**Analyzing H1B**

**Data by**

**Using Hadoop**

**Ecosystems**

**Presented By**

**Name: Pravinsing Prakashsing Hajari**

**Student ID:S171107500148**

**Center:Deccan Pune.**

**Abstract**

**H1B Data:**

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

As the World Wide Web grew in the late 1900s and early 2000s, search engines and indexes were created to help locate relevant information amid the text-based content. In the early years, search results were returned by humans. But as the web grew from dozens to millions of pages, automation was needed. Web crawlers were created, many as university-led research projects, and search engine start-ups took off (Yahoo, AltaVista, etc.)

One such project was an open-source web search engine called Nutch – the brainchild of Doug Cutting and Mike Cafarella. They wanted to return web search results faster by distributing data and calculations across different computers so multiple tasks could be accomplished simultaneously. During this time, another search engine project called Google was in progress. It was based on the same concept – storing and processing data in a distributed, automated way so that relevant web search results could be returned faster.

In 2006, Cutting joined Yahoo and took with him the Nutch project as well as ideas based on Google’s early work with automating distributed data storage and processing. The Nutch project was divided – the web crawler portion remained as Nutch and the distributed computing and processing portion became Hadoop (named after Cutting’s son’s toy elephant). In 2008, Yahoo released Hadoop as an open-source project. Today, Hadoop’s framework and ecosystem of technologies are managed and maintained by the non-profit Apache Software Foundation (ASF), a global community of software developers and contributors.

**Importance of Hadoop**

1. **Scalable:** Hadoop is a highly scalable storage platform, because it can store and distribute very large data sets.
2. **Cost effective:** Hadoop also offers a cost effective storage solution for businesses' exploding data sets.
3. **Flexible:** Hadoop enables businesses to easily access new data sources and tap into different types of data to generate value from that data.
4. A key advantage of using Hadoop is its **fault tolerance.**

**Hadoop Glosary**

**1. HDFS:** The **Hadoop Distributed File System** (**HDFS**) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks.

**2.YARN:** YARN (Yet Another Resource Negotiator) is a cluster management technology.

**Hadoop Ecosystems:**

**Mapreduce:** MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.

**Hive:** It is a platform used to develop SQL type scripts to do MapReduce operations.

**Pig:** 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:**

**Task-1:**

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

**Technology Used: Pig.**

**PigScript: h1b1a.pig**

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);

data2= foreach data1 generate s\_no,year,job\_title;

job= filter data2 by job\_title== 'DATA ENGINEER';

total= group job by year;

count= foreach total generate group as year,COUNT(job);

final= order count by $0 asc;

dump final;

STORE final INTO 'pig\_out/output\_1a' USING PigStorage();

**Output:**

Year Count

2011 18

2012 32

2013 41

2014 89

2015 160

2016 251

**Graph:**

**b) Find top 5 job titles who are having highest avg growth in applications.[ALL]**

**Technology Used: Pig**

**PigScript:h1b\_q1b.pig.**

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);

data2 = foreach data1 generate job\_title,year;

year1= filter data2 by year=='2011';

year2= filter data2 by year=='2012';

year3= filter data2 by year=='2013';

year4= filter data2 by year=='2014';

year5= filter data2 by year=='2015';

year6= filter data2 by year=='2016';

total1= group year1 by(job\_title);

total2= group year2 by(job\_title);

total3= group year3 by(job\_title);

total4= group year4 by(job\_title);

total5= group year5 by(job\_title);

total6= group year6 by(job\_title);

count1= foreach total1 generate group,COUNT(year1.job\_title);

count2= foreach total2 generate group,COUNT(year2.job\_title);

count3= foreach total3 generate group,COUNT(year3.job\_title);

count4= foreach total4 generate group,COUNT(year4.job\_title);

count5= foreach total5 generate group,COUNT(year5.job\_title);

count6= foreach total6 generate group,COUNT(year6.job\_title);

