**BIGDATA OVERVIEW**

“90% of the world’s data was generated in the last few years.”

Due to the advent of new technologies, devices, and communication means like social networking sites, the amount of data produced by mankind is growing rapidly every year. The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. If you pile up the data in the form of disks it may fill an entire football field. The same amount was created in every two days in 2011, and in every ten minutes in 2013. This rate is still growing enormously. Though all this information produced is meaningful and can be useful when processed, it is being neglected.

**BIGDATA**

Big Data is a collection of large datasets that cannot be processed using traditional computing techniques. It is not a single technique or a tool, rather it involves many areas of business and technology.

**WHAT COMES UNDER BIG DATA?**

Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

* Black Box Data: It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
* Social Media Data: Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.
* Stock Exchange Data: The stock exchange data holds information about the ‘buy’ and ‘sell’ decisions made on a share of different companies made by the customers.
* Power Grid Data: The power grid data holds information consumed by a particular node with respect to a base station.
* Transport Data: Transport data includes model, capacity, distance and availability of a vehicle.
* Search Engine Data: Search engines retrieve lots of data from different databases.

Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types.

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

**BIG DATA CHALLENGES**

The major challenges associated with big data are as follows:

* Capturing data
* Storage
* Searching
* Sharing
* Transfer
* Analysis

**BIGDATA SOLUTIONS**

**TRADITIONAL APPROACH**

In this approach, an enterprise will have a computer to store and process big data. For storage purpose, the programmers will take the help of their choice of database vendors such as Oracle, IBM, etc. In this approach, the user interacts with the application, which in turn handles the part of data storage and analysis.

**Limitation**

This approach works fine with those applications that process less voluminous data that can be accommodated by standard database servers, or up to the limit of the processor that is processing the data. But when it comes to dealing with huge amounts of scalable data, it is a hectic task to process such data through a single database bottleneck.

**GOOGLE’S SOLUTION**

Google solved this problem using an algorithm called MapReduce. This algorithm divides the task into small parts and assigns them to many computers, and collects the results from them which when integrated, form the result dataset. Using the solution provided by Google, Doug Cutting and his team developed an Open Source Project called HADOOP. Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel with others. In short, Hadoop is used to develop applications that could perform complete statistical analysis on huge amounts of data.

**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. The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage. At its core, Hadoop has two major layers namely:

* Processing/Computation layer (MapReduce),
* Storage layer (Hadoop Distributed File System).

**ADVANTAGES OF HADOOP**

* Scalable
* Cost Effective
* Flexible
* Fast
* Resilient to failure
* Compatible on all the platforms since it is Java based.

**HADOOP ECOSYSTEM**

**HDFS**

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on 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.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules:

* **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules.
* **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.

**MAPREDUCE**

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

**YARN**

**Apache Yarn** – “**Y**et **A**nother **R**esource **N**egotiator” is the resource management layer of **Hadoop**. The Yarn was introduced in Hadoop 2.x. Yarn allows different data processing engines like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in**HDFS** (Hadoop Distributed File System).

**HIVE**

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy. It provides SQL type language for querying called HiveQL or HQL. It stores schema in a database and processed data into HDFS. Traditional SQL queries must be implemented in the [MapReduce](https://en.wikipedia.org/wiki/MapReduce) Java API to execute SQL applications and queries over distributed data. Hive provides the necessary SQL abstraction to integrate SQL-like queries ([HiveQL](https://en.wikipedia.org/wiki/HiveQL" \o "HiveQL)) into the underlying Java without the need to implement queries in the low-level Java API. Since most data warehousing applications work with SQL-based querying languages, Hive aids portability of SQL-based applications to Hadoop.

**PIG**

Pig is a high level programming language useful for analyzing large data sets. Pig uses PigLatinlanguage. Pig was a result of development effort at Yahoo! Pig enables people to focus more on **analyzing bulk data sets and to spend less time in writing Map-Reduce programs.** Similar to Pigs, who eat anything, the Pig programming language is designed to work upon any kind of data. That's why the name, Pig!

Pig has two execution modes:

* Local mode : In this mode, Pig runs in a single JVM and makes use of local file system. This mode is suitable only for analysis of small data sets using Pig.
* Map Reduce mode: In this mode, queries written in Pig Latin are translated into MapReduce jobs and are run on a Hadoop cluster (cluster may be pseudo or fully distributed). MapReduce mode with fully distributed cluster is useful of running Pig on large data sets.

**HBase**

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. HBase is a data model that is similar to Google’s big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS). One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.

**Sqoop**

Sqoop imports data from external sources into Hadoop components like HDFS, HBase and Hive. It also exports data from Hadoop to other sources. Sqoop works with relational databases such as Teradata, Netezza, Oracle, MySQL, Postgres etc.

**Flume**

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

**Other Components**

* **Ambari**– A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig, and Sqoop.
* **Avro**– A data serialization system.
* **HCatalog**- Helps data processing tools read and write data on the grid. It supports MapReduce and Pig.
* **Mahout**– A scalable machine learning and data mining library.
* **Oozie** - Workflow scheduler that was developed as part of the Apache Hadoop project. It manages how workflows start and execute, and also controls the execution path.
* **Zookeper**– A high-performance coordination service for distributed applications.

**HADOOP INSTALLATION**

Hadoop is supported by GNU/Linux platform and its flavors. Therefore, we have to install a Linux operating system for setting up Hadoop environment. In case you have an OS other than Linux, you can install a Virtualbox software in it and have Linux inside the Virtualbox.

**Installing Java**

Java is the main prerequisite for Hadoop.0

**user@laptop(local directory):~$** **sudo apt-get update**

**user@laptop:~$** **sudo apt-get install default-jdk**

**user@laptop:~$ java -version**

**java version "1.7.0\_65"**

**Adding a dedicated Hadoop user**

**user@laptop:~$ sudo addgroup hadoop**

**Adding group `hadoop' (GID 1002) ...**

**Done.**

**user@laptop:~$ sudo adduser --ingroup hadoop hduser**

**Adding user `hduser' ...**

**Adding new user `hduser' (1001) with group `hadoop' ...**

**Creating home directory `/home/hduser' ...**

**Installing SSH**

**ssh** has two main components:

1. **ssh** : The command we use to connect to remote machines - the client.
2. **sshd** : The daemon that is running on the server and allows clients to connect to the server.

The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install **ssh** first. Use this command to do that :

**user@laptop:~$ sudo apt-get install ssh**

This will install ssh on our machine. If we get something similar to the following, we can think it is setup properly:

**user@laptop:~$ which ssh**

**/usr/bin/ssh**

**user@laptop:~$ which sshd**

**/usr/sbin/sshd**

**Create and Setup SSH Certificates**

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.

**user@laptop:~$ su hduser**

**Password:**

**hduser@laptop:/home/user$ cd ~**

**hduser@laptop:~$ ssh-keygen -t rsa -P ""**

**Your public key has been saved in /home/hduser/.ssh/id\_rsa.pub.**

**hduser@laptop:~$ cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys**

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password

**Install Hadoop**

**hduser@laptop:~$ su hduser**

**wget http://archive.apache.org/dist/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz**

**hduser@laptop:~$ tar xvzf hadoop-2.6.0.tar.gz**

**a folder would be created by the name of hadoop-2.6.0**

**rename this folder to hadoop**

**mv hadoop-2.6.0 hadoop**

We want to move the Hadoop installation to the **/usr/local/hadoop** directory using the following command:

**hduser@laptop:~$ sudo mv hadoop /usr/local/**

**[sudo] password for hduser:**

**hduser is not in the sudoers file. This incident will be reported.**

This error can be resolved by logging in as a root user, and then add **hduser** to **sudo**:

**hduser@laptop:~/hadoop-2.6.0$ su user (root user)**

**Password:**

**user@laptop:/home/hduser$ sudo adduser hduser sudo**

**[sudo] password for user:**

**Adding user `hduser' to group `sudo' ...**

**Adding user hduser to group sudo**

**Done.**

Now, the **hduser** has root priviledge, we can move the Hadoop installation to the**/usr/local/hadoop** directory without any problem.

**user@laptop:/home/hduser$ sudo su hduser**

**cd ~**

**hduser@laptop:~$ sudo mv hadoop /usr/local/**

**hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop**

**Setup Configuration Files**

**1. ~/.bashrc:**

Before editing the **.bashrc** file in our home directory, we need to find the path where Java has been installed to set the **JAVA\_HOME** environment variable using the following command:

**hduser@laptop update-alternatives --config java**

**There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java**

**Nothing to configure.**

Now we can append the following to the end of **~/.bashrc**:

**hduser@laptop:~$ nano ~/.bashrc OR gedit ~/.bashrc**

**#HADOOP VARIABLES START**

**export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64**

**export HADOOP\_INSTALL=/usr/local/hadoop**

**export PATH=$PATH:$HADOOP\_INSTALL/bin**

**export PATH=$PATH:$HADOOP\_INSTALL/sbin**

**export HADOOP\_MAPRED\_HOME=$HADOOP\_INSTALL**

**export HADOOP\_COMMON\_HOME=$HADOOP\_INSTALL**

**export HADOOP\_HDFS\_HOME=$HADOOP\_INSTALL**

**export YARN\_HOME=$HADOOP\_INSTALL**

**export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_INSTALL/lib/native**

**export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_INSTALL/lib"**

**#HADOOP VARIABLES END**

**hduser@laptop:~$ source ~/.bashrc**

note that the JAVA\_HOME should be set as the path just before the '.../bin/':

**hduser@laptop:~$ javac -version**

**javac 1.7.0\_75**

**2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh**

We need to set JAVA**\_**HOME by modifying hadoop-env.sh file.

