**H1B VISA Analysis**

**Step 1. Create block size into 64 MB.**

Step 1. Cd /hadoop-2.7.1/etc/hadoop

step 2. /hadoop-2.7.1/etc/hadoop$ gedit hdfs-site.xml

step 3. set the blocksize(67108864(64\*1024\*1024)) in hdfs-site.xml

<property>

<name>dfs.blocksize</name>

<value>67108864</value>

</property>

step 4. uploaded data into hdfs by using hadoop (hadoop fs -put gateway- /niit/ )command to check the block size . Before default block size was 128 MB and now it got changed into 64 MB.

**Step 2. creating table and loading the data in HDFS using hive queries.**

hive> create database niit1;

hive>use niit1;

hive> show tables;

hive> CREATE TABLE h1b\_applications( sno int, case\_status string, employer\_name string, soc\_name string, job\_title string, full\_time\_position string, prevailining\_wage int, year string, worksite string, longitude double, latitude double) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' WITH SERDEPROPERTIES ("separatorChar"=",","quotesChar"="\"") STORED AS TEXTFILE;

OK

Time taken: 1.251 seconds

hive> LOAD DATA LOCAL INPATH '/home/hduser/H1B/h1b.csv' OVERWRITE INTO TABLE h1b\_applications;

Loading data to table default.h1b\_application

Table default.h1b\_application stats: [numFiles=1, numRows=0, totalSize=492258374, rawDataSize=0]

OK

Time taken: 22.823 seconds

hive> describe h1b\_applications;

OK

sno string from deserializer

case\_status string from deserializer

employer\_name string from deserializer

soc\_name string from deserializer

job\_title string from deserializer

full\_time\_position string from deserializer

prevailining\_wage string from deserializer

year string from deserializer

worksite string from deserializer

longitude string from deserializer

latitude string from deserializer

Time taken: 0.197 seconds, Fetched: 11 row(s)

→ 3002458 records are loaded.

hive> select COUNT(\*) from h1b\_applications;

3002458

Time taken: 133.059 seconds, Fetched: 1 row(s)

hive> select \* from h1b\_applications limit 5;

OK

1 CERTIFIED-WITHDRAWN UNIVERSITY OF MICHIGAN BIOCHEMISTS AND BIOPHYSICISTS POSTDOCTORAL RESEARCH FELLOW N 36067 2016 ANN ARBOR, MICHIGAN -83.7430378 42.2808256

2 CERTIFIED-WITHDRAWN GOODMAN NETWORKS, INC. CHIEF EXECUTIVES CHIEF OPERATING OFFICER Y 242674 2016 PLANO, TEXAS -96.6988856 33.0198431

3 CERTIFIED-WITHDRAWN PORTS AMERICA GROUP, INC. CHIEF EXECUTIVESCHIEF PROCESS OFFICER Y 193066 2016 JERSEY CITY, NEW JERSEY -74.0776417 40.7281575

4 CERTIFIED-WITHDRAWN GATES CORPORATION, A WHOLLY-OWNED SUBSIDIARY OF TOMKINS PLC CHIEF EXECUTIVES REGIONAL PRESIDEN, AMERICAS Y 220314 2016 DENVER, COLORADO -104.990251 39.7392358

5 WITHDRAWN PEABODY INVESTMENTS CORP. CHIEF EXECUTIVES PRESIDENT MONGOLIA AND INDIA Y 157518.4 2016 ST. LOUIS, MISSOURI -90.1994042 38.6270025

Time taken: 4.089 seconds, Fetched: 5 row(s)

hive>

hive> select distinct case\_status from h1b\_applications;

Total MapReduce CPU Time Spent: 31 seconds 990 msec

OK

CERTIFIED-WITHDRAWN

PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED

REJECTED

WITHDRAWN

CERTIFIED

DENIED

INVALIDATED

NA

Time taken: 70.051 seconds, Fetched: 8 row(s)

hive> select distinct year from h1b\_applications;

59 HDFS Write: 36 SUCCESS

Total MapReduce CPU Time Spent: 43 seconds 100 msec

OK

NULL

2011

2013

2015

2012

2014

2016

NA

Time taken: 140.497 seconds, Fetched: 8 row(s)

hive>

hive> select year, COUNT(case\_status) from h1b\_applications group by year;

Total MapReduce CPU Time Spent: 40 seconds 190 msec

OK

NULL 7

2011 358767

2013 442110

2015 618727

2012 415605

2014 519426

2016 647803

NA 13

Time taken: 82.192 seconds, Fetched: 8 row(s)

hive>

**Step 3: formatting the table and removing the null values from the existing table by creating new table.**

CREATE TABLE h1b\_app2(sno int,case\_status string, employer\_name

string, soc\_name string, job\_title string, full\_time\_position

string,prevailining\_wage int,year string, worksite string, longitude

double, latitude double )

row format delimited

fields terminated by '\t'

STORED AS TEXTFILE;

INSERT OVERWRITE TABLE h1b\_app2 SELECT regexp\_replace(sno, "\t", ""),

regexp\_replace(case\_status, "\t", ""), regexp\_replace(employer\_name,

"\t", ""), regexp\_replace(soc\_name, "\t", ""),

regexp\_replace(job\_title, "\t", ""),

regexp\_replace(full\_time\_position, "\t", ""),

regexp\_replace(prevailining\_wage, "\t", ""), regexp\_replace(year, "\t",

""), regexp\_replace(worksite, "\t", ""), regexp\_replace(longitude,

"\t", ""), regexp\_replace(latitude, "\t", "") FROM h1b\_applications

where case\_status != "NA";

