*Presented by:*

*Mr. Arvind Govind Pednekar*

*Student Id: 171107500112*

*Pune Deccan Centre*

*Analysing H1B data*

*Using Hadoop ecosystem*

**Abstract**

What is Big Data?

Big data is a term that describes the large volume of data – both structured, unstructured and semi-structured data that overrun a business on a day-to-day basis. But it’s not the amount of data that’s important. It’s what organizations do with the data that matters. Big data can be analysed for insights that lead to better decisions and strategic business moves.

Hadoop

Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

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.

**Why Hadoop is important:**

* **Ability to store and process huge amounts of any kind of data, quickly.** With data volumes and varieties constantly increasing, especially from social media and the Internet of Things (IoT), that's a key consideration.
* **Computing power.** Hadoop's distributed computing model processes big data fast. The more computing nodes you use, the more processing power you have.
* **Fault tolerance.** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.
* **Flexibility.** Unlike traditional relational databases, you don’t have to pre-process data before storing it. You can store as much data as you want and decide how to use it later. That includes unstructured data like text, images and videos.
* **Low cost.** The open-source framework is free and uses commodity hardware to store large quantities of data.
* **Scalability.** You can easily grow your system to handle more data simply by adding nodes. Little administration is required.

**Hadoop Glossary**

**Hadoop Common** – the libraries and utilities used by other Hadoop modules.

**Hadoop Distributed File System (HDFS)**– the Java-based scalable system that stores data across multiple machines without prior organization.

**YARN** – (Yet Another Resource Negotiator) provides resource management for the processes running on Hadoop.

**MapReduce** – a parallel processing software framework. It is comprised of two steps. Map step is a master node that takes inputs and partitions them into smaller subproblems and then distributes them to worker nodes. After the map step has taken place, the master node takes the answers to all of the subproblems and combines them to produce output.

**Hadoop ecosystem components used in this project**

**Hive:**

A data warehousing and SQL-like query language that presents data in the form of tables. Hive programming is similar to database programming.

**Pig:**

A platform for manipulating data stored in HDFS that includes a compiler for MapReduce programs and a high-level language called Pig Latin. It provides a way to perform data extractions, transformations and loading, and basic analysis without having to write MapReduce programs.

**Sqoop:**

A connection and transfer mechanism that moves data between Hadoop and relational databases.

**Zookeeper:**

An application that coordinates distributed processing.

**Oozie:**

A Hadoop job scheduler.

**Acknowledgement**

I wish to thank my master trainer Mr. Sandeep Agarwal 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.

**Project Outline**

|  |  |
| --- | --- |
| Title | Big Data Analysis in Hadoop on H1B Data |
| Inputs | H1b Data |
| Data Elements | Sr.no, case status, employer name, soc name, job position, full time position, prevailing wage, year, worksite, longitude, latitude. |
| Purpose | To provide analyzed report to H1B Sponsers to help them to make plans for minimum wage provision and to get skilled workers from respective worksite. |
| Methodology | Agile |

**Project Implementation**

**Assumption:**

1. Hadoop cluster is running.

2. Ecosystem products (Pig, hive, Sqoop) are installed.

3. H1b data is available on HDFS in CSV format.

**Prerequisite for all the jobs:**

H1B data is in CSV format hence need to be converted into TextFile format in Hadoop FileSystem.

**Steps for conversion:**

1. Create a table in hive to read entire record as as on single row.
2. Create a table to convert and store a entire record into different tab separated fields which will create a file on Hadoop fileSystem in TextFile format.

**Note:** Look up supporting tables are created based on analysis.

**Create a block size of 64 MB on HDFS**

**Steps for changing block size:**

**A.On cloudera platform:**

1. First login through root :sudo su

Edit the following file

/etc/hadoop/conf/hdfs-site.xml

Set following properties

<property>

<name>dfs.blocksize</name>

<value>67108864</value>

</property>

2.Upload any file on HDFS and check ensure block size of 64 MB

**H1b analysis:**

**Task 1:**

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

**Technology Used: Map reduce**

Map Reduce program:

package problem1a;

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.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 DataEnggJobGrowth

{

public static class DataEnggMapper extends Mapper<LongWritable,Text,NullWritable,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 year=str[7];

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

{

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

}

}

catch(Exception e)

{

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

}

}

}

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

{

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

{

int count11=0,count12=0,count13=0,count14=0,count15=0,count16=0;

double cycle1=0,cycle2=0,cycle3=0,cycle4=0,cycle5=0,average=0;

for(Text val:values)

{

String year=val.toString();

if(year.equals("2011"))

{

count11++;

}

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

{

count12++;

}

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

{

count13++;

}

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

{

count14++;

}

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

{

count15++;

}

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

{

count16++;

}

}

if(count11!=0)

{

cycle1=(double)(count12-count11)\*100/(double)count11;

}

else

{

cycle1=0;

}

if(count12!=0)

{

cycle2=(double)(count13-count12)\*100/(double)count12;

}

else

{

cycle2=0;

}

if(count13!=0)

{

cycle3=(double)(count14-count13)\*100/(double)count13;

}

else

{

cycle3=0;

}

if(count14!=0)

{

cycle4=(double)(count15-count14)\*100/(double)count14;

}

else

{

cycle4=0;

}

if(count15!=0)

{

cycle5=(double)(count16-count15)\*100/(double)count15;

}

else

{

cycle5=0;

}

average=(cycle1+cycle2+cycle3+cycle4+cycle5)/5;

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

String total11=String.format("%d", count11);

String total12=String.format("%d", count12);

String total13=String.format("%d", count13);

String total14=String.format("%d", count14);

String total15=String.format("%d", count15);

String total16=String.format("%d", count16);

String myvalue =total11+"\t"+total12+"\t"+total13+"\t"+total14+"\t"+total15+"\t"+total16+"\t"+myaverage;

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

}

}

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

{

Configuration conf= new Configuration();

Job job=Job.getInstance(conf);

job.setJarByClass(DataEnggJobGrowth.class);

job.setMapperClass(DataEnggMapper.class);

job.setReducerClass(DataEnggReducer.class);

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:**

**[cloudera@quickstart project]$ hadoop jar prob1a.jar /project/h1b\_final /project/prob1a**

17/04/28 11:18:00 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

17/04/28 11:18:01 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

17/04/28 11:18:01 INFO input.FileInputFormat: Total input paths to process : 2

17/04/28 11:18:01 INFO mapreduce.JobSubmitter: number of splits:7

17/04/28 11:18:02 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1493397054698\_0004

17/04/28 11:18:02 INFO impl.YarnClientImpl: Submitted application application\_1493397054698\_0004

17/04/28 11:18:02 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application\_1493397054698\_0004/

17/04/28 11:18:02 INFO mapreduce.Job: Running job: job\_1493397054698\_0004

17/04/28 11:18:12 INFO mapreduce.Job: Job job\_1493397054698\_0004 running in uber mode : false

