

EXPERIMENT-1

AIM: Hadoop HDFS Practical

- A. HDFS Basics, Hadoop Ecosystem Tools Overview.
- B. Installing Hadoop.
- C. Copying File to Hadoop.
- D. Copy from Hadoop File system and delete file.
- E. Moving and displaying files in HDFS.
- F. Programming exercises on Hadoop

THEORY:

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

IMPLEMENTATION:

INSTALLATION OF HADOOP (CLOUDERA) IN VM

Setup Hadoop Virtual Environment:

To setup a single node cluster, on your laptop, you need following software:

* Oracle Virtual Box software to install one or more virtual machines. Get this software from <https://www.virtualbox.org/> Cloudera VM-this software acts as a virtual machine. By default, Cloudera consists of

packages of most of the cloud computing frameworks like Hadoop, Spark, Hive, Pig, etc. Download Cloudera QuickStart VM for Virtual Box platform, Go get this software, go to <http://www.cloudera.com/> Select Get Started Select Try Now Select Download Now inside QuickStarts box Select a platform as Virtual Box. Fill a form that pops up after this step and download Cloudera.. Alternatively, download it directly from here

STEP 1: Download VirtualBox from <https://www.virtualbox.org/wiki/Downloads>, install it and open

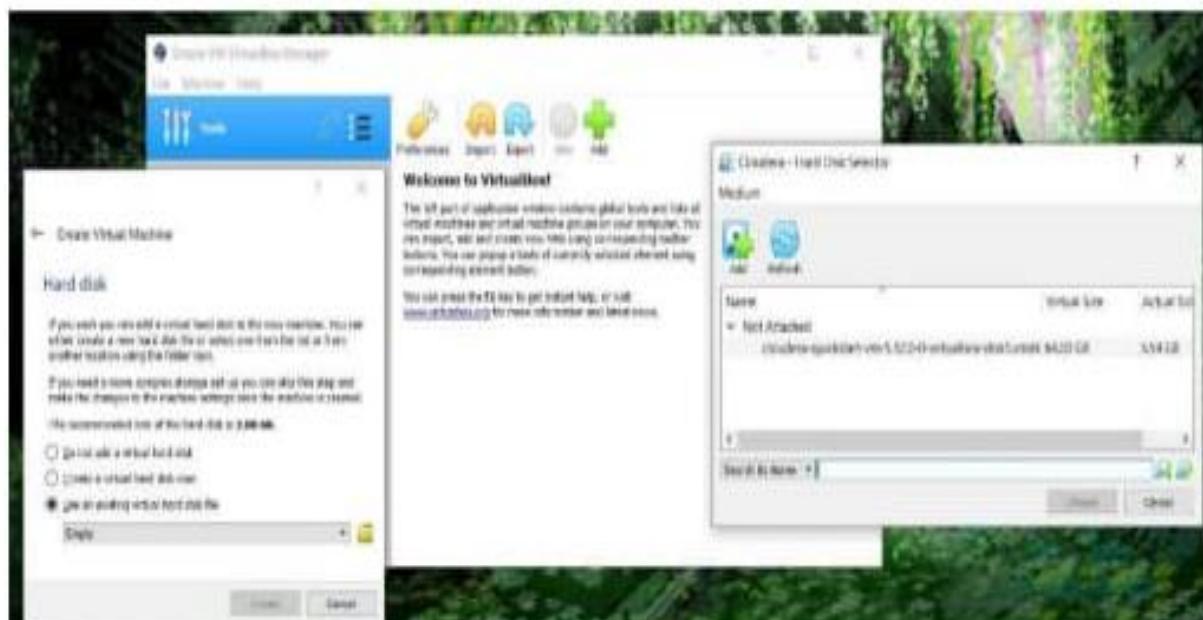
STEP 2: Create a new VM for cloudera and with the help of the following snapshots

STEP 3: Give your VM a name, TYPE: Other and VERSION: Unknown 64-bit

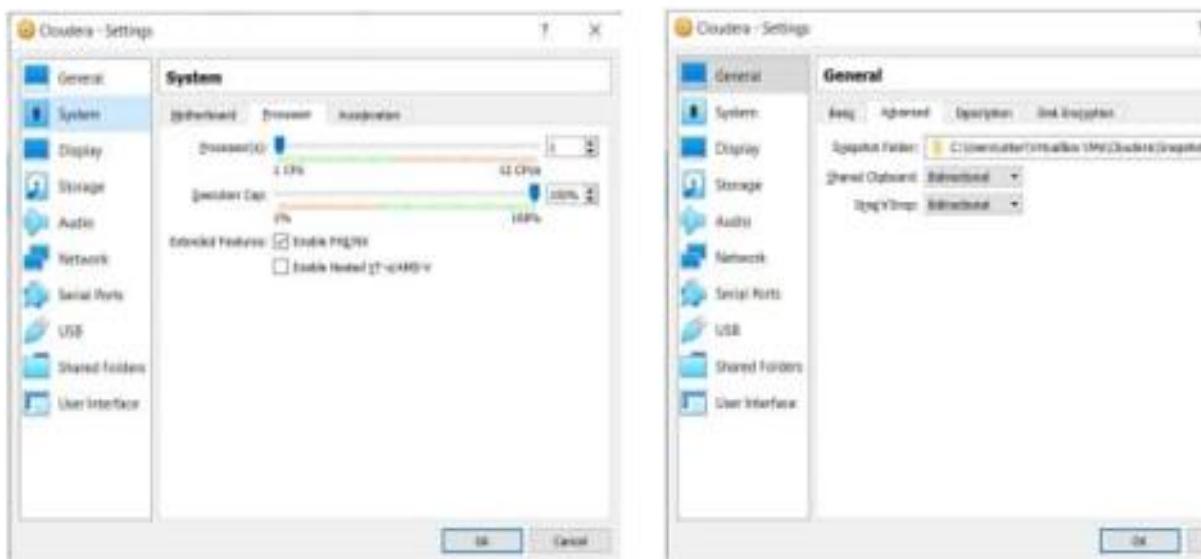
STEP 4: Give at least 4096 MB of RAM to the VM



