EXPERIMENT-1

AIM: Execute file system commands in HDFS shell and the files and folders verify in web UI.

(1) hdfs : Used to find out which commands are available in HDFS.

Syntax : hdfs

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ hdfs
Usage: hdfs [--config confdir] COMMAND
       where COMMAND is one of:
  dfs                  run a filesystem command on the file systems supported in Hadoop.
  namenode -format     format the DFS filesystem
  secondarynamenode    run the DFS secondary namenode
  namenode             run the DFS namenode
  journalnode          run the DFS journalnode
  zkfc                 run the ZK Failover Controller daemon
  datanode             run a DFS datanode
  dfsadmin             run a DFS admin client
  diskbalancer         Distributes data evenly among disks on a given node
  haadmin              run a DFS HA admin client
  fsck                 run a DFS filesystem checking utility
  balancer             run a cluster balancing utility
  jmxget               get JMX exported values from NameNode or DataNode.
  mover                run a utility to move block replicas across
                       storage types
  oiv                  apply the offline fsimage viewer to an fsimage
  oiv_legacy           apply the offline fsimage viewer to an legacy fsimage
  oev                  apply the offline edits viewer to an edits file
  fetchdt              fetch a delegation token from the NameNode
  getconf              get config values from configuration
  groups               get the groups which users belong to
  snapshotDiff         diff two snapshots of a directory or diff the
                       current directory contents with a snapshot
  lsSnapshottableDir   list all snapshottable dirs owned by the current user
                                     Use -help to see options
  portmap              run a portmap service
  nfs3                 run an NFS version 3 gateway
  cacheadmin           configure the HDFS cache
  crypto               configure HDFS encryption zones
  storagepolicies      list/get/set block storage policies
  version              print the version
```

(2) hdfs version :Used to find out Hadoop version which is installed.

Syntax: hdfs version

```
[cloudera@quickstart ~]$ hdfs version
Hadoop 2.6.0-cdh5.8.0
Subversion http://github.com/cloudera/hadoop -r 57e7b8556919574d517e874abfb7ebe31a366c2b
Compiled by jenkins on 2016-06-16T19:38Z
Compiled with protoc 2.5.0
From source with checksum 9e99ecd28376acfd5f78c325dd939fed
This command was run using /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.8.0.jar
```

(3) hdfs dfs : Used to run a filesystem command on the file system supported in Hadoop.

Syntax: hdfs dfs

```
[cloudera@quickstart ~]$ hdfs dfs
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] [-h] [-v] <path> ...]
        [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-find <path> ... <expression> ...]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getfattr [-R] {-n name | -d} [-e en] <path>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
        [-moveToLocal <src> <localdst>]
        [-mv <src> ... <dst>]
        [-put [-f] [-p] [-l] <localsrc> ... <dst>]
        [-renameSnapshot <snapshotDir> <oldName> <newName>]
        [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
        [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
        [-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]
        [-setfattr {-n name [-v value] | -x name} <path>]
        [-setrep [-R] [-w] <rep> <path> ...]
        [-stat [format] <path> ...]
        [-tail [-f] <file>]
        [-test -[defsz] <path>]
        [-text [-ignoreCrc] <src> ...]
        [-touchz <path> ...]
        [-usage [cmd ...]]
```

(4). ls : This command is used to list all the files. Use lsr for recursive approach.it is useful when we want a hierarchy of a folder.

Syntax: hdfs dfs -ls /

```
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 7 items
drwxrwxrwx   - hdfs     supergroup          0 2016-08-10 13:07 /benchmarks
drwxr-xr-x   - hbase    supergroup          0 2023-10-22 09:47 /hbase
drwxr-xr-x   - solr     solr                0 2016-08-10 13:09 /solr
drwxr-xr-x   - cloudera supergroup          0 2023-10-17 00:11 /suresh
drwxrwxrwt   - hdfs     supergroup          0 2023-10-13 02:48 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /user
```

(5). mkdir : To create a directory. In Hadoop dfs there is no home directory by default. So let's first create it.

Syntax: hdfs dfs -mkdir<folder_name>

[cloudera@quickstart ~]$ hdfs dfs -mkdir /csd

```
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 8 items
drwxrwxrwx   - hdfs     supergroup          0 2016-08-10 13:07 /benchmarks
drwxr-xr-x   - cloudera supergroup          0 2023-10-22 10:06 /csd
drwxr-xr-x   - hbase    supergroup          0 2023-10-22 09:47 /hbase
drwxr-xr-x   - solr     solr                0 2016-08-10 13:09 /solr
drwxr-xr-x   - cloudera supergroup          0 2023-10-17 00:11 /suresh
drwxrwxrwt   - hdfs     supergroup          0 2023-10-13 02:48 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /user
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /var
```

(6). touchz : It create an empty file.

Syntax: hdfs dfs -touchz <file_path>

[cloudera@quickstart ~]$ hdfs dfs -touchz /csd/myfile.txt

[cloudera@quickstart ~]$ hdfs dfs -ls /csd

Found 1 items

-rw-r—r—1 cloudera supergroup 0 2023-10-22 23:26 /csd/myfile.txt

(7). Create three files with vim editor

```
[cloudera@quickstart ~]$ vim input.txt
[cloudera@quickstart ~]$ vim input1.txt
[cloudera@quickstart ~]$ vim input2.txt
```

To read the data for above created files

```
[cloudera@quickstart ~]$ cat input.txt
Hi hello everyone
this is KMIT college
its in top ten college list
[cloudera@quickstart ~]$ cat input1.txt
THIS IS 3rd semester
we have BDH lab as well as theory
BDH lab contains 12 experiments
[cloudera@quickstart ~]$ cat input2.txt
Experiment-1 contains excuting hdfs
commands
experiment-2 contains read 7 write operations from
local file system to hdfs
```

(8). copyFromLocal or put: To copy files/floders from local file system to hdfs store. This is most important command. Local filesystem means the files present on the OS.

Syntax: hdfs dfs -copyFromLocal<local file path> <dest(present on hdfs)>

Let's suppose we have a file imput.txt on /home/cloudera which we want to copy to folder "csd"

Present on hdfs.

```
[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal input.txt /csd
[cloudera@quickstart ~]$ hdfs dfs -put input1.txt /csd
[cloudera@quickstart ~]$ hdfs dfs -ls /csd
Found 4 items
-rw-r--r--   1 cloudera supergroup         68 2023-10-22 11:09 /csd/input.txt
-rw-r--r--   1 cloudera supergroup         89 2023-10-22 11:09 /csd/input1.txt
-rw-r--r--   1 cloudera supergroup          0 2023-10-22 10:06 /csd/myfile.txt
-rw-r--r--   1 cloudera supergroup          0 2023-10-22 10:05 /csd/myfiletxt
```

*To print file contents

Syntax: hdfs dfs -cat <path>

```
[cloudera@quickstart ~]$ hdfs dfs -cat /csd/input.txt
Hi hello everyone
this is KMIT college
its in top ten college list
```

(9). copyToLocal (or) get : To copy files/folders from hdfs store to local file system.

Syntax : hdfs dfs -copyToLocal <<srcfile(on hdfs)> <local file dest>

```
[cloudera@quickstart ~]$ mkdir kmit
[cloudera@quickstart ~]$ ls
cloudera-manager  Documents  enterprise-deployment.json  input2.txt  kmit   parcels   Templates
cm_api.py         Downloads  express-deployment.json     input.txt   lib    Pictures  Videos
Desktop           eclipse    input1.txt                  kerberos    Music  Public    workspace
[cloudera@quickstart ~]$ hdfs dfs -copyToLocal /csd/input.txt /kmit
copyToLocal: /kmit._COPYING_ (Permission denied)
[cloudera@quickstart ~]$ hdfs dfs -get /csd/input1.txt kmit
[cloudera@quickstart ~]$ ls kmit
input1.txt
```

(10). moveFromLocal : This command will move file from local to hdfs.

Syntax: hdfs dfs -moveFromLocal <local src> <dest (on hdfs)>

```
[cloudera@quickstart ~]$ hdfs dfs -moveFromLocal input2.txt /csd
[cloudera@quickstart ~]$ hdfs dfs -ls /csd
Found 5 items
-rw-r--r--   1 cloudera supergroup         68 2023-10-22 11:09 /csd/input.txt
-rw-r--r--   1 cloudera supergroup         89 2023-10-22 11:09 /csd/input1.txt
-rw-r--r--   1 cloudera supergroup        126 2023-10-22 11:27 /csd/input2.txt
-rw-r--r--   1 cloudera supergroup          0 2023-10-22 10:06 /csd/myfile.txt
-rw-r--r--   1 cloudera supergroup          0 2023-10-22 10:05 /csd/myfiletxt
```

(11). cp : This command is used to copy files within hdfs. Lets copy folder csd to csd_copied.

Syntax : hdfs dfs -cp <src(on hdfs)> <dest(on hdfs)>

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir /csd_copied
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 9 items
drwxrwxrwx   - hdfs     supergroup          0 2016-08-10 13:07 /benchmarks
drwxr-xr-x   - cloudera supergroup          0 2023-10-22 11:27 /csd
drwxr-xr-x   - cloudera supergroup          0 2023-10-22 11:36 /csd_copied
drwxr-xr-x   - hbase    supergroup          0 2023-10-22 09:47 /hbase
drwxr-xr-x   - solr     solr                0 2016-08-10 13:09 /solr
drwxr-xr-x   - cloudera supergroup          0 2023-10-17 00:11 /suresh
drwxrwxrwt   - hdfs     supergroup          0 2023-10-13 02:48 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /user
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /var
[cloudera@quickstart ~]$ hdfs dfs -cp /csd/input.txt /csd_copied
[cloudera@quickstart ~]$ hdfs dfs -ls /csd_copied
Found 1 items
-rw-r--r--   1 cloudera supergroup         68 2023-10-22 11:37 /csd_copied/input.txt
```

(12). mv : This command is used to move files within hdfs. Lets cut-paste a file myfile.txt from geeks folder to geeks_copied.

Syntax: hdfs dfs -mv <src(on hdfs)> <src(on hdfs)>

```
[cloudera@quickstart ~]$ hdfs dfs -mv /csd/input1.txt /csd_copied
[cloudera@quickstart ~]$ hdfs dfs -ls /csd_copied
Found 2 items
-rw-r--r--   1 cloudera supergroup         68 2023-10-22 11:37 /csd_copied/input.txt
-rw-r--r--   1 cloudera supergroup         89 2023-10-22 11:09 /csd_copied/input1.txt
```

(13). -rm -r : this command delets a file from HDFS recursively. It is very useful command when you want to delete a non-empty directory.

