**BIG DATA PROJECT**

**H1B DATA ANALYSIS USING HADOOP**

**PRESENTED BY**

**PRIYANKA.G**

**S171169300155**

**NIIT-THANE**

**TECH MENTOR**

**MS.HEBA NOMANI**

**What is big data?**

Big data is a term that describes the large volume of data – both structured and unstructured – that inundates a business on a day-to-day basis. It is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.

### Characteristics of 'Big Data'

### 

**(1)Volume –** The name 'Big Data' itself is related to a size which is enormous. Size of data plays very crucial role in determining value out of data. Also, whether a particular data can actually be considered as a Big Data or not, is dependent upon volume of data. Hence, **'Volume'** is one characteristic which needs to be considered while dealing with 'Big Data'.

**(2)Variety –** The next aspect of 'Big Data' is its **variety**.

Variety refers to heterogeneous sources and the nature of data, both structured and unstructured. During earlier days, spreadsheets and databases were the only sources of data considered by most of the applications. Now days, data in the form of emails, photos, videos, monitoring devices, PDFs, audio, etc. is also being considered in the analysis applications. This variety of unstructured data poses certain issues for storage, mining and analysing data.

**(3)Velocity –** The term **'velocity'** refers to the speed of generation of data. How fast the data is generated and processed to meet the demands, determines real potential in the data.

Big Data Velocity deals with the speed at which data flows in from sources like business processes, application logs, networks and social media sites, sensors, Mobile devices, etc. The flow of data is massive and continuous.

**(4)Variability –** This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively.

**(5)Veracity -** Veracity refers to the trustworthiness of the data. Can the manager rely on the fact that the data is representative? Every good manager knows that there are inherent discrepancies in all the data collected.

## **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.

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.

## **Benefits of Big Data**

Big data is really critical to our life and its emerging as one of the most important technologies in modern world. Follow are just few benefits which are very much known to all of us:

* Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums.
* Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.
* Using the data regarding the previous medical history of patients, hospitals are providing better and quick service.

## **Big Data Challenges**

The major challenges associated with big data are as follows:

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

To fulfill the above challenges, organizations normally take the help of enterprise servers.

**Hadoop**

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

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

## **Hadoop Architecture**

Hadoop framework includes following four modules:

* **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
* **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.
* **Hadoop Distributed File System (HDFS):** A distributed file system that provides high-throughput access to application data.
* **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

**Hadoop Distributed File System (HDFS)**

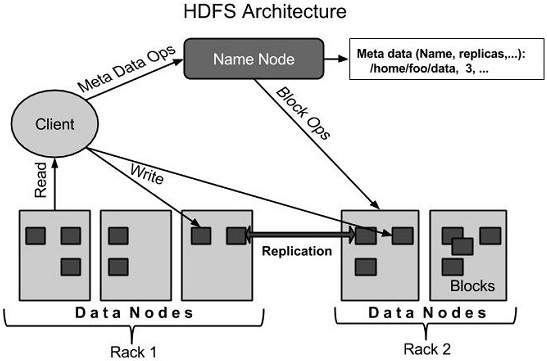
Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly faulttolerant and designed using low-cost hardware.

## **Features of HDFS**

* It is suitable for the distributed storage and processing.
* Hadoop provides a command interface to interact with HDFS.
* The built-in servers of namenode and datanode help users to easily check the status of cluster.
* Streaming access to file system data.
* HDFS provides file permissions and authentication.

## **HDFS Architecture**

Given below is the architecture of a Hadoop File System.

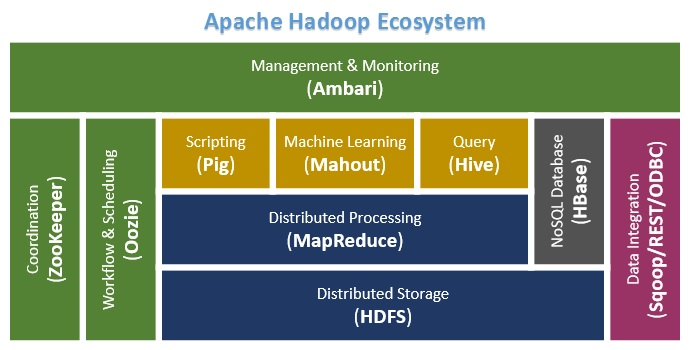


HDFS follows the master-slave architecture and it has the following elements.

## **Goals of HDFS**

* **Fault detection and recovery**: Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.
* **Huge datasets**: HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.
* **Hardware at data**: A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

**Apache Hadoop Ecosystem**



**Map Reduce**

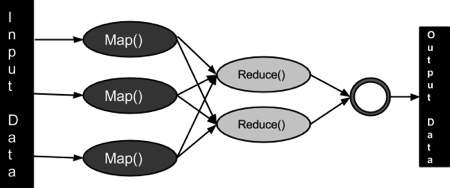
**MapReduce** is a [programming model](https://en.wikipedia.org/wiki/Programming_model) and an associated implementation for processing and generating [big data](https://en.wikipedia.org/wiki/Big_data) sets with a [parallel](https://en.wikipedia.org/wiki/Parallel_computing), [distributed](https://en.wikipedia.org/wiki/Distributed_computing) algorithm on a [cluster](https://en.wikipedia.org/wiki/Cluster_(computing)).

It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster.

