PROGRAM CODE

1a) Is the number of petitions with Data Engineer job title increasing over time?

QUERY : select year,COUNT(job\_title) from h1b\_final where job\_title=="DATA ENGINEER" group by year order by year;

1b)Find top 5 job titles who are having highest avg growth in application

CODE:

--Find top 5 job titles who are having highest avg growth in applications.

data = LOAD '/niit/h1b' USING PigStorage('\t') AS ( s\_no:int, case\_status:chararray,employer\_name:chararray, soc\_name:chararray, jobtitle:chararray, full\_time\_position:chararray, wage:long, year:chararray, worksite:chararray, longitude:long, latitude:long );

--dump data;

data = foreach data generate $4 as job, $7 as year;

--dump data;

dat = filter data by year=='2011';

a = group dat by job;

--dump a;

b1 = foreach a generate group as job, (float)COUNT(dat.job) as c1;

--dump b1;

dat = filter data by year=='2012';

a = group dat by job;

--dump a;

b2 = foreach a generate group as job, (float)COUNT(dat.job) as c2;

--dump b2;

dat = filter data by year=='2013';

a = group dat by job;

--dump a;

b3 = foreach a generate group as job, (float)COUNT(dat.job) as c3;

--dump b3;

dat = filter data by year=='2014';

a = group dat by job;

--dump a;

b4 = foreach a generate group as job, (float)COUNT(dat.job) as c4;

--dump b4;

dat = filter data by year=='2015';

a = group dat by job;

--dump a;

b5 = foreach a generate group as job, (float)COUNT(dat.job) as c5;

--dump b5;

dat = filter data by year=='2016';

a = group dat by job;

--dump a;

b6 = foreach a generate group as job, (float)COUNT(dat.job) as c6;

--dump b6;

c = join b1 by $0, b2 by $0;

c = foreach c generate $0, $1 as a11, $3 as a12;

--dump c;

d1 = foreach c generate $0, (a12-a11)/a11 \*100 as growth;

--dump d1;

c = join b2 by $0, b3 by $0;

c = foreach c generate $0, $1 as a12, $3 as a13;

--dump c;

d2 = foreach c generate $0, (a13-a12)/a12 \*100 as growth;

--dump d2;

c = join b3 by $0, b4 by $0;

c = foreach c generate $0, $1 as a13, $3 as a14;

--dump c;

d3 = foreach c generate $0, (a14-a13)/a13 \*100 as growth;

--dump d3;

c = join b4 by $0, b5 by $0;

c = foreach c generate $0, $1 as a14, $3 as a15;

--dump c;

d4 = foreach c generate $0, (a15-a14)/a14 \*100 as growth;

--dump d4;

c = join b5 by $0, b6 by $0;

c = foreach c generate $0, $1 as a15, $3 as a16;

--dump c;

d5 = foreach c generate $0, (a16-a15)/a15 \*100 as growth;

--dump d1;

final = join d1 by $0, d2 by $0, d3 by $0, d4 by $0, d5 by $0;

final = foreach final generate $0,$1,$3,$5,$7,$9;

--dump final;

final1 = foreach final generate $0, ($1+$2+$3+$4+$5)/5 as avg;

--dump final1;

e = limit( order final1 by avg desc) 5;

dump e;

2a)Which part of the US has the most Data Engineer jobs for each year?

CODE:

import java.io.\*;

import java.util.TreeMap;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.Reducer.Context;

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

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

import org.apache.hadoop.io.NullWritable;

public class Proj2a

{

public static class MapClass extends Mapper<LongWritable,Text,Text,Text>

{

public void map(LongWritable key, Text value, Context context)

{

try{

String[] str = value.toString().split("\t");

if(str[4].equals("DATA ENGINEER"))

{

//String jobandyear = str[4]+","+str[7];

//worksite and year

context.write(new Text(str[8]),new Text(str[7]));

}}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

public static class CaderPartitioner extends

Partitioner < Text, Text >

{

public int getPartition(Text key, Text value, int numReduceTasks)

{

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

int year = Integer.parseInt(str[0]);

if(year==2011)

{

return 0 % numReduceTasks;

}

else if(year==2012)

{

return 1 % numReduceTasks ;

}

else if(year==2013)

{

return 2 % numReduceTasks ;

}

else if(year==2014)

{

return 3 % numReduceTasks ;

}

else if(year==2015)

