PROJECT DOCUMENTATION

Name: MATHAN KUMAR.T
Student id: S180030100327
Batch: PDTBD

ACKNOWLEDGEMENT

I find immense pleasure to convey my sincere and grateful thanks to **NIIT** and the management for providing necessary facilities in carrying out this project.
I greatly indebted to my Tech Mentor **Ms. Amirtha,** the batch instructor **Mr. Annu Sharma** and the SLT faculty **Mr. Sandeep** for constant support throughout the course and also for useful suggestions, constant encouragement and kind advice in bringing out this project as a success.
I express my regards and sincere thanks to the Academic Leader **Mr. Oliver** for providing all necessary resource to complete the project. I extend my thanks to all staff members of NIIT for their kind co-operation for the completion of the project successfully. I am grateful to thank my family and all my friends for their valuable feedback, encouragement and suggestions.

1)INTRODUCTION TO BIG DATA:

Big data is data sets that are so voluminous and complex that traditional data processing application software are inadequate to deal with them.

Big data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating and information privacy. There are three dimensions to big data known as Volume, Variety and Velocity.

VOLUME:

The quantity of generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.

VARIETY:

The type and nature of the data. This helps people who analyse it to effectively use the resulting insight.

VELOCITY:

In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.

VARIABILITY:

Inconsistency of the data set can hamper processes to handle and manage it.

VERACITY:

The data quality of captured data can vary greatly, affecting the accurate analysis.

2)APACHE HADOOP

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing. It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover, it can be scaled up just by adding nodes in the cluster.

3)HDFS:

When a dataset outgrows the storage capacity of a single physical machine, it becomes necessary to partition it across a number of separate machines. Filesystems that manage the storage across a network of machines are called distributed filesystems. Since they are network based, all the complications of network programming kick in, thus making distributed filesystems more complex than regular disk filesystems. For example, one of the biggest challenges is making the filesystem tolerate node failure without suffering data loss.

Hadoop comes with a distributed filesystem called HDFS, which stands for Hadoop Distributed Filesystem. HDFS is Hadoop’s flagship filesystem and is the focus of this chapter, but Hadoop actually has a general-purpose filesystem abstraction, so we’ll see along the way how Hadoop integrates with other storage systems (such as the local filesystem and Amazon S3).

HADOOP ECOSYSTEM

Hadoop Distributed File System (HDFS)

It is the most important component of Hadoop Ecosystem. **HDFS** is the primary storage system of Hadoop. Hadoop distributed file system (HDFS) is a java based file system that provides scalable, fault tolerance, reliable and cost efficient data storage for **Big data**. HDFS is a distributed file system that runs on commodity hardware. HDFS is already configured with default configuration for many installations. Most of the time for large clusters configuration is needed. Hadoop interact directly with HDFS by shell-like commands.

Map Reduce

Hadoop Map Reduce is the core component of Hadoop which provides data processing. MapReduce is a software framework for easily writing applications that process the vast amount of structured and unstructured data stored in the Hadoop Distributed File system. MapReduce programs are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster. Thus, it improves the speed and reliability of cluster this parallel processing.

4) YARN:

Apache YARN (Yet Another Resource Negotiator) is Hadoop's cluster resource management system. YARN was introduced in Hadoop 2 to improve the MapReduce implementation, but it is general enough to support other distributed computing paradigms as well. YARN provides APIs for requesting and working with cluster resources, but these APIs are not typically used directly by user code. Instead, users write to higher-level APIs provided by distributed computing frameworks, which themselves are built on YARN and hide the resource management details from the user. The situation is illustrated in Figure, which shows some distributed computing frameworks (MapReduce, Spark, and so on) running as YARN applications on the cluster compute layer (YARN) and the cluster storage layer (HDFS and HBase).

There is also a layer of applications that build on the frameworks shown in Figure. Pig, Hive, and Crunch are all examples of processing frameworks that run on MapReduce, Spark, or Tez (or on all three), and don't interact with YARN directly.

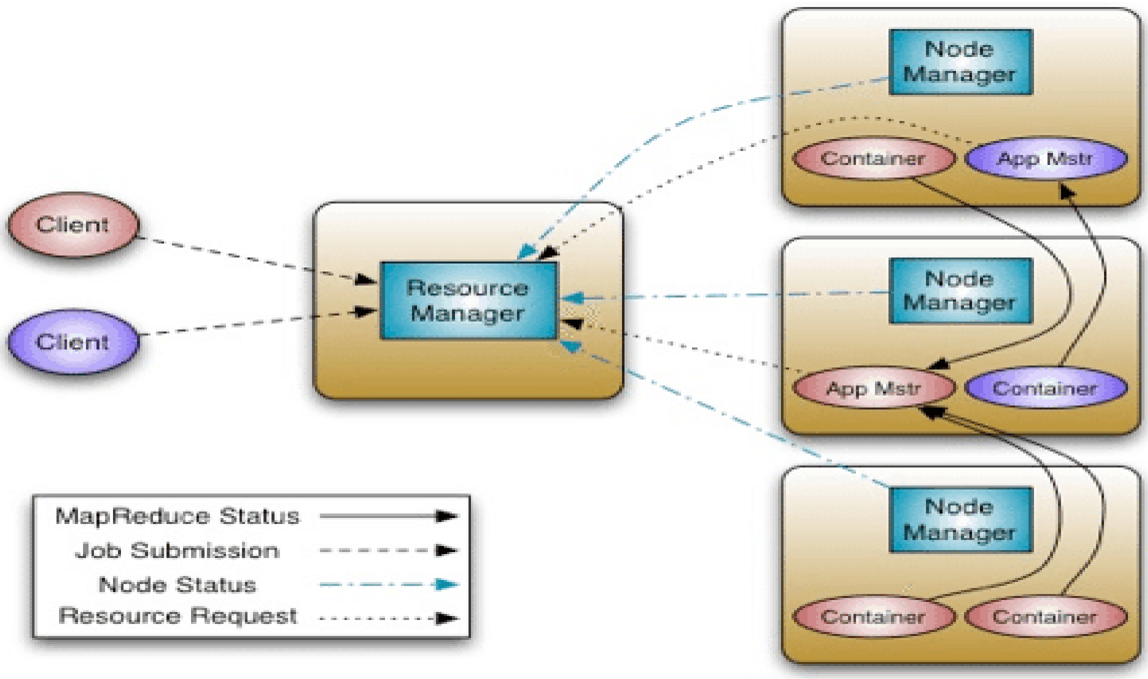
Secondary namenode and Standby namenode:

Secondary NameNode in hadoop is a specially dedicated node in HDFS cluster whose main function is to take checkpoints of the file system metadata present on namenode. It is not a backup namenode. It just checkpoints namenode's file system namespace. The Secondary NameNode is a helper to the primary NameNode but not replace for primary namenode.
As the NameNode is the single point of failure in HDFS, if NameNode fails entire HDFS file system is lost. So in order to overcome this, Hadoop implemented Secondary NameNode whose main function is to store a copy of FsImage file and edits log file.

FsImage is a snapshot of the HDFS file system metadata at a certain point of time and EditLog is a transaction log which contains records for every change that occurs to file system metadata. So, at any point of time, applying edits log records to FsImage (recently saved copy) will give the current status of FsImage, i.e. file system metadata.
Whenever a NameNode is restarted, the latest status of FsImage is built by applying edits records on last saved copy of FsImage. Since, NameNode merges FsImage and EditLog files only at start up, the edits log file might get very large over time on a busy cluster. That means, if the EditLog is very large, NameNode restart process result in some considerable delay in the availability of file system. So, it is important keep the edits log as small as possible which is one of the main functions of Secondary NameNode.

Secondary NameNode is not a true backup Namenode and cann't serve primary NameNode's operations.

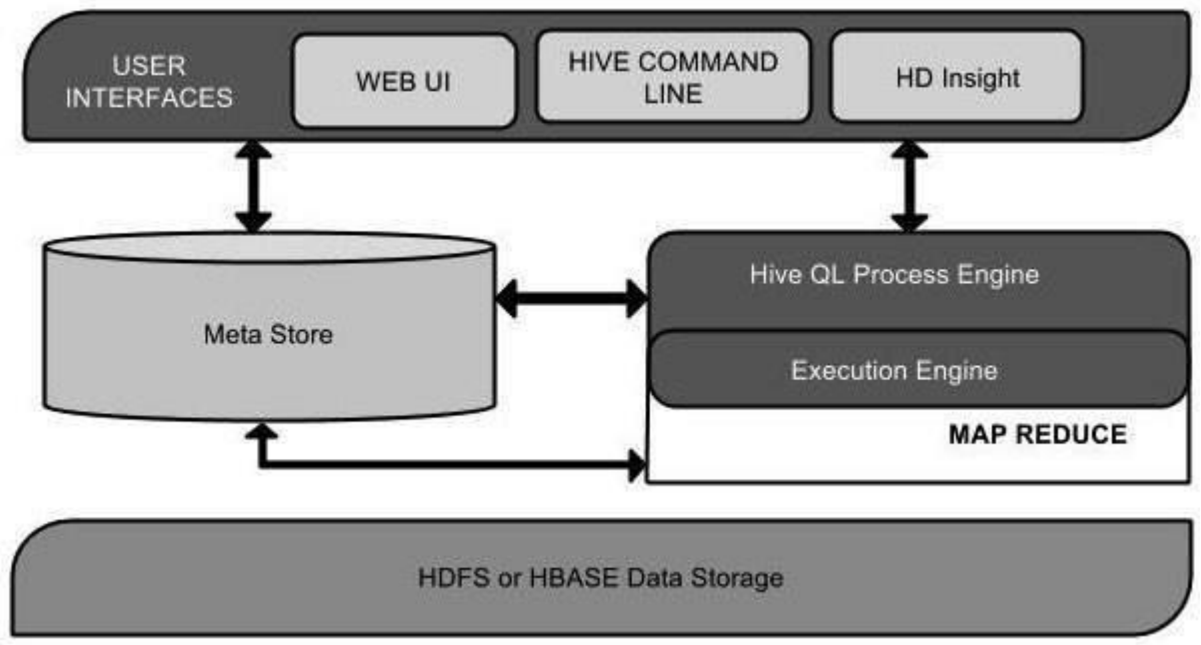
It usually runs on a different machine than the primary NameNode since its memory requirements are same as the primary NameNode.



YARN ARCHITECTURE

5) HIVE:

One of the biggest ingredients in the Information Platform built by Jeff 's team at Facebook was Apache Hive, a framework for data warehousing on top of Hadoop. Hive grew from a need to manage and learn from the huge volumes of data that Facebook was producing every day from its burgeoning social network. After trying a few different systems, the team chose Hadoop for storage and processing, since it was cost effective and met the scalability requirements. Hive was created to make it possible for analysts with strong SQL skills (but meagre Java programming skills) to run queries on the huge volumes of data that Facebook stored in HDFS. Today, Hive is a successful Apache project used by many organizations as a general-purpose, scalable data processing platform. Of course, SQL isn't ideal for every big data problem—it's not a good fit for building complex machine-learning algorithms, for example—but it's great for many analyses,
and it has the huge advantage of being very well known in the industry. What's more, SQL is the lingua franca in business intelligence tools (ODBC is a common bridge, for example), so Hive is well placed to integrate with these products.



HIVE ARCHITECTURE

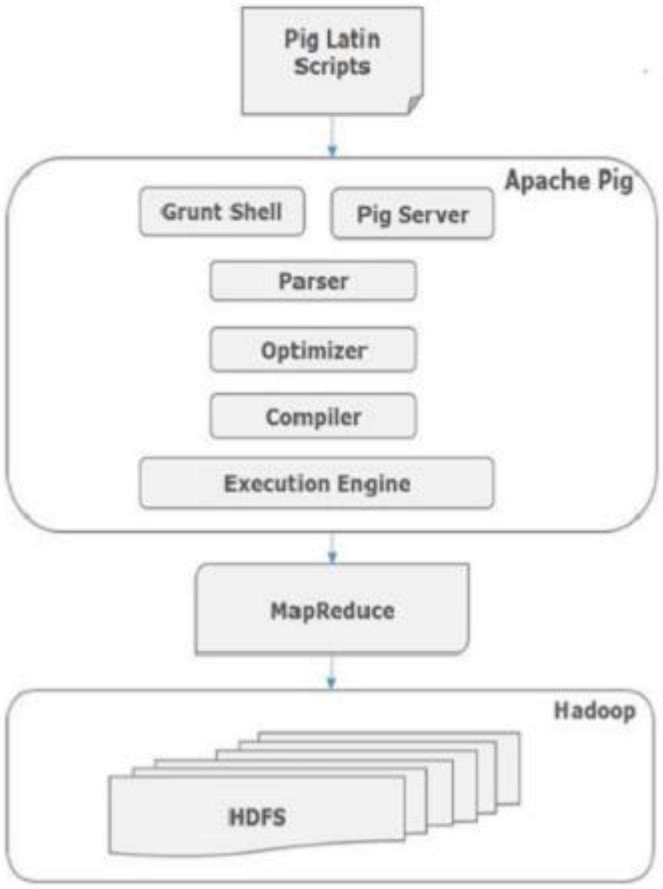
6)PIG:

Apache Pig raises the level of abstraction for processing large datasets. MapReduce allows you, as the programmer, to specify a map function followed by a reduce function, but working out how to fit your data processing into this pattern, which often requires multiple MapReduce stages, can be a challenge. With Pig, the data structures are much richer, typically being multivalued and nested, and the transformations you can apply to the data are much more powerful. They include joins, for example, which are not for the faint of heart in MapReduce. Pig is made up of two pieces:

- The language used to express data flows, called Pig Latin.
- The execution environment to run Pig Latin programs. There are currently two environments: local execution in a single JVM and distributed execution on a Hadoop cluster.