**hduser@laptop:~$ nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh**

**export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64**

**3. /usr/local/hadoop/etc/hadoop/core-site.xml**:

The /usr/local/hadoop/etc/hadoop/core-site.xml file contains configuration properties that Hadoop uses when starting up.   
This file can be used to override the default settings that Hadoop starts with.

**hduser@laptop:~$ sudo mkdir -p /app/hadoop/tmp**

**hduser@laptop:~$ sudo chown -R hduser:hadoop /app/hadoop/tmp**

**hduser@laptop:~$ nano /usr/local/hadoop/etc/hadoop/core-site.xml**

**<configuration>**

**<property>**

**<name>hadoop.tmp.dir</name>**

**<value>/app/hadoop/tmp</value>**

**<description>A base for other temporary directories.</description>**

**</property>**

**<property>**

**<name>fs.default.name</name>**

**<value>hdfs://localhost:54310</value>**

**<description>The name of the default file system. A URI whose**

**scheme and authority determine the FileSystem implementation. The**

**uri's scheme determines the config property (fs.SCHEME.impl) naming**

**the FileSystem implementation class. The uri's authority is used to**

**determine the host, port, etc. for a filesystem.</description>**

**</property>**

**</configuration>**

**4. /usr/local/hadoop/etc/hadoop/mapred-site.xml**

By default, the /usr/local/hadoop/etc/hadoop/ folder contains   
/usr/local/hadoop/etc/hadoop/mapred-site.xml.template   
file which has to be renamed/copied with the name mapred-site.xml:

**hduser@laptop:~$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml**

**hduser@laptop:~$ nano /usr/local/hadoop/etc/hadoop/mapred-site.xml**

The mapred-site.xml file is used to specify which framework is being used for MapReduce.

**<configuration>**

**<property>**

**<name>mapred.job.tracker</name>**

**<value>localhost:54311</value>**

**<description>The host and port that the MapReduce job tracker runs**

**at. If "local", then jobs are run in-process as a single map**

**and reduce task.**

**</description>**

**</property>**

**</configuration>**

**5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml**

The /usr/local/hadoop/etc/hadoop/hdfs-site.xml file needs to be configured for each host in the cluster that is being used.   
It is used to specify the directories which will be used as the namenode and thedatanode on that host.

Before editing this file, we need to create two directories which will contain the namenode and the datanode for this Hadoop installation.   
This can be done using the following commands:

**hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop\_store/hdfs/namenode**

**hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop\_store/hdfs/datanode**

**hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop\_store**

**hduser@laptop:~$ nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml**

**<configuration>**

**<property>**

**<name>dfs.replication</name>**

**<value>1</value>**

**<description>Default block replication.**

**The actual number of replications can be specified when the file is created.**

**The default is used if replication is not specified in create time.**

**</description>**

**</property>**

**<property>**

**<name>dfs.namenode.name.dir</name>**

**<value>file:/usr/local/hadoop\_store/hdfs/namenode</value>**

**</property>**

**<property>**

**<name>dfs.datanode.data.dir</name>**

**<value>file:/usr/local/hadoop\_store/hdfs/datanode</value>**

**</property>**

**</configuration>**

**Format the New Hadoop Filesystem**

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current** directory   
under **/usr/local/hadoop\_store/hdfs/namenode** folder:

**hduser@laptop:~$ hadoop namenode -format**

Note that hadoop namenode -format command should be executed once before we start using Hadoop. If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

**Starting Hadoop**

Now it's time to start the newly installed single node cluster.   
We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

**hduser@laptop:~$ start-all.sh**

**This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh**

**15/04/18 16:43:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable**

**Starting namenodes on [localhost]**

**localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-laptop.out**

**localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-laptop.out**

**Starting secondary namenodes [0.0.0.0]**

**0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-laptop.out**

**15/04/18 16:43:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable**

**starting yarn daemons**

**starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-laptop.out**

**localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-laptop.out**

We can check if it's really up and running:

**hduser@laptop:~$ jps**

**9026 NodeManager**

**7348 NameNode**

**9766 Jps**

**8887 ResourceManager**

**7507 DataNode**

**7350 Secondary Namenode**

**Stopping Hadoop**

We run **stop-all.sh** or (**stop-dfs.sh** and **stop-yarn.sh**) to stop all the daemons running on our machine:

**hduser@laptop:/usr/local/hadoop/sbin$ stop-all.sh**

**This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh**

**15/04/18 15:46:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable**

**Stopping namenodes on [localhost]**

**localhost: stopping namenode**

**localhost: stopping datanode**

**Stopping secondary namenodes [0.0.0.0]**

**0.0.0.0: no secondarynamenode to stop**

**15/04/18 15:46:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable**

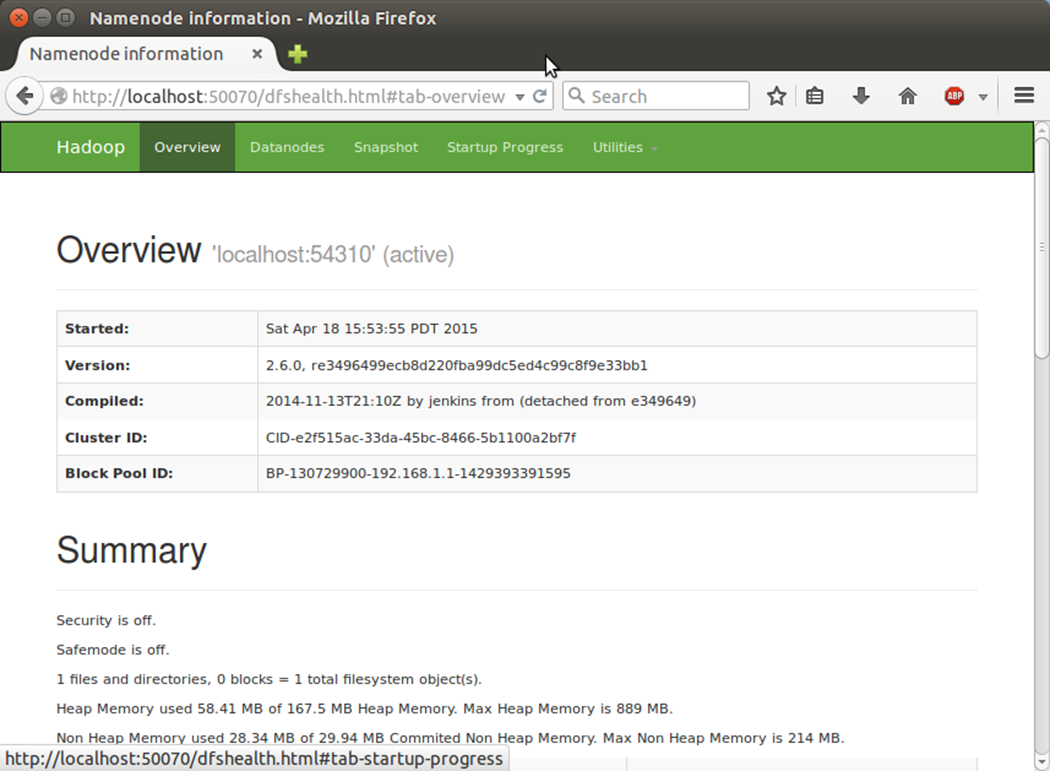
**stopping yarn daemons**

**stopping resourcemanager**

**localhost: stopping nodemanager**

**no proxyserver to stop**

**http://127.0.0.1:50070/ - web UI of the NameNode daemon**

****

**PROJECT: H1-B 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. The data set has nearly 3 million records.

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.

We will be performing analysis on the H1B visa applicants between the years 2011-2016. We need to cleanse the dataset first to analyse it easier.

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
* Certified withdrawn
* Denied
* Withdrawn
* Pending quality and compliance review
* Invalidated
* Rejected
* NA
* 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

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

full\_time\_position string,prevailing\_wage bigint,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;

select \* from h1b\_applications limit 3;

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

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

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

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", ""), prevailing\_wage,

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

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

"") FROM h1b\_applications where case\_status != "NA";

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 bigint,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,

case when prevailing\_wage is null then 100000

else prevailing\_wage end,

year, worksite, longitute, latitute

FROM h1b\_app2;

select case\_status, count(\*) from h1b\_final group by case\_status;

CERTIFIED-WITHDRAWN 202659

WITHDRAWN 89799

CERTIFIED 2615623

DENIED 94364

total records : 3002445

The following tools were used for analysis

HDFS, MapReduce, Hive, Pig, Sqoop, Mysql, Excel, Menu based using shell script.