**Step 4: creating final table to order the case\_status field in the existing tables.**

CREATE TABLE h1b\_final(sno int,case\_status string, employer\_name

string, soc\_name string, job\_title string, full\_time\_position

string,prevailining\_wage int,year string, worksite string, longitude

double, latitude double )

row format delimited

fields terminated by '\t'

STORED AS TEXTFILE;

INSERT OVERWRITE TABLE h1b\_final SELECT sno, case when trim(case\_status) = "PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED" then "DENIED" when trim(case\_status) = "REJECTED" then "DENIED" when trim(case\_status) = "INVALIDATED" then "DENIED"

else case\_status end,employer\_name, soc\_name, job\_title, full\_time\_position,prevailining\_wage,year, worksite, longitude, latitude FROM h1b\_app2;

hive> select distinct case\_status from h1b\_final;

CERTIFIED-WITHDRAWN

WITHDRAWN

CERTIFIED

DENIED

Time taken: 57.755 seconds, Fetched: 4 row(s)

select COUNT(\*) from h1b\_final;

3002445

select \* from h1b\_final limit 5;

OK

1 CERTIFIED-WITHDRAWN UNIVERSITY OF MICHIGAN BIOCHEMISTS AND BIOPHYSICISTS POSTDOCTORAL RESEARCH FELLOW N 36067 2016 ANN ARBOR, MICHIGAN -83.7430378 42.2808256

2 CERTIFIED-WITHDRAWN GOODMAN NETWORKS, INC. CHIEF EXECUTIVES CHIEF OPERATING OFFICER Y 242674 2016 PLANO, TEXAS -96.6988856 33.0198431

3 CERTIFIED-WITHDRAWN PORTS AMERICA GROUP, INC. CHIEF EXECUTIVESCHIEF PROCESS OFFICER Y 193066 2016 JERSEY CITY, NEW JERSEY -74.0776417 40.7281575

4 CERTIFIED-WITHDRAWN GATES CORPORATION, A WHOLLY-OWNED SUBSIDIARY OF TOMKINS PLC CHIEF EXECUTIVES REGIONAL PRESIDEN, AMERICAS Y 220314 2016 DENVER, COLORADO -104.990251 39.7392358

5 WITHDRAWN PEABODY INVESTMENTS CORP. CHIEF EXECUTIVES PRESIDENT MONGOLIA AND INDIA Y 157518 2016 ST. LOUIS, MISSOURI -90.1994042 38.6270025

Time taken: 0.126 seconds, Fetched: 5 row(s)

hive>

select distinct year from h1b\_final;

2011

2013

2015

2012

2014

2016

Data is stored in HDFS under /user/hive/warehouse/niit1.db/h1b\_final

**Hadoop Distributed File System:** HDFS, the storage layer of Hadoop, is a distributed, scalable, Java-based file system adept at storing large volumes of unstructured data.

using hadoop command coping data from hive to rool directory:

hadoop fs -cp -p /user/hive/warehouse/niit.db/h1b\_final /

**Analysis which I have done by using the tools mapreduce, hive, pig and sqoop**

**MapReduce**

**MapReduce:** MapReduce is a software framework that serves as the compute layer of Hadoop. MapReduce jobs are divided into two parts. The “Map” function divides a query into multiple parts and processes data at the node level. The “Reduce” function aggregates the results of the “Map” function to determine the “answer” to the query.

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

Ans:

package h1b\_project;

import java.io.\*;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

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

public class DataEngineer {

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

{

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

{

try{

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

String job\_title=record[4];

String year=record[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 ReduceClass extends Reducer<NullWritable,Text,NullWritable,Text>

{

Text result =new Text();

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

{

long count1=0,count2=0,count3=0,count4=0,count5=0,count6=0;

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

for (Text val : values)

{

String year=val.toString();

if(year.equals("2011"))

{

count1++;

}

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

{

count2++;

}

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

{

count3++;

}

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

{

count4++;

}

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

{

count5++;

}

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

{

count6++;

}

}

if(count1 !=0)

{

cycle1=((count2-count1)\*100)/count1;

}

else {

cycle1=0;

}

if (count2 !=0)

{

cycle2=((count3-count2)\*100)/count2;

}

else

{

cycle2=0;

}

if (count3 !=0)

{

cycle3=((count4-count3)\*100)/count3;

}

else

{

cycle3=0;

}

if (count4 !=0)

{

cycle4=((count5-count4)\*100)/count4;

}

else

{

cycle4=(count5-count4)\*10;

}

if (count5 !=0)

{

cycle5=((count6-count5)\*100)/count5;

}

else

{

cycle5=0;

}

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

String newavg=String.format("%.2f", avg);

String myrow=cycle1+","+cycle2+","+cycle3+","+","+cycle4+","+cycle5+","+newavg;

result.set(myrow);

context.write(key, result);

}

}

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

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

conf.set("mapreduce.output.textoutputformat.separator", ",");

job.setJarByClass(DataEngineer.class);

job.setMapperClass(MapClass.class);

job.setReducerClass(ReduceClass.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);