The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

## **The Algorithm**

* Generally MapReduce paradigm is based on sending the computer to where the data resides!
* MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
  + **Map stage**: The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
  + **Reduce stage**: This stage is the combination of the **Shuffle**stage and the **Reduce** stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
* During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
* The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
* Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
* After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



**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 analysingeasy. **Apache Hive**, allows SQL developers to write Hive Query Language (HQL) statements that are similar to standard SQL statements; now you should be aware that HQL is limited in the commands it understands, but it is still pretty useful. HQL statements are broken down by the Hive service into hive jobs and executed across a Hadoop cluster.

**ADVANTAGES OF USING APACHE HIVE**

1. Fits the low level interface requirement of Hadoop perfectly.

2. Supports external tables which make it possible to process data without actually storing in HDFS.

3. It has a rule based optimizer for optimizing logical plans.

4. Supports partitioning of data at the level of tables to improve performance.

5. Metastore or Metadata store is a big plus in the architecture which makes the lookup easy.

**DISADVANTAGES OF USING APACHE HIVE**

1. No support for update and delete.

2. No support for singleton inserts. Data is required to be loaded from a file using LOAD command.

3. No access control implementation.

4. Correlated sub queries are not supported.

**Apache Pig**

**Apache Pig** is a platform for analysing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

**Advantages**

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

**Sqoop**

Apache Sqoop efficiently transfers bulk data between Apache Hadoop and structured data stores such as relational databases. Sqoop helps offload certain tasks (such as ETL processing) from the EDW to Hadoop for efficient execution at a much lower cost. Sqoop can also be used to extract data from Hadoop and export it into external structured data stores. It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases. This is a brief tutorial that explains how to make use of Sqoop in Hadoop ecosystem.

**Flume**

**Apache Flume** is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS).

Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log data, events (etc...) from various webserves to a centralized data store.

**Oozie**

Apache Oozie is a server-based [workflow](https://en.wikipedia.org/wiki/Workflow) [scheduling](https://en.wikipedia.org/wiki/Scheduling_(computing)) system to manage [Hadoop](https://en.wikipedia.org/wiki/Apache_Hadoop) jobs.

Workflows in Oozie are defined as a collection of control flow and action [nodes](https://en.wikipedia.org/wiki/Vertex_(graph_theory))in a [directed acyclic graph](https://en.wikipedia.org/wiki/Directed_acyclic_graph). Control flow nodes define the beginning and the end of a workflow (start, end, and failure nodes) as well as a mechanism to control the workflow execution path (decision, fork, and join nodes). Action nodes are the mechanism by which a workflow triggers the execution of a computation/processing task. Oozie provides support for different types of actions including Hadoop [MapReduce](https://en.wikipedia.org/wiki/MapReduce), Hadoop distributed file system operations, [Pig](https://en.wikipedia.org/wiki/Pig_(programming_tool)), [SSH](https://en.wikipedia.org/wiki/Secure_Shell), and [email](https://en.wikipedia.org/wiki/Email).

**Zookeeper**

Zookeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them, which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed.

**PROJECT DESCRIPTION**

**H1B\_Visa Analysis:**

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.. 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.

The goal of the project is get insights into **H-1B** applications over the year 2011 to 2016 by performing analysis using different technologies like MapReduce,Pig,Hive

Dataset column description.

1. **CASE\_STATUS:**Status associated with the last significant event or decision. Valid values include.
   * "Certified": Employer filed the LCA, which was approved by DOL
   * "Certified-Withdrawn": LCA was approved but later withdrawn by employer
   * "Denied": LCA was denied by DOL
   * "Withdrawn": LCA was withdrawn by employer before
2. **EMPLOYER\_NAME**: Name of employer submitting labour condition application.
3. **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.
4. **JOB\_TITLE**: Title of the job.FULL\_TIME\_POSITION:
   * **Y** = Full Time Position.
   * **N** = Part Time Position.
5. **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.
6. **YEAR:** Year in which the H1B visa petition was filed.
7. **WORKSITE:** City and State information of the foreign worker’s intended area of employment.
8. **lon:** longitude of the Worksite. 9. **lat:** latitude of the Worksite

**Project Outline**

|  |  |
| --- | --- |
| Title | Bigdata analysis in hadoop on H1b Visa Data |
| Input | H1b visa Data |
| Data Elements | sno,case\_status,employer\_name,soc\_name, job\_title,full\_time\_position,prevailing\_wage,year,worksite,longitude,latitude. |
| purpose | By performing analysis on the H1B visa applicants between the years 2011-2015. To identify the growth cycle is increasing or not, which industry has granted more visas etc. |
| Methodology | Agile |

PROJECT CODE

MAP REDUCE

Q1a] Is the number of petitions with Data Engineer job title increasing over time?

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.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.Reducer.Context;

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

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

public class Question1a {

public static class MapperEx extends Mapper<LongWritable,Text,Text,IntWritable>{

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

String Line = value.toString();

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

String Year = arr[7];

String Job\_title = arr[4];

int count = 0;

if(Job\_title.equals("DATA ENGINEER")){

count++;

con.write(new Text(Year),new IntWritable(count));

}

}

}

public static class ReducerEx extends Reducer<Text,IntWritable,Text,Text>{

int year = 0;

String output = " ";

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

int count = 0;

double growthpercentage= 0;

for (IntWritable val:values){

count += val.get();

}

if(year!=0){

growthpercentage = ((double)(count-year)/year)\*100;

String.format("%.2f%%", growthpercentage);

}

output = String.format("%d",count)+","+growthpercentage;

year = count;

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

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Question1a");

job.setJarByClass(Question1a.class);

job.setMapperClass(MapperEx.class);

job.setReducerClass(ReducerEx.class);

job.setOutputKeyClass(Text.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.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);