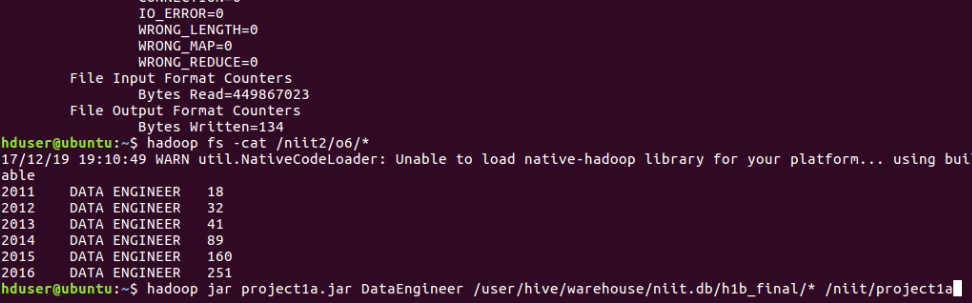
After analyzing the data, we can derive the following facts.

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

Analysed the data using Java MapReduce

|  |
| --- |
|  |
|  | Import java.io.IOException;  import org.apache.hadoop.conf.Configuration; |
|  | import org.apache.hadoop.fs.Path; |
|  | import org.apache.hadoop.io.LongWritable; |
|  | import org.apache.hadoop.io.Text; |
|  | import org.apache.hadoop.mapreduce.Job; |
|  | import org.apache.hadoop.mapreduce.Mapper; |
|  | import org.apache.hadoop.mapreduce.Reducer; |
|  | import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; |
|  | import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; |
|  |  |
|  |  |
|  | public class DataEngineer |
|  | { |
|  | 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"); |
|  | if(str[4].equals("DATA ENGINEER")) |
|  | { |
|  | con.write(new Text(str[7]), new Text(str[4])); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | public static class ReduceClass extends Reducer<Text, Text, Text, Text> |
|  | { |
|  | public void reduce(Text key, Iterable<Text> values, Context con) throws IOException, InterruptedException |
|  | { |
|  | long count=0; |
|  | String job = ""; |
|  | for(Text val:values) |
|  | { |
|  | job=val.toString(); |
|  | count++; |
|  | } |
|  | String myVal = job+"\t"+count; |
|  | con.write(key, new Text(myVal)); |
|  | } |
|  | } |
|  | public static void main(String[] args) throws Exception |
|  | { |
|  | Configuration conf = new Configuration(); |
|  | Job job = Job.getInstance(conf,"Year Application"); |
|  | job.setJarByClass(DataEngineer.class); |
|  | job.setMapperClass(MapClass.class); |
|  | job.setReducerClass(ReduceClass.class); |
|  | //job.setMapOutputKeyClass(Text.class); |
|  | //job.setMapOutputValueClass(Text.class); |
|  | job.setOutputKeyClass(Text.class); |
|  | job.setOutputValueClass(Text.class); |
|  | job.setNumReduceTasks(1); |
|  | FileInputFormat.addInputPath(job, new Path(args[0])); |
|  | FileOutputFormat.setOutputPath(job, new Path(args[1])); |
|  | System.exit(job.waitForCompletion(true) ? 0 : 1); |
|  | } |
|  | } |

Output:

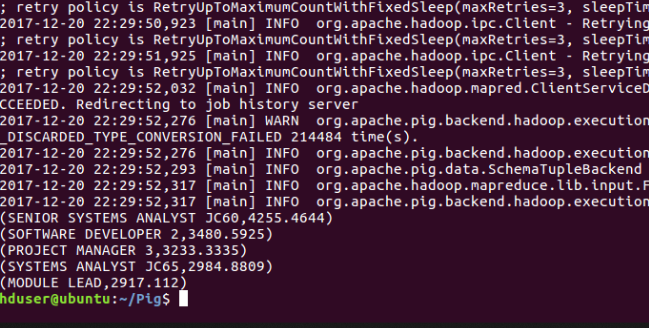
****

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

Analysed the data using Pig

|  |
| --- |
|  |
|  | h1b = load '/user/hive/warehouse/niit.db/h1b\_final/\*' using PigStorage('\t') as (sno:int, case\_status:chararray, emp\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_pos:chararray,wage:long, year:chararray, worksite:chararray, longitude:double, lattitude:double); |
|  | h1b1 = filter h1b by year=='2011'; |
|  | grp1 = group h1b1 by $4; |
|  | tot1 = foreach grp1 generate group, COUNT(h1b1.$1); |
|  | --dump tot1; |
|  |  |
|  | h1b1 = filter h1b by year=='2012'; |
|  | grp1 = group h1b1 by $4; |
|  | tot2 = foreach grp1 generate group, COUNT(h1b1.$1); |
|  |  |
|  | h1b1 = filter h1b by year=='2013'; |
|  | grp1 = group h1b1 by $4; |
|  | tot3 = foreach grp1 generate group, COUNT(h1b1.$1); |
|  |  |
|  | h1b1 = filter h1b by year=='2014'; |
|  | grp1 = group h1b1 by $4; |
|  | tot4 = foreach grp1 generate group, COUNT(h1b1.$1); |
|  |  |
|  | h1b1 = filter h1b by year=='2015'; |
|  | grp1 = group h1b1 by $4; |
|  | tot5 = foreach grp1 generate group, COUNT(h1b1.$1); |
|  |  |
|  | h1b1 = filter h1b by year=='2016'; |
|  | grp1 = group h1b1 by $4; |
|  | tot6 = foreach grp1 generate group, COUNT(h1b1.$1); |
|  |  |
|  | total = join tot1 by $0,tot2 by $0,tot3 by $0,tot4 by $0,tot5 by $0,tot6 by $0; |
|  | total = foreach total generate $0,$1,$3,$5,$7,$9,$11; |
|  | --dump total; |
|  |  |
|  | growth = foreach total generate $0,(float)($2-$1)\*100/$1,(float)($3-$2)\*100/$2,(float)($4-$3)\*100/$3,(float)($5-$4)\*100/$4,(float)($6-$5)\*100/$5; |
|  | --dump growth; |
|  | avg\_growth = foreach growth generate $0,($1+$2+$3+$4+$5)/5; |
|  | orderbydesc = limit(order avg\_growth by $1 desc) 5; |
|  | --dump orderbydesc; |
|  | store orderbydesc into '/niit/project1b'; |

Output:

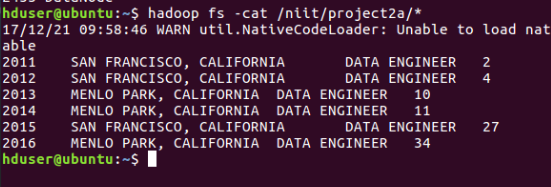


2a) Which part of the US has the most Data Engineer jobs for each year?

Analysed using Java MapReduce

|  |
| --- |
|  |
|  | 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 PartDe |
|  | { |
|  | 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 year=""; |
|  | //String job=""; |
|  | String myVal=""; |
|  | for(Text val:values) |
|  | { |
|  | String[] str = val.toString().split("\t"); |
|  | if((str[1].equals("CERTIFIED")) && (str[4].equals("DATA ENGINEER"))) |
|  | { |
|  | count++; |
|  | myVal = str[7]+"\t"+key+"\t"+str[4]; |
|  | } |
|  |  |
|  | } |
|  | String myValue = myVal+"\t"+count; |
|  | tm.put(new Long(count), new Text(myValue)); |
|  | if(tm.size()>1) |
|  | { |
|  | 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(PartDe.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:

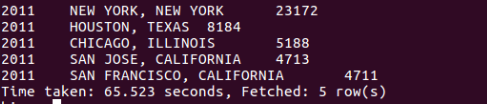


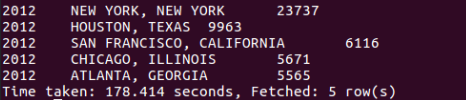
2b) find top 5 locations in the US who have got certified visa for each year.

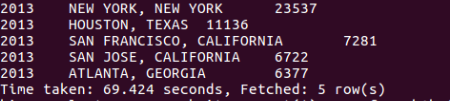
Analysed using Hive

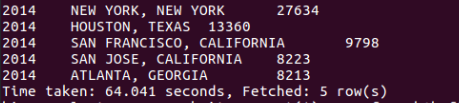
select year, worksite, count(\*) as n from h1b\_final where year=$year and case\_status='CERTIFIED' group by year, worksite order by n desc limit 5;

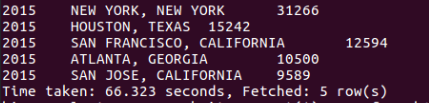
Output:

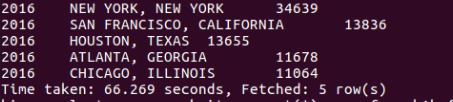










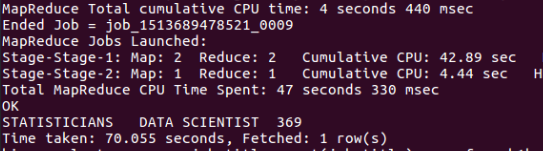


H3)Which industry(SOC\_NAME) has the most number of Data Scientist positions?