17/04/28 11:18:12 INFO mapreduce.Job: map 0% reduce 0%

17/04/28 11:18:53 INFO mapreduce.Job: map 3% reduce 0%

17/04/28 11:18:54 INFO mapreduce.Job: map 7% reduce 0%

17/04/28 11:18:55 INFO mapreduce.Job: map 11% reduce 0%

17/04/28 11:18:56 INFO mapreduce.Job: map 16% reduce 0%

17/04/28 11:18:57 INFO mapreduce.Job: map 24% reduce 0%

17/04/28 11:18:58 INFO mapreduce.Job: map 30% reduce 0%

17/04/28 11:18:59 INFO mapreduce.Job: map 35% reduce 0%

17/04/28 11:19:00 INFO mapreduce.Job: map 39% reduce 0%

17/04/28 11:19:01 INFO mapreduce.Job: map 66% reduce 0%

17/04/28 11:19:02 INFO mapreduce.Job: map 86% reduce 0%

17/04/28 11:19:15 INFO mapreduce.Job: map 100% reduce 0%

17/04/28 11:19:17 INFO mapreduce.Job: map 100% reduce 100%

17/04/28 11:19:17 INFO mapreduce.Job: Job job\_1493397054698\_0004 completed successfully

17/04/28 11:19:18 INFO mapreduce.Job: Counters: 50

File System Counters

FILE: Number of bytes read=12053

FILE: Number of bytes written=995207

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=449878993

HDFS: Number of bytes written=28

HDFS: Number of read operations=24

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Killed map tasks=1

Launched map tasks=8

Launched reduce tasks=1

Data-local map tasks=8

Total time spent by all maps in occupied slots (ms)=287584

Total time spent by all reduces in occupied slots (ms)=13477

Total time spent by all map tasks (ms)=287584

Total time spent by all reduce tasks (ms)=13477

Total vcore-seconds taken by all map tasks=287584

Total vcore-seconds taken by all reduce tasks=13477

Total megabyte-seconds taken by all map tasks=294486016

Total megabyte-seconds taken by all reduce tasks=13800448

Map-Reduce Framework

Map input records=3002446

Map output records=1721

Map output bytes=8605

Map output materialized bytes=12089

Input split bytes=861

Combine input records=0

Combine output records=0

Reduce input groups=1

Reduce shuffle bytes=12089

Reduce input records=1721

Reduce output records=1

Spilled Records=3442

Shuffled Maps =7

Failed Shuffles=0

Merged Map outputs=7

GC time elapsed (ms)=3152

CPU time spent (ms)=18110

Physical memory (bytes) snapshot=1819971584

Virtual memory (bytes) snapshot=12008919040

Total committed heap usage (bytes)=1582329856

Shuffle Errors

BAD\_ID=0

CONNECTION=0

IO\_ERROR=0

WRONG\_LENGTH=0

WRONG\_MAP=0

WRONG\_REDUCE=0

File Input Format Counters

Bytes Read=449878132

File Output Format Counters

Bytes Written=28

[cloudera@quickstart project]$ hadoop fs -cat /project/prob1a/p\*

Output:

2011 2012 2013 2014 2015 2016 average\_growth

60 81 151 249 394 786 68.81

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | Average growth % |
| 60 | 81 | 151 | 249 | 394 | 786 | 68.81 |

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

**Technology Used: MapReduce**

MapReduce Program:JobGrowth.java

package problem1b;

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 JobGrowth

{

public static class GrowthMapper 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 year=str[7];

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

}

catch(Exception e)

{

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

}

}

}

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

{

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

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

{

int count11=0,count12=0,count13=0,count14=0,count15=0,count16=0;

double cycle1=0,cycle2=0,cycle3=0,cycle4=0,cycle5=0,average=0;

for(Text val:values)

{

String year=val.toString();

if(year.equals("2011"))

{

count11++;

}

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

{

count12++;

}

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

{

count13++;

}

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

{

count14++;

}

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

{

count15++;

}

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

{

count16++;

}

}

if(count11!=0)

{

cycle1=(double)(count12-count11)\*100/(double)count11;

}

else

{

cycle1=0;

}

if(count12!=0)

{

cycle2=(double)(count13-count12)\*100/(double)count12;

}

else

{

cycle2=0;

}

if(count13!=0)

{

cycle3=(double)(count14-count13)\*100/(double)count13;

}

else

{

cycle3=0;

}

if(count14!=0)

{

cycle4=(double)(count15-count14)\*100/(double)count14;

}

else

{

cycle4=0;

}

if(count15!=0)

{

cycle5=(double)(count16-count15)\*100/(double)count15;

}

else

{

cycle5=0;

}

average=(cycle1+cycle2+cycle3+cycle4+cycle5)/5;

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

String total11=String.format("%d", count11);

String total12=String.format("%d", count12);

String total13=String.format("%d", count13);

String total14=String.format("%d", count14);

String total15=String.format("%d", count15);

String total16=String.format("%d", count16);

String mykey=key.toString();

String myvalue =mykey+"\t"+total11+"\t"+total12+"\t"+total13+"\t"+total14+"\t"+total15+"\t"+total16+"\t"+myaverage;

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=Job.getInstance(conf);

job.setJarByClass(JobGrowth.class);

job.setMapperClass(GrowthMapper.class);

job.setReducerClass(GrowthReducer.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:

Job\_title cnt11 cnt12 cnt13 cnt14 cnt15 cnt16 avg hrowth

[cloudera@quickstart project]$ hadoop fs -cat /project/prob1b/p\*

BUSINESS ANALYST 2 1 0 2 2 1 249 4930.00

SENIOR SYSTEMS ANALYST JC60 2 425 393 515 606 1128 4255.46

PROGRAMMER/ DEVELOPER 0 0 0 0 1 209 4160.00

BUSINESS SYSTEMS ANALYST 2 0 0 3 9 3 594 3966.67

SOFTWARE DEVELOPER 2 5 1 177 29 26 20 3480.59

**Task 2:**

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

**Technology used: Pig**

Pig script: problem2a.pig

h1bvisa = load '/user/hive/warehouse/h1b\_project.db/h1b\_final/' using PigStorage() as (sr\_no:chararray,case\_satus:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:double,year:chararray,work\_site:chararray,longitude:chararray,latitude:chararray);

data = foreach h1bvisa generate $4,$7,$8;

--data: {job\_title: chararray,year: chararray,work\_site: chararray}

year11 =filter data by year=='2011';

year12 =filter data by year=='2012';

year13 =filter data by year=='2013';

year14 =filter data by year=='2014';

year15 =filter data by year=='2015';

year16 =filter data by year=='2016';

/\*

now make logic to find which part of us has most data engineer jobs for each year?