Syntax : hdfs dfs -rm -r <filename/directoryName>

```
[cloudera@quickstart ~]$ hdfs dfs -rm -r /csd_copied
Deleted /csd_copied
```

EXPERIMENT – 2

AIM: Write a program to read, write operations from local file system to HDFS using File System API.

CODE :

```
import java.io.InputStream;
import java.net.URI;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;
public class Experiment2
{
        public static void main(String[] args) throws Exception
  {
      String uri=args[0];
      Configuration conf=new Configuration();
      FileSystem fs=FileSystem.get(URI.create(uri),conf);
      /*To copy files from localFS to HDFS */
      String hdfs=args[1];
      fs.copyFromLocalFile(new Path(uri),new Path(hdfs));
      /*To copy files from HDFS to localFS */
      String local=args[1];
      fs.copyToLocalFile(new Path(uri),new Path(local));
      /*To create a directory //mkdir command */
      boolean yesorno=fs.mkdirs(new Path(uri));
      System.out.println("Status of given directory is"+yesorno);
      /*To display content of a file */
      InputStream in =null;
      try
      {
         in=fs.open(new Path(uri));
        IOUtils.copyBytes(in,System.out,4096,false);
      }
      finally
      {
         IOUtils.closeStream(in);
      }
      /*To list the files/Directories */
      FileStatus[]=fs.listStatus(new Path(uri));
      for(FileStatus file:files)
      {
         System.out.println(file.getPath().getName());
      }
      /*To delete files // rm command */
      fs.delete(new Path(uri),true);      }    }
```

OUTPUT:

➔ To copy files from localFielSystem to HDFS

```
cloudera@quickstart:~/workspace
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ cd workspace/
[cloudera@quickstart workspace]$  vim in
[cloudera@quickstart workspace]$ ha
hadoop                  hadoop-fuse-dfs          hal-device               hal-find-by-
hadoop-0.20             hald                     hal-disable-polling      hal-find-by-
[cloudera@quickstart workspace]$ hadoop jar CSD.jar Experiment2 in /csd
[cloudera@quickstart workspace]$ hdfs dfs -ls /csd
Found 7 items
-rw-r--r--   1 cloudera supergroup          6 2023-10-26 23:45 /csd/in
-rw-r--r--   1 cloudera supergroup         68 2023-10-22 11:09 /csd/input.txt
-rw-r--r--   1 cloudera supergroup        126 2023-10-22 11:27 /csd/input2.txt
-rw-r--r--   1 cloudera supergroup          0 2023-10-22 10:06 /csd/myfile.txt
-rw-r--r--   1 cloudera supergroup          0 2023-10-22 10:05 /csd/myfiletxt
drwxr-xr-x   - cloudera supergroup          0 2023-10-23 01:28 /csd/output
-rw-r--r--   1 cloudera supergroup         86 2023-10-22 23:48 /csd/salary.txt
```

➔ To copy files from HDFS to LocalFileS

```
[cloudera@quickstart ~]$ cd workspace
[cloudera@quickstart workspace]$ rm sample.txt
rm: remove regular file `sample.txt'? yes
[cloudera@quickstart workspace]$ clear
[cloudera@quickstart workspace]$ ls
exp2.jar       Experiment 1  input.txt  sai           training
experiment02   Experiment2    main.txt   sample.jard
[cloudera@quickstart workspace]$ hdfs dfs -ls /
Found 11 items
drwxrwxrwx   - hdfs     supergroup          0 2016-08-10 13:07 /benchmarks
drwxr-xr-x   - hbase    supergroup          0 2023-11-02 08:27 /hbase
-rw-r--r--   1 cloudera supergroup          0 2023-11-02 09:10 /input.txt
-rw-r--r--   1 cloudera supergroup         25 2023-11-15 08:18 /main.txt
drwxr-xr-x   - cloudera supergroup          0 2023-11-02 23:49 /nag
drwxr-xr-x   - cloudera supergroup          0 2023-11-02 10:57 /nagendra2
-rw-r--r--   1 cloudera supergroup         15 2023-11-15 08:40 /sample.txt
drwxr-xr-x   - solr     solr                0 2016-08-10 13:09 /solr
drwxrwxrwt   - hdfs     supergroup          0 2023-11-02 08:28 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /user
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /var
[cloudera@quickstart workspace]$ hadoop jar exp2.jar Experiment2 /sample.txt .
[cloudera@quickstart workspace]$ ls
exp2.jar       Experiment 1  input.txt  sai           sample.txt
experiment02   Experiment2    main.txt   sample.jard   training
[cloudera@quickstart workspace]$
```

➔ To create a directory //mkdir command

```
[cloudera@quickstart workspace]$ hadoop jar exp2.jar Experiment2 /Sample
Status of given directpry istrue
[cloudera@quickstart workspace]$ hdfs dfs -ls /
Found 12 items
drwxr-xr-x   - cloudera supergroup          0 2023-11-15 08:48 /Sample
drwxrwxrwx   - hdfs     supergroup          0 2016-08-10 13:07 /benchmarks
drwxr-xr-x   - hbase    supergroup          0 2023-11-02 08:27 /hbase
-rw-r--r--   1 cloudera supergroup          0 2023-11-02 09:10 /input.txt
-rw-r--r--   1 cloudera supergroup         25 2023-11-15 08:18 /main.txt
drwxr-xr-x   - cloudera supergroup          0 2023-11-02 23:49 /nag
drwxr-xr-x   - cloudera supergroup          0 2023-11-02 10:57 /nagendra2
-rw-r--r--   1 cloudera supergroup         15 2023-11-15 08:40 /sample.txt
drwxr-xr-x   - solr     solr                0 2016-08-10 13:09 /solr
drwxrwxrwt   - hdfs     supergroup          0 2023-11-02 08:28 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /user
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /var
[cloudera@quickstart workspace]$
```

➔ To display content of a file

```
[cloudera@quickstart workspace]$ hadoop jar exp2.jar Experiment2 /sample.txt
hi hello
bye
```

➔ To list the files/Directories

```
[cloudera@quickstart workspace]$ hadoop jar exp2.jar Experiment2 /
Sample
benchmarks
hbase
input.txt
main.txt
nag
nagendra2
sample.txt
solr
tmp
user
var
[cloudera@quickstart workspace]$
```

➔ To delete files // rm command

```
[cloudera@quickstart workspace]$ hdfs dfs -ls /
Found 12 items
drwxr-xr-x   - cloudera supergroup          0 2023-11-15 08:48 /Sample
drwxrwxrwx   - hdfs     supergroup          0 2016-08-10 13:07 /benchmarks
drwxr-xr-x   - hbase    supergroup          0 2023-11-02 08:27 /hbase
-rw-r--r--   1 cloudera supergroup          0 2023-11-02 09:10 /input.txt
-rw-r--r--   1 cloudera supergroup         25 2023-11-15 08:18 /main.txt
drwxr-xr-x   - cloudera supergroup          0 2023-11-02 23:49 /nag
drwxr-xr-x   - cloudera supergroup          0 2023-11-02 10:57 /nagendra2
-rw-r--r--   1 cloudera supergroup         15 2023-11-15 08:40 /sample.txt
drwxr-xr-x   - solr     solr                0 2016-08-10 13:09 /solr
drwxrwxrwt   - hdfs     supergroup          0 2023-11-02 08:28 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /user
drwxr-xr-x   - hdfs     supergroup          0 2016-08-10 13:09 /var
```

EXPERIMENT – 3

AIM: Implement Map-Reduce application to find sum of salaries of employees for each department.

CODE:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class MR
{
   public static class MapperClass extends
   Mapper<LongWritable, Text, Text, IntWritable>
   {
      public void map(LongWritable key, Text empRecord, Context con)
      throws IOException, InterruptedException
      {
         String[] word = empRecord.toString().split(" ");
         if (word.length == 2)
         {
            String dept = word[0];
            int salary = Integer.parseInt(word[1]);
            con.write(new Text(dept), new IntWritable(salary));
         }
      }
   }
   public static class ReducerClass extends
   Reducer<Text, IntWritable, Text, IntWritable>
   {
         public void reduce(Text key, Iterable<IntWritable> valueList,Context con) throws
IOException, InterruptedException
      {
         int sum = 0;
         for (IntWritable var : valueList)
         {
            sum += var.get();
         }
         con.write(key, new IntWritable(sum));
      }
   }
```

```
    public static void main(String[] args) throws Exception
    {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "FindAverageAndTotalSalary");
        job.setJarByClass(MR.class);
        job.setMapperClass(MapperClass.class);
        job.setReducerClass(ReducerClass.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);
    }
}
```

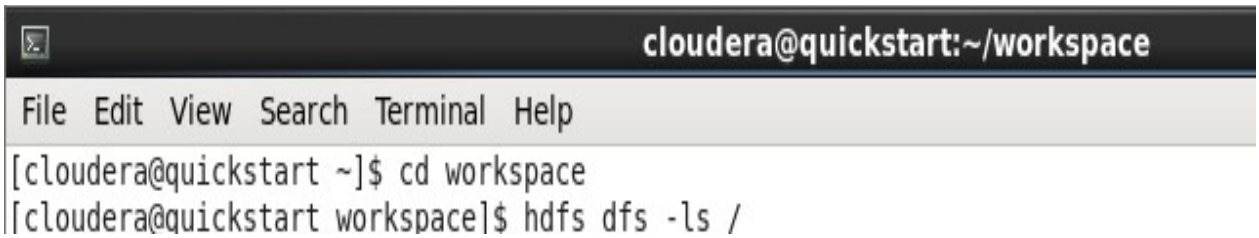
Steps: a) First create a file with name salary.txt

➔ Check that data is been written to the file

➔ Copy the file from localfile system to hdfs

[cloudera@quickstart ~]$ hdfs dfs -put salary.txt /csd

Output:



```
[cloudera@quickstart workspace]$ hadoop jar kmit.jar MR /csd/salary.txt /csd/output123
23/10/31 03:51:19 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
```

