**A project report on**

***Analysis of H1b Data using Hadoop Ecosystem***

**BY :**

**Supriya S. Sonawane**

**S171107500164**

**Pune Deccan**

**Abstract**

**What is big data?**

The big data refers to all the data that is being generated across the globe at an unprecedented rate. This data could be either structured or unstructured. This data can’t be processed using tradition computing techniques.

Hadoop is a complete eco-system of open source that provide the framework to deal with big data.

**Hadoop**

Hadoop is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment. It is part of the [Apache](http://searchcio-midmarket.techtarget.com/definition/Apache) project sponsored by the Apache Software Foundation.

**Hadoop history**

With an increase in the penetration of internet and the usage of the internet, the data captured by Google increased exponentially year on year. Just to give you an estimate of this number, in 2007 Google collected on an average 270 PB of data every month. The same number increased to 20000 PB every day in 2009. Obviously, Google needed a better platform to process such an enormous data. Google implemented a programming model called MapReduce, which could process this 20000 PB per day. Google ran these MapReduce operations on a special file system called Google File System (GFS). Sadly, GFS is not an open source.

Doug cutting and Yahoo! reverse engineered the model GFS and built a parallel Hadoop Distributed File System (HDFS). The software or framework that supports HDFS and MapReduce is known as Hadoop. Hadoop is an open source and distributed by Apache.

**Importance of Hadoop**

**1. Scalable***:* Hadoop is a highly scalable storage platform, because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel.

**2. Cost Effective:** It is cost effective because it is open source and file-system relies on commodity storage disks which are much less expensive. , the file-system shares the hardware with the computation framework as well.

**3. Flexible:** Hadoop enables you to easily access new data sources and processing of different kind of data like structured and unstructured.

**4. Fast processing of data :** the tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If you're dealing with large volumes of unstructured data, hadoop is able to efficiently process terabytes of data in just minutes,

**5. Fault tolerance :**a key advantage of using hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use

**Tools used in this project**

**MapReduce :**

Hadoop map-reduce is a software framework for easily writing applications which process vast amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

**Hive:**

*A*apache hive is an open-source data warehouse system for queryingand analyzing large datasets stored in hadoop.  It supports queries expressed in a language called HIVEQL, which automatically translates sql-like queries into mapreduce jobs executed on hadoop.

**PIG:**

Pig is a high level scripting language that is used with apache hadoop. Pig works with data from many sources, including structured and unstructured data. Pig scripts are translated into a series of mapreduce jobs that are run on the apache hadoop cluster.

**SQOOP*:***

SQOOP is used to import and export data from external data stores into hadoop distributed file system or related hadoop eco-systems like hive and Hbase. Similarly, sqoop can also be used to extract data from hadoop or its eco-systems

**Acknowledgement**

I wish to thank my master trainer Mr.SandeepAgarwal and my Tech mentor Mrs.Jyoti Mittal for providing complete learning on Big Data and Hadoop and guiding me in accomplishing the objectives of my project.

**OUTLINE:**

1. **Title :** H1B data analysis using Hadoop Ecosystem
2. **Input** :H1B data
3. **Data elements:** Sr.no, case status, employer name, soc name, job position, full time position,prevailing wage, year, worksite, longitude, latitude.
4. **Aim:** To perform certain analysis on H1b data. For future planning.

**TASK 1.**

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

**Technology Used: HIVE**

INSERT OVERWRITE DIRECTORY '/problem1b' ROW FORMAT delimited fields terminated by ',' select year, count(\*) from h1b\_final where job\_title = 'DATA ENGINEER' group by year order by year asc;

**output:**

**year count**

2011, 18

2012, 32

2013, 41

2014, 89

2015, 160

2016, 251

**TASK 1.**

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

**Technology used : 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 '/home/hduser/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 case\_status,job\_title,year;

year1 = FILTER data2 BY year == '2011';

group1 = group year1 by (job\_title);

Count1 = foreach group1 generate group, COUNT(year1.job\_title);

--dump Count1;

year2 = FILTER data2 BY year == '2012';

group2 = group year2 by (job\_title);

Count2 = foreach group2 generate group, COUNT(year2.job\_title);

year3 = FILTER data2 BY year == '2013';

group3 = group year3 by (job\_title);

