H1B VISA APPLICANTS CASE STUDY USING HADOOP FRAMEWORK

H1b is an employment based, immigrant visa category for temporary foreign national in US

KAMAL KANT

BATCH ID – S170026

1.**ABSTRACT**

The project is focussed on analysis of H1B VISA APPLICATIONS. Using Hadoop, an open source framework for distributed computing, the raw data of the h1b visa is processed and analysed.

For applying h1b visa, employer in US has to offer a job along with h1b visa petition with the US immigration department.

The dataset has nearly 3 million records.

**The dataset description is as follows:**

The columns in the dataset include:

1. **ID :** Unique ID of each record.
2. **CASE\_STATUS:** Status associated with the last significant event or decision. Valid values include “Certified,” “Certified-Withdrawn,” Denied,” and “Withdrawn”.

**Certified:** Employer filed the LCA, which was approved by DOL

**Certified Withdrawn:** LCA was approved but later withdrawn by employer

**Withdrawn:** LCA was withdrawn by employer before approval

**Denied:** LCA was denied by DOL

1. **EMPLOYER\_NAME**: Name of employer submitting labour condition application.
2. **SOC\_NAME:** the Occupational name associated with the SOC\_CODE.

SOC\_CODE is the occupational code associated with the job being requested for temporary labour condition, as classified by the Standard Occupational Classification (SOC) System.

1. **JOB\_TITLE**: Title of the job
2. **FULL\_TIME\_POSITION**: Y = Full Time Position; N = Part Time Position
3. **PREVAILING\_WAGE:** Prevailing Wage for the job being requested for temporary labour condition. The wage is listed at annual scale in USD. The prevailing wage for a job position is defined as the average wage paid to similarly employed workers in the requested occupation in the area of intended employment. The prevailing wage is based on the employer’s minimum requirements for the position.
4. **YEAR**: Year in which the H1B visa petition was filed.
5. **WORKSITE**: City and State information of the foreign worker’s intended area of employment
6. **lon**: longitude of the Worksite
7. **lat**: latitude of the Worksite

2.**BIG DATA**

Due to the advent of new technologies, devices, and communication means like social networking sites, applications the amount of data produced by mankind is growing rapidly every year.

Extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.

Data that is too big to be processed by a single machine.

It’s not easy to measure the total volume of data stored electronically.

This flood of data coming from many Sources:

* Facebook hosts more than 240 billion photos , growing at 7 PB per months
* The New York Stock Exchange generates 4-5 PB per day
* And there are lot of data - 4.4 ZB of data collected in 2003 and forecasting 44 ZB in 2020

**2.1Characteristics of BigData**

Big Data includes 4Vs:

* **Volume -** Rather than thousands or million of rows/records , Big data can be billions of rows and millions of columns
* **Variety -** Data can be any form ,structure
* **Velocity -** Big data can be describe as high velocity data i.e speed of new data creation is rapid ,high data ingestion and near real time analysis
* **Veracity -** The quality of the data being captured can vary greatly. Accuracy of analysis depends on the veracity of the source data

**2.2 There are three types of data available:**

* **Structured data**: Relational data.
* **Semi Structured data**: XML data.
* **Unstructured data**: Word, PDF, Text, Media Logs.

**2.3 Main Problem:**

THE PROBLEM HERE IS HOW TO ANALYSE AND STORE HUGE VOLUME OF DATA

* How to store & analyse such huge data?
* Problem :Lots of data of varied form

Need to process 100TB datasets On 1 node: – reading@ 100MB/s = 277.7 hrs. = 11 days

Suppose we have 1000 nodes then the same

Data can be read =16.2 minutes only if nodes working in parallel

3.**HADOOP**

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models.

Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

**3.1 Benefits of Hadoop**

* + Scalability
  + Scale out : Adding more nodes to a cluster
  + Scale up: Adding more CPU/ram/external hardware to the existing nodes
  + Reliability – In case of node failure processing is re-directed to the remaining nodes in the cluster and data is automatically re-replicated in preparation for future node failures.
  + Flexibility – Don’t have to create schemas on write, Schemas created on read
  + Low cost - Hadoop is open source and runs on low-cost commodity hardware



**Fig. Hadoop Ecosystem**

4.**HDFS** (HADOOP DISTRIBUTED FILE SYSTEM):

Hadoop File System was developed using distributed file system design. It runs on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware.

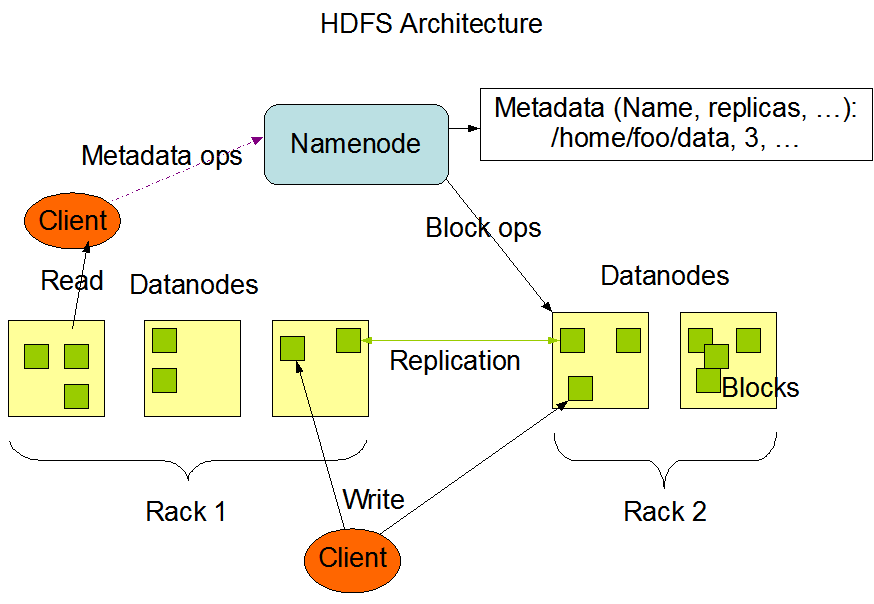
HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

**4.1 Features of HDFS**

* Distributed Storage – Data is stored in distributed manner
* Blocks – Data is split into blocks
* Replication – Blocks are replicated at different nodes
* High Availability – Data is highly available due to replication
* Data Reliability – Data is stored reliably in HDFS
* Fault tolerant – Data replication provides fault tolerance feature
* Scalability – Nodes in HDFS cluster can be increased on the fly
* High throughput access to application – Parallel processing provides high throughput access to application

**4.2 HDFS ARCHITECTURE:**

* Single Namenode which stores the metadata of the filesystem
* Multiple Datanodes which do the actual storage work
* Nodes are Arranged in racks and replicas of each block are stored in different racks such that in case of rack failure, we can still continue with processing since replicas are present in other racks.



**Fig. Hadoop Architecture**

**4.3 HDFS CLUSTER COMPONENTS:**

**HDFS Daemons**

There are two daemons which run for HDFS for storage of data:

* **Namenode**: This is the daemon that runs on all the masters. Name node stores metadata like filename, the number of blocks, number of replicas, a location of blocks, block IDs etc. This metadata is available in memory in the master for faster retrieval of data. In the local disk, a copy of metadata is available for persistence. So name node memory should be high as per the requirement. It should be deployed on reliable hardware.
* **Datanode**: This is the daemon that runs on the slave. These are actual worker nodes that store the data. These slave nodes are the actual worker nodes which do the tasks and serve read and write requests from the file system’s clients. They also perform block creation, deletion, and replication upon instruction from the NameNode. Once a block is written on a datanode, it replicates it to other datanode and process continues until the number of replicas mentioned is created. Datanodes can be deployed on commodity Hardware and we need not deploy them on very reliable hardware.

**BLOCK**:

HDFS splits huge files into small chunks known as blocks. Block is the smallest unit of data in a filesystem. We (client and admin) do not have any control on the block like block location. Namenode decides all such things.HDFS default block size is 128 MB which can be increased as per the requirement. This is unlike OS filesystem where the block size is 4 KB.If the data size is less than the block size, then block size will be equal to the data size. For example, if the file size is 129 MB, then 2 blocks will be created for it. One block will be of default size 128 MB and other will be 1 MB only and not 128 MB as it will waste the space (here block size is equal to data size). Hadoop is intelligent enough not to waste rest of 127 MB. So it is allocating 1 MB block only for 1 MB data.The major advantage of storing data in such block size is that it saves disk seek time and another advantage is in the case of processing as mapper processes 1 block at a time. So 1 mapper processes large data at a time.The file is split into blocks and each block is stored at different nodes with default 3 replicas of each block. Each replica of a block is stored at the different node to provide fault tolerant feature and the placement of these blocks on the different node is decided by Name node.

5. **PROJECT IMPLEMENTATION**

**5.1 Assumptions:**

• Hadoop Cluster is Running

• Ecosystem Products (Pig, Hive,sqoop) are installed

• H1B data is available on HDFS in CSV Format

**5.2 Prerequisites:**

The h1b data is in csv format in hadoop file system. But we need to remove white spaces,” ” and other punctuations in the file and also we need to trim 3 case –status i.e

PENDING QUALITY AND COMPLIANCE REVIEW, REJECTED, INVALIDATED to DENIED

**For pre-processing of data we use** **HIVE**.

6. **HIVE**

Apache Hive is a SQL Based data warehouse systembuilt on top of Hadoop for providing data summarization, query, and analysis. Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop.

