

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING (AUTONOMOUS)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

20CS62 - DATA ANALYTICS AND VISUALISATION LAB

Name of the Student:

Registered Number :

Branch and Section :

Academic Year : :

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING (AUTONOMOUS)



This is to certify that this is a bonafide record of the practical work done by Mr./Ms..... bearing Reg.No: Of **B. Tech** Semester Branch Section in the 20CS62 – **DATA ANALYTICS AND visualization LAB** during the Academic Year: **2024-25**.

No. of Experiments/Modules held: 11

No. of Experiments Done: 11

Date:/...../25.

INTERNAL EXAMINER

EXTERNAL EXAMINER

INDEX

S.no	Experiments	Page no.	Date	sign
1	Refreshing Linux Commands and Installation of Hadoop.			
2	Implementation of run a basic word count map reduce program.			
3	Implementation of Matrix multiplication with Hadoop map reduce.			
4	Implementation of Weather mining by taking weather data set using map reduce.			
5	Installation of Hive along with practice examples.			
6	Installation of Sqoop along with practice examples.			
7	Downloading and installing Tableau Understanding about importing data , saving , opening and sharing work books.			
8	Data preparation with Tableau.			
9	Charts: Bar charts , Legends , Filters , and Hierarchies , step charts , Line charts.			
10	Maps: Symbol maps , Filled Maps ,Density maps ,Maps with Pie charts.			
11	Interactive Dashboards.			

Vision of the Department

The Computer Science & Engineering aims at providing continuously stimulating educational environment to its students for attaining their professional goals and meet the global challenges.

Mission of the Department

- **DM1:** To develop a strong theoretical and practical background across the computer science discipline with an emphasis on problem solving.
- **DM2:** To inculcate professional behaviour with strong ethical values, leadership qualities, innovative thinking and analytical abilities into the student.
- **DM3:** Expose the students to cutting edge technologies which enhance their employability and knowledge.
- **DM4:** Facilitate the faculty to keep track of latest developments in their research areas and encourage the faculty to foster the healthy interaction with industry.

Program Educational Objectives (PEOs)

- **PEO1:** Pursue higher education, entrepreneurship and research to compete at global level.
- **PEO2:** Design and develop products innovatively in computer science and engineering and in other allied fields.
- **PEO3:** Function effectively as individuals and as members of a team in the conduct of interdisciplinary projects; and even at all the levels with ethics and necessary attitude.
- **PEO4:** Serve ever-changing needs of society with a pragmatic perception.

PROGRAMME OUTCOMES (POs):

PO 1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities

	and norms of the engineering practice.
PO 9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAMME SPECIFIC OUTCOMES (PSOs):

PSO 1	The ability to apply Software Engineering practices and strategies in software project development using open-source programming environment for the success of organization.
PSO 2	The ability to design and develop computer programs in networking, web applications and IoT as per the society needs.
PSO 3	To inculcate an ability to analyze, design and implement database applications.

Experiment 1:

AIM: Refreshing Linux Commands and Installation of Hadoop.

Hadoop Basic Commands :

1. Print the Hadoop version

#

hadoop version

2. List the contents of the root directory in HDFS

hadoop fs -ls /

3. Report the amount of space used and

available on currently mounted filesystem

#

hadoop fs -df hdfs:/

4. Count the number of directories,files and bytes under

the paths that match the specified file pattern

#

hadoop fs -count hdfs:/

5. Run a DFS filesystem checking utility

#

hadoop fsck - /

6. Run a cluster balancing utility

#

hadoop balancer

7. Create a new directory named “hadoop” below the

/user/training directory in HDFS. Since you’re

currently logged in with the “training” user ID,

/user/training is your home directory in HDFS.

#

hadoop fs -mkdir /user/training/hadoop

8. Add a sample text file from the local directory

named “data” to the new directory you created in HDFS

during the previous step.

#

hadoop fs -put data/sample.txt /user/training/hadoop

9. List the contents of this new directory in HDFS.

```

#
hadoop fs -ls /user/training/hadoop
# 10. Add the entire local directory called “retail” to the
# /user/training directory in HDFS.
#
hadoop fs -put data/retail /user/training/hadoop
# 11. Since /user/training is your home directory in HDFS,
# any command that does not have an absolute path is
# interpreted as relative to that directory. The next
# command will therefore list your home directory, and
# should show the items you’ve just added there.
#
hadoop fs -ls
# 12. See how much space this directory occupies in HDFS.
#
hadoop fs -du -s -h hadoop/retail
# 13. Delete a file ‘customers’ from the “retail” directory.
#
hadoop fs -rm hadoop/retail/customers
# 14. Ensure this file is no longer in HDFS.
#
hadoop fs -ls hadoop/retail/customers
# 15. Delete all files from the “retail” directory using a wildcard.
#
hadoop fs -rm hadoop/retail/*
# 16. To empty the trash
#
hadoop fs -expunge
# 17. Finally, remove the entire retail directory and all
# of its contents in HDFS.
#
hadoop fs -rm -r hadoop/retail
# 18. List the hadoop directory again
#
hadoop fs -ls hadoop
# 19. Add the purchases.txt file from the local directory
# named “/home/training/” to the hadoop directory you created in HDFS
#

```

```

hadoop fs -copyFromLocal /home/training/purchases.txt hadoop/
# 20. To view the contents of your text file purchases.txt
# which is present in your hadoop directory.
#
hadoop fs -cat hadoop/purchases.txt
# 21. Add the purchases.txt file from “hadoop” directory which is present in HDFS directory
# to the directory “data” which is present in your local directory
#
hadoop fs -copyToLocal hadoop/purchases.txt /home/training/data
# 22. cp is used to copy files between directories present in HDFS
#
hadoop fs -cp /user/training/*.txt /user/training/hadoop
# 23. ‘-get’ command can be used alternatively to ‘-copyToLocal’ command
#
hadoop fs -get hadoop/sample.txt /home/training/
# 24. Display last kilobyte of the file “purchases.txt” to stdout.
#
hadoop fs -tail hadoop/purchases.txt
# 25. Default file permissions are 666 in HDFS
# Use ‘-chmod’ command to change permissions of a file
#
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chmod 600 hadoop/purchases.txt
# 26. Default names of owner and group are training,training
# Use ‘-chown’ to change owner name and group name simultaneously
#
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chown root:root hadoop/purchases.txt
# 27. Default name of group is training
# Use ‘-chgrp’ command to change group name
#
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chgrp training hadoop/purchases.txt
# 28. Move a directory from one location to other
#
hadoop fs -mv hadoop apache_hadoop

```



```
# 29. Default replication factor to a file is 3.
# Use '-setrep' command to change replication factor of a file
#
hadoop fs -setrep -w 2 apache_hadoop/sample.txt
# 30. Copy a directory from one node in the cluster to another
# Use '-distcp' command to copy,
# -overwrite option to overwrite in an existing files
# -update command to synchronize both directories
#
hadoop fs -distcp hdfs://namenodeA/apache_hadoop hdfs://namenodeB/hadoop
# 31. Command to make the name node leave safe mode
#
hadoop fs -expunge
sudo -u hdfs hdfs dfsadmin -safemode leave
# 32. List all the hadoop file system shell commands
#
hadoop fs
# 33. Last but not least, always ask for help!
#
hadoop fs -help
```

Installation Of Hadoop:

Open terminal in UBUNTU and execute the fallowing commands one by one

- 1)sudo apt update
- 2)sudo apt install openjdk-8-jdk -y
- 3)java -version; javac -version
- 4)sudo apt install openssh-server openssh-client -y
- 5)ssh-keygen -t rsa -P " -f ~/.ssh/id_rsa

6)cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

7)chmod 0600 ~/.ssh/authorized_keys

8)ssh localhost

9)wget https://downloads.apache.org/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz

10)tar xzf hadoop-3.2.1.tar.gz

11)sudo nano .bashrc

#Hadoop Related Options

export HADOOP_HOME=/home/hadoop-3.2.1

export HADOOP_INSTALL=\$HADOOP_HOME

export HADOOP_MAPRED_HOME=\$HADOOP_HOME

export HADOOP_COMMON_HOME=\$HADOOP_HOME

export HADOOP_HDFS_HOME=\$HADOOP_HOME

export YARN_HOME=\$HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE_DIR=\$HADOOP_HOME/lib/native

export PATH=\$PATH:\$HADOOP_HOME/sbin:\$HADOOP_HOME/bin

export HADOOP_OPTS="-Djava.library.path=\$HADOOP_HOME/lib/native"

12)source ~/.bashrc

13)sudo nano \$HADOOP_HOME/etc/hadoop/hadoop-env.sh

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

which javac

readlink -f /usr/bin/javac

14)sudo nano \$HADOOP_HOME/etc/hadoop/core-site.xml

<configuration>

<property>

```
<name>hadoop.tmp.dir</name>
<value>/home/tmpdata</value>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>
```

15) sudo nano \$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<configuration>
<property>
  <name>dfs.data.dir</name>
  <value>/home/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
</configuration>
```

16) sudo nano \$HADOOP_HOME/etc/hadoop/mapred-site.xml

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

17) sudo nano \$HADOOP_HOME/etc/hadoop/yarn-site.xml

```
<configuration>
```

```

<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>127.0.0.1</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
</property>
</configuration>

```

18) hdfs namenode -format

Navigate to the hadoop-3.2.1/sbin directory

19) ./start-dfs.sh

20) ./start-yarn.sh

21) jps

22) http://localhost:9870

23) http://localhost:9864

24) <http://localhost:8080>

EXPERIMENT #2

AIM: Implementation of run a basic word count map reduce program.

WCDRIVER.java

```
// importing libraries
```

```
import java.io.IOException;
```

```
import org.apache.hadoop.conf.Configured;
```

```
import org.apache.hadoop.fs.Path;
```

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

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

```
import org.apache.hadoop.mapred.FileInputFormat;
```

```
import org.apache.hadoop.mapred.FileOutputFormat;
```

```
import org.apache.hadoop.mapred.JobClient;
```

```
import org.apache.hadoop.mapred.JobConf;
```

```
import org.apache.hadoop.util.Tool;
```

```
import org.apache.hadoop.util.ToolRunner;
```

```
public class WCDriver extends Configured implements Tool {
```

```
    public int run(String args[]) throws IOException
```

```
    {
```

```
        if (args.length < 2)
```

```
        {
```

```
            System.out.println("Please give valid inputs");
```

```
            return -1;
```

```
        }
```

```
        JobConf conf = new JobConf(WCDriver.class);
```

```
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
```

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

```
        conf.setMapperClass(WCMapper.class);
```

```
        conf.setReducerClass(WCReducer.class);
```

```
        conf.setMapOutputKeyClass(Text.class);
```

```
        conf.setMapOutputValueClass(IntWritable.class);
```

```
        conf.setOutputKeyClass(Text.class);
```

```

        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }
    // Main Method
    public static void main(String args[]) throws Exception
    {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println(exitCode);
    }
}

```

WCMAPPER.java

```

// Importing libraries
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.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,Text,
Text, IntWritable> {

    // Map function
    public void map(LongWritable key, Text value,OutputCollector
<Text,IntWritable> output, Reporter rep) throws IOException
    {
        String line = value.toString();
        // Splitting the line on spaces
        for (String word : line.split(" "))
        {
            if (word.length() > 0)
            {
                output.collect(new Text(word), new IntWritable(1));
            }
        }
    }
}

```

```

    }
}
}

```

WCREDUCER.java

```

// Importing libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer
<Text,IntWritable, Text, IntWritable> {
    // Reduce function

public void reduce(Text key, Iterator<IntWritable> value,
    OutputCollector<Text, IntWritable> output,Reporter rep) throws IOException
    {
        int count = 0;
        // Counting the frequency of each words
        while (value.hasNext())
        {
            IntWritable i = value.next();
            count += i.get();
        }
        output.collect(key, new IntWritable(count));
    }
}

```

Word Count process: Cloudera tool

Counting the number of words in any language is a piece of cake like in C, C++, Python, Java, etc. MapReduce also uses Java but it is very easy if you know the syntax on how to write it. It is the basic of MapReduce. You will first learn how to execute this code similar to “Hello World” program in other languages. So here are the steps which show how to write a MapReduce code for Word Count.

Example:

Input:

Hello I am GeeksforGeeks

Hello I am an Intern

Output:

GeeksforGeeks 1

Hello 2

I 2

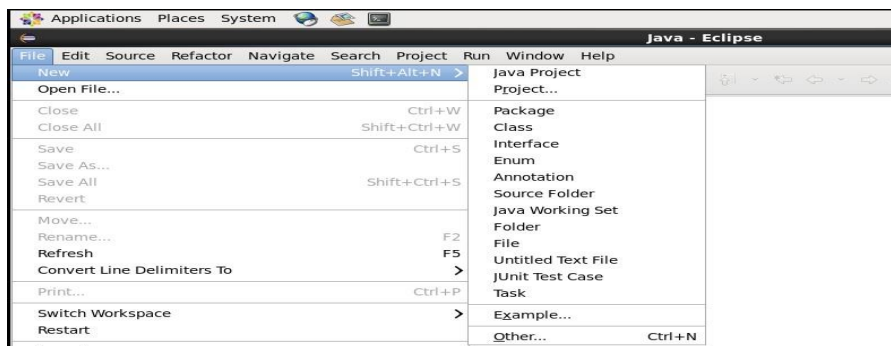
Intern 1

am 2

an 1

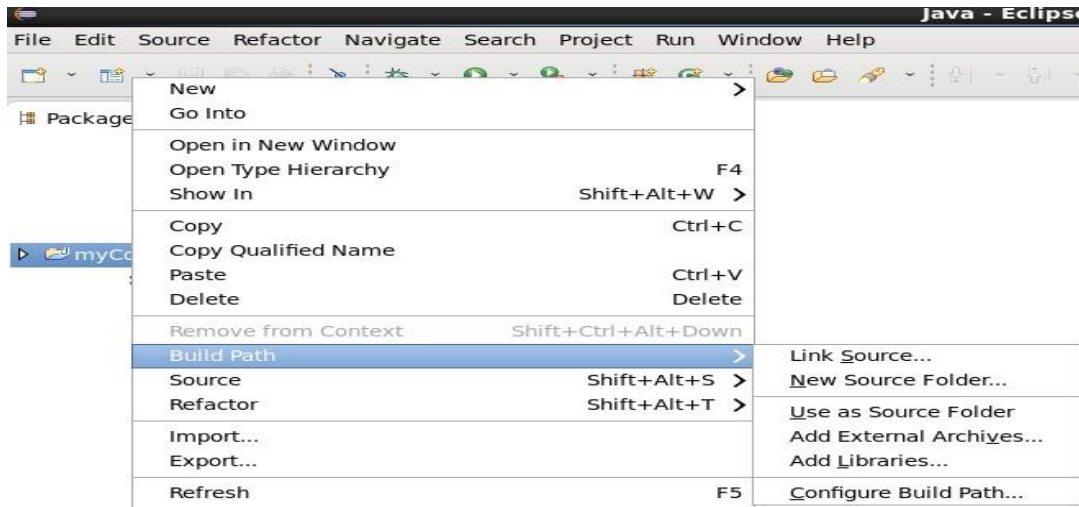
Steps:

- First Open **Eclipse** -> then select **File** -> **New** -> **Java Project** -> Name it **WordCount** -> then **Finish**.

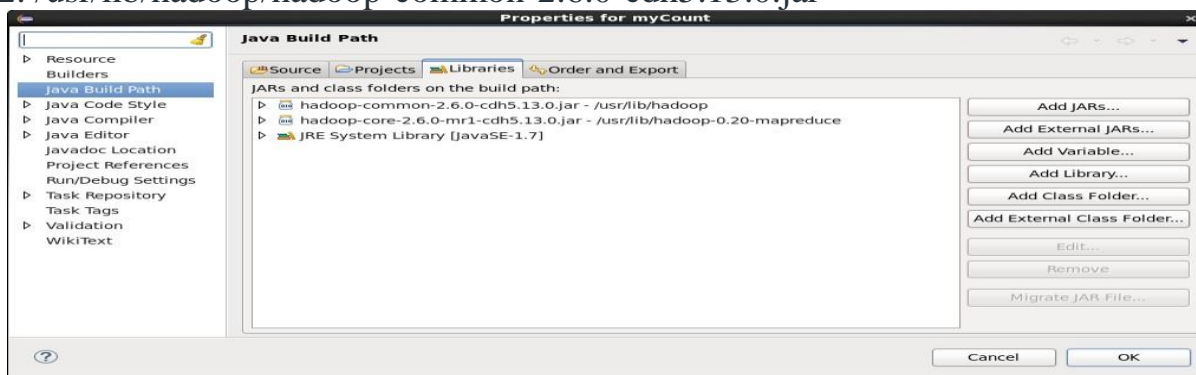


- Create Three Java Classes into the project. Name them **WCDriver**(having the main function), **WCMapper**, **WCReducer**.

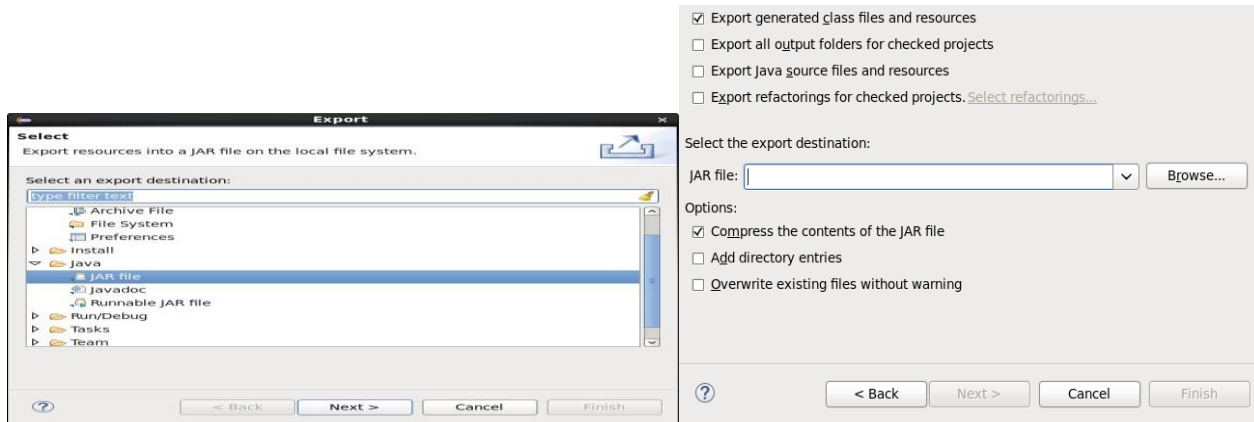
- You have to include two Reference Libraries for that:
Right Click on **Project** -> then select **Build Path**-> Click on **Configure Build Path**



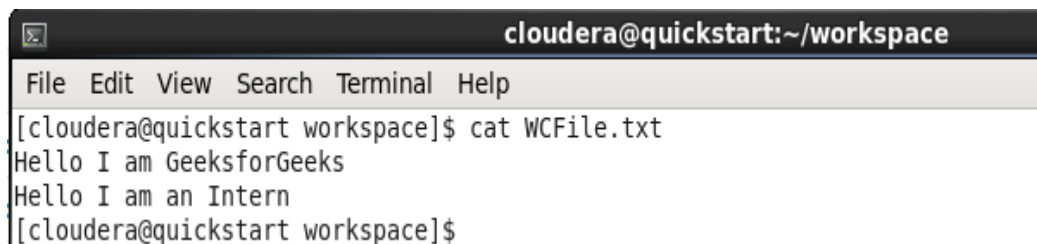
- In the above figure, you can see the Add External JARs option on the Right Hand Side. Click on it and add the below mention files. You can find these files in `/usr/lib/`
 - `/usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar`
 - `/usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar`



- Mapper Code:** You have to copy paste this program into the WCMapper Java Class file.
 - Reducer Code:** You have to copy paste this program into the WCReducer Java Class file.
 - Driver Code:** You have to copy paste this program into the WCDriver Java Class file.
 - After complete**
- Now you have to make a jar file. Right Click on **Project**-> **Click on Export**-> **Select export destination as Jar File**-> **Name the jar File**(WordCount.jar) -> **Click on next** -> at last **Click on Finish**. Now copy this file into the Workspace directory of Cloudera



- Open the terminal on CDH and change the directory to the workspace. You can do this by using “cd workspace/” command. Now, Create a text file(**WCFile.txt**) and move it to HDFS. For that open terminal and write this code(remember you should be in the same directory as jar file you have created just now).



AFTER CREATE A JAR FILE WORD COUNT EXECUTION PROCESS

STEP 1: After Creation of Jar file

--open command prompt in cloudera..

---create/set specified path---ex: you have save a files on desktop---use

----cd Desktop

STEP 2:

--Create Input file and store data

Ex: specified input data using for experiment .

STEP 3:

-----create a directory/folder in a Hadoop(store all data)

----hadoop fs -mkdir /foldername

---ex: Hadoop fs -mkdir /Nagababu

Note: check the folder create or not

----hadoop fs -ls

STEP 4:

-----copy the input file to Hadoop

----hadoop fs -put inputfilename /destination folder

---ex: Hadoop fs -put input.txt /Nagababu

Note: check the input file copy or not in specified directory

----hadoop fs -ls Nagababu

STEP 5:

----Execute file and store output in particular directory or same directory...using command

----Hadoop jar filename.jar /folder name/input filename /folder name/output storing foldername

---ex: hadoop jar wordcount.jar /Nagababu/wordinput.txt /Nagababu/wordcountoutput

(Or)

NOTE: If not attached a Driverclass program in jar file using command..

