AIT - Management

<u>PRACTICAL COURSE FILE</u>

MBA-BA

BIG DATA TECHNOLOGIES LAB

(Subject Code: **22BBH-704)**

(L:2 T:0 P:2 C:3)

Instructor: DR. ANAND SHARMA

Chandigarh University

List of Experiments

S.No	Objective of the Experiments
1	To implement the file management tasks in Hadoop: Adding files and directories, Retrieving files, Deleting files
2	Install and Run Hive then use Hive to create, load, alter, and drop databases, tables.
3	Implement Hive Partitioning with data set
4	Implement Hive bucketing with data set.
5	Implement sqoop commands
6	Working on POC with dataset
7	Implement Hbase commands with data set.
8	Install and Run Pig then write Pig Latin scripts to sort, group, join and filter your data.
9	Using Fluid Query with Big SQL.
10	Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm with data set

Experiment-1

Aim: Implement the following file management tasks in Hadoop:

(i) Adding files and directories
(ii) Retrieving files
(iii) Deleting files

The experiment was finished with the following steps:

(i) Adding files and directories

Step-1:
First, the command “hadoop fs -ls /user”was used to return the list of directories and each directory’s direct children.

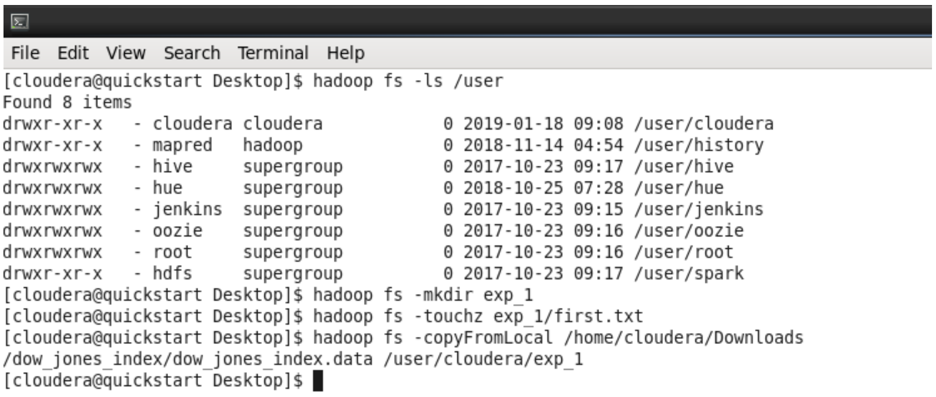
Step 2:
Next, the command ‘hadoop fs -mkdir exp_1’ is used to create a new directory in the Hadoop file system.

Step 3:
Next, the command ‘hadoop fs -touchz exp_1/first.txt’ is used to make a new file (with zero length) in the newly created directory exp_1.

(ii) Retrieving files

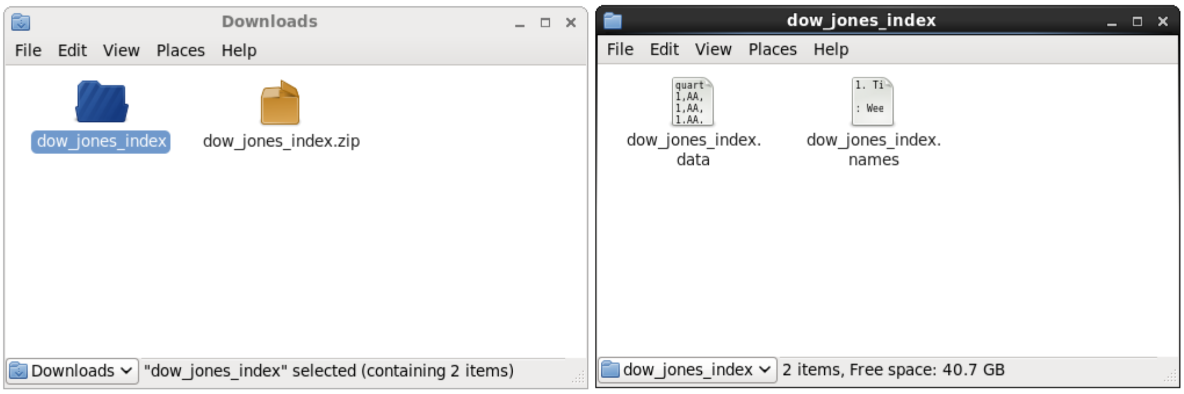
Step 4:
Then,usingthe command ‘hadoop fs -copyFromLocal /home/cloudera/Downloads/dow_jones_index/dow_jones_index.data /user/cloudea/exp_1’ is used to copy the downloaded data set form the UCI repository online, stored currently in the Local File system’s Downloads directory to the exp_1 directory in the Hadoop File System.

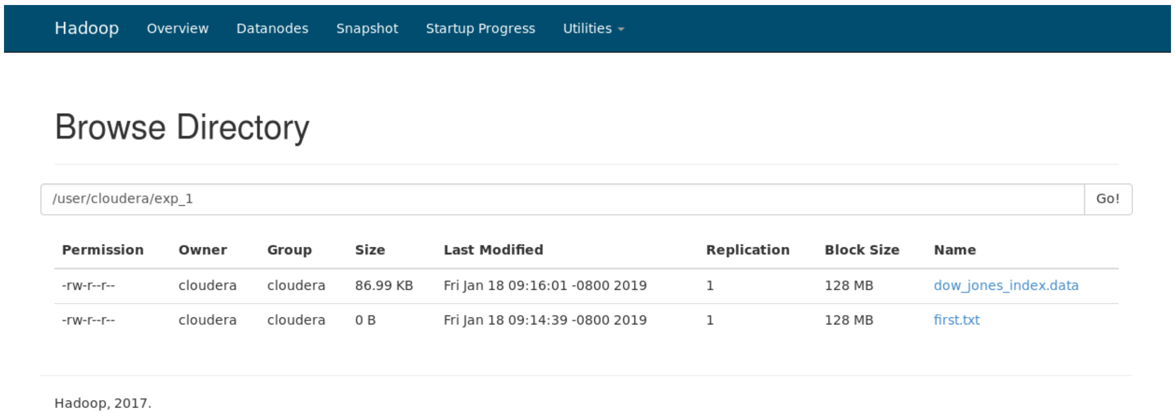
The demonstration of all the steps described above is shown below;



```
[cloudera@quickstart Desktop]$ hadoop fs -ls /user
Found 8 items
drwxr-xr-x   - cloudera cloudera          0 2019-01-18 09:08 /user/cloudera
drwxr-xr-x   - mapred   hadoop            0 2018-11-14 04:54 /user/history
drwxrwxrwx   - hive     supergroup        0 2017-10-23 09:17 /user/hive
drwxrwxrwx   - hue      supergroup        0 2018-10-25 07:28 /user/hue
drwxrwxrwx   - jenkins  supergroup        0 2017-10-23 09:15 /user/jenkins
drwxrwxrwx   - oozie    supergroup        0 2017-10-23 09:16 /user/oozie
drwxrwxrwx   - root     supergroup        0 2017-10-23 09:16 /user/root
drwxr-xr-x   - hdfs     supergroup        0 2017-10-23 09:17 /user/spark
[cloudera@quickstart Desktop]$ hadoop fs -mkdir exp_1
[cloudera@quickstart Desktop]$ hadoop fs -touchz exp_1/first.txt
[cloudera@quickstart Desktop]$ hadoop fs -copyFromLocal /home/cloudera/Downloads
/dow_jones_index/dow_jones_index.data /user/cloudera/exp_1
[cloudera@quickstart Desktop]$ 
```

We can also do the following, by going to the localhost:50070 and browse the file system, to view the downloaded data sets and the newly created files and directories, from there. The demonstration is shown below;


Downloads
File Edit View Places Help
dow_jones_index
dow_jones_index.zip
Downloads
"dow_jones_index" selected (containing 2 items)
dow_jones_index
File Edit View Places Help
dow_jones_index.data
dow_jones_index.names
dow_jones_index
2 items, Free space: 40.7 GB

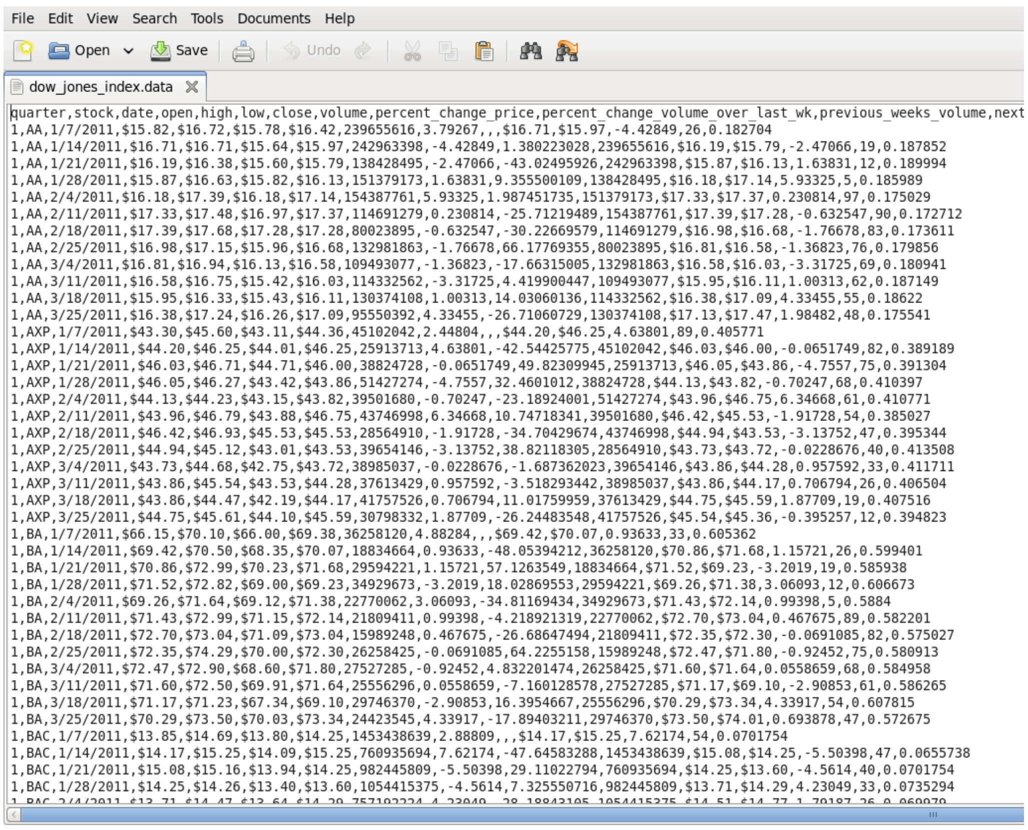


Browse Directory

/user/cloudera/exp_1

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	cloudera	86.99 KB	Fri Jan 18 09:16:01 -0800 2019	1	128 MB	dow_jones_index.data
-rw-r--r--	cloudera	cloudera	0 B	Fri Jan 18 09:14:39 -0800 2019	1	128 MB	first.txt

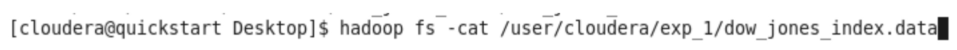
Hadoop, 2017.