**6.1 Steps for Conversion (Pre-processing – data cleansing)**

Created 3 tables in hive shown below

**1.** CREATE TABLE h1b\_applications(s\_no int,case\_status string,

employer\_name string, soc\_name string, job\_title string,

full\_time\_position string,prevailing\_wage int,year string, worksite

string, longitute double, latitute double )

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

"separatorChar" = ",",

"quoteChar" = "\""

) STORED AS TEXTFILE;

load data local inpath '/home/hduser/h1b.csv' overwrite into table

h1b\_applications;

**2.** CREATE TABLE h1b\_app2(s\_no int,case\_status string, employer\_name

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

string,prevailing\_wage int,year string, worksite string, longitute

double, latitute double )

row format delimited

fields terminated by '\t'

STORED AS TEXTFILE;

INSERT OVERWRITE TABLE h1b\_app2 SELECT regexp\_replace(s\_no, "\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(prevailing\_wage, "\t", ""), regexp\_replace(year, "\t",

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

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

where case\_status != "NA";

**3.** CREATE TABLE h1b\_final(s\_no int,case\_status string, employer\_name

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

string,prevailing\_wage int,year string, worksite string, longitute

double, latitute double )row format delimited

fields terminated by '\t'

STORED AS TEXTFILE;

INSERT OVERWRITE TABLE h1b\_final SELECT s\_no,

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,prevailing\_wage,year, worksite, longitute, latitute

FROM h1b\_app2;

**TASK 1**: **Which industry(SOC\_NAME) has the most number of Data Scientist positions?**

select soc\_name ,count(soc\_name) as soc\_count from h1b\_final where job\_title like'%DATA SCIENTIST%' group by soc\_name order by soc\_count desc limit 1;

**OUTPUT:**

****

**TASK 2: Find the most popular top 10 job positions for H1B visa applications for each year?**

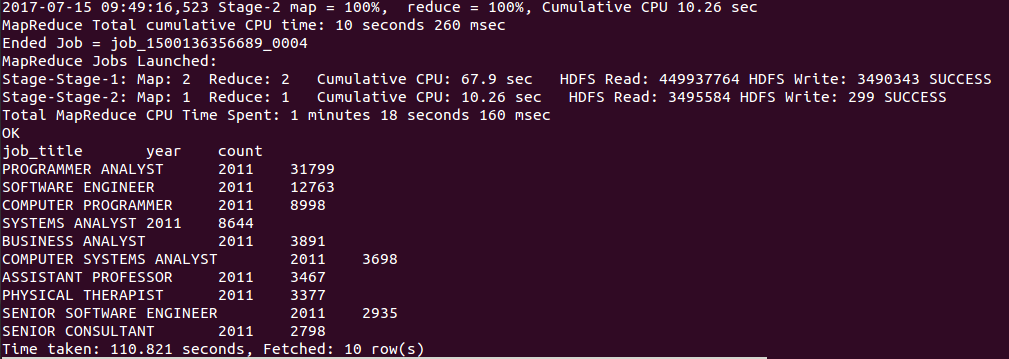
**a) for all the applications**

**b) for only certified applications.**

**a) for all the applications**

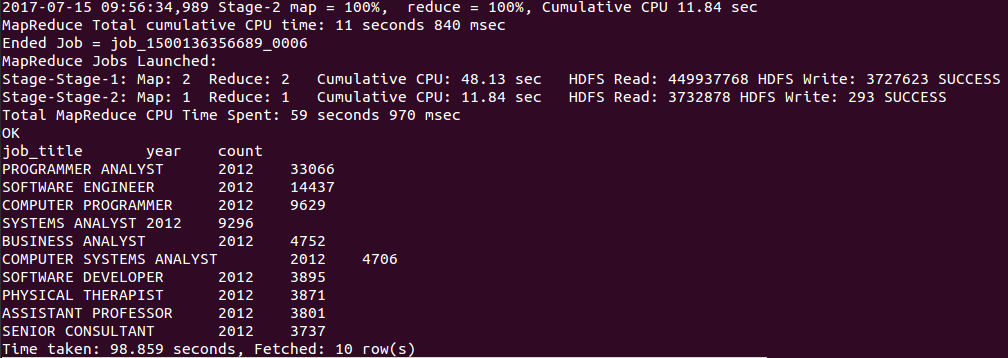
**For 2010 :**

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



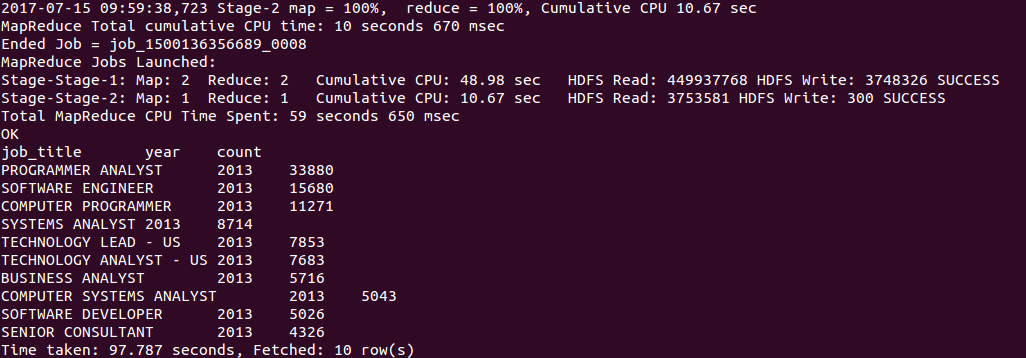
**For 2011 :**

select job\_title,year,count(job\_title) as count from h1b.h1b\_final where year=2012 group by job\_title,year order by count desc limit 10;

****

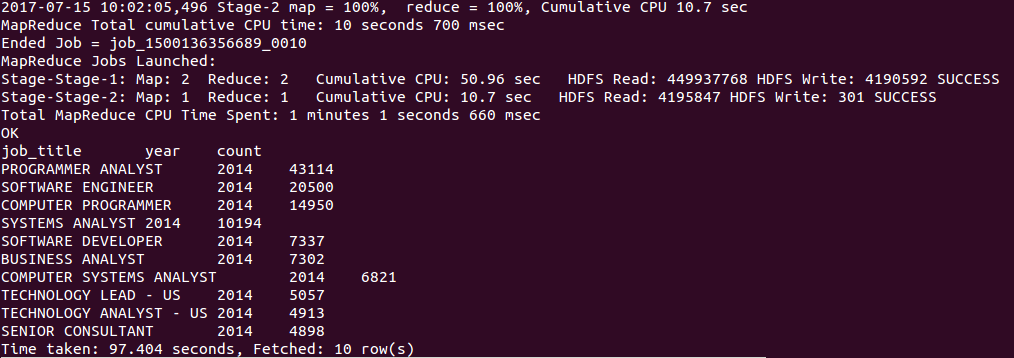
**For 2012 :**

select job\_title,year,count(job\_title) as count from h1b.h1b\_final where year=2013 group by job\_title,year order by count desc limit 10;

****

**For 2013 :**

select job\_title,year,count(job\_title) as count from h1b.h1b\_final where year=2014 group by job\_title,year order by count desc limit 10;

****

**For 2015 :**

select job\_title,year,count(job\_title) as count from h1b.h1b\_final where year=2015 group by job\_title,year order by count desc limit 10;



**For 2016 :**

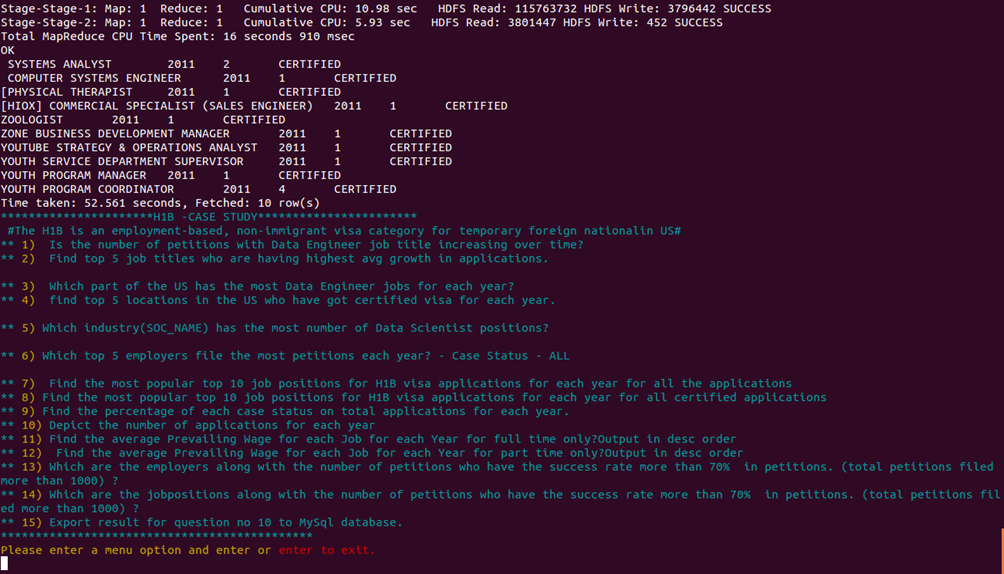
select job\_title,year,count(job\_title) as count from h1b.h1b\_final where year=2016 group by job\_title,year order by count desc limit 10;



