BIG DATA ANALYTICS LAB

COURSE HANDOUT

III/IV B. TECH, SEMESTER-II

R20 REGULATION

DEPARTMENT OF INFORMATION TECHNOLOGY

SIR C R REDDY COLLEGE OF ENGINEERING

Eluru-534007, West Godavari Dist, Andhra Pradesh, India

(Approved by AICTE, New Delhi & Permanently affiliated to JNTUK, Kakinada)

Accredited by NBA

Telephone No: 08812-230840, 230565, Fax: 08812-224193

Website: www.sircrrengg.ac.in

SIR C.R.REDDY COLLEGE OF ENGINEERING
ELURU-534007, ELURU DIST, A P., INDIA
(Approved by AICTE, New Delhi)
Accredited by NBA
Phone no: 08812-230840, 2300656 Fax: 08812-224193
Visit us at http://www.sircrrengg.ac.in
DEPARTMENT OF INFORMATION TECHNOLOGY

VISION OF THE DEPARTMENT

To be a premier Department in the region in the field of Information Technology through academic excellence and research that enable graduates to meet the challenges of industry and society

MISSION OF THE DEPARTMENT

- To Provide dynamic teaching-learning environment to make the students industry ready and advancement in career;
- To inculcate professional and leadership quality for better employability and entrepreneurship;
- To make high quality professional with moral and ethical values suitable for industry and society

SIR C.R.REDDY COLLEGE OF ENGINEERING
ELURU-534007, ELURU DIST, A P., INDIA
(Approved by AICTE, New Delhi)
Accredited by NBA
Phone no: 08812-230840, 2300656 Fax: 08812-224193
Visit us at http://www.sircrrengg.ac.in
DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAM OUTCOMES (POs)

PO1: Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/Development of Solutions: Design solutions for complex engineering problems and system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, society, and environmental considerations.

PO4: 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.

PO5: 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.

PO6: 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.

PO7: Environment and Sustainability: Understand the impact of the professional engineering solutions in society and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multi-disciplinary settings.

PO10: 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.

PO11: 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 multi-disciplinary environments.

PO12: Life-long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

SIR C.R.REDDY COLLEGE OF ENGINEERING
ELURU-534007, ELURU DIST, A P., INDIA
(Approved by AICTE, New Delhi)
Accredited by NBA
Phone no: 08812-230840, 2300656 Fax: 08812-224193
Visit us at http://www.sircrrengg.ac.in
DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Apply core information technology of system, architecture, information management, programming, networking for development of current technical concepts.

PSO2: Design and develop software by adopting emerging technologies for the need of IT Industry.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Solve real world problems through effective professional skills in Information Technology industry and academic research.

PEO2: Analyze and develop applications in Information Technology domain and adapt to changing technology trends with continuous learning.

PEO3: Practice the profession in society with ethical and moral values.

SIR C. R. REDDY COLLEGE OF ENGINEERING
ELURU-534007, ELURU DIST, A P., INDIA
(Approved by AICTE, New Delhi)
Accredited by NBA
Phone no: 08812-230840, 2300656 Fax: 08812-224193
Visit us at http://www.sircrrengg.ac.in
DEPARTMENT OF INFORMATION TECHNOLOGY

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
    a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.
    c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions.Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

SIR C. R. REDDY COLLEGE OF ENGINEERING
ELURU-534007, ELURU DIST, A P., INDIA
(Approved by AICTE, New Delhi)
Accredited by NBA
Phone no: 08812-230840, 2300656 Fax: 08812-224193
Visit us at http://www.sircrrengg.ac.in
DEPARTMENT OF INFORMATION TECHNOLOGY

SYLLABUS

List of Experiments:

1	**Experiment 1: Week 1, 2:**
	Implement the following Data structures in Java
	a) Linked Lists b) Stacks c) Queues d) Set e) Map
2	**Experiment 2: Week 3:**
	2. (i)Perform setting up and Installing Hadoop in its three operating modes: Standalone, Pseudo distributed, Fully distributed
	(ii)Use web based tools to monitor your Hadoop setup.
3	**Experiment 3: Week 4:**
	Implement the following file management tasks in Hadoop:
	• Adding files and directories
	• Retrieving files • Deleting files
4	**Experiment 4: Week 5:**
	Run a basic Word Count MapReduce program to understand MapReduce Paradigm.
5	**Experiment 5: Week 6:**
	Write a map reduce program that mines weather data. Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record-oriented.
6	**Experiment 6: Week 7:**
	Use MapReduce to find the shortest path between two people in a social graph. Hint: Use an adjacency list to model a graph, and for each node store the distance from the original node, as well as a back pointer to the original node. Use the mappers to propagate the distance to the original node, and the reducer to restore the state of the graph. Iterate until the target node has been reached.

7	**Experiment 7: Week 8:**
	Implement Friends-of-friends algorithm in MapReduce. Hint: Two MapReduce jobs are required to calculate the FoFs for each user in a social network .The first job calculates the common friends for each user, and the second job sorts the common friends by the number of connections to your friends.
8	**Experiment 8: Week 9:**
	Implement an iterative PageRank graph algorithm in MapReduce. Hint: PageRank can be implemented by iterating a MapReduce job until the graph has converged. The mappers are responsible for propagating node PageRank values to their adjacent nodes, and the reducers are responsible for calculating new PageRank values for each node, and for re-creating the original graph with the updated PageRank values.
9	**Experiment 9: Week 10:**
	Perform an efficient semi-join in MapReduce. Hint: Perform a semi-join by having the mappers load a Bloom filter from the Distributed Cache, and then filter results from the actual MapReduce data source by performing membership queries against the Bloom filter to determine which data source records should be emitted to the reducers.
10	**Experiment 10: Week 11:**
	Install and Run Pig then write Pig Latin scripts to sort, group, join, project, and filter your data.
11	**Experiment 11: Week 12:**
	Install and Run Hive then use Hive to create, alter, and drop databases, tables, views, functions, and indexes.

Software Configuration and Installation

Java is the primary requirement for running Hadoop on any system, So make sure you have Java installed on your system using the following command. If you don't have Java installed on your system, use one of the following links to install it first. Hadoop supports only JAVA 8 If already any other version is present then uninstall the following using these commands.

```
sudo apt-get purge openjdk-\* icedtea-\* icedtea6-\*
OR
sudo apt remove openjdk-8-jdk
```

Step 1.1 – Install Oracle Java 8 on Ubuntu

You need to enable additional repository to your system to install Java 8 on Ubuntu VPS. After that install Oracle Java 8 on an Ubuntu system using apt-get.This repository contains a package

named oracle-java8-installer, Which is notan actual Java package. Instead of that, this package contains a script toinstall Java on Ubuntu.Run below commands to install Java 8 on Ubuntu and LinuxMint.

sudo add-apt-repository ppa:webupd8team/java sudo apt-get

sudo apt-get install oracle-java8-installer

OR

sudo apt install openjdk-8-jre-headless sudo apt install openjdk-8-jdk

Step 1.2 – Verify Java Inatallation

The apt repository also provides package oracle-java8-set-default to set Java 8 as your default Java version. This package will be installed along with Java installation. To make sure run below command.

sudo apt-get install oracle-java8-set-default

After successfully installing Oracle Java 8 using the above steps, Let's verify the installed version using the following command.

java -version

java version "1.8.0_201"

ReferenceBooks:

1. Glenn J. Myatt, Making Sense of Data , John Wiley & Sons, 2007 Pete Warden, Big Data Glossary, O'Reilly, 2011.
2. Michael Berthold, David J.Hand, Intelligent Data Analysis, Spingers, 2007.
3. Chris Eaton, Dirk DeRoos, Tom Deutsch, George Lapis, Paul Zikopoulos, Uderstanding Big Data : Analytics for Enterprise Class Hadoop and Streaming Data, McGrawHill Publishing, 2012.
4. AnandRajaraman and Jeffrey David UIIman, Mining of Massive Datasets Cambridge University Press, 2012.

WebLinks:

www.data.gov.in
www.nseindia.in
www.censusindia.gov.in
www.importexportdata.in

SIR C. R. REDDY COLLEGE OF ENGINEERING
ELURU-534007, WEST GODAVARI DIST, A P., INDIA
(Approved by AICTE, New Delhi)
Accredited by NBA
Phone no: 08812-230840, 2300656 Fax: 08812-224193
Visit us at http://www.sircrrengg.ac.in
DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE OUTCOMES

At the end of the course students should be able to:

COs	OUTCOMES	LEVEL
1	**Understand** the basics of data structures like Linked list, stack, queue, set, map in Java and installation of Hadoop, Pig, Hive.	**L2**
2	**Demonstrate** the knowledge of big data analytics in different file management task in Hadoop.	**L3**
3	**Implement** Map Reduce programs in variety applications.	**L3**
4	**Apply** different operations on relations and databases using Hive.	**L3**
5	**Analyze** and perform different operations on data using Pig Latin scripts.	**L4**

CO-PO/PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1	2	1	-	2	-	-	-	-	-	-	-	2	2	2
CO2	2	1	1	2	-	-	-	-	-	-	-	3	2	2
CO3	2	2	1	1	-	-	-	-	-	-	-	2	3	2
CO4	2	-	1	2	-	-	-	-	-	-	-	3	3	2
C05	2	-	1	2	-	-	-	-	-	-	-	3	3	2
AVG	**2**	**1.3**	**1**	**1.8**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**2.6**	**2.6**	**2**

SIR C. R. REDDY COLLEGE OF ENGINEERING
ELURU-534007, WEST GODAVARI DIST, A P., INDIA
(Approved by AICTE, New Delhi)
Accredited by NBA
Phone no: 08812-230840, 2300656 Fax: 08812-224193
Visit us at http://www.sircrrengg.ac.in
DEPARTMENT OF INFORMATION TECHNOLOGY

CO-PO MAPPING JUSTIFICATION

CO1	**PO1**	**2**	Moderately having the Knowledge of the fundamental concepts of Big Data management and analytics that helps in solving complex engineering problems.
	PO2	**1**	Slightly having the knowledge of Big Data management and analytics concepts which can be used to design and conduct experiments.
	PO4	**2**	Moderately the student using the Statistical concepts and inferences can be used to design and conduct experiments to provide valid conclusions.
	PO12	**2**	The student will become aware of the need for lifelong learning and upgrading of technical knowledge.
	PSO1	**2**	Moderately the student will study fundamental concepts of Big Data management and analytics acquire skills to design, analyse and develop algorithms and implement them using high-level programming languages .
	PSO2	**2**	The student will have the knowledge of data analytics concepts that will contribute to skills in computing and knowledge engineering domain.
CO2	**PO1**	**2**	Moderately the student will have the knowledge of theoretical foundations of text and web mining that involves solving complex engineering problems.
	PO2	**1**	Slightly the student knows the principles of mathematics and engineering sciences that are used in theoretical foundations of text and web mining techniques.

	PO3	**1**	Slightly the student will have the knowledge of theoretical foundations of text and web mining that can be used to design and develop solutions for complex engineering problems.
	PO4	**2**	Moderately the Student will have Theoretical foundations of frequent web mining knowledge can be used to design and conduct experiments to provide valid conclusions.
	PO12	**3**	The student will become aware of the need for lifelong learning and the continued upgrading of technical knowledge.
	PSO1	**2**	Moderately the student will learn Theoretical foundations of the text and web mining and acquire skills to design, analyze and develop algorithms and implement those using high-level programming languages.
	PSO2	**2**	Moderately the student will learn Theoretical foundations of the text,web mining techniques that contribute to skill in computing and knowledge engineering domain
CO3	**PO1**	**2**	Moderately having Knowledge of Map Reduce paradigm and ApacheHadoop system that involves solving complex engineering problems.
	PO2	**2**	Moderately the student will know Principles of mathematics and engineering sciences that are used in various aspects of data analytic approaches.
	PO3	**1**	Moderately having Knowledge of Map Reduce paradigm and the Hadoop system that can be used to conduct experiments in real life problems to provide valid conclusions.
	PO4	**1**	Slightly the student will study the statistical concepts and inferences that can be used to design and conduct experiments to provide valid conclusions.
	PO12	**2**	Moderately the student will become aware of the need for lifelong learning and the continued upgrading of technical knowledge.
	PSO1	**3**	Highly the student will study fundamental concepts of Hadoop Map Reduce and Hadoop administration to acquire skills to design, analyze and develop algorithms and implement them

			using high-level programming languages
	PSO2	**2**	Moderately the student will have the knowledge of data analytics concepts that contribute to skills in computing and knowledge engineering domain.
CO4	**PO1**	**2**	Moderately the student will have the Knowledge of various data analytic approaches that involve solving complex engineering problems.
	PO3	**1**	Slightly the student using the knowledge of statistical concepts, can design and develop solutions for complex engineering problems.
	PO4	**2**	Moderately the student using the Statistical concepts and inferences can be used to design and conduct experiments to provide valid conclusions.
	PO12	**3**	The student will become aware of the need for lifelong learning and the continued upgrading of technical knowledge.
	PSO1	**3**	Highly the student will study the fundamental concepts of Business Intelligence, Data Mining, Data Warehousing, Data visualization and acquire skills to design, analyze and develop algorithms and implement them using high-level programming language.
	PSO2	**2**	The student will have the knowledge of data analytics concepts that contribute to skills in computing and knowledge engineering domain.
CO5	**PO1**	**2**	Moderately the student will have the knowledge of theoretical foundations of text and web mining that involves solving complex engineering problems.
	PO3	**1**	Slightly the student will have the knowledge of theoretical foundations of text, web mining that can be used to design and develop solutions for complex engineering problems.
	PO4	**2**	Moderately the Student will have Theoretical foundations of frequent web mining knowledge that can be used to design and conduct experiments to provide valid conclusions.

	PO12	**3**	The student will become aware of the need for lifelong learning and the continued upgrading of technical knowledge.
	PSO1	**3**	Highly the student will learn various data analytic approaches such as decision tree, neural networks and acquire skills to design, analyze and develop algorithms and implement them using high-level programming language.
	PSO2	**2**	Moderately the student will learn Different Classification concepts that contribute to skill in computing and knowledge engineering domain.

SIR C. R. REDDY COLLEGE OF ENGINEERING
ELURU-534007, ELURU DIST, A P., INDIA
(Approved by AICTE, New Delhi)
Accredited by NBA
Phone no: 08812-230840, 2300656 Fax: 08812-224193
Visit us at http://www.sircrrengg.ac.in
DEPARTMENT OF INFORMATION TECHNOLOGY

INDEX

S.No	Name of Experiment	CO
1	1. Implement the following Data structures in Java a) Linked Lists b) Stacks c) Queues d) Set e) Map	CO1
2	(i)Perform setting up and Installing Hadoop in its three operating modes: Standalone, Pseudo distributed, Fully distributed (ii)Use web based tools to monitor your Hadoop setup.	CO1
3	Implement the following file management tasks in Hadoop: Adding files and directories, Retrieving files, Deleting files	CO3
4	Run a basic Word Count MapReduce program to understand MapReduce Paradigm.	CO3
5	Write a map reduce program that mines weather data.	CO3
6	Use MapReduce to find the shortest path between two people in a social graph.	CO3
7	Implement Friends-of-friends algorithm in MapReduce.	CO3
8	Implement an iterative PageRank graph algorithm in MapReduce.	CO3
9	Perform an efficient semi-join in MapReduce.	CO3
10	Install and Run Pig then write Pig Latin scripts to sort, group,	CO1

	join, project, and filter your data.	& CO5
11	Install and Run Hive then use Hive to create, alter, and drop databases, tables, views, functions, and indexes.	CO1 & CO4
	Content Beyond the Syllabus	
12	Write a MapReduce Program for sorting following data according to given data: 001 chp, 002 vr, 003 pnr, 004 prp	CO3
13	Write Pig Latin Script Operators like Distinct, For-each, Union	CO5

EXPERIMENT -1

DATE:

Aim: Implement the following Data structures in Java

a) **Linked Lists**
b) **Stacks**
c) **Queues**
d) **Set**
e) **Map**

Program:

```
import java.util.*;
public class LinkedList1
{
	public static void main(String args[])
	{
		LinkedList<String> al=new LinkedList<String>();
		al.add("Ravi");
		al.add("Vijay");
		al.add("Ravi");
		al.add("Ajay");
		Iterator<String> itr=al.iterator();
		while(itr.hasNext())
		{
			System.out.println(itr.next());
		}
	}
}
```

Output:

Ravi

Vijay

Ravi

Ajay

b) Stacks

Program:

```
import java.io.*;
import java.util.*;
class StackDemo
{
	public static void main(String[] args)
	{
		Stack stack1 = new Stack();
		Stack<String> stack2 = new Stack<String>();
		stack1.push("Nakka");
		stack1.push("Sri Manohar");
		stack1.push("Reddy");
		System.out.println(stack1);
	}
}
```

Output:

[Nakka, Sri Manohar, Reddy]

c) Queues

Program:

```
import java.util.Queue;
import java.util.LinkedList;
class Main
{
   public static void main(String[] args)
        {
            Queue<Integer> numbers = new LinkedList<>();
            numbers.offer(1);
            numbers.offer(2);
            numbers.offer(3);
            System.out.println("Queue: " + numbers);
            numbers.poll();
            System.out.println("Queue after dequeue: " + numbers);
        }
}
```

Output:

Queue: [1, 2, 3]

Queue after dequeue: [2, 3]

d) Sets

Program:

```
import java.util.Set;
import java.util.HashSet;
public class SetInterfaceExample
 {
   public static void main(String[] args)
        {
          Set<Integer> numbers = new HashSet<>();
          numbers.add(10);
          numbers.add(20);
          numbers.add(30);
          System.out.println("Contains 20: " + numbers.contains(20));
          numbers.remove(30);
          System.out.println("Numbers Set: " + numbers);
        }
}
```

Output:

Contains 20: true

Numbers Set: [10, 20]

e) Map

Program:

```
import java.util.Map;
import java.util.TreeMap;
public class MapInterfaceExample
{
   public static void main(String[] args)
        {
            Map<String, String> countryCodes = new TreeMap<>();
            countryCodes.put("USA", "+1");
            countryCodes.put("India", "+91");
            countryCodes.put("UAE", "+97");
            System.out.println("Contains key 'India': " + countryCodes.containsKey("India"));
            countryCodes.remove("USA");
            System.out.println("Country Codes: " + countryCodes);
          }
}
```

Output:

Contains key 'India': true

Country Codes: {India=+91, UAE=+97}

EXPERIMENT -2

DATE:

Aim: (i) Perform setting up and Installing Hadoop in its three operating modes: Standalone, Pseudo distributed, fully distributed.

