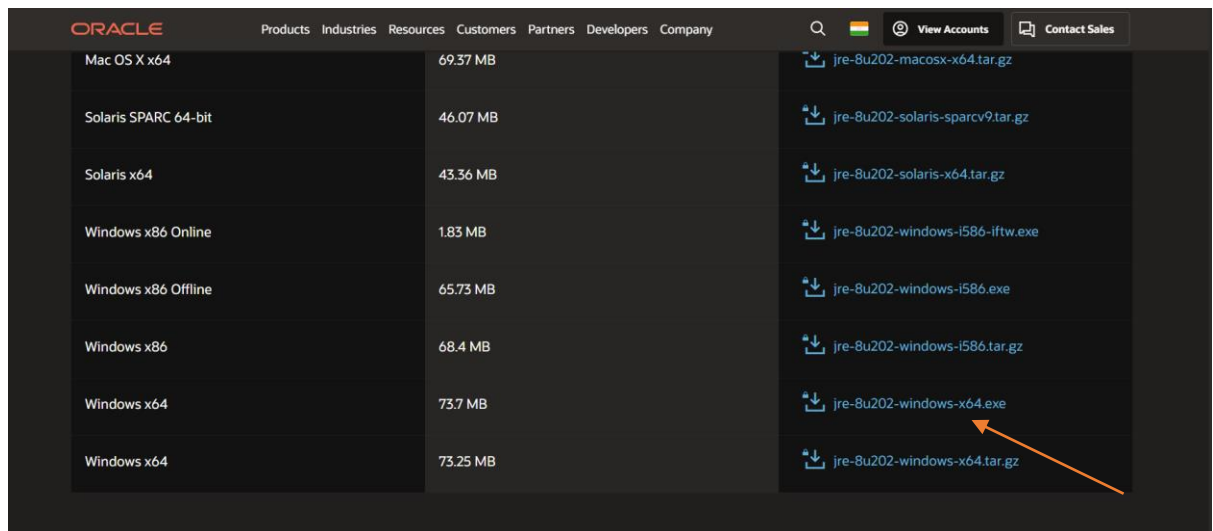


Experiment - 1

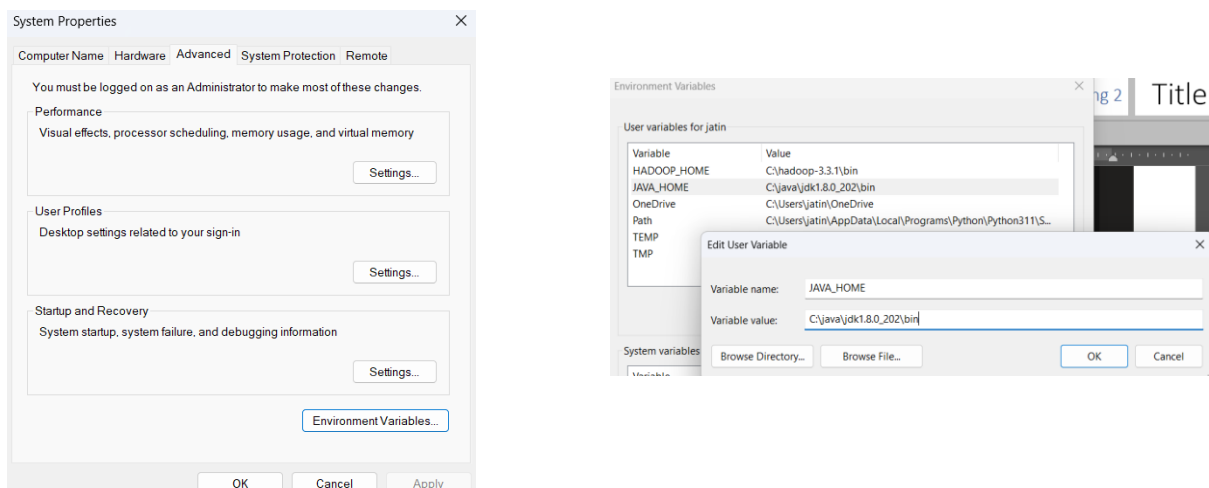
Objective: Installation of Single Node Hadoop Cluster on Windows 11.

Prerequisites

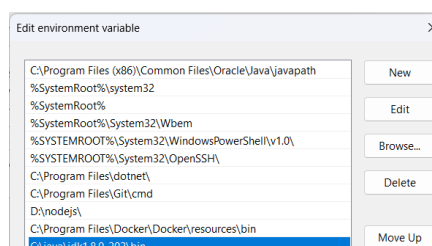
- Download and install **Java JDK (Java 8 or later)** from [Java Archive Downloads - Java SE 8 | Oracle India](#)



- Set `JAVA_HOME`



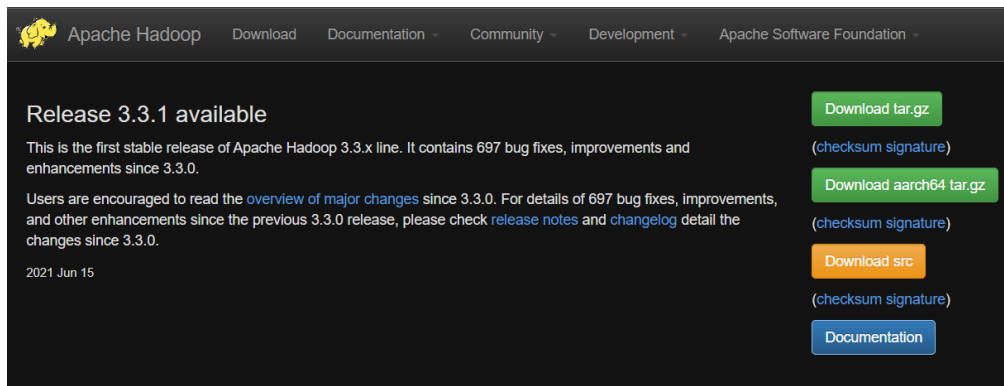
- Add java bin to Path.



Installation Steps

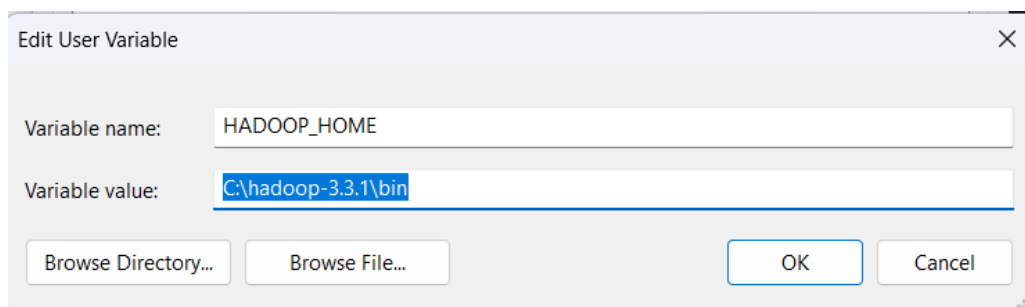
1. Download Hadoop

- Install Hadoop to C:\hadoop-3.3.1

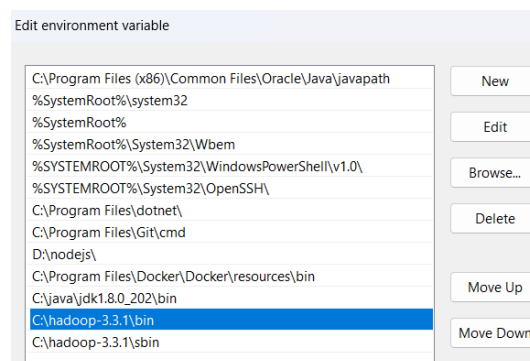


2. Set Environment Variables

- HADOOP_HOME = C:\hadoop



- Add Hadoop-3.3.1 bin to Path



3. Configure Hadoop

- Edit core-site.xml (Set hdfs://localhost:9000)

```
<?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>
```

- Modify `hdfs-site.xml` (Define Namenode & Datanode)

```
<!-- 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-3.3.1\data\namenode</value>
  </property>
  <property>
    <name>dfs.datanode.name.dir</name>
    <value>C:\hadoop-3.3.1\data\datanode</value>
  </property>
</configuration>
```

- Update `mapred-site.xml` (Set it to **local** mode)

```
<?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

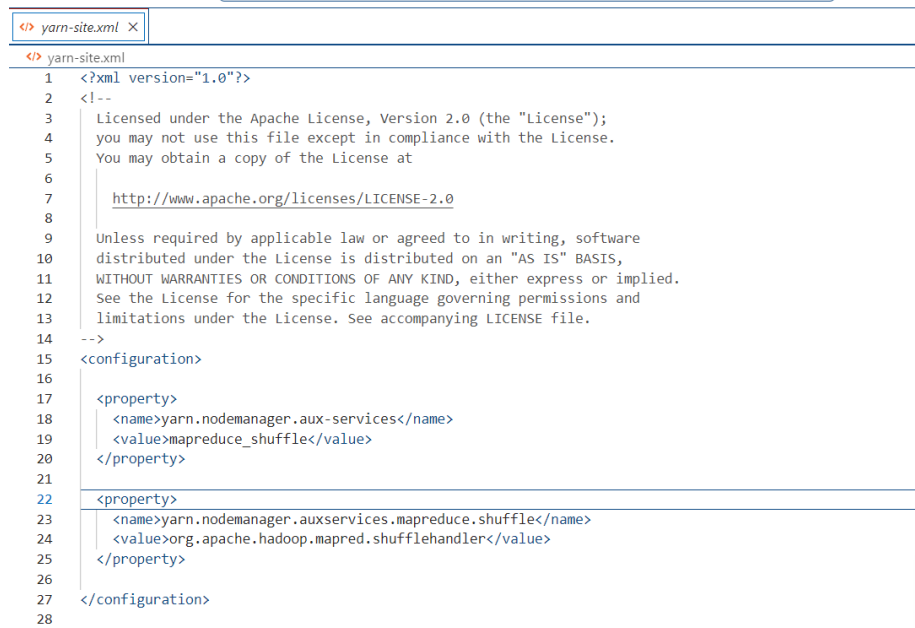
    http://www.apache.org/licenses/LICENSE-2.0

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>
```