A Pig Latin program is made up of a series of operations, or transformations, that are applied to the input data to produce output. Taken as a whole, the operations describe a data flow, which the Pig execution environment translates into an executable representation and then runs. Under the covers, Pig turns the transformations into a series of MapReduce jobs, but as a programmer you are mostly unaware of this, which allows you to focus on the data rather than the nature of the execution.

Pig is a scripting language for exploring large datasets. One criticism of MapReduce is that the development cycle is very long. Writing the mappers and reducers, compiling and packaging the code, submitting the job(s), and retrieving the results is a time consuming business, and even with Streaming, which removes the compile and package step, the experience is still involved. Pig’s sweet spot is its ability to process terabytes of data in response to a half-dozen lines of Pig Latin issued from the console.

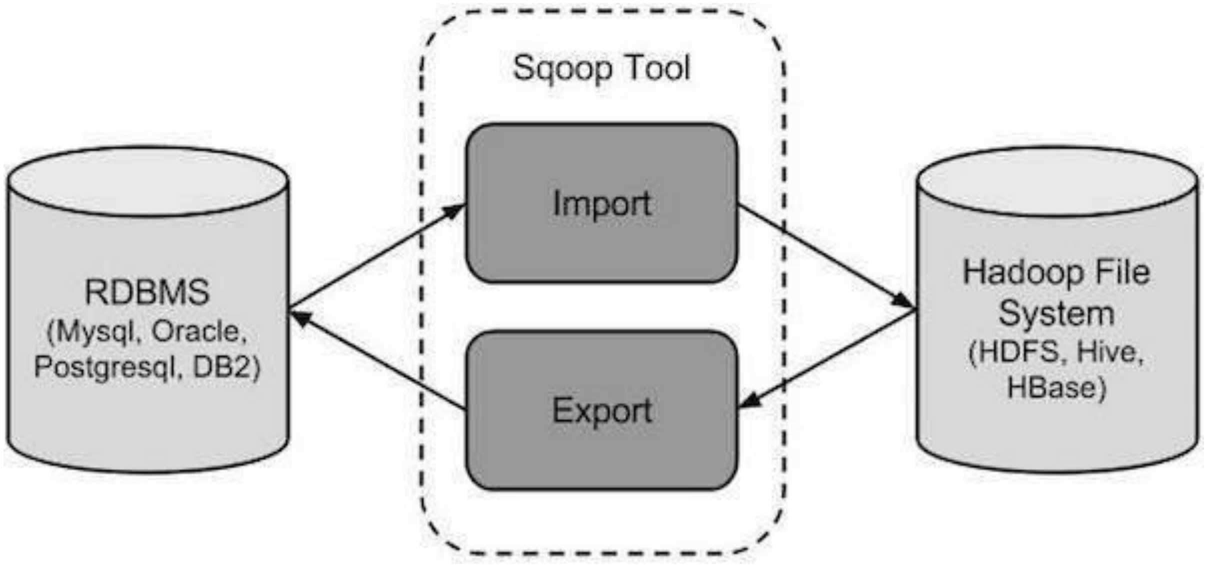


PIG ARCHITECTURE

7)SQOOP:

A great strength of the Hadoop platform is its ability to work with data in several different forms. HDFS can reliably store logs and other data from a plethora of sources, and MapReduce programs can parse diverse ad hoc data formats, extracting relevant information and combining multiple datasets into powerful results.

But to interact with data in storage repositories outside of HDFS, MapReduce programs need to use external APIs. Often, valuable data in an organization is stored in structured data stores such as relational database management systems (RDBMSs). Apache Sqoop is an open source tool that allows users to extract data from a structured data store into Hadoop for further processing. This processing can be done with MapReduce programs or other higherlevel tools such as Hive. (It’s even possible to use Sqoop to move data from a database into HBase.) When the final results of an analytic pipeline are available, Sqoop can export these results back to the data store for consumption by other clients.



8)H1B-CASE STUDY:

The H1B is an employment-based, non-immigrant visa category for temporary foreign workers in the United States. For a foreign national to apply for H1B visa, an US employer must offer a job and petition for H1B visa with the US immigration department. This is the most common visa status applied for and held by international students once they complete college/ higher education (Masters, Ph.D.) and work in a full-time position.

We will be performing analysis on the H1B visa applicants between the years 2011-2016. After analyzing the data, we can derive the following facts.

1 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.[ALL]

2 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.[certified]

3)Which industry(SOC_NAME) has the most number of Data Scientist positions?
[certified]

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

5) 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.

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

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

8) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order - [Certified and Certified Withdrawn.]

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

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

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

SUCCESS RATE % = (Certified + Certified Withdrawn)/Total x 100

The dataset has nearly 3 million records.

The dataset description is as follows:
The columns in the dataset include:

- 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

- EMPLOYER_NAME: Name of employer submitting labour condition application.
- 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.

- JOB_TITLE: Title of the job
- FULL_TIME_POSITION: Y = Full Time Position; N = Part Time Position
- 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.
- YEAR: Year in which the H1B visa petition was filed
- WORKSITE: City and State information of the foreign worker's intended area of employment
- lon: longitude of the Worksite
- lat: latitude of the Worksite

In the data, few columns are enclosed by double quotes and also we have comma's in a single column and the column is enclosed by double quotes. So we have used hive

csv serve to load the data. In the quoteChar, we have given **“(**double quote**).** So this will take the column value in between the double quotes.
Let’s create a table to load the h1b applicant’s data as shown below.

```
CREATE TABLE h1b_applications(s_no int,case_status string, employer_na
me string, soc_name string, job_title string, full_time_position string,prevailin
g_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;
```

<u>Use all the following tools</u>
HDFS
MapReduce - any 4 programs
Hive - any 4 programs
Pig - any 4 programs
Sqoop
Mysql
Excel
Project should Menu based using shell script in Unix

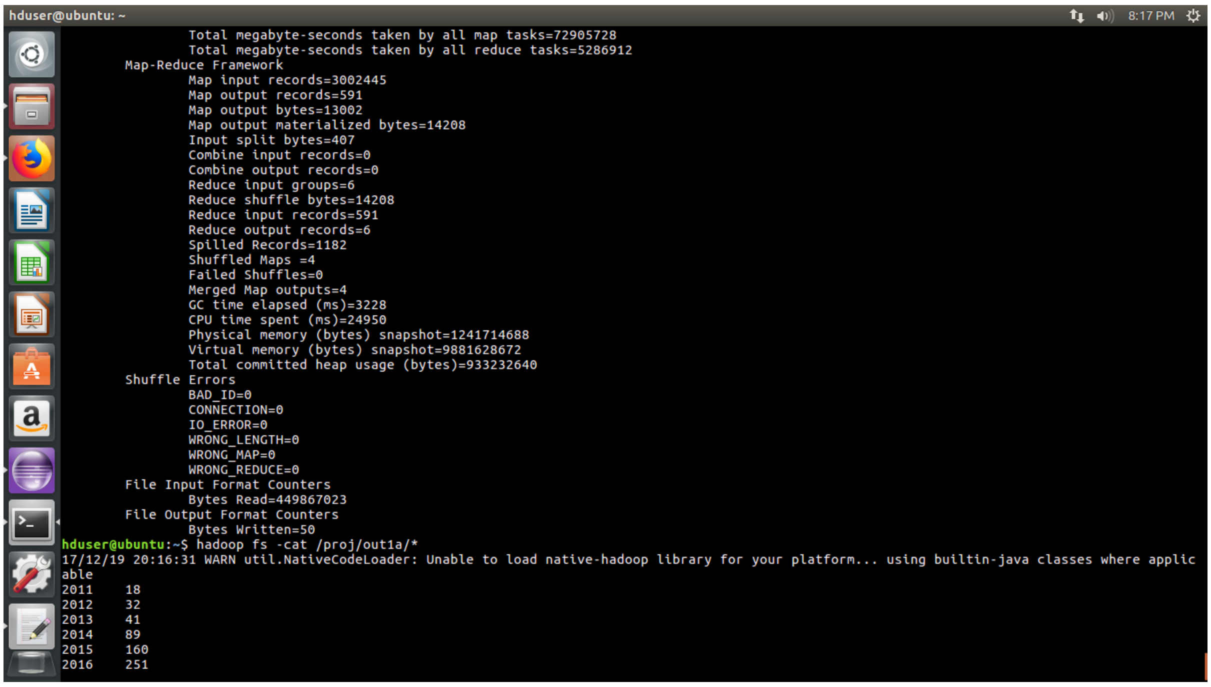
Case Status
1.certified
2.certified withdrawn
3. denied
4. withdrawn
5 pending quality and compliance review
6. invalidated
7 rejected
8.NA

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

```
import java.io.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
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 data {
public static class MapClass extends Mapper<LongWritable,Text,LongWritable,Text>
 {
                                        public void map(LongWritable key, Text value, Context context)
                                        {
                                          try{
                                                   String[] str = value.toString().split("\t");
                                                                   long year = Long.parseLong(str[7]);
                                                                   String jobname = str[4];
                                                          if(jobname.equals("DATA ENGINEER"))
                                                                   {

                                            context.write(new LongWritable(year),new Text(jobname));
                                          }}
                                          catch(Exception e)
                                          {
                                            System.out.println(e.getMessage());
                                          }
                                        }
                                      }

                                      public static class ReduceClass extends
Reducer<LongWritable,Text,LongWritable,LongWritable>
                                      {
                                           //private IntWritable result = new IntWritable(0);



                                           public void reduce(LongWritable key, Iterable<Text>
values,Context context) throws IOException, InterruptedException {
                                                  long count=0;
                                                          for (Text val : values)
                                              {

                                                                   count++;



                                                  }
                                                          context.write(key, new LongWritable(count));

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

```
                                    Configuration conf = new Configuration();
                                    Job job = Job.getInstance(conf, "Offence percentage");
                                    job.setJarByClass(data.class);
                                    job.setMapperClass(MapClass.class);
                                    //job.setCombinerClass(ReduceClass.class);
                                    job.setReducerClass(ReduceClass.class);
                                    //job.setNumReduceTasks(2);
                                    job.setMapOutputKeyClass(LongWritable.class);
                                    job.setMapOutputValueClass(Text.class);
                                    job.setOutputKeyClass(LongWritable.class);
                                    job.setOutputValueClass(LongWritable.class);
                                    FileInputFormat.addInputPath(job, new Path(args[0]));
                                    FileOutputFormat.setOutputPath(job, new Path(args[1]));
                                    System.exit(job.waitForCompletion(true) ? 0 : 1);
                                   }
}
```

OUTPUT SCREENSHOT:



```
hduser@ubuntu: ~                                                    8:17 PM
                Total megabyte-seconds taken by all map tasks=72905728
                Total megabyte-seconds taken by all reduce tasks=5286912
        Map-Reduce Framework
                Map input records=3002445
                Map output records=591
                Map output bytes=13002
                Map output materialized bytes=14208
                Input split bytes=407
                Combine input records=0
                Combine output records=0
                Reduce input groups=6
                Reduce shuffle bytes=14208
                Reduce input records=591
                Reduce output records=6
                Spilled Records=1182
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=3228
                CPU time spent (ms)=24950
                Physical memory (bytes) snapshot=1241714688
                Virtual memory (bytes) snapshot=9881628672
                Total committed heap usage (bytes)=933232640
        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=449867023
        File Output Format Counters
                Bytes Written=50
hduser@ubuntu:~$ hadoop fs -cat /proj/out1a/*
17/12/19 20:16:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
2011    18
2012    32
2013    41
2014    89
2015    160
2016    251
```

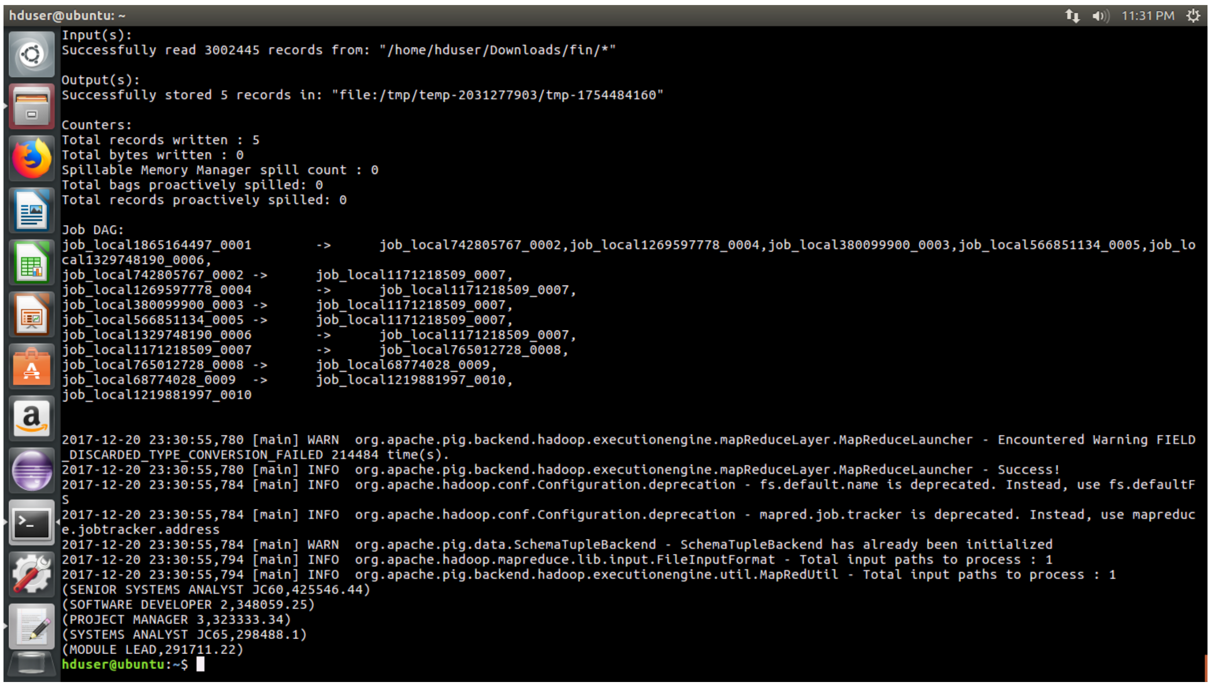
1 b) Find top 5 job titles who are having highest avg growth in applications.[ALL]

