**H1B Data Analysis:**

**Let’s create a table to load the h1b applicant’s data as shown below.**

CREATE TABLE h1b\_applications(s\_no int,case\_status string, employer\_name string, soc\_name string, job\_title string, full\_time\_position string,prevailing\_wage int,year string, worksite string, longitute double, latitute double )

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

"separatorChar" = ",",

"quoteChar" = "\""

) STORED AS TEXTFILE;

**We have created a table with name h1b\_applications. Let’s load the data into the table.**

load data local inpath '/home/manmath/Documents/datasets/h1b.csv' into table h1b\_applications;

**We have successfully loaded the data into the table.**

**We fired a query on the table to check whether the data is loaded successfully or not.**

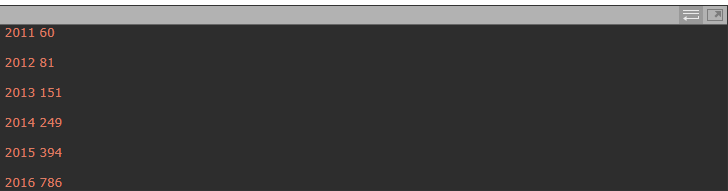
select \* from h1b\_applications limit 10;

**Q.1.a)Is the number of petitions with Data Engineer job title increasing over time?(solved by Hive)**

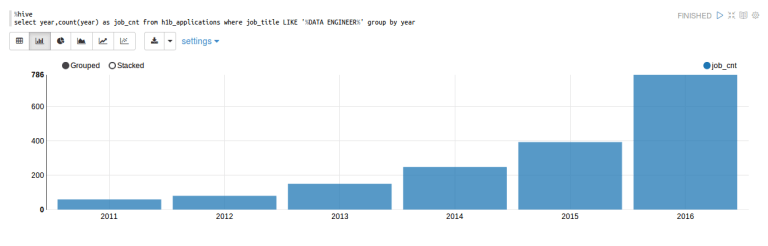
**Here we will find the number of applicants applied for the various kind of data engineer positions**.

select year,count(year)from h1b\_applications where job\_title LIKE'%DATA ENGINEER%'group by year;

Here is the result of this query:



In the below screenshot you can see the visualization graph.



**Solved by MapReduce:**

1. **Java Code:**

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

public class Question1a {

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

{

Configuration conf= new Configuration();

Job job= Job.getInstance(conf,"Question 1a");

job.setJarByClass(Question1a.class);

job.setMapperClass(Question1aMapper.class);

job.setReducerClass(Question1aReducer.class);

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)?1:0);

}

}

1. **Mapper:**

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

public class Question1a {

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

{

Configuration conf= new Configuration();

Job job= Job.getInstance(conf,"Question 1a");

job.setJarByClass(Question1a.class);

job.setMapperClass(Question1aMapper.class);

job.setReducerClass(Question1aReducer.class);

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)?1:0);

}

}

1. **Reducer:**

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class Question1aReducer extends Reducer <Text,LongWritable,Text,LongWritable>

{ LongWritable SUM=new LongWritable(0); int i=0;

String[] years={"2011","2012","2013","2014","2015","2016"};

long [] arr=new long[6];

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

{

long sum=0;

for(LongWritable val:values)

sum+=val.get();

arr[i++]=sum;

}

public void cleanup(Context context) throws IOException, InterruptedException

{

for (int i=0;i<6;i++)

if (i==0)

context.write(new Text(years[i]), new LongWritable(0));

else

context.write(new Text(years[i]),new LongWritable((arr[i]-arr[i-1])\*100/arr[i-1]));

}

}

**Q.1 (b) Find top 5 job titles who are having highest average growth in applications. (solved by Pig)**

REGISTER '/home/manmath/BigData/external\_jars/piggybank-0.13.0.jar'; --Register external jar 'Piggy Bank.jar'

DEFINE CSVExcelStorage org.apache.pig.piggybank.storage.CSVExcelStorage; -- within the jar define a function CSVExcelStorage()

data = LOAD '/home/ayush/DatasetsandCodes/h1b\_applications.csv' USING CSVExcelStorage() as

