G Pulla Reddy Engineering College (Autonomous): Kurnool

Department of Computer Science & Engineering

Lab Manual

Class	**B.Tech CSE VII Semester**
Course	**Big Data Analytics (BDA (P))**
Course Code	**CS404**
Scheme	**2017**
Academic Year	**2020-21**
Prepared by	**Dr. C. Sreedhar**

List of Experiments

1. Perform Hadoop setup in Local and Pseudo mode and monitor through web based UI.
2. Implementation of Hadoop shell commands on files.
3. Implementation of word count example using Hadoop MapReduce.
4. Write a MapReduce program that works on Gutenberg data.
5. Write a MapReduce program that mines weather data.
6. Write pig latin scripts on Describe, for each and order by operator.
7. Write pig latin scripts to perform set and sort operation.
8. Perform DDL operations on Hive.
9. Implementation of data management using NOSQL databases.

Video Tutorials	
https://www.youtube.com/channel/UC_6mhzMATOtsC1UXO0sHpwA	
Topic	**Youtube link**
Install Ubuntu in Virtualbox	https://www.youtube.com/watch?v=2QVz7715n5g
run Wordcount MapReduce	https://www.youtube.com/watch?v=G0xyw1ODi5A
MapReduce on Gutenberg	https://www.youtube.com/watch?v=q8INOCrU9HE
Pig Latin Operators	https://www.youtube.com/watch?v=2N9gP1l9_F4

01.	**Perform Hadoop setup in Local and Pseudo mode and monitor through web based UI.**
Expected Output	a) Successful installation of Hadoop in local, pseudo mode **hadoop version** b) Monitor Namenode,secondarynamenode,datanode,YARN RM, YARN NM information

Local (Standalone) mode:

Step	Details
1.	Prerequisites: a) VMWare b) Ubuntu 18.04 c) Jdk 8 d) Hadoop 2.10.0
2.	*Open Terminal and type in the following command* sudo apt-get install openjdk-8-jdk
3.	*Check whether java is installed or not using the command* java –version
4.	*Download Hadoop 2.10.0*
5.	cd /Downloads
6.	sudo tar xvf hadoop-2.10.0.tar.gz
7.	sudo mv hadoop-2.10.0 /opt
8.	cd /
9.	cd opt
10.	sudo chmod 777 hadoop-2.10.0
11.	cd /home/Sreedhar
12.	sudo gedit .bashrc *At the end of the file (after fi) add the following (export JAVA_HOME...)*

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi


export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
alias jps='/usr/lib/jvm/java-8-openjdk-amd64/bin/jps'
export HADOOP_HOME=/opt/hadoop-2.10.0/
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
```

13.	source .bashrc
14.	hadoop version

Pseudo mode

Step	Details
1.	Prerequisites: a) VMWare b) Ubuntu 18.04 c) Jdk 8 d) Hadoop 2.10.0
2.	*Open Terminal and type in the following command* sudo apt-get install openjdk-8-jdk
3.	*Check whether java is installed or not using the command* java –version
4.	sudo su
5.	adduser hduser *(Give password)*
6.	usermod –aG sudo hduser
7.	sudo su hduser
8.	sudo apt-get purge openssh-server
9.	sudo apt-get install openssh-server
10.	ssh-keygen –t rsa
11.	cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
12.	ssh localhost
13.	cd /home/hduser
14.	*Download Hadoop 2.10.0*
15.	sudo tar xvf hadoop-2.10.0.tar.gz
16.	sudo mv /home/hduser/hadoop-2.10.0 /opt
17.	cd /
18.	cd opt
19.	sudo chmod 777 hadoop-2.10.0
20.	cd /home/hduser
21.	sudo gedit .bashrc *At the end of the file add export JAVA_HOME...(Same as local mode)*
22.	source .bashrc
23.	cd /
24.	cd opt
25.	cd hadoop-2.10.0
26.	cd etc
27.	cd hadoop
28.	sudo gedit hadoop-env.sh *replace the following export JAVA_HOME=${JAVA_HOME}* `# The java implementation to use.` `#export JAVA_HOME=${JAVA_HOME}` `export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64`
29.	sudo gedit core-site.xml

add the following between <configuration> </configuration>

```
<configuration>
 <property>
  <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
    <value>/opt/hadoop-2.10.0/tmp</value>
</property>
</configuration>
```

30. sudo gedit hdfs-site.xml
add the following between <configuration> </configuration>