```
pro = load '/project/fin' using PigStorage('\t') AS (s_no:int,case_status:chararray,
employer_name:chararray, soc_name:chararray, job_title:chararray,
full_time_position:chararray,prevailing_wage:long,year:int, worksite:chararray,
longitute:long, latitute:long);
--dump pro;
```

```
kk = filter pro by year == 2011;
--dump kk;
ff = group kk by job_title;
--dump ff;
ss = foreach ff generate $0, (float)COUNT(kk.$1);
--dump ss;
kk12 = filter pro by year == 2012;
--dump kk12;
ff12 = group kk12 by job_title;
--dump ff12;
ss12 = foreach ff12 generate $0, (float)COUNT(kk12.$1);
--dump ss12;
jo = join ss by $0,ss12 by $0;
--dump jo;
gn = foreach jo generate $0, $1, $3, ((($3-$1)/$1)*100) as av;
--dump gn;
kk13 = filter pro by year == 2013;
--dump kk13;
ff13 = group kk13 by job_title;
--dump ff13;
ss13 = foreach ff13 generate $0, (float)COUNT(kk13.$1);
--dump ss13;
jo13 = join ss12 by $0,ss13 by $0;
--dump jo13;
gn13 = foreach jo13 generate $0, $1, $3, ((($3-$1)/$1)*100) as av1;
--dump gn13;
kk14 = filter pro by year == 2014;
--dump kk14;
ff14 = group kk14 by job_title;
--dump ff14;
ss14 = foreach ff14 generate $0, (float)COUNT(kk14.$1);
--dump ss14;
jo14 = join ss13 by $0,ss14 by $0;
--dump jo14;
gn14 = foreach jo14 generate $0, $1, $3, ((($3-$1)/$1)*100) as av2;
--dump gn14;
kk15 = filter pro by year == 2015;
--dump kk15;
ff15 = group kk15 by job_title;
--dump ff15;
ss15 = foreach ff15 generate $0, (float)COUNT(kk15.$1);
--dump ss15;
jo15 = join ss14 by $0,ss15 by $0;
```

```
--dump jo15;
gn15 = foreach jo15 generate $0, $1, $3, ((($3-$1)/$1)*100) as av3;
--dump gn15;
kk16 = filter pro by year == 2016;
--dump kk16;
ff16 = group kk16 by job_title;
--dump ff16;
ss16 = foreach ff16 generate $0, (float)COUNT(kk16.$1);
--dump ss16;
jo16 = join ss15 by $0,ss16 by $0;
--dump jo16;
gn16 = foreach jo16 generate $0, $1, $3, ((($3-$1)/$1)*100) as av4;
--dump gn16;
gg = join gn by $0,gn13 by $0,gn14 by $0,gn15 by $0,gn16 by $0;
--dump gg;
rr = foreach gg generate $0, ((av+av1+av2+av3+av4)/5) as cv;
--dump rr;
od = order rr by cv desc;
--dump od;
li = limit od 5;
dump li;

store li into '/home/hduser/Downloads/pro1band';
```

OUTPUT SCREENSHOT:



```
Input(s):
Successfully read 3002445 records from: "/home/hduser/Downloads/fin/*"

Output(s):
Successfully stored 5 records in: "file:/tmp/temp-2031277903/tmp-1754484160"

Counters:
Total records written : 5
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local1865164497_0001        ->      job_local742805767_0002,job_local1269597778_0004,job_local380099900_0003,job_local566851134_0005,job_lo
cal1329748190_0006,
job_local742805767_0002 ->      job_local1171218509_0007,
job_local1269597778_0004        ->      job_local1171218509_0007,
job_local380099900_0003 ->      job_local1171218509_0007,
job_local566851134_0005 ->      job_local1171218509_0007,
job_local1329748190_0006        ->      job_local1171218509_0007,
job_local1171218509_0007        ->      job_local765012728_0008,
job_local765012728_0008 ->      job_local68774028_0009,
job_local68774028_0009  ->      job_local1219881997_0010,
job_local1219881997_0010


2017-12-20 23:30:55,780 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FIELD
_DISCARDED_TYPE_CONVERSION_FAILED 214484 time(s).
2017-12-20 23:30:55,780 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-12-20 23:30:55,784 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultF
S
2017-12-20 23:30:55,784 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduc
e.jobtracker.address
2017-12-20 23:30:55,784 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-20 23:30:55,794 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-20 23:30:55,794 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(SENIOR SYSTEMS ANALYST JC60,425546.44)
(SOFTWARE DEVELOPER 2,348059.25)
(PROJECT MANAGER 3,323333.34)
(SYSTEMS ANALYST JC65,298488.1)
(MODULE LEAD,291711.22)
hduser@ubuntu:~$
```

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

select worksite,count(job_title) as tot from h1b_final where job_title = "DATA ENGINEER" and year = 2011 group by worksite order by tot desc limit 1;

select worksite,count(job_title) as tot from h1b_final where job_title = "DATA ENGINEER" and year = 2012 group by worksite order by tot desc limit 1;

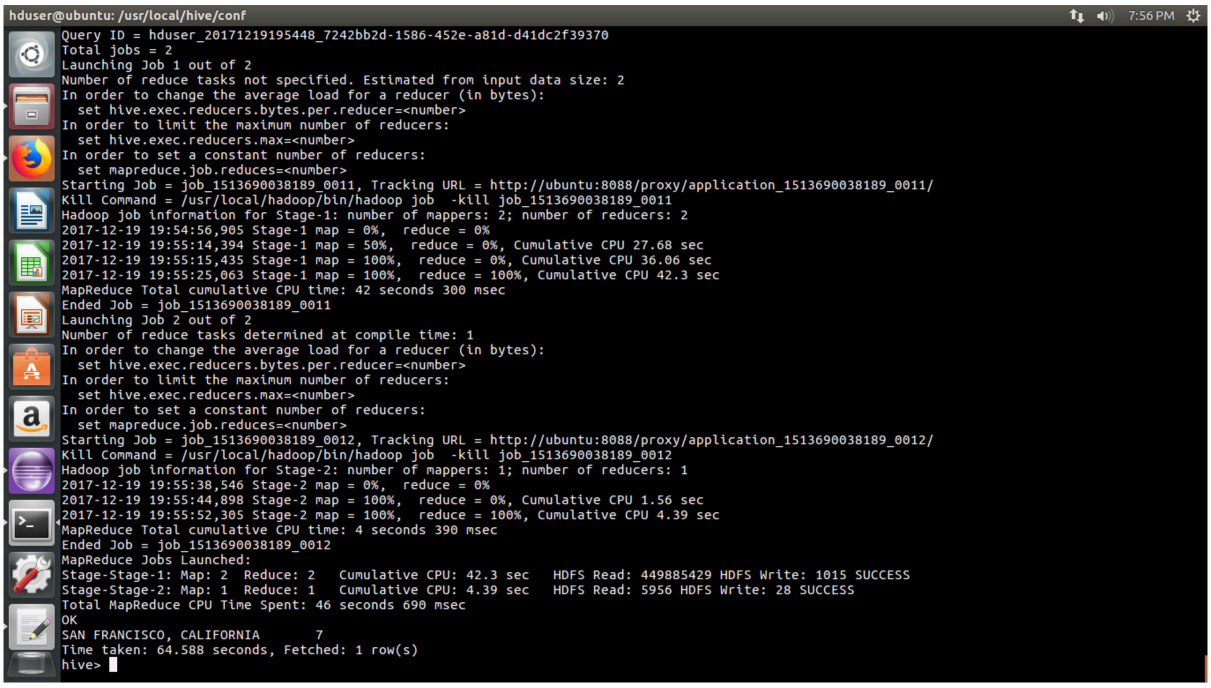
select worksite,count(job_title) as tot from h1b_final where job_title = "DATA ENGINEER" and year = 2013 group by worksite order by tot desc limit 1;

select worksite,count(job_title) as tot from h1b_final where job_title = "DATA ENGINEER" and year = 2014 group by worksite order by tot desc limit 1;

select worksite,count(job_title) as tot from h1b_final where job_title = "DATA ENGINEER" and year = 2015 group by worksite order by tot desc limit 1;

select worksite,count(job_title) as tot from h1b_final where job_title = "DATA ENGINEER" and year = 2016 group by worksite order by tot desc limit 1;

OUTPUT SCREENSHOT:

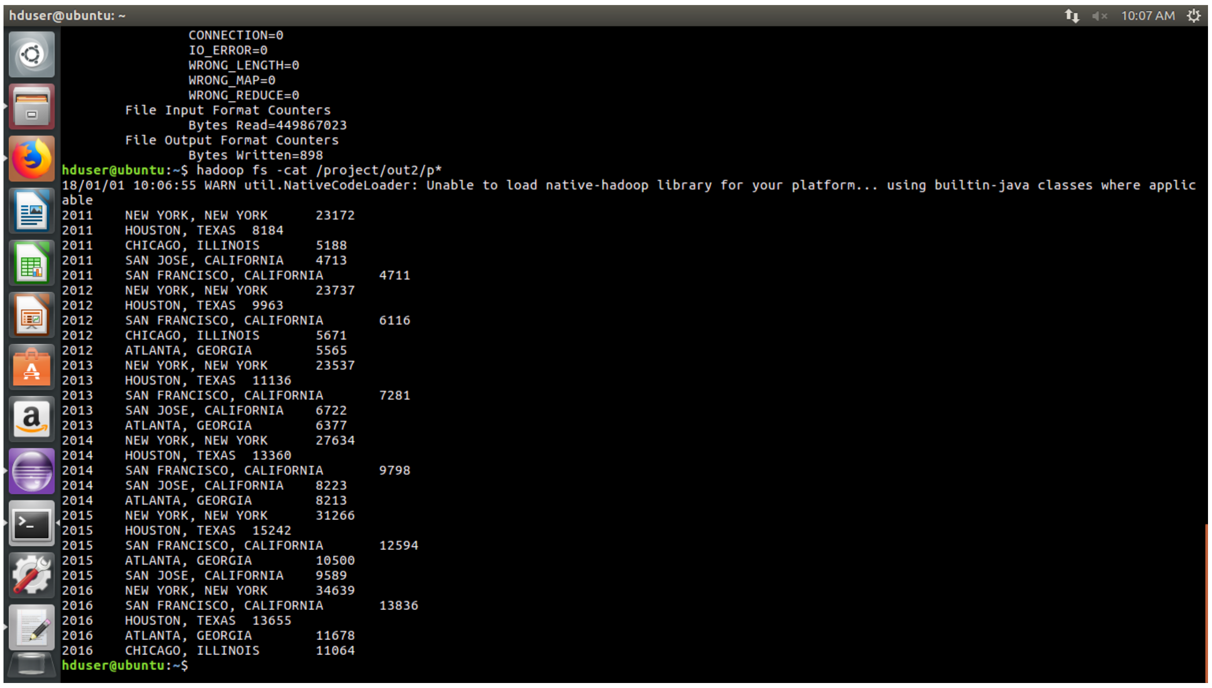