Analysed using Hive

select soc\_name,job\_title,count(job\_title) as n from h1b\_final where job\_title='DATA SCIENTIST' and case\_status='CERTIFIED' group by soc\_name,job\_title order by n desc limit 1;

Output:



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

Analysed using MapReduce

|  |
| --- |
|  |
|  | import java.io.IOException; |
|  |  |
|  | import org.apache.hadoop.conf.Configuration; |
|  | import org.apache.hadoop.fs.Path; |
|  | import org.apache.hadoop.io.LongWritable; |
|  | import org.apache.hadoop.io.NullWritable; |
|  | import org.apache.hadoop.io.Text; |
|  | import org.apache.hadoop.mapreduce.Job; |
|  | import org.apache.hadoop.mapreduce.Mapper; |
|  | import org.apache.hadoop.mapreduce.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 EmpPetition |
|  | { |
|  | 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[2]), new Text(values)); |
|  | } |
|  | } |
|  | public static class YearPartitioner 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 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, " Top petition"); |
|  | job.setJarByClass(EmpPetition.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:

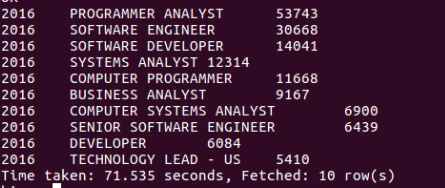
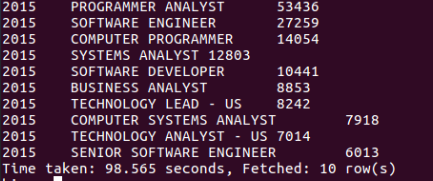
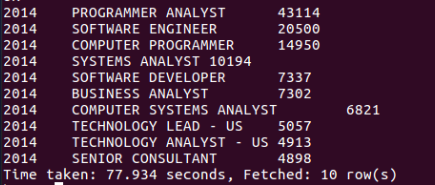
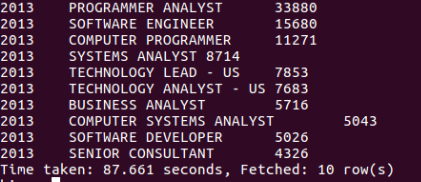
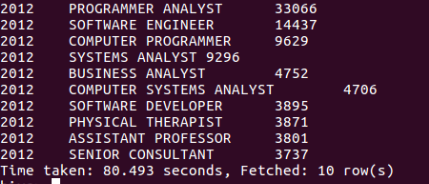
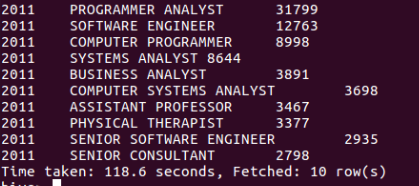


H5) Find the most popular top 10 job positions for H1B visa applications for each year? a) for all the applications

Analysed using Hive

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

Output:

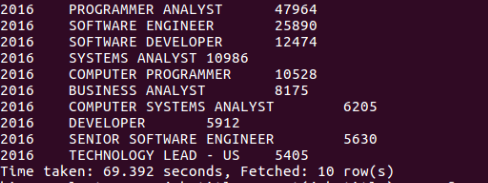
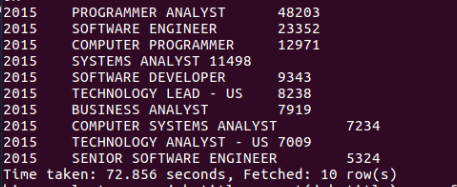
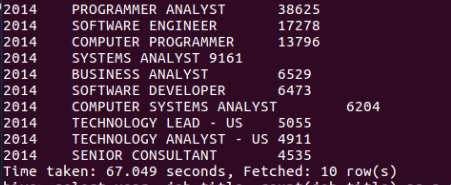
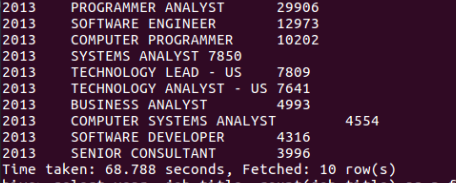
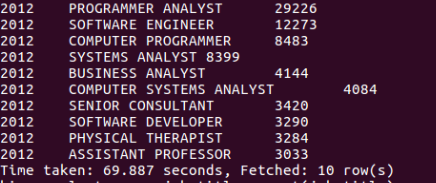
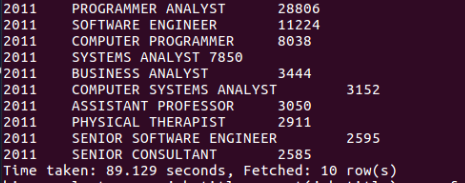


5) Find the most popular top 10 job positions for H1B visa applications for each year?b) for only certified applications.

Analysed using Hive

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

Output:

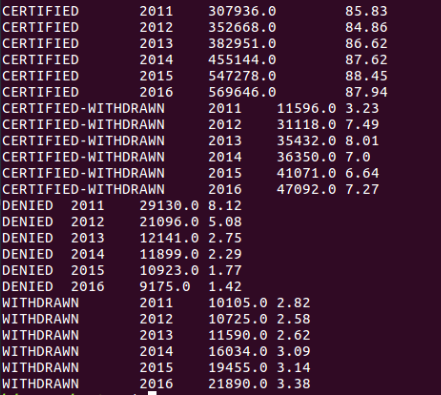


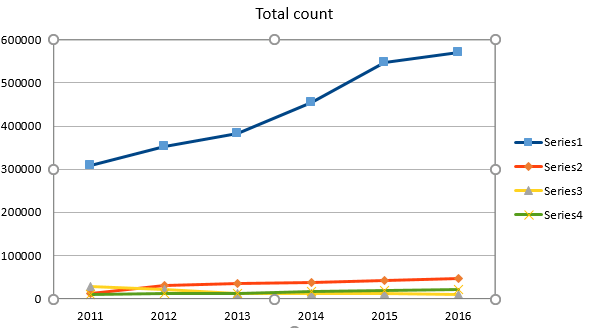
6) Find the percentage and the count of each case status on total applications for each year. Create a line graph depicting the pattern of All the cases over the period of time.

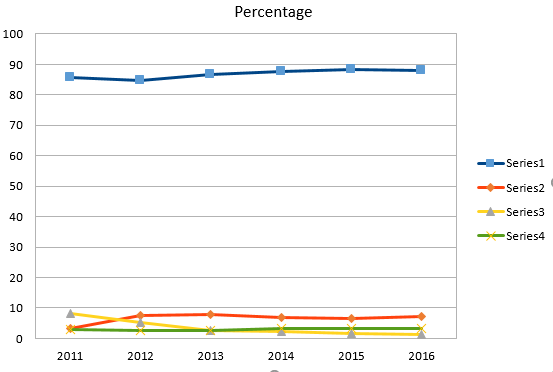
Analysed using Pig

|  |
| --- |
|  |
|  | |
|  | | h1b = load '/user/hive/warehouse/niit.db/h1b\_final/\*' using PigStorage('\t') as (sno:int, case\_status:chararray, emp\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_pos:chararray, wage:long, year:chararray, worksite:chararray, longitude:double, lattitude:double);  year\_grp = group h1b by $7; | |
|  | | total = foreach year\_grp generate group, (float)COUNT(h1b.$1) as tot; |
|  | | --dump total; |
|  | | grp = group h1b by ($7,$1); |
|  | | case\_total = foreach grp generate flatten(group), (float)COUNT(h1b.$1) as per; |
|  | | --dump case\_total; |
|  | | join\_grp = join total by $0,case\_total by $0; |
|  | | --dump join\_grp; |
|  | | final = foreach join\_grp generate $3,$0,$4,ROUND\_TO(($4/$1)\*100,2); |
|  | | --dump final; |
|  | | odby = order final by $0 asc, $1 asc; |
|  | | --dump odby; |
|  | | store odby into '/niit/project6'; |
|  | |  |

Output:





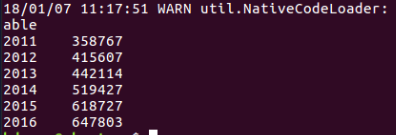


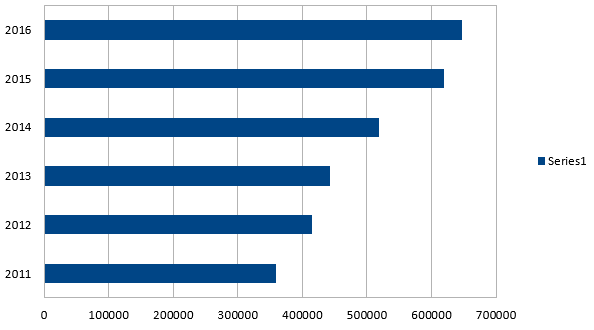
7) Create a bar graph to depict the number of applications for each year.

