**Analyzing H1B Data**

**Using Hadoop Ecosystem**

Presented by-

Name:Arti Hanmant Biradar

StudentID:s181107500054

Centre:NIIT Pune Deccan

**Abstract**

**H1B Data:**

* A Labor Condition Application (LCA) is used by employers as supporting evidence for the petition for an H-1B visa.
* That is billions of Big Data.

**What is Big Data:**

Big data is large volume of data and it may structured, Unstructured or Semi-structured.

**Hadoop:**

Hadoop is framework for storing the data. It uses different Ecosystems to analyzing H1b Big data.

**Hadoop Ecosystem**

* ***Mapreduce*** *: a parallel processing software framework. It is comprised of two steps. Map step is a master node that takes input and partitions them into smaller sub-problems and then distributes them to worker nodes. After the map step has taken place, the master node takes the answer to all of the sub-problems and combines them to produce output.*
* ***Hive*** *: a data warehousing and SQL like query language that presents the data in the form of tables. Hive programming is similar to data Warehousing.*
* ***Pig*** *: a platform for manipulating data stored in HDFS and that includes a compiler for map reduce programs and high level language called Pig Latin. : It is a procedural language platform used to develop a script for MapReduce operations.*
* ***Sqoop*** *:: It is used to import and export data to and from between HDFS and RDBMS*

**Aknowledgement**

I wish to thank my master trainer **Mr.Sandeep Agarwal** and my tech mentor **Mrs.Jyoti Mittal** for providing complete learning on Big data with hadoop and specially thanks for your guidance

**H1B Analysis:**

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

**Technology used: Mapreduce**

package question1a;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

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

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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 DataEnggJob{

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

{

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

{

try{

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

String job\_title=str[4];

String year=str[7];

if(job\_title.contains("DATA ENGINEER"))

{

context.write(NullWritable.get(),new Text(year));

//context.write(key,value);

}

}

catch(Exception e)

{

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

}

}

}

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

{

//private LongWritable result = new LongWritable();

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

int count2011=0;

int count2012=0;

int count2013=0;

int count2014=0;

int count2015=0;

int count2016=0;

long average=0;

long res1=0,res2=0,res3=0,res4=0,res5=0,res6=0;

for ( Text Val:inval)

{

String year=Val.toString();

if(year.equals("2011"))

{

count2011++;

}

else if(year.equals("2012"))

{

count2012++;

}

else if(year.equals("2013"))

{

count2013++;

}

else if(year.equals("2014"))

{

count2014++;

}

else if(year.equals("2015"))

{

count2015++;

}

else if(year.equals("2016"))

{

count2016++;

}

}

if(count2011!=0)

{

res1=(long)(count2012-2011)\*100/(long)count2011;

}

else

{

res1=0;

}

if(count2012!=0)

{

res2=(long)(count2013-2012)\*100/(long)count2012;

}

else

{

res2=0;

}

if(count2013!=0)

{

res3=(long)(count2014-2013)\*100/(long)count2013;

}

else

{

res3=0;

}

if(count2014!=0)

{

res4=(long)(count2015-2014)\*100/(long)count2014;

}

else

{

res4=0;

}

if(count2015!=0)

{

res5=(long)(count2016-2015)\*100/(long)count2015;

}

else

{

res5=0;

}

average=(res1+res2+res3+res4+res5)/5;

String resaverage =String.format("%2f",average);

String final2011=String.format("%d",count2011);

String final2012=String.format("%d",count2012);

String final2013=String.format("%d",count2013);

String final2014=String.format("%d",count2014);

String final2015=String.format("%d",count2015);

String final2016=String.format("%d",count2016);

String finaloutput=final2011+"\t"+final2012+"\t"+final2013+"\t"+final2014+"\t"+final2015+"\t"+final2016+"\t"+"resaverage";

context.write(NullWritable.get(),new Text(resaverage));

}

}

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

Configuration conf = new Configuration();

Job job = new Job (conf, "question1a");

job.setJarByClass(DataEnggJob.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.class);

//job.setNumReduceTasks(2);

job.setMapOutputKeyClass(NullWritable.class);

job.setMapOutputValueClass(Text.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: *year no.of data scientist***