```
Query ID = hduser_20171219195448_7242bb2d-1586-452e-a81d-d41dc2f39370
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 2
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1513690038189_0011, Tracking URL = http://ubuntu:8088/proxy/application_1513690038189_0011/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1513690038189_0011
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 2
2017-12-19 19:54:56,905 Stage-1 map = 0%,  reduce = 0%
2017-12-19 19:55:14,394 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 27.68 sec
2017-12-19 19:55:15,435 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 36.06 sec
2017-12-19 19:55:25,063 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 42.3 sec
MapReduce Total cumulative CPU time: 42 seconds 300 msec
Ended Job = job_1513690038189_0011
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1513690038189_0012, Tracking URL = http://ubuntu:8088/proxy/application_1513690038189_0012/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1513690038189_0012
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2017-12-19 19:55:38,546 Stage-2 map = 0%,  reduce = 0%
2017-12-19 19:55:44,898 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.56 sec
2017-12-19 19:55:52,305 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 4.39 sec
MapReduce Total cumulative CPU time: 4 seconds 390 msec
Ended Job = job_1513690038189_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 2   Cumulative CPU: 42.3 sec   HDFS Read: 449885429 HDFS Write: 1015 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.39 sec   HDFS Read: 5956 HDFS Write: 28 SUCCESS
Total MapReduce CPU Time Spent: 46 seconds 690 msec
OK
SAN FRANCISCO, CALIFORNIA       7
Time taken: 64.588 seconds, Fetched: 1 row(s)
hive> 
```

2b) find top 5 locations in the US who have got certified visa for each year.[certified]

```
import java.io.IOException;
import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class loc
{
	public static class MapClass extends Mapper<LongWritable, Text, Text, Text>
	{
		public void map(LongWritable key, Text values, Context con) throws IOException, 
InterruptedException
		{
			String[] str = values.toString().split("\t");
			con.write(new Text(str[8]), values);
		}
	}
	public static class YearPartitioner extends Partitioner<Text, Text>
	{
		public int getPartition(Text key, Text values, int numReduceTasks)
		{
			String[] str = values.toString().split("\t");
			if(str[7].equals("2011"))
			{
				return 0;
			}
			else if(str[7].equals("2012"))
			{
				return 1;
			}
			else if(str[7].equals("2013"))
			{
				return 2;
			}
			else if(str[7].equals("2014"))
			{
				return 3;
			}
			else if(str[7].equals("2015"))
			{
				return 4;
			}
			else
			{
				return 5;
				}
			}
		}
		public static class ReduceClass extends Reducer<Text, Text, NullWritable, Text>
		{
			public TreeMap<Long, Text> tm = new TreeMap<Long, Text>();
			public void reduce(Text key, Iterable<Text> values, Context con) throws IOException,
InterruptedException
			{
				long count=0;
				String tot="";
				for(Text val:values)
				{
					String[] str = val.toString().split("\t");
					if(str[1].equals("CERTIFIED"))
					{
						count++;
						tot = str[7]+"\t"+key;
					}

				}
				String totval = tot+"\t"+count;
				tm.put(new Long(count), new Text(totval));
				if(tm.size()>5)
				{
					tm.remove(tm.firstKey());
				}
			}
			public void cleanup(Context con) throws IOException, InterruptedException
			{
				for(Text t:tm.descendingMap().values())
				{
					con.write(NullWritable.get(), t);
				}
			}
		}
		public static void main(String[] args) throws Exception
		{
			Configuration conf = new Configuration();
			Job job = Job.getInstance(conf,"Most Data Scientist");
			job.setJarByClass(loc.class);
			job.setMapperClass(MapClass.class);
			job.setPartitionerClass(YearPartitioner.class);
			job.setReducerClass(ReduceClass.class);
			job.setNumReduceTasks(6);
			job.setMapOutputKeyClass(Text.class);
			job.setMapOutputValueClass(Text.class);
			job.setOutputKeyClass(NullWritable.class);
			job.setOutputValueClass(Text.class);
			FileInputFormat.addInputPath(job, new Path(args[0]));
			FileOutputFormat.setOutputPath(job, new Path(args[1]));
			System.exit(job.waitForCompletion(true) ? 0 : 1);
		}
}
```

OUTPUT SCREENSHOT:

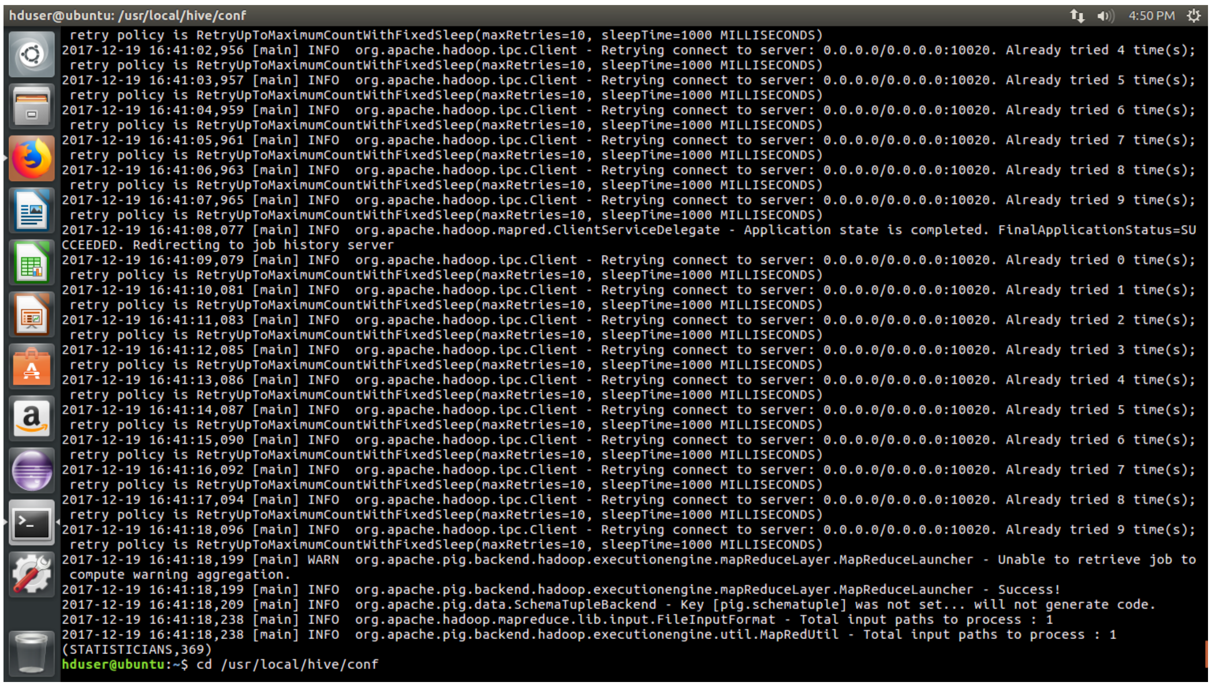


```
hduser@ubuntu: ~
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=449867023
        File Output Format Counters
                Bytes Written=898
hduser@ubuntu:~$ hadoop fs -cat /project/out2/p*
18/01/01 10:06:55 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
2011    NEW YORK, NEW YORK      23172
2011    HOUSTON, TEXAS  8184
2011    CHICAGO, ILLINOIS       5188
2011    SAN JOSE, CALIFORNIA    4713
2011    SAN FRANCISCO, CALIFORNIA       4711
2012    NEW YORK, NEW YORK      23737
2012    HOUSTON, TEXAS  9963
2012    SAN FRANCISCO, CALIFORNIA       6116
2012    CHICAGO, ILLINOIS       5671
2012    ATLANTA, GEORGIA        5565
2013    NEW YORK, NEW YORK      23537
2013    HOUSTON, TEXAS  11136
2013    SAN FRANCISCO, CALIFORNIA       7281
2013    SAN JOSE, CALIFORNIA    6722
2013    ATLANTA, GEORGIA        6377
2014    NEW YORK, NEW YORK      27634
2014    HOUSTON, TEXAS  13360
2014    SAN FRANCISCO, CALIFORNIA       9798
2014    SAN JOSE, CALIFORNIA    8223
2014    ATLANTA, GEORGIA        8213
2015    NEW YORK, NEW YORK      31266
2015    HOUSTON, TEXAS  15242
2015    SAN FRANCISCO, CALIFORNIA       12594
2015    ATLANTA, GEORGIA        10500
2015    SAN JOSE, CALIFORNIA    9589
2016    NEW YORK, NEW YORK      34639
2016    SAN FRANCISCO, CALIFORNIA       13836
2016    HOUSTON, TEXAS  13655
2016    ATLANTA, GEORGIA        11678
2016    CHICAGO, ILLINOIS       11064
hduser@ubuntu:~$
```

3)Which industry(SOC_NAME) has the most number of Data Scientist positions? [certified]

```
pro = load '/project/fin' using PigStorage('\t') AS (s_no:int,case_status:chararray, employer_name:chararray, soc_name:chararray, job_title:chararray, full_time_position:chararray,prevailing_wage:long,year:int, worksite:chararray, longitute:long, latitute:long);
--dump pro;
soc = filter pro by job_title == 'DATA SCIENTIST' and case_status == 'CERTIFIED';
--dump soc;
grp = group soc by $3;
--dump grp;
cou = foreach grp generate $0, COUNT(soc.job_title) as tot;
--dump cou;
od = order cou by tot desc;
--dump od;
li = limit od 1;
dump li;
store li into '/home/hduser/Downloads/pro3ans';
```

OUTPUT SCREENSHOT:



```
hduser@ubuntu: /usr/local/hive/conf
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:02,956 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 4 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:03,957 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 5 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:04,959 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 6 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:05,961 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 7 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:06,963 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 8 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:07,965 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 9 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:08,077 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SU
CCEEDED. Redirecting to job history server
2017-12-19 16:41:09,079 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 0 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:10,081 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 1 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:11,083 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 2 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:12,085 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 3 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:13,086 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 4 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:14,087 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 5 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:15,090 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 6 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:16,092 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 7 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:17,094 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 8 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:18,096 [main] INFO  org.apache.hadoop.ipc.Client - Retrying connect to server: 0.0.0.0/0.0.0.0:10020. Already tried 9 time(s);
 retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2017-12-19 16:41:18,199 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Unable to retrieve job to
 compute warning aggregation.
2017-12-19 16:41:18,199 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-12-19 16:41:18,209 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2017-12-19 16:41:18,238 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-19 16:41:18,238 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(STATISTICIANS,369)
hduser@ubuntu:~$ cd /usr/local/hive/conf
```

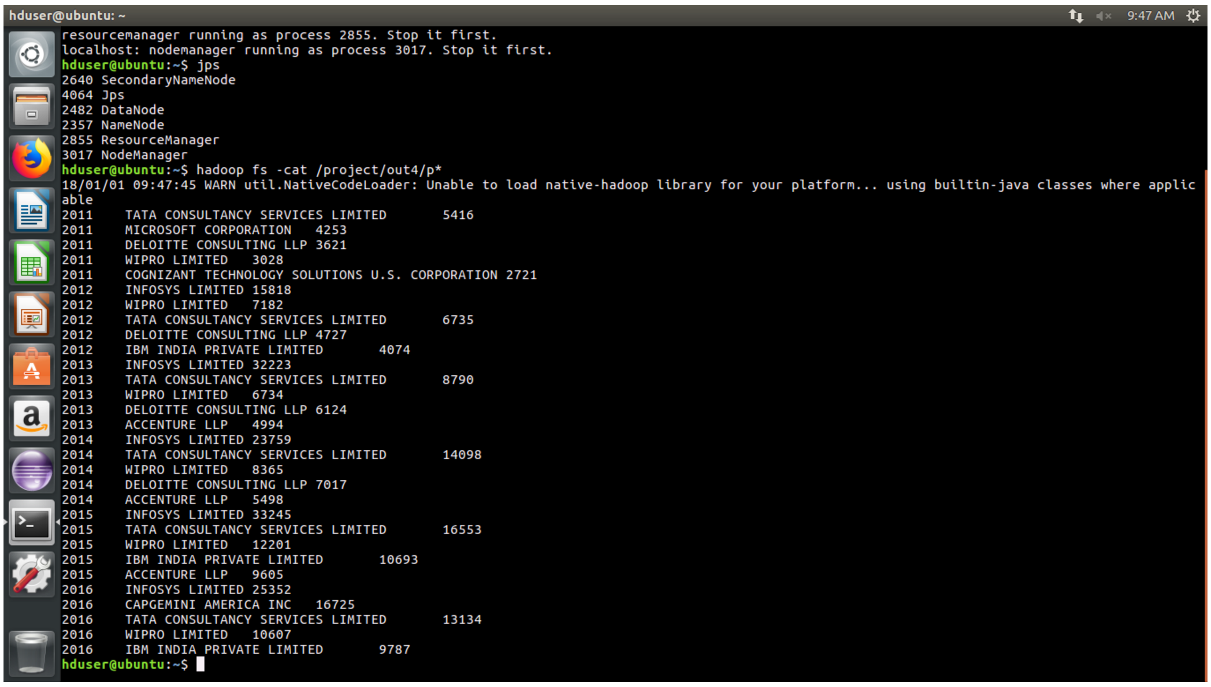
4)Which top 5 employers file the most petitions each year? - Case Status - ALL