}

}

O/P: 2011 18,0.0

|  |  |
| --- | --- |
|  | 2012 32,77.77777777777779 |
|  | 2013 41,28.125 |
|  | 2014 89,117.07317073170731 |
|  | 2015 160,79.7752808988764 |
|  | 2016 251,56.875 |

Q2 a] which part of the US has the most Data Engineer jobs for each year?

|  |
| --- |
|  |
|  |
|  |  |
|  |  |
|  | 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 Question2a { |
|  |  |
|  |  |
|  | public static class MapperEx extends Mapper<LongWritable, Text, Text, Text> { |
|  | public void map(LongWritable key, Text value, Context context) |
|  | throws IOException, InterruptedException { |
|  | String[] arr = value.toString().split("\t"); |
|  | String year = (arr[7]); |
|  | String job\_title = arr[4]; |
|  | String worksite = arr[8]; |
|  | if (job\_title.contains("DATA ENGINEER")) { |
|  | context.write(new Text(year + '\t' + worksite), new Text(worksite)); |
|  | } |
|  |  |
|  | } |
|  | } |
|  |  |
|  | public static class PartitionerEx extends Partitioner<Text, Text> { |
|  |  |
|  | @Override |
|  | public int getPartition(Text key, Text value, int numReduceTasks) { |
|  |  |
|  | String[] rec = key.toString().split("\t"); |
|  | String year = rec[0]; |
|  | if (year.equals("2011")) { |
|  | return 0; |
|  | } else if (year.equals("2012")) { |
|  | return 1; |
|  | } else if (year.equals("2013")) { |
|  | return 2; |
|  | } else if (year.equals("2014")) { |
|  | return 3; |
|  | } else if (year.equals("2015")) { |
|  | return 4; |
|  | } else { |
|  | return 5; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | public static class ReducerEx extends Reducer<Text, Text, NullWritable, Text> { |
|  | private TreeMap<Long, Text> rep = new TreeMap<Long, Text>(); |
|  |  |
|  | public void reduce(Text key, Iterable<Text> values, Context context) |
|  | throws IOException, InterruptedException { |
|  |  |
|  | int count = 0; |
|  | for (Text val : values) { |
|  | count++; |
|  | } |
|  |  |
|  |  |
|  | String myValue = key.toString(); |
|  | myValue = myValue + ',' + count; |
|  |  |
|  | rep.put(new Long(count), new Text(myValue)); |
|  | if (rep.size() > 1) { |
|  | rep.remove(rep.firstKey()); |
|  | } |
|  | } |
|  |  |
|  | protected void cleanup(Context context) throws IOException, |
|  | InterruptedException { |
|  | for (Text t : rep.values()) { |
|  | context.write(NullWritable.get(), t); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | public static void main(String[] args) throws IllegalArgumentException, IOException, ClassNotFoundException, InterruptedException { |
|  |  |
|  | Configuration conf = new Configuration(); |
|  | Job job = Job.getInstance(conf, "Q2a"); |
|  | job.setJarByClass(Question2a.class); |
|  | job.setMapperClass(MapperEx.class); |
|  | job.setPartitionerClass(PartitionerEx.class); |
|  | job.setReducerClass(ReducerEx.class); |
|  | job.setNumReduceTasks(6); |
|  | job.setMapOutputKeyClass(Text.class); |
|  | job.setMapOutputValueClass(Text.class); |
|  | job.setOutputKeyClass(NullWritable.class); |
|  | job.setOutputValueClass(Text.class); |
|  | FileInputFormat.setInputPaths(job, new Path(args[0])); |
|  | FileOutputFormat.setOutputPath(job, new Path(args[1])); |
|  | System.exit(job.waitForCompletion(true) ? 0 : 1); |
|  |  |
|  | } |
|  |  |
|  | } |
|  |  |
|  |  |

O/P:

2011 SEATTLE, WASHINGTON,20

|  |
| --- |
|  |
|  |
|  | 2012 SEATTLE, WASHINGTON,30 |
|  | 2013 SEATTLE, WASHINGTON,46 |
|  | 2014 SEATTLE, WASHINGTON,45 |
|  | 2015 SEATTLE, WASHINGTON,61 |
|  | 2016 SEATTLE, WASHINGTON,128 |