```
[cloudera@quickstart workspace]$ hdfs dfs -ls /csd/output123
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2023-10-31 03:39 /csd/output123/_SUCCESS
-rw-r--r--   1 cloudera supergroup         40 2023-10-31 03:39 /csd/output123/part-r-00000
[cloudera@quickstart workspace]$ hdfs dfs -cat /csd/output123/part-r-00000
Admin   40000
CEO     40000
HR      50000
IT      50000
```

EXPERIMENT – 4

AIM: Implement Map-Reduce application to find number of times a word is repeated in a text file.

CODE:

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWriatble;
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;
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.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.Iterator;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class WCMapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>
{
   static IntWriting one = new IntWritable(1);
   Text word=new Text();
   public void map(LongWritable Key,Text
value,OutputCollector<Text,InWritable>output,Reporter arg3) throws IoException
   {
      //To do Auto-generated method stub
      String line=value.toString();
      StringTokenizer tokens=new StringTokenizer(line);
      while(tokens.hasMoreTokens())
      {
         word.set(tokens.nextToken());
         output.collect(word,one);
      }
   }
}
public class WCReducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,Intwritable>
{
```

```
   public void reduce(Text Key,Iterator<IntWritable>value,
OutputCollector<Text,Intwritable> output, Reportor arg3) throws IOEXception
   {
      // To do Auto-generated method stub
      int sum=0;
      while(value.hasNext())
      {
         sum=sum+value.next().get();
      }
      output.collect(key,new Intwritable(sum));
   }
}
public class WC
{
   public static void main(String[] args) throws IOException
   {
      // To do Auto-generated method stub
      JobConf conf=new JobConf(WD.class);
      conf.setJobName("Suresh");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputKeyValue(IntWriatble.class);
      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);
      FileInputFormat.setInputPaths(conf,new Path(args[0]));
      FileOutputFormat.setOutputPaths(conf,new Path(args[1]));
      conf.setMapperClass(WCMapper.class);
      conf.setReducerClass(WCReducer.class);
      JobClient.runJob(conf);

   }
}
```

Step 1:

a) First create a file with name nagendra.txt

➔ Check that data is been written to the file

[cloudera@quickstart ~]$ vim nagendra.txt

[cloudera@quickstart ~]$ cat nagendra.txt

A a B b C c

A B C a b c

a b c A D E

dear bear

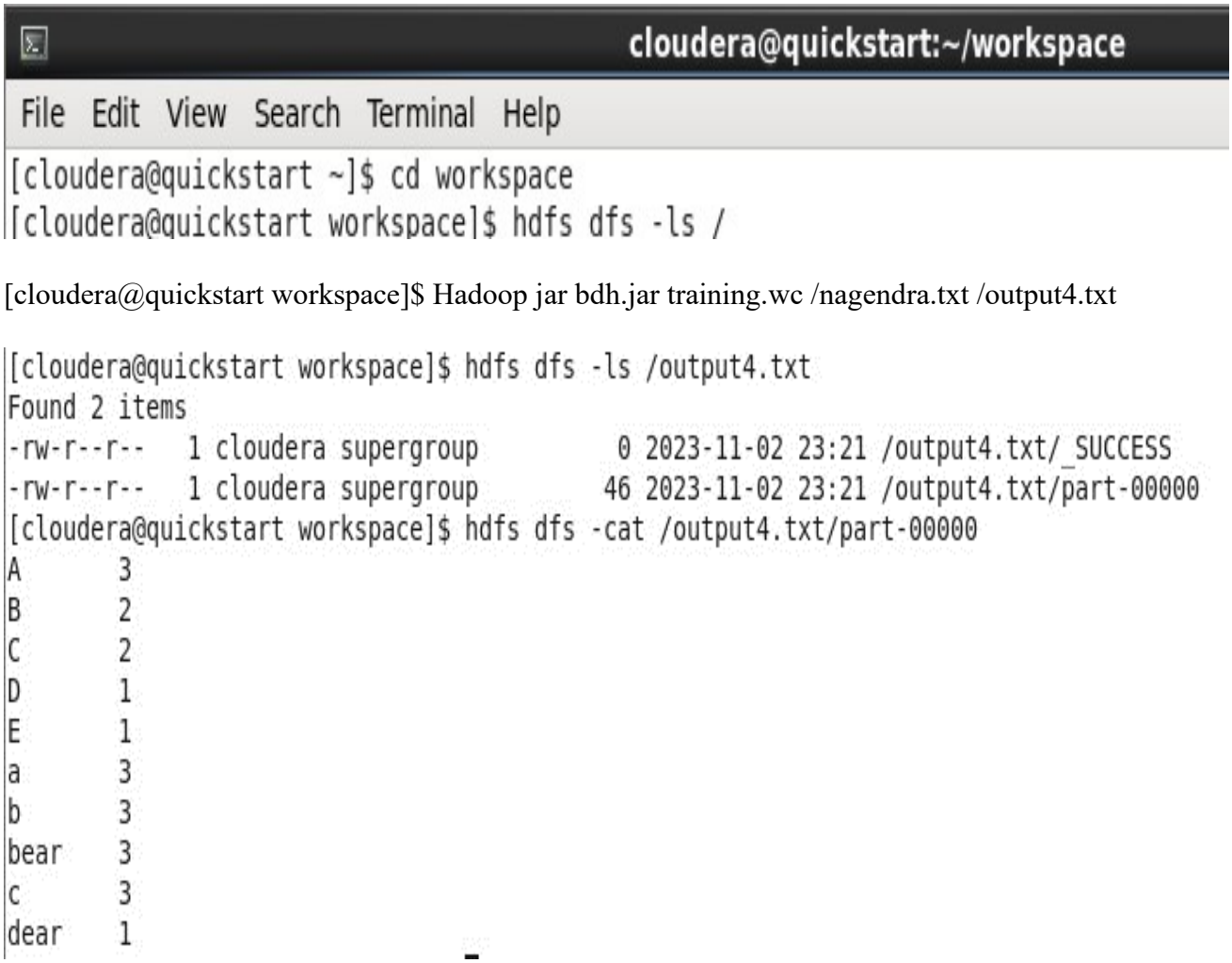
bear

bear

➔ Copy the file from localfile system to hdfs

[cloudera@quickstart ~]$ hdfs dfs -copyFromLocal nagendra.txt /

Output :



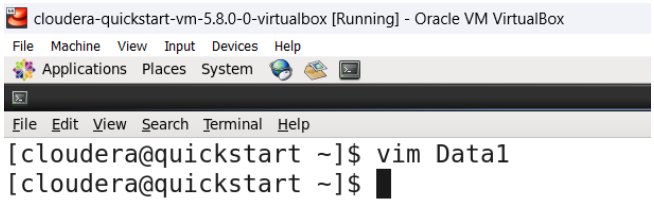
Experiment-5

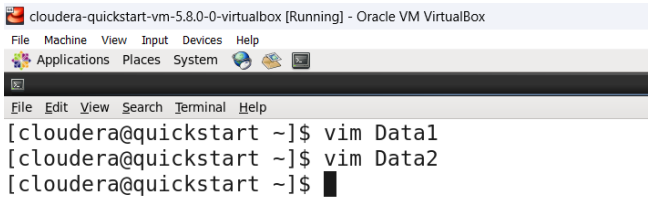
Aim: Write A Pig Script for File Analysis

Pig Latin Statements

First Let's Create Two Text Files '**Data1**' and '**Data2**':

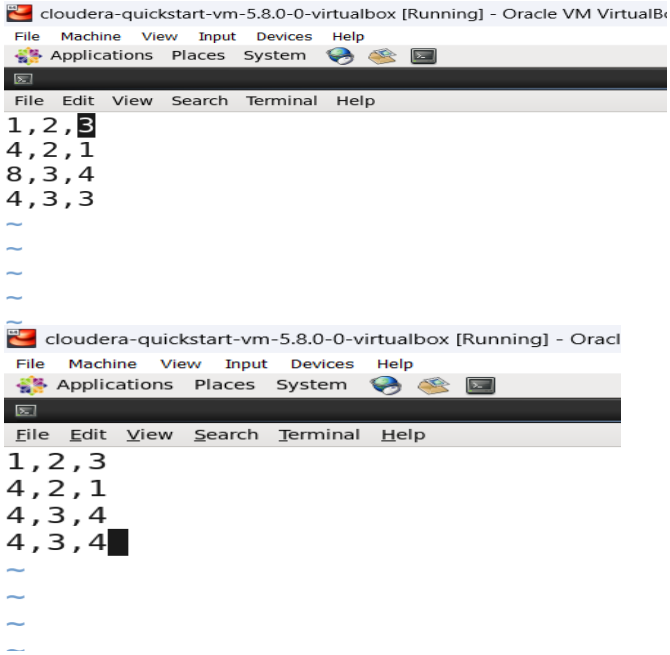
- use "**vim <filename>**"





In those Files, Provide the below Data respectively:

- Press ' **i** ' to enter insert mode.



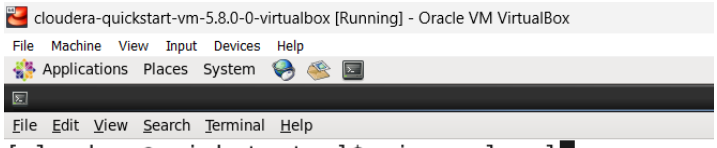
After entering the data, press ' **Esc** ' and ' **:wq** ' to leave the file.

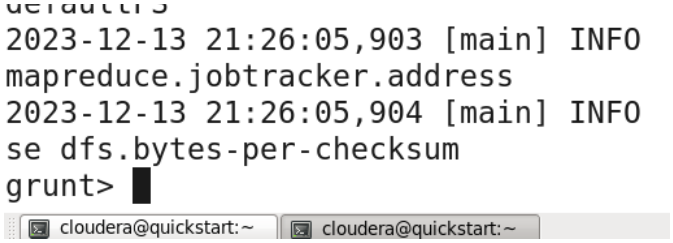
Loading And Storing Operations

A grunt shell is an interface where all pig statements are executed.

Let's enter grunt shell:

- Use "**pig -x local**"





After Entering Grunt Shell Type, the below commands:

1) Load

- **grunt> a = load '/home/cloudera/data1' using PigStorage(',')as(a1:int,a2:int,a3:int)**
- **grunt> b = load '/home/cloudera/data2' using PigStorage(',') as(a1:int,a2:int,a3:int)**

The above command loads **Data1** and **Data2** files existing in local directory.

To display the loaded data:

- Use **Dump a;** Here **dump** is a diagnostic operation to display on console.

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

- Use **Dump b;**

(1,2,3)

(4,2,1)

(4,3,4)

(3,2,1)

If the syntax and file names are properly given, data in those files will be displayed.

2) Store

- **grunt> store a into '/home/cloudera/dataout' using PigStorage('.');**

  The above command stores alias/relation '**a**' into local directory file.

  '**dataout**' is the output path where a is located.

  Let's navigate to the directory…

- **grunt> cd /home/cloudera/dataout**

  let's display the stored **'a'**

- **grunt> cat part-m-00000** Here part-m-00000 is the output as mapper file.

  1.2.3

  4.2.1

  8.3.4

  4.3.3

Combining, Splitting and Sorting Data

3) Union

- **grunt> c = union a,b;**

Union is used to display the combined data of both files.

Let's display union of a,b operators.