{

return 4 % numReduceTasks ;

}

else

{

return 5 % numReduceTasks;

}

}

}

public static class ReduceClass extends Reducer<Text,Text,NullWritable,Text>

{

private LongWritable result = new LongWritable();

private TreeMap<Long,Text> repToRecordMap = new TreeMap<Long,Text>();

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

long count=0;

String xx="";

String aa="";

for (Text val : values)

{//String[] str = val.toString().split(",");

//String job = str[0];

//if(job=="DATA ENGINEER")

//{

count++;

aa= val.toString();

//}

//key=new Text(job);

}

//result.set(count);

xx= key.toString();

//aa=values.toString();

xx= xx + ','+count+','+aa;

repToRecordMap.put(new Long(count), new Text(xx));

//repToRecordMap.put(key,result);

if (repToRecordMap.size() > 1) {

repToRecordMap.remove(repToRecordMap.firstKey()); //to remove 1st elements

//repToRecordMap.remove(repToRecordMap.lastKey()); To remove last elements

}

//context.write(key, result);

//context.write(key, new LongWritable(sum));

}

protected void cleanup(Context context) throws IOException,

InterruptedException {

for (Text t : repToRecordMap.descendingMap().values()) {

// Output our five records to the file system with a null key

context.write(NullWritable.get(), t);

}

}

}

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

Configuration conf = new Configuration();

//conf.set("name", "value")

//conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");

Job job = Job.getInstance(conf, "Volume Count");

job.setJarByClass(Proj2a.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setPartitionerClass(CaderPartitioner.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(6);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}

}

2b)find top 5 locations in the US who have got certified visa for each year

This is analyzed using Hive.

QUERY: select worksite,count(case\_status) as c,year from retail.h1b\_final where case\_status='CERTIFIED' and year==$y group by worksite,year order by c desc limit 5;

3)Which industry(SOC\_NAME) has the most number of Data Scientist positions?

QUERY:select soc\_name,COUNT(job\_title) as c from retail.h1b\_final where case\_status='CERTIFIED' and job\_title='DATA SCIENTIST' group by soc\_name order by c desc limit 1;

4)Which top 5 employers file the most petitions each year?

QUERY:select employer\_name,COUNT(case\_status) as ca,year from retail.h1b\_final where year==$y group by employer\_name,year order by ca desc limit 5;

5a)Find the most popular top 10 job positions for H1B visa applications for each year for all the applications?

CODE:

import java.io.\*;

import java.util.TreeMap;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.Reducer.Context;

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

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

import org.apache.hadoop.io.NullWritable;

public class Proj5

{

public static class MapClass extends Mapper<LongWritable,Text,Text,Text>

{

public void map(LongWritable key, Text value, Context context)

{

try{

String[] str = value.toString().split("\t");

//String job = str[7]+','+str[3];

context.write(new Text(str[4]),new Text(str[7]));

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

public static class CaderPartitioner extends

Partitioner < Text, Text >

{

public int getPartition(Text key, Text value, int numReduceTasks)

{

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

int year = Integer.parseInt(str[0]);

if(year==2011)

{

return 0 % numReduceTasks;

}

else if(year==2012)

{

return 1 % numReduceTasks ;

}

else if(year==2013)

{

return 2 % numReduceTasks ;

}

else if(year==2014)

{

return 3 % numReduceTasks ;

}

else if(year==2015)

{

return 4 % numReduceTasks ;

}

else

{

return 5 % numReduceTasks;

}

}

}

public static class ReduceClass extends Reducer<Text,Text,NullWritable,Text>

{

private LongWritable result = new LongWritable();

private TreeMap<Long,Text> repToRecordMap = new TreeMap<Long,Text>();

String xx="";

String aa="";

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

long count=0;

for (Text val : values)

{

count++;

aa= val.toString();

}

//result.set(count);

//context.write(key, result);

xx= key.toString();

xx= xx + ','+count+','+aa;

repToRecordMap.put(new Long(count), new Text(xx));

if (repToRecordMap.size() > 10) {

repToRecordMap.remove(repToRecordMap.firstKey()); //to remove 1st elements

//repToRecordMap.remove(repToRecordMap.lastKey()); To remove last elements

}

}

protected void cleanup(Context context) throws IOException,

InterruptedException {

for (Text t : repToRecordMap.descendingMap().values()) {

// Output our five records to the file system with a null key

context.write(NullWritable.get(), t);