}

}

17/04/30 04:37:58 INFO mapreduce.Job: Counters: 50

File System Counters

FILE: Number of bytes read=12053

FILE: Number of bytes written=630904

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=449865655

HDFS: Number of bytes written=31

HDFS: Number of read operations=15

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Killed map tasks=1

Launched map tasks=5

Launched reduce tasks=1

Data-local map tasks=5

Total time spent by all maps in occupied slots (ms)=204177

Total time spent by all reduces in occupied slots (ms)=12730

Total time spent by all map tasks (ms)=204177

Total time spent by all reduce tasks (ms)=12730

Total vcore-seconds taken by all map tasks=204177

Total vcore-seconds taken by all reduce tasks=12730

Total megabyte-seconds taken by all map tasks=209077248

Total megabyte-seconds taken by all reduce tasks=13035520

Map-Reduce Framework

Map input records=3002445

Map output records=1721

Map output bytes=8605

Map output materialized bytes=12071

Input split bytes=460

Combine input records=0

Combine output records=0

Reduce input groups=1

Reduce shuffle bytes=12071

Reduce input records=1721

Reduce output records=1

Spilled Records=3442

Shuffled Maps =4

Failed Shuffles=0

Merged Map outputs=4

GC time elapsed (ms)=2104

CPU time spent (ms)=13800

Physical memory (bytes) snapshot=1026654208

Virtual memory (bytes) snapshot=7510507520

Total committed heap usage (bytes)=771047424

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=449865195

File Output Format Counters

Bytes Written=31

35.0,86.0,64.0,,58.0,99.0,68.40

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

Ans:

package h1b\_project;

import java.io.\*;

import java.util.TreeMap;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

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

public class JobTitles {

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

{

Text job =new Text();

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

{

try{

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

String job\_title=record[4];

String year=record[7];

job.set(job\_title);

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

}

catch(Exception e)

{

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

}

}

}

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

{

private TreeMap<Double, Text> topRecordMap = new TreeMap<Double, Text>();

Text result =new Text();

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

{

long count1=0,count2=0,count3=0,count4=0,count5=0,count6=0;

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

for (Text val : values)

{

String year=val.toString();

if(year.equals("2011"))

{

count1++;

}

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

{

count2++;

}

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

{

count3++;

}

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

{

count4++;

}

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

{

count5++;

}

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

{

count6++;

}

}

if(count1 !=0)

{

cycle1=((count2-count1)\*100)/count1;

}

else {

cycle1=0;

}

if (count2 !=0)

{

cycle2=((count3-count2)\*100)/count2;

}

else

{

cycle2=0;

}

if (count3 !=0)

{

cycle3=((count4-count3)\*100)/count3;

}

else

{

cycle3=0;

}

if (count4 !=0)

{

cycle4=((count5-count4)\*100)/count4;

}

else

{

cycle4=0;

}

if (count5 !=0)

{

cycle5=((count6-count5)\*100)/count5;

}

else

{

cycle5=0;

}

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

String newavg=String.format("%.2f", avg);

String mykey=key.toString();

String myvalue=mykey+","+newavg;

topJobMap.put(new Double(newavg),new Text(myvalue));

if (topJobMap.size() > 5)

{

topJobMap.remove(topJobMap.firstKey());

}

}

protected void cleanup(Context context) throws IOException,

InterruptedException

{

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

{

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

}

}

}

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

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

conf.set("mapreduce.output.textoutputformat.separator", ",");

job.setJarByClass(JobTitles.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);

}

}

File System Counters

FILE: Number of bytes read=89991890

FILE: Number of bytes written=180590493

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=449865655

HDFS: Number of bytes written=157

HDFS: Number of read operations=15

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Launched map tasks=4

Launched reduce tasks=1

Data-local map tasks=4

Total time spent by all maps in occupied slots (ms)=294764

Total time spent by all reduces in occupied slots (ms)=28318

Total time spent by all map tasks (ms)=294764

Total time spent by all reduce tasks (ms)=28318

Total vcore-seconds taken by all map tasks=294764

Total vcore-seconds taken by all reduce tasks=28318

Total megabyte-seconds taken by all map tasks=301838336

Total megabyte-seconds taken by all reduce tasks=28997632

Map-Reduce Framework

Map input records=3002445

Map output records=3002445

Map output bytes=83986994

Map output materialized bytes=89991908

Input split bytes=460

Combine input records=0

Combine output records=0

Reduce input groups=287542

Reduce shuffle bytes=89991908

Reduce input records=3002445

Reduce output records=5

Spilled Records=6004890

Shuffled Maps =4

Failed Shuffles=0

Merged Map outputs=4

GC time elapsed (ms)=2934

CPU time spent (ms)=31930

Physical memory (bytes) snapshot=1044996096

Virtual memory (bytes) snapshot=7508353024

Total committed heap usage (bytes)=777949184

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=449865195

File Output Format Counters

Bytes Written=157

[cloudera@quickstart ~]$

BUSINESS ANALYST 2,4930.00

SENIOR SYSTEMS ANALYST JC60,4255.40

PROGRAMMER/ DEVELOPER,4160.00

BUSINESS SYSTEMS ANALYST 2,3966.80

SOFTWARE DEVELOPER 2,3480.80

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

package h1b2;