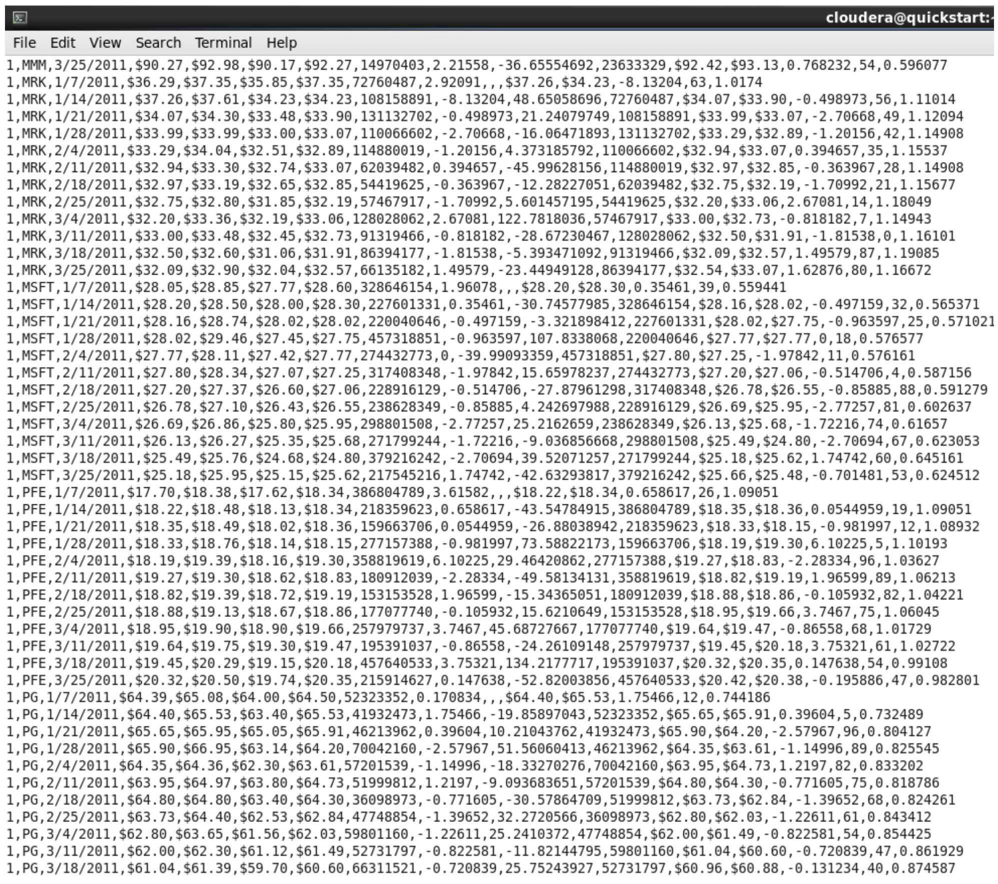
```
quarter,stock,date,open,high,low,close,volume,percent_change_price,percent_change_volume_over_last_wk,previous_weeks_volume,next
1,AA,1/7/2011,$15.82,$16.72,$15.78,$16.42,239655616,3.79267,,,$16.71,$15.97,-4.42849,26,0.182704
1,AA,1/14/2011,$16.71,$16.71,$15.64,$15.97,242963398,-4.42849,1.380223028,239655616,$16.19,$15.79,-2.47066,19,0.187852
1,AA,1/21/2011,$16.19,$16.38,$15.60,$15.79,138428495,-2.47066,-43.02495926,242963398,$15.87,$16.13,1.63831,12,0.189994
1,AA,1/28/2011,$15.87,$16.63,$15.82,$16.13,151379173,1.63831,9.355500109,138428495,$16.18,$17.14,5.93325,5,0.185989
1,AA,2/4/2011,$16.18,$17.39,$16.18,$17.14,154387761,5.93325,1.987451735,151379173,$17.33,$17.37,0.230814,97,0.175029
1,AA,2/11/2011,$17.33,$17.48,$16.97,$17.37,114691279,0.230814,-25.71219489,154387761,$17.39,$17.28,-0.632547,90,0.172712
1,AA,2/18/2011,$17.39,$17.68,$17.28,$17.28,80023895,-0.632547,-30.22669579,114691279,$16.98,$16.68,-1.76678,83,0.173611
1,AA,2/25/2011,$16.98,$17.15,$15.96,$16.68,132981863,-1.76678,66.17769355,80023895,$16.81,$16.58,-1.36823,76,0.179856
1,AA,3/4/2011,$16.81,$16.94,$16.13,$16.58,109493077,-1.36823,-17.66315005,132981863,$16.58,$16.03,-3.31725,69,0.180941
1,AA,3/11/2011,$16.58,$16.75,$15.42,$16.03,114332562,-3.31725,4.419900447,109493077,$15.95,$16.11,1.00313,62,0.187149
1,AA,3/18/2011,$15.95,$16.33,$15.43,$16.11,130374108,1.00313,14.03060136,114332562,$16.38,$17.09,4.33455,55,0.18622
1,AA,3/25/2011,$16.38,$17.24,$16.26,$17.09,95550392,4.33455,-26.71060729,130374108,$17.13,$17.47,1.98482,48,0.175541
1,AXP,1/7/2011,$43.30,$45.60,$43.11,$44.36,45102042,2.44804,,,$44.20,$46.25,4.63801,89,0.405771
1,AXP,1/14/2011,$44.20,$46.25,$44.01,$46.25,25913713,4.63801,-42.54425775,45102042,$46.03,$46.00,-0.0651749,82,0.389189
1,AXP,1/21/2011,$46.03,$46.71,$44.71,$46.00,38824728,-0.0651749,49.82309945,25913713,$46.05,$43.86,-4.7557,75,0.391304
1,AXP,1/28/2011,$46.05,$46.27,$43.42,$43.86,51427274,-4.7557,32.4601012,38824728,$44.13,$43.82,-0.70247,68,0.410397
1,AXP,2/4/2011,$44.13,$44.23,$43.15,$43.82,39501680,-0.70247,-23.18924001,51427274,$43.96,$46.75,6.34668,61,0.410771
1,AXP,2/11/2011,$43.96,$46.79,$43.88,$46.75,43746998,6.34668,10.74718341,39501680,$46.42,$45.53,-1.91728,54,0.385027
1,AXP,2/18/2011,$46.42,$46.93,$45.53,$45.53,28564910,-1.91728,-34.70429674,43746998,$44.94,$43.53,-3.13752,47,0.395344
1,AXP,2/25/2011,$44.94,$45.12,$43.01,$43.53,39654146,-3.13752,38.82118305,28564910,$43.73,$43.72,-0.0228676,40,0.413508
1,AXP,3/4/2011,$43.73,$44.68,$42.75,$43.72,38985037,-0.0228676,-1.687362023,39654146,$43.86,$44.28,0.957592,33,0.411711
1,AXP,3/11/2011,$43.86,$45.54,$43.53,$44.28,37613429,0.957592,-3.518293442,38985037,$43.86,$44.17,0.706794,26,0.406504
1,AXP,3/18/2011,$43.86,$44.47,$42.19,$44.17,41757526,0.706794,11.01759959,37613429,$44.75,$45.59,1.87709,19,0.407516
1,AXP,3/25/2011,$44.75,$45.61,$44.10,$45.59,30798332,1.87709,-26.24483548,41757526,$45.54,$45.36,-0.395257,12,0.394823
1,BA,1/7/2011,$66.15,$70.10,$66.00,$69.38,36258120,4.88284,,,$69.42,$70.07,0.93633,33,0.605362
1,BA,1/14/2011,$69.42,$70.50,$68.35,$70.07,18834664,0.93633,-48.05394212,36258120,$70.86,$71.68,1.15721,26,0.599401
1,BA,1/21/2011,$70.86,$72.99,$70.23,$71.68,29594221,1.15721,57.1263549,18834664,$71.52,$69.23,-3.2019,19,0.585938
1,BA,1/28/2011,$71.52,$72.82,$69.00,$69.23,34929673,-3.2019,18.02869553,29594221,$69.26,$71.38,3.06093,12,0.606673
1,BA,2/4/2011,$69.26,$71.64,$69.12,$71.38,22770062,3.06093,-34.81169434,34929673,$71.43,$72.14,0.99398,5,0.5884
1,BA,2/11/2011,$71.43,$72.99,$71.15,$72.14,21809411,0.99398,-4.218921319,22770062,$72.70,$73.04,0.467675,89,0.582201
1,BA,2/18/2011,$72.70,$73.04,$71.09,$73.04,15989248,0.467675,-26.68647494,21809411,$72.35,$72.30,-0.0691085,82,0.575027
1,BA,2/25/2011,$72.35,$74.29,$70.00,$72.30,26258425,-0.0691085,64.2255158,15989248,$72.47,$71.80,-0.92452,75,0.580913
1,BA,3/4/2011,$72.47,$72.90,$68.60,$71.80,27527285,-0.92452,4.832201474,26258425,$71.60,$71.64,0.0558659,68,0.584958
1,BA,3/11/2011,$71.60,$72.50,$69.91,$71.64,25556296,0.0558659,-7.160128578,27527285,$71.17,$69.10,-2.90853,61,0.586265
1,BA,3/18/2011,$71.17,$71.23,$67.34,$69.10,29746370,-2.90853,16.3954667,25556296,$70.29,$73.34,4.33917,54,0.607815
1,BA,3/25/2011,$70.29,$73.50,$70.03,$73.34,24423545,4.33917,-17.89403211,29746370,$73.50,$74.01,0.693878,47,0.572675
1,BAC,1/7/2011,$13.85,$14.69,$13.80,$14.25,1453438639,2.88809,,,$14.17,$15.25,7.62174,54,0.0701754
1,BAC,1/14/2011,$14.17,$15.25,$14.09,$15.25,760935694,7.62174,-47.64583288,1453438639,$15.08,$14.25,-5.50398,47,0.0655738
1,BAC,1/21/2011,$15.08,$15.16,$13.94,$14.25,982445809,-5.50398,29.11022794,760935694,$14.25,$13.60,-4.5614,40,0.0701754
1,BAC,1/28/2011,$14.25,$14.26,$13.40,$13.60,1054415375,-4.5614,7.325550716,982445809,$13.71,$14.29,4.23049,33,0.0735294
1,BAC,2/4/2011,$13.71,$14.47,$13.64,$14.29,757192224,4.23049,-28.18843105,1054415375,$14.51,$14.77,1.70187,26,0.069979
```

Step 5:

Next, the command 'hadoop fs -cat /user/cloudera/exp_1/dow_jones_index.data' is used to display the contents of the file, recently copied from the Local File System. The demonstration of the step is shown below;



```
[cloudera@quickstart Desktop]$ hadoop fs -cat /user/cloudera/exp_1/dow_jones_index.data
```



```
1,MMM,3/25/2011,$90.27,$92.98,$90.17,$92.27,14970403,2.21558,-36.65554692,23633329,$92.42,$93.13,0.768232,54,0.596077
1,MRK,1/7/2011,$36.29,$37.35,$35.85,$37.35,72760487,2.92091,,,$37.26,$34.23,-8.13204,63,1.0174
1,MRK,1/14/2011,$37.26,$37.61,$34.23,$34.23,108158891,-8.13204,48.65058696,72760487,$34.07,$33.90,-0.498973,56,1.11014
1,MRK,1/21/2011,$34.07,$34.30,$33.48,$33.90,131132702,-0.498973,21.24079749,108158891,$33.99,$33.07,-2.70668,49,1.12094
1,MRK,1/28/2011,$33.99,$33.99,$33.00,$33.07,110066602,-2.70668,-16.06471893,131132702,$33.29,$32.89,-1.20156,42,1.14908
1,MRK,2/4/2011,$33.29,$34.04,$32.51,$32.89,114880019,-1.20156,4.373185792,110066602,$32.94,$33.07,0.394657,35,1.15537
1,MRK,2/11/2011,$32.94,$33.30,$32.74,$33.07,62039482,0.394657,-45.99628156,114880019,$32.97,$32.85,-0.363967,28,1.14908
1,MRK,2/18/2011,$32.97,$33.19,$32.65,$32.85,54419625,-0.363967,-12.28227051,62039482,$32.75,$32.19,-1.70992,21,1.15677
1,MRK,2/25/2011,$32.75,$32.80,$31.85,$32.19,57467917,-1.70992,5.601457195,54419625,$32.20,$33.06,2.67081,14,1.18049
1,MRK,3/4/2011,$32.20,$33.36,$32.19,$33.06,128028062,2.67081,122.7818036,57467917,$33.00,$32.73,-0.818182,7,1.14943
1,MRK,3/11/2011,$33.00,$33.48,$32.45,$32.73,91319466,-0.818182,-28.67230467,128028062,$32.50,$31.91,-1.81538,0,1.16101
1,MRK,3/18/2011,$32.50,$32.60,$31.06,$31.91,86394177,-1.81538,-5.393471092,91319466,$32.09,$32.57,1.49579,87,1.19085
1,MRK,3/25/2011,$32.09,$32.90,$32.04,$32.57,66135182,1.49579,-23.44949128,86394177,$32.54,$33.07,1.62876,80,1.16672
1,MSFT,1/7/2011,$28.05,$28.85,$27.77,$28.60,328646154,1.96078,,,$28.20,$28.30,0.35461,39,0.559441
1,MSFT,1/14/2011,$28.20,$28.50,$28.00,$28.30,227601331,0.35461,-30.74577985,328646154,$28.16,$28.02,-0.497159,32,0.565371
1,MSFT,1/21/2011,$28.16,$28.74,$28.02,$28.02,220040646,-0.497159,-3.321898412,227601331,$28.02,$27.75,-0.963597,25,0.571021
1,MSFT,1/28/2011,$28.02,$29.46,$27.45,$27.75,457318851,-0.963597,107.8338068,220040646,$27.77,$27.77,0,18,0.576577
1,MSFT,2/4/2011,$27.77,$28.11,$27.42,$27.77,274432773,0,-39.99093359,457318851,$27.80,$27.25,-1.97842,11,0.576161
1,MSFT,2/11/2011,$27.80,$28.34,$27.07,$27.25,317408348,-1.97842,15.65978237,274432773,$27.20,$27.06,-0.514706,4,0.587156
1,MSFT,2/18/2011,$27.20,$27.37,$26.60,$27.06,228916129,-0.514706,-27.87961298,317408348,$26.78,$26.55,-0.85885,88,0.591279
1,MSFT,2/25/2011,$26.78,$27.10,$26.43,$26.55,238628349,-0.85885,4.242697988,228916129,$26.69,$25.95,-2.77257,81,0.602637
1,MSFT,3/4/2011,$26.69,$26.86,$25.80,$25.95,298801508,-2.77257,25.2162659,238628349,$26.13,$25.68,-1.72216,74,0.61657
1,MSFT,3/11/2011,$26.13,$26.27,$25.35,$25.68,271799244,-1.72216,-9.036856668,298801508,$25.49,$24.80,-2.70694,67,0.623053
1,MSFT,3/18/2011,$25.49,$25.76,$24.68,$24.80,379216242,-2.70694,39.52071257,271799244,$25.18,$25.62,1.74742,60,0.645161
1,MSFT,3/25/2011,$25.18,$25.95,$25.15,$25.62,217545216,1.74742,-42.63293817,379216242,$25.66,$25.48,-0.701481,53,0.624512
1,PFE,1/7/2011,$17.70,$18.38,$17.62,$18.34,386804789,3.61582,,,$18.22,$18.34,0.658617,26,1.09051
1,PFE,1/14/2011,$18.22,$18.48,$18.13,$18.34,218359623,0.658617,-43.54784915,386804789,$18.35,$18.36,0.0544959,19,1.09051
1,PFE,1/21/2011,$18.35,$18.49,$18.02,$18.36,159663706,0.0544959,-26.88038942,218359623,$18.33,$18.15,-0.981997,12,1.08932
1,PFE,1/28/2011,$18.33,$18.76,$18.14,$18.15,277157388,-0.981997,73.58822173,159663706,$18.19,$19.30,6.10225,5,1.10193
1,PFE,2/4/2011,$18.19,$19.39,$18.16,$19.30,358819619,6.10225,29.46420862,277157388,$19.27,$18.83,-2.28334,96,1.03627
1,PFE,2/11/2011,$19.27,$19.30,$18.62,$18.83,180912039,-2.28334,-49.58134131,358819619,$18.82,$19.19,1.96599,89,1.06213
1,PFE,2/18/2011,$18.82,$19.39,$18.72,$19.19,153153528,1.96599,-15.34365051,180912039,$18.88,$18.86,-0.105932,82,1.04221
1,PFE,2/25/2011,$18.88,$19.13,$18.67,$18.86,177077740,-0.105932,15.6210649,153153528,$18.95,$19.66,3.7467,75,1.06045
1,PFE,3/4/2011,$18.95,$19.90,$18.90,$19.66,257979737,3.7467,45.68727667,177077740,$19.64,$19.47,-0.86558,68,1.01729
1,PFE,3/11/2011,$19.64,$19.75,$19.30,$19.47,195391037,-0.86558,-24.26109148,257979737,$19.45,$20.18,3.75321,61,1.02722
1,PFE,3/18/2011,$19.45,$20.29,$19.15,$20.18,457640533,3.75321,134.2177717,195391037,$20.32,$20.35,0.147638,54,0.99108
1,PFE,3/25/2011,$20.32,$20.50,$19.74,$20.35,215914627,0.147638,-52.82003856,457640533,$20.42,$20.38,-0.195886,47,0.982801
1,PG,1/7/2011,$64.39,$65.08,$64.00,$64.50,52323352,0.170834,,,$64.40,$65.53,1.75466,12,0.744186
1,PG,1/14/2011,$64.40,$65.53,$63.40,$65.53,41932473,1.75466,-19.85897043,52323352,$65.65,$65.91,0.39604,5,0.732489
1,PG,1/21/2011,$65.65,$65.95,$65.05,$65.91,46213962,0.39604,10.21043762,41932473,$65.90,$64.20,-2.57967,96,0.804127
1,PG,1/28/2011,$65.90,$66.95,$63.14,$64.20,70042160,-2.57967,51.56060413,46213962,$64.35,$63.61,-1.14996,89,0.825545
1,PG,2/4/2011,$64.35,$64.36,$62.30,$63.61,57201539,-1.14996,-18.33270276,70042160,$63.95,$64.73,1.2197,82,0.833202
1,PG,2/11/2011,$63.95,$64.97,$63.80,$64.73,51999812,1.2197,-9.093683651,57201539,$64.80,$64.30,-0.771605,75,0.818786
1,PG,2/18/2011,$64.80,$64.80,$63.40,$64.30,36098973,-0.771605,-30.57864709,51999812,$63.73,$62.84,-1.39652,68,0.824261
1,PG,2/25/2011,$63.73,$64.40,$62.53,$62.84,47748854,-1.39652,32.2720566,36098973,$62.80,$62.03,-1.22611,61,0.843412
1,PG,3/4/2011,$62.80,$63.65,$61.56,$62.03,59801160,-1.22611,25.2410372,47748854,$62.00,$61.49,-0.822581,54,0.854425
1,PG,3/11/2011,$62.00,$62.30,$61.12,$61.49,52731797,-0.822581,-11.82144795,59801160,$61.04,$60.60,-0.720839,47,0.861929
1,PG,3/18/2011,$61.04,$61.39,$59.70,$60.60,66311521,-0.720839,25.75243927,52731797,$60.96,$60.88,-0.131234,40,0.874587
```

