**H1B visa dataset**

**BIG DATA - HADOOP**

AKHSHAYA J S | Professional Diploma in Digital Transformation – Big Data with Hadoop |

23.01.2018

Tools : Apache Hadoop Framework – HDFS, MapReduce, Hive, Pig, Sqoop, MySql and MS Excel for Data Visualization

S180030100216

NIIT

**H1B visa Data set Analysis**

**A PROJECT REPORT**

***Submitted by***

***AKHSHAYA J S***

***in the partial fulfillment for the award of the course***

***of***

**PROFESSIONAL DIPLOMA**

***in***

**BIG DATA WITH HADOOP**

**NIIT, CHENNAI**

**JANUARY 2018**

**ACKNOWLEDGEMENT**

I find immense pleasure to convey my sincere and grateful thanks to **NIIT** and the management for providing necessary facilities in carrying out this project.

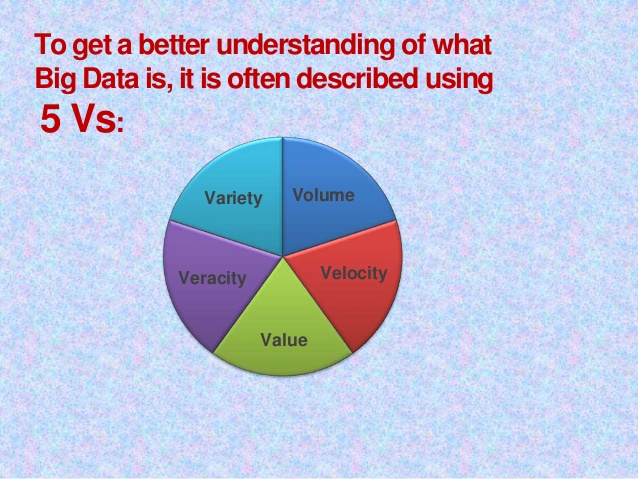
I greatly indebted to my Tech Mentor **Ms. Amirtha,** the batch instructor **Mr. Annu** and the SLT faculty **Mr. Sandeep** for constant support throughout the course and also for useful suggestions, constant encouragement and kind advice in bringing out this project as a success.

I express my regards and sincere thanks to the Academic Leader **Ms. Aishwarya** for providing all necessary resource to complete the project. I extend my thanks to all staff members of NIIT for their kind co-operation for the completion of the project successfully. I am grateful to thank my family and all my friends for their valuable feedback, encouragement and suggestions.

**1 Introduction**

**1.1 What is Big Data?**

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.

****

* **Volume**
  + The quantity of generated and stored data. The size of the data determines the value, whether it can actually be considered big data or not.
* **Variety**
  + The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.
* **Velocity**
  + The speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.
* **Value**
  + Here we are referring to the worthiness of the data being extracted.  Having endless amounts of data is one thing, but unless it can be turned into value it is useless.
* **Veracity**
  + It is the uncertainty of data.
  + The data quality of captured data can vary greatly, affecting the accurate analysis.

**1.2 Advantages of BigData :**

1. Cost reduction
2. Time reduction
3. Efficient.

**1.3 Some of the components of BigData Processing :**

1. Hadoop
2. HDFS
3. MapReduce
4. Hive
5. Pig
6. Sqoop

**2 Apache Hadoop**

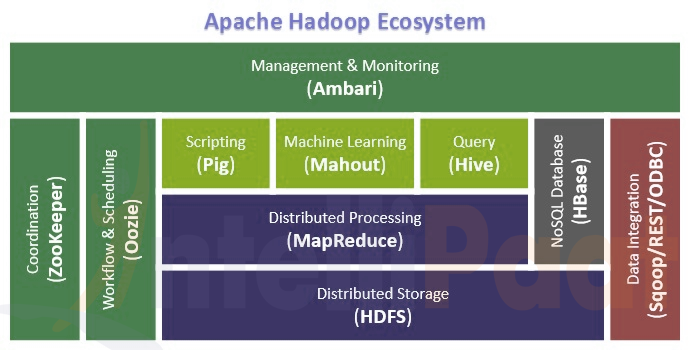
**Apache Hadoop** is the most important framework for working with Big Data. Hadoop biggest strength is scalability. It upgrades from working on a single node to thousands of nodes without any issue in a seamless manner.

The different domains of Big Data means we are able to manage the data’s are  from videos, text medium, transactional data, sensor information, statistical data, social media conversations, search engine queries, ecommerce data, financial information, weather data, news updates, forum discussions, executive reports, and so on

Google’s **Doug Cutting** and his team members developed an Open Source Project namely known as HADOOP which allows you to handle the very large amount of data. Hadoop runs the applications on the basis of MapReduce where the [data is processed in parallel](https://intellipaat.com/blog/working-with-hadoop-cluster-set-up/) and accomplish the entire statistical analysis on large amount of data.

It is a framework which is based on [java programming](https://intellipaat.com/java-training/). It is intended to work upon from a single server to thousands of machines each offering local computation and storage. It supports the large collection of data set in a distributed computing environment.

The Apache Hadoop software library based framework that gives permissions to distribute huge amount of data sets processing across clusters of computers using easy programming models.

[](https://cdn.intellipaat.com/mediaFiles/2015/07/Apache-Hadoop-Ecosystem.png)

**2.1 The Apache Hadoop Module :**

Hadoop Common: Includes the common utilities which supports the other Hadoop modules

[HDFS](https://intellipaat.com/interview-question/hdfs-interview-questions/): Hadoop Distributed File System provides unrestricted, high-speed access to the data application.

Hadoop YARN: This technology is basically used for scheduling of job and efficient management of the cluster resource.

[MapReduce](https://intellipaat.com/tutorial/mapreduce-tutorial/): This is a highly efficient methodology for parallel processing of huge volumes of data.

1. **Hadoop HDFS (Storage layer) –**[Hadoop Distributed File System](https://intellipaat.com/tutorial/hadoop-tutorial/hdfs-overview/)or[HDFS](https://intellipaat.com/tutorial/hadoop-tutorial/hdfs-overview/) is based on the Google File System (GFS) which provides a distributed file system that is especially designed to run on commodity hardware. It reduces the faults or errors and helps incorporate low-cost hardware. It gives high level processing throughput access to application data and is suitable for [applications with large datasets](https://intellipaat.com/blog/big-data-analytics-tools-performance-testing/).
2. **Hadoop Mapreduce (Processing/Computation layer) –**[MapReduce is a parallel programming model](https://intellipaat.com/tutorial/mapreduce-tutorial/introduction-of-mapreduce/) mainly used for writing large amount of data distribution applications devised from Google for efficient processing of large amounts of datasets, on large group of clusters.
3. Hadoop YARN –[Hadoop YARN is a framework](https://intellipaat.com/tutorial/big-data-and-hadoop-tutorial/hadoop-yarn-technology/)  used for job scheduling and cluster resource management.
4. Hadoop Common –This includes [Java libraries](https://intellipaat.com/tutorial/java-tutorial/introduction-java/) and utilities which provide those java files which are essential to start Hadoop.
5. Task Tracker –It is a node which is used to accept the tasks such as shuffle and Mapreduce form job tracker.
6. Job Tracker –It is a service provider which runs [Mapreduce jobs](https://intellipaat.com/jobs/jobs-in-big-data/) on cluster.
7. Name Node –It is a node where Hadoop stores all file location information(data stored location) in Hadoop distributed file system.
8. Data Node – The data is stored in the Hadoop distributed file system.

Big Data are categorized into:

1. Structured –which stores the data in rows and columns like relational data sets
2. Unstructured – here data cannot be stored in rows and columns like video, images, etc.
3. Semi-structured – data in format XML are readable by machines and human

There is a standardized methodology that Big Data follows highlighting usage [methodology of ETL](https://intellipaat.com/interview-question/etl-interview-questions/).

ETL – stands for Extract, Transform, and Load.

Extract –fetching the data from multiple sources

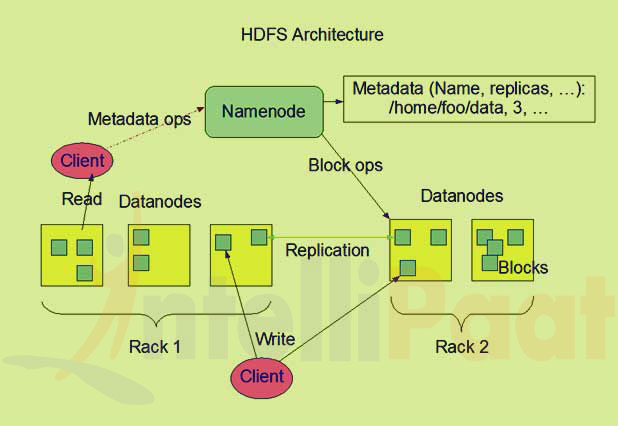
Transform – convert the existing data to fit into the analytical needs

Load –right systems to derive value in it.

**2.2 Namenode and DataNode**

The namenode is the commodity hardware that contains the GNU/Linux operating system and its library file setup and the namenode software. The [system containing namenode](https://intellipaat.com/tutorial/big-data-and-hadoop-tutorial/hadoop-multi-node-clusters/) acts as the master server and carries out following tasks:

* Manage namespace for file system.
* Provides client’s access to files.
* Execution of file system operations such as rename, open and close files and directories.
* There are a number of Data Nodes consists of one per node in the cluster, which helps to manage storage attached to the nodes that they run on.
* HDFS exposes a file system namespace and provides access to user data to be stored within files.
* Basically a file is split into one or more blocks and these blocks are stored in a set of DataNodes.
* The NameNode executes file system namespace operations containing opening, closing, and renaming files and directories which determine the mapping of blocks to DataNodes which is responsible for providing read and write requests from the file system’s clients.
* The DataNodes also perform functions such as block creation, deletion, and replication upon instruction from the NameNode.

[](https://cdn.intellipaat.com/mediaFiles/2015/07/Namenode-and-DataNode.jpg)

* These machines typically run a GNU/Linux operating system (OS).
* As we know HDFS is designed and implemented based on Java language; any machine that supports Java application can run the NameNode or the DataNode software.
* HDFS can be deployed on a wide range of machines as the usage of it is highly portable in Java language. A particular deployment has a dedicated machine that runs only the NameNode software. The other machines in the cluster run one instance of the DataNode software only.
* The architecture does not predicate based on running multiple DataNodes on the same machine but in a real deployment that is rarely happened.

The single NameNode in a cluster efficiently simplifies the architecture of the system. The NameNode is the main arbitrator and repository for all HDFS metadata sets. The system is designed in such a way that user data never flows through the NameNode.

## 2.3 Block

Data is mainly stored in HDFS’s file. These files are segregated into one or more segments and further stored in individual data node. These file segments are namely known as block. The default block size is 64MB which can be modified as per the requirements from HDFS configuration.

HDFS blocks are huge compared to disk blocks and the main reason is cost reduction. By making a particular set of block large enough hence here the time consumed to transfer the data from the disk can be made larger than the time to seek from the beginning of the block. Thus the time consumed to transfer a large file made of multiple blocks operates at the disk transfer rate.

# 2.4 MapReduce and Yarn

Mapreduce is mainly a data processing component of Hadoop. It is a programming model for processing large number of data sets. It contains the task of data processing and distributes the particular tasks across the nodes. It consists of two phases –

* Map
* Reduce

Map converts a typical dataset into another set of data where individual elements are divided into key/value pairs.

Reduce task takes the output files from a map considering as an input and then integrate the data tuples into a smaller set of tuples. Always it is been executed after the map job is done.

Features of Mapreduce are as follows:

* Framework is provided for Mapreduce execution
* Abstracts developer from the complexity of distributed programming languages.
* Partial failure of the processing cluster is expected and tolerable to fulfill the requirements.
* In-built Redundancy and fault tolerance is available.
* Mapreduce programming model system is language independent.
* Automatic parallelization and distribution are in charge.
* Fault tolerance
* Enable data local processing
* Shared nothing than architectural model
* Manages all the inter process communication
* Parallelly managing the distributed servers which are running across the various tasks.
* Managing all communications and data transfers between the various part of system module.
* Redundancy and failures are provided for overall management of the whole process.

Mapreduce simple steps follow:

1. Executes map function on each input is received
2. Map function emits key, value pair
3. Shuffle, Sort and Group the outputs
4. Executes the reduce function on the group
5. Emits the output results is given per group basis.

