Loyola – ICAM College of Engineering and Technology
(Autonomous)

Loyola Campus, Nungambakkam, Chennai -600034

Approved by AICTE
A Christian Minority Institution
Departments of CSE, IT, ECE, EEE and Mech. are NBA accredited

RECORD NOTEBOOK

Name of the Student	:	……
Register Number	:	……
Year / Semester	:	……
Branch	:	……
Course Code & Name	:	……

Loyola – ICAM College of Engineering and Technology

(Autonomous)

Loyola Campus, Nungambakkam, Chennai -600034

Certificate

Name :

Register Number :

Certified that this is a bonafide record of work done by the candidate in the ..... Semester of B.E. / B.Tech.....in the ..... laboratory during the year .....

Course Instructor **Head of the Department**

This record is submitted for the Semester End Practical Examination held on _____

Internal Examiner **External Examiner**

INDEX

Expt. No.	Date	Title of the Experiment	Page No.	Date Submission	Marks	Staff Initial

Remarks :

Signature of Staff

INDEX

Expt. No.	Date	Title of the Experiment	Page No.	Date Submission	Marks	Staff Initial

Remarks :

Signature of Staff

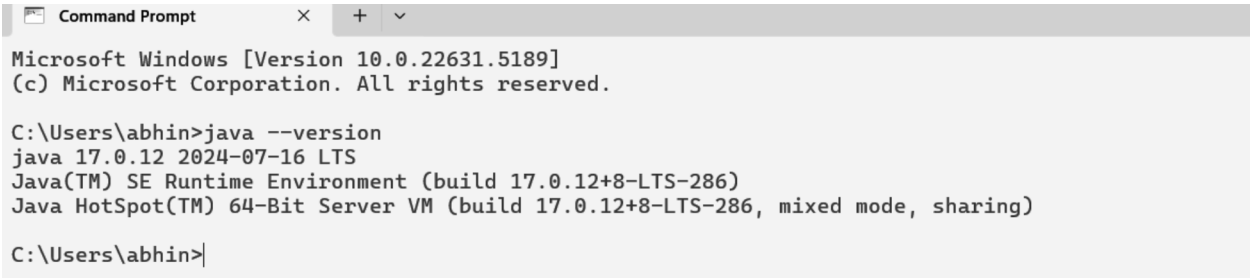
Ex:01 Downloading and installing Hadoop;Understanding different Hadoop modes,Startup scripts,Configuration files.
Date:

AIM: To Install Apache Hadoop.
Hadoop software can be installed in three modes of

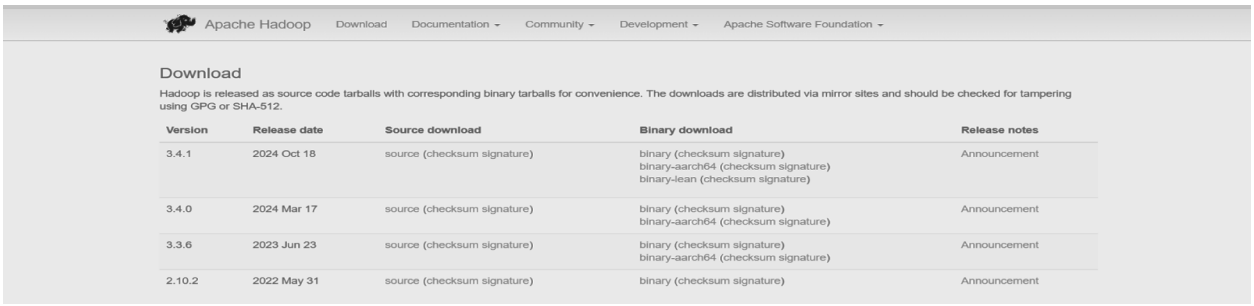
Hadoop is a Java-based programming framework that supports the processing and storage of extremely large datasets on a cluster of inexpensive machines. It was the first major open source project in the big data playing field and is sponsored by the Apache Software Foundation.

Hadoop-2.7.3 is comprised of four main layers:
Hadoop Common is the collection of utilities and libraries that support other Hadoop modules.
HDFS, which stands for Hadoop Distributed File System, is responsible for persisting data to disk.
YARN, short for Yet Another Resource Negotiator, is the "operating system" for HDFS.
MapReduce is the original processing model for Hadoop clusters. It distributes work within the cluster or map, then organizes and reduces the results from the nodes into a response to a query. Many other processing models are available for the 2.x version of Hadoop.
Hadoop clusters are relatively complex to set up, so the project includes a stand-alone mode which is suitable for learning about Hadoop, performing simple operations, and debugging.