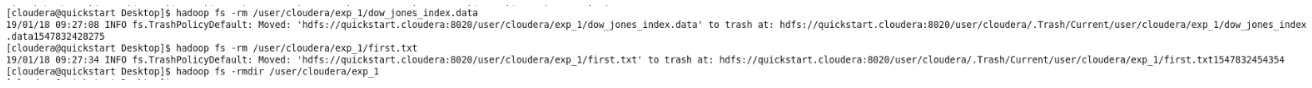
(iii) Deleting Files

Step 6:
Now, the files have to be deleted along with the directories. So, the commands 'hadoop fs-rm /user/cloudera/exp_1/dow_jones_index.data' and 'hadoop fs-rm /user/cloudera/exp_1/first.txt' is used to delete the specific files present in the exp_1 folder.This command will only delete non-empty directory and files.

Step 7:

Finally, the commands 'hadoop fs-rmdir /user/cloudera/exp_1'is used toremove the specific empty directories in the Hadoop File System.

The demonstration of the steps for deletion of files and directories as described above, is shown below;



```
[cloudera@quickstart Desktop]$ hadoop fs -rm /user/cloudera/exp_1/dow_jones_index.data
19/01/18 09:27:08 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstart.cloudera:8020/user/cloudera/exp_1/dow_jones_index.data' to trash at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/exp_1/dow_jones_index
.data1547832428275
[cloudera@quickstart Desktop]$ hadoop fs -rm /user/cloudera/exp_1/first.txt
19/01/18 09:27:34 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstart.cloudera:8020/user/cloudera/exp_1/first.txt' to trash at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/exp_1/first.txt1547832454354
[cloudera@quickstart Desktop]$ hadoop fs -rmdir /user/cloudera/exp_1
```

<u>Conclusion:</u>

In this experiment, we learnt about the commands used for navigating in Hadoop file system

<u>Experiment-2</u>

<u>Aim</u>: Implement the following steps in Hive:

(i) Creating databases and tables and loading data
(ii) Altering the table
(iii) Dropping the table and databases

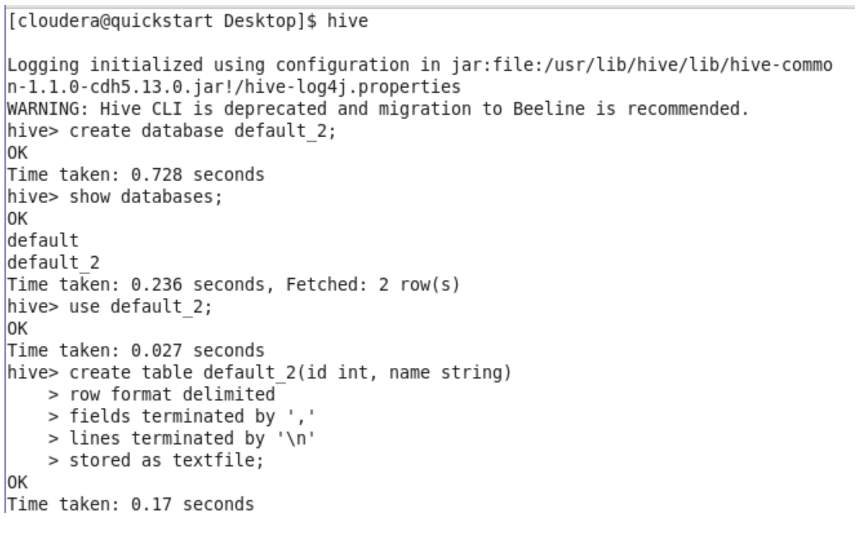
The experiment was finished with the following steps:

(i) Creating databases and tables and loading data

Step-1:
First, the command “hive” was used toinitialize hive in the CLI (Command Line Interface). Then, the command ‘Create database default_2;’ was used to create a database in hive. Next, the commands ‘Show databases;’ and ‘Use default_2;’ are used to display the already present databases in hive and use the desired database from those respectively.

Step 2:
Next, a table is declared with the command below;
Create table default_2(id int, name String)
Row format delimited
Fields terminated by ‘,’
Lines terminated by ‘\n’
Stored as textfile;

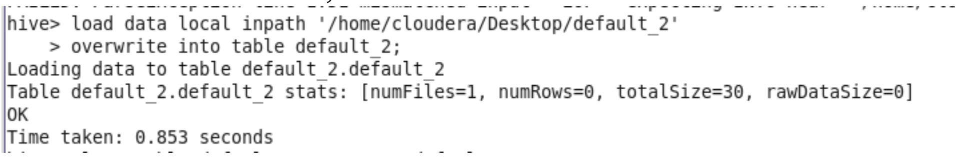
The demonstration of all the steps described above is shown below;



```
[cloudera@quickstart Desktop]$ hive

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-commo
n-1.1.0-cdh5.13.0.jar!/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> create database default_2;
OK
Time taken: 0.728 seconds
hive> show databases;
OK
default
default_2
Time taken: 0.236 seconds, Fetched: 2 row(s)
hive> use default_2;
OK
Time taken: 0.027 seconds
hive> create table default_2(id int, name string)
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.17 seconds
```

Step 3:
The data is loaded into the created table from a file on the desktop, which has fields terminated by ‘,’ and lines terminated by ‘\n’, by using the command given below;
Load data local inpath ‘/home/cloudera/Desktop/default_2’
Overwrite into table default_2;

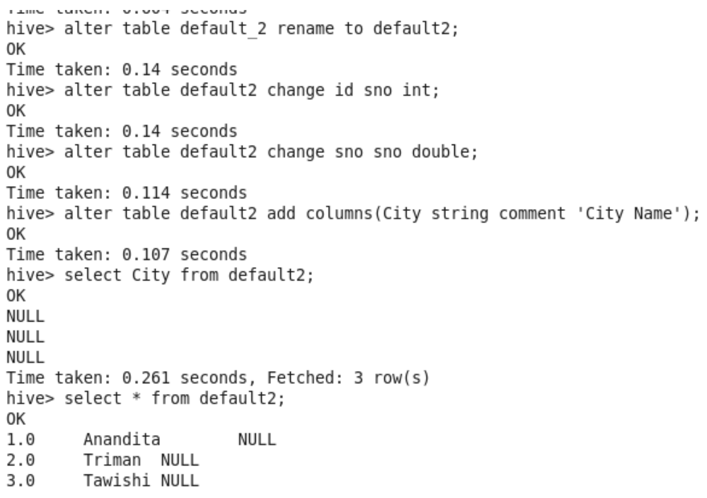
The demonstration is shown below;



```
hive> load data local inpath '/home/cloudera/Desktop/default_2'
    > overwrite into table default_2;
Loading data to table default_2.default_2
Table default_2.default_2 stats: [numFiles=1, numRows=0, totalSize=30, rawDataSize=0]
OK
Time taken: 0.853 seconds
```

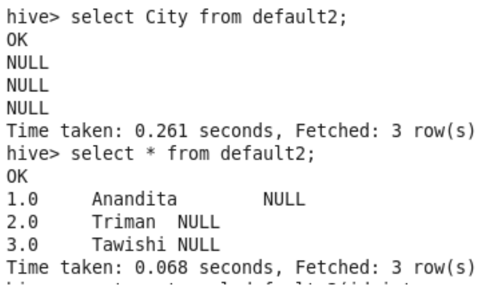
(ii) Altering the table

Step 4:
Next, the command ‘Alter table default_2 rename to default2’ was used to rename the already created table. The ‘Alter table default2 change id sno int’ command was used to rename an already existing column in the table default2. Then, the command ‘Alter table default2 change snosno double’ was used to change the data type of the column name ‘sno’. Next, the command ‘Alter table default2 add columns (City string comment ‘City Name’)’ was used to insert one more column named ‘City’ into the table default2.
The demonstration of all the steps described above is shown below;



```
hive> alter table default_2 rename to default2;
OK
Time taken: 0.14 seconds
hive> alter table default2 change id sno int;
OK
Time taken: 0.14 seconds
hive> alter table default2 change sno sno double;
OK
Time taken: 0.114 seconds
hive> alter table default2 add columns(City string comment 'City Name');
OK
Time taken: 0.107 seconds
hive> select City from default2;
OK
NULL
NULL
NULL
Time taken: 0.261 seconds, Fetched: 3 row(s)
hive> select * from default2;
OK
1.0     Anandita        NULL
2.0     Triman  NULL
3.0     Tawishi NULL
```

Further, we see the contents of the table by using the commands ‘Select City from default2;’ and ‘Select * from default2’. The newly created column is shown as NULL, as no values have been inserted in it yet, as shown below;