}

}

}

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

Configuration conf = new Configuration();

//conf.set("name", "value")

//conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");

Job job = Job.getInstance(conf, "Volume Count");

job.setJarByClass(Proj5.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setPartitionerClass(CaderPartitioner.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(6);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}

}

5b)Find the most popular top 10 job positions for H1B visa applications for each year or only certified applications

This is analysed using Map reduce

CODE:

import java.io.\*;

import java.util.TreeMap;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.Reducer.Context;

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

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

import org.apache.hadoop.io.NullWritable;

public class Proj5b

{

public static class MapClass extends Mapper<LongWritable,Text,Text,Text>

{

public void map(LongWritable key, Text value, Context context)

{

try{

String[] str = value.toString().split("\t");

//String job = str[7]+','+str[3];

if(str[1].contains("CERTIFIED"))

{

context.write(new Text(str[4]),new Text(str[7]));

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

public static class CaderPartitioner extends

Partitioner < Text, Text >

{

public int getPartition(Text key, Text value, int numReduceTasks)

{

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

int year = Integer.parseInt(str[0]);

if(year==2011)

{

return 0 % numReduceTasks;

}

else if(year==2012)

{

return 1 % numReduceTasks ;

}

else if(year==2013)

{

return 2 % numReduceTasks ;

}

else if(year==2014)

{

return 3 % numReduceTasks ;

}

else if(year==2015)

{

return 4 % numReduceTasks ;

}

else

{

return 5 % numReduceTasks;

}

}

}

public static class ReduceClass extends Reducer<Text,Text,NullWritable,Text>

{

private LongWritable result = new LongWritable();

private TreeMap<Long,Text> repToRecordMap = new TreeMap<Long,Text>();

String xx="";

String aa="";

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

long count=0;

for (Text val : values)

{

count++;

aa= val.toString();

}

//result.set(count);

//context.write(key, result);

xx= key.toString();

xx= xx + ','+count+','+aa;

repToRecordMap.put(new Long(count), new Text(xx));

if (repToRecordMap.size() > 10) {

repToRecordMap.remove(repToRecordMap.firstKey()); //to remove 1st elements

//repToRecordMap.remove(repToRecordMap.lastKey()); To remove last elements

}

}

protected void cleanup(Context context) throws IOException,

InterruptedException {

for (Text t : repToRecordMap.descendingMap().values()) {

// Output our five records to the file system with a null key

context.write(NullWritable.get(), t);

}

}

}

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

Configuration conf = new Configuration();

//conf.set("name", "value")

//conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");

Job job = Job.getInstance(conf, "Volume Count");

job.setJarByClass(Proj5b.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setPartitionerClass(CaderPartitioner.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(6);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}

}

6)Find the percentage and the count of each case status on total applications for each year. Create a line graph depicting the pattern of All the cases over the period of time.

This is analyzed using pig

CODE:

data = LOAD '/home/hduser/h1b' USING PigStorage('\t') AS ( s\_no:int, case\_status:chararray,employer\_name:chararray, soc\_name:chararray, jobtitle:chararray, full\_time\_position:chararray, wage:long, year:chararray, worksite:chararray, longitude:long, latitude:long );

--dump data;

data = foreach data generate $1 as s, $7 as year;

--dump data;

--q = group data all;

--r = foreach q generate (float)COUNT(data.s) as tot;

--dump r;

x = filter data by year=='2011';

--dump x;

q = group x all;

r = foreach q generate (float)COUNT(x.s) as tot;

a = group x by $0;

--dump a;

b = foreach a generate group as status, (float)COUNT(x.s) as c, FLATTEN(x.year);

--dump b;

a11 = DISTINCT b;

--dump a11;

w1 = foreach a11 generate $0, $1 as cc, $2 , ROUND\_TO((c\*100)/r.tot ,2) ;

--ww = foreach w generate $3/$1;

--dump w1;

x1 = filter data by year=='2012';

--dump x;

q = group x1 all;

r = foreach q generate (float)COUNT(x1.s) as tot;

a = group x1 by $0;

--dump a;

b = foreach a generate group as status, (float)COUNT(x1.s) as c, FLATTEN(x1.year);

--dump b;

a11 = DISTINCT b;

--dump a11;

w2 = foreach a11 generate $0, $1 as cc, $2 , ROUND\_TO((c\*100)/r.tot ,2) ;