1. Standalone Mode:

In Standalone Mode, Hadoop operates as a single monolithic Java process, beneficial for debugging and initial testing of MapReduce applications with small datasets. Here's a revised breakdown of the setup process:

1. **Download Hadoop Binary Package**:
   - Visit the Hadoop releases page and download the latest binary package.
   - Extract the downloaded package to a directory (e.g., `C:\hadoop-3.1.2`).
2. **Set Environment Variables**:
   - Set the following environment variables:
     - `HADOOP_HOME`: Path to the Hadoop installation directory (e.g., `C:\hadoop-3.1.2`).
     - `HADOOP_BIN`: Path to the Hadoop binary directory (e.g., `C:\hadoop-3.1.2\bin`).
     - `JAVA_HOME`: Path to your JDK installation (ensure it's version 1.8)
3. **Edit PATH Environment Variable**:
   - Add `%HADOOP_HOME%`, `%HADOOP_BIN%`, and `%JAVA_HOME%\bin` to your system's PATH variable.
4. **Create Folders for Datanode and Namenode**:
   - Inside the Hadoop installation directory (`C:\hadoop-3.1.2`), create an `input` folder.
   - Inside the `input` folder, create two subfolders: `datanode` and `namenode`.
5. **Configure Hadoop**:
   - Edit the Hadoop configuration files (e.g., `core-site.xml`, `hdfs-site.xml`, etc.) to specify the paths and settings.
6. **Run Hadoop Jobs**:
   - You can now run Hadoop jobs in Standalone Mode.

2. Pseudo-distributed Mode:

In Pseudo-Distributed Mode, Hadoop emulates a distributed environment on a single machine, running each daemon as a separate Java process. Hadoop software is installed on a Single Node, where various daemons of Hadoop run on the same machine as separate Java processes. These daemons include NameNode, DataNode, SecondaryNameNode, JobTracker, and TaskTracker, all operating within the confines of a single server.

1. **Repeat Steps 1-4 from Standalone Mode.**
2. **Configure Hadoop**:
   - Edit the Hadoop configuration files to set the appropriate values for the pseudo-distributed environment.
   - Specify `localhost` or `127.0.0.1` as the hostname for all daemons.
3. **Start Hadoop Services**:
   - Start the Hadoop daemons (Namenode, Datanode, ResourceManager, NodeManager, etc.) using the appropriate scripts.

3. Fully Distributed Mode:

In Fully Distributed Mode, the daemons NameNode, JobTracker, SecondaryNameNode (Optional and can be runon a separate node) run on the Master Node. The daemons DataNode and TaskTracker run on the Slave Node. In Fully Distributed Mode, Hadoop runs on a cluster of machines. Each machine hosts one or more Hadoop daemons. Follow these steps:

1. **Repeat Steps 1-4 from Standalone Mode**.
2. **Configure Hadoop**:
    - Edit the Hadoop configuration files to set the values specific to your cluster.
    - Specify the actual hostnames or IP addresses of the machines in the cluster.
3. **Start Hadoop Services**:
    - Start the Hadoop daemons on each machine in the cluster.
4. **Distribute Configuration Files**:
    - Copy the modified configuration files to all machines in the cluster.
5. **Run Hadoop Jobs**:
    - Submit Hadoop jobs to the cluster.

 # For core-site.xml

```
<property>

  <name>fs.defaultFS</name>

  <value>hdfs://localhost:9000</value>

</property>
```

 # For hdfs-site.xml or https-site.xml

```
<property>

  <name>dfs.replication</name>

  <value>1</value>

</property><property>

  <name>dfs.namenode.name.dir</name>

  <value>C:\hadoop\data\namenode</value>

</property><property>
```

```
  <name>dfs.datanode.data.dir</name>
  <value>C:\hadoop\data\datanode</value>
</property>
```

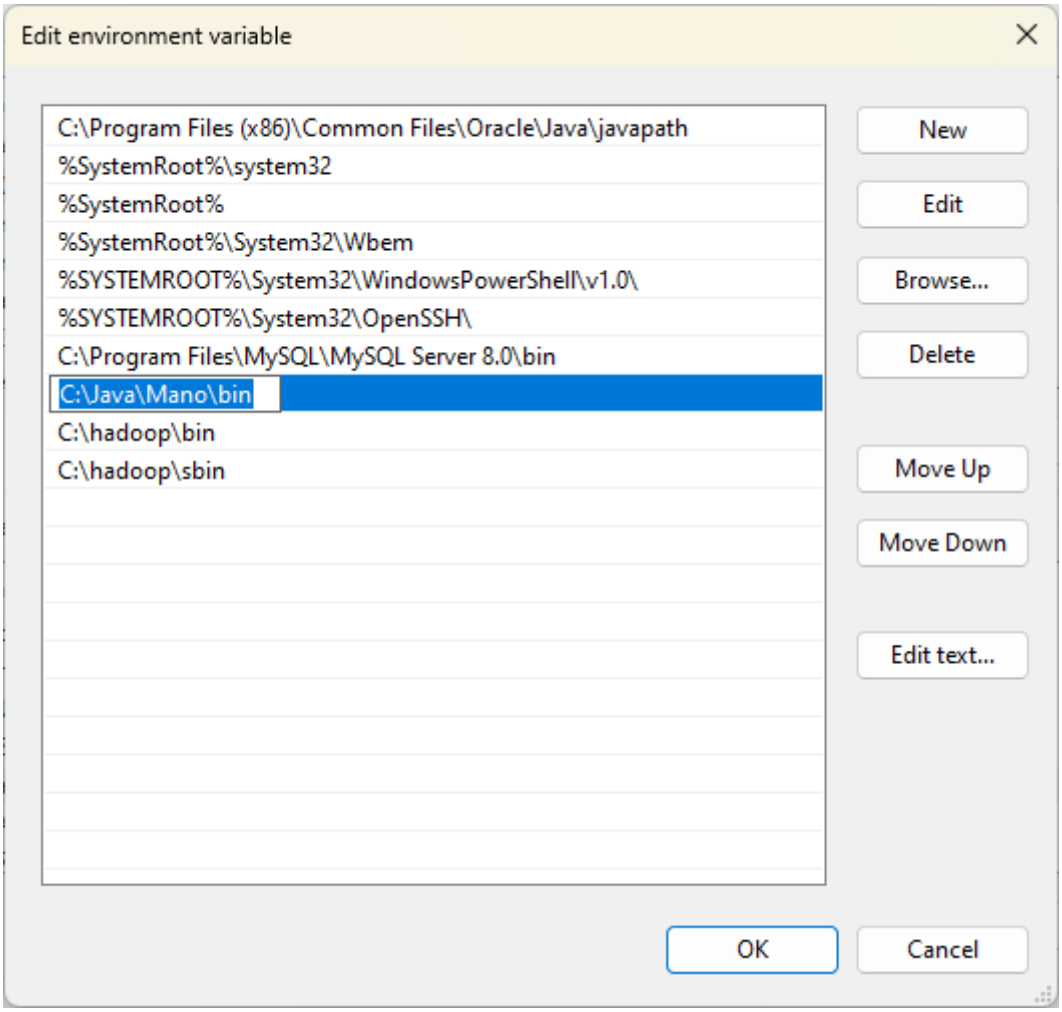
 # For mapred-site.xml

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

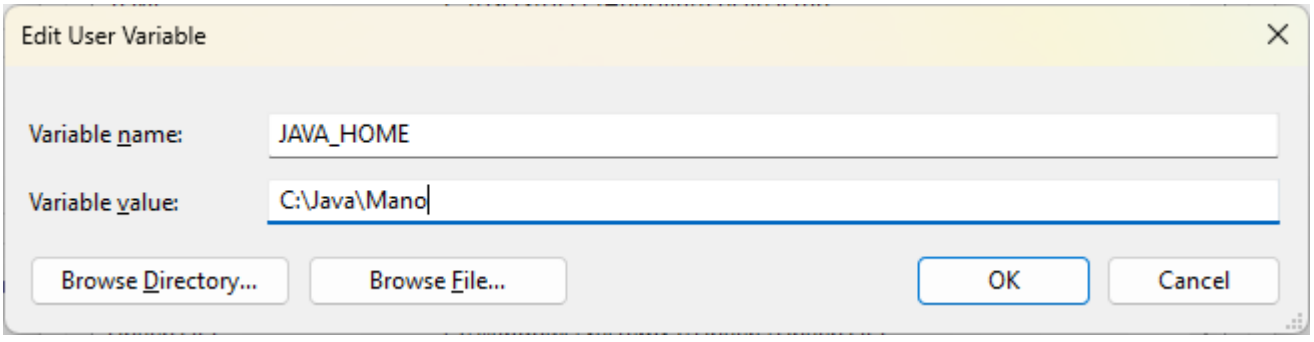
 # For yarn-site.xml

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property><property>
  <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

Setting up Java Environment for Hadoop:



Configuring Hadoop:



core-site.xml

File Edit View

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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

Ln 1, Col 1 | 864 characters | 100% | Unix (LF) | UTF-8

Fig. Core Site.xml

core-site.xml hdfs-site.xml

File Edit View

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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

Ln 1, Col 1 | 1,047 characters | 100% | Unix (LF) | UTF-8

Fig. hdfs-site.xml

core-site.xml hdfs-site.xml mapred-site.xml

File Edit View

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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