Count3 = foreach group3 generate group, COUNT(year3.job\_title);

year4 = FILTER data2 BY year == '2014';

group4 = group year4 by (job\_title);

Count4 = foreach group4 generate group, COUNT (year4.job\_title);

year5 = FILTER data2 BY year == '2015';

group5 = group year5 by (job\_title);

Count5 = foreach group5 generate group, COUNT (year5.job\_title);

year6 = FILTER data2 BY year == '2016';

group6 = group year6 by (job\_title);

Count6 = foreach group6 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;

data3 = foreach JOIN1 GENERATE $0,$1,$3,$5,$7,$9,$11;

average1 = FOREACH data3 GENERATE $0, (DOUBLE)((($2-$1)\*100)/$1+(($3-$2)\*100)/$2+(($4-$3)\*100)/$3+(($5-$4)\*100)/$4+(($6-$5)\*100)/$5);

RESULT1 = FOREACH average1 GENERATE $0, (DOUBLE)($1/5);

AVERAGE\_DESC\_1= LIMIT (ORDER RESULT1 BY $1 DESC) 5;

STORE AVERAGE\_DESC\_1 INTO 'pig\_out/prob\_1b' USING PigStorage();

**OUTPUT:**

JOB\_TITLE AVERAGE

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

**Task2**

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

**Technology Used :HIVE**

select worksite, year,count(worksite) as count from h1b\_final WHERE job\_title =='DATA ENGINEER' AND year=='2011' group by year,worksite order by count desc limit 1;

select worksite, year,count(worksite) as count from h1b\_final WHERE job\_title =='DATA ENGINEER' AND year=='2012' group by year,worksite order by count desc limit 1;

select worksite, year,count(worksite) as count from h1b\_final WHERE job\_title =='DATA ENGINEER' AND year=='2013' group by year,worksite order by count desc limit 1;

select worksite, year,count(worksite) as count from h1b\_final WHERE job\_title =='DATA ENGINEER' AND year=='2014' group by year,worksite order by count desc limit 1;

select worksite, year,count(worksite) as count from h1b\_final WHERE job\_title =='DATA ENGINEER' AND year=='2015' group by year,worksite order by count desc limit 1;

select worksite, year,count(worksite) as count from h1b\_final WHERE job\_title =='DATA ENGINEER' AND year=='2016' group by year,worksite order by count desc limit 1;

**OUTPUT:**

WORKSITE year count

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

**Task 2.**

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

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

select worksite, year,count(worksite) as count from h1b\_final where case\_status =='CERTIFIED' AND year =='2012'group by year,worksite order by count desc limit 5;

select worksite, year,count(worksite) as count from h1b\_final where case\_status =='CERTIFIED' AND year =='2013'group by year,worksite order by count desc limit 5;

select worksite, year,count(worksite) as count from h1b\_final where case\_status =='CERTIFIED' AND year =='2014'group by year,worksite order by count desc limit 5;

select worksite, year,count(worksite) as count from h1b\_final where case\_status =='CERTIFIED' AND year =='2015'group by year,worksite order by count desc limit 5;

select worksite, year,count(worksite) as count from h1b\_final where case\_status =='CERTIFIED' AND year =='2016'group by year,worksite order by count desc limit 5;