**b) for only certified applications.**

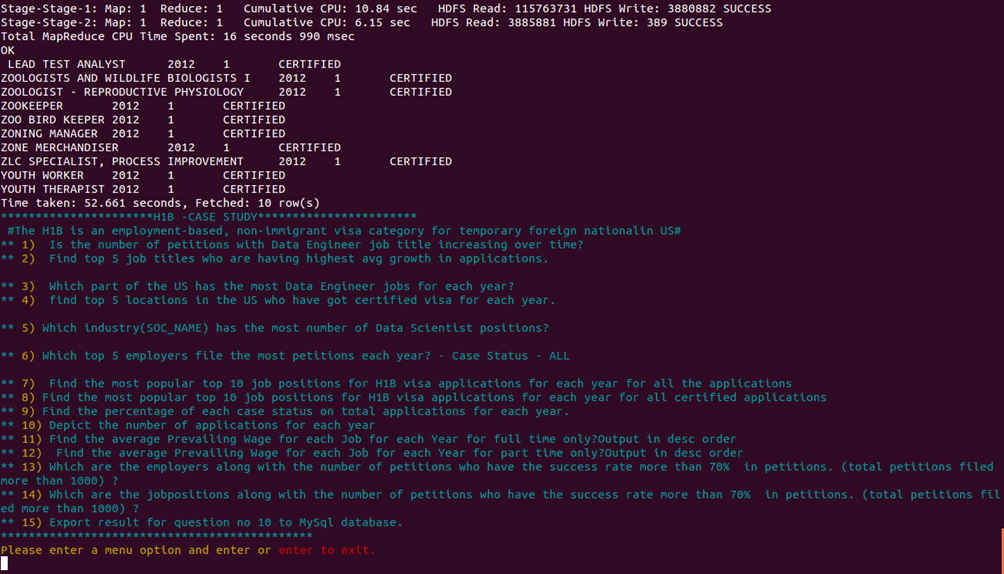
**For 2011 :**

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



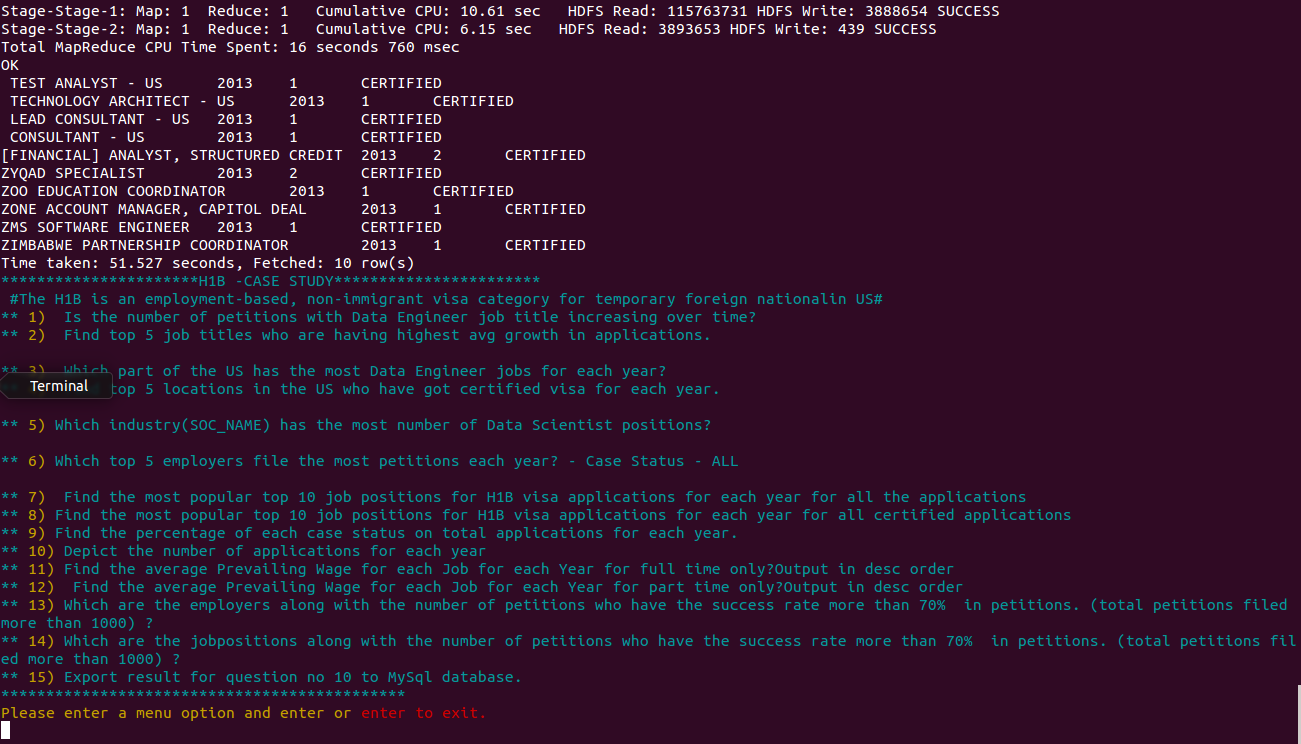
**For 2012 :**

select job\_title,year,count(job\_title),case\_status as count from h1b.h1b\_final where year=2012 and case\_status =='CERTIFIED' group by job\_title,year,case\_status order by count desc limit 10;



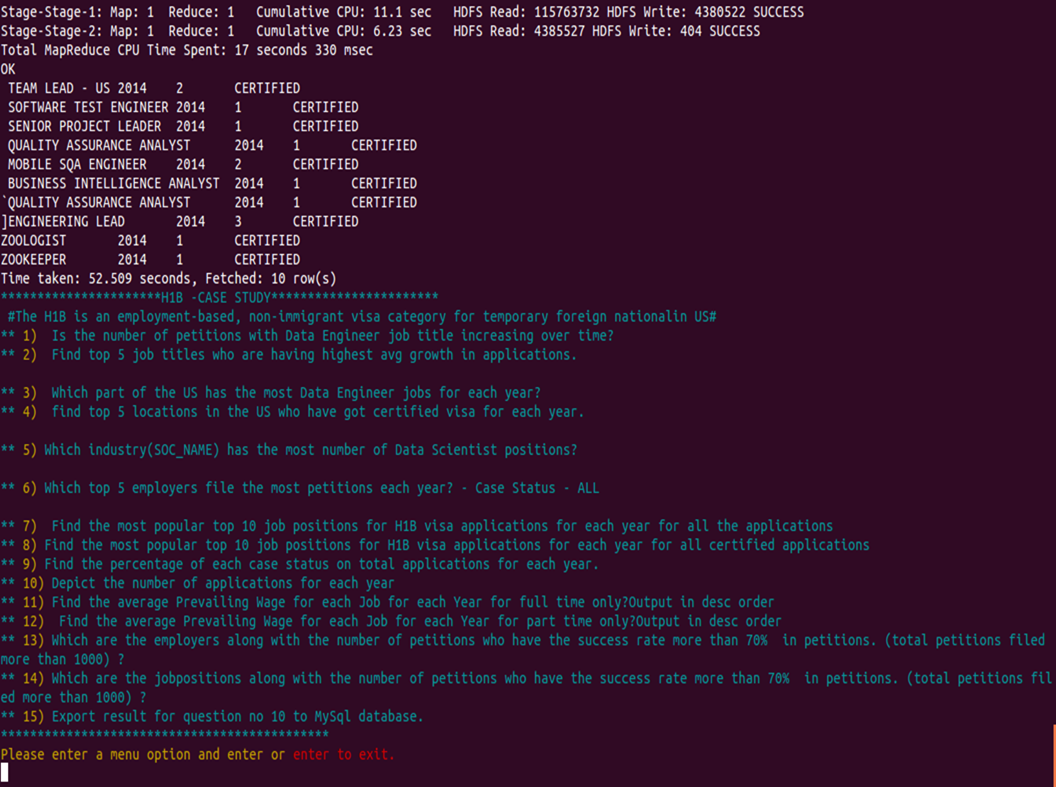
**For 2013 :**

select job\_title,year,count(job\_title),case\_status as count from h1b.h1b\_final where year=2013 and case\_status =='CERTIFIED' group by job\_title,year,case\_status order by count desc limit 10;



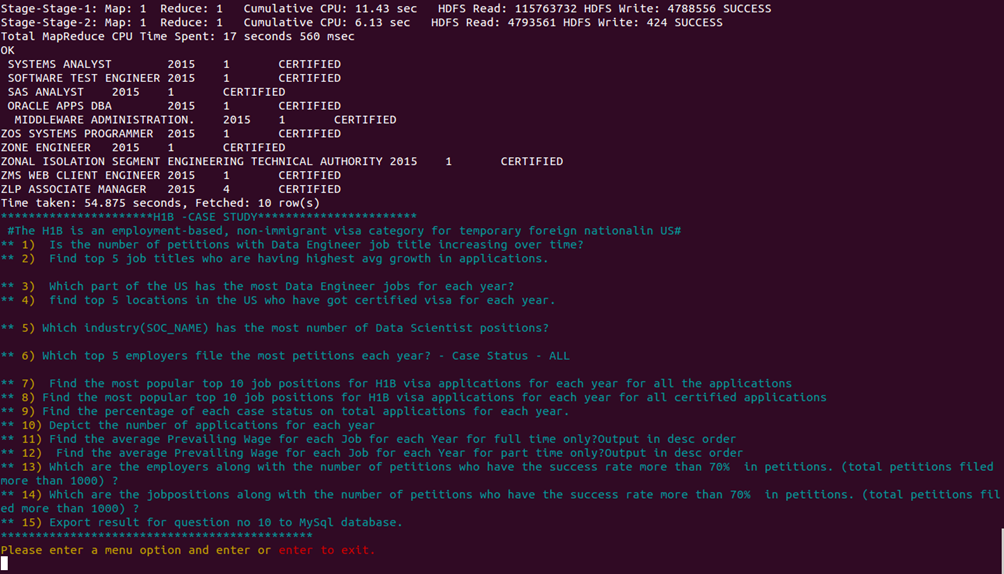
**For 2014 :**

select job\_title,year,count(job\_title),case\_status as count from h1b.h1b\_final where year=2013 and case\_status =='CERTIFIED' group by job\_title,year,case\_status order by count desc limit 10;