Analysed using Java MapReduce

|  |
| --- |
|  |
|  | import java.io.IOException;  import org.apache.hadoop.conf.Configuration; |
|  | import org.apache.hadoop.fs.Path; |
|  | import org.apache.hadoop.io.LongWritable; |
|  | import org.apache.hadoop.io.Text; |
|  | import org.apache.hadoop.mapreduce.Job; |
|  | import org.apache.hadoop.mapreduce.Mapper; |
|  | import org.apache.hadoop.mapreduce.Reducer; |
|  | import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; |
|  | import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; |
|  |  |
|  |  |
|  | public class YearApp |
|  | { |
|  | 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[7]), new Text(values)); |
|  | } |
|  | } |
|  |  |
|  | public static class ReduceClass extends Reducer<Text, Text, Text, LongWritable> |
|  | { |
|  | public void reduce(Text key, Iterable<Text> values, Context con) throws IOException, InterruptedException |
|  | { |
|  | long count=0; |
|  | for(Text val:values) |
|  | { |
|  | count++; |
|  | } |
|  | con.write(key, new LongWritable(count)); |
|  | } |
|  | } |
|  | public static void main(String[] args) throws Exception |
|  | { |
|  | Configuration conf = new Configuration(); |
|  | Job job = Job.getInstance(conf,"Year Application"); |
|  | job.setJarByClass(YearApp.class); |
|  | job.setMapperClass(MapClass.class); |
|  | job.setReducerClass(ReduceClass.class); |
|  | job.setMapOutputKeyClass(Text.class); |
|  | job.setMapOutputValueClass(Text.class); |
|  | job.setOutputKeyClass(Text.class); |
|  | job.setOutputValueClass(LongWritable.class); |
|  | job.setNumReduceTasks(1); |
|  | FileInputFormat.addInputPath(job, new Path(args[0])); |
|  | FileOutputFormat.setOutputPath(job, new Path(args[1])); |
|  | System.exit(job.waitForCompletion(true) ? 0 : 1); |
|  | } |
|  | } |

Output:





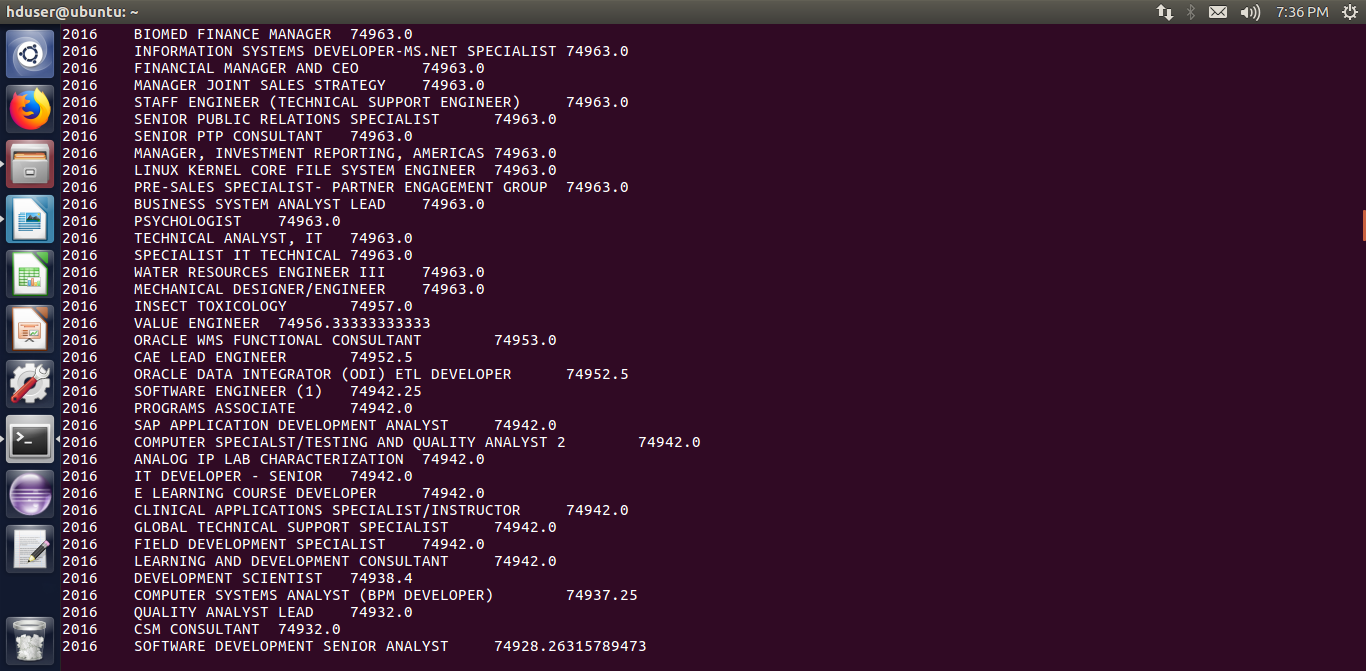
8) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order.

Analysed using Pig

For Full Time Jobs

|  |
| --- |
|  |
|  | h1b = load '/user/hive/warehouse/niit.db/h1b\_final/\*' using PigStorage('\t') as (sno:int, case\_status:chararray, emp\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_pos:chararray, wage:long, year:chararray, worksite:chararray, longitude:double, lattitude:double);  h1b = filter h1b by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN'; |
|  | h1b\_y = filter h1b by $5=='Y'; |
|  | y\_avg = group h1b\_y by ($7,$4); |
|  | final\_y = foreach y\_avg generate flatten(group), AVG(h1b\_y.$6); |
|  | --dump final; |
|  | f11 = filter final\_y by $0=='2011'; |
|  | f11 = order f11 by $2 desc; |
|  | f12 = filter final\_y by $0=='2012'; |
|  | f12 = order f12 by $2 desc; |
|  | f13 = filter final\_y by $0=='2013'; |
|  | f13 = order f13 by $2 desc; |
|  | f14 = filter final\_y by $0=='2014'; |
|  | f14 = order f14 by $2 desc; |
|  | f15 = filter final\_y by $0=='2015'; |
|  | f15 = order f15 by $2 desc; |
|  | f16 = filter final\_y by $0=='2016'; |
|  | f16 = order f16 by $2 desc; |
|  | fyes = union f11,f12,f13,f14,f15,f16; |
|  | store fyes into '/niit/project8y'; |

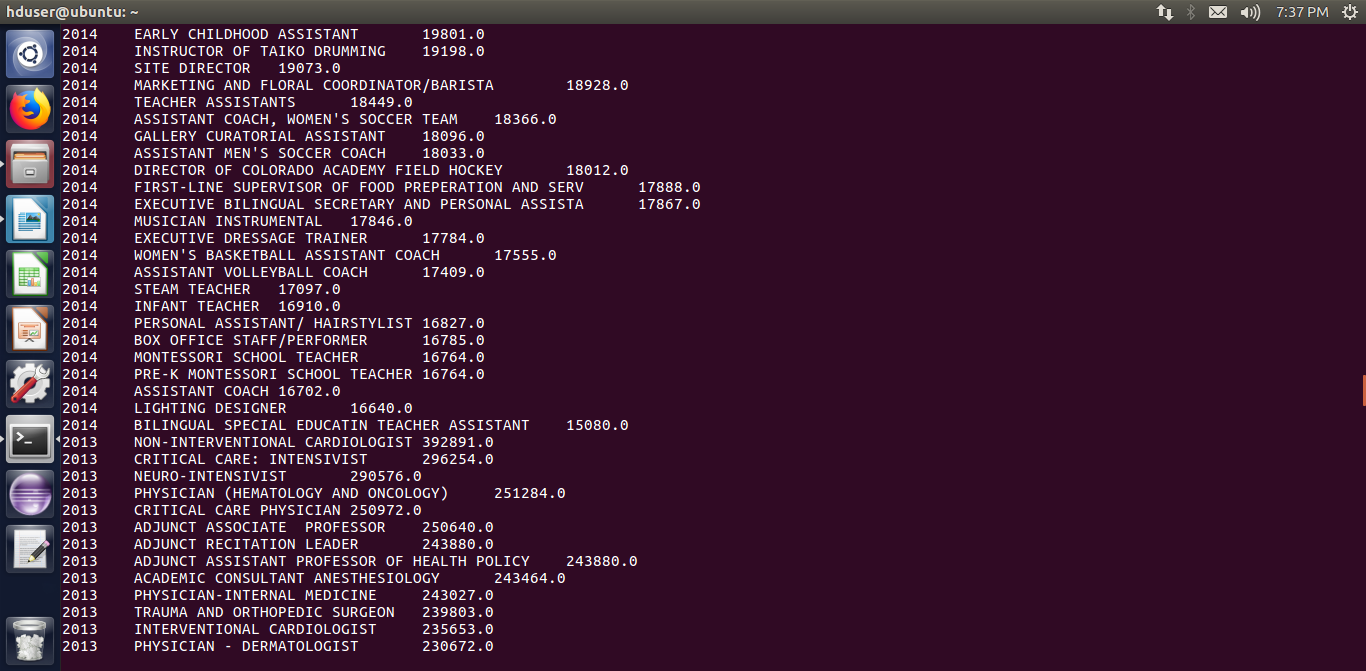
Output:



For Part Time Jobs

|  |
| --- |
|  |
|  | |
|  | | h1b = load '/user/hive/warehouse/niit.db/h1b\_final/\*' using PigStorage('\t') as (sno:int, case\_status:chararray, emp\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_pos:chararray, wage:long, year:chararray, worksite:chararray, longitude:double, lattitude:double);  h1b = filter h1b by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN'; |
|  | | h1b\_y = filter h1b by $5=='N'; |
|  | | y\_avg = group h1b\_y by ($7,$4); |
|  | | final\_y = foreach y\_avg generate flatten(group), AVG(h1b\_y.$6); |
|  | | --dump final; |
|  | | f11 = filter final\_y by $0=='2011'; |
|  | | f11 = order f11 by $2 desc; |
|  | | f12 = filter final\_y by $0=='2012'; |
|  | | f12 = order f12 by $2 desc; |
|  | | f13 = filter final\_y by $0=='2013'; |
|  | | f13 = order f13 by $2 desc; |
|  | | f14 = filter final\_y by $0=='2014'; |
|  | | f14 = order f14 by $2 desc; |
|  | | f15 = filter final\_y by $0=='2015'; |
|  | | f15 = order f15 by $2 desc; |
|  | | f16 = filter final\_y by $0=='2016'; |
|  | | f16 = order f16 by $2 desc; |
|  | | fyes = union f11,f12,f13,f14,f15,f16; |
|  | | store fyes into '/niit/project8n'; |
|  | |  |

Output:

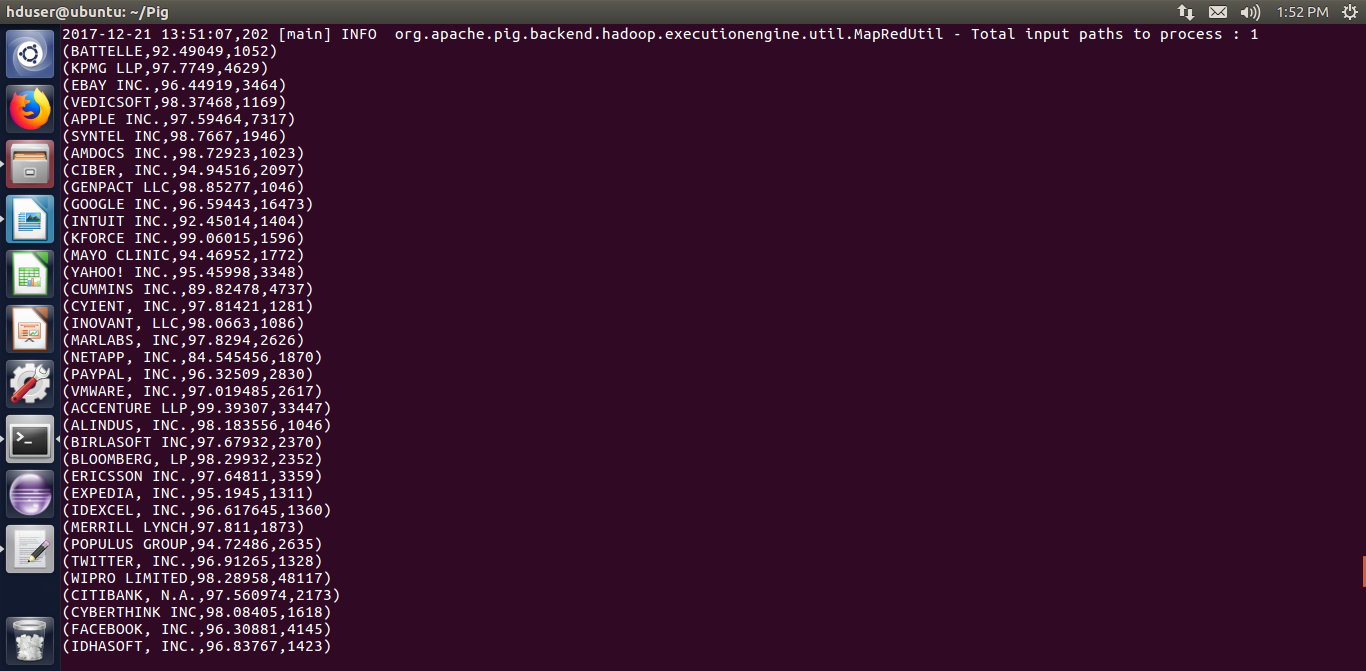


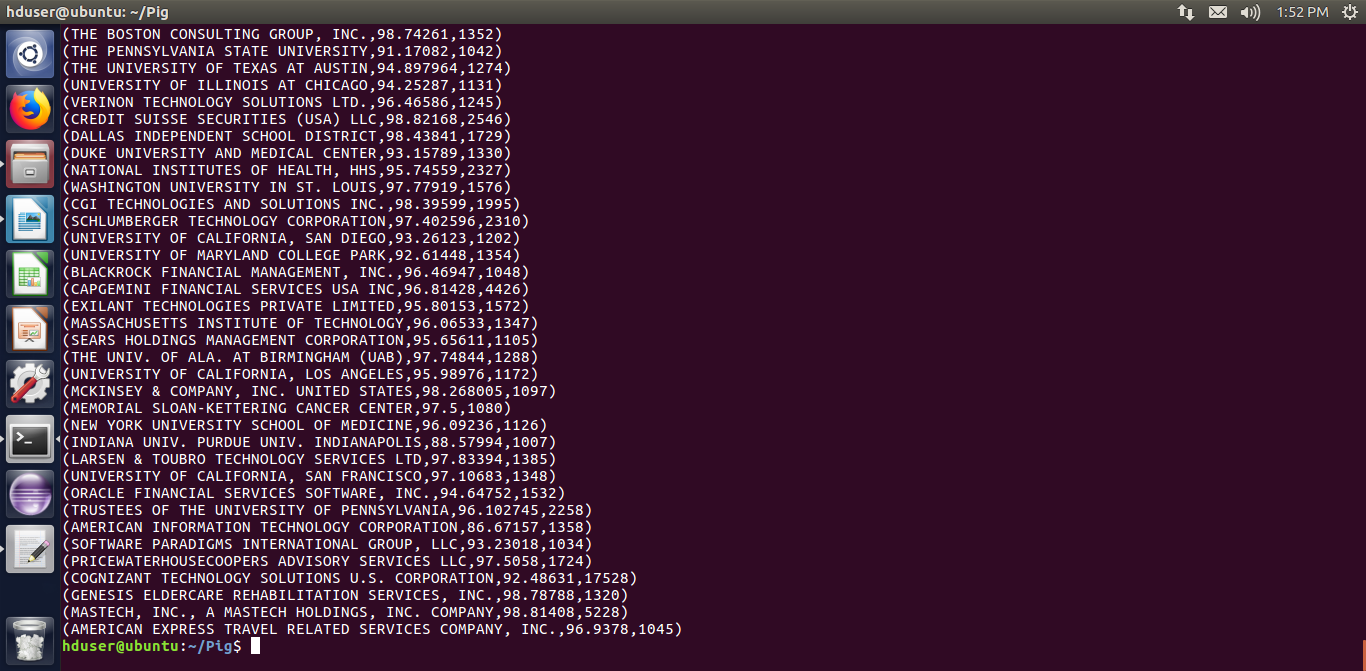
9) Which are the employers along with the number of petitions who have the success rate more than 70% in petitions. (total petitions filed 1000 OR more than 1000) ?