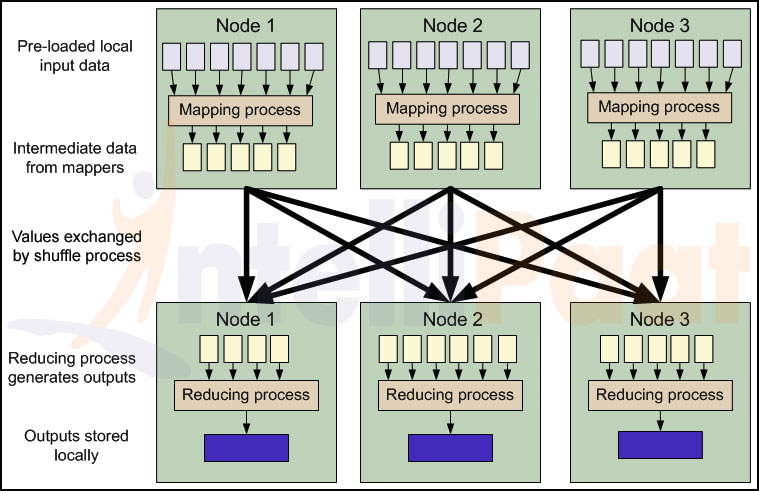
Map Function

Mainly operates on each key/value pair of data and then transforms the data based on the transformation logic provided in the map function. Map function always produces a key/value pair as output result.

Map (key1, value1) ->List (key2, value2)

Reduce Function

It takes list of value for each and every key transforms the data based on the (aggregation) logic provided in the reduce function.



MapReduce is the framework that is used for processing large amounts of data on commodity hardware on a huge dataset of cluster ecosystem. The MapReduce is a powerful method of processing data when there are large amounts of node connected to the cluster. The two important tasks of the MapReduce algorithm are: Map and Reduce.

The main motto of the Map task is to take a large set of data and convert it into another set of data which is broken down into tuples(rows) or Key/Value pairs. Later the Reduce task takes the tuple which is the form of an output of the Map task and makes the input for a reduction task. Here the data tuples are converted into a very smaller set of tuples. The Reduce task always follows as per the Map task.

The biggest strength of the MapReduce framework is its scalability. Once a MapReduce program is written then it can be easily extrapolated to work over a cluster which has hundreds or even thousands of nodes within it. In this framework, actually computation is sent to where the data resides.

## Hadoop Map Reduce – Key Features & Highlights

## Terminology

**PayLoad–**These are the applications that are implemented for the Map and Reduce functions.

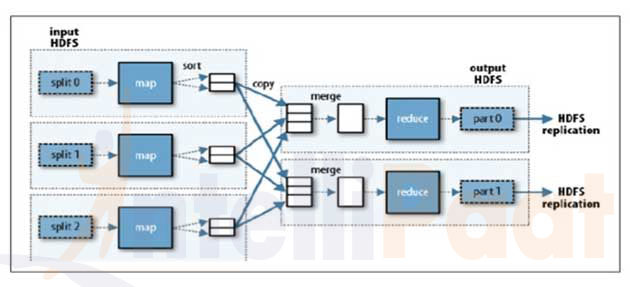
**Mapper–** This application helps to maps the input key/value pairs to a set of intermediate key/value pair.

**NamedNode–** This node manages the HDFS.

**DataNode–** DataNode is used where data is presented in a before any processing takes place.

**MasterNode–** MasterNode is used where JobTracker runs and receives job requests from clients.

**SlaveNode–** Map and Reduce program run particularly in this node.

[](https://cdn.intellipaat.com/mediaFiles/2015/07/Hadoop-Map-Reduce-–-Key-Features-Highlights.jpg)

**JobTracker–** This schedules the jobs and tracks the assigns the jobs to Task tracker.

**Task Tracker**– the Task Tracker status is reported to JobTracker after the task is being tracked.

**Job–** It is an execution process of a Mapper and Reducer.

**Task–** Task of an execution of a Mapper or a called as Reducer on a slice of data.

**Task Attempt–** This is an attempt to execute a task on a SlaveNode.

**2.5 Hadoop YARN Technology**

Yarn full form stands for yet another resource negotiator. It is a cluster management technology which is an open source platform distributed for processing framework. The main objective of YARN is to construct a framework on Hadoop that allows the cluster resources to be allocated to the specified applications and consider MapReduce has one of these applications.

It separates each tasks of the job tracker into separate entities. The job tracker maintains track of both job scheduling which matches the tasks with task tracker and another one is task progress monitoring that take care of tasks and starts again the failed or slower tasks and doing the task bookkeeping like as maintaining counter totals.

It divides these two roles into two independent daemons that are a mainly the resource manager which manages the usage of resources across the cluster and an application master which manage the lifecycle of applications running on the cluster.

Application master agrees with the resource manager for the sake of cluster resources which is expressed in terms of a number of containers each with a certain memory limit then runs application specific processes in these containers.

The containers are handled by node managers which are running on cluster nodes which ensure the application does not use more resources which are allocated to it.

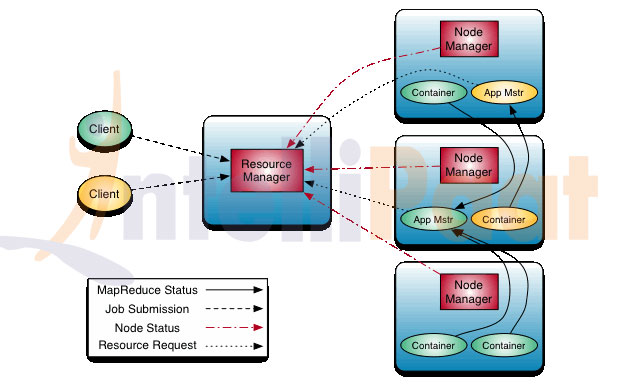
It is a very efficient technology which manages the Hadoop cluster. YARN is one of the parts of Hadoop 2 version under the aegis of the Apache Software Foundation.

YARN is has developed a completely new and innovative way of processing data and is now rightly at the center of The Hadoop architecture. Using this technology now it is possible to stream real-time, uses new interactive SQL, process data using multiple engines, manages the data using batch processing on a single platform and so on.

Map Reduce on YARN

MapReduce on YARN includes more entities compared to the classic MapReduce. They are:

* Client –Client submits the MapReduce job.

[](https://cdn.intellipaat.com/mediaFiles/2015/07/Hadoop-Yarn-Technology.jpg)

* YARN resource manager – This manages the allocation of compute resources based on the cluster.
* YARN node managers – It launches and monitors the compute containers on machines based on the cluster.
* Map Reduce application master – It manages and arranges the tasks running the MapReduce job. The application master and the MapReduce application tasks run correspondingly in the containers which are scheduled by the resource manager and managed by the node managers.
* Distributed file system (Normally HDFS) – It shares the job files created between the other entities.

How the YARN technology works?

* YARN technology lets Hadoop provides the enterprise level solutions, helping organizations achieve better resource management. It is the main platform for getting consistent solutions, high level of security and governing of data over the complete spectrum of the Hadoop cluster.
* There are various technologies that resides within the data center can also benefit from YARN. This procedure is possible to process and have linear-scale storage in a very cost effective way. Using YARN helps to come with applications that can access data and run in a Hadoop ecosystem on a consistent framework.

Some of the features of YARN

* High degree of compatibility: The applications created are using the Map Reduce framework which can easily run on YARN.
* Better cluster utilization: YARN allocates all the cluster resources in an efficient and dynamic manner and which leads to utilizes it in much better way compared to previous version of Hadoop.
* Utmost scalability: As and when the required number of nodes in the Hadoop cluster expands, the YARN Resource Manager ensures that it meets the user requirements and processing power of the data center does not face any problems in solving.
* Multi-tenancy: Various engines that access data on the Hadoop cluster can efficiently works all thanks goes to YARN being a highly versatile technology.

Key components of YARN

YARN came into existence because there was an urgent need to separate the two distinct tasks that go on in a Hadoop ecosystem and which are known as TaskTracker and the JobTracker entities. So consider the below mentioned key components of the YARN technology.

* Global Resource Manager
* Application Master per application
* Node Manager per node slave
* Container per application that runs on a Node Manager

Thus the Node Manager and the Resource Manager became the main reason on which the new distributed application works. The various resources manager are allocated to the system applications using the power of the Resource Manager. Application Master works along with the Node Manager and also works on specific framework to get resources from the Resource Manager to manage the various task components.

A scheduler works with the RM(Resource Manager) framework for the right allocation of resources and ensuring all the constraints of the user limit and queue capacities are adhered are provided at all times. As per the requirements of each application the scheduler will provide the right resource.

The Application Master works on basis of coordination with the scheduler in order to get the right resource containers keep an eye on the status and also keep tracking the progress of the process.

The Node Manager manages the application containers and launches it when it is required, tracks down the uses of the resources like the memory, processor, network and the disk utilization and gives the entire detailed report to the Resource Manager.

# 3 Apache Hive

Hive is open source platform mainly used for same purpose. These tools that ease the complexity of writing difficult/complexed programs of java based MapReduce. Hive is like a data warehouse that uses the MapReduce for the purpose of analyzing data stored on HDFS. It provides a query language called HiveQL that is familiar to the Structured Query Language (SQL) standard. It is developed based on facebook concepts. Hive was created who are posing strong analysts having strong SQL skills but few java programming skills are required to run queries on the large volumes of data that Face book stored in HDFS. Apache Pig and Hive are two projects that are consider as the top most layer of Hadoop and provide a higher-level language for using MapReduce library of Hadoop management.

**3.1 Why hive?**

It consists of a query language based on the standard SQL instead of giving a rapid development of map and reduces tasks. Hive takes HiveQL statements and then automatically transforms each and every query into one or more MapReduce jobs. Later it runs the overall MapReduce program and executes the output to the user whereas Hadoop streaming decreases the mandatory code, compile, and submit cycle. Hive removes it completely instead requires only the composition of HiveQL statements.

This interface to Hadoop not only accelerates the time required to produce results from data analysis but also it significantly expands for whom this Hadoop and MapReduce are helpful.

What makes Hive Hadoop popular?

* The users are provided with strong and powerful statistics functions.
* It is similar to SQL and hence it is very easy to understand the concepts.
* It can be combined with the HBase for querying the data in HBase. This kind of feature is not available in pig. Pig function named HbaseStorage () is mainly used for loading the data from HBase.
* Supported by Hue.
* Various user groups are considered such as CNET, Last.fm, Facebook, and Digg etc.

Example for Hive query -

hive>select \* form employee;  
 hive> describe employee;

* The Apache Hive is mainly data warehouse software which allows you to read, write and manage huge number volumes of datasets stored in a distributed environment using SQL. It is possible to project structure onto data that is termed as storage. Users can be connected to Hive using a JDBC driver and a command line tool.
* Hive is an open Source platform system. Use Hive for analyzing and querying in large number of datasets consisting the Hadoop files. It’s similar to the SQL programming. The current version of Hive is 0.13.1.
* Hive supports ACID transaction: Atomicity, Consistency, Isolation, and Durability. ACID transactions are provided at the row levels, those are Insert, Delete, and Update options so that Hive supports ACID transaction.
* Hive is not considered as a complete database. The design rules and regulations of Hadoop and HDFS put restrictions on what Hive can do in the field of programming.

Hive is most suitable for following data warehouse applications

* Analyzing the static data
* Less Responsive time
* No rapid changes in datasets.

Hive doesn’t provide fundamental features required for OLTP (Online Transaction Processing). Hive is proper usage for data warehouse applications in large data sets.

The two types of tables in Hive

1. Managed table
2. External table

We can change the settings within Hive session, using the command known as SET. It is used to change Hive job settings for a query to gain the exact results.

# 4 Apache Pig

Pig raises the level of abstraction for processing large amount of datasets. It is a fundamental platform for analyzing large amount of data sets which consists of a high level language for expressing data analysis programs. It is an open source platform developed by yahoo.

Advantages of Pig

* Reusing the code
* Faster development
* Less number of lines of code
* Schema and type checking etc

Pig is made up of two pieces:

* First is the language which allows to express data flows known as Pig Latin.
* Second one is execution environment created to run Pig Latin programs. There are now presently two environments that are local execution in a single JVM and distributed execution on the basis of Hadoop cluster.

A Pig Latin program is huge collection of series of operations or transformations which are implemented to the input data files to generate output. These operations express a data flow that the pig execution environment transforms into an executable representation and then runs it accurately.

What makes Pig Hadoop popular?

* Easy to learn read and write and implement if you know SQL.
* It implements a new approach of multi query.
* Provides a large number of nested data types such as Maps, Tuples and Bags which are not easily available in MapReduce along with some other data operations like Filters, Ordering and Joins.
* It consist of different user groups for instance up to 90% of Yahoo’s MapReduce is done by Pig and up to 80% of Twitter’s MapReduce is also done by Pig and various other companies like Sales force, LinkedIn and Nokia etc are majoritively using the Pig.