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

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

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class DataEngineerUS extends Configured implements Tool

{

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

{

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

{

try

{

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

String job\_title=ecord[4];

String year=record[7];

String worksite=record[8];

String data=year;

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

{

context.write(new Text(worksite),new Text(data));

}

}

catch(Exception e)

{

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

}

}

}

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

{

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

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

{

String mykey=key.toString();

String year=null;

long total\_petition = 0;

for(Text val:values)

{

total\_petition++;

year = val.toString();

}

String petitions=String.format("%d",total\_petition);

String myvalue=mykey+"\t"+year+"\t"+petitions;

topMap.put(new Long(total\_petition), new Text(myvalue));

}

public void cleanup(Context context) throws IOException, InterruptedException

{

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

{

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

}

}

}

public static class MyPartition extends Partitioner<Text,Text>

{

@Override

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

{

String year=value.toString();

if(year.equals("2011"))

{

return 0 %numReduceTasks;

}

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

{

return 1 %numReduceTasks;

}

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

{

return 2 %numReduceTasks;

}

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

{

return 3 %numReduceTasks;

}

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

{

return 4 %numReduceTasks;

}

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

{

return 5 %numReduceTasks;

}

else

{

return 6;

}

}

}

@Override

public int run(String[] arg) throws Exception

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

job.setJarByClass(DataEnginerUS.class);

job.setJobName("Find which part of US has most data engineer jobs ");

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setMapperClass(MapperClass.class);

job.setReducerClass(ReducerClass.class);

job.setPartitionerClass(MyPartition.class);

job.setNumReduceTasks(7);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(arg[0]));

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

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

return 0;

}

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

{

ToolRunner.run(new Configuration(),new DataEngineerUS (), args);

System.exit(0);

}

}

SEATTLE, WASHINGTON 2011 20

SAN FRANCISCO, CALIFORNIA 2011 4

SAN MATEO, CALIFORNIA 2011 3

WALTHAM, MASSACHUSETTS 2011 2

TALLAHASSEE, FLORIDA 2011 1

SEATTLE, WASHINGTON 2012 30

SAN FRANCISCO, CALIFORNIA 2012 10

PONTIAC, MICHIGAN 2012 3

SAN MATEO, CALIFORNIA 2012 2

WOODLAND HILLS, CALIFORNIA 2012 1

SEATTLE, WASHINGTON 2013 46

SAN FRANCISCO, CALIFORNIA 2013 17

MENLO PARK, CALIFORNIA 2013 12

NEW YORK, NEW YORK 2013 6

ATLANTA, GEORGIA 2013 5

MOUNTAIN VIEW, CALIFORNIA 2013 3

THOUSAND OAKS, CALIFORNIA 2013 2

WOODLAND HILLS, CALIFORNIA 2013 1

SEATTLE, WASHINGTON 2014 45

SAN FRANCISCO, CALIFORNIA 2014 34

MENLO PARK, CALIFORNIA 2014 21

NEW YORK, NEW YORK 2014 18

MOUNTAIN VIEW, CALIFORNIA 2014 13

SAN MATEO, CALIFORNIA 2014 8

IRVINE, CALIFORNIA 2014 7

REDWOOD CITY, CALIFORNIA 2014 5

SUNNYVALE, CALIFORNIA 2014 4

ST. PETERSBURG, FLORIDA 2014 3

WINSTON-SALEM, NORTH CAROLINA 2014 2

YONKERS, NEW YORK 2014 1

SEATTLE, WASHINGTON 2015 61

NEW YORK, NEW YORK 2015 41

MENLO PARK, CALIFORNIA 2015 23

MOUNTAIN VIEW, CALIFORNIA 2015 18

SAN MATEO, CALIFORNIA 2015 15

SANTA MONICA, CALIFORNIA 2015 13

SAN RAMON, CALIFORNIA 2015 8

SUNNYVALE, CALIFORNIA 2015 7

SAN JOSE, CALIFORNIA 2015 6

REDWOOD CITY, CALIFORNIA 2015 5

CHICAGO, ILLINOIS 2015 4

TROY, MICHIGAN 2015 3

WESTBOROUGH, MASSACHUSETTS 2015 2

WOODLAND HILLS, CALIFORNIA 2015 1

SEATTLE, WASHINGTON 2016 128

SAN FRANCISCO, CALIFORNIA 2016 90

NEW YORK, NEW YORK 2016 70

MENLO PARK, CALIFORNIA 2016 39

IRVINE, CALIFORNIA 2016 18

SUNNYVALE, CALIFORNIA 2016 16

SAN MATEO, CALIFORNIA 2016 14

CHICAGO, ILLINOIS 2016 13

SANTA CLARA, CALIFORNIA 2016 12

MOUNTAIN VIEW, CALIFORNIA 2016 11

SAN JOSE, CALIFORNIA 2016 10

PLANO, TEXAS 2016 9

SANTA MONICA, CALIFORNIA 2016 8

WALTHAM, MASSACHUSETTS 2016 7

BURLINGTON, MASSACHUSETTS 2016 6

SAN BRUNO, CALIFORNIA 2016 5

VIENNA, VIRGINIA 2016 4

VENICE, CALIFORNIA 2016 3

WILMINGTON, DELAWARE 2016 2

YORKTOWN HEIGHTS, NEW YORK 2016 1

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

Ans:

package h1b2;