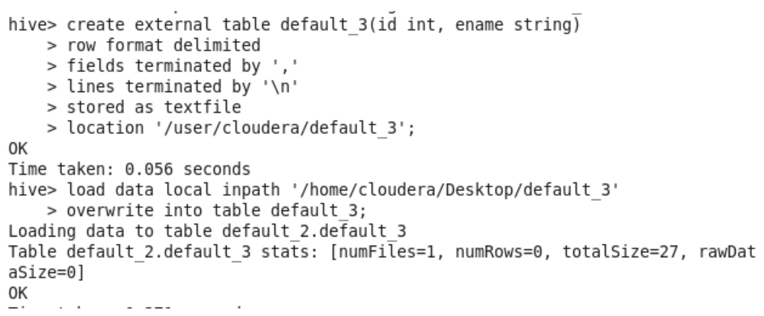
```
hive> select City from default2;
OK
NULL
NULL
NULL
Time taken: 0.261 seconds, Fetched: 3 row(s)
hive> select * from default2;
OK
1.0     Anandita        NULL
2.0     Triman  NULL
3.0     Tawishi NULL
Time taken: 0.068 seconds, Fetched: 3 row(s)
```

Step 5:
An external table is also created by specifying the location of the directory where we want to create the table. This table will be created in a directory separate from the data warehouse that HIVE provides. It is created using the following syntax;

Create external table default_3(id int, ename string)
Row format delimited
Fields terminated by ‘,’
Lines terminated by ‘\n’
Stored as textfile
Location ‘/user/cloudera/default_3’;

Step 6:
The data is loaded into the external table from a file on the desktop, which has fields terminated by ‘,’ and lines terminated by ‘\n’, by using the command given below;
Load data local inpath ‘/home/cloudera/Desktop/default_3’
Overwrite into table default_3;

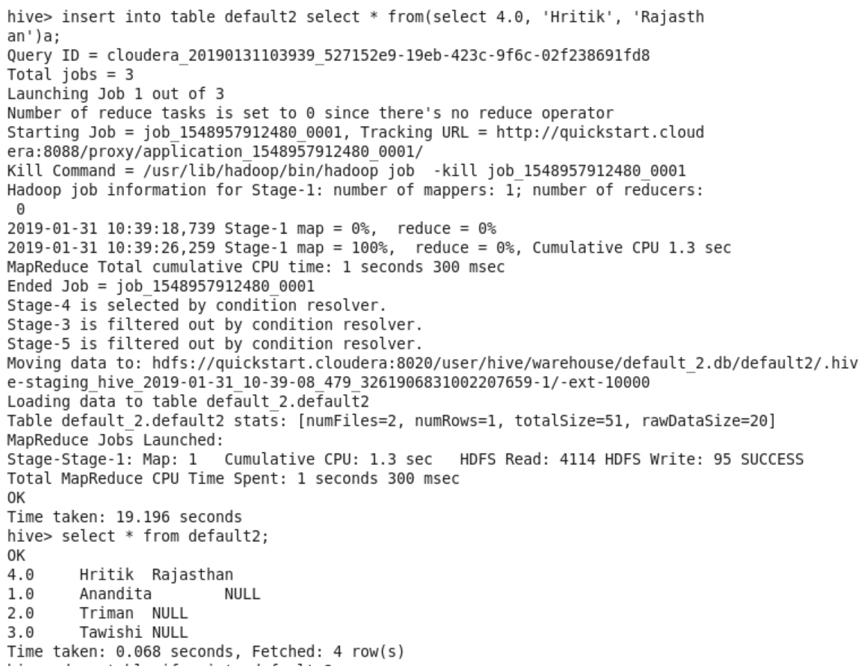
The demonstration of all the steps described above is shown below;



```
hive> create external table default_3(id int, ename string)
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile
    > location '/user/cloudera/default_3';
OK
Time taken: 0.056 seconds
hive> load data local inpath '/home/cloudera/Desktop/default_3'
    > overwrite into table default_3;
Loading data to table default_2.default_3
Table default_2.default_3 stats: [numFiles=1, numRows=0, totalSize=27, rawDat
aSize=0]
OK
```

Step 7:
Data can be inserted into the table as well, by using the command;
Insert into table default2 select * from (select 4.0, ‘Hritik’, ‘Rajasthan’)a;

The demonstration is shown below;



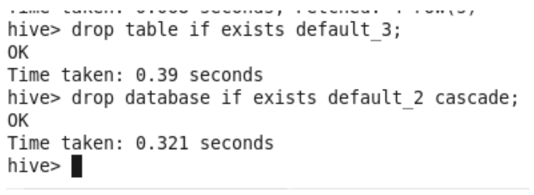
```
hive> insert into table default2 select * from(select 4.0, 'Hritik', 'Rajasth
an')a;
Query ID = cloudera_20190131103939_527152e9-19eb-423c-9f6c-02f238691fd8
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1548957912480_0001, Tracking URL = http://quickstart.cloud
era:8088/proxy/application_1548957912480_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1548957912480_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers:
 0
2019-01-31 10:39:18,739 Stage-1 map = 0%,  reduce = 0%
2019-01-31 10:39:26,259 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.3 sec
MapReduce Total cumulative CPU time: 1 seconds 300 msec
Ended Job = job_1548957912480_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/default_2.db/default2/.hiv
e-staging_hive_2019-01-31_10-39-08_479_3261906831002207659-1/-ext-10000
Loading data to table default_2.default2
Table default_2.default2 stats: [numFiles=2, numRows=1, totalSize=51, rawDataSize=20]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.3 sec   HDFS Read: 4114 HDFS Write: 95 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 300 msec
OK
Time taken: 19.196 seconds
hive> select * from default2;
OK
4.0     Hritik  Rajasthan
1.0     Anandita        NULL
2.0     Triman  NULL
3.0     Tawishi NULL
Time taken: 0.068 seconds, Fetched: 4 row(s)
```

As seen above, the table is viewed by the command;
Select * from default2;

(iii)Dropping the table and databases

Step 8:
The table and databases are removed from the HIVE storage using the following commands respectively;
Drop table if exists default_3;
Drop database if exists default_2 cascade;

The demonstration is shown below;



```
hive> drop table if exists default_3;
OK
Time taken: 0.39 seconds
hive> drop database if exists default_2 cascade;
OK
Time taken: 0.321 seconds
hive> █
```

Conclusion:

In this experiment, we learnt about the commands used for creating, altering and dropping tables and databases in HIVE using the HQL(Hive Query Language).

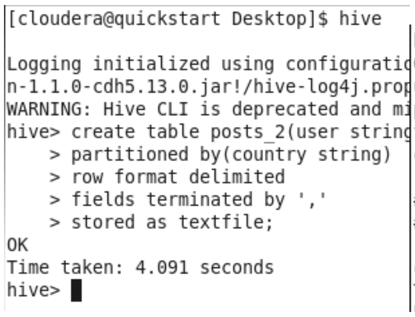
<u>Experiment-3</u>

<u>Aim</u>: Implement Hive Partitioning with data set.

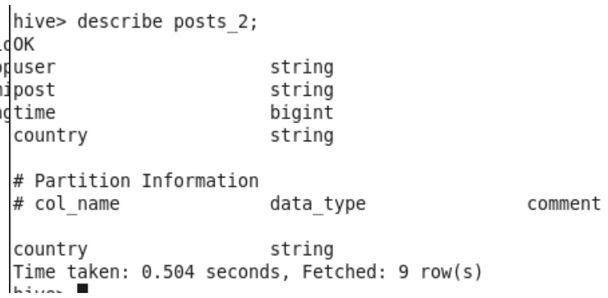
The experiment was finished with the following steps:

Partitioning in Hive:

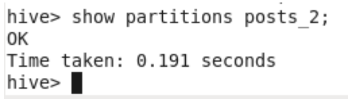
Partition is created in the table posts_2 with the column country. 2 partitions are created; one for country='US' and one for country='AUSTRALIA'. The following commands were used to do partitioning in Hive.



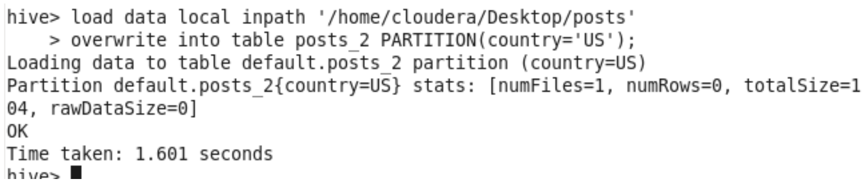
```
[cloudera@quickstart Desktop]$ hive

Logging initialized using configuratio
n-1.1.0-cdh5.13.0.jar!/hive-log4j.prop
WARNING: Hive CLI is deprecated and mi
hive> create table posts_2(user string
    > partitioned by(country string)
    > row format delimited
    > fields terminated by ','
    > stored as textfile;
OK
Time taken: 4.091 seconds
hive>
```



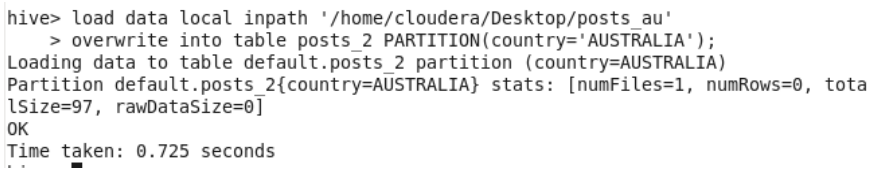
```
hive> describe posts_2;
OK
user                    string
post                    string
time                    bigint
country                 string

# Partition Information
# col_name              data_type               comment

country                 string
Time taken: 0.504 seconds, Fetched: 9 row(s)
```



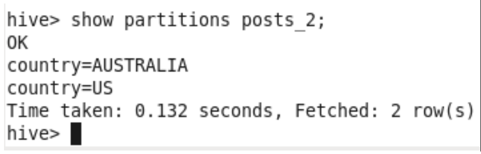
```
hive> show partitions posts_2;
OK
Time taken: 0.191 seconds
hive>
```



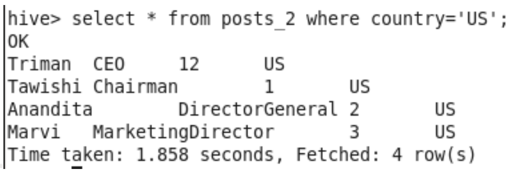
```
hive> load data local inpath '/home/cloudera/Desktop/posts'
    > overwrite into table posts_2 PARTITION(country='US');
Loading data to table default.posts_2 partition (country=US)
Partition default.posts_2{country=US} stats: [numFiles=1, numRows=0, totalSize=1
04, rawDataSize=0]
OK
Time taken: 1.601 seconds
hive>
```



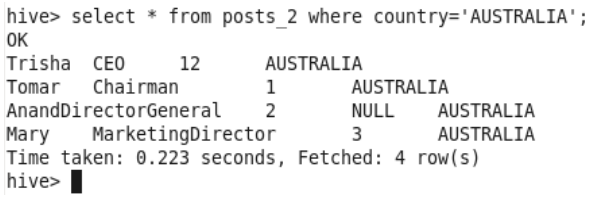
```
hive> load data local inpath '/home/cloudera/Desktop/posts_au'
    > overwrite into table posts_2 PARTITION(country='AUSTRALIA');
Loading data to table default.posts_2 partition (country=AUSTRALIA)
Partition default.posts_2{country=AUSTRALIA} stats: [numFiles=1, numRows=0, tota
lSize=97, rawDataSize=0]
OK
Time taken: 0.725 seconds
```



```
hive> show partitions posts_2;
OK
country=AUSTRALIA
country=US
Time taken: 0.132 seconds, Fetched: 2 row(s)
hive>
```



```
hive> select * from posts_2 where country='US';
OK
Triman  CEO     12      US
Tawishi Chairman        1       US
Anandita        DirectorGeneral 2       US
Marvi   MarketingDirector       3       US
Time taken: 1.858 seconds, Fetched: 4 row(s)
```



```
hive> select * from posts_2 where country='AUSTRALIA';
OK
Trisha  CEO     12      AUSTRALIA
Tomar   Chairman        1       AUSTRALIA
AnandDirectorGeneral    2       NULL    AUSTRALIA
Mary    MarketingDirector       3       AUSTRALIA
Time taken: 0.223 seconds, Fetched: 4 row(s)
hive>
```

Experiment-4

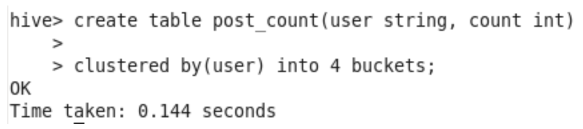
Aim: Implement Hive Bucketing with data set.

The experiment was finished with the following steps:

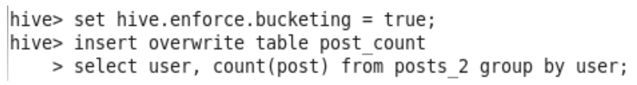
Bucketing in Hive:

In Bucketing Records with the same bucketed column will always be stored in the same bucket. We use CLUSTERED BY clause to divide the table into buckets.Physically, each bucket is just a file in the table directory, and Bucket numbering is 1-based. Here, the bucketed column is ‘user’ and 4 buckets have been made according to the number of posts they hold.

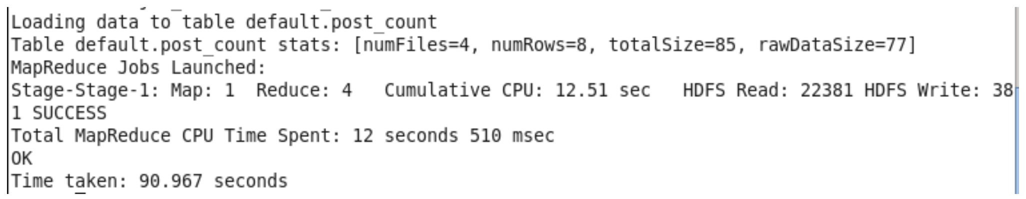
The commands used are demonstrated below.