JOIN1= join count1 by $0,count2 by $0,count3 by $0,count4 by $0,count5 by $0,count6 by $0;

output1= foreach JOIN1 generate $0,$1,$3,$5,$7,$9,$11;

finaloutput1= foreach output1 generate $0,(DOUBLE)((($2-$1)\*100)/$1+(($3-$2)\*100)/$2+(($4-$3)\*100)/$3+(($5-$4)\*100)/$4+(($6-$5)\*100)/$5);

finaloutput2= foreach finaloutput1 generate $0,(DOUBLE)($1/5);

finalavg= limit (order finaloutput2 by $1 desc) 5;

dump finalavg;

STORE finalavg INTO 'pig\_out/output\_1b' USING PigStorage();

**Output:**

**job\_title Average Growth**

SENIOR SYSTEMS ANALYST JC60, 4255.4

SOFTWARE DEVELOPER 2, 3480.8

PROJECT MANAGER 3, 3233.4

SYSTEMS ANALYST JC65, 2985.0

MODULE LEAD, 2917.2

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

**Technology Used:Pig**

**Pig Script: h1b\_q2a.pig.**

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);

data2 = foreach data1 generate worksite,job\_title,year;

job = filter data2 by job\_title== 'DATA ENGINEER' AND year== '2011';

job1 = filter data2 by job\_title== 'DATA ENGINEER' AND year== '2012';

job2= filter data2 by job\_title== 'DATA ENGINEER' AND year== '2013';

job3= filter data2 by job\_title== 'DATA ENGINEER' AND year== '2014';

job4= filter data2 by job\_title== 'DATA ENGINEER' AND year== '2015';

job5= filter data2 by job\_title== 'DATA ENGINEER' AND year== '2016';

total= group job by (year,worksite);

total1= group job1 by (year,worksite);

total2= group job2 by (year,worksite);

total3= group job3 by (year,worksite);

total4= group job4 by (year,worksite);

total5= group job5 by (year,worksite);

count= foreach total generate group as year,COUNT(job);

count1= foreach total1 generate group as year,COUNT(job1);

count2= foreach total2 generate group as year,COUNT(job2);

count3= foreach total3 generate group as year,COUNT(job3);

count4= foreach total4 generate group as year,COUNT(job4);

count5= foreach total5 generate group as year,COUNT(job5);

final= limit(order count by $1 desc) 1;

final1= limit(order count1 by $1 desc) 1;

final2= limit(order count2 by $1 desc) 1;

final3= limit(order count3 by $1 desc) 1;

final4= limit(order count4 by $1 desc) 1;

final5= limit(order count5 by $1 desc) 1;

final6= union final,final1,final2,final3,final4,final5;

dump final6;

STORE final INTO 'pig\_out/output\_2a' USING PigStorage();

**Output:**

**Worksite Year Count**

SAN FRANCISCO, CALIFORNIA 2011 03

SAN FRANCISCO, CALIFORNIA 2012 07

MENLO PARK, CALIFORNIA 2013 10

MENLO PARK, CALIFORNIA 2014 13

SAN FRANCISCO, CALIFORNIA 2015 33

MENLO PARK, CALIFORNIA 2016 35

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

**Technology Used: Hive.**

**select worksite, year,count(worksite) as location from h1b\_final where case\_Status= 'CERTIFIED'AND year = '2011' group by year,worksite order by location 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

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

**[certified**]

**Technology Used:Hive**

**select soc\_name,count(job\_title) as job from h1b\_final where job\_title= 'DATA SCIENTIST' AND case\_status= 'CERTIFIED'group by soc\_name order by job desc limit 1;**

**Output:**

**SOC\_NAME Count**

STATISTICIANS 369

**4)Which top 5 employers file the most petitions each year? - Case Status – ALL**

**Technology Used: Hive.**

**select year,employer\_name,count(\*) as total from h1b\_final where year='2011' group by year,employer\_name order by total desc limit 5;**