Procedure:
1.Check for java version if exist , or else download java version 8.



2.Download hadoop from apache official website and extract it.
Set the environmental variable.



3.Configure the file as follow:

i)core-site.xml

index.html | logins.txt | postrest.py

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.defaulterFS</name>
                <value>hdfs://localhost:900</value>
        </property>
</configuration>
```

ii)mapred-site.xml

core-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>
```

iii)yarn-site.xml

core-site.xml | mapred-site.xml | yarn-site.xn

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</name>
                <value>org.apache.hadoop.mapred.shufflehandler</value>
        </property>

</configuration>
```

iv)hdfs-site.xml

core-site.xml | mapred-site.xml | yarn-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:\Program Files\hadoop-3.4.1\data\namenode</value>
        </property>
        <property>
                <name>dfs.datanode.name.dir</name>
                <value>C:\Program Files\hadoop-3.4.1\data\datanode</value>
        </property></configuration>
```

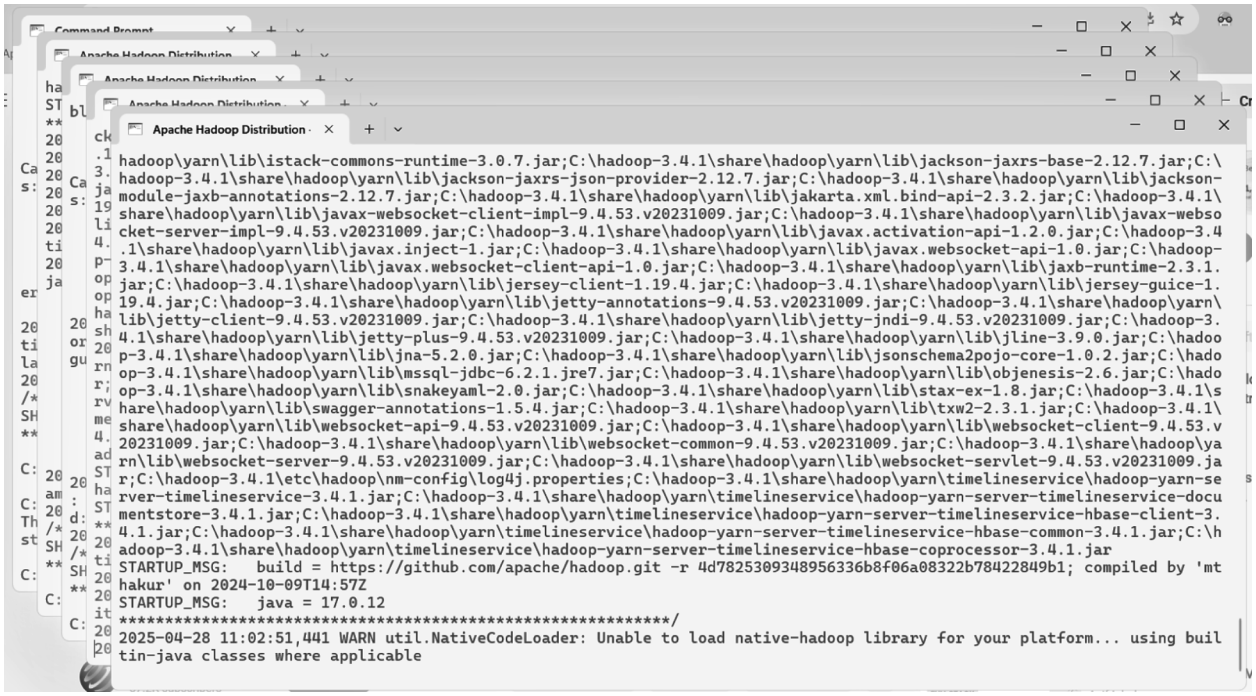
v)hadoop-env.cmd

4.Go to command promt and format the name node

```
C:\Users\abhin>hdfs namenode -format
2025-04-28 11:01:28,422 WARN util.Shell: Did not find winutils.exe: {}
java.io.FileNotFoundException: Could not locate Hadoop executable: C:\hadoop-3.4.1\bin\winutils.exe -see https://cwik
pache.org/confluence/display/HADOOP2/WindowsProblems
        at org.apache.hadoop.util.Shell.getQualifiedBinInner(Shell.java:672)
        at org.apache.hadoop.util.Shell.getQualifiedBin(Shell.java:645)