```
<configuration>
<property>
      <name>dfs.replication</name>
      <value>1</value>
 </property>
 <property>
      <name>dfs.namenode.name.dir</name>
      <value>/home/hduser/hadoop_tmp/hdfs/namenode</value>
 </property>
 <property>
      <name>dfs.datanode.data.dir</name>
      <value>/home/hduser/hadoop_tmp/hdfs/datanode</value>
 </property>
</configuration>
```

31. sudo gedit yarn-site.xml
add the following between <configuration> </configuration>

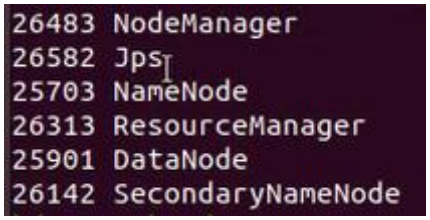
```
<configuration>
<property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
</property>
<property>
      <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
      <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<!-- Site specific YARN configuration properties -->

</configuration>
```

32. sudo cp mapred-site.xml.template mapred-site.xml
33. sudo gedit yarn-site.xml
add the following between <configuration> </configuration>

```
<configuration>
 <property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

34. cd /home/hduser
35. sudo mkdir –p hadoop_tmp/hdfs/namenode
36. sudo mkdir –p hadoop_tmp/hdfs/datanode
37. sudo chmod 777 –R hadoop_tmp/hdfs/namenode
38. sudo chmod 777 –R hadoop_tmp/hdfs/datanode
39. sudo chown –R hduser hadoop_tmp/hdfs/datanode
40. hdfs namenode -format
41. start-dfs.sh

42. start-yarn.sh
43. jps
    jps command shows the following output



44. To stop all hadoop daemon services, use the following command
    stop-dfs.sh
    stop-yarn.sh

Monitor through Web based UI	
Namenode information	localhost:50070
Secondarynamenode information	localhost:50090
Datanode information	localhost:50075
YARN Resource Manager	localhost:8088
YARN Node Manager	localhost:8042

02. Implementation of Hadoop shell commands on files

Syntax and Description	Example (Usage)
hadoop **version** displays the version of hadoop installed in the system	hadoop version
hadoop fs -**ls** / *Displays List of Files and Directories in HDFS file Path*	hadoop fs -ls /
hadoop fs –**mkdir** *create a directory on an HDFS environment.*	hadoop fs -mkdir /user/hadoop/
hadoop fs –**put** *used to copy files from the local file system to the HDFS filesystem*	hadoop fs -put sample.txt /user/data/
hadoop fs –**get** *used to copy files from HDFS file system to the local file system, just the opposite to put command.*	hadoop fs -get /user/data/sample.txt workspace/
hadoop fs -**cat** URI [URI …] *used for displaying the contents of a file on the console.*	hadoop fs -cat /user/data/sampletext.txt
hadoop fs -**cp** URI [URI …] <dest> *Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.*	hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2
hadoop fs -**appendToFile** <localsrc> … <dst> *Append single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and appends to destination file system.*	hadoop fs -appendToFile localfile /user/hadoop/hadoopfile

hadoop fs -**df** URI [URI …] *Displays free space*	hadoop dfs -df /user/hadoop/dir1
hadoop fs –**help**	hadoop fs –help
hadoop fs -**touchz** URI [URI …] *Create a file of zero length. An error is returned if the file exists with non-zero length*	hadoop -touchz pathname
hadoop fs -**rmdir** URI [URI …] *Delete a directory*	hadoop fs -rmdir /user/hadoop/emptydir
hadoop fs -**mv** URI [URI …] <dest> *Moves files from source to destination. This command allows multiple sources as well in which case the destination needs to be a directory.*	hadoop fs -mv /user/hadoop/file1 /user/hadoop/file2

03.	**Implementation of word count example using Hadoop MapReduce**
Expected Output	Display the frequency of each distinct word in the user given input.

Step	**Details**
1.	Prerequisites: a) VMWare or Virtualbox b) Cloudera (CDH5)
2.	File → New → Java Project → Project Name as WordCount → Libraries → Add External Jars
3.	Open Terminal cat > /home/cloudera/inputFile.txt --Enter words
4.	hdfs dfs -mkdir /inputnew hdfs dfs -put /home/cloudera/inputFile.txt /inputnew/
5.	hdfs dfs -cat /inputnew/inputFile.txt
6.	hadoop jar /home/cloudera/wordcount.jar WordCount /inputnew/inputFile.txt /output_new
7.	hdfs dfs -cat /output_new/part-r-00000

```
import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws
                       IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
```

```
      context.write(key, result);
    }
  }
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

04.	**Write a MapReduce program that works on Gutenberg data.**
Expected Output	Display the frequency of each distinct word in the Gutenberg dataset.

Step	**Details**
1.	Prerequisites: a) VMWare or Virtualbox b) Cloudera (CDH5)
2.	Download gutenberg dataset and paste into gutenbergdata folder http://www.gutenberg.org/cache/epub/4300/pg4300.txt
3.	Follow the similar steps as Wordcount MapReduce program
4.	Open Terminal
5.	Type the command: hdfs dfs -mkdir /guteninput
6.	hdfs dfs -put /home/cloudera/gutenbergdata/pg4300.txt /guteninput/
7.	hadoop jar /home/cloudera/Wordcount.jar WordCount /guteninput/pg4300.txt /gutenoutput
8.	hdfs dfs -cat /gutenoutput/part-r-00000
9.	You can also use hdfs dfs -cat /gutenoutput/* command instead of step 19

Source code:

Same as Wordcount MapReduce program

05.	**Write a MapReduce program that mines weather data.**
Expected Output	Generate the output with Max and min temperature for city: 25-Jan-2014 Time: 12:34:542 MinTemp: -22.3 Time: 05:12:345 MaxTemp: 35.7

Step	**Details**
1.	Prerequisites: a) VMWare or Virtualbox b) Cloudera (CDH5)
2.	Download the dataset (save in weatherdata folder) and jar file: https://drive.google.com/file/d/0B-ur4R5mlgGLcVRZMTZGekRpZWM/view https://drive.google.com/file/d/0B-ur4R5mlgGLMzVyTmdITTVmbjA/view
3.	Select File --> New --> Class --> Give name as CalculateMaxAndMinTemeratureWithTime
4.	Click on Finish
5.	Save the source code and name it as CalculateMaxAndMinTemeratureWithTime.java into workspace
6.	Open Terminal
7.	Type the command: hdfs dfs -mkdir /weatherinput
8.	hdfs dfs -put /home/cloudera/weatherdata/input_temp.txt /weatherinput/
9.	hadoop jar /home/cloudera/WeatherReportPOC.jar CalculateMaxAndMinTemeratureWithTime /weatherinput/input_temp.txt /weatheroutput
10.	hdfs dfs -cat /gutenoutput/Austin-r-00000

Source code:

```
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.MultipleOutputs;
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.FileOutputFormat;
```

```
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


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);
 }
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(CalculateMaxAndMinTemeratureWithTime.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();
 }
}
}
```

06.	**Write pig latin scripts on Describe, for each and order by operator**
Expected Output	Display the output for the operators: describe, foreach and order by

Operator	Description
DESCRIBE	**describe** operator is used to view the schema of a relation. Usage: DESCRIBE relationname;
FOREACH	**FOREACH** operator is used to generate specified data transformations based on the column data. Usage: relationname2 = FOREACH relationname1 GENERATE (required columndata);
ORDER BY	**ORDER BY** operator is used to display the contents of a relation in a sorted order based on one or more fields. Usage: relationname2 = ORDER relationname1 BY (ASC|DESC);

Step	**Details**
1.	Prerequisites: a) VMWare or Virtualbox b) Cloudera (CDH5)
2.	Open Terminal and type the command: pig
3.	gprec_data = LOAD 'gprec.txt' using PigStorage(',') as (branchid:int, branch:chararray,strength:int) *Assuming gprec.txt contains data*
4.	DUMP gprec_data;
5.	DESCRIBE gprec_data;
6.	foreach_opr = FOREACH gprec_data GENERATE branch,strength;
7.	DUMP foreach_opr;
8.	foreach_opr2 = FOREACH gprec_data GENERATE lower(branch); DUMP foreach_opr2;
9.	orderby_opr = ORDER gprec_data BY strength DESC;
10.	DUMP orderby_opr;

07.	Write pig latin scripts to perform set and sort operation

Set Operation: UNION

UNION operator of Pig Latin is used to merge the content of two relations.

To perform UNION operation on two relations, their columns and domains must be identical.

Syntax:
grunt> relationname3 = UNION relationname1, relationname2;

```
student1 = LOAD 'student1_data.txt' using PigStorage(',') as (studentid:int,
studentname:chararray,percentage:int)

student2 = LOAD 'student2_data.txt' using PigStorage(',') as (studentid:int,
studentname:chararray,percentage:int)

grunt> student = UNION student1, student2;

grunt> DUMP student
```

Set Operation: Join

Used to combine two or more relations

Assuming the files (customers.txt)	Order.txt
1,Ramesh,32,Ahmedabad,2000.00 2,Suresh,25,Delhi,1500.00 3,kuresh,23,Kota,2000.00 4,Kalesh,25,Mumbai,6500.00 5,Sailesh,27,Bhopal,8500.00 6,Komal,22,MP,4500.00 7,Dinesh,24,Indore,10000.00	102,2009-10-08 00:00:00,3,3000 100,2009-10-08 00:00:00,3,1500 101,2009-11-20 00:00:00,2,1560 103,2008-05-20 00:00:00,4,2060

```
grunt>customers = load '/home/cloudera/customers.txt' using PigStorage(',')as
(id:int, name:chararray, age:int, address:chararray, salary:int);
 grunt>orders = load 'home/cloudera/orders.txt' using PigStorage(',')as (oid:int,
date:chararray, customer_id:int, amount:int);
```

Self-join is used to join a table with itself as if the table were two relations.
Syntax: Relation3_name = join Relation1_name BY key, Relation2_name BY key

```
grunt> cust_realation1 = load '/home/cloudera/customers.txt' using
PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int);
grunt> cust_realation2 = load '/home/cloudera/customers.txt' using
PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int);
grunt> customers3 = JOIN cust_relation1 BY id, cust_relation2 BY id;
```

Inner Join

Inner join returns rows when there is a match in both tables.

Syntax: Relation3_name = join Relation1_name BY key, Relation2_name BY key

grunt> cust_realation1 = load '/home/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int);

grunt> cust_realation2 = load '/home/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:

grunt> customers3 = JOIN cust_relation1 BY id, cust_relation2 BY id;

SORT Operation

Assume the file (raw_sales.txt) with the following contents
CatZ,Prod22-cZ,30,60
CatA,Prod88-cA,15,50
CatY,Prod07-cY,20,40
CatB,Prod18-cB,10,50
CatX,Prod29-cZ,40,60
CatC,Prod09-cC,80,140
CatZ,Prod83-cZ,20,60
CatA,Prod17-cA,25,50
CatY,Prod98-cY,10,40
CatB,Prod99-cB,30,50
CatX,Prod19-cZ,10,60
CatC,Prod73-cC,50,140
CatZ,Prod52-cZ,10,60
CatA,Prod58-cA,15,50
CatY,Prod57-cY,10,40
CatB,Prod58-cB,10,50
CatX,Prod59-cZ,10,60
CatC,Prod59-cC,10,140

grunt> rawSales = LOAD 'raw_sales.txt' USING PigStorage(',') AS (category: chararray, product: chararray, sales: long, total_sales_category: long);
grunt> DUMP rawSales;

grpByCatTotals = GROUP rawSales BY (total_sales_category, category);
grunt> DUMP grpByCatTotals

sortGrpByCatTotals = ORDER grpByCatTotals BY group DESC;
grunt> sortGrpByCatTotals

topSalesCats = LIMIT sortGrpByCatTotals 2;
grunt> topSalesCats

08.	Perform DDL operations on Hive

DDL: Data Definition Language

1. CREATE
2. ALTER
3. DROP

CREATE TABLE

Creates a new table and specifies its characteristics.

```
hive> CREATE TABLE Employee (empid INT, empname STRING, empcity STRING);

hive> describe Employee;

hive> insert into Employee values (200,'Sreedhar','Kurnool');

hive> select * from Employee;
```

ALTER TABLE

Alter Table statement is used to alter a table in Hive.

```
hive> ALTER TABLE Employee RENAME to GPREmployee

hive> desc GPREmployee;

hive> ALTER TABLE GPREmployee ADD COLUMNS (Sal BIGINT);
```

DROP TABLE

DROP TABLE removes the table in Hive

```
hive> DROP TABLE GPREmployee;

hive> desc GPREmployee
```

09.	Implementation of data management using NOSQL databases.

HBASE:

HBase is a column oriented database management system derived from Google's NoSQL database BigTable that runs on top of HDFS.

Create table: Creates a table

```
hbase> create 'st_percentage', 'Rollno', 'Percentage'
```

Describe (or) **desc**: command returns the description of the table

```
hbase> desc 'st_percentage'
```

Insert: command used to insert the values into the table

```
hbase> Insert values into table: put 'st_percentage', '1001', 'Percentage:upto7thsem','98'
```

scan: command is used to view the data in table

```
hbase> scan 'st_percentage'
```

Alter: command used to make changes to an existing table

```
hbase> alter 'st_percentage','delete'=>'percentage'
```

disable: To delete a table, the table has to be disabled first using the disable command

```
hbase> disable 'st_percentage'
```

enable: command used to enable the table

```
hbase> enable 'st_percentage'
```

drop: command used to delete a table. Before dropping a table, it must be disabled.

```
hbase> drop 'st_percentage'
```

exists: command used to verify, whether the table is present in the database or not.

```
hbase> exists 'st_percentage'
```