The Apache Pig is a platform for managing large sets of data which consists of high-level programming to analyze the data as per the requirements assigned. Pig mainly consists of the infrastructure to evaluate the complexity of the program. The advantages of Pig programming is that it can easily handle parallel processes correspondingly managing a very large number of data. The programming on this platform is done by using the textual language Pig Latin.

Pig Latin comes with the following features:

* Simple programming: it is easy to code, execute and manage the program.
* Better optimization: system can automatically optimize the execution as per the requirement raised.
* Extensive nature: Used to achieve highly specific processing tasks.

Pig can be used for following purposes:

* ETL data pipeline
* Research on raw data
* Iterative processing.

The scalar data types in pig are in the form of int, float, double, long, chararray, and byte array. The complex data types in Pig are namely the map, tuple, and bag.

Map: The data element consisting the data type chararray where element has pig data type include complex data type

Example- [city’#’bang’,’pin’#560001]

In this city and pin are data element mapping the values here.

Tuple: Collection of data types and it has defined fixed length. It consists of multiple fields and those are ordered in sequence.

Bag: It is a huge collection of tuples ,unordered sequence , tuples arranged in the bag are separated by comma.

Example: {(‘Bangalore’, 560001),(‘Mysore’,570001),(‘Mumbai’,400001)

LOAD function: Load function helps to load the data from the file system. It is a known as a relational operator. In the first step in data-flow language it is required to mention the input, which is completed by using the keyword named as ‘load’.  
The LOAD syntax is

LOAD ‘mydata’ [USING function] [AS schema];

Example- A = LOAD ‘intellipaat.txt’;

A = LOAD ‘intellipaat.txt’ USINGPigStorage(‘\t’);

The relational operations in Pig segmentation is as follows:

foreach, order by, filters, group, distinct, join, limit.

foreach: Takes a set of expressions and applies them to almost all the records in the data pipeline to next operator.

A =LOAD ‘input’ as (emp\_name: charrarray, emp\_id: long, emp\_add : chararray, phone : chararray, preferences : map []);

B = foreach A generate emp\_name, emp\_id;

Filters: It contains a predicate and it provides us to select which records will be retained in our data pipeline permanently.

Syntax: alias = FILTER alias BY expression;

Otherwise it indicates the name of the relation, By indicates required keyword and the expression containing Boolean.

Example: M = FILTER N BY F5 == 4;

Difference between hive and pig

|  |  |
| --- | --- |
| Hive | Pig |
| Used for Data Analysis | Used for Data and Programs |
| Used as Structured Data | Pig is Semi-Structured Data |
| Hive has HiveQL | Pig has Latin |
| Hive is used for creating reports | Pig is used for programming |
| Hive works on the server side | Pig works on the client side |
| Hive does not support avro | Pig supports Avro |

## **Fundamentals of Apache Sqoop**

### ****5 Apache Sqoop****

### ****5.1 What is Sqoop?****

Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and external datastores such as relational databases, enterprise data warehouses.

Sqoop is used to import data from external datastores into Hadoop Distributed File System or related Hadoop eco-systems like Hive and HBase. Similarly, Sqoop can also be used to extract data from Hadoop or its eco-systems and export it to external datastores such as relational databases, enterprise data warehouses. Sqoop works with relational databases such as Teradata, Netezza, Oracle, MySQL, Postgres etc.

### ****5.2 Why is Sqoop used?****

For Hadoop developers, the interesting work starts after data is loaded into HDFS. Developers play around the data in order to find the magical insights concealed in that Big Data. For this, the data residing in the relational database management systems need to be transferred to HDFS, play around the data and might need to transfer back to relational database management systems. In reality of Big Data world, Developers feel the transferring of data between relational database systems and HDFS is not that interesting, tedious but too seldom required. Developers can always write custom scripts to transfer data in and out of Hadoop, but Apache Sqoop provides an alternative.

Sqoop automates most of the process, depends on the database to describe the schema of the data to be imported. Sqoop uses MapReduce framework to import and export the data, which provides parallel mechanism as well as fault tolerance. Sqoop makes developers life easy by providing command line interface. Developers just need to provide basic information like source, destination and database authentication details in the sqoop command. Sqoop takes care of remaining part.

Sqoop provides many salient features like:

1. Full Load
2. Incremental Load
3. Parallel import/export
4. Import results of SQL query
5. Compression
6. Connectors for all major RDBMS Databases
7. Kerberos Security Integration
8. Load data directly into Hive/Hbase
9. Support for Accumulo

Sqoop is Robust, has great community support and contributions. Sqoop is widely used in most of the Big Data companies to transfer data between relational databases and Hadoop.

### **Where is Sqoop used?**

Relational database systems are widely used to interact with the traditional business applications. So, relational database systems has become one of the sources that generate Big Data.

As we are dealing with Big Data, Hadoop stores and processes the Big Data using different processing frameworks like MapReduce, Hive, HBase, Cassandra, Pig etc and storage frameworks like HDFS to achieve benefit of distributed computing and distributed storage. In order to store and analyze the Big Data from relational databases, Data need to be transferred between database systems and Hadoop Distributed File System (HDFS). Here, Sqoop comes into picture. Sqoop acts like a intermediate layer between Hadoop and relational database systems.  You can import data and export data between relational database systems and Hadoop and its eco-systems directly using sqoop.



Sqoop provides command line interface to the end users. Sqoop can also be accessed using Java APIs. Sqoop command submitted by the end user is parsed by Sqoop and launches Hadoop Map only job to import or export data because Reduce phase is required only when aggregations are needed. Sqoop just imports and exports the data; it does not do any aggregations.

Sqoop parses the arguments provided in the command line and prepares the Map job. Map job launch multiple mappers depends on the number defined by user in the command line. For Sqoop import, each mapper task will be assigned with part of data to be imported based on key defined in the command line. Sqoop distributes the input data among the mappers equally to get high performance. Then each mapper creates connection with the database using JDBC and fetches the part of data assigned by Sqoop and writes it into HDFS or Hive or HBase based on the option provided in the command line.

## **Basic Commands and Syntax for Sqoop**

### **Sqoop-Import**

Sqoop import command imports a table from an RDBMS to HDFS. Each record from a table is considered as a separate record in HDFS. Records can be stored as text files, or in binary representation as Avro or SequenceFiles.

Generic Syntax:

$ sqoop import (generic args) (import args)

$ sqoop-import (generic args) (import args)

The Hadoop specific generic arguments must precede any import arguments, and the import arguments can be of any order.

### **Importing a Table into HDFS**

Syntax:

$ sqoop import --connect --table --username --password --target-dir

--connect        Takes JDBC url and connects to database  
--table             Source table name to be imported  
--username    Username to connect to database  
--password     Password of the connecting user  
--target-dir     Imports data to the specified directory

### **Importing Selected Data from Table**

Syntax:

$ sqoop import --connect --table --username --password --columns --where

--columns       Selects subset of columns  
--where           Retrieves the data which satisfies the condition

### **Importing Data from Query**

Syntax:

$ sqoop import --connect --table --username --password --query

--query           Executes the SQL query provided and imports the results

### **Incremental Exports**

Syntax:

$ sqoop import --connect --table --username --password --incremental --check-column --last-value

Sqoop import supports two types of incremental imports:

1. Append
2. Lastmodified.

Append mode is to be used when new rows are continually being added with increasing values. Column should also be specified which is continually increasing with --check-column. Sqoop imports rows whose value is greater than the one specified with --last-value. Lastmodified mode is to be used when records of the table might be updated, and each such update will set the current timestamp value to a last-modified column. Records whose check column timestamp is more recent than the timestamp specified with --last-value are imported.

Notes:

1. In JDBC connection string, database host shouldn't be used as “localhost” as Sqoop launches mappers on multiple data nodes and the mapper will not able to connect to DB host.
2. “–password” parameter is insecure as any one can read it from command line. –P option can be used, which prompts for password in console. Otherwise, it is recommended to use –password-file pointing to the file containing password (Make sure you have revoked permission to unauthorized users).

Few arguments helpful with Sqoop import:

| Argument | Description |
| --- | --- |
| --num-mappers,-m | Mappers to Launch |
| --fields-terminated-by | Field Separator |
| --lines-terminated-by | End of line seprator |

### **Importing Data into Hive**

Below mentioned Hive arguments is used with the sqoop import command to directly load data into Hive:

| Argument | Description |
| --- | --- |
| --hive-home | Override $HIVE\_HOME path |
| --hive-import | Import tables into Hive |
| --hive-overwrite | Overwrites existing Hive table data |
| --create-hive-table | Creates Hive table and fails if that table already exists |
| --hive-table | Sets the Hive table name to import |
| --hive-drop-import-delims | Drops delimiters like\n, \r, and \01 from string fields |
| --hive-delims-replacement | Replaces delimiters like \n, \r, and \01 from string fields with user defined delimiters |
| --hive-partition-key | Sets the Hive partition key |
| --hive-partition-value | Sets the Hive partition value |
| --map-column-hive | Overrides default mapping from SQL type datatypes to Hive datatypes |

Syntax:

$ sqoop import --connect --table --username --password --hive-import --hive-table

Specifying --hive-import, Sqoop imports data into Hive table rather than HDFS directory.

### **Importing Data into HBase**

Below mentioned HBase arguments is used with the sqoop import command to directly load data into HBase:

| Argument | Description |
| --- | --- |
| --column-family | Sets column family for the import |
| --hbase-create-table | If specified, creates missing HBase tables and fails if already exists |
| --hbase-row-key | Specifies which column to use as the row key |
| --hbase-table | Imports to Hbase table |

Syntax:

$ sqoop import --connect --table --username --password --hbase-table

Specifying –hbase-table, Sqoop will import data into HBase rather than HDFS directory.

### **Sqoop-Import-all-Tables**

The import-all-tables imports all tables in a RDBMS database to HDFS. Data from each table is stored in a separate directory in HDFS. Following conditions must be met in order to use sqoop-import-all-tables:

1. Each table should have a single-column primary key.

2. You should import all columns of each table.

3. You should not use splitting column, and should not check any conditions using where clause.

Generic Syntax:

$ sqoop import-all-tables (generic args) (import args)

$ sqoop-import-all-tables (generic args) (import args)

Sqoop specific arguments are similar with sqoop-import tool, but few options like --table, --split-by, --columns, and --where arguments are invalid.

Syntax:

$ sqoop-import-all-tables ---connect --username --password

### **Sqoop-Export**

Sqoop export command exports a set of files in a HDFS directory back to RDBMS tables. The target table should already exist in the database.

Generic Syntax:

$ sqoop export (generic args) (export args)

$ sqoop-export (generic args) (export args)

Sqoop export command prepares INSERT statements with set of input data then hits the database. It is for exporting new records, If the table has unique value constant with primary key, export job fails as the insert statement fails. If you have updates,  you can use --update-key option. Then Sqoop prepares UPDATE statement which updates the existing row, not the INSERT statements as earlier.

Syntax:

$ sqoop-export ---connect --username --password --export-dir

### **Sqoop-Job**

Sqoop job command allows us to create a job. Job remembers the parameters used to create job, so they can be invoked any time with same arguments.

Generic Syntax:

$ sqoop job (generic args) (job args) [-- [subtool name] (subtool args)]

$ sqoop-job (generic args) (job args) [-- [subtool name] (subtool args)]

Sqoop-job makes work easy when we are using incremental import. The last value imported is stored in the job configuration of the sqoop-job, so for the next execution it directly uses from configuration and imports the data.

Sqoop-job options:

| Argument | Description |
| --- | --- |
| --create | Defines a new job with the specified job-id (name). Actual sqoop import command should be seperated by “--“ |
| --delete | Deletes a saved job. |
| --exec | Executes the saved job. |
| --show | Show the save job configuration |
| --list | Lists all the saved jobs |

Syntax:

$ sqoop job --create -- import --connect --table

### **Sqoop-Codegen**

Sqoop-codegen command generates Java class files which encapsulate and interpret imported records. The Java definition of a record is initiated as part of the import process. For example, if Java source is lost, it can be recreated. New versions of a class can be created which use different delimiters between fields, and so on.

Generic Syntax:

$ sqoop codegen (generic args) (codegen args)

$ sqoop-codegen (generic args) (codegen args)

Syntax:

$ sqoop codegen --connect --table

|  |
| --- |
|  |

### **Sqoop-Eval**

Sqoop-eval command allows users to quickly run simple SQL queries against a database and the results are printed on to the console. Generic Syntax:

$ sqoop eval (generic args) (eval args)

$ sqoop-eval (generic args) (eval args)

Syntax:

$ sqoop eval --connect --query "SQL query"

Using this, users can be sure that they are importing the data as expected.