```
-----hadoop jar /home/cloudera/Desktop/Nagababu/Filename.jar com.lbrce.Drivername  
Nagababu/filename.txt /Nagababu/output
```

Process executed.....

STEP 6:

Check the output

```
hadoop fs -ls /folder name/output
```

```
----ex: hadoop fs -ls /Nagababu/wordcountoutput
```

Note : the output folder is after execution stored a two file 1.Success file

2.part file(stored out put)

STEP 7:

final out put enter command

```
-----hadoop fs -cat /foldername/outputfolder/part*
```

```
-----Ex: hadoop fs -cat /Nagababu/wordcountoutput/part*
```

Output:

Experiment #03

AIM: Implementation of Matrix Multiplication with Hadoop Map Reduce.

MATRIXDRIVER.java

```
import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

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

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

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

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


public class MatrixDriver

{

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

    {

        Configuration conf = new Configuration();

        // M is an m-by-n matrix; N is an n-by-p matrix.

        conf.set("m", "2");
```

```
    conf.set("n", "2");
    conf.set("p", "2");

    Job job = Job.getInstance(conf, "MatrixMultiplication");

    job.setJarByClass(MatrixDriver.class);

    job.setOutputKeyClass(Text.class);

    job.setOutputValueClass(Text.class);

    job.setMapperClass(MatrixMapper.class);

    job.setReducerClass(MatrixReducer.class);

    job.setInputFormatClass(TextInputFormat.class);

    job.setOutputFormatClass(TextOutputFormat.class);

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

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

    job.submit();
}
}
```

MATRIXMAPPER.java

```
import java.io.IOException;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

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

```

public class MatrixMapper extends Mapper<LongWritable, Text, Text, Text>
{

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

    {

        Configuration conf = context.getConfiguration();

        int m = Integer.parseInt(conf.get("m"));

        int p = Integer.parseInt(conf.get("p"));

        String line = value.toString();

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

        Text outputKey = new Text();

        Text outputValue = new Text();

        if (indicesAndValue[0].equals("M"))

        {

            for (int k = 0; k < p; k++)

            {

                outputKey.set(indicesAndValue[1] + "," + k);

                outputValue.set("M," + indicesAndValue[2] + "," + indicesAndValue[3]);

                context.write(outputKey, outputValue);
            }
        }
        else

```

```

for (int i = 0; i < m; i++)
{
    outputKey.set(i + "," + indicesAndValue[2]);
    outputValue.set("N," + indicesAndValue[1] + "," + indicesAndValue[3]);
    context.write(outputKey, outputValue);

}
}
}
}

```

MATRIXREDUCER.java

```

import java.io.IOException;

import java.util.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

public class MatrixReducer extends Reducer<Text, Text, Text, Text>

{

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

    {

        String[] value;

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

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

        for (Text val : values)

```



```

value = val.toString().split(",");

if (value[0].equals("M"))

{

hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));

}

else

{

hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));

}

}

int n = Integer.parseInt(context.getConfiguration().get("n")); float result = 0.0f;

float a_ij; float b_jk;

for (int j = 0; j < n; j++)

{

a_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;

b_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;

result += a_ij * b_jk;

}

if (result != 0.0f)

```

```

{

context.write(null, new Text(key.toString() + "," + Float.toString(result)));

}

}

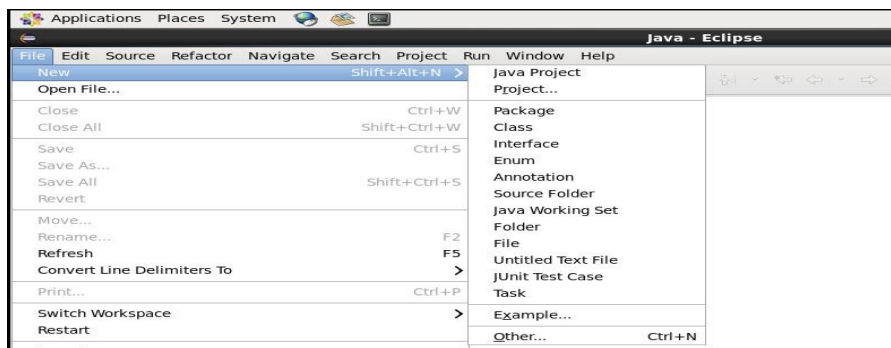
}

```

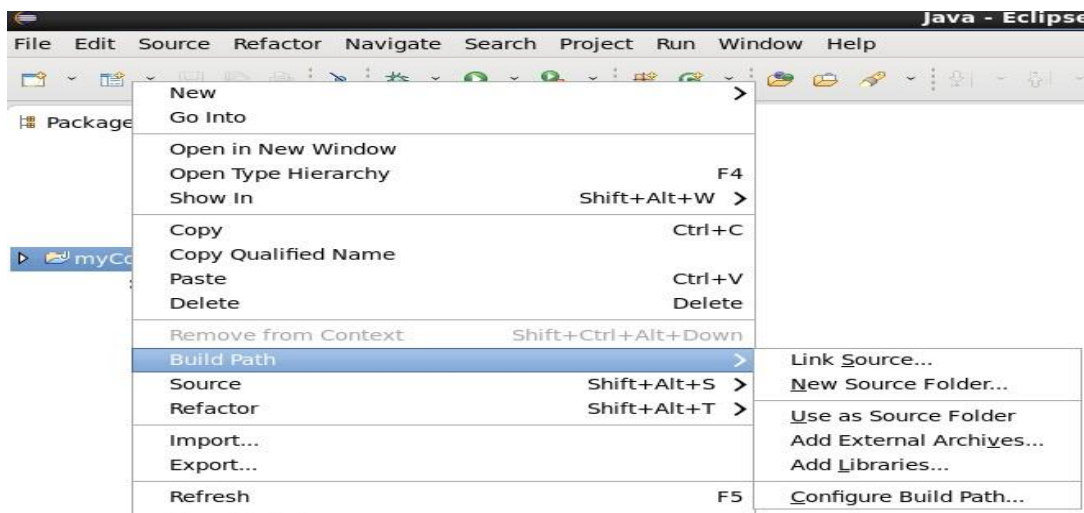
MATRIX MULTIPLICATION PROCEDURE on Cloudera tool

Steps:

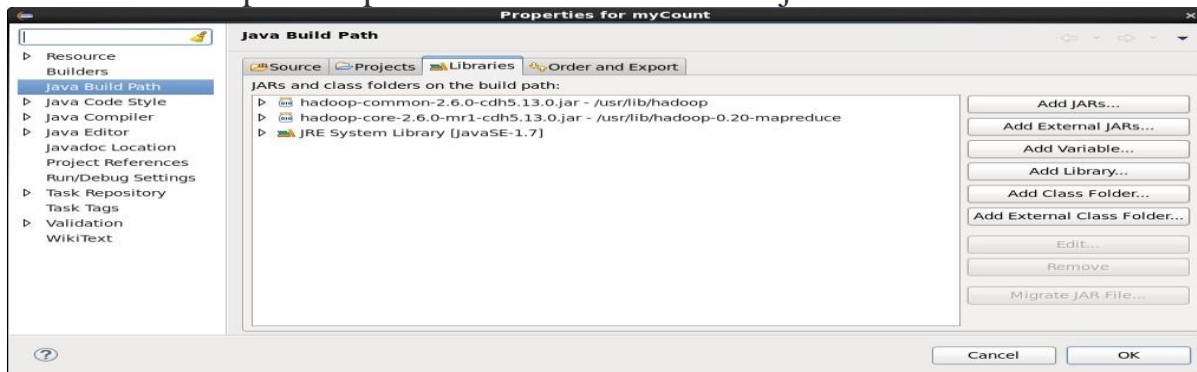
- First Open **Eclipse** -> then select **File** -> **New** -> **Java Project** -> Name it **MatrixMul** -> then **Finish**.



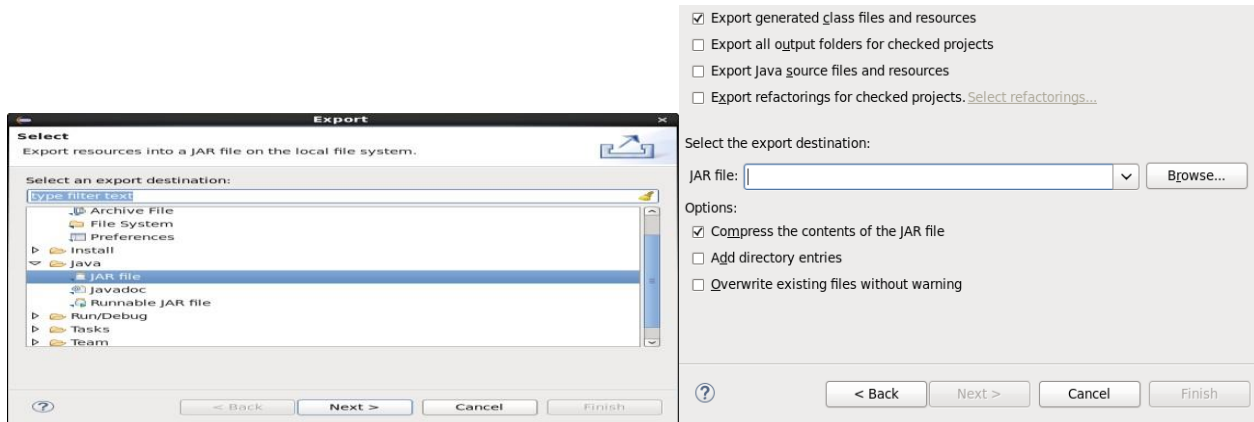
- Create Three Java Classes into the project. Name them **MatrixMulDriver**(having the main function), **MatrixMulMapper**, **MatrixMulReducer**.
- You have to include two Reference Libraries for that:
Right Click on **Project** -> then select **Build Path**-> Click on **Configure Build Path**



- In the above figure, you can see the Add External JARs option on the Right Hand Side. Click on it and add the below mention files. You can find these files in `/usr/lib/`
 1. `/usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar`
 2. `/usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar`



- - **Mapper Code:** You have to copy paste this program into the MatrixMulMapper Java Class file.
 - **Reducer Code:** You have to copy paste this program into the MatrixMulReducer Java Class file.
 - **Driver Code:** You have to copy paste this program into the MatrixMulDriver Java Class file.
 - **After complete**
- Now you have to make a jar file. Right Click on **Project**-> **Click on Export**-> **Select export destination as Jar File**-> **Name the jar File(MatrixMul.jar)** -> **Click on next** -> at last **Click on Finish**.



- Open the terminal on CDH and change the directory to the workspace. You can do this by using “cd workspace/” command. Now, Create a text file(**matinput.txt**) and move it to HDFS. For that open terminal and write this code(remember you should be in the same directory as jar file you have created just now).

```
matinput.txt M,0,0,1
              M,0,1,2
              M,1,0,3
              M,1,1,4
              N,0,0,1
              N,0,1,2
              N,1,0,3
              N,1,1,4
```

AFTER CREATION OF JAR FILE MATRIX MUL EXECUTION PROCESS

STEP 1: After Creation of Jar file

--open command prompt in cloudera..

---create/set specified path---ex: you have save a files on desktop---use

----cd Desktop

STEP 2:

--Create Input file and store data

Ex: specified input data using for experiment .

```
matinput.txt  M,0,0,1
               M,0,1,2
               M,1,0,3
               M,1,1,4
               N,0,0,1
               N,0,1,2
               N,1,0,3
               N,1,1,4
```

STEP 3:

-----create a directory/folder in a Hadoop(store all data)

-----hadoop fs -mkdir /foldername

----ex: Hadoop fs -mkdir /Nagababu

Note: check the folder create or not

----hadoop fs -ls

STEP 4:

-----copy the input file to Hadoop

-----hadoop fs -put inputfilename /destination folder

---ex: Hadoop fs -put matinput.txt /Nagababu

Note: check the input file copy or not in specified directory

-----hadoop fs -ls /Nagababu/

STEP 5:

----Execute file and store output in particular directory or same directory...using command

----Hadoop jar filename.jar /folder name/input filename /folder name/output storing foldername

----ex: hadoop jar matrix.jar /Nagababu/matinput.txt /Nagababu/matrixoutput

Process executed.....

STEP 6:

Check the output

hadoop fs -ls /folder name/output

----ex: hadoop fs -ls /Nagababu/matrixoutput

Note : the output folder is after execution stored a two file 1.Success file

2.part file(stored out put)

STEP 7:

final out put enter command

-----hadoop fs -cat /foldername/outputfolder/part*

-----Ex: hadoop fs -cat /Nagababu/matrixoutput/part*

Output :

0,0,7

0,1,10

1,0,15

1,0,22

Experiment #04

AIM: Implementation of Weather mining by taking weather data set using map reduce.