**OUTPUT:**

1. **FOR 2011**

**WORKSITE YEAR COUNT**

NEW YORK, NEW YORK 2011 23172

HOUSTON, TEXAS 2011 8184

CHICAGO, ILLINOIS 2011 5188

SAN JOSE, CALIFORNIA 2011 4713

SAN FRANCISCO, CALIFORNIA 2011 4711

1. **FOR 2012**

**WORKSITE YEAR COUNT**

NEW YORK, NEW YORK 2012 23737

HOUSTON, TEXAS 2012 9963

SAN FRANCISCO, CALIFORNIA 2012 6116

CHICAGO, ILLINOIS 2012 5671

ATLANTA, GEORGIA 2012 5565

1. **FOR 2013**

**WORKSITE YEAR COUNT**

NEW YORK, NEW YORK 2013 23537

HOUSTON, TEXAS 2013 11136

SAN FRANCISCO, CALIFORNIA 2013 7281

SAN JOSE, CALIFORNIA 2013 6722

ATLANTA, GEORGIA 2013 6377

**4) FOR 2014:**

**WORKSITE YEAR COUNT**

NEW YORK, NEW YORK 2014 27634

HOUSTON, TEXAS 2014 13360

SAN FRANCISCO, CALIFORNIA 2014 9798

SAN JOSE, CALIFORNIA 2014 8223

ATLANTA, GEORGIA 2014 8213

**5) FOR 2015:**

**WORKSITE YEAR COUNT**

NEW YORK, NEW YORK 2015 31266

HOUSTON, TEXAS 2015 15242

SAN FRANCISCO, CALIFORNIA 2015 12594

ATLANTA, GEORGIA 2015 10500

SAN JOSE, CALIFORNIA 2015 9589

**6) FOR 2016**

**WORKSITE YEAR COUNT**

NEW YORK, NEW YORK 2016 34639

SAN FRANCISCO, CALIFORNIA 2016 13836

HOUSTON, TEXAS 2016 13655

ATLANTA, GEORGIA 2016 11678

CHICAGO, ILLINOIS 2016 11064

**Task 3.**

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

**Technology Used : HIVE**

INSERT OVERWRITE DIRECTORY '/problem3' ROW FORMAT delimited fields terminated by ',' select soc\_name, count(\*) as final from h1b\_final where job\_title == 'DATA SCIENTIST' AND case\_status == 'CERTIFIED' group by soc\_name order by final desc limit 1;

**OUTPUT:**

STATISTICIANS 369

**Task4**

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

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

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

data1 = LOAD '/home/hduser/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 employer\_name, year;

--dump data2;

year1 = FILTER data2 BY year == '2011';

group1 = group year1 by (employer\_name,year);

Count1 = foreach group1 generate group, COUNT (year1.employer\_name);

Desc1 = limit (order Count1 by $1 desc) 5;

year2 = FILTER data2 BY year == '2012';

group2 = group year2 by (employer\_name,year);

Count2 = foreach group2 generate group, COUNT (year2.employer\_name);

Desc2 = limit (order Count2 by $1 desc) 5;

year3 = FILTER data2 BY year == '2013';

group3 = group year3 by (employer\_name,year);

Count3 = foreach group3 generate group, COUNT (year3.employer\_name);

Desc3 = limit (order Count3 by $1 desc) 5;

year4 = FILTER data2 BY year == '2014';

group4 = group year4 by (employer\_name,year);

Count4 = foreach group4 generate group, COUNT(year4.employer\_name);

Desc4 = limit (order Count4 by $1 desc) 5;

year5 = FILTER data2 BY year == '2015';

group5 = group year5 by (employer\_name,year);

Count5 = foreach group5 generate group, COUNT (year5.employer\_name);

Desc5 = limit (order Count5 by $1 desc) 5;

year6 = FILTER data2 BY year == '2016';

group6 = group year6 by (employer\_name,year);

Count6 = foreach group6 generate group, COUNT (year6.employer\_name);

Desc6 = limit (order Count6 by $1 desc) 5;

result = UNION Desc1, Desc2, Desc3, Desc4, Desc5, Desc6;

STORE result INTO 'pig\_out/prob4' USING PigStorage ()**;**

**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, 2016) 25352

(CAPGEMINI AMERICA INC, 2016) 16725

(TATA CONSULTANCY SERVICES LIMITED, 2016) 13134

(WIPRO LIMITED, 2016) 10607

(IBM INDIA PRIVATE LIMITED, 2016) 9787

(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, 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

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

**--a) for all the applications**

**Technology Used : 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 '/home/hduser/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;

year11 = FILTER data2 BY year == '2011';

group1 = group year11 by (job\_title, year);

Count1 = foreach group1 generate group, COUNT (year11.job\_title);

Desc1 = limit (order Count1 by $1 desc) 10;

year12 = FILTER data2 BY year == '2012';

group2 = group year12 by (job\_title, year);

Count2 = foreach group2 generate group, COUNT (year12.job\_title);

Desc2 = limit (order Count2 by $1 desc) 10;

year13 = FILTER data2 BY year == '2013';

group3 = group year13 by (job\_title, year);

Count3 = foreach group3 generate group, COUNT (year13.job\_title);