        at org.apache.hadoop.util.Shell.<clinit>(Shell.java:742)
        at org.apache.hadoop.util.StringUtils.<clinit>(StringUtils.java:80)
        at org.apache.hadoop.hdfs.server.common.HdfsServerConstants$RollingUpgradeStartupOption.getAllOptionString(Ho
erverConstants.java:130)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.<clinit>(NameNode.java:403)
2025-04-28 11:01:28,583 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ABINAYA/172.16.25.234
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.4.1
STARTUP_MSG:   classpath = C:\hadoop-3.4.1\etc\hadoop;C:\hadoop-3.4.1\share\hadoop\common;C:\hadoop-3.4.1\share\ha
ommon\lib\animal-sniffer-annotations-1.17.jar;C:\hadoop-3.4.1\share\hadoop\common\lib\audience-annotations-0.12.0.
\hadoop-3.4.1\share\hadoop\common\lib\avro-1.9.2.jar;C:\hadoop-3.4.1\share\hadoop\common\lib\bcprov-jdk18on-1.78.1
:\hadoop-3.4.1\share\hadoop\common\lib\checker-qual-2.5.2.jar;C:\hadoop-3.4.1\share\hadoop\common\lib\commons-bean
1.9.4.jar;C:\hadoop-3.4.1\share\hadoop\common\lib\commons-cli-1.5.0.jar;C:\hadoop-3.4.1\share\hadoop\common\lib\co
codec-1.15.jar;C:\hadoop-3.4.1\share\hadoop\common\lib\commons-collections-3.2.2.jar;C:\hadoop-3.4.1\share\hadoop\
```

5.Go to command prompt and in the sbin path start all the nodes .

```
C:\Users\abhin>cd C:\hadoop-3.4.1\sbin

C:\hadoop-3.4.1\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop-3.4.1\sbin>
```



Result:

Thus, the installation and the configuration of files in Hadoop is done successfully.

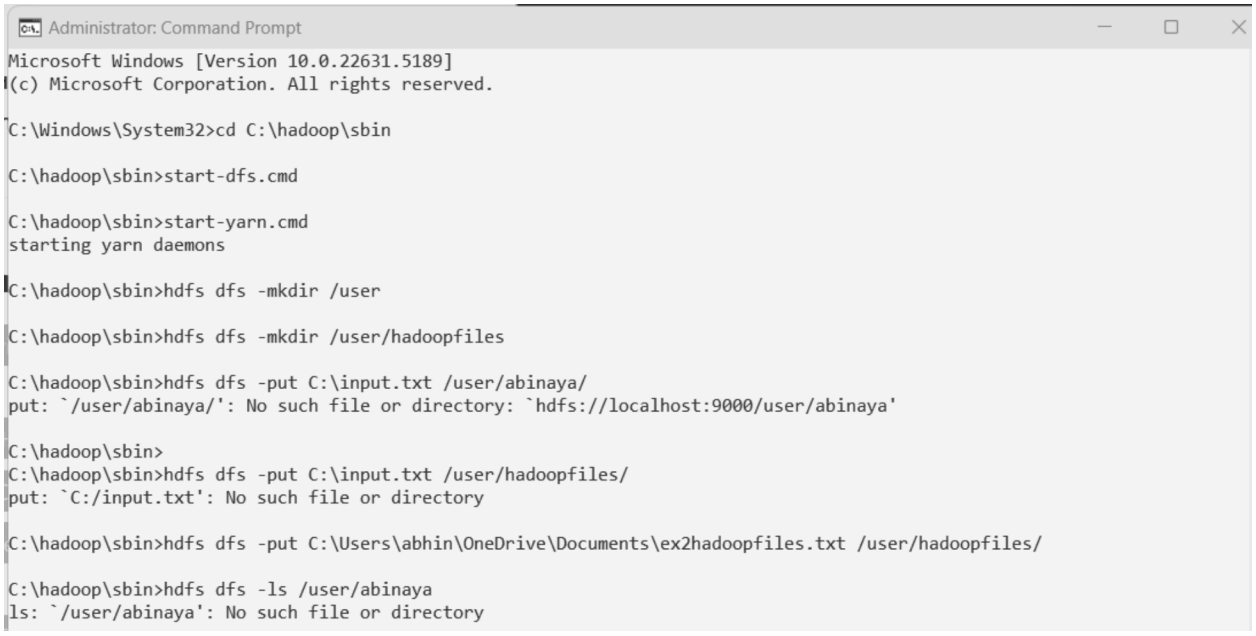
Ex:02 Hadoop Implementation of file management tasks,such as adding files and directories,retrieving files and deleting files.
Date:

Aim:

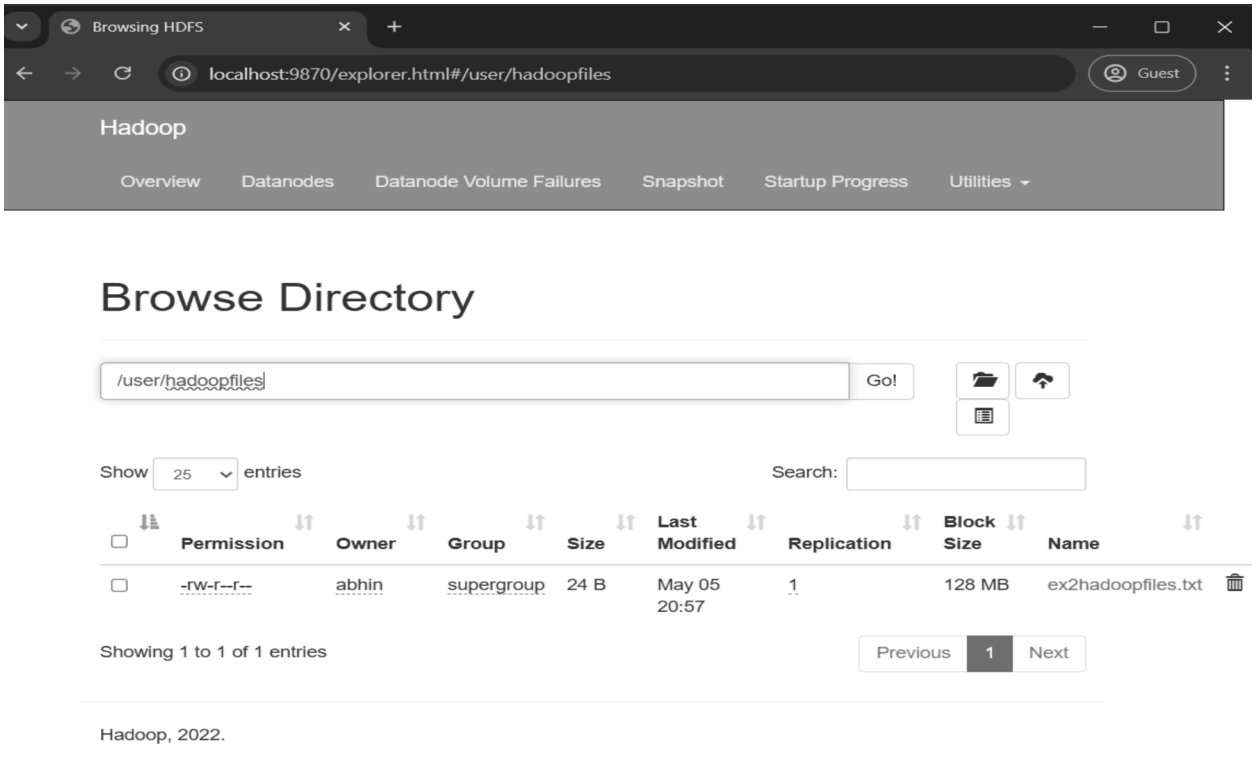
To implement basic file management tasks using Hadoop HDFS on a Windows system, including creating directories, adding files, retrieving files, and deleting files.

Procedure:

1.Open Command Prompt and navigate to the Hadoop sbin directory.
Run start-dfs.cmd to start NameNode and DataNode services.
Run start-yarn.cmd to start ResourceManager and NodeManager services (optional for file ops).
2.Create a directory in HDFS using hdfs dfs -mkdir /user.
3.Create a subdirectory using hdfs dfs -mkdir /user/abinaya.
4.Upload a local file to HDFS using hdfs dfs -put C:\path\to\file.txt /user/abinaya/.
5.List files in HDFS to confirm upload using hdfs dfs -ls /user/abinaya.
6.View the file content in HDFS using hdfs dfs -cat /user/abinaya/file.txt.
7.Download the file from HDFS to local using hdfs dfs -get /user/abinaya/file.txt C:\path\to\download\.
8.Delete a file in HDFS using hdfs dfs -rm /user/abinaya/file.txt.
9.Delete a directory in HDFS using hdfs dfs -rm -r /user/abinaya.



```
C:\hadoop\sbin>
C:\hadoop\sbin>hdfs dfs -ls /user/hadoopfiles
Found 1 items
-rw-r--r--   1 abhin supergroup         24 2025-05-05 20:57 /user/hadoopfiles/ex2hadoopfiles.txt

C:\hadoop\sbin>hdfs dfs -get /user/abinaya/input.txt C:\output.txt
get: `/user/abinaya/input.txt': No such file or directory

C:\hadoop\sbin>hdfs dfs -get /user/hadoopfiles/ex2hadoopfiles.txt C:\output.txt

C:\hadoop\sbin>hdfs dfs -cat /user/hadoopfiles/ex2hadoopfiles.txt
hello
Hadoop
files

C:\hadoop\sbin>hdfs dfs -rm /user/hadoopfiles/ex2hadoopfiles.txt
Deleted /user/hadoopfiles/ex2hadoopfiles.txt

C:\hadoop\sbin>
C:\hadoop\sbin>
C:\hadoop\sbin>hdfs dfs -rm /user/hadoopfiles
rm: `/user/hadoopfiles': Is a directory
```



Result:

Thus,to implement basic file management tasks using Hadoop HDFS including creating directories, adding files, retrieving files, and deleting files is done successfully.