### **Sqoop-List-Database**

Used to list all the database available on RDBMS server. Generic Syntax:

$ sqoop list-databases (generic args) (list databases args)

$ sqoop-list-databases (generic args) (list databases args)

Syntax:

$ sqoop list-databases --connect

### **Sqoop-List-Tables**

Used to list all the tables in a specified database. Generic Syntax:

$ sqoop list-tables (generic args) (list tables args)

$ sqoop-list-tables (generic args) (list tables args)

**6 PROJECT**

**About H1b Visa 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 dataset description is as follows:

The columns in the dataset include:

* CASE\_STATUS: Status associated with the last significant event or decision. Valid values include “Certified,” “Certified-Withdrawn,” Denied,” and “Withdrawn”.

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

Certified Withdrawn: LCA was approved but later withdrawn by employer

Withdrawn: LCA was withdrawn by employer before approval

Denied: LCA was denied by DOL

* EMPLOYER\_NAME: Name of employer submitting labour condition application.
* SOC\_NAME: the Occupational name associated with the SOC\_CODE. SOC\_CODE is the occupational code associated with the job being requested for temporary labour condition, as classified by the Standard Occupational Classification (SOC) System.
* JOB\_TITLE: Title of the job
* FULL\_TIME\_POSITION: Y = Full Time Position; N = Part Time Position
* PREVAILING\_WAGE: Prevailing Wage for the job being requested for temporary labour condition. The wage is listed at annual scale in USD. The prevailing wage for a job position is defined as the average wage paid to similarly employed workers in the requested occupation in the area of intended employment. The prevailing wage is based on the employer’s minimum requirements for the position.
* YEAR: Year in which the H1B visa petition was filed
* WORKSITE: City and State information of the foreign worker’s intended area of employment
* lon: longitude of the Worksite
* lat: latitude of the Worksite

**6.1 Cleansing Data**

Step 1 - A table h1b\_applications is created with the following columns s\_no, case\_status, employer\_name, soc\_name, job\_title, full\_time\_position, prevalling\_wage, year, worksite, longitute, latitude. SERDE functions are used to separate the values with “”.

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;

Step 2 - The data is loaded into the table from local file system.

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

h1b\_applications;

Step 3 - Table h1b\_app2 is created with field delimiter “\t”. Data is loaded from h1b\_applications table where field separators “,” is replaced by “\t”.

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

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

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

double, latitute double )

row format delimited

fields terminated by '\t'

STORED AS TEXTFILE;

Step 4 - A condition is declared to ignore records with NA in the case status.

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";

Step 5 - A new table h1b\_final is created and datas from the h1b\_app2 table are loaded but the case status column is filtered only for CERTIFIED, CERTIFIED-WITHDRAWN,DENIED AND WITHDRAWN. This table provides us the final cleansed dataset for which all the analysis has to be done.

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;

**6.2 Shell Script**

#!/bin/bash

show\_menu()

{

NORMAL=`echo "\033[m"`

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

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

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

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

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

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

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

echo -e "${MENU}\*\*${NUMBER} 2) ${MENU} Qus 1b ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 3) ${MENU} Qus 2a ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 4) ${MENU} Qus 2b ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 5) ${MENU} Qus 3 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 6) ${MENU} Qus 4 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 7) ${MENU} Qus 5a ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 8) ${MENU} Qus 5b ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 9) ${MENU} Qus 6 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 10)${MENU} Qus 7 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 11)${MENU} Qus 8a ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 12)${MENU} Qus 8b ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 13)${MENU} Qus 9 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 14)${MENU} Qus 10 ${NORMAL}"

echo -e "${MENU}\*\*${NUMBER} 15)${MENU} Qus 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/1\_a.sh

show\_menu;

;;

2) clear;

option\_picked "1b) Find top 5 job titles who are having highest avg growth in applications.";

pig /home/hduser/Qns1bquery.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/2\_a.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?";

pig /home/hduser/Project/Qns3query.pig

show\_menu;

;;

6) clear;

option\_picked "4)Which top 5 employers file the most petitions each year?";

bash /home/hduser/Desktop/4.sh

show\_menu;

;;

7) clear;

option\_picked "5) Find the most popular top 10 job positions for H1B visa applications for each year?a) for all the applications";

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/Qns6query.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 "8a) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order"-'[Certified and certified-withdrawn]';

echo "Enter the year"

read year

echo "you have selected ${year}"

hive -e "select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'Y' and year=$year group by job\_title, full\_time\_position, year order by average desc;"

show\_menu;

;;

12) clear;

option\_picked "8b) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order"-'[Certified and certified-withdrawn]';

echo "Enter the year"

read year

echo "you have selected ${year}"

hive -e "select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position ='Y' and year=$year group by job\_title, full\_time\_position, year order by average desc;"

show\_menu;

;;

13) 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/Qns9query.pig

show\_menu;

;;

14) 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/Qns10query.pig

show\_menu;

;;

15) 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

**6.3 Codes**

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

import java.io.\*;

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

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

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

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

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

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

public class DataEngineerJob {

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

{

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

{

try{

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

if(str[4].equals("DATA ENGINEER"))

context.write(new Text(str[7]),new Text (str[4]));

}

catch(Exception e)

{

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

}

}

}

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

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

int count=0;

for (Text val : values)

{

{ count++;

}

}

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

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

}

}

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

Configuration conf = new Configuration();

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

//conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");

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

job.setJarByClass(DataEngineerJob.class);

job.setMapperClass(MapClass.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

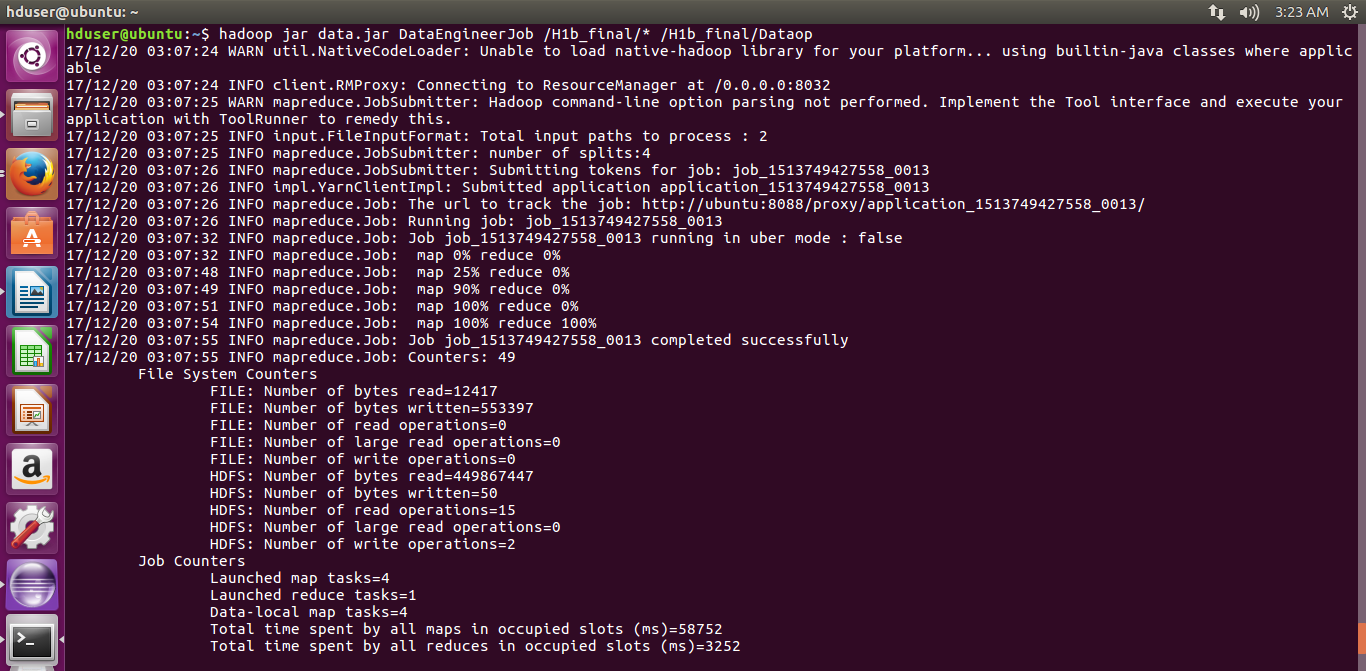
FileInputFormat.addInputPath(job, new Path(args[0]));

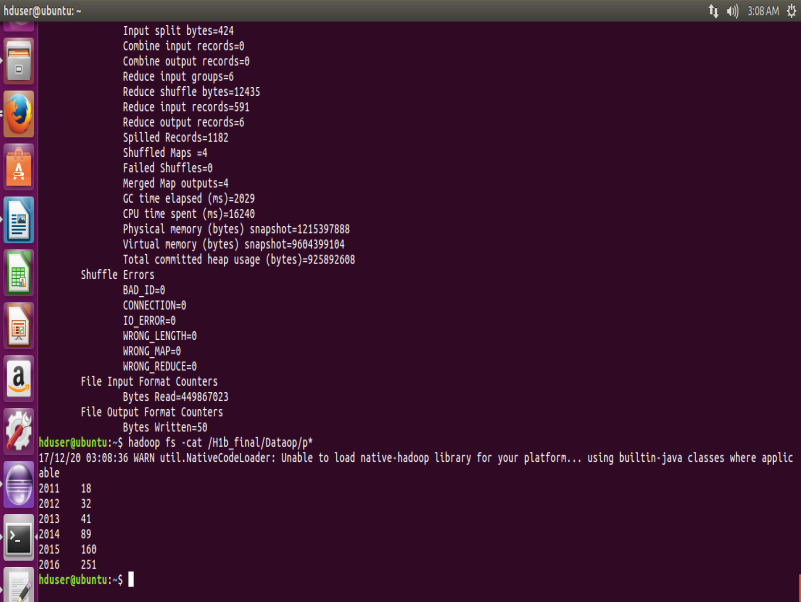
FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}

}





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

A = load '/user/hive/warehouse/h1b\_final/\*' using PigStorage('\t') AS (s\_no, case\_status:chararray, employer\_name, soc\_name:chararray, job\_title:chararray, full\_time\_position, prevailing\_wage, year, worksite, longitute, latitute);

--dump A;

A\_f1 = filter A by year=='2011';

--dump A\_f1;

A\_g1 = group A\_f1 by $4;

--dump A\_g1;

A\_gp1 = foreach A\_g1 generate group, COUNT(A\_f1.$1);

--dump A\_gp1;

A\_f2 = filter A by year=='2012';

--dump A\_f2;

A\_g2 = group A\_f2 by $4;

A\_gp2 = foreach A\_g2 generate group, COUNT(A\_f2.$1);

A\_f3 = filter A by year=='2013';

--dump A\_f3;

A\_g3 = group A\_f3 by $4;

A\_gp3 = foreach A\_g3 generate group, COUNT(A\_f3.$1);

A\_f4 = filter A by year=='2014';

--dump A\_f4;

A\_g4 = group A\_f4 by $4;

A\_gp4 = foreach A\_g4 generate group, COUNT(A\_f4.$1);

A\_f5 = filter A by year=='2015';

--dump A\_f5;

A\_g5 = group A\_f5 by $4;

A\_gp5 = foreach A\_g5 generate group, COUNT(A\_f5.$1);

A\_f6 = filter A by year=='2016';

--dump A\_f6;

A\_g6 = group A\_f6 by $4;

A\_gp6 = foreach A\_g6 generate group, COUNT(A\_f6.$1);

A\_j = join A\_gp1 by $0 ,A\_gp2 by $0 ,A\_gp3 by $0 ,A\_gp4 by $0 ,A\_gp5 by $0 ,A\_gp6 by $0;

--dump A\_j;