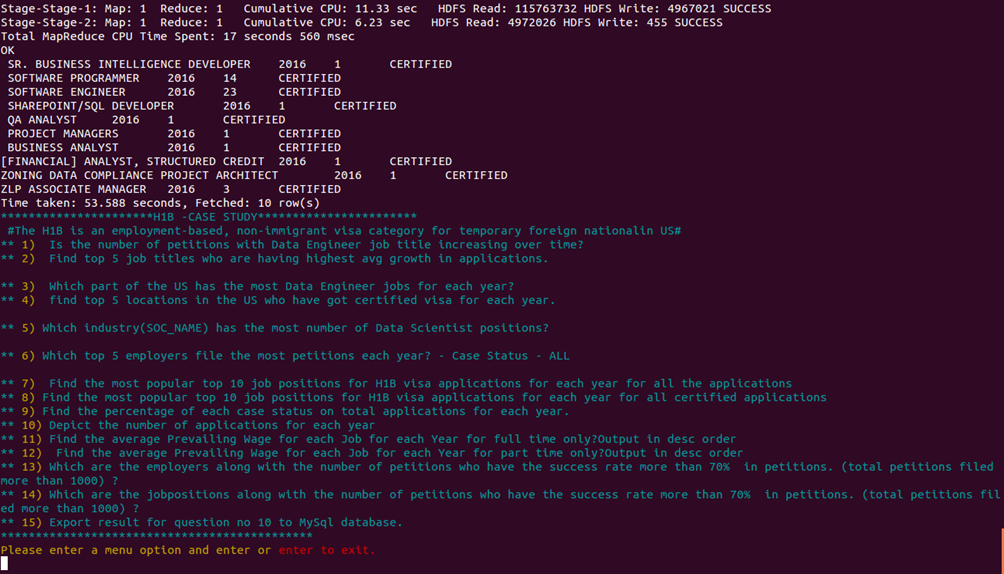
**For 2015 :**

select job\_title,year,count(job\_title),case\_status as count from h1b.h1b\_final where year=2015 and case\_status =='CERTIFIED' group by job\_title,year,case\_status order by count desc limit 10;



**For 2016 :**

select job\_title,year,count(job\_title),case\_status as count from h1b.h1b\_final where year=2016 and case\_status =='CERTIFIED' group by job\_title,year,case\_status order by count desc limit 10;



**TASK 3 : Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order.**

create table h1b\_finalByCat(s\_no int,case\_status string,employer\_name string,soc\_name string,job\_title string,prevailing\_wage int,year string,worksite string,longitute double,latitute double)

> partitioned by(full\_time\_position string)

> row format delimited fields

> terminated by'\t'

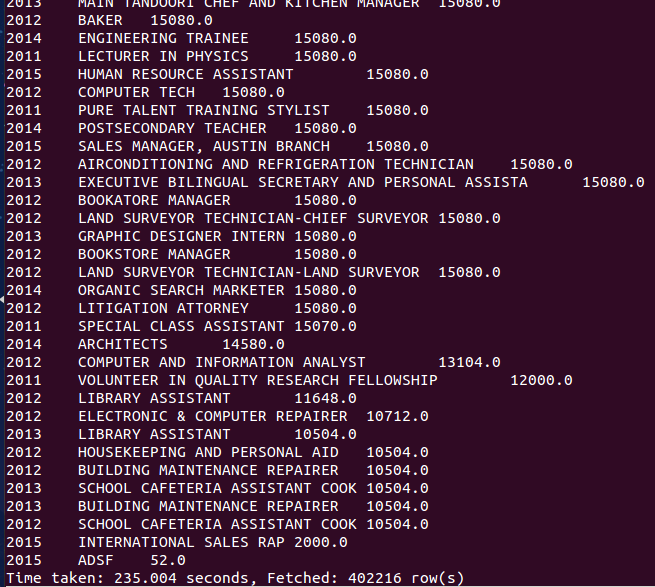
> stored as textfile;

hive (h1b)> SET hive.exec.dynamic.partition = true;

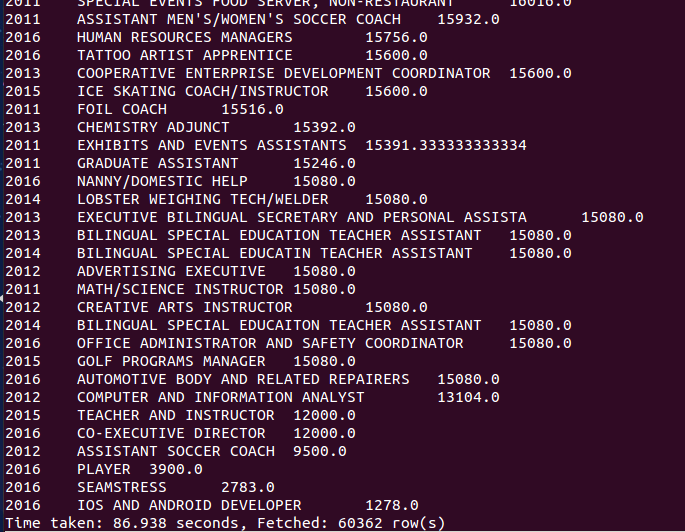
hive (h1b)> SET hive.exec.dynamic.partition.mode = nonstrict;

INSERT OVERWRITE TABLE h1b\_finalByCat PARTITION(full\_time\_position)select s\_no,case\_status,employer\_name,soc\_name,job\_title,prevailing\_wage,year,worksite,longitute,latitute,full\_time\_position from h1b\_final;

1. select year,job\_title,AVG(prevailing\_wage) as average from h1b\_finalByCat where full\_time\_position ='Y'and prevailing\_wage > 0 group by job\_title, year order by average desc;



1. select year,job\_title,AVG(prevailing\_wage) as average from h1b\_finalByCat where full\_time\_position ='N' and prevailing\_wage >0 group by job\_title,year order by average desc;

****

7. **PIG**

Apache Pig is a platform for analysing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets. Pig's language layer currently consists of a textual language called Pig Latin, which has the following key properties:

* Ease of programming. It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
* Optimization opportunities. The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
* Extensibility. Users can create their own functions to do special-purpose processing.

**TASK 1**

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

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

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

bag1 = load '/user/hive/warehouse/h1b.db/h1b\_final' using PigStorage() as (s\_no:int,case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:int,year:int,worksite:chararray,long:double,lati:double);

bag2 = filter bag1 by ($4 MATCHES '.\*DATA ENGINEER.\*');

bag3 = group bag2 by (year,worksite);

bag4 = foreach bag3 generate group,COUNT(bag2) as total;

bag5 = foreach bag4 generate FLATTEN(group),total;

bag6 = group bag5 by year;

top = foreach bag6 {

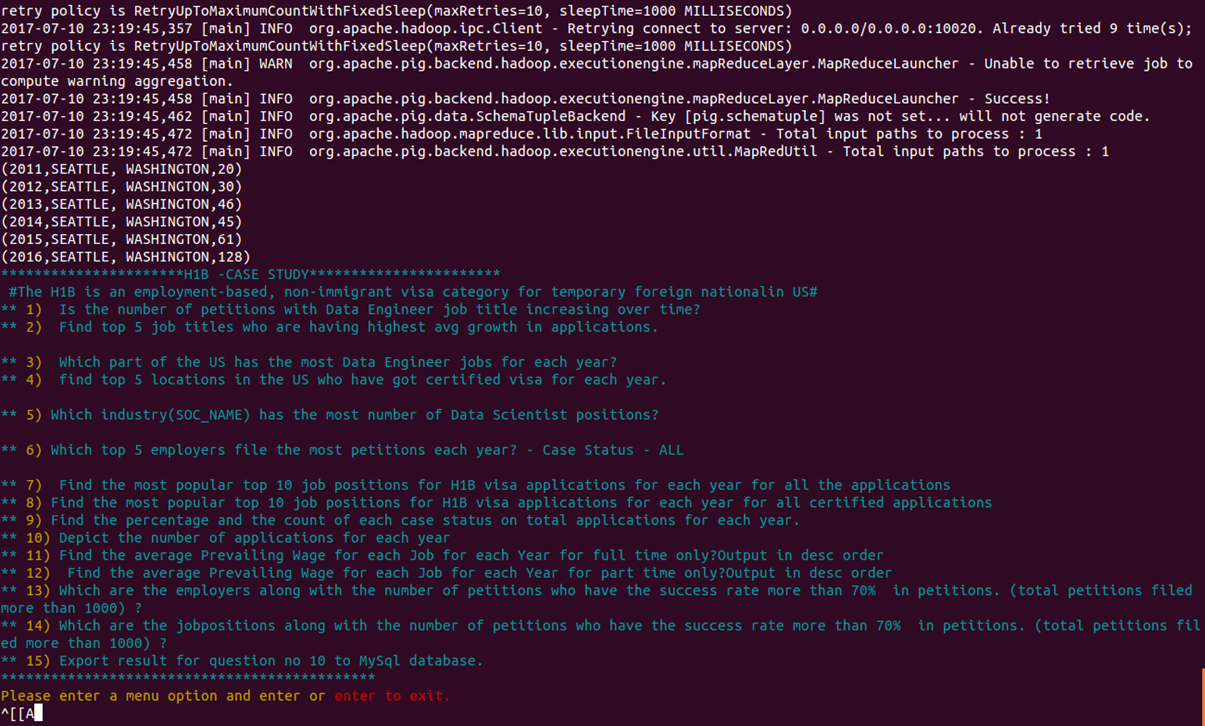
sorted = order bag5 by total desc;

top1 = limit sorted 1;

generate flatten(top1);