Ex:03 Implementation of Matrix Multiplication with Hadoop Map Reduce
Date:

Aim:

To implement matrix multiplication using the Hadoop MapReduce programming.

Procedure:

1. Install and configure Hadoop on your Windows machine.
2. Create two input matrices and save them as text files.
3. Write the MapReduce Java program for matrix multiplication.
4. Compile the Java code and create a JAR file (e.g., matrixmul.jar).
5. Start Hadoop services and create required HDFS directories.
6. Upload the matrix input files to the HDFS input folder.
7. Run the MapReduce job using the Hadoop jar command.
8. Download the output file from HDFS and view the result locally.

Program:

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;
public class MatrixMultiplication {
   public static class MapperClass extends Mapper<LongWritable, Text, Text, Text> {
      public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
         // Split line into components
         String[] parts = value.toString().split("\\s+");
         String matrixName = parts[0];
         int row = Integer.parseInt(parts[1]);
         int col = Integer.parseInt(parts[2]);
         int val = Integer.parseInt(parts[3]);

         Configuration conf = context.getConfiguration();
         int m = Integer.parseInt(conf.get("m"));
         int n = Integer.parseInt(conf.get("n"));
         int p = Integer.parseInt(conf.get("p"));

         if (matrixName.equals("A")) {
            for (int k = 0; k < p; k++) {
               context.write(new Text(row + "," + k), new Text("A," + col + "," + val));
```

```
            }
        } else {
            for (int i = 0; i < m; i++) {
                context.write(new Text(i + "," + col), new Text("B," + row + "," + val));
            }
        }
    }
}
public static class ReducerClass extends Reducer<Text, Text, Text, IntWritable> {
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
        Map<Integer, Integer> aMap = new HashMap<>();
        Map<Integer, Integer> bMap = new HashMap<>();

        for (Text val : values) {
            String[] parts = val.toString().split(",");
            String matrixName = parts[0];
            int index = Integer.parseInt(parts[1]);
            int value = Integer.parseInt(parts[2]);
            if (matrixName.equals("A")) {
                aMap.put(index, value);
            } else {
                bMap.put(index, value);
            }
        }
        int sum = 0;
        for (int k : aMap.keySet()) {
            if (bMap.containsKey(k)) {
                sum += aMap.get(k) * bMap.get(k);
            }
        }

        context.write(key, new IntWritable(sum));
    }
}
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    // Set matrix dimensions
    conf.set("m", "2");
    conf.set("n", "2");
    conf.set("p", "2");
    Job job = Job.getInstance(conf, "Matrix Multiplication");
    job.setJarByClass(MatrixMultiplication.class);
    job.setMapperClass(MapperClass.class);
```

```
        job.setReducerClass(ReducerClass.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path("input"));
        FileOutputFormat.setOutputPath(job, new Path("output"));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Output:

Commands

```
javac -classpath "$(hadoop classpath)" -d . MatrixMultiplication.java
jar -cvf matrixmul.jar *
hadoop fs -mkdir /user/abhin/input
hadoop fs -put input.txt /user/abhin/input
hadoop jar matrixmul.jar MatrixMultiplication /user/abhin/input /user/abhin/output
hadoop fs -get /user/abhin/output C:/mat_mul/output
```

input.txt

```
A 0 0 1
A 0 1 2
A 1 0 3
A 1 1 4
B 0 0 5
B 0 1 6
B 1 0 7
B 1 1 8
```

Output.txt

```
0,0     19
0,1     22
1,0     43
1,1     50
```

Result:

Thus, to implement the matrix multiplication using Hadoop Map Reduce is done successfully.

Ex:04 Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm
Date:
Aim:

To implement the basic word count program using Hadoop Map Reduce.