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

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

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class TopLocation extends Configured implements Tool

{

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

{

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

{

try

{

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

String year=record[7];

String job\_title=record[4];

String worksite=record[8];

String value=year

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

{

context.write(new Text(worksite),new Text(value));

}

}

catch(Exception e)

{

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

}

}

}

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

{

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

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

{

String mykey=key.toString();

long total\_petition = 0;

String year=null;

for(Text val:values)

{

total\_petition++;

year = val.toString();

}

String petition=String.format("%d",petition);

String myvalue=mykey+"\t"+year+"\t"+petition;

topMap.put(new Long(total\_petition),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 class MyPartition extends Partitioner<Text,Text>

{

@Override

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

{

String year=value.toString();

if(year.equals("2011"))

{

return 0 %numReduceTasks;

}

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

{

return 1 %numReduceTasks;

}

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

{

return 2 %numReduceTasks;

}

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

{

return 3 %numReduceTasks;

}

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

{

return 4 %numReduceTasks;

}

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

{

return 5 %numReduceTasks;

}

else

{

return 6;

}

}

}

@Override

public int run(String[] arg) throws Exception

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

job.setJarByClass(TopLocation.class);

job.setJobName("Find Top 5 location in US who got Certified visa in each year ");

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setMapperClass(MapperClass.class);

job.setReducerClass(ReducerClass.class);

job.setPartitionerClass(MyPartition.class);

job.setNumReduceTasks(7);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(arg[0]));

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

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

return 0;

}

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

{

ToolRunner.run(new Configuration(),new TopLocation(), args);

System.exit(0);

}

}

NEW YORK, NEW YORK 2011 23172

HOUSTON, TEXAS 2011 8184

CHICAGO, ILLINOIS 2011 5188

SAN JOSE, CALIFORNIA 2011 4713

SAN FRANCISCO, CALIFORNIA 2011 4711

NEW YORK, NEW YORK 2012 23737

HOUSTON, TEXAS 2012 9963

SAN FRANCISCO, CALIFORNIA 2012 6116

CHICAGO, ILLINOIS 2012 5671

ATLANTA, GEORGIA 2012 5565

NEW YORK, NEW YORK 2013 23537

HOUSTON, TEXAS 2013 11136

SAN FRANCISCO, CALIFORNIA 2013 7281

SAN JOSE, CALIFORNIA 2013 6722

ATLANTA, GEORGIA 2013 6377

NEW YORK, NEW YORK 2014 27634

HOUSTON, TEXAS 2014 13360

SAN FRANCISCO, CALIFORNIA 2014 9798

SAN JOSE, CALIFORNIA 2014 8223

ATLANTA, GEORGIA 2014 8213

NEW YORK, NEW YORK 2015 31266

HOUSTON, TEXAS 2015 15242

SAN FRANCISCO, CALIFORNIA 2015 12594

ATLANTA, GEORGIA 2015 10500

SAN JOSE, CALIFORNIA 2015 9589

NEW YORK, NEW YORK 2016 34639

SAN FRANCISCO, CALIFORNIA 2016 13836

HOUSTON, TEXAS 2016 13655

ATLANTA, GEORGIA 2016 11678

CHICAGO, ILLINOIS 2016 11064

**Hive**

**Hive:** Hive is a Hadoop-based data warehousing-like framework originally developed by Facebook. It allows users to write queries in a SQL-like language caled HiveQL, which are then converted to MapReduce. This allows SQL programmers with no MapReduce experience to use the warehouse and makes it easier to integrate with business intelligence and visualization tools such as Microstrategy, Tableau, Revolutions Analytics, etc.

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

Ans:

select soc\_name, count(job\_title) as count from h1b\_final where job\_title like '%DATA SCIENTIST%' group by soc\_name order by count desc limit 10;

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

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

Ans:

create view topemp as select employer\_name,year, count(case\_status) as cnt from h1b\_final where year in ('2011','2012','2013','2014','2015','2016') group by year, employer\_name sort by year, cnt desc;

select year, employer\_name, cnt ,rank from(select year, employer\_name, rank() over (partition by year order by cnt desc) as rank,cnt from topemp) ranked\_table where ranked\_table.rank <=5;

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

Time taken: 111.88 seconds, Fetched: 30 row(s)

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

Ans:

create view topjob as select job\_title,year, count(case\_status) as cnt from h1b\_final where year in ('2011','2012','2013','2014','2015','2016') group by year, job\_title sort by year, cnt desc;

select year, job\_title, cnt ,rank from(select year, job\_title, rank() over (partition by year order by cnt desc) as rank,cnt from topjob) ranked\_table where ranked\_table.rank <=10;

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

Time taken: 150.405 seconds, Fetched: 60 row(s)

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

Ans:

select year, COUNT(case\_status) from h1b\_final group by year;

2011 358767

2013 442114

2015 618727

2012 415607

2014 519427

2016 647803

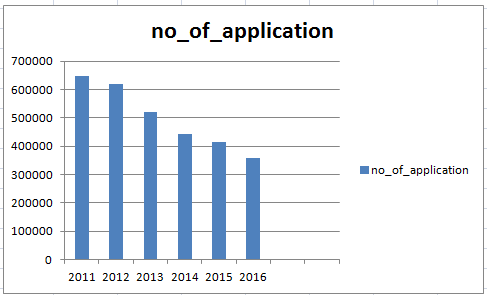


Figure 7: Number of application per year

**Pig Latin:**

**Pig:** Pig Latin is a Hadoop-based language developed by Yahoo. It is relatively easy to learn and is adept at very deep, very long data pipelines (a limitation of SQL.)