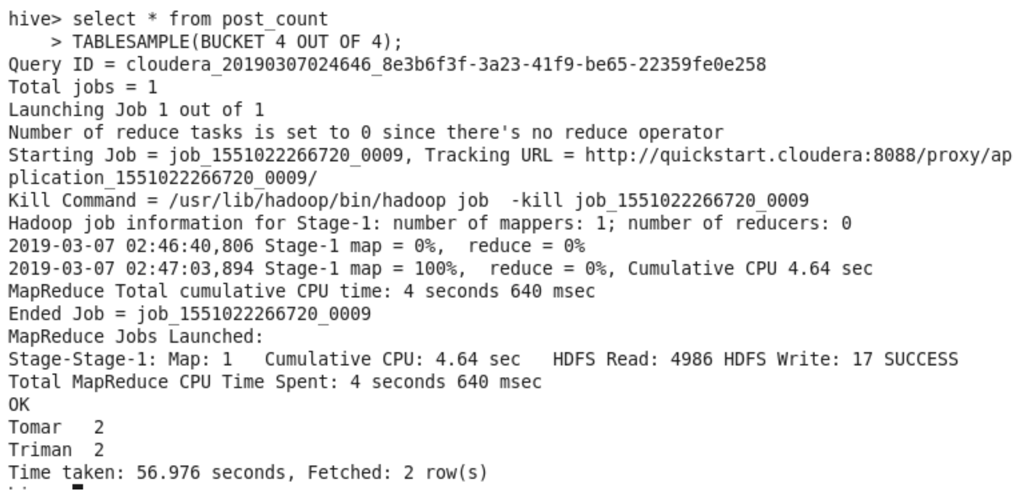
```
hive> create table post_count(user string, count int)
    >
    > clustered by(user) into 4 buckets;
OK
Time taken: 0.144 seconds
```



```
hive> set hive.enforce.bucketing = true;
hive> insert overwrite table post_count
    > select user, count(post) from posts_2 group by user;
```



```
Loading data to table default.post_count
Table default.post_count stats: [numFiles=4, numRows=8, totalSize=85, rawDataSize=77]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 4   Cumulative CPU: 12.51 sec   HDFS Read: 22381 HDFS Write: 38
1 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 510 msec
OK
Time taken: 90.967 seconds
```



```
hive> select * from post_count
    > TABLESAMPLE(BUCKET 4 OUT OF 4);
Query ID = cloudera_20190307024646_8e3b6f3f-3a23-41f9-be65-22359fe0e258
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1551022266720_0009, Tracking URL = http://quickstart.cloudera:8088/proxy/ap
plication_1551022266720_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1551022266720_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2019-03-07 02:46:40,806 Stage-1 map = 0%,  reduce = 0%
2019-03-07 02:47:03,894 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.64 sec
MapReduce Total cumulative CPU time: 4 seconds 640 msec
Ended Job = job_1551022266720_0009
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 4.64 sec   HDFS Read: 4986 HDFS Write: 17 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 640 msec
OK
Tomar   2
Triman  2
Time taken: 56.976 seconds, Fetched: 2 row(s)
```

Conclusion:

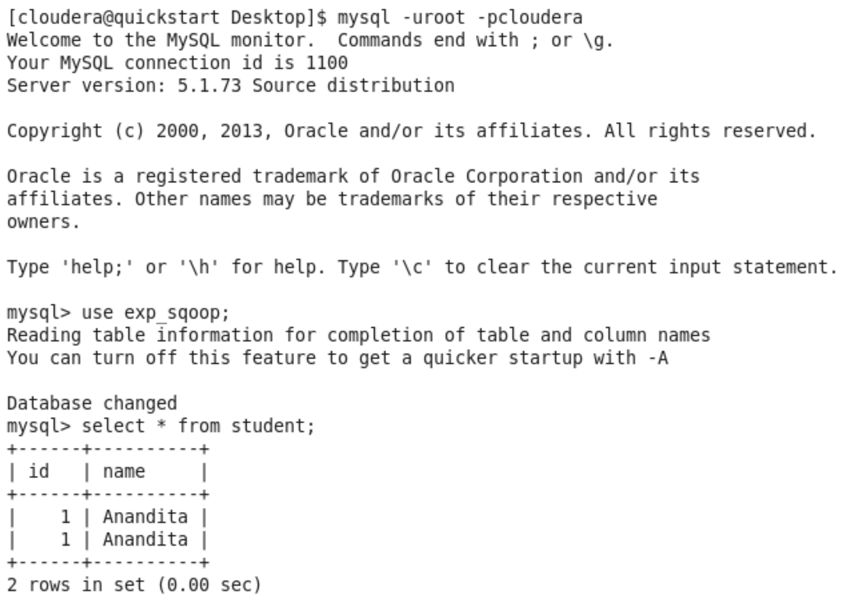
In this experiment, we learnt about the commands used for performing partitioning and bucketing in Hive.

Experiment-5

Aim: Implement Sqoop commands.

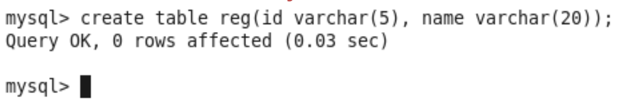
The following steps were executed in Sqoop:

1. First, MySQL was opened and the contents of a table from database are displayed. This is shown below.



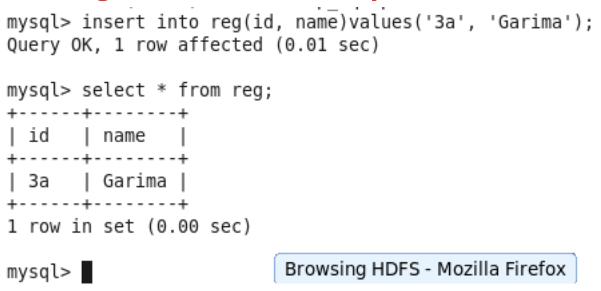
```
[cloudera@quickstart Desktop]$ mysql -uroot -pcloudera
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1100
Server version: 5.1.73 Source distribution

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> use exp_sqoop;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from student;
+------+----------+
| id   | name     |
+------+----------+
|    1 | Anandita |
|    1 | Anandita |
+------+----------+
2 rows in set (0.00 sec)
```

Creation of a table in MySQL:



```
mysql> create table reg(id varchar(5), name varchar(20));
Query OK, 0 rows affected (0.03 sec)

mysql> █
```

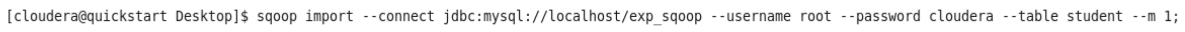
Inserting values in a table in MySQL:



```
mysql> insert into reg(id, name)values('3a', 'Garima');
Query OK, 1 row affected (0.01 sec)

mysql> select * from reg;
+------+--------+
| id   | name   |
+------+--------+
| 3a   | Garima |
+------+--------+
1 row in set (0.00 sec)

mysql> █
```

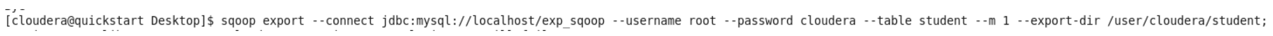
Browsing HDFS - Mozilla Firefox

2. Next, we exit from MySQL and open Sqoop and connect MySQL with it to import the table created or viewed into HDFS using the following commands.



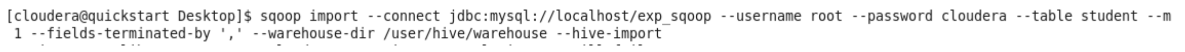
```
[cloudera@quickstart Desktop]$ sqoop import --connect jdbc:mysql://localhost/exp_sqoop --username root --password cloudera --table student --m 1;
```

'student' is the name of the newly created file:

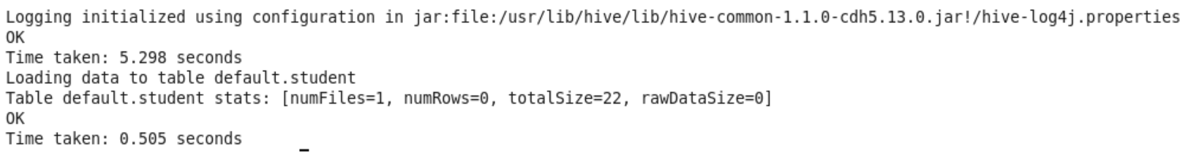


```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost/exp_sqoop --username root --password cloudera --table student --m 1 --export-dir /user/cloudera/student;
```

3. In order to import the table to hive, the following command was used and then the data was viewed from the imported directory in hive i.e. exp_sqoop.

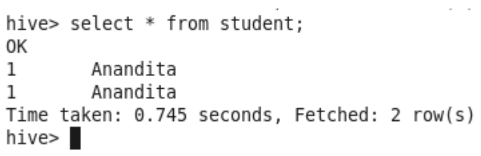


```
[cloudera@quickstart Desktop]$ sqoop import --connect jdbc:mysql://localhost/exp_sqoop --username root --password cloudera --table student --m
 1 --fields-terminated-by ',' --warehouse-dir /user/hive/warehouse --hive-import
```



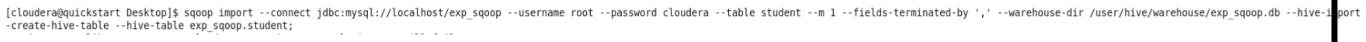
```
Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-1.1.0-cdh5.13.0.jar!/hive-log4j.properties
OK
Time taken: 5.298 seconds
Loading data to table default.student
Table default.student stats: [numFiles=1, numRows=0, totalSize=22, rawDataSize=0]
OK
Time taken: 0.505 seconds
```

After opening Hive:

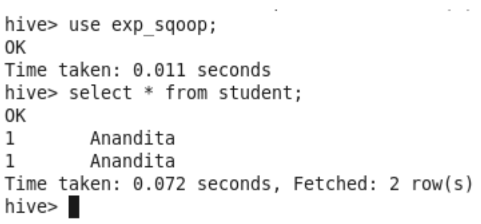


```
hive> select * from student;
OK
1       Anandita
1       Anandita
Time taken: 0.745 seconds, Fetched: 2 row(s)
hive>
```

'student' is the name of the new table created:



```
[cloudera@quickstart Desktop]$ sqoop import --connect jdbc:mysql://localhost/exp_sqoop --username root --password cloudera --table student --m 1 --fields-terminated-by ',' --warehouse-dir /user/hive/warehouse/exp_sqoop.db --hive-i port
-create-hive-table --hive-table exp_sqoop.student;
```



```
hive> use exp_sqoop;
OK
Time taken: 0.011 seconds
hive> select * from student;
OK
1       Anandita
1       Anandita
Time taken: 0.072 seconds, Fetched: 2 row(s)
hive>
```

Conclusion:

In this experiment, we learnt about the commands used for importing and exporting databases using Sqoop in Hadoop.

Experiment-6

Aim: Weather Report POC-Map Reduce Program to analyse time-temperature statistics and generate report with max/min temperature.

Problem Statement:
1. The system receives temperatures of various cities(Austin, Boston,etc) of USA captured at regular intervals of time on each day in an input file.
2. System will process the input data file and generates a report with Maximum and Minimum temperatures of each day along with time.
3. Generates a separate output report for each city.
Ex: Austin-r-00000
Boston-r-00000
Newjersy-r-00000
Baltimore-r-00000
California-r-00000
Newyork-r-00000
Expected output:- In each output file record should be like this:
25-Jan-2014 Time: 12:34:542 MinTemp: -22.3 Time: 05:12:345 MaxTemp:
35.7
First download input file which contains temperature statistics with time for multiple cities.
Schema of record set : CA_25-Jan-2014 00:12:345 15.7
01:19:345 23.1 02:34:542 12.3 ......
CA is city code, here it stands for California followed by date. After that each
pair of values represent time and temperature.
Mapper class and map method:-
The very first thing which is required for any map reduce problem is to understand what will be the type of keyIn, ValueIn, KeyOut,ValueOut for the given Mapper class and followed by type of map method parameters.
• public class WhetherForcastMapper extends Mapper <**Object, Text,
Text, Text**>
• Object (keyIn) - Offset for each line, line number 1, 2...
• Text **(**ValueIn**) -** Whole string for each line (CA_25-Jan-2014
00:12:345 ......)
• Text **(**KeyOut**) -** City information with date information as string
• Text **(**ValueOut**) -** Temperature and time information which need to be
passed to reducer as string.
• public void map(**Object keyOffset, Text dayReport, Context con**) { }
• *KeyOffset* is like line number for each line in input file.
• *dayreport* is input to map method - whole string present in one line of
input file.
• *con* is context where we write mapper output and it is used by reducer.
Reducer class and reducer method:-
Similarly,we have to decide what will be the type of keyIn, ValueIn, KeyOut,ValueOut for the given Reducer class and followed by type of reducer
method parameters.
• public class WhetherForcastReducer extends Reducer<**Text, Text,
Text, Text**>
• Text(keyIn) - it is same as keyOut of Mapper.
• Text(ValueIn)- it is same as valueOut of Mapper.
• Text(KeyOut)- date as string
• text(ValueOut) - reducer writes max and min temperature with time as
string
• public void reduce(**Text key, Iterable<Text> values, Context**

context)
• Text key is value of mapper output. i.e:- City & date information
• Iterable<Text> values - values stores multiple temperature values for a given city and date.
• context object is where reducer write it's processed outcome and finally written in file.

MultipleOutputs :- In general, reducer generates output file(i.e: part_r_0000), however in this use case we want to generate multiple output files. In order to deal with such scenario we need to use MultipleOutputs of "org.apache.hadoop.mapreduce.lib.output.MultipleOutputs" which provides a way to write multiple file depending on reducer outcome.

For each reducer task multipleoutput object is created and key/result is written to appropriate file.
Lets create a Map/Reduce project in eclipse and create a class file name it as CalculateMaxAndMinTemeratureWithTime.
For simplicity,here we have written mapper and reducer class as inner static class. Copy following code lines and paste in newly created class file.

```
/**
* Question:- To find Max and Min temperature from
record set stored in
* text file. Schema of record set :- tab
separated (\t) CA_25-Jan-2014
* 00:12:345 15.7 01:19:345 23.1 02:34:542
12.3 03:12:187 16 04:00:093
* -14 05:12:345 35.7 06:19:345 23.1 07:34:542
12.3 08:12:187 16
* 09:00:093 -7 10:12:345 15.7 11:19:345 23.1
12:34:542 -22.3 13:12:187
* 16 14:00:093 -7 15:12:345 15.7 16:19:345
23.1 19:34:542 12.3
* 20:12:187 16 22:00:093 -7
* Expected output:- Creates files for each city and store maximum & minimum
* temperature for each day along with time.
*/
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import
org.apache.hadoop.mapreduce.lib.output.MultipleOutput
s;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat
;
import
org.apache.hadoop.mapreduce.lib.output.FileOutputForm
at;
import
org.apache.hadoop.mapreduce.lib.output.TextOutputForm
at;
/**
* @author devinline
```

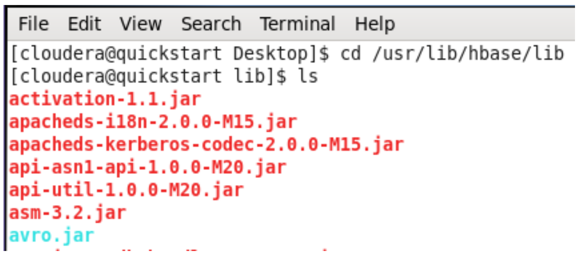
```
*/
public class CalculateMaxAndMinTemeratureWithTime {
public static String calOutputName =
"California";
public static String nyOutputName = "Newyork";
public static String njOutputName = "Newjersy";
public static String ausOutputName = "Austin";
public static String bosOutputName = "Boston";
public static String balOutputName =
"Baltimore";
public static class WhetherForcastMapper extends
Mapper<Object, Text, Text, Text> {
public void map(Object keyOffset, Text
dayReport, Context con)
throws IOException, InterruptedException {
StringTokenizer strTokens = new
StringTokenizer(
dayReport.toString(), "\t");
int counter = 0;
Float currnetTemp = null;
Float minTemp = Float.MAX_VALUE;
Float maxTemp = Float.MIN_VALUE;
String date = null;
String currentTime = null;
String minTempANDTime = null;
String maxTempANDTime = null;
while (strTokens.hasMoreElements()) {
if (counter == 0) {
date = strTokens.nextToken();
} else {
if (counter % 2 == 1) {
currentTime = strTokens.nextToken();
} else {
currnetTemp =
Float.parseFloat(strTokens.nextToken());
if (minTemp > currnetTemp) {
minTemp = currnetTemp;
minTempANDTime = minTemp + "AND" +
currentTime;
}
if (maxTemp < currnetTemp) {
maxTemp = currnetTemp;
maxTempANDTime = maxTemp + "AND" +
currentTime;
}
}
}
counter++;
}
// Write to context - MinTemp, MaxTemp and
corresponding time
Text temp = new Text();
temp.set(maxTempANDTime);
Text dateText = new Text();
dateText.set(date);
```

```
try {
con.write(dateText, temp);
} catch (Exception e) {
e.printStackTrace();
}
temp.set(minTempANDTime);
dateText.set(date);
con.write(dateText, temp);
}
}
public static class WhetherForcastReducer extends
Reducer<Text, Text, Text, Text> {
MultipleOutputs<Text, Text> mos;
public void setup(Context context) {
mos = new MultipleOutputs<Text,
Text>(context);
}
public void reduce(Text key, Iterable<Text>
values, Context context)
throws IOException, InterruptedException {
int counter = 0;
String reducerInputStr[] = null;
String f1Time = "";
String f2Time = "";
String f1 = "", f2 = "";
Text result = new Text();
for (Text value : values) {
if (counter == 0) {
reducerInputStr =
value.toString().split("AND");
f1 = reducerInputStr[0];
f1Time = reducerInputStr[1];
}
else {
reducerInputStr =
value.toString().split("AND");
f2 = reducerInputStr[0];
f2Time = reducerInputStr[1];
}
counter = counter + 1;
}
if (Float.parseFloat(f1) >
Float.parseFloat(f2)) {
result = new Text("Time: " + f2Time + "
MinTemp: " + f2 + "\t"
+ "Time: " + f1Time + " MaxTemp: " + f1);
} else {
result = new Text("Time: " + f1Time + "
MinTemp: " + f1 + "\t"
+ "Time: " + f2Time + " MaxTemp: " + f2);
}
String fileName = "";
if (key.toString().substring(0, 2).equals("CA"))
{
fileName =
CalculateMaxAndMinTemeratureTime.calOutputName;
} else if (key.toString().substring(0,
2).equals("NY")) {
fileName =
CalculateMaxAndMinTemeratureTime.nyOutputName;
} else if (key.toString().substring(0,
2).equals("NJ")) {
fileName =
CalculateMaxAndMinTemeratureTime.njOutputName;
} else if (key.toString().substring(0,
3).equals("AUS")) {
fileName =
CalculateMaxAndMinTemeratureTime.ausOutputName;
} else if (key.toString().substring(0,
3).equals("BOS")) {
fileName =
CalculateMaxAndMinTemeratureTime.bosOutputName;
} else if (key.toString().substring(0,
3).equals("BAL")) {
fileName =
CalculateMaxAndMinTemeratureTime.balOutputName;
}
String strArr[] = key.toString().split("_");
key.set(strArr[1]); //Key is date value
mos.write(fileName, key, result);
}
@Override
public void cleanup(Context context) throws
IOException,
InterruptedException {
mos.close();
}
}
public static void main(String[] args) throws
IOException,
ClassNotFoundException,
InterruptedException {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "Wheather
Statistics of USA");
job.setJarByClass(CalculateMaxAndMinTemeratureW
ithTime.class);
job.setMapperClass(WhetherForcastMapper.class);
job.setReducerClass(WhetherForcastReducer.class)
;
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
MultipleOutputs.addNamedOutput(job,
calOutputName,
TextOutputFormat.class, Text.class,
Text.class);
MultipleOutputs.addNamedOutput(job,
nyOutputName,
TextOutputFormat.class, Text.class,
Text.class);
MultipleOutputs.addNamedOutput(job,
njOutputName,
TextOutputFormat.class, Text.class,
Text.class);
MultipleOutputs.addNamedOutput(job,
bosOutputName,
TextOutputFormat.class, Text.class,
Text.class);
MultipleOutputs.addNamedOutput(job,
ausOutputName,
TextOutputFormat.class, Text.class,
Text.class);
MultipleOutputs.addNamedOutput(job,
balOutputName,
TextOutputFormat.class, Text.class,
Text.class);
// FileInputFormat.addInputPath(job, new
Path(args[0]));
// FileOutputFormat.setOutputPath(job, new
Path(args[1]));
Path pathInput = new Path(
"hdfs://192.168.213.133:54310/weatherInputData/
input_temp.txt");
Path pathOutputDir = new Path(
"hdfs://192.168.213.133:54310/user/hduser1/
testfs/output_mapred3");
FileInputFormat.addInputPath(job, pathInput);
FileOutputFormat.setOutputPath(job,
pathOutputDir);
try {
System.exit(job.waitForCompletion(true) ? 0 :
1);
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

Experiment-7

Aim: Implement HBase commands with data set.

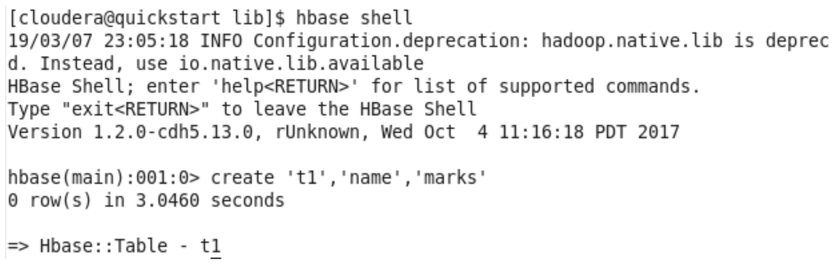
First, in order to open HBase, we go into the library of HBase in user folder in HDFS, by typing the command cd /usr/lib/hbase/lib. Then we type ls, to see if the hbase shell is available.



```
[cloudera@quickstart Desktop]$ cd /usr/lib/hbase/lib
[cloudera@quickstart lib]$ ls
activation-1.1.jar
apacheds-i18n-2.0.0-M15.jar
apacheds-kerberos-codec-2.0.0-M15.jar
api-asn1-api-1.0.0-M20.jar
api-util-1.0.0-M20.jar
asm-3.2.jar
avro.jar
```

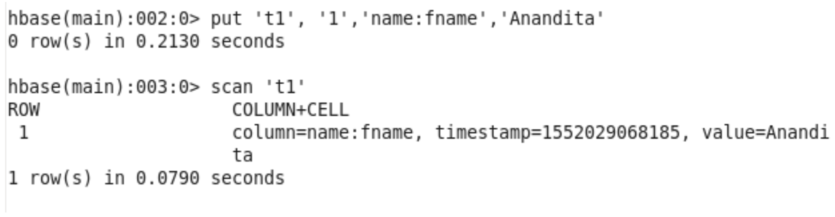
Next, the following steps are used:

1. Type ‘hbase shell’ to open hbase.
2. Create table t1 by typing the command create ‘t1’, ‘name’, ‘marks’. Here, name and marks are the column families.



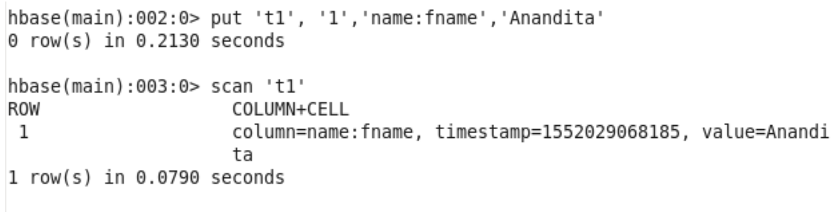
```
[cloudera@quickstart lib]$ hbase shell
19/03/07 23:05:18 INFO Configuration.deprecation: hadoop.native.lib is deprec
d. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> create 't1','name','marks'
0 row(s) in 3.0460 seconds

=> Hbase::Table - t1
```

3. Next, type the command, put ‘t1’, ‘1’, ‘name: fname’, ‘Anandita’. Here fname is the column name and ‘Anandita’ is the value of the column.



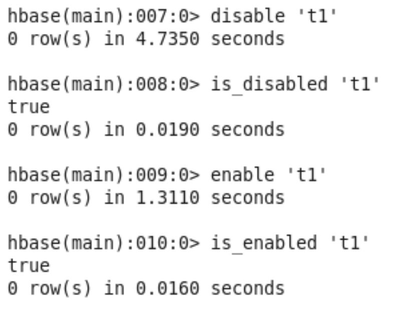
```
hbase(main):002:0> put 't1', '1','name:fname','Anandita'
0 row(s) in 0.2130 seconds

hbase(main):003:0> scan 't1'
ROW                    COLUMN+CELL
 1                     column=name:fname, timestamp=1552029068185, value=Anandi
                       ta
1 row(s) in 0.0790 seconds
```

4. To view the table timestamp, column i.e. the column and the cell, we use the command scan ‘t1’.



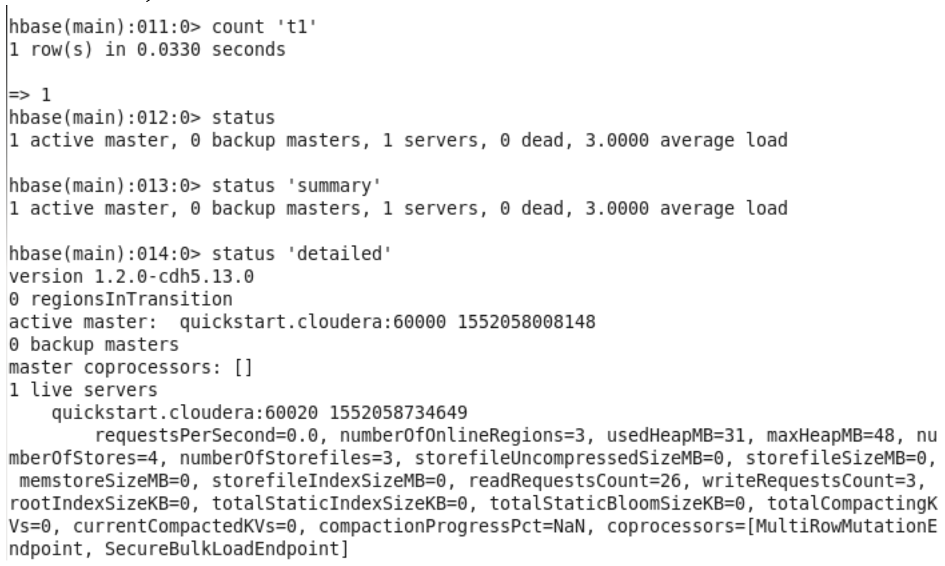
```
hbase(main):002:0> put 't1', '1','name:fname','Anandita'
0 row(s) in 0.2130 seconds

hbase(main):003:0> scan 't1'
ROW                    COLUMN+CELL
 1                     column=name:fname, timestamp=1552029068185, value=Anandi
                       ta
1 row(s) in 0.0790 seconds
```

5. In order to disable the table or check if the table is disabled, we use the following commands. Similar commands are used to check when the table is enabled and to enable it.



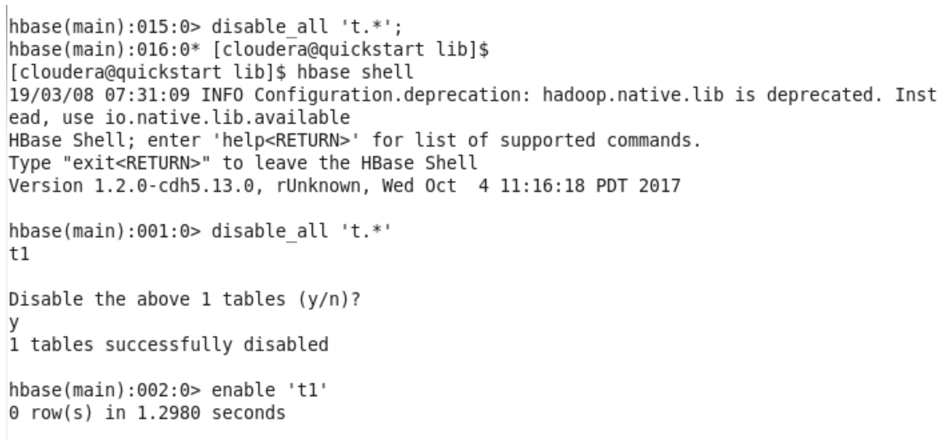
```
hbase(main):007:0> disable 't1'
0 row(s) in 4.7350 seconds

hbase(main):008:0> is_disabled 't1'
true
0 row(s) in 0.0190 seconds

hbase(main):009:0> enable 't1'
0 row(s) in 1.3110 seconds

hbase(main):010:0> is_enabled 't1'
true
0 row(s) in 0.0160 seconds
```

6. To count the no. of rows and check the status of the hbase nodes we use the following commands;



```
hbase(main):011:0> count 't1'
1 row(s) in 0.0330 seconds

=> 1
hbase(main):012:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 3.0000 average load

hbase(main):013:0> status 'summary'
1 active master, 0 backup masters, 1 servers, 0 dead, 3.0000 average load

hbase(main):014:0> status 'detailed'
version 1.2.0-cdh5.13.0
0 regionsInTransition
active master:  quickstart.cloudera:60000 1552058008148
0 backup masters
master coprocessors: []
1 live servers
    quickstart.cloudera:60020 1552058734649
        requestsPerSecond=0.0, numberOfOnlineRegions=3, usedHeapMB=31, maxHeapMB=48, nu
mberOfStores=4, numberOfStorefiles=3, storefileUncompressedSizeMB=0, storefileSizeMB=0,
 memstoreSizeMB=0, storefileIndexSizeMB=0, readRequestsCount=26, writeRequestsCount=3,
rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0, totalCompactingK
Vs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[MultiRowMutationE
ndpoint, SecureBulkLoadEndpoint]
```

7. In order to disable the tables starting with a specific letter and droop a table the following commands are used.



```
hbase(main):015:0> disable_all 't.*';
hbase(main):016:0* [cloudera@quickstart lib]$
[cloudera@quickstart lib]$ hbase shell
19/03/08 07:31:09 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Inst
ead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> disable_all 't.*'
t1

Disable the above 1 tables (y/n)?
y
1 tables successfully disabled

hbase(main):002:0> enable 't1'
0 row(s) in 1.2980 seconds
```

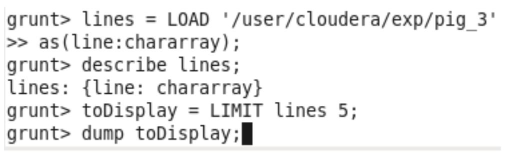
Experiment-8

Aim: Install & Run Pig then write Pig Latin scripts to sort, group, join & filter your data.

The experiment was finished with the following steps:

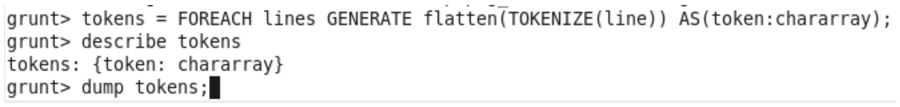
(i) Sort in Pig Latin:

Sorting is done by the command ‘order bag_name by tuple_no. desc’, as shown below, along with other important Pig Latin commands.

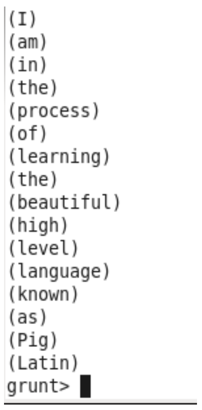


```
grunt> lines = LOAD '/user/cloudera/exp/pig_3'
>> as(line:chararray);
grunt> describe lines;
lines: {line: chararray}
grunt> toDisplay = LIMIT lines 5;
grunt> dump toDisplay;
```

TOKENIZE in capitals.

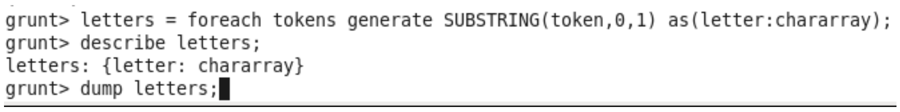


```
grunt> tokens = FOREACH lines GENERATE flatten(TOKENIZE(line)) AS(token:chararray);
grunt> describe tokens
tokens: {token: chararray}
grunt> dump tokens;
```

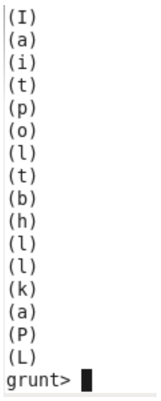


```
(I)
(am)
(in)
(the)
(process)
(of)
(learning)
(the)
(beautiful)
(high)
(level)
(language)
(known)
(as)
(Pig)
(Latin)
grunt>
```

SUBSTRING() in capitals.

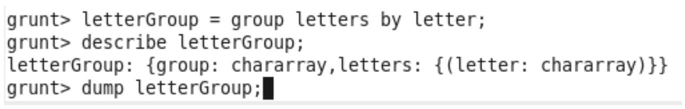


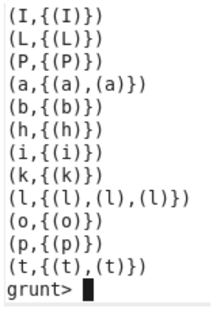
```
grunt> letters = foreach tokens generate SUBSTRING(token,0,1) as(letter:chararray);
grunt> describe letters;
letters: {letter: chararray}
grunt> dump letters;
```



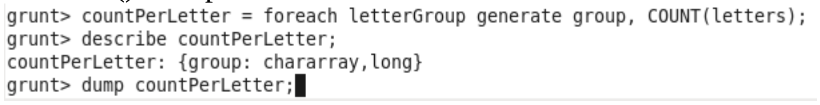
```
(I)
(a)
(i)
(t)
(p)
(o)
(l)
(t)
(b)
(h)
(l)
(l)
(k)
(a)
(P)
(L)
grunt>
```

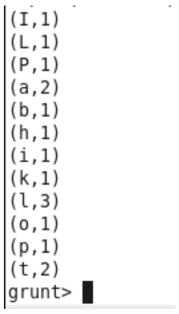
(ii) Group in Pig Latin:

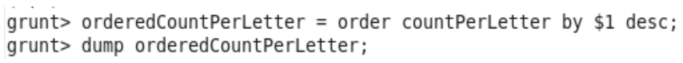

```
grunt> letterGroup = group letters by letter;
grunt> describe letterGroup;
letterGroup: {group: chararray,letters: {(letter: chararray)}}
grunt> dump letterGroup;
```

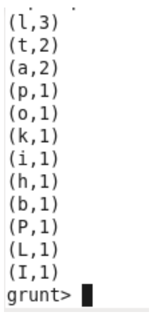

```
(I,{(I)})
(L,{(L)})
(P,{(P)})
(a,{(a),(a)})
(b,{(b)})
(h,{(h)})
(i,{(i)})
(k,{(k)})
(l,{(l),(l),(l)})
(o,{(o)})
(p,{(p)})
(t,{(t),(t)})
grunt>
```

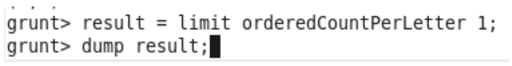
COUNT() in capitals.

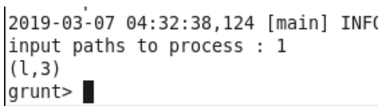

```
grunt> countPerLetter = foreach letterGroup generate group, COUNT(letters);
grunt> describe countPerLetter;
countPerLetter: {group: chararray,long}
grunt> dump countPerLetter;
```


```
(I,1)
(L,1)
(P,1)
(a,2)
(b,1)
(h,1)
(i,1)
(k,1)
(l,3)
(o,1)
(p,1)
(t,2)
grunt>
```

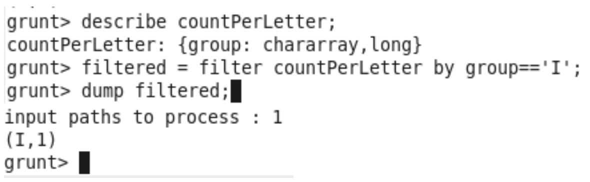

```
grunt> orderedCountPerLetter = order countPerLetter by $1 desc;
grunt> dump orderedCountPerLetter;
```


```
(l,3)
(t,2)
(a,2)
(p,1)
(o,1)
(k,1)
(i,1)
(h,1)
(b,1)
(P,1)
(L,1)
(I,1)
grunt>
```

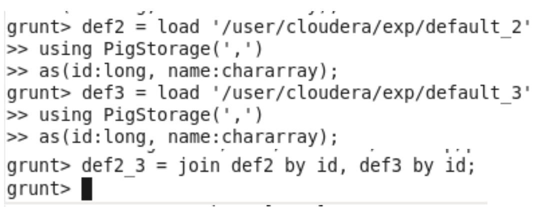

```
grunt> result = limit orderedCountPerLetter 1;
grunt> dump result;
```


```
2019-03-07 04:32:38,124 [main] INF(
input paths to process : 1
(l,3)
grunt>
```

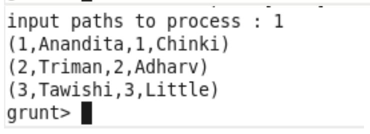
(iii) Filter in Pig Latin:


```
grunt> describe countPerLetter;
countPerLetter: {group: chararray,long}
grunt> filtered = filter countPerLetter by group=='I';
grunt> dump filtered;
input paths to process : 1
(I,1)
grunt>
```

(iv) Join in Pig Latin:



```
grunt> def2 = load '/user/cloudera/exp/default_2'
>> using PigStorage(',')
>> as(id:long, name:chararray);
grunt> def3 = load '/user/cloudera/exp/default_3'
>> using PigStorage(',')
>> as(id:long, name:chararray);
grunt> def2_3 = join def2 by id, def3 by id;
grunt> 
```