- **grunt> dump c;**

  (1,2,3)

  (4,2,1)

  (4,3,4)

  (3,2,1)

  (1,2,3)

  (4,2,1)

  (8,3,4)

  (4,3,3)

4) Split

- **grunt> describe c;**

  We use describe to view the schema of an operator

c: {a1: int,a2: int,a3: int}

- **grunt> split c into d if(a1>3),e if(a1>=1 and a1<=3);**

  Split's **'c'** into **'d'** and **'e'** based on some conditions.

- **grunt> dump d;**

  (4,2,1)

  (8,3,4)

  (4,3,3)

  (4,2,1)

  (4,3,4)

- **grunt> dump e;**

  (1,2,3)

  (1,2,3)

  (3,2,1)

5) **Order**

- **grunt> f = order b by $0 DESC;**

  Orders data of '**b**' by 1st index in descending order

  grunt> dump f;

  (4,3,4)

  (4,2,1)

  (3,2,1)

  (1,2,3)

- **grunt> g = order b by $0 ASC;**

  Orders data of '**b**' by 1st index in Ascending order

- **grunt> dump g;**

  (1,2,3)

  (3,2,1)

  (4,3,4)

  (4,2,1)

6) **Limit**

- **grunt> h = limit g 2;**

  here data of **'g'** is limited to 2 rows

grunt> dump h;

(1,2,3)

(3,2,1)

Filtering Data

7) **Filter**

- **grunt> i = filter c by $0 == 1 and $1 == 2;**
  filter data of **'c'** by 1st index and 2nd index upon conditions.
- **grunt> dump i;**
  (1,2,3)
  (1,2,3)

8) **Distinct**

- **grunt> j = distinct c;**
  from the union of data from **Data1** and **Data2**, let's select distinct rows
- **grunt> dump j;**
  (1,2,3)
  (3,2,1)
  (4,2,1)
  (4,3,3)
  (4,3,4)
  (8,3,4)

9) **Illustrate**

to view the step-by-step execution of a series of statements.

grunt> illustrate k;

```
-------------------------------------------------
| a | a1:int | a2:int | a3:int |
-------------------------------------------------
| |1 |2 |3 |
-------------------------------------------------
-------------------------------------------------
| b | a1:int | a2:int | a3:int |
-------------------------------------------------
| |1 |2 |3 |
-------------------------------------------------
```

| c | a1:int | a2:int | a3:int |

| |1 |2 |3 |
| |1 |2 |3 |

| j | a1:int | a2:int | a3:int |

| |1 |2 |3 |

| k | a1:int | a2:int | :int |

| |1 |2 |2 |

---

10) Join

grunt> l = join a by $0, b by $2;

Joins 2 inputs based on a key. Here a and b are the inputs and the indices are the keys.

grunt> dump l;

(1,2,3,3,2,1)

(1,2,3,4,2,1)

(4,3,3,4,3,4)

(4,2,1,4,3,4)

11) Left Outer Join

grunt> m = join a by $0 left Outer, b by $2;

grunt> dump m;

(1,2,3,3,2,1)

(1,2,3,4,2,1)

(4,3,3,4,3,4)

(4,2,1,4,3,4)

(8,3,4,,,)

12) Right Outer Join

grunt> n = join a by $0 right, b by $2;

grunt> dump n;

(1,2,3,3,2,1)

(1,2,3,4,2,1)

(,,,1,2,3)

(4,3,3,4,3,4)

(4,2,1,4,3,4)

13) Group

grunt> p = group c by $0;

Used to group the data in one or more relations. Here we are grouping input **c** based on **0^{th} index**.

grunt> dump p;

(1,{(1,2,3),(1,2,3)})

(3,{(3,2,1)})

(4,{(4,3,4),(4,2,1),(4,3,3),(4,2,1)})

(8,{(8,3,4)})

14) Co-Group

grunt> q = cogroup a by $0, b by $0;

Used to group the data in one or more relations. Here we are grouping input **a** and **b** based on **0^{th} index.** Difference between group and co-group is that here 2 or more relations are used.

grunt> dump q;

(1,{(1,2,3)},{(1,2,3)})

(3,{},{(3,2,1)})

(4,{(4,3,3),(4,2,1)},{(4,3,4),(4,2,1)})

(8,{(8,3,4)},{})

15) Cross

grunt> r = cross a,b;

Provides cross product of 2 relations. In our case **a,b** are 2 relations.

grunt> dump r;

(4,3,3,3,2,1)

(4,3,3,4,3,4)

(4,3,3,4,2,1)

(4,3,3,1,2,3)

(8,3,4,3,2,1)

(8,3,4,4,3,4)

(8,3,4,4,2,1)

(8,3,4,1,2,3)

……

We can also type all this commands into a script file. A script file is an ordinary file but created with an extension of **.pig**

$ vim script.pig

After we have written our commands in the file, we can run using

[cloudera@quickstart ~]$ pig -x local script.pig

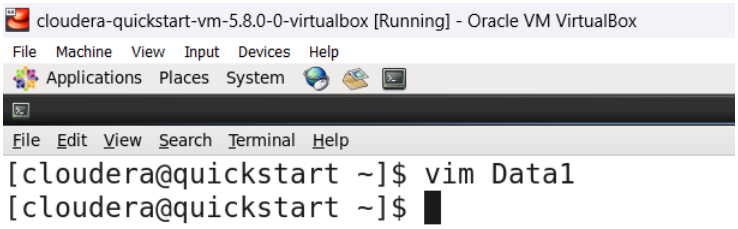
The name of script can be anything not necessarily as ‘script’.

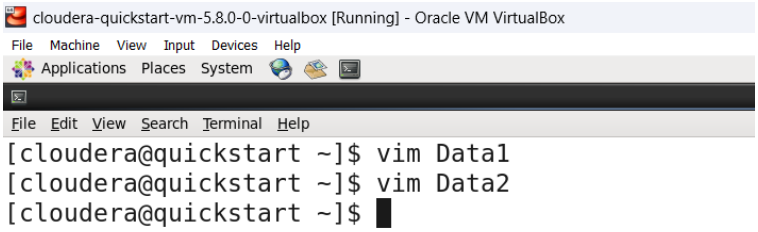
Experiment-6

Aim: Implement a pig script to analyse the data using UDFs.

In the previous Experiment we have executed commands. Now we will type them in format of functions in scripts.

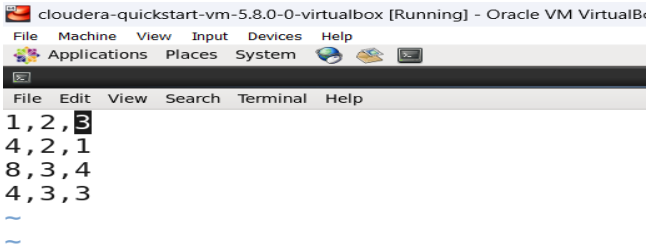
First Let's create document called **Data1** and **Data2** using **vim** command:

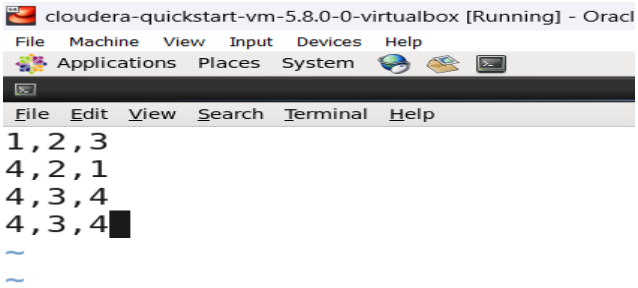




In those Files, Provide the below Data respectively:

- Press ' **i** ' to enter insert mode.

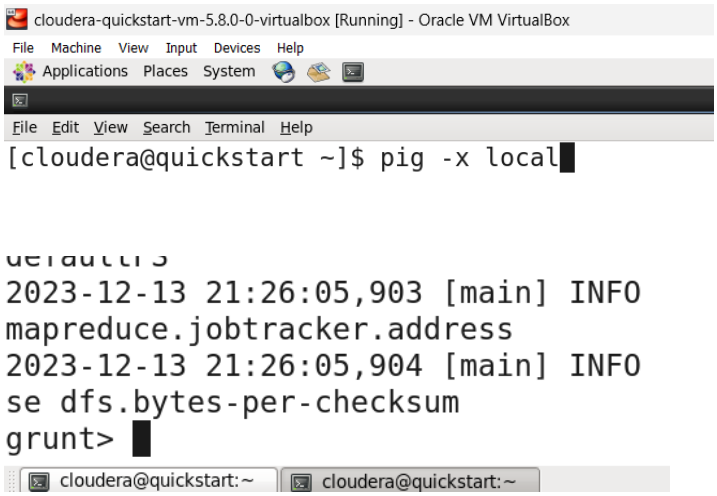




After entering the data, press ' **Esc** ' and ' **:wq** ' to leave the file.

Let’s enter grunt shell:

- Use “**pig -x local**”



Now load **Data1** and **Data2** files:

- **grunt> a = load '/home/cloudera/data1' using PigStorage(',')as(a1:int,a2:int,a3:int)**
- **grunt> b = load '/home/cloudera/data2’ using PigStorage(',') as(a1:int,a2:int,a3:int)**
- Use **Dump a**
  (1,2,3)
  (4,2,1)
  (8,3,4)
  (4,3,3)
- Use **Dump b;**
  (1,2,3)
  (4,2,1)
  (4,3,4)
  (3,2,1)

Let’s also perform the union operation, we can utilize it later.

- **grunt> c = union a,b;**
- **grunt> dump c;**
  (1,2,3)
  (4,2,1)
  (4,3,4)
  (3,2,1)
  (1,2,3)
  (4,2,1)
  (8,3,4)
  (4,3,3)

Now Execute these commands:

1) **Define**

- **grunt> DEFINE myfilter(relvar,colvar) returns x{ $x = filter $relvar by $colvar<=2; };**

Here define is used to create a function like interface. We have created a function **myfilter** having **relation variable** and **column variable** which returns the filtered values of an input in function declaration.

- **grunt> z = myfilter(c,$1);**

  Now upon declaration we have provided **c** as input and with **index 1.**

  Let's look at the result:

  grunt> dump z;

  (1,2,3)

  (4,2,1)

  (1,2,3)

  (4,2,1)

  (3,2,1)

  Now let's create a directory in HDFS:

  [cloudera@quickstart ~]$ hdfs dfs -mkdir /demo1

  [cloudera@quickstart ~]$ hdfs dfs -ls /

  drwxr-xr-x - cloudera supergroup 0 2023-12-21 09:34 /demo1

  After creation let's transfer the local files **Data1** and **Data2** into hdfs.