Desc3 = limit (order Count3 by $1 desc) 10;

year14 = FILTER data2 BY year == '2014';

group4 = group year14 by (job\_title, year);

Count4 = foreach group4 generate group, COUNT (year14.job\_title);

Desc4 = limit (order Count4 by $1 desc) 10;

year15 = FILTER data2 BY year== '2015';

group5 = group year15 by (job\_title, year);

Count5 = foreach group5 generate group, COUNT(year15.job\_title);

Desc5 = limit (order Count5 by $1 desc) 10;

year16 = FILTER data2 BY year == '2016';

group6 = group year16 by (job\_title, year);

Count6 = foreach group6 generate group, COUNT (year16.job\_title);

Desc6 = limit (order Count6 by $1 desc) 10;

RESULT = UNION Desc1,Desc2,Desc3,Desc4,Desc5,Desc6;

STORE RESULT INTO 'pig\_out/prob5a' USING PigStorage();

**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

(SENIOR CONSULTANT, 2012) 3737

(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

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

**b) for only certified applications**

**Technology Used : 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 '/home/hduser/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, case\_status;

year11 = FILTER data2 BY year == '2011' AND case\_status =='CERTIFIED';

group1 = group year11 by (job\_title,year);

Count1 = foreach group1 generate group, COUNT (year11.job\_title);

Desc1 = limit (order Count1 by $1 desc) 10;

year12 = FILTER data2 BY year == '2012' AND case\_status =='CERTIFIED';

group2 = group year12 by (job\_title,year);

Count2 = foreach group2 generate group, COUNT(year12.job\_title);

Desc2 = limit (order Count2 by $1 desc) 10;

year13 = FILTER data2 BY year == '2013' AND case\_status =='CERTIFIED';

group3 = group year13 by (job\_title,year);

Count3 = foreach group3 generate group, COUNT(year13.job\_title);

Desc3 = limit (order Count3 by $1 desc) 10;

year14 = FILTER data2 BY year == '2014' AND case\_status =='CERTIFIED';

group4 = group year14 by (job\_title,year);

Count4 = foreach group4 generate group, COUNT(year14.job\_title);

Desc4 = limit (order Count4 by $1 desc) 10;

year15 = FILTER data2 BY year == '2015' AND case\_status =='CERTIFIED';

group5 = group year15 by (job\_title,year);

Count5 = foreach group5 generate group, COUNT(year15.$0);

Desc5 = limit (order Count5 by $1 desc) 10;

year16 = FILTER data2 BY year == '2016' AND case\_status =='CERTIFIED';

group6 = group year16 by (job\_title,year);

Count6 = foreach group6 generate group, COUNT(year16.job\_title);

Desc6 = limit (order Count6 by $1 desc) 10;

RESULT = UNION Desc1,Desc2 ,Desc3, Desc4, Desc5, Desc6;

STORE RESULT INTO 'pig\_out/prob5b/out' USING PigStorage();

**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

(SENIOR CONSULTANT, 2013) 3996

(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

**Task.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 :MapReduce**

package problem6;

import java.io.\*;

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 percentage\_avg {

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 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 {

long totalcount= 0,certified\_count=0,certified\_withdrawn\_count=0,denied\_count=0,withdrawn\_count=0;

double certified\_AvgPerc=0,certified\_withdrawn\_AvgPerc=0,denied\_AvgPerc=0,

withdrawn\_AvgPerc=0;

for (Text T : values)

{

totalcount++;

String case\_status=T.toString();

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

{

certified\_count++;

}

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

{

certified\_withdrawn\_count++;

}

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

{

withdrawn\_count++;

}

else

{

denied\_count++;

}

}

certified\_AvgPerc = ((double)certified\_count/(double)totalcount)\*100;

certified\_withdrawn\_AvgPerc = ((double)certified\_withdrawn\_count/(double)totalcount)\*100;

withdrawn\_AvgPerc = ((double)withdrawn\_count/(double)totalcount)\*100;

denied\_AvgPerc = ((double)denied\_count/(double)totalcount)\*100;