**6. Find the percentage and the count of each case\_status on total applications for each year, Creating a graph depicting the pattern of all cases.**

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

new = foreach h1b generate $7, $1;

groupbyyear = group new by year;

newgroup = foreach groupbyyear generate group as year, COUNT(b)as total;

filterbycase= filter new by case\_status=='CERTIFIED';

filteryear= group filterbycase by year;

certified = foreach filteryear generate group as year, COUNT(x) as certified;

joindata = join newgroup by $0,certified by $0;

data = foreach joindata generate $0,$1,$3,((double)$3\*100/(double)$1) as percent;

dump data;

**year total certified percent**

(2011,358767,307936,85.83175152675692)

(2012,415607,352668,84.85612609989725)

(2013,442114,382951,86.61815730784367)

(2014,519427,455144,87.62424748809747)

(2015,618727,547278,88.45225761927313)

(2016,647803,569646,87.93506667922192)

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

new = foreach h1b generate $7, $1;

groupbyyear = group new by year;

newgroup = foreach groupbyyear generate group as year, COUNT(b)as total;

filterbycase= filter new by case\_status=='CERTIFIED-WITHDRAWN';

filteryear= group filterbycase by year;

certified = foreach filteryear generate group as year, COUNT(x) as certified;

joindata = join newgroup by $0,certified by $0;

data = foreach joindata generate $0,$1,$3,((double)$3\*100/(double)$1) as percent;

dump data;

(2011,358767,11596,3.2321813321738064)

(2012,415607,31118,7.487361858678993)

(2013,442114,35432,8.014222576077664)

(2014,519427,36350,6.9980959788382195)

(2015,618727,41071,6.637984119005636)

(2016,647803,47092,7.269493966529948)

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

new = foreach h1b generate $7, $1;

groupbyyear = group new by year;

newgroup = foreach groupbyyear generate group as year, COUNT(b)as total;

filterbycase= filter new by case\_status=='WITHDRAWN';

filteryear= group filterbycase by year;

certified = foreach filteryear generate group as year, COUNT(x) as certified;

joindata = join newgroup by $0,certified by $0;

data = foreach joindata generate $0,$1,$3,((double)$3\*100/(double)$1) as percent;

dump data;

(2011,358767,10105,2.816591269542628)

(2012,415607,10725,2.5805628875355806)

(2013,442114,11590,2.621495813297023)

(2014,519427,16034,3.086863024063054)

(2015,618727,19455,3.144359305477214)

(2016,647803,21890,3.3791137120389996)

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

new = foreach h1b generate $7, $1;

groupbyyear = group new by year;

newgroup = foreach groupbyyear generate group as year, COUNT(b)as total;

filterbycase= filter new by case\_status=='DENIED';

filteryear= group filterbycase by year;

certified = foreach filteryear generate group as year, COUNT(x) as certified;

joindata = join newgroup by $0,certified by $0;

data = foreach joindata generate $0,$1,$3,((double)$3\*100/(double)$1) as percent;

dump data;

(2011,358767,29130,8.119475871526646)

(2012,415607,21096,5.0759491538881685)

(2013,442114,12141,2.7461243027816353)

(2014,519427,11899,2.290793509001265)

(2015,618727,10923,1.76539895624403)

(2016,647803,9175,1.4163256422091284)

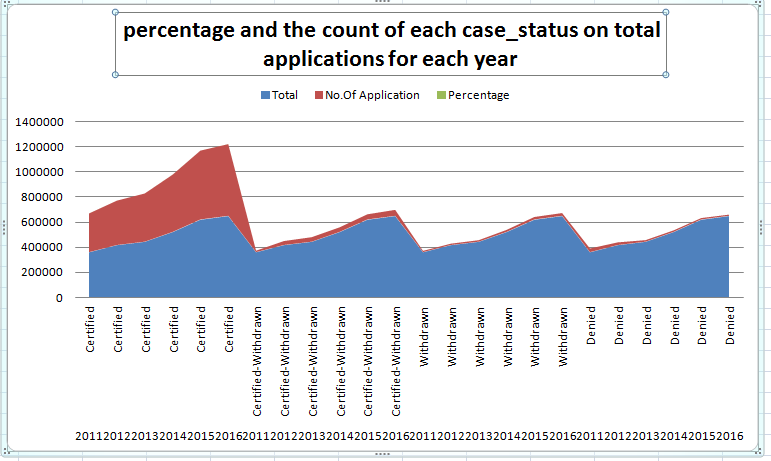


Figure 8: Percentage And Count

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

Ans:

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

b = filter a by year=='2011';

c = foreach b generate $4, $5, $6, $7;

d = group c by ($0, $1);

e = foreach d generate group as job\_title, COUNT(c),SUM(c.prevailining\_wage);

f = foreach e generate $0, ($2/$1)as avg;

g = order f by $0 desc;

h = limit g 10;

dump h;

