**H1B PROJECT REPORT**

1 a) Is the number of petitions with Data Engineer Job title increasing over time. (MapReduce).

**Source Code:**

import org.apache.hadoop.io.Text; import org.apache.hadoop.io.DoubleWritable; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.mapreduce.Job; import org.apache.hadoop.mapreduce.Mapper; import org.apache.hadoop.mapreduce.Reducer; import org.apache.hadoop.mapreduce.Reducer.Context; import org.apache.hadoop.conf.\*; import org.apache.hadoop.fs.\*; import org.apache.hadoop.mapreduce.lib.input.\*; import org.apache.hadoop.mapreduce.lib.output.\*; import java.io.\*; import java.util.Map; import java.util.TreeMap; import javax.naming.InterruptedNamingException;

public class DataEngineerJobs {

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

{

private LongWritable val = new LongWritable(1);

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

{

try{

if(key.get()>0)

{

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

if(str[4]!=null && str[4].contains("DATA ENGINEER") && str[7]!=null && str[7]!="NA")

{  
Text data = new Text (str[7]);

context.write(data, val);

}

}

}

catch(Exception e)

{

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

}

}

}

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

{

long count=0;

int i=0

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

{

for (LongWritable value : values)

{

count+=value.get();

}

context.write(key,new LongWritable(count));

}

}

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, "Data Enigineers growth");

job.setJarByClass(DataEngineerJobs.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.class);

//job.setNumReduceTasks(0);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(LongWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

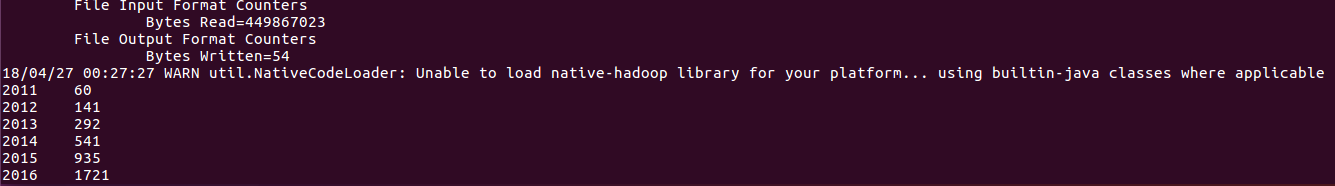
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}

**Output:**

1 b) Find the top 5 job titles who are having highest average growth in all applications. (Pig)

**Source Code:** --Query 1b h1b= load '/home/hduser/Downloads/h1bfinal' using PigStorage ( ) AS (s\_no:int, case\_status, employer\_name, soc\_name, job\_title,full\_time\_position, prevaling\_wage :long , year, worksite, longitude :double , latitude :double);

filterby2011= FILTER h1b by year=='2011'; groupbyjob2011 = GROUP filterby2011 by job\_title; count2011 = foreach groupbyjob2011 generate group as job\_title , COUNT(filterby2011) as headcount1;

filterby2012= FILTER h1b by year=='2012'; groupbyjob2012= GROUP filterby2012 by job\_title; count2012= foreach groupbyjob2012 generate group as job\_title , COUNT(filterby2012) as headcount2;

filterby2013= FILTER h1b by year=='2013'; groupbyjob2013= GROUP filterby2013 by job\_title; count2013= foreach groupbyjob2013 generate group as job\_title , COUNT(filterby2013) as headcount3;

filterby2014= FILTER h1b by year=='2014'; groupbyjob2014 = GROUP filterby2014 by job\_title; count2014 = foreach groupbyjob2014 generate group as job\_title , COUNT(filterby2014) as headcount4;

filterby2015= FILTER h1b by year=='2015'; groupbyjob2015= GROUP filterby2015 by job\_title; count2015= foreach groupbyjob2015 generate group as job\_title , COUNT(filterby2015) as headcount5;

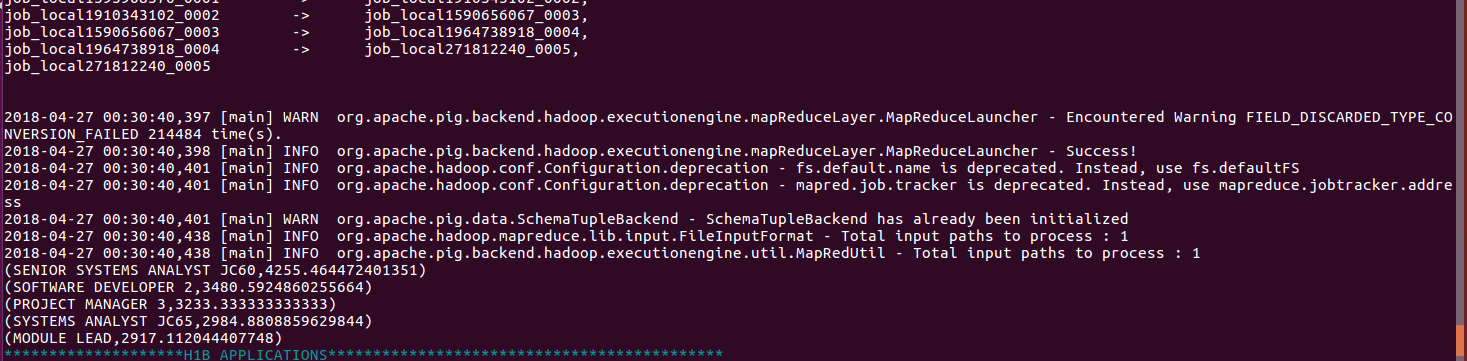
filterby2016= FILTER h1b by year=='2016'; groupbyjob2016= GROUP filterby2016 by job\_title; count2016= foreach groupbyjob2016 generate group as job\_title , COUNT(filterby2016) as headcount6;

join1 = JOIN count2011 by job\_title , count2012 by job\_title, count2013 by job\_title , count2014 by job\_title , count2015 by job\_title , count2016 by job\_title; joinfinal= foreach join1 generate $0, $1, $3, $5, $7, $9, $11;

growthcycle = foreach joinfinal generate $0 , (double)($2-$1)/$1\*100, (double)($3-$2)/$2\*100, (double)($4-$3)/$3\*100,(double)($5-$4)/$4\*100 , (double) ($6-$5)/$5 \*100; averagegrowth = foreach growthcycle generate $0 , ($1+$2+$3+$4+$5)/5 ;

orderby = order averagegrowth by $1 desc; top5jobs = LIMIT orderby 5 ; dump top5jobs;

**Output:**

****

2 a) Which part of US has most number of Data Engineer Jobs for each year? (MapReduce)

**Source Code :**

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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.lib.input.\*;

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

import java.io.\*;

import java.util.TreeMap;

public class Query2a{

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

{

private LongWritable val = new LongWritable(1);

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

{

try{

if(key.get()>0)

{

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

if(str[4]!=null && str[4].contains("DATA ENGINEER"))

{

//str[4]=job title

//str[7]=year

//str[8]=worksite

Text data = new Text (str[8]+"\t"+str[7]);

context.write(data, val);

}

}

}

catch(Exception e)

{

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

}

}

}

//Partitioner class

public static class CaderPartitioner extends

Partitioner < Text, LongWritable >

{

@Override

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

{

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

if(str[1].equals("2011"))

{

return 0 % numReduceTasks;

}

else if(str[1].equals("2012"))

{

return 1 % numReduceTasks ;

}

else if(str[1].equals("2013"))

{

return 2 % numReduceTasks ;

}

else if(str[1].equals("2014"))

{

return 3 % numReduceTasks;

}

else if(str[1].equals("2015"))

{

return 4 % numReduceTasks;

}

else if(str[1].equals("2016"))

{

return 5 % numReduceTasks ;

}

else

{

return 6;

}

}

}

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

{

private TreeMap<LongWritable, Text> worksite = new TreeMap<LongWritable, Text>();

long count=0;

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

{

for(LongWritable val:values)

{

count+=val.get();

}

worksite.put(new LongWritable(count),new Text(key+","+count));

if (worksite.size()>5)

worksite.remove(worksite.firstKey());

}

protected void cleanup(Context context)throws IOException, InterruptedException

{

for (Text t : worksite.descendingMap().values())

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, "Data engineers by worksite on each year");

job.setJarByClass(Query2a.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(LongWritable.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