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

|  |
| --- |
|  |
|  |  |
|  | import java.io.IOException; |
|  |  |
|  | import org.apache.hadoop.conf.Configuration; |
|  | import org.apache.hadoop.fs.Path; |
|  | import org.apache.hadoop.io.IntWritable; |
|  | 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.Mapper.Context; |
|  | import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; |
|  | import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; |
|  |  |
|  |  |
|  | public class Question3 { |
|  |  |
|  | public static class MapperEx extends Mapper<LongWritable,Text,Text,IntWritable> |
|  | { |
|  |  |
|  | public void map(LongWritable k,Text v, Context con) throws IOException, InterruptedException{ |
|  |  |
|  | String Line =v.toString(); |
|  | String[] arr =Line.split("\t"); |
|  |  |
|  | String SOC= arr[3]; |
|  | String Job\_Title= arr[4]; |
|  | int count=0; |
|  | if(Job\_Title.contains("DATA SCIENTIST")){ |
|  | count++; |
|  |  |
|  | con.write(new Text(SOC),new IntWritable(count)); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | } |
|  |  |
|  | public static class ReducerEx extends Reducer<Text,IntWritable,Text,IntWritable>{ |
|  | int max\_job =0; |
|  | String Industry=""; |
|  |  |
|  | public void reduce(Text key,Iterable<IntWritable> values, Context con) { |
|  |  |
|  | int count=0; |
|  |  |
|  | for(IntWritable val:values) |
|  | { |
|  | count+=val.get(); |
|  |  |
|  | if (max\_job < count){ |
|  | max\_job=count; |
|  |  |
|  | Industry=key.toString(); |
|  | } |
|  |  |
|  | } |
|  | } |
|  |  |
|  | public void cleanup(Context con) throws IOException, InterruptedException{ |
|  |  |
|  | con.write(new Text(Industry),new IntWritable(max\_job)); |
|  | } |
|  |  |
|  | } |
|  |  |
|  | public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException { |
|  |  |
|  | Configuration conf = new Configuration(); |
|  | Job job = Job.getInstance(conf, "Q3"); |
|  | job.setJarByClass(Question1a.class); |
|  | job.setMapperClass(MapperEx.class); |
|  | job.setReducerClass(ReducerEx.class); |
|  |  |
|  | job.setOutputKeyClass(Text.class); |
|  | job.setOutputValueClass(IntWritable.class); |
|  |  |
|  | /\*job.setMapOutputKeyClass(Text.class); |
|  | job.setMapOutputValueClass(IntWritable.class); |
|  | job.setOutputKeyClass(NullWritable.class); |
|  |  |
|  | job.setOutputValueClass(Text.class);\*/ |
|  |  |
|  | FileInputFormat.addInputPath(job, new Path(args[0])); |
|  | FileOutputFormat.setOutputPath(job, new Path(args[1])); |
|  | System.exit(job.waitForCompletion(true) ? 0 : 1); |
|  |  |
|  |  |
|  |  |
|  |  |
|  | } |
|  |  |
|  | } |

O/P: STATISTICIANS 649

Q7] Create a bar graph to depict the number of applications for each year

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

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 Question7 {

public static class map\_app extends Mapper<LongWritable,Text,Text,Text>

{

public void map(LongWritable key,Text Value,Context con) throws IOException, InterruptedException

{

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

String year=rec[7];

String application=rec[1];

con.write(new Text(year),new Text(application));

}

}

public static class red extends Reducer<Text,Text,Text,IntWritable>

{

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

{

int count=0;

for(Text val:value)

{

count++;

}

con.write(key, new IntWritable(count));

}

}

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

Configuration con=new Configuration();

Job job=Job.getInstance(con,"Q7");

job.setJarByClass(Question7.class);

job.setMapperClass(map\_app.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setReducerClass(red.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

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

}

}

O/P:

2011 358767

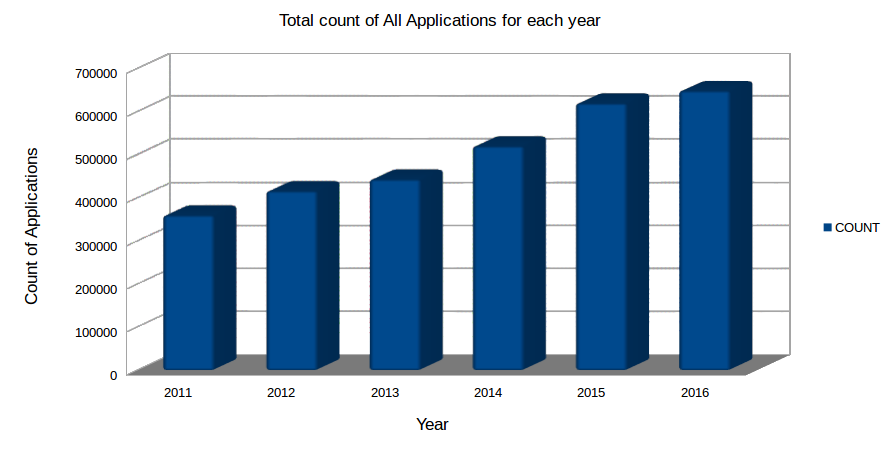
2012 415607

2013 442114

2014 519427

2015 618727

2016 647803



HIVE

 Q2 b] Find top 5 locations in the US who have got certified visa for each year.

select year,worksite,count(case\_status) as Count\_Cases from h1b\_final where case\_status='CERTIFIED' and year='2011' group by worksite,year order by Count\_Cases desc limit 5;

select year,worksite,count(case\_status) as Count\_Cases from h1b\_final where case\_status='CERTIFIED' and year='2012' group by worksite,year order by Count\_Cases desc limit 5;

select year,worksite,count(case\_status) as Count\_Cases from h1b\_final where case\_status='CERTIFIED' and year='2013' group by worksite,year order by Count\_Cases desc limit 5;

select year,worksite,count(case\_status) as Count\_Cases from h1b\_final where case\_status='CERTIFIED' and year='2014' group by worksite,year order by Count\_Cases desc limit 5;

select year,worksite,count(case\_status) as Count\_Cases from h1b\_final where case\_status='CERTIFIED' and year='2015' group by worksite,year order by Count\_Cases desc limit 5;

select year,worksite,count(case\_status) as Count\_Cases from h1b\_final where case\_status='CERTIFIED' and year='2016' group by worksite,year order by Count\_Cases desc limit 5;

O/P:

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

Q4] Which top 5 employers file the most petitions each year? - Case Status – ALL

select h1b\_final.year,h1b\_final.employer\_name,count(h1b\_final.employer\_name)as Total\_Applications from h1b\_final where h1b\_final.year='2011' group by h1b\_final.year, h1b\_final.employer\_name order by Total\_Applications desc limit 5;

select h1b\_final.year,h1b\_final.employer\_name,count(h1b\_final.employer\_name)as Total\_Applications from h1b\_final where h1b\_final.year='2012' group by h1b\_final.year, h1b\_final.employer\_name order by Total\_Applications desc limit 5;

select h1b\_final.year,h1b\_final.employer\_name,count(h1b\_final.employer\_name)as Total\_Applications from h1b\_final where h1b\_final.year='2013' group by h1b\_final.year, h1b\_final.employer\_name order by Total\_Applications desc limit 5;

select h1b\_final.year,h1b\_final.employer\_name,count(h1b\_final.employer\_name)as Total\_Applications from h1b\_final where h1b\_final.year='2014' group by h1b\_final.year, h1b\_final.employer\_name order by Total\_Applications desc limit 5;

select h1b\_final.year,h1b\_final.employer\_name,count(h1b\_final.employer\_name)as Total\_Applications from h1b\_final where h1b\_final.year='2015' group by h1b\_final.year, h1b\_final.employer\_name order by Total\_Applications desc limit 5;

select h1b\_final.year,h1b\_final.employer\_name,count(h1b\_final.employer\_name)as Total\_Applications from h1b\_final where h1b\_final.year='2016' group by h1b\_final.year, h1b\_final.employer\_name order by Total\_Applications desc limit 5;

O/P:

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

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

Q5]Find the most popular top 10 job positions for H1B visa applications for each year?

* 1. For all the applications

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

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

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

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

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

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

O/P:

2011 PROGRAMMER ANALYST 31799

2011 SOFTWARE ENGINEER 12763

2011 COMPUTER PROGRAMMER 8998

2011 SYSTEMS ANALYST 8644

2011 BUSINESS ANALYST 3891

2011 COMPUTER SYSTEMS ANALYST 3698

2011 ASSISTANT PROFESSOR 3467

2011 PHYSICAL THERAPIST 3377

2011 SENIOR SOFTWARE ENGINEER 2935

2011 SENIOR CONSULTANT 2798

2012 PROGRAMMER ANALYST 33066

2012 SOFTWARE ENGINEER 14437

2012 COMPUTER PROGRAMMER 9629

2012 SYSTEMS ANALYST 9296

2012 BUSINESS ANALYST 4752

2012 COMPUTER SYSTEMS ANALYST 4706

2012 SOFTWARE DEVELOPER 3895

2012 PHYSICAL THERAPIST 3871

2012 ASSISTANT PROFESSOR 3801

2012 SENIOR CONSULTANT 3737

2013 PROGRAMMER ANALYST 33880

2013 SOFTWARE ENGINEER 15680

2013 COMPUTER PROGRAMMER 11271

2013 SYSTEMS ANALYST 8714

2013 TECHNOLOGY LEAD - US 7853

2013 TECHNOLOGY ANALYST - US 7683

2013 BUSINESS ANALYST 5716

2013 COMPUTER SYSTEMS ANALYST 5043

2013 SOFTWARE DEVELOPER 5026

2013 SENIOR CONSULTANT 4326

2014 PROGRAMMER ANALYST 43114

2014 SOFTWARE ENGINEER 20500

2014 COMPUTER PROGRAMMER 14950

2014 SYSTEMS ANALYST 10194

2014 SOFTWARE DEVELOPER 7337

2014 BUSINESS ANALYST 7302

2014 COMPUTER SYSTEMS ANALYST 6821

2014 TECHNOLOGY LEAD - US 5057

2014 TECHNOLOGY ANALYST - US 4913

2014 SENIOR CONSULTANT 4898

2015 PROGRAMMER ANALYST 53436

2015 SOFTWARE ENGINEER 27259

2015 COMPUTER PROGRAMMER 14054

2015 SYSTEMS ANALYST 12803

2015 SOFTWARE DEVELOPER 10441

2015 BUSINESS ANALYST 8853

2015 TECHNOLOGY LEAD - US 8242

2015 COMPUTER SYSTEMS ANALYST 7918

2015 TECHNOLOGY ANALYST - US 7014

2015 SENIOR SOFTWARE ENGINEER 6013

2016 PROGRAMMER ANALYST 53743

2016 SOFTWARE ENGINEER 30668

2016 SOFTWARE DEVELOPER 14041

2016 SYSTEMS ANALYST 12314

2016 COMPUTER PROGRAMMER 11668

2016 BUSINESS ANALYST 9167

2016 COMPUTER SYSTEMS ANALYST 6900

2016 SENIOR SOFTWARE ENGINEER 6439

2016 DEVELOPER 6084

2016 TECHNOLOGY LEAD - US 5410

* 1. For only certified applications.

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

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

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

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

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

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

O/P:

2011 PROGRAMMER ANALYST 28806

2011 SOFTWARE ENGINEER 11224

2011 COMPUTER PROGRAMMER 8038

2011 SYSTEMS ANALYST 7850

2011 BUSINESS ANALYST 3444

2011 COMPUTER SYSTEMS ANALYST 3152

2011 ASSISTANT PROFESSOR 3050

2011 PHYSICAL THERAPIST 2911

2011 SENIOR SOFTWARE ENGINEER 2595

2011 SENIOR CONSULTANT 2585

2012 PROGRAMMER ANALYST 29226

2012 SOFTWARE ENGINEER 12273

2012 COMPUTER PROGRAMMER 8483

2012 SYSTEMS ANALYST 8399

2012 BUSINESS ANALYST 4144

2012 COMPUTER SYSTEMS ANALYST 4084

2012 SENIOR CONSULTANT 3420

2012 SOFTWARE DEVELOPER 3290

2012 PHYSICAL THERAPIST 3284

2012 ASSISTANT PROFESSOR 3033

2013 PROGRAMMER ANALYST 29906

2013 SOFTWARE ENGINEER 12973

2013 COMPUTER PROGRAMMER 10202

2013 SYSTEMS ANALYST 7850

2013 TECHNOLOGY LEAD - US 7809

2013 TECHNOLOGY ANALYST - US 7641

2013 BUSINESS ANALYST 4993

2013 COMPUTER SYSTEMS ANALYST 4554

2013 SOFTWARE DEVELOPER 4316

2013 SENIOR CONSULTANT 3996

2014 PROGRAMMER ANALYST 38625

2014 SOFTWARE ENGINEER 17278

2014 COMPUTER PROGRAMMER 13796

2014 SYSTEMS ANALYST 9161

2014 BUSINESS ANALYST 6529

2014 SOFTWARE DEVELOPER 6473

2014 COMPUTER SYSTEMS ANALYST 6204

2014 TECHNOLOGY LEAD - US 5055

2014 TECHNOLOGY ANALYST - US 4911

2014 SENIOR CONSULTANT 4535

2015 PROGRAMMER ANALYST 48203

2015 SOFTWARE ENGINEER 23352

2015 COMPUTER PROGRAMMER 12971

2015 SYSTEMS ANALYST 11498

2015 SOFTWARE DEVELOPER 9343

2015 TECHNOLOGY LEAD - US 8238

2015 BUSINESS ANALYST 7919

2015 COMPUTER SYSTEMS ANALYST 7234

2015 TECHNOLOGY ANALYST - US 7009

2015 SENIOR SOFTWARE ENGINEER 5324

2016 PROGRAMMER ANALYST 47964

2016 SOFTWARE ENGINEER 25890

2016 SOFTWARE DEVELOPER 12474

2016 SYSTEMS ANALYST 10986

2016 COMPUTER PROGRAMMER 10528

2016 BUSINESS ANALYST 8175

2016 COMPUTER SYSTEMS ANALYST 6205

2016 DEVELOPER 5912

2016 SENIOR SOFTWARE ENGINEER 5630

2016 TECHNOLOGY LEAD - US 5405

Q8]find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order.

FULL-TIME

--Full Time Position

insert overwrite directory '/user/hive/warehouse/h1b\_8/8a\_2011' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='Y' and h1b\_final.year='2011' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8a\_2012' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='Y' and h1b\_final.year='2012' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8a\_2013' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='Y' and h1b\_final.year='2013' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8a\_2014' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='Y' and h1b\_final.year='2014' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8a\_2015' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='Y' and h1b\_final.year='2015' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8a\_2016' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='Y' and h1b\_final.year='2016' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

O/P:

PART-TIME

--Part Time Position

insert overwrite directory '/user/hive/warehouse/h1b\_8/8b\_2011' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='N' and h1b\_final.year='2011' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8b\_2012' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='N' and h1b\_final.year='2012' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8b\_2013' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='N' and h1b\_final.year='2013' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8b\_2014' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='N' and h1b\_final.year='2014' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8b\_2015' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='N' and h1b\_final.year='2015' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

insert overwrite directory '/user/hive/warehouse/h1b\_8/8b\_2016' select h1b\_final.year,h1b\_final.job\_title,avg(h1b\_final.prevailing\_wage)as AVG from h1b\_final where h1b\_final.full\_time\_position='N' and h1b\_final.year='2016' group by h1b\_final.year,h1b\_final.job\_title order by AVG desc;

PIG

Q1b] Find top 5 job titles who are having highest avg growth in applications.

h1b = load '/user/hive/warehouse/h1b\_final' using PigStorage() as (s\_n\_no:int,case\_status:chararray,emp\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time:chararray,prevailing\_wage:int,year:chararray,worksite:chararray,longitude:int,latitude:int);

a1 = filter h1b by $7=='2011';

b1 = group a1 by $4;

c1 = foreach b1 generate group,COUNT($1);

a2 = filter h1b by $7=='2012';

b2 = group a2 by $4;

c2 = foreach b2 generate group,COUNT($1);

a3 = filter h1b by $7=='2013';

b3 = group a3 by $4;

c3 = foreach b3 generate group,COUNT($1);

a4 = filter h1b by $7=='2014';

b4 = group a4 by $4;

c4 = foreach b4 generate group,COUNT($1);

a5 = filter h1b by $7=='2015';

b5 = group a5 by $4;

c5 = foreach b5 generate group,COUNT($1);

a6 = filter h1b by $7=='2016';

b6 = group a6 by $4;

c6 = foreach b6 generate group,COUNT($1);

combine = join c1 by $0, c2 by $0, c3 by $0, c4 by $0, c5 by $0, c6 by $0;

job\_year\_count = foreach combine generate $0,$1,$3,$5,$7,$9,$11;

growth\_percent = foreach job\_year\_count generate $0, (float)(($6-$5)\*100)/$5,(float)(($5-$4)\*100)/$4, (float)(($4-$3)\*100)/$3, (float)(($3-$2)\*100)/$2, (float)(($2-$1)\*100)/$1;

avg\_growth\_percent = foreach growth\_percent generate $0,(($1+$2+$3+$4+$5)/5);

order\_agp = order avg\_growth\_percent by $1 desc;

top5 = limit order\_agp 5;

dump top5;

O/P:

(SENIOR SYSTEMS ANALYST JC60,4255.4644)

(SOFTWARE DEVELOPER 2,3480.5925)

(PROJECT MANAGER 3,3233.3335)

(SYSTEMS ANALYST JC65,2984.8809)

(MODULE LEAD,2917.112)

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.

h1b = load '/user/hive/warehouse/h1b\_final' using PigStorage() as (s\_n\_no:int,case\_status:chararray,emp\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time:chararray,prevailing\_wage:int,year:chararray,worksite:chararray,longitude:int,latitude:int);

a = foreach h1b generate $7,$1;

b = group a by year;

c = foreach b generate group as year, COUNT(a) as Total;

d= filter a by case\_status == 'CERTIFIED';

e= group d by year;

f = foreach e generate group as year,COUNT(d) as Certified;

g = join c by $0, f by $0;

h = foreach g generate $0,$1,$3,((float)$3\*100/(float)$1) as Percentage;

dump h;

O/P:

(2011,358767,307936,85.83175)

(2012,415607,352668,84.856125)

(2013,442114,382951,86.61816)

(2014,519427,455144,87.624245)

(2015,618727,547278,88.452255)

(2016,647803,569646,87.93507)

h1b = load '/user/hive/warehouse/h1b\_final' using PigStorage() as (s\_n\_no:int,case\_status:chararray,emp\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time:chararray,prevailing\_wage:int,year:chararray,worksite:chararray,longitude:int,latitude:int);

a = foreach h1b generate $7,$1;

b = group a by year;

c = foreach b generate group as year, COUNT(a) as Total;

d= filter a by case\_status == 'DENIED';

e= group d by year;

f = foreach e generate group as year,COUNT(d) as Certified;

g = join c by $0, f by $0;

h = foreach g generate $0,$1,$3,((float)$3\*100/(float)$1) as Percentage;

dump h;

O/P:

(2011,358767,29130,8.119476)

(2012,415607,21096,5.075949)

(2013,442114,12141,2.7461243)

(2014,519427,11899,2.2907934)

(2015,618727,10923,1.765399)

(2016,647803,9175,1.4163257)

h1b = load '/user/hive/warehouse/h1b\_final' using PigStorage() as (s\_n\_no:int,case\_status:chararray,emp\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time:chararray,prevailing\_wage:int,year:chararray,worksite:chararray,longitude:int,latitude:int);

a = foreach h1b generate $7,$1;

b = group a by year;

c = foreach b generate group as year, COUNT(a) as Total;

d= filter a by case\_status == 'CERTIFIED-WITHDRAWN';

e= group d by year;

f = foreach e generate group as year,COUNT(d) as Certified;

g = join c by $0, f by $0;

h = foreach g generate $0,$1,$3,((float)$3\*100/(float)$1) as Percentage;

dump h;

O/P:

(2011,358767,11596,3.2321813)

(2012,415607,31118,7.487362)

(2013,442114,35432,8.014222)

(2014,519427,36350,6.998096)

(2015,618727,41071,6.6379843)

(2016,647803,47092,7.269494)

h1b = load '/user/hive/warehouse/h1b\_final' using PigStorage() as (s\_n\_no:int,case\_status:chararray,emp\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time:chararray,prevailing\_wage:int,year:chararray,worksite:chararray,longitude:int,latitude:int);

a = foreach h1b generate $7,$1;

b = group a by year;

c = foreach b generate group as year, COUNT(a) as Total;

d= filter a by case\_status == 'WITHDRAWN';

e= group d by year;

f = foreach e generate group as year,COUNT(d) as Certified;

g = join c by $0, f by $0;

h = foreach g generate $0,$1,$3,((float)$3\*100/(float)$1) as Percentage;

dump h;

(2011,358767,10105,2.8165913)

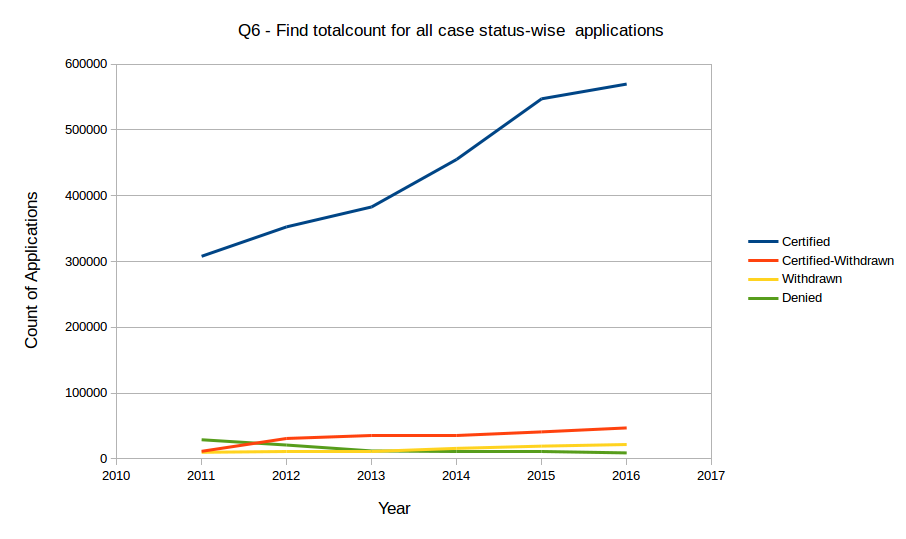
(2012,415607,10725,2.5805628)

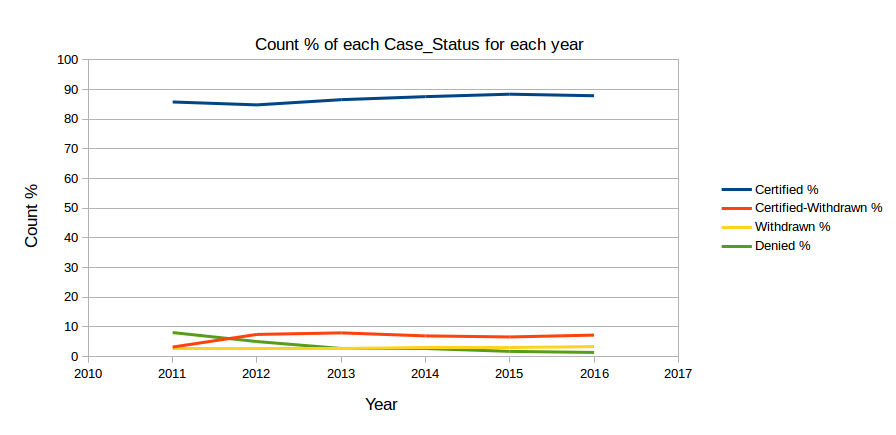
(2013,442114,11590,2.6214957)

(2014,519427,16034,3.086863)

(2015,618727,19455,3.1443594)

(2016,647803,21890,3.3791137)





Q9) Which are the employers along with the number of petitions who have the success rate more than 70% in petitions. (Total petitions filed more than 1000)?

h1b = load '/user/hive/warehouse/h1b\_final' using PigStorage() as (s\_n\_no:int,case\_status:chararray,emp\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time:chararray,prevailing\_wage:int,year:chararray,worksite:chararray,longitude:int,latitude:int);

a = foreach h1b generate emp\_name,case\_status;

b = filter a by case\_status == 'CERTIFIED-WITHDRAWN';

c = filter a by case\_status == 'CERTIFIED';

d = group a by emp\_name;

e = group c by emp\_name;

f = group b by emp\_name;

g = foreach d generate group as Employer, COUNT(a) as cnt;

h = foreach e generate group as Employer, COUNT(c) as cnt;

i = foreach f generate group as Employer, COUNT(b) as cnt;

j = join g by $0,h by $0,i by $0;

k = foreach j generate $0,$1,($3+$5);

l = foreach k generate $0,$1,((float)$2\*100/(float)$1)as l;

m = filter l by $1>1000 and $2>70.0;

n = order m by $2 desc;

dump n;

O/P:

(INFOSYS LIMITED,130592,99.54055)

(ACCENTURE LLP,33447,99.39307)

(TATA CONSULTANCY SERVICES LIMITED,64726,99.337204)

(HCL AMERICA, INC.,22678,99.26801)

(RELIABLE SOFTWARE RESOURCES, INC.,1992,99.14658)

(NTT DATA, INC.,4611,99.13251)

(ERP ANALYSTS, INC.,1785,99.10364)

(PATNI AMERICAS INC.,3149,99.07907)

(KFORCE INC.,1596,99.06015)

(GENPACT LLC,1046,98.852776)

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

h1b = load '/user/hive/warehouse/h1b\_final' using PigStorage('\t') as (s\_n\_no:int,case\_status:chararray,emp\_name:chararray,soc\_name:chararray,job\_title:chararray,full\_time:chararray,prevailing\_wage:int,year:chararray,worksite:chararray,longitude:int,latitude:int);

a = foreach h1b generate job\_title,case\_status;

b = filter a by case\_status == 'CERTIFIED-WITHDRAWN';

c = filter a by case\_status == 'CERTIFIED';

d = group a by job\_title;

e = group c by job\_title;

f = group b by job\_title;

g = foreach d generate group as Job\_Position, COUNT(a) as cnt;

h = foreach e generate group as Job\_Position, COUNT(c) as cnt;

i = foreach f generate group as Job\_Position, COUNT(b) as cnt;

j = join g by $0,h by $0,i by $0;

k = foreach j generate $0,$1,($3+$5);

l = foreach k generate $0,$1,((float)$2\*100/(float)$1)as l;

m = filter l by $1>1000 and $2>70.0;

n = order m by $2 desc;

dump n;

O/P:

(COMPUTER PROGRAMMER / CONFIGURER 2,1276,100.0)

(ASSOCIATE CONSULTANT - US,4393,99.93171)

(SYSTEMS ENGINEER - US,10036,99.90036)

(TEST ANALYST - US,4958,99.818474)

(CONSULTANT - US,7426,99.81147)

(TECHNOLOGY LEAD - US,28350,99.80247)

(TECHNICAL TEST LEAD - US,5374,99.79531)

(TECHNOLOGY ARCHITECT - US,4707,99.766304)

(TECHNOLOGY ANALYST - US,26055,99.76204)

(SENIOR PROJECT MANAGER - US,2774,99.74766)

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

>sqoop export --connect jdbc:mysql://localhost/H1B --username root --password 'hansolo' --table Question10 --update-mode allowinsert --export-dir /niit/q10 --input-fields-terminated-by ',' ;

**CONCLUSION:**

After dealing with the above technologies of Hadoop like Map Reduce, Pig and Hive, the following conclusions were made :

* Out of all the 3 technologies used it was observed that Hive provided with the o/p in lesser time as compared to Pig and MapReduce
* Given the Strict immigration laws of the US study of such a project sure does give an insight into the H1B visa application process
* MapReduce code in java makes the complex analysis quite easy. Code Required to be written to collect user input and performing complex join operations are handled efficiently by using this approach.
* Hive helps in the cleaning up of the data. csv data can be easily converted into text format using hive.
* For normal group by join and filter based data retrieval, pig is very efficient
* Sqoop is useful when we have data in sql tables that need to be imported in the hadoop file system.