Ln 1, Col 1 842 characters 100% Unix (LF) UTF-8

Fig. mapred-site.xml

core-site.xml hdfs-site.xml mapred-site.xml yarn-site.xml

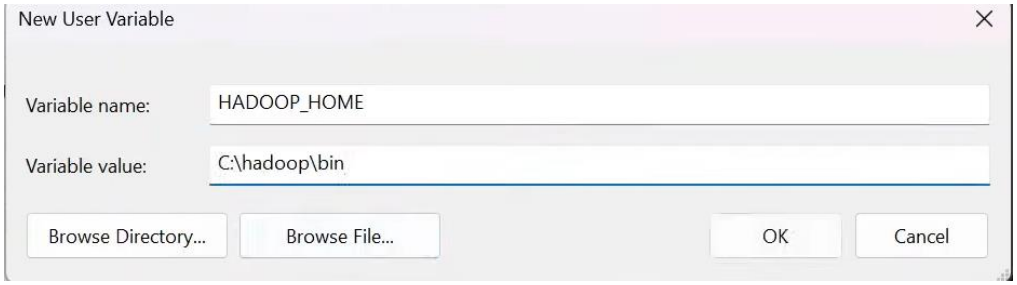
File Edit View

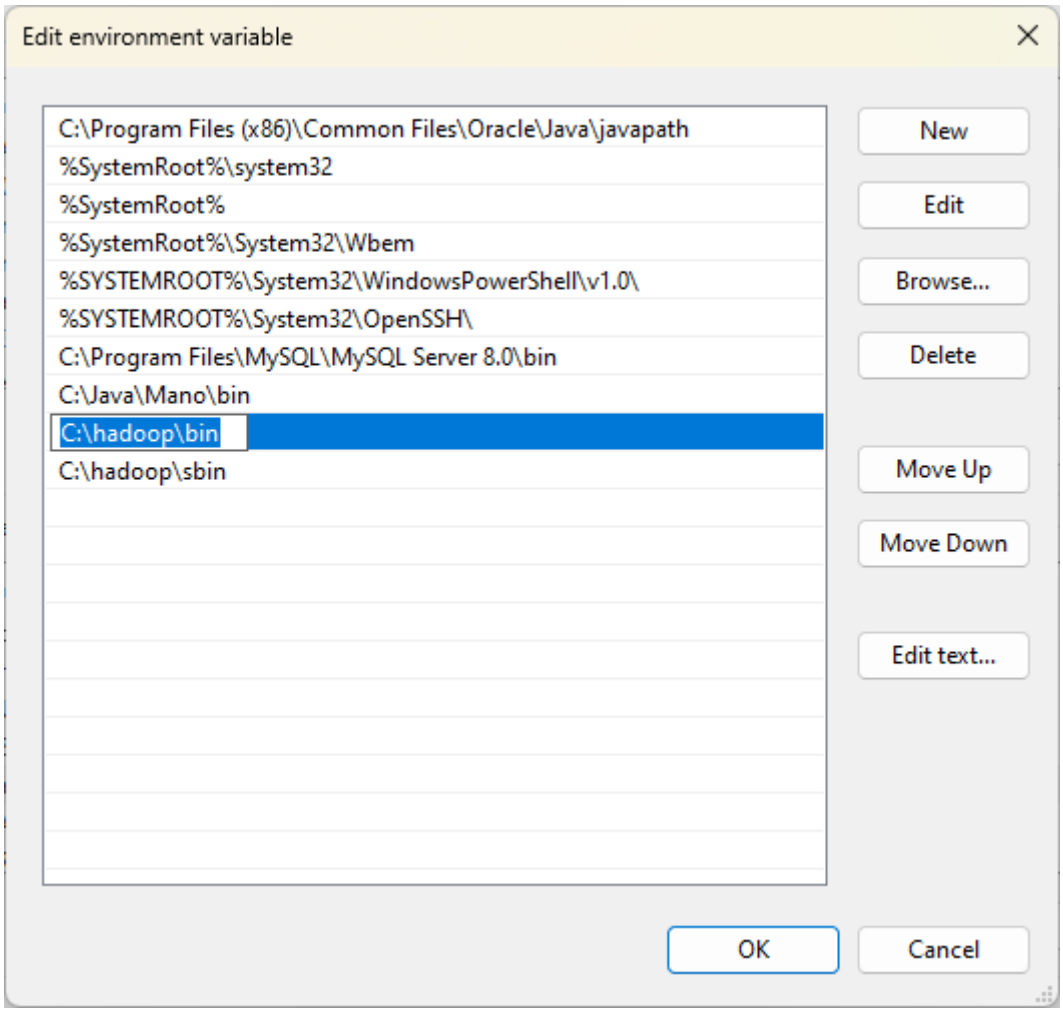
```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property><property>
  <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

Ln 1, Col 1 939 characters 100% Unix (LF) UTF-8

Fig. yarn-site.xml

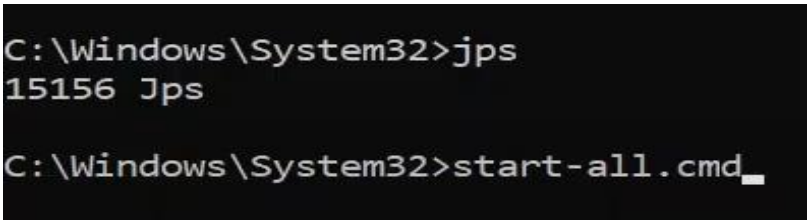
Setting up Hadoop Environment:





Starting the Hadoop Server:

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



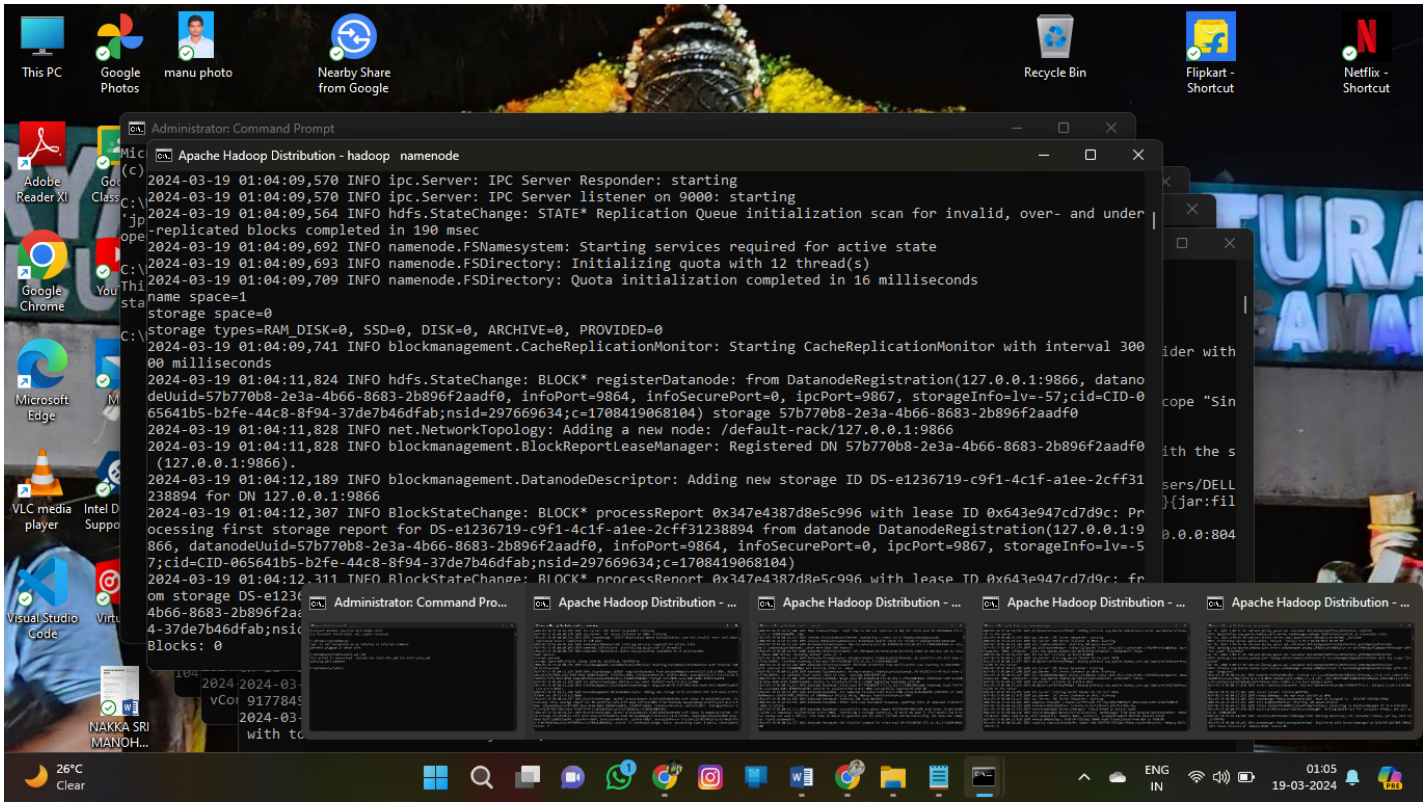


Fig.Demons Running

- 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 accesing 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

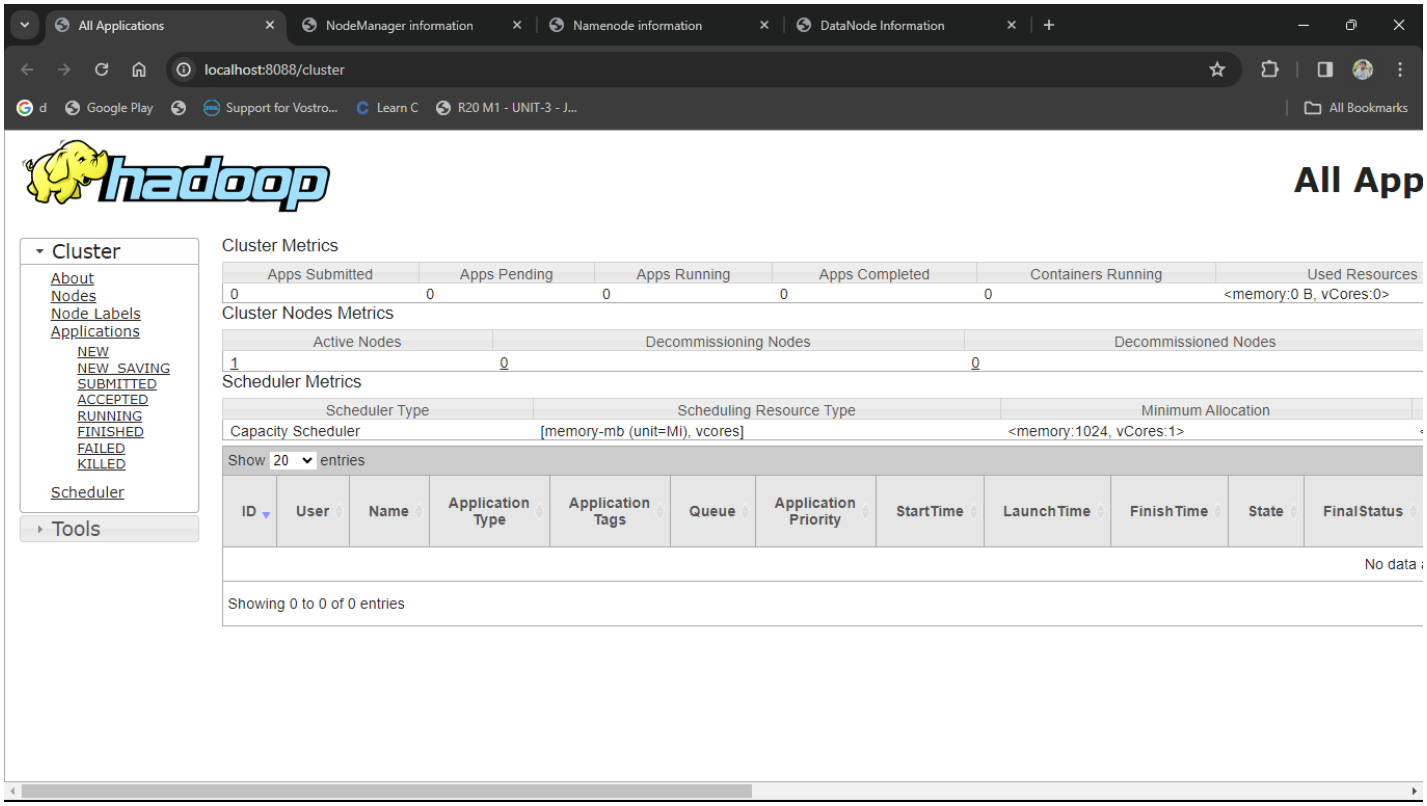


Fig. Applications

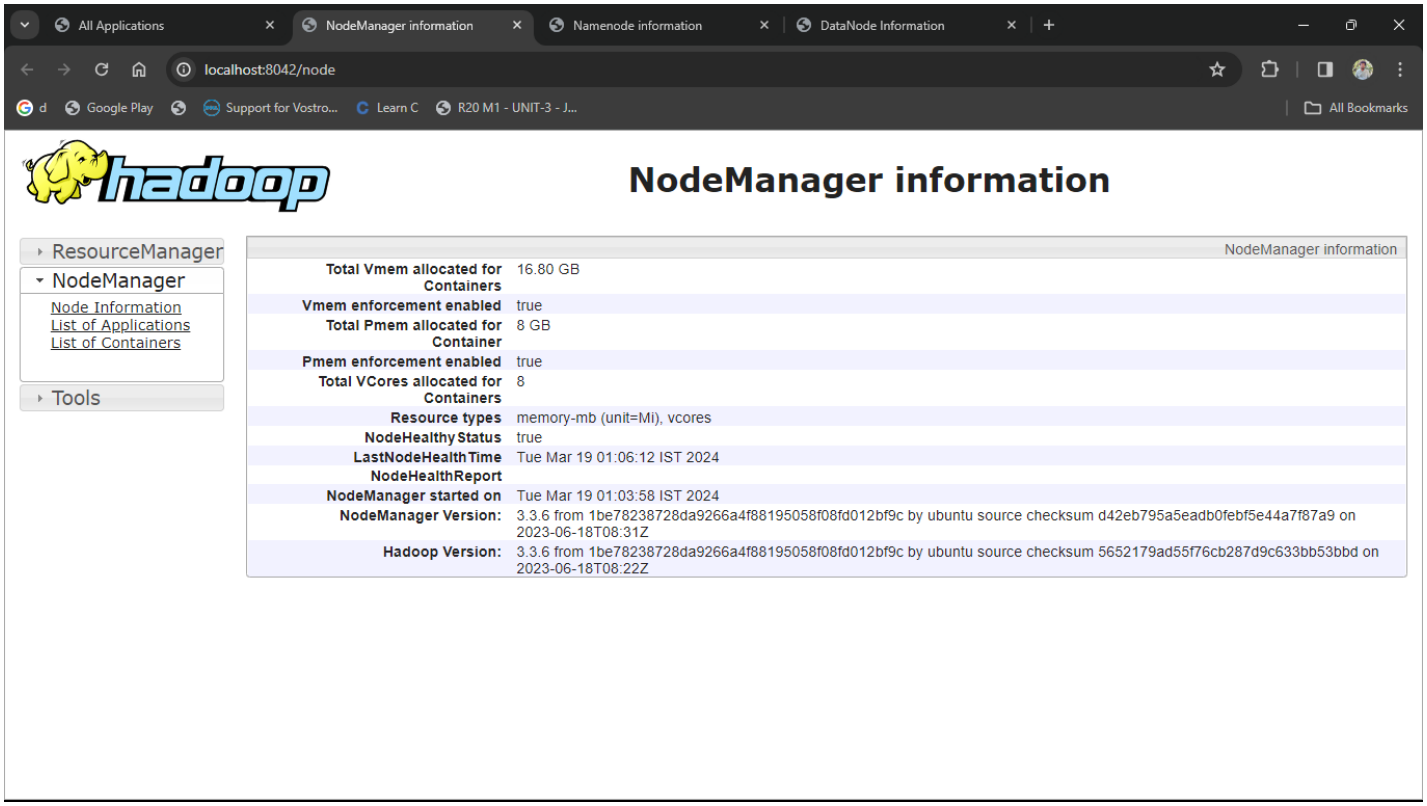


Fig. NodeManager information

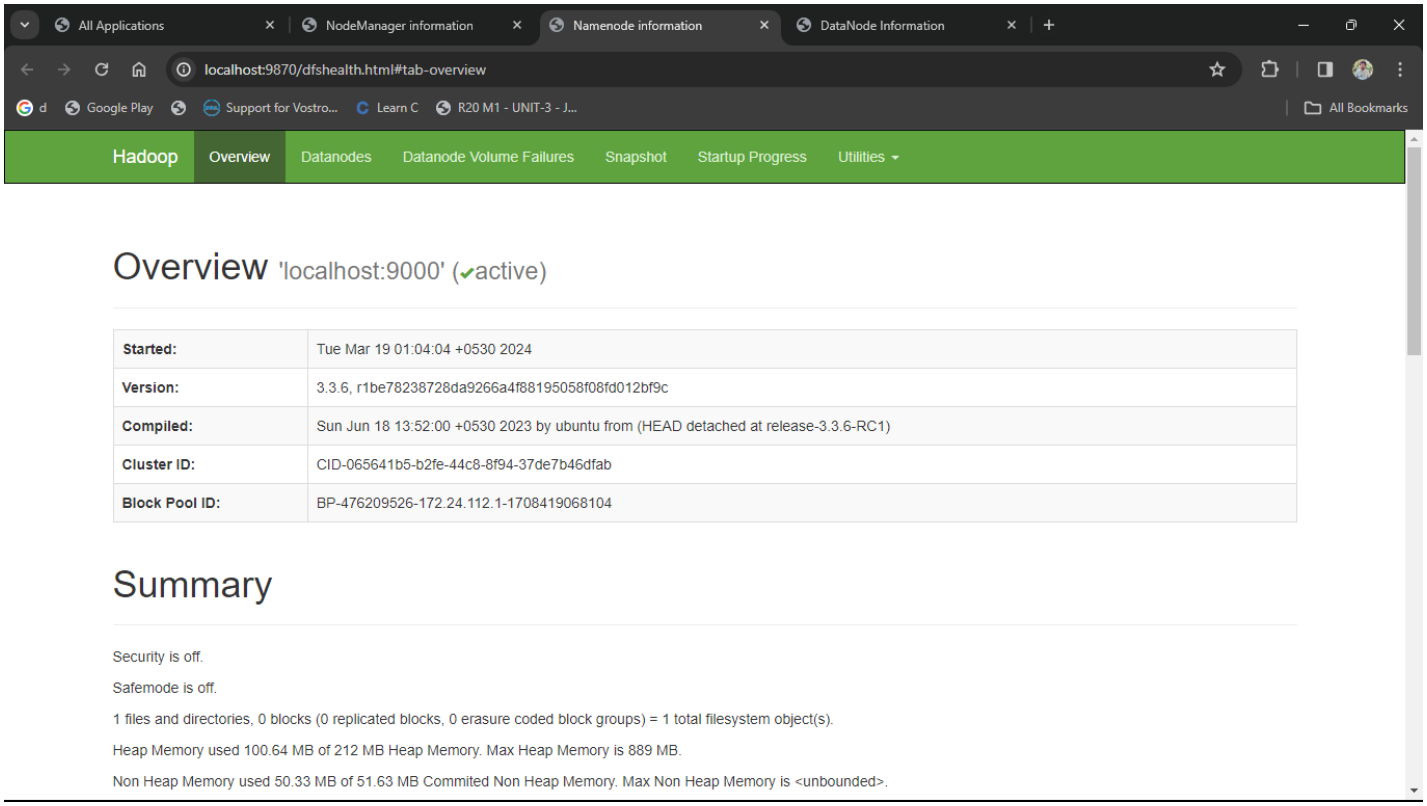


Fig. Namenode

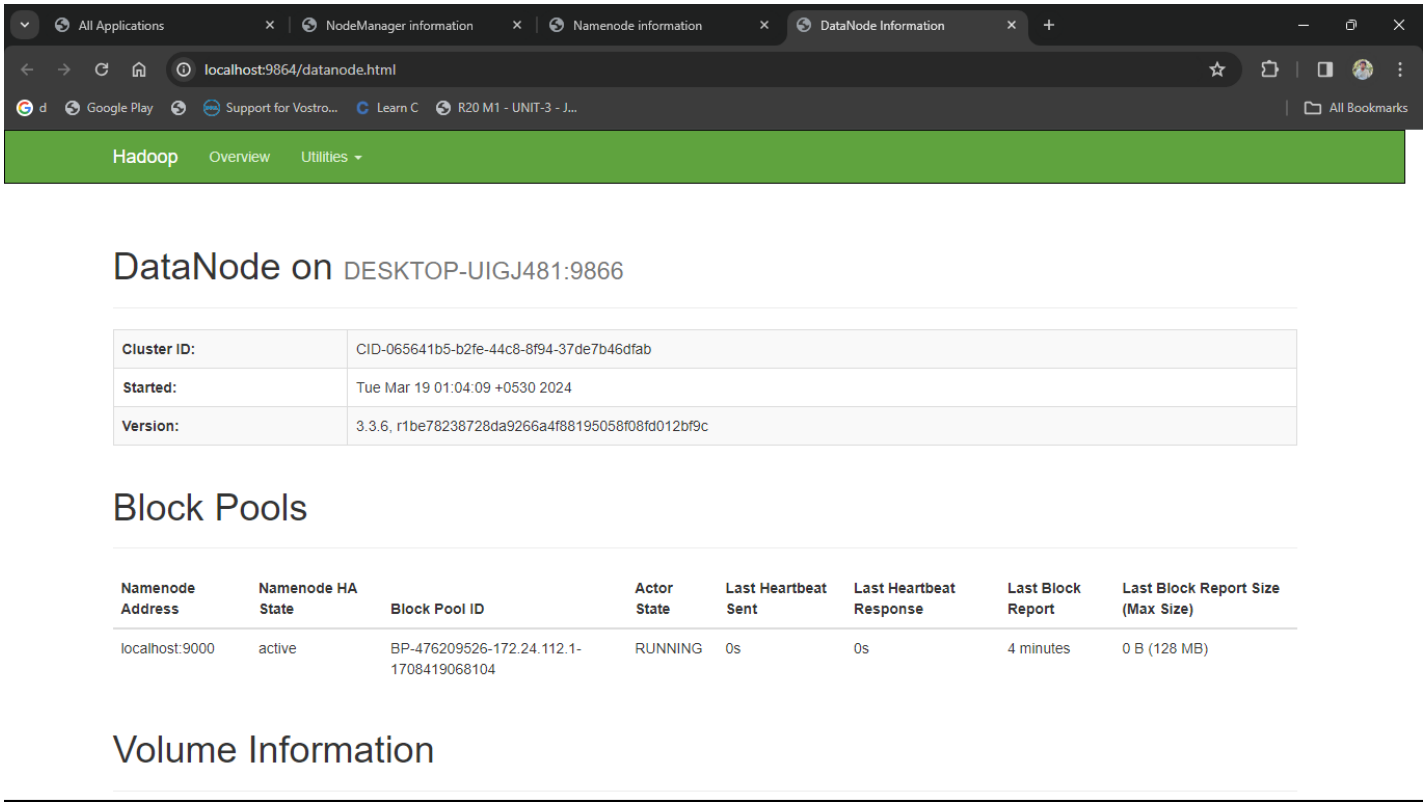


Fig. DataNode

EXPERIMENT -3

DATE:

Aim: Implement the following file management tasks in Hadoop:

- **Adding files and directories**
- **Retrieving files**
- **Deleting files**

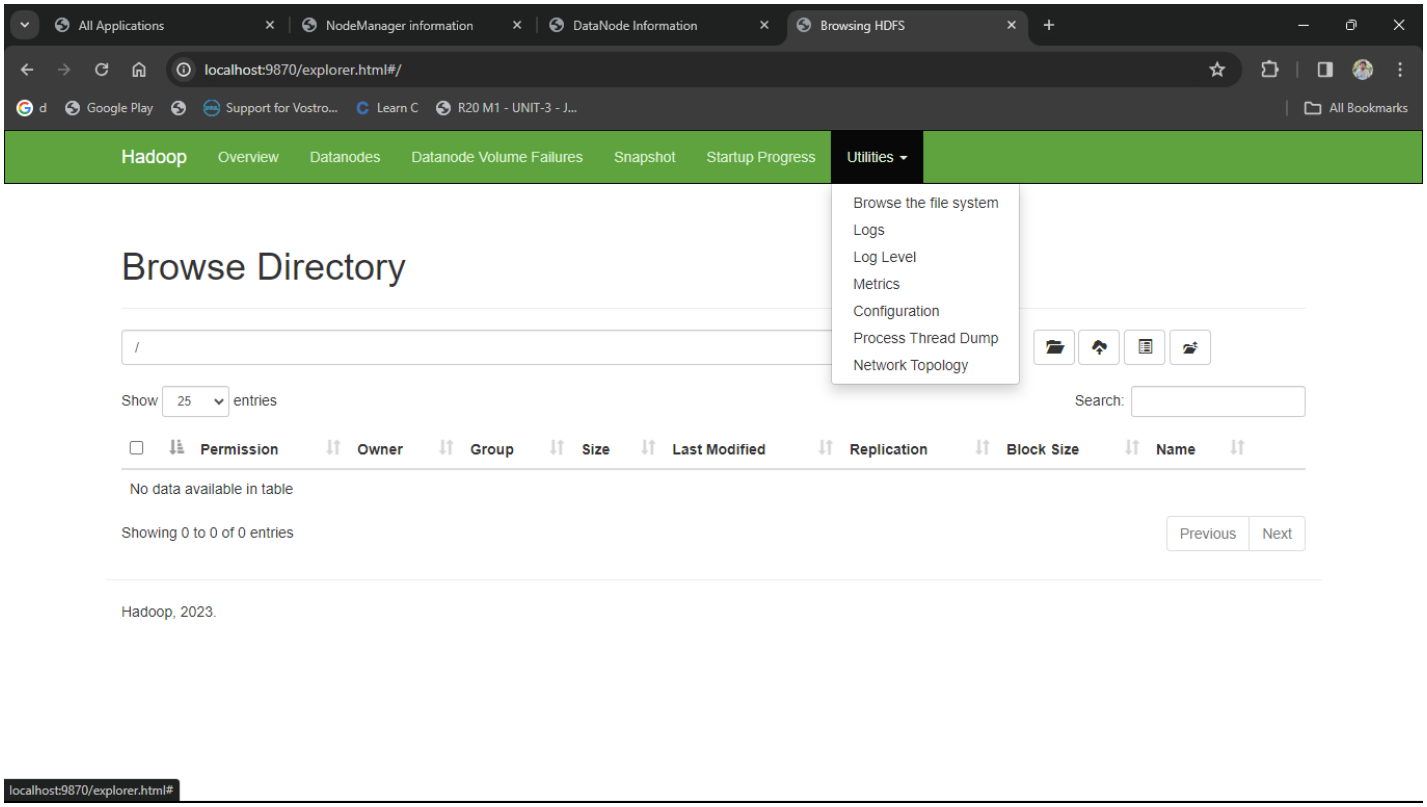
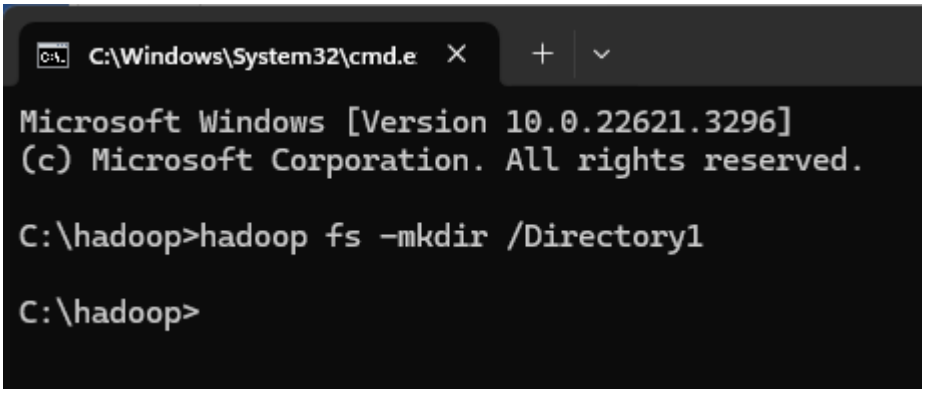


Fig. Browsing HDFS

- Create a directory in HDFS at given path(s)

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



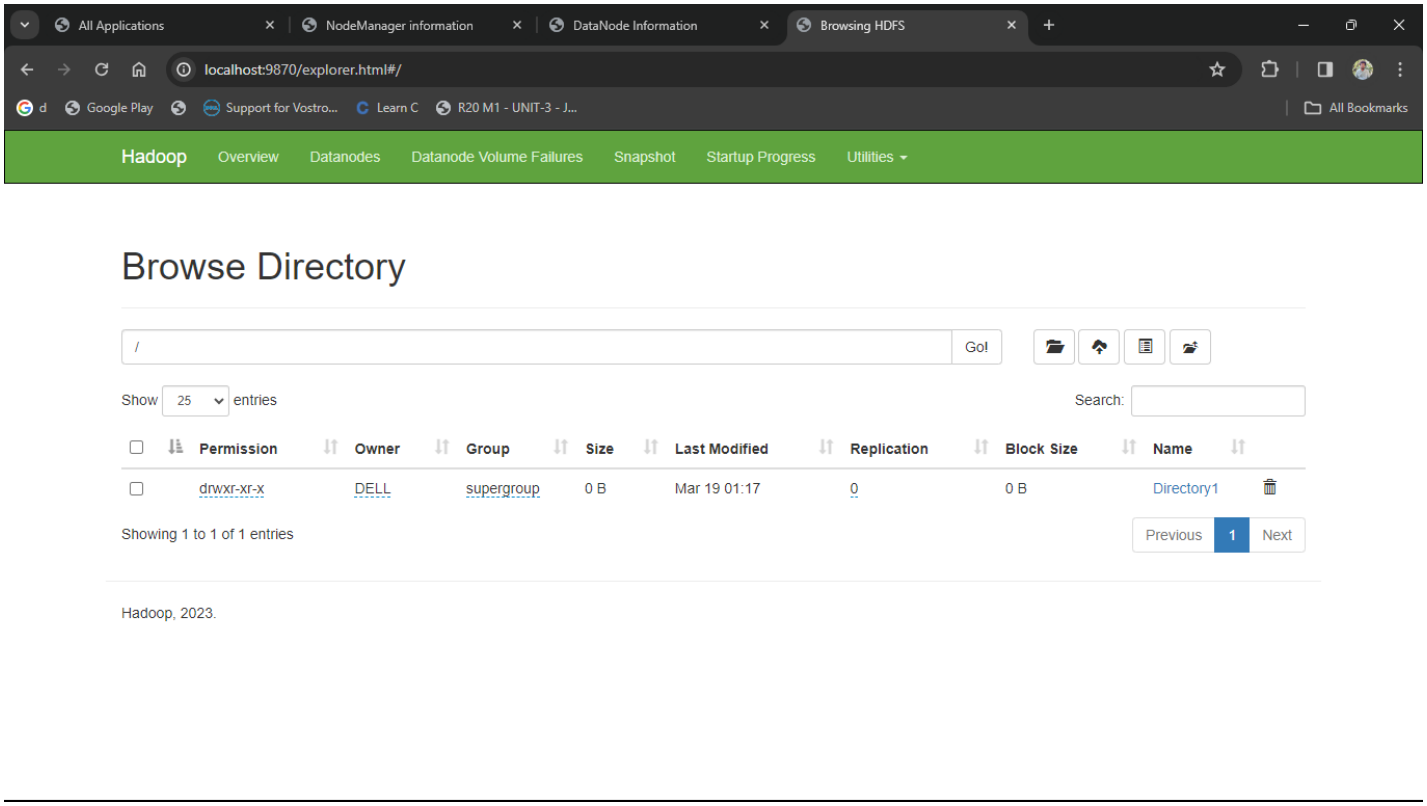


Fig. Directory1 Created

- List the contents of a directory.

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

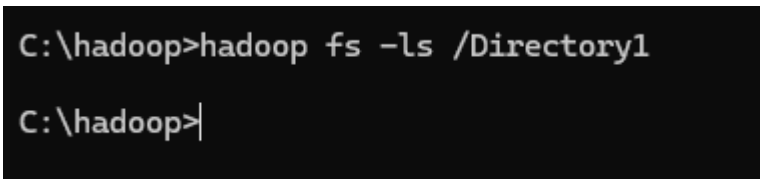
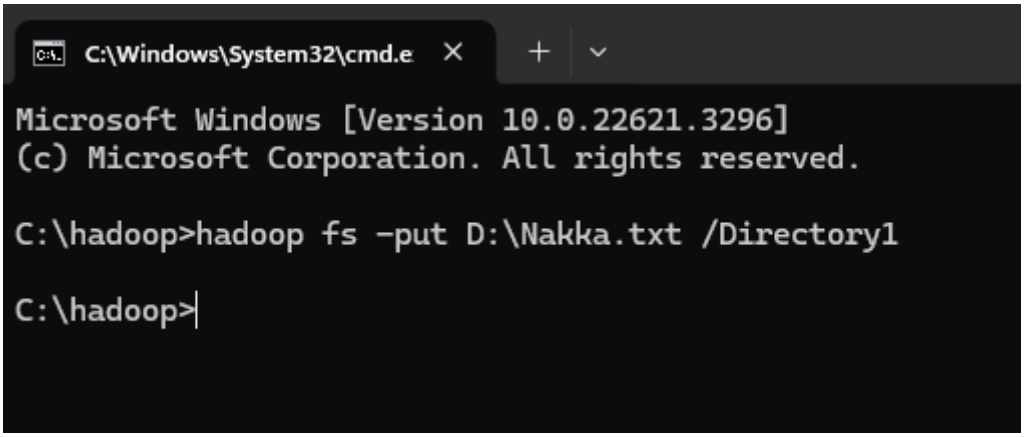


Fig. Creating Directory1

- 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 D:\Nakka.txt /Directory1



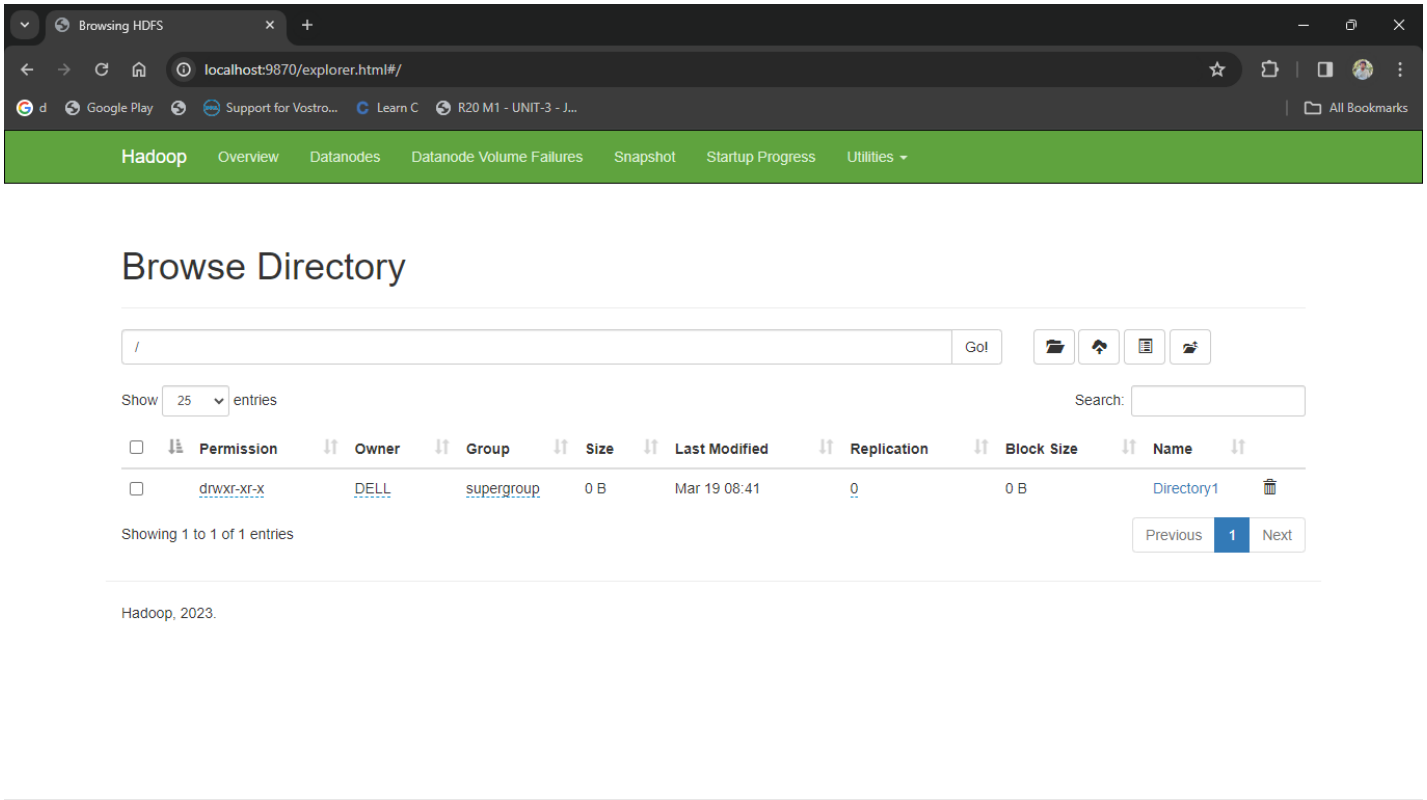


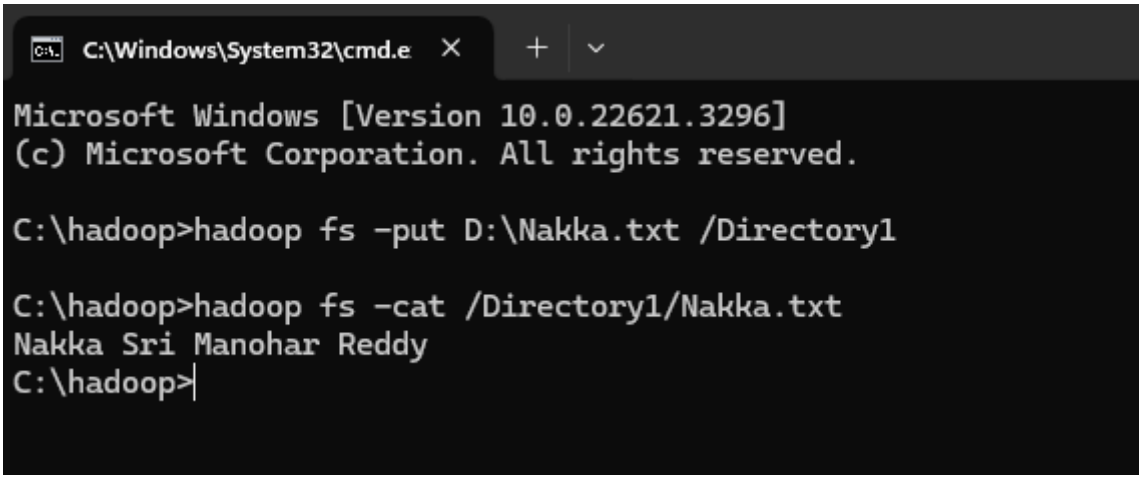
Fig. Sample.txt uploaded to HDFS

- Copies/Downloads files to the local file system

Syntax: C:\hadoop>hadoop fs -get /Directory1 D:\Nakka.txt

- See contents of a file

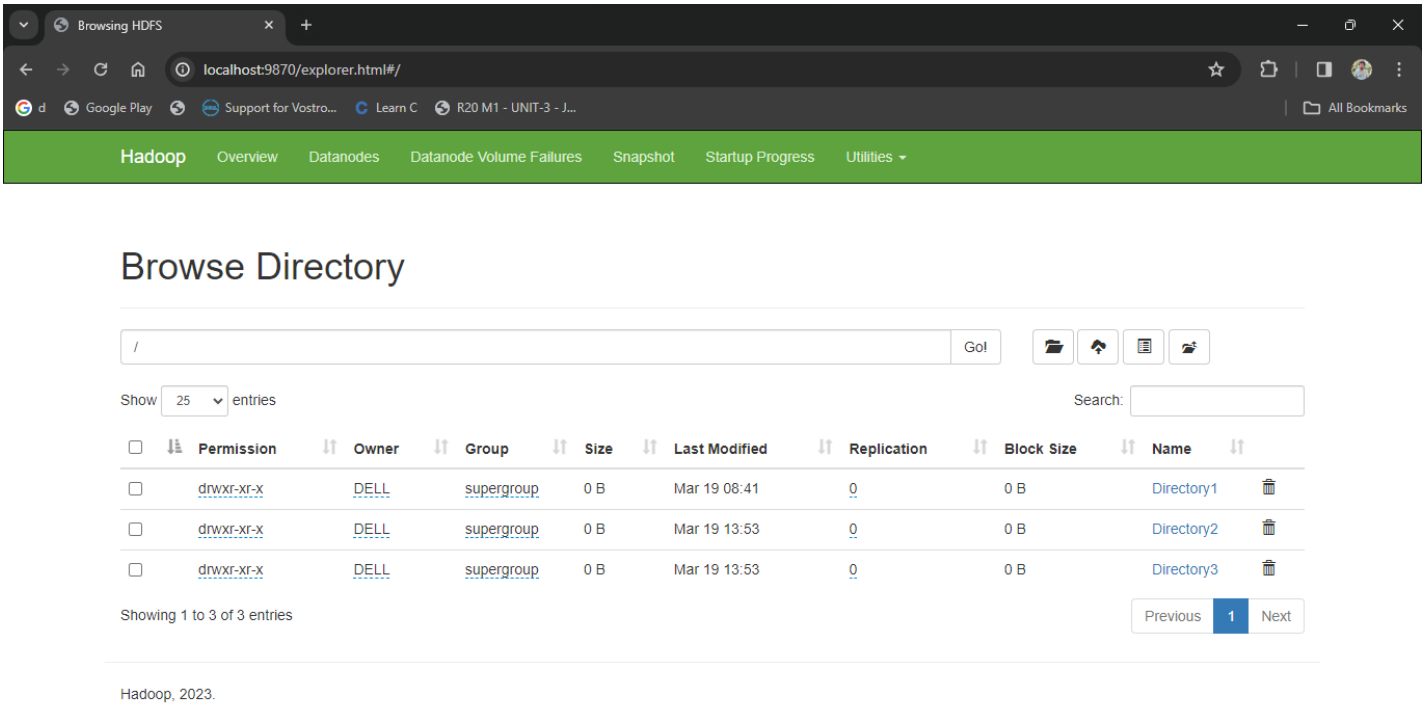
  Syntax: C:\hadoop>hadoop fs -cat /Directory1/Nakka.txt



- 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/Nakka.txt /Directory2

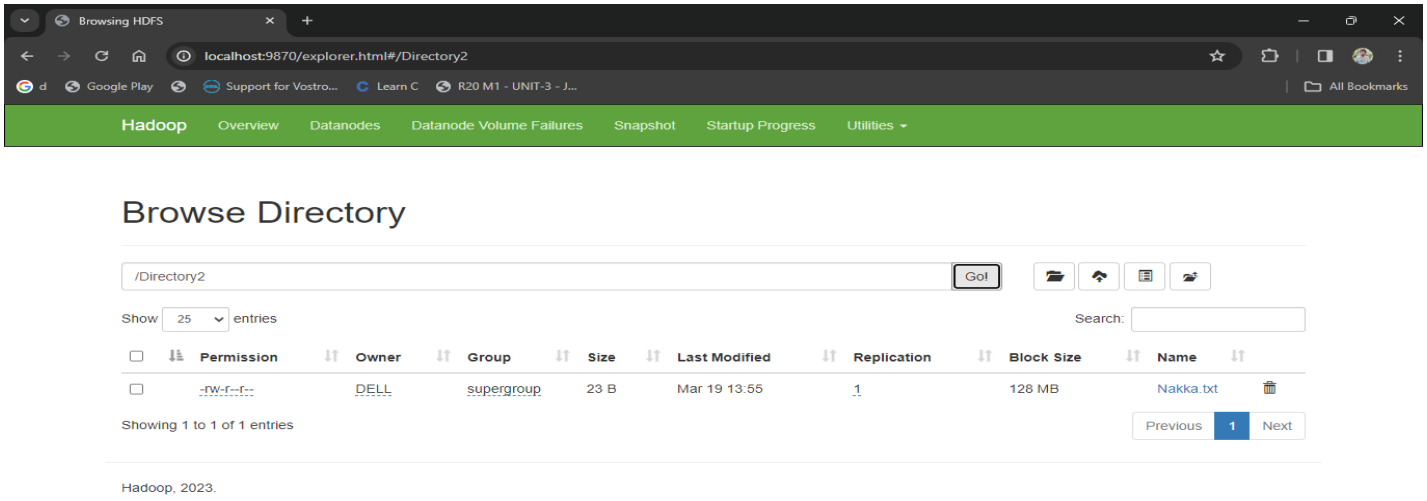


Fig. Copy to Local

- **Remove a file or directory in HDFS.**

  Remove files specified as argument. Deletes directory only when it is empty

  Syntax: C:\hadoop>hadoop fs -rm /Directory2/Nakka.txt