**OUTPUT:**

**Employer\_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 7182

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

INFOSYS LIMITED 2016 25352

CAPGEMINI AMERICA INC 2016 16725

TATA CONSULTANCY SERVICES LIMITED 2016 13134

WIPRO LIMITED 2016 10607

IBM INDIA PRIVATE LIMITED 2016 9787

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

**a) for all the applications**

**Technology Used: Hive**

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

**OUTPUT:**

**Job\_title Year Count**

PROGRAMMER ANALYST 2011 31799

SOFTWARE ENGINEER 2011 12763

COMPUTER PROGRAMMER 2011 8998

SYSTEMS ANALYST 2011 8644

BUSINESS ANALYST 2011 3891

COMPUTER SYSTEMS ANALYST 2011 3698

ASSISTANT PROFESSOR 2011 3467

PHYSICAL THERAPIST 2011 3377

SENIOR SOFTWARE ENGINEER 2011 2935

SENIOR CONSULTANT 2011 2798

PROGRAMMER ANALYST 2012 33066

SOFTWARE ENGINEER 2012 14437

COMPUTER PROGRAMMER 2012 9629

SYSTEMS ANALYST 2012 9296

BUSINESS ANALYST 2012 4752

COMPUTER SYSTEMS ANALYST 2012 4706

SOFTWARE DEVELOPER 2012 3895

PHYSICAL THERAPIST 2012 3871

ASSISTANT PROFESSOR 2012 3801

PROGRAMMER ANALYST 2013 33880

SOFTWARE ENGINEER 2013 15680

COMPUTER PROGRAMMER 2013 11271

SYSTEMS ANALYST 2013 8714

TECHNOLOGY LEAD - US 2013 7853

TECHNOLOGY ANALYST - US 2013 7683

BUSINESS ANALYST 2013 5716

COMPUTER SYSTEMS ANALYST 2013 5043

SOFTWARE DEVELOPER 2013 5026

SENIOR CONSULTANT 2013 4326

PROGRAMMER ANALYST 2014 43114

SOFTWARE ENGINEER 2014 20500

COMPUTER PROGRAMMER 2014 14950

SYSTEMS ANALYST 2014 10194

SOFTWARE DEVELOPER 2014 7337

BUSINESS ANALYST 2014 7302

COMPUTER SYSTEMS ANALYST 2014 6821

TECHNOLOGY LEAD - US 2014 5057

TECHNOLOGY ANALYST - US 2014 4913

SENIOR CONSULTANT 2014 4898

PROGRAMMER ANALYST 2015 53436

SOFTWARE ENGINEER 2015 27259

COMPUTER PROGRAMMER 2015 14054

SYSTEMS ANALYST 2015 12803

SOFTWARE DEVELOPER 2015 10441

BUSINESS ANALYST 2015 8853

TECHNOLOGY LEAD - US 2015 8242

COMPUTER SYSTEMS ANALYST 2015 7918

TECHNOLOGY ANALYST - US 2015 7014

SENIOR SOFTWARE ENGINEER 2015 6013

PROGRAMMER ANALYST 2016 53743

SOFTWARE ENGINEER 2016 30668

SOFTWARE DEVELOPER 2016 14041

SYSTEMS ANALYST 2016 12314

COMPUTER PROGRAMMER 2016 11668

BUSINESS ANALYST 2016 9167

COMPUTER SYSTEMS ANALYST 2016 6900

SENIOR SOFTWARE ENGINEER 2016 6439

DEVELOPER 2016 6084

TECHNOLOGY LEAD - US 2016 5410

**b) for only certified applications.**

**Technology Used: Hive**

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

**OUTPUT:**

**Job\_title Year Count**

PROGRAMMER ANALYST 2011 28806

SOFTWARE ENGINEER 2011 11224

COMPUTER PROGRAMMER 2011 8038

SYSTEMS ANALYST 2011 7850

BUSINESS ANALYST 2011 3444

COMPUTER SYSTEMS ANALYST 2011 3152

ASSISTANT PROFESSOR 2011 3050

PHYSICAL THERAPIST 2011 2911

SENIOR SOFTWARE ENGINEER 2011 2595

SENIOR CONSULTANT 2011 2585

PROGRAMMER ANALYST 2012 29226

SOFTWARE ENGINEER 2012 12273

COMPUTER PROGRAMMER 2012 8483

SYSTEMS ANALYST 2012 8399

BUSINESS ANALYST 2012 4144

COMPUTER SYSTEMS ANALYST 2012 4084

SENIOR CONSULTANT 2012 3420

SOFTWARE DEVELOPER 2012 3290

PHYSICAL THERAPIST 2012 3284

ASSISTANT PROFESSOR 2012 3033

PROGRAMMER ANALYST 2013 29906

SOFTWARE ENGINEER 2013 12973

COMPUTER PROGRAMMER 2013 10202

SYSTEMS ANALYST 2013 7850

TECHNOLOGY LEAD - US 2013 7809

TECHNOLOGY ANALYST - US 2013 7641

BUSINESS ANALYST 2013 4993

COMPUTER SYSTEMS ANALYST 2013 4554

SOFTWARE DEVELOPER 2013 4316

PROGRAMMER ANALYST 2014 38625

SOFTWARE ENGINEER 2014 17278

COMPUTER PROGRAMMER 2014 13796

SYSTEMS ANALYST 2014 9161

BUSINESS ANALYST 2014 6529

SOFTWARE DEVELOPER 2014 6473

COMPUTER SYSTEMS ANALYST 2014 6204

TECHNOLOGY LEAD - US 2014 5055

TECHNOLOGY ANALYST - US 2014 4911

SENIOR CONSULTANT 2014 4535

PROGRAMMER ANALYST 2015 48203

SOFTWARE ENGINEER 2015 23352

COMPUTER PROGRAMMER 2015 12971

SYSTEMS ANALYST 2015 11498

SOFTWARE DEVELOPER 2015 9343

TECHNOLOGY LEAD - US 2015 8238

BUSINESS ANALYST 2015 7919

COMPUTER SYSTEMS ANALYST 2015 7234

TECHNOLOGY ANALYST - US 2015 7009

SENIOR SOFTWARE ENGINEER 2015 5324

PROGRAMMER ANALYST 2016 47964

SOFTWARE ENGINEER 2016 25890

SOFTWARE DEVELOPER 2016 12474

SYSTEMS ANALYST 2016 10986

COMPUTER PROGRAMMER 2016 10528

BUSINESS ANALYST 2016 8175

COMPUTER SYSTEMS ANALYST 2016 6205

DEVELOPER 2016 5912

SENIOR SOFTWARE ENGINEER 2016 5630

TECHNOLOGY LEAD - US 2016 5405

**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.**

**Technology Used: Map**

**percentage\_app.java**

import java.io.IOException;

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.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 percentage\_app {public static class MapClass extends Mapper<LongWritable,Text,Text,Text>

{

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

{

try{

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

String year = str[7];

String case\_status = str[1];

context.write(new Text(year),new Text(case\_status));

}

catch(Exception e) {

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

}

}

}

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

{

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

double total=0;

double certified\_withdrawn=0;

double certified=0;

double withdrawn=0;

double denied=0;

double certified\_percent=0;

double certified\_withdrawn\_percent=0;

double withdrawn\_percent=0;

double denied\_percent=0;

for(Text v:values)

{

total++;

String case\_status=v.toString();

if(case\_status.equals("CERTIFIED") )

{

certified++;

}

else if ( case\_status.equals("CERTIFIED-WITHDRAWN"))

{

certified\_withdrawn++;

}

else if(case\_status.equals("WITHDRAWN"))

{

withdrawn++;

}

else

{

denied++;

}

}

certified\_percent=((certified\*100)/(total));

certified\_withdrawn\_percent=((certified\_withdrawn\*100)/(total));

withdrawn\_percent=((withdrawn\*100)/(total));

denied\_percent=((denied\*100)/(total));

String output=total+"\t"+certified+"\t"+certified\_percent+"\t"+certified\_withdrawn+"\t"+certified\_withdrawn\_percent+"\t"+withdrawn+"\t"+withdrawn\_percent+"\t"+denied+"\t"+denied\_percent;

context.write(new Text(key),new Text(output));

}

}

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 = new Job (conf, "Count");

job.setJarByClass(percentage\_app.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(Text.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 | certified | certiwit | 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 [All]**

**Technology Used: Map reduce**

**total\_app.java.**

import java.io.IOException;

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.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 total\_app {

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

{

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

{

try{

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

String year = str[7];

String sr\_no= str[0];

context.write(new Text(year),new Text(sr\_no));

}

catch(Exception e) {

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

}

}

}

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

{

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

long count=0;

for ( Text V:values)

{

count++;