*2011 18*

*2012 32*

*2013 41*

*2014 89*

*2015 160*

*2016 251*

1b)***Find top 5 job titles who are having highest growth in applicatios.***

***Technology used : Mapreduce***

*package question1b;*

*import java.io.IOException;*

*import java.util.TreeMap;*

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

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

*//import org.apache.hadoop.io.IntWritable;*

*import org.apache.hadoop.io.LongWritable;*

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

*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 GrowthApp {*

*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\_title=str[4];*

*String year=str[7];*

*{*

*context.write(new Text(job\_title),new Text(year));*

*//context.write(key,value);*

*}*

*}*

*catch(Exception e)*

*{*

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

*}*

*}*

*}*

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

*{*

*//private LongWritable result = new LongWritable();*

*TreeMap<Double,Text>topMap=new TreeMap<Double,Text>();*

*public void reduce(Text inkey, Iterable<Text> inval,Context context) throws IOException, InterruptedException {*

*int count2011=0;*

*int count2012=0;*

*int count2013=0;*

*int count2014=0;*

*int count2015=0;*

*int count2016=0;*

*long average=0;*

*long res1=0,res2=0,res3=0,res4=0,res5=0,res6=0;*

*for ( Text Val:inval)*

*{*

*if(str[7].equals("2011"))*

*{*

*count2011++;*

*}*

*else if(str[7].equals("2012"))*

*{*

*count2012++;*

*}*

*else if(str[7].equals("2013"))*

*{*

*count2013++;*

*}*

*else if(str[7].equals("2014"))*

*{*

*count2014++;*

*}*

*else if(str[7].equals("2015"))*

*{*

*count2015++;*

*}*

*else if(str[7].equals("2016"))*

*{*

*count2016++;*

*}*

*}*

*if(count2011!=0)*

*{*

*res1=(long)(count2012-2011)\*100/(long)count2011;*

*}*

*else*

*{*

*res1=0;*

*}*

*if(count2012!=0)*

*{*

*res2=(long)(count2013-2012)\*100/(long)count2012;*

*}*

*else*

*{*

*res2=0;*

*}*

*if(count2013!=0)*

*{*

*res3=(long)(count2014-2013)\*100/(long)count2013;*

*}*

*else*

*{*

*res3=0;*

*}*

*if(count2014!=0)*

*{*

*res4=(long)(count2015-2014)\*100/(long)count2014;*

*}*

*else*

*{*

*res4=0;*

*}*

*if(count2015!=0)*

*{*

*res5=(long)(count2016-2015)\*100/(long)count2015;*

*}*

*else*

*{*

*res5=0;*

*}*

*average=(res1+res2+res3+res4+res5)/5;*

*String myvalue=inkey.toString();*

*String resaverage =String.format("%2f",average);*

*String final2011=String.format("%d",count2011);*

*String final2012=String.format("%d",count2012);*

*String final2013=String.format("%d",count2013);*

*String final2014=String.format("%d",count2014);*

*String final2015=String.format("%d",count2015);*

*String final2016=String.format("%d",count2016);*

*String finaloutput=myvalue+"\t"+final2012+"\t"+final2013+"\t"+final2014+"\t"+final2015+"\t"+final2016+"\t"+"resaverage";*

*topMap.put(new Double(average),new Text(myvalue));*

*if(topMap.size()>5)*

*{*

*topMap.remove(topMap.firstKey());*

*}*

*}*

*public void cleanup(Context context)throws IOException,InterruptedException*

*{*

*for(Text t:topMap.descendingMap().values())*

*{*

*context.write(NullWritable.get(),new Text(t));*

*}*

*}*

*}*

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

*Configuration conf = new Configuration();*

*Job job = new Job (conf, "question1b");*

*job.setJarByClass(GrowthApp.class);*

*job.setMapperClass(MapClass.class);*

*//job.setCombinerClass(ReduceClass.class);*

*job.setReducerClass(ReduceClass.class);*

*//job.setNumReduceTasks(2);*

*job.setMapOutputKeyClass(Text.class);*

*job.setMapOutputValueClass(Text.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**:

***job\_title Average growth***

*SENIOR SYSTEMS ANYLIST JC60 4229.8*

*SOFTWARE DEVELOPER 2 3382.8*

*MODULE LEAD 3195.2*

*SYSTEM ANYLIST JC65 2969.8*

*LEAD 2507.0*

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

***Technology used:pig***

***Pig script:question2a.pig***

register /usr/local/hive/lib/hive-exec-1.2.1.jar

register /usr/local/hive/lib/hive-common-1.2.1.jar

data1 = LOAD '/user/hive/warehouse/h1b\_final' USING PigStorage('\t') as (s\_no:double,case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:double,year:chararray,worksite:chararray,longitude,latitude);

job\_title= filter data by $7=='2011';

a= group job\_title by $4;

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

describe step\_a;

job\_title1= filter data1 by $7=='2012';

b= group job\_title1 by $4;

step\_b= foreach b generate group,COUNT($1);

describe step\_b;

job\_title2= filter data1 by $7=='2013';

c= group job\_title2 by $4;

step\_c= foreach c generate group,COUNT($1);

describe step\_c;

job\_title3= filter data1 by $7=='2014';

d= group job\_title3 by $4;

step\_d= foreach d generate group,COUNT($1);

describe step\_d;

job\_title4= filter data1 by $7=='2015';

e= group job\_title4 by $4;

step\_e= foreach e generate group,COUNT($1);

describe step\_e;

job\_title5= filter data1 by $7=='2016';

f= group job\_title5 by $4;

step\_f= foreach f generate group,COUNT($1);

describe step\_f;

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;

describe joined;

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

final = order year by $1 desc;

dump final;

**OUTPUT: Worksite No Of Application**

*SAN FRANCISCO,CALIFORNIA 2011 3*

*SAN FRANCISCO,CALIFORNIA 2012 7*

*MENLO PARK,CALIFORNIA 2013 10*

*MENLO PARK,CALIFORNIA 2014 13*

*SAN FRANCISCO,CALIFORNIA 2015 33*

*MENLO PARK,CALIFORNIA 2016 35*

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

***Technology used:hive***

***Hive :question2b.sql***

*Use h1b\_final;*

**select worksite,count(case\_status),year as temp from h1b\_final where year ='2011' and case\_status='CERTIFIED' group by worksite,year,case\_status order by temp desc limit 5;**

**OUTPUT:**

NEW YORK, NEW YORK 2011 23172

HOUSTON, TEXAS 2011 8184

CHICAGO, ILLINOIS 2011 5188

SAN JOSE, CALIFORNIA 2011 4713

SAN FRANCISCO, CALIFORNIA 2011 4711

NEW YORK, NEW YORK 2012 23737

HOUSTON, TEXAS 2012 9963

SAN FRANCISCO, CALIFORNIA 2012 6116

CHICAGO, ILLINOIS 2012 5671

ATLANTA, GEORGIA 2012 5565

NEW YORK, NEW YORK 2013 23537

HOUSTON, TEXAS 2013 11136

SAN FRANCISCO, CALIFORNIA 2013 7281

SAN JOSE, CALIFORNIA 2013 6722

ATLANTA, GEORGIA 2013 6377

NEW YORK, NEW YORK 2014 27634

HOUSTON, TEXAS 2014 13360

SAN FRANCISCO, CALIFORNISA 2014 9798

SAN JOSE, CALIFORNIA 2014 8223

ATLANTA, GEORGIA 2014 8213

NEW YORK, NEW YORK 2015 31266

HOUSTON, TEXAS 2015 15242

SAN FRANCISCO, CALIFORNIA 2015 12594

ATLANTA, GEORGIA 2015 10500

SAN JOSE, CALIFORNIA 2015 9589

NEW YORK, NEW YORK 2016 34639

SAN FRANCISCO, CALIFORNIA 2016 13836

HOUSTON, TEXAS 2016 13655

ATLANTA, GEORGIA 2016 11678

CHICAGO, ILLINOIS 2016 11064

**Q3)which industry(SOC\_NAME)has the most number of Data Scientist positions?[certified]**

**Technology used : Mapreduce**

package question3;

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.Reducer;

import java.io.IOException;

import java.util.TreeMap;

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 DataScientistJob3

{

public static class DESOCMapper 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("\t");

if (token[4].contains("DATA SCIENTIST") && token[1].equals("CERTIFIED"))

{

Text answer = new Text(token[3]);

context.write(answer, one);

}

}

}

}

public static class DESOCReducer 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);

}

}

}

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

Configuration conf = new Configuration();

Job job =new Job(conf, "Data Scientist jobs");

job.setJarByClass(DataScientistJob3 .class);

job.setMapperClass(DESOCMapper.class);

job.setReducerClass(DESOCReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(LongWritable.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

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

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

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

}

}

**OUTPUT**:

**Soc\_Name** **Count**

STATISTICIANS,369

COMPUTER AND INFORMATION RESEARCH SCIENTISTS,283

OPERATIONS RESEARCH ANALYSTS,237

Computer and Information Research Scientists,115

COMPUTER OCCUPATIONS, ALL OTHER,113

MATHEMATICIANS,107

**Q4)which top 5 employers file the most petitions each year?-case status-ALL**

**Technology used : Mapreduce**

package question4;

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

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.mapreduce.lib.input.FileInputFormat;

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

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

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

public class DataEnggGrowth4 {

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

{

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

{

try

{

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

String job\_title=str[4];

String year=str[7];

{

context.write(new Text(job\_title),new Text(year));

}

}

catch (Exception e)

{

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

}

}

}

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

{

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());

}

public void cleanup(Context context) throws IOException,

InterruptedException {

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

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

}

}

public static class DataEnggGrowth extends Partitioner < Text, LongWritable >

{

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

{

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

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

{

return 0;

}

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

{

return 1;

}

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

{

return 2;

}

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

{

return 3;

}

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

{

return 4;

}

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

{

return 5;

}

else

{

return 6;

}

}

}

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

{

Configuration conf= new Configuration();

Job job= new Job(conf,"Question 4");

job.setJarByClass(DataEnggGrowth4.class);

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

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

job.setMapperClass(DataEngineerMapper.class);

job.setPartitionerClass(DataEnggGrowth.class);

job.setReducerClass(DataEngineerReducer.class);

job.setNumReduceTasks(7);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(LongWritable.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

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

}

}

**OUTPUT:**

**Employear Name Year Count**

TATA CONSULTANCY SERVICES LIMITED 2011 5416

MICROSOFT CORPORATION 2011 4253

DELOITTE CONSULTING LLP 2011 3621

WIPRO LIMITED 2011 3028

COGNIZANT TECHNOLOGY SOLUTIONS U.S. CORPORATION 2011 2721

INFOSYS LIMITED 2012 15818

WIPRO LIMITED 2012 7189

TATA CONSULTANCY SERVICES LIMITED 2012 6735

DELOITTE CONSULTING LLP 2012,4727

IBM INDIA PRIVATE LIMITED 2012,4074

INFOSYS LIMITED 2013,32223

TATA CONSULTANCY SERVICES LIMITED 2013,8790

WIPRO LIMITED 2013,6734

DELOITTE CONSULTING LLP 2013,6124

ACCENTURE LLP 2013,4994

INFOSYS LIMITED 2014,23759

TATA CONSULTANCY SERVICES LIMITED 2014,14098

WIPRO LIMITED 2014,8365

DELOITTE CONSULTING LLP 2014,7017

ACCENTURE LLP 2014,5498

INFOSYS LIMITED 2015,33245

TATA CONSULTANCY SERVICES LIMITED 2015,16553

WIPRO LIMITED 2015,12201

IBM INDIA PRIVATE LIMITED 2015,10693

ACCENTURE LLP 2015,9605

PROGRAMMER ANYLIST 2016,53743

SOFTWARE ENGINEER 2016,30668

SOFTWARE DEVELOPER LIMITED 2016,14041

SYSTEM ANYLIST 2016,12314

COMPUTER PROGRAMMER 2016,11668

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

**Technology used:hive**

**Hive file:question5.sql**

Use h1b\_final;

select count(job\_title),year,case\_status as temp from h1b\_final where year = 2011 group by job\_title,year,case\_status order by temp desc limit 10;