```
C:\hadoop>hadoop fs -rm /Directory2/Nakka.txt
Deleted /Directory2/Nakka.txt

C:\hadoop>
```

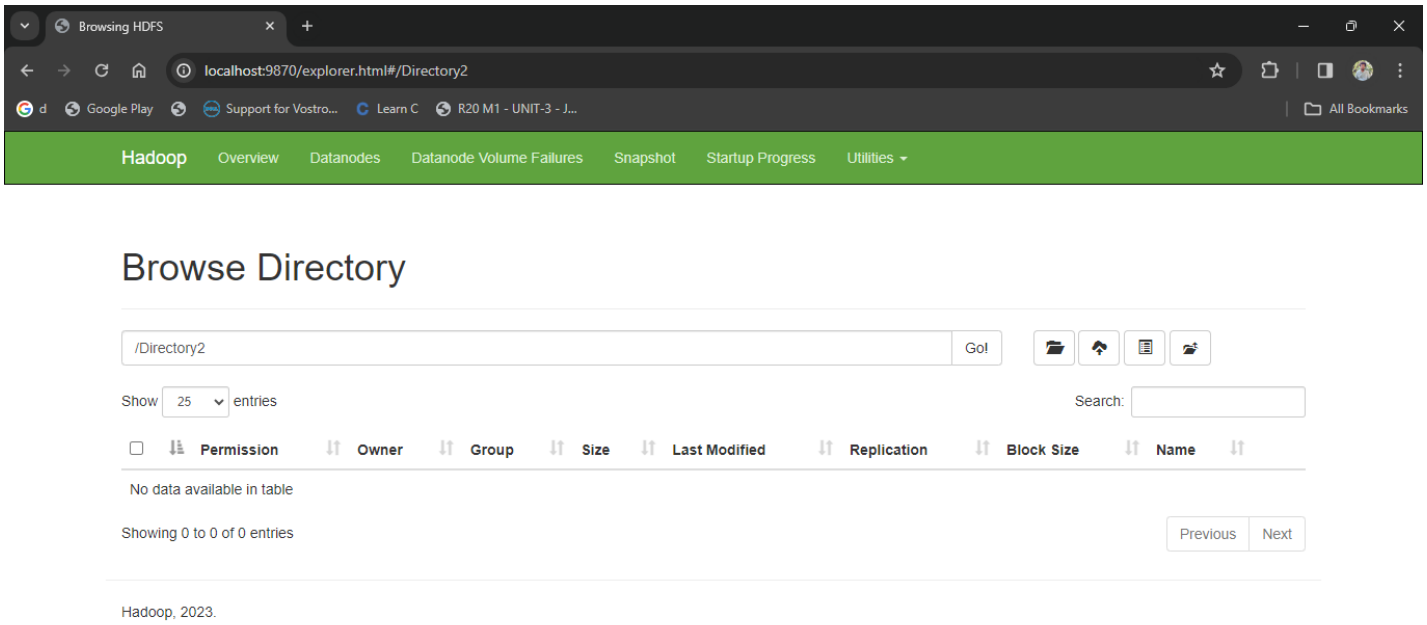


Fig. File Deleted in Directory2

- Display the aggregate length of a file.

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