  [cloudera@quickstart ~]$ hdfs dfs -put data /demo

  [cloudera@quickstart ~]$ hdfs dfs -put data1 /demo

  [cloudera@quickstart ~]$ hdfs dfs -ls /demo

  -rw-r--r-- 1 cloudera supergroup 24 2023-04-09 06:42 /demo/data

  -rw-r--r-- 1 cloudera supergroup 24 2023-04-09 06:42 /demo/data1

2) **Import**

Now after all the above process, create an **.macro** extension file. Input the below data in that file via insert mode:

[cloudera@quickstart ~]$ vim file.macro

```
DEFINE myfilter(relvar,colvar) returns x{ $x = filter $relvar by $colvar<=2; };
~
```

Let's also create a **.pig** file with following data:

[cloudera@quickstart ~]$ vim file1.pig

```
IMPORT '/home/cloudera/mymacro.macro';
a = load '/demo/data' using PigStorage(',') as (a1:int,a2:int,a3:int);
b = load '/demo/data1' using PigStorage(',') as (a1:int,a2:int,a3:int);
c = union a,b;
data1 = myfilter(c,a1);
dump data1;
```

Here we have imported the functionality of the **file.macro** file's **define** command into our **file1.pig.** In **Macro** file we have defined our function and in **pig** file we have declared it. Both files were bridged using **import** command.

3) **Register**

Registers a **JAR file** so that the **UDFs** in the file can be used.

In order to start with a **.jar** file, let's make one **via eclipse of cloudera**.

First create a **new project** and extract the required libraries, which are *the* **jar files of pig folder**. Now open a new class and type the code below:

```
import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;
public class Pig_Udf extends EvalFunc<String> {
public String exec(Tuple input) throws IOException {
if (input == null || input.size() == 0)
return null;
String str = (String) input.get(0);
return str.toUpperCase();
```

}

}

Now next export the project as an .jar file through export option.

Open a terminal and create a file named **"demo"** with the content:

[cloudera@quickstart ~]$ vim demo

1,vvv

2,xyx

3,yyy

4,uuu

Exit the file and open grunt shell:

[cloudera@quickstart ~]$ pig -x local

After opening the grunt shell, type the below commands:

grunt> REGISTER /home/cloudera/workspace/pigudf.jar;

we are registering the jar file we just created while ago.

grunt> DEFINE myfun Pig_Udf();

grunt> a = load '/home/cloudera/demo' using PigStorage(',') as (id:int,name:chararray);

grunt> emp_Upper = foreach a generate myfun(name);

In the above commands we have loaded the **demo** file which has some data.

Using **define** command we have made a function to **uppercase** all the data elements.

Upon displaying the changes:

grunt> dump emp_Upper;

(VVV)

(XYX)

(YYY)

(UUU)

Experiment 7

Create Managed table and load a csv file from local storage. Verify the location of table data in HDFS and perform query operations. Delete the table and verify the data in HDFS.

1)Create few files as follows:

```
[cloudera@quickstart ~]$ vi abc.txt
```

```
tejas,150000,IT
mohammad,200000,IT
ram,200000,HR
naresh,200000,FIN
```

```
[cloudera@quickstart ~]$ vi abc1.txt
```

```
srinivas,250000,IT
akilesh,250000,IT
aswini,250000,IT
```

```
[cloudera@quickstart ~]$ vi emp.txt
```

```
swetha,250000,IT
anamika,200000,FIN
tarun,300000,HR
```

2)Create a directory on hdfs using following command

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir /demo
```

3)Copy all the created files from local file system to HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -put abc.txt /demo
[cloudera@quickstart ~]$ hdfs dfs -put abc1.txt /demo
[cloudera@quickstart ~]$ hdfs dfs -put emp.txt /demo
```

4)Check all the files have copied to "/demo" directory on HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls /demo
Found 3 items
-rw-r--r--   1 cloudera supergroup         70 2023-12-17 02:29 /demo/abc.txt
-rw-r--r--   1 cloudera supergroup         57 2023-12-17 02:29 /demo/abc1.txt
-rw-r--r--   1 cloudera supergroup         56 2023-12-17 02:29 /demo/emp.txt
```

5)Launch the hive shell using following command:

```
[cloudera@quickstart ~]$ hive
2023-12-17 02:37:43,003 WARN  [main] mapreduce.TableMapReduceUtil: The hbase-pre
fix-tree module jar containing PrefixTreeCodec is not present.  Continuing witho
ut it.

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.p
roperties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>
```

6)Check the databases

```
hive> show databases;
OK
default
Time taken: 0.023 seconds, Fetched: 1 row(s)
```

7)Create a database

```
hive> create database kmit;
OK
Time taken: 0.148 seconds
```

8)Use the created database

```
hive> use kmit;
OK
Time taken: 0.033 seconds
```

9)Check the tables available in the database

```
hive> show tables;
OK
Time taken: 0.023 seconds
```

10)Create a Managed table

```
hive> create table employee(name string,salary float,dept string)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.186 seconds
```

11)Load contents from a file on local file system to Table in hive

```
hive> load data local inpath '/home/cloudera/abc.txt' into table employee;
Loading data to table kmit.employee
Table kmit.employee stats: [numFiles=1, totalSize=70]
OK
Time taken: 0.895 seconds
```

12)Read the data from Managed table named employee

```
hive> select * from employee;
OK
tejas   150000.0        IT
mohammad        200000.0        IT
ram     200000.0        HR
naresh  200000.0        FIN
Time taken: 0.072 seconds, Fetched: 7 row(s)
```

13)Now load the data from a file from HDFS to table

Check the location of file on HDFS

```
[cloudera@quickstart ~]$ hdfs dfs -ls /demo
Found 3 items
-rw-r--r--   1 cloudera supergroup         70 2023-12-17 02:29 /demo/abc.txt
-rw-r--r--   1 cloudera supergroup         57 2023-12-17 02:29 /demo/abc1.txt
-rw-r--r--   1 cloudera supergroup         56 2023-12-17 02:29 /demo/emp.txt
```

Now load the data (abc.txt) to the table (employee)

```
hive> load data inpath '/demo/abc1.txt' into table employee;
Loading data to table default.employee
Table default.employee stats: [numFiles=1, totalSize=57]
OK
Time taken: 0.581 seconds
```

Now check the files in the location

```
[cloudera@quickstart ~]$ hdfs dfs -ls /demo
Found 3 items
-rw-r--r--   1 cloudera supergroup         70 2023-12-17 02:29 /demo/abc.txt
-rw-r--r--   1 cloudera supergroup         57 2023-12-17 03:52 /demo/abc1.txt
-rw-r--r--   1 cloudera supergroup         56 2023-12-17 02:29 /demo/emp.txt
```

14)Check the Schema of the table

```
hive> describe employee;
OK
name                    string
salary                  float
dept                    string
Time taken: 0.532 seconds, Fetched: 3 row(s)
```

15)Check that you have created a managed table

```
hive> describe extended employee;
OK
name                    string
salary                  float
dept                    string

Detailed Table Information      Table(tableName:employee, dbName:default, owner:
cloudera, createTime:1702813319, lastAccessTime:0, retention:0, sd:StorageDescri
ptor(cols:[FieldSchema(name:name, type:string, comment:null), FieldSchema(name:s
alary, type:float, comment:null), FieldSchema(name:dept, type:string, comment:nu
ll)], location:hdfs://quickstart.cloudera:8020/user/hive/warehouse/employee, inp
utFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoo
p.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, ser
deInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.
LazySimpleSerDe, parameters:{serialization.format=,, field.delim=,}), bucketCols
:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewed
ColValues:[], skewedColValueLocationMaps:{}), storedAsSubDirectories:false), par
titionKeys:[], parameters:{numFiles=2, transient_lastDdlTime=1702814043, COLUMN_
STATS_ACCURATE=true, totalSize=114}, viewOriginalText:null, viewExpandedText:nul
l, tableType:MANAGED_TABLE)
Time taken: 0.067 seconds, Fetched: 5 row(s)
```

Experiment 8

Create External table and load a csv file from local storage. Perform query operations and verify the data in HDFS. Delete the table and verify the data in HDFS.

1) **Now create a external table**

- Now create a directory

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir /demo1
```

- Now copy the file (abc.txt) from /demo to /demo1

```
[cloudera@quickstart ~]$ hdfs dfs -cp /demo/abc.txt /demo1
```

```
[cloudera@quickstart ~]$ hdfs dfs -ls /demo1
Found 1 items
-rw-r--r--   1 cloudera supergroup         67 2023-12-16 20:45 /demo1/abc.txt
```

- Now create a external table

```
[cloudera@quickstart ~]$ hive
2023-12-16 20:45:51,543 WARN  [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree modul
e jar containing PrefixTreeCodec is not present.  Continuing without it.

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
```

```
hive> create external table employee2(name string,salary float,dept string)
    > row format delimited
    > fields terminated by','
    > location'/demo1';
OK
Time taken: 0.558 seconds
```

- Read the data from the external table

```
hive> select * from employee2;
OK
tejas   150000.0        IT
mohammad        200000.0        IT
ram     200000.0        HR
naresh  200000.0        FIN
Time taken: 0.416 seconds, Fetched: 4 row(s)
```

- How to overwrite data in table

```
hive> load data inpath'/demo/emp.txt'overwrite into table employee2;
Loading data to table default.employee2
Table default.employee2 stats: [numFiles=1, numRows=0, totalSize=51, rawDataSize=0]
OK
Time taken: 0.338 seconds
```

```
hive> select * from employee2;
OK
swetha  250000.0        IT
anamika 200000.0        IT
tarun   300000.0        HR
Time taken: 0.077 seconds, Fetched: 3 row(s)
```

- How to drop a table

```
hive> show tables;
OK
employee
employee2
Time taken: 0.056 seconds, Fetched: 2 row(s)
```

```
hive> drop table employee;
OK
Time taken: 0.429 seconds
```

```
hive> drop table employee2;
OK
Time taken: 0.087 seconds
```

```
hive> show tables;
OK
Time taken: 0.07 seconds
```

EXPERIMENT 9

AIM :- Create partitioned table using Static partitioning technique and load the data into corresponding partitions.

Context :- Partitions are defined at table creation time using the PARTITIONED BY clause, which takes a list of column definitions.