};

dump top;



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

bag1 = load '/user/hive/warehouse/h1b.db/h1b\_final' using PigStorage() as (s\_no:int,case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:int,year:int,worksite:chararray,long:double,lati:double);

bag2 = filter bag1 by ($1 MATCHES 'CERTIFIED');

bag3 = group bag2 by (year,worksite);

bag4 = foreach bag3 generate group,COUNT(bag2) as total;

bag5 = foreach bag4 generate FLATTEN(group),total;

bag6 = group bag5 by year;

bag7 = foreach bag6{

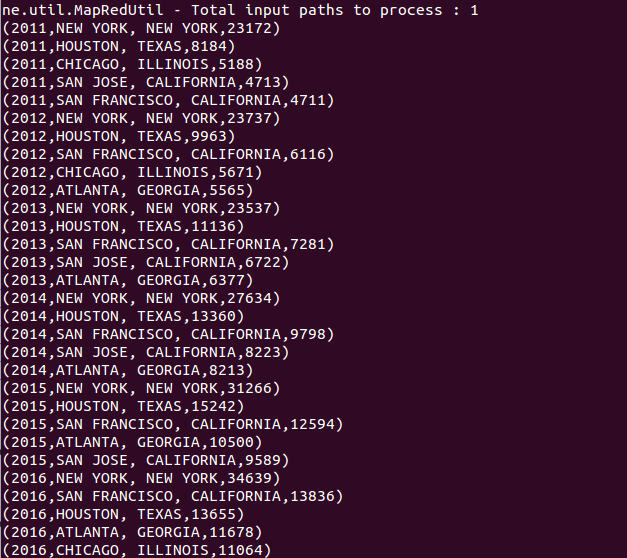
sorted = order bag5 by total desc;

top5 = limit sorted 5;

generate flatten(top5);

};

dump bag7;



**Task2 :**

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

split bag4 into bag5 if year ==2011,bag6 if year ==2012,bag7 if year ==2013,bag8 if year ==2014,bag9 if year ==2015,bag10 if year ==2016;

bag11 = order bag5 by $2 desc;

bag12 = limit bag11 5;

bag13 = order bag6 by $2 desc;

bag14 = limit bag13 5;

bag15 = order bag7 by $2 desc;

bag16 = limit bag15 5;

bag17 = order bag8 by $2 desc;

bag18 = limit bag17 5;

bag19 = order bag9 by $2 desc;

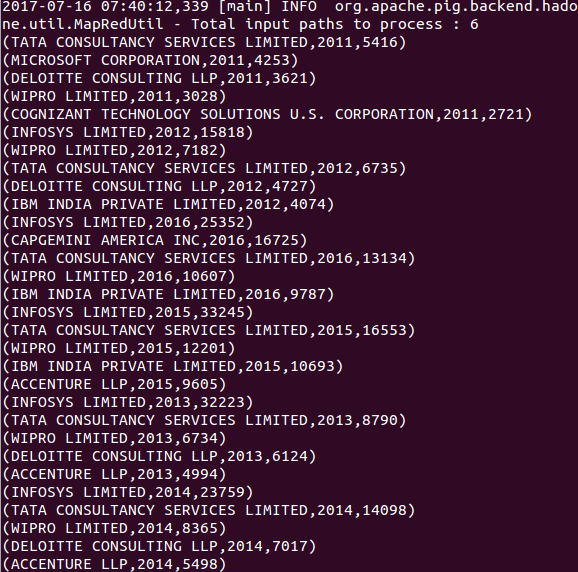
bag20 = limit bag19 5;

bag21 = order bag10 by $2 desc;

bag22 = limit bag21 5;

unionism = union bag12,bag14,bag16,bag18,bag20,bag22;

dump unionism;



**Task 3 :**

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

**COUNT and PERCENTAGE**

bag1 = load '/user/hive/warehouse/h1b.db/h1b\_final' using PigStorage() as (s\_no:int,case\_status:chararray,employer\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time\_position:chararray,prevailing\_wage:int,year:int,worksite:chararray,long:double,lati:double);

bag2 = group bag1 by year ;

bag4 = foreach bag2 generate group as year,bag1.case\_status;

bag5 = foreach bag4 generate year,FLATTEN($1);

split bag5 into bag6 if null::case\_status =='CERTIFIED-WITHDRAWN',bag7 if null::case\_status =='WITHDRAWN' ,bag8 if null::case\_status =='CERTIFIED',bag9 if null::case\_status =='DENIED';

bag10 = group bag6 by year ;

bag11 = group bag7 by year ;

bag12 = group bag8 by year;

bag13 = group bag9 by year;

bag14 = foreach bag10 generate group as year,COUNT(bag6.$1);

bag15 = foreach bag11 generate group as year,COUNT(bag7.$1);

bag16 = foreach bag12 generate group as year,COUNT(bag8.$1);

bag17 = foreach bag13 generate group as year,COUNT(bag9.$1);

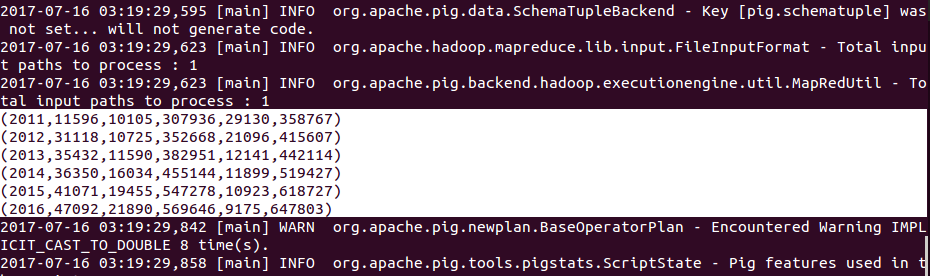
bag18 =join bag14 by $0,bag15 by $0,bag16 by $0,bag17 by $0;

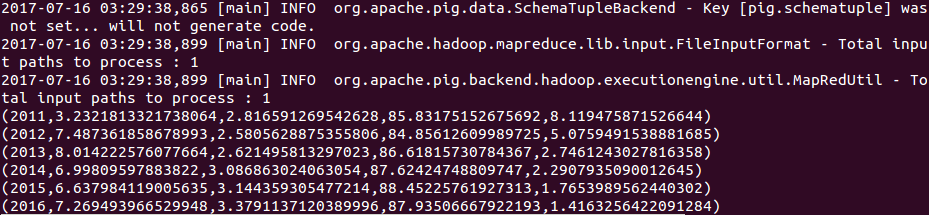
optimised = foreach bag18 generate $0,$1,$3,$5,$7,$1+$3+$5+$7;

dump optimised;

bag20 = foreach optimised generate $0,((double)$1/$5\*100),((double)$2/$5\*100),((double)$3/$5\*100),((double)$4/$5\*100);

dump bag20;





8. **MAPREDUCE**

MapReduce is a programming model suitable for processing of huge data. Hadoop is capable of running MapReduce programs written in various languages: Java, Ruby, Python, and C++. MapReduce programs are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster.

**MapReduce programs work in two phases:**

1. Map phase
2. Reduce phase.

Input to each phase are **key-value** pairs. In addition, every programmer needs to specify two functions: **map function** and **reduce function**

**TASK1**

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

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

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

import java.io.IOException;

import java.util.Map;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

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

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

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

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

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

public class Ques1a {

public static class myMapper extends Mapper<LongWritable, Text, IntWritable, LongWritable>

{

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

{

try{

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

String jobtitle = arr[4];

int year = Integer.parseInt(arr[7]);

if(jobtitle.equals("DATA ENGINEER"))

context.write(new IntWritable(year),new LongWritable(1));

}

catch(Exception e){

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

}

}

}

public static class myReducer extends Reducer<IntWritable, LongWritable, NullWritable, Text>

{

TreeMap<Integer, Long> tmap = new TreeMap<Integer, Long>();

static long temp = 0;

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

{

long sum=0;

for (LongWritable val : value)

{

sum+= val.get();

}

//context.write(key, new IntWritable(sum));

tmap.put(key.get(), sum);

}

@SuppressWarnings("rawtypes")

public void cleanup(Context context) throws IOException, InterruptedException

{

double growthperc = 0.0, sumgrowth = 0.0;

int count = 0;

//long temp = 0;

for( Map.Entry m:tmap.entrySet())

{

int mapKey = (int) m.getKey();

long mapValue = (long) m.getValue();

String tempResult = "";

if(temp != 0)

{

growthperc = (double)(mapValue - temp) \* 100 /temp;

tempResult = String.format("%.2f", growthperc);

temp = mapValue;

sumgrowth+=growthperc;

context.write(NullWritable.get(), new Text("" + mapKey + "\t" + mapValue + "\t \t" + tempResult + " %"));

}

else

{

String tempString = "Year" + "\t" + "Total\_Application" + "\t" + "Growth(%)" + "\n" + mapKey + "\t" + mapValue;

temp = mapValue;

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

}

count++;

}

String tempResult = "\n\n Average Growth -> " + String.format("%.2f", sumgrowth/count) + " %";

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

}

}

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

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

job.setJarByClass(Ques1a.class);

job.setMapperClass(myMapper.class);

job.setReducerClass(myReducer.class);

job.setMapOutputKeyClass(IntWritable.class);

job.setMapOutputValueClass(LongWritable.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

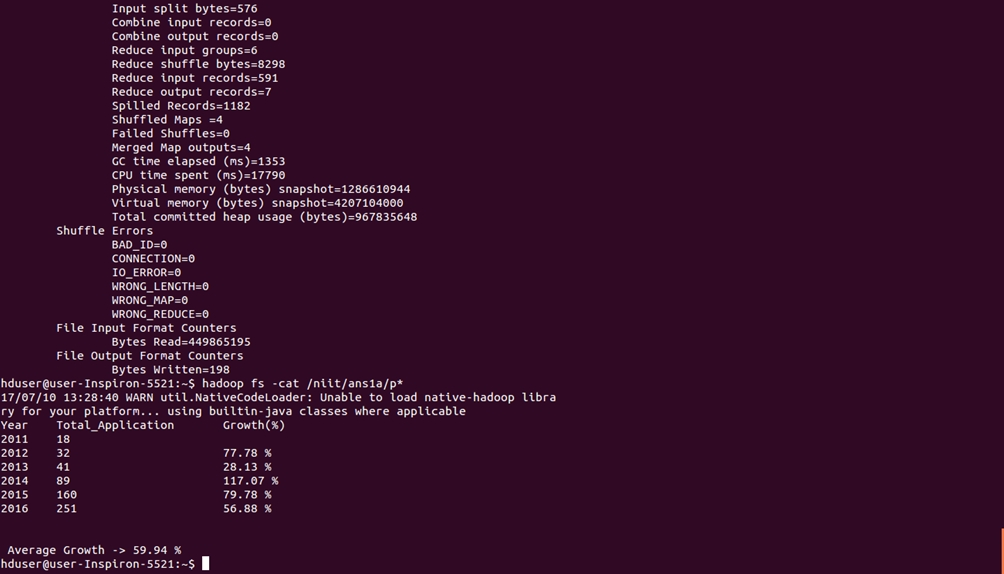
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}



|  |  |  |  |
| --- | --- | --- | --- |
| Year | Total\_Application | Growth(%) | DELTA Y/INCREASE |
| 2011 | 18 |  |  |
| 2012 | 32 | 78% | 14 |
| 2013 | 41 | 28% | 9 |
| 2014 | 89 | 117% | 48 |
| 2015 | 160 | 80% | 71 |
| 2016 | 251 | 57% | 91 |

1. **Find top 5 job titles who are having highest avg growth in applications**.

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

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

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

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

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

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

public class Ques1b {

public static class myMapper extends Mapper<LongWritable, Text, Text, IntWritable>

{

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

{

try{

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

int year = Integer.parseInt(arr[7]);

context.write(new Text(arr[4]), new IntWritable(year));

}catch(Exception e){

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

}

}

}

public static class myReducer extends Reducer<Text, IntWritable, NullWritable, Text>

{

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

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

{

long year11 = 0, year12 = 0, year13 = 0, year14 = 0, year15 =0, year16 =0;

double cyc1 = 0.0, cyc2 = 0.0, cyc3 = 0.0, cyc4 = 0.0, cyc5 = 0.0, finavg = 0.0;

for (IntWritable val : value)

{

if(val.get() == 2011)

year11++;

else if(val.get() == 2012)

year12++;

else if(val.get() == 2013)

year13++;

else if(val.get() == 2014)

year14++;

else if(val.get() == 2015)

year15++;

else if(val.get() == 2016)

year16++;

}

cyc1 = (year11!=0 ? ((year12 - year11) \* 100 /year11) : 0);

cyc2 = (year12!=0 ? ((year13 - year12) \* 100 /year12) : 0);

cyc3 = (year13!=0 ? ((year14 - year13) \* 100 /year13) : 0);

cyc4 = (year14!=0 ? ((year15 - year14) \* 100 /year14) : 0);

cyc5 = (year15!=0 ? ((year16 - year15) \* 100 /year15) : 0);

finavg = (cyc1 + cyc2 + cyc3 + cyc4 + cyc5)/5;

String temp = key + "\t" + "Cycle Growth -> ( " + cyc1 + "," + cyc2 + "," + cyc3 + "," + cyc4 + "," + cyc5 + ") \t" + "Average Growth -> " + finavg;

if((year11 + year12 + year13 + year14 + year15 + year16) > 1000)

tmap.put(finavg, temp);

/\*This condition is to exclude records like below

\* 2011 1, 2013 2, 2015 1, 2014 2, 2016 249,

\* BUSINESS ANALYST 2 Cycle Growth -> ( -100.0,0.0,0.0,-50.0,24800.0) Average Growth -> 4930.0

\*/

if(tmap.size() > 5)

tmap.remove(tmap.firstKey());

}

public void cleanup(Context context) throws IOException, InterruptedException

{

for( String m:tmap.descendingMap().values())

{

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

}

}

}

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

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

job.setJarByClass(Ques1b.class);

job.setMapperClass(myMapper.class);

job.setReducerClass(myReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

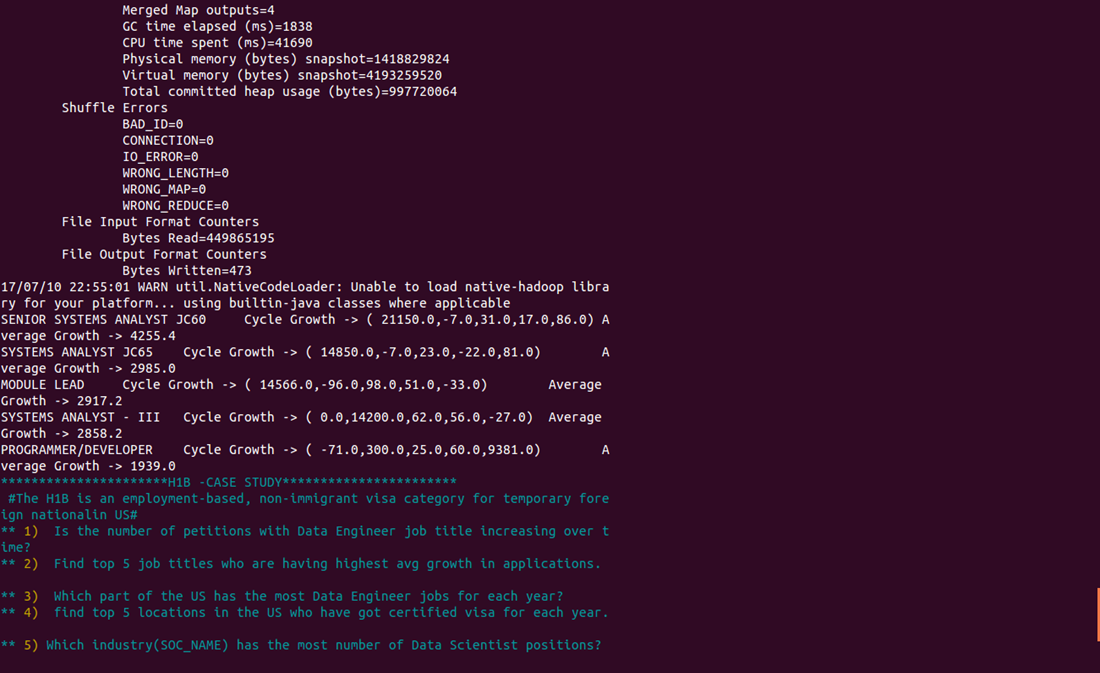
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}



**Task2** : **Create a bar graph to depict the number of applications for each year.**

import org.apache.hadoop.io.IntWritable;

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.\*;

import java.io.\*;

public class Ques7 {

public static class h1bMapper extends

Mapper<LongWritable, Text, Text, IntWritable> {

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

try {

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

context.write(new Text(str[7]), new IntWritable(1));

} catch (Exception e) {

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

}

}

}

public static class h1bReducer extends

Reducer<Text, IntWritable, Text, IntWritable> {

IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

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

Job job = Job.getInstance(conf, "count");

job.setJarByClass(Ques7.class);

job.setReducerClass(h1bReducer.class);

job.setMapperClass(h1bMapper.class);

// job.setNumReduceTasks(0);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

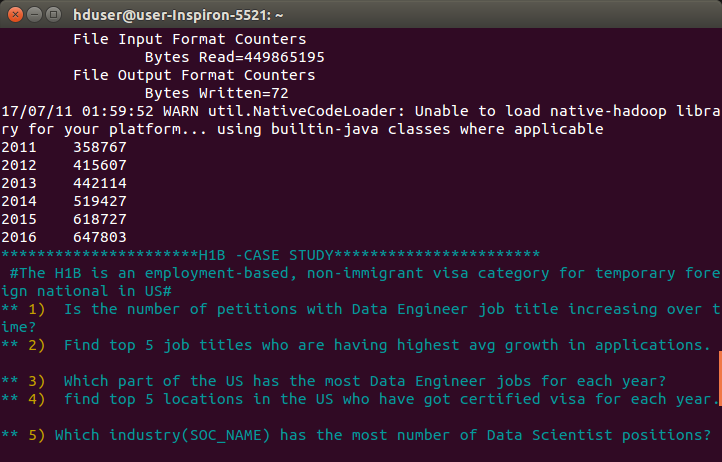
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}



**Task 3:**

**Which are the employers along with the number of petitions who have the success rate more than 70% in petitions. (total petitions filed more than 1000) ?**

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.FloatWritable;