2011

(( SYSTEMS ANALYST,Y),42078.5)

(( COMPUTER SYSTEMS ENGINEER,Y),46218.0)

((|NFORMATION MANAGEMENT SPECIALIST,Y),38875.0)

(([PHYSICAL THERAPIST,Y),69035.0)

(([HIOX] COMMERCIAL SPECIALIST (SALES ENGINEER),Y),63357.0)

((ZOOLOGIST,Y),40914.0)

((ZONE MANAGER, OPERATIONS & ANALYSIS,Y),89378.0)

((ZONE BUSINESS DEVELOPMENT MANAGER,Y),80912.0)

((YOUTUBE STRATEGY & OPERATIONS ANALYST,Y),72238.0)

((YOUTH SERVICE DEPARTMENT SUPERVISOR,Y),30514.0)

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

b = filter a by year=='2012';

c = foreach b generate $4, $5, $6, $7;

d = group c by ($0, $1);

e = foreach d generate group as job\_title, COUNT(c),SUM(c.prevailining\_wage);

f = foreach e generate $0, ($2/$1)as avg;

g = order f by $0 desc;

h = limit g 10;

dump h;

2012

(( LEAD TEST ANALYST,Y),69389.0)

((ZOOLOGISTS AND WILDLIFE BIOLOGISTS I,Y),31803.0)

((ZOOLOGIST - REPRODUCTIVE PHYSIOLOGY,Y),56222.0)

((ZOOKEEPER,Y),20800.0)

((ZOO BIRD KEEPER,Y),30784.0)

((ZONING MANAGER,Y),84635.0)

((ZONE MERCHANDISER,Y),64064.0)

((ZLC SPECIALIST, PROCESS IMPROVEMENT,Y),67080.0)

((YOUTH WORKER,Y),24044.0)

((YOUTH THERAPIST,Y),28558.0)

2013

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

b = filter a by year=='2013';

c = foreach b generate $4, $5, $6, $7;

d = group c by ($0, $1);

e = foreach d generate group as job\_title, COUNT(c),SUM(c.prevailining\_wage);

f = foreach e generate $0, ($2/$1)as avg;

g = order f by $0 desc;

h = limit g 10;

dump h;

(( TEST ANALYST - US,Y),53872.0)

(( TECHNOLOGY ARCHITECT - US,Y),96033.0)

(( LEAD CONSULTANT - US,Y),99652.0)

(( CONSULTANT - US,Y),56992.0)

(([FINANCIAL] ANALYST, STRUCTURED CREDIT,Y),73070.0)

((ZYQAD SPECIALIST,Y),88493.5)

((ZOO EDUCATION COORDINATOR,Y),36899.0)

((ZONE ACCOUNT MANAGER, CAPITOL DEAL,Y),30017.0)

((ZMS SOFTWARE ENGINEER,Y),105206.0)

((ZIMBABWE PARTNERSHIP COORDINATOR,N),25729.0)

2014

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

b = filter a by year=='2014';

c = foreach b generate $4, $5, $6, $7;

d = group c by ($0, $1);

e = foreach d generate group as job\_title, COUNT(c),SUM(c.prevailining\_wage);

f = foreach e generate $0, ($2/$1)as avg;

g = order f by $0 desc;

h = limit g 10;

dump h;

(( TEAM LEAD - US,Y),46779.0)

(( SOFTWARE TEST ENGINEER,Y),65936.0)

(( SENIOR PROJECT LEADER,Y),60778.0)

(( QUALITY ASSURANCE ANALYST,Y),77938.0)

(( MOBILE SQA ENGINEER   ,Y),41288.0)

(( BUSINESS INTELLIGENCE ANALYST,Y),77938.0)

((`QUALITY ASSURANCE ANALYST,Y),55682.0)

((]ENGINEERING LEAD,Y),51542.0)

((ZOOLOGIST,N),49795.0)

((ZOOKEEPER,Y),27745.0)

2015

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

b = filter a by year=='2015';

c = foreach b generate $4, $5, $6, $7;

d = group c by ($0, $1);

e = foreach d generate group as job\_title, COUNT(c),SUM(c.prevailining\_wage);

f = foreach e generate $0, ($2/$1)as avg;

g = order f by $0 desc;

h = limit g 10;

dump h;

(( SYSTEMS ANALYST,Y),61776.0)

(( SOFTWARE TEST ENGINEER,Y),78707.0)

(( SAS ANALYST,Y),55598.0)

(( ORACLE APPS DBA,Y),60674.0)

((  MIDDLEWARE ADMINISTRATION.,Y),57429.0)

((ZOS SYSTEMS PROGRAMMER,Y),87818.0)

((ZONE ENGINEER,Y),82118.0)

((ZONAL ISOLATION SEGMENT ENGINEERING TECHNICAL AUTHORITY,Y),112486.0)

((ZMS WEB CLIENT ENGINEER,Y),108534.0)

((ZMS ENGINEER,Y),89502.0)

2016

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

b = filter a by year=='2016';

c = foreach b generate $4, $5, $6, $7;

d = group c by ($0, $1);

e = foreach d generate group as job\_title, COUNT(c),SUM(c.prevailining\_wage);

f = foreach e generate $0, ($2/$1)as avg;

g = order f by $0 desc;

h = limit g 10;

dump h;