```
  import java.io.IOException;
import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class prtition4
{
public static class MapClass extends Mapper<LongWritable, Text, Text, Text>
{
        public void map(LongWritable key, Text values, Context context) throws IOException,
InterruptedException
        {
                String[] str = values.toString().split("\t");
                String epname=str[2];
                context.write(new Text(epname), new Text(values));
        }
}
public static class Year extends Partitioner<Text, Text>
{

        public int getPartition(Text key, Text values, int numReduceTasks)
        {
                String[] str = values.toString().split("\t");
                long year = Long.parseLong(str[7]);
                if(year==2011)
                {
                        return 0;
                }
                else if(year==2012)
                {
                        return 1;
                }
                else if(year==2013)
                {
                        return 2;
                }
                else if(year==2014)
                {
                        return 3;
                }
                else if(year==2015)
                {
                        return 4;
                }
                else
                {
                        return 5;
                }
        }
}
public static class ReduceClass extends Reducer<Text,Text,NullWritable,Text>
{
        private TreeMap<Long, Text> tm = new TreeMap<Long, Text>();
        public void reduce(Text key, Iterable<Text> values, Context con) throws IOException,
InterruptedException
        {
                long count=0;
                String year="";
                for(Text val:values)
                {
```

```
                String[] str = val.toString().split("\t");
                year = str[7];
                count++;
            }
            String myValue = year+"\t"+key+"\t"+count;
            tm.put(new Long(count), new Text(myValue));
            if(tm.size()>5)
            {
                tm.remove(tm.firstKey());
            }
        }
        protected void cleanup(Context context) throws IOException, InterruptedException
        {
            for(Text t:tm.descendingMap().values())
            {
                context.write(NullWritable.get(), t);
            }
        }
}
public static void main(String[] args) throws Exception
{
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, " Top petition");
        job.setJarByClass(prtition4.class);
        job.setMapperClass(MapClass.class);
        job.setPartitionerClass(Year.class);
        job.setReducerClass(ReduceClass.class);
        job.setNumReduceTasks(6);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(NullWritable.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

OUTPUT SCREENSHOT:



```
resourcemanager running as process 2855. Stop it first.
localhost: nodemanager running as process 3017. Stop it first.
hduser@ubuntu:~$ jps
2640 SecondaryNameNode
4064 Jps
2482 DataNode
2357 NameNode
2855 ResourceManager
3017 NodeManager
hduser@ubuntu:~$ hadoop fs -cat /project/out4/p*
18/01/01 09:47:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
2011    TATA CONSULTANCY SERVICES LIMITED       5416
2011    MICROSOFT CORPORATION   4253
2011    DELOITTE CONSULTING LLP 3621
2011    WIPRO LIMITED   3028
2011    COGNIZANT TECHNOLOGY SOLUTIONS U.S. CORPORATION 2721
2012    INFOSYS LIMITED 15818
2012    WIPRO LIMITED   7182
2012    TATA CONSULTANCY SERVICES LIMITED       6735
2012    DELOITTE CONSULTING LLP 4727
2012    IBM INDIA PRIVATE LIMITED       4074
2013    INFOSYS LIMITED 32223
2013    TATA CONSULTANCY SERVICES LIMITED       8790
2013    WIPRO LIMITED   6734
2013    DELOITTE CONSULTING LLP 6124
2013    ACCENTURE LLP   4994
2014    INFOSYS LIMITED 23759
2014    TATA CONSULTANCY SERVICES LIMITED       14098
2014    WIPRO LIMITED   8365
2014    DELOITTE CONSULTING LLP 7017
2014    ACCENTURE LLP   5498
2015    INFOSYS LIMITED 33245
2015    TATA CONSULTANCY SERVICES LIMITED       16553
2015    WIPRO LIMITED   12201
2015    IBM INDIA PRIVATE LIMITED       10693
2015    ACCENTURE LLP   9605
2016    INFOSYS LIMITED 25352
2016    CAPGEMINI AMERICA INC   16725
2016    TATA CONSULTANCY SERVICES LIMITED       13134
2016    WIPRO LIMITED   10607
2016    IBM INDIA PRIVATE LIMITED       9787
hduser@ubuntu:~$
```

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

select job_title,count(job_title) as tot from h1b_final where year = 2011 group by job_title order by tot desc limit 10;

select job_title,count(job_title) as tot from h1b_final where year = 2012 group by job_title order by tot desc limit 10;

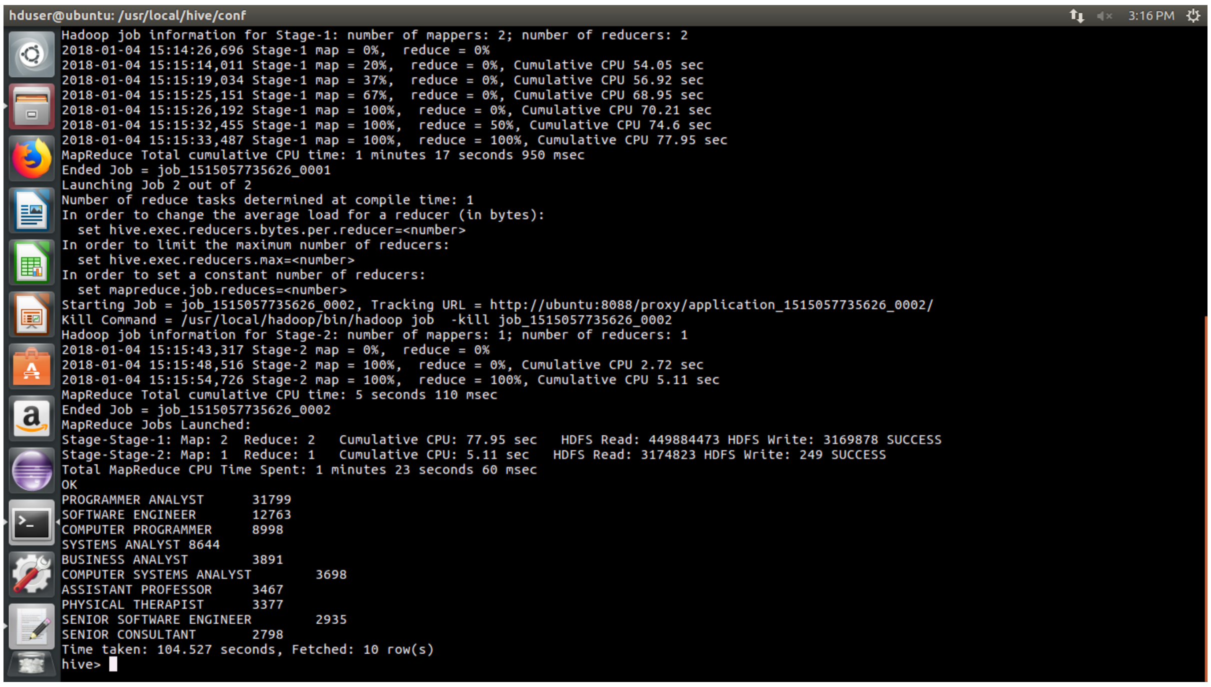
select job_title,count(job_title) as tot from h1b_final where year = 2013 group by job_title order by tot desc limit 10;

select job_title,count(job_title) as tot from h1b_final where year = 2014 group by job_title order by tot desc limit 10;

select job_title,count(job_title) as tot from h1b_final where year = 2015 group by job_title order by tot desc limit 10;

select job_title,count(job_title) as tot from h1b_final where year = 2016 group by job_title order by tot desc limit 10;

OUTPUT SCREENSHOT:



```
hduser@ubuntu: /usr/local/hive/conf
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 2
2018-01-04 15:14:26,696 Stage-1 map = 0%,  reduce = 0%
2018-01-04 15:15:14,011 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 54.05 sec
2018-01-04 15:15:19,034 Stage-1 map = 37%,  reduce = 0%, Cumulative CPU 56.92 sec
2018-01-04 15:15:25,151 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 68.95 sec
2018-01-04 15:15:26,192 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 70.21 sec
2018-01-04 15:15:32,455 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 74.6 sec
2018-01-04 15:15:33,487 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 77.95 sec
MapReduce Total cumulative CPU time: 1 minutes 17 seconds 950 msec
Ended Job = job_1515057735626_0001
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1515057735626_0002, Tracking URL = http://ubuntu:8088/proxy/application_1515057735626_0002/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1515057735626_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-01-04 15:15:43,317 Stage-2 map = 0%,  reduce = 0%
2018-01-04 15:15:48,516 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.72 sec
2018-01-04 15:15:54,726 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 5.11 sec
MapReduce Total cumulative CPU time: 5 seconds 110 msec
Ended Job = job_1515057735626_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 2   Cumulative CPU: 77.95 sec   HDFS Read: 449884473 HDFS Write: 3169878 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.11 sec   HDFS Read: 3174823 HDFS Write: 249 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 23 seconds 60 msec
OK
PROGRAMMER ANALYST      31799
SOFTWARE ENGINEER       12763
COMPUTER PROGRAMMER     8998
SYSTEMS ANALYST 8644
BUSINESS ANALYST        3891
COMPUTER SYSTEMS ANALYST        3698
ASSISTANT PROFESSOR     3467
PHYSICAL THERAPIST      3377
SENIOR SOFTWARE ENGINEER        2935
SENIOR CONSULTANT       2798
Time taken: 104.527 seconds, Fetched: 10 row(s)
hive>
```

5b) for only certified applications.

select job_title,count(job_title) as tot from h1b_final where year = 2011 and case_status = 'CERTIFIED' group by job_title order by tot desc limit 10;

select job_title,count(job_title) as tot from h1b_final where year = 2012 and case_status = 'CERTIFIED' group by job_title order by tot desc limit 10;

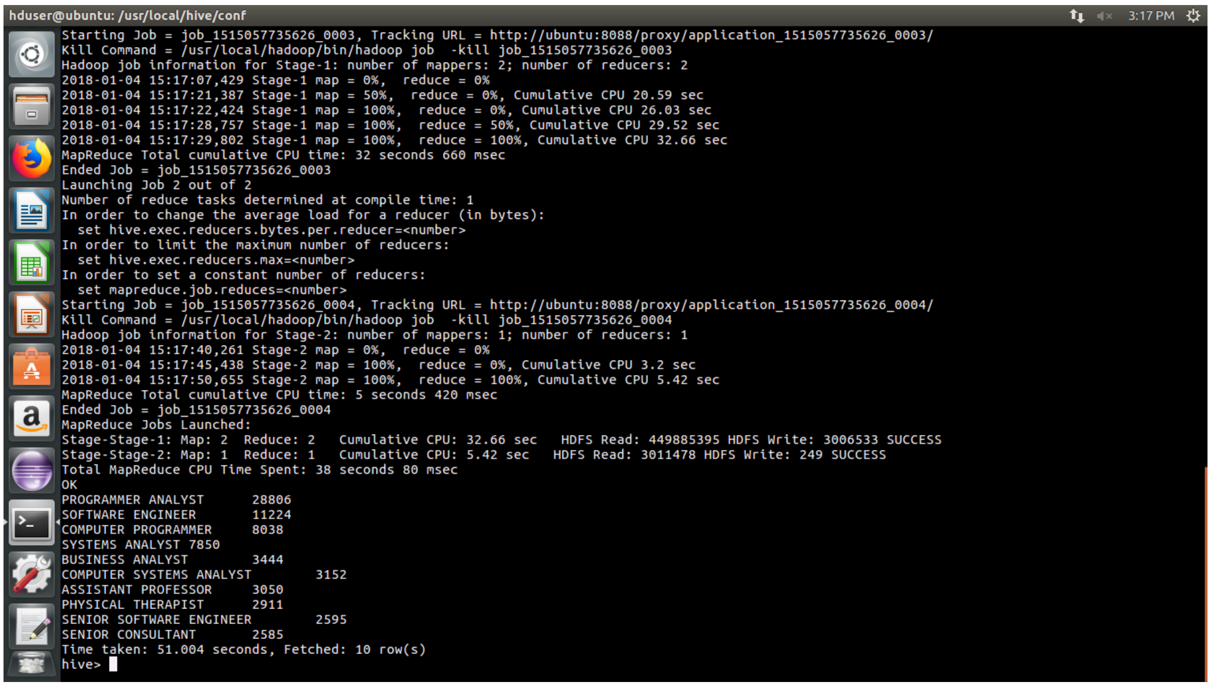
select job_title,count(job_title) as tot from h1b_final where year = 2013 and case_status = 'CERTIFIED' group by job_title order by tot desc limit 10;

select job_title,count(job_title) as tot from h1b_final where year = 2014 and case_status = 'CERTIFIED' group by job_title order by tot desc limit 10;

select job_title,count(job_title) as tot from h1b_final where year = 2015 and case_status = 'CERTIFIED' group by job_title order by tot desc limit 10;

select job_title,count(job_title) as tot from h1b_final where year = 2016 and case_status = 'CERTIFIED' group by job_title order by tot desc limit 10;

OUTPUT SCREENSHOT:



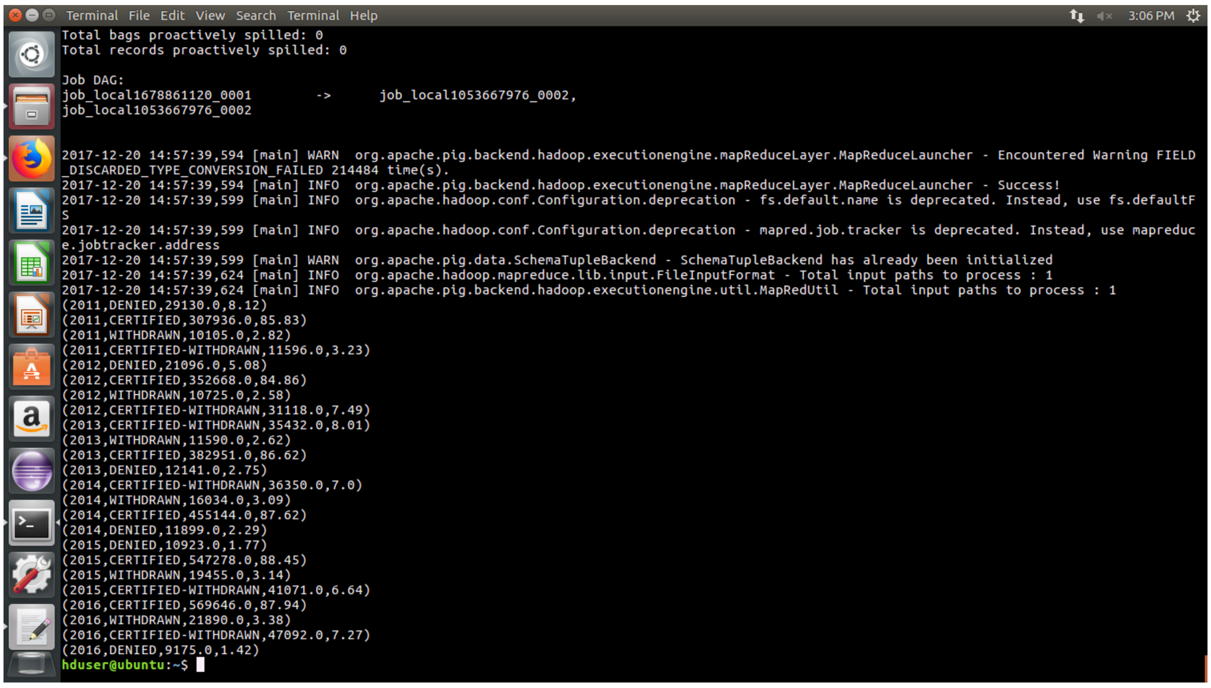
```
hduser@ubuntu: /usr/local/hive/conf                                     3:17 PM
Starting Job = job_1515057735626_0003, Tracking URL = http://ubuntu:8088/proxy/application_1515057735626_0003/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1515057735626_0003
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 2
2018-01-04 15:17:07,429 Stage-1 map = 0%,  reduce = 0%
2018-01-04 15:17:21,387 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 20.59 sec
2018-01-04 15:17:22,424 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 26.03 sec
2018-01-04 15:17:28,757 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 29.52 sec
2018-01-04 15:17:29,802 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 32.66 sec
MapReduce Total cumulative CPU time: 32 seconds 660 msec
Ended Job = job_1515057735626_0003
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1515057735626_0004, Tracking URL = http://ubuntu:8088/proxy/application_1515057735626_0004/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1515057735626_0004
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-01-04 15:17:40,261 Stage-2 map = 0%,  reduce = 0%
2018-01-04 15:17:45,438 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 3.2 sec
2018-01-04 15:17:50,655 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 5.42 sec
MapReduce Total cumulative CPU time: 5 seconds 420 msec
Ended Job = job_1515057735626_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 2   Cumulative CPU: 32.66 sec   HDFS Read: 449885395 HDFS Write: 3006533 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.42 sec   HDFS Read: 3011478 HDFS Write: 249 SUCCESS
Total MapReduce CPU Time Spent: 38 seconds 80 msec
OK
PROGRAMMER ANALYST      28806
SOFTWARE ENGINEER       11224
COMPUTER PROGRAMMER     8038
SYSTEMS ANALYST 7850
BUSINESS ANALYST        3444
COMPUTER SYSTEMS ANALYST        3152
ASSISTANT PROFESSOR     3050
PHYSICAL THERAPIST      2911
SENIOR SOFTWARE ENGINEER        2595
SENIOR CONSULTANT       2585
Time taken: 51.004 seconds, Fetched: 10 row(s)
hive> 
```

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

```
pro = load '/project/fin' using PigStorage('\t') AS (s_no:int,case_status:chararray, employer_name:chararray, soc_name:chararray, job_title:chararray, full_time_position:chararray,prevailing_wage:long,year:int, worksite:chararray, longitute:long, latitute:long);
--dump pro;
grp1 = group pro by ($1,$7);
--dump grp1;
fla = foreach grp1 generate flatten(group), (float)COUNT(pro.case_status);
--dump fla;
kk = group pro by year;
--dump kk;
fd = foreach kk generate $0, (float)COUNT(pro.$1);
--dump fd;
de = join fla by $1, fd by $0;
--dump de;
```

```
fr = foreach de generate $0,$1,$2,$4;
--dump fr;
fi = foreach fr generate $0,$1,$2, ROUND_TO((($2/$3)*100),2);
--dump fi;
om = order fi by $0 asc,$1 asc;
dump om;
store om into '/home/hduser/Downloads/6ans';
```

OUTPUT SCREENSHOT:

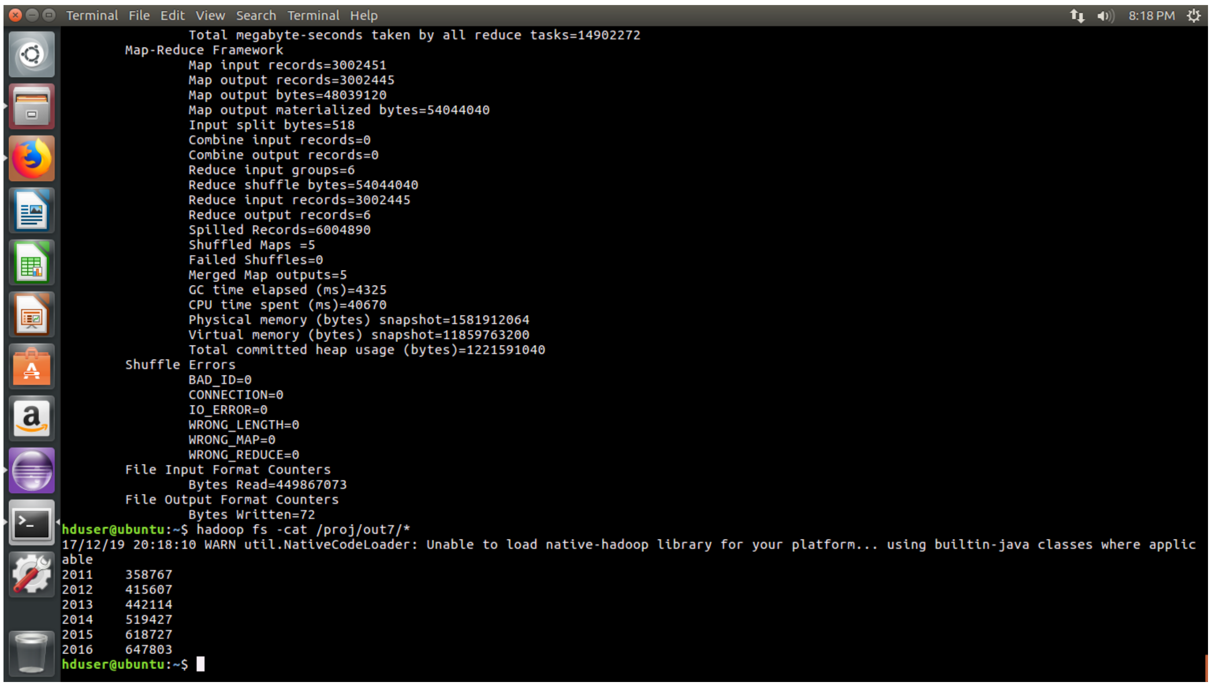


```
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local1678861120_0001        ->      job_local1053667976_0002,
job_local1053667976_0002


2017-12-20 14:57:39,594 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FIELD
_DISCARDED_TYPE_CONVERSION_FAILED 214484 time(s).
2017-12-20 14:57:39,594 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2017-12-20 14:57:39,599 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultF
S
2017-12-20 14:57:39,599 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduc
e.jobtracker.address
2017-12-20 14:57:39,599 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-20 14:57:39,624 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-20 14:57:39,624 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(2011,DENIED,29130.0,8.12)
(2011,CERTIFIED,307936.0,85.83)
(2011,WITHDRAWN,10105.0,2.82)
(2011,CERTIFIED-WITHDRAWN,11596.0,3.23)
(2012,DENIED,21096.0,5.08)
(2012,CERTIFIED,352668.0,84.86)
(2012,WITHDRAWN,10725.0,2.58)
(2012,CERTIFIED-WITHDRAWN,31118.0,7.49)
(2013,CERTIFIED-WITHDRAWN,35432.0,8.01)
(2013,WITHDRAWN,11590.0,2.62)
(2013,CERTIFIED,382951.0,86.62)
(2013,DENIED,12141.0,2.75)
(2014,CERTIFIED-WITHDRAWN,36350.0,7.0)
(2014,WITHDRAWN,16034.0,3.09)
(2014,CERTIFIED,455144.0,87.62)
(2014,DENIED,11899.0,2.29)
(2015,DENIED,10923.0,1.77)
(2015,CERTIFIED,547278.0,88.45)
(2015,WITHDRAWN,19455.0,3.14)
(2015,CERTIFIED-WITHDRAWN,41071.0,6.64)
(2016,CERTIFIED,569646.0,87.94)
(2016,WITHDRAWN,21890.0,3.38)
(2016,CERTIFIED-WITHDRAWN,47092.0,7.27)
(2016,DENIED,9175.0,1.42)
hduser@ubuntu:~$
```

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

```
import java.io.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
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 coun {

public static class MapClass extends Mapper<LongWritable,Text,LongWritable,LongWritable>
   {
      public void map(LongWritable key, Text value, Context context)
      {
        try{
              String[] str = value.toString().split("\t");
                                                  long year = Long.parseLong(str[7]);
                                                  long sno = Long.parseLong(str[0]);

            context.write(new LongWritable(year),new LongWritable(sno));
        }
        catch(Exception e)
        {
          System.out.println(e.getMessage());
        }
      }
   }

  public static class ReduceClass extends Reducer<LongWritable,LongWritable,LongWritable,LongWritable>
   {
                //private IntWritable result = new IntWritable(0);



                public void reduce(LongWritable key, Iterable<LongWritable> values,Context context) throws
IOException, InterruptedException {
                          long count=0;
                                      for (LongWritable val : values)
                      {

                                                  count++;

                          }

                  context.write(key, new LongWritable(count));
                }
   }
  public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                Job job = Job.getInstance(conf, "Offence percentage");
                job.setJarByClass(coun.class);
                job.setMapperClass(MapClass.class);
                //job.setCombinerClass(ReduceClass.class);
                job.setReducerClass(ReduceClass.class);
                //job.setNumReduceTasks(2);
                job.setMapOutputKeyClass(LongWritable.class);
                job.setMapOutputValueClass(LongWritable.class);
                job.setOutputKeyClass(LongWritable.class);
                job.setOutputValueClass(LongWritable.class);
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                System.exit(job.waitForCompletion(true) ? 0 : 1);
              }
}
```

OUTPUT SCREENSHOT:



```
                Total megabyte-seconds taken by all reduce tasks=14902272
        Map-Reduce Framework
                Map input records=3002451
                Map output records=3002445
                Map output bytes=48039120
                Map output materialized bytes=54044040
                Input split bytes=518
                Combine input records=0
                Combine output records=0
                Reduce input groups=6
                Reduce shuffle bytes=54044040
                Reduce input records=3002445
                Reduce output records=6
                Spilled Records=6004890
                Shuffled Maps =5
                Failed Shuffles=0
                Merged Map outputs=5
                GC time elapsed (ms)=4325
                CPU time spent (ms)=40670
                Physical memory (bytes) snapshot=1581912064
                Virtual memory (bytes) snapshot=11859763200
                Total committed heap usage (bytes)=1221591040
        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=449867073
        File Output Format Counters
                Bytes Written=72
hduser@ubuntu:~$ hadoop fs -cat /proj/out7/*
17/12/19 20:18:10 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
2011    358767
2012    415607
2013    442114
2014    519427
2015    618727
2016    647803
hduser@ubuntu:~$ 
```

8) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order - [Certified and Certified Withdrawn.]