}

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 = new Job (conf, "Count");

job.setJarByClass(total\_app.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(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:**

**Year Count**

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). Arrange the output in descending order - [Certified and Certified Withdrawn.]**

**Technology Used: Pig.**

**PigScript: h1b8.pig**

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);

data2 = FOREACH data1 GENERATE year,case\_status,job\_title,prevailing\_wage,full\_time\_position;

--dump data2;

year11 = FILTER data2 BY (year == '2011') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED')) AND full\_time\_position =='Y';

GROUP11 = group year11 by (job\_title,year,full\_time\_position);

AVG1 = foreach GROUP11 generate group, (double)AVG(year11.prevailing\_wage) as count;

RESULT\_11\_Y = order AVG1 by $1 desc;

year12 = FILTER data2 BY (year == '2012') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='Y';

GROUP12 = group year12 by (job\_title,year,full\_time\_position);

AVG2 = foreach GROUP12 generate group, (double)AVG(year12.prevailing\_wage) as count ;

RESULT\_12\_Y = order AVG2 by $1 desc;

year13 = FILTER data2 BY (year == '2013') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='Y';

GROUP13 = group year13 by (job\_title,year,full\_time\_position);

AVG3 = foreach GROUP13 generate group, (double)AVG(year13.prevailing\_wage) ;

RESULT\_13\_Y = order AVG3 by $1 desc;

year14 = FILTER data2 BY (year == '2014') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='Y';

GROUP14 = group year14 by (job\_title,year,full\_time\_position);

AVG4 = foreach GROUP14 generate group, (double)AVG(year14.prevailing\_wage) ;

RESULT\_14\_Y= order AVG4 by $1 desc;

year15 = FILTER data2 BY (year == '2015') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='Y';

GROUP15 = group year15 by (job\_title,year,full\_time\_position);

AVG5 = foreach GROUP15 generate group, (double)AVG(year15.prevailing\_wage) ;

RESULT\_15\_Y= order AVG5 by $1 desc;

year16 = FILTER data2 BY (year == '2016') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='Y';

GROUP16 = group year16 by (job\_title,year,full\_time\_position);

AVG6 = foreach GROUP16 generate group, (double)AVG(year16.prevailing\_wage) ;

RESULT\_16\_Y= order AVG6 by $1 desc;

FINAL\_Y = UNION RESULT\_11\_Y,RESULT\_12\_Y,RESULT\_13\_Y,RESULT\_14\_Y,RESULT\_15\_Y,RESULT\_16\_Y;

STORE FINAL\_Y INTO 'pig\_out/prob\_8\_Y' USING PigStorage();

--FOR PART\_TIME

year11 = FILTER data2 BY (year == '2011') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED')) AND full\_time\_position =='N';

GROUP11 = group year11 by (job\_title,year,full\_time\_position);

AVG1 = foreach GROUP11 generate group, (double)AVG(year11.prevailing\_wage) as count;

RESULT\_11\_N = order AVG1 by $1 desc;

year12 = FILTER data2 BY (year == '2012') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='N';

GROUP12 = group year12 by (job\_title,year,full\_time\_position);

AVG2 = foreach GROUP12 generate group, (double)AVG(year12.prevailing\_wage) ;

RESULT\_12\_N = order AVG2 by $1 desc;

year13 = FILTER data2 BY (year == '2013') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='N';

GROUP13 = group year13 by (job\_title,year,full\_time\_position);

AVG3 = foreach GROUP13 generate group, (double)AVG(year13.prevailing\_wage) ;

RESULT\_13\_N = order AVG3 by $1 desc;

year14 = FILTER data2 BY (year == '2014') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='N';

GROUP14 = group year14 by (job\_title,year,full\_time\_position);

AVG4 = foreach GROUP14 generate group, (double)AVG(year14.prevailing\_wage) ;

RESULT\_14\_N= order AVG4 by $1 desc;

year15 = FILTER data2 BY (year == '2015') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='N';

GROUP15 = group year15 by (job\_title,year,full\_time\_position);

AVG5 = foreach GROUP15 generate group, (double)AVG(year15.prevailing\_wage) ;