(( SR. BUSINESS INTELLIGENCE DEVELOPER,N),69909.0)

(( SOFTWARE PROGRAMMER,N),65042.0)

(( SOFTWARE ENGINEER,N),65042.0)

(( SHAREPOINT/SQL DEVELOPER,Y),76107.0)

(( QA ANALYST,N),56555.0)

(( PROJECT MANAGERS,N),60986.0)

(( BUSINESS INTELLIGENCE ANALYST,Y),77230.0)

(( BUSINESS ANALYST,N),60133.0)

((  MIDDLEWARE ADMINISTRATION.,N),57429.0)

(([FINANCIAL] ANALYST, STRUCTURED CREDIT,Y),80163.0)

**9) Which are top ten employers who have the highest success rate morethan 70% in petitions filed more than 1000?**

Ans:

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

new = foreach h1b generate employer\_name,case\_status;

filterbywithdrawn = filter new by case\_status=='CERTIFIED-WITHDRAWN';

filterbycertified = filter new by case\_status=='CERTIFIED';

newgroup = group new by employer\_name;

groupcertified = group filterbycertified by employer\_name;

groupwithdrawn = group filterbywithdraw by employer\_name;

petitions = foreach newgroup generate group as employer,COUNT(new) as cnt;

certifiedpetition= foreach groupcertified generate group as employer,COUNT(filterbycertified) as cnt;

withdrawnpetition = foreach groupwithdrawn generate group as employer,COUNT(filterbywithdrawn) as cnt;

joindata = join petitions by $0, certifiedpetition by $0, withdrawnpetition by $0;

value1 = foreach joindata generate $0,$1,($3+$5);

success = foreach value1 generate $0,$1,((double)$2\*100/(double)$1) as success;

filterbycondition = filter success by $1>=10000 and $2>70.0;

orderdata = order filterbycondition by $2 desc;

final =limit orderdata 10;

dump final;

(INFOSYS LIMITED,130592,99.5405537858368)

(ACCENTURE LLP,33447,99.393069632553)

(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)

(GOOGLE INC.,16473,96.59442724458205)

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

Ans:

h1b = load '/user/hive/warehouse/h1b\_final' using PigStorage() as

(sno:int, case\_status:chararray, employer\_name:chararray, soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailining\_wage:double, year:chararray, worksite:chararray, longitude:int, latitude:int);

new = foreach h1b generate $4,$1;

filterbycertified = filter new by case\_status=='CERTIFIED';

filterbywithdrawn = filter new by case\_status=='CERTIFIED-WITHDRAWN';

newgroup = group new by job\_title;

groupcertified = group filterbycertified by job\_title;

groupwithdrawn = group filterbywithdrawn by job\_title;

petitions = foreach newgroup generate group as job\_Position,COUNT(new) as cnt;

certifiedpetition = foreach groupcertified generate group as job\_Position,COUNT(v filterbycertified ) as cnt;

withdrawnpetition = foreach groupwithdrawn generate group as job\_Position,COUNT(filterbywithdrawn) as cnt;

joindata = join petitions by $0, certifiedpetition by $0, withdrawnpetition by $0;

value1 = foreach joindata generate $0,$1,($3+$5);

successrate = foreach value1 generate $0,$1,((double)$2\*100/(double)$1) as success;

result = filter successrate by $1>=10000 and $2>70.0

orderdata = order result by $2 desc;

result1 =limit orderdata 10;

dump result1;

(AGRIGENETICS D/B/A MYCOGEN CORPORATION, A SUBSIDIARY OF THE DOW CHEMIC,7,100.0)

(AUSCO PETROLEUM, INC. (D/B/A AUSTIN EXPLORATION, LTD. OR AUS-TEX EXPLO,12,100.0)

(DENTSPLY TULSA DENTAL SPECIALTIES (DIVISION OF DENTSPLY INTERNATIONAL),2,100.0)

(EUROFINS LANCASTER LABORATORIES, PROFESSIONAL SCIENTIFIC SERVICES, LLC,7,100.0)

(NEW YORK CITY CHARTER HS FOR ARCHIT., ENG'G, & CONSTRUCTION INDUSTRIES,2,100.0)

(ROHM AND HAAS CHEMICALS, LLC, A SUBSIDIARY OF THE DOW CHEMICAL COMPANY,15,100.0)

(ROHM AND HAAS ELECTRONIC MATERIALS LLC, A SUBSIDIARY OF THE DOW CHEMIC,17,100.0)

(TEJANO CENTER FOR COMMUNITY CONCERNS - RAUL YZAGUIRRE SCHOOL FOR SUCCE,3,100.0)

(THE GRADUATE SCHOOL AND UNIVERSITY CENTER OF THE CITY UNIVERSITY OF NY,3,100.0)

(VEOLIA NORTH AMERICA, INC. F/K/A VEOLIA ENVIRONNEMENT NORTH AMERICA OP,5,100.0)

**Sqoop**

**Sqoop:** Sqoop is a connectivity tool for moving data from non-Hadoop data stores – such as relational databases and data warehouses – into Hadoop. It allows users to specify the target location inside of Hadoop and instruct Sqoop to move data from Oracle, Teradata or other relational databases to the target.

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

Ans: