Dr. D. Y. Patil Pratishthan's

DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH

Approved by A.I.C.T.E, New Delhi , Maharashtra State Government, Affiliated to Savitribai Phule Pune University Sector No. 29, PCNTDA , Nigidi Pradhikaran, Akurdi, Pune411044. Phone: 020–27654470, Fax: 020- 27656566 Website :www.dypiemr.ac.in Email : principal.dypiemr@gmail.com

Department of Artificial Intelligence and Data Science

LAB MANUAL

Computer Laboratory IV
Semester-VIII

Prepared By:
Dr. Suvarna Patil

Computer Laboratory -IV

Course Code	Course Name	Teaching Scheme (Hrs./ Week)	Credits
417535	Computer Laboratory-IV: Big Data Analytics	2	2

Course Objectives:

- To understand the fundamental concepts and techniques of Virtual reality
- To understand Big Data Analytics Concepts
- To learn the fundamentals of software development for portable devices
- To understand fundamental concepts of Deep Learning
- To be familiar with the various application areas of augmented realities
- To introduce the concepts and components of Business Intelligence (BI)
- To understand the concepts of Information Systems

Course Outcomes:

After completion of the course, learners should be able to-

CO1: Apply basic principles of elective subjects to problem solving and modeling

CO2: Use tools and techniques in area of software development to build mini projects

CO3: Design and develop applications on subjects of their choice

CO4: Implement and manage deployment, administration & security

Operating System recommended: 64-bit Open source Linux or its derivative

Table of Contents

Lab Assignment No.	1
Title	Set up and Configuration Hadoop Using CloudEra/ Google Cloud BigQuery. Databricks Lakehouse Platform. Snowflake. Amazon Redshift
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

EXPERIMENT NO. 1 A (Group A)

Aim: Set up and Configuration Hadoop Using CloudEra/ Google Cloud BigQuery. Databricks Lakehouse Platform. Snowflake. Amazon Redshift

Theory:

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:

we'll install Hadoop in stand-alone mode and run one of the example example MapReduce programs it includes to verify the installation.

Prerequisites:

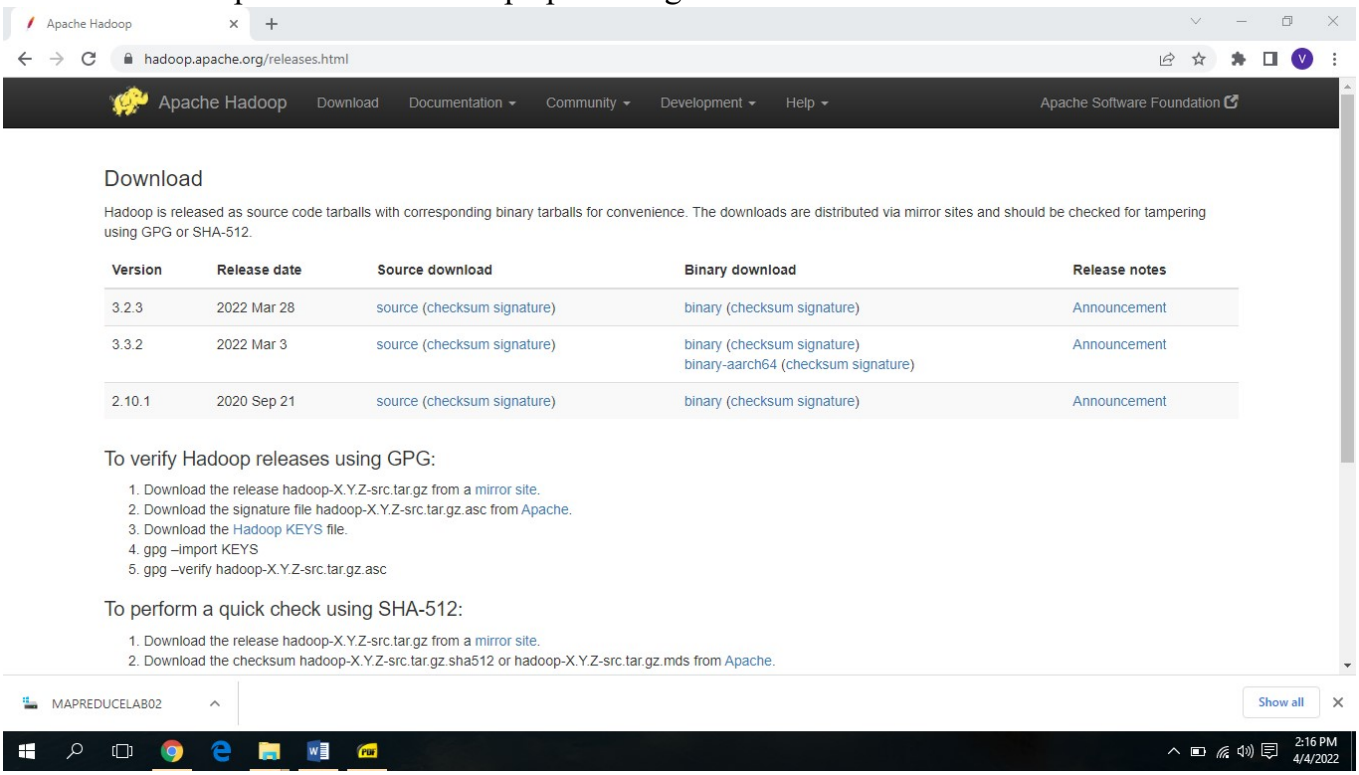
Step1: Installing Java 8 version.

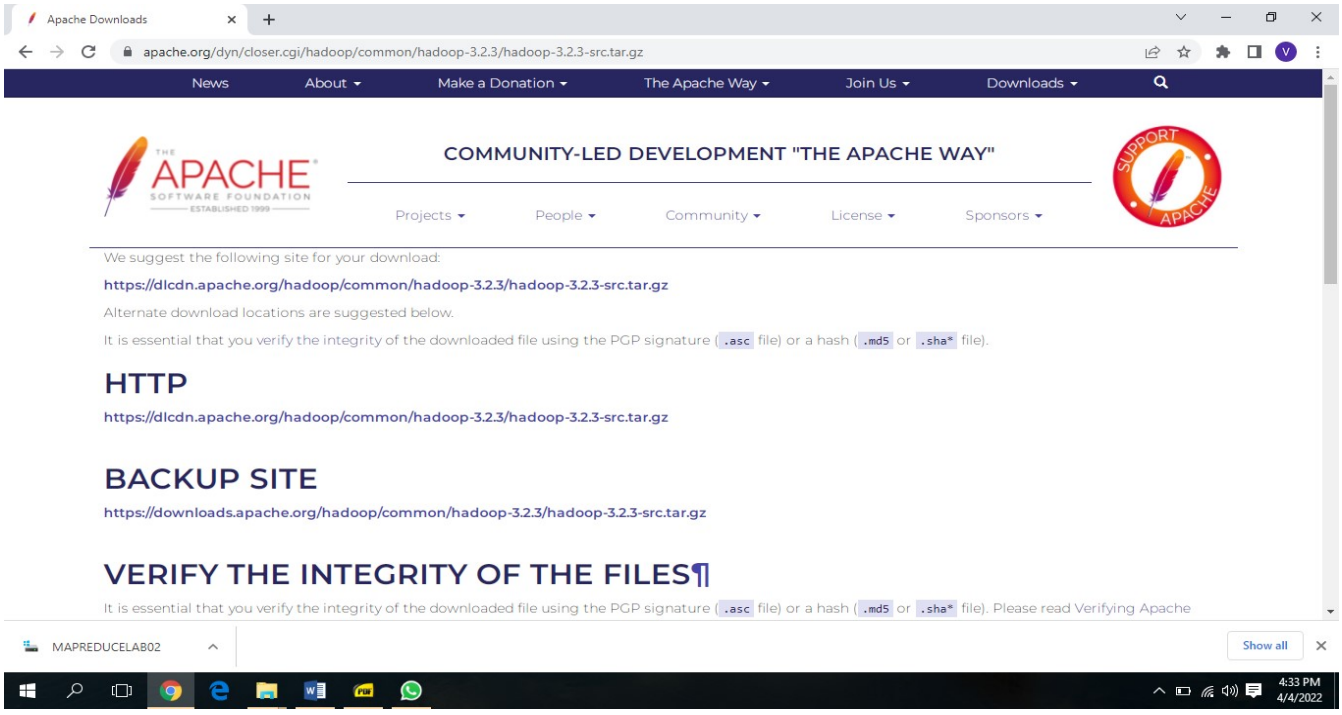
Openjdk version "1.8.0_91"
OpenJDK Runtime Environment (build 1.8.0_91-8u91-b14-3ubuntu1~16.04.1-b14)
OpenJDK 64-Bit Server VM (build 25.91-b14, mixed mode) This output verifies that OpenJDK has been successfully installed. **Note:** To set the path for environment variables. i.e. JAVA_HOME

Step2: Installing Hadoop

With Java in place, we'll visit the Apache Hadoop Releases page to find the most recent stable release. Follow the binary for the current release:

Download Hadoop from www.hadoop.apache.org





Procedure to Run Hadoop

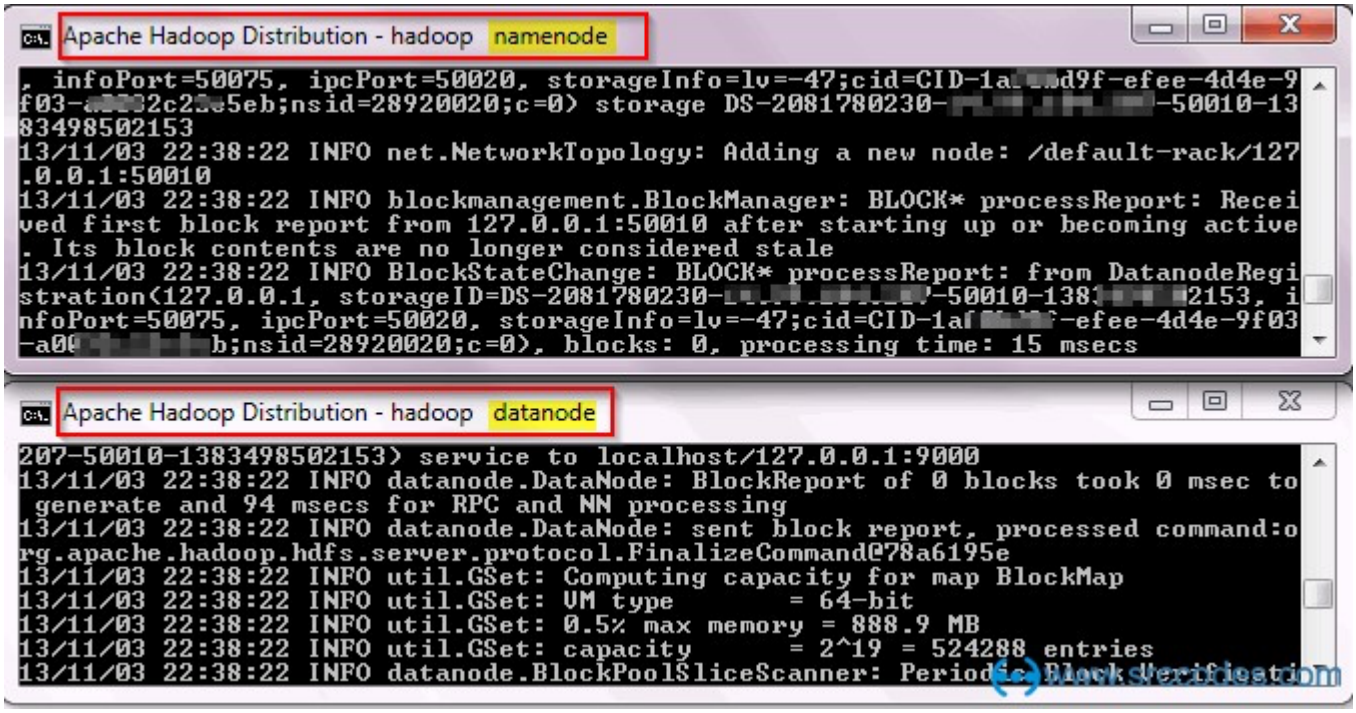
1. Install Apache Hadoop 2.2.0 in Microsoft Windows OS

If Apache Hadoop 2.2.0 is not already installed then follow the post Build, Install, Configure and Run Apache Hadoop 2.2.0 in Microsoft Windows OS.

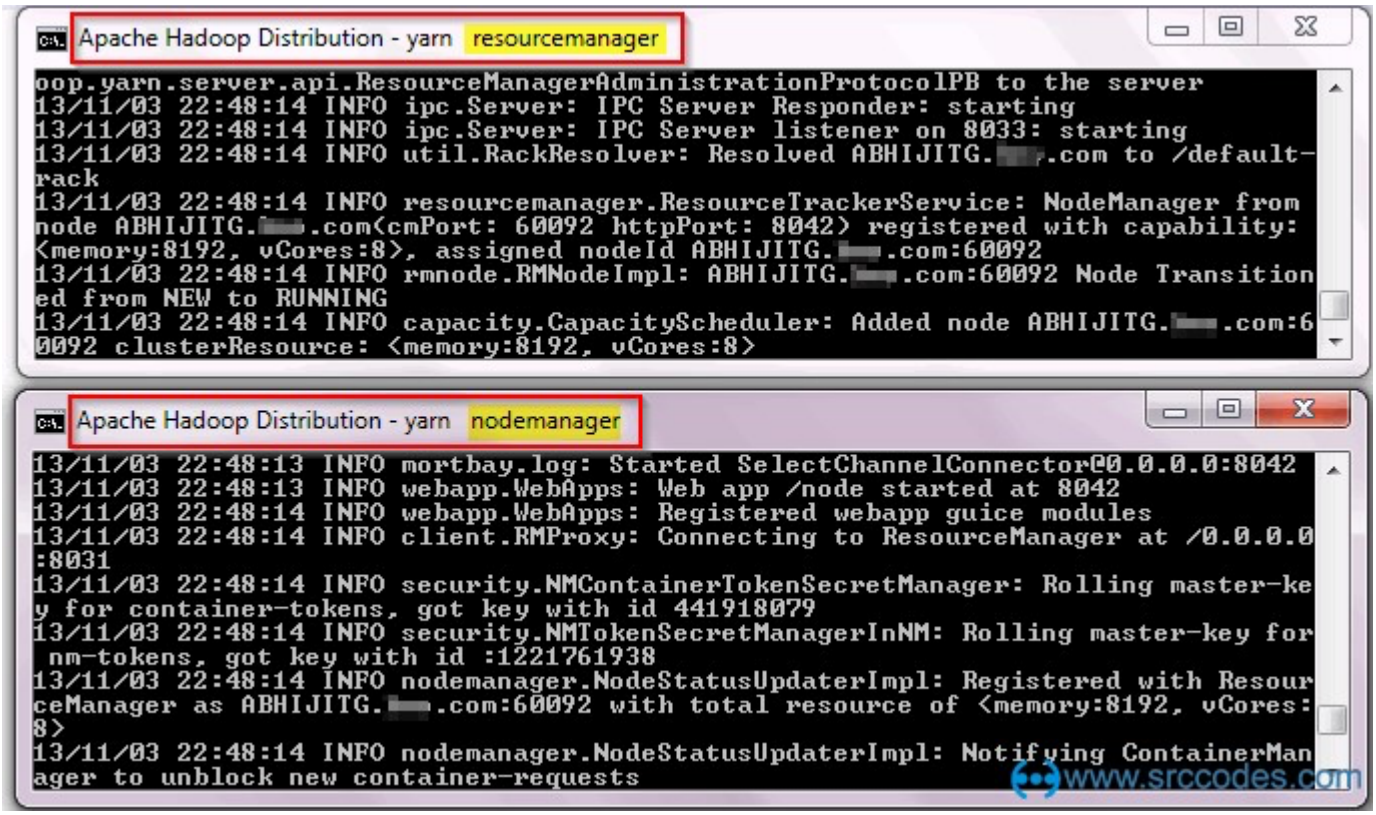
2. Start HDFS (Namenode and Datanode) and YARN (Resource Manager and Node Manager)

Run following commands.
Command Prompt
C:\Users\abhijitg>cd c:\hadoop
c:\hadoop>sbin\start-dfs
c:\hadoop>sbin\start-yarn starting yarn
daemons

Namenode, **Datanode**, **Resource Manager** and **Node Manager** will be started in few minutes and ready to execute Hadoop **MapReduce** job in the Single Node (pseudo-distributed mode) cluster.



Resource Manager & Node Manager:



Run wordcount MapReduce job

Now we'll run **wordcount** MapReduce job available in **%HADOOP_HOME%\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.2.0.jar**

Create a text file with some content. We'll pass this file as input to the **wordcount** MapReduce job for counting words.
C:\file1.txt

Install Hadoop

Run Hadoop Wordcount Mapreduce Example

Create a directory (say 'input') in HDFS to keep all the text files (say 'file1.txt') to be used for counting words.

C:\Users\abhijitg>cd c:\hadoop C:\hadoop>bin\hdfs dfs -mkdir input

Copy the text file(say 'file1.txt') from local disk to the newly created 'input' directory in HDFS.

C:\hadoop>bin\hdfs dfs -copyFromLocal c:/file1.txt input

Check content of the copied file.

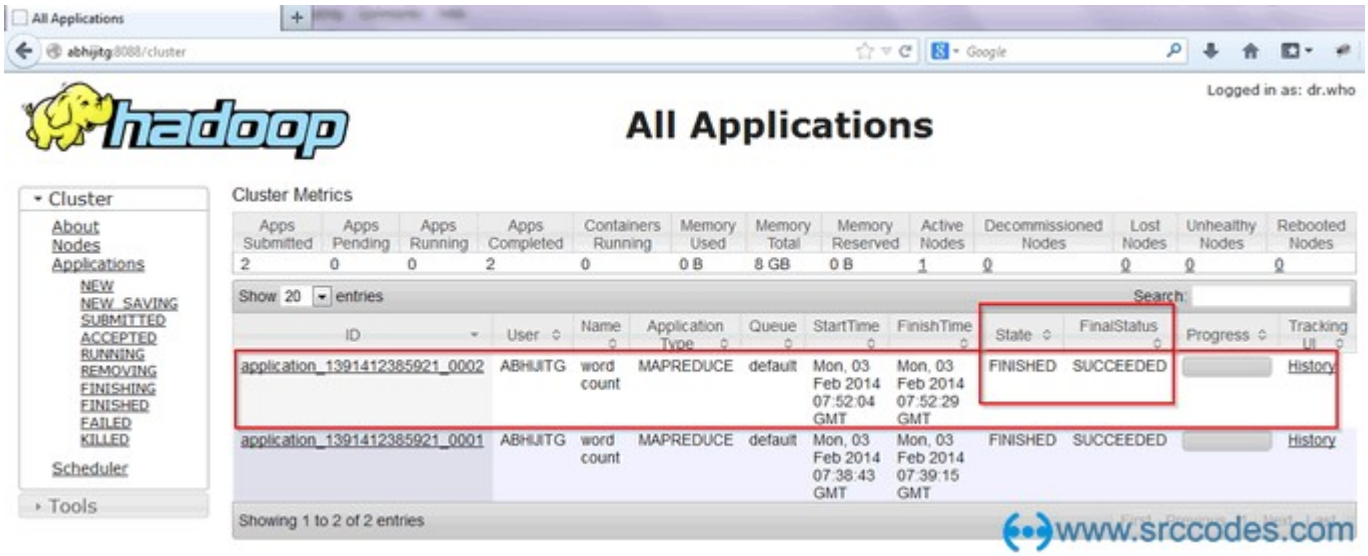
C:\hadoop>hdfs dfs -ls input
Found 1 items
-rw-r--r-- 1 ABHIJITG supergroup 55 2014-02-03 13:19 input/file1.txt

```
C:\hadoop>bin\hdfs dfs -cat input/file1.txt
Install Hadoop
Run Hadoop Wordcount Mapreduce Example
```

Run the wordcount MapReduce job provided
in %HADOOP_HOME%\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.2.0.jar

```
C:\hadoop>bin\yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-
2.2.0.jar wordcount input output
14/02/03 13:22:02 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
14/02/03 13:22:03 INFO input.FileInputFormat: Total input paths to process : 1 14/02/03
13:22:03 INFO mapreduce.JobSubmitter: number of splits:1
: :
14/02/03 13:22:04 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1391412385921_0002
14/02/03 13:22:04 INFO impl.YarnClientImpl: Submitted application
application_1391412385921_0002 to ResourceManager at /0.0.0.0:8032 14/02/03 13:22:04
INFO mapreduce.Job: The url to track the job:
http://ABHIJITG:8088/proxy/application_1391412385921_0002/
14/02/03 13:22:04 INFO mapreduce.Job: Running job: job_1391412385921_0002 14/02/03
13:22:14 INFO mapreduce.Job: Job job_1391412385921_0002 running in uber mode : false
14/02/03 13:22:14 INFO mapreduce.Job:  map 0% reduce 0%
14/02/03 13:22:22 INFO mapreduce.Job:  map 100% reduce 0%
14/02/03 13:22:30 INFO mapreduce.Job:  map 100% reduce 100%
14/02/03 13:22:30 INFO mapreduce.Job: Job job_1391412385921_0002 completed successfully
14/02/03 13:22:31 INFO mapreduce.Job: Counters: 43
        File System Counters
                FILE: Number of bytes read=89
                FILE: Number of bytes written=160142
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=171
                HDFS: Number of bytes written=59
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=5657
```

```
        Total time spent by all reduces in occupied slots (ms)=6128
Map-Reduce Framework
        Map input records=2
        Map output records=7
        Map output bytes=82
        Map output materialized bytes=89
        Input split bytes=116
        Combine input records=7
        Combine output records=6
        Reduce input groups=6
        Reduce shuffle bytes=89
        Reduce input records=6
        Reduce output records=6
        Spilled Records=12
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=145
        CPU time spent (ms)=1418
        Physical memory (bytes) snapshot=368246784
        Virtual memory (bytes) snapshot=513716224
        Total committed heap usage (bytes)=307757056
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0          WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=55
File Output Format Counters
        Bytes Written=59
```



Result: We've installed Hadoop in stand-alone mode and verified it by running an example program it provided.

Lab Assignment No.	2
Title	Develop a MapReduce program to calculate the frequency of a given word in a given file.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

AIM: To Develop a MapReduce program to calculate the frequency of a given word in agiven file

Map Function – It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

Example – (Map function in Word Count)

Input

Set of data

Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN

Output

Convert into another set of data

(Key,Value)

(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1),

(TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)

Reduce Function – Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

Example – (Reduce function in Word Count)

Input Set of Tuples

(output of Map function)

(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1),

(TRAIN,1),(BUS,1),

(buS,1),(caR,1),(CAR,1), (car,1), (BUS,1), (TRAIN,1)

Output **Converts into smaller set of tuples**

(BUS,7), (CAR,7), (TRAIN,4)

Work Flow of Program

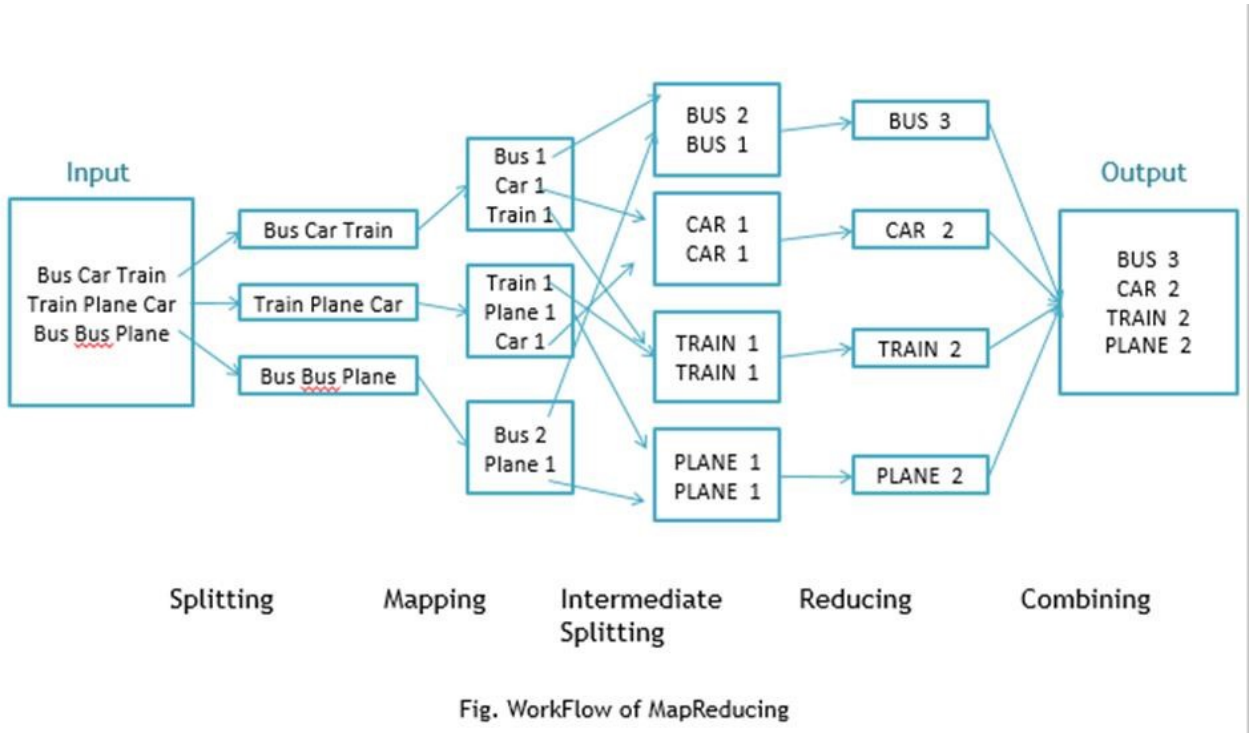


Fig. WorkFlow of MapReducing

Workflow of MapReduce consists of 5 steps

1. **Splitting –** The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line (‘\n’).
2. **Mapping –** as explained above
3. Intermediate splitting – the entire process in parallel on different clusters. In order to group them in “Reduce Phase” the similar KEY data should be on same cluster.
4. **Reduce –** it is nothing but mostly group by phase
5. **Combining –** The last phase where all the data (individual result set from each cluster) is combine together to form a Result

Now Let’s See the Word Count Program in Java

Make sure that Hadoop is installed on your system with java idk

Steps to follow

Step 1. Open Eclipse> File > New > Java Project > (Name it – MRProgramsDemo) > Finish Step 2. Right Click > New > Package (Name it - PackageDemo) > Finish Step 3. Right Click on Package > New > Class (Name it - WordCount) Step 4. Add Following Reference Libraries – Right Click on Project > Build Path> Add External Archivals

- /usr/lib/hadoop-0.20/hadoop-core.jar
- Usr/lib/hadoop-0.20/lib/Commons-cli-1.2.jar

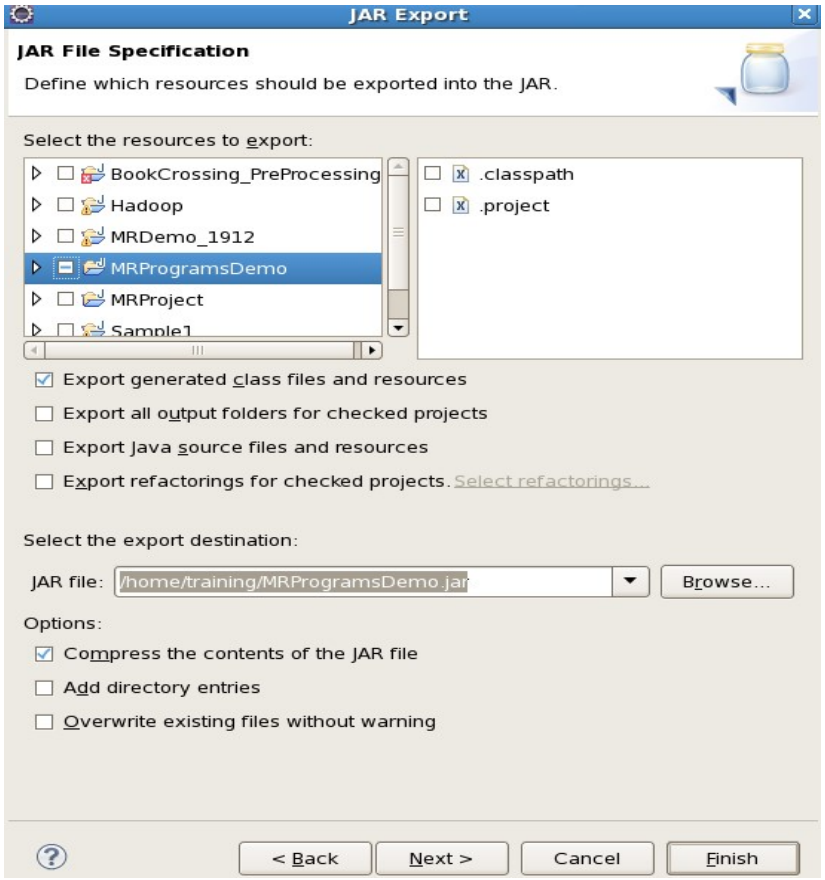
Program: Step 5. Type following Program :

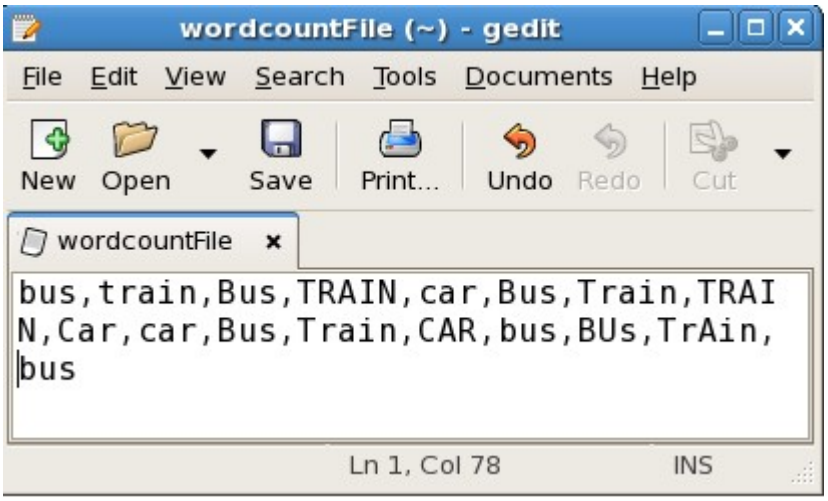
```
package PackageDemo; import
java.io.IOException;
import org.apache.hadoop.conf.Configuration; import
org.apache.hadoop.fs.Path; import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable; import
org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
org.apache.hadoop.util.GenericOptionsParser; public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new Configuration();
String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
Path input=new Path(files[0]);
Path output=new Path(files[1]);
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.class);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class); FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
public static class MapForWordCount extends Mapper<LongWritable, Text, Text,
IntWritable>{
public void map(LongWritable  key, Text  value, Context  con) throws IOException,
InterruptedException
{
String line = value.toString();
String[] words=line.split(","); for(String
word: words )
{
   Text outputKey = new Text(word.toUpperCase().trim());     IntWritable
outputValue = new IntWritable(1);     con.write(outputKey, outputValue);
}
} }
```

```
public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text,
IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con) throws
IOException,
InterruptedException
{ int sum = 0;
for(IntWritable value : values)
{
sum += value.get();
}
con.write(word, new IntWritable(sum));
}
}
}
```

Make Jar File

Right Click on Project> Export> Select export destination as Jar File > next> Finish





To Move this into Hadoop directly, open the terminal and enter the following commands:
[training@localhost ~]$ hadoop fs -put wordcountFile wordCountFile

Run Jar file
(Hadoop jar jarfilename.jar packageName.ClassName PathToInputTextFile PathToOutputDirectry)

[training@localhost ~]$ Hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile MRDir1

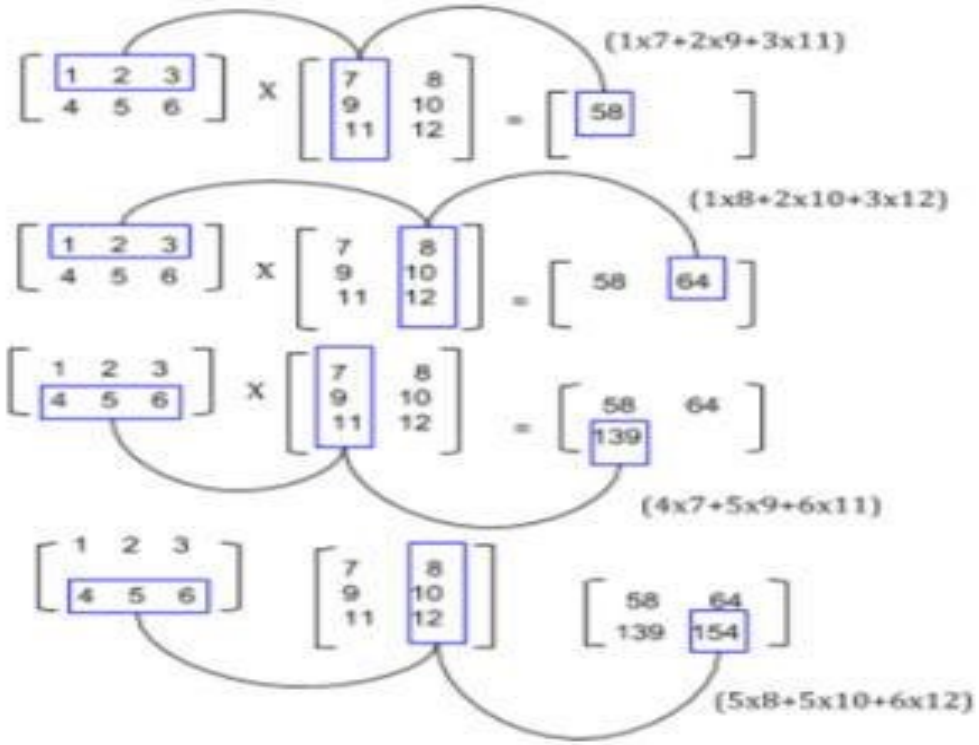
Result: Open Result

[training@localhost ~]$ hadoop fs -ls MRDir1
Found 3 items
-rw-r--r-- 1 training supergroup
0 2016-02-23 03:36 /user/training/MRDir1/_SUCCESS
drwxr-xr-x - training supergroup
0 2016-02-23 03:36 /user/training/MRDir1/_logs
-rw-r--r-- 1 training supergroup
20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000
[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000
BUS 7
CAR 4
TRAIN 6

Lab Assignment No.	3
Title	Implement Matrix Multiplication using Map-Reduce
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

AIM: To Develop a MapReduce program to implement Matrix Multiplication.

In **mathematics**, **matrix multiplication** or the **matrix product** is a binary operation that produces a matrix from two matrices. The definition is motivated by linear equations and linear transformations on vectors, which have numerous applications in applied mathematics, physics, and engineering. In more detail, if **A** is an $n \times m$ matrix and **B** is an $m \times p$ matrix, their matrix product **AB** is an $n \times p$ matrix, in which the m entries across a row of **A** are multiplied with the m entries down a column of **B** and summed to produce an entry of **AB**. When two linear transformations are represented by matrices, then the matrix product represents the composition of the two transformations.



Algorithm for Map Function.

a. for each element m_{ij} of M do
produce (key,value) pairs as ((i,k), (M,j,m_{ij}), for k=1,2,3,.. upto the number of columns of N

b. for each element njk of N do produce (key,value) pairs as ((i,k),(N,j,N_{jk}), for i = 1,2,3,.. Upto the number of rows of M.

c. return Set of (key,value) pairs that each key (i,k), has list with values (M,j,m_{ij}) and (N, j,n_{jk}) for all possible values of j.

Algorithm for Reduce Function.

d. for each key (i,k) do
e. sort values begin with M by j in listM sort values begin with N by j in listN multiply mij and njk for jth value of each list
f. sum up mij x njk return (i,k), Σj=1 mij x njk

Step 1. Download the hadoop jar files with these links.

Download Hadoop Common Jar files: https://goo.gl/G4MyHp
$ wget https://goo.gl/G4MyHp -O hadoop-common-2.2.0.jar
Download Hadoop Mapreduce Jar File: https://goo.gl/KT8yfB
$ wget https://goo.gl/KT8yfB -O hadoop-mapreduce-client-core-2.7.1.jar

Step 2. Creating Mapper file for Matrix Multiplication.

```
import java.io.DataInput; import
java.io.DataOutput; import
java.io.IOException;
import java.util.ArrayList;

import org.apache.hadoop.conf.Configuration; import
org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.DoubleWritable; import
org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text; import
org.apache.hadoop.io.Writable; import
org.apache.hadoop.io.WritableComparable; import
org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer; import
org.apache.hadoop.mapreduce.lib.input.*; import
org.apache.hadoop.mapreduce.lib.output.*;
import org.apache.hadoop.util.ReflectionUtils;

class Element implements Writable {
      int tag;          int index;       double
value;        Element() {
 tag = 0;   index = 0;
                                value = 0.0;
              }
```

```
                    Element(int tag, int index, double value) {
  this.tag = tag;   this.index = index;
                    this.value = value;
            }
            @Override
      public void readFields(DataInput input) throws IOException {
      tag = input.readInt();          index = input.readInt();
                                      value = input.readDouble();
            }
            @Override
      public void write(DataOutput output) throws IOException {
            output.writeInt(tag);           output.writeInt(index);
                                      output.writeDouble(value);
            }
 }
 class Pair implements WritableComparable<Pair> {
      int i;    int j;

      Pair() {
i = 0;
j = 0;
 }
 Pair(int i, int j) {
            this.i = i;
      this.j = j;
            }
            @Override
      public void readFields(DataInput input) throws IOException {
      i = input.readInt();            j = input.readInt();
            }
            @Override
      public void write(DataOutput output) throws IOException {
            output.writeInt(i);             output.writeInt(j);
            }
            @Override
                  public int compareTo(Pair compare) {
                    if (i > compare.i) {
                    return 1;
                          } else if ( i < compare.i) {
                  return -1;              }
 else {                   if(j > compare.j)
 {
```

```
                                        return 1;
                        } else if (j < compare.j) {
                return -1;
                                }
                        }
                        return 0;
                }
                public String toString() {
                                return i + " " + j + " ";
                }
}
public class Multiply {
public static class MatriceMapperM extends Mapper<Object,Text,IntWritable,Element>
{
@Override
public void map(Object key, Text value, Context context)
  throws IOException, InterruptedException {  String readLine = value.toString();
 String[] stringTokens = readLine.split(",");

 int index = Integer.parseInt(stringTokens[0]);
 double elementValue = Double.parseDouble(stringTokens[2]);
 Element e = new Element(0, index, elementValue);
 IntWritable keyValue = new
IntWritable(Integer.parseInt(stringTokens[1]));
 context.write(keyValue, e);
                        }
                }
 public static class MatriceMapperN extends Mapper<Object,Text,IntWritable,Element> {
                                @Override
 public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
                                                String readLine = value.toString();
                        String[] stringTokens = readLine.split(",");
      int index = Integer.parseInt(stringTokens[1]);
double elementValue = Double.parseDouble(stringTokens[2]);
Element e = new Element(1,index, elementValue);
 IntWritable keyValue = new
IntWritable(Integer.parseInt(stringTokens[0]));
                  context.write(keyValue, e);
                        }
                }
 public static class ReducerMxN extends Reducer<IntWritable,Element, Pair,
DoubleWritable> {
```

```
        @Override
 public void reduce(IntWritable key, Iterable<Element> values, Context context) throws
IOException, InterruptedException {
                                        ArrayList<Element> M = new ArrayList<Element>();
        ArrayList<Element> N = new ArrayList<Element>();
    Configuration conf = context.getConfiguration();
                                    for(Element element : values) {
        Element tempElement = ReflectionUtils.newInstance(Element.class, conf);
                                ReflectionUtils.copy(conf, element, tempElement);

          if (tempElement.tag == 0) {
                M.add(tempElement);
                                        } else if(tempElement.tag == 1) {
                                        N.add(tempElement);
                                }
                        }
                                    for(int i=0;i<M.size();i++) {
                                        for(int j=0;j<N.size();j++) {

     Pair p = new Pair(M.get(i).index,N.get(j).index);
double multiplyOutput = M.get(i).value * N.get(j).value;

                        context.write(p, new DoubleWritable(multiplyOutput));
                                    }
                                }
                        }
        }
                                        public static class MapMxN extends Mapper<Object, Text, Pair,
DoubleWritable> {
                                @Override
                                                public void map(Object key, Text value, Context context)
                                                        throws IOException, InterruptedException {
                                        String readLine = value.toString();
                                        String[] pairValue = readLine.split(" ");
                                    Pair p = new
Pair(Integer.parseInt(pairValue[0]),Integer.parseInt(pairValue[1]));
                                        DoubleWritable val = new
DoubleWritable(Double.parseDouble(pairValue[2]));
                                    context.write(p, val);
                        }
        }
 public static class ReduceMxN extends Reducer<Pair, DoubleWritable, Pair,
```

```
DoubleWritable> {
@Override
  public void reduce(Pair key, Iterable<DoubleWritable> values, Context context) throws
IOException, InterruptedException {
                                double sum = 0.0;
                                        for(DoubleWritable value : values) {
                        sum += value.get();
                  }
                  context.write(key, new DoubleWritable(sum));
}
                }
      public static void main(String[] args) throws Exception {
Job job = Job.getInstance();
      job.setJobName("MapIntermediate");
      job.setJarByClass(Project1.class);
MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,
MatriceMapperM.class);
MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,
MatriceMapperN.class);
                job.setReducerClass(ReducerMxN.class);
      job.setMapOutputKeyClass(IntWritable.class);
      job.setMapOutputValueClass(Element.class);
      job.setOutputKeyClass(Pair.class);
      job.setOutputValueClass(DoubleWritable.class);
      job.setOutputFormatClass(TextOutputFormat.class);
      FileOutputFormat.setOutputPath(job, new Path(args[2]));
                job.waitForCompletion(true);
      Job job2 = Job.getInstance();
      job2.setJobName("MapFinalOutput");
        job2.setJarByClass(Project1.class);

        job2.setMapperClass(MapMxN.class);
        job2.setReducerClass(ReduceMxN.class);

                job2.setMapOutputKeyClass(Pair.class);
      job2.setMapOutputValueClass(DoubleWritable.class);

                job2.setOutputKeyClass(Pair.class);
      job2.setOutputValueClass(DoubleWritable.class);

        job2.setInputFormatClass(TextInputFormat.class);
        job2.setOutputFormatClass(TextOutputFormat.class);
```

```
        FileInputFormat.setInputPaths(job2, new Path(args[2]));
      FileOutputFormat.setOutputPath(job2, new Path(args[3])); job2.waitForCompletion(true); }
}
```

Step 5. Compiling the program in particular folder named as operation

```
#!/bin/bash

rm -rf multiply.jar classes

module load hadoop/2.6.0

mkdir -p classes javac -d classes -cp classes:`$HADOOP_HOME/bin/hadoop classpath`
Multiply.java jar cf multiply.jar -C classes .

echo "end"
```

Step 6. Running the program in particular folder named as operation

```
export HADOOP_CONF_DIR=/home/$USER/cometcluster module
load hadoop/2.6.0
myhadoop-configure.sh
start-dfs.sh start-yarn.sh

hdfs dfs -mkdir -p /user/$USER hdfs dfs -put M-matrix-large.txt
/user/$USER/M-matrix-large.txt hdfs dfs -put N-matrix-large.txt
/user/$USER/N-matrix-large.txt
hadoop jar multiply.jar edu.uta.cse6331.Multiply /user/$USER/M-matrix-
large.txt
/user/$USER/N-matrix-large.txt /user/$USER/intermediate /user/$USER/output rm
-rf output-distr mkdir output-distr
hdfs dfs -get /user/$USER/output/part* output-distr

stop-yarn.sh
stop-dfs.sh myhadoop-cleanup.sh
```

Expected Output:

```
module load hadoop/2.6.0 rm -rf

output intermediate

hadoop --config $HOME jar multiply.jar edu.uta.cse6331.Multiply M-matrix-small.txt N-matrixsmall.txt
intermediate output
```

Lab Assignment No.	4
Title	Develop a MapReduce program to find the grades of students.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

AIM: To Develop a MapReduce program to find the grades of student's**.**

```
import java.util.Scanner; public
class JavaExample
{ public static void main(String args[])
{
/* This program assumes that the student has 6 subjects,  thats why I
     have created the array of size 6. You can * change this as per the
                                                          requirement. */
int marks[] = new int[6];
int i;
float total=0, avg;
Scanner scanner = new Scanner(System.in);
for(i=0; i<6; i++) {
System.out.print("Enter Marks of Subject"+(i+1)+":"); marks[i] =
scanner.nextInt(); total = total + marks[i];
}
scanner.close(); //Calculating
average here avg = total/6;
System.out.print("The student Grade is: ");
if(avg>=80)
{
System.out.print("A");
}
else if(avg>=60 && avg<80)
{
System.out.print("B");
}
else if(avg>=40 && avg<60)
{
                System.out.print("C");
}
    else {

                System.out.print("D");

}

}

}
```

Expected Output:

Enter Marks of Subject1:40
Enter Marks of Subject2:80
Enter Marks of Subject3:80
Enter Marks of Subject4:40
Enter Marks of Subject5:60
Enter Marks of Subject6:60
The student Grade is: B

Lab Assignment No.	5
Title	Develop a MapReduce program to analyze Titanic ship data and to find the average age of the people (only male) who died in the tragedy. How many persons are dead in each class (only female).
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

AIM: Develop a MapReduce program to analyze Titanic ship data and to find the average age of the people (both male and female) who died in the tragedy. How many persons are survived in each class.

The titanic data will be..

Column 1 :PassengerI d	Column 2 : Survived (survived=0 &died=1)
Column 3 :Pclass	Column 4 : Name
Column 5 : Sex	Column 6 : Age
Column 7 :SibSp	Column 8 :Parch
Column 9 : Ticket	Column 10 : Fare
Column 11 :Cabin	Column 12 : Embarked

Description:

There have been huge disasters in the history of Map reduce, but the magnitude of the Titanic's disaster ranks as high as the depth it sank too. So much so that subsequent disasters have always been described as "titanic in proportion" – implying huge losses.

Anyone who has read about the Titanic, know that a perfect combination of natural events and human errors led to the sinking of the Titanic on its fateful maiden journey from Southampton to New York on April 14, 1912.

There have been several questions put forward to understand the cause/s of the tragedy – foremost among them is: What made it sink and even more intriguing How can a 46,000 ton ship sink to the depth of 13,000 feet in a matter of 3 hours? This is a mind boggling question indeed!

There have been as many inquisitions as there have been questions raised and equally that many types of analysis methods applied to arrive at conclusions. But this blog is not about analyzing why or what made the Titanic sink – it is about analyzing the data that is present about the Titanic publicly. It actually uses Hadoop MapReduce to analyze and arrive at:

- The average age of the people (both male and female) who died in the tragedy using Hadoop MapReduce.
- How many persons survived – traveling class wise.

This blog is about analyzing the data of Titanic. This total analysis is performed in Hadoop MapReduce.

This Titanic data is publically available and the Titanic data set is described below under the heading Data Set Description.

Using that dataset we will perform some Analysis and will draw out some insights like finding the average age of male and females died in Titanic, Number of males and females died in each

compartment.

DATA SET DESCRIPTION
Column 1 : PassengerI
Column 2 : Survived (survived=0 & died=1) Column 3 : Pclass
Column 4 : Name
Column 5 : Sex
Column 6 : Age
Column 7 : SibSp
Column 8 : Parch
Column 9 : Ticket
Column 10 : Fare
Column 11 : Cabin
Column 12 : Embarked

Mapper code:

```
public class Average_age {
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {

private Text gender = new Text();

private IntWritable age = new IntWritable();

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

String line = value.toString();

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

if(str.length>6){

gender.set(str[4]);

if((str[1].equals("0")) ){

if(str[5].matches("\\d+")){

int i=Integer.parseInt(str[5]);

age.set(i);
}
} }
context.write(gender, age)
```

```
}
}
```

Reducer Code:

```
public static class Reduce extends Reducer<Text,IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException { int sum = 0; int l=0; for (IntWritable val : values)
{ l+=1; sum += val.get();
} sum=sum/l; context.write(key, new
IntWritable(sum));
}
}
```

Configuration Code: job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

https://github.com/kiran0541/Map-Reduce/blob/master/Average%20age%20of%20male%20and%20female%20people%20died%20in%20titanic

Way to to execute the Jar file to get the result of the first problem statement: *hadoop jar average.jar /TitanicData.txt /avg_out*

Here 'hadoop' specifies we are running a Hadoop command and jar specifies which type of application we are running and average.jar is the jar file which we have created which consists the above source code and the path of the Input file name in our case it is TitanicData.txt and the output file where to store the output here we have given it as avg out.

Way to view the output:

hadoop dfs –cat /avg_out/part-r-00000
Here 'hadoop' specifies that we are running a Hadoop command and 'dfs' specifies that we are performing an operation related to Hadoop Distributed File System and '- cat' is used to view the contents of a file and 'avg_out/part-r-00000' is the file where the output is stored. Part file is created by default by the TextInputFormat class of Hadoop.