**OUTPUT:**

**YEAR JOB\_TITLE COUNT**

2011 PROGRAMMER ANALYST 31799

2011 SOFTWARE ENGINEER 12763

2011 COMPUTER PROGRAMMER 8998

2011 SYSTEMS ANALYST 8644

2011 BUSINESS ANALYST 3891

2011 COMPUTER SYSTEMS ANALYST 3698

2011 ASSISTANT PROFESSOR 3467

2011 PHYSICAL THERAPIST 3377

2011 SENIOR SOFTWARE ENGINEER 2935

2011 SENIOR CONSULTANT 2798

2012 PROGRAMMER ANALYST 33066

2012 SOFTWARE ENGINEER 14437

2012 COMPUTER PROGRAMMER 9629

2012 SYSTEMS ANALYST 9296

2012 BUSINESS ANALYST 4752

2012 COMPUTER SYSTEMS ANALYST 4706

2012 SOFTWARE DEVELOPER 3895

2012 PHYSICAL THERAPIST 3871

2012 ASSISTANT PROFESSOR 3801

2012 SENIOR CONSULTANT 3737

2013 PROGRAMMER ANALYST 33880

2013 SOFTWARE ENGINEER 15680

2013 COMPUTER PROGRAMMER 11271

2013 SYSTEMS ANALYST 8714

2013 TECHNOLOGY LEAD - US 7853

2013 TECHNOLOGY ANALYST - US 7683

2013 BUSINESS ANALYST 5716

2013 COMPUTER SYSTEMS ANALYST 5043

2013 SOFTWARE DEVELOPER 5026

2013 SENIOR CONSULTANT 4326

2014 PROGRAMMER ANALYST 43114

2014 SOFTWARE ENGINEER 20500

2014 COMPUTER PROGRAMMER 14950

2014 SYSTEMS ANALYST 10194

2014 SOFTWARE DEVELOPER 7337

2014 BUSINESS ANALYST 7302

2014 COMPUTER SYSTEMS ANALYST 6821

2014 TECHNOLOGY LEAD - US 5057

2014 TECHNOLOGY ANALYST - US 4913

2014 SENIOR CONSULTANT 4898

2015 PROGRAMMER ANALYST 53436

2015 SOFTWARE ENGINEER 27259

2015 COMPUTER PROGRAMMER 14054

2015 SYSTEMS ANALYST 12803

2015 SOFTWARE DEVELOPER 10441

2015 BUSINESS ANALYST 8853

2015 TECHNOLOGY LEAD - US 8242

2015 COMPUTER SYSTEMS ANALYST 7918

2015 TECHNOLOGY ANALYST - US 7014

2015 SENIOR SOFTWARE ENGINEER 6013

2016 PROGRAMMER ANALYST 53743

2016 SOFTWARE ENGINEER 30668

2016 SOFTWARE DEVELOPER 14041

2016 SYSTEMS ANALYST 12314

2016 COMPUTER PROGRAMMER 11668

2016 BUSINESS ANALYST 9167

2016 COMPUTER SYSTEMS ANALYST 6900

2016 SENIOR SOFTWARE ENGINEER 6439

2016 DEVELOPER 6084

2016 TECHNOLOGY LEAD - US 5410

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

***Technology used:pig***

***Pig script:question6.pig***

register /usr/local/hive/lib/hive-exec-1.2.1.jar

register /usr/local/hive/lib/hive-common-1.2.1.jar

data = LOAD '/user/hive/warehouse/h1b\_final' USING PigStorage('\t') as (s\_no:double,case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:double,year:chararray,worksite:chararray,longitude,latitude);

number= filter data by $1 is not null and $1!='NA';

temp= group number by $7;

total= foreach temp generate group,COUNT(number.$0);

dump total;

number1= filter data by $7 is not null and $7!='NA';

temp1= group number1 by ($7,$1);

yearsoccount= foreach temp1 generate group,group.$0,COUNT($1);

dump yearsoccount;

joined= join yearsoccount by $1,total by $0;

ans= foreach joined generate FLATTEN($0),(long)($2\*100)/$4,$2;

dump ans;

**OUTPUT:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Year** | **certified** | **Certified Withdrawn** | **withdrawn** | **denied** |
|  |  |  |  |  |
| 2011 | 84.83 | 3.23 | 2.81 | 8.11 |
| 2012 | 84.85 | 7.48 | 2.58 | 5.07 |
| 2013 | 86.61 | 8.01 | 2.62 | 2.74 |
| 2014 | 87.62 | 6.99 | 3.08 | 2.29 |
| 2015 | 88.45 | 6.63 | 3.14 | 1.76 |
| 2016 | 87.93 | 7.26 | 3.37 | 1.41 |

**GRAPH:**

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

***Technology used:Hive***

***Hive file:question7.sql***

select year,count(\*)as num\_application from h1b\_final group by year order by num\_application;

**OUTPUT**:

***Year Applicant***

*2011 358767*

*2012 415607*

*2013 442114*

*2014 519427*

*2015 618727*

*2016 647803*

***GRAPH:***

***8) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate)***

***Technology used:Hive***

***Hive file:question8.sql***

*Use h1b\_final;*

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

**OUTPUT**:

**Job\_title full\_time\_position average\_prevailing\_wage**

For 2011:--

(ASSOCIATE SQA ENGINEER,Y) 9853122.909090908

(TEACHER (MATHEMATICS),Y) 9491354.0

(SENIOR AUDIT ASSOCIATE,Y) 9422870.636363637

(ELEMENTARY SCHOOL SPANISH TEACHER,Y) 9078950.0

(DATA ADMINISTRATOR,N) 8976796.8

(SENIOR FINANCE MANAGER,Y) 8831672.0625

(PRINCIPAL ARCHITECT,Y) 8453338.3

(AUDIT SUPERVISOR,Y) 8175795.071428572

(SR. VICEPRESIDENT, WORLD WIDE BUSINESS DEVELOPMEN,Y) 7991400.0

(TECHNOLOGY CONSULTANT III,Y) 7570872.04

For 2012:--

(LEAD SYSTEM CONSULTANT - SYSTEM ANALYSIS AND PROGR,Y) 9571493.714285715

(PRODUCER/DIRECTOR/WRITER,Y) 9214400.0

(SPANISH TEACHER,N) 9069436.3

(JUNIOR DEVELOPER,Y) 7275807.75

(STAFF CONSULTANT - JAVA DEVELOPER,Y) 6570255.85

(FINANCIAL TECHNOLOGY ASSOCIATE,Y) 6446600.25

(MOBILE APPLICATION DEVELOPER,Y) 6360284.6

(SVP STRATEGY,Y) 5978897.0

(SECURITY ENGINEER,Y) 5677916.903225807

(LIFE SCIENCE RESEARCH ASSISTANT,Y) 5479102.0869565215

(ARCHITECTURAL DESIGNER I,Y) 4769707.521739131

For 2013:---

(ANALYST - INVESTMENT BANK,Y) 9551639.57142857

(BUSINESS DEVELOPMENT COORDINATOR,Y) 9398888.363636363

(ORACLE FUNCTIONAL CONSULTANT,Y) 9140731.23076923

(CERTIFICATION ENGINEER,Y) 8550318.461538462

(DIRECTOR OF PRODUCT MANAGEMENT,Y) 8489674.136363637

(TAX PREPARER,Y) 8355148.666666667

(SAP MM FUNCTIONAL ANALYST,Y) 8247348.666666667

(INTEGRATION ARCHITECT,Y) 8184703.055555556

(VICE PRESIDENT OF BUSINESS DEVELOPMENT,Y) 8179956.75

(NETWORK AND SYSTEMS ADMINISTRATOR,Y) 7921504.388888889

For 2014:---

(TELECOMMUNICATIONS ENGINEER,Y) 9696668.804347826

(STATISTICAL PROGRAMMER 2,Y) 9377228.066666666

(IT DEVELOPER,Y) 9208398.0

(SENIOR PRICING ANALYST,Y) 9034237.222222222

(IT ENGINEER,Y) 9007724.764705881

(LEAD TECHNICAL ARCHITECT,Y) 8886245.0

(SENIOR ENGINEER PROCESS ENGINEERING,Y) 8360763.55

(SR. ELECTRICAL ENGINEER,Y) 7756294.352941177

(QUALITY ASSURANCE,Y) 7679735.5

(ASIC DESIGN ENGINEER,Y) 7586330.675675675

For 2015:----

(SENIOR PROGRAM OFFICER,Y) 9811738.7

(MIDDLE SCHOOL SPECIAL EDUCATION TEACHER,Y) 6633998.666666667

(CORPORATE APPLICATIONS ENGINEER,Y) 5990572.703703703

(ASSOCIATE DEVELOPER,Y) 5167618.24

(CONSTRUCTION PROJECT MANAGER,Y) 4099085.433333333

(ANALYTICS CONSULTANT,Y) 3692850.552631579

(EDI DEVELOPER,Y) 3549424.379310345

(JEWELRY DESIGNER,Y) 3333434.9411764704

(APPLICATION DESIGNER/DEVELOPER/ORACLE CONSULTANT,Y) 2546491.0

(MATHEMATICS TEACHER,Y) 2399001.112359551

For:2016---

(QA MANAGER,Y) 7307333.0

(INFORMATION TECHNOLOGY SPECIALIST,Y) 7100751.956521739

(DEVELOPMENT ENGINEER,Y) 6763973.636363637

(ESTIMATOR,Y) 6419952.428571428

(DATABASE ADMINISTRATOR (DBA),Y) 3325662.6153846155

(STATISTICAL PROGRAMMER,Y) 2917655.93877551

(ACCOUNTANT,Y) 2644324.888888889

(QA TESTER,Y) 1675558.3035714286

(HUMAN RESOURCES GENERALIST,Y) 1577342.0

(SYSTEMS ARCHITECT,Y) 1505638.5391304349

***9) Which are top ten employers who have the highest success rate in petitions?***

***Technology used:pig***

***Pig script:question9.pig***

register /usr/local/hive/lib/hive-exec-1.2.1.jar

register /usr/local/hive/lib/hive-common-1.2.1.jar

data = LOAD 'hdfs://localhost:54310/user/hive/warehouse/niit\_h1b.db/h1b\_final' USING PigStorage('\t') as (s\_no:double,case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:double,year:chararray,worksite:chararray,longitude,latitude);

number= filter data by $1 is not null and $1!='NA';

petitions= group number by $2;

total= foreach petitions generate group,COUNT(number.$0);

dump total;

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

petitions1= group certified by $2;

totalcertified= foreach petitions1 generate group,COUNT(certified.$0);

dump totalcertified;

certified\_with= filter data by $1 == 'CERTIFIED-WITHDRAWN';

petitions2= group certified\_with by $2;

totalcertifiedwithdrawn= foreach petitions2 generate group,COUNT(certified\_with.$0);

dump totalcertifiedwithdrawn;

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

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

dump joined;

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

dump intermediateoutput;

result = filter intermediateoutput by $1>70 and $2>1000;

final = order result by $1 desc;

dump final;

**OUTPUT:**

**Employer name Total SuccessRate**

HTC GLOBAL SERVICES, INC. 1164 100.0

INFOSYS LIMITED 130592 99.540

DIASPARK, INC. 1419.0 99.506

ACCENTURE LLP 33447.0 99.393

TECH MAHINDRA (AMERICAS),INC 10732.0 99.338

TATA CONSULTANCY SERVICES LIMITED 64726 99.337

***10) Which are the top 10 job positions which have the highest success rate in petitions?***

***Technology used:pig***

***Pig script:question10.pig***

register /usr/local/hive/lib/hive-exec-1.2.1.jar

register /usr/local/hive/lib/hive-common-1.2.1.jar

data = LOAD '/user/hive/warehouse/h1b\_final' USING PigStorage('\t') as (s\_no:double,case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:double,year:chararray,worksite:chararray,longitude,latitude);

number= filter data by $1 is not null and $1!='NA';

petitions= group number by $4;

total= foreach petitions generate group,COUNT(number.$0);

dump total;

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

petitions1= group certified by $4;

totalcertified= foreach petitions1 generate group,COUNT(certified.$0);

dump totalcertified;

certified\_with= filter data1 by $1 == 'CERTIFIED-WITHDRAWN';

prtitions2= group certified\_with by $4;

totalcertifiedwithdrawn= foreach petitions2 generate group,COUNT(certified\_with.$0);

dump totalcertifiedwithdrawn;

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

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

dump joined;

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

dump intermediateoutput;

result= filter intermediateoutput by $1>70 and $2>1000;

final= order result by $1 desc;

dump final;

**OUTPUT:**

**Job title Total Success Rate**

PRODUCTION SUPPORT LEAD – US 1301.0 100.0

ASSOCIATE CONSULTANT - US 4393.0 99.93

SYSTEMS ENGINEER - US 10036.0 99.90

TEST ENGINEER - US 2198.0 99.86

PRODUCTION SUPPORT ANALYST – US 1451.0 99.86

TEST ANALYST - US 4958.0 99.81

***11) Export result for question no 10 to MySql database***