String COUNT=totalcount+"\t"+certified\_count+"\t"+certified\_AvgPerc+"\t"+certified\_withdrawn\_count+"\t" +certified\_withdrawn\_AvgPerc+"\t"+withdrawn\_count+"\t"+withdrawn\_AvgPerc+"\t"+denied\_count+"\t"+ denied\_AvgPerc;

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

}

}

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

Configuration conf = new Configuration();

Job job = new Job(conf, "H1B DATA");

job.setJarByClass(percentage\_avg.class);

job.setMapperClass(MapClass.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setReducerClass(ReduceClass.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 : output format**

**[year total certified % cert\_w %**  **Withdrawn % denied % ]**

2011 358767 307936 85.83 11596 3.2321813321738064

10105 2.816591269542628 29130 8.119475871526644

2012 415607 352668 84.85612609989725 31118 7.487361858678993 10725 2.5805628875355806 21096 5.0759491538881685

2013 442114 382951 86.61815730784367 35432 8.014222576077664 11590 2.621495813297023 12141 2.7461243027816358

2014 519427 455144 87.62424748809747 36350 6.99809597883822 16034 3.086863024063054 11899 2.2907935090012645

2015 618727 547278 88.45225761927313 41071 6.637984119005635 19455 3.144359305477214 10923 1.7653989562440302

2016 647803 569646 87.93506667922193 47092 7.269493966529948 21890 3.3791137120389996 9175 1.4163256422091284

**Task7)**

**Create a bar graph to depict the number of applications for each year [All]**

**Technology Used : MapReduce**

import java.io.\*;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

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

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

public class Application\_count {

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

{

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

{

try{

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

long year = Long.parseLong(str[7]);

long s\_no = Long.parseLong(str[0]);

context.write(new LongWritable(year),new LongWritable(s\_no));

}

catch(Exception e)

{

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

}

}

}

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

{

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

int count= 0;

for (LongWritable T : values)

{

count++;

}

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

}

}

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

Configuration conf = new Configuration();

Job job = new Job(conf, "H1b DATA");

job.setJarByClass(Application\_count.class);

job.setMapperClass(MapClass.class);

job.setMapOutputKeyClass(LongWritable.class);

job.setMapOutputValueClass(LongWritable.class);

job.setReducerClass(ReduceClass.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(IntWritable.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

**Task 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**

--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.]

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

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

data1 = LOAD '/home/hduser/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 : FULL TIME POSITION**