FOR FULL TIME JOB:

pro = load '/project/fin' using PigStorage('\t') AS (s_no:int,case_status:chararray, employer_name:chararray, soc_name:chararray, job_title:chararray, full_time_position:chararray,prevailing_wage:long,year:chararray, worksite:chararray, longitute:long, latitute:long);
--dump pro;
kk = filter pro by case_status == 'CERTIFIED' or case_status == 'CERTIFIED-WITHDRAWN';
--dump kk;
pr = filter kk by $5 == 'Y';
--dump pr;
kl = group pr by ($7,$4);
--dump kl;
ad = foreach kl generate flatten(group), AVG(pr.$6);
--dump ad;

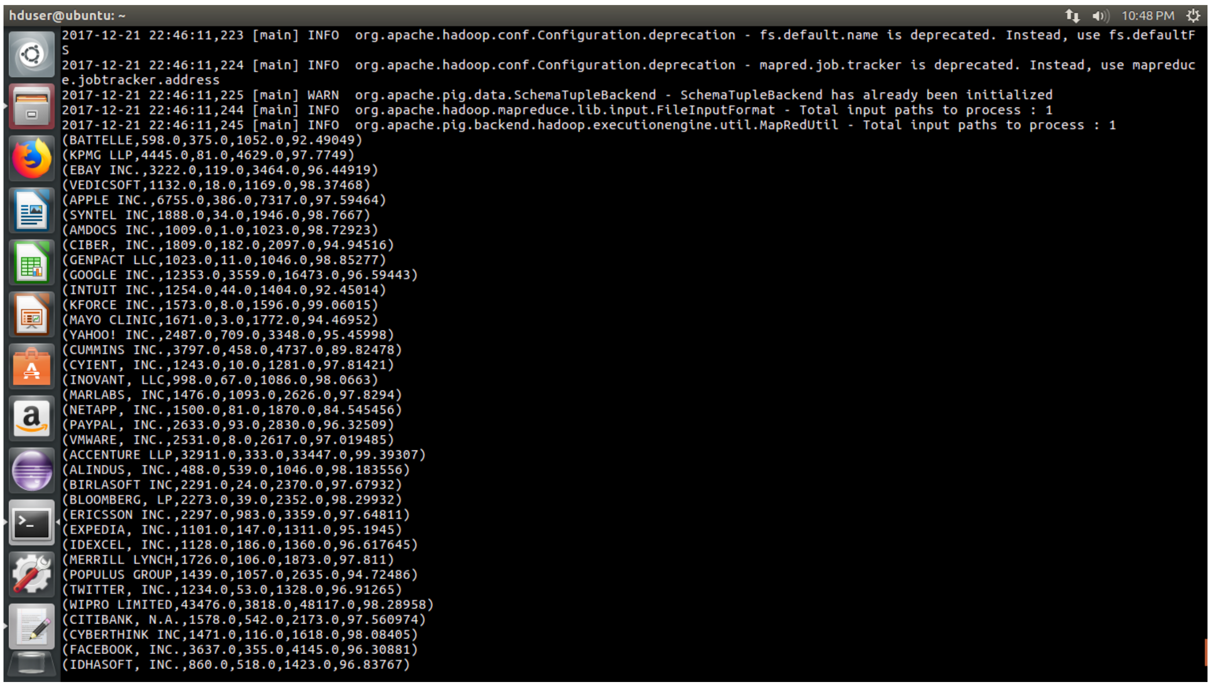
```
hy = filter ad by $0 == '2011';
--dump hy;
uy = order hy by $2 desc;
--dump uy;
hi = filter ad by $0 == '2012';
--dump hi;
uy12 = order hi by $2 desc;
--dump uy12;
hyu = filter ad by $0 == '2013';
--dump hyu;
uy13 = order hyu by $2 desc;
--dump uy13;.
hy14 = filter ad by $0 == '2014';
--dump hy14;
uy14 = order hy14 by $2 desc;
--dump uy14;
hy15 = filter ad by $0 == '2015';
--dump hy15;
uy15 = order hy15 by $2 desc;
--dump uy15;
hy16 = filter ad by $0 == '2016';
--dump hy16;
uy16 = order hy16 by $2 desc;
--dump uy16;
uni = UNION uy,uy12,uy13,uy14,uy15,uy16;
--dump uni;
de = order uni by $0 asc;
dump de;
store de into '/home/hduser/Downloads/8yans';
```

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

```
pro = load '/project/fin' using PigStorage('\t') AS (s_no:int,case_status:chararray,
employer_name:chararray, soc_name:chararray, job_title:chararray,
full_time_position:chararray,prevailing_wage:long,year:int, worksite:chararray,
longitute:long, latitute:long);
--dump pro;
kk = group pro by employer_name;
--dump kk;
```

```
pp = foreach kk generate $0, (float)COUNT(pro.employer_name);
--dump pp;
aa = filter pro by case_status == 'CERTIFIED' or case_status == 'CERTIFIED-WITHDRAWN';
--dump aa;
ff = group aa by $2;
ss = foreach ff generate $0, (float)COUNT(aa.$1);
--dump ss;
--aa1 = filter pro by case_status == 'CERTIFIED-WITHDRAWN';
--dump aa1;
--ss1 = foreach ff generate $0, (float)COUNT(aa.$1);
--dump ss1;
dd = join ss by $0, pp by $0;
--dump dd;
gh = foreach dd generate $0,$1,$3;
--dump gh;
vv = foreach gh generate $0,$2,($1/$2)*100;
--dump vv;
fg = filter vv by $1>=1000;;
--dump fg;
re = filter fg by $2>70.0;
--dump re;
fr = order re by $2 desc;
dump fr;
store fr into '/home/hduser/Downloads/9ans';
```

OUTPUT SCREENSHOT:



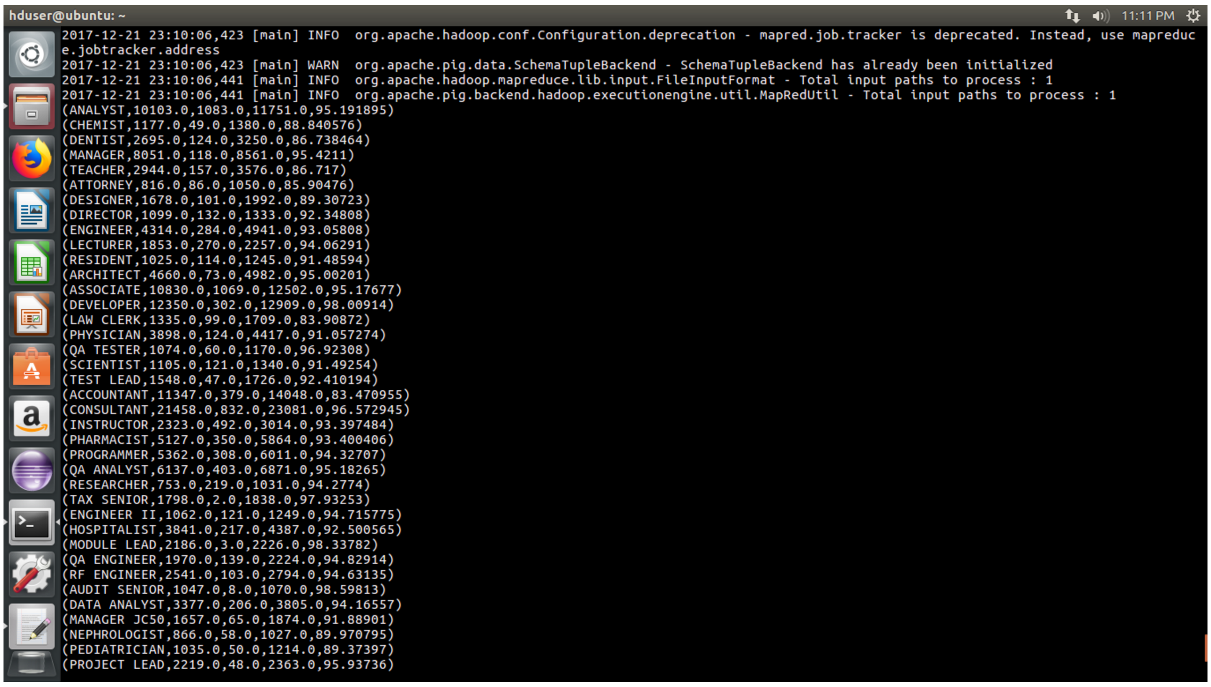
```
hduser@ubuntu: ~                                                    10:48 PM
2017-12-21 22:46:11,223 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultF
S
2017-12-21 22:46:11,224 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduc
e.jobtracker.address
2017-12-21 22:46:11,225 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-21 22:46:11,244 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-21 22:46:11,245 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(BATTELLE,598.0,375.0,1052.0,92.49049)
(KPMG LLP,4445.0,81.0,4629.0,97.7749)
(EBAY INC.,3222.0,119.0,3464.0,96.44919)
(VEDICSOFT,1132.0,18.0,1169.0,98.37468)
(APPLE INC.,6755.0,386.0,7317.0,97.59464)
(SYNTEL INC,1888.0,34.0,1946.0,98.7667)
(AMDOCS INC.,1009.0,1.0,1023.0,98.72923)
(CIBER, INC.,1809.0,182.0,2097.0,94.94516)
(GENPACT LLC,1023.0,11.0,1046.0,98.85277)
(GOOGLE INC.,12353.0,3559.0,16473.0,96.59443)
(INTUIT INC.,1254.0,44.0,1404.0,92.45014)
(KFORCE INC.,1573.0,8.0,1596.0,99.06015)
(MAYO CLINIC,1671.0,3.0,1772.0,94.46952)
(YAHOO! INC.,2487.0,709.0,3348.0,95.45998)
(CUMMINS INC.,3797.0,458.0,4737.0,89.82478)
(CYIENT, INC.,1243.0,10.0,1281.0,97.81421)
(INOVANT, LLC,998.0,67.0,1086.0,98.0663)
(MARLABS, INC,1476.0,1093.0,2626.0,97.8294)
(NETAPP, INC.,1500.0,81.0,1870.0,84.545456)
(PAYPAL, INC.,2633.0,93.0,2830.0,96.32509)
(VMWARE, INC.,2531.0,8.0,2617.0,97.019485)
(ACCENTURE LLP,32911.0,333.0,33447.0,99.39307)
(ALINDUS, INC.,488.0,539.0,1046.0,98.183556)
(BIRLASOFT INC,2291.0,24.0,2370.0,97.67932)
(BLOOMBERG, LP,2273.0,39.0,2352.0,98.29932)
(ERICSSON INC.,2297.0,983.0,3359.0,97.64811)
(EXPEDIA, INC.,1101.0,147.0,1311.0,95.1945)
(IDEXCEL, INC.,1128.0,186.0,1360.0,96.617645)
(MERRILL LYNCH,1726.0,106.0,1873.0,97.811)
(POPULUS GROUP,1439.0,1057.0,2635.0,94.72486)
(TWITTER, INC.,1234.0,53.0,1328.0,96.91265)
(WIPRO LIMITED,43476.0,3818.0,48117.0,98.28958)
(CITIBANK, N.A.,1578.0,542.0,2173.0,97.560974)
(CYBERTHINK INC,1471.0,116.0,1618.0,98.08405)
(FACEBOOK, INC.,3637.0,355.0,4145.0,96.30881)
(IDHASOFT, INC.,860.0,518.0,1423.0,96.83767)
```

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

```
pro = load '/project/fin' using PigStorage('\t') AS (s_no:int,case_status:chararray, employer_name:chararray, soc_name:chararray, job_title:chararray, full_time_position:chararray,prevailing_wage:long,year:int, worksite:chararray, longitute:long, latitute:long);
--dump pro;
kk = group pro by job_title;
--dump kk;
pp = foreach kk generate $0, (float)COUNT(pro.job_title);
--dump pp;
aa = filter pro by case_status == 'CERTIFIED' or case_status == 'CERTIFIED-WITHDRAWN';
--dump aa;
ff = group aa by $4;
ss = foreach ff generate $0, (float)COUNT(aa.$1);
--dump ss;
--aa1 = filter pro by case_status == 'CERTIFIED-WITHDRAWN';
--dump aa1;
```

```
--ss1 = foreach ff generate $0, (float)COUNT(aa.$1);
--dump ss1;
dd = join ss by $0, pp by $0;
--dump dd;
gh = foreach dd generate $0,$1,$3;
--dump gh;
vv = foreach gh generate $0,$2,($1/$2)*100;
--dump vv;
fg = filter vv by $1>=1000;;
--dump fg;
re = filter fg by $2>70.0;
--dump re;
fr = order re by $2 desc;
dump fr;
store fr into '/user/hive/warehouse/niit.db/h1b_final/out';
```

