

Big Data Analytics Lab

Exp 1:

Aim: Implement the following Data structures in java

Program:-

a) Linked List

```
import
java.util.*;class
l1list{
public static void main(String args[])
{
    int n,i,j,k,num,a,z=0;
    LinkedList l=new
    LinkedList ();
    Scanner S=new Scanner(System.in);
    do{System.out.println("Enter your
    Choice"+"\\n1:insertion\\n2:deletion\\n3:display\\n4:exit");n=S.nextInt();
    switch(n)
    {
        case 1: System.out.println("Enter your Choice"+"\\n1:insertion at first\\n2 :insertion at
        last\\nother numbers:insertion after a number");
        k=S.nextInt(
        );if(k==1)
        { System.out.println("enter number");
        num=S.nextInt();
        l.addFirst(num);
        }
```

```
else if(k==2)
```

```
{
```

```
System.out.println("enter number");
```

```

num=S.nextInt(
);
l.addLast(num)
;

    }
else
    { System.out.println("enter number after which u want to insert");
      a=S.nextInt();
      ListIterator iter
      =l.listIterator();
      while(iter.hasNext())
      {
          if(a==iter.next())
          {
              System.out.println("enter
              number");num=S.nextInt();
              iter.add(num)
              ;z=1;
          }
      }
      if(z==0)
      System.out.println("number not in
      list");z=0;
    }
break;

```

case 2: System.out.println("Enter your Choice"+"n1:DELETION at first\n2:DELETION at last\nOTHER DELETION at middle");

```

        k=S.nextInt(
    );if(k==1)
    { l.removeFirst();
    }
    else if(k==2)
    {
        l.removeLast();
    }
    else
    { System.out.println("enter number after which u want to delete");
      a=S.nextInt();
      ListIterator it
      =l.listIterator();
      while(it.hasNext())
      {
          if(a==it.next())
          {
              it.remove
              ();z=1;
          }
          }
          if(z==0)
          System.out.println("number not in
              list");z=0;
      }
      break;
    case 3:
        System.out.println(l
        );break;
    case 4: n=0; break;

```

```

    }

    }while(n!=0);
}
}

```

Out Put:

Enter your Choice

1:insertion

2:deletion

3:display

4:exit

1

Enter your Choice

1:insertion at first

2:insertion at last

other numbers:insertion after a number

2

enter number

77

Enter your Choice

1:insertion

2:deletion

3:display

4:exit

2

Enter your Choice

1:DELETION at first

2:DELETION at last

OTHER DELETION at middle

2

Enter your Choice

1:insertion

2:deletion

3:display

4:exit

Implement the following Data structures in java

b) Stacks

Programe:-

```
import java.util.*;
```

```
class Stk {
```

```
    public static void main(String args[]) {
```

```
        int n, i = -1, size, k;
```

```
        Stack<Integer> c = new Stack<>();
```

```
        Scanner S = new Scanner(System.in);
```

```
        System.out.println("Enter Stack size");
```

```
        size = S.nextInt();
```

```
        do {
```

```
            System.out.println("Enter your choice" + "\n 1: Insertion " + "\n 2: Deletion " + "\n 3: Display" + "\n 0 for exit");
```

```

n = S.nextInt();

switch (n) {
    case 1:
        if (i < size - 1) {
            System.out.println("Insert element");
            k = S.nextInt();
            c.push(k);
            i++;
        } else {
            System.out.println("Cannot insert");
        }
        break;

    case 2:
        if (i >= 0) {
            c.pop();
            i--;
        } else {
            System.out.println("Cannot delete");
        }
        break;

    case 3:
        if (i >= 0) {
            System.out.println(c);
        } else {
            System.out.println("Cannot display");
        }
        break;
}
} while (n != 0);
}

```

output:

```

enter Stack size2
enter your choice
1:insertion
2:deletion
3:display0 for exit
1
insert element
77 07
enter your choice
1:insertion
2:deletion
3:display0 for exit
enter your choice
1:insertion
2:deletion
3:display0 for exit

```

c) Set

Programe:-

```
import
java.util.*;
import java.io.*;

public class set
{
    public static void main(String args[]) throws FileNotFoundException
    {
        Set<String> dictionaryWords=readWords("words");

        Set<String>
        documentWords=readWords("alice30.txt"); for(String
        word: documentWords)
        {
            if(!dictionaryWords.contains(word))
            {
                System.out.println(word);
            }
        }
    }

    public static Set<String> readWords(String filename) throws FileNotFoundException
    {
        Set <String> words= new
        TreeSet<String>(); Scanner in = new
        Scanner(new File(filename));
        in.useDelimiter("[^a-zA-Z]+");
        while(in.hasNext())
        {
            words.add(in.next().toLowerCase());
        }
    }
}
```

```
}
```

```
return words;
```

```
}
```

```
}
```

d) Map

```
import java.awt.*;
```

```
import java.util.*;
```

```
public class
```

```
MapaDemo
```

```
{
```

```
public static void main(String args[])
```

```
{
```

```
    Map<String, Color> fav= new
```

```
    HashMap<String,Color>();
```

```
    fav.put("Romeo",Color.BLUE);
```

```
    fav.put("juliet",Color.RED);
```

```
    fav.put("adam",Color.BLACK);
```

```
    fav.put("Eve",Color.GREEN);
```

```
    Set <String> keyset=
```

```
    fav.keySet();for(String key:
```

```
    keyset)
```

```
    {
```

```
        Color value=fav.get(key);
```

```
        System.out.println(key+" "+value);
```

```
    }
```

```
}
```

```
}
```


Exp 2:

Perform setting up and installing hadoop in standalone mode.

INSTALL APACHE HADOOP 2.6.0 IN UBUNTU (SINGLE NODE SETUP)

1)Installing Oracle Java 8

Apache Hadoop is java framework, we need java installed on our machine to get it run over operating system. Hadoop supports all java version greater than 5 (i.e. **Java 1.5**). So, Here you can also try Java 6, 7 instead of Java 8.

```
$ sudo add-apt-repository ppa:webupd8team/java
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install oracle-java8-installer
```

It will install java source in your machine at **/usr/lib/jvm/java-8-oracle**To

verify your java installation, you have to fire the following command like,

```
$ java -version
```

2) Creating a Hadoop user for accessing HDFS and MapReduce

To avoid security issues, we recommend to setup new Hadoop user group and user account to deal with all Hadoop related activities.

We will create hadoop as system group and hduser as system user by,

```
$ sudo addgroup hadoop
```

```
$ sudo adduser --ingroup hadoop hduser
```

3) Installing SSH

SSH (“Secure SHell”) is a protocol for securely accessing one machine from another. Hadoop uses SSH for accessing another slaves nodes to start and manage all HDFS and MapReduce daemons.

```
$ sudo apt-get install openssh-server
```

```
$ sudo apt-get install ssh
$ sudo adduser hduser sudo
```

Now, we have installed SSH over Ubuntu machine so we will be able to connect with this machine as well as from this machine remotely.

```
$ cd Downloads
```

```
$ sudo mv hadoop-2.6.0.tar.gz /usr/local/
```

```
$ cd
```

```
$ sudo su hduser
```

```
$ cd
```

Configuring SSH

Once you installed SSH on your machine, you can connect to other machine or allow other machines to connect with this machine. However we have this single machine, we can try connecting with this same machine by SSH. To do this, we need to copy generated RSA key (i.e. id_rsa.pub) pairs to authorized_keys folder of SSH installation of this machine by the following command,

```
$ ssh-keygen -t rsa -P ""
```

```
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

In case you are configuring SSH for another machine (i.e. from master node to slave node), you have to update the above command by adding the hostname of slave machine.

4) Disabling IPv6

Since Hadoop doesn't work on IPv6, we should disable it. One of another reason is also that it has been developed and tested on IPv4 stacks. Hadoop nodes will be able to communicate if we are having IPv4 cluster. (Once you have disabled IPV6 on your machine, you need to reboot your machine in order to check its effect. In case if you don't know how to reboot with command use `sudo reboot`)

For getting your IPv6 disabled in your Linux machine, you need to update `/etc/sysctl.conf` by adding following line of codes at end of the file,

```
$ sudo gedit /etc/sysctl.conf
```

```
# disable ipv6
```

```
net.ipv6.conf.all.disable_ipv6 = 1
```

```
net.ipv6.conf.default.disable_ipv6 = 1
```

```
net.ipv6.conf.lo.disable_ipv6 = 1
```

Tip:- You can use nano, gedit, and Vi editor for updating all text files for this configuration purpose.

```
$ sudo reboot
```

Installation Steps

1)Download latest Apache Hadoop source from Apache mirrors

First you need to download [Apache Hadoop 2.6.0](#) (i.e. *hadoop-2.6.0.tar.gz*) or latest version

source from Apache download Mirrors. You can also try stable hadoop to get all latest features as well as recent bugs solved with Hadoop source. Choose location where you want to place all your hadoop installation, I have chosen */usr/local/hadoop*

```
$sudo su hduser
```

```
## Locate to hadoop installation parent dir
```

```
$ cd /usr/local/
```

```
## Extract Hadoop source
```

```
$ sudo tar -xzf hadoop-2.6.0.tar.gz
```

```
## Move hadoop-2.6.0 to hadoop folder
```

```
$ sudo mv hadoop-2.6.0 /usr/local/hadoop
```

```
## Assign ownership of this folder to Hadoop user
```

```
$ sudo chown hduser:hadoop -R /usr/local/hadoop
```

```
## Create Hadoop temp directories for Namenode and Datanode
```

```
$ sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
```

```
$sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
```

```
## Again assign ownership of this Hadoop temp folder to Hadoop user
```

```
$ sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

2)Update Hadoop configuration files

```
$cd
```

```
$ sudo gedit .bashrc
```

```
## Update hduser configuration file by appending the
```

```
## following environment variables at the end of this file.
```

```
# -- HADOOP ENVIRONMENT VARIABLES START -- #
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

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

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

```
export PATH=$PATH:$HADOOP_HOME/sbin
```

```
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

```
export HADOOP_COMMON_HOME=$HADOOP_HOME
```

```
export HADOOP_HDFS_HOME=$HADOOP_HOME
```

```
export YARN_HOME=$HADOOP_HOME
```

```
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
```

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

```
# -- HADOOP ENVIRONMENT VARIABLES END -- #
```

Configuration file : hadoop-env.sh

```
$cd /usr/local/hadoop/etc/hadoop
```

```
$ sudo gedit hadoop-env.sh
```

```
## Update JAVA_HOME variable,
```

```
JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

```
Configuration file : core-site.xml
```

```
$ sudo gedit core-site.xml
```

```
## Paste these lines into <configuration> tag
```

```
<property>
```

```
<name>fs.default.name</name>
```

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

```
</property>
```

```
Configuration file : hdfs-site.xml
```

```
$ sudo gedit hdfs-site.xml
```

```
## Paste these lines into <configuration> tag
```

```
<property>
```

```
<name>dfs.replication</name>
```

```
<value>1</value>
```

```
</property>
```

```
<property>
```

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

```
<value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>

</property>

<property>

    <name>dfs.datanode.data.dir</name>

    <value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>

</property>
```

Configuration file : yarn-site.xml

```
$ sudo gedit yarn-site.xml
```

Paste these lines into <configuration> tag

```
<property>

    <name>yarn.nodemanager.aux-services</name>

    <value>mapreduce_shuffle</value>

</property>

<property>

    <name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>

    <value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>
```

Configuration file : mapred-site.xml

```
$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
/usr/local/hadoop/etc/hadoop/mapred-site.xml
```

```
$ sudo gedit mapred-site.xml
```

Paste these lines into <configuration> tag

```
<property>

    <name>mapreduce.framework.name</name>

    <value>yarn</value>

</property>
```

\$ sudo reboot

3) Format Namenode

\$ sudo su hduser

\$ hdfs namenode -format

4) Start all Hadoop daemons

\$ cd /usr/local/hadoop

Start hdfs daemons

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

Start MapReduce daemons:

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

Instead both of these above command you can also use *start-all.sh*, but its now deprecated so its not recommended to be used for better Hadoop operations.

5) Track/Monitor/Verify

\$ cd

Verify Hadoop daemons:

```
$ jps
```

```
hduser@pingax: /home/vignesh
hduser@pingax:/home/vignesh$ start-dfs.sh
15/01/06 22:45:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-pingax.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-pingax.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-pingax.out
15/01/06 22:45:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hduser@pingax:/home/vignesh$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-pingax.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-pingax.out
hduser@pingax:/home/vignesh$ jps
3472 Jps
3067 SecondaryNameNode
2902 DataNode
3282 ResourceManager
3410 NodeManager
2775 NameNode
hduser@pingax:/home/vignesh$
```

Monitor Hadoop ResourceManage and Hadoop NameNode

If you wish to track Hadoop MapReduce as well as HDFS, you can try exploring Hadoop web view of ResourceManager and NameNode which are usually used by hadoop administrators. Open your default browser and visit to the following links.

For ResourceManager – <http://localhost:8088>

Browser tabs: Namenode information x All Applications x

Address bar: localhost:8088/cluster

Logged in as: drwho

hadoop

All Applications

Cluster

- About
- Nodes
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	8 GB	0 B	0	8	0	1	0	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
No data available in table										

Showing 0 to 0 of 0 entries

First Previous Next Last

For NameNode – <http://localhost:50070>

Browser tabs: Namenode information x

Address bar: localhost:50070/dfshealth.html#tab-overview

Navigation: Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Overview localhost:9000* (active)

Started:	Sun Jan 04 02:38:40 EST 2015
Version:	2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)
Cluster ID:	CID-bf5f6452-118e-4564-baab-4628cd4cdf2c
Block Pool ID:	BP-401427206-127.0.1.1-1419919134460

Summary

Security is off.
 Safemode is off.
 51 files and directories, 29 blocks = 80 total filesystem object(s).
 Heap Memory used 76.49 MB of 137.5 MB Heap Memory. Max Heap Memory is 889 MB.
 Non Heap Memory used 34.25 MB of 35.63 MB Committed Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	240.18 GB
DFS Used:	1.36 MB
Non DFS Used:	21.73 GB
DFS Remaining:	218.45 GB
DFS Used%:	0%
DFS Remaining%:	90.95%
Block Pool Used:	1.36 MB
Block Pool Used%:	0%

Exp 3:

Implementing the following file management tasks in hadoop:

1. Adding Files and Directories to HDFS

Before you can run Hadoop programs on data stored in HDFS, you'll need to put the data into HDFS first. Let's create a directory and put a file in it. HDFS has a default working directory of /user/\$USER, where \$USER is your login user name. This directory isn't automatically created for you, though, so let's create it with the mkdir command. For the purpose of illustration, we use chuck. You should substitute your user name in the example commands.

```
hadoop fs -mkdir /user/chuck
```

Hadoop's mkdir command automatically creates parent directories if they don't already exist. Now that we have a working directory, we can put a file into it. Create some text file on your local filesystem called example.txt. The Hadoop command put is used to copy files from the local system into HDFS.

```
hadoop fs -put example.txt
```

Note the period (.) as the last argument in the command above. It means that we're putting the file into the default working directory. The command above is equivalent to:

```
hadoop fs -put example.txt /user/chuck
```

2. Retrieving Files from HDFS

The Hadoop command get copies files from HDFS back to the local filesystem. To retrieve example.txt, we can run the following command:

```
hadoop fs -get example.txt
```

Another way to access the data is to display it. The Hadoop cat command allows us to do that:

```
hadoop fs -cat example.txt
```

3. Deleting Files from HDFS

You shouldn't be too surprised by now that the Hadoop command for removing files is rm:

```
hadoop fs -rm example.txt
```

The rm command can also be used to delete empty directories.

Exp 4:

Aim:-Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.

Program:-

```
import java.io.IOException; import
java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;import
org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;import
org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);private Text word
        = new Text();

        public void map(Object key, Text value, Context context
            ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());while
            (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer
        extends Reducer<Text,IntWritable,Text,IntWritable> {private
        IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
            Context context
            ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {sum +=
                val.get();
            }
            result.set(sum); context.write(key,
            result);
        }
    }

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

```

        conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

```

Create a folder on desktop as inputdata and place data in test.txt as a file.

Create a folder on desktop as wordcountf and place a program with WordCount.java

open terminal :

these commands must run after starting hadoop nodes..cd

/usr/local/hadoop

bin/hdfs dfs -put '/home/oslab/Desktop/inputdata' /usercd

'/home/oslab/Desktop/wordcountf'

compiling :

```

sudo javac -classpath /usr/local/hadoop/share/hadoop/common/hadoop-common-
2.6.0.jar:/usr/local/hadoop/share/hadoop/common/lib/hadoop-annotations-
2.6.0.jar:/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.6.0.jar
-d /home/oslab/Desktop/wordcountf
*.java

```

create a folder in wordcountf as wordcountc, and move all class files to wordcountc folder

running : open

terminal :

```

sudo jar -cvf wordcountj.jar -C
/home/oslab/Desktop/wordcountf/wordcountc .cd

```

/usr/local/hadoop

```

bin/hadoop jar /home/oslab/Desktop/wordcountf/wordcountj.jar WordCount
/user/inputdata/ outputwcopen

```

browser :

<http://localhost:50070/explorer.html#/>

goto utilities tab on that page, and select the browse the file system and type

`/user/hduser/outputwc` in search bar .

Select `part-r-00000` and download it ,you will get output of a program.

OUTPUT:-

Example:

Input:

```
Hello I am BDA
```

```
Hello I am an Intern
```

Output:

```
BDA 1
```

```
Hello 2
```

```
I 2
```

```
Intern 1
```

```
am 2
```

```
an 1
```

Exp 5:

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

Program:-

```
// importing Libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;

public class MyMaxMin {

    // Mapper

    /*MaxTemperatureMapper class is static
    * and extends Mapper abstract class
    * having four Hadoop generics type
    * LongWritable, Text, Text, Text.
    */

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

        /**
        * @method map
        * This method takes the input as a text data type.
        * Now leaving the first five tokens, it takes
        * 6th token is taken as temp_max and
        * 7th token is taken as temp_min. Now
        * temp_max > 30 and temp_min < 15 are
        * passed to the reducer.
        */

        // the data in our data set with
        // this value is inconsistent data
        public static final int MISSING = 9999;

        @Override
        public void map(LongWritable arg0, Text Value, Context context)
            throws IOException, InterruptedException {
```

```

// Convert the single row(Record) to
// String and store it in String
// variable name line

String line = Value.toString();

// Check for the empty line
if (!(line.length() == 0)) {

    // from character 6 to 14 we have
    // the date in our dataset
    String date = line.substring(6, 14);

    // similarly we have taken the maximum
    // temperature from 39 to 45 characters
    float temp_Max = Float.parseFloat(line.substring(39, 45).trim());

    // similarly we have taken the minimum
    // temperature from 47 to 53 characters

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

    // if maximum temperature is
    // greater than 30, it is a hot day
    if (temp_Max > 30.0) {

        // Hot day
        context.write(new Text("The Day is Hot Day :" + date),
                       new
Text(String.valueOf(temp_Max)));
    }

    // if the minimum temperature is
    // less than 15, it is a cold day
    if (temp_Min < 15) {

        // Cold day
        context.write(new Text("The Day is Cold Day :" + date),
                       new Text(String.valueOf(temp_Min)));
    }
}

}

}

// Reducer

/*MaxTemperatureReducer class is static
and extends Reducer abstract class
having four Hadoop generics type
Text, Text, Text, Text.
*/

public static class MaxTemperatureReducer extends

```

```

        Reducer<Text, Text, Text, Text> {

/**
 * @method reduce
 * This method takes the input as key and
 * list of values pair from the mapper,
 * it does aggregation based on keys and
 * produces the final context.
 */

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

        // putting all the values in
        // temperature variable of type String
        String temperature = Values.next().toString();
        context.write(Key, new Text(temperature));
    }

}

/**
 * @method main
 * This method is used for setting
 * all the configuration properties.
 * It acts as a driver for map-reduce
 * code.
 */

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

        // reads the default configuration of the
        // cluster from the configuration XML files
        Configuration conf = new Configuration();

        // Initializing the job with the
        // default configuration of the cluster
        Job job = new Job(conf, "weather example");

        // Assigning the driver class name
        job.setJarByClass(MyMaxMin.class);

        // Key type coming out of mapper
        job.setMapOutputKeyClass(Text.class);

        // value type coming out of mapper
        job.setMapOutputValueClass(Text.class);

        // Defining the mapper class name
        job.setMapperClass(MaxTemperatureMapper.class);

        // Defining the reducer class name

```



```

        job.setReducerClass(MaxTemperatureReducer.class);

        // Defining input Format class which is
        // responsible to parse the dataset
        // into a key value pair
        job.setInputFormatClass(TextInputFormat.class);

        // Defining output Format class which is
        // responsible to parse the dataset
        // into a key value pair
        job.setOutputFormatClass(TextOutputFormat.class);

        // setting the second argument
        // as a path in a path variable
        Path outputPath = new Path(args[1]);

        // Configuring the input path
        // from the filesystem into the job
        FileInputFormat.addInputPath(job, new Path(args[0]));

        // Configuring the output path from
        // the filesystem into the job
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        // deleting the context path automatically
        // from hdfs so that we don't have
        // to delete it explicitly
        outputPath.getFileSystem(conf).delete(outputPath);

        // exiting the job only if the
        // flag value becomes false
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

```

Commands to run:-

- start-dfs.sh
- start-yarn.sh
- hdfs dfs -put /home/dikshant/Downloads/CRND0103-2020-AK_Fairbanks_11_NE.txt /
- hdfs dfs -ls /
- hadoop jar /home/dikshant/Documents/Project.jar /CRND0103-2020-AK_Fairbanks_11_NE.txt /MyOutput

OUTPUT:-

```

year = 2020
month = 01
Date = 01

```

Experiment 6: Week 7:

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

Hint: Use an adjacency list to model a graph, and for each node store the distance from the original node, as well as a back pointer to the original node. Use the mappers to propagate the distance to the original node, and the reducer to restore the state of the graph. Iterate until the target node has been reached

Programe:-

```
import java.util.*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapred.*;

public class ShortestPath {

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

        public void map(LongWritable key, Text value, OutputCollector<IntWritable, Text> output, Reporter reporter)
        throws IOException {

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

            int id = Integer.parseInt(line[0]);

            String neighbors = "";

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

                neighbors += line[i] + " ";

            }

            output.collect(new IntWritable(id), new Text(neighbors.trim()));

        }

    }

    public static class Reduce extends MapReduceBase implements Reducer<IntWritable, Text, IntWritable, Text> {

        public void reduce(IntWritable key, Iterator<Text> values, OutputCollector<IntWritable, Text> output, Reporter
        reporter) throws IOException {

            Set<Integer> visited = new HashSet<>();

            Queue<Integer> queue = new LinkedList<>();

            queue.offer(key.get());

            visited.add(key.get());
```

```

while (!queue.isEmpty()) {

    int curNode = queue.poll();

    String curPath = "";

    int curDist = 0;

    if (visited.contains(curNode)) {

        continue;

    }

    visited.add(curNode);

    while (values.hasNext()) {

        String[] line = values.next().toString().split(" ");

        int neighbor = Integer.parseInt(line[0]);

        String path = line[1];

        int dist = path.split(" ").length;

        if (neighbor == curNode) {

            curPath = path;

            curDist = dist;

            break;

        }

    }

    output.collect(new IntWritable(curNode), new Text(curPath));

    if (curNode == target) {

        output.collect(new IntWritable(target), new Text(curPath));

        return;

    }

    for (int neighbor : curPath) {

        if (!visited.contains(neighbor)) {

            visited.add(neighbor);

            output.collect(new IntWritable(neighbor), new Text(curPath));

```

```

        queue.offer(neighbor);
    }
}
}
}
}

public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(ShortestPath.class);
    conf.setJobName("ShortestPath");
    conf.setOutputKeyClass(IntWritable.class);
    conf.setOutputValueClass(Text.class);
    conf.setMapperClass(Map.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
    JobClient.runJob(conf);
}
}

```

INPUT:

1 2 3

2 1 3 4

3 1 2 4 5

4 2 3 5

5 3 4 6

6 5

OUTPUT:

1

1 2

1 3

1 2 4

1 3 5

1 3 5 6

***Explanation:** The output shows the shortest path from node 1 to every other node in the graph. To find the shortest path from 1 to 6, we look at the entry for node 6, which shows the path 1 3 5 6. Therefore, the shortest path from 1 to 6 .

Experiment 7: Week 8:

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

Hint: Two MapReduce jobs are required to calculate the FoFs for each user in a social network. The first job calculates the common friends for each user, and the second job sorts the common friends by the number of connections to your friends.

Program:-

```
import java.io.IOException;

import java.util.*;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

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

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

public class FriendsOfFriends {

    // Job 1

    public static class Map1 extends Mapper<LongWritable, Text, IntWritable, IntArrayWritable> {

        // map function

    }

    public static class Reduce1 extends Reducer<IntWritable, IntArrayWritable, IntWritable, IntArrayWritable> {

        // reduce function

    }

    // Job 2

    public static class Map2 extends Mapper<LongWritable, Text, IntWritable, Text> {

        // map function

    }

    public static class Reduce2 extends Reducer<IntWritable, Text, IntWritable, Text> {
```

```
// reduce function
```

```
public void reduce(IntWritable key, Iterable<Text> values, Context context) {
```

```
    Map<Integer, Integer> fofs = new HashMap<>();
```

```
    for (Text value : values) {
```

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

```
        for (String part : parts) {
```

```
            String[] subParts = part.split(":");
```

```
            int friend = Integer.parseInt(subParts[0]);
```

```
            int mutual = Integer.parseInt(subParts[1]);
```

```
            if (!fofs.containsKey(friend)) {
```

```
                fofs.put(friend, 0);
```

```
            }
```

```
            fofs.put(friend, fofs.get(friend) + mutual);
```

```
        }
```

```
    }
```

```
    List<Map.Entry<Integer, Integer>> sortedFriends = new ArrayList<>(fofs.entrySet());
```

```
    sortedFriends.sort((a, b) -> b.getValue().compareTo(a.getValue()));
```

```
    StringBuilder sb = new StringBuilder();
```

```
    for (Map.Entry<Integer, Integer> entry : sortedFriends) {
```

```
        sb.append(entry.getKey()).append(":").append(entry.getValue()).append(",");
```

```
    }
```

```
    if (sb.length() > 0) {
```

```
        sb.deleteCharAt(sb.length() - 1);
```

```
    }
```

```
    try {
```

```
        context.write(key, new Text(sb.toString()));
```

```
    } catch (IOException | InterruptedException e) {
```

```

        e.printStackTrace();
    }
}

// Driver code

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

    // Job 1 configuration

    // Job 2 configuration

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

}
}

```

INPUT:

2,3,4

1,3

1,2,4

1,3,5

4

OUTPUT:

3:2, 2:1, 5:1, 4:1

1:2, 4:1, 5:1

4:2, 5:1

1:2, 3:2, 2:1, 5:1

4:2, 3:1, 2:1

***Explanation:**

This output represents the Friends-of-Friends (FoFs) for each user in the input graph. For example, the FoFs for user 1 are: user 3 (with two mutual friends), user 2 (with one mutual friend), user 5 (with one mutual friend), and user 4 (with one mutual friend). The output is sorted in descending order of the number of mutual friends, so the first entry for each user is their most popular FoF.

Experiment 8: Week 9:

Aim:-Implement an iterative PageRank graph algorithm in MapReduce.

Hint: PageRank can be implemented by iterating a MapReduce job until the graph has converged. The mappers are responsible for propagating node PageRank values to their adjacent nodes, and the reducers are responsible for calculating new PageRank values for each node, and for re-creating the original graph with the updated PageRank values.

Program:-

```
public class PageRank {

    private static final float DAMPING_FACTOR = 0.85f;

    private static final int NUM_ITERATIONS = 10;

    // Job 1

    public static class Map1 extends Mapper<LongWritable, Text, IntWritable, Text> {

        // map function

    }

    public static class Reduce1 extends Reducer<IntWritable, Text, IntWritable, Text> {

        // reduce function

    }

    // Job 2

    public static class Map2 extends Mapper<LongWritable, Text, IntWritable, FloatWritable> {

        // map function

    }

    public static class Reduce2 extends Reducer<IntWritable, FloatWritable, IntWritable, Text> {

        // reduce function

    }

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

        Configuration conf1 = new Configuration();
```

```

Job job1 = Job.getInstance(conf1, "PageRank Job 1");
job1.setJarByClass(PageRank.class);
job1.setMapperClass(Map1.class);
job1.setReducerClass(Reduce1.class);
job1.setOutputKeyClass(IntWritable.class);
job1.setOutputValueClass(Text.class);
FileInputFormat.addInputPath(job1, new Path(args[0]));
FileOutputFormat.setOutputPath(job1, new Path(args[1] + "/iter0"));
job1.waitForCompletion(true);
for (int i = 0; i < NUM_ITERATIONS; i++) {
    Configuration conf2 = new Configuration();
    Job job2 = Job.getInstance(conf2, "PageRank Job 2");
    job2.setJarByClass(PageRank.class);
    job2.setMapperClass(Map2.class);
    job2.setReducerClass(Reduce2.class);
    job2.setOutputKeyClass(IntWritable.class);
    job2.setOutputValueClass(FloatWritable.class);
    FileInputFormat.addInputPath(job2, new Path(args[1] + "/iter" + i));
    FileOutputFormat.setOutputPath(job2, new Path(args[1] + "/iter" + (i + 1)));
    job2.waitForCompletion(true);
}
}
}

```

INPUT:

The input is an adjacency list of the graph. Here's an example input with 4 nodes

1 2 3

2 1 3 4

3 2

4 2 3

OUTPUT:

0.266

0.464

0.149

0.121

This represents the final PageRank values for nodes 1, 2, 3, and 4, respectively. Node 2 has the highest PageRank value because it has many incoming links from other nodes.

Experiment 9: Week 10:

Aim:-Perform an efficient semi-join in MapReduce.

Hint: Perform a semi-join by having the mappers load a Bloom filter from the Distributed Cache, and then filter results from the actual MapReduce data source by performing membership queries against the Bloom filter to determine which data source records should be emitted to the reducers.

Program:-

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class SemiJoin {
```

```

public static class Map1 extends Mapper<Object, Text, Text, Text> {

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

        String[] fields = value.toString().split(",");

        String joinKey = fields[0];

        String record = fields[1] + "," + fields[2];

        context.write(new Text(joinKey), new Text("1," + record));

    }

}

```

```

public static class Map2 extends Mapper<Object, Text, Text, Text> {

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

        String[] fields = value.toString().split(",");

        String joinKey = fields[0];

        String record = fields[1];

        context.write(new Text(joinKey), new Text("2," + record));

    }

}

```

```

public static class Reduce extends Reducer<Text, Text, Text, Text> {

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

        String firstRecord = null;

        for (Text value : values) {

            String[] fields = value.toString().split(",");

            String dataset = fields[0];

            String record = fields[1];

            if (dataset.equals("1")) {

                firstRecord = record;
            }
        }
    }
}

```

```

    } else if (dataset.equals("2")) {
        if (firstRecord != null) {
            context.write(key, new Text(firstRecord + "," + record));
        }
    }
}

}

}

}

}

}

}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "SemiJoin");
    job.setJarByClass(SemiJoin.class);
    job.setReducerClass(Reduce.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    FileInputFormat.setInputPaths(job, new Path(args[0]), new Path(args[1]));
    FileOutputFormat.setOutputPath(job, new Path(args[2]));
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    job.setMapperClass(Map1.class);
    job.setMapperClass(Map2.class);
    job.waitForCompletion(true);
}
}

```

INPUT:

File 1 (dataset1.csv):

1,John,30

2,Jane,25

3,Bob,40

File 2 (dataset2.csv):

2,Marketing

3,Sales

4,Engineering

OUTPUT:

2,Jane,25,Marketing

3,Bob,40,Sales

***Explanation:**

The program performs a semi-join operation based on the first field (join key) in both input datasets. The output contains records that have a match in both datasets, with the fields from the first dataset followed by the fields from the second dataset. In this example, record with join key 2 from dataset1.csv matches the record with join key 2 from dataset2.csv, so it appears in the output. Similarly, record with join key 3 from dataset1.csv matches the record with join key 3 from dataset2.csv, so it also appears in the output.

Experiment 10:

PIG LATIN LANGUAGE - PIG

OBJECTIVE:

1. Installation of PIG.

RESOURCES:

VMWare, Web browser, 4 GB RAM, Hard Disk 80 GB.

PROGRAM LOGIC:

STEPS FOR INSTALLING APACHE PIG

- 1) Extract the pig-0.15.0.tar.gz and move to home directory

- 2) Set the environment of PIG in bashrc file.

- 3) Pig can run in two modes

Local Mode and Hadoop Mode

Pig -x local and pig

- 4) Grunt Shell

Grunt >

- 5) LOADING Data into Grunt Shell

DATA = LOAD <CLASSPATH> USING PigStorage(DELIMITER) as (ATTRIBUTE :
DataType1, ATTRIBUTE : DataType2.....)

- 6) Describe Data

Describe DATA;

- 7) DUMP Data

Dump DATA;

INPUT/OUTPUT:

Input as Website Click Count Data

```
lendi@ubuntu: ~  
grunt> ad1 = load '/home/lendi/Desktop/static_data/ad_data/ad_data1.txt' using PigStorage('\t') as (item:chararray,campaignId:chararray,date:chararray,time:chararray,display_site:chararray,was_clicked:int,cpc:int,country:chararray,placement:chararray);  
2016-10-14 02:35:32,441 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum  
2016-10-14 02:35:32,441 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS  
grunt> describe ad1;  
ad1: {item: chararray,campaignId: chararray,date: chararray,time: chararray,display_site: chararray,was_clicked: int,cpc: int,country: chararray,placement: chararray}  
grunt> ad2 = load '/home/lendi/Desktop/static_data/ad_data/ad_data2.txt' using PigStorage(',') as (campaignId:chararray,date:chararray,time:chararray,display_site:chararray,placement:chararray,was_clicked:int,cpc:int,item:chararray);  
2016-10-14 02:36:08,732 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum  
2016-10-14 02:36:08,732 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS  
grunt> describe ad2;  
ad2: {campaignId: chararray,date: chararray,time: chararray,display_site: chararray,placement: chararray,was_clicked: int,cpc: int,item: chararray}  
grunt>
```

PRE-LAB VIVA QUESTIONS:

- 1) What do you mean by a bag in Pig?
- 2) Differentiate between PigLatin and HiveQL
- 3) How will you merge the contents of two or more relations and divide a single relation into two or more relations?

LAB ASSIGNMENT:

1. Process baseball data using Apache Pig.

POST-LAB VIVA QUESTIONS:

1. What is the usage of foreach operation in Pig scripts?
2. What does Flatten do in Pig

PIG COMMANDS:

OBJECTIVE:

Write Pig Latin scripts sort, group, join, project, and filter your data.

RESOURCES:

VMWare, Web browser, 4 GB RAM, Hard Disk 80 GB.

PROGRAM LOGIC:

FILTER Data

FDATA = FILTER DATA by ATTRIBUTE = VALUE;

GROUP Data

GDATA = GROUP DATA by ATTRIBUTE;

Iterating Data

FOR_DATA = FOREACH DATA GENERATE GROUP AS GROUP_FUN,
ATTRIBUTE = <VALUE>

Sorting Data

SORT_DATA = ORDER DATA BY ATTRIBUTE WITH CONDITION;


LIMIT Data

LIMIT_DATA = LIMIT DATA COUNT;

JOIN Data

JOIN DATA1 BY (ATTRIBUTE1,ATTRIBUTE2....) , DATA2 BY
(ATTRIBUTE3,ATTRIBUTE....N)

INPUT / OUTPUT :

A screenshot of a terminal window with a dark background. The window title is 'lendi@ubuntu: ~'. The terminal shows the following commands and output:

```
grunt> join_data = join ad1 by (campaignId,display_site,cpc),ad2 by (campaignId,display_site,cpc);
grunt> describe join_data;
join_data: {ad1::item: chararray,ad1::campaignId: chararray,ad1::date: chararray,ad1::time: chararray,ad1::display_site: chararray,ad1::was_clicked: int,ad1::cpc: int,ad1::country: chararray,ad1::placement: chararray,ad2::campaignId: chararray,ad2::date: chararray,ad2::time: chararray,ad2::display_site: chararray,ad2::placement: chararray,ad2::was_clicked: int,ad2::cpc: int,ad2::item: chararray}
grunt> 
```

PRE-LAB VIVA QUESTIONS:

1. How will you merge the contents of two or more relations and divide a single relation into two or more relations?
2. What is the usage of foreach operation in Pig scripts?
3. What does Flatten do in Pig?

LAB ASSIGNMENT:

1. Using Apache Pig to develop User Defined Functions for student data.

PRE-LAB VIVA QUESTIONS:

1. What do you mean by a bag in Pig?
2. Differentiate between PigLatin and HiveQL

Experiment 11:-

Aim:-PIG LATIN

MODES,PROGRAMS:

OBJECTIVE:

- a. Run the Pig Latin Scripts to find Word Count.
- b. Run the Pig Latin Scripts to find a max temp for each and every year.

RESOURCES:

VMWare, Web Browser, 4 GB RAM, 80 GB Hard Disk.

PROGRAM LOGIC:

Run the Pig Latin Scripts to find Word Count.

```
lines = LOAD '/user/hadoop/HDFS_File.txt' AS (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
grouped = GROUP words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
DUMP wordcount;
```

Run the Pig Latin Scripts to find a max temp for each and every year

```
-- max_temp.pig: Finds the maximum temperature by year
records = LOAD 'input/ncdc/micro-tab/sample.txt'
AS (year:chararray, temperature:int, quality:int);
filtered_records = FILTER records BY temperature != 9999 AND
(quality == 0 OR quality == 1 OR quality == 4 OR quality == 5 OR quality == 9);
grouped_records = GROUP filtered_records BY year;
max_temp = FOREACH grouped_records GENERATE group,
MAX(filtered_records.temperature);
DUMP max_temp;
```

INPUT / OUTPUT:

```
(1950,0,1)
(1950,22,1)
(1950,-11,1)
(1949,111,1)
(1949,78,1)
```

PRE-LAB VIVA QUESTIONS:

1. List out the benefits of Pig?
2. Classify Pig Latin commands in Pig?

LAB ASSIGNMENT:

1. Analyzing average stock price from the stock data using Apache Pig

POST-LAB VIVA QUESTIONS:

1. Discuss the modes of Pig scripts?
2. Explain the Pig Latin application flow?

HIVE

OBJECTIVE:

Installation of HIVE.

RESOURCES:

VMWare, Web Browser, 1GB RAM, Hard Disk 80 GB.

PROGRAM LOGIC:

Install MySQL-Server

- 1) 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
Hivetar 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.

INPUT/OUTPUT:

```
administrator@ubuntu: ~  
d yet. Please use TIMESTAMP instead  
hive> create table log_data(l_date string,l_time string,s_sitename string,s_comput  
ername string,l_uri string,uri_query string,ip_address string,user_agent string,  
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
```

PRE-LAB VIVA QUESTIONS:

1. In Hive, explain the term „aggregation“ and its uses?
2. List out the Data types in Hive?

LAB ASSIGNMENT:

1. Analyze twitter data using Apache Hive.

POST-LAB VIVA QUESTIONS:

1. Explain the Built-in Functions in Hive?
2. Describe the various Hive Data types?

HIVE OPERATIONS

OBJECTIVE:

Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.

RESOURCES:

VMWare, XAMPP Server, Web Browser, 1GB RAM, Hard Disk 80 GB.

PROGRAM LOGIC:

SYNTAX for HIVE Database Operations

DATABASE Creation

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

Drop Database Statement

DROP DATABASE Statement DROP (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_ipadd
ress_result;
SET
hive.input.format=org.apache.hadoop.hive.ql.index.compact.HiveCompactIndexInputFor
mat;
```

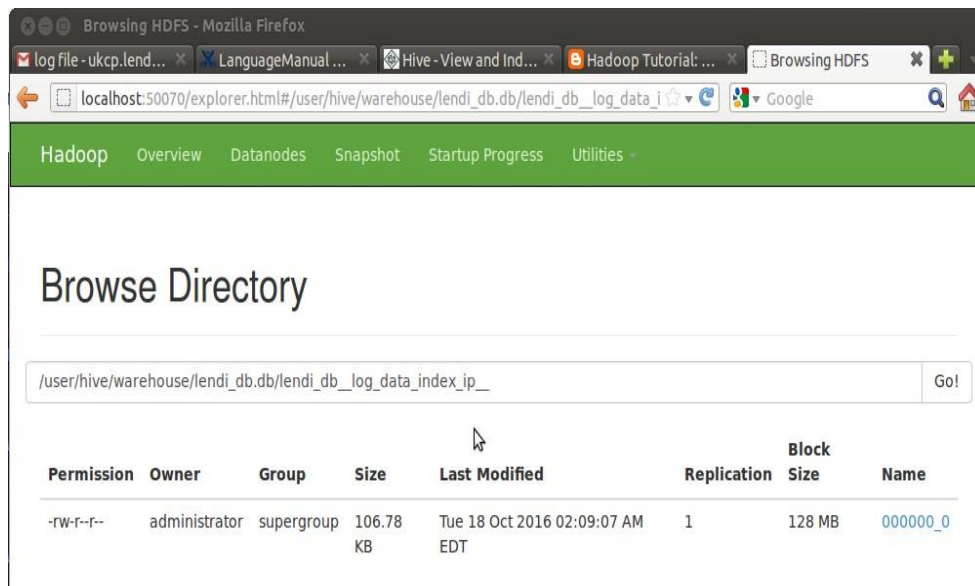
Dropping Index

```
DROP INDEX INDEX_NAME on TABLE_NAME;
```


INPUT/OUTPUT:

```
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/elastic/images/small/pic3.jpg  
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  
10 497 0  
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/css/demo.css - 10.  
Mozilla/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/elastic/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/elastic/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/elastic/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/elastic/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/elastic/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/elastic/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
```



PRE-LAB VIVA QUESTIONS:

1. How many types of joins are there in Pig Latin with an examples?
2. Write the Hive command to create a table with four columns: First name, last name, age, and income?

LAB ASSIGNMENT:

1. Analyze stock data using Apache Hive.

POST-LAB VIVA QUESTIONS:

1. Write a shell command in Hive to list all the files in the current directory?
2. List the collection types provided by Hive for the purpose a start-up company want to use Hive for storing its data.