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

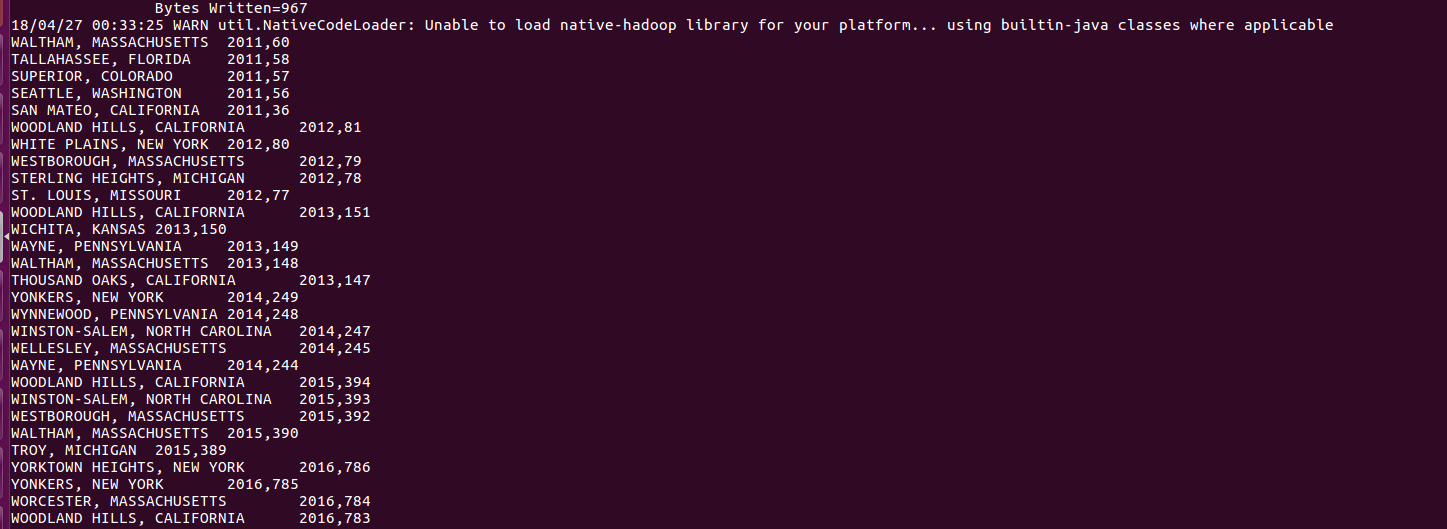
FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}

}

**Output:**



2 b) Find the top 5 locations in US who have got certified visa for each year. (MapReduce)

**Source Code :**

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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.lib.input.\*;

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

import java.io.\*;

import java.util.TreeMap;

public class Query2b{

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

{

private LongWritable val = new LongWritable(1);

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

{

try{

if(key.get()>0)

{

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

if(str[1]!=null && str[1].equals("CERTIFIED"))

{

//str[7]=year

//str[8]=worksite

Text data = new Text (str[8]+"\t"+str[7]);

context.write(data, val);

}

}

}

catch(Exception e)

{

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

}

}

}

//Partitioner class

public static class CaderPartitioner extends

Partitioner < Text, LongWritable >

{

@Override

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

{

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

if(str[1].equals("2011"))

{

return 0 % numReduceTasks;

}

else if(str[1].equals("2012"))

{

return 1 % numReduceTasks ;

}

else if(str[1].equals("2013"))

{

return 2 % numReduceTasks ;

}

else if(str[1].equals("2014"))

{

return 3 % numReduceTasks;

}

else if(str[1].equals("2015"))

{

return 4 % numReduceTasks;

}

else if(str[1].equals("2016"))

{

return 5 % numReduceTasks ;

}

else

{

return 6;

}

}

}

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

{

private TreeMap<LongWritable, Text> worksite = new TreeMap<LongWritable, Text>();

long count=0;

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

{

for(LongWritable val:values)

{

count+=val.get();

}

worksite.put(new LongWritable(count),new Text(key+","+count));

if (worksite.size()>5)

worksite.remove(worksite.firstKey());

}

protected void cleanup(Context context)throws IOException, InterruptedException

{

for (Text t : worksite.descendingMap().values())

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, "worksite with certifed each year");

job.setJarByClass(Query2b.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(LongWritable.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

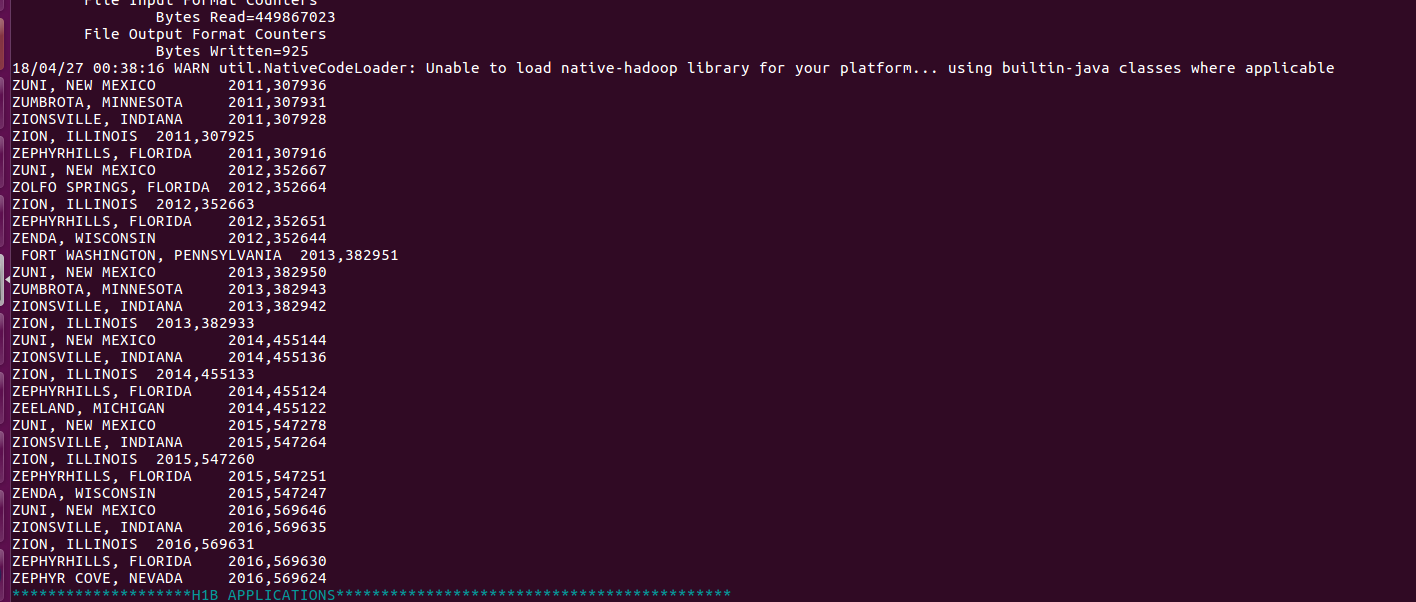
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}

**Output:**

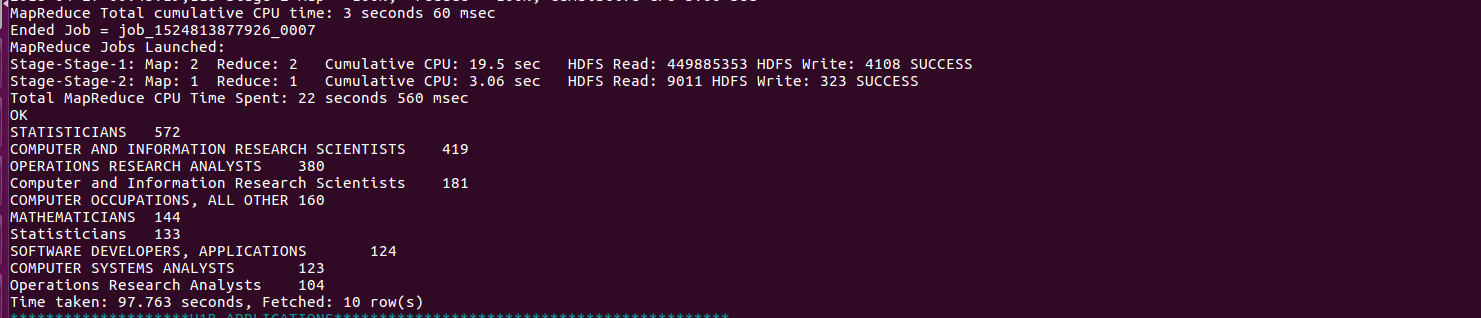
3) Which industry (SOC\_NAME) has most number of Data Scientisy positions.(Certified) (Hive)

**Source code:**

use h1b;

select soc\_name , count(\*) as headcount from h1b\_final where LOWER(job\_title) like "%data scientist%" and case\_status=="CERTIFIED" group by soc\_name order by headcount desc limit 10

**Output:**



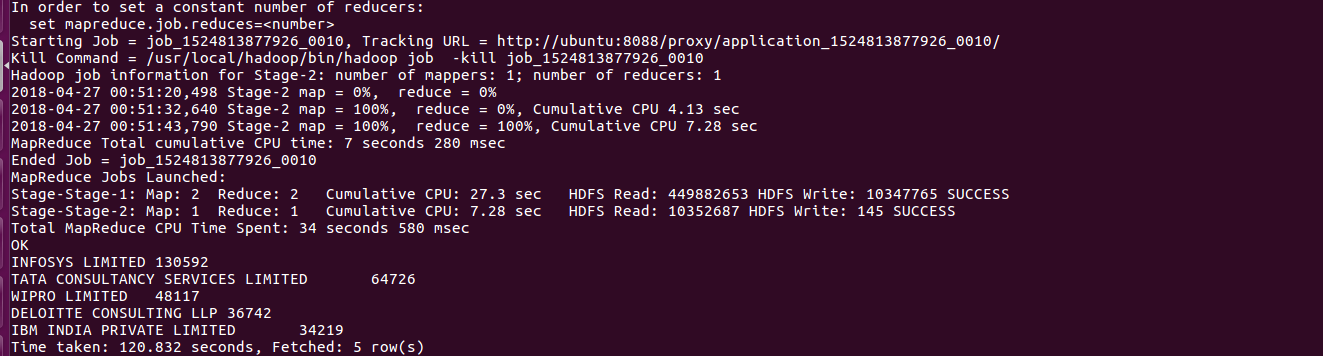
4) Which top 5 employers file the most petitions each year ? Case Status -ALL . (Hive)