Analysed using Pig

|  |
| --- |
|  |
|  | h1b = load '/user/hive/warehouse/niit.db/h1b\_final/\*' using PigStorage('\t') as (sno:int, case\_status:chararray, emp\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_pos:chararray,wage:long, year:chararray, worksite:chararray, longitude:double, lattitude:double); |
|  | grp = group h1b by $2; |
|  | total = foreach grp generate group, COUNT(h1b.$1); |
|  | --dump total; |
|  |  |
|  | h1b\_filter = filter h1b by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN'; |
|  | grp = group h1b\_filter by $2; |
|  | certified = foreach grp generate group, COUNT(h1b\_filter.$1); |
|  | --dump case\_count; |
|  |  |
|  | --h1b\_filter = filter h1b by case\_status == 'CERTIFIED-WITHDRAWN'; |
|  | --grp = group h1b\_filter by $2; |
|  | --certified\_withdrawn = foreach grp generate group, COUNT(h1b\_filter.$1); |
|  |  |
|  | emp\_join = join certified by $0, total by $0; |
|  | emp\_join = foreach emp\_join generate $0,$1,$3; |
|  | success = foreach emp\_join generate $0,(float)$1/$2\*100, $2; |
|  | --dump emp\_join; |
|  | final = filter success by $1>70 and $2>=1000; |
|  | dump final; |
|  | --store final into '/niit/project9'; |

Output:



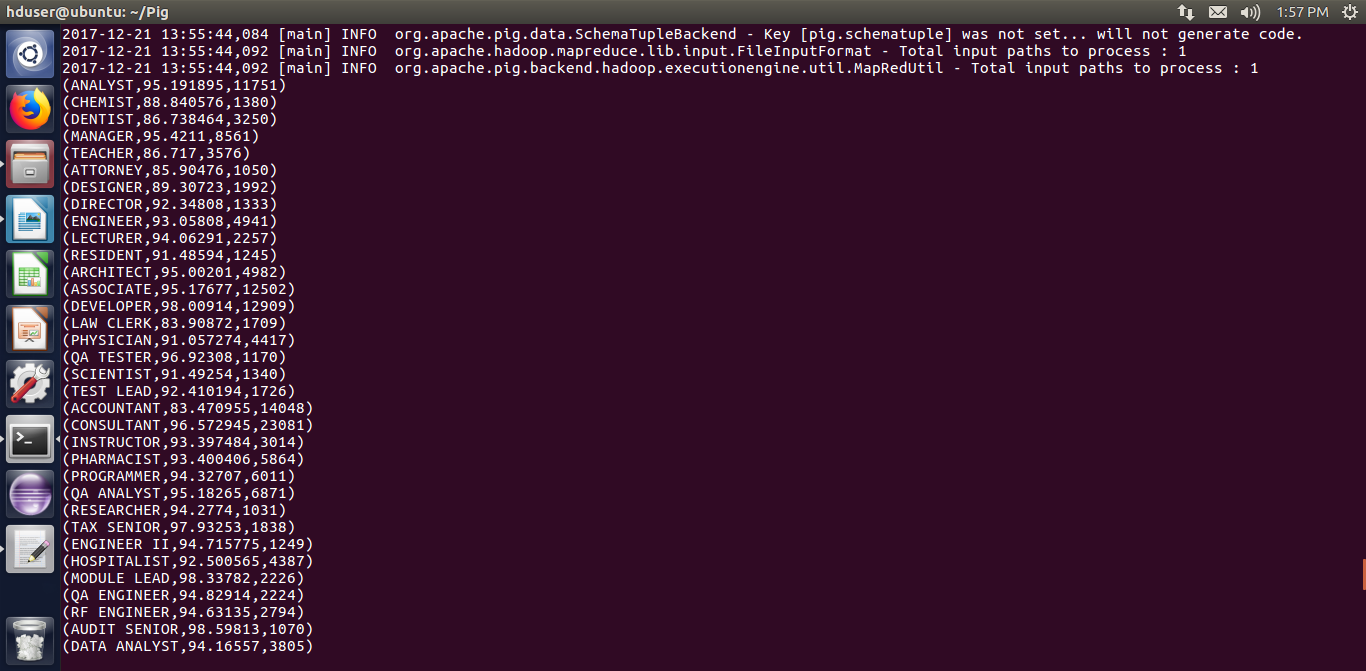


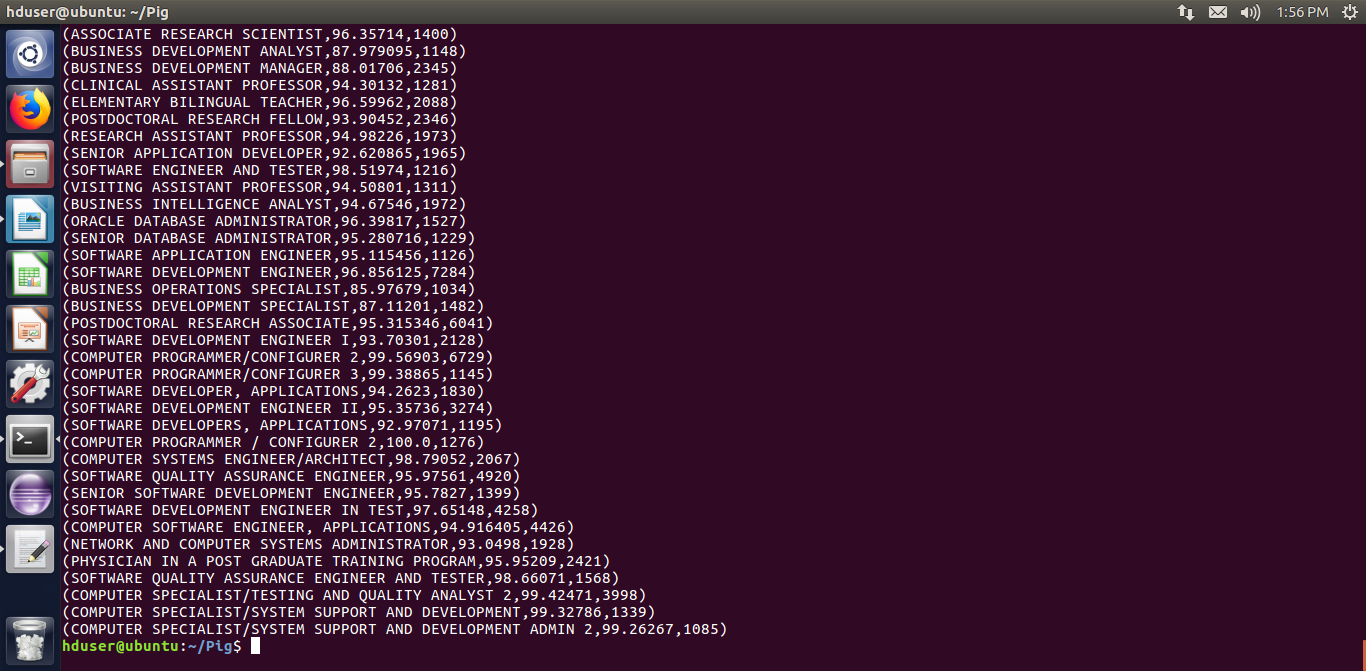
10) Which are the job positions along with the number of petitions which have the success rate more than 70% in petitions (total petitions filed 1000 OR more than 1000)?

Analysed using Pig

|  |
| --- |
|  |
|  | h1b = load '/user/hive/warehouse/niit.db/h1b\_final/\*' using PigStorage('\t') as (sno:int, case\_status:chararray, emp\_name:chararray, soc\_name:chararray, job\_title:chararray, full\_time\_pos:chararray,wage:long, year:chararray, worksite:chararray, longitude:double, lattitude:double); |
|  | grp = group h1b by $4; |
|  | total = foreach grp generate group, COUNT(h1b.$1); |
|  | --dump total; |
|  |  |
|  | h1b\_filter = filter h1b by case\_status == 'CERTIFIED' or case\_status == 'CERTIFIED-WITHDRAWN'; |
|  | grp = group h1b\_filter by $4; |
|  | certified = foreach grp generate group, COUNT(h1b\_filter.$1); |
|  | --dump case\_count; |
|  |  |
|  | --h1b\_filter = filter h1b by case\_status == 'CERTIFIED-WITHDRAWN'; |
|  | --grp = group h1b\_filter by $4; |
|  | --certified\_withdrawn = foreach grp generate group, COUNT(h1b\_filter.$1); |
|  |  |
|  | emp\_join = join certified by $0, total by $0; |
|  | emp\_join = foreach emp\_join generate $0,$1,$3; |
|  | success = foreach emp\_join generate $0,(float)$1/$2\*100, $2; |
|  | --dump emp\_join; |
|  | final = filter success by $1>70 and $2>=1000; |
|  | --dump final; |
|  | store final into '/niit/project10'; |

Output:





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

Performed by using Sqoop

Before exporting the data from HDFS to MySQL, a table must be created in MySQL.