(s\_no:int,

case\_status:chararray,

employer\_name:chararray,

soc\_name:chararray,

job\_title:chararray,

full\_time\_position:chararray,

prevailing\_wage:int,

year:chararray,

worksite:chararray,

longitute:double,

latitute:double); --Load data

noheader= filter data by $0>=1; --Remove header

cleansed= filter noheader by $7 matches '2011'; --filtering dataset by year

a= group cleansed by $4; --grouping by job

step\_a= foreach a generate group,COUNT($1); --generate year,job,count

leansed= filter noheader by $7 matches '2012'; --filtering dataset by year

a= group cleansed by $4; --grouping by job

step\_b= foreach a generate group,COUNT($1); --generate year,job,count

cleansed= filter noheader by $7 matches '2013'; --filtering dataset by year

a= group cleansed by $4; --grouping by job

step\_c= foreach a generate group,COUNT($1); --generate year,job,count

cleansed= filter noheader by $7 matches '2014'; --filtering dataset by year

a= group cleansed by $4; --grouping by job

step\_d= foreach a generate group,COUNT($1); --generate year,job,count

cleansed= filter noheader by $7 matches '2015'; --filtering dataset by year

a= group cleansed by $4; --grouping by job

step\_e= foreach a generate group,COUNT($1); --generate year,job,count

cleansed= filter noheader by $7 matches '2016'; --filtering dataset by year

a= group cleansed by $4; --grouping by job

step\_f= foreach a generate group,COUNT($1); --generate year,job,count

joined= join step\_a by $0,step\_b by $0,step\_c by $0,step\_d by $0,step\_e by $0,step\_f by $0;

yearwiseapplications= foreach joined generate $0,$1,$3,$5,$7,$9,$11;

--generate progressive growth

progressivegrowth= foreach yearwiseapplications generate $0,

(float)($6-$5)\*100/$5,(float)($5-$4)\*100/$4,

(float)($4-$3)\*100/$3,(float)($3-$2)\*100/$2,

(float)($2-$1)\*100/$1;

--average progressive growth

avgprogressivegrowth= foreach progressivegrowth generate $0,($1+$2+$3+$4+$5)/5;

---ordered progressive growth

orderedavggrowth= order avgprogressivegrowth by $1 desc;

--display top5 only

answer = limit orderedavggrowth 5;

dump answer;

**Q.2 a) Which part of the US has the most Data Engineer jobs for each year?(solved by Pig)**

REGISTER '/home/Manmath/BigData/external\_jars/piggybank-0.13.0.jar'; --Register external jar 'Piggy Bank.jar'

DEFINE CSVExcelStorage org.apache.pig.piggybank.storage.CSVExcelStorage; -- within the jar define a function CSVExcelStorage()

data = LOAD '/home/ayush/DatasetsandCodes/h1b\_applications.csv' USING CSVExcelStorage() as

(s\_no:int,

case\_status:chararray,

employer\_name:chararray,

soc\_name:chararray,

job\_title:chararray,

full\_time\_position:chararray,

prevailing\_wage:int,

year:chararray,

worksite:chararray,

longitute:double,

latitute:double); --Load data

noheader= filter data by $0>=1; --Remove header

cleansed= filter noheader by $4 matches '.\*DATA ENGINEER.\*' and $7 matches '2011'; --filtering dataset which contains 'DATA ENGINEER' .

a= group cleansed by $8; --grouping by worksite

step\_a= foreach a generate '2011',group,COUNT($1); --generate year,worksite,count

step\_b= order step\_a by $2 desc;

answer\_2011= limit step\_b 1;

cleansed= filter noheader by $4 matches '.\*DATA ENGINEER.\*' and $7 matches '2012'; --filtering dataset which contains 'DATA ENGINEER' .