MyMaxMin.java

```
import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

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

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

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

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

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.Configuration;

public class MyMaxMin {

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

    {

        public static final int MISSING=9999;
```

```

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

{

String line = Value.toString();

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

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

float temp_Min = Float.parseFloat(line.substring(39, 45).trim());

float temp_Max = Float.parseFloat(line.substring(47, 53).trim());

if (temp_Max > 3.0 && temp_Max!=MISSING)

{

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

}

if (temp_Min < 0 && temp_Max!=MISSING)

{

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

}

}

}

}

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

{

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

```



```

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

public static void main(String[] args) throws Exception
{
Configuration conf = new Configuration();
Job job = new Job(conf, "weather example");
job.setJarByClass(MyMaxMin.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setMapperClass(MaxTemperatureMapper.class);
job.setReducerClass(MaxTemperatureReducer.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);

Path outputPath = new Path(args[1]);

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

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

```

Weather Mining Procedure on Cloudera tool

Example:

Input:

-----Create/take a input file from:

-----URL:www.ncei.noaa.gov/process/crn/

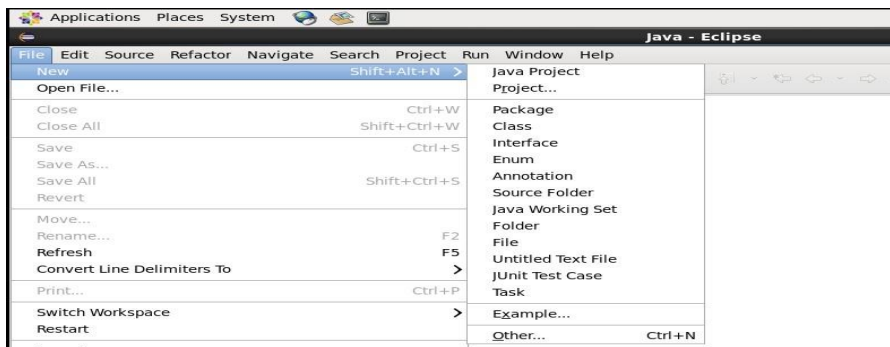
----- clock Get datasets/monthly/data accessFile

-----copy the data and store on desktop using

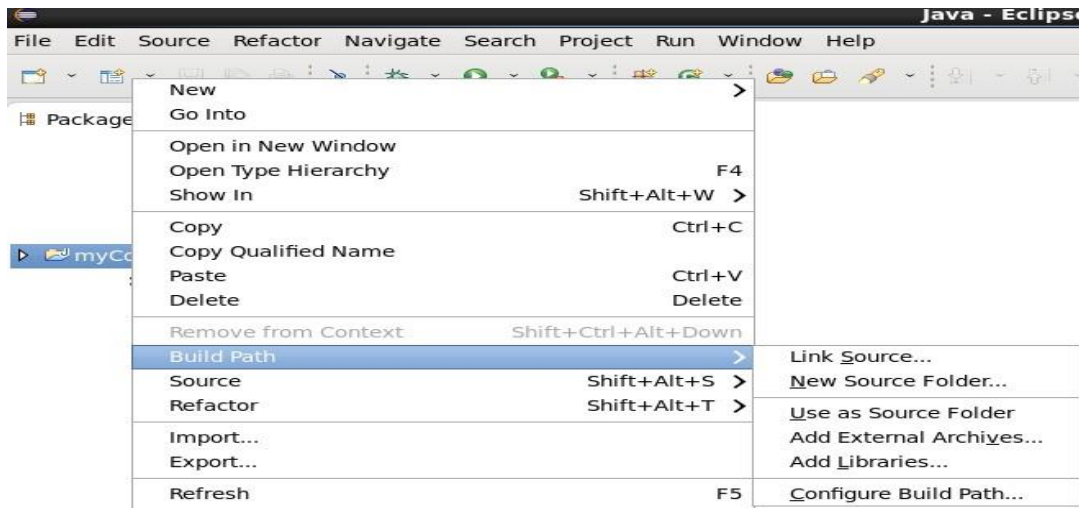
File name is “Temparature.txt”

Steps:

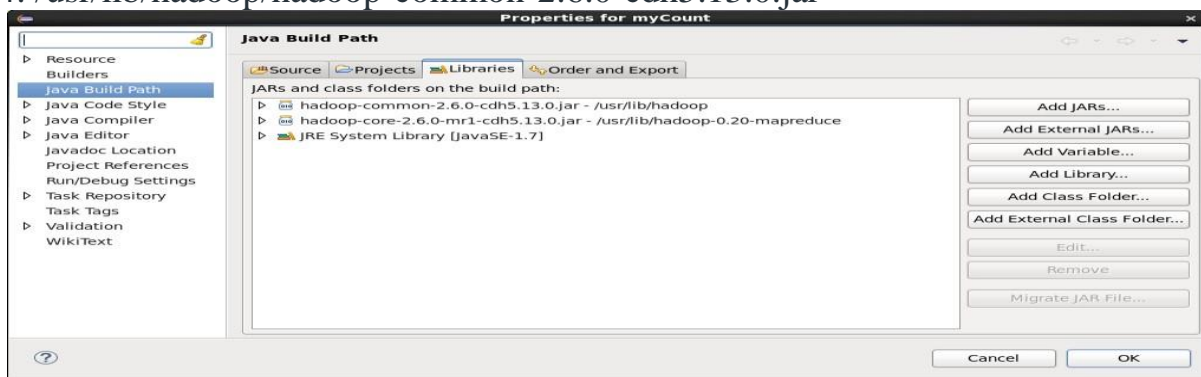
- First Open **Eclipse** -> then select **File** -> **New** -> **Java Project** ->Name it **WeatherMining** -> then **Finish**.



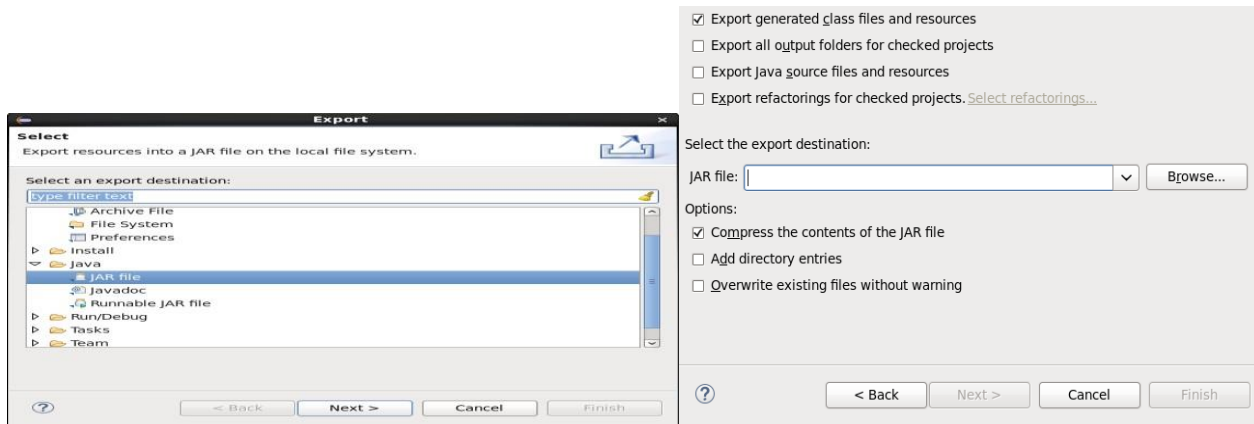
- Create one Java Classes into the project. Name them **MyMaxMin**(having the main function).
- You have to include two Reference Libraries for that:
Right Click on **Project** -> then select **Build Path**-> Click on **Configure Build Path**



- In the above figure, you can see the Add External JARs option on the Right Hand Side. Click on it and add the below mention files. You can find these files in `/usr/lib/`
 3. `/usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar`
 4. `/usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar`



- **Mapper+ Reducer + Driver Code:** You have to copy paste this program into the MyMaxMin Java Class file.
- **After complete**
- Now you have to make a jar file. Right Click on **Project**-> **Click on Export**-> **Select export destination as Jar File**-> **Name the jar File(WeatherMining.jar)** -> **Click on next** -> at last **Click on Finish**.



- Open the terminal on CDH and change the directory to the workspace. You can do this by using “cd workspace/” command. Now, Create a text file(**Temperature.txt**) and move it to HDFS. For that open terminal and write this code(remember you should be in the same directory as jar file you have created just now).

AFTER CREATION OF JAR FILE Weather(Temaratue) EXECUTION PROCESS

STEP 1: After Creation of Jar file

--open command prompt in cloudera..

---create/set specified path---ex: you have save a files on desktop---use

----cd Desktop

STEP 2:

--Create Input file and store data

Ex: specified input data using for experiment .

STEP 3:

-----create a directory/folder in a Hadoop(store all data)

-----hadoop fs -mkdir /foldername

----ex: Hadoop fs -mkdir /Nagababu

Note: check the folder create or not

----hadoop fs -ls

STEP 4:

-----copy the input file to Hadoop

----hadoop fs -put inputfilename /destination folder

---ex: Hadoop fs -put Temperatureinput.txt /Nagababu

Note: check the input file copy or not in specified directory

----hadoop fs -ls /Nagababu/

STEP 5:

----Execute file and store output in particular directory or same directory...using command

----Hadoop jar filename.jar /folder name/input filename /folder name/output storing foldername

----ex: hadoop jar matrix.jar /Nagababu/Temperatureinput.txt /Nagababu/Weatheroutput

Process executed.....

STEP 6:

Check the output

hadoop fs -ls /folder name/output

----ex: hadoop fs -ls /Nagababu/Weatheroutput

Note : the output folder is after execution stored a two file 1.Success file

2.part file(stored out put)

STEP 7:

final out put enter command

-----hadoop fs -cat /foldername/outputfolder/part*

-----Ex: hadoop fs -cat /Nagababu/Weatheroutput/part*

Output :

Experiment #5

AIM: Installation of Hive along with practice examples.

The following steps are required for installing Hive on your system. Let us assume the Hive archive is downloaded onto the /Downloads directory.

Extracting and verifying Hive Archive

The following command is used to verify the download and extract the hive archive:

```
$ tar zxvf apache-hive-0.14.0-bin.tar.gz  
$ ls
```

On successful download, you get to see the following response:

```
apache-hive-0.14.0-bin apache-hive-0.14.0-bin.tar.gz
```

Copying files to /usr/local/hive directory

We need to copy the files from the super user “su -”. The following commands are used to copy the files from the extracted directory to the /usr/local/hive” directory.

```
$ su -  
passwd:  
  
# cd /home/user/Download  
# mv apache-hive-0.14.0-bin /usr/local/hive  
# exit
```

Setting up environment for Hive

You can set up the Hive environment by appending the following lines to ~/.bashrc file:

```
export HIVE_HOME=/usr/local/hive  
export PATH=$PATH:$HIVE_HOME/bin  
export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/*:..  
export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/*:..
```

The following command is used to execute ~/.bashrc file.

```
$ source ~/.bashrc
```

Configuring Hive

To configure Hive with Hadoop, you need to edit the **hive-env.sh** file, which is placed in the **\$HIVE_HOME/conf** directory. The following commands redirect to Hive **config** folder and copy the template file:

```
$ cd $HIVE_HOME/conf
$ cp hive-env.sh.template hive-env.sh
```

Edit the **hive-env.sh** file by appending the following line:

```
export HADOOP_HOME=/usr/local/hadoop
```

Hive installation is completed successfully. Now you require an external database server to configure Metastore. We use Apache Derby database.

Downloading and Installing Apache Derby

Follow the steps given below to download and install Apache Derby:

Downloading Apache Derby

The following command is used to download Apache Derby. It takes some time to download.

```
$ cd ~
$ wget http://archive.apache.org/dist/db/derby/db-derby-10.4.2.0/db-derby-10.4.2.0-bin.tar.gz
```

The following command is used to verify the download:

```
$ ls
```

On successful download, you get to see the following response:

```
db-derby-10.4.2.0-bin.tar.gz
```

Extracting and verifying Derby archive

The following commands are used for extracting and verifying the Derby archive:

```
$ tar zxvf db-derby-10.4.2.0-bin.tar.gz
$ ls
```

On successful download, you get to see the following response:

```
db-derby-10.4.2.0-bin db-derby-10.4.2.0-bin.tar.gz
```

Copying files to /usr/local/derby directory

We need to copy from the super user “su -”. The following commands are used to copy the files from the extracted directory to the /usr/local/derby directory:

```
$ su -
passwd:
# cd /home/user
# mv db-derby-10.4.2.0-bin /usr/local/derby
```



```
# exit
```

Setting up environment for Derby

You can set up the Derby environment by appending the following lines to `~/.bashrc` file:

```
export DERBY_HOME=/usr/local/derby
export PATH=$PATH:$DERBY_HOME/bin
Apache Hive
18
export
CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbyto
ols.jar
```

The following command is used to execute `~/.bashrc` file:

```
$ source ~/.bashrc
```

Create a directory to store Metastore

Create a directory named `data` in `$DERBY_HOME` directory to store Metastore data.

```
$ mkdir $DERBY_HOME/data
```

Derby installation and environmental setup is now complete.

Configuring Metastore of Hive

Configuring Metastore means specifying to Hive where the database is stored. You can do this by editing the `hive-site.xml` file, which is in the `$HIVE_HOME/conf` directory. First of all, copy the template file using the following command:

```
$ cd $HIVE_HOME/conf
$ cp hive-default.xml.template hive-site.xml
```

Edit **hive-site.xml** and append the following lines between the `<configuration>` and `</configuration>` tags:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:derby://localhost:1527/metastore_db;create=true </value>
  <description>JDBC connect string for a JDBC metastore </description>
</property>
```

Create a file named `jpo.x.properties` and add the following lines into it:

```
javax.jdo.PersistenceManagerFactoryClass =
```

```
org.jpox.PersistenceManagerFactoryImpl
org.jpox.autoCreateSchema = false
org.jpox.validateTables = false
org.jpox.validateColumns = false
org.jpox.validateConstraints = false
org.jpox.storeManagerType = rdbms
org.jpox.autoCreateSchema = true
org.jpox.autoStartMechanismMode = checked
org.jpox.transactionIsolation = read_committed
javax.jdo.option.DetachAllOnCommit = true
javax.jdo.option.NontransactionalRead = true
javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver
javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore_db;create = true
javax.jdo.option.ConnectionUserName = APP
javax.jdo.option.ConnectionPassword = mine
```

Verifying Hive Installation

Before running Hive, you need to create the **/tmp** folder and a separate Hive folder in HDFS. Here, we use the **/user/hive/warehouse** folder. You need to set write permission for these newly created folders as shown below:

```
chmod g+w
```

Now set them in HDFS before verifying Hive. Use the following commands:

```
$ $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp
$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse
```

The following commands are used to verify Hive installation:

```
$ cd $HIVE_HOME
$ bin/hive
```

On successful installation of Hive, you get to see the following response:

```
Logging initialized using configuration in jar:file:/home/hadoop/hive-0.9.0/lib/hive-common-0.9.0.jar!/hive-log4j.properties
```

```
Hive history file=/tmp/hadoop/hive_job_log_hadoop_201312121621_1494929084.txt
```

```
.....
```

```
hive>
```

The following sample command is executed to display all the tables:

```
hive> show tables;
```

```
OK
```

```
Time taken: 2.798 seconds
```

```
hive>
```

HIVE COMMANDS

Hive is a database technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it. This chapter explains how to create Hive database. Hive contains a default database named **default**.

Create Database Statement

Create Database is a statement used to create a database in Hive. A database in Hive is a **namespace** or a collection of tables. The **syntax** for this statement is as follows:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command. The following query is executed to create a database named **userdb**:

```
hive> CREATE DATABASE [IF NOT EXISTS] userdb;
```

or

```
hive> CREATE SCHEMA userdb;
```

The following query is used to verify a databases list:

```
hive> SHOW DATABASES;  
default  
userdb
```

Drop Database Statement

Drop Database is a statement that drops all the tables and deletes the database. Its syntax is as follows:

```
DROP DATABASE Statement DROP (DATABASE|SCHEMA) [IF EXISTS] database_name  
[RESTRICT|CASCADE];
```

The following queries are used to drop a database. Let us assume that the database name is **userdb**.

```
hive> DROP DATABASE IF EXISTS userdb;
```

The following query drops the database using **CASCADE**. It means dropping respective tables before dropping the database.

```
hive> DROP DATABASE IF EXISTS userdb CASCADE;
```

The following query drops the database using **SCHEMA**.

```
hive> DROP SCHEMA userdb;
```

This clause was added in Hive 0.6.

Create Table Statement

Create Table is a statement used to create a table in Hive. The syntax and example are as follows:

Syntax

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name  
[(col_name data_type [COMMENT col_comment], ...)]  
[COMMENT table_comment]  
[ROW FORMAT row_format]  
[STORED AS file_format]
```

Example

Let us assume you need to create a table named **employee** using **CREATE TABLE** statement. The following table lists the fields and their data types in employee table:

Sr.No	Field Name	Data Type
1	Eid	int
2	Name	String
3	Salary	Float
4	Designation	string

The following data is a Comment, Row formatted fields such as Field terminator, Lines terminator, and Stored File type.

```
COMMENT 'Employee details'
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED IN TEXT FILE
```

The following query creates a table named **employee** using the above data.

```
hive> 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;
```

If you add the option IF NOT EXISTS, Hive ignores the statement in case the table already exists.

On successful creation of table, you get to see the following response:

Time taken: 5.905 seconds

hive>

Alter Table Statement

It is used to alter a table in Hive.

Syntax

The statement takes any of the following syntaxes based on what attributes we wish to modify in a table.

```
ALTER TABLE name RENAME TO new_name
```

```
ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
```

```
ALTER TABLE name DROP [COLUMN] column_name
```

```
ALTER TABLE name CHANGE column_name new_name new_type
```

```
ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])
```

Rename To... Statement

The following query renames the table from **employee** to **emp**.

```
hive> ALTER TABLE employee RENAME TO emp; Drop Table Statement
```

The syntax is as follows:

```
DROP TABLE [IF EXISTS] table_name;
```

The following query drops a table named **employee**:

```
hive> DROP TABLE IF EXISTS employee;
```

On successful execution of the query, you get to see the following response:

```
OK
Time taken: 5.3 seconds
hive>
```

Experiment #6

AIM: Installation of Sqoop along with Practice examples.

Installation of sqoop:

Step 1: Verifying JAVA Installation

You need to have Java installed on your system before installing Sqoop. Let us verify Java installation using the following command –

```
$ java -version
```

If Java is already installed on your system, you get to see the following response –

```
java version "1.7.0_71"  
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)  
Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
```

If Java is not installed on your system, then follow the steps given below.

Installing Java

Follow the simple steps given below to install Java on your system.

Step 1

Download Java (JDK <latest version> - X64.tar.gz) by visiting the following [link](#).

Then jdk-7u71-linux-x64.tar.gz will be downloaded onto your system.

Step 2

Generally, you can find the downloaded Java file in the Downloads folder. Verify it and extract the jdk-7u71-linux-x64.gz file using the following commands.

```
$ cd Downloads/  
$ ls  
jdk-7u71-linux-x64.gz  
$ tar xzf jdk-7u71-linux-x64.gz  
$ ls  
jdk1.7.0_71 jdk-7u71-linux-x64.gz
```

Step 3

To make Java available to all the users, you have to move it to the location “/usr/local/”. Open root, and type the following commands.

```
$ su  
password:  
  
# mv jdk1.7.0_71 /usr/local/java  
# exit
```

Step 4

For setting up PATH and JAVA_HOME variables, add the following commands to ~/.bashrc file.

```
export JAVA_HOME=/usr/local/java
export PATH=$PATH:$JAVA_HOME/bin
```

Now apply all the changes into the current running system.

```
$ source ~/.bashrc
```

Step 5

Use the following commands to configure Java alternatives –

```
# alternatives --install /usr/bin/java java usr/local/java/bin/java 2
# alternatives --install /usr/bin/javac javac usr/local/java/bin/javac 2
# alternatives --install /usr/bin/jar jar usr/local/java/bin/jar 2

# alternatives --set java usr/local/java/bin/java
# alternatives --set javac usr/local/java/bin/javac
# alternatives --set jar usr/local/java/bin/jar
```

Now verify the installation using the command **java -version** from the terminal as explained above.

Step 2: Verifying Hadoop Installation

Hadoop must be installed on your system before installing Sqoop. Let us verify the Hadoop installation using the following command –

```
$ hadoop version
```

If Hadoop is already installed on your system, then you will get the following response –

```
Hadoop 2.4.1
--
Subversion https://svn.apache.org/repos/asf/hadoop/common -r 1529768
Compiled by hortonmu on 2013-10-07T06:28Z
Compiled with protoc 2.5.0
From source with checksum 79e53ce7994d1628b240f09af91e1af4
```

If Hadoop is not installed on your system, then proceed with the following steps –

Downloading Hadoop

Download and extract Hadoop 2.4.1 from Apache Software Foundation using the following commands.


```
$ su
password:

# cd /usr/local
# wget http://apache.claz.org/hadoop/common/hadoop-2.4.1/
hadoop-2.4.1.tar.gz
# tar xzf hadoop-2.4.1.tar.gz
# mv hadoop-2.4.1/* to hadoop/
# exit
```

Installing Hadoop in Pseudo Distributed Mode

Follow the steps given below to install Hadoop 2.4.1 in pseudo-distributed mode.

Step 1: Setting up Hadoop

You can set Hadoop environment variables by appending the following commands to ~/.bashrc file.

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

Now, apply all the changes into the current running system.

```
$ source ~/.bashrc
```

Step 2: Hadoop Configuration

You can find all the Hadoop configuration files in the location “\$HADOOP_HOME/etc/hadoop”. You need to make suitable changes in those configuration files according to your Hadoop infrastructure.

```
$ cd $HADOOP_HOME/etc/hadoop
```

In order to develop Hadoop programs using java, you have to reset the java environment variables in **hadoop-env.sh** file by replacing JAVA_HOME value with the location of java in your system.

```
export JAVA_HOME=/usr/local/java
```

Given below is the list of files that you need to edit to configure Hadoop.

core-site.xml

The core-site.xml file contains information such as the port number used for Hadoop instance, memory allocated for the file system, memory limit for storing the data, and the size of Read/Write buffers.

Open the core-site.xml and add the following properties in between the <configuration> and </configuration> tags.

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

hdfs-site.xml

The hdfs-site.xml file contains information such as the value of replication data, namenode path, and datanode path of your local file systems. It means the place where you want to store the Hadoop infrastructure.

Let us assume the following data.

dfs.replication (data replication value) = 1

(In the following path /hadoop/ is the user name.

hadoopinfra/hdfs/namenode is the directory created by hdfs file system.)

namenode path = //home/hadoop/hadoopinfra/hdfs/namenode

(hadoopinfra/hdfs/datanode is the directory created by hdfs file system.)

datanode path = //home/hadoop/hadoopinfra/hdfs/datanode

Open this file and add the following properties in between the <configuration>, </configuration> tags in this file.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/namenode </value>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/datanode </value>
  </property>
</configuration>
```

Note – In the above file, all the property values are user-defined and you can make changes according to your Hadoop infrastructure.

yarn-site.xml

This file is used to configure yarn into Hadoop. Open the yarn-site.xml file and add the following properties in between the <configuration>, </configuration> tags in this file.

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

mapred-site.xml

This file is used to specify which MapReduce framework we are using. By default, Hadoop contains a template of yarn-site.xml. First of all, you need to copy the file from mapred-site.xml.template to mapred-site.xml file using the following command.

```
$ cp mapred-site.xml.template mapred-site.xml
```

Open mapred-site.xml file and add the following properties in between the <configuration>, </configuration> tags in this file.

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

Verifying Hadoop Installation

The following steps are used to verify the Hadoop installation.

Step 1: Name Node Setup

Set up the namenode using the command “hdfs namenode -format” as follows.

```
$ cd ~  
$ hdfs namenode -format
```

The expected result is as follows.

```
10/24/14 21:30:55 INFO namenode.NameNode: STARTUP_MSG:  
/*****  
STARTUP_MSG: Starting NameNode  
STARTUP_MSG: host = localhost/192.168.1.11  
STARTUP_MSG: args = [-format]  
STARTUP_MSG: version = 2.4.1  
...  
...  
10/24/14 21:30:56 INFO common.Storage: Storage directory  
/home/hadoop/hadoopinfra/hdfs/namenode has been successfully formatted.  
10/24/14 21:30:56 INFO namenode.NNStorageRetentionManager: Going to  
retain 1 images with txid >= 0  
10/24/14 21:30:56 INFO util.ExitUtil: Exiting with status 0  
10/24/14 21:30:56 INFO namenode.NameNode: SHUTDOWN_MSG:  
/*****
```

```
SHUTDOWN_MSG: Shutting down NameNode at localhost/192.168.1.11  
*****/
```

Step 2: Verifying Hadoop dfs

The following command is used to start dfs. Executing this command will start your Hadoop file system.

```
$ start-dfs.sh
```

The expected output is as follows –

```
10/24/14 21:37:56  
Starting namenodes on [localhost]  
localhost: starting namenode, logging to /home/hadoop/hadoop-  
2.4.1/logs/hadoop-hadoop-namenode-localhost.out  
localhost: starting datanode, logging to /home/hadoop/hadoop-  
2.4.1/logs/hadoop-hadoop-datanode-localhost.out  
Starting secondary namenodes [0.0.0.0]
```

Step 3: Verifying Yarn Script

The following command is used to start the yarn script. Executing this command will start your yarn daemons.

```
$ start-yarn.sh
```

The expected output is as follows –

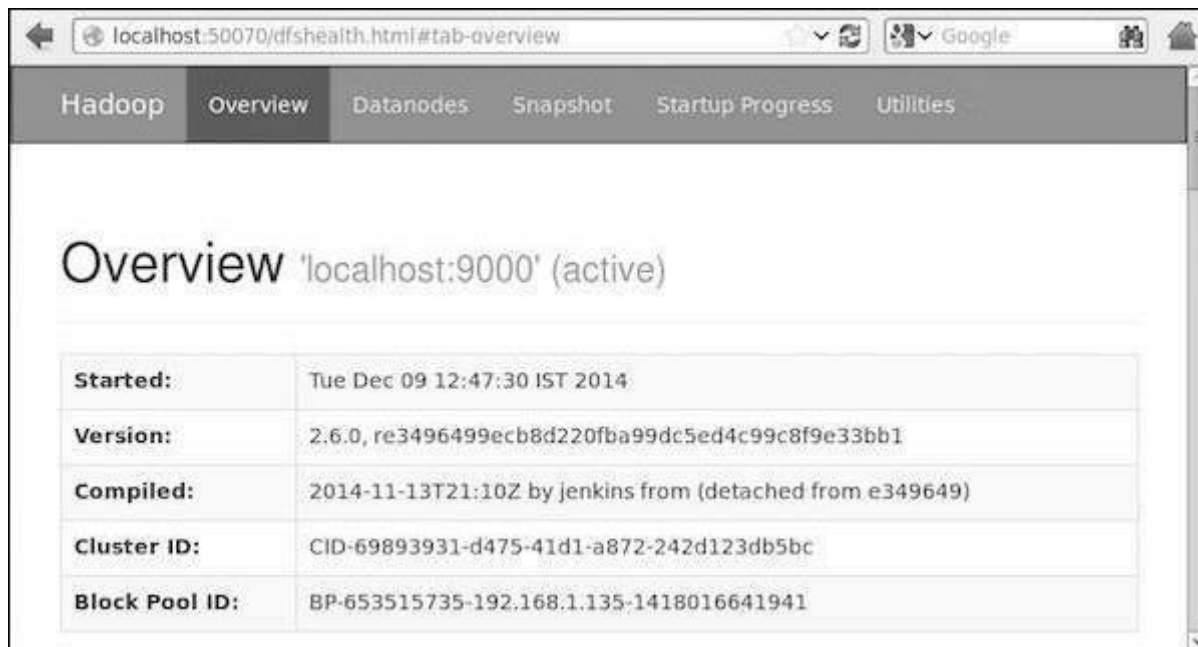
```
starting yarn daemons  
starting resourcemanager, logging to /home/hadoop/hadoop-  
2.4.1/logs/yarn-hadoop-resourcemanager-localhost.out  
localhost: starting node manager, logging to /home/hadoop/hadoop-  
2.4.1/logs/yarn-hadoop-nodemanager-localhost.out
```

Step 4: Accessing Hadoop on Browser

The default port number to access Hadoop is 50070. Use the following URL to get Hadoop services on your browser.

```
http://localhost:50070/
```

The following image depicts a Hadoop browser.

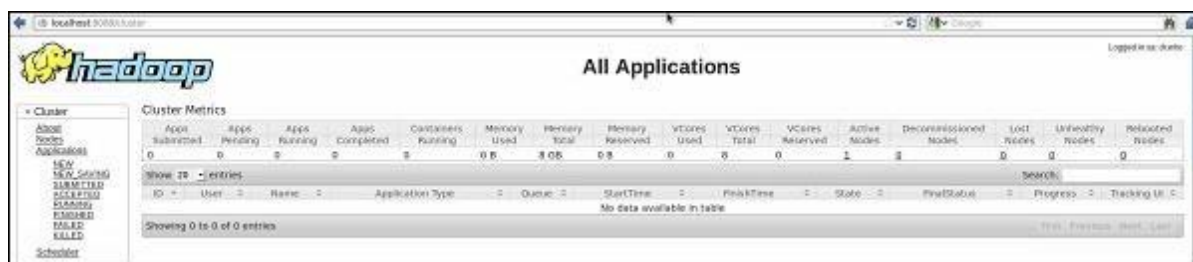


Step 5: Verify All Applications for Cluster

The default port number to access all applications of cluster is 8088. Use the following url to visit this service.

<http://localhost:8088/>

The following image depicts the Hadoop cluster browser.



Step 3: Downloading Sqoop

We can download the latest version of Sqoop from the following [link](#) For this tutorial, we are using version 1.4.5, that is, **sqoop-1.4.5.bin__hadoop-2.0.4-alpha.tar.gz**.

Step 4: Installing Sqoop

The following commands are used to extract the Sqoop tar ball and move it to “/usr/lib/sqoop” directory.

```
$tar -xvf sqoop-1.4.4.bin__hadoop-2.0.4-alpha.tar.gz
$ su
password:

# mv sqoop-1.4.4.bin__hadoop-2.0.4-alpha /usr/lib/sqoop
#exit
```

Step 5: Configuring bashrc

You have to set up the Sqoop environment by appending the following lines to ~/.bashrc file –

```
#Sqoop
export SQOOP_HOME=/usr/lib/sqoop export PATH=$PATH:$SQOOP_HOME/bin
```

The following command is used to execute ~/.bashrc file.

```
$ source ~/.bashrc
```

Step 6: Configuring Sqoop

To configure Sqoop with Hadoop, you need to edit the **sqoop-env.sh** file, which is placed in the **\$\$SQOOP_HOME/conf** directory. First of all, Redirect to Sqoop config directory and copy the template file using the following command –

```
$ cd $$SQOOP_HOME/conf
$ mv sqoop-env-template.sh sqoop-env.sh
```

Open **sqoop-env.sh** and edit the following lines –

```
export HADOOP_COMMON_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=/usr/local/hadoop
```

Step 7: Download and Configure mysql-connector-java

We can download **mysql-connector-java-5.1.30.tar.gz** file from the following [link](#).

The following commands are used to extract mysql-connector-java tarball and move **mysql-connector-java-5.1.30-bin.jar** to /usr/lib/sqoop/lib directory.

```
$ tar -zxf mysql-connector-java-5.1.30.tar.gz
$ su
password:

# cd mysql-connector-java-5.1.30
# mv mysql-connector-java-5.1.30-bin.jar /usr/lib/sqoop/lib
```

Step 8: Verifying Sqoop

The following command is used to verify the Sqoop version.

```
$ cd $$SQOOP_HOME/bin
$ sqoop-version
```

Expected output –

```
14/12/17 14:52:32 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5
Sqoop 1.4.5 git commit id 5b34accaca7de251fc91161733f906af2eddb83
Compiled by abe on Fri Aug 1 11:19:26 PDT 2014
```

Sqoop installation is complete.

SQOOP EXECUTION:

```
mysql -u root -p
```

```
create database demo;
```

```
use demo;
```

```
CREATE TABLE Employee (Id int Not NULL, Name VARCHAR(20),DOB DATE,Gender  
char(1), Salary int, PRIMARY KEY (Id));
```

```
insert into Employee values (1, 'Alex', '1987-12-12','M', 10000);
```

```
insert into Employee values (2, 'Julia', '1987-11-11', 'F', 20000);
```

```
insert into Employee values (3, 'John', '1987-10-10', 'M', 25000);
```

```
insert into Employee values (4, 'Khalishi', '1987-9-9', 'F', 30000);
```

```
sqoop import --connect jdbc:mysql://localhost/demo --username root --password cloudera --  
table Employee
```

```
hadoop dfs -cat Employee/*
```

```
insert into Employee values (5, 'JonSnow', '1987-8-9', 'M', 35000);
```

```
insert into Employee values (6, 'Arya', '1993-7-7', 'F', 40000);
```

```
sqoop import --connect jdbc:mysql://localhost/demo --username root --password cloudera --  
table Employee --incremental append --check-column id --last-value 4
```

```
hadoop dfs -cat Employee/*
```

Exporting HDFS Data to MySql Table

```
CREATE TABLE EmployeeNew (Id int Not NULL, Name VARCHAR(20),DOB DATE, _  
Gender char(1), Salary int, PRIMARY KEY (Id));
```

```
sqoop export --connect jdbc:mysql://localhost/demo --username root --password cloudera --  
export-dir /user/cloudera/Employee --update-mode allowinsert --table EmployeeNew
```

```
Select * from EmployeeNew;
```


Experiment #7

AIM: Downloading and installing Tableau Understanding about importing data , saving ,opening and sharing work books.

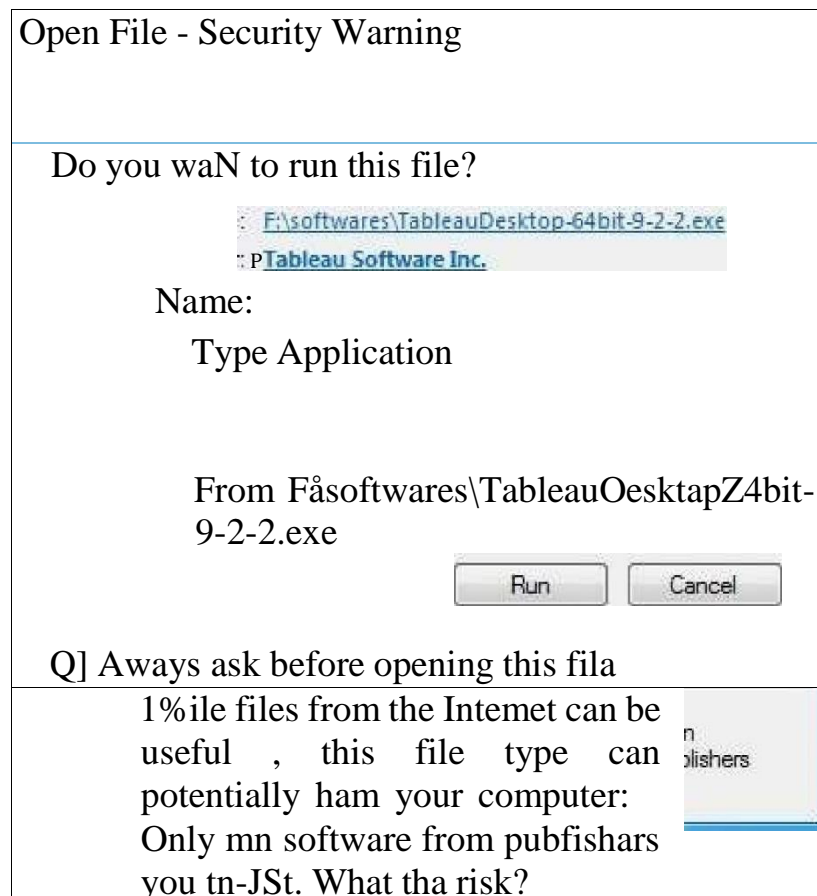
Download Tableau Desktop

The Free Personal Edition of Tableau Desktop can be downloaded from Tableau Desktop. You need to register with your details to be able to download.

After downloading, the installation is a very straightforward process in which you need to accept the license agreement and provide the target folder for installation. The following steps and screenshots describe the entire setup process.

Start the Installation Wizard

Double-click the TableauDesktop-64bit-9-2-2.exe. It will present a screen to allow the installation program to run. Click "Run".



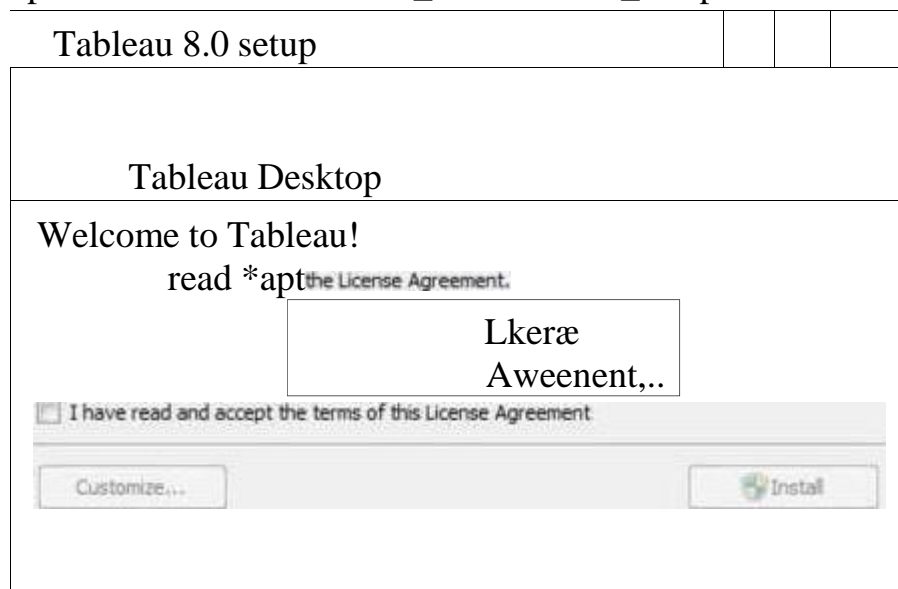
Accept the License Agreement

Read the license agreement and if you agree, choose the "I have read and accept the terms of

this license agreement" option. Then, click "Install".

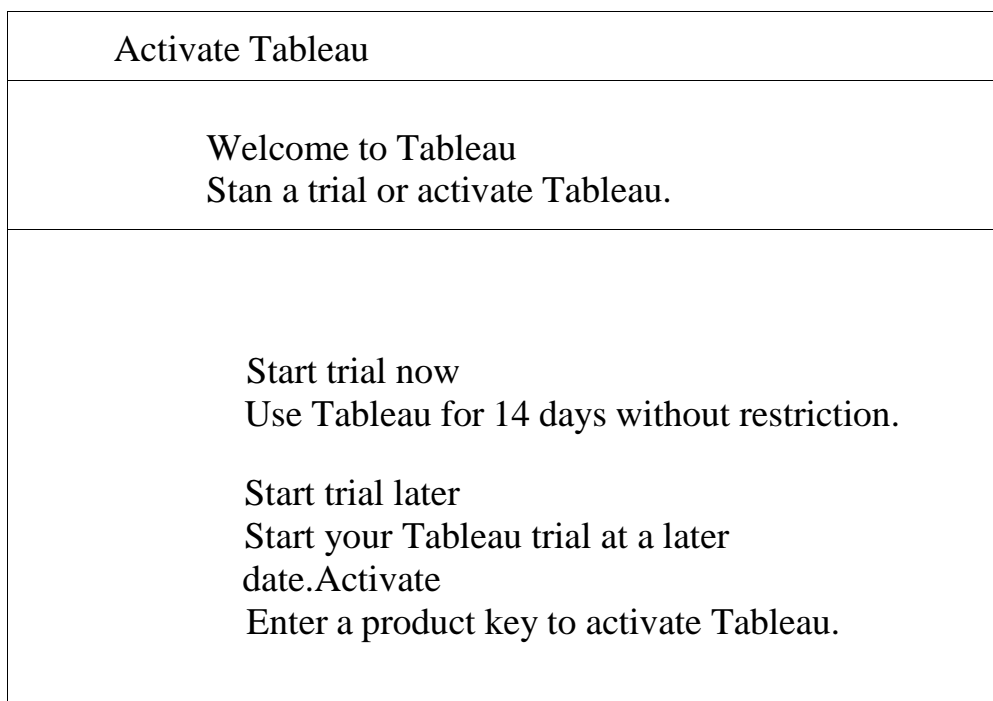
https://www.tutorialspoint.com/tableau/tableau_environment_setup.htm

1/4 FM



Start Trial

On completion Of the installation, the screen prompts you with the option to Start the trial now or later. You may choose to start it now. Also, if you have purchased Tableau then you may enter the License key.



Exit

Provide Your Details

Provide your name and organization details. Then, dick "Next"

Activate Tableau

Registration

++ Please complete all fields for the registered user.

First Name Last Name

Organization

Email

Phone

Job Title

CityPostal CodeDepartment

Country/RegionState/ProvinceIndustry

ster

Registration Complete

The registration completion screen appears. Click "Continue"

Activate Tableau	
++	Registration Start using the product.
Registration completed.	
<div>Continue</div>	

Verify the Installation

You can verify the installation by going to the Windows start menu. Click the Tableau icon. The following screen appears.

Tebleeu -r Book I

FileData

+

+

+

+

Connect

To a file

Excel

Text File

Access

Statistical File

Other Files

To a server

Tableau Server

Microsoft SQL Server

MySQL

Oracle

Amazon Redshift

More Servers...


Saved data sources

Sample - Superstore


World Indicators

Open

Sampte Workbooks



Superstore



Regional

Experiment #08

AIM: Data preparation with Tableau.

Tableau Prep offers various cleaning operations that you can use to clean and shape your data. Cleaning up dirty data makes it easier to combine and analyze your data or makes it easier for others to understand your data when sharing your data sets.

You can also clean your data using a pivot step or a script step to apply R or Python scripts to your flow. Script steps aren't supported in Tableau Cloud. For more information, see [Pivot Your Data\(Link opens in a new window\)](#) or [Use R and Python scripts in your flow](#)

About cleaning operations

You clean data by applying cleaning operations such as filtering, adding, renaming, splitting, grouping, or removing fields. You can perform cleaning operations in most step types in your flow. You can also perform cleaning operations in the data grid in a cleaning step.

You can apply limited cleaning operations in the Input step and can't apply cleaning operations in the output step. For more information about applying cleaning operations in the Input step, see [Apply cleaning operations in an input step\(Link opens in a new window\)](#).

Available cleaning operations

The following table shows which cleaning operations are available in each step type:

Input	Clean	Aggregate	Pivot	Join	Union	New Rows	Output	
Filter	X	X	X	X	X	X	X	
Group Values		X		X		X	X	
Clean		X		X	X	X	X	
Convert Dates		X	X	X	X	X	X	
Split Values		X		X	X	X	X	
Rename Field	X	X		X	X	X	X	
Rename Fields (in bulk)		X						
Duplicate Field		X		X	X	X	X	
Keep Only Field	X	X	X	X	X	X	X	
Remove Field	X	X	X	X	X	X	X	
Create Calculated Field		X		X	X	X	X	

Edit Value		X		X	X	X	X	
Change Data Type	X	X	X	X	X	X	X	

As you make changes to your data, annotations are added to the corresponding step in the **Flow** pane and an entry is added in the **Changes** pane to track your actions. If you make changes in the Input step, the annotation shows to the left of the step in the **Flow** pane and shows in the **Input profile** in the field list.

The order that you apply your changes matters. Changes made in Aggregate, Pivot, Join, and Union step types are performed either before or after those cleaning actions, depending on where the field is when you make the change. Where the change was made is shown in the **Changes** pane for the step.

The following example shows changes made to several fields in a Join step. The change is performed before the join action to give the corrected results.

The screenshot displays the Alteryx workflow and the 'Changes' pane for a Join step. The workflow consists of the following steps: 'Orders (Central)' (Input), 'Fix Dates' (Transform), 'All Orders' (Aggregate), 'Returns (all)' (Input), 'Clean Notes/Approver' (Transform), and 'Orders + Returns' (Join). The 'Orders + Returns' step is highlighted with a blue box in the flow.

The 'Changes (8)' pane for the 'Orders + Returns' step shows the following settings:

- Remove Field:** Table Names, File Paths
- Calculated Field:** Discount (IFNULL([Discount],0))
- Calculated Field:** Year of Sale (YEAR([Order Date]))
- Filter:** Discount (Exclude: (17.0 - 18.0))
- Join:** [Product ID] == [Product ID],[Order ID] == [Order ID]

The 'Join Clauses' section shows the 'Clean Notes/Approver' and 'All Orders' tables. The 'Clean Notes/Approver' table has columns 'Product ID' and 'Order ID'. The 'All Orders' table has columns 'Product ID' and 'Order ID'. The 'Join Clauses' section also includes a 'Show only mismatched values' dropdown.

Order of operations

The following table shows where the cleaning action is performed in Aggregate, Pivot, Join, and Union step types depending on where the field is in the step.

Action	Step Type:	Aggregate	Aggregate	Pivot	Pivot	Join	Join	Union	Union	New Rows
	<i>Field Location:</i>	<i>Grouped fields</i>	<i>Aggregated fields</i>	<i>Not in pivot</i>	<i>Created from pivot</i>	<i>Included in one table*</i>	<i>Included in both tables *</i>	<i>Mismatched fields</i>	<i>Combined fields</i>	<i>Field used to generate rows</i>
Filter		Before Aggregation	After Aggregation	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows
Group Values		NA	NA	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows
Clean		NA	NA	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows
Convert Dates		Before Aggregation	After Aggregation	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows
Split Values		NA	NA	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows
Rename Field		NA	NA	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	Before New Rows
Duplicate Field		NA	NA	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows

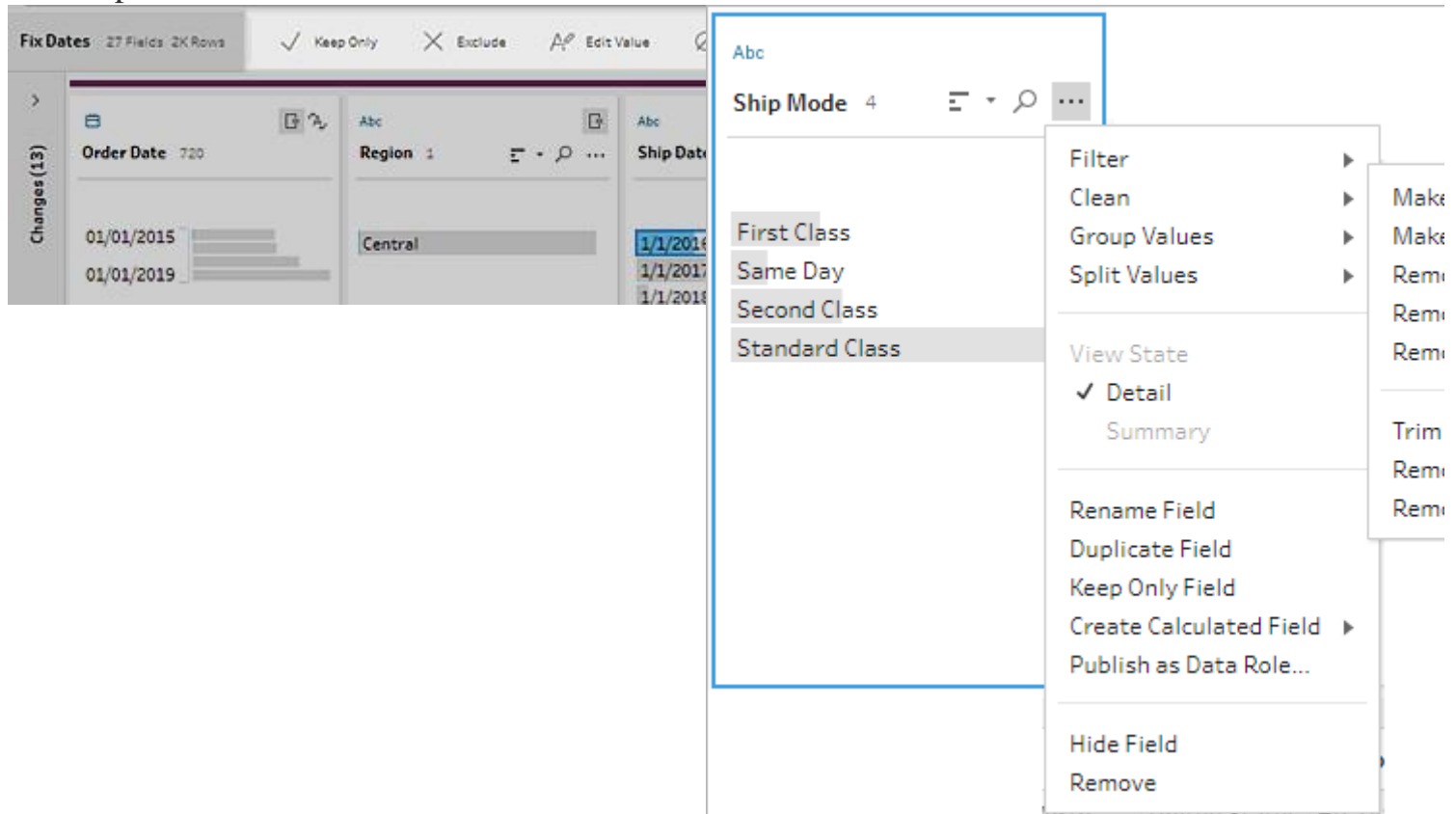
Keep Only Field		After Aggregation	After Aggregation	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows
Remove Field		Removes from Aggregation	Removes from Aggregation	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows
Create Calculated Field		NA	NA	Before Pivot	After Pivot	After Join	After Join	Before Union	After Union	After New Rows
Edit Value		NA	NA	Before Pivot	After Pivot	Before Join	After Join	Before Union	After Union	After New Rows
Change Data Type		Before Aggregation	After Aggregation	Before Pivot	After Pivot	Before Join	Before Join	Before Union	After Union	Before New Rows

Note: For joins, if the field is a calculated field that was created using a field from one table, the change is applied before the join. If the field is created with fields from both tables, the change is applied after the join.


Apply cleaning operations. To apply cleaning operations to fields, use the toolbar options or click **More options** ... on the field profile card, data grid, or Results pane to open the menu.

In Aggregate, Pivot, Join, and Union step types, the **More options** menu is available on the profile cards in the Results pane and corresponding data grid. If you perform the same cleaning operations or actions over and over throughout your flow, you can copy and paste your steps, actions, or even fields. For more information see [Copy steps, actions and fields](#).

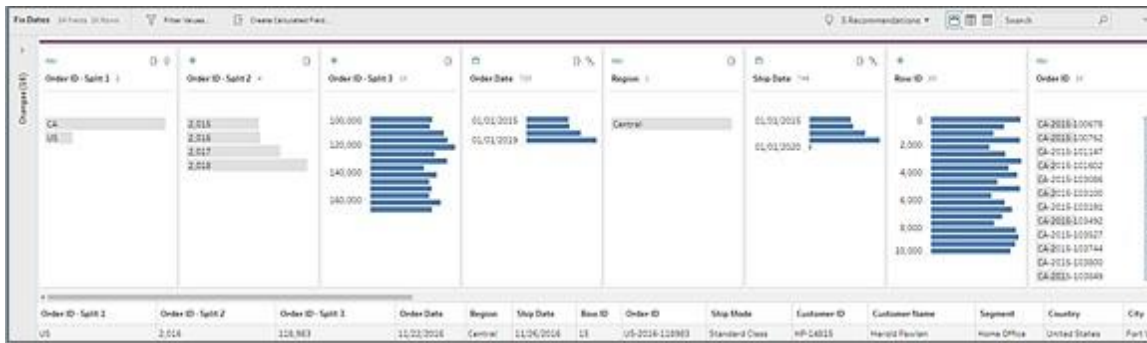
Profile pane toolbar




Select your view

You can perform cleaning operations outside of the profile or results pane in the data grid or in the list view. Use the view toolbar  (Tableau Prep Builder version 2019.3.2 and later and on the web) to change your view, then click **More options** on a field to open the cleaning menu.


- **Show profile pane:**  This is the default view. Select this button to go back to the Profile pane or Results pane view.





- **Show data grid:**  Collapse the profile or results pane to expand and show only the data grid. This view provides a more detailed view of your data and can be useful when you need to work with specific field values. After you select this option, this view state persists across all steps in your flow but you can change it at any time.

Note: Not all cleaning operations are available in the data grid. For example if you want to edit a value in-line, you must use the Profile pane.

Order ID - Split 1	Order ID - Split 2	Order ID - Split 3	Order Date	Region	Ship Date	Row ID	Order ID	Ship Mode
US	2.016	118.903	11/20/2016	Central	11/26/2016	18	US-2016-118903	Standard Class
US	2.016	118.903	11/22/2016	Central	11/26/2016	19	US-2016-118903	Standard Class
CA	2.016	106.893	11/11/2016	Central	11/26/2016	17	CA-2016-106893	Standard Class
CA	2.017	137.330	12/09/2017	Central	12/18/2017	22	CA-2017-137330	Standard Class
CA	2.017	137.330	12/09/2017	Central	12/18/2017	23	CA-2017-137330	Standard Class
CA	2.018	107.727	10/18/2018	Central	10/21/2018	35	CA-2018-107727	Second Class
CA	2.017	137.690	12/06/2017	Central	12/15/2017	36	CA-2017-137690	First Class
CA	2.017	137.690	12/06/2017	Central	12/15/2017	37	CA-2017-137690	First Class
CA	2.018	117.425	12/27/2018	Central	12/31/2018	38	CA-2018-117425	Standard Class
CA	2.016	117.425	12/27/2016	Central	12/31/2016	39	CA-2016-117425	Standard Class
CA	2.016	117.425	12/27/2016	Central	12/31/2016	40	CA-2016-117425	Standard Class
CA	2.018	117.425	12/27/2018	Central	12/31/2018	41	CA-2018-117425	Standard Class
CA	2.018	120.999	09/10/2018	Central	09/11/2018	42	CA-2018-120999	Standard Class
CA	2.017	138.255	03/11/2017	Central	03/12/2017	43	CA-2017-138255	First Class
CA	2.017	138.255	03/11/2017	Central	03/12/2017	44	CA-2017-138255	First Class

Show list view  (Tableau Prep Builder version 2019.3.2 and later and on the web): Convert the profile pane or results pane into a list. After you select this option, this view state persists across all steps in your flow but you can change it at any time.

In this view you can:

- Select and remove multiple rows using the **X** option.
- (version 2021.1.4 and later) Select and hide or unhide multiple rows using the  option.
- (version 2021.2.1 and later) Rename fields in bulk.
- Use the **More options**  menu to apply operations to selected fields.

If you assign a data role to the field, or select **Filter**, **Group Values**, **Clean**, or **Split Values**, you're returned to the Profile or Results view to complete those actions. All other options can be performed in the list view.

Fix Dates22 fields

Rename Fields...

Pivot Columns to Rows

Merge Fields

Keep Only Fields

Hide Fields

...

Search

Changes (13)

Type	Field Name	Changes
<div></div>	Order Date	<div></div>
Abc	Region	<div></div>
<div></div>	Ship Date	<div></div>
#	Row ID	
Abc	Order ID	
Abc	Ship Mode	
Abc	Customer ID	
Abc	Customer Name	
Abc	Segment	
Abc	Country	

Rename Fields...

Pivot Columns to Rows

Merge

Keep Only

Hide Fields

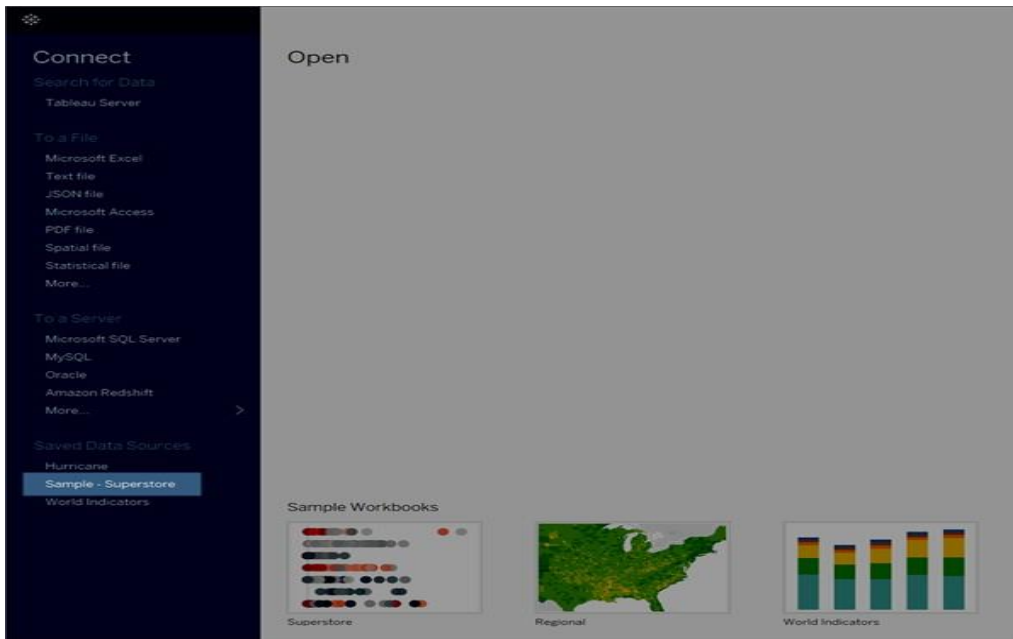
Remove

Experiment #09

AIM: Charts: Bar charts, Legends, Filters, and Hierarchies, step charts, Line charts.

To create a bar chart that displays total sales over a four-year period, follow these steps:

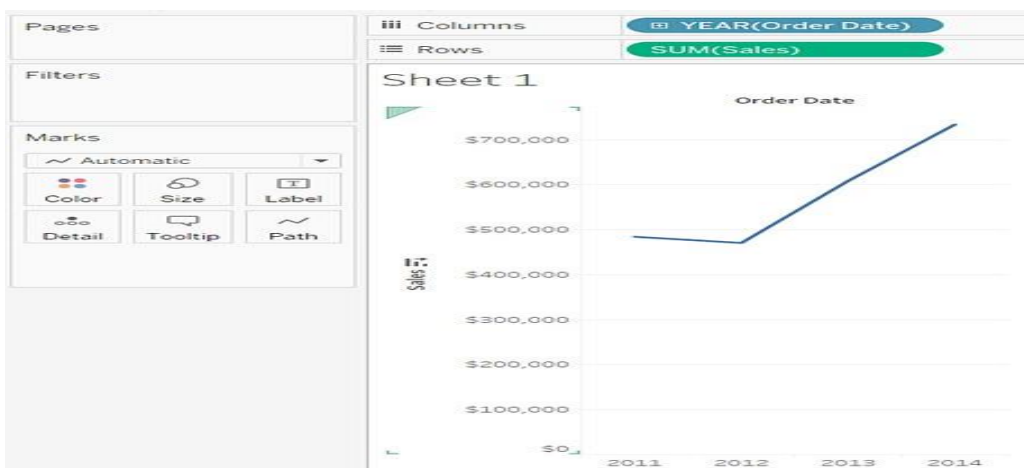
Connect to the **Sample - Superstore** data source.



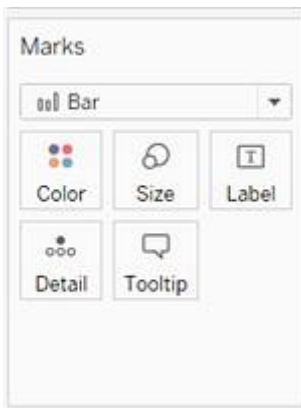
Note: In Tableau 2020.2 and later, the Data pane no longer shows Dimensions and Measures as labels. Fields are listed by table or folder.

Drag the **Order Date** dimension to **Columns** and drag the **Sales** measure to **Rows**.

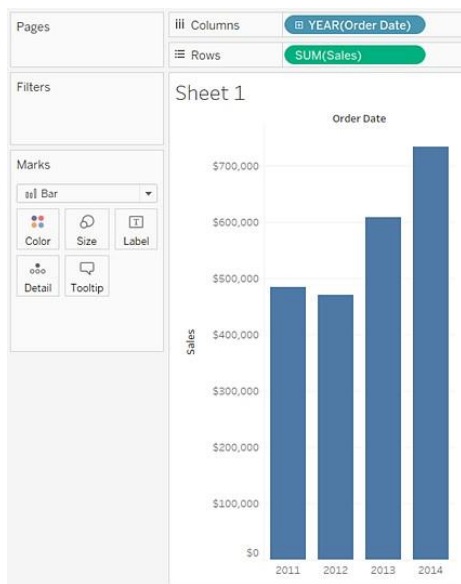
Notice that the data is aggregated by year and column headers appear. The Sales measure is aggregated as a sum and an axis is created, while the column headers move to the bottom of the view. Tableau uses **Line** as the mark type because you added the date dimension.



On the **Marks** card, select **Bar** from the drop-down list.



The view changes to a bar chart.

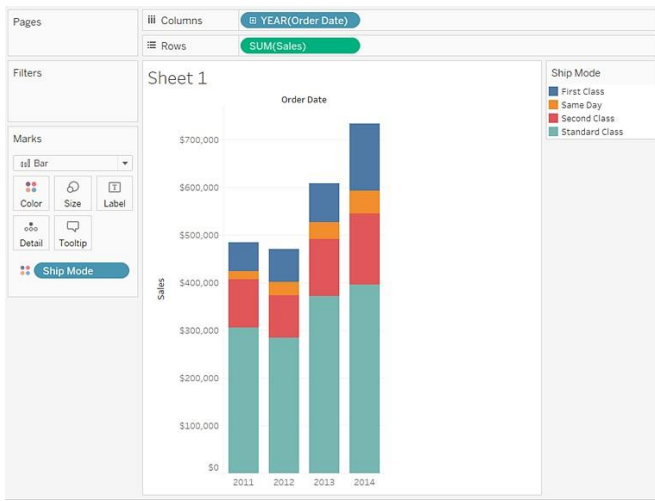


The marks (which are bars in this case) are vertical because the axis is vertical. The length of each mark represents the sum of the sales for that year. The actual numbers you see here might not match the numbers you see—the sample data changes from time to time.

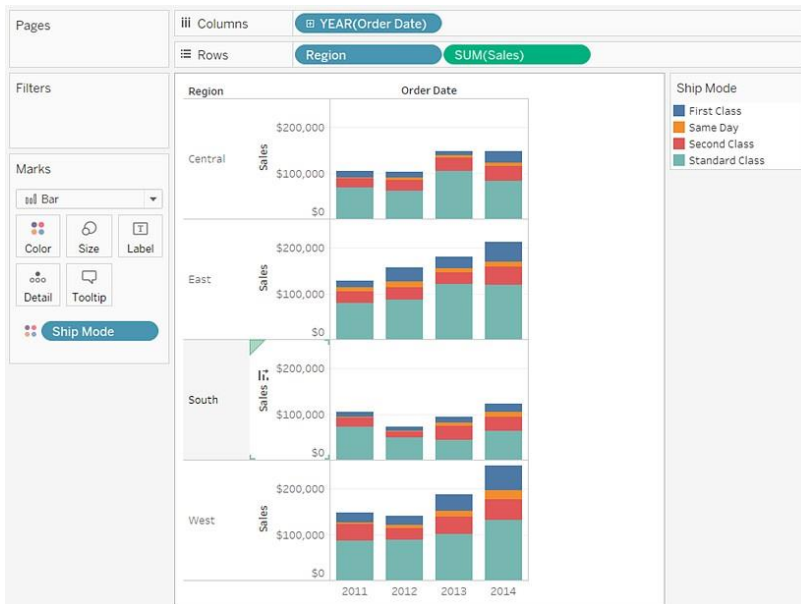


Drag the **Ship Mode** dimension to **Color** on the **Marks** card.

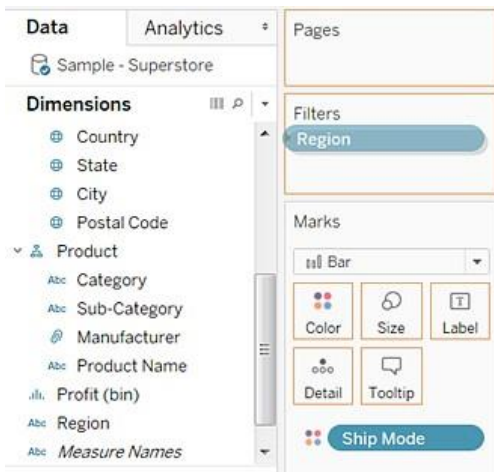
The view shows how different shipping modes have contributed to total sales over time. The ratios look consistent from year to year.



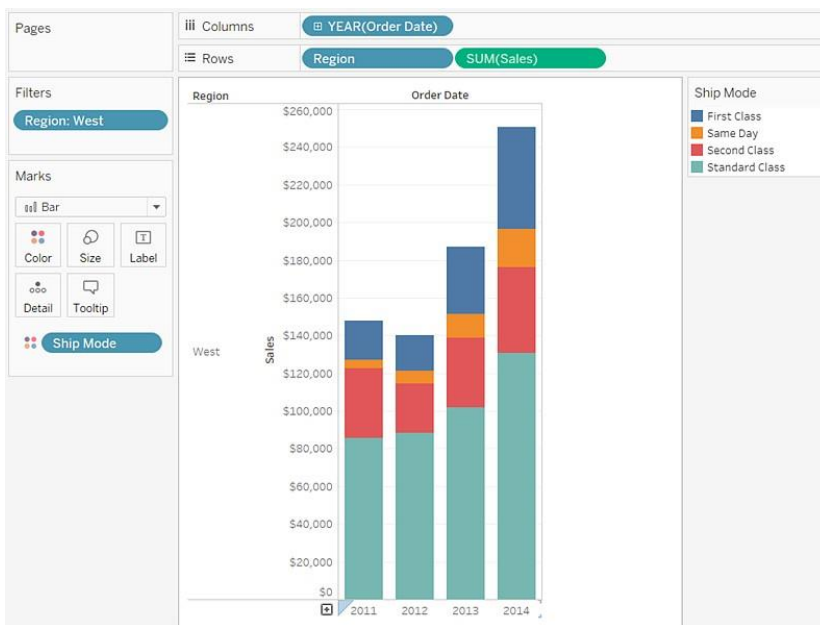
5. Drag the **Region** dimension to **Rows**, and drop it to the left of **Sales** to produce multiple axes for sales by region.



To view data in the **West** region only, you can filter out the other regions. To do this, drag the **Region** dimension again, this time from the **Data** pane to the **Filters** shelf.



In the Filter [Region] dialog box, clear the **Central**, **East**, and **South** check boxes, and then click **OK**.



This view gives you insight into your data—for example, how the ship mode changed in the West over the four-year period

Line chart

Line charts connect individual data points in a view. They provide a simple way to visualize a sequence of values and are useful when you want to see trends over time, or to forecast future values. For more information about the line mark type, see [Line mark](#).

To create a view that displays the sum of sales and the sum of profit for all years, and then uses forecasting to determine a trend, follow these steps:

1. Connect to the **Sample - Superstore** data source.

2. Drag the **Order Date** dimension to **Columns**.

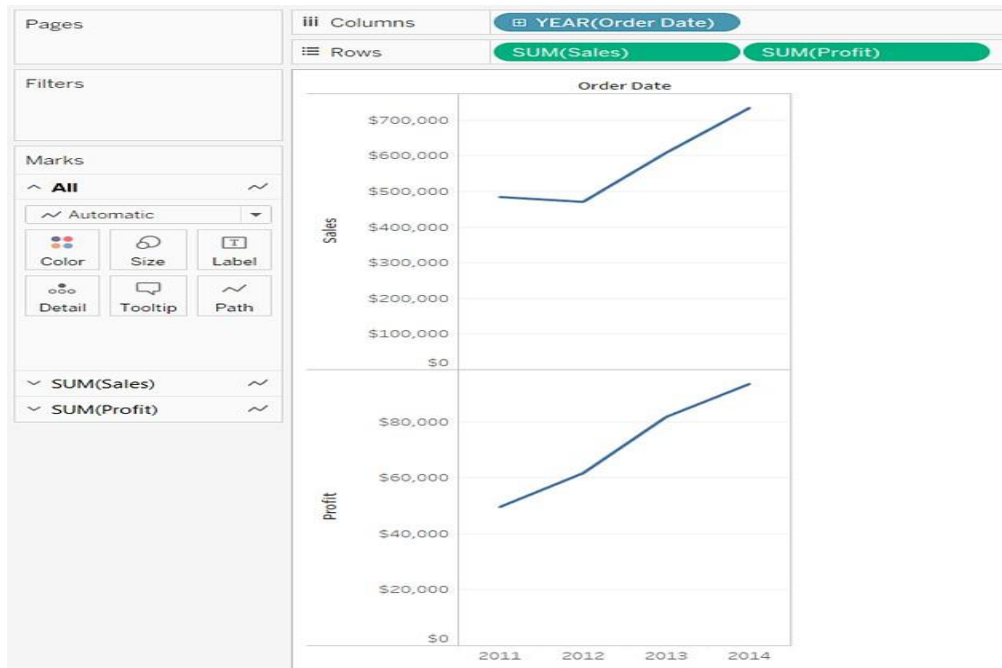
Tableau aggregates the date by year, and creates column headers.

3. Drag the **Sales** measure to **Rows**.

Tableau aggregates **Sales** as SUM and displays a simple line chart.

4. Drag the **Profit** measure to **Rows** and drop it to the right of the **Sales** measure.

Tableau creates separate axes along the left margin for **Sales** and **Profit**.



Notice that the scale of the two axes is different—the **Sales** axis scales from \$0 to \$700,000, whereas the **Profit** axis scales from \$0 to \$100,000. This can make it hard to see that sales values are much greater than profit values.

When you are displaying multiple measures in a line chart, you can align or merge axes to make it easier for users to compare values.

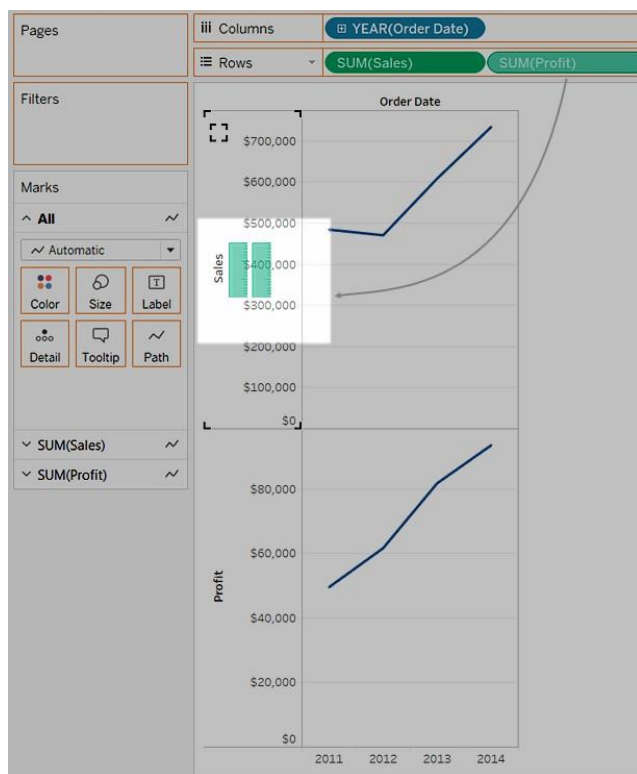
For more information about aligning the axes, see [Compare two measures using dual axes](#).

For more information about enforcing a single axis across multiple measures, see [Blend axes for multiple measures into a single axis](#).

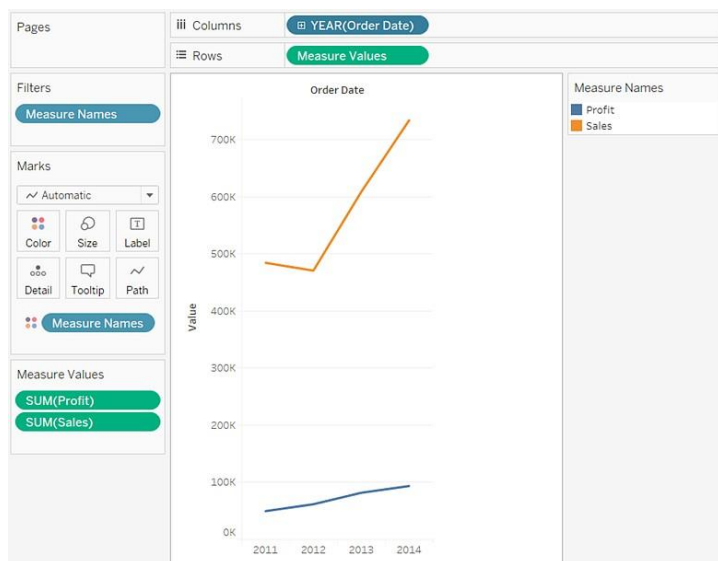
With either of these options, you can create a combination chart to change the mark type for one of your measures.

For more information, see [Create a combo chart \(assign different mark types to measures\)](#).

5. Drag the **SUM(Profit)** field from **Rows** to the **Sales** axis to create a blended axis. The two pale green parallel bars indicate that **Profit** and **Sales** will use a blended axis when you release the mouse button.

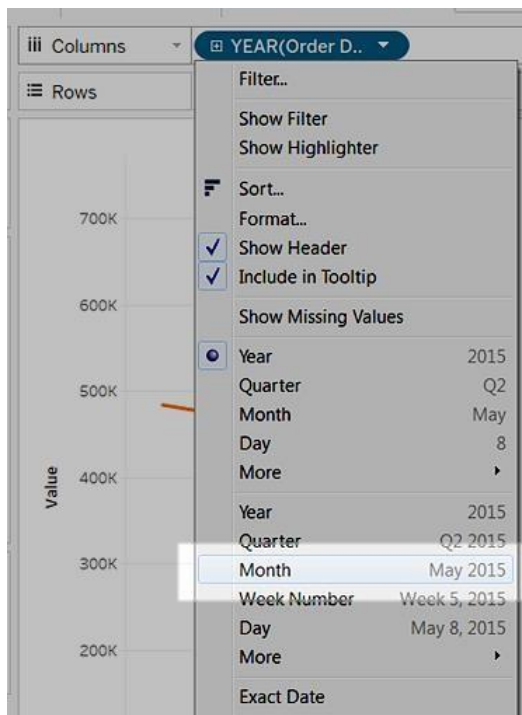


The view updates to look like this:

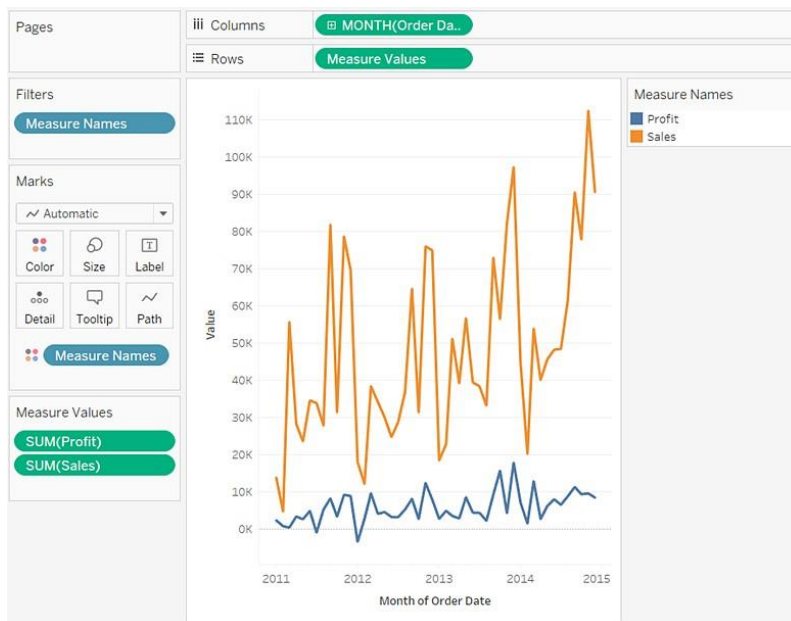


The view is rather sparse because we are looking at a summation of values on a per-year basis.

Click the drop-down arrow in the **Year(Order Date)** field on the **Columns** shelf and select **Month** in the lower part of the context menu to see a continuous range of values over the four-year period.

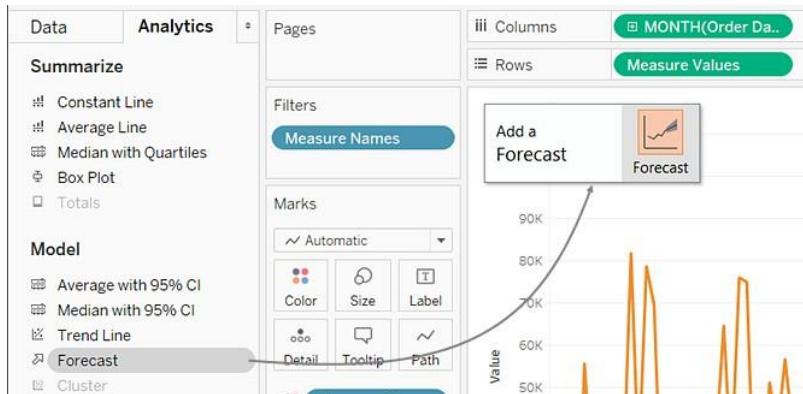


The resulting view is a lot more detailed than the original view:

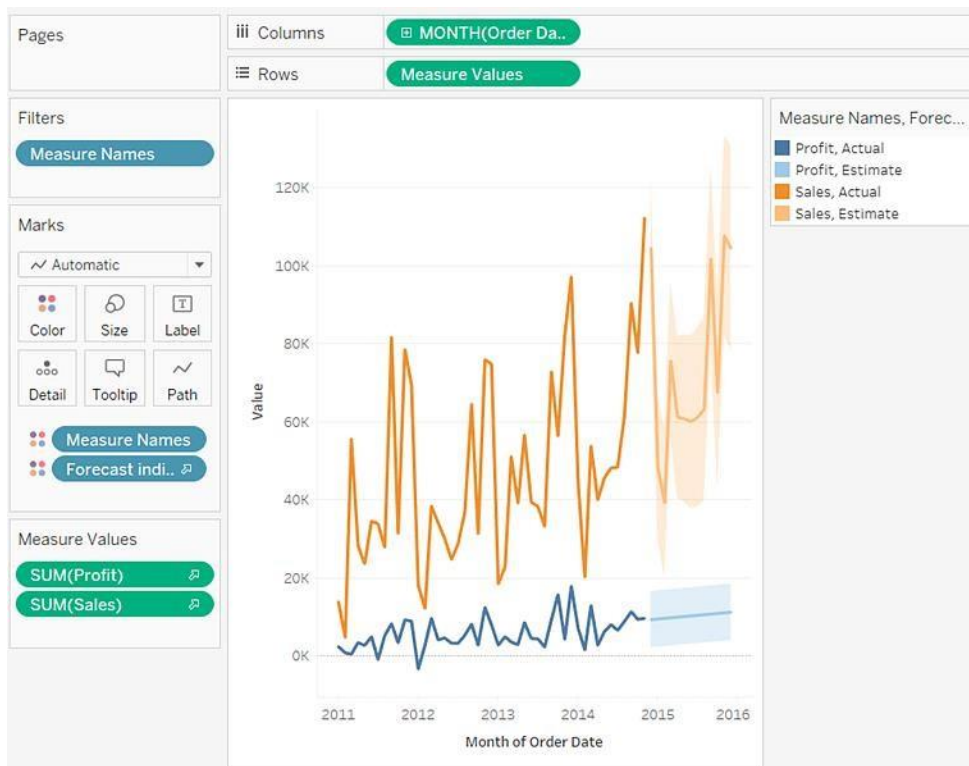


Notice that the values seem to go much higher just before the end of each year. A pattern like that is known as *seasonality*. If we turn on the forecasting feature in the view, we can see whether we should expect that the apparent seasonal trend will continue in the future.

To add a forecast, in the **Analytics** pane, drag the **Forecast** model to the view, and then drop it on **Forecast**.



We then see that, according to Tableau forecasting, the seasonal trend does continue into the future:



Experiment #10

AIM: Maps: Symbol maps , Filled Maps ,Density maps ,Maps with Pie charts.

Maps are one of the best ways to represent the data in Tableau. Data visualizations using maps look very attractive and appealing. Map charts suites better when you want to show the demographic data such as Population census, income, housing, household rates, etc. Maps give an easy-to-infer advantage over the other charts available in Tableau and are highly interactive when you want to plot demographic or geographical data on a plot.

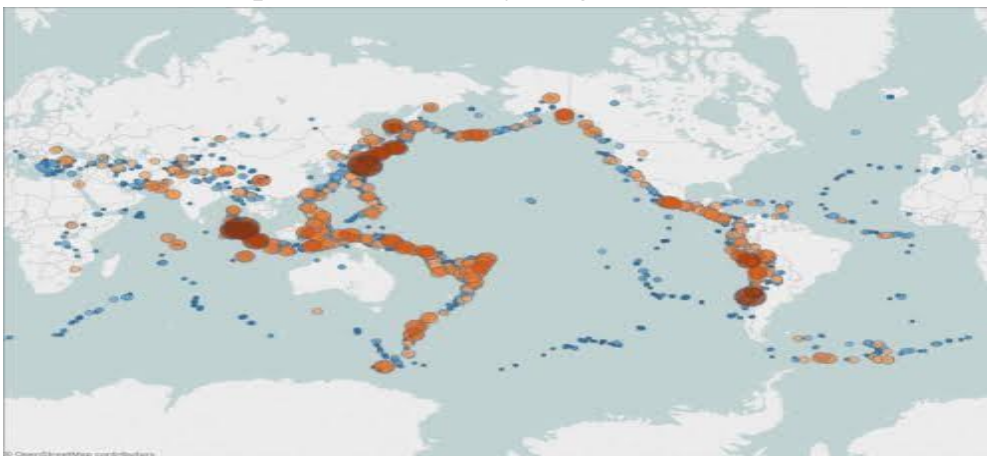
Here in tableau, we have different types of maps, which are:

- Tableau Proportional symbol maps
- Tableau Choropleth maps (filled maps)
- Tableau Point distribution maps
- Tableau Heatmaps (density maps)
- Tableau Flow maps (path maps)
- Tableau Spider maps (origin-destination maps)

Let's discuss each map in detail one by one.

Proportional symbol maps:

Proportional Symbol maps are suitable for visualizing the quantitative data for individual locations. It shows the data of one or two quantitative values per location. (Among two values, one value is encoded with color and the other is with size). Let's take an example of the Earthquakes that occurred between 1981 to 2014 around the Earth, and size them by magnitude. Tableau allows you to color the data point locations by magnitude for extra details.

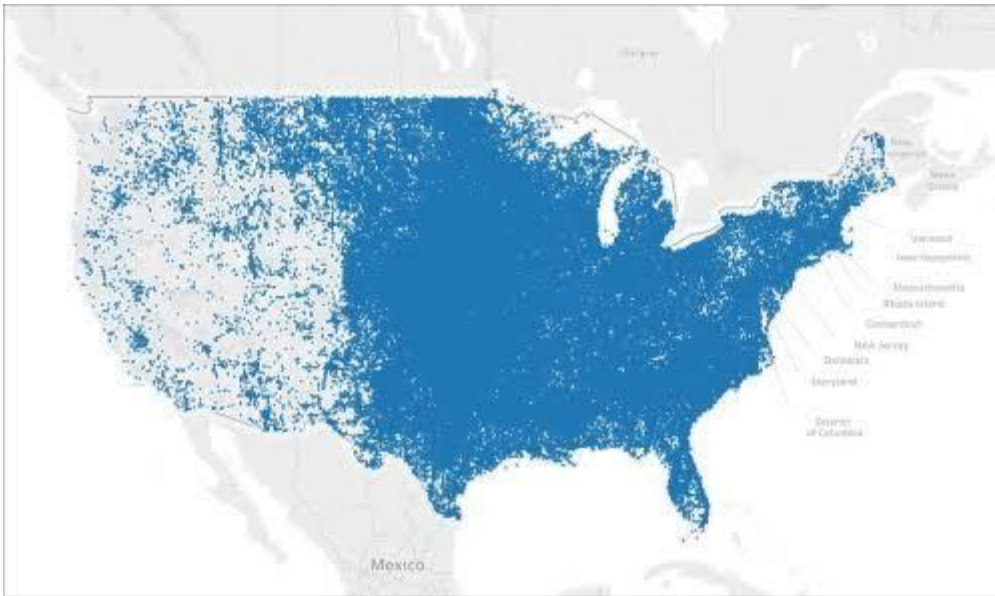


Choropleth maps (filled maps):



Choropleth maps are also called Filled maps in Tableau. These maps are best suitable for visualizing the ratio data or statistical data in the form of various shading patterns or symbols on specific geographic areas. Choropleth maps are best suitable for visualizing data for polygons. And the Polygons may be countries, states, regions, or any area that can be geocoded in Tableau.

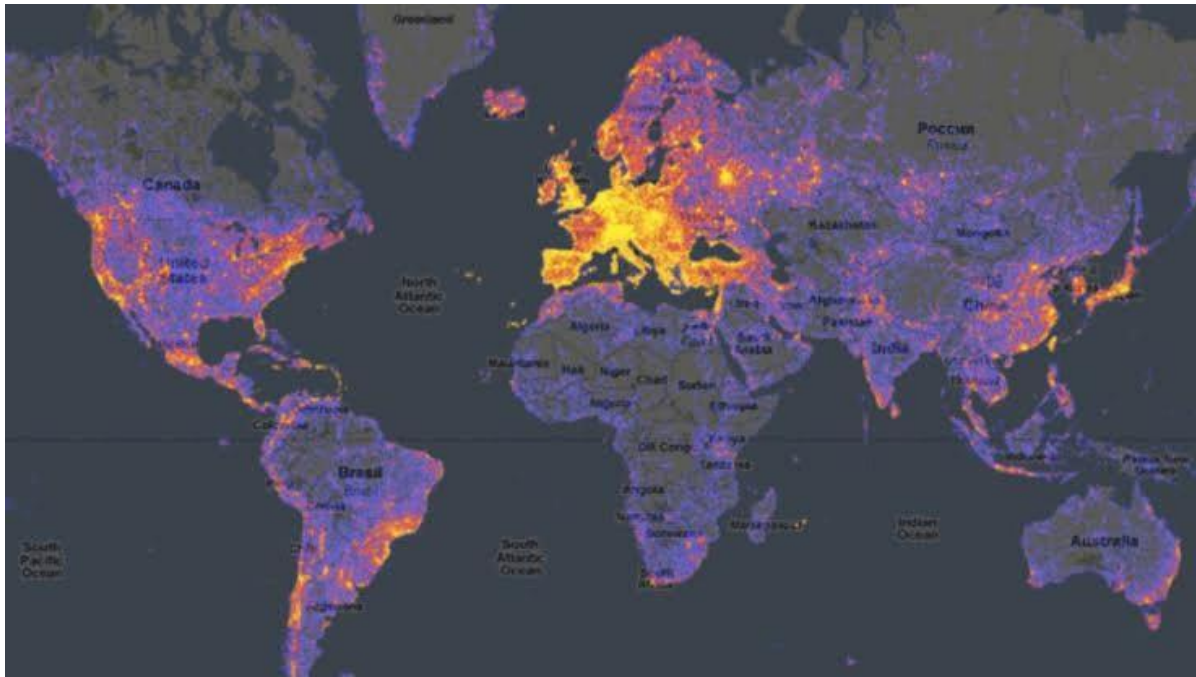
Point distribution maps:



Point distribution maps are those which help the organizations in visualizing how locations of particular data points are shared. Point distribution maps are often used to show approximate locations and are looking for visualized clusters of data. To create a data distribution map in Tableau, your data source should include Latitude and longitude coordinates.

Let us consider an example here, if you wish to find the hail storms that occurred in the U.S last year, you can use a point distribution map to see if you can spot any clusters.

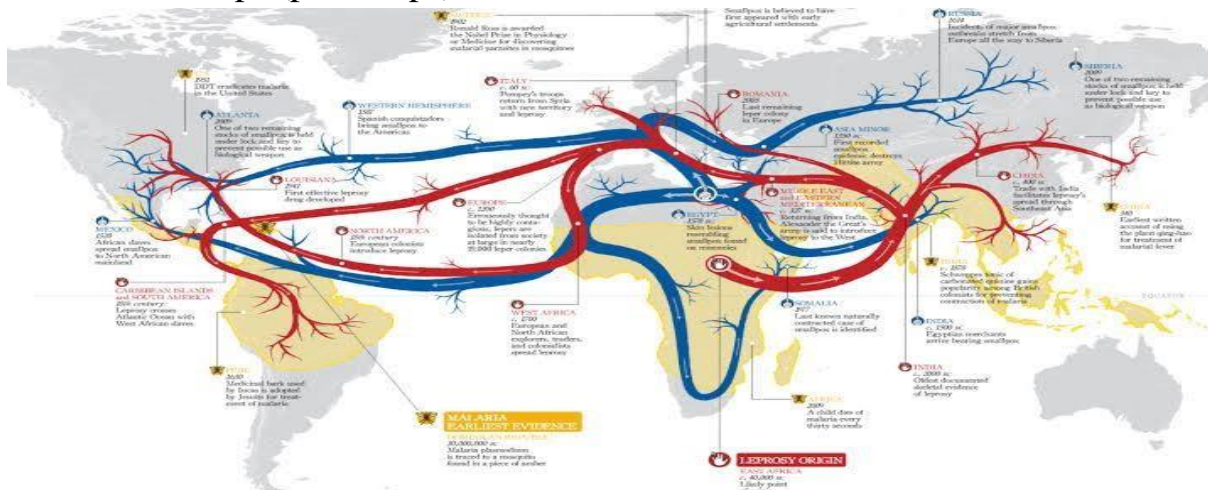
Heatmap (density maps):



Heat maps are also called density maps in tableau. Heat maps work efficiently to make the large volumes of data to be made comprehensible. Heat maps are one of the great ways to spot trends and help in what to do next. Heat maps are good at self-explanatory, and intuitive compared to other maps.

If you wish to find user engagement on a website, the Heat map will help you with that. For example, you would like to find which area or content on a page has got more & fewer clicks on a site, then apply a heat map which highlights the area where users have clicked a maximum number of times with black color and give light color where users have given no or less attention.

Flow maps (path maps):



Flow maps are also called path maps in tableau. These maps are often used in Tableau to show a Path over time. Flow suites perfectly to show the journey of a particular thing from one place to another over a period of time, as the path of a storm.

Experiment #11

AIM: Interactive Dashboards.

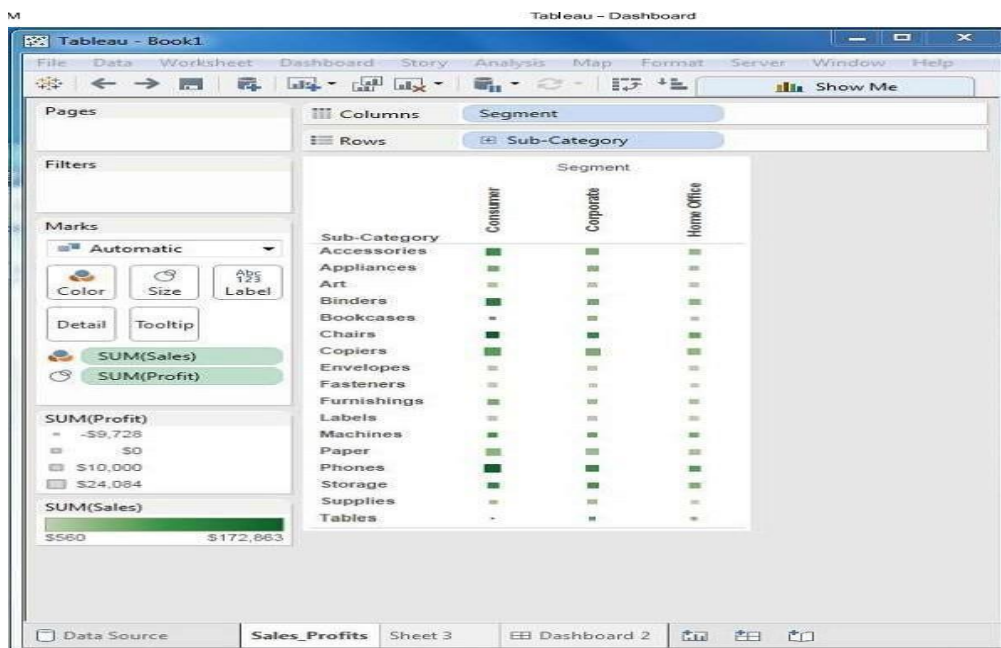
A dashboard is a consolidated display of many worksheets and related information in a single place. It is used to compare and monitor a variety of data simultaneously. The different data views are displayed all at once. Dashboards are shown as tabs at the bottom of the workbook and they usually get updated with the most recent data from the data source. While creating a dashboard, you can add views from any worksheet in the workbook along with many supporting objects such as text areas, web pages, and images.

Each view you add to the dashboard is connected to its corresponding worksheet. So when you modify the worksheet, the dashboard is updated and when you modify the view in the dashboard, the worksheet is updated.

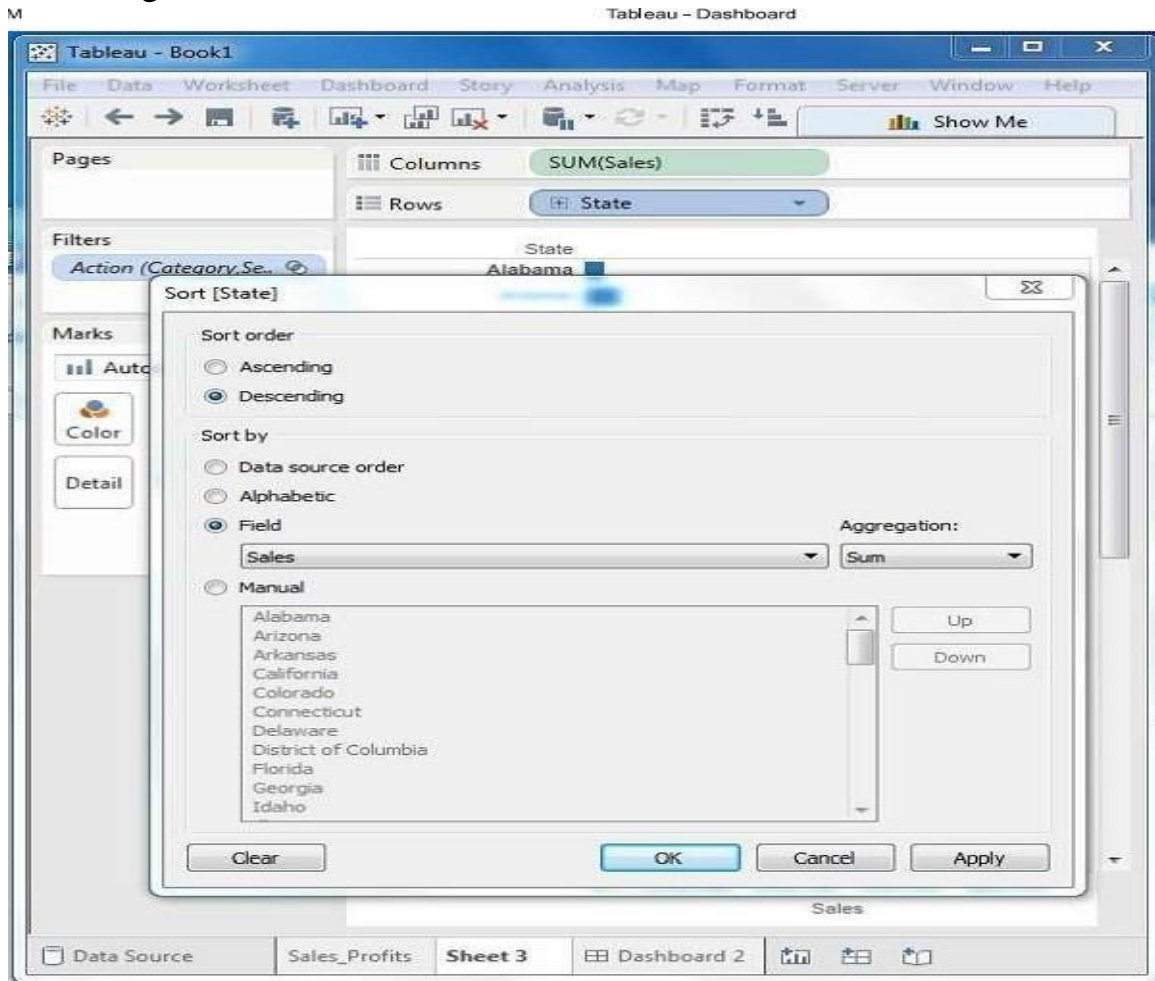
Creating a Dashboard

Using the Sample-superstore, plan to create a dashboard showing the sales and profits for different segments and Sub-Category of products across all the states. To achieve this objective, following are the steps.

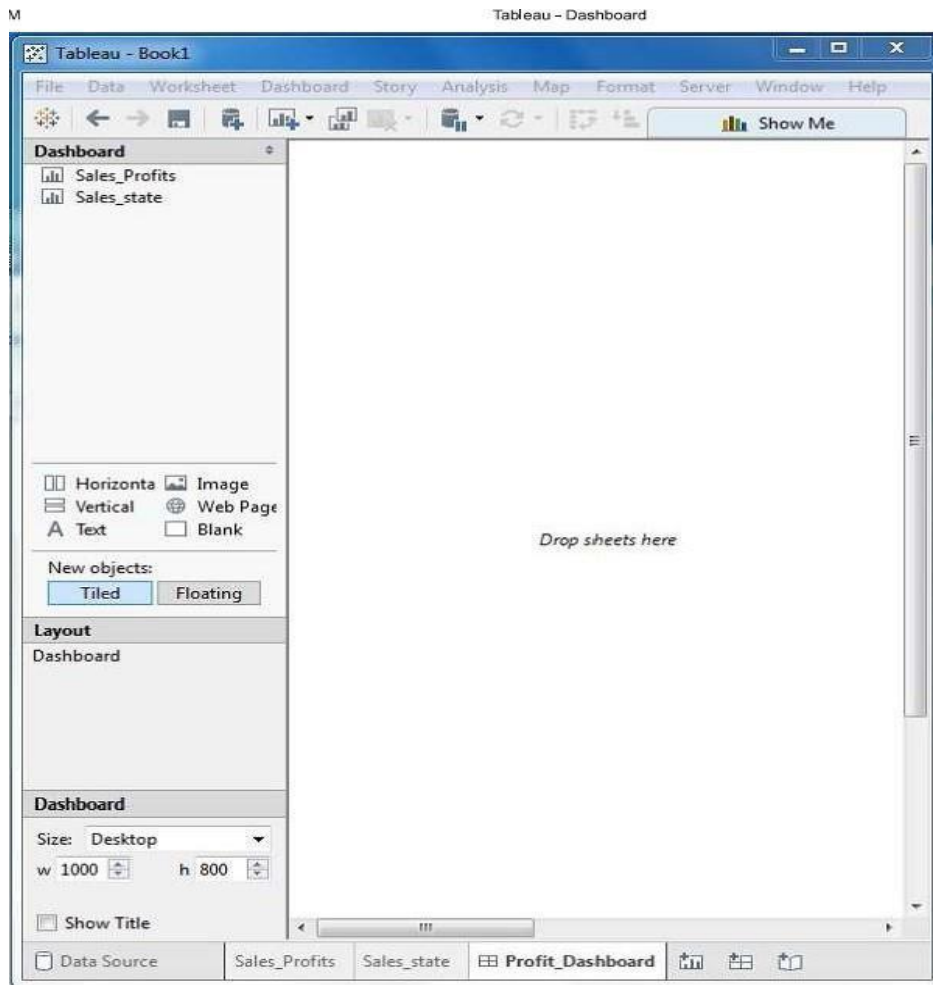
Step 1 - Create a blank worksheet by using the add worksheet icon located at the bottom of the workbook. Drag the dimension Segment to the columns shelf and the dimension Sub-Category to the Rows Shelf. Drag and drop the measure Sales to the Color shelf and the measure Profit to the Size shelf. This worksheet is referred as the Master worksheet. Right-click and rename the worksheet as Sales_Profits. The following chart appears.



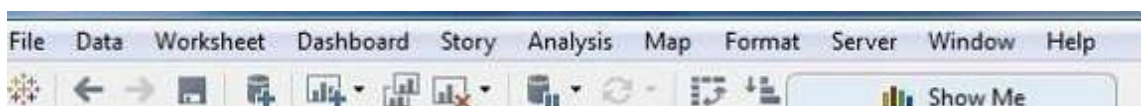
Step 2 - Create another sheet to hold the details of the Sales across the States. For this, drag the dimension State to the Rows shelf and the measure Sales to the Columns shelf as shown in the following screenshot. Next, apply a filter to the State field to arrange the Sales in a descending order. Right-dick and rename this worksheet as Sales_state



Step 3 - Next, create a blank dashboard by clicking the Create New Dashboard link at the bottom of the workbook. Right-click and rename the dashboard as Profit_Dashboard



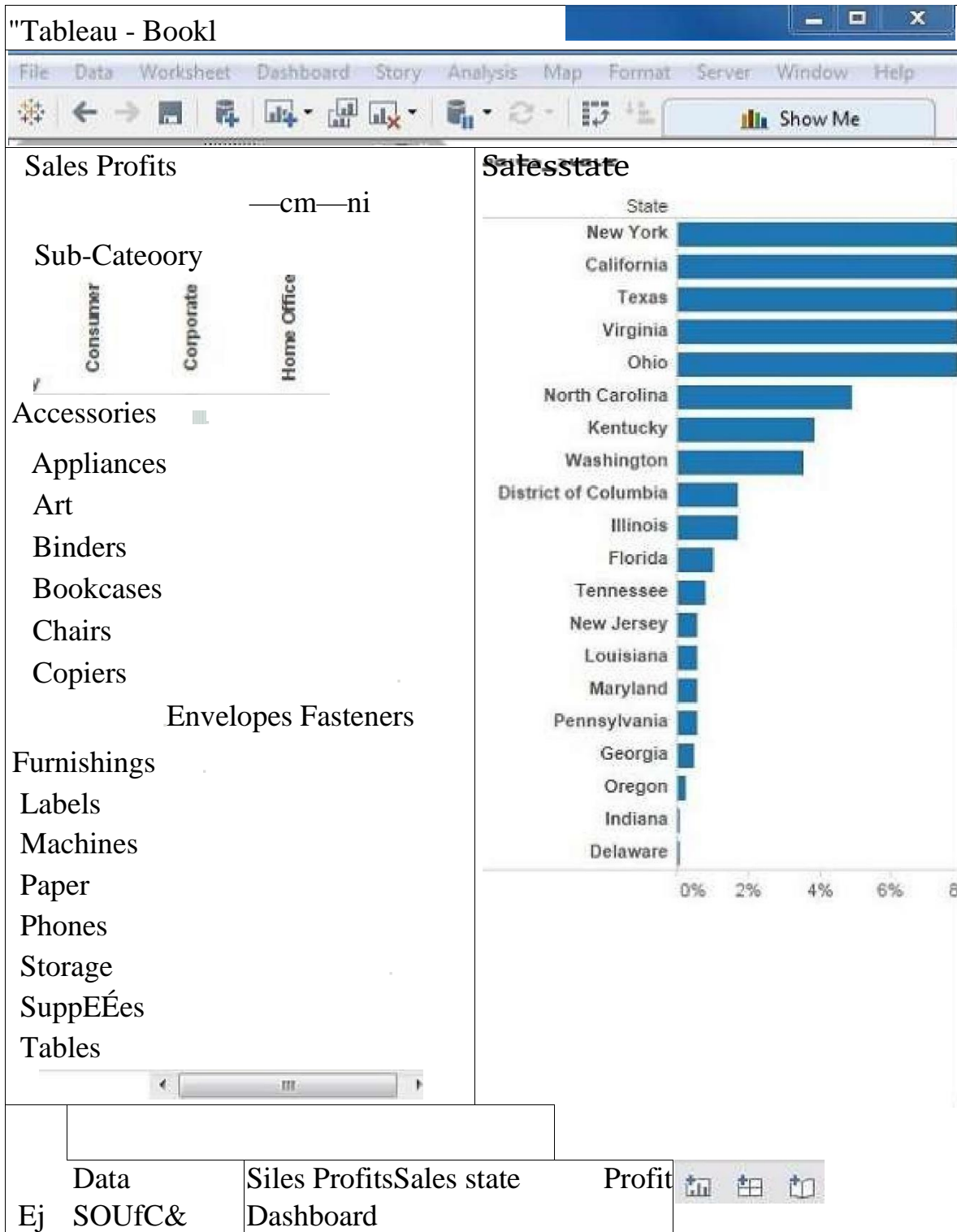
Step 4 - Drag the two worksheets to the dashboard. Near the top border line of Sales Profit worksheet, you can see three small icons. Click the middle one, which shows the prompt Use as Filter on hovering the mouse over it.





Step 5: Now in the dashboard click the box representing Sub-Category named **Machines** and segment named **Consumer**.

You can notice that only the states where the sales happened for this amount of profit are filtered out in the right pane named **Sales_state**. This illustrates how the sheets are linked in a dashboard



GOOD LUCK
THANK YOU

Prepared By:
Mr. P. Nagababu
Dept of CSE
LBRCE