```
C:\hadoop>hadoop fs -du /Directory1
23  23  /Directory1/Nakka.txt

C:\hadoop>
```

Fig. Length of a Directory1

- Removing a Directory.

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

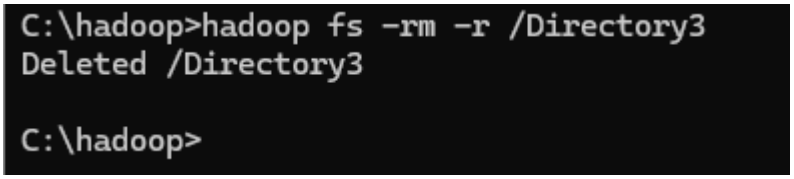


Fig. Directory3 Removed Successfully

EXPERIMENT -4

DATE:

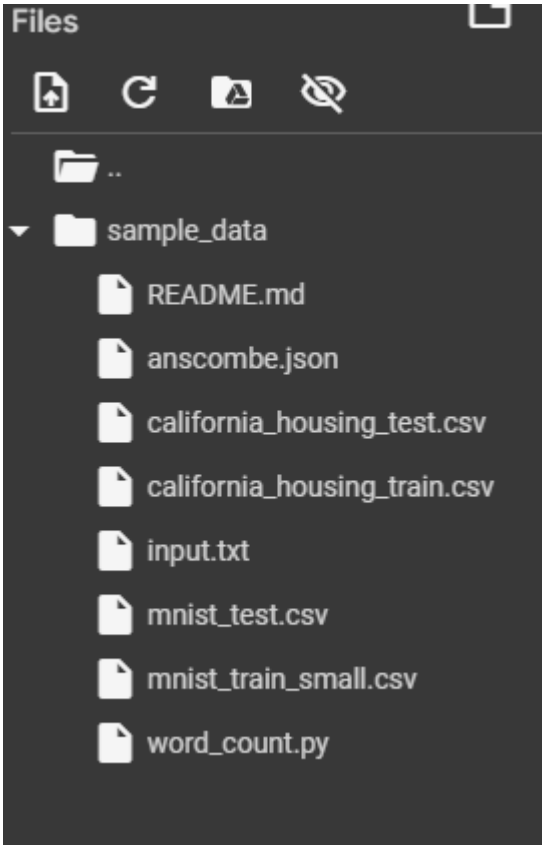
Aim: Run a basic Word Count MapReduce program to understand MapReduce Paradigm:

- Open the Google colab.
- Install the mrjob package by following command:

!pip install mrjob

```
pip install mrjob

Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.6/439.6 kB 4.4 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from mrjob) (6.0.1)
Installing collected packages: mrjob
Successfully installed mrjob-0.7.4
```

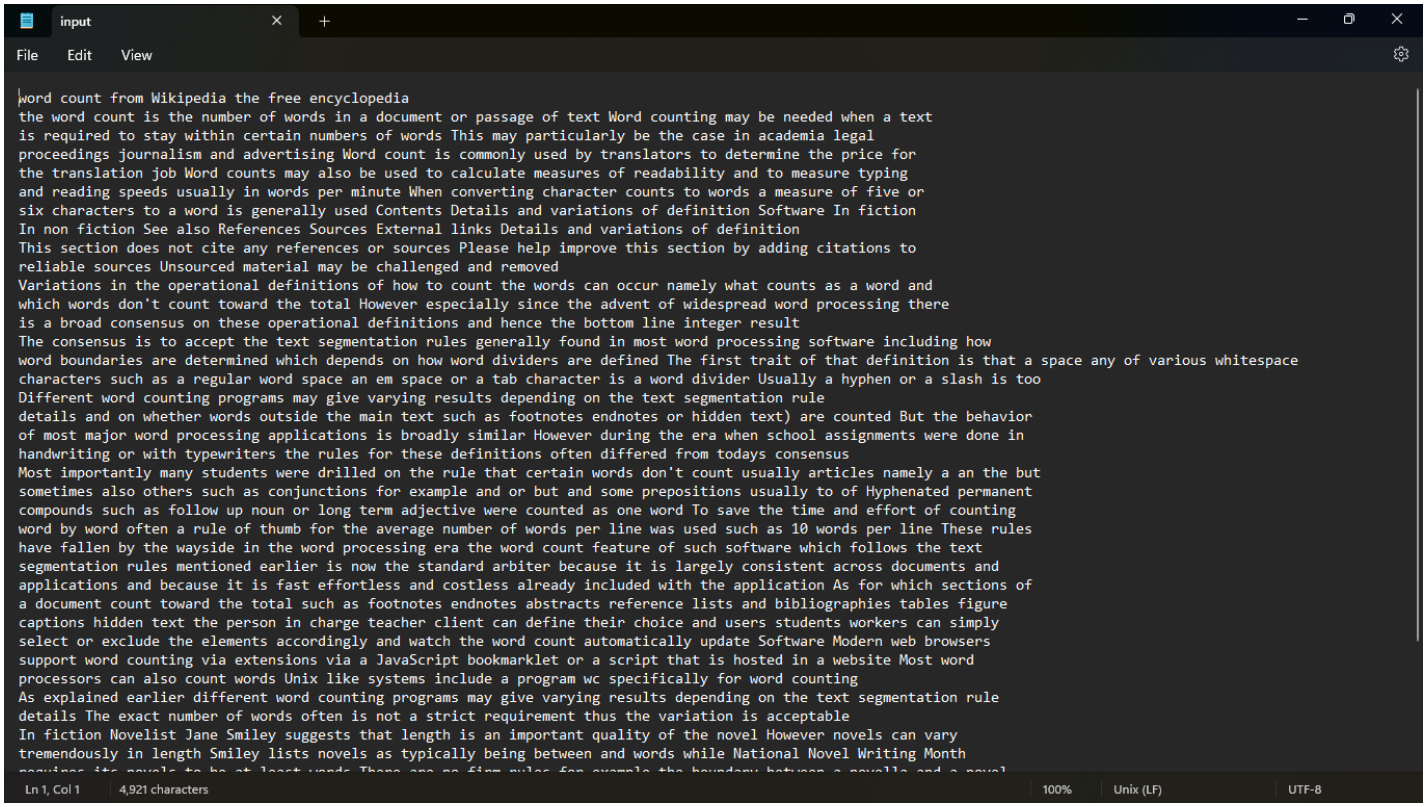
- Upload the input files to the data.



- Here is the MapReduce code for the word_count:

```
from mrjob.job import MRJob
import re
WORD_REGEXP = re.compile(r"[\w']+")
class MRWordCount(MRJob):
    def mapper(self, _, line):
        words = WORD_REGEXP.findall(line)
        for word in words:
            yield word.lower(), 1
    def reducer(self, word, counts):
        yield word, sum(counts)
if __name__ == '__main__':
    MRWordCount.run()
```

- Here is the input data:



- Here is the command to execute the program:

```
!python /content/sample_data/word_count.py /content/sample_data/input.txt
```

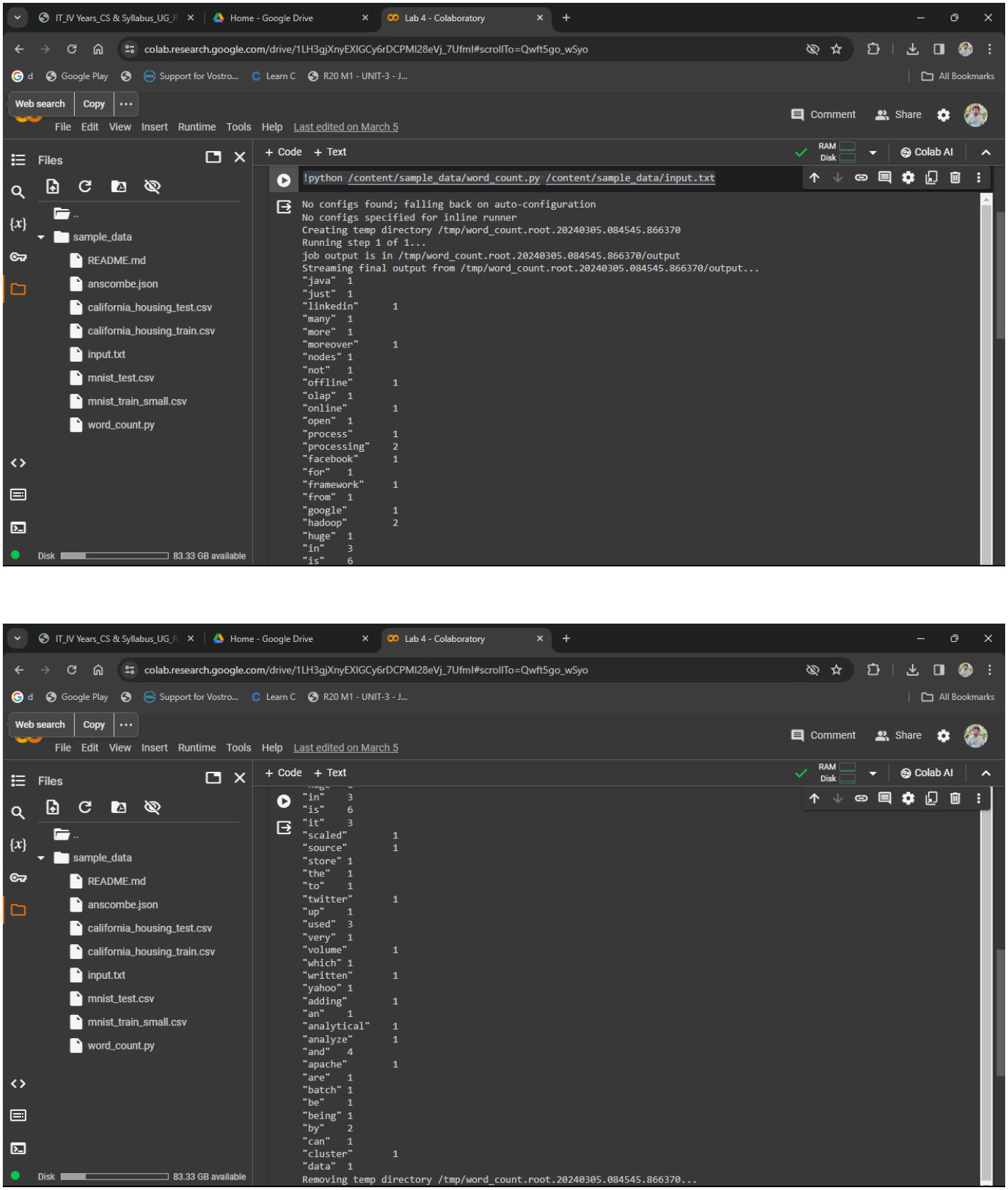


Fig: Word counts in the given data

EXPERIMENT -5

DATE:

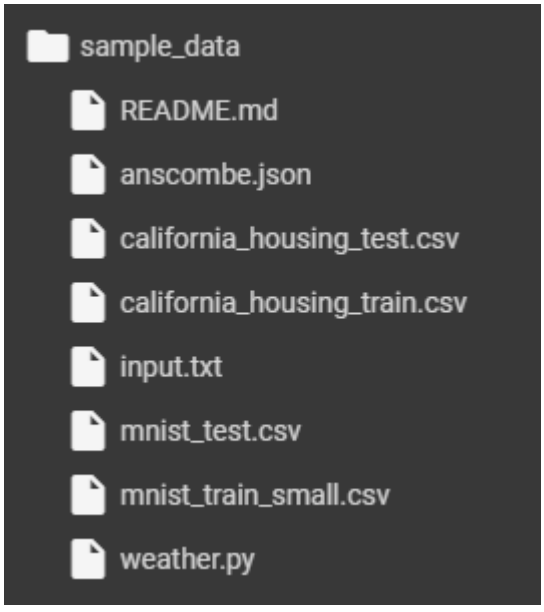
Aim: Write a map reduce program that mines weather data.

Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record-oriented.

- Open the Google Colab
- Install the mrjob package by using the following command:

```
+ Code  + Text

!pip install mrjob

Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.6/439.6 kB 3.2 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from mrjob) (6.0.1)
Installing collected packages: mrjob
Successfully installed mrjob-0.7.4
```

- Upload the required files to the data



- Here is the input data of the weather:

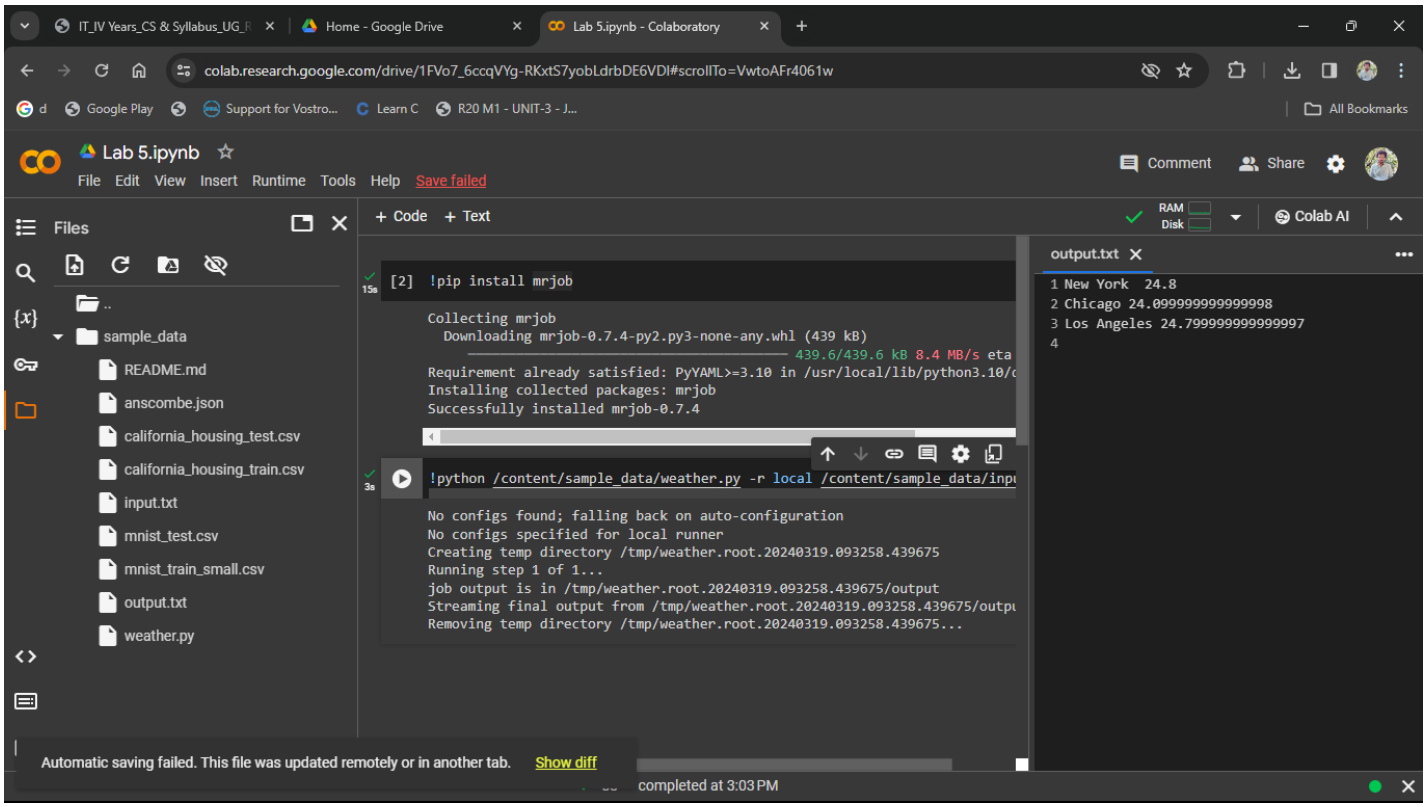
```
2024-03-01,New York,25.6
2024-03-01,Los Angeles,24.8
2024-03-02,New York,26.5
2024-03-02,Los Angeles,27.3
2024-03-02,Chicago,28.1
2024-03-03,New York,23.9
2024-03-03,Los Angeles,24.7
2024-03-03,Chicago,22.4
2024-03-04,New York,23.2
2024-03-04,Los Angeles,22.4
2024-03-04,Chicago,21.8
```

- Here is the MapReduce code for examining the weather data:

```
from mrjob.job import MRJob
class WeatherDataMining(MRJob):
    def mapper(self, _, line):
        # Split the line into fields
        fields = line.split(',')
        # Extract relevant data (e.g., location and temperature)
        location = fields[1]
        temperature = float(fields[2])
        # Emit key-value pairs with location as key and temperature as value
        yield location, temperature
    def reducer(self, key, values):
        # Calculate the average temperature for each location
        total_temperature = 0
        num_records = 0
        for temperature in values:
            total_temperature += temperature
            num_records += 1
        average_temperature = total_temperature / num_records
        # Write the output to a file
        with open('/content/sample_data/output.txt', 'a') as f:
```