--dump w2;

x = filter data by year=='2013';

--dump x;

q = group x all;

r = foreach q generate (float)COUNT(x.s) as tot;

a = group x by $0;

--dump a;

b = foreach a generate group as status, (float)COUNT(x.s) as c, FLATTEN(x.year);

--dump b;

a11 = DISTINCT b;

--dump a11;

w3 = foreach a11 generate $0, $1 as cc, $2 , ROUND\_TO((c\*100)/r.tot ,2) ;

--dump w3;

x = filter data by year=='2014';

--dump x;

q = group x all;

r = foreach q generate (float)COUNT(x.s) as tot;

a = group x by $0;

--dump a;

b = foreach a generate group as status, (float)COUNT(x.s) as c, FLATTEN(x.year);

--dump b;

a11 = DISTINCT b;

--dump a11;

w4 = foreach a11 generate $0, $1 as cc, $2 , ROUND\_TO((c\*100)/r.tot ,2) ;

--dump w4;

x = filter data by year=='2015';

--dump x;

q = group x all;

r = foreach q generate (float)COUNT(x.s) as tot;

a = group x by $0;

--dump a;

b = foreach a generate group as status, (float)COUNT(x.s) as c, FLATTEN(x.year);

--dump b;

a11 = DISTINCT b;

--dump a11;

w5 = foreach a11 generate $0, $1 as cc, $2 , ROUND\_TO((c\*100)/r.tot ,2) ;

--dump w5;

x = filter data by year=='2016';

--dump x;

q = group x all;

r = foreach q generate (float)COUNT(x.s) as tot;

a = group x by $0;

--dump a;

b = foreach a generate group as status, (float)COUNT(x.s) as c, FLATTEN(x.year);

--dump b;

a11 = DISTINCT b;

--dump a11;

w6 = foreach a11 generate $0, $1 as cc, $2 , ROUND\_TO((c\*100)/r.tot ,2) ;

--dump w6;

qq = union w1,w2,w3,w4,w5,w6;

--dump qq;

pp = order qq by $0 asc,$2 asc ;

--dump pp;

--ww= order pp by $0 asc;

--dump ww;

a =store pp into '/home/hduser/output\_c';

7) Create a bar graph to depict the number of applications for each year

This is analyzed using MapReduce

CODE:

import java.io.\*;

//import java.util.TreeMap;

//import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

//import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

//import org.apache.hadoop.mapreduce.Reducer.Context;

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

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

//import org.apache.hadoop.io.NullWritable;

public class Proj7

{

public static class MapClass extends Mapper<LongWritable,Text,Text,Text>

{

public void map(LongWritable key, Text value, Context context)

{

try{

String[] str = value.toString().split("\t");

//String jobandyear = str[4]+","+str[7];

//year and case status

context.write(new Text(str[7]),new Text(str[1]));

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

public static class ReduceClass extends Reducer<Text,Text,Text,LongWritable>

{

private LongWritable result = new LongWritable();

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

long count=0;

// String xx="";

//String aa="";

for (Text val : values)

{

count++;

// aa= val.toString();

//}

//key=new Text(job);

}

result.set(count);

context.write(key, result);

//context.write(key, new LongWritable(sum));

}

}

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

Configuration conf = new Configuration();

//conf.set("name", "value")

//conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");

Job job = Job.getInstance(conf, "Volume Count");

job.setJarByClass(Proj7.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

//job.setPartitionerClass(CaderPartitioner.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}}

8) Find the average Prevailing Wage for each Job for each YearArrange the output in descending order

This is analyzed using Pig

CODE:

(For full time jobs)

data = LOAD '/niit/h1b' USING PigStorage('\t') AS ( s\_no:int, case\_status:chararray,employer\_name:chararray, soc\_name:chararray, jobtitle:chararray, full\_time\_position:chararray, wage:long, year:chararray, worksite:chararray, longitude:long, latitude:long );

--dump data;

data = foreach data generate $1 as status,$4 as job, $5 as fp, $6 as wage, $7 as year;

--dump data;

x = filter data by status=='CERTIFIED' or status=='CERTIFIED-WITHDRAWN';

--dump x;

c = filter x by fp== 'Y';

--dump c;

d = filter c by year=='2011';

--dump d;

a = group d by job;

--dump a;

b = foreach a generate group as job,AVG(d.wage) as avg,flatten(d.year) as year;