- Edit yarn-site.xml (Enable YARN settings)



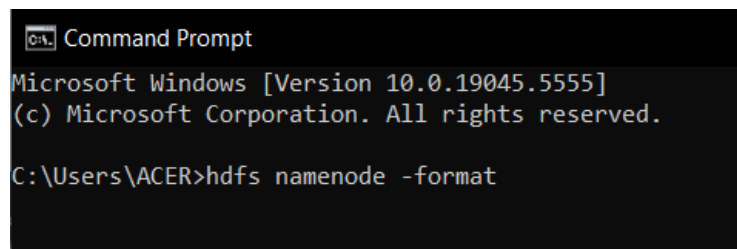
```
<?xml version="1.0"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

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

4. Format Namenode

- Run `hdfs namenode -format`

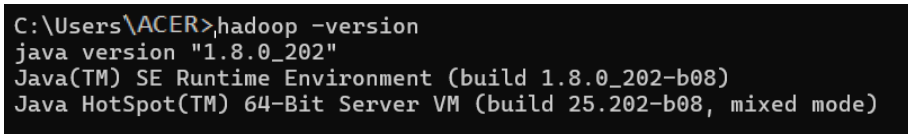


```
Microsoft Windows [Version 10.0.19045.5555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ACER>hdfs namenode -format
```

5. Check Hadoop Installed

- Run `Hadoop -version`



```
C:\Users\ACER>hadoop -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
```

Hence, Hadoop is installed on Windows 11.

Experiment - 2

Objective: Hadoop Programming: Word Count MapReduce Program Using Eclipse.

1. Make a Project Directory as below:

```
wordcount sources root, C:\HadoopPi
├── .idea
├── bin.src
├── input
│   └── input.txt
└── src
    ├── WordCountDriver
    ├── WordCountMapper
    └── WordCountReducer
```

2. Create Input File

- Create a sample text file (e.g., input.txt)

```
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ACER>echo "Hadoop MapReduce Word Count Example" > input.txt
C:\Users\ACER>echo "Hadoop is a framework" >> input.txt
C:\Users\ACER>echo "MapReduce processes big data" >> input.txt
```

- Copy the file to HDFS

```
C:\Users\ACER>hdfs dfs -mkdir /wordcount
```

```
C:\Users\ACER>hdfs dfs -put input.txt /wordcount/
```

- Input file created

```
input.txt x
1 Hello Hadoop
2 Hello World
3 Hadoop is great
4
```

3. Write Word Count MapReduce Program

- Mapper Class (WordCountMapper.java)

```
WordCountMapper.java x
1 package src;
2
3 > import ...
7
1 usage
8 public class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {
9     1 usage
10     private final static IntWritable one = new IntWritable( value: 1);
11     2 usages
12     private Text word = new Text();
13
14     public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
15         String[] words = value.toString().split( regex: "\\s+");
16         for (String w : words) {
17             word.set(w);
18             context.write(word, one);
19         }
20     }
21 }
```

- Reducer Class (WordCountReducer.java)

```
WordCountReducer.java x
1 pa C:\HadoopProjects\wordcount\src\WordCountReducer.java
2
3 import java.io.IOException;
4 import org.apache.hadoop.io.IntWritable;
5 import org.apache.hadoop.io.Text;
6 import org.apache.hadoop.mapreduce.Reducer;
7
8 1 usage
9 public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
10     @f@
11     public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
12         int sum = 0;
13         for (IntWritable val : values) {
14             sum += val.get();
15         }
16         context.write(key, new IntWritable(sum));
17     }
18 }
```

- Driver Class (WordCountDriver.java)

```
WordCountDriver.java x
1 package src;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.fs.Path;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.Job;
8 import org.apache.hadoop.mapreduce.Mapper;
9 import org.apache.hadoop.mapreduce.Reducer;
10 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
12
13 public class WordCountDriver {
14     public static void main(String[] args) throws Exception {
15         Configuration conf = new Configuration();
16         Job job = Job.getInstance(conf, "word count");
17
18         job.setJarByClass(WordCountDriver.class);
19         job.setMapperClass(WordCountMapper.class);
20         job.setReducerClass(WordCountReducer.class);
21
22         job.setOutputKeyClass(Text.class);
23         job.setOutputValueClass(IntWritable.class);
24
25         FileInputFormat.addInputPath(job, new Path(args[0]));
26         FileOutputFormat.setOutputPath(job, new Path(args[1]));
27
28         System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
29     }
30 }
```

4. Compile and Package the Program

- Open terminal in the wordcount directory
- Find Hadoop classpath using the following command:

```
PS C:\HadoopProjects\wordcount> hadoop classpath
C:\hadoop-3.3.1\etc\hadoop\*;C:\hadoop-3.3.1\share\hadoop\common;C:\hadoop-3.3.1\share\hadoop\common\lib\*;C:\hadoop-3.3.1\share\hadoop\common\*;C:\hadoop-3.3.1\share\hadoop\hdfs;C:\hadoop-3.3.1\share\hadoop\hdfs\lib\*;C:\hadoop-3.3.1\share\hadoop\hdfs\*;C:\hadoop-3.3.1\share\hadoop\yarn;C:\hadoop-3.3.1\share\hadoop\yarn\lib\*;C:\hadoop-3.3.1\share\hadoop\yarn\*;C:\hadoop-3.3.1\share\hadoop\mapreduce\*
PS C:\HadoopProjects\wordcount>
```

- Compile the Java files using given classpath in blue

```
PS C:\HadoopProjects\wordcount> javac -classpath "C:\hadoop-3.3.1\share\hadoop\common\*;C:\hadoop-3.3.1\share\hadoop\mapreduce\*;C:\hadoop-3.3.1\share\hadoop\hdfs\*;C:\hadoop-3.3.1\share\hadoop\yarn\*" -d bin src/*.java
PS C:\HadoopProjects\wordcount>
```

- Now, class files are in bin folder
- Create a JAR file:

```
PS C:\HadoopProjects\wordcount> jar -cvf wordcount.jar -C bin .
added manifest
adding: src/(in = 0) (out= 0)(stored 0%)
adding: src/WordCountDriver.class(in = 1386) (out= 757)(deflated 45%)
adding: src/WordCountMapper.class(in = 1787) (out= 771)(deflated 56%)
adding: src/WordCountReducer.class(in = 1606) (out= 673)(deflated 58%)
PS C:\HadoopProjects\wordcount>
```

5. Start Hadoop

- start-dfs.cmd
- start-yarn.cmd

```
PS C:\HadoopProjects\wordcount> start-dfs.cmd
PS C:\HadoopProjects\wordcount> start-yarn.cmd
starting yarn daemons
PS C:\HadoopProjects\wordcount>
```

6. Run the wordcount program using following command

```
PS C:\HadoopProjects\wordcount> hadoop jar wordcount.jar src.WordCountDriver /wordcount /wordcount_output
```

7. View the Output

```
PS C:\HadoopProjects\wordcount> hdfs dfs -cat /wordcount_output/part-r-00000
Hadoop      2
Hello       2
World       1
great       1
is          1
```

Conclusion:

- Successfully implemented **Word Count** using **Hadoop MapReduce**.

Experiment - 3

Objective: Implementing Matrix Multiplication Using One Map Reduce Step.

Step 1: Create Input Matrices

Matrix1:

m1.txt			
1	0	0	1
2	0	1	2
3	1	0	3
4	1	1	4
5	2	0	5
6	2	1	6

Matrix2:

m2.txt			
1	0	0	7
2	0	1	8
3	0	2	9
4	1	0	10
5	1	0	11
6	1	1	12

Step 2: Mapper Class

```
1  import org.apache.hadoop.io.*;
2  import org.apache.hadoop.mapreduce.*;
   1 usage
3  public class MatrixMapper extends Mapper<LongWritable, Text, Text, Text> {
   no usages
4  @ public void map(LongWritable key, Text value, Context context) throws
5  IOException, InterruptedException {
6      String[] tokens = value.toString().split(" ");
7      if (tokens[0].equals("A")) {
8          for (int j = 0; j < P; j++) {
9              context.write(new Text(tokens[1] + "," + j), new Text("A," + tokens[2] + ","
10                  + tokens[3]));
11          }
12      } else {
13          for (int i = 0; i < M; i++) {
14              context.write(new Text(i + "," + tokens[2]), new Text("B," + tokens[1] + ","
15                  + tokens[3]));
16          }
17      }
18  }
19 }
```


Step 3: Reducer Class

The Reducer processes the emitted pairs to compute the matrix multiplication result.