a= group cleansed by $8; --grouping by worksite

step\_a= foreach a generate '2012',group,COUNT($1); --generate year,worksite,count

step\_b= order step\_a by $2 desc;

answer\_2012= limit step\_b 1;

cleansed= filter noheader by $4 matches '.\*DATA ENGINEER.\*' and $7 matches '2013';

a= group cleansed by $8;

step\_a= foreach a generate '2013',group,COUNT($1);

step\_b= order step\_a by $2 desc;

answer\_2013= limit step\_b 1;

cleansed= filter noheader by $4 matches '.\*DATA ENGINEER.\*' and $7 matches '2014';

a= group cleansed by $8;

step\_a= foreach a generate '2014',group,COUNT($1);

step\_b= order step\_a by $2 desc;

answer\_2014= limit step\_b 1;

cleansed= filter noheader by $4 matches '.\*DATA ENGINEER.\*' and $7 matches '2015';

a= group cleansed by $8;

step\_a= foreach a generate '2015',group,COUNT($1);

step\_b= order step\_a by $2 desc;

answer\_2015= limit step\_b 1;

cleansed= filter noheader by $4 matches '.\*DATA ENGINEER.\*' and $7 matches '2016';

a= group cleansed by $8;

step\_a= foreach a generate '2016',group,COUNT($1);

step\_b= order step\_a by $2 desc;

answer\_2016= limit step\_b 1;

dump answer\_2011; --display answers for every year

dump answer\_2012;

dump answer\_2013;

dump answer\_2014;

dump answer\_2015;

dump answer\_2016;

**Solved by Map Reduce:**

1. **Java Code :**

import java.io.IOException;

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.conf.Configuration;

import org.apache.hadoop.fs.Path;

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

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

public class Question2a

{

public static void main(String args[]) throws IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Top 5 Data Engineer in a worksite");

job.setJarByClass(Question2a.class);

job.setMapperClass(Question2aMapper.class);

job.setPartitionerClass(Question2aPartitioner.class);

job.setReducerClass(Question2aReducer.class);

job.setNumReduceTasks(7);

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);

}

}

1. **Mapper:**

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class Question2aMapper extends Mapper < LongWritable, Text, Text, LongWritable > {

LongWritable one = new LongWritable(1);

public void map(LongWritable key, Text values, Context context) throws IOException,

InterruptedException {

if (key.get() > 0)

{

String[] token = values.toString().split(",(?=([^\"]\*\"[^\"]\*\")\*[^\"]\*$)");

if (token[4] != null && token[4].contains("DATA ENGINEER") && token[8] != null && !token[8].equals("NA")) {

Text answer = new Text(token[8].replaceAll("\"", "") + "\t" + token[7]);

context.write(answer, one);

}

}

}

}

1. **Partitioner:**

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Partitioner;

public class Question2aPartitioner 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;

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

return 1;

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

return 2;

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

return 3;

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

return 4;

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

return 5;

else

return 6;

}

}

1. **Reducer:**

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class Question2aReducer extends Reducer<Text,LongWritable,NullWritable,Text>

{

private TreeMap<LongWritable, Text> Top5DataEngineer = new TreeMap<LongWritable, Text>();

long sum=0;

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

{

sum=0;

for(LongWritable val:values)

{

sum+=val.get();

}

Top5DataEngineer.put(new LongWritable(sum),new Text(key+","+sum));

if (Top5DataEngineer.size()>5)

Top5DataEngineer.remove(Top5DataEngineer.firstKey());

}

protected void cleanup(Context context)throws IOException, InterruptedException

{

for (Text t : Top5DataEngineer.descendingMap().values())

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

}

}

**Q.2b) find top 5 locations in the US that have got certified visa for each year. (solved by Hive)**

select worksite,count(case\_status) as t,year from h1b\_applications where year ='2011' and case\_status='CERTIFIED' group by worksite,year order by t desc limit 5;

select worksite,count(case\_status) as t,year from h1b\_applications where year ='2012' and case\_status='CERTIFIED' group by worksite,year order by t desc limit 5;

select worksite,count(case\_status) as t,year from h1b\_applications where year ='2013' and case\_status='CERTIFIED' group by worksite,year order by t desc limit 5;

select worksite,count(case\_status) as t,year from h1b\_applications where year ='2014' and case\_status='CERTIFIED' group by worksite,year order by t desc limit 5;

select worksite,count(case\_status) as t,year from h1b\_applications where year ='2015' and case\_status='CERTIFIED' group by worksite,year order by t desc limit 5;

select worksite,count(case\_status) as t,year from h1b\_applications where year ='2016' and case\_status='CERTIFIED' group by worksite,year order by t desc limit 5;

**Q.3 Which industry has the most number of Data Scientist positions?(solved by Hive)**

select soc\_name,count(soc\_name)ascnt from h1b\_applications where job\_title LIKE'%DATA SCIENTIST%'group by soc\_name order by cnt desc;