\*/

data\_engineer\_jobs11 = FILTER year11 by job\_title matches '.\*DATA ENGINEER.\*';

grouped\_data11 = group data\_engineer\_jobs11 by work\_site;

--describe grouped\_data;

count\_jobs11 = foreach grouped\_data11 generate group as area,COUNT(data\_engineer\_jobs11) as no\_of\_job;

final11 = order count\_jobs11 by no\_of\_job desc;

store final11 into '/project/prob2a/2011' using PigStorage();

data\_engineer\_jobs12 = FILTER year12 by job\_title matches '.\*DATA ENGINEER.\*';

grouped\_data12 = group data\_engineer\_jobs12 by work\_site;

count\_jobs12 = foreach grouped\_data12 generate group as area,COUNT(data\_engineer\_jobs12) as no\_of\_job;

final12 = order count\_jobs12 by no\_of\_job desc;

store final12 into '/project/prob2a/2012' using PigStorage();

data\_engineer\_jobs13 = FILTER year13 by job\_title matches '.\*DATA ENGINEER.\*';

grouped\_data13 = group data\_engineer\_jobs13 by work\_site;

count\_jobs13 = foreach grouped\_data13 generate group as area,COUNT(data\_engineer\_jobs13) as no\_of\_job;

final13 = order count\_jobs13 by no\_of\_job desc;

store final13 into '/project/prob2a/2013' using PigStorage();

data\_engineer\_jobs14 = FILTER year14 by job\_title matches '.\*DATA ENGINEER.\*';

grouped\_data14 = group data\_engineer\_jobs14 by work\_site;

count\_jobs14 = foreach grouped\_data14 generate group as area,COUNT(data\_engineer\_jobs14) as no\_of\_job;

final14 = order count\_jobs14 by no\_of\_job desc;

store final14 into '/project/prob2a/2014' using PigStorage();

data\_engineer\_jobs15 = FILTER year15 by job\_title matches '.\*DATA ENGINEER.\*';

grouped\_data15 = group data\_engineer\_jobs15 by work\_site;

count\_jobs15 = foreach grouped\_data15 generate group as area,COUNT(data\_engineer\_jobs15) as no\_of\_job;

final15 = order count\_jobs15 by no\_of\_job desc;

store final15 into '/project/prob2a/2015' using PigStorage();

data\_engineer\_jobs16 = FILTER year16 by job\_title matches '.\*DATA ENGINEER.\*';

grouped\_data16 = group data\_engineer\_jobs16 by work\_site;

count\_jobs16 = foreach grouped\_data16 generate group as area,COUNT(data\_engineer\_jobs16) as no\_of\_job;

final16 = order count\_jobs16 by no\_of\_job desc;

store final16 into '/project/prob2a/2016' using PigStorage();

**Output:**

for year 2011:--

SEATTLE, WASHINGTON 20

for year 2012:--

SEATTLE, WASHINGTON 30

for year 2013:--

SEATTLE, WASHINGTON 46

for year 2014:--

SEATTLE, WASHINGTON 45

for year 2015:--

SEATTLE, WASHINGTON 61

for year 2016:--

SEATTLE, WASHINGTON 128

**Task:2**

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

**Technology Used: Pig**

Pig script: problem2b.pig

Command: pig problem2b.pig

h1bvisa = load '/user/hive/warehouse/project.db/h1b\_final/' using PigStorage() as (sr\_no:chararray,case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:double,year:chararray,work\_site:chararray,longitude:chararray,latitude:chararray);

data = foreach h1bvisa generate $1,$7,$8;

--data: {case\_status: chararray,year: chararray,work\_site: chararray}

certified\_data = filter data by case\_status=='CERTIFIED';

year11 =filter certified\_data by year=='2011';

year12 =filter certified\_data by year=='2012';

year13 =filter certified\_data by year=='2013';

year14 =filter certified\_data by year=='2014';

year15 =filter certified\_data by year=='2015';

year16 =filter certified\_data by year=='2016';

/\*

now make logic to find which part of us has got most certified visa for each year?

\*/

grouped\_data11 = group year11 by work\_site;

--describe grouped\_data11;

count\_jobs11 = foreach grouped\_data11 generate group as area,COUNT(year11) as no\_of\_job;

final11 = order count\_jobs11 by no\_of\_job desc;

result11 = limit final11 5;

store result11 into '/project/prob2b/2011' using PigStorage();

grouped\_data12 = group year12 by work\_site;

--describe grouped\_data12;

count\_jobs12 = foreach grouped\_data12 generate group as area,COUNT(year12) as no\_of\_job;

final12 = order count\_jobs12 by no\_of\_job desc;

result12 = limit final12 5;

store result12 into '/project/prob2b/2012' using PigStorage();

grouped\_data13 = group year13 by work\_site;

--describe grouped\_data13;

count\_jobs13 = foreach grouped\_data13 generate group as area,COUNT(year13) as no\_of\_job;

final13 = order count\_jobs13 by no\_of\_job desc;

result13 = limit final13 5;

store result13 into '/project/prob2b/2013' using PigStorage();

grouped\_data14 = group year14 by work\_site;

--describe grouped\_data14;

count\_jobs14 = foreach grouped\_data14 generate group as area,COUNT(year14) as no\_of\_job;

final14 = order count\_jobs14 by no\_of\_job desc;

result14 = limit final14 5;

store result14 into '/project/prob2b/2014' using PigStorage();

grouped\_data15 = group year15 by work\_site;

--describe grouped\_data15;

count\_jobs15 = foreach grouped\_data15 generate group as area,COUNT(year15) as no\_of\_job;

final15 = order count\_jobs15 by no\_of\_job desc;

result15 = limit final15 5;

store result15 into '/project/prob2b/2015' using PigStorage();

grouped\_data16 = group year16 by work\_site;

--describe grouped\_data16;

count\_jobs16 = foreach grouped\_data16 generate group as area,COUNT(year16) as no\_of\_job;

final16 = order count\_jobs16 by no\_of\_job desc;

result16 = limit final16 5;

store result16 into '/project/prob2b/2016' using PigStorage();

**Output:**

for year 2011:--

NEW YORK, NEW YORK 23172

HOUSTON, TEXAS 8184

CHICAGO, ILLINOIS 5188

SAN JOSE, CALIFORNIA 4713

SAN FRANCISCO, CALIFORNIA 4711

for year 2012:---

NEW YORK, NEW YORK 23736

HOUSTON, TEXAS 9963

SAN FRANCISCO, CALIFORNIA 6116

CHICAGO, ILLINOIS 5671

ATLANTA, GEORGIA 5565

For year 2013:----

NEW YORK, NEW YORK 23537

HOUSTON, TEXAS 11136

SAN FRANCISCO, CALIFORNIA 7281

SAN JOSE, CALIFORNIA 6722

ATLANTA, GEORGIA 6377

For year 2014:----

NEW YORK, NEW YORK 27634

HOUSTON, TEXAS 13360

SAN FRANCISCO, CALIFORNIA 9798

SAN JOSE, CALIFORNIA 8223

ATLANTA, GEORGIA 8213

For year 2015:----

NEW YORK, NEW YORK 31266

HOUSTON, TEXAS 15242

SAN FRANCISCO, CALIFORNIA 12594

ATLANTA, GEORGIA 10500

SAN JOSE, CALIFORNIA 9589

For year 2016:---

NEW YORK, NEW YORK 34639

SAN FRANCISCO, CALIFORNIA 13836

HOUSTON, TEXAS 13655

ATLANTA, GEORGIA 11678

CHICAGO, ILLINOIS 11064

**Task 3:**

**Which industry has the most number of Data Scientist positions?**

**Technology Used: Hive**

Hive file:problem3.sql

Command: hive –f problem3.sql

Use h1b\_project;

INSERT OVERWRITE DIRECTORY '/project/prob3' ROW FORMAT delimited fields terminated by ',' select soc\_name,count(soc\_name) as cnt from h1b\_final where job\_title LIKE '%DATA SCIENTIST%' group by soc\_name order by cnt desc;

**Output:**

INDUSTRY no\_of\_jobs:

STATISTICIANS 649

COMPUTER AND INFORMATION RESEARCH SCIENTISTS 500

OPERATIONS RESEARCH ANALYSTS 426

Computer and Information Research Scientists 208

COMPUTER OCCUPATIONS, ALL OTHER 179

Statisticians 152

SOFTWARE DEVELOPERS, APPLICATIONS 148

MATHEMATICIANS 147

COMPUTER SYSTEMS ANALYSTS 135

Operations Research Analysts 124

SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE 61

Software Developers, Applications 59

Computer Occupations, All Other 45

Computer Systems Analysts 30

BIOLOGICAL SCIENTISTS, ALL OTHER 27

Software Developers, Systems Software 22

Database Administrators 17

MARKET RESEARCH ANALYSTS AND MARKETING SPECIALISTS 15

FINANCIAL SPECIALISTS, ALL OTHER 15

Mathematicians 13

**Task 4:**

**Which top 5 employers file the most petitions each year?**

**Technology Used: Hive**

Hive file:problem4.sql

Command: hive –f problem4.sql

use h1b\_project;

/\*

create table h1b\_final\_partition (s\_no int,case\_status string,employer\_name string,soc\_name string,job\_title string,full\_time\_position string,prevailing\_wage int,

worksite string,longitude double,latitude double) partitioned by(year string)

ROW FORMAT delimited fields terminated by '\t' stored as textfile;

from h1b\_final INSERT OVERWRITE table h1b\_final\_partition PARTITION(year) select s\_no,case\_status,employer\_name,soc\_name,job\_title,

full\_time\_position,prevailing\_wage,worksite,longitude,latitude,year DISTRIBUTE By year;

create view top10employers as select employer\_name,year,count(\*) as total from h1b\_final\_partition group by year,employer\_name

sort by year,total desc;

\*/

INSERT OVERWRITE DIRECTORY 'project/prob4' ROW FORMAT delimited fields terminated by '\t' select year,employer\_name,total,rank from(select year,employer\_name,

rank() over (partition by year order by total desc) as rank,total from top10employers) ranked\_table where ranked\_table.rank <=5;

**Output:**

YEAR JOB\_TITLE COUNT RANK

2011 TATA CONSULTANCY SERVICES LIMITED 5416 1

2011 MICROSOFT CORPORATION 4253 2

2011 DELOITTE CONSULTING LLP 3621 3

2011 WIPRO LIMITED 3028 4

2011 COGNIZANT TECHNOLOGY SOLUTIONS U.S. CORPORATION 2721 5

2012 INFOSYS LIMITED 15818 1

2012 WIPRO LIMITED 7182 2

2012 TATA CONSULTANCY SERVICES LIMITED 6735 3

2012 DELOITTE CONSULTING LLP 4727 4

2012 IBM INDIA PRIVATE LIMITED 4074 5

2013 INFOSYS LIMITED 32223 1

2013 TATA CONSULTANCY SERVICES LIMITED 8790 2

2013 WIPRO LIMITED 6734 3

2013 DELOITTE CONSULTING LLP 6124 4

2013 ACCENTURE LLP 4994 5

2014 INFOSYS LIMITED 23759 1

2014 TATA CONSULTANCY SERVICES LIMITED 14098 2

2014 WIPRO LIMITED 8365 3

2014 DELOITTE CONSULTING LLP 7017 4

2014 ACCENTURE LLP 5498 5

2015 INFOSYS LIMITED 33245 1

2015 TATA CONSULTANCY SERVICES LIMITED 16553 2

2015 WIPRO LIMITED 12201 3

2015 IBM INDIA PRIVATE LIMITED 10693 4

2015 ACCENTURE LLP 9605 5

2016 INFOSYS LIMITED 25352 1

2016 CAPGEMINI AMERICA INC 16725 2

2016 TATA CONSULTANCY SERVICES LIMITED 13134 3

2016 WIPRO LIMITED 10607 4

2016 IBM INDIA PRIVATE LIMITED 9787 5

**Task 5:**

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

**Technology Used :Hive**

Hive file: problem5.sql

Command: hive –f problem5.sql

use h1b\_final;

/\*

create table h1b\_final\_partition (s\_no int,case\_status string,employer\_name string,soc\_name string,job\_title string,full\_time\_position string,prevailing\_wage int,

worksite string,longitute double,latitute double) partitioned by(year string)

ROW FORMAT delimited fields terminated by '\t' stored as textfile;

from h1b\_final INSERT OVERWRITE table h1b\_final\_partition PARTITION(year) select s\_no,case\_status,employer\_name,soc\_name,job\_title,

full\_time\_position,prevailing\_wage,worksite,longitute,latitute,year DISTRIBUTE By year;

create view top10jobs as select job\_title,year,count(\*) as total from h1b\_final\_partition group by year,job\_title

sort by year,total desc;

\*/

INSERT OVERWRITE DIRECTORY '/project/prob5' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' select year,job\_title,total,rank from(select year,job\_title,rank() over (partition by year order by total desc) as rank,total from top10jobs) ranked\_table

where ranked\_table.rank <=10;

**Output:**

YEAR JOB\_TITLE COUNT RANK

2011 PROGRAMMER ANALYST 31799 1

2011 SOFTWARE ENGINEER 12763 2

2011 COMPUTER PROGRAMMER 8998 3

2011 SYSTEMS ANALYST 8644 4

2011 BUSINESS ANALYST 3891 5

2011 COMPUTER SYSTEMS ANALYST 3698 6

2011 ASSISTANT PROFESSOR 3467 7

2011 PHYSICAL THERAPIST 3377 8

2011 SENIOR SOFTWARE ENGINEER 2935 9

2011 SENIOR CONSULTANT 2798 10

2012 PROGRAMMER ANALYST 33066 1

2012 SOFTWARE ENGINEER 14437 2

2012 COMPUTER PROGRAMMER 9629 3

2012 SYSTEMS ANALYST 9296 4

2012 BUSINESS ANALYST 4752 5

2012 COMPUTER SYSTEMS ANALYST 4706 6

2012 SOFTWARE DEVELOPER 3895 7

2012 PHYSICAL THERAPIST 3871 8

2012 ASSISTANT PROFESSOR 3801 9

2012 SENIOR CONSULTANT 3737 10

2013 PROGRAMMER ANALYST 33880 1

2013 SOFTWARE ENGINEER 15680 2

2013 COMPUTER PROGRAMMER 11271 3

2013 SYSTEMS ANALYST 8714 4

2013 TECHNOLOGY LEAD - US 7853 5

2013 TECHNOLOGY ANALYST - US 7683 6

2013 BUSINESS ANALYST 5716 7

2013 COMPUTER SYSTEMS ANALYST 5043 8

2013 SOFTWARE DEVELOPER 5026 9

2013 SENIOR CONSULTANT 4326 10

2014 PROGRAMMER ANALYST 43114 1

2014 SOFTWARE ENGINEER 20500 2

2014 COMPUTER PROGRAMMER 14950 3

2014 SYSTEMS ANALYST 10194 4

2014 SOFTWARE DEVELOPER 7337 5

2014 BUSINESS ANALYST 7302 6

2014 COMPUTER SYSTEMS ANALYST 6821 7

2014 TECHNOLOGY LEAD - US 5057 8

2014 TECHNOLOGY ANALYST - US 4913 9

2014 SENIOR CONSULTANT 4898 10

2015 PROGRAMMER ANALYST 53436 1

2015 SOFTWARE ENGINEER 27259 2

2015 COMPUTER PROGRAMMER 14054 3

2015 SYSTEMS ANALYST 12803 4

2015 SOFTWARE DEVELOPER 10441 5

2015 BUSINESS ANALYST 8853 6

2015 TECHNOLOGY LEAD - US 8242 7

2015 COMPUTER SYSTEMS ANALYST 7918 8

2015 TECHNOLOGY ANALYST - US 7014 9

2015 SENIOR SOFTWARE ENGINEER 6013 10

2016 PROGRAMMER ANALYST 53743 1

2016 SOFTWARE ENGINEER 30668 2

2016 SOFTWARE DEVELOPER 14041 3

2016 SYSTEMS ANALYST 12314 4

2016 COMPUTER PROGRAMMER 11668 5

2016 BUSINESS ANALYST 9167 6

2016 COMPUTER SYSTEMS ANALYST 6900 7

2016 SENIOR SOFTWARE ENGINEER 6439 8

2016 DEVELOPER 6084 9

2016 TECHNOLOGY LEAD - US 5410 10

**Task 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: MapReduce**

MapReduce Program: GraphPattern.java

package problem6;

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 GraphPattern

{

public static class GraphMapper 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 GraphReducer extends Reducer<Text,Text,Text,Text>

{

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

{

int total\_application=0,certified=0,certified\_withdrawn=0,withdrawn=0,denied=0;

double certified\_perct=0,certified\_withdrawn\_perct=0,withdrawn\_perct,denied\_perct;

for(Text val:values)

{

total\_application++;

String case\_status=val.toString();

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

{

certified++;

}

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

{

certified\_withdrawn++;

}

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

{

withdrawn++;

}

else

{

denied++;

}

}

certified\_perct=((double)certified\*100)/(double)total\_application;

certified\_withdrawn\_perct=((double)certified\_withdrawn\*100)/(double)total\_application;

withdrawn\_perct=((double)withdrawn\*100)/(double)total\_application;

denied\_perct=((double)denied\*100)/(double)total\_application;

String myapplication=String.format("%d",total\_application);

String string\_certified\_perct=String.format("%.2f",certified\_perct);

String string\_certified\_withdrawn\_perct=String.format("%.2f",certified\_withdrawn\_perct);

String string\_withdrawn\_perct=String.format("%.2f",withdrawn\_perct);

String string\_denied\_perct=String.format("%.2f",denied\_perct);

String myvalue=myapplication+"\t"+certified+"\t"+string\_certified\_perct+"\t"+certified\_withdrawn+"\t"+string\_certified\_withdrawn\_perct+"\t"+withdrawn+"\t"+string\_withdrawn\_perct+"\t"+denied+"\t"+string\_denied\_perct;

//Print the value in the format [year,total\_app,certified\_app,certified %,certified\_withdrawn\_app,certified\_withdrawn %,withdrawn\_app,withdrawn %,denied\_app,denied %]

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

}

}

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

{

Configuration conf= new Configuration();

Job job=Job.getInstance(conf);

job.setJarByClass(GraphPattern.class);

job.setJobName("find the count and percentage of each case status for each year");

job.setMapperClass(GraphMapper.class);

job.setReducerClass(GraphReducer.class);

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:**

17/04/29 23:06:38 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

17/04/29 23:06:39 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

17/04/29 23:06:39 INFO input.FileInputFormat: Total input paths to process : 2

17/04/29 23:06:39 INFO mapreduce.JobSubmitter: number of splits:7

17/04/29 23:06:40 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1493531330246\_0001

17/04/29 23:06:41 INFO impl.YarnClientImpl: Submitted application application\_1493531330246\_0001

17/04/29 23:06:41 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application\_1493531330246\_0001/

17/04/29 23:06:41 INFO mapreduce.Job: Running job: job\_1493531330246\_0001

17/04/29 23:06:55 INFO mapreduce.Job: Job job\_1493531330246\_0001 running in uber mode : false

17/04/29 23:06:55 INFO mapreduce.Job: map 0% reduce 0%

17/04/29 23:09:21 INFO mapreduce.Job: map 2% reduce 0%

17/04/29 23:09:22 INFO mapreduce.Job: map 3% reduce 0%

17/04/29 23:09:25 INFO mapreduce.Job: map 4% reduce 0%

17/04/29 23:09:29 INFO mapreduce.Job: map 6% reduce 0%

17/04/29 23:09:31 INFO mapreduce.Job: map 7% reduce 0%

17/04/29 23:09:32 INFO mapreduce.Job: map 8% reduce 0%

17/04/29 23:09:35 INFO mapreduce.Job: map 10% reduce 0%

17/04/29 23:09:39 INFO mapreduce.Job: map 11% reduce 0%

17/04/29 23:09:42 INFO mapreduce.Job: map 12% reduce 0%

17/04/29 23:09:43 INFO mapreduce.Job: map 14% reduce 0%

17/04/29 23:09:44 INFO mapreduce.Job: map 15% reduce 0%

17/04/29 23:09:45 INFO mapreduce.Job: map 16% reduce 0%

17/04/29 23:09:47 INFO mapreduce.Job: map 18% reduce 0%

17/04/29 23:09:48 INFO mapreduce.Job: map 23% reduce 0%

17/04/29 23:09:49 INFO mapreduce.Job: map 26% reduce 0%

17/04/29 23:09:52 INFO mapreduce.Job: map 28% reduce 0%

17/04/29 23:09:54 INFO mapreduce.Job: map 30% reduce 0%

17/04/29 23:09:56 INFO mapreduce.Job: map 39% reduce 0%

17/04/29 23:09:57 INFO mapreduce.Job: map 41% reduce 0%

17/04/29 23:09:58 INFO mapreduce.Job: map 44% reduce 0%

17/04/29 23:10:00 INFO mapreduce.Job: map 48% reduce 0%

17/04/29 23:10:01 INFO mapreduce.Job: map 57% reduce 0%

17/04/29 23:10:02 INFO mapreduce.Job: map 68% reduce 0%

17/04/29 23:10:03 INFO mapreduce.Job: map 79% reduce 0%

17/04/29 23:10:04 INFO mapreduce.Job: map 86% reduce 0%

17/04/29 23:10:35 INFO mapreduce.Job: map 100% reduce 0%

17/04/29 23:10:38 INFO mapreduce.Job: map 100% reduce 67%

17/04/29 23:10:41 INFO mapreduce.Job: map 100% reduce 100%

17/04/29 23:10:41 INFO mapreduce.Job: Job job\_1493531330246\_0001 completed successfully