```
        f.write(f'{key}\t{average_temperature}\n')

if __name__ == '__main__':
    WeatherDataMining.run()
```

- Here is the output:



Output:

New York 24.8

Chicago 24.099999999999998

Los Angeles 24.799999999999997

EXPERIMENT -6

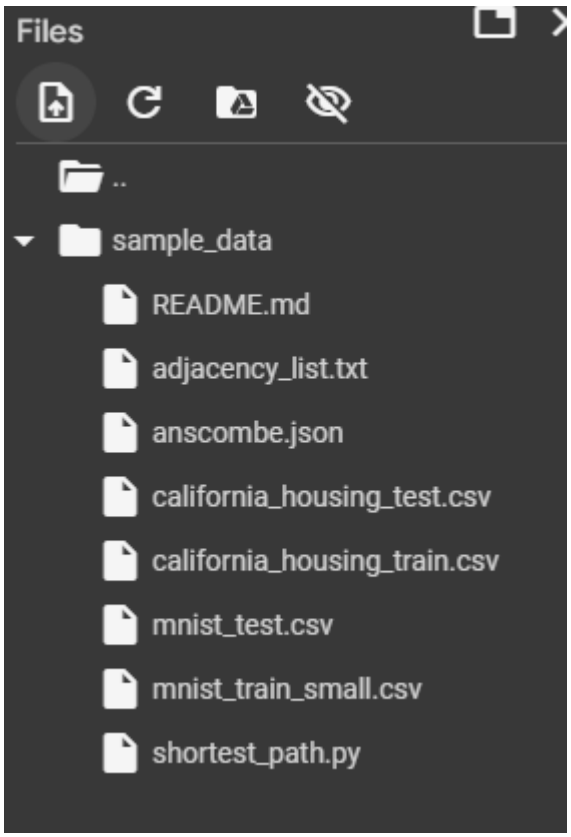
DATE:

Aim: Use MapReduce to find the shortest path between two people in a social graph

- Install the mrjob package to colab

```
[1] !pip install mrjob

Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.6/439.6 kB 3.3 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from mrjob) (6.0.1)
Installing collected packages: mrjob
Successfully installed mrjob-0.7.4
```

- Upload the required data files



- Here is the code to find shortest path

```
from collections import defaultdict
class MapReduceShortestPath:
    def __init__(self, adjacency_list):
        self.adjacency_list = adjacency_list
```

```
    self.distances = defaultdict(lambda: float('inf'))
    self.back_pointers = {}

def map_step(self, node, distance):
    for neighbor, edge_weight in self.adjacency_list[node]:
        if distance + edge_weight < self.distances[neighbor]:
            self.distances[neighbor] = distance + edge_weight
            self.back_pointers[neighbor] = node
            yield neighbor, self.distances[neighbor]

def reduce_step(self, mapped_nodes):
    updated = False
    for node, distance in mapped_nodes:
        if distance < self.distances[node]:
            self.distances[node] = distance
            updated = True
    return updated

def shortest_path(self, source, target):
    self.distances[source] = 0
    iteration = 0

    while True:
        iteration += 1
        print(f"Iteration {iteration}")

        # Map Step
        mapped_nodes = []
        for node in self.adjacency_list:
            mapped_nodes.extend(self.map_step(node, self.distances[node]))
```

```
            # Reduce Step
            updated = self.reduce_step(mapped_nodes)
            if not updated or self.distances[target] == float('inf'):
                break

        path = []
        node = target
        while node is not None:
            path.append(node)
            node = self.back_pointers.get(node)
        return path[::-1]

def read_adjacency_list(file_path):
    adjacency_list = defaultdict(list)
    with open(file_path, 'r') as file:
        for line in file:
            node1, node2, weight = line.split()
            adjacency_list[node1].append((node2, int(weight)))
            adjacency_list[node2].append((node1, int(weight)))  # Assuming undirected graph
    return adjacency_list

# Example usage:
if __name__ == "__main__":
    file_path = "/content/sample_data/adjacency_list.txt"  # Replace with the path to your adjacency list text file
    adjacency_list = read_adjacency_list(file_path)

    map_reduce_sp = MapReduceShortestPath(adjacency_list)
    shortest_path = map_reduce_sp.shortest_path('A', 'F')
    print("Shortest path:", shortest_path)
```

- Here is the input data to the path

A B 5

A C 3

B D 7

C D 2

D E 4

E F 6

- Here is the output for the data

```
!python /content/sample_data/shortest_path.py /content/sample_data/adjacency_list.txt.txt

Iteration 1
Shortest path: ['A', 'C', 'D', 'E', 'F']
```

EXPERIMENT -7

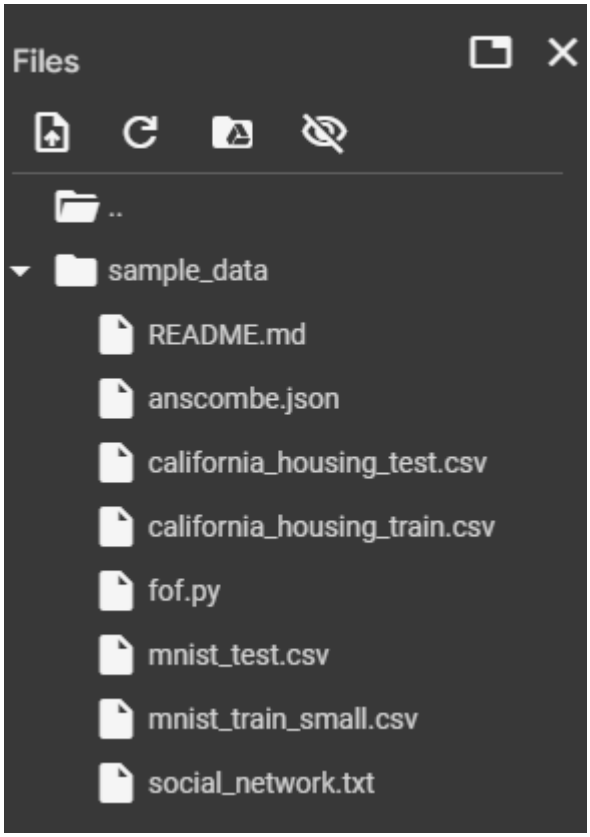
DATE:

Aim: Implement Friends-of-friends algorithm in MapReduce.

- Install the mrjob package

```
[1] !pip install mrjob

Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.6/439.6 kB 7.0 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from mrjob) (6.0.1)
Installing collected packages: mrjob
Successfully installed mrjob-0.7.4
```

- Upload the data files to local



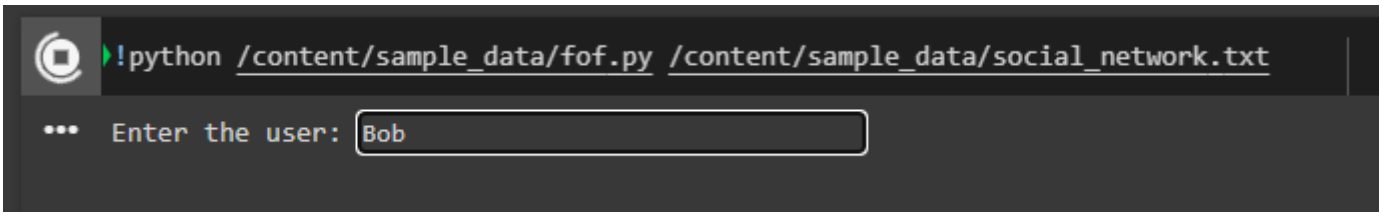
- Here is the python code for finding friends of friends in MapReduce

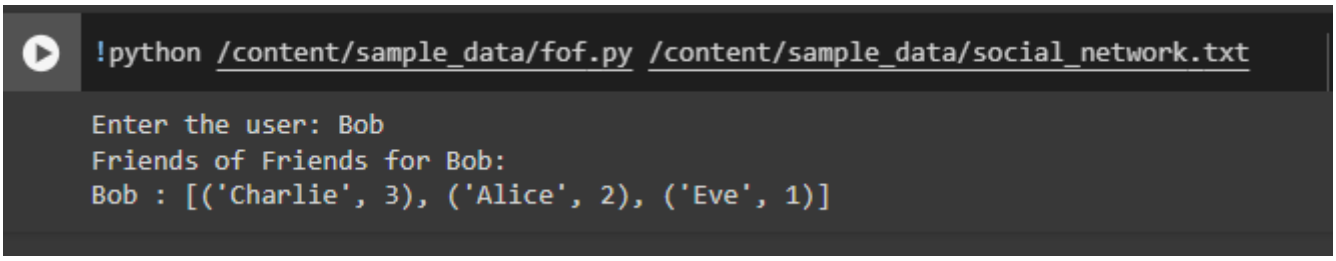
```
from collections import defaultdict
from operator import itemgetter


class FoFMapReduce:
    def __init__(self):
```

```
        self.common_friends = defaultdict(list)

    def map_common_friends(self, user, friends):
        for friend in friends:
            self.common_friends[friend].append(user)

    def reduce_common_friends(self):
        for friend, users in self.common_friends.items():
            users.sort()
            for i in range(len(users)):
                for j in range(i+1, len(users)):
                    yield (users[i], users[j]), friend

    def map_sort_common_friends(self, pair, friend):
        yield pair[0], (friend, 1)
        yield pair[1], (friend, 1)

    def reduce_sort_common_friends(self, user, common_friends):
        common_friends_count = defaultdict(int)
        for friend, count in common_friends:
            common_friends_count[friend] += count
        sorted_friends = sorted(common_friends_count.items(), key=itemgetter(1), reverse=True)
        yield user, sorted_friends

def read_social_network(file_path):
    social_network = defaultdict(list)
    with open(file_path, 'r') as file:
        for line in file:
            user, *friends = line.split()
            social_network[user] = friends
```

```
    return social_network

def get_input():
    user = input("Enter the user: ")
    return user

# Read social network graph from the text file
file_path = "/content/sample_data/social_network.txt"  # Replace with the path to your social network text file
social_network = read_social_network(file_path)

# Get user input
user = get_input()

# First MapReduce job: Calculate common friends
mr1 = FoFMapReduce()
for u, friends in social_network.items():
    mr1.map_common_friends(u, friends)
common_friends_pairs = mr1.reduce_common_friends()

# Second MapReduce job: Sort common friends
mr2 = FoFMapReduce()
for pair, friend in common_friends_pairs:
    for mapped_user, mapped_friend in mr2.map_sort_common_friends(pair, friend):
        mr2.common_friends[mapped_user].append(mapped_friend)
sorted_common_friends = mr2.reduce_sort_common_friends(user, mr2.common_friends[user])

# Print Friends-of-friends for the specified user
print("Friends of Friends for", user + ":")
for friend, count in sorted_common_friends:
    print(friend, ":", count)
```

- Here is the output





EXPERIMENT -8

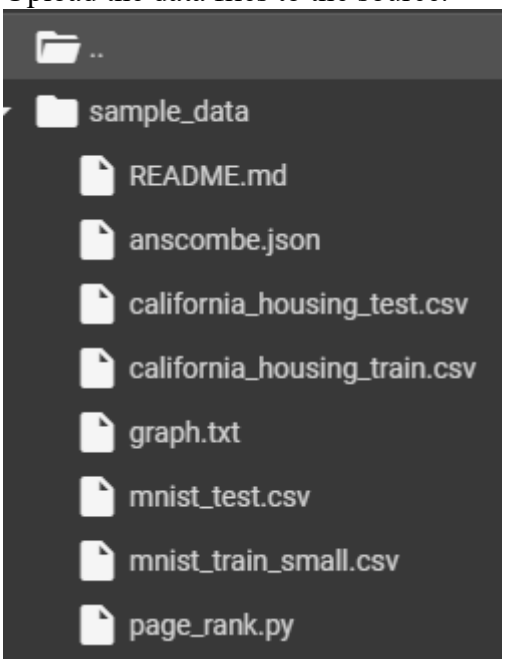
DATE:

Aim: Implement an iterative PageRank graph algorithm in MapReduce.

- Install the mrjob package

```
!pip install mrjob

Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.6/439.6 kB 5.1 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from mrjob) (6.0.1)
Installing collected packages: mrjob
Successfully installed mrjob-0.7.4
```

- Upload the data files to the source.



- Here is the code for Pagerank graph.

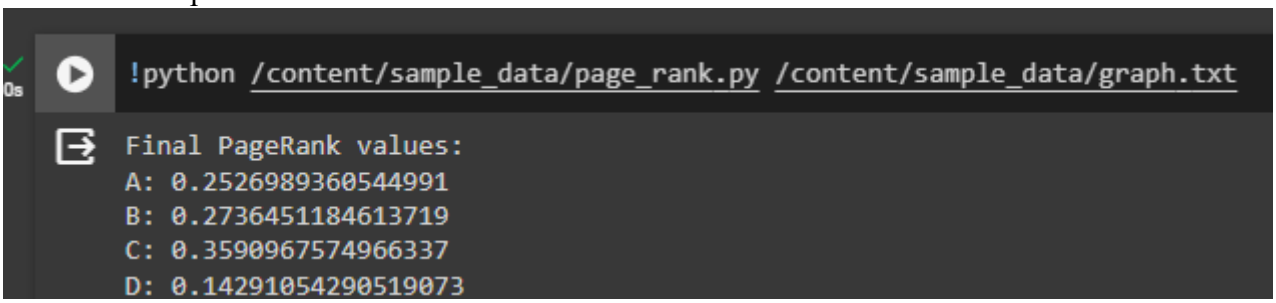
```
import numpy as np


class PageRankMapReduce:
    def __init__(self, graph, damping_factor=0.85, convergence_threshold=0.0001, max_iterations=100):
        self.graph = graph
        self.damping_factor = damping_factor
        self.convergence_threshold = convergence_threshold
```

```
        self.max_iterations = max_iterations
        self.num_nodes = len(graph)
        self.page_ranks = {node: 1.0 / self.num_nodes for node in graph}

    def map_step(self):
      for node, neighbors in self.graph.items():
        partial_page_rank = self.page_ranks[node] * self.damping_factor / len(neighbors)
        for neighbor in neighbors:
            yield neighbor, partial_page_rank



    def reduce_step(self, mapped_data):
      new_page_ranks = {}
      for node, partial_page_rank in mapped_data:
        if node not in new_page_ranks:
            new_page_ranks[node] = 0.0
        new_page_ranks[node] += partial_page_rank
        for node in new_page_ranks:
          new_page_ranks[node] = self.damping_factor * new_page_ranks[node] + (1 - self.damping_factor) /
self.num_nodes


      return new_page_ranks


    def run(self):
        iteration = 0
        while iteration < self.max_iterations:
            # Map Step
            mapped_data = list(self.map_step())


            # Reduce Step
            new_page_ranks = self.reduce_step(mapped_data)
```

```
            # Check convergence
            if self.check_convergence(new_page_ranks):
                break

            # Update PageRanks for next iteration
            self.page_ranks = new_page_ranks
            iteration += 1

    def check_convergence(self, new_page_ranks):
        convergence_values = [abs(new_page_ranks[node] - self.page_ranks[node]) for node in self.graph]
        max_convergence = max(convergence_values)
        return max_convergence < self.convergence_threshold

def read_graph(file_path):
    graph = {}
    with open(file_path, 'r') as file:
        for line in file:
            nodes = line.strip().split()
            graph[nodes[0]] = nodes[1:]
    return graph

# Example usage:
if __name__ == "__main__":
    # Read graph from the text file
    file_path = "/content/sample_data/graph.txt"  # Replace with the path to your graph text file
    graph = read_graph(file_path)

    # Initialize PageRank MapReduce object
    page_rank_mr = PageRankMapReduce(graph)
```

```
# Run the iterative PageRank algorithm
page_rank_mr.run()

# Print the final PageRank values
print("Final PageRank values:")
for node, page_rank in sorted(page_rank_mr.page_ranks.items()):
    print(f"{node}: {page_rank}")
```

- Here is the output



EXPERIMENT -9

DATE:

Aim: Perform an efficient semi-join in MapReduce.

- Install the mrjob package.

```
[1] !pip install mrjob

Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.6/439.6 kB 4.5 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from mrjob) (6.0.1)
Installing collected packages: mrjob
Successfully installed mrjob-0.7.4
```

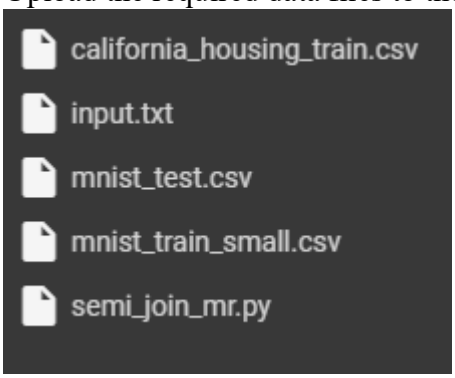
- Install the mmh3 package.

```
[3] !pip install mmh3

Collecting mmh3
  Downloading mmh3-4.1.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.6/67.6 kB 2.0 MB/s eta 0:00:00
Installing collected packages: mmh3
Successfully installed mmh3-4.1.0
```

- Install the bitarray package.

```
[5] !pip install bitarray

Collecting bitarray
  Downloading bitarray-2.9.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (288 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 288.3/288.3 kB 5.5 MB/s eta 0:00:00
Installing collected packages: bitarray
Successfully installed bitarray-2.9.2
```

- Upload the required data files to the notebook.



- Here is the code for semi-join in MapReduce.