Procedure:

1. Create WordCount Java files: Write WordCountMapper.java, WordCountReducer.java, and WordCountDriver.java.
2. Compile Java files: Use javac -cp ".;lib/*" -d wordcount_classes *.java to compile all files.
3. Create JAR file: Package classes using jar -cvf wordcount.jar -C wordcount_classes/ ..
4. Start HDFS: Start Hadoop services using start-dfs.cmd and start-yarn.cmd (on Windows).
5. Create input directory in HDFS: Run hdfs dfs -mkdir /wordcount and upload file with hdfs dfs -put input.txt /wordcount.
6. Run the MapReduce job: Execute hadoop jar wordcount.jar WordCountDriver /wordcount /wordcount_output.
7. Check output in HDFS: Use hdfs dfs -ls /wordcount_output to verify the result files exist.
8. Display results: Run hdfs dfs -cat /wordcount_output/part-r-00000 to view the word count output.

Program:

WordCountDriver.java

```
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 WordCountDriver {
   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "word count");
      job.setJarByClass(WordCountDriver.class);
      job.setMapperClass(WordCountMapper.class);
      job.setReducerClass(WordCountReducer.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}
```

WordCountMapper.java

```
import java.io.IOException;
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 WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {
      String[] words = value.toString().split("\\s+");
      for (String w : words) {
         word.set(w.toLowerCase().replaceAll("[^a-zA-Z0-9]", ""));
         context.write(word, one);
      }
   }
}
```

WordCountReducer.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
   public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
         sum += val.get();
      }
      context.write(key, new IntWritable(sum));
   }
}
```

Output:

Commands:

```
javac -cp ".;lib/*" -d wordcount_classes *.java
jar -cvf wordcount.jar -C wordcount_classes/ .
hdfs dfs -mkdir /wordcount
hdfs dfs -put input.txt /wordcount
hadoop jar wordcount.jar WordCountDriver /wordcount /wordcount_output
hdfs dfs -cat /wordcount_output/part-r-00000
```

Input.txt

Hadoop is an open source framework that allows for the distributed processing of large data sets

Output.txt

allows 1
an 1
data 1
distributed 1
for 1
framework 1
hadoop 1
is 1
large 1
of 1
open 1
processing 1
sets 1
source 1
that 1
the 1

Result:

Thus, to implement the basic word count using hadoop mapreduce is done successfully.

Ex:05 Installation of Hive along with practice examples
Date:

Aim:

To install hive and do practice examples.

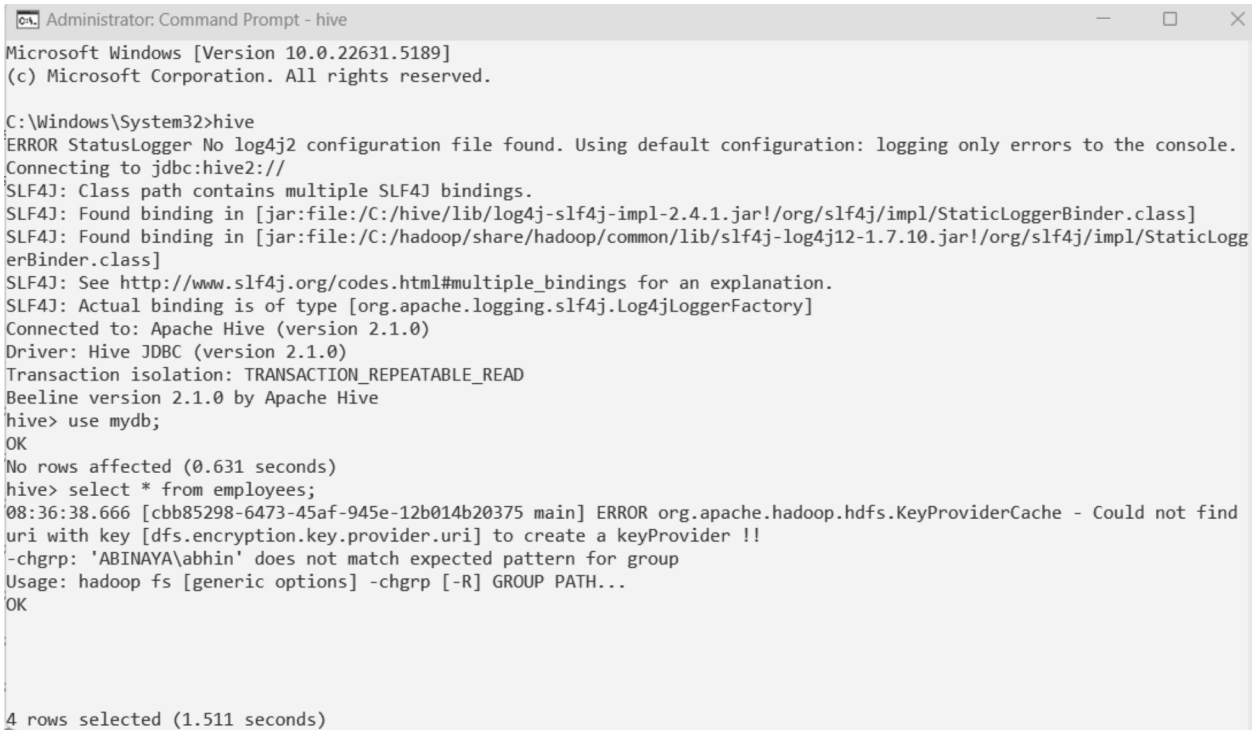
Procedure:

1.Download hadoop 2.7.0,hive 2.1.0 and derby 10.14.2.0.
2.Extract all and copy all files in lib folder of derby and paste it in the lib folder of hive.
3.Download hive-site.xml and paste in conf folder.
4.Set the environmental variable for both derby and hive.
5.Start the hadoop.
6.Start the derby server as