17/04/29 23:10:41 INFO mapreduce.Job: Counters: 50

File System Counters

FILE: Number of bytes read=52785734

FILE: Number of bytes written=106542897

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=449879154

HDFS: Number of bytes written=420

HDFS: Number of read operations=24

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Killed map tasks=1

Launched map tasks=8

Launched reduce tasks=1

Data-local map tasks=8

Total time spent by all maps in occupied slots (ms)=1136836

Total time spent by all reduces in occupied slots (ms)=35637

Total time spent by all map tasks (ms)=1136836

Total time spent by all reduce tasks (ms)=35637

Total vcore-seconds taken by all map tasks=1136836

Total vcore-seconds taken by all reduce tasks=35637

Total megabyte-seconds taken by all map tasks=1164120064

Total megabyte-seconds taken by all reduce tasks=36492288

Map-Reduce Framework

Map input records=3002446

Map output records=3002446

Map output bytes=46780836

Map output materialized bytes=52785770

Input split bytes=1022

Combine input records=0

Combine output records=0

Reduce input groups=8

Reduce shuffle bytes=52785770

Reduce input records=3002446

Reduce output records=8

Spilled Records=6004892

Shuffled Maps =7

Failed Shuffles=0

Merged Map outputs=7

GC time elapsed (ms)=16762

CPU time spent (ms)=85180

Physical memory (bytes) snapshot=1852166144

Virtual memory (bytes) snapshot=12008656896

Total committed heap usage (bytes)=1582329856

Shuffle Errors

BAD\_ID=0

CONNECTION=0

IO\_ERROR=0

WRONG\_LENGTH=0

WRONG\_MAP=0

WRONG\_REDUCE=0

File Input Format Counters

Bytes Read=449878132

File Output Format Counters

Bytes Written=420

**Output:**

Year t\_apps cert- % cer-w % with- % den- %

2011 358767 307936 85.83 11596 3.23 10105 2.82 29130 8.12

2012 415607 352668 84.86 31118 7.49 10725 2.58 21096 5.08

2013 442114 382951 86.62 35432 8.01 11590 2.62 12141 2.75

2014 519427 455144 87.62 36350 7.00 16034 3.09 11899 2.29

2015 618727 547278 88.45 41071 6.64 19455 3.14 10923 1.77

2016 647803 569646 87.94 47092 7.27 21890 3.38 9175 1.42

**Task 7:**

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

**Technology Used: Hive**

**Hive file: problem7.sql**

**Command: hive –f problem7.sql**

use h1b\_project;

/\*

LOAD DATA LOCAL INPATH '/home/hduser/h1b.csv' OVERWRITE INTO TABLE h1b\_application;

\*/

INSERT OVERWRITE DIRECTORY '/project/prob7' ROW FORMAT DELIMITED FIELDS TERMINATED by '\t' select year,count(\*) from h1b\_final group by year order by year;

**Output:**

**Year no of records**

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) ?**

Technology Used: Pig

Pig Script: problem8.pig

app = load '/user/hive/warehouse/h1b\_project.db/h1b\_final/' using PigStorage() as (sr\_no:int, case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:int,year:chararray,worksite:chararray,longitude:double, latitude:double);

year\_data11 = filter app by year=='2011';

year\_data12 = filter app by year=='2012';

year\_data13 = filter app by year=='2013';

year\_data14 = filter app by year=='2014';

year\_data15 = filter app by year=='2015';

year\_data16 = filter app by year=='2016';

data11 = foreach year\_data11 generate $4,$5,$6;

--describe data;

--data11: {job\_title: chararray,full\_time\_position: chararray,prevailing\_wage: double}

group\_data11 = group data11 by (job\_title,full\_time\_position);

--group\_data: {group: (job\_title: chararray,full\_time\_position: chararray),data: {(job\_title: chararray,full\_time\_position: chararray,prevailing\_wage: double)}}

average11 = foreach group\_data11 generate group as job,(double)AVG(data11.prevailing\_wage) as avg;

final11 = order average11 by avg desc;

store final11 into '/project/prob8/2011' using PigStorage();

data12 = foreach year\_data12 generate $4,$5,$6;

group\_data12 = group data12 by (job\_title,full\_time\_position);

average12 = foreach group\_data12 generate group as job,AVG(data12.prevailing\_wage) as avg;

final12 = order average12 by avg desc;

store final12 into '/project/prob8/2012' using PigStorage();

data13 = foreach year\_data13 generate $4,$5,$6;

group\_data13 = group data13 by (job\_title,full\_time\_position);

average13 = foreach group\_data13 generate group as job,AVG(data13.prevailing\_wage) as avg;

final13 = order average13 by avg desc;

store final13 into '/project/prob8/2013' using PigStorage();

data14 = foreach year\_data14 generate $4,$5,$6;

group\_data14 = group data14 by (job\_title,full\_time\_position);

average14 = foreach group\_data14 generate group as job,AVG(data14.prevailing\_wage) as avg;

final14 = order average14 by avg desc;

store final14 into '/project/prob8/2014' using PigStorage();

data15 = foreach year\_data15 generate $4,$5,$6;

group\_data15 = group data15 by (job\_title,full\_time\_position);

average15 = foreach group\_data15 generate group as job,AVG(data15.prevailing\_wage) as avg;

final15 = order average15 by avg desc;

store final15 into '/project/prob8/2015' using PigStorage();

data16 = foreach year\_data16 generate $4,$5,$6;

group\_data16 = group data16 by (job\_title,full\_time\_position);

average16 = foreach group\_data16 generate group as job,AVG(data16.prevailing\_wage) as avg;

final16 = order average16 by avg desc;

store final16 into '/project/prob8/2016' using PigStorage();

**Output:**

Output files 2011,2012,2013,2014,2015,2016

OutputFormat: (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

**Task 9:**

**Which are the top 10 employer which have the highest success rate more than 70% in petitions and file petitions more than 10000 ?**

Technology Used: Pig

Pig Script: problem9.pig

app = load '/user/hive/warehouse/h1b\_project.db/h1b\_final/' using PigStorage() as (sr\_no:int, case\_status:chararray, employer\_name:chararray,soc\_name:chararray, job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:int,year:chararray,worksite:chararray, longitude:double, latitude:double);

data = foreach app generate $2,$1;

validdata = filter data by case\_status=='CERTIFIED' or case\_status=='CERTIFIED-WITHDRAWN';

datagroup = group data by employer\_name;

validgroup = group validdata by employer\_name;

totalapp = foreach datagroup generate group as employer,COUNT(data) as no;

validapp = foreach validgroup generate group as employer,COUNT(validdata) as no;

joindata = join totalapp by $0,validapp by $0;

mydata = foreach joindata generate $0,$1,$3;

final = foreach mydata generate $0,$1,((double)$2\*100/(double)$1) as success;

result = filter final by $1>=10000 and $2>70.0;

final\_result = order result by $2 desc;

myresult =limit final\_result 10;

store myresult into '/project/prob9\_pig' using PigStorage();

**Output:**

Employer Name applications percentage of applications (%)

INFOSYS LIMITED 130592 99.5405537858368

ACCENTURE LLP 33447 99.393069632553

TECH MAHINDRA (AMERICAS),INC. 10732 99.33842713380544

TATA CONSULTANCY SERVICES LIMITED 64726 99.33720606865866

HCL AMERICA, INC. 22678 99.26801305229738

DELOITTE CONSULTING LLP 36742 98.32888792118013

WIPRO LIMITED 48117 98.28958580127606

MICROSOFT CORPORATION 25576 98.09196121363779

ERNST & YOUNG U.S. LLP 18232 98.0528740675735

CAPGEMINI AMERICA INC 16725 97.95515695067265

**Task 10:**

**Which are the top 10 job positions which have the highest success rate more than 70% in petitions and file petitions more than 10000?**

Technology Used: MapReduce

MapReduce Program: TopSuccess.java

package problem10;

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 TopSuccess

{

public static class SuccessMapper 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 case\_status=str[1];

String job\_title=str[4];

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

}

catch(Exception e)

{

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

}

}

}

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

{

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

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

{

double success\_rate=0;

int petitions=0,valid\_app=0;

int certified=0,certified\_withdrawn=0;

String myvalue=null;

String mykey=key.toString();

for(Text val:values)

{

petitions++;

String case\_status=val.toString();

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

{

certified++;

}

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

{

certified\_withdrawn++;

}

}

if(petitions>10000)

{

valid\_app=certified+certified\_withdrawn;

success\_rate=(double)valid\_app\*100/(double)petitions;

String mysuccessRate=String.format("%.2f",success\_rate);

String mypetitions=String.format("%.d", petitions);

myvalue=mykey+"\t"+mypetitions+"\t"+mysuccessRate;

}

if(success\_rate >70.00)

{

topMap.put(new Double(success\_rate),new Text(myvalue));

if(topMap.size() >10)

{

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=Job.getInstance(conf);

job.setJarByClass(TopSuccess.class);

job.setMapperClass(SuccessMapper.class);

job.setReducerClass(SuccessReducer.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:**

**[cloudera@quickstart project]$ hadoop jar prob10.jar /project/h1b\_final /project/prob10**

17/04/28 10:34:39 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

17/04/28 10:34:40 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

17/04/28 10:34:40 INFO input.FileInputFormat: Total input paths to process : 2

17/04/28 10:34:41 INFO mapreduce.JobSubmitter: number of splits:7

17/04/28 10:34:41 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1493397054698\_0002

17/04/28 10:34:41 INFO impl.YarnClientImpl: Submitted application application\_1493397054698\_0002

17/04/28 10:34:41 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application\_1493397054698\_0002/

17/04/28 10:34:41 INFO mapreduce.Job: Running job: job\_1493397054698\_0002

17/04/28 10:34:52 INFO mapreduce.Job: Job job\_1493397054698\_0002 running in uber mode : false

17/04/28 10:34:52 INFO mapreduce.Job: map 0% reduce 0%

17/04/28 10:35:28 INFO mapreduce.Job: map 3% reduce 0%

17/04/28 10:35:35 INFO mapreduce.Job: map 8% reduce 0%

17/04/28 10:35:36 INFO mapreduce.Job: map 11% reduce 0%

17/04/28 10:35:42 INFO mapreduce.Job: map 12% reduce 0%

17/04/28 10:35:50 INFO mapreduce.Job: map 14% reduce 0%

17/04/28 10:36:04 INFO mapreduce.Job: map 15% reduce 0%

17/04/28 10:36:10 INFO mapreduce.Job: map 16% reduce 0%

17/04/28 10:37:30 INFO mapreduce.Job: map 17% reduce 0%

17/04/28 10:37:31 INFO mapreduce.Job: map 44% reduce 0%

17/04/28 10:37:35 INFO mapreduce.Job: map 53% reduce 0%

17/04/28 10:37:37 INFO mapreduce.Job: map 57% reduce 0%

17/04/28 10:38:53 INFO mapreduce.Job: map 76% reduce 0%

17/04/28 10:39:06 INFO mapreduce.Job: map 86% reduce 0%

17/04/28 10:39:46 INFO mapreduce.Job: map 100% reduce 0%

17/04/28 10:39:50 INFO mapreduce.Job: map 100% reduce 68%

17/04/28 10:39:53 INFO mapreduce.Job: map 100% reduce 100%

17/04/28 10:39:54 INFO mapreduce.Job: Job job\_1493397054698\_0002 completed successfully

17/04/28 10:39:54 INFO mapreduce.Job: Counters: 50

File System Counters

FILE: Number of bytes read=106748417

FILE: Number of bytes written=214467743

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=449878993

HDFS: Number of bytes written=302

HDFS: Number of read operations=24

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Killed map tasks=1

Launched map tasks=8

Launched reduce tasks=1

Data-local map tasks=8

Total time spent by all maps in occupied slots (ms)=1504153

Total time spent by all reduces in occupied slots (ms)=41768

Total time spent by all map tasks (ms)=1504153

Total time spent by all reduce tasks (ms)=41768

Total vcore-seconds taken by all map tasks=1504153

Total vcore-seconds taken by all reduce tasks=41768

Total megabyte-seconds taken by all map tasks=1540252672

Total megabyte-seconds taken by all reduce tasks=42770432

Map-Reduce Framework

Map input records=3002446

Map output records=3002446

Map output bytes=100743519

Map output materialized bytes=106748453

Input split bytes=861

Combine input records=0

Combine output records=0

Reduce input groups=287550

Reduce shuffle bytes=106748453

Reduce input records=3002446

Reduce output records=10

Spilled Records=6004892

Shuffled Maps =7

Failed Shuffles=0

Merged Map outputs=7

GC time elapsed (ms)=70771

CPU time spent (ms)=60140

Physical memory (bytes) snapshot=1849171968

Virtual memory (bytes) snapshot=12009959424

Total committed heap usage (bytes)=1683791872

Shuffle Errors

BAD\_ID=0

CONNECTION=0

IO\_ERROR=0

WRONG\_LENGTH=0

WRONG\_MAP=0

WRONG\_REDUCE=0

File Input Format Counters

Bytes Read=449878132

File Output Format Counters

Bytes Written=302

Output

job\_title petitions success\_rate

 SYSTEMS ENGINEER - US 10036 99.84

TECHNOLOGY LEAD - US 28350 99.75

TECHNOLOGY ANALYST - US 26055 99.73

DEVELOPER 12909 95.67

COMPUTER SYSTEMS ENGINEER 11090 94.95

CONSULTANT 23081 92.97

SENIOR CONSULTANT 24904 92.33

LEAD ENGINEER 11157 91.37

COMPUTER PROGRAMMER 70570 90.72

SYSTEMS ANALYST 61965 89.96

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

**Technology Used: MySQL and Sqoop**

Steps to export data from hdfs into MySql database.