**Source code:**

use h1b;

select employer\_name , count(\*) as headcount from h1b\_final group by employer\_name order by headcount desc limit 5;

**Output:**

****

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

**Source Code:**

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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.lib.input.\*;

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

import java.io.\*;

import java.util.TreeMap;

public class Query5a{

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

{

private LongWritable val = new LongWritable(1);

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

{

try{

if(key.get()>0)

{

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

if(str[4]!=null && str[4].contains("DATA ENGINEER"))

{

Text data = new Text (str[4]+"\t"+str[7]);

context.write(data, val);

}

}

}

catch(Exception e)

{

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

}

}

}

//Partitioner class

public static class CaderPartitioner extends

Partitioner < Text, LongWritable >

{

@Override

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

{

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

if(str[1].equals("2011"))

{

return 0 % numReduceTasks;

}

else if(str[1].equals("2012"))

{

return 1 % numReduceTasks ;

}

else if(str[1].equals("2013"))

{

return 2 % numReduceTasks ;

}

else if(str[1].equals("2014"))

{

return 3 % numReduceTasks;

}

else if(str[1].equals("2015"))

{

return 4 % numReduceTasks;

}

else if(str[1].equals("2016"))

{

return 5 % numReduceTasks ;

}

else

{

return 6;

}

}

}

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

{

private TreeMap<LongWritable, Text> job\_title = new TreeMap<LongWritable, Text>();

long count=0;

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

{

for(LongWritable val:values)

{

count+=val.get();

}

job\_title.put(new LongWritable(count),new Text(key+","+count));

if (job\_title.size()>10)

job\_title.remove(job\_title.firstKey());

}

protected void cleanup(Context context)throws IOException, InterruptedException

{

for (Text t : job\_title.descendingMap().values())

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, "Popular Jobs for each year ");

job.setJarByClass(Query5a.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(LongWritable.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

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

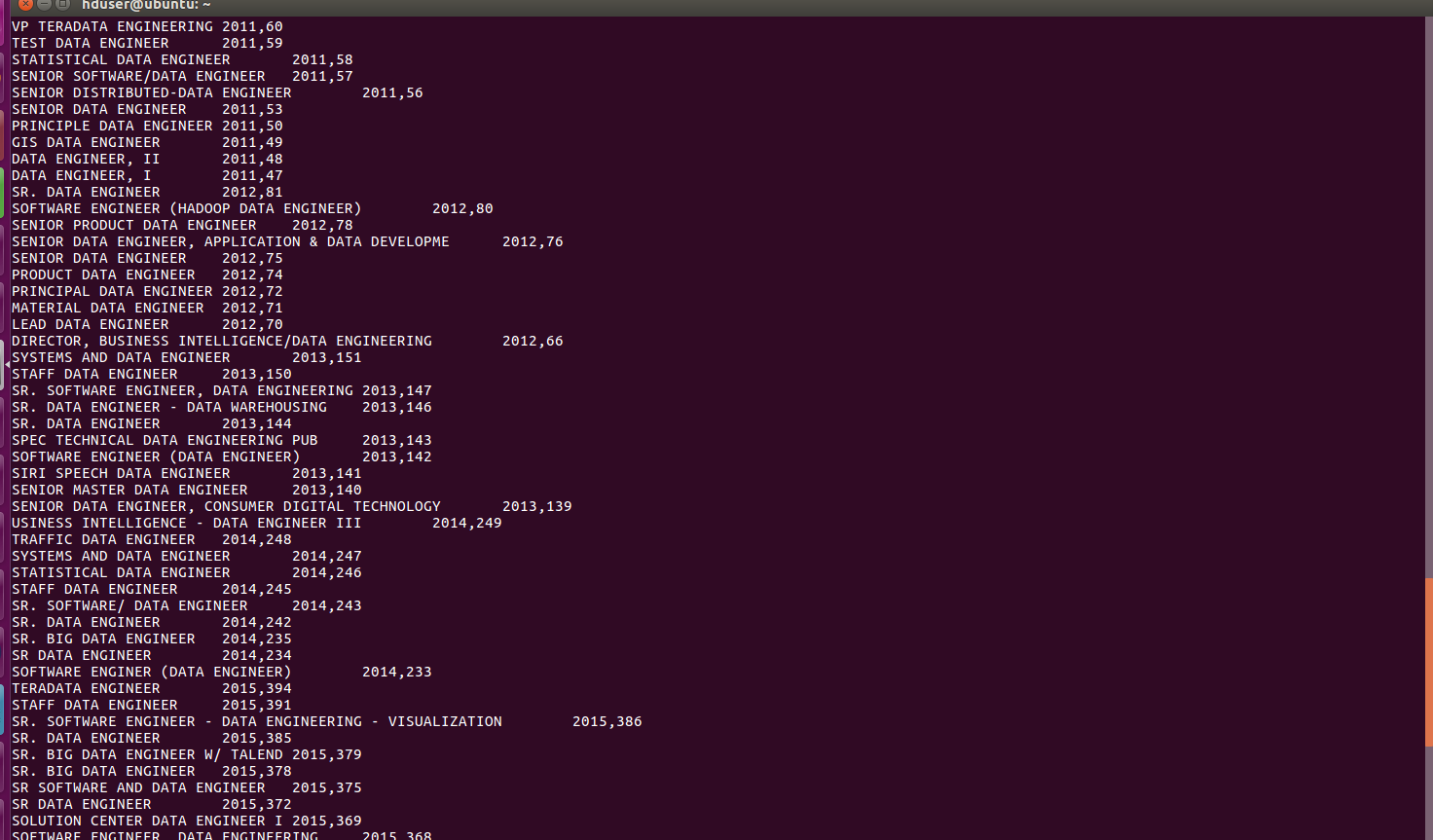
FileOutputFormat.setOutputPath(job, new Path(args[1]));

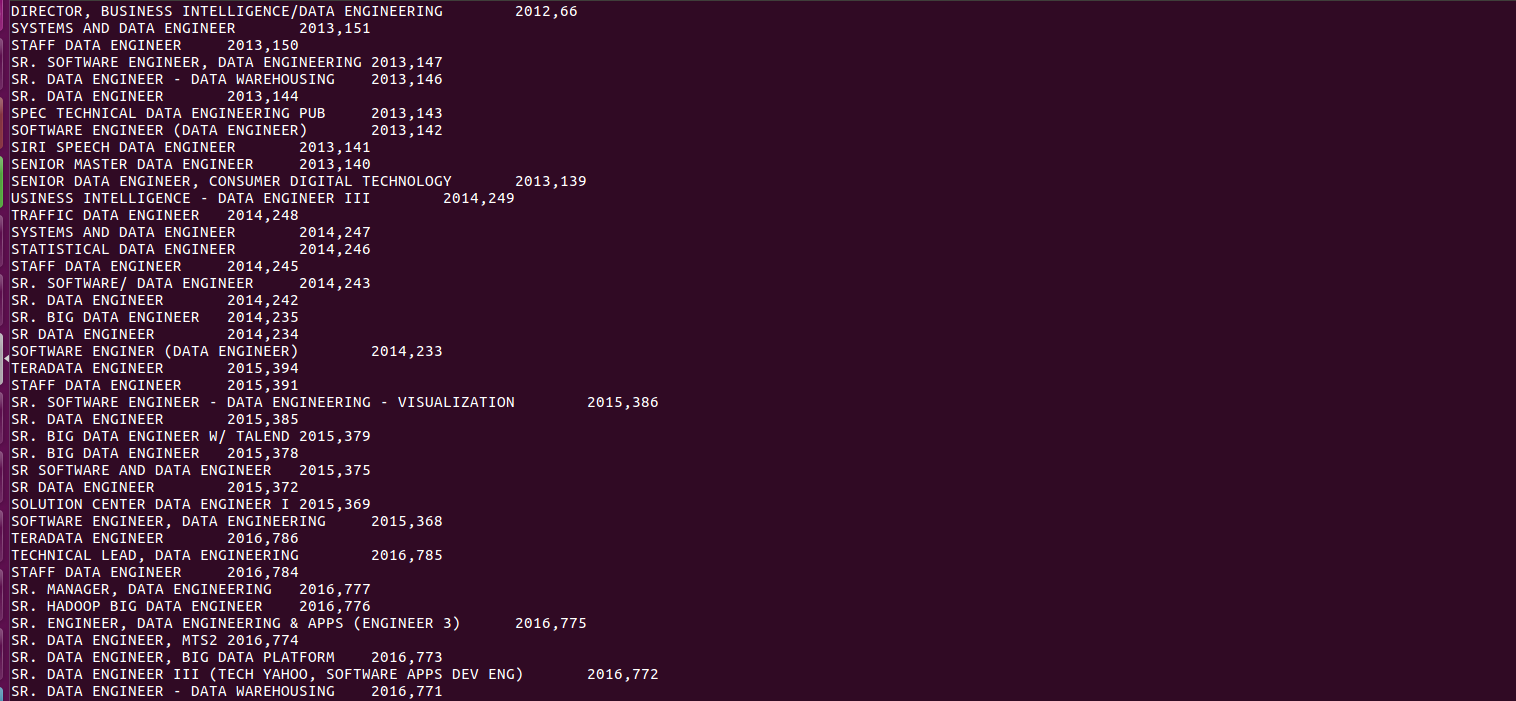
System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Output:**