```
input paths to process : 1
(1,Anandita,1,Chinki)
(2,Triman,2,Adharv)
(3,Tawishi,3,Little)
grunt> 
```

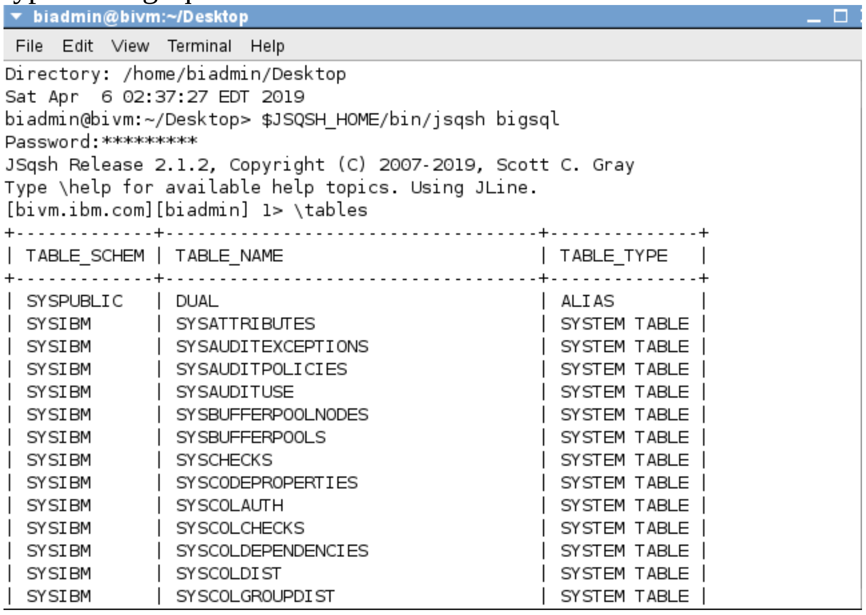
Conclusion:

In this experiment, we learnt about the commands used for performing sort, group, join & filter data using Pig.

Experiment-9

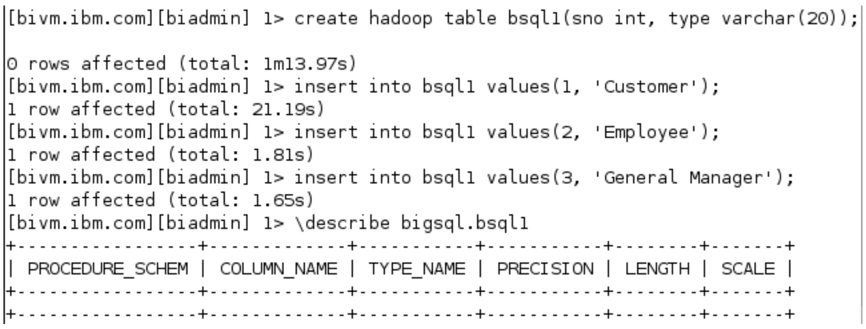
Aim: Using Fluid Query with Big SQL.

First, the jsqsh command line is enabled using the following commands;
'\table' command is used for displaying all the already present schemas and table names and types in BigSql.



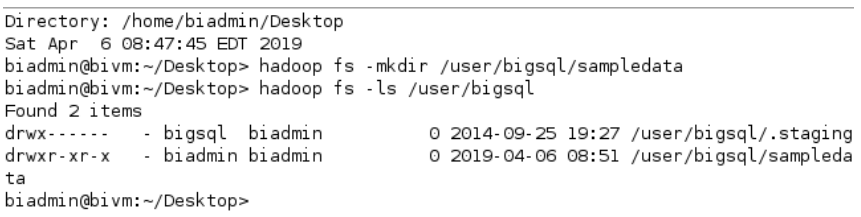
```
Directory: /home/biadmin/Desktop
Sat Apr  6 02:37:27 EDT 2019
biadmin@bivm:~/Desktop> $JSQSH_HOME/bin/jsqsh bigsql
Password:*********
JSqsh Release 2.1.2, Copyright (C) 2007-2019, Scott C. Gray
Type \help for available help topics. Using JLine.
[bivm.ibm.com][biadmin] 1> \tables
+------------+--------------------------------+-------------+
| TABLE_SCHEM | TABLE_NAME                     | TABLE_TYPE   |
+------------+--------------------------------+-------------+
| SYSPUBLIC   | DUAL                           | ALIAS        |
| SYSIBM      | SYSATTRIBUTES                  | SYSTEM TABLE |
| SYSIBM      | SYSAUDITEXCEPTIONS             | SYSTEM TABLE |
| SYSIBM      | SYSAUDITPOLICIES               | SYSTEM TABLE |
| SYSIBM      | SYSAUDITUSE                    | SYSTEM TABLE |
| SYSIBM      | SYSBUFFERPOOLNODES             | SYSTEM TABLE |
| SYSIBM      | SYSBUFFERPOOLS                 | SYSTEM TABLE |
| SYSIBM      | SYSCHECKS                      | SYSTEM TABLE |
| SYSIBM      | SYSCODEPROPERTIES              | SYSTEM TABLE |
| SYSIBM      | SYSCOLAUTH                     | SYSTEM TABLE |
| SYSIBM      | SYSCOLCHECKS                   | SYSTEM TABLE |
| SYSIBM      | SYSCOLDEPENDENCIES             | SYSTEM TABLE |
| SYSIBM      | SYSCOLDIST                     | SYSTEM TABLE |
| SYSIBM      | SYSCOLGROUPDIST                | SYSTEM TABLE |
```

Then, we create a table with column names and types and we can describe the table using the below given command;

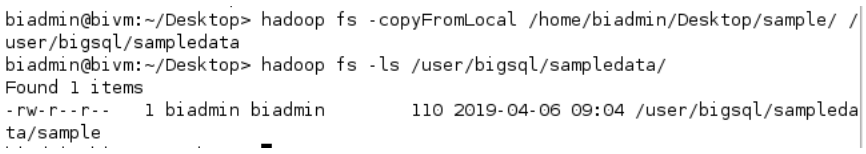


```
[bivm.ibm.com][biadmin] 1> create hadoop table bsql1(sno int, type varchar(20));

0 rows affected (total: 1m13.97s)
[bivm.ibm.com][biadmin] 1> insert into bsql1 values(1, 'Customer');
1 row affected (total: 21.19s)
[bivm.ibm.com][biadmin] 1> insert into bsql1 values(2, 'Employee');
1 row affected (total: 1.81s)
[bivm.ibm.com][biadmin] 1> insert into bsql1 values(3, 'General Manager');
1 row affected (total: 1.65s)
[bivm.ibm.com][biadmin] 1> \describe bigsql.bsql1
+----------------+------------+----------+----------+--------+-------+
| PROCEDURE_SCHEM | COLUMN_NAME | TYPE_NAME | PRECISION | LENGTH | SCALE |
+----------------+------------+----------+----------+--------+-------+
+----------------+------------+----------+----------+--------+-------+
```

Furthermore, we create a folder in bigsql folder called sampledata and we transfer dataset in the file called sample on the Desktop is moved into this bigsql folder.

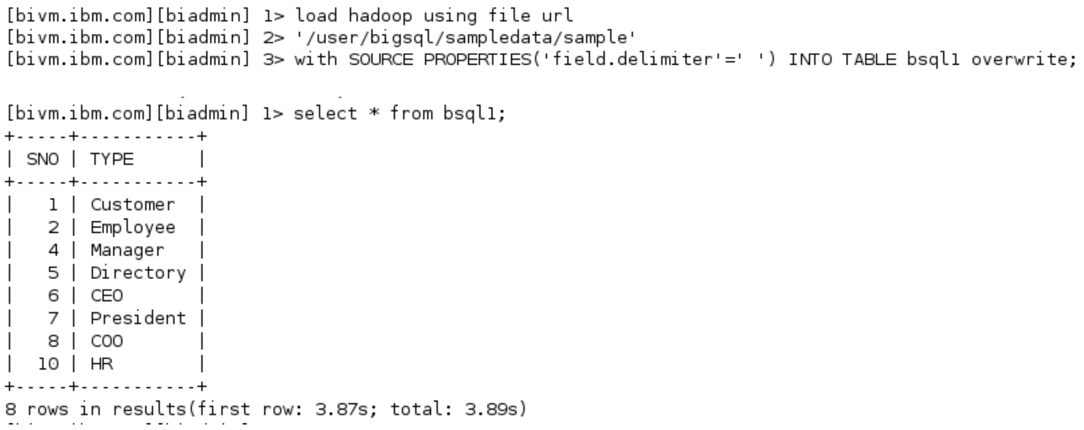


```
Directory: /home/biadmin/Desktop
Sat Apr  6 08:47:45 EDT 2019
biadmin@bivm:~/Desktop> hadoop fs -mkdir /user/bigsql/sampledata
biadmin@bivm:~/Desktop> hadoop fs -ls /user/bigsql
Found 2 items
drwx------   - bigsql  biadmin          0 2014-09-25 19:27 /user/bigsql/.staging
drwxr-xr-x   - biadmin biadmin          0 2019-04-06 08:51 /user/bigsql/sampleda
ta
biadmin@bivm:~/Desktop>
```



```
biadmin@bivm:~/Desktop> hadoop fs -copyFromLocal /home/biadmin/Desktop/sample/ /
user/bigsql/sampledata
biadmin@bivm:~/Desktop> hadoop fs -ls /user/bigsql/sampledata/
Found 1 items
-rw-r--r--   1 biadmin biadmin        110 2019-04-06 09:04 /user/bigsql/sampleda
ta/sample
```

Then, this sample data in the bigsql folder is overwritten in the table created in BigSql earlier. The procedure is shown below;



```
[bivm.ibm.com][biadmin] 1> load hadoop using file url
[bivm.ibm.com][biadmin] 2> '/user/bigsql/sampledata/sample'
[bivm.ibm.com][biadmin] 3> with SOURCE PROPERTIES('field.delimiter'=' ') INTO TABLE bsql1 overwrite;

[bivm.ibm.com][biadmin] 1> select * from bsql1;
+-----+----------+
| SNO | TYPE     |
+-----+----------+
|   1 | Customer  |
|   2 | Employee  |
|   4 | Manager   |
|   5 | Directory |
|   6 | CEO       |
|   7 | President |
|   8 | COO       |
|  10 | HR        |
+-----+----------+
8 rows in results(first row: 3.87s; total: 3.89s)
```

Experiment-10

Aim: Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm with data set

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper
2. Reducer
3. Driver

Source code:

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;
public class WordCount
{
public static class Map extends Mapper<LongWritable,Text,Text,IntWritable> {
public void map(LongWritable key, Text value,Context context) throws
IOException,InterruptedException{
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()) {
value.set(tokenizer.nextToken());
context.write(value, new IntWritable(1));
}
}
}
public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values,Context context)
throws IOException,InterruptedException {
int sum=0;
for(IntWritable x: values)
{
sum+=x.get();
}
context.write(key, new IntWritable(sum));
}
}
public static void main(String[] args) throws Exception {
Configuration conf= new Configuration();
Job job = new Job(conf,"My Word Count Program");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path outputPath = new Path(args[1]);
//Configuring the input/output path from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
//deleting the output path automatically from hdfs so that we don't have to
delete it explicitly
outputPath.getFileSystem(conf).delete(outputPath);
//exiting the job only if the flag value becomes false
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

The entire MapReduce program can be fundamentally divided
into three parts:

- Mapper Phase Code
- Reducer Phase Code
- Driver Code

We will understand the code for each of these three parts sequentially.

Mapper code:

```
public static class Map extends
Mapper<LongWritable,Text,Text,IntWritable> {
public void map(LongWritable key, Text value, Context context) throws
IOException,InterruptedException {
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()) {
value.set(tokenizer.nextToken());
context.write(value, new IntWritable(1));
}
```

We have created a class Map that extends the class Mapper which is already defined in the MapReduce Framework.

- We define the data types of input and output key/value pair after the class declaration using angle brackets.
- Both the input and output of the Mapper is a key/value pair.
- Input:
    - The key is nothing but the offset of each line in the text file:LongWritable
    - The value is each individual line (as shown in the figure at the right): Text
- Output:
    - The key is the tokenized words: Text
    - We have the hardcoded value in our case which is 1: IntWritable
    - Example – Dear 1, Bear 1, etc.
- We have written a java code where we have tokenized each word and assigned them a hardcoded value equal to 1.

Reducer Code:

```
public static class Reduce extends
Reducer<Text,IntWritable,Text,IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values,Context
context)
throws IOException,InterruptedException {
```

```
int sum=0;
for(IntWritable x: values)
{
sum+=x.get();
}
context.write(key, new IntWritable(sum));
}}
```

• We have created a class Reduce which extends class Reducer like that of Mapper.
• We define the data types of input and output key/value pair after the class declaration using angle brackets as done for Mapper.
• Both the input and the output of the Reducer is a keyvalue pair.
• Input:

The key nothing but those unique words which have been generated after the sorting and shuffling phase: Text
The value is a list of integers corresponding to each key: IntWritable
Example – Bear, [1, 1], etc.

• Output:

◦ The key is all the unique words present in the input text file: Text
◦ The value is the number of occurrences of each of the unique words: IntWritable
◦ Example – Bear, 2; Car, 3, etc.

We have aggregated the values present in each of the list corresponding to each key and produced the final answer.
In general, a single reducer is created for each of the unique words, but, you can specify the number of reducer in mapred-site.xml.
Driver Code:

```
        Configuration conf= new Configuration();
        Job job = new Job(conf,"My Word Count Program");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        Path outputPath = new Path(args[1]);
//Configuring the input/output path from the filesystem into the job
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

• In the driver class, we set the configuration of our
MapReduce job to run in Hadoop.
• We specify the name of the job , the data type of input/
output of the mapper and reducer.
• We also specify the names of the mapper and reducer
classes.
• The path of the input and output folder is also specified.
• The method setInputFormatClass () is used for specifying that how a Mapper will read the input data or what will be the unit of work. Here, we have chosen TextInputFormat so that single line is read by the mapper at a time from the input text file.
• The main () method is the entry point for the driver.
In this method, we instantiate a new Configuration object for the job.

Run the MapReduce code:

The command for running a MapReduce code is:

```
hadoop jar hadoop-mapreduce-example.jar WordCount /
sample/input /sample/output
```