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.\*;

import java.io.\*;

public class Ques9 {

public static class h1bMapper1 extends Mapper<LongWritable,Text,Text,Text>

{

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

{

try{

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

context.write(new Text(abc[2]), value);

}catch(Exception e){

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

}

}

}

public static class h1breducer1 extends Reducer<Text,Text,Text,FloatWritable>

{

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

{int count = 0,count1 = 0,count2 = 0;

float success;

long myval = 0 ;

for(Text t:value)

{

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

count++;

if(parts[1].equals("CERTIFIED"))

{

count1++;

}

if(parts[1].equals("CERTIFIED-WITHDRAWN"))

{

count2++;

}

}

if(count > 1000)

{

success = ((count1+count2)\*100)/count;

if(success > 70.0)

{

myval = (long) success;

}

context.write(key,new FloatWritable(myval));

}

}

}

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

Configuration conf = new Configuration();

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

Job job=Job.getInstance(conf, "count");

job.setJarByClass(Ques9.class);

job.setReducerClass(h1breducer1.class);

job.setMapperClass(h1bMapper1.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(FloatWritable.class);

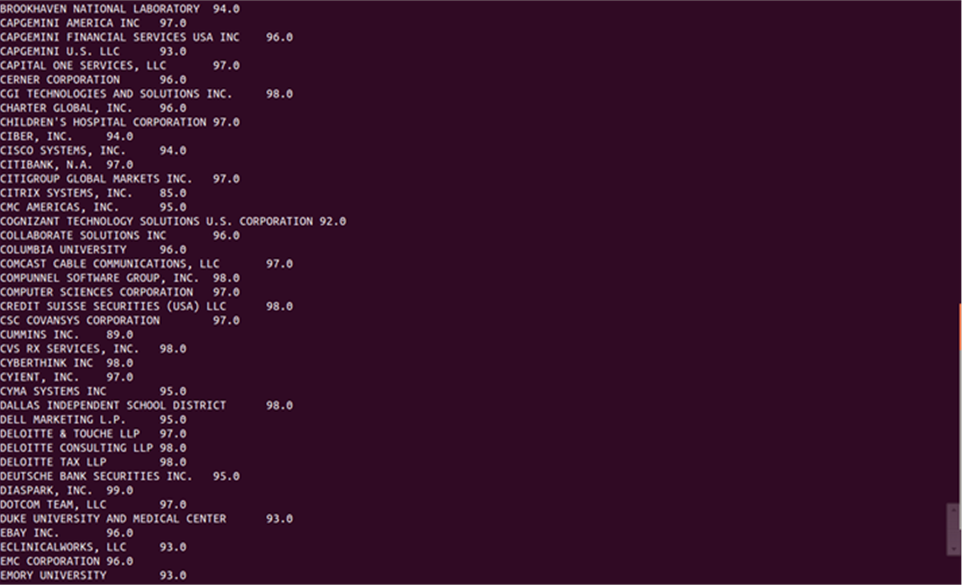
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}

****

**Task 4 : SUCCESS RATE =(CERTIFIED +CERTIFIED-WITHDRWAN)/TOTAL \*100**

**10) Which are the job positions along with the number of petitions which have the success rate more than 70% in petitions (total petitions filed more than 1000)?**

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.FloatWritable;

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.\*;

import java.io.\*;

public class Ques10 {

public static class h1bMapper1 extends

Mapper<LongWritable, Text, Text, Text> {

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

throws IOException, InterruptedException {

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

try{

context.write(new Text(abc[2]), value);

}catch(Exception e){

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

}

}

}

public static class h1breducer1 extends

Reducer<Text, Text, Text, FloatWritable> {

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

throws IOException, InterruptedException {

int count = 0, count1 = 0, count2 = 0;

float success;

long myval = 0;

for (Text t : value) {

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

count++;

if (parts[1].equals("CERTIFIED")) {

count1++;

}

if (parts[1].equals("CERTIFIED-WITHDRAWN")) {

count2++;

}

}

if (count > 1000) {

success = ((count1 + count2) \* 100) / count;

if (success > 70.0) {

myval = (long) success;

}

context.write(key, new FloatWritable(myval));

}

}

}

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

Configuration conf = new Configuration();

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

Job job = Job.getInstance(conf, "count");

//job.setNumReduceTasks(0);

job.setJarByClass(Ques10.class);

job.setMapperClass(h1bMapper1.class);

job.setReducerClass(h1breducer1.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(FloatWritable.class);

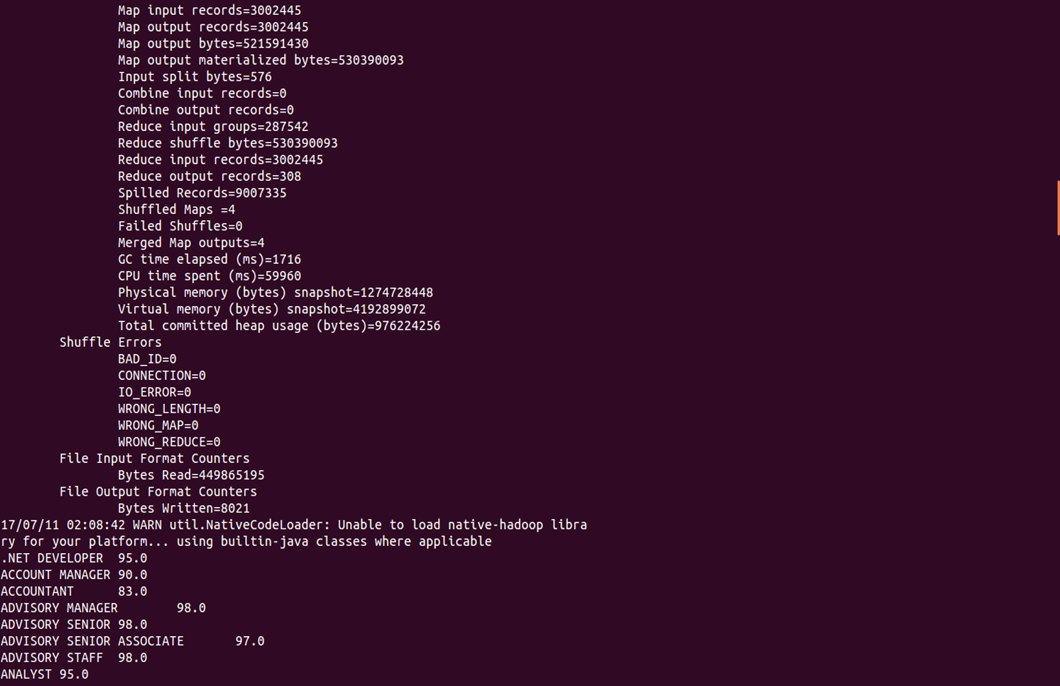
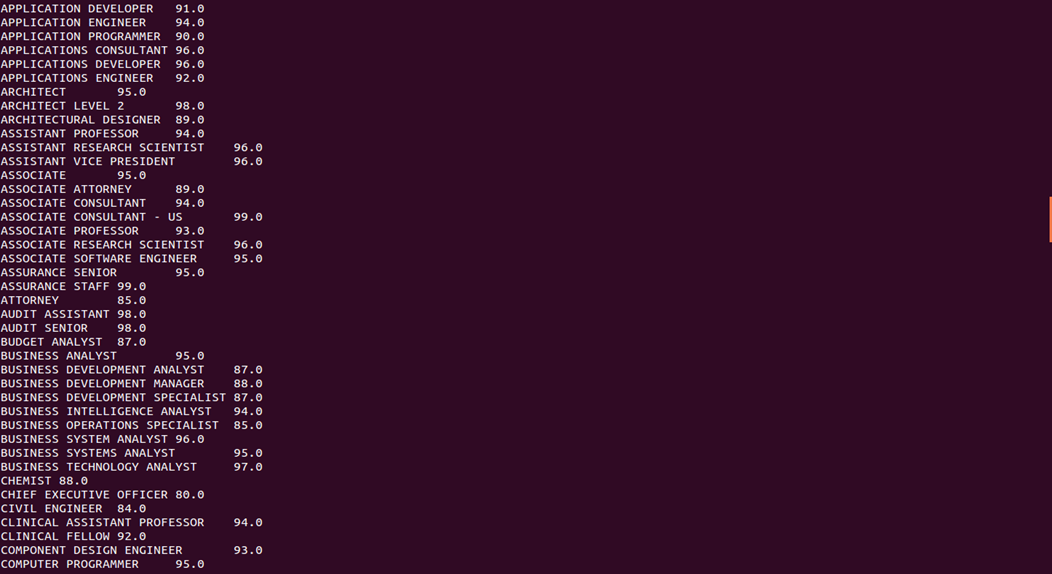
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}

****

**APACHE SQOOP**

Apache Sqoop(TM) is a tool designed for efficiently transferring bulk data between [Apache Hadoop](http://hadoop.apache.org/) and structured datastores such as relational databases.

**TASK1**

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

sqoop export --connect jdbc:mysql://localhost/h1b --username root --password '1231' --table exportques10 --export-dir /h1b/ques10output --input-fields-terminated-by '\t' ;



**Making Menu**

To make the application interactive and user friendly we have to make a User Interface form where the user can select the desired operation. So the user don’t have to go through or run the separate commands.