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

**Source Code:**

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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.lib.input.\*;

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

import java.io.\*;

import java.util.TreeMap;

public class Query5b{

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

{

private LongWritable val = new LongWritable(1);

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

{

try{

if(key.get()>0)

{

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

if(str[1]!=null && str[1].equals("CERTIFIED") && str[4]!=null && str[4].contains("DATA ENGINEER"))

{

Text data = new Text (str[4]+"\t"+str[7]);

context.write(data, val);

}

}

}

catch(Exception e)

{

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

}

}

}

//Partitioner class

public static class CaderPartitioner extends

Partitioner < Text, LongWritable >

{

@Override

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

{

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

if(str[1].equals("2011"))

{

return 0 % numReduceTasks;

}

else if(str[1].equals("2012"))

{

return 1 % numReduceTasks ;

}

else if(str[1].equals("2013"))

{

return 2 % numReduceTasks ;

}

else if(str[1].equals("2014"))

{

return 3 % numReduceTasks;

}

else if(str[1].equals("2015"))

{

return 4 % numReduceTasks;

}

else if(str[1].equals("2016"))

{

return 5 % numReduceTasks ;

}

else

{

return 6;

}

}

}

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

{

private TreeMap<LongWritable, Text> job\_title = new TreeMap<LongWritable, Text>();

long count=0;

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

{

for(LongWritable val:values)

{

count+=val.get();

}

job\_title.put(new LongWritable(count),new Text(key+","+count));

if (job\_title.size()>10)

job\_title.remove(job\_title.firstKey());

}

protected void cleanup(Context context)throws IOException, InterruptedException

{

for (Text t : job\_title.descendingMap().values())

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, "Popular Jobs for each year ");

job.setJarByClass(Query5b.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(LongWritable.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

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

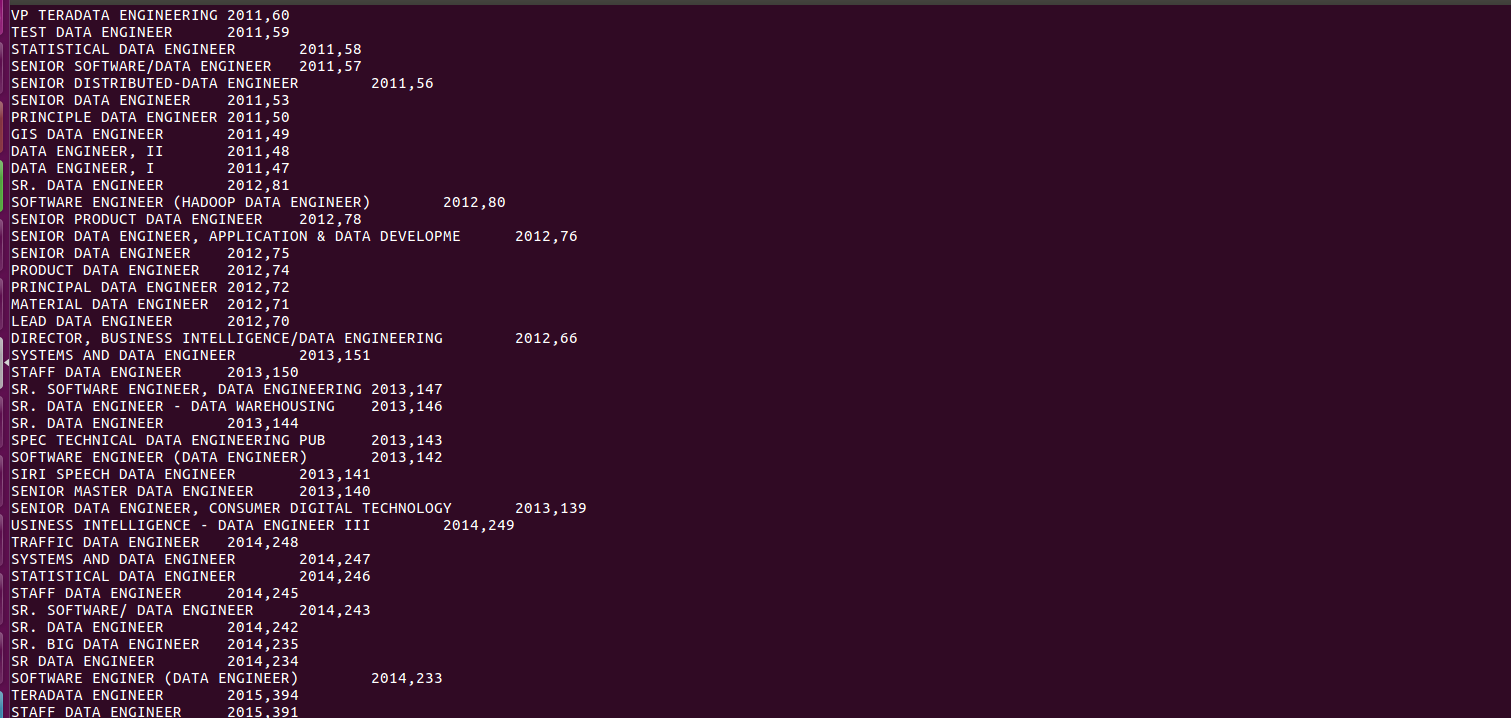
FileOutputFormat.setOutputPath(job, new Path(args[1]));

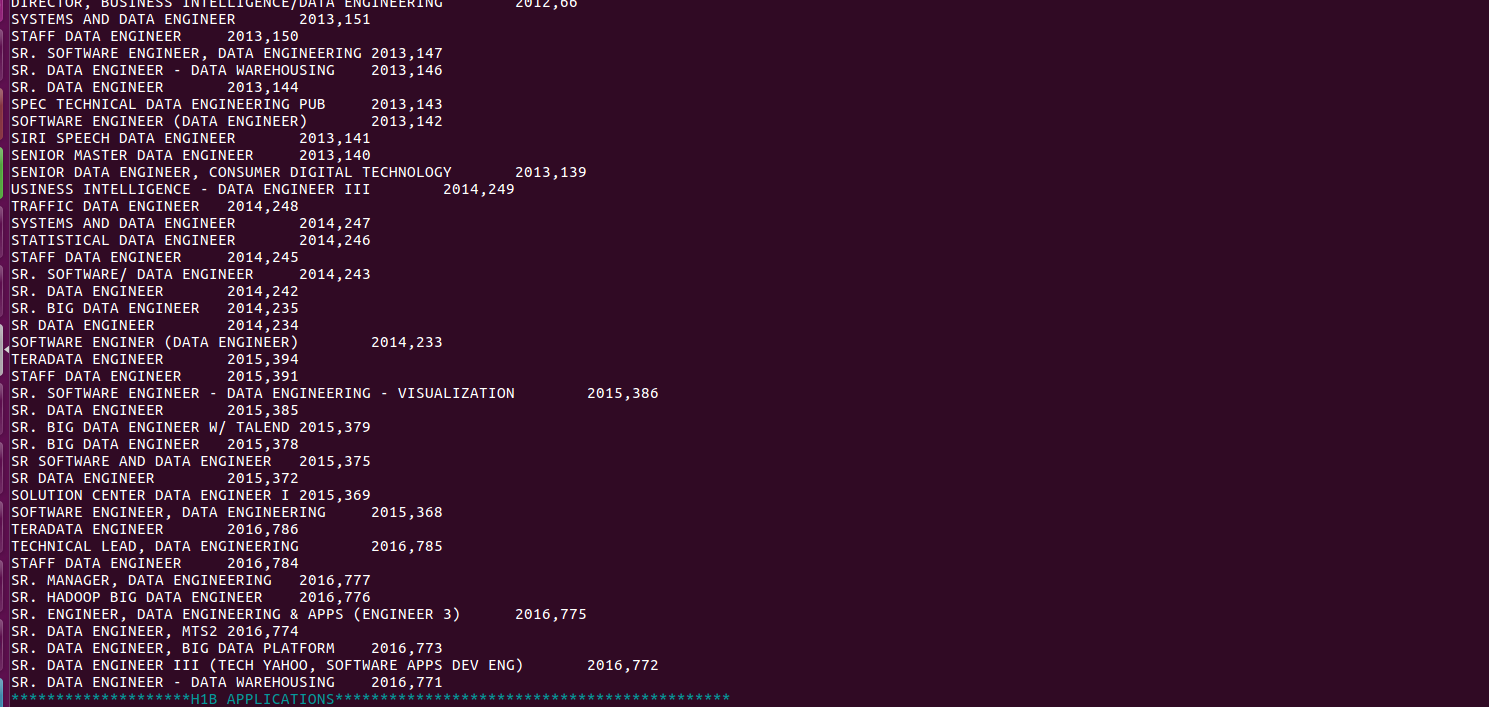
System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Output:**

****

****

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

**Source Code :**

--Query 6

h1b= load '/home/hduser/Downloads/h1bfinal' using PigStorage ( ) AS (s\_no:int, case\_status, employer\_name, soc\_name, job\_title,full\_time\_position, prevaling\_wage :long , year, worksite, longitude :double , latitude :double);

--groupbyyear= GROUP h1b by year ;