RESULT\_15\_N= order AVG5 by $1 desc;

year16 = FILTER data2 BY (year == '2016') AND ((case\_status =='CERTIFIED-WITHDRAWN') OR( case\_status == 'CERTIFIED'))AND full\_time\_position =='N';

GROUP16 = group year16 by (job\_title,year,full\_time\_position);

AVG6 = foreach GROUP16 generate group, (double)AVG(year16.prevailing\_wage) ;

RESULT\_16\_N= order AVG6 by $1 desc;

FINAL\_N = UNION RESULT\_11\_N,RESULT\_12\_N,RESULT\_13\_N,RESULT\_14\_N,RESULT\_15\_N,RESULT\_16\_N;

STORE FINAL\_N INTO 'pig\_out/prob\_8\_N' USING PigStorage();

**Output:**

JAPANESE TEACHING ASSISTANT 2013 N 17076.0

INSTRUCTOR, PSYCHOLOGY 2013 N 6972.0

ASSISTANT VOLLEYBALL COACH 2013 N 16889.0

(DIRECTOR OF RELIGIOUS ACTIVITIES AND EDUCATION,2013 N 16764.0

COSTUMER 2013 N 16640.0

ASSISTANT WOMEN'S SOCCER COACH 2013 N 16120.0

BILINGUAL SPECIAL EDUCATION TEACHER ASSISTANT 2013 N 15080.0

**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) ?**

**Technology Used: Map reduce.**

**Sucessrate.java**

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.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.\*;

public class sucessrate {

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

{

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

{

try{

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

String employer\_name = str[2];

String case\_status = str[1];

context.write(new Text(employer\_name),new Text(case\_status));

}

catch(Exception e) {

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

}

}

}

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

{

TreeMap<Double,String>topMap=new TreeMap<Double,String>();

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

double total=0;

double sucessrate=0;

double petition=0;

//double rate=0;

int certified\_withdrawn=0;

int certified=0;

String mykey=key.toString();

for(Text v:values)

{

petition++;

String case\_status=v.toString();

if(case\_status.equals("CERTIFIED") )

{

certified++;

}

else if ( case\_status.equals("CERTIFIED-WITHDRAWN"))

{

certified\_withdrawn++;

}

}

if(petition >=1000)

{

total= certified + certified\_withdrawn;

sucessrate=(total/petition)\*100;

}

if (sucessrate >70.00)

{

String finaloutput=mykey+"\t"+petition+"\t"+sucessrate;

topMap.put(sucessrate,finaloutput);

}

}

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

for (String val:topMap.descendingMap().values())

{

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

}

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(sucessrate.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:**

**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 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)?**

**Technology Used: Map reduce.**

**Sucessrate\_job.java**

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.Reducer;

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

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

public class sucessrate\_job {public static class MapClass extends Mapper<LongWritable,Text,Text,Text>

{

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

{

try{

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

String job\_title = str[4];

String case\_status = str[1];

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

}

catch(Exception e) {

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

}

}

}

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

{

TreeMap<Double,String>topMap=new TreeMap<Double,String>();

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

double total=0;

double sucessrate=0;

double petition=0;

//double rate=0;

int certified\_withdrawn=0;

int certified=0;

//String finaloutput=null;

String mykey=key.toString();

for(Text v:values)

{

petition++;

String case\_status=v.toString();

if(case\_status.equals("CERTIFIED") )

{

certified++;

}

else if ( case\_status.equals("CERTIFIED-WITHDRAWN"))

{

certified\_withdrawn++;

}

}

if(petition >=1000)

{

total= certified + certified\_withdrawn;

sucessrate=(total/petition)\*100;

}

if (sucessrate >70.00)

{

String finaloutput=mykey+"\t"+petition+"\t"+sucessrate;

topMap.put(sucessrate,finaloutput);

}

}

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

for(String val:topMap.descendingMap().values())

{

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

}

}

}

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

Configuration conf = new Configuration();

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

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

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

job.setJarByClass(sucessrate\_job.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 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**

1.mysql –u root –p

2. create database h1b;

3. use h1b;

4.CREATE TABLE success\_rate(job\_title varchar(100)NOT NULL,total\_no\_of\_appl INT NOT NULL,certifiedANDcertified\_withdrwan\_count INT NOT NULL,