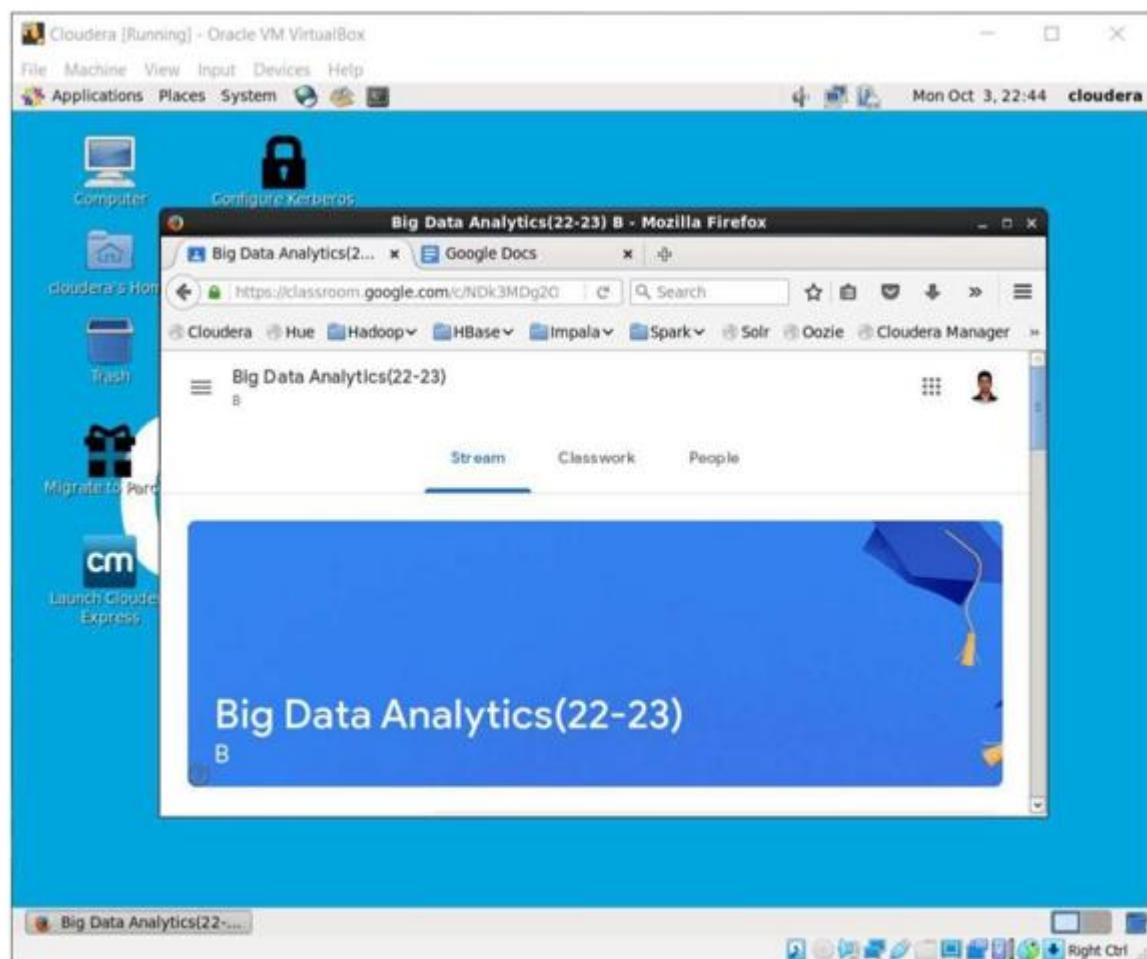
STEP 5: Select the option use an existing virtual hard disk file and click the browse link and then browse and select the downloaded vmdk file, click open and click on create.



STEP 6: Select the following settings



START the Virtual Machine and wait for all configurations to set up.



CONCLUSION:

We have successfully installed Cloudera VM on our VirtualBox and have understood the steps to do so as well.

Hadoop Commands Execution

Sr No	Description	Commands
1	Help	[cloudera@quickstart ~]\$ hdfs dfs -help ls
2	For listing of files	[cloudera@quickstart ~]\$ hdfs dfs -ls /
3	For making/creating new directory	[cloudera@quickstart ~]\$ hdfs dfs -mkdir Input
4	For listing files in root directory	[cloudera@quickstart ~]\$ hdfs dfs -ls Input
5	For listing files and directories recursively	[cloudera@quickstart ~]\$ hdfs dfs -ls -R /
6	Copying files from local system into HDFS	[cloudera@quickstart ~]\$ hdfs dfs -put /home/cloudera/Desktop/StudentInfo.txt Input/StuentInfo.txt
7	Retrieving files from HDFS – copies file from HDFS to current working directory. - Display file - Display head- first few lines of text file - Display tail – last few lines of text file	[cloudera@quickstart ~]\$ hdfs dfs -get Input/StudentInfo.txt home/cloudera/Desktop/StudentInfo1.txt [cloudera@quickstart ~]\$ hdfs dfs -cat Input/StudentInfo.txt [cloudera@quickstart ~]\$ hdfs dfs -cat Input/StudentInfo.txt head [cloudera@quickstart ~]\$ hdfs dfs -tail Input/StudentInfo.txt
8	Deleting files from HDFS	[cloudera@quickstart ~]\$ hdfs dfs -rm Input/StudentInfo.txt [cloudera@quickstart ~]\$ hdfs dfs -rm /Input

IMPLEMENTATION / EXECUTION:

The screenshot shows a Linux desktop environment with two terminal windows open. Both windows have a title bar reading "cloudera@cloudera: ~" and a status bar at the bottom showing "Mon Oct 3 23:50:28 cloudera".

The top terminal window displays the output of the command:

```
cloudera@cloudera: ~$ hdfs dfs -help ls  
Usage: hdfs dfs [-help] [-ls] [-lso] [-lsR] [-lsD] [-lsd] [-lsr] [-lsrd] [-lsRd] [-lsRrd] [-lsRdr] [-lsRdrd] [-lsRdrdR] [-lsRdrdRr] [-lsRdrdRrd] [-lsRdrdRrdR] [-lsRdrdRrdRr] [-lsRdrdRrdRrd] [-lsRdrdRrdRrdR] [-lsRdrdRrdRrdRr] [-lsRdrdRrdRrdRrd] [-lsRdrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdR] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRr] [-lsRdrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrdRrd]
```

The bottom terminal window displays the output of the command:

```
cloudera@cloudera: ~$ hdfs dfs -cat /bin/bash  
#!/bin/sh  
#  
# /etc/init.d/bash - start/stop shell service  
#  
# chkconfig: 2345 100 100  
# description: bash shell  
#  
# processname: bash  
# pidfile: /var/run/bash.pid  
#  
# source function library  
if [ -f /etc/init.d/functions ]; then . /etc/init.d/functions; fi  
#  
# check if bash is running  
if [ ! -e /var/run/bash.pid ]; then  
    echo "Starting bash shell"  
    /bin/bash > /var/run/bash.pid  
else  
    echo "bash shell already running"  
fi  
#  
# stop bash  
if [ -e /var/run/bash.pid ]; then  
    echo "Stopping bash shell"  
    kill `cat /var/run/bash.pid`  
    rm /var/run/bash.pid  
else  
    echo "bash shell not running"  
fi  
#  
# restart bash  
if [ -e /var/run/bash.pid ]; then  
    echo "Restarting bash shell"  
    kill `cat /var/run/bash.pid`  
    rm /var/run/bash.pid  
    /bin/bash > /var/run/bash.pid  
else  
    echo "Starting bash shell"  
    /bin/bash > /var/run/bash.pid  
fi  
#  
# reload bash  
if [ -e /var/run/bash.pid ]; then  
    echo "Reloading bash shell"  
    kill -HUP `cat /var/run/bash.pid`  
else  
    echo "bash shell not running"  
fi
```

CONCLUSION: We have successfully gained knowledge about various Hadoop commands and have also executed some of them.

EXPERIMENT - 3

AIM: To install and configure MongoDB/ Cassandra/ HBase/ Hypertable to execute NoSQL commands

THEORY:

HBASE =>

The objective of this lab is to get you started with the HBase shell and perform CRUD operations to create a HBase Table, put data into the table, retrieve data from the table and delete data from the table.

HBase Shell

HBase is the Hadoop database, which provides random, realtime read/write access to very large data. See the references on HBase for more information.

The HBase Shell is a ruby script that helps in interacting with the HBase system using a command line interface. This shell supports creating, deleting and altering tables and also performing other operations like inserting, listing, deleting data and to interact with HBase. You can get help with the shell commands here:

Sr No.	Description	Commands
1	Change the directory to hbase	\$ cd training_materials/hbase
2	Start Hbase Shell	\$ hbase shell
	o/p	<pre>[training@localhost hbase]\$ hbase shell 15/12/31 01:28:47 WARN conf.Configuration: hadoop.native.lib is deprecated. Instead, use io.native.lib.available HBase Shell; enter 'help<RETURN>' for list of supported commands. Type "exit<RETURN>" to leave the HBase Shell Version 0.92.1-cdh4.1.1, rUnknown, Tue Oct 16 11:44:19 PDT 2012 hbase(main):001:0></pre>
3	To Get help	hbase(main):001:0> help

o/p **COMMAND GROUPS:**

Group name: general

Commands: status, version, whoami

Group name: ddl

Commands: alter, alter_async, alter_status, create, describe, disable, disable_all, drop, drop_all, enable, enable_all, exists, is_disabled, is_enabled, list, show_filters

Group name: dml

Commands: count, delete, deleteall, get, get_counter, incr, put, scan, truncate
Group name: tools

		<p><i>Commands: assign, balance_switch, balancer, close_region, compact, flush, hlog_roll, major_compact, move, split, unassign, zk_dump</i></p> <p>Group name: replication</p> <p><i>Commands: add_peer, disable_peer, enable_peer, list_peers, remove_peer, start_replication, stop_replication</i></p> <p>Group name: security</p> <p><i>Commands: grant, revoke, user_permission</i></p>
4	Perform CRUD operations with HBase shell	The goal of this exercise is to create a table ‘ customer ’ with the data for Column Families address, order and columns –(city, state) and (date,order) for as shown below using HBase shell commands.
5	<i>Create a table in your current directory</i> <i>o/p</i>	create 'customer', {NAME=>'addr'}, {NAME=>'order'} <i>hbase(main):005:0> create 'customer','addr','order'</i> 0 row(s) in 1.0420 seconds
6	<i>Use ‘describe’ to get the description of the table.</i> <i>o/p</i>	hbase(main):006:0> describe 'customer' <i>DESCRIPTION ENABLED</i> <i>{NAME => 'customer', FAMILIES => [{NAME => 'addr', true}</i> <i>BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '0', VE</i> <i>RSIONS => '3', COMPRESSION => 'NONE',</i> <i>MIN VERSIONS => '0', TTL => '2147483647',</i> <i>BLOCKSIZE => '65536',</i> <i>IN MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'order', BLOOMFILTER => 'NONE',</i> <i>REPLICATION_SCOPE => '0', VERSIONS => '3',</i> <i>COMPRESSION => 'NONE', M</i> <i>IN VERSIONS => '0', TTL => '2147483647', BLOCKSIZE => '65536', IN MEMORY => 'false', BLOCKCACHE => 'true']}</i> 1 row(s) in 0.0360 seconds
7	Put some data into the table	<i>hbase(main):001:0> put 'customer','sujata','addr:city','chembur'</i>
8	Use get to	<i>hbase(main):002:0> get 'customer','sujata'</i>

	retrieve the data for ‘sujata’	
--	---------------------------------------	--

	<p>o/p - <u>This gets all the data for the row. How can we limit</u></p>	<p>COLUMN CELL addr:city timestamp=1451545949223, value=chembur 1 row(s) in 0.0300 seconds</p>
	<p>this to only one column family ? Ans :see step 10</p>	
9	<p>Put some more data into table</p>	<pre>hbase(main):003:0> put 'customer','sujata','order:numb', '1234' 0 row(s) in 0.0450 seconds hbase(main):004:0> put 'customer','sujata','order:numb', '1235' 0 row(s) in 0.0080 seconds hbase(main):005:0> put 'customer','sujata','order:numb', '1236' 0 row(s) in 0.0200 seconds</pre>
10	<p>Get the data about column family order for specific customer</p>	<pre>hbase(main):006:0> get 'customer','sujata',{COLUMNS=>['order:numb']}</pre>
11	<p>Ouput: NOTE This only return one record (latest according to timestamp) Note that you are getting the data for only one version per cell. How can you get more versions? Output:</p>	<p><i>COLUMN CELL order:numb timestamp=1451546824151, value=1236 1 row(s) in 0.0280 seconds</i></p> <pre>hbase(main):007:0> get 'customer','sujata', {COLUMNS=>['order:numb'], VERSIONS =>5}</pre> <p><i>COLUMN CELL order:numb timestamp=1451546824151, value=1236 order:numb timestamp=1451546805834, value=1235 order:numb timestamp=1451546787681, value=1234 3 row(s) in 0.0250 seconds</i></p>

17	Use 'count' to retrieve the number of rows in the table.	hbase(main):002:0> count 'customer'
18	o/p Delete data from the table.	hbase(main):002:0> count 'customer' <u>2 row(s) in 0.3160 seconds</u> Delete a column hbase(main):001:0> delete 'customer', 'jhon','addr:city' 0 row(s) in 0.3120 seconds
19	Delete a column family	hbase(main):001:0> delete 'customer','jhon','addr:' 0 row(s) in 0.3180 seconds
20	Retrieve rows with rowkey starting with 'sujata', addr column family	hbase(main):008:0> scan 'customer', {STARTROW=>'sujata',COLUMNS=>['addr']}
	o/p	<pre>hbase(main):008:0> scan 'customer', {STARTROW=>'sujata',COLUMNS=>['addr']} ROW COLUMN+CELL sujata column=addr:city, timestamp=1451545949223, value=chembur sujata column=addr:state, timestamp=1452547606923, value=MH\xA0 1 row(s) in 0.0130 seconds</pre>

Implementation :

```
[cloudera@quickstart Desktop]$ hbase shell
2022-08-17 07:30:50,258 INFO [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help[RETURN]' for list of supported commands.
Type "exit[RETURN]" to leave the HBase Shell
Version 1.0.0-cdh5.4.2, rUnknown, Tue May 19 17:07:29 PDT 2015

hbase(main):001:0> whoami
cloudera (auth:SIMPLE)
  groups: cloudera, default

hbase(main):002:0> version
1.0.0-cdh5.4.2, rUnknown, Tue May 19 17:07:29 PDT 2015

hbase(main):003:0> version
1.0.0-cdh5.4.2, rUnknown, Tue May 19 17:07:29 PDT 2015

hbase(main):004:0> status
1 servers, 0 dead, 5.0000 average load

hbase(main):005:0> ■
Launch Cloudera
cloudera@quickstart:...
```

```
cloudera-quickstart-vm-5.4.2-0-virtualbox [Running] - Oracle VM Virtu... - x
File Machine View Input Devices Help
Applications Places System cloudera cloudera Wed Aug 17, 7:37 AM
cloudera Change account settings and status - x
File Edit View Search Terminal Help

hbase(main):004:0> status
1 servers, 0 dead, 5.0000 average load

hbase(main):005:0> create 'genshin',{NAME=>'name'}, {NAME=>'element'}
0 row(s) in 0.2600 seconds

=> Hbase::Table -> genshin
hbase(main):006:0> describe 'genshin'
Table genshin is ENABLED
genshin
COLUMN FAMILIES DESCRIPTION
{NAME => 'element', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW',
REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSION => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'name', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN VERSION => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
2 row(s) in 0.0220 seconds

hbase(main):007:0> [ ]
```

The image contains three side-by-side screenshots of Oracle VM VirtualBox windows, each showing a terminal session on a Cloudera VM. The windows are titled "cloudera-quickstart-vm-5.4.2-0-virtualbox [Running] - Oracle VM Virtu..." and show the HBase command-line interface.

Screenshot 1 (Left):

```

cloudera@quickstart:~$ hbase(main):008:0> put 'genshin','xiao','element:anemo','liyue'
0 row(s) in 0.0888 seconds

hbase(main):009:0> put 'genshin','zhongli','element:geo','liyue'
0 row(s) in 0.0030 seconds

hbase(main):010:0> put 'genshin','yelan','element:hydro','liyue'
0 row(s) in 0.0030 seconds

hbase(main):011:0> put 'genshin','venti','element:anemo','mondstadt'
0 row(s) in 0.0038 seconds

hbase(main):012:0> put 'genshin','diluc','element:pyro','mondstadt'
0 row(s) in 0.0028 seconds

hbase(main):013:0> put 'genshin','jean','element:anemo','mondstadt'
0 row(s) in 0.0030 seconds

hbase(main):014:0> put 'genshin','shogun','element:electro','inazuma'
0 row(s) in 0.0030 seconds

hbase(main):015:0>

```

Screenshot 2 (Middle):

```

cloudera@quickstart:~$ hbase(main):015:0> get 'genshin','xiao'
COLUMN          CELL
element:anemo   timestamp=1660747157416, value=liyue
1 row(s) in 0.0400 seconds

hbase(main):016:0> get 'genshin','yelan'
COLUMN          CELL
element:hydro   timestamp=1660747187018, value=liyue
1 row(s) in 0.0050 seconds

hbase(main):017:0> get 'genshin','diluc'
COLUMN          CELL
element:pyro    timestamp=1660747215957, value=mondstadt
1 row(s) in 0.0060 seconds

hbase(main):018:0> get 'genshin','shogun'
COLUMN          CELL
element:electro timestamp=1660747286636, value=inazuma
1 row(s) in 0.0070 seconds

hbase(main):019:0>

```

Screenshot 3 (Right):

```

cloudera@quickstart:~$ hbase(main):057:0> scan 'genshin'
ROW
diluc           COLUMN+CELL
jean            column=element:pyro, timestamp=1660747215957, value=mondstadt
shogun          column=element:anemo, timestamp=1660747223279, value=mondstadt
venti            column=element:electro, timestamp=1660747286636, value=inazuma
xiao             column=element:anemo, timestamp=1660747157416, value=liyue
yelan            column=element:hydro, timestamp=1660747187018, value=liyue
zhongli          column=element:geo, timestamp=1660747172976, value=liyue
7 row(s) in 0.0110 seconds

hbase(main):058:0> scan 'genshin' {STARTROW=>'y'}
SyntaxError: (hbase):59: syntax error, unexpected tLCURLY

scan 'genshin' {STARTROW=>'y'}
^

hbase(main):060:0> scan 'genshin',{STARTROW=>'y'}
ROW
yelan           COLUMN+CELL
zhongli          column=element:hydro, timestamp=1660747187018, value=liyue
column=element:geo, timestamp=1660747172976, value=liyue
2 row(s) in 0.0050 seconds

hbase(main):061:0> count 'genshin'
7 row(s) in 0.0170 seconds

=> 7
hbase(main):062:0> delete 'genshin', 'xiao','element:anemo'
0 row(s) in 0.0130 seconds

hbase(main):063:0>

```

CONCLUSION:

In this experiment, we have understood how to configure HBase in our Hadoop VM and saw various HBase commands and have also executed them.

EXPERIMENT - 4

AIM: Experiment on Hadoop Map-Reduce: Write a program to implement a word count program using MapReduce.

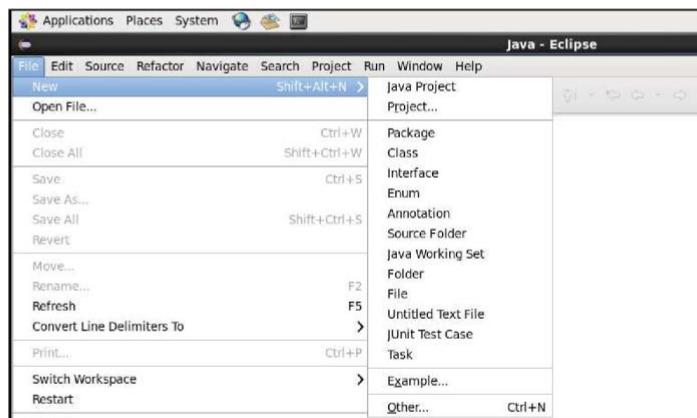
THEORY:

One of the three components of Hadoop is Map Reduce. The first component of Hadoop, that is, Hadoop Distributed File System (HDFS) is responsible for storing the file. The second component that is, Map Reduce is responsible for processing the file.

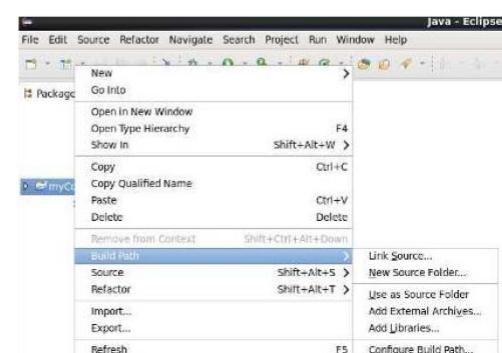
MapReduce also uses Java but it is very easy if you know the syntax on how to write it. It is the basis of MapReduce. So here are the steps which show how to write a MapReduce code for Word Count.

STEPS =>

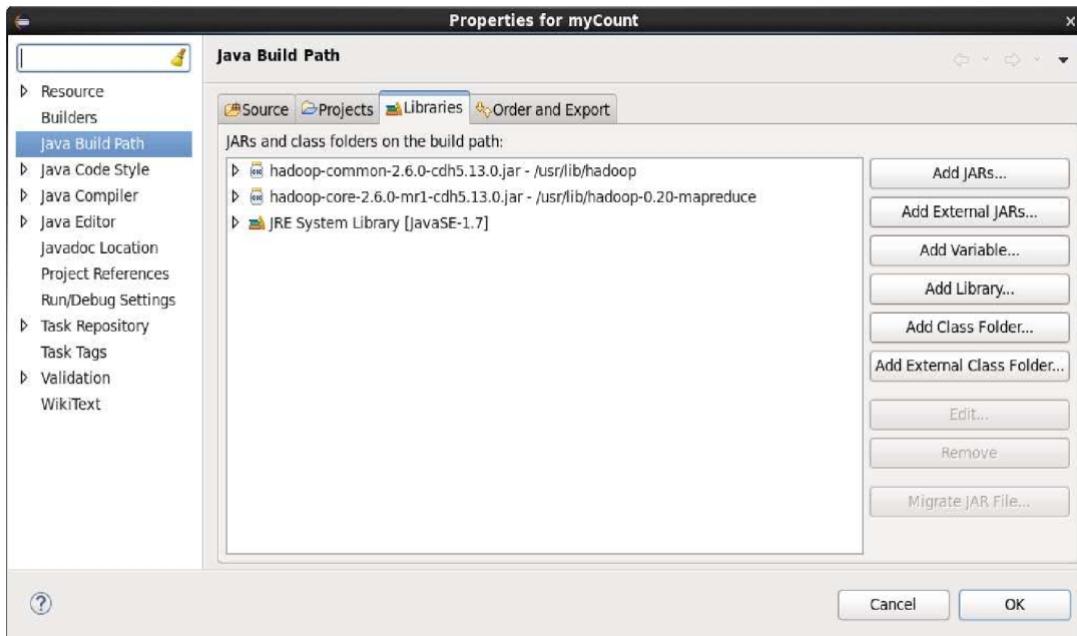
1. Open Eclipse → then select File → New → Java Project → Name it WordCount → then Finish



2. Create 3 Java Classes into the project. Name them WCDriver(having the main function), WCMapper, WCReducer.
3. You have to include two Reference Libraries for that:
Right Click on Project -> then select Build Path->
Click on Configure Build Path
4. In the figure below, you can see the Add External JARs option on the Right Hand Side. Click on it and add the below mentioned files. You can find these files in /usr/lib/



- a. /usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar
- b. /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar



Mapper Code:

```

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,Text, Text,
IntWritable> {
    public void map(LongWritable key, Text value, OutputCollector<Text,IntWritable> output,
    Reporter rep) throws IOException
    {
        String line = value.toString();
        for (String word : line.split(" ")){
            if (word.length() > 0){
                output.collect(new Text(word), new IntWritable(1));
            }
        }
    }
}

```

Reducer Code:

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

public class WCReducer extends MapReduceBase implements Reducer<Text,IntWritable, Text,
IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> value,OutputCollector<Text, IntWritable>
output,Reporter rep) throws IOException
    {
        int count = 0;
        while (value.hasNext()){
            IntWritable i = value.next();
            count += i.get();
        }
        output.collect(key, new IntWritable(count));
    }
}
```

Driver Code:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {
    public int run(String args[]) throws IOException {
        if (args.length < 2) {
            System.out.println("Please give valid inputs");
            return -1;
        }
    }
}
```

```

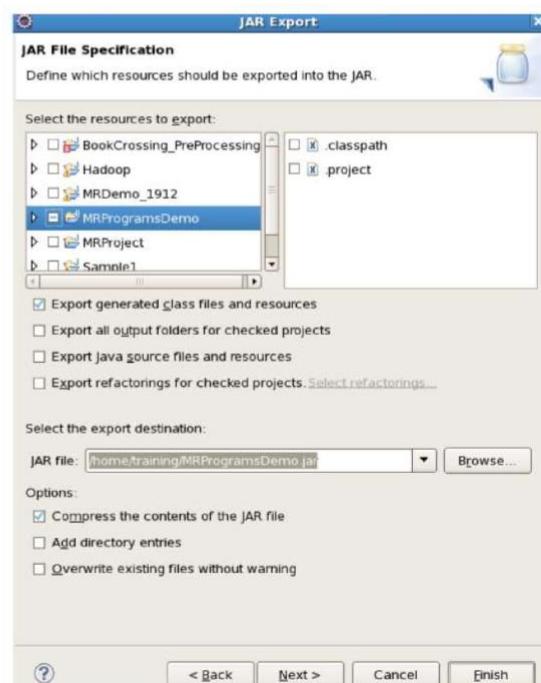
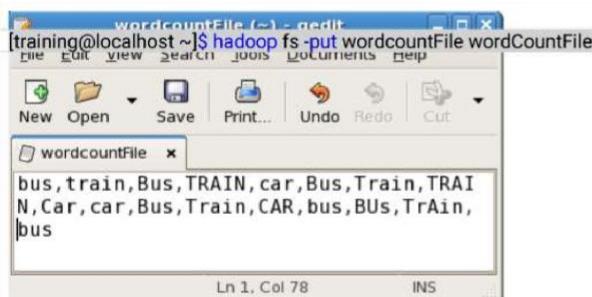
        }

    JobConf conf = new JobConf(WCDriver.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
    conf.setMapperClass(WCMapper.class);
    conf.setReducerClass(WCReducer.class);
    conf.setMapOutputKeyClass(Text.class);
    conf.setMapOutputValueClass(IntWritable.class);
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    JobClient.runJob(conf);
    return 0;
}

public static void main(String args[]) throws Exception {
    int exitCode = ToolRunner.run(new WCDriver(), args);
    System.out.println(exitCode);
}

```

5. Now you have to make a jar file. Right Click on Project-> Click on Export-> Select export destination as Jar File-> Name the jar File(WordCount.jar) -> Click on next -> at last Click on Finish. Now copy this file into the Workspace directory of Cloudera
6. Open the terminal on CDH and change the directory to the workspace. You can do this by using the “cd *workspace*” command. Now, Create a text file(WCFile.txt) and move it to HDFS. Open the terminal and write this code(remember you should be in the same directory as the jar file you have created just now).
7. Now, run this command to copy the file input file into the HDFS “**hadoop fs -put WCFile.txt WCFile.txt**”



8. Run Jar File: (Hadoopjarfilename.jarpackageName.ClassNamePathToInputTextFile PathToOutputDirectory)
9. After Executing the code, you can see the result in the WCOutput file or by writing the following command on terminal. "***hadoop fs -cat WCOutput/part-00000***"

```
[training@localhost ~]$ hadoop fs -ls MRDir1Found 3 items
-rw-r--r-- 1 trainingsupergroup          0 2016-02-23 03:36
ning/MRDir1/_SUCCESS
drwxr-xr-x  - trainingsupergroup          0 2016-02-23 03:36      /user/trai
ning/MRDir1/_logs
-rw-r--r-- 1 trainingsupergroup 20 2016-02-23 03:36      /user/trai
ning/MRDir1/part-r-00000
                                         /user/trai
```

```
[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000
BUS      7
CAR      4
TRAIN    6
```

CONCLUSION:

In this experiment, we have understood about the MapReduce in Hadoop and have also implemented the “WORD COUNT” program using it.

EXPERIMENT - 5

AIM: Experiment on Hadoop Map-Reduce: Implementing simple algorithms in Map-Reduce:
Matrix multiplication, Aggregates, Joins, Sorting, Searching, etc

THEORY:

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. As the processing component, MapReduce is the heart of Apache Hadoop. The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform.

The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

IMPLEMENTATION:

Let us consider the matrix multiplication example to visualize MapReduce. Consider the following matrix:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

taking two matrix of 2x2

2×2 matrices A and B

Here matrix A is a 2×2 matrix which means the number of rows(i)=2 and the number of columns(j)=2. Matrix B is also a 2×2 matrix where number of rows(j)=2 and number of columns(k)=2. Each cell of the matrix is labelled as A_{ij} and B_{jk} . Ex. element 3 in matrix A is called A_{21} i.e. 2nd-row 1st column. Now One step matrix multiplication has 1 mapper and 1 reducer. The Formula is:

Mapper for Matrix A (k, v)=((i, k), (A, j, A_{ij})) for all k

Mapper for Matrix B (k, v)=((i, k), (B, j, B_{jk})) for all i

Therefore computing the mapper for Matrix A:

```
# k, i, j computes the number of times it occurs.  
# Here all are 2, therefore when k=1, i can have  
# 2 values 1 & 2, each case can have 2 further  
# values of j=1 and j=2. Substituting all values  
# in formula
```

```
k=1 i=1 j=1 ((1, 1), (A, 1, 1))  
j=2 ((1, 1), (A, 2, 2))
```

```
i=2 j=1 ((2, 1), (A, 1, 3))  
j=2 ((2, 1), (A, 2, 4))
```

```
k=2 i=1 j= ((1, 2), (A, 1, 1))  
j=2 ((1, 2), (A, 2, 2))
```

```
i=2 j=1 ((2, 2), (A, 1, 3))  
j=2 ((2, 2), (A, 2, 4))
```

Computing the mapper for Matrix B

```
i=1 j=1 k=1 ((1, 1), (B, 1, 5))  
k=2 ((1, 2), (B, 1, 6))  
j=2 k=1 ((1, 1), (B, 2, 7))  
j=2 ((1, 2), (B, 2, 8))
```

```
i=2 j=1 k=1 ((2, 1), (B, 1, 5))  
k=2 ((2, 2), (B, 1, 6))  
j=2 k=1 ((2, 1), (B, 2, 7))  
k=2 ((2, 2), (B, 2, 8))
```

The formula for Reducer is:

Reducer(k, v)= (i, k) =>Make sorted Alist and Blist
 $(i, k) \Rightarrow$ Summation ($A_{ij} * B_{jk}$) for j
Output => $((i, k), \text{sum})$

Therefore computing the reducer:

```
# We can observe from Mapper computation  
# that 4 pairs are common (1, 1), (1, 2),
```

```
# (2, 1) and (2, 2)
# Make a list separate for Matrix A &
# B with adjoining values taken from
# Mapper step above:
```

(1, 1) => Alist = {(A, 1, 1), (A, 2, 2)}
Blist = {(B, 1, 5), (B, 2, 7)}
Now Aij x Bjk: [(1*5) + (2*7)] = 19 ----- (i)

(1, 2) => Alist = {(A, 1, 1), (A, 2, 2)}
Blist = {(B, 1, 6), (B, 2, 8)}
Now Aij x Bjk: [(1*6) + (2*8)] = 22 ----- (ii)

(2, 1) => Alist = {(A, 1, 3), (A, 2, 4)}
Blist = {(B, 1, 5), (B, 2, 7)}
Now Aij x Bjk: [(3*5) + (4*7)] = 43 ----- (iii)

(2, 2) => Alist = {(A, 1, 3), (A, 2, 4)}
Blist = {(B, 1, 6), (B, 2, 8)}
Now Aij x Bjk: [(3*6) + (4*8)] = 50 ----- (iv)

From (i), (ii), (iii) and (iv) we conclude that

((1, 1), 19)
((1, 2), 22)
((2, 1), 43)
((2, 2), 50)

OUTPUT:

$$\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

CONCLUSION:

In this experiment, we have understood about the MapReducer and how we can perform Matrix Multiplication using it.

EXPERIMENT - 6

AIM: Create Hive Database and Descriptive analytics-basic statistics

THEORY:

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Hadoop is an open-source framework to store and process Big Data in a distributed environment. It contains two modules, one is MapReduce and another is Hadoop Distributed File System (HDFS). The Hadoop ecosystem contains different sub-projects (tools) such as Sqoop, Pig, and Hive that are used to help Hadoop modules.

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy. Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies.

For example, Amazon uses it in Amazon Elastic MapReduce. Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage.

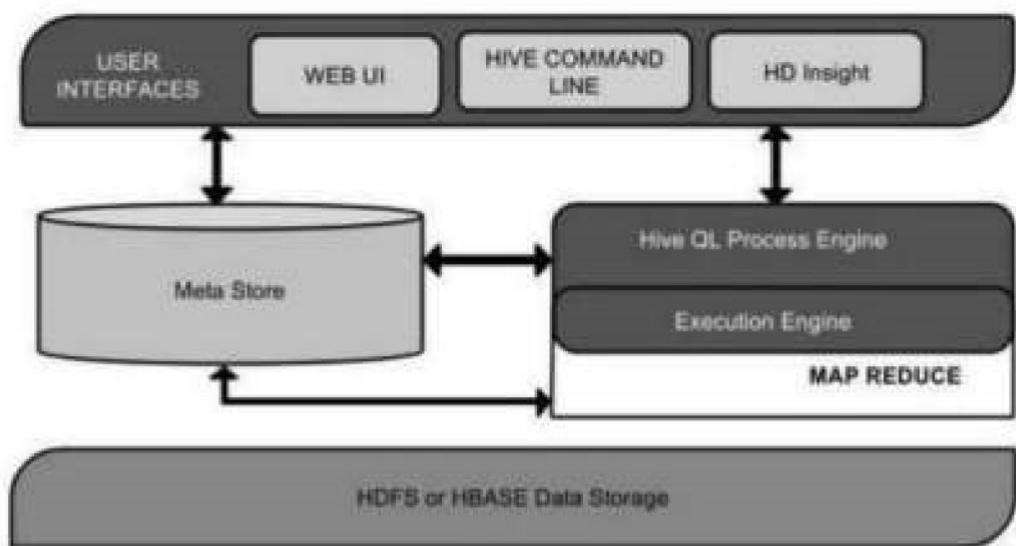
It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs. Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs.

Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Features of Hive

- It stores schema in a database and processed data into HDFS
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Architecture of Hive



Limitations of Hive commands:

1. Hive doesn't support sub queries.
2. Hive surely supports over-writing, but unfortunately, it doesn't support deletion and updates.
3. Hive is not designed for OLTP, but it is used for it.

Basic Hive Commands

1. Create: This will create the new database in the Hive.
create database database_name;
2. Show: show command will show all the databases residing in the Hive. show databases;
3. Use: The command to use the database.
use database_name;
4. Drop: The drop will remove a table from Hive
Drop table_name;
5. Create table: This command creates the table.
create table table_name (column names and types);
6. Alter: Alter command will help you rename the table or table columns. alter table table_name rename to new_table_name;
For renaming column name, replace table name with column name.
7. Describe: Describe command will help you with the information about the schema of the table.
describe table name;
8. LOAD, INSERT: The Load operation is used to move the data into corresponding Hive table.
LOAD data inpath into table [tablename];

S.No.	Description	Commands
1	Start HIVE	\$ hive

	o/p	Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
2	Create database	hive> create database retail;
	o/p	OK Time taken: 0.709 seconds
3	Show database	hive> show databases
	o/p	OK default retail Time taken: 0.284 seconds, Fetched: 2 row(s)
4	Use database	hive> use retail;
	o/p	OK Time taken: 0.087 seconds
5	Create table	hive> create table txnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING, state STRING, spendby STRING) row format delimited fields terminated by ',' stored as textfile;
	o/p	OK Time taken: 0.418 seconds
6	Describe table	hive> use retail; hive> describe txnrecords;
	o/p	OK txnno int txndate string custno int amount double category string product string state string spendby string Time taken: 0.165 seconds, Fetched: 8 row(s)
7	Create table	hive> use retail; hive> create table student(id INT, name STRING) row format delimited fields terminated by ',';
	o/p	OK Time taken: 0.081 seconds

8	Loading data	hive> load data local inpath '/home/cloudera/Desktop/BE_A.csv' overwrite into table student;
	o/p	Loading data to table retail.student Table retail.student stats: [numFiles=1, numRows=0, totalSize=26, rawDataSize=0] OK Time taken: 0.887 seconds
9	Fetching all rows	hive> select * from student;
	o/p	OK NULL name 1 XYZ
		2 ABC 3 LMN Time taken: 0.516 seconds, Fetched: 4 row(s)

OUTPUT:

The image displays three separate windows from an Oracle VM VirtualBox environment, each showing a terminal session on a Cloudera Quickstart VM. The sessions show the execution of various Hive commands to create a table, insert data from a CSV file, and perform a select query.

Screenshot 1 (Left): Shows the creation of a table named 'genshinImpact' with columns 'name', 'element', and 'level'. It then inserts three rows of data: ('xiao', 'anemo', 90), ('kazuba', 'anemo', 90), and ('zhongli', 'geo', 90). Finally, it performs a select query on the table.

```
hive> create table genshinImpact(name STRING, element STRING, level INT) row format delimited fields terminated by ',';
Time taken: 0.109 seconds
hive> describe genshinImpact;
OK
name          string
element        string
level          int
Time taken: 0.085 seconds, Fetched: 3 row(s)
hive>
```

Screenshot 2 (Middle): Shows the creation of a table named 'genshinImpact' and inserting data from a CSV file named 'charDetails.csv'. The table stats are shown as: numFiles=1, numRows=0, totalSize=135, rawDataSize=0. A select query is then run on the table.

```
hive> create table genshinImpact(name STRING, element STRING, level INT) row format delimited fields terminated by ',';
Time taken: 0.109 seconds
hive> describe genshinImpact;
OK
name          string
element        string
level          int
Time taken: 0.085 seconds, Fetched: 3 row(s)
hive> load data local inpath '/home/cloudera/Desktop/charDetails.csv' over write into table genshinImpact;
Time taken: 0.109 seconds
Table hoyoverse.genshinImpact stats: [numFiles=1, numRows=0, totalSize=135, rawDataSize=0]
hive> select * from genshinImpact;
Time taken: 0.372 seconds
xiao    anemo   90
kazuba  anemo   90
zhongli geo    90
traveller all    90
yanfei  pyro    70
yelan   hydro    90
ayato   hydro    90
ayaka   cryo    90
beita   cryo    90
NULL    NULL    NULL
Time taken: 0.17 seconds, Fetched: 10 row(s)
hive>
```

Screenshot 3 (Right): Shows the creation of a database named 'genshin', the creation of a table named 'genshinImpact' within the database, and a select query on the table.

```
hive> create database hoyoverse;
Time taken: 0.237 seconds, Fetched: 4 row(s)
hive> create database genshin;
OK
Time taken: 0.046 seconds
hive> use genshin;
OK
Time taken: 0.015 seconds
hive> show tables;
OK
Time taken: 0.043 seconds
hive>
```

CONCLUSION: We have successfully executed various Hive Queries using Hive in Cloudera

EXPERIMENT - 7

AIM: Social Network Analysis using R

THEORY:

Social Network Analysis in R, Social Network Analysis (SNA) is the process of exploring the social structure by using graph theory.

It is mainly used for measuring and analyzing the structural properties of the network.

It helps to measure social network relationships (Facebook, Twitter likes comments following etc..), Email connectivity, flows between groups, organizations, and other connected entities.

We will be using a small network that indicates a connection between the two columns' information.

Please make commonly used words in social network analysis.

A network is represented as a graph, which shows links (if any) between each vertex (or node) and its neighbors.

Edge: – A-line indicating a link between vertices.

Component: – A group of vertices that are mutually reachable by following edges on the graph.

Path: -The edges followed from one vertex to another are called a path.

IMPLEMENTATION:

Load Library: *library(igraph)*

Getting Data: *data <- read.csv("D:/RStudio/SocialNetworkAnalysis/socialnetworkdata.csv", header=T)*

The same dataset you can avail from GitHub [click here](#)

y <- data.frame(data\$first, data\$second)

Create network

```
net <- graph.data.frame(y, directed=T)
```

```
V(net)
```

```
+ 52/52 vertices, named, from 4c03f50:
```

```
[1] AA AB AF DD CD BA CB CC BC ED AE CA EB BF BB AC DC BD DB CF DF BE EA CE  
EE EF FF FD GB GC GD AD KA KF LC DA EC FA FB DE FC FE GA GE KB KC KD KE LB  
LA LD LE
```

```
E(net)
```

```
290/290 edges from 4c03f50 (vertex names):
```

```
[1] AA->DD AB->DD AF->BA DD->DA CD->EC DD->CE CD->FA CD->CC BA->AF  
CB->CA CC->CA CD->CA BC->CA DD->DA ED->AD AE->AC AB->BA CD->EC CA->CC
```

[20] EB->CC BF->CE BB->CD AC->AE CC->FB DC->BB BD->CF DB->DA DD->DA
DB->DD BC->AF CF->DE DF->BF CB->CA BE->CA EA->CA CB->CA CB->CA CC->CA

[39] CD->CA BC->CA BF->CA CE->CA AC->AD BD->BE AE->DF CB->DF AC->DF
AA->DD AA->DD AA->DD CD-

.....
[153] CA->CC CD->CC CA->CC BB->CC DF->CE CA->CE AE->GA DC->ED BB->ED
CD->ED BF->ED DF->ED CC->KB AD->EA EF->EA CF->KC EE->BB CC->BB BC->BB

[172] CD->KD AE->CF DF->AC DF->AC ED->AC KA->CA BB->CA CB->CA CC->CA
EB->CA BE->CC BE->CD CB->CD CB->KE ED->AB AB->KF BA->AF CC->AF CA->AF
+ ... omitted several edges

We got 52 vertices and 290 edges. Let's assign the labels.

```
V(net)$label <- V(net)$name
V(net)$degree <- degree(net)
```

Histogram of node degree

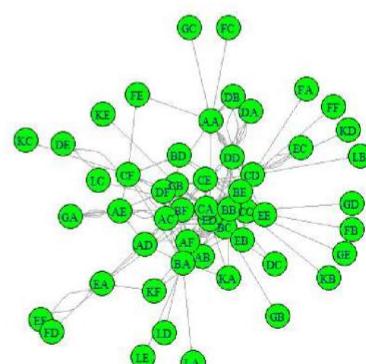
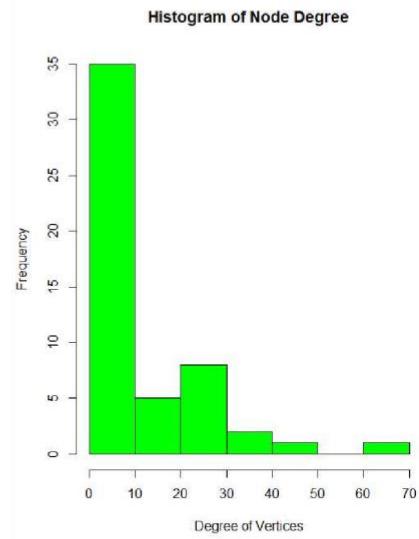
```
hist(V(net)$degree,
col = 'green',
main = 'Histogram of Node Degree',
ylab = 'Frequency',
xlab = 'Degree of Vertices')
```

Around 35 nodes with less than 10 degree and some nodes with high degree (60 to 70 connections) also.

It indicates that there are many nodes with few connections and few nodes with many connections.

Network diagram

```
set.seed(222)
plot(net,
vertex.color = 'green',
vertex.size = 2,
edge.arrow.size = 0.1,
```

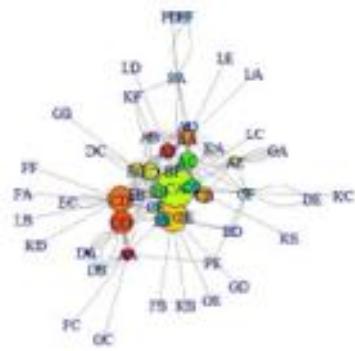


```
vertex.label.cex = 0.8)
```

Highlighting degrees & layouts

```
plot(net,
      vertex.color = rainbow(52),
      vertex.size = V(net)$degree*0.4,
      edge.arrow.size = 0.1,
      layout=layout.fruchterman.reingold)
```

CA has the highest degree followed by others.



```
plot(net,
      vertex.color = rainbow(52),
      vertex.size = V(net)$degree*0.4,
      edge.arrow.size = 0.1,
      layout=layout.graphopt)
```

```
plot(net,
      vertex.color = rainbow(52),
      vertex.size = V(net)$degree*0.4,
      edge.arrow.size = 0.1,
      layout=layout.kamada.kawai)
```

You can try out different layouts and you can use for best one suited to your requirements.



Hub and Authorities

```
hs <- hub_score(net)$vector
as <- authority.score(net)$vector
set.seed(123)
plot(net,
      vertex.size=hs*30,
      main = 'Hubs',
      vertex.color = rainbow(52),
      edge.arrow.size=0.1,
      layout = layout.kamada.kawai)
```

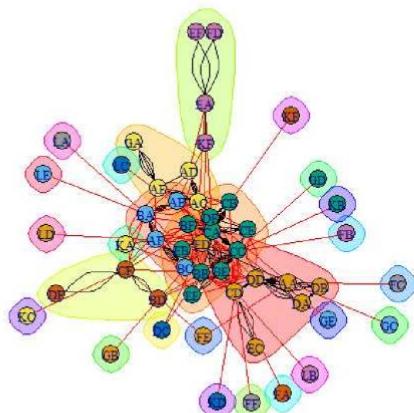


Community Detection

To detect densely connected nodes

```
net <- graph.data.frame(y, directed = F)
cnet <- cluster_edge_betweenness(net)
plot(cnet,
      net,
      vertex.size = 10,
      vertex.label.cex = 0.8)
```

Now you can see within the groups there are dense connections and between the groups with sparse connections.



CONCLUSION:

We have successfully performed the Social Network Analysis using R.

Experiment No. 2

Aim : Sqoop Installation

Sqoop installation



The screenshot shows a terminal window titled "cloudera@quickstart:~/Desktop". The window contains the following text:

```
File Edit View Search Terminal Help
[cloudera@quickstart Desktop]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ■
```

Create file using vi editor

```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
[cloudera@quickstart Desktop]$ vi file.csv
```

Insert data in file

```
cloudera@quickstart:~/Desktop
File Edit View Search Terminal Help
```

```
1, test1
2, test2
3, test3
```

Exit editor with **ctrl+Z** and type :wq and enter
Check contents of file using following command

```
[cloudera@quickstart Desktop]$ cat file.csv
```

```
1, test1
2, test2
3, test3
```

Move file from local file system to hdfs

```
[cloudera@quickstart Desktop]$ hadoop fs -put file.csv /user/cloudera/export_folder/
```

```
[cloudera@quickstart Desktop]$ hadoop fs -cat /user/cloudera/export_folder/file.csv
```

```
1, test1
2, test2
3, test3
```

Write sqoop command to export content of file to sql table

```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost/test_db --username root --password cloudera --table test1 --export-dir /user/cloudera/export_folder -m 1
```

Check Mysql table to check if data is exported

```
mysql> select * from test1;
+-----+-----+
| test_no | test_text |
+-----+-----+
| 1 | test1 |
| 2 | test2 |
| 3 | test3 |
+-----+-----+
3 rows in set (0.00 sec)

mysql>
```