```
2 import org.apache.hadoop.mapreduce.*;
3 import java.util.*;
4
5 public class MatrixReducer extends Reducer<Text, Text, Text, IntWritable> {
6     @Override
7     public void reduce(Text key, Iterable<Text> values, Context context) throws
8         IOException, InterruptedException {
9         HashMap<Integer, Integer> mapA = new HashMap<>();
10        HashMap<Integer, Integer> mapB = new HashMap<>();
11
12        for (Text val : values) {
13            String[] tokens = val.toString().split(",");
14            if (tokens[0].equals("A")) {
15                mapA.put(Integer.parseInt(tokens[1]), Integer.parseInt(tokens[2]));
16            } else {
17                mapB.put(Integer.parseInt(tokens[1]), Integer.parseInt(tokens[2]));
18            }
19        }
20        int sum = 0;
21        for (int k : mapA.keySet()) {
22            if (mapB.containsKey(k)) {
23                sum += mapA.get(k) * mapB.get(k);
24            }
25        }
26        context.write(key, new IntWritable(sum));
27    }
28 }
```

Step 4: Driver Class

```
1 import org.apache.hadoop.conf.*;
2 import org.apache.hadoop.fs.*;
3 import org.apache.hadoop.io.*;
4 import org.apache.hadoop.mapreduce.*;
5 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
6 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
7
8 public class MatrixMultiplicationDriver {
9     @Override
10    public static void main(String[] args) throws Exception {
11        Configuration conf = new Configuration();
12        Job job = Job.getInstance(conf, "Matrix Multiplication");
13
14        job.setJarByClass(MatrixMultiplicationDriver.class);
15        job.setMapperClass(MatrixMapper.class);
16        job.setReducerClass(MatrixReducer.class);
17        job.setOutputKeyClass(Text.class);
18        job.setOutputValueClass(Text.class);
19        FileInputFormat.addInputPath(job, new Path(args[0]));
20        FileOutputFormat.setOutputPath(job, new Path(args[1]));
21        System.exit(job.waitForCompletion(true) ? 0 : 1);
22    }
23 }
```

Step 5: Compile and run the file

```
PS C:\HadoopProjects\matrixmultiplication\multi\src> nano MatrixMapper.java
```

```
PS C:\HadoopProjects\matrixmultiplication\multi\src> nano MatrixReducer.java
```

```
PS C:\HadoopProjects\matrixmultiplication\multi\src> nano MatrixMultiplicationDriver.java
```

Compile the files

```
PS C:\HadoopProjects\matrixmultiplication\multi\src> javac -classpath "%HADOOP_CLASSPATH%" -d . MatrixMapper.java MatrixReducer.java MatrixMultiplicationDriver.java
```

Create a Jar File

```
PS C:\HadoopProjects\matrixmultiplication\multi\src> jar cf MatrixMultiplication.jar *.class
```

Run the Hadoop job

```
PS C:\HadoopProjects\matrixmultiplication\multi\src> hadoop jar MatrixMultiplication.jar MatrixMultiplicationDriver /input /output
```

View output:

```
PS C:\HadoopProjects\matrixmultiplication\multi\src> hdfs dfs -ls /output
```

```
PS C:\HadoopProjects\matrixmultiplication\multi\src> hdfs dfs -cat /output/part-r-000000
0,0      15
0,1      17
1,0      19
1,1      21
```

Experiment 4

4. Implementing Relational Algorithm on Pig

Objective:

To install, configure, and execute Apache Pig to perform relational operations (Selection, Projection, Join, Group By, Order By) on a dataset using Hadoop.

Prerequisites:

- Hadoop Installed & Running
- Java 8 or above installed

Step 1: Install Apache Pig

1.1 Install Pig using Homebrew

```
brew install pig
```

1.2 Verify Installation

```
pig -version
```

Step 2: Configure Apache Pig

2.1 Set Environment Variables

Open .zshrc file and add the following lines

```
nano ~/.zshrc
```

Add the following content:

```
export PIG_HOME=/opt/homebrew/Cellar/pig/<version>/libexec
```

```
export PATH=$PIG_HOME/bin:$PATH
```

```
export PIG_CLASSPATH=$HADOOP_HOME/conf
```

Save:

```
source ~/.zshrc
```

Step 3: Start Hadoop and Pig in Local Mode

```
start-dfs.sh
```

```
start-yarn.sh
```

```
pig -x local
```

Step 4: Prepare Input Data for Relational Operations

Create a sample dataset:

```
nano employee.txt
```

Add the following content:

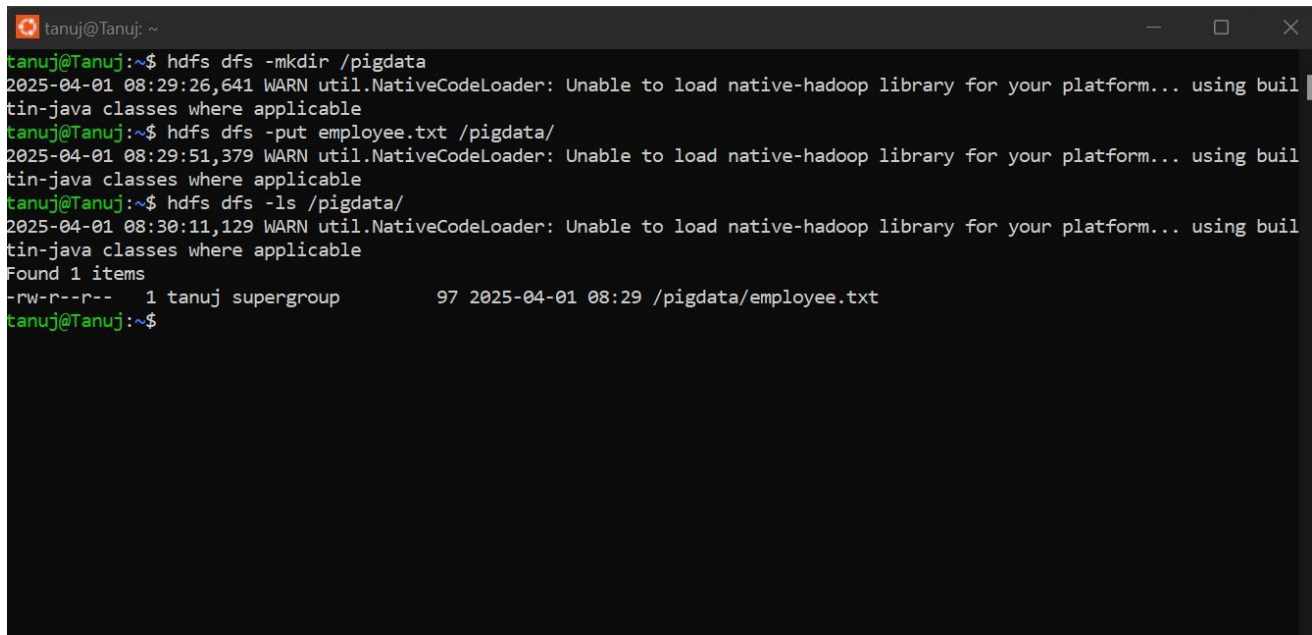
101,John,Sales,45000

102,Alice,HR,55000

103,Bob,Sales,40000

104,David,HR,60000

105,Eve,IT,70000

A terminal window titled 'tanuj@Tanuj: ~' showing a series of HDFS commands and their outputs. The commands are: 'hdfs dfs -mkdir /pigdata', 'hdfs dfs -put employee.txt /pigdata/', and 'hdfs dfs -ls /pigdata/'. The outputs show warnings about native-hadoop library loading and the successful creation and upload of the file 'employee.txt' to the '/pigdata/' directory. The file's permissions are shown as '-rw-r--r--' and its size as 97 bytes.

```
tanuj@Tanuj:~$ hdfs dfs -mkdir /pigdata
2025-04-01 08:29:26,641 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in-java classes where applicable
tanuj@Tanuj:~$ hdfs dfs -put employee.txt /pigdata/
2025-04-01 08:29:51,379 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in-java classes where applicable
tanuj@Tanuj:~$ hdfs dfs -ls /pigdata/
2025-04-01 08:30:11,129 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in-java classes where applicable
Found 1 items
-rw-r--r-- 1 tanuj supergroup          97 2025-04-01 08:29 /pigdata/employee.txt
tanuj@Tanuj:~$
```

Step 5 : Perform Relational Operations on Pig

5.1 Load Data into Pig

```
emp = LOAD 'hdfs://localhost:9000/pigdata/employee.txt'
      USING PigStorage(',') AS (id:int, name:chararray, dept:chararray, salary:int);
```

5.2 Selection (Filter Employees with Salary > 50,000)

```
high_salary = FILTER emp BY salary > 50000;
DUMP high_salary;
```

5.3 Projection (Select only name and Salary)

```
name_salary = FOREACH emp GENERATE name, salary;
DUMP name_salary;
```

5.4 Group by department

```
grouped_by_dept = GROUP emp BY dept;
DUMP grouped_by_dept;
```

5.5 Order by salary descending

```
ordered_salary = ORDER emp BY salary DESC;  
  
DUMP ordered_salary;
```

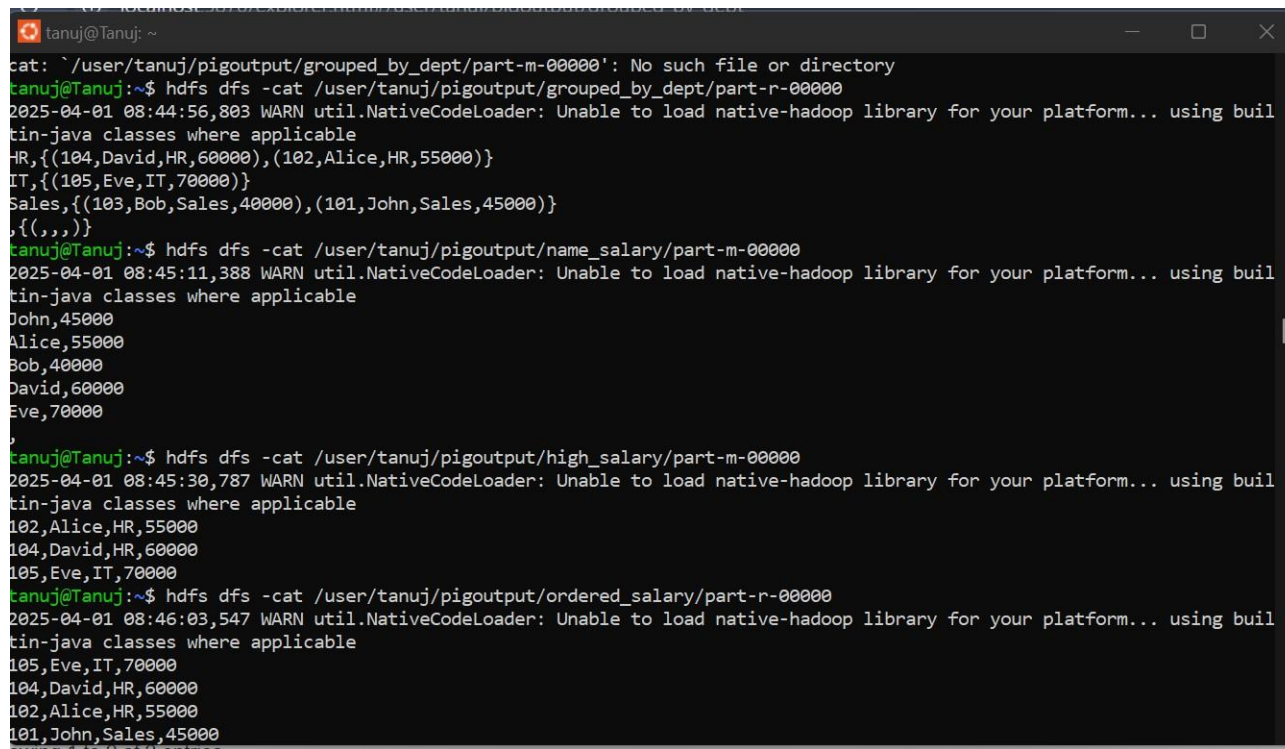
Step 6: Storing the Output

Now, we will store the output to HDFS for further analysis.

```
STORE grouped_by_dept INTO  
'hdfs://localhost:9000/user/tanuj/pigoutput/grouped_by_dept USING PigStorage(',');  
  
STORE name_salary INTO 'hdfs://localhost:9000/user/tanuj/pigoutput/name_salary  
USING PigStorage(',');  
  
STORE high_salary INTO 'hdfs://localhost:9000/user/tanuj/pigoutput/high_salary  
USING PigStorage(',');  
  
STORE ordered_salary INTO  
'hdfs://localhost:9000/user/tanuj/pigoutput/ordered_salary USING PigStorage(',');
```

Step 7: Verify the Output

Now, verify the results using the following HDFS commands:



```
tanuj@Tanuj: ~  
cat: `/user/tanuj/pigoutput/grouped_by_dept/part-m-00000': No such file or directory  
tanuj@Tanuj:~$ hdfs dfs -cat /user/tanuj/pigoutput/grouped_by_dept/part-r-00000  
2025-04-01 08:44:56,803 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable  
HR,{(104,David,HR,60000),(102,Alice,HR,55000)}  
IT,{(105,Eve,IT,70000)}  
Sales,{(103,Bob,Sales,40000),(101,John,Sales,45000)}  
,{(,,),}  
tanuj@Tanuj:~$ hdfs dfs -cat /user/tanuj/pigoutput/name_salary/part-m-00000  
2025-04-01 08:45:11,388 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable  
John,45000  
Alice,55000  
Bob,40000  
David,60000  
Eve,70000  
,  
tanuj@Tanuj:~$ hdfs dfs -cat /user/tanuj/pigoutput/high_salary/part-m-00000  
2025-04-01 08:45:30,787 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable  
102,Alice,HR,55000  
104,David,HR,60000  
105,Eve,IT,70000  
tanuj@Tanuj:~$ hdfs dfs -cat /user/tanuj/pigoutput/ordered_salary/part-r-00000  
2025-04-01 08:46:03,547 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable  
105,Eve,IT,70000  
104,David,HR,60000  
102,Alice,HR,55000  
101,John,Sales,45000
```

Experiment 5

5. Implementing Database operation on Hive

Objective: To understand and implement basic database operations using Apache Hive, including database creation, table creation, data insertion, retrieval, updating, and deletion.

Prerequisites:

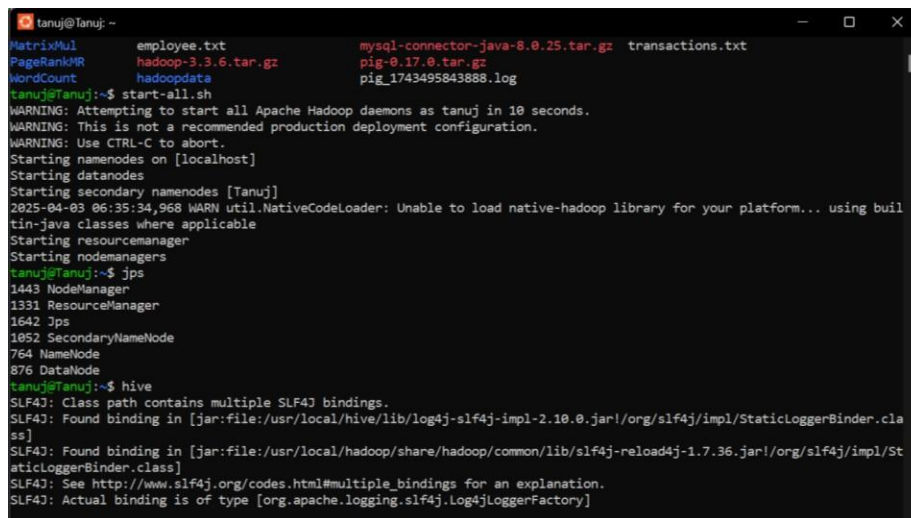
- Hadoop should be installed and running.
- Hive should be properly configured.
- Metastore should be initialized and running.

Step 1: Start Hadoop and Hive Services Before using Hive, ensure that Hadoop and Hive services are running:

start-dfs.sh

start-yarn.sh

hive --service metastore &

A terminal window titled 'tanuj@Tanuj: ~' showing the execution of Hadoop and Hive services. The user runs 'start-all.sh', which starts the Hadoop NameNode, DataNodes, and Yarn services. The output shows various warnings and status messages. After running 'jps', the user sees the running processes: NodeManager, ResourceManager, Jps, SecondaryNameNode, NameNode, and DataNode. Finally, the user runs 'hive', which shows SLF4J logging warnings and confirms the binding of the Hive logging framework.

```
tanuj@Tanuj: ~  
MatrixMul      employee.txt      mysql-connector-java-8.0.25.tar.gz transactions.txt  
PageRankMR     hadoop-3.3.6.tar.gz pig-0.17.0.tar.gz  
WordCount      hadoopdata      pig_1743495843888.log  
tanuj@Tanuj:~$ start-all.sh  
WARNING: Attempting to start all Apache Hadoop daemons as tanuj in 10 seconds.  
WARNING: This is not a recommended production deployment configuration.  
WARNING: Use CTRL-C to abort.  
Starting namenodes on [localhost]  
Starting datanodes  
Starting secondary namenodes [Tanuj]  
2025-04-03 06:35:34,968 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
Starting resourcemanager  
Starting nodemanagers  
tanuj@Tanuj:~$ jps  
1443 NodeManager  
1331 ResourceManager  
1642 Jps  
1052 SecondaryNameNode  
764 NameNode  
876 DataNode  
tanuj@Tanuj:~$ hive  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

Step 2: Creating a Database To create a database in Hive, use the following command:

CREATE DATABASE IF NOT EXISTS company;

To use the created database:

USE company;

Step 3: Creating Tables Creating an employee table:

*CREATE TABLE employees (
id INT,*

```
    name STRING,  
    age INT,  
    department STRING,  
    salary FLOAT  
)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ',';
```

Step 4: Loading Data into the Table Assuming the data file employees.csv exists in HDFS at /user/hive/data/:

```
hdfs dfs -put employees.csv /user/hive/data/
```

Load the data into the table:

```
LOAD DATA INPATH '/user/hive/data/employees.csv' INTO TABLE employees;
```

Step 5: Querying the Table To retrieve all records:

```
SELECT * FROM employees;
```



emp_id	name	age	dept_name	
1	Alice	25	HR	
2	Bob	30	IT	
3	Charlie	28	HR	
4	David	35	Finance	
5	Eva	27	IT	

Step 6: Updating and Deleting Data Hive does not support direct row updates/deletes in traditional tables, but we can use INSERT OVERWRITE or ACID Transactions (if enabled).

Updating salary for an employee (using INSERT OVERWRITE):

```
INSERT OVERWRITE TABLE employees SELECT id, name, age, department,  
    CASE WHEN id > 101 THEN 75000 ELSE salary END  
FROM employees;
```

emp_id	name	age	dept_id
2	Bob	30	102
4	David	35	103

Step 7: Dropping Tables and Databases To delete a table:

DROP TABLE employees;

To delete the database:

DROP DATABASE company CASCADE;

Experiment 8

8. Implementing Clustering algorithm using MapReduce.

Objective

To implement a clustering algorithm using the MapReduce framework in Hadoop. This experiment focuses on using the K-Means clustering algorithm to classify data points into clusters.

Requirements:

- Ubuntu (running on Windows WSL or Virtual Machine)
- Hadoop setup
- Java Development Kit (JDK)

Theory

Clustering is an unsupervised learning technique used to group similar data points into clusters. K-Means clustering is one of the most widely used clustering techniques. The algorithm follows these steps:

1. Select K initial cluster centroids.
2. Assign each data point to the nearest centroid.
3. Compute new centroids by averaging points in each cluster.
4. Repeat steps 2 and 3 until centroids stabilize.

MapReduce efficiently implements clustering by distributing computations across multiple nodes.

Implementation Steps:

1. Create Input Data File

Create a text file points.txt with sample data points:

```
nano points.txt
```

Upload it to HDFS:

```
hdfs dfs -mkdir /input
```

```
hdfs dfs -put points.txt /input/
```

```
tanuj@Tanuj: ~/Clustering
3,4 2
tanuj@Tanuj:~/FrequentItemset$ cd ..
tanuj@Tanuj:~$ mkdir Clustering
tanuj@Tanuj:~$ cd Clustering
tanuj@Tanuj:~/Clustering$ nano points.txt
tanuj@Tanuj:~/Clustering$ hdfs dfs -put points.txt /input/
2025-02-25 09:25:13,293 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
tanuj@Tanuj:~/Clustering$ nano KMeansMapper.java
tanuj@Tanuj:~/Clustering$ nano KMeansReducer.java
tanuj@Tanuj:~/Clustering$ nano KMeansDriver.java
tanuj@Tanuj:~/Clustering$ javac -classpath $HADOOP_CLASSPATH -d . KMeansMapper.java
tanuj@Tanuj:~/Clustering$ javac -classpath $HADOOP_CLASSPATH -d . KMeansReducer.java
tanuj@Tanuj:~/Clustering$ javac -classpath $HADOOP_CLASSPATH -d . KMeansDriver.java
tanuj@Tanuj:~/Clustering$ jar cf kmeans.jar *.class
tanuj@Tanuj:~/Clustering$ javac: invalid flag: .
Usage: javac <options> <source files>
use -help for a list of possible options
tanuj@Tanuj:~/Clustering$ javac -classpath $HADOOP_CLASSPATH -d . KMeansReducer.java
tanuj@Tanuj:~/Clustering$ javac: invalid flag: .
Usage: javac <options> <source files>
use -help for a list of possible options
tanuj@Tanuj:~/Clustering$ javac -classpath $HADOOP_CLASSPATH -d . KMeansDriver.java
tanuj@Tanuj:~/Clustering$ javac: invalid flag: .
Usage: javac <options> <source files>
use -help for a list of possible options
tanuj@Tanuj:~/Clustering$ jar cf kmeans.jar *.class
tanuj@Tanuj:~/Clustering$ javac -classpath 'hadoop classpath' -d . *.java
tanuj@Tanuj:~/Clustering$ jar cf kmeans.jar *.class
```

2. Write the K-Means Mapper (KMeansMapper.java)

```
import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

import java.io.IOException;

public class KMeansMapper extends Mapper<LongWritable, Text, IntWritable, Text> {

    public void map(LongWritable key, Text value, Context context) throws
    IOException, InterruptedException {

        String line = value.toString();

        if (line.trim().isEmpty()) return;

        String[] point = line.split(",");

        double x = Double.parseDouble(point[0]);

        double y = Double.parseDouble(point[1]);

        int clusterId = (x + y) > 10 ? 1 : 0;

        context.write(new IntWritable(clusterId), new Text(line)); } }
```

3. Write the K-Means Reducer (KMeansReducer.java)

```
import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

import java.io.IOException;

import java.util.ArrayList;
```

```

public class KMeansReducer extends Reducer<IntWritable, Text, IntWritable, Text>
{
    public void reduce(IntWritable key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
        double sumX = 0, sumY = 0;
        int count = 0;
        ArrayList<String> points = new ArrayList<>();
        for (Text value : values) {
            String[] coords = value.toString().split(",");
            double x = Double.parseDouble(coords[0]);
            double y = Double.parseDouble(coords[1]);
            sumX += x;
            sumY += y;
            count++;
            points.add(value.toString());
        }
        double centroidX = sumX / count;
        double centroidY = sumY / count;
        context.write(key, new Text("Centroid: " + centroidX + "," + centroidY + "
Points: " + points));}
}

```

4. Write the Driver Program (KMeansClustering.java)

```

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class KMeansClustering {
    public static void main(String[] args) throws Exception {
        Job job = Job.getInstance();
        job.setJarByClass(KMeansClustering.class);
        job.setJobName("K-Means Clustering");
    }
}

```

```

        job.setMapperClass(KMeansMapper.class);

        job.setReducerClass(KMeansReducer.class);

        job.setOutputKeyClass(IntWritable.class);

        job.setOutputValueClass(Text.class);

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

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

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

    }

}

```

5. Compile and Run

Compile Java files:

```

hadoop com.sun.tools.javac.Main *.java

jar cf kmeans.jar *.class

```

Run MapReduce Job:

```

hadoop jar kmeans.jar KMeansClustering /input/points.txt /output

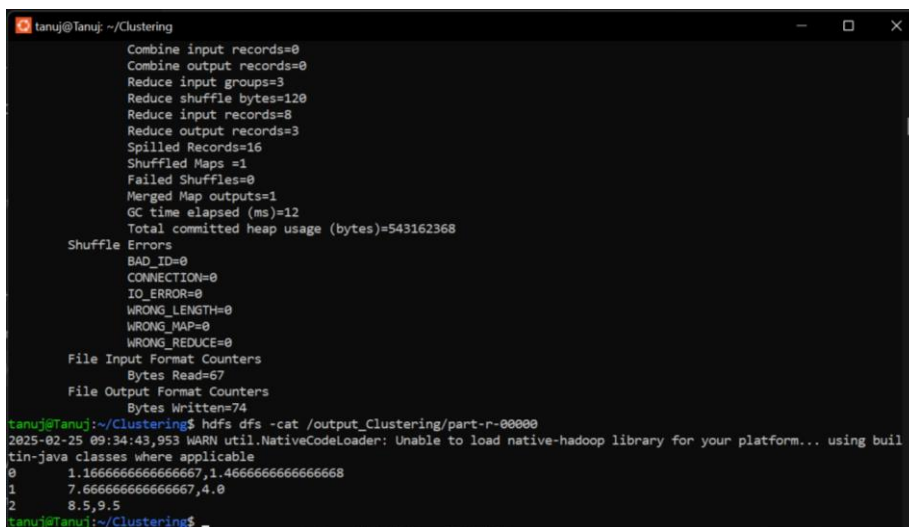
```

6. View Results

```

hdfs dfs -cat /output/part-r-00000

```



```

tanuj@Tanuj: ~/Clustering
Combine input records=0
Combine output records=0
Reduce input groups=3
Reduce shuffle bytes=120
Reduce input records=8
Reduce output records=3
Spilled Records=16
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=12
Total committed heap usage (bytes)=543162368

Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

File Input Format Counters
Bytes Read=67
File Output Format Counters
Bytes Written=74
tanuj@Tanuj:~/Clustering$ hdfs dfs -cat /output_Clustering/part-r-00000
2025-02-25 09:34:43,953 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
0 1.1666666666666667,1.4666666666666668
1 7.666666666666667,4.0
2 8.5,9.5
tanuj@Tanuj:~/Clustering$

```

Experiment 6

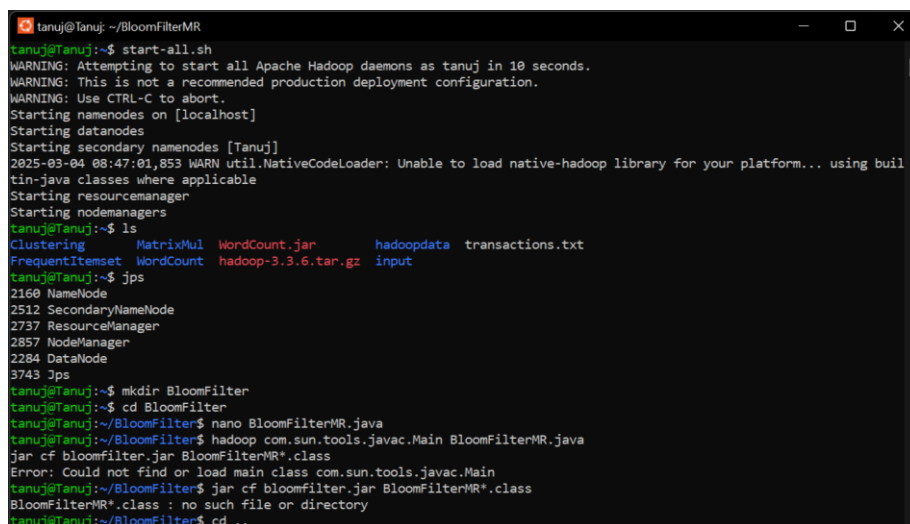
6. Implementing Bloom Filter using MapReduce

Objective

The goal of this lab is to implement a Bloom Filter using MapReduce in Hadoop. A Bloom Filter is a space-efficient probabilistic data structure that tests whether an element is a probable member of a set. The filter may return false positives but never false negatives.

Steps to Implement Bloom Filter in MapReduce

1. Create a directory for the project
2. Create and write the Mapper, Reducer, and Driver Java classes. These classes will perform Bloom filtering on input data.



```
tanuj@Tanuj: ~/BloomFilterMR
tanuj@Tanuj:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as tanuj in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Tanuj]
2025-03-04 08:47:01,853 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
Starting resourcemanager
Starting nodemanagers
tanuj@Tanuj:~$ ls
Clustering  MatrixMul  WordCount.jar  hadoopdata  transactions.txt
FrequentItemset  WordCount  hadoop-3.3.6.tar.gz  input
tanuj@Tanuj:~$ jps
2160 NameNode
2512 SecondaryNameNode
2737 ResourceManager
2857 NodeManager
2284 DataNode
3743 Jps
tanuj@Tanuj:~$ mkdir BloomFilter
tanuj@Tanuj:~$ cd BloomFilter
tanuj@Tanuj:~/BloomFilter$ nano BloomFilterMR.java
tanuj@Tanuj:~/BloomFilter$ hadoop com.sun.tools.javac.Main BloomFilterMR.java
jar cf bloomfilter.jar BloomFilterMR*.class
Error: Could not find or load main class com.sun.tools.javac.Main
tanuj@Tanuj:~/BloomFilter$ jar cf bloomfilter.jar BloomFilterMR*.class
BloomFilterMR*.class : no such file or directory
tanuj@Tanuj:~/BloomFilter$ cd ..
```

1. BloomFilterMapper.java (Mapper Class)

```
import java.io.IOException;

import java.util.BitSet;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

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

    private static final int BLOOM_FILTER_SIZE = 1000; // Size of Bloom Filter

    private BitSet bloomFilter = new BitSet(BLOOM_FILTER_SIZE);

    // Hash Function
```

```

private int hashFunction(int value) {
    return Math.abs(value % BLOOM_FILTER_SIZE);
}

@Override
protected void setup(Context context) throws IOException, InterruptedException {
    // Preloading Bloom Filter with predefined numbers
    int[] predefinedNumbers = {2, 4, 6, 8, 10}; // Numbers stored in Bloom filter
    for (int num : predefinedNumbers) {
        bloomFilter.set(hashFunction(num));
    }
}

@Override
public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
    String line = value.toString().trim();
    if (line.isEmpty()) {
        return; // Skip empty lines
    }
    try {
        int number = Integer.parseInt(line);
        if (bloomFilter.get(hashFunction(number))) {
            context.write(new IntWritable(number), new Text("Present"));
        } else {
            context.write(new IntWritable(number), new Text("Absent"));
        }
    } catch (NumberFormatException e) {
        System.err.println("Skipping invalid input: " + line);
    }
}

```

2. BloomFilterReducer.java (Reducer Class)

```

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class BloomFilterReducer extends Reducer<IntWritable, Text, IntWritable, Text> {

```

@Override

```
public void reduce(IntWritable key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
    for (Text value : values) {
        context.write(key, value);}}}
```

3. BloomFilterDriver.java (Driver Class)

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class BloomFilterDriver {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: BloomFilterDriver <input path> <output path>");
            System.exit(-1);}
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Bloom Filter Example");
        job.setJarByClass(BloomFilterDriver.class);
        job.setMapperClass(BloomFilterMapper.class);
        job.setReducerClass(BloomFilterReducer.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(Text.class)
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Compilation & Execution Steps

1. Compile the Java Code
2. Create a JAR File
3. Prepare Input Data
4. Upload Input File to HDFS
5. Run the Hadoop Job
6. View the Output

```
tanuj@Tanuj: ~/BloomFilterMR
at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
at java.lang.Class.forName0(Native Method)
at java.lang.Class.forName(Class.java:348)
at org.apache.hadoop.util.RunJar.run(RunJar.java:321)
at org.apache.hadoop.util.RunJar.main(RunJar.java:241)
tanuj@Tanuj:~/BloomFilterMR$ javac -classpath $(hadoop classpath) -d . BloomFilterMapper.java BloomFilterDriver.java
tanuj@Tanuj:~/BloomFilterMR$ find . -name "BloomFilterDriver.class"
./BloomFilterDriver.class
tanuj@Tanuj:~/BloomFilterMR$ jar cf bloomfilter.jar *.class
tanuj@Tanuj:~/BloomFilterMR$ jar tf bloomfilter.jar | grep BloomFilterDriver
BloomFilterDriver.class
tanuj@Tanuj:~/BloomFilterMR$ hdfs dfs -rm -r /output_Bloom
2025-03-04 09:15:20,728 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
Deleted /output_Bloom
tanuj@Tanuj:~/BloomFilterMR$ hadoop jar bloomfilter.jar BloomFilterDriver /input_bloom /output_Bloom
2025-03-04 09:15:25,295 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
2025-03-04 09:15:25,974 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-03-04 09:15:26,041 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-03-04 09:15:26,041 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-03-04 09:15:26,212 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-03-04 09:15:26,391 INFO input.FileInputFormat: Total input files to process : 1
2025-03-04 09:15:26,443 INFO mapreduce.JobSubmitter: number of splits:1
2025-03-04 09:15:24,103 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local168310920_0001
2025-03-04 09:15:24,103 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-03-04 09:15:24,264 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2025-03-04 09:15:24,266 INFO mapred.LocalJobRunner: OutputCommitter set in config null
```

Display the final results:

```
tanuj@Tanuj: ~/BloomFilterMR
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=21
File Output Format Counters
  Bytes Written=96
tanuj@Tanuj:~/BloomFilterMR$ hdfs dfs -cat /output_Bloom/part-r-00000
2025-03-04 09:22:30,929 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
1 Absent
2 Present
3 Absent
4 Present
5 Absent
6 Present
7 Absent
8 Present
9 Absent
10 Present
tanuj@Tanuj:~/BloomFilterMR$
```


Experiment 9

9. Implementing PageRank Algorithm using MapReduce

Objective

To implement the PageRank Algorithm using Hadoop MapReduce and understand how web pages are ranked based on their importance in a network.

Prerequisites

- Basic understanding of Hadoop and MapReduce.
- Hadoop installed and running on the system.

Step 1: Set Up the Working Directory

```
tanuj@Tanuj: ~/PageRankMR
tanuj@Tanuj:~$ mkdir PageRankMR
PageRankMR
tanuj@Tanuj:~$ cd PageRankMR
```

Step 2: Create Input Data File

Step 3: Upload Input Data to HDFS

```
tanuj@Tanuj: ~/PageRankMR
tanuj@Tanuj:~/PageRankMR$ nano input.txt
tanuj@Tanuj:~/PageRankMR$ hdfs dfs -mkdir /input_pagerank
s -put input.txt /input_pagerank

2025-03-11 05:00:58,105 WARN util.NativeCodeLoader: Unable to load nat
tin-java classes where applicable
tanuj@Tanuj:~/PageRankMR$ hdfs dfs -put input.txt /input_pagerank
2025-03-11 05:01:01,971 WARN util.NativeCodeLoader: Unable to load nat
tin-java classes where applicable
```

Step 4: Implement PageRank Algorithm in Java

1. PageRankMapper.java

```
import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class PageRankMapper extends Mapper<Object, Text, Text, Text> {

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

        String[] tokens = value.toString().split(" ");
```

```

if (tokens.length < 2) return;

String page = tokens[0];

StringBuilder outlinks = new StringBuilder();

for (int i = 1; i < tokens.length; i++) {

    context.write(new Text(tokens[i]), new Text(page));

    outlinks.append(tokens[i]).append(" ");}

context.write(new Text(page), new Text("LINKS " + outlinks.toString().trim()));} }

```

2. PageRankReducer.java

```

import java.io.IOException;

import java.util.ArrayList;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

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

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
    InterruptedException {

        ArrayList<String> outlinks = new ArrayList<>();

        ArrayList<String> inlinks = new ArrayList<>();

        for (Text val : values) {

            String value = val.toString();

            if (value.startsWith("LINKS")) {

                for (String link : value.substring(6).split(" ")) {

                    outlinks.add(link);} } else {

                    inlinks.add(value);} }

            context.write(key, new Text("IN " + String.join(" ", inlinks) + " OUT " + String.join(" ",
            outlinks)));} }

```

3. PageRankDriver.java

```

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

```

```

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

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

public class PageRankDriver {

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

        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "Page Rank Calculation");

        job.setJarByClass(PageRankDriver.class);

        job.setMapperClass(PageRankMapper.class);

        job.setReducerClass(PageRankReducer.class);

        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(Text.class);

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

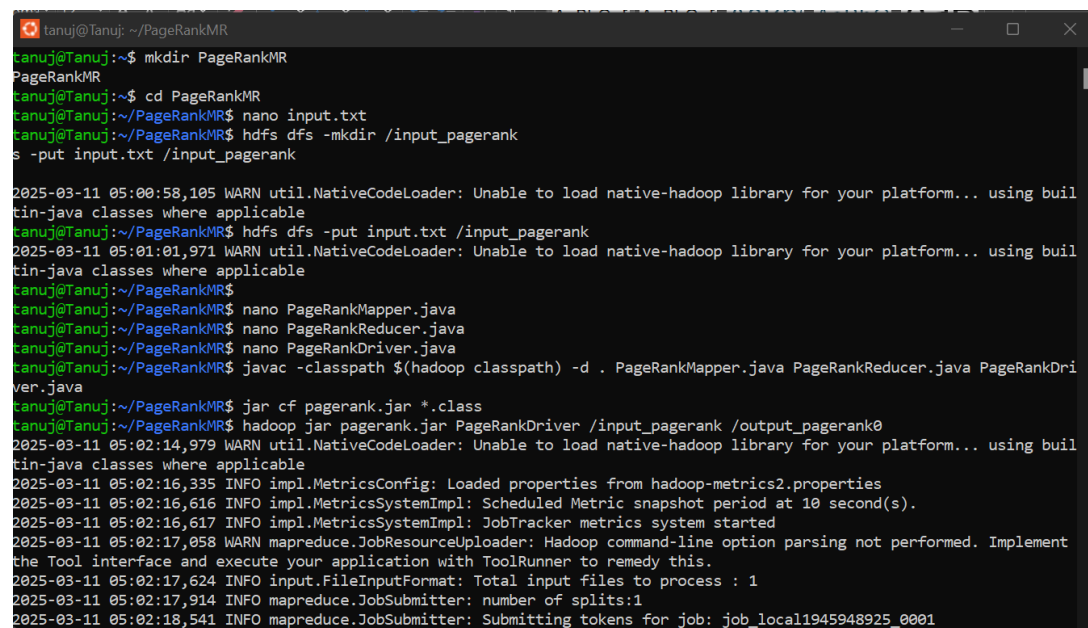
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

```

Step 5: Compile & Run the Program

1. Compile the Java Files
2. Create a JAR File
3. Run the MapReduce Job



```

tanuj@Tanuj: ~/PageRankMR
tanuj@Tanuj:~$ mkdir PageRankMR
PageRankMR
tanuj@Tanuj:~$ cd PageRankMR
tanuj@Tanuj:~/PageRankMR$ nano input.txt
tanuj@Tanuj:~/PageRankMR$ hdfs dfs -mkdir /input_pagerank
s -put input.txt /input_pagerank

2025-03-11 05:00:58,105 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
tanuj@Tanuj:~/PageRankMR$ hdfs dfs -put input.txt /input_pagerank
2025-03-11 05:01:01,971 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
tanuj@Tanuj:~/PageRankMR$
tanuj@Tanuj:~/PageRankMR$ nano PageRankMapper.java
tanuj@Tanuj:~/PageRankMR$ nano PageRankReducer.java
tanuj@Tanuj:~/PageRankMR$ nano PageRankDriver.java
tanuj@Tanuj:~/PageRankMR$ javac -classpath $(hadoop classpath) -d . PageRankMapper.java PageRankReducer.java PageRankDriver.java
tanuj@Tanuj:~/PageRankMR$ jar cf pagerank.jar *.class
tanuj@Tanuj:~/PageRankMR$ hadoop jar pagerank.jar PageRankDriver /input_pagerank /output_pagerank0
2025-03-11 05:02:14,979 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
2025-03-11 05:02:16,335 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-03-11 05:02:16,616 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-03-11 05:02:16,617 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-03-11 05:02:17,058 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-03-11 05:02:17,624 INFO input.FileInputFormat: Total input files to process : 1
2025-03-11 05:02:17,914 INFO mapreduce.JobSubmitter: number of splits:1
2025-03-11 05:02:18,541 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1945948925_0001

```

Step 6: View Results

1. Check Output Files

2. Print the Output

```
tanuj@Tanuj: ~/PageRankMR
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=23
File Output Format Counters
  Bytes Written=64
tanuj@Tanuj:~/PageRankMR$ hdfs dfs -ls /output_pagerank0
2025-03-11 05:02:22,896 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in-java classes where applicable
Found 2 items
-rw-r--r--  1 tanuj supergroup          0 2025-03-11 05:02 /output_pagerank0/_SUCCESS
-rw-r--r--  1 tanuj supergroup        64 2025-03-11 05:02 /output_pagerank0/part-r-00000
tanuj@Tanuj:~/PageRankMR$ hdfs dfs -cat /output_pagerank0/part-r-00000
2025-03-11 05:02:34,598 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in-java classes where applicable
A      IN D C OUT B C
B      IN D A OUT C D
C      IN B A OUT A
D      IN B OUT A B
tanuj@Tanuj:~/PageRankMR$
```

Experiment 7

7. Implementing Frequent Item set algorithm using MapReduce.

Objective

To implement the Frequent Itemset Mining algorithm using Hadoop MapReduce and process transactional datasets to find frequently occurring item combinations.

Theory

Frequent Itemset Mining identifies item groups that frequently appear together in a dataset. The MapReduce framework helps parallelize the computation, making the process efficient for large-scale datasets.

- **Map Phase:** Processes the transaction dataset and emits key-value pairs of itemsets and their counts.
 - **Shuffle & Sort Phase:** Groups itemsets with the same key.
 - **Reduce Phase:** Aggregates the counts of itemsets and filters out those that do not meet the minimum support threshold.
-

Implementation Steps

1. Create the required Java classes:
 - ItemsetMapper.java
 - ItemsetReducer.java
 - PairMapper.java
 - PairReducer.java
 - FrequentItemset.java
2. Compile the Java files.
3. Create a JAR file.
4. Upload the dataset to HDFS.
5. Execute the MapReduce job.
6. View the output results.

```

tanuj@Tanuj: ~/FrequentItemset
tanuj@Tanuj:~/WordCount$ cd ..
tanuj@Tanuj:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as tanuj in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Tanuj]
2025-02-25 09:07:46,064 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
Starting resource manager
Starting node managers
tanuj@Tanuj:~$ jps
1842 ResourceManager
1266 NameNode
1960 NodeManager
1595 SecondaryNameNode
2348 Jps
1390 DataNode
tanuj@Tanuj:~$ nano transactions.txt
tanuj@Tanuj:~$ hdfs dfs -put transactions.txt /input/
2025-02-25 09:13:33,268 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
tanuj@Tanuj:~$ hdfs dfs -ls /input
2025-02-25 09:14:01,256 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
Found 2 items
-rw-r--r-- 1 tanuj supergroup 64 2025-02-23 19:04 /input/cleaned_input.txt
-rw-r--r-- 1 tanuj supergroup 29 2025-02-25 09:13 /input/transactions.txt

```

```

tanuj@Tanuj: ~/FrequentItemset
tanuj@Tanuj:~$ hdfs dfs -rm /input/cleaned_input.txt
2025-02-25 09:14:39,196 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
Deleted /input/cleaned_input.txt
tanuj@Tanuj:~$ mkdir FrequentItemset
cd FrequentItemset
tanuj@Tanuj:~$ cd FrequentItemset
tanuj@Tanuj:~/FrequentItemset$ nano ItemsetMapper.java
tanuj@Tanuj:~/FrequentItemset$ nano ItemsetReducer.java
tanuj@Tanuj:~/FrequentItemset$ nano PairMapper.java
tanuj@Tanuj:~/FrequentItemset$ nano PairReducer.java
tanuj@Tanuj:~/FrequentItemset$ nano FrequentItemset.java
tanuj@Tanuj:~/FrequentItemset$ hadoop com.sun.tools.javac.Main *.java
FrequentItemset.jar *.class
Error: Could not find or load main class com.sun.tools.javac.Main
tanuj@Tanuj:~/FrequentItemset$ jar cf frequentitemset.jar *.class
*.class : no such file or directory
tanuj@Tanuj:~/FrequentItemset$ javac -classpath `hadoop classpath` -d . *.java
tanuj@Tanuj:~/FrequentItemset$ jar cf frequentitemset.jar *.class
tanuj@Tanuj:~/FrequentItemset$ ls
FrequentItemset.class  ItemsetMapper.java  PairMapper.class  PairReducer.java
FrequentItemset.java  ItemsetReducer.class  PairMapper.java  frequentitemset.jar
ItemsetMapper.class  ItemsetReducer.java  PairReducer.class
tanuj@Tanuj:~/FrequentItemset$ hadoop jar frequentitemset.jar FrequentItemset /input/transactions.txt /output/itemcount /output/paircount
2025-02-25 09:16:39,501 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
2025-02-25 09:16:40,587 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-02-25 09:16:40,836 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-02-25 09:16:40,836 INFO impl.MetricsSystemImpl: JobTracker metrics system started

```

Execution Steps

Step 1: Compile Java Code

```
hadoop com.sun.tools.javac.Main *.java
```

Step 2: Create JAR File

```
jar cf frequentitemset.jar *.class
```

Step 3: Upload Data to HDFS

```
hdfs dfs -mkdir /input
```

```
hdfs dfs -put transactions.txt /input/
```

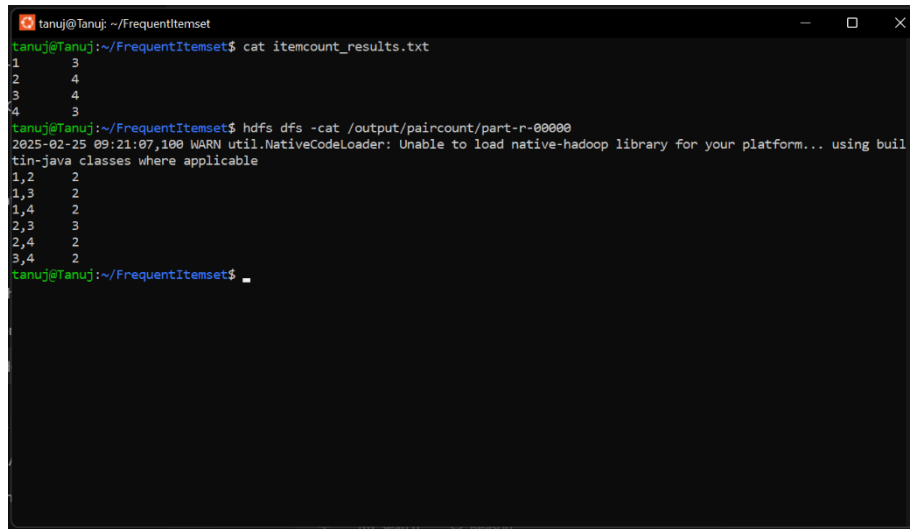
Step 4: Run the MapReduce Job

```
hadoop jar frequentitemset.jar FrequentItemset /input /output
```

Step 5: View Output

```
hdfs dfs -cat /output/part-r-00000
```

7. Expected Outcome



```
tanuj@Tanuj: ~/FrequentItemset
tanuj@Tanuj:~/FrequentItemset$ cat itemcount_results.txt
1      3
2      4
3      4
4      3
tanuj@Tanuj:~/FrequentItemset$ hdfs dfs -cat /output/paircount/part-r-00000
2025-02-25 09:21:07,100 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using built-in java classes where applicable
1,2    2
1,3    2
1,4    2
2,3    3
2,4    2
3,4    2
tanuj@Tanuj:~/FrequentItemset$
```

Value Added Experiment-A

Objective: Introduction to Tableau, Tableau workspace, connecting to a data source

The Tableau Workspace

Applies to: Tableau Cloud, Tableau Desktop, Tableau Server

The Tableau workspace consists of menus, a toolbar, the Data pane, cards and shelves, and one or more sheets. Sheets can be worksheets, dashboards, or stories. For details on dashboard or story workspaces, see [Create a Dashboard](#) or [The Story Workspace](#).