(JOB\_TITLE ,YEAR.FULL\_TIME\_POSITION) AVERAGE

**FOR 2011**

(AREA MANAGER, PHARMACEUTICAL PACKAGING,2011,Y) 2.1298784E8

(DEVELOPER (SOFTWARE SYSTEMS APPLICATIONS),2011,Y) 1.765608E8

(SYSTEMS ENGINEER (DIAGNOSTICS),2011,Y) 9.552608E7

(SENIOR COST CONSULTANT,2011,Y) 8.561904E7

(RADIATION ONCOLOGIST,2011,Y) 7.660264025E7

**FOR 2012**

(QA COORDINATOR,2012,Y) 2.36785424E7

(SAS PROGRAMMER,2012,Y) 1253141.9454545456

(LECTURER IN HORTICULTURE,2012,Y) 693360.0

(CARDIOLOGIST / ELECTROPHYSIOLOGIST,2012,Y) 475780.0

(TEACHER, DEAF & HARD OF HEARING,2012,Y) 473773.0

**FOR 2013**

(STAFF CONSULTANT - MICRO,2013,Y) 1.6950752E8

(QA ANALYST/ PROGRAMMER,2013,Y) 4.6735804E7

(SOFTWARE PROJ. MGR./ARCHITECT,2013,Y) 891072.0

(CLINICAL FELLOW, MINIMALLY INVASIVE SURGERY,2013,Y) 590913.0

(PEDIATRIC NEUROSURGEON,2013,Y) 401472.0

**FOR 2014**

(GASTROENTEROLOGIST PHYSICIAN,2014,Y) 631920.0

(PHYSICIAN/NEUROSURGEON,2014,Y) 523713.0

(MEDICAL ONCOLOGY PHYSICIAN,2014,Y) 483052.0

(PHYSICIAN CARDIOLOGIST,2014,Y) 467771.0

(PHYSICIAN (CARDIOLOGY/ CARDIAC ELECTROPHYSIOLOGY),2014,Y) 448100.0

**FOR 2015**

(MANAGER, GEORGIAN, CAUCASUS, AND EASTERN EUROPE REGIONAL MAN,2015,Y) 1.2308608E8

(ENGINEERING QUALITY ANALYST (15-1199.0),2015,Y) 9.993984E7

(SR. MANAGER, SOX & INTERNAL AUDIT GROUP,2015,Y) 9.06598055E7

(CHIEF EXECUTIVE OFFICER (CEO),2015,Y) 453870.5

(PHYSICIAN (GENERAL SURGERY),2015,Y) 356900.0

**FOR 2016**

(SYSTEMS ANALYSTS,2016,Y) 4216025.545454546

(CARDIOLOGIST/INTERVENTIONAL CARDIOLOGIST,2016,Y) 350000.0

(CARDIOLOGIST PHYSICIAN,2016,Y) 337800.0

(TRAUMA & GENERAL SURGEON,2016,Y) 328972.0

(MEDICAL ONCOLOGIST AND MEDICAL DIRECTOR,2016,Y) 292138.0

**Task 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 : Mapreduce**

package problem9;

import java.io.\*;

import java.util.TreeMap;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

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

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

public class employer\_success\_rate {

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

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

long totalcount= 0,certified\_count=0,certified\_withdrawn\_count=0 ,Total=0;

double emp\_SuccessRate=0;

String mykey = key.toString();

for (Text T : values)

{

totalcount++;

String case\_status=T.toString();

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

{

certified\_count++;

}

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

{

certified\_withdrawn\_count++;

}

}

if(totalcount>=1000)

{

Total = certified\_count+certified\_withdrawn\_count;

emp\_SuccessRate =((double)Total/(double)totalcount)\*100;

}

if(emp\_SuccessRate>=70)

{

String OUTPUT = mykey+","+totalcount+","+Total+","+emp\_SuccessRate;

MapOut.put(emp\_SuccessRate,OUTPUT);

}

}

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

for(String val : MapOut.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, "H1B DATA");

job.setJarByClass(employer\_success\_rate.class);

job.setMapperClass(MapClass.class);

job.setReducerClass(ReduceClass.class);

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 :**

**OUTPUT fromat**

**(EMPLOYER\_NAME, totalCOUNT ,CERTIFIED+CERTIFIED\_WITHDRAWN, SUCCESS\_RATES)**

HTC GLOBAL SERVICES, INC.,1164,1164,100.0

INFOSYS LIMITED,130592,129992,99.5405537858368

DIASPARK, INC.,1419,1412,99.50669485553206

ACCENTURE LLP,33447,33244,99.393069632553

TECH MAHINDRA (AMERICAS),INC.,10732,10661,99.33842713380544

TATA CONSULTANCY SERVICES LIMITED,64726,64297,99.33720606865866

YASH TECHNOLOGIES, INC.,2214,2198,99.2773261065944

YASH & LUJAN CONSULTING, INC.,1372,1362,99.27113702623906

HCL AMERICA, INC.,22678,22512,99.26801305229738

RELIABLE SOFTWARE RESOURCES, INC.,1992,1975,99.14658634538152

NTT DATA, INC.,4611,4571,99.13250921708958

ERP ANALYSTS, INC.,1785,1769,99.10364145658264

PATNI AMERICAS INC.,3149,3120,99.07907272149889

MINDTREE LIMITED,4067,4029,99.06565035652815

KFORCE INC.,1596,1581,99.06015037593986

TECH MAHINDRA ( AMERICAS), INC,1170,1159,99.05982905982907

GRANDISON MANAGEMENT, INC.,1386,1372,98.98989898989899

GENPACT LLC,1046,1034,98.8527724665392

SMARTPLAY, INC.,1377,1361,98.83805374001452

SYNTEL CONSULTING INC.,3167,3130,98.83170192611304

CREDIT SUISSE SECURITIES (USA) LLC,2546,2516,98.8216810683425

MASTECH, INC., A MASTECH HOLDINGS, INC. COMPANY,5228,5166,98.814078041316

GENESIS ELDERCARE REHABILITATION SERVICES, INC.,1320,1304,98.7878787878788

HORIZON TECHNOLOGIES INC,1731,1710,98.78682842287695

SYNTEL INC,1946,1922,98.7667009249743

THE BOSTON CONSULTING GROUP, INC.,1352,1335,98.74260355029585

**Task10)**

**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 : MapReduce**

package problem10;

import java.io.\*;

import java.util.TreeMap;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

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

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

public class JobTitle\_SuccessRate {

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

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

long totalcount= 0,certified\_count=0,certified\_withdrawn\_count=0 ,Total=0;

double job\_SuccesRate=0;