(1) CREATE a partitioned table

```
hive> create table employee(name string, salary float, dept string)
    > partitioned by(country string, state string)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 6.826 seconds
```

(2) Describe the table

```
hive> describe employee;
OK
name                    string
salary                  float
dept                    string
country                 string
state                   string

# Partition Information
# col_name              data_type               comment

country                 string
state                   string
Time taken: 3.616 seconds, Fetched: 11 row(s)
hive>
```

(3) Load the data from local file system to table employee

```
hive> load data local inpath 'abc.txt' into table employee
    > partition(country='INDIA',state='KAR');
Loading data to table kmit.employee partition (country=INDIA, state=KAR)
Partition kmit.employee{country=INDIA, state=KAR} stats: [numFiles=1, numRows=0, totalSize=67, rawDataSize=0]
OK
Time taken: 0.696 seconds
hive>
```

(4) Read the data from employee table

```
hive> select * from employee;
OK
tejas   150000.0        IT      INDIA   KAR
mohammad        200000.0        IT      INDIA   KAR
ram     200000.0        HR      INDIA   KAR
Naresh  200000.0        FIN     INDIA   KAR
Time taken: 1.031 seconds, Fetched: 4 row(s)
hive>
```

(5) Load the data from local file system to table employee

```
hive> load data local inpath 'abc1.txt' into table employee
    > partition(country='INDIA',state='MH');
Loading data to table kmit.employee partition (country=INDIA, state=MH)
Partition kmit.employee{country=INDIA, state=MH} stats: [numFiles=1, numRows=0, totalSize=54, rawDataSize=0]
OK
Time taken: 0.702 seconds
hive>
```

(6) Read the data from employee table

```
hive> select * from employee;
OK
tejas   150000.0        IT      INDIA   KAR
mohammad        200000.0        IT      INDIA   KAR
ram     200000.0        HR      INDIA   KAR
Naresh  200000.0        FIN     INDIA   KAR
srinivas        250000.0        IT      INDIA   MH
akilesh 250000.0        IT      INDIA   MH
aswini  250000.0        IT      INDIA   MH
Time taken: 0.106 seconds, Fetched: 7 row(s)
hive>
```

(7) Where condition

```
hive> select * from employee where state='MH';
OK
srinivas        250000.0        IT      INDIA   MH
akilesh 250000.0        IT      INDIA   MH
aswini  250000.0        IT      INDIA   MH
Time taken: 1.131 seconds, Fetched: 3 row(s)
hive>
```

(8) File system structure how Partitioned Tables will handle data

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/kmit.db/employee/country=INDIA
Found 2 items
drwxrwxrwx   - cloudera supergroup          0 2023-12-15 08:33 /user/hive/warehouse/kmit.db/employee/country=INDIA/state=KAR
drwxrwxrwx   - cloudera supergroup          0 2023-12-15 08:42 /user/hive/warehouse/kmit.db/employee/country=INDIA/state=MH
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/kmit.db/employee/country=INDIA/state=MH
Found 1 items
-rwxrwxrwx   1 cloudera supergroup         54 2023-12-15 08:42 /user/hive/warehouse/kmit.db/employee/country=INDIA/state=MH/abc1.txt
[cloudera@quickstart ~]$
```

Experiment 10

AIM:

Create and manage a table with bucketing concept and execute queries with UDFs.

Bucketing Concept:

1)Set the bucketing concept

```
[cloudera@quickstart ~]$ hive
2023-12-16 23:34:06,251 WARN  [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tree module jar containing Pre

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show databases;
OK
default
kmit
Time taken: 1.077 seconds, Fetched: 2 row(s)
hive> use kmit;
OK
Time taken: 0.176 seconds
hive> show tables;
OK
employee
employee2
Time taken: 0.358 seconds, Fetched: 2 row(s)
hive> set hive.enforce.bucketing=true;
```

2)To see the current value of any property

```
hive> set hive.enforce.bucketing=true;
hive> set hive.enforce.bucketing;
```

3) See the employee table data

```
hive> select * from employee;
OK
tejas   150000.0        IT      INDIA   KAR
mohammad        200000.0        IT      INDIA   KAR
ram     2000000.0       HR      INDIA   KAR
Naresh  2000000.0       FIN     INDIA   KAR
srinivas        250000.0        IT      INDIA   MH
akhilesh        250000.0        IT      INDIA   MH
aswini  250000.0        IT      INDIA   MH
Time taken: 1.412 seconds, Fetched: 7 row(s)
```

4) Create the bucketed table

```
hive> create table bucketed_table(name string,salary float,dept string,country string,state string)
    > clustered by(dept) into 4 buckets;
OK
Time taken: 0.616 seconds
```

5) Insert data from the table employee to bucketed table

```
hive> insert overwrite table bucketed_table
    > select * from employee;
Query ID = cloudera_20231216234343_9ab46110-4f13-485a-83f1-dd67504c656a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1698334154939_0006, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1698334154939
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1698334154939_0006
```

6) Read the data from the bucketed_table

```
hive> select * from bucketed_table
    > tablesample(bucket 3 out of 4 on dept);
OK
ram     2000000.0       HR      INDIA   KAR
Time taken: 0.367 seconds, Fetched: 1 row(s)
```

7) To view the structure of the file system

```
hive> quit;
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/kmit.db/bucketed_table
Found 4 items
-rwxrwxrwx   1 cloudera supergroup          0 2023-12-16 23:44 /user/hive/warehouse/kmit.db/bucketed_table/000000_0
-rwxrwxrwx   1 cloudera supergroup          0 2023-12-16 23:44 /user/hive/warehouse/kmit.db/bucketed_table/000001_0
-rwxrwxrwx   1 cloudera supergroup         27 2023-12-16 23:44 /user/hive/warehouse/kmit.db/bucketed_table/000002_0
-rwxrwxrwx   1 cloudera supergroup        178 2023-12-16 23:44 /user/hive/warehouse/kmit.db/bucketed_table/000003_0
[cloudera@quickstart ~]$
```

UDF in Hive

1.Create .java file

```
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.IntWritable;
public class Square extends UDF {
    public IntWritable evaluate(IntWritable input)
    {
        if(input ==null)
            return null;
        return new IntWritable(input.get()*input.get());
    }

}
```

2.Export as jar file SquareUDF.jar

3. Create a file

vim demo.txt

4.Create a table “demo”

```
hive > create table demo(id int)
    > row format delimited
      > fields terminated by ',';
```

5.Then type the above queries in Hive

```
hive> load data local inpath 'demo.txt' into table demo;
Loading data to table default.demo
Table default.demo stats: [numFiles=2, totalSize=38]
OK
Time taken: 1.492 seconds
hive> ADD JAR /home/cloudera/workspace/UDF.jar;
Added [/home/cloudera/workspace/UDF.jar] to class path
Added resources: [/home/cloudera/workspace/UDF.jar]
hive>  CREATE TEMPORARY FUNCTION square as 'Square';
OK
Time taken: 0.014 seconds
hive> select square(id) As squared_value
    > from demo;
OK
1
4
9
16
```

**** UDAF****

1.Create a java file

```
import org.apache.hadoop.hive.ql.exec.UDAF;
public class Maximum  extends UDAF{
    public static class MaximumIntUDAFEvaluator implements UDAFEvaluator
    {
        private IntWritable result;
        public void init()
        {
            result = null;
        }
        public boolean iterate(IntWritable value)
        {
            if(value == null)
            {
                return true;
            }
            if(result == null)
            {
                result =new IntWritable(value.get());
            }
            else
            {
                result.set(Math.max(result.get(),value.get()));
            }
            return true;
        }
        public IntWritable terminatePartial()
        {
            return result;
        }
        public boolean merge (IntWritable other)
        {
            return iterate(other);
        }
        public IntWritable terminate(){
            return result;
        }
    }
```

2. Then type the above queries in Hive

```
hive> ADD JAR /home/cloudera/workspace/MaximumUDF.jar;
Added [/home/cloudera/workspace/MaximumUDF.jar] to class path
Added resources: [/home/cloudera/workspace/MaximumUDF.jar]
hive>  CREATE TEMPORARY FUNCTION max as 'Maximum';
OK
Time taken: 0.01 seconds
hive> select max(id) As max__value
    > from demo;
Query ID = cloudera_20231217001313_ddb4d897-def2-4079-9661-cfb10dd8f874
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1698334154939_0007, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1698334154939_0007/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1698334154939_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-12-17 00:13:18,800 Stage-1 map = 0%,  reduce = 0%
2023-12-17 00:13:28,817 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.98 sec
2023-12-17 00:13:38,211 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.87 sec
MapReduce Total cumulative CPU time: 3 seconds 870 msec
Ended Job = job_1698334154939_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.87 sec   HDFS Read: 6570 HDFS Write: 3 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 870 msec
OK
51
Time taken: 37.765 seconds, Fetched: 1 row(s)
hive>
```

EXPERIMENT 11

AIM :- Implement flume script for spooling a directory from local file system to HDFS.

PART 1 :- Flume configuration using a spooling directory source (Local File System) and a logger (Console) sink

(1) Create a file named spool.properties

```
[cloudera@quickstart ~]$ vim spool.properties
[cloudera@quickstart ~]$ cat spool.properties
agent1.sources = source1
agent1.sinks = sink1
agent1.channels = channel1

agent1.sources.source1.channels = channel1
agent1.sinks.sink1.channel = channel1

agent1.sources.source1.type = spooldir
agent1.sources.source1.spoolDir = /home/cloudera/spooldir

agent1.sinks.sink1.type = logger
agent1.channels.channel1.type = file
[cloudera@quickstart ~]$
```

(2) Create a directory named spooldir

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ mkdir spooldir
```

(3) Create a file named input

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ vim input
[cloudera@quickstart ~]$ cat input
hi hello hi hello
hi hi hi hi
[cloudera@quickstart ~]$
```

(4) Run the flume command to execute spool.properties

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ flume-ng agent \
> --conf-file spool.properties \
> --name agent1 \
> --conf $FLUME_HOME/conf \
> -DFlume.root.logger=INFO,console
```

```
23/12/15 20:41:46 INFO file.FileChannel: Queue Size after replay: 0 [channel=channel1]
23/12/15 20:41:46 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: CHANNEL, name: channel1: Successfully r
egistered new MBean.
23/12/15 20:41:46 INFO instrumentation.MonitoredCounterGroup: Component type: CHANNEL, name: channel1 started
23/12/15 20:41:46 INFO node.Application: Starting Sink sink1
23/12/15 20:41:46 INFO node.Application: Starting Source source1
23/12/15 20:41:46 INFO source.SpoolDirectorySource: SpoolDirectorySource source starting with directory: /home/cloudera/spooldir
23/12/15 20:41:46 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: SOURCE, name: source1: Successfully reg
istered new MBean.
23/12/15 20:41:46 INFO instrumentation.MonitoredCounterGroup: Component type: SOURCE, name: source1 started
23/12/15 20:41:47 INFO avro.ReliableSpoolingFileEventReader: Last read took us just up to a file boundary. Rolling to the next file, if
 there is one.
23/12/15 20:41:47 INFO avro.ReliableSpoolingFileEventReader: Preparing to move file /home/cloudera/spooldir/hdfsflume to /home/cloudera
/spooldir/hdfsflume.COMPLETED
```