1.mysql –u root –p

Password$:cloudera

2.create database h1b;

3.use h1b;

4.create table problem10 (job\_title varchar(200) NOT NULL,applications bigint NOT NULL,success\_rate float NOT NULL) ;

5. mysql> describe problem10;

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

| Field | Type | Null | Key | Default | Extra |

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

| job\_title | varchar(200) | NO | | NULL | |

| applications | bigint(20) | NO | | NULL | |

| success\_rate | float | NO | | NULL | |

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

6. sqoop export --connect jdbc:mysql://localhost/h1b --username root --password cloudera --table problem10 --export-dir /project/prob10 --input-fields-terminated-by '\t' --mysql-delimiters -m 1;

[cloudera@quickstart ~]$ sqoop export --connect jdbc:mysql://localhost/h1b --username root --password cloudera --table problem10 --export-dir /project/prob10 --input-fields-terminated-by '\t' --mysql-delimiters -m 1;

Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.

Please set $ACCUMULO\_HOME to the root of your Accumulo installation.

17/05/02 01:03:40 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.10.0

17/05/02 01:03:40 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.

17/05/02 01:03:41 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.

17/05/02 01:03:41 INFO tool.CodeGenTool: Beginning code generation

17/05/02 01:03:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `problem10` AS t LIMIT 1

17/05/02 01:03:42 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `problem10` AS t LIMIT 1

17/05/02 01:03:42 INFO orm.CompilationManager: HADOOP\_MAPRED\_HOME is /usr/lib/hadoop-mapreduce

Note: /tmp/sqoop-cloudera/compile/c5097e84c01c036fbdcca9be9bd6b3f6/problem10.java uses or overrides a deprecated API.

Note: Recompile with -Xlint:deprecation for details.

17/05/02 01:03:48 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-cloudera/compile/c5097e84c01c036fbdcca9be9bd6b3f6/problem10.jar

17/05/02 01:03:48 INFO mapreduce.ExportJobBase: Beginning export of problem10

17/05/02 01:03:48 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address

17/05/02 01:03:48 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar

17/05/02 01:03:48 INFO Configuration.deprecation: mapred.map.max.attempts is deprecated. Instead, use mapreduce.map.maxattempts

17/05/02 01:03:51 INFO Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instead, use mapreduce.reduce.speculative

17/05/02 01:03:51 INFO Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead, use mapreduce.map.speculative

17/05/02 01:03:51 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps

17/05/02 01:03:51 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

17/05/02 01:04:05 INFO input.FileInputFormat: Total input paths to process : 1

17/05/02 01:04:05 INFO input.FileInputFormat: Total input paths to process : 1

17/05/02 01:04:05 INFO mapreduce.JobSubmitter: number of splits:1

17/05/02 01:04:05 INFO Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead, use mapreduce.map.speculative

17/05/02 01:04:06 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1493704540251\_0014

17/05/02 01:04:06 INFO impl.YarnClientImpl: Submitted application application\_1493704540251\_0014

17/05/02 01:04:07 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application\_1493704540251\_0014/

17/05/02 01:04:07 INFO mapreduce.Job: Running job: job\_1493704540251\_0014

17/05/02 01:04:36 INFO mapreduce.Job: Job job\_1493704540251\_0014 running in uber mode : false

17/05/02 01:04:36 INFO mapreduce.Job: map 0% reduce 0%

17/05/02 01:04:52 INFO mapreduce.Job: map 100% reduce 0%

17/05/02 01:04:52 INFO mapreduce.Job: Job job\_1493704540251\_0014 completed successfully

17/05/02 01:04:52 INFO mapreduce.Job: Counters: 30

File System Counters

FILE: Number of bytes read=0

FILE: Number of bytes written=146999

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=445

HDFS: Number of bytes written=0

HDFS: Number of read operations=4

HDFS: Number of large read operations=0

HDFS: Number of write operations=0

Job Counters

Launched map tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=12249

Total time spent by all reduces in occupied slots (ms)=0

Total time spent by all map tasks (ms)=12249

Total vcore-seconds taken by all map tasks=12249

Total megabyte-seconds taken by all map tasks=12542976

Map-Reduce Framework

Map input records=10

Map output records=10

Input split bytes=140

Spilled Records=0

Failed Shuffles=0

Merged Map outputs=0

GC time elapsed (ms)=449

CPU time spent (ms)=1120

Physical memory (bytes) snapshot=144302080

Virtual memory (bytes) snapshot=1503899648

Total committed heap usage (bytes)=106037248

File Input Format Counters

Bytes Read=0

File Output Format Counters

Bytes Written=0

17/05/02 01:04:52 INFO mapreduce.ExportJobBase: Transferred 445 bytes in 60.5575 seconds (7.3484 bytes/sec)

17/05/02 01:04:52 INFO mapreduce.ExportJobBase: Exported 10 records.

7. mysql> select \* from problem10;

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

| job\_title | applications | success\_rate |

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

| SYSTEMS ENGINEER - US | 10036 | 99.84 |

| TECHNOLOGY LEAD - US | 28350 | 99.75 |

| TECHNOLOGY ANALYST - US | 26055 | 99.73 |

| DEVELOPER | 12909 | 95.67 |

| COMPUTER SYSTEMS ENGINEER | 11090 | 94.95 |

| CONSULTANT | 23081 | 92.97 |

| SENIOR CONSULTANT | 24904 | 92.33 |

| LEAD ENGINEER | 11157 | 91.37 |

| COMPUTER PROGRAMMER | 70570 | 90.72 |

| SYSTEMS ANALYST | 61965 | 89.96 |

**Conclusion**

Following is the conclusion that we can draw based on the task performed by us:

1. Sqoop is Usefull when we have data on HDFS that need to be exported into the MySQL tables.
2. Hive helps in cleaning up data.CSV file can be easily converted into TSV Text file.
3. For normal group by join and filter based data retrieval ,Pig is very efficient.
4. MapReduce code written in java makes complex analysis quite easy. Codes required to be written to collects user inputs and performing complex join operations are handled efficiently using this approach.

**Webography**

**https://www.sas.com/en\_us/insights/big-data/hadoop.html**