--dump b;

s1 = distinct b;

--dump s1;

d = filter c by year=='2012';

--dump d;

a = group d by job;

--dump a;

b = foreach a generate group as job,AVG(d.wage) as avg,flatten(d.year) as year;

--dump b;

s2 = distinct b;

--dump s2;

d = filter c by year=='2013';

--dump d;

a = group d by job;

--dump a;

b = foreach a generate group as job,AVG(d.wage) as avg,flatten(d.year) as year;

--dump b;

s3 = distinct b;

--dump s3;

d = filter c by year=='2014';

--dump d;

a = group d by job;

--dump a;

b = foreach a generate group as job,AVG(d.wage) as avg,flatten(d.year) as year;

--dump b;

s4 = distinct b;

--dump s4;

d = filter c by year=='2015';

--dump d;

a = group d by job;

--dump a;

b = foreach a generate group as job,AVG(d.wage) as avg,flatten(d.year) as year;

--dump b;

s5 = distinct b;

--dump s5;

d = filter c by year=='2016';

--dump d;

a = group d by job;

--dump a;

b = foreach a generate group as job,AVG(d.wage) as avg,flatten(d.year) as year;

--dump b;

s6 = distinct b;

--dump s6;

xx = union s1, s2,s3,s4,s5,s6;

xx = order xx by $2 asc ,$1 desc;

dump xx;

For part timejobs, the same code is executed with the condition fp==’N’)

9)Which are the employers along with the number of petitions who have the success rate more than 70% in petitions. (total petitions filed 1000 OR more than 1000) ?

This is analyzed using Pig

CODE:

data = LOAD '/niit/h1b' USING PigStorage('\t') AS ( s\_no:int, case\_status:chararray,employer\_name:chararray, soc\_name:chararray, jobtitle:chararray, full\_time\_position:chararray, wage:long, year:chararray, worksite:chararray, longitude:long, latitude:long );

--dump data

c = group data by employer\_name;

--dump c;

--describe c;

g = foreach c generate group as employer, (float)COUNT(data.case\_status) as aa;

--dump g;

e = filter g by aa>=1000;

a = filter data by case\_status=='CERTIFIED' or case\_status=='CERTIFIED-WITHDRAWN';

--dump a;

v = group a by employer\_name;

--dump v;

b= foreach v generate group as employer, (float)COUNT(a.case\_status) as count2;

--dump b;

d = join e by $0, b by $0 ;

--dump d;

e= foreach d generate $0,$1 as g,$3 as b;

--dump e;

f = foreach e generate $0, $1,(b\*100)/g as rate;

--dump f;

q = filter f by rate>70;

--dump q;

q1 = order q by rate desc;

dump q1;

10)Which are the job positions along with the number of petitions which have the success rate more than 70% in petitions (total petitions filed 1000 OR more than 1000)?

This is analyzed using Pig.

CODE:

data = LOAD '/niit/h1b' USING PigStorage('\t') AS ( s\_no:int, case\_status:chararray,employer\_name:chararray, soc\_name:chararray, jobtitle:chararray, full\_time\_position:chararray, wage:long, year:chararray, worksite:chararray, longitude:long, latitude:long );

--dump data

c = group data by jobtitle;

--dump c;

--describe c;

g = foreach c generate group as job, (float)COUNT(data.case\_status) as aa;

--dump g;

e = filter g by aa>=1000;

a = filter data by case\_status=='CERTIFIED' or case\_status=='CERTIFIED-WITHDRAWN';

--dump a;

v = group a by jobtitle;

--dump v;

b= foreach v generate group as job, (float)COUNT(a.case\_status) as count2;

--dump b;

d = join e by $0, b by $0 ;

--dump d;

e= foreach d generate $0,$1 as g,$3 as b;

--dump e;

f = foreach e generate $0, $1,(b\*100)/g as rate;

--dump f;

q = filter f by rate>70;

q = order q by rate desc;

dump q;

--z = store q into '/home/hduser/output\_Proj10';

11)Export result for question no 10 to MySql databaser

This is done using Sqoop.

Before executing th eexport command a table is created in sql where the contents are to be exported.

QUERY: sqoop export --connect jdbc:mysql://localhost/retail --username root --password 'aysh02' --table PROJ11 --update-mode allowinsert --update-key Job --export-dir /niit/part-r-00000 --input-fields-terminated-by '\t';