```
from mrjob.job import MRJob
import mmh3
from bitarray import bitarray

# Constants
BLOOM_FILTER_SIZE = 1000000
NUM_HASH_FUNCTIONS = 3

class SemiJoinMR(MRJob):

    def mapper_init(self):
        # Load Bloom filter from Distributed Cache
        self.bloom_filter = bitarray(BLOOM_FILTER_SIZE)
        self.bloom_filter.setall(0)
        with open('/content/sample_data/bloom_filter.txt', 'r') as f:
            for line in f:
                position = int(line.strip())
                self.bloom_filter[position] = 1

    def mapper(self, _, line):
        # Perform semi-join using Bloom filter
        record = line.strip()
        if self.is_in_bloom_filter(record):
            yield record, 'A'

    def reducer(self, key, values):
        # Emit records that passed Bloom filter check
        if 'A' in values:
```

```
            yield key, None

    def is_in_bloom_filter(self, record):
        # Perform membership query against Bloom filter
        for i in range(NUM_HASH_FUNCTIONS):
            hash_val = mmh3.hash(record, i) % BLOOM_FILTER_SIZE
            if not self.bloom_filter[hash_val]:
                return False
        return True

if __name__ == '__main__':
    SemiJoinMR.run()
```

- Here is the command for output

```
!python /content/sample_data/semi_join_mr.py /content/sample_data/input.txt >output.txt

No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/semi_join_mr.root.20240321.140154.434153
Running step 1 of 1...
```

- Here is the data in bloom file.

bloom_filter.txt

```
1 BDA
2 MSD
3 ATCD
```

- Here is the output file

output.txt

```
1 BDA
2
```

EXPERIMENT -10

DATE:

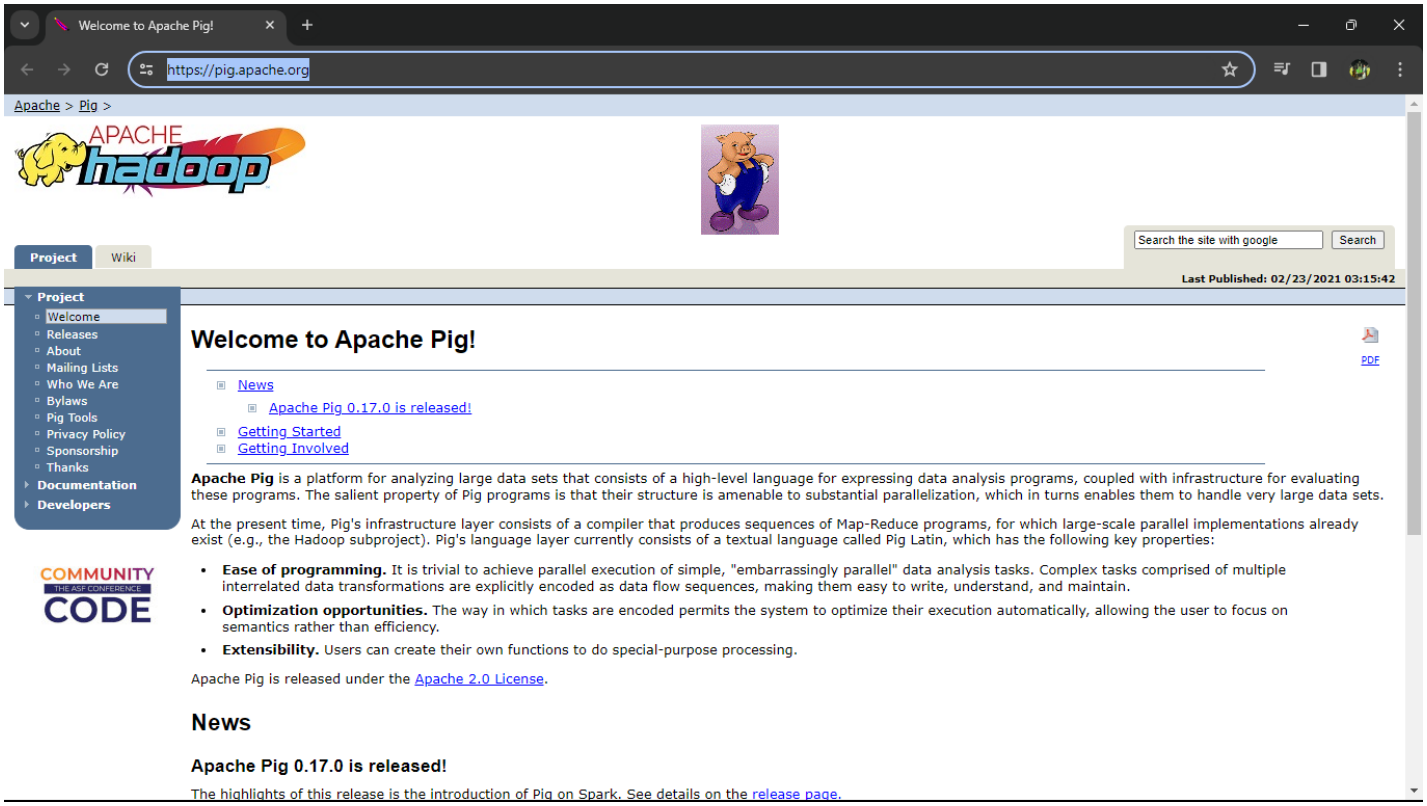
Aim: Install and Run Pig then write Pig Latin scripts to sort, group, join, project, and filter your data.

Download Apache Pig

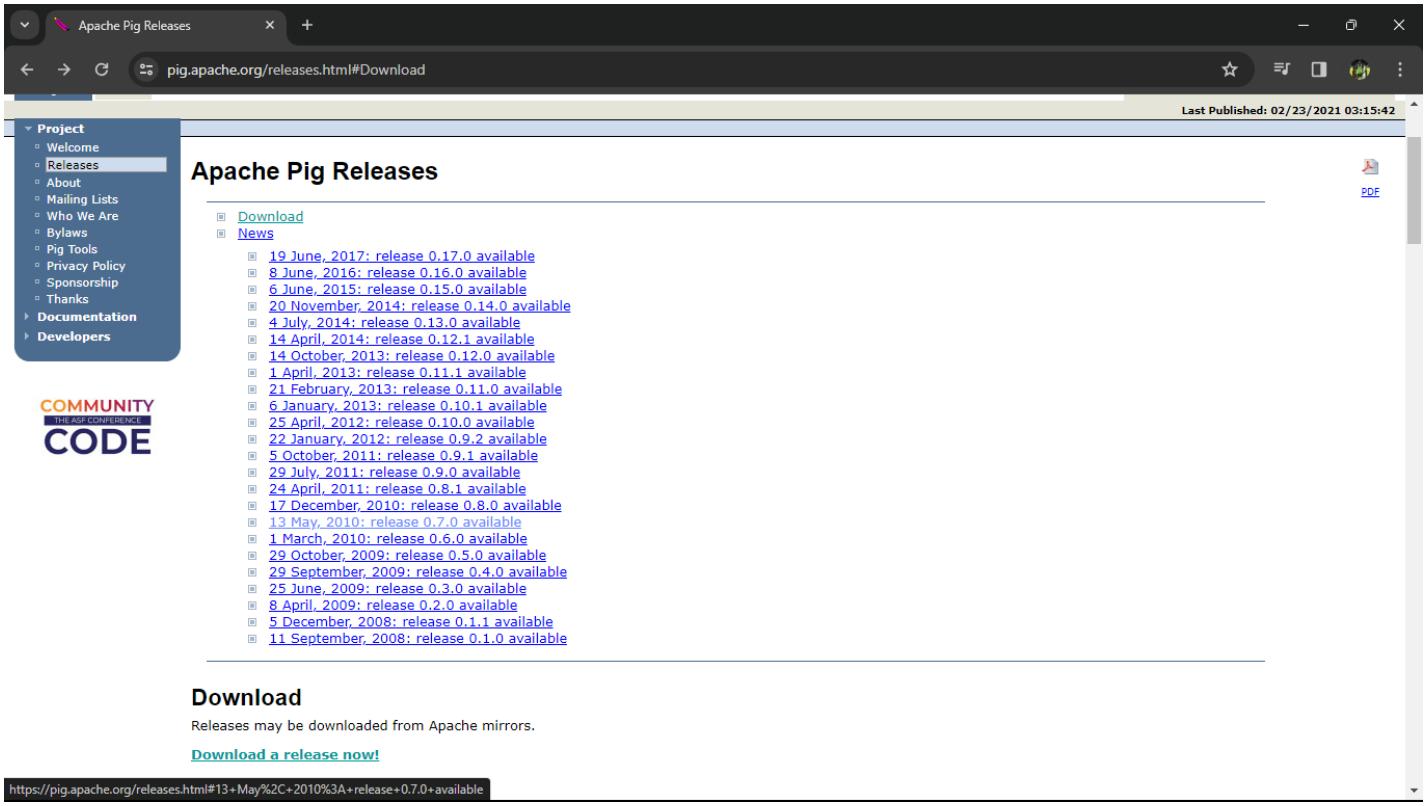
First of all, download the latest version of Apache Pig from the following website –

https://pig.apache.org/

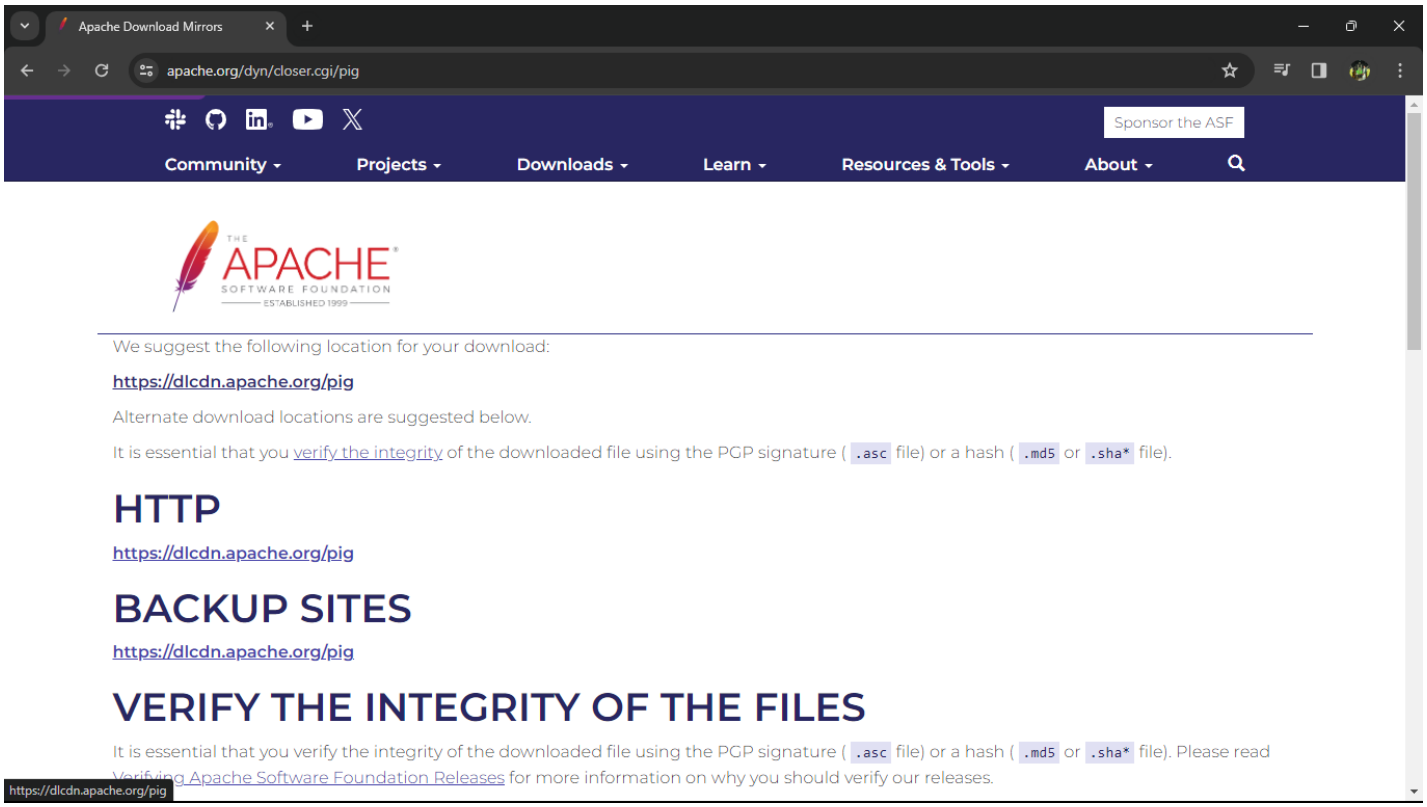
- Open the homepage of Apache Pig website. Under the section News, click on the link release page as shown in the following snapshot.



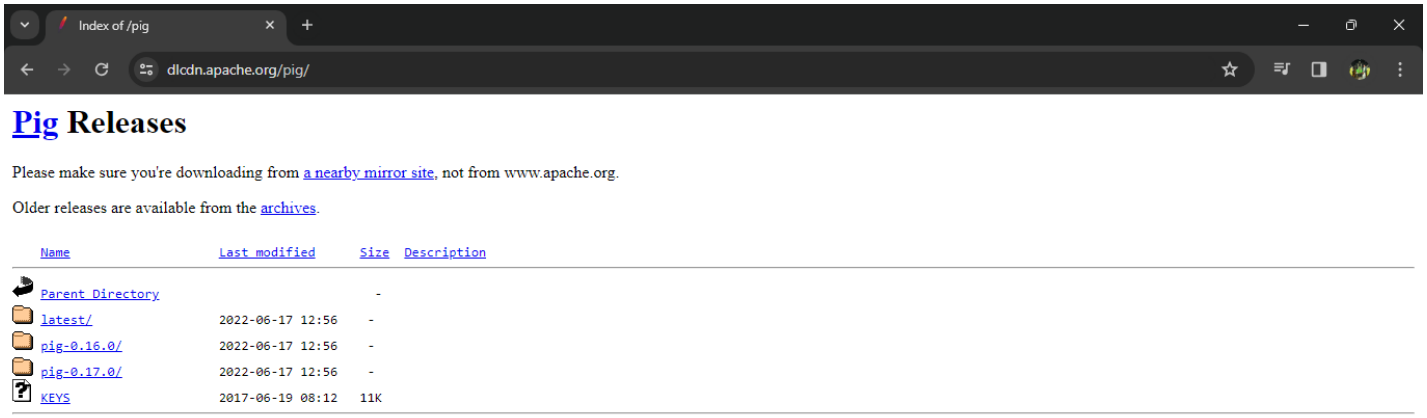
- On clicking the specified link, you will be redirected to the Apache Pig Releases page. On this page, under the Download section, you will have two links, namely, Pig 0.8 and later and Pig 0.7 and before. Click on the link Pig 0.8 and later, then you will be redirected to thepage having a set of mirrors.



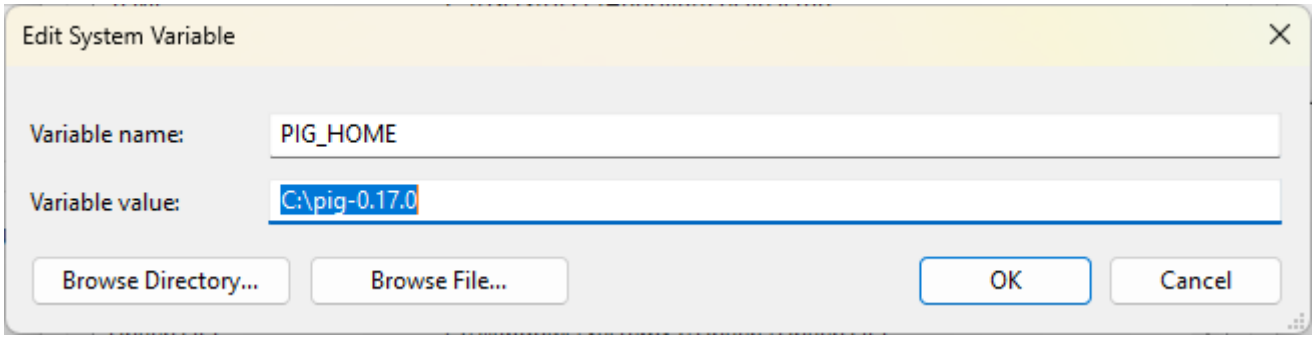
- Choose and click any one of these mirrors as shown below.



- These mirrors will take you to the Pig Releases page. This page contains various versions of Apache Pig. Click the latest version among them.



- Download the tar files of the source and binary files of Apache Pig 0.17, pig0.17.0-src.tar.gz and pig-0.17.0.tar.gz.
- Set the path in environment variables.



- **Grouping:**

  group_data = GROUP student_details by age;

  Output:

  (21,{(4,Preethi,Agarwal,21,9848022330,Pune),(1,Rajiv,Reddy,21,9848022337,Hyderabad)}),

  (22,{(3,Rajesh,Khanna,22,9848022339,Delhi),(2,siddarth,Battacharya,22,9848022338,Kolkata)}),

  (23,{(6,Archana,Mishra,23,9848022335,Chennai),(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)}),

  (24,{(8,Bharathi,Nambiayar,24,9848022333,Chennai),(7,Komal,Nayak,24,9848022334,trivendram)})

- **Sorting:**

  order_by_data = ORDER student_details BY age DESC;

  Output:

  (8,Bharathi,Nambiayar,24,9848022333,Chennai)

  (7,Komal,Nayak,24,9848022334,trivendram)

  (6,Archana,Mishra,23,9848022335,Chennai)

  (5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)

  (3,Rajesh,Khanna,22,9848022339,Delhi)

  (2,siddarth,Battacharya,22,9848022338,Kolkata)

  (4,Preethi,Agarwal,21,9848022330,Pune)

  (1,Rajiv,Reddy,21,9848022337,Hyderabad)

- **Joining:**

  customers3 = JOIN customers1 BY id, customers2 BY id;

  Output:

  (1,Ramesh,32,Ahmedabad,2000,1,Ramesh,32,Ahmedabad,2000)
  (2,Khilan,25,Delhi,1500,2,Khilan,25,Delhi,1500)
  (3,kaushik,23,Kota,2000,3,kaushik,23,Kota,2000)
  (4,Chaitali,25,Mumbai,6500,4,Chaitali,25,Mumbai,6500)
  (5,Hardik,27,Bhopal,8500,5,Hardik,27,Bhopal,8500)
  (6,Komal,22,MP,4500,6,Komal,22,MP,4500)
  (7,Muffy,24,Indore,10000,7,Muffy,24,Indore,10000)

- **Filtering:**

  filter_data = FILTER student_details BY city == 'Chennai';

  Output:

  (6,Archana,Mishra,23,9848022335,Chennai)
  (8,Bharathi,Nambiayar,24,9848022333,Chennai)

EXPERIMENT -11

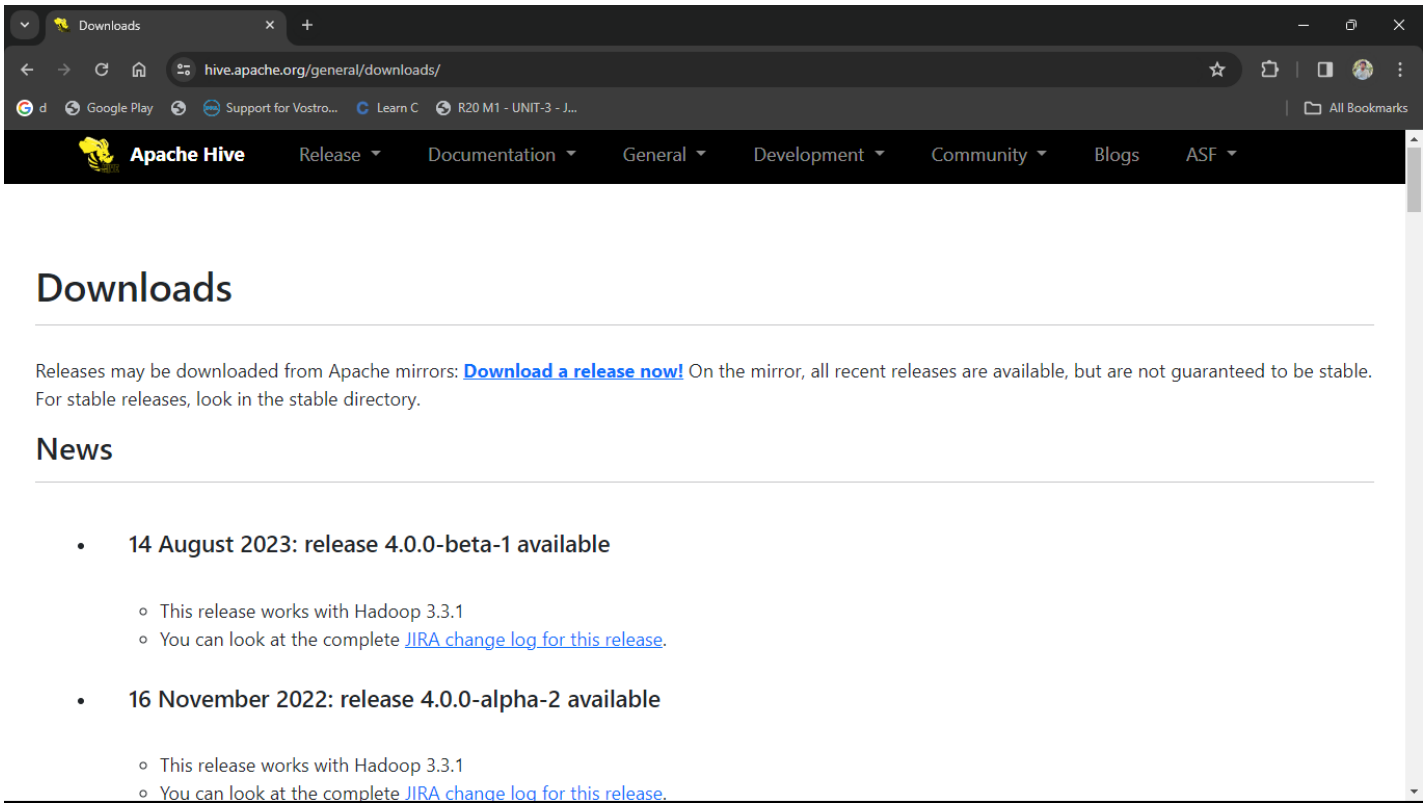
DATE:

Aim: Install and Run Hive then use Hive to create, alter, and drop databases, tables, views, functions, and indexes.

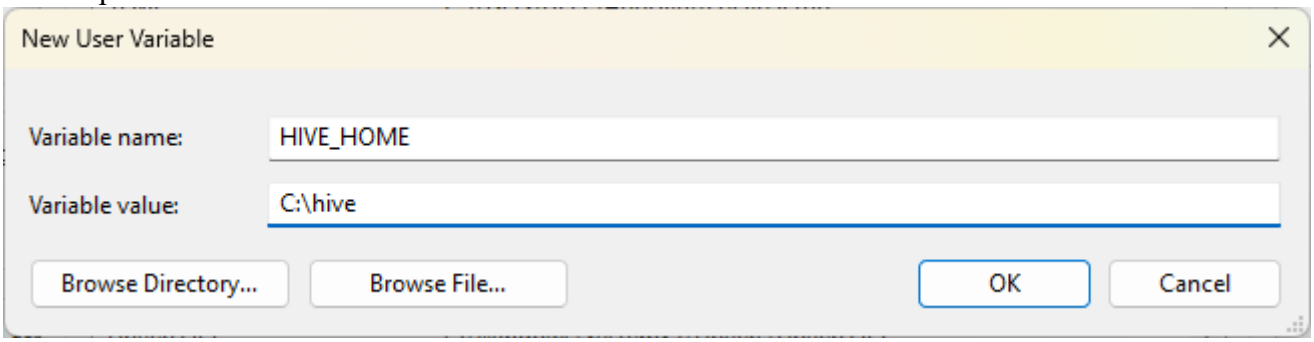
Download Apache Hive

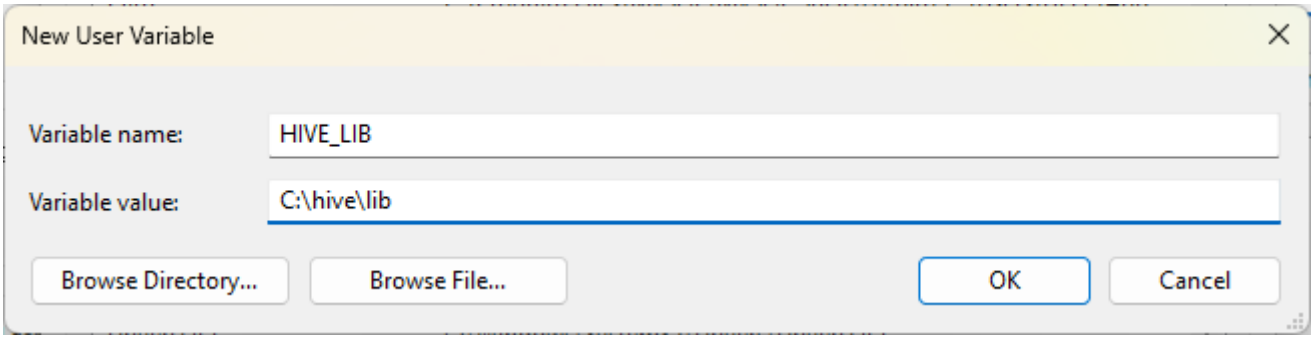
First of all, download the latest version of Apache Hive from the following website –

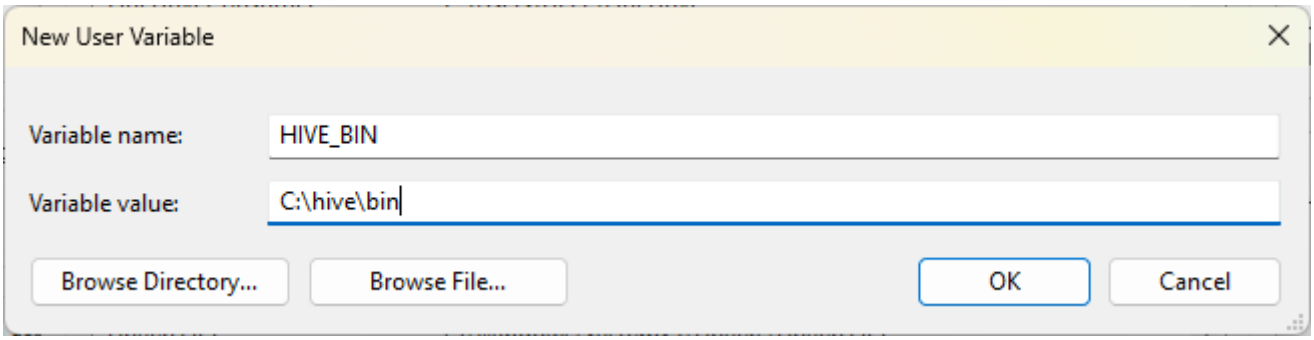
https://hive.apache.org/general/downloads/



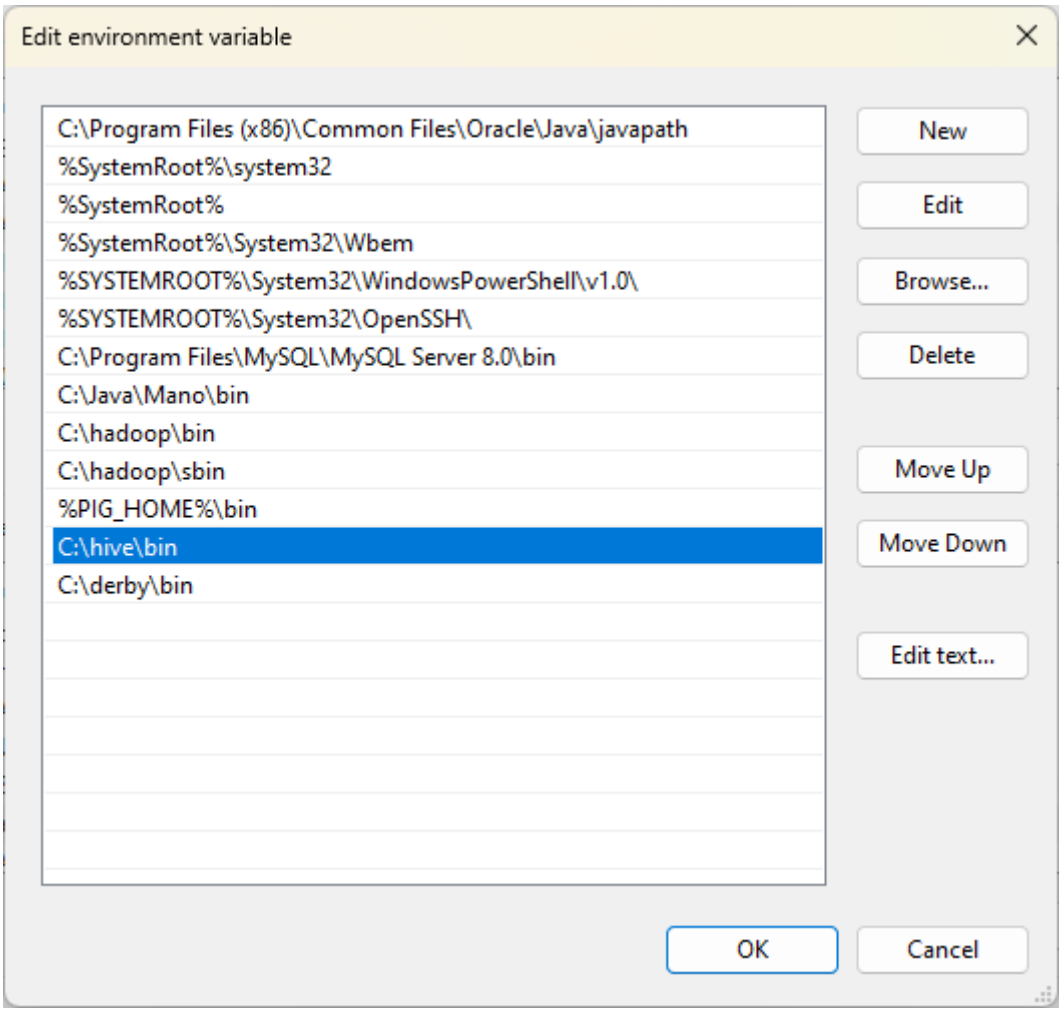
- Download the latest version
- Set the path in environmental variables.



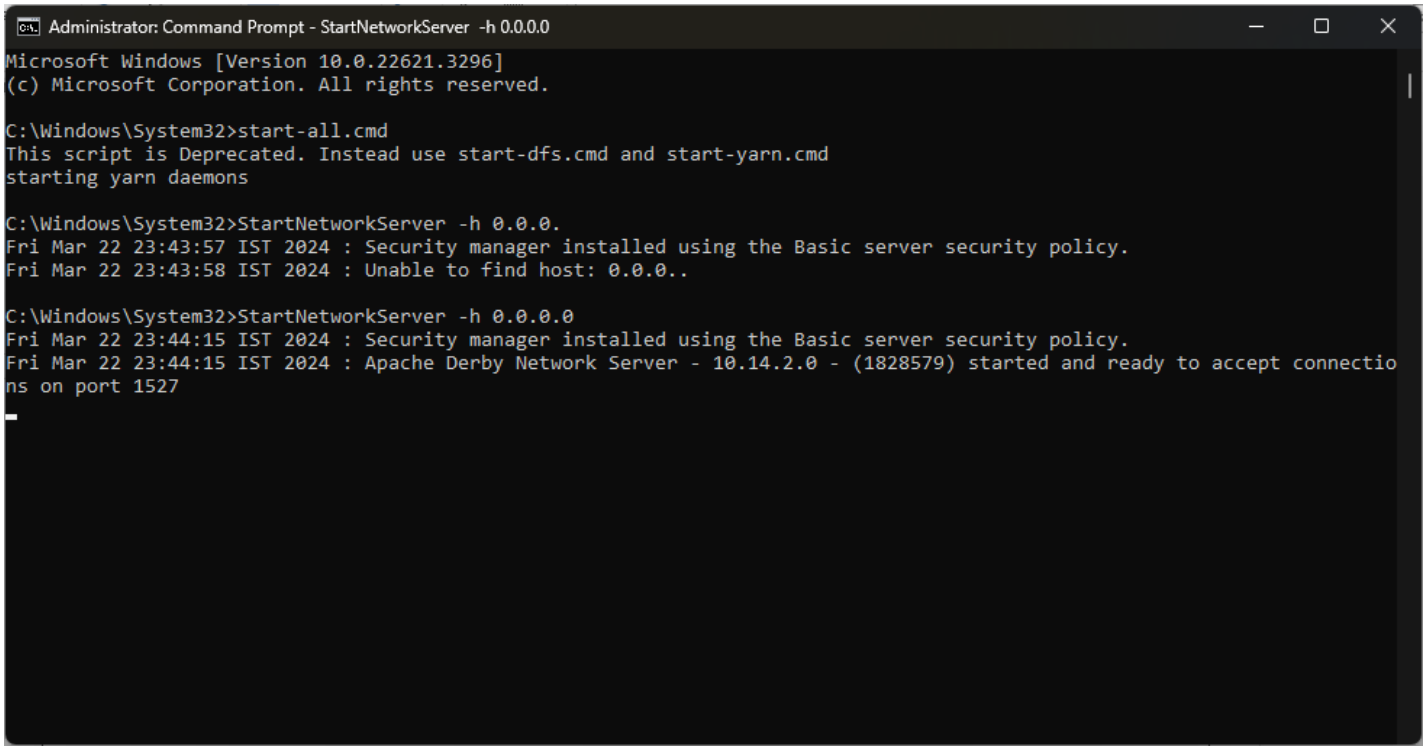




- Set the bin folder of hive to environmental variables.



- Server has been started with "StartNetworkServer –h 0.0.0.0".



- Creating a database:

  CREATE DATABASE [IF NOT EXISTS] userdb;

  Output:

  Database userdb created successfully.

- Creating a Table:

  CREATE TABLE IF NOT EXISTS employee (eid int, name String,

  salary String, destination String)

  COMMENT 'Employee details'

  ROW FORMAT DELIMITED

  FIELDS TERMINATED BY '\t'

  LINES TERMINATED BY '\n'

  STORED AS TEXTFILE;

  Output:

  Table employee created.

- Altering a Table:

  ALTER TABLE employee RENAME TO emp;

  Output:

  Table renamed successfully.

- Dropping a Table:

  DROP TABLE IF EXISTS employee;

  Output:

  Drop table successful.

- Creating a View:

  CREATE VIEW emp_30000 AS SELECT * FROM employee WHERE salary>30000;

- Dropping a View:

  DROP VIEW emp_30000;

- Creating an Index:

  CREATE INDEX inedx_salary ON TABLE employee(salary) AS
  'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler';

- Dropping an Index:

  DROP INDEX index_salary ON employee;

- Creating a Function:

  CREATE FUNCTION my_function AS ‘com.example.MyFunctionClass’ USING JAR ‘path/to/my/jar/file’;

- Dropping a Function:

  DROP FUNCTION IF EXISTS my_function;

CONTENT BEYOND THE SYLLABUS

EXPERIMENT -1

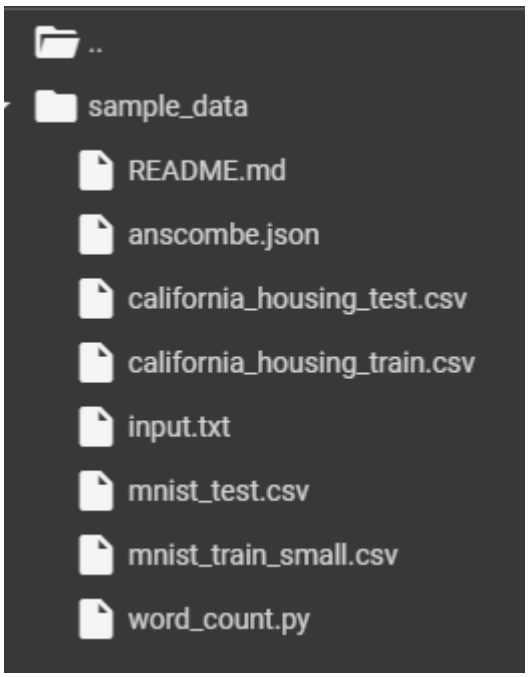
DATE:

Aim: Implement word count using combiner function in mapreduce

- Install the mrjob package.