String newkey = key.toString();

for (Text T : values)

{

totalcount++;

String case\_status=T.toString();

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

{

certified\_count++;

}

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

{

certified\_withdrawn\_count++;

}

}

if(totalcount>=1000)

{

Total = certified\_count+certified\_withdrawn\_count;

job\_SuccesRate =((double)Total/(double)totalcount)\*100;

}

if( job\_SuccesRate>=70)

{

String OUTPUT = newkey+","+totalcount+","+Total+","+ job\_SuccesRate;

Out.put( job\_SuccesRate,OUTPUT);

}

}

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

for(String val : Out.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, "H1B DATA");

job.setJarByClass(JobTitle\_SuccessRate.class);

job.setMapperClass(MapClass.class);

job.setReducerClass(ReduceClass.class);

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 :**

**Output format**

JOB\_TITLE,TOTALCUNT,CERTIFIED\_CERTIFIED\_WITHDRAWN,SUCCESS\_RATE

PRODUCTION SUPPORT LEAD - US,1301,1301,100.0

ASSOCIATE CONSULTANT - US,4393,4390,99.9317095379012

SYSTEMS ENGINEER - US,10036,10026,99.90035870864887

TEST ENGINEER - US,2198,2195,99.86351228389445

PRODUCTION SUPPORT ANALYST - US,1451,1449,99.86216402481047

TEST ANALYST - US,4958,4949,99.81847519160952

CONSULTANT - US,7426,7412,99.81147320226232

TECHNOLOGY LEAD - US,28350,28294,99.80246913580247

TECHNICAL TEST LEAD - US,5374,5363,99.7953107554894

SENIOR TECHNOLOGY ARCHITECT - US,1417,1414,99.78828510938604

TECHNOLOGY ARCHITECT - US,4707,4696,99.76630550244316

TECHNOLOGY ANALYST - US,26055,25993,99.7620418345807

SENIOR PROJECT MANAGER - US,2774,2767,99.74765681326603

DEVELOPER USER INTERFACE,5247,5232,99.71412235563179

COMPUTER SYSTEMS ANALYST 2,4031,4019,99.70230711982138

SYSTEMS ANALYST - II,1339,1335,99.70126960418223

PROJECT MANAGER - III,1651,1646,99.69715324046032

PROJECT MANAGER - US,7046,7024,99.68776610843032

PROGRAMMER ANALYST - II,3588,3576,99.66555183946488

LEAD CONSULTANT - US,3402,3390,99.64726631393297

COMPUTER SYSTEMS ANALYST 3,2170,2161,99.5852534562212

COMPUTER PROGRAMMER/CONFIGURER 2,6729,6700,99.56902957348788

PROGRAMMER ANALYST - I,1432,1425,99.51117318435753

SYSTEMS ANALYST - III,1006,1001,99.50298210735586

PRINCIPAL CONSULTANT - US,1352,1345,99.48224852071006

COMPUTER SPECIALIST/TESTING AND QUALITY ANALYST

**Task11)**

**Export result for question no 10 to MySql database.**

**Technology used :sqoop**

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 ',' --mysql-delimiters -m 1**;**

**OUTPUT :**

**mysql> select \*from success\_rate;**

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

| job\_title | total\_no\_of\_appl | certifiedANDcertified\_withdrwan\_count | Success\_rate |

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

| 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:**

1. SENIOR SYSTEMS ANALYST JC60 Have highest average growth in applications.
2. Petitions for DATA ENGINEER job position is increased over time.
3. STATISTICIAN Industry have most number data scientist.
4. Number of application are increased each year.
5. HTC GLOBAL SERVICES INC employer have max success rate than other employers.
6. Success rate is max for PRODUCTION SUPPORT LEAD – US.