--countbyyear= FOREACH groupbyyear GENERATE group as year, COUNT (h1b) as headcount;

h1b\_1 = GROUP h1b by (case\_status,year);

h1b\_2 = FOREACH h1b\_1 GENERATE FLATTEN(group) as (case\_status,year) , COUNT (h1b) as headcount;

--dump h1b\_2;

bag2011 = FILTER h1b\_2 by year==2011;

bag1op= FOREACH bag2011 GENERATE case\_status, year, headcount , 100\* (double)headcount/358767;

--dump bag1op;

bag2012 = FILTER h1b\_2 by year==2012;

bag2op= FOREACH bag2012 GENERATE case\_status, year, headcount , 100\* (double)headcount/415607;

--dump bag2op;

bag2013 = FILTER h1b\_2 by year==2013;

bag3op= FOREACH bag2013 GENERATE case\_status, year, headcount , 100\* (double)headcount/442114;

--dump bag3op;

bag2014 = FILTER h1b\_2 by year==2014;

bag4op= FOREACH bag2014 GENERATE case\_status, year, headcount , 100\* (double)headcount/519427;

--dump bag4op;

bag2015 = FILTER h1b\_2 by year==2015;

bag5op= FOREACH bag2015 GENERATE case\_status, year, headcount , 100\* (double)headcount/618727;

--dump bag5op;

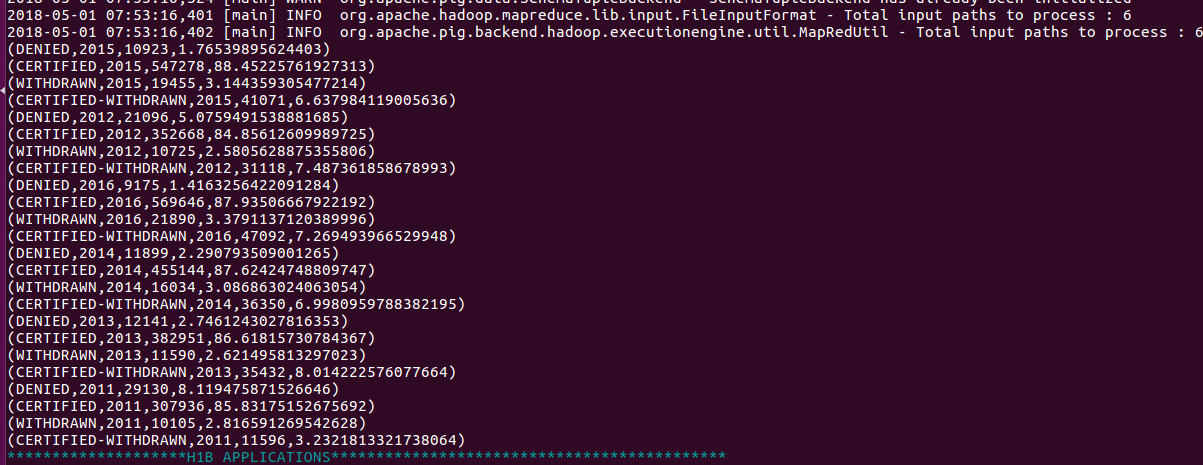
bag2016 = FILTER h1b\_2 by year==2016;

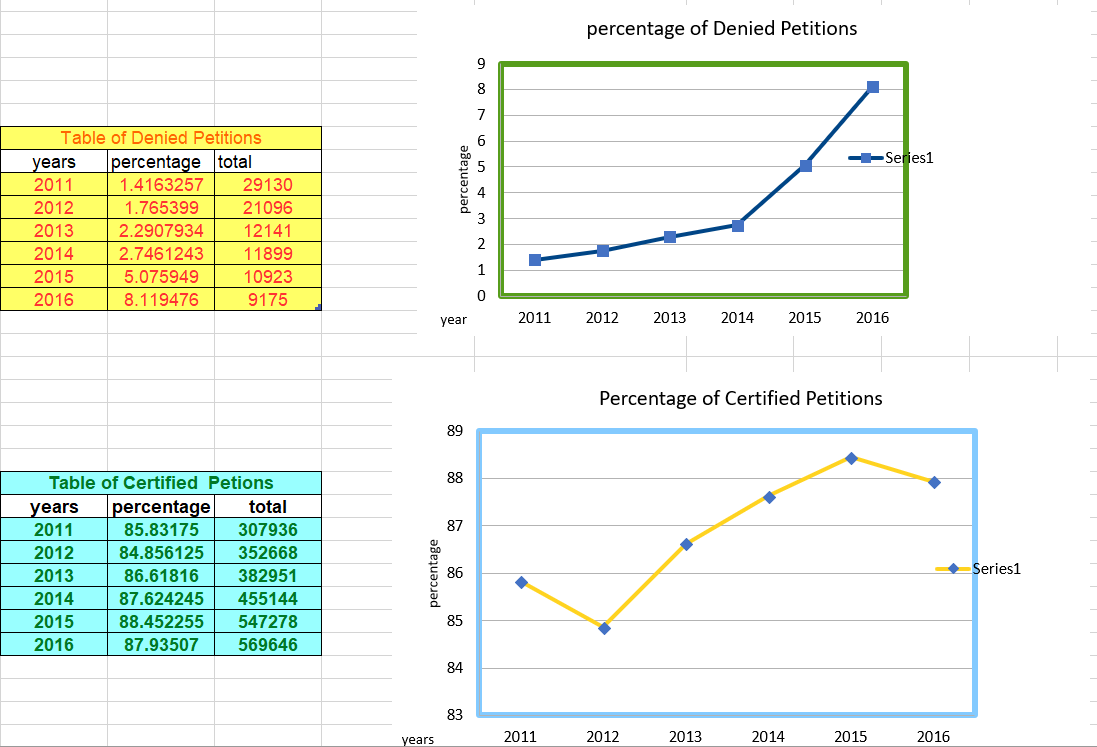
bag6op= FOREACH bag2016 GENERATE case\_status, year, headcount , 100\* (double)headcount/647803;

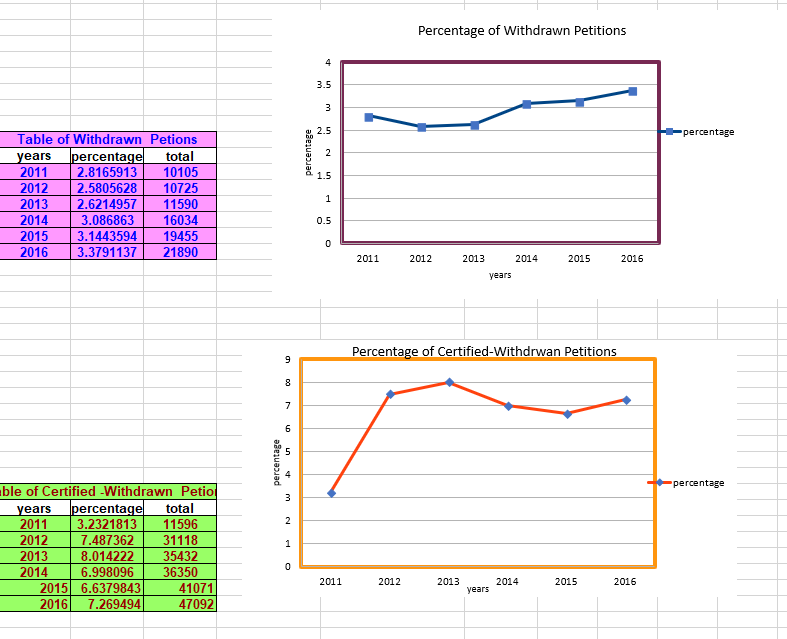
--dump bag6op;

finalop= UNION bag1op,bag2op,bag3op,bag4op,bag5op,bag6op;

dump finalop;

**Output:**

****

****

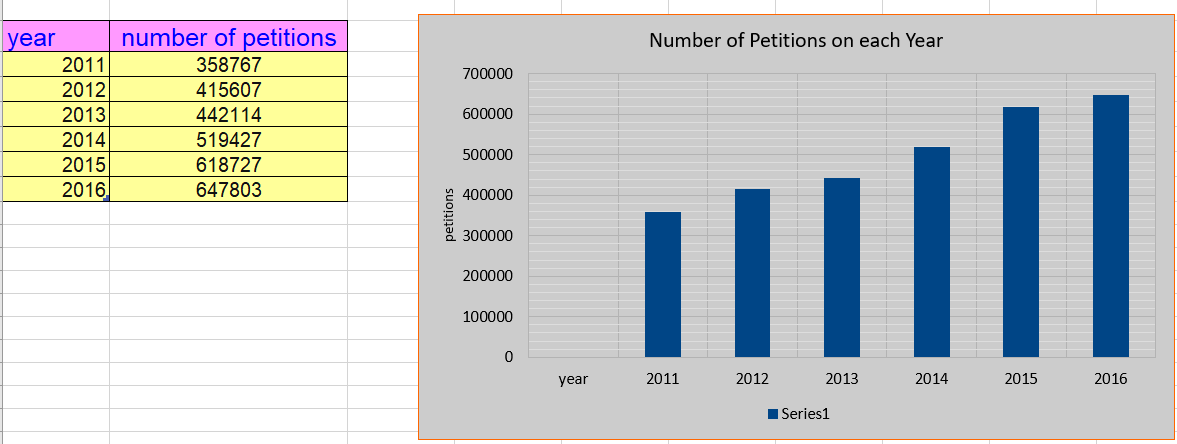
7) Create a bar graph to depict the number of applications for each year