```
pip install mrjob

Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.6/439.6 kB 4.4 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from mrjob) (6.0.1)
Installing collected packages: mrjob
Successfully installed mrjob-0.7.4
```

- Upload the required data files to the local.



- Here is the code for word count using combiner function in mapreduce.

```
from mrjob.job import MRJob
import re

WORD_REGEXP = re.compile(r"[\w']+")
```

```
class MRWordCountWithCombiner(MRJob):

    def mapper(self, _, line):
        words = WORD_REGEXP.findall(line)
        word_counts = {}
        for word in words:
            word = word.lower()
            if word in word_counts:
                word_counts[word] += 1
            else:
                word_counts[word] = 1

        for word, count in word_counts.items():
            yield word, count

    def combiner(self, word, counts):
        yield word, sum(counts)

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    MRWordCountWithCombiner.run()
```

- Here is the output:

```
!python /content/sample_data/word_count.py /content/sample_data/input.txt

No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/word_count.root.20240322.185329.473603
Running step 1 of 1...
job output is in /tmp/word_count.root.20240322.185329.473603/output
Streaming final output from /tmp/word_count.root.20240322.185329.473603/output
"10"    1
"a"     28
"about" 1
"abstracts"     1
"academia"      1
"academic"      1
"accept"        1
"acceptable"    2
"accordingly"   1
"across"        1
"adding"        1
"adjective"     1
"advent"        1
"advertising"   1
"already"       1
"also"  5
```

```
"hence" 1
"hidden"        2
"hosted"        1
"how"   3
"however"       3
"hyphen"        1
"hyphenated"    1
"important"     1
"importantly"   1
"improve"       1
"in"    15
"include"       1
"included"      1
"including"     1
"integer"       1
"is"    19
"it"    2
"its"   3
"jane"  1
"javascript"    1
"job"   1
"journalism"    1
"large" 1
"largely"       1
"least" 1
"legal" 1
"length"        5
"lengths"       2
Removing temp directory /tmp/word_count.root.20240322.185329.473603...
```

EXPERIMENT -2

DATE:

Aim: Implement following Pig operators

a) Distinct operator

b) foreach operator

c) Union operator

- Distinct operator

  distinct_data = DISTINCT student_details;

  Output:

  (1,Rajiv,Reddy,9848022337,Hyderabad)

  (2,siddarth,Battacharya,9848022338,Kolkata)

  (3,Rajesh,Khanna,9848022339,Delhi)

  (4,Preethi,Agarwal,9848022330,Pune)

  (5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar)

  (6,Archana,Mishra,9848022335,Chennai)

- Foreach operator

  foreach_data = FOREACH student_details GENERATE id,age,city;

  Output:

  (1,21,Hyderabad)

  (2,22,Kolkata)

  (3,22,Delhi)

  (4,21,Pune)

  (5,23,Bhuwaneshwar)

  (6,23,Chennai)

  (7,24,trivendram)

  (8,24,Chennai)

- Union operator

student1 = LOAD 'hdfs://localhost:9000/pig_data/student_data1.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);

student2 = LOAD 'hdfs://localhost:9000/pig_data/student_data2.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);

student = UNION student1, student2;

Output:

(1,Rajiv,Reddy,9848022337,Hyderabad)
(2,siddarth,Battacharya,9848022338,Kolkata)
(3,Rajesh,Khanna,9848022339,Delhi)
(4,Preethi,Agarwal,9848022330,Pune)
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,9848022335,Chennai)
(7,Komal,Nayak,9848022334,trivendram)
(8,Bharathi,Nambiayar,9848022333,Chennai)