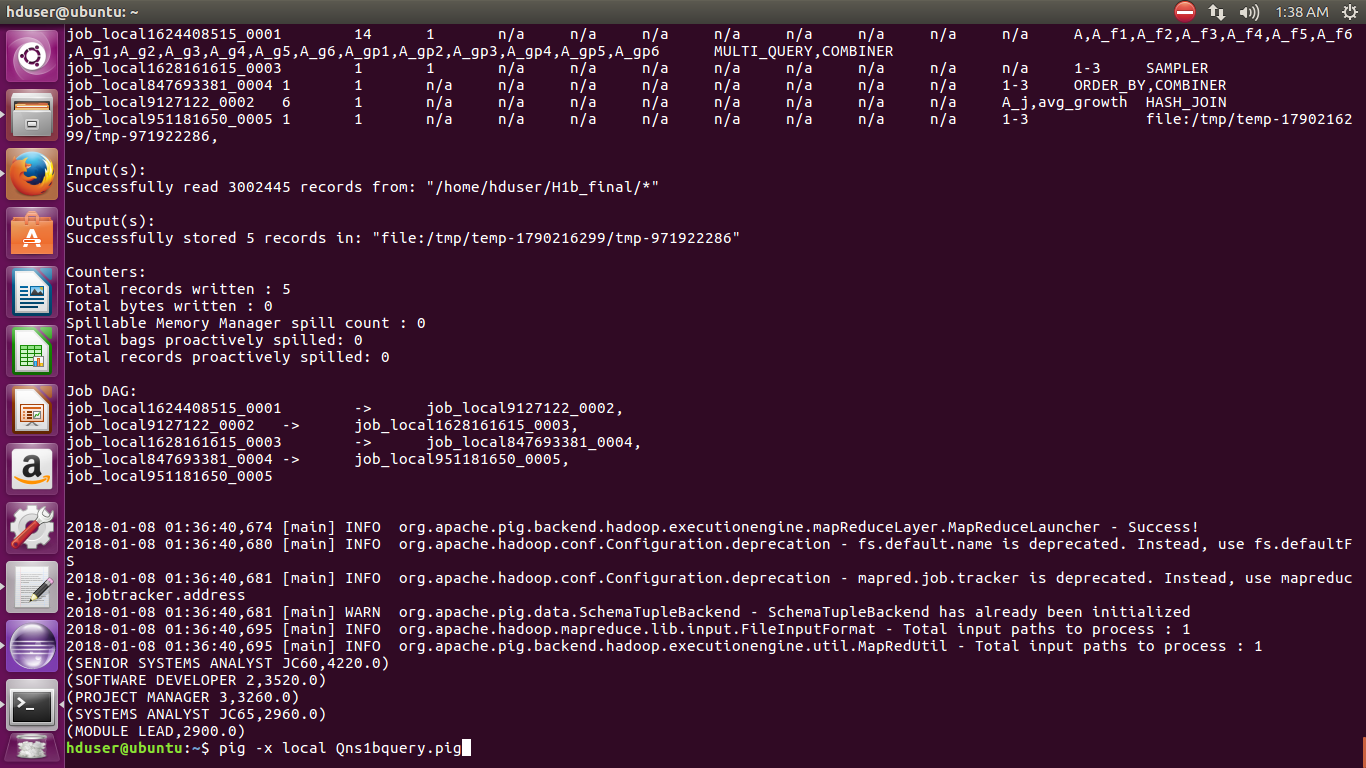
A\_j1 = foreach A\_j generate $0 , $1 , $3 ,$5 ,$7 ,$9 ,$11;

A\_growth = foreach A\_j1 generate $0 , (float)(($2-$1)/$1)\*100, (float)(($3-$2)/$2)\*100,(float)(($4-$3)/$3)\*100, (float)(($5-$4)/$4)\*100, (float)(($6-$5)/$5)\*100;

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

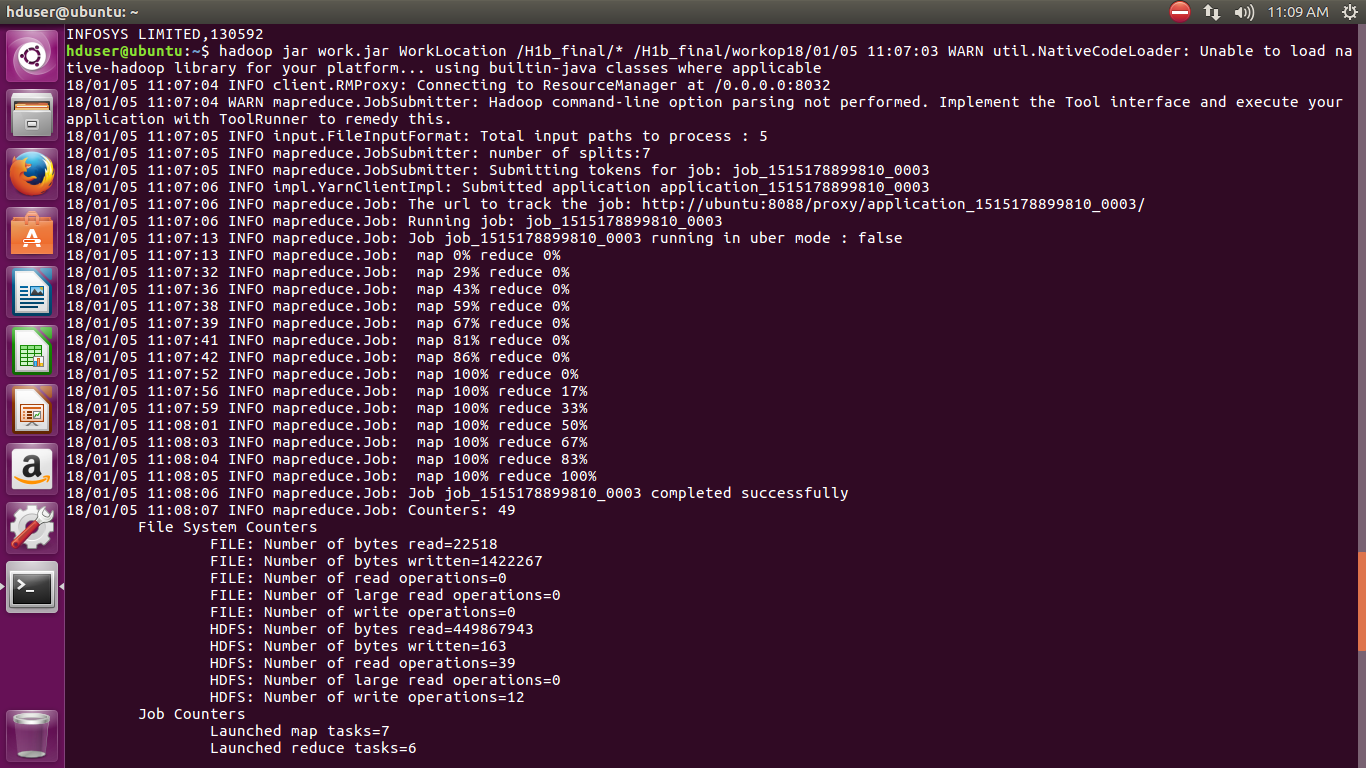
A\_lmt = limit (order avg\_growth by $1 desc) 5;

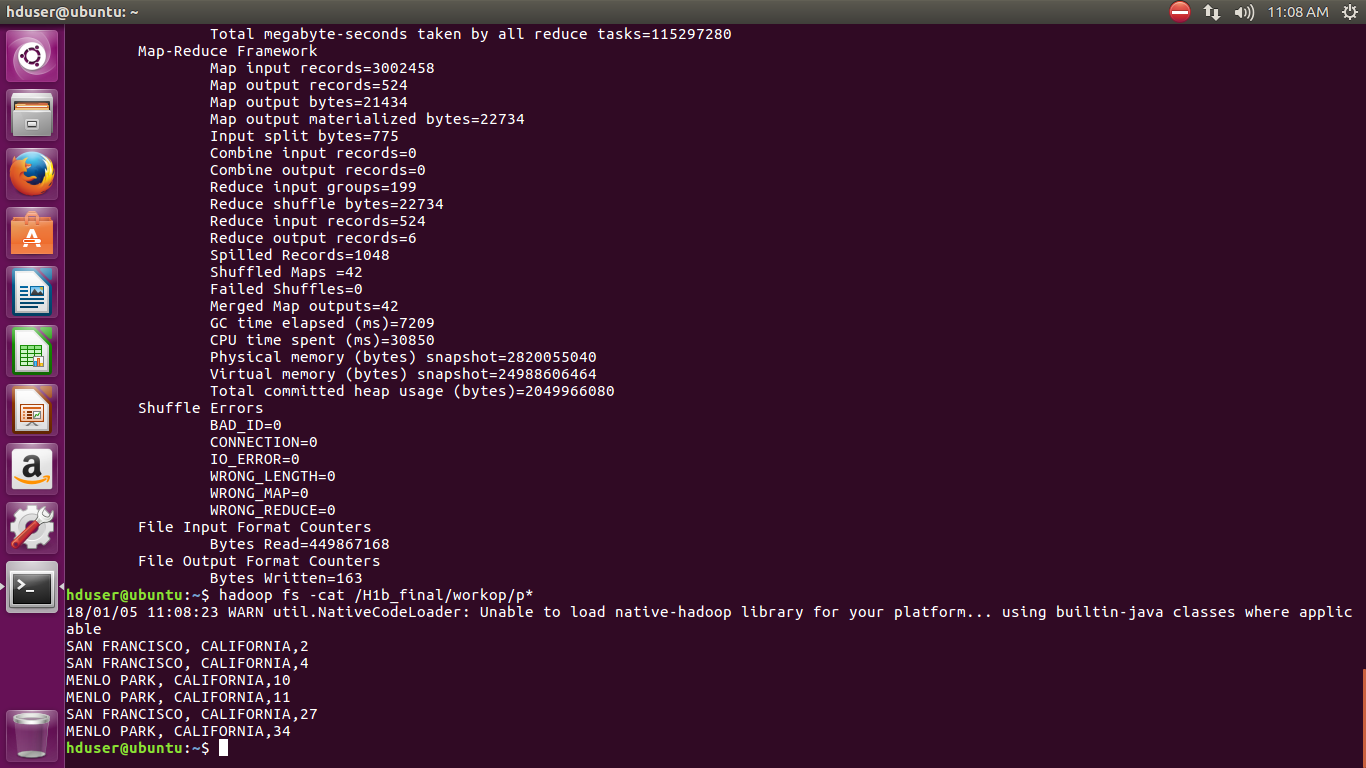
dump A\_lmt;



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

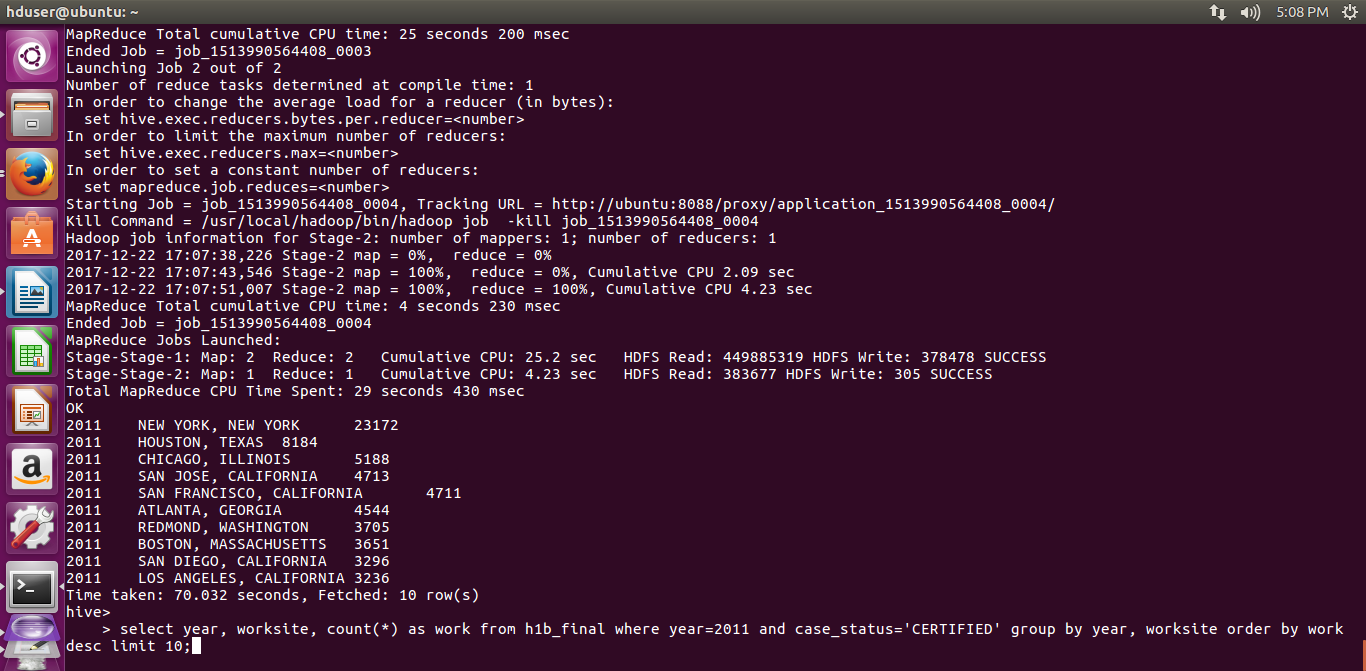
import java.io.IOException;  
import java.util.TreeMap;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Job;  
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 WorkLocation {  
          
  public static class Top5Mapper extends  
        Mapper<LongWritable, Text, Text, Text> {  
            public void map(LongWritable key, Text value, Context context  
            ) throws IOException, InterruptedException {  
                try {  
                String[] str = value.toString().split("\t");  
                 if(str[4].equals("DATA ENGINEER") && str[1].equals("CERTIFIED"))  
                    {  
                     String abc= str[4]+"\t"+str[7];                      
                     context.write(new Text(str[8]),new Text (abc));  
                    }  
                      
             }  
             catch(Exception e)  
             {  
                System.out.println(e.getMessage());  
             }  
          }  
       }  
      
public static class YearPartitioner extends  
   Partitioner < Text, Text >  
   {  
      public int getPartition(Text key, Text value, int numReduceTasks)  
      {  
         String[] str = value.toString().split("\t");  
         int year = Integer.parseInt(str[1]);  
  
         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 Top5Reducer 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 a="";  
            for(Text val:values)  
            {  
                String[] str = val.toString().split("\t");  
                  
                    count++;  
                    a = str[1]+"\t"+key+"\t"+str[0];  
                  
                  
            }  
            String myValue = a+"\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();  
    //conf.set("name", "value")  
    //conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");  
    Job job = Job.getInstance(conf, "Count");  
    job.setJarByClass(WorkLocation.class);  
    job.setMapperClass(Top5Mapper.class);  
    job.setPartitionerClass(YearPartitioner.class);  
    job.setReducerClass(Top5Reducer.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);  
  }  
}





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

1. select year, worksite, count(\*) as work from H1b\_final where year=2011 and case\_status='CERTIFIED' group by year, worksite order by work desc limit 10;
2. select year, worksite, count(\*) as work from H1b\_final where year=2012 and case\_status='CERTIFIED' group by year, worksite order by work desc limit 10;
3. select year, worksite, count(\*) as work from H1b\_final where year=2013 and case\_status='CERTIFIED' group by year, worksite order by work desc limit 10;
4. select year, worksite, count(\*) as work from H1b\_final where year=2014 and case\_status='CERTIFIED' group by year, worksite order by work desc limit 10;
5. select year, worksite, count(\*) as work from H1b\_final where year=2015 and case\_status='CERTIFIED' group by year, worksite order by work desc limit 10;
6. select year, worksite, count(\*) as work from H1b\_final where year=2016 and case\_status='CERTIFIED' group by year, worksite order by work desc limit 10;



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

A = LOAD '/user/hive/warehouse/h1b\_final' USING PigStorage('\t') AS (s\_no, case\_status:chararray, employer\_name, soc\_name:chararray, job\_title:chararray, full\_time\_position, prevailing\_wage, year, worksite, longitute, latitute);

--dump A;

A\_f = filter A by case\_status=='CERTIFIED' and job\_title=='DATA SCIENTIST';

--dump A\_f;

A\_g = GROUP A\_f by $3;

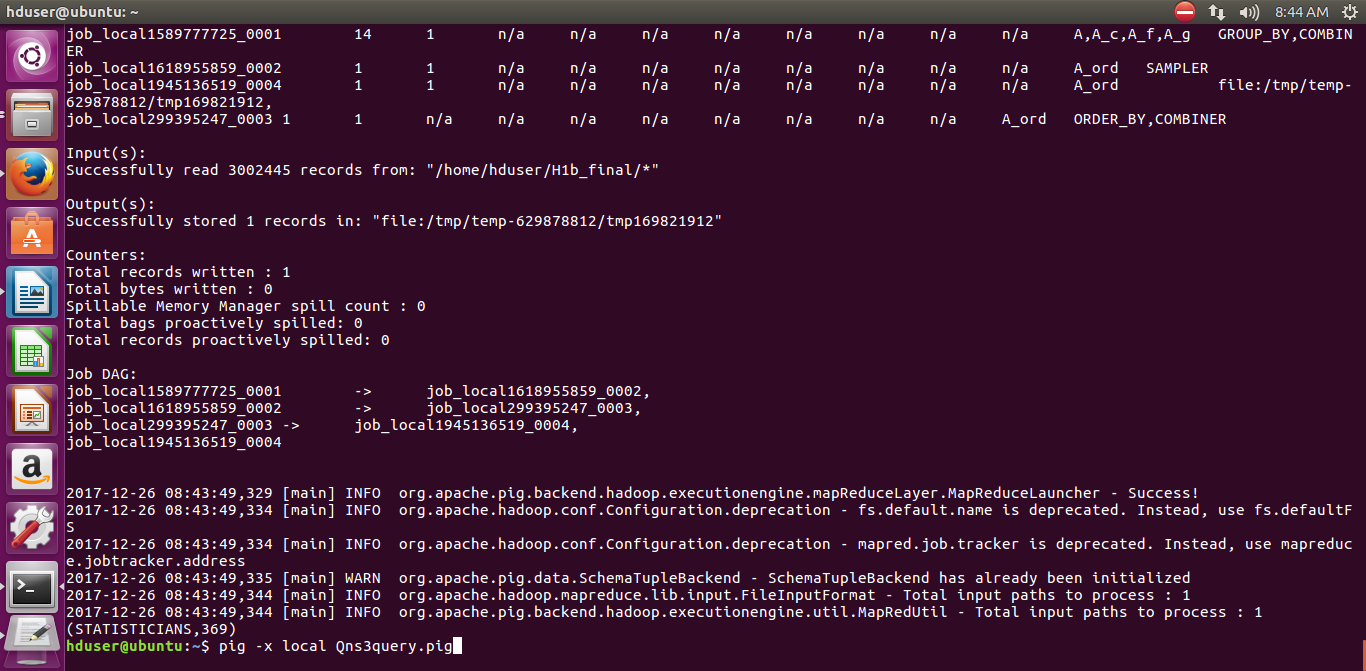
A\_c = foreach A\_g generate $0, COUNT(A\_f.$4);

--dump A\_c;

A\_ord = ORDER A\_c BY $1 DESC;

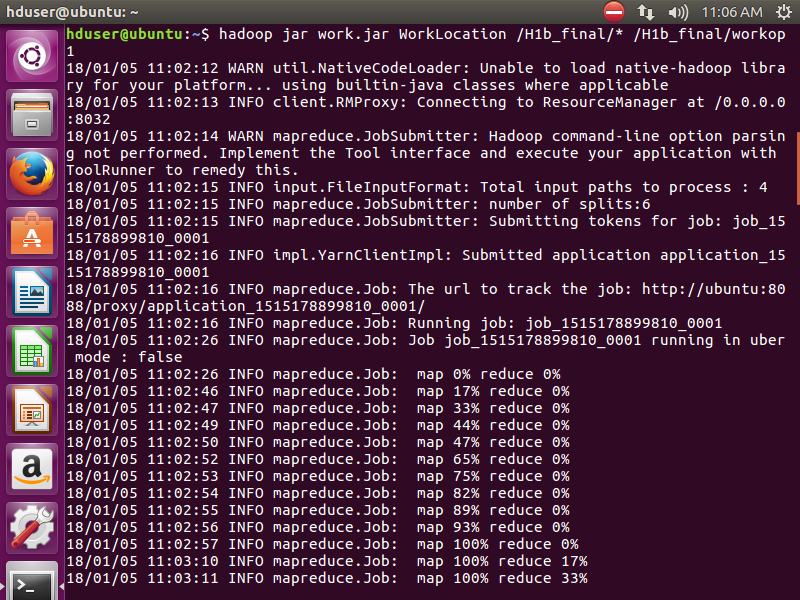
A\_lmt = LIMIT A\_ord 1;

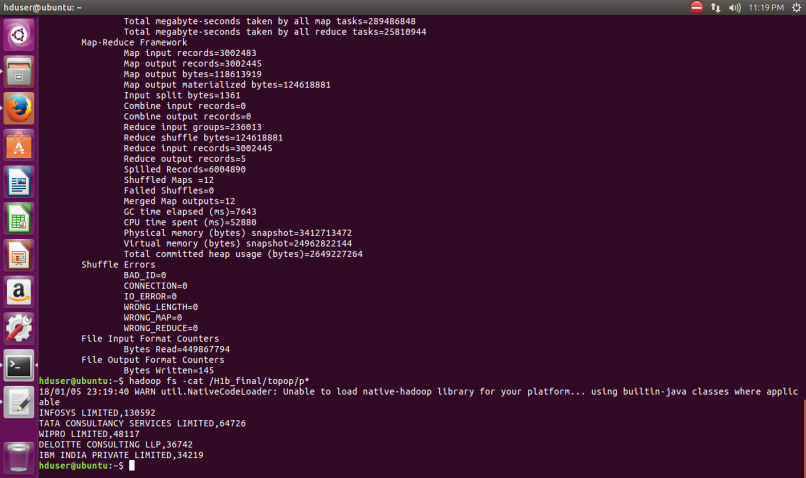
Dump A\_lmt;



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

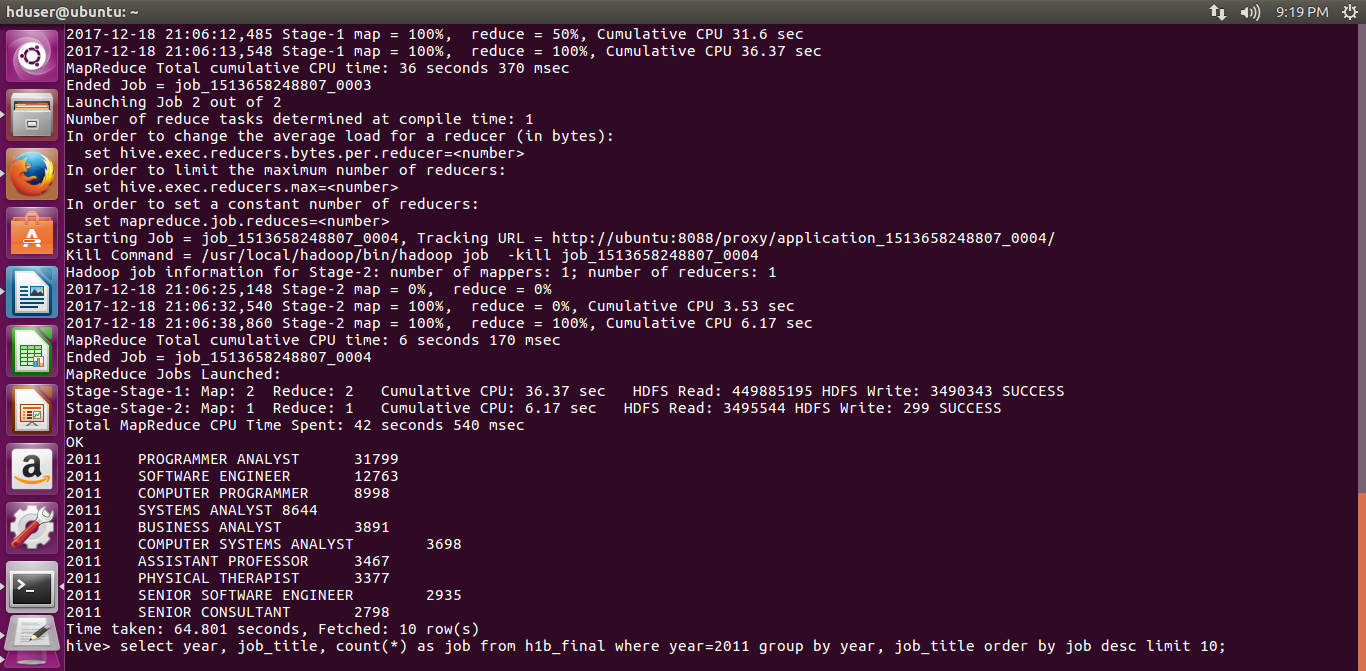
import java.io.IOException;  
import java.util.TreeMap;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.Mapper;  
import org.apache.hadoop.mapreduce.Mapper.Context;  
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 Top5Employeer {  
          
  public static class Top5Mapper extends  
        Mapper<LongWritable, Text, Text, Text> {  
            public void map(LongWritable key, Text value, Context context  
            ) throws IOException, InterruptedException {  
                try {  
                String[] str = value.toString().split("\t");  
                String abc= str[1]+"\t"+str[7];  
                context.write(new Text(str[2]),new Text(abc));  
                     
             }  
             catch(Exception e)  
             {  
                System.out.println(e.getMessage());  
             }  
          }  
       }  
      
public static class YearPartitioner extends  
   Partitioner < Text, Text >  
   {  
      public int getPartition(Text key, Text value, int numReduceTasks)  
      {  
         String[] str = value.toString().split("\t");  
         int year = Integer.parseInt(str[1]);  
  
         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 Top5Reducer 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 a="";  
            for(Text val:values)  
            {  
                String[] str = val.toString().split("\t");  
                  
                    count++;  
                    a = str[1]+"\t"+key+"\t"+str[0];  
                  
                  
            }  
            String myValue = a+"\t"+count;  
            tm.put(new Long(count), new Text(myValue));  
            if(tm.size()>5)  
            {  
                tm.remove(tm.firstKey());  
            }  
       }  
        public void cleanup(Context con) throws IOException, InterruptedException  
        {  
            for(Text t:tm.descendingMap().values())  
            {  
                con.write(NullWritable.get(), t);  
            }  
        }  
}  
  
              
        public static void main(String[] args) throws Exception {  
              
            Configuration conf = new Configuration();  
            Job job = Job.getInstance(conf, "Top 5 Employeer per year");  
            job.setJarByClass(Top5Employeer.class);  
            job.setMapperClass(Top5Mapper.class);  
            job.setPartitionerClass(YearPartitioner.class);  
            job.setReducerClass(Top5Reducer.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);  
          }  
    }





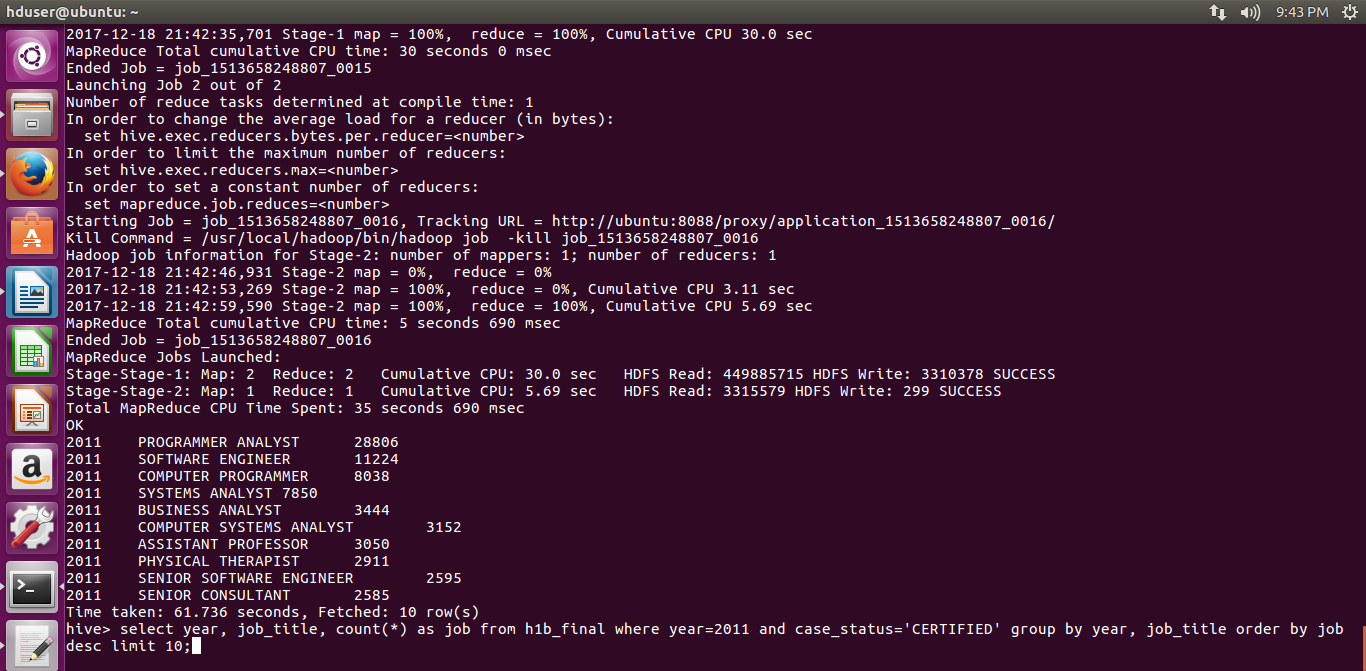
5 )a) Find the most popular top 10 job positions for H1B visa applications for each year? for all the applications

1. select year, job\_title, count(\*) as job from H1b\_final where year=2011 group by year, job\_title order by job desc limit 10;
2. select year, job\_title, count(\*) as job from H1b\_final where year=2012 group by year, job\_title order by job desc limit 10;
3. select year, job\_title, count(\*) as job from H1b\_final where year=2013 group by year, job\_title order by job desc limit 10;
4. select year, job\_title, count(\*) as job from H1b\_final where year=2014 group by year, job\_title order by job desc limit 10;
5. select year, job\_title, count(\*) as job from H1b\_final where year=2015 group by year, job\_title order by job desc limit 10;
6. select year, job\_title, count(\*) as job from H1b\_final where year=2016 group by year, job\_title order by job desc limit 10;



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

1. select year, job\_title, count(\*) as job from H1b\_final where year=2011 && case\_status=certified group by year, job\_title order by job desc limit 10;
2. select year, job\_title, count(\*) as job from H1b\_final where year=2012 && case\_status=certified group by year, job\_title order by job desc limit 10;
3. select year, job\_title, count(\*) as job from H1b\_final where year=2013 && case\_status=certified group by year, job\_title order by job desc limit 10;
4. select year, job\_title, count(\*) as job from H1b\_final where year=2014 && case\_status=certified group by year, job\_title order by job desc limit 10;
5. select year, job\_title, count(\*) as job from H1b\_final where year=2015 && case\_status=certified group by year, job\_title order by job desc limit 10;
6. select year, job\_title, count(\*) as job from H1b\_final where year=2016 && case\_status=certified group by year, job\_title order by job desc limit 10;



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.

A = load '/user/hive/warehouse/h1b\_final' using PigStorage('\t') AS (s\_no, case\_status:chararray, employer\_name, soc\_name:chararray, job\_title:chararray, full\_time\_position, prevailing\_wage, year, worksite, longitute, latitute);

--dump A;

A1 = foreach A generate year,case\_status;

--dump A1;

A1\_g = group A1 by year;

--dump A1\_g;

A1\_t = foreach A1\_g generate group, COUNT(A1.$1);

--dump A1\_t;

A1\_g1 = group A1 by (year,case\_status);

--dump A1\_g1;

A1\_c1 = foreach A1\_g1 generate group, group.year, COUNT(A1);

--dump A1\_c1;

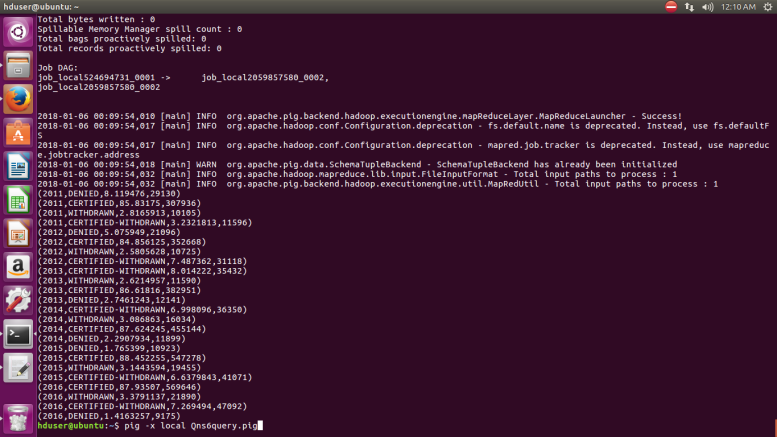
A1\_j = join A1\_c1 by $1, A1\_t by $0;

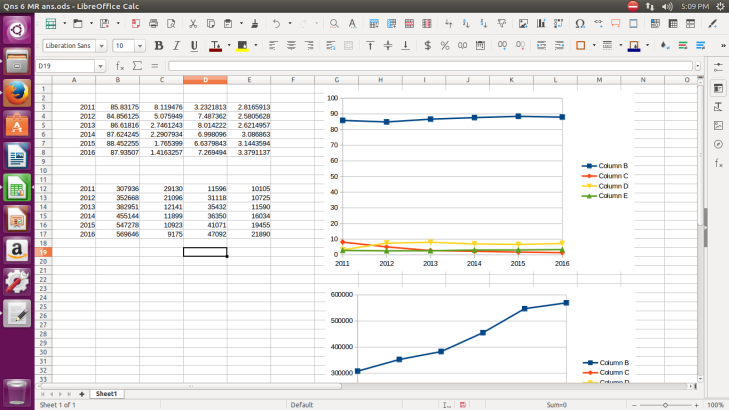
--dump A1\_j;

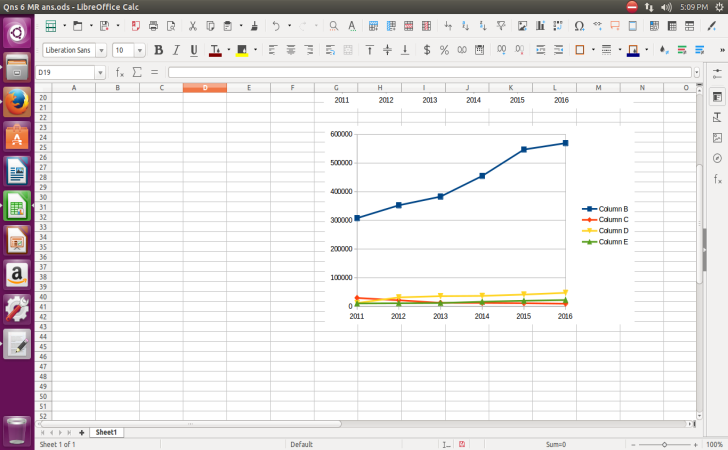
A1\_f = foreach A1\_j generate FLATTEN($0), (float)($2\*100)/$4,$2;

--dump A1\_f;

A\_op = store A1\_f into '/home/hduser/query6pig';







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

import java.io.\*;

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

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

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

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

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

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

public class NumberOfApplications {

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

{

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

{

try{

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

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

}

catch(Exception e)

{

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

}

}

}

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

{

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

int count=0;

for (Text val : values)

{

{ count++;

}

}

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

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

}

}

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

Configuration conf = new Configuration();

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

//conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");

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

job.setJarByClass(NumberOfApplications.class);

job.setMapperClass(MapClass.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

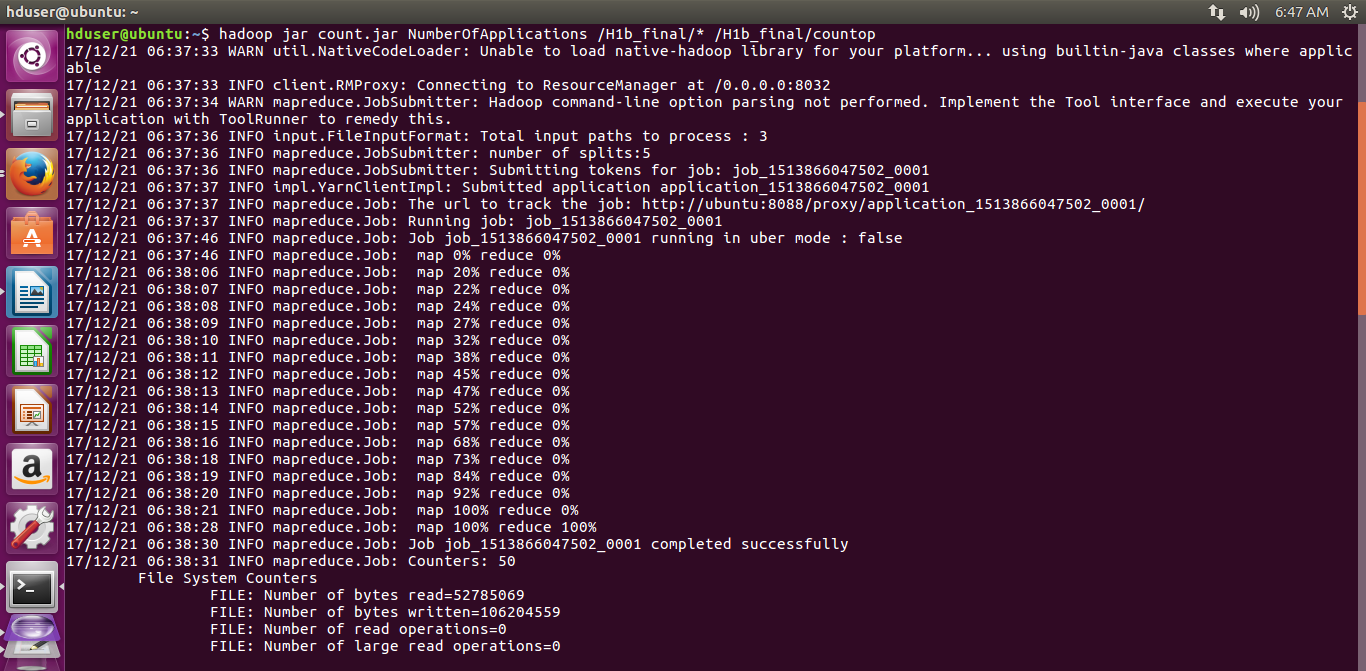
FileInputFormat.addInputPath(job, new Path(args[0]));

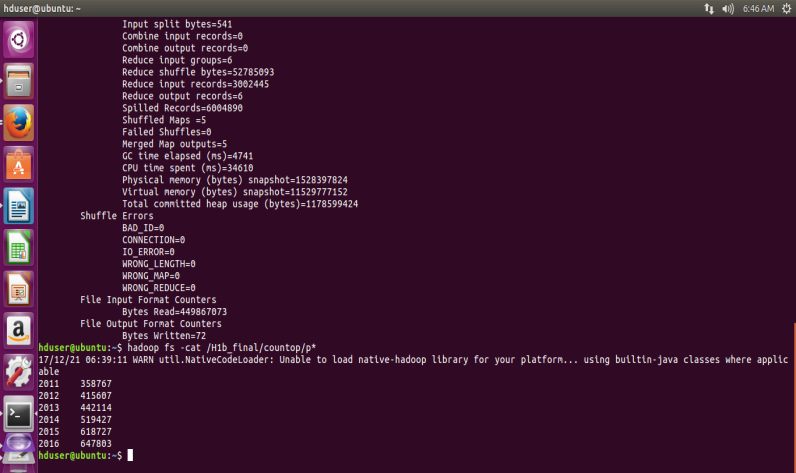
FileOutputFormat.setOutputPath(job, new Path(args[1]));

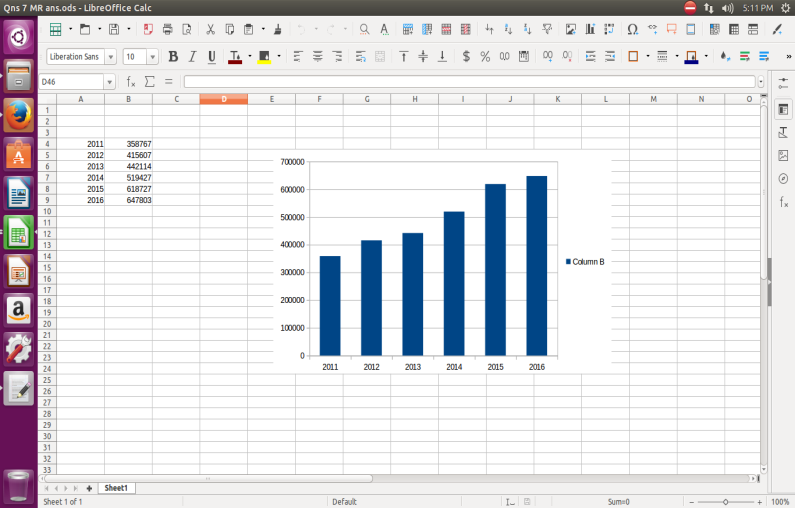
System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}



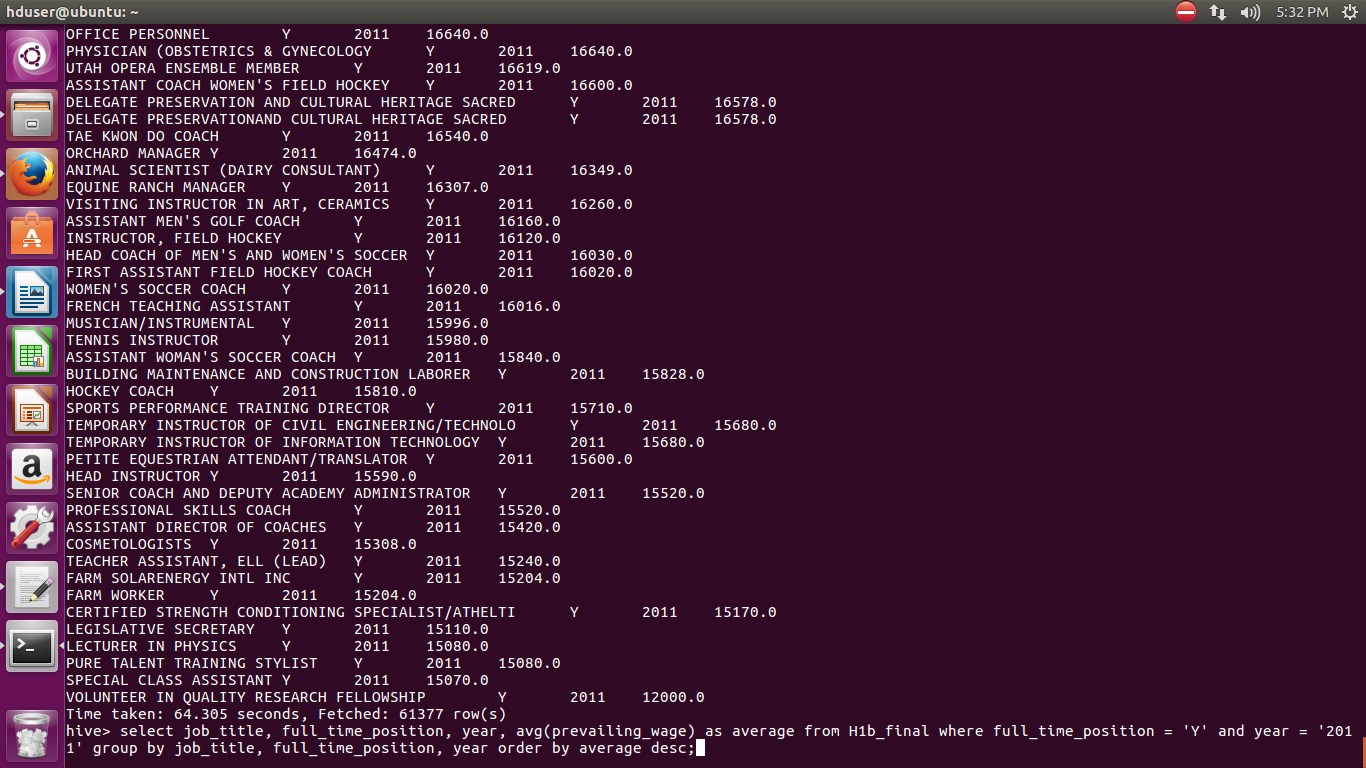




8) Find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order - [Certified and Certified Withdrawn.]

For full\_time\_position:

1. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'Y' and year = '2011' group by job\_title, full\_time\_position, year order by average desc;
2. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'Y' and year = '2012' group by job\_title, full\_time\_position, year order by average desc;
3. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'Y' and year = '2013' group by job\_title, full\_time\_position, year order by average desc;
4. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'Y' and year = '2014' group by job\_title, full\_time\_position, year order by average desc;
5. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'Y' and year = '2015' group by job\_title, full\_time\_position, year order by average desc;
6. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'Y' and year = '2016' group by job\_title, full\_time\_position, year order by average desc;



For part\_time\_position:

1. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'N' and year = '2011' group by job\_title, full\_time\_position, year order by average desc;
2. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'N' and year = '2012' group by job\_title, full\_time\_position, year order by average desc;
3. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'N' and year = '2013' group by job\_title, full\_time\_position, year order by average desc;
4. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'N' and year = '2014' group by job\_title, full\_time\_position, year order by average desc;
5. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'N' and year = '2015' group by job\_title, full\_time\_position, year order by average desc;
6. select job\_title, full\_time\_position, year, avg(prevailing\_wage) as average from H1b\_final where full\_time\_position = 'N' and year = '2016' group by job\_title, full\_time\_position, year order by average desc;



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) ?

bag1 = load '/user/hive/warehouse/h1b\_final' using PigStorage('\t') AS (s\_no, case\_status:chararray, employer\_name, soc\_name:chararray, job\_title:chararray, full\_time\_position, prevailing\_wage, year, worksite, longitute, latitute);

--dump bag1;

bag1\_g = GROUP bag1 by $2;

bag1\_c = foreach bag1\_g generate $0, COUNT(bag1.$1);

--dump bag1\_c;

bag1\_f = filter bag1 by case\_status=='CERTIFIED';

--dump bag1\_f;

bag1\_g1 = GROUP bag1\_f by $2;

bag1\_c1 = foreach bag1\_g1 generate $0, COUNT(bag1\_f.$1);

--dump bag1\_c1;

bag1\_f1 = filter bag1 by case\_status=='CERTIFIED-WITHDRAWN';

--dump bag1\_f1;

bag1\_g2 = GROUP bag1\_f1 by $2;

bag1\_c2 = foreach bag1\_g2 generate $0, COUNT(bag1\_f1.$1);

--dump bag1\_c2;

bag1\_j = join bag1\_c by $0, bag1\_c1 by $0, bag1\_c2 by $0;

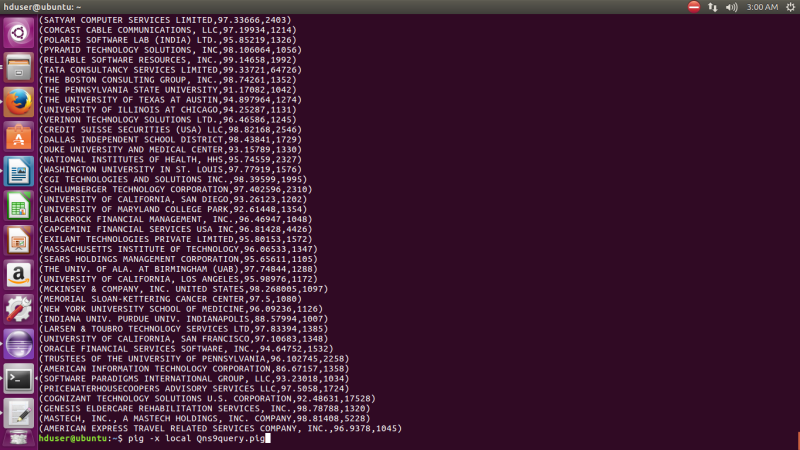
--dump bag1\_j;

bag1\_success = foreach bag1\_j generate $0, (float)($3+$5)/$1\*100 ,$1;

--dump bag1\_success;

bag1\_final = filter bag1\_success by $1>70 and $2>=1000;

dump bag1\_final;



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)?

A1 = load '/user/hive/warehouse/h1b\_final' using PigStorage('\t') AS (s\_no, case\_status:chararray, employer\_name, soc\_name:chararray, job\_title:chararray, full\_time\_position, prevailing\_wage, year, worksite, longitute, latitute);

--dump A1;

A1\_g = GROUP A1 by $4;

A1\_c = foreach A1\_g generate $0, COUNT(A1.$1);

--dump A1\_c;

A1\_f = filter A1 by case\_status=='CERTIFIED';

--dump A1\_f;

A1\_g1 = GROUP A1\_f by $4;

A1\_c1 = foreach A1\_g1 generate $0, COUNT(A1\_f.$1);

--dump A1\_c1;

A1\_f1 = filter A1 by case\_status=='CERTIFIED-WITHDRAWN';

--dump bag1\_f1;

A1\_g2 = GROUP A1\_f1 by $4;

A1\_c2 = foreach A1\_g2 generate $0, COUNT(A1\_f1.$1);

--dump A1\_c2;

A1\_j = join A1\_c by $0, A1\_c1 by $0, A1\_c2 by $0;

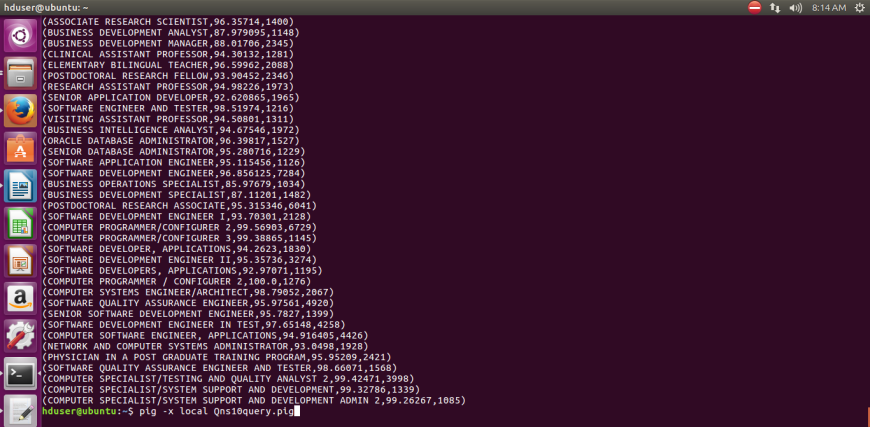
--dump bag1\_j;

A1\_success = foreach A1\_j generate $0, (float)($3+$5)/$1\*100 ,$1;

--dump A1\_success;

A1\_final = filter A1\_success by $1>70 and $2>=1000;

dump A1\_final;



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' ;