(5) Now copy the input file from other terminal

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ cp input spooldir/
[cloudera@quickstart ~]$
```

(6) Now check the output we find flume is reading the data from spooldir and displaying output on console

```
istered new MBean.
23/12/15 21:02:01 INFO instrumentation.MonitoredCounterGroup: Component type: SOURCE, name: source1 started
23/12/15 21:02:31 INFO avro.ReliableSpoolingFileEventReader: Last read took us just up to a file boundary. Rolling to the next file, if
 there is one.
23/12/15 21:02:31 INFO avro.ReliableSpoolingFileEventReader: Preparing to move file /home/cloudera/spooldir/input to /home/cloudera/spo
oldir/input.COMPLETED
23/12/15 21:02:31 INFO sink.LoggerSink: Event: { headers:{} body: 68 65 6C 6C 6F 20 68 69 20 68 65 6C 6C 6F       hello hi hello }
23/12/15 21:02:31 INFO sink.LoggerSink: Event: { headers:{} body: 68 65 6C 6C 6F 20 68 69 20 68 69 20 68 69 20 68 hello hi hi hi h }
23/12/15 21:02:31 INFO sink.LoggerSink: Event: { headers:{} body: }
```

PART 2 :- Flume configuration using a spooling directory source (Local File System) and a HDFS (hdfs) sink

(1) Create a file named hdfs.properties

```
[cloudera@quickstart ~]$ vim hdfs.properties
[cloudera@quickstart ~]$ cat hdfs.properties
agent1.sources = source1

agent1.sinks = sink1
agent1.channels = channel1

agent1.sources.source1.channels = channel1
agent1.sinks.sink1.channel = channel1

agent1.sources.source1.type = spooldir
agent1.sources.source1.spoolDir = /home/cloudera/spooldir
agent1.sinks.sink1.type = hdfs
agent1.sinks.sink1.hdfs.path = /hdfsdir
agent1.sinks.sink1.hdfs.filePrefix = events
agent1.sinks.sink1.hdfs.fileSuffix = .log
agent1.sinks.sink1.hdfs.inUsePrefix = _
agent1.sinks.sink1.hdfs.fileType = DataStream

agent1.channels.channel1.type = file
[cloudera@quickstart ~]$
```

(2) Create the directory on hdfs

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir /hdfsdir
[cloudera@quickstart ~]$
```

(3) Create a file named input

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ vim hdfsinput
[cloudera@quickstart ~]$ cat hdfsinput
i hello hii
hello hii hii
[cloudera@quickstart ~]$
```

(4) Run the flume command to execute hdfs.properties

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ flume-ng agent \
> --conf-file hdfs.properties \
> --name agent1 \
> --conf $FLUME_HOME/conf \
> -Dflume.root.logger=INFO,hdfs
```

```
23/12/15 21:18:37 INFO instrumentation.MonitoredCounterGroup: Component type: CHANNEL, name: channel1 started
23/12/15 21:18:37 INFO node.Application: Starting Sink sink1
23/12/15 21:18:37 INFO node.Application: Starting Source source1
23/12/15 21:18:37 INFO source.SpoolDirectorySource: SpoolDirectorySource source starting with directory: /home/cloudera/spooldir
23/12/15 21:18:37 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: SINK, name: sink1: Successfully registe
red new MBean.
23/12/15 21:18:37 INFO instrumentation.MonitoredCounterGroup: Component type: SINK, name: sink1 started
23/12/15 21:18:37 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type: SOURCE, name: source1: Successfully reg
istered new MBean.
23/12/15 21:18:37 INFO instrumentation.MonitoredCounterGroup: Component type: SOURCE, name: source1 started
```

(5) Now copy the hdfsinput file from other terminal

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ cp hdfsinput spooldir/
[cloudera@quickstart ~]$
```

(6) Now check the output we find flume is reading the data from spooldir and displaying output on console

```
23/12/15 21:26:36 INFO file.Log: Updated checkpoint for file: /home/cloudera/.flume/file-channel/data/log-2 position: 266 logWriteOrder
ID: 1702703916853
23/12/15 21:26:55 INFO hdfs.BucketWriter: Closing /hdfsdir/_events.1702704382246.log.tmp
23/12/15 21:26:55 INFO hdfs.BucketWriter: Renaming /hdfsdir/_events.1702704382246.log.tmp to /hdfsdir/events.1702704382246.log
23/12/15 21:26:55 INFO hdfs.HDFSEventSink: Writer callback called.
```

(7) Now check the file On HDFS under /hdfsdir

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ hdfs dfs -ls /hdfsdir
Found 1 items
-rw-r--r--   1 cloudera supergroup         26 2023-12-15 21:26 /hdfsdir/events.1702704382246.log
[cloudera@quickstart ~]$
```

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ hdfs dfs -cat /hdfsdir/events.1702704382246.log
i hello hii
hello hii hii
[cloudera@quickstart ~]$
```

PART 3 :- Flume configuration using a spooling directory source (Local File System) and a logger (Console) sink as well as HDFS sink

(1) Create a file named fan.properties

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ vim fan.properties
[cloudera@quickstart ~]$ cat fan.properties
gent1.sources = source1
agent1.sinks = sink1a sink1b
agent1.channels = channel1a channel1b

agent1.sources.source1.channels = channel1a channel1b
agent1.sinks.sink1a.channel = channel1a
agent1.sinks.sink1b.channel = channel1b

agent1.sources.source1.type = spooldir
agent1.sources.source1.spoolDir = /home/cloudera/spooldir
agent1.sinks.sink1a.type = hdfs
agent1.sinks.sink1a.hdfs.path = /hdfsdir
agent1.sinks.sink1a.hdfs.filePrefix = events
agent1.sinks.sink1a.hdfs.fileSuffix = .log

agent1.sinks.sink1a.hdfs.fileType = DataStream
agent1.sinks.sink1b.type = logger

agent1.channels.channel1a.type = file

agent1.channels.channel1b.type = memory
[cloudera@quickstart ~]$
```

2) Create a file named bothinput

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ vim bothinput
[cloudera@quickstart ~]$ cat bothinput
ii hello hiii
welcomee hii
[cloudera@quickstart ~]$
```

3) Run the flume command to execute fan.properties

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ flume-ng agent \
> --conf-file fan.properties \
> --name agent1 \
> --conf $FLUME_HOME/conf \
> -Dflume.root.logger=INFO,console,hdfs
```

(5) Now copy the hdfsinput file from other terminal

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ cp bothinput spooldir/
[cloudera@quickstart ~]$
```

6) Check the output

On console the output:

```
23/12/16 02:53:10 INFO sink.LoggerSink: Event: { headers:{} body: 69 68 69 69 69 69 69 69 69 20 68 65 6C 6F 6F
   ihiiiiiii heloo }
23/12/16 02:53:10 INFO sink.LoggerSink: Event: { headers:{} body: 68 65 6C 6C 6F 20 77 65 6C 63 6F 6D 65
   hello welcome }
23/12/16 02:53:13 INFO hdfs.HDFSDataStream: Serializer = TEXT, UseRawLocalFileSystem = false
23/12/16 02:53:13 INFO hdfs.BucketWriter: Creating /hdfsdir/events.1702723993063.log.tmp
23/12/16 02:53:23 INFO file.EventQueueBackingStoreFile: Start checkpoint for /home/cloudera/.flume/file-channel
/checkpoint/checkpoint, elements to sync = 2
```

On the hdfs output:

```
23/12/16 02:53:43 INFO hdfs.BucketWriter: Closing /hdfsdir/events.1702723993063.log.tmp
23/12/16 02:53:43 INFO hdfs.BucketWriter: Renaming /hdfsdir/events.1702723993063.log.tmp to /hdfsdir/events.170
2723993063.log
23/12/16 02:53:43 INFO hdfs.HDFSEventSink: Writer callback called.
```

```
[cloudera@quickstart ~]$ hdfs dfs -ls /hdfsdir
Found 3 items
-rw-r--r--   1 cloudera supergroup         26 2023-12-15 21:26 /hdfsdir/events.1702704382246.log
-rw-r--r--   1 cloudera supergroup         50 2023-12-16 02:50 /hdfsdir/events.1702723825138.log
-rw-r--r--   1 cloudera supergroup         30 2023-12-16 02:53 /hdfsdir/events.1702723993063.log

[cloudera@quickstart ~]$ hdfs dfs -cat /hdfsdir/events.1702723993063.log
ihiiiiiii heloo
hello welcome
[cloudera@quickstart ~]$
```

EXPERIMENT 12

AIM :- Import selective data from database into a specific directory in HDFS and analyze it. Export the summarized data into another table of database.

SYNTAX :- sqoop

(1) To view the version of the sqoop

```
[cloudera@quickstart ~]$ sqoop version
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
23/12/15 22:12:31 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.8.0
Sqoop 1.4.6-cdh5.8.0
git commit id
Compiled by jenkins on Thu Jun 16 12:25:21 PDT 2016
[cloudera@quickstart ~]$
```

(2) To view all the sqoop commands

```
[cloudera@quickstart ~]$ sqoop help
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
23/12/15 22:09:01 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.8.0
usage: sqoop COMMAND [ARGS]

Available commands:
  codegen            Generate code to interact with database records
  create-hive-table  Import a table definition into Hive
  eval               Evaluate a SQL statement and display the results
  export             Export an HDFS directory to a database table
  help               List available commands
  import             Import a table from a database to HDFS
  import-all-tables  Import tables from a database to HDFS
  import-mainframe   Import datasets from a mainframe server to HDFS
  job                Work with saved jobs
  list-databases     List available databases on a server
```

(3) To get into the sqoop command shell

The password for mysql is cloudera

```
[cloudera@quickstart ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 33
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>
```

(4) Create a database kmit

```
mysql> create database kmit;
Query OK, 1 row affected (0.02 sec)

mysql> use kmit;
Database changed
mysql>
```

```
mysql> GRANT ALL PRIVILEGES ON kmit.* TO ''@'localhost';

Query OK, 0 rows affected (0.00 sec)
```

(5) CREATE a table employee

```
mysql> create table employee(id int, name varchar(20), dept varchar(20));
Query OK, 0 rows affected (0.09 sec)

mysql>
```

(6) Insert data into the table

```
mysql> insert into employee values(101,'Raj','DS');
Query OK, 1 row affected (0.02 sec)

mysql> insert into employee values(102,'Sai','CSE');
Query OK, 1 row affected (0.00 sec)

mysql> insert into employee values(103,'Vedika','DS');
Query OK, 1 row affected (0.01 sec)

mysql> insert into employee values(104,'Tanu','DS');
Query OK, 1 row affected (0.01 sec)

mysql> insert into employee values(105,'Varun','CSM');
Query OK, 1 row affected (0.01 sec)
```

(7) CREATE a table employee1

```
mysql> create table employee1(id int, salary int);
Query OK, 0 rows affected (0.02 sec)

mysql>
```

(8) Read the data from employee

```
mysql> select * from employee;
+------+--------+------+
| id   | name   | dept |
+------+--------+------+
|  101 | Raj    | DS   |
|  102 | Sai    | CSE  |
|  103 | Vedika | DS   |
|  104 | Tanu   | DS   |
|  105 | Varun  | CSM  |
+------+--------+------+
5 rows in set (0.00 sec)
```

(9) quit mysql

```
mysql> quit
Bye
[cloudera@quickstart ~]$
```

(10) Transfering an entire table

```
[cloudera@quickstart ~]$ sqoop import \
> --connect jdbc:mysql://localhost/kmit \
> --username root \
> --password cloudera \
> --table employee \
> --m 1
```

```
                Total committed heap usage (bytes)=60751872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=63
23/12/15 22:50:36 INFO mapreduce.ImportJobBase: Transferred 63 bytes in 35.7249 seconds (1.7635 bytes/sec)
23/12/15 22:50:36 INFO mapreduce.ImportJobBase: Retrieved 5 records.
[cloudera@quickstart ~]$
```

(11) Checking if table is imported or not

```
[cloudera@quickstart ~]$ hdfs dfs -ls employee
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2023-12-15 22:50 employee/_SUCCESS
-rw-r--r--   1 cloudera cloudera         63 2023-12-15 22:50 employee/part-m-00000
[cloudera@quickstart ~]$ hdfs dfs -cat employee/part-m-00000
101,Raj,DS
102,Sai,CSE
103,Vedika,DS
104,Tanu,DS
105,Varun,CSM
[cloudera@quickstart ~]$
```

(12) Transferring an entire table to a specified directory

```
[cloudera@quickstart ~]$ sqoop import \
> --connect jdbc:mysql://localhost/kmit \
> --username root \
> --password cloudera \
> --table employee \
> --target-dir /specific --m 1
```

```
                Total committed heap usage (bytes)=60751872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=63
23/12/15 23:02:10 INFO mapreduce.ImportJobBase: Transferred 63 bytes in 25.0826 seconds (2.5117 bytes/sec)
23/12/15 23:02:10 INFO mapreduce.ImportJobBase: Retrieved 5 records.
[cloudera@quickstart ~]$
```

(13) Checking if table is imported or not

```
[cloudera@quickstart ~]$ hdfs dfs -ls /specific
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2023-12-15 23:02 /specific/_SUCCESS
-rw-r--r--   1 cloudera supergroup         63 2023-12-15 23:02 /specific/part-m-00000
[cloudera@quickstart ~]$ hdfs dfs -cat /specific/part-m-00000
101,Raj,DS
102,Sai,CSE
103,Vedika,DS
104,Tanu,DS
105,Varun,CSM
[cloudera@quickstart ~]$
```

(14) Importing a subset of data

```
[cloudera@quickstart ~]$ sqoop import \
> --connect jdbc:mysql://localhost/kmit \
> --username root \
> --password cloudera \
> --table employee \
> --target-dir /specific1 --where "id<103" --m 1
```

```
                Total committed heap usage (bytes)=60751872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=23
23/12/15 23:04:12 INFO mapreduce.ImportJobBase: Transferred 23 bytes in 22.5257 seconds (1.0211 bytes/sec)
23/12/15 23:04:12 INFO mapreduce.ImportJobBase: Retrieved 2 records.
[cloudera@quickstart ~]$
```

(15) Checking if table subset is imported or not

```
[cloudera@quickstart ~]$ hdfs dfs -ls /specific1
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2023-12-15 23:04 /specific1/_SUCCESS
-rw-r--r--   1 cloudera supergroup         23 2023-12-15 23:04 /specific1/part-m-00000
[cloudera@quickstart ~]$ hdfs dfs -cat /specific1/part-m-00000
101,Raj,DS
102,Sai,CSE
[cloudera@quickstart ~]$
```

(16) Importing all tables from database to HDFS using Import-all-tables

```
[cloudera@quickstart ~]$ hdfs dfs -rm -r employee
Deleted employee
[cloudera@quickstart ~]$ sqoop import-all-tables \
> --connect jdbc:mysql://localhost/kmit \
> --username root \
> --password cloudera \
> --m 1
```

```
                Total committed heap usage (bytes)=60751872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=51
23/12/15 23:19:19 INFO mapreduce.ImportJobBase: Transferred 51 bytes in 27.4257 seconds (1.8596 bytes/sec)
23/12/15 23:19:19 INFO mapreduce.ImportJobBase: Retrieved 5 records.
[cloudera@quickstart ~]$
```

(17) Checking if all tables are imported or not

```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 2 items
drwxr-xr-x   - cloudera cloudera          0 2023-12-15 23:18 employee
drwxr-xr-x   - cloudera cloudera          0 2023-12-15 23:19 employee1
[cloudera@quickstart ~]$
```

To import data between Mysql and Hive tables

Terminal 1:

(1) Create a database kmit

```
mysql> create database kmit;
Query OK, 1 row affected (0.02 sec)

mysql> use kmit;
Database changed
mysql>
```

(2) CREATE a table employee

```
mysql> create table employee(id int, name varchar(20), dept varchar(20));
Query OK, 0 rows affected (0.09 sec)

mysql>
```

(3) Insert data into the table

```
mysql> insert into employee values(101,'Raj','DS');
Query OK, 1 row affected (0.02 sec)

mysql> insert into employee values(102,'Sai','CSE');
Query OK, 1 row affected (0.00 sec)

mysql> insert into employee values(103,'Vedika','DS');
Query OK, 1 row affected (0.01 sec)

mysql> insert into employee values(104,'Tanu','DS');
Query OK, 1 row affected (0.01 sec)

mysql> insert into employee values(105,'Varun','CSM');
Query OK, 1 row affected (0.01 sec)
```

Terminal 2:

(1)Creating a database in hive

```
File  Edit  View  Search  Terminal  Help
hive> create database hi;
OK
Time taken: 1.105 seconds
hive>
```

```
File  Edit  View  Search  Terminal  Help
hive> show databases;
OK
default
hi
hive
kmit
Time taken: 0.838 seconds, Fetched: 4 row(s)
hive>
```

Terminal 3:

(1)Importing table from MySql to hive

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ sqoop import \
> --connect jdbc:mysql://localhost/kmit \
> --username root \
> --password cloudera \
> --table employee \
> --target-dir /hiveimport \
> --fields-terminated-by "," \
> --m 1 \
> --hive-import \
> --create-hive-table \
> --hive-table hi.employee
```

```
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=102
23/12/21 22:58:49 INFO mapreduce.ImportJobBase: Transferred 102 bytes in 26.0055 seconds (3.9222 bytes/sec)
23/12/21 22:58:49 INFO mapreduce.ImportJobBase: Retrieved 8 records.
23/12/21 22:58:49 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `employee` AS t LIMIT 1
23/12/21 22:58:49 INFO hive.HiveImport: Loading uploaded data into Hive
```

Terminal 2 :

(1)Checking if our table has been imported to hive

```
File  Edit  View  Search  Terminal  Help
hive> use hi;
OK
Time taken: 0.037 seconds
hive> show tables;
OK
employee
Time taken: 0.113 seconds, Fetched: 1 row(s)
hive> select * from employee;
OK
101     Raj     DS
102     Sai     CSE
103     Vedika  DS
104     Tanu    DS
105     Varun   CSM
106     Rajesh  IT
107     Raju    IT
108     Ramu    CSM
Time taken: 0.608 seconds, Fetched: 8 row(s)
hive>
```

Exporting data from HDFS to RDBMS

(1)Deleting the table employee from MySql

```
Browse and run installed applications
File  Edit  View  Search  Terminal  Help
mysql> select * from employee;
+------+--------+------+
| id   | name   | dept |
+------+--------+------+
|  101 | Raj    | DS   |
|  102 | Sai    | CSE  |
|  103 | Vedika | DS   |
|  104 | Tanu   | DS   |
|  105 | Varun  | CSM  |
|  106 | Rajesh | IT   |
|  107 | Raju   | IT   |
|  108 | Ramu   | CSM  |
+------+--------+------+
8 rows in set (0.00 sec)

mysql> delete from employee;
Query OK, 8 rows affected (0.01 sec)

mysql> select * from employee;
Empty set (0.00 sec)

mysql>
```

(2)Using sqoop export command to export table from hdfs to MySql

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ sqoop export \
> --connect jdbc:mysql://localhost/kmit \
> --username root \
> --password cloudera \
> --table employee \
> --export-dir /specific \
> --m 1
```

```
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
23/12/21 23:15:52 INFO mapreduce.ExportJobBase: Transferred 200 bytes in 26.2173 seconds (7.6286 bytes/sec)
23/12/21 23:15:52 INFO mapreduce.ExportJobBase: Exported 5 records.
[cloudera@quickstart ~]$
```

cloudera@quickstart:~ cloudera@quickstart:~ cloudera@quickstart:~

File Edit View Search Terminal Help

```
mysql> select * from employee;
+------+--------+------+
| id   | name   | dept |
+------+--------+------+
|  101 | Raj    | DS   |
|  102 | Sai    | CSE  |
|  103 | Vedika | DS   |
|  104 | Tanu   | DS   |
|  105 | Varun  | CSM  |
+------+--------+------+
5 rows in set (0.01 sec)

mysql>
```

Generating code

File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ sqoop codegen\
> --connect jdbc:mysql://localhost/kmit \
> --username root \
> --password cloudera \
> --table employee \
> --class-name Emp
```

File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ ls
abc1.txt             Emp.java                    hdfsflume            kmit.macro            pig_1700723356470.log  qwerty
abc.txt              employee1.java              hdfs.properties      lib                   pig_1700799386138.log  Rollno1.java
cloudera-manager     employee.java               home                 local                 pig_1700892957485.log  script.pig
cm_api.py            emp.txt                     hs_err_pid18347.log  Music                 pig_1700895436294.log  spooldir
console.properties   enterprise-deployment.json  input                parcels               pig_1700895747290.log  spool.properties
demo                 express-deployment.json     input1.txt           Pictures              pig_1700896054231.log  Templates
Desktop              fil                         input.txt            pig1                  pig_1700896117329.log  trishdir
Documents            file                        kerberos             pig_1700196771643.log pig_1700896180094.log  veds
Downloads            file1                       kmit                 pig_1700547851987.log pig_1700896632360.log  Videos
eclipse              fileout                     kmit1.pig            pig_1700549645403.log Public                workspace
[cloudera@quickstart ~]$ cat Emp.java
```