**Statisticians industry need more data scientists.**

**Solved by MapReduce:**

1. **Java Code:**

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class Question2aReducer extends Reducer<Text,LongWritable,NullWritable,Text>

{

private TreeMap<LongWritable, Text> Top5DataEngineer = new TreeMap<LongWritable, Text>();

long sum=0;

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

{

sum=0;

for(LongWritable val:values)

{

sum+=val.get();

}

Top5DataEngineer.put(new LongWritable(sum),new Text(key+","+sum));

if (Top5DataEngineer.size()>5)

Top5DataEngineer.remove(Top5DataEngineer.firstKey());

}

protected void cleanup(Context context)throws IOException, InterruptedException

{

for (Text t : Top5DataEngineer.descendingMap().values())

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

}

1. **Mapper:**

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class Question3Mapper extends Mapper < LongWritable, Text, Text, LongWritable > {

LongWritable one = new LongWritable(1);

public void map(LongWritable key, Text values, Context context) throws IOException,

InterruptedException {

if (key.get() > 0) {

String[] token = values.toString().split(",(?=([^\"]\*\"[^\"]\*\")\*[^\"]\*$)");

if (token[4].contains("DATA SCIENTIST")) {

Text answer = new Text(token[3].replaceAll("\"", ""));

context.write(answer, one);

}

}

}

}

1. **Reducer:**

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class Question3Reducer extends Reducer < Text, LongWritable, NullWritable, Text > {

private TreeMap < LongWritable,

Text > DataScientistJobs = new TreeMap < LongWritable,

Text > ();

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

InterruptedException {

long sum = 0;

for (LongWritable val: values)

sum += val.get();

DataScientistJobs.put(new LongWritable(sum), new Text(key.toString().replaceAll("\"", "") + "," + sum));

if (DataScientistJobs.size() > 5)

DataScientistJobs.remove(DataScientistJobs.firstKey());

}

protected void cleanup(Context context) throws IOException,

InterruptedException {

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

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

}

}

**Q.4 . Which employers file the most petitions each year?(solved by Hive)**

select employer\_name,year,count(year)ascnt from h1b\_applications group by year,employer\_name order by cnt;

**Infosys ltd has received more number of petitions for 2013,2014,2015,2016 years.**

**Solved by MapReduce:**

1. **Java Code:**

import java.io.IOException;

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.conf.Configuration;

import org.apache.hadoop.fs.Path;

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

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

public class Question4 {

public static void main(String args[]) throws IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Top 5 Employers");

job.setJarByClass(Question4.class);

job.setMapperClass(Question4Mapper.class);

job.setPartitionerClass(Question4Partitioner.class);

job.setReducerClass(Question4Reducer.class);

job.setNumReduceTasks(7);

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);

}

}

1. **Mapper:**

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class Question4Mapper extends Mapper < LongWritable, Text, Text, LongWritable > {

LongWritable one = new LongWritable(1);

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

InterruptedException {

if (key.get() > 0)

{

String[] token = value.toString().split(",(?=([^\"]\*\"[^\"]\*\")\*[^\"]\*$)");

if (!token[1].equals("NA") && token[1] != null && !token[2].equals("NA") && token[2] != null && !token[7].equals("NA") && token[7] != null) {

Text answer = new Text(token[2].replaceAll("\"", "") + "\t" + token[7]);

context.write(answer, one);

}

}

}

}

1. **Partitioner:**

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Partitioner;

public class Question4Partitioner 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;

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

return 1;

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

return 2;

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

return 3;

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

return 4;

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

return 5;

else

return 6;

}

}

1. **Reducer:**

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class Question4Reducer extends Reducer < Text, LongWritable, NullWritable, Text > {

private TreeMap < LongWritable,

Text > Top5Employers = new TreeMap < LongWritable,

Text > ();

long sum = 0;

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

InterruptedException {

sum = 0;

for (LongWritable val: values) {

sum += val.get();

}

Top5Employers.put(new LongWritable(sum), new Text(key + "," + sum));

if (Top5Employers.size() > 5)

Top5Employers.remove(Top5Employers.firstKey());

}

protected void cleanup(Context context) throws IOException,

InterruptedException {

for (Text t: Top5Employers.descendingMap().values())

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

}

}

**Q.5. Find the most popular top 10job positions for H1B visa applications for each year? (solved by Hive)**

select job\_title,year,count(case\_status ) as temp from h1b\_applications where year =2011group by job\_title,year order by temp desclimit10;

select job\_title,year,count(case\_status ) as temp from h1b\_applications where year =2012group by job\_title,year order by temp desclimit10;

select job\_title,year,count(case\_status ) as temp from h1b\_applications where year =2013group by job\_title,year order by temp desclimit10;

select job\_title,year,count(case\_status ) as temp from h1b\_applications where year =2014group by job\_title,year order by temp desclimit10;

select job\_title,year,count(case\_status ) as temp from h1b\_applications where year =2015group by job\_title,year order by temp desclimit10;

select job\_title,year,count(case\_status ) as temp from h1b\_applications where year =2016group by job\_title,year order by temp desclimit10;

**Q.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.(solved by Pig)**

REGISTER '/home/manmath/BigData/external\_jars/piggybank-0.13.0.jar'; --Register external jar 'Piggy Bank.jar'

DEFINE CSVExcelStorage org.apache.pig.piggybank.storage.CSVExcelStorage; -- within the jar define a function CSVExcelStorage()

data = LOAD '/home/ayush/DatasetsandCodes/h1b\_applications.csv' USING CSVExcelStorage() as

(s\_no:int,

case\_status:chararray,

employer\_name:chararray,

soc\_name:chararray,

job\_title:chararray,

full\_time\_position:chararray,

prevailing\_wage:int,

year:chararray,

worksite:chararray,

longitute:double,

latitute:double); --load data

noheader= filter data by $0>=1; --remove header

cleansed= filter noheader by $1 is not null and $1!='NA';

temp= group cleansed by $7;

total= foreach temp generate group,COUNT(cleansed.$1);

--describe total;dump total;

**Q.7. Create a bar graph to depict the number of applications for each year?(solved by Hive)**

select year,count(\*) from h1b\_applications where year !=null and year!='NA' group by year order by year;

**8) Find the average Prevailing Wage for each job for each year (take part time and full time separate). Arrange output in descending order.(solved by Hive)**

select job\_title,full\_time\_position,year,avg(prevailing\_wage) as average from h1b\_applications where full\_time\_position ='Y'and year='2011'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\_applications where full\_time\_position ='N'and year='2011'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\_applications where full\_time\_position ='Y'and year='2012'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\_applications where full\_time\_position ='N'and year='2012'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\_applications where full\_time\_position ='Y'and year='2013'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\_applications where full\_time\_position ='N'and year='2013'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\_applications where full\_time\_position ='Y'and year='2014'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\_applications where full\_time\_position ='N'and year='2014'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\_applications where full\_time\_position ='Y'and year='2015'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\_applications where full\_time\_position ='N'and year='2015'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\_applications where full\_time\_position ='Y'and year='2016'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\_applications where full\_time\_position ='N'and year='2016'group by job\_title,full\_time\_position,year order by average desc;

**Q.9) Which are employers who have the highest success rate in petitions more than 70% in petitions and total petitions filed more than 1000?(solved by Pig)**

REGISTER '/home/manmath/BigData/external\_jars/piggybank-0.13.0.jar'; --Register external jar 'Piggy Bank.jar'

DEFINE CSVExcelStorage org.apache.pig.piggybank.storage.CSVExcelStorage; -- within the jar define a function CSVExcelStorage()

data = LOAD '/home/ayush/DatasetsandCodes/h1b\_applications.csv' USING CSVExcelStorage() as

(s\_no:int,

case\_status:chararray,

employer\_name:chararray,

soc\_name:chararray,

job\_title:chararray,

full\_time\_position:chararray,

prevailing\_wage:int,

year:chararray,

worksite:chararray,

longitute:double,

latitute:double);

noheader= filter data by $0>=1; --Remove header

cleansed= filter noheader by $1 is not null and $1!='NA';

temp= group cleansed by $2;

total= foreach temp generate group,COUNT(cleansed.$1); --Group by employername and count the case statuses as a whole

certified= filter noheader by $1 == 'CERTIFIED';

temp1= group certified by $2;

totalcertified= foreach temp1 generate group,COUNT(certified.$1); --Group by employername and count the case status = 'CERTIFIED'

certified= filter noheader by $1 == 'CERTIFIED-WITHDRAWN';

temp2= group certified by $2;

totalcertifiedwithdrawn= foreach temp2 generate group,COUNT(certified.$1); --Group by employername and count the case status = 'CERTIFIED-WITHDRAWN'

joined= join totalcertified by $0,totalcertifiedwithdrawn by $0,total by $0;

--dump joined;

joined= foreach joined generate $0,$1,$3,$5;

intermediateoutput= foreach joined generate $0,(float)($1+$2)\*100/($3),$3;

intermediateoutput2= filter intermediateoutput by $1>70 and $2>1000; --Filter by success-rate greater than 70% and petition count above 1000

finaloutput= order intermediateoutput2 by $1 DESC;

dump final output;

**Q.10) Which are the top 10 job positions that have success rate more than 70% in petitions and total petitions filed more than 1000? (solved by Pig)**

REGISTER '/home/manmath/BigData/external\_jars/piggybank-0.13.0.jar'; --Register external jar 'Piggy Bank.jar'

DEFINE CSVExcelStorage org.apache.pig.piggybank.storage.CSVExcelStorage; -- within the jar define a function CSVExcelStorage()

data = LOAD '/home/ayush/DatasetsandCodes/h1b\_applications.csv' USING CSVExcelStorage() as

(s\_no:int,

case\_status:chararray,

employer\_name:chararray,

soc\_name:chararray,

job\_title:chararray,

full\_time\_position:chararray,

prevailing\_wage:int,

year:chararray,

worksite:chararray,

longitute:double,

latitute:double);

noheader= filter data by $0>=1; --Remove header

--Count Total Applications

cleansed= filter noheader by $1 is not null and $1!='NA';

temp= group cleansed by $4;

total= foreach temp generate group,COUNT(cleansed.$1);

--Count Total Applications who are 'CERTIFIED'

certified= filter noheader by $1 == 'CERTIFIED';

temp1= group certified by $4;

totalcertified= foreach temp1 generate group,COUNT(certified.$1);

--Count Total Applications who are 'CERTIFIED-WITHDRAWN'

certified= filter noheader by $1 == 'CERTIFIED-WITHDRAWN';

temp2= group certified by $4;

totalcertifiedwithdrawn= foreach temp2 generate group,COUNT(certified.$1);

--SUCCESS\_RATE=(CERTIFIED+CERTIFIED-WITHDRAWN)/TOTAL X 100

joined= join totalcertified by $0,totalcertifiedwithdrawn by $0,total by $0;

joined= foreach joined generate $0,$1,$3,$5;

intermediateoutput= foreach joined generate $0,(float)($1+$2)\*100/($3),$3;

intermediateoutput2= filter intermediateoutput by $1>70 and $2>1000; --Filter by success-rate greater than 70% and petition count above 1000

finaloutput= order intermediateoutput2 by $1 DESC;

--STORE DATA INTO TEXT FILE

store finaloutput into '/home/ayush/Pig/question10' using PigStorage('\t');

**Q.11) Export result for question no 10 to MySql database.(solved by Sqoop)**

|  |
| --- |
|  |
|  | Hadoop fs –rm –r –f /Pig/Question10  hadoop fs -mkdir -p /Pig/Question10 |
|  | hadoop fs -put /home/manmath/Pig/question10/p\* /Pig/Question10/ |
|  | mysql -u root -p'4397' -e 'drop database question11;create database if not exists question11;use question11;create table question11(job\_title varchar(100),success\_rate float,petitions int);'; |
|  | sqoop export --connect jdbc:mysql://localhost/question11 --username root --password '4397' --table question11 --update-mode allowinsert --export-dir /Pig/Question10/p\* --input-fields-terminated-by '\t'; |
|  | echo -e '\n\nDisplay contents from MySQL Database.\n\n' |
|  | echo -e '\n10) Which are the top 10 job positions that have success rate more than 70% in petitions and total petitions filed more than 1000?\n\n' |
|  | mysql -u root -p'4397' -e 'select \* from question11.question11'; |