```
Microsoft Windows [Version 10.0.22631.5189]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>start-dfs.cmd

C:\Windows\System32>start-yarn.cmd
starting yarn daemons

C:\Windows\System32>StartNetworkServer -h 0.0.0.0
Wed May 07 08:17:33 IST 2025 : Security manager installed using the Basic server security policy.
Wed May 07 08:17:34 IST 2025 : Apache Derby Network Server - 10.14.2.0 - (1828579) started and ready to acc
ns on port 1527
```

7.Type hive in another command promt and you will see hive>
Now start your db query.

```
Administrator: Command Prompt - hive
Microsoft Windows [Version 10.0.22631.5189]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>hive
ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the
Connecting to jdbc:hive2://
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.c
SLF4J: Found binding in [jar:file:/C:/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connected to: Apache Hive (version 2.1.0)
Driver: Hive JDBC (version 2.1.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.1.0 by Apache Hive
hive> create database mydb;
OK
No rows affected (0.816 seconds)
hive> use mydb;
OK
No rows affected (0.141 seconds)
hive> create table employees(id int,name string,age int,city string);
OK
No rows affected (0.365 seconds)
hive> load data local inpath 'C:/hive/employees.txt' into table employees;
Loading data to table mydb.employees
OK
No rows affected (1.177 seconds)
hive> select * from employees;
08:21:07.129 [4c135339-705f-4302-8076-74a217674519 main] ERROR org.apache.hadoop.hdfs.KeyProviderCache - Could
-chgrp: 'ABINAYA\abhin' does not match expected pattern for group
Usage: hadoop fs [generic options] -chgrp [-R] GROUP PATH...
OK


2 rows selected (1.255 seconds)
hive> SELECT * FROM employees WHERE age > 30;
-chgrp: 'ABINAYA\abhin' does not match expected pattern for group
Usage: hadoop fs [generic options] -chgrp [-R] GROUP PATH...
OK
No rows selected (0.508 seconds)
```



Result:

Thus, to install and practice the example in hive is done successfully.

Ex:06 Installation of HBase ,installing thrift along with practice examples
Date:

Aim:

To install hbase and thrift and do the practice problem.

Procedure:

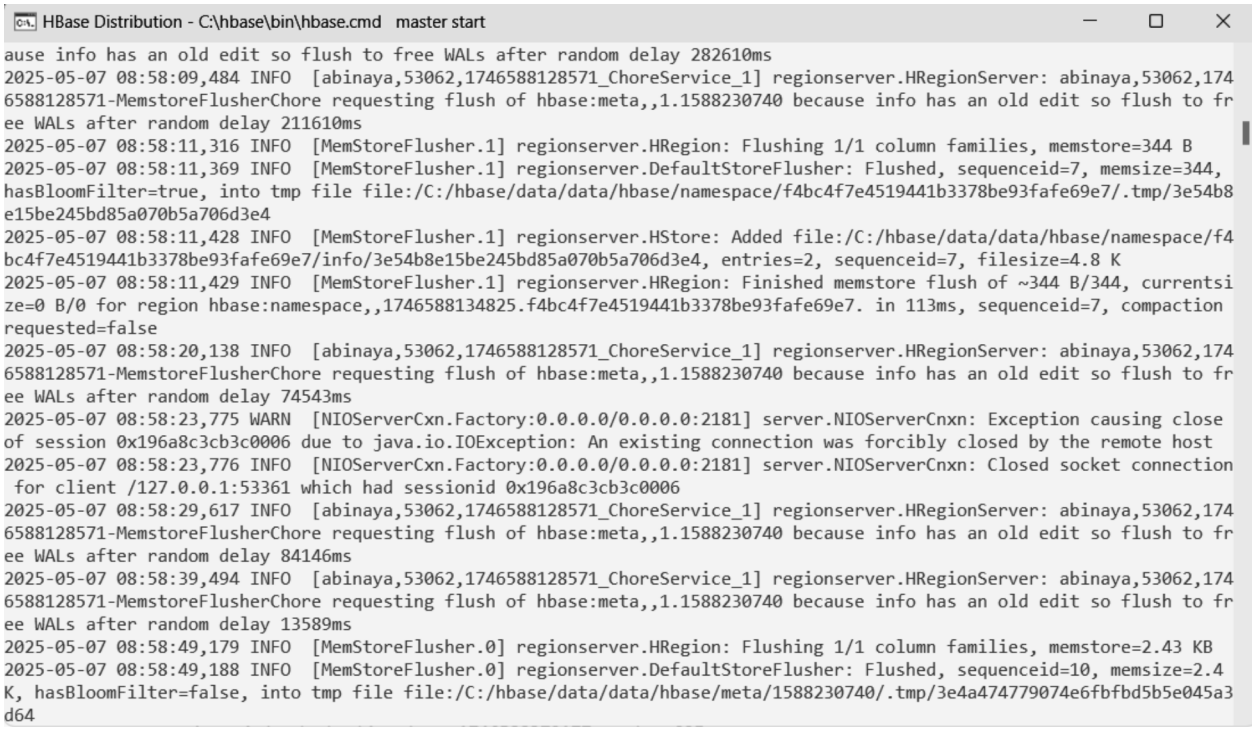
1. Install Java and set JAVA_HOME in system environment variables.
2. Extract HBase to C:\hbase and edit conf\hbase-site.xml to set root and zookeeper directories.
3. Add HBase /bin to PATH in environment variables.
4. Start Hadoop using start-all.cmd (if not already running).
5. Start HBase using start-hbase.cmd from C:\hbase\bin.
6. Open HBase shell by typing hbase shell in Command Prompt.
7. Create a table using create 'student', 'info'.
8. Insert data using put 'student', '1', 'info:name', 'Abinaya' and another put for dept.
9. Retrieve data using get 'student', '1'.
10. Scan the table using scan 'student'.
11. Copy all JARs from C:\hbase\lib to C:\hive\lib.
12. Start Hive CLI using hive in command prompt.
13. Create Hive external table mapped to HBase using provided CREATE EXTERNAL TABLE query.
14. Query the Hive table using SELECT * FROM hbase_student;.
15. (Optional) Exit HBase shell using exit.
16. (Optional) Disable and drop table using disable 'student' and drop 'student'.

Output: Configure hbse-site.xml as follow

```
<configuration>

  <property>

    <name>hbase.rootdir</name>

    <value>file:///C:/hbase/data</value>

  </property>

  <property>

    <name>hbase.zookeeper.property.dataDir</name>

    <value>C:/hbase/zookeeper</value>

  </property>

  <property>
```

```
  <name>hbase.cluster.distributed</name>
  <value>false</value>
 </property>
</configuration>
```

```
C:\Windows\System32>cd C:\hbase\bin

C:\hbase\bin>start-hbase.cmd

C:\hbase\bin>hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/hbase/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0> create 'student', 'info'
0 row(s) in 1.4480 seconds

=> Hbase::Table - student
hbase(main):002:0> put 'student', '1', 'info:name', 'Abinaya'
0 row(s) in 0.1110 seconds

hbase(main):003:0> put 'student', '1', 'info:dept', 'CSE'
0 row(s) in 0.0070 seconds

hbase(main):004:0> get 'student', '1'
COLUMN                  CELL
 info:dept              timestamp=1746588379177, value=CSE
 info:name              timestamp=1746588370587, value=Abinaya
2 row(s) in 0.0260 seconds

hbase(main):005:0> scan 'student'
ROW                     COLUMN+CELL
 1                      column=info:dept, timestamp=1746588379177, value=CSE
 1                      column=info:name, timestamp=1746588370587, value=Abinaya
1 row(s) in 0.0170 seconds

hbase(main):006:0>
hbase(main):007:0* stop
NameError: undefined local variable or method `stop' for #<Object:0xc11332b>

hbase(main):008:0> exit

C:\hbase\bin>hbase thrift start
2025-05-07 08:58:42,695 INFO  [main] util.VersionInfo: HBase 1.2.6
2025-05-07 08:58:42,696 INFO  [main] util.VersionInfo: Source code repository file:///home/busbey/projects/hbase/hbase-assembly/target/hbase-1.2.6 revision=Unknown
```



Result:

Thus, to install and practice example with hbase and thrift is done successfully.