90, department: 'ISE' },
  { _id: 2, name: 'Clara', age: 21, marks: 87, department: 'ISE' },
  { _id: 1, name: 'Alice', age: 21, marks: 85, department: 'CSE' },
  { _id: 3, name: 'Flavy', age: 22, marks: 80, department: 'CSE' },
  { _id: 5, name: 'Frank', age: 22, marks: 75, department: 'CSE' }
]
college_db>
```

Sort by name ascending and age descending:

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

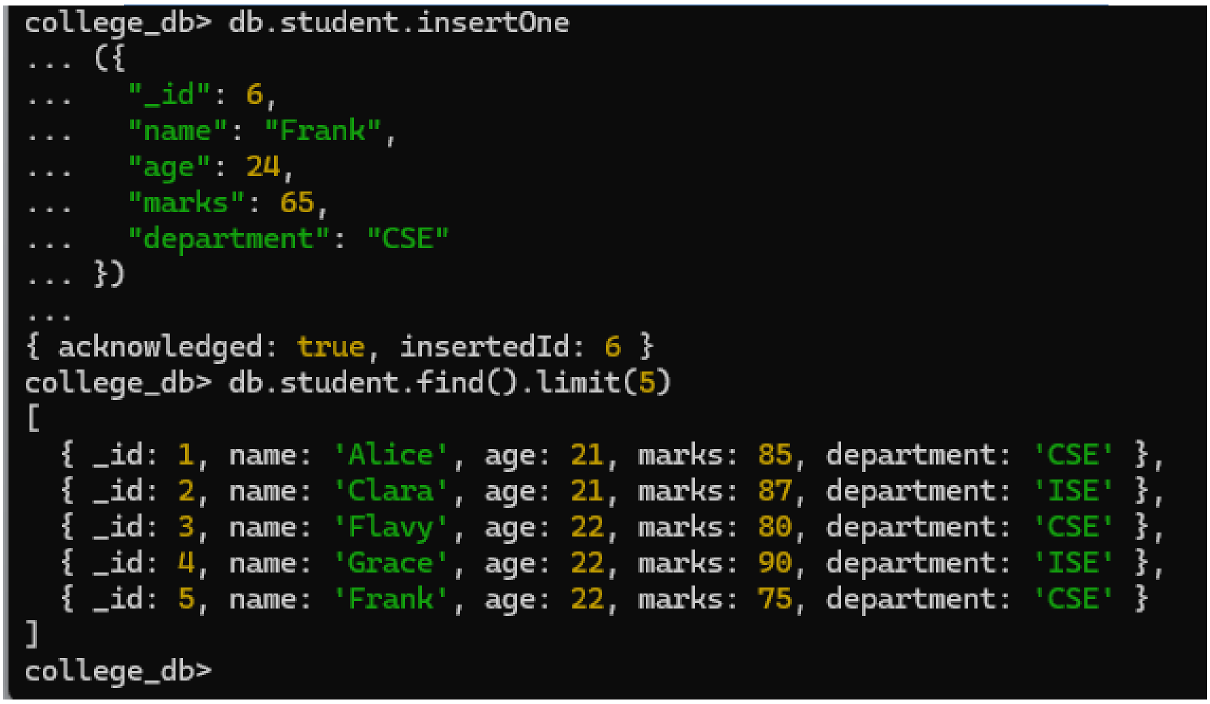


```
college_db> db.student.find().sort({ name: 1, age: -1 })
[
  { _id: 1, name: 'Alice', age: 21, marks: 85, department: 'CSE' },
  { _id: 2, name: 'Clara', age: 21, marks: 87, department: 'ISE' },
  { _id: 3, name: 'Flavy', age: 22, marks: 80, department: 'CSE' },
  { _id: 5, name: 'Frank', age: 22, marks: 75, department: 'CSE' },
  { _id: 4, name: 'Grace', age: 22, marks: 90, department: 'ISE' }
]
college_db>
```

3. Limit Results

Return top 5 students:

db.student.find().limit(5)

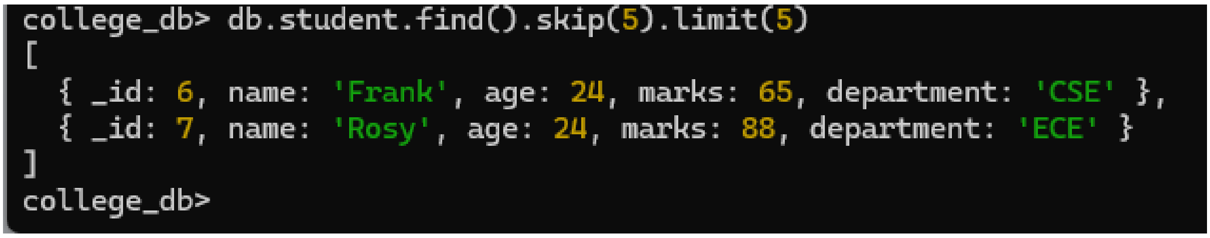


```
college_db> db.student.insertOne
... ({
...    "_id": 6,
...    "name": "Frank",
...    "age": 24,
...    "marks": 65,
...    "department": "CSE"
... })
...
{ acknowledged: true, insertedId: 6 }
college_db> db.student.find().limit(5)
[
  { _id: 1, name: 'Alice', age: 21, marks: 85, department: 'CSE' },
  { _id: 2, name: 'Clara', age: 21, marks: 87, department: 'ISE' },
  { _id: 3, name: 'Flavy', age: 22, marks: 80, department: 'CSE' },
  { _id: 4, name: 'Grace', age: 22, marks: 90, department: 'ISE' },
  { _id: 5, name: 'Frank', age: 22, marks: 75, department: 'CSE' }
]
college_db>
```

4. Skip Documents

Skip first 5 documents and get the next 5:

db.student.find().skip(5).limit(5)

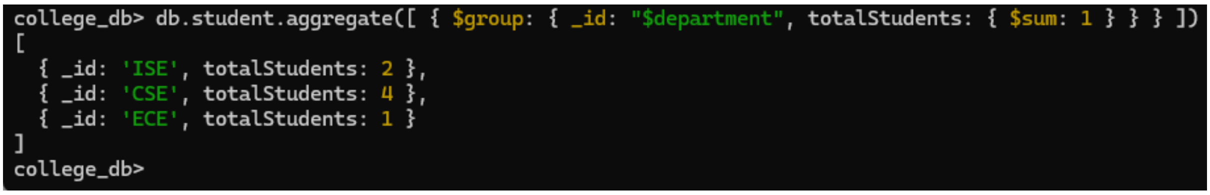


```
college_db> db.student.find().skip(5).limit(5)
[
  { _id: 6, name: 'Frank', age: 24, marks: 65, department: 'CSE' },
  { _id: 7, name: 'Rosy', age: 24, marks: 88, department: 'ECE' }
]
college_db>
```

5. Aggregate Documents

Group by Department and Count Students:

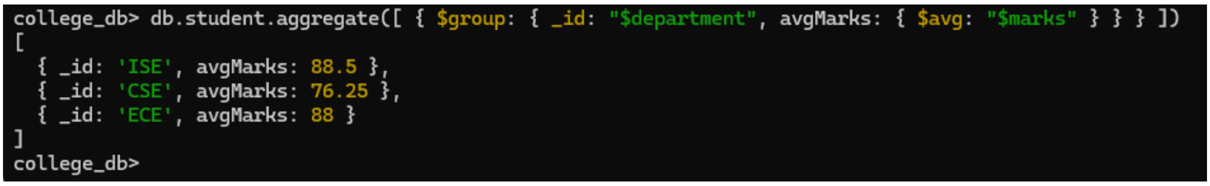
```
db.student.aggregate([ { $group: { _id: "$department", totalStudents: { $sum: 1 } } } ])
```



```
college_db> db.student.aggregate([ { $group: { _id: "$department", totalStudents: { $sum: 1 } } } ])
[
  { _id: 'ISE', totalStudents: 2 },
  { _id: 'CSE', totalStudents: 4 },
  { _id: 'ECE', totalStudents: 1 }
]
college_db>
```

Group by Department and Average Marks:

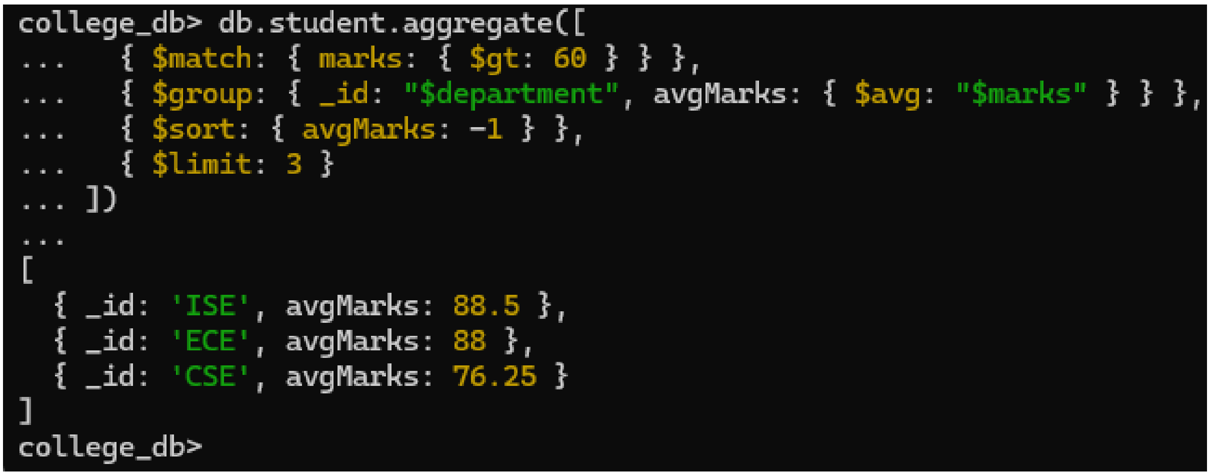
```
db.student.aggregate([ { $group: { _id: "$department", avgMarks: { $avg: "$marks" } } } ])
```



```
college_db> db.student.aggregate([ { $group: { _id: "$department", avgMarks: { $avg: "$marks" } } } ])
[
  { _id: 'ISE', avgMarks: 88.5 },
  { _id: 'CSE', avgMarks: 76.25 },
  { _id: 'ECE', avgMarks: 88 }
]
college_db>
```

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

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

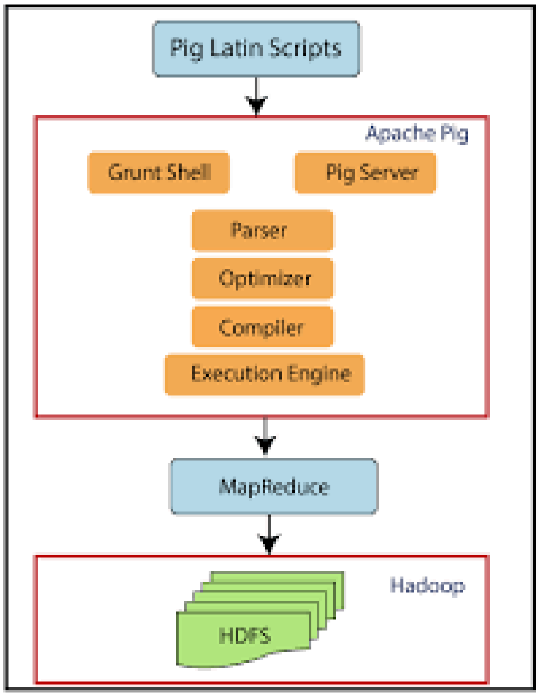


```
college_db> db.student.aggregate([
...   { $match: { marks: { $gt: 60 } } },
...   { $group: { _id: "$department", avgMarks: { $avg: "$marks" } } },
...   { $sort: { avgMarks: -1 } },
...   { $limit: 3 }
... ])
...
[
  { _id: 'ISE', avgMarks: 88.5 },
  { _id: 'ECE', avgMarks: 88 },
  { _id: 'CSE', avgMarks: 76.25 }
]
college_db>
```

Program 6

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

Pig Latin scripts are used with Apache Pig, a high-level platform for processing large datasets, to create data analysis codes. These scripts are written in Pig Latin, a language that abstracts the complexities of MapReduce, allowing users to focus on data analysis rather than low-level programming. Pig Latin scripts are submitted to the Apache Pig server, parsed, optimized, and then compiled into MapReduce code for execution on a Hadoop cluster.



Breakdown of key aspects:

1. Purpose:

- Pig Latin scripts are used to analyze data stored in Hadoop's distributed file system (HDFS).
- They provide a procedural data flow language with syntax and commands for implementing business logic.
- The scripts are converted into MapReduce jobs by the Pig Engine, abstracting the underlying MapReduce complexity from the user.

2. Structure:

- Pig Latin scripts consist of a series of statements that define data transformations.
- These statements include loading data, filtering, projecting, joining, grouping, and storing results.
- LOAD statements specify input data locations, format, and schema.
- STORE statements save the processed data, while DUMP statements display results on the command line.

3. Key Components:

- **Pig Latin Language**: The high-level language used for writing data analysis scripts.
- **Pig Engine**: The runtime engine that compiles Pig Latin scripts into MapReduce code.
- **HDFS**: The Hadoop Distributed File System where data is stored and retrieved.

4. Execution Flow:

- Pig Latin scripts are submitted to the Apache Pig server.
- The Pig Latin compiler parses, validates, and optimizes the script.
- The optimized script is then converted into a sequence of MapReduce jobs.
- These jobs are executed by the Pig Engine, leveraging the Hadoop cluster.

```
--load data
students = LOAD '/user/root/pigdata/students.txt'   USING
PigStorage(',')  AS (id:int, name:chararray, dept:chararray,
marks:int);

-- Filter students with marks > 80
high_scorers = FILTER students BY marks > 80;

-- Sort all students by marks descending
sorted_students = ORDER students BY marks DESC;

--Project only name and marks
projected = FOREACH students GENERATE name, marks;

-- Group by department and find average marks
grouped = GROUP students BY dept;
average_marks = FOREACH grouped GENERATE group AS department,
AVG(students.marks) AS avg_marks;

STORE sorted_students INTO '/user/root/pigoutput/sorted_students'
USING PigStorage(',');
STORE high_scorers INTO '/user/root/pigoutput/high_scorers' USING
PigStorage(',');
STORE projected INTO '/user/root/pigoutput/projected' USING
PigStorage(',');
STORE average_marks INTO '/user/root/pigoutput/average_marks' USING
PigStorage(',');
```

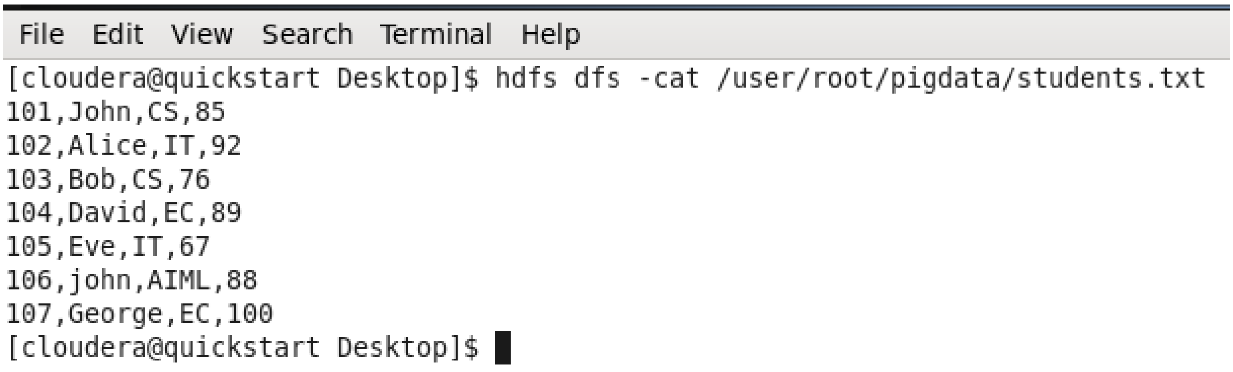
OUTPUT:

[cloudera@quickstart ~]$ hdfs dfs -mkdir /user/root/pigdata

[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal students.txt /user/root/pigdata

[cloudera@quickstart ~]$ hdfs dfs -cat /user/root/pigdata/students.txt

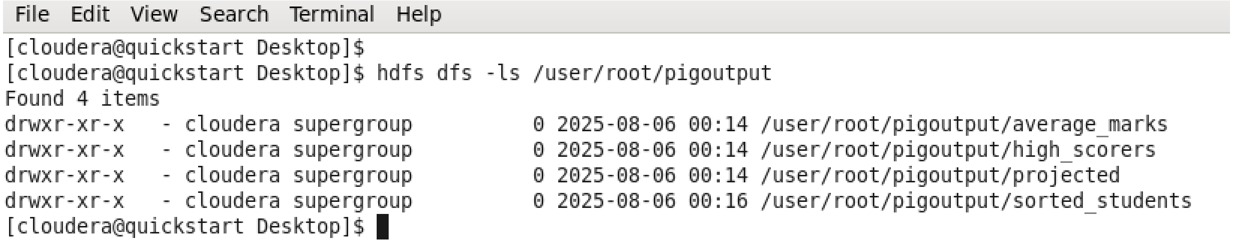
INPUT DATA : students.txt



```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /user/root/pigdata/students.txt
101,John,CS,85
102,Alice,IT,92
103,Bob,CS,76
104,David,EC,89
105,Eve,IT,67
106,john,AIML,88
107,George,EC,100
[cloudera@quickstart Desktop]$
```

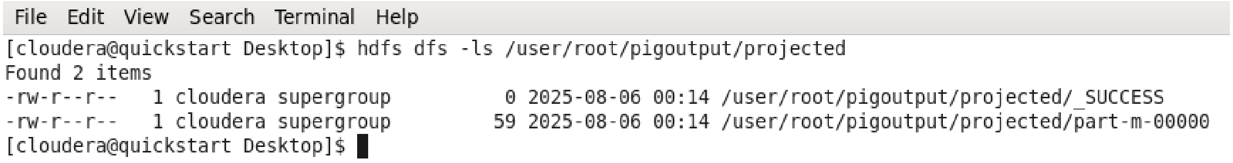
[cloudera@quickstart ~]$ pig -x mapreduce PigExample.pig /user/root/pigdata/studets.txt

[cloudera@quickstart ~]$ hdfs dfs -ls /user/root/pigoutput



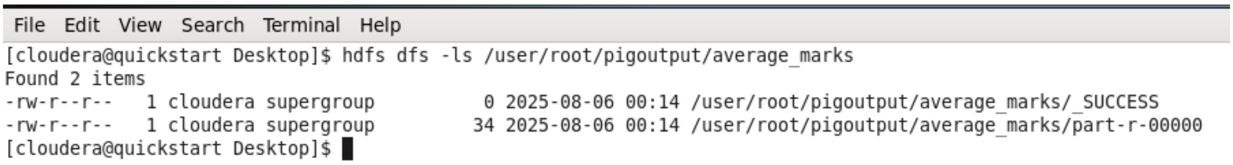
```
[cloudera@quickstart Desktop]$
[cloudera@quickstart Desktop]$ hdfs dfs -ls /user/root/pigoutput
Found 4 items
drwxr-xr-x   - cloudera supergroup          0 2025-08-06 00:14 /user/root/pigoutput/average_marks
drwxr-xr-x   - cloudera supergroup          0 2025-08-06 00:14 /user/root/pigoutput/high_scorers
drwxr-xr-x   - cloudera supergroup          0 2025-08-06 00:14 /user/root/pigoutput/projected
drwxr-xr-x   - cloudera supergroup          0 2025-08-06 00:16 /user/root/pigoutput/sorted_students
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart ~]$ hdfs dfs -ls /user/root/pigoutput/projected



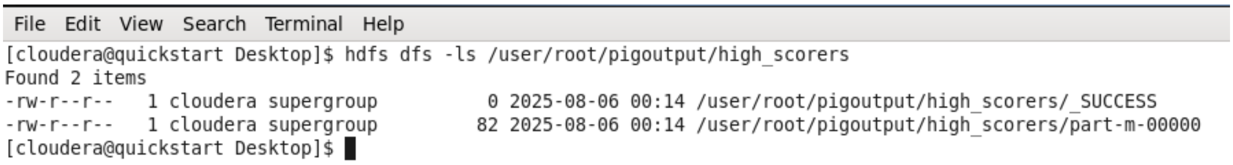
```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /user/root/pigoutput/projected
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2025-08-06 00:14 /user/root/pigoutput/projected/_SUCCESS
-rw-r--r--   1 cloudera supergroup         59 2025-08-06 00:14 /user/root/pigoutput/projected/part-m-00000
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart ~]$ hdfs dfs -ls /user/root/pigoutput/average_marks



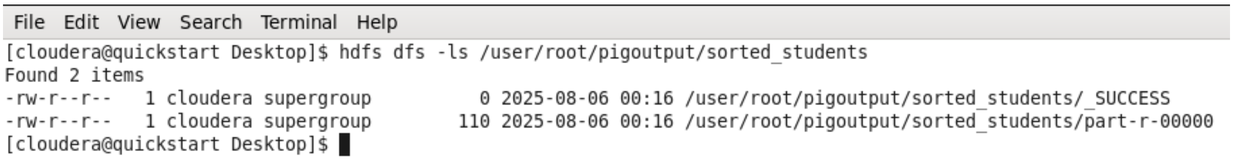
```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /user/root/pigoutput/average_marks
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2025-08-06 00:14 /user/root/pigoutput/average_marks/_SUCCESS
-rw-r--r--   1 cloudera supergroup         34 2025-08-06 00:14 /user/root/pigoutput/average_marks/part-r-00000
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart ~]$ hdfs dfs -ls /user/root/pigoutput/high_scorers



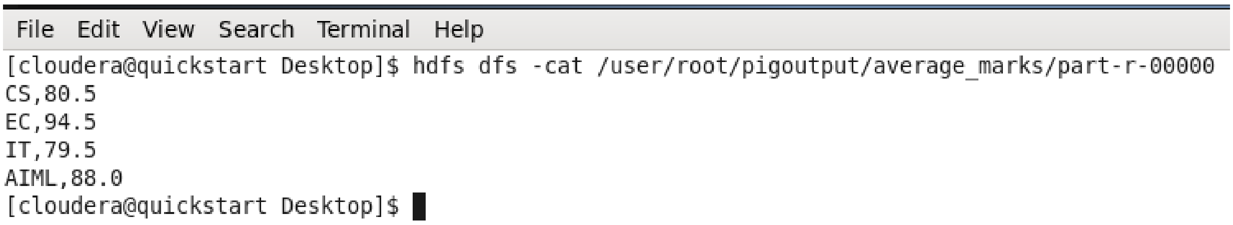
```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /user/root/pigoutput/high_scorers
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2025-08-06 00:14 /user/root/pigoutput/high_scorers/_SUCCESS
-rw-r--r--   1 cloudera supergroup         82 2025-08-06 00:14 /user/root/pigoutput/high_scorers/part-m-00000
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart ~]$ hdfs dfs -ls /user/root/pigoutput/sorted_students



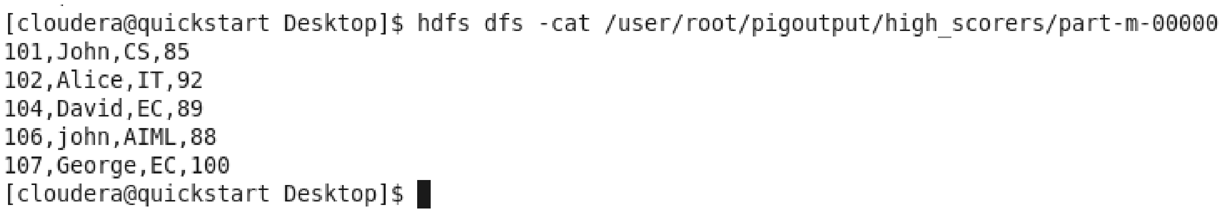
```
[cloudera@quickstart Desktop]$ hdfs dfs -ls /user/root/pigoutput/sorted_students
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2025-08-06 00:16 /user/root/pigoutput/sorted_students/_SUCCESS
-rw-r--r--   1 cloudera supergroup        110 2025-08-06 00:16 /user/root/pigoutput/sorted_students/part-r-00000
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart ~]$ hdfs dfs -cat /user/root/pigoutput/average_marks/part-r-00000



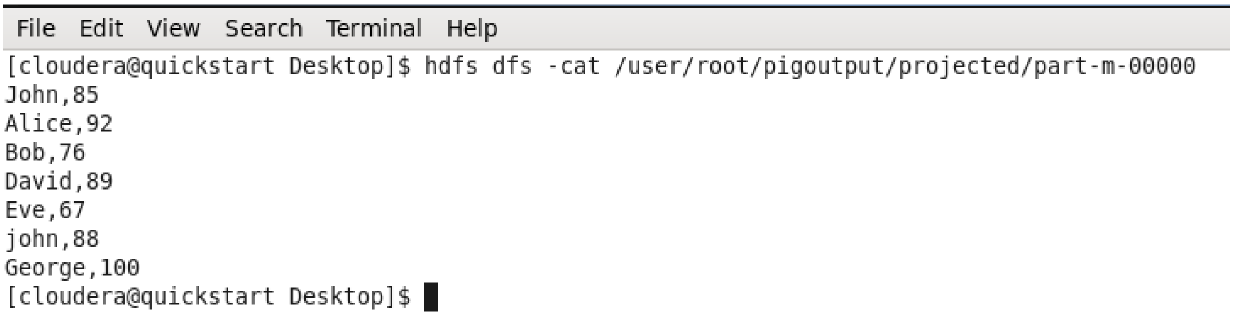
```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /user/root/pigoutput/average_marks/part-r-00000
CS,80.5
EC,94.5
IT,79.5
AIML,88.0
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart ~]$ hdfs dfs -cat /user/root/pigoutput/high_scorers/part-m-00000



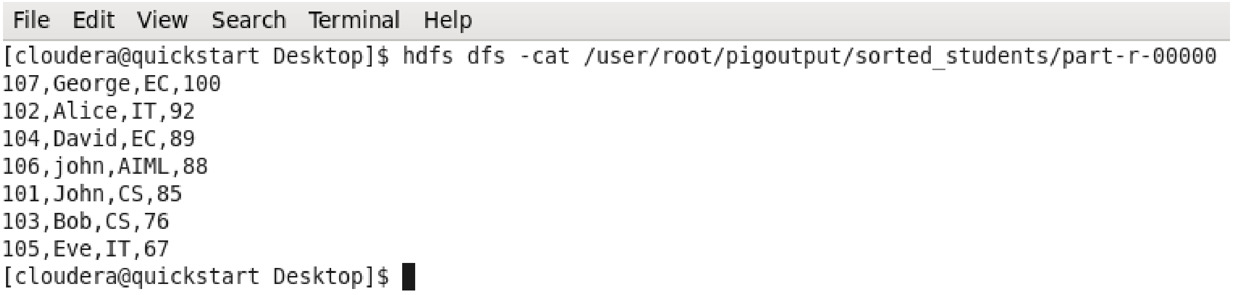
```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /user/root/pigoutput/high_scorers/part-m-00000
101,John,CS,85
102,Alice,IT,92
104,David,EC,89
106,john,AIML,88
107,George,EC,100
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart ~]$ hdfs dfs -cat /user/root/pigoutput/projected/part-m-00000



```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /user/root/pigoutput/projected/part-m-00000
John,85
Alice,92
Bob,76
David,89
Eve,67
john,88
George,100
[cloudera@quickstart Desktop]$
```

[cloudera@quickstart ~]$ hdfs dfs -cat /user/root/pigoutput/sorted_students/part-r-00000



```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /user/root/pigoutput/sorted_students/part-r-00000
107,George,EC,100
102,Alice,IT,92
104,David,EC,89
106,john,AIML,88
101,John,CS,85
103,Bob,CS,76
105,Eve,IT,67
[cloudera@quickstart Desktop]$
```

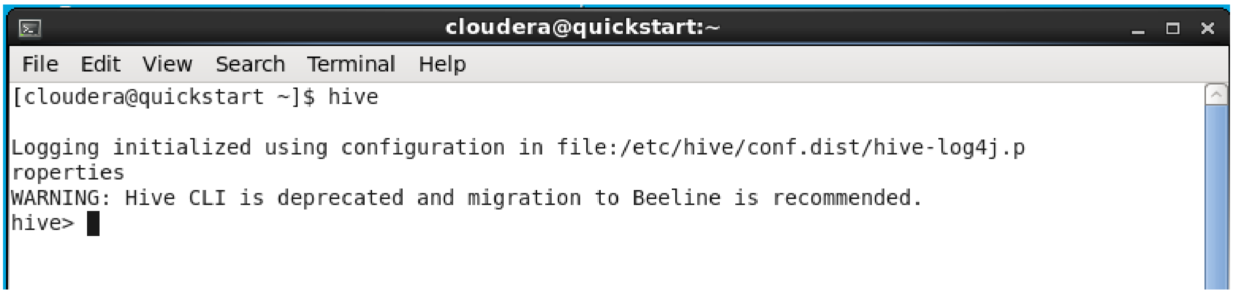
Program 7

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

HiveQL or HQL is a Hive query language that we used to process or query structured data on Hive. HQL syntaxes are very much similar to MySQL but have some significant differences. We will use the hive command, which is a bash shell script to complete our hive demo using CLI(Command Line Interface). We can easily start hive shell by simply typing hive in the terminal.

Databases in Apache Hive

The Database is a storage schema that contains multiple tables. The Hive Databases refer to the namespace of tables. If you don't specify the database name by default Hive uses its default database for table creation and other purposes. Creating a Database allows multiple users to create tables with a similar name in different schemas so that their names don't match.



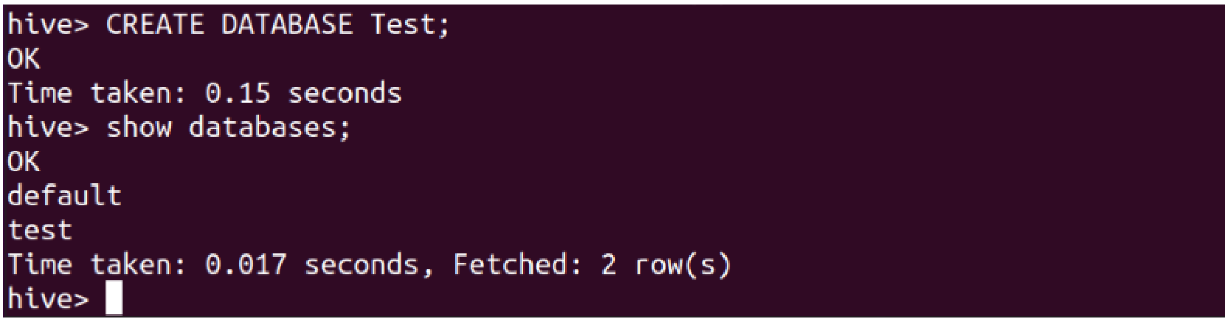
```
[cloudera@quickstart ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.p
roperties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>
```

Create Database Syntax:

We can create a database with the help of the below command but if the database already exists then, in that case, Hive will throw an error.

```
CREATE DATABASE|SCHEMA <database name>    # we can use DATABASE or SCHEMA for
creation of DB
```

Example:



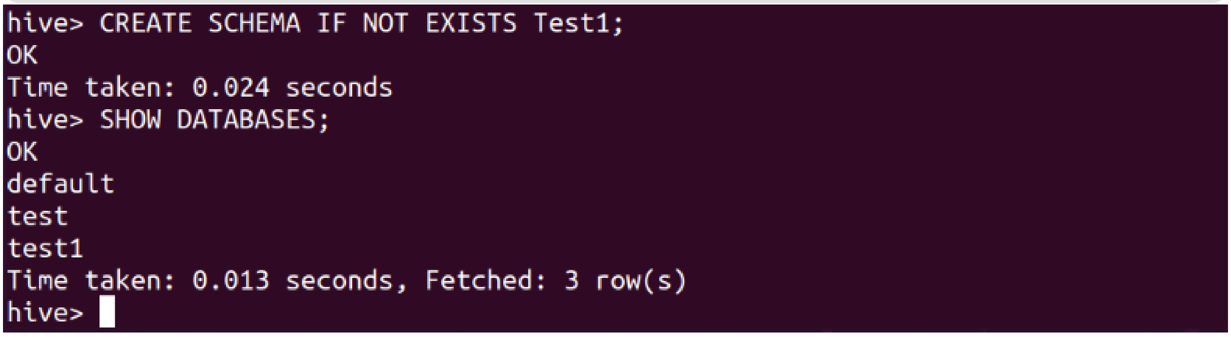
```
hive> CREATE DATABASE Test;
OK
Time taken: 0.15 seconds
hive> show databases;
OK
default
test
Time taken: 0.017 seconds, Fetched: 2 row(s)
hive>
```

If we again try to create a Test database hive will throw an error/warning that the database with the name Test already exists. In general, we don't want to get an error if the database exists. So we use the create database command with [IF NOT EXIST] clause. This will do not throw any error.

```
CREATE SCHEMA IF NOT EXISTS Test1;

SHOW DATABASES;
```

Example:

```
CREATE SCHEMA IF NOT EXISTS Test1;

SHOW DATABASES;
```



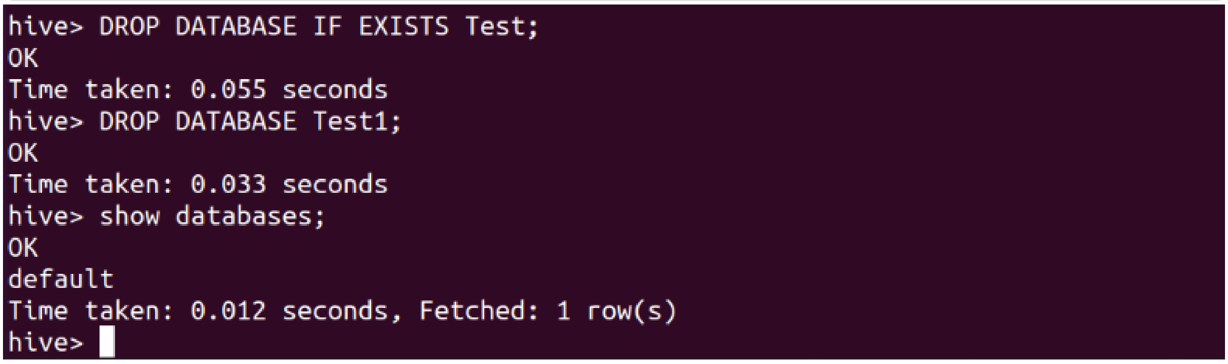
```
hive> CREATE SCHEMA IF NOT EXISTS Test1;
OK
Time taken: 0.024 seconds
hive> SHOW DATABASES;
OK
default
test
test1
Time taken: 0.013 seconds, Fetched: 3 row(s)
hive> 
```

Syntax To Drop Existing Databases:

```
DROP DATABASE <db_name>;  or  DROP DATABASE IF EXIST <db_name>  # The IF EXIST
clause again is used to suppress error
```

Example:

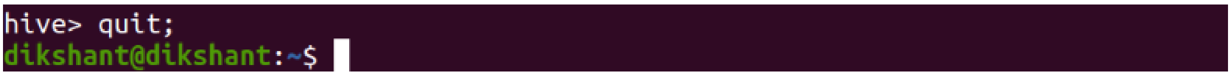
```
DROP DATABASE IF EXISTS Test;

DROP DATABASE Test1;
```



```
hive> DROP DATABASE IF EXISTS Test;
OK
Time taken: 0.055 seconds
hive> DROP DATABASE Test1;
OK
Time taken: 0.033 seconds
hive> show databases;
OK
default
Time taken: 0.012 seconds, Fetched: 1 row(s)
hive> 
```

Now quit hive shell with quit command.

```
quit;
```



```
hive> quit;
dikshant@dikshant:~$ 
```

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'='vijay');

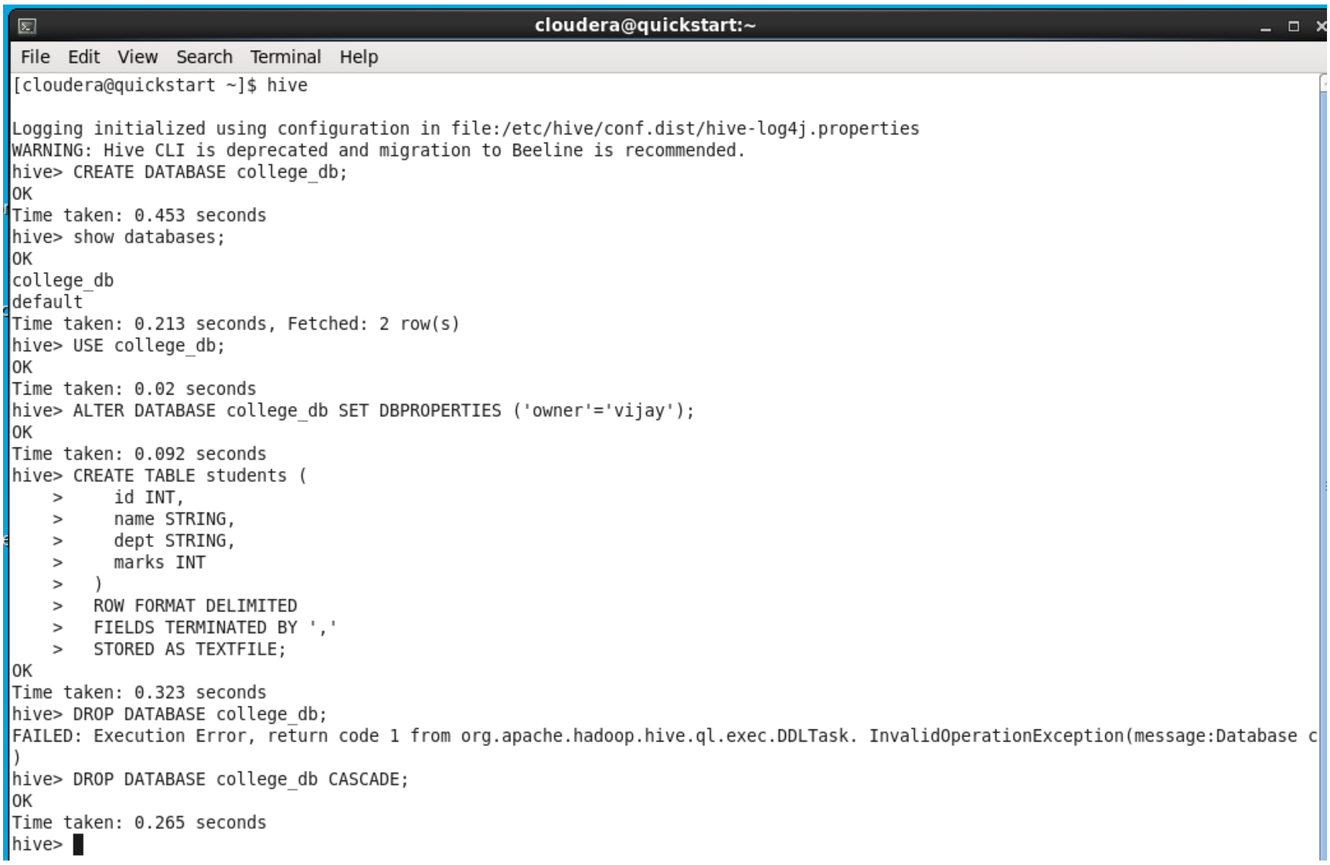
- **• Drop the Database:**

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;
```



```
[cloudera@quickstart ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> CREATE DATABASE college_db;
OK
Time taken: 0.453 seconds
hive> show databases;
OK
college_db
default
Time taken: 0.213 seconds, Fetched: 2 row(s)
hive> USE college_db;
OK
Time taken: 0.02 seconds
hive> ALTER DATABASE college_db SET DBPROPERTIES ('owner'='vijay');
OK
Time taken: 0.092 seconds
hive> CREATE TABLE students (
    >     id INT,
    >     name STRING,
    >     dept STRING,
    >     marks INT
    >   )
    >   ROW FORMAT DELIMITED
    >   FIELDS TERMINATED BY ','
    >   STORED AS TEXTFILE;
OK
Time taken: 0.323 seconds
hive> DROP DATABASE college_db;
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. InvalidOperationException(message:Database c
)
hive> DROP DATABASE college_db CASCADE;
OK
Time taken: 0.265 seconds
hive> 
```

- • **Load Data into Table:**

LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/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:**

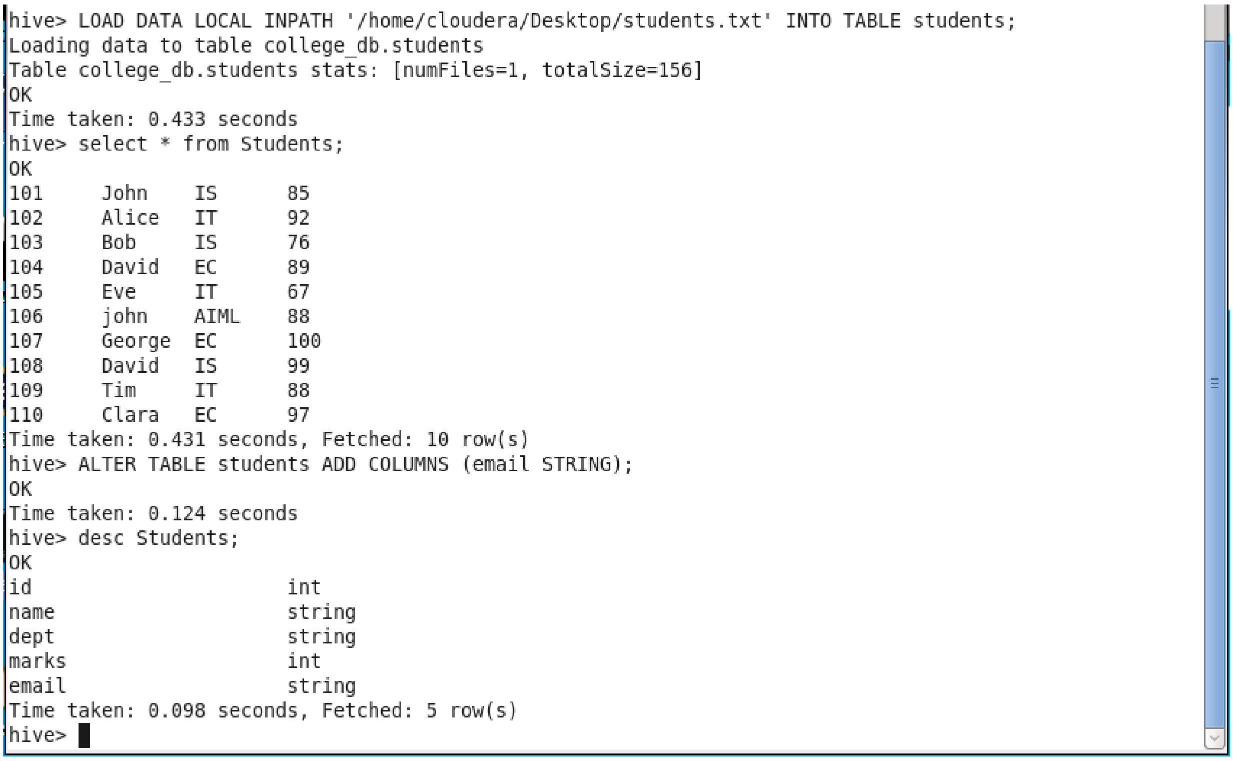
CREATE VIEW high_scorers AS SELECT name, marks FROM students WHERE marks > 80;

- • **Use/View it:**

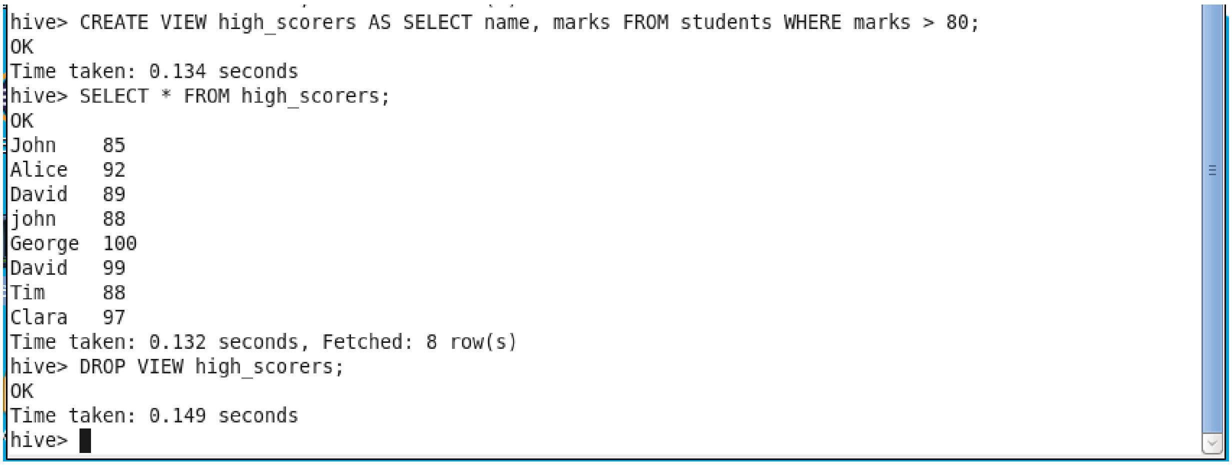
SELECT * FROM high_scorers;

- • **Drop View:**

DROP VIEW high_scorers;



```
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Desktop/students.txt' INTO TABLE students;
Loading data to table college_db.students
Table college_db.students stats: [numFiles=1, totalSize=156]
OK
Time taken: 0.433 seconds
hive> select * from Students;
OK
101     John    IS      85
102     Alice   IT      92
103     Bob     IS      76
104     David   EC      89
105     Eve     IT      67
106     john    AIML    88
107     George  EC      100
108     David   IS      99
109     Tim     IT      88
110     Clara   EC      97
Time taken: 0.431 seconds, Fetched: 10 row(s)
hive> ALTER TABLE students ADD COLUMNS (email STRING);
OK
Time taken: 0.124 seconds
hive> desc Students;
OK
id                      int
name                    string
dept                    string
marks                   int
email                   string
Time taken: 0.098 seconds, Fetched: 5 row(s)
hive> 
```



```
hive> CREATE VIEW high_scorers AS SELECT name, marks FROM students WHERE marks > 80;
OK
Time taken: 0.134 seconds
hive> SELECT * FROM high_scorers;
OK
John    85
Alice   92
David   89
john    88
George  100
David   99
Tim     88
Clara   97
Time taken: 0.132 seconds, Fetched: 8 row(s)
hive> DROP VIEW high_scorers;
OK
Time taken: 0.149 seconds
hive> 
```

4. Functions

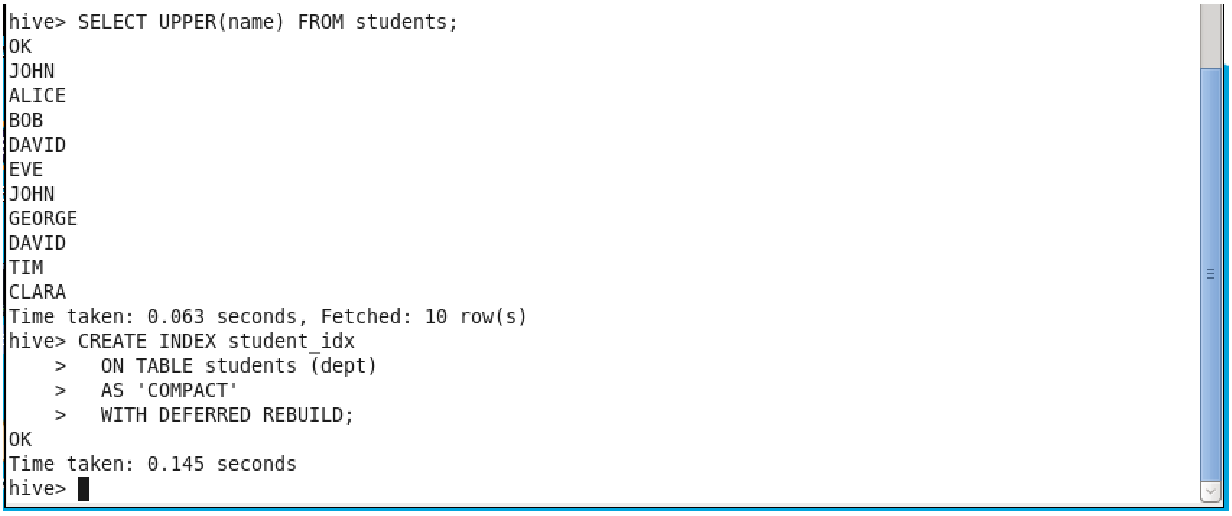
- **• Built-in Function Example:**

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;



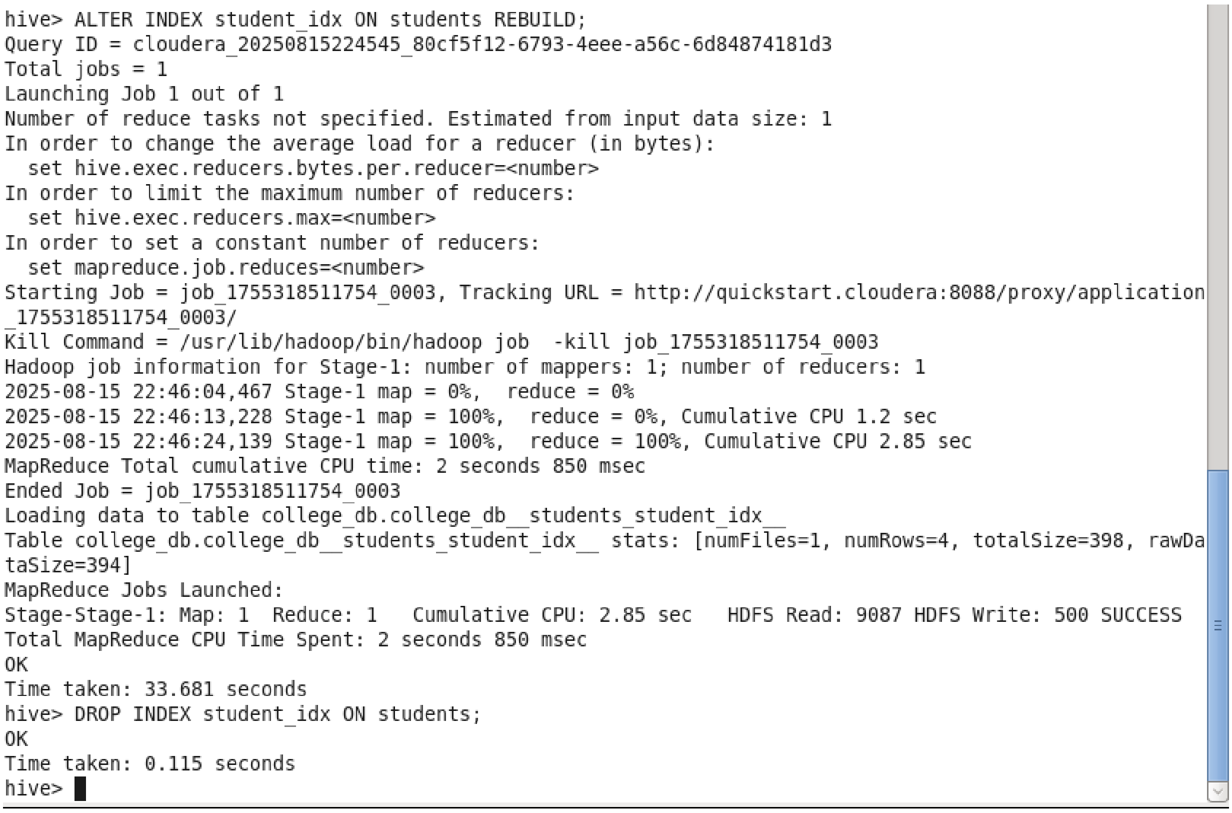
```
hive> SELECT UPPER(name) FROM students;
OK
JOHN
ALICE
BOB
DAVID
EVE
JOHN
GEORGE
DAVID
TIM
CLARA
Time taken: 0.063 seconds, Fetched: 10 row(s)
hive> CREATE INDEX student_idx
    >   ON TABLE students (dept)
    >   AS 'COMPACT'
    >   WITH DEFERRED REBUILD;
OK
Time taken: 0.145 seconds
hive> 
```

- **• Build Index:**

ALTER INDEX student_idx ON students REBUILD;

- **• Drop Index:**

DROP INDEX student_idx ON students;



```
hive> ALTER INDEX student_idx ON students REBUILD;
Query ID = cloudera_20250815224545_80cf5f12-6793-4eee-a56c-6d84874181d3
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1755318511754_0003, Tracking URL = http://quickstart.cloudera:8088/proxy/application
_1755318511754_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1755318511754_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-08-15 22:46:04,467 Stage-1 map = 0%,  reduce = 0%
2025-08-15 22:46:13,228 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.2 sec
2025-08-15 22:46:24,139 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.85 sec
MapReduce Total cumulative CPU time: 2 seconds 850 msec
Ended Job = job_1755318511754_0003
Loading data to table college_db.college_db__students_student_idx__
Table college_db.college_db__students_student_idx__ stats: [numFiles=1, numRows=4, totalSize=398, rawDa
taSize=394]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.85 sec   HDFS Read: 9087 HDFS Write: 500 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 850 msec
OK
Time taken: 33.681 seconds
hive> DROP INDEX student_idx ON students;
OK
Time taken: 0.115 seconds
hive> 
```

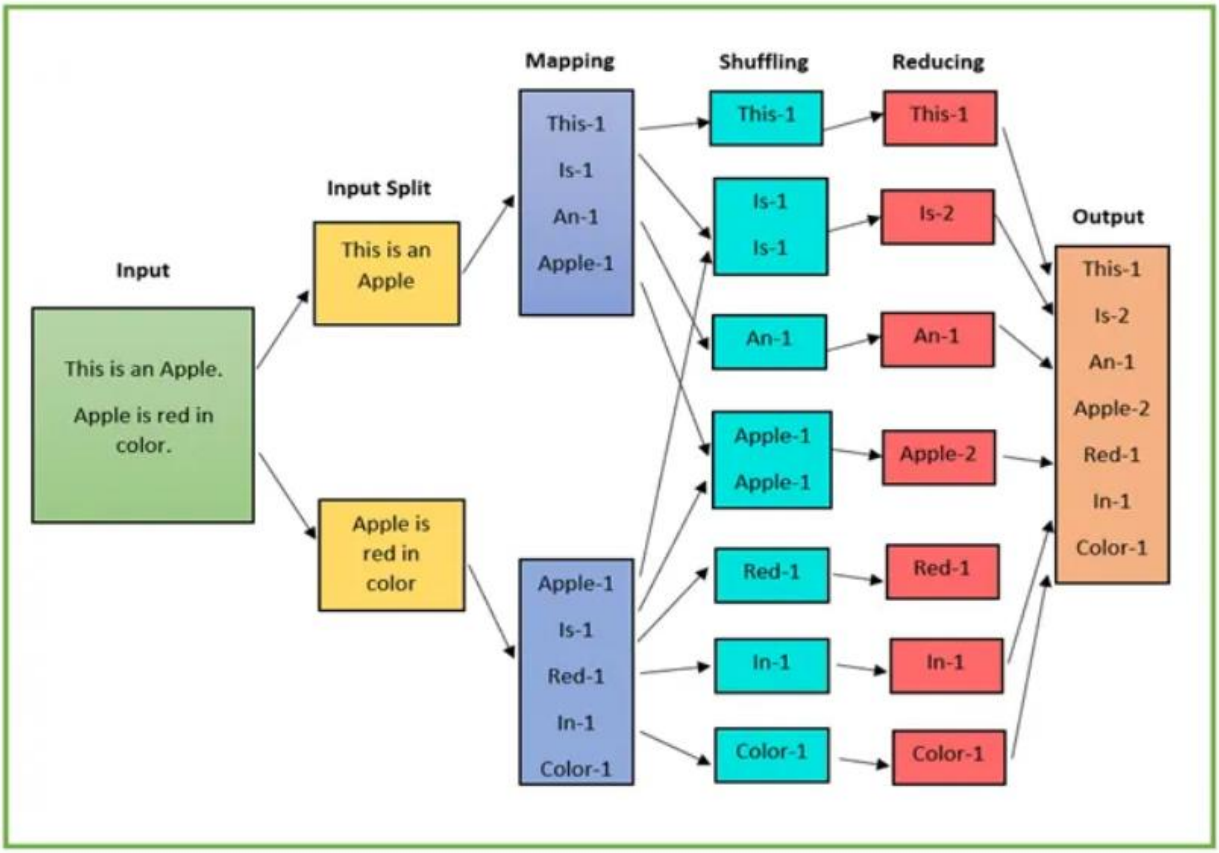
Program 8

Implement a word count program in Hadoop and Spark

MapReduce in Hadoop is a software framework for ease in writing applications of software, processing huge amounts of data. MapReduce provides the facility to distribute the workload (computations) among various nodes(analogous to commodity hardware). Hence, reducing the processing time as data on which the computation needs to be done is now divided into small chunks and individually processed. Through MapReduce you can achieve parallel processing resulting in faster execution of the job**.**

MapReduce Word Count is a framework which splits the chunk of data, sorts the map outputs and input to reduce tasks. A File-system stores the output and input of jobs. Re-execution of failed tasks, scheduling them and monitoring them is the task of the framework.

Figure below shows the architecture as well as working of MapReduce with an example:



Splitting: The parameter of splitter can be anything. By comma, space, by a new line or a semicolon.

Mapping: This is done as explained below.

Shuffle/Intermediate splitting: The process is usually parallel on cluster keys. The output of the map gets into the Reducer phase and all the similar keys of data are aligned in a cluster.

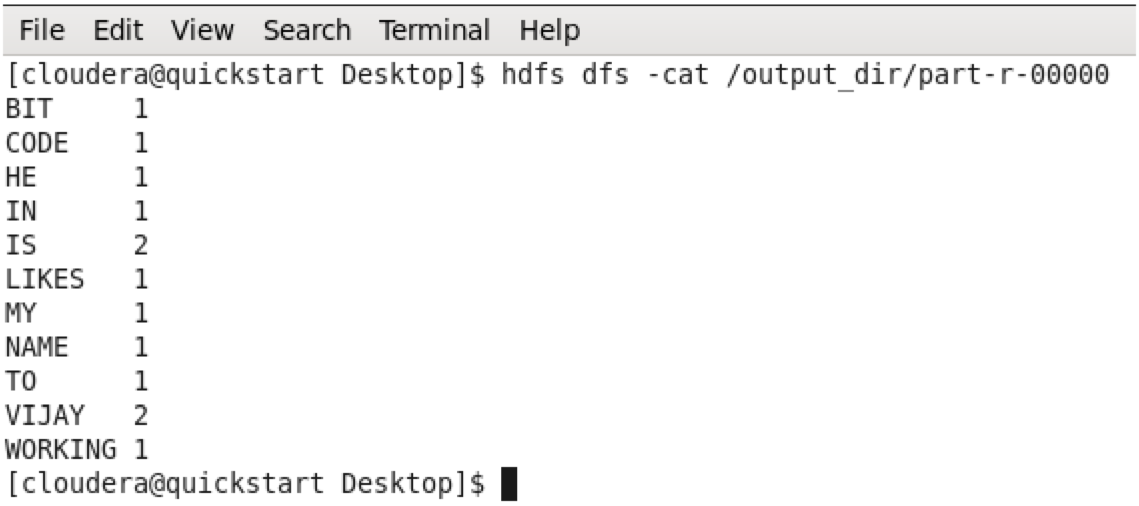
Reducing: This is done as explained below. Final result — All the data is clustered or combined to show the together form of a result.

Implementation a word count program in Hadoop

```
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();
    Path input=new Path(files[0]);
    Path output=new Path(files[1]);
    Job j=new Job(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);
  }
public static class MapForWordCount extends Mapper<LongWritable, Text, Text,
 IntWritable>{
  public void map(LongWritable key, Text value, Context con) throws
IOException, InterruptedException
  {
    String line = value.toString();
    String[] words=line.split(" ");
    for(String word: words )
    {
      Text outputKey = new Text(word.toUpperCase().trim());
      IntWritable outputValue = new IntWritable(1);
      con.write(outputKey, outputValue);
    }
  }
}
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));
  }
 }
}
```

OUTPUT:

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir /input_wordCount
[cloudera@quickstart ~]$ cd Desktop
[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal word.txt /input_wordCount
[cloudera@quickstart ~]$ hdfs dfs -cat /input_wordCount/word.txt
[cloudera@quickstart ~]$ hdfs dfs -ls /input_wordCount
[cloudera@quickstart ~]$ hadoop jar WordCount.jar WordCount /input_wordCount /output_dir
[cloudera@quickstart ~]$ hdfs dfs -cat /output_dir/*
```



```
[cloudera@quickstart Desktop]$ hdfs dfs -cat /output_dir/part-r-00000
BIT     1
CODE    1
HE      1
IN      1
IS      2
LIKES   1
MY      1
NAME    1
TO      1
VIJAY   2
WORKING 1
[cloudera@quickstart Desktop]$ 
```

Program 9

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

Cloudera Distribution Hadoop (CDH) was a popular open-source distribution of Apache Hadoop and related projects, designed for enterprise-level deployments. It offered a unified platform for storing and analyzing large datasets, integrating various components like HDFS, MapReduce, YARN, Spark, Hive, HBase, and more. Cloudera has since moved away from CDH and now offers the Cloudera Data Platform (CDP) which combines the strengths of CDH and Hortonworks Data Platform (HDP).

Key Features and Components of CDH:

- **Apache Hadoop Core:**
  CDH included core Hadoop components like HDFS (for distributed storage), MapReduce (for parallel processing), and YARN (for resource management).
- **Ecosystem Integration:**
  It integrated various other Apache projects to extend Hadoop's functionality, such as Spark (for fast data processing), Hive (for SQL-like querying), HBase (for NoSQL database), and more.
- **Cloudera Manager:**
  A management console for easy deployment, configuration, monitoring, and management of CDH clusters.
- **Enterprise-Ready:**
  CDH was designed for enterprise environments, providing features like security, high availability, and commercial support.
- **SQL-on-Hadoop:**
  Cloudera was a pioneer in SQL-on-Hadoop with its Impala query engine.
- **Node Templates:**
  CDH allowed for the creation of node templates with varying configurations within a Hadoop cluster, eliminating the need for uniform configurations.

Hue is a web-based interactive query editor that enables you to interact with databases and data warehouses. Data architects, SQL developers, and data engineers use Hue to create data models, clean data to prepare it for analysis, and to build and test SQL scripts for applications.

Hue offers powerful execution, debugging, and self-service capabilities to the following key Big Data personas:

Business Analysts
Data Engineers
Data Scientists
Power SQL users
Database Administrators
SQL Developers

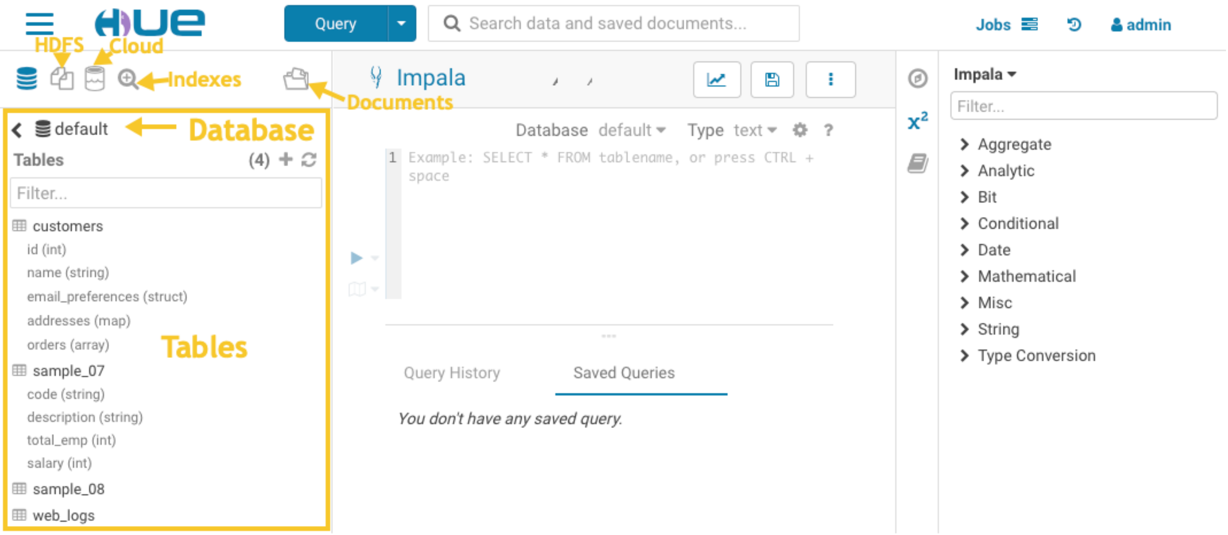
Steps to execute the program:

1. Create a table sales_data having following fields(id, date, region, product, qty, price, sales).
2. Query the data and visualize.

1. Create the table.

Open **Hue (**Hue is a web-based interactive query editor that enables you to interact with databases by running query).

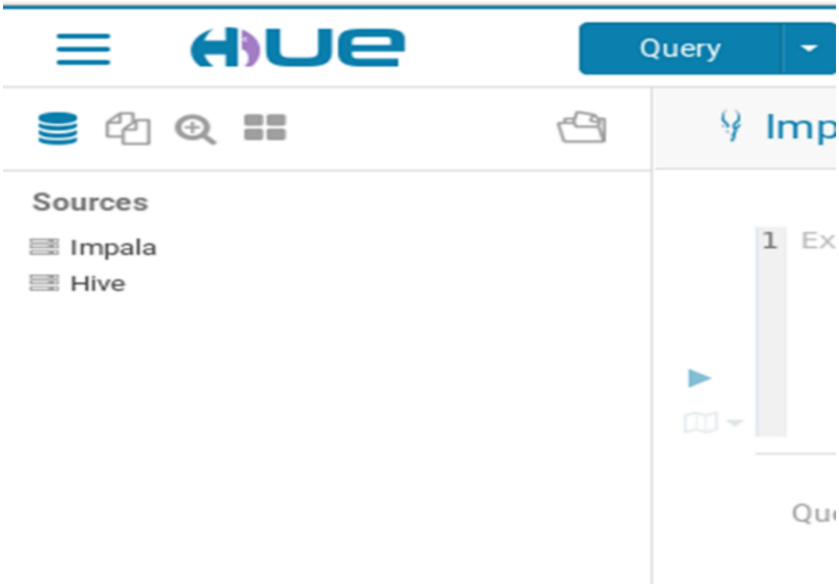
You can use Hue to: **Explore, browse, and import your data** through guided navigation in the left panel of the page:



Initially panel displays 2 options

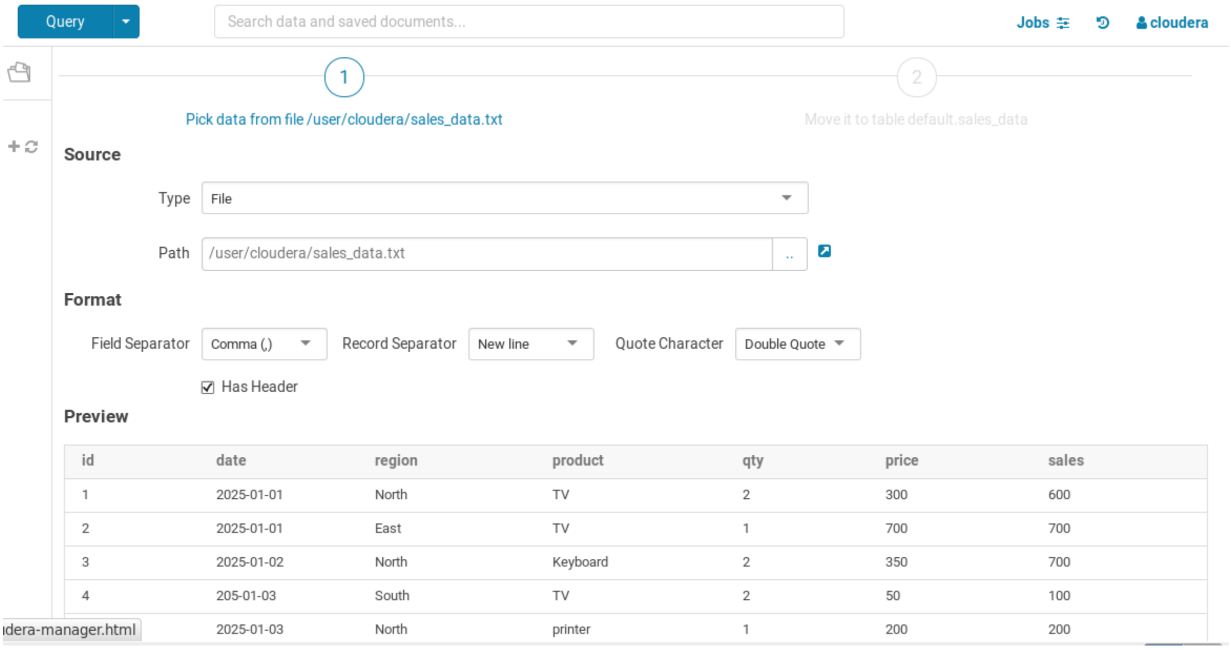
1. Impala

2. Hive as shown



- Select Hive
- Click on '+' to create tables under the Databases > Tables
- Under SOURCE, select Remote File from the Type drop-down menu.
- Click .. at the end of the Path field.
- The Choose a file modal is displayed.
- Browse and select the file you want to use to create a table. Hue displays the preview of the table along with the format.

  The following screenshot is the results of the above steps.



Pick data from file /user/cloudera/sales_data.txt

Move it to table default.sales_data

Source

Type File

Path /user/cloudera/sales_data.txt ..

Format

Field Separator Comma (,) Record Separator New line Quote Character Double Quote

☑ Has Header

Preview

id	date	region	product	qty	price	sales
1	2025-01-01	North	TV	2	300	600
2	2025-01-01	East	TV	1	700	700
3	2025-01-02	North	Keyboard	2	350	700
4	205-01-03	South	TV	2	50	100
ıdera-manager.html	2025-01-03	North	printer	1	200	200

- Click **Next**.
- The table destination and properties are displayed.
- **Optional:** Set the table destination, partitions, and change the column data types.
- Verify the settings and click **Submit** to create the table.

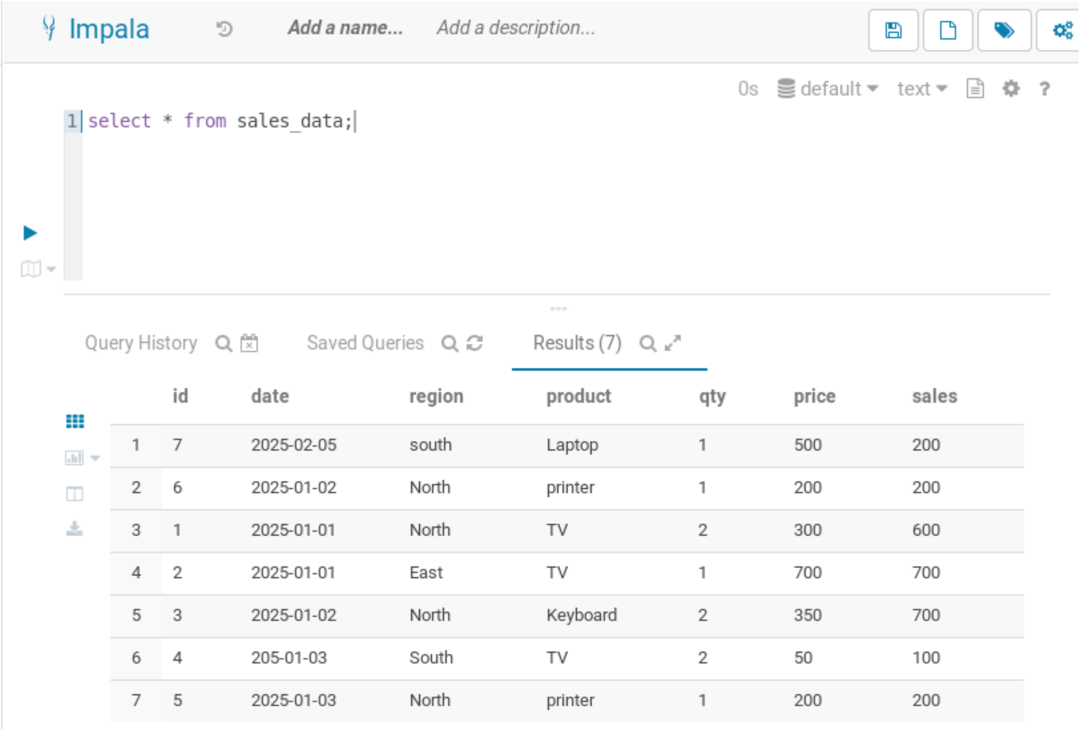
  The CREATE TABLE query is triggered.

Hue displays the logs and opens the Table Browser from which you can view the newly created table when the operation completes successfully.

2. Query and Visualize the data

To run a query:

- Click a database to view the tables it contains.
- When you click a database, it sets it as the target of your query in the main query editor panel.
- Type a query in the editor panel and click play button (to run the query).
- You can also run multiple queries by selecting them and clicking.



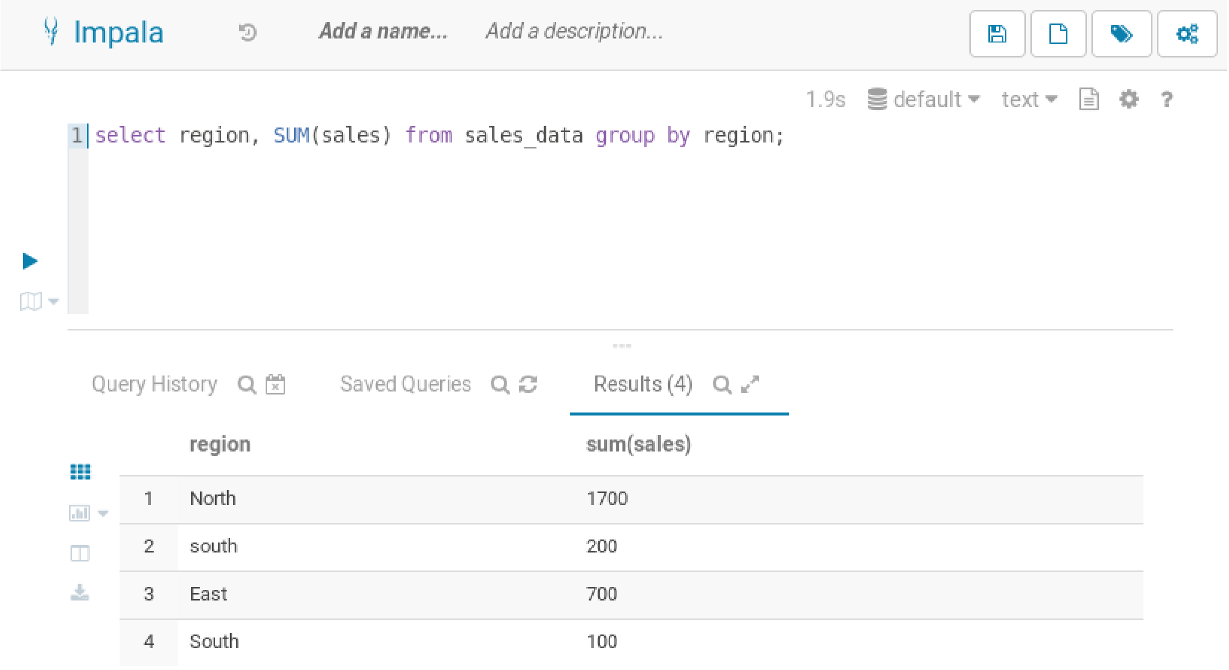
```
select * from sales_data;
```

	id	date	region	product	qty	price	sales
1	7	2025-02-05	south	Laptop	1	500	200
2	6	2025-01-02	North	printer	1	200	200
3	1	2025-01-01	North	TV	2	300	600
4	2	2025-01-01	East	TV	1	700	700
5	3	2025-01-02	North	Keyboard	2	350	700
6	4	205-01-03	South	TV	2	50	100
7	5	2025-01-03	North	printer	1	200	200

3. Reports and Charts in HUE

SQL Developers can use Hue to create data sets to generate reports and dashboards that are often consumed by other Business Intelligence (BI) tools, such as Cloudera Data Visualization.

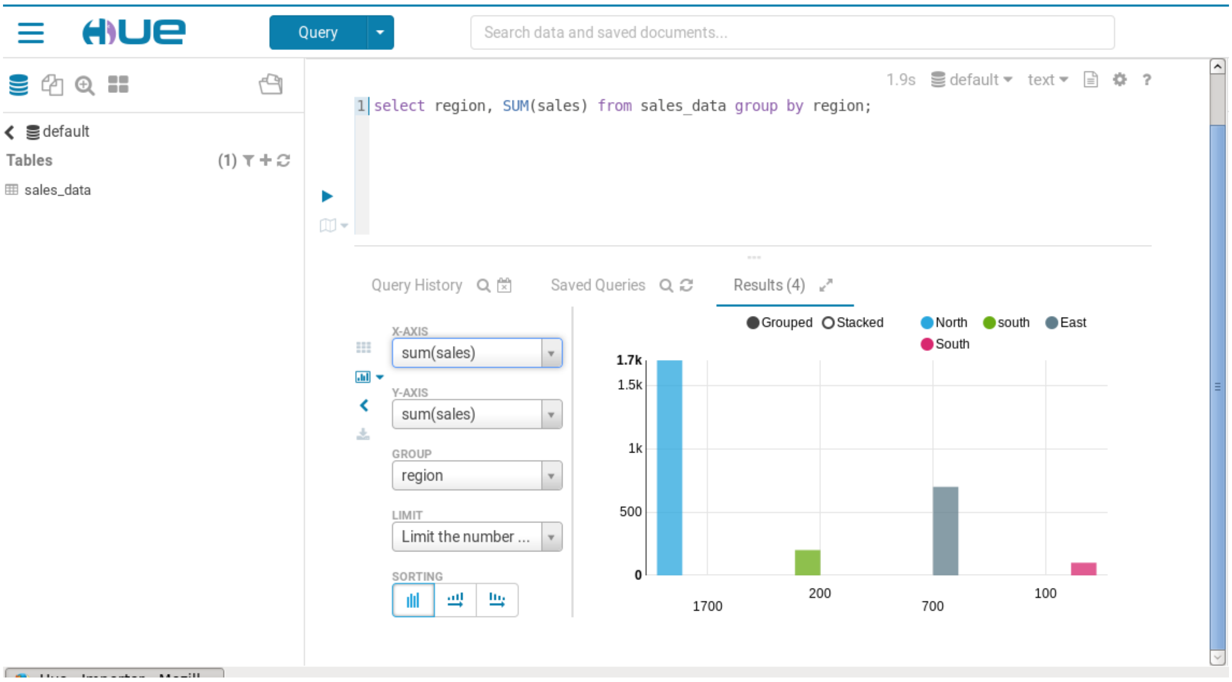
- Edit the query in the Query Editor.
- Run the query.
- Click the button below list (icon) button as shown in above screenshot.
- The output chart for the query to get the total sales region wise is shown below.



```
select region, SUM(sales) from sales_data group by region;
```

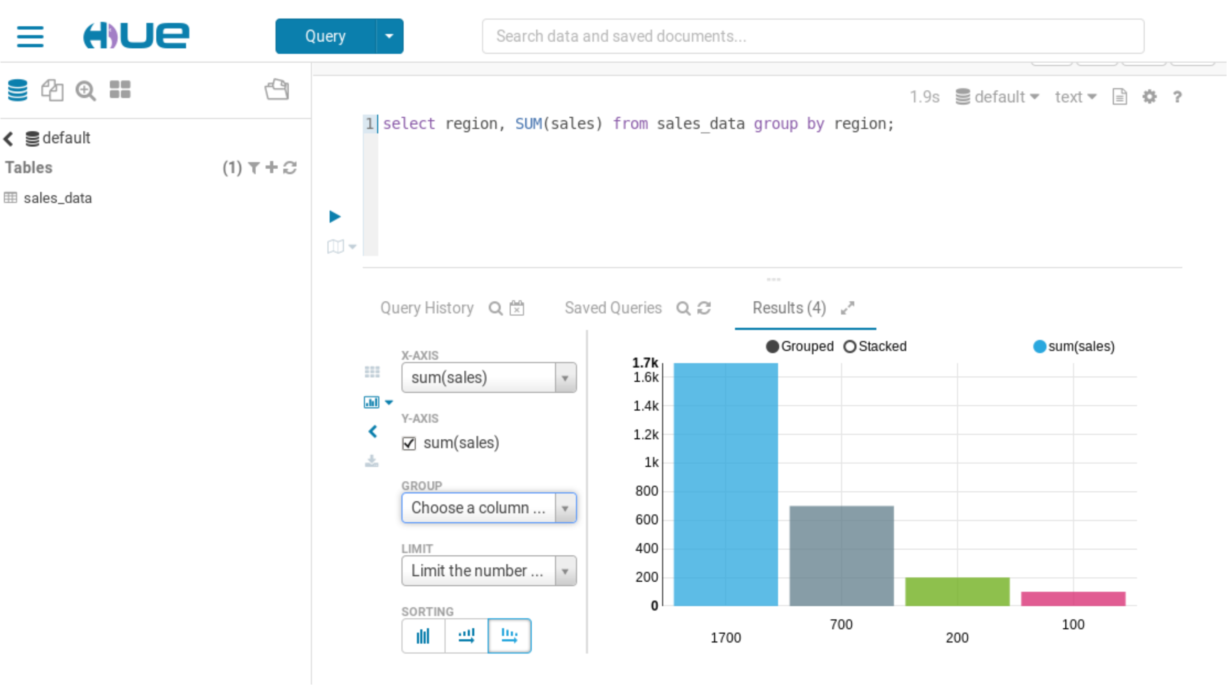
	region	sum(sales)
1	North	1700
2	south	200
3	East	700
4	South	100

- The chart got on clicking chart icon just below the list if icons in the list of icons. Set the x-axis and y-axis fields.



```
select region, SUM(sales) from sales_data group by region;
```

X-AXIS: sum(sales)
Y-AXIS: sum(sales)
GROUP: region
LIMIT: Limit the number ...



```
select region, SUM(sales) from sales_data group by region;
```

X-AXIS: sum(sales)
Y-AXIS: ☑ sum(sales)
GROUP: Choose a column ...
LIMIT: Limit the number ...