If you are using Tableau on the web, see [Creators: Get Started with Web Authoring and Tour Your Tableau Site](#).

Workspace area



A. Workbook name. A workbook contains sheets. A sheet can be a worksheet, a dashboard, or a story. For more information, see [Workbooks and Sheets](#).

B. Cards and shelves - Drag fields to the cards and shelves in the workspace to add data to your view.

C.Toolbar - Use the toolbar to access commands and analysis and navigation tools.

D.View - This is the canvas in the workspace where you create a visualization (also referred to as a "viz").

E. Click this icon to go to the Start page, where you can connect to data. For more information, see Start Page.

F.Side Bar - In a worksheet, the side bar area contains the Data pane and the Analytics pane.

G. Click this tab to go to the Data Source page and view your data. For more information, see Data Source Page.

H.Status bar - Displays information about the current view.






I. Sheet tabs - Tabs represent each sheet in your workbook. This can include worksheets, dashboards, and stories. For more information, see Workbooks and Sheets.










Tableau Toolbar Button Reference







When you are creating or editing a view, you can use the toolbar at the top of the view to perform common actions.





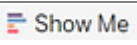
In Tableau Desktop, you can hide or display the Tableau toolbar by selecting Window > Show Toolbar.

The table below explains the functions of each toolbar button. Some buttons are not available in all Tableau products. Also see Visual Cues and Icons in Tableau Desktop.

Toolbar Button	Description
	Tableau icon: Navigates to the start page. For more information, see Start Page. Note: Tableau Desktop only.
	Undo: Reverses the most recent action in the workbook. You can undo an unlimited number of times, back to the last time you opened the workbook, even after you have saved. For more information, see Undo and Redo.
	Redo: Repeats the last action you reversed with the Undo button. You can redo an unlimited number of times.
	Save: In Tableau Desktop, saves the changes made to the workbook. For more information, see Save Your Work. In Tableau Server or Tableau Cloud, click File > Save or File > Save As to save your changes.
	New Data Source: In Tableau Desktop, opens the Connect pane where you can create a new connection or open a saved connection. For more information, see Connect to Your Data.

Toolbar Button	Description
	In Tableau Server or Tableau Cloud, opens the Connect to a Data Source page, where you can connect to a published data source. For more information, see Connect to published data sources while web authoring .
	Pause Auto Updates: Controls whether Tableau updates the view when changes are made. Use the drop-down menu to automatically update the entire sheet or just use filters. For more information, see Refresh Data or Pause Automatic Updates .
	Run Update: Runs a manual query of the data to update the view with changes when automatic updates are turned off. Use the drop-down menu to update the entire worksheet or just use filters. Note: Tableau Desktop only.
	New Worksheet: Creates a new blank worksheet, use the drop-down menu to create a new worksheet, dashboard, or story. For more information, see Create new worksheets, dashboards, or stories .
	Duplicate: Creates a new worksheet containing the same view as the current sheet. For more information, see Duplicate a sheet .
	Clear: Clears the current worksheet. Use the drop-down menu to clear specific parts of the view such as filters, formatting, sizing, and axis ranges.
	Swap: Moves the fields on the Rows shelf to the Columns shelf and vice versa. The Hide Empty Rows and Hide Empty Columns settings are always swapped with this button.
	Sort Ascending: Applies a sort in ascending order of a selected field based on the measures in the view. For more information, see Sort Data in a Visualization (Link opens in a new window).
	Sort Descending: Applies a sort in descending order of a selected field based on the measures in the view. For more information, see Sort Data in a Visualization (Link opens in a new window).
	Totals: You can compute grand totals and subtotals for the data in a view. Select from the following options: <ul style="list-style-type: none"> • Show Column Grand Totals: Adds a row showing totals for all columns in the view. • Show Row Grand Totals: Adds a column showing totals for all rows in the view.

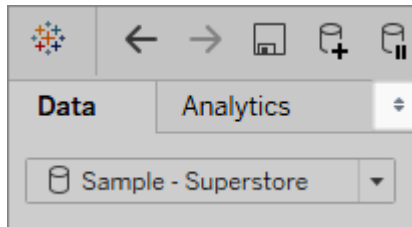
Toolbar Button	Description
	<ul style="list-style-type: none"> Row Totals to Left: Moves rows showing totals to the left of a crosstab or view. Column Totals to Top: Moves columns showing totals to the top of a crosstab or view. Add All Subtotals: Inserts subtotal rows and columns in the view, if you have multiple dimensions in a column or row. Remove All Subtotals: Removes subtotal rows or columns. <p>Note: Tableau Server and Tableau Cloud only. In Tableau Desktop, click Analysis > Totals. For more information, see Show Totals in a Visualization.</p>
	<p>Highlight: Turn on highlighting for the selected sheet. Use the options on the drop-down menu to define how values are highlighted. For more information, see Highlight Toolbar Button.</p>
	<p>Group Members: Creates a group by combining selected values. When multiple dimensions are selected, use the drop-down menu to specify whether to group on a specific dimension or across all dimensions. For more information, see Correct Data Errors or Combine Dimension Members by Grouping Your Data.</p> <p>Note: Tableau Desktop only. In Tableau Server and Tableau Cloud, create groups using the Group Members button on the tooltip.</p>
	<p>Show Mark Labels: Switches between showing and hiding mark labels for the current sheet. For more information, see Show, Hide, and Format Mark Labels.</p>
	<p>Fix Axes: switches between a locked axis that only shows a specific range and a dynamic axis that adjusts the range based on the minimum and maximum values in the view. For more information, see Edit Axes.</p> <p>Note: Tableau Desktop only.</p>
	<p>Format Workbook: Open the Format Workbook pane to change how fonts and titles look in every view in a workbook by specifying format settings at the workbook level instead of at the worksheet level.</p> <p>Note: Tableau Server and Tableau Cloud only. In Tableau Desktop, click Format > Workbook. For more information, see Format at the Workbook Level.</p>
 <div>Standard ▾</div>	<p>Fit: Specifies how the view should be sized within the window. Select Standard, Fit Width, Fit Height, or Entire View. Note: This menu is not available in geographic map views.</p>

Toolbar Button	Description
	The Cell Size commands have different effects depending on the type of visualization. To access the Cell Size menu in Tableau Desktop click Format > Cell Size.
	<p>Show/Hide Cards: Shows and hides specific cards in a worksheet. Select each card that you want to hide or show on the drop-down menu.</p> <p>In Tableau Server and Tableau Cloud, you can show and hide cards for the Title, Caption, Filter and Highlighter only.</p>
	<p>Presentation Mode: Switches between showing and hiding everything except the view (i.e., shelves, toolbar, Data pane). For more information, see Reorganizing the Workspace.</p> <p>Note: Tableau Desktop only.</p>
	<p>Download: Use the options under Download to capture parts of your view for use in other applications.</p> <ul style="list-style-type: none"> • Image: Displays the view, dashboard, or story as an image in a new browser tab. • Data: Displays the data from the view in a new browser window with two tabs: Summary, showing aggregated data for the fields shown in the view, and Underlying, showing underlying data for the selected marks in the visualization. If the new window does not open, you may need to disable your browser's popup blocker. • Crosstab: Saves the underlying data for the selected marks in the visualization to a CSV (comma-separated values) file which can then be opened in Microsoft Excel. • PDF: Opens the current view as a PDF in a new browser window. From there you can save it to a file. If the new window does not open, you may need to disable your browser's pop-up blocker. <p>Note: Tableau Server and Tableau Cloud only.</p>
	<p>Share Workbook With Others: Publish your workbook to Tableau Server or Tableau Cloud. For more information, see Simple Steps to Publish a Workbook.</p> <p>Note: Tableau Desktop only.</p>
	<p>Show Me: Helps you choose a view type by highlighting view types that work best with the field types in your data. An orange outline shows around the recommended chart type that is the best match for your data. For more information, see Use Show Me to Start a View .</p>

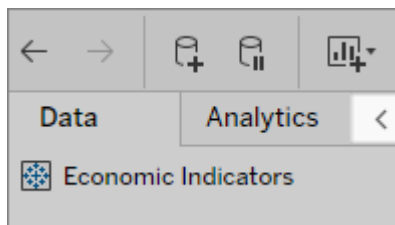
Show and Hide the Side Bar (Data pane)

The Side Bar contains the Data pane and the Analytics pane when you are editing a worksheet. Different panes are visible depending on what you are doing in the view (Data, Analytics, Story, Dashboard, Layout, Format). The most important thing to know about the Side Bar is that you can expand and collapse this area in the workspace.

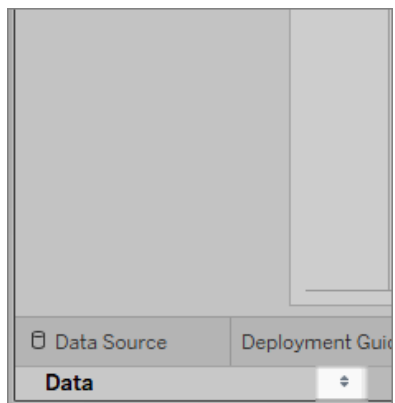
To hide the side bar in Tableau Desktop, click the collapse arrow in the side bar.



To hide the side bar on the web, click the collapse arrow in the side bar.



To show the side bar on Tableau Desktop, click the expand arrow in the bottom-left of the workspace (in the status bar).



To show the side bar on the web, click the expand arrow in the side bar.