5.desc success\_rate;

+---------------------------------------+--------------+------+-----+---------+-------+

| field | type | null | key | default | extra |

+---------------------------------------+--------------+------+-----+---------+-------+

| job\_title | varchar(100) | no | | null | |

| total\_no\_of\_appl | int(11) | no | | null | |

| certifiedandcertified\_withdrwan\_count | int(11) | no | | null | |

| success\_rate | float | no | | null | |

+---------------------------------------+--------------+------+-----+---------+-------+

6.Start sqoop…..connect to msql>h1b database

sqoop list-tables --connect jdbc:mysql://localhost/h1b --username root --password 'hduser'

7. export data from hdfs to msql >success\_rate table

sqoop export --connect jdbc:mysql://localhost/h1b --username 'root' --password 'hduser' --table success\_rate --export-dir /problem10 --input-fields-terminated-by '\t' --mysql-delimiters -m 1;

+------------------------------------------------------------+------------------+---------------------------------------+--------------+

| PRODUCTION SUPPORT LEAD - US | 1301 | 1301 | 100 |

| ASSOCIATE CONSULTANT - US | 4393 | 4390 | 99.9317 |

| SYSTEMS ENGINEER - US | 10036 | 10026 | 99.9004 |

| TEST ENGINEER - US | 2198 | 2195 | 99.8635 |

| PRODUCTION SUPPORT ANALYST - US | 1451 | 1449 | 99.8622 |

| TEST ANALYST - US | 4958 | 4949 | 99.8185 |

| CONSULTANT - US | 7426 | 7412 | 99.8115 |

| TECHNOLOGY LEAD - US | 28350 | 28294 | 99.8025 |

| TECHNICAL TEST LEAD - US | 5374 | 5363 | 99.7953 |

| SENIOR TECHNOLOGY ARCHITECT - US | 1417 | 1414 | 99.7883 |

| TECHNOLOGY ARCHITECT - US | 4707 | 4696 | 99.7663 |

| TECHNOLOGY ANALYST - US | 26055 | 25993 | 99.762 |

| SENIOR PROJECT MANAGER - US | 2774 | 2767 | 99.7477 |

| DEVELOPER USER INTERFACE | 5247 | 5232 | 99.7141 |

| COMPUTER SYSTEMS ANALYST 2 | 4031 | 4019 | 99.7023 |

| SYSTEMS ANALYST - II | 1339 | 1335 | 99.7013 |

| PROJECT MANAGER - III | 1651 | 1646 | 99.6972 |

| PROJECT MANAGER - US | 7046 | 7024 | 99.6878 |

| PROGRAMMER ANALYST - II | 3588 | 3576 | 99.6656 |

| LEAD CONSULTANT - US | 3402 | 3390 | 99.6473 |

| COMPUTER SYSTEMS ANALYST 3 | 2170 | 2161 | 99.5853 |

| COMPUTER PROGRAMMER/CONFIGURER 2 | 6729 | 6700 | 99.569 |

| PROGRAMMER ANALYST - I | 1432 | 1425 | 99.5112 |

| SYSTEMS ANALYST - III | 1006 | 1001 | 99.503 |

| PRINCIPAL CONSULTANT - US | 1352 | 1345 | 99.4822 |

| COMPUTER SPECIALIST/TESTING AND QUALITY ANALYST 2 | 3998 | 3975 | 99.4247 |

| COMPUTER PROGRAMMER/CONFIGURER 3 | 1145 | 1138 | 99.3886 |

| COMPUTER SPECIALIST/SYSTEM SUPPORT AND DEVELOPMENT | 1339 | 1330 | 99.3279 |

| COMPUTER SPECIALIST/SYSTEM SUPPORT AND DEVELOPMENT ADMIN 2 | 1085 | 1077 | 99.2627 |

| DATA WAREHOUSE SPECIALIST | 1631 | 1618 | 99.2029 |

.

**Conclusion**

By using Mapreduce We can easily analyaze the complex data.

Hive is very efficient for analyze structure data.

For normal group by join and filter based data retrieval ,Pig is very efficient

Sqoop is very important to import and export the data.