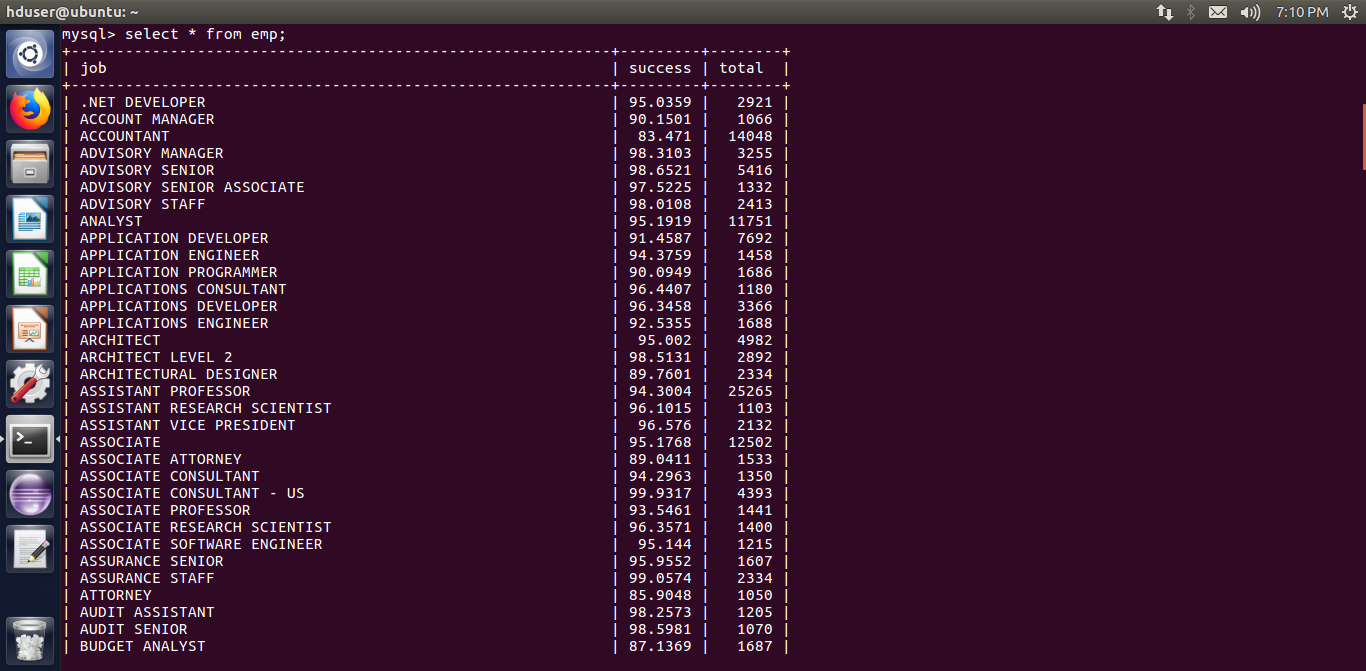
use project10;

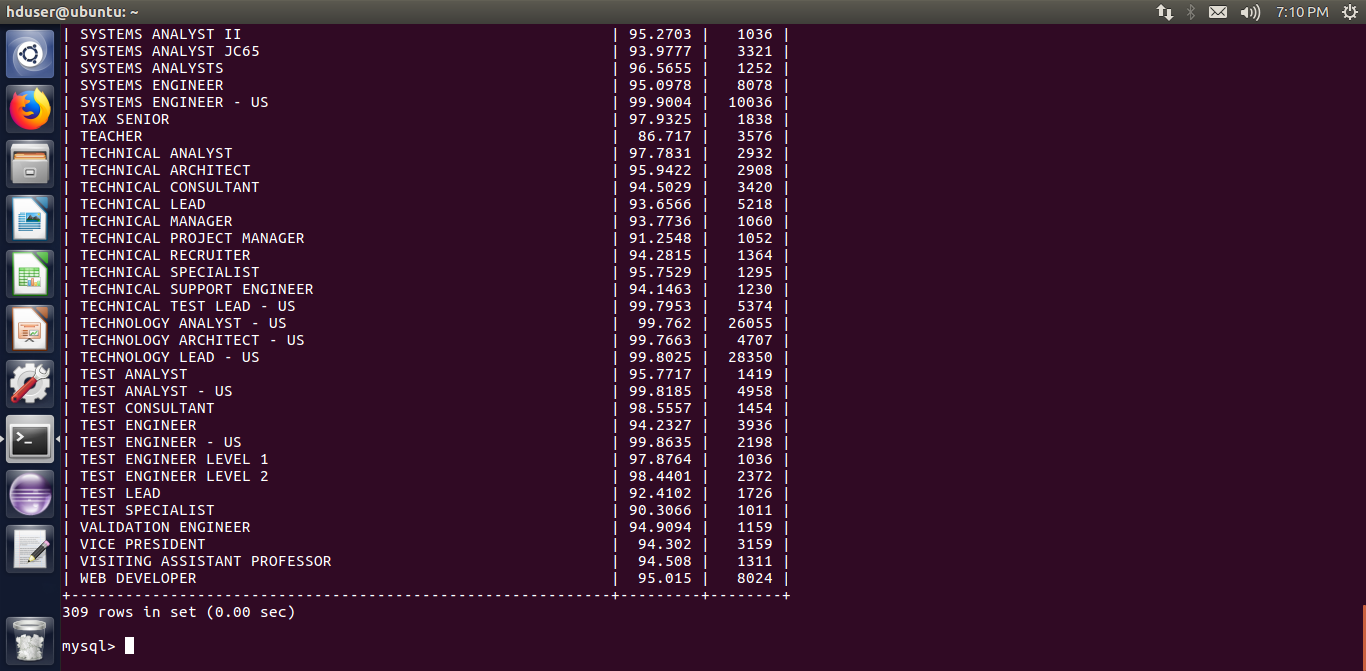
CREATE TABLE employee\_master( job VARCHAR(100) NOT NULL, percent FLOAT NOT NULL,

Total INT NOT NULL, PRIMARY KEY (job));

sqoop export --connect jdbc:mysql://localhost/project10 --username root --P --table emp --update-mode allowinsert --update-key job --export-dir /niit/project10/\* --input-fields-terminated-by '\t' ;

Output:





Project is a menu based one. Using shell script created a Menu based project

|  |
| --- |
|  |
|  | #!/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} Question 1a ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 2)${MENU} Question 1b ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 3)${MENU} Question 2a ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 4)${MENU} Question 2b ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 5)${MENU} Question 3 ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 6)${MENU} Question 4 ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 7)${MENU} Question 5a ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 8)${MENU} Question 5b ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 9)${MENU} Question 6 ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 10)${MENU} Question 7 ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 11)${MENU} Question 8 ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 12)${MENU} Question 9 ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 13)${MENU} Question 10 ${NORMAL}" |
|  | echo -e "${MENU}\*\*${NUMBER} 14)${MENU} Question 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/project/project1b.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/2a.sh |
|  | show\_menu; |
|  | ;; |
|  |  |
|  | 4) clear; |
|  | option\_picked "2b) find top 5 locations in the US who have got certified visa for each year"; |
|  | echo "Enter the year" |
|  | read year |
|  | echo "You've selected ${year}" |
|  | hive -e "select year, worksite, count(\*) as n from h1b\_final where year=$year and case\_status='CERTIFIED' group by year, worksite order by n desc limit 5;" |
|  | show\_menu; |
|  | ;; |
|  |  |
|  | 5) clear; |
|  | option\_picked "3)Which industry(SOC\_NAME) has the most number of Data Scientist positions?"; |
|  | hive -e "select soc\_name,job\_title,count(job\_title) as n from h1b\_final where job\_title='DATA SCIENTIST' and case\_status='CERTIFIED' group by soc\_name,job\_title order by n desc limit 1;" |
|  | 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"; |
|  | echo "Enter the year" |
|  | read year |
|  | echo "You've selected ${year}" |
|  | hive -e "select year, job\_title, count(job\_title) as n from h1b\_final where year=$year group by year, job\_title order by n desc limit 10;" |
|  | 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."; |
|  | echo "Enter the year" |
|  | read year |
|  | echo "You've selected ${year}" |
|  | hive -e "select year, job\_title, count(job\_title) as n from h1b\_final where year=$year and case\_status='CERTIFIED' group by year, job\_title order by n desc limit 10;" |
|  | 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/project/project6.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/project/project8y.pig |
|  | ;; |
|  |  |
|  | 2) echo "PART TIME JOB SELECTED" |
|  |  |
|  | pig /home/hduser/Proj/Proj8n.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/project/project9.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/project/project10.pig |
|  | show\_menu; |
|  | ;; |
|  |  |
|  | 14) clear; |
|  | option\_picked "11) Export result for question no 10 to MySql database."; |
|  | sqoop export --connect jdbc:mysql://localhost/project10 --username root --P --table emp --update-mode allowinsert --update-key job --export-dir /niit/project10/\* --input-fields-terminated-by '\t' ; |
|  | show\_menu; |
|  | ;; |
|  |  |
|  | \n) exit; |
|  | ;; |
|  |  |
|  | \*) clear; |
|  | option\_picked "Pick an option from the menu"; |
|  | show\_menu; |
|  | ;; |
|  | esac |
|  | fi |
|  | done |
|  |  |

hduser@ubuntu:~$ ./menu.sh

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*APP MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 1) Question 1a

\*\* 2) Question 1b

\*\* 3) Question 2a

\*\* 4) Question 2b

\*\* 5) Question 3

\*\* 6) Question 4

\*\* 7) Question 5a

\*\* 8) Question 5b

\*\* 9) Question 6

\*\* 10) Question 7

\*\* 11) Question 8

\*\* 12) Question 9

\*\* 13) Question 10

\*\* 14) Question 11

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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