For this we have used Linux shell scripting:

#!/bin/bash

show\_menu()

{

NORMAL=`echo "\033[m"`

MENU=`echo "\033[36m"` #Red

NUMBER=`echo "\033[33m"` #yellow

FGRED=`echo "\033[41m"`

RED\_TEXT=`echo "\033[31m"`

ENTER\_LINE=`echo "\033[33m"`

echo -e "${MENU}\e[1m\e[47m\e[95m\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*H1B DashBoard\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*${NORMAL}"

echo -e "${MENU} \e[39m#Use the following options to perform the operation on H1B dataset # ${NORMAL}"

echo -e "${MENU}${NUMBER} 1)${MENU} Is the number of petitions with Data Engineer job title increasing over time? ${NORMAL}"

echo -e "${MENU}${NUMBER} 2)${MENU} Find top 5 job titles who are having highest avg growth in applications.

${NORMAL}"

echo -e "${MENU}${NUMBER} 3)${MENU} Which part of the US has the most Data Engineer jobs for each year? ${NORMAL}"

echo -e "${MENU}${NUMBER} 4)${MENU} find top 5 locations in the US who have got certified visa for each year.

${NORMAL}"

echo -e "${MENU}${NUMBER} 5)${MENU} Which industry(SOC\_NAME) has the most number of Data Scientist positions?

${NORMAL}"

echo -e "${MENU}${NUMBER} 6)${MENU} Which top 5 employers file the most petitions each year? - Case Status - ALL

${NORMAL}"

echo -e "${MENU}${NUMBER} 7)${MENU} Find the most popular top 10 job positions for H1B visa applications for each year for all the applications${NORMAL}"

echo -e "${MENU}${NUMBER} 8)${MENU} Find the most popular top 10 job positions for H1B visa applications for each year for all certified applications${NORMAL}"

echo -e "${MENU}${NUMBER} 9)${MENU} Find the percentage of each case status on total applications for each year. "${NORMAL}

echo -e "${MENU}${NUMBER} 10)${MENU} Depict the number of applications for each year ${NORMAL}"

echo -e "${MENU}${NUMBER} 11)${MENU} Find the average Prevailing Wage for each Job for each Year for full time only?Output in desc order ${NORMAL}"

echo -e "${MENU}${NUMBER} 12)${MENU} Find the average Prevailing Wage for each Job for each Year for part time only?Output in desc order ${NORMAL}"

echo -e "${MENU}${NUMBER} 13)${MENU} Which are the employers along with the number of petitions who have the success rate more than 70% in petitions. (total petitions filed more than 1000) ? ${NORMAL}"

echo -e "${MENU}${NUMBER} 14)${MENU} Which are the jobpositions along with the number of petitions who have the success rate more than 70% in petitions. (total petitions filed more than 1000) ? ${NORMAL}"

echo -e "${MENU}${NUMBER} 15)${MENU} Export result for question no 10 to MySql database.${NORMAL}"

echo -e "${MENU}\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*${NORMAL}"

echo -e "${ENTER\_LINE}Please enter a menu option and enter or ${RED\_TEXT}enter to exit. ${NORMAL}"

read opt

}

function option\_picked()

{

COLOR='\033[01;31m' # bold red

RESET='\033[00;00m' # normal white

MESSAGE="$1" #modified to post the correct option selected

echo -e "${COLOR}${MESSAGE}${RESET}"

}

function getpinCodeBank(){

echo "in getPinCodebank"

echo $1

echo $2

#hive -e "Select \* from AppData where PinCode = $1 AND Bank = '$2'"

}

clear

show\_menu

while [ opt != '' ]

do

if [[ $opt = "" ]]; then

exit;

else

case $opt in

1) clear;

hadoop jar h1b.jar Ques1a /user/hive/warehouse/h1b.db/h1b\_final /h1b/output/ans1a;

hadoop fs -cat /h1b/ans1a/p\*;

show\_menu;

;;

2) clear;

hadoop jar h1b.jar Ques1b /user/hive/warehouse/h1b.db/h1b\_final /h1b/output/ans1b;

hadoop fs -cat /h1b/ans1b/p\*;

show\_menu;

;;

3) clear;

pig pig/ques2a.pig;

show\_menu;

;;

4) clear;

pig pig/ques2b.pig;

show\_menu;

;;

5) clear;

hive -f hive/ques3.sql;

show\_menu;

;;

6) clear;

pig pig/ques4.pig;

show\_menu;

;;

7) clear;

option\_picked "Search Based on Year";

echo -e "${MENU}\*\*${NUMBER} 1)${MENU} 2011 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 2)${MENU} 2012 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 3)${MENU} 2013 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 4)${MENU} 2014 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 5)${MENU} 2015 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 6)${MENU} 2016 ${NORMAL}"

read n

case $n in

1) echo "For 2011"

hive -f hive/ques5/ques5A2011.sql;

;;

2) echo "For 2012"

hive -f hive/ques5/ques5A2012.sql;

;;

3) echo "for 2013"

hive -f hive/ques5/ques5A2013.sql;

;;

4) echo "For 2014"

hive -f hive/ques5/ques5A2014.sql;

;;

5) echo "For 2015"

hive -f hive/ques5/ques5A2015.sql;

;;

6) echo "For 2016"

hive -f hive/ques5/ques5A2016.sql;

;;

\*) echo "Please Select one among the option[1-6]";;

esac

show\_menu;

;;

8) clear;

option\_picked "Search Based on Year";

echo -e "${MENU}\*\*${NUMBER} 1)${MENU} 2011 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 2)${MENU} 2012 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 3)${MENU} 2013 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 4)${MENU} 2014 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 5)${MENU} 2015 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 6)${MENU} 2016 ${NORMAL}"

read n

case $n in

1) echo "For 2011"

hive -f hive/ques5/ques5B2011.sql;

;;

2) echo "For 2012"

hive -f hive/ques5/ques5B2012.sql;

;;

3) echo "for 2013"

hive -f hive/ques5/ques5B2013.sql;

;;

4) echo "For 2014"

hive -f hive/ques5/ques5B2014.sql;

;;

5) echo "For 2015"

hive -f hive/ques5/ques5B2015.sql;

;;

6) echo "For 2016"

hive -f hive/ques5/ques5B2016.sql;

;;

\*) echo "Please Select one among the option[1-6]";;

esac

show\_menu;

;;

9) clear;

pig pig/ques6.pig ;

show\_menu;

;;

10) clear;

hadoop jar h1b.jar Ques7 /user/hive/warehouse/h1b.db/h1b\_final h1b/output/ans7;

hadoop fs -cat /output/ans7/p\*;

show\_menu;

;;

11) clear;

hive -e "select year,job\_title,AVG(prevailing\_wage) as average from h1b.h1b\_finalByCat where full\_time\_position ='Y'and prevailing\_wage >0 group by job\_title,year order by average desc;"

show\_menu;

;;

12) clear;

hive -e "select year,job\_title,AVG(prevailing\_wage) as average from h1b.h1b\_finalByCat where full\_time\_position ='N'and prevailing\_wage >0 group by job\_title,year order by average desc;"

show\_menu;

;;

13) clear;

hadoop jar h1b.jar Ques9 /user/hive/warehouse/h1b.db/h1b\_final h1b/output/ans9;

hadoop fs -cat /output/ans9/p\*;

show\_menu;

;;

14) clear;

hadoop jar h1b.jar Ques10 /user/hive/warehouse/h1b.db/h1b\_final h1b/output/ans10;

hadoop fs -cat /output/ans10/p\*;

show\_menu;

;;

15) clear;

option\_picked "Now lets export the Ans10 to MySql";

sqoop export --connect jdbc:mysql://localhost/h1b --username root --password '1231' --table exportques10 --export-dir /h1b/ques10output --input-fields-terminated-by '\t' ;

;;

\n) exit;

;;

\*) clear;

option\_picked "Pick an option from the menu";

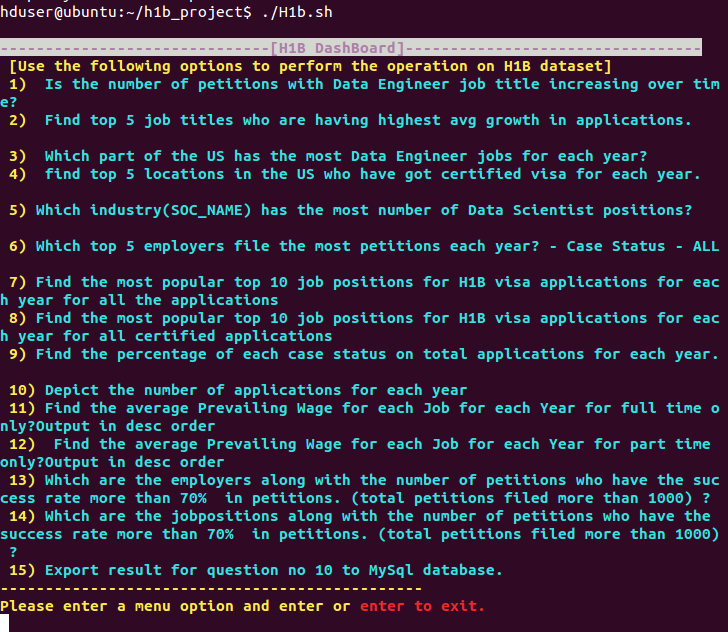
show\_menu;

;;

esac

fi

done



# 

# CONCLUSIONS:

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

* Though MapReduce code written in Java makes the complex analysis quite easy, Hive and Pig are easier to retrieve data efficiently and lines of code is also very less compared to MapReduce programs.
* The no of petitions filed for data engineering is increasing over time but the max growth rate(% INCREASE) is in 2014 from previous year(2013)
* Certified visa increases over year and denied percentage decreases.
* Statistians (SOC\_name) has the most number of Data Scientist positions.
* Total number of application increases over year.