Lab Assignment No.	6
Title	Mongo DB: Installation and Creation of database and Collection CRUD Document: Insert, Query, Update and Delete Document.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

Aim : Mongo DB: Installation and Creation of database and Collection CRUD Document: Insert, Query, Update and Delete Document.

Theory :

To get started with MongoDB, you have to install it in your system. You need to find and download the latest version of MongoDB, which will be compatible with your computer system. You can use this (http://www.mongodb.org/downloads) link and follow the instruction to install MongoDB in your PC.

The process of setting up MongoDB in different operating systems is also different, here various installation steps have been mentioned and according to your convenience, you can select it and follow it.

1 Install Mongo DB in Windows

The website of MongoDB provides all the installation instructions, and MongoDB is supported by Windows, Linux as well as Mac OS.

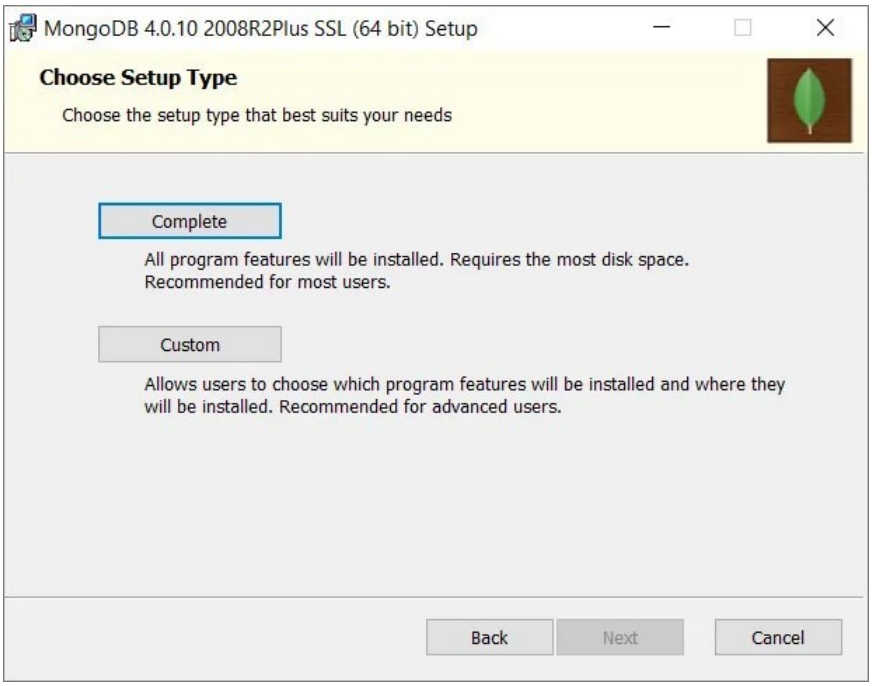
It is to be noted that, MongoDB will not run in Windows XP; so you need to install higher versions of windows to use this database.

Once you visit the link (http://www.mongodb.org/downloads), click the download button.

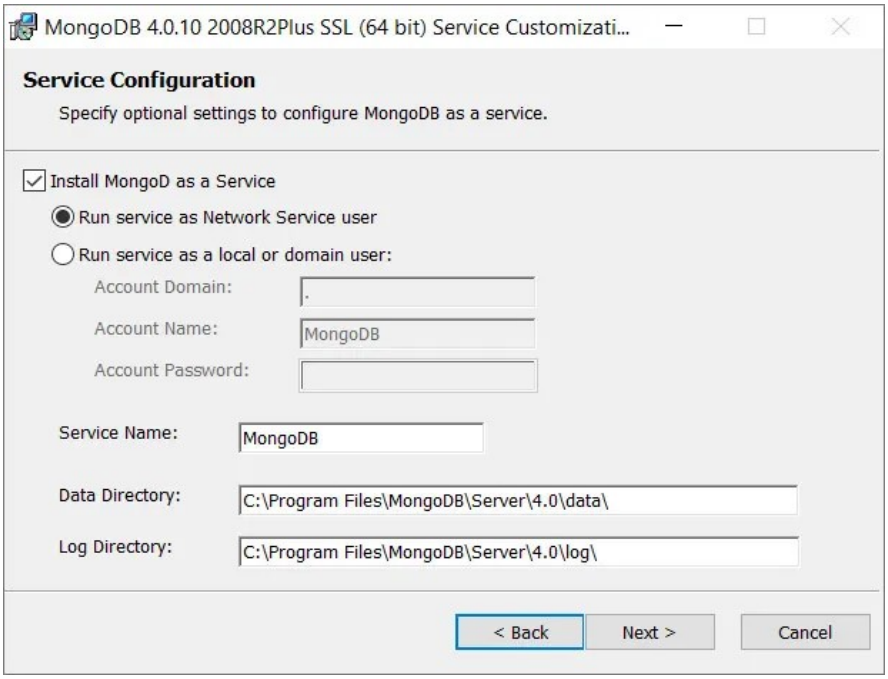
1. Once the download is complete, double click this setup file to install it. Follow the steps:

2

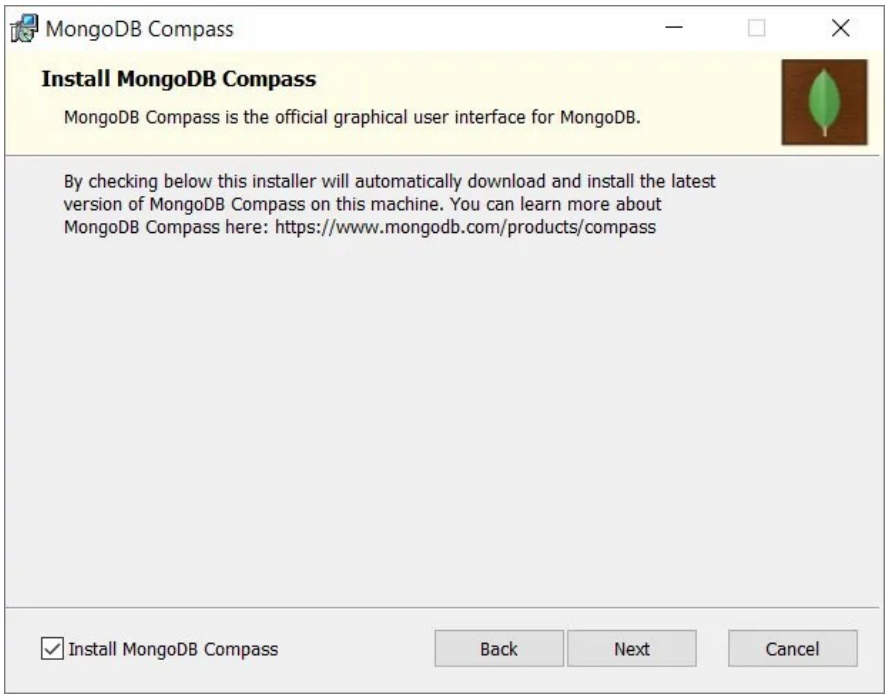
3. Now, choose Complete to install MongoDB completely.



4. Then, select the radio button "Run services as Network service user."



The setup system will also prompt you to install MongoDB Compass, which is MongoDB official graphical user interface (GUI). You can tick the checkbox to install that as well.

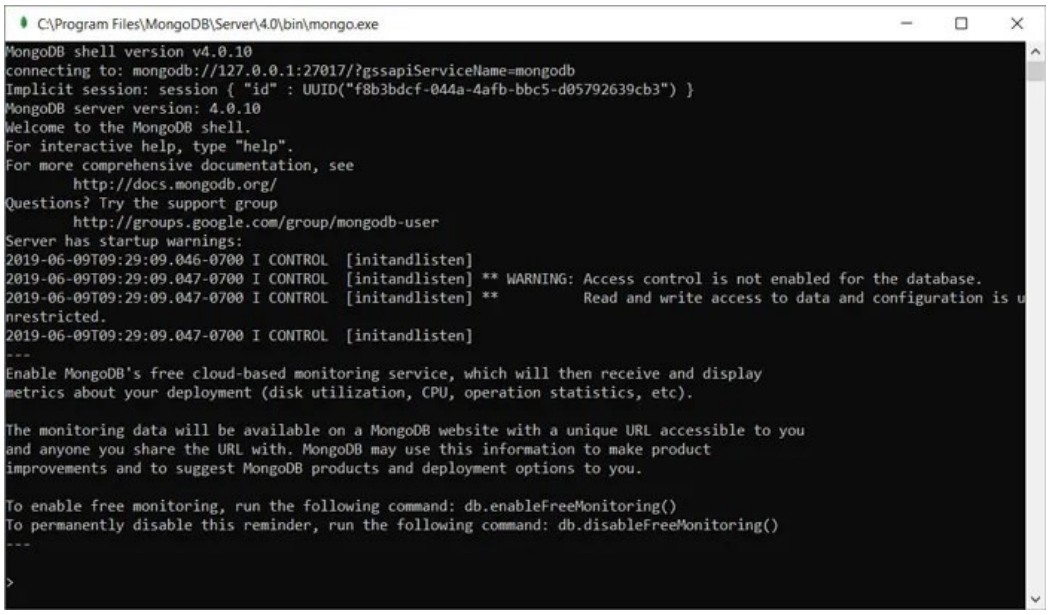


Once the installation is done completely, you need to start MongoDB and to do so follow the process:

1. Open Command Prompt.
2. Type: C:\Program Files\MongoDB\Server\4.0\bin
3. Now type the command simply: **mongodb** to run the server.

In this way, you can start your MongoDB database. Now, for running MongoDB primary client system, you have to use the command:

C:\Program Files\MongoDB\Server\4.0\bin>**mongo.exe**



CREATING A DATABASE USING MONGODB

To build a database in MongoDB, first construct a MongoClient object, and then supply a connection URL with the right IP address and the database name. If the database does not already exist, MongoDB will create it and connect to it.

Example: Create a database called “mydb”

```
var MongoClient = require('mongodb').MongoClient; var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) { if (err) throw err; console.log("Database created!");
  db.close(); });
```

Save the code above in a file called "demo_create_mongo_db.js" and run the file:

Run "demo_create_mongo_db.js"

C:\Users\Your Name>node demo_create_mongo_db.js This will give you this result:

Database created!

Note: MongoDB waits until you have created a collection (table), with at least one document

(record) before it actually creates the database (and collection).

The use Command

MongoDB use DATABASE_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax

Basic syntax of use DATABASE statement is as follows – use DATABASE_NAME

Example

If you want to use a database with name **<mydb>**, then **use DATABASE** statement would be as follows −

>use mydb switched to db mydb

To check your currently selected database, use the command **db**

>db

Mydb

If you want to check your databases list, use the command **show dbs**.

>show dbs local 0.78125GB test 0.23012GB

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

>db.movie.insert({"name":"tutorials point"})

>show dbs local 0.78125GB mydb 0.23012GB test 0.23012GB

In MongoDB default database is test. If you didn't create any database, then collections will be stored in test database.

DROP DATABASE

The dropDatabase() Method

MongoDB db.dropDatabase() command is used to drop a existing database.

Syntax

Basic syntax of dropDatabase() command is as follows – db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example

First, check the list of available databases by using the command, show dbs.

>show dbs local 0.78125GB mydb 0.23012GB test 0.23012GB

>

If you want to delete new database <mydb>, then dropDatabase() command would be as follows –

>use mydb switched to db mydb

>db.dropDatabase()

>{ "dropped" : "mydb", "ok" : 1 }

>

Now check list of databases.

>show dbs local 0.78125GB test 0.23012GB

>

CREATING COLLECTIONS IN MONGO DB

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

Syntax

Basic syntax of createCollection() command is as follows − db.createCollection(name, options)

In the command, **name** is name of collection to be created. **Options** is a document and is used to specify configuration of collection.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

Options parameter is optional, so you need to specify only the name of the collection. Following is the list of options you can use –

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexId	Boolean	(Optional) If true, automatically create index on _id field.s Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

While inserting the document, MongoDB first checks size field of capped collection, then it checks max field.

Examples

Basic syntax of **createCollection()** method without options is as follows −

```
>use test switched to db test
>db.createCollection("mycollection")
{ "ok" : 1 }
>
```

You can check the created collection by using the command show collections. >show collections mycollection system.indexes

The following example shows the syntax of **createCollection()** method with few important options −

> db.createCollection("mycol", { capped : true, autoIndexID : true, size : 6142800, max : 10000 }

){

"ok" : 0,

"errmsg" : "BSON field 'create.autoIndexID' is an unknown field.",

"code" : 40415,

"codeName" : "Location40415"

}

>

In MongoDB, you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

>db.tutorialspoint.insert({"name" : "tutorialspoint"}),

WriteResult({ "nInserted" : 1 })

>show collections mycol mycollection system.indexes tutorialspoint

>

The drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

Syntax

Basic syntax of **drop()** command is as follows − db.COLLECTION_NAME.drop()

Example

First, check the available collections into your database **mydb**.

>use mydb switched to db mydb >show collections mycol mycollection system.indexes tutorialspoint

>

Now drop the collection with the name **mycollection**.

>db.mycollection.drop() true

>

Again check the list of collections into database.

>show collections mycol system.indexes tutorialspoint >

drop() method will return true, if the selected collection is dropped successfully, otherwise it will return false.

CRUD DOCUMENT

As we know, we can use MongoDB for a variety of purposes such as building an application (including web and mobile), data analysis, or as an administrator of a MongoDB database. In all of these cases, we must interact with the MongoDB server to perform specific operations such as entering new data into the application, updating data in the application, deleting data from the application, and reading the data of the application.

MongoDB provides a set of simple yet fundamental operations known as CRUD operations that will allow you to quickly interact with the MongoDB server.

1. **Create Operations** — these operations are used to insert or add new documents to the collection. If a collection does not exist, a new collection will be created in the database. MongoDB provides the following methods for performing and creating operations:

Method	Description
db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple documents in the collection.

insertOne()

As the namesake, insertOne() allows you to insert one document into the collection. For this example, we're going to work with a collection called RecordsDB. We can insert a single entry into our collection by calling the insertOne() method on RecordsDB. We then provide the

information we want to insert in the form of key-value pairs, establishing the Schema.

Example:

```
db.RecordsDB.insertOne({ name: "Marsh", age: "6 years", species: "Dog", ownerAddress: "380 W. Fir Ave", chipped: true
})
```

If the create operation is successful, a new document is created. The function will return an object where “acknowledged” is “true” and “insertID” is the newly created “ObjectId.”

```
> db.RecordsDB.insertOne({
... name: "Marsh",
... age: "6 years",
... species: "Dog",
... ownerAddress: "380 W. Fir Ave",
... chipped: true
... })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5fd989674e6b9ceb8665c57d")
}
```

insertMany()

It’s possible to insert multiple items at one time by calling the insertMany() method on the desired collection. In this case, we pass multiple items into our chosen collection (RecordsDB) and separate them by commas. Within the parentheses, we use brackets to indicate that we are passing in a list of multiple entries. This is commonly referred to as a nested method.

Example:

```
db.RecordsDB.insertMany([{ name: "Marsh", age: "6 years", species: "Dog", ownerAddress: "380 W. Fir Ave", chipped: true}, {name: "Kitana", age: "4 years", species: "Cat", ownerAddress: "521 E. Cortland", chipped: true}])
```

```
db.RecordsDB.insertMany([{ name: "Marsh", age: "6 years", species:
"Dog",
ownerAddress: "380 W. Fir Ave", chipped: true}, {name: "Kitana", age: "4 years", species: "Cat",
ownerAddress: "521 E. Cortland", chipped: true}])
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("5fd98ea9ce6e8850d88270b4"),
        ObjectId("5fd98ea9ce6e8850d88270b5")
    ]
}
```

2. Read Operations

You can give specific query filters and criteria to the read operations to indicate the documents you desire. More information on the possible query filters can be found in the MongoDB manual. Query modifiers can also be used to vary the number of results returned.

MongoDB offers two ways to read documents from a collection:

- db.collection.find()
- db.collection.findOne()

find()

In order to get all the documents from a collection, we can simply use the find() method on our chosen collection. Executing just the find() method with no arguments will return all records currently in the collection.

db.RecordsDB.find findOne()

In order to get one document that satisfies the search criteria, we can simply use the findOne() method on our chosen collection. If multiple documents satisfy the query, this method returns the first document according to the natural order which reflects the order of documents on the disk. If no documents satisfy the search criteria, the function returns null. The function takes the following form of syntax.

db.{collection}.findOne({query}, {projection})

3 Update Operations

Update operations, like create operations, work on a single collection and are atomic at the document level. Filters and criteria are used to choose the documents to be updated during an update procedure.

You should be cautious while altering documents since alterations are permanent and cannot be reversed. This also applies to remove operations.

There are three techniques for updating documents in MongoDB CRUD**:**

- db.collection.updateOne()
- db.collection.updateMany() ● db.collection.replaceOne() **updateOne()**

With an update procedure, we may edit a single document and update an existing record. To do this, we use the updateOne() function on a specified collection, in this case "RecordsDB." To update a document, we pass two arguments to the method: an update filter and an update action. The update filter specifies which items should be updated, and the update action specifies how those items should be updated. We start with the update filter. Then we utilise the "$set" key and supply the values for the fields we wish to change. This function updates the first record that matches the specified filter.

updateMany()

updateMany() allows us to update multiple items by passing in a list of items, just as we did when inserting multiple items. This update operation uses the same syntax for updating a single document.

replaceOne()

The replaceOne() method is used to replace a single document in the specified collection. replaceOne() replaces the entire document, meaning fields in the old document not contained in the new will be lost.

4 Delete Operations

Delete operations, like update and create operations, work on a single collection. For a single document, delete actions are similarly atomic. You can provide delete actions with filters and criteria to indicate which documents from a collection you want to delete. The filter options use

the same syntax as the read operations.

MongoDB provides two ways for removing records from a collection:

● db.collection.deleteOne() ● db.collection.deleteMany() **deleteOne()**

deleteOne() is used to remove a document from a specified collection on the MongoDB server. A filter criterion is used to specify the item to delete.

It deletes the first record that matches the provided filter.

deleteMany()

deleteMany() is a method used to delete multiple documents from a desired collection with a single delete operation. A list is passed into the method and the individual items are defined with filter criteria as in deleteOne()

3.6 Let us sum up

- MongoDB is an open-source database that uses a document-oriented data model and a non-structured query language
- It is one of the most powerful NoSQL systems and databases around, today.
- The data model that MongoDB follows is a highly elastic one that lets you combine and store data of multivariate types without having to compromise on the powerful indexing options, data access, and validation rules.
- A group of database documents can be called a collection. The RDBMS equivalent to a collection is a table. The entire collection exists within a single database. There are no schemas when it comes

  to collections. Inside the collection, various documents can have varied fields, but mostly the documents within a collection are meant for the same purpose or for serving the same end goal.
- A set of key-value pairs can be designated as a document. Documents are associated with dynamic schemas. The benefit of having dynamic schemas is that a document in a single collection does not have to possess the same structure or fields. Also, the common fields in a collection document can have varied types of data.

Lab Assignment No.	7
Title	Hive: Introduction Creation of Database and Table, Hive Partition, Hive Built in Function and Operators, Hive View and Index.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

AIM:-

Install and Run Hive then use Hive to Create, alter and drop databases, tables, views, functions and Indexes.

DESCRIPTION

Hive, allows SQL developers to write Hive Query Language (HQL) statements that are similar to standard SQL statements; now you should be aware that HQL is limited in the commands it understands, but it is still pretty useful. HQL statements are broken down by the Hive service into MapReduce jobs and executed across a Hadoop cluster. Hive looks very much like traditional database code with SQL access. However, because Hive is based on Hadoop and MapReduce operations, there are several key differences. The first is that Hadoop is intended for long sequential scans, and because Hive is based on Hadoop, you can expect queries to have a very high latency (many minutes). This means that Hive would not be appropriate for applications that need very fast response times, as you would expect with a database such as DB2. Finally, Hive is read-based and therefore not appropriate for transaction processing that typically involves a high percentage of write operations.

ALGORITHM:

Apache HIVE INSTALLATION STEPS

1) Install MySQL-Server
   Sudo apt-get install mysql-server
2) Configuring MySQL UserName and Password
3) Creating User and granting all Privileges
   Mysql –uroot –proot
   Create user <USER_NAME> identified by <PASSWORD>
4) Extract and Configure Apache Hive

   tar xvfz apache-hive-1.0.1.bin.tar.gz
5) Move Apache Hive from Local directory to Home directory
6) Set CLASSPATH in bashrc
   Export HIVE_HOME = /home/apache-hive
   Export PATH = $PATH:$HIVE_HOME/bin

7) Configuring hive-default.xml by adding My SQL Server Credentials

```
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value> jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true
</value>
</property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
</property>
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>hadoop</value>
</property>
<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>hadoop</value>
</property>
```

8) Copying mysql-java-connector.jar to hive/lib directory.

SYNTAX for HIVE Database Operations

DATABASE Creation

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

Drop Database Statement

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

Creating and Dropping Table in HIVE

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]

Loading Data into table log_data

Syntax:

LOAD DATA LOCAL INPATH '<path>/u.data' OVERWRITE INTO TABLE u_data;

Alter Table in HIVE

Syntax

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 ...])

Creating and Dropping View

CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ...)] [COMMENT table_comment] AS SELECT ...

Dropping View

Syntax:

DROP VIEW view_name

Functions in HIVE

String Functions:- round(), ceil(), substr(), upper(), reg_exp() etc

Date and Time Functions:- year(), month(), day(), to_date() etc

Aggregate Functions :- sum(), min(), max(), count(), avg() etc

INDEXES

CREATE INDEX index_name ON TABLE base_table_name (col_name, ...)
AS 'index.handler.class.name'
[WITH DEFERRED REBUILD]
[IDXPROPERTIES (property_name=property_value, ...)]
[IN TABLE index_table_name]
[PARTITIONED BY (col_name, ...)]
[
[ROW FORMAT ...] STORED AS ...
| STORED BY ...
]
[LOCATION hdfs_path]
[TBLPROPERTIES (...)]

Creating Index

CREATE INDEX index_ip ON TABLE log_data(ip_address) AS
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;

Altering and Inserting Index

ALTER INDEX index_ip_address ON log_data REBUILD;

Storing Index Data in Metastore

SET hive.index.compact.file=/home/administrator/Desktop/big/metastore_db/tmp/index_ipaddress_result;

SET
hive.input.format=org.apache.hadoop.hive.ql.index.compact.HiveCompactIndexInputFormat;

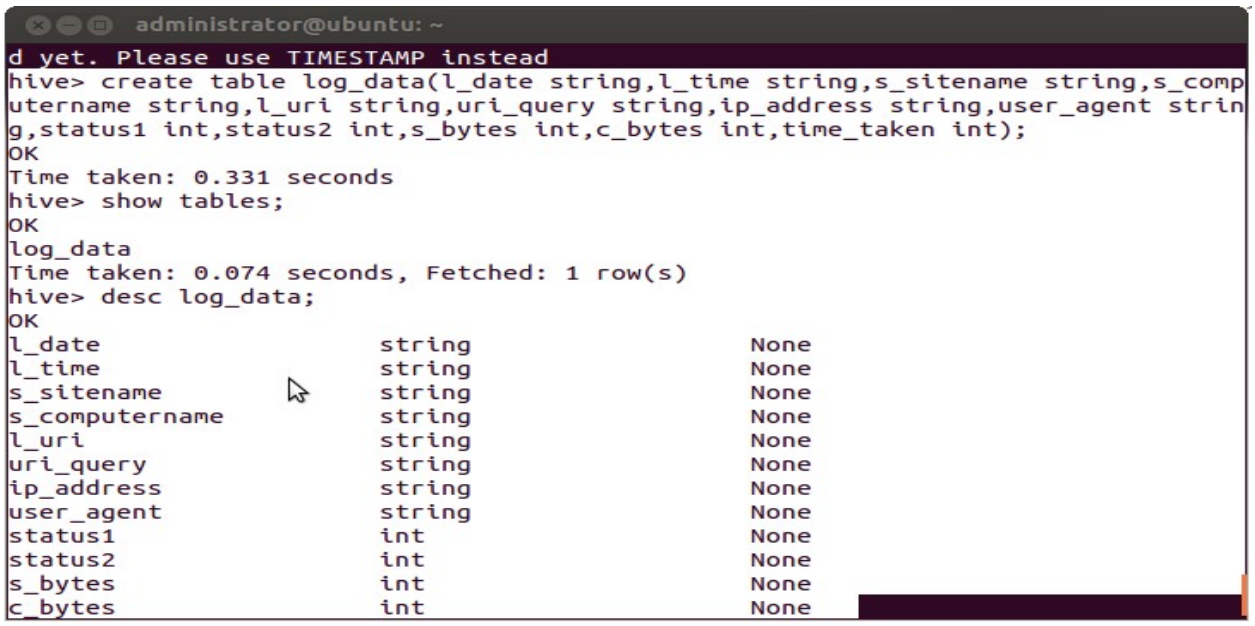
Dropping Index

DROP INDEX INDEX_NAME on TABLE_NAME;

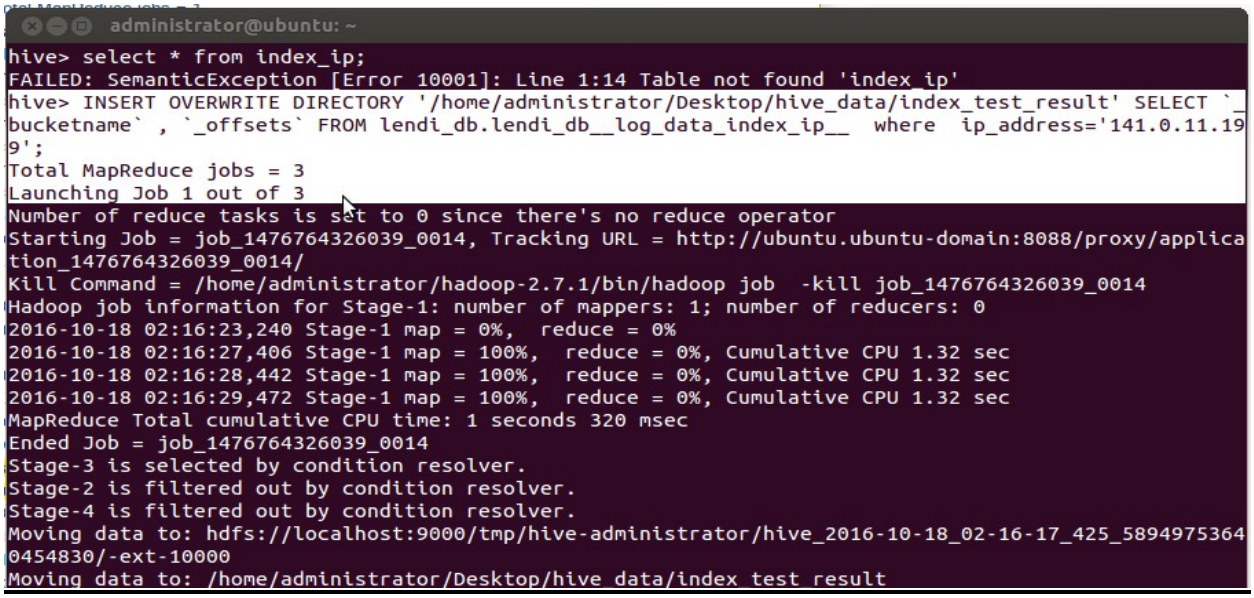
INPUT

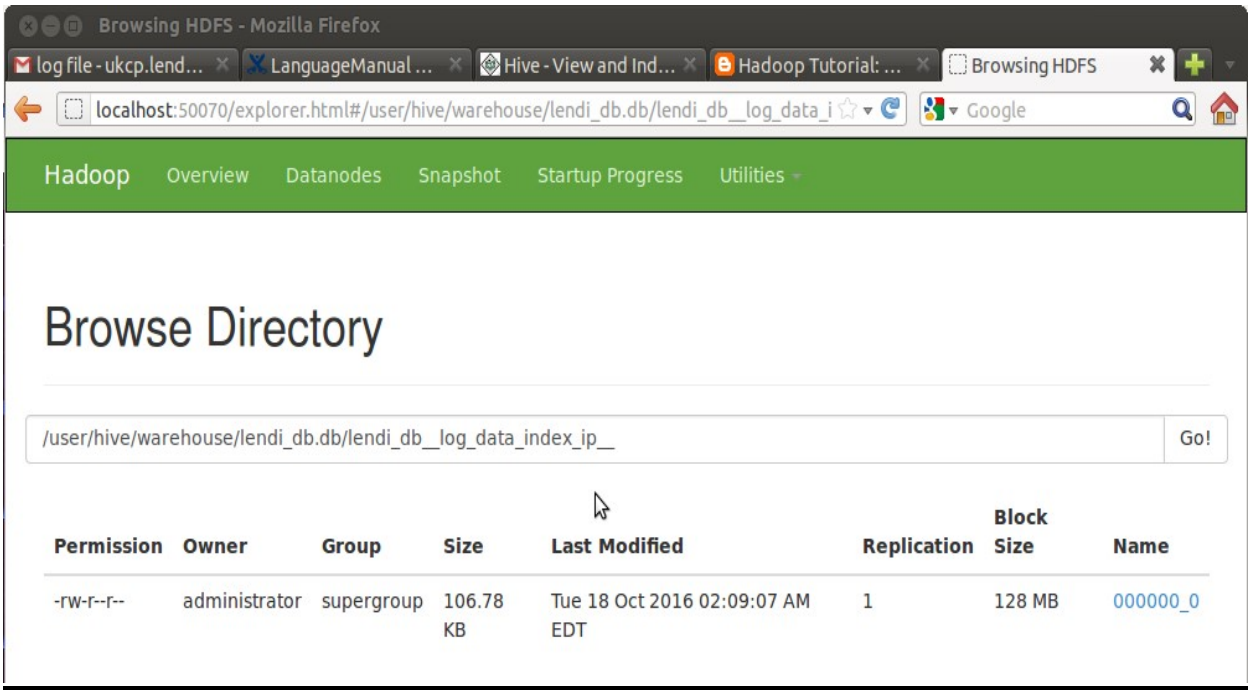
Input as Web Server Log Data

OUTPUT


administrator@ubuntu: ~
d yet. Please use TIMESTAMP instead
hive> create table log_data(l_date string,l_time string,s_sitename string,s_comp
utername string,l_uri string,uri_query string,ip_address string,user_agent strin
g,status1 int,status2 int,s_bytes int,c_bytes int,time_taken int);
OK
Time taken: 0.331 seconds
hive> show tables;
OK
log_data
Time taken: 0.074 seconds, Fetched: 1 row(s)
hive> desc log_data;
OK
l_date string None
l_time string None
s_sitename string None
s_computername string None
l_uri string None
uri_query string None
ip_address string None
user_agent string None
status1 int None
status2 int None
s_bytes int None
c_bytes int None


administrator@ubuntu: ~
0.6.20.6 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC
R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2) 304
11 498 0
2014-12-23 23:08:38 W3SVC1 NEWINTSERV2 /trf/elast/images/small/pic3.jpg
6.20.6 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC
R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2) 304
10 497 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elast/css/demo.css - 10.
ozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+CLR+3.0.0
CLR+1.1.4322;+InfoPath.2) 304 0 210 458 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elast/css/elastislide.css -
0.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+
06;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 210 465 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elast/images/small/pic11.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elast/images/small/pic12.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elast/images/small/pic10.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elast/images/small/pic9.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 210 467 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elast/images/small/pica.jpg


administrator@ubuntu: ~
hive> select * from index_ip;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'index_ip'
hive> INSERT OVERWRITE DIRECTORY '/home/administrator/Desktop/hive_data/index_test_result' SELECT `_
bucketname` , `_offsets` FROM lendi_db.lendi_db__log_data_index_ip__ where ip_address='141.0.11.19
9';
Total MapReduce jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1476764326039_0014, Tracking URL = http://ubuntu.ubuntu-domain:8088/proxy/applica
tion_1476764326039_0014/
Kill Command = /home/administrator/hadoop-2.7.1/bin/hadoop job -kill job_1476764326039_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-10-18 02:16:23,240 Stage-1 map = 0%, reduce = 0%
2016-10-18 02:16:27,406 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:28,442 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:29,472 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
MapReduce Total cumulative CPU time: 1 seconds 320 msec
Ended Job = job_1476764326039_0014
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/tmp/hive-administrator/hive_2016-10-18_02-16-17_425_5894975364
0454830/-ext-10000
Moving data to: /home/administrator/Desktop/hive_data/index_test_result


Browsing HDFS - Mozilla Firefox
log file - ukcp.lend...
LanguageManual ...
Hive - View and Ind...
Hadoop Tutorial: ...
Browsing HDFS
localhost:50070/explorer.html#/user/hive/warehouse/lendi_db.db/lendi_db__log_data_i
Google
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Browse Directory
/user/hive/warehouse/lendi_db.db/lendi_db__log_data_index_ip__
Go!
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r--
administrator
supergroup
106.78 KB
Tue 18 Oct 2016 02:09:07 AM EDT
1
128 MB
000000_0

Lab Assignment No.	8
Title	Visualization: Connect to data, Build Charts and Analyze Data, Create Dashboard, Create Stories using Tableau/PowerBI.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

Aim : Visualization: Connect to data, Build Charts and Analyze Data, Create Dashboard, Create Stories using Tableau/PowerBI.