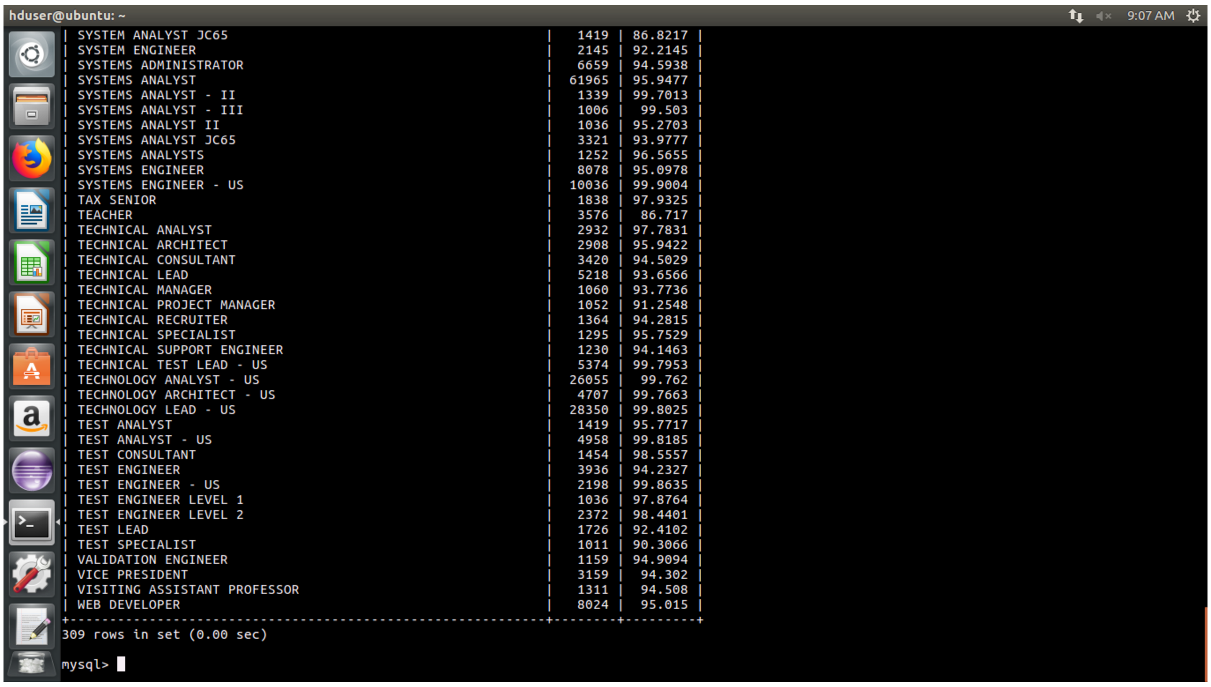
OUTPUT SCREENSHOT:



```
2017-12-21 23:10:06,423 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduc
e.jobtracker.address
2017-12-21 23:10:06,423 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-21 23:10:06,441 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-21 23:10:06,441 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(ANALYST,10103.0,1083.0,11751.0,95.191895)
(CHEMIST,1177.0,49.0,1380.0,88.840576)
(DENTIST,2695.0,124.0,3250.0,86.738464)
(MANAGER,8051.0,118.0,8561.0,95.4211)
(TEACHER,2944.0,157.0,3576.0,86.717)
(ATTORNEY,816.0,86.0,1050.0,85.90476)
(DESIGNER,1678.0,101.0,1992.0,89.30723)
(DIRECTOR,1099.0,132.0,1333.0,92.34808)
(ENGINEER,4314.0,284.0,4941.0,93.05808)
(LECTURER,1853.0,270.0,2257.0,94.06291)
(RESIDENT,1025.0,114.0,1245.0,91.48594)
(ARCHITECT,4660.0,73.0,4982.0,95.00201)
(ASSOCIATE,10830.0,1069.0,12502.0,95.17677)
(DEVELOPER,12350.0,302.0,12909.0,98.00914)
(LAW CLERK,1335.0,99.0,1709.0,83.90872)
(PHYSICIAN,3898.0,124.0,4417.0,91.057274)
(QA TESTER,1074.0,60.0,1170.0,96.92308)
(SCIENTIST,1105.0,121.0,1340.0,91.49254)
(TEST LEAD,1548.0,47.0,1726.0,92.410194)
(ACCOUNTANT,11347.0,379.0,14048.0,83.470955)
(CONSULTANT,21458.0,832.0,23081.0,96.572945)
(INSTRUCTOR,2323.0,492.0,3014.0,93.397484)
(PHARMACIST,5127.0,350.0,5864.0,93.400406)
(PROGRAMMER,5362.0,308.0,6011.0,94.32707)
(QA ANALYST,6137.0,403.0,6871.0,95.18265)
(RESEARCHER,753.0,219.0,1031.0,94.2774)
(TAX SENIOR,1798.0,2.0,1838.0,97.93253)
(ENGINEER II,1062.0,121.0,1249.0,94.715775)
(HOSPITALIST,3841.0,217.0,4387.0,92.500565)
(MODULE LEAD,2186.0,3.0,2226.0,98.33782)
(QA ENGINEER,1970.0,139.0,2224.0,94.82914)
(RF ENGINEER,2541.0,103.0,2794.0,94.63135)
(AUDIT SENIOR,1047.0,8.0,1070.0,98.59813)
(DATA ANALYST,3377.0,206.0,3805.0,94.16557)
(MANAGER JC50,1657.0,65.0,1874.0,91.88901)
(NEPHROLOGIST,866.0,58.0,1027.0,89.970795)
(PEDIATRICIAN,1035.0,50.0,1214.0,89.37397)
(PROJECT LEAD,2219.0,48.0,2363.0,95.93736)
```

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

```
sqoop export --connect jdbc:mysql://localhost/proj10 --username root --password 'secureword' --table sqoo --update-mode allowinsert --update-key job_title --export-dir /user/warehouse/hive/niit.db/h1b_final/out10 --input-fields-terminated-by '\t' ;
```



```
hduser@ubuntu: ~
| SYSTEM ANALYST JC65                                            |    1419 | 86.8217 |
| SYSTEM ENGINEER                                                |    2145 | 92.2145 |
| SYSTEMS ADMINISTRATOR                                          |    6659 | 94.5938 |
| SYSTEMS ANALYST                                                |   61965 | 95.9477 |
| SYSTEMS ANALYST - II                                           |    1339 | 99.7013 |
| SYSTEMS ANALYST - III                                          |    1006 |  99.503 |
| SYSTEMS ANALYST II                                             |    1036 | 95.2703 |
| SYSTEMS ANALYST JC65                                           |    3321 | 93.9777 |
| SYSTEMS ANALYSTS                                               |    1252 | 96.5655 |
| SYSTEMS ENGINEER                                               |    8078 | 95.0978 |
| SYSTEMS ENGINEER - US                                          |   10036 | 99.9004 |
| TAX SENIOR                                                     |    1838 | 97.9325 |
| TEACHER                                                        |    3576 |  86.717 |
| TECHNICAL ANALYST                                              |    2932 | 97.7831 |
| TECHNICAL ARCHITECT                                            |    2908 | 95.9422 |
| TECHNICAL CONSULTANT                                           |    3420 | 94.5029 |
| TECHNICAL LEAD                                                 |    5218 | 93.6566 |
| TECHNICAL MANAGER                                              |    1060 | 93.7736 |
| TECHNICAL PROJECT MANAGER                                      |    1052 | 91.2548 |
| TECHNICAL RECRUITER                                            |    1364 | 94.2815 |
| TECHNICAL SPECIALIST                                           |    1295 | 95.7529 |
| TECHNICAL SUPPORT ENGINEER                                     |    1230 | 94.1463 |
| TECHNICAL TEST LEAD - US                                       |    5374 | 99.7953 |
| TECHNOLOGY ANALYST - US                                        |   26055 |  99.762 |
| TECHNOLOGY ARCHITECT - US                                      |    4707 | 99.7663 |
| TECHNOLOGY LEAD - US                                           |   28350 | 99.8025 |
| TEST ANALYST                                                   |    1419 | 95.7717 |
| TEST ANALYST - US                                              |    4958 | 99.8185 |
| TEST CONSULTANT                                                |    1454 | 98.5557 |
| TEST ENGINEER                                                  |    3936 | 94.2327 |
| TEST ENGINEER - US                                             |    2198 | 99.8635 |
| TEST ENGINEER LEVEL 1                                          |    1036 | 97.8764 |
| TEST ENGINEER LEVEL 2                                          |    2372 | 98.4401 |
| TEST LEAD                                                      |    1726 | 92.4102 |
| TEST SPECIALIST                                                |    1011 | 90.3066 |
| VALIDATION ENGINEER                                            |    1159 | 94.9094 |
| VICE PRESIDENT                                                 |    3159 |  94.302 |
| VISITING ASSISTANT PROFESSOR                                   |    1311 |  94.508 |
| WEB DEVELOPER                                                  |    8024 |  95.015 |
+----------------------------------------------------------------+---------+---------+
309 rows in set (0.00 sec)

mysql>
```

Shell script

```
#!/bin/bash
show_menu()
{
    NORMAL=`echo "\033[m"`
    MENU=`echo "\033[36m"` #Blue
    NUMBER=`echo "\033[33m"` #yellow
    FGRED=`echo "\033[41m"`
    RED_TEXT=`echo "\033[31m"`
    ENTER_LINE=`echo "\033[33m"`
```

```
   echo -e "${MENU}*********************APP MENU**********************${NORMAL}"
   echo -e "${MENU}**${NUMBER} 1)${MENU}  1a ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 2)${MENU}  1b ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 3)${MENU}  2a ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 4)${MENU}  2b ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 5)${MENU}  3 ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 6)${MENU}  4 ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 7)${MENU}  5a ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 8)${MENU}  5b ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 9)${MENU}  6 ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 10)${MENU} 7 ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 11)${MENU} 8 ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 12)${MENU} 9 ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 13)${MENU} 10 ${NORMAL}"
   echo -e "${MENU}**${NUMBER} 14)${MENU} 11 ${NORMAL}"
   echo -e "${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;
      option_picked "1a) Is the number of petitions with Data Engineer job title
increasing over time?";
      bash /home/hduser/Desktop/1a.sh
show_menu;
      ;;

      2) clear;
      option_picked "1b) Find top 5 job titles who are having highest avg growth
in applications ";
      pig /home/hduser/1b.pig
      show_menu;
      ;;

      3) clear;
      option_picked "2a) Which part of the US has the most Data Engineer jobs
for each year";
bash /home/hduser/Desktop/3.sh

      show_menu;
      ;;

      4) clear;
      option_picked "2b) find top 5 locations in the US who have got certified
visa for each year";
      bash /home/hduser/Desktop/2b.sh
      show_menu;
      ;;

   5) clear;
      option_picked "3)Which industry(SOC_NAME) has the most number of
Data Scientist positions?";
pig /home/hduser/3.pig
      show_menu;
```

```
        ;;

        6) clear;
        option_picked "4)Which top 5 employers file the most petitions each
year?";
bash /home/hduser/Desktop/4.sh
        show_menu;
        ;;

        7) clear;
        option_picked "5) Find the most popular top 10 job positions for H1B visa
applications for each year?a) for all the applications";
bash /home/hduser/Desktop/5a.sh
        show_menu;
        ;;

    8) clear;
        option_picked "5) Find the most popular top 10 job positions for H1B visa
applications for each year?b) for only certified applications.";
       bash /home/hduser/Desktop/5b.sh
       show_menu;
       ;;
9) clear;
option_picked "6) Find the percentage and the count of each case status on total
applications for each year. Create a line graph depicting the pattern of All the
cases over the period of time ";
pig /home/hduser/6.pig
show_menu;
        ;;

10) clear;
        option_picked "7) Create a bar graph to depict the number of applications
for each year";
bash /home/hduser/Desktop/7.sh
        show_menu;
        ;;

11) clear;
```

```
        option_picked "8) Find the average Prevailing Wage for each Job for each
Year (take part time and full time separate). Arrange the output in descending
order";
 echo -e "${MENU}Select Full Time Job or Part Time Job ${NORMAL}"
        echo -e "${MENU}**${NUMBER} 1)${MENU} Full Time Job
${NORMAL}"
        echo -e "${MENU}**${NUMBER} 2)${MENU} Part Time Job
${NORMAL}"
        read job
        case $job in
                1)   echo "FULL TIME JOB SELECTED"
                        pig /home/hduser/8y.pig
                        ;;

                2)    echo "PART TIME JOB SELECTED"

                    pig /home/hduser/8n.pig
                    ;;
                   *) echo "Please Select one among the option[1-2]";;
                esac
        show_menu;
        ;;

12) clear;
        option_picked "9) Which are the employers along with the number of
petitions who have the success rate more than 70%  in petitions. (total petitions
filed 1000 OR more than 1000) ?";
pig /home/hduser/9.pig
        show_menu;
        ;;

13) clear;
        option_picked "10) Which are the  job positions along with the number of
petitions which have the success rate more than 70%  in petitions (total petitions
filed 1000 OR more than 1000)?";
pig /home/hduser/10.pig
        show_menu;
        ;;

14) clear;
```

```
        option_picked "11) Export result for question no 10 to MySql database.";
sqoop export --connect jdbc:mysql://localhost/proj10 --username root --P --table
sqoo --update-mode allowinsert --update-key job_title --export-dir
/user/hive/warehouse/niit.db/h1b_final/out --input-fields-terminated-by '\t' ;
        show_menu;
        ;;

        \n) exit;
        ;;

        *) clear;
        option_picked "Pick an option from the menu";
        show_menu;
        ;;
    esac
fi



done.
```

OUTPUT:

```
**********************APP MENU***********************
** 1)  1a
** 2)  1b
** 3)  2a
** 4)  2b
** 5)  3
** 6)  4
** 7)  5a
** 8)  5b
** 9)  6
** 10) 7
** 11) 8
** 12) 9
** 13) 10
** 14) 11
*********************************************
```

Please enter a menu option and enter or enter to exit.