**Source Code:**

Use h1b;

select year, count(\*) as headcount from h1b\_final group by year;

**Output:**

****

8) Find the average Prevailing wage for each job for each year (take part time and full time separate) . Arrange Output in descending order.(Hive)

**Source Code:**

select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_final where case\_status="CERTIFIED" and year='2011' and full\_time\_position="Y" group by job\_title,full\_time\_position,year order by average desc;

select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_final where case\_status="CERTIFIED" and year='2011' and full\_time\_position="N" group by job\_title,full\_time\_position,year order by average desc;

select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_final where case\_status="CERTIFIED-WITHDRAWN" and year='2011' and full\_time\_position="Y" group by job\_title,full\_time\_position,year order by average desc;

select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_final where case\_status="CERTIFIED-WITHDRAWN" and year='2011' and full\_time\_position="N" group by job\_title,full\_time\_position,year order by average desc;

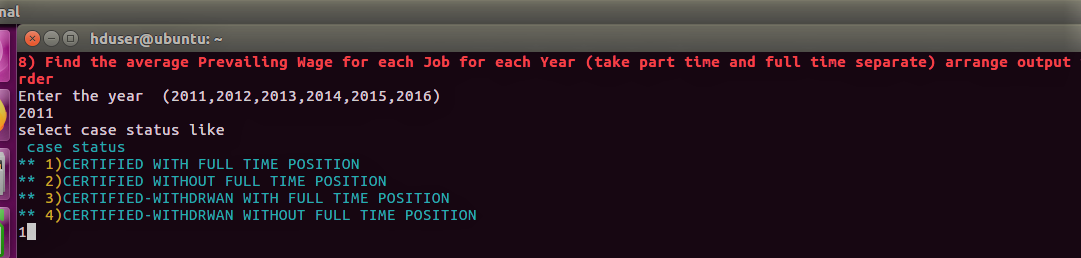
select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_final where case\_status="CERTIFIED" and year='2012' and full\_time\_position="Y" group by job\_title,full\_time\_position,year order by average desc;

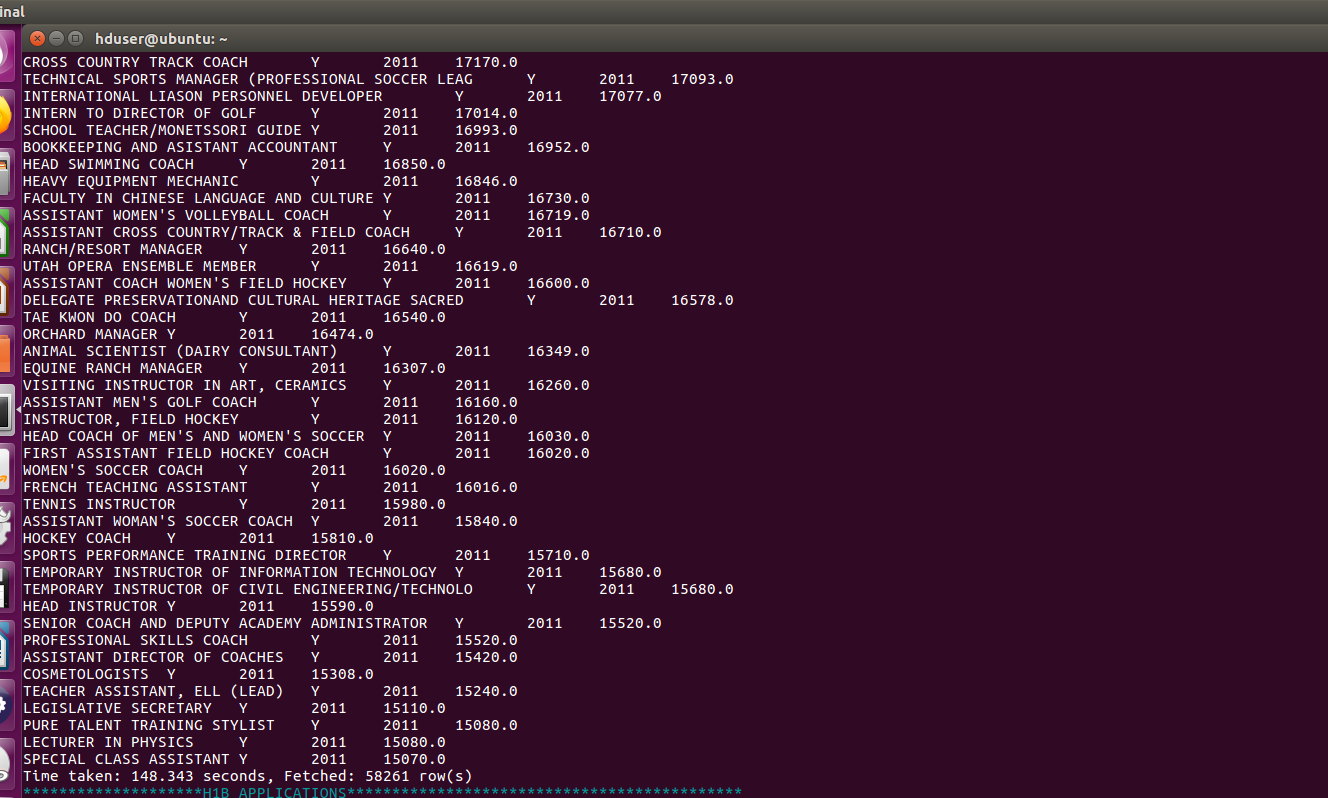
select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_final where case\_status="CERTIFIED" and year='2012' and full\_time\_position="N" group by job\_title,full\_time\_position,year order by average desc;

select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_final where case\_status="CERTIFIED-WITHDRAWN" and year='2012' and full\_time\_position="Y" group by job\_title,full\_time\_position,year order by average desc;

select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_final where case\_status="CERTIFIED-WITHDRAWN" and year='2012' and full\_time\_position="N" group by job\_title,full\_time\_position,year order by average desc;

**Output:**





9) Which are the employers along with number of petitions who have success rate more than 70% in petitions.(Pig)

**Source Code:**

--Query 9

h1b= load '/home/hduser/Downloads/h1bfinal' using PigStorage ( ) AS (s\_no:int, case\_status, employer\_name, soc\_name, job\_title,full\_time\_position, prevaling\_wage :long , year, worksite, longitude :double , latitude :double);

groupbyemp = GROUP h1b by employer\_name;

countbyemp= foreach groupbyemp GENERATE group as employer\_name, COUNT(h1b) as headcount;

totalpetition = FILTER countbyemp BY headcount>=1000;

filterbycase = FILTER h1b by case\_status=='CERTIFIED';

groupbycase = GROUP filterbycase by employer\_name;

countbycertied = foreach groupbycase GENERATE group as employer\_name, COUNT(filterbycase) as certifiedcount;

filterbycase2 = FILTER h1b by case\_status=='CERTIFIED-WITHDRAWN';

groupbycase2 = GROUP filterbycase2 by employer\_name;

countbycase2= foreach groupbycase2 generate group as employer\_name ,COUNT(filterbycase2) as certifiedwithdrawncount;

join1 = join totalpetition by employer\_name , countbycertied by employer\_name , countbycase2 by employer\_name ;

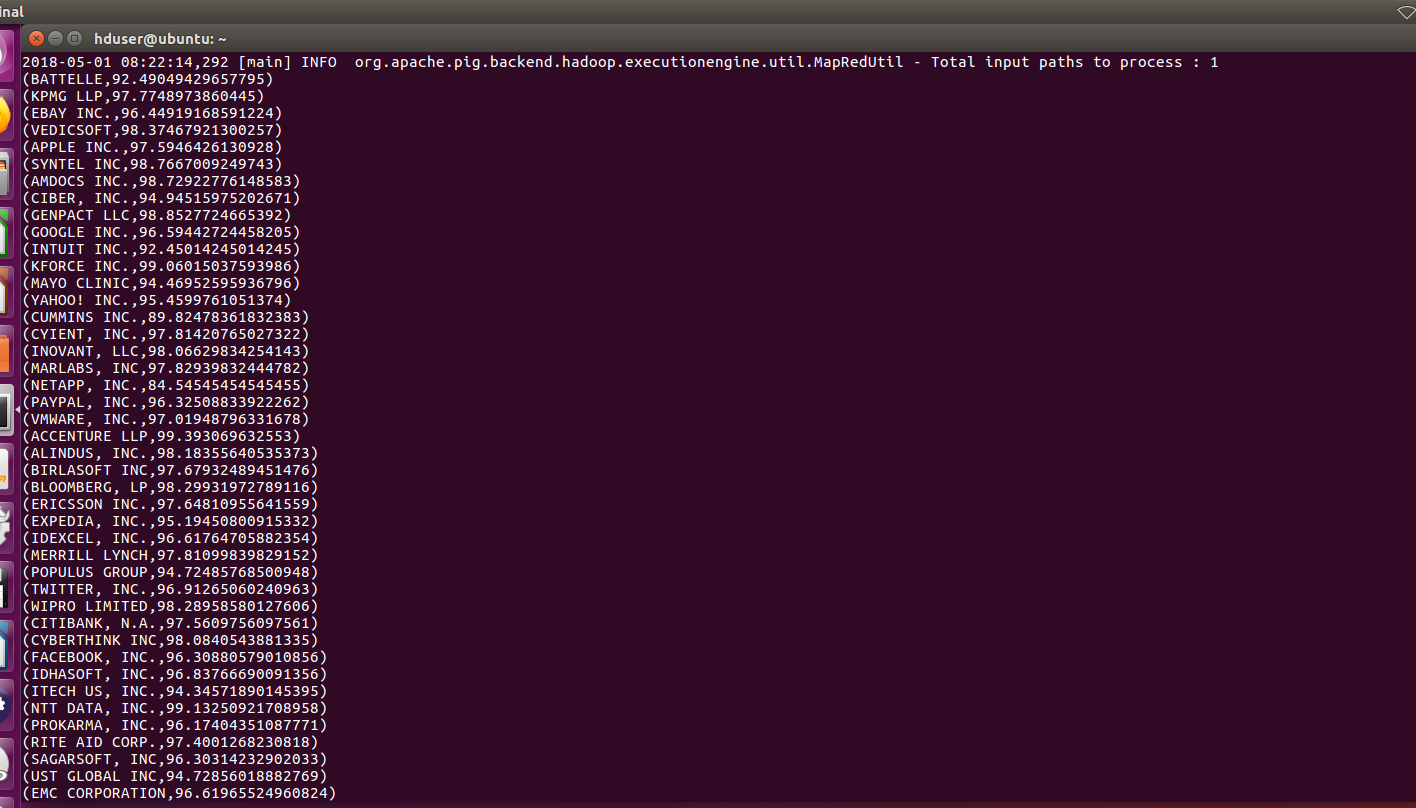
joinfinal= foreach join1 generate $0, $1, $3, $5;

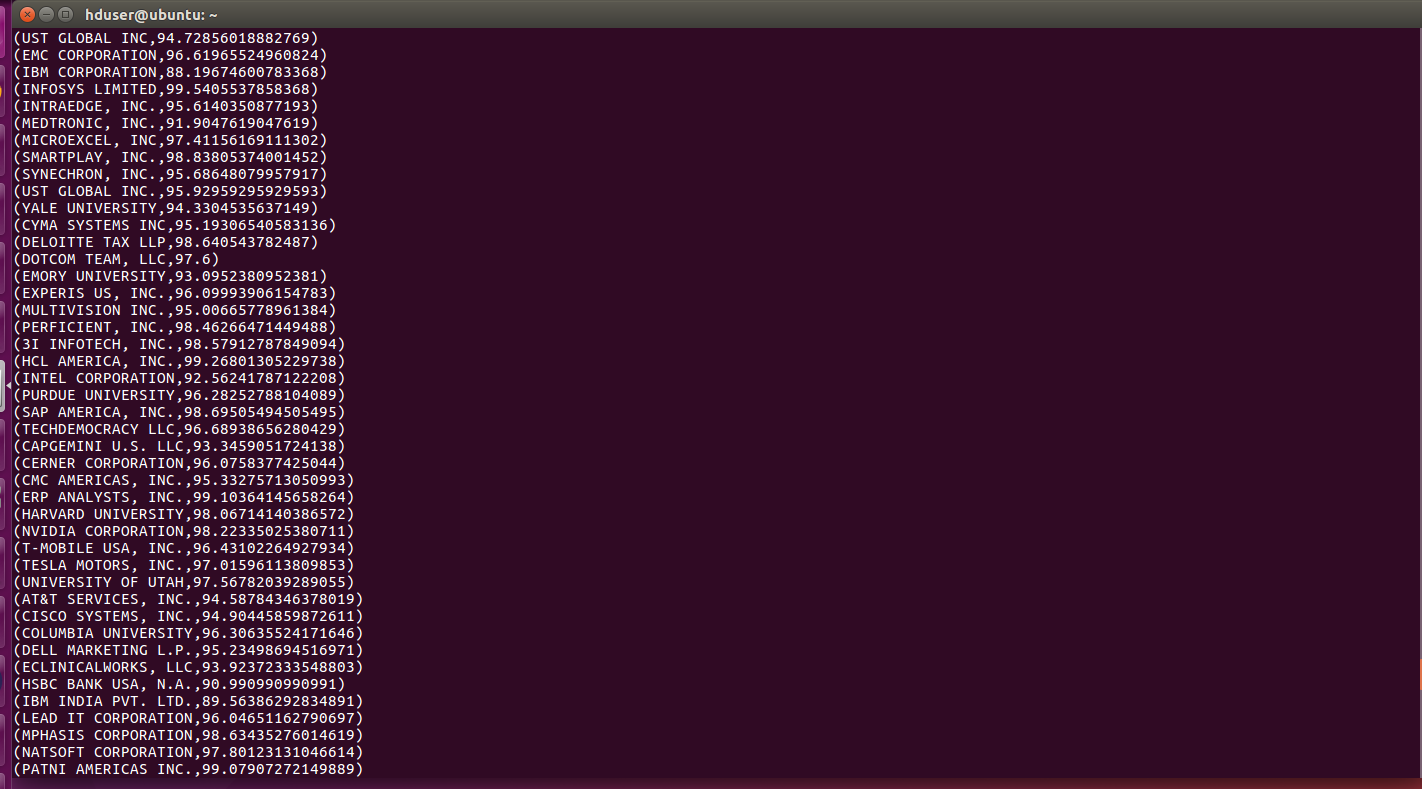
Successrate= foreach joinfinal generate $0, (double) ($2+$3)/$1 \* 100 as successrate;

Successrateop= FILTER Successrate by successrate>70;

dump Successrateop;

**Output:**

****

****

10) Which are the job positions along with number of petitions who have success rate more than 70% in petitions.(Pig)

**Source Code:**

--Query 10

h1b= load '/user/hive/warehouse/h1b.db/h1b\_final' using PigStorage ( ) AS (s\_no:int, case\_status, employer\_name, soc\_name, job\_title,full\_time\_position, prevaling\_wage :long , year, worksite, longitude :double , latitude :double);

groupbyjob = GROUP h1b by job\_title;

countbyjob= foreach groupbyjob GENERATE group as job\_title, COUNT(h1b) as headcount;

totalpetition = FILTER countbyjob BY headcount>=1000;

filterbycase = FILTER h1b by case\_status=='CERTIFIED';

groupbycase = GROUP filterbycase by job\_title;

countbycertied = foreach groupbycase GENERATE group as job\_title, COUNT(filterbycase) as certifiedcount;

filterbycase2 = FILTER h1b by case\_status=='CERTIFIED-WITHDRAWN';

groupbycase2 = GROUP filterbycase2 by job\_title;

countbycase2= foreach groupbycase2 generate group as job\_title ,COUNT(filterbycase2) as certifiedwithdrawncount;

join1 = join totalpetition by job\_title , countbycertied by job\_title , countbycase2 by job\_title;

joinfinal= foreach join1 generate $0, $1, $3, $5;

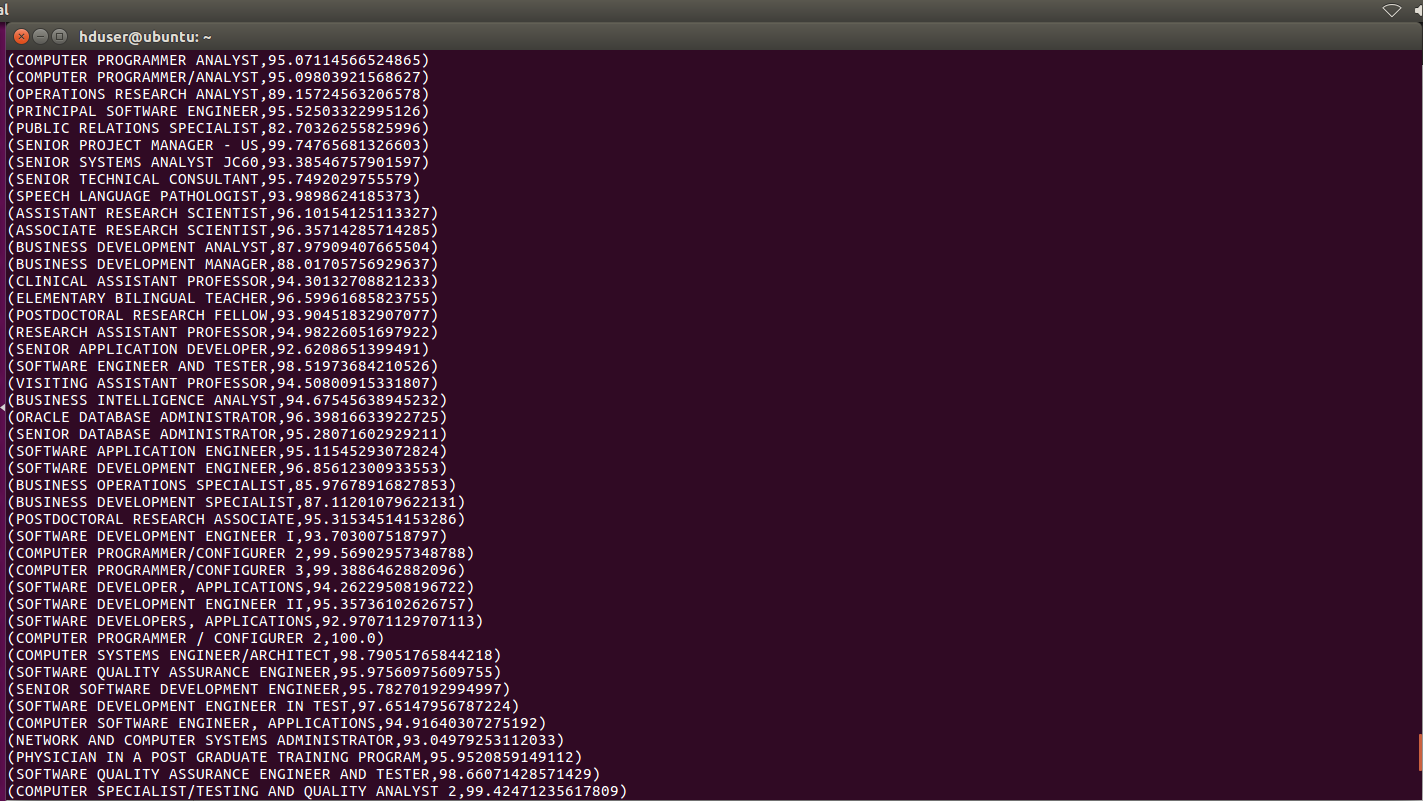
Successrate= foreach joinfinal generate $0, (double) ($2+$3)/$1 \* 100 as successrate;

Successrateop= FILTER Successrate by successrate>70;

dump Successrateop;

store Successrateop into '/home/hduser/h1bproject/query10op;

**Output:**

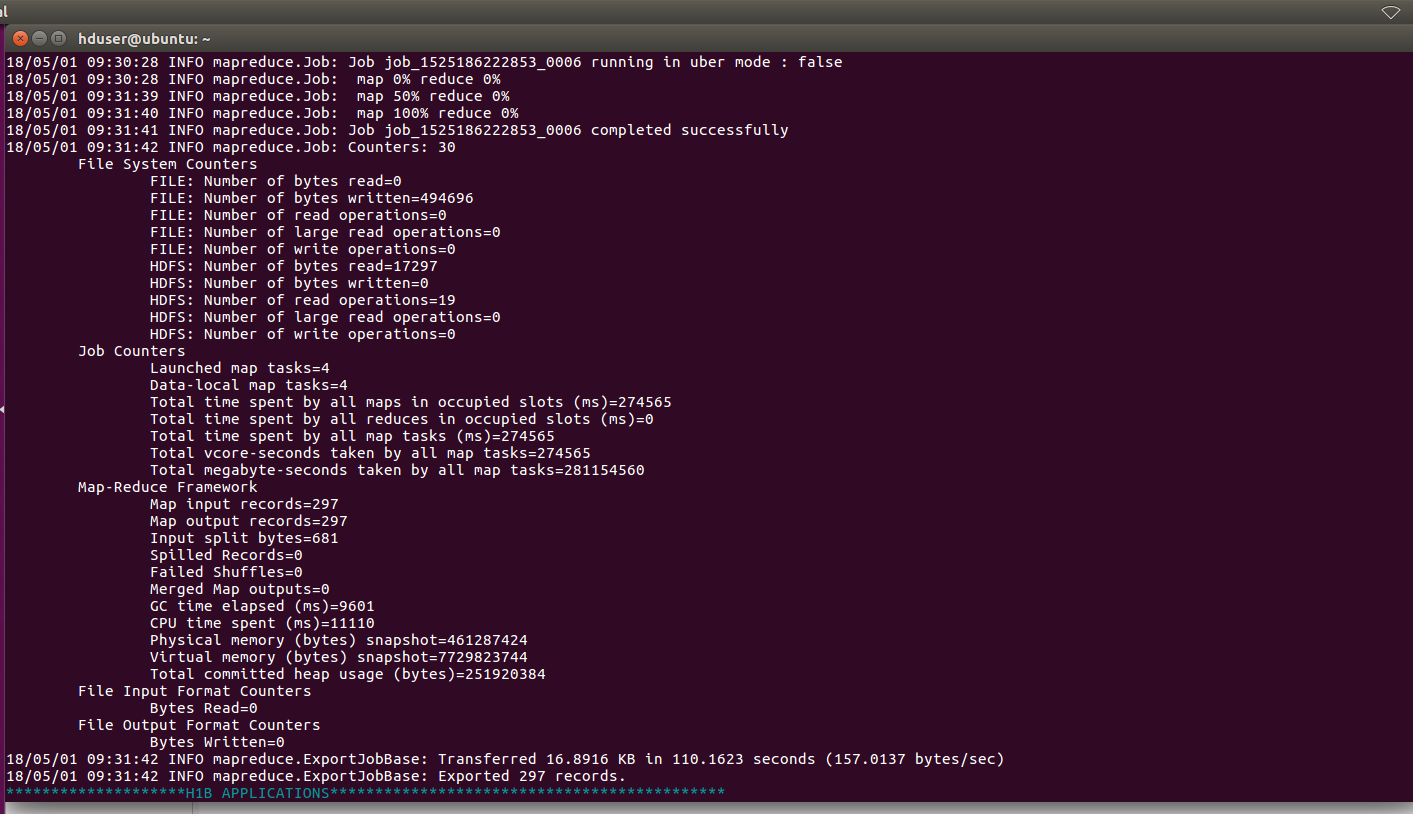
****

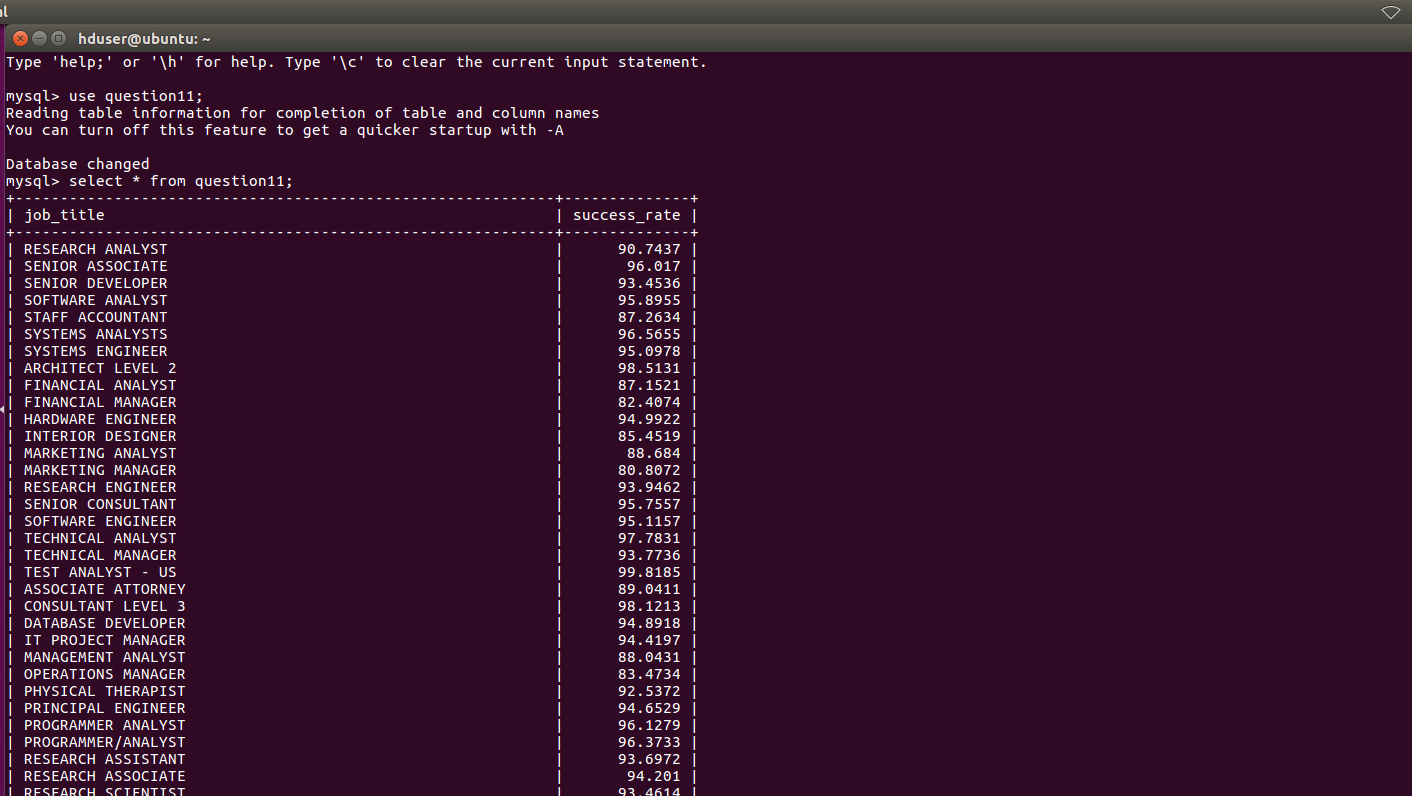
11) Export the results of question 10 to Mysql.

Using Sqoop.

sqoop export – connect jdbc:mysql://localhost/question11 –username hduser –password ‘hadoop’ – table question11 – update-mode allowinsert - - update-key job\_title --export – dir /home/hduser/h1bproject/query10op/p\* -- input-fields -terminated-by ‘ \t ’;

**Output:**

****

****