Theory :

Tableau is a visual analysis solution that allows people to explore and analyze data with simple drag and drop operations. It has a user-friendly interface that creates reports that look great right at the beginning. Tableau is a rapid BI Software.

Great Visualizations:

Allows to connect to the data, visualize and creative interactive, sharable dashboards in a few clicks.

Ease of use:

It's easy enough that any excel user can learn it, but powerful enough to satisfy even the most complex analytical problems

Fast:

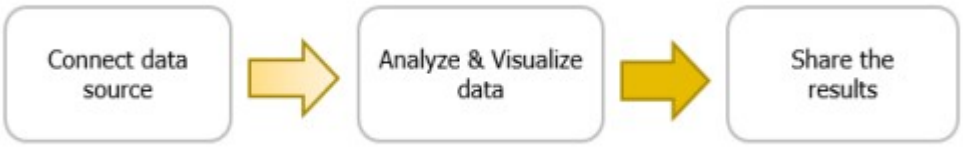
We can create interactive dashboards, quick filters and calculations.

WHY TABLEAU?

As per Gartner survey "Tableau was chosen more often for functionality than any other vendor in the survey, with the highest product functionality scores". "Tableau's products often fill an unmet need in organizations that already have a BI standard "and more frequently deployed than other interactive visualization vendors as a complementary capability to an existing BI platform.

Gartner positions tableau as "Leader". Companies frequently cite that breadth and ease of use along with high business benefits as the primary reasons that they choose Tableau.

Three main stages in Tableau



Connect data source: Connect Tableau to any data source like MS-Excel,

MySQL, and Oracle. Tableau connects data in two ways Live connect and Extract.

Analyze & Visualize: Analyze the data by filtering, sorting and Visualize the data using the relevant chart provided. Tableau automatically analyzes the data as follows: Dimensions: Tableau treats any field containing qualitative, categorical information as dimension. All dimensions are indicated by "Blue" color.

Measures: Tableau treats any field containing numeric information as a measure. All measures are indicated by "Green"

color Tableau suggests the recommended charts based on the dimensions and measures.

Share: Tableau Dashboards can be shared as word documents, pdf files, images.

Maps in Tableau

Tableau automatically assigns geographic roles with common geographically names, such as Country, State/Province, City etc. Fields with geographic role will automatically generates longitude and latitude coordinates on a map view. Fields with assigned geographic roles will have a globe icon next to them. In Tableau, there are two types of maps to choose from when creating a visualization with geographic data:

- ❖ Symbol Maps and
- ❖ Filled Maps.

Symbol Maps: These are simple maps that use a type of mark to represent a data point, such as a filled circle.

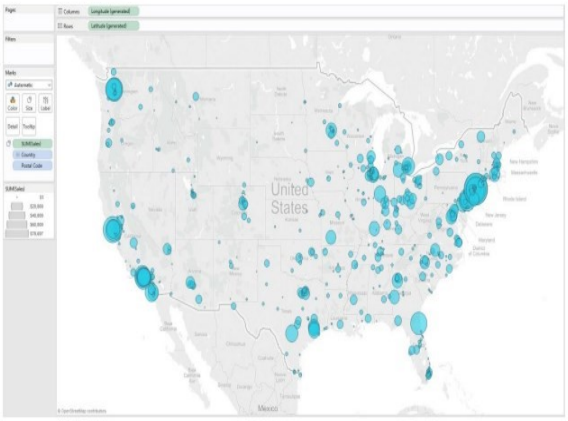


Fig . Symbol Map

Filled Maps: Instead of displaying data points, filled maps uses shading on a country or state basis to indicate relationships.

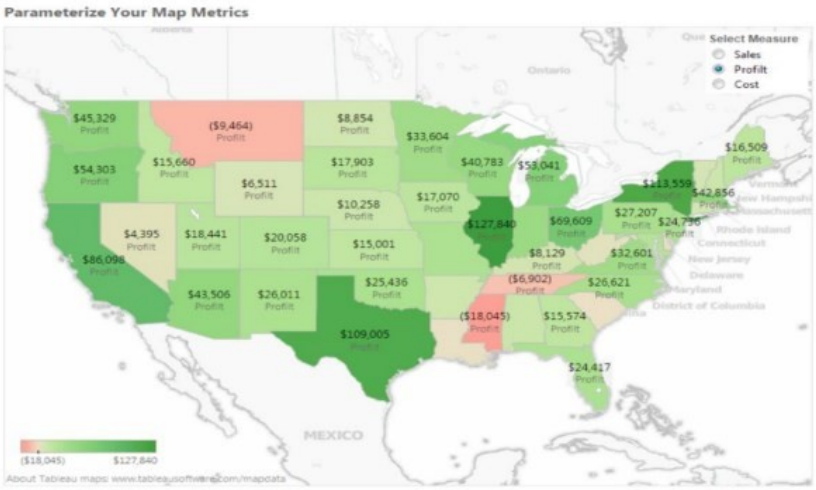


Fig . Filled Map

CUSTOM SQL QUERIES TABLEAU

What is Custom SQL Query?

Tableau queries each data source using SQL that's specific to the data type. Tableau allows the SQL used to query a data source to be customized in order to manipulate the joins, filters, and field lengths and types produce a more accurate output.

Sample SQL query:

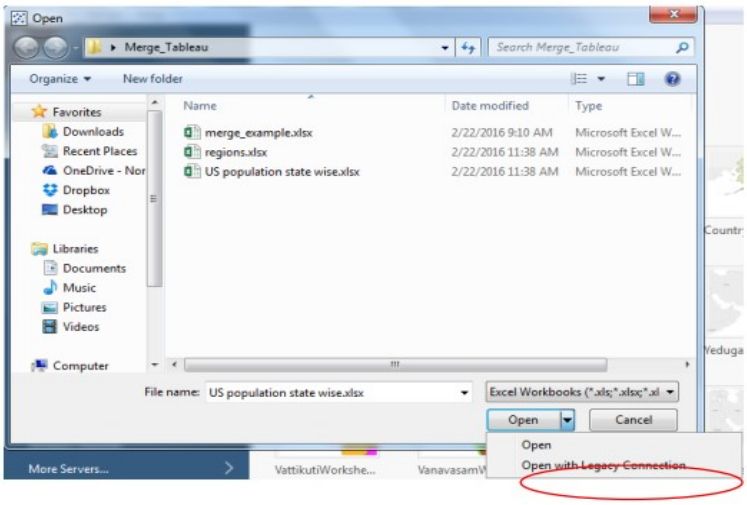
```
SELECT ['us states data 1$'].[State] AS [State],
['us states data 1$'].[Population] AS [Population],
['us states data 1$'].[Region] AS [Region]
FROM ['us states data 1$']
UNION
SELECT ['us states data 2$'].[State] AS [State],
['us states data 2$'].[Population] AS [Population],
['us states data 2$'].[Region] AS [Region]
FROM ['us states data 2$']
UNION
SELECT ['us states data 3$'].[State] AS [State],
['us states data 3$'].[Population] AS [Population],  ['us states data 3$'].[Region] AS [Region]
FROM ['us states data 3$']
UNION
SELECT ['us states data 4$'].[State] AS [State],
['us states data 4$'].[Population] AS [Population],
['us states data 4$'].[Region] AS [Region]
FROM ['us states data 4$']
```

Steps to visualize using SQL in Tableau:

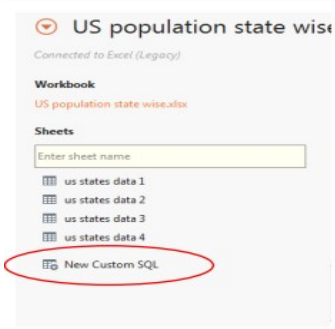
1. The following excel workbook consists of 4 sheets consisting of US population state wise.

US population state wise.xlsx

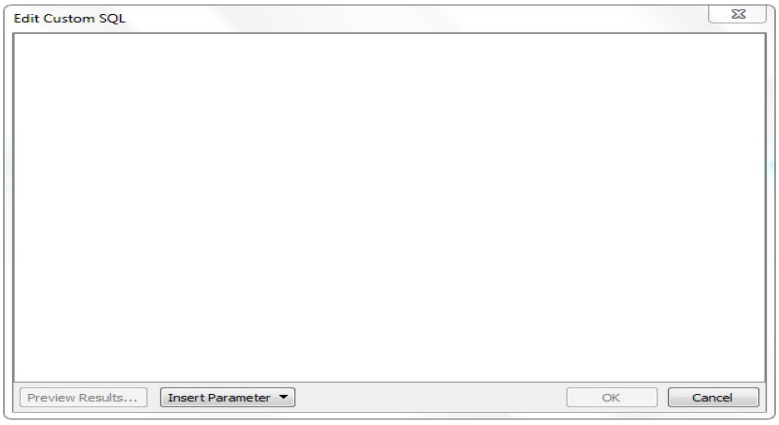
2. Open Tableau and connect the Excel data as Open with Legacy Connection.



3. Now your Tableau window looks like this with an additional sheet named as New Custom SQL.



4. Click on New Custom SQL then you will be prompted to Edit Custom SQL as shown.



5. Now we need to write SQL query to perform union operation on the excel sheets and generate a combined sheet.

Ex:

```
SELECT [Sheet1$].[ID] AS [ID],
[Sheet1$].[Type] AS [Type],
[Sheet1$].[Value] AS [Value]
```

```
FROM [Sheet1$]
UNION ALL
SELECT [Sheet2$].[ID] AS [ID],
[Sheet2$].[Type] AS [Type],
[Sheet2$].[Value] AS [Value]
FROM
[Sheet2$]
```

The above SQL query performs union operation on two sheets namely Sheet1 and Sheet2 with ID, Type, and Value as their columns.

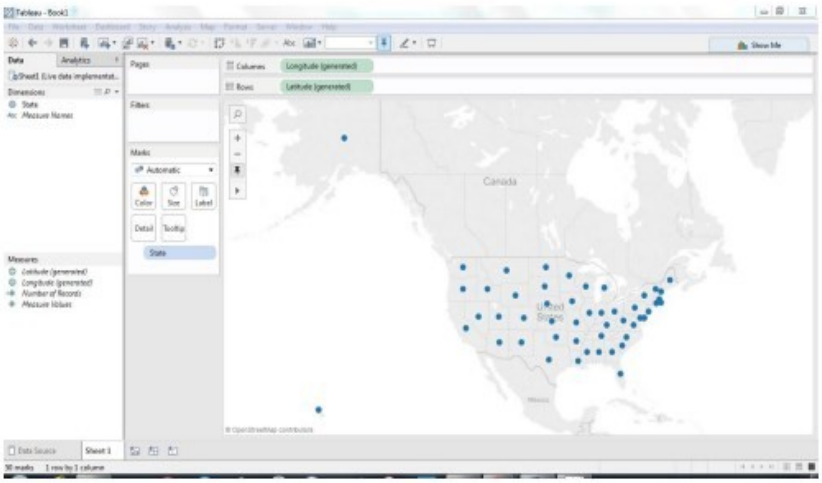
6. After writing the query click on Preview Results on the Edit Custom SQL window pane to check whether the query worked fine or not.
7. The resulting data should contain all the states, population and region name to which it belong to.
8. Click on the Tableau sheet to visualize the data.
9. Now you will see State, Region as Dimensions and Population as Measures.
10. Now visualize the data derived.

Live data implementation Tableau

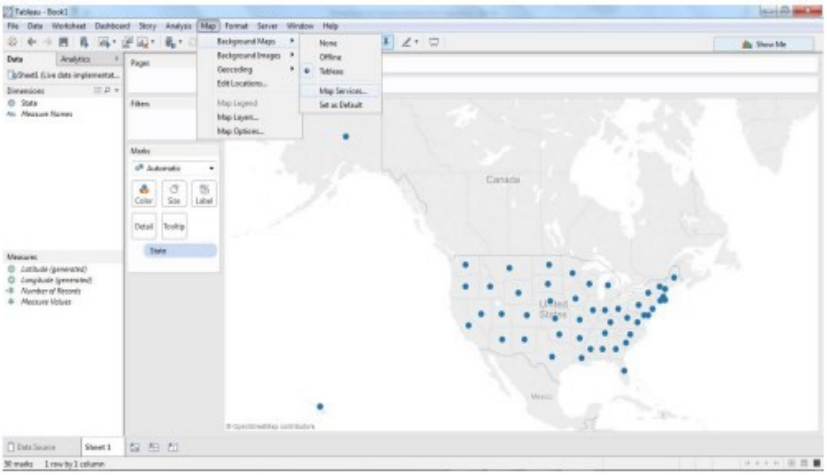
Live Data or Real-time data (RTD) denotes information that is delivered immediately after collection. There is no delay in the timeliness of the information provided. Real-time data is often used for navigation or tracking. A Web Map Service (WMS) is a standard protocol for serving georeferenced map images which a map server generates using data from a GIS database. A Web Map Service (WMS) produces an image (e.g. GIF, JPG) of geospatial data. Steps to implement live map using Tableau:

1. Open Tableau and connect the Excel data.
2. Drag and drop state Dimension in Detail menu provided in Marks window pane.
3. Now you can see Symbol map representing the states of United States.

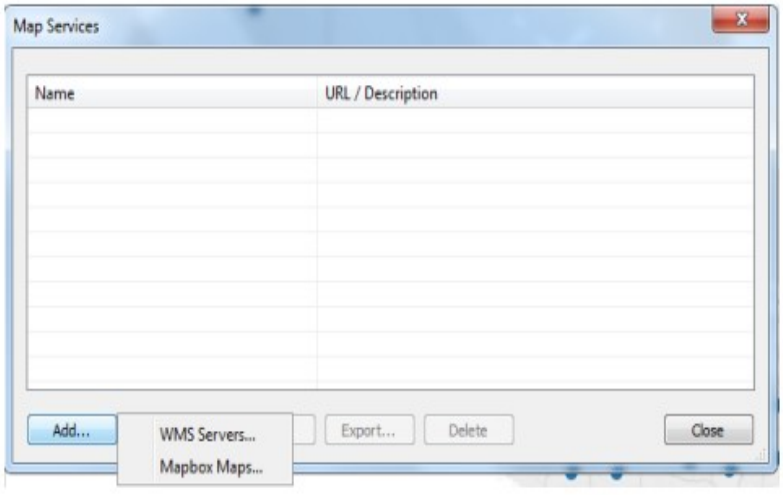
It looks as shown below



4. In this we will be using OGCWMS server to get the live geospatial data.
5. Click on Map item in the menu bar, then select Background Maps and select WMS Server as shown

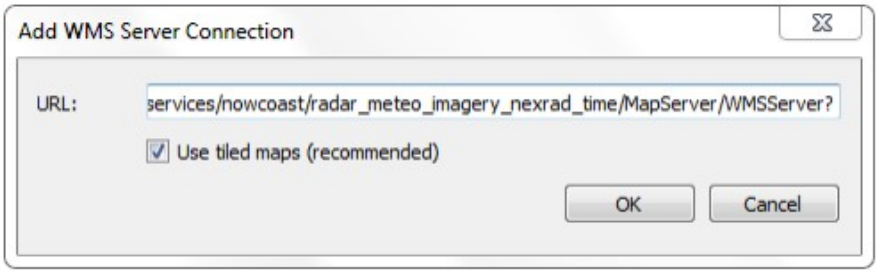


6. After clicking on Map Services you will be prompted a window as shown

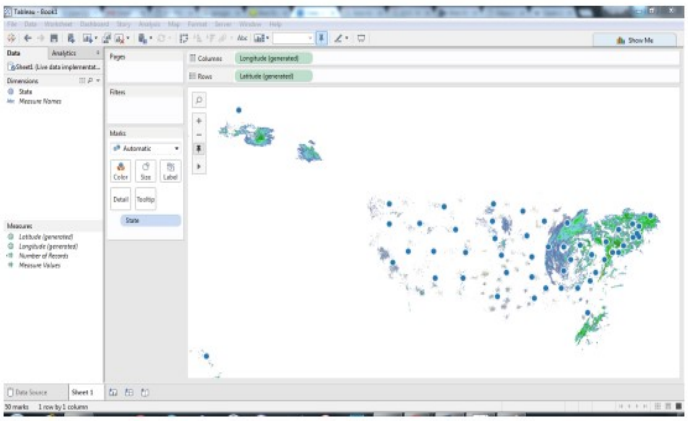


7. Click on Add and select WMS Servers. After clicking on WMS Servers a window pane as shown below will be prompted to enter the URL of the server.

http://nowcoast.noaa.gov/arcgis/services/nowcoast/radar_meteo_imagery_nexrad_time/M apServer/WMSServer?



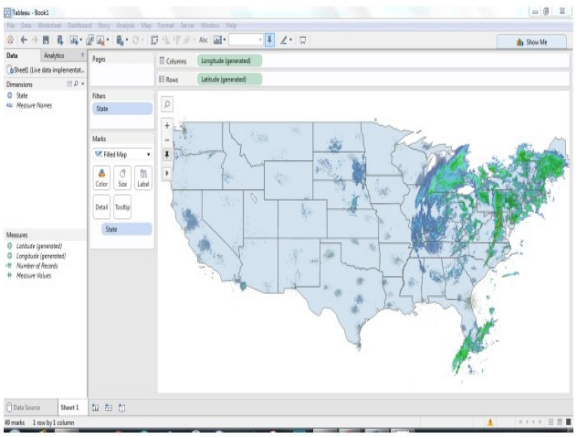
8. Now click on OK and then close the WMS server window pane and a map can be seen as shown



9. Change the map type to Filled map.

10. Change the color transparency to less than 20% so that the required map is visible.

11. T hen required map looks like this



Tree map displays hierarchical data by using nested rectangles which together represent a whole. They were invented by Ben Shneiderman in 1992 as a way to visualize tree structures in a space-constrained layout. In a treemaps, each branch of a tree is a rectangle, which is tiled with smaller rectangles to represent sub-branches. This allows the viewer to easily see patterns that would be hard to spot in bar charts or area charts. The main benefits of tree maps are, they make efficient use of compact space. Treemaps are becoming popular in infographics and presentations, especially for data analysis that requires detailed views of many items. With this feature one can explore the data hierarchy effortlessly and simultaneously decent level of estimation is also possible for quantitative aspects of information.

Steps to visualize Treemaps in Tableau:

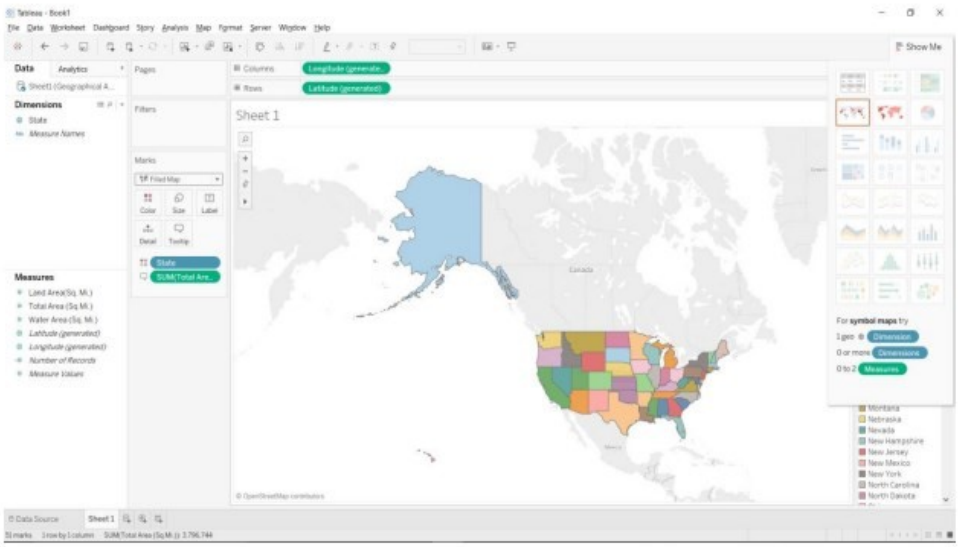
1. The data is about the various products sold at a fashion store in the cities of different states of the united states
2. Start with opening tableau and connect the data, which is of type excel to Tableau
3. Click on sheet1 which is at the bottom of the page to visualize the data
4. Now to the left of the screen you can see a small window showing Dimensions and Measures. The dimensions of the data are state, city, store name, product, SKU number, year, quarter, month and week. The measures of the data are Quantity sold and sales revenue
5. You can also see rows and columns at the top of the page, now drag sales revenue from the measures and drop it in the rows section. You can see a bar chart with single bar showing sales revenues
6. Now drag states from the dimension to the column section, you can see a bar chart showing the sales revenues of different states
7. Now click on Show Me button on the top right corner of the screen, you can see various maps suggested by the Tableau tool to visualize the data. The maps which are bright in color shows us that those maps are suitable to visualize the given data
8. Select the Tree Map by clicking it, you will see a Tree Map representing various states in green color with different shades
9. The rectangle which is large and bright represents the highest quantity according to sales revenue and similarly the smaller and lighter shade of the rectangles show less sales revenue
10. Now select the city tab from the dimension section and drop it on the Label in the Marks section. You can observe that the rectangles are split into even smaller rectangles according to the cities in each state
11. Now select the product from the dimension section and drop it on the Label in the Marks section. You can observe that the rectangles are split into even smaller rectangles according to the products.
12. Now select the Quantity sold tab from the dimension section and drop it on the Label in the Marks section. You can see that the actual quantity sold of each product.

DatatreeMap.xlsx

Geographical map visualization using Tableau:

Steps to visualize Filled Maps in Tableau:

1. Open Tableau and connect excel sheet.
2. The data with states and their geographical data is shown in Tableau.
3. Drag and drop states onto columns and Total Area onto rows.
4. Drag and drop state onto color and total area onto tool tip.
5. Now select the filled map. Then the map is represented with the geographical data of each state as shown below.



Visualization of Combo and Gauge Charts using QlikView

1. For this we have taken two different datasets.

• Dataset -1 (Cost_Student). Contains data which representing amount spend by the countries, on a student per year for primary to secondary education and postsecondary education.

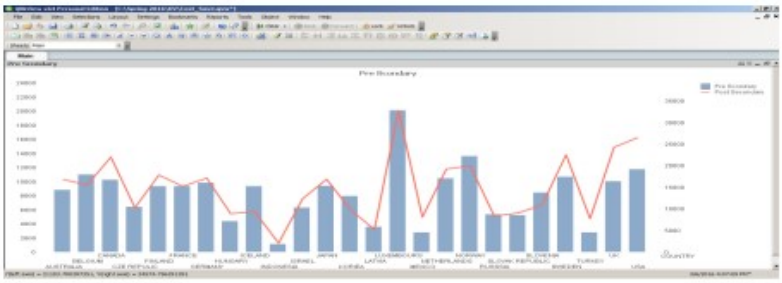
• Dataset - 2 (Sales_Rep). Contains data, which represents the sales made by the representatives of a stationary chain around USA over particular dates.

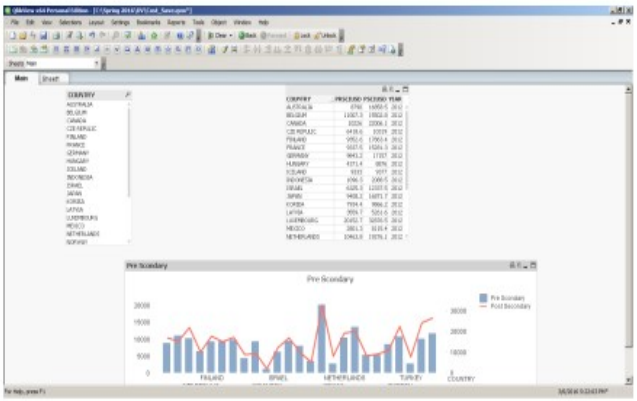
OrderDa	Regi	Rep	Iten	Sal	Unit Co	Revent	Targe
1/6/14	East	Jones	Pencil	95	1.99	189.05	115
1/23/14	Central	Kivell	Binder	50	19.99	999.50	115
2/9/14	Central	Jardine	Pencil	36	4.99	179.64	115
2/26/14	Central	Gill	Pen	27	19.99	539.73	115
3/15/14	West	Sorvino	Pencil	56	2.99	167.44	115
4/1/14	East	Jones	Binder	60	4.99	299.40	115
4/18/14	Central	Andrews	Pencil	75	1.99	149.25	115
5/5/14	Central	Jardine	Pencil	90	4.99	449.10	115
5/22/14	West	Thompson	Pencil	32	1.99	63.68	115
6/8/14	East	Jones	Binder	60	8.99	539.40	115
6/25/14	Central	Morgan	Pencil	90	4.99	449.10	115
7/12/14	East	Howard	Binder	29	1.99	57.71	115
7/29/14	East	Parent	Binder	81	19.99	1,619.19	115
8/15/14	East	Jones	Pencil	35	4.99	174.65	115
9/1/14	Central	Smith	Desk	2	125.00	250.00	15
9/18/14	East	Jones	Pen Set	16	15.99	255.84	115
10/5/14	Central	Morgan	Binder	28	8.99	251.72	115
10/22/14	East	Jones	Pen	64	8.99	575.36	115
11/8/14	East	Parent	Pen	15	19.99	299.85	115
11/25/14	Central	Kivell	Pen Set	96	4.99	479.04	115
12/12/14	Central	Smith	Pencil	67	1.29	86.43	115
12/29/14	East	Parent	Pen Set	74	15.99	1,183.26	115
1/15/15	Central	Gill	Binder	46	8.99	413.54	115
2/1/15	Central	Smith	Binder	87	15.00	1,305.00	115
2/18/15	East	Jones	Binder	4	4.99	19.96	115
3/7/15	West	Sorvino	Binder	7	19.99	139.93	115
3/24/15	Central	Jardine	Pen Set	50	4.99	249.50	115
4/10/15	Central	Andrews	Pencil	66	1.99	131.34	115
4/27/15	East	Howard	Pen	96	4.99	479.04	115
5/14/15	Central	Gill	Pencil	53	1.29	68.37	115
5/31/15	Central	Gill	Binder	80	8.99	719.20	115
6/17/15	Central	Kivell	Desk	5	125.00	625.00	15
7/4/15	East	Jones	Pen Set	62	4.99	309.38	115
7/21/15	Central	Morgan	Pen Set	55	12.49	686.95	115
8/7/15	Central	Kivell	Pen Set	42	23.95	1,005.90	115
8/24/15	West	Sorvino	Desk	3	275.00	825.00	15
9/10/15	Central	Gill	Pencil	7	1.29	9.03	115
9/27/15	West	Sorvino	Pen	76	1.99	151.24	115
10/14/15	West	Thompson	Binder	57	19.99	1,139.43	115
10/31/15	Central	Andrews	Pencil	14	1.29	18.06	115
11/17/15	Central	Jardine	Binder	11	4.99	54.89	115
12/4/15	Central	Jardine	Binder	94	19.99	1,879.06	115
12/21/15	Central	Andrews	Binder	28	4.99	139.72	115

2. Open QlikView. A QlikView getting started screen opens, now click on the New Document button on right below corner of the Wizard.
3. A new Window opens up (Kind of pop up) close the pop up window. (Since it is Auto chart generator, we do not want that wizard).
4. On closing the pop up, user will be able to see a Main blank sheet. Now have quick look on to the toolbar.
5. Over the tool bar below to the selections labe1, click on the Edit Script icon (Icon looks like pen on a paper).
6. A new Edit Script window opens up (We use this window to load an Excel file), bottom of the window under the Data from files Click Table Files button, and choose the saved Excel file, then click open.
6. A new File Wizard Type window opens up. Now select the Tables drop down and choose Cost_Student if

it has not defaulted and click the Labels drop down and choose Embedded Labels. Click next until, next button disables (If it disables you are on final screen). Select check box load all and click finish.

7. On Finish you will be able to see Edit Script screen , with script added to load excel file(Script shown below) On Edit Script window tool bar below file , click the reload icon.It asks you save the file. LOAD * FROM [C:\Manusha\DataViz\DataViz-Qlik\QlikView_Dataset.xls] (biff, embedded labels, table is Cost_Per_Student$);
8. Now a sheet property window opens, please choose Country (Which works as a filter).
9. Right Click anywhere on the sheet, a window opens choose New Sheet Object and select Table box. New window open again, add all the available fields and click apply. You will be able to see a table box with content same as to excel sheet.
10. Right Click anywhere on the sheet, a window opens choose New Sheet Object and select Chart. New window open again, Select Combo Chart from chart type and click next.
11. Choose Country as dimension and click next now choose the expressions. • Expression 1 for bar chart Sum (PRSCIUSD), now click Add button to add one more expression Sum (PSCIUSD) and click finish.
12. Now we have seen the Combo Chart.
13. If you observe both pre-Secondary and post-secondary are on same axis. Let us split them. Right click on the chart and choose the properties, and navigate to the Axes tab, Select Post-Secondary Expression and choose the position Right (Top) click apply and ok. You will see charts similar to below.





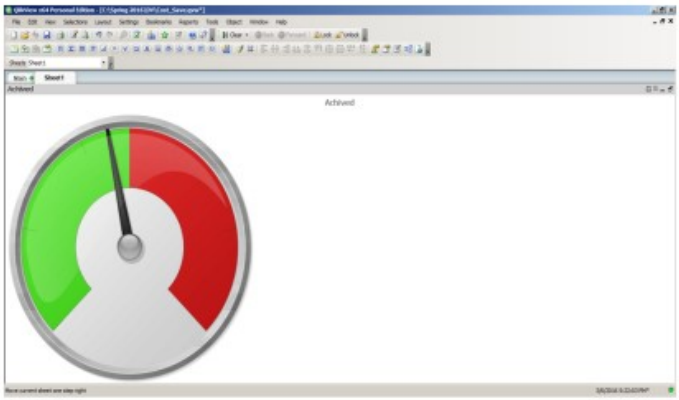
14. Now click on layout shown in the tool bar and add new sheet. A new sheet will open.

15. Repeat step 7 to step 12, in step 9 choose sales_Rep sheet and in step 11 Choose Rep to the filed display in list boxes.

16. Right Click anywhere on the sheet, a window opens choose New Sheet Object and select Chart. New window open again, Select Gauge from chart type and click next.

17. Gauge charts have no dimensions, so we do not select any dimension, will be moving directly to expression.

18. Since we are calculating the performance, we add a division between sales and targets (Sum (sales)/sum (targets)), add label for expression (Sales Achieved) finish you will see a chart like below.

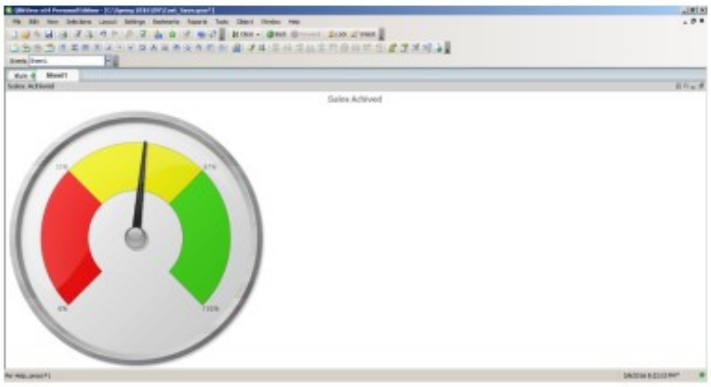


19. Let us make it more readable, right click, choose the properties, navigate to presentation add a segment in segment setup (we can go with 2 segments as well for clear user readability we choose 3) and change the colors of each segment by choosing color band (Red, Yellow, Green).

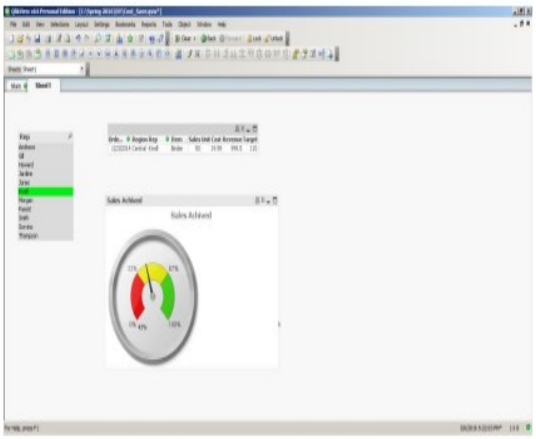
20. Now look for Show scale section left –below to segment setup add margin units a 4 and show labels on every major entry margin units 1

(Select the check box Show scale if it is not selected).

22. Now Switch to Number tab since we are representing it in percentage select the radio button Fixed to and select the check box Show in percentage and click apply.



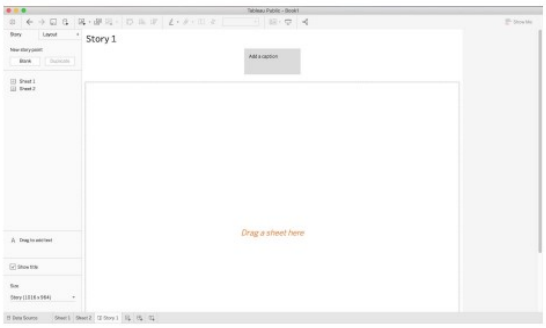
23. In order to see the needle value (Speedometer current value). Right click on the chart and select properties move to presentation tab and select Add button next to Text in Chart (located below right corner of the window) and add this formula(=num(Sum(Sales)/sum(Target),'##%')), and click apply, value appears on the top left corner of the chart. Press



CTL +SHIFT to drag the text anywhere in the chart. Final Gauge chart with data along with chart looks like below.

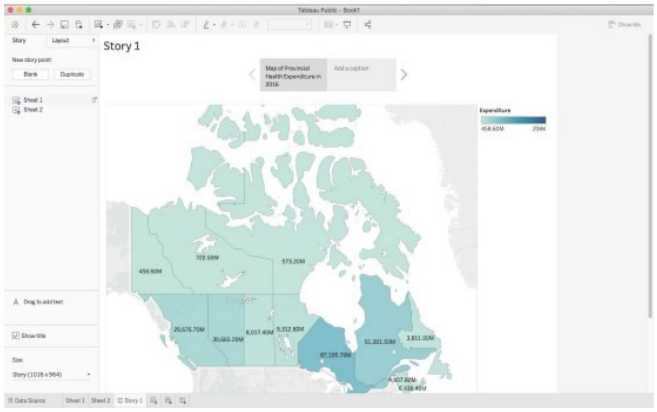
7 CREATING A STORY WITH TABLEAU PUBLIC

With Tableau public, you are able to organize your data in order to tell a meaningful story. This is beneficial when you are doing a presentation, creating an article, or uploading to a website, as it helps your audience understand your data. Stories are created through assembling the different worksheets and dashboards. We can highlight important data points, add text box and pictures to help convey our story. We will use our health expenditure worksheets to create a tailoring in story and illustrate the changes in Canada's spending in a meaningful way. To begin, select "New Story" at the bottom right of your screen

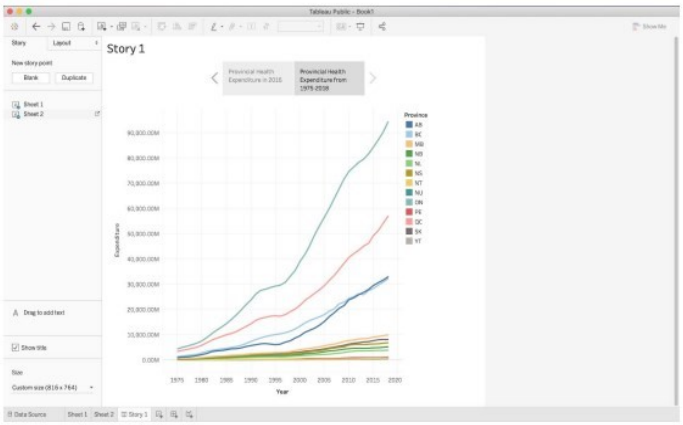


Drag "Sheet 1" and "Sheet 2" on to "Drag a sheet here". We can rename each storyboard by clicking "Add a caption".

Rename Sheet 1 to “Provincial Health Expenditure in 2016”.

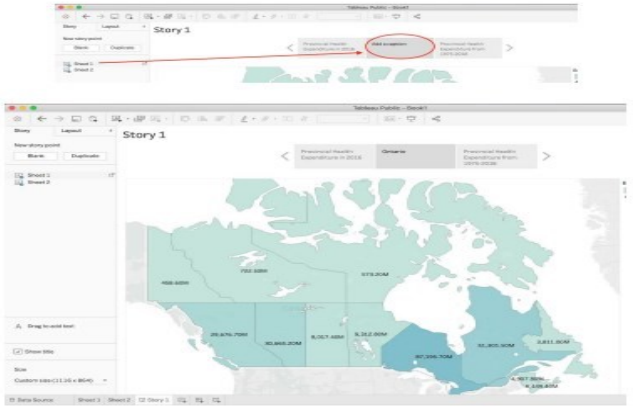


Use the arrows located on the side of the caption field to navigate to Sheet 2. Click on “Add a caption” and rename Sheet 2 to “Provincial Health Expenditure from 1975-2018”.



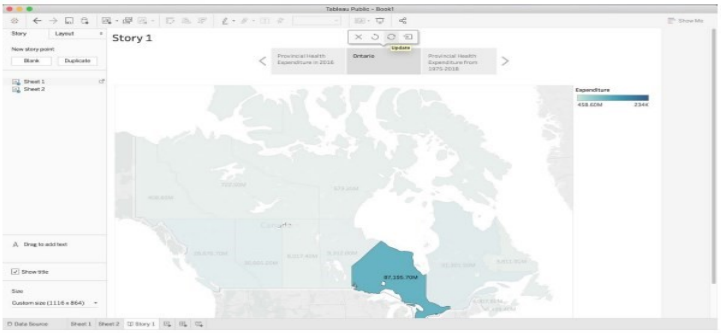
174174

In this story, we are going to narrow in and draw attention to the province or territory that is spending the most amount of money on health. Drag an additional copy of “Sheet 1” and drop it between the two existing sheets. Select “Add a caption” and rename it to “Ontario”.

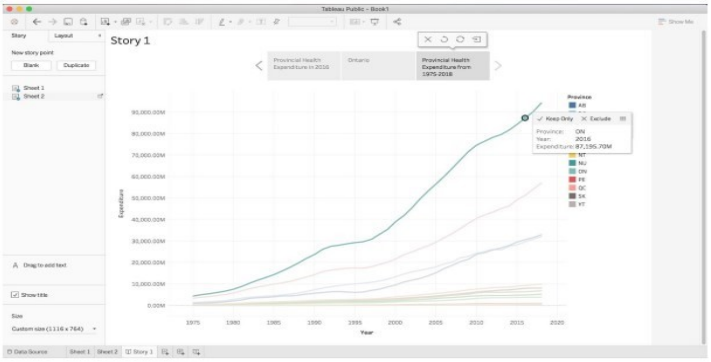


On the map, click on the province Ontario and then navigate to the caption field and select “Update”. Your screen will

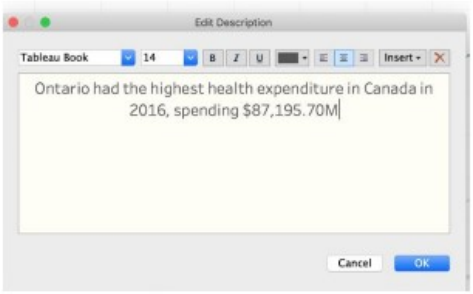
show Ontario highlighted from the rest of Canada.



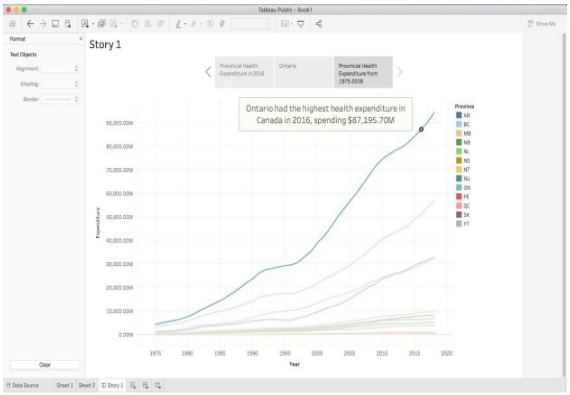
Select the right arrow to navigate to “Provincial Health Expenditure from 1975-2018”. Hover over the line representing Ontario and select the data point representing health expenditure during the year 2016. Then click “Update”.



We can add a textbox to label the highlighted pointed by dragging “Drag to add text” on to the line graph. Write a key message in the textbox, such as “Ontario had the highest health expenditure in Canada in 2016, spending $87,195.70M”. Select “OK”.



You can the edit the text box by selecting “More options” which will open a drop-down menu. Expand the text box by dragging the borders in order to show the full message.

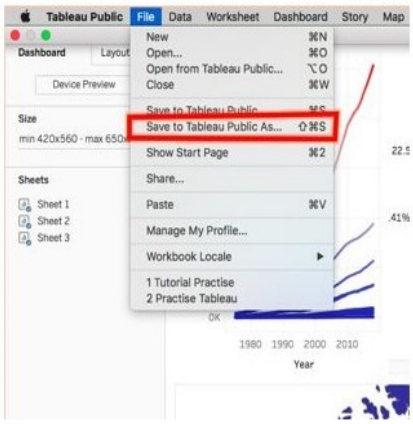


We have now created a story with three sheets of how Ontario had the highest health expenditure in the year 2016. If you choose to add a dashboard, it will allow your audience to play with data. You can navigate between the story as shown below:

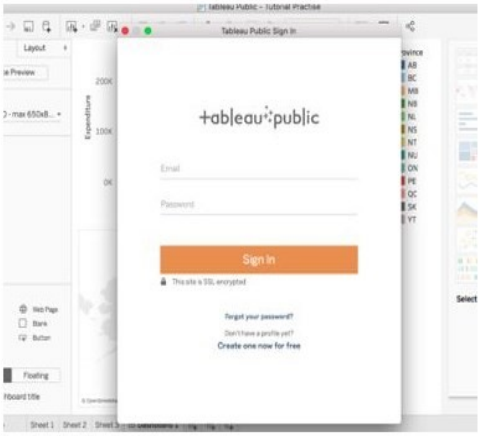
176176

SAVING AND PUBLISHING YOUR TABLEAU PUBLIC WORKBOOK

Once satisfied with your workbook, which includes sheets, dashboards, and stories, you can publish it to the Tableau Public website. This is the only way to save your work when using Tableau Public, so make sure to do it if you wish to return to the workbook in the future. Once ready to publish, select the “Save to Tableau Public As…” option under the “File” tab.

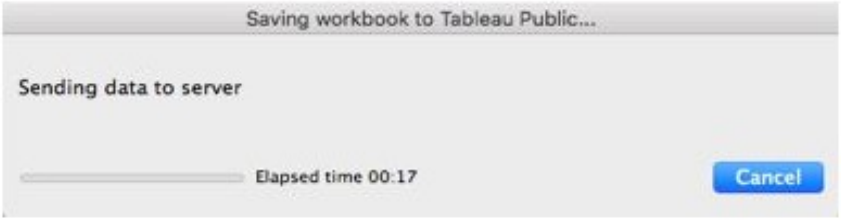


From here, you will likely be prompted to log in to the Tableau Public website. You can create an account for free if you have not already.



Then, create a title for your workbook.

Your workbook will then be uploaded to the Tableau Public server. This may take a couple minutes.



You will then be directed to the webpage on which your workbook is publicly available for download. Your workbook

has now been published and saved! This page includes information on you, including a link to your Tableau Public profile, as well as additional information on the workbook. There are several options for making use of your work data visualizations that can be found at the bottom right of the workbook image. By clicking Share (A), you can get the code for embedding the workbook into a website or the link to find the workbook on tableau public. The Share button also gives you the option to share your work over email, Facebook, or Twitter. By clicking Download (B), you or any other Tableau Public users can download your workbook and make use of the data themselves.

On Tableau Public, all saved data visualizations are uploaded and available to all other users. This means you can use other users' work for your website, presentations, or research as well. To search for interesting data, there is a search function (D) at the top right corner of the webpage, along with highlighted visualizations (A), authors (B), and blogs (C).

178178

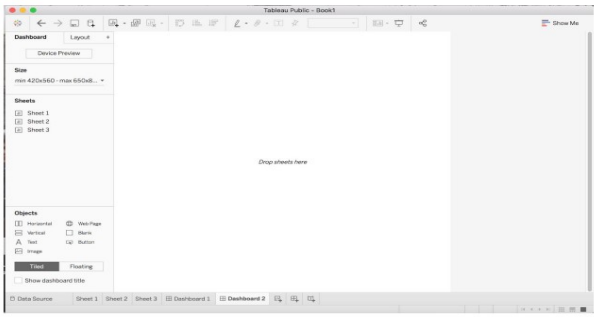
When you wish to access your published workbook, or any public workbooks you've downloaded, in the future, simply open the Tableau Public application and your workbooks will be there waiting for you!

A DASHBOARD WITH TABLEAU

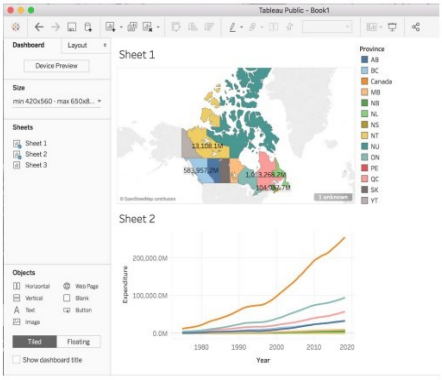
Dashboards are a great way to combine your data visualizations and have them interact with one another. A lot of businesses use dashboards to keep up-to-date in real time about key performance indicators at a glance. In this example, we will combine just two of our data visualizations, the map and the line graph from the first section of the tutorial, but in reality, it can be used to combine much visualization at once.

The first step in creating your dashboard is to open up the Dashboard tab at the bottom of the screen:

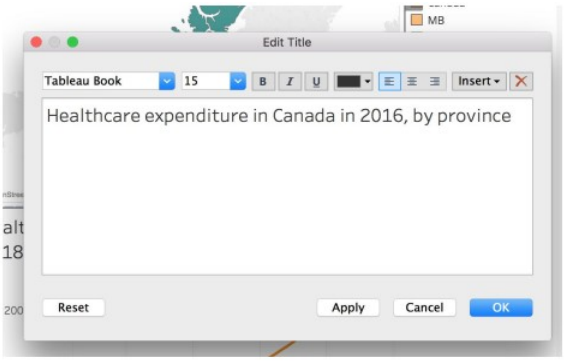
After clicking this icon, your screen should open to this:



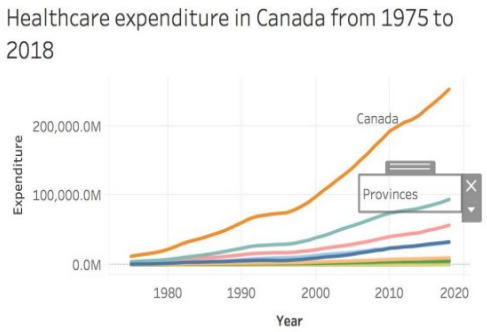
This is your Dashboard Sheet. On the left side you can see that there is a list of the sheets you have made from your current data source. To build your dashboard, drag the sheet you want in to the centre where it says Drop sheets here. For our purposes, we will need to drag Sheet 1 and Sheet 2 where the map and line graph are saved. When you drag, you will notice an area of your screen will shade over where your graph will drop when you put it down. Organize your dashboard to look like the following:



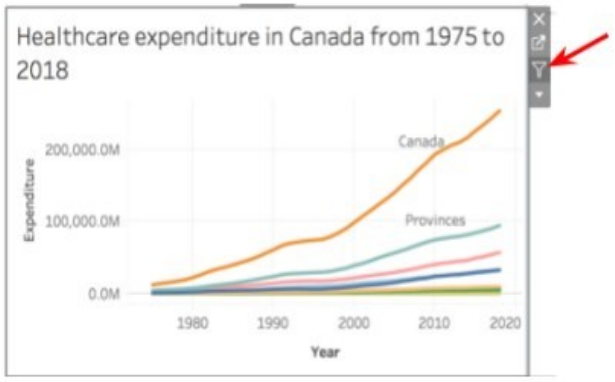
Now to add titles to the graphs that were chosen, double click on the automatic titles generated based on the sheet name, and a new window should appear, type in a title that describes the graph like so:



We can also add additional titles and objects to the dashboard by choosing an object from the Objects side panel and dragging it to the dashboard. We are going to add titles to the bottom line graph to differentiate between the Canada line and the provinces. To do this, drag A Text to the area near the orange line that corresponds to the sum of all provinces expenditure throughout the years. Type in "Canada". Drag A Text once more to label the remaining provinces. Your bottom graph should look like this:



Now, to add an interactive layer between the graphs, we can choose a graph that can act as a filter to the other. We will choose the line graph to act as a filter to the map. To do this, click on the line graph and a grey sidebar should appear. From this bar, click the filter icon to use this graph as a filter:



Now, when you click a given line, it will be highlighted on the above map: